Informer, an Information Organization Transformer Architecture

Cristian David Estupiñán Ojeda, Cayetano Nicolás Guerra Artal
and Francisco Mario Hernández Tejera
University Institute SIANI, University of Las Palmas de Gran Canaria, 35017, Las Palmas de Gran Canaria, Spain

Keywords: Deep Learning, Linear Transformer, Informer, Convolution, Self Attention, Organization, Neural Machine Translation.

Abstract: The use of architectures based on transformers presents a state of the art revolution in natural language processing (NLP). The employment of these architectures with high computational costs has increased in the last few months, despite the existing use of parallelization techniques. This is due to the high performance that is obtained by increasing the size of the learnable parameters for these kinds of architectures, while maintaining the models' predictability. This relates to the fact that it is difficult to do research with limited computational resources. A restrictive element is the memory usage, which seriously affects the replication of experiments. We are presenting a new architecture called Informer, which seeks to exploit the concept of information organization. For the sake of evaluation, we use a neural machine translation (NMT) dataset, the English-Vietnamese IWSLT15 dataset (Luong and Manning, 2015). In this paper, we also compare this proposal with architectures that reduce the computational cost to $O(n \cdot r)$, such as Linformer (Wang et al., 2020). In addition, we have managed to improve the SOTA of the BLEU score from 33.27 to 35.11.

1 INTRODUCTION

Within the field of natural language processing (NLP), there are a lot of different types of applications, such as neural machine translation (NMT), as well as document classification, language modelling, sentiment analysis, question-answering, and others. There is a great diversity of architectures to face these challenges. In recent months, those based on transformers have gained great relevance. For example we have the Large-BERT network (Devlin et al., 2019), which is mainly focused on the task of language modeling and can also be considered a special type of word embedding. The development of Large-BERT resulted in GPT-3 (Brown et al., 2020) which has been a revolution by using a high amount of trainable parameters (around 175 Billion). Likewise, there is another type of network exemplified by G-Shard (Lepikhin et al., 2020). This network addresses the task of neural machine translation, which has a total of 600 Billion parameters.

It should be noted that these kinds of neural networks are practically impossible to replicate if there is not a great amount of computational resources available, due to the high computational cost they present. Even when techniques to improve their efficiency are used, there are still memory and training costs restrictions. The use of techniques such as transfer learning has become very popular for NLP tasks by taking advantage of this kind of architecture (Devlin et al., 2019), (Pennington et al., 2014). However, it is not always possible to make use of this training technique, as it depends highly on the source and target domains.

That is why new architectures have emerged, trying to deal with these root causes, and reduce the complexity to a linear order during training. The Linformer (Wang et al., 2020) is a technique that addresses this problem by reducing the complexity of the self-attention layer. This is the layer with the highest complexity and computational cost, becoming $O(n^2)$ in the original transformer (Vaswani et al., 2017). The authors of the Linformer architecture have demonstrated, both theoretically and empirically, that self-attention can be reduced to a low-rank matrix. This is achieved through the decomposition of dot-product attentions into multiple attentions of linear projections. This approach allows the factorization of the original attention.

In this work, we will focus on the application of efficient architectures in the field of neural machine translation task (Ott et al., 2018). For experimental evaluation purposes, the English-Vietnamese dataset will be used (Luong and Manning, 2015). This is because it works reasonably well with low-resources machines, and this dataset is ideal for evaluation using graphic cards with limited video memory. The main idea is to evaluate in a comparative fashion the performance of four different transformer architectures: firstly the original transformer, proposed in Attention is All You Need (Vaswani et al., 2017), a Linformer model with linear projections as self-attention (applied to the encoder stage), and other model proposed by us corresponding to the use of convolutional layers and max-pooling in self-attention. Finally, we also propose and evaluate a new model, the Informer model, which uses these techniques, as well as an information organization module replacing the original Feed-Forward.

This information organization module is based on organizing the information coming from the self-attention output. It allows the network to organize the features by reducing their dimension with convolutional layers at the encoder level. That means that we do not add the restriction that the input must be the same as the output of the module, such as an auto-encoder. The network has full freedom to organize the information. We demonstrate that The Informer presents the best performance, with a BLEU-1 metric value of 35.1119 in the inference test set and a PERPLEXITY value of 20.2263 in the inference validation set.

