PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 115 OCTOBER 2020 51–82

Every Layer Counts: Multi-Layer Multi-Head Attention for Neural Machine Translation

Isaac Kojo Essel Ampomah,[a] Sally McClean,[a] Lin Zhiwei,[b] Glenn Hawe[a]

[a] School of Computing, Ulster University, Belfast, UK
[b] Mathematical Science Research Center, LG006 Lanyon Building, Queen's University, Belfast, UK

Abstract

The neural framework employed for the task of neural machine translation (NMT) usually consists of a stack of multiple encoding and decoding layers. However, only the source feature representation from the top-level encoder layer is leveraged by the decoder subnetwork during the generation of target sequence. These models do not fully exploit the useful source representations learned by the lower-level encoder layers. Furthermore, there is no guarantee that the top-level encoder layer encodes all the necessary source information required by the decoder for the target generation. Inspired by recent advances in deep representation learning, this paper proposes a *Multi-Layer Multi-Head Attention* (MLMHA) module to exploit the different source representations from the multi-layer encoder subnetwork. Specifically, the decoder is allowed a more direct access to multiple encoder layers during the target generation. This technique further improves the translation performance of the model. Also, exposing multiple encoder layers enhances the flow of gradient information between the two subnetworks. Experimental results on two IWSLT language translation tasks (Spanish-English and English-Vietnamese) and WMT'14 English-German demonstrate the effectiveness of allowing the decoder access to representations from multiple encoder layers. Specifically, the MLMHA approaches explored in this paper achieve improvements up to +0.71, +0.75 and +0.49 BLEU points over the Transformer baseline model on the English-German, Spanish-English, and English-Vietnamese translation tasks respectively.

1. Introduction

Neural machine translation (NMT) architectures (Luong et al., 2015; Vaswani et al., 2017; Gehring et al., 2017) have achieved significant improvement over statistical ma-

 Corresponding author: ampomah-i@ulster.ac.uk
Cite as: Isaac Kojo Essel Ampomah, Sally McClean, Lin Zhiwei, Glenn Hawe. Every Layer Counts: Multi-Layer Multi-Head Attention for Neural Machine Translation . The Prague Bulletin of Mathematical Linguistics No. 115, 2020, pp. 51–82. doi: 10.14712/00326585.005.

chine translation techniques (Och et al., 1999; Callison-Burch et al., 2011; Koehn and Schroeder, 2007) without the need for extensive feature engineering. The backbone of these architectures is the encoder-decoder framework. The task of the encoder 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.

Recent state-of-the-art (SOTA) 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 minimize this problem, recent models (Vaswani et al., 2017; Gehring et al., 2017) employ shortcut connections such as residual units (He et al., 2016) between the layers to enhance the flow of information across the multiple layers. Furthermore, recent works (Raganato et al., 2018; Belinkov et al., 2017) also have revealed that each encoding layer extracts different levels of abstraction of the source representation. For example, Belinkov et al. (2017) evaluated representations extracted from different encoder layers on tasks such as part-of-speech tagging (POS) and semantic tagging. They argue that the lower-level encoder layers focus more on learning word-level information/properties whilst the higher-level layers encode more semantic information. All these representations can be exploited to further improve the task of sequence to sequence (seq2seq) generation. However, current NMT models generate the target sequence based on representation from only the top-level encoding layer. These models fail to fully explore the multiple useful source representations generated in the lower-level encoder layers during the target generation. A problem with this approach is that there is little to no guarantee that the necessary source information required by the decoder subnetwork is encoded in the top-level encoder layer (Wang et al., 2018; Dou et al., 2018).

Research works from the field of computer vision (Yu et al., 2018; Huang et al., 2017) have proven the benefits and the performance impact of exploiting representations from multiple top-level and lower-level layers. Inspired by this, several feature aggregation techniques have been proposed to improve the performance of NMT models (Dou et al., 2018; Wang et al., 2018; Bapna et al., 2018). These aggregation approaches focus on generating a single source representation as a combination of all representations from the multiple encoder layers. Even though these techniques provide a simple way to exploiting the multiple source representations, this work argues that allowing the decoder more direct access to the encoding layers can further improve the flow of gradient information and enhance the overall performance of the model. This paper is motivated by the findings in our previous work (Ampomah et al., 2019).

In our previous work, the performance of an RNN based seq2seq model was improved by performing the neural attention computations jointly across source representations from all encoding layers. The encoder employed comprised of multiple Bidirectional LSTM (BiLSTM) layers whilst the decoder consisted of a single LSTM

layer. Allowing the decoder more direct access to the stack of encoder layers significantly improved the performance of the model on the task of paraphrase generation. This work aims at enhancing the translation performance of a current SOTA model, namely the Transformer architecture (Vaswani et al., 2017), on the more challenging task of translating sentences from one language to another. Unlike (Ampomah et al., 2019), both the encoder and decoder subnetworks of the Transformer model employed in this work consist of multiple layers. Each of the decoding layers employs an encoder-decoder multi-head attention (MHA) sublayer to learn the source-target context information based on the source representation from the top-level encoder layer. To generate the contextual information based on the n source representations from the multiple encoder layers, the standard encoder-decoder multi-head attention sublayer is replaced with a MLMHA sublayer. The n source representations are aggregated by a *Source Feature Collector* module based on the outputs from the top-n encoding layers. The MLMHA module allows each decoding layer to interact with different levels of abstraction of the source sequence to further improve the translation quality. This also enhances the propagation of gradient information between the encoder-decoder subnetworks as each encoder layer receives error signals from all the decoding layers. 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 direct access to representations from multiple encoder layers. The contributions of this work are:

- proposing the *Multi-Layer Multi-Head Attention* module which allows the decoding layers to 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 encoder 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 MLMHA module within the decoding layers.

The remainder of the paper is organized as follows: Section 2 briefly reviews the related works and Section 3 provides a background to neural machine translation. The Multi-Layer Multi-Head Attention approaches are presented in Section 4. The experiments conducted are presented in Section 5, and the results are compared and discussed in Section 6. Also, Section 7 presents a detailed analysis performed to investigate the impact of exploiting multiple source representations from the encoder subnetwork via the MLMHA unit. Finally, the conclusion is presented in Section 8.

2. Related works

The proposed MLMHA framework is 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. 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 SOTA models for learning problems employing neural networks. With respect to machine translation, 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 SOTA 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 decoding subnetwork.

Making use of source representations from multiple encoding layers has been shown to improve the generalization performance of deep NMT models. To learn better source representation, (Wang et al., 2018) 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. To ensure that all layers capture diverse source information, they further proposed to train the neural model with a diversity promoting auxiliary learning objective. The static layer aggregation approaches from (Dou et al., 2018; Wang et al., 2018) (such as the linear feature combination method) as argued by Dou et al. (2019) sometimes ignore 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 N encoding and M decoding layers, M different joint source representations are generated (one for each decoding layer) from the weighted combination of outputs from all the encoder 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 optimization of deeper models.

A common theme among these works is the generation of a single source feature representation as an aggregation of representations from different encoder layers. The decoder layers perform the source-target attention computations based on the aggregated joint source representation. These layer aggregation approaches provide a simplistic mechanism to enhance the source-target attention mechanism whilst improving the flow of gradient information from the decoding subnetwork to the encoding layers. In contrast, this work hypothesizes that providing the decoding network more direct access to representations from each encoding layer can further improve the performance of the model and further enhance gradient flow to each encoder layer. Specifically, this work proposes to perform the neural attention computations directly across outputs from different encoding layers via a *Multi-Layer Multi-Head Attention* module.

