В. В. Понамарёв[1], *В. В. Китов*[2], *В. А. Китов*[3]

УЧЁТ ИЕРАРХИИ КЛАССОВ ПРИ КЛАССИФИКАЦИИ ОБЪЕКТОВ С ПОМОЩЬЮ СИАМСКИХ НЕЙРОННЫХ СЕТЕЙ*

Введение

Задача классификации объектов активно решается во многих прикладных областях. Наличие дефекта в изготовленной автозапчасти, болен пациент или здоров, принадлежат ли фотографии одному человеку или разным людям — это примеры подобных задач. Важно их решать автоматически без участия человека с высокой точностью. Помимо стандартных подходов классификации возрастающей популярностью для решения указанной задачи пользуются *сиамские нейронные сети*. Сиамская нейронная сеть автоматически выучивает векторное представление объектов (*эмбеддинг*) таким образом, чтобы объектам одного класса соответствовали близкие эмбеддинги, а объектам разных классов — далёкие. После того, как эмбеддинги выучены задача классификации может быть решена простейшими метрическими методами в пространстве эмбеддингов. Например, можно назначить объект тому классу, который в пространстве эмбеддингов ближе всего к усреднённому эмбеддингу, отвечающему за класс. Пользуясь подобным механизмом, можно математически описать степень близости между, например, изображениями, текстами, звуковыми дорожками или любыми оцифрованными объектами реального мира [18, 22, 21]. Благодаря используемому механизму попарного сравнения квадратично увеличивается обучающая выборка и появляется возможность

[1]аналитик-разработчик программного обеспечения ООО «Яндекс.Технологии», e-mail: valera.pon.vp@gmail.com .

[2]к.ф.-м.н., с.н.с. лаборатории искусственного интеллекта РЭУ имени Г. В. Плеханова, доцент факультета ВМК МГУ имени М. В. Ломоносова, e-mail: v.v.kitov@yandex.ru .

[3]к.т.н., с.н.с. лаборатории искусственного интеллекта РЭУ имени Г. В. Плеханова, e-mail: kitov.va@rea.ru .

*Работа выполнена в рамках государственного задания в сфере научной деятельности Министерства науки и высшего образования РФ на тему «Модели, методы и алгоритмы искусственного интеллекта в задачах экономики для анализа и стилизации многомерных данных, прогнозирования временных рядов и проектирования рекомендательных систем», номер проекта FSSW-2023-0004.

использования небольшого количества обучающих данных [13] (*few-shot learning*). Именно поэтому сиамские сети чаще остальных архитектур используются в идентификации людей по лицам [13, 17] в условиях ограниченных начальных данных и большого набора классов. В последнее время сиамские сети [12, 11] зарекомендовали себя как хорошее решение задач отслеживания объектов на изображениях. Другим достоинством сиамских нейронных сетей является их способность работать с динамически увеличивающимся числом классов. Если появляется объект нового класса, то для него просто выучивается новый эмбеддинг, отличный от эмбеддингов ранее встреченных классов.

Рис. 1. Примеры изображений из CIFAR-100 [8].

В данной работе предлагаются модификации сиамских сетей, улучшающие качество и точность классификации. Популярными методами обучения сиамских сетей являются Contrastive Loss [1], Triplet Loss [19], их модификации Quadruplet Loss [2], но до текущего момента времени модель плохо либо вообще не учитывала иерархическое взаимоотношение объектов и тот факт, что объекты могут быть неравнозначными. Например, сеть классифицирует картинки по тому, что на них изображено: кошки, собаки, машины и т. д., см. рис. 1. Для обычной модели все классы непохожи в одинаковой степени, что в действительности не так. Различия между кошкой и собакой, а, тем более, их породами должно быть намного меньше, чем различия объектами классов из разных областей, таких как собака и катер. В работе показывается, что если учитывать информацию о степени непохожести классов посредством их иерархии, то можно существенно улучшить качество классификации.

Предлагаемый в работе подход позволяет учитывать иерархию объектов и цену ошибки за неправильную классификацию — животные в одной группе, техника во второй, группа животных делится на собак, кошек и т. д., при этом за неправильное определение породы модель

штрафуется слабее, чем за плохой прогноз, когда она перепутала классы, относящиеся к разным областям, например, определила машину как животное. Предложенный метод позволяет на первых этапах учить модель различать базовые группы — животных от машин, мебель от растений и так далее, а на поздних стадиях выучиваются внутриклассовые различия — породы собак, марки машин и т. п. В рамках работы были проведены эксперименты, показывающие улучшение качества классификации относительно базовых подходов на задачах классификации изображений и текстов.

1. Обзор литературы

1.1 Сиамские сети и плоская классификация

Архитектура сиамских нейронных сетей выучивает нейросеть, преобразующую исходное признаковое описание объекта в автоматически выучиваемое векторное представление $f : X \times \Theta \to \mathbb{R}^n$ есть отображение (нейросеть с параметрами Θ), переводящее объекты из X с параметрами в многомерное числовое пространство $\mathbb{R}^n$ (эмбеддинг), Также известна обучающая выборка $\{x_i, y_i\}$, $i = \overline{1,N}$, $x_i \in X$, $y_i \in \{1,\ldots,M\}$, M — число уникальных классов. Для настройки сети обычно используется квадрат Евклидового расстояния за $d : X \times X \times \Theta \to \mathbb{R}$:

$$d(x_i, x_j, \theta) = ||f(x_i, \theta) - f(x_j, \theta)||_2^2, \tag{1}$$

где $x_i, x_j \in X$ — объекты, с которыми мы работаем, $\theta \in \Theta$ — набор весов нейросети f. В процессе обучения на вход сети подаются сразу несколько объектов, и параметры сети оптимизируются для всего набора целиком. Базовым способом обучения является использование функции-потерь Triplet Loss [19]. Это функция-потерь, оперирующая тройками объектов. Из обучающей выборки генерируются тройки объектов x_a, x_p, x_n. Здесь x_a — так называемый *якорь (anchor)*, объект принадлежащий классу y_a, x_p — *позитив*, объект из того же класса, что и якорь, т.е. $y_p = y_a$, и x_n — *негатив*, объект из любого другого класса отличного от класса якоря, т.е. $y_n \neq y_a$. Тогда тройные потери (Triplet loss) по которым настраиваются веса сети, выглядит следующим образом:

$$\max\left\{0, d(x_a, x_p, \theta) - d(x_a, x_n, \theta) + m\right\} \to \min_{\theta},$$

где $m > 0$ — расстояние (гиперпараметр), на которое хотим отдалять якорь и негатив. Алгоритм настраивает сеть f таким образом, чтобы эмбеддинги якоря и позитива в пространстве были близки друг к другу, а эмбеддинг негатива был на расстоянии m от якоря. Визуализация процесса показана на рис. 2.

