Heterogeneous Encoders Scaling In The Transformer For Neural Machine Translation

Jia Cheng Hu[1,*], Roberto Cavicchioli[2], Giulia Berardinelli[1] and Alessandro Capotondi[1]

[1]*Department of Physics, Informatics and Mathematics (University of Modena and Reggio Emilia), via Campi 213/A, Modena, 41125, Italy*

[2]*Department of Communication and Economics (University of Modena and Reggio Emilia), viale Antonio Allegri 9, Reggio Emilia, 42121, Italy*

Abstract

Although the Transformer is currently the best-performing architecture in the homogeneous configuration (self-attention only) in Neural Machine Translation, many State-of-the-Art models in Natural Language Processing are made of a combination of different Deep Learning approaches. However, these models often focus on combining a couple of techniques only and it is unclear why some methods are chosen over others. In this work, we investigate the effectiveness of integrating an increasing number of heterogeneous methods. Based on a simple combination strategy and performance-driven synergy criteria, we designed the Multi-Encoder Transformer, which consists of up to five diverse encoders. Results showcased that our approach can improve the quality of the translation across a variety of languages and dataset sizes and it is particularly effective in low-resource languages where we observed a maximum increase of 7.16 BLEU compared to the single-encoder model.

Keywords

Machine-Translation, Transformer, Encoder, Scaling, Low-resource-language

1. Introduction

Neural Machine Translation (NMT), in Natural Language Processing (NLP), describes the problem of automatically translating a text sentence from one language into another using Neural Networks. Over the past years, translation systems performed increasingly better thanks to the design of more sophisticated and effective models. Models consist of an encoder that extracts a representation from the input sequence in the source language and a decoder that generates the same sentence in the target language.

There are currently several types of neural processing methods. The most popular ones consist of self-attentive networks (SANs or ANNs) [2], recurrent neural networks [3] (RNNs) and convolutional (CNN) ones [4, 5, 6]. More recently, the works of [7] and [8, 9] proposed two additional variants. These methods differ in the way they process a sequence: for instance, recurrent neural networks [3] apply the same operation over the sequence in a sequential

manner and each element is given access to the previous ones thanks to one or two hidden states. Self-attentive neural networks [2] instead, make use of the Attention mechanism [10, 11], a method that constructs representations made of the Softmax-weighted sum of all the elements in the sequence. Each of these approaches presents advantages and drawbacks on its own. For instance, RNNs better capture the sequentiality in the data compared to stateless methods [12], such as CNNs and SANs, that rely on simple numeric encodings. However, as a drawback, the first suffers from computational limitations and a limited receptive field.

In the presence of a variety of possible processing strategies, State-of-the-Art models often leverage the strength of a combination of multiple approaches in both NLP [13, 14, 15, 16, 17] and Vision [9, 12]. The effectiveness of hybrid models outlines the benefits of introducing heterogeneous ways of processing inside the same architecture. However, often in these architectures, only two methods are considered and to the best of our knowledge, little has been done on the combination of a larger number of heterogeneous encoding methods. Additionally, different techniques in State-of-the-Art models are often closely intertwined in one single processing block. As a matter of fact, there is a lack of consensus on how to combine effectively each technique. This type of integration approach is limiting in terms of applicability since not all encoding strategies can be easily combined. For instance, in a single encoder architecture, combining either the convolution or the self-attention with recurrent methods ultimately results in a recurrent encoder.

In this work, we propose an investigation into the effectiveness of combining an increasing number of diverse neural network processing methods. In particular, we adopt the Transformer decoder [2] and an encoder that is made of an increasing number of heterogeneous methods. Moreover, we propose a very simple encoder combination strategy such as the simple sum. On one hand, it is bound to be less performing than *ad-hoc* hybrid designs. On the other, its simplicity enables the addition of an arbitrary number of independent and diverse encoders.

The paper is organized as follows. Section 2 presents related works and several encoding techniques. Then, in Section 3, we introduce the Multi-Encoder Transformer, which encoder combines up to five processing methods: Self-Attention, Convolutional, LSTM, Fourier Transform (FNet), and Static Expansion. In Section 4, we present five translation datasets on which our models are trained and tested. Finally, in Section 5, we present the results and discuss the advantages and limitations of our multi-encoder transformer. In particular, we discover that different methods synergize differently when combined with each other and that low-resource languages benefit the most from the increase in the number of heterogeneous encoders.

In summary, our contributions are as follows: *(i)* we analyze the effectiveness of simply summing multiple encoding strategies in the Transformer, in particular adopting the RNN, CNN, SAN, Static Expansion and Fourier Transform (FNet) across a variety of Language translation tasks; *(ii)* we analyze the synergies of the five processing methods and design a dual-encoder, triple encoder, quadruple and quintuple encoder based upon the results; *(iii)* we show that the multi-encoding achieves better performances with respect to the baseline Transformer, despite each encoder presenting a poorer performance in the single instance; *(iv)* we show that low-resource languages are those that benefit the most from combining heterogeneous encoders.

