УДК 004.9

ИСПОЛЬЗОВАНИЕ КОМПЬЮТЕРНОГО ЗРЕНИЯ ДЛЯ ОБНАРУЖЕНИЯ ПНЕВМОНИИ

Кивишев К.А., студент гр. ПИб-192, II курс
Научный руководитель: Киреева К.А., ассистент
Кузбасский государственный технический университет
имени Т.Ф. Горбачева
г. Кемерово

Почти у каждого человека есть смартфон, который хранит в себе функции различных устройств, таких как: часы; диктофон; фотоаппарат; видеокамера. Все эти функции находятся в одном устройстве изначально, но так же на него можно устанавливать приложения, которые с использованием возможностей смартфона, выполняют свои функции. К примеру, можно взять приложения, которые могут менять лица местами или накладывать маски в реальном времени. Так же разрабатывается функция дополненной реальности, которая позволит нам направить камеру на какой-нибудь дом или достопримечательность и выведется различная информация, какая улица, дом и что в нём находится. Все эти технологии работают на основе обработки изображений, которые относятся к Компьютерному зрению.

Компьютерное зрение – это область науки, которая занимается задачами, связанными с анализом изображений и видео[1]. Изображение для компьютера выглядит совсем иначе, в отличие от человека. Оно состоит из матрицы чисел, каждый элемент которой хранит в себе код цвета каждого пикселя. Для того, чтобы компьютер мог получить представление об изображении, его обрабатывают с помощью различных алгоритмов, таких как: разность гауссиан; метод Оцу. Такие методы позволяют выявить значимые места на изображении – объекты или их границы. После того как объекты были обнаружены, компьютер начинает анализировать изображение и выводить результат, что изображено на картинке.

В самом начале работы компьютер не понимает, что изображено на фотографии, кошка или же собака, и для решения данной проблемы используется машинное обучение.

Преимуществом машинного обучения является то, что программист не обязан будет писать инструкции, которые будут учитывать все возможные проблемы и будут содержать решение этих проблем. Вместо этого в компьютер (или отдельную программу) закладывают алгоритм самостоятельного нахождения решений путём комплексного использования статистических данных, из которых выводятся закономерности и на основе которых делаются прогнозы[2]. Для того, чтобы процесс машинного обучения был запущен, в компьютер загружают набор данных, на которых алгоритм будет учиться обрабатывать запросы. В наборе данных могут быть изображения кошек и со-

бак, на которых будут метки, обозначающие к кому они относятся. После того, как компьютер был обучен с помощью набора данных, он сможет различать кошек и собак на изображениях без меток. Точность прогнозов зависит от количества верно обработанных данных.

Основная масса задач, решаемых при помощи методов машинного обучения, относится к двум разным видам: обучение с учителем, либо без него. В обоих видах обучения машине предоставляются исходные данные, которые ей предстоит проанализировать и найти закономерности. Различие лишь в том, что при обучении с учителем есть ряд гипотез, которые необходимо опровергнуть или подтвердить. В качестве примера машинного обучения с учителем можно взять следующее: подтвердить или опровергнуть наличие рака у пациента, зная все его медицинские показатели. Данная задача относятся к задачам на классификацию[2].

Задача классификации – получение категориального ответа на основе набора признаков. Имеет конечное количество ответов: есть ли на фотографии кот, является ли изображение человеческим лицом, болен ли пациент раком[2]. Классификация данных происходит для того, чтобы появилась возможность в дальнейшем находить общие признаки одного класса, а также признаки, по которым можно будет различать эти классы. После того как обучение будет пройдено, машина сможет распознавать объекты на изображении и выводить информацию о том, что она видит.

Для работы с изображениями была использована библиотека «OpenCV». Данная библиотека способна считывать изображения, обрабатывать их с помощью алгоритмов, обрезать, выделять определённую область, сохранять изображение и многие другие функции, позволяющие выполнять различные операции над изображением. Так же была использована библиотека «numpy», позволяющая пользователю работать с массивами, матрицами, а также использовать математические функции, предназначенные для многомерных массивов. Для анализа изображений с помощью нейронной сети будет использоваться библиотека «Keras», а также для машинного обучения используется библиотека «TensorFlow».

Для загрузки набора данных, были созданы следующие элементы: переменная, которая хранит в себе путь к набору данных; пустой список данных, в котором будут храниться путь к изображению и его классификатор; массив данных, который хранит себе наименование классификатор и его ключ.