2 RELATED WORK

There are different architectures in the state of the art which try to improve the efficiency of transformer networks (Tay et al., 2020c), (See Table 1). Among them, the most prominent is the Linformer (Wang et al., 2020), which achieves a linear cost during training in self-attention. It is positioned as one of the most attractive alternatives when training a low-cost model. Thanks to this potential, it will be used as the basis for conducting the experiments in this paper.

Another model is the Reformer (Kitaev et al., 2020). This facilitates handling datasets with quite long sequences-sizes greater than 2048. The main advantage of the Reformer is that it uses less memory. The complexity is reduced to $O(n \log n)$ by using the Locally Sensitive Hashing technique (LSH).

One technique that has been used by default in Linformer is mixed precision (Micikevicius et al., 2017). It is based on using floating-point types, of 16bits and 32bits respectively, in such a way that the memory used when training is reduced. However, due to compatibility terms when training the models, it was decided not to implement this technique in the experiments. Another interesting technique is the Sparse Transformer (Child et al., 2019). It consists of the addition of sparsity in self-attention, performing the calculations from the diagonal of the generated P matrix. It is useful in predicting longer sequences.

Finally, one of the recent results with the IWSLT15 English-Vietnamese dataset achieves a BLEU-1 score of 33.27 (Provilkov et al., 2020). It is based on subword regularization and it uses Byte-Pair Encoding (BPE) (Sennrich et al., 2016) along with dropout during training and standard BPE during inference. (See Table 2) for paper results with the English-Vietnamese dataset. There is another article that obtains a BLEU score of 43.60 (Phan-Vu et al., 2018), however, this article does not only use the IWSLT15 EN-VI dataset to perform the training. It also uses a set of subtitles extracted from the internet, increasing the dataset from 133,317 samples to 1,103,456 samples. That is why we cannot make a fair comparison with this article.

3 DATASET

The dataset that will be used is the IWSLT15 English-Vietnamese dataset (Luong and Manning, 2015) from the International Workshop on Spoken Language Translation. This dataset has the following main characteristics:

- Training samples: 133,317 (Files: X = train.en, Y = train.vi)
- Validation samples: 1,553 (Files: X = tst2012.en, Y = tst2012.vi)
- Test samples: 1,268 (Files: X = tst2013.en, Y = tst2013.vi)

In order to deal with low video memory, we decided to remove sequences higher than 200 tokens.

- Samples eliminated from training set: 40. (0.03% of the data)
- Samples eliminated from validation set: 0
- Samples eliminated from test set: 0

The number of total samples of the dataset after this operation is:

- Training samples: 133,277
- Validation samples: 1,553
- Test samples: 1,268

The average sequence size of the samples is: 17.79, and the maximum sequence sizes are (English: 193, Vietnamese: 199).

Table 1: Extracted from (Tay et al., 2020c). "Summary of Efficient Transformer Models presented in chronological order of their first public disclosure. Some papers presented sequentially may first appear at the same time, e.g., as an ICLR submission. Papers annotated with a superscript † are peer-reviewed papers. Class abbreviations include: FP = Fixed Patterns or Combinations of Fixed Patterns, M = Memory, LP = Learnable Pattern, LR = Low Rank, KR = Kernel and RC = Recurrence. Furthermore, n generally refers to the sequence length and b is the local window (or block) size. We use subscript g on n to denote global memory length and n_c to denote convolutionally compressed sequence lengths".

