объясни код import tensorflow as tf

train\_ds, test\_ds = tf.keras.utils.image\_dataset\_from\_directory(

"D:\Projects\ML\data\data3", # путь к изображениям

validation\_split=0.2, # процент на тест

subset="both", # берем и обучающую и тестовую выборку

seed=42, # сид генератора случайных чисел

image\_size=(200,400), # целевой размер изображения (СНАЧАЛА ВЫСОТА, ПОТОМ ШИРИНА)

batch\_size=32, # размер пакета (батча)

)

Этот код использует библиотеку TensorFlow для загрузки изображений из директории и создания обучающей и тестовой выборок. Функция tf.keras.utils.image\_dataset\_from\_directory используется для создания набора данных изображений из директории, где каждый подкаталог содержит изображения одного класса.

В этом примере, путь к изображениям указан как "D:\Projects\ML\data\data3". Параметр validation\_split=0.2 указывает, что 20% данных будут использоваться для тестирования, а остальные 80% - для обучения. Параметр subset="both" указывает, что будут возвращены оба набора данных - обучающий и тестовый. Параметр seed=42 устанавливает сид генератора случайных чисел для повторяемости результатов.

Параметр image\_size=(200,400) указывает целевой размер изображения (высота 200 пикселей и ширина 400 пикселей). Параметр batch\_size=32 указывает размер пакета (батча) для обучения модели.

Функция возвращает два объекта: train\_ds и test\_ds, которые являются обучающим и тестовым наборами данных соответственно.

2из30

объясни код import matplotlib.pyplot as plt

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

# берем один батч

for images, labels in train\_ds.take(1):

# отрисуем 9 картинок из батча

for i in range(9):

ax = plt.subplot(3, 3, i + 1)

# выводим картинку

plt.imshow(images[i].numpy().astype("uint8"))

plt.title(int(labels[i]))

# выключаем оси (чтобы они не отображались)

plt.axis("off")

Этот код использует библиотеку Matplotlib для визуализации изображений из обучающего набора данных train\_ds. Сначала создается новый рисунок с размером 10 на 10 дюймов с помощью функции plt.figure(figsize=(10, 10)).

Затем, с помощью цикла for images, labels in train\_ds.take(1):, берется один батч из обучающего набора данных. Вложенный цикл for i in range(9): используется для отрисовки 9 изображений из этого батча.

Для каждого изображения создается новый подграфик с помощью функции plt.subplot(3, 3, i + 1). Затем, изображение выводится на экран с помощью функции plt.imshow(images[i].numpy().astype("uint8")). Над изображением выводится метка класса с помощью функции plt.title(int(labels[i])). Оси выключаются с помощью функции plt.axis("off"), чтобы они не отображались на графике.

3из30

объясни клод data\_augmentation = tf.keras.Sequential(

[

tf.keras.layers.RandomFlip("horizontal"),

tf.keras.layers.RandomRotation(0.1),

tf.keras.layers.RandomTranslation(height\_factor=0.2, width\_factor=0.2, fill\_mode="constant")

]

)

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

for images, \_ in train\_ds.take(1):

for i in range(9):

augmented\_images = data\_augmentation(images)

ax = plt.subplot(3, 3, i + 1)

plt.imshow(augmented\_images[i].numpy().astype("uint8"))

plt.axis("off")

Этот код создает последовательную модель data\_augmentation с помощью библиотеки TensorFlow, которая используется для увеличения данных (data augmentation) путем применения случайных преобразований к изображениям. Модель состоит из трех слоев: tf.keras.layers.RandomFlip("horizontal"), который случайным образом отражает изображения по горизонтали; tf.keras.layers.RandomRotation(0.1), который случайным образом поворачивает изображения на угол до 0.1 радиана; и tf.keras.layers.RandomTranslation(height\_factor=0.2, width\_factor=0.2, fill\_mode="constant"), который случайным образом перемещает изображения по вертикали и горизонтали на расстояние, равное 20% от высоты и ширины соответственно.

Далее, создается новый рисунок с размером 10 на 10 дюймов с помощью функции plt.figure(figsize=(10, 10)). С помощью цикла for images, \_ in train\_ds.take(1): берется один батч из обучающего набора данных. Вложенный цикл for i in range(9): используется для отрисовки 9 увеличенных изображений из этого батча.

Для каждого изображения применяется модель увеличения данных с помощью функции augmented\_images = data\_augmentation(images). Затем, увеличенное изображение выводится на экран с помощью функции plt.imshow(augmented\_images[i].numpy().astype("uint8")). Оси выключаются с помощью функции plt.axis("off"), чтобы они не отображались на графике.

