Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Панкина Алина Алексеевна

Исследование методов обучения представлению данных

Курсовая работа

Научный руководитель:
к.ф.-м.н.
Буряк Дмитрий Юрьевич

Москва, 2024

Аннотация

Исследование методов обучения представлению данных

Панкина Алина Алексеевна

С каждым днем растёт потребность в регулярном использовании различных методов машинного обучения для решения практических задач. В следствие чего зачастую возникают проблемы при обработке больших объемов данных высоких размеров, связанные не только с большой вычислительной нагрузкой, но также и с невысоким качеством работы алгоритмов.

Для предотвращения данных проблем необходимо уметь понижать размерность данных, сохраняя их информативность. Для этой задачи используются методы обучения представлению данных. Работа посвящена разбору и анализу существующих методов.

Содержание

1 Введение

В настоящее время нейронные сети и другие алгоритмы машинного обучения используются повсеместно для огромного спектра различных задач, таких как распознавание образов и изображений, анализ текста и естественного языка, прогнозирование временных рядов, рекомендательные системы и т.д. С резким ростом размера доступных наборов данных как с точки зрения количества выборок, так и количества признаков в каждой выборке, важной задачей становится уменьшение размерности данных и сохранение числа признаков как можно более низким. Данная задача в машинном обучении называется "Обучение представлению данных".

Зачастую, если данные содержат большое количество признаков, часть этих признаков являются шумовыми, то есть не содержат в себе полезной информации, а часть избыточными, то есть коррелируют с другими признаками и дублируют уже известную информацию. Если не избавляться от таких признаков, можно столкнуться со следующим рядом проблем:

- переобучение;
- плохая интерпретируемость модели;
- невозможность визуализация данных;
- большая нагрузка на вычислительные ресурсы.

Обучение представлению данных является действительно актуальной и важной задачей. Оно позволяет не только ускорить процесс обучения, но и повысить точность работы алгоритмов. Методы обучения представлению данных активно используются в различных прикладных сферах машинного обучения. Одной из распространенных сфер является задача обработки изображений. Для выделения признаков из изображений методы как правило обучаются и тестируются на наборе ImageNet [1].

Рассмотрим насколько полезным может быть обучение представлению данных на примере задачи классификации изображений из данного набора. Он состоит из изображений разного размера, поэтому перед использованием их необходимо привести к единому формату. Популярные сверточные сети, используемые для решения данной задачи, такие как VGG [2], ResNet [3], GoogleNet [4] и т.д., принимают на вход предобработанные изображения размером 224×224. То есть обрабатывают около 50.000

признаков. Современные методы обучения представлению данных позволяют сократить такой объем до вектора размерностью 2048 [5]. Таким образом количество признаков уменьшается почти в 25 раз.

Благодаря тому, что размер полученного представления в разы меньше, чем размер исходных данных, мы можем использовать более простой линейный классификатор для решения нашей задачи. Обучение будет производиться гораздо быстрее, при этом качество его работы останется на том же уровне.

Методы обучения представлению данных можно разделить на две основные группы: методы, основанные на отборе признаков, и методы, основанные на извлечении признаков. Методы отбора признаков уменьшают размерность, выбирая подмножество признаков из исходного набора. Методы извлечения признаков преобразуют существующие признаки в новое пространство на основе линейных и нелинейных комбинаций из исходного набора.

Несмотря на то, что методы отбора признаков могут быть полезными, они имеют большой недостаток по сравнению с методами извлечения признаков: они не учитывают взаимодействия между признаками внутри выборки, что может привести к упущению значимых паттернов в данных. В целом, извлечение признаков обычно предпочтительнее отбора признаков, так как позволяет более гибко и эффективно использовать информацию из исходных данных.

Целью данной работы является исследование методов извлечения признаков. Для генерации нелинейных комбинаций они используют глубокие нейронные сети.

2 Обзор существующих подходов

Среди методов извлечения признаков можно выделить три основных подхода.

2.1 Методы, основанные на обучении без учителя

Данная группа методов была предложена совсем недавно. Они не требует размеченных данных для обучения, при этом показывают высокое качество работы. Именно поэтому они имеют большое преимущество по сравнению с многими методами, которые использовались ранее для задачи извлечения признаков.

Общая идея данного подхода заключается в выполнении следующих двух задач:

1. Сохранить инвариантность для всех положительных пар набора.

2. Получать представления с декоррелированным пространством признаков.

Понятие "положительная пара" имеет разное определение в зависимости от контекста задачи. Если речь идет про обучение с учителем, то положительной парой будем называть представления, полученные из разных объектов одного класса. Если же речь идет про обучение без учителя, как в нашем случае, то положительными парами будем называть два разных представления, полученных из одного и того же объекта путем аугментаций и прочих преобразований.

Представления - это и есть наши целевые векторы, которые содержат извлеченные признаки.

Выполнение первой задачи необходимо, чтобы модель не извлекала шумовые признаки, и чтобы таким образом представления получались как можно более информативными.

Выполнение второй задачи тоже помогает повысить информативность представлений, поскольку декорреляция уменьшает избыточность признакового пространства. Однако это не единственная причина, по которой требуется выполнение данной задачи. Дело в том, что если выполнять только первую задачу инвариантности, то можно столкнуться с проблемой коллапса. Эта проблема заключается в том, что модель начинает игнорировать входные данные и создает идентичные и постоянные выходы для всех объектов. Другими словами, она находит тривиальное решение для выполнения инвариантности положительных пар, делая инвариантными все пары объектов. Однако если

выполняется декорреляция признаков, то модель уже не может генерировать константные выходы. Таким образом с помощью декорреляции мы решаем сразу две подзадачи: уменьшаем избыточность признакового пространства и предотвращаем проблему коллапса.

Все методы данной группы очень похожи, но немного отличаются в реализации основной идеи. Например, в 2021 году был предложен метод Whitening-MSE [6]. Его особенностью является то, что для деккореляции признакового пространства он использует дополнительный модуль, который преобразует представления в собственное пространство их ковариационной матрицы, и векторы, полученные таким образом, равномерно распределяются на единичную сферу. Однако такой подход влечет за собой громоздкие матричные вычисления, что является очень дорогостоящим использованием вычислительных ресурсов. Чуть позже в 2021 был предложен метод Barlow Twins [7], который предлагает более простую реализацию. С помощью функции потерь он нормализует матрицу кросс-корреляции, в которой каждый элемент представляет собой коэффициент корреляции между выходами двух ветвей. Нормализация такой матрицы позволяет решить сразу две вышеописанные задачи. В январе 2022 года был предложен метод VICReg [5]. Он заимствует механизм декорреляции у метода Barlow Twins. Однако вместо матрицы кросс-корреляции он использует матрицу ковариации, которая составляется для каждого из выходов ветвей. Данную матрицу он также нормализует с помощью функции потерь, и кроме этого использует дополнительные компоненты дисперсии и инвариантности.

