Moscow State University M.V.Lomonosov

Faculty of Computational Mathematics and Cybernetics

Department of Mathematical Methods of Forecasting

COURSE WORK

«Fully-connected neural networks for computer vision and natural language processing tasks»

Completed by:

a student of 3 courses of 317 group

Kurtsev D.V.

Scientific director:

s.s. *Kropotov D.A.*

Moscow, 2022

Contents

Abstract

Transformers are the state-of-the-art models in the field of deep learning. They have achieved many breakthroughs over the past few years in NLP and CV tasks. Convolutional neural networks (CNNs) are also popular models in CV. This paper examines simple architectures based on multilayer perceptron, MLP-Mixer [1] and gMLP [2]. They question the necessity for a self-attention and a convolution layers to achieve good accuracy. MLP networks obtain competitive results in text and image classification tasks, with pre-train and fine-tuning costs comparable to sota.

1 Introduction

Currently, there are many neural network architectures that can achieve good quality. Convolutional (CNNs) and recurrent (RNNs) neural networks were the best models for computer vision and text processing tasks. The state-of-the-art architecture of late is Transformers, especially in NLP. Their main idea is the attention blocks (self-attention). This figure briefly presents the Vision Transformer (ViT) [4] scheme:

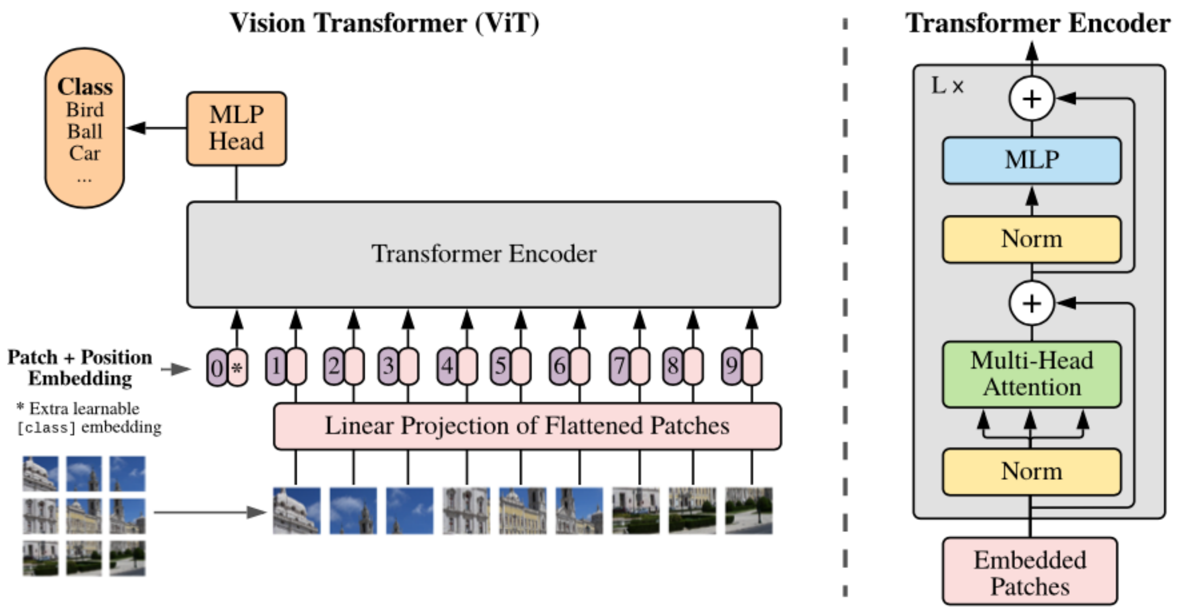


Figure 1: ViT

This paper examines conceptually simple models based only on linear transformations - *MLP-Mixer* and *gMLP*. They do not use convolutions and attention mechanisms. This examines the need to use technically complex models: Transformers and CNNs to achieve a good result.

MLP models have two different units: channel projection and spatial projection. These networks are able to show competitive results for the task of classifying texts and images.

These neural networks can be applied to tasks from different areas - CV and NLP.

