Санкт-Петербургский государственный морской технический университет

Отчёт

по практической работе №1

«Введение в фреймворки и настройки сетей»

Выполнил: Студент 2 курса

Факультета Цифровых

Промышленных Технологий

группы 20221 Хохлов Д.Р.

Преподаватель:

Кафедра киберфизических систем

Кайнова Татьяна Денисовна

Санкт-Петербург

2025

Оглавление

[Введение 2](#_Toc191287017)

[Постановка задачи 3](#_Toc191287018)

[Действия для выполнения задачи 3](#_Toc191287019)

[Библиотеки TensorFlow и Keras 3](#_Toc191287020)

[Датасет «MNIST» 3](#_Toc191287021)

[Нормализация данных из датасета 4](#_Toc191287022)

[Построение нейросети 4](#_Toc191287023)

[Результаты обучения нейросети 5](#_Toc191287024)

[Тестирование нейросети 6](#_Toc191287025)

[Заключение 7](#_Toc191287026)

[Приложение А 8](#_Toc191287027)

# Введение

Цель работы: изучение основ работы с фреймворками для глубокого обучения, такими как TensorFlow и Keras, а также исследование влияния различных методов инициализации, нормализации и архитектуры сети на процесс обучения нейронных сетей.

## Постановка задачи

Задачей данной лабораторной работы заключается в реализации программы для классификации рукописных цифр с использованием фреймворка TensorFlow/Keras в соответствии с вариантом задания. Проанализировать результаты обучения нейросети.

## Действия для выполнения задачи

Для выполнения задачи необходимо:

1. Установить и импортировать библиотеки Tensorflow и Keras для работы с нейросетями
2. Установить и импортировать датасет «MNIST»
3. Нормализовать данные из датасета
4. Собрать нейросеть с помощью импортированных библиотек
5. Обучить нейросеть на датасете
6. Вывести график обучения

## Библиотеки TensorFlow и Keras

В работе используются такие библиотеки как Tensorflow и Keras.

Tensorflow – это открытая библиотека для машинного обучения разработанная компанией Google.

Keras — это открытая библиотека для языка программирования Python, которая предназначена для глубокого машинного обучения. Библиотека Keras уже включена в Tensorflow и устанавливать её отдельно не требуется.

## Датасет «MNIST»

Датасет «MNIST» - набор данных, используемый для обучения и тестирования алгоритмов машинного обучения в задачах распознавания рукописных цифр.

Датасет состоит из 60000 изображений с рукописными цифрами, размером 28х28 пикселей (Рисунок 1).

