Министерство науки и высшего образования Российской Федерации

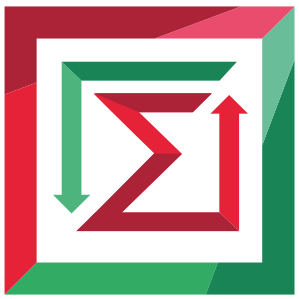
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ

Кафедра теоретической и прикладной информатики

Лабораторная работа №5
по дисциплине «Основы теории машинного обучения»

КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ



ФАКУЛЬТЕТ:	ПМИ
ГРУППА:	ПМИМ-01
СТУДЕНТЫ:	Ершов П.К. Малышкина Е.Д. Слободчикова А.Э.
ВАРИАНТ:	4
ПРЕПОДАВАТЕЛЬ:	Попов А.А.

Новосибирск

2021

1. Цель.

Получить практические навыки по решению задачи классификации изображений с применением свёрточных нейронных сетей (CNN).

2. Содержание работы.

1. Ознакомление с теоретическими основами используемых архитектур нейронных сетей (НС).
2. Ознакомление с возможностями их реализации в рамках свободных библиотек типа TensorFlow.
3. Поиск или формирование необходимых датасетов для выбранной задачи.
4. Кодирование и отладка программы. Обучение и тестирование НС.
5. Написание отчета.
6. Защита лабораторной работы.

3. Теоретическая часть

Свёрточные нейронные сети (CNN) – это специальная архитектура нейронных сетей, предложенная Яном Лекуном в 1988 для распознавания образов. Суть архитектуры в чередовании свёрточных слоёв, то есть слоёв, которые выделяют определённые признаки, и слоёв пулинга, то есть слоёв, которые уплотняют данные до менее подробных и тем самым позволяют не переобучаться сети.

В данной работе используется сеть из 13 слоёв.

В качестве слоёв свёртки используются 2 последовательно расположенных Conv2D.

В качестве слоёв пулинга используются MaxPool2D, идущий сразу за слоями свёртки.

В качестве слоя активации используется Dropout, которые помогает решить проблему переобучения.

Функцией активации свёрточных слоёв выберем ReLu, так как она является хорошим аппроксиматором.

Для выходного слоя функцией активации выберем softmax, так как её выходные значения нейронов зависят от суммы предыдущих нейронов.

