Natural Language Processing with Deep Latent Variable Models: Methods and Applications

by

Jialin Yu

Supervised by: Prof. Alexandra I. Cristea

A Thesis presented for the degree of
Doctor of Philosophy

Department of Computer Science
Durham University
United Kingdom
December 2022

Abstract

Due to their unparalleled performance and versatility, deep learning has become the de facto standard for building natural language processing (NLP) applications. Compared with conventional machine learning approaches, deep learning replaces extensive hand-engineered features in every task with end-to-end representation learning. Several concerns, however, have been raised in the research communities regarding their *robustness, trustworthiness, explainability, and interpretability.* Although these limitations of deep learning methods are widely acknowledged, work in methods and applications to alleviate these concerns in NLP is contrastingly limited. To address this research gap and explore a more robust approach for building NLP applications with deep learning, in this thesis, we studied deep latent variable models (DLVMs) in terms of *methods* (under supervised and semi-supervised learning settings) and *applications* (natural language understanding and generation) perspective for building natural language processing applications. We demonstrate the strength and benefits of DLVMs for NLP applications and discuss their effectiveness in addressing some of these concerns later in this thesis.

For contributions from a methods perspective, we studied the benefits of deep latent variable models in supervised and semi-supervised learning settings. These studies suggested that deep latent variable models are competitive in performance against standard deep learning methods; while offering additional robustness, trustworthiness, explainability and interoperability in various applications. For semi-supervised learning, particularly, we achieve state-of-the-art performance and prove the great potential of using deep latent variable models for semi-supervised learning problems.

For contributions from an applications perspective, we first presented two applications for language understanding problems, followed by two more applications for language generation problems. Our first application concerns a binary text classification task in the educational domain and pioneers the first research on how Bayesian deep learning can be applied to this text-based educational application. Our second application focuses on multilabel text classification tasks, and we present an

efficient uncertainty quantification framework as our contribution. We demonstrate the effectiveness and generalisation of this framework with diverse architectures and present the first research on using deep latent variable models for efficient uncertainty quantification purposes in multilabel text classification tasks. Our third application deals with multiple explanation generation for an explainable artificial intelligence (XAI) task, and we present a first study on how deep latent variable models can be used to generate multiple explanations in the Stanford natural language inference task. In our last application, we explore paraphrase generation tasks and present the first study of DLVMs in a semi-supervised learning setting in paraphrase generation tasks; the DLVMs can enhance paraphrase generation performance when incorporating unlabelled data in a semi-supervised manner.

The findings in this thesis are of practical value to deep learning practitioners, researchers, and engineers working on a variety of problems in the field of natural language processing and deep learning.

Declaration

The work in this thesis is based on research carried out within the Artificial Intelligence and Human Systems (AIHS) group at the Department of Computer Science, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification, and it is all the author's work unless referenced to the contrary below.

Note on Publications Included in This Thesis: At the time of submission, four chapters of this thesis are heavily based on papers submitted for publication or published in conferences and journals:

Chapter4: *Yu, J., Alrajhi, L., Harit, A., Sun, Z., Cristea, A.I. and Shi, L., 2021, June.* ***Exploring Bayesian Deep Learning for Urgent Instructor Intervention Need in MOOC Forums.*** *In International Conference on Intelligent Tutoring Systems (pp. 78-90). Springer, Cham.*

Chapter5: *Yu, J., Cristea, A.I., Harit, A., Sun, Z., Aduragba, O.T., Shi, L. and Al Moubayed, N., 2022, July.* ***Efficient Uncertainty Quantification for Multilabel Text Classification.*** *In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.*

Chapter6: *Yu, J., Cristea, A.I., Harit, A., Sun, Z., Aduragba, O.T., Shi, L. and Al Moubayed, N., 2022, July.* ***INTERACTION: A Generative XAI Framework for Natural Language Inference Explanations.*** *In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.*

Chapter7: *Yu, J., Cristea, A.I., Harit, A., Sun, Z., Aduragba, O.T., Shi, L. and Al Moubayed, N., 2022.* ***Language as a Latent Sequence: Deep Latent Variable Models for Semi-Supervised Paraphrase Generation.*** *Under review (PLOS One journal). PLOS.*

Note on Publications Not Included in This Thesis: As well as the above papers, the following works have been published during the period of research for this thesis; they have helped my deeper understanding of the work towards this thesis; however, these publications do not fit into the narrative of this thesis and have not been included in the text.

- *Aljohani, T., Yu, J. and Cristea, A.I., 2020.* ***Author profiling: prediction of learners gender on a MOOC platform based on learners comments.*** *International Journal of Computer and Information Engineering, 14(1), pp.29-36.*
- *Yu, J., Aduragba, O.T., Sun, Z., Black, S., Stewart, C., Shi, L. and Cristea, A., 2020, August.* ***Temporal Sentiment Analysis of Learners: Public Versus Private Social Media Communication Channels in a Women-in-Tech Conversion Course.*** *In 2020 15th International Conference on Computer Science & Education (ICCSE) (pp. 182-187). IEEE.*
- *Aduragba, O.T., Yu, J., Cristea, A.I., Hardey, M. and Black, S., 2020, August.* ***Digital Inclusion in Northern England: Training Women from Underrepresented Communities in Tech: A Data Analytics Case Study.*** *In 2020 15th International Conference on Computer Science & Education (ICCSE) (pp. 162-168). IEEE.*
- *Aduragba, O.T., Yu, J., Senthilnathan, G. and Crsitea, A., 2020, December.* ***Sentence contextual encoder with BERT and BiLSTM for automatic classification with imbalanced medication tweets.*** *In Proceedings of the Fifth Social Media Mining for Health Applications Workshop & Shared Task (pp. 165-167).*
- *Sun, Z., Harit, A., Yu, J., Cristea, A.I. and Shi, L., 2021, June.* ***A brief survey of deep learning approaches for learning analytics on MOOCs.*** *In International Conference on Intelligent Tutoring Systems (pp. 28-37). Springer, Cham.*
- *Sun, Z., Harit, A., Yu, J., Cristea, A.I. and Al Moubayed, N., 2021, July.* ***A Generative Bayesian Graph Attention Network for Semi-Supervised Classification on Scarce Data.*** *In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7). IEEE.*

- *Aduragba, O.T., Yu, J., Cristea, A.I. and Shi, L., 2021.* ***Detecting Fine-Grained Emotions on Social Media during Major Disease Outbreaks: Health and Well-being before and during the COVID-19 Pandemic.*** *In AMIA Annual Symposium Proceedings (Vol. 2021, p. 187). American Medical Informatics Association.*

- *Sun, Z., Harit, A., Cristea, A.I., Yu, J., Shi, L. and Al Moubayed, N., 2022, July.* ***Contrastive Learning with Heterogeneous Graph Attention Networks on Short Text Classification.*** *In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-6). IEEE.*

- *Sun, Z., Harit, A., Cristea, A.I., Yu, J. and Al Moubayed, N., 2022, December.* ***Is Unimodal Bias Always Bad for Visual Question Answering? A Medical Domain Study with Dynamic Attention.*** *In 2022 IEEE International Conference on Big Data (Big Data) (pp. 5352-5360). IEEE.*

- *Aduragba, O.T., Yu, J., Cristea, A.I. and Yang, L., 2023, April.* ***Improving Health Mention Classification Through Emphasising Literal Meanings: A Study Towards Diversity and Generalisation for Public Health Surveillance.*** *In the Web Conference 2023 (accepted). ACM.*

Acknowledgements

This thesis would not be possible without the support of many people. First, I would like to thank my esteemed supervisor, Prof. Alexandra I. Cristea, for her invaluable supervision, support, and tutelage during my PhD study over the past few years. Her continuous support and encouragement give me the strength to explore the challenges and help me through the difficulties.

My gratitude extends to my review teams, Prof. Boguslaw Obara, Dr. Noura Al Moubayed, and Dr. Tom Friedetzky, for their time and insightful comments on my yearly research progress. Special thanks to all my friends, lab mates, colleagues and co-authors, and research collaborators, especially Mr. Zhongtian Sun, Miss. Anoushka Harit, Mr. Jack Barker, Mr Olanrewaju Tahir Aduragba, Mr Brian Kostadinov Shalon Isaac-Medina, Dr. Yona Falinie Abd. Gaus and Dr. Neelanjan Bhowmik for providing me with generous support, helpful suggestions, inspiration, contributions, and directions. Special thanks also to the members of our research groups and department for all the insightful discussions and advice, especially to my co-authors: Mr. Zhongtian Sun, Miss. Anoushka Harit, Mr. Olanrewaju Tahir Aduragba, Dr. Tahani M. M. Aljohani, and Mrs. Laila Alrajhi, Dr. Noura Al Moubayed, and Prof. Alexandra I. Cristea. Additionally, special thanks to my advisers, Dr. Lei Shi and Dr. Noura Al Moubayed, for providing me with helpful suggestions and academic advice. Special thanks to my girlfriend, Miss. Meng Yan, for supporting me with food for both my thoughts and stomach. I would also like to express my gratitude to the Faculty of Science for hosting the prestigious Durham Doctoral Studentship (DDS) funding opportunity for my studies at the Department of Computer Science, Durham University. Additionally, I would like to express my gratitude to my PhD thesis examiners: Prof. Yulan He from King's College London and Prof. Effie Lai-Chong Law from Durham University, for their kinds suggestions and insightful discussions during my viva, which helped me to improve the presentation of this thesis.

Last but not least, my appreciation also goes out to my family, especially my amazing parents, Mr. Hongwen Yu and Mrs. Hongzhe Xu, for their encouragement and support through my studies over the past decays.

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

$\boldsymbol{A}$	Matrix
$\boldsymbol{a}$	Vector
a	Scalar
$\boldsymbol{\theta}$	Parameter Matrix
$\boldsymbol{\phi}$	Approximated Parameter Matrix
$\mathcal{D}$	Dataset
$\boldsymbol{X}$	Dataset Inputs i.e. $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)^\top$
$\boldsymbol{Y}$	Dataset Outputs i.e. $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_N)^\top$
$\boldsymbol{x}_n$	Input Data Point for Model
$\boldsymbol{y}_n$	Output Data Point for Model
$\hat{\boldsymbol{y}}_n$	Model Output on Input $\boldsymbol{x}_n$
$\boldsymbol{\mathcal{X}}$	Random Variable
P	Probability Distribution
$\mathbb{E}$	Expectation
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$	Multivariate Gaussian with Mean $\boldsymbol{\mu}$ and Covariance Matrix $\boldsymbol{\sigma}$

Acronyms / Abbreviations

NLP	Natural Language Processing
AI	Artificial Intelligence
BP	Back Propagation
BDL	Bayesian Deep Learning
DBL	Deep Bayesian Learning
VI	Variational Inference
NVI	Neural Variational Inference
MOOCs	Massive Open Online Courses
XAI	Explainable Artificial Intelligence
SVM	Support Vector Machine
NER	Named Entity Recognition
SRL	Semantic Role Labelling
POS	Part of Speech
OOV	Out of Vocabulary
ELMO	Embeddings from Language Models
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-Training
CNNs	Convolutional Neural Networks
RNNs	Recurrent Neural Networks
MLPs	Multi-Layer Perceptrons
LSTM	Long Short Term Memory
CRF	Conditional Random Field
LVMs	Latent Variable Models
DLVMs	Deep Latent Variable Models
PGMs	Probabilistic Graphical Models
MCMC	Monte Carlo Markov Chain

MAP	Maximum a Posteriori
CVAE	Conditional Variational AutoEncoder
VAE	Variational AutoEncoder
BNNs	Bayesian Neural Networks
ELBO	Evidence Lower Bound
KL	Kullback Leibler
i.i.d.	Independent and identically distributed
ML	Machine Learning
DL	Deep Learning
NLI	Natural Language Inference
EDM	Educational Data Mining
LA	Learning Analytics
VSAR	Variational Sequence Auto-encoding Reconstruction
DDL	Dual Directional Learning

CHAPTER 1

Introduction

1.1 Background and Motivation

Deep learning refers to a new paradigm that solves machine learning problems with deep neural network models [1, 7] and has revolutionised **natural language processing** (NLP) in recent years [8–10]. Compared with traditional machine learning, deep learning delivers powerful and superior performance in a wide range of NLP applications and has been adopted to create profitable products in industries[1][2][3]. Over the past decade, the great advancements and success of deep learning have triggered an explosion in **artificial intelligence** (AI) investment, industrial and scientific applications, hardware, startups, and education[4]. As an important AI application, NLP tasks require an understanding of complex human languages, such as sentiment analysis [11], text classification [12], machine translation [13], summarisation [14], information extraction [15] and paraphrase generation [16], are all now dominated by deep neural networks [9]. Almost every NLP task falls into one

[1] https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html
[2] https://blog.google/products/search/search-language-understanding-bert/
[3] https://ai.facebook.com/blog/heres-how-were-using-ai-to-help-detect-misinformation/
[4] https://www.stateof.ai/

of the following categories [8, 10]: natural language understanding, which assigns one or multiple labels, given associated inputs; and natural language generation, which generates a piece of text, given associated inputs. In this thesis, we explore natural language understanding problems in Chapter 4 and 5 and natural language generation problems in Chapter 6 and 7.

Deep learning approaches rely on end-to-end representation learning and have discarded many decades of hand-crafted feature engineering [1, 7]. Representation learning refers to not only learning the representations that map features to results but also the representation of the features themselves [1, 7]. In the context of NLP, deep learning learns the parameter from the raw symbolic representation of text, which is modelled with a one-hot form vector dictionary, to final numerical outputs [8]. Deep learning utilises deep neural networks to learn rich, highly non-linear data representations through numerical optimisation techniques such as stochastic gradient descent [1, 7, 8]. For each training data instance, consisting of an input and an associated output, a scalar error value called the loss is computed by passing through multiple layers of complex data transformations with their associated parameters [1, 7, 8]. This loss reflects the difference between the predicted and ground truth output and is then differentiated with respect to each parameter in each layer with the **back-propagation** (BP) algorithm [17]. The BP algorithm allows us to nudge the parameters in a direction that will, on average, result in a lower loss the next time the same input is encountered.

This rapid growth and early success of deep learning for NLP have been primarily driven by the emergence of large-scale publicly available benchmark datasets, distributed word representations, and high-performance parallel computing hardware [9]. Despite remarkable advancements having been made in these benchmark tasks, many practical concerns are raised on whether deep learning models understand the tasks themselves [18–20] instead of exploiting bias, and spurious corre-

lations in data [21][5][6][7]. Deep learning methods result in a stationary set of model parameters that fail to provide legitimate uncertainty measurements. Hence, deep learning models are often considered as 'black-box' methods with less interpretability and explainability when compared with traditional statistical models [22]. These practical issues raise concerns about the robustness, trustworthiness, explainability and interpretability of deep learning [23–25]. Although the limitations of deep learning methods are widely acknowledged, work on practical solutions to address these concerns in NLP applications is contrastingly limited. These obstacles still stand in the way of applying neural networks to real-world problems, especially applications in critical fields such as healthcare, finance, and education. Overcoming these obstacles is necessary for machine learning researchers, engineers and practitioners to allow deep learning methods to create a meaningful impact on the economy and society.

On the contrary, these obstacles are not observed in the human learning process: humans possess an explainability and interpretability understanding of the world (equivalent to a model) and can perform robust and trustworthy decision-making (based on this model). The premise that humans possess a model-based explainable, and interpretable understanding of the world, is substantiated by the theories put forth by Daniel Kahneman in the field of decision-making under uncertainty [26,27]. Kahneman argues that individuals engage in two distinct cognitive processes in the decision-making process, the "associative" and the "reasoning". The "associative" process links current observations to previous experiences (possibly through a learned model). In contrast, the "reasoning" process allows for informed decision-making based on the present context (based on the model). By integrating these two processes, individuals can form an understandable and explicable understanding of the world, which serves as a model to facilitate the creation of robust and trustworthy decision-making.

[5]https://sitn.hms.harvard.edu/flash/2020/racial-discrimination-in-face-recognition-technology/

[6]https://hbr.org/2019/10/what-do-we-do-about-the-biases-in-ai

[7]https://www.forbes.com/sites/forbestechcouncil/2021/02/04/the-role-of-bias-in-artificial-intelligence/?sh=78584326579d

Based on the above analysis, one key difference between human learning and deep learning is that humans not only *invest the effort to understand the data observed* but also to *figure out how the data is generated*. For deep learning methods, however, the learning process only involves understanding the data observed. The critical distinction here motivates us to seek for more generative approach for deep learning: *methods that can infer cause from observations*[8]. In this thesis, we adopt a well-established computational framework to mimic the human cognition process and understand the world in a generative manner [28, 29]. Using the language of probability together with this framework, we can represent our belief with probability distributions and update our belief with the Bayes rule given the evidence observed [28, 29]. The process of understanding the world with Bayesian methods is known as *Bayesian inference*.

In this thesis, we explore methods that perform Bayesian inference in deep learning, a field referred to as **Bayesian deep learning** (BDL) or **deep Bayesian learning** (DBL). The benefits of introducing the Bayesian side into deep learning allow the models to *invest the effort to understand the data observed* (via deep learning) and also to *figure out how the data is generated* (via Bayesian inference). Furthermore, the incorporation of Bayesian methodology in deep learning models holds the potential to address some concerns, such as robustness, trustworthiness, explainability, and interpretability in NLP applications. This is achieved by introducing latent variable assumptions that help explain and represent the underlying causes of the data, as demonstrated in the methodology Chapter 3 and later in Chapters 4, 5, 6 and 7 with various NLP applications. In particular, we focus on a specific set of model families, known as the **deep latent variable models** (DLVMs), which is widely adopted for performing Bayesian deep learning in the research community. Latent variable models assume there exist unobserved latent variables that generate the data we observed. For latent variable models in Bayesian deep learning, the central task concerns the posterior inference for latent variables. In this thesis, the **variational inference** (VI) method and, in particular, **neural variational**

[8]Here, the cause refers to the statistical association instead of causality.

inference (NVI) [4–6] is adopted as the Bayesian inference technique. NVI can be seamlessly integrated into deep learning along with DLVMs, as explained later in Chapter 3 and can be used to build various NLP applications, as described in Chapter 4, 5, 6 and 7.

1.2 Research Questions

The main research questions of this thesis can be summarised as follows:

Research Theme 1: **Exploring Bayesian Deep Learning for Urgent Instructor Intervention Needs Identification in MOOCs Forums.**

Our first research theme pertains to exploring Bayesian deep learning for urgent instructor intervention needs identification in **massive open online courses** (MOOCs) forums. MOOCs have become a popular choice for e-learning thanks to their great flexibility. However, due to the large number of learners and their diverse backgrounds, it is taxing to offer real-time support. Learners may post their feelings of confusion and struggle in the respective MOOC forums, but with the large volume of posts and high workloads for MOOC instructors, it is unlikely that the instructors can identify all learners requiring intervention. This problem has been studied as an NLP task with deep learning recently and is known to be challenging, due to the imbalance of the data and the complex nature of the task. However, the performance of deep learning is known to be unstable with regard to random weight initialisation and has limited capability to incorporate uncertainty. One way to address these problems is to apply probabilistic modelling techniques in deep learning, and thus, in Chapter 4, we focus on the research question:

Research Question 1: ***How do Bayesian methods, in particular, deep latent variable models, benefit deep learning for a user needs prediction task in education?***

Research Theme 2: **Proposing an Efficient Uncertainty Quantification Framework for Multilabel Text Classification.**

In our first research theme, we empirically show the benefits of two Bayesian deep learning methods (DLVMs and BNNs) to improve robustness and trustworthiness for urgent user intervention prediction in MOOCs. The conclusions from our first research theme lead to the following question: *Does this solution popularise from an education-specific domain to a more general natural language understanding domain?* Hence, our second research theme explores the application of DLVM for efficient uncertainty quantification in multi-label text classification tasks[9][10]. Despite rapid advances of modern AI, there is a growing concern regarding its capacity to be explainable, transparent, and accountable. One crucial step towards such AI systems involves reliable and efficient uncertainty quantification methods. Existing approaches to uncertainty quantification in NLP rely on **Monte Carlo dropout** (MCD) technique, which is known to not be computationally efficient in testing time due to layer-wised sampling; and hinders its applicability in real-time rending. Thus, in Chapter 5, we focus on the research question:

Research Question 2: ***How do we use deep latent variable models to achieve efficient uncertainty quantification in multi-label text classification tasks?***

Research Theme 3: **Proposing a Deep Generative XAI Framework for Natural Language Inference Explanation.**

To summarise our first two research themes, we have shown the benefits of DLVM in natural language understanding problems, leading to a natural question: *How can DLVM address these concerns in natural language generation problems?* Reflecting on this question, our third research theme explores an application of DLVM for **explainable artificial intelligence** (XAI) multiple explanation generation task. XAI with natural language processing aims to produce human-readable explanations as evidence for AI decision-making, which addresses explainability and transparency.

[9]https://www.ibm.com/blogs/watson/2020/11/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/

[10]https://huggingface.co/tasks/text-classification

However, from an HCI perspective, the current approaches only focus on delivering a single explanation, which fails to account for the diversity of human thoughts and experiences in language. Thus, to address this gap, in Chapter 6, we focus on the research question:

Research Question 3: ***How do we use deep latent variable models to generate multiple explanations for a natural language inference task?***

Research Theme 4: **Semi-Supervised Deep Latent Variable Models for Paraphrase Generation.**

Up to this point, our first three research themes focus on supervised machine learning setup; however, for NLP problems, the availability of large-scale annotated data is often not possible. To address this challenge, we present another question: *Can we use DLVM for semi-supervised machine learning problems in natural language generation tasks?* In responding to this question, our fourth research theme explores DLVM for semi-supervised learning problems in paraphrase generation tasks, where the missing target pair for unsupervised learning is modelled as a latent paraphrase sequence. Additionally, in our previous three contributions, we have utilised an assumption that the latent variable can be represented as a smooth and continuous distribution (diagonal multivariate Gaussian); however, for NLP applications, the continuous latent variable is inherently less interpretable. Thus, in Chapter 7, we explore the usage of the discrete latent variable in DLVM and focus on the research question:

Research Question 4: ***How to perform semi-supervised learning with deep latent variable models for paraphrase generation tasks?***

1.3 Research Objectives

Based on the research questions in section 1.2, the corresponding research objectives with respect to each research question are listed below:

Research Question 1: ***How do Bayesian methods, in particular, deep latent variable models, benefit deep learning for a user needs prediction task in education?***

This research question is addressed by the following objectives:

RO1.1: Explore learner-based text posts for an intervention need prediction task with two Bayesian methods in deep learning;

RO1.2: Compare Bayesian deep learning models to its baseline deep learning models under similar circumstances;

RO1.3: Present results and analysis of the benefits of Bayesian deep learning on adding more trust and robustness to AI in education;

RO1.4: Propose methods to achieve similar or better performance compared to non-probabilistic neural networks and are more robust with random initialisation.

Research Question 2: ***How do we use deep latent variable models to achieve efficient uncertainty quantification in multi-label text classification tasks?***

This research question is addressed by the following objectives:

RO2.1: Propose methods of representing epistemic and aleatoric uncertainties conditional on text with deep latent variable models;

RO2.2: Propose efficient uncertainty quantification methods with posterior analysis in the (approximated) latent and data space;

RO2.3: Conduct extensive experiments and studies on diverse neural network architectures;

RO2.4: Prove the benefits of explicitly modelling uncertainty in neural networks.

Research Question 3: ***How do we use deep latent variable models to generate multiple explanations for a natural language inference task?***

This research question is addressed by the following objectives:

RO3.1: Propose a deep generative XAI framework based on the Transformer architecture;

RO3.2: Show that the proposed framework can generate multiple explanations in two steps;

RO3.3: Conduct experiments and analysis on the proposed framework to validate its effectiveness.

Research Question 4: ***How to perform semi-supervised learning with deep latent variable models for paraphrase generation tasks?***

This research question is addressed by the following objectives:

RO4.1: Study semi-supervised learning with DLVMs for paraphrase generation task;

RO4.2: Propose a novel unsupervised model named variational sequence auto-encoding reconstruction (VSAR);

RO4.3: Propose a novel supervised model named dual directional learning (DDL);

RO4.4: Combine VSAR and DDL model for semi-supervised learning in paraphrase generation tasks;

RO4.5: Empirically show the effectiveness of the combined model.

1.4 Main Contributions

This thesis contributes to the field of deep learning and natural language processing; the main contribution of this thesis can be summarised as follows:

Exploring and proposing novel methods (supervised and semi-supervised) and applications (natural language understanding and generation) with deep latent variable models for natural language processing.

Further contributions are as follows:

- Studied both *supervised learning* and *semi-supervised learning* paradigms with deep latent variable models:

  - Studied the benefits of deep latent variable models against their corresponding deep learning models (equivalent in model architectures and the number of trainable parameters) under supervised learning paradigm in Chapter 4, 5 and 6. These studies suggested that deep latent variable models are competitive in performance against deep learning methods while offering additional robustness, trustworthiness, explainability and interpretability.
  - Studied the benefits of deep latent variable models against their corresponding deep learning models (equivalent in model architectures and the number of trainable parameters) under semi-supervised learning paradigm in Chapter 7. This study showed that semi-supervised deep latent variable models are able to achieve state-of-the-art performance in paraphrase generation tasks and have great potential for semi-supervised learning settings.

- Proposed novel *NLP applications* for both *natural language understanding* and *natural language generation* problems with deep latent variable models:
  - Propose novel Bayesian deep learning models for urgent instructor intervention need identification in MOOCs Forum, in Chapter 4, as a natural language understanding problem. This is the first research on how Bayesian deep learning can be applied to a text-based learning analytics application.
  - Propose novel deep latent variable models for multilabel text classification tasks, in Chapter 5, as a natural language understanding problem. This is the first research on using deep latent variable models for efficient uncertainty quantification purposes in multilabel text classification tasks.
  - Propose novel deep latent variable models for the natural language inference explanations generation task, in Chapter 6, as a natural language generation problem. This is the first research on how deep latent variable models can be applied to generate multiple explanations in the Stanford natural language inference task.

– Propose novel deep latent variable models for paraphrase generation tasks, in Chapter 7, as a natural language generation problem. This is the first study on how deep latent variable models can be used for semi-supervised learning in paraphrase generation tasks.

1.5 Thesis Structure

This thesis contains in total of 8 Chapters and is organised as follows:

- **Chapter 1** presents the general background and motivation, research questions and their corresponding research objectives, main contributions, as well as the structure of this thesis.
- **Chapter 2** introduces relevant literature in deep learning, natural language processing, and deep latent variable models, which are associated with this thesis, and align our contributions with the literature.
- **Chapter 3** provides the technical knowledge and general methodology for readers to understand the content and contributions of this thesis.
- **Chapter 4** presents an exploratory study on Bayesian deep learning methods, in particular, deep latent variable models for text classification, using MOOC forum data from an educational domain (**RQ**1).
- **Chapter 5** presents a reliable and efficient Bayesian uncertainty quantification framework for both epistemic and aleatoric uncertainty on multi-labelled text classification tasks (**RQ**2).
- **Chapter 6** moves on from natural language understanding to the generation task and presents a novel conditional deep generative XAI framework for multiple natural language explanations generation (**RQ**3).
- **Chapter 7** further explores the natural language generation field and presents a novel semi-supervised deep generative framework for paraphrasing generation tasks (**RQ**4).

- **Chapter 8** summarises the works presented in this thesis, answers the research questions and outlines future research venues.

CHAPTER 2

Literature Review

This chapter starts by introducing the general categories of natural language processing, with a special focus on deep learning, and then points out the associated contributions in this thesis in the field of natural language processing. Next, this chapter presents recent works on deep latent variable models for text applications, which is the main focus of this thesis. Specifically, this chapter reviews the main lines of research on deep latent variable models for texts from an application perspective and demonstrates how the work in this thesis identifies existing gaps and presents new solutions and contributions.

2.1 Deep Learning based Natural language Processing

Natural language processing (NLP) concerns the field of study on the interactions between computational processing, and human languages [30], involves the engineering of computational models and processes to solve practical problems in order to understand human languages. The origin of research on NLP can date back to 1950 [31] when Alan Turing at that time published the well-known article and

proposed the Turing test. The Turing test involves the automated interpretation and generation of natural language by a computer agent, hence can be considered an early form of NLP task. In the 1950s, early work of NLP research preliminary relied on rule-based approaches; from the 1980s, the NLP field moved towards using data-driven computation methods involving techniques borrowed from statistics, probability, and machine learning [32,33]. In recent years (especially the 2010s onwards), a major shift in NLP research technique was the introduction of deep learning [1, 7], harnessed by computational advancements in graphical processing units [34, 35] and more recently tensor processing units [36]. Another important factor resulting in the blossom of deep neural network based models is the increasing amount of data available [9]. Up to this point (year 2022), deep learning is no doubt the dominant approach for natural language processing and has achieved state-of-the-art performance in almost all NLP tasks [10, 37].

For decades, solutions for NLP problems have been based on machine learning approaches with shallow models (e.g. **support vector machine** (SVM) [38], and logistic regression [39]) trained on very high dimensional and sparse features, and liaised heavily on hand-crafted features [40]. Early works of NLP research using neural network models have produced superior results on various NLP tasks, with the success of distributed word embeddings [41, 42] and deep learning technique [7]. The success of deep learning triggered a revolution in NLP research and has made deep learning the most popular choice of machine learning approaches. However, even before deep learning was widely adopted, back in 2011, Collobert et al. [43] already demonstrated that a simple deep learning framework could reach state-of-the-art performance with raw text data in various NLP tasks, such as **named entity recognition** (NER), **semantic role labelling** (SRL), and **part-of-speech** (POS) tagging. In the year 2022, deep learning is no doubt the state-of-the-art method for NLP [10, 37].

Research for NLP and deep learning can be divided into two general aspects: distributed representation and deep learning architecture. The first aspect, distributed representation, is often used as the first data processing layer in a deep learning model to convert text into numerical representations [40]. For distributed

representation, motivation date back to [44], which aimed to learn semantic representations in lower dimensions. Successful instances of distributed representations at word-level include word2vec [41, 42] and Glove [45]. Limitations of such word-level representations include the inability to represent phrases when combining two individual works, later addressed by region embedding [46]; and sentiment ambiguity as a result of small window size [47], later addressed by sentiment-specific embedding [48]. For word-level representations, however, an inherent limitation of these representations for language with a large vocabulary is the problem of **out-of-vocabulary** (OOV) [40]. One solution to this is character embedding which considers each word as the composition of individual characters [49] and has shown significant benefits [50, 51] in various NLP applications. In recent years, contextual representations have raised attention in NLP research and have been found essential to improving the performance of NLP tasks. One important challenge for word-level and character-level distributed representation is that no context-specific representation is considered, while for downstream tasks in NLP, understanding the actual context is necessary [52]. Some popular types of contextual embeddings include **Embeddings from Language Models** (ELMO) [53], and other pre-trained embeddings such as **Bidirectional Encoder Representations from Transformers** (BERT) [54] and **Generative Pre-Training** (GPT) model family [55, 56].

The second aspect of NLP research with deep learning concerns model architectures, three main categories of deep learning model architectures are **convolutional neural networks** (CNNs), **recurrent neural networks** (RNNs), and attention-mechanism neural networks (Transformer); for general introduction with these architectures please refers to section 3.2. For NLP, the motivation for using CNNs is to extract higher-level features from constituting words or n-grams [40]. The use of CNNs for sentence modelling is pioneered by [43] and is later proliferated to the NLP community by [57, 58]. With pooling operations such as max, min and sum, deep CNNs models can grasp a high-level abstraction of a sentence. Apart from a single abstract representation, architectures based on CNNs can work in word-based prediction tasks such as NER, POS tagging, and SRL with window approach [59] or combine with structured prediction techniques such as **conditional random**

field (CRF) [60]. For application-wise, CNNs have been applied to text classification [57, 58, 61], representation learning [62, 63], information retrieval [64] and many other NLP applications [65]. In Chapter 5 and 6, we use architecture based on CNNs, motivated by [57, 58] for text classification and latent variable inference purpose, respectively.

The CNNs model architecture is wired in a way to capture the essence of information, as the pooling operation results in representation in a translation-invariant form; however, this might not be helpful in tasks such as text generation where sequential information needs to be preserved. In the NLP field especially, most tasks can generally be framed as sequential modelling problems. The motivation for RNNs model architecture is to capture the inherent sequential nature preserved in language, where units can be characters, words, or even sentences [40]. From a linguistic perspective, words in a language develop their semantical meaning based on the previous words and current context in the sentence [66]. These linguistic aspects grant a strong motivation for researchers to use RNNs over CNNs for the NLP domain. Another factor to support RNN's suitability for sequence modelling tasks comes from its ability to handle variable lengths of text, including very long sentences, paragraphs and even documents [67]. The modelling capability for unbounded context [68] is the main selling point of why RNNs model architecture is adopted widely in the thesis, and hence are used in Chapter 4 and 5. These motivations, however, do not lead to a conclusion on the superiority of RNNs over other deep learning architectures [69, 70]. For application-wise, RNNs have been applied to word-level classification tasks, such as NER [71] and language modelling [72, 73]; sentence-level classification tasks, such as sentiment analysis [74] and text classification [12, 75]; and language generation tasks, such as machine translation [13, 76], conversation modelling [77, 78] and visual question answering [79].

One potential problem when using RNNs model architecture is that it is hard to encode information-rich and long sequences in a fixed vector [80], which inspires the attention mechanism. The attention mechanism attempts to provide dynamic attention weights to all the hidden state vectors in the sequence and conditions on a context vector [40]. A recent milestone for attention mechanism is the invention of

Transformer architecture [81], which is primarily based on the self-attention mechanism and achieved state-of-the-art performance in sequence-to-sequence learning problems. With the development of self-supervised learning [82, 83], pre-trained language models with transformer architectures represent the state-of-the-art for NLP [10, 37]. In Chapter 6 and 7, we use the Transformer-based encoder-decoder architecture for **explainable artificial intelligence** (XAI) explanation generation and paraphrase generation, respectively.

In this thesis, we mainly focus on end-to-end deep learning models for NLP applications, which consider distributed representations and deep learning architecture as complete components. Broadly speaking, from an applications perspective, the problem of natural language processing[1] can be further separated as natural language understanding, and natural language generation [84, 85]. We explore the problem of natural language understanding application in Chapter 4 and 5, and the problem of natural language generation application in Chapter 6 and 7.

In Chapter 4, we explored a domain-specific natural language understanding application in education, which automatically identifies the intervention needs of learners from online forum data. A short review of work related to identifying user intervention needs is presented in section 4.2.1. In Chapter 5, we further explore the application of natural language understanding with multilabel classification tasks. We present a novel uncertainty quantification framework and conduct extensive experiments to understand its effectiveness in improving results across diverse neural network architectures. A brief review of uncertainty quantification literature for NLP is presented in section 5.2. In Chapter 6, we explore a relatively recent natural language generation task, named explainable-NLP, which aims to present structured, human-readable text-form explanations as evidence to support deep neural network decisions. A short review of XAI techniques applied to NLP is presented in section 6.2.1. In Chapter 7, we explore semi-supervised learning of paraphrase generation tasks with deep latent variable models and a relevant review for work associated with paraphrase generation is presented in section 7.5. In summary, in this thesis,

[1]Note here we do not consider the field of research associated with spoken language processing.

we present various NLP applications as our contributions to both text understanding and generation problems.

2.2 Deep Latent Variable Models for Text

Latent variable models (LVMs) and **probabilistic graphical models**[2] (PGMs), more generally, are rooted in statistics [28, 86, 87] and are primarily based on the assumption of the presence of latent unobserved variables which are used to account for overdispersion or to implicitly model association structures. [87] suggests that many of the existing statistical models can be fitted within the general unified framework of LVMs. As the latent variables are never observed in data, the central task for LVMs is concerned with the posterior inference for latent variables [86–88]. Traditional Bayesian inference for posterior inference methods in these models utilises the power of exponential family and conjugate priors for exact inference [89], which at the same time requires less consideration for potential issues with computation complexity and computation cost [90]. However, for complex real-world problems and data coming from rich, expressive classes, the exact inference methods may not even exist and suffer from their limited expressiveness [91]. Hence, approximate inference methods are used to compute the complicated intractable posterior distributions and have become a hot spot in the machine learning research community [9, 92].

There are mainly two families of approximation methods in Bayesian research, which are the stochastic approximation and the deterministic approximation [28]. The stochastic approximation refers to the sampling-based methods, and a large family of these algorithms have been proposed in the rich history of Bayesian statistics, the most famous one named as **Monte Carlo Markov Chain** (MCMC) [29]. For stochastic approximation, we need to carefully design and define a good sampler, and transformation matrices for the Markov chain in order to converge to the true posterior [28, 29]. The deterministic methods are mainly developed based on the **variational inference** (VI) [28, 93], which introduces the latent variable to

[2]Note here we implicitly refer to directed graphical models with latent variables in the scope of PGMs.

represent the generation process of data and utilises an approximated posterior to estimate the intractable posterior distribution. In this thesis particularly, we explore the problem of Bayesian inference through the lens of VI, as a key technique for performing approximate posterior inference [94]. For VI in deep learning, in particular, **neural variational inference** (NVI) [4–6] is adopted as it allows scalable learning with stochastic gradient optimisation algorithms initially designed for deep learning. Combining NVI with deep learning results in a new type of modelling framework called the **deep latent variable models** (DLVMs), which represents the central focus of our research in this thesis.

Before the deep learning era, LVMs are widely adopted for NLP applications. LVMs offer a flexible and unified framework to declare prior knowledge and structural relationships in complex datasets [94] and have a long and rich history in building NLP applications, such as statistical alignment for translation [95], topic modelling [96], unsupervised part-of-speech tagging [97], grammar induction [98], document modelling, document clustering, topic modelling and parsing [29]. With recent advancements in deep learning for NLP applications, such as language modelling, machine translation, and question answering [9], DLVMs offer a natural interface to combine the advantages of LVMs on probabilistic modelling and the advantages of deep learning on its performance. In recent years, DLVMs have been widely studied for NLP applications [99–103]. Prior research for NLP applications with DLVMs can be generally placed into one of two categories: the first category focus on the assumptions and properties of the latent variable, whether continuous or discrete; the second category focuses on specific NLP applications with DLVMs.

From the first category, the most common and widely adopted DLVM for NLP application is the standard VAE [4] model with a Gaussian prior [104] published by Bowman et al. in 2015, also known as the sentence VAE [94]. Since then, sentence VAE has been used as a powerful tool for unsupervised representation learning and also later extended for conditional representation learning [105–109]. However, DLVMs, in general, suffer from posterior collapse [110, 111]. In this thesis, we presented a study on posterior collapse when using DLVMs for uncertainty quantification purposes in Chapter 5. In literature, multiple studies have been con-

ducted to combat this issue [111–117], in particular β-VAE [112] presents a simple but effective solution and introduces a penalty term to balance the reconstruction term and prior-posterior regularisation term intuitively and is adopted as one of our baselines in Chapter 7. While much of the research focuses on continuous latent variable models, the text is naturally presented in discrete form and may not be well represented with continuous latent variables [99, 118]. Early work on discrete deep latent variable models [118–121] adopted the REINFORCE algorithm [6, 122]; however, it suffers from very high variance [99, 123]. With the recent advancement in statistical relaxation techniques, Gumbel-Trick [3, 124] was utilised, to model discrete structures in the latent variable model of the text [123, 125–127].

The second category of study for DLVM focuses on particular novel NLP applications. As stated in Chapter 1, natural language processing applications can be further separated as natural language understanding and natural language generation. For natural language understanding tasks, DLVMs have been used for sequence labelling [128], dialogue classification [129], language modelling [130–133], document modelling [134], question and answering [134], topic modelling [135] and unsupervised parsing [136]. For natural language generation tasks, DLVMs have been used for machine translation with missing words [137], dynamic sentence construction [138], dialogue generation [119, 139, 140], content selection [141], summarisation [118, 142], data-to-text generation [143] and machine translation [144].

In this thesis, we explored DLVMs with a continuous latent variable assumption in Chapter 4, 5 and 6; and adopted Gumbel-Trick with subset sampling [145] for discrete latent variable assumption in Chapter 7. However, the main focus is to use DLVMs to construct novel NLP applications. We explored the problem of natural language understanding application in Chapter 4 and 5, and the problem of natural language generation application in Chapter 6 and 7. In Chapter 4, we explored the usage of DLVMs for reducing variance against random initialisation (robustness) and providing uncertainty measurement (trustworthiness) for an AI in education application which automatically identifies the intervention needs of learners from online forum data. In Chapter 5, we explored the usage of DLVMs as a component to efficiently model uncertainty (trustworthiness) via analytical solution in latent

space for general text understanding tasks. In Chapter 6, we explored the usage of DLVMs for an XAI explanation generation task, which used a latent variable to model the semantic content for explanations (explainability and interpretability). In Chapter 7, we explored the usage of DLVMs for semi-supervised learning in paraphrase generation tasks. The latent variable sequence represents the associated missing paraphrase label for each data used for unsupervised learning (explainability), and we additionally use the discrete latent variable to represent tokens from the vocabulary (interpretability). Our contributions provide insights into how DLVMs can be constructed and used for NLP applications.

CHAPTER 3

Technical Background and Methodology

This chapter provides the necessary technical background and methodology for readers to understand this thesis. It begins with an introduction to probabilistic machine learning and the deep learning paradigm. It follows with a description of the Representation Theorem, leading to our assumption on modelling data. Then, this chapter introduces learning and inference in deep latent variable models. Later, this chapter describes some advanced techniques for probabilistic machine learning. Finally, this chapter ends with a description of the general methodology for the implementation of the research included in this thesis and justifies why the methodology is adopted in this thesis. The actual model implementation details for each application are presented in the later chapters of this thesis.

3.1 Probabilistic Machine Learning

Machine learning (ML) concerns the study of computer algorithms that leverage data to improve their performance on some set of tasks [146]. A formal definition of machine learning, borrowed from [147], based on [146], states that:

"A computer program is said to learn from data $\boldsymbol{D}$ *with respect to some class of*

tasks $\boldsymbol{T}$*, and performance measure* $\boldsymbol{P}$*; if its performance at tasks in* $\boldsymbol{T}$*, as measured by* $\boldsymbol{P}$*, improves with data* $\boldsymbol{D}$*."*

Machine learning algorithms build mathematical models $\boldsymbol{M}$ (either deterministic or probabilistic [148]) to represent the tasks $\boldsymbol{T}$ information based on sample data $\mathcal{D}_{train}$ observed (training set[1]) to make predictions or decisions on new data $\mathcal{D}_{test}$ (test set) without much explicit supervision [149]. Based on the category of model $\boldsymbol{M}$, machine learning could be interpreted from two perspectives: (1) from an optimisation perspective, where the ultimate aim is to seek an optimal function for data; and (2) from a probabilistic perspective, where the ultimate goal is to find a probability distribution for data.