2.2 Контрастное обучение

Контрастное обучение по своей структуре схоже с методами, основанными на обучении без учителя, однако вместо декорреляции использует другой механизм. Общую идею также можно разбить на две задачи:

1. Сохранить инвариантность для всех положительных пар набора, минимизируя расстояние между ними.

2. Максимизировать расстояние для всех отрицательных пар набора.

Отрицательной парой, по аналогии с положительной, будем называть представления, полученные из объектов разных классов, если речь идет про обучение с учителем.

Или же представления, полученные из разных объектов, если речь идет про обучение без учителя.

Таким образом, в полученном пространстве представлений положительные пары будут располагаться близко, а отрицательные далеко. Данная идея позволяет получать информативные представления, избегая попадания шума, а также не позволяет сети "сломаться"и создавать идентичные представления для всех объектов, тем самым избегая проблемы коллапса.

Методы контрастного обучения дают хорошие результаты на практике. Однако для обучения им требуется большое количество контрастных пар. Как правило, данный подход подразумевает обучение с учителем. То есть эти пары задаются непосредственно в выборке, как например в методе SimCLR [8], предложенном в 2020 году. Но такой подход очень ресурсозатратный, поэтому существуют некоторые более сложные методы, которые способны обучаться без размеченных данных. Например метод MoCo [9], предложенный ранее в 2020 году, как и методы из предыдущего раздела, использует аугментации для преобразований исходных объектов, а также банк памяти для хранения представлений всех объектов. За счет чего сам генерирует положительные и отрицательные пары.

2.3 Обучение с учителем

Стоит также рассказать про более примитивный подход, с которым в дальнейшем будут сравниваться рассмотренные выше методы. Несмотря на простую идею, он все же является довольно распространенным и используется на практике.

Алгоритм работы данного методы можно описать следующим образом:

1. Обучить на большой базе объектов нейронную сеть для задачи классификации.

2. Использовать обученную сеть без последнего слоя классификации для выделения признаков.

Дело в том, что в процессе обучения нейронная сеть итак учится извлекать сложные и дискриминативные признаки, которые будут полезными для решения задачи. Поэтому представления, полученные из предпоследнего слоя, будут довольно информативными.

В статьях [10] и [11] описаны примеры использования данного метода для задачи выделения ключевых слов.

Конечно, предыдущие подходы, которые используют дополнительные вычисления для повышения информативности представлений, зачастую дают лучшие результаты на практике. Однако данный метод использует гораздо меньше вычислений при обучении.

3 Обзор методов извлечения признаков на основе обучения без учителя

В разделе 2 приведено описание данных методов. В этой главе рассмотрим детали трех перечисленных подходов.

3.1 Barlow Twins

Начнем обзор с метода Barlow Twins, который был предложен в июне 2021 года. Его архитектура представлена ниже на рисунке 1. Он достаточно нагляден и предлагает довольно эффективную реализацию вышеописанных задач.

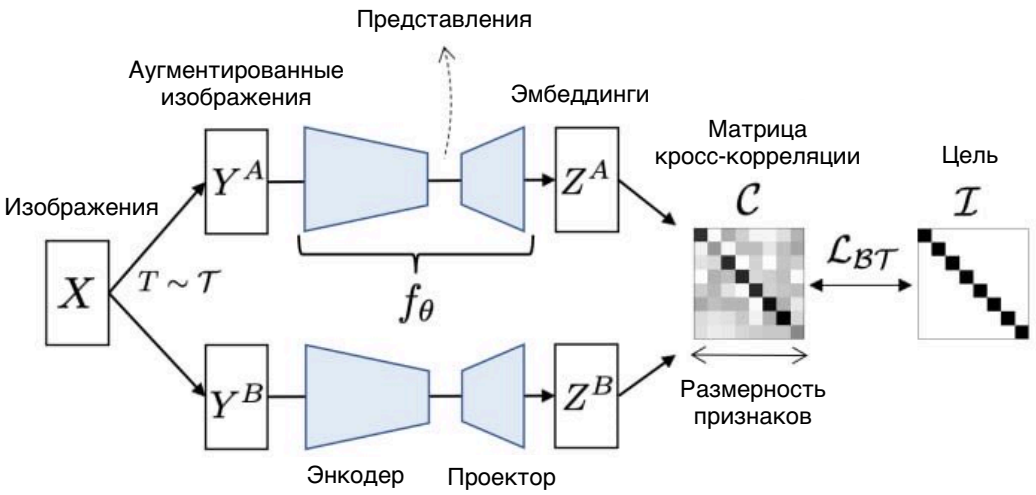


Рис. 1: Архтектура метода Barlow Twins

Для простоты понимания алгоритм работы метода можно разбить на три этапа.

3.1.1 Этап 1

На вход поступает выборка объектов X. Объекты могут быть любые, в зависимости от конкретной задачи: тексты, геоданные, биологические данные. В данной работе многие этапы будут рассматриваться на примере изображений в качестве входных данных.

Для каждого объекта из исходной выборки создаются две разные аугментации, таким образом выборка подразделяется на две выборки Y^A и Y^B, каждая из которых содержит аугментированные объекты из исходного набора.

Для изображений в качестве аугментаций как правило используются следующие преобразования:

- Применяются всегда:
    1. Случайное кадрирование (random cropping);
    2. Изменение размера на 224×224 (resizing).
- Применяются с некоторыми вероятностями (одинаковыми для двух аугментаций):
    1. Зеркальное отображение по горизонтали (horizontal flipping);
    2. Color jittering;
    3. Преобразование в черно-белый формат (converting to grayscale).
- Применяются с некоторыми вероятностями (различными для двух аугментаций):
    1. Размытие по Гауссу (Gaussian blurring);
    2. Соляризация (solarization).

Они используются конкретно в методе Barlow Twins и чаще всего в других методах тоже. Но стоит сказать, что для других методов аугментации могут и немного различаться. Например, могут использоваться не все из последних пяти перечисленных, или могут использоваться дополнительно и другие аугментации, такие как аффинные преобразования, фильтрация Собеля и тд.