4из30

объясни код train\_ds = train\_ds.map(

lambda img, label: (data\_augmentation(img), label)

)

Этот код применяет модель увеличения данных data\_augmentation к обучающему набору данных train\_ds. Метод map используется для применения функции к каждому элементу набора данных. В этом случае, функция определяется как лямбда-выражение lambda img, label: (data\_augmentation(img), label), которое принимает на вход изображение и метку класса и возвращает кортеж, состоящий из увеличенного изображения и метки класса.

Таким образом, после выполнения этого кода, все изображения в обучающем наборе данных будут увеличены с помощью модели data\_augmentation.

5из30

объясни код # входной слой - размер (200, 400, 3)

inputs = tf.keras.Input(shape=(200, 400, 3))

# преобразуем значения пикселей из [0, 255] к [0, 1]

x = tf.keras.layers.Rescaling(1.0 / 255)(inputs)

# первый сверточный слой

x = tf.keras.layers.Conv2D(64, 3, strides=2, padding="same")(x)

# первый пулинг

x = tf.keras.layers.MaxPooling2D(3, strides=2, padding="same")(x)

# второй сверточный слой

x = tf.keras.layers.Conv2D(128, 4, strides=2, padding="same")(x)

# второй пулинг

x = tf.keras.layers.MaxPooling2D(3, strides=2, padding="same")(x)

# выпрямляем многомерный массив

x = tf.keras.layers.Flatten()(x)

# полносвязный слой с 64 нейронами

x = tf.keras.layers.Dense(64, activation="relu")(x)

# выходной слой с функцией sofrmax

outputs = tf.keras.layers.Dense(2, activation="softmax")(x)

model = tf.keras.Model(inputs, outputs)

Этот код создает сверточную нейронную сеть с помощью библиотеки TensorFlow. Сначала создается входной слой с размером (200, 400, 3) с помощью функции tf.keras.Input(shape=(200, 400, 3)). Затем, применяется слой tf.keras.layers.Rescaling(1.0 / 255), который преобразует значения пикселей из диапазона [0, 255] в диапазон [0, 1].

Далее, добавляется первый сверточный слой с 64 фильтрами размером 3 на 3 пикселя и шагом 2 с помощью функции tf.keras.layers.Conv2D(64, 3, strides=2, padding="same"). Затем, применяется первый слой пулинга с размером окна 3 на 3 пикселя и шагом 2 с помощью функции tf.keras.layers.MaxPooling2D(3, strides=2, padding="same").

Добавляется второй сверточный слой с 128 фильтрами размером 4 на 4 пикселя и шагом 2 с помощью функции tf.keras.layers.Conv2D(128, 4, strides=2, padding="same"). Затем, применяется второй слой пулинга с размером окна 3 на 3 пикселя и шагом 2 с помощью функции tf.keras.layers.MaxPooling2D(3, strides=2, padding="same").

Следующий слой - tf.keras.layers.Flatten() - выпрямляет многомерный массив в одномерный. Затем, добавляется полносвязный слой с 64 нейронами и функцией активации ReLU с помощью функции tf.keras.layers.Dense(64, activation="relu").

Выходной слой состоит из двух нейронов и использует функцию активации softmax с помощью функции tf.keras.layers.Dense(2, activation="softmax"). В конце создается модель model с помощью функции tf.keras.Model(inputs, outputs), которая принимает на вход входные данные и выходные данные.

6из30

объясни код model.compile(optimizer=tf.keras.optimizers.Adam(1e-3), loss="sparse\_categorical\_crossentropy", metrics=["accuracy"])

model.fit(train\_ds, epochs=5, validation\_data=test\_ds)

Этот код компилирует и обучает модель нейронной сети, созданную ранее. Метод compile используется для настройки процесса обучения модели. В этом случае, оптимизатор устанавливается как tf.keras.optimizers.Adam(1e-3), функция потерь - как "sparse\_categorical\_crossentropy", а метрика - как "accuracy".

Метод fit используется для обучения модели на обучающем наборе данных train\_ds в течение 5 эпох. Параметр validation\_data=test\_ds указывает, что тестовый набор данных test\_ds будет использоваться для оценки качества модели после каждой эпохи обучения.

7из30

объясни код img = tf.keras.preprocessing.image.load\_img(

r"D:\Projects\ML\data\data3\dogs\2.jpg", target\_size=(200, 400)

)

img\_array = tf.keras.preprocessing.image.img\_to\_array(img)

img\_array = tf.expand\_dims(img\_array, 0)

predictions = model.predict(img\_array)

predictions

Этот код загружает изображение из файла, преобразует его в массив и использует обученную модель для предсказания класса этого изображения. Сначала, изображение загружается из файла r"D:\Projects\ML\data\data3\dogs\2.jpg" с помощью функции tf.keras.preprocessing.image.load\_img и изменяется до размера (200, 400) пикселей.

