

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Центр цифровых
образовательных технологий

09.04.01 Информатика и вычислительная техника

Распознавание объектов на изображениях
Вариант 2

ЛАБОРАТОРНАЯ РАБОТА № 3

по дисциплине:
Машинное обучение

Исполнитель:

студент группы

8BM42

Текере Ричард

Руководитель:

доцент

ОИТ, ИШИТР

Друки А.А.

Томск - 2025

ВВЕДЕНИЕ

Искусственные нейронные сети (ИНС) являются основой современных подходов в задачах компьютерного зрения. Библиотека Keras предоставляет удобный и быстрый способ создания и обучения нейронных моделей.

Данная лабораторная работа посвящена распознаванию изображений элементов одежды с использованием ИНС на наборе Fashion MNIST.

Цель работы

Получение практических навыков построения и обучения нейронных сетей для задачи классификации изображений элементов одежды, а также оценка эффективности разработанной модели на реальных данных. Задачи работы:

1. Реализация топологии НС;
2. Подготовка данных для обучения НС;
3. Реализация топологии НС;
4. Обучение НС;
5. Тестирование НС;
6. Расчеты на основе обученной НС.

Ход работы

Методика выполнения каждой части лабораторной работы соответствует стандартному конвейеру машинного обучения. На первом этапе проводится исследование и предварительная обработка данных для изучения особенностей набора данных и подготовки признаков для построения модели.

Затем разрабатывается модель с использованием библиотеки Keras, где определяется архитектура нейронной сети, подходящая для задачи классификации..

1. Импорт библиотек и загрузка данных

На данном этапе производится импорт необходимых библиотек TensorFlow и matplotlib для работы с нейронными сетями и визуализацией результатов. После этого загружается датасет Fashion MNIST, который автоматически делится на обучающую и тестовую выборки.

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

fashion_mnist = tf.keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

Для визуального ознакомления с данными выводим первые 25 изображений:

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_train[i], cmap=plt.cm.binary)
    plt.xlabel(y_train[i])
plt.show()
```

Вывод:

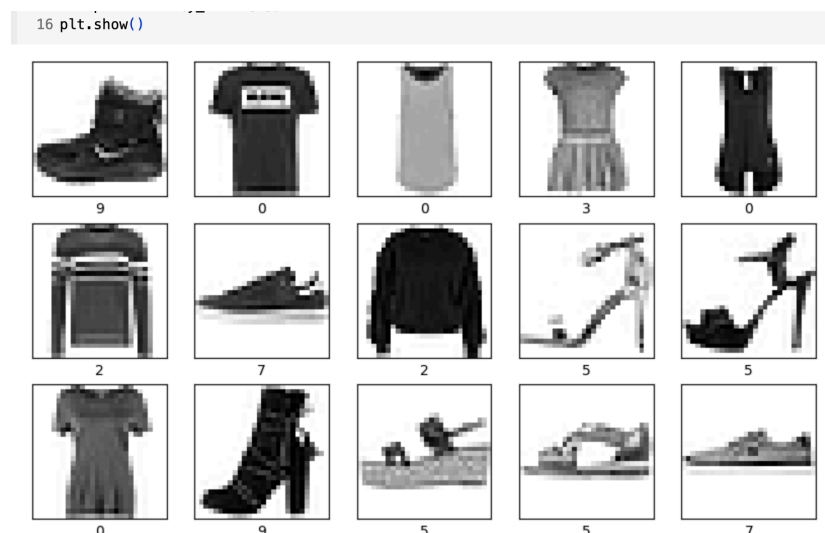


Рисунок 1 — Примеры изображений элементов одежды из обучающей выборки .

2 Предобработка данных

Нормализация изображений проводится для приведения значений пикселей в диапазон $[0, 1]$, что помогает ускорить и стабилизировать обучение нейронной сети.

```
X_train = X_train / 255.0  
X_test = X_test / 255.0
```

3. Построение модели нейронной сети

На данном этапе создается базовая полносвязная нейронная сеть, состоящая из одного скрытого слоя с 128 нейронами и выходного слоя из 10 нейронов (по числу классов одежды).

```
model = models.Sequential([  
    layers.Flatten(input_shape=(28, 28)),  
    layers.Dense(128, activation='relu'),  
    layers.Dense(10, activation='softmax')  
)  
  