3.1.2 Этап 2

Далее каждая из выборок подается на вход нейронной сети, обозначенной как f_θ. Она состоит из двух частей: энкодера и проектора. Для преобразования изображений в качестве энкодера обычно используются архитектура сети ResNet-50 [3] без последнего слоя классификации (2048 нейронов на выходе). Далее следует нейронная сеть проектора. В Barlow Twins она состоит из трех полносвязных слоев, каждый из которых имеет выход 8192 нейрона. После первого и второго слоя проектора идет слой Batch Normalization.

На выходе из энкодера мы получаем векторы, которые как раз и называются представлениями. А векторы, полученные на выходе из проектора, будем называть эмбедингами. Здесь важно отметить: разница в том, что представление - это и есть наш целевой вектор, который далее мы будем использовать для практических задач, а эмбединги мы передаем в функцию потерь, то есть используем их исключительно для обучения нейронной сети. В данном методе проектор используется для того, чтобы немного расширить признаковое пространство и далее работать с большей размерностью.

3.1.3 Этап 3

Итак, мы получили две выборки, состоящие из эмбедингов, обозначенные как Z^A и Z^B. Мы составляем для них матрицу кросс-корреляции, элементы которой считаются по следующей формуле:

$$c_{ij} = \frac{\sum_b z_{ib}^A z_{bj}^B}{\sqrt{\sum_b (z_{ib}^A)^2} \sqrt{\sum_b (z_{bj}^B)^2}}$$

здесь $b = \overline{1, N}$, где N - количество элементов в исходной выборке, $i, j = \overline{1, M}$, где M - размерность эмбедингов в выборках Z^A, Z^B.

Далее мы применяем функцию потерь, которая выглядит следующим образом:

$$L_{BT} = \sum_i (1 - c_{ii})^2 + \lambda \sum_i \sum_{j \neq i} c_{ij}^2$$

где λ - положительный гиперпараметр, введенный для регуляризации второго слагаемого функции.

Как можем видеть, цель данной функции - приблизить нашу матрицу кросс-корреляции к единичной матрице. Минимизируя первое слагаемое, мы устремляем диагональные элементы матрицы к единице. В результате чего все положительные пары эмбедингов должны коррелировать друг с другом, что обеспечивает их инвариантность. Таким образом выполняется первая задача, описанная в разделе 2.1. Минимизируя второе слагаемое функции, мы устремляем недиагональные элементы матрицы к нулю, тем самым декоррелируя пространство признаков внутри каждого эмбединга. И теперь уже выполнена и вторая задача.

Как видим, Barlow Twins эффективно решает две задачи, совмещая их в функции потерь. В последующих разделах все рассмотренные методы будут оцениваться на

практических задачах. Соответственно, данный метод, как и другие нижеописанные, еще будет упоминаться, а также сравниваться друг с другом.

3.2 Whitening MSE

Метод Whitening-MSE был предложен чуть раньше метода Barlow Twins, в мае 2021 года. Он использует другие, но не менее интересные подходы для реализации двух описанных задач.

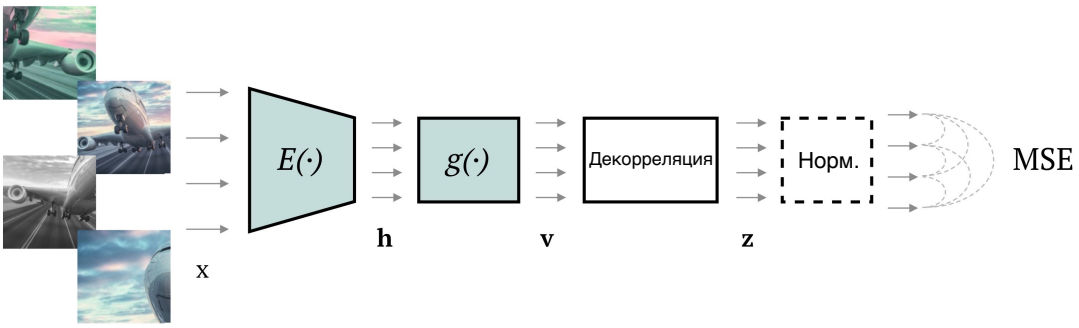


Рис. 2: Архитектура метода Whitening-MSE

Первый этап данного метода аналогичен методу Barlow Twins. Разница заключается только в том, что W-MSE предполагает любое количество аугментаций для каждого объекта выборки, тем самым увеличивая количество положительных пар в наборе.

Второй этап тоже аналогичен. Различаются только архитектуры проекторов. Проектор W-MSE состоит всего из одного полносвязного слоя, который имеет 1024 нейрона на выходе. За ним следует слой Batch Normalization. Здесь наоборот необходимо сузить размерность признакового пространства для успешного выполнения последующих преобразований.

Далее, в отличие от метода Barlow Twins, данный метод имеет дополнительный этап, который в англоязычных источниках называется Whitening. Мы для упрощения будем называть его декорреляцией.

Итак, на выходе из проектора g(x) мы получаем выборку из $N \times d$ эмбедингов, где N - количество объектов в исходной выборке, d - количество аугментаций для каждого объекта. Пусть $K = N \times d$. Тогда для каждого из эмбедингов v мы применяем следующее

преобразование:

$$Whitening(v) = W_v(v - \mu_v)$$

Здесь:

- μ_v - вектор, рассчитанный как среднее значение по выборке: $\mu_v = \frac{1}{k}\sum_k v_k$;
- W_v - такая матрица, что: $W_v W_v^T = \Sigma_v^{-1}$, где Σ_v- матрица ковариации:

$$\Sigma_v = \frac{1}{K-1}\sum_k (v_k - \mu_k)(v_k - \mu_k)^T.$$

Для вычисления матрицы W_v используется разложение Холецкого. Данное разложение позволяет посчитать для любой симметричной положительно определенной матрицы A матрицу L, такую что $A = LL^T$, где L - нижняя треугольная матрица со строго положительными элементами на диагонали. Можно также записать разложение в эквивалентной форме: $A = U^T U$, где U - верхняя треугольная матрица со строго положительными элементами на диагонали.

Разложение Холецкого всегда существует и единственно для любой симметричной положительно определённой матрицы. Наша матрица Σ_v удовлетворяет этим критериям, поэтому мы всегда сможем найти матрицу W_v.

