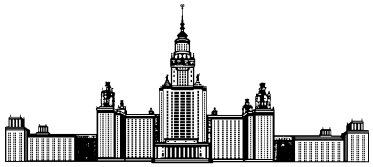
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

КУРСОВАЯ РАБОТА СТУДЕНТА 517 ГРУППЫ

«Сравнительный анализ нейросетевых архитектур в задаче поиска изображений»

Выполнил:

студент 1 курса магистратуры 517 группы

Федоров Илья Сергеевич

Научный руководитель:

д.ф-м.н., профессор

Дьяконов Александр Геннадьевич

Москва, 2022

Содержание

Аннотация

В работе представлено сравнение концептуально разных архитектур нейросетей применительно к задаче поиска изображений с помощью обучения метрики: сверточные сети, трансформеры, полносвязные сети. Для тестирования использовались наборы данных CUB-200 (2010), Stanford Online Product, In-Shop. В качестве функции потерь использовался классический Triplet Loss. В результате экспериментов было установлено, что наилучшие результаты показывают сети, основанные на архитектуре трансформера. При этом с точки зрения баланса точности и скорости работы лучшие результаты показал Swin Transformer.

1 Введение

Поисковые системы давно стали неотъемлемой частью жизни современного человека. Можно без преувеличения сказать, что на сегодняшний день быстрый доступ к актуальной информации чрезвычайно необходим для большинства исследователей, инженеров, программистов, и в целом представителей быстро развивающихся профессий. Можно отметить, что в последнее десятилетие активно развивается поиск не только по тексту, но и по другому модальностям. Наиболее успешными областями можно назвать поиск по изображениям (Google Images, Яндекс.Картинки) и поиск по аудио (Shazam). Данная работа сосредоточена на поиске похожих изображений.

1.1 Постановка задачи и цель исследования

Задача поиска схожих объектов хорошо известна исследователям. В общем виде она обычно ставится следующим образом: имея множество объектов S и объект-запрос q, система должна выдать k наиболее похожих объектов из S, наиболее похожих на q. При этом определение схожести зависит от задачи. Задача легко решается, если удается выбрать удачное математическое представление объектов и метрику. К примеру, в случае, когда объекты описываются векторами из n-мерного пространства $\mathbb{R}^n$, а мерой схожести является типичная математическая метрика (к примеру, евклидово расстояние или косинусная мера), то задача может быть легко решена как непосредственным перебором, так и с помощью более эффективных приближенных методов поиска ближайших соседей[13].

Задачу построения по множеству объектов их векторных представлений решает область машинного обучения под названием обучение метрики (metric learning). Данная область, в свою очередь, подразделяется на две подобласти: обучение с учителем и без него. Наличие "учителя"означает, что, помимо непосредственно самих объектов, также доступны метки (labels, лейблы) классов, к которым принадлежат объекты обучающей выборки. К примеру, решая задачу распознавания лиц, нашими объектами будут изображения человеческих лиц, а метками – идентификаторы людей, которым они принадлежат. Целью задачи обучения метрики с учителем является преобразование пространства представлений объектов таким образом, чтобы

сэмплы с одинаковыми метками классов были расположены ближе (относительно некоторой фиксированной метрики), а с разными – далеко. Непосредственно преобразование представлений производится с помощью некоторой функции, к примеру, в случае работы с гомогенными данными, такими как изображения, аудио, видео и текст, чаще всего используются нейросети. В этом случае говорят о глубоком обучении метрики (deep metric learning). При этом ключевой задачей при решении подобной задачи с помощью нейросетевой модели является её обобщающая способность: правильно обученная модель должна быть способна располагать близко к друг к другу объекты даже тех классов, которые не были представлены в обучающей выборке. К примеру, при решении задачи распознавания лиц, в обучающей выборке мы имеем лишь конечно множество людей и соответствующих им лиц. При этом мы хотим, чтобы в дальнейшем модель также могла различать и те лица, которые не были представлены в обучающей выборке.

