САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Голядкин Максим Юрьевич

Выпускная квалификационная работа

Определение меры близости для автомобилей по наборам изображений с помощью свёрточных нейронных сетей

Уровень образования: бакалавриат
Направление 01.03.02 "Прикладная математика и информатика"
Основная образовательная программа СВ.5005. "Прикладная математика, фундаментальная информатика и программирование"

Научный руководитель:

старший преподаватель

Малинин К. А

Рецензент:

к.ф.-м.н., доцент

Козынченко В. А.

Санкт-Петербург
2019 г.

Содержание

1 Введение

В последнее десятилетие, в связи с ростом вычислительной мощности и количества доступных данных, большой подъём испытывает область машинного обучения - глубокое обучение. В этой области произошли значительные успехи в способности компьютеров понимать данные всевозможных форматов: изображения, речь, языки и так далее.

Алгоритмы машинного обучения, и глубокое обучение в частности, отличаются от других видов компьютерного программирования тем, что они преобразуют входные данные в выходные с помощью статистических правил, которые автоматически выводятся из большого набора примеров, а не задаются явно людьми. Исторически сложилось так, что для построения системы машинного обучения требовались знания предметной области, чтобы спроектировать функции, преобразовывающие необработанные данные в подходящие представления, из которых алгоритм обучения мог бы определять некоторые паттерны. Модели глубокого обучения, напротив, способны сами из необработанных данных выделять признаки, предназначенные для распознавания паттернов.

Модели глубокого изучения масштабируются до больших наборов данных, отчасти благодаря их способности работать на специализированном компьютерном оборудовании, и продолжают совершенствоваться с увеличением объема данных, позволяя им превосходить многие классические подходы машинного обучения. Наиболее распространенные модели обучаются с использованием обучения с учителем, в котором наборы данных состоят из точек входных данных (например, изображений собак) и соответствующих меток выходных данных (например, порода этой собаки).

Актуальность темы исследования обуславливается тем, что в настоящее время производится огромное количество данных, которые человек не может обработать самостоятельно, и методы интеллектуального анализа данных являются подходящим инструментом для решения этой проблемы.

2 Постановка задачи

Целью данной работы является построение некоторого отображения:

$$f : \mathcal{P}(A) \to \mathbb{R}^n,$$

где A - множество всевозможных изображений и $\mathcal{P}(A)$ - множество всех подмножеств множества A, которое сопоставляет набор изображений, содержащихся в объявлении о продаже некоторого автомобиля, некоторому вещественному вектору заданного размера n. Близость наборов изображений в этом пространстве по некоторой метрике будет говорить о том, что автомобили, расположенные на них, идентичны.

Так как в качестве инструмента были выбраны нейронные сети, то для достижения поставленной цели необходимо решить следующие задачи:

- Во-первых, необходимо изучить имеющиеся данные, оценить их качество, выявить некоторые паттерны и предложить способы их эксплуатации.

- Во-вторых, на основе произведённого анализа, предложить способы предобработки изображений для улучшения качества работы основной модели.

- В-третьих, представить архитектуру нейронной сети, которая представляет собой искомое отображение, и обучить её на предобработанных данных

- В-четвёртых, с помощью тестовой выборки оценить предложенные методы преобработки и проверить необходимость отдельных элементов нейронной сети. После чего выбрать наиболее оптимальную архитектуру с точки зрения как производительности и затрат памяти, так и качества работы.

3 Обзор литературы

Для получения базовых и продвинутых знаний о нейронных сетях и глубоком обучении была использована книга Goodfellow, Bengio, Courville. Deep Learning[12]. Будучи написанной тремя крупными учёными в данной области, она содержит подробные теоретические сведения о различных архитектурах нейронных сетей, методиках их обучениях, проблемах, возникающих при обучении, и способах их решения.

В получении знаний о практическом применении нейронных сетей оказалась полезной книга «Глубокое обучение. Погружение в мир нейронных сетей» за авторством С. Николенко. Данная работа содержит в себе как множество примеров задач, которые могут быть решены с помощью нейронных сетей, так и их программная реализация на языке Python. К тому же эта книга содержит более обширный обзор архитектур и методов обучения нейронных сетей, чем предыдущая, так как была написана на несколько лет позже, что является значительным сроком в столь быстроразвивающейся области.

Статья M.Wang, W.Deng. Deep Face Recognition: A Survey[28] содержит всесторонний обзор архитектур нейронных сетей и подходов к их обучению, которые позволяю получить наилучшие из достигнутых результаты для задачи повторной идентификации человека по изображению его лица. Хотя лицо человека и автомобиль объекты достаточно неблизкие, универсальность нейронных сетей позволяет переносить различные идеи от одной задачи к другой с сохранением их эффективности.

Из статьи Bag of Tricks and A Strong Baseline for Deep Person Re-identification [4] за авторством Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, Wei Jiang можно узнать о подходах обучения нейронных сетей, которые позволяют улучшить результат для большого спектра задач повторной идентификации.

Наконец, из работы L.Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay[26] были почерпнуты знания о выборе гиперпараметров для обучения нейронной сети, что является важнейшим этапом обучения нейронных сетей.

4 Глава 1. Теоретические сведения о нейронных сетях

4.1 Нейронные сети прямого распространения

Нейроннной сетью прямого распространения называется отображение:

$$f : X \times \Theta \to Y,$$

где X - пространство входов, Θ - пространство параметров (весов) и Y - пространство выходов. Данная функция f - является композицией некоторого числа функций $f^{(i)}$. То есть:

$$f(x, \theta) = f^{(n)}(f^{(n-1)}(...f^{(1)}(x, \theta_1), \theta_{n-1}), \theta_n).$$

В контексте нейронных сетей функция $f^{(i)}$ называется слоем сети. Слои $f^{(i)}, i = \overline{1, n-1}$ называются скрытыми слоями, а слой $f^{(n)}$ называется выходным слоем. Обучением нейронной сети называется нахождение набора параметров $\theta \in \Theta$, позволяющего наилучшим образом приблизить некоторую функцию f^* на некотором наборе данных, представляющем собой пары $(x, y), x \in X, y \in Y$, где:

$$f^* : X \to Y \tag{1}$$

Приведём пример. Пусть решается задачу классификации изображений. Тогда множеством X будет множество всевозможных изображений, множеством Y - множество возможных классов, а f^* - функция, ставящая в соответствие каждому изображению некоторый класс, определяющий, что находится на изображении.

Представленные нейронные сети называются нейронными сетями прямого распространения (feedforward), потому что информация, подающаяся на вход, проходит поочерёдно через каждый слой и выходы слоёв никаким образом не попадают обратно им на вход.

Количеством слоёв сети n называется глубина сети. Поэтому нейронные сети с большим количеством слоёв называются глубокими нейронными сетя-

ми, а область, изучающая их обучение, называется глубоким обучением.

Каждый слой нейронной сети прямого распространения состоит из нейронов. Каждый нейрон представляет собой линейный взвешенный сумматор, к выходу которого применяется некоторая нелинейная функция, называемая функцией активации. То есть выход i-го нейрона j-го слоя представляет собой:

$$h_{i,j} = g^{(j)}(\sum_{k=1}^{m_{j-1}} w_{i,j}^{k} h_{k,j-1} + b_{i,j}) = g^{(j)}(z_{i,j}), \tag{2}$$

где m_{j-1} - размерность выхода $(j-1)$-го слоя, $\boldsymbol{w_{i,j}}$ - вектор весов i-го нейрона j-го слоя, $b_{i,j}$ - смещение (также является параметром сети), $\boldsymbol{h_{j-1}}$ - вектор выхода $(j-1)$-го слоя, $g^{(j)}$ - функция активации j-го слоя.

Нелинейные функции необходимы для создания многослойных архитектур, потому что без них результат выходных нейронов представлял бы собой линейную комбинацию входных данных вне зависимости от количества слоёв (так как линейная комбинация линейных комбинаций некоторых элементов есть линейная комбинация этих элементов).

Для удобства $b_{i,j}$ обозначают как $w_{i,j}^{0}$ и $h_{0,j} = 1$ и получают запись в матричной форме:

$$\boldsymbol{h_j} = g^{(j)}(\boldsymbol{W_j^{\intercal}} \boldsymbol{x}),$$

где $\boldsymbol{W_j} \in \mathbb{R}^{(m_{j-1}+1)\times(m_j+1)}$.

Слой имеющий вид (2) называется полносвязным слоем.

Несколько лет назад в качестве функции активации как скрытых слоёв, так и выходного слоя использовали функции гиперболического тангенса и сигмоиду:

$$tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$sigmoid = \frac{1}{1 + e^{-x}}$$

В данный момент они не используются в качестве функции активации скрытых слоёв, но могут использоваться как функция активации выходного слоя.

