On the rate of convergence of a classifier based on a Transformer encoder[1]

Iryna Gurevych[†], Michael Kohler[*,2], Gözde Gül Şahin[†]

* Department of Mathematics, Technical University of Darmstadt, Schlossgartenstr. 7, 64289 Darmstadt, Germany, email: kohler@mathematik.tu-darmstadt.de
† Ubiquitous Knowledge Processing (UKP) Lab, Technical University of Darmstadt, Germany, email: gurevych@ukp.informatik.tu-darmstadt.de, sahin@ukp.informatik.tu-darmstadt.de.

November 22, 2021

Abstract

Pattern recognition based on a high-dimensional predictor is considered. A classifier is defined which is based on a Transformer encoder. The rate of convergence of the misclassification probability of the classifier towards the optimal misclassification probability is analyzed. It is shown that this classifier is able to circumvent the curse of dimensionality provided the aposteriori probability satisfies a suitable hierarchical composition model. Furthermore, the difference between Transformer classifiers analyzed theoretically in this paper and Transformer classifiers used nowadays in practice are illustrated by considering classification problems in natural language processing.

AMS classification: Primary 62G05; secondary 62G20.

Key words and phrases: Curse of dimensionality, Transformer, classification, rate of convergence.

1 Introduction

Deep learning has achieved impressive progress in natural language processing (NLP), e.g. in the areas of understanding, summarizing or generating text, see., e.g., Young et al. (2018) and the literature cited therein. Among the most successful techniques in this field are Transformers introduced by Vaswani et al. (2017). They used a decoder-encoder structure based on multi-head attention and piecewise feedforward layers and achieved a top performance in applications in machine translation. Devlin et al. (2019) describe how Transformers can be combined with unsupervised pre-training such that the same pre-trained Transformer encoder can be fine-tuned to a variety of natural language processing tasks.

Besides the huge practical success of these estimates, their theoretical performance has not been studied intensively until now. This is in sharp contrast to deep neural networks,

---

[1] Running title: *Rate of convergence of a Transformer classifier*
[2] Corresponding author. Tel: +49-6151-16-23382

where various results concerning the approximation power of deep neural networks (cf., e.g., Yarotsky (2017), Yarotsky and Zhevnerchuck (2020), Lu et al. (2020), Langer (2021b) and the literature cited therein) or concerning the statistical risk of corresponding estimates (cf., e.g., Bauer and Kohler (2019), Kohler and Krzyżak (2017), Schmidt-Hieber (2019), Kohler and Langer (2020), Langer (2021a), Imaizumi and Fukumizu (2019), Suzuki (2018), Suzuki and Nitanda (2019), and the literature cited therein) have been derived in the last few years.

In this paper we aim to shed light on the theoretical performance of Transformers. To do this, we focus on pattern recognition and consider estimates based on a Transformer encoder. Here we simplify the learning problem by defining our estimate as a plug-in classification rule based on an abstract least squares estimates, i.e., we ignore the gradient descent usually applied in practice as this gradient descent is even for deep neural networks nowadays not well understood. After this simplification the main remaining challenges in view of a theoretical understanding are the approximation properties and the generalization abilities of Transformer encoders.

We study these estimates in the context of pattern recognition. Here, (X, Y), (X_1, Y_1), ..., (X_n, Y_n) are independent and identically distributed random variables with values in $\mathbb{R}^{d \cdot l} \times \{0, 1\}$, and given the data set

$$\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

the goal is to construct a classifier

$$\eta_n(\cdot) = \eta_n(\cdot, \mathcal{D}_n) : \mathbb{R}^{d \cdot l} \to \{0, 1\}$$

such that the misclassification probability

$$\mathbf{P}\{\eta_n(X) \neq Y | \mathcal{D}_n\}$$

is as small as possible. Here the predictor variable X describes the encoding of a sequence of lenght l consisting of words or tokens, and each word or token is encoded by a value in $\mathbb{R}^d$.

Let

$$m(x) = \mathbf{P}\{Y = 1 | X = x\} \quad (x \in \mathbb{R}^{d \cdot l}) \tag{1}$$

be the so–called aposteriori probability of class 1. Then

$$\eta^*(x) = \begin{cases} 1, & \text{if } m(x) > \frac{1}{2} \\ 0, & \text{elsewhere} \end{cases}$$

is the so–called Bayes classifier, i.e., it satisfies

$$\mathbf{P}\{\eta^*(X) \neq Y\} = \min_{\eta : \mathbb{R}^{d \cdot l} \to \{0, 1\}} \mathbf{P}\{\eta(X) \neq Y\}$$

(cf., e.g., Theorem 2.1 in Devroye, Györfi and Lugosi (1996)).

In the sequel we try to derive upper bounds on

$$\mathbf{E}\left\{\mathbf{P}\{\eta_n(X) \neq Y | \mathcal{D}_n\} - \mathbf{P}\{\eta^*(X) \neq Y\}\right\}$$
$$= \mathbf{P}\{\eta_n(X) \neq Y\} - \min_{\eta:\mathbb{R}^{d \cdot l} \to \{0,1\}} \mathbf{P}\{\eta(X) \neq Y\}. \tag{2}$$

It is well-known that in order to derive nontrivial rate of convergence results on the difference between the misclassification risk of any estimate and the minimal possible value it is necessary to restrict the class of distributions (cf., Cover (1968) and Devroye (1982)). In this context we will assume that the aposteriori probability is smooth, and for this we will use our next definition.

Definition 1. *Let $p = q + s$ for some $q \in \mathbb{N}_0$ and $0 < s \leq 1$. A function $m : \mathbb{R}^{d \cdot l} \to \mathbb{R}$ is called (p, C)-smooth, if for every $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{d \cdot l}) \in \mathbb{N}_0^{d \cdot l}$ with $\sum_{j=1}^{d \cdot l} \alpha_j = q$ the partial derivative $\partial^q m / (\partial x_1^{\alpha_1} \ldots \partial x_{d \cdot l}^{\alpha_{d \cdot l}})$ exists and satisfies*

$$\left| \frac{\partial^q m}{\partial x_1^{\alpha_1} \ldots \partial x_{d \cdot l}^{\alpha_{d \cdot l}}}(x) - \frac{\partial^q m}{\partial x_1^{\alpha_1} \ldots \partial x_{d \cdot l}^{\alpha_{d \cdot l}}}(z) \right| \leq C \|\mathbf{x} - \mathbf{z}\|^s$$

for all $\mathbf{x}, \mathbf{z} \in \mathbb{R}^{d \cdot l}$, where $\| \cdot \|$ denotes the Euclidean norm.

In order to be able to show good rate of convergences even for high-dimensional predictors we use a hierarchical composition model as in Schmidt-Hieber (2019), where the aposteriori probability is represented by a composition of several functions and where each of these functions depends only on a few variables. We use the following definition of Kohler and Langer (2020) to formalize this assumption.

Definition 2. *Let $d, l \in \mathbb{N}$, $m : \mathbb{R}^{d \cdot l} \to \mathbb{R}$ and let $\mathcal{P}$ be a subset of $(0, \infty) \times \mathbb{N}$.*
a) *We say that m satisfies a hierarchical composition model of level 0 with order and smoothness constraint $\mathcal{P}$, if there exists $K \in \{1, \ldots, d \cdot l\}$ such that*

$$m(\mathbf{x}) = x^{(K)} \quad \text{for all } \mathbf{x} = (x^{(1)}, \ldots, x^{(d \cdot l)})^\top \in \mathbb{R}^{d \cdot l}.$$

b) *Let $\kappa \in \mathbb{N}_0$. We say that m satisfies a hierarchical composition model of level $\kappa + 1$ with order and smoothness constraint $\mathcal{P}$, if there exist $(p, K) \in \mathcal{P}$, $C > 0$, $g : \mathbb{R}^K \to \mathbb{R}$ and $f_1, \ldots, f_K : \mathbb{R}^d \to \mathbb{R}$, such that g is (p, C)-smooth, $f_1, \ldots, f_K$ satisfy a hierarchical composition model of level κ with order and smoothness constraint $\mathcal{P}$ and*

$$m(\mathbf{x}) = g(f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})) \quad \text{for all } \mathbf{x} \in \mathbb{R}^{d \cdot l}.$$

Let $\mathcal{H}(\kappa, \mathcal{P})$ be the set of all functions $m : \mathbb{R}^{d \cdot l} \to \mathbb{R}$ which satisfy a hierarchical composition model of level κ with order and smoothness constraint $\mathcal{P}$.

It was shown in Bauer and Kohler (2019), Schmidt-Hieber (2019) and Kohler and Langer (2020) that deep neural networks are able to circumvent the curse of dimensionality in case that the function to be estimated is contained in a suitably defined hierarchical composition model. The main contribution of this paper is to show that classifiers based on Transformer encoders, which we will introduce in the next section,

3

have this property, too. More precisely, we will show that the classifier η_n, which is introduced in the next section on the basis of a Transformer encoder, satisfies

$$\mathbf{P}\{\eta_n(X) \neq Y\} - \min_{f:\mathbb{R}^{d \cdot l} \to \{0,1\}} \mathbf{P}\{f(X) \neq Y\} \leq c_1 \cdot (\log n)^3 \cdot \max_{(p,K) \in \mathcal{P}} n^{-\frac{p}{2p+K}} \qquad (3)$$

provided the aposteriori probability satisfies a hierarchical composition model with some finite level and smoothness and order constraint $\mathcal{P}$. Since the above rate of convergence does not depend on the dimension $d \cdot l$ of X, our classifier is able to circumvent the curse of dimensionality if the aposteriori probability satisfies a suitable hierarchical composition model.

In order to show (3) we derive new approximation properties and generalization bounds for Transformer decoders. The main idea here is to show that the combination of attention units with piecewise feedforward neural networks enables us to reconstruct piecewise polynomials by Transformer decoders, and to generalize a bound on the VC dimension of deep neural networks from Bartlett et al. (2019) such that it is applicable to Transformer encoders. The Transformer classifiers analyzed theoretically in this paper are different from the one used in practice. We illustrate this by describing classification problems in natural language processing and the methods nowadays used in practice to solve these problems. This gives us useful hints in which way the theoretical results in this paper should be generalized in future work.

1.1 Notation

Throughout this paper, the following notation is used: The sets of natural numbers, natural numbers including 0, integers and real numbers are denoted by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$ and $\mathbb{R}$, respectively. For $z \in \mathbb{R}$, we denote the smallest integer greater than or equal to z by $\lceil z \rceil$. Furthermore we set $z_+ = \max\{z, 0\}$. Let $D \subseteq \mathbb{R}^d$ and let $f : \mathbb{R}^d \to \mathbb{R}$ be a real-valued function defined on $\mathbb{R}^d$. We write $x = \arg\min_{z \in D} f(z)$ if $\min_{z \in \mathcal{D}} f(z)$ exists and if x satisfies $x \in D$ and $f(x) = \min_{z \in \mathcal{D}} f(z)$. For $f : \mathbb{R}^d \to \mathbb{R}$

$$\|f\|_\infty = \sup_{x \in \mathbb{R}^d} |f(x)|$$

is its supremum norm, and the supremum norm of f on a set $A \subseteq \mathbb{R}^d$ is denoted by

$$\|f\|_{A,\infty} = \sup_{x \in A} |f(x)|.$$

Furthermore we define the norm $\| \cdot \|_{C^q(A)}$ of the smooth function space $C^q(A)$ by

$$\|f\|_{C^q(A)} := \max \left\{ \|\partial^{\mathbf{j}} f\|_{\infty,A} : \|\mathbf{j}\|_1 \leq q, \mathbf{j} \in \mathbb{N}^d \right\}$$

for any $f \in C^q(A)$, where

$$\partial^{\mathbf{j}} f = \frac{\partial^{j_1 + \cdots + j_d} f}{\partial x_1^{j_1} \ldots \partial x_d^{j_d}} \quad \text{for } \mathbf{j} = (j_1, \ldots, j_d)^T \in \mathbb{N}^d.$$

Let $\mathcal{F}$ be a set of functions $f : \mathbb{R}^d \to \mathbb{R}$, let $x_1, \ldots, x_n \in \mathbb{R}^d$ and set $x_1^n = (x_1, \ldots, x_n)$. A finite collection $f_1, \ldots, f_N : \mathbb{R}^d \to \mathbb{R}$ is called an ε– cover of $\mathcal{F}$ on x_1^n if for any $f \in \mathcal{F}$ there exists $i \in \{1, \ldots, N\}$ such that

$$\frac{1}{n} \sum_{k=1}^{n} |f(x_k) - f_i(x_k)| < \varepsilon.$$

The ε–covering number of $\mathcal{F}$ on x_1^n is the size N of the smallest ε–cover of $\mathcal{F}$ on x_1^n and is denoted by $\mathcal{N}_1(\varepsilon, \mathcal{F}, x_1^n)$.

