Плешкова Алина Артуровна

Сопоставление текстов и изображений с использованием только трансформер-подобной архитектуры

Выпускная квалификационная работа

Санкт-Петербург

2021

Оглавление

Введение

С развитием технологий людям все чаще приходится работать с мультимедийным контентом – это информация, представление которой сочетает в себе различные способы. Примером может быть кино, которое сочетает в себе аудио и изображение на экране, или пост в социальных сетях, который обычно содержит текст и изображение, подходящее к тексту. С появлением мультимедийных данных появляется проблема связи информации разных типов. Одной из таких проблем является сопоставление текста и изображения. Решение данной проблемы очень важно в текстово-визуальном поиске, то есть поиске картинки по тексту или наоборот — текста по картинке.

На данный момент существует большое количество работ, которые разными способами решают данную задачу.

Авторы [6], [9], [24] переводят изображение и текст в пространство $\mathbb{R}^d$. Полученные вектора должны находиться близко, если текст и изображение несут одну и туже информацию. Недостаток данного подхода состоит в том, что для проецирования изображения и текста используется две различные сети, которые принимают на вход либо изображение, либо текст. Сети не видят изображение и текст вместе, поэтому не знают, на что нужно обратить внимание на изображении по отношению к тексту и наоборот — на тексте по отношению к изображению.

В статье [21] используется механизм внимания (attention) для реше-

ние вышеуказанной проблемы. В статьях [14], [23] пошли дальше и решили использовать трансформер. В данных работах делается предобработка изображения — выделяются определенные области изображения, которые несут важную информацию. Такая предобработка является минусом, так как сеть не работает со всем изображением, то есть получает не всю информацию о нем.

Модели на основе трансформера такие как BERT [2], GPT-3 [17] хорошо показали себя в NLP задачах. В недавних статьях [11], [13] авторы используют трансформер для классификации изображения и показывают очень хорошие результаты. Отличительной чертой от предыдущих работ является то, что изображения никак не предобрабатываются.

Авторы статьи ViLT [16] применили модель ViT [13] для задачи сопоставления текста и изображения. Но данная модель обучается на большом количестве данных и с большим размером батча. В небольших компаниях или университетах часто не хватает ресурсов, чтобы обучить такую модель.

В данной работе представлена модель для сопоставления текста и изображения. За основу была взята модель ViT [13].

Цель работы – построение трансформер-подобной модели для задачи сопоставления текста и изображения, способной обучаться при условии ограниченности ресурсов.

Задачи:

- адаптировать ViLT для обучения на маленьком датасете с неболь-

шим размером батча;

- сконструировать архитектуру для дистилляции модели;
- сравнить результат адаптированного ViLT и результат модели после дистилляции.

Работа состоит из 3 глав.

В первой главе приведены использующиеся в работе термины и описаны существующие работы по данной проблеме.

Во второй главе представленна адаптированная модель ViLT, архитектура для дистилляции модели и описан сценарий обучения.

В третьей главе представлены проведенные эксперименты и полученные результаты.

В заключении сделаны выводы о проделанной работе и предлагаются возможные направления дальнейших исследований.

Глава 1

Обзор литературы

В данной главе приведен обзор существующих моделей, которые решают проблему сопоставления текста и изображения. Также приведены базовые определения и архитектуры.

1.1. Базовые понятия машинного обучения

1.1.1. Векторизация

Чтобы сложные объекты можно было обрабатывать нейронной сетью, необходимо представить их в виде вектора.

Определение 1. Эмбеддинг – обучаемое векторное представление объекта.

Эмбеддинги можно разделить на 3 вида: эмбеддинг объекта (текста, картинки), позиционный эмбеддинг, эмбеддинг типа.

Пример 1. Есть некоторое предложение. Для каждого слова создается вектор, который будет эмбеддингом объекта. Позицию каждого слова можно представить в виде позиционного эмбеддинга. Если предложений два, то для слов из первого предложения будет соответствовать эмбеддинг типа 1, а для слов из второго предложения – эмбеддинг типа 2.

1.1.2. Данные

В ходе выполнения работы использовались данные состоящие из пар (текст, изображение).

Определение 2. Данные будем называть парными, если в паре (текст, изображение) текст описывает данное изображение. Также будем говорить, что текст и изображение несут одинаковую информацию.

1.2. NLP модели

BERT

Определение 3. BERT (Bidirectional Encoder Representations from Transformers) – нейронная сеть, основанная на энкодере трансформера для решения задач обработки естественного языка. Архитектура модели представлена на рисунке 1.1.

Архитектура модели BERT представляет из себя последовательность блоков. На вход первому блоку подается последовательность эмбеддингов, на выходе получаем последовательность векторов той же длины. Пусть z^0— входящая последовательность эмбеддингов. Уравнения 1.1–1.2 описывают архитектуру.

$$\hat{z}^l = \mathbf{LN}(\mathbf{MSA}(z^{l-1}) + z^{l-1}), \qquad l = 1 \cdots L, \tag{1.1}$$

$$z^l = \mathbf{LN}(\mathbf{MLP}(\hat{z}^l) + \hat{z}^l), \qquad l = 1 \cdots L, \tag{1.2}$$

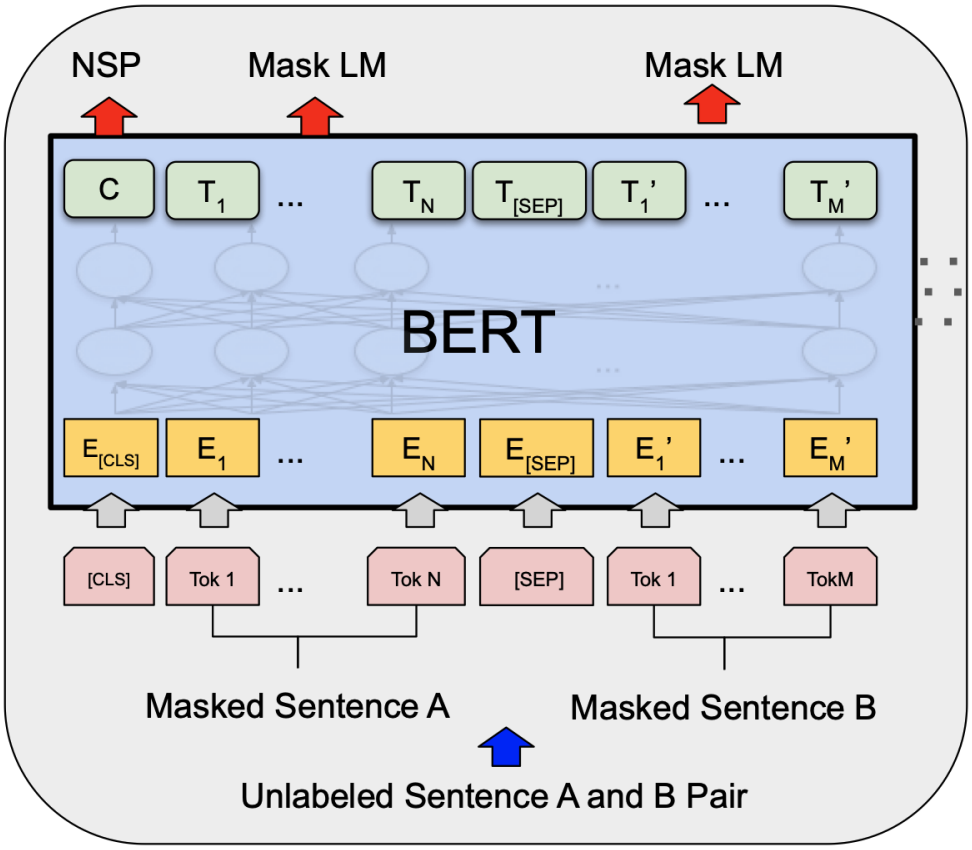

Рис. 1.1. BERT.

где **MSA** (multi-head self-attention) — множественное внимание, **MLP** (multilayer perceptron) — многослойный перцептрон, **LN** — layer normalization.

Модель была обучена на задаче предсказания замаскированных токенов (masked language modeling (MLM)) и на задаче предсказания следующего предложения (next sentence prediction (NSP)). На вход модели подавались вектора, каждый из которых это сумма эмбеддинга слова, позиционного эмбеддинга и эмбеддинга типа (рис.1.2).

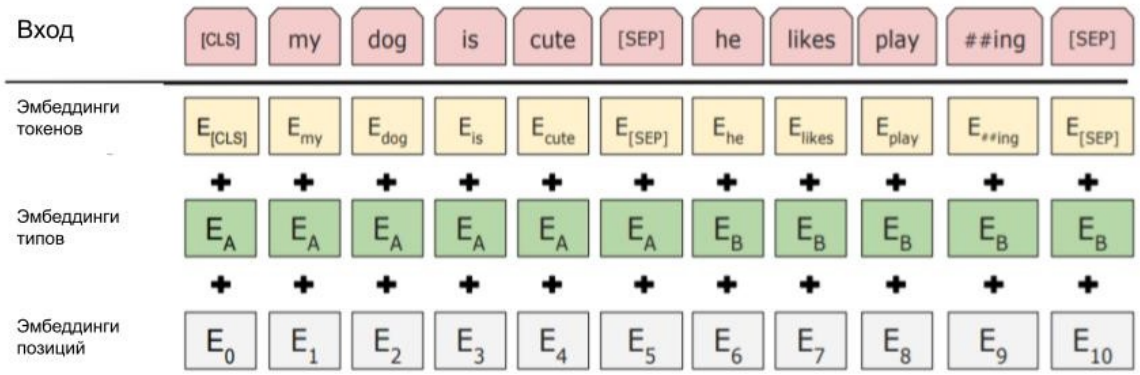

Рис. 1.2. Эмбеддинги.

Для обучения первой задаче каждый токен в предложение маскировался с вероятностью 0.15. При этом замаскированные токены:

- с вероятностью 0.8 заменяются на токен [MASK];
- с вероятностью 0.1 заменяются на случайный токен;
- с вероятностью 0.1 токен остается прежним.

Модель училась определять, какой токен должен стоять на месте замаскированного.

Дополнительно модель обучалась предсказывать, является ли второе предложение логическим продолжением первого. Таким образом, модель училась понимать, как соотносятся предложения друг с другом. Для решения этой задачи на вход подавалось два предложения, разделенные токеном [SEP]. Эмбеддинги типа так же нужны для того, чтобы модель могла понять, где слова первого предложения, а где второго. Так-

же к началу первого предложения конкатинируется токен [CLS], который будет служить для классификации предложений (рис.1.2).