In the task of classifying images on the input of the model comes the picture and it is necessary to determine what it represents (to relate some class). In this paper such datasets as ImageNet, CIFAR-10, CIFAR-100 and Pets were considered.

Text classification tasks differ depending on the set of data. The following datasets were considered. SST-2 consists of sentences that are comments to films. It is necessary to determine if they have the positive or negative shade. In the MNLI dataset, each object is a crowdsourced collection of sentence pairs. Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis (entailment), contradicts the hypothesis (contradiction), or neither (neutral)

1.1 Problem statement

The paper explored the applicability of simpler models such as MLP-Mixer and gMLP to computer vision and text processing. The main goal is show that under certain conditions they are able to get comparable quality with modern architectures. A study was also carried out to adjust the parameters of the networks in question.

2 MLP-Mixer

2.1 Architecture

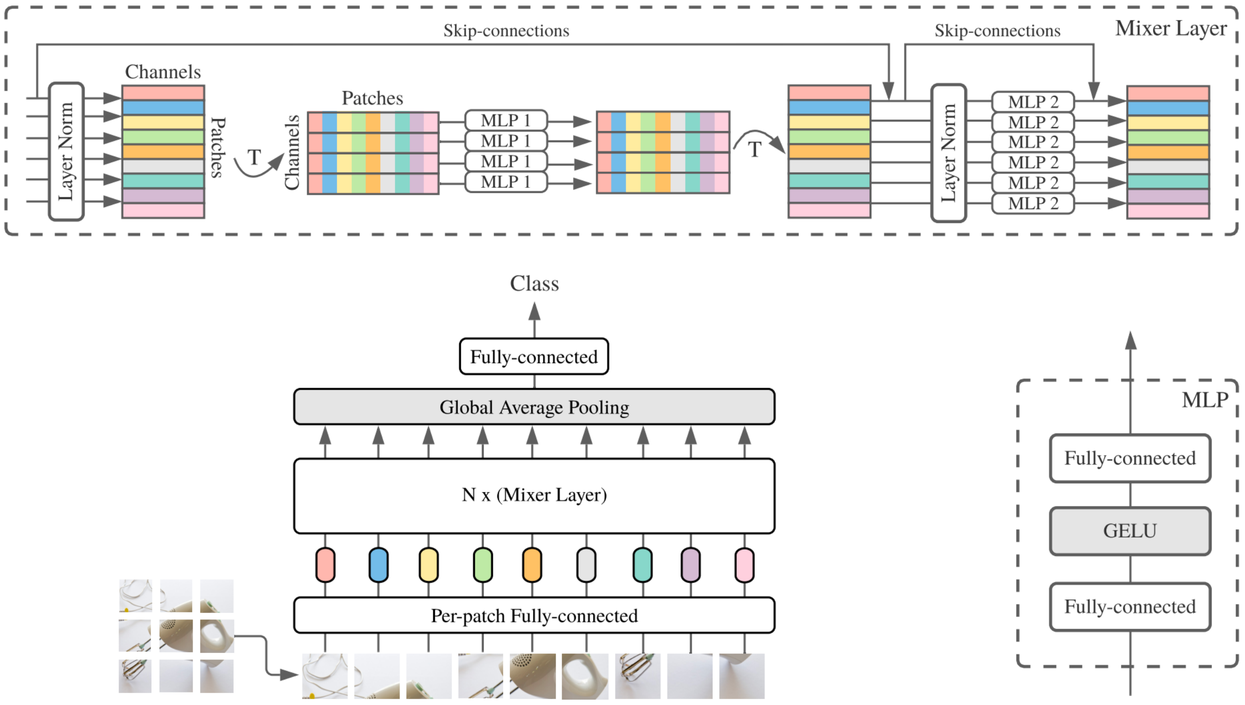


Figure 2: MLP-Mixer

This picture shows the architecture of MLP-Mixer. The neural network considered was used only for CV tasks.