В результате данных преобразований мы получаем новую выборку $Z = \{z_1, \ldots, z_K\}$, $z_i = Whitening(v_i)$, $i = \overline{1, K}$, в которой декоррелировано пространство признаков внутри каждого эмбединга. Таким образом мы выполнили задачу декорреляции.

Далее, переходя к последнему этапу, мы передаем выборку Z в функцию потерь, которая выглядит следующим образом:

$$L_{W\_MSE} = \frac{2}{Nd(d-1)}\sum dist(z_i, z_j)$$

где $dist(z_i, z_j)$ - расстояние между положительными парами z_i и z_j:

$$dist(z_i, z_j) = \left\| \frac{z_i}{\|z_i\|_2} - \frac{z_j}{\|z_j\|_2} \right\|_2^2$$

N и d - как уже говорилось ранее, количество объектов в исходной выборке и количество аугментаций, тогда:

- $\frac{d(d-1)}{2}$ - количество положительных пар для каждого объекта;

- $\frac{Nd(d-1)}{2}$ - общее количество положительных пар в наборе.

С помощью данной функции потерь мы минимизируем среднее значения расстояния между полученными эмбедингами для всех положительных пар набора. То есть мы требуем, чтобы положительные представления находились близко друг к другу в общем пространстве представлений. Данное условие выполняет первую задачу инвариантности.

3.3 VICReg

Метод VICReg (Variance-Invariance-Covariance Regularization) очень похож на метод Barlow Twins. Он заимствует оттуда механизм декорреляции, однако использует более усовершенствованную функции потерь и другой подход для сохранения инвариантности.

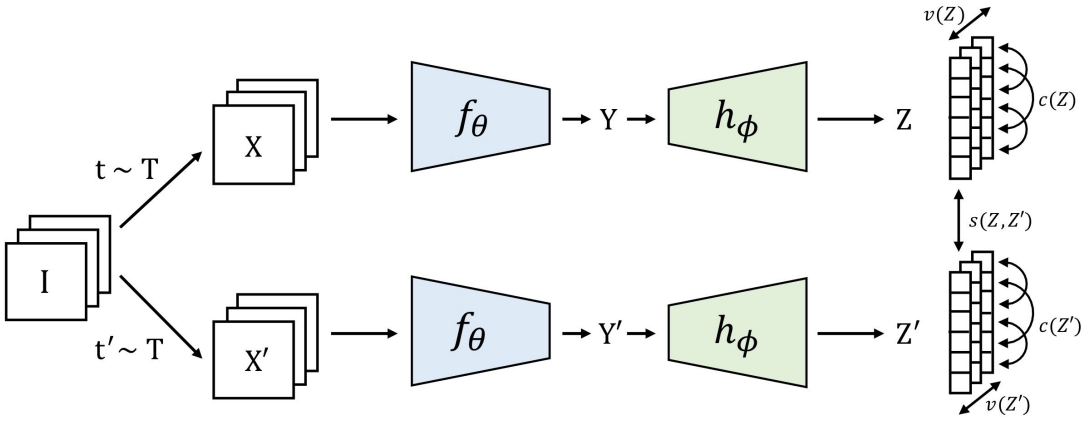


Рис. 3: Архтектура метода VICReg

Первые да этапа полностью аналогичны методу Barlow Twins, архитектура проектора полностью совпадает. Поэтому перейдем сразу к последнему этапу и рассмотрим функцию потерь.

Итак, мы получили две выборки эмбедингов Z и Z'. Для них мы рассчитываем следующую функцию потерь:

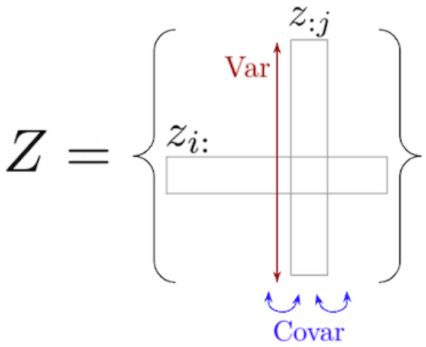
$$L_{VICReg} = \sum_{i \in D} \sum_{t' \in T} \left[\lambda s(Z, Z') + \mu \{ v(Z) + v(Z') \} + \nu \{ c(Z) + c(Z') \} \right]$$

здесь D - исходная выборка, T - примененные аугментации, λ, μ, ν - положительные гиперпараметры, введенные для регуляризации каждого из слагаемых функции.

Данная функция является композицией трех функций:

- $v(Z)$ - дисперсия;
- $s(Z, Z')$ - инвариантность;
- $c(Z)$ - ковариация.

Разберем каждую функцию по отдельности. Пусть N - количество объектов в исходной выборке, d - размерность полученных эмбедингов в выборках Z, Z'. Также для удобства введем матрицу Z, которая составляется для каждой из выборок:



В строках данной матрицы содержатся эмбединги z_i, $i = \overline{1, N}$, а в столбцах признаки z_j, $i = \overline{1, d}$. Таким образом, размерность матрицы: $N \times d$.

Начнем с дисперсии. Введем следующую функцию:

$$Var(z_j) = \frac{1}{n-1} \sum_{i=1}^{N} (z_{ij} - \overline{z_j})^2$$

здесь $\overline{z_j} = \frac{1}{n} \sum_{i=1}^{N} z_{ij}$ - среднее значение вектора z_j, $j = \overline{1, d}$, которое рассчитывается для каждого признака. Как видим, функция $Var(z_j)$ считает для каждого признака значение дисперсии.

Тогда функция $v(Z)$ выглядит следующим образом:

$$v(Z) = \frac{1}{d} \sum_{j=1}^{d} max\left(0, \gamma - \sqrt{Var(z_j) + \epsilon}\right)$$

Цель данной функции заключается в том, чтобы сделать среднеквадратичное отклонение для каждого признака выборки превышающим значение некоторого гиперпараметра γ. Функция $v(Z)$ используется исключительно для дополнительного предотвращения проблемы коллапса, поскольку порог среднеквадратичного отклонения не позволит модели генерировать константные выходы.

Функция инвариантности $s(Z, Z')$ выглядит следующим образом:

$$s(Z, Z') = \frac{1}{n} \sum_i \|z_i - z'_i\|_2^2$$

Цель данной функции, как и в W-MSE, сохранить инвариантность представлений, за счет минимизации расстояния между положительными парами. Здесь используется более простая функция: евклидова метрика. Таким образом мы решаем первую задачу.