С началом активного развития нейросетевых технологий, исследователи постепенно приходили к различным архитектурам, наиболее подходящим для той или иной задачи. Изначально изобретенный многослойный перспептрон (multilayer perceptron, MLP) демонстрировал достаточно слабые результаты при работе с изображениями. Для работы с данными данной модальности до недавних пор доминирующее место занимали сверточные нейросети (convolutional neural networks, CNN). Наиболее успешное применение данной архитектуры обязано наличию операций (свертки, пулинги), которые спроектированны специально для данного домена. Выбор удачных для определенного домена базовых операций в архитектуре называется индуктивным смещением (inductive bias, термин редко встречается на русском языке). Они позволяют модели эффективно обучиться на обучающей выборке сравнительно небольшого размера. Однако с увеличением размеров доступных наборов данных (datasets, датасетов), исследователи стали[4] замечать, архитектура играет уже не столь высокую роль, как раньше. В связи с этим были изобретены такие модели, как ViT[4], основанный на архитектуре Трансформера, изначально изобретенного для обработки естественного языка, и ResMLP[22], который в сущности является немного измененным MLP. Таким образом, можно считать, что эволюция нейросетевых архитектур сделала полный круг.

Целью данной работы является сравнение концептуально различных архитектур нейросетей для решения задачи глубокого обучения метрики применительно к изображениям. Рассмотрены уже описанные выше архитектуры:

1. сверточные нейросети (EfficientNet V1[20]/V2[21]),
2. нейросети, основанные на трансформере (ViT[4], SWin[10]),
3. полносвязные архитектуры (ResMLP[22]).

Сравнение проводится на трех наиболее популярных наборах данных для поиска изображений: CUB-200[24], Stanford Online Product[17], In-Shop[11]. При этом используется используется наиболее популярная функция потерь: Triplet Loss[15].

1.2 Обзор литературы

Подобные сравнение периодически публикуются, по ходу выхода новых прорывных нейросетевых моделей. Наиболее похожей работой на данную является[5]. В ней авторы также проводят аналогичные эксперименты, однако выбирают немного другой набор моделей. К примеру, на момент выхода статьи еще не были предложены модели Swin Transformer и ResMLP, а также вместо сверточной архитектуры EfficientNet, показывающей на данный момент лучшие результате в бенчмарке ImageNet, была использована более старая архитектура ResNet-50.

2 Постановка эксперимента

Прежде чем перейти к описанию датасетов, отметим важные детали тестирования metric learning моделей. Одной важной особенностью является метод разделения выборки. Иногда встречается сплит на три подвыборки: обучение (train) T, галерея (gallery) G и запросы (query) Q. В этом случае предлагается обучить модель на T, составить базу (индекс) объектов из G и выполнять для объектов из Q ближайших соседей в этой базе. В ином случае составителями датасетов предлагается более типичное для задачи классификации разделение на две подвыборки: обучение (train) и тест (test). В этом случае множества G и Q совпадают, то есть для каждого объекта

Рис. 1: Примеры из CUB-200

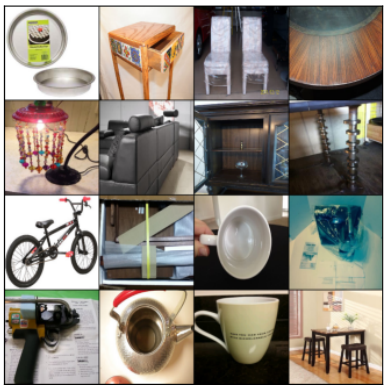


Рис. 2: Примеры из SOP

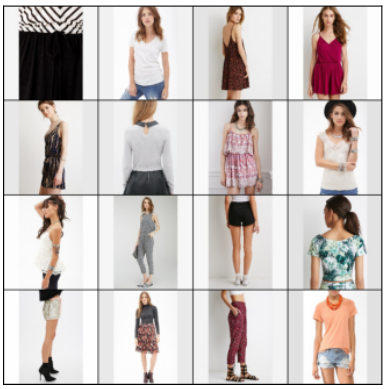


Рис. 3: Примеры из In-Shop

из тестовой выборки ведется поиск ближайших соседей в ней же. При этом первого соседа при таком поиске обычно не учитывают, поскольку он всегда совпадает с запросом.

2.1 Наборы данных

В качестве используемых наборов данных для обучения моделей и тестирования точности используются датасеты малого и среднего размера CUB-200, Stanford Online Product и In-Shop, поскольку именно эти наборы данных чаще всего используются в других аналогичных работах про сравнению моделей. На рис. 1-3 представлены примеры изображений из этих датасетов. Отметим, что все разделения на обучающую выборку и тестирование были предложены авторами наборов данных, так позволяет корректно сравнивать результаты с другими работами. Приведем детальное описание каждого из них.