Input is an images with size $H \times W$. Each of which is broken into overlapping $P \times P$ patches. We will get that their total number is $S = \frac{HW}{P^2}$. All patches are linearly projected into a vector of dimension C. This operation can be done as a convolution with a kernel size of P and a stride P. Thus, the first block receives a matrix $\mathbf{X} \in \mathbb{R}^{S \times C}$. It is a sequence of S patches, each represented by vectors with C channels.

Modern neural networks consist of layers that are capable of mixing through channels and patches. For example, in convolutional neural networks, this can be done both at once using the kernel size of $N \times N (N > 1)$. Transformers also produce mixing simultaneously due to attention block (self-attention). The idea of MLP-Mixer is to perform these operations strictly in separately.

As seen from Figure.2 MLP-Mixer consists of blocks, each of which is represented by two types of levels: spatial projections (token-mixing) and channel projections (channel-mixing). The output of each layer has the same dimension as the input. That is, token-mixing can be thought of as a function acting under the following rule: $\mathbb{R}^S \mapsto \mathbb{R}^S$, where all transformations apply to each column of the X matrix independently. A channel-mixing: $\mathbb{R}^C \mapsto \mathbb{R}^C$. Thus, each block of MLP-Mixer receives as the input a matrix $X \in \mathbb{R}^{S \times C}$ of the same size. In this it is similar to Transformers and recurrent neural networks. But it differs from convolutional architectures, where image resolution decreases from block to block, but increases the number of channels.

The output uses a standard linear classification layer that matches each image with classes scores.

Each MLP unit consists of two fully-connected layers and a non-linear activation function between them:

$$\mathbf{U}_{*,i} = \mathbf{X}_{*,i} + \mathbf{W}_2\, \sigma(\mathbf{W}_1\, LayerNorm(\mathbf{X})_{*,i}), \quad i = \overline{1, C} \quad \text{- token-mixing}$$

$$\mathbf{Y}_{j,*} = \mathbf{U}_{j,*} + \mathbf{W}_4\, \sigma(\mathbf{W}_3\, LayerNorm(\mathbf{U})_{j,*}), \quad j = \overline{1, S} \quad \text{- channel-mixing}$$

Where the activation function is $\sigma =$ GELU:

$$GELU(x) = x\Phi(x) = 0.5x(1 + tanh[\sqrt{2/\pi}(x + 0.044715x^3)])$$

Input of any block is normalized by LayerNorm through channels. Skip-connection mechanism and biases of each projection are also used. The parameters of each layer are four matrices: $W_1 \in \mathbb{R}^{D_S \times S}$, $W_2 \in \mathbb{R}^{S \times D_S}$, $W_3 \in \mathbb{R}^{D_C \times C}$, $W_4 \in \mathbb{R}^{C \times D_C}$, where D_S и D_C are hyperparameters of the model. They are selected independently of S and C. But selected so that the inequalities are true $D_S > S$ and $D_C > C$.

Calculate the computational complexity of each MLP-Mixer layer. Token-mixing takes action on the columns of the X matrix, for all channels. We get only $2D_SSC$ of operations. Similarly, the channel-mixing layer performs transformations over the strings of the input matrix for each patch. The total has $2D_CSC$ operations. Thus, we get that the computational complexity of the MLP-Mixer is linear by the number of channels and patches. While ViT is quadratic by S.

Note that in extreme cases, the channel-mixing block can be considered as a 1×1 convolution, which has a simpler implementation, unlike CNNs. Also MLP-Mixer does not use positional encoding, as the token-mixing layer allows to catch connections between different patches and take into account their location.

2.2 Experiments

MLP-Mixer has been tested on various data sets - ImageNet, CIFAR-10, CIFAR-100 and Pets. To check the quality of the model, the accuracy metric was used, which equals the correct response rate. Several experiments were carried out with the previously pre-trained model and then fine-tuning on the downstream tasks and trained from scratch with random initialization. Pre-training was made on two datasets: ImageNet-21k and JFT-300M. The first includes 14 million images and 21 thousand class labels. The latter consists of 300 millions of data and 18 thousand classes. The images in these datasets have sizes 224×224. Therefore, during fine-tuning the model on other datasets that have other permissions, the images were changed to the desired size. MLP-Mixer was pre-trained with 224 resolution, Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, linear learning rate warmup of 10k steps and linear decay, batch size 4096, weight decay and gradient clipping with norm 1. As it was empirically proven it's necessary to add regularization. Otherwise Mixer can be heavily overtrained.

