SENTENCE REPRESENTATION LEARNING AND GENERATION FOR NEURAL MACHINE TRANSLATION

A THESIS
SUBMITTED TO THE SCHOOL OF COMPUTING
FACULTY OF COMPUTING, ENGINEERING
AND THE BUILT ENVIRONMENT
OF ULSTER UNIVERSITY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY (Ph.D.)

ISAAC KOJO ESSEL AMPOMAH
NOVEMBER 2020

I confirm that the word count of this thesis is less than 100,000 words.

Dedicated to the loving memory of my mother Dora Essel for her love, encouragement and support throughout my life. It is sad that you are not here but I know you would have been proud of this achievement.

Declaration

I hereby declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree diploma or other qualifications. The thesis may be freely copied and distributed provided the source is explicitly acknowledged.

Signed: _____

Isaac Kojo Essel Ampomah

Acknowledgments

First and foremost, I gratefully acknowledge the support from the Vice-Chancellor's Research Scholarship provided by Ulster University.

I would like to express my profound gratitude to my supervisors Prof. Sally McClean, Dr. Glenn Hawe and Dr. Lin Zhiwei. They have always been there for every stage of this PhD research work despite their busy schedules. Their words of encouragement and creative suggestions guided me from the problem definitions, models' designs and implementations. Training deep learning models is computational resource intensive and without the GPU Cluster access provided by Andrew Ennis and Dr. Joseph Rafferty, training the models presented in this thesis wouldn't have been possible. I would also like to thank the staff at the Doctoral College (DOC) in Jordanstown for their support during the period of study. They made the challenging task of completing a PhD into a rewarding and incredibly fun experience. To all the friends I made at Ulster University (Wulme Derry, Linda Hooper, Zora Saskova, Howard Ayo, Fawule Emmanuel and Joseph Ansong), thanks for providing the perfect balance between academic works and distractions and above all for keeping me sane. I'll truly miss our time together.

Last but not the least, my sincere gratitude goes to my parents, my sisters and brothers for their unconditional love and support during the different stages of my education. I appreciate them for being there for me and will always have a special place in my heart for them. Finally, I would like to thank my biggest fans, my wife Kate Amo Mensah and children (Jason and Kevin) for sharing both the difficult and happy times with me. The life of a PhD student with family can be challenging, so thank you for the love and bringing a different meaning to my life. As this journey is over, I know you will always be there for me as we forge forward in life.

List of Publications

Conference Papers

[1]. **Isaac K. E. Ampomah**, Zhiwei Lin, Sally McClean, and Glenn Hawe. "Gated Task Interaction Framework for Multi-task Sequence Tagging." In 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1-8. IEEE, 2019.

[2]. **Isaac K. E. Ampomah,**, Sally McClean, Zhiwei Lin, and Glenn Hawe. "JASs: Joint Attention Strategies for Paraphrase Generation." In International Conference on Applications of Natural Language to Information Systems, (pp. 92-104). Springer, 2019.

Journal Articles

[1]. **Isaac K. E. Ampomah**, Sally McClean, Zhiwei Lin, and Glenn Hawe. “Every Layer Counts: Multi-Layer Multi-Head Attention for Neural Machine Translation.” In The Prague Bulletin of Mathematical Linguistics 115:51–82.

Under-review Articles

[1]. **Isaac K. E. Ampomah**, Sally McClean, and Glenn Hawe. “Dual Contextual Module for Neural Machine Translation.” under review in Machine Translation, Springer.

[2]. **Isaac K. E. Ampomah**, Sally McClean, Zhiwei Lin, and Glenn Hawe. “Encoder-based Multi-level Supervision for Neural Machine Translation.”, under review in Neural Computing and Applications, Springer.

ABSTRACT

Machine translation (MT) systems have become indispensable tools allowing for the automatic translation of texts from one language to another. Earlier MT models ranged from rule-based systems to statistical MT (SMT) models. However, in recent years, neural machine translation (NMT) has attracted greater attention within the MT research community. The strength and success of NMT models over prior systems can be attributed to their ability to automatically learn the linguistic features required for the translation task without explicit feature engineering. Typically, NMT systems employ an encoder-decoder architecture to model and learn the target translation. The encoder learns the semantic representation of the source sentence from which the decoder generates the target translation. Therefore, the translation performance of an NMT model relies heavily on the representation and generation ability of both encoder and decoder subnetworks. Besides, the overall performance of any MT system is also affected by the linguistic structures and properties of the language pairs under consideration. This implies that the architectural design of the NMT system is of significant importance to efficiently learn the necessary linguistic information to achieve higher translation performance across different language pairs. Accordingly, the overall aim of this thesis is to design and build architectures to perform the translation task more efficiently. The contributions of the thesis are in two main folds: (1) To ensure minimal loss of source information during the target translation, two main joint attention strategies are presented in Chapter 3 to allow the decoder access to source information captured by different encoding layers. (2) Enhancing the sentence representational ability of the encoder and decoder subnetworks using strategies including (a) exploiting the strengths of multitask learning and auxiliary training approaches to design an encoder-based multi-level supervision framework in Chapter 4; (b) improving the performance of the self-attention mechanism at capturing efficiently the local and global contextual information and dependencies in Chapter 5. Evaluations on multiple language translation tasks show that the approaches proposed in this work significantly enhance the sentence representation and generation ability of the encoder-decoder architecture consequently improving the overall translation performance of the NMT model.

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Introduction

Machine Translation (MT) is a field of natural language processing (NLP) that investigates the automatic translation of texts from a source language to texts in a target language. The primary goal of MT research is to achieve translation quality comparable to human translators. However, this is a challenging task mainly due to reasons such as the ambiguity and flexibility of human language.

1.1 Brief overview of Machine Translation

The building of machine translation systems has been the theme of many research works and contributions since the late 1940s. The beginning of MT can be traced to Warren Weaver's proposals for computer-based machine translation (Raley, 2003; Hutchins, 1993). Machine translations systems can be categorised along two possible dimensions: (1) depth of analysis and generation and (2) the linguistic level at which transfer is performed. The Vauquois triangle in Figure 1.1 defines three main levels of translation abstractions (Sangodkar and Damani, 2012; Saers, 2011).

- **Direct translation**: the words in the input sentence or text are directly translated without any intermediary representation. The translation is performed with little to no linguistic knowledge. For each source word, a dictionary specifying a set of rules for translating the word is employed.

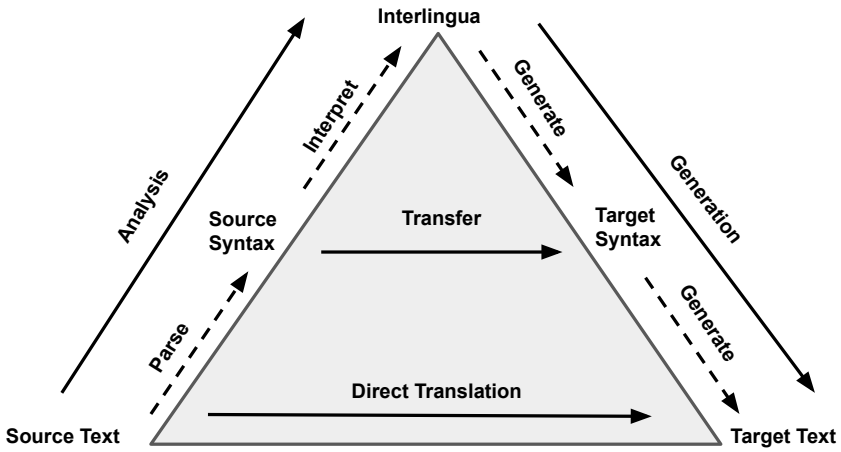


Figure 1.1: Vauquois Triangle showing the levels of translation abstractions.

- **Transfer-based translation**: generates the translation based on intermediate representation in three stages: analysis, transfer and generation. The analysis stage converts the source sentence into an abstract or intermediate representation. During the transfer stage, the abstract representation of the source sentence is converted into the corresponding abstract representation of the target using a set of transfer rules. Finally, the generation stage converts the target-language representation from the transfer stage into the output translation.

- **Interlingua-based translation**: unlike the transfer-based approach, this approach generates the output translation using only the analysis and generation stages. Here, the analysis performed on the source language text produces a language-independent representation of meaning. The generation phase converts the meaning representation of the sentence into the output translation. This approach allows for the easy development of a multilingual system. Despite the beauty of this approach, it is difficult to compose a truly language-independent representation.

Over the years, different types of translation systems have been proposed. These are the Knowledge-driven (Rule-based), and data/corpus-driven (Example-based, Statistical-based and most recently the Neural network-based systems) approaches.

Rule-based: This is one of the earliest machine translation systems. As the name implies, it employs a set of rules designed by linguistic experts to guide the translation process. These sets of rules are usually used along with word dictionaries to translate a source text into the corresponding target text. Notable examples of rule-based systems include the Apertium (Forcada et al., 2011) and Lucy LT (Alonso and Thurmair, 2003). Rule-based systems are employed to translate at either the interlingua or transfer levels. Generally, the rules are applied in 3 phases: analysis, transfer, and generation. The rules are language pair specific as such, cannot generalise to new language pairs. This requires extensive human effort to define the translation rules and the word dictionaries. Despite the above pitfall, rule-based systems usually generate translations with well defined grammatical structures.

Example-based MT (EBMT): This is a data-driven approach to MT, which does not rely on linguistic rules to perform the translation task. EBMT systems such as (Zhang et al., 2001, 2011; Rana and Atique, 2016; Chua et al., 2017) build a source fragments translation database from a bilingual parallel corpus. The translation of any given sentence is generated using example translations of similar source language texts found within the database. The translation is performed in three steps: matching, retrieval/extraction, and recombination. Under EBMT, the input sentence is segmented into fragments/phrases, which are then matched against source text fragments in the database. The final target sentence is generated from the recombination of the translations of the matched phrases in the database. To achieve higher translation performance, EBMT systems require larger parallel corpora, which in most cases, is not readily available.

Statistical-based MT (SMT): Similar to the EBMT, statistical MT systems (Gimpel and Smith, 2008; He et al., 2008; Huang et al., 2006) do not rely on translation rules to convert the source sentence into the corresponding target sentence. The translation is performed via statistical models obtained by analyzing source-target pairs in bilingual text corpora. There are different types of SMT models which are word-based MT (Tillmann and Ney, 2003; Germann et al., 2004), phrase-based MT (PBMT) (Gimpel and Smith, 2008; Zens and Ney, 2004), and Syntax-based MT models (He et al., 2008; Huang et al., 2006). PBMT models have been shown to significantly best performance among the SMT systems. A typical SMT system consists of the language model and the translation model. The translation model trained on the bilingual corpus is employed to generate target hypotheses for

the given source sentence. In contrast, the language model trained on a monolingual corpus of the target language is employed to estimate the probability of each target hypothesis and playing a significant role in ensuring fluency of the output translations. The decoding unit combines the estimates from the two sub-models to generate translations for new sentences. Although SMT models, in general, produce high-quality translations, statistical anomalies can significantly degrade its performance. Besides, SMT has problems dealing with different word-order in different source and target languages.

Neural network-based MT (NMT): The recent advances in AI applications to problem-solving and the advent of high-performance computing devices and resources have given rise to new machine translation models based on deep neural networks. The NMT models such as (Bahdanau et al., 2015; Sutskever et al., 2014; Gehring et al., 2017; Vaswani et al., 2017) have been shown to outperform the statistical-based models, including Syntax-based and PBMT on shared translation tasks and different data settings. Furthermore, the work by Hassan et al. (2018) shows that it is possible to achieve professional human-level translation with deep NMT models. The translation problem is formulated as a sequence to sequence (Seq2Seq) problem, where the NMT model is trained on a large parallel corpus to learn the target sentence generation from a given source sentence. NMT models are generally based on the *Encoder-Decoder* architectural framework. The encoder component is employed to generate the semantic representation of the source sentence. Based on the generated source representation, the decoder generates the corresponding target translation. Different from SMT models, NMT directly models the machine translation as a conditional language model without the explicit use of language models trained on the target monolingual corpus. To date, the majority of NMT systems are recurrent neural network-based models (Bahdanau et al., 2015; Sutskever et al., 2014; Wu et al., 2016). However, recent research works by Gehring et al. (2017) and Vaswani et al. (2017) have successfully paved the way for new models based entirely on convolutional neural networks and attention mechanisms. These are further discussed in Section 2.3.3. Despite the potential performance gain of NMT, it suffers from many issues common to all deep learning models including:

- Generally, the translation performance of any NMT model is significantly affected by the amount of training data. It requires a large amount of training data to efficiently and effectively optimise the model parameters, which are usually in the millions.

This makes it difficult to train any reasonable MT models on language pairs with extremely low amounts of parallel data. Recent attempts made to improve the performance on low-resource languages involve incorporating monolingual corpora to train the model (Siddhant et al., 2020; Currey et al., 2017).

- The network structure and the number of trainable parameters affect the overall computational complexity of the NMT model during both the training and decoding phases. Generally, the more complicated the network, the slower the decoding and training processes. Speedup can be achieved via model and data parallelization techniques using specialised hardware such as GPUs and TPUs.

NMT is built upon the ability of deep neural networks to automatically learn the required features for the translation task directly from the source-target parallel corpus with little to no external linguistic features. This is different from SMT and EBMT systems which usually require high-quality linguistic features such as the parts-of-speech (POS) tags, and orthographic features to achieve higher translation performance.

1.2 Research Questions

The translation performance of an NMT system depends to a large extent on the representation and generation ability of both encoder and decoder subnetworks. Besides, the translation performance of any MT system is also affected by the linguistic structures and properties of the language pairs under consideration. Therefore, the design of NMT architectures is of crucial importance to effectively and efficiently learn the necessary linguistic information to further enhance the quality of the sentence translations across different language pairs. This thesis aims to study and design deep neural architectures for performing sequence to sequence learning more effectively. Below is the list of research questions considered:

Question 1: Each encoding layer captures a different level of abstraction of the source sentence (Raganato et al., 2018; Belinkov et al., 2017; Peters et al., 2018). The encoder employs the entire stack of layers to learn the hidden source representation for the translation task. However for a deeper network, there is no guarantee that the last encoder layer's output is the best representation for the target generation due to the

nature of information flow across the time-steps and layers of the encoder subnetwork (Wang et al., 2018a; Dou et al., 2018; Ampomah et al., 2019b). An interesting question here is: **How can we ensure minimal loss of information when mapping from the source sequence to the target sequence?** The main focus of this research question is improving the target generation ability of the decoder subnetwork by leveraging source representations from multiple encoding layers.

Question 2: NMT frameworks generally have a single decoding subnetwork (referred to the *Main-Decoder*) connected to the final encoding layer. The ability of encoder subnetwork at learning the source information can significantly affect the overall performance of the translation model. **To improve the source sentence representational ability of the encoder subnetwork, what motivation or inspiration can be drawn from both deep representational learning and multi-task learning (MTL) approaches to NMT and sequence labelling tasks such as Named Entity Recognition?** The focus here is to evaluate whether the translation performance of the *Main-Decoder* can be improved by jointly training along with auxiliary decoders connected to lower-level encoder layers on the same target sequence generation task.

Question 3: **How to leverage the local contextual information provided by surrounding words effectively without sacrificing the global context learning ability of the self-attention mechanism?** This research question focuses on improving the sentence representational ability of the self-attention mechanism by leveraging both the global and local contextual information.

1.3 Research Objectives

1. To explore neural attention computation strategies to leverage the source representations from multiple encoding layers.

2. To improve the source representation ability of the encoder subnetwork via Multi-level supervision.

3. To implement the Dual Contextual module, an extension to self-attention operation to leverage the global and local contextual information without restricting the scope

or span of the self-attention mechanism.

1.4 Thesis Outline

This chapter introduced the task of machine translation (MT), the research questions and our contributions to NMT. The subsequent chapters are tailored towards answering the research questions under consideration. The primary contributions of this thesis are presented in the Chapters 3 to 5. The outline of the reminder of this thesis are follows:

- **Chapter 2 Background**: This chapter provides background knowledge on the sequence to sequence application for the task of NMT. The chapter starts with a brief overview of neural language models. The subsequent sections explore the neural architecture and the techniques, including the data preprocessing and inference algorithms employed for the task of sequence to sequence learning. The last section of the chapter presents the discussion on the relevant related works.

- **Chapter 3 JASs: Joint Attention Strategies**: This chapter introduces the *Joint Attention Strategies* to leveraging source representations from multiple encoding layers. The goal is to ensure the minimal loss of source information when generating the target sequence from the source sequence. The experimental results on three language pairs confirm the potential performance gain over models exploiting the source representation from only the final encoder layer.

- **Chapter 4 Multi-level Supervision Strategies**: An approach to further enhance the source representation ability of the encoding subnetwork by connecting multiple decoding subnetworks to different encoding layers is presented. This training strategy benefits from the inductive bias aspect of conventional multi-task learning (MTL). Specifically, the encoder receives gradient signals from all connected decoders during training. Therefore, to support the target generation from each connected decoder, the lower-level layers are forced to capture the necessary source linguistic information. The experimental results and analysis on four language pairs demonstrate consistent improvement over models with a single decoder connected to only the final encoder layer.

- **Chapter 5 Dual Contextual Modelling**: In this chapter, we introduce the *Dual Contextual* (DC) module to leverage the local contextual information without restricting the performance of the self-attention mechanism in terms of learning the long-range dependencies between the tokens. Further analysis suggests the proposed DC module improves the model's performance at learning the surface (such as the word content and sentence length) and syntactic features without sacrificing the ability to learning deeper semantic features required for the translation task.
- **Chapter 6 Conclusion and Future work**: This chapter provides the summary of our findings and contributions of research works presented in this thesis. Furthermore, possible avenues of future work are also presented.

Chapter 2

Background: Neural Machine Translation

This chapter provides the background knowledge for neural network-based sequence generation. The main focus of this thesis is to improve the performance of the application of deep neural networks to the task of machine translation. However, most of the discussions presented here are applicable to other sequence generation problems such as paraphrase generation (Hasan et al., 2016; Ampomah et al., 2019b), image caption generation (Hossain et al., 2019; Su et al., 2020) and document summarization (Gui et al., 2019; Song et al., 2019b; Mohd et al., 2020). Brief introductions to the n-gram language model and the neural language model are provided in Sections 2.1 and 2.2 respectively. Section 2.3 presents an overview of Sequence to Sequence (Seq2Seq) architectures and Section 2.7 briefly reviews the relevant related works.

2.1 Language Modelling

A language model (LM) predicts the next likely word in a sequence based on the preceding context. Given a sentence $X = (x_0, x_1, \cdots, x_M)$ (where x_t is the t^{th} token in the sequence), the probability of the sentence $\mathsf{P}(X)$ is estimated as:

$$\mathsf{P}(X) = \prod_{t=0}^{M} \mathsf{P}(x_t \mid x_{<t}) \tag{2.1}$$

where $\mathsf{P}(x_t \mid x_{<t})$ is the conditional probability of the current token x_t conditioned on the context provided by the partial sequence of tokens $x_{<t} = x_0, \cdots, x_{t-1}$. However, it is difficult to directly estimate $\mathsf{P}(x_t \mid x_{<t})$ as the size of the context provided by $x_{<t}$ increases. A simple solution can be derived via the Markov assumption, where the context to consider is limited to a fixed window of size $n-1$ words. The LM based on this assumption is termed as the n-gram LM. For example, the *Unigram* LM is where $n = 1$. Formally, for the n-gram LM, the $\mathsf{P}(X)$ is reformulated as:

$$\mathsf{P}(X) = \prod_{t}^{M} \mathsf{P}(x_t \mid x_{t-n+1:t-1}) \tag{2.2}$$

It is possible to have $(t - n + 1)$ smaller than 0 especially when $t \leq n - 2$. To handle this, Bengio et al. recommends using a special token (e.g. <BOS>[1]) to represent the tokens where $t \leq n - 2$. The n-gram LM has to keep track of (or store) all possible n-grams available within the training corpus to achieve higher performance. This is both memory and computationally expensive especially on larger corpus (Pibiri and Venturini, 2019). Specifically, the number of possible word sequences grows exponentially as the size of the vocabulary employed to train the LM increases. Furthermore, n-gram based statistical language models such as Niesler et al. (1998); Stolcke (2002); Buck et al. (2014); Federico et al. (2008) do not generalise well across different domains and also on unknown n-gram sequences not explicitly present in the training corpus.

2.2 Neural Language Models

To address the above problems of the n-gram LM, recent language models (Bengio et al., 2003; Mikolov and Zweig, 2012; Kim et al., 2016; Peters et al., 2018; Devlin et al., 2019) are based on the neural network architecture. Generally, the input to the network is a sequence of discrete tokens which are then mapped into their corresponding continuous vector representation (word vector) space. This is known as the word embedding technique. The embedding approach captures the linguistic information, including the syntactic and semantic properties of words within the input sentence (Collobert et al., 2011). Therefore, it can provide a good generalization over unseen words or sequences. The neural LMs are

[1] Beginning Of Sentence

Function	Formula
Hyperbolic Tangent (tanh)	$\mathrm{g}(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$
Sigmoid (σ)	$\mathrm{g}(z) = \frac{1}{1+\exp(-z)}$
Rectified Linear Units (ReLu)	$\mathrm{g}(z) = \max(0, z)$

Table 2.1: Non-linear activation functions widely used.

usually based on either the *Feed-Forward Network* (FFN) (Bengio et al., 2003) or *Recurrent Neural Networks* (RNNs) (Mikolov and Zweig, 2012; Kim et al., 2016). These neural LMs are presented in the following subsections.

2.2.1 Feed-Forward Network

The earliest neural LM was proposed by (Bengio et al., 2003) where a L-layer feed-forward network was trained to estimate $\mathsf{P}(x_t|x_{t-n+1:t-1})$. A word embedding lookup table $w(\cdot)$ (with a vocabulary size of V tokens) is employed to generate the continuous representation for each token in the preceding $n-1$ sequence. The resulting embedding vectors are concatenated to generate the representation h_t^0 which is passed as input to the FFN:

$$h_t^0 = [w(x_{t-n+1}); w(x_{t-n+2}); \cdots ; w(x_{t-1})] \in \mathbb{R}^{(n-1)\cdot d}$$

where $w(x_t)$ is the embedding vector of the x_t token and $[\cdot;\cdot]$ denotes the vector concatenation operation. Based on the generated h_t^0, the L-layer network computes the hidden

representation h_t^L through a series of non-linear transformations as:

$$\begin{aligned} h_t^1 &= \mathrm{g}\left(h_t^0 W^1 + b^1\right) \\ h_t^2 &= \mathrm{g}\left(h_t^1 W^2 + b^2\right) \\ &\vdots \\ h_t^l &= \mathrm{g}\left(h_t^{(l-1)} W^l + b^l\right) \\ &\vdots \\ h_t^L &= h_t^{(L-1)} W^o + b^o \end{aligned} \tag{2.3}$$

where W^l and b^l are, respectively, the trainable weight and bias employed by the l-th layer (where $l < L$) to transform the input representation $h_t^{(l-1)}$. The parameters of the output layer L are $W^o \in \mathbb{R}^{d \times V}$ and $b^o \in \mathbb{R}^V$. $\mathrm{g}(\cdot)$ is a non-linear activation function (see Table 2.1). Given h_t^L, the probability of the next word is computed as:

$$\mathsf{P}(x_t | x_{t-n+1:t-1}) \approx \mathsf{P}(x_t | x_1, \cdots, x_{t-1}) = \mathrm{softmax}\left(h_t^L\right)$$

where

$$\mathrm{softmax}\left(z\right)_j = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)} \textit{ for j = 1,...,K and z=}(z_1, \ldots, z_K)$$

Optimisation algorithms including the Stochastic Gradient (SGD) and Adam (Kingma and Ba, 2014) are employed to train the neural network by minimizing the negative log-likelihood of the training corpus. The FFN achieved significant improvement over state-of-the-art n-gram models (Niesler et al., 1998; Chen and Goodman, 1999).

2.2.2 Recurrent Neural Networks (RNN) based LM

Despite the performance gain over traditional n-gram LMs, FFN based LM predicts the next word based on a fixed-length context. The fixed-length contextual constraint on the FFN can be alleviated with RNN. In theory, RNN can leverage unlimited contextual information (Mikolov and Zweig, 2012). Motivated by this, (Mikolov and Zweig, 2012; Mikolov et al., 2013) proposed the Recurrent LM based on the Elman network (Elman, 1990) (simple RNN). The network architecture of the Recurrent LM is illustrated in Figure 2.1.

Similar to the FFN, the RNN model employs a word embedding layer to generate the

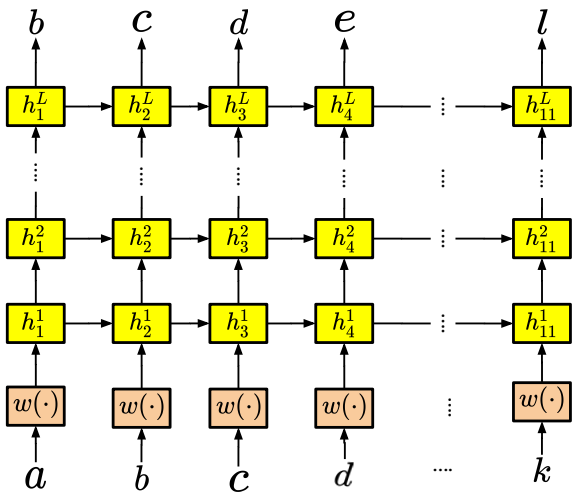


Figure 2.1: An illustration of a multi-layer Recurrent Neural Network (RNN) based language model.

continuous representation of each token in the input sequence. However, the output of the l-th layer is calculated as:

$$h_t^l = \mathrm{g}(h_{t-1}^l U^l + h_t^{l-1} W^l) \tag{2.4}$$

where $\mathrm{g}(\cdot)$ is the non-linear activation which is usually a *logistic sigmoid* or $\tanh$. h_t^l is the hidden state at time step $t \in [1, 2, \ldots, T]$. The hidden state h_{t-1}^l provides the summary of the contextual information with respect to tokens processed before the time step t. For the depth zero (i.e. $l = 0$), the input to the network is $h_t^0 = w(x_t)$, the embedding vector for the x_t token. $\{W^l, U^l\} \in \mathbb{R}^{d \times d}$ are the model weights of layer l shared across all the time steps. The output from the L-th layer is used to estimate the probability over the next token:

$$\mathsf{P}(x_t | x_1, \cdots, x_{t-1}) = \mathrm{softmax}(h_t^L W^o) \tag{2.5}$$

where $W^o \in \mathbb{R}^{d \times V}$ is the weight matrix employed to project the hidden state h_t^L before the application of the softmax activation function. Backpropagation through time (BPTT) (Werbos, 1988) is generally employed to train the RNN model efficiently. Due to the nature of information flow across the time steps, the simple RNN is generally difficult to train mainly due to the exploding and vanishing gradient problems (Bengio et al., 1994). In (Pascanu et al., 2012), the gradient norm clipping technique is shown to be effective in

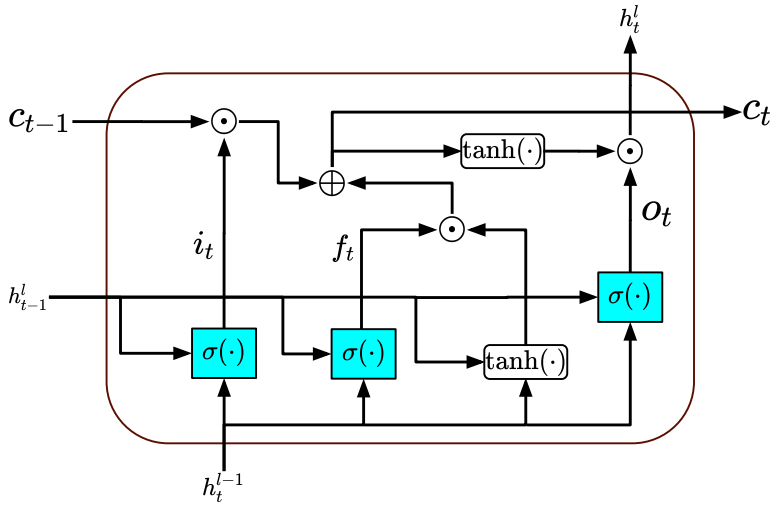


Figure 2.2: Structure of the LSTM cell unit comprising of a memory unit (c_t) and three gating units (the input gate i_t, forget gate f_t and the output gate o_t) to control the flow of information.

dealing with the exploding gradient problem. Specifically, the gradient γ is clipped based on its norm $||\gamma||$ if γ is greater than a predefined threshold η (i.e. $\gamma \leftarrow \frac{\eta\gamma}{||\gamma||}$).

2.2.3 Long Short-term Memory Networks (LSTM)

LSTM is a variant of RNN introduced by Hochreiter and Schmidhuber (1997) to tackle the vanishing gradient problem of the simple RNN. The main difference between the simple RNN and LSTM is the computation of the hidden state h_t at time step t. Specifically, the RNN node is replaced with an LSTM cell. Figure 2.2 illustrates the network architecture of the LSTM cell. As shown, the LSTM cell employs a memory unit (c_t) and three gating units (the forget gate f_t, the input gate i_t and the output gate o_t) to control the flow of

information. Unlike Equation (2.4), the hidden state h_t^l is reformulated as:

$$\begin{aligned} i_t &= \sigma\left(W_i h_t^{l-1} + U_i h_{t-1}^l + b_i\right) \\ f_t &= \sigma\left(W_f h_t^{l-1} + U_f h_{t-1}^l + b_f\right) \\ \hat{C} &= \tanh\left(W_c h_t^{l-1} + U_c h_{t-1}^l + b_c\right) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \hat{C} \\ o_t &= \sigma\left(W_o h_t^{l-1} + U_o h_{t-1}^l + b_o\right) \\ h_t^l &= o_t \odot \tanh\left(c_t\right) \end{aligned} \tag{2.6}$$

where $\sigma(\cdot)$ is sigmoid activation function and $\odot$ denotes the element-wise multiplication. The $\{W_i, W_f, W_c, U_i, U_f, U_c, W_o\}$ and $\{b_i, b_f, b_c, b_i, b_f, b_c, b_o\}$ are, respectively, the trainable weights and biases employed to compute the hidden state h_t^l. c_t is the memory cell state for the current time step t and $\hat{C}$ is the information used to update the memory cell. The f_t decides the part of the memory c_{t-1} to be discarded (forgotten) and i_t controls the amount of information in $\hat{c}$ to be considered in the computation of c_t. Finally, the output gate o_t determines the part of c_t to use to generate h_t^l.

A simplified alternative of LSTM, the Gated Recurrent Unit (GRU) (Cho et al., 2014; Chung et al., 2015), has been shown to achieve comparable performance to the LSTM cell with a fewer number of parameters. A notable difference between GRU and LSTM is that the "update" gate unit of the GRU combines the operations of the input and forget gates.

2.3 Sequence to Sequence (Seq2Seq) Models

The Seq2Seq model generates a target sequence $Y = [y_1, y_2, \cdots, y_Z]$ of length Z from a given source sequence $X = [x_1, x_2, \cdots, x_M]$, where x_i and y_t are the i^{th} and t^{th} tokens of X and Y respectively. For the task of NMT, x is termed as a sentence in the source language and Y is the corresponding sentence in the target language. Figure 2.3 illustrates the network architecture employed to generate the target sequence from the source sentence. As shown, the architecture comprises the encoder and decoder subnetworks. The goal of the encoder subnetwork is the generation of the hidden representation $v \in \mathbb{R}^d$ (where d is the dimension) based on the source sentence. On the other hand, the decoder subnetwork takes as input the source representation v to predict the probability distribution of each

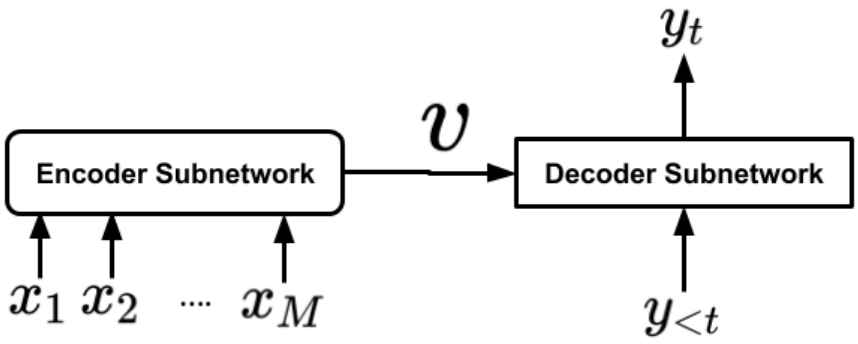


Figure 2.3: Encoder-Decoder framework for the task of sequence to sequence generation. $X = [x_1, x_2, \cdots ; x_M]$ is the input source sequence. $v \in \mathbb{R}^d$ is the fixed hidden representation of the source sequence. y_t is the target token at decoding step t and $y_{<t} = [y_1, y_2, \cdots ; y_{t-1}]$ denotes the partial sequence of target tokens generated before time step t.

target token y_t. Therefore, the decoder subnetwork can be considered as a language model generating the target tokens conditioned on the source sequence.

2.3.1 Attention Mechanism

As shown in Figure 2.3, the encoder subnetwork encodes the source sentence of any arbitrary length into a fixed-length vector passed to the decoder for the target generation. However, learning a fixed-length representation v creates a bottleneck during the target generation especially on longer sentences (Bahdanau et al., 2015). This is because as information flows through the encoder, the local contextual information gets diluted which results in loss of information and consequently resulting in poor performance. As a remedy, Bahdanau et al. (2015) proposed exploiting the neural attention mechanism, further improving the performance. Under the neural attention mechanism, the contextual information of each source token is kept and is referenced during the generation of each target token y_t. Specifically, during the decoding step t, the attention mechanism allows the decoder subnetwork to peak through encoder outputs to utilise the local contextual information to improve the mapping from the source sequence to the target sequence.

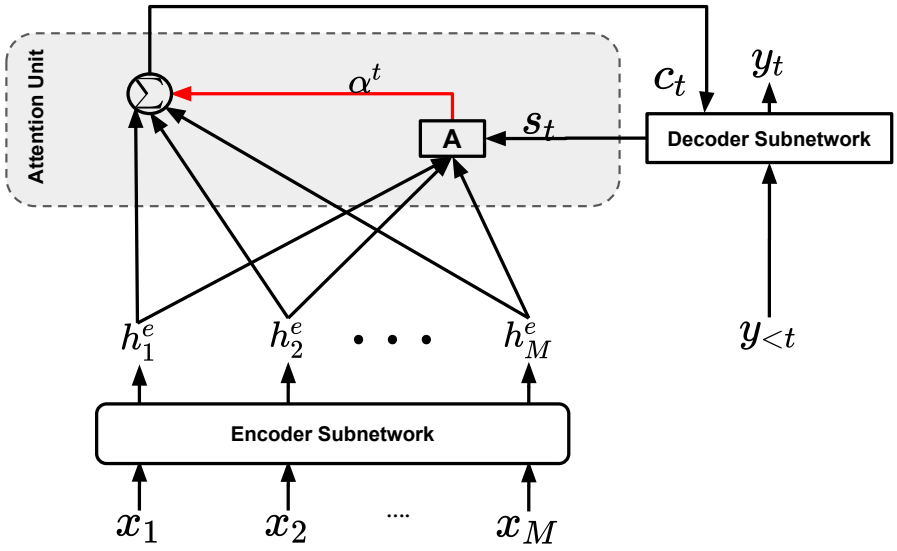


Figure 2.4: Illustration of the attention computation at decoding step t. c_t is the contextual representation generate from the source representation $H^e = [h_1^e, h_2^e, \cdots, h_M^e]$, where h_i^e is the hidden representation of x_i. α_t is the attention weight distribution. s_t denotes the decoder's hidden state.

A typical attention mechanism is visualised in Figure 2.4. As shown, the encoder generates the hidden representation $H^e = [h_1^e, h_2^e, \cdots, h_M^e]$, where $h_i^e \in \mathbb{R}^d$ is the hidden representation of the source token x_i. The decoder computes the probability distribution of the target token y_t based on the decoder's hidden state s_t and the contextual representation c_t obtained via a neural attention over $H^e \in \mathbb{R}^{M \times d}$:

$$\mathsf{P}(y_t|y_1, \cdots, y_{t-1}) = \text{softmax}(W_o[c_t; s_t] + b_o) \tag{2.7}$$

where W_o and b_o are trainable parameters and $[\cdot;\cdot]$ denotes the concatenation operation. The contextual vector c_t is obtained from the weighted summation of all the h_i^e vectors of source tokens:

$$c_t = \sum_{i=1}^{M} \alpha_i^t h_i^e \tag{2.8}$$

where $\alpha^t = [\alpha_1^t, \alpha_2^t, \cdots, \alpha_M^t]$ is the attention weight distribution over the source tokens at the decoding step t. α_i^t is the attention weight with respect to the i^{th} source token calculated

as:

$$\alpha_i^t = \frac{\exp(\text{score}(s_t, h_i^e))}{\sum_{k=1}^{M} \exp(\text{score}(s_t, h_k^e))}$$

where $\text{score}(\cdot)$ is the scoring function to model the alignment between the s_t and h_i^e. A number of scoring functions have been explored in literature. The three most widely used ones are:

- **Additive**: $\text{score}(a, b) = v^\intercal \tanh(W_1 a + W_2 b)$
- **Bilinear**: $\text{score}(a, b) = a^\intercal W_1 b$
- **Dot-Product**: $\text{score}(a, b) = a^\intercal b$

where W_1 and W_2 are the weight matrices employed to compute the attention score.

2.3.2 Training Objectives

NMT networks are conditional language models trained to model the probability $\mathsf{P}(Y|X)$ of generating the sequence Y in the target language based on the given source sentence X. Given a parallel corpus $(\hat{X}, \hat{Y})$ of *N* training sentence pairs (where $X \in \hat{X}$ and $Y \in \hat{Y}$ are the source sentence and target sentence pair), the training objective can be formulated as minimizing the negative log-likelihood across the training corpus:

$$J((\hat{X}, \hat{Y}), \theta) = -\frac{1}{N} \sum_{i}^{N} \log \mathsf{P}(\hat{Y}^i | \hat{X}^i; \theta) \tag{2.9}$$

where $\theta = \{\theta_e, \theta_d\}$ is the set of trainable model parameters. θ_e and θ_d are the parameters of the encoder and decoder subnetworks, respectively.

