1 Introduction and Motivation

The convolution operation has precedent mainly in image processing for being able to effectively extract spatial information by pooling multiple pixels together across the entire image. We propose this methodology can also be effective for NLP tasks in extracting temporal information by utilizing information from other timesteps.

2 Architecture

2.1 Classical Multi-Head Attention

The multi-head attention layer takes a sequence of embeddings as input. That is,

$$X \in \mathbb{R}^{t \times d_e}$$

where t is the number of timesteps (how many words or tokens are in the sentence to be translated) and d_e is the dimensionality of the token embedding. The embedding is calculated by passing one-hot encoded token inputs through a standard linear transformation (without bias parameters) to generate a lower-dimensional encoding.

The purpose of the attention layer is to learn which tokens are important in predicting output tokens. For example, when translating the beginning of a sentence, it might not need information from the end of the sentence to generate that prediction. Attention layers attempt to capture this idea.

We introduce 3 matrices, Q, K, V which are all functions of the original input. f is a standard linear transformation. That is,

$$Q = f(W_Q, X)$$
$$K = f(W_K, X)$$
$$V = f(W_V, X)$$

where

$$W_Q \in \mathbb{R}^{d_e \times d_m}$$
$$W_K \in \mathbb{R}^{d_e \times d_m}$$
$$W_V \in \mathbb{R}^{d_e \times d_m}$$

and d_m is the dimensionality of the model. We choose d_m and d_e to be equal.

Each "head" learns a different attention feature.

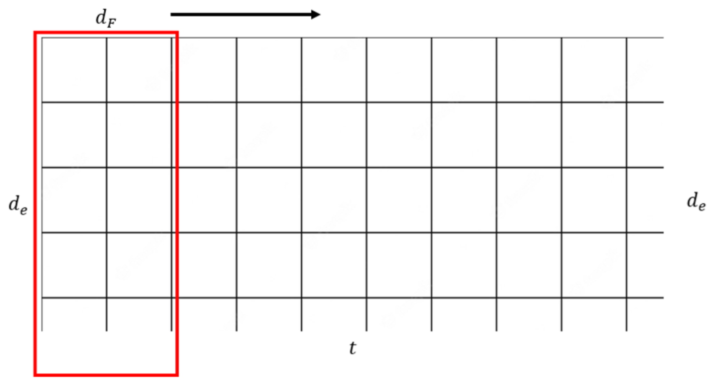
$$d_m = h \cdot d_h$$
$$reshape(Q) \rightarrow Q \in \mathbb{R}^{h \times t \times d_h}$$
$$reshape(K) \rightarrow K \in \mathbb{R}^{h \times t \times d_h}$$
$$reshape(V) \rightarrow V \in \mathbb{R}^{h \times t \times d_h}$$

$$attention\ matrix = QK^T \in \mathbb{R}^{h \times t \times t}$$
$$scaled\ attention\ matrix = \frac{QK^T}{\sqrt{d_m}} \in \mathbb{R}^{h \times t \times t}$$
$$attention\ probabilities = softmax(\frac{QK^T}{\sqrt{d_m}}) \in \mathbb{R}^{h \times t \times t}$$
$$scaled\ dot\ product = softmax(\frac{QK^T}{\sqrt{d_m}})V \in \mathbb{R}^{h \times t \times d_h}$$
$$reshape(SDP) \rightarrow SDP \in \mathbb{R}^{t \times d_m}$$

2.2 Convolution Attention

We introduce a new type of Q, K, V mapping which uses convolutional transformation instead of linear transformation.

$$Q = X * F_Q$$
$$K = X * F_K$$
$$V = X * F_V$$



The main difference between CV kernels and our NLP kernel is the shape. Our kernel has a height that spans the entire input dimension, instead of a square kernel. This is because it is too hard to learn spatial locality in the embedding dimension, so it is easier to just combine all the information along the entire embedding dimension into one convolution.

$$F_Q \in \mathbb{R}^{d_F \times d_e \times 1}$$
$$F_K \in \mathbb{R}^{d_F \times d_e \times 1}$$
$$F_V \in \mathbb{R}^{d_F \times d_e \times 1}$$

For each Q, K, V, we choose d_m number of filters to match the output dimensionality of classical attention. The input is also padded such that the output width (along the temporal dimension) is the same as the input.

$$Q \in \mathbb{R}^{d_m \times t \times 1}$$
$$K \in \mathbb{R}^{d_m \times t \times 1}$$
$$V \in \mathbb{R}^{d_m \times t \times 1}$$
$$reshape(Q) \rightarrow Q \in \mathbb{R}^{t \times d_m} \rightarrow Q \in \mathbb{R}^{h \times t \times d_h}$$
$$reshape(K) \rightarrow K \in \mathbb{R}^{t \times d_m} \rightarrow K \in \mathbb{R}^{h \times t \times d_h}$$
$$reshape(V) \rightarrow V \in \mathbb{R}^{t \times d_m} \rightarrow V \in \mathbb{R}^{h \times t \times d_h}$$

The rest of the attention calculations proceed identically as listed above.