Model / Paper	Complexity	Decode	class
Memory Compressed† (Liu et al., 2018)	$O(n_c^2)$	yes	FP+M
Image Transformer† (Parmar et al., 2018)	$O(n \cdot m)$	yes	FP
Set Transformer† (Lee et al., 2019)	$O(nk)$	no	M
Transformer-XL† (Dai et al., 2019)	$O(n^2)$	yes	RC
Sparse Transformer (Child et al., 2019)	$O(n\sqrt{n})$	yes	FP
Reformer† (Kitaev et al., 2020)	$O(n \log n)$	yes	LP
Routing Transformer (Roy et al., 2020)	$O(n \log n)$	yes	LP
Axial Transformer (Ho et al., 2019)	$O(n\sqrt{n})$	yes	FP
Compressive Transformer† (Rae et al., 2020)	$O(n^2)$	yes	RC
Sinkhorn Transformer† (Tay et al., 2020b)	$O(b^2)$	yes	LP
Longformer (Beltagy et al., 2020)	$O(n(k+m))$	yes	FP+M
ETC (Ainslie et al., 2020)	$O(n_g^2 + nn_g)$	no	FP+M
Synthesizer (Tay et al., 2020a)	$O(n^2)$	yes	LR+LP
Performer (Choromanski et al., 2020)	$O(n)$	yes	KR
Linformer (Wang et al., 2020)	$O(n)$	no	LR
Linear Transformers† (Katharopoulos et al., 2020)	$O(n)$	yes	KR
Big Bird (Zaheer et al., 2020)	$O(n)$	no	FP+M

Table 2: English-Vietnamese results. Metric used: BLEU.

Model	BLEU Score
Transformer+BPE-Dropout (Provilkov et al., 2020)	33.27
Transformer+BPE+FixNorm+ScaleNorm (Nguyen and Salazar, 2019)	32.8
Transformer+LayerNorm-simple (Xu et al., 2019)	31.4
CVT (Clark et al., 2018)	29.6
Self-Adaptive Control of Temperature (Lin et al., 2018a)	29.12
SAWR (Zhang et al., 2019)	29.09
DeconvDec (Lin et al., 2018b)	28.47
LSTM+Attention+Ensemble (Luong and Manning, 2015)	26.4

The core idea when using this dataset is to evaluate whether it is possible to have a performance equal to, or greater than, the original transformer with a reduced sequence size "r". This size will be less than the average sequence size of the samples. To do this, we use a reduction size that is approximately half of that value, $r = 8$. Models that use max-pooling techniques, will reduce the size to $r = 4$.

4 MODEL COMPARISON

For this work, we will compare four different transformer models. We will evaluate the performance of the architectures as techniques that are applied to improve their efficiency. To this end, we have a first model based on the original transformer (Vaswani et al., 2017), in such a way that it serves as a basis for determining the performance of the rest of the models. The second model implements the linear projections proposed in the Linformer paper (Wang et al., 2020). The third model applies convolutional techniques when making the projections. The latest model also includes an information organization stage that replaces the conventional feed-forward submodule. It is important to note that these models focus on applying efficiency improvement techniques only in the encoder stage. This is because we want to maintain the flexibility provided by the decoder stage. Depending on the type of data that is being used, it could also be applied to the decoder stage.

4.1 Original Transformer

For this first baseline model, we replicate the original transformer model as presented in (Vaswani et al., 2017) (See Figure 1). As well as the default self-attention layer. (See Figure 2). The original transformer is made up of various clearly differentiated modules. In a first module we can find the input to the encoder and decoder modules respectively. This module consists of the sum of an input embedding (one embedding is used for encoder and the other embedding for decoder) with the positional encoding matrix.

Then we have the encoder modules, which can be stacked. These are made up of a multi-head attention submodule, as well as a feed-forward submodule. These submodules are further joined with a residual connection.

Regarding to multi-head attention, (Figure 2), we need to compute the Query, Key and Value matrices. To do this, the entries are multiplied by the respective Query, Key and Value weight matrices. Later, we calculate the Scores matrix through the matrix multiplication between the Query matrix and the transpose of the Key matrix. We divide this Scores matrix by the root of the head dimension, thereby stabilizing the gradient. Likewise, the values will be normalized through a softmax function. Finally, the results are computed through matrix multiplication between the Scores matrix and the Value matrix. (See equation 1).

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

Each layer of the decoder follows a similar scheme. This module presents the inclusion of a masked multi-head attention submodule. In it, we masked the input of the decoder; in this way we managed to prevent the decoder from seeing tokens of the sentence that did not correspond to it. With this, it is possible to simulate the state of inference.