In this thesis, the focus lies on the interpretation from Bayesian (or more generally probabilistic) perspective. In this perspective, we treat all unknown quantities as random variables, that are endowed with probability distributions. There are two main reasons we adopt a probabilistic approach [147]. First, this allows decision-making under uncertainty, which is critical in many applications such as finance, healthcare and education. The first reason is justified with research presented in Chapter 4 and 5. Second, probabilistic modelling is the language used by most other areas of science and engineering, and thus provides a unifying framework between these fields. The second reason is justified when we use Bayesian statistics as a tool to analyse and model data, as presented in Chapter 5, 6 and 7. The advantages of these two reasons also strengthen the reason for us to choose a Bayesian (probabilistic) interpretation, as the scope of this thesis aims to study and provide insights concerning the robustness, trustworthiness, interpretability and explainability of deep learning. For the philosophy behind Bayesian statistics, the readers may refer themselves to [150] and [151] for more information.

[1]In standard machine learning setting, a small portion of the training dataset is used for model comparison and selection, known as the validation set.

3.2 Deep Learning

Deep learning (DL) is a sub-field of machine learning which is primarily based on neural networks, and representation learning [1, 152]. Traditional machine learning methods heavily rely on expert hand-craft features [153], known as feature engineering. The performance of these traditional machine learning algorithms predominantly depends on the feature representation of the raw data provided. This dependence on the quality of data representations is a general phenomenon throughout computer science and even in our daily life. Take an example from computer science; operations such as searching a target file from a data collection can proceed exponentially faster if the data is structured and organised. Take an example in daily life; people can efficiently perform arithmetic calculations with Arabic numbers but find the same operations on Roman numbers much more time-consuming. Hence, it is not surprising that the choice of representation has an enormous impact on the performance of machine learning algorithms. A simple visual example, as suggested in [1], is presented in Figure 3.1.

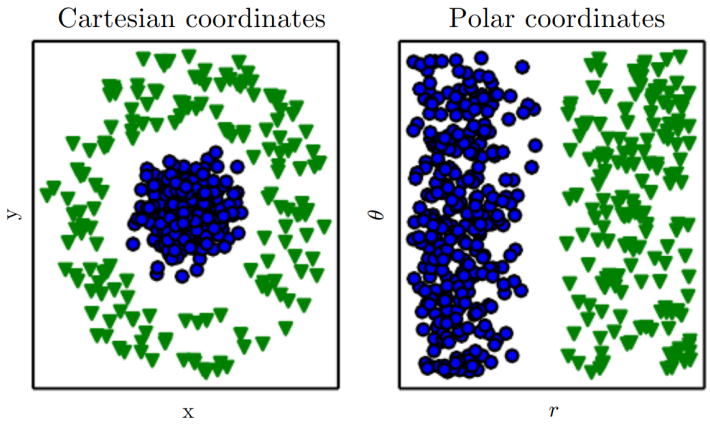

Figure 3.1: Chapter 3: Example of different representations based on a simple machine learning classification problem that we would like to draw a line between two categories of data. On the left, we represent data with Cartesian coordinates, and the task is impossible. On the right, we represent the data with polar coordinates, and the task becomes simple to solve. (Figure taken from [1].)

Many machine learning tasks can be solved by designing the right set of features to extract from data. For many complex tasks, however, it is often an ill-defined

problem[2] to know what features should be extracted automatically (even for experts sometimes). For example, suppose that we would like to write a machine learning algorithm for machine translation purpose, which translate English to French. The solution to this problem is to use machine learning to discover not only the representation of English to French but also the representation of English and French themselves. This approach is known as representation learning [152]. However, for an ill-defined problem, it can be difficult to extract high-level, complex, abstract features from raw data directly, especially when it is nearly as difficult to obtain a representation as to solve the original problem.

Deep learning solves this central problem (i.e. extracting high-level abstract features from raw data directly) in representation learning by introducing representations that are expressed in terms of other, simpler representations [2][3], as demonstrated with a computer vision example in Figure 3.2. It has been suggested that similarly, phenomenons appear in natural language processing, deep learning model can capture structural information about language [154] and linguistic knowledge [155] in different layers of the model.

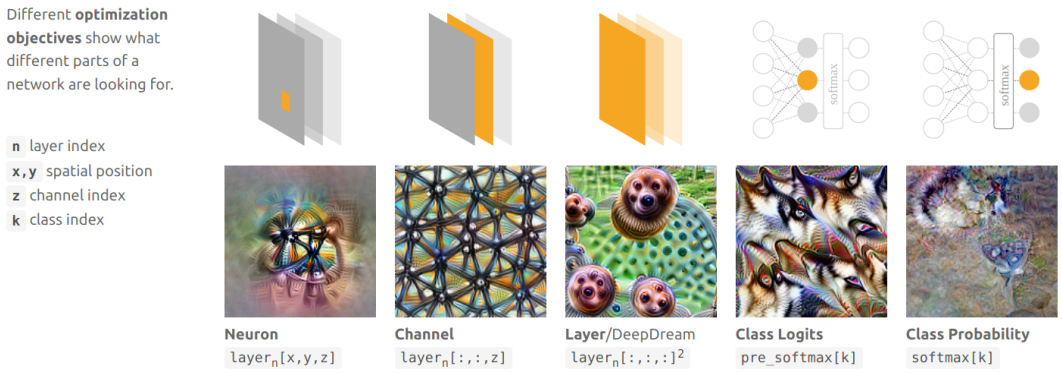

Figure 3.2: Chapter 3: A visual example of representation captured in different neural network layers. (Figure taken from [2].)

The quintessential example of a deep learning model is the feed-forward deep network, or so-called **multilayer perceptrons** (MLPs). An MLP consists of at

[2]In the study of problem-solving, any problem in which either the starting position, the allowable operations, or the goal state is not clearly specified or a unique solution cannot be shown to exist is called an ill-defined problem; also known as an ill-structured problem.

[3]Please also check: https://microscope.openai.com/models

least three layers of nodes: an input layer, a hidden layer, and an output layer [156]. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. Each neuron can be considered as an instance of the perceptron learning algorithm [157], invented in 1958 at the Cornell Aeronautical Laboratory. An illustration of a neuron node is presented in Figure 3.3. An MLP can be interpreted as a mathematical function mapping some set of input values $\boldsymbol{X}$ to output values $\boldsymbol{Y}$. The complex function is formed by composing many simpler functions, layer by layer. As shown in [91, 158–160], MLPs are universal function approximators: with enough neurons in the hidden layer, they are capable of approximating arbitrary functions between finite dimensional vector spaces with arbitrary accuracy. Hence, MLPs can be used to create mathematical models for regression analysis problems. As classification is a special case of regression when the response variable is categorical [156], MLPs are equivalently applicable to classification problems.

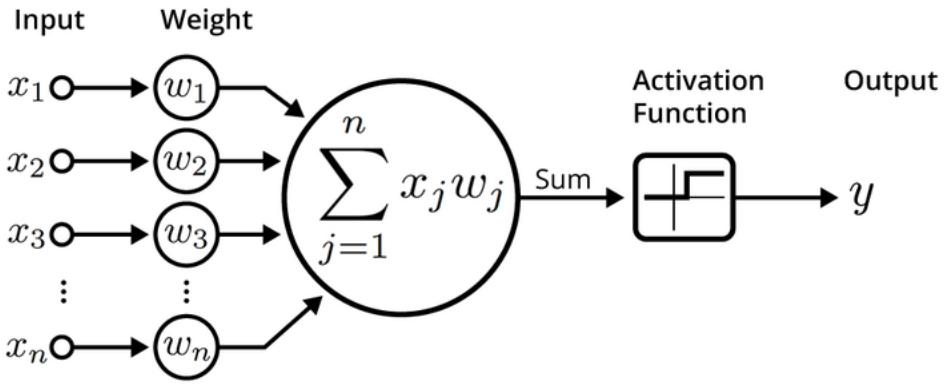

Figure 3.3: Chapter 3: An illustration of an artificial neuron. Source from Becoming Human

For deep learning, many other types of network architectures are developed, and we will briefly describe some classical ones here. The first type of architecture is the **convolutional neural networks** (CNNs), which are inspired by the receptive field [161] in visual cortex [162] and initially invented for image processing tasks [163,164]. In 2012, a CNN model called AlexNet [165] won the ImageNet large-scale visual recognition challenge and triggered attention and interest in deep learning techniques [1]. CNN models are later extended to solve natural language processing problems with applications such as semantic parsing [166], sentence modelling [57],

and text classification [58]. CNN models are used in this thesis in Chapter 5 and 6.

The second type of architecture is the **recurrent neural networks** (RNNs) [164, 167], which are designed specialised at modelling temporal dynamic behaviour in data. Three main adopted RNN models are Elman network [168], Long Short Term Memory network [169] and Gated Recurrent Unit network [170]. RNN models have been found in many applications such as natural language understanding [171], natural language generation [13, 172], handwriting recognition [173, 174] and speech recognition [72, 175]. RNN models are used in this thesis in Chapter 4 and 5.

The last type of architecture is the **Transformer** [81], which is primarily based on the self-attention mechanism, initially introduced from the machine translation field [80, 176]. The Transformer architecture significantly advanced the **artificial intelligence** (AI) field and is currently the backbone for most of the state-of-the-art models in both computer vision [177], and natural language processing [10]. Transformer models are used in this thesis in Chapter 5, 6 and 7. In the scope of this thesis, we do not focus on designing novel neural network architectures, but rather consider them as parameterised functions with a structural inductive bias to approximate the underlying probability distribution of the observed and unobserved random variables. For a detailed description of the network architectures, the readers are recommended to refer to [1] for MLP, CNN and RNN; and [81, 178] for Transformer[4].

3.3 Exchangeability and Representation Theorem

In previous sections, we have introduced the advantages of the probabilistic machine learning and deep learning paradigm which is used in this thesis. Before moving towards mathematical expressions of probabilistic deep learning and other technical details, we would like to briefly discuss our assumptions about how a sequence of data is represented.

Given a sequence of random variables $(\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, ..., \boldsymbol{\mathcal{X}}_N)$ as observations, to represent the statistical belief concerning this sequence in the form of probability dis-

[4]Please also check: https://jalammar.github.io/illustrated-transformer/

tributions, more assumptions need to be enforced to alleviate the impact of the order information for the random variables in this sequence. A commonly appropriate assumption in statistical machine learning is that the order of the random variables contains no information. In other words, the statistical belief should be invariant to the permutation of the random variables in this sequence. Hence, a natural assumption follows this is that the observations are independent and identically distributed (i.i.d.). However, this assumption implies mutual independence between random variables in an observed sequence, which is an extreme assumption. Many of the random variables and their associated values we observe for machine learning problems might be correlated.

An alternative, more generalised form of independent and identically distributed assumption for the sequence of random variables is that it may instead consider as an exchangeable sequence:

$$(\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, ..., \boldsymbol{\mathcal{X}}_N) \overset{d}{=} (\boldsymbol{\mathcal{X}}_{p(1)}, \boldsymbol{\mathcal{X}}_{p(2)}, ..., \boldsymbol{\mathcal{X}}_{pn}) \quad \forall p \in \mathbb{S}_\infty \tag{3.1}$$

where $\overset{d}{=}$ denotes equality in distribution, and $\mathbb{S}_\infty$ is the set of all permutations of the natural numbers, from 1 to N, which permute at most a finite number of elements (bounded by N).

De Finettis Theorem [179], also known as Representation Theorem, suggests that a sequence of random variables can be considered as an exchangeable sequence, if and only if there exists a probability measure for $\boldsymbol{\theta}$, such that $\boldsymbol{\mathcal{X}}_{1:N} \overset{i.i.d.}{\sim} \boldsymbol{\theta}$. In other words, observations are conditionally independent and identically distributed given the probability measurement $\boldsymbol{\theta}$. In mathematical terms, the Representation Theorem is defined as:

$$P(\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, ..., \boldsymbol{\mathcal{X}}_N) = \int \prod_{i=1}^{N} P(\boldsymbol{\mathcal{X}}_i|\boldsymbol{\theta}) P(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{3.2}$$

The Representation Theorem explains why we have model parameters $\boldsymbol{\theta}$ in the first place, and there is no constraint on the dimension of the probability measure $\boldsymbol{\theta}$. Hence it can have either finite (parametric model) or infinite (non-parametric model) dimensions. Another insight from the Representation Theorem is that Bayesian in-

ference is a natural selection for statistical modelling as it is based on more generalised assumptions about data. This further strengthens the reason for us to choose a Bayesian (probabilistic) interpretation, as suggested in section 3.1.

3.4 Learning and Inference for Deep Latent Variable Models

Given the background introduced so far, here we briefly describe parameter learning and latent variable inference problem in deep learning in mathematical language. In the first part, we introduce the general method for parameter learning under the deep learning paradigm. Based on our introduction about the **De Finettis Theorem** in section 3.3, from a statistical learning perspective, we seek for an estimation of the parameter set $\boldsymbol{\theta}$. In deep learning, the estimation of the parameter set $\boldsymbol{\theta}$ is conducted through defining the maximum likelihood function, $L(\boldsymbol{\theta}; \boldsymbol{X})$, which is based on observed data $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)^\top$ and parameter set $\boldsymbol{\theta}$. The objective here is to find the optimal parameter set $\boldsymbol{\theta}^*$ that maximises the likelihood of the data, as:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \boldsymbol{X})$$

For deep learning problems in general, the likelihood function for a sequence of observations can be factorised as follows:

$$L(\boldsymbol{\theta}; \boldsymbol{X}) = \log P(\boldsymbol{\mathcal{X}}; \boldsymbol{\theta}) = \sum_{n=1}^{N} \log P(\boldsymbol{x}_n; \boldsymbol{\theta})$$

In practice, it is optimised through minimising the negative likelihood function as the loss $l(\boldsymbol{\theta})$ with gradient-based stochastic optimisation algorithms [180–182], as a standard practice in deep learning [1]:

$$l(\boldsymbol{\theta}) = -L(\boldsymbol{\theta}; \boldsymbol{X}) = -\sum_{n=1}^{N} \log P(\boldsymbol{x}_n; \boldsymbol{\theta})$$

These gradient-based stochastic optimisation algorithms perform parameter learn-

ing in the following general form:

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \bigtriangledown l(\boldsymbol{\theta}^{(i)})$$

Where i is the current optimisation step, η is a hyper-parameter called the learning rate, and $\bigtriangledown l(\boldsymbol{\theta^{(i)}})$ is the gradient of $\boldsymbol{\theta^{(i)}}$ given the likelihood function $l((\boldsymbol{\theta}))$, calculated via back-propagation algorithm [17]. To scale up parameter learning for large-scale data, in deep learning, we alternatively perform optimisation on mini-batch (size of $M \ll N$) [183], as:

$$L(\boldsymbol{\theta}; \boldsymbol{X}) \approx \sum_{n=1}^{M} \log P(\boldsymbol{x}_n; \boldsymbol{\theta})$$

When M is big enough, we essentially optimise the original objective; however, some research [184] suggests that small batch training leads to better results and generation in deep learning.

In the second part, we introduce the general method for inference in deep latent variable models. In the deep learning paradigm, by default, we make the assumption that our data $\boldsymbol{X}$ is fully observed, as a random variable $\boldsymbol{\mathcal{X}}$. In reality, however, this might not be true most of the time as the data $\boldsymbol{X}$ we observed only count for a subset of the population. Hence we assume that there exists an unobserved latent variable $\boldsymbol{\mathcal{Z}} = (\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N)$[5], which is relevant to our data observed. The probability of $\boldsymbol{\mathcal{X}}$ given parameter $\boldsymbol{\theta}$ can be retrieved via marginalising out the latent variable $\boldsymbol{\mathcal{Z}}$ from the joint probability distribution $P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}; \boldsymbol{\theta})$.

For latent random variable $\boldsymbol{\mathcal{Z}}$ with discrete probability, it is:

$$P(\boldsymbol{\mathcal{X}}; \boldsymbol{\theta}) = \sum_{\boldsymbol{\mathcal{Z}}} P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}; \boldsymbol{\theta})$$

Where for latent random variable $\boldsymbol{\mathcal{Z}}$ with continuous probability, it is:

$$P(\boldsymbol{\mathcal{X}}; \boldsymbol{\theta}) = \int_{\boldsymbol{\mathcal{Z}}} P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}; \boldsymbol{\theta}) d\boldsymbol{\mathcal{Z}}$$

[5]This thesis mainly focus on local latent variables, derivation of variational inference applied to the graphical model with both local and global latent variables can be found in [9].

The difference in the latent variable assumption leads to further considerations in optimisation, as suggested in section 3.5.1 later; however, here, we use the continuous case for the latent variable $\boldsymbol{\mathcal{Z}}$ for formula derivation in the rest of this section. In order to marginalise out the latent variable from the joint probability distribution $P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}; \boldsymbol{\theta})$, we need to infer $\boldsymbol{\mathcal{Z}}$ based on observed $\boldsymbol{\mathcal{X}}$ via Bayesian inference:

$$P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}}; \boldsymbol{\theta}) = \frac{P(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{Z}}; \boldsymbol{\theta})P(\boldsymbol{\mathcal{Z}})}{P(\boldsymbol{\mathcal{X}}; \boldsymbol{\theta})} = \frac{P(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{Z}}; \boldsymbol{\theta})P(\boldsymbol{\mathcal{Z}})}{\int_{\boldsymbol{\mathcal{Z}}} P(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{Z}}; \boldsymbol{\theta})P(\boldsymbol{\mathcal{Z}})d\boldsymbol{\mathcal{Z}}}$$

However, in general, it is intractable to calculate the normalising constant $P(\boldsymbol{\mathcal{X}})$, due to marginalise out all possible $\boldsymbol{\mathcal{Z}}$ from the joint probability $P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}})$. With latent variable $\boldsymbol{\mathcal{Z}}$, the deep learning training objective becomes:

$$L(\boldsymbol{\theta}; \boldsymbol{X}) = \log P(\boldsymbol{\mathcal{X}}; \boldsymbol{\theta}) = \log(\int_{\boldsymbol{\mathcal{Z}}} P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}; \boldsymbol{\theta})d\boldsymbol{\mathcal{Z}})$$

Parameter learning with this new training objective is difficult when latent variable $\boldsymbol{\mathcal{Z}}$ is introduced, due to the integration inside the log-probability; hence we alternatively use a well-developed approximated Bayesian inference technique called the **variational inference** (VI) [93, 185] to perform Bayesian inference over the training data $\boldsymbol{X}$. The origins of variational inference date back to the 1980s and are rooted in statistical physics [9]. Classical variational inference is restricted to conjugate or conditional conjugate exponential family models, as discussed in [93, 185]. Stochastic variational inference [186] is first proposed to allow variational inference performed on batch sampled data, which improves the scalability of the algorithm. The restriction on conjugate models is lifted with the development of black box variational inference [187], which allows Bayesian inference performed via gradient calculation with respect to data. Recent advances in amortised variational inference [4–6] further limit restrictions and allow using complex function approximation with deep neural network and efficient inference of local latent variables condition on data. Readers could refer to [9, 92] for a more comprehensive review of the variational inference technique.

In this thesis, we particularly draw our attention to the application of both black box and amortised variational inference, as our focus is in the context of

deep learning. The variational inference is a deterministic Bayesian approximation technique which introduces an extra parameter set $\boldsymbol{\phi}$ to approximate the real posterior distribution $P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta})$. It decompose the negative likelihood function into two components: **evidence lower bound** (ELBO) and gap between true posterior $P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta})$ and approximated posterior $q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})$ measured by **KullbackLeibler** (KL) divergence.

$$L(\boldsymbol{\theta};\boldsymbol{X}) = \text{ELBO}(\boldsymbol{\theta},\boldsymbol{\phi}) + \text{KL}(\boldsymbol{\theta},\boldsymbol{\phi})$$

For any approximated posterior distribution $q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})$ over $\boldsymbol{\mathcal{Z}}$ which satisfied the condition of $supp(q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi}) \subset supp(P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta}))$, we can write the likelihood function $L(\boldsymbol{\theta};\boldsymbol{X})$ in the following form:

$$L(\boldsymbol{\theta};\boldsymbol{X}) = \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log \frac{P(\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Z}};\boldsymbol{\theta})}{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}] + \text{KL}[q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})||P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta})] \tag{3.3}$$

Where

$$\text{ELBO}(\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log \frac{P(\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Z}};\boldsymbol{\theta})}{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}]$$

And

$$\text{KL}(\boldsymbol{\theta},\boldsymbol{\phi}) = \text{KL}[q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})||P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta})]$$

Since the value for KL term is always non-negative, thus we have $L(\boldsymbol{\theta};\boldsymbol{X}) \geq \text{ELBO}(\boldsymbol{\theta},\boldsymbol{\phi})$. A step by step derivation of the ELBO and KL term is given as the following:

$$\begin{aligned}
L(\boldsymbol{\theta};\boldsymbol{X}) &= \log P(\boldsymbol{\mathcal{X}};\boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log P(\boldsymbol{\mathcal{X}};\boldsymbol{\theta})] = \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log \frac{P(\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Z}};\boldsymbol{\theta})}{P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta})}] \\
&= \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log \frac{P(\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Z}};\boldsymbol{\theta})q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}{P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta})q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}] \\
&= \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log \frac{P(\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Z}};\boldsymbol{\theta})}{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}] + \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log \frac{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}{P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta})}] \\
&= \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log \frac{P(\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Z}};\boldsymbol{\theta})}{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}] + \text{KL}[q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})||P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}};\boldsymbol{\theta})]
\end{aligned} \tag{3.4}$$

The ELBO can be written as a function of data $\boldsymbol{X}$ with generative model parameter $\boldsymbol{\theta}$ from the jointly probability $P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}}; \boldsymbol{\theta})$ and variational parameter $\boldsymbol{\phi}$ from the variational family probability distribution $q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})$; where both parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ need to be learnt from data $\boldsymbol{X}$. Since the real likelihood function $L(\boldsymbol{\theta}; \boldsymbol{X})$ can not be directly calculated and we have $\mathrm{KL}[q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})||P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}}; \boldsymbol{\theta}) \geq 0$, the maximisation of the likelihood function essentially turns into maximisation of the $\mathrm{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi})$. The learning objective function for the fully observed data can be replaced with the problem of finding a set of two optimal parameters:

$$
\begin{aligned}
(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*) &= \arg\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathrm{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \arg\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \sum_{n=1}^{N} \mathrm{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) \\
&= \arg\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})}[\log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n; \boldsymbol{\phi})}]
\end{aligned}
$$

3.4.1 Variational Family

The variational family $q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})$ plays an essential role in the variational inference procedure; it is flexible in choosing the type of variational family $q(\boldsymbol{\mathcal{Z}})$ and its associated parameter $\boldsymbol{\phi}$. However, different selections of the $q(\boldsymbol{\mathcal{Z}})$ will lead to various choices and designs of the inference algorithm [92]. Note that the variational family $q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})$ does not necessarily have to condition on data; in the context of deep learning, we discuss here briefly two general forms of variational family, based on whether the inference method takes data as input: black box posterior form and amortised form. In this thesis, our works are mainly based on the assumption of amortised posterior form.

Black Box Posterior Form

In the black box posterior form, the variational parameter $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_N)$ is a concatenation of local variational parameter $\boldsymbol{\phi}_n$, and it increases with the size of the available data. Each local variational parameter $\boldsymbol{\phi}_n$ is not inferred from its associated data $\boldsymbol{x}_n$. To infer such approximated posterior distribution, we random initialise the value for each local variational parameter $\boldsymbol{\phi}_n$ and perform stochastic gradient optimisation until convergence of the evidence lower bound. This is primarily the

form of latent variable assumption (local latent variable) we have in the scope of the thesis [6].

Amortised Posterior Form

Amortised variational inference [4–6], also known as neural variational inference, is invented to further reduce the complexity of optimisation via introducing a global neural network (as encoder or inference network) to approximate the local variational parameter given data $\boldsymbol{X}$. The amortised form is mainly adopted in deep learning and formalises one of the most successful frameworks for deep generative models, called variational auto-encoder [4]. Under the amortised form, the neural network is used to run over data $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)$ to produce its associated local latent variable $\boldsymbol{\mathcal{Z}} = (\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N)$ with a shared variational parameter $\boldsymbol{\phi}$. Note that although the amortised form is usually used for a local latent variable, as in [4], they can be amended to work for a global latent variable; however, this is not in the scope of this thesis.

3.4.2 Statistical Assumptions for Local Latent Variable

We have briefly discussed the black box posterior form and the amortised form for variational families based on whether the posterior is conditional on the data. Here, we discuss further concerns about the statistical assumptions of the local latent variable, and we briefly describe two types of approaches adopted in this thesis: mean-field form and auto-regressive (structural) form.

Mean-field Form

The mean-field form in variational family [28, 188] has been extensively adopted and studied in the literature, where the dimensions of the local latent variable $\boldsymbol{z}_n$ are assumed to be mutually independent. This assumption greatly simplifies the complexity of the optimization process, especially in high dimensions. A generic

[6] For global latent variable, please check [9].

form of the mean-field variational family for a local latent variable $\boldsymbol{z}_n$ with a total of m dimensions is represented as:

$$q(\boldsymbol{z}_n) = \prod_{j=1}^{m} q_j(z_j) \tag{3.5}$$

The mean-field form is the preliminary assumption we take in this thesis, as suggested in Chapter 4, 5 and 6.

Auto-regressive Form

Another form of popular choice for the variational family is the auto-regressive form, which assumes that each distinct dimension of the variable depends on the previous dimension of the variable. The auto-regressive form assumes that there exist correlations between the dimensions of each local latent variable. A generic form of the auto-regressive variational family for a local latent variable $\boldsymbol{z}_n$ with a total of m dimensions is represented as:

$$q(\boldsymbol{z}_n) = q_1(z_1) \prod_{j=2}^{m} q_j(z_j | z_{1:j-1}) \tag{3.6}$$

The auto-regressive form is explored in Chapter 7.

3.4.3 Optimise the model parameter for ELBO

If we tried to optimise the model parameters separately, maximising the ELBO over $\boldsymbol{\theta}$ is equivalent to maximising the likelihood function of $\boldsymbol{\theta}$ given random draws from $q(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}}; \boldsymbol{\phi})$:

$$
\begin{aligned}
\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n; \boldsymbol{\phi})}] \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})] \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log P(\boldsymbol{x}_n|\boldsymbol{z}_n; \boldsymbol{\theta}) + \log P(\boldsymbol{z}_n)] \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log P(\boldsymbol{x}_n|\boldsymbol{z}_n; \boldsymbol{\theta}))]
\end{aligned}
\tag{3.7}
$$

Since $q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})$ is not dependent on $\boldsymbol{\theta}$, we can get an unbiased estimate of the gradient with respect to $\boldsymbol{\theta}$ given data $\boldsymbol{x}_n$:

$$
\begin{aligned}
\bigtriangledown_{\boldsymbol{\theta}} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) &= \bigtriangledown_{\boldsymbol{\theta}} \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n; \boldsymbol{\theta}))] \\
&= \mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\bigtriangledown_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n|\boldsymbol{z}_n; \boldsymbol{\theta}))]
\end{aligned}
\tag{3.8}
$$

The gradient of this term is measured through samples from $q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})$, and in practice, a single sample is sufficient for most cases.

3.4.4 Optimise the variational parameter for ELBO

Optimising the ELBO over $\boldsymbol{\phi}$ is more challenging compared to $\boldsymbol{\theta}$, the optimisation over $\boldsymbol{\phi}$ is equivalent to minimising the gap between the actual posterior distribution $P(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$ and the approximate variational family distribution $q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})$:

$$
\begin{aligned}
\boldsymbol{\phi}^* &= \arg\max_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) \\
&= \arg\max_{\boldsymbol{\phi}} \sum_{n=1}^{N} \log P(\boldsymbol{x}_n; \boldsymbol{\theta}) - \sum_{n=1}^{N} \text{KL}[q(\boldsymbol{z}_n; \boldsymbol{\phi})||P(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\theta})] \\
&= \arg\min_{\boldsymbol{\phi}} \sum_{n=1}^{N} \text{KL}[q(\boldsymbol{z}_n; \boldsymbol{\phi})||P(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\theta})]
\end{aligned}
\tag{3.9}
$$

However, the problem for optimisation over $\boldsymbol{\phi}$ is that the gradient calculation under the $\mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}$ is difficult when $q(\boldsymbol{\mathcal{Z}}; \boldsymbol{\phi})$ is prameterised by $\boldsymbol{\phi}$. Hence, the $\bigtriangledown_{\boldsymbol{\phi}}$ can not move inside the $\mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}$ given data $\boldsymbol{x}_n$:

$$\begin{aligned}\bigtriangledown_{\boldsymbol{\phi}}\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) &= \bigtriangledown_{\boldsymbol{\phi}}\mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\phi})}[\log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n; \boldsymbol{\phi})}] \\ &\neq \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\phi})}[\bigtriangledown_{\boldsymbol{\phi}} \log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n; \boldsymbol{\phi})}]\end{aligned}$$

In order to calculate the gradient under this circumstance, many methods were proposed. In this thesis, our focus concerns sampling-based methods which can be integrated into the deep learning optimisation paradigm. The two types of sampling-based methods we discuss later in section 3.5 are: the score function and the reparameterisation technique.

3.5 Sampling Based Inference Strategy

The problem with the naive sampling algorithm is that we fail to get a non-zero gradient when calculating the following term:

$$\begin{aligned}\bigtriangledown_{\boldsymbol{\phi}}\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) &= \bigtriangledown_{\boldsymbol{\phi}}\mathbb{E}_q[\log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi})}] \\ &= \bigtriangledown_{\boldsymbol{\phi}}\mathbb{E}_q[\log P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})] - \bigtriangledown_{\boldsymbol{\phi}}\mathbb{E}_q[\log q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi})]\end{aligned}$$

When we have J number of samples $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, ..., \boldsymbol{z}^{(J)} \sim q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi})$, we always have:

$$\bigtriangledown_{\boldsymbol{\phi}} \frac{1}{J} \sum_{j=1}^{J} [\log P(\boldsymbol{x}_n, \boldsymbol{z}_n^j; \boldsymbol{\theta})] = 0$$

3.5.1 Score Function Gradient Estimator

One way to overcome the zero gradient problem is to use the policy-gradient training methods, or so-called the reinforce algorithm or the score function gradient estimator [189] as the following:

$$\bigtriangledown \log(q) = \frac{\bigtriangledown q}{q} \Rightarrow \bigtriangledown q = q \bigtriangledown \log(q) \tag{3.10}$$

The first term can be derived as (we ignore the subscript n for clarity):

$$\begin{aligned}
\bigtriangledown_{\boldsymbol{\phi}} \mathbb{E}_q[\log P(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta})] &= \bigtriangledown_{\boldsymbol{\phi}} \sum_z q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \log P(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \\
&= \sum_z \bigtriangledown_{\boldsymbol{\phi}} q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \log P(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \\
&= \sum_z q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) \log P(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \\
&= \mathbb{E}_{q(z)}[\log P(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}))]
\end{aligned}$$

The second term can be derived as (also ignore the subscript n):

$$\begin{aligned}
&\bigtriangledown_{\boldsymbol{\phi}} \mathbb{E}_q[\log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}))] = \sum_z \bigtriangledown_{\boldsymbol{\phi}}[q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}))] \\
&= \sum_z [\bigtriangledown_{\boldsymbol{\phi}} q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) + q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}))] \\
&= \sum_z [\log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) + q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \frac{\bigtriangledown_{\boldsymbol{\phi}} q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})}{q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})}] \\
&= \sum_z [\log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) + \bigtriangledown_{\phi} q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})] \\
&= \sum_z [\log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}))] + \sum_z [\bigtriangledown_{\boldsymbol{\phi}} q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})] \\
&= \sum_z [\log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}))] + \bigtriangledown_{\boldsymbol{\phi}} \sum_z q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \\
&= \sum_z [\log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}))] + \bigtriangledown_{\boldsymbol{\phi}} \mathbb{1} \\
&= \mathbb{E}_{q(z)}[\log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}))]
\end{aligned}$$

Put these two terms together, we have:

$$\begin{aligned}
\bigtriangledown_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) &= \bigtriangledown_{\boldsymbol{\phi}} \mathbb{E}_q[\log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi})}] \\
&= \mathbb{E}_{q(\boldsymbol{Z}; \boldsymbol{\phi})}[\log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi})} \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi}))] \qquad (3.11) \\
&= \mathbb{E}_{q(\boldsymbol{Z}; \boldsymbol{\phi})}[\Re_{\boldsymbol{\theta}, \boldsymbol{\phi}}(\boldsymbol{x}_n, \boldsymbol{z}_n) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi}))]
\end{aligned}$$

Now the gradient is inside the expectation; thus we can have J samples $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, ..., \boldsymbol{z}^{(J)} \sim$

$q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi})$ and get an unbiased estimator with Monte Carlo sampling:

$$\begin{aligned}&\mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n,\boldsymbol{z}_n)\bigtriangledown_{\boldsymbol{\phi}}\log(q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi}))]\\&\approx\frac{1}{J}\sum_{j=1}^{J}[\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n,\boldsymbol{z}_n^{(j)})\bigtriangledown_{\boldsymbol{\phi}}\log(q(\boldsymbol{z}_n^{(j)}|\boldsymbol{x}_n;\boldsymbol{\phi}))]\end{aligned}\tag{3.12}$$

The $\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n,\boldsymbol{z}_n)$ term here is essentially the reward function seen in reinforcement learning. When a sample $\boldsymbol{z}_n$ have high reward $\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n,\boldsymbol{z}_n)$, the probability of sample $\boldsymbol{z}_n$ occur again is increased by moving along the gradient of $\bigtriangledown_{\boldsymbol{\phi}}\log(q(\boldsymbol{z}_n^{(j)}|\boldsymbol{x}_n;\boldsymbol{\phi}))$. The score function gradient is generally applicable regardless of the distribution over the approximated latent variable $q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})$; however, this generality comes with a cost of high variance as a result of the "black-box" reward. In practice, we apply a control variable B inside the expectation to reduce the variance, as the following:

$$\begin{aligned}&\bigtriangledown_{\boldsymbol{\phi}}\,\mathrm{ELBO}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x}_n)\\&=\mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[(\log\frac{P(\boldsymbol{x}_n,\boldsymbol{z}_n;\boldsymbol{\theta})}{q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi})}-B)\bigtriangledown_{\boldsymbol{\phi}}\log(q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi}))]\\&=\mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[(\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n,\boldsymbol{z}_n)-B)\bigtriangledown_{\boldsymbol{\phi}}\log(q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi}))]\\&\approx\frac{1}{J}\sum_{j=1}^{J}[(\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n,\boldsymbol{z}_n^{(j)})-B)\bigtriangledown_{\boldsymbol{\phi}}\log(q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi}))]\end{aligned}\tag{3.13}$$

The control variable B can also be estimated with another neural network [6]. To show that the control variable has no impact on the gradient estimator, we have:

$$\begin{aligned}&\mathbb{E}_{q(\boldsymbol{\mathcal{Z}};\boldsymbol{\phi})}[(B)\bigtriangledown_{\boldsymbol{\phi}}\log(q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi}))]\\&=(B)\sum_{z}q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})\bigtriangledown_{\boldsymbol{\phi}}\log(q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi}))\\&=(B)\sum_{z}\bigtriangledown_{\boldsymbol{\phi}}q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi})\\&=(B)\bigtriangledown_{\boldsymbol{\phi}}(\sum_{z}q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi}))\\&=(B)\bigtriangledown_{\boldsymbol{\phi}}\,\mathbb{1}=0\end{aligned}$$

3.5.2 Reparameterisation Technique

Another useful method is the reparameterisation technique, which is developed based on the assumption that we can find a deterministic, differentiable function transformer g with parameter $\boldsymbol{\phi}$ that maps a random noise $\boldsymbol{\varepsilon}$ to the latent variable $\boldsymbol{\mathcal{Z}}$:

$$\boldsymbol{\varepsilon} \sim \mathbb{U}; \boldsymbol{\mathcal{Z}} = g(\boldsymbol{\varepsilon}, \boldsymbol{\phi})$$

The gradient can be calculated through sampling:
Draw samples:

$$\boldsymbol{\varepsilon}^{(1)}, \boldsymbol{\varepsilon}^{(2)}, ..., \boldsymbol{\varepsilon}^{(J)} \sim \mathbb{U}[0, 1]$$

And we have the gradient calculated as follows:

$$\begin{aligned}
&\bigtriangledown_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) \\
&= \bigtriangledown_{\boldsymbol{\phi}} \mathbb{E}_{q(\boldsymbol{\mathcal{Z}})}[\log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n | \boldsymbol{x}_n; \boldsymbol{\phi})}] \\
&= \bigtriangledown_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\varepsilon} \sim \mathbb{U}}[\log \frac{P(\boldsymbol{x}_n, g(\boldsymbol{\varepsilon}, \boldsymbol{\phi}); \boldsymbol{\theta})}{q(g(\boldsymbol{\varepsilon}, \boldsymbol{\phi}) | \boldsymbol{x}_n; \boldsymbol{\phi})}] \\
&= \mathbb{E}_{\varepsilon \sim \mathbb{U}}[\bigtriangledown_{\boldsymbol{\phi}} \log \frac{P(\boldsymbol{x}_n, g(\boldsymbol{\varepsilon}, \boldsymbol{\phi}); \boldsymbol{\theta})}{q(g(\boldsymbol{\varepsilon}, \boldsymbol{\phi}) | \boldsymbol{x}_n; \boldsymbol{\phi})}] \\
&\approx \frac{1}{J} \sum_{j=1}^{J} [\bigtriangledown_{\boldsymbol{\phi}} \log \frac{P(\boldsymbol{x}_n, g(\boldsymbol{\varepsilon}^{(j)}, \boldsymbol{\phi}); \boldsymbol{\theta})}{q(g(\boldsymbol{\varepsilon}^{(j)}) | \boldsymbol{x}_n; \boldsymbol{\phi})}] \\
&= \frac{1}{J} \sum_{j=1}^{J} [\bigtriangledown_{\boldsymbol{\phi}} \log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n^{(j)}; \boldsymbol{\theta})}{q(\boldsymbol{z}_n^{(j)} | \boldsymbol{x}_n; \boldsymbol{\phi}))}]
\end{aligned}$$

The reparameterisation technique offers an unbiased estimator, like the score function but with less variance. In practice, a single sample of $\boldsymbol{z}_n$ is often sufficient. The limitation of the reparameterisation technique is that it only allows continuous latent variable z. However, as suggested later in section 3.6, the reparameterisation technique can be extended to use for discrete latent variable inference. In this thesis, we adopt the reparameterisation technique as it is empirically reported to be more stable than the score function estimator [99, 100].

3.5.3 Score Function V.S. Reparameterisation Technique

In this section, we briefly discuss some insights on why the reparameterisation technique produces better results than the sore function; recall that:

$$\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n, \boldsymbol{z}_n) = \log \frac{P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})}{q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi})}$$

For the score function, we have:

$$\bigtriangledown_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) = \mathbb{E}_{z\sim q(\boldsymbol{z})}[\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n, \boldsymbol{z}_n) \bigtriangledown_{\boldsymbol{\phi}} \log(q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi}))]$$

For the reparameterisation technique, we have the following:

$$\bigtriangledown_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) = \mathbb{E}_{\varepsilon\sim\mathbb{N}(0,1)}[\bigtriangledown_{\boldsymbol{\phi}} \Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n, g(\boldsymbol{\varepsilon}; \boldsymbol{\phi}))]$$

The reparameterisation technique allow gradient differentiation over the reward $\Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n, \boldsymbol{z}_n)$, which contain information on both $P(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})$ and $q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi})$; on comparison with only information about $q(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\phi})$ in the score function. This thus grants the reparameterisation technique a better knowledge about the latent variable and the observed variable compared to the score function.

3.6 Advanced Techniques for Machine Learning

3.6.1 Gumbel-softmax

Gumbel-softmax [3, 124] extends the reparameterisation technique [4] and allow it to be used for discrete latent variable. It is developed based on the "Gumbel-max" trick [190], which can be used to sample variables following a discrete distribution from a continuous distribution. To sample from a K categorical distribution, follows:

$$P(\boldsymbol{z}_k = 1; \alpha) = \frac{\alpha_k}{\sum_{j=1}^{K} \alpha_j}$$

Where $\boldsymbol{z} = [0, 0, ..., 1, ..., 0]$ is a one-hot vector with K dimensions. Then we can draw samples from $P(\boldsymbol{z}; \alpha)$ by:

- Draw independent Gumbel noise $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_1, .., \boldsymbol{\varepsilon}_K$

  $\boldsymbol{\varepsilon}_k = -\log(-\log(\boldsymbol{\mu}_k)); \boldsymbol{\mu}_k \sim \mathbb{U}[0, 1]$

- Add $\boldsymbol{\varepsilon}_k$ to $\log \boldsymbol{\mu}_k$, getting the softmax

  $\boldsymbol{z}_k = \text{softmax}(\frac{\log \alpha_k + \boldsymbol{\varepsilon}_k}{\tau})_{\boldsymbol{z}_k}$

Here, τ is a temperature term, which interpolates between discrete one-hot-encoded categorical distributions and continuous categorical densities, as shown in Figure 3.4. In sub-figure (a), for low temperatures (such as $\tau = 0.1$, $\tau = 0.5$), the expected value of a Gumbel-Softmax random variable approaches the expected value of a categorical random variable with the same logits. As the temperature increases (such as $\tau = 1.0$, $\tau = 10.0$), the expected value converges to a uniform distribution over the categories. In sub-figure (b), samples from Gumbel-Softmax distributions are identical to samples from a categorical distribution as τ approaching 0. at higher temperatures, Gumbel-Softmax samples are no longer one-hot and become uniform as τ approaches ∞. By using the Gumbel-softmax, we can calculate the gradient for parameterised technique:

$$\begin{aligned}
&\bigtriangledown_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_n) = \mathbb{E}_{q(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\phi})}[\bigtriangledown_{\boldsymbol{\phi}} \Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}_n, \boldsymbol{z}_n)] \\
&\approx \mathbb{E}_{\boldsymbol{\varepsilon}\sim\text{Gumbel}}[\bigtriangledown_{\boldsymbol{\phi}} \Re_{\boldsymbol{\theta},\boldsymbol{\phi}}(\text{softmax}(\frac{\log \alpha_k + \boldsymbol{\varepsilon}_k}{\tau}))]
\end{aligned}$$

The 'Gumbel-softmax' technique empirically has low variance, but it is a biased estimator for the gradient, and we explore the 'Gumbel-softmax' technique in Chapter 7.

