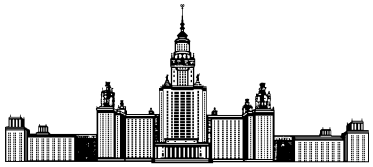
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

КУРСОВАЯ РАБОТА

«Полносвязные нейронные сети для задач компьютерного зрения и обработки естественного языка»
«Fully-connected neural networks for computer vision and natural language processing tasks»

Выполнил:

студент 3 курса 317 группы

Курцев Дмитрий Витальевич

Научный руководитель:

н.с. *Кропотов Дмитрий Александрович*

Москва, 2022

Содержание

Аннотация

Трансформеры являются передовыми моделями в области глубокого обучения. Они позволили добиться многих прорывов за последние несколько лет в задачах обработки изображений и текстов. Также популярными моделями в области компьютерного зрения являются свёрточные нейронные сети. В данной работе рассматриваются простые архитектуры, основанные на многослойном перцептроне, MLP-Mixer [1] и gMLP [2]. Они ставят под сомнение необходимость слоя внимания и свёрток для достижения хорошей точности. Полносвязные сети получают конкурентоспособные результаты в задачах классификации текстов и изображений, при этом затраты на предобучение и точную настройку сопоставимы с современными архитектурами.

1 Введение

В настоящее время существует множество архитектур нейронных сетей, способных достичь хорошего качества. Свёрточные и рекуррентные нейронные сети были наилучшими моделями для задач компьютерного зрения и обработки текстов. В последнее время наилучшей архитектурой являются Трансформеры, особенно в работе с текстами. Их основная идея заключается в блоках внимания. На данном рисунке кратко представлена схема Vision Transformer (ViT) [4]:

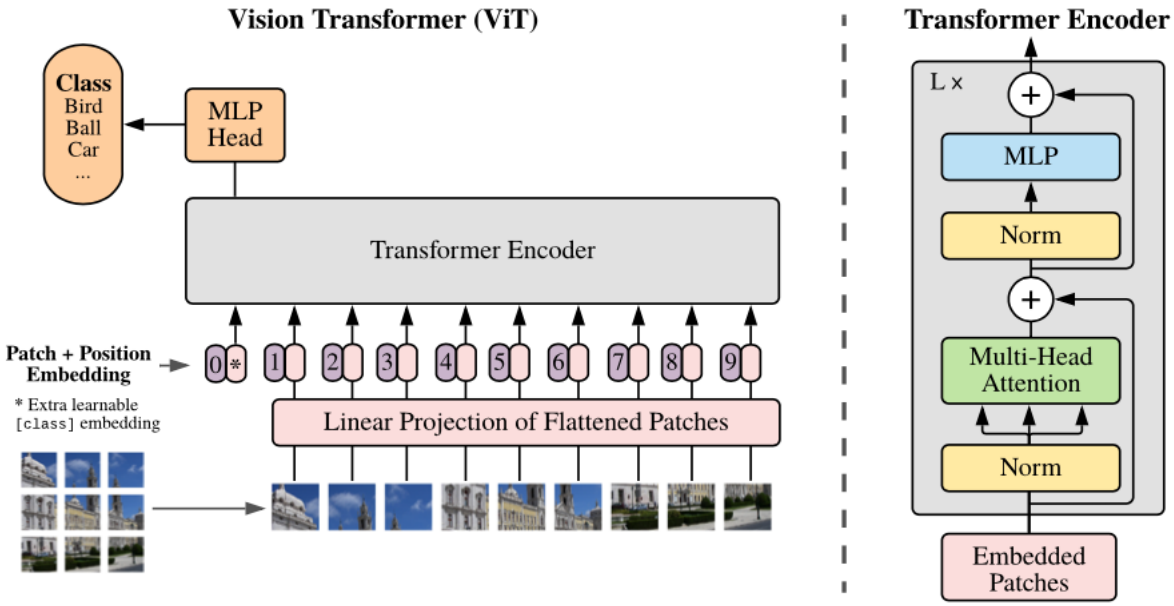


Рис. 1: ViT

В данной работе рассматриваются концептуально простые модели, основанные только на линейных преобразованиях - *MLP-Mixer* и *gMLP*. Они не используют свёртки и механизмы внимания. Тем самым исследуются вопрос о необходимости использования технически сложных моделей трансформеров и свёрточных сетей, чтобы достичь хорошего результата.