Чтобы загрузить данные используется цикл, который проходит по каталогу, состоящего из каталогов данных здорового лёгкого и лёгкого с пневмонией. Если каталог имеет наименование здорового лёгкого, то программа заходит в этот каталог и все взятые изображения будут переданы в список данных, где будет указан путь до изображения и его класс. То же самое происходит с каталогом, который хранит в себе изображения лёгких с пневмонией, программа добавляет в список данных путь к изображению и его класс. Далее

выполняется вывод общего количества изображений, представленных в обучающих данных, а также выводится количество классов (рис. 1).

```
(5216, 2)
```

Рисунок 1 – вывод количества изображений и классов

Далее были созданы количественные переменные, первая хранит в себе общее число изображений, а вторая и третья хранят в себе общее количество изображений нормального лёгкого и лёгкого с пневмонией соответственно. После чего были созданы 2 списка, в одном из которых будут храниться нормальные изображения, а в другом изображения пневмонии. Также необходимо создать цикл, который проходит по списку со всеми изображениями и проверяет ключ данного изображения. Если ключ изображения относится к нормальным, то изображение добавляется в список с нормальными и к переменной нормального лёгкого прибавляется единица, в ином случае изображение добавляется в список с пневмонией, и единица прибавляется к переменной с пневмонией. После того как цикл пройдёт по всему списку изображений, выведем размеры данных списков и общее количество изображений (рис. 2).

```
Всего изображений =  5216
Всего нормальных изображений =  1341
Всего изображений с пневмонией =  3875
```

Рисунок 2 – Вывод общего количества изображений, количества нормальных и изображений с пневмонией

Далее для распознавания изображений создаётся Сверточная нейронная сеть (СНС). Исходное изображение представляется для компьютера в виде матрицы чисел и поэтому выполняется произведение матрицы изображения на ядро свертки. Ядро свертки представляет собой квадратную матрицу чисел. Произведение матриц происходит следующим образом: берется область матрицы изображения, размеры которой равны матрице ядра и после происходит произведение области на ядро свертки и результат записывается в матрицу, которую называют сверточным слоем. После чего область сдвигается на один столбец в сторону и происходит произведение области на ядро. Сверточный слой служит для того, чтобы нейросеть могла распознавать абстрактные образы. Для создания сверточного слоя необходимо указать следующие параметры: число ядер; размер ядра; матрицу изображения в градациях серого. Далее необходимо усреднить значения фильтров, указывая следующие параметры: размер окна; шаг сканирования. В итоге получается усредненная матрица, размеры которой в 2 раза меньше матрицы изображения. Далее можно вывести структуру нейронной сети, где указаны размеры матрицы, количество фильтров и количество коэффициентов в каждом слое (рис. 3).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 150, 150, 32)      320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 150, 150, 64)      18496
_________________________________________________________________
batch_normalization (BatchNo (None, 150, 150, 64)      256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 75, 75, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 75, 75, 128)       73856
_________________________________________________________________
batch_normalization_1 (Batch (None, 75, 75, 128)       512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 37, 37, 128)       0
_________________________________________________________________
dropout (Dropout)            (None, 37, 37, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 37, 37, 128)       147584
_________________________________________________________________
batch_normalization_2 (Batch (None, 37, 37, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 18, 18, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 18, 18, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 18, 18, 128)       147584
_________________________________________________________________
batch_normalization_3 (Batch (None, 18, 18, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 9, 9, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 9, 9, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 10368)             0
_________________________________________________________________
dense (Dense)                (None, 512)               5308928
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 5,699,073
Trainable params: 5,698,177
Non-trainable params: 896
```

Рисунок 3 – Структура нейронной сети

Далее необходимо реализовать машинное обучение, для этого необходимо указать модель для обучения, количество проходов по всему набору данных, данные по которым можно будет оценить потери и любые метрики после каждого прохода и будут получены результаты (рис. 4 и 5).