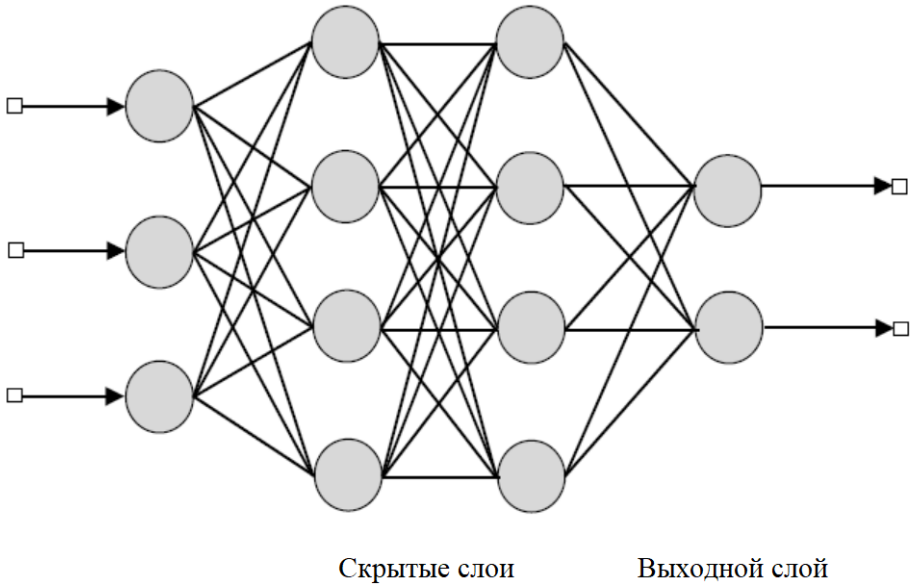

Рис. 1: Архитектура нейронной сети прямого распространения.

Вместо них используется rectified linear unit (ReLU):

$$ReLU(x) = max(0, x)$$

С одной стороны её производная, в отличие от ранее представленных функций, не требует вычислений, так как:

$$\frac{\partial ReLU}{\partial x} = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x < 0. \end{cases}$$

Это позволяет использовать нелинейность функции, при этом не затрудняя обучение. С другой стороны эта функция обеспечивает разреженность активаций (часть выходов слоя равна 0). Считается, что это позволяет нейронам чётче выделять отдельные признаки в данных, а не их комбинации, что улучшает итоговое качество работы модели. Предположение о полезности разреженности активаций получено из исследований мозга[22], предположения о работе которого легли в основу нейронных сетей. Согласно этим исследованиям в каждый момент времени лишь от 1 до 4% нейронов мозга активны, что соответствует отличию от нуля выходного значения нейрона сети.

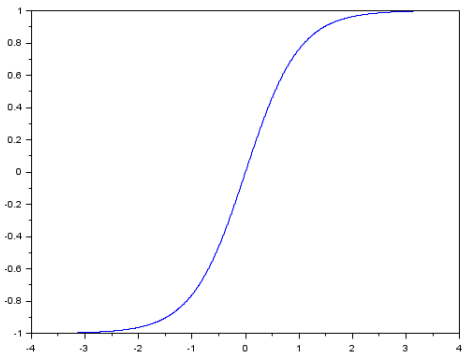

График гиперболического тангенса

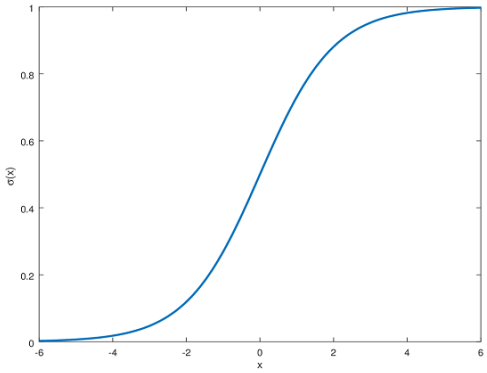

График сигмоиды

Рис. 2: Графики функций активаций.

Каждый слой нейронной сети прямого распространения по сути представляет собой нелинейное отображение пространства одной размерности в пространство другой размерности. Таким образом, входные данные постепенно отображаются из одного пространства в другое для того, чтобы на выходе из предпоследнего слоя получить вектор, называемый вектором признаков. После чего к нему применяется линейная регрессия и, возможно, некоторая нелинейная функция. Следовательно, при обучении первые $n-1$ слоёв сети пытаются выучить такое преобразование вектора входных данных в вектор признаков, чтобы последний слой мог наиболее качественно совершать соответствующие задаче предсказания, основываясь на полученном векторе признаков. Например, если у выходного слоя функцией активации является сигмоида и решаемой задачей является задача бинарной классификации, то скрытые слои сети пытаются выучить преобразование, отображающее входные данные в линейно-разделимое пространство (объекты различных классов разделимы гиперплоскостями), чтобы выходной слой, являющийся логистической регрессией, мог безошибочно осуществлять предсказания классов.

4.2 Обучение нейронных сетей прямого распространения

Обучение нейронных сетей основано на решении оптимизационной задачи. Мы минимизируем некоторую неотрицательную дифференцируемую функцию $J(\theta; X, y)$, называемой функцией потерь, которая принимает на вход выходы сети на тренировочном наборе данных и эталонные значения, которые сеть должна предсказывать, и возвращает скаляр. Чем сильнее отличаются предсказанные значения от эталонных, тем функция потерь должна быть больше, и наоборот. От выбора функции потерь напрямую зависит результат обучения. Значение функции потерь вычисляется для каждого элемента обучающей выборки, после чего берётся среднее арифметическое от этих значений и эта величина минимизируется. Значит искомыми параметрами сети являются те, которые доставляют минимум функции потерь на обучающей выборке:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} J(\theta; X, y)$$

Из-за того, что зачастую количество параметров сети составляет несколько миллионов, решить оптимизационную задачу в аналитическом виде не представляется возможным. Соответственно, для решения задачи минимизации используется итеративный метод градиентного спуска или его модификации. Он состоит в следующем:

1. Производится вычисление всех частных производных функции потерь по весам модели $\frac{\partial J}{\partial w_{i,j}^k}$

2. Значение весов изменяется в сторону антиградиента пропорционально некоторому значению η, называемому скоростью обучения:

$$w_{i,j}^k = w_{i,j}^k - \eta \frac{\partial J}{\partial w_{i,j}^k}$$

3. Данный процесс повторяется, пока функция потерь не перестанет уменьшаться. Тогда либо значение скорости обучения уменьшается и оптимизация продолжается, либо оптимизация останавливается и найденные

веса считаются оптимальными.

Для вычисления градиентов используется метод обратного распространения ошибки[25]. Каждую частную производную можно вычислить по формуле производной сложной функции:

$$\nabla_{\boldsymbol{w_{i,j}}} J = \frac{\partial J}{\partial h_{i,j}} \frac{\partial h_{i,j}}{\partial z_{i,j}} \nabla_{\boldsymbol{w_{i,j}}} z_{i,j} = \frac{\partial J}{\partial h_{i,j}} \frac{\partial h_{i,j}}{\partial z_{i,j}} \boldsymbol{h_{j-1}}, \quad j = \overline{1,n}, \quad i = \overline{1,m_j}$$

$$\frac{\partial J}{\partial h_{i,j}} = \sum_{k=1}^{m_{j+1}} \frac{\partial J}{\partial h_{k,j+1}} \frac{\partial h_{k,j+1}}{\partial z_{k,j+1}} \frac{\partial z_{k,j+1}}{\partial h_{i,j}} = \sum_{k=1}^{m_{j+1}} \frac{\partial J}{\partial h_{k,j+1}} \frac{\partial h_{k,j+1}}{\partial z_{k,j+1}} w^i_{k,j+1},$$
$$j = \overline{1,n-1}, \quad i = \overline{1,m_j}$$

Поочерёдное преобразование информации во время прямого прохода позволяет так же поочерёдно, слой за слоем, вычислять градиенты. То есть сначала вычисляются производные для весов последнего слоя. Далее, используя информацию полученную во время прямого прохода и производные полученные из последнего слоя, вычисляются частные производные для весов предпоследнего слоя и так далее. Следовательно, этот алгоритм позволяет вычислительно эффективно получать градиенты для всех слоёв сети.

На практике подсчёт градиента по всему обучающему множеству занимает слишком много времени и требует слишком много памяти. Для решения этой проблемы используется смесь обычного градиентного спуска и его стохастического варианта - градиентный спуск по мини-батчам. Мини-батч - это некоторое подмножество обучающего набора, размер которого по традиции принимается равным степени двойки. Например, 32 или 64. Этот метод как обладает плюсами стохастического спуска (быстрая скорость обучения и помощь в борьбе с локальными минимумами и седловыми точками), так и позволяет использовать преимущества матричной арифметики на видеокартах, что ещё сильнее ускоряет обучение.

Проход по всему набору данных называется эпохой.

В данной работе используется метод оптимизации Adam[20]. Он содержит в себе достоинства методов градиентного спуска с инерцией и адаптивных

методов градиентного спуска.

Метод градиентного спуска с инерцией можно понять, представив некоторый шарик, движущийся по ландшафту функции потерь. При движении шарик приобретает некоторую скорость, которую он не может мгновенно изменить. Это позволяет ему выбираться из точек локального минимума и седловых точек, от которых страдает обычный метод градиентного спуска. Инерция реализована с помощью экспоненциального скользящего среднего.

Таким образом, обновление параметров модели на шаге t происходит по формуле:

$$\boldsymbol{m}_{\boldsymbol{i,j}}^{\boldsymbol{t}} = \gamma \boldsymbol{m}_{\boldsymbol{i,j}}^{\boldsymbol{t-1}} + (1-\gamma)\nabla_{\boldsymbol{w}_{\boldsymbol{i,j}}} J$$

$$\boldsymbol{w}_{\boldsymbol{i,j}} = \boldsymbol{w}_{\boldsymbol{i,j}} - \eta \boldsymbol{m}_{\boldsymbol{i,j}}^{\boldsymbol{t}},$$

где параметр $\gamma \in [0; 1]$ как раз соответствует инерции.