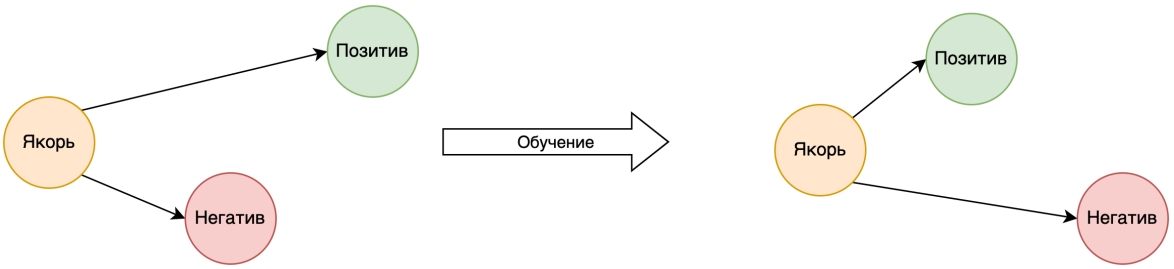


Рис. 2. Функция-потерь Triplet Loss [19] в процессе обучения минимизирует расстояние между якорем и позитивом, одновременно с этим максимизируя расстояния между якорем и негативом.

При использовании сети классифицируемые объекты относятся к тому классу, к которому они ближе всего в пространстве эмбеддингов (для каждого класса нужно рассчитать его усреднённый эмбеддинг по обучающей выборке). Обучение сиамской нейросети хорошо интерпретируемо. Однако оно обладает рядом недостатков. Во-первых, классы могут быть неравнозначны по своей природе, начиная от количества представителей, заканчивая различной ценой за различные типы ошибок. Во-вторых, оно никак не учитывает иерархию классов. Довольно часто при решении задачи классификации можно выстроить иерархию классов. Например, изображения кошек и собак определяются в надкласс «животные», а лодки и автомобили в надкласс «транспорт». Используя иерарахию классов, можно добиться повышения качества прогнозов.

1.2 Иерархическая классификация

Рассмотрим существующие подходы, которые задействуют знание об иерархии классов. Очень часто используют ступенчатую классификацию [16, 15]. В этих подходят строят ансамбли из нейросетей. Начиная с корня дерева иерархии, каждая следующая модель определяет надкласс объекта, например, для иерархии из рис. 3 первая модель определяет одну сущность из трех: «Животные», «Растения» и «Транспорт», вторая модель определяет базовый класс: «Кошка», «Собака» и т.д. Далее ответы агрегируются в окончательный результат. Такой подход прост в реализации, но требует дополнительных ресурсов для поддержки сразу нескольких классификаторов.

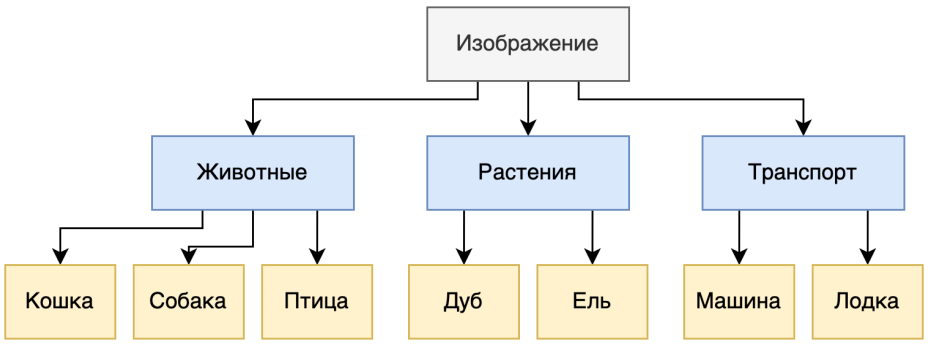


Рис. 3. Пример иерархии классов. Три уровня иерархии и семь базовых классов.

В работе [24] авторы использовали интересный подход в построении эмбеддингов для базовых классов вверху иерархии. А затем обучали модель, не из семейства сиамских сетей, таким образом, чтобы эмбеддинги объектов сосредотачивались вокруг центрующих эмбеддингов классов. Таким образом авторам удалось достичь повышения качества классификации.

Важные идеи были предложены в статье [25]. В ней авторы предлагают модифицировать функцию-потерь, которая используется в обучении сиамских сетей, таким образом, чтобы прямо в процессе обучения была задействована информация об иерархии классов. За основу была взята описанная ранее функция-потерь Triplet Loss [19]. Обозначим за $D : X \times X \times X \times \Theta \to \mathbb{R}$ разность расстояний между якорем и позитивом и якорем и негативом:

$$D(x_a, x_p, x_n, \theta) = d(x_a, x_p, \theta) - d(x_a, x_n, \theta).$$

Тогда функция-потерь перепишется в виде:

$$\max\left\{0, D(x_a, x_p, x_n, \theta) + m\right\} \to \min_{\theta}. \tag{2}$$

Была предложена следующая модификация:

$$\max\left\{0, D(x_a, x_p, x_n, \theta) \cdot h_{pn} \cdot c_p + m_p\right\} \to \min_{\theta} \tag{3}$$

Ключевое отличие заключается в добавлении мультипликаторов. h_{pn} — это иерархический гиперпараметр, который задает различие позитивного и негативного классов. Этот гиперпараметр необходимо определять вручную так, чтобы для непохожих классов значение было большое, а для схожих классов значение было близко к единице. Коэффициент c_p задает значимость класса и определяется как обратная величина к количеству объектов в классе, т.е. $c_p = \frac{1}{N_p}$, где N_p — число объектов в классе p. Также была изменена величина межклассового сдвига m_p, теперь она для каждого класса разная, т.е. по-разному отдаляется каждый класс от всех остальных. Авторы статьи [26] предложили использовать в качестве m_p модифицированный вариант

расстояния до общего предка между классами с учетом мощности каждого класса.

2. Предлагаемый подход

Опишем предлагаемый подход в обучении сиамских сетей и модификации, позволяющие достичь прироста качества классификации объектов. Предлагаемые улучшения будут вводиться поэтапно, с каждым новым элементом все больше усложняя оригинальную функцию потерь Triplet Loss [19].

