Transformers: Statistical Interpretation, Architectures and Applications

Fanfei Meng, Yuxin Wang

Northwestern University

1 Abstract

Transformers have been widely recognized as powerful tools to analyze multiple tasks due to its state-of art multi-head attention spaces, such as Natural Language Processing (NLP), Computer Vision (CV) and Speech Recognition (SR). Inspired by its abundant designs and strong functions on analyzing input data, we would like to start from the various architectures, further proceed to the investigation on its statistical mechanism and inference and then introduce its applications on dominant tasks. The underlying statistical mechanisms arouse our interests and intrigue us to investigate it in a higher level, and this surveys will focus on its mathematical foundations and then use the principles to try to analyze the reasons for its excellent performance on many recognition scenarios.

Key Words: Transformer, Natural Language Processing (NLP), Computer Vision (CV), Speech Recognition (SR), Deep Learning

2 Architectures

2.1 Vanilla Transformers

Transformer is unparalleled in its multi-head attention mechanism, which is to score one sample based on one-by-one projections. In linguistic tasks, the projections aim at analyzing the sequential relationships between tokens and then quantify the importance of specific tokens exerted by other tokens. Compared with sequence to sequence structure [1], transformers are advantageous for enabling whole samples to be computed in parallelism [2], avoiding the time-series restriction imposed by token orders.

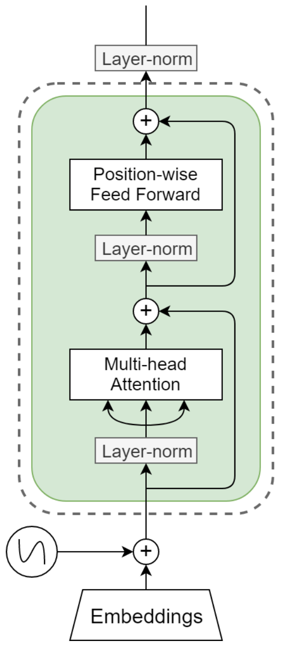


Figure 1: Transformers Configuration

Although transformers [3] are progressive in many pattern recognition aspects [4], there are also visible disadvantages due to its parallel subspace mechanism. Firstly, not each head is positive to the model training and inference even impedes the further improvement of the recognition accuracies. To be extended, Facebook AI team [5], [6], [7], , [8], , [9], [10] illustrates that some heads are less useful in model performance. Based on their one-by-one head pruning approaches, the performance of the model will be significantly impaires when more than half heads are excluded from the multi-head attention layers. In order to deploy the automatic algorithm on head pruning, they develop the layer-wise methods to specify the importance score of each head and mask the heads with lower scores. This method facilitates the investigation on dynamics of transformers and then inspires many enlightening methods on elaborating the architectures.

2.2 Dynamic Transformers

2.2.1 Head Dynamics

Dynamic mechanism of neural networks has been widely investigated in recent years and demonstrates the vigorous potential to be adaptive to environmental change. When it comes to dynamic transformers, there are three main kinds of dynamics: head dynamics, layer dynamics and attention dynamics. For head dynamics, besides previous mentioned Facebook research, other researchers try to explain the dynamics from different aspects. To be specific, [11] explains the roles of parallel heads from linguistic aspects, including semantic and syntactic etc, which comprehensively summarizes that the heads set-up should be adaptive according to the input. For the input with higher semantic analysis needs, specific heads would like to be preserved and similarly, heads endowed with higher synaptic functions are preferred to be deployed for input with this kind of needs. In the end of the literature, it applies L0 normalization to dynamically tune the heads to be pruned in specific layers and achieves better model performance.

[12] introduces head dynamics via adaptive weighted quantification, which is inspired by the channel of convolutional neural networks to tune the weights of different heads tracking deeper self-attention layers. In summary, the different heads embedded into different layers is an effective approach to improve the performance of inference, and what is more, less heads mean less computational cost in both training as well as inference.

2.2.2 Attention Span Dynamics

Inherited from the head pruning idea, some scholars propose to dynamically tune the lifespan of multi-head attention mechanisms, which not only elevate the model performance but also open a new door to attention design. In the first phase, Facebook [13] illustrates the adaptive softmax-function in measuring distances and scoring the past representations of all trained samples. For offline training, the past samples are iterated for multiple epochs and their features will be learnt from multiple training. This approach adaptively processes the past representations to grasp the more attention information that should be repetitively utilized. By Contrast, the less useful feature representations in parallel attention are abandoned.

Following the exploration, [14] deploys deep reinforcement learning to dynamically tune the attention span, replacing the previous method in traditional entropy estimation to sensitively perceive the feedback from neutral network optimization as well as updates. Given the specific layer, the attention span is adaptively prolonged or pruned to explore better configuration at specific timestamp. Similarly, the attention span is also dynamic at inference time and the controller selects the optimal attention space for mini-batch data or large batch data. In summary, the dynamics concentrates on attention has been widely investigated from multiple aspects to fulfill the potential of attention mechanism, which is manipulating the concept of projection to simulate the toke relationship and furthering the optimization on the highlighted tokens. Transformer attentions are divided into multiple attention subspace, the more excellent subspace will be spared with higher weights before head communication via fully connected layers. The head pruning method, to some extent, is to prune the less useful components and the attention span optimization is developed to adjust the sparsity or intensity of the network. Absolutely, an overcomplete neutral network is a common challenge to deep learning training, which is mostly demonstrated as most parameters are less meaningful to actual recognition or classification. The overestimated parameters are centralized in such a small bound and play extremely trivial roles in accuracy metrics, and pruning or revising network architectures are an effective approach to tackle this problem.