It's reports to take the following hyperparameters, as shown in Table 1.

The main goal of this work was to show that conventional MLP models are able to achieve the same quality as modern architectures, rather than show a new state-of-the-art

Specification	S/32	S/16	B/32	B/16	L/32	L/16	H/14
Number of layers	8	8	12	12	24	24	32
Patch resolution $P \times P$	32×32	16×16	32×32	16×16	32×32	16×16	14×14
Hidden size C	512	512	768	768	1024	1024	1280
Sequence length S	49	196	49	196	49	196	256
MLP dimension D_C	2048	2048	3072	3072	4096	4096	5120
MLP dimension D_S	256	256	384	384	512	512	640
Parameters (M)	19	18	60	59	206	207	431

Table 1: Specification MLP-Mixer

result.

The article [1] compares with a number of different neural networks. Convolutional neural networks: ResNets, NFNet-F4+. Models based on self-attention: HaloNet, ViT-L/16. The experiments were carried out on ImageNet, ReaL datasets, including 1.3M images and 1K classes. The networks were also tested on five downstream tasks: ImageNet, CIFAR-10, CIFAR-100, Pets and Flowers. The results can be seen in Table 2 in the column “Avg 5” - average on these five datasets. There were made 19 tasks from the VTAB-1k dataset.

Table 2 presents models’ results on different datasets.

As can be seen, Mixer shows good accuracy on all downstream tasks, quite a bit behind the advanced models. It can be seen, that pre-trained with a smaller dataset (ImageNet-21k) Mixer is slightly worse than ViT and CNNs, because MLP behind over 1% on ImageNet. At the same time on VTAB it shows the quality much better. Despite high throughput comparable to HaloNet, Mixer-L/16 training several times longer than a Transformer.

It is noticeable that by increasing the network size and amount of data on which the pre-training were made, Mixer-H/14 shows a comparable quality. He beat BiT-R152x4 on some tasks, losing ViT less than one percent. Mixer remains a computationally efficient model, processing several times more images than ViT and BiT-R152x4. In addition, the time for training the MLP model is significantly reduced compared to others. It learns

	ImNet top-1	ReaL top-1	Avg 5 top-1	VTAB-1k 19 tasks	Throughput img/sec/core	TPUv3 core-days
Pre-trained on ImageNet-21k (publik)						
HaloNet	85.8	–	–	–	120	0.10k
Mixer-L/16	84.15	87.86	93.91	74.95	105	0.41k
ViT-L/16	85.30	88.62	94.39	72.72	32	0.18k
BiT-R152x4	85.39	–	94.04	70.64	26	0.94k
Pre-trained on JFT-300M (proprietary)						
NFNet-F4+	89.2	–	–	–	46	1.86k
Mixer-H/14	87.94	90.18	95.71	75.33	40	1.01k
BiT-R152x4	87.54	90.54	95.33	76.29	26	9.90k
ViT-H/14	88.55	90.72	95.97	77.63	15	2.30k

Table 2: Results

twice as fast as a transformer. And overtakes even NFNet.

In this work the own version of MLP-Mixer was implemented and the experiments on various datasets were carried out. The neural network was training using Adam with $\beta_1 = 0.9$, $\beta_- = 0.999$, learning rate $6 \cdot 10^{-6}$, batch size 64 and weight decay 10^{-3}. The results can be seen in Table 3.