2.1 Иерархия классов

Как было описано выше, авторы [25] продемонстрировали один из способов встраивания иерархии классов в функции потерь в виде дополнительно мультипликатора h_{pn} в формуле (3). Здесь есть несколько важных проблем, во-первых, слияние мультипликаторов h_{pn} и c_p может нивелировать эффект всего множителя, во-вторых, h_{pn} является гиперпараметром, который настраивает пользователь, что дополнительно усложняет обучение сети. Он должен самостоятельно определить, как классы друг от друга отличаются численно. Первая проблема решается разделением множителей, они также будут настраивать разделение классов в пространстве, но осуществлять это несколько другим способом. А вторая проблема будет решена, если заменить гиперпараметр h_{pn} на величину, напрямую зависящую от классов без участия пользователя. Формула (2) приобретет следующий вид:

$$h_{pn} \cdot \max\left\{0, D(x_a, x_p, x_n, \theta) \cdot c_p + m\right\} \to \min_{\theta}, \tag{4}$$

где $h_{pn} \in (0,1]$, $c_p \in (0,1]$. Теперь h_{pn} стал глобальным множителем, который будет полностью масштабировать весь максимум, а не только отдельную его часть, также сильнее отделён от c_p, что позволяет коэффициентам по отдельности внести более существенный вклад. Само значение h_{pn} будет определяться как расстояние до ближайшего общего предка в дереве иерархии, или задача наименьшего общего предка (НОИ) [6]:

$$h_{pn} = \frac{\mathrm{LCA}(p,n)}{d-1}, \tag{5}$$

где p,n — позитивный и негативный классы, $p,n \in [1,\ldots,M]$, а d — глубина дерева иерархии, $d \in \mathbb{N}$. Величина d нужна для нормировки весов h_{ij}. Таким образом величина h_{pn} будет задавать степень различия классов p и n и $h_{pn} = h_{np}$, элементы h_{ij} составляют симметричную матрицу $H \in \mathbb{R}^{M \times M}$. Если классы сильно удалены в дереве иерархии, $h_{pn} \simeq 1$, то их представителей нужно отодвигать друг от друга на большое расстояние, а если значение h_{pn} невелико, то отдалять классы можно на меньшую дистанцию. Этот алгоритм снимает с пользователя

необходимость вручную подбирать коэффициенты h_{pn} различия классов. Пример иерархии с описанием представлен на рис. 4.

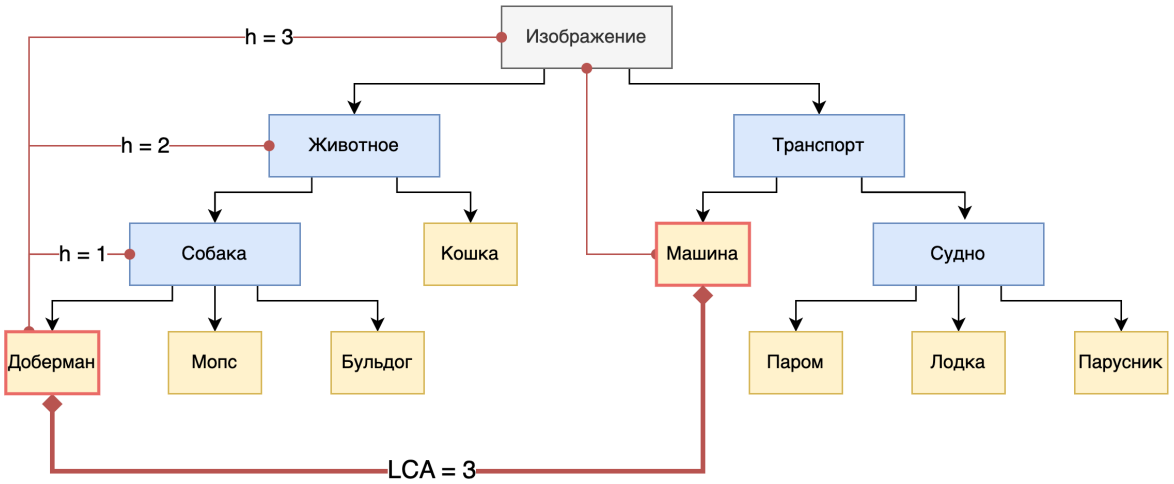


Рис. 4. Пример иерархии классов. Для каждой пары классов i и j определяется значение h_{ij}. Например, для классов «Доберман» и «Машина» $\mathrm{LCA}(i,j)$ [6] принимает значение 3 как расстояние до ближайшего общего предка «Изображение». Учитывая глубину иерархии $d = 4$, получим $h_{ij} = 1$.

Также вместе с h_{pn} был изменен мультипликатор c_p, до этого он определялся обратной пропорциональностью от мощности класса, то есть от общего количества объектов в нем. Такой механизм позволял сильнее отодвигать в пространстве эмбеддингов большие классы (содержащие много объектов). Вместо этого предлагается использовать:

$$c_p = \frac{N_p}{\max\limits_{i=1,\dots,K} N_i}, \tag{6}$$

где N_i, $N_i \in \mathbb{N}$ — число объектов в классе i, $i = \overline{1,K}$. Теперь мультипликатор c_p пропорционален нормированной величине N_p на мощность наибольшего класса. Чем коэффициент меньше, тем больше в процессе обучения отдаляются классы с небольшим числом объектов, акцент происходит именно на них. Мотивация формы вычисление мультипликатора в таком виде заключается в том, что модель в процессе обучения так или иначе научится различать большие классы. Этому способствует значительное количество объектов в них, и модели будет хватать данных, чтобы установить межклассовые различия. С другой стороны классы малой мощности в пространстве будут располагаться на удалении от крупных классов, не смешиваясь с ними, что позволит ошибаться при классификации на них значительно реже.