2.3.3 Network Architectures

A number of deep neural architectures have been proposed for the NMT task. These models can be categorised into three main groups; RNN (LSTM/GRU) based models (Luong et al., 2015b; Bahdanau et al., 2015; Britz et al., 2017), convolutional network (CNN) based (Gehring et al., 2017; Kaiser et al., 2018) and self-attention networks (SAN) such as the Transformer network (Vaswani et al., 2017). Recent architectures also proposes a

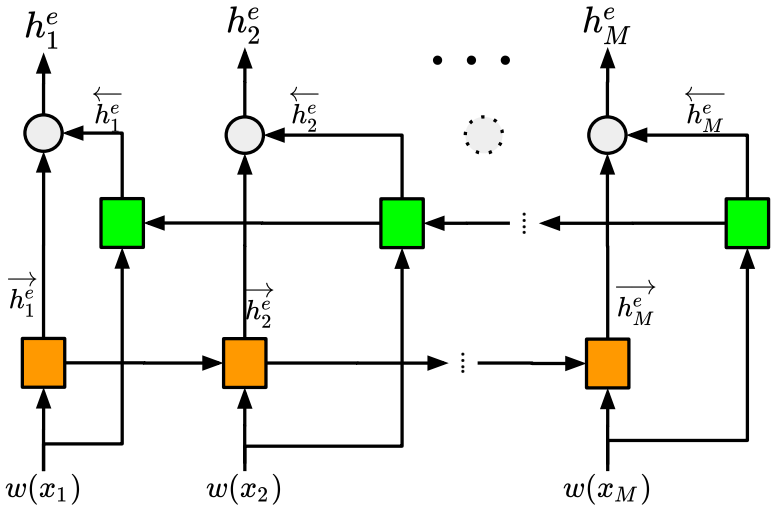


Figure 2.5: Architecture of the Bi-directional RNN sequentially processing the source tokens in both the forward (denoted by the orange nodes) and backward (denoted by the light green nodes) directions. $w(\cdot)$ is the embedding lookup operator employed to generate the word embedding for the input tokens. $H^e = [h_1^e, h_2^e, \cdots, h_M^e]$ is the output representation of the source sequence.

combination of these baseline models. For example, (Chen et al., 2018, 2019) explored the combination of RNN and self-attention to achieve a near state-of-the-art performance on NMT.

All experiments conducted in this thesis are performed on the recently proposed Transformer network (Vaswani et al., 2017). However, the approaches introduced are model agnostic hence can be applied to any generic encoder-decoder framework. This section discusses the major building blocks of the RNN and Transformer based NMT models.

RNN based models

To date, the majority of NMT models (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015b; Wu et al., 2016) have RNN as the backbone architecture. Both the encoder and decoder subnetworks are implemented using RNN and its variants such as GRU (Cho et al., 2014; Chung et al., 2015) and LSTM (Hochreiter and Schmidhuber, 1997). The encoder

is usually implemented with bi-directional RNNs (Bi-RNN/ Bi-LSTM/ Bi-GRU). The bi-directional RNN consists of two independent RNNs, one (the *Forward RNN*) processing the input sequence from left to right and the other (the *Backward RNN*) to generate the sentence representation from right to left. As shown in Figure 2.5, the input to the Bi-RNN is the word embeddings of the source tokens generated via the embedding lookup $w(\cdot)$. The Bi-RNN network generates the source representation $H^e = [h_1^e, h_2^e, \cdots, h_M^e]$, where each $h_i^e = [\overrightarrow{h_i^e}; \overleftarrow{h_i^e}]$. $\{\overrightarrow{h_i^e}, \overleftarrow{h_i^e}\}$ are the hidden states of x_i generated by the forward and backward RNN respectively.

The decoder is usually a unidirectional RNN generating the target sequence based on the source sentence hidden representation. In non-attention based RNN models, the fixed-length vector $v = [\overrightarrow{h_M^e}; \overleftarrow{h_1^e}]$. However with an attention mechanism, the hidden states of all the source tokens are exposed to the decoder subnetwork, as shown in Figure 2.4.

Transformer model

Similar to RNN (LSTM/GRU) (Bahdanau et al., 2015; Cho et al., 2014) and CNN (Gehring et al., 2017) based NMT models, the Transformer network (Vaswani et al., 2017) is based on the Encoder-Decoder architectural structure. As mentioned in Section 2.2.2, RNN based models to some extent can leverage contextual information across sequences of arbitrary length. Figure 2.6 illustrates the architectural structure of the Transformer network. As shown, the Transformer architecture does not employ RNN to learn the contextual information of the source and target sequences. Therefore, to model the contextual representation of an input sequence, the encoder and decoder subnetworks employ an attention mechanism and a feed-forward network across the multiple layers. Similar to conventional neural NLP models, the inputs to the encoder and decoder subnetworks are the continuous representations of the source sequence X and the target sequence $y_{<t}$ generated by the *Embedding* units.

The encoder subnetwork is composed of a stack of L layers. Each encoding layer consists of a self-attention sublayer and a position-wise feed-forward network sublayer ($\mathrm{FFN}(\cdot)$). Residual connection (He et al., 2016) and layer normalization layer ($\mathrm{LayerNorm}(\cdot)$) (Ba et al., 2016) are employed around each sublayer to ease training and further improve

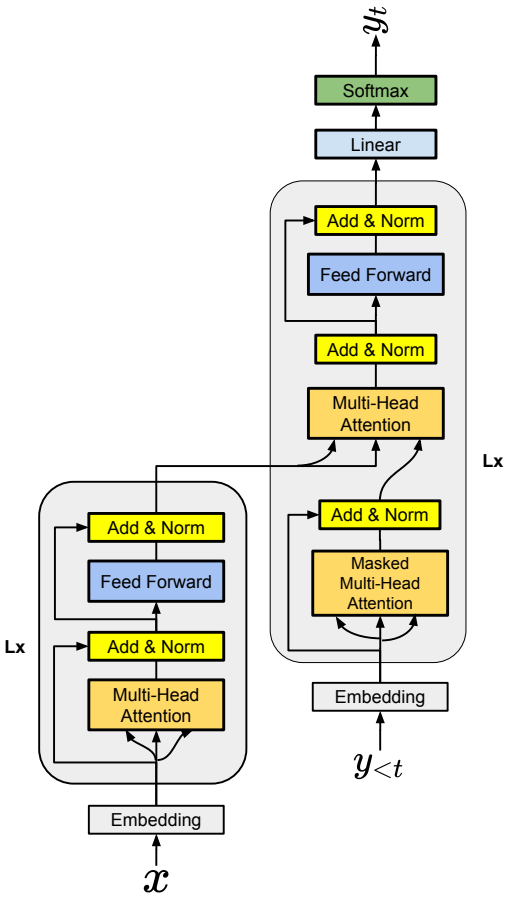


Figure 2.6: The architecture of the Transformer Network. Both the encoder and decoder consist of an identical stack of L layers.

performance. Formally, the output of each layer l ($H_e^l \in \mathbb{R}^{M \times d_{model}}$) is computed as:

$$\begin{aligned} S_e^l &= \text{LayerNorm}\left(\text{MHA}(H_e^{l-1}, H_e^{l-1}, H_e^{l-1}) + H_e^{l-1}\right) \\ H_e^l &= \text{LayerNorm}\left(\text{FFN}(S_e^l) + S_e^l\right) \end{aligned} \tag{2.10}$$

where $\text{MHA}(\cdot)$ refers to the multi-head attention unit employed by the self-attention sublayer. S_e^l is the output of the self-attention sublayer computed based on the H_e^{l-1} (source sentence representation of the preceding encoder layer $(l-1)$).

The decoder subnetwork is also a stack of L identical layers. However, unlike the encoder subnetwork, each decoding layer comprises three sublayers. Similar to the encoding

layer, it has multi-head self-attention and FFN sublayers but in between these sublayers is an encoder-decoder MHA sublayer. The encoder-decoder MHA sublayer is employed to perform multi-head attention computations over the output of the encoding subnetwork H_e^L. Specifically, the target representation ($H_d^l \in \mathbb{R}^{Z \times d_{model}}$) from each decoding layer l is computed as:

$$\begin{aligned} S_d^l &= \mathrm{LayerNorm}\left(\mathrm{MHA}(H_d^{l-1}, H_d^{l-1}, H_d^{l-1}) + H_d^{l-1}\right), \\ E_d^l &= \mathrm{LayerNorm}\left(\mathrm{MHA}(S_d^l, H_e^L, H_e^L) + S_d^l\right), \\ H_d^l &= \mathrm{LayerNorm}\left(\mathrm{FFN}(E_d^l) + E_d^l\right) \end{aligned} \tag{2.11}$$

where $S_d^l \in \mathbb{R}^{Z \times d_{model}}$ is the output of the self attention sublayer generated from the target representation of the preceding decoder layer $(l-1)$, H_d^{l-1}. $E_d^l \in \mathbb{R}^{Z \times d_{model}}$ is the output of the multi-head attention generated based on $S_d^l \in \mathbb{R}^{Z \times d_{model}}$ and $H_e^L \in \mathbb{R}^{M \times d_{model}}$.

A linear transformation layer with a softmax activation is employed to convert the output representation H_d^L from the final decoder layer into output probability distributions over the target vocabulary. Recent works (Pappas et al., 2018; Inan et al., 2016) propose sharing the same weights between the word embedding layer of the decoder subnetwork and the linear transformation layer to further improve the model's performance. This strategy reduces the size of the model in terms of the number of trainable parameters. Therefore, the models presented in this thesis are trained using this strategy.

Embedding Module

As shown in Figure 2.6, the Transformer network employs an *Embedding* module to generate the continuous representations of the source and target tokens. The *Embedding* module consists of two sub-units, namely an embedding lookup table and a *position embedding* unit. The embedding lookup table maps a given token to its corresponding continuous vector. The *position embedding* unit is a very important component employed to encode the positions of tokens (word order information) in the input sequence. This is because, unlike RNNs, the Transformer network cannot directly capture the position information of the input tokens (Vaswani et al., 2017). The embedding of a token u ($e_u \in \mathbb{R}^{d_{model}}$) in the input sequence s is computed as:

$$e_u = w(u) + \mathrm{PE}(s_u, f)$$

where $w(\cdot)$ is the conventional embedding lookup table, $\mathrm{PE}(\cdot)$ is the *position embedding* unit and s_u is the index or position of token u in the given sequence s. f is a dimensional parameter employed PE.

The position embedding unit can be fixed or learnable (Gehring et al., 2017). Experimental analysis performed by Vaswani et al. (2017) showed that both the fixed and learnable variants achieve comparable performance. However, the fixed variant is usually preferred as it introduces no new trainable parameters into the model. Therefore following (Vaswani et al., 2017), in this thesis the position embedding of the k^{th} input token is calculated as:

$$\mathrm{PE}(k, 2i) = \sin\left(\frac{k}{1000^{2i/d_{model}}}\right)$$
$$\mathrm{PE}(k, 2i+1) = \cos\left(\frac{k}{1000^{2i/d_{model}}}\right)$$

Fully connected Feed-forward network (FFN)

The FFN, as shown in Figure 2.6, is employed as the final sublayer of each layer within the encoder and decoder subnetworks. The network structure of the FFN comprises two linear transformation layers with ReLu activation across the output of the first layer. This structure is identical to a stack of two convolutions with a filter or kernel size of 1:

$$\mathrm{FFN}(x) = OW_o + b_o$$
$$O = \mathrm{ReLu}(xW_x + b_x)$$

where $\{W_x \in \mathbb{R}^{d_{model} \times d_{ff}}, b_x \in \mathbb{R}^{d_{ff}}\}$ and $\{W_o \in \mathbb{R}^{d_{ff} \times d_{model}}, b_o \in \mathbb{R}^{d_{model}}\}$ are the trainable parameters of the first layer and the second layer respectively.

Multi-Head Attention (MHA)

As mentioned in Section 2.3.1, the neural attention mechanism is a crucial component in the Seq2Seq architecture for many sequence generation problems including NMT (He et al., 2018; Bahdanau et al., 2015), paraphrase generation (Ampomah et al., 2019b; Hasan et al., 2016) and document summarization (Song et al., 2019b; Al-Sabahi et al., 2018). The Transformer model uses the Scale dot-product attention scoring function which takes three vectors as input, namely the query Q, value V and key K. It maps a given query and key-value pairs to an output which is the weighted sum of the values. The attention weights

indicate the correlation between each query and key. This attention is shown as follows:

$$\begin{aligned} \text{Attention}(Q, K, V) &= \text{softmax}(\alpha)V \\ \alpha &= \text{score}\,(Q, K) \\ \text{score}(Q, K) &= \frac{Q \times K^{\intercal}}{\sqrt{d_k}} \end{aligned} \tag{2.12}$$

where $K \in \mathbb{R}^{J \times d_k}$ is the key, $V \in \mathbb{R}^{J \times d_v}$ is the value and $Q \in \mathbb{R}^{Z \times d_k}$ is the query. Z and J are the lengths of the sequences represented by the Q and (V, K), respectively. d_k and d_v are the dimensions of the key and value vectors, respectively. The dimension of the query is also set as d_k to allow for the dot-product operation between Q and K. The division of $Q \times K^{\intercal}$ by $\sqrt{d_k}$ is done to scale the result of the product operation hence stabilizing the computation (Vaswani et al., 2017). Applying the softmax$(\cdot)$ operator to the attention score $\alpha \in \mathbb{R}^{Z \times J}$ produces the attention weight distribution for the context generation.

For better performance, the Transformer architecture employs a multi-head attention (MHA) composing of N_h (number of attention heads) scaled dot-product attention operations. Given the Q, K, and V, the multi-head attention computations are performed as follows:

$$\begin{aligned} MHA(Q, K, V) &= O \\ O &= \hat{C}W_o \\ \hat{C} &= \text{Concat}\,(head_1, head_2, \cdots, head_{N_h}) \\ head_h &= \text{Attention}(QW_h^Q, KW_h^K, VW_h^V) \end{aligned} \tag{2.13}$$

where $\{QW_h^Q, KW_h^K, VW_h^V\}$ are projections of the query, key and value vectors respectively for the h^{th} head. The projections are performed with the matrices $W_h^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_h^K \in \mathbb{R}^{d_{model} \times d_k}$ and $W_h^V \in \mathbb{R}^{d_{model} \times d_v}$. The inputs to the $MHA(\cdot)$ are $K \in \mathbb{R}^{J \times d_{model}}$, $V \in \mathbb{R}^{J \times d_{model}}$ and $Q \in \mathbb{R}^{Z \times d_{model}}$. $head_h \in \mathbb{R}^{Z \times d_v}$ is the result of the scaled dot-product operation for the h^{th} head. The N_h scaled dot-product operations are combined by the concatenation function $\text{Concat}(\cdot)$ to generate $H \in \mathbb{R}^{Z \times (N_h \cdot d_v)}$. Finally, the output $O \in \mathbb{R}^{Z \times d_{model}}$ is generated from the projection of $\hat{C}$ using the weight matrix $W_o \in \mathbb{R}^{(N_h \cdot d_v) \times d_{model}}$. The multi-head attention has the same number of trainable parameters as the vanilla (single-head) attention when $d_k = d_v = d_{model}/N_h$.

2.4 Data Preparation and Preprocessing

The early stages of solving NLP problems include data collection and preprocessing. Data collection mainly involves curating the training corpus tailored to the problem at hand. Unlike sentence classification problems such as Sentiment Analysis (Kathuria et al., 2019) and Topic Modelling (Song et al., 2019a), the NMT task requires a parallel corpus consisting of sentence pairs in the source and target languages. Deep NMT is a data-intensive task and, as such, requires a larger amount of "clean" sentence pairs to achieve higher translation performance. The preprocessing steps include activities such as cleaning the data to remove noisy information such as incomplete translation pairs, unknown symbols, and numbers.

The data cleaning step is followed by the creation of the vocabulary for each language under consideration. The training corpus typically contains millions of words, therefore training the NMT model with a vocabulary containing all the words in the training corpus is unfeasible. This is because larger vocabulary increases the model size causing an increase in the overall computational complexity of the model, especially in terms of the training speed and the memory or space requirement. A conventional solution is to limit the vocabulary to a fixed size of words selected based on the corresponding word frequencies in the training dataset. Rare words or words not in the vocabulary are represented with the "<UNK>" token. Despite the solution presented, this approach makes the translation model less robust to words not seen in the training corpus. For example, the model can generate target translations containing only <UNK> tokens. The above phenomenon is referred to as the *Out-Of-Vocabulary* (OOV) problem.

Recent works (Sennrich et al., 2016; Kudo, 2018) suggest using subword level information rather than training the model based on the word-level vocabulary. For all experiments presented in this thesis, the sentences are tokenized by breaking the words into subword units using byte-pair encoding (BPE) (Sennrich et al., 2016). Under BPE, the vocabulary is created from the training corpus as follows:

1. Initialise the vocabulary with all characters in the training data.
2. Split each word into a sequence of characters.
3. Iteratively compute the frequency of character pairs within the words in the corpus.
4. Merge and add the most frequent character n-gram pairs to the subword vocabulary.

5. Repeat step (4) until the desired vocabulary size is achieved.

Subword level information presents a form of *open vocabulary* to improve the generalization performance of the NMT model on unseen words in the training datasets (Sennrich et al., 2016). This is because a word not in the training corpus can be segmented into a sequence of character n-grams present in the subword vocabulary. Following common practice, a shared subword vocabulary for both the target and source sentences is employed to train our NMT models. For language pairs (such as German-English and Spanish-English) sharing a common alphabet, learning the BPE operation jointly on both the source and target languages improves the consistency of word segmentation. Besides, this reduces the problem of character deletion or insertion, which usually occurs during transliteration of names (Sennrich et al., 2016).

2.5 Inference Algorithms

The NMT network is trained to maximise the likelihood $\mathsf{P}(Y|X)$. The probability distribution for each target token is generated, as shown in Equation (2.7). Given a target vocabulary V_T of size $|V_T|$, the number of probable translations of the source sequence grows exponentially with the length of the target sequence Z (i.e. $|V_T|^Z$). Due to the size of $|V_T|^Z$, search algorithms such as greedy search and beam search are usually employed to approximate $\mathsf{P}(Y|X)$. The goal of these algorithms is to find the target translation in the search space of size $|V_T|^Z$ that maximizes the conditional probability:

$$\arg\max_{y} \mathsf{P}(y_1, y_2, \cdots, y_Z|X)$$

Greedy Search

During the target generation, greedy search approximates the best translation by considering only the tokens in V_T with the highest probability at each decoding step t. This search algorithm is computationally inexpensive in terms of both time and memory space required. However, the greedy search is sub-optimal, often producing low-quality translations.

Beam Search

Unlike greedy search, beam search expands and considers all possible choices for the next target token. During the search, the algorithm keeps track of the B most likely token sequences. B is termed as the beam size and is a parameter that determines the number of candidate tokens with the highest probability to consider at each decoding step. The greedy search algorithm is a variant of beam search with $B = 1$. The value of B has a greater impact on the translation performance of the NMT model. To achieve higher performance, state-of-the-art models generally employ larger values of B to generate the sentence translation. However, beam search is a computationally expensive technique. This is because the algorithm evaluates $B \times V_T$ probable token sequences at each step which requires a lot of memory. At each decoding step t, the algorithm computes the score $\mathsf{P}(y_t^f|X)$ for each possible path f (of length t) in the search space of size $B \times V_T$:

$$\mathsf{P}(y_t^f|X, y_{<t}^f) = \mathsf{P}(y_1^f|X)\mathsf{P}(y_2^f|X)\cdots\mathsf{P}(y_t^f|X) = \prod_{i=1}^{t}\mathsf{P}(y_i^f|X)$$

where y^f is the target sequence for the f^{th} search path (hypothesis). Based on the scores $\mathsf{P}(y_t^f|X)$, the algorithm selects the top B hypotheses for the next generation step $t+1$. The stopping criterion of the search algorithm is generally indicated by the generation of the <EOS>[2] token for all the B best paths or when the maximum target length Z is reached. Among the top B hypotheses, the target translation is the hypothesis with the highest score. That is, the search algorithm selects the hypothesis that maximizes the likelihood, $\mathsf{P}(Y|X)$. This can be expressed as:

$$\arg\max_{y}\prod_{t=1}^{Z}\mathsf{P}(y_t|X)$$

Since the probabilities are usually small numbers, the multiplication operation can result in numerical underflow of the floating-point numbers (Wu et al., 2016). As a remedy, the natural logarithm of the probabilities are multiplied:

$$\arg\max_{y}\prod_{t=1}^{Z}\log\mathsf{P}(y_t|X) = \arg\max_{y}\sum_{t=1}^{Z}\log\mathsf{P}(y_t|X)$$

[2]A special token denoting end of the sequence.

A well-known weakness of the vanilla beam search algorithm is, it tends to favour shorter sentences over longer sentences. This problem can be attributed to how the score for each probable sentence is obtained from the multiplication of token probabilities (addition of negative log-probabilities). Longer sentences usually have lower scores than shorter sentences. Generating shorter sentences than the desired target length can result in low translation quality. To further improve the performance of the search algorithm, *Length Normalization* (Wu et al., 2016) is usually employed to reduce the penalties of generating longer translations. With *Length Normalization*, the search objective is reformulated as:

$$\arg\max_{y} \frac{1}{\mathrm{LN}(Z)} \sum_{t=1}^{Z} \log \mathsf{P}(y_t|X)$$

where $\mathrm{LN}(\cdot)$ is the length normalization function. The three most widely used normalization functions are:

$$\mathrm{LN}(Z) = Z^{\alpha}$$
$$\mathrm{LN}(Z) = (1+Z)^{\alpha}$$
$$\mathrm{LN}(Z) = \frac{(5+Z)^{\alpha}}{6^{\alpha}}$$

where α is termed as the length penalty.

2.6 Automatic Evaluation

Estimating the translation quality of a given NMT system is usually performed by comparing the candidate translations from the system to one or more reference translations. The automatic evaluation metrics are designed to achieve a reasonable correlation with a human-based judgement of the translation quality of the candidate translations. Over the years, several automatic evaluation metrics have been explored, including the BiLingual Evaluation Understudy (BLEU) (Papineni et al., 2002), Translation Edit Rate (TER) (Snover et al., 2006), and Metric for Evaluation of Translation with Explicit ORdering (METEOR) (Banerjee and Lavie, 2005). These metrics primarily differ by how they handle the token synonyms and the order of tokens within the system translations and the reference

translations. In this thesis, the performance of the proposed models is evaluated based on the BLEU score metric.

BiLingual Evaluation Understudy (BLEU) (Papineni et al., 2002) designed by IBM scores the performance of NMT models based on the percentage of exact n-gram overlaps between the reference translations and systems translations. The score is calculated from the geometric mean of the n-gram precisions computed based on different n-gram sequence lengths. The final score is obtained by scaling the geometric mean by a brevity penalty (BP). The BP is calculated from the comparison of the length of the candidate translations (T) and the reference translations (R):

$$BP = \min\left(1.0, e^{1-\text{len}(R)/\text{len}(T)}\right)$$

where $\text{len}(R)$ and $\text{len}(T)$ are the lengths of the reference and the candidate translations respectively. It is noteworthy that BP is equal to 1.0 when $\text{len}(R) \leq \text{len}(T)$. The BP is employed to penalise models for generating shorter translations. Formally, the BLEU score of n-gram sequences up to the length *N* for any given candidate translations and reference translations is calculated as:

$$BLEU = BP \cdot \left(\prod_{n=1}^{N} \frac{\text{n-grams}(R \cap T)}{\text{n-grams}(T)}\right) \tag{2.14}$$

The BLEU score is on a scale of 0 to 100, and the higher the score, the better the model performance. Despite the simplicity and popularity of BLEU, it has some weaknesses. The order of the matching n-grams is not considered hence it does penalise models for n-gram scrambling. Furthermore, synonyms of words are ignored, resulting in the unfortunate penalisation of translations of similar in meaning.

2.7 Related Works

This section presents an overview of existing and related neural approaches to machine translation. These related approaches are categorised under *Multi-Task Learning* (MTL), *Deep Representational Learning* and *Contextual Modelling*.

2.7.1 Multi-Task Learning (MTL) for NMT

Following the work by Caruana (1998), multi-task learning (MTL) has been extensively applied to NLP problems including sequence tagging (Peng and Dredze, 2017; Rei, 2017; Ampomah et al., 2019a), document summarization (Li et al., 2018), and NMT (Luong et al., 2015a; Dong et al., 2015; Anastasopoulos and Chiang, 2018). Some of the earlier works on MTL application to NMT were Luong et al. (2015a); Dong et al. (2015). In (Luong et al., 2015a), the authors examine three MTL settings for Seq2Seq, where encoder and decoder subnetworks are shared between several different sequence generation tasks. To perform multi-lingual translation from a single source language, (Dong et al., 2015) connected multiple decoders to a single encoding subnetwork. Each decoder is trained to generate a target translation in a different language based on the common source representation from the top-level layer of the encoding subnetwork. To exploit a large amount of target-side monolingual data, (Domhan and Hieber, 2017) applied MTL to the decoder subnetwork to jointly learn the bilingual language translation task along with the target-side language modelling task. While the proposed approach by Domhan and Hieber (2017) seeks to strengthen the decoder subnetwork with the target-side monolingual dataset, (Zoph and Knight, 2016) investigates MTL for Seq2Seq learning with source-side monolingual data to further improve the performance of the encoder subnetwork. Motivated by the findings of (Luong et al., 2015a; Niehues and Cho, 2017; Baniata et al., 2018; Nadejde et al., 2017) introduce linguistic knowledge to enhance the translation quality of the NMT model by jointly training an MTL architecture to simultaneously learn target language translation along with linguistic tasks such as CCG supertags, chunking, and POS tagging. To leverage target-side syntactic structures, the SD-NMT model (Wu et al., 2017) jointly learns both the target word sequence generation and its corresponding dependency tree structure. Specifically, two decoding RNNs were connected to the encoder subnetwork, with one generating the target translation and the other constructing the dependency parse tree for the target sentence. For the task of cross-lingual information retrieval (search query translation), (Sarwar et al., 2019) argues that NMT systems usually fail to perform well due to the vocabulary mismatch between the vocabulary employed to train the NMT and the vocabulary distribution of the documents in the retrieval corpus. In response, an MTL-based NMT model is trained with a Relevance-based Auxiliary Task (RAT). The RAT is employed to make the NMT aware of the vocabulary of the retrieval corpus.

In (Tu et al., 2017), an auxiliary decoder is connected on top of the main decoding subnetwork for the task of source sentence reconstruction. This MTL strategy enhances translation performance by encouraging the main decoder to exploit the complete source representation along with learning the target sentence translation. The experimental results proved that exploiting the complete source information improves the adequacy of the output translations. Similarly, the approach proposed by Zhou et al. (2019) employs two decoders connected to a single encoding subnetwork to improve the robustness of the Transformer network to noisy source sentences. The two decoders are trained with two different learning objectives. Specifically, the first decoder is employed as a denoising unit generating clean source sentences. On the other hand, the target translation task is done by the second decoder based on the hidden representation of the noisy source sentence from the encoder and the clean sentence representation from the denoising decoder.

In Chapter 4, the proposed *Multi-level Supervision* (MS) schemes employ multiple decoders connected to a single encoder subnetwork similar to MTL architectures proposed by Dong et al. (2015); Zoph and Knight (2016) for the task of NMT. In contrast, all decoders employed in our multi-level supervision network generate target translations in the same language based on source representations from different encoder layers. The encoder subnetwork receives gradient signals from all connected decoders. Therefore, the lower-levels are encouraged to capture the necessary source semantic information to support the target generation from each connected decoder. This presents a form of regularisation effect to further improve the source representation ability of the encoder subnetwork. Furthermore, unlike conventional multi-task NMT models (Niehues and Cho, 2017; Baniata et al., 2018; Luong et al., 2015a; Malaviya et al., 2017), all the decoders connected to the encoding subnetwork are trained jointly end-to-end on the same target language generation task based on source representation captured by the corresponding or connected encoding layers.

2.7.2 Deep Representation Learning

The *Joint Attention Strategies* and the MS schemes presented in Chapters 3 and 4 respectively are motivated by research and advances in deep representation learning. Effective propagation of gradient information across the multiple layers of a neural network can significantly improve its performance at learning a given task (Srivastava et al., 2015; He et al.,

2016; Huang et al., 2017). To achieve this, several techniques, including residual connections (He et al., 2016), highway network connections (Srivastava et al., 2015), and dense connections (Huang et al., 2017), have been extensively explored in areas such as computer vision and NLP. These approaches improve the propagation of features and error information across the multiple layers of the neural network via direct information paths between the layers. The simplicity and effectiveness of these skip-connection techniques allow for easy integration and have become the standard for state-of-the-art models for learning problems employing neural networks. With respect to NMT, models such as the self-attention based Transformer model (Vaswani et al., 2017), CNN based ConvS2S (Gehring et al., 2017) and LSTM/GRU based model (Wu et al., 2016) achieved state-of-the-art performance by employing residual connections between the layers. As noted by Irie et al. (2019) and Vaswani et al. (2017), the performance of the Transformer model significantly degrades when trained without residual connections between the multiple sublayers. Across these models, source representations from the lower-level encoding layers are not considered during the target generation as only the top-level encoder layer's output is passed to the decoder subnetwork.

Making use of source representations from multiple encoding layers can improve the generalization performance of deep NMT models. Each layer captures a different level of abstraction of the source representation (Raganato et al., 2018; Belinkov et al., 2017; Peters et al., 2018). For example, analysing a multi-layer LSTM based network, (Peters et al., 2018) shows that the LSTM states within the lower-level layers are useful for syntax-based tasks such as POS while the states across the top-level layers capture the semantic meaning of the input sentence and its constituent tokens, making them useful for NLP tasks such as word sense disambiguation. Similar observations were made by Raganato et al. (2018) and Belinkov et al. (2017). The findings of these works motivate the idea of exploiting the representations captured by the layers within a multi-layer neural network to solve a given NLP task. A linear combination approach was proposed by Peters et al. (2018) to generate a single sentence representation based on the representations aggregated from the multiple layers. To learn better source representation, (Wang et al., 2018a) presents three information fusion techniques to combine representations from multiple encoding layers via a single information fusion layer. Similarly, (Dou et al., 2018) explored different representation aggregation approaches to combine source features generated from different

encoder layers. Besides, they proposed training the NMT model with a diversity promoting auxiliary learning objective to ensure that the layers encode diverse source information. The static layer aggregation approaches from (Dou et al., 2018; Wang et al., 2018a; Peters et al., 2018) (such as the linear feature combination method) as argued by Dou et al. (2019) sometimes ignores useful contextual information that can improve performance. In response, they propose dynamic layer aggregation with routing-by-agreement mechanisms where each decoding layer receives a different aggregation of source representations from each of the encoding layers. Similarly, Bapna et al. (2018) proposed *Transparent Attention Mechanism* where different joint source representation is generated for each decoding layer. Specifically, for a model with M encoding and N decoding layers, N different joint source representations are generated (one for each decoding layer) from the weighted combination of outputs from all the encoding layers, including the word embedding layer. Via the *Transparent Attention Mechanism*, Bapna et al. (2018) were able to train (2-3x) deeper NMT models. The performance gain is attributed to the *Transparent Attention Mechanism* easing the optimisation of deeper models.

A common theme among these works is the generation of a single source feature representation as an aggregation of output representations from different encoder layers. The decoder performs the source-target attention based on the aggregated joint source representation. These approaches provide a simplistic mechanism to enhance the source-target attention mechanism while improving the flow of gradient information from the decoding subnetwork to the encoding layers. In contrast, the work presented in Chapter 3 hypothesises that providing the decoder network more direct access to representations from multiple encoding layers can further improve the performance of the model and further enhance gradient flow to each encoder layer, especially for deeper encoder networks. Specifically, we propose to perform the neural attention computations directly across source representations from different encoding layers via a *Multi-Layer Multi-Head Attention* module. In Chapter 3 both the *Layer Aggregation* strategies explored in (Dou et al., 2018; Wang et al., 2018a; Peters et al., 2018; Ampomah et al., 2019b) and the proposed *Multi-Layer Multi-Head Attention* are referred to as the *Joint Attention Strategies* (JASs). For each of these approaches, a joint source-target attention computation is performed directly (*Multi-Layer Multi-Head Attention*) and indirectly (*Layer Aggregation*) across the output representations from multiple encoding layers.

The primary goal of the JASs is to enhance the performance of the connected decoder subnetwork by exposing source representations from multiple encoding layers. However, the performance of an encoder-decoder architecture does not only rely on the generation ability of the decoding unit but also the ability of the encoder to learn the best source representation. The MS schemes presented in Chapter 4 seek to improve the performance of the encoder subnetwork by connecting auxiliary decoders to some of the lower-level layers, which are then trained along with the decoder subnetwork connected to the top-level layer. These decoders are trained to learn the generation of the same target sequence. The proposed MS forces the encoder to learn the necessary source semantic information to support the target generation from all the connected decoders.

2.7.3 Contextual Modelling

Exploiting the useful contextual information has been shown to be critical to achieving higher translation performance in machine translation models (Gimpel and Smith, 2008; He et al., 2008; Marton and Resnik, 2008; Luong et al., 2015b). During the target generation, contextual information is required by the translation model to effectively perform tasks such as the phrase and word sense disambiguation (Wu et al., 2014; Marvin and Koehn, 2018). To leverage the rich source side information, (Gimpel and Smith, 2008) proposed augmenting the phrase-based SMT model with contextual features of the phrases to be translated. They employed contextual features such as parts-of-speech (POS) tags, local syntactic features, and lexical features based on nearby words. The work by He et al. (2008) argues that a weakness of conventional syntax-based SMT models (Chiang, 2005; Huang et al., 2006) is their failure to distinguish and exploit the different structures of the source sentence. In response, they proposed maximum entropy-based rule selection (*MaxEnt RS*) model combined with rich context features such as POS, lexical features, and length information. The *MaxEnt RS* allows the decoder to leverage the source contextual information required to efficiently select the best translation rules. Similarly, (Marton and Resnik, 2008) proposed to guide the rule selection process with automatically induced syntactic features. In (Wu et al., 2014), the quality of the bilingual word embedding for SMT is improved by exploiting discrete contextual information such as the position and POS features.

One of the earlier works leveraging both the local and global contextual information

to improving the performance of NMT was (Luong et al., 2015b). Specifically, to utilise the local and global information of the source tokens, the authors employ two attention mechanisms, namely, the global and local attention units. During the decoding step t, the global attention considers all the source tokens while the local attention computation is performed across a subset of the source tokens. The performance improvement highlights the importance of capturing both global and local contextual information. Motivated by this, recent works (Yang et al., 2018; Xu et al., 2019; Sperber et al., 2018) investigate strategies to learning both the long-distance and short-range dependencies between the tokens to further enhancing the sentence representational ability of the self-attention mechanism. To model localness, (Yang et al., 2018) proposed a modification of the self-attention mechanism with a learnable Gaussian bias which specifies the central position and window of tokens neighbourhood to pay more attention to. Similarly, (Sperber et al., 2018) explored two masking techniques for controlling the contextual range of self-attention for the task of acoustic modelling. Other works (Wu et al., 2018; Yang et al., 2019b) explored convolutional concepts to restricting the attention scope or span to a window of neighbouring tokens.

The ability of self-attention to effectively capture the global contextual information has been identified as one of its salient strengths improving the performance downstream NLP tasks such as semantic modelling (Yang et al., 2019a), and constituency Parsing (Kitaev and Klein, 2018). However, the approaches by Wu et al. (2018); Yang et al. (2019b) restricting the attention scope to some degree can result in loss of important global and long-distance dependencies (Xu et al., 2019). In (Xu et al., 2019), the authors proposed a hybrid attention mechanism to learn the local contextual information without restricting the self-attention mechanism's ability to model the global and long-distance dependencies. The *QANet* proposed by Yu et al. (2018a) employs a multi-layer depth-wise separable CNN to initially model the local contextual representation before the application of the self-attention unit. The resulting model significantly outperformed RNN based models for the task of machine comprehension. In a similar direction, Chapter 5 proposes the *Dual Contextual* (DC) module to leverage both the local and global information to further improve the performance of NMT. However unlike Xu et al. (2019); Shen et al. (2018); Yu et al. (2018a), the *DC* module employs two separate attention units to generate the sentence representations. Furthermore,

this thesis argues that the decoder subnetwork requires rich contextual source representation to achieve higher translation quality. This implies that the output of the encoder subnetwork has to "fully" encapsulate both the local and global contextual information of the source tokens. Therefore, the modelling of the local and global contextual information is applied to all layers within the encoder and decoder subnetworks.

2.8 Summary

This chapter presented an overview of the basic neural LM as well as the Seq2Seq application to Neural Machine Translation (NMT). Furthermore, a review of existing related works is also provided. The remainder of this thesis presents the contributions to achieving our aim to study and design neural architectures to perform sequence generation effectively. The next chapter presents the JASs to exploit the source representations generated by multiple layers of the encoder subnetwork.

Part II

Leveraging Multiple Source Representations

Chapter 3

JASs: Joint Attention Strategies for Neural Machine Translation

The encoder-decoder network frameworks employed for the task of Neural Machine Translation (NMT) usually consist of a stack of multiple layers. Each encoding layer learns a different level of abstraction of the input source sequence. However, only the source representation from the top-level encoder layer is leveraged by the decoder subnetwork during the generation of the target sequence. These models do not fully exploit the useful source representations learned by the lower-level encoder layers. Leveraging these source representations has the potential to further enhance the translation performance of the NMT model. Inspired by recent advances in deep representation learning, this chapter addresses the goal of improving the performance of the decoding subnetwork by exploiting the source linguistic information captured across multiple encoding layers. Two *Joint Attention Strategies* (JASs) are explored:

- ***Layer Aggregation***: the decoder generates the target sequence based on a joint source representation computed as the combination of source representations from multiple encoding layers. This provides the decoder subnetwork indirect access to the layers of the encoder subnetwork
- ***Multi-Layer Attention***: provides the decoder subnetwork more direct access to the multiple encoder layers via a *Multi-Layer Multi-Head Attention* module to improve the translation performance.

This chapter is based on the work presented in (Ampomah et al., 2019a, 2020).

3.1 Introduction

The neural approaches to solving sequence to sequence (Seq2Seq) problems including Neural Machine Translation (NMT) (Luong et al., 2015b; Vaswani et al., 2017; Gehring et al., 2017), Document Summarization (Al-Sabahi et al., 2018; Song et al., 2019b) and Paraphrase Generation (Hasan et al., 2016; Ampomah et al., 2019b) have achieved significant improvement over traditional machine learning approaches (Och et al., 1999; Callison-Burch et al., 2011; Celikyilmaz and Hakkani-Tur, 2010; Litvak and Last, 2008) without the need for extensive feature engineering. The backbone of these neural architectures is the encoder-decoder framework. The task of the encoding subnetwork is the generation of the semantic information from the source sequence. On the other hand, the decoder is charged with the target sequence generation based on the source semantic representation captured by the encoder.

The basic Seq2Seq model first encodes an input sentence of arbitrary length into a fixed-length hidden representation which is then processed by the decoder in order to generate a satisfactory target sentence. As observed by Bahdanau et al. (2015), learning the fixed-length hidden representation creates a bottleneck during the decoding phase. The local contextual information gets diluted as information flows through the encoder network resulting in the loss of information and consequently poor target sentence generation. To improve the performance, the attention neural mechanism (Bahdanau et al., 2015; Luong et al., 2015b) was introduced to compute the alignment between the encoder network's output and the current decoding step. That is, instead of learning a fixed-length vector, the contextual information for every source token is kept and later referenced by the decoder. During decoding, this mechanism makes it possible for the decoder to peek through the encoder to utilise the local and global contextual information for better mapping of the input sequence to the output sequence. Recent state-of-the-art NMT models (Vaswani et al., 2017; Gehring et al., 2017) implement each of the encoder and decoder subnetworks as a stack of multiple layers. The propagation of information between the two subnetworks becomes difficult as the number of layers increases. To minimise this problem, these models use shortcut connections such as residual units (He et al., 2016) between the layers to

enhance the flow of information across the multiple layers.