2. Related Works

Many works in several research fields explored the benefits of combining multiple, sometimes complementary, encoding strategies. QRNNs [5] tackled the computational burden of RNNs and proposed to exploit the efficiency of the convolution operation while preserving the strong sequential awareness using recurrent pooling layers in various NLP tasks. In Neural Machine Translation [13] integrated the self-attention layer in the GNMT [18], a full RNN network based on stacks of LSTM networks [19]. Conversely, [14] explored the opposite solution and integrated the LSTM inside the Transformer. The works of [20] and [15] merged the self-attention method and convolution on a deep level, forming a moving window in which relationship among sequence tokens are extracted on a local level instead of a global one. On a similar level, in Audio Speech Recognition, best-performing architectures such as the Conformer [17] and SqueezeFormer [16] rely on a combination of convolution layers and self-attentive blocks to capture both local and global correlations respectively in the spectrum representation. Similar ideas were applied in other tasks such as Reading Comprehension [21] and Image Classification [22]. The adoption of multiple encoders is natural for some problems, such as those involving multiple input sequences [23, 24] which also led to the investigation of optimal interconnection between the encoders and the decoder [25, 14, 13]. Also, the scalability of Transformers has been deeply investigated in a variety of contexts [26, 27, 28, 29, 30] which mostly focus on increasing the number of layers or sequence length.

In contrast to these works, our work is the only one that investigates the effectiveness of increasing the number of heterogeneous encoding methods in the Transformer and present hybrid architectures made of an unprecedented number of strategies.

3. Heterogeneous Multi-Encoder

3.1. Encoding Strategies

Given the input sequence $X = \{x_1, x_2, \ldots, x_N\}$, $x_i \in \mathbb{R}^d$ and d the hidden dimension, we describe five encoding strategies based on recurrency, convolution, self-attention, Fourier transform and static expansion.

LSTM. Recurrent Neural Networks are sequence modelling methods inspired by recurrent functions, which output at time step $t \in \{1, 2, \ldots, N\}$ are affected by all the input of the previous steps by means of the hidden states. The hidden state consists of one or multiple vectors depending on the architecture. In this work, we consider the LSTM [19] encoder whose formula is showcased below:

$$\begin{cases} f_t = sigmoid(W_f x_{-1} + U_f h_{-1} + b_f) \\ i_t = sigmoid(W_i x_{-1} + U_i h_{-1} + b_i) \\ o_t = sigmoid(W_o x_{-1} + U_o h_{-1} + b_o) \\ z_t = tanh(W_c x_{-1} + U_c h_{-1} + b_c) \\ c_t = f_t \odot c_{t-1} + i_t \odot z_t \\ h_t = o_t \odot tanh(c_t) \end{cases}$$

where $W, U \in \mathbb{R}^{4 \cdot d \times d}$, $b \in \mathbb{R}^{4 \cdot d}$ and f_t, i_t, o_t are the forget, input, output gates, whereas

h_t, c_t represent the hidden states. We restrict to the uni-directional formulation because the bidirectional processing of the input is already provided by the bidirectional nature of self-attention, which supports the goal of experimenting with diverse encoding strategies.

ConvS2S. Convolution networks were partially introduced in Natural Language Processing to overcome the computational inefficiency of recurrent models [5] but later proved to be an effective alternative to other sequence modelling architectures and are even able to achieve State-of-the-Art performances [6, 15, 31, 20]. In this work, we will focus on the ConvS2S encoder [6]. Similar to the Vision case, its working principle lies in the filter banks learning the local and most simple semantics among elements in the sequence during earlier stages and extracting longer and more complex relationships in the higher-order layers. Given an input sequence X and an odd kernel size k, the encoder is made of a set of filters $W \in \mathbb{R}^{2 \cdot k \times 2 \cdot d}$ that perform the convolution operation for each time step $t = \{1, 2, \ldots, N\}$ in the input sequence. The output dimension is twice the model dimension in order to be fed into the GLU function, which represents the non-linear component of the layer:

$$GLU(a, b) = a \otimes sigmoid(b)$$

where $a, b \in \mathbb{R}^d$ are the two halves of the output vector.

Self-Attention. The attention mechanism was first introduced in NLP in [10], and it was applied to recurrent models drastically improving the performances by allowing the encoder information to be distributed on each encoder input rather than one single state vector. The method was later refined in [11] and the two works provided the foundation of the Transformer [2]. Its main component, the self-attentive layer, consists of the following operations (considering the single-head case):

$$Self\text{-}Attention(Q, K, V) = softmax(\frac{QK^\intercal}{\sqrt{d}})V$$

where $Q, K, V \in \mathbb{R}^{N \times d}$ are linear projections of the same input. The method has been proven to be able to extract the syntax and semantics as well as capture long-range relationships of the sequence thanks to the global and bidirectional access to the whole input right at the first stage of the network compared to the previous method.