MLP модели имеют два различных блока: проекции по каналам и по пространственной переменной. Данные сети способны показать конкурентоспособные результаты для задач классификации текстов и изображений.

Рассматриваемые архитектуры можно применить к задачам из различных областей - компьютерного зрения и обработки естественного языка.

В задаче классификации изображений на вход модели поступает картинка и необходимо определить, что на ней изображено (соотнести некоторому классу). В данной работе рассматривались такие датасеты, как ImageNet, CIFAR-10, CIFAR-100 и Pets.

Задачи классификации текстов отличаются в зависимости от набора данных. Были рассмотрены следующие датасеты. SST-2 состоит из предложений, являющихся комментариями к фильмам. Необходимо определить положительный или негативный они несут оттенок. В наборе данных MNLI каждый объект - это пара предложений. Задача состоит в том, чтобы определить является ли второе следствием первого.

1.1 Постановка задачи

Целью данной работы было исследовать применимость более простых моделей таких, как MLP-Mixer и gMLP, для решения задач компьютерного зрения и обработки текста. Показать, что они при определённых условиях способны получить сравнимое качество с современными архитектурами. Также проводилось исследование по настройке параметров рассматриваемых сетей.

2 MLP-Mixer

2.1 Архитектура

На данном рисунке (Рис. 2) представлена архитектура MLP-Mixer. Рассмотренная нейронная сеть применялась только для задач классификации изображений.

На вход поступают изображения размера $H \times W$. Каждое, из которых разбивается на пересекающиеся патчи размера $P \times P$. Получим, что их общее количество равно $S = \frac{HW}{P^2}$. Все патчи линейно проецируются в вектор размерности C. Данную операцию можно сделать свёрткой с размером ядра P и шагом P. Таким образом, на вход первому блоку поступает матрица $\mathbf{X} \in \mathbb{R}^{S \times C}$. Она представляет из себя последовательность S патчей, каждый из которых представлен векторм с C каналами.

Современные нейронные сети состоят из слоёв, которые способны производить смешивание по каналам и пространству. К примеру, в свёрточных нейронных сетях это можно сделать одновременно, используя размер ядра $N \times N (N > 1)$. Трансформеры так же производят смешивание одновременно за счёт блока внимания (self-