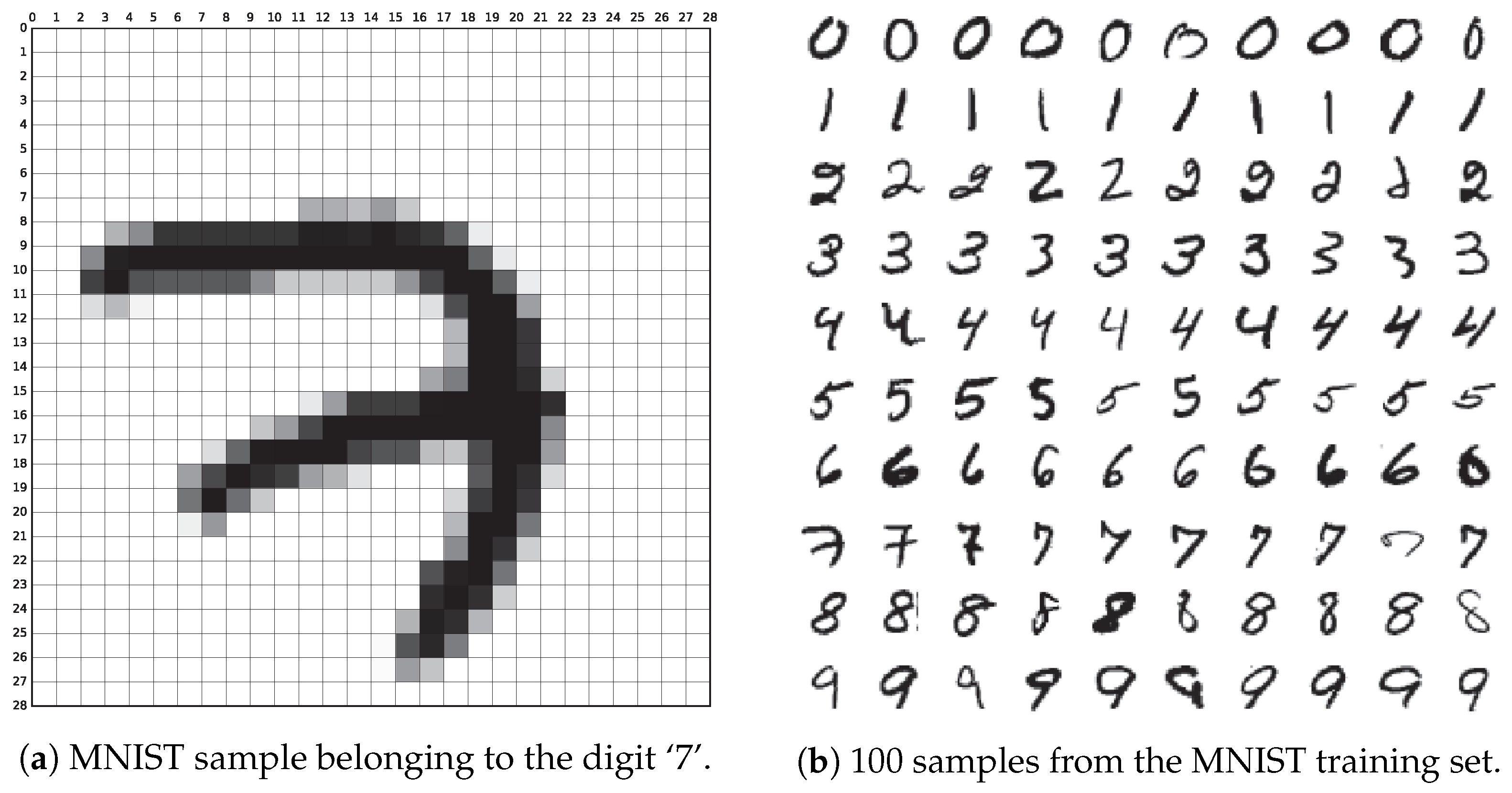


Рисунок 1 – Пример данных из датасета

Установка датасета уже встроена в библиотеку Tensorflow, она импортируется через встроенные методы.

## Нормализация данных из датасета

Каждый элемент подписи следует переформировать из строчного типа в тип вектора, состоящего из 10 значений 0 или 1, где 1 будет стоять под элементом подписи из датасета, а 0 – остальные цифры (Рисунок 2).

Рисунок 2 – пример переформирования подписи в вектор

Перевести строчный тип в тип вектора можно при помощи функции to\_categorical() из библиотеки Tensorflow.

## Построение нейросети

Нейросеть собирается с помощью классов и методов библиотеки Tensorflow и Keras.

Структура нейросети (Рисунок 3):

* Входного слоя из 784 нейронов
* 4-х скрытых слоёв (128/64/32/16 – количество нейронов в каждом слое)
* Выходного слоя из 10 нейронов

Для предотвращения насыщения нейронов после каждой функции активации в скрытых слоях используется пакетная нормализация (BatchNormalization).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Структура нейросети

В качестве функции потерь используется функция CCE (Categorical Cross-Entropy).

Для оптимизации и ускорения обучения нейросети используется оптимизатор SGD (Stochastic gradient descent).

В качестве инициализатора выбран инициализатор «He».