	CIFAR-10	CIFAR-100	Pets
Pre-trained on ImageNet-21k			
our Mixer-B/16	97.7	89.7	97.4
Google's Mixer-B/16	96.8	-	-
ViT-B/16	98.8	91.9	-
ResNet152	98.9	88.5	96.6
From scratch			
our Mixer	82.3	68.8	79.4
ViT-B/16	98.6	89.1	93.1
ResNet152	98.2	87.8	93.3

Table 3: Results

As seen from MLP experiments, the network is highly susceptible from pre-training. If you take the pre-trained model and fine-tuning it to the downstream task, it starts to show very good results, which are comparable to Transformers and convolutional neural networks. On the Pets dataset, Mixer even slightly wins against ResNet152. If you try to run models without the pre-training (from scratch), they show worse results. However, MLP is beginning to outperform modern architectures by 15-20%. Empirically it was derived that the Mixer with random initialization, works better if you keep the size of the images unchanged without increasing them to 224. At the same time, you should reduce the hyperparameters: $P = 4$, $S = 64$, $C = 256$, $D_C = 1024$, $D_S = 256$.

2.3 Findings

Thus, it can be concluded that the MLP-Mixer is highly dependent on from pre-training and selection of parameters. By increasing the learning rate or changing the width of the layers, the network may stop learning and begin to show completely bad results.

Also because of the simple implementation of this architecture, token-mixing layers depend on the length S of the sequence, which is a significant limitation of this model. Since it can only work with fixed image sizes. By changing the resolution of the patch, you have to retool the neural network.

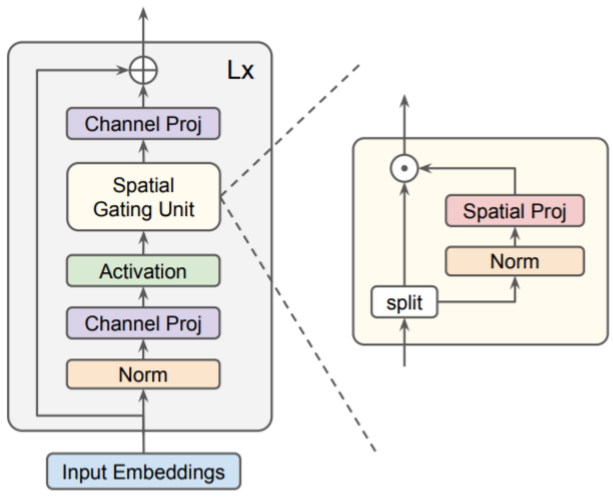
3 gMLP

This neural network was used for CV and NLP tasks. Work with the images exactly coincides with MLP-Mixer [1] and ViT [4]. At the input, the image is divided into non-overlapping patches, which are linearly projected into the vector of the hidden dimension C. Then the input matrix $\mathbf{X} \in \mathbb{R}^{S \times C}$ is formed, which then passes through all network blocks. The output also uses a standard linear classification layer.

Text processing is similar to BERT [3]. The original sentence is split into S of tokens using a 30K size dictionary. Next, the WordPiece technique produces the embeddings of each token. This creates the $\mathbf{X} \in \mathbb{R}^{S \times C}$ matrix, over which the same manipulations are made.

3.1 Architecture

This figure shows the gMLP block diagram.



Pseudo-code for the gMLP block

```
def gmlp_block(x, d_model, d_ffn):
  shortcut = x
  x = norm(x, axis="channel")
  x = proj(x, d_ffn, axis="channel")
  x = gelu(x)
  x = spatial_gating_unit(x)
  x = proj(x, d_model, axis="channel")
  return x + shortcut

def spatial_gating_unit(x):
  u, v = split(x, axis="channel")
  v = norm(v, axis="channel")
  n = get_dim(v, axis="spatial")
  v = proj(v, n, axis="spatial", init_bias=1)
  return u * v
```

Figure 3: gMLP

gMLP consists of a stack of L blocks with identical structure indicated in the figure. Unlike Mixer, this architecture has three fully-connected layers at each block. The mixing of channels and tokens does not go hand in hand, but is mixed. gMLP blocks can be described as follows:

$$Z = GELU(XU), \quad \widetilde{Z} = s(Z), \quad Y = \widetilde{Z}V + X$$

Where model parameters are the $U \in \mathbb{R}^{C \times D_C}, V \in \mathbb{R}^{D_C \times C}$ matrices, which perform a channel projection (equivalent to channel-mixing in MLP-Mixer). GELU is used as the activation function.

