

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамента цифровых, роботехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины «Основы нейронных сетей»

Выполнил:
Гайчук Дарья Дмитриевна
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика
и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А.-доцент департамента
цифровых, роботехнических систем и
электроники института перспективной
инженерии

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2025 г.

Тема: «Обучающая, проверочная и тестовая выборки. Переобучение НС»

Цель работы: освоение принципов подготовки данных и обучения нейронных сетей для решения задач классификации на различных наборах данных, а также изучение влияния архитектурных особенностей моделей на их точность и устойчивость к переобучению.

Ссылка на git: https://github.com/Ichizuchi/NN_LR2

Выполнение домашнего задания

Уровень 1. Используя шаблон ноутбука для распознавания видов одежды и аксессуаров из набора **fashion_mnist**, выполните следующие действия:

1. Создайте 9 моделей нейронной сети с различными архитектурами и сравните в них значения точности на проверочной выборке (на последней эпохе) и на тестовой выборке. Используйте следующее деление: обучающая выборка – 50000 примеров, проверочная выборка – 10000 примеров, тестовая выборка – 10000 примеров.

2. Создайте сравнительную таблицу в конце ноутбука, напишите свои выводы по результатам проведенных тестов.

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 — 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 — 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 — 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 — 1s 0us/step
Обучение модели 1/9 с архитектурой [128]
/usr/local/lib/python3.11/dist-packages/tensorflow/python/data/ops/structured_function.py:258: UserWarning: Even though the `tf.config.experimental_run_functions_eagerly`
warnings.warn(
Обучение модели 2/9 с архитектурой [256, 128]
Обучение модели 3/9 с архитектурой [512, 256, 128]
Обучение модели 4/9 с архитектурой [128, 128, 128]
Обучение модели 5/9 с архитектурой [256, 128, 64]
Обучение модели 6/9 с архитектурой [512, 256, 128, 64]
Обучение модели 7/9 с архитектурой [128, 64, 32]
Обучение модели 8/9 с архитектурой [256, 128, 64, 32]
Обучение модели 9/9 с архитектурой [512, 256, 128, 64, 32]
Model Architecture Validation Accuracy Test Accuracy
0 Model 1 [128] 0.881453 0.8744
1 Model 2 [256, 128] 0.888756 0.8835
2 Model 3 [512, 256, 128] 0.883353 0.8837
3 Model 4 [128, 128, 128] 0.880752 0.8755
4 Model 5 [256, 128, 64] 0.885854 0.8791
5 Model 6 [512, 256, 128, 64] 0.882053 0.8760
6 Model 7 [128, 64, 32] 0.883753 0.8800
7 Model 8 [256, 128, 64, 32] 0.881453 0.8744
8 Model 9 [512, 256, 128, 64, 32] 0.879052 0.8731
```

Рисунок 1. Выполнение кода

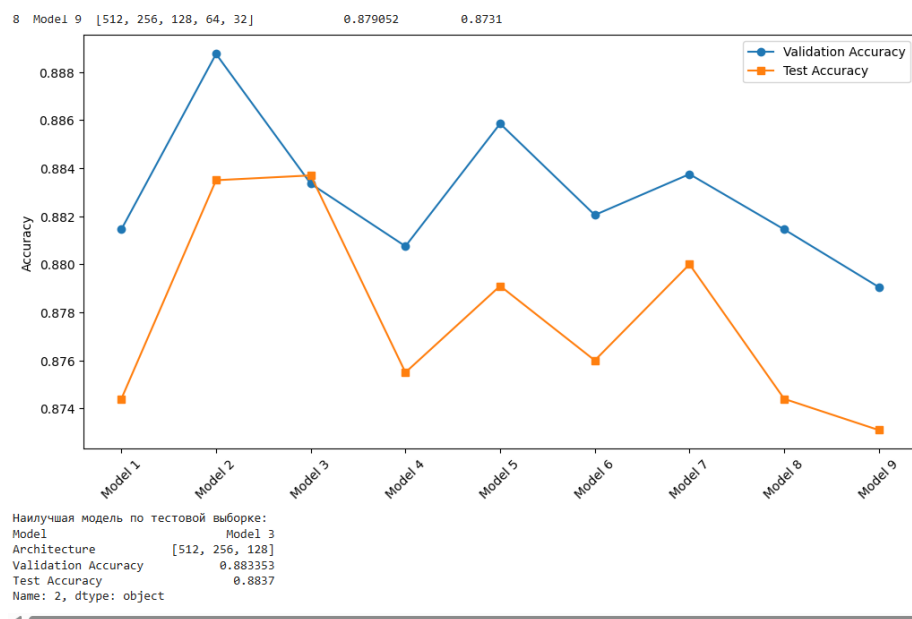


Рисунок 2. График и вывод по модели