3. Background

The goal of a seq2seq generation model is to generate the target sequence $y = (y_1, \cdots, y_N)$ of length N given a source sequence $x = (x_1, \cdots, x_M)$ of length *M*, where x_i is the i^{th} source token and y_t is the t^{th} target word. The parameters of the model are learned by maximizing the likelihood function:

$$P(y \mid x; \theta) = \prod_{t=1}^{N} P(y_t \mid y_{<t}, x; \theta) \tag{1}$$

where $y_{<t} = y_1, \cdots, y_{t-1}$ is the generated target sub-sequence. Typically, seq2seq models employ an *encoder-decoder* architecture to model $P(y \mid x; \theta)$. The encoder generates the source semantic representation h^e from a given sentence *x*. Specifically, for each source token x_i, a distributed representation vector $e_i \in \mathbb{R}^d$, where d is the dimension of the vector, is generated by the word embedding layer. Based on the source embedding vectors $E_x = [e_1, e_2, \cdots, e_M]$, the encoder generates the hidden representation $h^e = [h^e_1, h^e_2, \cdots, h^e_M]$. The target sequence *y* is generated by the decoder based on the output of the encoder. During the decoding step *t*, the decoder computes the probability distribution of the target token y_t based on the output of the encoder and the partial target sequence $y_{<t} = y_1, \cdots, y_{t-1}$ as shown in Eq. (1).

The majority of earlier seq2seq architectures are RNN based models (Bahdanau et al., 2015; Cho et al., 2014; Ampomah et al., 2019), but recently architectures employing CNN (Gehring et al., 2017) and self-attention (Vaswani et al., 2017; Shaw et al., 2018) have gained significant attention.

3.1. The Transformer Model

In this work, all experiments and discussions are based on the recently proposed Transformer model (Vaswani et al., 2017). However, the explored attention mechanisms are applicable to other architectures including RNN (LSTM/GRU) based models (Bahdanau et al., 2015; Cho et al., 2014) and CNN (Gehring et al., 2017). The encoder and decoder subnetworks of the Transformer architecture employ attention mechanisms and a standard feed-forward network to model sequences of arbitrary length without the need for a recurrent unit or CNN. The attention operation employed across the different layers are based on the *multi-head attention* (MHA) (see Section 3.1.1).

The encoder subnetwork is composed of a stack of L identical layers. Each encoding layer consists of a *multi-head self-attention* sublayer and a *position-wise feed-forward sublayer* (FFN) sublayer. To ease training and improve performance, residual connection (He et al., 2016) and layer normalization layer (LayerNorm) (Ba et al., 2016) are employed around each sublayer. Formally, the output of each layer l (H_e^l) is computed as:

$$\begin{aligned} S_e^l &= \text{LayerNorm}\left(\text{MHA}(H_e^{l-1}, H_e^{l-1}, H_e^{l-1}) + H_e^{l-1}\right) \\ H_e^l &= \text{LayerNorm}\left(\text{FFN}(S_e^l) + S_e^l\right) \end{aligned} \tag{2}$$

where S_e^l is the output of the multi-head self-attention sublayer computed based on the source sentence representation of the preceding encoder layer $(l-1)$.

The decoder is also composed of a stack of L identical layers. Unlike the encoder subnetwork, each decoding layer consists of three sublayers. Similar to the encoding layer, it has *multi-head self-attention* and FFN sublayers, however, in between them is an *encoder-decoder MHA* sublayer. The *encoder-decoder MHA* sublayer is employed to perform attention computations over the output of the encoder H_e^L. Specifically, the output of each decoding layer l (H_d^l) is computed as:

$$\begin{aligned} S_d^l &= \text{LayerNorm}\left(\text{MHA}(H_d^{l-1}, H_d^{l-1}, H_d^{l-1}) + H_d^{l-1}\right), \\ E_d^l &= \text{LayerNorm}\left(\text{MHA}(S_d^l, H_e^L, H_e^L) + S_d^l\right), \\ H_d^l &= \text{LayerNorm}\left(\text{FFN}(E_d^l) + E_d^l\right) \end{aligned} \tag{3}$$

where S_d^l is the output of the multi-head self attention sublayer computed from the target representation from the preceding decoder layer $(l-1)$. E_d^l is the output of the multi-head encoder-decoder attention sublayer generated based on S_d^l and H_e^L. The top-level layer output (H_d^L) of the decoder is used by a linear transformation layer to generate the target sequence. Specifically, the linear transformation layer via softmax activation computes the output probability distribution over the target vocabulary.

3.1.1. Multi-Head Attention (MHA)

A neural attention mechanism is a crucial component in seq2seq architecture for many sequence generation problems including document summarization (Al-Sabahi et al., 2018) and NMT (He et al., 2018; Bahdanau et al., 2015) etc. The Transformer model uses the scale dot-product attention function. This takes three vectors as input, namely the queries Q, values V and keys K. It maps a given query and key-value pairs to an output which is the weighted sum of the values. The weights indicate the correlation between each query and key. This attention is shown as follows:

$$\begin{aligned} \text{Attention}(Q, K, V) &= \text{softmax}(\alpha)V \\ \alpha &= \text{score}\,(Q, K) \\ \text{score}(Q, K) &= \frac{Q \times K^{\intercal}}{\sqrt{d_k}} \end{aligned} \tag{4}$$

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 Q and K respectively. d_k and d_v are the dimension of the key and value vectors respectively. The dimension of the query is also d_k to allow for the dot-product operation. 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). The overall attention weight distribution is obtained by applying the softmax$(\cdot)$ operation to the attention score $\alpha \in \mathbb{R}^{Z \times J}$.

For better performance, the Transformer architecture uses MHA which is composed of N_h(number of attention heads) scaled dot-product attention operations. Given the Q, K, and V, the multi-head attention is computed as follows:

$$\begin{aligned} \text{MHA}(Q, K, V) &= O \\ O &= HW_o \\ H &= \text{Concat}\,(head_1, head_2, \cdots, head_{N_h}) \\ head_h &= \text{Attention}(QW_h^Q, KW_h^K, VW_h^V) \end{aligned} \tag{5}$$

where QW_h^Q, KW_h^K, and 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 $Q \in \mathbb{R}^{Z \times d_{model}}$. $head_h \in \mathbb{R}^{J \times d_v}$ 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 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 H 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 parameters as the vanilla (single-head) attention if

$$d_k = d_v = \frac{d_{model}}{N_h}$$

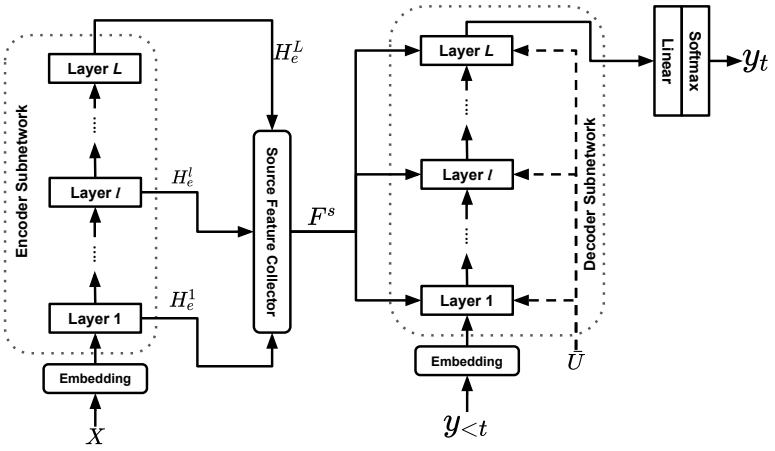


Figure 1: Illustration of the proposed approach to exploiting source representations from multiple encoding layers. X is the input sequence. y_t is the target token generated at step t and $y_{<t}$ is the generated target sub-sequence. F^s is a list of source sentence representations obtained by the *Source Feature Collector* module. The value of $\bar{U} = [\bar{U}_0, \bar{U}_1]$, (where $\bar{U}_i \in \{0, 1\}$) controls the attention computation across the source representations in F^s.

