

Министерство образования Республики Беларусь  
Учреждение образования  
"Белорусский Государственный университет информатики  
и радиоэлектроники"

Лабораторная работа №1  
**“Логистическая регрессия в качестве нейронной сети”**  
по учебной дисциплине “Машинное обучение”

Выполнил:

Студент гр. 956241 Дубовик Н.О.

Минск 2020

**Данные:** В работе предлагается использовать набор данных notMNIST, который состоит из изображений размерностью  $28 \times 28$  первых 10 букв латинского алфавита (A ... J, соответственно). Обучающая выборка содержит порядка 500 тыс. изображений, а тестовая – около 19 тыс.

Данные можно скачать по ссылке:

- [https://commondatastorage.googleapis.com/books1000/notMNIST\\_large.tar.gz](https://commondatastorage.googleapis.com/books1000/notMNIST_large.tar.gz) (большой набор данных);
- [https://commondatastorage.googleapis.com/books1000/notMNIST\\_small.tar.gz](https://commondatastorage.googleapis.com/books1000/notMNIST_small.tar.gz) (маленький набор данных);

Описание данных на английском языке доступно по ссылке:

<http://yaroslavvb.blogspot.sg/2011/09/notmnist-dataset.html>

Результат выполнения заданий опишите в отчете.

В ходе выполнения лабораторной работы был использован датасет notMNIST\_large

### **Задание 1.**

Загрузите данные и отобразите на экране несколько из изображений с помощью языка Python;

Следующие изображения, из предоставленного датасета, были показаны с помощью библиотеки matplotlib.pyplot, рисунок 1.

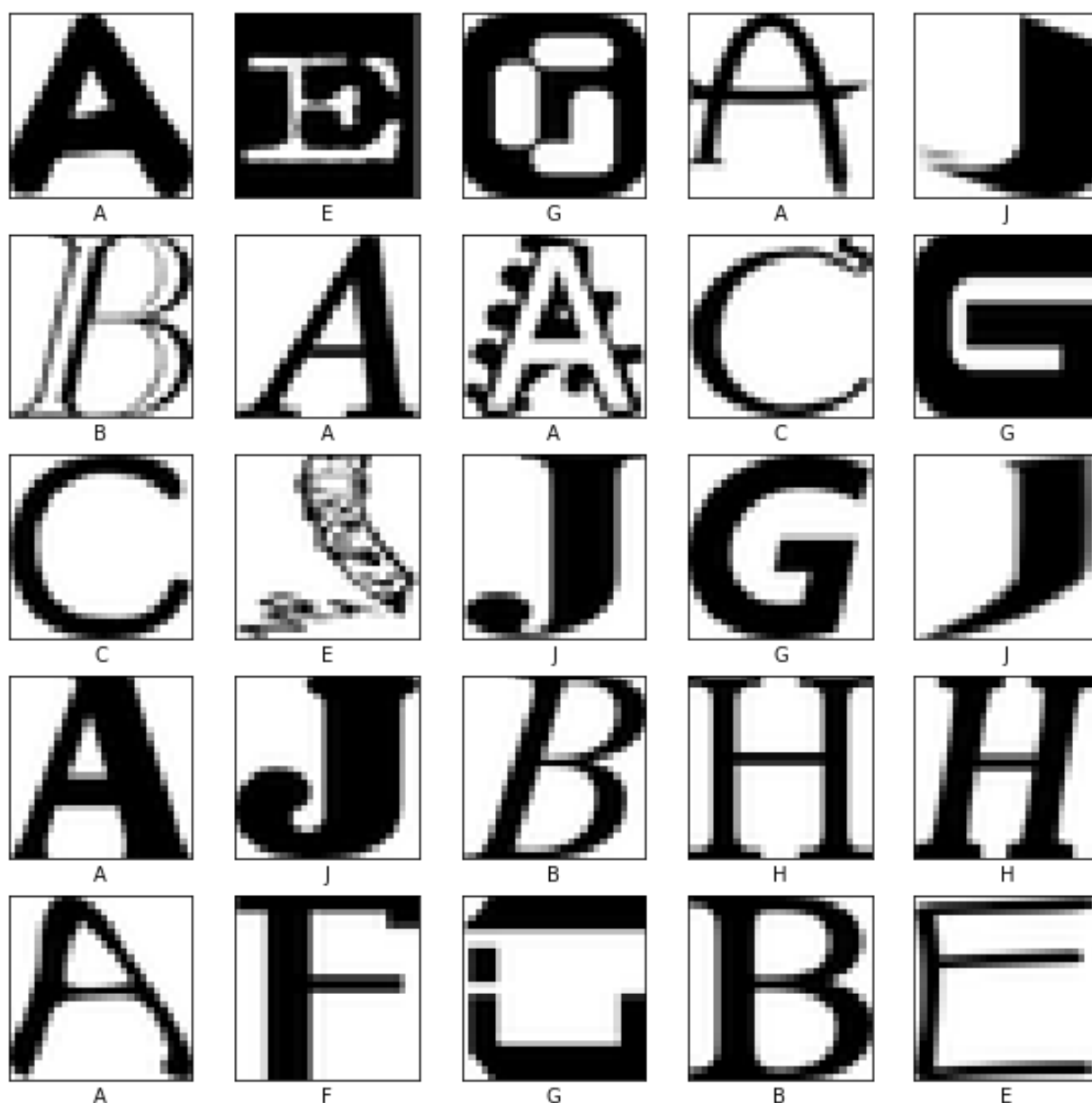


Рисунок 1 – Изображения из датасета

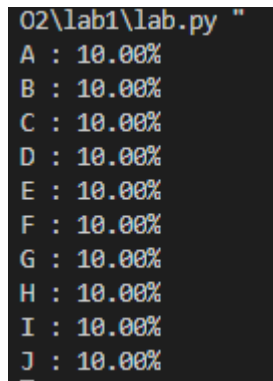
### Задание 2.

Проверьте, что классы являются сбалансированными, т.е. количество изображений, принадлежащих каждому из классов, примерно одинаково (В данной задаче 10 классов).

Для этого задания была использована следующая функция:

```
def show_percentages(Y, classes):
    total=Y.shape[1]
    for i in range(len(classes)):
        count=np.count_nonzero(Y==i)
        print("{0} : {1:.2f}%".format(classes[i],count/total*100))
```

И ее результат, рисунок 2.



```
02\lab1\lab.py "  
A : 10.00%  
B : 10.00%  
C : 10.00%  
D : 10.00%  
E : 10.00%  
F : 10.00%  
G : 10.00%  
H : 10.00%  
I : 10.00%  
J : 10.00%
```

Рисунок 2 – Результат проверки на сбалансированность классов

### Задания 3/4.

Разделите данные на три подвыборки: обучающую (200 тыс. изображений), валидационную (10 тыс. изображений) и контрольную (тестовую) (19 тыс. изображений);

Проверьте, что данные из обучающей выборки не пересекаются с данными из валидационной и контрольной выборок. Другими словами, избавьтесь от дубликатов в обучающей выборке.

Для этих заданий использовался метод:

```
def split_dataset(X,Y,train_size, valid_size,test_size):  
    train_index=train_size  
    valid_index=train_index+valid_size  
    test_index=valid_index+test_size  
  
    p=np.random.permutation(X.shape[1])  
  
    X_split=np.hsplit(X[:,p], [train_index,valid_index,test_index])  
    Y_split=np.hsplit(Y[:,p], [train_index,valid_index,test_index])  
    return X_split[0],X_split[1],X_split[2],Y_split[0],Y_split[1],Y_split[2]
```

С аргументами:

```
split_dataset(X,Y,200000,10000,19000)
```

### Задание 5.

Постройте простейший классификатор (например, с помощью логистической регрессии). Постройте график зависимости точности классификатора от размера обучающей выборки (50, 100, 1000, 50000). Для

построения классификатора можете использовать библиотеку SkLearn (<http://scikit-learn.org>).

Был использован классификатор OneVsRestClassifier из библиотеки sklearn.multiclass.

Сам метод выглядит следующим образом:

```
def train(X_train,Y_train,X_valid,Y_valid):  
    model=OneVsRestClassifier(LogisticRegression(solver="lbfgs",max_iter=1000))  
    .fit(X_train.T,Y_train.T)  
    print("train score: {}".format(model.score(X_train.T,Y_train.T)))  
    print("validation score: {}".format(model.score(X_valid.T,Y_valid.T)))  
    return model
```

Итоги его выполнения на выборках различного размера показаны на рисунке 3. По данному рисунку можно сделать вывод, что чем больше размер обучающей выборки, тем выше будет оценка.