Выходами сети являются вектора такого же размера, как входные эмбеддинги. Чтобы предсказать, является ли второе предложение продолжением первого, выход соответствующий токену [CLS] проходит через линейные слои, которые так же обучаются. Аналогично для задачи предсказания замаскированных токенов – выход замаскированного токена подается в линейные слои. В конечном итоге мы получаем вектор размера словаря, состоящих из логитов. Пропустив данный вектор через Softmax получаем, с какой вероятностью каждый токен может стоять на данном месте.

Модель BERT хорошо показывает себя в разных задачах обработки естественного языка. Библиотека Hugging Face [22] предоставляет реализацию данной модели вместе с обученными весами.

1.3. Первые подходы к задаче кросс-модального поиска

Данная работа больше сфокусирована на трансформер-подобной архитектурах в задаче кросс-модального поиска. Ниже приведено краткое описание более ранних подходов.

Одним из первых подходов является Canonical Correlation Analysis (CCA) [12], в котором максимизируют корреляцию между двумя линейными проекциями текстов и изображений. Kernel CCA (KCCA) [12] –

расширение предыдущего метода, где используется нелинейные проекции на некоторое пространство. Однако оба метода плохо масштабируются на большие объемы данных. В [4], [26] данный подход улучшили с помощью применения глубоких нейронных сетей.

1.4. Подходы с семантическим пространством

Как правило, решения, основанные на данном подходе, используют модели, проецирующие текст и изображение в вектора. Полученные эмбеддинги принадлежат одному пространству, размерность которого чаще всего намного меньше, чем размерность исходного пространства объекта. Данное пространство называется семантическим.

Семантическое пространство обеспечивает общее представление разных объектов (текста и картинки) и позволяет их сравнивать. Идея метода состоит в том, что текст и картинка, передающие одинаковую информацию, в семантическом пространстве будут находиться рядом. Таким образом, схожесть объектов определяется некоторой метрикой расстояния их эмбеддингов. В качестве расстояния можно взять евклидово или косинусное. Во время обучения модель старается отдалить друг от друга непарные объекты и приблизить парные.

На рисунке 1.3 представленна стандартная архитектура данного метода. Такая модель используется в работах [9], [6], [24], [25]. Изображение *image* и текст *text* передаются в модели *image model* и *text model*

соответственно. На выходе мы получаем эмбеддинг изображения ***image embedding*** и эмбеддинг текста ***text embedding***, которые отправляются в функцию S, вычисляющую близость двух векторов.

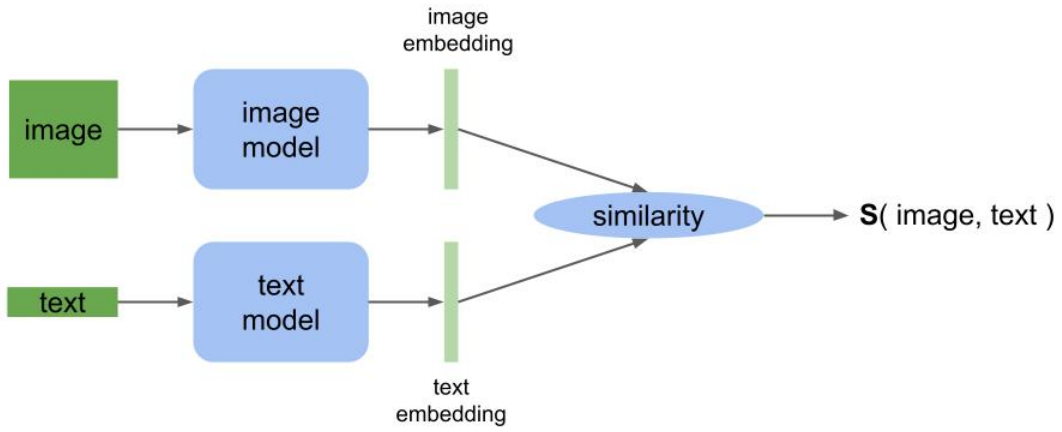

Рис. 1.3. Архитектура моделей, базирующихся на семантическом пространстве.

Положительной чертой данного метода является то, что можно заранее вычислить эмбеддинги объектов. Так как обе модели, по сути, работают только с одним типом объектов, то не нужно для каждой пары (текст, картинка) заново пересчитывать эмбеддинги. При получении запроса считается его эмбеддинг, далее из предварительно построенного множества эмбеддингов ищутся ближайшие. Это может сильно сократить время поиска.

Данный плюс является недостатком данного подхода. Проблема за-

ключается в том, что каждая модель видит либо картинку, либо текст. Поэтому модель не может сделать выводы по картинке относительно текста и наоборот – по тексту относительно картинки. Модель просто не знает, на что нужно обратить внимание в каждом объекте.

1.5. Подходы использующие механизм внимания

Механизмы внимания позволяют моделям сосредоточиться на необходимых частях визуального и текстового входа, таким образом решая проблемы предыдущего подхода.

Примером такого подхода может послужить модель SCAN [21].

Первое на что хочется обратить внимание в этой модели — предобработка изображения. Авторы следуют следующей идее: в описании изображения часто встречаются слова обозначающие конкретные объекты, а также их свойства и действия. Другими словами, каждое слово в предложении соответствует некоторой области на изображении. Следуя этому утверждению, авторы выделяют конкретные области на картинке с помощью Faster R-CNN [8]. Таким образом, каждой картинке соответствует некоторое количество фрагментов изображения.

После предобработки данные состоят из двух последовательностей: первая — последовательность фрагментов изображения, вторая — последовательность слов в предложении.

Используя сверточные нейронные сети, авторы получают эмбеддин-

ги выделенных регионов изображения $\{v_1, \ldots, v_k\}$ и эмбеддинги слов предложения $\{e_1, \ldots, e_n\}$ с помощью двунаправленной сети GRU [1], [20].

Далее строится матрица сходства:

$$s_{ij} = \frac{v_i^T e_j}{\|v_i\| \|e_j\|}, \; i \in [1,k], \; j \in [1,n]. \tag{1.3}$$

В данном случае s_{ij} показывает схожесть i-го региона и j-го слова.

Дальнейшие действия выполняются относительно региона изображения.

Авторы устанавливают порог и нормализуют матрицу сходства:

$$\bar{s}_{ij} = [s_{ij}]_+ / \sqrt{\sum_{i=1}^{k} [s_{ij}]_+^2}, \tag{1.4}$$

где $[x]_+ \equiv max(0,x)$.

Далее считаются веса α_{ij}, которые показывают, насколько важно слово j относительно региона изображения i. Строится вектор внимания предложения a_i^t относительно региона i.

$$a_i^t = \sum_{j=1}^{n} \alpha_{ij} e_j, \tag{1.5}$$

где

$$\alpha_{ij} = \frac{\exp(\lambda_1 \bar{s}_{ij})}{\sum\limits_{j=1}^{n} \exp(\lambda_1 \bar{s}_{ij})} \tag{1.6}$$

и λ_1 – это температура в функции Softmax.

Чтобы определить важность фрагмента изображения относительно

предложения, считается следующее значение:

$$R(v_i, a_i^t) = \frac{v_i^T a_i^t}{\|v_i\| \|a_i^t\|}. \tag{1.7}$$

Тогда величина сходства изображения I и текста T вычисляется, как

$$S(I,T) = \log\Big(\sum_{i=1}^{k} \exp(\lambda_2 R(v_i, a_i^t))\Big)^{(1/\lambda_2)}, \tag{1.8}$$

где λ_2 определяет, насколько важными будут наиболее релевантные пары фрагмента изображения v_i и вектора внимания a_i^t.

Заметим, что все уравнения вычислялись относительно некоторого фрагмента изображения i. Далее авторы предлагают вычислить те же значения 1.4–1.8 уже относительно слова j в предложении. Таким образом, получаем вторую величину сходства изображения I и текста T

$$S'(I,T) = \log\Big(\sum_{i=1}^{k} \exp(\lambda_2 R'(e_j, a_j^v))\Big)^{(1/\lambda_2)}, \tag{1.9}$$

где $R'(e_j, a_j^v)$ вычисленно относительно слова j. Итоговая $S(I,T)$ берется, как комбинация 1.8 и 1.9.

Для обучения используется Triplet Loss:

$$l(I,T) = \sum_{\hat{T}} \Big[\alpha - S(I,T) + S(I,\hat{T})\Big]_+ + \sum_{\hat{I}} \Big[\alpha - S(I,T) + S(\hat{I},T)\Big]_+,$$

где $[x]_+ \equiv max(0,x)$, $(I,\ T)$ – парные объекты, $\hat{I}$ – непарное изображение к тексту T и $\hat{T}$ – непарный текст к изображению I.

Данная функция потерь добивается того, чтобы для любого объекта парный для него объект находился как минимум на величину α ближе,

чем любой непарный. Заметим, что если непарный объект находится на α дальше парного, то это слагаемое будет равняться нулю.

Плюсом данного подхода является то, что модель знает, на что нужно обращать внимание на картинке относительно текста и наоборот – на тексте относительно изображения.

1.6. Подходы на основе трансформеров

Наилучшие результаты среди моделей с механизмом внимания показывают модели на основе трансформера. Авторы статей [14], [23] используют BERT для сопоставления текста и изображения. На рисунке 1.4 представлена архитектура ImageBERT [14].