Leveraging source representations from multiple layers can further improve the Seq2Seq generation task. However, current NMT models generate the target sequence based on representation from only the final encoding layer. These models fail to fully explore the potentially useful representations generated by the lower-level encoder layers during the target generation. A problem with this approach is that there is no guarantee that the necessary source information required by the decoder subnetwork is encoded in the final encoder layer. Each layer within the encoding sub-network learns a particular set of features, which are then passed to the upper layers for further processing and feature extraction (Raganato et al., 2018; Belinkov et al., 2017; Peters et al., 2018). These layers extracts different levels of abstraction of the source representation. For example, Belinkov et al. (2017) evaluated source representations from different encoder layers on NLP tasks such as part-of-speech tagging (POS) and semantic tagging. They argue the lower-level encoder layers tend to focus more on learning word-level information or properties while the higher-level layers encode more semantic information. Therefore it seems reasonable to leverage source representations captured across the multiple layers within the encoding subnetwork. Furthermore, research works from the field of computer vision (Yu et al., 2018b; Huang et al., 2017) have proven the benefits and the performance impact of exploiting representations from multiple top-level and lower-level layers.

This chapter investigates the *Joint Attention Strategies* (JASs) to exploit the source representations extracted from multiple encoding layers. Specifically, during the target generation, the techniques presented generate the source-target contextual information based on n source representations obtained from the encoding subnetwork. The n source representations are aggregated by a *Source Feature Collector* module based on the outputs from the top-n encoding layers. Two JASs are explored to better exploit the source representations collected by the *Source Feature Collector* module. These are the *Layer Aggregation* (Dou et al., 2018; Wang et al., 2018a; Peters et al., 2018; Ampomah et al., 2019b) and *Multi-Layer Attention* (Ampomah et al., 2020) strategies.

For the *Layer Aggregation* approach, each decoding layer receives a joint source representation generated from the combination of the n representations returned by the *Source Feature Collector* module. The decoding subnetwork performs the neural attention computation across the generated joint source representation providing indirect access to multiple

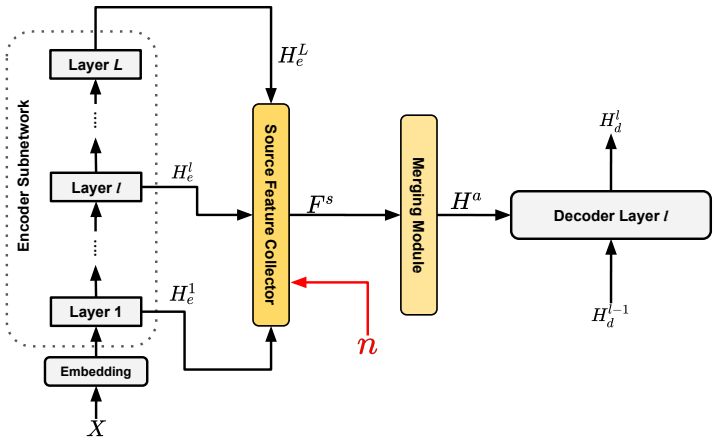
encoding layers. Even though the *Layer Aggregation* provides a simple way to leverage the multiple source representations, the work presented in this chapter argues that allowing the decoder more direct access to the encoding layers can significantly improve the flow of gradient information and enhance the overall translation performance of the model.

Under the *Multi-Layer Attention* approach, a more direct access to the encoding layers is provided by performing attention computations directly across the outputs from the top-n encoding layers. The *Multi-Layer Attention* is performed via a *Multi-Layer Multi-Head Attention* (MLMHA) module, which allows each decoding layer to directly interact with different levels of abstraction of the source sequence to further improve the translation quality. For a deeper encoder subnetwork, this also enhances the propagation of gradient information between the encoder-decoder subnetworks as each encoder layer receives error signals directly from all the decoding layers.

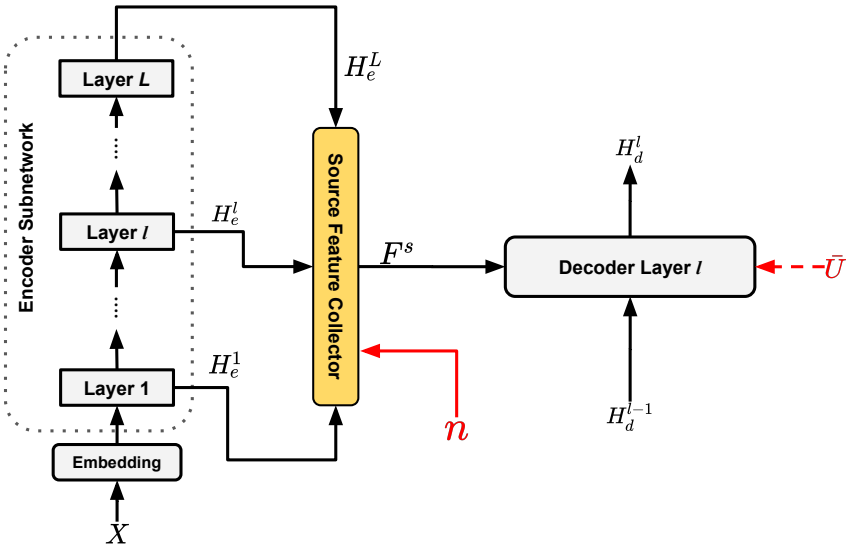
All experiments and analyses conducted in this chapter are based on a current state-of-the-art model, namely the Transformer architecture (Vaswani et al., 2017). Experimental results on two IWSLT language translation tasks (Spanish-English and English-Vietnamese) and WMT'14 English-German translation demonstrate the effectiveness of allowing each decoding layer access to representations from multiple encoding layers. The main contributions of this chapter are:

- Proposing the *Multi-Layer Multi-Head Attention* (MLMHA) module which allows the decoding layers to directly exploit source representations captured by multiple encoding layers.
- Demonstrating consistent improvement over models exploiting only the source representation from the top-level encoder layer.
- Providing analysis on the encoding subnetwork to understand the impact of exposing all encoder layers to the decoder subnetwork.
- Providing analysis on the impact of varying the number of encoder layers outputs (n) that are considered by the *Source Feature Collector*.

The remainder of the chapter is organised as follows: The next section presents the JASs explored. Section 3.3 presents the experiments conducted on the datasets under consideration. Furthermore, the experimental results are compared and discussed in Section 3.4.



(a) Layer Aggregation Approach: Each decoding layer l performs the source-target neural attention mechanism across a joint source representation H^a generated via a *Merging Module*.



(b) Multi-Layer Attention: Each decoding layer l performs the source-target neural attention computation directly across multiple encoding layers. $\bar{U} = [\bar{U}_0, \bar{U}_1]$ (where $\bar{U}_i \in \{0, 1\}$) controls how alignment computations are performed across the source representations in F^s.

Figure 3.1: Illustration of the *Joint Attention Strategies* to exploiting source representations from multiple encoding layers. F^s is a list of source sentence representations obtained by the *Source Feature Collector* module from the encoding layers. n is the number source representations exposed to the decoder subnetwork. X is the input sequence. H_d^{l-1} and H_d^l denotes the output representations from the decoding layers $l-1$ and l, respectively.

Section 3.5 presents the extensive analyses performed to investigate the impact of exploiting multiple source representations from the encoding subnetwork via the JASs. Finally, the conclusion is presented in Section 3.6.

3.2 JASs: Joint Attention Strategies

In a conventional encoder-decoder architecture (Vaswani et al., 2017; Gehring et al., 2017; Bahdanau et al., 2015) with L number encoding layers, only the representation of the source sequence[1] from the final encoding layer ($H_e^L \in \mathbb{R}^{J \times d_{model}}$) is passed to the decoding subnetwork during the target sequence generation. As the depth of the network increases, it becomes difficult to efficiently and effectively train the model due to vanishing and exploding gradients.

The encoder employs the entire stack of layers to learn the source semantic information. For a deeper network, there is no guarantee that the last encoder layer's output is the best representation for the target generation (Wang et al., 2018a; Dou et al., 2018; Ampomah et al., 2019b). To enhance the flow of information between the encoder and decoder subnetwork during the forward and backward propagation, two approaches to exploiting the source representations learned by multiple layers in the encoding subnetwork are investigated in this chapter. Besides, the proposed strategies allow the model to fully exploit the structural composition of the source sentence.

To this end, a list of source representations $F^s = [f^1, f^2, \cdots, f^{n-1}, f^n]$ is obtained by a *Source Feature Collector* module from the encoding subnetwork as shown in Figure 3.1. Specifically, the *Source Feature Collector* returns a list of source representations F^s aggregated from outputs of the top n encoding layers[2]. It is noteworthy that F^s contains source representations from all layers in a L-layer encoder subnetwork when $n = L$. The Seq2Seq models (Vaswani et al., 2017; Bahdanau et al., 2015; Gehring et al., 2017) using only the top-level encoder output H_e^L corresponds to setting $n = 1$. To exploit the multiple source representation F^s, two JASs are explored in this chapter. The first is the *Layer Aggregation* strategy which generates a joint source representation $H^a \in \mathbb{R}^{J \times d_{model}}$ as a combination of features in F^s via a *Merging Module* as displayed in Figure 3.1a. The

[1]The representation from an encoder layer consists of the hidden states of all source tokens.

[2]n is considered as a hyperparameter in this work.

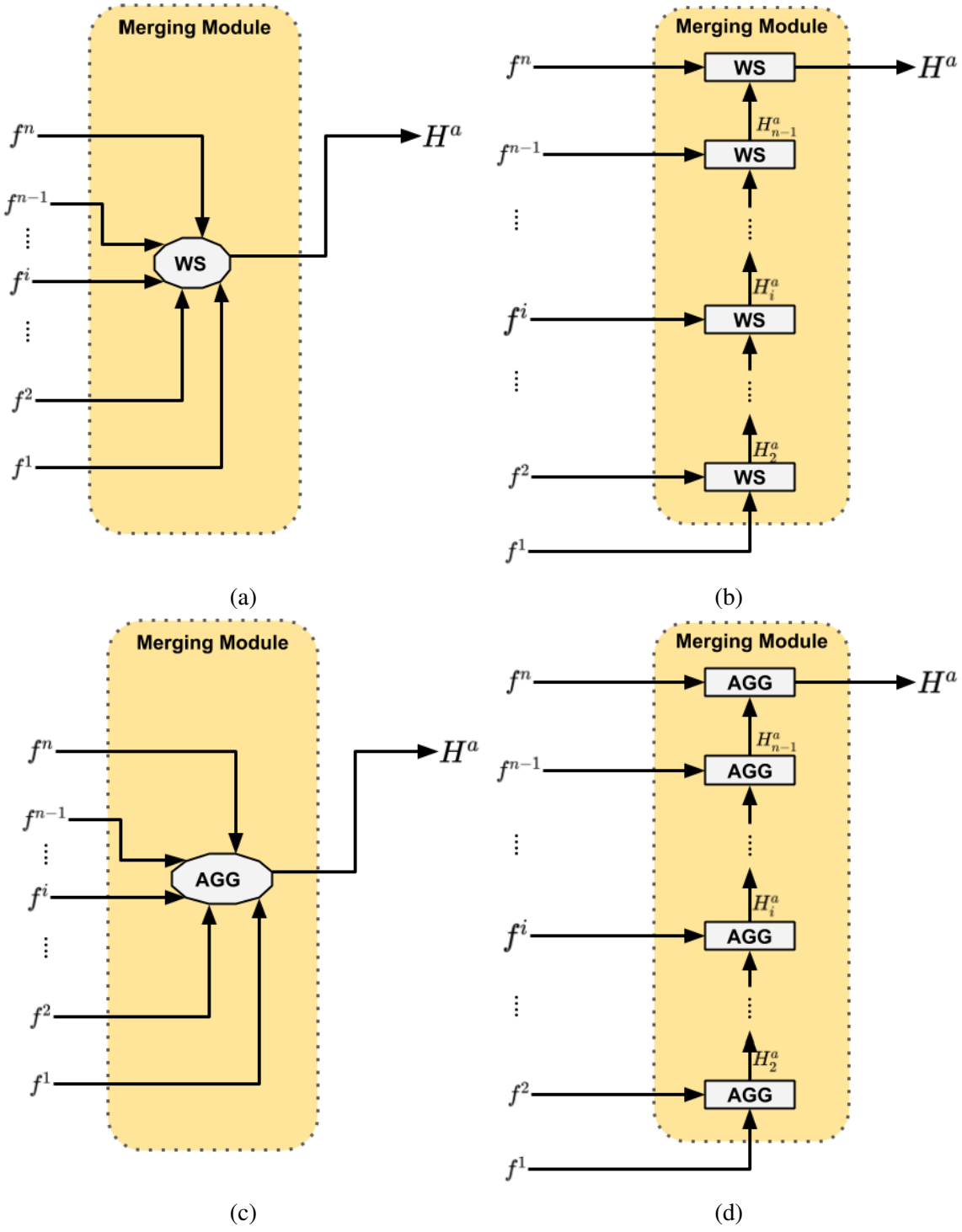


Figure 3.2: Feature aggregation strategies employed by the *Merging Module* to generates the joint source representation H^a from representations in $F^s = [f^1, f^2, \cdots, f^{n-1}, f^n]$. (a), (b), (c) and (d) illustrate the Linear Feature Summation, Iterative Feature Summation, Linear Feature Concatenation and Iterative Feature Concatenation strategies, respectively. The WS and AGG are the units employed to combine the input features in F^s.

combination functions explored in this work are *Feature Summation* based (Linear Feature Summation and Iterative Feature Summation) and *Feature Concatenation* based (Linear Feature Concatenation and Iterative Feature Concatenation). These are illustrated in Figure 3.2. Each decoding layer receives the H^a instead of $H_e^L \in \mathbb{R}^{J \times d_{model}}$ during the target generation. The second technique is the *Multi-Layer Attention*. Here, each decoding layer l receives the list source representations F^s as shown in Figure 3.1b. A *Multi-Layer Multi-Head Attention* (MLMHA) module is employed within each decoding layer to perform the neural attention computation directly and jointly across all source representations in F^s. Unlike the *Layer Aggregation* strategies, this approach provides more direct access to the representations learned by each encoding layer.

3.2.1 Layer Aggregation

Feature Summation

Feature Summation is the simplest approach to combining multiple source features. A joint source representation H^a is computed by the *Merging Module* as the weighted summation of source representations in F^s via a *Weighted Summation* (WS) unit, as shown in Figures 3.2a and 3.2b. Given the input representations $r = [r^1, r^2, \cdots, r^i, \cdots, r^b]$, the WS generates the H^a as the weighted summation of elements in r:

$$\begin{aligned} H^a &= \mathrm{WS}(r) \\ \mathrm{WS}(r) &= \sum_i^b W^i r^i \end{aligned} \tag{3.1}$$

where $W^i \in \mathbb{R}^{d_{model} \times d_{model}}$ is a weight parameter used to control the contribution of the i^{th} representation $r^i \in \mathbb{R}^{J \times d_{model}}$. Two feature summation approaches are explored in this work, namely, the Linear Feature Summation and Iterative Feature Summation. As shown in Figure 3.2a, the Linear Feature Summation employs a single WS unit to combine all the features in F^s. That is each f^i corresponds to the WS's input representation r^i.

Alternatively, multiple WS units can be used to combine the source representations F^s in an iterative fashion, as shown in Figure 3.2b. This approach is termed as Iterative Feature Summation. As shown, WS takes as input the two representations f^i and H_{i-1}^a (the output from the previous WS unit in the stack). Given $r = [f^i, H_{i-1}^a]$, the output after the Iterative

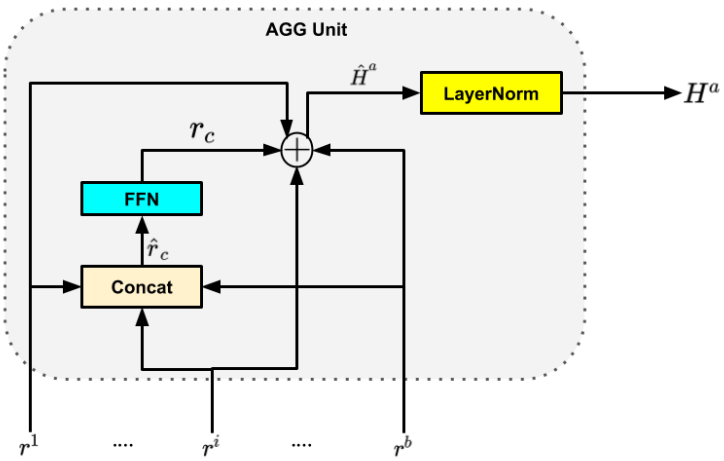


Figure 3.3: Illustration of the aggregation unit (AGG) generating the joint representation H^a based on the input representations or features $r = [r^1, \cdots, r^i, \cdots, r^b]$

Feature Summation step i, H^a_i can be formulated as:

$$H^a_i = \mathrm{WS}([f^i, H^a_{i-1}]) \tag{3.2}$$

It should be noted that under Iterative Feature Summation, the $H^a_1 = f^1$ and $H^a = H^a_b$.

Feature Concatenation

As noted by Dou et al. (2018) and Wang et al. (2018a), the *Feature Summation* approaches employing weights $[W^1, W^2, W^{\cdots}, W^b]$ usually tend to ignore important source contextual information that could further enhance the performance of the model. Furthermore, the summation can prevent the efficient flow of gradient information across the network, as argued by Huang et al. (2017). Therefore, to preserve much of the source contexts captured by encoding layers, two *Feature Concatenation* approaches (the Linear Feature Concatenation and Iterative Feature Concatenation) are explored. Specifically, the H^a is generated based on the concatenation of the source representations in F^s. As displayed in Figure 3.2c and Figure 3.2d, the *Merging Module* employs the aggregation unit (AGG) to perform the feature combination. Given an input list of features (of length *b*), $r = [r^1, r^2, \cdots, r^i, \cdots, r^b]$, the AGG unit generates the H^a as illustrated in Figure 3.3. As displayed, position-wise

feed-forward network (FFN) processes the concatenation of the input representations $\hat{r}_c$ producing the hidden representation r_c; this is followed by residual connections from input representations in r. Finally, the output H^a is generated by the $\mathrm{LayerNorm}$. H^a is computed as:

$$\begin{aligned} H^a &= \mathrm{AGG}\left(r\right) \\ \mathrm{AGG}\left(r\right) &= \mathrm{LayerNorm}(\hat{H}^a) \\ \hat{H}^a &= r_c + \sum_{i=1}^{b} r^i \\ r_c &= \mathrm{FFN}\left(\mathrm{Concat}\left(r^1, r^2, \cdots, r^b\right)\right) \end{aligned} \tag{3.3}$$

Unlike Dou et al. (2018), the FFN employed by the AGG is a two-layer feed-forward network with ReLu activation in between the layers. The Linear Feature Concatenation (shown in Figure 3.2c) uses a single AGG unit to combine the representations in F^s and each r^i corresponds to f^i:

$$H^a = \mathrm{AGG}\left([f^1, f^2, \cdots, f^n]\right) \tag{3.4}$$

The Iterative Feature Concatenation approach employs multiple AGG units to combine source representations in an iterative fashion. Similar to the Iterative Feature Summation approach, each AGG unit takes as input two representations f^i and H^a_{i-1} (the output from the previous AGG unit combining representations from the lower-level of source abstraction) as shown in Figure 3.2d. The Iterative Feature Concatenation is formulated as:

$$H^a_i = \mathrm{AGG}\left([f^i, H^a_{i-1}]\right) \tag{3.5}$$

where $H^a_1 = f^1$ and $H^a = H^a_b$. The Iterative Feature Concatenation and Iterative Feature Summation approaches are motivated by the iterative stacking employed by the Transformer model.

3.2.2 Multi-Layer Attention

The goal of the *Multi-Layer Attention* is to allow the decoder subnetwork more direct access to multiple encoding layers to further enhance the translation performance of the

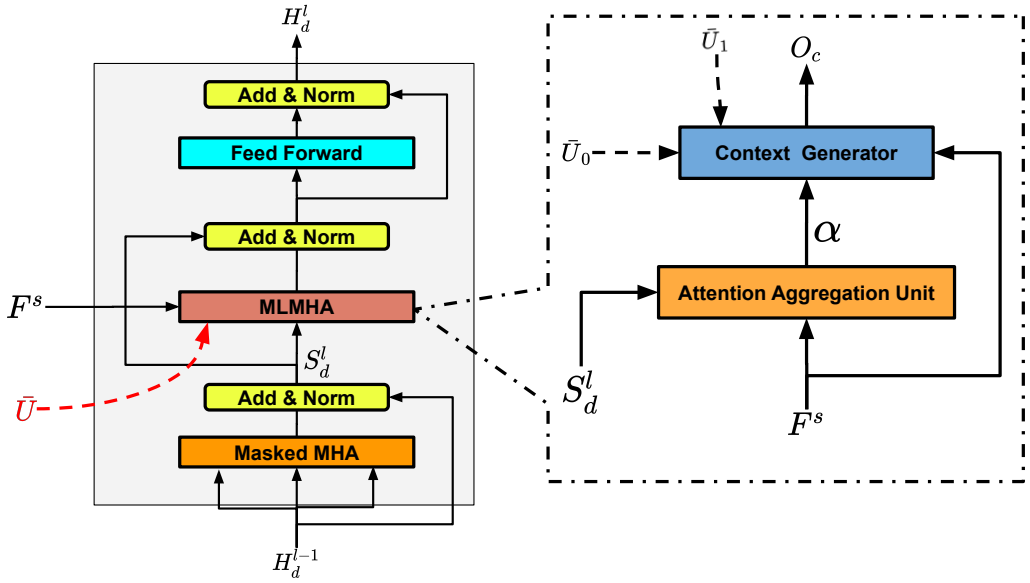


Figure 3.4: Illustration of a decoding layer with *Multi-Layer Multi-Head Attention* (MLMHA) sublayer to perform the attention computation across multiple features F^s received from the encoding stack. $\alpha = [\alpha^1, \alpha^2, \cdots, \alpha^n]$ is the list of attention weights (where α^i corresponds to attention weight with respect to f^i in F^s), and O_c is the joint context vector across all features in F^s.

model. To this end, each decoding layer receives a list of source sentence representations $F^s = [f^1, f^2, \cdots, f^n]$ aggregated by the *Source Feature Collector* module as shown in Figure 3.1.

In addition to F^s, each decoder layer receives a binary vector $\bar{U} = [\bar{U}_0, \bar{U}_1]$, where $\bar{U}_i \in \{0, 1\}$. $\bar{U}$ controls how the attention computation is performed across the multiple encoder representations in F^s. Specifically, the values of $\bar{U}_0$ and $\bar{U}_1$ determine the strategy employed to generate the contextual representation base on all the source representations in F^s. Formally, the encoder-decoder multi-head attention (encoder-decoder MHA) sublayer is extended to consider the multiple source representations F^s. To this end, the encoder-decoder MHA sublayer is replaced with a *Multi-Layer Multi-Head Attention* (MLMHA)

module, as illustrated in Figure 3.4. The computations across each decoder layer (see Equation (2.11)) are re-formulated as follows:

$$\begin{aligned} S_d^l &= \text{LayerNorm}\left(MHA(H_d^{l-1}, H_d^{l-1}, H_d^{l-1}) + H_d^{l-1}\right), \\ E_d^l &= \text{LayerNorm}\left(\text{MLMHA}(S_d^l, F^s, \bar{U}) + S_d^l\right), \\ H_d^l &= \text{LayerNorm}\left(\text{FFN}(E_d^l) + E_d^l\right) \end{aligned} \tag{3.6}$$

Multi-Layer Multi-Head Attention (MLMHA)

MLMHA employs two sub-modules, namely the *Attention Aggregation Unit* and the *Context Generator*, to perform the attention computations across all representations in F^s as shown in Figure 3.4. The *Attention Aggregation Unit* outputs a list of attention weights $\alpha = [\alpha^1, \alpha^2, \cdots, \alpha^n]$, where α^i is the multi-head attention weight with respect to f^i. Specifically, $\alpha^i \in \mathbb{R}^{N_h \times Z \times J}$ is calculated as:

$$\begin{aligned} \alpha^i &= \text{Concat}\left(\alpha_1^i, \alpha_2^i, \cdots, \alpha_{N_h}^i\right) \\ \alpha_h^i &= \text{score}\left(QW_h^q, KW_h^k\right) \end{aligned} \tag{3.7}$$

where $\alpha_h^i \in \mathbb{R}^{Z \times J}$ is the attention score with respect to the attention head h based on f^i. QW_h^q, and KW_h^k are, respectively, the projections of the query (S_d^l) and key (f^i) vectors for the h^{th} attention head. The projections are performed with the matrices $W_h^q \in \mathbb{R}^{d_{model} \times d_k}$, $W_h^k \in \mathbb{R}^{d_{model} \times d_k}$. $\text{score}(\cdot)$ is the attention score function defined in Equation (2.12).

Based on the α and F^s, the *Context Generator* computes the joint contextual representation O_c. The operation of the *Context Generator* is controlled by the values of $\bar{U}_0$ and $\bar{U}_1$. To be specific, $\bar{U}_0$ controls the generation of the context vector c^i with respect to f^i. Depending on the value of $\bar{U}_0$, the MLMHA module computes the c^i using either a *representation-specific-attention weight* or a *joint-attention weight*. For the case of $\bar{U}_0 = 1$, the contextual vector for the h^{th} attention head with respect to f^i, $c_h^i \in \mathbb{R}^{Z \times d_k}$ is calculated using the *representation-specific-attention weight* $\text{softmax}(\alpha_h^i)$:

$$c_h^i = \text{softmax}(\alpha_h^i) \cdot VW_h^v \tag{3.8}$$

where VW_h^v is the transformation of the value vector (f^i) with the projection weight $W_h^v \in \mathbb{R}^{d_{model} \times d_k}$. In contrast, for the case of $\bar{U}_0 = 0$, a *joint-attention weight* $\hat{\alpha} \in \mathbb{R}^{N_h \times Z \times J}$ is

employed to obtain c_h^i for each f^i and attention head *h*. $\hat{\alpha}$ is computed as:

$$\hat{\alpha} = \text{softmax}(\sum_{i=1}^{n} \alpha^i)$$

Analogous to Equation (3.8), c_h^i is generated with $\hat{\alpha}$ as:

$$c_h^i = \hat{\alpha}_h \cdot VW_h^v \tag{3.9}$$

where $\hat{\alpha}_h \in \mathbb{R}^{Z \times J}$ is the *joint-attention weight* with respect to the attention head h. In summary, the c^i with respect to f^i is calculated as:

$$\begin{aligned} c^i &= \text{Concat}\left(c_1^i, c_2^i, \cdots, c_{N_h}^i\right) \\ c_h^i &= \begin{cases} \text{softmax}(\sum_{i=1}^{n} \alpha^i)_h \cdot VW_h^v & \bar{U}_0 = 0 \\ \text{softmax}(\alpha_h^i) \cdot VW_h^v & \bar{U}_0 = 1 \end{cases} \end{aligned} \tag{3.10}$$

As shown above, when $\bar{U}_0 = 0$, the c_h^i is generated with the *joint-attention weight* ($\hat{\alpha}_h$) with respect to each f^i. In contrast, c_h^i is computed with the *representation-specific-attention weight* (softmax(α_h^i)).

Given the contexts $C = [c^1, c^2, \cdots, c^n]$ computed across F^s, a joint context O_c is generated as a combination of all vectors in C. The choice of contexts combination function (either *contexts-summation* or *contexts-concatenation*) is determined by the $\bar{U}_1$. When $\bar{U}_1 = 1$, O_c is generated via the summation of all the contexts representations (*contexts-summation*) in C. However for $\bar{U}_1 = 0$, O_c is obtained from the concatenation of all the contexts (*contexts-concatenation*) in C. The O_c is formulated as:

$$\begin{aligned} O_c &= \hat{C} W_o \\ \hat{C} &= \begin{cases} \text{Concat}\,(c^1, c^2, \cdots, c^n) & \bar{U}_1 = 0 \\ \sum_{i=1}^{n} c^i & \bar{U}_1 = 1 \end{cases} \end{aligned} \tag{3.11}$$

where $W_o \in \mathbb{R}^{d_c \times d_{model}}$ is the projection matrix for transforming the intermediate context representation $\hat{C} \in \mathbb{R}^{Z \times d_c}$ into $O_c \in \mathbb{R}^{Z \times d_{model}}$. It is noteworthy that the dimension size d_c is equal to $n \cdot d_{model}$ when *contexts-concatenation* is employed. However when O_c is

Models	$\bar{U}_0$	$\bar{U}_1$	c_h^i	$\hat{C}$
M-00	0	0	$\text{softmax}(\sum_{i=1}^{n} \alpha^i)_h \cdot VW_h^v$	$\text{Concat}\,(c^1, c^2, \cdots, c^n)$
M-01	0	1	$\text{softmax}(\sum_{i=1}^{n} \alpha^i)_h \cdot VW_h^v$	$\sum_{i=1}^{n} c^i$
M-10	1	0	$\text{softmax}(\alpha_h^i) \cdot VW_h^v$	$\text{Concat}\,(c^1, c^2, \cdots, c^n)$
M-11	1	1	$\text{softmax}(\alpha_h^i) \cdot VW_h^v$	$\sum_{i=1}^{n} c^i$

Table 3.1: Configurations of the *MLMHA* as determined by the values of $\bar{U}_0$ and $\bar{U}_1$. c_h^i is the context vector for the attention head *h* with respect to f^i and $\hat{C}$ is the overall context vector across the n source representations in F^s.

generated via *contexts-summation* ($\bar{U}_1 = 1$), d_c is equal to d_{model}. In summary, the value of the binary vector $\bar{U} = [\bar{U}_0, \bar{U}_1]$ (where $\bar{U}_i \in \{0, 1\}$) presents four possible configurations of the MLMHA module in the decoding layer. For simplicity, the model *M-*ij denotes the configuration where $\bar{U}_0 = i$ and $\bar{U}_1 = j$ as summarized in Table 3.1. As shown, the *M-10* and *M-11* models compute the context c_h^i using the *representation-specific-attention weight* while the *joint-attention weight* is employed by the *M-00* and *M-01* models. The *contexts-summation* approach is employed by the M-01 and M-11 models to output contextual representation O_c. In contrast, for the M-00 and M-10 models, the *contexts-concatenation* approach is employed.

3.3 Experimental Setup

3.3.1 Datasets

The effectiveness of the JASs explored in this chapter are evaluated on the following language translation tasks: WMT'14 English-German (En→De), Spanish-English (Es→En), and English-Vietnamese (En→Vi).

For the En→De task, the models are trained on the widely-available WMT'14 dataset comprising about 4.56 million sentence pairs for training. Following (Dou et al., 2018; Gehring et al., 2017), the newstest2013 and newstest2014 are used as the validation and test

sets, respectively. For the Es→En task, the dataset employed is from the IWSLT 2014 evaluation campaign[3] (Cettolo et al., 2014). The training set consists of 183k training sentences pairs, and the tst 2014 split is used as the test set. The validation set consisting of about 5593 sentence pairs is created by concatenating dev2010, tst2010, tst2011, and tst2012 splits. The dataset for the En→Vi translation task is from the IWSLT 2015 English-Vietnamese track (Cettolo et al., 2015). The training set comprises 133k sentence pairs. The validation and test sets are from the TED tst 2012 (1553 sentences) and TED tst 2013 (1268 sentence pairs), respectively.

To alleviate the Out-of-Vocabulary (OOV) problem, a shared vocabulary[4] generated via byte-pair-encoding (BPE)[5] (Sennrich et al., 2016) is employed to encode the source and target sentences. In the case of the En→De translation task, the shared vocabulary comprises about 32k subword tokens. For the En→Vi and Es→En translation tasks, the shared vocabulary consists of 21k and 34k subword tokens, respectively.

3.3.2 Model Setup, Training and Inference

For experiments on the WMT'14 En→De, the base configuration of the the Transformer architecture (Vaswani et al., 2017) is employed due to the size of the dataset. Specifically, the hidden size, filter size, and the number of attention heads are 512, 2048, and 8, respectively. Both the encoder and decoder subnetworks have 6 layers. The experiments on the Es→En and En→Vi tasks are conducted based on the small configuration with the word embedding dimension, hidden state size, and the number of attention heads set as 256, 256, and 4, respectively. The position-wise FFN has a filter of a dimension of 1024. The regularization dropout rate is 0.1. A label smoothing (Vaswani et al., 2017; Pereyra et al., 2017) with 0.1 weight is applied to obtain the uniform prior distribution over the target vocabulary to further improve the translation quality. Finally, The models trained on each IWSLT task consists of a 4-layer encoder subnetwork and 4-layer decoder subnetwork. On each language translation task under consideration, the same model configuration (in terms of the number of layers and hidden dimensions) is employed to train the Transformer baseline model for a fair comparison.

[3] https://wit3.fbk.eu/mt.php?release=2014-01

[4] The original casing for the tokens in each sentence is preserved.

[5] https://github.com/rsennrich/subword-nmt

For experiments on each dataset, the value of the hyperparameter n for the *Source Feature Collector* is set to the number of layers present in the encoding subnetwork i.e. $n = L$. That is, on the WMT'14 En→De, and IWSLT tasks n is set as 6 and 4, respectively. As mentioned in Section 3.2.2, the model *M-*ij denotes the configuration where $\bar{U}_0 = i$ and $\bar{U}_1 = j$. Similarly, C-Agg, Iter-C-Agg, S-Agg and Iter-S-Agg, denote respectively, the models employing the Linear Feature Concatenation, Iterative Feature Concatenation, Linear Feature Summation, and Iterative Feature Summation approaches to combining the representations in F^s.

For the WMT'14 En→De task, the models are trained for 160k iterations with a batch size of 4960 tokens. Evaluations are performed every 10k iterations. On the IWSLT tasks (En→Vi and Es→En), all models are trained with a batch size of 2048 tokens for a total of 200k iterations with evaluations performed at every 4k iterations. The maximum sequence length is limited to 200 subword tokens for the En→De task and 150 subword tokens for the IWSLT tasks. The optimizer employed to train the models in this chapter is the Adam optimizer (Kingma and Ba, 2014) (with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^9$). Varying the value of the learning rate over the course of training has been shown to yield higher performance. The choice of learning rate scheduling algorithm has a high impact on the convergence of the learning model. Following (So et al., 2019), a preliminary experiment was performed using the En→Vi dataset to select the best scheduling algorithm among the *linear decay*, *single-cosine-cycle* (Loshchilov and Hutter, 2017) and the *noam* (Vaswani et al., 2017) given by:

$$lr_t = d_{model}^{-0.5} \cdot \min\left(t^{-0.5}, t^{-0.5} \cdot warmup\_steps\right)$$

where lr_t is the learning rate for the training step *t*. Using the performance on the En→Vi development set, we found both the Transformer baseline and our models performed well when trained with the *single-cosine-cycle* with warmup steps. Therefore all experiments are performed using the *single-cosine-cycle* scheduling algorithm.

During inference, the target sentences are generated by the models via beam search algorithm. On the En→De task, the beam size and the length penalty are 4 and 0.6, respectively. For the IWSLT translation tasks, a beam size of 6 and a length penalty of 1.1 is employed. Following common practice, the translation quality on the En→De, case-sensitive detokenized BLEU (Papineni et al., 2002) computed with mteval-v13a.pl[6] is employed as

[6]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/mteval-v13a.pl

Model	#Params (M)	Train	BLEU
8-Layer RNN (Wu et al., 2016)	-	-	26.30
ConvSeq2Seq (Gehring et al., 2017)	-	-	26.36
Transformer-Base (Vaswani et al., 2017)	-	-	27.31
Transformer+EM Routing (Dou et al., 2019)	-	-	28.81
Transformer+Layer Aggregation (Dou et al., 2018)	-	-	28.78
Our NMT Systems			
Transformer	61.2	3.66	28.37
With Layer Aggregation			
S-Agg	62.8	3.55	28.24(-0.13)
Iter-S-Agg	63.9	3.52	28.29(-0.08)
C-Agg	68.6	3.38	28.17(-0.20)
Iter-C-Agg	77.0	3.11	28.81(+0.44)†
With MLMHA			
M-00	92.7	2.66	28.80 ($+0.43$)†
M-01	84.9	2.74	28.54 ($+0.17$)
M-10	92.7	2.59	**29.08** ($+0.71$)‡
M-11	84.9	2.70	28.51 ($+0.14$)

Table 3.2: Evaluation of translation performance on the WMT'14 English-German (En→De). **#Params** and **Train** denotes the number of trainable model parameters and the training speed measured in terms of the steps per second, respectively. In parentheses are the progressive gain between JASS models and the reimplementation of the Transformer baseline. "†" and "‡" indicate statistically significant difference with $\rho < 0.05$ and $\rho < 0.01$, respectively.

the evaluation metric. For the Es→En, case-sensitive BLEU metric with multi-bleu.pl[7] is used for the evaluations. Finally, the translation quality for the En→Vi is reported based on the case-sensitive BLEU score computed with sacreBLEU[8][9]. The BLEU score measures the translation quality based on the exact matching between the system generated translations and reference translations by considering the n-gram overlaps. The higher the BLEU

[7]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl
[8]https://github.com/mjpost/sacrebleu
[9]with the signature BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.4.13

Models	BLEU Scores	
	En→Vi	**Es→En**
Transformer	30.58	39.80
With Layer Aggregation		
S-Agg	30.91 (+0.33)	39.92 (+0.12)
Iter-S-Agg	31.01 (+0.43)	39.77 (−0.03)
C-Agg	31.03 (+0.45)	39.71 (−0.09)
Iter-C-Agg	**31.13** (+0.55)	40.31 (+0.51)
With MLMHA		
M-00	30.88 (+0.30)	**40.99** (+1.19)
M-01	31.07 (+0.49)	40.61 (+0.81)
M-10	30.71 (+0.13)	40.57 (+0.77)
M-11	30.78 (+0.17)	40.55 (+0.75)
Existing NMT Systems		
Luong & Manning (Luong and Manning, 2015)	23.30	-
NPMT (Huang et al., 2018)	27.69	-
NPMT + LM (Huang et al., 2018)	28.07	-
CNN + Reinforcement learning (Edunov et al., 2018)	-	-
LSTM + Variational Attention (Deng et al., 2018)	-	-
Model-level dual learning (Xia et al., 2018)	-	-
Tied Transformer (Xia et al., 2019)	-	40.51
Layer-wise Coordination (He et al., 2018)	-	40.50
UEDIN (Cettolo et al., 2014)	-	37.29

Table 3.3: Evaluation of translation performance of the JASs models on the IWSLT En→Vi, and Es→En translation tasks.

score, the better the model. Compare-mt[10] (Neubig et al., 2019) is employed to evaluate and measure the statistical significance using paired bootstrap resampling (Koehn, 2004). The sample size is set as 1000 reference and model-generated sentence pairs. In this approach, 1000 pairs of (reference, hypothesis) are sampled with-replacement from the reference translations and our model-generated translations. The sampling was performed 1000 times resulting in 1000 different new test sets. Computing the BLEU score for each of the new test sets produces a distribution over the BLEU scores that can be employed as a good substitute for evaluating the performance of the models on a very large number of sentences that is more representative of all possible reference sentences. To quantify the significance

[10]https://github.com/neulab/compare-mt

of the difference in the performance between the baseline model and one of our models, the BLEU metric score is computed for each of these sampled test sets with respect to each model under consideration. Under this paired bootstrap resampling performance evaluation, the best model is the one that performed better across the trials with 1000 sampled test sets. Following (Koehn, 2004), we conclude that a model is better than the other with $\gamma\%$ statistical significance if it achieved higher performance $\gamma\%$ of the time. For simplicity, the statistical significance test is performed on the WMT'14 En→De task mainly due to the size of the test dataset.