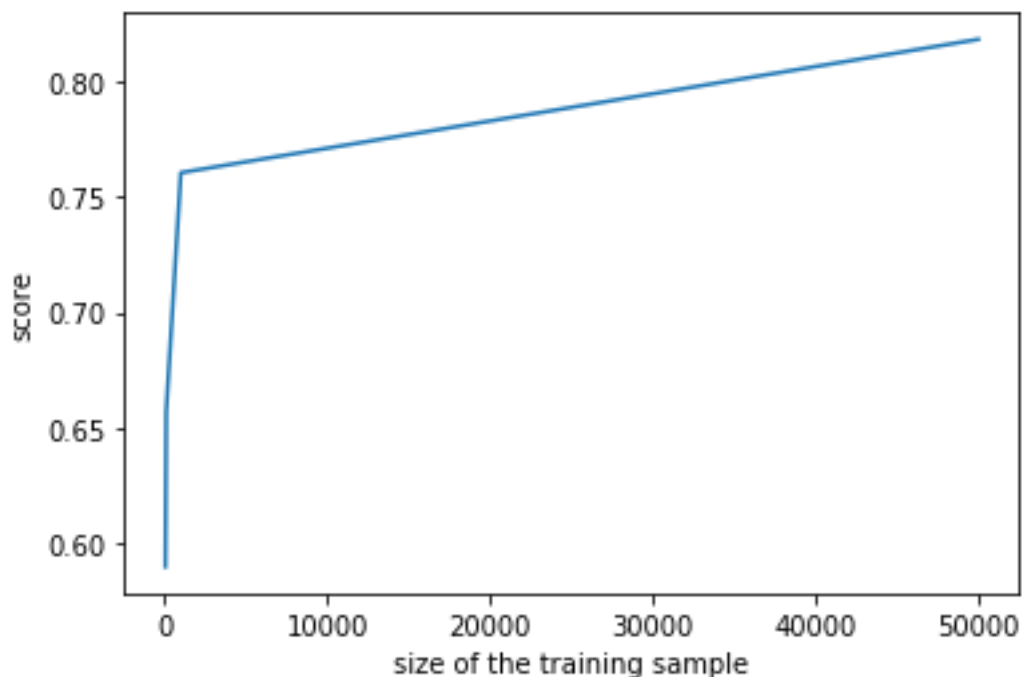


Рисунок 3 – Зависимость точности классификатора от размера обучающей выборки

Министерство образования Республики Беларусь  
Учреждение образования  
"Белорусский Государственный университет информатики  
и радиоэлектроники"

Лабораторная работа №2  
**“Реализация глубокой нейронной сети”**  
по учебной дисциплине “Машинное обучение”

Выполнил:

Студент гр. 956241 Дубовик Н.О.

Минск 2020

**Данные:** В работе предлагается использовать набор данных notMNIST, который состоит из изображений размерностью  $28 \times 28$  первых 10 букв латинского алфавита (A ... J, соответственно). Обучающая выборка содержит порядка 500 тыс. изображений, а тестовая – около 19 тыс.

Данные можно скачать по ссылке:

- [https://commondatastorage.googleapis.com/books1000/notMNIST\\_large.tar.gz](https://commondatastorage.googleapis.com/books1000/notMNIST_large.tar.gz) (большой набор данных);
- [https://commondatastorage.googleapis.com/books1000/notMNIST\\_small.tar.gz](https://commondatastorage.googleapis.com/books1000/notMNIST_small.tar.gz) (маленький набор данных);

Описание данных на английском языке доступно по ссылке:

<http://yaroslavvb.blogspot.sg/2011/09/notmnist-dataset.html>

Результат выполнения заданий опишите в отчете.

### **Задание 1.**

Реализуйте полносвязную нейронную сеть с помощью библиотеки Tensor Flow. В качестве алгоритма оптимизации можно использовать, например, стохастический градиент (Stochastic Gradient Descent, SGD). Определите количество скрытых слоев от 1 до 5, количество нейронов в каждом из слоев до нескольких сотен, а также их функции активации (кусочно-линейная, сигмоидная, гиперболический тангенс и т.д.).

Была создана нейронная сеть с двумя скрытыми слоями, в каждом из которых 128 нейронов с функцией активации ReLU. В выходном слое 10 нейронов с функцией активации “softmax”.

Также был использован оптимизатор Adam и функция потерь “sparse\_categorical\_crossentropy”.

Реализация представлена ниже, здесь количество эпох равняется 100:

```
model=tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(img_height, img_width)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(x_train, y_train, epochs=epochs)
model.evaluate(x_test, y_test)
```

### **Задание 2.**

Как улучшилась точность классификатора по сравнению с логистической регрессией?

По итогам обучения были достигнуты следующие результаты: 94.09% и 90.71% для обучающей и тестовой выборки соответственно. Таким образом, точность классификатора по сравнению с логистической регрессией повысилась на 7%.

### **Задание 3.**

Используйте регуляризацию и метод сброса нейронов (dropout) для борьбы с переобучением. Как улучшилось качество классификации?

Для каждого слоя были добавлены регуляризации с коэффициентом 0.0001 и метод сброса с коэффициентом 0.2:

```
model=tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(img_height, img_width)),
    tf.keras.layers.Dense(128,
activation='relu',kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(128,
activation='relu',kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=epochs)
model.evaluate(x_test, y_test)
```

### **Задание 4.**

Воспользуйтесь динамически изменяемой скоростью обучения (learning rate). Наилучшая точность, достигнутая с помощью данной модели составляет 97.1%. Какую точность демонстрирует Ваша реализованная модель?

Получилось достигнуть 87.34% процента точности на обучающей и 88.82% процента на тестовой выборках.



Министерство образования Республики Беларусь  
Учреждение образования  
"Белорусский Государственный университет информатики  
и радиоэлектроники"

Лабораторная работа №3  
**“Реализация сверточной нейронной сети”**  
по учебной дисциплине “Машинное обучение”

Выполнил:

Студент гр. 956241 Дубовик Н.О.

Минск 2020

**Данные:** В работе предлагается использовать набор данных notMNIST, который состоит из изображений размерностью  $28 \times 28$  первых 10 букв латинского алфавита (A ... J, соответственно). Обучающая выборка содержит порядка 500 тыс. изображений, а тестовая – около 19 тыс.

Данные можно скачать по ссылке:

- [https://commondatastorage.googleapis.com/books1000/notMNIST\\_large.tar.gz](https://commondatastorage.googleapis.com/books1000/notMNIST_large.tar.gz) (большой набор данных);
- [https://commondatastorage.googleapis.com/books1000/notMNIST\\_small.tar.gz](https://commondatastorage.googleapis.com/books1000/notMNIST_small.tar.gz) (маленький набор данных);

Описание данных на английском языке доступно по ссылке:

<http://yaroslavvb.blogspot.sg/2011/09/notmnist-dataset.html>

Результат выполнения заданий опишите в отчете.

### Задание 1.

Реализуйте нейронную сеть с двумя сверточными слоями, и одним полносвязным с нейронами с кусочно-линейной функцией активации. Какова точность построенной модели?

Была создана нейронная сеть, состоящая из двух сверточных слоев, с фильтрами размера  $3 \times 3$  и функцией активации ReLU. Первой сверточный слой состоит из 8 фильтров, в то время как второй из 16.

Второй сверточный слой связан с полносвязным слоем, состоящим из 128 нейронов с функцией активации ReLU.

Выходной слой состоит из 10 нейронов с функцией активации “softmax”.

Был использован оптимизатор Adam и функция потерь “sparse\_categorical\_crossentropy”.

Реализация, здесь количество эпох равно 50:

```
def train_conv(x_train,y_train,epochs,img_height,img_width,save_path):
    model=tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(8,(3,3),activation='relu',input_shape=(img_height, img_width,1)),
        tf.keras.layers.Conv2D(16,(3,3),activation='relu'),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=epochs)
    model.save(save_path)
```

После обучения получились следующие показатели: 99.41% и 92.48% на обучающей и тестовой выборках соответственно.

### **Задание 2.**

Замените один из сверточных слоев на слой, реализующий операцию пулинга (Pooling) с функцией максимума или среднего. Как это повлияло на точность классификатора?

```
def train_conv_pool(x_train,y_train,epochs,img_height,img_width,save_path):
    model=tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(8,(3,3),activation='relu',input_shape=(img_height, img
g_width,1)),
        tf.keras.layers.AveragePooling2D(),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
    ])

    model.compile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])

    model.fit(x_train, y_train, epochs=epochs)
    model.save(save_path)
```

После замены были достигнуты следующие показатели: 98.41% на обучающей и 91.73% на тестовой выборке.

### **Задание 3.**

Реализуйте классическую архитектуру сверточных сетей LeNet-5 (<http://yann.lecun.com/exdb/lenet/>).

Архитектура реализованной сети получилась следующая:

- сверточный слой с 6-ю фильтрами размера 5x5 и функцией активацией ReLU;
- слой, реализующий операцию пулинга, с функцией среднего;
- сверточный слой с 16-ю фильтрами размера 5x5 и функцией активацией ReLU;
- слой, реализующий операцию пулинга, с функцией среднего;
- полносвязанный слой с 120 нейронами и функцией активацией ReLU;
- полносвязанный слой с 84 нейронами и функцией активацией ReLU;
- выходной слой с 10 нейронами и функцией активацией "softmax".

```
def train_LeNet5(x_train,y_train,epochs,save_path):
    model=tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(filters=6, kernel_size=(5, 5), activation='relu', input_shape=(32,32,1)),
        tf.keras.layers.AveragePooling2D(),
        tf.keras.layers.Conv2D(filters=16, kernel_size=(5, 5), activation='relu'),
        tf.keras.layers.AveragePooling2D(),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(120, activation='relu'),
        tf.keras.layers.Dense(84, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    model.fit(x_train, y_train, epochs=epochs)
    model.save(save_path)
```

После обучения получились следующие показатели: 95.58% и 92.77% на обучающей и тестовой выборках соответственно.