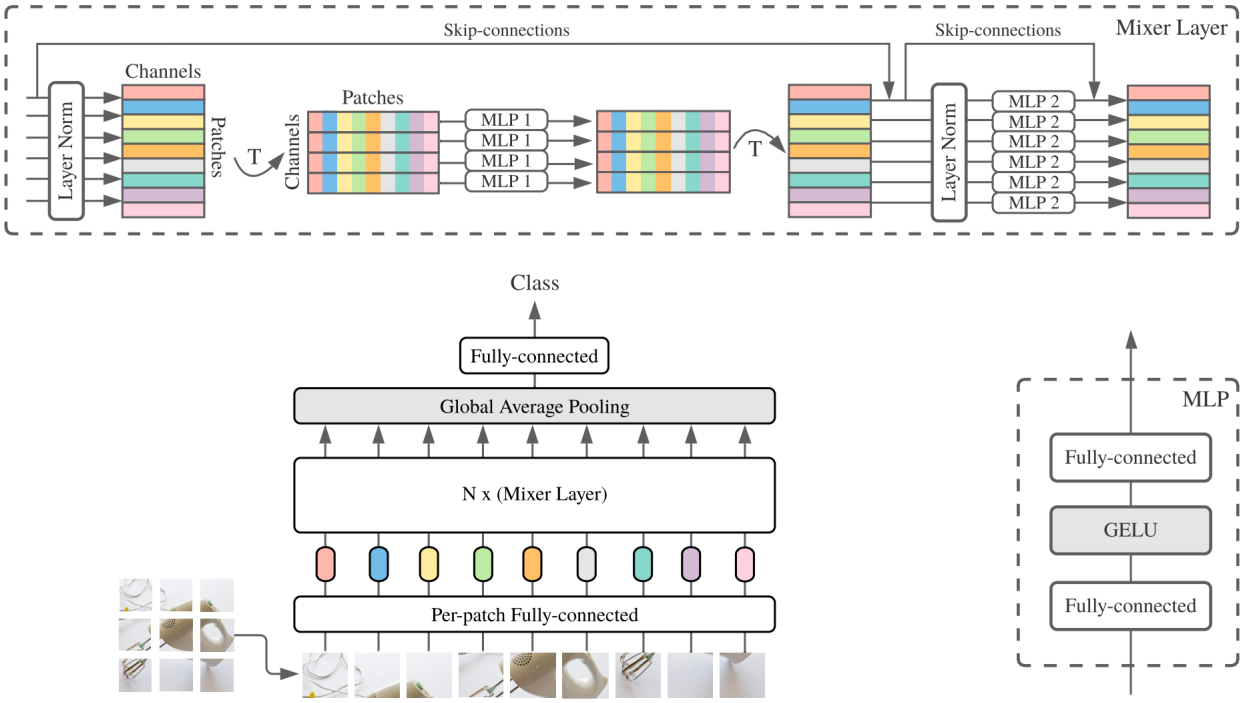


Рис. 2: MLP-Mixer

attention). Идея MLP-Mixer состоит в том, чтобы выполнять данные операции строго по очереди.

Как видно из Рис.2 MLP-Mixer состоит из блоков, каждый из которых представлен двумя видами уровней: проекция по пространству (token-mixing) и по каналам (channel-mixing). Причём выход каждого слоя имеет ту же размерность, что и вход. То есть token-mixing можно представить, как функцию, действующую по следующему правилу: $\mathbb{R}^S \mapsto \mathbb{R}^S$, где все преобразования применяются к каждому столбцу матрицы X независимо. А channel-mixing: $\mathbb{R}^C \mapsto \mathbb{R}^C$. Таким образом, каждый блок MLP-Mixer получает на вход матрицу $X \in \mathbb{R}^{S \times C}$ одного и того же размера. В этом он похож на Трансформеры и рекуррентные нейронные сети. Но отличается от свёрточных архитектур, у которых разрешение изображений уменьшается от блока к блоку, но растёт количество каналов.

На выходе используется стандартный линейный слой классификации, который каждому изображению сопоставляет веса классов.

Каждый MLP блок состоит из двух полносвязных слоёв и нелинейной функции активации между ними:

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

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

Где в качестве функции активации используется σ = GELU:

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

Вход любого блока нормализуется с помощью LayerNorm по каналам. Так же используется механизм пробрасывания связей и сдвиги каждой проекции.

Параметрами каждого слоя являются четыре матрицы: $W_1 \in \mathbb{R}^{D_S \times S}$, $W_2 \in \mathbb{R}^{S \times D_S}$, $W_3 \in \mathbb{R}^{D_C \times C}$, $W_4 \in \mathbb{R}^{C \times D_C}$, где D_S и D_C являются гиперпараметрами модели. Они выбираются независимо от S и C. Но подбираюся так, чтобы были верны неравенства $D_S > S$ и $D_C > C$.

Посмотрим на вычислительную сложность каждого уровня MLP-Mixer. В token-mixing производятся действия над столбцами матрицы X, для всех каналов. Получим всего $2D_SSC$ операций. Аналогично слой channel-mixing выполняет преобразования над строками входной матрицы для каждого патча. Всего имеем $2D_CSC$ операций. Таким образом, получим, что вычислительная сложность MLP-Mixer линейна по количеству каналов и патчей. В то время, как ViT квадратичен по S.

Отметим, что в крайнем случае блок channel-mixing можно рассматривать как 1×1 свёртку, которая имеет более простую реализацию, в отличие от свёрточных нейронных сетей. Так же MLP-Mixer не использует позиционное кодирование, так как слой token-mixing благодаря своей реализации позволяет улавливать связи между различными патчами и учитывает их местоположение.