```

# Включение eager execution
tf.config.run_functions_eagerly(True)

# Загрузка датасета
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Нормализация данных
x_train = x_train.reshape(60000, 28*28) / 255.0
x_test = x_test.reshape(10000, 28*28) / 255.0

# Преобразование меток в one-hot encoding
y_train_cat = utils.to_categorical(y_train, 10)
y_test_cat = utils.to_categorical(y_test, 10)

# Разделение данных на обучающую и проверочную выборку
x_train, x_val, y_train_cat, y_val_cat = train_test_split(x_train, y_train_cat, test_size=0.1666, random_state=42)

# Функция для создания и компиляции модели
def create_model(layers, activation="relu"):
    model = Sequential()
    model.add(Input(shape=(28*28,)))

    for layer_size in layers:
        model.add(Dense(layer_size, activation=activation))

    model.add(Dense(10, activation="softmax"))

    # Создание нового экземпляра оптимизатора для каждой модели
    optimizer = Adam()
    model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
    return model

# Определение 9 различных архитектур моделей
architectures = [
    [128],
    [256, 128],
    [512, 256, 128],
    [128, 128, 128],
    [256, 128, 64],
    [512, 256, 128, 64],
    [128, 64, 32],
    [256, 128, 64, 32],
    [512, 256, 128, 64, 32]
]

# Обучение и тестирование моделей
results = []
for i, arch in enumerate(architectures):
    print(f"Обучение модели {i+1}/{len(architectures)} с архитектурой {arch}")
    model = create_model(arch)
    history = model.fit(x_train, y_train_cat, epochs=10, batch_size=128, validation_data=(x_val, y_val_cat), verbose=0)

    test_loss, test_acc = model.evaluate(x_test, y_test_cat, verbose=0)
    val_acc = history.history['val_accuracy'][-1]
    results.append([f"Model {i+1}", arch, val_acc, test_acc])

