Universiteit Leiden

Master Computer Science

Few-Shot Transformers

Name:	Oliver König
Student ID:	s2423286
Date:	15/07/2021
Specialisation:	Data Science: Computer Science
1st supervisor:	Dr. J.N. van Rijn
2nd supervisor:	Prof.dr. T.H.W. Bäck

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgements

The following study in the field of applied machine learning and transformer models has been an exciting journey for the last nine months. Without the support of many different people and organizations, it would not have been possible in its final extent and form. Being extremely grateful for all the assistance, I would like to say a few words.

First of all, my supervisor, Jan N. van Rijn, has been supporting my work on an extremely high frequency while giving supportive and constructive feedback all time long. I am still stunned by how much he cares for his students, how much time he allocates, and how positive his mindset remains in difficult times. Working with him was a very pleasing experience, and I am looking forward to collaborating with him in future times again. Furthermore, the feedback of my second supervisor, Thomas Bäck, has been precious as to keep the report and structure streamlined, by putting his long-term and rich experience of science and academia into play.

Second, I would like to thank the ICT Shared Service Centre (ISSC) of Leiden University for their consistent and long-term maintenance of a large-scale cluster of compute machines, for running my experiments on GPU machines. Having been in close contact, I am grateful for their kindness and support in keeping all machines alive for such a long period. Besides the ISSC, I have been using the ALICE High-Performance Computing facility for more heavy compute tasks such as MiniImageNet or larger architectures of my model. Without these machines, larger experiments would not have been possible. I would also like to thank the service of Weights Biases ([Bie20]) for their amazing service of monitoring and tuning Machine Learning models in an extremely easy and fast manner. Countless hours of their service have been used, and countless hours of work have been saved due to their stunning API. Strong recommendations for similar use cases are given out to the fellow reader.

Furthermore, my family and friends have given me the support I needed to keep up such a long project, especially during the difficulties we experienced in the last 18 months of the COVID-19 pandemic. Many calls, and a few in-person meetings, have given me the help to keep up a positive spirit. With the end of this project and some light at the end of the tunnel of the pandemic, we are all starting to gaining more positive energy and optimism towards the future.

Abstract

Since the rise of transformer models and the breakthroughs of the attention mechanism, many researchers have been putting work on understanding and improving those mechanics. While transformer models have been developed in the domain of natural language processing (NLP), there is now a trend visible of modifying them for other data modalities such as image or audio data. However, most of these advancements have rather modified the way the data is fed into the attention mechanism than changing the mechanism itself. This has led to specialized transformer models applicable for single data modalities only. In this work, we focus on the attention mechanism itself to make it more applicable to different types of data modalities.

To test our model on different data modalities, we utilize the domain of few-shot learning, a subbranch of meta-learning. Besides being inherently data modality agnostic, we motivate this choice by the close relationship between NLP and few-shot learning. It has been recognized that not only few-shot learning but also tasks from NLP can be considered as a sort of task-based learning. From this perspective, we wonder how effective current transformers perform on such tasks but with different data modalities and how such transformers can be improved in terms of classification performance.

We thereby propose three novelties around the self-attention mechanism: Opposed to a feature-wise transformation of the input batch, we propose to stack a sequence of tasks and to transform them all at once. We propose to replace linear with non-linear transformations to increase the complexity and to cope with a larger variance of the input data. Finally, we propose to generate attention weights directly by an MLP as opposed to the scaled dot-product attention.

Besides the main research around Few-Shot Transformers, the novel type of transformer we are proposing, we also introduce a dataset that has been created in the early phase of this study. This dataset is very suitable for the few-shot setting and can be sampled infinitely large. Both in the experimental chapter, as well as in the appendix we elaborate on the details.

Contents

List of Tables

List of Figures

1. Introduction

Since the arrival of gated neural networks, such as recurrent neural networks, the idea of logical data flow within gradient-based estimators has been evolved, namely in the form of neural-turing-machines [GWD14] and lately in the form of transformers [VSP+17]. The latter one specifically implements a form of information routing, called self-attention, which allows a transformation on word-vectors, or tokens, from the setting of natural-language-processing (NLP). By self-attention, tokens are routed and exchanged between sequences to learn which parts of a sentence to pay the most attention to. While those models arose in the field of NLP, successors have been developed to directly work with different data modalities such as images by vision transformers [DBK+20]. The original self-attention mechanism is thereby used in a way such that those models can even compete with traditional convolution-based models. While the self-attention mechanism is kept in the original implementation, the way of feeding data in and out of it is specifically designed per application, such as image or text data. While this approach has been successful in an attempt to replace classical convolutional neural networks, they suffer from a low sample-efficiency leading to a new of even larger datasets [DBK+20]. Besides advancements in terms of different data type applicability, other studies have been focusing on more computationally efficient attention mechanisms by approximating the attention matrix. This usually trades performance and efficiency.

The domain of NLP, from which transformer models arose, is from a high-level perspective comparable to the field of few-shot learning. Here, based on a set of tasks presented to a model at once a novel task has to be solved. Specifically, we call the set of tasks the support samples and the task to be solved the query sample. While in NLP the tasks correspond to embeddings of word vectors, they can be of any general form in the context of few-shot learning. Few-shot learning is a subbranch of the field of meta-learning. Here, estimators are required to learn abstract rules of learning, which in many cases boils down to learning how to compare samples in the input space. Since transformer models have also been described as few-shot learners, we are keen to explore further opportunities and challenges in this direction. Given that few-shot learning settings are in general data modality agnostic, meaning that tasks are build from images, audio signal data, or all other kinds of tabular data, we are interested in a model which reads data in the same shape and format and regardless of the data type being used. This is quite opposite to recent advancements like vision transformers [DBK+20], which specifically adapts original transformers to work on image data, but at the same time on image data only.

We are motivating our research by a modified transformer model which performs on all kinds of data modalities and the few-shot setting better than original transformers, in terms of classification accuracy on a validation set. For this, we are interested in highlighting the attention mechanism itself and to explore opportunities for proposing a

more general and complex attention mechanism, capable of solving a multitude of different data modalities. In the underlying work, we concern ourselves with three major questions, all centred around the development of self-attention in the context of transformer models. First of all, we are interested in the performance of original transformer models in the few-shot setting. Therefore, we select both standard datasets and propose a novel one by ourselves. Besides the transformer model, we also compare ourselves against a state-of-the-art algorithm in the field of few-shot learning. Further, we discuss modifications to the self-attention mechanism for increasing the computational complexity and to achieve a larger field of application w.r.t different datasets. Finally, we empirically test our novel model on the proposed datasets. We summarize our three research questions as follows:

RQ1: What is the performance of original transformers on non-NLP tasks?

RQ2: What modifications increase the performance of transformers on non-NLP tasks?

RQ3: How competitive are those modifications on non-NLP tasks?

Remaining work In the following background chapter, we elaborate on the historical development which led to the popular self-attention mechanism of transformer models. We discuss different types of the alignment model used in attention, before introducing the field of meta- and few-shot learning. Most importantly, we describe the data format being used in this setting and for all models in the later following experiments. Following, we discuss modifications to the attention mechanism implemented by our proposed Few-Shot Transformer in the methods chapter. In the experiments chapter, we explain the experimental setup and procedure. The results are presented afterwards within the results chapter. We finalise our study with a discussion and conclusion of the experimental findings in the last chapter.

2. Background

2.1. Gated Neural Cells

Since the developments around neural long short-term memory (LSTM) [HS97] cells, ways of building neural layers in the perspective of data processing units have been sought. By combining logic gates such as *OR*, *AND*, or *NAND*, information routing within these blocks has been constrained in the desired fashion. Figure 2.1 depicts such a long short-term memory cell, with its parts of an input, output, and forget gate. Here, the input

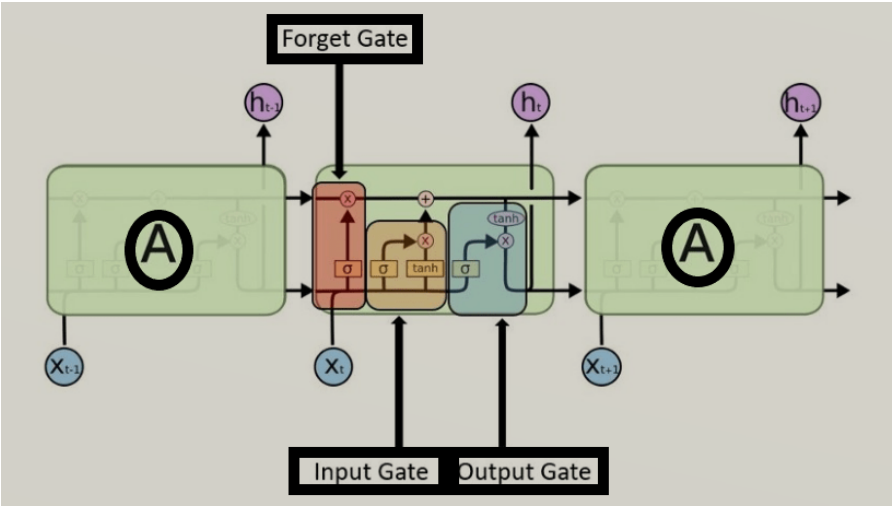

Figure 2.1.: Idea behind LSTM, figure by Tondak (2021) [Ton21]: Information flow is controlled by logical gates of *AND* and *OR*. This is achieved by an input, forget, and output gate, respectively.

gate learns which input information to accept, the forget gate controls which information *not* to accept, and the output gate toggles which information to propagate to the next layer. While LSTMs enable sequence-processing, giving rise to many applications such as NLP or audio-processing, some issues remained unsolved and others were introduced by their architectures:

1. While LSTMs do allow the modelling of long-term dependencies and attending to short-term patterns in general, their weaknesses have been spotted in terms of long-long-term dependency identification. While for smaller sequences LSTMs turned out to be quite powerful, their performance crucially drops with the increasing length of the sequence.

2. They suffer from their recurrent design which hinders effective backpropagation and parallelization during training time, as found by Pascanu et al. (2013) [PMB13]. Since the sequence needs to be unrolled over time, only a single worker can process this batch and thus distributed training becomes much harder.

These issues motivated further research since both are particularly hindering large-scale and complex real-world applications, i.e in the processing of full books rather than short sequences from social media. Such applications not only require learning long-term dependencies but also an efficient training mechanism to compute such long sequences.