Идея адаптивных методов градиентного спуска состоит в том, что единая скорость обучения для всех параметров сети может мешать обучению. Например, некоторые параметры могут быть близки к своему оптимальному значению, и их нужно изменять очень аккуратно (соответственно с меньшей скоростью обучения). Для корректировки скорости обучения используется экспоненциальное скользящее среднее значений квадратов градиентов,полученных на предыдущих шагах, соответствующих весов :

$$\boldsymbol{v}_{\boldsymbol{i,j}}^{\boldsymbol{t}} = \gamma \boldsymbol{v}_{\boldsymbol{i,j}}^{\boldsymbol{t-1}} + (1-\gamma)(\nabla_{\boldsymbol{w}_{\boldsymbol{i,j}}} J)^2$$

$$\boldsymbol{w}_{\boldsymbol{i,j}} = \boldsymbol{w}_{\boldsymbol{i,j}} - \frac{\eta}{\sqrt{\boldsymbol{v}_{\boldsymbol{i,j}}^{\boldsymbol{t}} + \epsilon}} \nabla_{\boldsymbol{w}_{\boldsymbol{i,j}}} J$$

Собирая все эти идеи вместе, получим шаг алгоритма Adam на шаге t:

1. Вычисляется экспоненциальное скользящее среднее градиента весов:

$$\boldsymbol{m}_{\boldsymbol{i,j}}^{\boldsymbol{t}} = \beta_1 \boldsymbol{m}_{\boldsymbol{i,j}}^{\boldsymbol{t-1}} + (1-\beta_1)\nabla_{\boldsymbol{w}_{\boldsymbol{i,j}}} J$$

2. Вычисляется экспоненциальное скользящее среднее квадрата градиента

весов:

$$\boldsymbol{v}_{\boldsymbol{i,j}}^{\boldsymbol{t}} = \beta_2 \boldsymbol{v}_{\boldsymbol{i,j}}^{\boldsymbol{t-1}} + (1 - \beta_2)(\nabla_{\boldsymbol{w}_{i,j}} J)^2$$

3. Из-за того, что экспоненциальные скользящие средние инициализируются нулями, значения $\boldsymbol{v}_{\boldsymbol{i,j}}^{\boldsymbol{t}}$ и $\boldsymbol{m}_{\boldsymbol{i,j}}^{\boldsymbol{t}}$ смещены к нулю в начале обучения. Чтобы это исправить проделывается следующий шаг:

$$\hat{\boldsymbol{m}}_{\boldsymbol{i,j}}^{\boldsymbol{t}} = \frac{\boldsymbol{m}_{\boldsymbol{i,j}}^{\boldsymbol{t}}}{1 - \beta_1^t}$$

$$\hat{\boldsymbol{v}}_{\boldsymbol{i,j}}^{\boldsymbol{t}} = \frac{\boldsymbol{v}_{\boldsymbol{i,j}}^{\boldsymbol{t}}}{1 - \beta_2^t}$$

4. Наконец, изменяются веса модели:

$$\boldsymbol{w}_{\boldsymbol{i,j}} = \boldsymbol{w}_{\boldsymbol{i,j}} - \frac{\eta \hat{\boldsymbol{m}}_{\boldsymbol{i,j}}^{\boldsymbol{t}}}{\sqrt{\hat{\boldsymbol{v}}_{\boldsymbol{i,j}}^{\boldsymbol{t}} + \epsilon}}$$

В оригинальной статье предложены следующие значения параметров: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$.

4.3 Трудности обучения нейронных сетей

Так как нам не доступны методы глобальной оптимизации для обучения нейронных сетей и мы используем локальные методы, то важным вопросом оптимизации является выбор начальной точки, то есть инициализация весов нейронной сети. Выбор неподходящей функции активации и неправильная инициализация весов может привести к проблеме затухающих градиентов. В работе [11] показано, что при прохождении сигнала по нейронной сети изменение абсолютных значений активаций и градиентов при обратном распространении ошибки зависит от значений весов. Если значения градиентов будут достаточно малы, то процесс обучения просто затормозится и нейронная сеть не сможет обучиться. В той же работе для борьбы с этой проблемой предложена схема инициализации весов слоёв с симметричной функцией активации с единичной производной в окрестности нуля, называемая инициализацией Ксавье. Предлагается инициализировать веса j-го слоя, семплируя из равномерного распределения на интервале:

$$[\frac{-\sqrt{6}}{\sqrt{m_j + m_{j+1}}}; \frac{\sqrt{6}}{\sqrt{m_j + m_{j+1}}}]$$

Проблема затухающих градиентов была частично решена использованием функции ReLU в скрытых слоях. Однако, инициализация Ксавье оказалась не совсем подходящей для этой функции активации, так как она не симметрична относительно нуля и имеет свойство обнулять часть активаций, в то время как в инициализации Ксавье использование величины, соответствующей количеству нейронов в слое, для определения границ равномерного распределения подразумевает, что все активации будут отличны от нуля. Поэтому в работе[9] была предложена инициализация весов слоёв с функцией активации ReLU, называемая инициализацией Хе. Авторы предалагают инициализировать веса j-го слоя, семплируя из нормального распределения со средним 0 и стандартным отклонением $\frac{2}{\sqrt{m_j}}$.

В данный момент использование функции ReLU в качестве функции активации скрытых слоёв, инициализированных с помощью инициализации Хе, и использование инициализации Ксавье для выходных слоёв позволяет избе-

жать проблемы затухающего градиента для не очень глубоких сетей.

Другой трудностью, возникающей при обучении, является проблема переобучения пере- в значении "сверх меры"). Каждая модель имеет некоторую ёмкость. Эта величина, определяемая архитектурой сети и количеством параметров, обозначает способность сети выделять признаки данных, которые определяют значение функции $f^*(1)$. Чем выше ёмкость сети, тем более сложные признаки она способна находить, и наоборот. Зачастую, если ёмкость сети больше, чем того требует задача, после некоторого этапа обучения нейронная сеть начинает выделять признаки, которые характерны только для тренировочного набора данных и которые ухудшают предсказания сети на ранее не виденных данных. Способность сети качественно работать на новых данных называется обобщающей способностью сети.

Для определения переобучения используется тестовая выборка данных. Эти данные не используются для обучения, и вычисление некоторой метрики качества помогает понять, улучшается ли обобщающая способность сети или нет.

Бороться с переобучением можно просто увеличивая количество обучающих данных, что на практике весьма часто невыгодно, поэтому используются подходы называемые методами регуляризации. Они могут заключаться в использовании специальных слоёв, добавлении регуляризатора (функция от весов сети) в функцию потерь или некотором преобразовании данных. В данной работе используется метод регуляризации, называемый аугментацией данных, который обычно применяется для изображений. Он заключается в том, чтобы совершать такие преобразования над изображениями, как зеркальное отображение, поворот изображения на случайное число градусов, случайное обрезание части изображения, преобразование яркости, контрастности, добавление шума, инвертирование цветов и многие другие. Это в некотором смысле позволяет увеличить количество тренировочных данных с помощью уже имеющихся изображений.

4.4 Сверточные сети

Сверточные сети были изобретены Яном Лекуном для работы с изображениями [13]. Этот подход заключает в себе 3 идеи:

1. Локальные рецепторные поля (каждый нейрон получает на вход только часть изображения);

2. Разделяемые веса (для большого количества соединений используется небольшое количество весов);

3. Субдискретизация (уменьшение пространственной размерности изображения).

Эти идеи позволяют сети обладать инвариантностью к масштабу, сдвигу, пространственным искажениям и улучшают обобщающую способность сети.

На вход такой сети подаётся многомерный массив формы ширина × высота × количество каналов. В случае, если изображение чёрно-белое, то количество каналов равно одному, а если цветное, то трём. На выходе свёрточная нейронная сеть в зависимости от задачи может выдавать как отдельное число, вектор, так и многомерный массив.

Рассмотрим отдельно слои, составляющие свёрточные нейронные сети.

- **Сверточный слой**.

  Веса свёрточного слоя представляются в виде многомерных массивов размером $\text{ширина}_1 \times \text{высота}_1 \times$ количество каналов предыдущего слоя, называемых ядрами.

  Результатом применения операции свёртки между входным многомерным массивом и одним ядром является обычная матрица. При применении всех ядер к входным данным мы получаем набор матриц, называемых картами признаков, которые при объединении на канальном уровне, дают выходной массив размера $\text{ширина}_2 \times \text{высота}_2 \times$ количество каналов данного слоя.

  Сама операция свёртки представляет собой проход ядра в качестве скользящего окна по входному изображению, где на каждом шаге веса ядра

скалярно умножаются на соответствующую им в данный момент часть входных данных и получившееся значение помещается в выходную матрицу, причём их расположение соответствует положению скользящего окна, при котором они были получены.

После этого к данной матрице применяется некоторая нелинейная функция. В подавляющем большинстве случаев этой функцией является ReLU или её модификации.