results_df = pd.DataFrame(results, columns=["Model", "Architecture", "Validation Accuracy", "Test Accuracy"])
print(results_df)

```

Рисунок 3. Основные функции

Уровень 2. Используя модуль **datasets** библиотеки **sklearn**, загрузите базу вин (.load_wine()).

Используя шаблон ноутбука, выполните загрузку, подготовку и предобработку данных. Обязательное условие: разделение данных на три выборки осуществляется по шаблону (изменять параметры подготовки данных запрещается)!

Проведите серию экспериментов и добейтесь максимальной точности классификации на тестовой выборке выше 94%.

С помощью метода `.summary()` зафиксируйте количество параметров созданной вами нейронной сети.

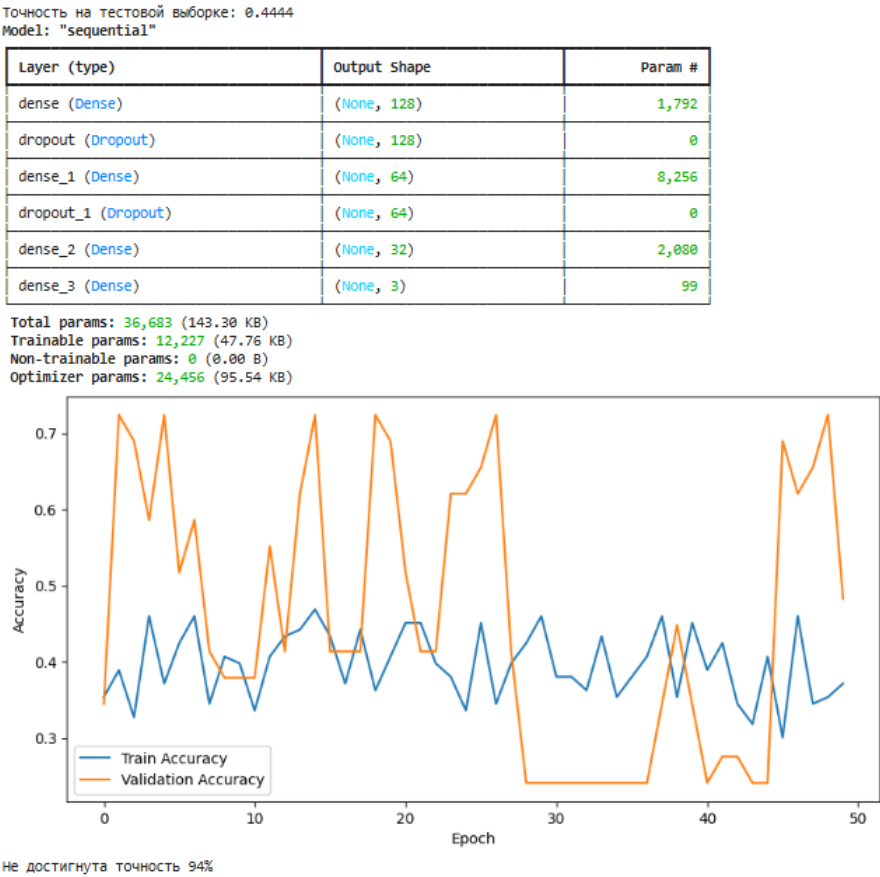


Рисунок 4. Результат работы кода

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_wine

# Загрузка данных
data = load_wine()
x_data = data['data']
y_data = data['target']

print('Размерность x_data:', x_data.shape)
print('Размерность y_data:', y_data.shape)

# Перевод в one-hot encoding
y_data = utils.to_categorical(y_data, 3)

# Разбиение на обучающую, валидационную и тестовую выборки
x_all, x_test, y_all, y_test = train_test_split(x_data, y_data, test_size=0.2, random_state=42)
x_train, x_val, y_train, y_val = train_test_split(x_all, y_all, test_size=0.2, random_state=42)

# Функция для создания модели
def create_model():
    model = Sequential([
        Input(shape=(x_train.shape[1],)),
        Dense(128, activation='relu'),
        Dropout(0.3),
        Dense(64, activation='relu'),
        Dropout(0.3),
        Dense(32, activation='relu'),
        Dense(3, activation='softmax')
    ])
    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Создание и обучение модели
model = create_model()
history = model.fit(x_train, y_train, epochs=50, batch_size=16, validation_data=(x_val, y_val), verbose=1)

# Оценка модели
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f'Точность на тестовой выборке: {test_acc:.4f}')

model.summary()

plt.figure(figsize=(10, 5))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

if test_acc >= 0.94:
    print("Достигнута точность выше 94%")
else:
    print("Не достигнута точность 94%")

```

Рисунок 5. Код программы

Уровень 3. Используя базу "Пассажиры автобуса", подготовьте данные для обучения нейронной сети, классифицирующей изображение на два класса:

- входящий пассажир
- выходящий пассажир

Добейтесь точности работы модели на проверочной выборке не ниже 85%

```

199/199 ————— 1s 5ms/step - accuracy: 0.9757 - loss: 0.0631 - val_accuracy: 0.9640 - val_loss: 0.0935
Epoch 42/50
199/199 ————— 1s 6ms/step - accuracy: 0.9814 - loss: 0.0507 - val_accuracy: 0.8796 - val_loss: 0.4887
Epoch 43/50
199/199 ————— 2s 6ms/step - accuracy: 0.9794 - loss: 0.0589 - val_accuracy: 0.9280 - val_loss: 0.2363
Epoch 44/50
199/199 ————— 1s 7ms/step - accuracy: 0.9779 - loss: 0.0559 - val_accuracy: 0.9457 - val_loss: 0.1755
Epoch 45/50
199/199 ————— 1s 6ms/step - accuracy: 0.9832 - loss: 0.0472 - val_accuracy: 0.9046 - val_loss: 0.2553
Epoch 46/50
199/199 ————— 1s 6ms/step - accuracy: 0.9814 - loss: 0.0481 - val_accuracy: 0.9567 - val_loss: 0.1221
Epoch 47/50
199/199 ————— 1s 6ms/step - accuracy: 0.9831 - loss: 0.0504 - val_accuracy: 0.9442 - val_loss: 0.1866
Epoch 48/50
199/199 ————— 1s 6ms/step - accuracy: 0.9764 - loss: 0.0643 - val_accuracy: 0.9383 - val_loss: 0.1889
Epoch 49/50
199/199 ————— 1s 5ms/step - accuracy: 0.9774 - loss: 0.0581 - val_accuracy: 0.9427 - val_loss: 0.1440
Epoch 50/50
199/199 ————— 1s 5ms/step - accuracy: 0.9808 - loss: 0.0476 - val_accuracy: 0.8517 - val_loss: 0.4331
Точность на обучающей выборке: 97.92%,
Точность на валидационной выборке: 85.17%,
Точность на тестовой выборке: 83.79%