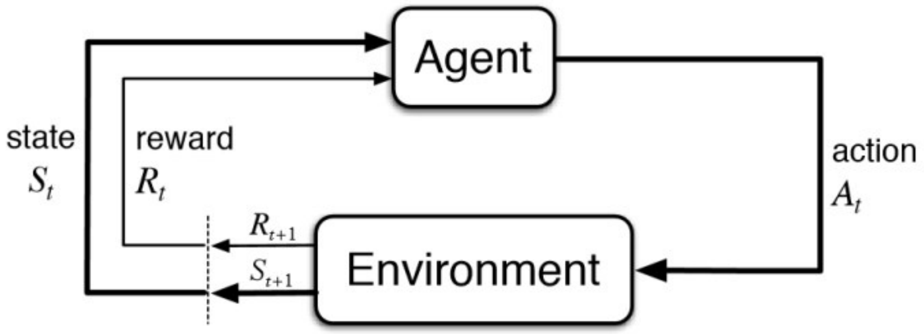


Figure 2: Reinforcement Learning

2.2.3 Deep Network Compression

On the other hand, network compressions are firstly systematically summarized by [15], which comprehensively illustrates the methodology on enabling network to be deployed and saves at relatively low memory, storage and computing costs. [16] propose the pipeline on deep network compression in time order: network pruning, weights sharing and clustering, huffman coding. Layer-wise pruning is similar as previous head pruning, in which lights are shed on the whole layer without substantial contributions to the model performance. When it comes to the cluster sharing, [9] firstly clusters the weights via K-means algorithm and the number of clusters is based on the actual compression bitwidth.

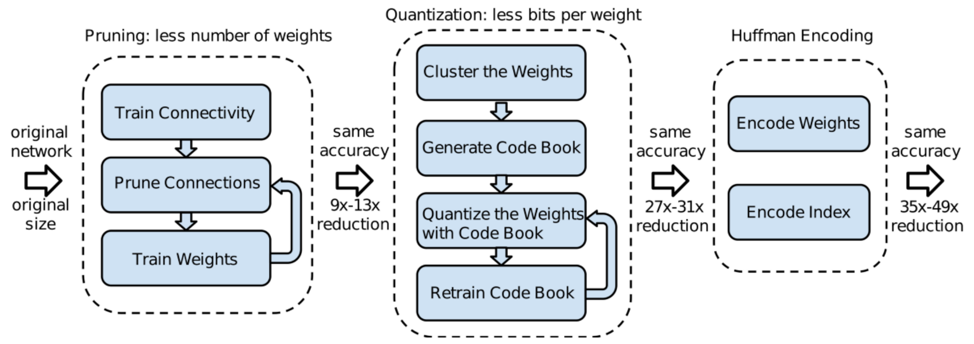


Figure 3: Pipeline of Deep Network Compression

After that, it navigates the centroid center to encode all weights in the same cluster. In this way, various weights are compressed into a single value to enable the network to be saved and deployed in a more economical configuration. Finally, huffman encoding is employed to further the compression regarding the very closed value distributions. According to the experimental results, [16] effectively compress the neutral networks up to 50% and significantly lower the needs for saving and inference.

Although the compression concept is quite helpful to us to explore on-device machine learning, the pipeline demands such a high time consuming. The three steps are facilitated in timestamp, which is unrealistic to be deployed for online training. Online machine learning is focused on receiving training samples online without knowing the following samples at each time. As a consequence, [17] propose a parallel pruning and quantization methodology, achieving better time saving with relatively low space requirements. These exploration are also enlightening and beneficial for tiny transformers all economic transformer for inclusive scenarios and tasks.

2.2.4 Layer Dynamics

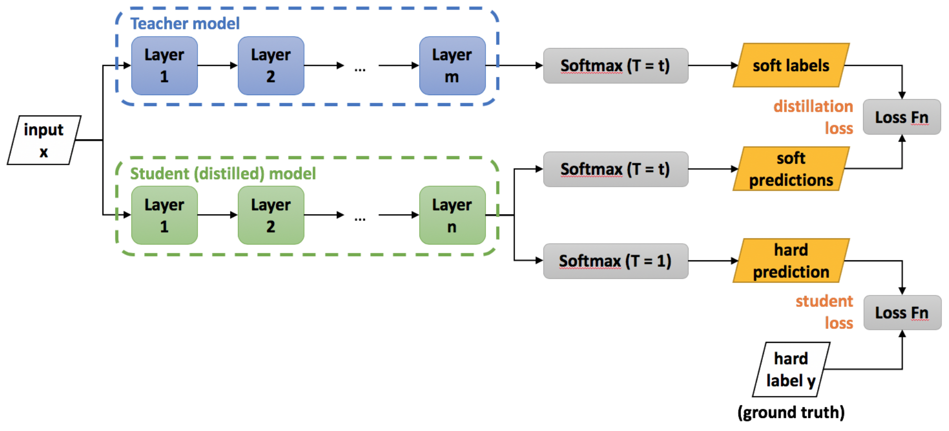


Figure 4: Pipeline of Knowledge Distillation

Head is an essential component of the self-attention layer, and the self-attention layer is a crucial part in the whole transformer framework. Some scholars shed light on the dynamic self-attention layer mechanism, which stepwisely decreases the layer use in whole training as well inference. [18] introduces a state-of-art integrated layer and width dynamics and illustrates the concrete procedures. They use the concept of knowledge distillation to train the transformers with both dynamics. To be specific, they train two separate neural networks: teacher network and student network and each two has their respective responsibilities. For the teacher network, it is prototypes as vanilla transformers with fixed layers and fixed heads to train the input sample. Based on the training circumstance of teacher work, the student network will transfer the learning experiences from teacher network to optimize its own layer width and number training selection and routines.

3 Statistical Interpretation

3.1 Statistical Mapping and Word Embedding

In dominant linguistic model focuses on applying joint probabilities to forecast contextual information and then proceed to semantic linguistic meaning. Here let me take an example: the appearance probability of word "tiger" in all Wikipedia pages is only about 0.06%, however, when it comes to the page title with the word, the probability of far larger than the former case. It absolutely indicates that the concrete linguistic scenarios play essential roles in analyzing the semantic information and then to make better decisions on classification or recognition. Based on this fact, the concept of continuous cache is proposed to enhance the language model.