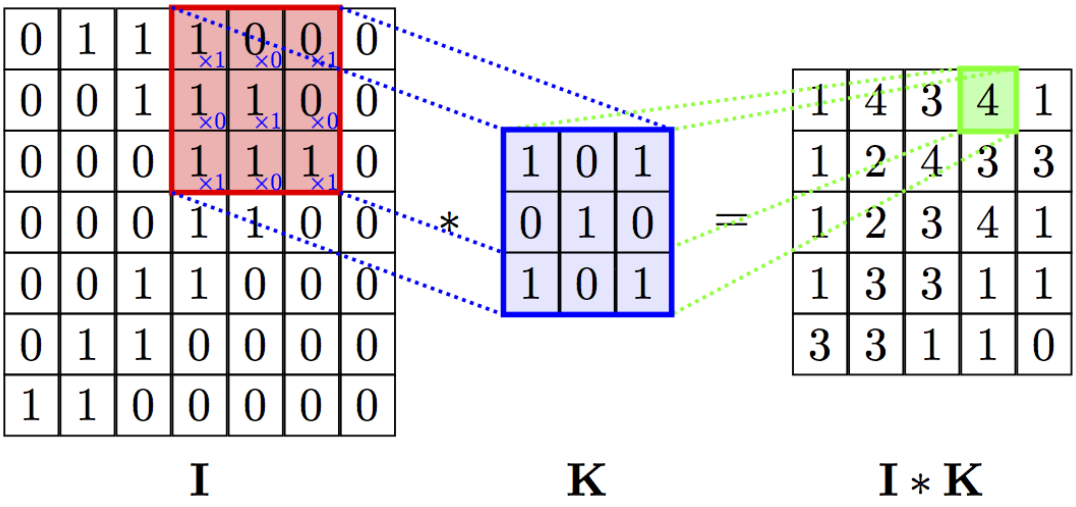

Рис. 3: Пример операции свёртки с одноканальными входными данными.

В общем случае, пространственные размеры ядра могут быть любыми, но наибольшую популярность имеют квадратные ядра 3×3 и в меньшей степени 5×5 и 7×7. Длина стороны выбирается нечётной, так как в этом случае у ядра есть центральный элемент и это позволяет сопоставить каждому значению входного массива некоторое значение карты признаков. Шаг скользящего окна обычно равен одному.

По определению операции свёртки видно, что пространственный размер карт признаков, получаемых на выходе, меньше пространственного размера входных карт признаков. Если этот эффект не желателен, то к краям входных карт признаков добавляют несколько рядов нулей в зависимости от размера ядра.

Во время обучения свёрточные слои, находящиеся в начале сети, учатся выделять различные линии и совокупности линий на изображении. Слои,

находящиеся в середине, учатся находить различные комбинации этих линий и выделять некоторые образы (признаки), а слои, расположенные в конце сети, уже выделяют более сложные образы, которые ищет свёрточная нейронная сеть, для того чтобы сделать некоторое предсказание.

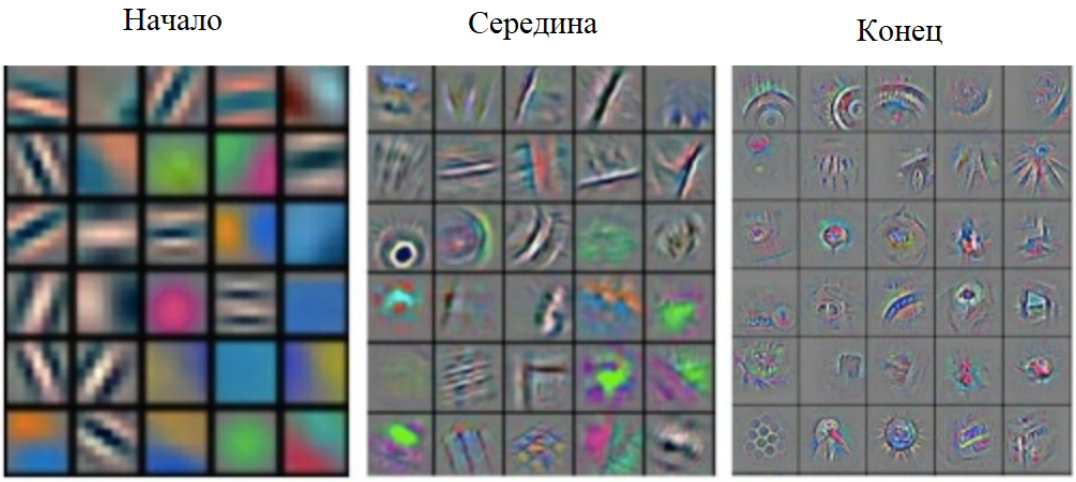

Рис. 4: Пример образов, которые выделяет свёрточная нейронная сеть[30]

- **Слой субдискретизации**. Данный слой не имеет каких-либо весов, но только представляет операцию субдискретизации (pooling). По каждой карте признаков проходит окно и на каждом шаге возвращает либо максимальный элемент из тех, что соответствуют данному положению окна (max-pooling), либо их среднее значение (average-pooling). Из полученных значений так же, как и при операции свёртки, составляется карта признаков, которые объединяются на канальном уровне. Следовательно, операция субдескретизации не изменяет количество карт признаков, но значительно уменьшает их пространственный размер, так как обычно размер скользящего окна равен двум и перемещается это окно с шагом 2, что уменьшает размер карт признаков в два раза.

  Это делается для того, чтобы увеличить рецептивное поле (часть изображения, информация о котором содержится в элементе карты признака) элементов карт признаков, а так же уменьшение размера помогает ускорить работу и снизить требования к памяти при обучении и использовании нейронной сети.

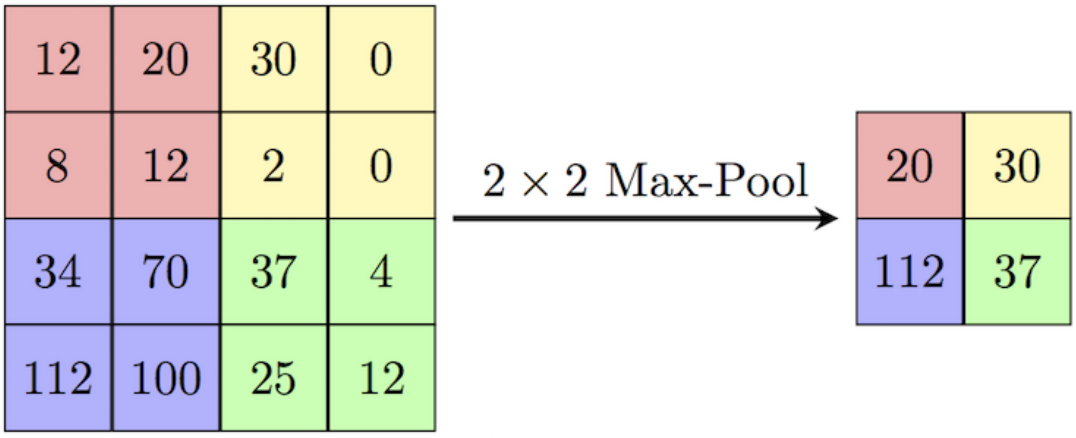

Рис. 5: Пример операции Max-pooling

В современных архитектурах слой субдискертизации может быть заменён на свёрточный слой с шагом 2.

- **Нормализация по мини-батчу[19]**. Данный слой направлен на решение проблемы внутреннего сдвига переменных (internal covariance shift). Чтобы объяснить это явление, рассмотрим двухслойную нейронную сеть. Первый слой получает данные из одного и того же распределения и учится работать с данными именно из этого распределения. Но следующий за ним слой получает результаты работы первого слоя, распределение которых меняется после каждого обновления весов. Значит, то, чему он научился на предыдущем шаге, становится бесполезным и ему необходимо обучаться заново. Получается, что пока первый слой достаточно не обучится и его веса перестанут значительно изменяться, второй слой не сможет обучиться чему-то полезному. Это сильно замедляет обучение и препятствует обучению глубоких архитектур.

  Для решения этой проблемы используется нормализация по мини-батчу. Если мы нормализуем входные сигналы каждого слоя, то все слои смогут учиться одновременно. Однако, вычисление среднего и дисперсии всей обучающей выборки после каждого обновления весов вычислительно неэффективно, поэтому эти статистики считаются по мини-батчам, которые поступают сети во время обучения. Обычно нормализация по мини-батчам применяется перед применением функции активации к кар-

там признаков.

Алгоритм нормализации по мини-батчу для изображений выглядит так:

1. Вычисление среднего значения для каждой карты признака:

$$\mu_l = \frac{1}{bwh} \sum_{k=1}^{b} \sum_{i=1}^{w} \sum_{j=1}^{h} x_{i,j,l}^{k}, \quad l = \overline{1, C},$$

   где b - размер мини-батча, C - количество карт признаков, $x_{i,j,l}^{k}$ - элемент i-й строки j-го столбца l-ой карты признака k-го объекта мини-батча, w и h - пространственные размеры карт признаков.

2. Вычесление дисперсии для каждой карты признака:

$$\sigma_l^2 = \frac{1}{bwh} \sum_{k=1}^{b} \sum_{i=1}^{w} \sum_{j=1}^{h} (x_{i,j,l}^{k} - \mu_l)^2, \quad l = \overline{1, C},$$

3. Нормализация карт признаков:

$$\hat{x}_{i,j,l}^{k} = \frac{x_{i,j,l}^{k} - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}}$$

4. Изменение масштаба и сдвиг:

$$y^k = \gamma \hat{X}^k + \beta$$

   Параметры γ и β - векторы размерности C, которые позволяют операции нормализации не уменьшать ёмкость сети. То есть, если в некотором месте нормализация не нужна, то они могут обучиться таким значениям, что операция нормализации по мини-батчу будет просто тождественным отображением.