#### **Задание 4.**

Сравните максимальные точности моделей, построенных в лабораторных работах 1-3. Как можно объяснить полученные различия?

Исходя их показателей, полученных в ходе лабораторных работ, можно сделать вывод, что сверточные нейронный сети показывают лучший результат из-за более точного извлечения свойств изображения

Министерство образования Республики Беларусь  
Учреждение образования  
"Белорусский Государственный университет информатики  
и радиоэлектроники"

Лабораторная работа №4  
**“Реализация приложения по распознаванию номеров домов”**  
по учебной дисциплине “Машинное обучение”

Выполнил:

Студент гр. 956241 Дубовик Н.О.

Минск 2020

**Данные:** Набор изображений из Google Street View с изображениями номеров домов, содержащий 10 классов, соответствующих цифрам от 0 до 9.

- 73257 изображений цифр в обучающей выборке;
- 26032 изображения цифр в тестовой выборке;
- 531131 изображения, которые можно использовать как дополнение к обучающей выборке;
- В двух форматах:
  - Оригинальные изображения с выделенными цифрами;
  - Изображения размером  $32 \times 32$ , содержащих одну цифру;
- Данные первого формата можно скачать по ссылкам:
  - <http://ufldl.stanford.edu/housenumbers/train.tar.gz> (обучающая выборка);
  - <http://ufldl.stanford.edu/housenumbers/test.tar.gz> (тестовая выборка);
  - <http://ufldl.stanford.edu/housenumbers/extra.tar.gz> (дополнительные данные);
- Данные второго формата можно скачать по ссылкам:
  - [http://ufldl.stanford.edu/housenumbers/train\\_32x32.mat](http://ufldl.stanford.edu/housenumbers/train_32x32.mat) (обучающая выборка);
  - [http://ufldl.stanford.edu/housenumbers/test\\_32x32.mat](http://ufldl.stanford.edu/housenumbers/test_32x32.mat) (тестовая выборка);
  - [http://ufldl.stanford.edu/housenumbers/extra\\_32x32.mat](http://ufldl.stanford.edu/housenumbers/extra_32x32.mat) (дополнительные данные);
- Описание данных на английском языке доступно по ссылке: <http://ufldl.stanford.edu/housenumbers/>

Результат выполнения заданий опишите в отчете.

### Задание 1.

Реализуйте глубокую нейронную сеть (полносвязную или сверточную) и обучите ее на синтетических данных (например, наборы MNIST (<http://yann.lecun.com/exdb/mnist/>) или notMNIST).

Ознакомьтесь с имеющимися работами по данной тематике: англоязычная статья (<http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42241.pdf>), видео на YouTube ([https://www.youtube.com/watch?v=vGPI\\_JvLoN0](https://www.youtube.com/watch?v=vGPI_JvLoN0)).

Для решения поставленной задачи была создана следующая архитектура нейронной сети:

- 8 сверточных слоев;
- 2 полносвязных слоя;
- 6 выходящих слоев;
- первый слой определяет количество цифр в номере дома;

- второй, третий, четвёртый, пятый и шестой выходящий слой определяет первую цифру, вторую, третью, четвёртую и пятую цифру в номере соответственно;
- использовался оптимизатор Adam;
- функция потерь “sparse\_categorical\_crossentropy”.

На рисунке 1 представлена таблица, описывающая получившуюся архитектуру. Данная таблица была получена с помощью метода model.summary().

Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	[(None, 128, 128, 3)]	0	
conv2d (Conv2D)	(None, 128, 128, 48)	3648	input[0][0]
batch_normalization (BatchNormaliza	(None, 128, 128, 48)	192	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 64, 64, 48)	0	batch_normalization[0][0]
dropout (Dropout)	(None, 64, 64, 48)	0	max_pooling2d[0][0]
conv2d_1 (Conv2D)	(None, 64, 64, 64)	76864	dropout[0][0]
batch_normalization_1 (BatchNor	(None, 64, 64, 64)	256	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0	batch_normalization_1[0][0]
dropout_1 (Dropout)	(None, 64, 64, 64)	0	max_pooling2d_1[0][0]
conv2d_2 (Conv2D)	(None, 64, 64, 128)	204928	dropout_1[0][0]
batch_normalization_2 (BatchNor	(None, 64, 64, 128)	512	conv2d_2[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 128)	0	batch_normalization_2[0][0]
dropout_2 (Dropout)	(None, 32, 32, 128)	0	max_pooling2d_2[0][0]
conv2d_3 (Conv2D)	(None, 32, 32, 160)	512160	dropout_2[0][0]
batch_normalization_3 (BatchNor	(None, 32, 32, 160)	640	conv2d_3[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 32, 32, 160)	0	batch_normalization_3[0][0]
dropout_3 (Dropout)	(None, 32, 32, 160)	0	max_pooling2d_3[0][0]
conv2d_4 (Conv2D)	(None, 32, 32, 192)	768192	dropout_3[0][0]
batch_normalization_4 (BatchNor	(None, 32, 32, 192)	768	conv2d_4[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 16, 16, 192)	0	batch_normalization_4[0][0]
dropout_4 (Dropout)	(None, 16, 16, 192)	0	max_pooling2d_4[0][0]
conv2d_5 (Conv2D)	(None, 16, 16, 192)	921792	dropout_4[0][0]

Рисунок 1 – Архитектура созданной нейронной сети

max_pooling2d_5 (MaxPooling2D)	(None, 16, 16, 192)	0	batch_normalization_5[0][0]
dropout_5 (Dropout)	(None, 16, 16, 192)	0	max_pooling2d_5[0][0]
conv2d_6 (Conv2D)	(None, 16, 16, 192)	921792	dropout_5[0][0]
batch_normalization_6 (BatchNor	(None, 16, 16, 192)	768	conv2d_6[0][0]
max_pooling2d_6 (MaxPooling2D)	(None, 8, 8, 192)	0	batch_normalization_6[0][0]
dropout_6 (Dropout)	(None, 8, 8, 192)	0	max_pooling2d_6[0][0]
conv2d_7 (Conv2D)	(None, 8, 8, 192)	921792	dropout_6[0][0]
batch_normalization_7 (BatchNor	(None, 8, 8, 192)	768	conv2d_7[0][0]
max_pooling2d_7 (MaxPooling2D)	(None, 8, 8, 192)	0	batch_normalization_7[0][0]
dropout_7 (Dropout)	(None, 8, 8, 192)	0	max_pooling2d_7[0][0]
flatten (Flatten)	(None, 12288)	0	dropout_7[0][0]
dense (Dense)	(None, 4096)	50335744	flatten[0][0]
dense_1 (Dense)	(None, 4096)	16781312	dense[0][0]
length (Dense)	(None, 6)	24582	dense_1[0][0]
digit1 (Dense)	(None, 11)	45067	dense_1[0][0]
digit2 (Dense)	(None, 11)	45067	dense_1[0][0]
digit3 (Dense)	(None, 11)	45067	dense_1[0][0]
digit4 (Dense)	(None, 11)	45067	dense_1[0][0]
digit5 (Dense)	(None, 11)	45067	dense_1[0][0]
=====			

Рисунок 1, лист 2

## Задание 2.

После уточнения модели на синтетических данных попробуйте обучить ее на реальных данных (набор Google Street View). Что изменилось в модели?

После обучения на реальных данных были получены следующие результаты:

Точность на обучающей выборке составляет 98.64% для первого слоя; 95.60% для второго слоя; 94.35% для третьего слоя; 96.99 % для четвертого слоя; 99.48% для пятого слоя; 99.97% для шестого слоя.

Точность на валидационной выборке составляет 89.47% для первого слоя; 85.35% для второго слоя; 81.51% для третьего слоя; 88.48% для четвертого слоя; 97.72% для пятого слоя; 99.98% для шестого слоя.

Точность на тестовой выборке составляет 80.76% для первого слоя; 82.15% для второго слоя; 80.77% для третьего слоя; 86.03% для четвертого слоя; 99.12% для пятого слоя; 99.98% для шестого слоя.



### Задание 3.

Реализуйте приложение для ОС Android, которое может распознавать цифры в номерах домов, используя разработанный ранее классификатор. Какова доля правильных классификаций?

Нейронная сеть была использована для распознавания четырех изображений с камеры телефона.

В результате только два изображения из четырех были распознаны верно, рисунок 2.



Рисунок 2 – Распознавание сфотографированных номеров домов

Министерство образования Республики Беларусь  
Учреждение образования  
"Белорусский Государственный университет информатики  
и радиоэлектроники"

Лабораторная работа №5  
**“Применение сверточных нейронных сетей (бинарная классификация)”**  
по учебной дисциплине “Машинное обучение”

Выполнил:

Студент гр. 956241 Дубовик Н.О.