For $z \in \mathbb{R}$ and $\beta > 0$ we define $T_\beta z = \max\{-\beta, \min\{\beta, z\}\}$. For $i, j \in \mathbb{N}_0$ we let $\delta_{i,j}$ be the Kronecker delta, i.e., we set

$$\delta_{i,j} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{else.} \end{cases}$$

If W is a matrix and b is a vector then we denote the number of nonzero components in W and b by $\|W\|_0$ and $\|b\|_0$, respectively.

1.2 Outline

In Section 2 our classifier based on a Transformer encoder is defined. The main result is presented in Section 3 and proven in Section 5. Section 4 describes the application of Transformer classifiers in natural language processing.

2 Definition of a classifier based on a Transformer encoder

The Transformer encoder which we introduce in this section becomes as input a sequence

$$x = (x_1, \ldots, x_l) \in \mathbb{R}^{l \cdot d} \tag{4}$$

of length l consisting of components $x_j \in \mathbb{R}^d$ $(j = 1, \ldots, l)$, where l and d are natural numbers. As a first step it produces from this sequence a new representation

$$z_0 = (z_{0,1}, \ldots, z_{0,l}) \in \mathbb{R}^{l \cdot d_{model}} \tag{5}$$

for some $d_{model} \in \mathbb{N}$. This new representation is defined as follows: We choose $h, I \in \mathbb{N}$ and set

$$d_{model} = h \cdot I \cdot (d + l + 4). \tag{6}$$

Here we repeat a coding of the input which includes the original data, a coding of the position and additional auxiliary values used for later computation of function values $h \cdot I$ times. More precisely we set for all $k \in \{1, \ldots, h \cdot I\}$

$$z_{0,j}^{((k-1) \cdot (d+l+4)+s)} = \begin{cases} x_j^{(s)} & \text{if } s \in \{1, \ldots, d\} \\ 1 & \text{if } s = d+1 \\ \delta_{s-d-1,j} & \text{if } s \in \{d+2, \ldots, d+1+l\} \\ 1 & \text{if } s = d+l+3 \\ 0 & \text{if } s \in \{d+l+2, d+l+4\} \end{cases}$$

After that it computes successively representations

$$z_r = (z_{r,1}, \ldots, z_{r,l}) \in \mathbb{R}^{l \cdot d_{model}} \tag{7}$$

of the input for $r = 1, \ldots, N$, and uses z_N in order to predict a value $y \in \{0, 1\}$. Here N is the number of pairs of attention and pointwise feedforward layers of our Transformer encoder.

Given z_{r-1} for some $r \in \{1, \ldots, N\}$ we compute z_r by applying first a multi-head attention and by applying second pointwise a feedforward neural network with one hidden layer. Both times we will use an additional residual connection.

The computation of the multi-head attention depends on matrices

$$W_{Q,r,s}, W_{K,r,s} \in \mathbb{R}^{d_k \times d_{model}} \quad \text{and} \quad W_{V,r,s} \in \mathbb{R}^{d_v \times d_{model}} \quad (s = 1, \ldots, h), \tag{8}$$

where $h \in \mathbb{N}$ is the number of attentions which we compute in parallel, where $d_k \in \mathbb{N}$ is the dimension of the queries and the keys, and where $d_v = d_{model}/h = I \cdot (d + l + 4)$ is the dimension of the values. We use these matrices to compute for each component $z_{r-1,i}$ of z_{r-1} corresponding queries

$$q_{r-1,s,i} = W_{Q,r,s} \cdot z_{r-1,i}, \tag{9}$$

keys

$$k_{r-1,s,i} = W_{K,r,s} \cdot z_{r-1,i} \tag{10}$$

and values

$$v_{r-1,s,i} = W_{V,r,s} \cdot z_{r-1,i} \tag{11}$$

$(s \in \{1, \ldots, h\}, i \in \{1, \ldots, l\})$. Then the so-called attention between the component i of z_{r-1} and the component j of z_{r-1} is defined as the scalar product

$$< q_{r-1,s,i}, k_{r-1,s,j} > \tag{12}$$

and the index j for which the maximal value occurs, i.e.,

$$\hat{j}_{r-1,s,i} = \underset{j \in \{1,\ldots,l\}}{\arg \max} < q_{r-1,s,i}, k_{r-1,s,j} >, \tag{13}$$

is determined. The value corresponding to this index is multiplied with the maximal attention in (12) in order to define

$$
\begin{aligned}
\bar{y}_{r,s,i} &= v_{r-1,s,\hat{j}_{r-1,s,i}} \cdot \underset{j \in \{1,\ldots,l\}}{\max} < q_{r-1,s,i}, k_{r-1,s,j} > \\
&= v_{r-1,s,\hat{j}_{r-1,s,i}} \cdot < q_{r-1,s,i}, k_{r-1,s,\hat{j}_{r-1,s,i}} > \quad (s \in \{1, \ldots, h\}, i \in \{1, \ldots, l\}). \tag{14}
\end{aligned}
$$

Using a residual connection we compute the output of the multi-head attention by

$$y_r = z_{r-1} + (\bar{y}_{r,1}, \ldots, \bar{y}_{r,l}) \tag{15}$$

where

$$\bar{y}_{r,s} = (\bar{y}_{r,s,1}, \ldots, \bar{y}_{r,s,h}) \in \mathbb{R}^{d_v \cdot h} = \mathbb{R}^{d_{model}} \quad (s \in \{1, \ldots, l\}).$$

Here $y_r \in \mathbb{R}^{d_{model} \cdot l}$ has the same dimension as z_{r-1}.

The output of the pointwise feedforward neural network depends on parameters

$$W_{r,1} \in \mathbb{R}^{d_{ff} \times d_{model}}, b_{r,1} \in \mathbb{R}^{d_{ff}}, W_{r,2} \in \mathbb{R}^{d_{model} \times d_{ff}}, b_{r,2} \in \mathbb{R}^{d_{model}}, \qquad (16)$$

which describe the weights in a feedforward neural network with one hidden layer and $d_{ff} \in \mathbb{N}$ hidden neurons. This feedward neural network is applied to each component of (15) (which is analoguous to a convolutionary neural network) and computes

$$z_{r,s} = y_{r,s} + W_{r,2} \cdot \sigma\left(W_{r,1} \cdot y_{r,s} + b_{r,1}\right) + b_{r,2} \quad (s \in \{1, \ldots, l\}), \qquad (17)$$

where we use again a residual connection. Here

$$\sigma(x) = \max\{x, 0\}$$

is the ReLU activation function, which is applied to a vector by applying it in each component of the vector separately.

Given the output z_N of the sequence of N multi-head attention and pointwise feedforward layers, our final classifier is computed by

$$\hat{y} = \begin{cases} 1 & \text{if } z_N \cdot w + b \geq 1/2, \\ 0 & \text{else}, \end{cases}$$

where $w \in \mathbb{R}^{d_{model} \cdot l}$ and $b \in \mathbb{R}$ are parameters of our neural network. I.e., we compute a linear transformation of z_N and use a plug-in classification rule corresponding to the function

$$(x_1, \ldots, x_l) \mapsto z_N \cdot w + b.$$

This function depends on a parameter vector

$$\bar{\theta} = \left((W_{Q,r,s}, W_{K,r,s}, W_{V,r,s})_{r \in \{1, \ldots, N\}, s \in \{1, \ldots, h\}}, (W_{r,1}, b_{r,1}, W_{r,2}, b_{r,2})_{r \in \{1, \ldots, N\}}, w, b \right).$$

We denote this function by $f_{\bar{\theta}}$, i.e., $f_{\bar{\theta}} : \mathbb{R}^{d \cdot l} \to \mathbb{R}$ is the function

$$f_{\bar{\theta}}(x_1, \ldots, x_l) = z_N \cdot w + b,$$

where z_N is the value computed as described above on the basis of the matrices and vectors contained in $\bar{\theta}$.

In order to learn this function from observed data (X_1, Y_1), ..., (X_n, Y_n), we use the principle of least squares to fit such a function to the observed data under a sparsity constraint. To do this, we denote the number of nonzero parameters of our transformer network by

$$\begin{aligned} \|\bar{\theta}\|_0 &= \sum_{r=1}^{N} \sum_{s=1}^{l} \left(\|W_{Q,r,s}\|_0 + \|W_{K,r,s}\|_0 + \|W_{V,r,s}\|_0 \right) \\ &+ \sum_{r=1}^{N} \left(\|W_{r,1}\|_0 + \|b_{r,1}\|_0 + \|W_{r,2}\|_0 + \|b_{r,2}\|_0 \right) + \|w\|_0 + \|b\|_0, \end{aligned}$$

7

where $\| \cdot \|_0$ is the number of nonzero entries in a matrix or in a vector. We then choose a sparsity index $L_n \in \mathbb{N}$ and define

$$m_n(\cdot) = \arg \min_{f \in \{f_{\bar{\theta}} : \|\bar{\theta}\|_0 \leq L_n\}} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(X_i)|^2 \qquad (18)$$

and

$$\eta_n(x) = \begin{cases} 1 & \text{if } m_n(x) \geq 1/2, \\ 0 & \text{else.} \end{cases} \qquad (19)$$

In (18) we assume for simplicity that the minima does indeed exist. If this is not the case, our main result below also holds for any estimate which minimizes the empirical L_2 risk in (18) only up to some additional term of order $1/\sqrt{n}$.

3 Main result

Our main result is the following theorem, which gives an upper bound on the difference between the misclassification probability of the estimate introduced in Section 2 and the optimal misclassification probability.

Theorem 1. *Let $A > 0$. Let (X,Y), (X_1,Y_1), ..., (X_n,Y_n) be independent and identically distributed $[-A,A]^{d \cdot l} \times \{0,1\}$–valued random variables, and let $\eta(x) = \mathbf{P}\{Y = 1 | X = x\}$ be the corresponding aposterio probability. Let $\mathcal{P}$ be a finite subset of $(0,\infty) \times \mathbb{N}$ and assume that η satisfies a hierarchical composition model with some finite level and smoothness and order constraint $\mathcal{P}$. Set*

$$h = \max_{(p,K) \in \mathcal{P}} n^{K/(2p+K)}, I = \log n, L_n = (\log n)^2 \cdot (\max_{(p,K) \in \mathcal{P}} p) \cdot (\max_{(p,K) \in \mathcal{P}} K) \cdot \max_{(p,K) \in \mathcal{P}} n^{K/(2 \cdot p + K)},$$

choose $N \in \mathbb{N}$ sufficiently large, $d_K \geq 2$, $d_{ff} \geq 2 \cdot h + 2$, and define the estimate η_n by (18) and (19). Then we have for n sufficiently large

$$\mathbf{P}\{\eta_n(X) \neq Y\} - \min_{f : \mathbb{R}^{d \cdot l} \to \{0,1\}} \mathbf{P}\{f(X) \neq Y\} \leq c_1 \cdot (\log n)^3 \cdot \max_{(p,K) \in \mathcal{P}} n^{-\frac{p}{2p+K}}.$$

Remark 1. It follows from the proof of Theorem 1 that it also holds if we set $I = c_2$ for some sufficiently large $c_2 \in \mathbb{N}$, which depends on the level of the hierarchical composition model and on $\mathcal{P}$.

Remark 2. The structure of the Transformer encoder in Theorem 1, i.e., the multi-head attention and piecewise feedforward layers and the residual connection, is as it is proposed in Vaswani et al. (2017). But different from the model proposed there is our choice of the encoding of the input. It is an open problem whether a similar result as in Theorem 1 also holds if the coding of the input position is done as in Vaswani et al. (2017).