## Результаты обучения нейросети

После 15 эпох обучения нейросеть выдала следующие результаты (Рисунок 4):

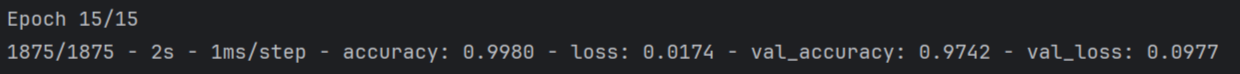


Рисунок 4 – Результаты обучения

Был получен график изменения ошибки во время изменения эпох (Рисунок 5).

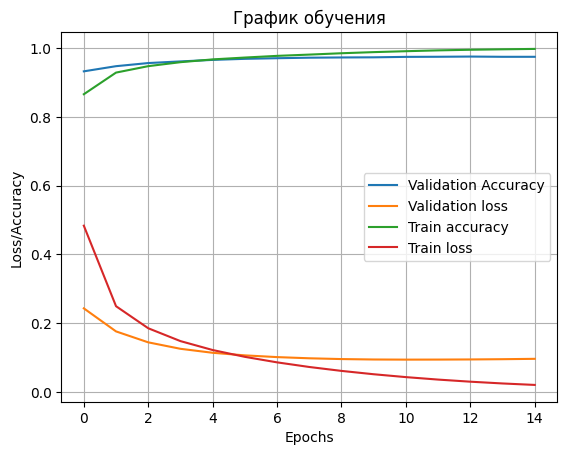


Рисунок 5 – График изменения ошибки во время обучения

Как видно на графике, тренировочная ошибка и тестовая ошибка уменьшается во время обучения, а точность в обоих случаях увеличивается.

## Тестирование нейросети

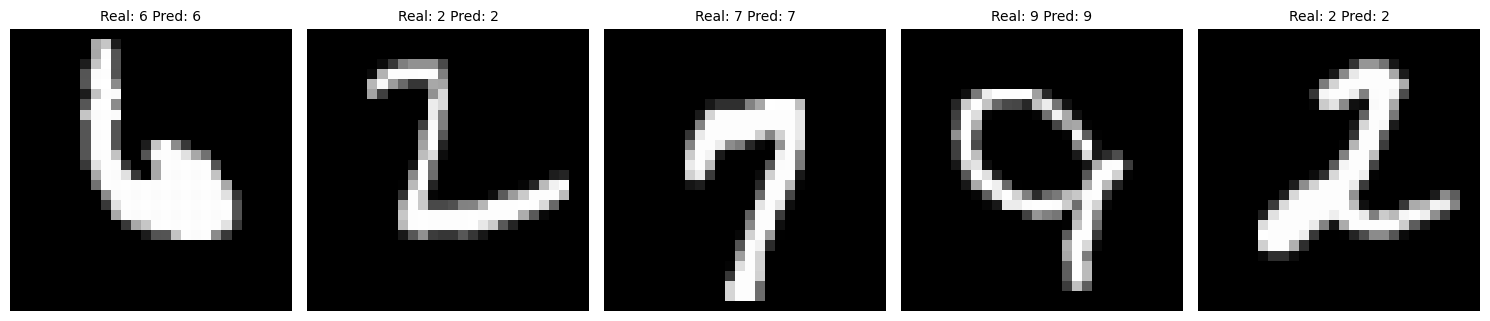
Для проверки работоспособности нейросети было взято несколько вариантов из валидационных данных датасета (Рисунок 6). 

Рисунок 6 – Тестирование нейросети

В ходе тестирования нейросети было выявлено, что нейросеть работоспособна и справляется со своей задачей. Она хорошо определяет цифры на изображениях.

# Заключение

В ходе практической работы были изучены основы работы с фреймворком TensorFlow/Keras. Была реализована и модифицирована модель для классификации рукописных цифр. Результаты показали, что использование методов инициализации, нормализации и добавление скрытых слоёв позволяет улучшить качество модели. Наиболее эффективным оказалось применение пакетной нормализации, которая предотвращает насыщение нейронов и улучшает процесс обучения.