4.2 Linformer Encoder

In the Linformer encoder (Wang et al., 2020), we keep the overall structure provided by the original transformer model. However, the self-attention stage presents the technique of using linear projections in order to reduce the model complexity. (See Figure 3). The authors demonstrate that the self-attention mechanism, in the context of the mapping matrix P, is low-rank.

When making the projections, we decided to share the projections at all levels by default. This implies that the same projection will be used for Keys and

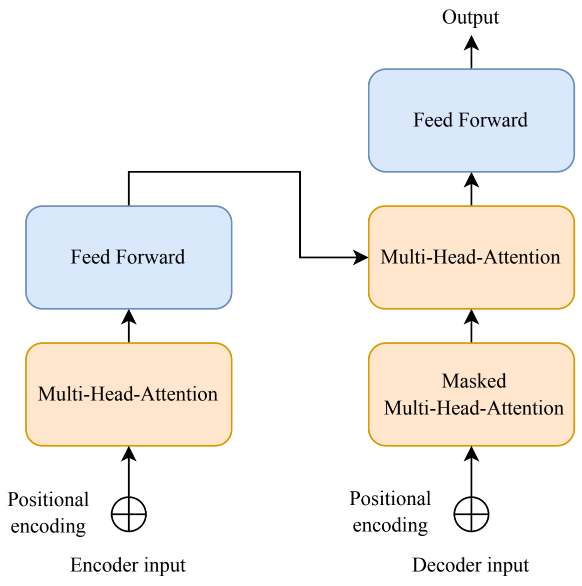

Figure 1: Original Transformer architecture.

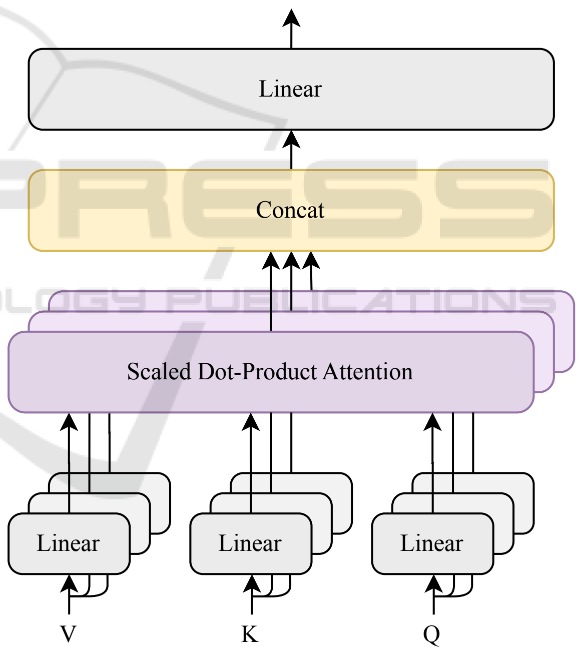

Figure 2: Original Transformer Multi-Head Attention.

Values. Likewise, all the self-attention heads will use the same projection, as well as the N encoder layers. Thus, we managed to reduce the number of trainable parameters to carry out the projections. However, this is a parameter that could be adjusted depending on the type of data and the performance of the model. In (Figure 4), the inner workings of the self-attention layer are shown, we emphasize the shapes of tensors for a better understanding.

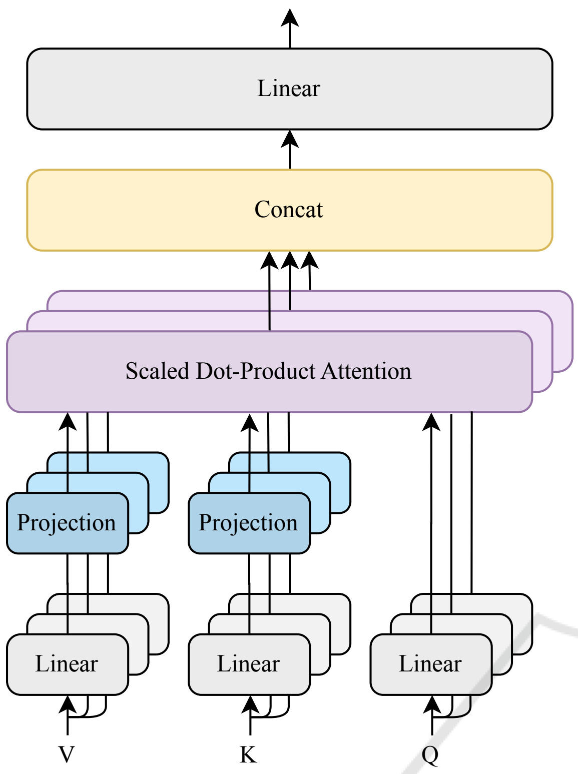

Figure 3: Multi-Head Attention operation on the Linformer.