The key location in this architecture is the function $s(\cdot)$ - SGU (Spatial Gating Unit). The transformations described above work with each token independently. To account for the relationships between them, we need to have a layer that mixes patches (token-mixing in MLP-Mixer). The simplest implementation is a linear transformation. SGU projects tokens with a $f(\cdot)$ function, and the gateway of the model is $s(\cdot)$:

$$f(Z) = WZ + b$$

$$s(Z) = Z \odot f(Z)$$

where W and b are the model parameters and $\odot$ is the element-wise multiplication. Since this architecture uses only one layer of linear spatial projection (in MLP-Mixer two), because of the need to keep a dimension (for $s(\cdot)$), the W matrix must be square. That is $W \in \mathbb{R}^{S \times S}$, S - sequence length.

Initially W is initialized with values close to zero and b to one. Thus it turns out that at the first steps of learning $f(Z) \approx 1 \quad \Rightarrow \quad s(Z) \approx Z$. This initialization allows all tokens to be processed independently of each other. That is, the MLP block in the initial stages of learning becomes similar to the normal FFN in Transformers.

It has been found that to improve neural network performance, the matrix Z that comes at the input of the SGU can be splited into two independent parts (Z_1, Z_2) by the channels. In this case, the output of the SGU unit is determined as follows:

$$s(Z) = Z_1 \odot f(Z_2)$$

Also in this architecture used biases, normalizations through channels and mechanisms of skip connections that were not previously issued, so as not to increase the formulas. Likewise MLP-Mixer, gMLP does not use positional encoding, as this technique is redundant. The SGU block, using patch projections, is able to account for relationships and token locations. Also, each gMLP layer, like Mixer, recurrent neural networks and Transformers, accepts the same size $S \times C$ matrix input.

Let's look at the computational complexity of each gMLP block. The two layers performing channels projections, analogical channel-mixing in MLP-Mixer have a total of

$2D_CSC$ operations. The SGU performs actions on the columns of the Z matrix, for all channels. So, it turns out D_CS^2 operations in the function $f(\cdot)$ and D_CS in $s(\cdot)$. Thus, the computational complexity of gMLP is linear in terms of the number of channels, but quadratic in terms of tokens. Transformers have the same difficulty.

3.2 Experiments CV

In this section, the gMLP neural network has been applied to the image classification tasks. Studies have been conducted on ImageNet, CIFAR-10, CIFAR-100 and Pets. The original images were divided into non-overlapping patches of 16×16 for ImageNet (initial resolution 224×224) and 4×4 for CIFAR-10, CIFAR-100 (original 32×32). The authors of [2] suggest taking the following hyperparameters:

	Number of layers	C	D_C	Params(M)
gMLP-Ti	30	128	768	5.9
gMLP-S	30	256	1536	19.5
gMLP-B	30	512	3072	73.4

Table 4: Specification gMLP

Table 5 shows the results of various models (CNNs, Transformers, MLP) that have been trained without extra data for random initialization on the reviewed datasets.

On the basis of Tables 2 and 5 it is possible to conclude the same findings as for MLP-Mixer. Fully-connected neural networks without pre-training show much inferior quality than modern models. They are also much more difficult to configure (to select hyperparameters). However, as seen from the ImageNet results, sometimes MLP models show comparable quality with ViT.

Model	ImNet Top-1	CIFAR-10	CIFAR-100	Pets	Params (M)
Convolutional neural networks					
ResNet-152	78.3	98.2	87.8	93.3	60
EfficientNet-B7	84.3	–	–	–	66
NFNet-F0	83.6	–	–	–	72
Transformers					
ViT-B/16	77.9	98.6	89.1	93.1	86
DeiT-B (ViT+reg)	81.8	–	–	–	86
MLP					
Mixer-B/16	76.4	–	–	–	59
gMLP-S	79.6	–	–	–	20
our Mixer	–	82.3	68.8	79.4	86
our gMLP	–	80.2	69.7	80.3	86

Table 5: CV results

3.3 NLP Experiments

This section examines the application of gMLP to the Masked Language Model (MLM) task. All network inputs and outputs are analogical BERT [3]. Pre-training was made on the C4/RealNews dataset, which included 5,000 different news topics. All texts are of different lengths, so in order to work correctly with data and matrix calculations, a special <*pad*> token is added at the end to maximize the length. gMLP was pre-trained with AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, linear learning rate decay with 125K steps of warmup steps with a peak value of $7 \cdot 10^{-4}$, batch size 2048, weight decay and a maximum length of 128 tokens.

