

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Ярославский государственный университет им. П.Г. Демидова»

Кафедра дискретного анализа

«Допустить к защите»
Заведующий кафедрой
д.ф.-м.н., профессор
_____ Бондаренко В.А.
«__» _____ 2020 г.

Выпускная квалификационная работа бакалавра
по направлению 01.03.02 Прикладная математика и информатика

**Анализ алгоритмов глубокого машинного обучения в задачах
распознавания изображений**

Научный руководитель
к.т.н., старший преподаватель
_____ Матвеев Д.В.
«__» _____ 2020 г.

Студент группы ИВТ-41БО
_____ Коротков А.С.
«__» _____ 2020 г.

Ярославль, 2020

РЕФЕРАТ

Выпускная квалификационная работа: 20 стр., 3 гл., 4 рис., 1 таблиц, 10 источников.

Ключевые слова: машинное обучение, глубокие нейронные сети, распознавание изображений, TensorFlow, OpenCV, Keras.

Объектом исследования являются модели глубоких нейронных сетей для задач распознавания изображений.

Цель работы – изучить применение глубокого машинного обучения в задачах распознавания изображений.

В результате работы были разработаны и реализованы нейронная сети для решения ????. Проведен анализ полученных результатов и сделан вывод о качестве работы нейронных сетей в задачах распознавания изображений.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 МАШИННОЕ ОБУЧЕНИЕ	6
1.1 Понятие искусственной нейронной сети	6
1.2 Активационная функция	7
1.3 Обучение нейронных сетей	8
1.4 Сверточные нейронные сети	10
1.5 Рекуррентные нейронные сети	10
1.6 Инструменты машинного обучения	11
2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	12
2.1 Данные	12
3 АНАЛИЗ РАБОТЫ ПРОГРАММ	13
ЗАКЛЮЧЕНИЕ	14
СПИСОК ЛИТЕРАТУРЫ	15
СПИСОК ИЛЛЮСТРАТИВНОГО МАТЕРИАЛА	16
СПИСОК ТАБЛИЧНОГО МАТЕРИАЛА	17
ПРИЛОЖЕНИЕ А	18

ВВЕДЕНИЕ

В настоящее время, в связи со стремительным развитием цифровых технологий, использование автоматизированных и роботизированных систем распространилось на множество областей как в промышленности, науке, так и в повседневной жизни. В следствие этого, возрастает необходимость в эффективной обработке информации, представленной, в частности, в формате видео и изображений.

На текущий момент изображения тесно влились в жизнь человека. Поэтому многие автоматизированные системы используют их в качестве основного источника информации. Нахождение, локализация, классификация и анализ объектов на изображении компьютером – сложная задача компьютерного зрения.

Компьютерное (машинное) зрение (Computer Vision) – это совокупность программно-технических решений в сфере искусственного интеллекта (ИИ), нацеленных на считывание и получение информации из изображений, в реальном времени и без участия человека.

В процессе обработки информации, получаемой из глаз, человеческий мозг پردازывает колоссальный объем работы. Человек без труда сможет описать что находится и что происходит на случайно взятой фотографии. Изображения могут нести в себе колоссальное количество деталей и отличаться множеством параметров, таких как: разрешение, цветность, качество, яркость, наличие шума и т.д. Объекты на изображениях также могут обладать множеством особенностей: масштаб, положение, цвет, поворот, наклон и т.д. Однако, в цифровом формате, каждое изображение представляет собой лишь массив числовых данных. Научить компьютер находить и классифицировать образы на изображении с учетом всех факторов – очень сложная алгоритмическая задача. Для её решения активно применяют технологии машинного обучения.

Большое количество информации человек получает при помощи зрения. Изображения способны хранить огромное количество информации. Как следствие их использование в компьютерных системах способствует увеличению их производительности. Однако такие технологии требуют сложных вычислений. Задачей компьютерного зрения является разработка эффективных алгоритмов, выполняющих извлечение и анализ данных из изображений или видео.