CUB-200. Прежде всего стоит отметить, что используется версия 2010-го года. Состоит из изображений птиц различных видов. Всего в датасете 6033 сэмплов, каждый из которых принадлежит к одному из 200 классов. Авторы предлагают разбиение на 2 части: train (3000 примеров) и test (3033 примеров).

Stanford Online Product (SOP). Состоит из изображений различных онлайн товаров. Всего в датасете 120053 сэмплов, каждый из которых принадлежит к одному из 59551 классов. При этом первые 11318 классов (59551 изображений) представляют собой обучающую выборку, а другие 11316 классов (60502 изображений) – тестирующую.

In-Shop. Представляет собой подвыборку популярного датасета DeepFashion. Состоит из изображений различных типов одежды на людях. Всего в датасете 52712 изображений, каждое из которых принадлежит к одному из 7982 классов. В отличие от CUB-200 и SOP, для In-Shop авторы предлагают разбиение не на 2 подвыборки, а на 3: train (25882 картинок, 3997 классов), gallery (12612 картинок, 3985 классов), query (14218 картинок, 3985 классов). Подробное описание данного подхода к тестирование см. выше. Множества классов объектов из gallery и query совпадают и не пересекаются с классами train.

2.2 Модели

Как и было сказано во введении, основной целью данной работы является сравнение концептуально разных архитектур нейросетей применительно к задаче обучения метрики. Отметим, что несмотря на отличающиеся подходы к построению аппроксимирующей функции (backbone), на выходе всех моделей должен быть получен вектор-эмбеддинг с признаковым описанием объекта. В данной работе эмбеддинг приводится к размерности 1024. Для тех рассматриваемых в исследовании нейросетей, у которых выходной вектор имел другую размерность, был добавлен дополнительный обучаемый линейный слой, преобразующий эмбеддинг к нужной размерности. Кратко опишем рассматриваемые архитектуры.

Сверточные нейросети. Это наиболее популярный вид нейросетей при работе с изображениями. Ключевыми операциями в таких архитектурах являются свертки (convolutions) и пулинги (poolings). Такие сети прошли довольно большой путь развития: AlexNet[8], VGG[16], ResNet[6], EfficientNet[20]. В современной практике наиболее часто применяются последние две. Можно отметить, что уже продолжительное время семейство EffNet занимается лидирующие позиции на соревновании по классификации картинок ImageNet[2]. В связи с этим, в данной работе мы рассмотрим именно эту нейросеть, а именно её первую и вторую, более новую и эффективную, версии.

Нейросети, основанные на Трансформере. Архитиктура Transformer[23], вышедшая в свет в 2017 году, произвела настоящую революцию в области обработки естественного языка. Изначально изобретенная для решения задачи машинного пере-

вода, она стала[3] основой для практически всех современных моделей глубокого обучения в этой области. Спустя довольно короткое время, были предприняты попытки применения основного строительного блока Трансформера – механизма внимания, для задач компьютерного зрения. Обычно предлагалось использовать их в качестве дополнения к уже существующим сверткам. Подобные эксперименты не приносили существенного успеха до изобретения Vision Transformer (ViT). Эта архитектура полностью отказалась от использования сверток, и практически в исходном виде использовала Трансформер: изображение разбивалось на куски (т.н. патчи, patches), которые служили токенами входной последовательности в Трансформер. Авторы объясняют успех этой модели тем, что современные объемы данных позволяют отказаться от использования в архитектурах т.н. индуктивного смещения, смысл которого был объяснен в введении данной работы. Позже вышла усовершенствованная версия ViT: Shifted Window Transformer (Swin). Проблема исходной модели заключалась в том, что патчи имели одинаковый, и при этом достаточно большой размер, что позволяло хорошо решать задачи, требующие некоторой суммаризации картинки. к примеру, классификации. Однако, в силу такой архитектуры, проигрывала в задачах, в которых было необходимо выделять мелкие детали, к примеру, в детекции. В задаче глубокого обучения метрики некритично выделять небольшие детали изображения, поэтому мы сравним обе эти модели. Более того, можно отметить, что данная работа является одной из первых, в которых Swin Transformer применяется именно для этой задачи.