To be specific, the hidden states in recurrent neutral network is manipulated for conserving the long dependence memory, quantifying the correlation between current input and past inputs in time stamp. Continuous cache [19] deploys the ordered hidden states and infers the joint probabilities of current input based on continuous cache in contextual spaces. In this way, the long-memory diminishing is mitigated to some extents and on the other hand, the joint probability with multiple previous states as well as present sates is able to take as many as possible contexts into very comprehensive consideration. Furthermore, it summarizes some experimental results based on different datasets. The result demonstrates that continuous cache mechanism is suitable for enhancing language model. The cache design is inspires by the cache design in computer system, which is centralized on absorbing as many as possible prior experiences into the present analysis and inference. However, the accuracy is directly decided by the number of long-memory information preserved in the cache and in the meanwhile, too many memories also present the further linguistic recognition.

The core objective of Natural Language Processing is to project a space and map the abstract linguistic information to embedded features. [20] comprehensively introduces a way to measure the semantic relationship of word, distinguishing the words/samples with closed semantic meanings.In actual practice, the judgement or inference on specific meaning or sentence roles vary from each other. Regarding the situation, the literature proposes some constructive methods in light of closet words. To be specific, it outlines two pairs of words: debug and debugging, streaming and stream. Although they are actually different words, in most cases, they have similar meaning and content roles in sentences or paragraphs. As a consequence, the effecting word embedding should map them as closed feature vectors to further the accurate recognition. When it comes to some concrete and important details, the cosine computation is proposed as fundamental mathematical quantification to compute the similarities of pair words.

In light of space representation, the specific tokens will extract their mathematical representation from corpus space. Following the process, the subsequent tokens are integrated into joint probability computations. After that, the correlation analysis will be performed to measure and infer the quantitative vocal analogies, combining the developed mathematical transformation to facilitate the inference and elaborations.

Following the previous literature, it is clear that better representations can lead to better recognition and awareness. However, with the rising the development in the area of Natural Language Processing, embedded features gradually become more and more complicates and the amount of features also present the exploded tendencies. In the actual training process, the redundant features won't be positive to the model performance and accuracy, and more computational resources and memory and storage requirements also exert huge pressures on the feasibility and effectiveness of model training. Concerning the fact, [21] proposes two state-of-art approaches to comparing different corpus. For the first type, it is centralized on measuring the correlations of tokens with mathematical feature spaces via log-function to approximate it into a co-occurrence probability. Meanwhile, the another methodology is focused on cosine value to simulate the similarity, which is the special instance-based with combining with another contextual vector to facilitate the measuring process. In summary, this literature is very meaningful to compare two different and discuss their advantages and disadvantages in statistical layer, and what is more important, it opens a door for us to tackle the complexity of words under specific scenarios and formalize the specific adaptive strategies.

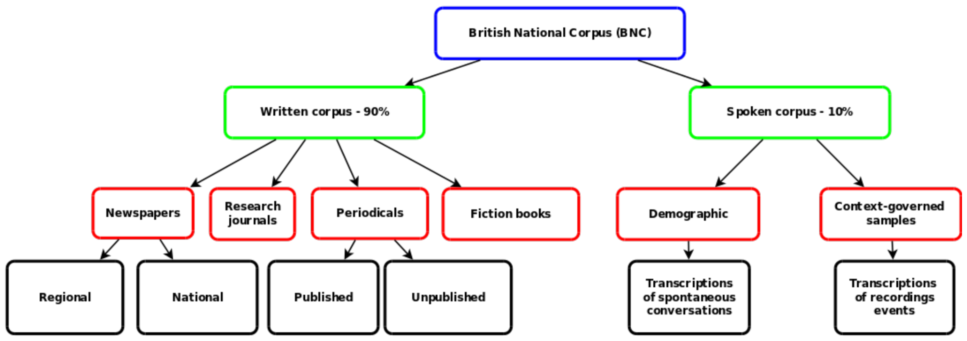


Figure 5: Text Corpus in NLP

Embedding is a strong tool to enable the NLP training to be tunes regarding the semantic spaces of input tokens. Normally, the embedding spaces are initialized by by recursive neutral network. Nevertheless, recurrent neutral network is probability generating the embedding features with very high dimensions, causing great difficulties in effectively training the model. On the other hand, the long-term analysis in time domain also consumes a recorded number of computation energies, which further imposes the great obstacles on dynamic training. For instance, the word king is mathematically decomposed as: queen + woman - man, woman and man are more common words in different context and have their relatively fixed mathematical mapping vectors. Regarding the formulated word relationship, the retrained RNN tackle the equation and generate the appropriate embedding spaces as features vectors for the word: king. In this way, the corpus are also saved memory to store such a large amount words in dictionary, which only enables fundamental vocabulary to be in stock.

3.2 Muti-Layer Neutral Network

[22] Deep neutral networks are developed from multiple layer perceptrons [23], which is capable of extracting features automatically without any manual selection. [24] comprehensively introduces the basic pipeline of neutral network, including the elementary mathematical equation of neutral layer, the way to connect different neutral layers and how layers are integrated into a whole network and finally how data goes into the network for classification as well as back-propagation process. In addition, some good approaches to reducing error rate are also proposed. [25]visualizes important details, especially the mechanism of data flowing and optimization, the paper analyzes them in an easy way. In neutral network, different neutral nodes are connected with each neutral node in last layer. When the data goes through the network, they are calculated via the nodes and from both size and shape. In the layer of mathematics, it is similar as matrix product process, which nodes work as a filter to product with original data and further to be trainable for update.

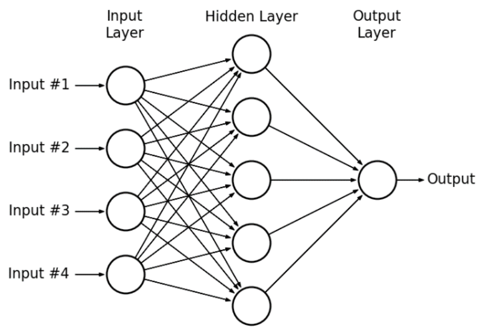


Figure 6: Architecture of Multi-layer Perceptron

In light of back-propagation process, the paper comprehensively demonstrates the gradient descent process. To be specific, after the data pass the whole neutral network, the dataset will be changed to a new result. The loss function computed its difference between label and the new result for extending loss. Based on the value of loss, we are able to acknowledge which direction is the right way leading to gradient descent and then the trainable part in the network will be updated. For the classical neutral network, the function is one-order so the derivative is fixed for each kind of variable. Finally, it summarizes some ways to avoid over-fitting and error elimination, which is beneficial in actual optimization activities.