.

In a conventional encoder-decoder architecture (Vaswani et al., 2017; Gehring et al., 2017; Bahdanau et al., 2015) only the source representation from the top-level encoding layer is passed to the decoding subnetwork during the target sequence generation. As the depth of the network increases, it becomes difficult to efficiently train the model due to vanishing and exploding gradients. Furthermore, the encoder employs the entire stack of layers to learn the source semantic information. For a model with a single layer encoder subnetwork, there is a higher possibility that the top-level layer captures most of the necessary information needed to generate the target sequence. In contrast, 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 different time steps and the multiple layers (Wang et al., 2018; Dou et al., 2018). This work presents approaches to exploit source representations learned by multiple layers in the encoder to enhance the flow of information between the encoder and decoder subnetwork during both the forward and backward propagation.

4. Approach

The overall goal is to allow the decoder subnetwork direct access to multiple encoding layers to further enhance the translation performance of the 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 Fig. 1.

The *Source Feature Collector* returns a list of source representations F^s aggregated from outputs of the top n encoding layers. n is considered as a hyperparameter in this work. It is noteworthy that if $n = L$, then F^s contains out representations from all layers in a *L*-layer encoder subnetwork. The seq2seq model (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 $n = 1$. 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 computations are 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. In this work, four multi-layer attention strategies are explored.

Formally, the encoder-decoder multi-head attention sublayer is extended to consider the multiple source representations F^s. To this end, the encoder-decoder MHA is replaced with a MLMHA module as shown in Fig. 2. The computations across each decoder layer (see Eq. (3)) is re-formulated as follows:

$$\begin{aligned} S_d^l &= \text{LayerNorm}\left(\text{MHA}(H_d^{l-1}, H_d^{l-1}, H_d^{l-1}) + H_d^{l-1}\right), \\ E_d^l &= \text{LayerNorm}\left(\text{MLMHA}(S_d^l, F^s, \bar{U}) + S_d^l\right), \\ H_d^l &= \text{LayerNorm}\left(\text{FFN}(E_d^l) + E_d^l\right) \end{aligned} \tag{6}$$

4.1. 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 Fig. 2. 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, α^i is calculated as:

$$\begin{aligned} \alpha^i &= \text{Concat}\left(\alpha_1^i, \alpha_2^i, \cdots, \alpha_{N_h}^i\right) \\ \alpha_h^i &= \text{score}\left(QW_h^q, KW_h^k\right) \end{aligned} \tag{7}$$

where α_h^i is the attention score with respect to the attention head h and 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 $W_h^q \in \mathbb{R}^{d_{model} \times d_k}$, $W_h^k \in \mathbb{R}^{d_{model} \times d_k}$.

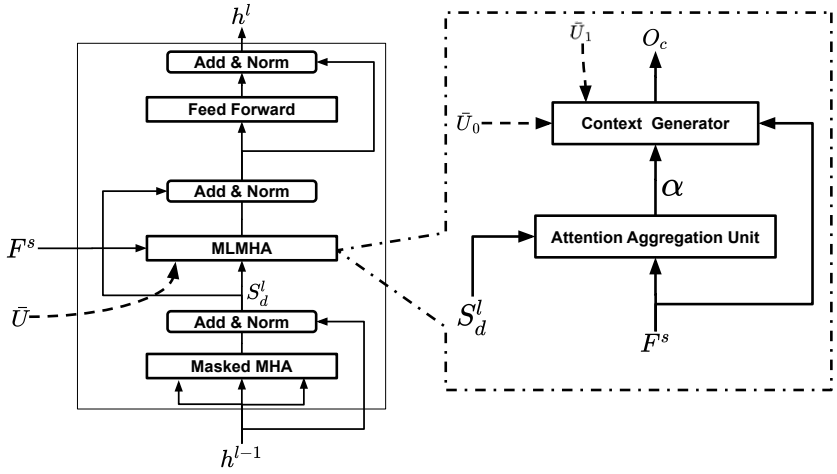


Figure 2: Illustration of a decoder layer with *Multi-Layer Multi-Head Attention* (MLMHA) sublayer to perform the attention computation across multiple features F^s received from the encoding stack. $\alpha = [\alpha^1, \alpha^2, \cdots, \alpha^n]$ is the list of attention weights (where α^i corresponds 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.

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 $c^i \in R^{Z \times d_{model}}$ with respect to f^i. Depending on the value of $\bar{U}_0$, the MLMHA module computes the c^i using either a "layer-specific-attention" weight or a "joint-attention" weight. For the case of $\bar{U}_0 = 1$, $c_h^i \in R^{Z \times d_k}$ (the context vector for the attention head h with respect to f^i) generated using the layer-specific-attention weight softmax(α_h^i) as:

$$c_h^i = \text{softmax}(\alpha_h^i) \cdot VW_h^v \tag{8}$$

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 $\hat{\alpha}$ is employed to obtain c^i for each f^i. $\hat{\alpha}$ is calculated as:

$$\hat{\alpha} = \text{softmax}(\sum_{i=1}^{n} \alpha^i) \tag{9}$$

Analogous to Eq. (8), c_h^i is computed with $\hat{\alpha}$ as:

$$c_h^i = \hat{\alpha}_h \cdot VW_h^v \tag{10}$$

In summary, the c^i with respect to f^i is calculated as:

$$\begin{aligned} c^i &= \text{Concat}\left(c^i_1, c^i_2, \cdots, c^i_{N_h}\right) \\ c^i_h &= \begin{cases} \text{softmax}(\sum_{i=1}^{n} \alpha^i)_h \cdot VW^v_h & \bar{U}_0 = 0 \\ \text{softmax}(\alpha^i_h) \cdot VW^v_h & \bar{U}_0 = 1 \end{cases} \end{aligned} \tag{11}$$

As shown above, the c^i_h with respect to each f^i is generated using the *layer-specific-attention weight* ($\text{softmax}(\alpha^i_h)$) when $\bar{U}_0 = 1$. In contrast for $\bar{U}_0 = 0$, the c^i_h is computed with the joint-attention weight ($\text{softmax}(\sum_{i=1}^{n} \alpha^i)_h$).

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

$$\begin{aligned} O_c &= \hat{C} W_o \\ \hat{C} &= \begin{cases} \text{Concat}\left(c^1, c^2, \cdots, c^n\right) & \bar{U}_1 = 0 \\ \sum_{i=1}^{n} c^i & \bar{U}_1 = 1 \end{cases} \end{aligned} \tag{12}$$

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 d_{model} when contexts-summation ($\bar{U}_1 = 1$) is employed. In contrast, it is equal to $n \cdot d_{model}$ for contexts-concatenation ($\bar{U}_1 = 0$). 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 decoder layer. For simplicity, the model *M-ij* denotes the configuration where $\bar{U}_0 = i$ and $\bar{U}_1 = j$ as summarized in Table 1. As shown, the *M-00* and *M-01* models generate the context c^i_h using the joint-attention weight whilst the layer-specific-attention weights are employed by the *M-10* and *M-11* 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.

5. Experimental Setup

5.1. Datasets

The MLMHA strategies explored in this work are evaluated on the following language translation tasks: Spanish-English (briefly, Es-En), English-Vietnamese (briefly, En-Vi), and English-German (briefly, En-De).

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

Table 1: Models based on the 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 O_c is the overall context vector across the n source representations in F^s.