Static Expansion. The Static Expansion [9] proposes to distribute the sequence into an arbitrary number of elements. In its essence, the idea consists of a Forward phase where the input of length is transformed into a new sequence one featuring a new target length, and a Backward where it's transformed back to the original one. In practice, the input is linearly projected four times, producing $K, V_A, V_B, S \in \mathbb{R}^{N \times d}$. Given the desired length N_E, $2 \cdot N_E$ expansion vectors are considered, denoted as $E_Q, E_B \in \mathbb{R}^{N_E \times d}$. The static expansion layer performs three operations.

(i). First, the dot product similarity is computed between the expanded queries and the keys obtaining the length transformation matrix L:

$$L = \frac{E_Q K^\intercal}{\sqrt{d}} \tag{1}$$

The result is fed into a ReLU function and normalized. The sequences featuring new lengths are computed in the following way:

$$P_i^{fw} = \Psi(ReLU((-1)^{i+1}L), \epsilon) \quad i \in 1, 2$$
$$F_i^{fw} = P_i^{fw}(V_i + E_i) \quad i \in 1, 2 \tag{2}$$

$\Psi : (X, \epsilon) \to Y, X, Y \in \mathbb{R}^{N_1 \times N_2}$ is the normalization function and $\epsilon \in \mathbb{R}_{>0}$ the coefficient ensuring numeric stability.

(ii). Using the same matrix of Equation 1, the sequence is transformed back to its original length (Backward step):

$$P_i^{bw} = \Psi(ReLU((-1)^{i+1}L^\intercal), \epsilon) \quad i \in 1, 2$$
$$B_i^{bw} = ReLU(P_i^{bw}F_i^{fw}) \quad i \in 1, 2 \tag{3}$$

(iii). The two results are combined through a sigmoid gate (sigmoid function indicated with σ):

$$out = \sigma(C) \odot B_1^{bw} + (1 - \sigma(C)) \odot B_2^{bw} \tag{4}$$

It can be seen in the first version of the Static Expansion, the collection of $2 \cdot N_E$ expansion vectors are learnable parameters of size d. Since the original formulation was developed for Image Captioning instead of NLP tasks, we enrich the formulation by providing each expansion vector more awareness about the sequence by feeding them into a bilinear projection described as:

$$E_Q' = E_Q(\frac{X^\intercal(WX)}{\sqrt{T}})$$
$$E_B' = E_B(\frac{X^\intercal(WX)}{\sqrt{T}}) \tag{5}$$

where $X \in \mathbb{R}^{T \times d}$ is the input sequence, $W \in \mathbb{R}^{d \times d}$ and $E_Q, E_B \in \mathbb{R}^{N_E \times d}$ are learnable weights.

FNet layer. The Fourier Transform describes a mathematical operation that converts a function defined in the time domain into its spectral representation. In particular, it extracts the coefficients of its sinusoidal components localized in the frequency domain. The Discrete Fourier Transform consists of the discrete formulation which is designed to address discontinuous and sampled signals. In [7] they adopt such a method and replace the self-attention in BERT, achieving remarkably close performances despite a parameter-less formulation. The discrete Fourier transform is defined as:

$$f_{k+1} = \sum_{t=0}^{N-1} x_{t+1}e^{-\frac{2\pi i}{N}tk}, \quad 0 \le k \le N - 1$$

which is denoted in vector form as $\mathcal{F}(X) \in \mathbb{C}^{N \times d}$. The FNet layer is made of two DFTs applied over the sequence dimension first and then over the hidden dimension, finally only the real part of the results is preserved:

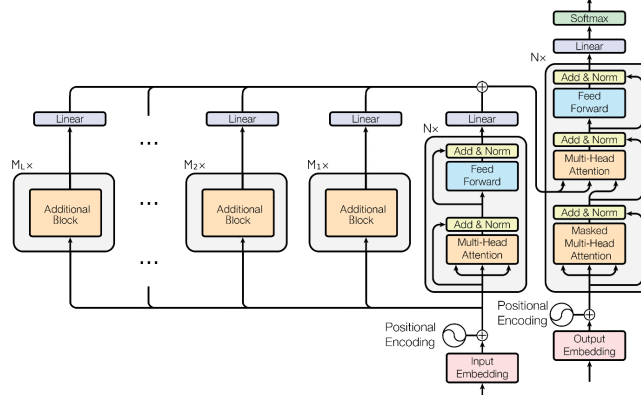$$\mathcal{F}_{FNet}(X) = Re(\mathcal{F}_d(\mathcal{F}_N(X)))$$

Figure 1: Multi-Encoder Transformer.