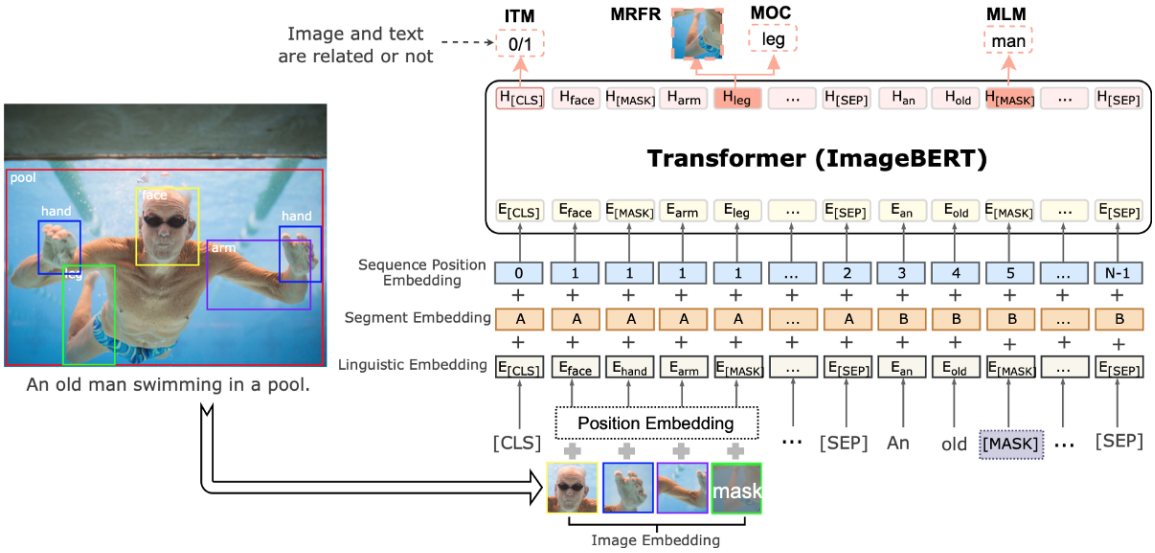

Рис. 1.4. ImageBERT.

Аналогично предыдущему подходу осуществляется предобработка

изображения. Выполняется детекция и выделяются некоторые фрагменты изображения. Полученные фрагменты проецируются в вектор, который является их эмбеддингом. Далее в BERT отправляются эмбеддинги, каждый из которых является суммой трех:

- эмбеддинг фрагмента изображения или эмбеддинг слова в предложение;
- позиционный эмбеддинг;
- эмбеддинг типа.

Позиционные эмбеддинги для любого фрагмента изображения одинаковые. Эмбеддинг типа помогает сети разделять, какие входы относятся к картинке, а какие к тексту. Также между картинкой и текстом стоит токен [SEP], который разделяет два объекта. Токен [CLS] отвечает за классификацию — схожи ли картинка и текст.

Во время обучения модель учится решать 3 задачи:

- предсказание замаскированных токенов в предложение;
- предсказание замаскированных эмбеддингов фрагментов картинки, либо классификация фрагмента изображения;
- классификация текста и картинки — парные данные или нет.

Рассмотренные подходы с использованием механизма внимания, а также на основе трансформеров выделяют некоторые важные фрагмен-

ты изображения. Но важность каждого фрагмента определяется не самой моделью, что является недостатком. Модель не видит все изображение целиком, а только те фрагменты, которые оказались задетектированы.

1.7. Подходы без предобработки изображения для классификации картинок

В недавних статьях [13], [11] представлены модели для классификации изображений на основе трансформеров. Отличительной чертой данных подходов является то, что в них нет предобработки изображения.

В статье [11] описаны две архитектуры на основе BERT и GPT-2, которые на вход принимают изображение вытянутое в вектор. Я не буду описывать подробно две эти модели, так как я не беру их за основу в своей работе.

Остановимся поподробнее на модели ViT [13](рис. 1.5).

Модель ViT обучалась на задаче классификации картинок. Архитектура представляет из себя энкодер трансформера, аналогично модели BERT. Изображение $x \in \mathbb{R}^{H \times W \times C}$ предварительно делится на равные части. Таким образом получают N фрагментов $x_p \in \mathbb{R}^{P \times P \times C}$, где H, W – изначальный размер изображения, C — число каналов, P — размер фрагмента, $N = HW/P^2$ — число фрагментов. Далее каждый фрагмент вытягивается в вектор $x_p \in \mathbb{R}^{1 \times P^2 \cdot C}$ и проецируется в вектор размера D.

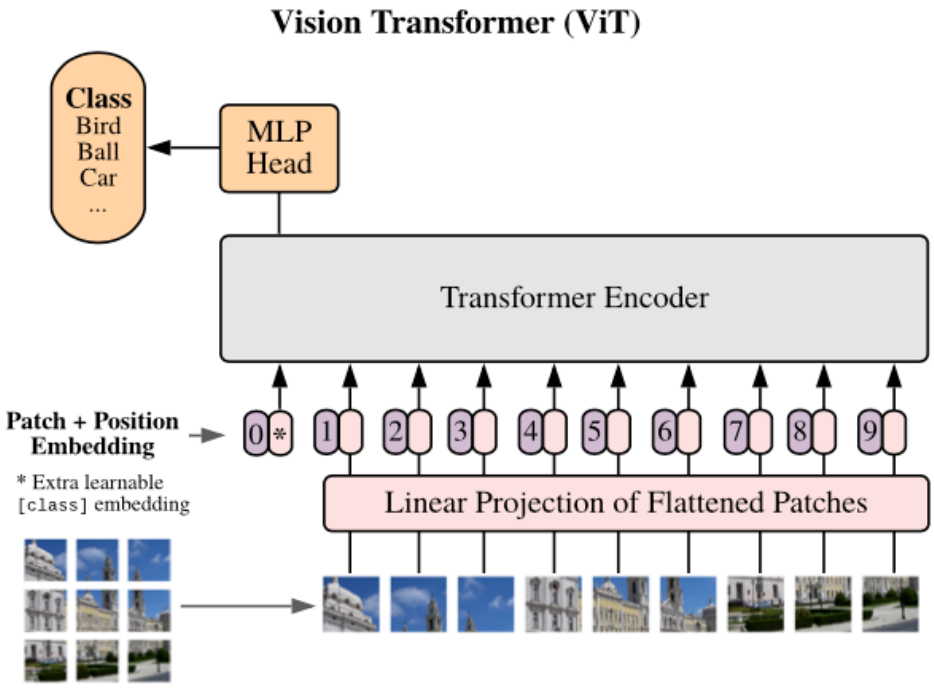

Рис. 1.5. ViT.

Данные вектора назовем эмбеддингами фрагментов изображения. Аналогично BERT к ним конкатенируется вектор классификации x_{cls} и прибавляются позиционные эмбеддинги. Таким образом, входящая последовательность принимает следующий вид:

$$z^0 = [x_{cls}, x_p^1\mathbf{E}, \ldots, x_p^N\mathbf{E}] + \mathbf{E}_{pos},\ \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D},$$

где $\mathbf{E}$ — матрица, которая преобразует фрагмент в вектор, а $\mathbf{E}_{pos}$ — позиционные эмбеддинги.

На вход трансформеру подается полученная последовательность век-

торов z^0.

$$\hat{z}^l = \mathbf{MSA}(\mathbf{LN}(z^{l-1})) + z^{l-1}, \qquad l = 1 \cdots L, \tag{1.10}$$

$$z^l = \mathbf{MLP}(\mathbf{LN}(\hat{z}^l)) + \hat{z}^l, \qquad l = 1 \cdots L, \tag{1.11}$$

$$y = \mathbf{LN}(z_0^L), \tag{1.12}$$

где **MSA** (multi-head self-attention) — множественное внимание, **MLP** (multilayer perceptron) — многослойный перцептрон, **LN** — layer normalization.

Выход токена классификации y подается в линейный слой. На выходе получам логиты, которые используются для классификации изображения.

Уравнения 1.10–1.11 показывают, как выглядит блок ViT. Заметим, что данный блок отличается от блока BERT. На рисунке 1.6 представлены блоки двух этих моделей, на котором красными стрелками и знаком плюс показано сложение выходов (residual block). Положение слоя нормализации (LN) в ViT является единственным отличием от BERT: LN идет после MSA и MLP в BERT и перед в ViT.

Данная модель была предобучена на датасете ImageNet-21k [15]. Авторы данной статьи предоставляют предобученные веса модели.

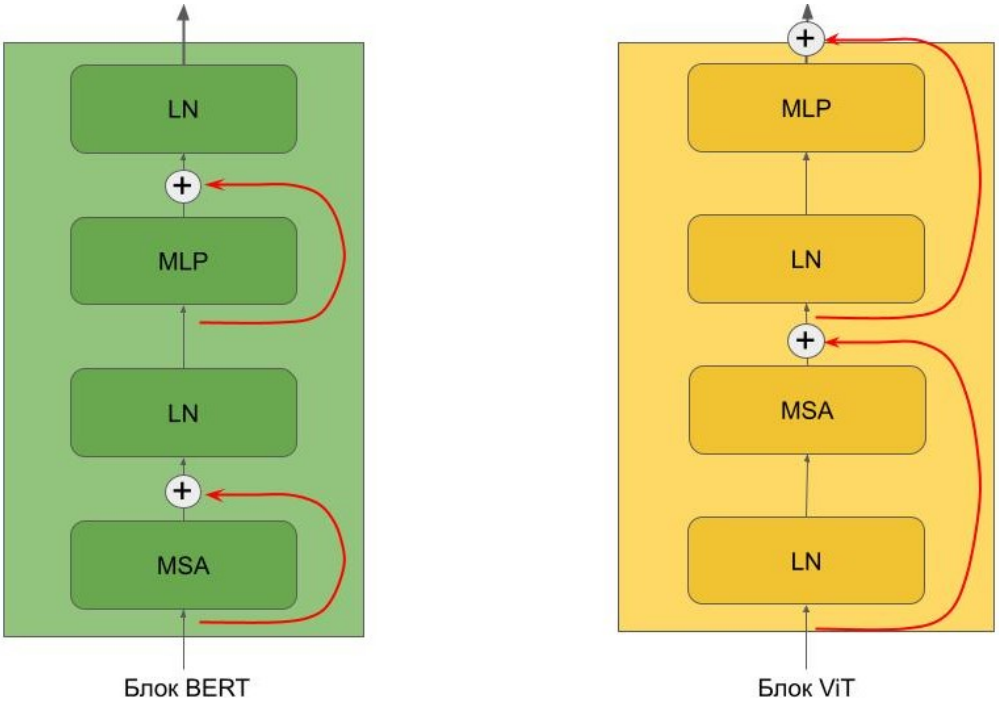

Рис. 1.6. Отличие блоков BERT и ViT.

1.8. Подходы без предобработки изображения для сопоставления текста и изображения

Модель ViT послужила толчком для реализации модели для сопоставления текста и изображения. Архитектура модели ViLT [16] продемонстрирована на рисунке 1.7.

