Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра: «Вычислительная техника»

Дисциплина: «Системы искусственного интеллекта»

Лабораторная работа №6

«Сверточные сети»

Вариант 15

Выполнила

Студентка группы ИВТАСбд-31

Микка Е. И.

Ульяновск, 2024

# Общее задание на лабораторную работу

Перед выполнением лабораторной работы необходимо загрузить набор данных.

1. С использованием библиотеки Keras загрузить обучающую выборку, создать модель сверточной сети, обучить ее на обучающей выборке, сохранить модель в файл.
2. Написать дополнительную программу, которая загружает модель из файла, и предоставляет возможность загрузить файл или данные любым иным способом, чтобы проверить точность классификатора.

# Ход выполнения с описанием программной реализации

Задание 1.

Используется функция fashion\_mnist.load\_data() для загрузки обучающей и тестовой выборок, а затем эти данные разделяются на X\_train, Y\_train и X\_test, Y\_test. Задаются названия категорий классов (class\_names) для дальнейшего использования при представлении результатов. Входные данные нормализуются путем деления на 255, чтобы значения находились в диапазоне от 0 до 1, что помогает улучшить процесс обучения.

# Обучающая и тестовая выборки

(X\_train, Y\_train), (X\_test, Y\_test) = fashion\_mnist.load\_data()  
  
# Категории  
class\_names = ['Футболка или топ', 'Штаны', 'Свитер', 'Платье', 'Пальто', 'Сандалии', 'Рубашка', 'Кроссовки', 'Сумка', 'Ботинки']  
  
# Нормализация данных (данные находятся в диапозоне от 0 до 1)  
X\_train = X\_train / 255  
X\_test = X\_test / 255

Создается последовательная модель с использованием двух слоев Dense: входной слой с 128 нейронами и функцией активации RELU, и выходной слой с 10 нейронами (результаты для 10 категорий) и функцией активации softmax. Модель компилируется с оптимизатором SGD (стохастический градиентный спуск), функцией потерь sparse\_categorical\_crossentropy и метрикой accuracy (точность). Модель обучается на обучающих данных X\_train и метках Y\_train в течение 30 эпох. Модель оценивается на тестовых данных X\_test, Y\_test, вычисляются потери и точность предсказаний, что выводится на экран.

# Создание модели нейронной сети  
model = keras.Sequential([ # Последовательные слои  
 keras.layers.Flatten(input\_shape=(28, 28)), #Преобразоваие формата изображения  
 keras.layers.Dense(128, activation='relu'), #Входной слой (поступает 28\*28 от каждого изображения)  
 keras.layers.Dense(10, activation='softmax') #Выходной слой  
])  
  
# Компиляция модели  
model.compile(optimizer=keras.optimizers.SGD(), loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])  
  
# Вывод параметров модели  
#model.summary()  
  
# Обучение модели  
model.fit(X\_train, Y\_train, epochs=30) # 10 эпох  
  
# Проверка точности предсказаний на тестовой выборке  
test\_loss, test\_acc = model.evaluate(X\_test, Y\_test)  
print(f'Предсказания по тестовой выборке:\n'  
 f'Потери: {test\_loss}\n'  
 f'Точность: {test\_acc}\n')

Задание 2.

Создается еще один Python файл, который представляет из себя бесконечный цикл, в котором пользователю предлагается выбирать и обрабатывать изображения из двух источников: fashion\_mnist (набор данных изображений одежды) или собственные изображения. Обработка такого или иного изображения происходит в функции my\_model().