Полносвязные нейросети. Архитектура ResMLP похожа на ViT в способе представления изображения: оно разбивается на патчи и подается виде последовательности на вход нейросети. Однако внутренние слои этой модели не содержат в себе никаких сверток и механизмой вниманий, присутствуют только полносвязные слои, нормализации, нелинейности и прокидывание связей (skip-connection, которые в сущности не являются слоями). Архитектура не представляет из себя в точности многослойный перспептрон, однако их строительные блоки совпадают. Несмотря на то что данная модель никогда не являлась state-of-the-art для задачи классификации изображений на датасете ImageNet, её важным преимуществом по сравнению с тем же ViT является линейная масштабируемость в зависимости от числа патчей.

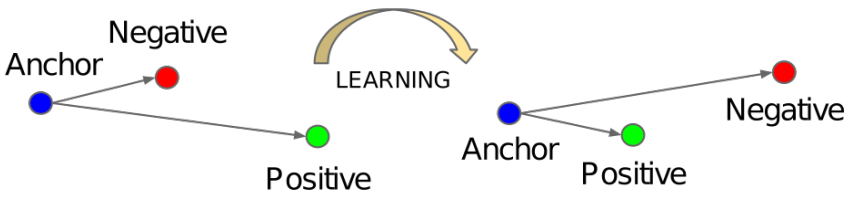


Рис. 4: Иллюстрация процесса обучения Triplet Loss[15]

Ведь, как известно, механизм внимания работает квадратичное относительно входной последовательности время, что создает существенные трудности при измельчении изображения на более мелкие патчи. Также стоит отметить, что данная модель показала близкие к SOTA результатам в тех случаях, когда объем обучающей выборки был очень велик (к примеру, на ImageNet-1k ResMLP существенно проигрывает ViT, а на гигантском датасете JFT-300M[18] – всего на доли процента).

2.3 Функция потерь

В качестве функции потерь (лосса, loss) используется Triplet Loss, который на данный момент является одним из самых популярных на практике. Для подсчета данного лосса необходима тройка объектов: x_a, x_p, x_n. При этом класс x_a (который называется якорным, anchor) должен совпадать с классом x_p (positive sample), но отличаться от класса x_n (negative sample). В этом случае лосс будет считать по следующей формуле:

$$\mathcal{L}(x_a, x_p, x_n) = \max\left[0, \mathcal{D}_{f_\theta}(x_a, x_p) - \mathcal{D}_{f_\theta}(x_a, x_n) + \alpha\right]$$

Здесь $\mathcal{D}$ – некоторая функция расстрояния. В данной работе рассматривается стандартное евклидово расстояние. $\mathcal{D}_{f_\theta}(x, y)$ означает, что расстояние берется между образами объектов x и y после преобразования в вектора признаков (embeddings, эмбеддинги) нейросетью f, в свою очередь параметризованную оптимизируемыми параметрами θ. Гиперпараметр α называется отступом (margin), в данной работе используется $\alpha = 0.02$. В сущности, это формула означает, что объекты одного класса должны быть расположены друг к другу ближе, чем к объектам другого класса. При этом, если разница расстояний становится больше, чем α, то модель за это не

штрафуется. Это сделано для того, чтобы на поверхности лосса был выраженный минимум, поскольку в ином случае можно будет оптимизировать функцию потерь до бесконечности, лишь отдаляя некоторые два класса друг от друга на неограниченные расстояния. Таким образом, этот подход можно считать регуляризацией, повышающей обобщающую способность модели. На рис. 4 изображена идея обучения модели, использующей Triplet Loss.

На практике сэмплирование таких троек объектов, на которых лосс не будет равняться нулю, является достаточно сложной задачей (которая еще и усложняется по мере обучения модели). В данной работе был применен следующий подход: с помощью специального сэмплера, минибатч составляется не случайным образом, а так, чтобы в нем было по 4 примера из 4 случайных классов. Все эти объекты пропускаются через нейросеть для получения эмбеддингов. Далее, из данного множества векторов перебираются всевозможные подходящие тройки объектов x_a, x_p, x_n. С помощью функции-майнера (miner) выделяются только те тройки, на которых лосс не был бы равен нулю. Эти лоссы усредняются, и по полученному значению запускается алгоритм обратного распространения ошибки для получения градиентов параметров модели и их обновления. Ключевым значением функции-майнера является тот факт, что её использование позволяет существенно ускорить обучения модели, поскольку градиенты не будут подсчитываться для тех троек, для которых значение лосса равно нулю.