Данная модель использует предобученную модель ViT для определения схожести текста и картинки. В предобученный ViT отправляется последовательность эмбеддингов. Пусть L — это длина текста, N — количество фрагментов изображения, P — размер фрагмента изображения, $|V|$ — размер словаря. Текст $t \in \mathbb{R}^{L \times |V|}$ представляется в виде по-

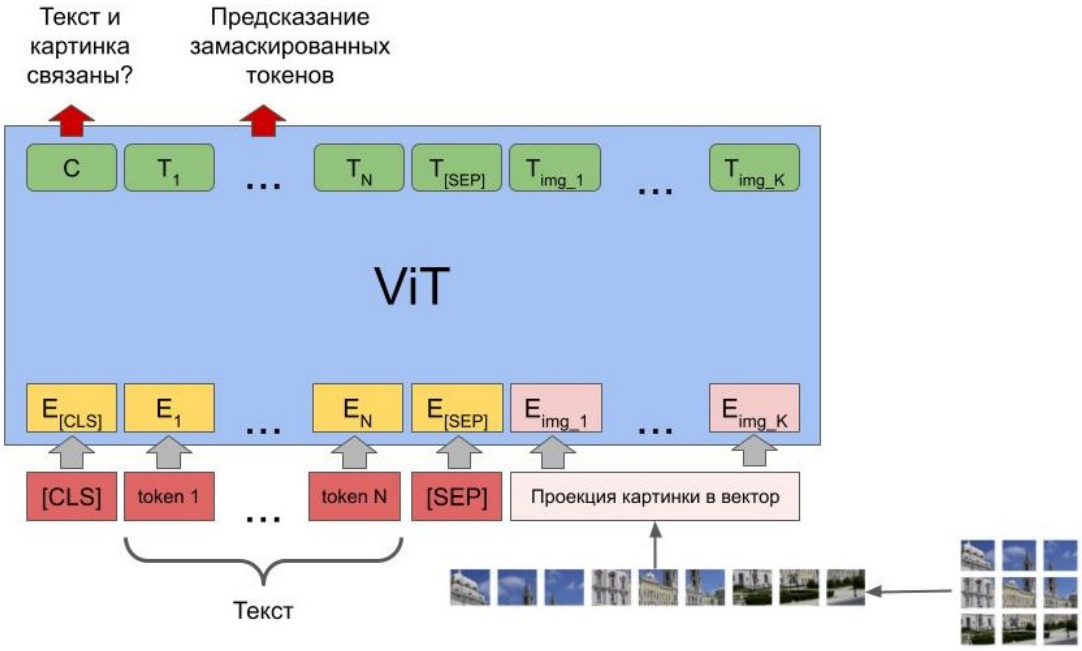

Рис. 1.7. ViT.

следовательности эмбеддингов $\bar{t} \in \mathbb{R}^{L \times D}$ с помощью матрицы эмбеддингов $\mathbf{T} \in \mathbb{R}^{|V| \times D}$, где D — размер эмбеддинга. Аналогично ViT фрагменты изображения проецируется в вектора. Разделителем для текста и изображения служит токен [SEP] и эмбеддинги типа. Заметим, что нумерация токенов текста и фрагментов изображения не сплошная. Для текста и изображения две отдельные нумерации, следовательно, у каждого свои позиционные эмбеддинги. Тогда входная последовательность выглядит следующим образом:

$$\bar{t} = [t_{cls}, t_1\mathbf{T}, \ldots, t_L\mathbf{T}, t_{sep}] + \mathbf{T}_{pos},\ \mathbf{T} \in \mathbb{R}^{|V|\times D},\ \mathbf{T}_{pos} \in \mathbb{R}^{(L+2)\times D},$$
$$\bar{v} = [v_1\mathbf{V}, \ldots, v_N\mathbf{V}] + \mathbf{V}_{pos},\ \mathbf{V} \in \mathbb{R}^{(P^2\cdot C)\times D},\ \mathbf{V}_{pos} \in \mathbb{R}^{N\times D},$$
$$z^0 = [\bar{t} + t_{type}, \bar{v} + v_{type}]$$

Полученная последовательность z^0 отправляется в ViT ($z^{end} = ViT(z^0)$).

Данная модель учится на двух задачах: предсказание замаскированных токенов слов и сопоставления текста и изображения.

1.8.1. Сопоставление текста и изображения

Пусть z^{end} — выход сети, тогда z_0^{end} — это выход соответствующий токену [CLS]. Далее данный вектор проходит через линейный слой и функцию активации гиперболический тангенс

$$p = tanh(z_0^{end}W_1),\ W_1 \in \mathbb{R}^{D\times D}.$$

Полученный вектор проходит через линейный слой, для получения вероятностей к полученным логитам применяется функция Softmax: $probs = Softmax(pW_2)$, $W_2 \in \mathbb{R}^{D\times 2}$. В качестве функции потерь берется кросс-энтропия.

1.8.2. Предсказание замаскированных токенов текста

Аналогично BERT токены маскируются с вероятностью 0.15. Далее выходы замаскированных токенов пропускаются через многослойный

персептрон. В качестве функции потерь считается кросс-энтропия для замаскированных токенов.

Для обучения модели использовались 4 датасета: Microsoft COCO (MSCOCO) [18], Visual Genome, SBU Captions [19] и Google Conceptual Captions [3]. В общей сложности более 9 миллионов пар (текст, изображение). Во время обучение прошло 100 тысяч шагов с размером батча 4096.

Модель ViLT решает проблемы предыдущих подходов. Описанный подход использует механизм внимание и принимает на вход все изображение целиком. Но большое количество данных и большой размер батча, требующиеся для обучения, являются недостатком данного подхода. Если небольшая компания захочет сделать свою систему для поиска, то повторить данные результаты будет крайне проблематично.

1.9. Выводы

Задача текстово-визуального поиска весьма популярна. В данной области существует большое количество различных подходов, которые имеют как и достоинства, так и свои недостатки. Хотелось бы создать модель, которая смогла обучаться в рамках ограниченных ресурсов. При этом модель должна обрабатывать изображение и текст вместе и получать на вход изображение целиком. Поэтому имеет смысл попробовать адаптировать модель ViLT для обучения на небольшом датасете, с

небольшим размером батча.

Следует заметить, что архитектуры моделей ViT и BERT очень похожи, а также для них есть предобученные веса. Этим фактом можно попробовать воспользоваться для создания одной модели, которая сможет работать и с текстом, и с изображением.

Глава 2

Модель

В этой главе описывается адаптация модели ViLT для обучения на небольшом количестве данных и с маленьким размером батча. Подробно показаны изменения, внесенные в архитектуру, и способ обучения. Дополнительно представлена идея дистилляции модели, которая позволяет создать одну модель для работы с текстом и изображением.

2.1. Адаптированный ViLT

В начале моей работы осуществлялась попытка обучить оригинальный ViLT на небольшом датасете с маленьким размером батча. Про полученные результаты рассказано в следующей главе. Забегая вперед, скажу, что обучить ViLT в доступных мне условиях не удалось. Поэтому потребовалось адаптировать модель для обучения на небольшом количестве данных и с маленьким размером батча.

Модель ViLT хороша тем, что благодаря механизму внимания она знает, на что нужно смотреть на картинке относительно текста и наоборот. Также модель принимает все изображение целиком, а не некоторые выборочные фрагменты. Это позволяет модели самой определить, что важно на изображение, а что нет.

Но такой модели сложно учиться определять, является ли пара

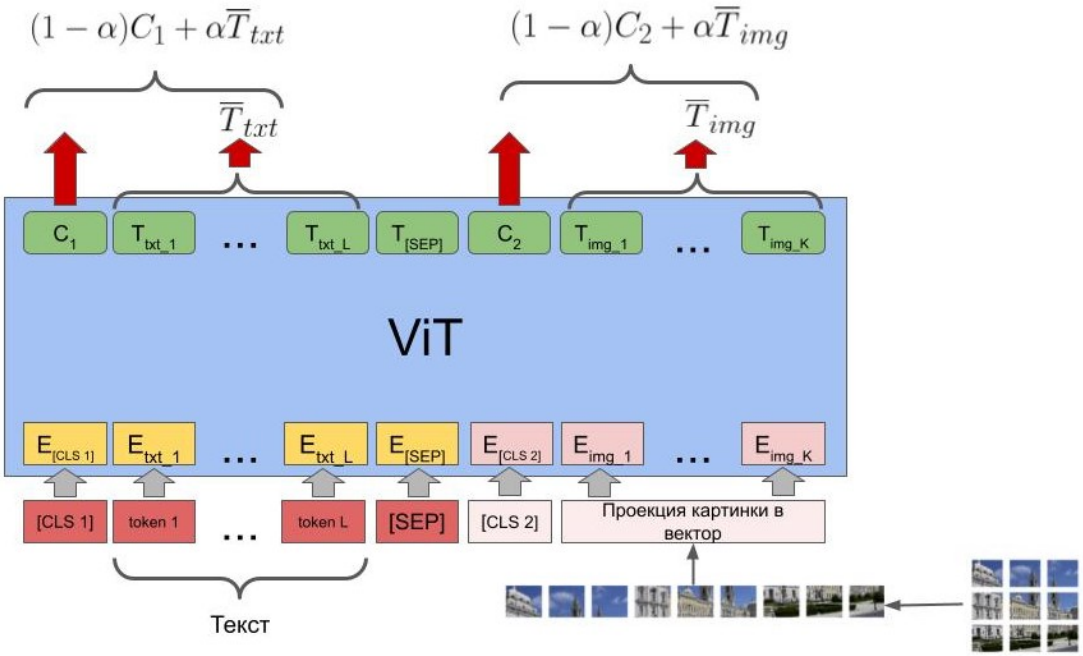

Рис. 2.1. ViLT.

(текст, изображение) верной. Появилась идея, обучать модель не просто определять, парные данные или нет, а из множества картинок искать наиболее подходящую для текущего текста. Аналогично для картинки искать парный текст. Такой способ обучения должен быть легче, так как модель будет смотреть, чем одна картинка отличается от других.

2.1.1. Архитектура

На рисунке 2.1 представлена архитектура адаптированного ViLT. Сама архитектура отличается от оригинального ViLT только тем, что добавлен второй токен [CLS 2], который находится между токенами текста и фрагментами изображений. Таким образом, есть два токена, которые

будут отвечать за классификацию пары и определять, подходит ли текст к картинке или нет. Идея заключается в том, что выходы данных токенов (на рисунке это C_1 и C_2) должны содержать в себе всю информацию о том, на что нужно обратить внимание на тексте (картинке) относительно картинки (текста).