Yet, the idea of building neural data pipelines which decide when to store, when to retrieve and how to use this information for processing the input has been followed by many different scientists. One foundation for the remaining work is the neural turing machine (NTM) proposed by [GWD14]. The network, consisting of a memory unit and a controller as depicted in Figure 2.2, can learn general operations and functions like copying, repeatedly pasting, or attending to associative contents. It follows the idea of gated

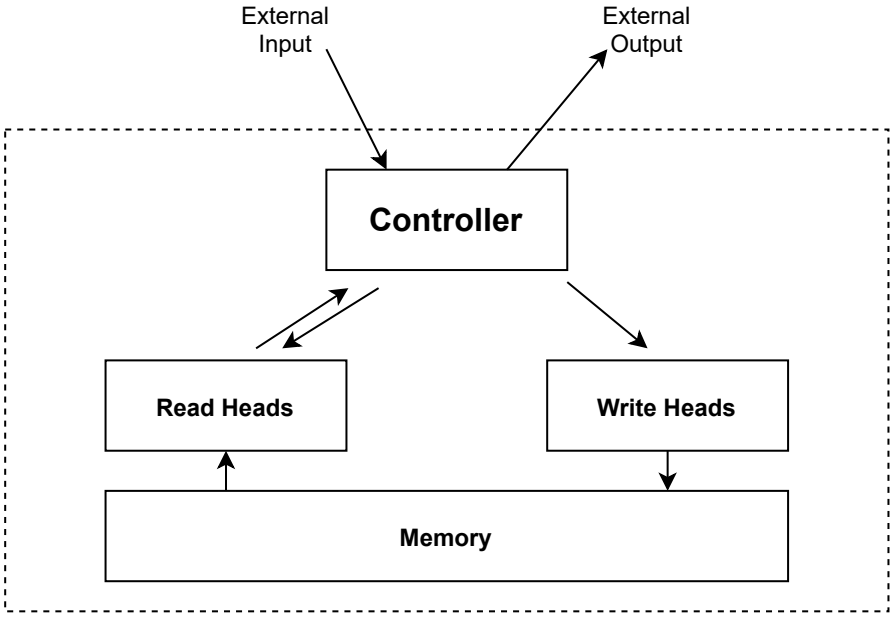

Figure 2.2.: Idea behind Neural-Turing-Machines, figure inspired by [GWD14]: A recurrent neural controller, often a LSTM unit, accesses a memory vector by read or write operations to store features.

data pipelines but extends LSTMs by a second and dedicated controller deciding which information to put into the memory and how to use it after retrieving it. While NTMs empirically work, they are still notoriously difficult to train as shown in the original study. Thus, as an observation of the authors, NTMs failed to become mainstream. However, a continuation of the idea has finalized itself on the concept of *attending* to right pieces of information in the input, and to transform the input repetitively by this operation as

long as it has been successfully transformed. Figure 2.2 can be seen as an early attempt of an attention mechanism since the controller unit decides which information to read from and write to the memory for inferring new data.

[VSP+17] proposed such an attentive and transforming model by combining scaled pot-product and feedforward networks in a simultaneous encoder-decoder fashion, naming it the transformer model. While in this scenario we do not have gates anymore, we argue these gates are learned by the linear, attention projections inside of each attention layer. From the high-level perspective, attention works in the following way: Information is routed by retrieving certain *keys* by a set of *queries* and projecting this onto a set of *values*, the mechanism itself can also be described as the gated decision of which information, or on the tongue of NLP, tokens should be exchanged in the vectors of data. This is comparable to a controller determining which information to read from or to write into a head. However, instead of having a dedicated LSTM-based controller unit, attention works by the input data itself and by finding transformations of the input as *queries*, *keys*, and *values* which then by some alignment determine which information to keep, exchange, or discard. Thus, each attention layer learns to pay attention to different parts of the input space and thereby transforms this space such that the deeper layers become more effective in their task. Remarkable breakthroughs have been achieved by [DCLT18, RWC+19, BMR+20]. We will cover attention in more detail in the following section.

2.2. Generalized attention

Generalized attention introduced the idea of handling a query vector $\mathbf{q}$, for which a most attentive piece of information needs to be retrieved and injected into some data as to transform $\mathbf{q}$. For this, we imagine a set of key-value pairs ($\mathbf{K}$, $\mathbf{V}$) which we regard as the internal memory of the attention layer. Given an alignment model e, we normalize the attention score into a similarity vector over ($\mathbf{K}$, $\mathbf{V}$) given $\mathbf{q}$ in the range $[0, 1]$. By multiplying this into $\mathbf{V}$, we access the i-th row of $\mathbf{V}$ and thus retrieve the attentive piece of information we were looking for. Put differently, generalized attention is a mechanism to compute a score given a pre-defined metric over a query and a set of key-value pairs. Intuitively, it allows the model to retrieve a piece of stored information, by accessing a memory slot close to $\mathbf{K}$ based on $\mathbf{q}$. The closeness is defined by a score, normalized into a probability distribution by the Softmax function and thus resembles a sort of similarity metric between samples in the input-sequence space. The piece of information restored equals the memory value $\mathbf{V}$. This is formally described in Equation 2.1.

$$A(q, K, V) = \sum_{i}^{N} \frac{\exp(e_{qk_i})}{\sum_{j}^{M} \exp(e_{qk_j})} v_i \tag{2.1}$$

Various forms of the alignment model Given an alignment model, a set of vector scores $e_1, \ldots, e_n$ are computed which then are normalized into a probability density distribution over E. The alignment scores $e_1, \ldots, e_n \in \mathbb{R}^d$ are computed over a query vector $q \in \mathbb{R}^{d_2}$

and a set of key vectors $k_1, \ldots, k_n \in \mathbb{R}^{d_1}$. In general, there are three popular ways of computing e_i:

- Additive attention: $e_i = W_3^T \tanh(W_1 k_i + W_2 q)$ (2.2)
- Multiplicative attention: $e_i = q^T W k_i$ (2.3)
- Basic dot-product attention: $e_i = q^T k_i$ (2.4)

In the case of additive attention $W_{1,2,3}$ are defined by $W_1 \in \mathbb{R}^{d_3 x d_1}$, $W_2 \in \mathbb{R}^{d_3 x d_2}$, and $W_3 \in \mathbb{R}^{d_3}$. While additive attention equals basic dot-product attention in theoretical complexity, the latter can practically be implemented much faster by matrix multiplication code as shown by [VSP+17].

2.3. Self-attention

Self-attention is the specialized implementation that regards the input as three distinct and linear projections upon which an alignment model is computed. Instead of handling a query **q** and a set of pairs (**K, V**), self-attention transforms the input into three distinct vectors **Q**, **K**, and **V**. This allows the model to find high-dimensional representations of the input allowing to find desired similarities and thus letting information flow to the values. Since self-attention uses the dot-product alignment model on high dimensional feature vectors, alignment scores are prone to grow to extremely large values causing instability during training. This effect of exploding gradients can be mitigated by scaling the dot-product by the square root of the feature vector dimensionality. The resulting alignment model is the scaled dot-product attention. We observe the following notation defined in Equation 2.5:

$$A(Q, K, V) = \text{SoftMax}(\frac{QK^T}{\sqrt{n}})V \tag{2.5}$$

In Figure 2.3 the self-attention mechanism is illustrated. An input tensor is transformed three times by linear projections into **Q**, **K**, and **V**. Performing the dot-product on queries and keys results in the attention matrix, or attention score, which is then projected onto the values. Put different, by differentiation and backpropagation three distinct transformations θ, ϕ, g are sought which align two copies of the input in such as way that the dot product between both reveals the most important pieces of information to attend to. By transforming this similarity matrix, or *attention score*, into a probability density by the $Softmax$ function, a lookup table is generated which allows copying bits of information, otherwise also called tokes, from one part of the vector **V** to a different part. This is achieved by the dot-product of the attention score to **V**.

Another novelty of the same authors is the proposal of multi-head attention, a mechanism that also has become more or less standard in the context of Transformer models. Here, the projected input the form of **Q**, **K**, and **V** are split into sub-sequences of the same length over which the attention model is computed. The authors argue that such a

mechanism allows each attention head to focus on different sub-spaces, thus increasing the sample-efficiency while keeping the computational complexity roughly the same. The final result is concatenated to a vector of the original size and send to the next layer.

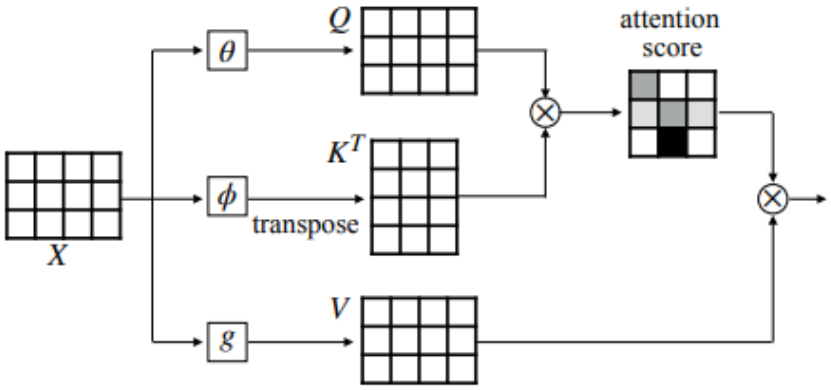

Figure 2.3.: Idea behind Self-Attention, figure by [Sin20]: Input data is transformed by three distinct linear transformations which align **Q** and **K** in a way such that optimal correlations between input rows are found. These correlations, or attention weights, are applied onto **V**.

2.4. Meta- and Few-Shot Learning

Meta-learning is a field is concerned with enabling learning models to learn *how* to learn a given task or set of tasks. While many studies focus on automatic and differentiable models such as neural networks, others are model-free as well. The problem space is broadly discretized into *model-based*, *metric-based*, and *optimization-based* approaches. In the latter case, typical implementations use a two-model setup where one base-model focuses on a particular task per episode, whereas a second model learns how to optimize such a type of base-model over a set of distinct tasks. One of the earliest and famous approaches, to the best of the authors' knowledge, is Learning to learn by Gradient descent by Gradient descent by [ADG+16]. In this early approach, the authors show how an LSTM can replace a rule-based optimizer such as adam [KB14] by a learned model in general tasks like image classification. Further developments partially adapted the idea in the form of LSTM-based meta learners by [RL16] and in novel directions as in [FAL17]. These models were specifically designed to learn how to distinguish and discriminate tasks in the set of few-shot Learning, by training a meta-network to learn how to learn these discriminating boundaries. In the second branch of meta-learning, model-based approaches are pursued. Here, often a memory-controller architecture steers the process of information storing and retrieval as to learn which generalized features need to be stored for the task of processing novel features of a close-by distribution. In particular, [SBB+16] proposed an NTM-like neural network implementing such a memory-controller

architecture. Networks of this type can store general types of features during their training time, which are recalled during inference enabling to recognize specific parts of information and consequently retrieving information from the memory to transform the input signal in a discriminating way. Finally, there is the branch of metric-based approaches in which a particular similarity-metric over the input or sequence space is learned over a set of episodes. Popular discoveries here are [VBL+16, SYZ+18, SSZ17]. While many approaches focus on model parameters, others concern with hyperparameter optimization. Well-known studies are [VRH18, vRPT+18]. For a deep survey into the broadness of this field, we highly recommend a study of Huismann, van Rijn and Plaat from 2020 [HvRP21]. Finally, since the field of meta-learning has gotten raised attention during the last couple of years, many interesting ways of developing new ideas and frameworks have been proposed. We would briefly like to mention one of these challenges [met], which contributed by raising attention to the field of meta-learning by hosting competitive challenges around meta-learning datasets and algorithms.