   Авторы работы[16] попытались выяснить, как нормализация по мини-батчам помогает процессу оптимизации и пришли к выводу, что с одной стороны улучшение сходимости никак не связано с решением проблемы внутреннего сдвига переменных, а с другой стороны нор-

мализация по мини-батчам даже не решает проблему внутреннего сдвига переменных. В свою очередь авторы сделали предположение, что нормализация по мини-батчам делает ландшафт функции потерь более гладким, что упрощает процесс нахождения минимума.

Таким образом, классические свёрточные нейронные сети состоят из последовательностей свёрточных слоёв, между которыми находятся слои субдискретизации, уменьшающие пространственный размер карт признаков. После какого-то слоя карта признаков превращается в вектор, называемый вектором признаков, который подаётся на вход одному или нескольким полносвязным слоям, совершающим предсказание, соответствующее решаемой задаче.

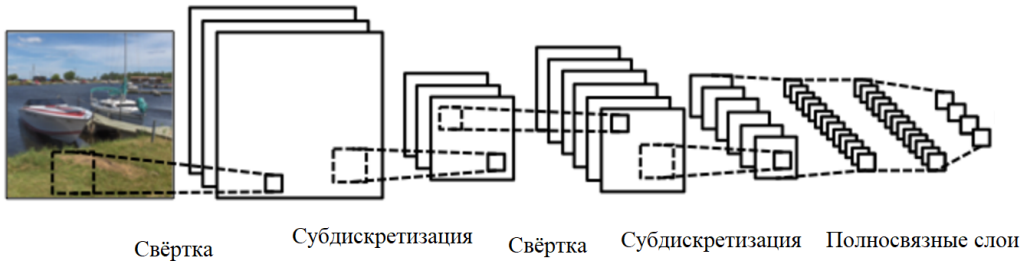

Рис. 6: Пример работы свёрточной нейронной сети.

4.5 Перенос обучения

Переносом обучения (transfer learning) называется использование весов модели, обученной для решения задачи P_1 на наборе данных D_1, в целях решения задачи P_2 на наборе данных D_2. Эта техника полезна при работе со свёрточными нейронными сетями, так как свёрточные слои в начале сети обучаются нахождению низкоуровневых признаков, которые едины для различных визуальных объектов. То есть, чем более похожи наборы данных D_1 и D_2, тем лучше веса сети подходят для решения задачи P_2. Например, если D_1 представляет собой изображения собак и кошек, а D_2 - изображения коров и овец, то даже признаки извлекаемые свёрточными слоями из середины сети будут полезны, но если D_2 представляет собой набор МРТ-снимков, то полезными могут быть только самые низкоуровневые признаки, однако веса предобученной сети можно использовать как хорошую инициализацию.

Использование переноса обучения как просто ускоряет обучение, так и может быть полезно в случае если набор данных D_1 содержит большое количество изображений, когда D_2 имеет недостаточное количество данных для качественного обучения. Тогда сначала обучив нейронную сеть на D_1 задаче P_1 похожей на задачу P_2, а потом дообучив сеть на наборе данных D_2 задаче P_2, можно получить качественное решение.

Задачи P_1 и P_2 чаще всего различны, поэтому принято отбрасывать полносвязные слои, которые решали по картам признаков задачу P_1 и заменять их новыми полносвязными слоями, соответствующими задаче P_2.

Алгоритм переноса обучения:

1. Заменить полносвязные слои, инициализировав новые подходящим способом.

2. Несколько эпох обучать нейронную сеть, изменяя веса только полносвязных слоёв и заморозив остальные.

3. Обучать всю нейронную сеть до сходимости.

4.6 Distance Metric Learning

Distance Metric Learning - это подход к созданию архитектур и обучению нейронных сетей для решения задачи верификации (определение для пары объектов принадлежности к одному классу) и повторный идентификации (предсказание класса объекта при помощи других объектов). Примером таких задач может служить задача верификация голоса или повторная идентификация целовека по лицу.

Суть подхода состоит в определении некоторой метрики ρ в пространстве входных данных X:

$$\rho(x, y) = d(f(x), f(y)), \quad x \in X, \quad y \in X$$

$$f : X \to \mathbb{R}^n,$$

где функция f является нейронной сетью, отображающей исходный объект в некоторое пространство признаков, в котором близость объектов по определённой в этом пространстве метрике d (зачастую это либо евклидово, либо косинусное расстояние) говорит об их сущностной близости (на картинке изображен один и тот же объект, или голос на звуковых записях принадлежит одному и тому же человеку).

Задача повторной идентификации близка к задаче многоклассовой классификации, но при этом количество классов, которые могут быть распознаны, не ограничено количеством классов тренировочных данных, что позволяет изменять их число во время эксплуатации модели.

5 Глава 2. Подход к решению задачи

5.1 Анализ данных

Имеющиеся данные представлены некоторым набором сущностей "автомобиль", которому соответствуют одно или несколько объявлений о продаже одного и того же автомобиля на сайтах **auto.ru**, **avito.ru** и **drom.ru**.

Каждому объявлению соответствует набор характеристик автомобиля, размещённых в этом объявлении, и некоторое количество изображений, на которых присутствует как интерьер и экстерьер автомобиля, так и некоторые другие объекты, косвенно относящиеся к автомобилю.

Цель решаемой задачи состоит в том, чтобы для новопоступающего объявления определить - принадлежит ли оно к одной из ранее известных сущностей "автомобиль" или представляет новую сущность.

Для большинства сущностей характерно следующее: наборы изображений объявлений либо совпадают, либо пересекаются между собой, причём изображения совпадают не попиксельно, так как разрешения изображений, полученных с разных сайтов, различно. Ранее представленные сервисы по разному обрабатывают получаемые изображения, поэтому на краях изначально одинаковые изображения могут отличаться.Так же следует заметить, что сервисы помечают изображения водяными знаками и закрывают номера автомобилей, что увеличивает их попиксельное различие.

По случайной выборке в 12000 изображений было определено, что соотношение количества изображений, на которых находится целостный экстерьер автомобиля к количеству остальных изображений равно 55:45. В связи с этим был сделан выбор об использовании только тех изображений, на которых виден весь автомобиль с некоторого угла обзора, так как для подхода Distance Metric Learning желательно, чтобы объекты, содержащиеся во входных данных, были однородны, то есть требование того, чтобы изображения автомобиля и его радиаторной решётки находились бы близко в пространстве признаков, может оказаться не лучшей идеей.

5.2 Алгоритм решения задачи

Ко всем объявлениям применяется следующая последовательность действий:

1. Для изображений определяется - находится ли на них экстерьер автомобиля, и те, на которых он изображен, используются далее.

2. Совершается некоторая предобработка изображений, направленная на улучшение качества работы основной модели.

3. Изображения по отдельности подаются на вход модели, реализующей подход Distance Metric Learning, в результате чего они трансформируются в набор векторов.

4. Этот набор векторов усредняется, в результате чего мы получаем вектор, соответствующий объявлению.

Для новопоступивших объявлений применяется этот же алгоритм, после чего вычисляется расстояние в пространстве признаков между новым объявлением и каждым из уже имеющихся объявлений, соответствующих той же модели автомобиля, что и новопоступившее. По получившимся расстояниям определяется, принадлежат ли объявления к одной и той же сущности "автомобиль" или нет.

В качестве предобработки изображения в данной работе рассматриваются:

- Детектирование автомобиля на картинке, то есть нахождение окаймляющего автомобиль прямоугольника, и использование только этой части изображения в дальнейшем.

- Сегментация изображения, то есть попиксельное выделение области изображения, на которой находится автомобиль, с последующим перекрытием шумом части изображения, на которой автомобиля нет, и обрезанием краёв изображения, содержащих только шум.

Смысл предобработки заключается в том, чтобы выделить главный объект на изображении, однако в данной задаче фон может содержать полезную информацию о месте, где была сделана фотография, так что вывод о том, какая

предобработка предпочтительнее и нужна ли она вообще, сделан в главе 3 с помощью экспериментов и сравнительного анализа.

5.3 Архитектура нейронных сетей

Пункты 1-3 предложенного алгоритма реализуются с помощью нейронных сетей. В каждом пункте для извлечения признаков использовалась свёрточная нейронная сеть ResNet18[8]. Ключевой особенностью этого семейства архитектур является использование остаточных соединений (residual connections), которые позволяют передавать информацию не только между соседними слоями, что способствует распространению градиента в методе обратном распространении ошибки. Существует несколько объяснений эффективности ResNet, однако важно то, что данная архитектура показывает отличные результаты на большинстве задач компьютерного зрения. В качестве начальных весов использовались веса сети, обученной для задачи классификации на наборе данных ImageNet[18].

Рассмотрим архитектуры для решения каждой из задач:

1. **Задача классификации**.

   Для задачи классификации вектор признаков подавался на вход полносвязному слою, выходом которого являлось одно значение, к которому применялась сигмоидальная функция активации. Значения данной функции лежат на интервале $[0, 1]$, что позволяет интерпретировать выходное значение как вероятность того, что на изображении находится автомобиль.

2. **Задача детектирования**.

   При решении задачи детектирования вектор признаков подаётся полносвязному слою с 4 значениями на выходе:

   - x_1 - абсцисса левого верхнего угла ограничивающего прямоугольника;
   - y_1 - ордината левого верхнего угла ограничивающего прямоугольника;
   - x_2 - абсцисса правого нижнего угла ограничивающего прямоугольника;

- y_2 - ордината правого нижнего угла ограничивающего прямоугольника.