3.3 From RNN to Transformers: Why RNN is expensive?

Neutral network-based Natural Language Processing is very popular in current AI research,among which Recurrent neutral network is one kind of deep network model applied in the assignment and have achieved great recognition accuracy and very progressive in many aspects, including text classification, natural machines translation, knowledge graph and dialogue generations.

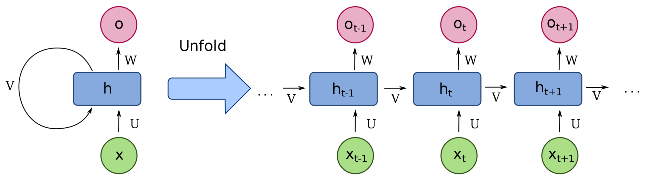


Figure 7: Architecture of General RNN

In recurrent neutral network[26], there is a chain of states conveyed in the architecture for analyzing semantic relationship of sentences. To be detailed, h_t is the hidden state at time step t, v_t is the circulating weights to work as previous reference for present input analysis. From Figure 2, we can clearly see the two important important components and their folded relationship in time-ordered based analysis. x_t represent the token in time t, which is only one or several words in one sentence. For example, we can assume a long sentence: My father and I are looking for a house to be purchased and compare the items in different price. In this case, each token is as "father", "different", "item" and "looking for". The aforementioned token such as "father" and "house" are continuously to be integrated into v_t and then be conveyed to next tokens. Absolutely, their roles or importance gradually decay with the time order going through. Google AI team indicates that the forget stats is the most important part in Recurrent Neutral Network architectures due to its leading roles on deciding the mutual effects exerted by each token in time stamp, so how to fine-tune the forget gate weights are the most challenging part in Recurrent Neutral Network architectures.

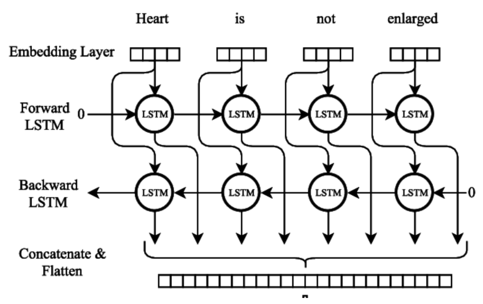


Figure 8: Input Token in RNN Processing

However, the one-by-one input token analysis is processed in time-reliance order, which significantly constrain the efficiencies of model training and inference. To be clear, the Recurrent Neutral Network architecture single unit is responsible for each token in time order, so the token is reliable on state of previous token, which imposed the great restriction on parallel computing capacities. In other words, the success that Recurrent Neutral Network based architecture achieves is at the cost of extreme computational costs and memory demands.

Regarding the issue, some scholars put their efforts into RNN comupting efficiency and elaborations. [27] discusses an economic and simple computational iteration to tackle the compexity of the embedding spaces of sentences efficiently, which is called as scalable log-bilinear models. To be specific, the algorithm is split into two layers: a contextual information feature vectors as well as the scoring function (Softmax Function) to quantify the correlations between the target tokens. The Softmax function will finally estimate the probabilities of each case the select the contextual information as the previously mentioned mathematical reference (the semantic relationship inference).

[1] is another very important milestone in Natural Language Processing, or in other words, it is a very remarkable capstone in end to end architectures. Figure 5 demonstrates its general framework it have also laid the foundations on the following appearance of transformers architectures.

Figure 5 is an example of sequence to sequence learning based on the machine translation scenarios. To be detailed, there are two components in the framework: Encoder and Decoder. The encoder is responsible for processing the input tokens as traditional approach of Recurrent Neutral Network and the final unit will generate the final state that summarizes all information of the whole sentence (the blue unit that is marked as ”DONE”).

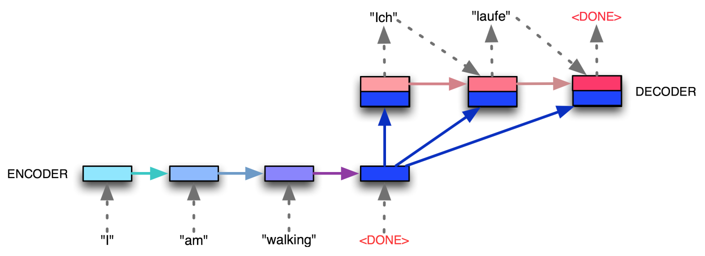
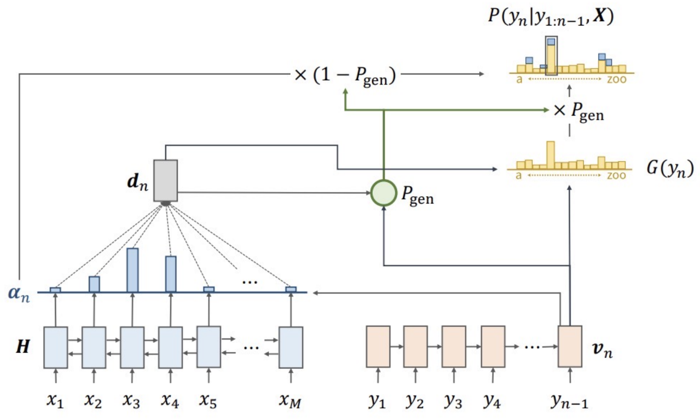


Figure 9: Sequence to Sequence Learning

We can clearly see that the blue states will be conveyed to the decoder, which is responsible to generate the texts that we are expecting. In machine translation scenarios, the decoder is to generate the language that we preset. In other scenarios such as text classification, the decoder is designed to geneate the classification results (normally it is just a number) and the response to the input chat conversations. The blue state will simutaneouly map to all units in decoder and then all unit will go through in time time to process the input from previous unit. Different from the order that encoder follows, the decoder generate the state in the inverse order with the encoder.

According to the reported experimental results, the state-of-art sequence to sequence learning model achieves up to 6% improvement on multiple dominant natural machine translation dataset.