The mixing of tokens along the two dimensions efficiently provides enough accessibility of the sequence to the higher order units, such as non-linear projections, in order to form meaningful compositions of its elements.

3.2. Multi-Encoder Transformer

The Multi-Encoder Transformer is made of the Transformer itself and multiple encoding blocks joined together by means of a simple sum as depicted in Figure 1. Encoder blocks, depicted in Figure 1, consist possibly of a stack of convolution filters, self-attentive layers, LSTMs, FNet layer, or Static Expansion. All of them include skip-connections and normalization layers, with the exception of the LSTM for the first and ConvS2S for the latter.

The computational burden of the additional encoder is mitigated by the absence, in almost all cases, of the Feed-Forward layer which represents one of the computational bottlenecks of the transformer model. The only two exceptions are the baseline Transformer encoder, where the component is kept to maintain consistency with the original formulation, and Fnet layers, in which the Feed-Forward is necessary for the otherwise parameter-less definition of the module.

4. Experimental setup

4.1. Datasets

We used five datasets in our experiments. The IWSLT 2015 English-Vietnamese (En-Vi) corpus, the IWSLT 2017 English-Italian (En-It) corpus, two TED talks in Galician-English (Gl-En), Slovak-English (Sk-En) and a Spanish-English (Sp-En) corpus provided by the European Language Resource Coordination (ELRC). Low-resource languages commonly refer to less studied, resource scarce, and less commonly taught languages among other definitions [32]. In our work, we denote low-resource languages as those that involve less than 30,000 training pairs. Although "Sp-En" is a high-resource language, the dataset size simulates a low-resource setup.

Sequences whose post-tokenization length is greater than a certain threshold are discarded, which results in a negligible smaller training size but a notably smaller peak GPU memory

Table 1
Data size and training configuration for each translation task.

Source	Language Pair	Total num. pairs	Max epoch	Vocab. size	Training max. length
TED Talks	Gl-En	10017	150	4139	150
ELRC	Sp-En	21007	200	4288	200
TED Talks	Sk-En	61470	350	4232	150
IWSLT15	En-Vi	133317	400	4384	150
IWSLT17	En-It	231619	200	4246	200

footprint. In all training instances, the target and the source language vocabulary are shared. Each vocabulary is created using the BPE algorithm [33]. The maximum number of epochs is chosen based on the number of training steps required for the validation accuracy to start decreasing. For instance, the En-It training requires significantly fewer epochs than En-Vi most likely because of a higher similarity between source and target languages. More Details are reported in Table 1. Analytic experiments are performed mostly on the "En-Vi" pair, whose size is close to the average size of adopted datasets and is preferred over "Sk-En" for better generalization.

4.2. Models

The baseline model denoted as "Base" consists of the Transformer with 6 layers, d=512, num_heads=8, d_{ff}=2048. When an encoder is indicated as "Base", we refer to the "Self-Attention + FeedForward" case. For the multi-encoder Transformer, the number of additional encoders, as well as the number of additional layers for each block, depends on the experiment configuration and is selected in the following range of values 3, 6, 12, 18. The Static Expansion layer uses the following 12 choices of static expansion coefficients { 6, 6, 12, 8, 12, 8, 6, 6, 12, 8, 12, 8 }. The choices of these specific coefficients are arbitrarily made and it is out of the scope of this work to search for the optimal configuration. All kernels in the ConvS2S layers are of width 3.

4.3. Training and Evaluation Details

Words are split according to the Byte Pair Encoding (BPE) [33], and no lower casing is applied. Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$) and Label Smoothing regularization (smoothness $\gamma = 0.1$) are adopted. We use the Noam learning rate scheduler as described in [2] and 4,000 warm-up steps. Training pairs are sampled without replacement using a target length-oriented bucketing until the mini-batch reaches 4,096 words. Finally, models are trained up to the number of epochs reported in Table 1.

We evaluate the performances using the BLEU metric [34]. In particular SacreBLEU [35][1]. The inference algorithm consists of the Beam Search with beam size 4 and in contrast to the standard practice, no checkpoint averaging is done since the selection criteria are often time-based and lead to different results according to the computational resources.

---

[1]SacreBLEU signature: BLEU+case.mixed+ lang.[src-lang]-[dst-lang]+numrefs.1+ smooth.exp+tok.13a+version.2.0.0

Table 2

Single encoder performances on the "En-Vi" test set. "Base" denotes the Self-Attention encoder. The encoding method is reported in the Header, whereas the BLEU is shown in the rows.