2.2 Матрицы цен

Ещё одна модификация связана с использованием матрицы цен [3] за неправильную классификацию объектов. Штрафуя по-разному ошибки разных типов можно управлять их распределением и улучшать процедуру обучения. В Triplet Loss [19] использовалась классы удалялись друг от друга на единую константу $m > 0$ для всех классов. В [25] было предложено использование слагаемого m_i, который задавал удаленность класса i от всех остальных. Наша предлагаемая модификация опирается на идею использования попарного расстояния между классами, тогда формула (4) приобретет новый вид:

$$h_{pn} \cdot \max\left\{0, D(x_a, x_p, x_n, \theta) \cdot c_p + m_{pn}\right\} \to \min_{\theta},$$

где m_{pn} задает базовое расстояние между классами. Это квадратная матрица $M \times M$ (попарных расстояний между классами). Задавать значения матрицы можно вручную, если пользователь заранее знает, насколько отдалять классы. В нашей работе же предлагается автоматическая процедура заполнения значений этой матрицы на основе матрицы ошибок [4], проделав следующие шаги:

1. Обучить сиамскую нейронную сеть с помощью Triplet Loss [19] на известной обучающей выборке.
2. Рассчитать матрицу ошибок $E \in \mathbb{R}^{M \times M}$ на валидационной выборке.
3. Построчно нормируем матрицу E, получая матрицу $\hat{E}$:
$$\hat{E}_{ij} = \frac{E_{ij}}{\sum\limits_k E_{ik}}.$$
4. Определяем коэффициенты m_{pn}:
$$m_{pn} = \begin{cases} 0 & p = n \\ 1 + \hat{E}_{pn} & p \neq n. \end{cases} \tag{7}$$

Стоит отметить, что необходимо обучить сначала одну модель, а затем, по её ошибкам, вторую, зато, как показывают эксперименты ниже, точность прогнозов от этого возрастет.

2.3 Убывающий иерархический мультипликатор

В секции ранее мы описали способ использования иерархии классов (4) в обучении сиамских сетей. На протяжении всего обучения модель делает больший акцент в пользу отделения далеких по иерархии классов, $h_{pn} \simeq 1$. Нам действительно важно точно определять существенно разные группы, но в таком механизме есть риск игнорирования различий между похожими классами. Например, с высокой точностью модель будет различать изображения кошек, машин и

деревьев, но плохо отличать породы собак. Чтобы бороться с этой проблемой, алгоритму в какой-то момент времени нужно уменьшить влияние веса иерархии h_{pn}. Для этого предлагается модификация в виде убывающего иерархического коэффициента $h_{pn}^{\phi(s)}$:

$$\mathscr{L} = h_{pn}^{\phi(s)} \cdot \max\left\{0, D(x_a, x_p, x_n, \theta) \cdot c_p + m_{pn}\right\} \to \min_{\theta}, \tag{8}$$

где используется монотонно убывающая функция $\phi(s)$ от номера итерации обучения s. Благодаря добавлению $\phi(s)$ можно контролировать, как иерархическая структура будет влиять на обучение с течением времени. А именно выстраивается такая схема, при которой на первых итерациях модель учится различать совсем разные объекты, например рис. 4, изображения собак и машин, а на поздних итерациях влияние иерархии сводится к нулю, модель хорошо начинает отделять сложно различимые классы, например, породы животных. Этого удается достичь за счет монотонности функции $\phi(s)$, $\phi(s) \to 0$ при $s \to \infty$, что приближает мультипликатор h_{pn} к единице.

Здесь есть некоторая свобода выбора функции $\phi(s)$. Чем быстрее она стремится к нулю, тем раньше иерархия H выключается из процесса обучения. При выборе $\phi(s)$ стоит ориентироваться на общее количество эпох и итераций в обучении и подбирать функцию под эти значения. Например, если число итераций измеряется в миллионах, то стоит выбрать $\phi(s) = 1/\ln(s+1)$, если несколько тысяч, то $\phi(s) = s^{-0.5}$. В любом случае, $\phi(s)$ должна монотонно убывать к нулю.

Так как современные нейронные сети обучаются сразу на нескольких объектах минибатча [5], а мы используем изменяющийся во времени иерархический мультиприкатор (8), то для лучшей настройки минибатчи также рекомендуется подбирать особым образом. На первых итерациях обучения модель должна хорошо выучить различия между далекими и непохожими классами, поэтому минибатч вначале обучения рекомендуется формировать так, чтобы туда попадали тройки объектов из как можно более разных частей иерархии классов. В таком случае идея с нисходящим коэффициентом $h_{pn}^{\psi(s)}$ будет хорошо работать. Один из возможных способов формирования минибатча:

1. Пусть размер минибатча равен B, $B \in \mathbb{N}$. Случайно выбираем $3B$ троек x_a, x_p, x_n, где x_a — якорь, x_p — позитив, x_n — негатив, (напомним, что якорь и позитив —объекты из одного класса, а негатив — объект другого класса).
2. Для каждой тройки вычисляем значение h_{pn} — иерархический вес между негативным и позитивным классом.
3. Сортируем тройки по убыванию иерархического веса h_{pn}.

4. Первые αB, $\alpha \in [0,1]$, элементов отсортированной последовательности помещаем в минибатч, а остальные $(1-\alpha)B$ позиций в минибатче выбираем случайно из оставшихся в последовательности троек объектов.

5. Изменяем α, например, прибавляем 0.001. Идея здесь такая же: на первых шагах обучения выбираем максимально непохожие классы, а позже можно свести α к нулю и сэмплировать классы случайно.

3. Эксперименты

В этой секции будут представлены эксперименты на разных данных, будут описаны параметры запусков, результаты и сравнения разных методов.

3.1 Датасеты и метрики качества

3.1.1 CIFAR-100

Датасет CIFAR-100 [8] состоит из 60,000 изображений 32×32 пикселей и 100 классов, пример на рис. 5. На каждый класс приходится 400 изображений для обучения, 100 изображений для валидации и 100 изображений для тестирования. Каждый из 100 плоских классов определен в иерархию, у которой общая глубина $d = 5$.