In statistical pattern recognition, scholars widely take actions to specifically mask some tokens/pixels/frames to improve the predictions on the dataset, especially in time-series based analysis. [28] proposes an very effective masking methods in sequence to sequence learning. Generally, no matter in encoder or decoder, the previous state is essential to the training of next state. However, if the we mask specific tokens in one sentence, such as "I XXX(like) my dog because he is very cute". "like" is masked in the sentence but it can be infered from the contextual information "he is cute". Similarly, in statistical inference masking tokens are beneficial to improve the predicting abilities. It is like to train human brains with higher-level difficult tasks to intrigue better generalization abilities. The literature is focused on this phenomenon and investigate the start-to-point methodology to dynamically mask specific units in both encoders and decoders to achieve better model performances.



$P(y_n|y_{1:n-1}, \boldsymbol{X})$
$\times (1 - P_{\text{gen}})$
$\times P_{\text{gen}}$
$G(y_n)$
$\boldsymbol{d}_n$
P_{gen}
$\boldsymbol{\alpha}_n$
$\boldsymbol{H}$
$\boldsymbol{v}_n$
x_1 x_2 x_3 x_4 x_5 x_M
y_1 y_2 y_3 y_4 y_{n-1}
a
zoo

Figure 10: Start-to-Point Masking in Seq2Seq Learning

In light of the more improvements on sequence to sequence learning, the pretrained methods are introduced to enhance the generalism of linguistic tasks. [29] indicates a state-of-art approach to tackle the long-memory diminishing in seq2seq, which is also serious in this architectures. The literature firstly proposed to train the recurrent neutral networks using limited dataset or the dataset with unrelevant i.i.d. In this way, the weights in each recurrent neutral network will be roughly tuned to achieve preset accuracy threshold. After that, the full dataset or the target datasey will be applied in retraining with less training epochs and effectively eliminate the overfitting parameters to some extents. The approach is also introduces into following BERT (Bidirectional Early Training Transformer) but transformers have better generalism abilities compares with the sequence to sequence learning system.

3.4 Why Transformers are Efficient?

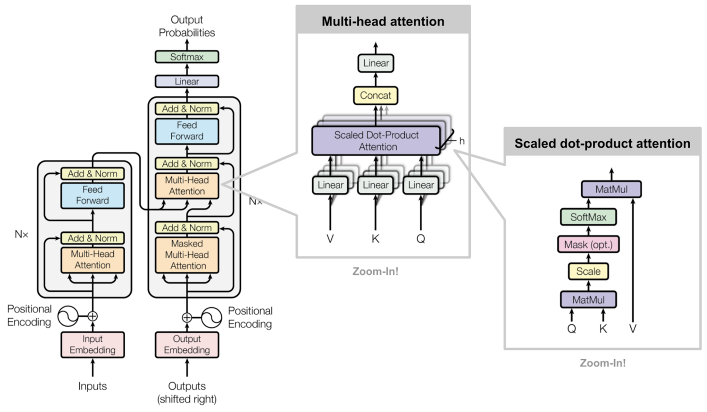


Figure 11: Multi-head Attention in Transformers

Figure 7 presents the detailed architecture of transformers, including the data pipeline (input embedding, positional encoding, encoder and decoder, output embedding), the feature vector projections in decoder and encoder and the scaled computations. Diving into ther statistical mechanism, it is similar as sequence to sequence learning that the final state of encoder will conver all the information from all units to decoder. Nevertheless, the difference is that sequence to sequence learning is centralized on processing tokens one-by-one, which decompose the whole sentence and consume much time. For multi-head attention framework, the whole sentence is not split into multiple tokens and then embedded, on the contrary, they are embedded together and performed positional encoding to mark the relative relation, which is simulating the one-by-one unit multiplication in seq2seq learning.

Based on the positional marks, transformer deploy the token projection to score the relative correlation via k, q, v trainable weights. The dimension of the k,q,v matrices are the same as the embedding size of input samples. The essence of the projection is to project the whole sentence into a feature space (k), and then k projects to another feature space. In this way, each token will obtain a normalized score from every other tockens, representing the semantic correlations with each. Compared with sequence to sequence learning, the multi-head attention statistical mechanism does not require specific input in time stamp and enabling the sequential learning to be deployed in parallelism.

On the other hand, the bridge between encoder and decoder also include more abundance information without forgetting unnecessary contextual representations. In previous descriptions on sequence to sequence learning, the feature representations are only generated from last unit in the architecture of recurrent neutral networks. By contrast, transformers analyze the whole sentence in parallelism and then load the complete representations into encoder, which provides very complete inference on the encoder and facilitate the computing efficiencies.

[30] illustrates the importance of forget gate in Long Short Term Memory architecture, which outperforms other components in this structure. However, although transformers utilize the concrete correlation score to quantify the importance of tokens, it is not as effective as LSTM to specifically "forget" some feature information so as to decrease the general accuracy in some

task.

What is more, transformer is worse in long sentence process compared with LSTM-like architecture due to the distant token information fastly decaying with the facilitation on k, q, v weight training. In order address the issue, many researchers propose some enlightening ways to tacker the long-term decaying regarding the transformer framework.

[31] integrate the objective of long short term memory architecture into transformer framework and demonstrate state-of-art long term processing abilities. The literature grasp the projection score mechanism, enabling each token projection to to correlated with each other using additional sequential communications. From Figure 9, each tokens is connected with each other one to simulate the configuration of LSTM and tuned during gradient descent.