3.4 Results

This section presents the performance evaluations of the strategies to leverage the multiple source representations presented in this chapter on the three language translation tasks. For each language pair under consideration, the performance obtained by each of the *M-ij* models and the *Layer Aggregation* models (S-Agg, C-Agg, Iter-S-Agg and Iter-C-Agg) is compared to results from existing NMT models. Tables 3.2 and 3.3 summarizes the translation performance on the WMT'14 En→De task and the IWSLT tasks (Es→En and En→Vi), respectively. In each table, the value in parentheses represents the performance gain over the Transformer baseline model (Vaswani et al., 2017) re-implemented in this chapter.

For the WMT'14 En→De task, the Iter-C-Agg model and all the configurations of the MLMHA outperform the standard Transformer model (Vaswani et al., 2017) as shown in Table 3.2. Furthermore, the M-10 model outperform all existing models. As shown, our reimplementation of the Transformer baseline outperforms the original model from (Vaswani et al., 2017). The S-Agg, Iter-S-Agg and the C-Agg models failed to match the performance of our reimplementation of the baseline model. Only the Iter-C-Agg model produced a statistically significant gain over the Transformer baseline among the *Layer Aggregation* approaches. For the Transformer models trained with MLMHA module, only the two *contexts-concatenation* based models (M-10 and M-00) achieved significant gains of $+0.71$ BLEU and $+0.43$ BLEU in the translation quality. In contrast, the performance gains of the *contexts-summation* models (M-10 and M-11) are statistically insignificant. Lower performance gains were obtained with the M-01 ($+0.17$ BLEU) and M-11 ($+0.14$

BLEU). Table 3.3 summarizes the performance gains of the joint attention models on the IWSLT tasks. For the Es→En translation task, only the Iter-C-Agg model and MLMHA based models further improved the performance of the Transformer baseline. Compared to the *Layer Aggregation* models, the *M-*ij models produced a higher improvement in the translation quality. On this dataset, the M-00 model achieved the overall best performance with a 40.99 BLUE ($+1.19$ increase in translation quality). Finally, On the En→Vi task, only the M-00, M-01, and the *Layer Aggregation* models produced a higher performance improvement in the BLEU score.

The translation results presented in Tables 3.2 and 3.3 demonstrate the potential performance gain of leveraging source representations from multiple encoding layers. However, the improvement in translation performance is shown to be dependent on the approach employed to leverage the multiple source representations. On the En→De and Es→En tasks, providing the decoder direct access to the multiple encoder layers via the MLMHA is shown to outperform (in most cases) the indirect access provided by the *Layer Aggregation* techniques. However, on the En→Vi dataset, only the M-00 and M-01 models achieved comparable performance to the *Layer Aggregation* models. On average, the M-11 achieved the overall worse performance among the *M-*ij models with the only higher gain achieved on the Es→En task. In contrast, the M-00 demonstrates a better generalization ability than M-11 as it consistently achieved higher translation performance gains across the different translation tasks. The translation performance can be attributed to the *joint-attention weight* and *contexts-concatenation* techniques employed by the M-00 model as shown Table 3.1. The *joint-attention weight* is collaboratively computed across the multiple encoder layers' outputs. Compared to employing the *representation-specific-attention weight*, generating the context representation c^i via this strategy enhances information sharing across the encoder layers, further improving the robustness of the NMT model. Unlike *contexts-summation* ($\bar{U}_1 = 1$), the *contexts-concatenation* technique preserves much of the contextual information required for the translation task (see Section 3.5.2). Finally, among the *Layer Aggregation* based models, the Iter-C-Agg produced consistent performance gain over the baseline model, demonstrating better generalization.

The performance gain via the MLMHA comes at a higher computational cost in terms of the number of parameters and training speed, as shown in Table 3.2. The *Layer Aggregation* approaches have a lower impact on the training speed. For example, the S-Agg and

Iter-S-Agg techniques degrade the speed by about 0.12 steps per second. The MLMHA introduces additional trainable parameters as each decoder layer employs[11] n different set of weights to compute the attention weights. The M-00 and M-10 models have larger number of parameters due to the *contexts-concatenation* strategy. Consequently, this decreases the training speed as more effort is required to optimise the parameters of the *M-*ij models effectively. Section 3.5.2 further explores the impact of n on the translation performance and training speed of the JASs based models.

3.5 Analysis

This section presents further analyses performed to better understand the impact of the JASs on the performance of the Transformer model. This includes analysis to understand the impact of:

- the source sentence length on the translation performance for the joint attention models.
- varying the number of source representations considered (the hyperparameter n from the *Source Feature Collector* module) on the performance of the joint attention models.
- exposing all the encoding layers to the decoding subnetwork on encoder subnetwork.

All the above analyses are performed on the En→De due to the size of the dataset as well as the number of layers employed to train the models.

3.5.1 Length of source sentence

Capturing effectively the contextual information, as well as the long-distance dependencies between the tokens of the source sentence, can further enhance the translation quality on longer sentences (Dou et al., 2018). Following (Luong et al., 2015b), sentences of similar lengths (in terms of the number of source tokens) are grouped. The choice of range for the grouping is based on the sentence lengths (the number of subword tokens in each source

[11] See

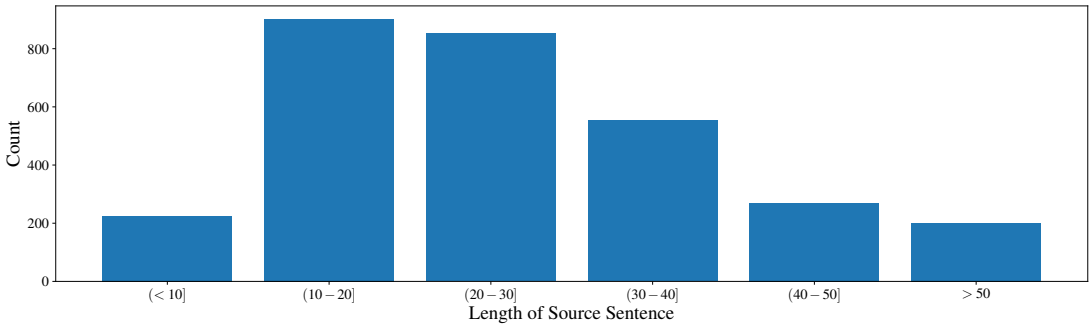


Figure 3.5: Distribution of the number of sentences from the WMT'14 En→De test set across the different sentence length groups.

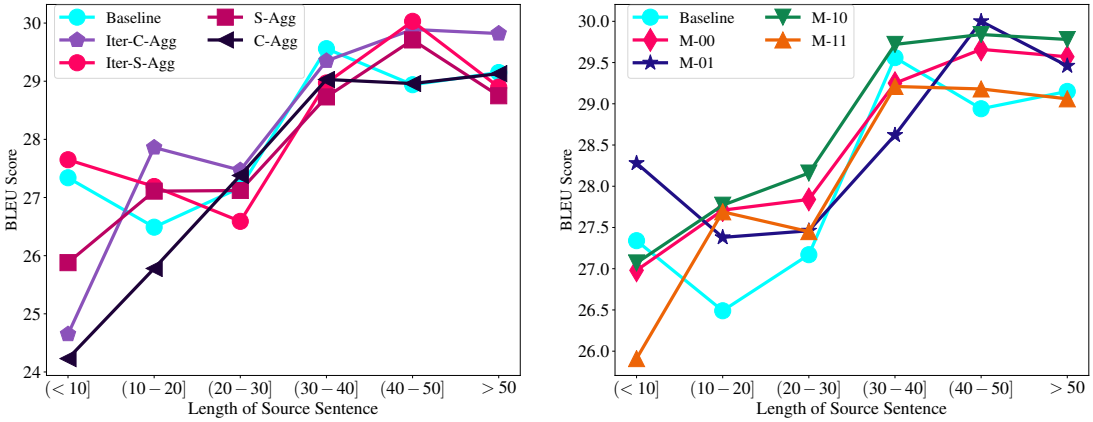


Figure 3.6: BLEU scores on the WMT'14 En→De test set for the Transformer baseline model, the *Layer Aggregation* based models, and the *M-ij* models with respect to the different source sentence lengths. **Left**: Transformer baseline vs *Layer Aggregation* models. **Right**: Transformer baseline vs *Multi-Layer Attention* (*M-ij*) models.

sentence) across the En→De test set. About 62% of the sentences (1.8k) have a sequence length of less than 31 subword tokens. Therefore, the comparison presented in this section is based on the following sentence length groups: <10, 10-20, 20-30, 30-40, 40-50, and >50. Figure 3.5 shows the distribution of the number of sentences across the different sentence length groups. For each group, the BLEU score is calculated for outputs from the models under consideration. As can be seen in the Figure 3.6, the performance of the baseline model (Transformer) generally improves with increasing input sentence lengths, especially for sentence lengths between 10 and 40 subword tokens. The Transformer model via the self-attention sublayers is able to model or capture the contextual information and

global dependencies between the tokens irrespective of their distances or locations within the input sentence.

Among the *Layer Aggregation* models, the Iter-C-Agg model is shown to consistently outperform the baseline across the sentences with length greater than 40. The variations across the groups clearly explains why the Iter-S-Agg, C-Agg, and S-Agg models performed poorly on the En→De as shown in Table 3.2. On average, the Iter-S-Agg and S-Agg models achieved higher translation quality on lengths less than 21 tokens. For sentence lengths greater than 21, these models achieved the worst performance among all the JASS models introduced in this chapter. On the other hand, the performance of the C-Agg and Iter-C-Agg models generally improved with increasing sentence length. Surprisingly, the C-Agg model achieved the worst performance for sequences with fewer than 20 tokens.

As shown in Figure 3.6, across the sentences with lengths greater than 10, some of the *M-*ij models generally outperform the baseline model. This is true especially in the case of the M-10 model. It achieves the overall best translation performance for sentences longer than 20 tokens. The performance of the M-10 and M-00 models improve consistently with increasing sentence length. The M-01 achieved the best translation quality on sentences with fewer than ten tokens. However, similar to the baseline, performance degrades for sentences with lengths between 10 and 20 before improving for a longer sentence. Besides, among the *M-*ij models, it has the overall worse performance on sentences with lengths between 10 and 40. The M-11 model, on the other hand, performed poorly on the shorter sentences (fewer than ten tokens) with the lowest BLEU score (25.91). This might explain the lower BLEU score of the *contexts-summation* based models (M-01 and M-11) as shown in Table 3.2.

Overall, the performance of the *M-*ij models and the *Layer Aggregation* models obtained across the different groups motivates the hypothesis that the JASs further improves the performance of the self-attention sublayers of the encoder at capturing effectively the global and long-range dependencies between tokens of the input sentence hence improving the translation performance even on longer sentences. Section 3.5.3 explores the impact of these JASs on the self-attention unit of each encoding layer.

Models		#Params (M)	Train	BLEU
Baseline	B0	61.2	3.65	28.37
	B1	86.5	2.95	28.49
	B2	90.7	2.60	28.59
S-Agg	n=2	61.8	3.64	28.57
	n=3	62.1	3.63	28.13
	n=4	62.3	3.63	28.02
	n=5	62.6	3.59	28.37
	n=6	62.8	3.57	28.24
Iter-S-Agg	n=2	61.8	3.63	28.57
	n=3	62.3	3.61	28.50
	n=4	62.8	3.61	28.48
	n=5	63.4	3.54	28.33
	n=6	63.4	3.52	28.29
C-Agg	n=2	64.4	3.56	28.60
	n=3	65.5	3.53	28.80
	n=4	66.5	3.49	28.76
	n=5	67.6	3.46	28.49
	n=6	68.6	3.41	28.15
Iter-C-Agg	n=2	64.4	3.56	28.60
	n=3	67.6	3.45	28.36
	n=4	70.7	3.33	28.81
	n=5	73.9	3.22	28.57
	n=6	77.0	3.11	28.82

(a)

Models		#Params (M)	Train	BLEU
M-00	n=2	67.5	3.41	28.43
	n=3	73.8	3.18	28.53
	n=4	80.2	3.03	28.72
	n=5	86.4	2.77	28.66
	n=6	92.7	2.65	28.80
M-01	n=2	66.0	3.45	28.82
	n=3	70.7	3.20	28.76
	n=4	75.4	3.06	28.66
	n=5	80.2	2.93	28.46
	n=6	84.9	2.74	28.54
M-10	n=2	67.5	3.39	28.42
	n=3	73.8	3.15	28.41
	n=4	80.2	3.01	28.59
	n=5	86.4	2.76	29.12
	n=6	92.7	2.59	29.08
M-11	n=2	66.0	3.44	28.72
	n=3	70.7	3.16	28.71
	n=4	75.4	3.01	28.60
	n=5	80.2	2.83	28.62
	n=6	84.9	2.70	28.51

(b)

Table 3.4: Impact of n (the number of encoding layers considered by the *Source Feature Collector* module) on the performance of the JASs based models. B0, B1 and B2 refers to the Transformer baseline model trained with different configurations in terms of the number of layers and the filter size FFN sublayer.

3.5.2 Impact of the hyperparameter *n*

As summarized by the Tables 3.2 and 3.3, the decoder subnetwork exploiting the multiple source representations extracted from different encoding layer (in most cases) significantly improves the performance of the NMT model. This section investigates the impact of varying the value of n (i.e. using only the representations from the top n encoding layers) from 2 to 6. As mentioned in Section 3.2, $n = 1$ corresponds to the Transformer baseline model,

which employs source representation from only the final encoder layer. Specifically, each model under consideration is trained with different values of n. It is noteworthy that when $n = 2$, the iterative feature combination approaches Iter-S-Agg and Iter-C-Agg become the same as the corresponding linear approach S-Agg and C-Agg, respectively.

Model Complexity

The training speed or computation speed of any given model is affected by the model size (number of trainable model parameters), the optimizer employed as well as any other computations that directly modify or alter the formulation of the network structure (Popel and Bojar, 2018). The *Layer Aggregation* approaches employ an additional set of weights to generate the joint source representation from the outputs of the encoder layers. Besides, the MLMHA approach introduces new trainable parameters as each decoding layer employs n different set of weights to perform the attention computations across the multiple encoding layers as shown in Section 3.2.2. Therefore, to investigate the impact of the number of parameters on the overall training speed, we train two additional Transformer baseline models (B1 and B2) with different configurations. Specifically, the model B0 is the original Transformer presented in Table 3.2. The baseline B1 is trained with hidden size, filter size, and the number of attention heads set as 512, 4098, and 8, respectively. The main difference between B0 and B2 models is that B2 employs four additional encoding and decoding layers to generate the target translations. The configurations of the B0, B1, and B2 models result in a comparable increase in the number of trainable parameters as that of the MLMHA and *Layer Aggregation* models (when $n = 6$). For example, the B1 model has roughly the same number of parameters as the M-01 and M-11 models. As shown in Table 3.4, increasing the number of parameters generally results in a decrease in the training speed. The new parameters introduced by B1 and B2 configurations degrade the training speed by about 19.2% and 28.77%, respectively. For the *Layer Aggregation* models, the extra model parameters introduced depend on the technique employed to generate the joint source representation H^a. Compared to the linear combination strategies, the iterative source feature combination approaches resulted in bigger model size. For example, C-Agg approach increases the model size by about 7.41M parameters while the Iter-C-Agg had about twice the increase in the number of trainable parameters.

Among the *M-*ij models, the M-00 and M-10 have the worst training speed compared

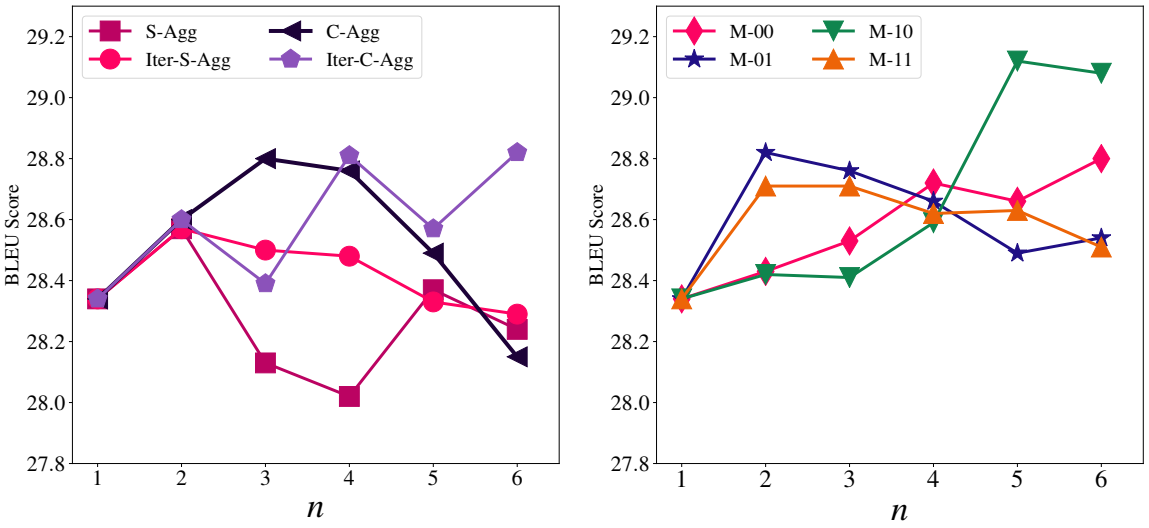


Figure 3.7: Impact of n (the number of encoding layers considered by the *Source Feature Collector* module) on the performance of the JASs. **Left**: *Layer Aggregation* models. **Right**: *Multi-Layer Attention* models

.

to the M-01 and M-11 models. Similar to the *Layer Aggregation* models, the number of new parameters is dependent on the strategy employed to generate the joint context O_c (see Equations (3.7) to (3.11)) across the multiple representations from the encoder subnetwork. When $n = 6$, the MLMHA introduces about 23.7M new parameters due to the n different weights employed to perform the MHA operations across each encoder output as shown in Equations (3.7) and (3.8). For the *contexts-concatenation* based *M-ij* models (with $\bar{U}_1 = 0$), a further 7.8M new parameters are introduced due to the concatenation operation on the context representations $C = [c^1, c^2, \cdots, c^n]$. As shown in Table 3.4, for our MLMHA models, there is a corresponding reduction in the training speed from about 7.13% (when $n = 2$) to 29.04% (for $n = 6$) as the value of n increases. Finally, the B1 model has roughly the same number of parameters as the M-01 and M-11 models. However, the training speed for these *contexts-summation* models is (slightly) slower than B1. This is attributed to the additional attention computations and aggregations across the multiple decoding layers.

Translation Quality

The variation in the translation performance with respect to the values of n is illustrated in Figure 3.7. As seen, for all the *Layer Aggregation* and *Multi-Layer Attention* models, there is a significant change in performance as the value of n increases from 1 to 6. Similar to the observations in Table 3.2, the value of $\bar{U}$ is shown to significantly affect the overall performance of the *Multi-Layer Attention* models. The performance of the models with $\bar{U}_1 = 0$ (M-00 and M-10) improves as the number of encoder layers considered (n) increases. For the value of $n > 4$, these *Multi-Layer Attention* models achieve their best performance. In contrast, the M-01, and M-11 models ($\bar{U}_1 = 1$) achieved their highest translation performance when $n = 2$, but the quality of the output translation degrades for $n > 3$ (with the minimum BLEU score at $n = 6$ for M-11 and at $n = 5$ for the M-01 model). Specifically, the M-01 and M-11 models achieve their highest performance when only the top two encoder layer outputs are considered.

For the *Layer Aggregation* models, the concatenation based feature combination approaches (Iter-C-Agg and C-Agg) are shown to outperform the summation based methods (Iter-S-Agg and S-Agg) for the values of $n \leq 4$. The performance of the C-Agg degrades rapidly when the value n increases especially for $n > 3$. On the other hand, the Iterative Feature Concatenation approach (Iter-C-Agg) is able to effectively utilise the contextual source information obtained across the multiple encoding layers resulting in higher performance than the C-Agg model for $n \geq 4$. Similarly, the Iter-S-Agg performs better than the S-Agg model across the different values of n except at $n \geq 5$ (there is no significant difference in their performance when $n = 5$). However, similar to the M-01, and M-11 models, the performance of these summation based *Layer Aggregation* models is peaked when using source representations from only the top two encoding layers (i.e. H_e^{L-1} and H_e^L). Among the *Layer Aggregation* models, the S-Agg and C-Agg produced worse performance for the values of $2 \leq n \leq 5$ and $n = 6$, respectively.

Overall, the results obtained by the joint attention-based models indicate that performing the encoder-decoder attention across multiple encoder layers can significantly improve the performance of the NMT model. This is dependant on the value n in the case of both the *Layer Aggregation* and *Multi-Layer Attention* approaches and the values of $\bar{U}$ for the *Multi-Layer Attention* strategies as shown in Figure 3.7. However, the performance gain comes at a higher computational cost, especially in the case of M-00, M-10, and Iter-C-Agg models.

Finally, unlike the context concatenation based models (Iter-C-Agg, M-00, and M-10), the performance of context summation based models (S-Agg, Iter-S-Agg, M-01, and M-11) decreases as the value of n increases.

3.5.3 Impact on the Encoder's Self-attention

The performance of the encoding layers depends on the ability of the multiple heads of the self-attention unit within each layer to capture the necessary structural information of the input source sequence. These attention heads capture structural information of the source sequence at varying degrees. As noted by Vig and Belinkov (2019); Raganato et al. (2018), while some self-attention heads focus on long-distance relationships, other attention heads capture the shorter distance relationships between the input tokens. This allows the transformer model to capture the structural information for the given source sentence to improve the translation performance effectively (Raganato et al., 2018). As stated earlier, the operations of the MLMHA module within each decoding layer and the *Layer Aggregation* approaches affect how the source information is processed across the layer of the encoder subnetwork. Following (Vig and Belinkov, 2019), this hypothesis is tested by analysing the attention entropy and the attention distance spanned by the multiple attention heads within each encoding layer's self-attention unit.

The mean distance $\bar{D}_h^l$ spanned by the attention head h with respect to the encoding layer l is computed as the weighted average distance between token pairs in all sentences in a given corpus *X*. That is:

$$\bar{D}_h^l = \frac{\sum_{x \in X} \sum_{i=1}^{|x|} \sum_{j=1}^{i} w_{i,j}^h \cdot (i-j)}{\sum_{x \in X} \sum_{i=1}^{|x|} \sum_{j=1}^{i} w_{i,j}^h}$$

where $w_{i,j}^h$ is attention weight from the input token x_i to x_j for the attention head h. i and j denotes the locations of tokens x_i and x_j in the source sentence x. Aggregating the attention distance for each head, the mean attention distance spanned $\bar{D}^l$ with respect to the encoding layer l is calculated as:

$$\bar{D}^l = \frac{1}{N_h} \cdot \sum_{h=1}^{N_h} \bar{D}_h^l$$

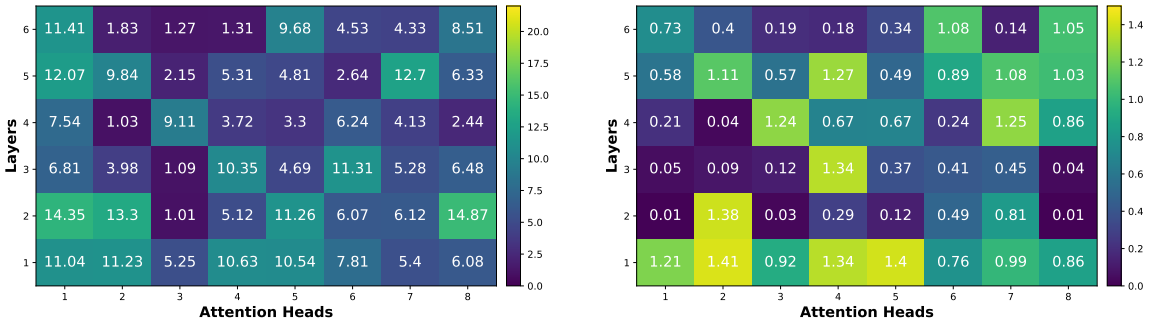


Figure 3.8: Variation of the mean attention distance span and attention distribution entropy with respect to the encoding layers and the attention heads for the Transformer baseline. **Left**: Mean attention distance. **Right**: Entropy of attention distribution.

where N_h denotes the number of attention heads employed within the layer.

The mean attention distance does not offer any information on the distribution of the attention weight across the input tokens for a given attention head. The attention head with a higher mean attention distance can be concentrating on similar token sequences, which might be further apart from each other (Vig and Belinkov, 2019; Ghader and Monz, 2017). To measure the concentration or the dispersion pattern of an attention head h within layer l for the input token x_i, the entropy of the attention distribution (Ghader and Monz, 2017), $E_h^l(x_i)$ for the attention head h is computed as:

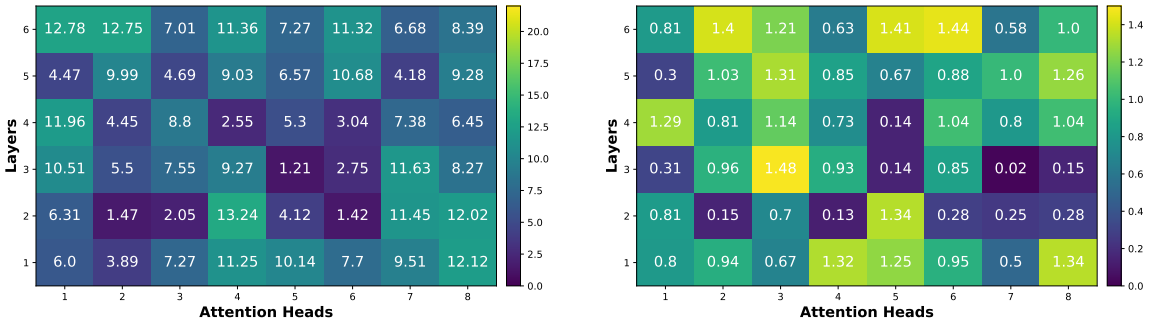
$$E_h^l(x_i) = -\sum_{j=1}^{i} w_{i,j}^h \log w_{i,j}^h$$

Similar to the attention distance spanned, the mean entropy of attention distribution for the encoding layer l is calculated as:

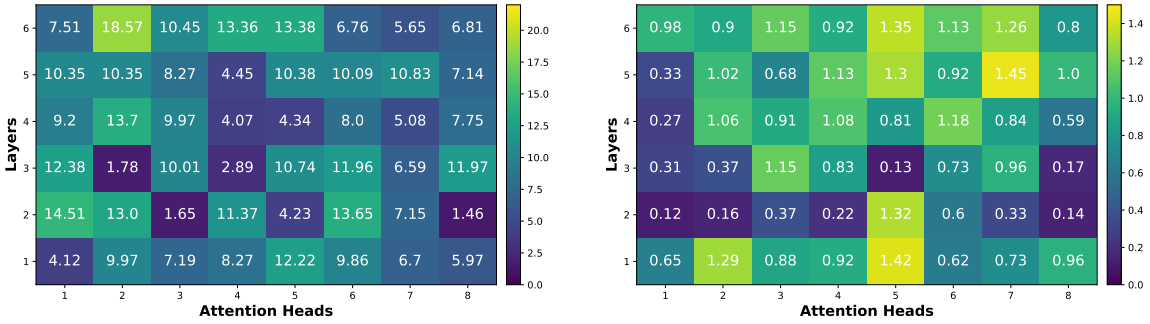
$$E^l(x_i) = \frac{1}{N_h} \sum_{h=1}^{N_h} E_h^l(x_i)$$

Attention heads with higher entropy are termed as having a more dispersed attention pattern while the lower the entropy, the more concentrated the attention weight distribution.

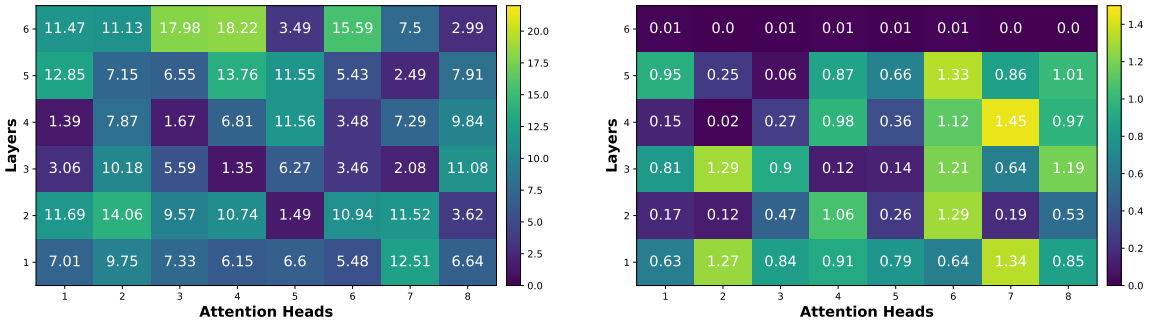
The attention distance and entropy of attention distribution analysis are performed based on the attention weights generated for 1500 randomly sampled sentences from the



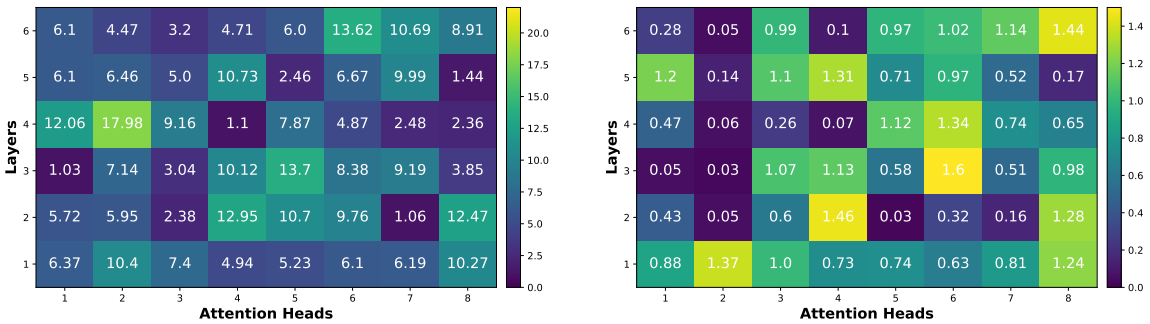
(a) S-Agg



(b) Iter-S-Agg

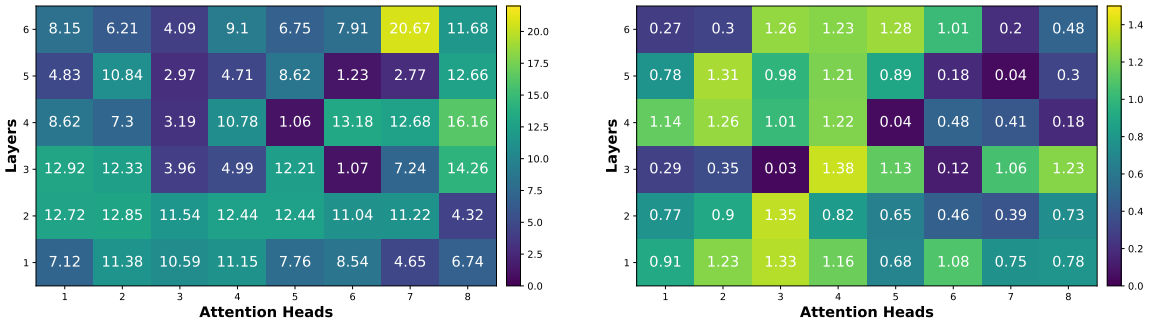


(c) C-Agg

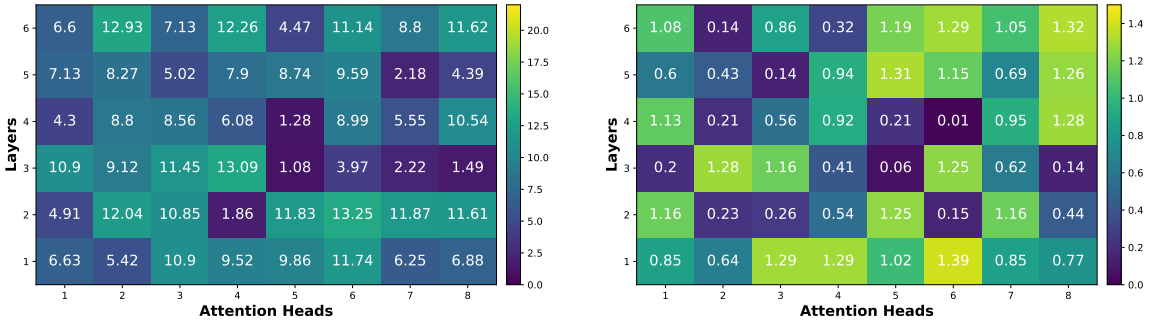


(d) Iter-C-Agg

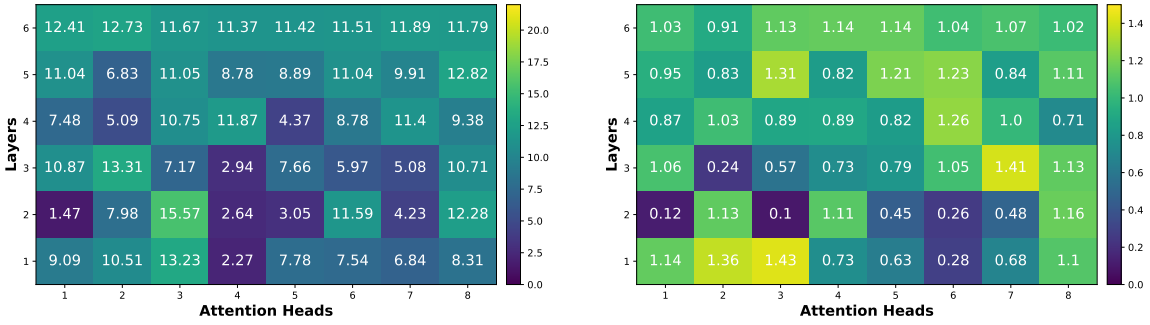
Figure 3.9: Variation of the mean attention distance and entropy of attention distribution for the attention heads across the encoding layers with respect to the *Layer Aggregation* models. For each plot, **Left**: Mean attention distance. **Right**: Mean entropy of attention distribution.



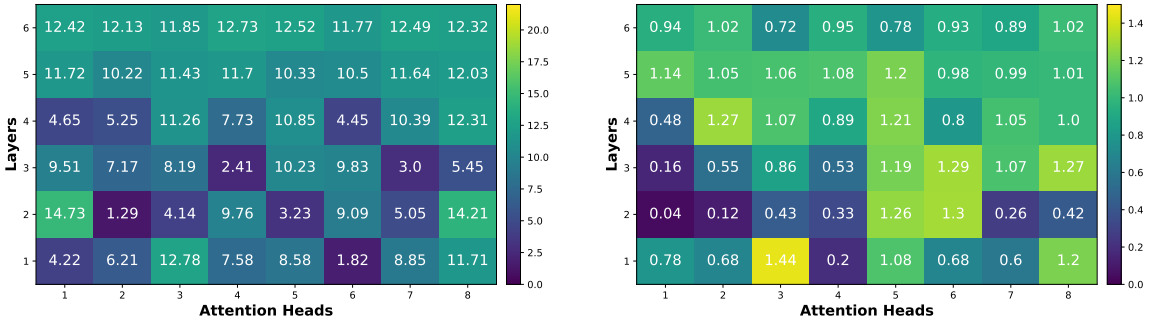
(a) M-00



(b) M-01



(c) M-10



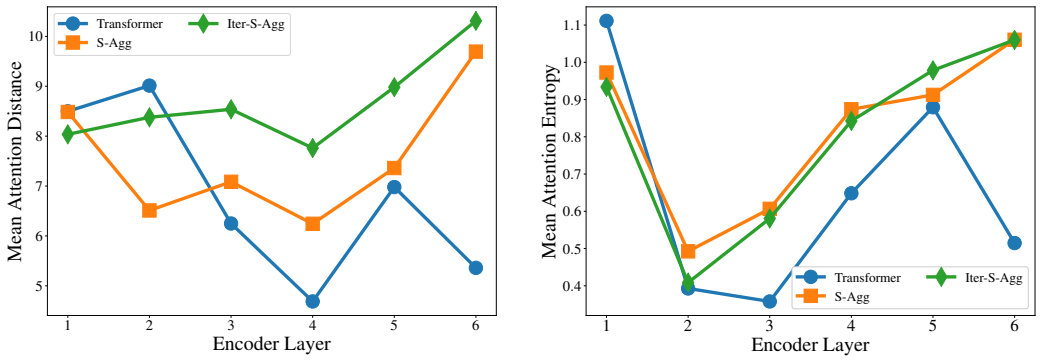
(d) M-11

Figure 3.10: Variation of the mean attention distance and entropy of attention distribution for the attention heads across the encoding layers with respect to the *Multi-Layer Attention* models. For each plot, **Left**: Mean attention distance. **Right**: Mean entropy of attention distribution.