Минск 2020

**Данные:** Набор данных DogsVsCats, который состоит из изображений различной размерности, содержащих фотографии собак и кошек. Обучающая выборка включает в себя 25 тыс. изображений (12,5 тыс. кошек: cat.0.jpg, ..., cat.12499.jpg и 12,5 тыс. собак: dog.0.jpg, ..., dog.12499.jpg), а контрольная выборка содержит 12,5 тыс. неразмеченных изображений. Скачать данные, а также проверить качество классификатора на тестовой выборке можно на сайте Kaggle -> <https://www.kaggle.com/c/dogs-vs-cats/data>

Результат выполнения заданий опишите в отчете.

### **Задание 1.**

Загрузите данные. Разделите исходный набор данных на обучающую, валидационную и контрольную выборки.

Данные были разделены в отношении 60/20/20.

### **Задание 2.**

Реализуйте глубокую нейронную сеть с как минимум тремя сверточными слоями. Какое качество классификации получено?

Была создана следующая архитектура нейронной сети:

- 3 сверточных слоя с функцией активации ReLU в каждом;
- в каждом слое есть фильтры размера 3x3, первый слой содержит 32 фильтра, второй содержит 64 фильтра и третий слой содержит 128 фильтров;
- каждый сверточный слой связан с пулинг слоем с функцией среднего;
- третий сверточный слой связан с полносвязным слоем, с функцией активацией ReLU и содержащий 512 нейронов;
- выходной слой состоит из 2 нейронов с функцией активацией “softmax”;
- использовался оптимизатор Adam и функция потерь categorical\_crossentropy.

Реализация, данная модель обучалась на протяжении 10 эпох:

```
tf.keras.layers.Conv2D(32,(3,3),activation='relu',input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)),  
tf.keras.layers.AveragePooling2D(),  
tf.keras.layers.Dropout(0.2),  
  
tf.keras.layers.Conv2D(64,(3,3),activation='relu'),  
tf.keras.layers.AveragePooling2D(),  
tf.keras.layers.Dropout(0.2),
```

```

tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
tf.keras.layers.AveragePooling2D(),
tf.keras.layers.Dropout(0.2),

tf.keras.layers.Flatten(),
tf.keras.layers.Dense(512, activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(2, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

```

Получены следующие результаты: 98.72% на обучающей выборке, 78.55% на тестовой выборке и 74.06% на валидационной выборке, рисунок 1.

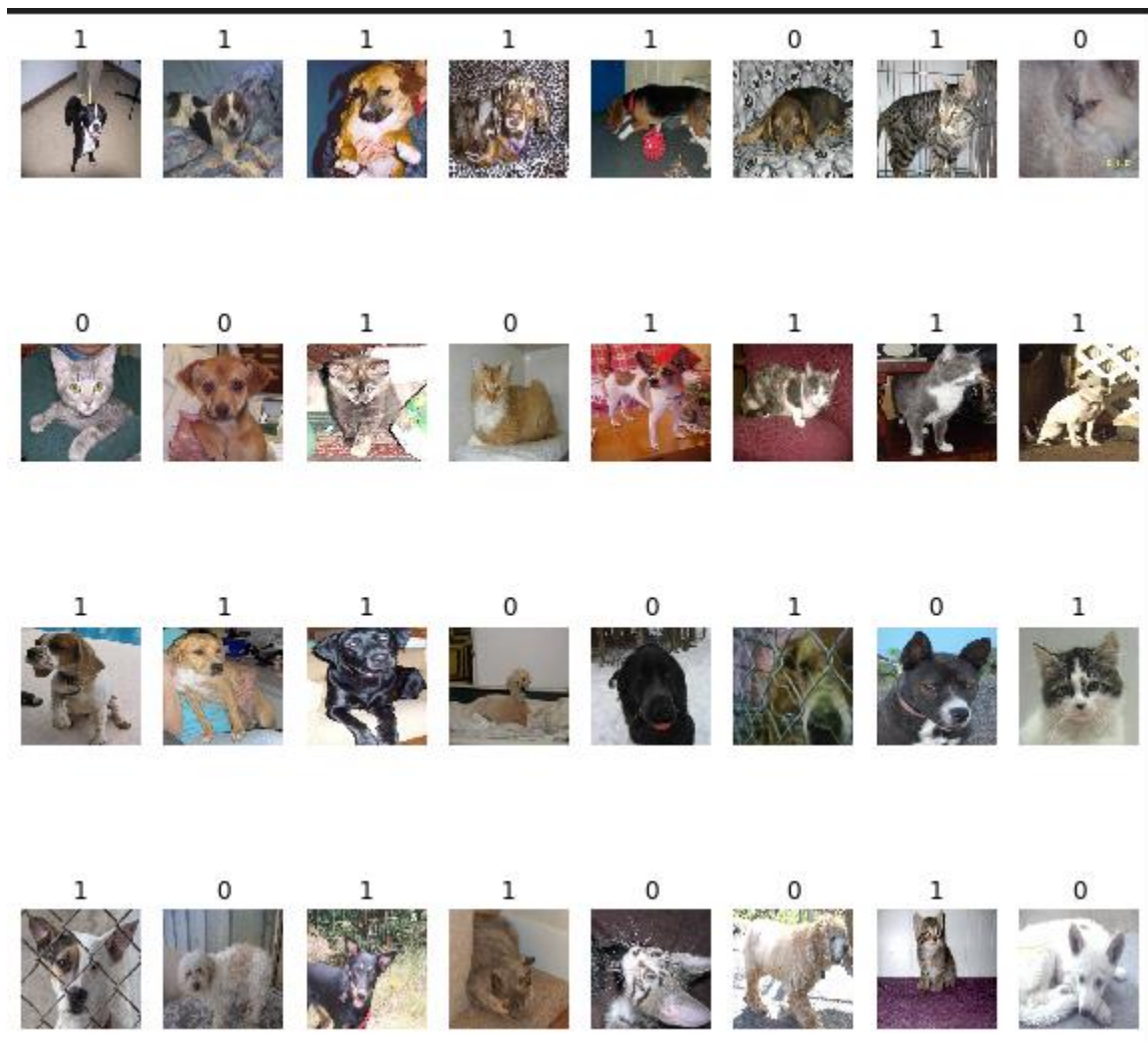


Рисунок 1 – Результат работы глубокой нейронной сети

### Задание 3.

Примените дополнение данных (data augmentation). Как это повлияло на качество классификатора?

Были использованы техники вращения и сдвига изображения:

```
img_augmentation_generator=tf.keras.preprocessing.image.ImageDataGenerator(  
    rotation_range=45,  
    shear_range=2,  
    rescale=1./255  
)
```

После чего были получены следующие результаты: 96.64% на обучающей выборке, 89.6% на тестовой выборке и 85.16% на валидационной выборке, рисунок 2.