2.4 Предварительная обработка данных

Перед непосредственной подачей в нейросеть, изображения проходят ряд преобразований. Прежде всего, все картинки приводятся к разрешению 224x224, поскольку некоторые из моделей, используемых в исследовании, предобучены именно на такой размер. Далее, если изображение является объектом обучающей выборки, к ней применяется ряд аугментаций, то есть операций, немного изменяющих изображение, но не меняющих её метку класса. Конкретный их набор был заимствован из решения победителей соревнования Google Landmark Retrieval[19], которое также было посвящено глубокому обучению метрики: отражения по вертикали, небольшое сжатие изображение, повороты и растяжения, а также маскирование случайных квадрат-

ных частей изображения. Каждое из этих преобразований может быть применено случайным образом применено с некоторой вероятностью. В конце все изображения проходят нормализацию, т.е. вычитание среднего значения и деление на стандартное отклонение, с соответствующими параметрами, посчитанными на датасете ImageNet. Это необходимо для того, чтобы распределение входных изображений совпадало с тем распределением, на котором были предобучены исследуемые модели.

2.5 Обучение модели

Обучение проводилось с помощью оптимизатора AdamW[12]. Данная версия представляет с собой обычный Adam[7], но с исправленной регуляризацией параметров модели (weight decay, WD). В проведенных экспериментах WD был положен равен 10^{-2}. Темп обучения (learning rate) изменялся по линейному закону от начального значения 10^{-5} до 10^{-6} в течение всего процесса оптимизации. Обучение проводилось в течение 500 эпох, либо до достижения плато, т.е. до того момента, когда точность на тестовой выборке переставала существенно меняться. Данный выбор параметров показал наиболее высокие результаты в совокупности для всех моделей. Безусловно, ту или иную модель можно обучить немного лучше, произведя ещё более точную настройку гиперпараметров.

2.6 Оценка точности

Для оценки качества моделей при решении задачи metric learning часто используется метрика Mean Average Precision at k (maP@k, на русском языке практически не встречается). Её определение вводится постепенно. Для начала, определим Precision@k (P@k) в задаче поиска. Стоит отметить, что это определение отличается от понятия точности при решении задачи классификации. Предположим, что алгоритм поиск выдал k объектов, являющихся кандидатами на наиболее похожие объекты к запросу q. Скажем, что, на самом деле, среди них имеется лишь m релевантных (похожих, подходящих) запросу. В этом случае P@k будет равно $\frac{m}{k}$. Заметим, что такая метрика никак не отражает порядок, в котором были выданы кандидаты. Хотелось бы, чтобы метрика отражала, насколько правильно модель расставила кандидатов: более релевантные объекты должны предлагаться раньше, чем менее реле-

вантные. Немного модифицировав метрику для решения этой проблемы, мы придем к определению Average Precision @ k (AP@k):

$$\text{AP}_q\text{@k} = \frac{1}{N_q(k)} \sum_{i=1}^{k} P_q@k * rel_q(k),$$

где значение $rel_q(k)$ равно единице, в случае, если k-й выданный объект релевантен запросу q, иначе 0, а $N_q(k) = \min(k, M_q)$, где M_q – общее число объектов в индексе, релевантных запросу q. В контексте этой работы релевантность означает, что объект-кандидат принадлежит тому же классу, что и запрос. Наконец, легко обобщить AP@k до mAP@k: для этого достаточно усреднить AP@k по всем запросам q_j в выборке:

$$\text{mAP@k} = \frac{1}{M} \sum_{j=1}^{M} \frac{1}{N_{q_j}(k)} \sum_{i=1}^{k} P_{q_j}@k * rel_{q_j}(k),$$

В данной работе для отслеживания точности модели использовались метрики mAP@1 и mAP@5. Легко заметить, что mAP@1 на самом деле представляет из себя точность классификации алгоритма KNN при K=1. Также отметим, что, как уже было указано выше, в случае, когда тестирующие выборка-индекс и выборка-запрос совпадают, первый ближайший сосед опускается.

2.7 Технические детали

Эксперименты проводились на языке программирования Python, с использованием в качестве основы библиотеки PyTorch. Все реализации архитектур и предобученные веса были получены из библиотеки Pytorch Image Models (timm)[25]. Часть реализаций методов, специфичных для обучения метрики, были получены из библиотеки Pytorch Metric Learning[14]. Для аугментаций использовалась библиотека Albumentations[1]. Весь код для воспроизведения экспериментов, а также логи доступны в GitHub репозитории[1]. Вычисления проводились на видеокарте NVIDIA GeForce RTX 3090 (24 Гб видеопамяти) приблизительно в течение недели.

