EECS 6322 Week 4, Paper 1
Convolutional Sequence to Sequence Learning

Shivani Sheth (Student #: 218011783)
shivs29@yorku.ca

I. Summary and contributions

The paper proposes a novel convolutional architecture for sequence to sequence learning which consists of Gated Linear Units (GLUs) and Attention functions at each decoder layer. The model uses position embeddings that embeds both input and output elements along with their absolute positions to determine the portion of the sequence that the model is dealing with. The input elements x_m are embedded in distributional space w_m and their absolute positions are stored as p_m. Hence the input elements are represented by combining both vectors as (w_1 + p_1 + ... + w_m + p_m). A similar representation is also used for the output vector.

The intermediate states of the encoder and decoder are represented by block structures based on fixed numbers of input elements. The block structures are followed by a one dimensional convolution layer and a non linearity function that either enhances the important input features or lowers the effect of unimportant input features. The output of the l^{th} block for the decoder network is denoted by $h^l = (h^l_1, ..., h^l_n)$ and for the encoder network is denoted by $z^l = (z^l_1, ..., z^l_n)$. Each convolution layer takes in a concatenation of k input vectors embedded in d dimension and maps it into a single output vector y.

The architecture uses Gated Linear Units (GLUs) to embed non linearity into the model given by $v([AB]) = A \otimes \sigma(B)$. Here the $\sigma(B)$ acts as a filter that increases the value of relevant features and decreases the value of unimportant features thus improving the flow of data. The model also uses residual connections to enable deep layers where the input of each convolution is added to the output of the block. The inputs to the encoder and decoder are padded to maintain dimensions in the model.

The decoder takes in a weighted sum of the conditional output from the encoder layer, along with the input vector embeddings. These input vector embeddings were found to be beneficial to the system. A multi step attention model is added at the decoder layer where the output of the attention layer at time t is added as an input to the next $t + 1$ time step. Hence, a given layer in the decoder has the history of all the attention layers up to its previous point which helps the model understand the previous inputs that it has already attended. The variance of activation functions are maintained throughout the network in the forward and backward passes by scaling outputs of attention/ residual layers and careful initialization of weights.

The architecture was tested on several datasets such as WMT'16 English-Romanian, WMT'14 English-German, and WMT'14 English-French. It was implemented using 512 hidden units in both encoders and decoders, and the convolutional models were trained with Nesterov's accelerated gradient method using a momentum value of 0.99. A learning rate of 0.25 was applied to the model and mini batches of 64 sentences were used. The translations were generated by a beam model using a beam width equal to 5. The results were computed using case-sensitive tokenized BLEU method for all datasets except for WMT'16 English-Romanian, for which detokenized BLEU was used.

On the WMT'16 English-Romanian dataset translation, the architecture outperformed the previous winning entry of the dataset, that used attention-based sequence to sequence architecture, by 1.9 BLEU with a BPE encoding and 1.3 BLEU with a word-factored vocabulary. On the WMT'14 English to German dataset translation, it outperformed GNMT that used eight encoder LSTMs and eight decoder LSTMs by 0.5 BLEU. Similarly, on the WMT'14 English to French dataset translation, it performs better than GNMT by a score of 1.6 BLEU on average. The generation speed of the architecture is proved to be significantly higher than its counterparts and is about 13.7 times faster with a better BLEU score on GPUs and about 9.3 times faster with a better BLEU score on CPUs.

Overall, the convolutional sequence to sequence architecture performed significantly better with lower computational costs. The positional embeddings helped the model perform better but experiments on the model without the positional information also performed almost as well. The multi step attention mechanism helped the model perform significantly better in terms of efficiency, and deeper architectures with narrow kernels were found to perform better than the architectures with wide kernels.

II. Strengths

The convolutional networks do not depend on the previous time steps and hence allow parallel computations that decrease complexity and resource cost as compared to Recurrent Neural Networks which depend on the hidden layer of the previous time step. They can control the maximum length of dependencies in the network by creating representations for fixed size contexts. By using these networks, long range dependencies from the input vector can be captured in a hierarchical structure where the nearby input elements interact at a lower level, while the distant elements interact at a higher

level. Convolutional networks also increase the learning rate of the network by increasing the nonlinearity functions that the input passes through, as compared to Recurrent Networks where nonlinear functions are only applied at the start and end of the network.

III. Weaknesses

A few weaknesses of the architecture can be summed up as follows. The model requires careful initialization of weights and scaling up of the output layers for its optimal performance. This can take a considerable amount of work and time in the initial stages of the architectural implementation. The models are also not optimized for applications where the input sequences need to be reduced, that is, it takes a long input vector and outputs a shorter version of the input. The existing architectures currently perform better than proposed architecture for these applications. Another limitation of the model is that the deeper layers in the decoder seem to decrease the overall performance of the model, which is counter intuitive.

IV. Correctness

The claims made by the authors are based on convolutional networks that are applied in sequence models and are found to be correct through a series of experiments. The benefits of convolutional networks are combined with the modified architecture of sequence models which not only seem effective theoretically, but also prove to be quite efficient in different applications. A fair comparison has been made on several big datasets with its state of the art counterparts which strengthen the claims made by the authors.

V. Clarity

The paper is well written in general with adequate illustrations and graphs which makes it easier for the reader to understand. The calculations for the weight initialization and the activation function have also been detailed at the end of the paper which clarifies the theory of the implementation details. The architectural details as implemented on each dataset has also been noted along with the training time and computational resources, which gives a clear picture about the experimental results.

VI. Relation to prior work

Previous sequence to sequence learning architectures were mainly based only on recurrent neural networks that used encoders and decoders to generate an output sequence. The generic flow of these models used encoders to compute a 'many-to-one' generation model that took in an input sequence and converted it to a single input vector, known as the Context Vector. The second part of the model then used layers of decoders to compute a 'one-to-many' generation model that took the Context vector as an input and generated output sequences at each time step 't'. The models used weight vectors and hidden layers where each hidden layer was computed based on its previous hidden layer and input, multiplied by their respective weights. This was the general architecture used for most of the previous models.

A few models also used the attention mechanism to specify the parts of the input sequence that the model should pay more attention to while generating the output sequence. This attention was computed by a weighted sum of attention parameters multiplied by state representations z_m. Some of the other popular sequence network architectures such as LSTM and GRU used gated connections to maintain long term dependencies. It made backpropagation more effective which increased the accuracy of the models and helped with the exploding/ vanishing gradients problem faced in earlier models.

The convolutional sequence to sequence model, on the other hand, is the first fully convolutional model used for sequence learning that performs significantly better than the previous architectures. The model introduces a multi step attention mechanism for each decoder layer which increases the overall efficiency of the model. The convolutional layers also contribute to parallel computations across the input layers which significantly increases the generating time of the model.

VII. Reproducibility

The proposed architecture can be reproduced in common deep learning frameworks such as Pytorch. The source code details, as provided in the GitHub link, provides adequate information on how to implement the architecture and also includes a summary of the package/ computational requirements, installation procedures, and training/ generating procedures.

VIII. Additional comments

Although an overview of the architecture is illustrated in Figure 1, a more base-level illustration of the architecture depicting the change over each time-step on the encoder and decoder side would help make the understanding more intuitive.