Remark 3. The rate of convergence in Theorem 1 does not depend on the dimension $d \cdot l$ of the predictor variable, hence the Transformer encoder is able to circumvent the curse of dimensionality in case that the aposteriori probability satisfies a hierarchical composition model with suitable order and smoothness constraints.

4 Classification problems in natural language processing

In this section, we describe some of the text classification datasets that have been traditionally used to evaluate neural models along with the typical NLP models to solve these tasks, including the Transformer based classifier. Then, we discuss the evaluation and the performance of the models on these datasets, and point out the differences between the Transformer classifiers theoretically analyzed in this paper and the ones used in practice. This gives us hints for useful generalizations of the theoretical results in this paper in future work.

4.1 Datasets

The datasets have been constructed in a previous study by Zhang, Zhao and LeCun (2015) to foster the progress in text classification tasks and to empirically show that convolutional neural networks can provide competitive results on such tasks.

YELP   contains reviews written by the customers about their experience in a location such as restaurants, bars and doctors. The dataset defines the classification task as predicting the number of stars ranging from 1 to 5 given the review text. It is a balanced dataset containing equal amount per label of training and testing samples (130K training, 10K testing). This is known as Yelp F. (Full) where the task is to predict the full stars. There is also a polarity classification version of this dataset, where 1 and 2 star reviews are merged under the Negative label, and 4 and 5 stars are labeled as Positive. This version is referred to as Yelp P. (Polarity).

DBPedia   is a crowdsourcing project that maps Wikipedia infoboxes to a shared ontology (e.g., hierarchical classes of objects) that contains around 320 classes. The dataset that is derived from DBPedia consists of titles and abstracts of Wikipedia articles and their corresponding classes in the ontology. The authors construct the dataset by randomly sampling 40K training and 5K test samples from 14 non-overlapping classes from the DBPedia 2014 release.

4.2 Models

We present three techniques to tackle the aforementioned multilabel classification problems: a strong baseline model using tfidf, the early state-of-the-art-model based on convolutional neural networks and the transformer based model.

4.2.1 Baseline

As a baseline, we choose the traditonal n-gram tf-idf model as in Zhang, Zhao and LeCun (2015). *n-gram* is one of the most common NLP terms that has been traditionally used to model text sequences. Here, n refers to the number of sequential text units (typically words) extracted from running text such as newspaper, wikipedia or web articles. In very simple terms, given the sentence "I love NLP", 1-grams that can be extracted from this

sentence are { 'I', 'love', 'NLP'}, while 2-grams are { 'I love', 'love NLP' }. This baseline starts with extracting the most frequent 500,000 n-grams from the training sets, where n is between 1 and 5. Then it defines TF (term-frequency) by

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{20}$$

Here $f_{t,d}$ is a raw count of the term (in this case a specific $n - gram$) that occurs in document d, which is then normalized by the number of times other terms (t') occur in the same document, i.e., by the number of different terms which occur in d. It can be interpreted as certain types of documents containing certain n-grams, such as a sports article having a higher tf for the 2-gram "football game". Furthermore it defines IDF (Inverse Document Frequency) by

$$idf(t,D) = log \frac{N}{|\{d \in D : t \in d\}|} \tag{21}$$

Here N is the total number of documents in the training set D. The denominator refers to the number of documents that contains the term t. It can therefore be interpreted as the amount of information a term provides. For instance the 1-gram "the" would have a low idf since it is a common word, while "football" would yield a higher idf since it occurs only on a subset of documents. Using these two concepts it introduces *tfidf* as follows:

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \tag{22}$$

The values from the above equation are then used as features to train a multinomial logistic regression model.

4.2.2 Character-level Convolutional Neural Networks

The overall architecture of the character CNN is given in Fig. 1. First step is the **quantization** step, which is also referred to as the *encoding* step, where characters are encoded as one-hot vectors, by marking the character to be represented with the value 1 in a zero-vector (e.g., $[0, 0, 1, 0, 0, .., 0]$). More formally, given an alphabet of k characters, each character is indexed and represented with a vector $c \in \mathbb{R}^k$, where the component corresponding to the number of the character in the alphabet is 1, and in all other components the value is 0. In Zhang, Zhao and LeCun (2015) an alphabet of 70 characters is defined which is a combination of 26-letter English alphabet, digits, punctuations and the space character.

1D Convolution : Given the sequence of characters $c_{1:n} = (c_1, ..., c_n)$, a one dimensional convolution can simply be defined as sliding a window of size k over the character sequence to apply a convoluton filter, a.k.a., kernel, to each window. More formally, given the filter $u = (u_1, ..., u_k) \in \mathbb{R}^{d \times k}$, and the window of characters $x_i = (c_i, c_{i+1}, ..., c_{i+k}) \in \mathbb{R}^{d \times k}$, the convolution is defined as a dot-product between x_i and u, i.e., by

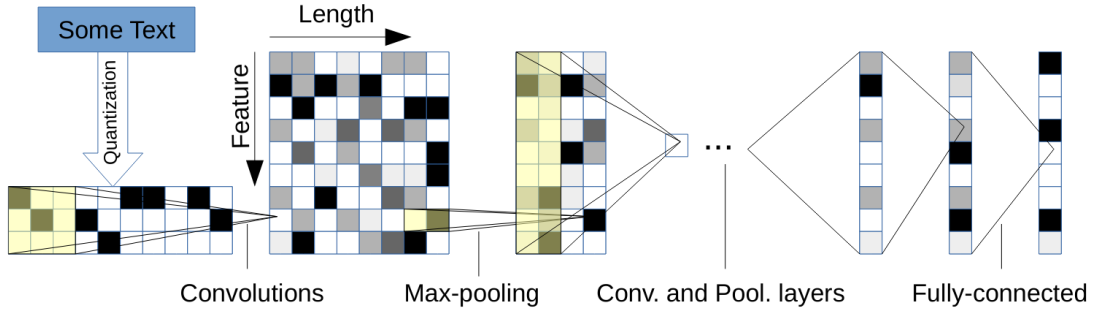$$u \cdot x_j = \sum_{j=1}^{k} u_j \cdot c_{i+j-1}.$$

Figure 1: Character-level CNN architecture for text classification taken from Zhang, Zhao and LeCun (2015).

In practice, however, we use a series of l filters $U_1, \ldots, U_l \in \mathbb{R}^{d \times k}$ and a bias term $b = (b_1, \ldots, b_l) \in \mathbb{R}^l$. The convolution is then followed by the nonlinear ReLU activation function $g(x) = max\{0, x\}$, i.e., the result of the convolution is

$$s_{i,j} = g(x_i \cdot U_j + b) \quad (i = 1, \ldots, n - k, j = 1, \ldots, l). \tag{23}$$

Max Pooling : This operation simply chooses the maximum of the values from all s_i vectors, which yields the fixed l dimensional vector $h = (h_1, \ldots, h_l)^T$ where

$$h_j = \max_{i=1,\ldots,n-k} s_{i,j}.$$

The sequence of convolution and max pooling operations are referred to as a **convolutional layer**.

The convolution layer is then followed by a linear layer with weight matrix, W, and a bias term b. The output of the linear layer is then fed to a softmax function to calculate the label probabilities for each class, l_i, as follows:

$$p(l_i) = softmax(W \cdot h + b). \tag{24}$$

The model is trained with back propagation algorithm using stochastic gradient descent to minimize the cross-entropy loss, L_{CE},

$$L_{CE} = -\sum_{i=1}^{n} t_i \cdot log(p(l_i)) \tag{25}$$

where n refers to the number of classes, and t is a one-hot vector of dimension n where the true label of the sample is flagged with 1 (e.g., if the test sample is from class 2, then $t = (t_1, \ldots, t_n)$ will be $(0, 1, 0, .., 0)$).

The final model consists of 6 convolutional layers which are responsible of extracting relevant textual features, which is then followed by 3 linear layers that capture the relation between the features and the classification labels.

4.2.3 Transformer based Language Models

One of the recent breakthroughs in natural language processing is introduction of pre-trained large language models, a.k.a., LLMs, that use the Transformer architecture such as Bidirectional Encoder Representations from Transformers (BERT), cf., Devlin et al. (2019). Such pretrained models also trigger a new paradigm for training high-level NLP tasks such as document classification, named entity recognition and question answering. This new training regime, called **fine-tuning**, allows the NLP practitioners and researchers to benefit from unlabeled text via performing small weight modifications to LLMs by adding a task-specific output layer and performing a minimal supervised re-training on the task data. Since BERT has been the predominantly used LLM, we experiment with BERT in this paper. Let us explain the architecture, training objective and the fine-tuning process in more details.

Architecture   The architecture of the BERT model is identical to the multi-layer bidirectional Transformer encoder described in Vaswani et al. (2017) and released in *tensor2tensor* library[3]. This means that it corresponds exactly to the Transformer encoder defined in Section 2.

Training Objective   Traditional language modeling objective has been defined as *predicting the next token given the previous context*. More formally, the objective has been to maximize the log probability $\log P(w_{i+1}|w_i, w_{i-1}, ..., w_1)$ where w_i denotes the word at index i. Later, researchers introduced a secondary objective which is to *predict the previous token given the future context*, i.e., to maximize $\log P(w_{i-1}|w_i, w_{i+1}, ..., w_n)$ where the maximum number of words is n. Traditionally, two different representations are learned using these two objectives which are then concatenated and used as the final token representation.

The calculation of this objective has been trivial when sequential neural models such as recurrent neural networks have been used. However, since Transformer architecture is not designed to be autoregressive, such bidirectional training objective has not been that intuitive. To address this gap, BERT uses a combination of different language modeling objectives. The first objective is called a **Masked Language Model (MLM)**. It is defined as predicting the tokens that are masked at random. To learn representations that can better model the relation between sentences, BERT defines the second self-explanatory objective: **Next Sentence Prediction (NSP)**.

In BERT, words are represented as a sequence of subwords using the WordPiece algorithm which is inspired from an old byte compression idea. The authors introduce a special token named [CLS] that is placed at the beginning of each sequence. The final hidden state of this token from the multi-layer bidirectional Transformer encoder, i.e., the hidden state of the pointwise feedforward neural network in (17) corresponding to this token, is considered as the sequence representation and used for the classification tasks. Since BERT aims to provide a unified architecture for NLP downstream tasks, including

---

[3]https://github.com/tensorflow/tensor2tensor

tasks that take a *pair* of inputs (e.g., question and answer), it introduces another special token called [SEP] to separate two sentences from each other. The input representation of BERT is then the sum of three embeddings: the token (or the WordPiece) embedding, E_{tok}, the segment embedding $E_{A|B}$ that identifies whether the sentence is the prior (segment A) or the next (segment B) sentence, and a special **Positional Embedding (PE)** using sinus and cosine wave frequencies defined as follows:

$$PE_{pos,2i} = sin\frac{pos}{10000^{\frac{2i}{d}}}, \qquad PE_{pos,2i+1} = cos\frac{pos}{10000^{\frac{2i}{d}}}, \qquad (26)$$

where d is the fixed dimension size of the input representation, i refers to the index in the d dimensional Positional Embedding (PE) and *pos* refers to the position of the token within the input. For instance the *pos* value of the token "I" in the input "I love NLP" would be 0, and "love" would be 1.

Pretraining   The model is pretrained with the two objectives, **Masked Language Model (MLM)** and **Next Sentence Prediction (NSP)**, that are mentioned above. To elaborate, for MLM, 15% of the WordPiece tokens are masked randomly. Later the hidden vectors for the masked tokens are passed onto a softmax layer that calculates the probability distribution over the full vocabulary. NSP is formulated as a binary classification problem, where [CLS] token is classified either as ISNEXT or NOTNEXT. It can be interpreted as given two sentences, the second sentence is either the *next* sentence following the previous one, or not. The training loss is then the sum of the mean MLM and NSP likelihood. The model is pretrained on a corpus of a large collection (in total 3,3 billion words) of books and wikipedia articles. It should be noted that, in practice, sequence pairs do not directly refer to linguistically well-formed sentence structures. For instance, a sequence might contain an unfinished sentence since it contains a fixed number of tokens and does not take sentence boundaries into account. They are typically longer spans of text which may contain multiple sentences. For training proper segment embeddings, training data is constructed in a way that, 50% of the time the second sentence follows the first sentence (i.e., has the label isNext) and 50% it is a randomly chosen sentence (i.e., has the label notNext).