Base [38]	Base (ours)	LSTM	ConvS2S	Static Exp	FNet
31.11	31.13	29.21	30.95	29.40	29.56

Table 3

Baseline scores on the "En-Vi" test set. The language pair is reported in the Header, whereas the BLEU is shown in the rows. In Gl-En, Sp-En models are made of 3 layers. In contrast, 6 layers are adopted in the cases of Sk-En, En-Vi and En-It.

Gl-En	Sp-En	Sk-En	En-Vi	En-It
13.03	27.22	27.15	31.13	32.86

5. Results

5.1. Single Encoder Performances

We evaluate the performances of each single encoder strategy by replacing the encoder block in the Transformer (with both six encoding and decoding layers) with all five encoding methods in the "En-Vi" translation task. This data set is chosen for the analytic experiments throughout the rest of the experiments.

Each single encoder performance is reported in Table 2. Since transformers are sensible to hyperparameters and implementation details [36, 37] we show that our baseline achieves a very similar performance compared to the ones typically observed in the literature. In the particular case of [38] it presents a negligible difference of 0.02 BLEU, paving the way for a fair comparison. In the single-encoder case, the baseline model outperforms all the other ones. Table 3 reports the baseline performances across all translation tasks. Hence, the Self-Attention method performs better than the others in the single-encoder configuration.

5.2. Dual-encoder Transformer and Synergy Study

In Table 4 we show how the Dual-Encoder Transformer performs on the "En-Vi" translation task, when the baseline, whose encoder is made of Self-Attention layers, is combined with other encoding techniques. The performance seems to increase proportionally to the number of additional layers M in some cases. This is particularly evident in the case of the LSTM whereas the impact of increasing the number of layers is less significant in other cases. The number of layers for the models presented throughout this work is based on the results in Table 4.

We observe that combining two encoders does not guarantee the overall system performs at least as well as the single encoder case. All dual-encoders yielded a worse score compared to the baseline with the exception of the "Base + LSTM$_{M=18}$" and "Base + Static Exp$_{M=12}$" which instead scored better with a margin of 0.09 and 0.60 BLEU. This means that such improvements are caused by neither the architectural structure nor the increase of parameters alone since other instances performed similarly if not worse. This claim is further supported by the poor

Table 4

Performances comparison of the Dual-Encoder Transformer made of the baseline and other encoding methods on the "En-Vi" test set.

Encoder	BLEU
Base	31.13
Base + ConvS2S$_{M=6}$	30.34
Base + ConvS2S$_{M=12}$	30.21
Base + FNet$_{M=6}$	29.20
Base + FNet$_{M=12}$	29.38
Base + Static Exp$_{M=6}$	31.51
Base + Static Exp$_{M=12}$	31.73
Base + LSTM$_{M=6}$	29.52
Base + LSTM$_{M=12}$	30.84
Base + LSTM$_{M=18}$	31.22

Table 5

Twin Dual-Encoder Transformer SacreBLEU scores on the "En-Vi" test set. δ represents the difference w.r.t the baseline.

Method	Single-Encoder	Twin Dual-Encoder (δ)
Base	31.13	30.64 (-0.49)
LSTM	29.21	28.32 (-0.89)
ConvS2S	30.95	30.88 (-0.07)
Static Exp	29.40	29.12 (-0.28)
FNet	29.56	26.51 (-3.05)

Table 6

Synergy matrix of dual-encoders on the "En-Vi" test set. For brevity, encoders and the number of layers are denoted with letters. B=Base$_{M=6}$, L=LSTM$_{M=18}$, C=ConvS2S$_{M=6}$, S=Static Exp$_{M=12}$ and F=FNet$_{M=6}$. Each entry in row i and column j represents the difference in terms of BLEU score between the dual encoder $i + j$ and the single encoder performance of method j.

	B	L	C	S	F
B	-0.49	+2.01	-0.61	+2.33	-2.19
L	+0.09	-0.89	-0.18	+0.19	-2.08
C	-0.79	+1.56	-0.07	+0.67	-1.44
S	+0.60	+0.38	-0.88	-0.28	-3.39
F	-3.76	-1.73	-2.83	-3.23	-3.05

performances reported in Table 5 in which encoders are duplicated and summed together.

Table 4 also shows that, despite the simplicity of the combination strategy, it is possible to design a Dual-Encoder Transformer that performs better than its respective single-encoder counterparts. To further investigate this aspect, given the set of encoding methods E={B, L, C, S, F} representing the self-attention, LSTM, Static Expansion and FNet respectively, we measure the synergy $s_{ij} \in \mathbb{R}$ between method i and j simply as the performance difference between the dual-encoder $i + j$ and the single-encoder made of the i method $i + i$. The Synergy matrix S is reported in Table 6.