2.2 Эксперименты

Работа MLP-Mixer была протестирована на различных наборах данных - ImageNet, CIFAR-10, CIFAR-100 и Pets. Чтобы проверить качество модели была использована метрика accuracy, которая равняется доле верных ответов. Было проведено несколько экспериментов с предобученной ранее моделью и обученной с нуля со случайной инициализацией. Предобучение производилось на двух датасетах: ImageNet-21k и JFT-300M. Первый включает в себя 14 миллионов изображений и 21 тыс. меток класса. Последний состоит из 300 миллинов данных and 18 тыс. классов.

Изображения в этих датасетах имеют размер 224×224. Поэтому при настройке модели на других наборах данных, которые имеют другие разрешения, изображения были изменены до нужных размеров. MLP-Mixer был предобучен с разрешением 224, Adam с $\beta_1 = 0.9$, $\beta_2 = 0.999$, линенйным нагревом learning rate на 10 тыс. шагов, размером батча 4096 и нормировкой градиента с нормой 1. Как было эмпирически проверено необходимо добавлять регуляризацию. Иначе Mixer может сильно переобучиться.

Предполагается брать следующие гиперпараметры:

Гиперпараметры	S/16	B/32	B/16	L/32	L/16	H/14
Количество слоёв	8	12	12	24	24	32
Размер патча $P \times P$	16×16	32×32	16×16	32×32	16×16	14×14
Скрытая размерность C	512	768	768	1024	1024	1280
Длина последовательности S	196	49	196	49	196	256
MLP размерность D_C	2048	3072	3072	4096	4096	5120
MLP размерность D_S	256	384	384	512	512	640
Количество параметров(M)	18	60	59	206	207	431

Таблица 1: Спецификация MLP-Mixer

Основной целью данной работы было показать, что обычные MLP модели способны достичь того же качества, что и современные архитектуры, а не показать новый наилучший результат.

В статье [1] сравнения проводились с рядом различных нейросетевых архитектур. Свёрточные нейронные сети: ResNets, NFNet-F4+. Модели, основанные на внимании: HaloNet, ViT-L/16. Эксперименты были запущены на датасетах ImageNet, ReaL, включающих в себя 1.3M изображений и 1 тыс. классов. Также работа сетей была проверена на пяти задачах: ImageNet, CIFAR-10, CIFAR-100, Pets и Flowers. Результат можно увидеть в Таблице 2 в колонке “Avg 5” - среднее на этих пяти датасетах. Были проведены эскперименты на 19 задачах из набора данных VTAB-1k.

В Таблице 2 представлены результаты работы моделей на различных наборах данных.

	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

Таблица 2: Результаты

Как можно видеть, Mixer показывает хорошую точность на всех наборах данных, совсем немного уступая передовым моделям. Заметно, что, предобучившись на более маленьком наборе данных (ImageNet-21k), Mixer работает чуть хуже, чем ViT и свёрточные нейронные сети, уступая более 1% на ImageNet. При этом на VTAB он показывает качество намного лучше. Несмотря на высокую пропускную способность, сравнимую с HaloNet, Mixer-L/16 обучается в несколько раз дольше чем Трансформер.

Видно, что, увеличив размер сети и набора данных, на котором производилось предобучение, Mixer-H/14 показывает сравнимое качество. Он выигрывает у BiT-R152x4 на некоторых задачах, при этом уступает ViT менее процента. Также Mixer остаётся вычислительно эффективной моделью, обрабатывая в несколько раз больше изображений, чем ViT и BiT-R152x4. К тому же, время на обучение MLP модели значительно сокращается по сравнению с другими. Он обучается в 2 раза быстрее, чем трансформер. И обгоняет даже NFNet.