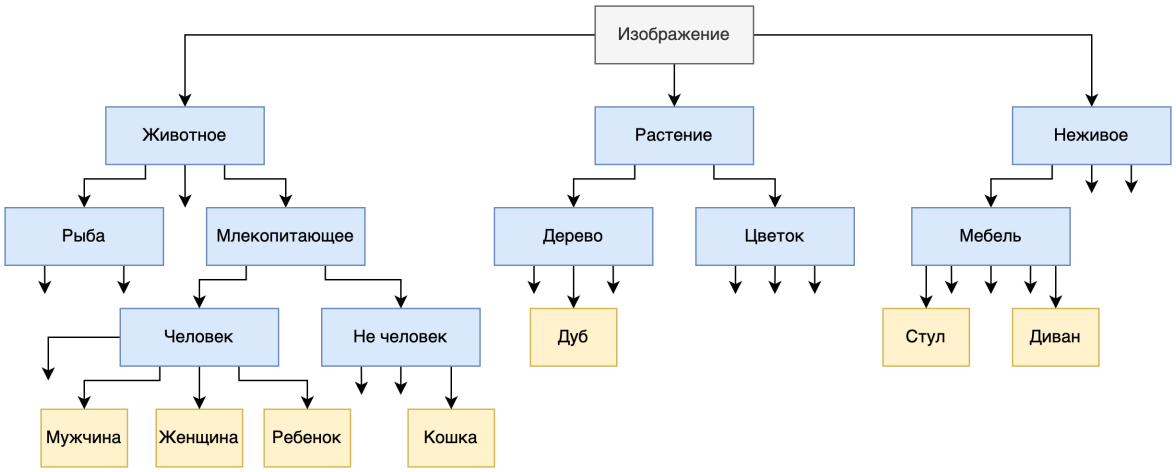


Рис. 5. Часть дерева иерархии датасета CIFAR-100 [8] с глубиной $d = 5$. Плоские классы являются листьями дерева, помечены желтым цветом.

3.1.2 Amazon Product Reviews

Датасет Amazon Product Reviews (APR) [27] состоит из отзывов на товары интернет-магазина Amazon. Он содержит 510 плоских классов, 50,000 обучающих, 10,000 валидационных и 10,000 тестовых объектов. Каждый объект — это отзыв на конкретный товар. Товары имеют иерархию с глубиной $d = 4$, отзывы наследуют эту структуру. В датасете

определены 6 базовых классов: гигиена, игрушки и игры, косметика, домашние животные, детские товары и бакалея для гурманов.

3.1.3 Fashion MNIST

В третьем датасете Fashion MNIST [28] содержатся черно-белые картинки элементов одежды 28×28 пикселей. Всего в датасете 10 классов, которые разбиваются на надклассы: верхняя одежда, нижняя одежда, обувь и сумки. Так выстраивается иерархии с глубиной $d = 3$. Обучающая выборка — 50,000 изображений, валидационная и тестовая содержат по 10,000 изображений в каждой. Пример изображений Fashion MNIST [28] продемонстрирован на рис. 6.

Рис. 6. Пример изображений из датасета Fashion MNIST [28].

3.1.4 Метрики качества

Чтобы оценить точность классификации, будем использовать две метрики качества: Accuracy и F_1-macro. Пусть есть множество пар объект-метка $X = \{x_i, y_i\}_{i=1}^{N}$, где x_i признаковое описание объекта i, а y_i номер класса, $y_i \in [1, \ldots, K]$, и исследуемая модель $f : X \times \Theta \to Y$ предсказала объекту x_i класс $\hat{y}_i$. Модель могла сделать это правильно, а могла ошибиться. Метрика Accuracy задается как отношение количества правильно определенных классов на общее число объектов:

$$\mathrm{Acc}(f, X) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[y_i = \hat{y}_i].$$

Метрика качества F_1-macro задается несколько сложнее. Для начала нужно построить матрицу ошибок, каждый элемент (i, j) которой есть количество раз, когда модель определила объект из класса i в классе j. Ответы бывают правильные T_i (True), а бывают ложные F_i (False). Пример представлен на таблице ниже:

Тогда F_1-macro определяется как среднее следующих величин:

$$\mathrm{Pr}_i(f, X) = \frac{T_{ii}}{T_{ii} + \sum\limits_{j=1, j \neq i}^{K} F_{ij}},$$

Predicted \ True	$y=1$	$y=2$	$y=3$
$\hat{y}=1$	T_{11}	F_{12}	F_{13}
$\hat{y}=2$	F_{21}	T_{22}	F_{23}
$\hat{y}=3$	F_{31}	F_{32}	T_{33}

Табл. 1. Пример матрицы ошибок для трех классов. По горизонтали отмечены истинные классы, по вертикали — спрогнозированные моделью.

$$R_i(f,X) = \frac{T_{ii}}{T_{ii} + \sum\limits_{j=1, j\neq i}^{K} F_{ji}},$$

$$F_1\text{-macro}(f,X) = \frac{1}{K}\sum_{i=1}^{K} \frac{2 \cdot Pr_i(f,X) \cdot R_i(f,X)}{Pr_i(f,X) + R_i(f,X)}.$$

3.2 Архитектуры нейросетей

На всех трех указанный датасетах произведены запуски разных моделей с целью сравнения точности классификации. В качестве моделей для обработки изображений будут использованы нейронные сети, базирующиеся на архитектурах VGG-11 [35] и ResNet-18 [30]. Нейросети предобучены на большой коллекции изображений ImageNet [14]. Нейронная сеть, базирующаяся на VGG-11 [35], состоит из пяти

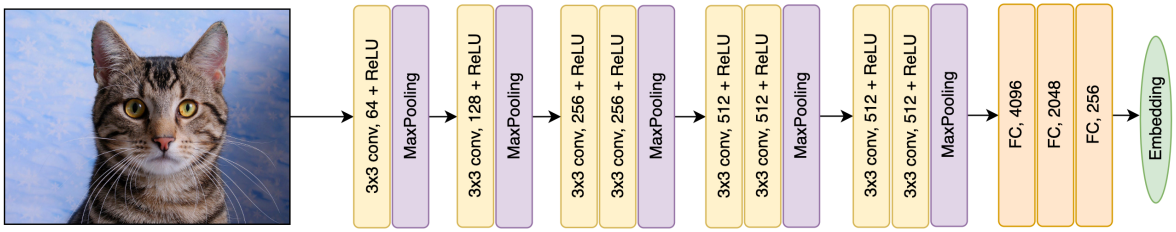


Рис. 7. Нейронная сеть VGG-11, используемая при классификации изображений CIFAR-100 [8] и Fashion MNIST [28].

сверточных блоках и финальных полносвязных слоев, рис. 7. Веса всех сверточных блоков у сети были заморожены, обучались лишь последние полносвязаные. Сеть ResNet-18 [30] использует предобученные параметры за исключением последних двух полносвязных слоев, их будем обучать под описанные задачи классификации изображений. Эмбеддинг изображения извлекается из последнего полносвязного слоя размера 256.

Для классификации текстов будем использовать использовать модель BiLSTM [31]. Для слов будут использованы предобученные эмбеддинги Word2Vec [33]. Размерность входных эмбеддингов слов и

выходных эмбеддингов — 300. Длина входной последовательности не превышает 1000 слов. В скрытых состояниях 600 нейронов. Сеть BiLSTM [31] состоит из двух слоёв: первый передает скрытые состояния слева направо, второй — в обратную сторону.