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])  
  
model.summary()
```

Вывод:

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|-------------------|--------------|---------|
| flatten (Flatten) | (None, 784) | 0 |
| dense (Dense) | (None, 128) | 100,480 |
| dense_1 (Dense) | (None, 10) | 1,290 |

Total params: 101,770 (397.54 KB)
Trainable params: 101,770 (397.54 KB)
Non-trainable params: 0 (0.00 B)

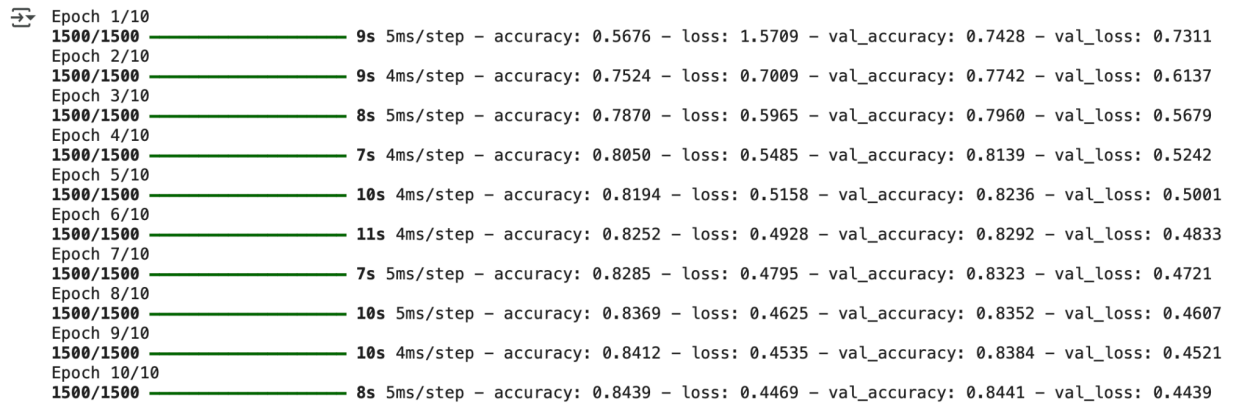
Рисунок 2 — Архитектура модели.

4 Обучение модели

Производится обучение модели на обучающей выборке с выделением 20% данных для валидации. Обучение проводится в течение 10 эпох.

```
history = model.fit(X_train, y_train, epochs=10, validation_split=0.2)
```

Вывод:



```
Epoch 1/10 1500/1500 ————— 9s 5ms/step - accuracy: 0.5676 - loss: 1.5709 - val_accuracy: 0.7428 - val_loss: 0.7311
Epoch 2/10 1500/1500 ————— 9s 4ms/step - accuracy: 0.7524 - loss: 0.7009 - val_accuracy: 0.7742 - val_loss: 0.6137
Epoch 3/10 1500/1500 ————— 8s 5ms/step - accuracy: 0.7870 - loss: 0.5965 - val_accuracy: 0.7960 - val_loss: 0.5679
Epoch 4/10 1500/1500 ————— 7s 4ms/step - accuracy: 0.8050 - loss: 0.5485 - val_accuracy: 0.8139 - val_loss: 0.5242
Epoch 5/10 1500/1500 ————— 10s 4ms/step - accuracy: 0.8194 - loss: 0.5158 - val_accuracy: 0.8236 - val_loss: 0.5001
Epoch 6/10 1500/1500 ————— 11s 4ms/step - accuracy: 0.8252 - loss: 0.4928 - val_accuracy: 0.8292 - val_loss: 0.4833
Epoch 7/10 1500/1500 ————— 7s 5ms/step - accuracy: 0.8285 - loss: 0.4795 - val_accuracy: 0.8323 - val_loss: 0.4721
Epoch 8/10 1500/1500 ————— 10s 5ms/step - accuracy: 0.8369 - loss: 0.4625 - val_accuracy: 0.8352 - val_loss: 0.4607
Epoch 9/10 1500/1500 ————— 10s 4ms/step - accuracy: 0.8412 - loss: 0.4535 - val_accuracy: 0.8384 - val_loss: 0.4521
Epoch 10/10 1500/1500 ————— 8s 5ms/step - accuracy: 0.8439 - loss: 0.4469 - val_accuracy: 0.8441 - val_loss: 0.4439
```

Рисунок 3 —Графики обучения.

Для построения графиков процесса обучения используем следующий код:

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Accuracy')

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')
plt.show()
```

Вывод:

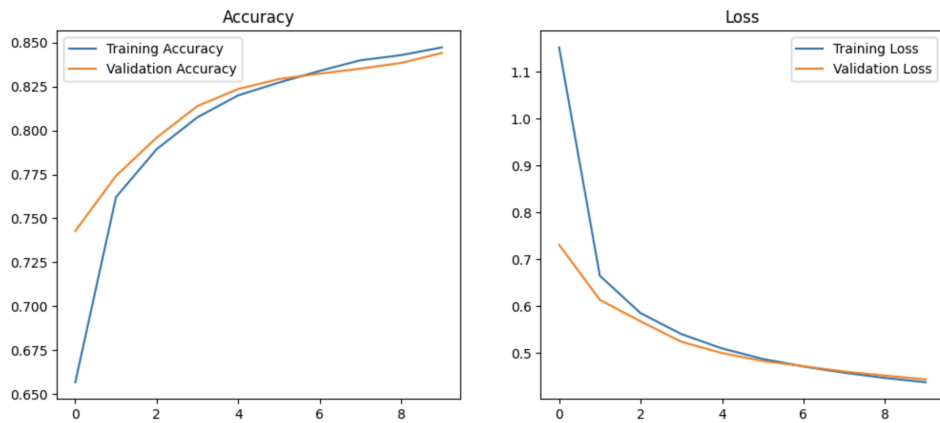


Рисунок 4 — Визуализация для обучения модели.