В данной работе была реализована собственная версия MLP-Mixer и проведены эсперименты на различных наборах данных. Нейронная сеть была обучена с ис-

пользованием Adam с β_1= 0.9, β_2 = 0.999, learning rate $6 \cdot 10^{-6}$, размер батча 64 и коэффициент регуляризации 10^{-3}. Результаты можно увидеть в Таблице 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

Таблица 3: Результаты

Как видно из экспериментов полносвязная сеть сильно зависит от предобучения. Если взять предобученную модель и дообучить её на заданную задачу, то она начинает показывать очень хорошие результаты, которые сравнимы с Трансформерами и свёрточными нейронными сетями. На датасете Pets, Mixer даже чуть выигрывает у ResNet152. Если же попытаться запустить модели без предобучения (from scratch), то они показывают результаты хуже. Однако, при прочих равных, MLP начинает сильно уступать современным архитектурам, проигрывая по 15-20%. Эмпирически было получено, что Mixer со случайной инициализацией, лучше работает, если оставлять размеры изображений неизменными, не увеличивая их до 224. Вместе с этим необходимо уменьшить гиперпараметры: $P = 4$, $S = 64$, $C = 256$, $D_C = 1024$, $D_S = 256$.

2.3 Выводы

Таким образом, можно заключить, что MLP-Mixer сильно зависит от предобучения и подбора параметров. Увеличивая learning rate или изменяя ширину слоёв, сеть может перестать обучаться и начать показывать совершенно плохие результаты.

Также из-за простой реализации данной архитектуры слои token-mixing зависят

от длины последовательности S, что является существенным ограничением данной модели. Так как она может работать только с фиксированными размерами изображений. Изменяя разрешение патча, приходится заново переобучать нейронную сеть.

3 gMLP

Рассмотренная нейронная сеть применялась для задач компьютерного зрения и обработки естественного языка. Работа с изображенияеми в точности совпадает с MLP-Mixer [1] и ViT [4]. На входе происходит разбиение картинки на непересекающиеся патчи, которые линейно проецируются в вектора скрытой размерности C. Затем формируется матрица входных данных $\mathbf{X} \in \mathbb{R}^{S \times C}$, которая далее проходит через все блоки сети. На выходе также используется стандартный линейный слой классификации.

Работа с текстами аналогична BERT [3]. Исходное предложение разбивается на S токенов, используя словарь размерности 30 тыс. Далее техникой WordPiece получаются эмбеддинги каждого токена размерности C. Таким образом, создаётся матрица $\mathbf{X} \in \mathbb{R}^{S \times C}$, над которой производятся те же самые манипуляции.

3.1 Архитектура

На данном рисунке представлена схема блока gMLP.