```
Epoch 1/30
163/163 [==============================] - 672s 4s/step - loss: 0.5959 - accuracy: 0.7950 - val_loss: 3.2672 - val_accuracy: 0.
7429
Epoch 2/30
163/163 [==============================] - 662s 4s/step - loss: 0.3442 - accuracy: 0.8576 - val_loss: 5.5916 - val_accuracy: 0.
7429
Epoch 3/30
163/163 [==============================] - 652s 4s/step - loss: 0.2777 - accuracy: 0.8755 - val_loss: 5.9664 - val_accuracy: 0.
7429
Epoch 4/30
163/163 [==============================] - 652s 4s/step - loss: 0.2598 - accuracy: 0.8912 - val_loss: 2.3122 - val_accuracy: 0.
7441
Epoch 5/30
163/163 [==============================] - 657s 4s/step - loss: 0.2463 - accuracy: 0.8987 - val_loss: 0.6892 - val_accuracy: 0.
8200
Epoch 6/30
163/163 [==============================] - 650s 4s/step - loss: 0.2335 - accuracy: 0.9025 - val_loss: 0.2726 - val_accuracy: 0.
9030
Epoch 7/30
163/163 [==============================] - 650s 4s/step - loss: 0.2353 - accuracy: 0.8947 - val_loss: 0.2236 - val_accuracy: 0.
9099
Epoch 8/30
163/163 [==============================] - 660s 4s/step - loss: 0.2138 - accuracy: 0.9088 - val_loss: 0.3565 - val_accuracy: 0.
8871
Epoch 9/30
163/163 [==============================] - 660s 4s/step - loss: 0.2151 - accuracy: 0.9120 - val_loss: 0.6088 - val_accuracy: 0.
8372
Epoch 10/30
163/163 [==============================] - 661s 4s/step - loss: 0.2218 - accuracy: 0.9104 - val_loss: 0.2226 - val_accuracy: 0.
9097
Epoch 11/30
163/163 [==============================] - 661s 4s/step - loss: 0.1961 - accuracy: 0.9276 - val_loss: 0.4210 - val_accuracy: 0.
8729
Epoch 12/30
163/163 [==============================] - 662s 4s/step - loss: 0.1929 - accuracy: 0.9198 - val_loss: 4.9503 - val_accuracy: 0.
7429
Epoch 13/30
163/163 [==============================] - 659s 4s/step - loss: 0.2159 - accuracy: 0.9115 - val_loss: 0.4005 - val_accuracy: 0.
8683
Epoch 14/30
163/163 [==============================] - 658s 4s/step - loss: 0.1816 - accuracy: 0.9302 - val_loss: 0.9585 - val_accuracy: 0.
7071
Epoch 15/30
163/163 [==============================] - 656s 4s/step - loss: 0.1828 - accuracy: 0.9277 - val_loss: 0.3109 - val_accuracy: 0.
8892
Epoch 16/30
163/163 [==============================] - 651s 4s/step - loss: 0.1873 - accuracy: 0.9263 - val_loss: 0.5682 - val_accuracy: 0.
7885
```

Рисунок 4 – Результаты обучения