Рассмотрим теперь функцию ковариации. Для этого введем матрицу ковариции:

$$C(Z) = \frac{1}{n-1} \sum_{i=1}^{N} (z_i - \overline{z})(z_i - \overline{z})^T$$

где $\overline{z} = \frac{1}{n} \sum_{i=1}^{N} z_i$ - среднее значение вектора по выборке.

Тогда функция $c(Z)$ имеет вид:

$$c(Z) = \frac{1}{d} \sum_{l \neq m} C(Z)_{lm}^2$$

Цель функции $c(Z)$ - приблизить все недиагональные элементы матрицы ковариции $C(Z)$ к нулю. С помощью этого мы декоррелируем признаковое пространство внутри эмбедингов, тем самым выполняя вторую задачу.

3.4 Оценка и сравнение рассмотренных методов

Оценим результаты работы методов на задаче классификации изображений из набора ImageNet. Для оценивания использовалось два способа: линейная оценка, оценка частичного обучения.

Линейная оценка проводится следующим образом:

1. Обучаем модель для задачи извлечения признаков, назовем ее "базовая сеть";
2. Дописываем линейный классификатор, принимающий на вход представления из базовой сети;

3. Замораживаем базовую сеть;

4. Дообучаем только линейный классификатор на полном наборе размеченных данных.

Оценка частичного обучения проводится следующим образом:

1. Обучаем модель для задачи извлечения признаков;

2. Дописываем линейный классификатор, принимающий на вход представления из базовой сети;

3. Дообучаем базовую сеть и линейный классификатор на неполном наборе размеченных данных.

В таблице представлена точность Top-1 и Top-5. А также процент размеченных данных, используемых для частичного обучения.

Метод	Линейная оценка		Частичное обучение			
	Top-1	Top-5	Top-1		Top-5	
			1%	10%	1%	10%
Обучение с учителем	76.5	-	25.4	56.4	48.4	80.4
MoCo [9]	60.6	-	-	-	-	-
SimCLR [8]	69.3	89.0	48.3	65.6	75.5	87.8
MoCo V2 [12]	71.1	90.1	-	-	-	-
W-MSE 4 [6]	69.4	-	-	-	-	-
Barlow Twins [7]	73.2	91.0	55.0	69.7	79.2	89.3
VICReg [5]	73.2	91.1	54.8	69.5	79.4	89.5

Для всех методов в таблице использовалась архитектура ResNet-50 [3] в качестве энкодера. Обучение базовой сети проводилось на 1000 эпохах. Для линейной оценки дообучение проводилась на 100 эпохах, для оценки частичного обучения - на 20 эпохах. Для обоих оценок размер пакета составлял 256.

В таблице подчеркнуты лучшие результаты. Как можем видеть, лидируют методы на основе обучения без учителя, в частности Barlow Twins [7] и VICReg [5].

4 Постановка задачи

В рамках задачи будем исследовать метод VICReg.

Рассмотрим два непересекающихся множества исходных данных. Обозначим их A и B. Оба множества являются размеченными. Обозначим метки данных множеств как M_A и M_B для A и B соответственно. Множества меток M_A и M_B также являются непересекающимися.

Разделим каждое из множеств A и B на два размеченных набора - тренировочный и тестовый. Обозначим данные наборы для множества A: a_1 и a_2, для множества B: b_1 и b_2, как тренировочный и тестовый набор соответственно.

Также введем множество неразмеченных данных a_1^n, которое будет представлять собой тренировочный набор множества A без меток: $a_1 \setminus M_A$.

Множество a_1^n будем использовать для обучения энкодера с помощью метода VICReg.

Задача будет тестироваться для обработки изображений, поэтому в качестве архитектуры энкодера будем использовать сверточную нейронную сеть.

Цель заключается в том, чтобы проанализировать качество обучения энкодера с помощью метода VICReg. Для этого предлагается проверить точность классификации эмбедингов, полученных на выходе из энкодера после обучения.

Опишем подробнее шаги, которые необходимо выполнить для данной задачи.

1. Выбрать сверточную сеть для архитектуры энкодера.
2. Обучить энкодер на множестве a_1^n.
3. Провести тестирование на множествах A и B:
   (a) С помощью обученного энкодера получить эбединги для наборов a_1 и b_1. Множество полученных эмбедингов обозначим α_1 и β_1 для каждого набора соответственно.
   (b) Обучить однослойный линейный классификатор на множествах α_1 и β_1.
4. Оценить точность обученного классификатора на наборах a_2 и b_2:
   (a) Для оценки точности классификации будем использовать метрику Accuracy и проверять точность для Top-1 и Top-5 предсказаний.

5. В конце сравним:

   (a) Насколько различаются точности классификации набора a_2, если:

   - обучать энкодер на множестве a_1 с помощью обучения с учителем;
   - обучать энкодер на множестве a_1^n с помощью метода VICReg.

   (b) Насколько различаются точности классификации наборов a_2 и b_2.

5 Предложенный подход

5.1 Архитектура энкодера

Было рассмотрено две разных архитектуры энкодера.

В качестве первого варианта будем использовать собственную архитектуру, состоящую из пяти сверточных слоев. Она обрабатывает 403.530 параметров. Обозначим данную архитектуру N_1. Она изображена подробно на рисунке ниже.

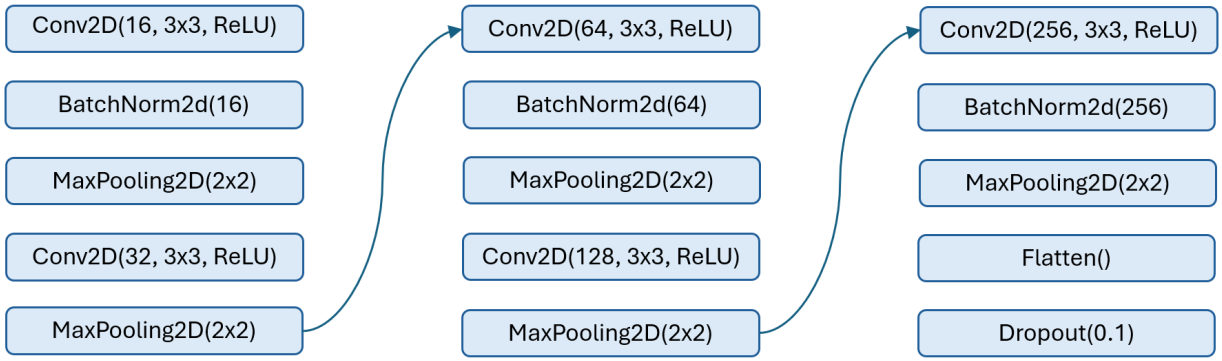


Рис. 4: Архитектура 1

На выходе из нее получим эмбединги размером 256.