Датасет	CUB-200 (K)		SOP (K)		In-Shop (K)	
Модель	1	5	1	5	1	5
EffNet V1	0.61	0.66	0.75	0.77	0.57	0.63
EffNet V2	0.69	0.73	0.77	0.79	0.79	0.81
ViT	?	?	0.85	0.86	0.90	0.90
Swin	**0.90**	**0.91**	**0.87**	**0.89**	**0.91**	**0.91**
ResMLP	0.73	0.76	0.81	0.82	0.86	0.87

Таблица 1: Результаты экспериментов. Значения метрики mAP@k для различных значений k, моделей и датасетов.

Модель	Точная версия (из timm)	Число параметров	Пропускная способность (картинок в сек.)
EffNet V1	tf_efficientnet_b5	28.340.784	282
EffNet V2	tf_efficientnetv2_m	54.170.100	253
ViT	vit_base_patch8_224	86.595.328	72
Swin	swin_base_patch4_window7_224	87.792.824	175
ResMLP	resmlp_36_distilled_224	44.699.728	332

Таблица 2: Информация о моделях

3 Результаты экспериментов

Результаты экспериментов представлены в таблице 1. Можно заметить, что с существенным отрывом побеждают модели, основанные на Трансформере: ViT и Swin. В сравнении с ними, привычные сверточные сети демонстрируют достаточно слабые результаты. Можно предположить, что это связано с тем, что более современные модели предобучаются на большем количестве данных, к примеру, на датасете ImageNet-21k. В таблице 2 представлены более точные спецификации моделей, а также дополнительная информация о них: число параметров и пропускная способность, то есть количество картинок размера 224x224 пикселя, которые модель может обработать за одну секунду в режиме инференса.

[1]https://github.com/Sorrow321/cmc_seminar_master

Можно сделать вывод, что на данный момент наилучшей архитектурой для решения задачи поиска изображений является семейство моделей, основанных на трансформере. Более того, несмотря на то что ViT и Swin демонстрируют достаточно близкие с точки зрения точности результаты, Swin обладает большей пропускной способностью. Это связано с тем, что ViT используется "глобальный"механизм внимания: все входные токены взаимодействуют друг с другом, что ведет к квадратичной вычислительной сложности в зависимости от числа токенов. Однако Swin использует механизм внимания лишь внутри ограниченного окна фиксированного размера, что позволяет перейти от квадратичной сложности масштабирования к линейной.

4 Заключение

Проведено сравнение наиболее современных нейросетевых архитектур компьютерного зрения в задаче поиска изображений. В результате было установлено, что наилучшие результаты на данный момент демонстрируют модели, основанные на архитектуре Трансформера, при этом лучшей с точки зрения баланса точности и скорости работы оказалась модель Shifted Window (Swin) Transformer. Она показала лучшие показатели mAP@k, при этом работая значительно быстрее, чем Vision Transformer (ViT). Было установлено, что доминировавшие в области компьютерного зрения сверточные нейросети демонстрируют существенно более низкие результаты, чем описанные выше модели, обладая при этом сравнимым количеством параметров. Архитектура ResMLP, представляющая собой улучшенную версию многослойного перспептрона, показала средние результаты между сверточными моделями и трансформерами.

Дополнить данное сравнение можно было бы более новой версией Swin трансформера: Swin V2[9], превосходящий по производительности оригинальную модель. Однако, к сожалению, на момент проведения исследования авторами данной архитектуры не были представлены в общий доступ обученные веса, а самостоятельное предобучение этой модели на датасете ImageNet-21k требует колоссальных вычислительных ресурсов.

Список литературы

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[5] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval, 2021.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[9] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. 2021.

[10] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[11] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.

[13] Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2016.

[14] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. Pytorch metric learning, 2020.

[15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. 2015.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[17] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era, 2017.

[19] B. Cao T. Weyand, A. Araujo and J. Sim. Google landmarks dataset v2 - a large-scale benchmark for instance-level recognition and retrieval, 2020.

[20] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019.

[21] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. 2021.

[22] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Resmlp: Feedforward networks for image classification with data-efficient training, 2021.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[24] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 09 2010.

[25] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.