

Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)

Физтех-школа прикладной математики и информатики
Кафедра распознавания изображений и обработки текста (РИОТ)

Выпускная квалификационная работа бакалавра

Аугментации в задаче распознавания рукописного текста

Автор:
Студент 020 группы
Курочкин Павел Сергеевич

Научный руководитель:
Ушинский Андрей Леонидович



Москва 2024

Аннотация

Аугментации в задаче распознавания рукописного текста
Курочкин Павел Сергеевич

В современных методах распознавания рукописного текста ключевым элементом является наличие обширного массива аннотированных данных.

Однако, в случае ограниченности данных, эффективное применение методов аугментаций становится критически важным для повышения точности моделей. Manifold Mixup [1] - современный метод аугментации данных, он представляет собой подход, в котором объединяются два или более изображения или их признаковые карты для создания новых образцов данных. В данном исследовании предлагается метод адаптации Manifold Mixup для использования с подходами, основанными на Connectionist Temporal Classification [2], в контексте задачи распознавания рукописного текста. Проводится всестороннее исследование данного метода, анализируется его влияние на процесс обучения нейронной сети. Результаты исследования показывают значительное улучшение результатов распознавания текста при использовании Manifold Mixup.

Abstract

Augmentations in handwritten text recognition

Содержание

1 Введение	4
1.1 Проблема	4
1.2 Постановка задачи	5
2 Архитектура решения	6
2.1 Извлечение визуальных признаков с помощью сверточной сети	6
2.1.1 Residual connection	7
2.1.2 ResNet	8
2.2 Последовательный энкодер	8
2.2.1 Рекуррентные нейронные сети	8
2.2.2 Self-Attention	9
2.3 Декодер	10
2.3.1 Transformer	10
2.3.2 Connectionist Temporal Classification	10
3 Обзор существующих методов аугментации изображений для задачи распознавания рукописного текста	13
3.1 CutMix	13
3.1.1 StackMix	14
3.2 Mixup	15
3.2.1 Manifold mixup	15
4 Конфигурация	18
4.1 Описание модели	18
4.1.1 ConvBN	18
4.1.2 ResNetBlock	18
4.1.3 FeatureBlock	19
4.2 Описание датасета	19
4.3 Оптимизатор	19
4.4 Реализация mixup	20
5 Эксперименты	21
5.1 Влияние положения Mixup	21
5.2 Зависимость от количества данных	22
6 Результаты экспериментов	23
7 Вывод	24
Приложение	25
7.1 Позиция mixup	25
7.2 Количество данных	28

1 Введение

Распознавание текста является важным этапом в большинстве приложений по анализу изображений документов. Оно позволяет автоматически получать доступ к информации, содержащейся на страницах. За последнее десятилетие произошло огромное улучшение систем распознавания рукописного текста, связанное с появлением новых подходов и архитектур [2], [3], [4], [5], [6], [7], [8].

1.1 Проблема

Использование современных методов нейронных сетей помогает решить проблему высокой стилистической гетерогенности рукописного текста. Однако для таких мощных моделей с большим количеством параметров требуется значительное количество аннотированных изображений для обучения. Было предложено несколько методов, позволяющих уменьшить необходимость аннотированных данных при обучении систем распознавания текста.

Во-первых, обучающий набор можно расширить, добавив синтетические изображения. Это можно сделать, используя рукописные шрифты [9] или создавая изображения текстовых строк на основе индивидуально извлеченных изображений реальных букв [10]. Кроме того, для создания рукописных текстов можно применять генеративно-состязательные сети [11].

Также, при ограниченном объеме данных, необходимо бороться с проблемой переобучения. Для этого было предложено множество подходов к регуляризации, таких как weight decay, dropout [12], batch normalization [21] или раннее прекращение обучения, которые могут быть использованы в процессе обучения сети.

Наконец, для увеличения разнообразия обучающего набора, улучшения обобщающей способности модели и снижения риска переобучения широко применяются методы аугментации данных. Изображения могут быть отражены горизонтально и вертикально, подвергнуты поворотам и угловым трансформациям, изменены по размеру и масштабу. Также возможно добавление шума, обрезка изображений, цветовые трансформации и искажения в различных формах [13], [5]. Подробнее методы аугментации для задачи распознавания рукописного текста описаны в Глава 3.

В [14] для создания новых образцов было предложено интерполировать признаки образцов одного класса. В [15] было предложено использовать данный подход для признаков образцов на выходе промежуточного слоя нейронной сети. В [1] объединяются обе эти идеи и предлагается смешивать целевую переменную и изображения из разных классов, или их промежуточные признаки на различных уровнях сети. Последний подход показывают значительное улучшение качества на новых и состязательных данных. Последний подход был назван Manifold Mixup. В [16] представлен способ адаптации данного подхода для работы с решениями, основанными на Connectionist Temporal Classification [2].

Цель данного исследования заключается в анализе методов аугментации изображений в контексте распознавания рукописного текста и более детальном рассмотрении различных аспектов метода Manifold Mixup. Выбор использования Manifold Mixup обоснован следующими уникальными характеристиками данного метода:

1. Его легкость в обобщении на широкий спектр задач компьютерного зрения, решаемых с помощью нейронных сетей.
2. Согласно результатам исследований [1], [16], данный метод значительно способствует качеству и обобщенности обученной модели.

1.2 Постановка задачи

Рассматриваемые методы аугментации изображений, включая метод Manifold Mixup [1], исследуемый в данной работе, предназначены для преодоления следующих проблем, возникающих в задаче распознавания рукописного текста:

1. Разнообразие изображений рукописного текста из-за различий в стилях авторов и фонах документов.
2. Недостаток аннотированных данных, что затрудняет обобщение моделей.
3. Риск переобучения из-за упомянутых выше проблем и большого количества параметров в сети.

В данной статье проводится анализ различных методов аугментаций, решающих данные проблемы. А также исследуются следующие аспекты метода Manifold Mixup:

1. Методы формирования батчей из изображений различных длин и их влияние на стабильность процесса обучения.
2. Роль выбора промежуточного слоя в процессе обучения для различных архитектур.
3. Влияние размера батча на обучение.
4. Воздействие Manifold Mixup на процесс обучения в зависимости от объема имеющихся данных.

В Главе 2 приводится обзор существующих архитектур для решения задачи распознавания рукописного текста. Также представлена архитектура, применяемая в данном исследовании, обосновывается выбор данной архитектуры.

Глава 3 посвящена обзору существующих методов аугментации изображений для задачи распознавания рукописного текста, включая Manifold Mixup. Оценивается эффективность их применения для решения вышеупомянутых задач.

В Главе 4 и Главе 5 представлено описание практической части: конфигурация и эксперименты соответственно.

Глава 6 анализирует различные аспекты метода Manifold Mixup на основе проведенных экспериментов

И, наконец, в Главе 7 содержатся выводы из проведенного исследования.

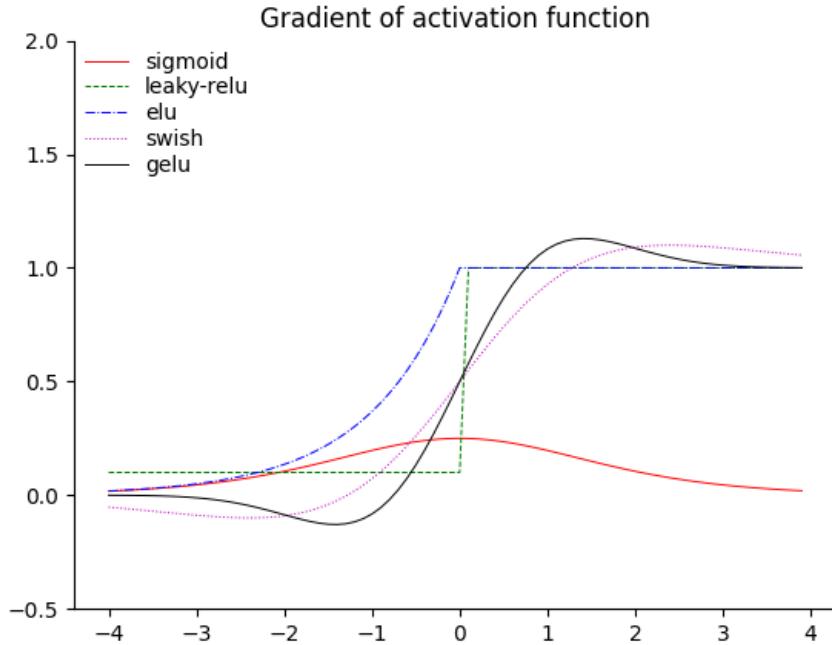


Рис. 1: Градиенты некоторых популярных функций активации.

2 Архитектура решения

Современные модели, применяемые в задаче распознавания рукописного текста, обычно включают в себя три ключевых компонента:

1. Сверточную нейронную сеть, используемую для извлечения признаков из входных данных.
2. Последовательный энкодер
3. Декодер, который завершает процесс, трансформируя закодированные данные в окончательную текстовую транскрипцию с учетом информации, полученной от энкодера.

2.1 Извлечение визуальных признаков с помощью сверточной сети

Подобно другим задачам компьютерного зрения, сверточная сеть используется для извлечения соответствующих визуальных признаков из текстовых изображений. Следующие архитектуры являются популярным выбором: ResNet [17], Inception [18], MobileNet [19]. Увеличение сложности сверточной сети, как правило, приводит к умеренному приросту точности модели [20]. В этой работе используется архитектура ResNet.

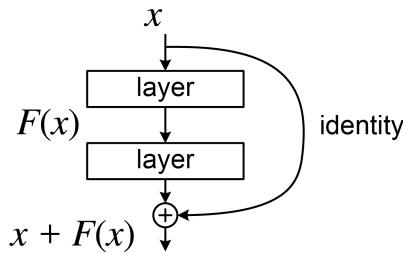


Рис. 2: Residual блок.

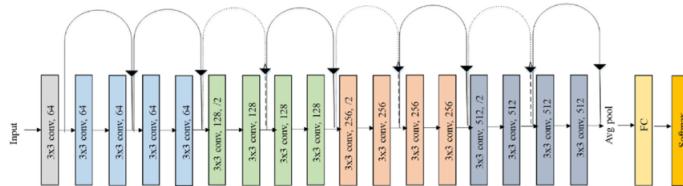


Рис. 3: Архитектура ResNet-18.

2.1.1 Residual connection

При обучении глубоких моделей градиент имеет тенденцию становиться очень маленьким: это называется проблема исчезающего градиента. Это связано с тем, что градиент проходит через ряд слоев, каждый из которых может уменьшить его. Например, градиент многих популярных функций активации приближается к нулю на значительной части числовой прямой [Рис 1].