В качестве второго варианта была выбрана архитектура ResNet-18 [13]. Она состоит из 17 сверточных слоев. Сначала идут первый сверточный слой с 64 фильтрами и слой MaxPooling. Далее идут 4 блока, каждый из которых состоит из 4 последовательных сверточных слоев и слоев Batch Normalization. В первом блоке каждый сверточный слой имеет 64 фильтра, в последующих блоках число фильтров увеличивается вдвое. Данная архитектура обрабатывает 11.681.896 параметров. Обозначим ее N_2. На выходе из нее получим эмбединги размером 512.

Таким образом имеем две архитектуры энкодера, которые различаются глубиной и размером выходного пространства эмбедингов.

5.2 Проектор

Для обучения модели также используется проектор, который последовательно соединяется с энкодером. Он представляет собой несколько полносвязных слоев. Для каждой

архитектуры будем использовать три полносвязных слоя, каждый из которых имеет 1024 нейрона на выходе. Также после каждого слоя будем использовать Batch Normalization.

6 Программная реализация

Для программной реализации метода VICReg был использован язык Python 3.10.12. В качестве фреймворка была выбрана библиотека PyTorch 2.2.1.

Для удобного написания кода была использована среда PyCharm. Также использовалась среда Google Collab для запуска программы с графическим ускорителем NVIDIA Tesla T4 и для красивой визуализации данных.

6.1 Структура программы

Программа состоит из пяти модулей:

1. Модуль для обучения метода VICReg;
2. Модуль для тестирования метода VICReg;
3. Модуль с описанием архитектуры энкодера;
4. Модуль с аугментациями.

Опишем подробнее каждый из них.

В первом модуле начальным этапом происходит загрузка тренировочной и тестовой частей датасета. Датасет загружается из папки с файлами с помощью функции DataLoader.

Далее создается объект класса VICREG. Данный класс описан в этом же модуле. Он наследуется от torch.nn.Module и переопределяет функцию forward. В этой функции происходит последовательный проход двух ветвей через энкодер и проектор, а также вычисляется функция ошибки:

Листинг 1: Функция forward класса VICREG

```
1     def forward(self, x, y):
2
3         x = self.projector(self.backbone(x))
4         y = self.projector(self.backbone(y))
5
6         var = variance(x) + variance(y)
7         inv = invariance(x, y)
```

```
        cov = covariance(x) + covariance(y)

        var_coeff, inv_coeff, cov_coeff = 25, 25, 1

        loss = var_coeff*var + inv_coeff*inv + cov_coeff*cov

        return loss
```

Листинг 2: Вычисления значений компонент функции потерь

```
    def variance(z):
        return relu(1 - z.std(0)).mean()

    def invariance(x, y):
        return mse_loss(x, y)

    def covariance(z):
        n, d = z.shape
        m = z.mean(0)
        cov = torch.einsum("ni,nj->ij", z-m, z-m) / (n - 1)
        off_diag = cov.pow(2).sum() - cov.pow(2).diag().sum()
        return off_diag / d
```

Затем создаются необходимые файлы формата .json и .pth для сохранения значений функции ошибки на каждой эпохе и для сохранения обученных весов модели соответственно.

Последним этапом происходит обучение модели. В качестве оптимизатора используется Adam.

Во втором модуле первым этапом создается энкодер. Он должен иметь такую же архитектуру, которая использовалась в предыдущем модуле при обучении. С помощью метода load_state_dict в энкодер загружаются обученные веса.

Далее создается модель, которая последовательно соединяет энкодер и однослойный линейный классификатор. Если тестирование проводится для линейной оценки, то с помощью функции requires_grad_ модель настраивается таким образом, чтобы веса энкодера были заморожены и обучался только линейный классификатор. Если же тестирование проводится для частичного обучения, то функция requires_grad_ не используется и обучается вся модель. Способ тестирования можно определить с помощью

передачи аргумента командной строки. По умолчанию проводится линейная оценка.

Затем, аналогично первому модулю, создаются необходимые файлы, и происходит загрузка датасета.

Последним этапом происходит обучение классификатора. В качестве оптимизатора используется SGD, в качестве функции ошибки используется CrossEntropyLoss.

Третий модуль содержит классы, которые описывают вышеупомянутые архитектуры энкодера. Они, как и класс VICREG, наследуются от torch.nn.Module.

Четвертый модуль содержит аугментации, которые необходимы, чтобы разделить изначальную выборку на две ветви. Они написаны с помощью модуля torchvision.transforms.

7 Результаты тестирования

7.1 Датасеты

В качестве множества A будем рассматривать датасет CIFAR-10 [14]. Он состоит из 60.000 цветных изображений размером 32×32. Изображения разделены на 10 классов. В каждом классе содержится 5000 изображений для обучения и 1000 для тестирования. Таким образом, обучающая и тестовая выборка составляют 50.000 и 10.000 изображений соответственно.

В качестве множества B будем рассматривать датасет Tiny ImageNet [15]. Он состоит из 120.000 цветных изображений размером 64×64. Данный датасет больше предыдущего, и в нем изображения разделены на 200 классов. В каждом классе содержится 500 изображений для обучения, 50 для валидации и 50 для тестирования. Таким образом, обучающая и тестовая выборка составляют 100.000 и 10.000 изображений соответственно.

7.2 Обучение с учителем

В первую очередь было проверено, какую точность классификации изображений можно получить, если использовать выбранные архитектуры для классического обучения с учителем.

В каждую архитектуру был добавлен последний линейный слой классификации, после чего проводилось обучение.

Обе архитектуры обучались на 30 эпохах с размером пакета 256. Для обучения использовались оптимизатор SGD и функция ошибки CrossEntropyLoss.

На рисунках 5 и 6 изображены зависимости функции ошибки от количества итераций для архитектуры N_1 и N_2 соответственно.