# Приложение А

Листинг программы

Листинг 1 – Установка/Импортирование библиотек

!pip install tensorflow

import tensorflow as tf  
from tensorflow import keras  
from tensorflow.keras.utils import to\_categorical  
import numpy as np  
import logging  
import os  
os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'  
#%%  
tf.get\_logger().setLevel(logging.ERROR)  
tf.random.set\_seed(7)

Листинг 2 – Работа с датасетом

#Скачивание датасета  
mnist = keras.datasets.mnist  
(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()  
#Нормализация датасета  
mean = np.mean(train\_images)  
sttdev = np.std(train\_images)  
print(mean)  
print(sttdev)  
train\_images = (train\_images-mean) / sttdev  
test\_images = (test\_images-mean) / sttdev  
print(test\_images.shape)  
train\_labels = to\_categorical(train\_labels)  
test\_labels = to\_categorical(test\_labels)

Листинг 3 – Сборка нейросети

initializer = keras.initializers.he\_normal()  
model = keras.Sequential([  
 keras.layers.Flatten(input\_shape=(28, 28)),  
 keras.layers.Dense(128, activation='sigmoid',  
 kernel\_initializer=initializer,  
 bias\_initializer='zeros'),  
 keras.layers.BatchNormalization(),  
 keras.layers.Dense(64, activation='sigmoid',  
 kernel\_initializer=initializer,  
 bias\_initializer='zeros'),  
 keras.layers.BatchNormalization(),  
 keras.layers.Dense(32, activation='sigmoid',  
 kernel\_initializer=initializer,  
 bias\_initializer='zeros'),  
 keras.layers.BatchNormalization(),  
 keras.layers.Dense(16, activation='sigmoid',  
 kernel\_initializer=initializer,  
 bias\_initializer='zeros'),  
 keras.layers.BatchNormalization(),  
 keras.layers.Dense(10, activation='sigmoid',  
 kernel\_initializer=initializer,  
 bias\_initializer='zeros')  
  
])  
  
model.summary() #вывод информации о структуре нейросети  
opt = keras.optimizers.SGD(learning\_rate=LEARNING\_RATE)  
model.compile(optimizer=opt, loss='categorical\_crossentropy', metrics=['accuracy'])  
#Обучение нейросети  
history = model.fit(train\_images, train\_labels,validation\_data = (test\_images,test\_labels), epochs=EPOCHS,verbose = 2, batch\_size=BATCH\_SIZE, shuffle=True)

#Тестирование нейросети

from matplotlib import pyplot as plt  
plt.figure(figsize=(15, 7))  
for i in range(5):  
 plt.subplot(1, 5, i+1)  
 idx = np.random.randint(0, test\_images.shape[0])  
 img = test\_images[idx].reshape(28, 28)  
 real\_label = np.argmax(test\_labels[idx])  
 pred\_label = np.argmax(model.predict(img.reshape(1, 28, 28)))  
 plt.imshow(img, cmap='gray')  
 plt.title(f'Real: {real\_label} Pred: {pred\_label}', fontsize=10)  
 plt.axis('off')  
plt.tight\_layout()  
plt.show()

Листинг 4 – Работа с графиками

from matplotlib import pyplot as plt  
plt.figure()  
plt.plot(list(range(EPOCHS)), history.history['val\_accuracy'], label='Validation Accuracy')  
plt.plot(list(range(EPOCHS)), history.history['val\_loss'], label='Validation loss')  
plt.plot(list(range(EPOCHS)), history.history['accuracy'], label='Train accuracy')  
plt.plot(list(range(EPOCHS)), history.history['loss'], label='Train loss')  
plt.title('График обучения')  
plt.ylabel('Loss/Accuracy')  
plt.xlabel('Epochs')  
plt.grid(True)  
plt.legend()  
plt.show()