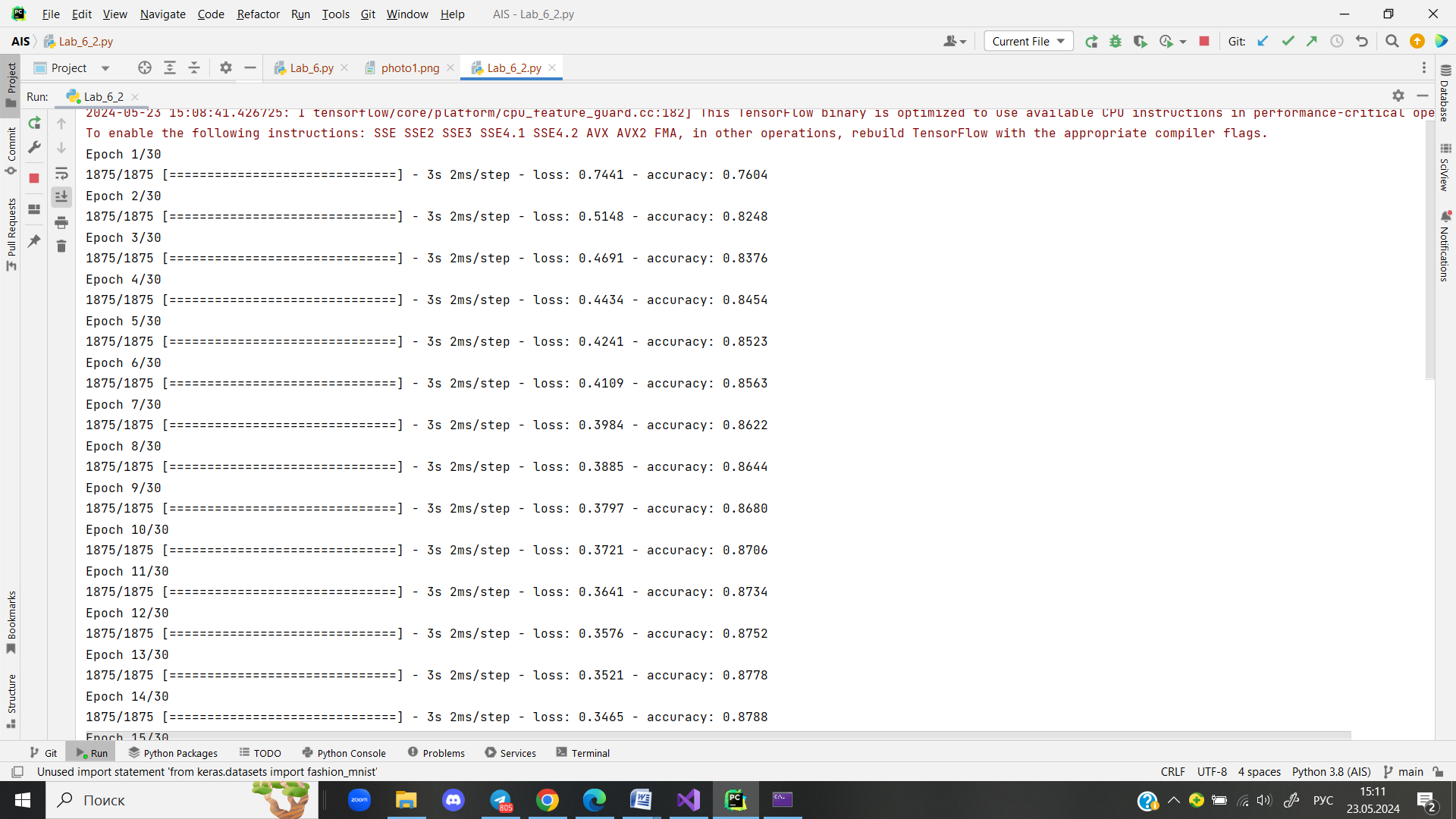
Если choice == 1, функция ожидает, что address представляет собой индекс изображения в обучающем наборе данных. Затем модель используется для предсказания класса для этого изображения. Результаты предсказания возвращаются вместе с самим изображением.

Если choice == 2, функция ожидает, что address представляет собой путь к изображению на диске. Изображение загружается, изменяется на размер 28x28 пикселей, преобразуется в оттенки серого и нормализуется. Если верхний левый пиксель изображения имеет яркость более 10 (на предполагаемом фоне), то все пиксели инвертируются. Затем модель используется для предсказания класса для этого изображения. Результаты предсказания возвращаются вместе с самим изображением.

def my\_model(choice, address):  
 # Заданное предсказание  
 if choice == 1:  
 address = int(address)  
 prediction = model.predict(X\_train)  
 index = np.argmax(prediction[address])  
 return X\_train[address], class\_names[index]  
 if choice == 2:  
 img = Image.open(address).resize((28, 28)).convert('L') # Преобразование в подходящий размер и серый формат  
 if img.getpixel((0, 0)) > 10:  
 for x in range(28):  
 for y in range(28):  
 brightness = 255 - img.getpixel((x, y))  
 img.putpixel((x, y), brightness)  
 img\_array = np.array(img) / 255 # Нормализация значений пикселей  
 img\_input = np.expand\_dims(img\_array, axis=0)  
 prediction = model.predict(img\_input)  
 index = np.argmax(prediction)  
 return img, class\_names[index]