К полученным значениям применяется сигмоидальная функция активации для получения интерпретируемых значений.

3. **Задача сегментации**.

Для решения задачи сегментации использовалась UNet[24]-подобная архитектура называемая TernausNet[17].

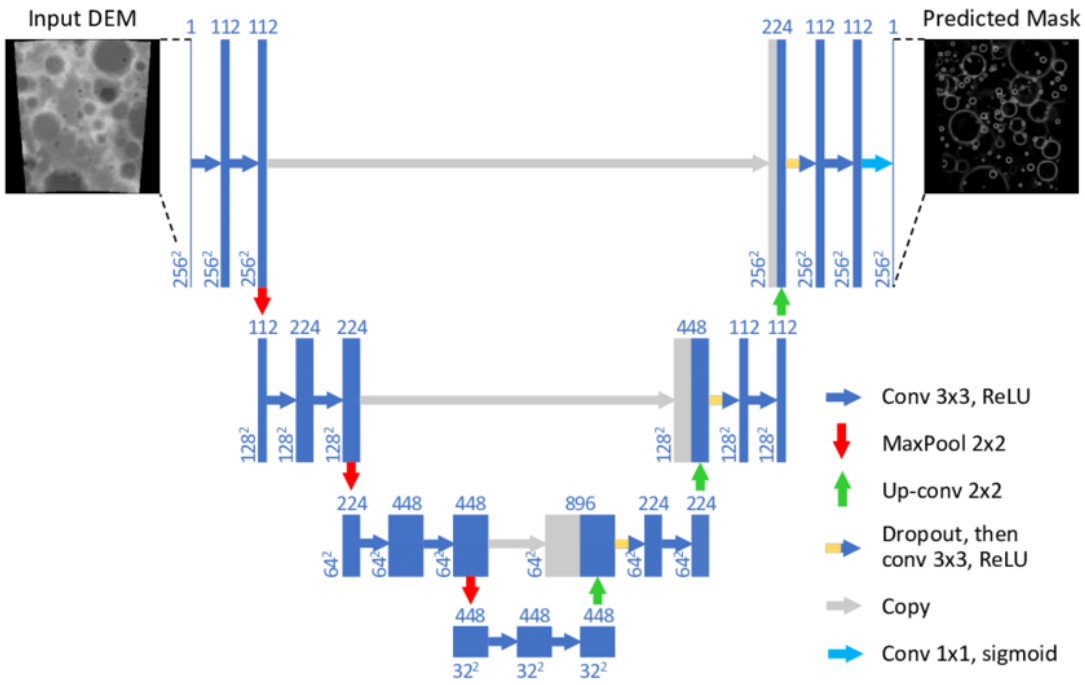

Рис. 7: Архитектура UNet.

Данная архитектура состоит из 2 частей: кодировщика (слева на изображении) и декодировщика (справа). Первая часть превращает входное изображение в карту признаков (некоторая матрица), а декодировщик восстанавливает её до размеров исходного изображения с помощью интерполяции и сверточных слоёв.

Ключевой особенностью UNet является связь кодировщика и декодировщика посредством конткатенации карт признаков, получаемых из промежуточных слоёв кодировщика, и карт признаков декодировщика соответствующего размера. К выходу сети, которым является матрица такого же

размера, как и входное изображение, применяется сигмоидальная функция активации, что позволяет определить значение каждого элемента матрицы, как вероятность принадлежности соответствующего пикселя к искомому объекту. При бинаризации при помощи некоторого порога (например, 0.6) получается искомая маска.

Главным отличием TernausNet от UNet является то, что в качестве кодировщика используется некоторая предобученная свёрточная нейронная сеть (в нашем случае ResNet-18), что позволяет получать более качественные результаты на меньшем количестве тренировочных данных.

4. **Задача верификации.** В данном случае вектор признаков, извлекаемый ResNet18 подаётся на вход слою нормализации по мини-батчу, у которого параметр β равен нулю, после чего нормализованный вектор признаков поступает на вход полносвязному слою, у которого параметр смещения равен нулю и выходом которого является вектор длиной, равной количеству сущностей "автомобиль" в тренировочном множестве.

   Агрегация набора векторов для получения вектора объявления происходит либо простым взятием среднего арифметического, либо используя блок, предложенный в статье[29]. В данной работе набор векторов усредняется с помощью взвешенного среднего арифметического, веса которого предсказываются отдельной нейронной сетью.Их величина зависит от качества изображения и его информативности.

   Получение предсказания о принадлежности двух объявлений к одной сущности осуществляется с помощью определения порога. Если расстояние между двумя объявлениями в пространстве признаков меньше или больше (в зависимости от меры) некоторого порога, то считается, что объекты принадлежат к одной сущности, в ином случае считается, что объявления относятся к разным сущностям.

6 Глава 3. Реализация и результаты

6.1 Подготовка данных

Так как большинство изображений имеет различный размер, то все они изменены в размерах до разрешения 224 × 224, удобного для ResNet18, с помощью библиотеки **OpenCV**[6].

- Для решения задачи классификации было размечено 12000 фотографий с помощью приложения **LabelMe**[21], которые были разделены на обучающую и тестовую выборку (8500 и 3500, соответственно). В добавок для обучающей выборки из набора данных ImageNet было получено 7000 изображений велосипедов, поездов и других транспортных средств, не являющихся автомобилями, а из набора данных **Cars**[2] взято 11000 изображений автомобилей.

Рис. 8: Пример изображений, содержащих автомобиль.

Рис. 9: Пример изображений, не содержащих автомобиль.

- Для решения задачи детектирования и сегментации случайным образом было выбрано 500 изображений, которые были размечены с помощью онлайн-приложения Supervisely[27] и разделены на обучающую и тестовую выборку(400 и 100, соответственно). В добавок для задачи детекти-

рования в обучающую выборку были добавлены изображения из набора данных Cars, для которых существует соответствующая разметка, а для задачи сегментации в обучающую выборку был добавлен набор данных Carvana[1], содержащий 5000 изображений.

Рис. 10: Пример изображений с окаймляющей рамкой.

Изображение Маска

Рис. 11: Пример изображения и соответствующей ему сегментационной маски.

- Данные для задачи верификации были получены после обучения ранее упомянутых сетей и применении их к наборам изображений соответствующих объявлениям следующим образом:

  1. Изменение размеров изображений набора до 224×224.
  2. Отбор изображений путём классификации.
  3. Предсказание окаймляющих рамок или сегментационных масок.

4. Перевод размеров рамок и сегментационных масок до размеров оригинальных изображений.
5. Вырезание соответствующих частей из оригинальных изображений и изменение их размеров до 224×224.

Обучающая выборка состоит из 83000 объявлений, часть которых принадлежит к одинаковым сущностям. Количество различных моделей автомобилей равно 1500.

Тестовая выборка состоит из 13000 объявлений, часть которых принадлежит к одинаковым сущностям. Количество различных моделей автомобилей равно 3.

6.2 Обучение моделей

Обучение и эксплуатация моделей реализованы с помошью фреймворка **PyTorch**[3] для языка **Python**.

- Для классификации в качестве функции потерь использовалась бинарная перекрестная энтропия:

$$H(y, \hat{y}) = -y \log \hat{y} - \hat{y} \log y,$$

где y - истинное значение, $\hat{y}$ - предсказание сети.

Обучение сети происходило в течение 30 эпох с размером мини-батча 64 при изменении скорости обучения оптимизатора Adam с помощью косинусного отжига[23] от $\eta_{max} = 10^{-3}$ до $\eta_{min} = 10^{-5}$:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + \cos(\frac{T_{cur}}{T_{max}}\pi)),$$

где T_{cur} - номер эпохи, T_{max} - общее количество эпох, η_t - скорость обучения на эпохе T_cur.

В качестве аугментации данных использовались: отзеркаливание отображения, случайный поворот с последующим обрезанием краёв, случайное изменение яркости и контрастности, размытие изображения с помощью медианного фильтра и добавление гауссовского шума.

Для оценки качества работы использовалась метрика точность (процент правильно предсказанных примеров).

- Для задачи детектирования в качестве функции потерь использовалась сумма среднеквадратичных ошибок для каждой координаты. Обучение сети происходило в течение 35 эпох с размером мини-батча 32 при изменении скорости обучения с помощью косинусного отжига от $\eta_{max} = 10^{-3}$ до $\eta_{min} = 10^{-5}$.

Для аугментации данных использовались: отзеркаливание отображения, случайное обрезание изображения с частичным сохранением окаймля-

ющей рамки, случайное изменение яркости и контрастности, размытие изображения с помощью медианного фильтра и добавление гауссовского шума.

В качестве метрики качества использовалась Intersection over union:

$$IoU = \frac{S_I}{S_U},$$

где S_I - площадь пересечения правильной ограничивающей рамки и предсказанной, S_U - площадь объединения правильной и предсказанной рамок.

- Для обучения нейронной сети, решающей задачу сегментации, в качестве функции потерь в течение первых 20 эпох использовалась бинарная перекрестная энтропия, а в оставшиеся эпохи применялась функция потерь Ловаса[5].

  Обучение сети происходило в течение 30 эпох с размером мини-батча 16 при изменении скорости обучения с помощью косинусного отжига от $\eta_{max} = 10^{-4}$ до $\eta_{min} = 10^{-6}$.