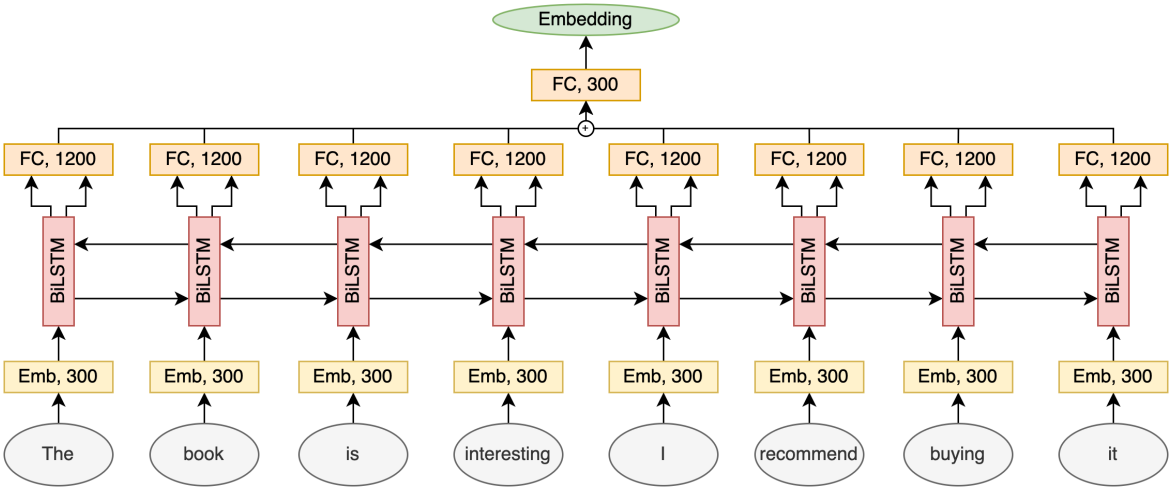


Рис. 8. Архитектура сети BiLSTM [31], в качестве эмбеддингов используются Word2Vec [33] представления. Ячейка BiLSTM состоит из двух слоев, каждый из них возвращает скрытое состояние, их конкатенация подается на вход полносвязному слою.

3.3 Сравнение методов

Приведём детали и результаты проведённых экспериментов. В таблицах (2), (3) находятся сравнения качества предсказания моделей на разных данных по метрикам F_1-macro и Accuracy. Далее будем обозначать сиамскую сеть, обученную на Triplet Loss [19], как Triplet Loss, а модель, обученную на Hierarchical Triplet Loss [25], как Hierarchical Triplet Loss соответственно.

Method	CIFAR-100		FMNIST		APR
	VGG-11	ResNet-18	VGG-11	ResNet-18	BiLSTM
Std.Classification	0.651	0.652	0.825	0.845	0.612
Std.Classification+L2	0.686	0.710	0.873	0.880	0.627
Triplet Loss [19]	0.693	0.703	0.870	0.874	0.614
Ladder-based classification [16]	0.699	0.718	0.883	0.886	0.622
Hierarchical Triplet Loss [25]	0.705	0.712	**0.889**	0.883	0.632
Предложенный метод (8)	**0.726**	**0.738**	0.887	**0.905**	**0.641**

Табл. 2. Сравнение методов по метрике F_1-macro на разных датасетах и моделях.

Все модели обучались с помощью оптимизатора Adam [32] с параметром $lr = 3 \cdot 10^{-5}$. Помимо сиамского способа обучения,

Метод	CIFAR-100		FMNIST		APR
	VGG-11	ResNet-18	VGG-11	ResNet-18	BiLSTM
Std.Classification	0.683	0.707	0.860	0.897	0.633
Std.Classification+L2	0.728	0.741	0.888	0.921	0.655
Triplet Loss [19]	0.713	0.734	0.903	0.909	0.638
Ladder-based classification [16]	0.708	0.720	0.899	0.919	0.641
Hierarchical Triplet Loss [25]	0.719	0.729	0.892	0.901	0.639
Предложенный метод (8)	**0.741**	**0.752**	**0.918**	**0.933**	**0.664**

Табл. 3. Сравнение методов по метрике Accuracy на разных датасетах и моделях.

рассмотрены классические подходы классификации — Std.Classification и Std.Classification+L2. В них модель напрямую классифицирует объект без сравнения с другими, используются те же архитектуры сетей, но с кросс-энтропийной функцией-потерь [36], а в конце после полносвязных слоев используется SoftMax, чтобы от логитов перейти в вероятностям. В модели Std.Classification+L2 используется дополнительный L2 регуляризатор для весов модели. Модель, обучаемая на Triplet Loss [19], использует параметр $m = 1$. В Ladder-based classification [16] для каждого уровня иерархии обучается сиамская сеть с помощью Triplet Loss с параметром $m = 1$. В подходе Hierarchical Triplet Loss [25] вес иерархии формируется из расстояния до общего предка (5), а все $m_i = 1$. Размер минибатча оставался постоянным $B = 64$. В предложенном методе $\phi(s) = \ln^{-1}(s+1)$, α стартует со значения $\alpha = 1$, уменьшаясь на 0.0005 после каждой итерации.

После обучения для каждого класса формировался центральный объект как усреднее всех многомерных векторов объектов обучающей выборки. Эти центральные эмбеддинги, или центроиды, нужны для классификации объектов. Каждый из объектов тестовой выборки пропускается через нейронную сеть, для них формируются векторные представления. Класс объекта определяется как класс ближайшего центроида по мере d (1). Есть альтернативный способ классификации объектов — присваивать тестовому объекту класс ближайшего объекта из обучающей выборки. Но в таком случае сильно увеличивается время классификации, так как необходимо посчитать расстояние до каждого объекта из обучающей выборки. Экспериментально значительной разницы между двумя подходами не выявлено, далее использовался подход с центроидами.

Предложенный метод отлично себя показал при классификации изображений. Качество относительно существующих методов обучения значительно выше, за исключением эксперимента на FMNIST моделью VGG-11, где качество очень близко к оптимальному.