For the Es-En task, the dataset employed is from the IWSLT 2014 evaluation campaign[1] (Cettolo et al., 2014). The training set comprises of 183k training sentences pairs, and the tst 2014 split is used as the test set. The validation consisting of about 5593 sentence pairs is created by concatenating dev2010, tst2010, tst2011, and tst2012 splits. For the En-Vi translation task, the dataset is from the IWSLT 2015 English-Vietnamese track (Cettolo et al., 2015). The training set consists of 133k sentence pairs. The validation and test sets are from the TED tst 2012 (1553 sentences) and TED tst 2013 (1268 sentence pairs), respectively. For the En-De task, the models are trained on the widely-available WMT'14 dataset comprising of 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.

To alleviate the Out-of-Vocabulary (OOV) problem, a shared vocabulary[2] generated via byte-pair-encoding (BPE)[3] (Sennrich et al., 2016) is employed to encode the source and target sentences. In the case of Es-En, the shared vocabulary comprises of about 34k sub-word tokens. For the En-Vi and En-De translation tasks, the shared vocabulary consists of 21k and 32k sub-word tokens respectively.

5.2. Model Setup

The experiments on the IWSLT tasks are conducted based on the small configuration of the Transformer architecture (Vaswani et al., 2017) 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 models trained on the Es-En and En-Vi tasks consists of a 4-layer encoder subnetwork and

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

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

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

4-layer decoder subnetwork. For experiments on the En-De, the base configuration 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. 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 encoder i.e. $n = L$. That is, on the En-De, and IWSLT tasks n is set as 6 and 4 respectively.

5.3. Training and Inference

For the En-De task, the models are trained for 160k iterations with a batch size of 4960 tokens and a maximum sequence length is limited to 200 sub-word tokens. 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. Besides, the maximum sub-word token length is limited to 150 sub-word tokens. The optimizer employed to train the models in this work is the Adam optimizer (Kingma and Ba, 2014) (with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^9$). Following (So et al., 2019), *single-cosine-cycle* with warm-up is employed as the learning rate scheduling algorithm.

During inference, the target sentences are generated via beam search. For the IWSLT translation tasks, a beam size of 6 and a length penalty of 1.1 is employed. On the WMT'14 En-De task, the beam size of 4 and a length penalty of 0.6 is employed. common practice, the translation quality on the WMT'14 En-De, case-sensitive detokenized BLEU (Papineni et al., 2002) computed with mteval-v13a.pl[4] is employed as the evaluation metric. For the Es-En, case-sensitive BLEU metric with multi-bleu.pl[5] 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[6] (with the signature BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.4.13). The statistical significance is analyzed with paired bootstrap resampling (Koehn, 2004) using compare-mt[7] (Neubig et al., 2019) with 1000 resamples. The source code will be made available at `https://github.com/kaeflint/Multi-layerMHA`.

5.4. Baselines

The Transformer network (Vaswani et al., 2017) is employed as our main baseline models. However across the different languages under consideration, the performance of the MLMHA based models are compared to relevant NMT related works

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

[5] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

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

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

including *Layer Aggregation* based models. For these models, the source-target attention module of each decoder layer receives a joint source representation generated from a combination of the outputs from all the encoder layers. The joint source representation provides each decoding layer an in-direct access to multiple encoding layers. The layer aggregation approaches considered in this work are the *Linear Feature Combination* (Dou et al., 2018; Ampomah et al., 2019), the *Transparent Attention Mechanism* (Bapna et al., 2018) and the *Iterative Feature Combination* (Dou et al., 2018). For the *Linear Feature Combination*, a single joint source representation generated from the weighted combination of the outputs from the encoder layers is passed to all the layers within the decoding subnetwork. Each weight $W^l \in \mathrm{R}^{d_{model} \times d_{model}}$ controls the contribution of the l^{th} encoder layer. In contrast, for a model with N encoding and M decoding layers, *Transparent Attention Mechanism* defines single weight parameter $W \in \mathrm{R}^{(N+1)\times M}$ to generate M different joint source representations (one for each decoding layer) from the weighted combination of outputs from all the encoder layers including the word embedding layer. The *Iterative Feature Combination* proposed by Dou et al. (2018) generates the joint source representation by combining the outputs from the encoder layers in an iterative fashion starting from the lower-level layers. At each combination step s, an aggregation module consisting of a FFN, LayerNorm and residual connections is employed to merge the output from the previous step and the output from encoder layer s. Under the *Transparent Attention Mechanism* and our MLMHA, the decoder receives multiple source representations. For the *Transparent Attention Mechanism*, a joint source representation, generated from a weighted combination of the outputs from all the encoder layers is passed to each decoder layer. However, for the MLMHA mechanism, the outputs from multiple encoder layers are passed directly to each decoder layer without any modification.

6. Results

This section presents the performance evaluations of the MLMHA strategies proposed in this work on the three language translation tasks. For each language pair, the performance obtained for our MLMHA based models is compared to results from existing NMT models. The results on the WMT'14 En-De task are summarized in Table 2. For the IWSLT tasks, Table 3 and Table 4 presents the results on the Es-En and En-Vi tasks respectively. In each table, the value in parentheses represents the translation performance gain over the Transformer baseline model reimplemented in this work. On each translation task, the results obtained for each configuration of the MLMHA shows the impact of the choice of the values of $\bar{\mathrm{U}}_0$ and $\bar{\mathrm{U}}_1$.

As shown in Table 2, only the *Iterative Feature Combination* produced a statistically significant gain over the Transformer baseline among the layer aggregation approaches. The Transparent Attention produced marginal gain ($+0.13$ BLEU) whilst with the Linear Feature Combination, the performance reduced by -0.13 BLEU. For the Transformer models trained with MLMHA, the two contexts-concatenation based

Model	#Params (M)	Train	BLEU
Transformer	61.2	3.65	28.37
With Layer Aggregation			
Transformer + Linear Feature Combination	62.8	3.55	28.24 (−0.13)
Transformer + Iterative Feature Combination	77.0	3.11	28.79 (+0.42)†
Transformer + Transparent Attention	61.2	3.53	28.48 (+0.13)
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)
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)	65.0	-	27.31
Transformer+EM Routing (Dou et al., 2019)	144.8	-	28.81
Transformer+Layer Aggregation (Dou et al., 2018)	121.1	-	28.78
Layer-wise Coordination (He et al., 2018)	-	-	28.33

Table 2: Evaluation of translation performance on the WMT'14 English-German (En-De). #Params and *Train* respectively denote the number of trainable model parameters and the training speed in terms of number of steps/second. "‡" and "†" indicate statistically significant difference with $\rho < 0.01$ and $\rho < 0.05$, respectively.

models (M-00 and M-10) achieved significant gains of +0.43 BLEU and +0.71 BLEU. In contrast, the performance of contexts-summation based models (M-01 and M-11) are statistically insignificant. Table 3 summarizes the performance gains of the *M-ij* models on the IWSLT Spanish-English task. As shown, both the layer aggregation based (except the Layer Feature Combination) and our MLMHA models significantly improve the performance of the Transformer model. Compared to the layer aggregation models, our MLMHA models produced a higher gain in the translation performance. On this dataset, the overall best performance was achieved by the M-00. On the En-Vi translation task, only the M-00, M-01, Iterative Feature Combination and the Transparent Attention approaches produced significant translation quality gains.

The translation results presented in Tables 2 to 4 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 exploit 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

Model	BLEU
Transformer	39.80
With Layer Aggregation	
Transformer + Linear Feature Combination	39.92 (+0.12)
Transformer + Iterative Feature Combination	40.31 (+0.51)†
Transformer + Transparent Attention	40.25 (+0.45)†
With MLMHA	
M-00	40.99 (+1.19) †
M-01	40.61 (+0.81) †
M-10	40.57 (+0.77) †
M-11	40.55 (+0.75) †
Existing NMT Systems	
UEDIN (Cettolo et al., 2014)	37.29
Tied Transformer (Xia et al., 2019)	40.51
Layer-wise Coordination (He et al., 2018)	40.50

Table 3: Evaluation of translation performance on the IWSLT Spanish-English (Es-En). "†" indicates statistically significant difference with $\rho < 0.05$.