Пусть L — длина последовательности токенов текста, $K = HW/P^2$ — количество фрагментов, $H \times W$ — изначальные размеры картинки, $P \times P$ — размер фрагмента, D — размер эмбеддинга. За T_{txt_i} обозначим выходы сети, соответствующие токенам текста, а за T_{img_i} — выходы, соответствующие фрагментам изображений, t_i — токен текста, v_i — фрагмент изображения.

Вход в сеть выглядит следующим образом:

$$\bar{t} = [t_{cls}, t_1\mathbf{T}, \ldots, t_L\mathbf{T}, t_{sep}] + \mathbf{T}_{pos},\ \mathbf{T} \in \mathbb{R}^{|V| \times D},\ \mathbf{T}_{pos} \in \mathbb{R}^{(L+2) \times D},$$
$$\bar{v} = [v_{cls}, v_1\mathbf{V}, \ldots, v_N\mathbf{V}] + \mathbf{V}_{pos},\ \mathbf{V} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{V}_{pos} \in \mathbb{R}^{K \times D},$$
$$z = [\bar{t} + t_{type}, \bar{v} + v_{type}].$$

Полученная последовательность отправляется в ViT, на выходе получаем следующее:

$$\mathrm{ViT}(z) = [C_1, T_{txt_1}, \ldots, T_{txt_L}, T_{[SEP]}, C_2, T_{img_1}, \ldots, T_{img_K}].$$

В итоге хочется, чтобы информация о тексте содержалась в векторе C_1, а о картинке в векторе C_2. Но в начале обучения никакой информации в них нет. Более того, вектора $(T_{txt_1}, \ldots, T_{txt_L})$ и $(T_{img_1}, \ldots, T_{img_K})$ содержат

больше информации о данных. Поэтому логично в начале обучения использовать вектора $(T_{txt_1}, \ldots, T_{txt_L})$ и $(T_{img_1}, \ldots, T_{img_K})$, чтобы подтолкнуть модель к обучению. Но не хочется, чтобы вся информация о тексте и картинке размазывалась по этим векторам. Будем стремиться к тому, чтобы информация концентрировалась только в двух векторах C_1 и C_2. Поэтому используется следующий трюк:

$$\hat{C}_1 = (1-\alpha)C_1 + \alpha\overline{T}_{txt}, \tag{2.1}$$

$$\hat{C}_2 = (1-\alpha)C_2 + \alpha\overline{T}_{img}, \tag{2.2}$$

где

$$\overline{T}_{txt} = \frac{1}{L}\sum_{i=1}^{L} T_{txt_i}, \quad \overline{T}_{img} = \frac{1}{K}\sum_{i=1}^{K} T_{img_i} \quad \alpha \in [0,1]$$

и α уменьшается с 1 до 0 во время обучения.

Полученные взвешенные суммы векторов проходят через линейные слои:

$$o_{txt} = f_{txt}(\hat{C}_1) \triangleq \hat{C}_1 \mathbf{W}_{txt},$$

$$o_{img} = f_{img}(\hat{C}_2) \triangleq \hat{C}_2 \mathbf{W}_{img},$$

где $\mathbf{W}_{txt} \in \mathbb{R}^{D\times D}$, $\mathbf{W}_{img} \in \mathbb{R}^{D\times D}$. Полученные вектора скалярно умножаем, тогда выход сети $out = o_{txt} \cdot o_{img}^{\mathrm{T}}$. Таким образом, $sigmoid(out)$ будет показывать, насколько текст и картинка схожи.

2.1.2. Сценарий обучения

Как было указано выше, хочется, чтобы вектор C_1 владел всей информацией о тексте при условие, что он поступил на вход с определенным изображением. А вектор C_2 владел всей информацией об изображение при условие, что он поступил на вход с определенным текстом. Но в начале обучения они ничего не знают про данные, и модели очень тяжело учиться, используя только эти два вектора. Поэтому для обучения дополнительно используются средние значения выходов текста и изображения (2.1–2.2).

В начале обучения $\alpha = 1$, но по мере обучения уменьшается до нуля. Во время обучения были установлены следующие критерии уменьшения α:

- новая $new\_\alpha = c \cdot \alpha$, где $c = 0.9$;
- α уменьшается, если функция потерь меньше b, где $b = 1$.

Ясно, что со значениями коэффициентов c, b можно экспериментировать.

Обучение происходит батчами размера N. После прохождения батча через сеть на выходе получаем N векторов $\hat{C}_1$ и такое же количество векторов $\hat{C}_2$. Если скалярно перемножить каждый вектор с каждым, то получим матрицу $\mathbf{M} \in \mathbb{R}^{N \times N}$.

На рисунке 2.2 слева представлена полученная матрица $\mathbf{M}$, если все входные данные парные. Строки матрицы соответствуют определенному тексту, а столбцы — изображению. Значения на диагонали соответствует

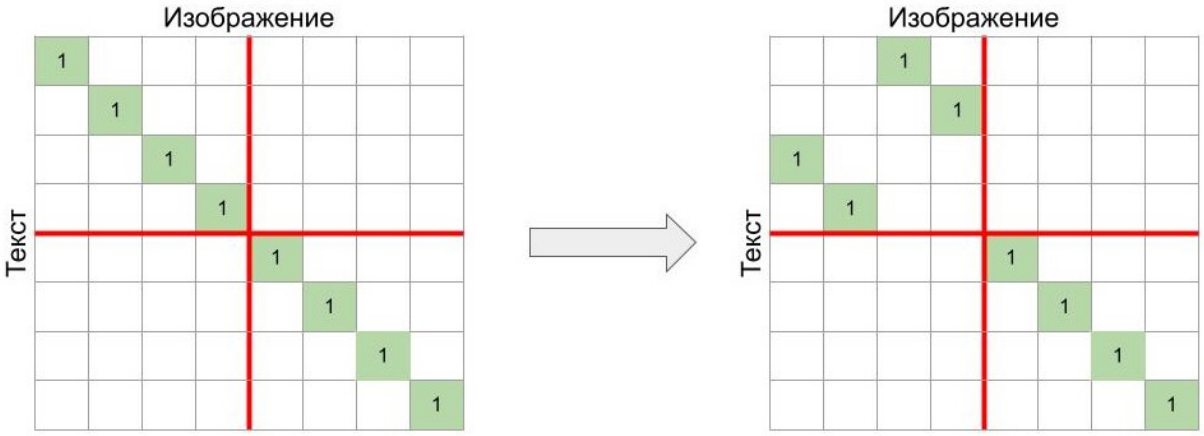

Рис. 2.2. Выход сети.

тексту и изображению, которые подавались в модель вместе. Зеленная клетка с единичкой на рисунке 2.2 означает, что соответствующие текст и изображение парные.

Так как обучение должно проходить и на парных, и непарных данных, то половина данных в батче перемешивается. И матрица приобретает вид такой, как на рисунке 2.2 справа.

Обучение происходит с помощью трех функций потерь:

1. бинарная кросс-энтропия для предсказания схожести текста и картинки;

2. кросс-энтропия по строке для предсказания подходящей картинки для текста;

3. кросс-энтропия по столбцу для предсказания подходящего текста для картинки;

Пусть $y_i \in \{0,\ 1\}$ означает парность элементов, $\sigma(x) = \frac{1}{1+e^{-x}}$ – функция сигмоида. Тогда итоговая формула для функции потерь следующая:

$$\mathscr{L}_1 = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log\big(\sigma(\mathrm{M}_{ii})\big) + (1-y_i)\log\big(1-\sigma(\mathrm{M}_{ii})\big)\Big],$$

$$\mathscr{L}_2 = -\frac{1}{N}\sum_{i=1}^{N}\frac{\exp(\mathrm{M}_{ii^*})}{\sum\limits_{j=1}^{N}\exp(\mathrm{M}_{ij})},$$

$$\mathscr{L}_3 = -\frac{1}{N}\sum_{j=1}^{N}\frac{\exp(\mathrm{M}_{j^*j})}{\sum\limits_{i=1}^{N}\exp(\mathrm{M}_{ij})},$$

$$\mathscr{L} = \mathscr{L}_1 + \mathscr{L}_2 + \mathscr{L}_3,$$

где i^* обозначает номер столбца (картинки) парный для строки (текста) i, а j^* — номер строки парной для столбца j.

Переход к такому обучению дает модели более явный сигнал, чем использование только бинарной кросс энтропии, поэтому модели гораздо легче обучаться. К тому же такой подход требует от модели уделять больше внимания взаимосвязи токенов одной модальности. Это нужно, так как мы не хотим излишних взаимосвязей токенов текста и изображения на непарных данных.

2.2. Дистилляция модели

Определение 4. Дистилляция модели — это построение простой модели на основе сложной или нескольких моделей.

Есть две большие модели BERT и ViT, которые хорошо работают каждый со своим типом данных. Эти модели будем называть учителями. Хотелось бы обучить одну модель, которая будет хорошо работать и с текстом, и с изображениями. Эту модель назовем сетью-студентом.

2.2.1. Первый способ

На рисунке 2.3 представленна архитектура для дистилляции модели. За основу была взята статья [5], где описывается дистилляция модели BERT.

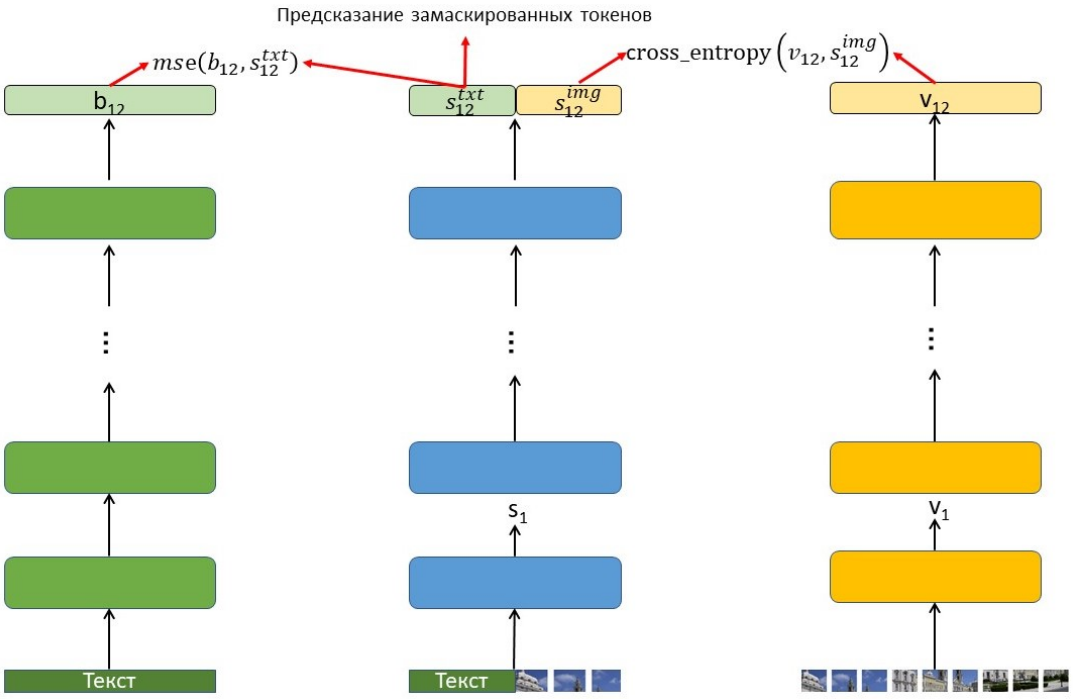

Рис. 2.3. Первый способ дистилляции модели.