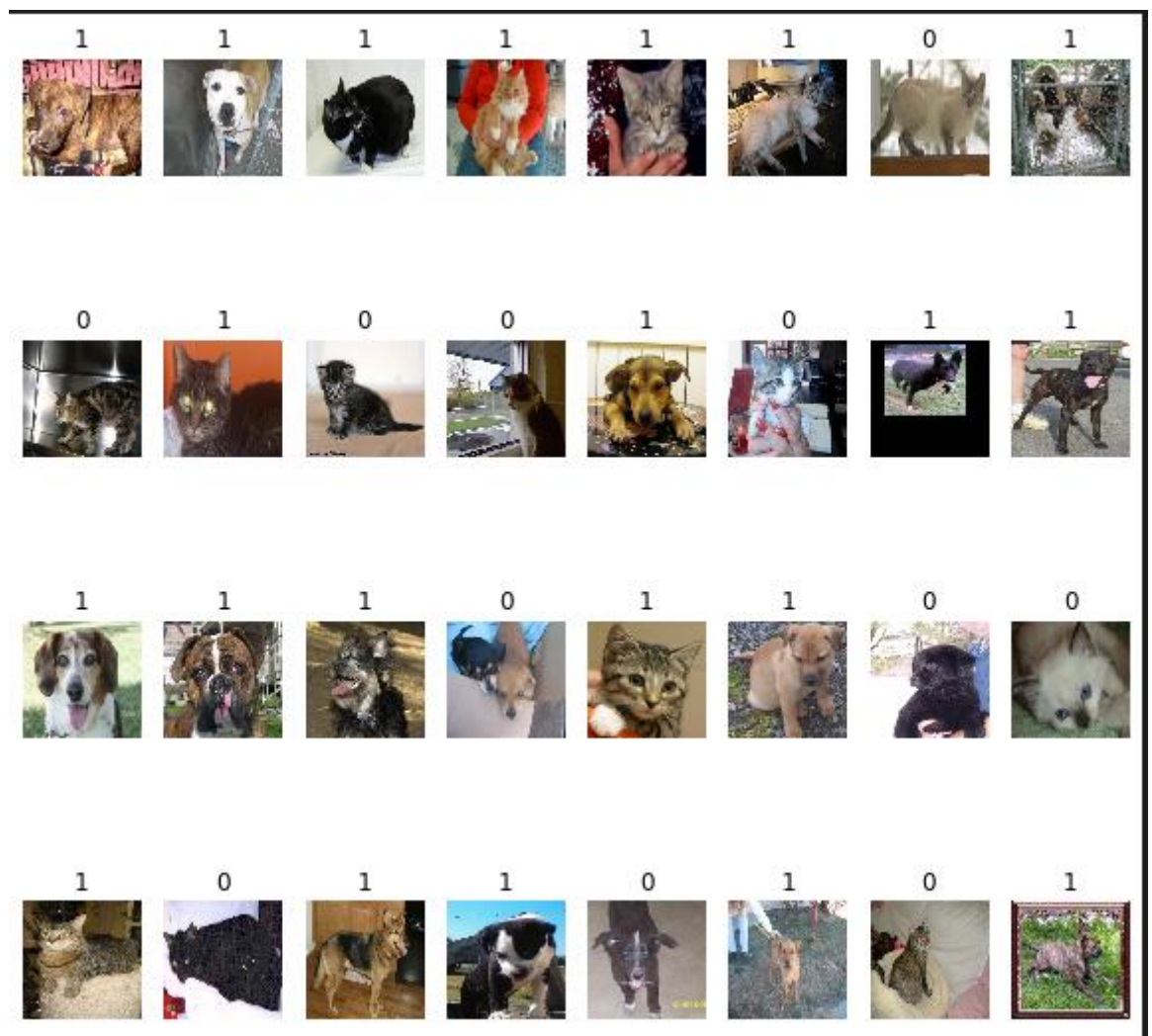


Рисунок 2 – Результат работы глубокой нейронной сети после дополнения данных

#### Задание 4.

Поэкспериментируйте с готовыми нейронными сетями (например, AlexNet, VGG16, Inception и т.п.), применив передаточное обучение. Как это повлияло на качество классификатора?

Какой максимальный результат удалось получить на сайте Kaggle? Почему?

Было использовано передаточное обучение на архитектуре VGG16.

Из данной сети был убран исходящий слой и были добавлены полносвязанный слой, с функцией активацией ReLU и содержащий 512 нейронов, и выходной слой из 2 нейронов с функцией активацией “softmax”. Набор данных обучался в течение 10 эпох.

```
base_model=tf.keras.applications.VGG16(  
    input_shape=(IMAGE_WIDTH,IMAGE_HEIGHT,IMAGE_CHANNELS),  
    include_top=False,  
    weights='imagenet'  
)  
base_model.trainable = False  
  
pretrained_model=tf.keras.models.Sequential([  
    base_model,  
    tf.keras.layers.Flatten(),  
    tf.keras.layers.Dense(512, activation='relu'),  
    tf.keras.layers.Dropout(0.2),  
    tf.keras.layers.Dense(2, activation='softmax')  
)  
  
pretrained_model.compile(optimizer='adam',  
    loss='categorical_crossentropy',  
    metrics=['accuracy'])
```

Были получены следующие результаты: точность на обучающей выборке составляет 98.03%, 95.34% на тестовой выборке и 91.5% на валидационной выборке, рисунок 3.



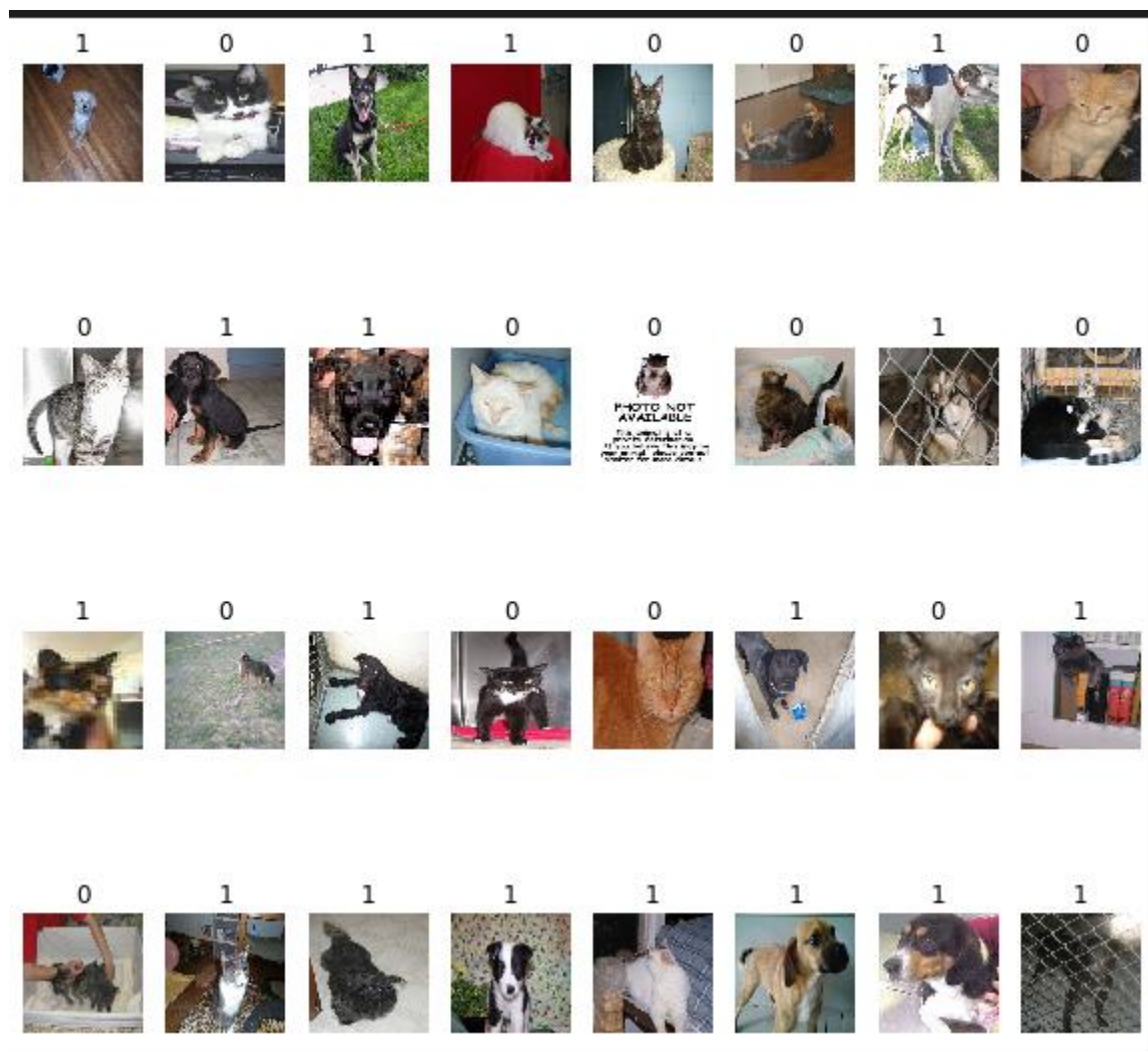


Рисунок 3 – Использование архитектуры VGG16

Министерство образования Республики Беларусь  
Учреждение образования  
"Белорусский Государственный университет информатики  
и радиоэлектроники"

Лабораторная работа №6  
**“Применение сверточных нейронных сетей (многоклассовая  
классификация)”**  
по учебной дисциплине “Машинное обучение”

Выполнил:

Студент гр. 956241 Дубовик Н.О.

Минск 2020



**Данные:** Набор данных для распознавания языка жестов, который состоит из изображений размерности 28x28 в оттенках серого (значение пикселя от 0 до 255). Каждое из изображений обозначает букву латинского алфавита, обозначенную с помощью жеста, как показано на рисунке ниже (рисунок цветной, а изображения в наборе данных в оттенках серого). Обучающая выборка включает в себя 27,455 изображений, а контрольная выборка содержит 7172 изображения. Данные в виде csv-файлов можно скачать на сайте Kaggle -> <https://www.kaggle.com/datamunge/sign-language-mnist>

Результат выполнения заданий опишите в отчете.

### **Задание 1.**

Загрузите данные. Разделите исходный набор данных на обучающую и валидационную выборки.

Данные были разделены в отношении 80 к 20.

### **Задание 2.**

Реализуйте глубокую нейронную сеть со сверточными слоями. Какое качество классификации получено? Какая архитектура сети была использована?

Была создана следующая архитектура нейронной сети:

- 3 сверточных слоя с функцией активации ReLU в каждом;
- в каждом слое есть фильтры размера 3x3, первый слой содержит 8 фильтров, второй содержит 16 фильтров и третий слой содержит 32 фильтра;
- каждый сверточный слой связан с пулингом с функцией среднего;
- третий сверточный слой связан с полносвязным слоем, с функцией активацией ReLU и содержащий 128 нейронов;
- выходной слой состоит из 25 нейронов с функцией активацией “softmax”;
- использовался оптимизатор Adam и функция потерь `sparse_categorical_crossentropy`.

Реализация, данная модель обучалась на протяжении 30 эпох:

```
model=tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(8,(3,3),activation='relu',input_shape=(IMAGE_WIDTH,
    IMAGE_HEIGHT, IMAGE_CHANNELS)),
    tf.keras.layers.AveragePooling2D(),
```

```

tf.keras.layers.Conv2D(16,(3,3),activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.AveragePooling2D(),

tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.AveragePooling2D(),

tf.keras.layers.Flatten(),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(25, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

```

Получены следующие результаты: 95.01% на обучающей выборке и 99.06% на валидационной.

