ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики

Моя первая нейронная сеть. Распознавание номера дома по фотографии.

Проектная работа студента образовательной программы бакалавриата
«Прикладная математика»
по направлению подготовки 01.03.04 ПРИКЛАДНАЯ МАТЕМАТИКА

Студент:
Морозов Д.С.

Руководитель:
Профессор В. Ю. Попов

Москва 2024г.

Содержание

1 Введение

Распознавание объектов по фотографиям – одна из самых увлекательных задач глубокого машинного обучения. Навык анализировать визуальные данные с помощью алгоритмов машинного обучения является одним из самых ценных и применяемых в наше время. Эта способность не может быть переоценена, так как используется в самых разных областях науки, техники.

Эта работа сосредоточена на распознании номеров домов по фотографиям. Такой навык может быть полезен инновационным роботам доставщикам. Для этого мы используем некоторые нейросетевые инструменты, о которых будет рассказано далее. Нам нужно добиться хороших результатов в классификации номеров по их фотографиям с улицы. Мы будем использовать базу данных SVHN, которая содержит около 600000 идентифицированных цифр, вырезанных из фотографий уличных номеров. Эта база данных довольно популярна. Она использовалась в нейронных сетях Google, с целью автоматически улучшить качество нумерации домов на онлайн картах.

Цель нашего исследования заключается в том, чтобы создать эффективную модель для предсказывания номера дома по фотографии, используя библиотеку TensorFlow и язык программирования Python.

2 Какие библиотеки нам понадобятся и почему?

TensorFlow – эта одна из самых популярных и прогрессивных библиотек, используемых для машинного обучения. В ней содержится широкий набор инструментов, с помощью которых решается великое множество современных задач.

Ряд преимуществ этой библиотеки:

- Библиотека поддерживает такие алгоритмы машинного обучения как: сверточные, рекуррентнтые нейросети, автоэнкодеры и другие.

- Библиотека успешно масштабируется для разных задач: от небольших и простых, до сложных, требующих массивных облачных вычислений.
- TensorFlow совместим с другими библиотеками Python: NumPy, Pandas, Matplotlib и другими.

Также нам важна такая библиотека как Keras. Keras – это высокоуровневый API для нейросетей, фактически уже встроенный в TensorFlow и способный работать поверх него. Keras упрощает построение последовательных моделей, состоящих из слоёв. Также эта библиотека позволяет работать с большими объёмами данных и проводить аугментацию.

Такой набор инструментов позволит нам создать, обучить, оценить собственную модель нейронной сети. Эта модель научится решать поставленную задачу: распознавать номер по реальному фото.

3 База данных

1. Для начала нам необходимо найти и загрузить необходимую Базу данных. Чтобы это сделать я воспользовался сайтом Kaggle, на котором находится огромное множество различных баз данных с абсолютно разной информационной нагрузкой. Для нашей задачи был выбран датасет SVHN с тренировочными данными. После импорта необходимых библиотек посмотрим, что в нём хранится.

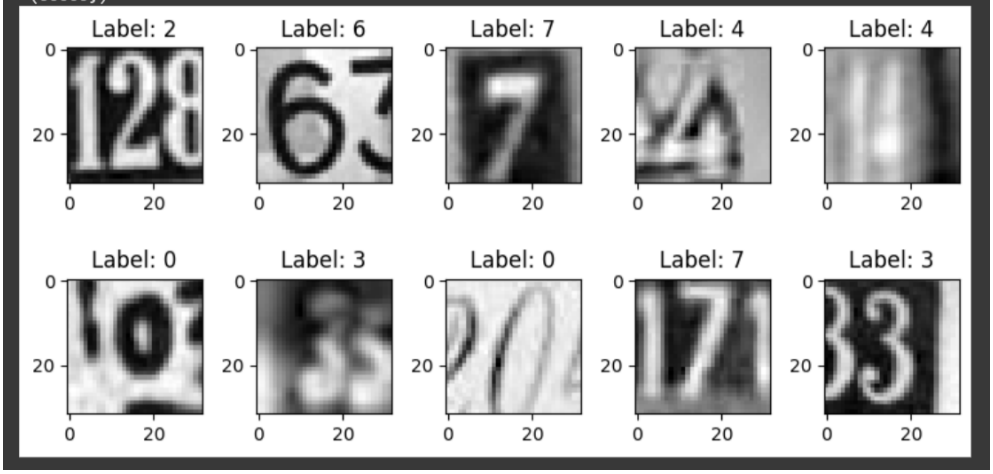
```
filepath = "/content/drive/MyDrive/Colab Notebooks/SVHN_single_grey1.h5"
df = h5py.File(filepath, 'r')
dataframe = np.array(df)
ls = list(dataframe)
print("List of datasets in this file: \n", ls)

X_test = np.array(df['X_test'])
X_train = np.array(df['X_train'])
X_val = np.array(df['X_val'])

y_test = np.array(df['y_test'])
y_train = np.array(df['y_train'])
y_val = np.array(df['y_val'])
print("Shape of X_train: \n", X_train.shape)
print("Shape of y_train: \n", y_train.shape)
print("Shape of X_test: \n", X_test.shape)
print("Shape of y_test: \n", y_test.shape)
print("Shape of X_val: \n", X_val.shape)
print("Shape of y_val: \n", y_val.shape)
```