M-01 models achieved comparable performance to the the layer aggregation models. Among our proposed models, the M-11 has the overall worse performance with the only significant gain achieved on the Es-En task. In contrast, the M-00 shows a better generalization ability as it consistently achieved statistically significant 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 1. The joint-attention weight is collaboratively computed across the multiple encoder layers' outputs. Compared to employing the layer-specific-attention weights, 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 7.2). 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 2. The layer aggregation approaches have lower impact on the training speed. For example, the Linear Feature Combination and Transparent Attention techniques degrade the speed by about 0.12 steps/second. The MLMHA introduce additional trainable parameters as each decoder layer employs 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. This decreases the

Model	BLEU
Transformer	30.58
With Layer Aggregation	
Transformer + Linear Feature Combination	30.91 (+0.33)
Transformer + Iterative Feature Combination	31.13 (+0.55)†
Transformer + Transparent Attention	31.16 (+0.58)†
With MLMHA	
M-00	30.88 (+0.30)†
M-01	31.07 (+0.49)†
M-10	30.71 (+0.13)
M-11	30.78 (+0.17)
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

Table 4: Evaluation performance on the IWSLT English-Vietnamese translation task. "†" indicates statistically significant difference with $\rho < 0.05$.

training speed as more effort is required to efficiently optimize the parameters of the MLMHA based models. Section 7.2 further investigates the computational complexities of the MLMHA. Overall, based on the translation performance summarized in Tables 2 to 4, this work recommends the MLMHA with contexts-concatenation strategy to combine the contextual representations generated across the outputs from the encoder layers. The joint-attention weight technique is recommended for shallow networks of fewer number of layers, however for deeper networks, we suggest using the layer-specific-attention weight to compute the contextual representation c^i with respect to each source representation in F^s.

7. Analysis

Table 5 shows sample translations from the *M-ij* models and the Transformer baseline on the En-De translation task . This section presents further analyses performed to better understand the impact of the proposed MLMHA strategies on the performance of the Transformer model. This includes analysis to understand (a) impact on the translation quality for each *M-ij* configuration with respect to the source sentence length, (b) the impact of varying the number of source representations considered (the hyperparameter n from the *Source Feature Collector* module) on the performance of the MLMHA strategies, (c) impact on the encoder self-attention with respect to each MLMHA configuration and (d) an ablation study is conducted to understand

Source	The Aachen resident suffered serious injuries and had to be taken to the hospital for treatment.
Target	Der Aachener erlitt schwere Verletzungen und musste zur Behandlung ins Krankenhaus gebracht werden.
Baseline	Der Wohnsitz Aachen erlitt schwere Verletzungen und musste in das Krankenhaus für die Behandlung gebracht werden.
	With MLMHA
M-00	Der Aachener Resident erlitt schwere Verletzungen und musste zur Behandlung ins Krankenhaus gebracht werden.
M-01	Der Aachener Wohnsitz erlitt schwere Verletzungen und musste zur Behandlung ins Krankenhaus gebracht werden.
M-10	Der Aachener Einwohner erlitt schwere Verletzungen und musste zur Behandlung ins Krankenhaus gebracht werden.
M-11	Der Wohnsitz Aachens erlitt schwere Verletzungen und musste in das Krankenhaus gebracht werden.
Source	When the fire service arrived, the flames were already bursting out of a window.
Target	Als die Feuerwehr eintraf, schlugen die Flammen bereits aus einem Fenster.
Baseline	Als der Feuerwehr eintrat, wurden die Flammen bereits aus einem Fenster begraben.
	With MLMHA
M-00	Als der Feuerdienst eintraf, platzten die Flammen bereits aus einem Fenster.
M-01	Als der Feuerdienst eintraf, brannten die Flammen bereits aus einem Fenster.
M-10	Als der Feuerwehr eintraf, platzten die Flammen bereits aus einem Fenster.
M-11	Als der Feuerdienst ankam, brannten die Flammen bereits aus einem Fenster.

Table 5: Sample translations on the En-De task from the Transformer baseline and our MLMHA based models.

the contribution of each encoder layer to the overall translation performance of each *M-ij* model. These analyses are performed on the WMT'14 En-De due to the size of the dataset as well as the number of layers employed to train the models. For simplicity, each analysis is based on the only Transformer baseline and our *M-ij* models.

7.1. Length of source sentence

Capturing efficiently the contextual information, as well as the long-distance dependencies between the tokens of the source sentence, can significantly enhance the translation quality on longer sentences (Dou et al., 2018). Following (Luong et al.,

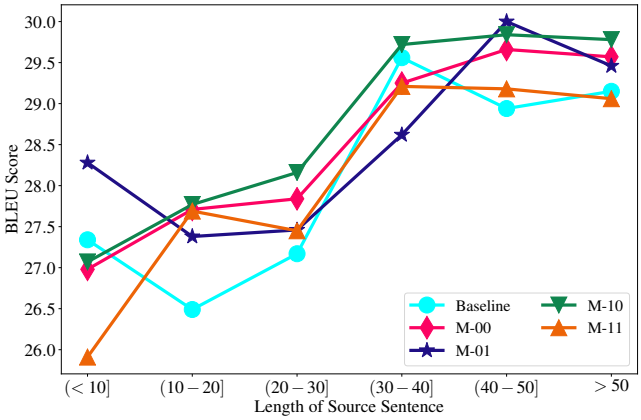


Figure 3: BLEU scores on the WMT'14 En-De test set for the Transformer baseline model, and the MLMHA models with respect to the different source sentence lengths.

2015), sentences of similar lengths (in terms of the number of source tokens) are grouped together. The choice of range for the grouping is based on the sentence lengths (the number of sub-word tokens in each source sentence) across the En-De test set. About 62% of the sentences (1,839) have sequence lengths less than 31 sub-word 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. For each group, the BLEU score is calculated for outputs from the models under consideration. As can be seen in the Fig. 3 , the performance of the baseline model (Transformer) generally improves with increasing input sentence lengths especially for lengths between 10 and 40 sub-word 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.

As shown in Fig. 3, across the sentences with lengths greater than 10, some of our 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 less than 10 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 MLMHA 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 (less than 10 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 2. Overall, the perfor-

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
M-00	n=2	67.5	3.41	28.43
	n=3	73.8	3.18	28.53
	n=4	80.1	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	80.1	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	80.1	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	80.1	2.83	28.62
	n=6	84.9	2.70	28.51

Table 6: Impact of n (the number of encoding layers considered by the *Source Feature Collector* module) on the performance of our MLMHA 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.

mance of the *M-ij* models obtained across the different groups motivates the hypothesis that the MLMHA sublayers within the decoding subnetwork further improves the performance of the self-attention sublayers of the encoder at capturing efficiently and effectively the global dependencies between words of the input sentence. Section 7.3 explores the impact of the MLMHA on self-attention unit of each encoding layer.

7.2. Impact of the hyperparameter *n*

As shown in the Tables 2 to 4, performing the encoder-decoder multi-head attention across 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. Specifically, each MLMHA based model is trained with different values of n. The results are summarized in Fig. 4. As seen, for all the models, there is (in most cases) a significant change in performance as the value of n increases from 1 to 6. As mentioned in Section 4, $n = 1$ correspond to the Transformer baseline model which employs output from only the top-level encoder layer.

Model Complexity

The training speed or computation speed of any given model is affected by the model size, the optimizer employed as well as any other computations that directly modify or alter the formulation of the network structure (Popel and Bojar, 2018). As shown in Section 4.1, 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. 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 from Table 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. As shown in Table 6, 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.