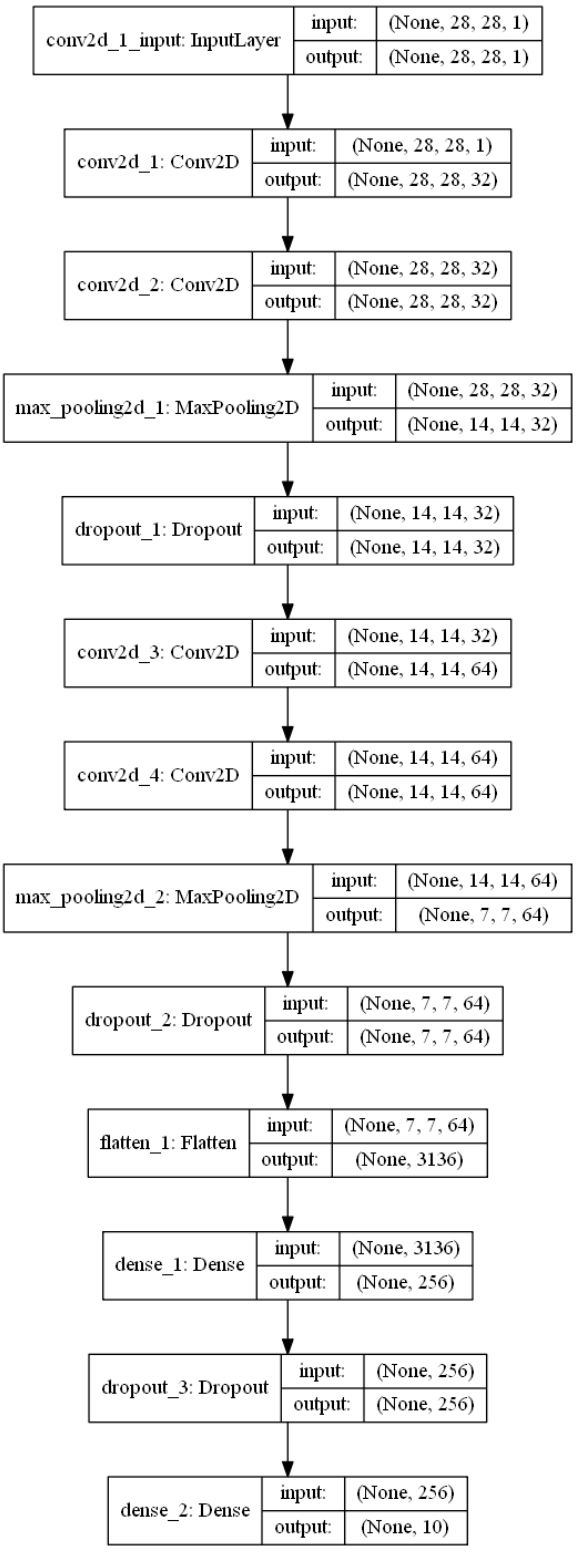


Рисунок 1. Структура нейронной сети

Обучение будет проводиться с помощью MNIST - набора данных, представляющих рукописные цифровые изображения.

Число эпох 10. Число батчей в эпохах 256.