Fine-tuning   Fine-tuning process, in general, aims to introduce a minimal amount of task-specific parameters which are used to define a task-specific output layer. This output layer depends on the NLP task. For instance, for token-level tasks such as Named Entity Recognition (NER) (i.e., labeling individual tokens as entity names such as organization, company and city name), token representations are fed into the output layer. In case of classification tasks such classifying restaurant reviews as in this paper, the final hidden state of the [CLS] token is fed into the output layer. Finally, either all parameters or only the task-specific parameters are updated end-to-end using the task-specific objective (e.g., binary cross-entropy loss for binary classification, categorical cross-entropy loss for multiple classes) on the task-specific labeled data (e.g., the classification datasets described in Sec. 4.1). An illustration of the pretraining and the fine-tuning processes is given in Fig. 2.
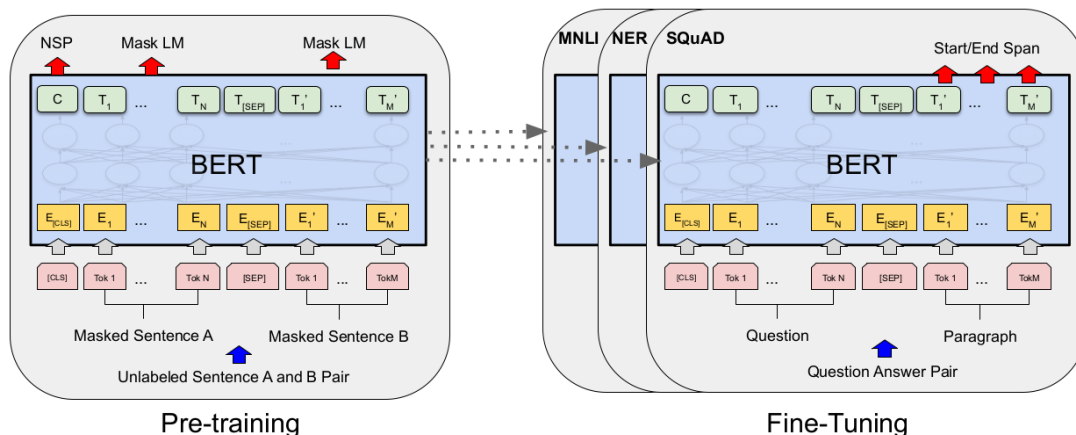
Figure 2: Illustration of pretraining and finetuning of BERT taken from Devlin et al. (2019). E refers to embedding, [CLS] and [SEP] denote special tokens designed to perform classification and to separate two sentences, accordingly. NSP denotes "next sentence prediction". MNLI (Multi-Genre Natural Language Inference), NER (Named Entity Recognition) and SQuAD (Stanford Question Answering Dataset) are popular NLP tasks for on which the BERT model is finetuned.

4.3 Evaluation and Results

All models are evaluated on the standard test splits as provided by the data created in Zhang, Zhao and LeCun (2015) using the empirical misclassification error on the testing data as the evaluation measure. We show the scores of the baseline as reported by Zhang, Zhao and LeCun (2015).

We finetune the BERT-base model that consists of 12 Transformer encoder blocks with hidden dimensionality of 768, and 12 self-attention heads. BERT outputs a representation of an input sequence including a representation for the special [CLS] token. The maximum number of tokens that can be processed by BERT is 512. However, since processing 512 tokens is quite memory demanding for a single GPU, we set the threshold to 128 tokens. We take the final hidden state h of the [CLS] token and introduce a task specific weight matrix W. The output is then fed to a softmax classifier to calculate the probability of class l by

$$p(l|h) = softmax(Wh). \tag{27}$$

BERT has been shown to capture lower-level linguistic information like sentence structure in lower layers, while higher-level layers capture higher-level information such as the sentence semantics. Therefore, instead of fine-tuning all layers, we fine-tune only the last two layers. This strategy does not only reduce the computational workload, but also has been shown to provide competitive scores, cf., Sun et al. (2019). Finally, we maximize the log-probability of the correct label by updating the parameters in W and the last

two layers. We fine-tune the uncased model (all lower case) for 3 epochs with the batch size as 32 and the learning rate as 0.00002. We train both the character-level CNN and the fine-tuned BERT models using five different random seeds and report the averaged scores in Table 1.

Our results show that, even a simple fine-tuning strategy on the large pretrained Transformer-based language model BERT provides substantial improvements over the baseline and the character-level CNNs.

Model	Yelp P.	Yelp F.	DBP
n-gram TFIDF (Zhang, Zhao and LeCun (2015))	4.56	45.20	1.31
Char CNN	4.96	39.25	1.78
BERT Fine-Tuning	2.92	32.94	0.75

Table 1: Comparison of test error rates of the baseline, character-level CNN and the finetuned BERT model.

4.4 Comparison of theory and practice

If we compare our classifier introduced in Section 2 with the ones used in practice, we see that the basic structure of the classifiers, i.e., the use of a sequence of pairs of mulithead attention and pointwise feedforward layers, is the same. Different are the coding of the input (here we used a separate part of the vector in order to encode the position instead of adding some kind of encoded wave to the vector) and the learning method. In particular, our theory totally ignores the idea of using a pretrained model as used in BERT. Instead we focused only on the approximation and generalization properties of Transformer networks in order to simplify the theory.

This indicates what the main question for future research in this area is: Can one also come up with a theory for Transformer classifiers which uses firstly pretraining and secondly learning of the classifiers by gradient descent?

5 Proofs

5.1 Proof of Theorem 1

Set $\beta = 1$. Since $T_\beta z \geq 1/2$ if and only if $z \geq 1/2$ for any $z \in \mathbb{R}$ we have

$$\eta_n(x) = \begin{cases} 1 & \text{if } T_\beta m_n(x) \geq 1/2, \\ 0 & \text{else.} \end{cases}$$

By Theorem 1.1 in Györfi et al. (2002) we know

$$\mathbf{P}\{\eta_n(X) \neq Y | \mathcal{D}_n\} - \min_{f:\mathbb{R}^{d \cdot l} \to \{0,1\}} \mathbf{P}\{f(X) \neq Y\} \leq 2 \cdot \sqrt{\int |T_\beta m_n(x) - \eta(x)|^2 \mathbf{P}_X(dx)},$$

hence it suffices to show

$$\mathbf{E} \int |T_\beta m_n(x) - \eta(x)|^2 \mathbf{P}_X(dx) \le c_3 \cdot (\log n)^6 \cdot \max_{(p,K) \in \mathcal{P}} n^{-\frac{2p}{2p+K}}. \tag{28}$$

By standard error bounds from empirical process theory (cf., e.g., Lemma 1 in Bauer and Kohler (2019)) we know

$$\mathbf{E} \int |T_\beta m_n(x) - \eta(x)|^2 \mathbf{P}_X(dx)$$

$$\le \frac{c_4 \cdot \left(\log n\right)^2 \cdot \sup_{x_1^n} \left(\log \left(\mathcal{N}_1 \left(\frac{1}{n \cdot \beta}, \{T_\beta f_{\bar\theta} \,:\, \|\bar\theta\|_0 \le L_n\} \right), x_1^n \right) \right) + 1 \right)}{n}$$

$$+ 2 \cdot \inf_{f \in \{f_{\bar\theta} \,:\, \|\bar\theta\|_0 \le L_n\}} \int |f(x) - \eta(x)|^2 \mathbf{P}_X(dx).$$

We will show in Lemma 8 below that we have

$$\sup_{x_1^n} \log \left(\mathcal{N}_1 \left(\frac{1}{n \cdot \beta}, \{T_\beta f_{\bar\theta} \,:\, \|\bar\theta\|_0 \le L_n\}, x_1^n \right) \right) \quad \le \quad c_5 \cdot L_n \cdot (\log n)^2$$

$$= \quad c_6 \cdot (\log n)^4 \cdot \max_{(p,K) \in \mathcal{P}} n^{K/(2 \cdot p + K)}.$$

And by Theorem 2 below we get

$$\inf_{f \in \{f_{\bar\theta} \,:\, \|\bar\theta\|_0 \le L_n\}} \int |f(x) - \eta(x)|^2 \mathbf{P}_X(dx) \le c_7 \cdot \max_{(p,K) \in \mathcal{P}} h^{-2 \cdot p/K}.$$

Choose $(\bar p, \bar K) \in \mathcal{P}$ with

$$\frac{\bar p}{\bar K} = \min_{(p,K) \in \mathcal{P}} \frac{p}{K}.$$

Then the above results imply

$$\mathbf{E} \int |T_\beta m_n(x) - \eta(x)|^2 \mathbf{P}_X(dx) \quad \le \quad \frac{c_6 \cdot (\log n)^6 \cdot n^{\bar K/(2 \cdot \bar p + \bar K)}}{n} + c_7 \cdot h^{-2 \cdot \bar p / \bar K}$$

$$\le \quad c_8 \cdot (\log n)^6 \cdot n^{-2 \cdot \bar p/(2 \cdot \bar p + \bar K)}$$

$$= \quad c_8 \cdot (\log n)^6 \cdot \max_{(p,K) \in \mathcal{P}} n^{-\frac{2p}{2p+K}}.$$

$\square$

5.2 Approximation of piecewise polynomials by transformer networks with single-head attention

In this subsection we present results concerning the approximation of piecewise polynomials by Transformer encoders with single-head attention. In the next subsection we will generalize these results to Transformer encoders with multi-head attention.

In case of a Transformer encoder with a single-head attention we represent the input sequence (4) by

$$z_0 = (z_{0,1}, \dots, z_{0,l}) \in \mathbb{R}^{l \cdot d_{model}},$$

where

$$d_{model} = d + l + 4. \tag{29}$$

Here we use d components to represent the original data, we use l components to represent the position in the original sequence, we use 1 component to have available the constant 1 in our computations, and we use 3 components to be able to save auxiliary results during our computation. More precisely we set

$$z_{0,j}^{(s)} = \begin{cases} x_j^{(s)} & \text{if } s \in \{1, \dots, d\} \\ 1 & \text{if } s = d+1 \\ \delta_{s-d-1,j} & \text{if } s \in \{d+2, \dots, d+1+l\} \\ 1 & \text{if } s = d+l+3 \\ 0 & \text{if } s \in \{d+l+2, d+l+4\} \end{cases}$$

We start with a first lemma that shows that a single-head attention unit can be used to compute linear polynomials in one variable.