Among the our proposed models, the M-00 and M-10 have the worst training speed compared to the M-01 and M-11 models. The number of new parameters is dependent on the strategy employed to generate the joint context O_c (see Eqs. (7) to (12)) across the multiple representations from the encoder subnetwork. When $n = 6$, the MLMHA introduces about 23.7M new parameters due the n different weights employed to perform the MHA operations across each encoder output as shown in Eq. 7 and Eq. 8. Finally 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 6, for our MLMHA models, as the value of n increases, there is a corresponding reduction in the training speed from about 7.13% (when $n = 2$) to 29.04% (for $n = 6$). The configurations of the B1 and B2 models result in similar increase in the number of parameters as that of the MLMHA model (when $n = 6$). For example, the B1 model

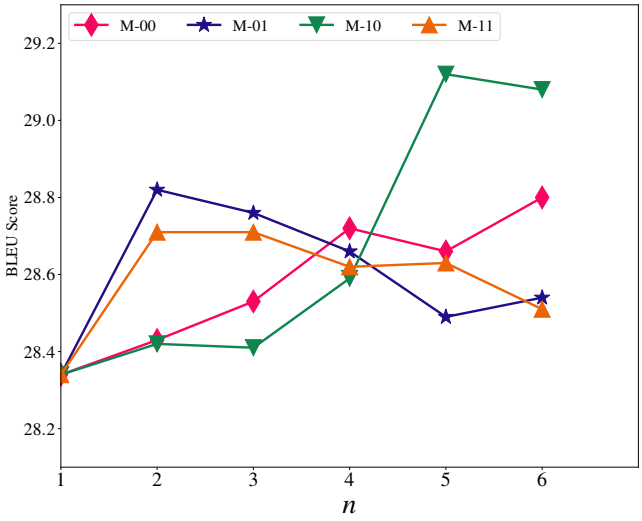


Figure 4: Variation in the translation quality across our MLMHA based models for different values of $n \leq L$.

has roughly the same number of parameters as the M-01 and M-11 models. However, the training speed of these contexts-summation models is (slightly) slower than B1. This is attributed to the additional attention computations and aggregations across the multiple encoding layers.

Translation Quality

Generally, the performance of a neural network model can be improved by either adding more layers or increasing the size of the hidden layers. As shown in Table 6, adding four encoding and decoding layers (in the case of B2) enhanced the translation quality by about $+0.2$ BLEU. For all our models, the improvement in the translation quality across the different values of n comes with an increase in the number of parameters as a result of the additional attention computations. However, unlike B1 and B2 models, we attribute the improvement in the BLEU score to the MLMHA sublayer computing a joint contextual representation O_c from the multiple source representations from the encoder. For example the M-10 (when $n = 5$) and the B1 have roughly identical number of parameters, however whilst M-10 model significantly improves the performance of B0 by $+0.75$ BLEU ($\rho < 0.01$), the B1 achieved a marginal improvement of 0.12 BLEU.

The values of n and $\bar{U}$ are shown to affect the overall translation performance of the MLMHA models. The performance of the contexts-concatenation based models (M-00 and M-10) generally improves as the number of encoder layers considered (n)

increases. These MLMHA models achieve their best performance for the values of $n > 4$ and their worst performance when $n < 4$. In contrast, the contexts-summation based models (with $\bar{U}_1 = 1$), M-01, and M-11 achieved their highest performance when $n = 2$, but the performance degrades for $n > 2$ (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 outputs from the top two encoder layers are considered. Unfortunately, these models show no statistically significant BLEU improvement over the baselines (B1 and B2) with a comparable number of parameters when $n = 6$. Among the contexts-concatenation models, only the M-10 achieved significant improvement of $+0.52$ BLEU ($\rho < 0.05$) over the B2 baseline model.

Unlike the contexts-concatenation based models, the performance of contexts-summation based models decreases as the value of n increases. This can be attributed to the fact that for the models with $\bar{U}_1 = 1$, the summation of the contextual information calculated across the source features F^s has the risk of losing some important contextual information for larger values of n. The context concatenation operation, on the other hand, preserves much of the contextual information which as shown in Table 6 improves the model's performance for larger values of n. Overall, the results obtained by the MLMHA models prove that performing the encoder-decoder attention across multiple encoder layers can further improve the performance of the NMT model. But the performance gain comes at a higher computational cost especially in the case of M-00 and M-10 models.

7.3. 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. These attention heads capture structural information at varying degrees. As noted by Raganato et al. (2018) and Vig and Belinkov (2019), while some self-attention heads focus on long-distance relationships, other heads capture the shorter distance relationships between the input tokens. This allows the Transformer model to capture effectively the structural information for the given source sentence to improve the performance (Raganato et al., 2018). As stated earlier, the operations of the MLMHA module within each decoding layer affects how the source information is processed across the layer of the encoder subnetwork. Following (Vig and Belinkov, 2019), this hypothesis is tested by analyzing the attention entropy as well as 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:

$$\bar{D}_h^l = \frac{\sum_{x \in X} \sum_{i=1}^{|x|} \sum_{y=1}^{i} w_{i,j}^h \cdot (i-j)}{\sum_{x \in X} \sum_{i=1}^{|x|} \sum_{y=1}^{i} w_{i,j}^h} \tag{13}$$

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 sentences. 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:

$$\bar{D}^l = \frac{1}{N_h} \cdot \sum_{h=1}^{N_h} \bar{D}_h^l \tag{14}$$

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:

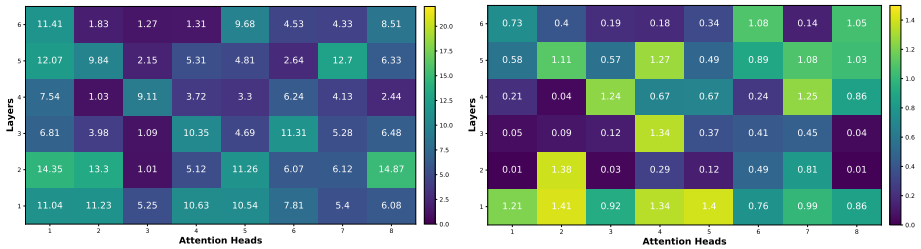
$$E_h^l(x_i) = -\sum_{j=1}^{i} w_{i,j}^h \log w_{i,j}^h \tag{15}$$

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

$$E^l(x_i) = \frac{1}{N_h} \sum_{h=1}^{N_h} E_h^l(x_i) \tag{16}$$

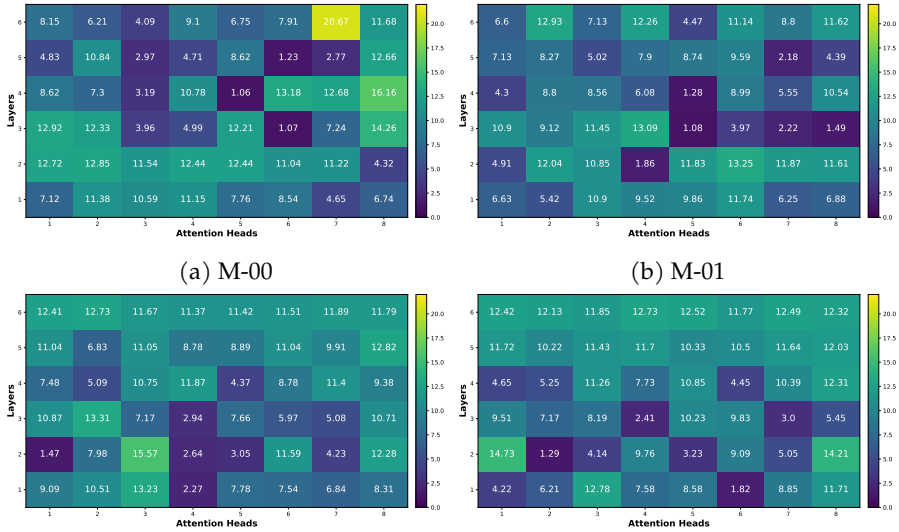
Attention heads with higher entropy are termed as having a more dispersed attention pattern while the lower the entropy, the more concentrated the attention weight distribution.