  Для регуляризации были применены следующие аугментации данных: отзеркаливание отображения, случайное обрезание изображения, случайное изменение яркости и контрастности и добавление гауссовского шума.

  В качестве метрики качества использовался индекс Дайса:

$$dice = \frac{2S_I}{S_T + S_P},$$

  где S_T - площадь правильной маски, S_P - площадь предсказанной маски, S_I - площадь их пересечения.

- Во время обучения модели для решения задачи верификации мини-батчи формировались особым образом, подобно способу, изложенному в работе[14]:

  1. Случайным образом выбирается некоторое количество сущностей (в данной работе это значение равно 16).

2. Для каждой сущности выбирается 3 соответствующих ей объявления. Если ей соответствует 2 объявления, то третье выбирается случайном образом из первых двух. Сущности, содержащие одно объявление, не используются.
3. Для каждого объявления случайным образом выбирается изображение из набора изображений.
4. Все эти изображения объединяются в один мини-батч.

Таким образом, мини-батч состоит из 48 изображений, где для каждого изображения в мини-батче находится 2 изображения одной с ним сущности (расстояние между ними в пространстве признаков должно быть мало), и 45 изображений, принадлежащих к другим сущностям (расстояние между ними в пространстве признаков должно быть велико).

Для достижения требуемых свойств от пространства признаков используется модификация обычного Triplet loss[15] - Batch Hard Triplet Loss[14], упрощающий поиск сложных примеров, способствующих обучению сети:

$$BHTL = \sum_{i=1}^{P} \sum_{a=1}^{K} \ln(1 + exp(D_a^i)),$$

$$D_a^i = \max_{p=1\ldots K} d(f(x_a^i), f(x_p^i)) - \min_{\substack{j=1\ldots P \\ n=1\ldots K \\ j \neq i}} d(f(x_a^i), f(x_n^j)),$$

где x_i^a - изображение a-го объявления i-ой сущности, f - свёрточная нейронная сеть, d - метрика в пространстве признаков (в данном случае евклидово расстояние), $P = 16$ - количество сущностей, $K = 3$ - количество изображений, соответствующих каждой сущности.

Использование этой функции потерь помогает добиться того, чтобы расстояние в пространстве признаков между изображениями одной сущности было меньше, чем расстояние между изображениями разных сущностей.

Однако, согласно экспериментам[4], использование косинусного расстояние в качестве метрики пространства признаков улучшает качество ра-

боты сети, поэтому используется ещё одна функция потерь Large Margin Cosine Loss[7], являющаяся модификацией перекрестной энтропии, применённой к значению функции Softmax от выхода полносвязного слоя:

$$LMCL = \frac{1}{N}\sum_{i} -log\frac{e^{s(cos(\theta_{y_i},i)-m)}}{e^{s(cos(\theta_{yi},i)-m)} + \sum_{j\neq y_i} e^{s\ cos(\theta_j,i)}},$$

$$W = \frac{W^*}{\|W^*\|},$$

$$x = \frac{x^*}{\|x^*\|},$$

$$cos(\theta_j, i) = W_j^T x_i,$$

где W^* - матрица весов полносвязного слоя, x^* - вектор признаков, s - некоторая константа, улучшающая распространение градиента по сети во время обучения (в данной работе равна 16), m - константа, увеличивающая косинусное расстояние между сущностями в пространстве признаков (в данной работе равна 0.4).

Данная функция потерь способствует разделению сущностей в пространстве признаков с точки зрения косинусного расстояния.

Для обоснования полезности элементов архитектуры предложенной нейронной сети, решающей задачу верификации, было обучено несколько вариантов моделей:

1. Модель, в которой используются только вектора признаков. Обучается с помощью Batch Hard Triplet loss[14], которому на вход подаются эти вектора. Метрикой расстояния в пространстве признаков является евклидово расстояние.
2. Модель, в которой используются и векторы признаков, и предсказания о принадлежности к сущностям "автомобиль", полученные с помощью полносвязного слоя. Обучается с помощью Batch Hard Triplet loss, которому на вход подаются векторы признаков, и перекрестной

энтропии, которой на вход подается выход полносвязного слоя, к которому была применена функция Softmax:

$$\sigma(\mathbf{z}) = \left(\frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}} \right)_{i=1}^{n},$$

где $\mathbf{z}$ - вектор размерности n. Метрикой расстояния в пространстве признаков является косинусное расстояние между нормализованными векторами признаков.

3. Модель, в которой используются и векторы признаков, и предсказания о преинадлежности к сущностям "автомобиль", полученные с помощью полносвязного слоя. Обучается с помощью Batch Hard Triplet loss, которому на вход подаются векторы признаков, и Large Margin Cosine Loss, которому на вход подается выход полносвязного слоя. Метрикой расстояния в пространстве признаков является косинусное расстояние между нормализованными векторами признаков.

Обучение сети происходило в течение 120 эпох со следующими значениями скорости обучения:

$$\eta = \begin{cases} 3.5 \times 10^{-5} \times \frac{T}{10} & \text{if } T \leq 10, \\ 3.5 \times 10^{-4} & \text{if } 10 < T \leq 40, \\ 3.5 \times 10^{-5} & \text{if } 40 < T \leq 70, \\ 3.5 \times 10^{-6} & \text{if } 70 < T \leq 120, \end{cases}$$

где T - номер эпохи.

Для регуляризации были применены следующие аугментации данных: отзеркаливание отображения, случайное обрезание изображения, случайное вырезание частей изображения[10].

Для оценки качества модели был составлен набор данных, состоящий из всевозможных пар объявлений одной модели автомобиля.

В качестве метрики качества используется полнота (recall) и частота

неправильных срабатываний (FAR):

$$recall = \frac{TA}{PP},$$

где TA - множество пар из набора данных, для которых было правильно определено, что они относятся к одной сущности, PP - множество пар объявлений, относящихся к одной сущности.

$$FAR = \frac{FA}{NP},$$

где FA - множество пар из набора данных, для которых было неправильно определено, что они относятся к одной сущности, NP - множество пар объявлений, относящихся к разным сущностям.

Очевидно, что при изменении порога обе метрики будут либо увеличиваться, либо уменьшаться. Однако об улучшении решения свидетельствует как увеличение первой метрики, так и уменьшение второй. Поэтому для оценки результата необходимо сравнивать значения одной метрики, при одинаковых значениях другой.

6.3 Результаты

- Для задачи классификации на тестовой выборке размером 3500 изображений получена точность **98.5%**.

- Для задачи локализации на тестовой выборке из 100 изображений получено значение метрики Intersection over union равное **0.945**.

- Для задачи сегментации на тестовой выборке из 100 изображений получено значение индекса Дайса равное **0.975**.

- Для задачи верификации к данным была применена обработка, основанная на детектировании.

FAR / Модель	10^{-4}	10^{-5}
Модель 1	0.85	0.705
Модель 2	0.878	0.701
Модель 3	0.893	0.703

Таблица 1: Значения полноты для различных моделей и FAR.

Из таблицы видно, что все части предложенной ранее архитектуры улучшают качество работы модели.

Способ для получения вектора объявления путём взятия взвешенного среднего арифметического, предложенный в главе 2, лишь ухудшил результаты модели. Возможно, это связано с тем, что использование сразу нескольких изображений, соответствующих одному объявлению, сильно упростило процесс обучения, что ухудшило обобщающую способность сети. Поэтому было решено использовать обычное среднее арифметическое.

Далее итоговая модель была использована для оценки влияния предобработки на результат.

FAR / Предобработка	10^{-4}	10^{-5}
Без предобработки	0.873	0.658
Детектирование	0.893	0.703
Сегментация	0.836	0.693

Таблица 2: Значения полноты для различных вариантов предобработки и FAR.

Из полученных результатов можно сделать следующие выводы:

1. Использование предобработки детектированием позволяет получить наилучшие результаты. Это связано с тем, что изображения, полученные из различных источников, могут содержать разное количество фона. Из-за этого изменение размера изображения может по разному изменить форму автомобиля, что затруднит верификацию. При этом детектирование не отбрасывает весь фон.
2. На данных, предобработанных сегментацией, модель работает хуже, чем на данных без предобработки. Это объясняется тем, что автомобили могут быть очень похожи друг на друга и использование фона позволяет отличить одно изображение от другого.

Значит, наилучшие результаты показывает модель, обученная на данных, предобработанных детектированием, с помощью Batch Hard Triplet Loss и Large Margin Cosine Loss. В качестве пространства признаков используется пространство нормализованных векторов признаков и метрикой в этом пространстве является косинусное расстояние. Для получения вектора объявления из набора векторов признаков изображений, соответствующих этому объявлению, используется среднее арифметическое.

Полученная модель зачастую ошибается в следующих случаях:

1. Наборы изображений объявлений не пересекаются и фотографии сделаны в разных местах или в разное время (из-за этого освещение и вид автомобиля может различаться). Эта ситуация представляет наибольшую трудность, так как даже человек не всегда может от-

личить два похожих автомобиля друг от друга, поэтому эту ошибку можно считать неустранимой.

2. Наборы изображений пересекаются, но один из наборов содержит фотографии из одного места, а другой из разных. Учитывая, что вектор признаков содержит в себе информацию как о внешнем виде автомобиля, так и о фоне, можно понять, что наличие разных фонов в наборе изображений, отдаляет в пространстве признаков этот набор от набора с одним фоном.