Одним из решений проблемы исчезающего градиента является использование в качестве слоев нейронной сети residual блока [Рис 2], определенного следующим образом:

$$\mathcal{F}'_l(x) = \mathcal{F}_l(x) + x \quad (1)$$

где \mathcal{F}_l - стандартное нелинейное отображение (например, линейное - функция активации - линейное). Зачастую легче научиться предсказывать небольшое возмущение на входе, чем результат напрямую.

Модель с residual блоком имеет такое же количество параметров, как и модель без него, но ее легче тренировать. Причина в том, что градиенты могут перетекать непосредственно из выходных данных в более ранние слои. Чтобы убедиться в этом, рассмотрим градиент функции ошибки по параметрам слоя l . Имеем

$$z_L = z_l + \sum_{i=l}^{L-1} \mathcal{F}_i(z_i; \theta_i) \quad (2)$$

где z_i - признаки на выходе из i -го слоя сети. Таким образом, мы можем вычислить градиент функции потерь относительно параметров l -го слоя следующим образом:

$$\frac{\partial \mathcal{L}}{\partial \theta_l} = \frac{\partial z_l}{\partial \theta_l} \frac{\partial \mathcal{L}}{\partial z_L} \left(1 + \sum_{i=l}^{L-1} \frac{\partial \mathcal{F}_i(z_i; \theta_i)}{\partial z_l} \right) \quad (3)$$

Таким образом, мы видим, что градиент на слое l напрямую зависит от градиента на слое L , причем независимо от глубины сети.

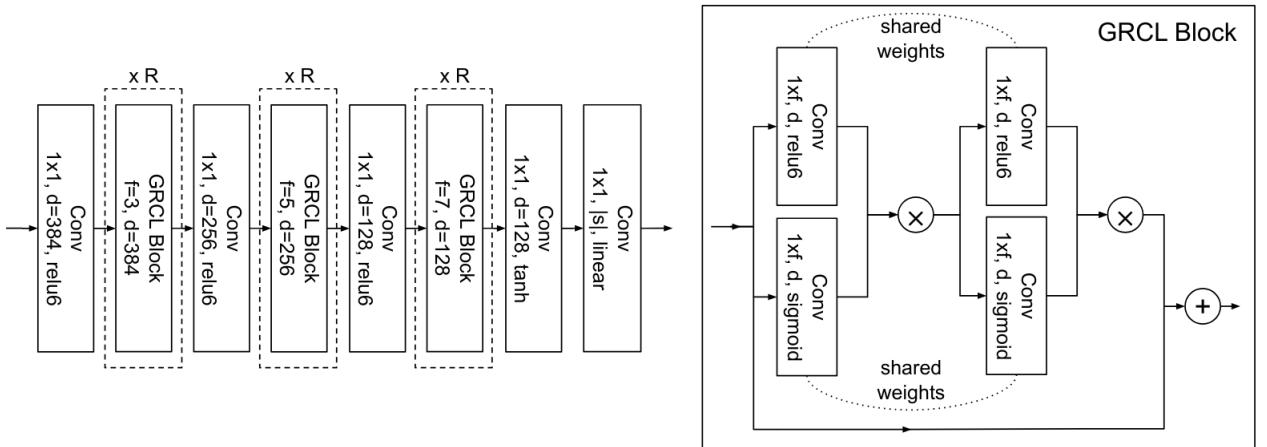


Рис. 4: Архитектура GRCL.

2.1.2 ResNet

Победителем конкурса по компьютерному зрению ImageNet 2015 года стала команда Microsoft, предложившая модель, известную сейчас как ResNet [Рис 3]. Данная модель состоит из residual блоков, имеющих следующий вид: свертка-BN-relu-свертка-BN, где BN - батч нормализация [21]). Подобная архитектура позволяет обучать очень глубокие модели, а также решает проблему затухания градиентов. Именно из-за указанных преимуществ, а также из практического опыта, данная архитектура была отобрана для проведения данного исследования.

2.2 Последовательный энкодер

Сврточная сеть, предназначенная для извлечения визуальных признаков, имеет ограниченное рецептивное поле, что ограничивает ее способность учитывать широкий контекст информации. Это создает сложности при обработке длинных последовательностей в сложных сценариях, таких как распознавание рукописного текста. Для учета контекста на больших расстояниях используется энкодер. Существует много различных архитектур энкодера, далее будут рассмотрены основные из них.

2.2.1 Рекуррентные нейронные сети

Рекуррентная нейронная сеть — это нейронная сеть, которая отображает входное пространство последовательности в выходное пространство последовательностей с сохранением состояния. То есть элемент выходной последовательности y_t зависит не только от элемента входной последовательности x_t , но и от скрытого состояния системы h_t , которое обновляется. Для простоты обозначений пусть Т будет длиной вывода (с учетом того, что она выбирается динамически). Тогда рекуррентная сеть соответствует следующей условной генеративной модели:

$$p(y_{1:T}|x) = \sum_{h_{1:T}} p(y_{1:T}, h_{1:T}|x) = \sum_{h_{1:T}} \prod_{t=1}^T p(y_t|h_t)p(h_t|h_{t-1}, y_{t-1}, x) \quad (4)$$

где h_t — скрытое состояние, и где мы определяем $p(h_1|h_0, y_0, x) = p(h_1|x)$ как начальное скрытое состояние. Мы предполагаем, что скрытое состояние вычисляется детерминированно следующим образом:

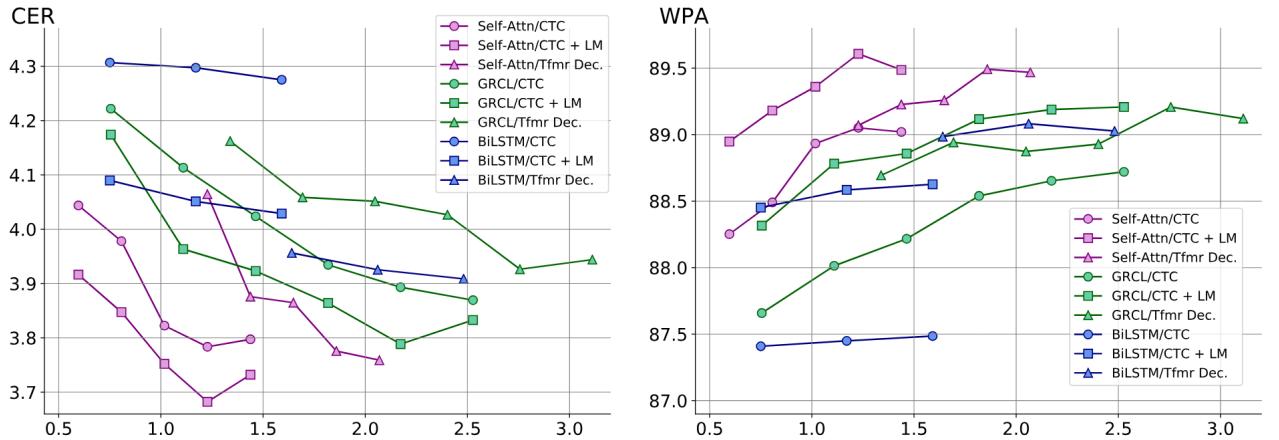


Рис. 5: Сравнение различных архитектур для задачи распознавания текста.
Взято из [20].

$$p(h_t | h_{t-1}, y_{t-1}, x) = \mathbb{I}(h_t = f(h_{t-1}, y_{t-1}, x)) \quad (5)$$

для некоторой детерминированной функции f . Функция обновления f обычно задается выражением

$$h_t = \varphi(W_{xh}[x; y_{t-1}] + W_{hh}h_{t-1} + b_h) \quad (6)$$

Рекуррентные сети с достаточным количеством скрытых единиц в принципе могут запоминать входные данные из далекого прошлого. Однако сети с "ванильной" архитектурой не могут этого сделать из-за проблемы исчезающего градиента. Существует архитектурное решение данной проблемы, в котором мы обновляем скрытое состояние аддитивным способом, аналогично residual блокам в ResNet: GRU и LSTM.

Различные варианты рекуррентных сетей использовались для решения задачи распознавания рукописного текста: LSTM [5] [4], BiLSTM [6] [7], Gated Recurrent Convolution Layer (GRCL) [Рис 4] [22].

2.2.2 Self-Attention

Self-Attention энкодер, предложенный в [3] широко используется в задачах из NLP и компьютерного зрения. Распознавание текстовых строк, как задача преобразования изображения в последовательность, не является исключением. Self-attention способен улавливать долгосрочные зависимости в последовательностях лучше, чем рекуррентные сети, благодаря возможности обрабатывать взаимодействия между всеми элементами последовательности одновременно. Также, в отличие от рекуррентных сетей, self-attention не сталкивается с проблемой затухания градиентов.

Выход сверточной сети, с удаленным измерением высоты изображения ($X \in \mathbb{R}^{n \times d}$), поступает в энкодер. Выход Y слоя Attention получается следующим образом:

$$\begin{aligned} Q &= XW_q \\ K &= XW_k \\ V &= XW_v \\ Y &= \text{softmax}\left(\frac{QK^T}{\sqrt{d}}V\right) \end{aligned} \quad (7)$$

где W_q , W_k и W_v — матрицы параметров размера $d \times d$, которые задают проекцию входной последовательности X в пространство запросов, ключей и значений соответственно. Закодированный признак Y представляет собой выпуклую комбинацию вычисленных значений V , матрица сходства вычисляется как скалярное произведение запросов и ключей.

Эффективность "ванильного" Attention может быть низкой, поскольку данный слой инвариантен к перестановкам, и, следовательно, игнорирует порядок элементов входной последовательности. Чтобы преодолеть эту проблему, к признакам элементов входной последовательности добавляется информация о позиции элемента (positional embedding). Можно представить positional embedding как матрицу $\mathbf{P}^{n \times d}$.

В [3] было предложено использовать синусоидальный базис следующего вида:

$$p_{i,2j} = \sin\left(\frac{i}{C^{\frac{2j}{d}}}\right), \quad p_{i,2j+1} = \cos\left(\frac{i}{C^{\frac{2j}{d}}}\right) \quad (8)$$

где C соответствует некоторой максимальной длине последовательности. Одним из важных плюсов данного представления является то, что представление одной позиции линейно предсказуемо относительно любой другой, если известно их относительное расстояние. В частности, выполняется $p_{t+\phi} = f(p_t)$, где f - некоторое линейное отображение. А именно

$$\begin{pmatrix} \sin(\omega_k(t + \phi)) \\ \cos(\omega_k(t + \phi)) \end{pmatrix} = \begin{pmatrix} \cos(\omega_k\phi) & \sin(\omega_k\phi) \\ -\sin(\omega_k\phi) & \cos(\omega_k\phi) \end{pmatrix} \begin{pmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{pmatrix} \quad (9)$$

То есть при маленьком ϕ имеем $p_{t+\phi} \approx p_t$. Positional embedding, как правило, прибавляется ко входу, то есть: $PosEmbed(X) = X + \mathbf{P}$.