5 Оценка модели

Оцениваем итоговую производительность модели на ранее не использованных данных тестовой выборки.

```
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f'\nTest accuracy: {test_acc:.4f}')
```

```
313/313 - 1s - 3ms/step - accuracy: 0.8328 - loss: 0.4693
Test accuracy: 0.8328
```

Рисунок 5 — Точность модели на тестовой выборке..

6 Предсказания и визуализация результатов

Для качественной оценки работы модели визуализируем предсказания и сравниваем их с реальными метками классов.

```
predictions = model.predict(X_test)

plt.figure(figsize=(12,8))
for i in range(15):
    plt.subplot(3,5,i+1) # 3 строки, 5 столбцов
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_test[i], cmap=plt.cm.binary)
```

```

predicted_label = tf.argmax(predictions[i]).numpy()
true_label = y_test[i]
plt.xlabel(f"True: {true_label}\nPred: {predicted_label}")
plt.tight_layout()
plt.show()

```

Вывод:

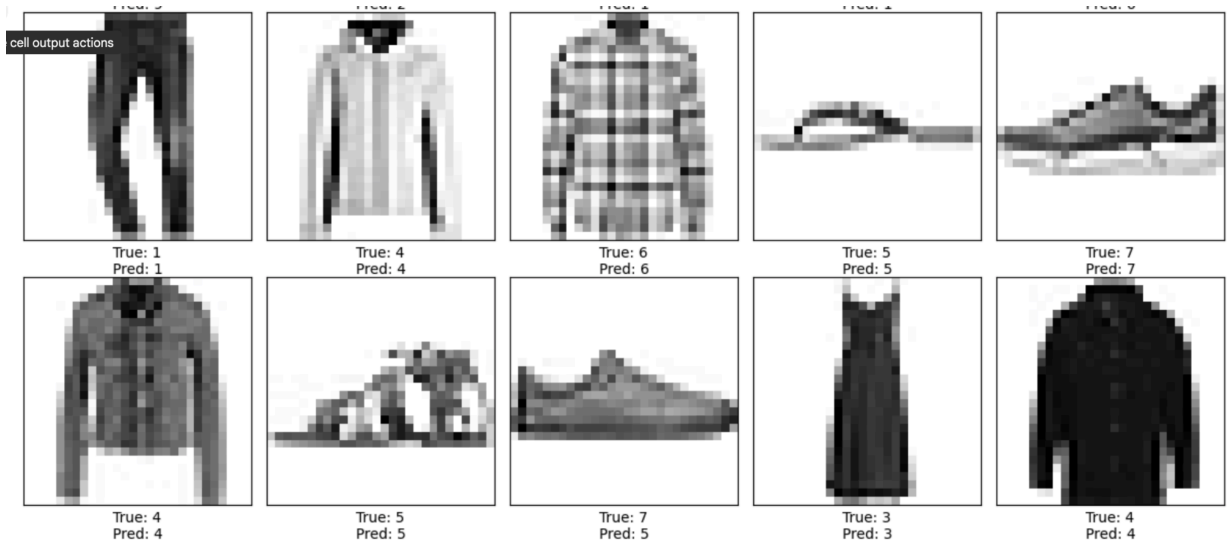


Рисунок 6 — Примеры правильных и ошибочных предсказаний модели.

Заключение

В ходе лабораторной работы были приобретены навыки построения, обучения и тестирования нейронных сетей для решения задачи классификации изображений. Использование библиотеки Keras позволило реализовать модель с высокой точностью и провести полноценный анализ результатов работы сети.

Приложение А

[Google Colab Link](#)

```
# 1. Импорт библиотек и загрузка данных
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

fashion_mnist = tf.keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

# 2. Предобработка данных
X_train = X_train / 255.0
X_test = X_test / 255.0

# 3. Построение модели
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

model.summary()

# 4. Обучение модели
history = model.fit(X_train, y_train, epochs=10, validation_split=0.2)

# Визуализация процесса обучения
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Training and Validation Loss')
plt.show()

# 5. Оценка модели на тестовой выборке
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f"\nTest accuracy: {test_acc:.4f}")
```



```
# 6. Визуализация предсказаний
predictions = model.predict(X_test)

plt.figure(figsize=(12,8))
for i in range(15):
    plt.subplot(3,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_test[i], cmap=plt.cm.binary)
    predicted_label = tf.argmax(predictions[i]).numpy()
    true_label = y_test[i]
    plt.xlabel(f'True: {true_label}\nPred: {predicted_label}')
plt.tight_layout()
plt.show()
```