Здесь мы считываем данные из файла, извлекаем тренировочные, валидационные и тестовые данные. Также визуализируем первые 10 изображений из тренировочного набора с их метками:

```
List of datasets in this file:
 ['X_test', 'X_train', 'X_val', 'y_test', 'y_train', 'y_val']
Shape of X_train:
 (42000, 32, 32)
Shape of y_train:
 (42000,)
Shape of X_test:
 (18000, 32, 32)
Shape of y_test:
 (18000,)
Shape of X_val:
 (60000, 32, 32)
Shape of y_val:
 (60000,)
```



2. Проанализировав данные, структуру датасета, мы можем приступить к предобработке данных.

Предобработка данных – важный этап подготовки данных в машинном обучении. На этом этапе проводятся такие процессы как сглаживание, нормализация данных, снижение шума и прочее. Всё это повышает качество данных, приводит их к ожидаемому моделью виду, и, соответственно, повышает эффективность обучения самой модели.

```
x_train = np.expand_dims(X_train, axis=-1)
x_train = x_train.astype('float32') / 255
x_val = np.expand_dims(X_val, axis=-1)
x_val = x_val.astype('float32') / 255
x_test = np.expand_dims(X_test, axis=-1)
x_test = x_test.astype('float32') / 255
print("Shape of x_train:", x_train.shape)
print("Shape of x_val:", x_val.shape)
print("Shape of x_test:", x_test.shape)

y_train = keras.utils.to_categorical(y_train)
y_val = keras.utils.to_categorical(y_val)
y_test = keras.utils.to_categorical(y_test)

print("Shape of ytrain:", y_train.shape)
print("Shape of yval:", y_val.shape)
print("Shape of ytest:", y_test.shape)
```

```
Shape of x_train: (42000, 32, 32, 1)
Shape of x_val: (60000, 32, 32, 1)
Shape of x_test: (18000, 32, 32, 1)
Shape of ytrain: (42000, 10)
Shape of yval: (60000, 10)
Shape of ytest: (18000, 10)
```

Здесь мы добавляем ось канала, масштабируем значения пикселей, преобразуем классы в удобную для модели форму. Таким образом мы повышаем стабильность и качество обучения. Этот этап окажется очень важным для успешной работы всей остальной программы. Следующим шагом будет построение модели.

4 Построение модели

1. Сперва нам нужно описать структуру модели. Мы используем модель Sequential, что означает “последовательная”. Эта модель представляет собой линейный набор слоёв в Keras, каждый из этих слоёв имеет свой вес, один вход и один выход. Мы можем последовательно добавлять по одному слою за раз.

2. Далее мы используем сверточные слои, применяющиеся ко входным данным с использованием фильтров.

Также мы нормализируем каждый предыдущий слой с помощью BatchNormalization.

Используем слой Dropout, отключающий часть нейронов для предотвращения переобучения. Задаём, кол-во эпох, пакетный размер и т.д.

3. За преобразования, полносвязные слои, классификацию отвечают соответственно Flatten, Dense, softmax.

```
model2 = keras.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 1)),
    keras.layers.BatchNormalization(),

    keras.layers.Conv2D(32, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),

    keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
    keras.layers.BatchNormalization(),

    keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),

    keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
    keras.layers.BatchNormalization(),

    keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(0.5),

    keras.layers.Flatten(),

    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.3),

    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.1),

    keras.layers.Dense(10, activation='softmax')
])
```

5 Обучение модели

```
model2.compile("adam", "categorical_crossentropy", metrics=['accuracy'])

history2 = model2.fit(x=x_train, y=y_train,
                      validation_data=(x_val, y_val),
                      batch_size=32,
                      epochs=20,
                      verbose=1)
```

На этом этапе начинается самое интересное – тренировка.