Существуют также другие варианты. Например, relative positional embedding [23].

2.3 Декодер

Декодеры берут признаки закодированной последовательности и пытаются декодировать ее текстовое содержимое. Архитектуры декодеров для рекуррентных сетей были описаны ранее. Два других популярных подхода к решению этой задачи: transformer [3] и Connectionist Temporal Classification [2] декодеры.

2.3.1 Transformer

Декодер Transformer стал предпочтительным декодером для задач прогнозирования последовательности, таких как машинный перевод. Он также широко применяется в задаче распознавания рукописного текста. Например, state-of-the-art решение согласно бенчмарку IAM принадлежит на момент написания этого текста decoder-only transformer архитектуре [8]. Минус данной архитектуры - это то, что она является более громоздкой. Она требует большего количества параметров и, соответственно, данных для обучения [Рис 5].

2.3.2 Connectionist Temporal Classification

Декодер Connectionist Temporal Classification изначально использовался для распознавания речи, позже исследователи добились огромного успеха, применив его для задач распознавания текста. Далее представлен математический формализм временной классификации, а также ошибки, используемой в качестве метрики в данном исследовании.

	Rect.	Encoder / Decoder	Train Dataset	IAM ↓	RIMES ↓	IIIT5K ↑	SVT ↑	IC13 ↑	IC15 ↑
Bluche and Messina [6]	No	GCRNN/CTC	IAM (50k lexicon)	3.2	1.9	-	-	-	-
Michael et al. [37]	No	LSTM/LSTM w/Attn	IAM	4.87	-	-	-	-	-
Kang et al. [28]	No	Transformer	IAM	4.67	-	-	-	-	-
Bleeker and de Rijke [4]	No	Transformer	MJ + ST	-	-	94.7	89.0	93.4	75.7
Lu et al. [34]	No	Global Context Attn / Tfmr Dec.	MJ + ST + SA	-	-	95.0	90.6	95.3	79.4
Qiao et al. [42]	No	LSTM / LSTM + Attn	MJ + ST	-	-	94.4	90.1	93.3	77.1
Sheng et al. [45]	Yes	S-Attn / Attn	MJ + ST	-	-	93.4	89.5	91.8	76.1
Shi et al. [48]	Yes	LSTM / LSTM + Attn	MJ + ST	-	-	91.93	93.49	89.75	-
Wang et al. [55]	No	FCN / GRU	MJ + ST	6.4	2.7	94.3	89.2	93.9	74.5
SCATTER [32]	Yes	CNN / BiLSTM	MJ + ST + SA	-	-	93.9	92.7	94.7	82.8
Yu et al. [57]	No	Attn / Semantic Attn.	MJ + ST	-	-	94.8	91.5	95.5	82.7
Ours	No	S-Attn / CTC	MJ + ST + Public	-	-	92.06	87.94	92.15	72.36
	No	S-Attn / CTC + LM	MJ + ST + Public	-	-	93.63	91.50	93.61	74.77
	No	Transformer	MJ + ST + Public	-	-	93.93	92.27	93.88	77.08
	No	S-Attn / CTC	Internal	4.62	10.80	96.26	91.96	94.43	78.43
	No	S-Attn / CTC + LM	Internal	3.15	7.79	96.83	94.59	95.98	80.36
	No	Transformer	Internal	3.99	9.71	96.54	92.59	94.34	79.68
	No	S-Attn / CTC	Internal + Public	3.53	2.48	95.66	91.34	93.70	78.38
	No	S-Attn / CTC + LM	Internal + Public	2.75	1.99	96.43	93.66	95.25	79.92
	No	Transformer	Internal + Public	2.96	2.01	96.44	92.50	93.97	80.45

Рис. 6: Сравнение различных архитектур для задачи распознавания текста на различных бенчмарках.

Взято из [20].

Пусть S - это набор обучающих примеров, выбранных из фиксированного распределения $D_{\mathcal{X} \times \mathcal{Z}}$. Пространство входных данных $\mathcal{X} = (\mathbb{R}^m)^*$ представляет собой множество всех последовательностей из m -мерных векторов вещественных чисел. Целевое пространство $\mathcal{Z} = L^*$ представляет собой множество всех последовательностей из (конечного) алфавита L . В общем случае, мы обозначаем элементы L^* как последовательности символов. Каждый пример в S состоит из пары последовательностей (x, z) . Целевая последовательность $z = (z_1, z_2, \dots, z_U)$ не длиннее входной последовательности $x = (x_1, x_2, \dots, x_T)$, то есть $U \leq T$. Поскольку входные и целевые последовательности обычно не имеют одинаковой длины, нет априорного способа их выравнивания.

Цель состоит в использовании S для обучения временного классификатора $h : \mathcal{X} \rightarrow \mathcal{Z}$, который классифицирует ранее не виденные входные последовательности таким образом, чтобы минимизировать некоторую специфическую ошибку задачи.

Общепризнанным выбором является следующая ошибка: для заданного тестового набора $S_0 \subset D_{\mathcal{X} \times \mathcal{Z}}$, не пересекающегося с S , определяется коэффициент ошибок меток (label error rate, LER) временного классификатора h как нормализованное расстояние Левенштейна между его предсказаниями и ответом на S_0 , т.е.:

$$LER(h, S_0) = \frac{\sum_{(x,z) \in S_0} ED(h(x), z)}{\sum_{(x,z) \in S_0} |z|} \quad (10)$$

где $ED(p, q)$ - расстояние Левенштейна между двумя последовательностями p и q , т.е. минимальное количество вставок, замен и удалений, необходимых для преобразования p в q . Это естественная метрика для задач (таких как распознавание речи или рукописного текста), где целью является минимизация частоты ошибок транскрипции. Популярными метриками в задаче распознавания рукописного текста являются WER (word error rate) и CER (character error rate) - описанное выше расстояние Левенштейна на уровне слов и символов соответственно.

Также в данном исследовании будет использоваться метрика LabelAccuracy (Label может быть словом, строкой или фрагментом). Определяется она так:

$$LabelAcc(h, S_0) = 1 - LER(h, S_0) \quad (11)$$

Пусть мы имеем некоторую нейронную сеть $N_w : (\mathbb{R}^m)^T \rightarrow (\mathbb{R}^n)^T$. Пусть $y = N_w(x)$ - последовательность выходов сети, и обозначим y_t^k активацию выходного узла k в момент времени t (последний слой softmax). Тогда y_t^k интерпретируется как вероятность наблюдения символа k в момент времени t , что определяет распределение над множеством L_0^T длины T последовательностей алфавита $L_0 = L \cup \{\text{blank}\}$. Тут мы добавляем в алфавит символ *blank*, обозначающий пропуск. Элементы L_0^T обычно называют путями.

$$p(\pi|x) = \prod_{t=1}^T y_t^{\pi_t}, \quad \forall \pi \in L_0^T \quad (12)$$

Далее определим отображение $B : L_0^T \rightarrow L_{\leq T}$, где $L_{\leq T}$ - это множество последовательностей длиной менее или равной T по оригинальному алфавиту L . Мы делаем это, просто удаляя все пробелы и повторяющиеся символы из путей. Интуитивно это соответствует выводу нового символа, когда сеть переходит от предсказания отсутствия символа к предсказанию символа, или от предсказания одного символа к другому. Наконец, используем B для определения условной вероятности данного предсказания $I \in L_{\leq T}$ как сумму вероятностей всех путей, соответствующих ему:

$$p(I|x) = \sum_{\pi \in B^{-1}(I)} p(\pi|x) \quad (13)$$

Вывод классификатора должен быть наиболее вероятным предсказанием для входной последовательности:

$$h(x) = \arg \max_{I \in L_{\leq T}} p(I|x) \quad (14)$$

В данном исследовании поиск такого предсказания строится на предположении, что наиболее вероятный путь будет соответствовать наиболее вероятному предсказанию:

$$\begin{aligned} h(x) &\approx B(\pi^*) \\ \pi^* &= \arg \max_{\pi \in L_0^T} p(\pi|x). \end{aligned} \quad (15)$$

Это легко найти, так как π^* представляет собой просто конкатенацию самых вероятных символов на каждом этапе. Такой подход не гарантирует нахождение наиболее вероятного предсказания, но достаточно хорошо его приближает.

Модель обучается методом максимального правдоподобия. Для вычисления условных вероятностей $p(I|x)$ отдельных предсказаний используется динамическое программирование [2]. Также включение явной языковой модели поверх логитов может значительно повысить точность [24].

Во время выбора архитектуры учитывался собственный практический опыт, а также результаты исследований по данной теме. Например, в [20] проводится сравнительный анализ различных архитектур энкодеров / декодеров [Рис 6]. Было решено выбрать в качестве:

1. Сверточной нейронной сети: ResNet
2. Последовательного энкодера: TransformerEncoder
3. Декодера: Connectionist Temporal Classification

	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	78.6 (+2.3)
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	47.3 (+1.0)
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	76.7 (+1.1)

Рис. 7: Результаты различных техник аугментаций на ImageNet.
Взято из [26].

3 Обзор существующих методов аугментации изображений для задачи распознавания рукописного текста

Помимо распространенных техник аугментации, которые используются в широком спектре задач компьютерного зрения, таких как:

1. отражение по горизонтали/вертикали,
2. повороты и угловые трансформации,
3. изменение масштаба,
4. добавление шума,
5. обрезка,
6. цветовые преобразования,
7. искажения различных форм,

существуют также методы аугментации, которые специфически применяются для решения определенных задач. Далее будут описаны некоторые из таких методов аугментации. В каждом из рассмотренных видов аугментации новые объекты создаются путем комбинирования двух или более объектов из набора данных.