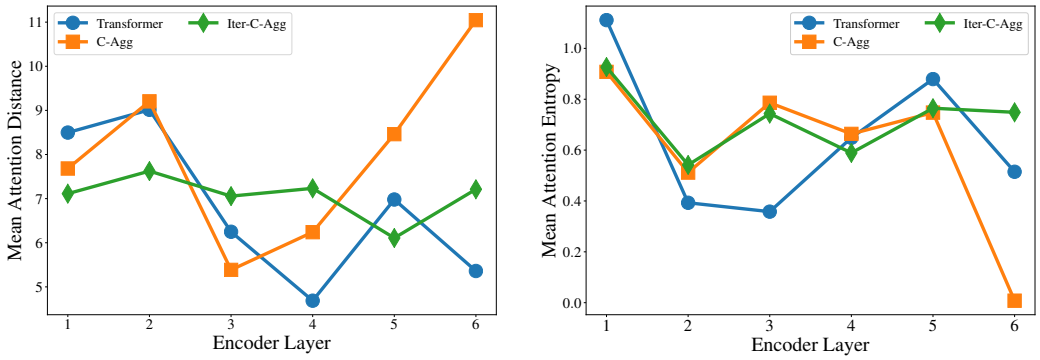
En→De task's test split (newstest2014). Figures 3.8 to 3.10 show the mean attention distance span and mean entropy of attention distribution for every attention head with respect to each encoding layer for the Transformer baseline and the JASs based models, respectively. As shown, while some heads focus on the shorter-distance relationships, other heads capture the longer-distance relations among the input tokens. Similarly, the entropy of the attention distribution also varies across the layers and even for attention heads within the same layer. This is consistent with the findings of both Vig and Belinkov (2019); Ghader and Monz (2017). Figures 3.11 and 3.12 show the mean average attention distance and entropy for all the self-attention heads across the layers of the encoding subnetwork. Each plot compares the Transformer baseline and a joint attention-based model, the variations of the attention distance span, and attention entropy across the encoding layers.

For the Transformer baseline, the majority of attention heads with a higher mean attention span and a more diverse attention distribution are across the first layer. However, a higher mean attention distance does not always imply a diverse attention distribution. In the subsequent layers, there are a number of attention heads with a higher distance span but with a much more concentrated attention weight distribution. For example, layer 2 attention head 1 and head 8 have the highest mean attention spans (14.34 and 14.87 respectively) but with the lowest mean entropy scores (0.0085 and 0.0094). As noted by Vig and Belinkov (2019), these attention heads with higher mean attention distance span concentrate their attention on words in repeated phrases at different locations within the input sentence. This could explain their lower entropy of weight distribution across the sequence of input tokens. Attention heads with diverse or concentrated weight distribution and lower attention distance span focus more on nearby tokens. Clearly, these heads with varying mean attention distance and entropy allow the Transformer to effectively learn and capture variable structural information across its layers. This explains the superiority of the Transformer model over other Seq2Seq architectures such as RNN (Luong and Manning, 2015; Bahdanau et al., 2015), and CNN (Gehring et al., 2017).

For the *Layer Aggregation* and the *Multi-Layer Attention* based models, the impact of the different strategies on the self-attention unit within the associated encoding layer is of greater interest. Exposing all the encoding layers to the decoding subnetwork significantly alters how the source information is learned across the encoder subnetwork, as displayed in Figures 3.9 to 3.12. Compared to the Transformer baseline, the attention heads of the



(a)



(b)

Figure 3.11: Variation of the average mean attention distance and entropy of head attention distribution across the encoding layers for the Transformer baseline model and the *Layer Aggregation* based models: (a) *Feature Summation* and (b) *Feature Concatenation*. For each plot, **Left**: the average of all the attention head mean distance with respect to each encoder layer. **Right**: the average entropy of head attention distribution per encoder layer.

Feature Summation approaches (i.e. S-Agg and Iter-S-Agg) generally have higher attention distance span and entropy across the top-level encoding layers $l \geq 2$. As illustrated in Figure 3.11a, for the S-Agg model, the entropy of the attention distribution and attention distance span is maximum across the layers 1 and 6. The attention heads across the second layer are more concentrated than all the other encoding layers. The model trained via the Iterative Feature Summation displays a similar variation with respect to the entropy of attention distribution. However, across the encoding layers of the Iter-S-Agg model, the majority of attention heads have higher attention distance and are more diverse than that of the S-Agg.

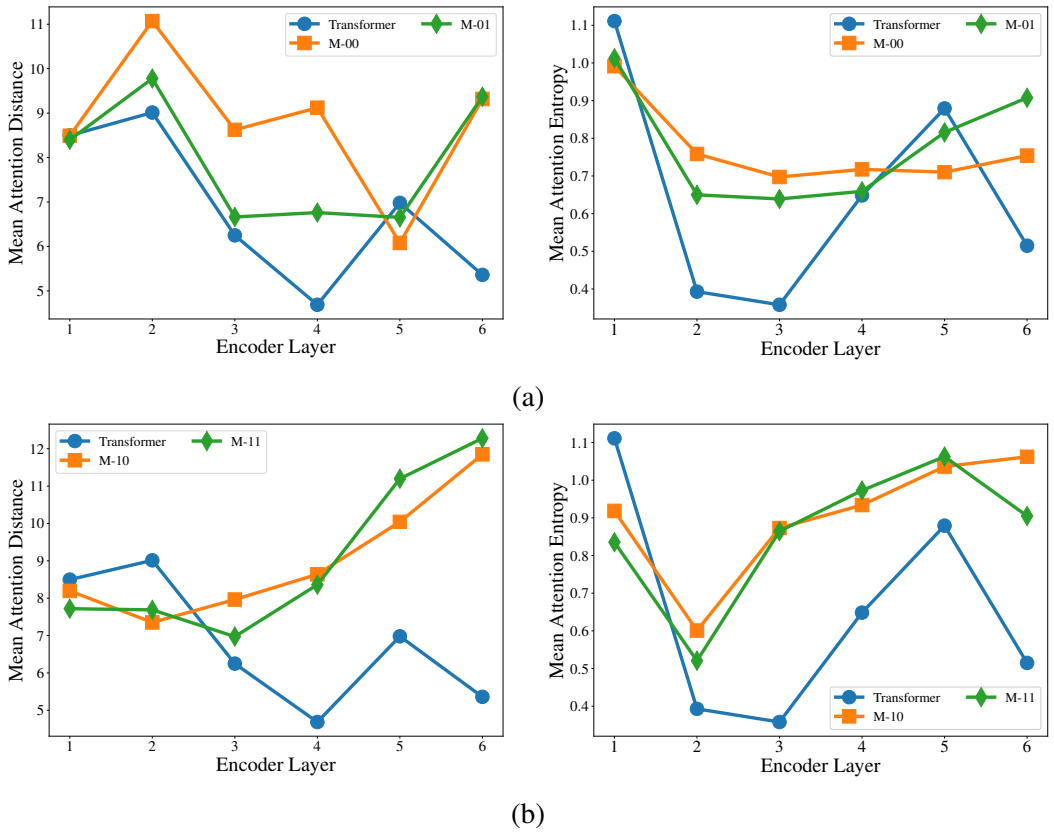


Figure 3.12: Variation of the average mean attention distance and entropy of head attention distribution across the encoding layers for the Transformer baseline model and our *Multi-Layer Attention* models: (a) *M-ij* models trained with *joint-attention weight* (when $\bar{U}_0 = 0$) and (a) *M-ij* models trained with $\bar{U}_0 = 1$, *layer-specific-attention weights*. For each plot, **Left**: the average of all the attention head mean distance with respect to each encoder layer. **Right**: the average entropy of head attention distribution per encoder layer.

The changes in the average mean attention distance and entropy of attention distribution across the encoder layers of the *Feature Concatenation* approaches (i.e. C-Agg and Iter-C-Agg) is summarized in Figure 3.11b. Under C-Agg, the attention distance increases rapidly across the layers $3 \leq l \leq 6$. The final layer has the highest distance span, with the highly concentrated attention heads with the entropy of about $8e^{-4}$. Compared to C-Agg, the Iter-C-Agg produces a moderate variation in the attention distance span and entropy across the layers. This allows the Iter-C-Agg to effectively model both the long and short-term interactions and dependencies between the source tokens. This might explain the difference in performance between the C-Agg and Iter-C-Agg across the different sentence groups, as

Models \ Layer	1	2	3	4	5	6
Layer Aggregation						
S-Agg	-27.01‡	-0.12	+0.10	-0.20	-0.11	-0.94‡
C-Agg	-21.56‡	-0.14	+0.06	-0.26	-0.37	1.85‡
Iter-S-Agg	-0.03	-0.01	-0.12	-0.05	0.0	-28.29‡
Iter-C-Agg	-4.53‡	-0.54‡	-0.59‡	-2.21‡	-4.0‡	-27.16‡
Multi-Layer Attention						
M-00	-15.25‡	-0.49†	-0.27	0.03	-1.79‡	-9.85‡
M-01	-24.99‡	-0.19	-0.09	0.13	-0.81‡	-1.49‡
M-10	-0.83‡	-1.32 ‡	-1.05 ‡	-1.0 ‡	-1.50‡	-1.09‡
M-11	-0.26	-1.13‡	-0.83 ‡	-1.43‡	-1.34‡	-0.1

Table 3.5: Difference in BLEU scores for each encoding layer masked (i.e. replacing the corresponding $f^i \in F^s$ with zeros) with respect to the models when $n = L$. "‡" and "†" indicate statistically significant difference with $\rho < 0.01$ and $\rho < 0.05$, respectively. The base-BLEU scores for the *Layer Aggregation* and *Multi-Layer Attention* models are shown in Table 3.2.

shown in Figure 3.6.

As evident from Figure 3.12, for the *Multi-Layer Attention* models, the change in terms of the average mean attention distance span and entropy of attention weight distribution for the multiple attention heads across the different encoder layers is dependent on the value of the $\bar{U}_0$. The M-00 and M-01 models (with $\bar{U}_0 = 0$) have concentrated attention heads with shorter attention distance span across the intermediate layers $3 \leq l \leq 5$. These intermediate layers are used to learn the local contextual information within the neighbourhood of the input source tokens. In contrast, the M-10 and M-11 models (with $\bar{U}_0 = 1$) employs the first few layers ($l \leq 3$) to learn the short-term information while the upper layers model the long-distance interaction between the input tokens.

Overall, each joint strategy significantly modifies how the source information is captured across the multiple attention heads and layers within the encoding subnetwork, as shown by attention distance and entropy of attention weight distribution. This further enhances the network's performance at learning the deeper source semantic information needed to improve the translation quality, especially in the case of the Iter-C-Agg and *M-*ij models.

3.5.4 An ablation study: Encoder Layer Dependency

The translation performance of the *Layer Aggregation* and *Multi-Layer Attention* models reported in Table 3.2 are based on exposing all the encoding layers to the decoder (i.e. $n = L$). However, it is worth understanding the contribution of each encoder layer to the overall performance of each *Layer Aggregation* and *Multi-Layer Attention* models. To this end, the translation quality of each model is evaluated while masking the entry in F^s corresponding to the encoder layer of interest. Here, masking an entry in F^s implies replacing the corresponding f^i with zeros. If the performance without the output of the encoder layer l (i.e. H_e^l) is significantly worse than the full model, then the H_e^l is clearly important. In contrast, H_e^l is considered redundant if the difference in translation performance is comparable.

Table 3.5 shows the difference in the performance of the models for each masked output of the encoder. As shown, masking one of the outputs of the encoder layers (in most cases) degrades the translation quality significantly. The S-Agg and C-Agg models have identical dependencies on the representations from the different encoder layers. For these models, only the first and last encoder layers are shown to have a significant impact on translation performance. Masking the source representations from the intermediate layers ($3 \leq l \leq 5$) does not significantly affect the performance. Interestingly, the translation performance of the Iter-C-Agg is shown to be dependent outputs from all the encoder layers. In contrast, only the output from the final encoder layer significantly contributes to the overall performance of the Iter-S-Agg model.

The performance *M-*ij models also show different dependencies on the representations from the encoder subnetwork. For example, without the output of the first encoder layer, the performance of both M-00 and M-01 model decreases by -15.25 BLEU and -24.99 BLEU, respectively. Surprisingly without the output from the encoder layer 4, there is a marginal improvement (not statistically significant) in the translation quality of these models. Notably, the source representations from first and final encoding layers are shown to be redundant to the translation performance of the M-11 model, however, the outputs from these layers have a statistically significant impact on the overall performance of the M-00, M-01 and M-10 models. The results in Table 3.5 demonstrates that for S-Agg, Iter-S-Agg, C-Agg, M-00, M-01, and M-11 models, the outputs from some of the encoder layers are redundant during testing and can be removed without significantly reducing the translation

quality. Consistent with the observation in Section 3.5.2, the translation performance of the Iter-C-Agg and M-10 models is shown to be highly dependent on source representations from all encoder layers. Removing the output of any of these layers causes a statistically significant change in performance. Overall, the *Multi-Layer Attention* models are shown to effectively utilise the multiple representations from the encoder subnetwork compared to all the *Layer Aggregation* models (except Iter-C-Agg model).

3.6 Summary

In this chapter, the performance of the Transformer model is improved by leveraging multiple source representations captured by different encoding layers. Specifically, the decoding subnetwork is allowed access to the entire stack of encoding layers to extract better source-target contextual information. Experimental results on the IWSLT tasks (Spanish-English and English-Vietnamese) and the WMT'14 English-German translation task show that the JASs can further improve the performance of the NMT model. The value of n (number of encoding layers considered) is shown to affect the performance of all the joint attention strategies (*Layer Aggregation* and MLMHA based models) explored. However, for the *M-*ij models, the performance gain is also dependent on the values of the binary vector $\bar{U}$. Further analysis also demonstrates that exposing the layers of the encoding subnetwork significantly alters how the global and local source contextual information is captured by the self-MHA sublayer employed within each encoder layer.

Chapter 4

Encoder-based Multi-level Supervision for Neural Machine Translation

The work presented in the previous chapter emphasised specifically on improving the target generation ability of the decoding subnetwork by leveraging source representations from multiple encoder layers. However, compared to the decoder, the encoding subnetwork has to some extent a greater impact on the overall performance of the NMT model. Therefore, enhancing the source representation ability of the encoder can significantly improve the performance of the model. Similar to the previous chapter, the work presented in this chapter also leverages multiple source representations from the encoder subnetwork. However, the goal here is to exploit the strengths of MTL, auxiliary training, and deep representational learning to improve the performance of the encoder subnetwork. To this end, the chapter presents *Encoder-based Multi-level Supervision* strategies whereby multiple decoding subnetworks (connected to different encoding layers) are jointly trained end-to-end on the same target generation task. Via inductive bias, the lower-level encoder layers are trained to learn the necessary source information to support the target generation from all the connected decoders.

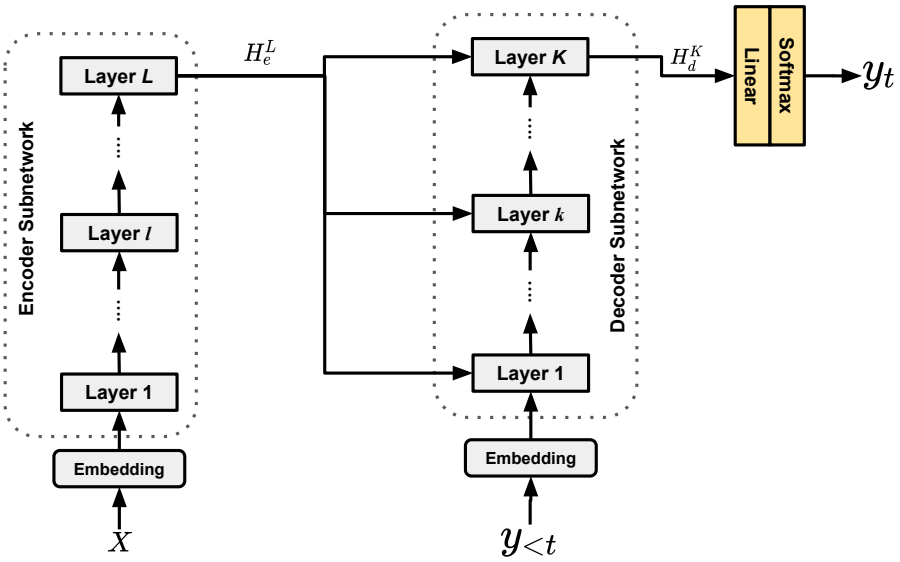


Figure 4.1: Illustration of encoder-decoder framework for sequence generation tasks such as NMT, and document summarization. X denotes the input source sequence. y_t target token at time step t and $y_{<t} = y_1, \cdots, y_{t-1}$ denotes the partial target sequence generated before step t. *Embedding* denotes the word embedding layers employed to generate input representations for the encoder and decoder subnetworks.

4.1 Introduction

State-of-the-art NMT models (Vaswani et al., 2017; Gehring et al., 2017) usually implement the encoder subnetwork with multiple layers to learn the source semantic features efficiently and effectively. The target sequence generation is performed by the decoding subnetwork based on only the source semantic features from the final encoding layer, as shown in Figure 4.1 with or without the neural attention mechanism. Each encoding layer captures different linguistic levels of source representations, including word-level, syntactic, and semantic information (Belinkov et al., 2017; Hashimoto et al., 2017; Raganato et al., 2018). The lower layers are best at learning word-level properties such as parts-of-speech tagging (POS) and morphological tags. In contrast, the higher-level layers focus more on learning semantic information based on the properties encoded by the bottom encoding layers. This implies that understanding the kind of information encoded across the different layers can further enhance the performance of sequence generation models. Based on these findings, the previous chapter explored the *Joint Attention Strategies* (JASs) to leverage the

different levels of abstractions of source sequence generated from the multi-layer encoder subnetwork. The *Layer Aggregation* and *Multi-Layer Attention* strategies achieved (in most cases) significant performance improvement over the conventional architecture that generates the target sequence based on the source representation from only the final encoder layer. The common theme among these approaches is improving the performance of the decoding subnetwork. However, the translation performance of an NMT system relies on the sentence representation and generation ability of both encoder and decoder subnetworks. Since the decoder subnetwork largely depends on the encoder subnetwork to learn the necessary source information for the target generation task, the encoder has to some extent has a greater impact on the overall translation performance of the NMT model.

Inspired by recent advances in multi-task learning (MTL) and deep representation learning, this chapter investigates approaches to enhance the source representation ability of the encoding subnetwork by exploiting the multiple source representations across the layers via *Multi-level Supervision* (MS). Specifically, multiple decoders are connected to different layers of the encoder. Unlike conventional multi-task NMT models (Niehues and Cho, 2017; Baniata et al., 2018; Luong et al., 2015a; Malaviya et al., 2017), all the decoders connected to the encoding subnetwork are jointly trained end-to-end on the same target language generation task based on source representation captured by the corresponding encoding layers. The decoder connected to the final encoding layer is termed as the *Main-Decoder* while other decoders attached to any of the lower-level layers are termed the *auxiliary decoders*. During training, the encoder subnetwork receives gradient signals from all connected decoders. Therefore, to support the target generation from each connected decoder, the lower-level layers are forced to capture the necessary source semantic information. This presents a form of inductive bias effect on the encoder subnetwork, further improving its ability to generate high-quality source representations effectively. Finally, this work argues that further performance improvement can be achieved by sharing information directly between the *Main-Decoder* and the auxiliaries. Specifically, an *Auxiliary Information Fusion* module is employed by the *Main-Decoder* to exploit the target representations learned by the auxiliary decoders.

The MS approaches explored in this chapter are model-agnostic as such they can be applied to most existing encoder-decoder architectures including CNN (Gehring et al., 2017), RNN (Luong et al., 2015b; Wu et al., 2016) and Transformer (Vaswani et al., 2017). As

with the experiments presented in the previous chapter, all experiments and analyses are conducted using the Transformer model. The MS approaches explored here are evaluated on four language translation tasks: IWSLT'14 German-English, Spanish-English, English-Vietnamese from IWSLT'15, and the WMT'14 English-German. The results on these translation tasks demonstrate the performance gain from training the original encoder-decoder model with an auxiliary decoder connected to any of the lower-level encoder layers. The improvement achieved is attributed to the enhancement of the source representational ability of the encoder subnetwork via the multi-level supervision. This hypothesis is verified by the results based on ten linguistic probing tasks (Conneau et al., 2018) presented in Section 4.5.4. The contributions of this chapter are:

- Proposing multi-level supervision approaches to improve the source representation ability of the encoding subnetwork.
- Demonstrating consistent improvement over models with a single decoder connected to the final encoder layer.
- Providing analysis on the impact of the auxiliary decoder connected to any of the lower-level layers on the entire encoder.
- Providing analysis on the impact of the *Auxiliary Information Fusion* module on the *Main-Decoder*.

The remainder of the chapter is organised as follows: The multi-level supervision approaches are presented in Section 4.2. The experiments conducted are presented in Section 4.3, and the results are compared and discussed in Section 4.4. Furthermore, Section 4.5 presents a detailed analysis on the impact of connecting an auxiliary decoder to any lower-level encoder layer and the conclusion is presented in Section 4.6.

4.2 Approach

The vanilla Seq2Seq architecture employs a single decoder subnetwork to generate the target sequence based on the output of only the final encoder layer (H_e^L), as shown in Figure 4.1. However, a multi-layer encoder network can provide multiple levels of abstraction of the source sequence, and each of these abstractions can be used to generate the target

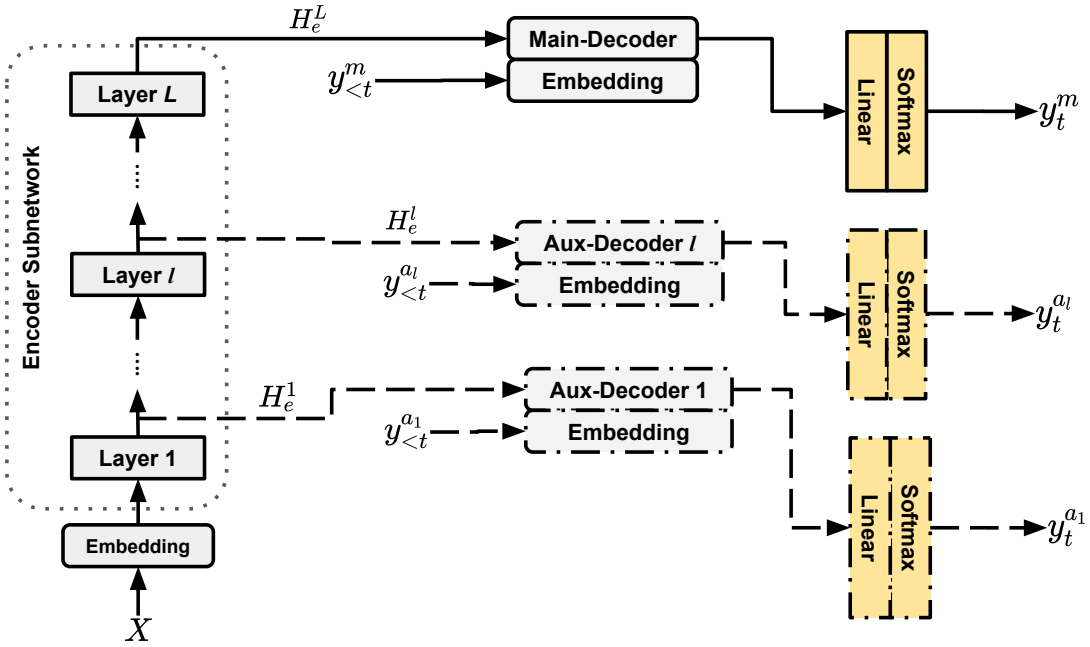


Figure 4.2: Encoder-based Multi-level supervision approach for sequence generation where X is the input sequence. *Main-Decoder* denotes decoding subnetwork connected to the top most layer in a L-layers encoding network and *Aux-Decoder l* denotes the auxiliary decoder connected to the l-th encoding layer (where $l < L$).

sequence. In this section, an *Encoder-based* MS approach to leverage the source representations from multiple encoding layers is introduced.

Multi-level Supervision (MS)

Multi-level supervision is a learning technique where predictors connected to some or all of the lower-level layers of a neural network are trained along with the predictor connected to the final layer. Auxiliary training (Szegedy et al., 2015; Gehlot et al., 2020; Nekrasov et al., 2019) and some multi-task learning (MTL) approaches (Hashimoto et al., 2017; Ampomah et al., 2019a) are typical examples of multi-level supervisions. The auxiliary training approach improves the convergence of deep networks by attaching auxiliary classifiers to certain intermediate layers. In addition to the original network, the network structures of these auxiliary classifiers, as noted by Song and Chai (2018) and Jin et al. (2016) require specific new designs. Under MTL, multiple different tasks can be supervised simultaneously across

multiple layers. This approach has been shown to produce better performance, especially when there is some form of linguistic hierarchies between the main and auxiliary tasks (Hashimoto et al., 2017; Ampomah et al., 2019a).

The encoder-decoder architecture of the Seq2Seq model provides two possible directions to performing multi-level supervision, namely the *Encoder-based* MS and *Decoder-based* MS. The *Encoder-based* MS approach seeks to exploit the features learned from the input source sequence across multiple encoding layers to further improve the target generation. On the other hand, the *Decoder-based* MS approach uses the representation of both the target and source sequences captured across each decoder layer to improve the model's performance. In this chapter, only *Encoder-based* MS is considered[1]. Specifically, this work mainly investigates whether the performance of the main generator connected to the final encoder layer can be improved by jointly training with auxiliary generators attached to any of the bottom or intermediate encoding layers.

Encoder-based Multi-level Supervision

The *Encoder-based* multi-level supervision as shown in Figure 4.2 leverages multiple outputs of the encoder subnetwork. To this end, auxiliary decoding subnetworks are added to the original network, which utilises outputs from some selected lower-level encoding layers. The decoder generating the target based on the final encoding layer's output (H_e^L) is termed the *Main-Decoder*. Unlike typical auxiliary training-based models (Szegedy et al., 2015; Nekrasov et al., 2019), each of the connected auxiliary generators has an identically similar network structure as the *Main-Decoder*. $Y^m = (y_1^m, y_2^m, \cdots, y_M^m)$ is the output of the *Main-Decoder* and $Y^{a_l} = (y_1^{a_l}, y_2^{a_l}, \cdots, y_M^{a_l})$ is the output of auxiliary decoder attached to the lower-level encoding layer l (i.e. *Aux-Decoder l*). It is noteworthy that these decoders are different target generation subnetworks connected to different encoding layers. Therefore, ideally we expect $y_t^m = y_t^{a_l} = Y_t$. For simplicity, the auxiliary decoders are depicted with dotted lines to indicate that they are optional, as shown in Figure 4.2.

The number of parameters of the resulting model increases as the number of auxiliary decoders connected to the encoder increases. Therefore to reduce the number of trainable parameters of the resulting model, the same output linear layer is employed to generate the

[1]This is because the encoder subnetwork has a greater impact on the performance of the model compared to the decoding subnetwork.

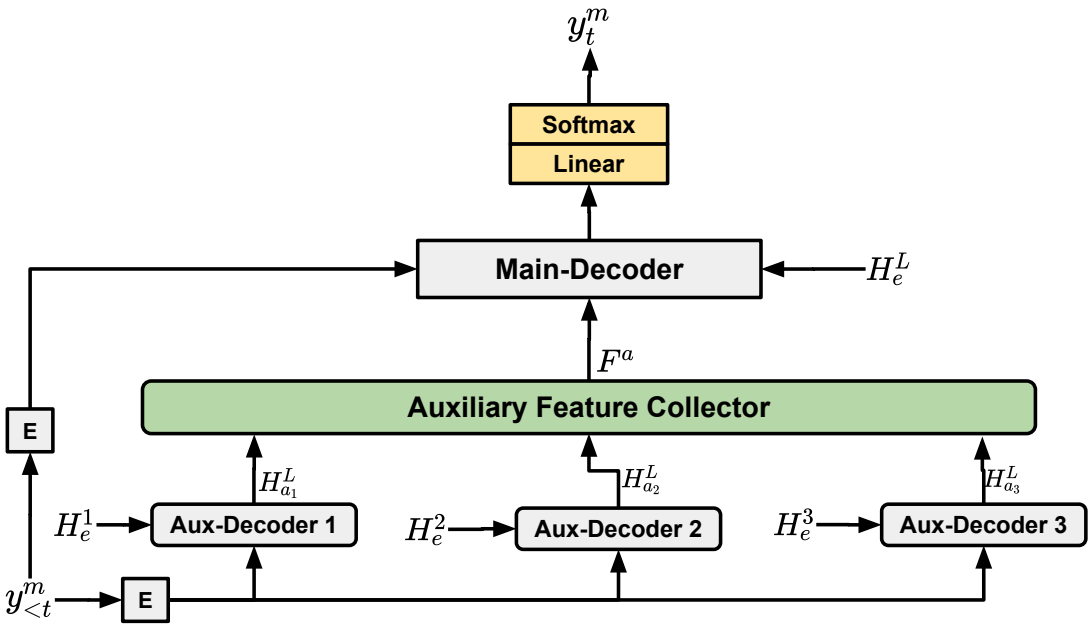


Figure 4.3: Illustration of the *Main-Decoder* exploiting auxiliary representations F^a aggregated by the *Auxiliary Feature Collector* module based on the target representations, $[H_{a_1}^L, H_{a_2}^L, H_{a_3}^L]$, from three connected auxiliary decoders. For simplicity, we denote the word embedding layer as *E*.

target sequence from the *Main-Decoder* and the auxiliaries. The resulting model is trained jointly end-to-end using multi-output cross-entropy loss computed across each decoder under consideration. The joint learning objective $\mathcal{J}\left(Y|X;\theta\right)$ is formulated as:

$$\mathcal{J}\left(Y|X;\theta\right) = \mathcal{L}^m\left(Y|X;\theta\right) + \gamma \sum_l \mathcal{L}^{a_l}\left(Y|X;\theta\right) \tag{4.1}$$

where $\mathcal{L}^m\left(Y|X;\theta\right)$ is the loss from the *Main-Decoder*, and $\mathcal{L}^{a_l}\left(Y|X;\theta\right)$ is the loss computed across the outputs of *Aux-Decoder l*. γ is a hyper-parameter or a discount weight controlling the impact of the auxiliary decoders. For simplicity, the discount weight is set to 1 in all our experiments. This implies that all decoding subnetworks have an equal impact on the learning process.

Learning the target sequence generation from multiple layers serves as an inductive bias mechanism to improve the source representation ability of the encoder subnetwork. During backpropagation, the lower layers receive error signals from not only the *Main-Decoder* but

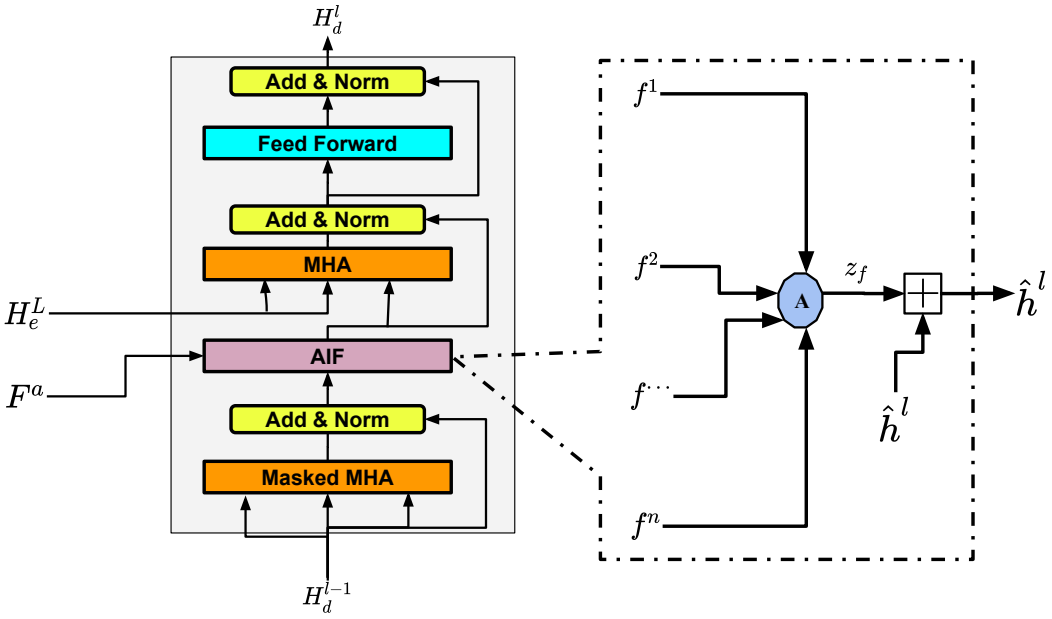


Figure 4.4: A layer of the *Main-Decoder* subnetwork with the *Auxiliary Information Fusion* (AIF) module to processes the auxiliary feature representations $F^a = [f^1, f^2, \cdots, f^n]$. H_d^{l-1} and H_d^l are the input and output of the *Main-Decoder*'s layer l. $\hat{h}^l$ is the output of the masked self-attention sublayer. z_f is the joint auxiliary representation computed by the aggregation unit A as a weighted summation of all the F^a vectors.

also from all the connected auxiliary decoders. Therefore, each lower layer is expected to fully learn the necessary source semantic information required by the connected auxiliary decoders as well as the *Main-Decoder* to successfully generate the target sequence. All connected decoders are capable of generating the target translation based on the source representation from the corresponding encoder layer. However, during inference, the target for the given source sequence is generated from only the *Main-Decoder*. The auxiliary decoders are only employed to further enhance the performance of the encoder subnetwork and, as such, can be discarded after training.

Auxiliary Information Fusion (AIF)

In the basic *Encoder-based* MS approach presented above, only the lower layers of the encoder subnetwork are shared by all connected decoders. During training and inference,

there is no explicit information passing between the *Main-Decoder* and the connected auxiliaries. However, as shown by Ampomah et al. (2019a), information passing between the auxiliaries and the *Main-Decoder* can further improve the performance of the model. This allows gradient information from the *Main-Decoder* to flow through the encoder network as well as the auxiliary decoders.

To this end, two modules (the *Auxiliary Feature Collector* and the *Auxiliary Information Fusion (AIF)*) are employed to collect and process the target representations from the connected auxiliary decoders. As illustrated in Figure 4.3, the *Auxiliary Feature Collector* module generates the auxiliary feature list $F^a = [f^1, f^2, \cdots, f^i, \cdots, f^{n-1}, f^n]$, where $f^i \in \mathbb{R}^{Z \times d_{model}}$ is the hidden representation generated by the i^{th} auxiliary decoder based on the target sequence and the source representation from the associated lower-level encoder layer. The F^a is passed to each layer of the *Main-Decoder* which is processed by the AIF module as displayed in Figure 4.4.

In each layer, the information fusion is performed immediately after the masked-self-attention sublayer. The goal is to update the output of the *Main-Decoder*'s self-attention output ($\hat{h}^l$) with the target information learned by auxiliary decoders. Within the AIF module, a joint auxiliary representation z_f is computed by the aggregation unit A as a weighted summation of all the auxiliary features vectors F^a:

$$z^f = \sum_{i=1}^{n} W_l^{a_i} f^i \tag{4.2}$$

where $\{W_l^{a_1}, W_l^{a_2}, \cdots, W_l^{a_n}\}$ are trainable weight parameters for AIF module of the l^{th} layer of the *Main-Decoder* subnetwork. Each $W_l^{a_i} \in \mathbb{R}^{d_{model} \times d_{model}}$ controlling the contribution of auxiliary feature f^i in the computation of z_f. Finally, the *Main-Decoder*'s internal representation $\hat{h}^l$ is updated via simple summation:

$$\hat{h}^l = \hat{h}^l + z^f \tag{4.3}$$

It should be noted that, when all the $\{W_l^{a_1}, W_l^{a_2}, \cdots, W_l^{a_n}\}$ weight vectors become zero (i.e. when $z^f = 0$), the *Main-Decoder* becomes similar to performing the multi-level supervision without the AIF module as shown in Equation (4.3). Unlike the basic *Encoder-based*

MS, the auxiliary decoders are not discarded during inference, as they contribute to the generation of the target sequence via the AIF module.

4.3 Experimental Setup

4.3.1 Datasets

The proposed multi-level supervision approaches are evaluated on three IWSLT tasks (Spanish-English (Es→En), German-English (De→En), English-Vietnamese (En→Vi)) and WMT'14 English-German (En→De) translation tasks.

The datasets for the Es→En, En→Vi and WMT'14 En→De translation tasks were introduced in Section 3.3. As with the Es→En task, the dataset employed to train the models on the De→En is from the IWSLT 2014 evaluation campaign (Cettolo et al., 2014). The training dataset consists of about 160k sentence pairs. Following (He et al., 2018; Bahdanau et al., 2017; Huang et al., 2018), the validation set is generated by a random selection of 7k sentences from the training data, and the models are trained on the remaining 153k. The dev2010, tst2010, tst2011, and tst2012 are combined to create the test-set. However, unlike the other translation tasks under consideration, all tokens in both the source and target sentences are lower cased[2] for this translation task. The sentences are encoded with byte-pair-encoding (BPE)[3] (Sennrich et al., 2016) based subword tokens with 32k merge operations (resulting in a shared vocabulary of 31k subword tokens).

4.3.2 Model Configuration

The small configuration of the Transformer network is employed in all evaluations on the IWSLT datasets (i.e. En→Vi, Es→En and De→En translation tasks). Specifically, the word embedding dimension, hidden state size (d_{model}), and the number of attention heads are set as 256, 256, and 4, respectively. Furthermore, the position-wise FFN sublayer has a filter of dimension $d_{ff} = 1024$. The number of layers employed by the encoder and decoder subnetworks is 4. The 4-layers of the encoding subnetwork presents three possible locations to connect an auxiliary decoder. Following (Dou et al., 2018, 2019), the network trained on

[2]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/lowercase.perl
[3]https://github.com/rsennrich/subword-nmt

the WMT'14 En→De translation task is based on the base configuration of the Transformer network. Unlike the small configuration, the hidden state size, filter size, and the number of attention heads for the base configuration are 512, 2048, and 8, respectively. Both sub-networks (the encoder and decoder) comprise 6 layers. For a 6-layer encoder subnetwork, there are five possible locations to connect an auxiliary decoder.

For simplicity, the *Encoder-based MS* model with the auxiliary decoder connected to the lower encoding layer l is denoted as *MS-l*. Furthermore, *MS-l+AIF* denotes the *MS-l* model sharing information between the *Main-Decoder* and the auxiliary decoder via the *Auxiliary Feature Collector* and the AIF module. The regularization dropout rate is 0.1. A label smoothing (Vaswani et al., 2017; Pereyra et al., 2017) with 0.1 weight is applied to obtain the uniform prior distribution over the target vocabulary to further improve the translation quality of the connected decoders.

4.3.3 Model Training and Inference

For the IWSLT tasks, all models are trained with a batch-size of 2048 tokens for a total of 200k iterations. In contrast, for the WMT'14 En→De task, the *MS-l* and *MS-l+AIF* models are trained for 160k iterations with a batch size of 4960 tokens. The models are trained using the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^9$. Following (So et al., 2019), a *single-cosine-cycle* with warm-up is employed as the learning rate scheduling algorithm.

During inference, for all the IWSLT tasks, the target sentences are generated using beam search with a beam size of 6 and length penalty $\alpha = 1.1$. The beam search on the WMT'14 En→De task is performed with a beam size and α set as 4 and 0.6, respectively. The BLEU (Papineni et al., 2002) score metric is employed to measure the translation quality. Consistent with previous works, higher BLEU score implies better translation quality. Following common practice, the translation quality is evaluated using the case-insensitive and case-sensitive BLEU metric with multi-bleu.pl for the De→En and Es→En, respectively. For the En→Vi translation task, sacreBLEU is employed to compute the case-sensitive BLEU scores. Finally, on the WMT'14 En→De, the evaluation metric is case-sensitive detokenized BLEU computed with mteval-v13a.pl[4].