Используем метод fit для тренировки модели на тренировочных данных.

Передаём входные данные и метки для тренировки и набор данных для оценки модели, задаём количество эпох, выводим информацию о ходе тренировки.

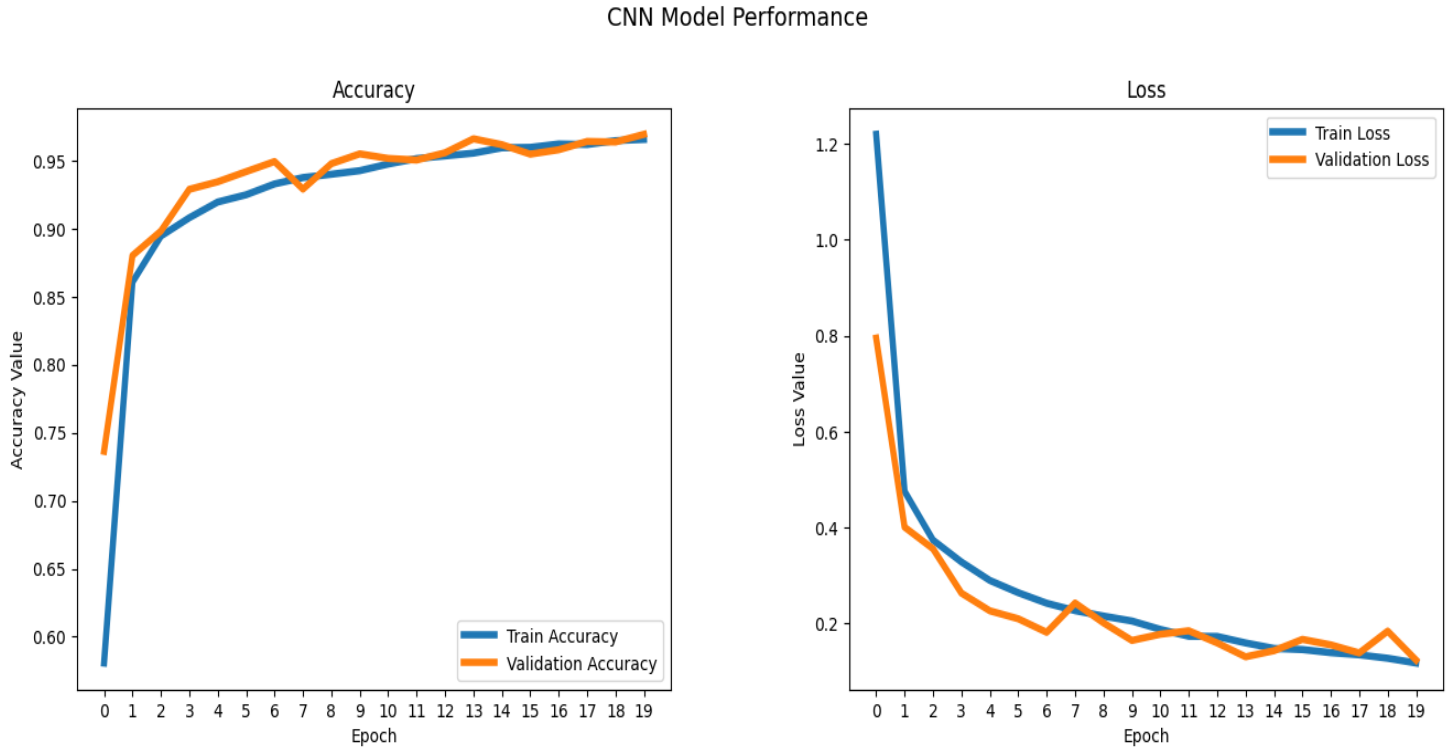
Такие функции позволяют нам проследить за процессом тренировки нейросети. Мы можем посмотреть информацию о точности, стабильности, времени тренировки.

```
Epoch 11/20
1313/1313 [==============================] - 316s 241ms/step - loss: 0.1880 - accuracy: 0.9474 - val_loss: 0.1780 - val_accuracy: 0.9515
Epoch 12/20
1313/1313 [==============================] - 318s 243ms/step - loss: 0.1738 - accuracy: 0.9514 - val_loss: 0.1852 - val_accuracy: 0.9503
Epoch 13/20
1313/1313 [==============================] - 378s 288ms/step - loss: 0.1734 - accuracy: 0.9534 - val_loss: 0.1599 - val_accuracy: 0.9557
Epoch 14/20
1313/1313 [==============================] - 319s 243ms/step - loss: 0.1599 - accuracy: 0.9554 - val_loss: 0.1310 - val_accuracy: 0.9660
Epoch 15/20
1313/1313 [==============================] - 318s 242ms/step - loss: 0.1484 - accuracy: 0.9593 - val_loss: 0.1437 - val_accuracy: 0.9616
Epoch 16/20
1313/1313 [==============================] - 380s 289ms/step - loss: 0.1461 - accuracy: 0.9596 - val_loss: 0.1673 - val_accuracy: 0.9547
Epoch 17/20
1313/1313 [==============================] - 320s 244ms/step - loss: 0.1396 - accuracy: 0.9621 - val_loss: 0.1558 - val_accuracy: 0.9579
Epoch 18/20
1313/1313 [==============================] - 323s 246ms/step - loss: 0.1351 - accuracy: 0.9617 - val_loss: 0.1390 - val_accuracy: 0.9640
Epoch 19/20
1313/1313 [==============================] - 383s 292ms/step - loss: 0.1280 - accuracy: 0.9646 - val_loss: 0.1844 - val_accuracy: 0.9636
Epoch 20/20
1313/1313 [==============================] - 323s 246ms/step - loss: 0.1181 - accuracy: 0.9655 - val_loss: 0.1242 - val_accuracy: 0.9692
563/563 [==============================] - 25s 44ms/step - loss: 0.2411 - accuracy: 0.9444
TEST SET: accuracy: 94.44%
```