Lemma 1. *Let $x_j \in \mathbb{R}^d$ and $b_j \in \mathbb{R}$ $(j = 1, \dots, l)$. Let $z_0 = (z_{0,1}, \dots, z_{0,l}) \in \mathbb{R}^d$ be given by*

$$z_{0,j}^{(s)} = \begin{cases} x_j^{(s)} & \text{if } s \in \{1, \dots, d\} \\ 1 & \text{if } s = d+1 \\ \delta_{s-d-1,j} & \text{if } s \in \{d+2, \dots, d+1+l\} \\ b_j & \text{if } s = d+l+3 \\ 0 & \text{if } s \in \{d+l+2, d+l+4\} \end{cases}$$

Let $j \in \{1, \dots, l\}$, $k \in \{1, \dots, d\}$ and $u \in \mathbb{R}$ be arbitrary. Let

$$B > 2 \cdot \max_{s=1,\dots,d, j=1,\dots,l} |x_j^{(s)}|.$$

Then there exists matrices $W_Q, W_K \in \mathbb{R}^{2 \times d_{model}}$ and $W_V \in \mathbb{R}^{d_{model} \times d_{model}}$, where each matrix contains at most 3 nonzero entries and where all entries depend only on u and B, such that $q_i = W_Q \cdot z_{0,i}$, $k_i = W_K \cdot z_{0,i}$, $v_i = W_V \cdot z_{0,i}$,

$$\hat{j}_i = \arg\max_{t \in \{1,\dots,l\}} <q_i, k_t> \quad (i \in \{1, \dots, l\})$$

and

$$y_i = z_{0,i} + v_{\hat{j}_i} \cdot <q_i, k_{\hat{j}_i}> \quad (i \in \{1, \dots, l\})$$

result in

$$y_1^{(s)} = \begin{cases} x_1^{(s)} & \text{if } s \in \{1, \ldots, d\} \\ 1 & \text{if } s = d+1 \\ \delta_{s-d-1,j} & \text{if } s \in \{d+2, \ldots, d+1+l\} \\ x_j^{(k)} - u & \text{if } s = d+l+2 \\ b_1 & \text{if } s = d+l+3 \\ 0 & \text{if } s = d+l+4 \end{cases}$$

and

$$y_j = z_{0,j} \quad for \ j \in \{2, \ldots, l\}.$$

Proof. We start our proof by defining the matrices W_Q, W_K and W_V. We set

$$W_Q = \begin{pmatrix} 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & B & 0 & \ldots & 0 \end{pmatrix}$$

where all colums are zero except column number $d+2$,

$$W_K = \begin{pmatrix} 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & -u-B & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{pmatrix}$$

where all colums are zero except column number k, column number $d+1$ and column number $d+1+j$, and

$$W_V = \begin{pmatrix} 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \ldots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \ldots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

where all rows and all colums are zero except row number $d+l+2$ and column number $d+1$. For these matrices we get

$$q_i = W_Q \cdot z_{0,i} = \begin{pmatrix} 1 \cdot \delta_{i,1} \\ B \cdot \delta_{i,1} \end{pmatrix}, \quad k_i = W_K \cdot z_{0,i} = \begin{pmatrix} x_i^{(k)} - u - B \\ 1 \cdot \delta_{i,j} \end{pmatrix},$$

$$v_i = W_V \cdot z_{0,i} = (\delta_{1,d+l+2}, \delta_{2,d+l+2}, \ldots, \delta_{d+l+4,d+l+2})^T = e_{d+l+2},$$

where e_{d+l+2} is the $d+l+2$-th unit vector in $\mathbb{R}^{d+l+4}$. Consequently we have

$$< q_r, k_s > = (x_s^{(k)} - u - B) \cdot \delta_{r,1} + B \cdot \delta_{r,1} \cdot \delta_{s,j},$$

$$< q_1, k_j > = (x_j^{(k)} - u - B) + B > x_s^{(k)} - u - B = < q_1, k_s >$$

18

for all $s \in \{1, \ldots, d\} \setminus \{j\}$ and

$$\hat{j}_1 = j,$$

which implies

$$v_{\hat{j}_1} \cdot < q_1, k_{\hat{j}_1} >= (x_j^{(k)} - u) \cdot e_{d+l+2}$$

and

$$v_{\hat{j}_i} \cdot < q_i, k_{\hat{j}_i} >= e_{d+l+2} \cdot < 0, k_{\hat{j}_i} >= 0 \cdot e_{d+l+2}$$

for $i > 1$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Remark 4. It follows from the proof of Lemma 1 that we can modify W_K such that $y_1^{(d+l+2)} = 1$ holds instead of $y_1^{(d+l+2)} = x_j^{(k)} - u$.

Our next lemma shows that a single-head attention unit can be used to compute products.

Lemma 2. Let $x_s \in \mathbb{R}^d$ and $a_s, b_s \in \mathbb{R}$ $(s = 1, \ldots, l)$. Let $z_0 = (z_{0,1}, \ldots, z_{0,l}) \in \mathbb{R}^d$ be given by

$$z_{0,j}^{(s)} = \begin{cases} x_j^{(s)} & \text{if } s \in \{1, \ldots, d\} \\ 1 & \text{if } s = d+1 \\ \delta_{s-d-1,j} & \text{if } s \in \{d+2, \ldots, d+1+l\} \\ a_j & \text{if } s = d+l+2 \\ b_j & \text{if } s = d+l+3 \\ 0 & \text{if } s = d+l+4 \end{cases}$$

Let $j \in \{1, \ldots, l\}$. Let $B > 2 \cdot \max_{r,s} |a_r \cdot b_s|$. Then there exists matrices W_Q, $W_K \in \mathbb{R}^{2 \times d_{model}}$ and $W_V \in \mathbb{R}^{d_{model} \times d_{model}}$, where each matrix contains at most 2 nonzero entries and where all entries depend only on B, such that $q_i = W_Q \cdot z_{0,i}$, $k_i = W_K \cdot z_{0,i}$, $v_i = W_V \cdot z_{0,i}$,

$$\hat{j}_i = \arg\max_{t \in \{1, \ldots, l\}} < q_i, k_t > \quad (i \in \{1, \ldots, l\})$$

and

$$y_i = z_{0,i} + v_{\hat{j}_i} \cdot < q_i, k_{\hat{j}_i} > \quad (i \in \{1, \ldots, l\})$$

result in

$$y_1^{(s)} = \begin{cases} x_1^{(s)} & \text{if } s \in \{1, \ldots, d\} \\ 1 & \text{if } s = d+1 \\ \delta_{s-d-1,j} & \text{if } s \in \{d+2, \ldots, d+1+l\} \\ a_1 & \text{if } s = d+l+2 \\ b_1 & \text{if } s = d+l+3 \\ b_1 \cdot a_j + B & \text{if } s = d+l+4 \end{cases}$$

and

$$y_s^{(i)} = z_{0,s}^{(i)} \quad \text{for } i \in \{1, \ldots, d+l+3\}, s \in \{1, \ldots, l\}.$$

Proof. Define W_V as in the proof of Lemma 1 such that all rows and all colums are zero except row number $d+l+4$ and column number $d+1$, and set

$$W_Q = \begin{pmatrix} 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & B & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

where all colums are zero except column number $d+2$ and column number $d+l+3$ and

$$W_K = \begin{pmatrix} 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

where all colums are zero except column number $d+1+j$ and column number $d+l+2$.

For these matrices we get

$$q_i = W_Q \cdot z_{0,i} = \begin{pmatrix} b_i \\ B \cdot \delta_{i,1} \end{pmatrix}, \quad k_i = W_K \cdot z_{0,i} = \begin{pmatrix} a_i \\ 1 \cdot \delta_{i,j} \end{pmatrix},$$

$$v_i = W_V \cdot z_{0,i} = (0, \ldots, 0, 1)^T = e_{d+l+4}.$$

Consequently we have

$$< q_r, k_s > = b_r \cdot a_s + B \cdot \delta_{r,1} \cdot \delta_{s,j}.$$

This implies

$$\hat{j}_1 = j$$

and

$$< q_i, k_{\hat{j}_i} > = \max_s b_i \cdot a_s$$

for $i > 1$, from which we can conclude

$$v_{\hat{j}_1} \cdot < q_1, k_{\hat{j}_1} > = (b_1 \cdot a_j + B) \cdot e_{d+l+4}$$

and

$$v_{\hat{j}_i} \cdot < q_i, k_{\hat{j}_i} > = (\max_s b_i \cdot a_s) \cdot e_{d+l+4}$$

for $i > 1$. $\qquad\square$

Our next lemma defines a special pointwise feedforward neural network, which applies the function

$$x \mapsto \alpha \cdot (x - B)$$

to component number $d+l+4$ of each element in the decoding of the sequence of inputs and writes the result in component number $d+l+3$, and which sets the entries in components $d+l+2$ and $d+l+4$ to zero.

Lemma 3. *Let $d_{model} = d+l+4$ and let $y = (y_1, \ldots, y_l)$ for some $y_i \in \mathbb{R}^{d_{model}}$. Let $d_{ff} \geq 8$ and let $\alpha \in \mathbb{R}$. Then there exists matrices and vectors*

$$W_1 \in \mathbb{R}^{d_{ff} \times d_{model}}, b_1 \in \mathbb{R}^{d_{ff}}, W_2 \in \mathbb{R}^{d_{model} \times d_{ff}}, b_2 \in \mathbb{R}^{d_{model}},$$

which depend only on B and α and which have at most 10 nonzero entries such that

$$z_s = y_s + W_2 \cdot \sigma\left(W_1 \cdot y_s + b_1\right) + b_2$$

results in

$$z_s^{(i)} = \begin{cases} y_s^{(i)} & \text{if } i \in \{1, \ldots, d+l+1\}, \\ \alpha \cdot (y_s^{(d+l+4)} - B) & \text{if } i = d+l+3, \\ 0 & \text{if } i \in \{d+l+2, d+l+4\}. \end{cases}$$

Proof. W.l.o.g. we assume $d_{ff} = 8$. We choose $b_1 = 0$, $b_2 = 0$,

$$W_1 = \begin{pmatrix} 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 1 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & -1 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 1 & 0 \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & -1 & 0 \\ 0 & \ldots & 0 & -B & 0 & \ldots & 0 & 0 & 0 & 1 \\ 0 & \ldots & 0 & B & 0 & \ldots & 0 & 0 & 0 & -1 \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 1 \\ 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & -1 \end{pmatrix},$$

where all columns except columns number $d+1$, $d+l+2$, $d+l+3$ and $d+l+4$ are zero, and

$$W_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & \alpha & -\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix},$$

where all rows except row number $d+l+2$, $d+l+3$ and $d+l+4$ are zero. Then we have

$$W_2 \cdot \sigma\left(W_1 \cdot y_s + b_1\right) + b_2$$
$$= \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -(\sigma(y_s^{(d+l+2)}) - \sigma(-y_s^{(d+l+2)})) \\ -(\sigma(y_s^{(d+l+3)}) - \sigma(-y_s^{(d+l+3)})) + \alpha \cdot \sigma(y_s^{(d+l+4)} - B) - \alpha \cdot \sigma(B - y_s^{(d+l+4)}) \\ -(\sigma(y_s^{(d+l+4)}) - \sigma(-y_s^{(d+l+4)})) \end{pmatrix}.$$

Because of

$$\sigma(u) - \sigma(-u) = u$$

for $u \in \mathbb{R}$ this implies the assertion. $\square$

Lemma 4. *Let $d_{model} = d + l + 4$ and let $y = (y_1, \ldots, y_l)$ for some $y_i \in \mathbb{R}^{d_{model}}$. Let $d_{ff} \geq 3$ and let $\alpha \in \mathbb{R}$. Then there exists matrices and vectors*

$$W_1 \in \mathbb{R}^{d_{ff} \times d_{model}}, b_1 \in \mathbb{R}^{d_{ff}}, W_2 \in \mathbb{R}^{d_{model} \times d_{ff}}, b_2 \in \mathbb{R}^{d_{model}},$$

which have at most three nonzero entries such that

$$z_s = y_s + W_2 \cdot \sigma\left(W_1 \cdot y_s + b_1\right) + b_2$$

results in

$$z_s^{(i)} = \begin{cases} y_s^{(i)} & \text{if } i \in \{1, \ldots, d+l+4\} \setminus \{d+l+2\}, \\ \max\{y_s^{(d+l+4)}, 0\} & \text{if } i = d+l+2 \end{cases}$$

($s \in \{1, \ldots, l\}$).

Proof. Follows as in the proof of Lemma 3 by choosing W_2, b_2, W_1 and b_1 such that

$$W_2 \cdot \sigma\left(W_1 \cdot y_s + b_1\right) + b_2 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -(\sigma(y_s^{(d+l+2)}) - \sigma(-y_s^{(d+l+2)})) + \sigma(y_s^{(d+l+4)}) \\ 0 \\ 0 \end{pmatrix}$$

holds. $\qquad\square$

Remark 5. Assume $d_{ff} \geq 4$. It follows from the proof of Lemma 4 that we can modify W_2 such that at most four of its entries are nonzero and $z_s^{(d+l+2)} = y_s^{(d+l+4)}$ holds instead of $z_s^{(d+l+2)} = \max\{y_s^{(d+l+4)}, 0\}$.

Next we combine the previous results to construct a Transformer encoder which uses $2 \cdot M \cdot d$ pairs of single-attention layers and piecewise feedforward layers in order to compute a basis function of a tensor product spline space of degree M.