[4]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/mteval-v13a.pl

Model	**BLEU**
Luong & Manning (Luong and Manning, 2015)	23.30
NPMT (Huang et al., 2018)	27.69
NPMT + LM (Huang et al., 2018)	28.07
Our NMT Systems	
Transformer	30.58
With Joint Attention Strategies	
Layer Aggregation (Iter-C-Agg)	31.13 (+0.55)
Multi-Layer Attention (M-01)	31.07 (+0.49)
Multi-Layer Supervision	
MS-1	**31.32** (+0.74)
+ AIF	31.29 (−0.03)
MS-2	31.13 (+0.55)
+ AIF	31.52 (+0.39)
MS-3	31.03 (+0.45)
+ AIF	**31.64** (+0.61)

Table 4.1: Evaluation of translation performance on the IWSLT English-Vietnamese (En→Vi) compared with Transformer baseline and other existing models. "MS-1", "MS-2" and "MS-3" detonates the multi-level supervision (MS) with the auxiliary decoder connected to encoding layer 1, 2, and 3, respectively. "+ AIF" detonates training the *MS-l* model with (i.e. *MS-l+AIF*). The values in parentheses indicate the progressive gains between the *MS-l* models and the corresponding *MS-l+AIF* models and the performance gains in the case of the JASs models.

4.4 Results

This section evaluates the performance of the proposed MS approaches presented in this chapter on the four translation tasks. The results reported here are based on the translation performance of the *Main-Decoder*[5]. For each language translation task, the translation performance of MS models is compared to the previously proposed encoder-decoder frameworks. Furthermore, for the En→Vi, Es→En, and WMT'14 En→De translation tasks, the translation performance are also compared to the JASs explored in Chapter 3. Table 4.1

[5]The translation performance of the auxiliary decoders is presented in Section 4.5.3.

Model	**BLEU**
NPMT (Huang et al., 2018)	29.92
NPMT + LM (Huang et al., 2018)	30.08
Dual Transfer Learning (Wang et al., 2018b)	32.35
Reinforcement Learning (Edunov et al., 2018)	32.85
Variational Attention (Deng et al., 2018)	33.10
Model-level dual learning (Xia et al., 2018)	34.71
Tied Transformer (Xia et al., 2019)	35.10
Layer-wise Coordination (He et al., 2018)	35.07
Our NMT Systems	
Transformer	34.53
Multi-Layer Supervision	
MS-1	34.87 (+0.34)
+ AIF	35.12 (+0.25)
MS-2	34.96 (+0.43)
+ AIF	35.20 (+0.24)
MS-3	34.83 (+0.30)
+ AIF	35.14 (+0.31)

Table 4.2: Evaluation of translation performance on the IWSLT German-English (De→En) compared with Transformer baseline and other existing models.

displays the results on the En→Vi translation task. The results for the De→En, Es→En, and WMT'14 En→De translation tasks are summarized in Tables 4.2 to 4.4, respectively. In each table, the progressive gains over the baseline model between the *MS-l* and the corresponding *MS-l+AIF* models are given in parentheses. For the JASs models, only the best performing models for each language task are considered here. Across the different translation tasks, the Transformer baseline consistently and significantly outperformed strong RNN baselines including (Huang et al., 2018; Deng et al., 2018; Cettolo et al., 2014), demonstrating its superiority. A further performance gain is achieved on the Es→En and De→En tasks via the parameter sharing approaches proposed by (He et al., 2018; Xia et al., 2019, 2018). The results achieved by these models show the importance of parameter sharing between the encoder and decoding subnetworks.

Model	BLEU
UEDIN (Cettolo et al., 2014)	37.29
Tied Transformer (Xia et al., 2019)	40.51
Layer-wise Coordination (He et al., 2018)	40.50
Our NMT Systems	
Transformer	39.80
With Joint Attention Strategies	
Layer Aggregation (Iter-C-Agg)	40.31 (+0.51)
Multi-Layer Attention (M-00)	40.99 (+1.19)
Multi-Layer Supervision	
MS-1	40.60 (+0.8)
+ AIF	40.92 (+0.32)
MS-2	40.71 (+0.91)
+ AIF	40.83 (+0.12)
MS-3	40.65 (+0.85)
+ AIF	41.22 (+0.57)

Table 4.3: Evaluation of translation performance on the IWSLT Spanish-English (Es→En)

On the En→Vi tasks, the MS approaches consistently outperforms the standard Transformer across all possible configurations. Furthermore, each of the MS configurations outperforms all previously existing models (Luong and Manning, 2015; Huang et al., 2018). Among the MS models without the AIF module, the MS-1 achieved the best performance with a 31.32 BLEU score. Specifically, the MS-1 model achieves about $+0.74$ BLEU improvement over the standard Transformer. There is no significant change in translation quality for the MS-1 with the AIF module between the decoders (i.e. MS-1+AIF). In contrast, adding the AIF module further improves the performance of both MS-2 and MS-3 models. For example, the translation performance of MS-3 increases from 31.03 to 31.64, representing a further gain of $+0.61$ BLEU while the MS-2 with AIF gives a further $+0.39$ BLEU. The MS-1 and MS-1+AIF models consistently outperform the JASs models. On the other hand, training the MS-3 model without the AIF module produced lower performance than that achieved by the JASs model, M-01 and Iter-C-Agg. Similarly, the MS-2 achieve translation quality gain identical to that achieved by the best *Layer Aggregation* model.

Model	**BLEU**
8-Layer RNN (Wu et al., 2016)	26.30
ConvSeq2Seq (Gehring et al., 2017)	26.36
Transformer-Base (Vaswani et al., 2017)	27.31
Transformer+EM Routing (Dou et al., 2019)	28.81
Transformer+Layer Aggreagation (Dou et al., 2018)	28.78
Our NMT Systems	
Transformer	28.37
With Joint Attention Strategies (JASs)	
Layer Aggregation (Iter-C-Agg)	28.81 ($+0.44$)
Multi-Layer Attention (M-10)	29.08 ($+0.71$)
Multi-Layer Supervision	
MS-1	29.07 ($+0.70$)
+ AIF	29.32 ($+0.25$)
MS-2	29.02 ($+0.65$)
+ AIF	28.88 (-0.14)
MS-3	28.90 ($+0.53$)
+ AIF	28.97 ($+0.07$)
MS-4	29.16 ($+0.79$)
+ AIF	29.11 (-0.05)
MS-5	28.74 ($+0.37$)
+ AIF	29.14 ($+0.40$)

Table 4.4: Evaluation of translation performance on the WMT14 English-German (En→De).

However, with the AIF module, these models (i.e. MS-2+AIF and MS-3+AIF) outperform the JASs models producing much higher performance gains of $+0.94$ BLEU and $+1.06$ BLEU, respectively, over the Transformer baseline.

For the IWSLT'14 De→En and Es→En tasks, every MS configuration outperforms the Transformer baseline. Without the AIF module, MS-2 obtained the best performance on both De→En and Es→En tasks. Specifically, it achieves a gain of $+0.43$ BLEU and $+0.91$ BLEU on the De→En and Es→En tasks, respectively. Unlike the En→Vi task, all the

MS configurations achieved an improvement in performance with the addition of the AIF module. On the Es→En task, the MS-3+AIF achieved the overall best performance with the BLEU score of 41.22, about $+1.42$ improvement over the baseline model. In contrast, on the De→En task, there is not much difference in the BLEU score between the *MS-l+AIF* models with the MS-2+AIF model achieving the best performance. Compared to the JASs based models, the *MS-l* and *MS-l+AIF* models consistently outperform the *Layer Aggregation* and *Multi-Layer Attention* based models.

Table 4.4 summarizes the results on the En→De translation task. Similar to the observations on the IWSLT datasets, connecting an auxiliary decoder to a lower-level encoding layer produces a notable performance gain of up to $+0.79$. A further performance gain is achieved (in most cases) with the AIF module (i.e. *MS-l+AIF* models). For example, the MS-1 yields a $+0.70$ boost in the BLEU score and achieves a further improvement of $+0.25$ BLEU with the AIF module between the decoders. Compared to the performance of the $+0.77$ BLEU produced by the M-10, the *MS-l* and *MS-l+AIF* achieved performance gains up to $+0.95$ BLEU over the Transformer baseline.

In summary, as shown in Tables 4.1 to 4.4, multi-level supervision can further improve the generalization performance of the sequence generation model. Furthermore, consistent across all the translation tasks dataset, the performance gain of both the *MS-l* and *MS-l+AIF* models is dependent on the choice of lower-level encoding layers connected to the auxiliary decoders. This is further explored in Sections 4.5.2 and 4.5.3.

4.5 Analysis

Table 4.5 shows sample translations from *Main-Decoder* and *Aux-Decoder l* subnetworks for *MS-l* and *MS-l+AIF* models on the IWSLT translation tasks under consideration. Ideally at each decoding step t, we expect $y_t^m = y_t^{a_l} = Y_t$. However, as shown in Table 4.5, in most cases the the *Main-Decoder* and *Aux-Decoder l* generate different target sequences. The difference in target translation is attributed to each decoding subnetwork generating the target sentence based on source representations from different encoding layer. For example, the y^{a_l} of the MS-1 model is generated based output of the first encoder layer whilst the y^m is from the final encoding layer.

Source	The passion that the person has for her own growth is the most important thing .	
Target	Cái khát vọng của người phụ nữ có cho sự phát triển của bản thân là thứ quan trọng nhất .	
Baseline	Sự đam mê mà con người có cho sự tăng trưởng của cô là điều quan trọng nhất .	
Multi-Layer Supervision		
MS-1	y^m	Sự đam mê mà con người có cho sự phát triển của mình là điều quan trọng nhất .
	y^{a_l}	niềm đam mê mà con người có cho sự phát triển của chính mình là điều quan trọng nhất .
MS-1 + AIF	y^m	Niềm đam mê mà con người có cho sự phát triển của chính mình là điều quan trọng nhất .
	y^{a_l}	niềm đam mê mà người dân có cho sự phát triển của chính mình là điều quan trọng nhất .
MS-2	y^m	đam mê rằng người đó có sự phát triển của riêng mình là điều quan trọng nhất .
	y^{a_l}	Niềm đam mê rằng người đó có cho sự phát triển của chính mình là điều quan trọng nhất .
MS-2 + AIF	y^m	Sự đam mê mà con người có cho sự phát triển của riêng nó là điều quan trọng nhất .
	y^{a_l}	Sự đam mê mà người đó có cho sự phát triển của mình là điều quan trọng nhất .
MS-3	y^m	Niềm đam mê mà người đó có cho sự phát triển của riêng mình là điều quan trọng nhất .
	y^{a_l}	Sự đam mê mà người đó có cho sự phát triển của riêng mình là điều quan trọng nhất .
MS-3 + AIF	y^m	Niềm đam mê mà con người có cho sự phát triển của chính nó là điều quan trọng nhất .
	y^{a_l}	niềm đam mê mà người đó có cho sự phát triển của chính mình là điều quan trọng nhất .

(a)

Source	Este es un sol colocado con el origen , porque Japón está al este de China .	
Target	This is a sun placed with the origin , because Japan lies to the east of China .	
Baseline	This is a sun sitting with the origin , because Japan is in Eastern China .	
Multi-Layer Supervision		
MS-1	y^m	This is a sun placed with the origin , because Japan is in the east of China .
	y^{a_l}	This is a sun put in the origin , because Japan is east of China .
MS-1 + AIF	y^m	This is a sun that was put in the origin , because Japan is east of China .
	y^{a_l}	This is a sun put in the origin , because Japan is at the east of China .
MS-2	y^m	This is a sun sitting with the origin , because Japan is east of China .
	y^{a_l}	This is a sun placed with the origin , because Japan is east of China .
MS-2 + AIF	y^m	This is a sun placed with the origin , because Japan is east of China .
	y^{a_l}	This is a sun that was placed with the origin , because Japan is to East China .
MS-3	y^m	This is a sun sitting with the origin , because Japan is in east China .
	y^{a_l}	This is a sun sitting with the origin , because Japan is east of China .
MS-3 + AIF	y^m	This is a sun placed with the origin , because Japan is east of China .
	y^{a_l}	This is a sun put with the origin , because Japan is in the east of China .

(b)

Source	wenn jemand sie bitten würde , ihre markenidentität zu beschreiben , ihre markenpersönlichkeit , wie wäre sie ?	
Target	if somebody asked you to describe your brand identity , your brand personality , what would you be ?	
Baseline	if someone asked them to describe their brand identity , their brand character , how would it be ?	
Multi-Layer Supervision		
MS-1	y^m	if someone asked you to describe your branded identity , your brand personality , how about you ?
	y^{a_l}	if someone asked you to describe your brand identity , your brand personality , what would it be like ?
MS-1 + AIF	y^m	if somebody asked you to describe your branded identity , your brand personality , what would it be like ?
	y^{a_l}	if someone asked them to describe their brand identity , their brand personality , what would it be like ?
MS-2	y^m	if someone asked you to describe your brand identity , your brand personality , what would it be like ?
	y^{a_l}	if someone would ask you to describe your brand identity , your brand personality , how would it be ?
MS-2 + AIF	y^m	if someone asked you to describe your brand identity , your brand personality , what would it be ?
	y^{a_l}	if somebody asked you to describe your brand identity , your brand personality , what would it be ?
MS-3	y^m	if somebody asked you to describe your brand identity , your brand personality , how would it be ?
	y^{a_l}	if somebody asked you to describe your brand identity , your brand personality , how would it be ?
MS-3 + AIF	y^m	if someone asked you to describe your brand identity , your brand personality , what would it be like ?
	y^{a_l}	if someone asked them to describe their brand identity , their brand personality , what would it be like ?

(c)

Table 4.5: Sample translations from the baseline, *MS-l* and *MS-l+AIF* models on the (a) En→Vi task,(b) Es→En task and (c) De→En task. y^m and y^{a_l} denote the output translations from the *Main-Decoder* and *Aux-Decoder l* subnetworks, respectively.

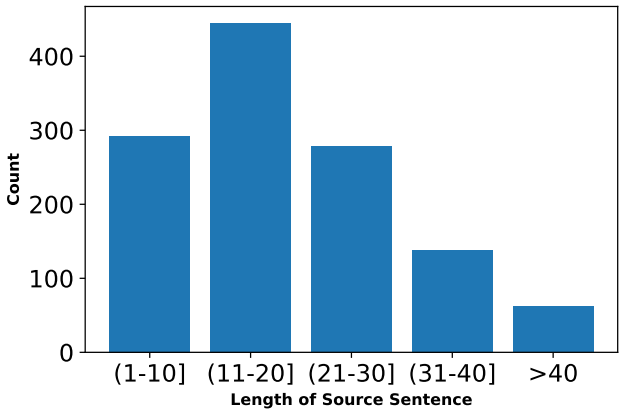


Figure 4.5: Distribution of the number of sentences from the En→Vi test set across the different sentence length groups.

This section presents analyses performed to better understand the impact of the proposed *Encoder-based* MS approaches. First, the impact of the source sequence length on the translation quality is analysed. The subsequent subsections are dedicated to analyses, including investigating the impact of encoder layer choice for the auxiliary decoder and the impact of the AIF module between the main and the auxiliary decoders. These analyses are performed based on the En→Vi translation task.

4.5.1 Length of Source Sentence

Following (Luong et al., 2015b; Zhang et al., 2018), sentences of similar length are grouped, and the BLEU score of the outputs from the MS model for each group is calculated. The choice of range for the grouping when evaluating the impact of sentence length is usually based on the distribution of the sentence lengths (the number of tokens in each source sentence) across the test set. On the En→Vi task, the minimum and maximum sentence lengths are two tokens and 102 tokens, respectively. However, about 60% of the sentences in the test set have sequence lengths between 2 and 20 tokens. Besides, the mean sentence length is 21.08 tokens. Therefore the comparison presented here is based on five-sentence length groups: 1-10, 11-20, 21-30, 31-40, and greater than 40 (>40). Figure 4.5 shows the distribution of the number of sentences across the different sentence length groups. Each plot in Figure 4.6 shows the variation in the performance per group for the Transformer baseline,

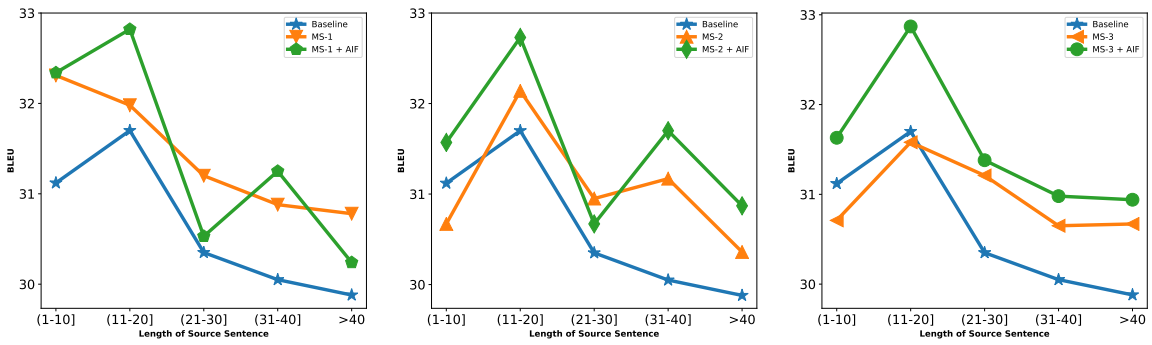


Figure 4.6: BLEU scores on the En→Vi task for the Transformer baseline model, the *MS-l* models and the *MS-l+AIF* models with respect to varying source sentence length. **Left**: Transformer baseline vs MS-1 vs MS-1+AIF . **Middle**: Transformer baseline vs MS-2 vs MS-2+AIF. **Right**: Transformer baseline vs MS-3 vs MS-3+AIF.

the *MS-l* and the *MS-l+AIF* models. As noted by Dou et al. (2018), to achieve higher translation performance on long sentences, the model should be able to capture effectively the contextual information and long-distance dependencies between the tokens.

As displayed in Figure 4.6, the multi-level supervision models consistently outperform the baseline model for longer sentences. There is also a considerable difference in translation performance among our models across the sentence groups. Unfortunately, similar to the baseline, the translation quality decreases significantly for sentences with length greater than 20 for most of the models, with the exception of the MS-2, MS-3+AIF, and MS-3 models, which have the least difference in BLEU scores for most sentences (of length >20). Overall, the higher performance of our models can be attributed to the enhancement the multi-level supervision introduces into the encoder subnetwork. The lower layers are able to capture both the local and global source contextual information to support the generation of the target sequence from all the connected decoders. Furthermore, the AIF module allows the decoders to share information on the output sequence based on the source contextual information processed by each subnetwork during the target generation. This explains the higher performance of *MS-l+AIF*'s *Main-Decoder* compared to that of the *MS-l* on longer sentences. This is true especially in the case of the MS-2 vs MS-2+AIF and MS-3 vs MS-3+AIF as shown in Figure 4.6.

#Aux-Decs	Model	#Params (M)	Train	Decode
0	Baseline	12.8	8.58	298.96
1	*MS-i*	17.0	5.50	310.46
	+AIF	17.3	5.42	184.67
2	*MS-ij*	21.2	3.90	310.46
	+AIF	21.7	3.77	141.10
3	*MS-ijk*	25.4	3.08	310.46
	+AIF	26.2	3.04	106.0

(a)

#Aux-Decs	Model	BLEU
0	Baseline	30.58
1	MS-1	31.32
	+AIF	31.29
	MS-2	31.13
	+AIF	31.52
	MS-3	31.03
	+AIF	31.64
2	MS-12	31.49
	+ AIF	31.47
	MS-13	31.30
	+ AIF	31.63
	MS-23	31.32
	+ AIF	31.28
3	MS-123	31.47
	+ AIF	31.70

(b)

Table 4.6: Encoder-based MS models with multiple auxiliary decoders on the IWSLT'15 En→Vi translation task. "**#Aux-Decs**" indicates the number of auxiliary decoders employed. (a) Complexities of the MS models: "**#Params**" denotes the number of trainable model parameters. "**Train**" and "**Decode**" respectively denote the training speed (steps per second) and decoding speed (tokens per second) on GTX Geforce GPU. (b) Impact on translation quality based on the choice of encoding layers.

4.5.2 Impact of choice of encoding layer and multiple auxiliary decoders

Learning a better structural representation of the source sentence can improve translation quality. As mentioned earlier, the lower encoding layers capture more word-level features of the source sentence, while top-level layers encode more semantic information needed by the *Main-Decoder* to successfully perform the target generation task. As shown in Tables 4.1 to 4.4, performing multi-level supervision with the auxiliary decoder connected to any of the lower or intermediate encoding layers can significantly improve the generalization performance of the *Main-Decoder* connected to the last encoder layer. Surprisingly, the performance gain of both the *MS-l* and *MS-l+AIF* models is dependent on the choice of encoding layer for the auxiliary decoder. In most cases, *MS-l* performing the auxiliary

decoding based on the source representation from the first few layers provides a better performance enhancement compared to connecting the auxiliary to the upper encoder layers.

MS models with multiple auxiliary decoders were trained on the En→Vi translation task to test the validity of these observations. The encoder employed is a 4-layer network, and all possible combinations of encoding layers were explored to train the *MS-l* and *MS-l+AIF* models. The results are summarised in Table 4.6b. The MS model with a single auxiliary decoder connected encoding layer i is denoted as *MS-i*. *MS-ij* is the MS model with the *Aux-Decoder i* and *Aux-Decoder j* connected to encoder layers i and j, respectively. Similarly *MS-ijk*, denotes the MS model with the *Aux-Decoder i*, *Aux-Decoder j* and *Aux-Decoder k* connected to encoder layers i, j and k, respectively.

Model Complexity

The performance gain achieved by the MS models comes at a higher computational complexity. This is mainly due to the new trainable parameters introduced by the auxiliary decoding subnetworks. Table 4.6a summarizes the training and decoding complexities of the MS on the En→Vi translation task. As noted by Popel and Bojar (2018), the computational complexity of any given neural model is affected by quantities such as the number of model parameters, the optimizer, and other explicit computations that directly modify the formulations of the network structure. For both *MS-l* and *MS-l+AIF* models, training with multiple decoding subnetworks significantly reduces the training speed as more effort is required to optimise the parameters of the overall translation network effectively. As mentioned in Section 4.2, the target sentence is generated from only the *Main-Decoder* for the *MS-l* models. Accordingly, the decoding speed of these models is identical to the Transformer baseline. However, the *MS-l+AIF* models have lower decoding speeds compared to their corresponding *MS-l* models. This can be attributed to the *Main-Decoder* leveraging the target representations from the auxiliary decoders. For example, there is about a 65% decrease in the decoding speed of the *MS-ijk* model when the AIF module is employed.

Translation Quality

As shown in Table 4.6b, connecting additional decoders (auxiliary decoders) is shown to further improve the translation performance of the *Main-Decoder*. For example, MS-12

achieved a higher BLEU score than the MS-2 and MS-1 models. Similarly, the MS-23 performs better than the MS-2 and MS-3 models. Among the MS models with two or more auxiliary decoders, both the MS-12 and the MS-123 achieved similar performance (BLEU score) higher than the MS-13 and MS-23 models. When the AIF module is employed, the MS-3, MS-13, and MS-123 models achieved the highest performance. The performance obtained on all the translation tasks motivates the hypothesis that decoding from the lower or intermediate layers forces these lower-level layers to capture not only the word-level representations but also to encode the source semantic information needed by the corresponding auxiliary decoder. This is attributed to the inductive bias effect on the encoder subnetwork, which further improves the learning of the deep linguistic information required by the *Main-Decoder* to generate the target sentence.

Overall, it is evident from Table 4.6 that it is not necessary to connect auxiliary decoders to all encoding layers to achieve higher performance. As shown in Table 4.6b, in most cases there is no significant difference in the translation performance between the *MS-i*, *MS-ij* and *MS-ijk* models with or without the AIF module. For example, the MS-12 model with fewer trainable parameters achieved identically similar performance as the MS-123 model. Similarly, there is a marginal difference between the BLEU scores achieved by the MS-123+AIF, the MS-3+AIF, and the MS-13+AIF models. Despite the increase in the computational complexity, the AIF module is shown to further improve the translation performance of the *Main-Decoder*. Base on the above observations, this chapter suggests using a single auxiliary decoder. When training without the AIF module, we recommend limiting the application of the auxiliary decoder to any of the first few lower-level layers. Generally, with AIF, the best performance gain is achieved when the auxiliary decoder is connected to one of the top-level encoding layers.

4.5.3 Impact of the performance of the *Aux-Decoder l*

For both *MS-l* and *MS-l+AIF* models, the *Aux-Decoder l* connected to top-level layers produced higher translation performance than those decoding from the lower-level layers as summarized in Table 4.7. For example, on the Es→En and De→En tasks, the performance of the *Aux-Decoder l* employed in the MS-3 setting is greater than the auxiliary decoder of the MS-1 and MS-2 models. Similarly, the *Aux-Decoder l* of MS-3 +AIF usually produced higher translation quality than the *Aux-Decoder l* of MS-1+AIF model. The difference in

Model	BLEU					
	En→Vi		De→En		Es→En	
	MD	AD	MD	AD	MD	AD
MS-1	31.32	30.63	34.87	34.05	40.60	39.71
+ AIF	31.29	30.60	35.12	33.93	40.92	39.61
MS-2	31.13	30.91	34.96	34.38	40.71	40.10
+ AIF	31.52	30.89	35.20	34.33	40.83	40.08
MS-3	31.08	30.88	34.83	34.77	40.65	40.26
+ AIF	31.64	31.24	35.14	34.62	41.22	39.73

Table 4.7: Comparison of the performance of the *Main-Decoder* (denoted as MD) and *Aux-Decoder l* (denoted by AD) subnetworks on the En→Vi, De→En, and Es→En translation tasks.

translation quality of the auxiliary decoders is attributed to the fact that the performance of any given decoding subnetwork is dependant on the quality of source representation passed to it from the encoder subnetwork. Each encoding layer extracts different levels of abstraction of the source sentence. As mentioned in Section 4.1, the top-level layers encode more refined semantic information, while the lower-level layer generally emphasises on learning the lexical and morphological source information.

As shown in Section 4.4, Tables 4.6 and 4.7, the performance of the *Main-Decoder* is affected by the choice of encoding layer connected to the *Aux-Decoder l* subnetwork. A natural question here is: "Is there any relationship between the performance of the *Main-Decoder* and *Aux-Decoder l* subnetworks?". Under the *MS-l* models, the performance of the *Main-Decoder* is shown to be independent of the translation quality of the *Aux-Decoder l* subnetwork. For example, the auxiliary decoder of the MS-3 model performed better than that of the MS-1 model across the languages under consideration. However, the *Main-Decoder* of the MS-1 outperforms that of the MS-3. For the *MS-l* models, the performance is affected by the quality of source representation employed by the *Main-Decoder* to generate the target translations. As demonstrated in Table 4.8, the encoder subnetworks of MS-1 and MS-2 models produced source representation of higher quality than that of the MS-3 model. Unlike the *MS-l* model, the *Main-Decoder* of the *MS-l+AIF* model leverages the

target representations from the auxiliary decoders via the AIF module. Accordingly, for these models, the translation quality of the *Main-Decoder* subnetwork is affected by the target generation ability of the *Aux-Decoder l* subnetworks, as demonstrated in Table 4.7.

4.5.4 Linguistic Probing

The translation performance of the MS-based models is attributed to the enhancement of the source representation ability of the encoder subnetwork. However, little is known about the linguistic perspectives or properties enhanced by the *Encoder-based* MS strategy. Following (Conneau et al., 2018; Li et al., 2019), ten classification tasks are conducted to study the linguistic properties improved by the MS training strategy. These linguistic probing tasks are divided into three categories:

Surface (Surf): focuses on the surface-level features captured by the sentence representation or embedding. This category consists of the *Sentence Length* (SentLen) and *Word Content* (WC) classification tasks. Under the SentLen task, the aim is to predict the length of the input sentence based on the number of the constituent tokens. The WC task tests the possibility of recovering information about the original word in a sentence given its embedding.

Syntactic (Sync): evaluate the ability of the encoder subnetwork on learning or capturing the syntactic information. Three syntactic tasks are under consideration: *Tree Depth* (TDep), *Bigram Shift* (BShift) and *Top Constituent* (ToCo). The TDep task tests the capability of the encoder at inferring the hierarchical structure of the input sentence. For BShift, the sensitivity of the encoder to the legal word order is evaluated. Specifically, the goal is to predict if two consecutive tokens within the sentence have been inverted or not. Finally, under the ToCo task, the sentence is classified in terms of the sequence of top constituents immediately below the sentence node.

Semantic (Sem): evaluate the capabilities of the encoder on understanding the denotation of a given sentence. The sentence embedding should encapsulate the syntactic structure to achieve higher performance on this task (Conneau et al., 2018). Here, there are five sub-tasks under consideration. The first semantic probing task is the *Tense* classification task, which evaluates the tense of the main clause verb

Tasks \ Models		Transformer	MS-1	MS-2	MS-3
Surface	SentLen	87.63	**88.65**	88.35	87.88
	WC	61.7	61.17	60.38	**62.50**
	#AVG	74.67	74.91	74.37	**75.19**
Syntactic	TDep	34.63	**35.67**	34.78	34.65
	BShift	54.53	**56.58**	56.20	55.60
	ToCo	63.25	**65.45**	**65.44**	63.12
	#AVG	50.80	**52.57**	52.14	51.12
Semantic	Tense	78.88	**79.22**	78.52	78.48
	SubjN	74.86	**75.03**	74.98	74.21
	ObjN	75.80	74.79	**76.20**	75.19
	SoMo	**51.67**	51.47	51.38	51.4
	CoIn	58.07	**58.31**	58.14	57.34
	#AVG	**67.86**	67.77	**67.84**	67.32

Table 4.8: Performance on the 10 probing tasks to evaluating the linguistic information ("Surface", "Syntactic" and "Semantic") learned by the encoding subnetwork of the Transformer baseline and our *MS-l* models. **#AVG** denotes the mean across each category

(whether it is in the past or present tense). The next is the *Subject Number* (SubjN), which focuses on predicting the number of the subject in the main clause. In contrast, the *Object Number* (ObjN) predicts the number of direct object of the main clause. Under the *Coordination Inversion* (CoIn), the sentences are divided into two coordinate clauses. In half of the sentences, the order of the clauses is inverted and the task here is to check if a sentence is modified or intact. Finally is the *Semantic odd man out* (SoMo) task, where the sentences are modified by randomly replacing a noun or verb with another noun or verb. Here, the task is to predict whether a sentence has been modified or not.

Experimental Settings: These analyses are performed based on the source representation from only the encoder subnetworks of the pre-trained NMT models presented in this chapter. Each probing task is performed using a two-layer MLP classifier on top of the pre-trained encoder subnetwork. $L2$ regularisation of $\lambda = 1e^{-4}$ is applied across the hidden layer of classifiers. For each probing task, the input representation to the classifier is the mean of the sentence representation from the final encoder layer. The resulting classification models are evaluated on the dataset[6] presented by Conneau et al. (2018). The training, validation, and test sets for each probing task comprises 100k sentences, 10k sentences, and 10k sentences, respectively. The probing models are trained for 100 epochs using RMSProp optimizer with the learning rate of $1e^{-3}$ and min-batch size of 512. The training early-stops based on the accuracy on the validation set. During training, the parameters of the pre-trained encoder subnetwork are fixed to quantify the linguistic properties and information captured by the pre-trained encoder subnetworks of the Transformer baseline and our *Encoder-based* MS models. Therefore, only the parameters of the classifier are updated. To better quantify the performance impact on the encoder subnetwork, the analyses presented are limited to only the *MS-l* models.

Results: Table 4.8 summarizes the performance of the Transformer baseline and the *MS-l* models on the test sets of linguistic probing tasks. As shown, training the NMT model with an auxiliary decoder connected to any of the lower-level layers affects the source representation ability of the encoder subnetwork. The MS-1 model achieved the best performance over the baseline model across 7 out of the 10 probing tasks. On average, there is a marginal difference between the performance of the *MS-l* models (MS-1 and MS-2) and the baseline on semantic tasks. However, the most notable difference between the performance of *MS-l* models and the baseline is across the Surface and Syntactic tasks. We argue that multi-level supervision strategy allows the encoder subnetwork to effectively learn deeper linguistic information (Semantic and Syntactic) as well as the superficial information to further improve the translation performance. Among the *MS-l* models, the source representations from the MS-1 and MS-2 models showed the overall best performance over the baseline model across all the linguistic probing tasks. This observation supports our recommendation of limiting the auxiliary decoder connections to only the first few lower-level

[6]https://github.com/facebookresearch/SentEval/tree/master/data/probing

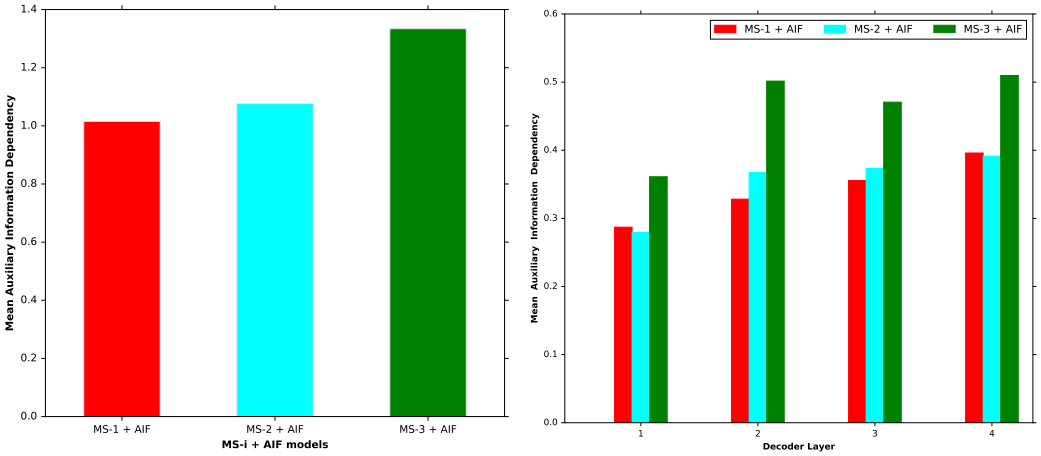
encoding layers.

4.5.5 Impact of the AIF: Auxiliary Context Dependency

For the *MS-l+AIF* models, the *Main-Decoder* leverages representations learned by the connected auxiliary decoders during both the training and inference phases via the AIF module. This approach as shown in most cases further enhances the translation quality of the *Main-Decoder*. We hypothesise that, if the *Main-Decoder* properly utilises representations from the auxiliary decoders, then there should be a significant difference between its output distributions with and without the auxiliary information processed by the AIF module. To this end, we propose to measure the auxiliary information dependency of the *Main-Decoder* via a distribution based distance metric. This will help quantify the impact of information sharing between the decoders. After the model training, an "auxiliary" independent decoder called *ind-decoder* is created, based on the network structure of the *Main-Decoder*. The *ind-decoder* shares the same parameters with the *Main-Decoder*, except that the weights $\{W_l^{a_1}, W_l^{a_2}, \cdots, W_l^{a_n}\}$ within the AIF module in each decoding layer are set to zero. As shown in Equations (4.2) and (4.3), zeroing out the weights within the AIF module implies the input to the AIF module is the same as its output. For every inference step, we set the *ind-decoder*'s previously generated target word embedding to that of the *Main-Decoder*. At decoding step t, two target distributions for generating output token y_t, $\mathsf{P}^m(y_t)$ and $\mathsf{P}^{ind}(y_t)$ are produced by the *Main-Decoder* and *ind-decoder*, respectively. Based on these two distributions, the auxiliary information dependency (D_{y_t}) with respect to the target token y_t is measure via KL-divergence, as shown below:

$$D_{y_t} = D_{KL}(\mathsf{P}^m(y_t) \;||\; \mathsf{P}^{ind}(y_t)) \tag{4.4}$$

where $\mathsf{P}^m(y_t) = \mathsf{P}(y_t^m | y_{<t}^m, x; \theta^m)$ denotes the distribution from the *Main-Decoder* with the auxiliary information and $\mathsf{P}^{ind}(y_t) = \mathsf{P}(y_t^{ind} | y_{<t}^m, x; \theta^{ind})$ denotes the distribution from the *ind-decoder* (i.e the *Main-Decoder* without the auxiliary information from the auxiliary decoder). $y_{<t}^m$ denotes the previously generated target token from the *Main-Decoder*. The



(a) (b)

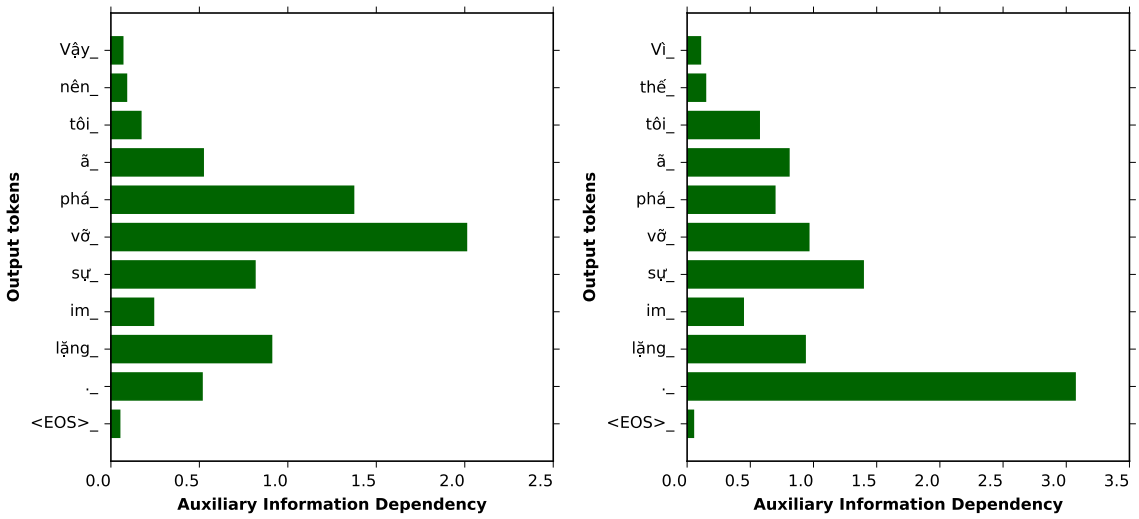
Figure 4.7: Means of AIF Dependency across the layers within the *MS-l+AIF* models and with respect to each model's output generation. (a) Mean Auxiliary Information Dependency per model based on the output distributions. (b) Mean Auxiliary Information Dependency per decoding layer for the *MS-l+AIF* models based on the multi-head attention distributions.

auxiliary information dependency across the entire output sequence is defined as:

$$D_Y = \frac{1}{|Y|} \sum_{t=1}^{|Y|} D_{y_t} \tag{4.5}$$

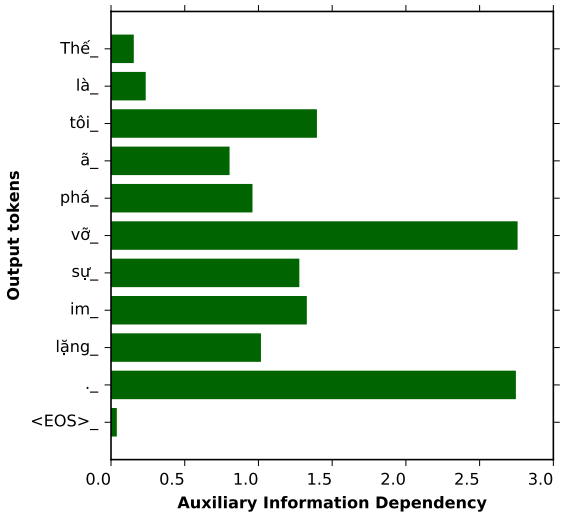
As shown in Figure 4.4, the auxiliary features are combined with the *Main-Decoder*'s internal representation by the AIF module before the multi-head attention computation on the output of the encoder subnetwork is performed. Therefore, we can also measure the auxiliary dependency for each attention head with respect to each layer to further examine the impact of the information passed from the connected auxiliary decoder. This is done by computing the divergence between the attention head weight distributions generated by the *Main-Decoder* and the *ind-decoder* for each layer. For the attention head h in the decoding layer l, the dependency is defined as:

$$D_h^l = D_{KL}(A_h^m \mid\mid A_h^{ind}) \tag{4.6}$$



(a) MS-1+AIF

(b) MS-2+AIF



(c) MS-3+AIF

Figure 4.8: Auxiliary Information dependency of the MS-1+AIF (a), MS-2+AIF (b) and MS-3+AIF (c) models for the En→Vi translation of the English sentence "So I broke the silence.". "<EOS >_" is the end token symbol that signals the end of the output generation.

where $A_h^m \in \mathbb{R}^{Z \times J}$ and $A_h^{ind} \in \mathbb{R}^{Z \times J}$ are the weight distributions for the attention head h generated by the *Main-Decoder* and the *ind-decoder*, respectively. Z and J denote the length of the target and source sentences, respectively. D_h^l is the KL-divergence between

A_h^m and A_h^{ind}. The overall auxiliary dependency D^l for the decoding layer l is:

$$D^l = \frac{1}{N_h} \sum_{h=1}^{N_h} D_h^l \tag{4.7}$$

where N_h is the number of attention heads employed by the multi-head attention sublayer to extract the source-target contextual information in each layer. For simplicity, these analyses are performed using the greedy search algorithm to generate the target distributions.