3. Наборы изображений малы (1-3 изображения) и содержат фотографии, сделанные из необычных ракурсов. Эти изображения могут находиться поодаль в пространстве признаков от основной массы изображений, соответствующих этому объявлению. Из-за этого выброса его среднее арифметическое сильно изменяется и расстояние в пространстве признаков между объявлениями увеличивается. Это можно исправить введя более сложную систему агрегации.

4. Автомобили сильно похожи и фон похож. Степень похожести, необходимая для совершения ошибки, определяется выбором порога верификации.Основную трудность вызывают изображения автомобилей на снегу. Эту проблему можно решить используя более сложную архитектуру нейронной сети, что повлечёт за собой ужесточение требований к оборудованию.

6.4 Оценка быстродействия

Для оценки быстродействия из набора данных была сделана выборка в количестве 10000 объявлений. К ней был применён алгоритм из параграфа 5.2 и измерена скорость выполнения каждого шага (шаг 3 и шаг 4 были объединены), причём время чтения с жёсткого диска не учитывалось. Вычисления производились на процессоре Intel Core i5-8300H и на видеокарте NVIDIA GeForce GTX 1060, и были получены следующие результаты:

Оборудование / Шаг	GPU	CPU
Шаг 1	0.019	0.172
Шаг 2	0.014	0.095
Шаги 3 и 4	0.005	0.093

Таблица 3: Среднее время(в секундах) для одного объявления.

Также была измерена скорость вычисления расстояния между набором объявлений и новым объявлением посредством умножения матрицы на вектор с помощью библиотеки NumPy для языка Python. Размер вектора равен 512.

Количество объявлений	$5 * 10^4$	10^5	$2.5 * 10^5$	$5 * 10^5$
Время выполнения	0.014	0.026	0.066	0.131

Таблица 4: Среднее время умножения (в секундах).

7 Выводы

После анализов результатов становится ясно, что свёрточные нейронные сети способны показывать достойные результаты на предложенном наборе данных при решении задач классификации, детектирования, сегментации и верификации. При этом предложенная модель способна работать как с данными с предобработкой, так и без неё, но важно помнить, что выбор предобработки способен как улучшить качество работы нейронной сети, так и ухудшить его.

Улучшение результата возможно при изменении архитектуры сети на более сложную, что приведёт к увеличению требований к памяти и производительности. Однако, это улучшение может быть лишь небольшим ввиду особенностей данных.

8 Заключение

В данной работе был собран, проанализирован и частично размечен набор данных, содержащий изображения автомобилей. На нём были обучены нейронные сети для решения задач классификации, детектирования, сегментации и верификации. На основе этим сетей были предложены способы предобработки изображений. С помощью тестового набора данных была проанализирована работа всех предложенных подходов и выбран лучший из них.

Отталкиваясь от полученных результатов, можно сказать, что нейронные сети способны в достаточной мере решать поставленную задачу на имеющемся наборе данных. Из-за ограничений производительности, ограничивающих выбор архитектуры сети, предложенный подход не смог показать максимально возможное качество работы, однако имеющийся результат, тем не менее, можно считать успешным.

Список литературы

[1] https://www.kaggle.com/c/carvana-image-masking-challenge/data.

[2] 3d object representations for fine-grained categorization / Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei // 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13). — Sydney, Australia, 2013.

[3] Automatic differentiation in pytorch / Adam Paszke, Sam Gross, Soumith Chintala et al. // NIPS-W. — 2017.

[4] Bag of tricks and A strong baseline for deep person re-identification / Hao Luo, Youzhi Gu, Xingyu Liao et al. // CoRR. — 2019. — Vol. abs/1903.07071. — http://arxiv.org/abs/1903.07071.

[5] Berman, M. Optimization of the jaccard index for image segmentation with the lovász hinge / Maxim Berman, Matthew B. Blaschko // CoRR. — 2017. — Vol. abs/1705.08790. — http://arxiv.org/abs/1705.08790.

[6] Bradski, G. The OpenCV Library / G. Bradski // Dr. Dobb's Journal of Software Tools. — 2000.

[7] Cosface: Large margin cosine loss for deep face recognition / Hao Wang, Yitong Wang, Zheng Zhou et al. // CoRR. — 2018. — Vol. abs/1801.09414. — http://arxiv.org/abs/1801.09414.

[8] Deep residual learning for image recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun // CoRR. — 2015. — Vol. abs/1512.03385. — http://arxiv.org/abs/1512.03385.

[9] Delving deep into rectifiers: Surpassing human-level performance on imagenet classification / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun // CoRR. — 2015. — Vol. abs/1502.01852. — http://arxiv.org/abs/1502.01852.

[10] Devries, T. Improved regularization of convolutional neural networks with cutout / Terrance Devries, Graham W. Taylor // CoRR. — 2017. — Vol. abs/1708.04552. — http://arxiv.org/abs/1708.04552.

[11] Glorot, X. Understanding the difficulty of training deep feedforward neural networks / Xavier Glorot, Yoshua Bengio // Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics / Ed. by Yee Whye Teh, Mike Titterington. — Vol. 9 of Proceedings of Machine Learning Research. — Chia Laguna Resort, Sardinia, Italy: PMLR, 2010. — 13–15 May. — P. 249–256. — http://proceedings.mlr.press/v9/glorot10a.html.

[12] Goodfellow, I. Deep Learning / Ian Goodfellow, Yoshua Bengio, Aaron Courville. — MIT Press, 2016. — http://www.deeplearningbook.org.

[13] Gradient-based learning applied to document recognition / Y. Lecun, L. Bottou, Y. Bengio, P. Haffner // Proceedings of the IEEE. — 1998. — Nov. — Vol. 86, no. 11. — P. 2278–2324.

[14] Hermans, A. In defense of the triplet loss for person re-identification / Alexander Hermans, Lucas Beyer, Bastian Leibe // CoRR. — 2017. — Vol. abs/1703.07737. — http://arxiv.org/abs/1703.07737.

[15] Hoffer, E. Deep metric learning using triplet network / Elad Hoffer, Nir Ailon // Similarity-Based Pattern Recognition / Ed. by Aasa Feragen, Marcello Pelillo, Marco Loog. — Cham: Springer International Publishing, 2015. — P. 84–92.

[16] How does batch normalization help optimization? / Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madry // Advances in Neural Information Processing Systems 31 / Ed. by S. Bengio, H. Wallach, H. Larochelle et al. — Curran Associates, Inc., 2018. — P. 2483–2493. — http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf.

[17] Iglovikov, V. Ternausnet: U-net with VGG11 encoder pre-trained on imagenet for image segmentation / Vladimir Iglovikov, Alexey Shvets // CoRR. — 2018. — Vol. abs/1801.05746. — http://arxiv.org/abs/1801.05746.

[18] Imagenet large scale visual recognition challenge / Olga Russakovsky, Jia Deng, Hao Su et al. // CoRR. — 2014. — Vol. abs/1409.0575. — http://arxiv.org/abs/1409.0575.

[19] Ioffe, S. Batch normalization: Accelerating deep network training by reducing internal covariate shift / Sergey Ioffe, Christian Szegedy // CoRR. — 2015. — Vol. abs/1502.03167. — http://arxiv.org/abs/1502.03167.

[20] Kingma, D. P. Adam: A method for stochastic optimization / Diederik P. Kingma, Jimmy Ba // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. — 2015. — http://arxiv.org/abs/1412.6980.

[21] Labelme: A database and web-based tool for image annotation / Bryan C. Russell, Antonio Torralba, Kevin Murphy, William T. Freeman // International Journal of Computer Vision. — 2008. — 05. — Vol. 77.

[22] Lennie, P. The cost of cortical computation / Peter Lennie // Current Biology. — 2003. — Vol. 13, no. 6. — P. 493 – 497. — http://www.sciencedirect.com/science/article/pii/S0960982203001350.

[23] Loshchilov, I. SGDR: stochastic gradient descent with restarts / Ilya Loshchilov, Frank Hutter // CoRR. — 2016. — Vol. abs/1608.03983. — http://arxiv.org/abs/1608.03983.

[24] Ronneberger, O. U-net: Convolutional networks for biomedical image segmentation / Olaf Ronneberger, Philipp Fischer, Thomas Brox // CoRR. — 2015. — Vol. abs/1505.04597. — http://arxiv.org/abs/1505.04597.

[25] Rumelhart, D. E. Learning representations by back-propagating errors / D. E. Rumelhart, G. E. Hinton, R. J. Williams // . — 1986. — Oct. — Vol. 323. — P. 533–536.

[26] Smith, L. N. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay / Leslie N. Smith // CoRR. — 2018. — Vol. abs/1803.09820. — http://arxiv.org/abs/1803.09820.

[27] Supervisely. — https://supervise.ly.

[28] Wang, M. Deep face recognition: A survey / Mei Wang, Weihong Deng // CoRR. — 2018. — Vol. abs/1804.06655. — http://arxiv.org/abs/1804.06655.

[29] Xie, W. Multicolumn networks for face recognition / Weidi Xie, Andrew Zisserman // CoRR. — 2018. — Vol. abs/1807.09192. — http://arxiv.org/abs/1807.09192.

[30] Zeiler, M. D. Visualizing and understanding convolutional networks / Matthew D. Zeiler, Rob Fergus // CoRR. — 2013. — Vol. abs/1311.2901. — http://arxiv.org/abs/1311.2901.