ETH zürich

Networked Systems
ETH Zürich — seit 2015

Thesis Title

Master Thesis

Author: Siddhant Ray

Tutors: Alexander Dietmüller, Dr Romain Jacob

Supervisor: Prof. Dr. Laurent Vanbever

Februrary 2022 to August 2022

Abstract

An abstract is a short summary placed prior to the introduction used to help readers determine the purpose of the thesis. While the length of the abstract varies by field of study, it is typically a paragraph in length (3-5 sentences), and never more than a page.

Contents

Chapter 1

Introduction

Motivate the problem, introduce your work and give an overview of the report.

1.1 Motivation

The motivation to write this thesis was ...

1.2 Task and Goals

Describe your task and the goals you achieved.

1.3 Overview

Section 2 describes ..., Section 3 presents ...

Chapter 2

Background and Related Work

A large part of the work undertaken during this project requires a significant understanding on how a particular deep learning architecture, the Transformer works. In this section, we will cover some of the required background and insights drawn from the Transformer architecture which were needed to model and solve our problem of prediction on packet data. We also present adaptations of the Transformer architecture to solve problems in several fields such as NLP/CV and relevant ideas which could be adapted to our tasks.

2.1 Background on Transformers

2.1.1 Sequence modelling with attention

Transformers are built around the *attention mechanism*, which maps an input sequence to an output sequence of the same length. Every output encodes its own information and its context, ie information from related elements in the sequence, regardless of how far they are apart. The process involves the scalar multiplication of the input feature matrix attention matrix as a dot product operation, which allows the deep neural network to focus on certain parts of the sequence at a time, based on the values of the attention weight matrix. This allows the network to attend to parts of the sequence in parallel, rather than in sequence, which allows highly efficient computation. Also, as the attention weights are learnable parameters for the network, the Transformer over time, learns to choose the best weights which allow optimum learning of structure within the sequence of data. Computing attention is efficient as all elements in the sequence can be processed in parallel with matrix operations that are highly optimised on most hardware. These properties have made Transformer based neural networks the state-of-the-art solution for solving many sequence modelling tasks. We refer to an excellent illustrated guide to the Transformer here[1].

While attention originated as an improvement to Recurrent Neural Networks(RNNs), it was soon realised that the mechanism could replace them entirely.[8] RNNs were the initial state-of-the art deep learning architectures in sequence modelling problems, however they suffer from several issues. Training RNNs is usually limited by one or all of the following problems:

1. RNNs are not computationally efficient for training long sequences, as they require n sequential operations to learn a sequence, n being the length of the given sequence. This makes the training process extremely slow.

2. RNNs suffer from the problem of *vanishing gradients* i.e. as elements in the input need to be processed in a sequence over time, the gradients used by the optimiser[7] for the elements at

the end of very long sequences, become extremely small and numerically unstable to converge to the desired value.

3. RNNs struggle to learn *long-term dependencies*, i.e. learning relations between elements far apart in the sequence is challenging.

We present an excellent summary of the challenges and differences between several deep learning architectures which have attempted to solve the sequence modelling problem in Table 2.1.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Transformer(Self-Attention)	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent NN	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional NN	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k n)$
Restricted(Self-Attention)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Table 2.1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighbourhood in restricted self-attention. SOURCE: Original paper[8]

Augmenting RNNs with attention solves some of these issues[2], and replacing them with *Transfomers* has shown to solve all these problems. The authors propose an architecture for translation tasks that contains:

- a learnable *embedding* layer mapping words to vectors
- a *transformer encoder* encoding the input sequence
- a *transformer decoder* generating an output sequence based on the encoded input

Each transformer block alternates between attention and linear layers, ie between encoding context and refining features. The attention mechanism helps learn ”context rich” representations of every element, mapping relevant information from the surrounding elements which surround it and the linear layers help map this learn information into a form useful for downstream prediction. Figure 2.1 shows details of the original Transformer architecture and the functions of its layers.