В настоящий момент, подобные технологии применяются для решения таких сложных задач как:

- OCR – Optical character recognition (Оптическое распознавание символов): преобразование текста на изображении в редактируемый.
- Фотограмметрия – технология создания трехмерной модели объекта на основе фотографий, сделанных с различных ракурсов.
- Motion capture – технология, широко применяемая в киноиндустрии, позволяющая преобразовывать движения реальных людей в компьютерную анимацию.
- Дополненная реальность (AR) – технология, позволяющая в реальном времени проецировать виртуальные объекты на изображение реального окружения.
- Медицинская диагностика – обнаружение раковых клеток на ранней стадии, увеличение качества МРТ изображений, их анализ и т.д.

В данной работе был проведен анализ алгоритмов глубокого машинного обучения для решения задач распознавания изображений. ???

Первая глава содержит основные сведения об машинном обучении.

Во второй главе проведено проектирование и реализация алгоритмов для распознавания изображений.

В третьей главе выполнен сравнительный анализ работы разработанных алгоритмов.

1 МАШИННОЕ ОБУЧЕНИЕ

1.1 Понятие искусственной нейронной сети

Машинное обучение (Machine Learning) – раздел исследований в сфере ИИ, в основе которых лежат методы разработки систем способных к обучению. Алгоритмы машинного обучения эффективно себя показывают в задачах, в которых требуется у заранее подготовленных (обучающих) данных определить общие признаки и по ним идентифицировать новые данные. В проектировании таких, обучающихся, систем часто применяют искусственные нейронные сети.

Искусственная нейронная сеть (ИНС) – компьютерная модель, в основе которой лежат принципы работы биологической нейронной сети - совокупности связанных между собой нервных клеток - нейронов. Каждый нейрон имеет набор входных связей - синапсов, по которым он получает информацию, представленную в виде импульсов, от других нейронов. По полученным данным нейрон формирует своё состояние и, с помощью аксона, сообщает его другим нейронам, обеспечивая функционирование системы. В процессе формирования системы одни нейронные связи укрепляются, а другие ослабевают, обеспечивая обучаемость сети.



Рисунок 1.1 – Типичная структура биологического нейрона

Искусственный нейрон представляет собой упрощенную модель биологического нейрона. Принцип его работы представлен на рисунке 1.2. Сначала нейрон получает n -мерный вектор входных значений $X = (x_1, \dots, x_n)$ и вектор весов $W = (w_1, \dots, w_n)$, обозначающий «укрепленность» межнейронных связей. Вычисляется сумма произведения входных значений и весов s_j . Затем к полученному результату применяется функция активации (φ). В некоторых случаях, к сумме прибавляется величина смещения b_j .

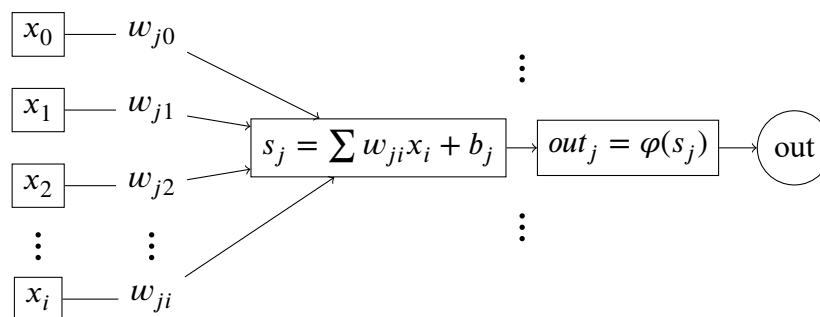


Рисунок 1.2 – Схема искусственного нейрона

Множества нейронов формируют слои, слои в свою очередь формируют нейронную сеть. Входной слой получает данные, обрабатывает и передает нейронам скрытого слоя. Аналогично сработает каждый последующий слой вплоть до выходного.