Пусть архитектура модели-студента полностью повторяет архитектуру модели ViT и инициализируется предобученными весами ViT. На

вход модель-студент принимает текст и изображение вместе с токенами [CLS 1], [CLS 2], [SEP] (аналогично адаптированному ViLT). Чтобы модель-студент научилась работать с текстом, будем учить ее выдавать выход близкий к выходу BERT. Дополнительно будем маскировать токены с вероятностью 0.15 и обучать сеть предсказывать их. ViT обучался на задаче классификации изображений. Поэтому сеть-студент будет учиться выдавать классификацию аналогичную ViT.

Введем обозначения. Пусть $v_i,\ i = 1,\ldots,12$ — выходы блоков ViT, $b_i,\ i = 1,\ldots,12$ — выходы блоков BERT, $s_i,\ i = 1,\ldots,12$ — выходы блоков сети-студента. Заметим, что выходы s_i можно поделить на две части: первая часть выхода относится к тексту, вторая — к картинке. Обозначим за $s_i^{txt} = [(s_i)_2,\ldots(s_i)_{L+2}]$ (без токена [CLS 1], но с [SEP]), где L — длина текста, $s_i^{img} = [(s_i)_{L+4},\ldots(s_i)_{L+3+K}]$ (без токена [CLS 2]), где K — количество фрагментов изображения. Пусть s_{class}^{img} соответствует выходу токена классификации [CLS 2], v_{class} – выходу токена классификации сети ViT.

Чтобы сеть предсказывала замаскированные токены, пропустим s_{12}^{txt} через многослойный перцептрон: $l = \mathbf{MLP}(s_{12}^{txt}) \in \mathbb{R}^{L\times|V|}$, где $|V|$ — размер словаря. Для получение вероятностей воспользуемся функцией Softmax: $y_i = Softmax(l_i)$. Пусть $t_i \in \mathbb{R}^{|V|}$ — one-hot вектор, показывающий, какой токен должен стоять на i-ом месте в тексте. Для задачи классификации выходы ViT и сети-студента проходят через линейный слой и функцию Softmax: $y^s = Softmax(s_{class}^{img}W_s),\ y^v = Softmax(v_{class}W_v)$, где $\mathbf{W}_s \in \mathbb{R}^{D\times M},\ \mathbf{W}_v \in \mathbb{R}^{D\times M}$, M — количество классов.

Тогда функция потерь выглядит следующим образом:

$$\mathscr{L}_1 = mse(b_{12}, s_{12}^{txt}), \tag{2.3}$$

$$\mathscr{L}_2 = -\sum_i t_i \log y_i, \tag{2.4}$$

$$\mathscr{L}_{BERT} = (1-\alpha)\mathscr{L}_1 + \alpha\mathscr{L}_2,\ \alpha \in [0,1] \tag{2.5}$$

$$\mathscr{L}_{ViT} = -\sum_{i=1}^{M} y_i^v \log(y_i^s), \tag{2.6}$$

$$\mathscr{L} = a\mathscr{L}_{BERT} + b\mathscr{L}_{ViT},\ a,\ b \in \mathbb{R}_+. \tag{2.7}$$

Авторы модели ViT предоставляют предобученные веса. К сожалению, они не содержат весов линейного слоя для классификации (в текущих обозначениях это матрица W_v). Данный факт не позволяет использовать предложенный способ дистилляции. Но также авторы предоставляют веса после дообучения (fine-tuning) на датасете imagenet2012, которые содержат веса W_v.

2.2.2. Второй способ

Второй способ дистилляции модели (рис. 2.4) позволяет использовать веса ViT после предобучения.

В первой главе описывалась архитектура модели ViT и BERT. Там же было замечено, что хотя обе модели построены на основе трансформеров, но отличаются порядком слоя LN (Layer Normalization).

Исходя из этого замечания, предлагается следующий способ дистилляции модели. Часть, отвечающая за дистилляцию модели BERT, оста-

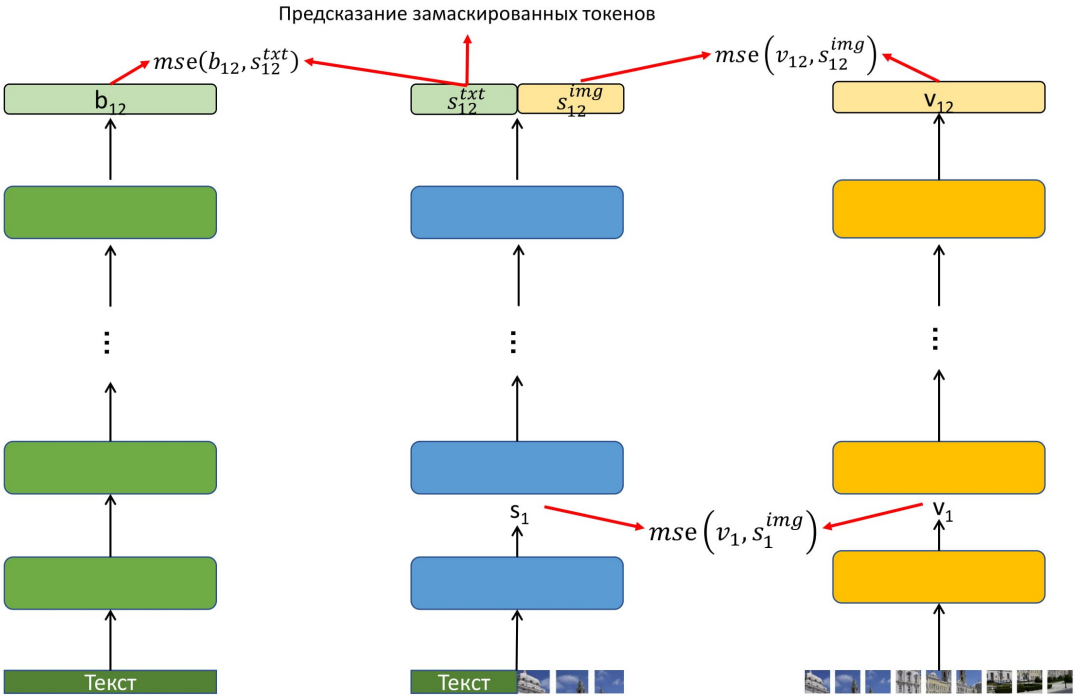

Рис. 2.4. Второй способ дистилляции модели.

ется прежней. Меняется только способ дистилляции модели ViT. Сеть-студент будет учиться выдавать после каждого блока такие же значения, какие выдают блоки ViT.

Тогда функция потерь выглядит следующим образом:

$$\mathscr{L}_1 = mse(b_{12}, s_{12}^{txt}), \tag{2.8}$$

$$\mathscr{L}_2 = -\sum_i t_i \log y_i, \tag{2.9}$$

$$\mathscr{L}_{BERT} = (1-\alpha)\mathscr{L}_1 + \alpha\mathscr{L}_2, \ \alpha \in [0,1] \tag{2.10}$$

$$\mathscr{L}_{ViT} = mse([s_1^{img}, \ldots, s_{12}^{img}], [v_1, \ldots, v_{12}]), \tag{2.11}$$

$$\mathscr{L} = a\mathscr{L}_{BERT} + b\mathscr{L}_{ViT}, \ a, \ b \in \mathbb{R}_+. \tag{2.12}$$

Заметим, что нельзя было для BERT сделать функцию потерь по блокам аналогично ViT. Проблема заключается в разной последовательности слоя LN в блоках BERT и ViT. Блоки BERT выдают нормализованные выходы в отличие от блоков сети-студента. Следовательно, приближать ненормализованные выходы к нормализованным было бы неверно.

2.3. Выводы

В этой главе подробно описана предложенная архитектура для сопоставления текста и изображения, а также способ ее обучения. Данная архитектура сохраняет плюсы модели ViLT: основывается на трансформере и принимает на вход изображение целиком. Кроме того, в этой главе представлены 2 способа дистилляции модели, которые позволяют объединить модели BERT и ViT в одну.

Глава 3

Эксперименты

В предыдущей главе подробно описана архитектура адаптированного ViLT и способ обучения. А также приведены способы дистилляции модели. В этой главе рассказано о данных, на которых обучалась модель, о технических деталях модели и ее обучения, а также описаны различные эксперименты и их результаты.

3.1. Данные

Самые популярные датасеты для задачи сопоставления текста и изображения — это Flickr30k [10] и MS COCO [18]. Оба датасета устроены одинаково: для каждой картинки есть 5 текстовых описаний. На рисунке 3.1 представлены примеры изображений с подписями из датасета MS COCO. Тренировочная выборка Flickr30k содержит 31,783 изображений, а MS COCO 82,783. Обычно в статьях приводят результаты обучения на обоих датасетах, которые чаще всего коррелируют. В представленных экспериментах использовался датасет MS COCO. Дистилляция так же проходила на этом датасете. Данный датасет содержит в валидационной выборке 40,504 изображений, из которых 5,000 изображений выделено для тестирования модели. В таблице 3.1 представленно конечное распределение данных.

Рис. 3.1. Примеры изображений с подписями из датасета MS COCO.

	Train	Validation	Test
MS COCO	82,783	35,504	5,000

Таблица 3.1. Распределение данных.