Few-Shot Learning is the specialisation of meta-learning which aims at learning under a high constraint of information density. Whereas in academia, datasets are often well-prepared to benchmark a particular set of tasks, industrial datasets often come in the form of sparsity w.r.t to either features or labels. Examples of such sparsity are either a small dataset size, underrepresented class samples, or a lack of labels at all. Few-Shot Learning tackles these issues by models which inherently learn to cope with such constrained environment during training time. We impose such constraints by shifting the training process from pairs of samples and labels (S, L) to tensors of support and query samples. Given a set of these *labelled* support samples, the learner's task is to determine the label(s) of the unlabelled query set. While different types of approaches can be used, many contributions find themselves classified as metric-based approaches. Whereas in supervised learning, we are handling pairs of labelled examples for which we like to approximate a discriminating function, the data model in few-shot learning is slightly more complex. Since in most instances tasks shall be separated from each other, the tensor model is expanded from $\mathbb{R}^2$ to $\mathbb{R}^3$. The first dimension yields a batch of sequences, the second dimension holds a sequence of tasks, and the last dimension describes the features of a task. Figure 2.4 depicts such a data tensor. The number of tasks inside a sequence is determined by the exact few-shot setting: N-Way K-Shot Learning defines the training of N distinct classes populated by each K instances. Therefore, a single batch consists of $N \cdot K$ labelled examples, and the $N \cdot K + 1$-th example without a label. An important constraint here is that the unlabelled example belongs to *one* of the classes of tasks shown in the same sequence. For this reason, many approaches [SYZ+18] aim at defining or learning a similarity metric over the sequence-space for finding the closest distance between two tasks given by their same class belonging.

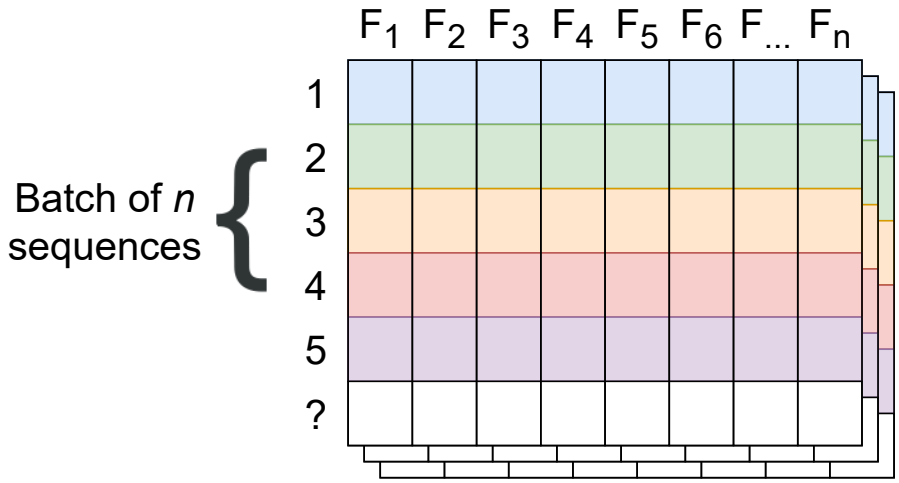

Figure 2.4.: Tensor shape of a typical Few-Shot Learning: Tasks $\mathbf{T}$ of vector length N are aranged in a sequence $\mathbf{S}$. Sequences are stacked in batches $\mathbf{B}$.

2.5. Attention in Few-Shot Learning

Attention rose from the domain of NLP, a type of problem where the data model is arranged as a tensor of batched sequences of partly labelled words. Such a problem is on a high-level perspective a meta-learning challenge, where words are called tasks. Here, the model is not learning a functional dependency between a sample $s_i \in \mathbb{S}$ and its corresponding label $l_i \in \mathbb{L}$ but an implicit model-based and metric-like function $\mathcal{F}$ which compares tasks within a single sequence. Such a problem definition seems intuitively close to the approaches of attention, as the dot-product alignment between two vectors v_1 and v_2 can geometrically be interpreted as the similarity of those in their high-dimensional space. Further, as attention scales these probits into a probability distribution one can reason about the task similarity within a sequence.

2.6. Related Work

Since the breakthrough of transformer models on NLP tasks, many research activities have been laid around improving different characteristics of both transformer models, the attention mechanism, or the combination of both. Without claim of completeness, a literature review has shown us three major branches in which academia has funnelled their activities: *Performance*, *scalability*, and *efficiency*. In performance, the main objective is to solve specific tasks or sets of tasks better than previous models. Here, the efficiency of the model is of secondary interest or sometimes even out of interest. In the objective of efficiency, the objectives are turned around compared to performance-based optimization. Scalability concerns around techniques of making models which parameter counts up to billions or even trillions stable during training, as well as improving the training process

in terms of efficiency.

Scalability: In the latter branch of scalability, approaches are sought for increasing the model complexity to large-scale magnitudes of up to trillions of parameters. These models have shown surprising effects as to generalize to tasks for which the models never were meant to be used for, as shown with GPT-3 being able to solve numerical tasks [BMR+20]. Another model being able to scale up to massive numbers of parameters are the switch transformers by Google Brain [FZS21]. By using sparsity and hard routing they are not only able to increase the parameters but also able to keep speed and sample-efficiency within reasonable limits.

Efficiency: Transformer models are known to suffer efficiency in terms of computational effort since the scaled dot-product attention scales quadratically with the length of the input feature vector sizes. Many studies have been centred around reducing the complexity to linear bounds, as with linformers [WLK+20]. The authors recognize that in many instances the information inside the attention matrix is of lower rank, and thus can be approximated. The approximation can be computed in linear scale by keeping an almost comparable performance to traditional transformer models. Based on fast weight memory systems, another paper proposes new update rules and kernels for computing attention weights with the aim of not only linearizing but also avoiding problems other approaches suffered, such as the usage of random features [SIS21].

Performance: In terms of performance, the study [FLG+20] suggests a feedback memory to stream information between higher and lower layers bidirectionally, thus decreasing the loss of computational capabilities of the overall model. While they suffer training speed, they achieve remarkable improvements in long-term dependency tasks. Other studies are intersecting closer between efficiency and performance, as with the perceiver model [JGB+21]. They both solve the quadratic bottleneck problem, as well as being agnostic on the modality of the input data. The latter is achieved by feeding the input data by cross-attention multiple times. The last paper we would like to highlight is big bird: transformers for longer sequences [ZGD+20]. Here, the authors address the quadratic compute issue by a combination of random, window, and global attention, thus allowing to avoid a full attention mask over the complete input space at once. This allows the processing of longer sequences with state-of-the-art results.

Although these models are highly interesting and clear advancements in the development of transformer models and attention, they do not resemble applicable competitors to our setting since they were not specifically designed for data modality agnostic and few-shot learning settings. Comparable studies are [LLO+20, LHL+20, JKZF20, MRCA17]. The first set of authors from [LLO+20] proposes to append a task embedding vector that represents information about the task of interest to the input sequence of token embeddings. Besides, they utilize a default transformer model with self-attention. Although their methods produced competitive results, the experiments are unfortunately only conducted on artificial tasks in the domains of sequence classification, sequence transduction, and

pathfinding, making them difficult to compare to different tasks such as computer vision. The second set of authors from [LHL+20] proposes a novel transformer layer, called the Universal Representation Transformer layer which leverages universal features for few-shot classification by dynamically re-weighting and composing the most appropriate domain-specific representations. In comparison to our ambitions, this study does not change the self-attention mechanism itself, but rather the exchange of features before and after the attention transform. Finally, the study of [JKZF20] is very interesting for the idea of directly generating attention weights by an MLP, as a means to replace self-attention. The authors propose an attention module that, specifically for image data, computes attention weights based on a meta-weight generator. Given the domain of image data, the meta-weight generator is implemented as a single- or two-layer point-wise non-linear projection with shared weights over the $\mathbb{R}^2$ axis. Utilizing a convolutional or fully-connected network for directly computing attention weights, opposed to the scaled dot-product attention over multiple linear feature projections opens a new way of thought of transforming data. However, since the solely works for image data we choose not to compare it directly with our method. Still, we recommend the study to the fellow reader.

Most similar to our ambitions is SNAIL by [MRCA17], which combines self-attention with different mechanisms such as wavenet-connection for solving data modality agnostic tasks in the few-shot setting. While attention allows attending to all previous tasks, wavenet restricts this to one-way dilated connections forcing the model to learn larger generalizations between tasks. The authors test their method not only on *N-way K-Shot* few-shot settings but also on reinforcement learning problems. For this reason the general applicability of SNAIL, we choose the algorithm as a competitor to our model.

2.7. Motivation for the remainder

Based on the success of the attention mechanism, we are motivated to explore extensions and modifications to the attention mechanism. We measure the performance of these modifications by comparing them against state-of-the-art algorithms on different data modalities and datasets in the few-shot setting. While developments of transformer models for i.e computer vision have modified the way data is fed into the self-attention mechanism, the mechanism itself has often not been modified. For this reason, we are interested in studying which adaptions of the self-attention mechanism facilitate learning on data modality agnostic problems. In contrast to recent advancements such as vision transformers, the data will be fed into the model in an application-agnostic manner i.e. will not be split up into tiles by rather transformed at once. In the same fashion, all kinds of problems can directly be fed into our proposed model without the need for specific adaptions. We propose to solve this by increasing the complexity of the attention mechanism by methods explained in the next chapter.