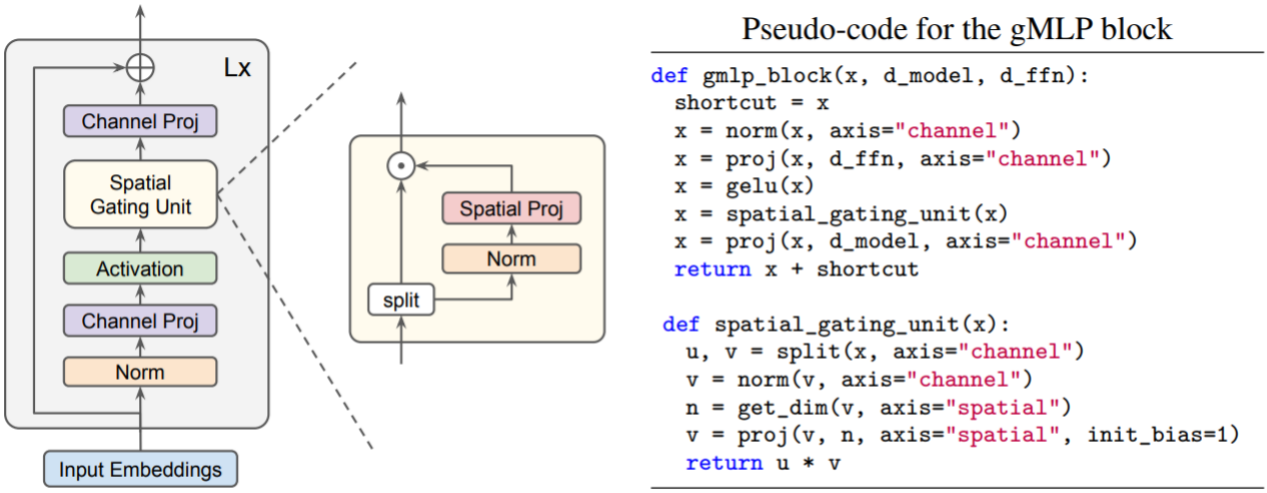


Рис. 3: gMLP

gMLP состоит из нескольких подряд идущих блоков, указанных на рисунке. В отличие от Mixer рассматриваемая архитектура имеет в каждом своём уровне 3 полносвязных слоя. Причём смешивания по каналам и токенам идут не чётко друг за другом, а вперемешку. Блоки gMLP можно описать следующим образом:

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

Где параметрами модели являются матрицы $U \in \mathbb{R}^{C \times D_C}, V \in \mathbb{R}^{D_C \times C}$, которые осуществляют проекцию по каналам (эквивалент channel-mixing в MLP-Mixer). В качестве функции активации используется GELU.

Ключевым местом в данной архитектуре является функция $s(\cdot)$ - SGU (Spatial Gating Unit). Преобразования, описанные ранее, работают с каждым токеном независимо. Чтобы учитывать взаимосвязи между ними необходимо иметь слой, который смешивает патчи (token-mixing в MLP-Mixer). Простейшей реализацией является линейное преобразование. SGU осуществляет проекцию по токенам благодаря функции $f(\cdot)$, а гейтовость модели заключается в $s(\cdot)$:

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

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

где W и b - параметры модели, а $\odot$ - поэлементное умножение. Так как в данной архитектуре используется всего один слой линейной проекции по пространственной переменной (в MLP-Mixer два), то из-за необходимости сохранять размерность (для $s(\cdot)$), матрица W должна быть квадратной. То есть $W \in \mathbb{R}^{S \times S}$, S - длина последовательности.

Изначально W инициализируется значениями близкими к нулю, а b к единице. Таким образом получается, что на первых шагах обучения $f(Z) \approx 1 \quad \Rightarrow \quad s(Z) \approx Z$. Такая инициализация позволяет обрабатывать все токены независимо друг от друга. То есть полносвязный блок на начальных стадиях обучения становится похож на обычный полносвязный слой в блоках Трансформеров.

Было обнаружено, что для улучшения работы нейронной сети можно разбить по каналам матрицу Z, которая приходит на вход SGU, на две независимые части (Z_1, Z_2). В таком случае выход блока определяется следующим способом:

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

Также в данной архитектуре используются сдвиги, нормировки по каналам и механизмы пробрасывания связей, которые ранее не были выписаны, чтобы не загромождать формулы. Аналогично MLP-Mixer gMLP не использует позиционное кодирование, так как данная техника является излишней. Блок SGU, используя проекции по патчам, способен учитывать взаимосвязи и местоположения токенов. Также каждый слой gMLP, как и Mixer, рекуррентные нейронные сети и Трансформеры, принимает на вход матрицу одного и того же размера $S \times C$.

Посмотрим на вычислительную сложность каждого блока gMLP. Два слоя, выполняющие проекции по каналам, аналогично channel-mixing в MLP-Mixer имеют в общей сложности $2D_CSC$ операций. В SGU производятся действия над столбцами матрицы Z, для всех каналов. Так, получается D_CS^2 операций в функции $f(\cdot)$ и D_CS в $s(\cdot)$. Таким образом, получим, что вычислительная сложность gMLP линейна по количеству каналов, но квадратична по токенам. Такую же сложность имеют Трансформеры.