### **Задание 3.**

Примените дополнение данных (data augmentation). Как это повлияло на качество классификатора?

Были использованы техники вращения и сдвига изображения:

```

img_rotation_generator=tf.keras.preprocessing.image.ImageDataGenerator(rotation_range=90)
img_shear_generator=tf.keras.preprocessing.image.ImageDataGenerator(shear_range=45.0)

```

После чего были получены следующие результаты: 93.13% на обучающей выборке и 99.64% на валидационной.

### **Задание 4.**

Поэкспериментируйте с готовыми нейронными сетями (например, AlexNet, VGG16, Inception и т.п.), применив передаточное обучение. Как это повлияло на качество классификатора? Можно ли было обойтись без него?

Какой максимальный результат удалось получить на контрольной выборке?

Было использовано передаточное обучение на архитектуре VGG16.

Из данной сети был убран исходящий слой и были добавлены пулинг слой с функцией среднего, и выходной слой из 25 нейронов с функцией активацией “softmax”. Набор данных обучался в течение 30 эпох.

```
base_model=tf.keras.applications.VGG16(  
    input_shape=(IMAGE_WIDTH,IMAGE_HEIGHT,IMAGE_CHANNELS),  
    include_top=False,  
    weights='imagenet'  
)  
base_model.trainable = False  
pretrained_model=tf.keras.models.Sequential([  
    base_model,  
    tf.keras.layers.GlobalAveragePooling2D(),  
    tf.keras.layers.Dense(25, activation='softmax')  
)  
  
pretrained_model.compile(optimizer='adam',  
    loss='sparse_categorical_crossentropy',  
    metrics=['accuracy'])
```

Были получены следующие результаты: точность на обучающей выборке составляет 97.72% и 97.78% на валидационной.

Министерство образования Республики Беларусь  
Учреждение образования  
"Белорусский Государственный университет информатики  
и радиоэлектроники"

Лабораторная работа №7  
**“Рекуррентные нейронные сети для анализа текста”**  
по учебной дисциплине “Машинное обучение”

Выполнил:

Студент гр. 956241 Дубовик Н.О.

Минск 2020

**Данные:** Набор данных для предсказания оценок для отзывов, собранных с сайта [imdb.com](http://imdb.com), который состоит из 50,000 отзывов в виде текстовых файлов. Отзывы разделены на положительные (25,000) и отрицательные (25,000). Данные предварительно токенизированы по принципу “мешка слов”, индексы слов можно взять из словаря ([imdb.vocab](http://imdb.vocab)). Обучающая выборка включает в себя 12,500 положительных и 12,500 отрицательных отзывов, контрольная выборка также содержит 12,500 положительных и 12,500 отрицательных отзывов, а также. Данные можно скачать по ссылке <https://ai.stanford.edu/~amaas/data/sentiment/>

Результат выполнения заданий опишите в отчете.

### Задание 1.

Загрузите данные. Преобразуйте текстовые файлы во внутренние структуры данных, которые используют индексы вместо слов.

Данные были загружены с помощью библиотеки `tensorflow_datasets`

```
dataset, info = tfds.load('imdb_reviews/subwords8k', with_info=True,
                          as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
encoder = info.features['text'].encoder
train_dataset = train_dataset.shuffle(1000).padded_batch(BATCH_SIZE, padded_shapes=((None,), ()))
test_dataset = test_dataset.padded_batch(BATCH_SIZE, padded_shapes=((None,), ()))
```

### Задание 2.

Реализуйте и обучите двунаправленную рекуррентную сеть (LSTM или GRU). Какого качества классификации удалось достичь?

Созданная нейронная сеть состоит из слоя LSTM. Выходной слой состоит из 1 нейрона с функцией активацией “sigmoid”. Использовался оптимизатор Adam и функция потерь `binary_crossentropy`. Набор данных обучался в течение 20 эпох.

```
model_1 = tf.keras.Sequential([
    tf.keras.layers.Embedding(encoder.vocab_size, 100),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
model_1.compile(optimizer='adam',  
    loss='binary_crossentropy',  
    metrics=['accuracy'])
```

Были получены следующие результаты: точность на обучающей выборке составляет 98,98%, 84.53% на валидационной, 84.03% на тестовой.

### **Задание 3.**

Используйте индексы слов и их различное внутреннее представление (word2vec, glove). Как влияет данное преобразование на качество классификации?

Было использовано представление glove.

Для этого использовались следующие данные - <http://nlp.stanford.edu/data/glove.840B.300d.zip>

Реализация представлена ниже:

```
embeddings_index = { }  
with open("glove.840B.300d.txt", "r") as in_file:  
    for line in in_file:  
        values = line.split()  
  
        try:  
            word = values[0]  
            embeddings_index[word] = np.asarray(values[1:], dtype=np.float32)  
        except:  
            pass  
  
embedding_matrix = np.zeros((encoder.vocab_size, 300))  
  
for index, word in enumerate(encoder.subwords, 1):  
    word = word.lower()  
  
    if word.endswith("_"):  
        word = word[:-1]  
  
    embedding_vector = embeddings_index.get(word)  
    if embedding_vector is not None:  
        embedding_matrix[index] = embedding_vector
```

И обучена следующая модель

```

model = tf.keras.Sequential([
    tf.keras.layers.Embedding(encoder.vocab_size, 300, weights=[embedding_matrix], trainable=False),
    tf.keras.layers.Bidirectional(tf.keras.layers.GRU(64, return_sequences=True),
    merge_mode='concat'),
    tf.keras.layers.Bidirectional(tf.keras.layers.GRU(64), merge_mode='concat'),
    tf.keras.layers.Dense(64, activation='elu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1)
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

```

В ходе чего были получены следующие результаты: 85.28% на обучающей выборке и 86.46% на валидационной.

#### **Задание 4.**

Поэкспериментируйте со структурой сети (добавьте больше рекуррентных, полносвязных или сверточных слоев). Как это повлияло на качество классификации?

Были добавлены 1 рекуррентный слой LSTM и 1 полносвязанный слой с 64 нейронами и функцией активацией relu.

Результаты: точность на обучающей выборке составляет 99,83%, 86,04% на валидационной, 86,76% на тестовой.

#### **Задание 5.**

Используйте предобученную рекуррентную нейронную сеть (например, DeerpMoji или что-то подобное).

Какой максимальный результат удалось получить на контрольной выборке?

Была использована сеть DeerpMoji, для этого использовалась библиотека deerpmoji.model\_def и ее метод deerpmoji\_architecture, а также предоставляемый пример, связанный с imdb.

После применения данной сети получилось добиться 82.15% точности на контрольной выборке

Министерство образования Республики Беларусь  
Учреждение образования  
"Белорусский Государственный университет информатики  
и радиоэлектроники"

Лабораторная работа №8  
**“Рекуррентные нейронные сети для анализа временных рядов”**  
по учебной дисциплине “Машинное обучение”

Выполнил:

Студент гр. 956241 Дубовик Н.О.

Минск 2020



**Данные:** Набор данных для прогнозирования временных рядов, который состоит из среднемесячного числа пятен на солнце, наблюдаемых с января 1749 по август 2017. Данные в виде csv-файла можно скачать на сайте Kaggle -> <https://www.kaggle.com/robervalt/sunspots/>

Результат выполнения заданий опишите в отчете.

### Задание 1.

Загрузите данные. Изобразите ряд в виде графика. Вычислите основные характеристики временного ряда (сезонность, тренд, автокорреляцию).

На рисунке 1 изображен график ряда, на рисунке 2 был построен и изображен тренд, на рисунке 3 была найдена сезонность, на рисунке 4 была вычислена автокорреляция.

```
plt.figure(figsize=(12,6))
plt.plot(dataset[DATE_COLUMN_NAME], dataset[VALUE_COLUMN_NAME], "-")
plt.xlabel("Year")
plt.ylabel("Value")
plt.grid(True)
```