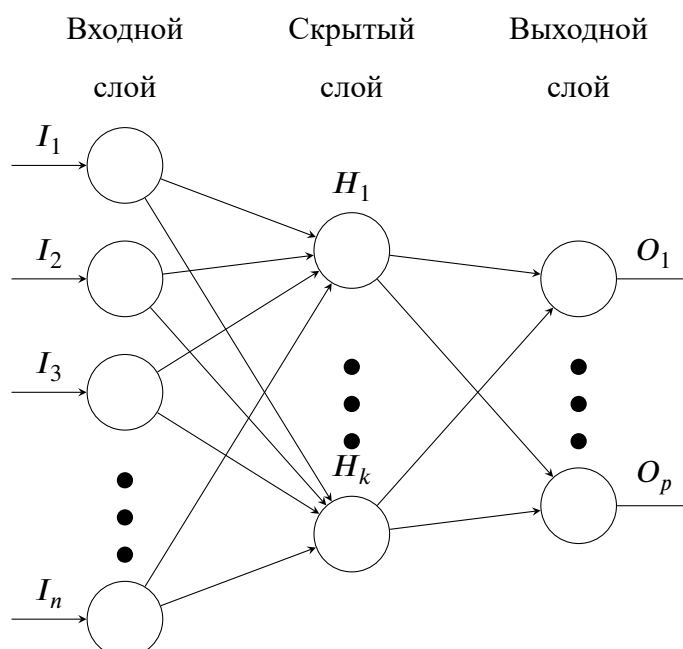


Рисунок 1.3 – Схема простой нейронной сети

Нейросети с большим количеством скрытых слоев называется глубокими. Область машинного обучения, в которой используются глубокие нейронные сети называется - глубоким обучением (Deep Learning).

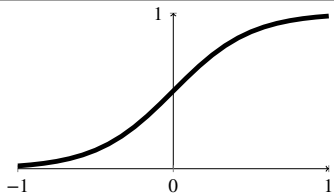
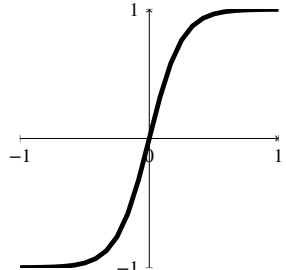
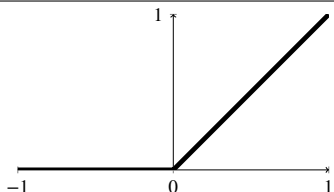
1.2 Активационная функция

Взвешенная сумма входов представляет собой линейную комбинацию, из чего следует, что независимо от количества слоев, значения выходного слоя зависят только от входов первого слоя. Активационная функция нейрона обеспечивает нормализацию

посчитанной суммы и нелинейность нейронной сети. Для многих моделей нейронных сетей также требуется, чтобы активационная функция была монотонной и непрерывно-дифференцируемой на всей области определения.

Существует большое количество функций активации. Наиболее распространенные из них представлены в таблице 1.1.

Таблица 1.1 – Популярные активационные функции

Название	Функция	Вид
Сигмоидная	$\sigma(x) = \frac{1}{1+e^{-x}}$	
Гиперболический тангенс	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
ReLU	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0. \end{cases}$	

1.3 Обучение нейронных сетей

Под обучением нейронных сетей подразумевают настройку значений весов связей для эффективного решения поставленной задачи. Изначально, веса устанавливаются случайно. Затем, в процессе прогона через сеть тестовых данных, веса корректируются так, чтобы в конечном итоге сеть выдавала правильные ответы.

Существует несколько подходов к обучению нейронных сетей:

Обучение с учителем

Обучение с учителем - это такой тип обучения, в котором сеть получает набор входных данных с заранее известными правильными ответами. Веса корректируются в зависимости от того, правильный ли ответ дала сеть.

Одним из популярных методов обучения с учителем является алгоритм обратного распространения ошибки [4], который часто применяется в обучении глубоких нейронных сетей.

Пусть сеть имеет L слоев, a^l, w^l, b^l - векторы значений, весов и смещений нейронов на l -м слое.. Также имеется N обучающих пар (x, y) . В процессе обучения циклично происходят следующие итерации:

- а) На вход сети подается вектор x из обучающего множества, для каждого слоя вычислить значения:
$$z^l = w^l a^{l-1} + b^l a^l = \sigma(z^l)$$
- б) Вычислить значение функции стоимости:
$$C = \frac{1}{2} \sum_j (y_j - a_j^L)^2$$
- в) Вычислить значения ошибок выходного слоя:
$$\delta_j^L = \frac{\delta C}{\delta a_j^L} \sigma'(z_j^L)$$
- г) Вычислить ошибки для каждого предыдущего слоя:
$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l)$$
- д) Вычислить градиент функции стоимости:
$$\frac{\delta C}{\delta w_{jk}^l} = a_k^{l-1} \delta_j^l$$
- е) Обновить веса связей:
$$w_{ij}^l = w_{ij}^l - \mu \frac{\delta C}{\delta w_{jk}^l}, \quad 0 < \mu \leq 1$$

Обучение с подкреплением

Данный метод подразумевает наличие некоторой окружающей среды в которой действует сеть. Такая среда реагирует на действия модели и подает ей определенные сигналы.