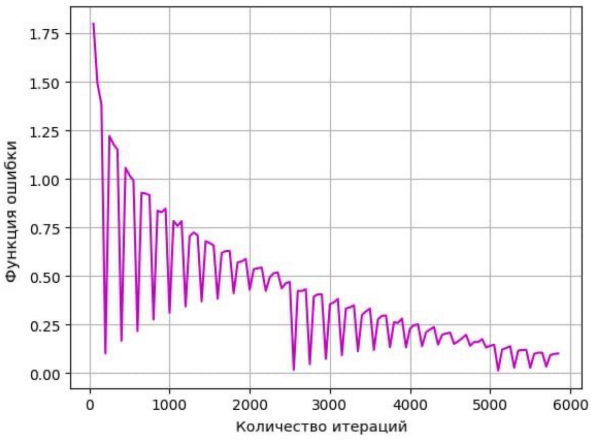


Рис. 5: Архитектура N_1

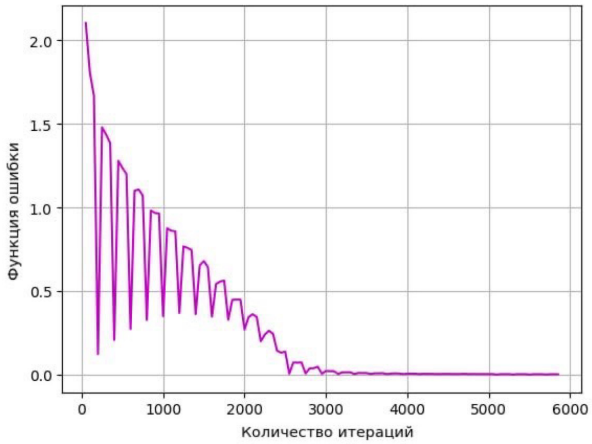


Рис. 6: Архитектура N_2

Для архитектуры N_1 точность Top-1 и Top-5 составили 70.6% и 97.6% соответственно. Для архитектуры N_2 результаты получились хуже на 2.5%: точность Top-1 и Top-5 составили 68.1% и 96.5% соответственно.

7.3 Обучение с помощью метода VICReg

Поскольку изображения в датасете CIFAR-10 небольшие, было принято решение перед обучением использовать следующие аугментации для обработки тренировочного набора:

- RandomResizedCrop(32);
- RandomHorizontalFlip(p=0.5).

Далее для обеих архитектур обучение проводилось на разном количестве эпох. Было попробовано 20, 30 и 40 эпох. Размер пакета для всех случаев составлял 256. Оптимальным оказалось 30 эпох, так как меньшего количества не хватало для хороших результатов точности, а при большем количестве начиналось переобучение.

Для регуляризации компонент функции потерь использовались значения 25, 25, 1 для дисперсии, инвариантности и ковариации соответственно.

На рисунках 7 и 8 изображены зависимости функции ошибки от количества итераций на 30 эпохах для архитектуры N_1 и N_2 соответственно.

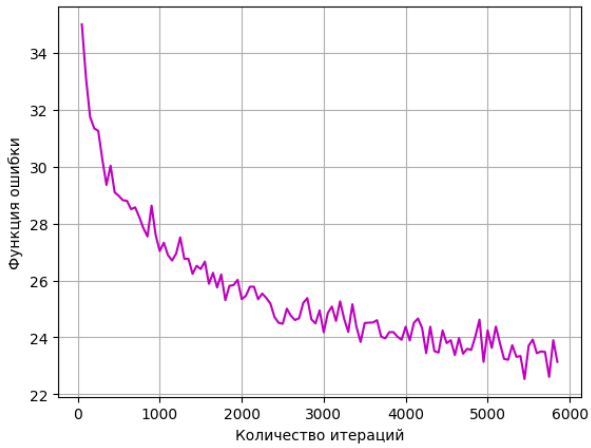


Рис. 7: Архитектура N_1

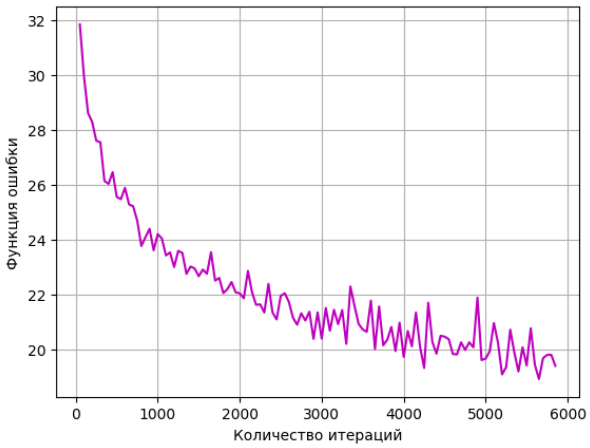


Рис. 8: Архитектура N_2

Также для наглядности к полученным после обучения эмбедингам был применен метод T-SNE [16].

На рисунке 9 изображена проекция на двумерную плоскость изначальной выборки.

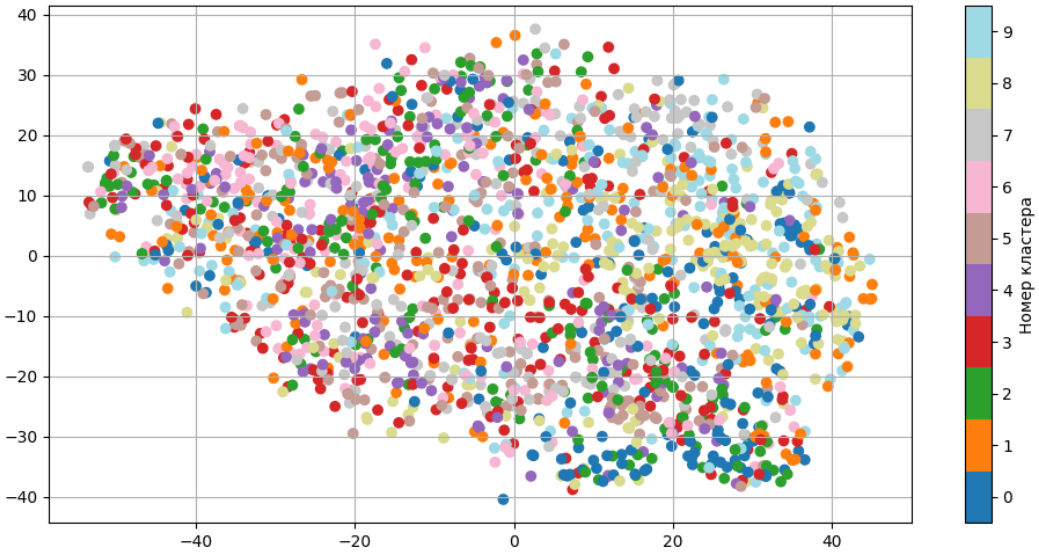


Рис. 9: Применение метода T-SNE для изначальной выборки

На рисунках 10 и 11 изображены отображения для эмбедингов, полученных с помощью архитектуры N_1 и N_2 соответственно