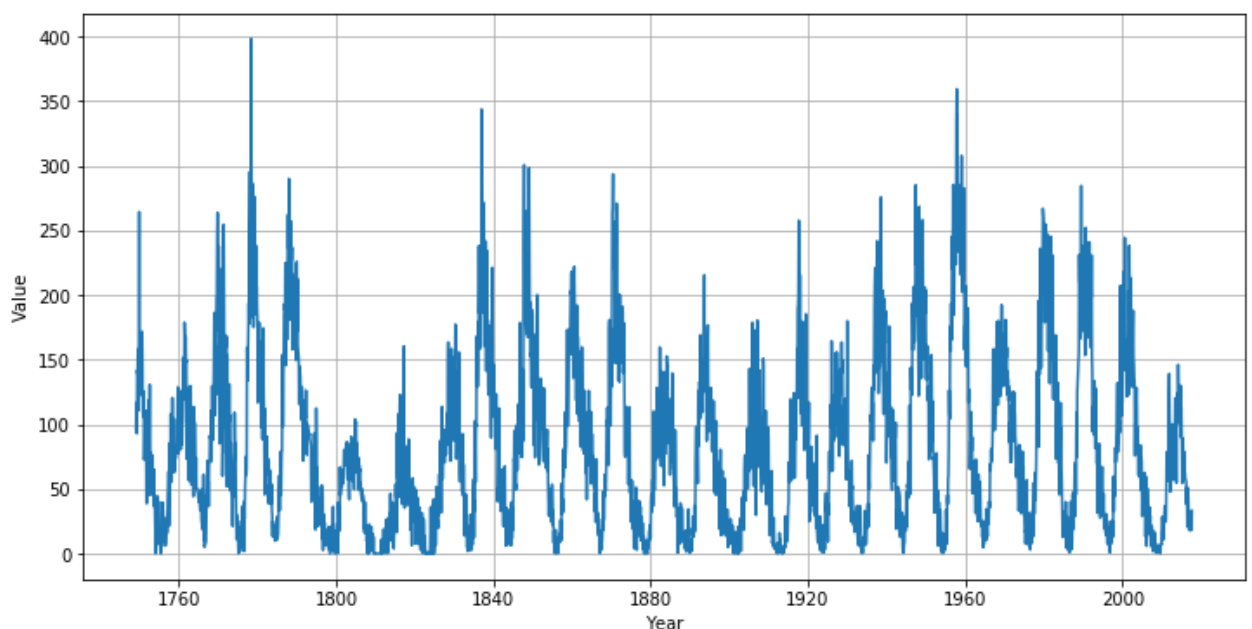


Рисунок 1 – График ряда

```
sunspot_number=dataset[VALUE_COLUMN_NAME]
trend=sunspot_number.rolling(12).mean()
plt.figure(figsize=(12,6))
```

```
plt.plot(dataset[DATE_COLUMN_NAME], trend, "-")
plt.xlabel("Year")
plt.ylabel("Value")
plt.grid(True)
```

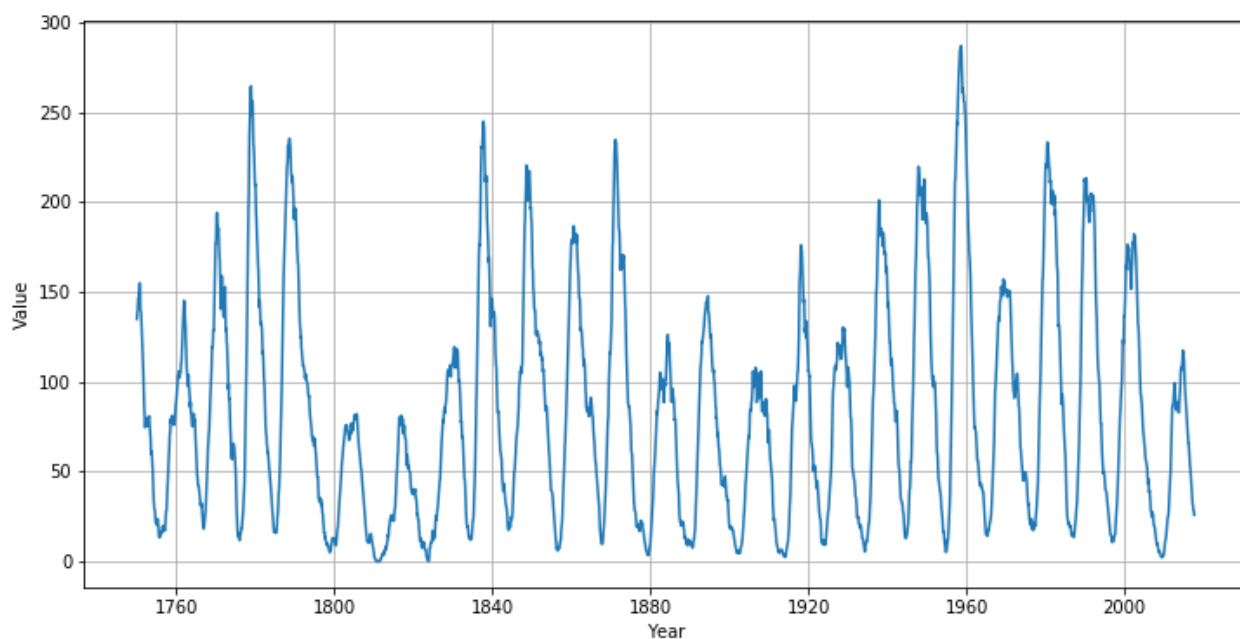


Рисунок 2 – Тренд

```
import statsmodels.tsa.seasonal as seasonal
dataset.index = dataset[DATE_COLUMN_NAME]
decomposition = seasonal.seasonal_decompose(dataset[VALUE_COLUMN_NAME], model='additive')
```

```
plt.figure(figsize=(20,4))
plt.plot(decomposition.seasonal)
plt.ylabel("Value")
plt.xlabel("Year")
```

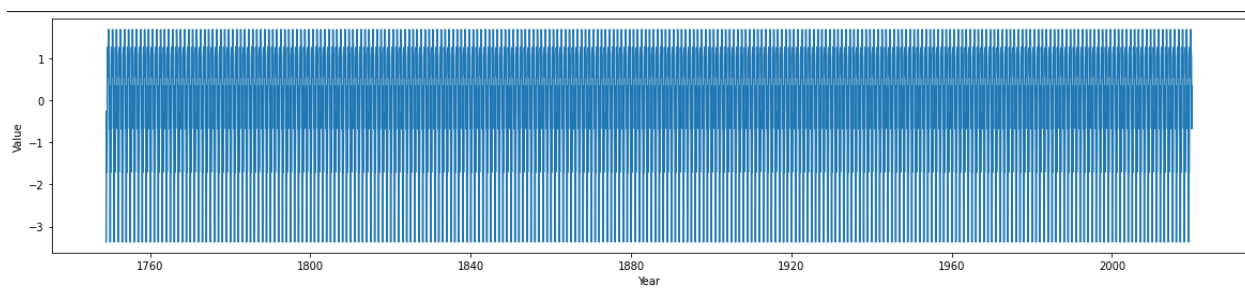


Рисунок 3 – Сезонность

```
plt.figure(figsize=(16,8))
pd.plotting.autocorrelation_plot(dataset[VALUE_COLUMN_NAME])
```

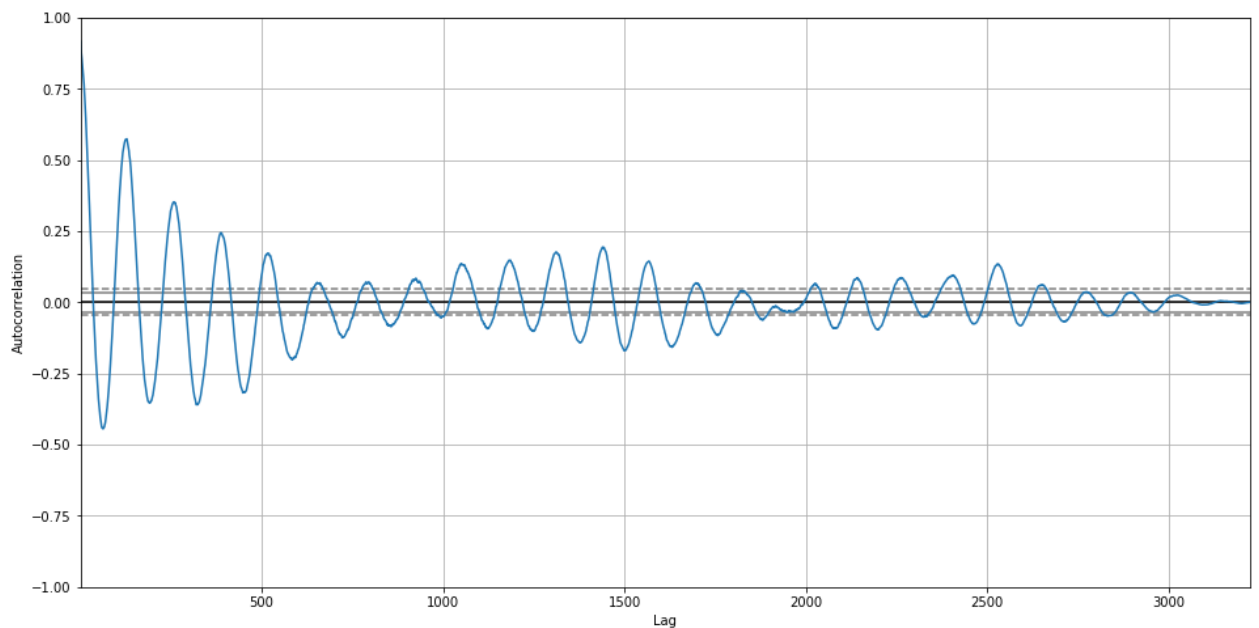


Рисунок 4 – Автокорреляция

### Задание 2.

Для прогнозирования разделите временной ряд на обучающую, валидационную и контрольную выборки.

Данные были разделены в отношении 60/20/20.

### Задание 3.

Примените модель ARIMA для прогнозирования значений данного временного ряда.