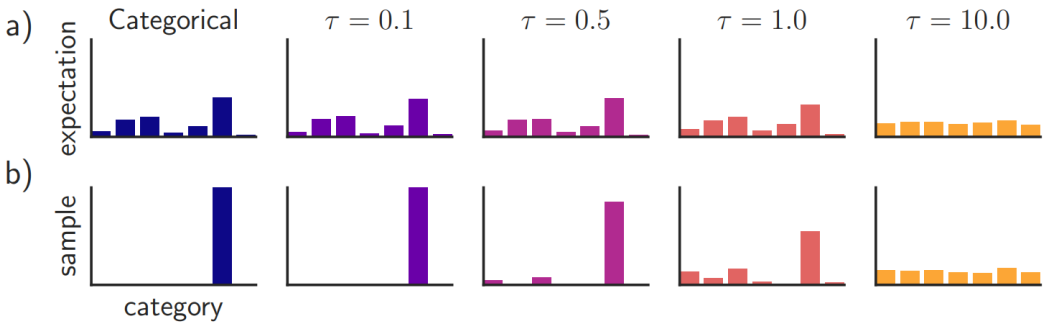

Figure 3.4: Chapter 3: Demonstration of how temperature τ impacts the sampled distribution when the 'Gumbel-softmax' technique is used. (Figure taken from [3].)

3.6.2 Straight Through Gradient

Straight-through gradient technique [191] is proposed to 'back-propagate' stochastic neurons with discrete distribution and is considered as an alternative approach to high variance gradient from the score function method introduced in section 3.5.1. From a probabilistic perspective, the sampling process is replaced with the expectation of the distribution in its first-order form. With the straight-through gradient technique, we are able to handle non-differentiable gradients over discrete distribution, such as binary variables, as shown in Figure 3.5.

During the forward pass, the straight through gradient estimator, use the threshold function to estimate the expectation of the discrete distribution and use this expected value as the input; and during the backward pass, the straight through gradient estimator use the gradient of the original variable. We explore the straight-through gradient estimator as a method to obtain discrete latent variables and for reducing variance in Chapter 7.

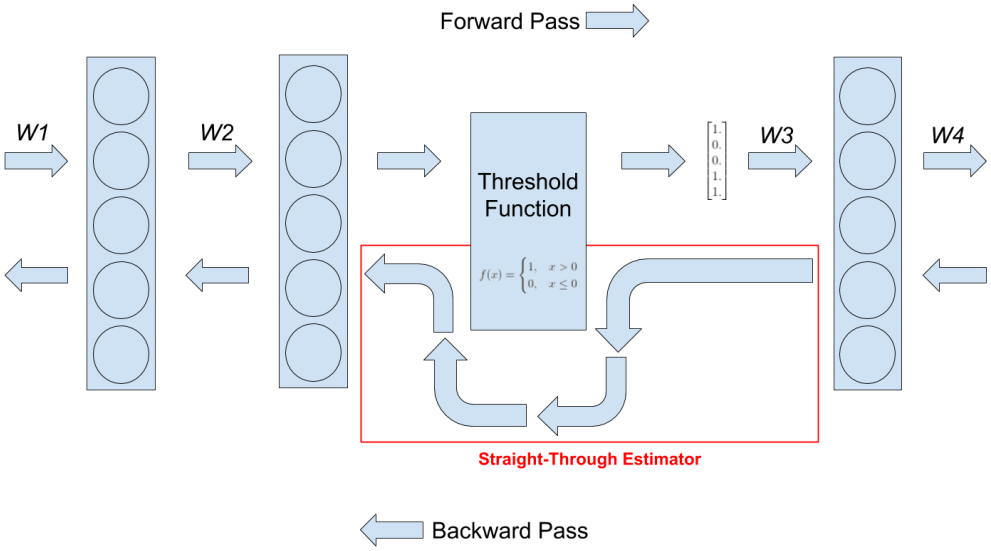

Figure 3.5: Chapter 3: A visual demonstration of the straight-through gradient of representation captured in different neural network layers. Source from Tech Blog

3.6.3 Approximated Bayesian Computation Method

Approximate Bayesian computation constitutes a class of computational methods rooted in Bayesian statistics that aims to estimate the posterior distributions of

model parameters. In model-based statistical inference, the likelihood function plays a central role since it expresses the probability of the observed data under a particular statistical model. This allows us to handle uncertainty and perform model selection, given data. However, an analytical formula of the likelihood function might be intractable for complex models, such as deep neural networks. In this way, approximate Bayesian computation methods offer an approach to widen the realm of models to which statistical inference can be applied. Approximate Bayesian computation methods are mathematically well-founded, but they inevitably make assumptions and approximations whose impact must be carefully assessed. In the scope of deep learning, one particular well-known instance is the **Monte Carlo Dropout** (MCD) [192, 193].

The MCD use one particular stochastic regularisation technique, called Dropout [194], to estimate the posterior distribution of deep learning models. The Dropout technique is initially proposed to solve over-fitting problems in deep learning. With a proper setup of the Dropout technique in deep neural network models, the models can be considered as **Bayesian neural networks** (BNNs) [7]. With Dropout turned on, each forward pass through the network can be considered as a single sample of the posterior distribution [193], as shown in Figure 3.6. MCD method is used in Chapter 4 as one of the Bayesian deep learning methods we explored in the educational domain, and as one of the baseline algorithms in Chapter 5.

3.7 Methodology and Justifications

In this section, we briefly describe our methodology in the form of a general framework, which we used for the implementation of our research included in this thesis. As shown in section 3.4.3 and 3.4.4, the model parameter and variational parameter need to be optimised individually with coordinate descent, as suggested in [28, 29]. Thankfully with the development of amortised inference [4–6], as introduced in section 3.4.1, the coordinate descent optimisation can be bypassed when we use a surrogate function to approximate the variational parameter. Under the **variational**

[7] For more about Bayesian neural networks, please see Chapter 5.

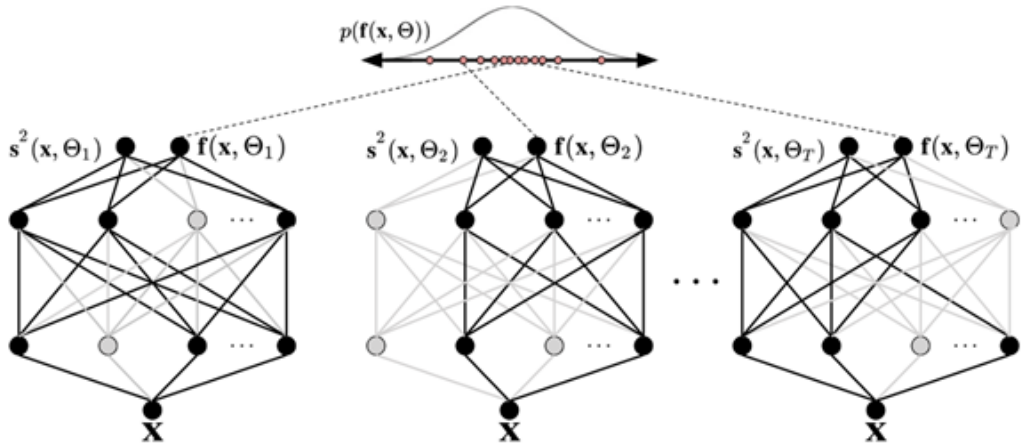

Figure 3.6: Chapter 3: A visual example of representation captured in different neural network layers. Source from Amazon AWS

autoencoder (VAE) framework [4], the variational parameter and model parameter can be jointly optimised together with standard stochastic gradient descent algorithms, which are widely adopted in the deep learning community [1]. Although this does not equivalent to performing coordinate descent, however, the quality of the resulting models from the VAE framework and their results can be regarded as a good approximation. In this thesis, we adopt this surrogate approximation as the general framework and methodology used for the implementation of our various NLP applications.

Based on this general framework, DLVMs can be interchangeably viewed from the lens of an encoder-decoder framework, and we continue using encoder and decoder in this thesis when explaining our NLP applications. An illustration of the framework is presented in Figure 3.7 below. The encoder and decoder correspond to the inference and learning in the DLVMs, respectively. In later Chapters (4, 5, 6 and 7), we build our NLP applications generally follow this framework. In comparison, the standard deep learning model can be illustrated in Figure 3.8, with no explicit latent variable and no inference process.

When comparing these two frameworks, we can identify why DLVMs can naturally address the concerns on robustness, trustworthiness, interpretability and explainability, which justified the link between our methodology and our research motivation (in section 1.1):

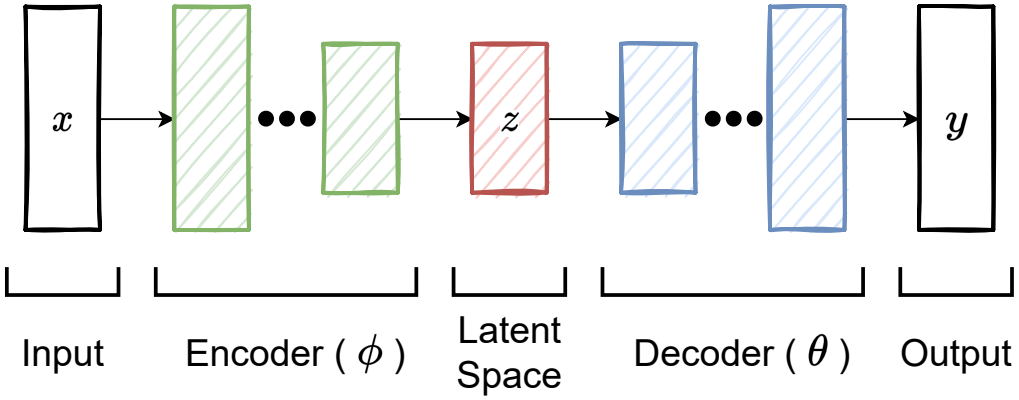

Figure 3.7: Chapter 3: An illustration of the general framework for building NLP applications in this thesis, based on the VAE learning and inference framework [4–6].)

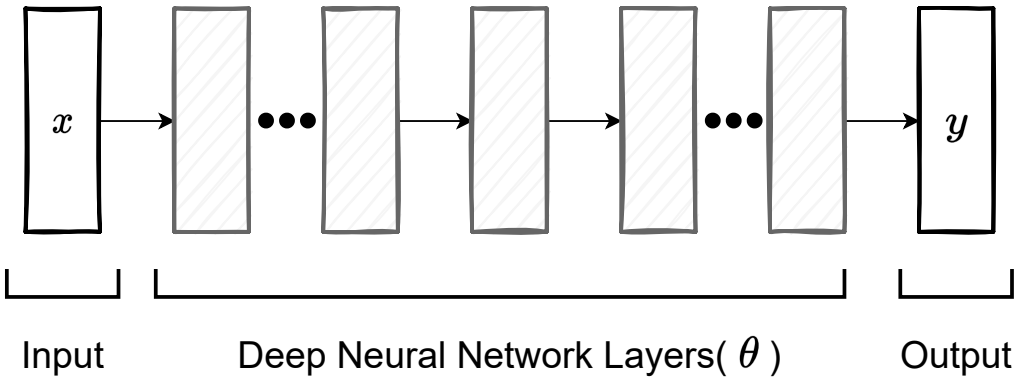

Figure 3.8: Chapter 3: An illustration of the general framework for building NLP applications with deep learning.)

- **Robustness**: Compared with deep learning models, the variations of representation distribution in the latent space of DLVMs during training enhance model robustness. Later in Chapter 4, when comparing with a non-probabilistic deep learning model (using a very strong RNN model architecture as the baseline), we show that deep latent variable models grant more robust learning during training with random weight initialisation. This further enhances the justification that deep latent variable models enhance robustness during training.

- **Trustworthiness**: Compared with deep learning models, DLVMs naturally handle uncertainty with probabilistic modelling when making predictions. Later in Chapter 4 and 5, deep latent models allow an additional metric, entropy, as an uncertainty quantification measurement; together with the standard deep

learning metrics as a part of the model outputs. This uncertainty quantification measurement contains more information on the confidence of our model and improves the trustworthiness of deep learning models, especially for decision-making purposes.

- **Interpretability**: Compared with deep learning models, the encoder in DLVMs allows model interpretability from input space to its associated latent space. The latent space naturally encodes the cause of the input space. Later in Chapter 6, we learn a direct mapping from the premise and hypothesis input text pair in the Stanford natural language inference tasks to the explanation semantic space. And in Chapter 7, we learn a direct mapping from the source sentence to the unseen target sentence for semi-supervised learning, where the unseen target is a sequence of probability across the distribution of words in the directory. In both of the applications, deep latent variables models, in principle, enhance the interpretability of the models.

- **Explainability**: Compared with deep learning models, shared latent space in DLVMs allows individual points to support the explainability of each other. Later in Chapter 6, by interpolating through the latent space, we are able to create multiple semantic equivalent explanations, in which each explanation support explainability of each other else. And in Chapter 7, we are able to recover the unobserved target text from the source text in the form of a discrete sequence form, which the latent target sequence ensures its explainability. In both of the applications, deep latent variables models, in principle, enhance the explainability of the models.

The distinction between interpretability and explainability continues to be a subject of disagreement across various research fields. The rise of deep learning techniques, known for their 'black-box' nature, has spurred a new area of research in explainable AI, which aims to make AI models more transparent and hence more explainable. Despite several attempts to provide definitions and taxonomies for these terms from the perspective of explainable AI [195, 196], a universally accepted view is yet to be achieved.

In the scope of this thesis, our definitions of interpretability and explainability draw largely from the perspective outlined in [195]. Interpretability, in this context, refers to the characteristics of the model and involves using latent variables to understand the underlying causes of the observed data. On the other hand, explainability focuses on the human aspect and concerns how the model outputs assist and support human understanding of the results.

Despite the ongoing debate surrounding these terms, it is widely acknowledged that the recent advancements in deep learning techniques have heightened the importance of interpretability and explainability in AI models (e.g. with the surprising performance of large-scale language models). As AI increasingly permeates various aspects of our lives, it is imperative to develop models that not only produce accurate results but also provide insights into their workings (some of the works in Chapter 6 and 7 of this thesis tried to address this aspect). This is particularly important for applications in critical domains such as healthcare, finance, and education, where the consequences of errors can be significant.

CHAPTER 4

Exploring Bayesian Deep Learning for Text Classification: A Case Study of AI in Education

Prologue

In Chapter 3, we have provided a basic introduction to Bayesian deep learning with a particular focus on **deep latent variable models** (DLVMs), **variational inference** (VI) algorithms and some associated advanced techniques for probabilistic machine learning. This knowledge will be vital to understanding the contributions we develop in this and subsequent chapters. In this chapter, we present a contribution in exploring Bayesian deep learning for a text classification task, in the form of a case study in education domain applications, on the task of identifying urgent intervention needs in **Massive Open Online Courses** (MOOCs) forums.

This chapter presents the first research on how Bayesian deep learning can be applied to a text-based learning analytics application. Here, the aim is to predict instructor intervention needs in the educational domain and is studied as a **natural language processing** (NLP) problem. We explore, for the first time, not only one but two Bayesian deep learning methods on the task of classifying learners'

posts based on their urgency, namely **Monte Carlo Dropout** (MCD) and VI. These two methods subsequently convert our original deep neural networks into a **Bayesian neural network** (BNN) and a DLVM. We empirically show the benefits of Bayesian deep learning for this task and discuss the differences between our two Bayesian approaches (BNN and DLVM). We achieve competitive results in this task and obtain a lower variance when training with small-size data samples. We apply this approach to text-based processing on posts in MOOCs - a source generally available across all MOOC providers. Thus our approach is widely applicable - generalisable to foresee instructors' intervention needs in MOOCs and to support the elusive problem of MOOC dropout.

Declaration: This chapter is based on the following publication:

Yu, J., Alrajhi, L., Harit, A., Sun, Z., Cristea, A.I. and Shi, L., 2021, June. ***Exploring Bayesian Deep Learning for Urgent Instructor Intervention Need in MOOC Forums.*** *In International Conference on Intelligent Tutoring Systems (pp. 78-90). Springer, Cham.* (Core A Ranked Conference, Accepted for Full Paper)

This chapter is presented largely as accepted, although referencing and notation have been altered and cross-referencing added for consistency across this thesis. Some stylistic changes have been made for consistency. The majority of the text is verbatim, with some minor wording and formatting changes.

4.1 Introduction

MOOCs are well-known for their high dropout rates [197, 198]. Whilst learners may discuss their problems in the forums before actually dropping out, the sheer volume of posts renders it almost impossible for instructors to address them. Thus, many of these urgent posts are overlooked or discarded. Hence, a few researchers proposed [199, 200] automated machine learning models for need prediction based on learners' posts in MOOC forums. Such an approach would allow instructors to

identify learners who require urgent intervention, in order to, ultimately, prevent potential dropouts (see our following research, where we have shown only 13% of learners passing urgent intervention messages complete the course [201]).

More recently, techniques for applying deep neural networks to interpret texts from the educational field have emerged [202], including identifying learners' needs based on their posts in forums [203–205]. Despite their success, standard deep learning models have limited capability to incorporate uncertainty. Another challenge is that post data is notoriously imbalanced, with urgent posts representing a very low percentage of the overall body of posts - the proverbial 'needle in the haystack'. The imbalanced nature of the data tends to make a neural network overfit and ignore the urgent posts, resulting in a large variance in model predictions.

To address the above two challenges, we apply Bayesian probabilistic modelling to standard neural networks. Recent advances in Bayesian deep learning offer a new theory-grounded methodology to apply probabilistic modelling using neural networks. This important approach is yet to be introduced in the **Educational Data Mining** (EDM) and **Learning Analytics** (LA) field.

4.2 Related Work

4.2.1 Urgent Intervention Need in MOOCs

Detection of the need for urgent instructor intervention is arguably one of the most important challenges in MOOC environments. The problem was first proposed and tackled [206] as a binary prediction task on instructors' intervention histories based on statistical machine learning. A follow-up study [200] proposed the use of $L1$ regularisation techniques during the training and used an additional feature about the type of forum (thread), besides the linguistic features of posts. Another study [199] tried to build a generalised model, using different shallow ML models with linguistic features extracted by NLP tools, metadata and term frequency. In general, this problem was attempted based on two types of data formats: text-only [204, 205, 207–209] or a mixture of text and post features [199, 200]. From a machine learning perspective, both traditional machine learning methods [199, 200, 207] and

deep learning based methods [204, 205, 208, 209] were proposed and explored; with more recent studies being in favour of deep neural network-based approaches [202]. However, one critical problem for deep neural networks is that they do not offer a robust estimation of the prediction values. Also, we can not perform efficient learning on small sample size data. Thus, in this paper, we explore the benefits of Bayesian deep learning to predict learners who require urgent interventions from an instructor. We use text-only features in our study, as it is the first study to explore the benefits of this new approach, and we leave future optimisation for further work.

To the best of our knowledge, this is the first study of Bayesian deep learning methods for learners' urgent intervention need classification. Our research sheds light on a new direction for other researchers in the fields of EDM and LA.

4.2.2 Bayesian Neural Networks

Modern neural networks are self-adaptive models with a learnable parameter set $\boldsymbol{\theta}$. In a supervised learning setting, given data $\mathcal{D} = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{N}$, we aim to learn a function through the neural network $\boldsymbol{y} = f_{\text{NN}}(\boldsymbol{x})$ that maps the inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$. A point estimation version of the model parameter set $\boldsymbol{\theta}^*$ is obtained through a gradient-based optimisation technique and with a respective cost function.

Bayesian neural networks (BNNs) [210–212], alternatively, consider the probability of the distribution over the parameter set $\boldsymbol{\theta}$ and introduce a prior over the neural network parameter set $P(\boldsymbol{\theta})$. The posterior probability distribution $P(\boldsymbol{\theta}|\mathcal{D})$ is learnt in a data-driven fashion through Bayesian inference. BNNs grant us a distribution over the parameter set $\boldsymbol{\theta}$ other than a static point estimation, which allows us to model uncertainty in the neural network prediction. In the prediction phase, we sample model parameters from the posterior distribution i.e. $\boldsymbol{\theta}^{(j)} \sim P(\boldsymbol{\theta}|\mathcal{D})$ and predict results with $f_{\text{NN}}^{\boldsymbol{\theta}^{(j)}}(\boldsymbol{x})$ for the corresponding $\boldsymbol{y}$. We marginalise the $\boldsymbol{\theta}$ samples and obtain an expected prediction. Due to the complexity and non-linearity of neural networks, exact inference for BNNs is rarely possible; hence various approximation inference methods have been developed [193, 213–215]. The most widely adopted approximation method is the **Monte Carlo Dropout** (MCD) [193], with applications in natural language processing, data analytics, and computer vi-

sion [18, 192, 216–218]. In this paper, we adopt the same idea and use MCD [193] to approximate the neural network as a BNN.

4.2.3 Variational Inference

Variational inference (VI) [28, 93, 185] is a general framework for Bayesian statistical modelling and inference under a maximum likelihood learning scheme. It introduces an unobserved random variable as the generative component to model the probabilistic uncertainty. Given fully observed data $\mathcal{D} = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^N$, we consider them as random variables and use capital letters $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ to represent them. The unobserved random variable introduced with VI is denoted as $\boldsymbol{\mathcal{Z}}$ and passes the information from $\boldsymbol{\mathcal{X}}$ to $\boldsymbol{\mathcal{Y}}$. It can be marginalised out with Bayes' rule as:

$$P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}; \boldsymbol{\theta}) = \sum_{\boldsymbol{\mathcal{Z}}} P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{Z}}; \boldsymbol{\theta}) = P(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{X}}; \boldsymbol{\theta}) P(\boldsymbol{\mathcal{Z}}|\boldsymbol{\mathcal{X}}; \boldsymbol{\theta}) \tag{4.1}$$

Under a mean-field assumption [188] over the unobserved random variable $\boldsymbol{\mathcal{Z}}$, we can factorise it as local random variables:

$$P(\boldsymbol{\mathcal{Z}}; \boldsymbol{\theta}) = P(\boldsymbol{z}_1, ..., \boldsymbol{z}_N; \boldsymbol{\theta}) = \prod_{i=1}^{N} P(\boldsymbol{z}_i; \boldsymbol{\theta}) \tag{4.2}$$

Hence for each pair of data, $\boldsymbol{x}$ and $\boldsymbol{y}$, the maximum likelihood learning method delivers the following objective with respect to θ:

$$\log P(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \log \int_{\boldsymbol{z}} P(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}) d\boldsymbol{z} \tag{4.3}$$

Given observed data $\mathcal{D} = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^N$, we can not directly model the distribution of unobserved $\boldsymbol{z}$ and hence the probability distribution $P(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$ is intractable for data-driven models, such as neural networks. With VI, an additional variational parameter ϕ with its associated variational family distribution $q(\boldsymbol{z}; \boldsymbol{\phi})$ is introduced, to approximate the real probability $P(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$. During the learning process, we minimise the distance between $q(\boldsymbol{z}; \boldsymbol{\phi})$ and $P(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$ through the KullbackLeibler divergence, a term that measures the distance between two probability distributions. Hence, the learning of the intractable probability distribution problem is converted

to an optimisation problem over the **evidence lower bound** (ELBO), where $\mathbb{D}_{KL}$ refers to the KullbackLeibler divergence:

$$\log P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) \geq \mathcal{L}(\text{ELBO}) = \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\phi})}[\log P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})] - \mathbb{D}_{KL}[q(\boldsymbol{z};\boldsymbol{\phi})||p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})] \tag{4.4}$$

VI was initially developed to solve a specific class of modelling problems where conditional conjugacy is presumed, and variational parameter $\boldsymbol{\phi}$ is updated through closed-form coordinate ascent [219]. However, conditional conjugacy is not practical in most real-world problems; thus, further advancements [4, 9, 92, 186, 187] extend VI to large-scale datasets and non-conjugate models.

4.3 Methodology

In this section, we first introduce the baseline model built based on recurrent neural networks (RNNs) and an attention mechanism. Then we present our two approaches for applying Bayesian deep learning with our baseline model: 1) Monte Carlo Dropout and 2) Variational Inference.

4.3.1 Baseline Deep Learning Model

In this section, we first introduce our non-Bayesian model, which serves as our baseline model. The model consists of three different components: an embedding layer, a two-layer recurrent neural network (RNN), and a prediction layer. We use attention based on the output of the RNNs to create a contextual representation over the RNN hidden outputs and then concatenate it with the last layer of RNN outputs. The model architecture is presented in Figure 4.1.

Given the data $\mathcal{D} = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{N}$, where each sentence $\boldsymbol{x}_i$ consists of a sequence of tokens $\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, ..., \boldsymbol{x}_i^s$ where s denotes the sequence length. For our baseline model, given a sentence $\boldsymbol{x}_i$, we first pass it through the embedding layer and obtain a sequence of word embeddings:

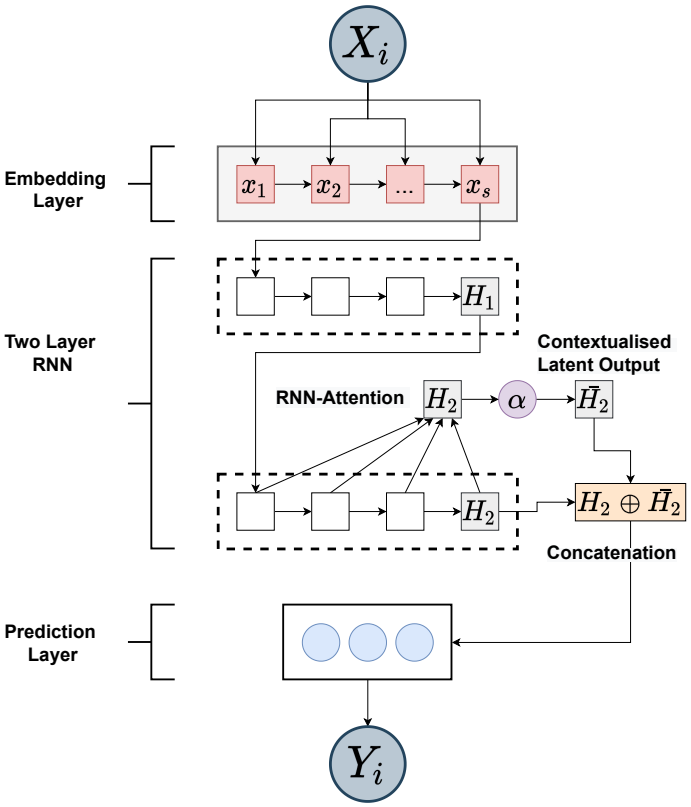

Figure 4.1: Chapter 4: A visual demonstration of model architecture for baseline model (symbol $\oplus$ refers to the concatenation operation).

$$E = (\text{emb}(\boldsymbol{x}_i^1), \text{emb}(\boldsymbol{x}_i^2), ..., \text{emb}(\boldsymbol{x}_i^s)) \tag{4.5}$$

Where emb is the embedding function we used for our experiment with d dimensions. Here, $\boldsymbol{x}_i^m$ denotes the m^{th} word in the sentence $\boldsymbol{x}_i$. For the initial sentence $\boldsymbol{x}_i \in \mathbb{R}^{s \times 1}$, we derive a sentence $\boldsymbol{x}_i \in \mathbb{R}^{s \times d}$ after the embedding layer. Then we feed this as a sequence input through a two-layer **long-short-term memory** (LSTM) model as in [205]. The initial hidden state $\boldsymbol{h}_0$ is set to 0, and we calculate the sequence of hidden states as:

$$\boldsymbol{h}_m = \text{LSTM}(\boldsymbol{h}_{m-1}, \boldsymbol{x}_i^m) \tag{4.6}$$

Where we have $m = 1, ..., s$. The last layer of hidden states provides a sequence output $H \in \mathbb{R}^{s \times h}$, where h here represents the hidden dimension size. In order to utilise the contextual information through the LSTM encoding process, we calculate

the attention score α based on the last hidden state outputs H_2 ($H_2 = \boldsymbol{h}_s$) and each hidden state in the sequence of H, as:

$$\alpha_m = \frac{H_2 * \boldsymbol{h}_m}{\sum_{m=1}^{s} H_2 * \boldsymbol{h}_m} \tag{4.7}$$

Then we calculate the contextual $\bar{H}_2$ as:

$$\bar{H}_2 = \sum_{m=1}^{s} \alpha_m \boldsymbol{h}_m \tag{4.8}$$

Finally, we concatenate them and feed them through a fully connected layer with the output dimension equal to the number of classes for our task. This fully connected layer is represented as a prediction layer in Figure 4.1.

4.3.2 Model Uncertainty with Monte Carlo Dropout

In this section, we present how to convert our baseline model into a BNN. With MCD [193], we only need to use the dropout technique [194] right after each layer containing the parameter set $\boldsymbol{\theta}$. In our case, we add a dropout layer after the first and second LSTM layers, as well as after the fully connected layer, which takes the input as the concatenation of $\bar{H}_2$ and H_2.

Compared with the standard dropout technique, which works as a regularisation technique in the training phase only, the MCD technique requires the dropout layer to be activated in both the training and testing phases. This allows the standard neural network model to work as a BNN [193]. Each dropout works as a sample of $\boldsymbol{\theta}$ from its probabilistic distribution space and hence allows us to measure the uncertainty of the model, as shown in Figure 4.2. In the testing phase, we predict the output through sampling M times [193] and the expectation of $\boldsymbol{y}$ can be calculated as:

$$\mathbb{E}(\boldsymbol{y}|\boldsymbol{x}) \approx \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{f}_{\mathrm{NN}}^{\boldsymbol{\theta}_i}(\boldsymbol{x}) \tag{4.9}$$

We use this expectation as the final logits value, and in our experiments, we use a total sample M of 50 as in [18].

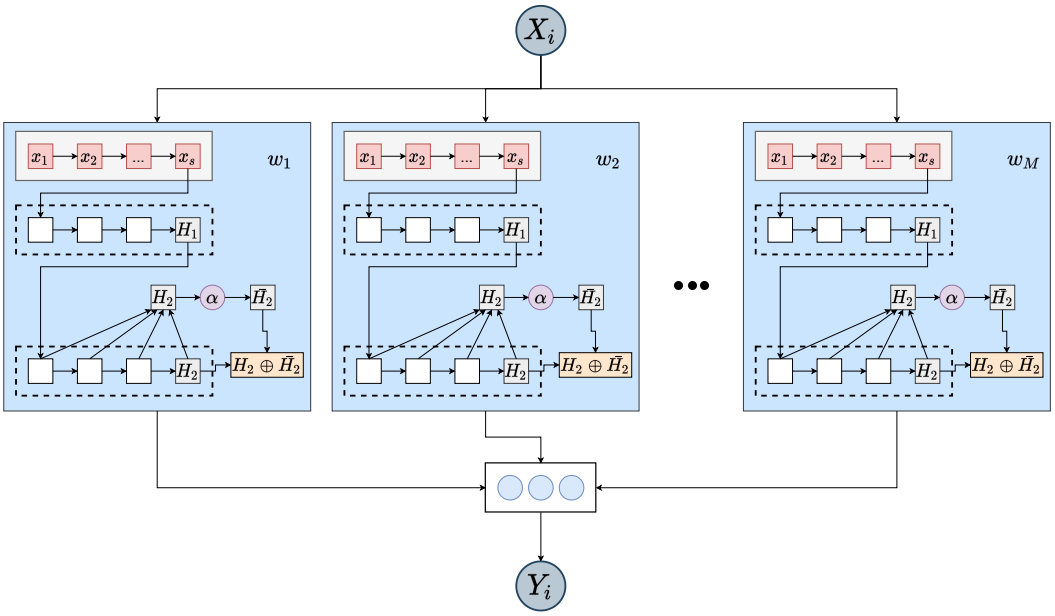

Figure 4.2: Chapter 4: A visual demonstration of the BNN with Monte Carlo Dropout in the test phase. We run the model for M times for M different prediction results and then calculate their average as the prediction layer output.

4.3.3 Model Uncertainty with Variational Inference

As discussed in the section 4.2.3, VI introduces an additional random variable $\boldsymbol{z}$ with probability distribution $q(\boldsymbol{z};\boldsymbol{\phi})$ to the original model. This variational family $q(\boldsymbol{z};\boldsymbol{\phi})$ here approximates the posterior distribution $P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ as $q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y};\boldsymbol{\phi})$. The model architecture is presented in Figure 4.3. Following [134], we define $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$ as:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{y}), \mathrm{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{y}))) \tag{4.10}$$

We have:

$$\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{y}) = l_1(\boldsymbol{\pi}_{\boldsymbol{\phi}}) \tag{4.11}$$

$$\log \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{y}) = l_2(\boldsymbol{\pi}_{\boldsymbol{\phi}}) \tag{4.12}$$

Where:

$$\boldsymbol{\pi}_{\boldsymbol{\phi}} = g_{\phi}(H_2, f_{\mathrm{y}}(\boldsymbol{y})) \tag{4.13}$$

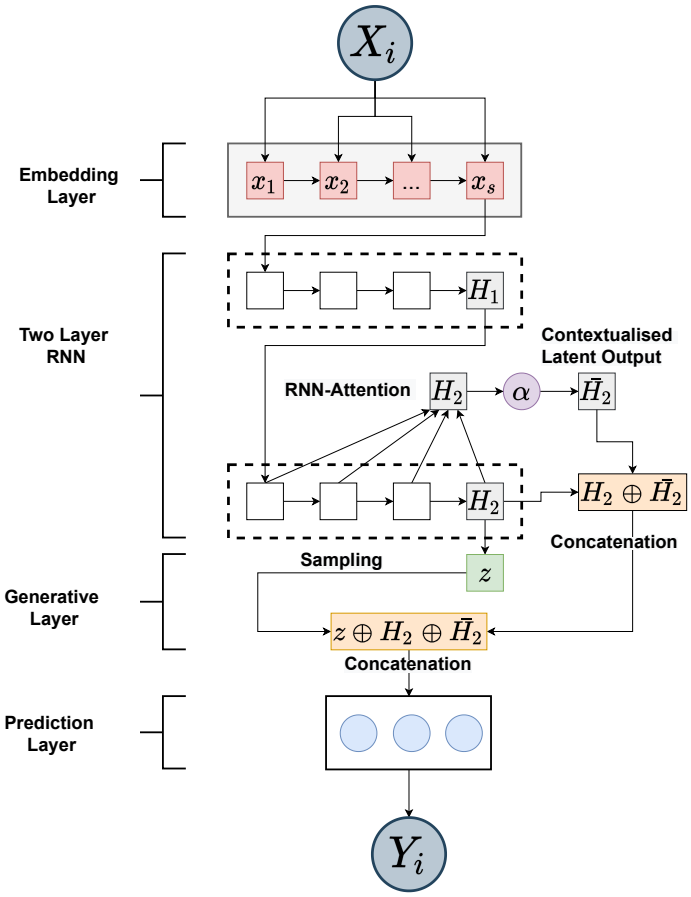

Figure 4.3: Chapter 4: A visual demonstration of our DLVM architecture using the VI method, based on our baseline model.

Where $f_{\mathrm{y}}(\boldsymbol{y})$ is an affine transformation from output $\boldsymbol{y} \in \mathbb{R}^K$, where K representing the number of class, to a vector space size $s_y \in \mathbb{R}^h$. The H_2 is the final latent state output of the second LSTM network layer as stated in Figure 4.1. The latent variable $\boldsymbol{z} \in \mathbb{R}^h$ can be reparameterised as $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}$, known as the 'reparameterisation trick' [220] with sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{I})$. For the conditional distribution $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$, we can model it as:

$$P(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\boldsymbol{x}))) \tag{4.14}$$

Where we have:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}) = l_3(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \tag{4.15}$$

$$\log \boldsymbol{\sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}) = l_4(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \tag{4.16}$$

And:

$$\boldsymbol{\pi_\theta} = g_{\boldsymbol{\theta}}(H_2) \tag{4.17}$$

Where l_1, l_2, l_3 and l_4 are four affine transformation functions. Since both $P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ and $q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y};\boldsymbol{\phi})$ are multivariate Gaussian distributions, this allows us to have a closed-form solution for the **KullbackLeibler** (KL) divergence term [4]. For the reconstruction term $\log P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})$ with Monte Carlo approximation [134], the final total loss can be calculated as:

$$\begin{aligned} l &= \mathbb{E}_{q(\boldsymbol{z})}[\log P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})] + \mathcal{D}_{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{z})||P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})] \\ &\approx \frac{1}{M}\sum_{m=1}^{M} \log P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}) + \mathcal{D}_{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})||P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})] \end{aligned} \tag{4.18}$$

Where M is the number of samples from the posterior distribution $\boldsymbol{z}$. We use a single sample of $M = 1$ during training based on [94], and $M = 20$ during testing based on [134]. In the training phase, $\boldsymbol{z}$ is sampled from $q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y};\boldsymbol{\phi})$ and in the test phase, from $P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$.

4.4 Experiments

4.4.1 Dataset

Here, we used the benchmark posts dataset from the Stanford MOOC forum [221], containing 29604 anonymised posts collected from 11 different courses. Each post is manually labelled by three independent human experts and with agreements for the gold label. Apart from the text content, each post is evaluated based on six categories, amongst which urgency, which is the one we used here. Its range is 1 to 7, with 1 meaning no reason to read the post and 7 meaning extremely urgent for instructor interventions.

An example urgent message is "I hope any course staff member can help us to solve this confusion asap!!!"; whilst a non-urgent would be "Good luck to everyone.".

See more details on their website[1]. Similar to [204], we convert the problem of detecting urgent posts to a binary classification task. A threshold of 4 is used as in [204] to create two need categories as: 1) *Need for urgent intervention (value > 4) with label 1*; and 2) *No need for intervention (values ≤ 4) with label 0*. This allows us to obtain a total of 29,597 posts, with 23,991 labelled as 0 and 5,606 labelled as 1. We tokenise the text and create a vocabulary based on a frequency-based cutoff [176] of 5 and use the special token $< pad >$ for padding and the unknown token $< unk >$ for out-of-vocabulary words. We initialise the embedding layer with a 300-dimensional GloVe vector [45] if found in the pre-trained token list.

4.4.2 Experiment Setup and Evaluation

In this paper, we have implemented 3 different models: a baseline model (Base), as shown in Figure 4.1; a baseline model converted to a BNN through Monte Carlo Dropout (MCD), as shown in Figure 4.2; and a baseline model converted to a LDVM with variational inference (VI), as shown in Figure 4.3. For the evaluation, we report mean accuracy; F1 score, Precision score, and Recall score for all three models under each class (the higher, the better); and entropy based on the prediction layer [18,218] (the lower, the better).

We conduct two sets of experiments. For the first set, we follow the setup in [203]. At each run of the experiment, we randomly split this data into training and testing sets, each with a ratio of 80% and 20%, respectively, with stratified sampling on a random state. In the second set of experiments, we use fewer training examples, since the intervention case is rare compared with non-intervention, and we compare the robustness of our model given smaller size samples, and we use a split of 40%, 60% for training and testing. The results for the two experiments are reported in Table 4.1 and Table 4.2, respectively, and we run both experiments 10 times. In Table 4.1, we report the best run of the model, and in Table 4.2, we report the mean and variance. All the evaluation metrics results reported here in this paper are based on the test dataset only. In the first table, we use bold text to denote

[1]https://datastage.stanford.edu/StanfordMoocPosts/

			Non-urgent (0)			Urgent (1)		
	Accuracy	**Entropy**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Text [203]	.878	-	.90	.95	.93	.73	.56	.64
Base	**.883**	.095	**.937**	.918	.927	.677	**.738**	**.697**
MCD	.883	**.085**	**.939**	.915	.926	.675	**.742**	**.698**
VI	.873	.103	**.940**	.901	.919	.644	**.752**	.687

Table 4.1: Chapter 4: Results compare deep learning baseline model and Bayesian deep learning approaches in accuracy, precision, recall, and F1 score.

the results that outperform results in [203] and in the second table, we use bold to denote results outperforming the (Base) model.

4.5 Results and Discussions

			Non-urgent (0)			Urgent (1)		
	Accuracy	**Entropy**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Base	.870+- .0039	.1126+- .0041	.930+- .0039	.908+- .0088	.918+- .0030	.645+- .0159	.707+- .0215	.664+- .0052
MCD	.869+- **.0013**	**0.101**+- .0326	.929+- .0128	.908+- .0319	.917+- .0104	.652+- **.0574**	.703+- .0693	.660+- **.0042**
VI	.867+- **.0019**	**0.078**+- .0296	.924+- **.0028**	**.910**+- **.0058**	.916+- **.0017**	**.642**+- **.0093**	.680+- **.0164**	.649+- **.0034**

Table 4.2: Chapter 4: Results compare mean and variance of deep learning and Bayesian deep learning approach based on 10 runs, reported in mean and variance.

The main results are presented in Table 4.1. The baseline model (Base) performs competitively against a strong model [203], especially in the recall and F1 score for the 'urgent' class and the precision score for the 'non-urgent' class. For the Monte Carlo Dropout (MCD) and Variational Inference (VI) models, we achieve better performance in these measurements against the baseline model (Base). Importantly, as an indication of the uncertainty measurement, we note that the entropy dropped for the MCD model. In Table 4.2, we can see that Bayesian deep learning methods generally achieve similar or better performance compared to the non-Bayesian base model, but hold lower variance and lower entropy against small sample size data.

The imbalance of data is often the case in real-life scenarios, where the label 'need intervention' is scarce. A probabilistic approach works as a natural regularisation technique when neural network models are generally over-parameterised. We can

conclude that Bayesian deep learning mitigates this issue of over-parametrisation with lower variance and entropy. This is especially clear for the VI methods. The result from a Wilcoxon test shows that, compared with the Base model, the experiment results of the VI model are statistically significant at the *.05* level, with *p=.022* for the entropy value and with *p=.007* for the recall, in the 'urgent' case. Comparing MCD and VI models, the latter achieves better performance in most metrics, as shown in both tables, especially with a higher recall score. The recall score is preferable to precision in this task, where we have a comparatively small number of positive examples. However, the implementation of MCD models is more accessible to researchers interested in introducing uncertainty into their neural networks. This should be considered in using them in practice.

In conclusion, we reinforce the importance of using Bayesian methods in constructing classifiers for educational domain applications. Our results demonstrate that when classifications are correct, the use of MCD and VI yields comparable performance against conventional deep learning models, with a reduction in entropy score, implying a lower level of uncertainty. Additionally, we have observed that modelling uncertainties also present additional benefits in instances where the base model is incorrect but with high confidence. While the MCD and VI predictions in such cases may still be erroneous, the confidence level is relatively lower, particularly for long forum comments. The ability to decrease uncertainty when confident and increase uncertainty when not confident underscores the potential of Bayesian deep learning in creating AI applications that are more robust and trustworthy.