A article [2] conducted the following experiments, researching different variants of SGU, presented in Table 6. Each gMLP model has 36 blocks, $C = 512$ and $D_C = 3072$.

Further, a study was conducted on two previously described datasets: SST-2 and MNLI. To check the quality of the model, the accuracy metric was used. The results of gMLP and BERT are shown in Table 7.

As can be seen, the MLP model is able to show a comparable result with Transformers. You can see that by increasing the sizes of a fully-connected neural network, it becomes

SGU	Perplexity	Params(M)
Linear, $s(Z) = f(Z)$	5.14	92
Additive, $s(Z) = Z + f(Z)$	4.97	92
Multiplicative, $s(Z) = Z \odot f(Z)$	4.53	92
Multiplicative, Split, $s(Z) = Z_1 \odot f(Z_2)$	4.35	102

Table 6: SGU results

Model	#L	Params (M)	Perplexity	SST-2	MNLI
BERT	6	67	**4.91**	90.4	**81.5**
gMLP	18	59	5.25	**91.2**	77.7
BERT	12	110	**4.26**	91.3	**83.3**
gMLP	36	102	4.35	**92.3**	80.9
BERT	24	195	3.83	92.1	**85.2**
gMLP	72	187	**3.79**	**93.5**	82.8
BERT	48	365	3.47	92.8	**86.3**
gMLP	144	357	**3.43**	**95.1**	84.6

Table 7: NLP results

better to learning on the MLM task. Perplexity decreases with the growth of the model. On the other hand, gMLP can beat BERT (SST-2) or lose (MNLI) on various dates. This is likely due to the different types of tasks that are addressed in these datasets, where MNLI is more complex.

In this paper, gMLP was implemented and tested on the same datasets. It was trained from scratch with parameters suggested by the authors of [2]: AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, linear learning rate decay on 5 epochs of warmup steps with a peak value of $2 \cdot 10^{-5}$, batch size 32, weight decay and a maximum length of 128 tokens.

It has been empirically observed that gMLP always learns a W matrix in a function $f(\cdot)$ (SGU) Toeplitz-like matrices. This allows models to be invariant to word shifts. Also, the limitation on the Toeplitz-like allows to significantly reduce the number of learning parameters. The next model with this restriction is called tMLP.

Since no weights were given to the pre-trained network to avoid this computationally complex process, the author of this work got good word embeddings thanks to the previously pre-trained BERT and trained gMLP on them. The results of the experiments are presented in Table 8:

Model (#L = 18)	SST-2	MNLI-m
our gMLP	80.4	65.1
our gMLP + BERT	88.1	74.6
our gMLP_split + BERT	88.6	75.2
our tMLP_split + BERT	89.8	75.5
Google's gMLP_split	91.2	77.7
BERT	90.4	81.5

Table 8: NLP results

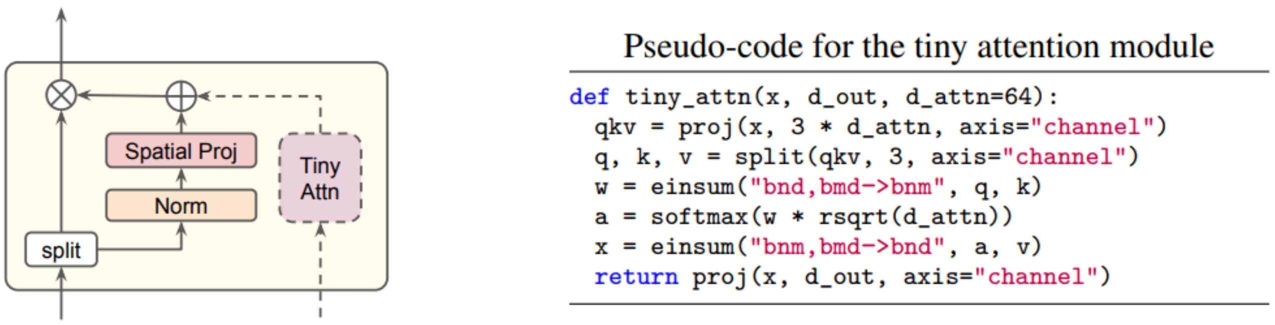
Here the similar conclusions have been drawn as for Mixer. Without pre-training MLP model fails to perform well, lagging far behind the BERT (10-15%). If we apply contextual embeddings to the network, it starts to learn much better and show a quality comparable to what happened in [2], lagging behind BERT by only a few percent.