Lemma 5. *Let $K \in \mathbb{N}$, let $u_k \in \mathbb{R}$ ($k = 1, \ldots, K-1$), set*

$$B_j(x) = x^j \quad \text{for } j = 0, 1, \ldots, M$$

and set

$$B_j(x) = (x - u_{j-M})_+^M \quad \text{for } j = M+1, M+2, \ldots, M+K-1.$$

Let $j_1, \ldots, j_d \in \{0, 1, \ldots, M+K-1\}$ and $\alpha \in \mathbb{R}$. Then there exists a transformer network consisting of $2 \cdot M \cdot d$ pairs of layers layers, where in each pair the first layer is a single-attention layer and the second layer is a piecewise feedforward neural network, where all matrices and vectors have at most 10 nonzero entries and where all matrices

and vectors depend only on $(u_k)_k$, B and α, which gets as input z as in Lemma 1 (with $b_j = 1$ ($j = 1, \ldots, d$)) and produces as output $z_{M \cdot d}$ which satisfies

$$z_{M \cdot d}^{(d+l+3)} = \alpha \prod_{k=1}^{d} B_{j_k}(x^{(k)}).$$

Proof. Each $B_j(x)$ can be written as

$$B_j(x) = \prod_{k=1}^{M} B_{j,k}(x)$$

where $B_{j,k}(x)$ is one of the functions

$$x \mapsto 1, \quad x \mapsto x \quad \text{and} \quad x \mapsto (x - u_r)_+.$$

Using Lemma 1, Lemma 2, Remark 4, Lemma 3, Lemma 4, Remark 5 and Lemma 5 we can combine two pairs of attention layers and piecewise feedforward layers such that they produce from the input $z_{0,j}$ given as in Lemma 1 with $b_j \geq 0$ an output y_j where $y_1^{(d+l+3)}$ is the product of b_j and one of the functions

$$x^{(k)} \mapsto 1, \quad x^{(k)} \mapsto x^{(k)} \quad \text{and} \quad x^{(k)} \mapsto (x^{(k)} - u_r)_+$$

and where $y_j^{(s)}$ is equal to $z_{0,j}^{(s)}$ otherwise. Using this repeatedly we get the assertion. $\square$

5.3 Approximation of piecewise polynomials by transformer networks with multi-head attention

In this subsection we generalize the results from the previous subsection to Transformer encoders with multi-head attention. The basic idea is to extend the coding of the input by repeating the previous coding h times and to define the multi-head attention such that each attention unit makes the computation of the previous subsection in one of the copies of the original coding.

We use the following coding of the original input: We represent the input sequence (4) by

$$z_0 = (z_{0,1}, \ldots, z_{0,l}) \in \mathbb{R}^{l \cdot d_{model}}, \tag{30}$$

where

$$d_{model} = h \cdot (d + l + 4). \tag{31}$$

Here we repeat the coding of the previous subsection h times. More precisely we set for all $k \in \{1, \ldots, h\}$

$$z_{0,j}^{((k-1) \cdot (d+l+4)+s)} = \begin{cases} x_j^{(s)} & \text{if } s \in \{1, \ldots, d\} \\ 1 & \text{if } s = d+1 \\ \delta_{s-d-1,j} & \text{if } s \in \{d+2, \ldots, d+1+l\} \\ 1 & \text{if } s = d+l+3 \\ 0 & \text{if } s \in \{d+l+2, d+l+4\} \end{cases}$$

In our next result we compute h basic functions of the truncated power basis in parallel by using a multi-head attention with h attention heads.

Lemma 6. *Let K, u_k, M and B_j be as in Lemma 5. Let $d_k \geq 2$ and $d_{ff} \geq 10$. Let $h \in \mathbb{N}$ and for $s \in \{1, \ldots, h\}$ let $j_{s,1}, \ldots, j_{s,d} \in \{0, 1, \ldots, M + K - 1\}$ and $\alpha_s \in \mathbb{R}$. Then there exists a Transformer encoder consisting of $2 \cdot M \cdot d$ pairs of layers, where the first layer is a multi-head attention layer with h attention units and the second layer is a piecewise feedforward neural network, and where all matrices and vectors have at most 10 nonzero entries and where all matrices and vectors depend only on $(u_k)_k$, B and α_s ($s \in \{1, \ldots, k\}$), which gets as input z_0 defined in (30) and produces as output $z_{M \cdot d}$ which satisfies*

$$z_{M \cdot d, 1}^{(s-1) \cdot (d+l+4) + (d+l+3)} = \alpha_s \prod_{k=1}^{d} B_{j_{s,k}}(x^{(k)})$$

for all $s \in \{1, \ldots, h\}$.

Proof. The result is a straightforward extension of the proof of Lemma 5. The basic idea is as follows. Each attention head of the network works only on one of the h copies of the coding of the previous subsection. In each attention unit it makes the same computations as in the proof of Lemma 5, using only its special part of the coding of the input. $\qquad\square$

Lemma 7. *Let K, u_k, M and B_j be as in Lemma 5. Let $h \in \mathbb{N}$ and for $s \in \{1, \ldots, h\}$ let $j_{s,1}, \ldots, j_{s,d} \in \{0, 1, \ldots, M + K - 1\}$ and $\alpha_s \in \mathbb{R}$. Let $d_k \geq 2$ and $d_{ff} \geq \max\{10, 2 \cdot h + 2\}$. Then there exists a Transformer encoder consisting of $2 \cdot M \cdot d + 1$ pairs of layers, where the first layer is a multi-head attention layer with h attention units and the second layer is a piecewise feedforward neural network, and where all matrices and vectors in the first $2 \cdot M \cdot d$ pairs of layers have at most 10 nonzero entries, where the matrices and vectors in the $2 \cdot M \cdot d + 1$-th pair of layers have together at most $2 \cdot h + 2$ nonzero components and where all matrices and vectors depend only on $(u_k)_k$, B and α_s ($s \in \{1, \ldots, k\}$), which gets as input z_0 defined in (30) and produces as output $z_{M \cdot d + 1}$ which satisfies*

$$z_{M \cdot d + 1, 1}^{h \cdot (d+l+4)} = \sum_{s=1}^{h} \alpha_s \prod_{k=1}^{d} B_{j_{s,k}}(x^{(k)}).$$

Proof. We use the construction of the proof of Lemma 6 to define the first $2 \cdot M \cdot d$ pairs of layers. After that we choose $W_{V,M \cdot d} = 0$ (which results in $y_{M \cdot d + 1} = z_{M \cdot d}$) and choose $W_{1,M \cdot d + 1}$, $b_{1,M \cdot d + 1}$, $W_{2,M \cdot d + 1}$, $b_{2,M \cdot d + 1}$, such that

$W_{2,M \cdot d + 1} \cdot \sigma \left(W_{1,M \cdot d + 1} \cdot y_s + b_{1,M \cdot d + 1} \right) + b_{2,M \cdot d + 1}$

$$= \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -(\sigma(y_{M \cdot d + 1, 1}^{h \cdot (d+l+4)}) - \sigma(-y_{M \cdot d + 1, 1}^{h \cdot (d+l+h)})) + \sigma(\sum_{s=1}^{h} y_{M \cdot d + 1, 1}^{s \cdot (d+l+4)}) - \sigma(-\sum_{s=1}^{h} y_{M \cdot d + 1, 1}^{s \cdot (d+l+4)}) \end{pmatrix}$$

holds. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Remark 6. The transformer network in Lemma 7 has

$$N \cdot h \cdot (2 \cdot d_k \cdot d_{model} + d_v \cdot d_{model} + 2 \cdot d_{ff} \cdot d_{model} + d_{ff} + d_{model})$$
$$= (2 \cdot M \cdot d + 1) \cdot h \cdot (2 \cdot d_k \cdot h \cdot (d + l + 4) + (d + l + 4) \cdot h \cdot (d + l + 4)$$
$$+ 2 \cdot d_{ff} \cdot h \cdot (d + l + 4) + d_{ff} + h \cdot (d + l + 4))$$
$$\leq 235 \cdot M \cdot (\max\{l, d, d_k, d_{ff}\})^3 \cdot h^2$$

many parameters, and of these parameters at most $144 \cdot M \cdot d \cdot h$ are not equal to zero. Here the positions are fixed where nonzero parameters are allowed to appear.

Next we show how we can approximate a function which satisfies a hierarchical composition model by a Transformer encoder. In order to formulate this result, we introduce some additional notation. In order to compute a function $h_1^{(\kappa)} \in \mathcal{H}(\kappa, \mathcal{P})$ one has to compute different hierarchical composition models of some level i ($i \in \{1, \ldots, \kappa - 1\}$). Let $\tilde{N}_i$ denote the number of hierarchical composition models of level i, needed to compute $h_1^{(\kappa)}$. Let

$$h_j^{(i)} : \mathbb{R}^d \to \mathbb{R} \tag{32}$$

be the j–th hierarchical composition model of some level i ($j \in \{1, \ldots, \tilde{N}_i\}, i \in \{1, \ldots, \kappa\}$), that applies a $(p_j^{(i)}, C)$–smooth function $g_j^{(i)} : \mathbb{R}^{K_j^{(i)}} \to \mathbb{R}$ with $p_j^{(i)} = q_j^{(i)} + s_j^{(i)}$, $q_j^{(i)} \in \mathbb{N}_0$ and $s_j^{(i)} \in (0, 1]$, where $(p_j^{(i)}, K_j^{(i)}) \in \mathcal{P}$. With this notation we can describe the computation of $h_1^{(\kappa)}(\mathbf{x})$ recursively as follows:

$$h_j^{(i)}(\mathbf{x}) = g_j^{(i)}\left(h_{\sum_{t=1}^{j-1} K_t^{(i)}+1}^{(i-1)}(\mathbf{x}), \ldots, h_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right) \tag{33}$$

for $j \in \{1, \ldots, \tilde{N}_i\}$ and $i \in \{2, \ldots, \kappa\}$, and

$$h_j^{(1)}(\mathbf{x}) = g_j^{(1)}\left(x^{\left(\pi(\sum_{t=1}^{j-1} K_t^{(1)}+1)\right)}, \ldots, x^{\left(\pi(\sum_{t=1}^{j} K_t^{(1)})\right)}\right) \tag{34}$$

holds for $j \in \{1, \ldots, \tilde{N}_1\}$ for some function $\pi : \{1, \ldots, \tilde{N}_1\} \to \{1, \ldots, d\}$. Here the recursion

$$\tilde{N}_l = 1 \text{ and } \tilde{N}_i = \sum_{j=1}^{\tilde{N}_{i+1}} K_j^{(i+1)} \quad (i \in \{1, \ldots, \kappa - 1\}) \tag{35}$$

holds.

Theorem 2. *Let $m : \mathbb{R}^d \to \mathbb{R}$ be contained in the class $\mathcal{H}(\kappa, \mathcal{P})$ for some $\kappa \in \mathbb{N}$ and $\mathcal{P} \subseteq [1, \infty) \times \mathbb{N}$. Let $\tilde{N}_i$ be defined as in (35). Each m consists of different functions*

$h_j^{(i)}$ ($j \in \{1, \dots, \tilde{N}_i\}$, $i \in \{1, \dots, \kappa\}$) *defined as in* (32), (33) *and* (34). *Assume that the corresponding functions* $g_j^{(i)}$ *are Lipschitz continuous with Lipschitz constant* $C_{Lip} \geq 1$ *and satisfy*

$$\|g_j^{(i)}\|_{C^{q_j^{(i)}}(\mathbb{R}^d)} \leq c_9$$

for some constant $c_9 > 0$. *Denote by* $K_{max} = \max_{i,j} K_j^{(i)} < \infty$ *the maximal input dimension and set* $q_{max} = \max_{i,j} q_j^{(i)} < \infty$, *where* $q_j^{(i)}$ *is the integer part of the smoothness* $p_j^{(i)}$ *of* $g_j^{(i)}$. *Let* $A \geq 1$. *Choose* $h \in \mathbb{N}$ *such that*

$$h \geq c_{10} \tag{36}$$

holds for some sufficiently large constant c_{10}, *choose*

$$I \geq \sum_{i=1}^{\kappa} \tilde{N}_i \quad and \quad d_k \geq 2$$

and set

$$N = I \cdot (2 \cdot q_{max} \cdot K_{max} + 1), \quad d_{model} = d_v = h \cdot I \cdot (d + l + 4)$$

and

$$L_n = 144 \cdot (q_{max} + 1) \cdot K_{max} \cdot I \cdot h.$$

Then there exists a transformer network $f_{\bar{\theta}}$ *with* $\|\bar{\theta}\|_0 \leq L_n$ *which satisfies*

$$\|f_{\bar{\theta}} - m\|_{\infty, [-A,A]^d} \leq c_{11} \cdot (K_{max} + 1)^{\kappa} \cdot \max_{j,i} h^{-p_j^{(i)}/K_j^{(i)}}.$$

Proof. From the Lipschitz continuity of the $g_j^{(i)}$ and the recursive definition of the $h_j^{(i)}$ we can conclude that there exists $\bar{A} \geq A$ such that

$$h_j^{(i)}(x) \in [-\bar{A}, \bar{A}] \tag{37}$$

holds for all $x \in [-A, A]^d$, $j \in \{1, \dots, \tilde{N}_i\}$ and $i \in \{1, \dots, \kappa - 1\}$.