4.6 Conclusion

Identifying the need for learner interventions for instructors is an extremely important issue in MOOC environments. In this paper, we have explored the benefits of a Bayesian deep learning approach to this problem for the first time. We have implemented two different approaches to Bayesian deep learning, namely Monte Carlo dropout and variational inference. Both offer a critical probabilistic measurement in neural networks. We have demonstrated the effectiveness of both approaches in

decreasing the epistemic uncertainty of the original neural network and granting equivalent or even better performance. We have thus provided guidelines for researchers interested in building safer, more statistically sound neural network-based models in the EDM and LA fields, where the entropy measures a classifier's confidence level. In intelligent tutoring systems, high confidence (thus low entropy) is essential. With Bayesian deep learning, we turn NN models into probabilistic models, allowing more explainability and trust. For future research, these can be extended and applied in more areas.

Epilogue

The approaches presented in this chapter succeed in including uncertainty into the deep learning model, which grants its ability to present uncertainty measurement as an additional metric on top of the standard machine learning metrics (Accuracy, F1, Precision, and Recall). Additionally, it showed that Bayesian methods own advantages for training with imbalanced labelled and smaller size data (with reduced variance given random model initialisation). To the best of our knowledge, this exploration study is the first empirical study on Bayesian deep learning techniques applied to the text-based educational domain. With the two approaches (BNN and DLVM), BNN is more accessible for researchers interested in using it as a plug-and-play tool to model uncertainty; in comparison, the DLVM method is more customised, and generally performs better; however, it requires additional efforts in designing model and training mechanism.

We demonstrate that Bayesian deep learning has the potential to address the concerns over *robustness* (lower variance with random initialisation) and *trustworthiness* (allowing entropy as measurement apart from standard metrics such as F1, Precision and Recall); while maintaining the performance of deep learning. In comparison with the two methods, DLVM allows more interpretability due to the existence of an explicit latent variable and hence is further explored in later chapters of this thesis (Chapter 5, 6 and 7).

CHAPTER 5

Efficient Uncertainty Quantification Framework for Multi-label Text Classification Tasks

Prologue

In Chapter 4, we have presented an empirical study on Bayesian deep learning methods (i.e. **Bayesian neural networks** (BNNs) and **deep latent variable models** (DLVMs)) on a text-based AI in education application. We have shown that probabilistic modelling in such applications allows us to provide an estimation of its uncertainty and results in lower variance with random initialisation. These benefits mitigate concerns over the *robustness* and *trustworthiness* of deep learning based NLP applications. In this chapter, we further explore the concern of *trustworthiness* in NLP applications. In particular, Chapter 4 explored only the epistemic uncertainty, as a result of ignorance in model assumptions [1]; another type of uncertainty in a neural network is called aleatoric uncertainty, which comes as a result of noise

[1]The model assumptions are related to bias in defining the model stage based on the problem or data observed. The hypothesis could be whether the model is linear or non-linear, how many degrees of the polynomial, etc.

in data [2]. In this chapter, we explore both of these uncertainties and present a contribution in novel methods of representing epistemic and aleatoric uncertainties conditioned on text.

We present a novel, efficient uncertainty quantification framework (both epistemic and aleatoric uncertainty) for multi-label text classification. This is the first research on using deep latent variable models for efficient uncertainty quantification purposes in multilabel text classification tasks. For epistemic uncertainty, we compare the effectiveness of modelling epistemic uncertainty between DLVM and BNN, which is based on the widely adopted technique, Monte Carlo Dropout (MCD) [18]. We show that the DLVM-based method achieves competitive performance, while the DLVM method is around 13 to 45 times faster than the MCD method, depending on the architecture. We discuss various strategies for training a DLVM for epistemic uncertainty modelling in multi-label text classification tasks. For aleatoric uncertainty, we extend methods in [18, 218] and use an analytical solution in the data space. We show the benefits of modelling epistemic and aleatoric uncertainties in three text classification tasks with diverse neural network architectures.

Declaration: This chapter is based on the following publication:

Yu, J., Cristea, A.I., Harit, A., Sun, Z., Aduragba, O.T., Shi, L. and Al Moubayed, N., 2022, July. ***Efficient Uncertainty Quantification for Multilabel Text Classification.*** *In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.* (Core A Ranked Conference, Accepted for Oral Presentation)

This chapter is presented largely as accepted, although referencing and notation have been altered and cross-referencing added for consistency across this thesis. Some stylistic changes have been made for consistency. The majority of the text is verbatim, with some minor wording and formatting changes.

[2]The noise can come from missing data, lack of calibration of the data collection device, distribution shift in the sample data collected, etc.

5.1 Introduction

Deep neural networks have been successfully applied in a wide range of natural language processing (NLP) tasks, such as text classification, question answering, and natural language inference [54]. However, modern deep neural networks are (mainly) discriminative models, with only point estimation. They can make predictions *'blindly'* [18], raising, in practice, concerns over AI safety and social bias [19]. In the field of AI safety, one of the critical areas of concern is the robustness and explainability of AI systems. For example, in the case of autonomous vehicles, it is essential to ensure that the system is able to identify situations where it lacks confidence in making decisions, such as whether it should stop or continue driving. Furthermore, it is necessary for the system to provide an explanation for its decision-making process to demonstrate why one decision was made over another. This enhances the AI system's transparency and accountability and helps build trust in its ability to operate safely. One natural solution for such trustworthy and robust AI systems is to combine the predictive power of Deep Learning with the statistical robustness of Bayesian Learning [222].

Combining these two powerful tools inspires two different directions, as shown in Figure 5.1: **Bayesian Deep Learning** (BDL) and **Deep Bayesian Learning** (DBL). In the deep learning approach, each neuron learns a fixed value representing its parameters; in contrast, the BDL approach allows each neuron to learn a distribution of its parameters; and the DBL approach infers a latent variable and learns its distribution instead. A widely adopted BDL approach is the **Bayesian Neural Network** (BNN) [210–212]; while a known DBL approach is the **Deep Latent Variable Model** (DLVM) [4, 105, 106, 223].

From a Bayesian modelling perspective, there exist two main types of uncertainty inside a neural network [224], namely *epistemic* uncertainty and *aleatoric* uncertainty. Epistemic uncertainty represents model uncertainty resulting from ignorance about model assumptions. It can be reduced when more data are observed, as more information leads to better model assumptions. Aleatoric uncertainty is known as data uncertainty and reflects noise inherent in the data, i.e., the deviation between ground truth and observed values. It cannot be reduced, even if more data

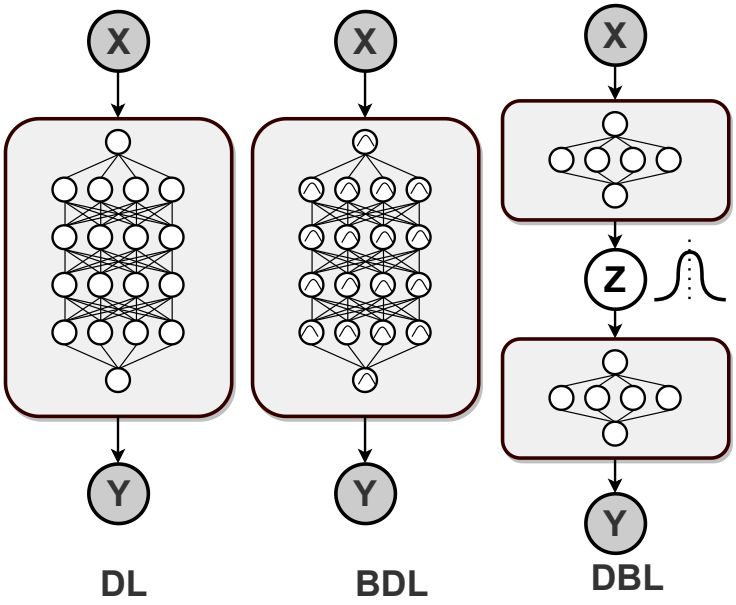

Figure 5.1: Chapter 5: Graphical illustrations of the differences between 'typical' Deep Learning (DL), Bayesian Deep Learning (BDL) and Deep Bayesian Learning (DBL). Note that for BDL, the prior is placed upon the weight of the neurons.

are observed, as the deviation comes from the data itself. Aleatoric uncertainty can be further categorised as: *homoscedastic*, which captures the data-invariant noise across the whole dataset; and *heteroscedastic*, which captures the data-dependent noise over each data instance [18].

BNN is widely used for quantifying epistemic uncertainty on the strength of its robustness to the distribution shift of data [24]. However, for modern deep neural networks, it is computationally expensive to build and train them as BNNs. Alternatively, several works [18, 218] model epistemic uncertainty in deep learning with BNN, via an approximation technique named **Monte Carlo Dropout** (MCD) [193]. The MCD only requires performing a Dropout operation [194] before every weight matrix in a standard neural network, hence eliminating the extra parameters cost for BNN. However, it is still computationally expensive for real-time rendering, due to sampling through a deep neural network many times at each layer, which is regarded as a critical challenge [18].

Recently, amortised variational inference-based [4] DLVMs are proposed as an alternative approach to uncertainty quantification [225, 226]. Compared to BNN, a DLVM enables computationally tractable uncertainty quantification in the form of

posterior analysis in the approximated latent space. Hence it avoids expensive sampling as in MCD. Therefore, in this paper, we explore using DLVM for quantifying epistemic uncertainty as an alternative replacement for MCD in four multi-label text classification tasks. For quantifying aleatoric uncertainty, we place a distribution on prediction outputs as in [18, 218].

5.2 Related Work

Research on practical methods of quantifying uncertainty in deep learning from a Bayesian perspective has only recently been endeavoured [227]. Methods were initially proposed to capture either epistemic uncertainty or aleatoric uncertainty, alone. For epistemic uncertainty, the approach involved a Bayesian approximation technique named **Monte Carlo Dropout** (MCD) [193], based on a widely adopted regularisation technique called Dropout [194]. MCD allows Dropout to be considered equivalent to applying **Variational Inference** (VI) [28, 93, 185] over the full parameter set in a deep neural network. The posterior distribution can then be approximated via multiple runs of the same model with Dropout applied, using the same input data. This practical tool for epistemic uncertainty estimation has been successfully used on a wide range of applications, such as semantic segmentation [216], language modelling [192], diabetic retinopathy [228], transport data analysis [217], magnetic resonance imaging (MRI) segmentation [229], text classification [230] and learning analytics [231].

Most of the works mentioned above focused on modelling epistemic uncertainty alone; and they overlooked the existence of the aleatoric uncertainty, which is equally essential for real-life applications [224]. To bridge this research gap, an uncertainty quantification framework, jointly modelling these two types of uncertainties, was proposed and applied first to computer vision [18] and later extended to natural language processing [218]. It modelled the heteroscedastic aleatoric uncertainty, by placing a distribution on prediction outputs. This distribution is jointly learnt with an additional neural network, during the training process with the original network. The epistemic uncertainty is modelled by creating a BNN using MCD; however,

when applying MCD to modern neural network models, the computational cost in testing time induced by sampling is a critical challenge [18].

Recently, **deep latent variable models** (DLVMs) have been proposed, as an alternative approach to uncertainty quantification. It has been successfully applied to the inverse problem [225] and to image denoising [226]. Compared with a BNN, a DLVM enables computationally tractable uncertainty quantification in the form of posterior analysis in approximated latent space. Hence, it avoids expensive sampling, as required by MCD during testing time. DLVMs for uncertainty quantification purposes have not been well studied in the NLP domain. To address this research gap, we thus propose novel methods of quantifying uncertainties conditioned on text. We compare the performance of a DLVM epistemic modelling on text with the widely adopted method, MCD. We demonstrate the benefits of modelling uncertainties with our novel methods on four multi-label text classification problems. To the best of our knowledge, this is the first time *uncertainty quantification methods conditioned on text* are proposed, allowing for *efficient posterior analysis*. We apply them to diverse neural architectures on text classification tasks with empirical experiments.

5.3 Methodology

This section presents a detailed explanation of our novel methods of modelling uncertainties in deep neural networks. We start with defining the problem and introducing our baseline deep learning approach, and then we articulate how epistemic uncertainty and aleatoric uncertainty can be modelled.

We use three diverse neural network architectures (LSTM, CNN, and Transformer) throughout our experiments. Since our methods are invariant to specific architectures, we use an encoder network, '$\boldsymbol{Encoder()}$', to represent a generic architecture choice. In the context of this study, it can refer to one of these three network architectures; however, it is not limited to the architectures mentioned above. Other neural networks that can produce a fixed dimensional vector representation of a given text could be interchanged as the '$\boldsymbol{Encoder()}$'.

5.3.1 Problem Definition

A multi-label text classification task can be defined as: given training data in the form of N data pairs $\{(x_n, y_n)\}_{n=1}^{N}$, with each pair consisting of the text (denoted by x_n) and their associated label (denoted by y_n). For the n^{th} pair, $x_n = \{w_1, ..., w_L\}$ denotes the set of L words from the input, where each word $w_l \in \mathcal{V}_x$ is an instance of a discrete random variable from the dictionary $\mathcal{V}_x$; $y_n \in \mathcal{V}_y$, an integer, is an instance of a discrete random variable from the set $\mathcal{V}_y$. The purpose is to find the right prediction $\hat{y}^*$, given new data x^*.

In the following descriptions, we omit the data pair index n and use bold characters to represent vector form representations, i.e., $\boldsymbol{x}$ and $\boldsymbol{y}$. These representations will be learnt in an end-to-end fashion.

5.3.2 Deep Learning Approach (Baseline)

In a traditional **deep learning** (DL) approach, used here as a baseline, we build a deep neural network to learn a deterministic function as the approximation for the probability $P(\hat{\boldsymbol{y}}|\boldsymbol{x})$ of the prediction $\hat{\boldsymbol{y}}$, given input $\boldsymbol{x}$. In our experiments, we use a standard architecture setup for text classification. Our architecture consists of an encoder network, as explained above, followed by an affine transformation with an output, where the dimension is equal to the associated classes.

Given an input sequence of words $\boldsymbol{x} = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_L\}$, the encoder network outputs a representation:

$$\boldsymbol{x_{rep}} = \text{Encoder}(\boldsymbol{E}_{\boldsymbol{w}^l}(\boldsymbol{x})) \tag{5.1}$$

Where $\boldsymbol{E}_{\boldsymbol{w}^l}$ is the learnt word embedding for the l^{th} word $\boldsymbol{w^l} \in \boldsymbol{x}$; $\boldsymbol{x_{rep}}$ is then fed through the affine transformation for the prediction $\hat{\boldsymbol{y}}$. With negative cross-entropy loss, for a mutually exclusive K-class multinomial classification, we have the following cost function:

$$\boldsymbol{\theta^*} = \arg\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} (-\boldsymbol{y}^{(k)}) \log \hat{\boldsymbol{y}}^{(k)} \tag{5.2}$$

As suggested in [18], entropy (denoted as $\boldsymbol{H}$) is a measurement of prediction uncertainty[3], and a low entropy indicates the neural network is confident when making predictions. Entropy can be calculated via the logits layer values:

$$\boldsymbol{H}(P(\hat{\boldsymbol{y}})) = -\sum_{k=1}^{K} P(\hat{\boldsymbol{y}}^{(k)}) \log P(\hat{\boldsymbol{y}}^{(k)}) \tag{5.3}$$

Note that in a standard deep learning approach, however, neither epistemic nor aleatoric uncertainty is explicitly modelled. This entropy value is an estimate of 'uncertainty' for 'in-domain' data. A plethora of research has demonstrated that it is easy to find or synthesise inputs for which a standard neural network is highly confident, yet wrong [232].

5.3.3 Modelling Epistemic Uncertainty

To model epistemic uncertainty in a standard neural network model, we introduce an additional unobserved random variable $\boldsymbol{z}$ and place a distribution on it. This essentially turns our model into a conditional variational auto-encoder (CVAE) [105,106]. CVAE has been explored as a supervised generative model for text classification [134, 233]. In this paper, we follow [225, 226], and study the effectiveness of CVAE as a tool to model epistemic uncertainty in a deep neural network. For each observed data pair $\{\boldsymbol{x}, \boldsymbol{y}\}$ and its associated latent variable $\boldsymbol{z}$, the joint distribution conditional on $\boldsymbol{x}$ can be factorised as follows:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x_{rep}}) = P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x_{rep}}) P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x_{rep}}) \tag{5.4}$$

Where $\boldsymbol{\theta}$ is the set of neural network parameters and $\boldsymbol{x_{rep}}$ comes from the same encoder architecture as for the standard deep learning approach. We use amortised variational inference [4] and adopt the same assumption for continuous multivariate Gaussian with diagonalised co-variance matrix as in [134, 233]. The evidence lower bound ($\mathcal{L}$(ELBO)) for the marginal likelihood is:

[3]We discuss the applicability of entropy as uncertainty estimation in section 5.5

$$\log P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \geq \mathcal{L}(\text{ELBO}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\log P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x_{rep}})] - \mathcal{D}_{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x_{rep}}, \boldsymbol{y})||P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x_{rep}})] \tag{5.5}$$

The first term of $\mathcal{L}$(ELBO) is the reconstruction loss and is measured via a multi-class cross-entropy loss. The second term is the **Kullback Leibler** (KL) divergence between $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x_{rep}})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x_{rep}}, \boldsymbol{y})$. The variational family $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ here approximates the posterior distribution as in Figure 5.2:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x_{rep}}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x_{rep}}, \boldsymbol{y}), \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x_{rep}}, \boldsymbol{y}))) \tag{5.6}$$

Where we have:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x_{rep}}, \boldsymbol{y}) &= l_1(\boldsymbol{\pi}_{\boldsymbol{\phi}}) \\ \log \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x_{rep}}, \boldsymbol{y}) &= l_2(\boldsymbol{\pi}_{\boldsymbol{\phi}}) \\ \boldsymbol{\pi}_{\boldsymbol{\phi}} &= g_{\boldsymbol{\phi}}(\boldsymbol{x_{rep}}, \boldsymbol{y}) \end{aligned} \tag{5.7}$$

Where l_1 and l_2 are two separate affine transformation functions from $\boldsymbol{\pi}_{\boldsymbol{\phi}}$, $g_{\boldsymbol{\phi}}$ is an MLP unit, and $\boldsymbol{y}$ is the one-hot encoding form of the label. The latent variable $\boldsymbol{z}$ can be reparameterised as $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}$, known as the "reparameterisation trick" [220], with sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$. We adopt the inference and generation network for both P and q distributions, similar to [134,233], shown in Figure 5.2. For the conditional distribution $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x_{rep}})$, we model it as:

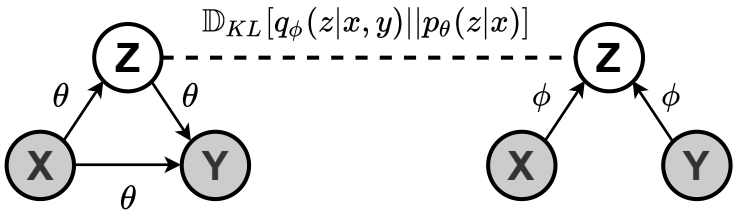

Figure 5.2: Chapter 5: Graphical model for generation network (left) and inference network (right) for CVAE, using amortised VI [4].

$$P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x_{rep}}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu_\theta}(\boldsymbol{x_{rep}}), \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\boldsymbol{x_{rep}}))) \tag{5.8}$$

Where we have:

$$\begin{aligned} \boldsymbol{\mu_\theta}(\boldsymbol{x_{rep}}) &= l_3(\boldsymbol{\pi_\theta}) \\ \log \boldsymbol{\sigma_\theta}(\boldsymbol{x_{rep}}) &= l_4(\boldsymbol{\pi_\theta}) \\ \boldsymbol{\pi_\theta} &= g_{\boldsymbol{\theta}}(\boldsymbol{x_{rep}}) \end{aligned} \tag{5.9}$$

Similarly, l_3 and l_4 are two separate affine transformation functions from $\boldsymbol{\pi_\theta}$, and $g_{\boldsymbol{\theta}}$ is an MLP unit. The KL term in ELBO has a closed-form solution [4], and the first term of ELBO can be calculated via a Monte Carlo approximation, as:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\log P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x_{rep}})] \approx \frac{1}{M}\sum_{m=1}^{M} \log P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z^{(m)}}, \boldsymbol{x_{rep}}) \tag{5.10}$$

The Monte Carlo approximation term here is an unbiased estimator. $\boldsymbol{z}^{(m)}$ is the m^{th} sample from the probability distribution $P(\boldsymbol{z})$, and M is the total number of samples. We use $M = 1$ during training as per standard practice for amortised VI [4] and adopt **maximum a-posteriori** (MAP) as estimation during testing. During training, $\boldsymbol{z}$ is sampled from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x_{rep}}, \boldsymbol{y})$ and during testing, MAP is calculated based on $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x_{rep}})$. Given our prediction $\hat{\boldsymbol{y}}$, the cost function is:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \sum_{k=1}^{K} (-\boldsymbol{y}^{(k)}) \log \hat{\boldsymbol{y}}^{(k)} + \beta \mathcal{D}_{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{z})||P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x_{rep}})] \tag{5.11}$$

We obtain the real $\mathcal{L}$(ELBO) when $\beta = 1$. CVAE models suffer from posterior collapse as discussed in [109]. Here, "posterior collapse" refers to the issue where the latent variable $\boldsymbol{z}$ does not contribute much to the model output. We explore three different methods to address this issue during our training in the later section (as shown later in Experiment 1).

Compared with the **Monte Carlo Dropout** (MCD), CVAE enables computationally tractable uncertainty quantification in the form of posterior analysis in (approximated) latent space. To calculate the epistemic entropy, we use the same formula as in equation 5.3. Due to the isotropic Gaussian assumption for the latent space, the prediction entropy for a single data point can be calculated via posterior analysis in the latent space. As opposed to MCD, CVAE allows more efficient estimates, without computationally expensive sampling [18].

5.3.4 Modelling Heteroscedastic Aleatoric Uncertainty

We follow [18,218] and focus only on *heteroscedastic* aleatoric uncertainty. To model the heteroscedastic aleatoric uncertainty in a standard neural network model, we place a distribution on the network output and define the following generative process:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{y_a}} &= l_5(\hat{\boldsymbol{y}}) \\ \boldsymbol{\sigma}_{\boldsymbol{y_a}} &= l_6(\hat{\boldsymbol{y}}) \\ \boldsymbol{y_a} &\sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{y_a}}, \boldsymbol{\sigma}_{\boldsymbol{y_a}}) \\ \boldsymbol{y} &\sim Categorical(\text{Softmax}(\boldsymbol{y_a})) \end{aligned} \tag{5.12}$$

Where l_5 and l_6 are two separate affine transformation functions and, during training, the empirical mean ($\bar{\boldsymbol{y_a}}$) can be calculated based on sampling. For the aleatoric uncertainty, we have the following loss function, where K is the number of class labels:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} (-\boldsymbol{y}^{(k)}) \log \bar{\boldsymbol{y}_{\boldsymbol{a}}}^{(k)} \tag{5.13}$$

To quantify the aleatoric uncertainty, we can also calculate the entropy value based on the layer value of $\bar{\boldsymbol{y}_{\boldsymbol{a}}}$. Note that here $\boldsymbol{y_a}$ is not learnt through variational inference, hence does not measure the epistemic uncertainty of deep learning. Thus,

the entropy here only represents the uncertainty associated with the noise in the data. To calculate the aleatoric entropy, we use the same formula as in equation 5.3, but replace $\hat{\boldsymbol{y}}$ with $\boldsymbol{y_a}$. Emulating the epistemic modelling case, we use the analytical solution to calculate individual aleatoric uncertainty.

In this study, we adopted a simplified approach for modelling the aleatoric uncertainty by first assuming the output $\boldsymbol{y_a}$ to be a Gaussian distribution, followed by normalizing its value using an additional softmax operation, and then retrieve a categorical distribution $\boldsymbol{y}$. This methodology was chosen for ease of optimization, and a more straightforward analytical solution in the data space, similar to the methodology presented in [218]. This method is related to statistical relaxation techniques, such as reparameterizing Gaussian distributions as categorical distributions, which have been explored in previous research through various techniques, such as Gaussian Softmax Construction (GSM), Gaussian Stick-breaking Construction (GSB), and Recurrent Stick-breaking Construction (RSB) [99]. A recent well-established relaxation technique is called the Gumbel-softmax technique [3, 124], which is mentioned in Chapter 3.6. Although we have adopted a similar idea as the GSM approach, it is worth noting that an ideal assumption would be to consider $\boldsymbol{y}$ to follow a multinomial distribution with a Dirichlet prior, which may be explored in future work.

For modelling the heteroscedastic aleatoric uncertainty, we have jointly trained $\hat{\boldsymbol{y}}$ along with our defined neural networks with the prediction network, which takes $\boldsymbol{x}$ as input. This joint training enables a connection between $\boldsymbol{x}$ and $\boldsymbol{y}$, thus enabling the capturing of aleatoric uncertainty. Our results, as shown in tables 5.4, 5.5, 5.6, and 5.7, indicate that aleatoric uncertainty is captured; however, there is no theoretical measure of the accuracy of our method in approximating the posterior of the heteroscedastic aleatoric distribution. An alternative approach for capturing various types of uncertainty is through the use of a prior network, as presented in [234], which implicitly measures the Bayesian distribution over distributions on a simplex through a neural network during training.

5.4 Experiments

5.4.1 Data

We conduct experiments on three public text classification benchmark datasets, presented in [12]. These datasets can be used in two different types of tasks: (1) *topic classification* (using AG's News and DBPedia) and (2) *sentiment analysis* (using Yelp-P). We denote these datasets as 'AG', 'DB', and 'Y-P', respectively, shown in Table 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6.

A summary of the datasets is provided in Table 5.1[4] and a summary of token statistics is presented in Table 5.2. In our experiments, we use the full testing data set (hence the difficulty of the task is the same) and use only partially the original training data, split as our training set, which we will explain in the following section 5.4.2. We set the maximum token length as 110 (by removing the tokens beyond the first 110) for the 'AG' and 'DB' datasets; and 450 (by removing the tokens beyond the first 450) for the 'Y-P' dataset. The maximum token length is selected based on Table 5.2.

5.4.2 Vocabulary and Sampling

Before performing sampling, we first create the vocabulary for each dataset, which is shared across all the models. We create each vocabulary based on a minimum frequency of 5 and a maximum size of top $20K$ tokens from the complete training data points, with four additional special tokens: $< pad >$, $< unk >$, $< bos >$, and $< eos >$. These tokens are used to denote batched computation padding, out-of-vocabulary words, and the beginning and the end of the sequence, respectively.

We first perform stratified sampling on the original data points to retrieve our class-balanced data points (we sample $20K$ points per class in each task), and then we apply again stratified sampling, to further split these data points into the class-

[4]Note here for the Y-P dataset, the testing data size is bigger than the training data size, which is not a standard setup in machine learning. There are two main reasons for this: first, as explained in section 5.4.2, this ensures class-balanced data; and second, since the Y-P dataset is a relatively easy task (polarised sentiment analysis), use training data of size $32,000$ already grants a decent result.

balanced training set and validation set, with a percentage of 80% and 20%, respectively.

Dataset	Training	Validation	Testing	Number of Classes
AG	64,000	16,000	7,600	4
DBP	224,000	56,000	70,000	14
Y-P	32,000	8,000	38,000	2

Table 5.1: Chapter 5: Dataset summary

Dataset	Mean	Standard Deviation	Min	Max
AG	45	13	12	214
DBP	57	26	3	1500
Y-P	156	143	1	1202

Table 5.2: Chapter 5: Token length statistics, all numbers round to integer.

5.4.3 Experimental Setup

We conduct experiments over three diverse neural network architectures as the encoder network: LSTM [169], CNN [165], and Transformer [81]. For epistemic uncertainty modelling and aleatoric uncertainty modelling, we add '+EP' and '+AL' to the model name, respectively, in the tables below (Tables 5.4, 5.5 and 5.6).

To compare with the widely adopted epistemic uncertainty modelling technique [193][5], we implement the Monte Carlo Dropout technique for each network, denoted by the addition of '+MC', following the implementation guideline in [218]. We fix the dropout rate as 0.5 based on our empirical experiments. We use Adam [182] as our optimiser in all experiments. The batch size is set to 32 and all training runs for a maximum of 10 epochs. All experiments are conducted on a computer with an Ubuntu operating system and a single RTX 2080 Ti GPU.

Next, we introduce the setup of the three model architectures (LSTM, CNN, and Transformer) below. Each architecture is based on a widely adopted model for text classification tasks in the NLP domain.

[5]Here, we only compare with [193] since [218] adopted a similar approach as in [193] and both of them use Monte Carlo Dropout to model the epistemic uncertainty. [193] focused on applications to computer vision while [218] focused on applications to natural language processing.

LSTM

For the LSTM experiment, we use an embedding size of 56 for each unique word and build a single-layer bi-directional LSTM (Bi-LSTM) network architecture with a hidden dimension of 56 (resulting in 112 latent dimensions). We use a size of 56 for the CVAE latent variable $\boldsymbol{z}$. We use the last hidden state of the Bi-LSTM network as the $\boldsymbol{x_{rep}}$.

CNN

For the CNN experiment, we apply the $2d$-convolution operation (over sequence length and embedding dimension) as in [58] on our text input and use learnable filter sizes of 1, 2 and 3 to represent 'unigram', 'bigram', and 'trigram' information from the text sequence. We use a max pooling operation over each filter output to alleviate various sequence length issues and concatenate them as $\boldsymbol{x_{rep}}$. For each $2d$-convolution operation, we use an input channel size of 1 and an output channel size of 56. We use an embedding size of 56 for each unique word and a dimension of 56 for the CVAE latent variable $\boldsymbol{z}$ for our epistemic modelling experiments.

Transformer

For the Transformer experiment, we use the same architecture setup as in [81], but with a stack of $N = 1$ encoder layer and with no decoder (as it is not required for text classification). For a detailed description of the transformer network, please refer to [81]. We use the same positional encoding methods (sine and cosine) as in [81] and use a hidden dimension of 56, feedforward neural network intermediate layer dimension of 256, and 8 attention heads. This grants us an output dimension of 56, and we use the same size (56) for the CVAE latent variable $\boldsymbol{z}$. As the transformer encoder creates a sequence of contextual representations for a given input sequence, we pre-process text by adding additional special tokens $< bos >$ and $< eos >$, before and after the sequence input. We use the output at the $< bos >$ position as $\boldsymbol{x_{rep}}$, which is similar to how a BERT model [54] is used for text classification.

Model Architecture Complexity

For a fair comparison, we set each model architecture with similar complexity in terms of the number of trainable parameters. For the LSTM, CNN, and Transformer models defined in the experiments, there are around $1.4M$, $1.5M$, and $1.4M$ trainable parameters, respectively (calculated based on the epistemic uncertainty experiment). There are slight variations in the number of parameters in other experiments (Table 5.4, 5.5 and 5.6, results without the '+EP' flag), but the model complexities always remain on the same scale.

5.4.4 Experimental Run

For reporting results, we first select the best learning rate from among $1e-2$, $1e-3$, $1e-4$ and $1e-5$, based on the mean average run from three random seeds: 1000, 2000, and 3000. Then, we run each model 3 times with the best learning rate and use seeds from 1111, 2222, and 3333 to initialise model parameters. For each run of the model, we use the training, validation, and testing sets as shown in Table 5.1. We evaluate the model performance on the validation set after each epoch and save the best model based on the $F1$ score, calculated over the validation set. For testing, we apply the best-performing model on the test set and report the evaluation metric.

Based on our experiments, the optimal learning rates for our models are $1e-3$ and $1e-4$ in general, while most of the time, $1e-3$ performs the best. Thus we use a learning rate of $1e-3$ when reporting the results unless specified otherwise. During training, we use a gradient clip of 1, to avoid gradient overflow.

5.4.5 Evaluation Metric

We report the *mean* and the *variance* (in brackets) amongst three runs on the standard evaluation methods for the text classification task (macro-averaged $F1$) in *Experiment* 1 and *Experiment* 2. For the $F1$ scores, all values are reported in percentage. We report additional entropy measurement [18] calculated with Equation 5.3 over the test set in *Experiment* 2. For the entropy measurement, we report the mean average over the three runs.

5.5 Results and Discussion

5.5.1 Experiment 1: Posterior Collapse

In the first experiment, we address the commonly encountered problem in CVAE training, i.e., the posterior collapse [104,109,111], in the context of text classification. We compare three different methods for training a CVAE, including using standard KL, KL Annealing, and KL Coefficient. We apply each of the three methods to the epistemic uncertainty modelling task over three diverse architectures. Here, we briefly describe these three methods:

1. **Standard KL**

   During training, we use the standard CVAE formula (as in Equation 5.11) with $\beta = 1$ as the KL term.

2. **KL Annealing**

   During training, we apply KL annealing on β, as in [104,109], to induce the cost function and thus maintain a substantial KL value. In all of our experiments, we run a total of 10 epochs. Specifically, we linearly anneal β from 0 to 1 over the first 5 epochs and then use a value of 1 for the remaining 5 epochs. The KL annealing essentially forces the model to explore and utilise the information from the latent variable $\boldsymbol{z}$ [111].

3. **KL Coefficient**

   During training, we use a reduced $\beta = 0.2$ for the KL term as in [109,118]. This allows us to adjust the weight of the KL penalty related to the reconstruction loss.

Experimental results on macro-average $F1$ for the LSTM, CNN, and Transformer architectures, respectively, are presented in Table 5.3. Training with standard KL, KL Annealing, and KL Coefficient are denoted as 'EP ($\beta = 1$)', 'EP (Ann.)', and 'EP (Coe.)', respectively. Throughout the experiments, we have discovered that training with the KL annealing method provides us with the best $F1$ score in general on both *mean* and *variance* across three random runs with different seeds (see

Table 5.3) when the MAP decoding method is adopted, especially for the LSTM architecture, which is reported similarly in literature [104, 109, 111]. Based on the results shown in Table 5.3, we adopt the KL Annealing training technique in the follow-up experiments, although it requires an additional hyperparameter tuning (rate for linear annealing reduction).

To demonstrate the efficacy of the results, we apply the Wilcoxon signed-rank test to the results (KL Annealing against the other two in Table 5.3) from multiple 3 runs of our models. In terms of the AG dataset, the $F1$ score is better, although not statistically significant ($p > .05$) for Transformer; and statistically significantly better ($p < .05$) for LSTM and CNN. For the DB dataset, $F1$ is better, but not statistically significant ($p > .05$) for LSTM and CNN; and not better for Transformer. For the YP dataset, $F1$ is statistically significantly better ($p < .05$) for LSTM and Transformer; and better, but not statistically significant ($p > .05$), for CNN.

5.5.2 Experiment 2: Uncertainty Modelling

In the second experiment, we apply uncertainty modelling over the four datasets with three diverse neural network architectures. For epistemic uncertainty modelling, we use the KL annealing technique during training, followed by the results in experiment 1 (Table 5.3). The experimental results on macro-average $F1$ and entropy for LSTM, CNN, and Transformer architectures are presented in Tables 5.4, 5.5, and 5.6, respectively. The results suggest that uncertainty plays an important role in text classification tasks. In general, modelling uncertainty, as we observe, mostly comes with a benefit. We use bold font to denote when results are better than the deep learning baseline models, and use underline to denote the best performing methods in Tables 5.4, 5.5, and 5.6.

Epistemic Uncertainty

Compared with the '+MC' method using Monte Carlo dropout, we observe that our epistemic uncertainty modelling method '+EP' achieves competitive or better performance across most experiments (based on mean and variance) for the $F1$ score. However, the '+EP' model grants much lower entropy in general, compared

Model	AG	DBP	Y-P
LSTM (%)			
EP (Ann.)	**91.13 (0.33)**	**98.36 (0.08)**	**91.70 (0.23)**
EP (Coe.)	90.94 (0.44)	98.33 (0.04)	90.69 (0.12)
EP ($\beta = 1.$)	91.12 (0.39)	98.33 (0.02)	91.52 (0.15)
CNN (%)			
EP (Ann.)	91.38 (0.30)	**98.22 (0.04)**	**92.69 (0.01)**
EP (Coe.)	91.42 (0.52)	98.15 (0.08)	92.21 (0.16)
EP ($\beta = 1.$)	**91.47 (0.28)**	98.15 (0.02)	92.33 (0.21)
Transformer (%)			
EP (Ann.)	91.29 (0.15)	97.52 (0.14)	**92.16 (0.10)**
EP (Coe.)	91.03 (0.19)	**97.68 (0.01)**	91.88 (0.11)
EP ($\beta = 1.$)	**91.50 (0.29)**	97.63 (0.04)	91.99 (0.26)

Table 5.3: Chapter 5: Posterior Collapse Experiment on LSTM, CNN and Transformer architectures. Results presented in Macro F1-score based on the test dataset.

Model	AG	DBP	Y-P
Macro-F1%			
LSTM	91.17 (0.28)	98.39 (0.03)	91.20 (0.17)
LSTM + EP	91.13 (0.33)	98.36 (0.08)	**91.70 (0.23)**
LSTM + MC	**91.49 (0.21)**	**98.58 (0.02)**	**91.74 (0.35)**
LSTM + AL	91.16 (0.20)	**98.40 (0.03)**	91.20 (0.29)
Entropy (Ave.)			
LSTM	0.1866	0.0360	0.2166
LSTM + EP	**0.0434**	**0.0262**	**0.0338**
LSTM + MC	**0.1766**	**0.0193**	**0.1816**
LSTM + AL	0.2359	**0.0315**	**0.1589**

Table 5.4: Chaper 5: Uncertainty modelling results for LSTM model, results presented in Macro F1-score and entropy based on the test dataset.

to the MC dropout method, as shown in Tables 5.4, 5.5, and 5.6. The entropy value is regarded as a critical measurement of information uncertainty for the predictive distribution [18]. The lower the entropy, the more confident the classifier is in its decision. '+EP' results in a lower entropy than the '+MC' method in general, demonstrating its effectiveness in reducing epistemic uncertainty.

To demonstrate the efficacy of the results, we apply the Wilcoxon signed-rank test on all the results (compare baseline model, i.e. LSTM, CNN, and Transformer; and with uncertainty modelling, i.e.'+EP') from the multiple runs of our models. The $F1$ score is generally competitive, or better than the baseline model. The entropy

Model	AG	DBP	Y-P
Macro-F1%			
CNN	91.49 (0.27)	98.32 (0.06)	92.49 (0.12)
CNN + EP	91.38 (0.30)	98.22 (0.04)	**92.69 (0.01)**
CNN + MC	**91.77 (0.18)**	**98.39 (0.01)**	92.20 (0.23)
CNN + AL	**91.70 (0.37)**	**98.35 (0.06)**	92.40 (0.10)
Entropy (Ave.)			
CNN	0.1960	0.0327	0.1619
CNN + EP	**0.0651**	**0.0203**	**0.0240**
CNN + MC	0.2014	0.0379	**0.1607**
CNN + AL	**0.1762**	0.0357	**0.1261**

Table 5.5: Chapter 5: Uncertainty modelling results for CNN model, results presented in Macro F1-score and entropy based on the test dataset.

Model	AG	DBP	Y-P
Macro-F1%			
Transformer	91.32 (0.16)	97.88 (0.13)	92.10 (0.03)
Transformer + EP	91.29 (0.15)	97.52 (0.14)	**92.16 (0.10)**
Transformer + MC	**91.43 (0.18)**	97.84 (0.14)	91.87 (0.32)
Transformer + AL	**91.41 (0.22)**	**97.89 (0.01)**	92.07 (0.06)
Entropy (Ave.)			
Transformer	0.2274	0.0543	0.2548
Transformer + EP	**0.0592**	**0.0341**	**0.0319**
Transformer + MC	**0.2106**	**0.0339**	**0.2164**
Transformer + AL	**0.1969**	**0.0444**	**0.1559**

Table 5.6: Chapter 5: Uncertainty modelling results for Transformer model, results presented in Macro F1-score and entropy based on the test dataset.

value is mostly statistically significantly better ($p < .05$), in comparison with the baseline model. When combining results from both the $F1$ score and entropy value, we can claim that it is beneficial to model epistemic uncertainty in deep neural networks.

Aleatoric Uncertainty

Compared with the baseline model, our aleatoric uncertainty modelling method '+AL' achieves competitive or better performance across most experiments (based on *mean* and *variance*) for the $F1$ score. Again, for the Wilcoxon signed-rank test on all results (comparing baseline model, i.e. LSTM, CNN, and Transformer; with

uncertainty modelling, i.e.'+AL') from the multiple runs of our model, $F1$ is better, although not statistically significantly so. The entropy value is mostly statistically significantly better ($p < .05$), in comparison to the baseline model. When combining results from both the $F1$ score and entropy value, we can claim that it is beneficial to model aleatoric uncertainty in deep neural networks. Additionally, since modelling aleatoric uncertainty only requires tiny changes in the cost function during training, it is inexpensive and hence is recommended for text classification tasks whenever possible.

To illustrate what we modelled for aleatoric uncertainty, we provide examples of high and low uncertainty of data in the Yelp-P dataset, presented in Table 5.7 (based on the LSTM model). The Yelp-P data is used for the polarised sentiment analysis task (negative or positive). We can observe that the low aleatoric uncertainty examples contain clear sentiment words, such as *'not very good'*, *'don't like'*, and *'horrible'*. On the contrary, the high aleatoric uncertainty examples contain vague sentiment words, such as *'little high'* and *'little too sweet'*; or words with vague meaning, such as *'seriously.'*; or even sentiment words with contradictory meanings, such as *'great'*, *'greasy'* and *'sad'* concomitantly. These observations align with the results presented in [218].

5.5.3 Discussion on Entropy as Uncertainty Estimation

In this paper, entropy is used as an uncertainty estimation, as in [18, 218]. However, the reason for choosing entropy as the measurement is not clear in the literature, so here we discuss briefly the reason and whether it is applicable. From a Bayesian modelling perspective, uncertainty is best presented as the variance of the density function for the posterior distribution in regression tasks. In this paper, we instead explore classification tasks, and thus the variance is best represented as the entropy measurement over the posterior distribution, as in equation 5.3. In this case, a lower entropy value indicates a decrease in classification variance for a predictor, which represents a decrease in uncertainty. However, a lower entropy value only can not denote an improvement in model performance, so we additionally adopt the $F1$ score as another metric measurement.