Static Expansion and the LSTM combined well with the Transformer encoder, the latter case was expected thanks to the works of [13, 14]. On the contrary, the ConvS2S case contradicts the effectiveness observed in literature [20, 15, 17, 16] which highlights the limitations of our trivial combination strategy. The pair "Base + FNet" seems to perform the worst.

5.3. Scaling Encoders

Based on the Synergy matrix reported in Table 6 we increase the number of encoders in the Multi-Encoder Transformer and discuss their performances across all data sets. To decide which methods are used in the dual and triple encoder configurations, we select the top three values of the formula $max_{i,j\in\{1,...,5\}}\{s_{ij} + s_{ji}\}$ applied on the Synergy matrix reported in Table 6. In particular, the top three values are represented by 2.93, 2.10 and 1.38 corresponding to "Base + Static Exp$_{M=12}$", "Base + LSTM$_{M=18}$" and "LSTM$_{M=18}$ + ConvS2S$_{M=6}$". For this reason, we design the Dual-Encoder as "Base + Static Exp", the Triple-Encoder is made of "Base + Static Exp + LSTM", the Quadruple-Encoder is made of "Base + Static Exp + LSTM + ConvS2S" and the Quintuple-Encoder is made of all encoding methods. The number of layers for each additional encoder is configured according to the size of the training set, in particular, for "Gl-En" and "Sp-En" we deploy a smaller version of the Multi-Encoder Transformer.

Several observations can be made from the results reported in Table 7. *(i)*. First of all, the Dual-Encoder outperforms the baseline across all data sets of different sizes and languages, whereas increasing the number of encoders beyond two leads to mixed results. In particular, the performance increase can be appreciated most in the case of two encoders; beyond this case, the improvement is smaller if not even worse. This may explain why State-of-the-Art models in the literature typically combine two deep learning methods at most. *(ii)*. The Multi-Encoder Transformer achieved a maximum increase of 5.35 and 7.16 BLEU in the case of "Gl-En" and "Sp-En" respectively, suggesting that richer encoder representations can compensate for the lack of data. Therefore, adopting multiple heterogeneous encoders can be a suitable strategy for low-resource languages. *(iii)*. In the case of bigger datasets, the Multi-Encoder Transformer can be beneficial even though there is no clear relationship between data size and the increase in performance. For example, "En-It" and "Sk-En" tasks benefit more from the Dual-Encoder configuration compared to "En-Vi" despite having a greater and a smaller number of training samples respectively. *(iv)*. The case of the Quintuple-Encoder appears to be the breaking point of the benefits of our approach, however, this may be caused by the Fourier Transform method not combining well with others rather than a limitation of the scaling method.

In Table 8 we compare the best results against MarianMT [39], a Transformer model pre-trained on 1M samples from OPUS-100 [40] and fine-tuned on our datasets for 50 epochs. Our model performs reasonably worse in low-resource setups, due to the lack of data, but comparably or better otherwise. This suggests that MarianMT architecture can be improved with our approach to achieve even better results. Overall, increasing the number of deep learning methods in the encoder can be beneficial but results may depend on the application and data size. Deep learning practitioners may consider a trade-off between performance and computational cost in case they decide to follow this strategy.
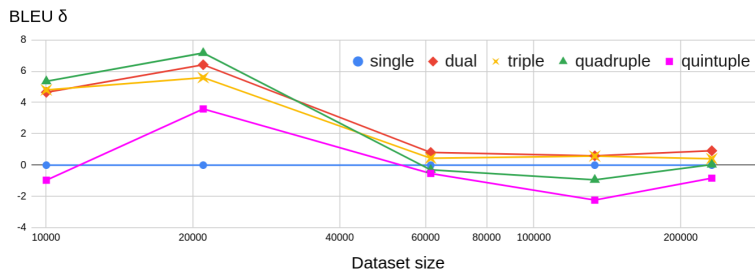
Table 7

BLEU scores of the Multi-Encoder Transformer across all translation tasks. N represents the number of Transformer layers. M, Q, U and V represent the number of additional Static Expansion layers, LSTM layers, ConvS2S layers and Fourier layers in the Dual-Encoder, Triple Encoder, Quadruple and Quintuple Encoder respectively. δ denotes the difference with respect to the baseline. The best scores for each translation task are underlined.