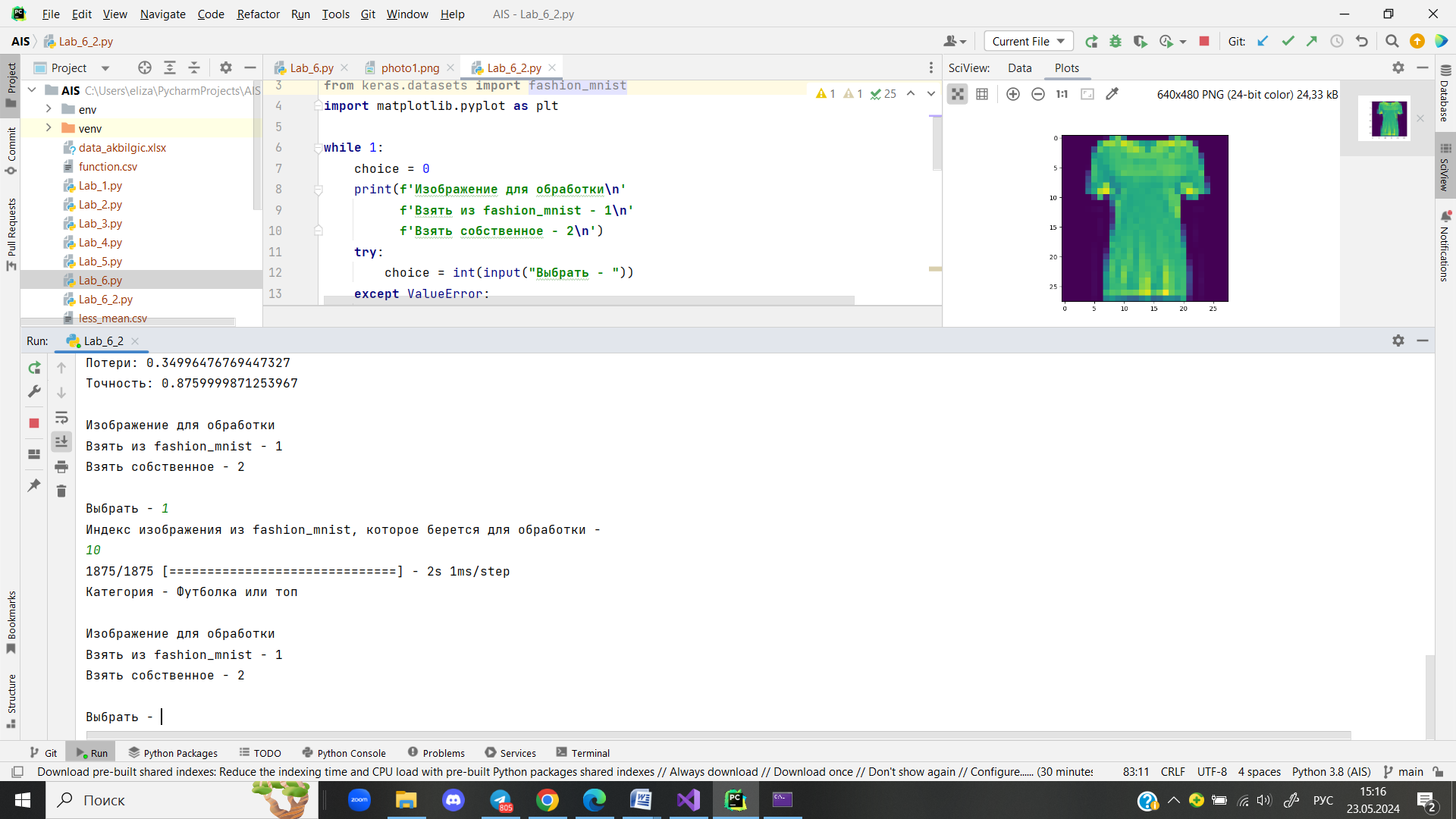
**Тестирование**

Обучение модели



Тест 1. Изображение из стандартного dataset

Вывод из консоли