High Aleatoric Uncertainty
The price is a **little high** for me. They don't give enough bean sprouts without asking for more. The pho broth is a **little too sweet** for me also.
I like their beer ... seriously.
Was a **great** place to eat, now food is **greasy**, and it doesn't taste like it used to. Used to have a more of a homemade taste, now tastes like Costco business center. **Sad**. I **loved** going here .
Low Aleatoric Uncertainty
I went there last night and I ordered the calamari. It had **no taste** and was **very expensive**. And the service was **not very good**. I **will not be back**
Oh my... **dont like** the food here. Tried the Pad thai and the chicken fried rice ... the pad thai was **disgusting** and the fried rice was not $< unk >$ either ... **wont be** going back ... ugh
Wow this place has gone down hill. **Old** smelly rooms. Service is **horrible**

Table 5.7: Chapter 5: High versus Low aleatoric uncertainty examples for Yelp-P dataset.

In this work, the quality of uncertainty estimation has been evaluated using a combination of the $F1$ score and entropy. However, there is a lack of other evaluations, such as misclassification detection, out-of-distribution input prediction, and adversarial attack detection, which are important metrics for measuring the quality of uncertainty estimation results. Although the $F1$ score and entropy provide a decent evaluation of the sharpness of the distribution on the simplex, a more comprehensive assessment is necessary to fully understand the effectiveness of the uncertainty measurements.

It is worth noting that the main focus of this work is on introducing a novel framework for modelling both epistemic and aleatoric uncertainty conditional on text, along with empirical experiments on various neural network architectures. As a result, further exploration into the quality of uncertainty estimation has been left as a future direction. Despite the limitations, the combination of the $F1$ score and entropy provides a good starting point for evaluating the quality of uncertainty estimation results from a Bayesian perspective.

Model	AG	DBP	Y-P
LSTM (s)			
LSTM	0.700	7.746	12.607
LSTM + EP	**0.834 (1.08)**	**8.132 (1.04)**	**14.51 (1.15)**
LSTM + MC	26.990 (34.91)	299.867 (38.70)	582.844 (46.20)
LSTM + AL	0.783 (1.01)	8.034 (1.03)	12.775 (1.01)
CNN (s)			
CNN	0.278	2.426	1.877
CNN + EP	**0.358 (1.28)**	**3.077 (1.26)**	**2.100 (1.11)**
CNN + MC	3.597 (12.90)	33.350 (13.7)	46.011 (24.50)
CNN + AL	0.282 (1.01)	2.550 (1.05)	1.887 (1.01)
Transformer (s)			
Transformer	0.412	4.186	19.689
Transformer + EP	**0.439 (1.06)**	**4.421 (1.05)**	**20.185 (1.02)**
Transformer + MC	11.185 (27.10)	129.108 (30.80)	848.486 (43.10)
Transformer + AL	0.415 (1.01)	4.604 (1.09)	20.878 (1.06)

Table 5.8: Chapter 5: Run Time Results for Uncertainty modelling for LSTM, CNN and Transformer models, results presented in seconds based on the test dataset (in brackets the number of times it is faster, compared to its respective baseline model).

5.5.4 Analysis on Run Time Efficiency

Based on the information provided for the experimental setup in sections 5.4.3 and 5.4.3, we present an analysis of the run time efficiency in Table 5.8. In particular, we present the run time for each testing set and the number of times it is faster than its base model (i.e. LSTM, CNN, and Transformer). We also underline the multipliers for '+MC' and '+EP'. We observe that, in general, modelling aleatoric uncertainty ('+AL') does not increase run time during testing. However, with epistemic uncertainty ('+EP' and '+MC'), the current widely adopted approach ('+MC') requires significantly more time, as noted in [18]. While our approach barely changes the run time and is a lot faster (between 13 to 45 times faster).

5.6 Conclusion

This chapter presents novel uncertainty quantification methods that allow efficient analysis of posterior inference. We demonstrate their effectiveness on four multi-label text classification tasks. Our framework allows efficient posterior analysis,

and our experiments affirm the benefits of modelling uncertainty. We show the consistency of our results over multiple experiments with diverse neural network architectures. Our work can serve as a baseline for applying uncertainty quantification to text classification tasks and contributes to further research in this domain.

Epilogue

In the second work, we extend our contribution of the benefits of Bayesian deep learning in the first chapter from an educational NLP application to a general NLP application using DLVMs. The approaches presented in this chapter shows the benefits of explicitly modelling uncertainties (epistemic and aleatoric) in natural language understanding problems on multi-label text classification tasks with DLVMs. Prior research has also proved benefits when modelling these uncertainties conditional on text; however, these methods are mainly based on MCD and hence require expensive computational sampling techniques to approximate the posterior distribution. We alternatively propose an efficient uncertainty quantification framework based on posterior analysis to model these uncertainties. We demonstrate the effectiveness and generalisation of our approach on three benchmark datasets and three types of network architectures (LSTM, CNN and Transformer). In comparison with the current framework, our proposed methods require significantly less inference time.

We demonstrate that DLVM addresses the concerns over *trustworthiness* (statistically significant lower entropy value); while achieving competitively or sometimes even better performance compared with deep learning (general competitive F1 score). This chapter (Chapter 5) and the previous chapter (Chapter 4) explored the benefits of deep latent variable models for natural language understanding problems. For building NLP applications, apart from natural language understanding problems, another kind of challenge is natural language generation problems. Hence in the next few chapters (Chapter 6 and 7), we further explore the benefits of deep latent variable models for natural language generation problems.

CHAPTER 6

A Deep Generative XAI Framework for Natural Language Inference Explanations Generation

Prologue

In Chapter 4 and 5, we have focused on text classification tasks, which belong to natural language understanding problems for **natural language processing** (NLP). We have presented studies on the benefits of uncertainty modelling with Bayesian Deep learning methods empirically, especially with **deep latent variable models** (DLVMs). In this chapter, we move our focus toward natural language generation problems and explore the applications of DLVMs for such problems.

In this chapter, we present a novel deep generative explainable artificial intelligence (XAI) framework for natural language inference explanation generation. This is the first research on how deep latent variable models can be applied to generate multiple explanations in the Stanford natural language inference task. Our main contributions include: *(i)* a novel two-step generative XAI framework, named INTERACTION, which presents explanations in two steps: (step one) Explanation and Label Prediction; and (step two) Diverse Evidence Generation; *(ii)* the first

study on spurious correlation on the e-SNLI dataset with Transformer architecture; *(iii)* demonstrating the benefits of our framework, against state-of-the-art baseline models with empirical experiments; and *(iv)* a solid deep generative model baseline for future research in the XAI field.

Declaration: This chapter is based on the following publication:

Yu, J., Cristea, A.I., Harit, A., Sun, Z., Aduragba, O.T., Shi, L. and Al Moubayed, N., 2022, July. ***INTERACTION: A Generative XAI Framework for Natural Language Inference Explanations.*** *In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.* (Core A Ranked Conference, Accepted for Oral Presentation)

This chapter is presented largely as accepted, although referencing and notation have been altered and cross-referencing added for consistency across this thesis. Some stylistic changes have been made for consistency. The majority of the text is verbatim, with some minor wording and formatting changes.

6.1 Introduction

Traditionally, **natural language processing** (NLP) applications are built based on techniques that are inherently more explainable. Examples of such techniques are often referred to as *'white box'* techniques, including rule-based heuristic systems, decision trees, hidden Markov models, etc. In recent years, due to the advancement of deep learning, a *'black box'* technique, deep neural network, has become the dominant approach [235]. With the advancement of deep neural networks, their ubiquitousness comes at the expense of less interpretability. Hence, concerns have been raised on whether deep neural networks can make reasonable judgements [19, 236], which further triggers an interest in **explainable artificial intelligence** (XAI) research [237].

With XAI techniques in NLP applications, researchers first focused on *feature*-based [238,239], *model*-based [240], and *example*-based [241] explanation techniques.

However, even for experts working as data scientists in the industry, interpreting results from these models was found to be hard and bias-prone [242]. To reduce human interpretation bias, directly generating natural language explanations seemed a better medium for presentation. Rather than based on carefully designed additional tools, *XAI with natural language* produced human-readable explanations as evidence for AI decision-making [243].

The current state-of-the-art approaches, such as those in [244,245], are limited by presenting a single explanation only. However, from an HCI research perspective, it is hard to account for the diversity of human thoughts, and experience [246]. Indeed, natural language allows expressing the same semantic content in various 'correct' (i.e., semantically similar) forms, subject to cognitive biases, social expectations, and socio-cultural backgrounds [247].

This paper addresses this gap by, proposing a generative XAI framework, which presents explanations in two steps: (step one) Explanation and Label Prediction, and (step two) Diverse Evidence Generation. In step one, we offer the most probable explanation and label prediction, similar to other prior work in literature [244, 245]. In our original step two, we adopt deep generative models, to generate multiple diverse explanations via posterior analysis in the latent space.

We evaluate our method specifically on a **natural language inference** (NLI) task [248], which determines whether a 'hypothesis' is true (entailment), false (contradiction), or undetermined (neutral), given a 'premise'. To perform this, an appropriate dataset is needed. Current NLI datasets, however, contain annotation artefacts, which allow the models to make predictions based on spurious correlations [21] [1]. To address annotation artefacts in data, Camburu *et al.* [20] suggest that spurious correlations are much harder to be captured with natural language explanations and propose a large-scale benchmark dataset (e-SNLI), which contains NLI data points and their associated explanations. In this paper, we present our

[1]NLI is an essential yet challenging task, requiring common sense reasoning on the semantic relationships between premise and hypothesis sentence pairs. However, as shown in [21], current NLI datasets contain annotation artefacts, allowing the models to make predictions based on spurious correlations in data. A simple neural network (here a *fastText* classifier [249]) can make correct predictions 67% of the time when only having access to the hypothesis. This phenomenon is evidence of model capture 'spurious correlation' in data.

studies thus on this dataset, with the Transformer architecture, as further explained in Section 6.4 and Section 6.5.

6.2 Related Work

6.2.1 Explainable Artificial Intelligence for Natural Language Processing

General XAI approaches can be categorised in two main ways: [250, 251]: 1) Local vs Global, and 2) Self-Explaining vs Post-Hoc. Our work contributes to explainable artificial intelligence (XAI) from two perspectives: *Local* and *Self-Explaining*, as we provide explanations based on a fine-granularity individual input, and our explanations are directly interpretable.

In terms of explanation techniques and their applications to NLP there are, in general, five different types [235]: 1) feature importance, 2) surrogate model, 3) example-driven, 4) provenance-based, and 5) declarative induction. The first three are more widely adopted and have already been described briefly in section 6.1. The provenance-based technique refers to visualising some or all of the prediction process, such as in [252, 253]. Our work uses the *declarative induction technique*, which tackles the challenging task of providing human-readable representations as part of the results, such as in [20, 254]. Our work further extends [20] with a *probabilistic treatment*.

6.2.2 Supervised Deep Generative Models for Natural Language Processing

Our work is associated with deep generative models, which are based on **Neural Variational Inference** (NVI) [4–6]. NVI is also known as amortised variational inference in the literature and can be considered as an extension of the mean-field variational inference [28, 93]. The NVI technique uses data-driven neural networks instead of more restrictive statistical inference techniques. NVI allows us to infer unobservable latent random variables that generate the observed data and are thus

very efficient for data with hidden structures, such as natural language.

NVI has been successfully applied in various NLP applications, including topic modelling [134, 255], machine translation [109, 256], text classification [134], conversation generation [108, 257], and story generation [258]. This paper explores the *potential for XAI with natural language inference explanation generation* with a novel deep generative framework. A very recently published paper [259] adopts a similar approach as in this paper; however, the research gap for multiple explanations generation is not explored or discussed. This paper is thus, to the best of our knowledge, the *first work to address the concern on the diversity of human languages* in XAI within the natural language inference task.

6.3 Technical Background

This section provides a brief overview of the **Conditional Variational Autoencoder** (CVAE), the Transformer architecture, and a description of the data.

6.3.1 Conditional Variational Autoencoder

CVAE [105, 106] is a class of **deep latent variable models** (DLVMs) and extended based on the **variational autoencoder** (VAE) model [4, 5]. Both models allow learning rich, nonlinear representations for high-dimensional inputs. When compared with VAE (performing inferences for the latent representation $\boldsymbol{z}$, based on the input $\boldsymbol{x}$, only), CVAE performs inference for the latent representation $\boldsymbol{z}$, based on **both** the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$, together. CVAE can be considered as a neural network framework based on supervised Neural Variational Inference.

CVAE generally includes two components: an encoder and a decoder. We consider the joint probability distribution and its factorisation, in the form of $P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}) = P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x})P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ as in [108, 109, 134, 257, 258]. The encoder $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ takes the observed input $\boldsymbol{x}$ and produces a corresponding latent vector $\boldsymbol{z}$ as the output with parameter θ. The decoder $P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x})$ takes the observed input $\boldsymbol{x}$ and its corresponding latent vector sample $\boldsymbol{z}$ as the total input and produces an output $\boldsymbol{y}$ with the parameter θ. The latent variable $\boldsymbol{z}$ in the joint probability $P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{x})$ can be

marginalised out by taking samples from $P(\boldsymbol{z})$.

For CVAE, we optimise the following **evidence lower bound** (ELBO) for the log-likelihood during training:

$$\log P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \geq \mathcal{L}(\text{ELBO}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\log P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{x})] - D_{\text{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})||P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})] \tag{6.1}$$

The first term of ELBO is the reconstruction loss and is measured via cross-entropy matching between predicted versus real targets $\boldsymbol{y}$. The second term is the **KullbackLeibler** (KL) divergence between two distributions $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$. As the true posterior distribution $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is intractable to compute, a variational family distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$ is introduced as its approximation. We consider that both $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$ are in the form of isotropic Gaussian distributions, as $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}), \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\boldsymbol{x})))$ and $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{y}), \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{y})))$. Our work takes a similar assumption, but the key difference lies in the design of our novel model architectures (section 6.5), together with using the Transformer model [81] as a building block. We provide a detailed explanation of the Transformer model in the next section.

6.3.2 Transformer Architecture

The Transformer architecture, proposed in [81], is the first neural network architecture entirely built upon the self-attention mechanism. It has been used as the main building block for most of the current state-of-the-art models in NLP, such as BERT [54], GPT3 [56], and BART [260].

The Transformer architecture can be divided into three main components: an embedding part, an encoder, and a decoder. The embedding part takes the input $\boldsymbol{x} \in R^{s_1 \times 1}$ in the form of a sequence with length s_1 and uses an input embedding to create $E(\boldsymbol{x}) \in R^{s_1 \times E}$, where E is the embedded dimension size. Due to the permutation-invariant self-attention mechanism, [81] further introduces positional encoding, to encode sequential order information, as $P(\boldsymbol{x}) \in R^{s_1 \times E}$. The sum of positional encoding and input embedding is used as the final embedding of the input x. In [81], sine and cosine functions of different frequencies are adopted as positional

encoding methods. Further work on large-scale transformers [54,56,260] use a *learnt positional embedding*, which is what we utilise in this paper. For the encoder and the decoder, we use precisely the same Transformer architecture as in the original paper [81]. In our experiments, if an encoder and a decoder are used simultaneously, they each have a separate embedding part. We use the official implementation in the Pytorch library[2].

6.3.3 Data Description

Our training data is in the form of N data quadruplets $\{x_n^{(p)}, x_n^{(h)}, y_n^{(l)}, y_n^{(e)}\}_{n=1}^{N}$, with each quadruplet consisting of the *premise* (denoted by $x_n^{(p)}$), the *hypothesis* (denoted by $x_n^{(h)}$) their *associated label* (denoted by $y_n^{(l)}$), and *explanation* (denoted by $y_n^{(e)}$). For the n^{th} quadruplet, $x_n^{(p)} = \{w_1^{(p)}, ..., w_{L_p}^{(p)}\}$, $x_n^{(h)} = \{w_1^{(h)}, ..., w_{L_h}^{(h)}\}$, $y_n^{(l)} = \{w^{(l)}\}$, and $y_n^{(e)} = \{w_1^{(e)}, ..., w_{L_e}^{(e)}\}$ denote the set of L_p words from the premise sentence, L_h words from the hypothesis sentence, a single word $w^{(l)}$ from the label, and L_e words from the explanation sentence, respectively.

Our validation and testing data are similar to data quadruplets as the training data; however, we have three ($y_n^{(e_1)}$, $y_n^{(e_2)}$ and $y_n^{(e_3)}$) instead of one explanation $y_n^{(e)}$, all created by human experts. During training, we update model parameters based on one explanation $y_n^{(e)}$ for n^{th} data entry; and during validation and testing, we perform model selection and inference based on the mean average loss of the three explanations ($y_n^{(e_1)}$, $y_n^{(e_2)}$ and $y_n^{(e_3)}$). In the following, we omit the data quadruplet index n and use bold characters to represent vector form representations, as $\boldsymbol{x^{(p)}}$, $\boldsymbol{x^{(h)}}$, $\boldsymbol{y^{(l)}}$, and $\boldsymbol{y^{(e)}}$. All these representations mentioned above will be learnt in an end-to-end fashion.

6.4 Preliminary Experiments

We present two preliminary experiments in this section. We use the architecture setting similar to the *base* version of the Transformer model [81], which is a 6-layer

[2]https://pytorch.org/docs/stable/nn.html#transformer-layers

model with 512 hidden units and 8 heads for each encoder-decoder network. Based on an inspection of token length statistics (Table 6.1), we set the maximum length of 25 for positional encoding. See section 6.6.5 for a detailed description of all model complexity in this paper.

Model	Mean	Median	Standard Deviation	Min	Max
Premise	17	15	7	4	84
Hypothesis	11	10	4	3	64
Explanation	16	15	7	2	189

Table 6.1: Chapter 6: Token length statistics for the e-SNLI dataset, all numbers round to integer.

We generally follow the vocabulary processing steps as in [20]. Our detailed dataset statistics are presented in Table 6.1, to help reproduce the experiment results, we provide a detailed description of our pre-processing and tokenisation process. We start by stripping out any space in front of and behind the original sentence. And then tokenise it using the Spacy English tokeniser tool based on the '*en_core_web_sm*' lexicon resource. The tokenised text is then used to create the complete vocabulary for training. We follow [20] and remove tokens that appear less than 15 times. We additional include special tokens '$< unk >$', '$< pad >$', '$< bos >$' and '$< eos >$' in the vocabulary. Before we use each sentence, we append '$< bos >$' at the beginning of this sentence and append '$< eos >$' at the end of this sentence, with a space in between.

We report our quantitative assessment results based on 3 random seeds (1000, 2000, and 3000), and report the average performance with its standard deviation in parenthesis. Regarding quantitative assessment, we use automatic evaluation metrics (Perplexity and BLEU [261]) over the entire test data points. Regarding qualitative assessment (Correct@100, as in Table 6.3 and Table 6.7), we report results based on the seed 1000. We adopt the criterion as in [20] and evaluate the Correct@100 score based on the first 100 test examples only[3]. For evaluation, the lower the perplexity, the higher the BLEU score and the higher the Correct@100

[3]The score is related to the correctness for a generated explanation based on the annotations, details described in section 6.6.4.

score, the better the model performs.

We use the **maximum a posteriori** (MAP) estimate decoding for the conditional generation. MAP decoding, whilst not always the optimal choice, has a reasonably good performance and is widely adopted and cheap to compute [262]. For the network optimisation, we use Adam [182] as our optimiser with default hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$). We conduct all the experiments with a batch size of 16 and a learning rate of $1e-5$ for a total of 10 epochs on a machine with an Ubuntu 20.04 operating system and a GTX 2080Ti GPU.

6.4.1 Architecture Selection and Spurious Correlation

In the first experiment, we answer two questions: **Q(i)** *What is a good Transformer model architecture choice for the e-SNLI text classification task?* **Q(ii)** *How easily can a Transformer model pick up the spurious correlation, when only a hypothesis sentence is observed?*

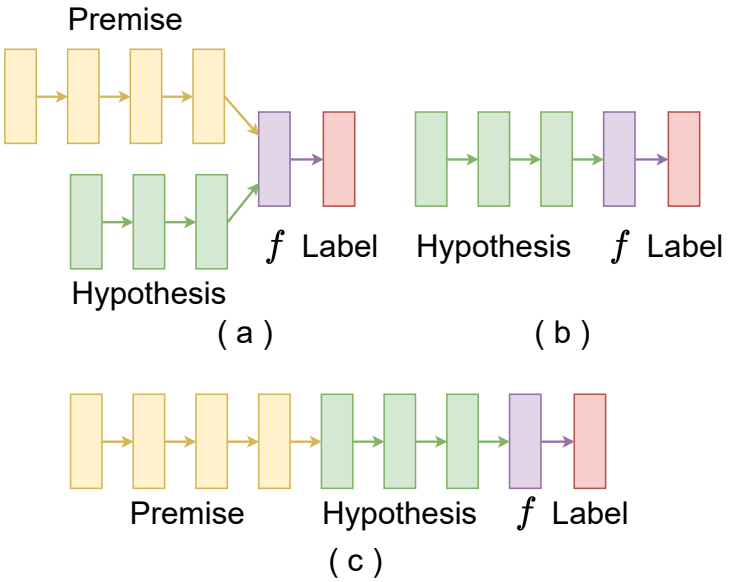

Figure 6.1: Chapter 6: Graphical overview of architectures used in section 6.4.1. (a) Separate Transformer Encoder; (b) Premise Agnostic Encoder; and (c) Mixture Transformer Encoder.

To answer **Q(i)**, we experiment on two candidate model architectures: *(1) Separate Transformer Encoder*: an architecture with two separate encoders, one each for the premise and hypothesis sentences, respectively (Fig. 6.1a). *(2) Mixture*

Model	Accuracy (%)
Separate Transformer Encoder	73.97 (0.34)
Mixture Transformer Encoder	78.98 (1.44)
Premise Agnostic Encoder	65.43 (0.72)

Table 6.2: Chapter 6: Architecture Selection and Spurious Correlation Experiments.

Transformer Encoder: an architecture with a mixture encoder for both premise and hypothesis sentence together (Fig 6.1c).

We choose these two candidates for the following reasons: the first candidate architecture is widely adopted in early NLI literature [263–265], where f refers to algorithmic operations (identity, subtraction, multiplication) as in [266]. The latter candidate architecture is adopted by the BERT model [54], where f refers to an affine transformation operation and has achieved state-of-the-art performance for NLI tasks. To answer **Q(ii)**, we perform the premise-agnostic prediction experiment on the *Premise Agnostic Encoder* model (Fig 6.1b), where f refers to an affine transformation operation.

For the above two experiments, results are presented in Table 6.2. For the *Separate Transformer Encoder*, we use the encoder outputs at two separate '$<$ *bos* $>$' positions for algorithmic operations (identity, subtraction, and multiplication). For *Mixture Transformer Encoder* and *Premise Agnostic Encoder*, we use the output at the first '$<$ *bos* $>$' position. We apply an affine transformation operation for predicting the label. The results suggest the *Mixture Transformer Encoder* outperforms the *Separate Transformer Encoder*, in a statistically significant way ($p < .05$; Wilcoxon test). The *Premise Agnostic Encoder* achieves 82.84% (based on 65.43/78.98) of the *Mixture Transformer Encoder* performance, suggesting that Transformer models tend to capture spurious correlations very easily for the NLI label prediction task.

6.4.2 Premise-Agnostic and Full Generation

In the second experiment, we address two further questions: **Q(iii)** *Is providing explanations as output reducing the impact of spurious correlation in a Transformer model, compared to predicting the label only?* **Q(iv)** *How much better are explana-*

tions based on premise and hypothesis together, instead of hypothesis-only?

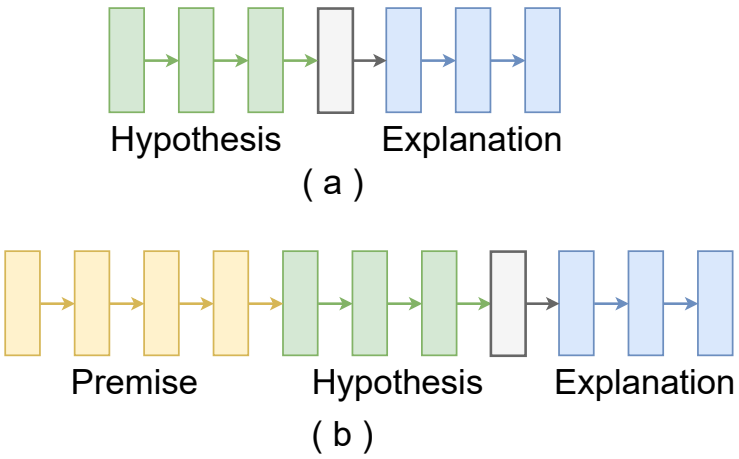

Figure 6.2: Chapter 6: Graphical overview of architectures used in section 6.4.2. (a) Agnostic Generation; (b) Full Generation.

Model	Perplexity	BLEU	Correct@100
Agnostic Generation	7.66 (0.03)	25.74 (0.8)	42.87
Full Generation	5.53 (0.05)	33.14 (0.5)	57.45

Table 6.3: Chapter 6: Premise Agnostic Generation Experiments.

Test Data Number	22
Premise	***one tan girl*** with a wool hat is running and leaning over an object , while another person in a wool hat is sitting on the ground .
Hypothesis	***a boy*** runs into a wall
Explanation	a boy is not a tan girl .
Agnostic Generation	a boy is not a girl .
Full Generation	a boy is not a girl .
Test Data Number	30
Premise	***a couple walk*** hand in hand down a street .
Hypothesis	***a couple is sitting*** on a bench .
Explanation	the couple can not be walking and sitting a the same time .
Agnostic Generation	a couple can not be sitting on a bench and walking down a street at the same time .
Full Generation	the couple can not be walking and sitting at the same time .
Test Data Number	91
Premise	***a dog*** jumping for a frisbee in the snow .
Hypothesis	***a cat*** washes his face and whiskers with his front paw .
Explanation	a dog is not a cat .
Agnostic Generation	a dog is not a cat .
Full Generation	a dog is not a cat .

Table 6.4: Chapter 6: Selected spurious correlation examples.

To answer **Q(iii)**, we follow and extend the *'PremiseAgnostic'* experiment [20]. We use the model architecture shown in Fig. 6.2a, and we are interested in evaluating how well the model can generate an explanation from the premise-agnostic scenario

Test Data Number	4
Premise	***a woman*** with a green headscarf , blue shirt and a very big grin .
Hypothesis	the ***woman is young*** .
Explanation	not all women are young .
Agnostic Generation	the woman is young is the same as the woman is young .
Full Generation	not all women are young .
Test Data Number	9
Premise	an ***old man*** with a package ***poses in front of an advertisement*** .
Hypothesis	a ***man walks by an ad*** .
Explanation	a man can not be walking by an ad while posing in front of it .
Agnostic Generation	a man walks by an ad is the same as a man walks by an ad .
Full Generation	the man either poses or walks by .
Test Data Number	26
Premise	a young ***family*** enjoys ***feeling ocean waves*** lap at their feet .
Hypothesis	a ***family*** is out ***at a restaurant*** .
Explanation	family can't be at restraint if feeling ocean waves
Agnostic Generation	a family is at a restaurant is a rephrasing of a family is out at a restaurant .
Full Generation	a family can not be at a restaurant and at the ocean at the same time .
Test Data Number	69
Premise	an older ***women tending to a garden*** .
Hypothesis	the ***lady*** is ***cooking dinner***
Explanation	the lady can not be cooking dinner if she is tending to a garden
Agnostic Generation	the lady can not be cooking dinner and sitting on a bench at the same time .
Full Generation	the lady can not be tending to a garden and cooking dinner at the same time .
Test Data Number	77
Premise	***a man*** in a black shirt is ***looking at a bike*** in a workshop .
Hypothesis	***a man*** is ***deciding which bike to buy***
Explanation	the man looking at the bike may not be deciding to buy a bike at all .
Agnostic Generation	a man is not a woman .
Full Generation	looking at a bike does not imply deciding to buy .
Test Data Number	97
Premise	***a girl*** playing ***a violin*** along with a group of people
Hypothesis	***a girl*** is playing ***an instrument*** .
Explanation	the violin is an instrument .
Agnostic Generation	a girl is playing an instrument is a rephrasing of a girl is playing an instrument .
Full Generation	a violin is an instrument .

Table 6.5: Chapter 6: Selected non-spurious correlation examples.

(only premise observed). To answer **Q(iv)**, we implement the seq2seq framework [13] with the Transformer architecture. We compare the agnostic generation scenario with the full generation scenario (both premise and hypothesis observed, as shown in Fig. 6.2b).

Our results, presented in Table 6.3, suggest that the agnostic generation significantly reduces ($p < .05$; Wilcoxon test) the ability to generate correct explanations, with only 72.19% (based on 5.53/7.66) for perplexity, 77.67% (based on 25.74/33.14) for the BLEU score, and 74.62% (based on 42.87/57.45) for the Correct@100 score (compared to 82.84% in section 6.4.1).

Additionally, we present qualitative examples taken from these experiments in Table 6.4 and 6.5; these examples are from two scenarios. *(i)* agnostic experiment

where the agnostic generation model can pick up spurious correlation to generate the correct explanations, even when the premise information is not offered. Hence, the explanation generation should ideally be incorrect. *(ii)* agnostic experiment where the agnostic generation model cannot pick up the spurious correlation. In contrast, the full generation model can generate the correct explanations. In the first 100 test examples, case *(ii)* happens a lot more than case *(i)*, suggesting that checking spurious correlation in language generation tasks can be a helpful indicator of the model quality.

6.5 Proposed Deep Generative XAI Framework

In this section, we explain in detail our novel framework, **INTERACTION** - (expla**I**n a**N**d predic**T** th**E**n que**R**y with contextu**A**l **C**ondi**T**ional var**I**ational aut**O**-e**N**coder). Our framework presents explanation in two steps: (step one) *Explanation and Label Prediction*; and (step two) *Diverse Evidence Generation*. We present a workflow diagram for our framework in Fig. 6.3, which consists of four components as follows.

6.5.1 Neural Encoder

Given a pair of premise $\boldsymbol{x}^{(p)}$ and hypothesis $\boldsymbol{x}^{(h)}$, with their associated explanation $\boldsymbol{y}^{(e)}$, the encoder network outputs two sequences of representations:

$$\begin{aligned} \boldsymbol{x_h} &= \text{Encoder}([\boldsymbol{x}^{(p)}; \boldsymbol{x}^{(h)}]) \\ \boldsymbol{y_h} &= \text{Encoder}([\boldsymbol{y}^{(e)}]) \end{aligned} \tag{6.2}$$

Here Encoder refers to the *Transformer Mixture Encoder*, which is selected based on experiments in section 6.4.1. $\boldsymbol{x_h}$ is the contextual representations for the premise $\boldsymbol{x}^{(p)}$ and hypothesis $\boldsymbol{x}^{(h)}$ pair. $\boldsymbol{y_h}$ is the contextual representation for explanation $\boldsymbol{y}^{(e)}$. We share the same encoder network parameters for producing $\boldsymbol{x_h}$ and $\boldsymbol{y_h}$. $\boldsymbol{x_h}$ has the same sequence length as the sum of premise and hypothesis length. $\boldsymbol{y_h}$ has the same sequence length as the explanation length. $[\boldsymbol{a}; \boldsymbol{b}]$ refers to the

concatenation operation of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

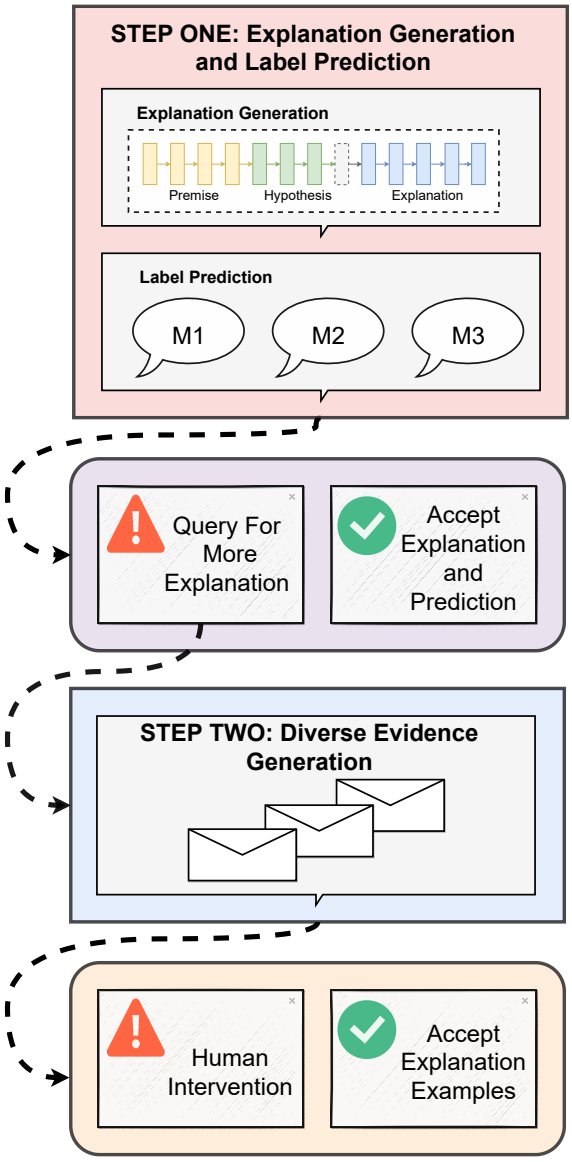

Figure 6.3: Chapter 6: Graphical overview of our framework, **INTERACTION**, introduced in section 6.5.

6.5.2 Neural Inferer

The neural inferer can be divided into two separate components: the prior and the posterior networks, as demonstrated by the ELBO equation 6.1. The parameters of the prior are computed by the prior network, which only takes the inputs: $\boldsymbol{x}^{(p)}$ and $\boldsymbol{x}^{(h)}$. The posterior parameters are determined from both inputs and outputs: $\boldsymbol{x}^{(p)}$, $\boldsymbol{x}^{(h)}$ and $\boldsymbol{y}^{(e)}$.

Contextual Convolutional Neural Encoder

Before introducing the neural prior and posterior, we first present our novel approach to dealing with various lengths of output from the Transformer encoder. We first adopt the 2d-convolution operations (over the sequence length and hidden dimension) as in [58] and apply it directly to the encoded outputs $\boldsymbol{x_h}$ and $\boldsymbol{y_h}$. For the convolution operations, we use learnable filters with sizes of 1, 2, and 3 to represent '*unigram*', '*bigram*', and '*trigram*' contextual information from the sequences. Then, we use a max-pooling operation over each filter output, to alleviate various sequence-length issues and concatenate them as one single output vector. Finally, we apply an affine transformation on the output vector and return the original vector dimension, but with a sequence length of 1. We name the whole set of operations here **c**ontextual c**o**nvolutional **n**eural en**coder** (denoted in short as Concoder).

In contrast, a standard CVAE model uses a fixed position from the sequence instead, to handle various sequence-length issues. We implement a standard CVAE with the $< bos >$ position output as the final output, denoted as *CVAE Generation*. We use this as a comparison with our novel solution (Concoder), denoted as *ConCVAE Generation* (with results shown in Table 6.7).

Neural Prior

The prior distribution is denoted as:

$$P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu_\theta}(\boldsymbol{x}), \text{diag}(\boldsymbol{\sigma^2_\theta}(\boldsymbol{x}))) \tag{6.3}$$

Where $P_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is an isotropic multivariate Gaussian with mean and variance matrices parameterised by neural networks. To deal with the challenge of the variable-length sentence as input, we first use a contextual convolutional neural network, introduced in section 6.5.2, to retrieve a fixed output $\boldsymbol{x_c}$. Then, we apply two additional affine transformations, f_1 and f_2, to parameterise the mean and variance matrices for the neural prior. The tanh() function here introduces additional non-linearity and also contributes to numerical stability during parameter optimisation. Thus:

$$\begin{aligned} \boldsymbol{x_c} &= \text{Concoder}([\boldsymbol{x_h}]) \\ \boldsymbol{\mu_\theta} &= f_1([\boldsymbol{x_c}]) \\ \log \boldsymbol{\sigma_\theta} &= \tanh(f_2([\boldsymbol{x_c}])) \end{aligned} \tag{6.4}$$

Neural Posterior

During training, the latent variable will be sampled from the posterior distribution:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_\phi(\boldsymbol{x}, \boldsymbol{y}), \text{diag}(\boldsymbol{\sigma}^2_\phi(\boldsymbol{x}, \boldsymbol{y}))) \tag{6.5}$$

Where $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ is also an isotropic multivariate Gaussian with mean and variance matrices parameterised by neural networks. However, the parameters here are inferred based on both inputs and outputs. We use the same Concoder network to handle the various lengths of inputs and outputs ($\boldsymbol{x^{(p)}}$, $\boldsymbol{x^{(h)}}$, and $\boldsymbol{y^{(e)}}$). As for the neural prior, we apply two additional affine transformations, f_3 and f_4, to parameterise the mean and variance matrices. Thus:

$$\begin{aligned} \boldsymbol{y_c} &= \text{Concoder}([\boldsymbol{y_h}]) \\ \boldsymbol{\mu_\phi} &= f_3([\boldsymbol{x_c}; \boldsymbol{y_c}]) \\ \log \boldsymbol{\sigma_\phi} &= \tanh(f_4([\boldsymbol{x_c}; \boldsymbol{y_c}])) \end{aligned} \tag{6.6}$$

6.5.3 Neural Decoder

The decoder models the probability of the explanation $\boldsymbol{y^{(e)}}$ in an auto-regressive manner, given the predicted label $\boldsymbol{y_p}$, the encoded premise and hypothesis pair $\boldsymbol{x_h}$, and the latent vector $\boldsymbol{z}$. We obtain the explanation sequence via:

$$\boldsymbol{y^{(e)}} = \text{Decoder}([\boldsymbol{z}; \boldsymbol{x_{(h)}}]) \tag{6.7}$$

Here, Decoder refers to the Transformer decoder. Given an explanation with a

total sequence length of T, at time step j ($j < T$), it produces the j^{th} word with a softmax selection from the vocabulary based on all the past $j-1$ words.

6.5.4 Neural Predictor

In our novel **INTERACTION** framework, the label can be predicted based on one of the three options: *(i)* **M1 Model**: predicted based on the premise and hypothesis only, *(ii)* **M2 Model**: predicted based on the explanation only, and *(iii)* **M3 Model**: predicted based on the premise, hypothesis, and explanation altogether. With the Transformer architecture, we first concatenate the vector outputs of the information at each first '$<$ *bos* $>$' position into a single vector for each model. Then we apply an affine transformation operation f to the concatenated vector. We jointly train the neural predictor together with the generative model ConCVAE. We compare the performance of these three models in our experiments (Table 6.7).

6.6 Experiments

In this section, to evaluate our proposed framework **INTERACTION**, we conduct experiments comparing with our baseline models.

6.6.1 Baseline Models

We define two types of baseline models: *generation model* and *prediction model*. We consider the following works as baseline models:

- seq2seq (*generation model*, our implementation): a sequence-to-sequence learning framework developed by [13]. We implement it with the Transformer architecture and present the results as *Full Generation* in Table 6.7.
- CVAE (*generation model*, our implementation): a strong probabilistic conditional generation framework introduced by [105, 106]. We implement it with the Transformer architecture and present the results as *CVAE Generation* in Table 6.7.

- Transformer (*prediction model*, our implementation): a very strong baseline model for NLI task developed by [81]. We present the results as *Mixture Transformer Encoder* in Table 6.7.

6.6.2 Experiment Setup

To evaluate the explanation generative model of our **INTERACTION** framework, we implement our novel *ConCVAE* model and use the MAP decoding over the latent variable during both training and testing to generate a single explanation. For the label prediction task, we implement the **INTERACTION M1**, **M2**, and **M3** models (as in section 6.5.4), and compare their performance with our predictive and generative baseline models. Regarding network architectures, vocabulary, and training, we use the same experimental setting as in section 6.4.

6.6.3 Diverse Evidence Generation via Interpolation

We present a study on the generation of diverse evidence to support explanation, as in *step two* from Figure 6.3. To generate multiple explanations, we perform a posterior analysis over the latent space. We choose to linearly interpolate the isotropic multivariate Gaussians over its 95.44% region (left and right of 2σ from μ). This interpolation produces 5 samples calculated based on the $\mu - 2\sigma$, $\mu - \sigma$, μ, $\mu + \sigma$, and $\mu + 2\sigma$ coordinates. Examples of interpolation results from the *ConCVAE Generation* experiment are presented in Table 6.8, and we only show the examples which are different. Note that the latent space is smooth, continuous and infinitely unbounded; here, we use 5 samples here just to demonstrate the possibility of generating diverse examples.

Due to the difficulty of the task (end-to-end learning from data directly, with no pre-trained models used) and limited data access (due to the constraint of the e-SNLI dataset only). Gradually interpolation through the latent space does not present a lot of changes as demonstrated in Table 6.8 (usually 2 or 3 out 5, although some data points might have significant changes). Some discussions on improving the diversity of interpolation are presented in section 6.7.5.

6.6.4 Qualitative Evaluation

We calculate the qualitative assessment score, Correct@100, as suggested in [20]: we manually grade the correctness of the first 100 test examples, each with a score between 0 (incorrect) and 1 (correct) and give partial scores of k/n if only k out of n required arguments were mentioned. The required arguments are publicly available on GitHub[4], and we take the mean average of three annotations as the final score.

6.6.5 Model Complexity

We present the model complexity in Table 6.6, with separate counts for prediction, generation and total network components, the one with the '$--$' mark is denoted as not applicable.

Model	Prediction	Generation	Total
Separate Transformer Encoder	48.6M	–	48.6M
Mixture Transformer Encoder	24.3M	–	24.3M
Premise Agnostic Encoder	24.3M	–	24.3M
Agnostic Generation	–	63.6M	63.6M
Full Generation	–	63.6M	63.6M
CVAE Generation	–	65.9M	65.9M
ConTrCVAE Generation	–	68.3M	68.3M
INTERACTION M1	24.3M	68.3M	68.3M
INTERACTION M2	24.3M	68.3M	68.3M
INTERACTION M3	24.3M	68.3M	68.3M

Table 6.6: Number of parameters for each model, with separate counts for prediction and generation component.

6.7 Results and Discussions

6.7.1 Explanation Generation Only

The main results are presented in Table 6.7. For the explanation generation evaluation, we first compare a deep generative model (*CVAE Generation*) with a standard

[4]https://github.com/OanaMariaCamburu/e-SNLI/tree/master/dataset

neural network model (*Full Generation*). The results suggest that the *Full Generation* model performs better, as the perplexity is reduced by $(7.58 - 5.53 = 2.05)$, the BLEU score increases by $(33.14 - 25.70 = 7.4\%)$, and the Correct@100 score increases $(57.45 - 43.04 = 14.4)$. All the results here are statistically significant $(p < .05)$ based on the Wilcoxon signed-rank test. However, deep generative models, such as *CVAE Generation*, allow generating multiple explanations via a posterior analysis over the latent space, as shown in section 6.6.3. With our novel contextual deep generative model *ConCVAE*, we achieve competitive performance with the *Full Generation* model, evidenced in both quantitative (perplexity, BLEU score) and qualitative (Correct @100) results.