3.1 CutMix

Пусть $x \in \mathbb{R}^{W \times H \times C}$ и y обозначают изображение и его целевую переменную соответственно. Целью метода CutMix является создание нового образца (\tilde{x}, \tilde{y}) путем комбинирования двух образцов (x_A, y_A) и (x_B, y_B) . [Рис 7.] Созданный образец (\tilde{x}, \tilde{y}) используется для обучения модели с использованием ее исходной функции потерь. Операция комбинирования определяется следующим образом:

$$\begin{aligned}\tilde{x} &= M \odot x_A + (1 - M) \odot x_B \\ \tilde{y} &= \lambda y_A + (1 - \lambda) y_B,\end{aligned}\tag{16}$$

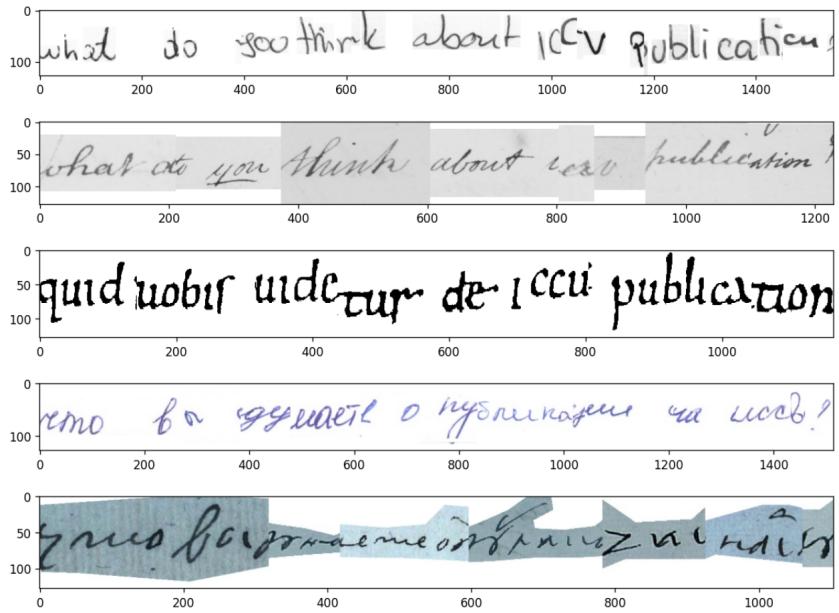


Рис. 8: Примеры изображений, созданных с помощью StackMix.
Взято из [27].

где $M \in \{0, 1\}^{W \times H}$ обозначает бинарную маску, 1 - бинарная маска, заполненная единицами, и \odot - покомпонентное умножение. λ выбирается из бета-распределения $Beta(\alpha, \alpha)$. В отличие от mixup [25], CutMix [26] заменяет область изображения патчем из другого обучающего изображения и создает более локально естественное изображение.

Для выборки бинарной маски M мы сначала выбираем координаты ограничивающего прямоугольника $B = (r_x, r_y, r_w, r_h)$. Область B в x_A удаляется и заполняется частью, обрезанной из B на x_B . Выбираются прямоугольные маски M , соотношение сторон которых пропорционально оригинальному изображению. Координаты прямоугольников выбираются равномерно в соответствии со следующим правилом:

$$\begin{aligned} r_x &\sim Unif(0, W), r_w = W\sqrt{1 - \lambda} \\ r_y &\sim Unif(0, H), r_h = H\sqrt{1 - \lambda} \end{aligned} \quad (17)$$

соотношение площадей картинок есть

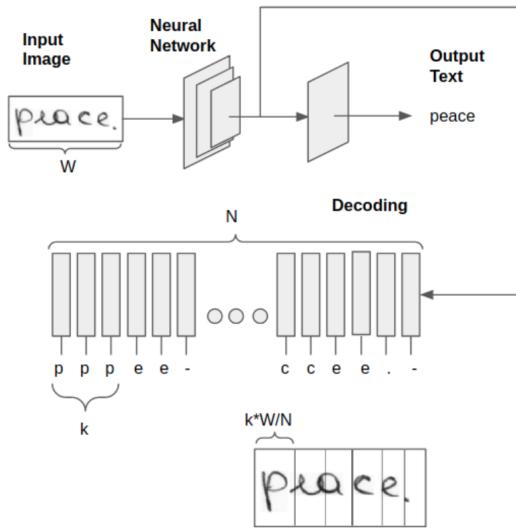
$$\frac{r_w r_h}{WH} = 1 - \lambda \quad (18)$$

На каждой итерации обучения CutMix-образец (\tilde{x}, \tilde{y}) генерируется путем комбинирования случайно выбранных двух образцов в батче.

3.1.1 StackMix

В [27] этот подход был адаптирован для задачи распознавания рукописного текста. В StackMix различные фрагменты рукописного текста склеиваются в некотором порядке. [Рис 8.]

Чтобы применить предложенный подход к задаче распознавания рукописного текста, требуется дополнительная разметка данных, которая точно отмечает границы символов. Для достижения этой цели используется подход с автоматической сегментацией обучающих изображений на символы с использованием постобработки контролируемой предварительно обученной нейронной сети (с декодером Connectionist Temporal Classification).



	Datasets					
	BenthamR0			IAM-B		
	CER, %	WER, %	ACC, %	CER, %	WER, %	ACC, %
base	2.99 ± 0.06	11.8 ± 0.3	52.1 ± 0.8	5.80 ± 0.08	18.9 ± 0.2	29.3 ± 0.4
augs	2.85 ± 0.05	11.3 ± 0.1	52.8 ± 1.0	5.43 ± 0.04	17.8 ± 0.1	30.7 ± 0.5
blots	2.27 ± 0.04	9.5 ± 0.2	57.0 ± 0.4	4.59 ± 0.03	15.0 ± 0.1	36.4 ± 0.5
stackmix	2.08 ± 0.03	9.0 ± 0.1	58.2 ± 0.8	4.90 ± 0.07	16.4 ± 0.2	35.2 ± 0.5
all	1.73 ± 0.02	7.9 ± 0.1	61.9 ± 1.1	3.77 ± 0.06	12.8 ± 0.2	43.6 ± 0.6

	Digital Peter						IAM-D		
	CER, %			WER, %			ACC, %		
	CER, %	WER, %	ACC, %	CER, %	WER, %	ACC, %	CER, %	WER, %	ACC, %
base	4.44 ± 0.02	24.3 ± 0.2	43.7 ± 0.5	4.55 ± 0.06	14.5 ± 0.2	35.5 ± 0.7			
augs	4.15 ± 0.04	23.0 ± 0.2	45.7 ± 0.4	4.38 ± 0.04	14.0 ± 0.2	36.3 ± 0.6			
blots	3.39 ± 0.04	19.3 ± 0.4	51.9 ± 0.4	3.70 ± 0.03	11.8 ± 0.1	42.3 ± 0.6			
stackmix	3.40 ± 0.05	19.2 ± 0.3	51.6 ± 0.7	3.77 ± 0.04	12.3 ± 0.1	42.4 ± 0.8			
all	2.50 ± 0.03	14.6 ± 0.2	60.8 ± 0.8	3.01 ± 0.02	9.8 ± 0.1	50.7 ± 0.3			

	HKR			Saint Gall		
	CER, %	WER, %	ACC, %	CER, %	WER, %	ACC, %
base	6.71 ± 0.18	22.5 ± 0.3	71.1 ± 0.5	4.71 ± 0.05	32.5 ± 0.3	2.2 ± 0.5
augs	6.67 ± 0.24	21.5 ± 0.3	72.2 ± 0.2	4.49 ± 0.11	31.3 ± 0.5	3.4 ± 0.7
blots	7.40 ± 0.26	23.0 ± 0.5	72.6 ± 0.4	4.08 ± 0.03	28.1 ± 0.2	4.6 ± 1.1
stackmix	3.69 ± 0.13	14.4 ± 0.4	80.0 ± 0.3	4.06 ± 0.12	28.8 ± 0.8	6.1 ± 0.8
all	3.49 ± 0.08	13.0 ± 0.3	82.0 ± 0.5	3.65 ± 0.11	26.2 ± 0.6	11.8 ± 2.0

Рис. 9: Схема постобработки для получения границ символов в StackMix. А также результаты работы.

Взято из [27].

Основная идея заключалась в том, чтобы соединить последний слой RNN (после применения активации softmax для каждого символа из функций изображения) и ширину изображения, чтобы получить границы символов, используя только слабо контролируемое обучение без какой-либо ручной разметки [Рис 9.]. Для обучения нейронной сети можно использовать базовую схему без каких-либо дополнений и ухищрений. Для получения качественной разметки границ символов необходимо использовать выборку с этапа обучения.

Один из недостатков данного метода состоит в сильной зависимости от качества алгоритма, определяющего границы символов. Генерированные образцы отличаются именно в местах "склейки". В [27] одним из свойств датасетов является четкость границы между символами в среднем (это было замечено эмпирически после их анализа). Однако не все рукописные тексты обладают таким качеством.

3.2 Mixup

В данном методе предлагается создавать новый образец (\tilde{x}, \tilde{y}) следующим образом:

$$\begin{aligned}\tilde{x} &= \lambda x_i + (1 - \lambda)x_j \\ \tilde{y} &= \lambda y_i + (1 - \lambda)y_j\end{aligned}\quad (19)$$

где (x_i, y_i) и (x_j, y_j) - это два пары (x - вектор признаков, y - целевой объект), выбранные случайным образом из обучающих данных, а $\lambda \sim Beta(\alpha, \alpha)$, $\alpha \in (0, \infty)$, $\lambda \in [0, 1]$. Гиперпараметр α контролирует силу интерполяции между парами признаков-целевых объектов.

Mixup можно понимать как форму аугментации, которая побуждает модель вести себя линейно между по обучающим примерам. Утверждается, что такое линейное поведение уменьшает количество нежелательных колебаний при прогнозировании вне обучающих примеров. Обобщением этой идеи является manifold mixup.

3.2.1 Manifold mixup

В [1] исследователи обнаружили несколько свойств, касающихся скрытых представлений и границ решений современных нейронных сетей. Во-первых, граница решения

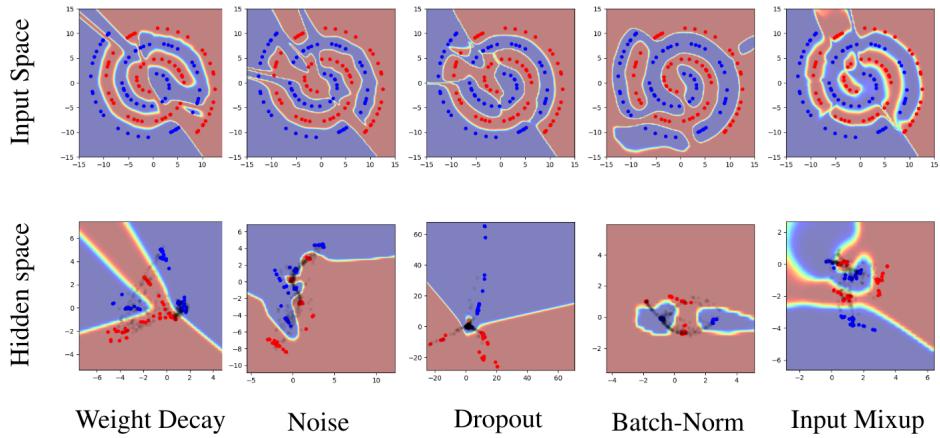


Рис. 10: Эксперимент с сетью, обученной на 2D наборе спиральных данных с использованием различных регуляризаторов. Эксперимент показывает, что эффект обеспечения широкой области низкой достоверности между областями классов не достигается другими регуляризаторами. Batch Normalization и Dropout на всех слоях, Dropout с вероятностью 0.5, под шумом подразумевается гауссовский шум.