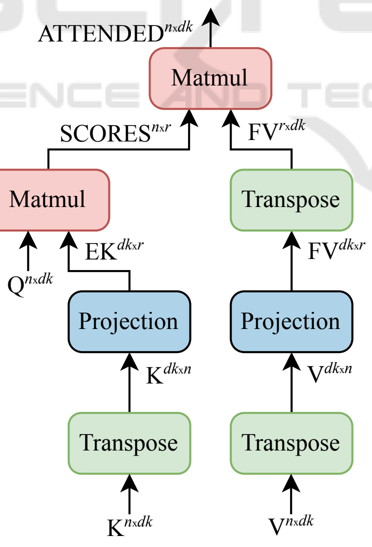

Figure 4: Internal operations of self-attention in the Linformer model with linear projections.

4.3 Convolutional Linformer Encoder

In this model a convolutional layer is applied, in addition to max-pooling, when generating the projections of the Keys and Values matrices. This presents a certain difference when it comes to computing, with respect to the previous model, so it is necessary to apply the projections before splitting heads. (See Figure 5).

Given this, it is possible to apply the convolutional layer on the projections. The kernel and stride size suggested in the Linformer paper (Wang et al., 2020) is followed for this model. The rest of the architecture is exactly the same as the original Linformer. Thanks to this, it is possible to evaluate the differences between linear-type projections and convolutional-type projections. By default, max-pooling is used by halving the default value of ”r”, from $r = 8$ to $r = 4$. This means compression to half the encoder sequence size indicated on the Linformer encoder model. We also apply a Leaky-ReLU activation layer.

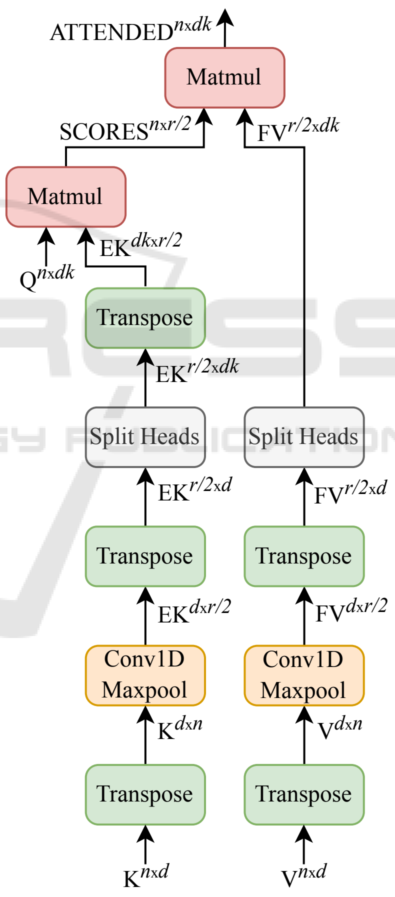

Figure 5: Inner operations of the Convolutional Linformer.

4.4 Informer Encoder

The proposed model in this document goes one step further in the field of optimization. The main idea in this model is the use of an information organization module that allows the network to organize the features produced by the self-attention layer. The hypothesis is that the Feed-Forward layer is oversized and makes training convergence slower. Incorporating this organization to an encoder level also facilitates a drastic reduction of the number of parameters. In that way, we achieve a more efficient architecture. This submodule differs from a one-dimensional convolutional auto-encoder because the input and output must not be the same. We allow the submodule full freedom to organize the features in each encoder module and make its own representations.