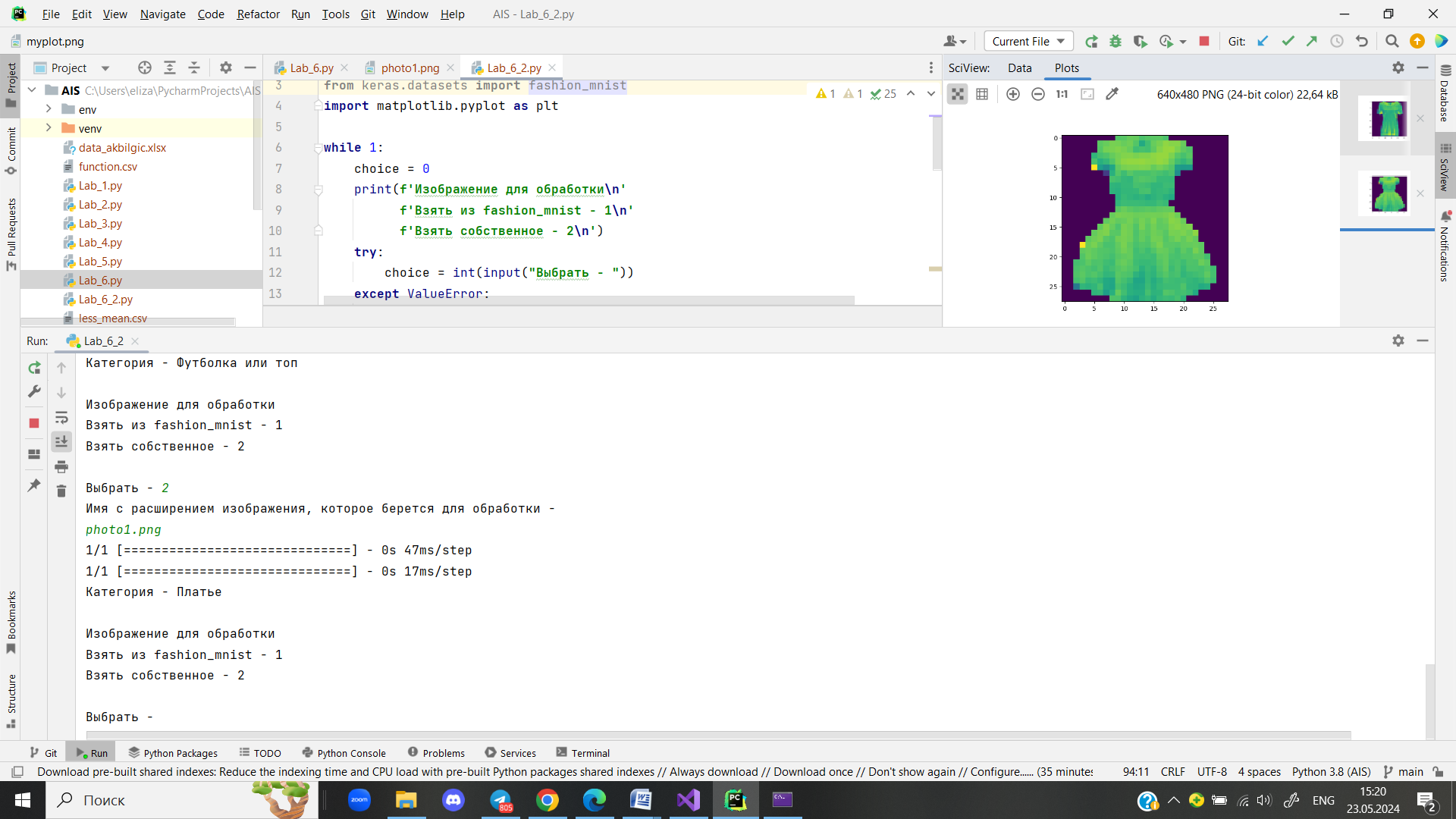
****

Изображение из dataset



Тест 2. Изображение с диска

Вывод из консоли



Изображение с диска