Task	N	Single	M	Dual (δ)	Q	Triple (δ)	U	Quadruple (δ)	V	Quintuple (δ)
Gl-En	3	13.03	6	17.68 ($+4.65$)	6	17.83 ($+4.80$)	6	18.38 ($+5.35$)	6	12.07 (-0.96)
Sp-En	3	27.22	6	33.63 ($+6.41$)	6	32.81 ($+5.59$)	6	34.38 ($+7.16$)	6	30.81 ($+3.59$)
Sk-En	6	27.15	12	27.96 ($+0.81$)	18	27.59 ($+0.44$)	6	26.85 (-0.3)	6	26.62 (-0.53)
En-Vi	6	31.13	12	31.73 ($+0.60$)	18	31.71 ($+0.58$)	6	30.19 (-0.94)	6	28.90 (-2.23)
En-It	6	32.86	12	33.78 ($+0.92$)	18	33.27 ($+0.41$)	6	32.90 ($+0.04$)	6	32.03 (-0.83)

B=Self-Attention, S=Static Expansion, L=LSTM, C=ConvS2S, F=Fourier
Single=B, Dual=B+S, Triple=B+S+L, Quadruple=B+S+L+C, Quintuple=B+S+L+C+F

Figure 2: Multi-Encoder Transformer performances.

Table 8

Performance Multi-Encoder Transformer best BLEU scores compared to the fine-tuned MarianMT.

Model \\ Task	Gl-En	Sp-En	Sk-En	En-Vi	En-It
Ours	18.38	34.38	27.96	31.73	33.78
MarianMT	28.29	42.24	31.01	31.16	34.54

5.4. Limitations

Although we removed, wherever possible, the feed-forward layer, the additional encoders inevitably introduce a significant additional computational cost and memory footprint as can be seen from Table 9. As a result, the Multi-Encoder Transformer is increasingly less efficient than the single-encoder baseline depending on the number of additional encoders. Fortunately, benefits can be observed early in the dual-encoder configuration and improvements can be appreciated by using less than double the amount of training time spent for the baseline.

The performances reported in this work do not necessarily represent the full capabilities of each model. We focused only on one configuration of the entire hyperparameter space. In particular, we adopted the learning rate scheduler, optimizer, and initialization method of [2]

Table 9

Number of parameters (denoted with θ) and the number GFLOPS (denoted as ω) required by the Multi-Encoder Transformer for the forward step of a sequence of 128 tokens in both encoder and decoder.

Model Task	Single		Dual		Triple		Quadruple		Quintuple	
	ω	θ	ω	θ	ω	θ	ω	θ	ω	θ
Gl-En, Sp-En	5.9	26M	8.3	31M	11.6	46M	12.8	51M	16.0	64M
Sk-En, En-Vi, En-It	11.8	48M	16.7	61M	26.4	99M	27.6	109M	30.8	135M

and the same batch size is used for most configurations. While such choices are ideal for the Transformer, they may not be suitable for all the architectures introduced in this work. The exploration of a variety of configurations is often recommended in order to fully appreciate new models. Such fine-tuning practice is beyond the scope of the work.

6. Conclusions

In this work, we conducted an empirical study of the effectiveness of implementing an encoder with an increasing number of heterogeneous Deep learning methods. We considered five Deep Learning methods: self-attention, LSTM, convolution, static expansion, and Fourier transform. We discovered that the Multi-Encoder Transformer can be an effective architectural design for two encoders. However, by increasing the number of encoders beyond two, the impact is limited and can be both positive, such as in the case of low-resource languages, and harmful in other instances.

Despite combining different encoders with a simple sum, the Multi-Encoder Transformer, particularly the Dual-Encoder configuration, outperformed the baseline across all benchmarks. However, connecting more than two encoders resulted in mixed impact. We suspect this might be due to the extreme simplicity of our combination method. Future works can investigate better combining each encoder representation or even formulating an interconnection module specifically designed to leverage the diversity of different Deep Learning methods.

References

[1] E. Bassignana, D. Brunato, M. Polignano, A. Ramponi, Preface to the Seventh Workshop on Natural Language for Artificial Intelligence (NL4AI), in: Proceedings of the Seventh Workshop on Natural Language for Artificial Intelligence (NL4AI 2023) co-located with 22th International Conference of the Italian Association for Artificial Intelligence (AI* IA 2023), 2023.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[3] N. Kalchbrenner, P. Blunsom, Recurrent continuous translation models, in: Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1700–1709.

[4] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Advances in neural information processing systems, 2015, pp. 649–657.

[5] J. Bradbury, S. Merity, C. Xiong, R. Socher, Quasi-recurrent neural networks, arXiv preprint arXiv:1611.01576 (2016).

[6] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y. N. Dauphin, Convolutional sequence to sequence learning, in: International Conference on Machine Learning, PMLR, 2017, pp. 1243–1252.

[7] J. Lee-Thorp, J. Ainslie, I. Eckstein, S. Ontanon, Fnet: Mixing tokens with fourier transforms, arXiv preprint arXiv:2105.03824 (2021).

[8] J. C. Hu, R. Cavicchioli, A. Capotondi, Exploring the sequence length bottleneck in the transformer for image captioning, arXiv preprint arXiv:2207.03327 (2022).

[9] J. C. Hu, R. Cavicchioli, A. Capotondi, Expansionnet v2: Block static expansion in fast end to end training for image captioning, arXiv preprint arXiv:2208.06551 (2022).