Model	Label Accuracy	Perplexity	BLEU	Correct@100
Premise Agnostic Encoder (lower bound)	65.43 (0.72)	−−	−−	−−
Mixture Transformer Encoder (baseline)	**78.98** (1.44)	−−	−−	−−
Full Generation (baseline, non-probabilistic)	−−	**5.53** (0.05)	**33.14** (0.50)	**57.45**
CVAE Generation (baseline, probabilistic)	−−	7.58 (0.27)	25.70 (1.04)	43.04
ConCVAE Generation (our model, probabilistic)	−−	5.69 (**0.03**)	32.74 (**0.09**)	55.27
INTERACTION M1 (our model)	**83.42 (0.31)**	6.73(0.16)	30.46(0.33)	47.04
INTERACTION M2 (our model)	73.73(1.54)	**5.75 (0.01)**	32.68(0.64)	52.29
INTERACTION M3 (our model)	79.85(0.35)	5.93(0.02)	**32.70 (0.28)**	**58.06**

Table 6.7: Chapter 6: XAI with natural language processing Results ('−−' refers to results not applicable).

6.7.2 Explanation Generation and Label Prediction

We implement three variants of our **INTERACTION** framework (**M1**, **M2** and **M3**) to perform generation and prediction simultaneously. Regarding label prediction, results suggest that generating a valid explanation from the premise and hypothesis sentence-pair allows the encoder to better understand the semantics of the words and hence further enhances the accuracy of prediction. This leads to a boost in prediction performance (83.42% for **M1** and 79.85% for **M3**), compared to the *Mixture Transformer Encoder* (78.98%), with the same number of parameters. However, with **M1**, a significant improvement in classification accuracy results in the worst generation quality (based on Correct@100) among all three models. Additionally, as shown in **M2** model, the label prediction accuracy is the worst when using explanation only. This could potentially be explained since only 52.29% of the explanations are considered correct (based on Correct@100).

Regarding explanation generation, we observe that the **M3** model achieves competitive results for the quantitative assessment (perplexity and BLEU) as the *Full Generation* model. Additionally, it achieves the best performance in qualitative assessment (Correct@100) among all models. The results from Table 6.7 suggest that label prediction and explanation generation can complement each other and hence enhance the importance of *XAI with natural languages* in practice. When choosing amongst these three models: for the prediction performance, the **M1** model fits the best; however, for the generation performance, the **M3** model is preferable.

Test Data Number	29
Premise	a **couple** walk hand in hand down a street .
Hypothesis	the **couple** is **married** .
Explanation	just because the couple is hand in hand does n't mean they are married .
Generated Explanation 1	not all couple walking down street are married .
Generated Explanation 2	not all couple in hand is married .
Generated Explanation 3	not all couples are married .
Test Data Number	50
Premise	a little boy in a gray and white striped sweater and tan pants is **playing** on a piece of **playground equipment** .
Hypothesis	the boy is **sitting on** the school bus on his **way home** .
Explanation	the boy is either playing on a piece of playground equipment or sitting on the school bus on his way home .
Generated Explanation 1	the boy can not be playing on a playground and sitting on his way home at the same time .
Generated Explanation 2	the boy can not be playing on a playground and sitting on his way home simultaneously .
Generated Explanation 3	the boy can not be playing on a playground and sitting on the bus at the same time .
Test Data Number	64
Premise	people jump over **a mountain** crevasse on a rope .
Hypothesis	people are jumping **outside** .
Explanation	the jumping over the mountain crevasse must be outside .
Generated Explanation 1	people jump over a mountain so they must be outside .
Generated Explanation 2	a mountain is outside .
Test Data Number	77
Premise	a man in a black shirt is **looking at** a bike in a workshop .
Hypothesis	a man is deciding which bike **to buy**
Explanation	just because the man is looking at a bike does n't mean he is deciding which bike to buy .
Generated Explanation 1	just because a man is looking at a bike in a workshop does n't mean he is deciding to buy .
Generated Explanation 2	just because a man is looking at a bike in a workshop does n't mean he is deciding what to buy .

Table 6.8: Chapter 6: Selected diverse evidence generation examples.

6.7.3 Diversity of Explanation

The main contribution of this paper is to build a model (**INTERACTION**) capable of providing multiple explanations, reflecting the diversity in natural languages. The motivation is that a natural language usually works in a way such that humans often provide more than one explanation for their actions, and hence may find systems that reply 'monosyllabically', or too briefly, potentially frustrating, or even non-informative [247, 267]. Still, our approach raises other questions, e.g., do humans have enough time to read multiple explanations? How do they pick the best or most faithful one? In a recent paper [268], the authors propose first to generate multiple paraphrases and then select the most faithful one. In our paper (Fig. 6.3), we alternatively select the most faithful one based on MAP decoding in *step one* (the maximum likelihood for data), then provide multiple explanations in *step two*. The richness and diversity of the generation of multiple explanations can be observed in Table 6.8 (e.g., for test data numbers 29 and 64). In practice, the MAP decoding might not offer the best results; however, it is a faithful response from the model, given the context of using a 'data-driven' approach with deep learning.

This paper explores the concept of diversity from two perspectives: semantic diversity and syntactic diversity. Semantic diversity refers to selecting words and phrases from the premise and hypothesis that share the same semantic information and support the explanation, as demonstrated by test data number 29. Syntactic diversity refers to variations in sentence order and structure, as illustrated by test data number 64. Both perspectives can be observed from examples in Table 6.8.

6.7.4 Data-driven Template for Explanation

In [20], the authors made effects of removing template-based explanation and using observed templates to refine annotated data. However, in our explanation examples, we were surprised to identify the existence of data-driven templates learnt by our model; similar observations are made in [269]. As challenged by authors in [243], structured information such as template-based explanation might be helpful and sometimes even more faithful. We present some of the data-driven templates dis-

covered by our model here.

The concept of the ‘general template refers to explanations that take the entire premise or hypothesis as the final output, which has been deemed non-informative in previous studies (as stated in [20]). However, despite this, a significant portion of expert annotations still consist of explanations structured in this manner, and the deep neural networks in the paper recognize this general template as a valid output.

The ‘contradict, ‘entailment, and ‘neutral templates were chosen based on the gold standard label of the explanations. It is noteworthy that all of these templates were selected by deep neural networks in a data-driven manner without significant human intervention. Furthermore, when reviewing the annotations provided by human experts, similar templates were observed, indicating that deep neural networks tend to pick up data biases (in this case, the templates) from the data. Therefore, extra caution should be taken in the data annotation process to avoid introducing excessive bias, which remains a standard and predominant approach in the deep learning field.

General Templates

- <premise>
- <hypothesis>

Contradiction Templates

- <XXX> is either <XXX> or<XXX>
- <XXX> is not the same as <XXX>
- <XXX> can not be both <XXX> and <XXX> at the same time
- <XXX> is not <XXX>
- <XXX> can not <XXX>
- <XXX> is <XXX>, not <XXX>

Entailment Templates

- <XXX> is the same as <XXX>
- <XXX> is a type of <XXX>
- <XXX> is a <XXX>
- <XXX> is a rephrasing of <XXX>
- <XXX> so <XXX>

Neutral Templates

- <XXX> does not mean <XXX>
- just because <XXX> does not mean <XXX>
- <XXX> is not necessarily <XXX>
- <XXX> does not imply <XXX>
- not all <XXX> are <XXX>

6.7.5 Limitations and Future Works

In this chapter, the aim is to create personalized explanations that are suited to the background of the end users. It is well-known that people from different cultures may have varying preferences for the form of explanations provided by AI systems. For instance, individuals from western cultures might prefer more straightforward explanations, while those from eastern cultures might lean towards more indirect explanations. Additionally, full-time students might have different expectations for explanations than those with other occupations.

Despite these differences, current NLP generation tasks do not effectively capture the diversity of human expression. The work in this chapter provides a good starting point for generating diverse explanations that maintain the same semantic information yet are structured differently with regard to semantic elements such as words and phrases, as well as syntactic elements such as sentence structure and

grammar. This is demonstrated through the use of interpolation (section 6.6.3) and examples shown in Table 6.8.

However, there are limitations to the proposed approach. It does not explore the possibility of more fine-grained control over the generation of explanations. For example, it would be beneficial to model the users' background or intention to generate explanations specifically tailored to them. This could include factors such as age, gender, occupation, and so on. Fine-grained control in explanation generation has the potential to make a wider impact in the field of XAI and increase the diversity of generated explanations. Further research in this area is necessary and could benefit the overall quality of generated explanations.

Another limitation of this work is evaluating the quality of the generated explanations. Although some efforts have been made to establish qualitative metrics, such as those introduced in [20] for the first 100 test examples, a more proper evaluation of the explanations would require user-based tests. However, due to limitations in time and resources, this research did not conduct any user-based studies. Additionally, since the focus of this research field is not centred on human-computer interaction, conducting user-based studies is left as a potential avenue for future research. Further discussions about the limitations of user-based studies can be found in section 8.3.

6.8 Conclusion

Here, we have presented **INTERACTION**, a novel deep generative XAI framework, with explanations in two steps: (1) Explanation and Label Prediction; and (2) Diverse Evidence Generation. INTERACTION is the first study which, to the best of our knowledge, *addresses the concern on the diversity of human languages* in XAI, within the natural language processing task. INTERACTION achieves competitive or better performance against state-of-the-art baseline models on both generation (4.7% improvement in BLEU) and prediction (4.4% improvement in accuracy) tasks. We observe that label prediction and explanation generation can complement each other, which further confirms the benefits of *XAI with natural*

languages research in practice.

Epilogue

In our third contribution, we explore DLVM for a relatively recent natural language generation problem: the NLI explanation generation task. This task requires extra text-form explanation generated apart from the standard NLI class prediction. The current approach to this task focuses on single explanation generation, and we propose a deep generative XAI framework based on DLVM to generate multiple instances of explanations and make predictions simultaneously. We use a latent variable to model the semantic information of the explanation explicitly, and this allows us to interpolate the latent space to retrieve diverse explanation instances.

We demonstrate that DLVM addresses the concerns over *explainability* (explicit model explanation semantic with latent variable) and *interpretability* (interpolate to generation diverse explanations), and the effectiveness of our approach with improved generation and prediction performance over the state-of-the-art Transformer architecture baseline. Up to this point, in previous chapters, we have focused on NLP applications which can be solved in a supervised learning setting; next, we explored semi-supervised learning for a natural language generation problem in the next chapter (Chapter 7), which allows our model to pick up information from unlabelled data.

CHAPTER 7

Deep Latent Variable Models for Semi-Supervised Paraphrase Generation

Prologue

In Chapter 6, we have explored deep latent variable models (DLVMs) for a natural language generation problem and shown the benefits of using DLVMs in addressing the problem of *interpretability* and *explainability*. In this chapter, we further explore another natural language generation problem: paraphrase generation. Additionally, in Chapter 4, 5 and 6, we have explored deep latent variable models (DLVMs) for text in supervised learning setup. However, specifically for text, having good quality annotated data is rare in industrial settings; in this chapter, we present a study on semi-supervised learning for paraphrase generation tasks with DLVMs.

This is the first study on how deep latent variable models can be used for semi-supervised learning in paraphrase generation tasks. We introduce two novel models: VSAR (unsupervised) and DDL (supervised). The VSAR and DDL model can be combined for semi-supervised learning; however, when we use this combined model(DDL+VSAR), it suffers from a cold start problem when training from

scratch. Hence we present a novel training scheme to deal with this problem when used in semi-supervised learning. We also study semi-supervised learning scenarios with the combined model on the full fraction of data with a fraction of the labelled data.

Declaration: This chapter is based on the following manuscript, which is now under review:

Yu, J., Cristea, A.I., Harit, A., Sun, Z., Aduragba, O.T., Shi, L. and Al Moubayed, N., 2022, October. ***Language as a Latent Sequence: Deep Latent Variable Models for Semi-Supervised Paraphrase Generation.*** *Under review for PLOS One Journal.* (IF: 3.752, Submitted and Under Review)

This chapter is presented largely as the manuscript submitted, although referencing and notation have been altered and cross-referencing added for consistency across this thesis. Some stylistic changes have been made for consistency. The majority of the text is verbatim, with some minor wording and formatting changes.

7.1 Introduction

Paraphrase generation is an important Natural Language Processing (NLP) problem, useful in many NLP applications, such as question answering [270], information retrieval [271], information extraction [272] and summarisation [273]. Natural language itself is complicated and may be expressed in various alternative surface forms of the same underlying semantic content [247, 274]. Hence it is critically important to integrate the paraphrase generation model as a component in real-world NLP systems to offer robust responses to end users' inputs. Traditional solutions to paraphrase generation are generally rule-based [275, 276], utilising lexical resources, such as WordNet [277], to find word replacements. The recent trend brings to fore neural network models [267, 278–280], which are typically based on a sequence-to-sequence learning paradigm [13].

These models have achieved remarkable success for paraphrase generation due

to complex architectures and sophisticated conditioning mechanisms, *e.g. soft, hard and self-attention*. However, the advancement of such models is primarily based on the availability of large-scale labelled data pairs. Instead, this paper explores semi-supervised learning scenarios where only a fraction of the labels are available. This semi-supervised learning setting is favourable and extremely useful for industry scenarios due to the effort of time and money to obtain good quality human annotations. A semi-supervised learning model often consists of two components: an unsupervised learning model and a supervised learning model. For unsupervised learning, we propose a novel deep generative model, motivated by the classic variational autoencoder (VAE) [4–6]. Furthermore, we propose a novel supervised model that can be integrated with our proposed VAE model. Combining both unsupervised and supervised models enable semi-supervised learning by exploiting the VAEs' ability to marginalise latent variables for unlabelled data.

Traditional VAEs typically embed data representations in a fixed latent space, with the general purpose of dimensionality reduction [281]. This paper alternatively considers a latent variable in the form of a discrete language sequence with various lengths. This additionally can enhance the model interpretability, as language is naturally preserved as discrete variables [126]. Following the recent prior works successfully incorporating discrete latent variables to improve paraphrasing [274, 282], we propose a novel model with more expressive discrete latent variable: **variational sequence auto-encoding reconstruction** (VSAR) model introduced in section 7.2.

In order to further boost the model performance on paraphrase generation, motivated by dual learning [283–286], we employ a novel **Dual Directional Learning** (DDL) model trained on labelled data, presented in Section 7.3. The DDL model shares model parameters with the VSAR model and can be jointly optimised for semi-supervised scenarios, and we discuss this in Section 7.4.

In order to further boost the model performance on paraphrase generation, motivated by dual learning [283–286], we employ a novel **Discriminative Dual Learning** (DDL) model trained on labelled data, presented in Section 7.3. The DDL model shares model parameters with the VSAR model and can be jointly optimised

for semi-supervised scenarios, and we discuss this in Section 7.4.

Our contributions in this paper include:

- presenting the first study on semi-supervised learning for paraphrasing with deep latent variable models;
- introducing two novel models: VSAR (unsupervised) and DDL (supervised), which can be combined for semi-supervised learning;
- proposing a novel training scheme (Knowledge Reinforced Learning) to deal with cold start problem in the combined model (DDL+VSAR) when used in semi-supervised learning;
- studying semi-supervised learning scenarios with the combined model on the full fraction of data and empirically showing that our model achieves competitive state-of-the-art results;
- presenting a study of semi-supervised scenarios with a fraction of the labelled data, and our models show significantly better results than very strong supervised baselines.

7.2 Variational Sequence Auto-Encoding Reconstruction (VSAR)

In this section, we present the VSAR model (Figure 7.1)[1]. The model consists of four separate neural network models - a source encoder, a target decoder, a target encoder, and a source decoder. Under the unsupervised learning setup, we only observe source text $\boldsymbol{s}$ and no target text $\boldsymbol{t}$. We reformulate the problem of modelling fully observed source text $\boldsymbol{s}$ as modelling partially observed parallel source text $\boldsymbol{s}$ and its associated latent target pair $\bar{\boldsymbol{t}}$. We adopt Bayesian inference to marginalise the latent target string $\bar{\boldsymbol{t}}$ from the joint probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{s}, \bar{\boldsymbol{t}})$ as shown

[1]The language model prior and weak supervision decoding is omitted for clarity.

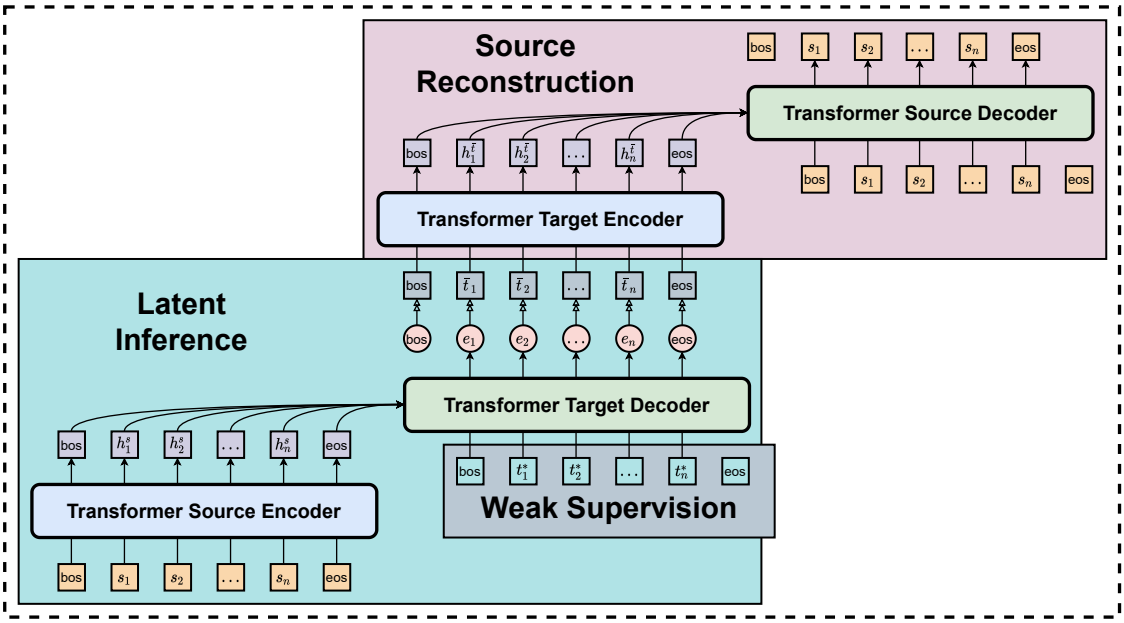

Figure 7.1: Chapter 7: Variational Sequence Auto-Encoding Reconstruction Model.

in Figure 7.1 based on Equation 7.7 and hence only the observed source text $\boldsymbol{s}$ is required for VSAR model.

In the VSAR model, the **latent inference** network, parameterised as $q_{\boldsymbol{\phi}}(\bar{\boldsymbol{t}}|\boldsymbol{s})$, takes source text $\boldsymbol{s}$ and generates a latent target sample $\bar{\boldsymbol{t}}$. The **source reconstruction** network, parameterised as $p_{\boldsymbol{\theta}}(\boldsymbol{s}|\bar{\boldsymbol{t}})$, reconstructs the observed source text $\boldsymbol{s}$ back, based on the latent target sample $\bar{\boldsymbol{t}}$. As the prior distribution, a language model is pre-trained on unlabelled source text corpus to approximate the prior distribution $p(\bar{\boldsymbol{t}})$[2]. The prior is introduced for regularisation purposes [118, 123], which enforces samples are more likely to be reasonable natural language text.

Motivated by the benefits of **parameters sharing** in multi-task learning for natural language generation [287–290], we share model parameters for the source encoder and the target encoder, denoted as $\boldsymbol{f_{encode}}$; similarly, we share model parameters for the source decoder and the target decoder, denoted as $\boldsymbol{f_{decode}}$. In the following sections, we use $\boldsymbol{f_{encode}}$ and $\boldsymbol{f_{decode}}$ to represent all encoders and decoders in the VSAR model, respectively.

[2]We leverage linguistic knowledge of paraphrase generation task, in which a paraphrase text string can be considered as its own paraphrase.

7.2.1 Weak Supervision

In the VSAR model, we empirically found that the quality of latent sequence $\bar{\boldsymbol{t}}$ is very unstable, especially at the beginning of the training. To combat this issue, motivated by the idea of weak supervision [291,292], we propose to use pseudo-labels to guide VSAR throughout training. Before each model forward pass, we first assign pseudo-labels to each token in unobserved latent target sample $\bar{\boldsymbol{t}}$ with the current model parameter. The pseudo-labels are detached from the computational graph; hence no gradient is updated during the weak supervision process. The pseudo-labels can be considered as a weak supervision signal for 'teacher forcing training' [293].

The encoder model takes the source string $\boldsymbol{s} = (s_1, ..., s_n)$ as input and produces its corresponding contextual vector $\boldsymbol{h^s} = (h_1^s, ..., h_n^s)$:

$$\boldsymbol{h^s} = f_{\text{encode}}(\boldsymbol{s}) \tag{7.1}$$

We adopt a greedy decoding scheme to assign pseudo-target labels $\boldsymbol{t^*}$ and assume that a good paraphrase ought to have a similar length as the original sentence [294, 295]; such that $\boldsymbol{t^*} = (t_1^*, ..., t_n^*)$. Let t_i^* be the i^{th} word in the pseudo target sequence; we construct this sequence in an auto-regressive manner:

$$t_i^* = f_{\text{decode}}(\boldsymbol{h^s}; t_{1:i-1}^*) \tag{7.2}$$

7.2.2 Target Inference

Once the pseudo-target labels $\boldsymbol{t^*}$ are assigned, we perform latent variable inference with the latent inference network. Since the source string, $\boldsymbol{s}$ remains the same; we reuse the value of the contextual vector $\boldsymbol{h^s}$ in Section 7.2.1. Let $\bar{t}_j$ be the j^{th} words in the latent sample and e_j be the corresponding output of the target decoder model. We construct the latent sample $\bar{\boldsymbol{t}}$ using contextual vector $\boldsymbol{h_s}$ and all $t_{1:j-1}^*$ words in the pseudo-labels:

$$\begin{aligned} e_j &= f_{\text{encode}}(\boldsymbol{h_s}; t_{1:j-1}^*) \\ \bar{t}_j &\sim \textit{Gumbel}\text{-TOP}k(e_j, \tau) \end{aligned} \tag{7.3}$$

The $\bar{t}_i$ is drawn via the Gumbel-Trick [3, 124] and TOP-k subset sampling technique [145] based on temperature τ, which controls the probability distribution of the samples. At a high temperature τ, we equivalently sample from a uniform distribution; at a low temperature τ, we equivalently sample from a categorical distribution.

We explore two different schemes commonly used in the literature: (1) we use a fixed temperature τ of 0.1, as in [296]; and (2) we gradually anneal the temperature τ from a high temperature of 10 to a low temperature of 0.01, as in [297]. Our empirical results suggest that annealing the temperature τ during training yields significantly better results ($p < .05$; Wilcoxon test) and are thus used to report the final results. We use a k-value of 10 as suggested in [126].

7.2.3 Source Reconstruction

For the source reconstruction network, the encoder model takes the latent target sequence string $\bar{\boldsymbol{t}} = (\bar{t}_1, ..., \bar{t}_n)$ as input and produces its corresponding contextual vector $\boldsymbol{h}^{\bar{\boldsymbol{t}}} = (h_1^{\bar{t}}, ..., h_n^{\bar{t}})$:

$$\boldsymbol{h}^{\bar{\boldsymbol{t}}} = f_{\text{encode}}(\bar{\boldsymbol{t}}) \tag{7.4}$$

Let $\hat{s}_k$ be the k^{th} word in the reconstructed source string, during the training; we retrieve the reconstructed source string $\hat{\boldsymbol{s}}$ via:

$$\hat{s}_k = f_{\text{decode}}(\boldsymbol{h}^{\bar{\boldsymbol{t}}}; s_{1:k-1}) \tag{7.5}$$

7.2.4 Learning and Inference for VSAR

In the SVAR model, there are two sets of parameters, $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, which are required to be updated. Let $\boldsymbol{S}$ be the observed random variable for the source text, $\bar{\boldsymbol{T}}$ be the latent random variable for the target text, and N be the total number of the unlabelled source text. We have the following joint likelihood for the SVAR model, parameterised by $\boldsymbol{\theta}$:

$$P(\boldsymbol{S}, \bar{\boldsymbol{T}}; \boldsymbol{\theta}) = \prod_{i=1}^{N} P(s_{(i)}|\bar{t}_{(i)}; \boldsymbol{\theta}) P(\bar{t}_{(i)}) \tag{7.6}$$

The log marginal likelihood $\boldsymbol{L_1}$ of the observed data that we will be approximated during training is $\log p(\boldsymbol{S}; \boldsymbol{\theta})$. We adopt amortised variational inference [4–6] and build a surrogate function approximated with a neural network $q(\bar{\boldsymbol{T}}|\boldsymbol{S}; \boldsymbol{\phi})$, parameterised by $\boldsymbol{\phi}$, to derive the evidence lower bound (ELBO) for the joint likelihood:

$$\begin{aligned} \boldsymbol{L_1} &= \log \sum_{\bar{\boldsymbol{T}}} p(\boldsymbol{S}, \bar{\boldsymbol{T}}; \boldsymbol{\theta}) \geq \mathcal{L}_{\text{ELBO}}(\boldsymbol{S}, \bar{\boldsymbol{T}}; \boldsymbol{\theta}, \boldsymbol{\phi}) \\ &= \sum_{i=1}^{N} \{\mathbb{E}_{q(\bar{t}|s_{(i)};\boldsymbol{\phi})}[\log p(s_{(i)}|\bar{t}; \boldsymbol{\theta})] - \mathbb{D}_{KL}[q(\bar{t}|s_{(i)}; \boldsymbol{\phi})||p(\bar{t})]\} \end{aligned} \tag{7.7}$$

The most common variational family in the VAE framework relies on the reparameterisation trick [4], which is not applicable to the non-differentiable discrete latent variable. An approach for optimising learning with such latent variables uses the REINFORCE algorithm [6, 122]; however, this algorithm generally suffers from high variance. In this paper, we instead use Gumbel-Softmax [3, 124] with differentiable subset sampling [145] to retrieve top-k samples without replacement. Nevertheless, since sampling a one-hot form vector induces high variance, we apply the straight-through technique [191] as a biased estimator of the gradient, to combat this variance.

During training, while optimising the log-likelihood, we perform learning ($\boldsymbol{\theta}$) and inference ($\boldsymbol{\phi}$) at the same time. The parameters are jointly optimised with the same optimiser. Since we are sharing parameters in our model, in practice, we are updating the same set of parameters (shared by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$) with source data only.

7.3 Dual Directional Learning (DDL)

In this section, we introduce the Dual Directional Learning (DDL) model, which we use for supervised paraphrase generation. The DDL model consists of two sets of standard Transformer models [81], each with its own separate neural networks - an encoder and a decoder. We perform standard sequence-to-sequence learning,

with fully observed parallel source text $\boldsymbol{s}$ and its associated target pair $\boldsymbol{t}$, in dual directions. The **target generation** network $p_{\boldsymbol{\theta}_{\boldsymbol{t}|\boldsymbol{s}}}(\boldsymbol{t}|\boldsymbol{s})$ takes source text $\boldsymbol{s}$ as input and generates target text $\boldsymbol{t}$ and the **source generation** network $p_{\boldsymbol{\theta}_{\boldsymbol{s}|\boldsymbol{t}}}(\boldsymbol{s}|\boldsymbol{t})$ takes target text $\boldsymbol{t}$ as input and generates source text $\boldsymbol{s}$.

7.3.1 Parameter Learning

In the DDL model, there are two sets of parameters, $\boldsymbol{\theta}_{\boldsymbol{s}|\boldsymbol{t}}$ and $\boldsymbol{\theta}_{\boldsymbol{t}|\boldsymbol{s}}$, which are required to be updated. Let $\boldsymbol{S}$ be the observed random variable for source text, $\boldsymbol{T}$ be the observed random variable for target text, and M be the number of labelled pairs; we then have the following conditional likelihood for our DDL model:

$$\begin{aligned} P(\boldsymbol{S}|\boldsymbol{T};\boldsymbol{\theta}_{\boldsymbol{s}|\boldsymbol{t}}) &= \prod_{i=1}^{M} P(s_{(i)}|t_{(i)};\boldsymbol{\theta}_{\boldsymbol{s}|\boldsymbol{t}}) \\ P(\boldsymbol{T}|\boldsymbol{S};\boldsymbol{\theta}_{\boldsymbol{t}|\boldsymbol{s}}) &= \prod_{i=1}^{M} P(t_{(i)}|s_{(i)};\boldsymbol{\theta}_{\boldsymbol{t}|\boldsymbol{s}}) \end{aligned} \tag{7.8}$$

The log conditional likelihood $\boldsymbol{L_2}$ of the observed data pairs can be jointly learnt during training as:

$$\boldsymbol{L_2} = \sum_{i=1}^{M} (\log P(s_{(i)}|t_{(i)};\boldsymbol{\theta}_{\boldsymbol{s}|\boldsymbol{t}}) + \log P(t_{(i)}|s_{(i)};\boldsymbol{\theta}_{\boldsymbol{t}|\boldsymbol{s}})) \tag{7.9}$$

During training, we perform dual learning ($\boldsymbol{\theta}_{\boldsymbol{s}|\boldsymbol{t}}$ and $\boldsymbol{\theta}_{\boldsymbol{t}|\boldsymbol{s}}$) at the same time and the parameters are jointly optimised with the same optimiser.

7.3.2 Parameter Sharing

Once again, motivated by the benefits of multi-task learning for natural language generation [287–290], we share model parameters for the target generation and the source generation network. Although sharing parameters is a very simple technique, as shown in Table 7.1 and Table 7.2, the DDL model significantly improves the performance of paraphrase generation with respect to the Transformer baseline ($p < .05$; Wilcoxon test), which only handles sequence to sequence learning in a single direction.

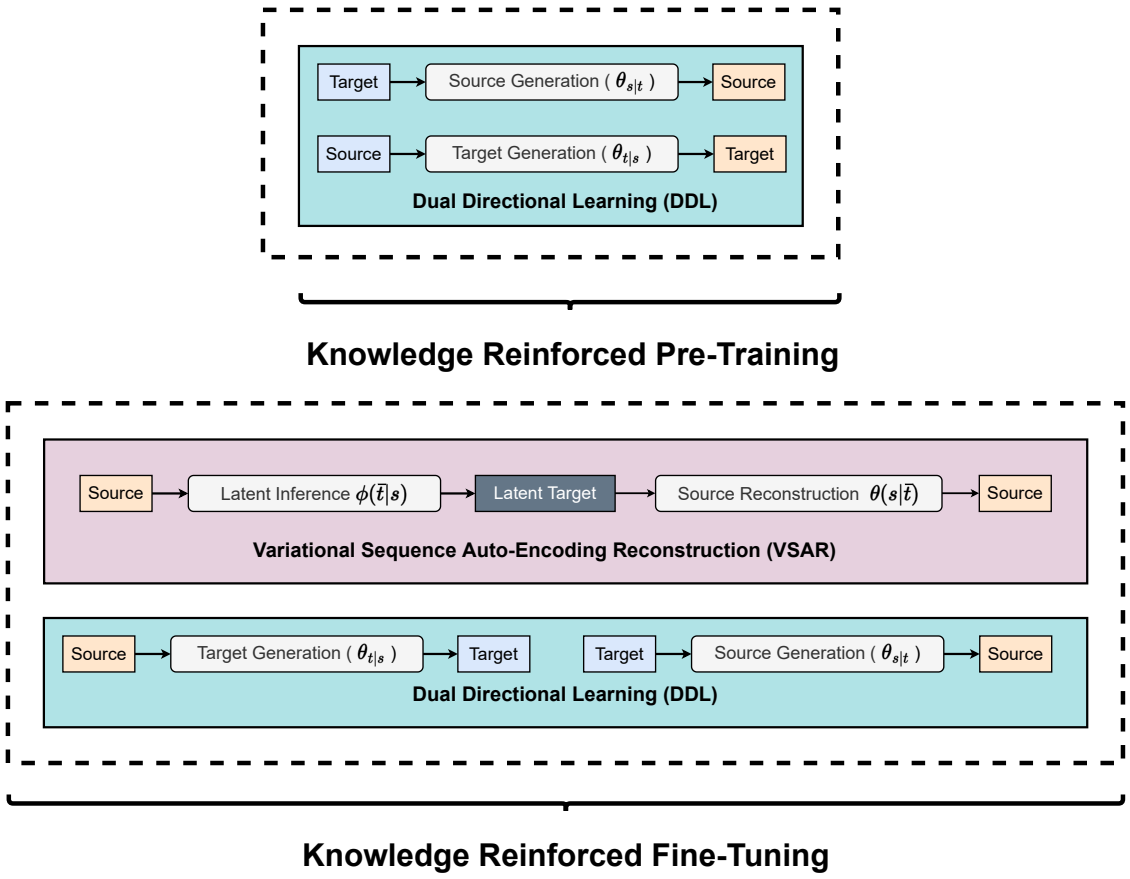

Figure 7.2: Chapter 7: Knowledge Reinforced Learning.

7.4 Combining VSAR and DDL for Semi-supervised Learning

In this section, we introduce our semi-supervised learning model (VSAR+DDL), which combines models presented in sections 7.2 and 7.3. For semi-supervised learning, the log-likelihood of the data can be expressed as follow:

$$\begin{aligned} \boldsymbol{L} &= \boldsymbol{L_1} + \boldsymbol{L_2} \\ &= \sum_{i=1}^{N} \{\mathbb{E}_{q(\bar{t}|s_{(i)};\boldsymbol{\phi})}[\log P(s_{(i)}|\bar{t};\boldsymbol{\theta})] - \mathbb{D}_{KL}[q(\bar{t}|s_{(i)};\boldsymbol{\phi})||P(\bar{t})]\} \\ &+ \sum_{i=1}^{M} (\log P(s_{(i)}|t_{(i)};\boldsymbol{\theta_{s|t}}) + \log P(t_{(i)}|s_{(i)};\boldsymbol{\theta_{t|s}})) \end{aligned} \tag{7.10}$$

As suggested in equation 7.10, for unsupervised learning and supervised learning, the likelihood function involves the same set of conditional probability between $\boldsymbol{s}$ and

$\boldsymbol{t}$. We hypothesise that sharing parameters between these two models is beneficial, and we share two sets of neural network parameters from the VSAR and DDL models (i.e. $q_{\boldsymbol{\phi}}(\bar{\boldsymbol{t}}|\boldsymbol{s}) \equiv p_{\boldsymbol{\theta}_{\boldsymbol{t}|\boldsymbol{s}}}(\boldsymbol{t}|\boldsymbol{s})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{s}|\bar{\boldsymbol{t}}) \equiv p_{\boldsymbol{\theta}_{\boldsymbol{s}|\boldsymbol{t}}}(\boldsymbol{s}|\boldsymbol{t})$). This allows the strong supervision signal from the DDL model to directly contribute to the VSAR model. At the same time, the unsupervised signal from the VSAR model can benefit the generalisation of the DDL model.

7.4.1 Knowledge Reinforced Learning

Our empirical experiments suggest that our combined model (DDL+VSAR) suffers from a cold-start problem for parameter optimisation when conducting semi-supervised learning from scratch. We found that a key to the success of our model is to have **better initialisation** of the model weight. Hence, we present a novel training scheme called **knowledge reinforced learning** (Figure 7.2), which includes two-stage training. In stage one (pre-training), we conduct supervised learning with our DDL model on paired training sets, as demonstrated in Algorithm 1. In stage two (fine-tuning), we initialise the VSAR model parameter with the best performance DDL model from stage one; and we conduct semi-supervised learning with labelled and unlabelled data, as demonstrated in Algorithm 2. The intuition is to inject better preliminary information into training the SVAR model.

7.4.2 Effect of Language Model Prior

In literature [112, 118, 123, 298], a language model prior is introduced for regularisation purposes, which enforces samples to more likely contain a 'reasonable' natural language. Hence, we adopt the same approach and use a prior in our model. We empirically found the prior useful when the labelled dataset is relatively small. However, surprisingly, we found that training without a prior in the VSAR model yields better results when the dataset is large with our parameter initialisation method. The improvement is significant ($p < .05$; Wilcoxon test), as shown in Table 7.3 and Table 7.4. We report the results without language model prior as DDL +VSAR*, and the log-likelihood becomes:

Algorithm 1 Knowledge Reinforced Pre-Training

Input:
Supervised Training Data ($\mathcal{D}_T^S = \{(s_1, t_1), ..., (s_N, t_N)\}$), Supervised Validation Data ($\mathcal{D}_V^S$)
Parameter:
DDL Model: $\boldsymbol{\theta_{s|t}}$ and $\boldsymbol{\theta_{t|s}}$
Parameter Sharing:
Set $\boldsymbol{\theta_{s|t}}$ equals to $\boldsymbol{\theta_{t|s}}$ through out knowledge reinforced pre-training
Output: $\boldsymbol{\theta_{s|t}}^*$ and $\boldsymbol{\theta_{t|s}}^*$

1: Initialise $\boldsymbol{\theta_{s|t}}$ and $\boldsymbol{\theta_{t|s}}$ with a random seed; set maximum training epochs as $\boldsymbol{T}$; set $\boldsymbol{L_2}^* = 0$
2: **while** Maximum epochs not reached **do**
3: Update $\boldsymbol{\theta_{s|t}}$ and $\boldsymbol{\theta_{t|s}}$ with mini-batch data from $\mathcal{D}_T^S$ based on Equation 7.9
4: **if** $\boldsymbol{L_2}$ in Equation 7.9 calculated based on $\mathcal{D}_V^S$ bigger than $\boldsymbol{L_2}^*$ **then**
5: Set $\boldsymbol{L_2}^* \leftarrow \boldsymbol{L_2}$
6: Set $\boldsymbol{\theta_{s|t}}^* \leftarrow \boldsymbol{\theta_{s|t}}$
7: Set $\boldsymbol{\theta_{t|s}}^* \leftarrow \boldsymbol{\theta_{t|s}}$
8: **end if**
9: **end while**
Return: $\boldsymbol{\theta_{s|t}}^*$ and $\boldsymbol{\theta_{t|s}}^*$

$$\boldsymbol{L}^* = \sum_{i=1}^{N} \{\mathbb{E}_{q(\bar{t}|s_{(i)};\boldsymbol{\phi})}[\log P(s_{(i)}|\bar{t};\boldsymbol{\theta})]\} + \sum_{i=1}^{M} (\log P(s_{(i)}|t_{(i)};\boldsymbol{\theta_{s|t}}) + \log P(t_{(i)}|s_{(i)};\boldsymbol{\theta_{t|s}})) \tag{7.11}$$

To further investigate this issue, we conducted experiments to compare the performance of semi-supervised learning when training with Equation 7.10 (with prior) and 7.11 (without prior) under different data portion setting. We empirically found that with a low portion of labelled data, the combined model (DDL+VSAR) with a prior grant significantly ($p < .05$; Wilcoxon test) better performances and is more stable. This aligns with the observations in [112, 118, 123, 298]. However, with a large portion of labelled data, the combined model (DDL+VSAR) without the prior is significantly ($p < .05$; Wilcoxon test) better.

We argue that this phenomenon relates to our choice of the prior as it is pre-trained on unlabelled source text corpus instead of on the target text corpus. This approximation leads to a distribution shift from the true prior distribution $p(\bar{\boldsymbol{t}})$. Thus, when a low portion of labelled data is used in Algorithm 1, the final DDL

Algorithm 2 Knowledge Reinforced Fine-Training

Input:
Unsupervised Data ($\mathcal{D}^U = \{s_1, ..., s_M\}$)
Supervised Training Data ($\mathcal{D}_T^S = \{(s_1, t_1), ..., (s_N, t_N)\}$), Supervised Validation Data ($\mathcal{D}_V^S$)
Parameter:
VSAR Model: $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$; DDL Model: $\boldsymbol{\theta_{s|t}}$ and $\boldsymbol{\theta_{t|s}}$
Parameter Sharing:
Set $\boldsymbol{\phi}$ equals to $\boldsymbol{\theta_{t|s}}$; $\boldsymbol{\theta}$ equals to $\boldsymbol{\theta_{s|t}}$; and $\boldsymbol{\theta_{s|t}}$ equals to $\boldsymbol{\theta_{t|s}}$ through out knowledge reinforced fine-tuning
Output: $\boldsymbol{\theta_{s|t}}^{**}$, $\boldsymbol{\theta_{t|s}}^{**}$; $\boldsymbol{\phi}^{**}$ and $\boldsymbol{\theta}^{**}$

Initialise $\boldsymbol{\phi}$ and $\boldsymbol{\theta_{t|s}}$ with $\boldsymbol{\theta_{t|s}}^{*}$; and initialise $\boldsymbol{\theta}$ and $\boldsymbol{\theta_{s|t}}$ with $\boldsymbol{\theta_{s|t}}^{*}$; set maximum training epochs as $\boldsymbol{T}$; set $\boldsymbol{L_2}^{*} = 0$.
while Maximum epochs not reached **do**
  Update $\boldsymbol{\theta_{s|t}}$ and $\boldsymbol{\theta_{t|s}}$ with mini-batch data from $\mathcal{D}_T^S$ based on Equation 7.9
  Update $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ with mini-batch data from $\mathcal{D}^U$ based on Equation 7.7
  if $\boldsymbol{L_2}$ in Equation 7.9 calculated based on $\mathcal{D}_V^S$ bigger than $\boldsymbol{L_2}^{*}$ **then**
    Set $\boldsymbol{L_2}^{*} \leftarrow \boldsymbol{L_2}$
    Set $\boldsymbol{\theta_{s|t}}^{**} \leftarrow \boldsymbol{\theta_{s|t}}$
    Set $\boldsymbol{\theta_{t|s}}^{**} \leftarrow \boldsymbol{\theta_{t|s}}$
    Set $\boldsymbol{\phi}^{**} \leftarrow \boldsymbol{\phi}$
    Set $\boldsymbol{\theta}^{**} \leftarrow \boldsymbol{\theta}$
  end if
end while
Return: $\boldsymbol{\theta_{s|t}}^{**}$, $\boldsymbol{\theta_{t|s}}^{**}$; $\boldsymbol{\phi}^{**}$ and $\boldsymbol{\theta}^{**}$

parameters $\boldsymbol{\theta_{s|t}}^{*}$ and $\boldsymbol{\theta_{t|s}}^{*}$ for initialisation VSAR model in Algorithm 2 is not good enough. The prior, in this case, can still benefit the combined model in the semi-supervised learning setting. However, with a large portion of labelled data, the initialisation is good enough, and the distribution shift can harm the combined model in this case.