2.1.2 Pre-training and fine-tuning

Due to the highly efficient and parallelizable nature of Transformers, they can be widely used a variety of tasks based on the principle of *transfer learning.* The most common strategy for that is to use the architecture for two phases, *pre-training* and *fine-tuning.* Inspired by the original Transformers success on the task of language translation, the use of Transformers has become ubiquitous in solving NLP problems. We present one such state of the the art NLP Transformer model, called BERT[4], one of the most widely used transformer models today. While the original Transformer had both an encoder and decoder with attention, BERT uses only the transformer encoder followed by a small and replaceable decoder. This decoder is usually a set of linear layers and acts as a multilayer perceptron (MLP); and is usually called the ’MLP head’ in the deep learning community. The principle of transfer-learning works for BERT as follows:

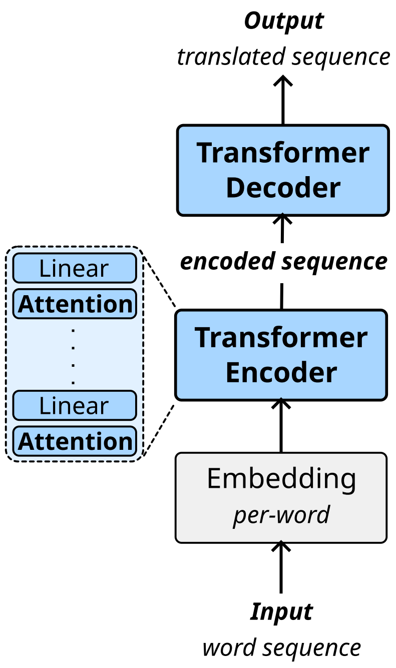

Figure 2.1: Original Transfomer Architecture, Credits: Alex, HotNets '22

- In the first step, BERT is *pre-trained* with a task that requires learning the underlying language structure. Linguists' research has shown that a Masked Language Model is optimal for deep learning models for learning structure in natural languages.[9] Concretely, a fraction of words in the input sequence is masked out (15% in the original model), and the decoder is tasked to predict the original words from the encoded input sequence. BERT is used to generate contextual encodings of, which is only possible due to the bi-directionality of the attention mechanism in BERT. This allows the model to infer the context of the word from both sides in a gicen sentence, which was not possible earlier when elements in a sequence were only processed in a given order.[1]

- In the second step, the unique pre-trained model can be fine-tuned to many different tasks by replacing the small decoder with task-specific ones, eg language understanding, question answering, or text generation. The fine-tuning process involves resumption of learning from the saved weights of the pre-training phase, but for a new task or learning in a new environment. The new model has already learned to encode a general language context and only needs to learn to extract the task-relevant information from this context.This requires far less data compared to starting from scratch and makes the pre-training process faster.

  Furthermore, BERTs pre-training step is unsupervised/self-supervised, i.e. it requires only "cheap" unlabelled data and no labelled signal from the data a target value. As procuring labelled data is harder, this problem is mitigated by having a pre-training phase on "generic" data and then using "expensive" labeled data, eg for text classification, during the fine-tuning phase. Figure 2.2 shows the details of BERT's pre-training and fine-tuning phase.

[1]BERT is pre-trained from text corpora with several billion words and fine-tuned with ∼100 thousand examples per task.

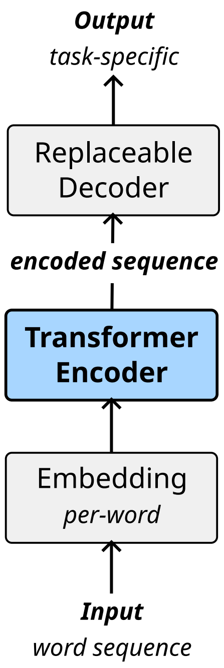

Figure 2.2: Original BERT Architecture, Credits: Alex, HotNets '22

Another extremely useful feature of Transformer is its generalization ability, which is possible due to its highly efficient parallelization during training, which in turn helps transfer of knowledge to a variety of tasks during the process of fine-tuning. The OpenAI GPT-3[3] model that investigates few-shot learning with only a pre-trained model. As it is just a pre-trained model, it does not outperform fine-tuned models on specific tasks. However, GPT-3 delivers impressive performance on various tasks, including translation, determining the sentiment of tweets, or generating completely novel text. As per requirement, it can also be fine-tuned on specific tasks, which showcase further, the generalization power of transformers.

2.1.3 Vision transformers

Following their success in NLP, the use of Transformers were explored as a possibility to learn structural information in other kinds of data. One such field where they gained a lot of traction in in the field of CV. However adapting the use of Transformers to vision tasks came with a few major challenges:

- Image data does not have a natural sequence, as they are a spatial collection of pixels.
- Learning encodings at the pixel level for a large number of images, proved to be too fine-grained to scale to big datasets.
- The Masked Language Model theory couldn't be efficiently transferred to the context of learning structure in images, as the relationship between the units (pixels) does not follow the same logic as in natural languages.