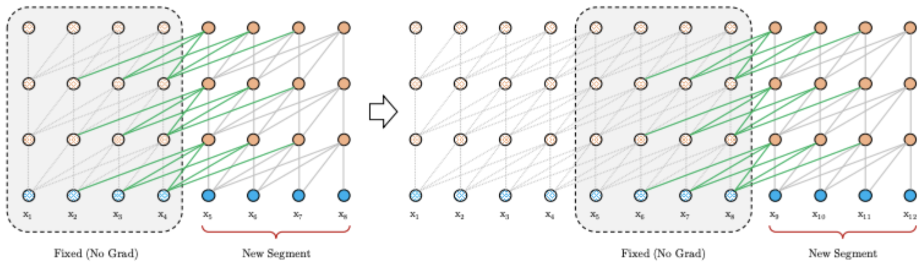


Figure 12: Architecture of TransformerXL

[32] is another important capstone in transformer framework training, which is called Deep Bidirectional Transformers. There are two existing strate-

gies for applying pre-trained language representations to downstream tasks: feature-based and fine-tuning. The paper discusses the two issues and introduces its unique way to explore better pre-trained modes. To be detailed, there are two core objectives in author's exploration: 1. The paper points out that the essence of language representations is to utilize mask mechanism to activate pretrained deep bidirectional representations. 2. Pre-trained representations simplify the power-consuming architecture training stage, which is potential to adapt to all kinds of language task.

In order to process the mask issue, the masked tokens go through the whole architecture and make predictions based on its context. What is more, Masked LM and Next Sentence Prediction (NSP) are introduced to pre-train BERT and they serve as finely tuning the parameters in the architecture to adapt to high performance. These ways contribute to advance the same pre-trained model to successfully tackle various NLP tasks.

4 Application

As we have discussed in the previous chapter that multi-head attention mechanism has achieved great success and comprehensively analyze the rooted reason for this. In this chapter, we are getting down to the specific application scenarios and introduce the roles that transformers play in these areas.

4.1 Natural Language Processing

Natural Language processing is definitely the widest area that transformer are applied. [33] and [34] demonstrates that the application of transformer in text classification as well as sentimental analysis.

To be detailed, there are two main design on the text classification task. The first one is to only preserve the encoder part without the decoder part, then the encoder directly links with a classifier. Normally, when the feature vectors of input samples will be performed mean-pooling on the length size, and then the word embedding dimension product with a multi-layer perceptron to be transmitted to the vectors with the sample class number as labels. As Figure 11 shows, we can clearly see that MLP replaces the encoder layer and works as the classifier to process the final feature vectors.

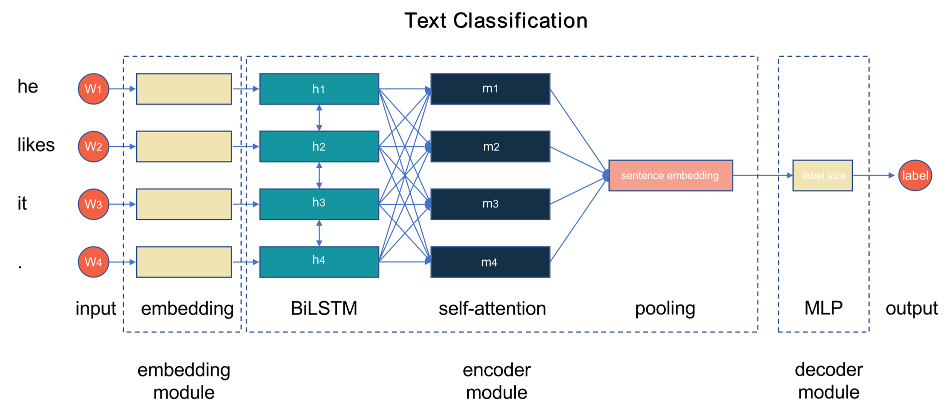


Figure 13: Text Classification

On the other side, some researcher don't thoroughly substitute the classifier for MLP, which means that they deploy the attention as a part of classifier. We all know that tranditional linguistic model contains encoder, decoder as the main components. The decoder works as predictions for target feature vectors from encoder using attention mechanism. Nevertheless, the preservation on decoder is not very effective and meaningful for token correlation in a whole sentence regarding the function conducted by encoder. In addition, the text classification is not for generating concreted information/sentence in respect to inputs, only enabling the network to generate the probabilities of each class or even a judging label. As a consequence, the first approach is popular in current transformer framework.

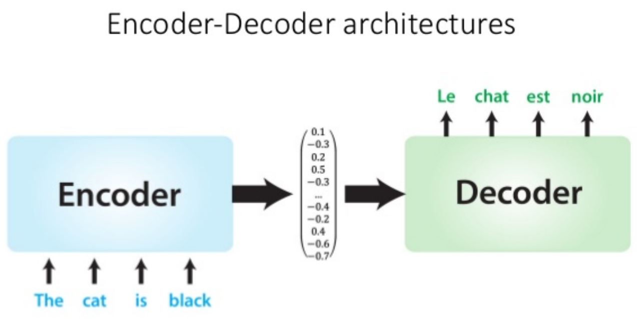


Figure 14: Machine Translation

4.2 Machine Translation

Machine translation is a very typical scenarios for end-to-end sequence learning, which concentrates on deploying end-to-end framework to generate possible mapping translation and finalizing the sentences with the largest probabilities. Actually, translation is also a kind of classification issue. The encoder structure in transformer framework analyze each tokens and produce a score matrix [35], [36], [37]. The score matrix go into the decoder attention layer [38],[39] which searches for the best match for each token. The "searching" process is to employ softmax function to look for the token with highest probability to target tokens [40].

As what we have mentioned in sequence to sequence learning system, whole sentences are split into a series tokens and the tokens are loaded into the sequential neutral units to be processed and trained in its order. The transformers framework totally abandon the structure to pay attention to the whole sentence analysis [41]. The complete sentences are projected but lack the long-dependency generality, which is implemented by Transformer XL architecture. Nevertheless, according to the experiments performed by [42], the medium and short length exactly demonstrated the good performance and even outperform the sequence to sequence learning framework. As a consequence, the sequence to sequence learning and the multi-head attention learning have their own advantages in machine translation domain and the step-by-step learning framework is much better in long-dependency situations.

4.3 Knowledge Graph

[43] introduces the application of transformer framework in knowledge graph generations. In this paper, the author proposes to utilize Transformer to automatically generate commonsense, which is similar to the function of knowledge graph construction. Transformer is a multi-head attention mechanism

to take care of the semantic relationship and investigate its contextual meaning. To be specific, authors apply K,Q,V trainable matrices to do projection on the word and obtain the possibilities of certain words in commonsense meaning. It could be explained as a mapping relationship from specific word to matched knowledge generation.