We start from the previous model that uses convolutional operations at the self-attention level. In fact, the same parameters and layers will be used to make the comparison between both models. Furthermore, this model incorporates the use of a convolutional auto-encoder submodule. This submodule aims to replace the feed-forward from the encoder layers, in such a way that a considerable decrease in the number of parameters is achieved.

The feed-forward submodule consists of two linear layers: the first linear layer that goes from a model dimension size (512) to a higher size, (2048) along with the activation function (Leaky-Relu), followed by a second layer that transforms the features back to the model dimension. We take into account that in Linformer-based models, it is necessary to maintain a fixed encoder sequence size, therefore, we can take advantage of that handicap.

To do this, we designed a structure similar to an auto-encoder instead of following the feed-forward scheme. First, a one-dimensional convolutional layer with a kernel and stride sizes of d_model/h is applied, followed by a Leaky-ReLU activation layer. This reduces the size of the model dimension to the size of a head dimension. Then max-pooling is applied to halve this size. Therefore, the latent dimension of the auto-encoder will be $dk/2$. We finish this process by building a mirror decoder, with an unpooling layer, as well as a convolutional transpose with the same kernel and stride sizes. (See Figure 6).

As we are based on the Linformer article, whose complexity is $O(n \cdot r)$, the complexity of the Informer will be the same. The main change lies in the replacement of the feed-forward sub-module by operations of linear complexity (The Information Organization sub-module).

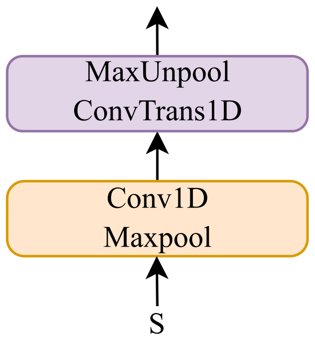

Figure 6: Informer submodule. S corresponds to the encoder submodule 1 output.

5 EXPERIMENTS AND RESULTS

5.1 Hyper Parameters

To make a correct comparison between the different models, we have chosen to set the same values for Label Smoothing, Dropout, Batch Size, Epochs, Heads, Model Dimension, Head Dimension, and encoder/decoder layers. Except for the original transformer, the rest of the encoder sentence lengths and reduced sequence dimensions will also be the same. We also apply max-pooling for the convolutional Linformer and Informer models. (See Table 3).

5.2 Training Details

In order to obtain the vocabularies, one vocabulary is used for inputs (English), and another vocabulary for outputs (Vietnamese). To do this, we start from the X and Y training sets. A list of all the unique words is obtained, and a numerical identifier is assigned for each word/token in the list. Then we have two dictionaries for each X and Y set (One to convert from token/word to integer, and another one for the opposite operation).

On the other hand, when making the inference, there are techniques that try to solve the problem of bias exposure, as is the case of beam search (Wiseman and Rush, 2016). However, after comparing different architectures, we chose not to use this technique, in order to avoid incorporating parameters like beam size or penalty length that may affect the comparison results. Instead, greedy search has been chosen. The metrics that will be used are BLEU-1 (Papineni et al., 2002) and perplexity. Finally, a limit of 40 epochs has been established in order to compare the models.

Table 3: Hyper-Parameters used in the models comparison.

Hyper-parameter	Transformer	Linformer	Conv. Linformer	Informer
Heads h	8	8	8	8
Model Dimension d_model	512	512	512	512
Head Dimension dk	64	64	64	64
Encoder/Decoder layers N	6	6	6	6
Encoder fixed length n	Not applied	200	200	200
Max-pooling Size	Not applied	Not applied	2	2
Feed-Forward Size ff	2048	2048	2048	Not applied
Reduced Sequence Dimension r	Not applied	8	8	8
Label Smoothing	True	True	True	True
Dropout	0.1	0.1	0.1	0.1
Batch Size	8	8	8	8
Epochs	40	40	40	40

5.3 Results Analysis

If we show the results obtained during training, it is clearly observed that the Informer model outperforms the rest of the models. (See Figures 7 and 8). Also, as we make changes to the architectures, we see different performance gains. (See Tables 4 and 5).

Table 4: Inference BLEU-1 results.

Model	Validation	Test
Original Transformer	26.2469	26.0866
Linformer	27.5829	27.5111
Conv. Linformer	28.3883	29.0317
Informer	31.7617	31.9736

Table 5: Perplexity results.