7.4.3 Semi-supervised Learning Setup

Under the semi-supervised learning setting, we limit the size of the supervised source and target pairs to be less than or equal to the unsupervised source text ($M \leq N$), as we could otherwise just conduct supervised learning to take full advantage of observed data pairs. Experimental results under this setting are presented in Table 7.1 and Table 7.2.

7.5 Related Work

7.5.1 Paraphrase Generation

Paraphrases express the surface forms of the underlying semantic content [274] and capture the essence of language diversity [299]. Early work on automatic generation of paraphrase are generally rule-based [275, 276], but the recent trend brings to fore neural network solutions [16, 126, 274, 278–280, 282]. Current research for paraphrasing mainly focuses on supervised methods, which require the availability of a large number of source and target pairs. In this work, we alternatively explore a semi-supervised paraphrasing method, where only a fraction of source and target pairs are observed, and where a large number of unlabelled source text exists. We made an assumption that each missing target text can be considered as a latent variable in deep generative models. In this paper, we present two models and combine them for paraphrasing: one for unsupervised learning and one for supervised learning. Our combined model extends [126] and models jointly the distribution of source and target, instead of the conditional probability of a target, given the source. Furthermore, our combined model is associated with prior works that introduce a discrete latent variable [274, 282], and it uses an arguably more expressive latent variable, in the form of language.

7.5.2 Deep Latent Variable Models for Text

Deep latent variable models (DLVMs) have been studied for text modelling [94, 134]. The most common and widely adopted latent variable model is the standard VAE model with a Gaussian prior [104], which suffers from posterior collapse [110, 111]. Multiple studies have been conducted to combat this issue [112, 113, 116]. In particular, β-VAE [112] introduces a penalty term to balance VAE reconstruction and prior regularisation intuitively and is adopted as one of our baselines.

While much of the research focuses on continuous latent variable models, the text is naturally presented in discrete form and may not be well represented with continuous latent variables. Early work on discrete deep latent variable models [118, 119] adopted the REINFORCE algorithm [6, 122]; however, it suffers from very high

variance. With the recent advancement in statistical relaxation techniques, Gumbel-Trick [3, 124] was utilised, to model discrete structures in the latent variable model of the text [123, 125–127]. Our work adopts Gumbel-Trick with subset sampling for natural language generation tasks and, for the first time, studies latent variables as a discrete language sequence for the paraphrasing task. Our proposed model is strongly associated with [118, 123]; however, we study the problem under the semi-supervised setup for the paraphrase generation tasks. Furthermore, we present a novel inference algorithm (our knowledge reinforced learning scheme) to help aid learning in deep generative models and achieve competitive performance for both full data and data fraction settings.

7.6 Experiments

Here, we describe the datasets, experimental setup, evaluation metrics and experimental results.

7.6.1 Datasets

MSCOCO [300]: This dataset has been widely adopted to evaluate paraphrase generation methods and contains human-annotated captions of images. Each image is associated with five captions from different annotators, who describe the most prominent object or action in an image. We use the 2017 version for our experiments; from the five captions accompanying each image, we randomly choose one as the source string and one as the target string for training. We randomly choose one as the source string for testing and used the rest four as references.

Quora[3]: This dataset consists of 150K lines of question duplicate pairs, and it has been used as a benchmark dataset for paraphrase generation since 2017. However, since this dataset does not contain a specific split for training and testing, prior models are evaluated based on different subset sizes of data.

For both datasets (MSCOCO and Quora), in order to improve the reproducibility

[3]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

of our results, we use a pre-trained tokenizer ('bert-base-uncased' version) from [54][4] and set the maximum token length as 20 (by removing the tokens beyond the first 20). Following [126, 280, 301], we use training, validation and test sets as 100K, 4K and 20K for Quora dataset; and 93K, 4K and 20K for MSCOCO. For the complementary study in Table 7.5 and Table 7.6, we use training, validation and test sets as 100K, 24K and 24K for Quora dataset; and 100K, 5K and 5K for MSCOCO, in order to have a fair comparison with the results reported in [274, 282].

7.6.2 Baselines

We consider several state-of-the-art baselines, presented in Table 7.3, Table 7.4, Table 7.5, and Table 7.6. Note that these experimental results are directly taken from [280][5] and [274]. For evaluation, we start with our implementation of the Transformer model as the absolute baseline, which achieves competitive performance as reported in [280]. The Transformer model [81] is considered as the SOTA model, which is very 'hard to beat'. We report our model performance based on a similar setup as in [280] and [274].

7.6.3 Experimental Setup

In this section, we introduce our primary experimental setup. We do not use any external word embedding such as Glove [45], word2vec [42] or BERT [54] for initialisation; rather, we obtain word embedding with end-to-end training, in order not to use any prior knowledge and better understand the impact of our model. We use the 'base' version of the Transformer model [81], which is a 6-layer model with 512 hidden units and 8 heads for each encoder and decoder network. In each encoder and decoder, we have a separate learnable position embedding and its associated word embedding component.

We use a greedy decoding scheme for paraphrase generation, which is fast and cheap to compute. For model optimisation, we use Adam [182] as our optimiser

[4]https://github.com/huggingface/transformers
[5]The authors do not make their code publicly available.

with default hyper-parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$). We conduct all the experiments with a batch size of 512 for the Quora and MSCOCO datasets. We set the learning rate as $1e-4$ for MSCOCO and $2e-4$ for Quora based on empirical experiments. All experiments are run for a maximum of 30 epochs on NVidia GPU Cluster with A100 GPU. Experiments are repeated three times with different random seeds (1000, 2000 and 3000) and the average result is reported in Tables 1-6.

7.6.4 Evaluation

In this paper, we evaluate our models based on quantitative metrics: BLEU [261][6], ROUGE [302][7], and i-BLEU [303]. BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores are based on 'n-gram' coverage between system-generated paraphrase(s) and reference sentences. They have been used widely to automatically evaluate the quality and accuracy of natural language generation tasks.

Previous work has shown that automatic evaluation metrics can perform well for paraphrase identification tasks [304] and correlate well with human judgements in evaluating generated paraphrases [305]. Recent papers introduce additional i-BLEU [303] metrics to balance the fidelity of generated outputs to reference paraphrases (BLEU) as well as the level of diversity introduced (self-B). For all metrics apart from self-B, the higher the value, the better the model performs.

7.6.5 Results and Discussion

Learning with Unlabelled Data Only

In an initial experiment, we explored the ability of the VSAR model to perform paraphrase generation tasks using only unlabelled data. This experiment was conducted to see if the model could accurately capture the information required for paraphrase generation without the aid of labelled data. However, the results of the experi-

[6]https://www.nltk.org/
[7]https://github.com/huggingface/datasets/tree/master/metrics/rouge

ment showed that the VSAR model alone was not able to produce high-quality paraphrases and often resulted in sentences that were either incomprehensible or meaningless. These results led us to pursue a semi-supervised learning solution for paraphrase generation, which would provide the model with guidance from labelled data. For unsupervised learning with VSAR, we found that while the lower bound indicated by $\boldsymbol{L}_1$ (Equation 7.7) decreased during training in a fully unsupervised setting, the model still generated low-quality paraphrases. This further validated the need for a semi-supervised learning solution, which is introduced in section 7.4 of this paper.

Model	Labelled	Unlabelled	B-1	B-2	B-3	B-4	i-B	R-1	R-2	R-L
DDL	20K	–	46.68	33.44	25.46	20.18	11.08	47.57	25.42	45.50
DDL+VSAR$_1$	20K	20K	47.80 ↑	34.33 ↑	26.17 ↑	20.76 ↑	11.25 ↑	48.03 ↑	25.82 ↑	45.84 ↑
DDL+VSAR$_2$	20K	100K	**50.26** ↑	**36.87** ↑	**28.50** ↑	**22.82** ↑	**11.60** ↑	**51.51** ↑	**28.45** ↑	**49.07** ↑
DDL	50K	–	53.31	40.22	31.70	25.80	**13.80**	**55.63**	**32.15**	**53.13**
DDL+VSAR$_1$	50K	50K	53.33 ↑	39.93 ↓	31.39 ↓	25.49 ↓	13.45 ↓	55.51 ↓	31.90 ↓	52.95 ↓
DDL+VSAR$_2$	50K	100K	**53.79** ↑	**40.47** ↑	**31.86** ↑	**25.93** ↑	13.67 ↓	55.58 ↓	31.89 ↓	52.93 ↓

Table 7.1: Chapter 7: Semi-Supervised Learning Experiment Results for Quora.

Model	Labelled	Unlabelled	B-1	B-2	B-3	B-4	i-B	R-1	R-2	R-L
DDL	20K	–	66.82	47.25	33.14	23.75	16.66	40.53	14.95	36.94
DDL+VSAR$_1$	20K	20K	66.98 ↑	47.28 ↑	33.10 ↓	23.72 ↓	16.54 ↓	40.60 ↑	14.95 ↑	36.94 ↑
DDL+VSAR$_2$	20K	93K	**67.64** ↑	**48.00** ↑	**33.96** ↑	**24.55** ↑	**16.68** ↑	**40.87** ↑	**15.12** ↑	**37.01** ↑
DDL	50K	–	69.39	50.17	36.06	26.49	18.43	42.08	16.31	38.27
DDL+VSAR$_1$	50K	50K	69.43 ↑	50.21 ↑	36.08 ↑	26.45 ↓	18.31 ↓	42.20 ↑	16.33 ↑	38.31 ↑
DDL+VSAR$_2$	50K	93K	**69.91** ↑	**50.65** ↑	**36.52** ↑	**26.93** ↑	**18.51** ↑	**42.39** ↑	**16.46** ↑	**38.40** ↑

Table 7.2: Chapter 7: Semi-Supervised Learning Experiment Results for MSCOCO.

Learning with a Fraction of Data

In this section, we present results which are based on a fraction of labelled data in Table 7.1 and Table 7.2. In both tables, we present the results of two models - the supervised learning model, DDL and the semi-supervised learning model, DDL + VSAR. In a semi-supervised learning setting, VSAR is trained on unlabelled data, and DDL is trained on labelled data. The DDL+VSAR$_1$ model employs equivalent-sized labelled and unlabelled datasets, which come from the same source and target pairs, so there is no additional information applied in this case. The DDL+VSAR$_2$ model employs the full unlabelled dataset in addition to the existing labelled dataset, which is the true semi-supervised setting.

Results suggest that the DDL+VSAR$_1$ model achieves competitive or better performance on most metrics' scores compared to the supervised DDL model only trained on labelled data; especially with a lower fraction of the data (for example, the significant improvement for 20K is more noticeable than for 50K). Furthermore, fixing the labelled data size, the DDL+VSAR$_2$ model achieves significantly better performance by using additional unlabelled data than all other models reported in both tables ($p < .05$; Wilcoxon test), which means the semi-supervised learning does work in this scenario.

Model	B-1	B-2	B-3	B-4	i-B	R-1	R-2	R-L
Upper Bound (Copy Source)	63.36	49.99	40.47	33.54	-	63.04	38.15	59.64
Lower Bound (Random Select)	16.10	4.50	1.94	0.79	-	9.13	1.54	8.79
Residual-LSTM [306]	53.59	39.49	30.25	23.69	15.93	55.10	33.86	53.61
β-VAE [112]	47.86	33.21	24.96	19.73	10.28	47.62	25.49	45.46
Transformer [81]	53.56	40.47	32.11	25.01	17.98	57.82	32.58	56.26
LBOW-TOPk [126]	55.79	42.03	32.71	26.17	19.03	58.79	34.57	56.43
IANet+X [280]	56.06	42.69	33.38	26.52	19.62	59.33	35.01	57.13
Transformer (our implementation)	54.73	41.59	32.96	26.94	14.50	56.90	33.28	54.29
DDL (our model)	55.97 ↑	43.02 ↑	34.32 ↑	28.19 ↑	14.83 ↑	**58.80** ↑	35.00 ↑	56.11 ↑
DDL + SVAR (our model)	55.79 ↑	42.79 ↑	34.11 ↑	28.01 ↑	**14.92** ↑	58.61 ↑	34.75 ↑	55.91 ↑
DDL + SVAR* (our model)	**55.99** ↑	**43.05** ↑	**34.37** ↑	**28.23** ↑	14.81 ↑	58.79 ↑	**35.02** ↑	**56.14** ↑

Table 7.3: Chapter 7: Main Experiment Results for Quora.

Model	B-1	B-2	B-3	B-4	i-B	R-1	R-2	R-L
Upper Bound (Copy Source)	64.97	44.90	30.69	21.30	-	39.18	12.96	34.61
Lower Bound (Random Select)	32.34	10.99	3.81	1.68	-	17.58	1.51	16.27
Residual-LSTM [306]	70.24	48.65	34.04	23.66	18.72	41.07	15.26	37.35
β-VAE [112]	70.04	47.59	32.29	22.54	18.34	40.72	14.75	36.75
Transformer [81]	71.31	49.86	35.55	24.68	19.81	41.49	15.84	37.09
LBOW-TOPk [126]	72.60	51.14	35.66	25.27	21.07	42.08	16.13	38.16
IANet+X [280]	72.10	52.22	37.39	26.06	21.28	43.81	16.35	39.65
Transformer (our implementation)	68.72	49.64	35.87	26.63	18.59	42.09	16.53	38.35
DDL (our model)	70.75 ↑	51.72 ↑	37.62 ↑	27.95 ↑	19.37 ↑	43.00 ↑	17.01 ↑	39.06 ↑
DDL + SVAR (our model)	70.84 ↑	51.84 ↑	37.75 ↑	28.04 ↑	19.39 ↑	**43.05** ↑	**17.04** ↑	**39.07** ↑
DDL + SVAR* (our model)	**70.99** ↑	**51.91** ↑	**37.82** ↑	**28.12** ↑	**19.39** ↑	43.00 ↑	17.03 ↑	39.02 ↑

Table 7.4: Chapter 7: Main Experiment Results for MSCOCO.

Model	B-4	self-B	i-B
Separator [282]	23.68	24.20	14.10
HRQ-VAE [274]	33.11	40.35	18.42
Transformer (our implementation)	26.92	**35.33**	14.47
DDL + SVAR (our model)	28.15 ↑	38.92 ↓	**14.73** ↑
DDL + SVAR* (our model)	**28.16** ↑	39.07 ↓	14.71 ↑

Table 7.5: Chapter 7: Complement Results for Quora.

Model	B-4	self-B	i-B
Separator [282]	20.59	12.76	13.92
HRQ-VAE [274]	27.90	16.58	19.04
Transformer (our implementation)	26.87	**13.50**	18.79
DDL + SVAR (our model)	27.87 ↑	15.42 ↓	19.21 ↑
DDL + SVAR* (our model)	**27.92** ↑	15.21 ↓	**19.29** ↑

Table 7.6: Chapter 7: Complement Results for MSCOCO.

Learning with Complete Data

In this section, we present results based on all labelled data in Table 7.3 and Table 7.4. Each table comes with three sections. In the first section, we present an upper bound (copying the source as a paraphrase) and a lower bound (randomly selecting ground truth as a paraphrase) calculated based on the test split. This is used as an indication of how well the model performs. In the second section, we present major state-of-the-art models published in recent years. In the third section, we present our own implementation of the Transformer model, which we consider as our absolute baseline, and present results for our models. Our implementation is competitive with the ones reported in recent papers. For our models, DDL is our supervised model, DDL+VSAR is our semi-supervised model, and DDL+VSAR* is our model with no prior use. Compared with state-of-the-art supervised models, our models achieve better BLEU scores and competitive Rouge scores for both datasets. Our complementary experimental results are presented in Table 7.5 and Table 7.6, which we compare with two more recent state-of-the-art models. Our models once again achieve better or competitive performance than the reported, which means our semi-supervised model is competitive with state-of-the-art supervised baselines.

Qualitative Evaluation for Supervised Learning with labelled Data

In table 7.7, we present examples from the test set on Quora data, based on the outputs of our proposed supervised learning model DDL (introduced in section 7.3) and a very strong baseline model Transformer (denoted as TRANS), using varying amounts of training data. The table shows the source and golden reference first, followed by the outputs of the models trained on 20K, 50K, and 100K labelled pairs

Source:	what are best courses for journalism ?
Reference:	what are the best courses on journalism ?
TRANS (20K)	which is the best software for beginner ?
DDL (20K)	what is the best digital marketing course ?
TRANS (50K)	what are the best courses on nagpur ?
DDL (50K)	what are the best courses for journalism ?
TRANS (100K)	what are the best courses about journalism ?
DDL (100K)	what are the best courses for journalism ?
Source:	what helps asthma without an inhaler ?
Reference:	what are some ways to help someone with asthma without an inhaler ?
TRANS (20K)	what happens if a range of a range ofr collides with a range ofr 000r
DDL (20K)	how can i save my without doing waves in a month ?
TRANS (50K)	what can be done to work in a ppr ?
DDL (50K)	what are some of the uses an asthma without an inhaler ?
TRANS (100K)	what is the procedure to be an emergency inr ?
DDL (100K)	how can i help asthma without an inhaler ?
Source:	how can i get 1 million users to sign up to my app ?
Reference:	how can i get a million users on my social app ?
TRANS (20K)	how can i get a friend store ?
DDL (20K)	how can i get a game of app on my app ?
TRANS (50K)	how can i get a million million on my startup ?
DDL (50K)	how do i get the first million users app in my app ?
TRANS (100K)	how can i get a million users to write an app ?
DDL (100K)	how do i get a million users to sign up for my app
Source:	can anyone suggest me the best laptop under 35 k in india ?
Reference:	which is the best laptop under 35 , 000 inr ?
TRANS (20K)	what are the best laptop options available for a laptoprsrsrsrsrsrsrsrsrsrs
DDL (20K)	which is the best laptop to buy in india ?
TRANS (50K)	which is the best laptop under rs . 50000 in india ?
DDL (50K)	what are the best laptops under 35 , 000 in india ?
TRANS (100K)	which is the best laptop to buy under rs . 50000 in india ?
DDL (100K)	which is the best laptop to buy under 35 , 000 in india ?
Source:	how do i manage my microsoft account ?
Reference:	how can i do manage my microsoft account ?
TRANS (20K)	how do i manage my google account ?
DDL (20K)	how do i manage my microsoft office ?
TRANS (50K)	how do i manage my ip address ?
DDL (50K)	how do i manage microsoft microsoft office ?
TRANS (100K)	how do i manage my microsoft account ?
DDL (100K)	how do i manage my microsoft account ?

Table 7.7: Chapter 7: Selected paraphrase generation results for Transformer (TRANS) versus DDL model with different amounts of labelled data (denoted in brackets), represented in the case of Quora dataset.

from Quora. Each example was generated based on a random seed setting of 1000, ensuring a fair qualitative evaluation.

It is clear from the results that the generated paraphrase is more accurate and matches better with the reference as more labelled data is used by the model. Additionally, our DDL model demonstrated a clear advantage over the TRANS model

in capturing the essence of the information, as seen in its ability to capture critical details (e.g. capture asthma and inhaler in the second example; capture number 35000 instead of 50000 in the fourth example). The DDL model also showed more efficient learning, as it was able to achieve comparable results using 50K data as opposed to 100K data for the TRANS model in several examples. These observations further reinforce the effectiveness and significance of our contribution to the field with the proposed DDL model in section 7.3.

Qualitative Evaluation for Incorporating Unlabelled Data

In table 7.8, we presented examples from the test set on Quora data based on model outputs from our proposed semi-supervised learning model DDL + VSAR (from section 7.4) with the same amount of labelled data (20K, same instance also) plus difference size of unlabelled data (20K, 50K and 100K). For comparison, model output with DDL (section 7.3), trained with the same 20K examples, is provided. Similarly, as in table 7.7, we use the same random seed of 1000 to generate the examples. In this table, we first presented the source and golden reference, followed by model outputs trained based on 20K labelled pairs by the DDL model, the same 20K pairs used for the DDL+VSAR model (similar to DDL+VSAR$_1$ setting in table 7.1 and 7.2), the same 20K with 50K unlabelled data (extra 30K) for the DDL+VSAR model and the same 20K with 100K unlabelled data (extra 80K) for the DDL+VSAR model (similar to DDL+VSAR$_2$ setting in table 7.1 and 7.2).

We can clearly observe that the generated paraphrase matches better with the reference when more unlabelled data is utilised during the training of the DDL+VSAR model. The latent sequence generated when incorporating unlabelled data in the VSAR model improve the paraphrase generation performance. In general, we observe a progressive improvement in capturing the essence of information when incorporating more unlabelled examples. Compared to the preliminary experiment where the VSAR model failed to learn in a fully unsupervised scenario (section 7.6.5); through qualitative results in table 7.8 and quantitative results in table 7.1 and 7.2; we show that we are able to conduct semi-supervised learning with our unsupervised model VSAR and our supervised model DDL.

Source:	is it possible to go to the core of the earth ?
Reference:	if i really wanted to , can i dig all the way to the core of the earth ?
DDL(20K)	is it possible to go to the earth ?
DDL(20K) + VSAR(20K)	how do i go about the earth ?
DDL(20K) + VSAR(50K)	how do i go about the earth ?
DDL(20K) + VSAR(100K)	is it possible to go to the core of the earth ?
Source:	what are best courses for journalism ?
Reference:	what are the best courses on journalism ?
DDL(20K)	what is the best digital marketing course ?
DDL(20K) + VSAR(20K)	what is the best digital marketing agency ?
DDL(20K) + VSAR(50K)	what is the best digital marketing course ?
DDL(20K) + VSAR(100K)	what are the best courses for journalism ?
Source:	how should i stop thinking about someone ?
Reference:	how do i stop thinking about someone ?
DDL(20K)	how do i stop thinking about me ?
DDL(20K) + VSAR(20K)	how do i stop thinking about thinking ?
DDL(20K) + VSAR(50K)	how do i stop thinking about something ?
DDL(20K) + VSAR(100K)	how do i stop thinking about someone ?
Source:	what motivates all people ?
Reference:	what motivates people ?
DDL(20K)	why do people often keep all people ?
DDL(20K) + VSAR(20K)	why do people get tattoos ?
DDL(20K) + VSAR(50K)	what inspires to be so hard ?
DDL(20K) + VSAR(100K)	what motivates people ?
Source:	how can i get 1 million users to sign up to my app ?
Reference:	how can i get a million users on my social app ?
DDL(20K)	how can i get a game of app on my app ?
DDL(20K) + VSAR(20K)	how do i get a person from a app ?
DDL(20K) + VSAR(50K)	how can i get a billionaire by youtube ?
DDL(20K) + VSAR(100K)	how can i get 1 million users back from my app ?
Source:	is vegetarian good for health or non - vegetarian ?
Reference:	which is good food for our health : vegetarian or non - vegetarian ?
DDL(20K)	is smoking considered a vegetarian vegetarian ?
DDL(20K) + VSAR(20K)	is vegetarian considered good for health ?
DDL(20K) + VSAR(50K)	is vegetarian better than vegetarian ?
DDL(20K) + VSAR(100K)	is vegetarian health good or bad ?
Source:	how do i manage my microsoft account ?
Reference:	how can i do manage my microsoft account ?
DDL(20K)	how do i manage my microsoft office ?
DDL(20K) + VSAR(20K)	how do i manage my microsoft size ?
DDL(20K) + VSAR(50K)	how do i manage my google account ?
DDL(20K) + VSAR(100K)	how do i manage my microsoft account ?

Table 7.8: Chapter 7: Selected paraphrase generation results for semi-supervised model (DDL+VSAR) when incorporating different amounts of unlabelled data (denoted in brackets) and the same amount of labelled data (20K), represented in the case of Quora dataset.

7.7 Conclusions

In this paper, we have introduced a semi-supervised deep generative model for paraphrase generation. The unsupervised model is based on the variational auto-encoding framework and provides an effective method to handle missing labels. The supervised model conducts dual learning and injects supervised information into the

unsupervised model. With our novel knowledge reinforced training scheme, we empirically demonstrate that semi-supervised learning benefits our combined model, given unlabelled data and a fraction of the paired data. The evaluation results show that our combined model improves upon a very strong baseline model in a semi-supervised setting. We also observe that, even for the full dataset, our combined model achieves competitive performance with the state-of-the-art models for two paraphrase generation benchmark datasets. Additionally, we are able to model language as a discrete latent variable sequence for paraphrase generation tasks.

Epilogue

Compared with Chapter 4, 5 and 6, which focused on supervised learning setting and only labelled datasets are used; this chapter (Chapter 7) studies semi-supervised learning and utilised a mixture of labelled and unlabelled data. The semi-supervised setting more commonly occurs for real-world problems as large-scale labelled data is often expensive and hard to collect. Continue with Chapter 7, here we study another natural language generation problem and explore the application of DLVMs for a semi-supervised paraphrase generation problem. We present a novel unsupervised approach and model the missing label of a single unlabelled data point as a latent variable. Additionally, we present a novel supervised model to leverage information from paired data; combining these two models allows us to conduct semi-supervised learning.

We demonstrate that DLVM addresses the concerns over *interpretability* (represent model missing data for semi-supervised learning; use a discrete latent variable to represent words in a sentence sequence) and *explainability* (infer the corresponding paraphrase given a sentence), and the effectiveness of our approach with improved performance when incorporating unlabelled data and competitive to the state-of-the-art performance when trained with complete data.

CHAPTER 8

Conclusions and Future Work

8.1 Thesis Summary and Contributions

Due to their unparalleled performance and versatility, deep learning has become the de facto standard for building natural language processing (NLP) applications. Several concerns, however, have been raised in the research communities regarding their robustness, trustworthiness, explainability, and interpretability. Although these limitations of deep learning methods are widely acknowledged, work in methods and applications to alleviate these concerns in NLP is contrastingly limited. To address this research gap and explore a more robust approach for building NLP applications with deep learning, in this thesis, we studied deep latent variable models (DLVMs) from methods and applications perspectives for building natural language processing applications.

For contributions from a methods perspective, we studied the benefits of deep latent variable models in supervised and semi-supervised learning settings. These studies suggested that deep latent variable models are competitive in performance against standard deep learning methods; while offering additional robustness, trustworthiness, explainability and interoperability in various applications.

For contributions from an applications perspective, we first presented two applications for language understanding problems, followed by two more applications for language generation problems. Our first application concerns a binary text classification task in the educational domain and pioneers the first research on how Bayesian deep learning can be applied to this text-based educational application. Our second application focuses on multilabel text classification tasks, and we present an efficient uncertainty quantification framework as our contribution; this is the first research on using deep latent variable models for efficient uncertainty quantification purposes in multilabel text classification tasks. Our third application deals with multiple explanation generation for an explainable artificial intelligence (XAI) task, and we present a first study on how deep latent variable models can be used to generate multiple explanations in the Stanford natural language inference task. In our last application, we explore paraphrase generation tasks and present the first study of DLVMs in a semi-supervised learning setting in paraphrase generation tasks.

To sum up, in this thesis, we have explored and proposed novel methods and applications with deep latent variable models for both natural language understanding and natural language generation problems. We have contributed by designing and implementing deep latent variable models under both supervised and semi-supervised learning settings with four novel applications.

8.2 Answers to Research Questions

Concretely, this thesis presents the first research theme in exploring the benefits of Bayesian deep learning for an educational AI application, with two Bayesian methods in Chapter 4 to answer **RQ**1:

- We have explored learner-based text posts prediction task with two Bayesian methods: Bayesian Neural Network (BNN) and Deep Latent Variable Model (DLVM), as a new solution to assessing the need for instructor interventions;
- To the best of our knowledge, we presented the first study on Bayesian deep learning methods for AI in education contexts;

- We compared models based on our proposed methods with probabilistic modelling to its baseline deep learning models under similar circumstances for different cases of applying prediction;
- We presented results to suggest that Bayesian deep learning offers a critical uncertainty measurement apart from standard prediction measurements that are not supplied by traditional neural networks. This adds more explainability, trust and robustness to AI, which is crucial in education-based applications;
- We showed that our methods can achieve similar or better performance compared to non-probabilistic neural networks and are more robust with lower variance using random initialisation.

To conclude the first research question, we empirically showed the benefits of Bayesian deep learning methods to improve the *robustness* and *trustworthiness* of a text classification task in education domain. In comparison with the two methods, DLVM allows more interpretability of the model due to the existence of an explicit latent variable and hence is identified as the main focus of our research in this thesis. Furthermore, in the second research theme, we have developed an efficient uncertainty quantification framework for general text classification tasks, which extends our **RQ**1 from natural language understanding in education to a more general NLP domain. We present a novel framework built based on DLVMs to offer efficient uncertainty quantification conditional on text in Chapter 5 to answer **RQ**2:

- We proposed methods of representing epistemic and aleatoric uncertainties to enable efficient uncertainty quantification conditional on text;
- We proposed efficient uncertainty quantification methods with posterior analysis in the (approximated) latent and data space;
- We conducted extensive experiments and studies on diverse neural network architectures (LSTM, CNN and Transformer) with proposed methods;
- We proved the benefits of explicitly modelling uncertainty in neural networks.

For **RQ**1 and **RQ**2, we showed that DLVMs improve *robustness* and *trustworthiness* and, at the same time, retain performance in deep learning for natural language understanding problems in NLP. Next, we studied the benefits of DLVMs in natural language generation problems. In our third research theme, we presented a framework to study how DLVMs can contribute to generating multiple explanations for XAI in Chapter 6 to answer **RQ**3:

- We proposed a deep generative XAI framework based on the Transformer architecture;
- We showed that the proposed framework presents explanations in two steps: STEP ONE, explanation and label prediction; and STEP TWO, diverse evidence generation;
- We empirically showed our framework achieves competitive or better performance on explanation generation and prediction in STEP ONE, and is able to generate diverse explanations in STEP TWO.

For **RQ**3, we use a latent variable as an explicit representation of the semantic information. By interpolating through the latent space, our framework can generate diverse explanations with similar semantic content. Our third research theme suggested that DLVM improves the *explainability* and *interpretability* of deep learning for this natural language generation problem. For **RQ**1, **RQ**2 and **RQ**3, our research themes focus on supervised machine learning setup; however, for real-world NLP problems, the availability of large-scale annotated data is often not possible. In Chapter 7, we explored our last research theme and used DLVMs for paraphrase generation tasks in a semi-supervised setting to answer **RQ**4:

- We proposed a novel unsupervised model named variational sequence autoencoding reconstruction (VSAR);
- We proposed a novel supervised model named dual discriminative learning (DDL);
- We showed that combining VSAR and DDL model for semi-supervised learning in paraphrase generation tasks;

- We empirically suggest the effectiveness of the combined model for semi-supervised learning.

In Chapter 4, 5, 6 and 7, we explored and proposed novel methods and applications with deep latent variable models for both natural language understanding and natural language generation problems. Through these applications, we demonstrate that deep latent variable models help to address concerns for *robustness*, *trustworthiness*, *interpretability*, and *explainability* in natural language processing applications with deep latent variable models. The discussion about our methodology and justifications has been elaborated in section 1.5. From a methodology perspective, the exploration study conducted to answer **RQ**1 in the educational AI field has been adjusted and extended to answer **RQ**2 in general natural language understanding problems in the NLP field. Our studies conducted to answer **RQ**1 and **RQ**2 naturally lead to **RQ**3 and **RQ**4 in natural language generation problems in the NLP field. Our limitations for supervised learning studies for **RQ**1, **RQ**2 and **RQ**3 lead to **RQ**4 which explored semi-supervised learning with DLVMs.

8.3 Limitations of This Research

In this thesis, there are some research limitations due to time, cost, research equipment, and other unforeseen reasons. And we will present these points from two perspectives: machine learning methods and natural language processing applications.

From a methods perspective, we have adopted DLVMs as a general learning and inference framework to build NLP applications. One limitation of the study in this thesis concerns the usage of data. Labelled datasets were employed in Chapter 4, 5 and 6. The performance of models in supervised learning setup heavily relied on the labelled datasets. However, the data annotation process can be very labour-intensive and time-consuming. More importantly, in real-life scenarios, large-scale data with good-quality annotation might not be possible. One possibility to tackle this limitation is to employ semi-supervised learning techniques, which additionally exploit knowledge from the massive unlabelled dataset. While some preliminary

attempts at semi-supervised learning have been made in Chapter 7, more efforts and experiments are needed to demonstrate the full potential of DLVMs in a semi-supervised setting.

From an applications perspective, we have presented NLP applications built with DLVMs in both natural language understanding (Chapter 4 and 5) and generation (Chapter 6 and 7) domain. One limitation of the study in this thesis concerns the level of modelling for these NLP applications, as most of them are built to model the sentence level. The complex nature of many NLP tasks goes beyond sentence levels, such as paraphrase-level and even document-level modelling. One possibility to address this limitation is to use multi-level, hierarchical DLVMs to capture semantic and syntactic variants of a language in different sentences. Further studies in this direction are required to demonstrate the benefits of DLVMs for NLP applications. Another limitation worth mentioning is that study in this thesis does not base on pre-trained language models (PLMs), which represent the state-of-the-art methods in NLP. This is based on the concerns of computational resources and carbon footprint[1]. From a theoretical perspective, our result will generalise to large-scale PLMs; however, the lack of research in building deep latent variable models with PLMs motivates future experiments in this direction.

The utilization of PLMs has a substantial effect on the environment, due to the intensive computational demands and energy consumption incurred during both training and operation. This results in elevated levels of carbon emissions and exacerbates sustainability problems. As an ethical concern, the deployment of PLMs necessitates organizations to contemplate the environmental impact and adopt sustainable practices, such as minimizing carbon footprint and fostering green computing. This raises questions about the accountability of technology corporations in guaranteeing environmentally responsible practices in the creation and application of these models.

In the field of NLP, the human aspect holds a crucial role, particularly from a computational linguistic perspective. Although natural language can be flexible and

[1]https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/

versatile (e.g. similar semantic content can be expressed in various valid forms), it can also be prone to bias when evaluating annotations or generated text, as there is no standard by which to assess the quality of the language.

Two limitations have been identified in this research. The first limitation involves the quantitative evaluation metrics, such as BLEU, perplexity, and rouge, used in this thesis. These metrics are based on the overlap of n-gram contexts between generated outputs and reference text and while they are commonly used to compare with published results, they cannot directly assess the quality of the generated text. More recent evaluation metrics, such as BERT, have been proposed but there is still no universal agreement on the best method.

The second limitation is the absence of user-based studies to evaluate NLP systems, which is often considered the most accurate way to assess the effectiveness of NLP in real life. Current evaluations in this thesis rely on public benchmark datasets, which may be biased towards a subset of end-users, due to factors such as time and cost constraints, availability of experts, and the high cost of labor. User-based evaluations provide a more accurate understanding of whether NLP systems meet the needs of end-users, but due to resource constraints, user-based evaluations were not performed in this research and represent a promising area for future study.

8.4 Lessons and Future Directions

In this thesis, we have explored deep latent variable models for natural language processing applications from methods and applications perspectives. Throughout the thesis, deep latent variable models have proven their excellent for building NLP applications by maintaining the superior performance of deep learning while allowing a principled way to incorporate uncertainty into deep learning via (approximated) Bayesian inference. Additionally, we have shown in previous Chapters (4, 5, 6 and 7), deep latent variable models alleviate concerns in robustness, trustworthiness, interpretability and explainability of deep learning, which is the main motivation behind the works in this thesis.

However, no methods are perfect; with these great advantages, the deep latent

variable models suffer some drawbacks: first, due to the use of probabilistic modelling, there is a trade-off between the optimal performance of the models with respect to how much uncertainty they manage to capture, thus commonly results in slightly decrease of model performance in standard metric measurements against non-probabilistic deep learning models, as shown in Chapter 4, 5, 6 and 7; second, compared with non-probabilistic deep learning models, deep latent variable models are not very intuitive due to existence of learning and inference process during model training, this prevents them from widely adopted by a wider research community, as explained in Chapter 3; third and final, training deep latent variables can be difficult and may require additional techniques on dealing with cost functions, as shown in Chapter 5 and 7. Although compared with their advantages as well as their mathematical and statistical soundness, in general, deep latent variable models are definitely recommended as an important tool for researchers working in the deep learning field. With various NLP applications presented in Chapter 4, 5, 6 and 7 of this thesis, deep latent variable models can be a valuable asset and add up for researchers working in the NLP field.

Regarding future research avenues, many of the applications presented in this thesis are worth further exploration. The approaches and applications proposed in this thesis can be further extended to other related tasks and create broader impacts. For example, the exploration study for Bayesian deep learning methods in Chapter 4 can be extended to other applications in the educational data mining and analytical field on tasks such as dropout prediction, learner profile prediction and learner behaviour modelling, etc. The efficient uncertainty quantification framework in Chapter 5 can be tested more broadly on text classification-based natural languages understanding tasks, such as aspect-sentiment analysis, question and answer selection, information retrieval and other more broad domains. Our proposed deep generative XAI framework in Chapter 6 can be extended to other natural language generation tasks which require multiple semantic equivalent instances to be generated. The semi-supervised learning framework presented in Chapter 7 can also be extended to other semi-supervised generation tasks and also tested with PLMs.

The findings in this thesis are of practical value to deep learning practitioners,

researchers and engineers working on a variety of problems in the field of natural language processing and deep learning and are concerned about the robustness, trustworthiness, explainability and interpretability of these applications.