```
Epoch 17/30
163/163 [==============================] - 652s 4s/step - loss: 0.1842 - accuracy: 0.9240 - val_loss: 2.2566 - val_accuracy: 0.
7444
Epoch 18/30
163/163 [==============================] - 652s 4s/step - loss: 0.1861 - accuracy: 0.9277 - val_loss: 0.2970 - val_accuracy: 0.
9007
Epoch 19/30
163/163 [==============================] - 652s 4s/step - loss: 0.1841 - accuracy: 0.9307 - val_loss: 0.3289 - val_accuracy: 0.
8815
Epoch 20/30
163/163 [==============================] - 652s 4s/step - loss: 0.1761 - accuracy: 0.9259 - val_loss: 4.1424 - val_accuracy: 0.
7429
Epoch 21/30
163/163 [==============================] - 664s 4s/step - loss: 0.1778 - accuracy: 0.9324 - val_loss: 2.6071 - val_accuracy: 0.
5163
Epoch 22/30
163/163 [==============================] - 653s 4s/step - loss: 0.1701 - accuracy: 0.9270 - val_loss: 1.1552 - val_accuracy: 0.
7605
Epoch 23/30
163/163 [==============================] - 654s 4s/step - loss: 0.1840 - accuracy: 0.9279 - val_loss: 0.3073 - val_accuracy: 0.
8829
Epoch 24/30
163/163 [==============================] - 651s 4s/step - loss: 0.1744 - accuracy: 0.9276 - val_loss: 4.3867 - val_accuracy: 0.
7429
Epoch 25/30
163/163 [==============================] - 672s 4s/step - loss: 0.1792 - accuracy: 0.9283 - val_loss: 0.7817 - val_accuracy: 0.
8200
Epoch 26/30
163/163 [==============================] - 711s 4s/step - loss: 0.1598 - accuracy: 0.9361 - val_loss: 0.9121 - val_accuracy: 0.
7776
Epoch 27/30
163/163 [==============================] - 674s 4s/step - loss: 0.1829 - accuracy: 0.9306 - val_loss: 1.3561 - val_accuracy: 0.
6378
Epoch 28/30
163/163 [==============================] - 685s 4s/step - loss: 0.1550 - accuracy: 0.9399 - val_loss: 2.1096 - val_accuracy: 0.
7444
Epoch 29/30
163/163 [==============================] - 683s 4s/step - loss: 0.1640 - accuracy: 0.9396 - val_loss: 0.2175 - val_accuracy: 0.
9116
Epoch 30/30
163/163 [==============================] - 672s 4s/step - loss: 0.1564 - accuracy: 0.9402 - val_loss: 0.9863 - val_accuracy: 0.
8131
```

Рисунок 5 – Результаты обучения

Результаты показывают следующие параметры: общее время одного прохода; время обработки одного изображения; потери тренировочной выборки; точность тренировочной выборки; потери валидационной выборки; точность валидационной выборки. В самом начале обучения коэффициент

ошибок сильно высок, что является вполне нормальным явлением, потому что это только первый проход по данным и с каждым проходом можно заметить, что коэффициент постепенно падает.

Для визуализации результатов можно использовать графики, на которых будут изображены количества проходов, коэффициент правильно спрогнозированных изображений и коэффициент потерь (рис. 6). На левом графике можем заметить то, что на тренировочном наборе данных компьютер выполнял довольно-таки точные прогнозы в отличие от валидационного. Такое явление обуславливается тем, что валидационный набор данных хранит в себе не идеальные данные, имеется в виду, что такие данные относятся какому-то классу, но только по некоторым признакам и компьютеру сложно определить, к какому классу относятся данные.

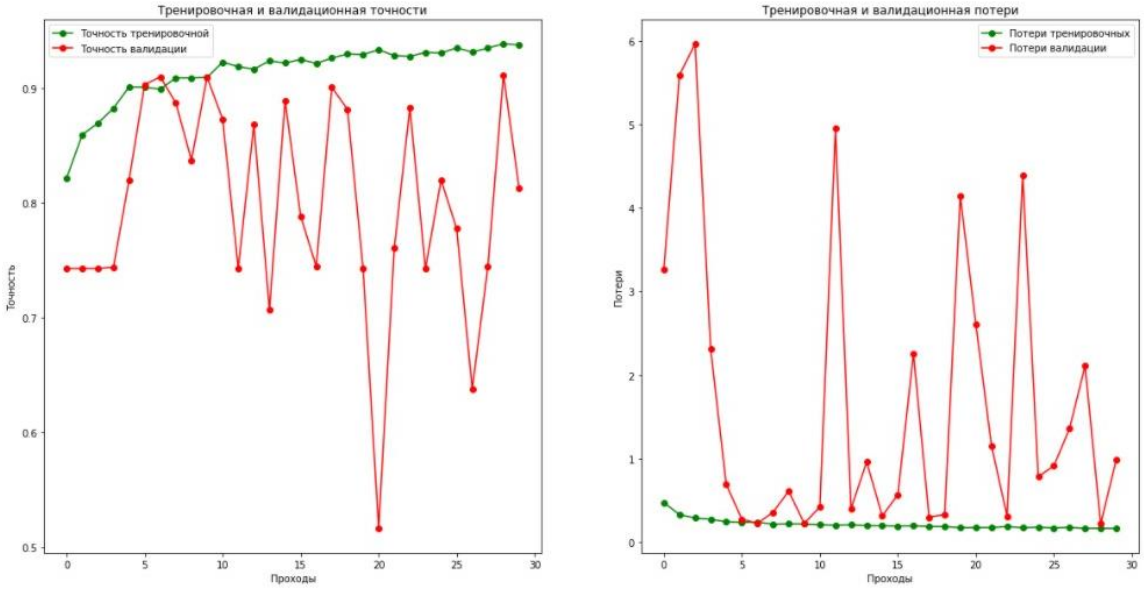

Рисунок 6 – Графики правильных прогнозов и потерь

В рамках данной работы было реализовано машинное обучение, которое можно использовать в реальной жизни. Т.к. задача состоит в анализе изображений, то необходимо использовать машинное обучение. Было выяснено, что машина обучается с помощью набора данных, которые классифицированы и обучение состоит из тренировочной и валидационного набора данных. Тренировочная является непосредственно для тренировки компьютера, чтобы он мог распознавать признаки отличия, а валидационная является проверочной выборкой.

Список литературы:

1. Как работает компьютерное зрение. Изучаем основные алгоритмы, чтобы овладеть data science [Электронный ресурс] – Режим доступа: https://xakep.ru/2019/01/14/yandex-ds/ свободный (дата обращения: 10.03.2021).
2. Введение в машинное обучение [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/448892/ свободный (дата обращения: 10.03.2021).