Мы получим такой список данных об обучении на каждой эпохе. Можно ли сделать анализ чуть удобнее? Конечно можно!

6 Визуализация данных

С помощью визуализации мы упростим анализ результатов обучения нашей модели. Это важно сделать чтобы понять, всё ли идёт хорошо, не надо ли применить ещё какие-то отдельные методы для улучшения качества обучения?



С помощью языка Python и соответствующих библиотек для визуализации данных мы смогли построить два замечательных графика, взглянув на которые мы можем оценить процесс обучения нашей нейросети.

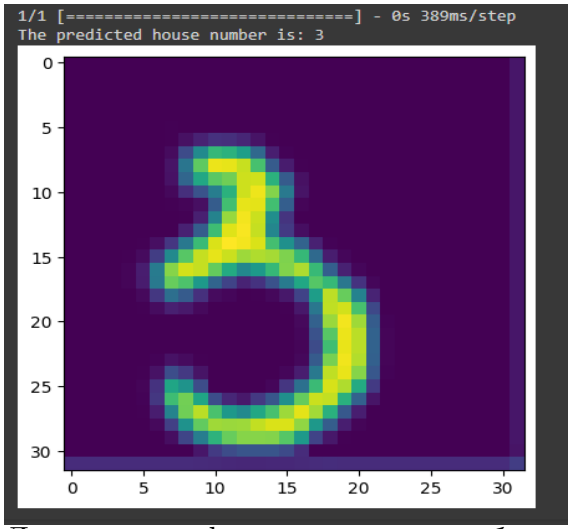
7 Заключение и тесты

Как было сказано ранее, в нашем датасете SVHN хранится около 600000 идентифицированных отдельных цифр. Значит наша нейросеть пока не сможет предсказать двузначное или трёхзначное число полностью. Тем не менее нам удалось создать модель, которая довольно успешно определяет отдельные цифры.