3.4 Вклад улучшений

Предложенный метод (8) состоит из нескольких модификаций Triplet Loss [19]. Каждое улучшение вносит свой вклад. В этой секции проводится эксперимент, выявляющий степень вкладов каждого улучшения из (8). На таблицах 4, 5 представлены результаты этого сравнения. За основу берется базовая версия Triplet Loss [19] и дополняется по одной модификации: h_{pn}, $h_{pn}^{\phi(s)}$, c_p, m_{pn}. Модели обучаются на функции-потерь [19] с добавлением одной модификации. Предложенная модификация иерархического веса h_{pn} вносит значительный вклад в качество классификации. Но наибольший эффект достигается уменьшением влияния иерархии с течением времени при добавлении степени $\phi(s)$. Остальные модификации также вносят вклад, но немного меньше, чем иерархичесий коэффициент. Однако в совокупности все модификации значительно повышают точность классификации объектов.

Метод	CIFAR-100		FMNIST		APR
	VGG-11	ResNet-18	VGG-11	ResNet-18	BiLSTM
Triplet Loss [19]	0.693	0.703	0.870	0.874	0.614
TL + h_{pn}, (5)	0.708	0.717	0.876	0.885	0.629
TL + $h_{pn}^{\phi(s)}$, (8)	0.719	0.723	0.883	0.891	0.632
TL + c_p, (6)	0.688	0.711	0.879	0.886	0.619
TL + m_{pn}, (7)	0.707	0.712	0.877	0.880	0.625
Предложенный метод (8)	**0.726**	**0.738**	**0.887**	**0.905**	**0.641**

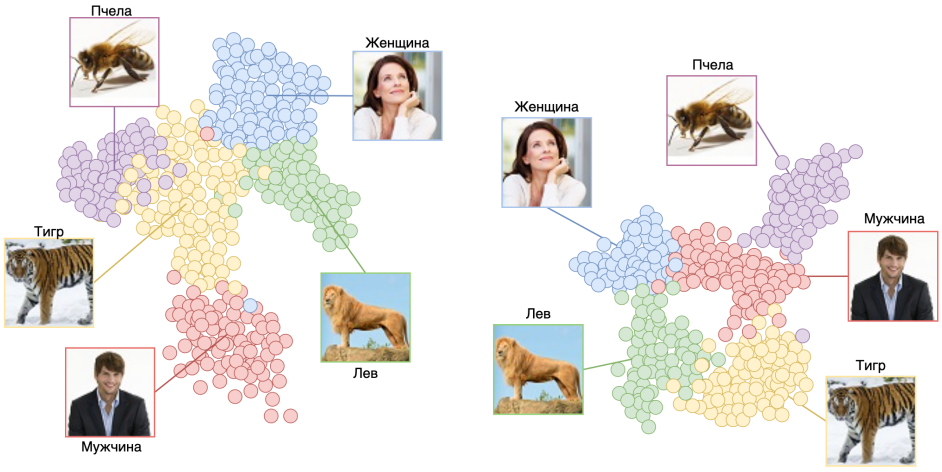
Табл. 4. Сравнение вкладов отдельных элементов предложенной схемы по метрике F_1-macro на разных датасетах и моделях. Предложенный метод — совокупность всех модификаций.

Метод	CIFAR-100		FMNIST		APR
	VGG-11	ResNet-18	VGG-11	ResNet-18	BiLSTM
Triplet Loss [19]	0.713	0.734	0.903	0.909	0.638
TL + h_{pn}, (5)	0.728	0.739	0.908	0.915	0.644
TL + $h_{pn}^{\phi(s)}$, (8)	0.735	0.746	0.911	0.928	0.650
TL + c_p, (6)	0.721	0.738	0.905	0.910	0.641
TL + m_{pn}, (7)	0.724	0.740	0.907	0.912	0.651
Предложенный метод (8)	**0.741**	**0.752**	**0.918**	**0.933**	**0.664**

Табл. 5. Сравнение вкладов отдельных элементов предложенной схемы по метрике Accuracy на разных датасетах и моделях. Предложенный метод — совокупность всех модификаций.

Предложенный метод хорошо работает в разных задачах, но стоит учитывать, что у него есть определённые ограничения. Во-первых, нужно затратить дополнительные ресуры для построения иерархии классов H,

если она заранее не специфицирована. А во-вторых, чтобы отдалять классы по-разному, предлагается сначала обучить сиамскую сеть на базовом Triplet Loss [19], а затем по ошибкам построить попарные сдвиги m_{pn} (7), что требует дополнительных вычислительных ресурсов.



(a) Triplet Loss [19] (b) Предложенный метод

Рис. 9. Распределение объектов валидационной выборки в многомерном пространстве после PCA [34] преобразования на двумерную плоскость для разных методов обучения сиамской сети.

На рисунке 9 изображено распределение классов изображений датасета CIFAR-100 [8, для простоты визуализации использовано небольшое подмножество классов, набор многомерных векторов отображён на двумерную плоскость с помощью метода главных компонент [34]. При обучении стандартным способом с помощью Triplet Loss [19] классы в пространстве могут располагаться как угодно, что может нарушать логику о близком расположении объектов похожих классов. Например, на рисунке 9a классы «Мужчина» и «Женщина» удалены друг от друга, а классы «Пчела» и «Тигр» имеют большое пересечение. Преимуществом предложенного метода является выстраиваение иерархических эмбеддингов, то есть похожие объекты, или объекты близкие по иерархии, в пространсве будут располагаться рядом. После обучения сиамской сети предложенным методом классы «Мужчина» и «Женщина» располагаются рядом, а классы «Тигр» и «Пчела», благодаря учёту иерархии при обучении, отдалены друг от друга.

Заключение

Сиамские нейронные сети представляют собой эффективный способ классификации объектов, лучше использующий органиченную обучающую выборку за счёт обучения не на отдельных объёктах, а на

тройках объектов. Это позволяет классифицировать даже редкие классы всего по нескольким примерам. Также указанная архитектура позволяет органично встраивать в обучение новые классы — достаточно лишь выучить для них новые эмбеддинги, отличные от ранее встреченных классов. В работе предложен ряд подходов по улучшенной настройке сиамских нейронных сетей за счёт учёта иерархии классов. В начале обучения нейросеть учится различать классы с существенными отличиями (например, животных от машин), а потом учится разделять уже более похожие классы (например, породы собак или марки машин). Экспериментально показано, что предложенный подход способен точнее решать задачу классификации изображений и текстов в большинстве случаев.