To solve this problems, the authors of the Vision Transformer(ViT)[5], came up with multiple ideas to solve each of these problems, in order to have the data in a form, on which a Transformer could be efficiently trained.

- *Serialize and aggregate the data:* As image data does not have a natural sequence structure, such a structure was artificially introduced. Every image was split at the pixel level and aggregated into patches of dimension 16×16 and each of these patches became a member of the new input “sequence”. As Transformers scale quadratically with increasing sequence size 2.1, this also solved the problem of efficiency and that encodings at the pixel level are too fine-grained to be useful. The embedding and transformer layers were then applied to the resulting sequence of patches, using an architecture similar to BERT.

- *Domain Specific Pre-training:* At the heart, most CV problems have very similar training and inference objectives, one of the most common being image classification, which makes most of the vision tasks similar in objective, differing only in environment. This made classification a much more suited task for image data pre-training as opposed to masking and reconstruction. This was exploited by the ViT and both the pre-training and fine-tuning was done with the objective of classification, with only a change in environment between the stages. This also meant that understanding structure in image data, not only required information from the neighbouring patches but also from the whole image, which was possible by using all the encoded patches for classification. We present the details of ViT’s pre-training and fine-tuning in Figure 2.3

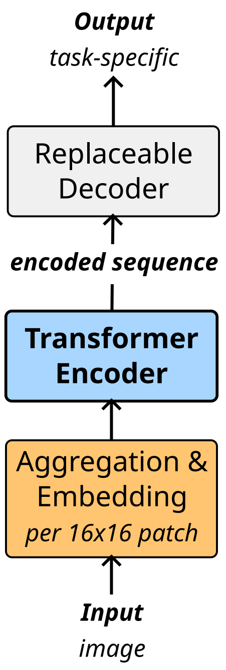

Figure 2.3: Original ViT Architecture, Credits: Alex, HotNets ’22

Finally, ViT’s authors also observe that domain-specific architectures that implicitly encode structure, like convolutional networks (CNNs) work extremely well for image datasets and actually result in better performance on small datasets.However, given enough pre-training data, learning sequences with attention beats architectures that encode structure implicitly, making the process a tradeoff between the utility and the amount of resources required for pre-training. Later research in the field has also shown advancements in vision based transformers which use a masked reconstruction approach[6], such details however are beyond the scope of this section.

2.2 Related Work

Is there much to add here? We claim that no one really tried to solve this problem as it was a ”lost” cause. However, maybe some references such as MimicNet/Aurora can be added here added here, as we did look at them to an extent.

If we refer to another paper, book, website, . . . we add a citation.

Chapter 3

Design

3.1 Dataset Generation

Note: Will decide later if we want to move this to a new chapter or not.

3.2 Network Traffic Transformer (NTT)

This chapter describes your work in detail and is normally the longest chapter. You can also create multiple chapters describing your work.

It is often useful to format your text with **bold** or *italic words*. In addition, it can be helpful to format text as verbatim:

```
Verbatim representation
    of     my
text.
```

Finally, you can also add a mathematical formula inline $a^2 + b^2 = c^2$ or in its own block:

$$\cos(2\theta) = \cos^2\theta - \sin^2\theta \tag{3.1}$$

Chapter 4

Evaluation

Another important chapter is the evaluation which should clearly describe the experiments you performed (setup, number of measurements, . . .), show the results you achieved in figures and/or tables and discuss and compare the results.

Chapter 5

Outlook

What are consequences of your work? Do you see possibilities for future work?

Chapter 6

Summary

Give a final, short summary of your work.

Bibliography

[1] The illustrated transformer. https://jalammar.github.io/illustrated-transformer/. Accessed: 2022-07-15.

[2] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate, 2014.

[3] 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. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]* (July 2020).

[4] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]* (May 2019).

[5] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv:2010.11929 [cs]* (June 2021).

[6] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked Autoencoders Are Scalable Vision Learners, Dec. 2021.

[7] Robbins, H. E. A stochastic approximation method. *Annals of Mathematical Statistics 22* (2007), 400–407.

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention Is All You Need, Dec. 2017.

[9] Wettig, A., Gao, T., and Chen, Z. Z. D. Should You Mask 15% in Masked Language Modeling? 12.