Обучение без учителя

Обучение при котором сеть заранее не располагает правильными ответами и самостоятельно ищет общие и отличительные признаки у входных данных.

Генетические алгоритмы обучения

Алгоритмы, имитирующие эволюционные механизмы развития биологической популяции, выступают как альтернатива алгоритму обратного распространения ошибки. Значение произвольного весового коэффициента в нейронной сети называется геном. Гены формируют хромосомы, а хромосомы - популяцию. Далее, в пределах одного цикла (эпохи) с определенными вероятностями происходит:

- Скрещивание хромосом - формирование новой хромосомы из генов двух других
- мутация - случайное изменение произвольного гена
- приспособление - (хромосомы показавшие худшие результаты уничтожаются из популяции.

+/- ???

1.4 Сверточные нейронные сети

Стандартные нейронные сети состоят из полносвязных слоев - слоев, в которых каждый нейрон связан с каждым нейроном следующего слоя, что значительно увеличивает вычислительную сложность системы при увеличении количества нейронов. В типовых сверточных нейронных сетях к полносвязным добавляются сверточные и подвыборочные слои.

Сверточные слои характеризуются использованием матриц весов, называемых фильтрами или ядрами, которые обладают размерностью меньше исходных данных. Такое ядро с определенным шагом проходит по набору входных данных (I) и вычисляет суммы произведений соответствующих значений ячеек и весов, формируя карту признаков ($I * K$). Один сверточный слой может содержать несколько ядер и соответственно несколько карт признаков.

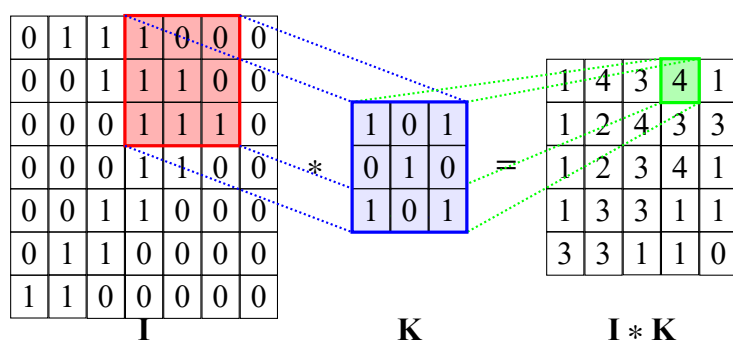


Рисунок 1.4 – Операция свертки

Так как признаки уже обнаружены, для упрощения дальнейших вычислений можно снизить детализацию входных данных. Это обеспечивает подвыборочный (пулинговый) слой, сжимая карты признаков, полученные от сверточного слоя, что снижает количество параметров, используемых в дальнейших вычислениях сети.

СНС может иметь несколько пар чередующихся сверточных и подвыборочных слоев. Завершается СНС стандартными полносвязными слоями.

1.5 Рекуррентные нейронные сети

Все вышеуказанные архитектуры нейронных сетей являются статическими - имеют заранее заданные параметры, без возможности их корректирования в процессе работы сети.

1.6 Инструменты машинного обучения

TensorFlow - многофункциональный фреймворк для машинного обучения с открытым исходным кодом

???

2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1 Данные

В качестве источника обучающих и тестовых данных был использован ресурс: ???