Our Transformer encoder successively approximates $h_1^{(1)}(x)$, ..., $h_{N_1}^{(1)}(x)$, $h_1^{(2)}(x)$, ..., $h_{N_2}^{(2)}(x)$, ..., $h_1^{(\kappa)}(x)$. Here $h_i^{(j)}$ is approximated by computing in a first step truncated power basis of a tensorproduct spline space of degree $q_i^{(j)}$ on a equidistant grid in

$$[-\bar{A} - 1, \bar{A} + 1]^{K_i^{(j)}}$$

consisting of h basis functions, which are evaluated at the arguments of $h_i^{(j)}$ in (34), and by using in a second step a linear combination of these basis functions to approximate

$$g_j^{(i)}\left(h_{\sum_{t=1}^{j-1} K_t^{(i)} + 1}^{(i-1)}(\mathbf{x}), \dots, h_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right).$$

The computation of this truncated power basis can be done as in Lemma 7 using layers $(\tilde{N}_{i-1}+j-1)\cdot(2\cdot q_{max}\cdot K_{max}+1)+1$ till $(\tilde{N}_{i-1}+j)\cdot(2\cdot q_{max}\cdot K_{max}+1)$ of our Transformer encoder and proceeding otherwise as in Lemma 6. Using standard approximation results from spline theory (cf., e.g., Theorem 15.2 and proof of Theorem 15.1 in Györfi et al. (2002) and Lemma 1 in Kohler (2014)) this results in an approximation

$$\tilde{g}_j^{(i)}$$

of $g_j^{(i)}$ which satisfies

$$\|\tilde{g}_j^{(i)} - g_j^{(i)}\|_{\infty,[-\bar{A}-1,\bar{A}+1]^{K_j^{(i)}}} \leq c_{12}\cdot h^{-p_j^{(i)}/K_j^{(i)}}. \tag{38}$$

The approximation $\tilde{h}_1^{(\kappa)}(\mathbf{x})$ of $h_1^{(\kappa)}(\mathbf{x})$ which our Transformer encoder computes is defined as follows:

$$\tilde{h}_j^{(1)}(\mathbf{x}) = \tilde{g}_j^{(1)}\left(x^{\left(\pi(\sum_{t=1}^{j-1} K_t^{(1)}+1)\right)}, \ldots, x^{\left(\pi(\sum_{t=1}^{j} K_t^{(1)})\right)}\right)$$

for $j \in \{1, \ldots, \tilde{N}_1\}$ and

$$\tilde{h}_j^{(i)}(\mathbf{x}) = \tilde{g}_j^{(i)}\left(\tilde{h}_{\sum_{t=1}^{j-1} K_t^{(i)}+1}^{(i-1)}(\mathbf{x}), \ldots, \tilde{h}_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right)$$

for $j \in \{1, \ldots, \tilde{N}_i\}$ and $i \in \{2, \ldots, \kappa\}$.

From (36), (37) and (38) we can conclude

$$|\tilde{h}_j^{(i)}(\mathbf{x})| \leq |\tilde{h}_j^{(i)}(\mathbf{x}) - h_j^{(i)}(\mathbf{x})| + |h_j^{(i)}(\mathbf{x})| \leq \bar{A}+1.$$

Consequently we get from (38)

$$|\tilde{h}_j^{(i)}(\mathbf{x}) - h_j^{(i)}(\mathbf{x})|$$
$$\leq |\tilde{g}_j^{(i)}\left(\tilde{h}_{\sum_{t=1}^{j-1} K_t^{(i)}+1}^{(i-1)}(\mathbf{x}), \ldots, \tilde{h}_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right) - g_j^{(i)}\left(\tilde{h}_{\sum_{t=1}^{j-1} K_t^{(i)}+1}^{(i-1)}(\mathbf{x}), \ldots, \tilde{h}_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right)|$$
$$\quad + |g_j^{(i)}\left(\tilde{h}_{\sum_{t=1}^{j-1} K_t^{(i)}+1}^{(i-1)}(\mathbf{x}), \ldots, \tilde{h}_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right) - g_j^{(i)}\left(h_{\sum_{t=1}^{j-1} K_t^{(i)}+1}^{(i-1)}(\mathbf{x}), \ldots, h_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right)|$$
$$\leq c_{12}\cdot h^{-p_j^{(i)}/K_j^{(i)}} +$$
$$\quad |g_j^{(i)}\left(\tilde{h}_{\sum_{t=1}^{j-1} K_t^{(i)}+1}^{(i-1)}(\mathbf{x}), \ldots, \tilde{h}_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right) - g_j^{(i)}\left(h_{\sum_{t=1}^{j-1} K_t^{(i)}+1}^{(i-1)}(\mathbf{x}), \ldots, h_{\sum_{t=1}^{j} K_t^{(i)}}^{(i-1)}(\mathbf{x})\right)|$$
$$\leq c_{12}\cdot h^{-p_j^{(i)}/K_j^{(i)}} + c_{13}\cdot \sum_{s=1}^{K_j^{(i)}} |\tilde{h}_{\sum_{t=1}^{j-1} K_t^{(i)}+s}^{(i-1)}(\mathbf{x}) - h_{\sum_{t=1}^{j-1} K_t^{(i)}+s}^{(i-1)}(\mathbf{x})|,$$

where the last inequality holds due to the the Lipschitz continuity of $g_j^{(i)}$. Together with

$$|\tilde{h}_j^{(1)}(\mathbf{x}) - h_j^{(1)}(\mathbf{x})| \leq c_{14}\cdot h^{-p_j^{(1)}/K_j^{(1)}},$$

which follows again from (38), an easy induction shows

$$|\tilde{h}_1^{(\kappa)}(\mathbf{x}) - h_1^{(\kappa)}(\mathbf{x})| \leq c_{15} \cdot (K_{max} + 1)^\kappa \cdot \max_{j,i} h^{-p_j^{(i)}/K_j^{(i)}}.$$

$\square$

5.4 A bound on the covering number

In this subsection we prove the following bound on the covering number.

Lemma 8. *Let $\mathcal{F}$ be the set of all functions*

$$(x_1, \ldots, x_l) \mapsto z_N \cdot w + b,$$

where z_N is defined in Section 2 depending on

$$(W_{Q,r,s}, W_{K,r,s}, W_{V,r,s})_{r \in \{1,\ldots,N\}, s \in \{1,\ldots,h\}}, \quad (W_{r,1}, b_{r,1}, W_{r,2}, b_{r,2})_{r \in \{1,\ldots,N\}} \tag{39}$$

and where the total number of nonzero components in (39) and w and b is bounded by $L \in \mathbb{N}$. Let $\beta \geq 0$ and let $T_\beta \mathcal{F}$ be the set of all functions in $\mathcal{F}$ truncated on height β and $-\beta$. Then we have for any $0 < \epsilon < \beta/2$

$$\sup_{z_1^n \in (\mathbb{R}^{d \cdot l})^n)} \log \mathcal{N}(\epsilon, T_\beta \mathcal{F}, z_1^n) \leq c_{16} \cdot L \cdot N^2 \cdot \log(\max\{N, h, d_{ff}, d_k, l\}) \log\left(\frac{\beta}{\epsilon}\right).$$

In order to prove Lemma 8 we will first show the following bound on the VC-dimension of subsets of $\mathcal{F}$, where the nonzero components appear only at fixed positions.

Lemma 9. *Let $\mathcal{G}$ be the set of all functions*

$$(x_1, \ldots, x_l) \mapsto z_N \cdot w + b,$$

where z_N is defined in Section 2 depending on (39) and where there are at most L fixed components in (39) and w and b where the entries are allowed to be nonzero. Then we have

$$V_{\mathcal{F}^+} \leq c_{17} \cdot L \cdot N^2 \cdot \log(\max\{N, h, d_{ff}, l\})$$

The proof of Lemma 9 is a modification of the proof of Theorem 6 in Bartlett et al. (1999). In the proof of Lemma 9 we will need the following two auxiliary results.

Lemma 10. *Suppose $W \leq m$ and let $f_1, \ldots, f_m$ be polynomials of degree at most D in W variables. Define*

$$K := |\{(\text{sgn}(f_1(\mathbf{a})), \ldots, \text{sgn}(f_m(\mathbf{a}))) : \mathbf{a} \in \mathbb{R}^W\}|.$$

Then we have

$$K \leq 2 \cdot \left(\frac{2 \cdot e \cdot m \cdot D}{W}\right)^W.$$

28

Proof. See Theorem 8.3 in Anthony and Bartlett (1999). □

Lemma 11. *Suppose that $2^m \leq 2^L \cdot (m \cdot R/w)^w$ for some $R \geq 16$ and $m \geq w \geq L \geq 0$. Then,*

$$m \leq L + w \cdot \log_2(2 \cdot R \cdot \log_2(R)).$$

Proof. See Lemma 16 in Bartlett et al. (2019). □

Proof of Lemma 9. Let $\mathcal{H}$ be the set of all functions h defined by

$$h : (\mathbb{R}^d)^l \times \mathbb{R} \to \mathbb{R}, \quad h(x, y) = g(x) - y$$

for some $g \in \mathcal{G}$. Let $(x_1, y_1), \ldots (x_m, y_m) \in \mathbb{R}^{d \cdot l} \times \mathbb{R}$ be such that

$$|\{(sgn(h(x_1, y_1)), \ldots, sgn(h(x_m, y_m))) \, : \, h \in \mathcal{H}\}| = 2^m. \tag{40}$$

It suffices to show

$$m \leq c_{17} \cdot L \cdot N^2 \cdot \log(\max\{N, h, d_{ff}, l\}). \tag{41}$$

To show this we partition $\mathcal{G}$ in subsets such that for each subset all

$$g(x_i) \quad (i = 1, \ldots, m)$$

are polynomials of some fixed degree and use Lemma 10 in order to derive an upper bound on the left-hand side of (40). This upper bound will depend polynomially on m which will enable us to conclude (41) by an application of Lemma 11.

Let

$$\bar{\theta} = \big((W_{Q,r,s}, W_{K,r,s}, W_{V,r,s})_{r \in \{1,\ldots,N\}, s \in \{1,\ldots,h\}}, (W_{r,1}, b_{r,1}, W_{r,2}, b_{r,2})_{r \in \{1,\ldots,N\}}, w, b\big)$$

be the parameters which determine a function in $\mathcal{F}$. By assumption, each function in $\mathcal{G}$ can be also described by such a parameter vector. Here only L components of the matrices and vectors occuring in the parameter vector are allowed to be nonzero and the positions where these nonzero parameters can occur are fixed. Denote the vector in $\mathbb{R}^L$ which contains all values of these possible nonzero parameters by θ. Then we can write

$$\mathcal{G} = \{g(\cdot, \theta) : \mathbb{R}^{d \cdot l} \to \mathbb{R} \, : \, \theta \in \mathbb{R}^L\}.$$

In the sequel we construct a partition $\mathcal{P}_{N+1}$ of $\mathbb{R}^L$ such that for all $S \in \mathcal{P}_{N+1}$ we have that

$$g(x_1, \theta), \ldots, g(x_m, \theta)$$

(considered as functions of θ) are polynomials of degree at most 8^{N+1} for $\theta \in S$.

In order to construct this partition we construct recursively partionions $\mathcal{P}_0, \ldots, \mathcal{P}_N$ of $\mathbb{R}^L$ such that for each $r \in \{1, \ldots, N\}$ and all $S \in \mathcal{P}_r$ all components in

$$z_r$$

(considered as a function of θ) are polynomials of degree at most 8^r in θ for $\theta \in S$.