The analysis of the impact of the AIF module is performed using 1k sentence pairs randomly selected from the En→Vi test-set split. The average input sequence length after the subword tokenization is 26.3. Figure 4.7a shows the mean auxiliary information dependency per the models based on the output distributions. As displayed, there is a fairly high dependency on the auxiliary information. The MS-1+AIF model has one of the lowest dependency on the feature information produced by the auxiliary decoder (connected to the encoding layer 1) meaning that on average, the performance of the *Main-Decoder* is not that significantly affected by the auxiliary information. This might explain the closeness of the translation quality obtained by the MS-1 and MS-1+AIF models as shown in Table 4.1. On the other hand, for the MS-2+AIF and MS-3+AIF models, there is a significant difference in their output distributions with and without the extra information generated by the auxiliary decoder. Among all the *MS-l+AIF* models, the MS-3+AIF showed the highest auxiliary information dependency. This implies that at each decoding step t, the *Main-Decoder* highly considers the supplied auxiliary information in the generation of the output tokens. Figure 4.8 shows an example of auxiliary information dependency measurement for the translation of the English sentence "So I broke the silence." to Vietnamese for each *MS-l+AIF* model. As shown, each model successfully generates the correct translation of the source sentence. While each model produces a different translation for the input token "So" (for example, MS-3+AIF generates the sub-string "Thế là" , and MS-1+AIF generates "Vậy nên"), they all produce the same translation for remaining source sub-string. But under each model, the tokens corresponding to the translation of "I broke the silence." have different dependency scores. During the generation of each token in the target sub-string "tôi đã phá vỡ sự im lặng" , the MS-3+AIF has relatively higher dependency scores compared to the MS-1+AIF and MS-2+AIF models.

On average, there is also a significant difference in multi-head attention weights across

the layers within the *Main-Decoder* for each *MS-l+AIF* model, as shown in Figure 4.7b. For each model, the dependency on the auxiliary information is very high in the upper decoding layers. Consistent with Figure 4.7a, the MS-3+AIF model showed the highest mean auxiliary information dependency across its layers, with the final layer reporting the significant difference in the attention distribution for each attention head with and without the auxiliary information.

4.6 Summary

In this chapter, we improved the performance of the encoder subnetwork by attaching decoders to top-level layers as well as some of the lower-level encoding layers. Specifically, the encoding subnetwork is trained to learn the necessary source semantic to improve the performance of *Main-Decoder* connected to the final layer and the auxiliary decoders connected to the lower-level layers. This presents a form of inductive bias to further enhance the source representation ability of the encoder. Evaluations on the state-of-the-art Transformer architecture shows that the *Encoder-based* MS achieves competitive performance on four language translation tasks. The performance gain is attributed to the impact of the *Encoder-based* MS on the source representation ability of the encoder subnetwork. This conclusion is supported by the linguistic probing analysis performed, which showed that training the NMT model via the MS encourages the encoder subnetwork to learn the surface-level and the deeper linguistic features required for the translation task. The improvement in the translation quality, via MS, is dependent on the choice of a lower-level encoding layer providing source representation to the auxiliary decoder. Besides, our evaluations on the *MS-l* models with AIF (i.e. *MS-l+AIF*) also show that sharing information between the decoders via the AIF module (in most cases) significantly improves the performance of the *Main-Decoder*.

The following chapter presents the *Dual Contextual* module, an extension to the self-attention unit, to leverage both the global and local contextual representation to further improve the sentence representational ability of the encoding and decoding subnetworks.

Part III

Contextual Modelling

Chapter 5

Dual Contextual Modelling for Neural Machine Translation

To effectively learn the semantic representations of the input sentence pairs, encoder-decoder frameworks such as the Transformer network (Vaswani et al., 2017) employs the self-attention unit. Recent research (Yang et al., 2018; Xu et al., 2019; Sperber et al., 2018) suggest the conventional self-attention unit tends to focus more on learning the global contextual representations with less emphasis on the local contextual information. However, to achieve higher performance on the translation task, the decoding subnetwork requires rich sentence representations encapsulating both the global and local contextual information. Therefore to further enhance the translation quality, this chapter presents our approach to extending the self-attention unit to leverage both the local and global contextual information.

5.1 Introduction

Leveraging the contextual information has been shown to significantly enhance the translation performance of both statistical-based and neural-based MT models (Marton and Resnik, 2008; He et al., 2008; Gimpel and Smith, 2008; Marvin and Koehn, 2018). Neural

models exploit token contexts usually via the attention mechanism. Self-Attention Networks (SAN) (Lin et al., 2017; Parikh et al., 2016) have been shown to significantly enhance the performance of neural models for natural language processing (NLP) tasks including document summarization (Al-Sabahi et al., 2018; Wang and Ren, 2018), acoustic modelling (Sperber et al., 2018), Neural Machine Translation (NMT) (Gehring et al., 2017; Vaswani et al., 2017; Dou et al., 2019), and reading comprehension (Yu et al., 2018a). The state-of-the-art NMT model, the Transformer network (Vaswani et al., 2017), is one of the prominent non-RNN models based entirely on self-attention. One of the salient strengths of the SAN is the ability to capture the long-range dependencies (global information) between the tokens within a given sentence by attending to all tokens present irrespective of their distance (Yang et al., 2019a; Sperber et al., 2018). That is, to generate the contextual representation for a token, self-attention tends to consider all tokens in the sentence. The implication of this approach is that there is a higher probability that the self-attention mechanism may overlook the relations between the neighbouring tokens (Yang et al., 2019a; Sperber et al., 2018). In summary, SAN tends to concentrate on modelling the global contextual information with less emphasis on capturing important short-range dependencies between the tokens.

Recently, there has been a growing number of research works dedicated to improving the performance of SAN at capturing both the short and long-range dependencies. For example, (Sperber et al., 2018) applied a locality restriction approach to limit the attention scope or span of the attention mechanism to the neighbouring elements to further enhance performance on the task of acoustic modelling. Similarly, (Yang et al., 2018) employed a learnable Gaussian bias to model localness by revising the attention weight distribution to focus more on a dynamically varying window of tokens. In a different direction, other works (Yang et al., 2019b; Wu et al., 2018) explored convolutional concepts to restricting the attention span to a fixed size window. Even though these approaches enhance the sentence representation ability of the SAN, Xu et al. (2019) argues that restricting the attention scope to some extent limits the ability of the self-attention mechanism to learn the global contextual information effectively.

An interesting question here is *how to exploit the local contextual information together with the global contextual information effectively, without restricting the attention scope of the self-attention mechanism.* To this end, this chapter presents the *Dual Contextual* (DC)

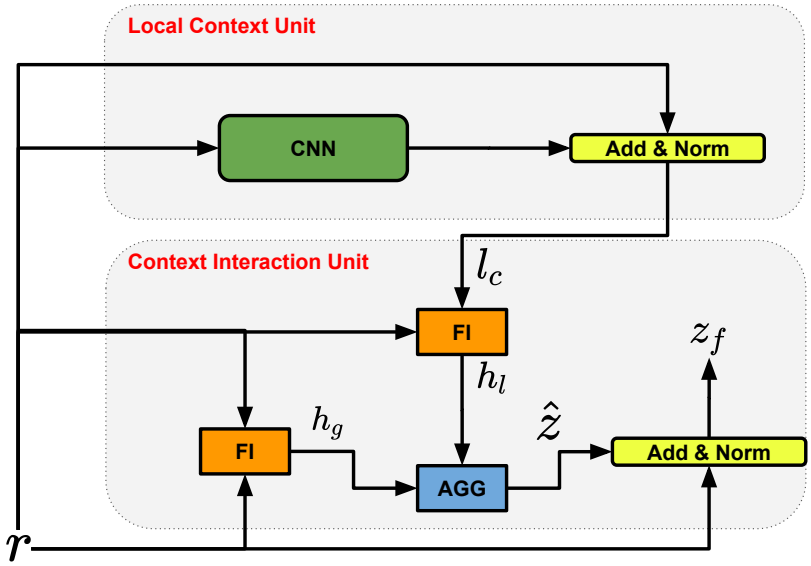


Figure 5.1: The proposed *DC* module employed to leverage the local and global information. z_f is the context rich representation generated based on the input sentence representation r. l_c is the contextual representation generated by the Local Contextual Unit and *AGG* denotes the aggregation unit employed to combine the hidden representations h_l and h_g generated by Feature Interaction (FI) units.

module, an extension to the SAN to leveraging both the local and global contextual information to further enhance the translation quality. The *DC* module shown in Figure 5.1 comprises two sub-modules, namely *Local Contextual Unit* and *Context Interaction Unit*. The Local Contextual Unit is a CNN-based network employed to capture the local contextual information respective to a neighbourhood window determined by the convolution filter size. The generated sentence representation from the Local Contextual Unit is passed to the Context Interaction Unit which employs two multi-head attention based units and an aggregation unit to model the interaction between the global and local contextual information. The proposed approach imposes no restrictions on the attention scope, allowing the self-attention to fully capture the long-range dependencies. The learning of the short-range dependency information is assigned to the Local Contextual Unit. One of the multi-head attention units in the Context Interaction Unit is employed to perform the self-attention computation across the input representation. The *DC* module is integrated into the Transformer network. Evaluations and analysis on nine language translation directions show that

the proposed *DC* module can significantly enhance the overall translation quality of the NMT model. Furthermore, analysis based on ten linguistic probing tasks (Conneau et al., 2018) performed demonstrates that leveraging both global and local contextual information further enhances the model's ability to effectively capture the surface, syntactic and semantic features required for the source translation task. The contributions of this chapter are:

- Proposing the *Dual Contextual* module to leverage both the local and global contextual information to further improve the translation performance.
- Demonstrating consistent improvement over the strong Transformer baseline across nine language translation tasks.
- Providing analysis on the performance impact of limiting the application of the *DC* module to the layers of either the encoder subnetwork or the decoding subnetwork.
- Providing an ablation study and analysis on the impact of limiting the contextual modelling via the *DC* module to only some combinations of encoding layers.

The remainder of the chapter is organised as follows: Section 5.2 briefly introduces the self-attention mechanism. The proposed *Dual Contextual* module is presented in Section 5.3. Following that, the experiments performed are presented in Section 5.4, and the results are compared and discussed in Section 5.5. Section 5.6 presents a detailed investigation into the impact of *DC* module on the translation performance of the Transformer network. Finally, the conclusion is presented in Section 5.7.

5.2 Background

Self-Attention Mechanism

An attention mechanism aims at modelling the direct relations between tokens of a given pair of sequence representations. Formally, the attention mechanism generates the token-to-token contextual representation c:

$$\begin{aligned} c &= \mathrm{ATT}\left(Q, K, V\right) \\ \mathrm{ATT}\left(Q, K, V\right) &= \alpha \cdot V \\ \alpha &= \text{softmax}\left(\mathrm{score}(Q, K)\right) \end{aligned} \tag{5.1}$$

where $Q \in \mathbb{R}^{Z \times d_{model}}$, $K \in \mathbb{R}^{J \times d_{model}}$, and $V \in \mathbb{R}^{J \times d_{model}}$ are the query, key and value vectors respectively. α is the attention weight distribution computed across the tokens represented by K. Z and J are the number of tokens in the given sequence representations Q and K respectively. d_{model} denotes dimension of the hidden representation. Finally, $\mathrm{score}(\cdot)$ is a scaled dot product function defined as:

$$\mathrm{score}(Q, K) = \frac{Q \times K^{\intercal}}{\sqrt{d_k}}$$

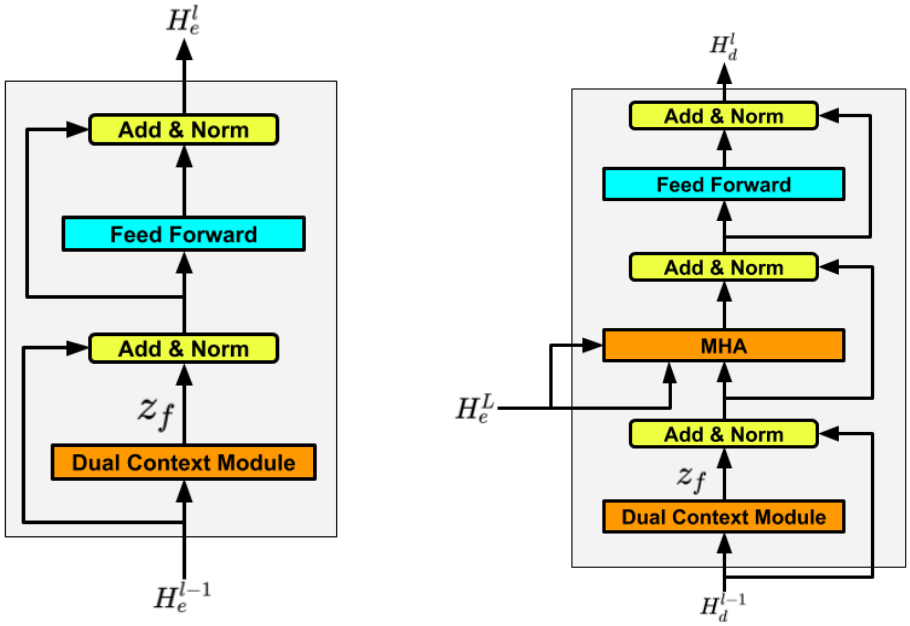
where $\sqrt{d_k}$ is a scaling factor employed to stabilise the attention computation.

Self-attention, a variant of the attention mechanism, performs the attention computation across a single sentence representation. Specifically, self-attention models the long-range dependencies between the tokens within the input sequence. Recent works including (Vaswani et al., 2017; Gehring et al., 2017; Shaw et al., 2018; Yu et al., 2018a) have shown the potential performance gain of self-attention mechanism over RNN at capturing the long-range contextual information and dependencies between the pairs of tokens within the given sequence.

For the layer[1] l, the self-attention unit computes the sentence representation h^l by attending to the hidden representation H^{l-1} from the preceding layer[2] $(l-1)$. The first stage of the self-attention unit is the computation of the query (Q), value (V) and key (K) based

[1] A layer of either the encoder or decoding subnetwork.

[2] For $l = 0$, H^{l-1} is the output of the embedding layer.



(a) (b)

Figure 5.2: Illustration of the application of the proposed *DC* module to (a) the encoder layer and (b) the decoder layer. H_d^{l-1} and H_d^l denote the output representations from the decoding layers $l-1$ and l respectively. Similarly, H_e^{l-1} and H_e^l are the input and output representations of the encoder layer l. H_e^L is the output of the final encoding layer passed to the encoder-decoder MHA sublayer of the decoding layer.

on three separate projections of the H^{l-1}:

$$\begin{aligned} Q &= H^{l-1}W^Q \in \mathbb{R}^{J\times d_{model}} \\ V &= H^{l-1}W^V \in \mathbb{R}^{J\times d_{model}} \\ K &= H^{l-1}W^K \in \mathbb{R}^{J\times d_{model}} \end{aligned} \tag{5.2}$$

where $\{W^V, W^Q, W^K\} \in \mathbb{R}^{d_{model}\times d_{model}}$ are the projection weights employed to generate the value, query and key vectors respectively. The self-attention unit employed by the Transformer model is based on the multi-head attention (MHA) mechanism. Specifically, the MHA defines N_h attention heads where each $head_i$ generates a separate attention weight distribution α^i. The attention head $head_i$ attends to tokens at different positions based on the α^i when generating the contextual representation $c^i \in \mathbb{R}^{J\times \frac{d_{model}}{N_h}}$. The MHA

is formulated as follow:

$$\begin{aligned} h^l &= \mathrm{MHA}\left(Q, K, V\right) \\ \mathrm{MHA}\left(Q, K, V\right) &= O_c W_o \\ O_c &= \mathrm{Concat}(c^1, c^2, \cdots, c^{N_h}) \\ c^i &= \mathrm{ATT}(Q^i, K^i, V^i) \end{aligned} \tag{5.3}$$

where $W_o \in \mathbb{R}^{d_{model} \times d_{model}}$ is a trainable weight parameter employed to generate the output representation $h^l \in \mathbb{R}^{J \times d_{model}}$ based on concatenation of the contextual representations from all the attention heads O_c. $\{Q^i, K^i, V^i\} \in \mathbb{R}^{J \times \frac{d_{model}}{N_h}}$ are the query, key and value vectors with respect to the i^{th} attention head respectively.

5.3 Dual Contextual (DC) Module

Our approach seeks to further improve the translation performance by leveraging both the local and the global contextual information to enhance the sentence representation ability of the encoding and decoding subnetworks. To this end, for a given layer (of either encoder or decoder subnetwork), the self-attention unit is replaced with a *DC* module as illustrated in Figure 5.2. As shown in Figure 5.1, the *DC* module consists of two sub-modules, namely the Local Contextual Unit and the Context Interaction Unit. Layer normalization and residual connections are applied across the output of each sub-module to further enhance the flow of gradient information. For the encoder layer *l*, the input sentence representation to the *DC* module (r) denotes the output of the $(l-1)^{\text{th}}$ encoder layer H_e^{l-1}. Similarly, for the decoding layer *l*, r is the output of the preceding layer $l-1$ (i.e. H_d^{l-1}).

Local Context Unit

To capture the local contextual information, this unit employs a single layer one-dimensional (1-D) convolution network with Gated Linear Unit (GLU) activation (Dauphin et al., 2017). The 1-D convolution can learn the local dependencies between the source tokens within a neighbourhood of width determined by the kernel size of the filter (Yu et al., 2018a; Song

et al., 2018). The local contextual representation l_c is generated as:

$$\begin{aligned} l_c &= \text{LayerNorm}(\hat{r} + r) \\ \hat{r} &= \text{Concat}(\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_J) \\ \hat{r}_t &= \text{g}\left([r_{t-f/2}, \cdots, r_{t+f/2}]W^r + b^r\right) \end{aligned} \tag{5.4}$$

where $W^r \in \mathbb{R}^{f \cdot d_{model} \times 2 \cdot d_{model}}$ and f are the convolution filter and kernel size respectively. b^r is the bias and $\text{g}(\cdot)$ is the GLU activation. $\hat{r}_t$ is the local contextual representation of the t^{th} token in the sequence. As shown in the above equation $\hat{r}$ is generated from the concatenation of the local contextual representations with respect to all input tokens.

Context Interaction Unit

This is the core of the *DC* module computing the context rich representation z_f based on the l_c from the Local Contextual Unit and r. Given l_c and r, the Context Interaction Unit generates the hidden representations h_l and h_g via two *Feature Interaction* (FI) units as shown in Figure 5.1. Each FI employs multi-head attention mechanism to model the interactions between the unit's inputs:

$$\begin{aligned} h_l &= \text{FI}(r, l_c) \\ h_g &= \text{FI}(r, r) \\ \text{FI}(A, B) &= \text{Concat}(M_1, M_2, \cdots, M_{N_h}) \\ M_i &= \text{ATT}(AW_i^q, BW_i^k, BW_i^v) \end{aligned} \tag{5.5}$$

where $AW_i^q, BW_i^k, BW_i^v \in \mathbb{R}^{J \times \frac{d_{model}}{N_h}}$ are the projections of the query, key and value vectors with respect to the attention head $head_i$ sub-space respectively. As shown in Figure 5.1, h_g is the global contextual representation obtained from a self-attention operation across the r. The h_l is sentence representation generated from the attention operation between r and the l_c (from the Local Contextual Unit). A feature aggregation unit, AGG, is employed to generate the hidden representation $\hat{z}$ from the combination of the outputs of the FI units:

$$\hat{z} = [h_l; h_g]W_z + b_z \tag{5.6}$$

where $W_z \in \mathbb{R}^{2 \cdot d_{model} \times d_{model}}$ and $b_z \in \mathbb{R}^{d_{model}}$ are trainable model parameters. $[\cdot;\cdot]$ denotes the concatenation operation. Formally, the Context Interaction Unit computes the z_f as:

$$z_f = \text{LayerNorm}\left(\hat{z} + r\right) \tag{5.7}$$

As illustrated in Figure 5.2, the generated z_f is passed to the subsequent sublayers for further processing. In case of the encoder subnetwork, it fed into the position-wise feed-forward network. For the decoder subnetwork, it is one of the inputs to the encoder-decoder MHA unit.

5.4 Experimental Setup

5.4.1 Datasets

The effectiveness of the proposed approach is evaluated on WMT'14 English-German (En→De) and eight IWSLT tasks: Spanish-English (Es↔En), English-Vietnamese (En↔Vi), English-French (En↔Fr) and Romanian-English (Ro↔En) translation tasks. The datasets for the Es↔En, En↔Vi and En→De translation tasks were introduced in Section 3.3.

Similar to the Es↔En tasks, the dataset for the Ro↔En tasks is from the Romanian-English translation track of the IWSLT 2014 evaluation campaign[3] (Cettolo et al., 2014). The training sets for these tasks consist of 182k sentence pairs, the test set is the tst2014 split, and the validation set is created by concatenating the tst2010, tst2011, tst2012 and dev2010. For the En↔Fr task, the dataset consisting of 207k training sentence pairs is from the IWSLT 2015 campaign. The test set is from the concatenation of the tst2014 and tst2015 splits. For each translation task, subword -based shared vocabulary is employed to encode the sentence pairs (source and target sentences)[4]. The vocabularies for the En→De, Es↔En, En↔Vi, Ro↔En and En↔Fr translation tasks consist of 32k, 34k, 21k, 32k and 31k subword tokens respectively.

[3]https://wit3.fbk.eu/mt.php?release=2014-01

[4]The original casing for the tokens in each sentence is preserved.

Task	L	N_h	d_{model}	d_{ff}	P_{drop}
En→De	6	8	512	2048	0.1
{Vi, Es, Fr, Ro}↔ En	4	4	256	512	0.1

Table 5.1: Model hyperparameters: L, N_h, d_{model}, d_{ff} and P_{drop} respectively denote the number of layers, number of attention heads, the hidden size, filter of dimension of the FFN sublayer and the dropout rate.

5.4.2 Model Settings, Training and Inference

The proposed *DC* module is integrated into Transformer network (Vaswani et al., 2017). The parameter settings such as the number of encoder and decoder layers (L), number of attention heads (N_h), FFN filter size (d_{ff}) and hidden size (d_{model}) employed for each translation task are chosen based on the number of sentence-pairs within the training set. These hyperparameters are summarised in Table 5.1. For the IWSLT translation tasks, the models are trained with a batch size of 2048 tokens and the number of training iterations is 200k. The batch size and the number of iterations employed to train the model on the En→De are 4960 tokens, 160k iterations respectively. Adam (Kingma and Ba, 2014) (with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^9$) is used as the optimizer to train the models. Following the works presented in Chapters 3 and 4, the learning rate scheduling algorithm employed in this chapter is the *single-cosine-cycle* with warm-up (So et al., 2019).

During inference, the target translations are generated using the beam search algorithm. For the En→De task, a beam-size of 4 and a length penalty of 0.6 is employed. Finally, for the IWSLT tasks, the beam size and length penalty are 6 and 1.1, respectively. The proposed *DC* module can be applied to both subnetworks of the Transformer model. Furthermore, it can also be limited to either encoder or decoder subnetwork. For simplicity, the Transformer model trained with the *DC* module applied to only the encoder is denoted as Enc-DC. Dec-DC is the model trained with the *DC* module employed within only the decoder subnetwork. Finally, the model trained with the *DC* module applied to both the encoder and decoder subnetworks is denoted as Full-DC.

The BLEU (Papineni et al., 2002) metric is employed to evaluate the translation quality. Specifically, the 4-gram case-sensitive BLEU metric computed with mteval-v13a.pl[5]

[5]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/mteval-v13a.pl

Model	#Params (M)	Train	BLEU
Transformer	61.2	3.65	28.37
With Joint Attention Strategies			
Layer Aggregation (Iter-C-Agg)	77.0	3.11	28.81 (+0.44)†
Multi-Layer Attention (M-10)	92.7	2.59	29.08 (+0.71)‡
With Multi-Layer Supervision			
MS-l (MS-4)	86.5	2.20	29.16 (+0.79)‡
MS-l+AIF (MS-1+AIF)	88.0	2.14	29.32 (+0.95)‡
With Dual Contextual			
Enc-DC	73.9	3.01	29.26 (+0.89)‡
Dec-DC	73.9	3.08	28.86 (+0.49)†
Full-DC	86.5	2.53	29.11 (+0.74)‡
Existing NMT Systems			
8-Layer RNN (Wu et al., 2016)	-	-	26.30
ConvSeq2Seq (Gehring et al., 2017)	-	-	26.36
Transformer-Base (Vaswani et al., 2017)	-	-	27.31
Transformer+EM Routing (Dou et al., 2019)	-	-	28.81
Transformer+Layer Aggregation (Dou et al., 2018)	-	-	28.78
Transformer+2D-CSANs (Yang et al., 2019b)	-	-	28.18

Table 5.2: Evaluation of translation performance on the WMT'14 English-German (En→De). The progressive gain between our implementation of the Transformer baseline and our approach is shown in parenthesis. #**Params** denotes the number of trainable parameters per model. **Train** indicates the training speed (steps/second). "†" and "‡" indicate statistically significant difference with $\rho < 0.05$ and $\rho < 0.01$, respectively.

is employed as the evaluation metric for the En→De task. The translation quality for the En↔Vi is reported based on the case-sensitive BLEU score computed with sacreBLEU[6]. Finally, for the other IWSLT translation tasks, case-sensitive BLEU metric evaluated with

[6]https://github.com/mjpost/sacrebleu

multi-bleu.pl[7] is employed. The statistical significance of the BLEU scores between the baseline and our *DC* based models is evaluated with paired bootstrap resampling (Koehn, 2004) using the compare-mt[8] (Neubig et al., 2019) library with 1000 resamples.

5.5 Results

This section presents the evaluation performance of the proposed *Dual Contextual* based model on the nine translation tasks under consideration. Tables 5.2 and 5.3 show the translation performance on the WMT'14 En→De[9] dataset and the eight IWSLT tasks respectively. Table 5.4 summarises the performance of existing models on the IWSLT En→Vi and Es→En translation tasks. In addition to comparison to existing NMT systems, the performance of the *DC* based models are also compared with the JASs and the Encoder-based MS models introduced in Chapters 3 and 4 respectively.

On the En→De translation task, the *DC* unit significantly improves the performance of the Transformer model by $+0.89$ BLEU in the case of the Enc-DC model and $+0.49$ BLEU with respect to the Dec-DC model. The performance of the Dec-DC model is improved by $+0.25$ BLEU when the *DC* module is applied to both subnetworks. However, the performance achieved by the Full-DC model is lower than that achieved by the Enc-DC model. While the primary goal of the JASs is to enhance the performance of the decoder subnetwork, the Enc-DC and the *MS-l* models employ strategies with the specific aim to improve the sentence representation ability of the encoder subnetwork. As shown, the MS and Enc-DC models generally achieve higher translation quality than the JASs models. This highlights the potential performance gain from further improving the effectiveness of encoding subnetworks. There is no significant performance difference between the Enc-DC and the *MS-l+AIF* (MS-1+AIF) models. However, the performance gain achieved by the former comes at minimal impact on the training complexity. Specifically as shown in Table 5.2 (column "**Train**"), there is only about 17.81% decrease in the training speed when the *DC* module is applied to the encoder subnetwork of Transformer baseline compared to the 41.37% decrease with respect to the *MS-l+AIF* (MS-1+AIF) model. A similar conclusion can be made between the Full-DC and *MS-l* (MS-4) models. As mentioned in

[7]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl
[8]https://github.com/neulab/compare-mt
[9]The value in parenthesis denotes the progressive gain over the Transformer baseline.

Task	Models			
	Transformer	Enc-DC	Dec-DC	Full-DC
Vi→En	29.03	29.42	29.20	**29.70**
En→Vi	30.58	**31.28**	31.24	31.10
Es→En	39.80	**40.18**	39.96	40.15
En→Es	37.90	**38.59**	38.28	38.10
Fr→En	32.85	33.35	33.15	**33.42**
En→Fr	33.15	**33.94**	33.43	33.60
Ro→En	26.66	**27.15**	26.81	27.14
En→Ro	20.91	21.20	21.22	**21.27**

Table 5.3: Evaluation of translation performance on the IWSLT tasks ({Vi, Es, Fr, Ro }↔En)

Section 4.5.2, generally, more effort is required to optimise the MS models effectively. Besides, the translation performance gain obtained via the *DC* module is higher than all the existing models, further demonstrating the superiority of the proposed approach.

The languages under consideration for the IWSLT tasks belong to different language families. The translation quality achieved further demonstrates the effectiveness of our proposed *DC* module based models at translating between languages of different families, as shown in Table 5.3. On average, the Enc-DC model consistently achieves higher performance gain over the Transformer baseline across all the IWSLT translation tasks. Compared to existing works, the baseline Transformer model consistently outperforms the models (Luong and Manning, 2015; Huang et al., 2018) with performance improvement of about $+2.51$ BLEU score on the En→Vi translation task as summarised in Tables 5.3 and 5.4. The proposed Enc-DC and Dec-DC models respectively achieved a BLEU score of 31.28 and 31.24. This represents a further performance gain of up to $+0.7$. On the other hand, a lower gain in the translation quality of $+0.52$ BLEU is achieved when the *DC* module is applied to both subnetworks (i.e. Full-DC). However, these *DC* based models have lower gains in the translation quality compared to the MS models. On the Es→En task, all our models outperform the Transformer baseline. A marginal performance gain of $+0.16$ BLEU is

Model	**BLEU**
Luong & Manning (Luong and Manning, 2015)	23.30
NPMT (Huang et al., 2018)	27.69
NPMT + LM (Huang et al., 2018)	28.07
Joint Attention Strategies	
Layer Aggregation (Iter-C-Agg)	31.13 (+0.55)
Multi-Layer Attention (M-01)	31.07 (+0.49)
Multi-Layer Supervision	
MS-l (MS-1)	31.32 (+0.74)
MS-l+AIF (MS-3+AIF)	31.64 (+1.06)

(a)

Model	**BLEU**
UEDIN (Cettolo et al., 2014)	37.29
Tied Transformer (Xia et al., 2019)	40.51
Layer-wise Coordination (He et al., 2018)	40.50
Joint Attention Strategies	
Layer Aggregation (Iter-C-Agg)	40.31 (+0.51)
Multi-Layer Attention (M-00)	40.99 (+1.19)
Multi-Layer Supervision	
MS-l (MS-2)	40.71 (+0.91)
MS-l+AIF (MS-3+AIF)	41.22 (+1.42)

(b)

Table 5.4: Existing Results on the IWSLT (a) En→Vi and (b) Es→En translation tasks.

achieved when the *DC* module is applied to only the decoder subnetwork (i.e. Dec-DC). On this dataset, the best performance is achieved by the Enc-DC and Full-DC models enhancing the translation quality by $+0.38$ BLEU and $+0.35$ BLEU, respectively. However, the translation performance of the Enc-DC, Dec-DC and Full-DC models is lower than that

achieved by Xia et al. (2019), He et al. (2018) and the JASs (Iter-C-Agg and *M-*ij models) and the MS (*MS-l* and *MS-l+AIF* models), outperforming only the (Cettolo et al., 2014).

Overall, the performance achieved across all the translation tasks indicate the benefits of leveraging both the local and global contextual information. Across the different datasets, the Enc-DC model consistently outperforms the Dec-DC. This can be attributed to the attention bias or mask employed by self-attention units within the decoding layers. During the decoding step t, the attention bias limits the decoder's self-attention to only consider the target sub-sequence (i.e. $y_{<t}$) generated so far. This implies the decoder layer is able to exploit the local information within the neighbourhood of target tokens $[y_1, y_2, \cdots, y_{t-1}]$. Therefore unlike the encoder subnetwork, the CNN unit generating l_c has a limited impact on the performance of the decoding layers and the decoder subnetwork. This is consistent with the observations made by Zhang et al. (2018). Furthermore, applying the *DC* module to both subnetworks (Full-DC) in most cases only improves the performance from the Dec-DC model's perspective. Besides, the Full-DC achieved higher performance gain over the Enc-DC model only on the Vi→En. However, there is no significant difference in the performance of the Full-DC and Enc-DC models on translation tasks including Es→En, Fr→En, and Ro→En.

As displayed in Table 5.2, the *DC* module introduces new parameters mainly due to the additional attention computation and the Local Contextual Unit across the different layers. The Full-DC leveraging the *DC* module across the encoder and decoder subnetworks results in about 25M new parameters compared to the 12.7M by the Dec-DC and Enc-DC models. As mentioned in Section 3.5.2, properties such as the optimizer employed, the number of trainable network parameters and other computations that directly alter the formulation of the network affects the overall computational speed of any given neural network model. Accordingly, the Full-DC, Dec-DC, and Enc-DC models have lower training speeds compared to the baseline model. Compared to Full-DC, the Dec-DC and Enc-DC models are shown to have the least decrease in the training speed. Based on the translation performance across the different language pairs and the overall impact on the computation speed, this work recommends limiting the explicit modelling of the local contextual information to only the encoding subnetwork.

5.6 Analysis

This section presents analyses performed to understand the performance improvement introduced by the *DC* module. These analyses are performed on the WMT'14 En→De dataset. As shown in Section 5.5, the best performance is obtained when the *DC* module is applied across the encoder subnetwork hence the analyses presented in Sections 5.6.1 to 5.6.3 are performed only on the Enc-DC model.

5.6.1 Linguistic Evaluations

As shown in Tables 5.2 and 5.3, the Enc-DC model significantly improves the performance of the Transformer baseline model across the different translation tasks under consideration. Aside from model properties, including model size, the optimizer employed, the translation performance of an NMT model is also affected by the linguistic and syntactic properties or structures of the language pairs under consideration. Therefore investigating the linguistic perspectives or properties improved by the proposed module will help quantify the performance gain via the *DC* module. Following (Conneau et al., 2018; Li et al., 2019), ten probing analyses are conducted to study the linguistic properties improved by the *DC* module presented in this chapter. The ten classification tasks are divided into three categories: *Surface* (Surf), *Syntactic* (Sync) and *Semantic* (Sem). For the **Surf** tasks, the emphasis is on evaluating the surface-level information captured by the sentence representation. On the other hand, the **Sync** tasks investigate the syntactic properties captured by the sentence representation. Finally, the ability of the encoder subnetwork to understand the denotation of a given sentence is evaluated under the **Sem** tasks. Details about these probing tasks are presented in Section 4.5.4.

The above tasks are performed based on the source representation generated by the encoding subnetwork. Specifically, the decoding subnetwork of the pre-trained NMT model is replaced with a two-layer MLP classifier. For each classification task, the input representation to the classifier is the mean of the output representation from the final encoding layer. $L2$ regularisation of $\lambda = 1e^{-4}$ is applied to the hidden layer of the classifier. The resulting models are trained and evaluated on the dataset[10] presented by Conneau et al. (2018). The training corpus for each task comprises 100k sentences, 10k sentences for validation

[10]https://github.com/facebookresearch/SentEval/tree/master/data/probing

Tasks		Models	
		Transformer	Enc-DC
Surface	SentLen	93.38	93.45
	WC	71.50	75.95
	#Avg	82.44	**84.70**
Syntactic	TDep	43.93	44.29
	BShift	69.49	74.11
	ToCo	74.56	75.44
	#Avg	62.66	**64.61**
Semantic	Tense	88.40	88.65
	SubjN	85.01	85.33
	ObjN	85.9	85.13
	SoMo	52.67	52.49
	CoIn	62.30	62.56
	#Avg	**74.86**	74.83

Table 5.5: Classification performance on the 10 probing tasks to evaluating the linguistic information ("Surface", "Syntactic" and "Semantic") learned by the encoding subnetwork of the Transformer baseline and our proposed model. "#Avg" indicates the average score across the sub-tasks under each category.

and 10k sentences for testing. The classifiers are trained for 100 epochs with RMProp optimizer using the learning rate of $2.5e^{-4}$ and mini-batch size of 64. Early stopping criterion is applied during training based on the accuracy score on the validation set. During training, only the parameters of the classifier are updated. The parameters of the encoding subnetwork are fixed to quantify the linguistic properties and information captured by the pre-trained encoder subnetwork of the Transformer baseline and our Enc-DC model.