Concluding, we can say that, via this thesis, we have made some significant contributions in the cross-disciplinary areas of deep learning and natural language processing, opening at the same time new directions for future researchers to explore further.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016. (document), 1.1, 2.1, 3.2, 3.1, 3.2, 3.4, 3.7

[2] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017. https://distill.pub/2017/feature-visualization. (document), 3.2, 3.2

[3] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. (document), 2.2, 3.6.1, 3.4, 5.3.4, 7.2.2, 7.2.4, 7.5.2

[4] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2014. (document), 1.1, 2.2, 3.4, 3.4.1, 3.6.1, 3.7, 3.7, 4.2.3, 4.3.3, 5.1, 5.1, 5.3.3, 5.2, 5.3.3, 5.3.3, 6.2.2, 6.3.1, 7.1, 7.2.4, 7.2.4

[5] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, vol. 32 of *JMLR Workshop and Conference Proceedings*, pp. 1278–1286, JMLR.org, 2014. (document), 1.1, 2.2, 3.4, 3.4.1, 3.7, 3.7, 6.2.2, 6.3.1, 7.1, 7.2.4

[6] A. Mnih and K. Gregor, "Neural variational inference and learning in belief networks," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, vol. 32 of *JMLR Workshop and Conference Proceedings*, pp. 1791–1799, JMLR.org, 2014. (document), 1.1, 2.2, 3.4, 3.4.1, 3.5.1, 3.7, 3.7, 6.2.2, 7.1, 7.2.4, 7.2.4, 7.5.2

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. 1.1, 2.1

[8] Y. Goldberg, "Neural network methods for natural language processing," *Synthesis lectures on human language technologies*, vol. 10, no. 1, pp. 1–309, 2017. 1.1

[9] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 2008–2026, 2018. 1.1, 2.1, 2.2, 5, 3.4, 6, 4.2.3

[10] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 2020. 1.1, 2.1, 3.2

[11] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 6294–6305, 2017. 1.1

[12] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 649–657, 2015. 1.1, 2.1, 5.4.1

[13] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 3104–3112, 2014. 1.1, 2.1, 3.2, 6.4.2, 6.6.1, 7.1

[14] A. M. Rush, S. Harvard, S. Chopra, and J. Weston, "A neural attention model for sentence summarization," in *ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing*, 2017. 1.1

[15] L. Galárraga, G. Heitz, K. Murphy, and F. M. Suchanek, "Canonicalizing open knowledge bases," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014* (J. Li, X. S. Wang, M. N. Garofalakis, I. Soboroff, T. Suel, and M. Wang, eds.), pp. 1679–1688, ACM, 2014. 1.1

[16] A. Gupta, A. Agarwal, P. Singh, and P. Rai, "A deep generative framework for paraphrase generation," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA,*

February 2-7, 2018 (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 5149–5156, AAAI Press, 2018. 1.1, 7.5.1

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986. 1.1, 3.4

[18] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 5574–5584, 2017. 1.1, 4.2.2, 4.3.2, 4.4.2, 5, 5.1, 5.1, 5.2, 5.3.2, 5.3.3, 5.3.4, 5.4.5, 5.5.2, 5.5.3, 5.5.4

[19] R. McAllister, Y. Gal, A. Kendall, M. van der Wilk, A. Shah, R. Cipolla, and A. Weller, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017* (C. Sierra, ed.), pp. 4745–4753, ijcai.org, 2017. 1.1, 5.1, 6.1

[20] O. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom, "e-snli: Natural language inference with natural language explanations," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 9560–9572, 2018. 1.1, 6.1, 6.2.1, 6.4, 6.4.2, 6.6.4, 6.7.4, 6.7.5

[21] S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. Bowman, and N. A. Smith, "Annotation artifacts in natural language inference data," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, (New Orleans, Louisiana), pp. 107–112, Association for Computational Linguistics, 2018. 1.1, 6.1, 1

[22] D. Castelvecchi, "Can we open the black box of ai?," *Nature News*, vol. 538, no. 7623, p. 20, 2016. 1.1

[23] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020. 1.1

[24] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021. 1.1, 5.1

[25] T. Räukur, A. Ho, S. Casper, and D. Hadfield-Menell, "Toward transparent ai: A survey on interpreting the inner structures of deep neural networks," *arXiv preprint arXiv:2207.13243*, 2022. 1.1

[26] D. Kahneman, S. P. Slovic, P. Slovic, and A. Tversky, *Judgment under uncertainty: Heuristics and biases*. Cambridge university press, 1982. 1.1

[27] K. Daniel, *Thinking, fast and slow*. 2017. 1.1

[28] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006. 1.1, 2.2, 3.4.2, 3.7, 4.2.3, 5.2, 6.2.2

[29] S. Cohen, "Bayesian analysis in natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 9, no. 2, pp. 1–274, 2016. 1.1, 2.2, 3.7

[30] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 604–624, 2020. 2.1

[31] A. M. Turing, "Computing machinery and intelligence," in *Parsing the turing test*, pp. 23–65, Springer, 2009. 2.1

[32] K. S. Jones, "Natural language processing: a historical review," *Current issues in computational linguistics: in honour of Don Walker*, pp. 3–16, 1994. 2.1

[33] S. C. Salveter, "Natural language processing," *American Journal of Computational Linguistics*, vol. 7, no. 4, 1981. 2.1

[34] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009* (A. P. Danyluk, L. Bottou, and M. L. Littman, eds.), vol. 382 of *ACM International Conference Proceeding Series*, pp. 873–880, ACM, 2009. 2.1

[35] A. Coates, B. Huval, T. Wang, D. J. Wu, B. Catanzaro, and A. Y. Ng, "Deep learning with COTS HPC systems," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, vol. 28 of *JMLR Workshop and Conference Proceedings*, pp. 1337–1345, JMLR.org, 2013. 2.1

[36] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, pp. 1–12, 2017. 2.1

[37] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heinz, and D. Roth, "Recent advances in natural language processing via large pre-trained language models: A survey," *arXiv preprint arXiv:2111.01243*, 2021. 2.1

[38] W. S. Noble, "What is a support vector machine?," *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006. 2.1

[39] S. Menard, *Applied logistic regression analysis*. No. 106, Sage, 2002. 2.1

[40] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018. 2.1

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013. 2.1

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States* (C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 3111–3119, 2013. 2.1, 7.6.3

[43] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011. 2.1

[44] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), pp. 932–938, MIT Press, 2000. 2.1

[45] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, 2014. 2.1, 4.4.1, 7.6.3

[46] R. Johnson and T. Zhang, "Semi-supervised convolutional neural networks for text categorization via region embedding," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 919–927, 2015. 2.1

[47] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, (Edinburgh, Scotland, UK.), pp. 151–161, Association for Computational Linguistics, 2011. 2.1

[48] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for Twitter sentiment classification," in *Proceedings*

of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), (Baltimore, Maryland), pp. 1555–1565, Association for Computational Linguistics, 2014. 2.1

[49] X. Chen, L. Xu, Z. Liu, M. Sun, and H. Luan, "Joint learning of character and word embeddings," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015* (Q. Yang and M. J. Wooldridge, eds.), pp. 1236–1242, AAAI Press, 2015. 2.1

[50] H. Peng, E. Cambria, and X. Zou, "Radical-based hierarchical embeddings for chinese sentiment analysis at sentence level," in *The Thirtieth International Flairs Conference*, 2017. 2.1

[51] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. 2.1

[52] S. Wang, W. Zhou, and C. Jiang, "A survey of word embeddings based on deep learning," *Computing*, vol. 102, no. 3, pp. 717–740, 2020. 2.1

[53] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, 2018. 2.1

[54] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, 2019. 2.1, 5.1, 5.4.3, 6.3.2, 6.4.1, 7.6.1, 7.6.3

[55] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019. 2.1

[56] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020. 2.1, 6.3.2

[57] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Baltimore, Maryland), pp. 655–665, Association for Computational Linguistics, 2014. 2.1, 3.2

[58] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, Association for Computational Linguistics, 2014. 2.1, 3.2, 5.4.3, 6.5.2

[59] S. Poria, E. Cambria, and A. Gelbukh, "Aspect extraction for opinion mining with a deep convolutional neural network," *Knowledge-Based Systems*, vol. 108, pp. 42–49, 2016. 2.1

[60] A. Kirillov, D. Schlesinger, W. Forkel, A. Zelenin, S. Zheng, P. Torr, and C. Rother, "Efficient likelihood learning of a generic cnn-crf model for semantic segmentation," *arXiv preprint arXiv:1511.05067*, 2015. 2.1

[61] S. Ruder, P. Ghaffari, and J. G. Breslin, "INSIGHT-1 at SemEval-2016 task 5: Deep learning for multilingual aspect-based sentiment analysis," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, (San Diego, California), pp. 330–336, Association for Computational Linguistics, 2016. 2.1

[62] M. Denil, A. Demiraj, N. Kalchbrenner, P. Blunsom, and N. de Freitas, "Modelling, visualising and summarising documents with a single convolutional neural network," *arXiv preprint arXiv:1406.3830*, 2014. 2.1

[63] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao, "Semantic clustering and convolutional neural network for short text categorization," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, (Beijing, China), pp. 352–357, Association for Computational Linguistics, 2015. 2.1

[64] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014* (J. Li, X. S. Wang, M. N. Garofalakis, I. Soboroff, T. Suel, and M. Wang, eds.), pp. 101–110, ACM, 2014. 2.1

[65] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021. 2.1

[66] D. Jurafsky, *Speech & language processing*. Pearson Education India, 2000. 2.1

[67] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 1422–1432, Association for Computational Linguistics, 2015. 2.1

[68] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014. 2.1

[69] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 933–941, PMLR, 2017. 2.1

[70] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *arXiv preprint arXiv:1702.01923*, 2017. 2.1

[71] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (San Diego, California), pp. 260–270, Association for Computational Linguistics, 2016. 2.1

[72] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, Ieee, 2013. 2.1, 3.2

[73] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, 2015. 2.1

[74] X. Wang, Y. Liu, C. Sun, B. Wang, and X. Wang, "Predicting polarities of tweets by composing word embeddings with long short-term memory," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Beijing, China), pp. 1343–1353, Association for Computational Linguistics, 2015. 2.1

[75] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems," in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, (Prague, Czech Republic), pp. 285–294, Association for Computational Linguistics, 2015. 2.1

[76] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the*

2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), (Doha, Qatar), pp. 1724–1734, Association for Computational Linguistics, 2014. 2.1

[77] O. Vinyals and Q. Le, "A neural conversational model," *arXiv preprint arXiv:1506.05869*, 2015. 2.1

[78] J. Li, M. Galley, C. Brockett, G. Spithourakis, J. Gao, and B. Dolan, "A persona-based neural conversation model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Berlin, Germany), pp. 994–1003, Association for Computational Linguistics, 2016. 2.1

[79] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 1–9, IEEE Computer Society, 2015. 2.1

[80] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015. 2.1, 3.2

[81] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, 2017. 2.1, 3.2, 5.4.3, 5.4.3, 6.3.1, 6.3.2, 6.4, 6.6.1, 7.3, 7.6.2, 7.6.3, 7.6.5, 7.6.5

[82] Y. LeCun and I. Misra, "Self-supervised learning: The dark matter of intelligence," *Meta AI*, vol. 23, 2021. 2.1

[83] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, 2021. 2.1

[84] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H. Hon, "Unified language model pre-training for natural language understanding and generation," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 13042–13054, 2019. 2.1

[85] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer.," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020. 2.1

[86] A. Skrondal and S. Rabe-Hesketh, "Latent variable modelling: A survey," *Scandinavian Journal of Statistics*, vol. 34, no. 4, pp. 712–745, 2007. 2.2

[87] G. Verbeke and G. Molenberghs, "Modeling through latent variables," *Annual Review of Statistics and Its Application*, vol. 4, no. 1, pp. 267–282, 2017. 2.2

[88] Z. Ge, "Process data analytics via probabilistic latent variable models: A tutorial review," *Industrial & Engineering Chemistry Research*, vol. 57, no. 38, pp. 12646–12661, 2018. 2.2

[89] P. Diaconis and D. Ylvisaker, "Conjugate priors for exponential families," *The Annals of statistics*, pp. 269–281, 1979. 2.2

[90] S. Dalal and W. Hall, "Approximating priors by mixtures of natural conjugate priors," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 45, no. 2, pp. 278–286, 1983. 2.2

[91] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 6231–6239, 2017. 2.2, 3.2

[92] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017. 2.2, 3.4, 3.4.1, 4.2.3

[93] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999. 2.2, 3.4, 4.2.3, 5.2, 6.2.2

[94] Y. Kim, S. Wiseman, and A. M. Rush, "A tutorial on deep latent variable models of natural language," *arXiv preprint arXiv:1812.06834*, 2018. 2.2, 4.3.3, 7.5.2

[95] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational Linguistics*, vol. 19, no. 2, pp. 263–311, 1993. 2.2

[96] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," in *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]* (T. G. Dietterich, S. Becker, and Z. Ghahramani, eds.), pp. 601–608, MIT Press, 2001. 2.2

[97] P. F. Brown, V. J. Della Pietra, P. V. Desouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–480, 1992. 2.2

[98] D. Klein and C. Manning, "Corpus-based induction of syntactic structure: Models of dependency and constituency," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, (Barcelona, Spain), pp. 478–485, 2004. 2.2

[99] Y. Miao, *Deep generative models for natural language processing*. PhD thesis, University of Oxford, 2017. 2.2, 3.5.2, 5.3.4

[100] A. Rush, Y. Kim, and S. Wiseman, "Deep latent variable models of natural language," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, (Melbourne, Australia), Association for Computational Linguistics, 2018. 2.2, 3.5.2

[101] R. Li, *Deep Latent Variable Models for Text Modelling*. PhD thesis, University of Sheffield, 2021. 2.2

[102] H. B. Shah, *Deep Generative Models for Natural Language*. PhD thesis, UCL (University College London), 2021. 2.2

[103] X. Shen, "Deep latent-variable models for text generation," *arXiv preprint arXiv:2203.02055*, 2022. 2.2

[104] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, (Berlin, Germany), pp. 10–21, Association for Computational Linguistics, 2016. 2.2, 5.5.1, 2, 5.5.1, 7.5.2

[105] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 3483–3491, 2015. 2.2, 5.1, 5.3.3, 6.3.1, 6.6.1

[106] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (M. Balcan and K. Q. Weinberger, eds.), vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 1558–1566, JMLR.org, 2016. 2.2, 5.1, 5.3.3, 6.3.1, 6.6.1

[107] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1587–1596, PMLR, 2017. 2.2

[108] T. Zhao, R. Zhao, and M. Eskenazi, "Learning discourse-level diversity for neural dialog models using conditional variational autoencoders," in *Proceedings*

of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), (Vancouver, Canada), pp. 654–664, Association for Computational Linguistics, 2017. 2.2, 6.2.2, 6.3.1

[109] A. Pagnoni, K. Liu, and S. Li, "Conditional variational autoencoder for neural machine translation," *arXiv preprint arXiv:1812.04405*, 2018. 2.2, 5.3.3, 5.5.1, 2, 3, 5.5.1, 6.2.2, 6.3.1

[110] A. B. Dieng, Y. Kim, A. M. Rush, and D. M. Blei, "Avoiding latent variable collapse with generative skip models," in *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan* (K. Chaudhuri and M. Sugiyama, eds.), vol. 89 of *Proceedings of Machine Learning Research*, pp. 2397–2405, PMLR, 2019. 2.2, 7.5.2

[111] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick, "Lagging inference networks and posterior collapse in variational autoencoders," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019. 2.2, 5.5.1, 2, 5.5.1, 7.5.2

[112] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 2.2, 7.4.2, 7.4.2, 7.5.2, 7.6.5, 7.6.5

[113] A. Razavi, A. van den Oord, B. Poole, and O. Vinyals, "Preventing posterior collapse with delta-vaes," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019. 2.2, 7.5.2

[114] B. Li, J. He, G. Neubig, T. Berg-Kirkpatrick, and Y. Yang, "A surprisingly effective fix for deep latent variable modeling of text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 3603–3614, Association for Computational Linguistics, 2019. 2.2

[115] Q. Zhu, W. Bi, X. Liu, X. Ma, X. Li, and D. Wu, "A batch normalized inference network keeps the KL vanishing away," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 2636–2649, Association for Computational Linguistics, 2020. 2.2

[116] Y. Wang, D. Blei, and J. P. Cunningham, "Posterior collapse and latent variable non-identifiability," *Advances in Neural Information Processing Systems*, vol. 34, 2021. 2.2, 7.5.2

[117] J.-T. Chien and C.-W. Wang, "Hierarchical and self-attended sequence autoencoder," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2.2

[118] Y. Miao and P. Blunsom, "Language as a latent variable: Discrete generative models for sentence compression," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 319–328, Association for Computational Linguistics, 2016. 2.2, 3, 7.2, 7.4.2, 7.4.2, 7.5.2

[119] T. Wen, Y. Miao, P. Blunsom, and S. J. Young, "Latent intention dialogue models," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 3732–3741, PMLR, 2017. 2.2, 7.5.2

[120] D. Yarats and M. Lewis, "Hierarchical text generation and planning for strategic dialogue," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (J. G. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 5587–5595, PMLR, 2018. 2.2

[121] T. Zhao, K. Xie, and M. Eskenazi, "Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 1208–1218, Association for Computational Linguistics, 2019. 2.2

[122] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2204–2212, 2014. 2.2, 7.2.4, 7.5.2

[123] J. He, X. Wang, G. Neubig, and T. Berg-Kirkpatrick, "A probabilistic formulation of unsupervised text style transfer," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020. 2.2, 7.2, 7.4.2, 7.4.2, 7.5.2

[124] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 2.2, 3.6.1, 5.3.4, 7.2.2, 7.2.4, 7.5.2

[125] J. Choi, K. M. Yoo, and S. Lee, "Learning to compose task-specific tree structures," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February*

2-7, 2018 (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 5094–5101, AAAI Press, 2018. 2.2, 7.5.2

[126] Y. Fu, Y. Feng, and J. P. Cunningham, "Paraphrase generation with latent bag of words," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 13623–13634, 2019. 2.2, 7.1, 7.2.2, 7.5.1, 7.5.2, 7.6.1, 7.6.5, 7.6.5

[127] G. Mercatali and A. Freitas, "Disentangling generative factors in natural language with discrete variational autoencoders," *arXiv preprint arXiv:2109.07169*, 2021. 2.2, 7.5.2

[128] H. Shah, T. Xiao, and D. Barber, "Locally-contextual nonlinear crfs for sequence labeling," *arXiv preprint arXiv:2103.16210*, 2021. 2.2

[129] R. Li, C. Lin, M. Collinson, X. Li, and G. Chen, "A dual-attention hierarchical recurrent neural network for dialogue act classification," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, (Hong Kong, China), pp. 383–392, Association for Computational Linguistics, 2019. 2.2

[130] Y. Kim, S. Wiseman, A. C. Miller, D. A. Sontag, and A. M. Rush, "Semi-amortized variational autoencoders," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (J. G. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 2683–2692, PMLR, 2018. 2.2

[131] R. Li, X. Li, C. Lin, M. Collinson, and R. Mao, "A stable variational autoencoder for text modelling," in *Proceedings of the 12th International Conference on Natural Language Generation*, (Tokyo, Japan), pp. 594–599, Association for Computational Linguistics, 2019. 2.2

[132] Y. Kim, A. Rush, L. Yu, A. Kuncoro, C. Dyer, and G. Melis, "Unsupervised recurrent neural network grammars," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 1105–1117, Association for Computational Linguistics, 2019. 2.2

[133] R. Li, X. Li, G. Chen, and C. Lin, "Improving variational autoencoder for text modelling with timestep-wise regularisation," in *Proceedings of the 28th International Conference on Computational Linguistics*, (Barcelona, Spain (Online)), pp. 2381–2397, International Committee on Computational Linguistics, 2020. 2.2

[134] Y. Miao, L. Yu, and P. Blunsom, "Neural variational inference for text processing," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (M. Balcan

and K. Q. Weinberger, eds.), vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 1727–1736, JMLR.org, 2016. 2.2, 4.3.3, 4.3.3, 4.3.3, 5.3.3, 5.3.3, 5.3.3, 6.2.2, 6.3.1, 7.5.2

[135] Y. Miao, E. Grefenstette, and P. Blunsom, "Discovering discrete latent topics with neural variational inference," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 2410–2419, PMLR, 2017. 2.2

[136] Y. Kim, C. Dyer, and A. Rush, "Compound probabilistic context-free grammars for grammar induction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 2369–2385, Association for Computational Linguistics, 2019. 2.2

[137] H. Shah and D. Barber, "Generative neural machine translation," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 1353–1362, 2018. 2.2

[138] H. Shah, B. Zheng, and D. Barber, "Generating sentences using a dynamic canvas," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 5430–5437, AAAI Press, 2018. 2.2

[139] X. Shen, H. Su, S. Niu, and V. Demberg, "Improving variational encoder-decoders in dialogue generation," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 5456–5463, AAAI Press, 2018. 2.2

[140] X. Shen, H. Su, W. Li, and D. Klakow, "NEXUS network: Connecting the preceding and the following in dialogue generation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 4316–4327, Association for Computational Linguistics, 2018. 2.2

[141] X. Shen, J. Suzuki, K. Inui, H. Su, D. Klakow, and S. Sekine, "Select and attend: Towards controllable content selection in text generation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 579–590, Association for Computational Linguistics, 2019. 2.2

[142] X. Shen, Y. Zhao, H. Su, and D. Klakow, "Improving latent alignment in text summarization by generalizing the pointer generator," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 3762–3773, Association for Computational Linguistics, 2019. 2.2

[143] X. Shen, E. Chang, H. Su, C. Niu, and D. Klakow, "Neural data-to-text generation via jointly learning the segmentation and correspondence," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 7155–7165, Association for Computational Linguistics, 2020. 2.2

[144] Y. Deng, Y. Kim, J. T. Chiu, D. Guo, and A. M. Rush, "Latent alignment and variational attention," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 9735–9747, 2018. 2.2

[145] S. M. Xie and S. Ermon, "Reparameterizable subset sampling via continuous relaxations," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019* (S. Kraus, ed.), pp. 3919–3925, ijcai.org, 2019. 2.2, 7.2.2, 7.2.4

[146] T. M. Mitchell *et al.*, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997. 3.1

[147] K. P. Murphy, *Probabilistic machine learning: an introduction*. MIT press, 2022. 3.1

[148] N. A. Gershenfeld and N. Gershenfeld, *The nature of mathematical modeling*. Cambridge university press, 1999. 3.1

[149] A. L. Samuel, "Machine learning," *The Technology Review*, vol. 62, no. 1, pp. 42–45, 1959. 3.1

[150] E. T. Jaynes, *Probability theory: The logic of science*. Cambridge university press, 2003. 3.1

[151] V. Barnett, *Comparative statistical inference*, vol. 522. John Wiley & Sons, 1999. 3.1

[152] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013. 3.2, 3.2

[153] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997. 3.2

[154] G. Jawahar, B. Sagot, and D. Seddah, "What does BERT learn about the structure of language?," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 3651–3657, Association for Computational Linguistics, 2019. 3.2

[155] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? an analysis of BERT's attention," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, (Florence, Italy), pp. 276–286, Association for Computational Linguistics, 2019. 3.2

[156] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009. 3.2

[157] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957. 3.2

[158] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989. 3.2

[159] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989. 3.2

[160] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991. 3.2

[161] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, p. 574, 1959. 3.2

[162] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968. 3.2

[163] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989. 3.2

[164] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 3.2

[165] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States* (P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1106–1114, 2012. 3.2, 5.4.3

[166] E. Grefenstette, P. Blunsom, N. de Freitas, and K. M. Hermann, "A deep architecture for semantic parsing," in *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, (Baltimore, MD), pp. 22–27, Association for Computational Linguistics, 2014. 3.2

[167] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, vol. 1, no. 4, pp. 339–356, 1988. 3.2

[168] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990. 3.2

[169] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 3.2, 5.4.3

[170] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, (Doha, Qatar), pp. 103–111, Association for Computational Linguistics, 2014. 3.2

[171] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model.," in *Interspeech*, vol. 2, pp. 1045–1048, Makuhari, 2010. 3.2

[172] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, (Seattle, Washington, USA), pp. 1700–1709, Association for Computational Linguistics, 2013. 3.2

[173] A. Graves, S. Fernández, M. Liwicki, H. Bunke, and J. Schmidhuber, "Unconstrained on-line handwriting recognition with recurrent neural networks," in *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007* (J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, eds.), pp. 577–584, Curran Associates, Inc., 2007. 3.2

[174] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), pp. 545–552, Curran Associates, Inc., 2008. 3.2

[175] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005. 3.2

[176] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 1412–1421, Association for Computational Linguistics, 2015. 3.2, 4.4.1

[177] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, *et al.*, "A survey on vision transformer," *IEEE transactions on pattern analysis and machine intelligence*, 2022. 3.2

[178] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush, "OpenNMT: Open-source toolkit for neural machine translation," in *Proceedings of ACL 2017, System Demonstrations*, (Vancouver, Canada), pp. 67–72, Association for Computational Linguistics, 2017. 3.2

[179] O. Kallenberg, *Probabilistic symmetries and invariance principles*. Springer Science & Business Media, 2006. 3.3

[180] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, vol. 28 of *JMLR Workshop and Conference Proceedings*, pp. 1139–1147, JMLR.org, 2013. 3.4

[181] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE international conference on neural networks*, pp. 586–591, IEEE, 1993. 3.4

[182] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015. 3.4, 5.4.3, 6.4, 7.6.3

[183] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010* (A. T. Kalai and M. Mohri, eds.), pp. 257–269, Omnipress, 2010. 3.4

[184] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 3.4

[185] M. J. Wainwright, M. I. Jordan, *et al.*, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008. 3.4, 4.2.3, 5.2

[186] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, 2013. 3.4, 4.2.3

[187] R. Ranganath, S. Gerrish, and D. M. Blei, "Black box variational inference," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, vol. 33 of *JMLR Workshop and Conference Proceedings*, pp. 814–822, JMLR.org, 2014. 3.4, 4.2.3

[188] T. Tanaka, "A theory of mean field approximation," in *Advances in Neural Information Processing Systems*, pp. 351–360, 1999. 3.4.2, 4.2.3

[189] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992. 3.5.1

[190] G. Papandreou and A. L. Yuille, "Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models," in *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011* (D. N. Metaxas, L. Quan, A. Sanfeliu, and L. V. Gool, eds.), pp. 193–200, IEEE Computer Society, 2011. 3.6.1

[191] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013. 3.6.2, 7.2.4

[192] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds.), pp. 1019–1027, 2016. 3.6.3, 4.2.2, 5.2

[193] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (M. Balcan and K. Q. Weinberger, eds.), vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 1050–1059, JMLR.org, 2016. 3.6.3, 4.2.2, 4.3.2, 4.3.2, 5.1, 5.2, 5.4.3, 5

[194] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. 3.6.3, 4.3.2, 5.1, 5.2

[195] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, 2020. 3.7

[196] A. F. Markus, J. A. Kors, and P. R. Rijnbeek, "The role of explainability in creating trustworthy artificial intelligence for health care: a comprehensive survey of the terminology, design choices, and evaluation strategies," *Journal of Biomedical Informatics*, vol. 113, p. 103655, 2021. 3.7

[197] A. Alamri, M. Alshehri, A. Cristea, F. D. Pereira, E. Oliveira, L. Shi, and C. Stewart, "Predicting moocs dropout using only two easily obtainable features from the first weeks activities," in *International Conference on Intelligent Tutoring Systems*, pp. 163–173, Springer, 2019. 4.1

[198] A. Alamri, Z. Sun, A. I. Cristea, G. Senthilnathan, L. Shi, and C. Stewart, "Is mooc learning different for dropouts? a visually-driven, multi-granularity explanatory ml approach," in *International Conference on Intelligent Tutoring Systems*, pp. 353–363, Springer, 2020. 4.1

[199] O. Almatrafi, A. Johri, and H. Rangwala, "Needle in a haystack: Identifying learner posts that require urgent response in mooc discussion forums," *Computers & Education*, vol. 118, pp. 1–9, 2018. 4.1, 4.2.1

[200] M. K. Chandrasekaran, M.-Y. Kan, B. C. Tan, and K. Ragupathi, "Learning instructor intervention from mooc forums: Early results and issues," *arXiv preprint arXiv:1504.07206*, 2015. 4.1, 4.2.1

[201] L. Alrajhi, A. Alamri, F. D. Pereira, and A. I. Cristea, "Urgency analysis of learners comments: An automated intervention priority model for mooc," in *International Conference on Intelligent Tutoring Systems*, pp. 148–160, Springer, 2021. 4.1

[202] A. Hernández-Blanco, B. Herrera-Flores, D. Tomás, and B. Navarro-Colorado, "A systematic review of deep learning approaches to educational data mining," *Complexity*, vol. 2019, 2019. 4.1, 4.2.1

[203] L. Alrajhi, K. Alharbi, and A. I. Cristea, "A multidimensional deep learner model of urgent instructor intervention need in mooc forum posts," in *International Conference on Intelligent Tutoring Systems*, pp. 226–236, Springer, 2020. 4.1, 4.4.2, 4.5, 4.5

[204] S. X. Guo, X. Sun, S. X. Wang, Y. Gao, and J. Feng, "Attention-based character-word hybrid neural networks with semantic and structural information for identifying of urgent posts in mooc discussion forums," *IEEE Access*, vol. 7, pp. 120522–120532, 2019. 4.1, 4.2.1, 4.4.1

[205] X. Sun, S. Guo, Y. Gao, J. Zhang, X. Xiao, and J. Feng, "Identification of urgent posts in mooc discussion forums using an improved rcnn," in *2019 IEEE World Conference on Engineering Education (EDUNINE)*, pp. 1–5, IEEE, 2019. 4.1, 4.2.1, 4.3.1

[206] S. Chaturvedi, D. Goldwasser, and H. Daumé III, "Predicting instructor's intervention in MOOC forums," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Baltimore, Maryland), pp. 1501–1511, Association for Computational Linguistics, 2014. 4.2.1

[207] A. Bakharia, "Towards cross-domain mooc forum post classification," in *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pp. 253–256, 2016. 4.2.1

[208] B. Clavié and K. Gal, "Edubert: Pretrained deep language models for learning analytics," *arXiv preprint arXiv:1912.00690*, 2019. 4.2.1

[209] X. Wei, H. Lin, L. Yang, and Y. Yu, "A convolution-lstm-based deep neural network for cross-domain mooc forum post classification," *Information*, vol. 8, no. 3, p. 92, 2017. 4.2.1

[210] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992. 4.2.2, 5.1

[211] D. J. MacKay, "Probable networks and plausible predictionsa review of practical bayesian methods for supervised neural networks," *Network: computation in neural systems*, vol. 6, no. 3, pp. 469–505, 1995. 4.2.2, 5.1

[212] R. M. Neal, *Bayesian learning for neural networks*, vol. 118. Springer Science & Business Media, 2012. 4.2.2, 5.1

[213] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain* (J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, eds.), pp. 2348–2356, 2011. 4.2.2

[214] J. M. Hernández-Lobato and R. P. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 1861–1869, JMLR.org, 2015. 4.2.2

[215] J. M. Hernández-Lobato, M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani, "Predictive entropy search for bayesian optimization with unknown constraints," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 1699–1707, JMLR.org, 2015. 4.2.2

[216] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," in *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*, BMVA Press, 2017. 4.2.2, 5.2

[217] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 103–110, IEEE, 2017. 4.2.2, 5.2

[218] Y. Xiao and W. Y. Wang, "Quantifying uncertainties in natural language processing tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7322–7329, 2019. 4.2.2, 4.4.2, 5, 5.1, 5.2, 5.3.4, 5.3.4, 5.4.3, 5, 5.5.2, 5.5.3

[219] Z. Ghahramani and M. J. Beal, "Propagation algorithms for variational bayesian learning," in *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), pp. 507–513, MIT Press, 2000. 4.2.3

[220] A. Blum, N. Haghtalab, and A. D. Procaccia, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2575–2583, 2015. 4.3.3, 5.3.3

[221] A. Agrawal, J. Venkatraman, S. Leonard, and A. Paepcke, "Youedu: addressing confusion in mooc discussion forums by recommending instructional video clips," 2015. 4.4.1

[222] C. Li, *Towards better representations with deep/Bayesian learning*. PhD thesis, Duke University, 2018. 5.1

[223] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 3581–3589, 2014. 5.1

[224] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?," *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009. 5.1, 5.2

[225] V. Böhm, F. Lanusse, and U. Seljak, "Uncertainty quantification with generative models," *arXiv preprint arXiv:1910.10046*, 2019. 5.1, 5.2, 5.3.3

[226] M. Prakash, A. Krull, and F. Jug, "Fully unsupervised diversity denoising with convolutional variational autoencoders," *arXiv preprint arXiv:2006.06072*, 2020. 5.1, 5.2, 5.3.3

[227] Y. Gal, "Uncertainty in deep learning," *University of Cambridge*, vol. 1, no. 3, p. 4, 2016. 5.2

[228] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl, "Leveraging uncertainty information from deep neural networks for disease detection," *Scientific reports*, vol. 7, no. 1, pp. 1–14, 2017. 5.2

[229] A. G. Roy, S. Conjeti, N. Navab, C. Wachinger, A. D. N. Initiative, *et al.*, "Bayesian quicknat: model uncertainty in deep whole-brain segmentation for structure-wise quality control," *NeuroImage*, vol. 195, pp. 11–22, 2019. 5.2

[230] W. Chen, B. Zhang, and M. Lu, "Uncertainty quantification for multilabel text classification," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 6, p. e1384, 2020. 5.2

[231] J. Yu, L. Alrajhi, A. Harit, Z. Sun, A. I. Cristea, and L. Shi, "Exploring bayesian deep learning for urgent instructor intervention need in mooc forums," in *International Conference on Intelligent Tutoring Systems*, pp. 78–90, Springer, 2021. 5.2

[232] E. T. Nalisnick, A. Matsukawa, Y. W. Teh, D. Görür, and B. Lakshminarayanan, "Do deep generative models know what they don't know?," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019. 5.3.2

[233] J. Qian, M. ElSherief, E. Belding, and W. Y. Wang, "Hierarchical CVAE for fine-grained hate speech classification," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 3550–3559, Association for Computational Linguistics, 2018. 5.3.3, 5.3.3, 5.3.3

[234] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," *Advances in neural information processing systems*, vol. 31, 2018. 5.3.4

[235] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen, "A survey of the state of explainable AI for natural language processing," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, (Suzhou, China), pp. 447–459, Association for Computational Linguistics, 2020. 6.1, 6.2.1

[236] R. Challen, J. Denny, M. Pitt, L. Gompels, T. Edwards, and K. Tsaneva-Atanasova, "Artificial intelligence, bias and clinical safety," *BMJ Quality & Safety*, vol. 28, no. 3, pp. 231–237, 2019. 6.1

[237] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020. 6.1

[238] N. Voskarides, E. Meij, M. Tsagkias, M. de Rijke, and W. Weerkamp, "Learning to explain entity relationships in knowledge graphs," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Beijing, China), pp. 564–574, Association for Computational Linguistics, 2015. 6.1

[239] F. Godin, K. Demuynck, J. Dambre, W. De Neve, and T. Demeester, "Explaining character-aware neural networks for word-level prediction: Do they discover linguistic rules?," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 3275–3284, Association for Computational Linguistics, 2018. 6.1

[240] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016* (B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, eds.), pp. 1135–1144, ACM, 2016. 6.1

[241] D. Croce, D. Rossini, and R. Basili, "Auditing deep learning processes through kernel-based explanatory models," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 4037–4046, Association for Computational Linguistics, 2019. 6.1

[242] H. Kaur, H. Nori, S. Jenkins, R. Caruana, H. M. Wallach, and J. W. Vaughan, "Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning," in *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020* (R. Bernhaupt, F. F. Mueller, D. Verweij, J. Andres, J. McGrenere, A. Cockburn, I. Avellino, A. Goguey, P. Bjøn, S. Zhao, B. P. Samson, and R. Kocielnik, eds.), pp. 1–14, ACM, 2020. 6.1

[243] S. Wiegreffe and A. Marasović, "Teach me to explain: A review of datasets for explainable nlp," *arXiv preprint arXiv:2102.12060*, 2021. 6.1, 6.7.4

[244] S. Narang, C. Raffel, K. Lee, A. Roberts, N. Fiedel, and K. Malkan, "Wt5?! training text-to-text models to explain their predictions," *arXiv preprint arXiv:2004.14546*, 2020. 6.1

[245] S. Kumar and P. Talukdar, "NILE : Natural language inference with faithful natural language explanations," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 8730–8742, Association for Computational Linguistics, 2020. 6.1

[246] L. Aroyo and C. Welty, "Truth is a lie: Crowd truth and the seven myths of human annotation," *AI Magazine*, vol. 36, no. 1, pp. 15–24, 2015. 6.1

[247] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019. 6.1, 6.7.3, 7.1

[248] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 632–642, Association for Computational Linguistics, 2015. 6.1

[249] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, (Valencia, Spain), pp. 427–431, Association for Computational Linguistics, 2017. 1

[250] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018. 6.2.1

[251] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE Transactions on Neural Networks and Learning Systems*, 2020. 6.2.1

[252] M. Zhou, M. Huang, and X. Zhu, "An interpretable reasoning network for multi-relation question answering," in *Proceedings of the 27th International Conference on Computational Linguistics*, (Santa Fe, New Mexico, USA), pp. 2010–2022, Association for Computational Linguistics, 2018. 6.2.1

[253] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, "MathQA: Towards interpretable math word problem solving with operation-based formalisms," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 2357–2367, Association for Computational Linguistics, 2019. 6.2.1

[254] N. Pröllochs, S. Feuerriegel, and D. Neumann, "Learning interpretable negation rules via weak supervision at document level: A reinforcement learning approach," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 407–413, Association for Computational Linguistics, 2019. 6.2.1

[255] A. Srivastava and C. Sutton, "Autoencoding variational inference for topic models," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 6.2.2

[256] J. Su, S. Wu, D. Xiong, Y. Lu, X. Han, and B. Zhang, "Variational recurrent neural machine translation," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 5488–5495, AAAI Press, 2018. 6.2.2

[257] J. Gao, W. Bi, X. Liu, J. Li, G. Zhou, and S. Shi, "A discrete CVAE for response generation on short-text conversation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 1898–1908, Association for Computational Linguistics, 2019. 6.2.2, 6.3.1

[258] L. Fang, T. Zeng, C. Liu, L. Bo, W. Dong, and C. Chen, "Transformer-based conditional variational autoencoder for controllable story generation," *arXiv preprint arXiv:2101.00828*, 2021. 6.2.2, 6.3.1

[259] Z. Cheng, X. Dai, S. Huang, and J. Chen, "Variational explanation generator: Generating explanation for natural language inference using variational auto-encoder," *International Journal of Computer and Information Engineering*, vol. 15, no. 2, pp. 119–125, 2021. 6.2.2

[260] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 7871–7880, Association for Computational Linguistics, 2020. 6.3.2

[261] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, (Philadelphia, Pennsylvania, USA), pp. 311–318, Association for Computational Linguistics, 2002. 6.4, 7.6.4

[262] B. Eikema and W. Aziz, "Is MAP decoding all you need? the inadequacy of the mode in neural machine translation," in *Proceedings of the 28th International Conference on Computational Linguistics*, (Barcelona, Spain (Online)), pp. 4506–4520, International Committee on Computational Linguistics, 2020. 6.4

[263] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 2249–2255, Association for Computational Linguistics, 2016. 6.4.1

[264] Q. Chen, X. Zhu, Z.-H. Ling, D. Inkpen, and S. Wei, "Neural natural language inference models enhanced with external knowledge," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 2406–2417, Association for Computational Linguistics, 2018. 6.4.1

[265] Y. Gong, H. Luo, and J. Zhang, "Natural language inference over interaction space," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018. 6.4.1

[266] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 670–680, Association for Computational Linguistics, 2017. 6.4.1

[267] J. Zhou and S. Bhat, "Paraphrase generation: A survey of the state of the art," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5075–5086, 2021. 6.7.3, 7.1

[268] E. Cho, H. Xie, and W. M. Campbell, "Paraphrase generation for semi-supervised learning in NLU," in *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, (Minneapolis, Minnesota), pp. 45–54, Association for Computational Linguistics, 2019. 6.7.3

[269] O.-M. Camburu, B. Shillingford, P. Minervini, T. Lukasiewicz, and P. Blunsom, "Make up your mind! adversarial generation of inconsistent natural language explanations," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 4157–4165, Association for Computational Linguistics, 2020. 6.7.4

[270] L. Dong, J. Mallinson, S. Reddy, and M. Lapata, "Learning to paraphrase for question answering," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 875–886, Association for Computational Linguistics, 2017. 7.1

[271] M. Lee, J. Cimino, H. R. Zhu, C. Sable, V. Shanker, J. Ely, and H. Yu, "Beyond information retrievalmedical question answering," in *AMIA annual symposium proceedings*, vol. 2006, p. 469, American Medical Informatics Association, 2006. 7.1

[272] X. Yao and B. Van Durme, "Information extraction over structured data: Question answering with Freebase," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Baltimore, Maryland), pp. 956–966, Association for Computational Linguistics, 2014. 7.1

[273] Y. Liu, S. Li, Y. Cao, C.-Y. Lin, D. Han, and Y. Yu, "Understanding and summarizing answers in community-based question answering services," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, (Manchester, UK), pp. 497–504, Coling 2008 Organizing Committee, 2008. 7.1

[274] T. Hosking, H. Tang, and M. Lapata, "Hierarchical sketch induction for paraphrase generation," *arXiv preprint arXiv:2203.03463*, 2022. 7.1, 7.5.1, 7.6.1, 7.6.2, 7.6.5, 7.6.5

[275] D. Kauchak and R. Barzilay, "Paraphrasing for automatic evaluation," in *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, (New York City, USA), pp. 455–462, Association for Computational Linguistics, 2006. 7.1, 7.5.1

[276] S. Narayan, S. Reddy, and S. B. Cohen, "Paraphrase generation from latent-variable PCFGs for semantic parsing," in *Proceedings of the 9th International Natural Language Generation conference*, (Edinburgh, UK), pp. 153–162, Association for Computational Linguistics, 2016. 7.1, 7.5.1

[277] G. A. Miller, "WordNet: A lexical database for English," in *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992. 7.1

[278] A. Kumar, K. Ahuja, R. Vadapalli, and P. Talukdar, "Syntax-guided controlled generation of paraphrases," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 329–345, 2020. 7.1, 7.5.1

[279] Y. Meng, X. Ao, Q. He, X. Sun, Q. Han, F. Wu, J. Li, *et al.*, "Conrpg: Paraphrase generation using contexts as regularizer," *arXiv preprint arXiv:2109.00363*, 2021. 7.1, 7.5.1

[280] Y. Su, D. Vandyke, S. Baker, Y. Wang, and N. Collier, "Keep the primary, rewrite the secondary: A two-stage approach for paraphrase generation," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, (Online), pp. 560–569, Association for Computational Linguistics, 2021. 7.1, 7.5.1, 7.6.1, 7.6.2, 7.6.5, 7.6.5

[281] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006. 7.1

[282] T. Hosking and M. Lapata, "Factorising meaning and form for intent-preserving paraphrasing," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Online), pp. 1405–1418, Association for Computational Linguistics, 2021. 7.1, 7.5.1, 7.6.1, 7.6.5, 7.6.5

[283] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, and W. Ma, "Dual learning for machine translation," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds.), pp. 820–828, 2016. 7.1

[284] S.-Y. Su, C.-W. Huang, and Y.-N. Chen, "Dual supervised learning for natural language understanding and generation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 5472–5477, Association for Computational Linguistics, 2019. 7.1

[285] S.-Y. Su, Y.-S. Chuang, and Y.-N. Chen, "Dual inference for improving language understanding and generation," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, (Online), pp. 4930–4936, Association for Computational Linguistics, 2020. 7.1

[286] S.-Y. Su, C.-W. Huang, and Y.-N. Chen, "Towards unsupervised language understanding and generation by joint dual learning," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 671–680, Association for Computational Linguistics, 2020. 7.1

[287] M. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016. 7.2, 7.3.2

[288] H. Guo, R. Pasunuru, and M. Bansal, "Dynamic multi-level multi-task learning for sentence simplification," in *Proceedings of the 27th International Conference on Computational Linguistics*, (Santa Fe, New Mexico, USA), pp. 462–476, Association for Computational Linguistics, 2018. 7.2, 7.3.2

[289] H. Guo, R. Pasunuru, and M. Bansal, "Soft layer-specific multi-task summarization with entailment and question generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 687–697, Association for Computational Linguistics, 2018. 7.2, 7.3.2

[290] Y. Wang, C. Zhai, and H. Hassan, "Multi-task learning for multilingual neural machine translation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 1022–1034, Association for Computational Linguistics, 2020. 7.2, 7.3.2

[291] J. Du, E. Grave, B. Gunel, V. Chaudhary, O. Celebi, M. Auli, V. Stoyanov, and A. Conneau, "Self-training improves pre-training for natural language understanding," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Online), pp. 5408–5418, Association for Computational Linguistics, 2021. 7.2.1

[292] E. Chang, V. Demberg, and A. Marin, "Jointly improving language understanding and generation with quality-weighted weak supervision of automatic labeling," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, (Online), pp. 818–829, Association for Computational Linguistics, 2021. 7.2.1

[293] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989. 7.2.1

[294] S. Burrows, M. Potthast, and B. Stein, "Paraphrase acquisition via crowdsourcing and machine learning," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 3, pp. 1–21, 2013. 7.2.1

[295] Z. Cao, C. Luo, W. Li, and S. Li, "Joint copying and restricted generation for paraphrase," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA* (S. P. Singh and S. Markovitch, eds.), pp. 3152–3158, AAAI Press, 2017. 7.2.1

[296] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *Proceedings of the 35th International Conference on Machine Learning, ICML*

2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (J. G. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 882–891, PMLR, 2018. 7.2.2

[297] M. F. Balin, A. Abid, and J. Y. Zou, "Concrete autoencoders: Differentiable feature selection and reconstruction," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 444–453, PMLR, 2019. 7.2.2

[298] Z. Yang, Z. Hu, C. Dyer, E. P. Xing, and T. Berg-Kirkpatrick, "Unsupervised text style transfer using language models as discriminators," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 7298–7309, 2018. 7.4.2, 7.4.2

[299] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, (Beijing, China), pp. 425–430, Association for Computational Linguistics, 2015. 7.5.1

[300] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014. 7.6.1

[301] Z. Li, X. Jiang, L. Shang, and Q. Liu, "Decomposable neural paraphrase generation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 3403–3414, Association for Computational Linguistics, 2019. 7.6.1

[302] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, (Barcelona, Spain), pp. 74–81, Association for Computational Linguistics, 2004. 7.6.4

[303] H. Sun and M. Zhou, "Joint learning of a dual SMT system for paraphrase generation," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (Jeju Island, Korea), pp. 38–42, Association for Computational Linguistics, 2012. 7.6.4

[304] N. Madnani, J. Tetreault, and M. Chodorow, "Re-examining machine translation metrics for paraphrase identification," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Montréal, Canada), pp. 182–190, Association for Computational Linguistics, 2012. 7.6.4

[305] S. Wubben, A. van den Bosch, and E. Krahmer, "Paraphrase generation as monolingual translation: Data and evaluation," in *Proceedings of the 6th International Natural Language Generation Conference*, 2010. 7.6.4

[306] A. Prakash, S. A. Hasan, K. Lee, V. Datla, A. Qadir, J. Liu, and O. Farri, "Neural paraphrase generation with stacked residual LSTM networks," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, (Osaka, Japan), pp. 2923–2934, The COLING 2016 Organizing Committee, 2016. 7.6.5, 7.6.5