Взято из [1].

часто резкая и близка к данным. Во-вторых подавляющая часть пространства признаков соответствует предсказаниям с высокой степенью достоверности, как внутри, так и вне многообразия данных. Руководствуясь этими интуициями, был придуман Manifold Mixup: простой регуляризатор, который устраняет некоторые из этих недостатков путем обучения нейронных сетей на линейных комбинациях скрытых представлений обучающих примеров.

Manifold Mixup улучшает обобщение глубоких нейронных сетей по следующим причинам:

1. Приводит к более плавным границам принятия решений, которые находятся дальше от обучающих данных, на нескольких уровнях пространства признаков. Гладкость и маржа являются общепризнанными факторами генерализации [28].
2. Использует интерполяцию в более глубоких скрытых слоях, которые собирают информацию более высокого уровня, для обеспечения дополнительного обучающего сигнала.
3. Сглаживает представления классов, значительно сокращая их количество направлений дисперсии (будет описано далее).

Метод Рассмотрим обучение глубокой нейронной сети $f(x) = f_k(g_k(x))$, где g_k обозначает часть нейронной сети, отображающую входные данные в скрытое представление на уровне k , а f_k обозначает часть, отображающую данное скрытое представление в выход $f(x)$.

Вначале мы выбираем случайный уровень k из набора допустимых уровней S в нейронной сети. Этот набор может включать в себя самый первый уровень $g_0(x)$. На практике k выбирается равновероятно из некоторых фиксированных уровней.

Далее считается скрытое представление образцов (x, y) и (x_0, y_0) до уровня k . После этого есть промежуточные образцы $(g_k(x), y)$ и $(g_k(x_0), y_0)$. На практике (x, y) и (x_0, y_0) берутся из одного батча. Подробности в Главе 4.

После слоя k происходит mixup [25]:

$$(\tilde{g}_k, \tilde{y}) = (\text{Mix}_\lambda(g_k(x), g_k(x_0)), \text{Mix}_\lambda(y, y_0)) \quad (20)$$

где $\text{Mix}_\lambda(a, b) = \lambda \cdot a + (1 - \lambda) \cdot b$. Здесь (y, y_0) - это целевые переменные, а коэффициент λ выбирается из распределения $\text{Beta}(\alpha, \alpha)$, как и в [25].

Наконец, данное скрытое представление отображается в выход обычным образом и считается функция ошибки.

Формально говоря, Manifold Mixup минимизирует:

$$L(f) = \mathbb{E}_{(x,y) \sim P} \mathbb{E}_{(x_0,y_0) \sim P} \mathbb{E}_{\lambda \sim \text{Beta}(\alpha, \alpha)} \mathbb{E}_{k \sim S} l(f_k(\text{Mix}_\lambda(g_k(x), g_k(x_0))), \text{Mix}_\lambda(y, y_0)). \quad (21)$$

где l - функция ошибки, P - распределение входных данных.

Теоретическое обоснование Предположим, что \mathcal{X} и \mathcal{H} обозначают пространства входных данных и признаков соответственно. Обозначим целевое множество \mathcal{Y} , а $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Пусть $\mathcal{G} \subseteq \mathcal{H}^{\mathcal{X}}$ обозначает множество функций, реализуемых нейронной сетью, от ввода к данному пространству признаков (до слоя k). Аналогично, пусть $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{H}}$ будет множеством всех функций, реализуемых нейронной сетью, от представления к ответу. Тогда решение задачи можно описать в следующем виде:

$$J(P) = \inf_{g \in \mathcal{G}, f \in \mathcal{F}} \mathbb{E}_{(x,y), (x_0,y_0), \lambda} l(f(\text{Mix}_\lambda(g(x), g(x_0))), \text{Mix}_\lambda(y, y_0)) \quad (22)$$

Точнее говоря, пусть P_D - эпиритическое распределение, $D = \{(x_i, y_i)\}_{i=1}^n$. Затем, пусть $f^* \in \mathcal{F}$ и $g^* \in \mathcal{G}$ будут минимизаторами $J(P)$ для $P = P_D$. Также пусть $\mathcal{G} = \mathcal{H}^{\mathcal{X}}$ и $\mathcal{F} = \mathcal{Y}^{\mathcal{H}}$, и \mathcal{H} является векторным пространством. Эти условия утверждают, что отображения, реализуемые большими нейронными сетями, плотны во множестве всех непрерывных ограниченных функций. В этом случае мы покажем, что минимизатор f^* является линейной функцией из \mathcal{H} в \mathcal{Y} . В таком случае, $J(P)$ может быть переписано следующим образом:

$$J(P_D) = \inf_{h_1, \dots, h_n \in \mathcal{H}} \frac{1}{n(n-1)} \sum_{i \neq j}^n \inf_{f \in \mathcal{F}} \int_0^1 l(f(\text{Mix}_\lambda(h_i, h_j)), \text{Mix}_\lambda(y_i, y_j)) p(\lambda) d\lambda \quad (23)$$

где $h_i = g(x_i)$.

Theorem 1. Пусть \mathcal{H} - векторное пространство размерности $\dim(\mathcal{H})$, и пусть $d \in \mathbb{N}$ обозначает количество классов, содержащихся в датасете D . Если $\dim(\mathcal{H}) \geq d - 1$, то $J(P_D) = 0$ и соответствующий минимизатор $f^* : \mathcal{H} \rightarrow \mathbb{R}^d$ является линейной функцией.

Доказательство. Сначала заметим, что если $\dim(\mathcal{H}) \geq d - 1$, то:

$$\exists A, H \in \mathbb{R}^{\dim(\mathcal{H}) \times d}, b \in \mathbb{R}^d : A^T H + b 1_d^T = I_{d \times d},$$

где $I_{d \times d}$ и 1_d обозначают d -мерную единичную матрицу и вектор из единиц соответственно. Фактически, $b 1_d^T$ является матрицей ранга один, тогда как ранг единичной матрицы равен d . Таким образом, для $A^T H$ достаточно только ранга $d - 1$.

Пусть $f^*(h) = A^T h + b$ для всех $h \in \mathcal{H}$. Пусть $g^*(x_i) = H_{\zeta_i,:}$, пусть ζ_i -м столбцом матрицы H , где $\zeta_i \in \{1, \dots, d\}$ обозначает индекс класса примера x_i . Эти выборы минимизируют (2), так как:

$$`((f^*(\text{Mix}_\lambda(g^*(x_i), g^*(x_j))), \text{Mix}_\lambda(y_i, y_j))) = `((A^T \text{Mix}_\lambda(H_{\zeta_i,:}, H_{\zeta_j,:}) + b, \text{Mix}_\lambda(y_i, \zeta_i, y_j, \zeta_j))) = `((u, u))$$

Результат следует из $A^T H_{\zeta_i,:} + b = y_i, \zeta_i$ для всех i . \square

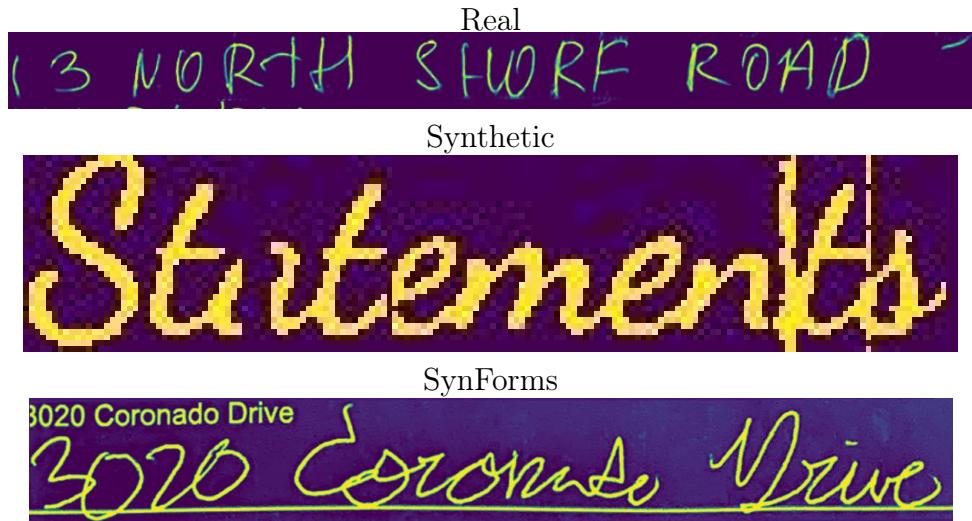


Рис. 11: Примеры изображений из датасета.

4 Конфигурация

4.1 Описание модели

В качестве энкодера использовался TransformerEncoder с следующими параметрами:

1. количество слоев: 6
2. nhead в multiheadattention): 4
3. размерность feedforward сети: 768
4. размерность пространства признаков: 128

В качестве сверточной нейронной сети для извлечения визуальных признаков использовался ResNet [17]. В качестве функции активации использовалась ReLU. Далее архитектура ResNet описывается чуть подробнее:

4.1.1 ConvBN

Слой ConvBN представляет из себя композицию свертки (conv) и батч-нормализации (BN). Параметры BN следующие:

1. eps: 10^{-12}
2. momentum: 0.01

За filters будем обозначать количество фильтров свертки. За kernel размер ядра, padding и stride также есть соответствующие параметры свертки.

4.1.2 ResNetBlock

Этот residual блок предствалляет из себя композицию трех ConvBN (с размерами ядра: [1, 3, 1], padding: [0, 1, 0]). Функция активации: ReLU (как было сказано ранее).

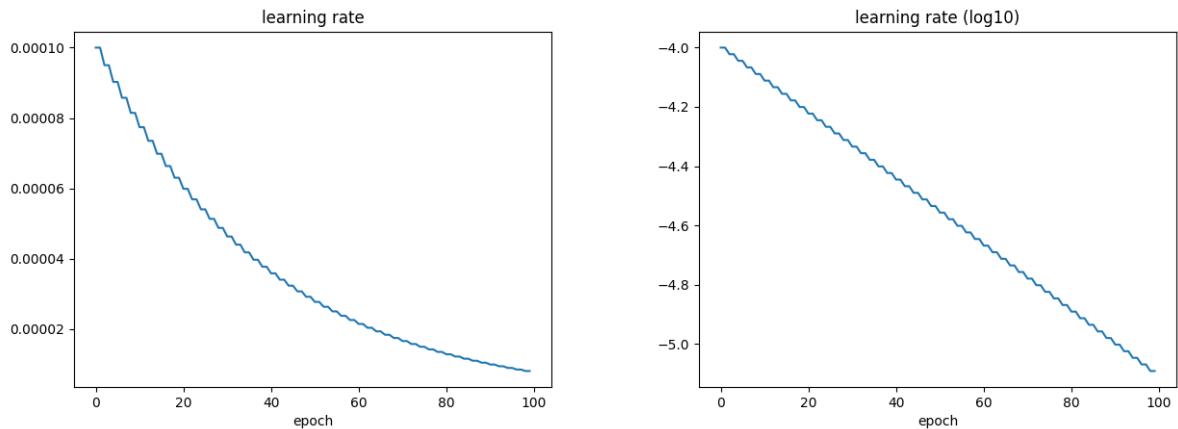


Рис. 12: Как менялся learning rate во время обучения.