3.2 Эксперименты по обработке изображений

В данном разделе нейронная сеть gMLP была применена к задаче классификации изображений. Исследования были проведены на датасетах ImageNet, CIFAR-10, CIFAR-100 и Pets. Исходные изображения разбивались на неперсекающиеся патчи размера 16×16 для ImageNet (исходное разрешение 224×224) и 4×4 для CIFAR-10, CIFAR-100 (исходное 32×32). Авторы статьи [2] предлагают брать следующие гиперпараметры:

	Количество слоёв	C	D_C	Количество параметров(M)
gMLP-Ti	30	128	768	5.9
gMLP-S	30	256	1536	19.5
gMLP-B	30	512	3072	73.4

Таблица 4: Спецификация gMLP

В Таблице 5 продемонстрированы результаты различных моделей (свёрточные и полносвязные сети, Трансформеры), которые были запущены без предобучения со случайной инициализацией на рассмотренных наборах данных:

Model	ImNet Top-1	CIFAR-10	CIFAR-100	Pets	Params (M)
Свёрточные сети					
ResNet-152	78.3	98.2	87.8	93.3	60
EfficientNet-B7	84.3	–	–	–	66
NFNet-F0	83.6	–	–	–	72
Трансформеры					
ViT-B/16	77.9	98.6	89.1	93.1	86
DeiT-B (ViT+reg)	81.8	–	–	–	86
Полносвязные сети					
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

Таблица 5: CV результаты

На основе Таблиц 2 и 5 можно заключить те же выводы, что и про MLP-Mixer. Полносвязные нейронные сети без предобучения показывают качество намного хуже, чем современные модели. Так же их намного труднее настраивать (подбирать гиперпараметры). Однако, как видно из результатов на ImageNet, иногда MLP модели показывают сравнимое качество с ViT.

3.3 Эксперименты по обработке текстов

В данном разделе изучается применение gMLP к задаче маскированной языковой модели. Все входы и выходы сети аналогичны BERT [3]. Предобучение производилось на датасете C4/RealNews, включающим в себя 5000 различных новостных топиков. Все тексты имеют разную длину, поэтому, чтобы корректно работать с данными и матричными выслениями, в конце добавляется специальный токен $<pad>$ для увеличения длины до максимальной. gMLP был предобучен с AdamW с $\beta_1 = 0.9$, $\beta_2 = 0.999$, линейным нагревом learning rate на 125 тыс. шагов с максимальным значением $7 \cdot 10^{-4}$, размером батча 2048, регуляризацией и максимальной длиной текста 128

токенов.

Авторами статьи [2] были проведены следующие эксперименты, исследующие различные варианты SGU. Каждая gMLP модель имеет 36 блоков, $C = 512$ и $D_C = 3072$:

SGU	Перплексия	Количество параметров (M)
Линеный, $s(Z) = f(Z)$	5.14	92
Сумма, $s(Z) = Z + f(Z)$	4.97	92
Произведение, $s(Z) = Z \odot f(Z)$	4.53	92
Произведение, разбитие, $s(Z) = Z_1 \odot f(Z_2)$	4.35	102

Таблица 6: SGU результаты

Далее было проведено исследование на двух ранее описанных наборах данных: SST-2 и MNLI. Чтобы проверить качество модели была использована метрика accuracy. Результаты работы gMLP и BERT приведены в Таблице 7:

Модель	#L	Количество параметров (M)	Перплексия	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

Таблица 7: NLP результаты

Как можно заметить, MLP модель способна показать сравнимый результат с Трансформерами. Видно, что, увеличивая размеры полносвязной нейронной сети, она становится лучше обучаться на MLM задаче. Перплексия уменьшается с ростом модели. Напротив, на дообучении на различных датасетах gMLP может, как сильно выигрывать у BERT (на SST-2), так и проигрывать (на MNLI). Вероятно, это связано с разными типами задач, решаемых в этих наборах данных, где MNLI является

более сложной.

В данной работе была реализована собственная версия gMLP и проверена на тех же датасетах. Она была обучена с нуля (from scratch) с параметрами предлагаемыми авторами статьи [2]: AdamW с $\beta_1 = 0.9$, $\beta_2 = 0.999$, линейным нагревом learning rate на 5 эпох с максимальным значением $2 \cdot 10^{-5}$, размером батча 32, регуляризацией и максимальной длиной текста 128 токенов.

Было эмпирически замечено, что gMLP всегда выучивает матрицу W в функции $f(\cdot)$ (SGU) подобной Тёплицевой матрице. Это позволяет быть модели инвариантной к сдвигам слов. Также ограничение на тёплицевость позволяет существенно сократить количество обучаемых параметров. Далее модель с этим ограничением именуется как tMLP.