Results The results of the probing tasks are summarised in Table 5.5. Despite the *DC* based encoding subnetwork of the Enc-DC model achieving the best performance on three of the five ***Semantic*** tasks, on average, it achieved almost identical performance as the baseline. The actual performance gain introduced by the *DC* module can be seen across the ***Surface*** and ***Syntactic*** tasks. Across these tasks, the Enc-DC model significantly outperformed the Transformer baseline. The tasks such as WC and CoIn require global contextual

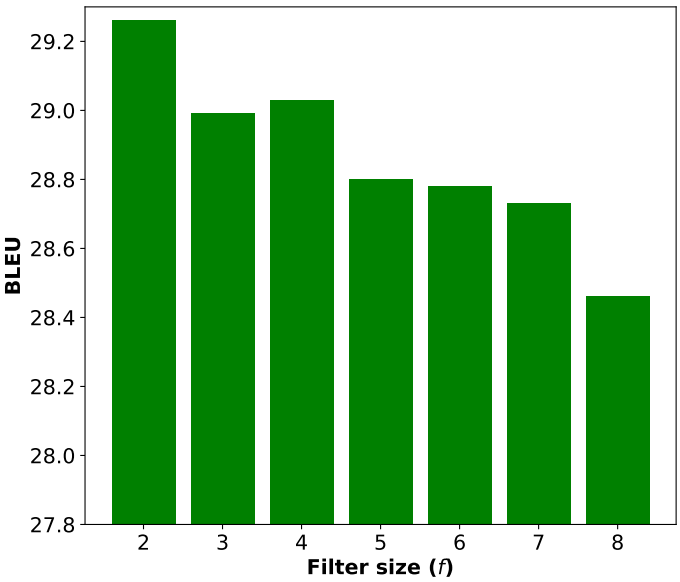


Figure 5.3: Impact of f (the convolution filter size employed by the Local Contextual Unit unit) on the performance of the Enc-DC model.

information while local contextual information is generally required to achieve higher performance on tasks, including ToCo, BShift, and SentLen. Overall, the performance on the ***Syntactic***, ***Surface***, and ***Semantic*** probing tasks demonstrates the effectiveness of the *DC* module on learning both the global and local contextual information required to enhance the translation performance. The *DC* module allows the encoder subnetwork to learn the surface and syntactic information of the source sentence with minimal impact on its ability to encode the deeper semantic properties.

5.6.2 Effect of CNN Kernel size

To analyse the impact of the CNN kernel size (employed by the Local Contextual Unit) on the translation quality, different Enc-DC models are trained with the kernel size $f \in [2, 3, 4, 5, 6, 7, 8]$. Figure 5.3 summarises the translation performance of each Enc-DC model. As shown, the Enc-DC models achieved almost identical BLEU score when trained with the filter size $5 \leq f \leq 7$. This is also true for the values of $f \in [3, 4]$. The worse performance is obtained with $f = 8$, producing a marginal performance gain of about $+0.10$ BLEU over the Transformer baseline. However, the best performance is achieved with $f \in [2, 3, 4]$.

Overall, the performance of the Enc-DC model generally decreases as the filter size increases. One possible reason is that for larger filter sizes, there is a higher possibility of information overlap between l_c and h_g. As a result, the FI unit generating the h_l (based on l_c and H_e^{l-1}) has limited impact on the overall performance of the *DC* module at learning the contextual sentence representation. This is because the generation of the h_g and h_l will be more useful and meaningful if each FI unit captures diverse information.

The above hypothesis is investigated by analysing the relationship between the attention weight distributions α_g and α_l associated with the generation of the hidden contextual representations h_g and h_l respectively. As mentioned in Section 5.3, each FI unit is a multi-head attention unit employing N_h attention heads. For simplicity, the attention weights for each encoding layer is represented by the attention head with the maximum weight distribution among all the N_h heads employed by the FI units:

$$\alpha_g = \max\left([\alpha_g^1, \alpha_g^2, \cdots, \alpha_g^{N_h}]\right)$$
$$\alpha_l = \max\left([\alpha_l^1, \alpha_l^2, \cdots, \alpha_l^{N_h}]\right)$$

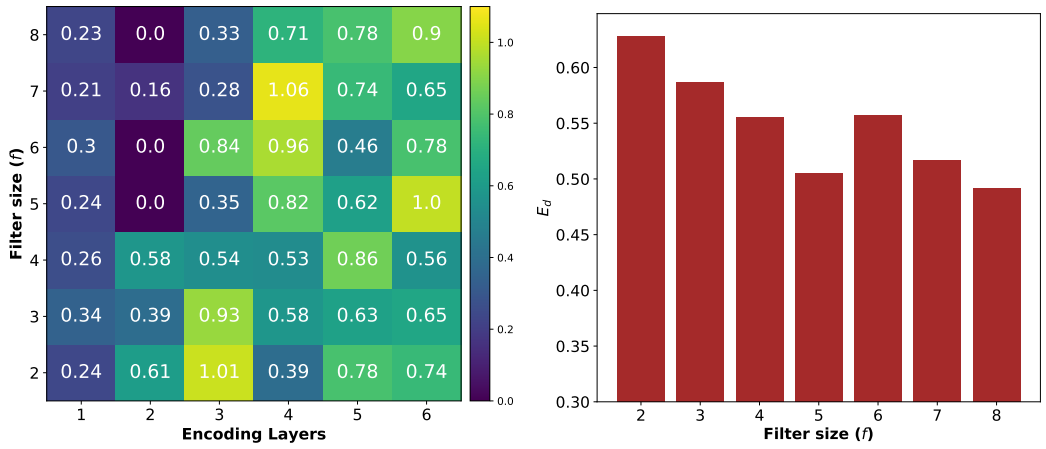
where α_g^i and α_l^i respectively are the weight distribution employed by the i^{th} attention head. To this end, Jensen-Shannon divergence $JS(P \mid\mid Q)$, is employed as a distance metric to measure how far the α_g and α_l are from each other. The distance is measured using $JS(P \mid\mid Q)$ because it is symmetric and bounded. Given the multi-dimensional attention weight distributions α_l and α_g, the divergence with respect to the encoding layer l, dv^l is computed as:

$$dv^l = JS(\alpha_g \mid\mid \alpha_l)$$
$$JS(\alpha_g \mid\mid \alpha_l) = \frac{1}{2} \times D_{KL}(\alpha_g \mid\mid m) + \frac{1}{2} \times D_{KL}(\alpha_l \mid\mid m)$$
$$m = \frac{1}{2} \times (\alpha_g + \alpha_l)$$

where $D_{KL}(P \mid\mid Q)$ is the Kullback-Leibler divergence formulated as:

$$D_{KL}(P \mid\mid Q) = \sum_x P(x) \log P(x) - \sum_x P(x) \log Q(x)$$

The overall divergence (E_d) for the encoding subnetwork is computed as the average of the



(a) (b)

Figure 5.4: Variation of the divergence between attention weights α_g and α_l for the difference filter size $f \in [2, 3, 4, 5, 6, 7, 8]$. (a) Divergence across each encoding layer. (b) Mean divergence (E_d) across all the layers within the encoding subnetwork with respect to the difference filter sizes.

divergence dv^l computed for each encoding layer:

$$E_d = \frac{1}{L}\sum_{l=1}^{L} dv^l$$

Larger values of E_d indicate that on average across the multiple encoding layers, the two FI units (of the *DC* module) capture more diverse information using the attention weights α_l and α_g.

Figure 5.4 illustrates the divergence between α_l and α_g for the different filter kernel sizes. As displayed in Figure 5.4a, variation in the divergence across the encoding layers is dependent on the value of f employed to train the Enc-DC model. In all cases, the divergence is larger across the top-4 encoding layers. Interestingly, the dv^l is zero or closer to zero for values of $f \geq 5$ across the second encoding layer compared to that of the models trained with $f \in [2, 3, 4]$. This implies that for these models with $f \geq 5$, the two FI units capture almost identical contextual information. Overall, the models trained with $f \in [2, 3, 4]$ have the lowest divergence across the top three layers among all the values of f under consideration. In contrast, the models with $f \geq 5$ have the lowest divergence

#	**Layers**	**BLEU**	Δ
1	[1-6]	29.26	**-**
2	[1-1]	29.04	-0.22
3	[1-2]	28.98	-0.28
4	[1-3]	28.93	-0.33
5	[1-4]	29.04	-0.22
6	[1-5]	28.94	-0.32
7	[5-6]	28.47	-0.79
8	[4-6]	28.57	-0.69
9	[3-6]	28.78	-0.48

Table 5.6: Translation performance of different combination of encoder layers of the Enc-DC model. [*i-j*] denotes limiting the application of the *DC* module to the encoding subnetwork from layer *i* to layer *j*. Δ indicates the difference in performance between the [1-6] and the [*i-j*] encoder layer combinations.

between the FI units across the first three layers. Figure 5.4b demonstrates that, on average, the divergence across the entire encoding subnetwork is higher for smaller values of f. Specifically, the E_d for the models trained with $f \in [2, 3, 4]$ is greater than that of $f \geq 5$. This implies that each FI unit within the encoding layers for the models with $f \in [2, 3, 4]$ can capture more diverse contextual information when generating h_g and h_l. However, there is less diversity with $f \geq 5$. Overall, the variation of the divergence as shown in Figure 5.4 and the corresponding translation quality displayed in Figure 5.3 confirm our hypothesis that smaller values of f allow the *DC* module to effectively capture more diverse contextual information to further improve the performance of the translation model.

5.6.3 Layers to consider

Recent works (Peters et al., 2018; Raganato et al., 2018; Belinkov et al., 2017) have revealed that each encoder layer captures different levels of abstraction of the source sequence. Therefore, these layers tend to have different requirements for the local contextual

and short-range dependency information. Based on these findings, Yang et al. (2018), Xu et al. (2019) and Shen et al. (2018) argue limiting the localness modelling to the first few encoding layers to allow the top-level layers to focus more on capturing the global contextual information. In contrast, this work argues that higher performance can be achieved when the local and global information modelling is applied across all encoder layers. To investigate this, an ablation study is performed whereby the *DC* module is applied to different combinations of the encoding layers.

Table 5.6 summarises the translation quality for the different layers combinations. As shown, all combination of the encoder layers consistently outperforms the Transformer baseline model further confirming the requirement of learning both the local and global contextual source information. For example, training the model with *DC* module across only the first two encoder layers (Row 3) produced a performance gain of about $+0.61$ BLEU over the baseline model. Among the different combinations of layers, limiting the *DC* module to only lower-level layers (Rows 2-6) achieved higher translation quality compared to when employed across only the top-level layers (Rows 7, 8 and 9). The worst performance (28.47 BLEU) is obtained when the *DC* module is incorporated into only the top-level encoding layers [5-6] (Row 7). Besides, the different combinations across the lower-level layers (Rows 2-6) produced fairly identical results with [1-1] (Row 2) and [1-4] (Row 5) achieving the best performance. This is consistent with the observations made by Yang et al. (2018), Shen et al. (2018) and Xu et al. (2019). However, our NMT model achieved the best translation performance when the *DC* module is applied to all the encoding layers.

5.6.4 Sentence Length

As mentioned in Sections 3.5.1 and 4.5.1, the performance of an NMT model is affected by the number of tokens within the source sentence. The encoder-decoder model should be able to effectively capture both the long-distance and short-range dependencies between the tokens to achieve higher performance on source sentences of any arbitrary length (Dou et al., 2018). Following (Luong et al., 2015b), sentences of similar length are grouped, and the translation quality of the outputs from the models for each group is calculated. The comparison presented here is based on the following sentence length groups: $<$10, 10-20, 20-30, 30-40, 40-50, and $>$50. For each length group, the translation quality is evaluated

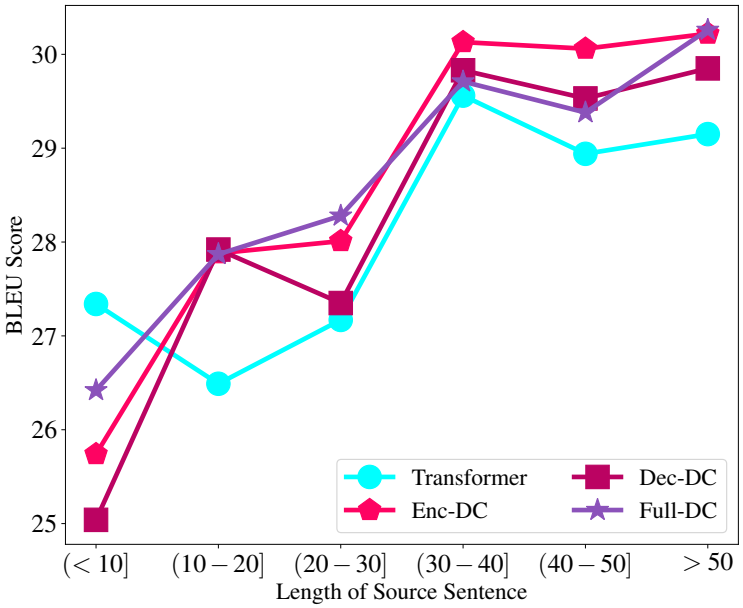


Figure 5.5: BLEU scores on the En→De task for the Transformer baseline, the *DC* module based models with respect to varying source sentence length.

for outputs from the models under consideration.

Figure 5.5 summarises the impact of the length of the source sentence. As shown, both the Transformer baseline and our *DC* based models (Enc-DC, Dec-DC and Full-DC) display identical variation in the translation quality across the different source sentence lengths. This is true especially for sentences with length greater than or equal to 20 subword tokens. However, our *DC* based models consistently outperform the baseline model across the different sentence groups with greater than 10 subword tokens. The performance gain achieved by our models is attributed to the addition of the *DC* module which allows both the encoder (in the case of Enc-DC), the decoder (in the case of Dec-DC) or both (in the case of Full-DC) to effectively exploit both the local and global contextual information required to improve the generation of the target translation. For shorter sentences with fewer than ten subword tokens, the global contextual model of the self-attention module is shown to be effective enough for the target generation. In contrast, for longer source sentences, further improvement in the translation quality is achieved when both the global and local contextual modelling are employed as shown across the sentence groups 10-20, 20-30, 30-40, 40-50 and >50. Overall, the best performance across the sentence groups

(greater than 10 tokens) is achieved when the *DC* module is applied to only the encoding subnetwork (i.e. Enc-DC). This supports our recommendation to limit the global and local modelling to only the encoder subnetwork.

5.7 Summary

This chapter proposed the *Dual Contextual* (DC) module to leverage the local and global contextual information to improve the translation performance of the Transformer model. Three possible applications of the *DC* module, namely, Enc-DC, Dec-DC and Full-DC were presented. The analyses performed indicate that:

- exploiting both the global and local contextual information is beneficial to the overall performance of the translation model.
- the best performance is achieved when the *DC* module is applied to only the encoding layers (i.e. Enc-DC). The decoding subnetwork employing the *DC* module (in the case of Dec-DC and Full-DC) produces lower performance gain over the baseline model.
- in contrast to the findings of recent works (Yang et al., 2018; Xu et al., 2019; Shen et al., 2018), applying the local and global contextual modelling across all the encoding layers significantly improves the translation performance compared to limiting its application to only the first few lower-level encoding layers (e.g. from layer 1 to 3).

Part IV

Conclusion

Chapter 6

Conclusion and Future Work

This chapter summarizes the experimental results, analyses and the contributions of the thesis. Finally, the potential avenues for future work are presented.

6.1 Conclusion

This thesis has aimed to study and design deep neural architectures for performing sequence to sequence learning more effectively. Specifically, three approaches to improving the translation performance of deep NMT models were presented. The higher performance of our proposed models is attributed to the enhancement of sentence representation and generation ability of the encoder-decoder framework for the NMT task. The contributions of the thesis are in two main folds: (1) To ensure minimal loss of source information during the target translation, two main joint attention strategies are presented in Chapter 3 to allow the decoder access to source information captured by different encoding layers. (2) Enhancing the sentence representational ability of the encoder and decoder subnetworks using strategies including (a) exploiting the strengths of multitask learning and auxiliary training approaches to design an encoder-based multi-level supervision framework in Chapter 4; (b) improving the performance of the self-attention mechanism at learning the local contextual information without limiting it's ability to model the global contextual information and dependencies in Chapter 5.

Among the three different approaches presented, this work recommends employing the Dual Contextual (DC) module for the language translation. This recommendation is

based on the overall impact on the computation speed and the translation performance achieved across the different language pairs under consideration. For example, as shown in Section 5.5, there is no significant difference in translation performance between the Enc-DC and *MS-l* on the WMT'14 English-German (En→De) task. However, the performance gain of the later comes at higher training computational cost compared to the former.

Joint Attention Strategies (JASs)

Chapter 3 presents our answer to the research question related to ensuring minimal loss of source information when mapping from the source sequence to the target sequence. Specifically, the chapter presents two main joint attention computation approaches to exploiting source representations from multiple encoding layers. These strategies are the *Layer Aggregation* and *Multi-Layer Attention* approaches. The *Layer Aggregation* strategies compute a joint source representation as a combination of sentence representations from multiple encoding layers. The resulting source representation is passed to the decoder subnetwork during the target generation. Each of the four *Layer Aggregation* strategies presented provides the decoder subnetwork indirect access to source information from different layers within the encoding subnetwork. Under the *Multi-Layer Attention*, the decoder performs attention computations directly across the outputs from multiple encoding layers. Unlike the *Layer Aggregation* strategies, this approach provides the decoder with direct access to the representations generated by multiple encoder layers. A salient goal of these JASs is to enhance the flow of gradient information between the encoder-decoder subnetworks. Consequently, this significantly alters how the long and short-range contextual information is learned by the self-attention unit employed within each encoding layer, as shown in Section 3.5.3. Overall, the experimental results on three translation tasks demonstrate the improvement over models exploiting only the source representation from the final encoder layer. However, the translation performance of the models is shown to be dependent on the strategy employed and the number of encoder layers exposed to the decoder subnetwork.

Multi-level Supervision Strategies

The target generation ability of the decoder subnetwork is dependent on the performance of the encoder at learning the source semantic representation required for the translation task.

Our attempt to answer the question "To improve the source sentence representation ability of the encoder subnetwork, what motivation or inspiration can be drawn from multi-task learning (MTL) approaches to neural machine translation and sequence labelling tasks such as Named Entity Recognition?" is presented in Chapter 4. Multiple decoders are connected to different layers of the encoder subnetwork. Unlike conventional MTL approaches to NMT (Niehues and Cho, 2017; Baniata et al., 2018; Luong et al., 2015a; Malaviya et al., 2017), all decoders are trained end-to-end on the same target generation task. This learning strategy is referred to as *Encoder-based Multi-level Supervision*. The translation results demonstrate that training the network with decoders connected to the lower-level layers can improve the translation performance of the *Main-Decoder* connected to the final encoder layer. Similar to the JASs models, *Encoder-based Multi-level Supervision* models (*MS-l* and *MS-l+AIF*) also leverage source representations from multiple encoder layers. However, the *MS-l* and *MS-l+AIF* models consistently outperform the *Layer Aggregation* and *Multi-Layer Attention* approaches on all the translation tasks under consideration. The higher performance improvement of the *MS-l* and *MS-l+AIF* models is attributed to the *Encoder-based Multi-level Supervision* benefiting from the strength of MTL. The lower-level layers are shared across multiple decoders connected to the top-level encoder layers. Therefore via inductive bias, these lower-level encoder layers are encouraged to capture the necessary source information required by all connected decoders. Further performance improvement is achieved from the direct information passing between the decoders via the *Auxiliary Information Fusion* (AIF) in the case of the *MS-l+AIF* models. In-depth analyses performed showed that the translation performance gain of the *MS-l* and *MS-l+AIF* models is dependent on the choice of lower-level encoder layers connected auxiliary decoders. Besides, it is evident from Table 4.6 that it is not necessary to connect auxiliary decoders to all encoding layers to achieve higher performance. A single auxiliary decoder is recommended. In the case of the *MS-l* model, limiting the application of the auxiliary decoder to any of the first few lower-level layers produced a better performance than when connected to the top-level layers. In contrast, the *MS-l+AIF* model generally performs better with the auxiliary decoder connected to the top-level encoding layers. Furthermore, unlike the *MS-l* model, the *Main-Decoder* of the *MS-l+AIF* model leverages the target representations from the auxiliary decoders via the AIF unit. Consequently, for these models, the translation quality of the *Main-Decoder* subnetwork of the *MS-l+AIF* model is affected by

the target generation ability of the *Aux-Decoder l* subnetworks.

Dual Contextual Modelling

During the target generation, rich contextual information is required by the translation model to effectively perform tasks such as the phrase and word sense disambiguation (Wu et al., 2014; Marvin and Koehn, 2018). Chapter 5 presents our solution related to the research question of leveraging both the global and local contextual information more effectively to enhance the quality translation outputs. Specifically, the chapter proposes the *Dual Contextual* (DC) module, an extension to the self-attention unit to exploiting both the local and global contextual information without restricting the attention scope. The chapter explored three possible applications of the proposed *DC* module. These are Enc-DC model (only the encoding subnetwork employs the *DC* module), Dec-DC model (the *DC* module is employed within the layers of only the decoding subnetwork), and Full-DC model (employs the *DC* module across the multiple layers within the encoding and decoding subnetworks). The linguistic probing analysis presented in Section 5.6.1 suggests that the *DC* module allows the encoder subnetwork to learn the surface and syntactic features of the source sentence with minimal impact on its ability to effectively encode the deeper semantic linguistic information. By leveraging both the local and global contextual information, our *DC* based models consistently achieved significant gain in the translation quality across nine translation tasks. Furthermore, the results across these translation tasks suggest the explicit modelling of the local and global information via the *DC* module is beneficial when applied to only the encoding subnetwork (i.e. Enc-DC model). Within the decoder subnetwork, the performance of the Local Contextual Unit of the *DC* module is limited by the attention bias mask employed.

6.2 Future Work

The works presented in this thesis all focus exclusively on improving the translation performance of NMT systems. However, other NLP tasks, including sequence tagging, document summarization, and paraphrase generation, could also benefit from the approaches proposed in the previous chapters. The proposed models can be directly employed to learn the document summarization and paraphrase generation tasks with little to no modification

to the network architecture. The sequence tagging task such as Named Entity Recognition (NER) will significantly benefit from the local and global contextual information learning ability of the *DC* module. However, unlike the document summarization task, the decoder subnetwork of the Enc-DC can be replaced with a softmax classifier or a Conditional Random Fields (CRF) classifier (Lafferty et al., 2001) to effectively models the tagging decisions.

Despite the high performance achieved by our proposed models, the main challenges that remain and future work related to the overall aim of this thesis are:

- **Model Compression**: The main focus of this thesis has been building effective neural models and improving performance in terms of translation quality. In most cases, the proposed approach results in a significant increase in the number of model parameters, further increasing the computational complexity of the resulting model. For example, the *Multi-Layer Attention* strategies as shown in Table 3.2 introduce up to 31M new parameters due to the additional attention computations. This is shown to affect the training speed as more effort is required to efficiently optimise the extra parameters. Similarly unlike the *MS-l* models, the *MS-l+AIF* uses all decoders during both the training and testing phases resulting in a significant decrease in both the decoding and training speeds. During deployment, the inference time is a critical component of any machine learning model to consider. The future direction related to the above problems could focus on reducing the model size and the overall computational complexities by building light-weight models via model compression (with minimal loss in terms of the overall translation performance). This will involve compressing the network parameters to fit the computational resource constraints of portable devices such as mobile phones and smartwatches using techniques including network pruning (Han et al., 2016, 2015), knowledge distillation (Kim and Rush, 2016; Tan et al., 2018) and Low-rank factorization (Denil et al., 2013; Lu et al., 2017). The JASs models presented are good candidates for network pruning. The results in Section 3.5.4 demonstrates that for C-Agg, S-Agg, Iter-S-Agg, M-00, M-01, and M-11 models, the outputs from some of the encoder layers are shown to be redundant during inference and thus can be removed from the joint attention computations without significantly reducing the translation quality. For example, in the case of the C-Agg and S-Agg models, only the first and final layers of the encoder

subnetwork contribute significantly to the overall translation performance compared to the intermediate layers.

- **Leveraging Linguistic Information**: As mentioned in Section 3.4, the linguistic properties and structures of the language pairs under consideration have a significant impact on the overall translation performance of the neural models. An effective model should be able to implicitly learn the necessary linguistic patterns of syntax, morphology, and semantics. However, NMT systems are generally limited by their capacity to fully capture more complicated patterns and structures of the source and target languages resulting in lower performance compared to human translators. An interesting thread of future work could investigate strategies to leverage or inject additional linguistic information to further improve the translation performance of the proposed models especially, on morphologically complex languages such as Czech and Arabic. The linguistic information can be directly passed as additional input features to the NMT model. Alternatively, the learning task can be formulated as an MTL problem where the goal is to learn the machine translation task and linguistic features (such as POS tags, named entity tags and dependency labels) simultaneously. For example, it is possible to convert any of the models proposed in Chapters 3 and 5 to an MTL network by connecting to the encoder subnetwork, an additional subnetwork dedicated to learning source linguistic information. Besides, the *Gated Task Interaction* (GTI) framework proposed by Ampomah et al. (2019a) can be employed to capture the mutual dependencies between learning tasks to improve the performance of the resulting MTL model. In the case of the *MS-l* and *MS-l+AIF* models, one of the auxiliary decoders can be re-purposed to learn the linguistic information of the source sequence.

References

Kamal Al-Sabahi, Zhang Zuping, and Mohammed Nadher. 2018. A hierarchical structured self-attentive model for extractive document summarization (hssas). *IEEE Access* 6:24205–24212.

Juan A Alonso and Gregor Thurmair. 2003. The comprendium translator system. In *Proceedings of the 9th Machine Translation Summit*.

Isaac K. E. Ampomah, Sally McClean, Lin Zhiwei, and Glenn Hawe. 2020. Every Layer Counts: Multi-Layer Multi-Head Attention for Neural Machine Translation . *The Prague Bulletin of Mathematical Linguistics* 115:51–82.

Isaac K. E. Ampomah, Sally I. McClean, Zhiwei Lin, and Glenn I. Hawe. 2019a. Gated task interaction framework for multi-task sequence tagging. In *IJCNN*. pages 1–8.

Isaac KE Ampomah, Sally McClean, Zhiwei Lin, and Glenn Hawe. 2019b. Jass: Joint attention strategies for paraphrase generation. In *International Conference on Applications of Natural Language to Information Systems*. Springer, pages 92–104.

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the NAACL:HLT, Volume 1 (Long Papers)*. pages 82–91.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv:1607.06450* .

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. *Proceedings of the 5th ICLR* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR'15, arXiv:1409.0473* .

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. pages 65–72.

Laith H Baniata, Seyoung Park, and Seong-Bae Park. 2018. A neural machine translation model for arabic dialects that utilizes multitask learning (mtl). *Computational intelligence and neuroscience* 2018.

Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. In *Proceedings of the 2018 Conference on EMNLP*. pages 3028–3033.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the 8th IJCNLP (Volume 1: Long Papers)*. pages 1–10.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* pages 1137–1155.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* pages 157–166.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the EMNLP*. pages 1442–1451.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*. pages 3579–3584.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar F Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the 6th Workshop on SMT*. ACL, pages 22–64.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

Asli Celikyilmaz and Dilek Hakkani-Tur. 2010. A hybrid hierarchical model for multi-document summarization. In *Proceedings of the 48th Annual Meeting of the ACL*. ACL, USA, ACL '10, page 815–824.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the IWSLT*. page 57.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2015. The iwslt 2015 evaluation campaign. In *International Conference on Spoken Language*. page 57.

Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2019. Recurrent positional embedding for neural machine translation. In *Proceedings of the EMNLP-IJCNLP*. ACL, pages 1361–1367.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the ACL (Volume 1: Long Papers)*. pages 76–86.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language* 13(4):359–394.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL*. ACL, pages 263–270.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the EMNL)*. pages 1724–1734.

Chong Chai Chua, Tek Yong Lim, Lay-Ki Soon, Enya Kong Tang, and Bali Ranaivo-Malancon. 2017. Meaning preservation in example-based machine translation with structural semantics. *Expert Systems with Applications* 78:242–258.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *International conference on machine learning*. pages 2067–2075.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the ACL (Volume 1: Long Papers)*. pages 2126–2136.

Anna Currey, Antonio Valerio Miceli-Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*. pages 148–156.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pages 933–941.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*. pages 9712–9724.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. 2013. Predicting parameters in deep learning. In *Advances in neural information processing systems*. pages 2148–2156.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1500–1505.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd of the ACL and the 7th IJCNLP*. pages 1723–1732.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *Proceedings of the 2018 Conference on EMNLP*. ACL, pages 4253–4262.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. 2019. Dynamic layer aggregation for neural machine translation with routing-by-agreement. *arXiv:1902.05770* .

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, et al. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. pages 355–364.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. Irstlm: an open source toolkit for handling large scale language models. In *9th Annual Conference of the International Speech Communication Association*.

Mikel L Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine translation* 25(2):127–144.

Shiv Gehlot, Anubha Gupta, and Ritu Gupta. 2020. Sdct-auxnetθ: Dct augmented stain deconvolutional cnn with auxiliary classifier for cancer diagnosis. *Medical Image Analysis* 61:101661.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pages 1243–1252.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2004. Fast and optimal decoding for machine translation. *Artificial Intelligence* 154(1-2):127–143.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the 8th IJCNLP (Volume 1: Long Papers)*. pages 30–39.

Kevin Gimpel and Noah A Smith. 2008. Rich source-side context for statistical machine translation. In *Proceedings of the third workshop on SMT*. ACL, pages 9–17.

Min Gui, Junfeng Tian, Rui Wang, and Zhenglu Yang. 2019. Attention optimization for abstractive document summarization. In *Proceedings of the 2019 Conference on the EMNLP-IJCNLP*. pages 1222–1228.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR'16,arXiv:1510.00149* .

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*. pages 1135–1143.

Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, Oladimeji Farri, et al. 2016. Neural paraphrase generation with stacked residual lstm networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 2923–2934.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on EMNLP*. pages 1923–1933.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv:1803.05567* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems*. pages 7944–7954.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proceedings of the Coling 2008*. pages 321–328.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*. pages 473–479.

MD Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. 2019. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys (CSUR)* 51(6):1–36.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on CVPR*. pages 4700–4708.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*. Cambridge, MA, pages 66–73.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. *ICLR* .

John Hutchins. 1993. Latest developments in machine translation technology: beginning a new era in mt research. *Proceedings MT Summit IV.: International cooperation for global communication* pages 11–34.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv:1611.01462* .

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. Language modeling with deep transformers. *Proc. Interspeech 2019* pages 3905–3909.

Xiaojie Jin, Yunpeng Chen, Jian Dong, Jiashi Feng, and Shuicheng Yan. 2016. Collaborative layer-wise discriminative learning in deep neural networks. In *European Conference on Computer Vision*. Springer, pages 733–749.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2018. Depthwise separable convolutions for neural machine translation. *ICLR'18* .

Ramandeep Singh Kathuria, Siddharth Gautam, Arjan Singh, Smarth Khatri, and Nishant Yadav. 2019. Real time sentiment analysis on twitter data using deep learning (keras). In *ICCCIS*. IEEE, pages 69–73.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the AAAI*. pages 2741–2749.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on EMNLP*. pages 1317–1327.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* .

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the ACL*. pages 2676–2686.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on EMNLP*. pages 388–395.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 66–75.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

Haoran Li, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2018. Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 1430–1441.

Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R Lyu, and Zhaopeng Tu. 2019. Information aggregation for multi-head attention with routing-by-agreement. In *Proceedings of NAACL:HLT*. pages 3566–3575.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *ICLR, arXiv:1703.03130* .

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*. ACL, pages 17–24.

Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. *ICLR'17,arXiv:1608.03983* .

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 2017. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 5334–5343.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114* .

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the IWSLT*. pages 76–79.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Proceedings of the 2017 Conference on EMNLP*. pages 2529–2535.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-08: HLT*. pages 1003–1011.

Rebecca Marvin and Philipp Koehn. 2018. Exploring word sense disambiguation abilities of neural machine translation systems (non-archival extended abstract). In *Proceedings of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*. pages 125–131.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781* .

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pages 234–239.

Mudasir Mohd, Rafiya Jan, and Muzaffar Shah. 2020. Text document summarization using word embedding. *Expert Systems with Applications* 143:112958.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Predicting target language ccg supertags improves neural machine translation. In *Proceedings of the Second Conference on Machine Translation*. pages 68–79.

Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. 2019. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*. pages 9126–9135.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt: A tool for holistic comparison of language generation systems. In *Proceedings of the 2019 Conference of the NAACL*. pages 35–41.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the 2nd Conference on Machine Translation*. pages 80–89.

Thomas R Niesler, Edward WD Whittaker, and Philip C Woodland. 1998. Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *Proceedings of the ICASSP*. IEEE, volume 1, pages 177–180.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on EMNLP and Very Large Corpora*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on ACL*. ACL, pages 311–318.

Nikolaos Pappas, Lesly Miculicich, and James Henderson. 2018. Beyond weight tying: Learning joint input-output embeddings for neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. pages 73–83.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the EMNLP*. pages 2249–2255.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR, abs/1211.5063* 2:417.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. pages 91–100.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *Proceedings of the ICLR Workshop, 2017, arXiv:1701.06548* .

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*. pages 2227–2237.

Giulio Ermanno Pibiri and Rossano Venturini. 2019. Handling massive n-gram datasets efficiently. *TOIS* 37(2):1–41.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics* 110:43–70.

Alessandro Raganato, Jörg Tiedemann, et al. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. ACL.

Rita Raley. 2003. Machine translation and global english. *The Yale Journal of Criticism* 16(2):291–313.

Manish Rana and Mohammad Atique. 2016. Enhancing bi-lingual example based machine translation approach. *International Journal of Advanced Engineering Research and Science* 3(10):236860.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. *ACL* .

Markus Saers. 2011. *Translation as linear transduction: Models and algorithms for efficient learning in statistical machine translation*. Ph.D. thesis, Acta Universitatis Upsaliensis.

Amit Sangodkar and Om P Damani. 2012. Re-ordering source sentences for smt. In *LREC*. pages 2164–2171.

Sheikh Muhammad Sarwar, Hamed Bonab, and James Allan. 2019. A multi-task architecture on relevance-based neural query translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. pages 6339–6344.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 1715–1725.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the NAACL:HLT, Volume 2 (Short Papers)*. pages 464–468.

T Shen, T Zhou, G Long, J Jiang, and C Zhang. 2018. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *Proceedings of the ICLR*.

Aditya Siddhant, Ankur Bapna, Yuan Cao, Orhan Firat, Mia Chen, Sneha Kudugunta, Naveen Arivazhagan, and Yonghui Wu. 2020. Leveraging monolingual data with self-supervision for multilingual neural machine translation. *arXiv preprint arXiv:2005.04816* .

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*. Cambridge, MA, volume 200.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*. pages 5877–5886.

Chang-Woo Song, Hoill Jung, and Kyungyong Chung. 2019a. Development of a medical big-data mining process using topic modeling. *Cluster Computing* 22(1):1949–1958.

Guocong Song and Wei Chai. 2018. Collaborative learning for deep neural networks. In *Advances in Neural Information Processing Systems*. pages 1832–1841.

Kaitao Song, Xu Tan, Di He, Jianfeng Lu, Tao Qin, and Tie-Yan Liu. 2018. Double path networks for sequence to sequence learning. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 3064–3074.

Shengli Song, Haitao Huang, and Tongxiao Ruan. 2019b. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools and Applications* 78(1):857–875.

Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. 2018. Self-attentional acoustic models. *Proc. Interspeech 2018* pages 3723–3727.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv :1505.00387* .

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *ICSLP*.

Yuting Su, Yuqian Li, Ning Xu, and An-An Liu. 2020. Hierarchical deep neural network for image captioning. *Neural Processing Letters* 52(2):1057–1067.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 1–9.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2018. Multilingual neural machine translation with knowledge distillation. In *ICLR*.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational linguistics* 29(1):97–133.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. pages 5998–6008.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284* .

Hongli Wang and Jiangtao Ren. 2018. A self-attentive hierarchical model for jointly improving text summarization and sentiment classification. In *Asian Conference on Machine Learning*. pages 630–645.

Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li, and Jingbo Zhu. 2018a. Multi-layer representation fusion for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 3015–3026.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, Guiquan Liu, and Tie-Yan Liu. 2018b. Dual transfer learning for neural machine translation with marginal distribution regularization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Paul J Werbos. 1988. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks* 1(4):339–356.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2018. Pay less attention with lightweight and dynamic convolutions. In *Proceedings of the ICLR*.

Haiyang Wu, Daxiang Dong, Xiaoguang Hu, Dianhai Yu, Wei He, Hua Wu, Haifeng Wang, and Ting Liu. 2014. Improve statistical machine translation with context-sensitive bilingual semantic embedding model. In *Proceedings of the EMNLP*. ACL, pages 142–146.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 1: Long Papers)*. pages 698–707.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144* .

Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. 2019. Tied transformers: Neural machine translation with shared encoder and decoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*. volume 33, pages 5466–5473.

Yingce Xia, Xu Tan, Fei Tian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2018. Model-level dual learning. In *International Conference on Machine Learning*. pages 5383–5392.

Mingzhou Xu, Derek F Wong, Baosong Yang, Yue Zhang, and Lidia S Chao. 2019. Leveraging local and global patterns for self-attention networks. In *Proceedings of the 57th Annual Meeting of the ACL*. pages 3069–3075.

Baosong Yang, Jian Li, Derek F Wong, Lidia S Chao, Xing Wang, and Zhaopeng Tu. 2019a. Context-aware self-attention networks. In *Proceedings of the AAAI*. volume 33, pages 387–394.

Baosong Yang, Zhaopeng Tu, Derek F Wong, Fandong Meng, Lidia S Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *Proceedings of the EMNLP*. pages 4449–4458.

Baosong Yang, Longyue Wang, Derek F Wong, Lidia S Chao, and Zhaopeng Tu. 2019b. Convolutional self-attention networks. In *Proceedings of NAACL-HLT*. pages 4040–4045.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018a. Qanet: Combining local convolution with global self-attention for reading comprehension. In *Proceedings of the ICLR*.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. 2018b. Deep layer aggregation. In *Proceedings of the IEEE conference on CVPR*. pages 2403–2412.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of the HLT-NAACL*. pages 257–264.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the ACL*. pages 1789–1798.

Chun-Xiang Zhang, Ming-Yuan Ren, Zhi-Mao Lu, Ying-Hong Liang, Da-Song Sun, and Yong Liu. 2011. Extraction of chinese-english phrase translation pairs. In *International Workshop on Computer Science for Environmental Engineering and EcoInformatics*. Springer, pages 315–318.

Ying Zhang, Ralf D Brown, and Robert Frederking. 2001. Adapting an example-based translation system to chinese. In *Proceedings of the first international conference on Human language technology research*.

Shuyan Zhou, Xiangkai Zeng, Yingqi Zhou, Antonios Anastasopoulos, and Graham Neubig. 2019. Improving robustness of neural machine translation with multi-task learning. In *4th Conference on Machine Translation (WMT)*. Florence, Italy.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the NAACL: HLT*. pages 30–34.