4. Ход работы

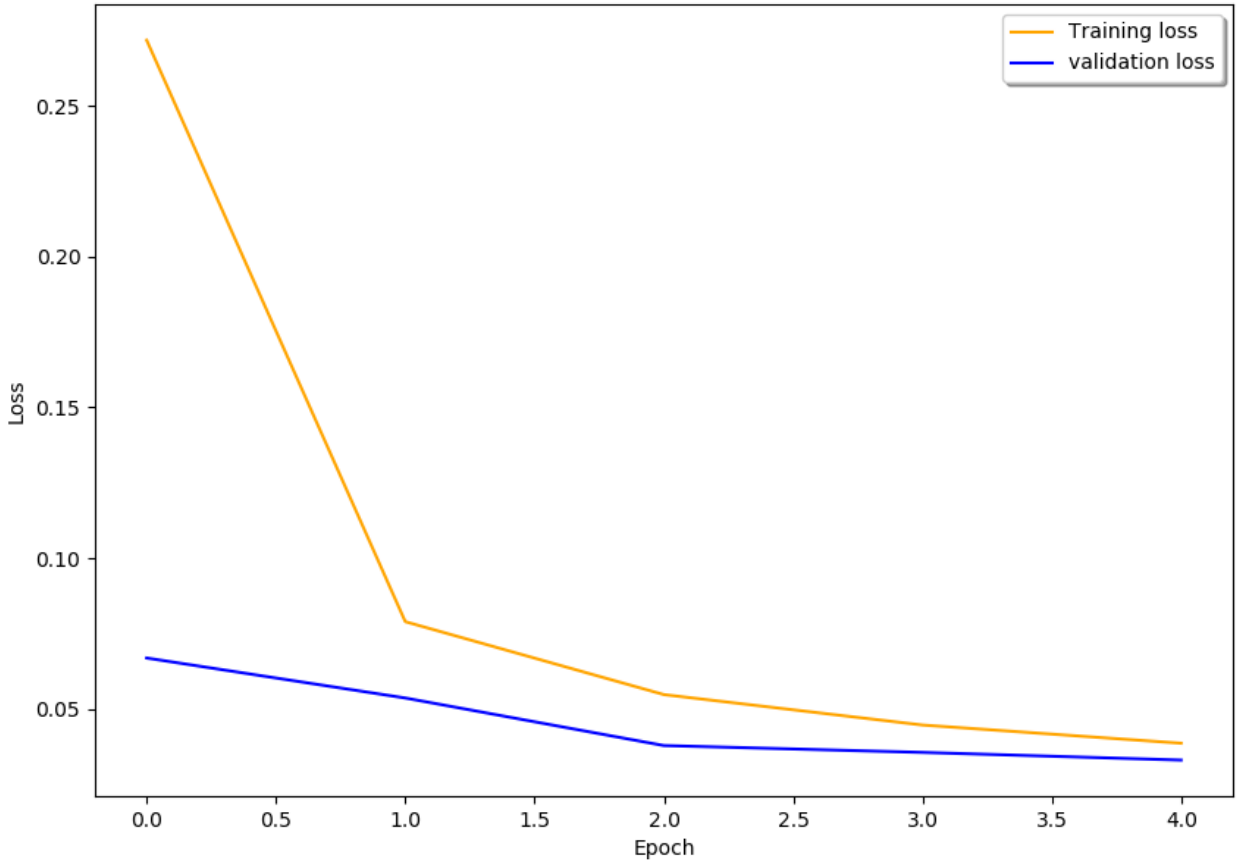


Рисунок 2. График функции потерь (синим – тренировочная выборка, оранжевым – верификационная)