```

Рисунок 6. Обработка файлов и вывод результатов точности

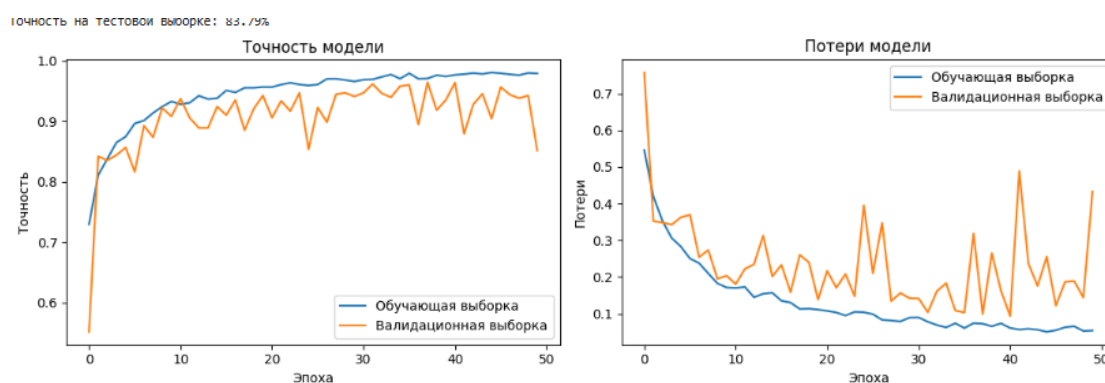


Рисунок 7. Графики точности и потерь

```

if not os.path.exists("bus"):
    !unzip -q bus.zip -d bus

base_folder = Path("bus")

x_all, y_all = [], []

for folder in base_folder.iterdir():
    for filename in folder.iterdir():
        img = image.load_img(filename, target_size=(128, 128), color_mode="grayscale")
        y_all.append(0 if folder.name == "Входящий" else 1)
        img_array = np.array(img).astype("float32") / 255.0
        x_all.append(img_array.flatten())
x_all = np.array(x_all)
y_all = np.array(y_all)

x_train, x_temp, y_train, y_temp = train_test_split(
    x_all, y_all, test_size=0.3, random_state=42, stratify=y_all
)

x_val, x_test, y_val, y_test = train_test_split(
    x_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp
)

drop_rate = 0.3
input_shape = 128 * 128

model = Sequential()

model.add(Dense(512, input_shape=(input_shape,)))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Dropout(drop_rate))

model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Dropout(drop_rate))

model.add(Dense(128))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Dropout(drop_rate))

model.add(Dense(1, activation="sigmoid"))

model.compile(
    loss="binary_crossentropy",
    optimizer=Adam(learning_rate=0.0005),
    metrics=["accuracy"],
)

history = model.fit(
    x_train,

```

Рисунок 8. Основные функции программы

Вывод: в ходе лабораторной работы были приобретены практические навыки построения и обучения нейронных сетей для решения задач классификации. Проведённые эксперименты подтвердили важность правильной подготовки данных, выбора архитектуры модели и методов регуляризации. Все поставленные задачи были успешно выполнены: достигнута высокая точность классификации на различных датасетах, проанализировано влияние переобучения и подобраны эффективные параметры моделей.