4.1.3 FeatureBlock

Это композиция нескольких ResNetBlock. Также для всех FeatureBlock, кроме последнего, после ResNetBlock-ов есть еще ConvBN слой (kernel: 3, padding: 1). В некоторых случаях (кроме последних двух) данный блок завершается max-pool (kernel: 2). После FeatureBlock для регуляризации использовался dropout [12] (с вероятностью 0.1).

Первый FeatureBlock отличается от остальных: он представляет из себя композицию двух ConvBN (kernel: [3, 3], padding: [1, 1]) и max-pool (kernel: 2) слоя.

ResNet представляет из себя композицию нескольких FeatureBlock-ов (всего 5, с количествами ResNetBlock: [1, 2, 5, 3], первый FeatureBlock отличается от остальных). После FeatureBlock-ов идут два ConvBN и dropout (с вероятностью 0.2).

4.2 Описание датасета

Данные можно разделить на три категории:

1. Real: Данные, собранные с документов. Обычно это поля различных контрактов, форм, или поля на счетах
2. SynForms: Написанные людьми строки текста
3. Synthetic: Псевдорукописные шрифты после применения некоторых аугментаций

Примеры изображений можно найти на [Рис 11.]. Для обучения брались данные из всех трех групп в соотношении 1 : 1 : 1.

4.3 Оптимизатор

В качестве оптимизатора использовался AdamW [29] с параметрами:

1. betas: (0.9, 0.999)
2. weight decay: 0.01

Learning rate начинался с 0.0001 и умножался на 0.95 каждые две эпохи [Рис 12.]. Также во всех экспериментах размер батча был равен 64. Использовалась точность mixed precision в PyTorch. Также, если не сказано иного, обучение ведется в течение 100 эпох.

4.4 Реализация mixup

Для более эффективной работы, образцы (x, y) и (x_0, y_0) смешиваются каждый раз внутри одного батча. Точнее говоря, если мы имеем батч (x_1, \dots, x_N) , то операция mixup выглядит следующим образом:

$$\begin{aligned} y_{1:N} &= \text{shuffle}(x_{1:N}) \\ \tilde{x}_{1:N} &= \lambda x_{1:N} + (1 - \lambda)y_{1:N} \end{aligned} \tag{24}$$

Основываясь на результатах [16], по умолчанию (если не сказано иного) $\alpha = 0.5$, то есть $\lambda \sim Beta(0.5, 0.5)$. Каждый раз, во время обучения, mixup происходит лишь в одной позиции. Эта позиция выбирается случайно из набора.

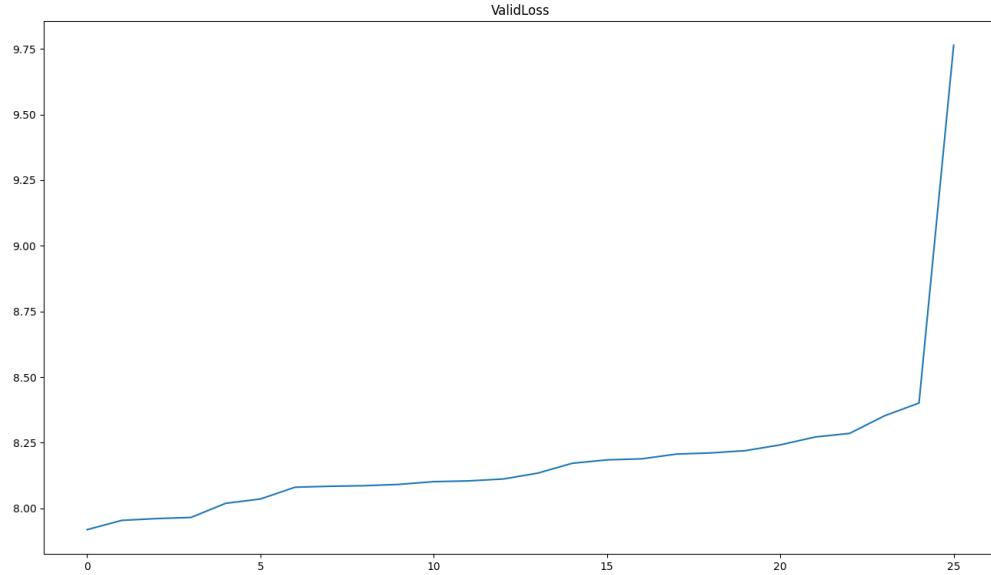


Рис. 13: Функция ошибки для всех рассматриваемых наборов позиций mixup.

5 Эксперименты

5.1 Влияние положения Mixup

Рассматривались положения mixup между FeatureBlock-ами, а также между последними двумя ConvBN. Точнее говоря, если описывать псевдокодом модель ResNet, рассматриваемые точки для mixup выглядят следующим образом:

```
for i in range(5):
    x = mixup(x)                      # before_block_i
    x = self.blocks[i](x)

    x = mixup(x)                      # after_blocks
    x = self.final_conv1(x)
    x = mixup(x)                      # after_final_conv1
    x = self.final_conv2(x)
    x = self.final_drop(x)
```

Основываясь на результатах [16], были проанализированы все варианты набора позиций для mixup, подходящие под следующие условия:

1. В каждом наборе присутствуют лишь 2 или 3 точки, где происходит mixup.
2. Позиции в наборе не идут последовательно друг за другом в порядке операций.

Каждый набор позиций был проверен на части датасета размером в $N = 128'000$ образцов [Таблица 2]. Как по всем метрикам, так и по функции ошибки, набор before-block0;before-block3 оказался наилучшим [Рис 13.] [Таблица 3].

type	ValidLoss	CharAcc	FragAcc	WordAcc
vanilla	5.55	87.56	56.28	59.93
mixup	4.85	88.05	56.87	60.93

Таблица 1: Функция ошибки и метрики при $N = 640'000$ образцах

5.2 Зависимость от количества данных

6 Результаты экспериментов

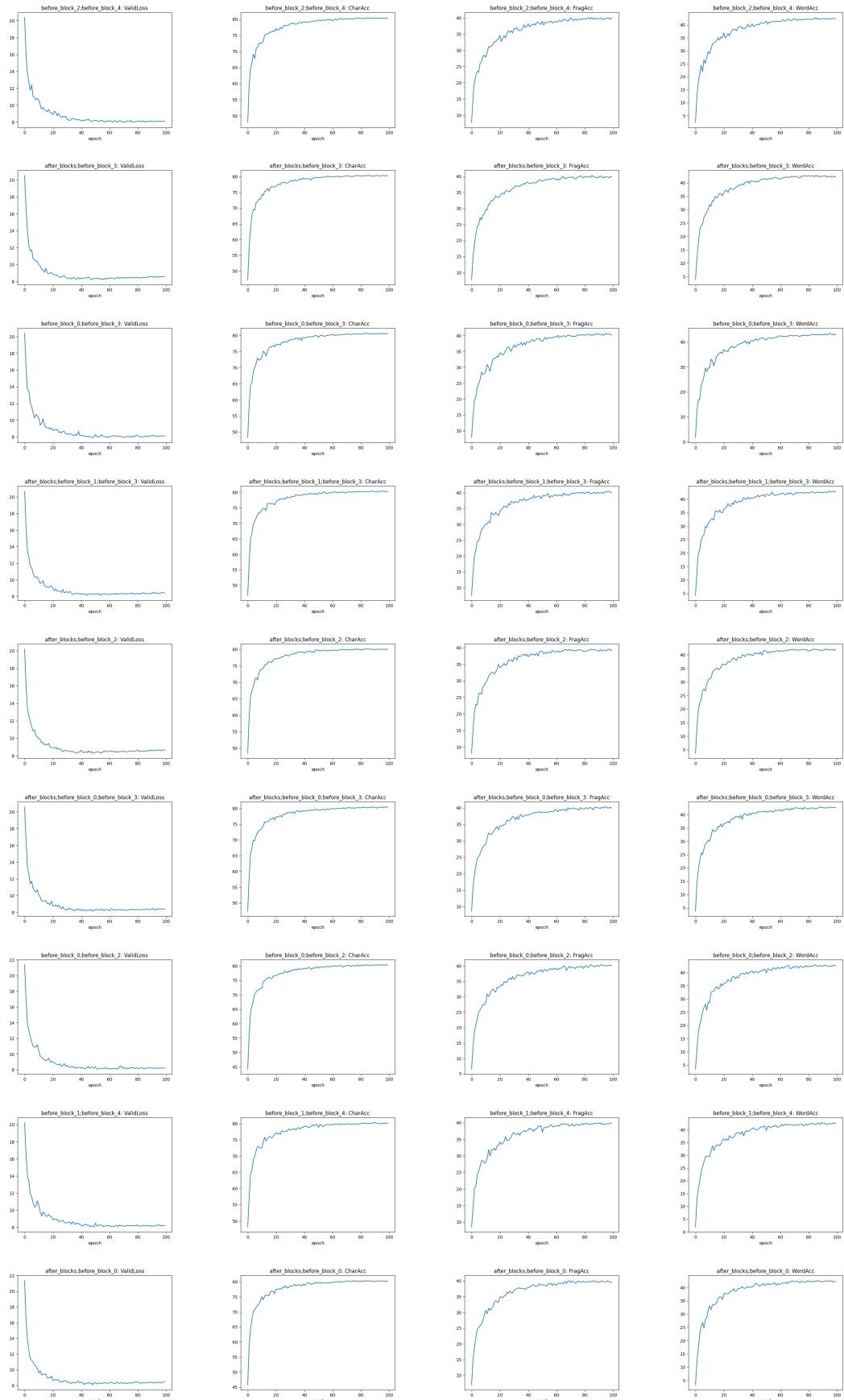
Здесь надо перечислить все результаты, полученные в ходе работы. Из текста должно быть понятно, в какой мере решена поставленная задача.