To be detailed, multiple pair of commonsense work as training set, which are be in respect to encoder inputs and decoder output respectively. Similar to the machine translation or text classification mask in transformers, the training pairs serve as mapping words/sentences from encodes and decoder in Transformer architecture and encoder pairs pass the transformer and generate the mapping words. The mapping words is computed probabilities and obtain the loss to do gradient descent. In this process, mask is a hard part to process because the same token position in one sentence does not mean the same semantic representation.

The paper process it randomly mask the same presentation in both encoding information and decoding information before training. After that, author utilize multiple dataset to test the performance of Transformer in commonsense knowledge representation. The results demonstrate that Transformer is capable of processing mapping representation in the commonsense graph system.

4.4 Speech Recognition

Speech recognition [44] is an aspect in traditional signal processing and recently also apply multiple deep model to analysis specific sound source and signal pattern. In general, speech recognition [45] is focused on the timbre recognition and classification, blind source separations as well as sound generation. No matter what kind of topics or pattern, the signal or sound [46] is also a kind of time series data and the end-to-end framework [47] is able to process it [48].

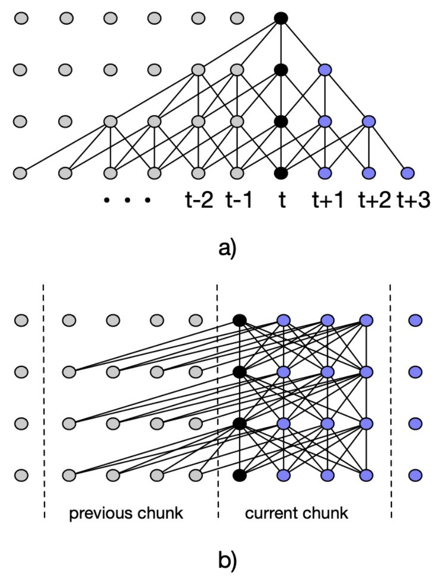


Figure 15: Speech Recognition Using Transformer

[49] illustrates the application of Transformer and Transformer XL in speech

recognition, which process the sound frame as tokens. Similar as the sentence token concept in natural language processing. As what show in Figure 15, the chunks are correlated with each other in multi-head attention system and they are parallel to be processed. The chunk is the audio frame for split speech inputs.

Another issue in speech recognition-based transformer is the chunk masking mechanism. As what describes in the literature, ASR systems [50] are usually deployed in the streaming fashion which produces real-time speech transcription with certain latency constraint. In other words, the streaming features are likely to disable the possibility of contextual analysis for the whole sentence projection and mapping space, which causes great difficulties in head-based scoring mechanism.

Regarding the issue, the author design another system to tackle the streaming ASR in speech recognition. To be specific, they constrain the available feature context for training and evaluation period. Usually, other people prefer to take actions on considering lookahead with an attention mask, enabling every node to be accessed with small amount of activiation from from both the future and the past timesteps. Nevertheless, the method experiences risky context expansion, which means that the size of context will explode with the sample going into deeper architecture. As a result, the paper imposes the limitation on the number of feature that can enter each layer via small context window for fitting into the latency constraints.

4.5 Computer Vision

Deep learning model has achieved great success in computer vision tasks, including object detection, image recognition and image segmentation. Figure 15 is an example for object detection regarding one example image, which navigates the cars and traffic signs.

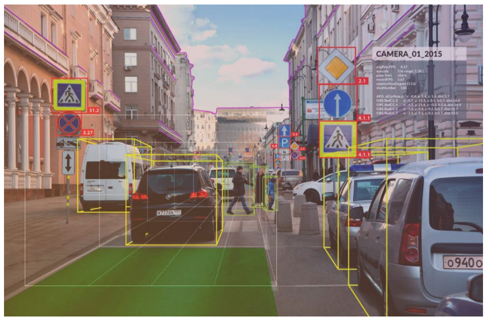


Figure 16: Object Detection in Computer Vision

For the most beginning boosting of computer vision via deep learning framework, convolution neutral network is the pioneering model in this task. The layer of convolution effectively grasp the pixel-based matrix via convolution computation and then the feature matrices are conveyed into pooling layer to further the feature compression as well as extraction. However, the recurrent neutral network and transformer framework [51], [52], [10] is developed from sequential information processing, which is not very suitable for processing the static feature without ordered sequence.

[53] makes investigation on deploying transformers for computer vision tasks. Generally, researchers propose to combine the attention mechanism with

convolution neutral networks. For more details, we can assume that there is a input image waiting to be processed, it will firstly to go thorough the convolution layer and then pooling layer as common CNN pipeline. After that, the compressed feature vectors will enter the multi-head attention layer for scoring analysis as what normal natural language processing do without the positional encoding. According to the experimental results, the network combination is able to outperform the vanilla convolution neutral network in multiple computer vision tasks.

Diving into the reason for the improvement, we can find that the attention mechanism takes into consideration that the whole configurations. For the image processing, it is definitely that the object in the picture is not sequential and they are in specific time order [54]. Nevertheless, like the object detection task, the object are also related to each other. For example, in Figure 30 we can clearly see that it is to analyze the traffic-related object and they are around a busy road. The attention mechanism is able to manipulate some unrelevant object with target object, which means that even through the road and walkers are not what we would like to recognize but they are valuation to enable the analysis to obtain more meaningful information. The traffic signs are always around the road side and it is much easier for the convolution networks to rapidly grasp these features.

On the other hand, the visual transformers are not bound to enhance the classification or recognition performance. And as well, the additional attention network escalate the overcomplete model issues, which means the deeper layer are overfitted and then play very trivial roles in the whole task recognition. Although the residual connections are widely applied, the large computing consumption are another great concerns.

5 Conclusions

This literature survey comprehensively summarizes the architecture of transformer framework, the statistical inference of its multi-head attention mechanism and clarify its wide applications in machine translation, text classification, knowledge graph, speech recognition and computer visions. Mant related and enlightening literatures are cited to enhance the acknowledgement on this state-of-art architectures.