Model	Validation
Original Transformer	26.0275
Linformer	26.2831
Conv. Linformer	25.5365
Informer	23.2831

Then, we can confirm that the use of an information organization submodule helps to obtain a better performance in neural machine translation problems. If we compare the number of parameters between the Linformer convolutional encoder model and the Informer model, we can point out that the Informer has around 9 million parameters less than the Linformer convolutional encoder, which also makes it a more efficient model. (See Table 6).

Table 6: Total model parameters comparison.

Model	Total Parameters
Conv. Linformer	135,368,888
Informer	126,613,016

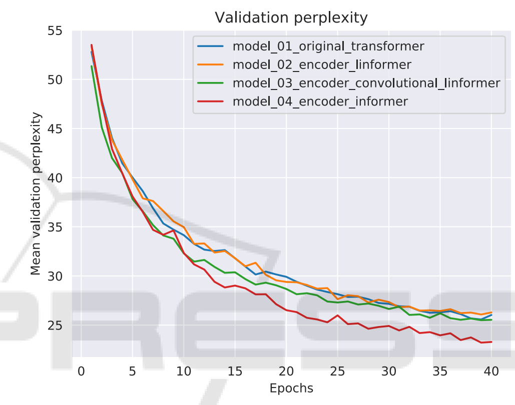

Figure 7: Validation Perplexity comparison.

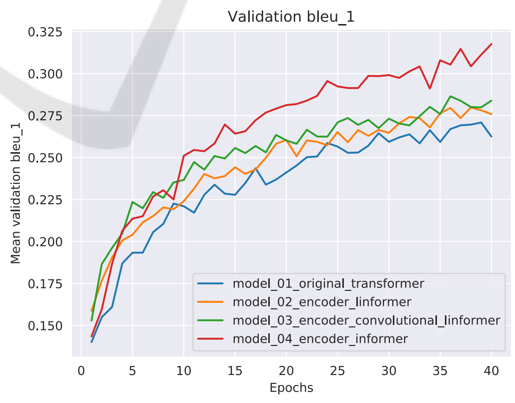

Figure 8: Validation Inference BLEU-1 comparison.

We can see that if we allow more training time (from 40 epochs to 120 epochs), the SOTA of this dataset is obtained, 35.1119 BLEU-1. Both the perplexity and the BLEU improve as the epochs increase.

We can also observe that the performance had stabilized before the one hundredth epoch. It is from this epoch that we can see a slight improvement in the behavior of the network. We can indicate that there is no overfitting, either. Complete scores in (Table 7). We present the final results in (Figures 9, 10).

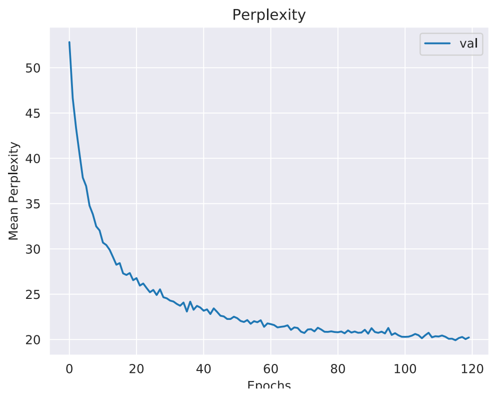

Figure 9: Validation Perplexity.

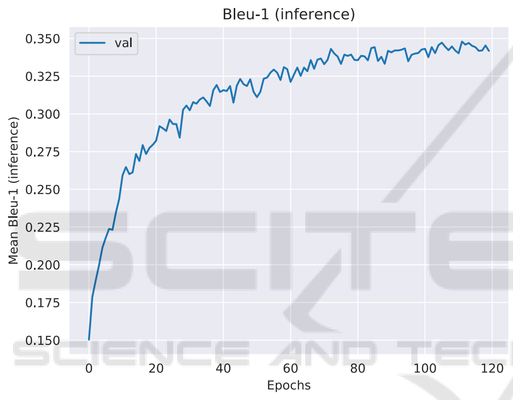

Figure 10: Validation Inference BLEU-1.