Литература

1. Chopra S., Hadsell R., LeCun Y. Learning a similarity metric discriminatively, with application to face verification //2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). – IEEE, 2005. – Т. 1. – С. 539-546. DOI: 10.1109/CVPR.2005.202.
2. *Chen W. et al.* Beyond triplet loss: a deep quadruplet network for person re-identification //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 403-412.
3. *Ciraco M., Rogalewski M., Weiss G.* Improving classifier utility by altering the misclassification cost ratio //Proceedings of the 1st international workshop on Utility-based data mining. – 2005. – С. 46-52. DOI: 10.1145/1089827.1089833.
4. *Liang J.* Confusion matrix: Machine learning //POGIL Activity Clearinghouse. – 2022. – Т. 3. – №. 4.
5. *Masters D., Luschi C.* Revisiting small batch training for deep neural networks //arXiv preprint arXiv:1804.07612. – 2018.
6. *Bender M. A. et al.* Lowest common ancestors in trees and directed acyclic graphs //Journal of Algorithms. – 2005. – Т. 57. – №. 2. – С. 75-94.
7. *Dong X., Shen J.* Triplet loss in siamese network for object tracking //Proceedings of the European conference on computer vision (ECCV). – 2018. – С. 459-474.
8. Krizhevsky A. et al. Learning multiple layers of features from tiny images. – 2009.
9. Memon S. A., Khan K. A., Naveed H. HECNet: a hierarchical approach to enzyme function classification using a Siamese Triplet Network //Bioinformatics. – 2020. – Т. 36. – №. 17. – С. 4583-4589. DOI: 10.1093/bioinformatics/btaa536.

10. Barz B., Denzler J. Hierarchy-based image embeddings for semantic image retrieval //2019 IEEE winter conference on applications of computer vision (WACV). – IEEE, 2019. – C. 638-647. DOI: 10.1109/WACV.2019.00073.
11. Lee N., Hong S., Kim H. Single-Trace Attack Using One-Shot Learning With Siamese Network in Non-Profiled Setting //IEEE Access. – 2022. – T. 10. – C. 60778-60789. DOI: 10.1109/ACCESS.2022.3180742.
12. Wu Y. et al. A novel Siamese network object tracking algorithm based on tensor space mapping and memory-learning mechanism //Journal of Visual Communication and Image Representation. – 2023. – T. 91. – C. 103742.
13. Heidari M., Fouladi-Ghaleh K. Using Siamese networks with transfer learning for face recognition on small-samples datasets //2020 International Conference on Machine Vision and Image Processing (MVIP). – IEEE, 2020. – C. 1-4. DOI: 10.1109/MVIP49855.2020.9116915.
14. Deng J. et al. Imagenet: A large-scale hierarchical image database //2009 IEEE conference on computer vision and pattern recognition. – Ieee, 2009. – C. 248-255. DOI: 10.1109/CVPR.2009.5206848.
15. Amin A. H. M., Khan A. I. One-shot classification of 2-D leaf shapes using distributed hierarchical graph neuron (DHGN) scheme with k-NN classifier //Procedia Computer Science. – 2013. – T. 24. – C. 84-96.
16. VILCEK A. et al. Transformer-Based Deep Siamese Network for At-Scale Product Matching and One-Shot Hierarchy Classification. – 2018.
17. Song C., Ji S. Face Recognition Method Based on Siamese Networks Under Non-Restricted Conditions //IEEE Access. – 2022. – T. 10. – C. 40432-40444. DOI: 10.1109/ACCESS.2022.3167143.
18. Melekhov I., Kannala J., Rahtu E. Siamese network features for image matching //2016 23rd international conference on pattern recognition (ICPR). – IEEE, 2016. – C. 378-383. DOI: 10.1109/ICPR.2016.7899663.
19. Schroff F., Kalenichenko D., Philbin J. Facenet: A unified embedding for face recognition and clustering //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2015. – C. 815-823.
20. Daudt R. C., Le Saux B., Boulch A. Fully convolutional siamese networks for change detection //2018 25th IEEE International Conference on Image Processing (ICIP). – IEEE, 2018. – C. 4063-4067. DOI: 10.1109/ICIP.2018.8451652.
21. Wu Y. et al. Siamese capsule networks with global and local features for text classification //Neurocomputing. – 2020. – T. 390. – C. 88-98.

22. Wang C., Tzanetakis G. Singing style investigation by residual siamese convolutional neural networks //2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – IEEE, 2018. – С. 116-120. DOI: 10.1109/ICASSP.2018.8461660.

23. He A. et al. A twofold siamese network for real-time object tracking //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2018. – С. 4834-4843.

24. Barz B., Denzler J. Hierarchy-based image embeddings for semantic image retrieval //2019 IEEE winter conference on applications of computer vision (WACV). – IEEE, 2019. – С. 638-647. DOI: 10.1109/WACV.2019.00073.

25. Memon S. A., Khan K. A., Naveed H. HECNet: a hierarchical approach to enzyme function classification using a Siamese Triplet Network //Bioinformatics. – 2020. – Т. 36. – №. 17. – С. 4583-4589. DOI: 10.1093/bioinformatics/btaa536.

26. Ge W. Deep metric learning with hierarchical triplet loss //Proceedings of the European conference on computer vision (ECCV). – 2018. – С. 269-285.

27. Haque T. U., Saber N. N., Shah F. M. Sentiment analysis on large scale Amazon product reviews //2018 IEEE international conference on innovative research and development (ICIRD). – IEEE, 2018. – С. 1-6. DOI: 10.1109/ICIRD.2018.8376299.

28. Xiao H., Rasul K., Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms //arXiv preprint arXiv:1708.07747. – 2017.

29. Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks //Advances in neural information processing systems. – 2012. – Т. 25. DOI: 10.1145/3065386.

30. He K. et al. Deep residual learning for image recognition //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – С. 770-778.

31. Sak H., Senior A., Beaufays F. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition //arXiv preprint arXiv:1402.1128. – 2014.

32. Kingma D. P., Ba J. Adam: A method for stochastic optimization //arXiv preprint arXiv:1412.6980. – 2014.

33. Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.

34. Maćkiewicz A., Ratajczak W. Principal components analysis (PCA) //Computers & Geosciences. – 1993. – Т. 19. – №. 3. – С. 303-342.

35. Gan Y., Yang J., Lai W. Video object forgery detection algorithm based on VGG-11 convolutional neural network //2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS). – IEEE, 2019. – С. 575-580. DOI: 10.1109/ICICAS48597.2019.00126.

36. Mannor S., Peleg D., Rubinstein R. The cross entropy method for classification //Proceedings of the 22nd international conference on Machine learning. – 2005. – С. 561-568.