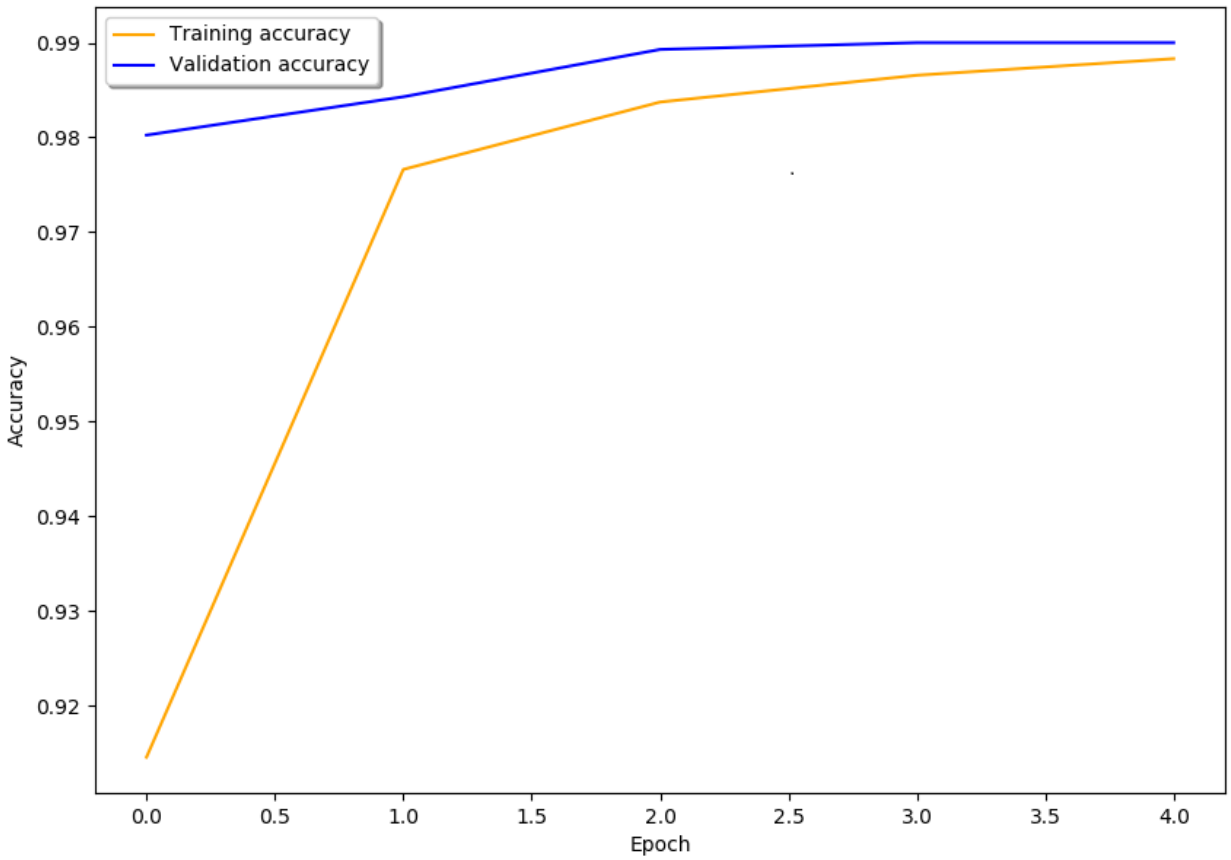


Рисунок 3. График точности

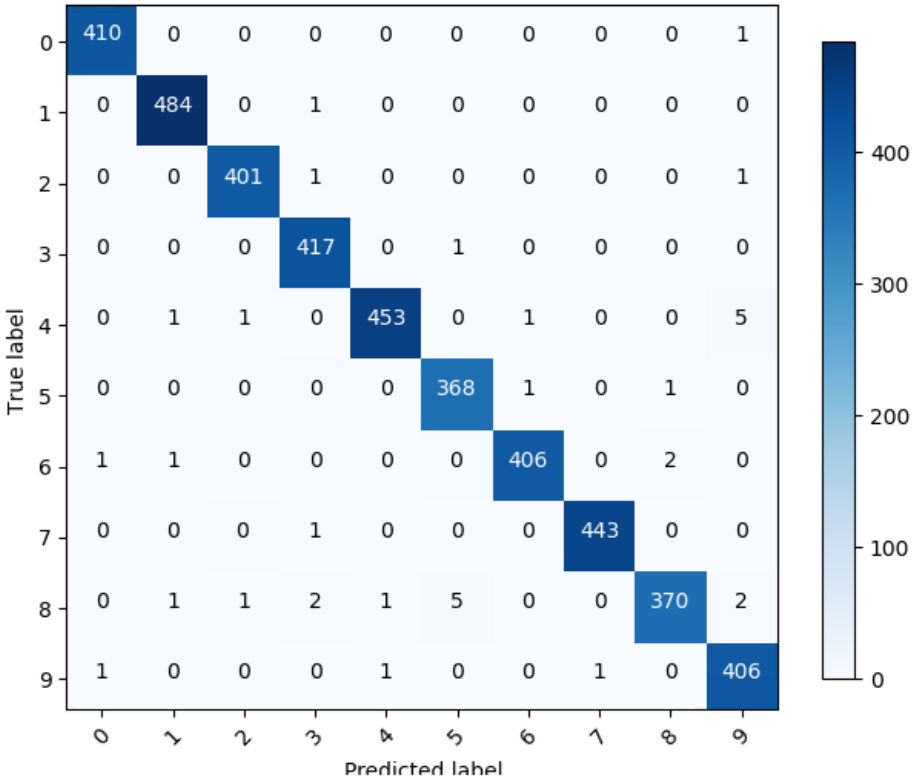


Рисунок 4. Матрица ошибок.

Исходя из результатов в матрице ошибок можно сделать вывод, что модель достаточно точная. Наблюдаются некоторые затруднения с 8 и 9, что подтверждается данными по ошибками на рисунке 5.

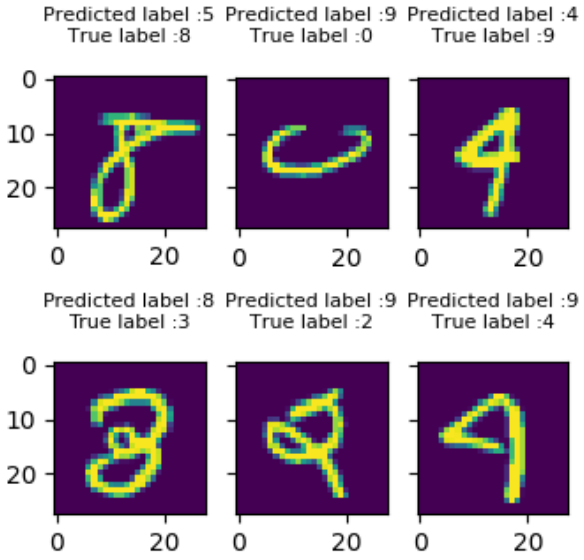


Рисунок 5. Ошибки классификации

5. Текст программы:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns

np.random.seed(2)

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from keras.utils.vis_utils import plot_model

# Загрузка данных
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
Y_train = train["label"]

# Отбросить столбец с надписью "label"
X_train = train.drop(labels = ["label"],axis = 1)

del train

Y_train.value_counts()

# Проверка данных
X_train.isnull().any().describe()
test.isnull().any().describe()

# Нормализуем данные
X_train = X_train / 255.0
test = test / 255.0

# Изменение формы изображения в 3-х измерениях (высота = 28 пикселей, ширина = 28
пикселей, канал = 1)
X_train = X_train.values.reshape(-1,28,28,1)
test = test.values.reshape(-1,28,28,1)