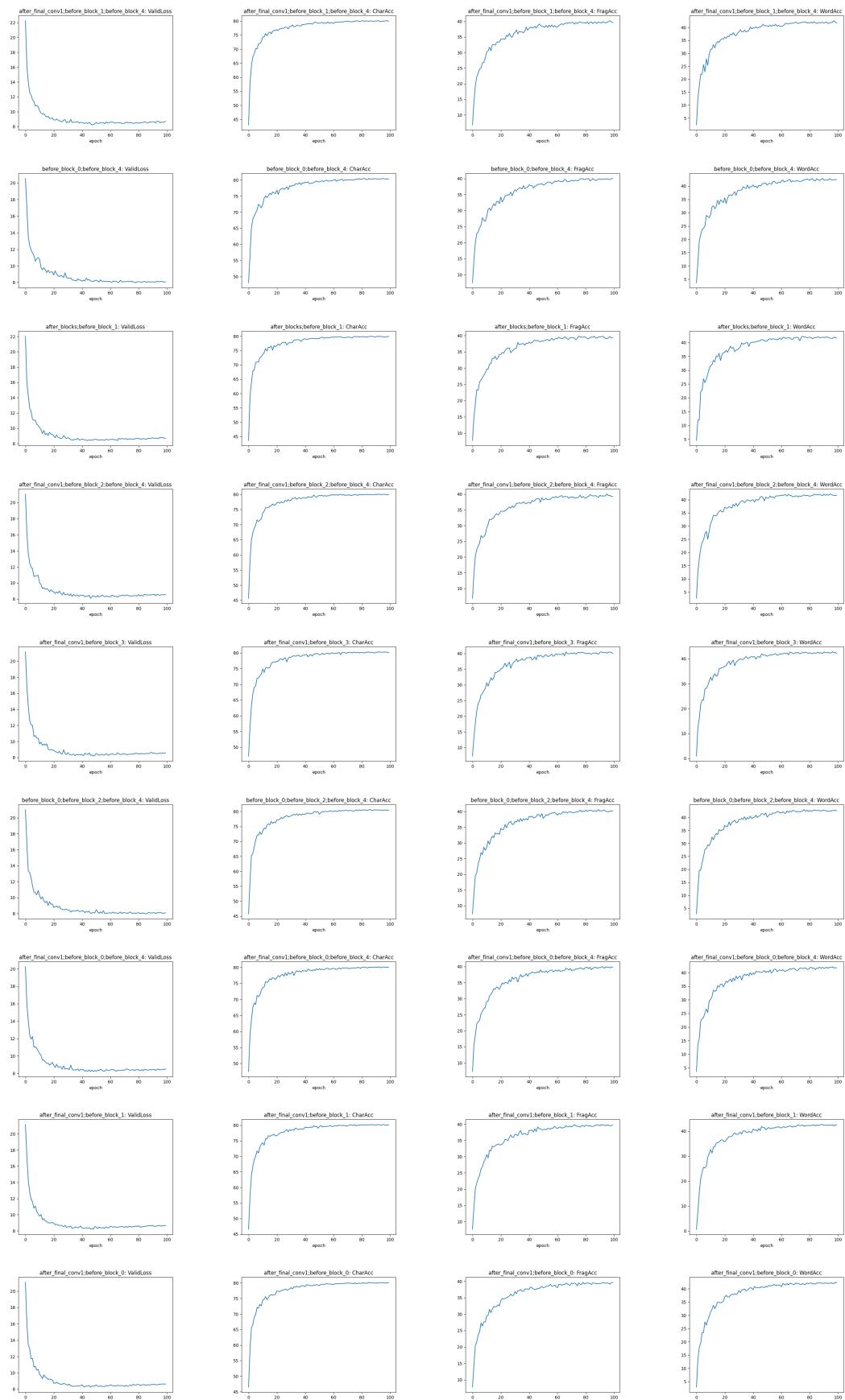
7 Вывод

Здесь надо перечислить все результаты, полученные в ходе работы. Из текста должно быть понятно, в какой мере решена поставленная задача.

Приложение

7.1 Позиция mixup





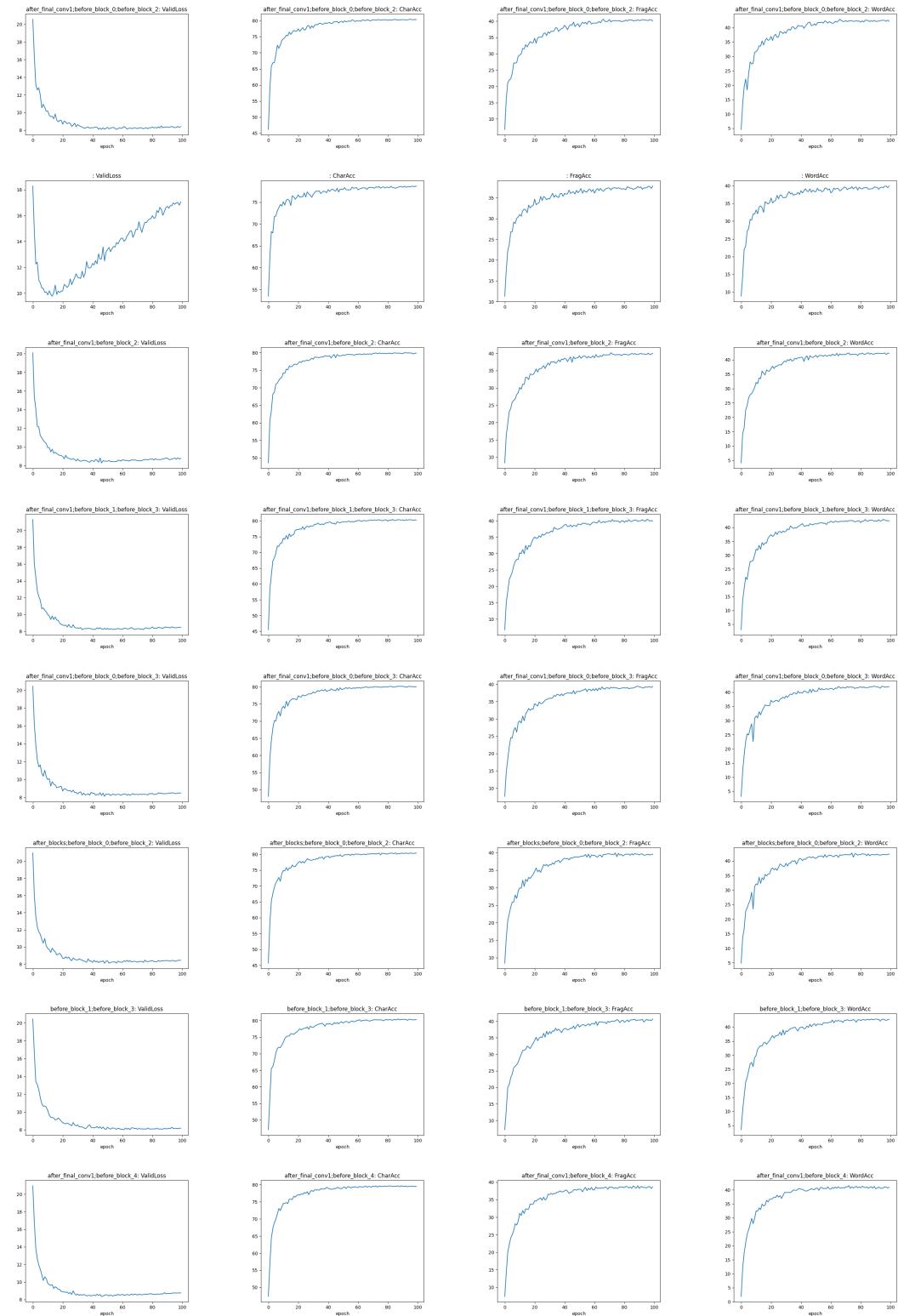


Таблица 2: Графики функции ошибки и некоторых метрик для каждого варианта позиций mixup.

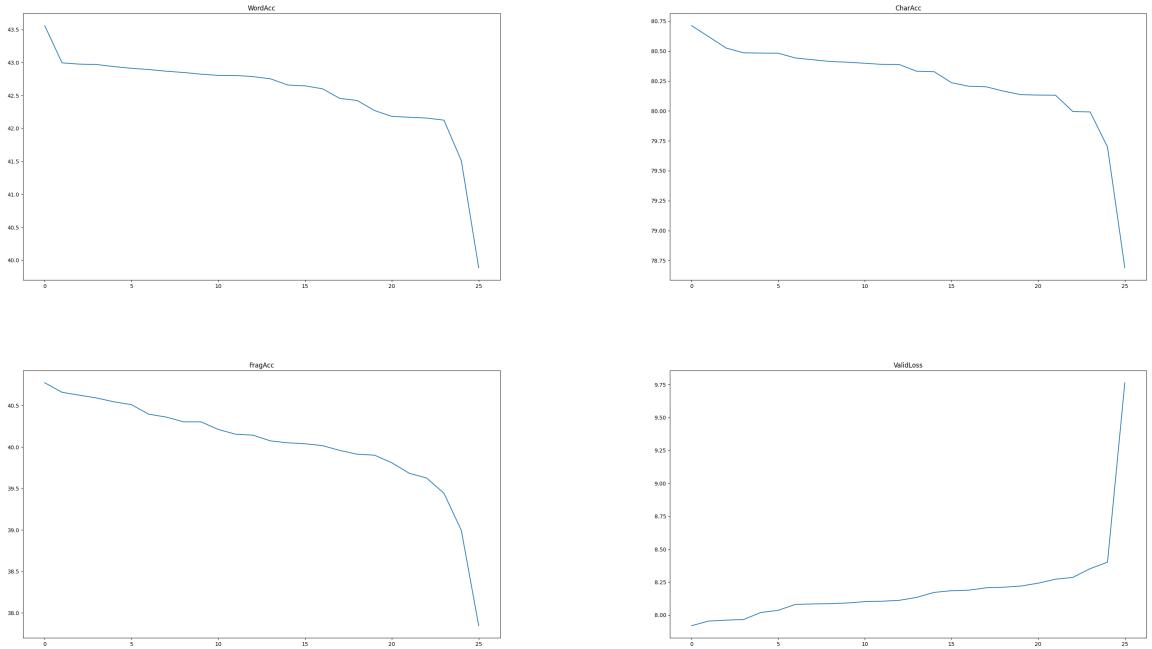
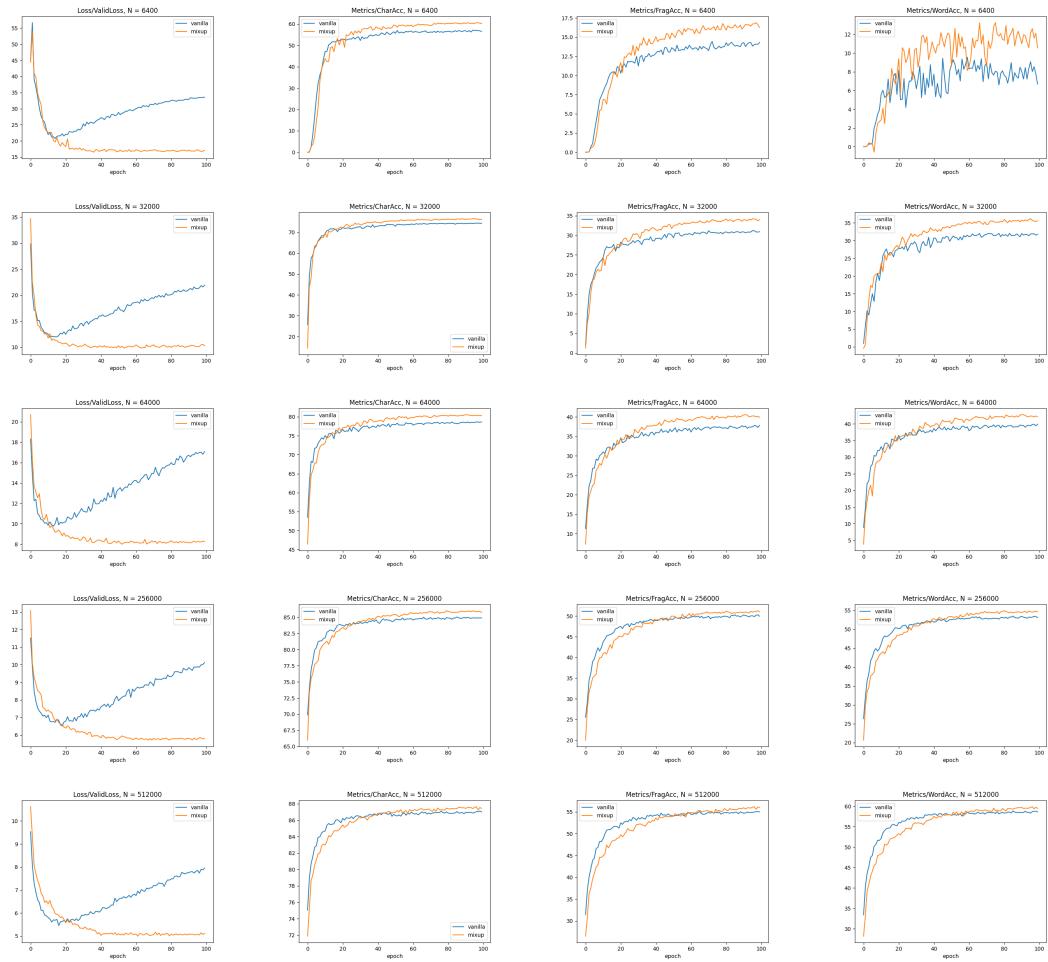