3.2. Параметры обучения

Как уже говорилось выше, авторы модели ViT предоставляют предобученные веса. Для своих экспериментов я взяла веса модели ViT-B/16 после дообучения на датасете imagenet2012. Данные веса позволят использовать первый способ дистилляции модели. Число 16 означает ширину и высоту фрагментов, на которые делят изображение ($P = 16$). Буква B означает ***Base***, такая модель имеет 12 блоков, размер эмбеддинга $D = 768$, 12 голов в трансформере и размер выхода промежуточного слоя в MLP равный 3072.

Максимальная длина входной последовательности у ViT 577 (на токен [CLS] и фрагменты изображений). Следовательно, максимальный размер картинки 384×384. Но у адаптированной модели, так же как и у модели ViLT, вход состоит из текста и картинки, поэтому во всех проведенных экспериментах размер изображения приводился к 224×224.

В работе используется токенайзер bert-base-uncased для входного текста. В качестве эмбеддингов текста берутся предобученные эмбеддинги BERT. Все это предоставляется в библиотеке Hugging Face [22].

Для реализации адаптированного ViLT использовалась библиотека Hugging Face. А для обучения — библиотека PyTorch Lightning [7], которая позволяет распараллелить обучение на несколько GPU.

3.3. Обучение оригинального ViLT

В начала исследования предпринималась попытка обучить ViLT на датасете MS COCO на 3 GPU с общим размером батча 300. График обучения представлен на рисунке 3.2. Во время всего обучения значение бинарной кросс-энтропии колышется около значения 0.69. После получения таких результатов была проведена серия экспериментов. Например, проверялось может ли модель выучить датасет, где картинки только черные или белые, а подходящий текст "black" и "white" соответственно. Модель успешно справилась с заданием, хорошо определяя, верную пару.

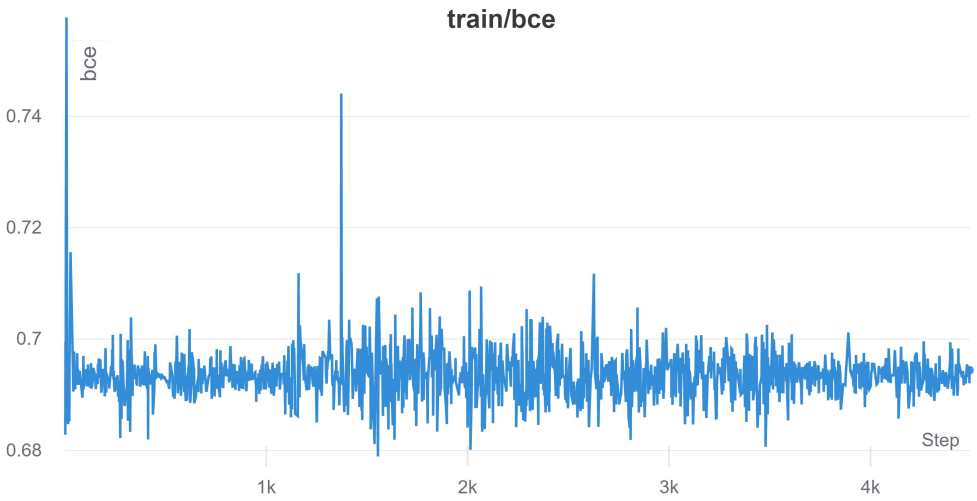

Рис. 3.2. График обучения ViLT.

Также ViLT обучался всего лишь на четырех парах из датасета MS COCO. График обучения представлен на рисунке 3.3. Модель выучила выбранные пары и могла определить, когда пара была верная, а когда нет. Исходя из этих экспериментов, был сделан вывод, что модель ViLT

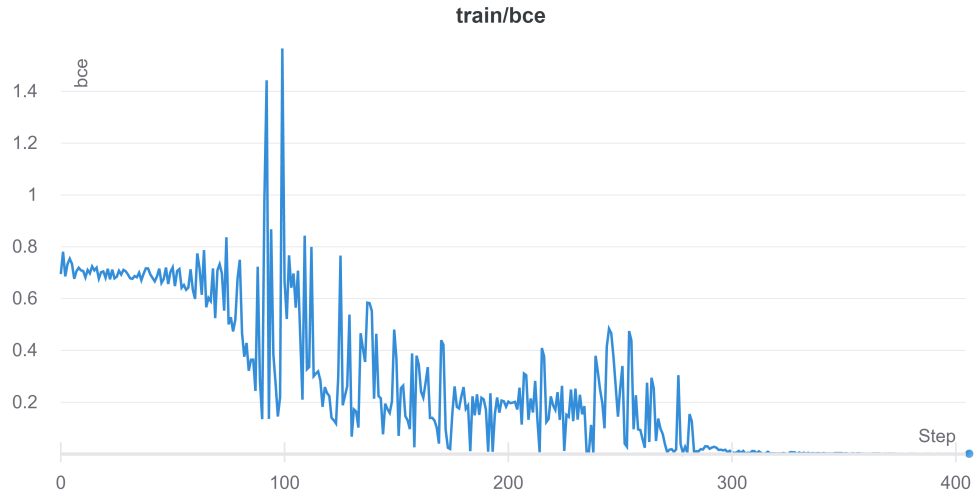

Рис. 3.3. График обучения ViLT на 4 парах.

может обучаться, но не в доступных мне условиях.

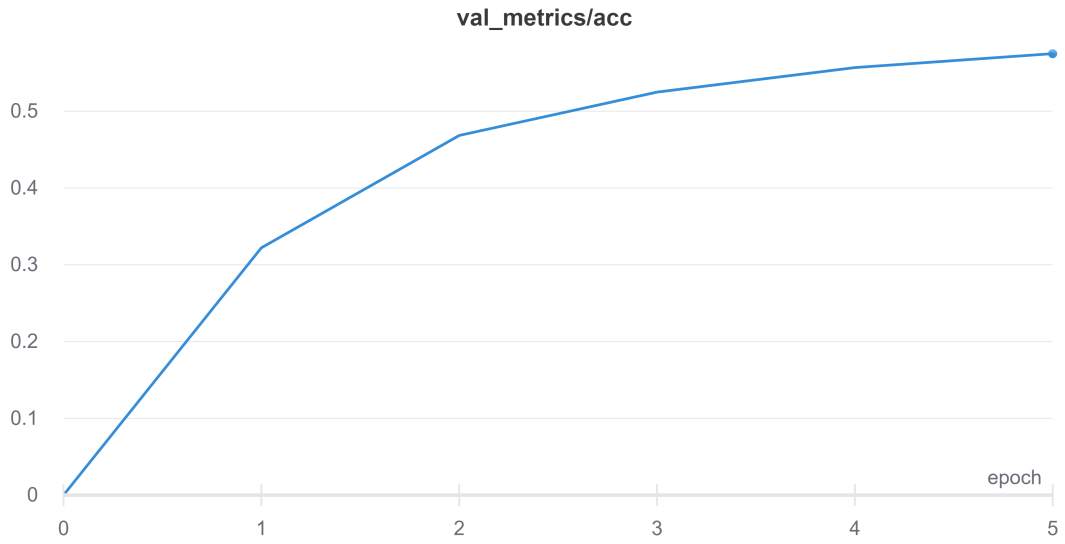

Рис. 3.4. График точности предсказания замаскированных токенов на валидационной выборке.

3.4. Дистилляция модели

Дистилляция модели происходила первым способом, на непарных данных, на 2 GPU с общим размером батча 80. Использовался датасет MS COCO, но пары перемешивались, чтобы на вход не подавались парные данные.

К сожалению второй способ дистилляции не был проверен. Данная дистилляция остается для дальнейшего исследования.

На рисунке 3.4 изображено, как растет точность предсказания замаскированных токенов во время обучения на валидационной выборке. Следовательно, можно утверждать что модель перенимает свойства BERT. К сожалению, такой проверки для свойств ViT нет, так как нам неизвестны истинные классы изображений в датасете MS COCO. Но бы-

ло просмотренно некоторое колличество классификаций модели для разных изображений из датасета imagenet2012. В таблице 3.2 представленны классификации (класс и его вероятность) модели ViT и модели-студента после дистилляции для изображения 3.5.

Рис. 3.5. Изображение.

Таблица 3.2. Классификация изображения.

Модель-студент	ViT
0.615 : thresher	0.665 : harvester
0.356 : harvester	0.306 : thresher
0.0074 : tractor	0.0121 : hay
0.0042 : hay	0.0071 : tractor
0.0019 : plough	0.0017 : plough

3.5. Обучение адаптированного ViLT

Адаптированный ViLT так же обучался на датасете MS COCO на 2 GPU с общим размером батча 200. Обучение происходило с разным количеством данных: с 1%, 5%, 10%, 100% данных от тренировочной выборки. Таким образом, можно проследить, как дистилляция на большом объеме непарных данных улучшает результат обучения на маленьком объеме данных.

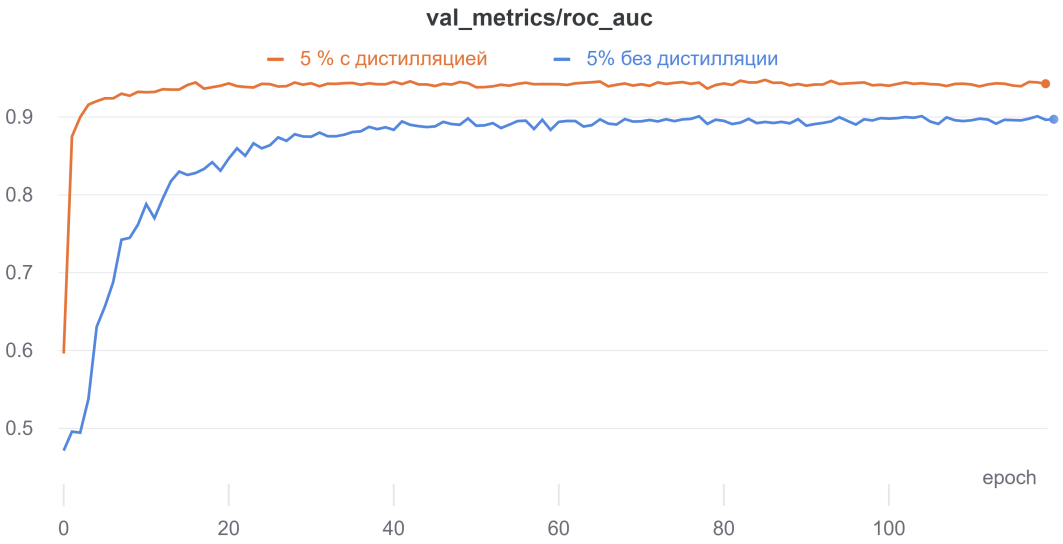

Рис. 3.6. График ROC AUC на валидационной выборке во время обучения на 5% тренировочной выборки. Красный график — адаптированный ViLT после дистилляции. Синий график — адаптированный ViLT с весами ViT.