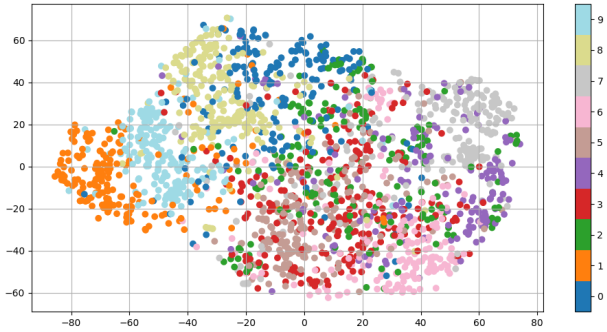


Рис. 10: Архитектура N_1

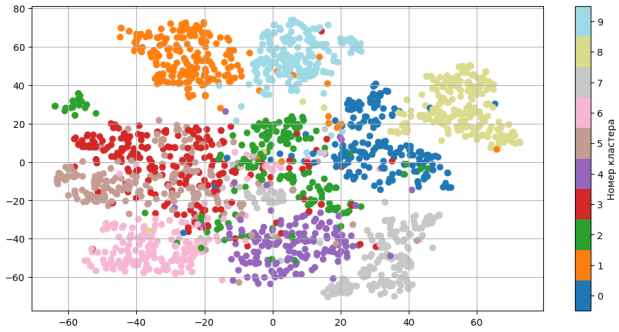


Рис. 11: Архитектура N_2

7.4 Результаты

Тестирование проводилось на 20 эпохах с размером пакета 256. В таблице ниже приведены результаты точности из раздела 7.2, полученные после обучения с учителем, и результаты точности, полученные с помощью метода VICReg на наборе CIFAR-10.

Метод	Архитектура N_1		Архитектура N_2	
	Точность		Точность	
	Top-1	Top-5	Top-1	Top-5
Обучение с учителем [14]	70.6	97.6	68.1	96.5
VICReg [15]	64.8	96.9	89.6	99.7

Как видно из таблицы, с помощью метода VICReg удалось получить более высокую точность, чем при обучении с учителем. Но хороший результат получается только при использовании архитектуры N_2, которая является достаточно сложной, и имеет примерно в 29 раз больше параметров, чем архитектура N_1.

Для архитектуры N_2 было проведено тестирование на наборе Tiny ImageNet. Для тестирования также использовалось частичное обучение, которое описано в разделе 3.4. Тестирование проводилось на 20 эпохах с размером пакета 256. В таблице ниже для сравнения представлены результаты точности на наборе CIFAR-10 и на наборе Tiny ImageNet.

Датасет	Линейная оценка		Частичное обучение			
	Top-1	Top-5	Top-1		Top-5	
			1%	10%	1%	10%
CIFAR-10 [14]	89.6	99.7	85.5	90.3	99.2	99.7
Tiny ImageNet [15]	52.3	74.0	79.7	86.4	94.7	95.9

8 Заключение

В рамках работы были исследованы методы обучения представлению данных. Был проведен обзор существующих подходов, на основе которого для дальнейших экспериментов был выбран метод извлечения признаков на основе обучения без учителя - VICReg.

Цель экспериментов заключалась в исследовании эффективности данного метода. Для исследования была реализована многомодульная программа, включающая в себя обучение нейронной сети с возможностью выбора архитектуры для получения компактных эмбедингов, а также тестирование, которое заключалось в обучении однослойного классификатора, принимающего на вход полученные эмбединги.

В результате эксперимента удалось получить энкодер для генерации эмбедингов, который позволяет решать задачу классификации с более высокой точностью. Результаты были подтверждены на базе CIFAR-10. Энкодер, обученный с помощью метода VICReg повысил точность классификации на 21.5% по сравнению с той же архитектурой, обученной с помощью обучения с учителем.

Полученный энкодер был успешно применен к базе Tiny ImageNet. При этом при дообучении на 10% размеченных данных точность классификации достигла 86.4%.

Список литературы

[1] ImageNet: A large-scale hierarchical image database / Jia Deng, Wei Dong, Richard Socher et al. — 2009. — Pp. 248–255.

[2] *Simonyan, Karen.* Very Deep Convolutional Networks for Large-Scale Image Recognition. — 2015.

[3] *He, Kaiming.* Deep Residual Learning for Image Recognition. — 2015.

[4] *Szegedy, Christian.* Going Deeper with Convolutions. — 2014.

[5] *Bardes, Adrien.* VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. — 2022.

[6] *Ermolov, Aleksandr.* Whitening for Self-Supervised Representation Learning. — 2021.

[7] *Zbontar, Jure.* Barlow Twins: Self-Supervised Learning via Redundancy Reduction. — 2021.

[8] *Chen, Ting.* A Simple Framework for Contrastive Learning of Visual Representations. — 2020.

[9] *He, Kaiming.* Momentum Contrast for Unsupervised Visual Representation Learning. — 2020.

[10] *Lin, James.* Training Keyword Spotters with Limited and Synthesized Speech Data. — 2020.

[11] *Mazumder, Mark.* Few-Shot Keyword Spotting in Any Language. — 2021. — . `http://dx.doi.org/10.21437/Interspeech.2021-1966`.

[12] *Chen, Xinlei.* Improved Baselines with Momentum Contrastive Learning. — 2020.

[13] *Sai Abhishek, Allena Venkata.* Resnet18 Model With Sequential Layer For Computing Accuracy On Image Classification Dataset / Allena Venkata Sai Abhishek. — 2022. — 07. — Vol. 10. — Pp. 2320–2882.

[14] University of Toronto, The CIFAR-10 dataset. — `https://www.cs.toronto.edu/~kriz/cifar.html`.

[15] Papers with code, Tiny ImageNet. — https://paperswithcode.com/dataset/tiny-imagenet.

[16] Scikit-learn, T-SNE. — https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.