As can been seen, if we split the Z matrix into two parts through the channels in the SGU block, the model learns even better and shows a slightly higher result. It is also noticeable that W limit in Toeplitz-like matrices of $f(\cdot)$ allows not only to reduce the number of parameters, but also to increase the accuracy on these tasks. Next, gMLP uses a split of Z and Toeplitz matrices.

3.4 gMLP + Tiny Attention

As can be seen from the results of Tables 6-8, attention blocks are not a necessary mechanism for achieving good results. The MLP model sometimes even wins against the BERT, showing quality a few percent better (on the SST-2). At the same time, the fully-connected network works worse than the Transformer on MNLI. This is probably due to the fact that MNLI consists of pairs of sentences and the attention mechanism allows take this into account. For MLP it is much more difficult. Therefore, to improve the quality of network operation, it is proposed to use a small attention block with one head. gMLP network is able to learn well without it, so the added block is important only by its presence. Then the gMLP with the tiny attention is labeled aMLP.

This figure shows the diagram of the aMLP SGU block:



Pseudo-code for the tiny attention module

```
def tiny_attn(x, d_out, d_attn=64):
  qkv = proj(x, 3 * d_attn, axis="channel")
  q, k, v = split(qkv, 3, axis="channel")
  w = einsum("bnd,bmd->bnm", q, k)
  a = softmax(w * rsqrt(d_attn))
  x = einsum("bnm,bmd->bnd", a, v)
  return proj(x, d_out, axis="channel")
```

Figure 4: aMLP

3.5 Experiments gMLP + Tiny Attention

Model (#L = 18)	SST-2	MNLI-m
our gMLP + BERT	89.8	75.5
our aMLP + BERT	90.7	77.1
Google's gMLP	91.2	77.7
Google's aMLP	91.9	82.4
BERT	90.4	81.5

Table 9: NLP results

As can be seen from the results presented in Table 9, a large attention block (multi-head self-attention) is superfluous to achieve good quality. Most of its functions can be contained in linear token projections. Adding only a small and one-head attention may be enough to show better quality than BERT. This block allows take into account the biases of words and sentences in the text.

4 Conclusion

In this paper, the fully-connected neural networks MLP-Mixer and gMLP were implemented and detailed. The need of self-attention layer to achieve good quality was examined.

Models are able to show good quality, sometimes even better than advanced architectures. By increasing the size of a fully-connected network, it is possible to significantly reduce the gap with Transformers and convolutional neural networks. By adding a small attention block, the model is able to achieve even higher results, than Transformers.

Despite the simplicity of MLP networks on and high quality fine-tuning for the downstream tasks, they are very difficult to adjust. It takes a long time to select hyperparameters to get reasonable accuracy.

Also these neural networks are highly susceptible to pre-training. It has been shown, that they are not able to achieve as good results as Transformers and CNNs. In addition, the architectures under consideration apply to a narrow range of tasks - classification of texts and images (encoder-only architecture). It is not clear how to make, for example, a language model without using cross-attention. This is a strong restriction of these models.

5 References

[1] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. arXiv preprint arXiv:2105.01601, 2021.

[2] Hanxiao Liu, Zihang Dai, David R. So, Quoc V. Le. Pay Attention to MLPs.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL, 2018.

[4] 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. In ICLR, 2021.