Since all components of z_0 are constant as functions of θ this holds if we set $\mathcal{P}_0 = \{\mathbb{R}^L\}$.

Let $r \in \{1, \ldots, N\}$ and assume that for all $S \in \mathcal{P}_{r-1}$ all components in

$$z_{r-1}$$

(considered as a function of θ) are polynomials of degree at most 8^{r-1} in θ for $\theta \in S$. Then all components in

$$q_{r-1,s,i}, k_{r-1,s,i} \quad \text{and} \quad v_{r-1,s,i}$$

are on each set $S \in \mathcal{P}_{r-1}$ polynomials of degree $8^{r-1}+1$. Consequently, for each $S \in \mathcal{P}_{r-1}$ each value

$$< q_{r-1,s,i}, k_{r-1,s,j} >$$

is (considered as a function of θ) a polynomial of degree at most $2 \cdot 8^{r-1} + 2$ for $\theta \in S$. Application of Lemma 10 yields that

$$< q_{r-1,s,i}, k_{r-1,s,j_1} > - < q_{r-1,s,i}, k_{r-1,s,j_2} > \quad (s \in \{1, \ldots, h\}, i, j_1, j_2 \in \{1, \ldots, l\})$$

has at most

$$\Delta = 2 \cdot \left(\frac{2 \cdot e \cdot h \cdot l^3 \cdot (2 \cdot 8^{r-1} + 2)}{L} \right)^L$$

difference sign patterns. If we partiton in each set in $\mathcal{P}_{r-1}$ according to these sign patterns in Δ subsets, then on each set in the new partition all components in

$$v_{r-1,s,\hat{j}_{r-1,s,i}} \cdot < q_{r-1,s,i}, k_{r-1,s,\hat{j}_{r-1,s,i}} >$$

are polynomials of degree at most $3 \cdot 8^{r-1}+3$ (since on each such set $< q_{r-1,s,i}, k_{r-1,s,\hat{j}_{r-1,s,i}} >$ is equal to one of the $< q_{r-1,s,i}, k_{r-1,s,j} >$). On each set within this partition every component of the $\mathbb{R}^{d_{ff}}$-valued vectors

$$W_{r,1} \cdot y_{r,s} + b_{r,1} \quad (s = 1, \ldots, h)$$

is (considered as a function of θ) a polynomial of degree at most $3 \cdot 8^{r-1} + 4$.

By another application of Lemma 10 we can refine each set in this partition into

$$2 \cdot \left(\frac{2 \cdot e \cdot h \cdot d_{ff} \cdot (3 \cdot 8^{r-1} + 4)}{L} \right)^L$$

sets such that all components in

$$W_{r,1} \cdot y_{r,s} + b_{r,1} \tag{42}$$

have the same sign patterns within the refined partition. We call this refined partion $\mathcal{P}_r$. Since on each set of $\mathcal{P}_r$ the sign of all components in (42) does not change we can conclude that all components in

$$\sigma(W_{r,1} \cdot y_{r,s} + b_{r,1}) \tag{43}$$

are either equal to zero or they are equal to a polynomial of degree at most $3 \cdot 8^{r-1} + 4$. Consequently we have that on each set in $\mathcal{P}_r$ all components of

$$z_r$$

are equal to a polynomial of degree at most $3 \cdot 8^{r-1} + 5 \le 8^r$.

Using $\mathcal{P}_{N+1} = \mathcal{P}_N$ we have constructed a partition with

$$|\mathcal{P}_{N+1}| = \prod_{r=1}^{N} \frac{|\mathcal{P}_r|}{|\mathcal{P}_{r-1}|} \le \prod_{r=1}^{N} 2 \cdot \left(\frac{2 \cdot e \cdot h \cdot l^3 \cdot 8^r}{L} \right)^L \cdot 2 \cdot \left(\frac{2 \cdot e \cdot h \cdot d_{ff} \cdot 8^r}{L} \right)^L$$

such that for each set in this partition for all $(x, y) \in \{(x_1, y_1), \ldots, (x_m, y_m)\}$

$$g(x) = z_N \cdot w + b \quad \text{and} \quad h(x, y) = z_N \cdot w + b - y$$

(considered as a function of θ) are polynomials of degree at most $8^N + 1 \le 8^{N+1}$ in θ for $\theta \in S$.

Using

$$|\{(sgn(h(x_1, y_1)), \ldots, sgn(h(x_m, y_m))) \ : \ h \in \mathcal{H}\}|$$
$$\le \sum_{S \in \mathcal{P}_{N+1}} |\{(sgn(g(x_1, \theta) - y_1), \ldots, sgn(g(x_m, \theta) - y_m)) \ : \ \theta \in S\}|$$

we can apply one more time Lemma 10 to conclude

$$
\begin{aligned}
2^m \ &= \ |\{(sgn(h(x_1, y_1)), \ldots, sgn(h(x_m, y_m))) \ : \ h \in \mathcal{H}\}| \\
&\le \ |\mathcal{P}_{N+1}| \cdot 2 \cdot \left(\frac{2 \cdot e \cdot m \cdot 8^{N+1}}{L} \right)^L \\
&\le \ 2 \cdot \left(\frac{2 \cdot e \cdot m \cdot 8^{N+1}}{L} \right)^L \cdot \prod_{r=1}^{N} 2 \cdot \left(\frac{2 \cdot e \cdot h \cdot l^3 \cdot 8^r}{L} \right)^L \cdot 2 \cdot \left(\frac{2 \cdot e \cdot h \cdot d_{ff} \cdot 8^r}{L} \right)^L \\
&\le \ 2^{2 \cdot N + 1} \cdot \left(\frac{m \cdot 2 \cdot e \cdot (2N+1) \cdot h \cdot (\max\{l, d_{ff}\})^3 \cdot 8^{N+1}}{(2N+1) \cdot L} \right)^{(2N+1) \cdot L}.
\end{aligned}
$$

Assume $m \ge (2N+1) \cdot L$. Application of Lemma 11 with $L = 2 \cdot N + 1$, $R = 2 \cdot e \cdot (2N+1) \cdot h \cdot (\max\{l, d_{ff}\})^3 \cdot 8^{N+1} \ge 16$ and $w = (2N+1) \cdot L$ yields

$$m \le (2 \cdot N + 1) + (2 \cdot N + 1) \cdot L \cdot \log_2(2 \cdot R \cdot \log_2(R)) \le c_{18} \cdot L \cdot N^2 \cdot \log(\max\{N, h, d_{ff}, l\}),$$

which implies (41). $\qquad\square$

Proof of Lemma 8. The functions in the function set $\mathcal{F}$ depend on at most

$$\lceil c_{19} \cdot N \cdot h^2 \cdot (\max\{d_k, d_{ff}, d, l\})^3 \rceil$$

many parameters, and of these parameters at most L are allowed to be nonzero. We have

$$\binom{\lceil c_{19} \cdot N \cdot h^2 \cdot (\max\{d_k, d_{ff}, d, l\})^3 \rceil}{L} \le \left(\lceil c_{19} \cdot N \cdot h^2 \cdot (\max\{d_k, d_{ff}, d, l\})^3 \rceil \right)^L$$

many possibilities to choose these positions. If we fix these positions, we get one function space $\mathcal{G}$ for which we can bound its VC dimension by Lemma 9. Using Lemma 9, $V_{T_\beta \mathcal{G}^+} \leq V_{\mathcal{G}^+}$, Lemma 9.2 and Theorem 9.4 in Györfi et al. (2002) we get

$$
\begin{aligned}
\mathcal{N}_1\left(\epsilon, T_\beta \mathcal{G}, \mathbf{x}_1^n\right) &\leq 3 \cdot \left(\frac{4e \cdot \beta}{\epsilon} \cdot \log \frac{6e \cdot \beta}{\epsilon}\right)^{V_{T_\beta \mathcal{G}^+}} \\
&\leq 3 \cdot \left(\frac{6e \cdot \beta}{\epsilon}\right)^{c_{20} \cdot L \cdot N^2 \cdot \log(\max\{N, h, d_{ff}, l\})} .
\end{aligned}
$$

From this we can conclude

$$
\begin{aligned}
\sup_{z_1^n \in (\mathbb{R}^{d \cdot l})^n)} &\log \mathcal{N}(\epsilon, T_\beta \mathcal{F}, z_1^n) \\
&\leq L \cdot \log\left(\lceil c_{19} \cdot N \cdot h^2 \cdot (\max\{d_k, d_{ff}, d, l\})^3 \rceil\right) \\
&\quad + c_{20} \cdot L \cdot N^2 \cdot \log(\max\{N, h, d_{ff}, l\}) \cdot \log\left(\frac{\beta}{\epsilon}\right) \\
&\leq c_{21} \cdot L \cdot N^2 \cdot \log(\max\{N, h, d_{ff}, d_k, l\}) \cdot \log\left(\frac{\beta}{\epsilon}\right) .
\end{aligned}
$$

$\square$

References

[1] Anthony, M., and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.

[2] Bauer, B., and Kohler, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *Annals of Statistics*, **47**, pp. 2261-2285.

[3] Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. (2019). Nearly-tight VC-dimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, **20**, pp. 1–17.

[4] Cover, T. M. (1968). Rates of convergence of nearest neighbor procedures. In *Proceedings of the Hawaii International Conference on Systems Sciences*, pp. 413-415, Honolulu, HI.

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

[6] Devroye, L. (1982). Necessary and sufficient conditions for the almost everywhere convergence of nearest neighbor regression function estimates. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, **61**, pp. 467-481.

[7] Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory Of Pattern Recognition*. Springer.

[8] Györfi, L., Kohler, M., Krzyżak, A., and Walk, H. (2002). *A Distribution-Free Theory of Nonparametric Regression*. Springer.

[9] Imaizumi, M. and Fukamizu, K. (2019). Deep neural networks learn non-smooth functions effectively. *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS 2019)*. Naha, Okinawa, Japan.

[10] Kohler, M. (2014). Optimal global rates of convergence for noiseless regression estimation problems with adaptively chosen design. it Journal of Multivariate Analysis, **132**, pp. 197-208.

[11] Kohler, M. and Krzyżak, A. (2017). Nonparametric regression based on hierarchical interaction models. *IEEE Transaction on Information Theory*, **63**, pp. 1620–1630.

[12] Kohler, M. and Langer, S. (2020). On the rate of convergence of fully connected deep neural network regression estimates. To appear in *Ann. Stat.* arXiv: 1908.11133

[13] Langer, S. (2021a). Analysis of the rate of convergence of fully connected deep neural network regression estimates with smooth activation function, *Journal of Multivariate Analysis*, **182**, pp. 104695

[14] Langer, S. (2021b). Approximating smooth functions by deep neural networks with sigmoid activation function, *Journal of Multivariate Analysis*, **182**, pp. 104696

[15] Lu, J., Shen, Z., Yang, H. and Zhang, S. (2020) Deep Network Approximation for Smooth Functions. arxiv: 2001.03040

[16] Schmidt-Hieber, J. (2019). Deep ReLU networks approximation of functions on a manifold. arxiv:1908.00695

[17] Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019). How to fine-tune BERT for text classification? *In China National Conference on Chinese Computational Linguistics*, pp. 194-206. Springer, Cham.

[18] Suzuki, T. (2018). Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality. arXiv: 1810.08033.

[19] Suzuki, T. and Nitanda, A. (2019). Deep learning is adaptive to intrinsic dimensionality of model smoothness in anisotropic Besov space. arXiv: 1910.12799.

[20] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin,I. (2017). Attention is all you need. Arxiv 1706.03762.

[21] Yarotsky, D. (2017). Error bounds for approximations with deep ReLU networks, *Neural Networks*, **94**, pp. 103–114.

[22] Yarotsky, D. and Zhevnerchuk, A. (2020). The phase diagram of approximation rates for deep neural networks. In *Advances in Neural Information Processing Systems*, **33**, pp. 13005–13015.

[23] Young, T., Hazarika, D., Poria, S., and Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, **13**, pp. 55-75.

[24] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in neural information processing systems*, **28**, 649-657.