The attention distance and entropy of attention distribution analysis are performed based on the attention weights generated for 1500 randomly sampled sentences from the En-De task's test split (newstest2014). Fig. 5 and Figs. 6 and 7 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 our MLMHA models respectively. As shown, while some heads focus on the shorter-distance relationships, other heads capture the longer-distance relations among the



(a) (b)

Figure 5: Variation of of the mean attention distance span and attention distribution entropy with respect to the encoding layers and the attention heads for the Transformer baseline. (a) Mean attention distance. (b) Entropy of attention distribution.

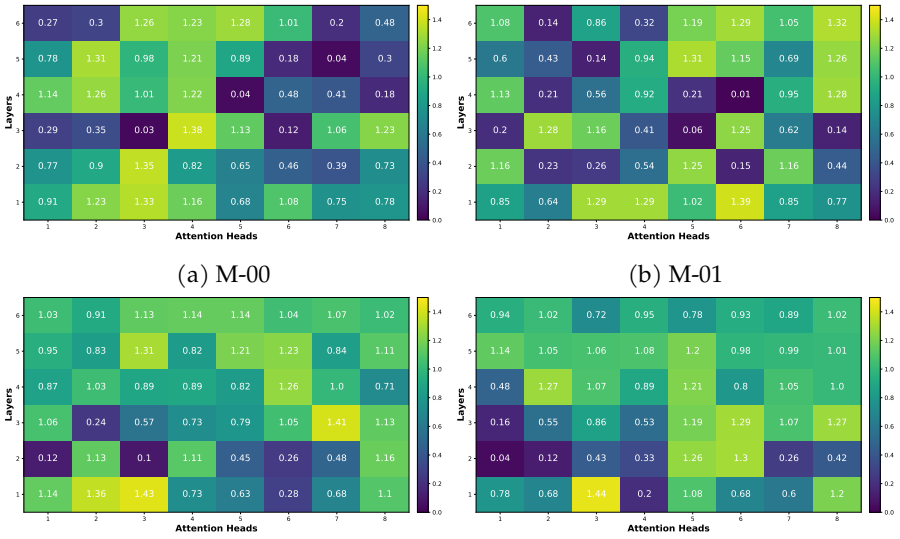


(a) M-00 (b) M-01

(c) M-10 (d) M-11

Figure 6: Variation of the mean attention distance span for the attention heads across the encoding layers with respect to the MLMHA models: (a) M-00, (b) M-01, (c) M-10, and (d) M-11.

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 (Vig and Belinkov, 2019; Ghader and Monz, 2017). Figs. 8 and 9 show the mean average attention distance and entropy for all the self-attention heads across the layers of the encoder respectively. Each plot compares between the Transformer base-



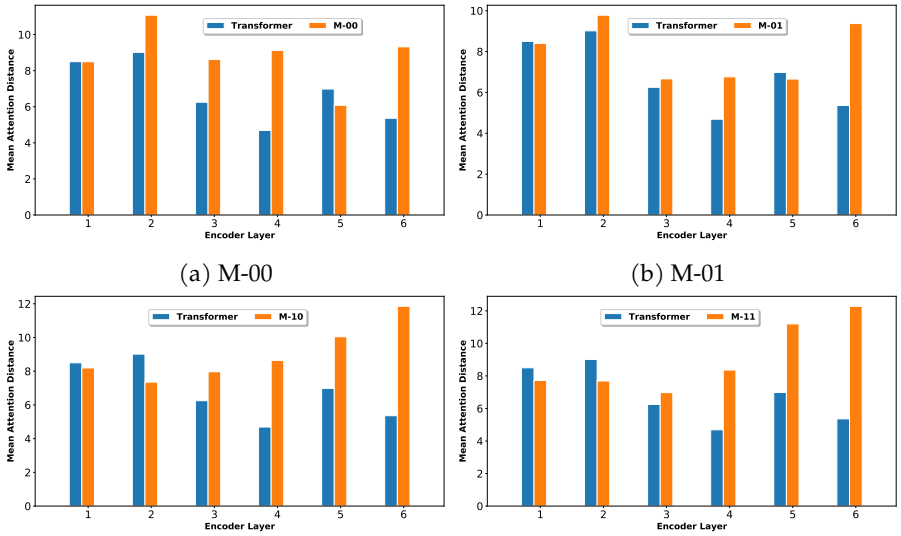
(a) M-00 (b) M-01

(c) M-10 (d) M-11

Figure 7: Variation of the entropy of attention distribution for the attention heads across the encoding layers with respect to the MLMHA models: (a) M-00, (b) M-01, (c) M-10, and (d) M-11.

line and a MLMHA 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. But a higher mean attention distance does not always imply diverse attention distribution. In the subsequent layers, there are a number of attention heads with a higher distance span but with much more concentrated attention weights 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), 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 efficiently learn/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).

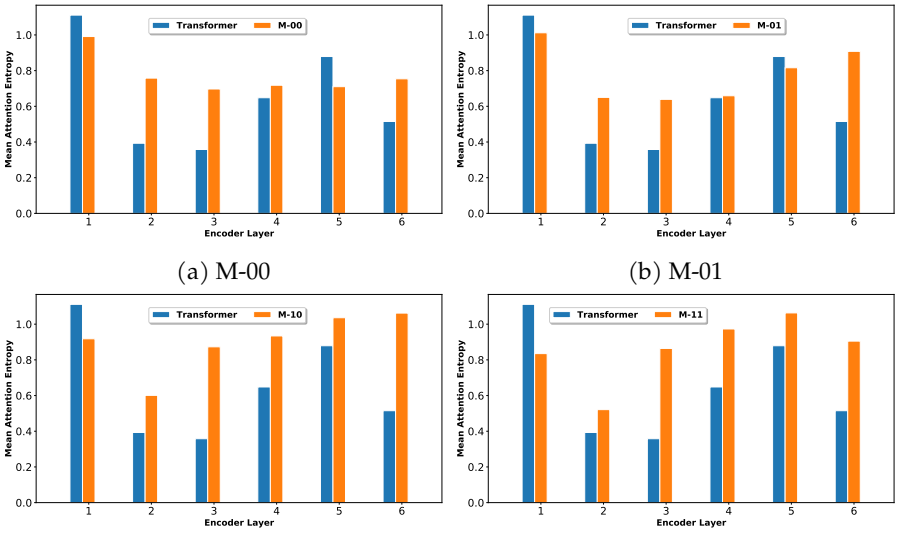


(a) M-00 (b) M-01

(c) M-10 (d) M-11

Figure 8: Variation of the average mean attention distance with respect to each encoder layer for the Transformer baseline model and our MLMHA models: (a) M-00, (b) M-01, (c) M-10, and (d) M-11. Each plot represents the average of all the attention head mean distance with respect to each encoder layer and model.