2.3 Inception-Style Convolution Attention

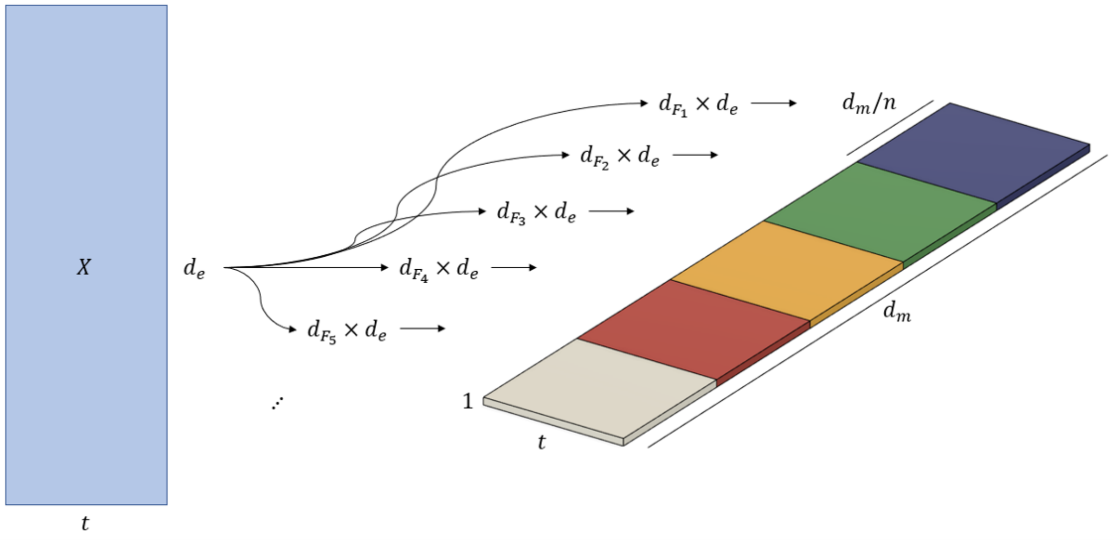
Rarely does a word's context depend on a constant width of timesteps around that word. That is, using a kernel with constant width does not make sense from an NLP standpoint. To address this, we introduce multiple kernel sizes and stacking in an Inception-like architecture style.

For example, we choose two convolution kernel sizes, d_{F1} and d_{F2}. Then, for each Q, K, V, we only perform each convolution on half the output dimension. That is,

$$Q \in \mathbb{R}^{d_m/2 \times t \times 1}$$

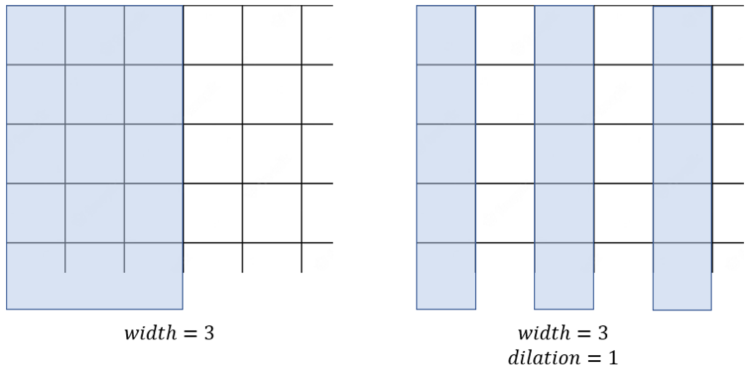
$\vdots$

The output size is still the same, but different convolution types can be used and are stacked along the $num\ filters$ dimension. For n number of convolution types (different kernel widths), each will have d_m/n number of filters.



2.4 Inception-Style Convolution Attention with Dilation

Lastly, we introduce dilated convolution to this architecture. Dilated convolution inserts holes between kernels elements to have a larger temporal reach than traditional convolutions.



3 Results

Initial testing was done on the WMT NewsCommentary v14 dataset, a collection of $\sim$ 300000 sentences pulled from news articles. Our benchmark currently is the base Transformer architecture. We build an identical Transformer architecture which instead uses convolution attention at each Multi-Head Attention layer instead of the original implementation.

The numbers in parentheses denote the convolution width. A “d” denotes the dilation size.

Architecture	Accuracy	Epochs Taken (max 100)
Transformer	.90	61
	.95	never reached
Conv Transformer (1, 3, 5, 7)	.90	41
	.95	68
Conv Transformer (1, 3, 5, 7, 3d2, 5d2, 7d2)	.90	40
	.95	64

So far, we observe much faster time-to-convergence and potentially better accuracy. Our current best model trains to 90% accuracy 34% faster. It trains to 95% accuracy in 64 epochs, compared to Transformer which never reaches 95% after 100 epochs (>36% faster).

It is important to note that all three models had almost the exact same number of parameters, thus, our model is not performing better because of more parameters, but because it is able to learn at a faster rate.

Additionally, we do not report on accuracy metrics at the current moment. This is because it is not clear whether our model has superior time-to-convergence as well as accuracy, or just time-to-convergence.

4 Future Work

- Extensive parameter testing (different filter sizes, different dilation sizes, etc.) Default configurations of (1, 3, 5, 7) were used out of convenience but better configurations probably exist and need to be tested.
- Compare BLEU scores as opposed to accuracy. BLEU is a better metric for evaluating language translation.
- Evaluate how well our model scales. Transformer has several variants with different numbers of layers, feature dimensions, etc., so smaller/bigger versions of our model need to be tested to see how it compares to other versions of Transformer.
- Investigate papers released since Transformer which have improved translation quality. The original Transformer is no longer state-of-the-art, and though most new architectures are based on Transformer, we must evaluate them to see how our model compares against current architectures. Currently, we use the original Transformer as a good benchmark to see how our model behaves.