Так как не было предоставлено весов предобученной сети, чтобы избежать этот вычислительно сложный процесс, автор данной работы получил хорошие эмбеддинги слов благодаря ранее предобученному BERT и обучил gMLP на них. Результаты экспериментов представлены в Таблице 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

Таблица 8: NLP результаты

Здесь можно сделать аналогичные выводы, что и для Mixer. Без предобучения MLP модель не способна показать хорошего результата, она сильно отстаёт от BERT (по 10-15%). Если же на вход сети подавать контекстуальные эмбеддинги, то она начинает намного лучше обучаться и показывать качество сравнимое с тем, что получилось в [2], отставая от BERT всего на несколько процентов.

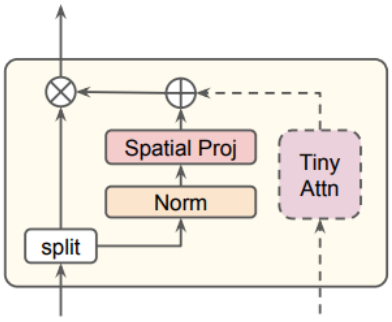
Видно, что если проводить в блоке SGU разбиение матрицы Z на две части по

каналам, то модель ещё лучше обучается и показывает результат немного выше. Также заметно, что ограничение на тёплицевость матриц W в $f(\cdot)$ позволяет не только уменьшить число настраиваемых параметров, но и повысить точность на данных задачах. Всюду далее в gMLP используется разбиение Z и Тёплицевы матрицы.

3.4 gMLP + внимание

Как видно из результатов Таблиц 6-8 блоки внимания не являются необходимым механизмом для достижения хороших результатов. MLP модель иногда даже выигрывает у BERT, показывая качество на несколько процентов лучше (на SST-2). В то же время, полносвязная сеть работает хуже Трансформера на MNLI. Вероятно, это связано с тем, что MNLI состоит из пар предложений и механизм внимания позволяет учитывать это. Для MLP это оказывается намного труднее. Поэтому, чтобы улучшить качество работы сети предлагается использовать маленький блок внимания с одной головкой. gMLP сеть способна и без него хорошо обучиться, поэтому добавляемый блок важен лишь своим наличием. Далее gMLP с блоком внимания обозначается как aMLP.

На данном рисунке представлена схема блока SGU aMLP:



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")
```

Рис. 4: aMLP

3.5 Эксперименты gMLP + внимание

Как можно заметить из результатов, представленных в Таблице 9, большой блок внимания является излишним для достижения хорошего качества. Большинство его функций может быть заключено в линейных проекциях по токенам. Добавление

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

Таблица 9: NLP результаты

лишь незначительного и маленького с одной головкой внимания может оказаться вполне достаточным, чтобы показать качество лучше, чем BERT. Этот блок позволяет, учитывать смещения слов и предложений в тексте.

4 Заключение

В данной работе были реализованы и подробно разобраны полносвязные нейронные сети MLP-Mixer и gMLP. Был изучен вопрос о необходимости слоя внимания для достижения хорошего качества.

Модели способны показывать хорошее качество, иногда даже лучше, чем у передовых архитектур. Увеличивая размер, полносвязной сети, можно существенно сократить отставание от Трансформеров и свёрточных нейронных сетей. Добавляя маленький блок внимания, модель способна достичь ещё более высокого результата, обгоняя при этом Трансформеры.

Несмотря на простоту MLP сетей на и высокое качество на обучении под задачу, их очень тяжело настраивать. Приходится долго подбирать параметры, чтобы получать разумную точность.

Также данные нейронные сети сильно зависимы от предобучения. Было показано, что они не способны достигать таких же хороших результатов, как Трансформеры и свёрточные сети. Кроме того, рассматриваемые архитектуры применимы к узкому кругу задач - классификация текстов и изображений (архитектуры - кодировщики). Непонятно, как сделать, например, языковую модель, не используя блоки кросс-внимания. Это является сильным ограничением данных моделей.

5 Литература

[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. arXiv preprint arXiv:2105.08050, 2021.

[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.