# Кодировать метки в один вектор
Y_train = to_categorical(Y_train, num_classes = 10)

random_seed = 2

# Разделение тренировочного набора (90%) и набора проверки (10%)
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1,
random_state=random_seed)

# Пример изучаемого объекта
g = plt.imshow(X_train[0][:,:,0])

# Установка слоев модели
```

```
model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same', activation
='relu', input_shape = (28,28,1)))
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same', activation
='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))


model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same', activation
='relu'))
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same', activation
='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))


model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation = "softmax"))

# Определение оптимизации
optimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)

model.compile(optimizer = optimizer , loss = "categorical_crossentropy",
metrics=["accuracy"])
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)

learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience=3,
verbose=1, factor=0.5, min_lr=0.00001)
epochs = 10
batch_size = 256

history = model.fit(X_train, Y_train, batch_size = batch_size, epochs = epochs,
          validation_data = (X_val, Y_val), verbose = 2)


def plot_loss(history):

    plt.figure(figsize=(10, 7))

    plt.plot(history.history['loss'], color='orange', label="Training loss")
    plt.plot(history.history['val_loss'], color='b', label="validation loss")
    plt.legend(loc='best', shadow=True)
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.savefig('loss_plot.png')


def plot_acc(history):

    plt.figure(figsize=(10, 7))
    plt.plot(history.history['accuracy'], color='orange', label="Training accuracy")
    plt.plot(history.history['val_accuracy'], color='b',label="Validation accuracy")
    plt.legend(loc='best', shadow=True)
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.savefig('accuracy_plot.png')
```

```
def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix',
cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.savefig('loss_matrix.png')

# Предсказание значения из набора данных проверки
Y_pred = model.predict(X_val)
# Преобразование классов прогнозов в одни векторы
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Преобразование проверочных наблюдений в одни векторы
Y_true = np.argmax(Y_val,axis = 1)
# Вычисление матрицы ошибок
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# Построение матрицы ошибок
plot_confusion_matrix(confusion_mtx, classes = range(10))

errors = (Y_pred_classes - Y_true != 0)

Y_pred_classes_errors = Y_pred_classes[errors]
Y_pred_errors = Y_pred[errors]
Y_true_errors = Y_true[errors]
X_val_errors = X_val[errors]

def display_errors(errors_index,img_errors,pred_errors, obs_errors):
    n = 0
    nrows = 2
    ncols = 5
    fig, ax = plt.subplots(nrows,ncols,sharex=True,sharey=True)
    for row in range(nrows):
        for col in range(ncols):
            error = errors_index[n]
            ax[row,col].imshow((img_errors[error]).reshape((28,28)))
            ax[row,col].set_title("Predicted label :{}\nTrue label
:{}\n".format(pred_errors[error],obs_errors[error]), fontdict={'fontsize': 8,
'fontweight': 'medium'})
            n += 1
    plt.savefig('error.png')

# Вероятности неверных предсказанных чисел
Y_pred_errors_prob = np.max(Y_pred_errors,axis = 1)

# Прогнозируемые вероятности истинных значений в наборе ошибок
true_prob_errors = np.diagonal(np.take(Y_pred_errors, Y_true_errors, axis=1))
```

```
# Разница между вероятностью предсказанного значения и истинным
delta_pred_true_errors = Y_pred_errors_prob - true_prob_errors

# Отсортированный список ошибок
sorted_dela_errors = np.argsort(delta_pred_true_errors)

# 6 ошибок
most_important_errors = sorted_dela_errors[-10:]

# Демонстрация данных ошибок
display_errors(most_important_errors, X_val_errors, Y_pred_classes_errors,
Y_true_errors)

# Помещение результатов предсказаний в файл .csv
# Предсказанные результаты
results = model.predict(test)

results = np.argmax(results,axis = 1)

results = pd.Series(results,name="Label")

submission = pd.concat([pd.Series(range(1,28001),name = "ImageId"),results],axis = 1)

submission.to_csv("cnn_mnist_datagen.csv",index=False)

plot_loss(history)
plot_acc(history)
```