Table 7: Encoder Informer long execution results.

Encoder Informer	Value
Validation Perplexity	20.2263
Validation Inference BLEU-1	34.1775
Test Inference BLEU-1	35.1119

6 FUTURE WORK

Models based on techniques such as the Linformer and Informer provide the basis for other areas of natural language processing. This is the case of the Question Answering task, where alternatives that use this type of techniques can be proposed when generating the representations of contexts and questions. We will take advantage of the Linformer or Informer when it comes to obtaining context interpretations of higher sequence sizes.

7 CONCLUSIONS

In this work we propose an architecture, Informer, with a computational cost $O(n \cdot r)$, and compare it with different models in the literature that have the objective of reducing the complexity of the critical parts of the transformer. Specifically, it has been applied to the neural machine translation task with the English-Vietnamese dataset. This lays the foundation for the use of more efficient transformer models that can be computed by low resource workstations, as well as facilitating a better reproducibility of experiments.

The Informer new architecture introduces a new concept in transformers (Information Organization) that allows the network to reorganize the features obtained by the self-attention layer and reduce the learnable parameters. We have managed to improve SOTA with the English-Vietnamese dataset, with a BLEU-1 score of 35.11. It is a sub-module that could be applied in new transformer architectures and could potentially achieve better results, while decreasing the memory usage and training costs.

ACKNOWLEDGEMENTS

We are grateful to Cayetano Guerra Artal and Mario Hernández Tejera for their active support.

REFERENCES

Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. (2020). Etc: Encoding long and structured inputs in transformers.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Belanger, D., Colwell, L., and Weller, A. (2020). Masked language modeling for proteins via linearly scalable long-context transformers.

Clark, K., Luong, M.-T., Manning, C. D., and Le, Q. V. (2018). Semi-supervised sequence modeling with cross-view training.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. (2019). Axial attention in multidimensional transformers.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are rnns: Fast autoregressive transformers with linear attention.

Kitaev, N., Łukasz Kaiser, and Levskaya, A. (2020). Reformer: The efficient transformer.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. (2020). Gshard: Scaling giant models with conditional computation and automatic sharding.

Lin, J., Sun, X., Ren, X., Li, M., and Su, Q. (2018a). Learning when to concentrate or divert attention: Self-adaptive attention temperature for neural machine translation.

Lin, J., Sun, X., Ren, X., Ma, S., Su, J., and Su, Q. (2018b). Deconvolution-based global decoding for neural machine translation.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences.

Luong, M.-T. and Manning, C. D. (2015). Stanford neural machine translation systems for spoken language domains. In *Stanford*.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. (2017). Mixed precision training.

Nguyen, T. Q. and Salazar, J. (2019). Transformers without tears: Improving the normalization of self-attention. *CoRR*.

Ott, M., Edunov, S., Grangier, D., and Auli, M. (2018). Scaling neural machine translation.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Parmar, N., Vaswani, A., Uszkoreit, J., Łukasz Kaiser, Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Phan-Vu, H.-H., Tran, V.-T., Nguyen, V.-N., Dang, H.-V., and Do, P.-T. (2018). Machine translation between vietnamese and english: an empirical study.

Provilkov, I., Emelianenko, D., and Voita, E. (2020). BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. (2020). Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. (2020). Efficient content-based sparse attention with routing transformers.

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

Tay, Y., Bahri, D., Metzler, D., Juan, D.-C., Zhao, Z., and Zheng, C. (2020a). Synthesizer: Rethinking self-attention in transformer models.

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. (2020b). Sparse sinkhorn attention.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2020c). Efficient transformers: A survey.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *ArXiv*.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity.

Wiseman, S. and Rush, A. M. (2016). Sequence-to-sequence learning as beam-search optimization.

Xu, J., Sun, X., Zhang, Z., Zhao, G., and Lin, J. (2019). Understanding and improving layer normalization.

Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020). Big bird: Transformers for longer sequences.

Zhang, M., Li, Z., Fu, G., and Zhang, M. (2019). Syntax-enhanced neural machine translation with syntax-aware word representations.

APPENDIX

Code, dataset and results are available to download through the following link: (Informer Project).