3. Methods

We motivate our research by the intuitive closeness of the attention mechanism to the challenges of few-shot learning. On the shoulders of proposals of [VSP+17, MRCA17, JKZF20] we propose a novel Few-Shot Transformer which aims to be applicable in multiple fields rather than only NLP or computer vision based models, by providing comparable performance. Similar to [JKZF20] we propose a parameter-based approach of computing attention weights, but also to transform feature vectors. Since we kept the feed-forward networks as proposed by [VSP+17], we will discuss our modifications of the attention mechanism.

3.1. Model embedding

In order to keep the computational effort per layer constant, the initial tasks or their embedding $t \in \mathcal{T}$ are fed through a linear transformation to produce a set of feature vectors $f \in \mathcal{F}$ of dimensionality d_{model}. We follow [VSP+17] on this idea and recommend it for further use for many reasons such as hyperparameter choice reduction as well as simplified computational effort prediction. Further, we use an application-specific encoder, like a resnet for image data, to create an embedding of the initial tasks, if necessary. This follows the procedure of similar algorithms such as [MRCA17, SYZ+18].

3.2. Encoders-Decoder Stack

Encoder The encoder-decoder architecture depicted in Figure 3.1 allows, similar to transformers, multiple successive encoding steps. Each encoder module performs two operations, each finalized by a skip connection and layer normalization for a smoother gradient flow. The first operation is the *Deep Attention*, a novelty proposed by our findings. This operation allows information routing in the batches of sequenced tasks as to flow data between the support and query samples. Opposed to transformers, the alignment score is computed non-linearly and by a parametric approach. Specifically, a two-layer, position-wise fully-connected and non-linear transformation is used, which outputs a set of attention weights. These attention weights are applied on a non-linear transform of the values to produce the transformed feature vector. We call this step the Deep Attention transform, given the computation of attention weights by a deep MLP. The second step feeds the transformed sequence, similar to the original proposal of [VSP+17], through a position-wise fully-connected feed-forward network. The input to the encoder module corresponds to the typical few-shot learning batch depicted in Figure

2.4, where tasks are arranged in a sequence. Multiple stacked sequences resemble a batch on which gradient descent is performed on the feed-forward and backpropagation step.

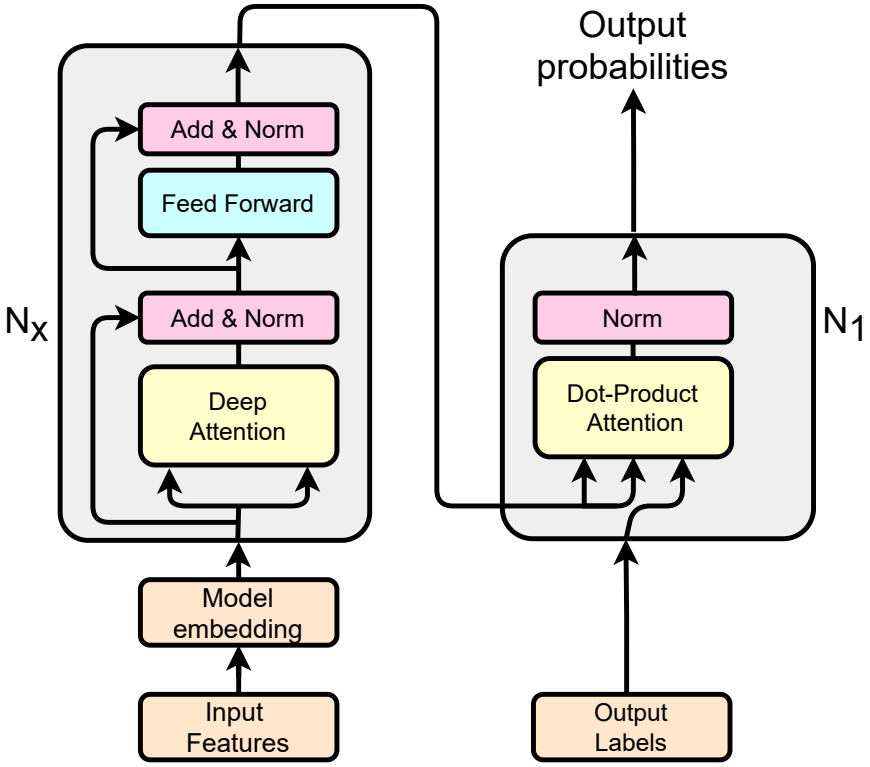

Figure 3.1.: Architecture of the Few-Shot Transformer. Multiple Encoder-Modules are stacked, from which each transforms the input of stacked sequences of tasks by transferring tokens based on Deep Attention. The architecture is concluded by a single Decoder module. An attention score is computed, which intuitively expresses the similarity among tasks. Projecting this score onto the labels broadcasts them in a way such that the $N \cdot K + 1$ example will yield the true label.

Decoder Whereas transformers decode multiple times, usually as often as they encode. For the setting of few-shot learning these computations do not yield any benefit since the decoder's input simply holds class labels for each of the sequence steps of a batch. As a result, the proposed architecture consists of a single decoder, which expects the feature vectors to be aligned in their high-dimensional space accordingly to their class affiliation. Put different, the transformed features shall be placed in a latent space such they are grouped closely together according to their class label. While this grouping is achieved by the transformation of the encoder modules, the decoder module exploits the distance between similar and non-similar tasks by the final step of the scaled dot-product attention.

Here, an attention matrix is computed which then corresponds to the similarity of tasks w.r.t their class label. Projecting this similarity matrix onto the labels broadcasts the label of the most similar support sample to the query sample onto the label position of the query sample.

3.3. Deep Attention

Self-attention locates local and global dependencies in the sequence-space by minimizing the error on three distinct linear projects on the input, which after successive alignment highlight sub-spaces to attend to. In the specialized domains such as NLP or computer vision, where the distribution of the feature space is strictly controlled by constraining functions such as vector embeddings or convolutions, these linear projections of the feature space have empirically proven to be powerful and effective. However, in the case of meta- and few-shot learning no such assumptions can be drawn around the input space since the sequences here are examples of features that are not bound to specific domains such as audio processing or signal analysis. For this reason, we argue that a more complex transformation on the features is sought. In this sense, we are proposing three novelties:

1. Attending to the full sequence: As original transformers employ self-attention, they project the input sequence linearly and feature-wise into a set of queries **Q, K**, and **V**. By feature-wise, we are referring to the stacked batch of sequences shown in Figure 2.4 in which each row resembles one feature vector being projected linearly and independently of all other feature vectors in the same sequence. Since various kinds of problems are expected to be fed into the learner, no assumptions about data distribution and closeness of features within a sequence or batches are given. Thus, we argue that an artificial increase in data density is needed. Instead of performing feature-wise projections, our first contribution proposes to perform sequence-wise projections instead. Rather than attending to dimensions of a single feature, we attend to all positions within a sequence of features, and thereby not only transform the dimensions of a single feature but instead to all features of a sequence at once. This enables the model to perceive a larger field over the few-shot learning task and increases the variance of the data input. Put different, instead of transforming the support and query samples one by one, all samples are stacked together and presented at once to the Few-Shot Transformer. Hence, instead of transforming each row separately of the tensor presented in Figure 2.4, all examples of the sequence are considered as a single feature and as a whole projected.

2. Non-linear projection: Following on the idea that the data density has to be increased for enabling the information storage, retrieval and transformation process, we concurrently argue that the **Q, K**, and **V** projector has to be of higher complexity as well. While the original implementation proposed linear and position-wise transformations, we propose a two-layer position-wise feed-forward network with a non-linearity in-between. This allows the approximation of a non-linear dependency between a sequence and the features we

need to pay the most attention to. We define a base-network $\mathcal{G}$ in the form of equation 3.1:

$$\mathcal{G}_n : max(0, xW_1 + b_1)W_2 \cdot b_2 \tag{3.1}$$

where $W_1 \in \mathbb{R}^{d_1 x d_2}$, $W_2 \in \mathbb{R}^{d_2 x d_n}$ for an output dimensionality of n. Hence, this type of network $\mathcal{G}$ can be instantiated onto a network $\mathcal{G}$ by defining the output dimensionality n. Such a base function helps us reuse a certain type of architecture multiple times in our model. Another way of looking at it is a two-layer MLP without a yet defined output shape. n defines the output shape, and a specific network can be instantiated from $\mathcal{G}$ by passing a shape n into it. We highlight that this non-linear projection can both be used with self-attention, as well as with our proposal of directly computing the attention weights. In order to transform feature vectors similar to self-attention, but in a non-linear way, an instantiation of $\mathcal{G}$ in the form of equation 3.2 is proposed:

$$G^1(x) = \mathcal{G}_{|S|x|N|}(x) \tag{3.2}$$

3. Attention weights projection: Since attending to a whole sequence at once scales the computational effort of the first layer of a network of type $\mathcal{G}$ by a factor of $s = |S|$, we are discussing a method to mitigate this impact. While original transformers compute the alignment model e based on the two projected vectors $\mathbf{Q}$ and $\mathbf{K}$ by self-attention we propose, similar to [JKZF20], to utilize the type of network $\mathcal{G}$ from equation 3.1 as to directly compute attention weights through an instantiated network G^2. By this mechanism we replace the alignment model e of self-attention with a non-linear, parametric approach to directly generate attention weights. We soft-max these weights accordingly to equation 2.5 as to receive a similarity mask in the form of probability densities from features to features. We describe G^2 formally as in equation 3.3:

$$G^2(x) = \mathcal{G}_{|S|x|S|}(x) \tag{3.3}$$

By concurrently scaling up the data density and model complexity, we allow more complex analysis between features w.r.t their closeness and similarity. Further, by applying these attention weights on stacked and non-linearly transformed values we allow the model to perform aggregations of features with the target of merging features of the same class while moving non-similar features further away. The architecture of both weight projectors is shown in Figure 3.2. We denote our Deep Attention mechanism the following:

$$A(X) = \text{SoftMax}(G^2(x))G^1(x) \tag{3.4}$$

Finally, we summarize the key differences of our Deep Attention mechanism to self-attention as the three novelties discussed:

1. We compute attention weights directly by a parametric approach as opposed to the scaled dot-product mechanism commonly used.

2. By attending to the full sequence at once, we increase the signal variance read by the projection. The stacking of features is applied two times: One time to generate the attention weights, and a second time to compute $\mathbf{V}$.