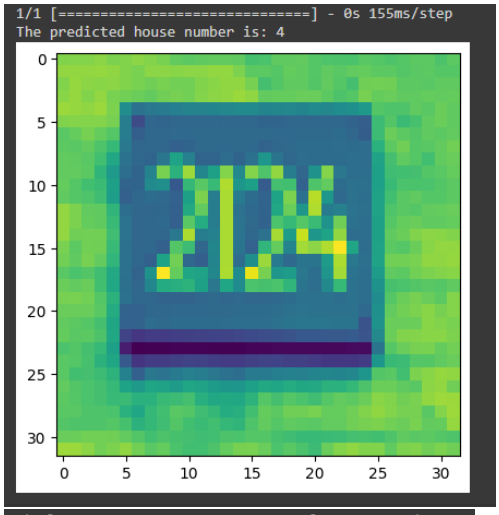
Возьмём пару фотографий из интернета:

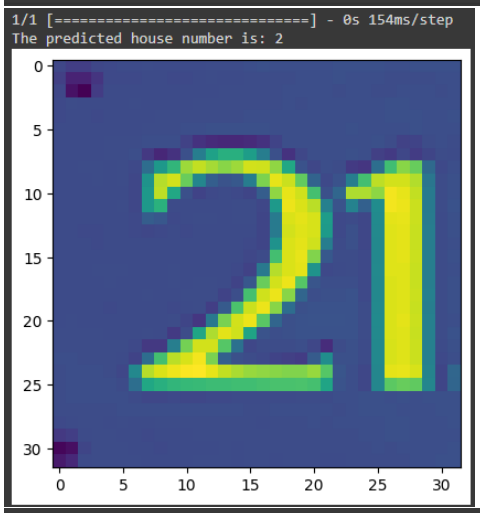
И сгенерированное изображение:

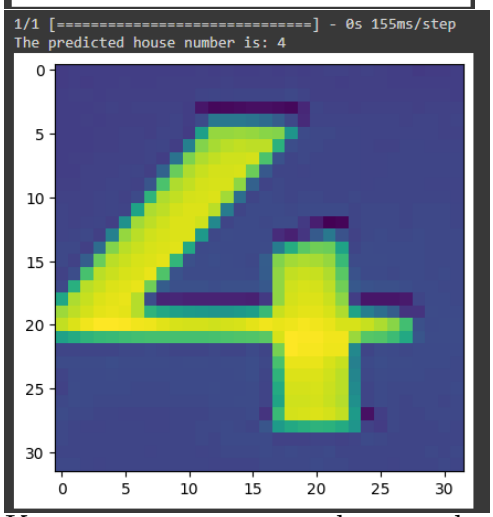
Результаты:



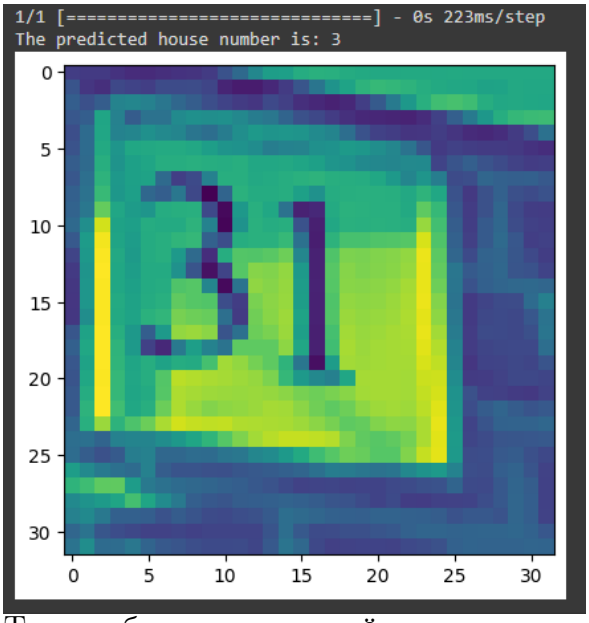
Для одной цифры нейросеть отработала штатно, цифра определена верно.







Как можем видеть, на фотографии с множеством цифр нейросеть успешно распознаёт только одну из них.



Таким образом, наша нейронная сеть может быть улучшена до второй ступени – распознавание нескольких цифр одновременно. Такая нейросеть может оказаться полезной в создании карт, в обучении роботов доставщиков, в опознавании номерных автомобильных знаков и многом другом.

8 Список литературы

1. Документация библиотеки TensorFlow:

https://www.tensorflow.org/guide?hl=ru

2. Документация Keras:

https://keras.io/guides/

3. Документация Matplotlib:

https://matplotlib.org/3.3.3/contents.html

4. База данных SVHN:

https://www.kaggle.com/datasets/sasha18/street-view-house-nos-h5-file/code

5. Пример кода и дополнительная информация по этой базе данных:

https://www.kaggle.com/code/mdriponmiah/cnn-svhn-street-view-housing-number-digit