Based on the three pipelines, we can clearly see that current efforts made to elaborate transformer architectures are concentrated on improving its performance on target task and its efficiencies. Regarding the concrete approach, the layer dynamics centralized on decreasing the total number of layers, head dynamics centralized on masking or pruning specific heads, attention dynamics centralized on elaborating the attention space for different heads are three main tendencies. The elaboration methodology can be roughly split to direct statistical observation and reinforcement learning based on indirect observation from network training as well as inference.

For our analysis on the statistical mechanism of transformers, we mainly compare it with another popular framework: Recurrent Neutral Network and its sequence-to-sequence learning design. We clearly demonstrate that what is the differences between these two mechanisms and how it can be shown the closed capabilities of processing token in timestamps. We also discuss some new developments on the architecture to tackle long dependency issue and the model pre-trained strategy for wider industrialized application.

When it comes to the applications, we have discussed how transformer is employed for specific tasks and how the features are extracted though multi-head attention mechanism. It is obvious that transformer become more and more general architecture and capable of processing many types of tasks with less computational costs and high recognition performance.

References

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[2] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. Compressing large-scale transformer-based models: A case study on bert, 2020.

[3] Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A. Gers.

How does bert answer questions? a layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1823–1832, New York, NY, USA, 2019. Association for Computing Machinery.

[4] Z. Song, Y. Xie, W. Huang, and H. Wang. Classification of traditional chinese medicine cases based on character-level bert and deep learning. In *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pages 1383–1387, 2019.

[5] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one?, 2019.

[6] Fanfei Meng, Lalita Jagadeesan, and Marina Thottan. Model-based reinforcement learning for service mesh fault resiliency in a web application-level, 2021.

[7] Chenao Wang Fanfei Meng. A dynamic interactive learning interface for computer science education: Programming decomposition tool. anonymous preprint under review, 2023.

[8] Yu Chen Fanfei Meng, Lele Zhang. Fedemb: An efficient vertical and hybrid federated learning algorithm using partial network embedding. anonymous preprint under review, 2023.

[9] Yu Chen Yuxin Wang Fanfei Meng, Lele Zhang. Sample-based dynamic hierarchical transformer with layer and head flexibility via contextual bandit. anonymous preprint under review, 2023.

[10] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

[11] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

[12] Ruxin Tan, Jiahui Sun, Bo Su, and Gongshen Liu. Transformer-dw: A transformer network with dynamic and weighted head. In Tom Gedeon, Kok Wai Wong, and Minho Lee, editors, *Neural Information Processing - 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12-15, 2019, Proceedings, Part II*, volume 11954 of *Lecture Notes in Computer Science*, pages 504–515. Springer, 2019.

[13] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 331–335, Florence, Italy, July 2019. Association for Computational Linguistics.

[14] Shakti Kumar, Jerrod Parker, and Panteha Naderian. Adaptive transformers in rl, 2020.

[15] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015.

[16] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[17] James O' Neill. An overview of neural network compression, 2020.

[18] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9782–9793. Curran Associates, Inc., 2020.

[19] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache, 2016.

[20] Tal Linzen. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 13–18, Berlin, Germany, August 2016. Association for Computational Linguistics.

[21] Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics.

[22] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications, 2017.

[23] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[24] Maciej Krawczak. *Multilayer Neural Networks: A Generalized Net Perspective*. Springer Publishing Company, Incorporated, 2013.

[25] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 2017.

[26] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.

[27] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[28] Mana Ihori, Naoki Makishima, Tomohiro Tanaka, Akihiko Takashima, Shota Orihashi, and Ryo Masumura. Mapgn: Masked pointer-generator network for sequence-to-sequence pre-training, 2021.

[29] Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. Unsupervised pretraining for sequence to sequence learning, 2018.

[30] A. Pulver and S. Lyu. Lstm with working memory. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 845–851, 2017.

[31] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[33] Linyi Yang, Eoin M. Kenny, Tin Lok James Ng, Yi Yang, Barry Smyth, and Ruihai Dong. Generating plausible counterfactual explanations for deep transformers in financial text classification, 2020.

[34] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[35] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning deep transformer models for machine translation, 2019.

[36] Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. Tied transformers: Neural machine translation with shared encoder and decoder. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5466–5473, Jul. 2019.

[37] Shaowei Yao and Xiaojun Wan. Multimodal transformer for multimodal machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4346–4350, Online, July 2020. Association for Computational Linguistics.

[38] Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. Very deep transformers for neural machine translation, 2020.

[39] Nikolay Banar, Walter Daelemans, and Mike Kestemont. Character-level transformer-based neural machine translation, 2020.

[40] Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted transformer network for machine translation, 2017.

[41] Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. Fixed encoder self-attention patterns in transformer-based machine translation, 2020.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[43] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction, 2019.

[44] Laura Cross Vila, Carlos Escolano, José AR Fonollosa, and Marta R Costa-Jussa. End-to-end speech translation with the transformer. In *IberSPEECH*, pages 60–63, 2018.

[45] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

[46] Yongqiang Wang, Abdelrahman Mohamed, Due Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, et al. Transformer-based acoustic modeling for hybrid speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6874–6878. IEEE, 2020.

[47] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. A comparative study on transformer vs rnn in speech applications. In *2019 IEEE Automatic*

Speech Recognition and Understanding Workshop (ASRU), pages 449–456. IEEE, 2019.

[48] Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. Improving the transformer translation model with document-level context. *arXiv preprint arXiv:1810.03581*, 2018.

[49] Liang Lu, Changliang Liu, Jinyu Li, and Yifan Gong. Exploring transformers for large-scale speech recognition, 2020.

[50] Li Deng and Xiao Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, 2013.

[51] Dong Chen, Gang Hua, Fang Wen, and Jian Sun. Supervised transformer network for efficient face detection. In *European Conference on Computer Vision*, pages 122–138. Springer, 2016.

[52] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*, 2020.

[53] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A survey on visual transformer, 2021.

[54] Tae Hyun Kim, Mehdi SM Sajjadi, Michael Hirsch, and Bernhard Scholkopf. Spatio-temporal transformer network for video restoration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 106–122, 2018.