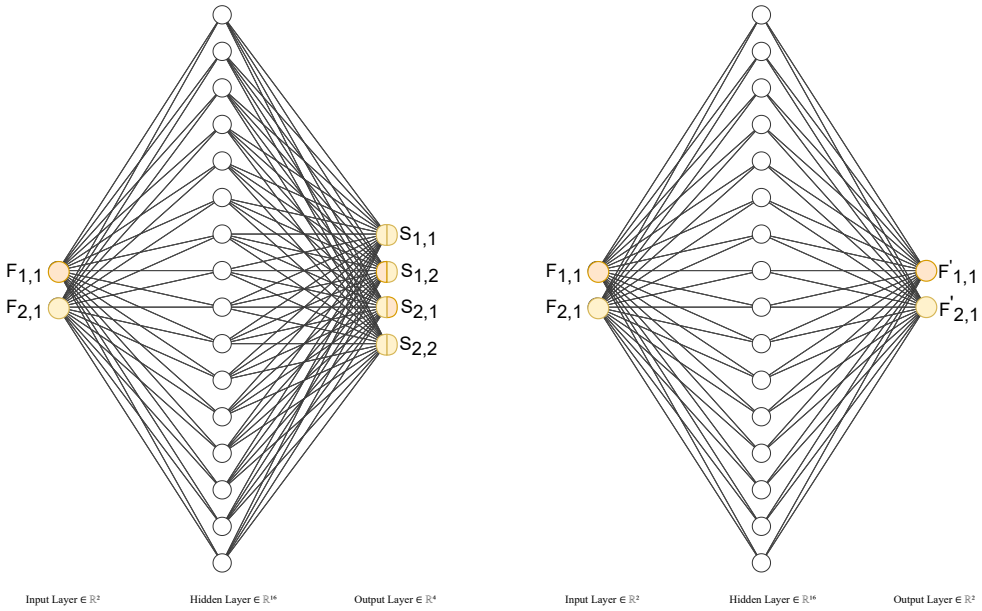

Figure 3.2.: Examplic depictions of $\mathcal{G}_{|S|x|S|}(x)$ and $\mathcal{G}_{|S|x|N|}(x)$ for an input X with $X \in \mathcal{R}^{(\cdot,2,1)}$. Since visualizations of larger feature vectors become quite spacious, this example only shows the underlying idea. In real-world applications, the input X will come frome a space $X \in \mathcal{R}^{(\cdot,|S|,d_{model})}$.

3. Concurrently we increase the complexity of the projection mechanism from a linear to a non-linear transformation. The non-linear projection is both used on the full sequence transform which generates the attention weights, as well as on the transformation of $\mathbf{V}$.

3.4. Model complexity

In the following we are interested in the model complexity w.r.t to layers and number of parameters. Since the proposed architecture consists of a stack of encoder layers concluded by a single step of decoding, there are at most $n_{enc} + 1$ layers. Each encoder layer consists of two operations, performed by Deep Attention and the feed-forward modules, respectively. A Deep Attention module consists of two functions, G_1 and G_2. All MLPs used inside the encoder layer have a latent dimension d_{latent} which scales by its input, hence by the product $|S| \cdot d_{model}$. Historically, we have found that many approaches scale the latent dimension by a constant of the input dimension. We denote the following equations to comput the number of parameters as a function of the number of tasks $|S|$ and the model size d_{model}:

$$
\begin{aligned}
p_{G^1}(|S|, d_{model}) &= (|S| \cdot d_{model}) \cdot d_{latent} + d_{latent} + d_{latent} \cdot |S|^2 + |S|^2 \\
&= d_{latent} \cdot ((|S| \cdot d_{model}) + 1) + |S|^2 \cdot (d_{latent} + 1)
\end{aligned} \tag{3.5}
$$

$$
\begin{aligned}
p_{G^2}(|S|, d_{model}) &= (|S| \cdot d_{model}) \cdot d_{latent} + d_{latent} + d_{latent} \cdot (|S| \cdot d_{model}) + (|S| \cdot d_{model}) \\
&= d_{latent} \cdot ((|S| \cdot d_{model}) + 1) + (|S| \cdot d_{model}) \cdot (d_{latent} + 1)
\end{aligned}
$$

Finally, we observe a much smaller number of parameters for the second operation, the feed-forward network. Since the projection is here similarly to the original implementation w.r.t to single tasks instead of the full sequence, a much smaller number of parameters will be observed:

$$
\begin{aligned}
p_{FF}(d_{model}) &= d_{model} \cdot d_{latent} + d_{latent} + d_{latent} \cdot d_{model} + d_{model} \\
&= 2 \cdot (d_{model} \cdot d_{latent}) + d_{latent} + d_{model}
\end{aligned} \tag{3.6}
$$

Since the decoder layer only performs the scaled-dot product attention on the transformed features, there are not a significant number of weights being added. Precisely, the only weights in the decoder layer are inside the layer normalization, which normalizes the probits to a standard normal distribution. We ignore these weights since their contribution is fairly small.

3.5. Model training

We train our Few-Shot transformer in a step-wise and gradient-based optimization scheme by utilizing the adam optimizer. For the learning rate, we study the effect of both a constant as well as an adaptive scheduling system, namely the noamlr from [VSP+17]. We illustrate the noamlr in Figure 3.3. The batch size has been 32 for all settings, and the constant or maximum learning rate is found by a bayesian optimization search.

Hyperparameter search We determine the set of hyperparameters for our Few-Shot Transformer by searching over the space of tunable hyperparameters. For this, we consider the options of grid search, random search, and bayesian optimization search. While grid search covers the full space precisely, it comes with the highest compute consumption. Empirical research has shown that instead of sequentially testing out all options, a random sampling process over the space of hyperparameters often outperforms in terms of compute and time [BB12]. Since we are, at least in most cases, only interested in the best model and not the overall dependency between hyperparameters and their induced performance an even more efficient approach can be taken by utilizing a meta-model. This meta-model spans a gaussian process over the space of hyperparameters to their observed performance, and by balancing exploration and exploitation it is theoretically able to converge to an

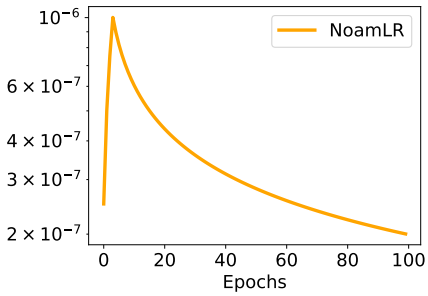

Figure 3.3.: Noamlr scheduling: The learning rate is linearly accelerated over a period of *warm_up* steps, then exponentially decayed over the remaining training time.

optimum set. This bayesian approach [FSH15] was chosen to find the best architecture per dataset. To speed this process up, we trained each model for only 50 epochs, of each 10.000 steps. While early stopping techniques were considered in the beginning, they have not been employed finally since any bias on late-performing models was not preferred by the author.

Fine-tuning Once the set of best-performing hyperparameters, or put simply the final architecture, per dataset had been discovered, we trained our models for the fine-tuning process. Here, we increased the training to in total 100 epochs, each with again 10.000 iterations. For both hyperparameter search and fine-tuning an equal amount of training and testing steps has been performed.

4. Experiments

In the following, we will introduce several experiments for internal and external analysis of our proposed Few-Shot Transformer. We will thereby start by an internal analysis of the model performance w.r.t. a pre-defined grid of hyperparameters. For computational reasons, we will limit these experiments to a single dataset. From the perspective of external analysis, we will introduce a set of datasets for a wide variety of different tasks, as well as the benchmark algorithm per dataset. Finally, a list of ablation studies will be discussed to test the behaviour of isolated contributions of our model.

4.1. In-depth analysis

For the internal analysis we are mainly interested in the performance of our model w.r.t. model dimensionality d_{model} and depth n_{layers}. Since each attention layer also implements dropout for countering overfitting, we include the dropout probability p_{dp} into the grid as well. The learning rate lr will be scheduled either by NoamLR, similar to original Transformers by [VSP+17]. The grid will be explored randomly on the Omniglot[LST15] dataset. Besides the evaluation of hyperparameter configurations to the accuracy of the

Hyperparameter					
d_{model}	16	32	64	-	
n_{layers}	1	3	6	9	
p_{dp}	0	0.1	0.3	-	
lr	1E-06	5E-06	1E-05	5E-05	1E-04

Table 4.1.: Hyperparameter grid for model exploration: d_{model}, n_layers, p_dp, and lr are defined in a discrete space.

validation set, we will investigate the best performing model w.r.t. the gradient flow. This serves the purpose to analyse whether all layers and weights are truly used for storing information, opposed to being invariant pass-through parameters.

4.2. Datasets

Since our goal is to design a Few-Shot Transformer capable to compete on a large variety of data types, we will experiment on datasets coming from various domains as well. Specifically, we will utilize image data, audio data, tabular data as well as text data. All problems will be cast as a classification problem in the N-Way K-Shot fashion. Specifically, we will be using 5-Way 1-Shot learning and thus handle 5 different tasks at a time of a single instance each. The query sample originates from one of these 5 classes but is unequal to the support samples of the same sequence. In the following, we will briefly the datasets in use:

Omniglot One of most famous few-shot image-classification datasets is the Omniglot [LST15] dataset. Yielding 1623 different handwritten characters from 50 different alphabets, it is in the light of numclasses-to-numsamples ratio often referred to as the transpose of MNIST by [LeC98].

MiniImageNet The dataset has been released as part of a novel One-Shot Learner for image classification by [VBL$^+$16]. Its complexity is high due to the use of ImageNet images but requires fewer resources and infrastructure than running on the full ImageNet dataset. In total, there are 100 classes with 600 samples of 84×84 colour images per class. These 100 classes are divided into 64, 16, and 20 classes respectively for sampling tasks for meta-training, meta-validation, and meta-test.

Deep Symbolic Regression on Differentiable Models A novel dataset which has been produced as part of this study as well. The dataset contains weights of neural networks of a certain architecture together with a label. The weights have been determined by training the network on a meta-dataset yielding a function $f(X) = p_0 + X \cdot p_1$. Thus, the label presents the tuple (p_0, p_1). Since the neural networks have been initialized randomly and since they are massively over parametrized for the meta-dataset, even for two networks n_1, n_2 with $l_1 = l_2$, we can expect their weights to be different as different solutions will be found by gradient descent. This provides a nice way to sample massive amounts of data in the $\mathcal{R}^{(nxn)}$ space, with n resembling the number of nodes of the base architecture. The dataset has not yet been released but will be elaborated on in the appendix.

4.3. Competing algorithms

In the following, we will discuss the competing algorithms. While all algorithms have been introduced in chapter 2, we will now discuss the relevant ones in more depth:

Transformer The vanilla Transformer implementation of Vaswani et al. (2017) [VSP$^+$17], see Figure 4.1, is one of the most important architectures to compare against since it yields the first attention mechanism leading to significant break-throughs in the domain

of Natural Language Processing. It implements self-attention in combination with multi-head attention by linearly projecting the input feature-wise three different times. The main difference to our Few-Shot Transformer lies thereby in:

1. The feature-wise instead of sequence-wise transformation of the input data.
2. The non-linear and MLP-based instead of a linear transformation of the queries, keys, and values.
3. The direct generation of attention weights by an MLP instead of the scaled dot-product attention.