На рисунке 3.6 представлены два графика ROC AUC на валидационной выборке во время обучения на 5% тренировочных данных. Синий график соответствует модели инициализированной весами ViT. Красный график соответствует модели с весами после дистилляции первым способом. По графикам видно, что модель с весами после дистилляции учится намного лучше.

3.6. Результаты

Качество моделей для текстово-визуального поиска измеряется метрикой $R@k$. Считается она следующим образом: для конкретного текста берется 1000 картинок, одна из которых парная к рассматриваемому тек-

Имя модели	Доля данных	Поиск текста по картинке			Поиск картинки по тексту		
		R@1	**R@5**	**R@10**	**R@1**	**R@5**	**R@10**
ViLT[a]	100%	**61.8**	**86.2**	**92.6**	**41.3**	**72.0**	**82.5**
Adapt ViLT	100%	18.3	37.2	49.8	11.8	31.4	45.0
D Adapt ViLT	100%	**20.2**	**43.6**	**56.4**	**16.6**	**38.1**	**51.0**
Adapt ViLT	10%	2.9	10.8	17.3	3.3	10.4	16.8
D Adapt ViLT	10%	**5.8**	**18.7**	**27.9**	**4.1**	**13.5**	**21.8**
Adapt ViLT	5%	1.1	6.5	10.5	1.1	5.1	9.5
D Adapt ViLT	5%	**4.1**	**14.0**	**21.8**	**3.2**	**11.9**	**19.1**
Adapt ViLT	1%	0.9	2.8	4.5	0.4	1.9	3.9
D Adapt ViLT	1%	**2.4**	**8.9**	**14.9**	**1.3**	**5.5**	**9.7**

Таблица 3.3. Метрика $R@k$.

[a] $R@k$ считались на тестовой выборке размера 5000.

сту. Все картинки ранжируются по выходу сети. Далее смотрят, попала ли парная картинка в топ k. Так проходят по всем текстам и считают среднее попадание в топ. Аналогично для поиска текста по картинке.

В таблице 3.3 представлены посчитанные $R@k$ для адаптированного ViLT (Adapt ViLT) с весами ViT и с весами после дистилляции модели (D Adapt ViLT), а также результаты оригинально ViLT, взятые из статьи. Модели обучались на различных долях данных: 100%, 10%, 5%, 1%.

Видно, что адаптированный ViLT не смог добиться таких же результатов, как оригинальный ViLT. Но замечу, что обычный ViLT вовсе не учился на таком маленьком количестве данных и с небольшим размером батча.

Также видно, что дистилляция во всех случаях улучшила результаты. Полученная мульти-модальная модель хорошо работает с двумя разными типами данных, поэтому ей было легче находить связи между текстом и картинкой и обучиться их сопоставлению.

Данные результаты показывают, что увеличение непарных данных на дистилляции может улучшить конечный результат дообучения на маленьком датасете с парными данными. Так как непарные данные найти достаточно легко, в отличие от парных, то следует дистиллировать модель на большом объеме непарных данных. Далее дообучить на задаче сопоставления текста и изображения на небольшом датасете, таком как MS COCO. Это должно намного улучшить значение метрик $R@k$.

3.7. Выводы

В этой главе описаны детали обучения, рассказано о данных, на которых обучалась модель, и о технических деталях реализации модели. Также описаны проводимые эксперименты и представлены результаты.

Заключение

В данной работе разобрана задача сопоставления текста и изображения. Описаны различные подходы решения, отмечены их достоинства и недостатки.

Главным результатом дипломной работы стала адаптация модели ViLT для обучения на небольшом датасете и с маленьким размером батча. Полученная модель основывается на трансформере и принимает на вход все изображение, что является плюсом. Такими качествами на данный момент может похвастаться только модель ViLT, но которую, к сожалению, нельзя обучить в рамках ограниченного количества ресурсов.

Еще одним важным результатом является предложенный способ дистилляции модели. Показано, что данная дистилляция намного улучшает качество модели.

В ходе работы были получены следующие результаты:

- продемонстрирована неспособность к обучению модели ViLT при ограниченных ресурсах;
- реализованна модель адаптированного ViLT и показано, что полученная модель обучается на небольшом датасете и с маленьким размером батча
- приведены способы дистилляции модели;
- продемонстрированы результаты модели с дистилляцией и без ди-

стилляции и показано, что дистилляция улучшает качество итоговой модели.

К сожалению, полученная модель не достигает таких же результатов, как ViLT, обученный на большом датасете и с большим размером батча. Но это может стать одним из направлений дальнейшего исследования.

По-моему мнению, одно из приоритетных направлений является исследование взаимосвязи результатов обучения адаптированной модели от количества тренировочных данных. Нужно определить, может ли данная модель достигнуть схожих результатов с ViLT при небольшом увеличение размера датасета.

Также следует попробовать сделать дистилляцию модели на большом количестве непарных данных. Это должно намного улучшить качество модели. Данное направление очень перспективно, так как мультимодальная модель, которая может хорошо обрабатывать как текстовые, так и визуальные данные, может помочь в решение задачи текстово-визуального поиска.

Список литературы

1. Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate // arXiv:1409.0473. — 2016.
2. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). — Minneapolis, Minnesota : Association for Computational Linguistics, 2019. — Jun. — P. 4171–4186. — Access mode: https://www.aclweb.org/anthology/N19-1423.
3. Conceptual captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning / Piyush Sharma, Nan Ding, Sebastian Goodman, Radu Soricut // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Melbourne, Australia : Association for Computational Linguistics, 2018. — Jul. — P. 2556–2565. — Access mode: https://www.aclweb.org/anthology/P18-1238.
4. Deep Canonical Correlation Analysis / Galen Andrew, Raman Arora, Jeff Bilmes, Karen Livescu // Proceedings of the 30th International Conference on Machine Learning / Ed. by Sanjoy Dasgupta, David McAllester. — Vol. 28 of Proceedings of Machine Learning Research. — Atlanta, Geor-

gia, USA : PMLR, 2013. — P. 1247–1255. — Access mode: http://proceedings.mlr.press/v28/andrew13.html.

5. Distilling Task-Specific knowledge from BERT into Simple Neural Networks / Raphael Tang, Yao Lu, Linqing Liu et al. // arXiv:1903.12136. — 2019.
6. Dual-Path Convolutional Image-text Embedding with Instance Loss / Zhedong Zheng, Liang Zheng, Michael Garrett et al. // arXiv:1711.05535. — 2018.
7. Falcon WA e. a. Pytorch Lightning // GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning. — 2019. — Vol. 3.
8. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks / Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun // arXiv:1506.01497. — 2016.
9. Finding beans in burgers: Deep semantic-visual embedding with localization / Martin Engilberge, Louis Chevallier, Patrick Perez, Matthieu Cord // arXiv:1804.01720. — 2018.
10. Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models / Bryan A. Plummer, Liwei Wang, Chris M. Cervantes et al. // arXiv:1505.04870. — 2016.
11. Generative Pretraining From Pixels / Mark Chen, Alec Radford, Rewon Child et al. // Proceedings of the 37th International Conference on Machine Learning / Ed. by Hal Daume III, Aarti Singh. — Vol. 119 of Proceedings of Machine Learning Research. — PMLR, 2020. — P. 1691–1703. —

Access mode: http://proceedings.mlr.press/v119/chen20s.html.

12. Hardoon D. R., Szedmak S., Shawe-Taylor J. Canonical Correlation Analysis: An Overview with Application to Learning Methods // Neural computation, 16(12):2639–2664. — 2004.
13. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale / Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov et al. // arXiv preprint arXiv:2010.11929. — 2020.
14. ImageBERT: Cross-modal Pre-training with Large-scale Weak-supervised Image-Text Data / Di Qi, Lin Su, Jia Song et al. // arXiv:2001.07966. — 2020.
15. Imagenet: A large-scale hierarchical image database / Jia Deng, Wei Dong, Richard Socher et al. // 2009 IEEE Conference on Computer Vision and Pattern Recognition. — 2009. — P. 248–255.
16. Kim W., Son B., Kim I. ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision // arXiv:2102.03334. — 2021.
17. Language Models are Few-Shot Learners / Tom B. Brown, Benjamin Mann, Nick Ryder et al. // arXiv:2005.14165. — 2020.
18. Microsoft COCO: Common Objects in Context / Tsung-Yi Lin, Michael Maire, Serge Belongie et al. // arXiv:1405.0312. — 2015.
19. Ordonez V., Kulkarni G., Berg T. Im2Text: Describing Images Using 1 Million Captioned Photographs // Advances in Neural Information Processing Systems / Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett et al. — Vol. 24. — Curran Associates, Inc., 2011. — Ac-

cess mode: https://proceedings.neurips.cc/paper/2011/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

20. Schuster M., Paliwal K. Bidirectional recurrent neural networks // IEEE Transactions on Signal Processing. — 1997. — Vol. 45, no. 11. — P. 2673–2681.
21. Stacked Cross Attention for Image-Text Matching / Kuang-Huei Lee, Xi Chen, Gang Hua et al. // arXiv:1803.08024. — 2018.
22. Transformers: State-of-the-Art Natural Language Processing / Thomas Wolf, Lysandre Debut, Victor Sanh et al. // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. — Online : Association for Computational Linguistics, 2020. — P. 38–45. — Access mode: https://www.aclweb.org/anthology/2020.emnlp-demos.6.
23. VisualBERT: A Simple and Performant Baseline for Vision and Language / Liunian Harold Li, Mark Yatskar, Da Yin et al. // arXiv:1908.03557. — 2019.
24. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives / Fartash Faghri, David J. Fleet, Jamie Ryan Kiros, Sanja Fidler // arXiv:1707.05612. — 2018.
25. Wang L., Li Y., Lazebnik S. Learning deep structure-preserving image-text embeddings // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2016. — P. 5005–5013.
26. Yan F., Mikolajczyk K. Deep correlation for matching images and text //

2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2015. — P. 3441–3450.