[10] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).

[11] T. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1412–1421. URL: https://aclanthology.org/D15-1166. doi:10.18653/v1/D15-1166.

[12] L. Huang, W. Wang, J. Chen, X.-Y. Wei, Attention on attention for image captioning, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 4634–4643.

[13] M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, N. Parmar, M. Schuster, Z. Chen, et al., The best of both worlds: Combining recent advances in neural machine translation, arXiv preprint arXiv:1804.09849 (2018).

[14] J. Hao, X. Wang, B. Yang, L. Wang, J. Zhang, Z. Tu, Modeling recurrence for transformer, arXiv preprint arXiv:1904.03092 (2019).

[15] B. Yang, L. Wang, D. Wong, L. S. Chao, Z. Tu, Convolutional self-attention networks, arXiv preprint arXiv:1904.03107 (2019).

[16] S. Kim, A. Gholami, A. Shaw, N. Lee, K. Mangalam, J. Malik, M. W. Mahoney, K. Keutzer, Squeezeformer: An efficient transformer for automatic speech recognition, arXiv preprint arXiv:2206.00888 (2022).

[17] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al., Conformer: Convolution-augmented transformer for speech recognition, arXiv preprint arXiv:2005.08100 (2020).

[18] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google's neural machine translation system: Bridging the gap between human and machine translation, arXiv preprint arXiv:1609.08144 (2016).

[19] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[20] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, M. Auli, Pay less attention with lightweight and dynamic convolutions, arXiv preprint arXiv:1901.10430 (2019).

[21] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, Q. V. Le, Qanet: Combining local convolution with global self-attention for reading comprehension, arXiv preprint

arXiv:1804.09541 (2018).

[22] Z. Dai, H. Liu, Q. V. Le, M. Tan, Coatnet: Marrying convolution and attention for all data sizes, Advances in Neural Information Processing Systems 34 (2021) 3965–3977.

[23] J. Shin, J.-H. Lee, Multi-encoder transformer network for automatic post-editing, in: Proceedings of the Third Conference on Machine Translation: Shared Task Papers, 2018, pp. 840–845.

[24] S. Humeau, K. Shuster, M.-A. Lachaux, J. Weston, Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring, arXiv preprint arXiv:1905.01969 (2019).

[25] J. Libovickỳ, J. Helcl, D. Mareček, Input combination strategies for multi-source transformer decoder, arXiv preprint arXiv:1811.04716 (2018).

[26] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, F. Wei, Deepnet: Scaling transformers to 1,000 layers, arXiv preprint arXiv:2203.00555 (2022).

[27] S. Jaszczur, A. Chowdhery, A. Mohiuddin, L. Kaiser, W. Gajewski, H. Michalewski, J. Kanerva, Sparse is enough in scaling transformers, Advances in Neural Information Processing Systems 34 (2021) 9895–9907.

[28] X. Liu, K. Duh, L. Liu, J. Gao, Very deep transformers for neural machine translation, arXiv preprint arXiv:2008.07772 (2020).

[29] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, A. Waibel, Very deep self-attention networks for end-to-end speech recognition, arXiv preprint arXiv:1904.13377 (2019).

[30] I. Beltagy, M. E. Peters, A. Cohan, Longformer: The long-document transformer, arXiv preprint arXiv:2004.05150 (2020).

[31] Z. Liu, S. Luo, W. Li, J. Lu, Y. Wu, S. Sun, C. Li, L. Yang, Convtransformer: A convolutional transformer network for video frame synthesis, arXiv preprint arXiv:2011.10185 (2020).

[32] A. Magueresse, V. Carles, E. Heetderks, Low-resource languages: A review of past work and future challenges, arXiv preprint arXiv:2006.07264 (2020).

[33] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, arXiv preprint arXiv:1508.07909 (2015).

[34] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[35] M. Post, A call for clarity in reporting bleu scores, arXiv preprint arXiv:1804.08771 (2018).

[36] T. Q. Nguyen, J. Salazar, Transformers without tears: Improving the normalization of self-attention, arXiv preprint arXiv:1910.05895 (2019).

[37] M. Popel, O. Bojar, Training tips for the transformer model, The Prague Bulletin of Mathematical Linguistics 110 (2018). doi:10.2478/pralin-2018-0002.

[38] I. Provilkov, D. Emelianenko, E. Voita, Bpe-dropout: Simple and effective subword regularization, arXiv preprint arXiv:1910.13267 (2019).

[39] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. F. Aji, N. Bogoychev, et al., Marian: Fast neural machine translation in c++, arXiv preprint arXiv:1804.00344 (2018).

[40] J. Tiedemann, Opus-parallel corpora for everyone., Baltic Journal of Modern Computing 4 (2016).