The only deviation from the original transformers we implemented is in the usage of a single decoder module, instead of having as many encoder as decoder modules as proposed by [VSP+17]. We argue that in the case of Few-Shot Learning, where the outputs are solely vectors of binary labels multiple decoding steps are neither helpful nor necessary. Thus, after having processed the input multiple times by the encoder module the input data has been transformed sufficiently to broadcast the true label to the query position in the vector of labels, as done by the scaled dot-product attention.

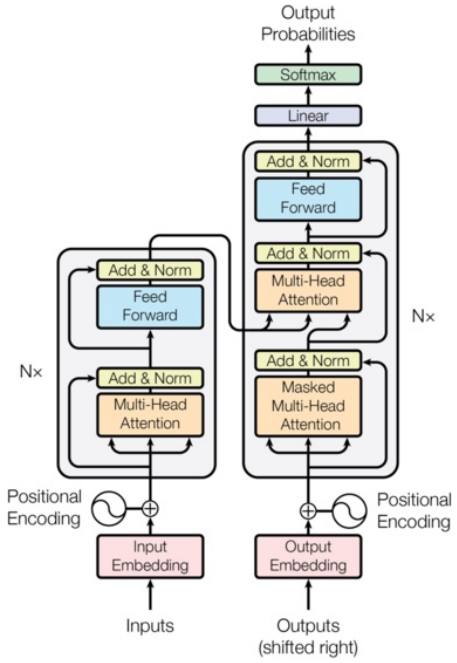

Figure 4.1.: Architecture of TRANSFORMER, from the original paper [VSP+17].

SNAIL SNAIL [MRCA17], illustrated in Figure 4.2, implements both attention and wavenet-like connections and was designed to to specifically solve few-shot problems. The authors tested SNAIL on Omniglot and MiniImageNet for image classification

problems, but also with different datasets for reinforcement learning problems. Attention is constructed in the way of self-attention but without multiple heads. We keep the architecture of SNAIL the same for all experiments, similar to how the authors kept the same architecture for all of their experiments. As opposed to the original paper of SNAIL, we use the same vision encoder for all experiments.

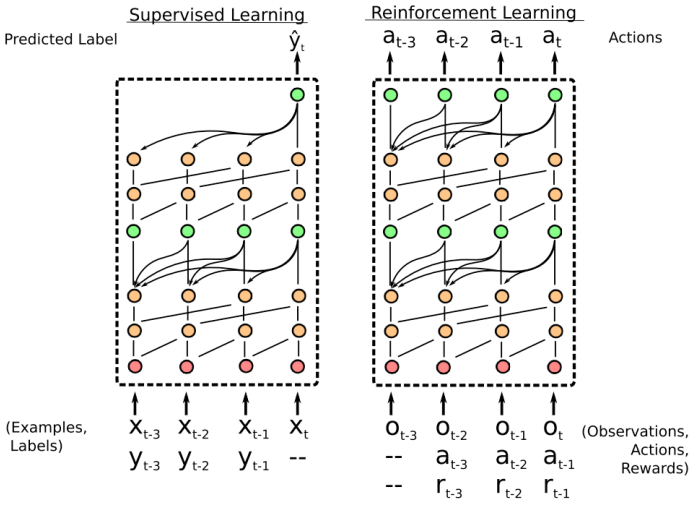

Figure 4.2.: Architecture of SNAIL, from the original paper [MRCA17]: Wavenet-connections (orange) and self-attention (green) are alternating being used to transform the input.

Statistical significance In order to assure statistical significance over the comparison of different algorithms over different datasets, we will collect multiple repetitions and perform statistical tests. Given totally three algorithms to evaluate, we compare three pairs of algorithms, or groups, against each other. All following tests will be computed over the validation accuracy after the last training epoch by a set of 10 repetitions per algorithm and dataset. In order to be able to perform a student's t-test, equal variances among groups have to be assured. For this we perform a levene's test according to equation 4.1 as follows:

$$W = \frac{(N-k)}{(k-1)} \cdot \frac{\sum_{i=1}^{k} N_i (Z_i - Z_{..})^2}{\sum_{i=1}^{k} \sum_{j=1}^{N_i} (Z_{ij} - Z_{i.})^2} \tag{4.1}$$

Further, Z_{ij}, $Z_{i.}$, and $Z_{..}$ are defined according to equation 4.2, equation 4.3, and equation 4.4, respectively:

$$Z_{ij} = \begin{cases} |Y_{ij} - \bar{Y_i}|, & \bar{Y_i}. \text{ is a mean of the } i\text{-th group} \\ |Y_{ij} - \tilde{Y_i}|, & \tilde{Y_i}. \text{ is a median of the } i\text{-th group} \end{cases} \tag{4.2}$$

$$Z_{i\cdot} = \frac{1}{N_i} \sum_{j=1}^{N_i} Z_{ij} \tag{4.3}$$

$$Z_{\cdot\cdot} = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{N_i} Z_{ij} \tag{4.4}$$

Since only two groups are compared at a time, it holds that $k = 2$, $N_1 = N_2 = 10$, and consequently $N = 20$. Further, Y_{ij} corresponds to the measured validation accuracy of a single run and a single dataset. For each pair fulfilling the assumption of equal variance, we will perform a two-sided student's t-test. The following following equations 4.5 and 4.6 are applied for the student's t-test:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{s_p \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{4.5}$$

$$s_p = \sqrt{\frac{(n_1 - 1)s_{X_1}^2 + (n_2 - 1)s_{X_2}^2}{n_1 + n_2 - 2}} \tag{4.6}$$

Where $\bar{X}$ equals the group means, and s^2 corresponds to the group standard deviations. Further it holds that $n_1 = n_2 = 10$. For computing the p_{value}, we set $\alpha = 0.01$ and consider $n_1 + n_2 - 2$ degrees of freedom.

4.4. Ablation studies

Finally, we want to analyse isolated contributions for measuring their impact on the final architecture. For this, we will create several ablation studies implementing different types of attention. We conduct these ablation studies by measuring the impact of each novelty compared to the original transformer model initially. Further, we will combine isolated novelties to explore which interactions of novelties perform the best together. We use the same hyperparameter settings for all ablation studies, and train for the same amount of training epochs and steps. Each model is optimized by the same optimizer setting for 100 epochs with each 10.000 steps. The following hyperparameters have been used: $n\_layers = 2$, $d\_model = 64$, $p\_dp = 0.0$, $lr = 1e-54$, with noamlr scheduling. To our regret and for computational reasons, we were only able to collect four repetitions per ablation. We are interested in the following ablation studies:

A1. **Feature-wise, linear, and vector-based attention:** This ablation corresponds to the original transformer as proposed by [VSP+17]. As mentioned before, we will utilize a $N - 1$ architecture, meaning that $N-$encoder layers and a single decoder layer will be used.

A2. **Feature-wise, linear, and weight-based attention:** Since weight-generation is based on having not only a single task out of $N \cdot K + 1$ but all at only, this ablation

is not feasible. Here, a feature-wise transform is used meaning that each task is transformed without information of the other tasks of the same sequence.

A3. **Feature-wise, non-linear, and vector-based attention:** The linear projection of transformers will be replaced by a non-linear MLP-projection. The resulting vectors $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ will be processed by the scaled dot-product attention into attention weights. The non-linear MLP corresponds as with all other cases to a two-layer MLP with a non-linearity in-between.

A4. **Feature-wise, non-linear, and weight-based attention:** With the same reasoning as in A2., this ablation is not feasible.

A5. **Sequence-wise, linear, and vector-based attention:** Compared to the original transformer, the only modification lies in the sequence-wise rather than feature-wise linear projection. The projected vectors will be translated into attention weights by the default scaled dot-product attention mechanism.

A6. **Sequence-wise, linear, and weight-based attention:** Sequences are stacked and linearly projected. Instead of having projected vectors $\mathbf{Q}$, $\mathbf{K}$, attention weights will be directly produced in a non-linear way.

A7. **Sequence-wise, non-linear, and vector-based attention:** Sequences are stacked and non-linearly projected. Attention weights are produced by the scaled dot-product attention and applied onto a non-linear projection of $\mathbf{V}$.

A8. **Sequence-wise, non-linear, and weight-based attention:** This study is equivalent to the suggested and discussed Few-Shot Transformer. The feature-wise transform is being replaced by stacking all tasks together and transforming them at once by a non-linear MLP. This MLP does not produce feature vectors, but an attention weight matrix directly.

5. Results

In this section, we will present and discuss the experimental results proposed in the previous section. For this, we start by the in-depth analysis of the hyperparameter importance and impact on model performance. We will proceed with the performance of the best Few-Shot Transformer per dataset in comparison to competing algorithms per dataset or task. Finally, we will present and discuss the ablation studies, showing the impact of different novelties.

5.1. In-depth analysis

For the in-depth analysis, we evaluate on total 180 unique configurations of hyperparameters. The performance is discussed based on the validation accuracy of the Omniglot dataset. We present the results in the form of a parallel coordinates chart as shown in Figure 5.1. The chart the relationship of hyperparameters to the validation accuracy. Most importantly to the model performance of our suggested Few-Shot Transformer is the

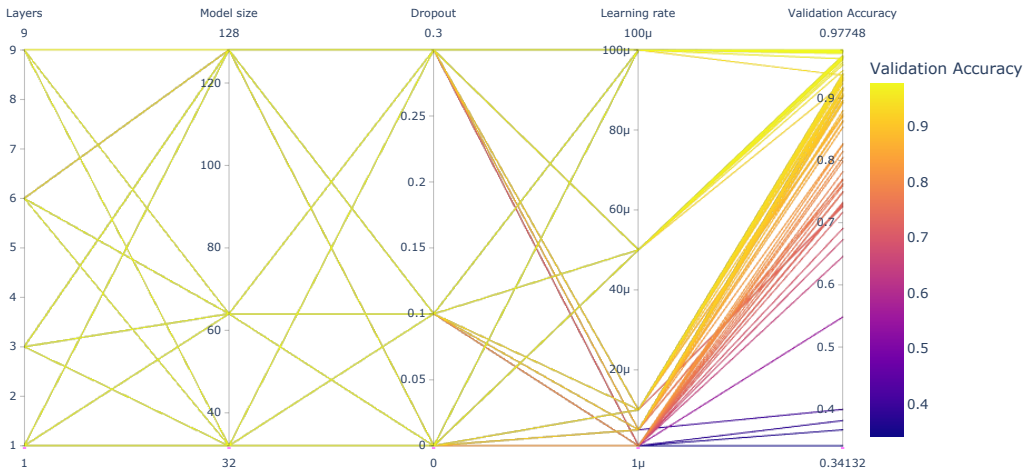

Figure 5.1.: Parallel coordinates grid of hyperparameter configuration vs. Accuracy on the validation set of Omniglot.

learning rate *lr*. We observe weak performances on learning rates below $5e-05$ and best

performances for all learning rates larger than this threshold. We observe this knowledge from Figure 5.1. Further, we recognize the model as being agnostic to different sets of hyperparameters since for each set of hyperparameters there seems to be a countering hyperparameter leading to a good performance.