For the *M-ij* models, the impact of the different MLMHA approaches (employed by the decoder subnetwork) on the self-attention unit within the associated encoding layer is of greater interest. As shown in Figs. 6 to 9, exposing all the encoding layers to the decoding subnetwork can alter how the source information is learned across the encoder subnetwork. 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 $\bar{U}_0$ as evident from Figs. 8 and 9. For example, as displayed in Figs. 9a and 9b and Figs. 8a and 8b, the joint-attention weight models (M-00 and M-01) 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 short-range (local) contextual information within the neighborhood of the input source tokens. In contrast, the layer-specifi-atention weight models (M-10 and M-11) employs the first few layers ($l \leq 3$) to learn the short-term information whilst the upper layers model the long distance interaction between the input tokens as shown in Figs. 9c and 9d and Figs. 8c and 8d. Overall, each MLMHA strategy is shown to modify how source information is captured across the multiple attention heads and layers in the encoder as shown by attention distance and entropy of attention weight distribution. This further enhances the net-



(a) M-00 (b) M-01

(c) M-10 (d) M-11

Figure 9: Variation of the average entropy of head attention distribution across the encoding layers for the Transformer baseline model and our MLMHA models: (a) M-00, (b) M-01, (c) M-10, and (d) M-11. Each plot represents the average entropy of head attention distribution per encoder layer.

work's performance at learning the source semantic information needed to improve the translation quality.

7.4. An ablation study: Encoder Layer Dependency

The translation performance of the MLMHA models reported in Table 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 model. To this end, the translation quality of each MLMHA 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 7 shows the difference in performance of our proposed models for each masked output of the encoder. As shown in most cases masking one of outputs of the encoder layers significantly degrades the translation quality. For example, without the output of the first encoder layer, the performance of both M-00 and M-01 model

	Models			
Layer	M-00	M-01	M-10	M-11
1	-15.25‡	-24.99‡	-0.83‡	-0.26
2	-0.49†	-0.19	-1.32‡	-1.13‡
3	-0.27	-0.09	-1.05‡	-0.83‡
4	0.03	0.13	-1.00‡	-1.43‡
5	-1.79‡	-0.81‡	-1.50‡	-1.34‡
6	-9.85‡	-1.49‡	-1.09‡	-0.10

Table 7: 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 MLMHA models when $n = L$. "‡" and "†" indicate statistically significant difference with $p < 0.01$ and $p < 0.05$, respectively. The base-BLEU scores for the M-00, M-01, M-10 and M-11 are 28.80, 28.54, 29.08 and 28.51, respectively.

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 statistically significant impact on the overall performance of the M-00, M-01 and M-10 models. Overall, the results in Table 7 demonstrates that for 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 7.2, the translation performance of the M-10 model is shown to be highly dependent on source representations from all encoder layers. Removing the output of any of these layers cause statistically significant change in performance.

8. Conclusion

In this work, the performance of the Transformer model is improved by exploiting multiple source representations captured by different encoding layers. Specifically, the decoding subnetwork is allowed direct access to the entire stack of encoding layers to extract better source-target contextual information. This technique also improves the flow of gradient information between the two subnetworks. Experimental results on IWSLT tasks (Spanish-English and English-Vietnamese) and on the WMT'14 English-German translation task show that the proposed MLMHA module can further improve the performance of the Transformer baseline. However, the analysis performed reveals that the performance gain is dependent on the values of the binary vector $\bar{U}$ and n (the number of encoding layers considered by the MLMHA module). Overall, the MLMHA with joint-attention weight ($\bar{U}_0 = 0$) showed better

generalization than with $\bar{U}_0 = 1$ across all the translation tasks under consideration. Further analysis also reveals that directly exposing the layers of the encoder subnetwork alters significantly how the global and local source contextual information is captured by the self-MHA sublayer employed within each encoder layer.

Future works include evaluating the performance of the *MLMHA* module on other NLP tasks such as document summarization and machine reading comprehension. Another interesting direction will consider investigating the potential performance gain from the combination of the MLMHA module and Layer Aggregation approaches such as the Transparent Attention (Bapna et al., 2018).

Bibliography

Al-Sabahi, Kamal, Zhang Zuping, and Mohammed Nadher. A hierarchical structured self-attentive model for extractive document summarization (HSSAS). *IEEE Access*, 6:24205–24212, 2018. doi: 10.1109/ACCESS.2018.2829199.

Ampomah, Isaac KE, 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*, pages 92–104. Springer, 2019. doi: 10.1007/978-3-030-23281-8_8.

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

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

Bapna, Ankur, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. Training Deeper Neural Machine Translation Models with Transparent Attention. In *Proceedings of the 2018 Conference on EMNLP*, pages 3028–3033, 2018. doi: 10.18653/v1/D18-1338.

Belinkov, Yonatan, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Evaluating Layers of Representation in Neural Machine Translation on Part-of-Speech and Semantic Tagging Tasks. In *Proceedings of the 8th IJCNLP*, pages 1–10, 2017.

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

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

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

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014. doi: 10.3115/v1/D14-1179.

Dou, Zi-Yi, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. Exploiting Deep Representations for Neural Machine Translation. In *Proceedings of the 2018 Conference on EMNLP*, pages 4253–4262. ACL, 2018. doi: 10.18653/v1/D18-1457.

Dou, Zi-Yi, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. Dynamic layer aggregation for neural machine translation with routing-by-agreement. *arXiv:1902.05770*, 2019. doi: 10.1609/aaai.v33i01.330186.

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

Ghader, Hamidreza and Christof Monz. What does Attention in Neural Machine Translation Pay Attention to? In *Proceedings of the 8th IJCNLP*, pages 30–39, 2017.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

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

Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on CVPR*, pages 4700–4708, 2017. doi: 10.1109/CVPR.2017.243.

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

Irie, Kazuki, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Language Modeling with Deep Transformers. *Proc. Interspeech 2019*, pages 3905–3909, 2019. doi: 10.21437/Interspeech.2019-2225.

Kingma, Diederik P and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014.

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

Koehn, Philipp and Josh Schroeder. Experiments in domain adaptation for statistical machine translation. In *Proc. 2nd WMT*, pages 224–227, 2007. doi: 10.3115/1626355.1626388.

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

Luong, Thang, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proc. EMNLP*, pages 1412–1421, Lisbon, Portugal, Sept. 2015. ACL. doi: 10.18653/v1/D15-1166.

Neubig, Graham, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. compare-mt: A Tool for Holistic Comparison of Language Generation Systems. In *Proceedings of the 2019 Conference of the NAACL*, pages 35–41, 2019. doi: 10.18653/v1/N19-4007.

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

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on ACL*, pages 311–318. ACL, 2002. doi: 10.3115/1073083.1073135.

Popel, Martin and Ondřej Bojar. Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*, (110):43–70, 2018. doi: 10.2478/pralin-2018-0002.

Raganato, Alessandro, Jörg Tiedemann, et al. An analysis of encoder representations in transformer-based machine translation. In *Proc. of EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. ACL, 2018. doi: 10.18653/v1/W18-5431.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proc. ACL*, pages 1715–1725, 2016. doi: 10.18653/v1/P16-1162.

Shaw, Peter, Jakob Uszkoreit, and Ashish Vaswani. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the NAACL:HLT, Volume 2 (Short Papers)*, pages 464–468, 2018. doi: 10.18653/v1/N18-2074.

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

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

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

Vig, Jesse and Yonatan Belinkov. Analyzing the Structure of Attention in a Transformer Language Model. *arXiv preprint arXiv:1906.04284*, 2019. doi: 10.18653/v1/W19-4808.

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

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

Xia, Yingce, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. Tied transformers: Neural machine translation with shared encoder and decoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5466–5473, 2019. doi: 10.1609/aaai.v33i01.33015466.

Yu, Fisher, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on CVPR*, pages 2403–2412, 2018. doi: 10.1109/CVPR.2018.00255.

Address for correspondence:
Isaac Kojo Essel Ampomah
ampomah-i@ulster.ac.uk
School of Computing, Ulster University,
York Street, Belfast, County Antrim, BT15 1ED, United Kingdom