Были применены следующие гиперпараметры при работе с ARIMA, рисунок 5:

Были использованы параметры  $p=5$ ,  $d=1$ ,  $q=0$ , где

$p$ : Порядок авторегрессии тренда

$d$ : Порядок изменения тренда

$q$ : Тренд скользящей средней

```
model = ARIMA(train_dataset[VALUE_COLUMN_NAME], (9,0,1), dates
=train_dataset[DATE_COLUMN_NAME])
```

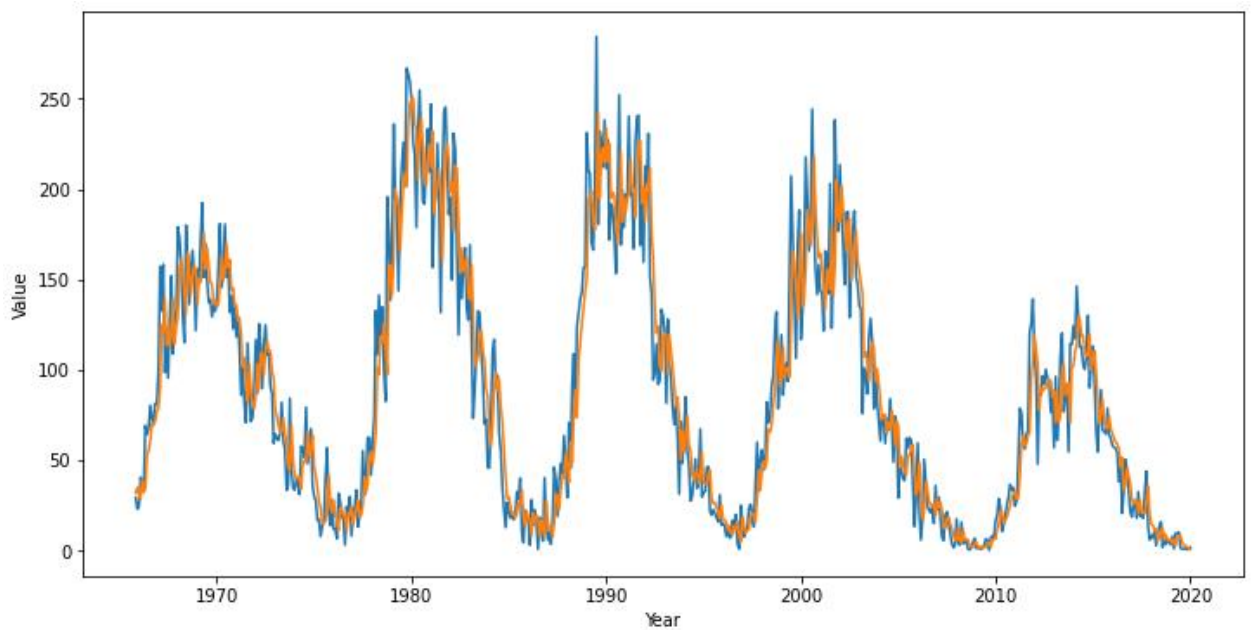


Рисунок 5 – График ряда и прогноза ARIMA

#### Задание 4.

Повторите эксперимент по прогнозированию, реализовав рекуррентную нейронную сеть (с как минимум 2 рекуррентными слоями).

Была создана сеть, состоящая из 2 слоев LSTM. В выходном слое один нейрон. Также были применены оптимизатор Adam и функция потерь “mae”.

Реализация:

```
model = tf.keras.Sequential([
    tf.keras.layers.LSTM(8, input_shape=x_train.shape[-
2:], return_sequences=True),
    tf.keras.layers.LSTM(8),
    tf.keras.layers.Dense(1)
])
model.compile(optimizer='adam', loss='mae')
```

Результат обучения на рисунках 6 и 7.

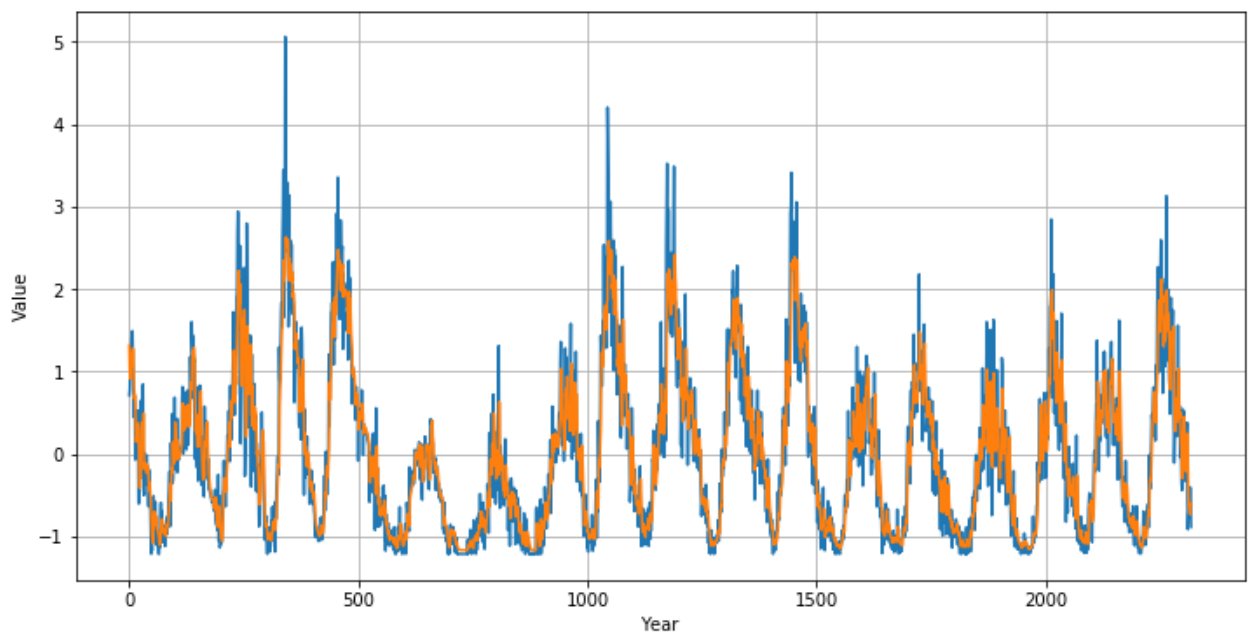


Рисунок 6 – График валидационного ряда и прогноза нейронной сети

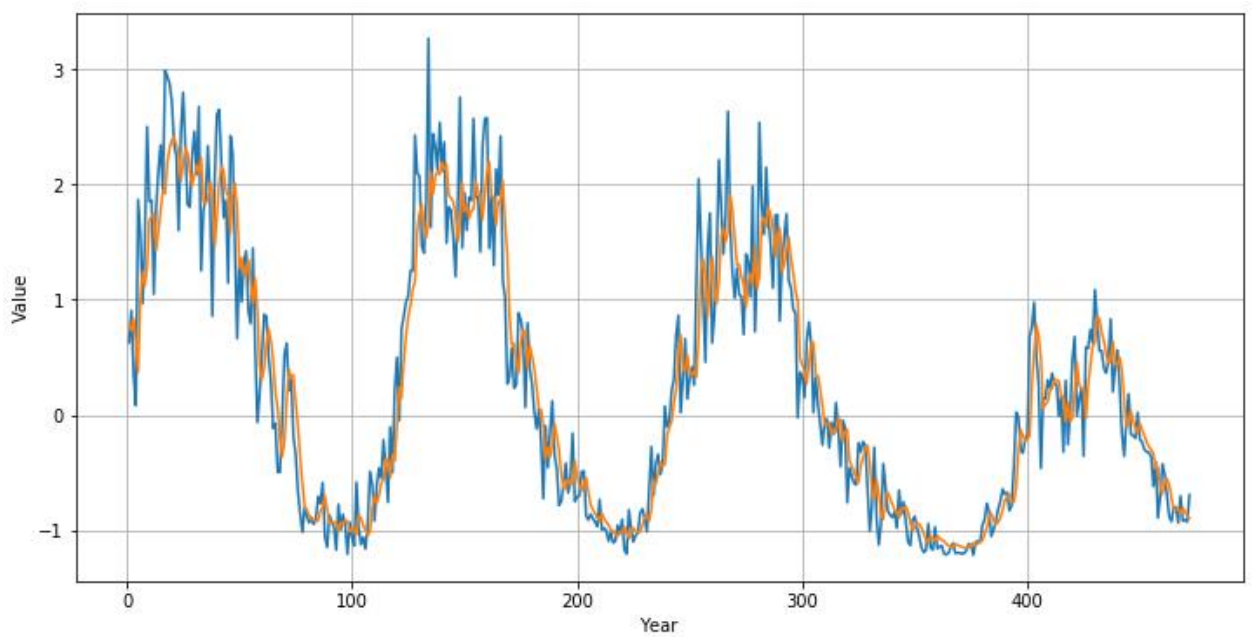


Рисунок 7 – График тестового ряда и прогноза нейронной сети

### Задание 5.

Сравните качество прогноза моделей.

Наилучший результат удалось получить на модели ARIMA.