Таблица 3: Функция ошибки и различные метрики для всех рассматриваемых наборов позиций mixup.

7.2 Количество данных



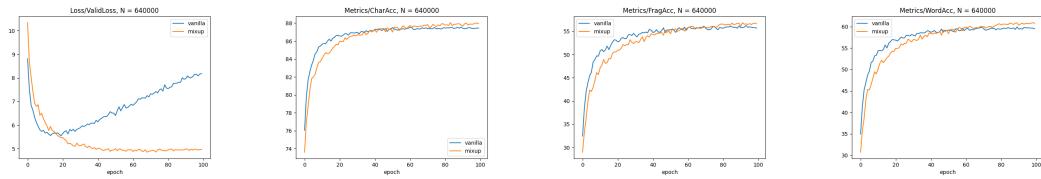


Таблица 4: Сравнение ванильной модели и mixup на различных размерах датасета.

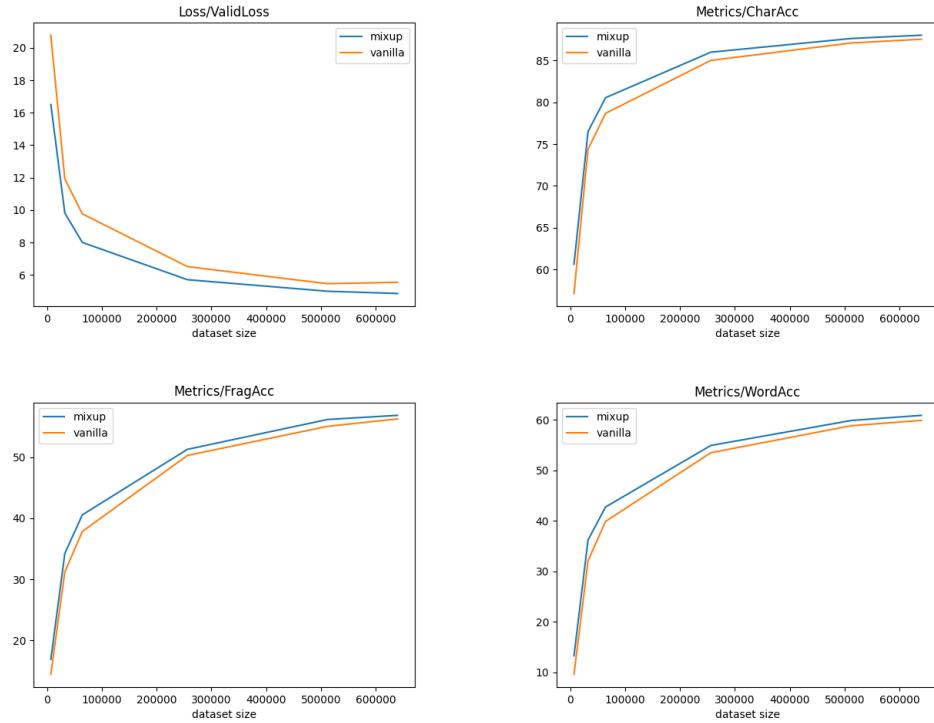


Таблица 5: Сравнение метрик для ванильной модели и mixup в зависимости от размера датасета.

Список литературы

- [1] Manifold Mixup: Better Representations by Interpolating Hidden States / Verma Vikas, Lamb Alex, Beckham Christopher et al. // *arXiv:1806.05236*. — 2018.
- [2] Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks / Graves Alex, Fern Santiago, Gomez Faustino, Schmidhuber Jürgen // International Conference on Machine Learning. — 2006.
- [3] Attention Is All You Need / Vaswani Ashish, Shazeer Noam, Parmar Niki et al. // *arXiv:1706.03762*. — 2017.
- [4] *Alex, Graves.* Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks / Graves Alex, Fern, Schmidhuber Jürgen // Conference on Neural Information Processing Systems. — 2008.
- [5] The A2iA Multi-lingual Text Recognition System at the Second Maurdor Evaluation / Bastien Moysset, Théodore Bluche, Maxime Knibbe et al. // 14th International Conference on Frontiers in Handwriting Recognition. — 2014.

- [6] *Puigcerver, Joan.* Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? / Joan Puigcerver // International Conference on Document Analysis and Recognition. — 2017.
- [7] *Bluche, Theodore.* Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition / Theodore Bluche, Ronaldo Messina // International Conference on Document Analysis and Recognition. — 2017.
- [8] *Fujitake, Masato.* DTrOCR: Decoder-only Transformer for Optical Character Recognition / Masato Fujitake // *arXiv:2308.15996*. — 2023.
- [9] *Helmers, Muriel.* Generation and Use of Synthetic Training Data in Cursive Handwriting Recognition / Muriel Helmers, Horst Bunke // Iberian Conference on Pattern Recognition and Image Analysis. — 2003.
- [10] *Shen, Xi.* A Method of Synthesizing Handwritten Chinese Images for Data Augmentation / Xi Shen, Ronaldo Messina // 15th International Conference on Frontiers in Handwriting Recognition (ICFHR). — 2016.
- [11] *Alonso, Eloi.* Adversarial Generation of Handwritten Text Images Conditioned on Sequences / Eloi Alonso, Bastien Moysset, Ronaldo Messina // *arXiv:1903.00277*. — 2019.
- [12] Dropout: A Simple Way to Prevent Neural Networks from Overfitting / Sutskever Ilya, Krizhevsky Alex, Hinton Geoffrey et al. // *Journal of Machine Learning Research 15*. — 2014.
- [13] Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network / Curtis Wigington, Seth Stewart, Brian Davis et al. // 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). — 2017.
- [14] SMOTE: Synthetic Minority Over-sampling Technique / Chawla V. N., Bowyer W. K., Hall O. L., Kegelmeyer P. W. // *arXiv:1106.1813*. — 2011.
- [15] *Terrance, DeVries.* Dataset Augmentation in Feature Space / DeVries Terrance, Taylor W. Graham // *arXiv:1702.05538*. — 2017.
- [16] *Moysset, Bastien.* Manifold Mixup improves text recognition with CTC loss / Bastien Moysset, Ronaldo Messina // *arXiv:1903.04246*. — 2019.
- [17] Deep Residual Learning for Image Recognition / He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian // *arXiv:1512.03385*. — 2015.
- [18] Going Deeper with Convolutions / Szegedy Christian, Liu Wei, Jia Yangqing et al. // *arXiv:1409.4842*. — 2014.
- [19] MobileNetV2: Inverted Residuals and Linear Bottlenecks / Sandler Mark, Howard Andrew, Zhu Menglong et al. // *arXiv:1801.04381*. — 2018.
- [20] Rethinking Text Line Recognition Models / Diaz Hernandez Daniel, Qin Siyang, Ingle Reeve et al. // *arXiv:2104.07787*. — 2021.
- [21] *Sergey, Ioffe.* Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift / Ioffe Sergey, Szegedy Christian // *arXiv:1502.03167*. — 2015.

- [22] Gated recurrent convolution neural network for ocr. / In I. Guyon, U. V. Luxburg, S. Bengio et al. // Advances in Neural Information Processing Systems. — Curran Associates, 2017.
- [23] Peter, Shaw. Self-Attention with Relative Position Representations / Shaw Peter, Uszkoreit Jakob, Vaswani Ashish // arXiv:1803.02155. — 2018.
- [24] Sequence-to-Label Script Identification for Multilingual OCR / Fujii Yasuhisa, Driesen Karel, Baccash Jonathan et al. // arXiv:1708.04671. — 2017.
- [25] mixup: Beyond Empirical Risk Minimization / Zhang Hongyi, Cisse Moustapha, Dauphin N. Yann, Lopez-Paz David // arXiv:1710.09412. — 2017.
- [26] CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features / Yun Sangdoo, Han Dongyoong, Oh Joon Seong et al. // arXiv:1905.04899. — 2019.
- [27] StackMix and Blot Augmentations for Handwritten Text Recognition / Shonenkov Alex, Karachev Denis, Novopoltsev Maxim et al. // arXiv:2108.11667. — 2021.
- [28] Bartlett, Peter. Generalization Performance of Support Vector Machines and Other Pattern Classifiers / Peter Bartlett, John Shawe-Taylor // Advances in Kernel Methods: Support Vector Learning. — 1998.
- [29] Ilya, Loshchilov. Decoupled Weight Decay Regularization / Loshchilov Ilya, Hutter Frank // arXiv:1711.05101. — 2017.