5.2. Image tasks

Image tasks are drawn from the Omniglot and MiniImageNet dataset but are trained and evaluated separately. We compare the run quality and final accuracy after the full training time.

Omniglot The evaluation of the three models Few-Shot Transformer, Transformer, and SNAIL is shown in Figure 5.2. Our proposed model of $n\_layers = 1$, $d\_model = 64$, $p\_dp = 0.0$, $lr = 1e-4$, with a constant scheduling and adam optimizer, outperforms a Transformer model of $n\_layers = 3$, $d\_model = 32$, $p\_dp = 0.0$, $lr = 5e-3$ with noamlr scheduling. SNAIL has been used in the original implementation with $lr = 3.3e-5$. All three models converge after the set amount of training steps and iterations. The Few Shot

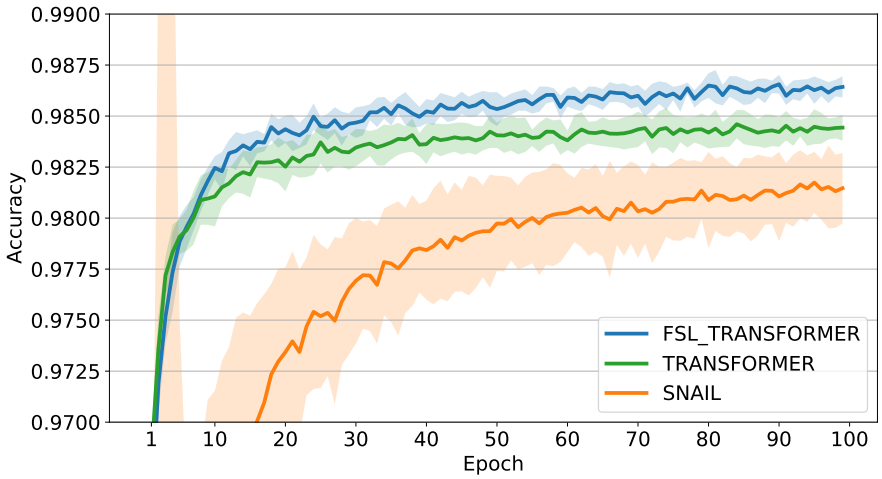

Figure 5.2.: Comparing performances on Omniglot, with 10 runs per model.

Transformer achieves a mean accuracy after the final epoch on the validation set of 98.64%, followed by Transformer with 98.44% and SNAIL with 98.14%. For all pairs of FSL Transformer and Transformer, FSL Transformer and SNAIL, Transformer and SNAIL, the levene test is significant, thus allowing to test the group means by the student's t-test. Further, the student's t-test is significant for all groups by the previously set significance level of $\alpha = 0.01$. We report p-values of 4.64E-07, 7.53E-08, 7.73E-05 for the pairs of (FSL_Transformer, Transformer), (FSL_Transformer, SNAIL) and (Transformer, SNAIL), respectively.

MiniImageNet On MiniImageNet, the Few-Shot Transformel of configuration with $n\_layers = 3$, $d\_model = 128$, $p\_dp = 0.1$, $lr = 1e-5$, with noamlr scheduling outperforms all other models in our scope. The competing Transformer model has been configured with $n\_layers = 1$, $d\_model = 64$, $p\_dp = 0.0$, $lr = 4e-7$ and a constant learning rate with adam optimization. SNAIL has been used originally with a learning rate $lr = 1.186e-4$. The final results are shown in Figure 5.3. The Few Shot Transformer achieves a mean

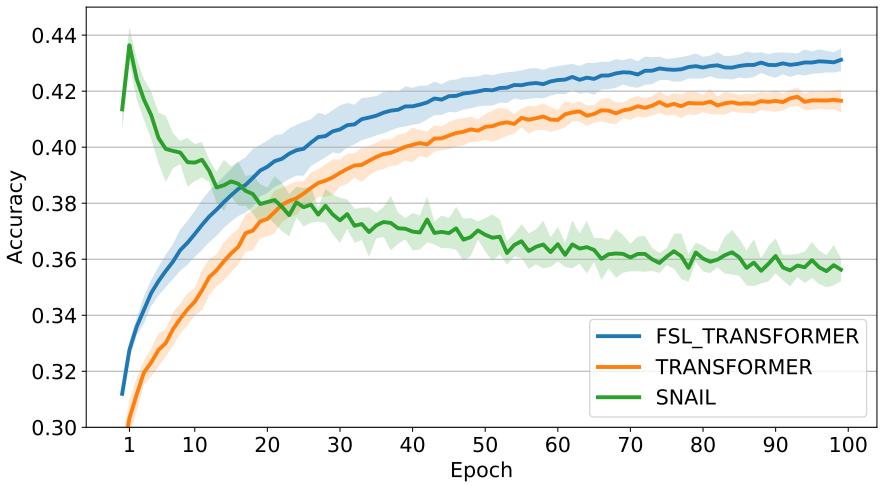

Figure 5.3.: Comparing performances on MiniImageNet, with 10 runs per model.

accuracy after the final epoch on the validation set of 43.12%, followed by Transformer with 41.66% and SNAIL with 35.63%. For all pairs of FSL Transformer and Transformer, FSL Transformer and SNAIL, Transformer and SNAIL, the levene test is significant, thus allowing to test the group means by the student's t-test. Further, the student's t-test is significant for all groups by the previously set significance level of $\alpha = 0.01$. We report p-values of 2.29E-07, 2.83E-18, 1.18E-16 for the pairs of (FSL_Transformer, Transformer), (FSL_Transformer, SNAIL) and (Transformer, SNAIL), respectively.

5.3. Tabular tasks

Symbolic Regression On the Symbolic Regression task, we see an equal performance between our Few-Shot Transformer and the original Transformer model. Our model has been configured with $n\_layers = 3$, $d\_model = 32$, $p\_dp = 0.1$, $lr = 3.5e-7$ and a constant learning rate scheduled by adam. The Transformer model had the following set of hyperparameters: $n\_layers = 3$, $d\_model = 64$, $p\_dp = 0.1$, $lr = 1.0e-6$ and a constant learning rate scheduled by adam. For SNAIL, the learning rate $lr = 2.5e-6$ has been used. The final results are depicted in Figure 5.4. The Few Shot Transformer achieves a mean accuracy after the final epoch on the validation set of 79.06%, followed by Transformer with 78.87% and SNAIL with 76.93%. For all pairs of FSL Transformer and Transformer, FSL Transformer and SNAIL, Transformer and SNAIL, the levene test is significant, thus

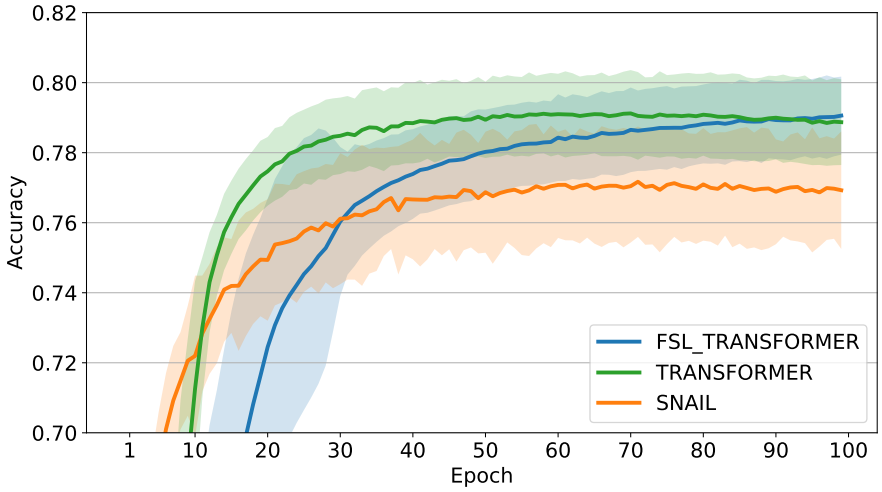

Figure 5.4.: Comparing performances on Symbolic Regression, with 10 runs per model.

allowing to test the group means by the student's t-test. Further and by the previously set level of significance at $\alpha = 0.01$, the pair of FSL Transformer and Transformer are not different in their group means by the student's t-test. However, all other group tests are significantly different. We report p-values of 0.72, 0.0036, 0.0084 for the pairs of (FSL_Transformer, Transformer), (FSL_Transformer, SNAIL) and (Transformer, SNAIL), respectively. On a closer look, our proposed Few-Shot Transformer slightly outperforms the original transformer model in an average of 10 runs. Further, while the original Transformer seems to be fully converged, the Few-Shot Transformer still has a positive trend in accuracy versus epoch.