3 АНАЛИЗ РАБОТЫ ПРОГРАММ

ЗАКЛЮЧЕНИЕ

В результате

СПИСОК ЛИТЕРАТУРЫ

1. *Farinella G. M., Battiato S., Cipolla R.* Advanced Topics in Computer Vision. — Springer, 2013. — С. 475.
2. *Jamil M., Sharma S., Singh R.* Fault detection and classification in electrical power transmission system using artificial neural network // SpringerPlus. — 2015. — Июль. — Т. 4. — С. 334. — DOI: 10.1186/s40064-015-1080-x.
3. *Michelucci U.* Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks. — Apress, 2018. — С. 431.
4. *Nielsen M. A.* Neural Networks and Deep Learning. / Determination Press. — 2015. — URL: <http://neuralnetworksanddeeplearning.com/>.
5. *Singh P., Manure A.* Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python. — Apress, 2020. — С. 195.
6. *Жерон О.* Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. — ООО "Альфа-книга", 2018. — С. 688.
7. *Клетте Р.* Компьютерное зрение. Теория и алгоритмы. — ДМК Пресс, 2019. — С. 506.
8. *Конец Д.* Классические задачи Computer Science на языке Python. — Питер, 2020. — С. 256.
9. *Нишант Ш.* Машинное обучение и TensorFlow. — Питер, 2019. — С. 336.
10. *Шолле Ф.* Глубокое обучение на Python. — Питер, 2018. — С. 400.

СПИСОК ИЛЛЮСТРАТИВНОГО МАТЕРИАЛА

Рисунок 1.1	Типичная структура биологического нейрона	6
Рисунок 1.2	Схема искусственного нейрона	7
Рисунок 1.3	Схема простой нейронной сети	7
Рисунок 1.4	Операция свертки	10

СПИСОК ТАБЛИЧНОГО МАТЕРИАЛА

Таблица 1.1	Популярные активационные функции	8
-------------	--	---

ПРИЛОЖЕНИЕ А

Листинг 1

```
1 from __future__ import absolute_import, division, print_function,
    unicode_literals
2
3 import numpy as np
4 import cv2
5 from tensorflow.keras.callbacks import ModelCheckpoint
6 from tensorflow.keras.layers import Conv2D, Flatten, MaxPooling2D,
    Dense, Dropout
7 from tensorflow.keras.models import Sequential
8 from tensorflow.keras.preprocessing.image import ImageDataGenerator,
    img_to_array, load_img, array_to_img
9 import random, os, glob
10 import matplotlib.pyplot as plt
11
12 DIR_PATH = '/home/alexandr/dev/datasets/garbage-classification/
    garbage_classification/Garbage_classification'
13 img_list = glob.glob(os.path.join(DIR_PATH, '*/*.jpg'))
14
15
16 def load_data():
17     train=ImageDataGenerator(
18         horizontal_flip=True, vertical_flip=True, validation_split
19         =0.1, rescale=1./255,
20         shear_range=0.1,
21         zoom_range=0.1,
22         width_shift_range=0.1,
23         height_shift_range=0.1)
24     test=ImageDataGenerator(rescale=1/255, validation_split=0.1)
25     train_generator=train.flow_from_directory(DIR_PATH,target_size
        =(300,300),batch_size=32,
        class_mode='categorical',
        subset='training')
```

```

26 test_generator=test.flow_from_directory(DIR_PATH,target_size
    =(300,300),batch_size=32,
27                                     class_mode='categorical',
                                     subset='validation')
28 labels = (train_generator.class_indices)
29 labels = dict((v,k) for k,v in labels.items())
30
31 print(labels)
32
33 model=Sequential()
34
35 model.add(Conv2D(32,(3,3), padding='same',input_shape=(300,300,3)
    ,activation='relu'))
36 model.add(MaxPooling2D(pool_size=2))
37 model.add(Conv2D(64,(3,3), padding='same',activation='relu'))
38 model.add(MaxPooling2D(pool_size=2))
39 model.add(Conv2D(32,(3,3), padding='same',activation='relu'))
40 model.add(MaxPooling2D(pool_size=2))
41 model.add(Flatten())
42 model.add(Dense(64,activation='relu'))
43 model.add(Dense(6,activation='softmax'))
44
45 filepath="trained_model.h5"
46 checkpoint1 = ModelCheckpoint(filepath, monitor='val_acc',
    verbose=1, save_best_only=True, mode='max')
47 callbacks_list = [checkpoint1]
48 model.summary()
49 model.compile(loss='categorical_crossentropy', optimizer='adam',
    metrics=['acc'])
50 model.fit(train_generator, epochs=150, steps_per_epoch=2276//32,
    validation_data=test_generator,
51             validation_steps=251//32,callbacks=callbacks_list
    )
52 model.save("model.h5")
53
54 if __name__ == '__main__':

```

```
55     os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'
56     load_data()
```