5.4. Ablation studies

The ablations studies investigate the question of which novelty or combination of novelties leads to the largest impact on model performance. For computational reasons, we were unfortunately only able to collect four evaluations per ablation. Nevertheless, we highlight that the same settings in terms of configuration, training time, and dataset have been used for all studies. The results are presented in Figure 5.5. While the original transformer ($A1$) achieves a validation set accuracy of 95.53%, the effect of replacing a linear with a non-linear transform ($A3$) actually decreases the performance to 94.43%. All other ablations, which implement the sequence-wise transform, outperform by a margin of at least 2.03%. From the group of sequence-wise transform ablations we cannot observe significant differences between linear and non-linear transformations, or between vector- and weight-based attention. The levene test for all pairs is significant, verifying that all variances are equal among groups compared. This allows to test all groups for their mean by the student's t-test. The result is of all pairs is equivalent to what Figure 5.5 shows, namely that ($A5$, $A6$, $A7$, $A8$) are not significantly differnent to each other. On the

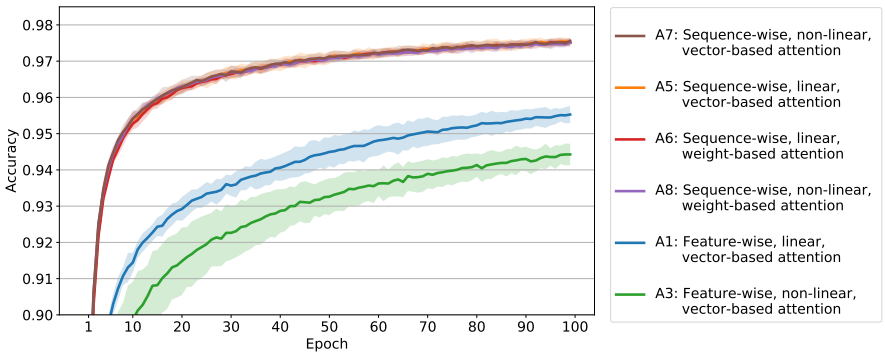

Figure 5.5.: Comparison of the ablation studies: *A1* and *A3*, both implementing feature-wise transforms, perform the worst. All high-performing ablations implement sequence-wise transforms. There is no significant difference between *A7*, *A5*, *A6*, and *A8*.

otherside, these groups are different to (*A1*, *A3*). Finally, *A1* and *A3* are significantly different to each other. We report the following p-values: 4.48E-06, 0.0011, 4.54E-06, 3.36E-06, 3.50E-06, 1.13E-06, 0.8794, 0.8581, 0.5952, 1.14E-06, 9.40E-07, 9.64E-07, 0.9982, 0.4893, 0.4127 for the pairs of (*A1*, *A5*), (*A1*, *A3*), (*A1*, *A6*), (*A1*, *A8*), (*A1*, *A7*), (*A5*, *A3*), (*A5*, *A6*), (*A5*, *A8*), (*A5*, *A7*), (*A3*, *A6*), (*A3*, *A8*), (*A3*, *A7*), (*A6*, *A8*), (*A6*, *A7*) and (*A8*, *A7*), respectively.

6. Discussion and Conclusion.

In this study, we investigated the performance of transformer models on Few-Shot Learning settings. For this, we captured the historical development of attention, beginning from neural-turing-machines up to self-attention as implemented by transformer models. We highlighted recent advances in transformer-based models and found how many improvements can be classified into efficiency, performance, or scalability on singular tasks such as NLP or computer vision. However, for the lack of specifically testing and improving transformer models in the Few-Shot Learning setting, we were motivated to research modifications to the self-attention mechanism.

Our novelties are centred around the idea of empowering from the typical data tensor within Few-Shot Learning, where support and query samples are stacked at showed to the model as a single instance. While original transformers regard each sample as a different task and thus transform each task independently of all others, we proposed to stack all tasks together before their transformation. This significantly increases the variance of a single instance, and hence leads to a denser data structure and higher sample-efficiency. Given the higher data density, we further proposed more complex transformations by shifting from linear to non-linear and deep transformations. Finally, instead of computing attention weights by the self-attention mechanism, we investigated the effect of directly computing such weights by an MLP. Our three proposed novelties are summarized as:

N1. Stacking sequences of features.

N2. Replacing linear transforms with non-linear transforms.

N3. Replacing scaled dot-product attention with an MLP-based attention weight generation.

By an empirical study, we showed how novelty *N1* leads to the most significant improvement compared to original transformer models on the Few-Shot Learning setting, as tested on the Omniglot dataset. Thus, we see our hypothesis of benefiting from a higher input data variance to the model performance as confirmed. However and to our surprise, we were not able to measure significant effects on the dataset for *N2* and *N3*. Neither replacing the linear with a non-linear transformation nor replacing the scaled dot-product attention with a direct weight generation accelerated the model performance on the used datasets. We assume that these datasets may do yield a complexity level high enough such that a more complex transformation would lead to higher performance. Yet, to confirm this assumption further experiments on more complex datasets such as

MiniImagenet are advised to the fellow reader. For computational and time bounds, we were not able to conduct these studies by ourselves.

Further, our proposed Few-Shot Transformer outperforms not only the original transformer but also state-of-the-art algorithms such as SNAIL[MRCA17] on competitive tasks such as Omniglot and MiniImagenet. On the novel dataset produced by ourselves, the performance of both our proposed model and SNAIL were competitive on the number of training steps conducted. In all instances, our model outperformed original transformer models, giving highlights on how to improve and use such attention-based models in Few-Shot Learning settings. To our surprise, the transformer model also outperformed SNAIL in many instances. We remark this since SNAIL was specifically designed for the few-shot setting. Another way of interpreting this result is also that NLP, the original domain of transformers, can be regarded as a few-shot learning setting, hence transformer models also perform outside NLP but in the few-shot learning setting powerfully.

Concerning our research questions, we quickly summarize our findings as follows:

RQ1: Original transformers can outperform models specifically designed for the few-shot learning setting, but with data modalities outside of NLP. This raises the idea that NLP can be regarded as a form of few-shot learning, thus making transformer models naturally powerful even on modalities they were not specifically designed for.

RQ2: Three modifications have been proposed: the stacking of tasks within a sequence, a non-linear transform, and the direct computation of attention weights.

RQ3: Most importantly, the stacking of tasks of a sequence raises the performance significantly in comparison to feature-wise transforms only.

For future work, we would like to see more studies shining light on why non-linear transformations and direct attention weight generation did not lead to distinguishable results. As mentioned before, we recommend studies on different datasets. Since there is evidence for the effect of direct attention weight generation by [JKZF20], we would like to see more studies in this direction. Finally, for reasons of limited time and resources, we were not able to test out different algorithms such as the perceiver[JGB$^+$21]. Such competitors are also interesting to compare to, even though we excluded them for not being designed for the few-shot learning setting.

Bibliography

[ADG+16] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *arXiv preprint arXiv:1606.04474*, 2016.

[BB12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[Bie20] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[BMR+20] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[DBK+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[FLG+20] Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*, 2020.

[FSH15] Matthias Feurer, Jost Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[FZS21] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

[GWD14] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HvRP21] Mike Huisman, Jan N van Rijn, and Aske Plaat. A survey of deep meta-learning. *Artificial Intelligence Review*, pages 1–59, 2021.

[JGB+21] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*, 2021.

[JKZF20] Zihang Jiang, Bingyi Kang, Kuangqi Zhou, and Jiashi Feng. Few-shot classification via adaptive attention. *arXiv preprint arXiv:2008.02465*, 2020.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[LeC98] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[LHL+20] Lu Liu, William Hamilton, Guodong Long, Jing Jiang, and Hugo Larochelle. A universal representation transformer layer for few-shot image classification. *arXiv preprint arXiv:2006.11702*, 2020.

[LLO+20] Lajanugen Logeswaran, Ann Lee, Myle Ott, Honglak Lee, Marc'Aurelio Ranzato, and Arthur Szlam. Few-shot sequence learning with transformers. *arXiv preprint arXiv:2012.09543*, 2020.

[LST15] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[met] Meta-learning challenges by chalearn.org: `https://metalearning.chalearn.org/`.

[MRCA17] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[RL16] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[RWC+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[SBB+16] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[Sin20] Praphul Singh. Multi-head self-attention in nlp. *https://blogs.oracle.com/ai-and-datascience/post/multi-head-self-attention-in-nlp*, 2020.

[SIS21] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight memory systems. *arXiv preprint arXiv:2102.11174*, 2021.

[SSZ17] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[SYZ+18] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[Ton21] Akshay Tondak. Introduction to recurrent neural networks (rnn). *https://k21academy.com/datascience/machine-learning/recurrent-neural-networks/Disadvantages$_O f_R NN_s$*, 2021.

[VBL+16] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[VRH18] Jan N Van Rijn and Frank Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2367–2376, 2018.

[vRPT+18] Jan N van Rijn, Florian Pfisterer, Janek Thomas, Andreas Muller, Bernd Bischl, and Joaquin Vanschoren. Meta learning for defaults: Symbolic defaults. In *Neural Information Processing Workshop on Meta-Learning*, 2018.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[WLK+20] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[ZGD+20] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

A. Appendix

A.1. Symbolic Regression dataset

A.1.1. Introduction

During the thesis, we have been working on various problems and ideas. One of the early ideas was surrounded by the idea of simplifying the encoded information captured by differentiable models such as neural networks. We regard a neural network as a function approximator that, by the number of its parameters, can capture any arbitrary function as long as the minimum length of two layers has been reached. This allows a user to utilize a neural network blindly on incoming data since no assumptions on linear or non-linear relationships have to be fulfilled. The downside of such an approach lies of course in the difficulty of interpreting such a massively over-parametrized model. Therefore there is a trade-off in carefully selecting the sleekest model for a specific use case versus selecting a neural network of sufficient size for any possible task, which then is unfortunately not interpretable.

A.1.2. Motivation

The dataset generation has been motivated by a Meta-Learning approach, in which we ask whether a meta-learner may be able to ingest a base neural network and output the function the base network has learned in a symbolic way.

A.1.3. Methods

For this, we define a set of functions $f \in \mathcal{F}$, i.e in the form of $f(p) = X \cdot p$. Over a fixed range of X, we compute a dataset by evaluating f at each point $x \in X$. For each function we result with a dataset we can use to fit a base network. For this, we first define an architecture of the base network, in terms of layers and learnable parameters per layer, which we keep fixed for all future steps. For the meta-network, we similarity define an architecture. This architecture may be larger and more complex than the base architecture since a more complex dependency has to be captured. Finally, we define the output format of the meta-learner. Conceptually, the learned function of the base network shall be outputted, preferably in a symbolic fashion. For the scope of our project, we handled constant, linear, and quadratic functions. Thus, we settled for a model with four outputs, in which each output estimated a parameter of the possible function.

A.1.4. Training

Base networks

From the set of datasets per function, we compute a set of base networks. Each base network is randomly instantiated from the base architecture and then trained on a dataset, function, respectively, until the loss decreased below a certain threshold. Since over-parametrized networks are initialized randomly and converged stochastically, a novel solution with respect to the parameters can be expected on almost every new instance. This allows to train the same base architecture on the same dataset or function, respectively, multiple times and still have different solutions with respect to the trained parameters. Finally, this results in a dataset of trained base networks, for which we exactly know the function it was trained on.

Meta networks

Having a dataset of 10.000 base networks per function, we can encode those networks in a way a meta-network can ingest them and learn the desired dependency we are looking for. Since the base architecture is always a multi-layer perceptron of the same architecture, we only feed the weights as training data points into the meta-network. The output of the meta networks is multi-output, where each output corresponds to a parameter $p \in P$ of the original function $f_P = X \cdot P$.