Затем, изображение преобразуется в массив с помощью функции tf.keras.preprocessing.image.img\_to\_array(img). Массив расширяется с помощью функции tf.expand\_dims(img\_array, 0), чтобы добавить размер пакета (батча).

Далее, массив передается в модель с помощью метода predict, который возвращает предсказания для каждого класса. В этом случае, предсказания сохраняются в переменной predictions и выводятся на экран.

8из30

объясни код train\_ds, test\_ds = tf.keras.utils.image\_dataset\_from\_directory(

r"D:\Projects\ML\data\caltech-101\101\_ObjectCategories\MY", # путь к изображениям

validation\_split=0.15, # процент на тест

subset="both", # берем и обучающую и тестовую выборку

seed=42, # сид генератора случайных чисел

image\_size=(224,224), # целевой размер изображения (СНАЧАЛА ВЫСОТА, ПОТОМ ШИРИНА)

batch\_size=32, # размер пакета (батча)

)

Этот код использует библиотеку TensorFlow для загрузки изображений из директории и создания обучающей и тестовой выборок. Функция tf.keras.utils.image\_dataset\_from\_directory используется для создания набора данных изображений из директории, где каждый подкаталог содержит изображения одного класса.

В этом примере, путь к изображениям указан как r"D:\Projects\ML\data\caltech-101\101\_ObjectCategories\MY". Параметр validation\_split=0.15 указывает, что 15% данных будут использоваться для тестирования, а остальные 85% - для обучения. Параметр subset="both" указывает, что будут возвращены оба набора данных - обучающий и тестовый. Параметр seed=42 устанавливает сид генератора случайных чисел для повторяемости результатов.

Параметр image\_size=(224,224) указывает целевой размер изображения (высота 224 пикселя и ширина 224 пикселя). Параметр batch\_size=32 указывает размер пакета (батча) для обучения модели.

Функция возвращает два объекта: train\_ds и test\_ds, которые являются обучающим и тестовым наборами данных соответственно.

9из30

объясни код plt.figure(figsize=(10, 10))

# берем один батч

for images, labels in train\_ds.take(1):

# отрисуем 9 картинок из батча

for i in range(9):

ax = plt.subplot(3, 3, i + 1)

# выводим картинку

plt.imshow(images[i].numpy().astype("uint8"))

plt.title(int(labels[i]))

# выключаем оси (чтобы они не отображались)

plt.axis("off")

Этот код использует библиотеку Matplotlib для визуализации изображений из обучающего набора данных train\_ds. Сначала создается новый рисунок с размером 10 на 10 дюймов с помощью функции plt.figure(figsize=(10, 10)).

Затем, с помощью цикла for images, labels in train\_ds.take(1):, берется один батч из обучающего набора данных. Вложенный цикл for i in range(9): используется для отрисовки 9 изображений из этого батча.

Для каждого изображения создается новый подграфик с помощью функции plt.subplot(3, 3, i + 1). Затем, изображение выводится на экран с помощью функции plt.imshow(images[i].numpy().astype("uint8")). Над изображением выводится метка класса с помощью функции plt.title(int(labels[i])). Оси выключаются с помощью функции plt.axis("off"), чтобы они не отображались на графике.

10из30

объясни код vgg = tf.keras.applications.VGG19(weights='imagenet', include\_top=True)

vgg.summary()

Этот код загружает предварительно обученную модель VGG19 с помощью библиотеки TensorFlow. Функция tf.keras.applications.VGG19 используется для создания экземпляра модели VGG19. Параметр weights='imagenet' указывает, что модель должна быть инициализирована весами, обученными на наборе данных ImageNet. Параметр include\_top=True указывает, что в модель должен быть включен верхний (выходной) слой.

Метод summary используется для вывода информации о структуре модели, включая количество слоев, типы слоев и количество параметров в каждом слое.

11из30

объясни код inp = vgg.input

# создадим новый слой классификации по нашему количеству классов

new\_classification\_layer = tf.keras.layers.Dense(3, activation='softmax')

# соединим предпоследний слой VGG и наш новый слой классификации

out = new\_classification\_layer(vgg.layers[-2].output)

# создадим новую модель

model\_new = tf.keras.Model(inp, out)

Этот код создает новую модель, используя предварительно обученную модель VGG19. Сначала, входной слой модели VGG19 сохраняется в переменной inp с помощью свойства vgg.input.

Затем, создается новый полносвязный слой с 3 нейронами и функцией активации softmax с помощью функции tf.keras.layers.Dense(3, activation='softmax'). Этот слой будет использоваться для классификации изображений на 3 класса.

Далее, новый слой соединяется с предпоследним слоем модели VGG19 с помощью метода output и функции new\_classification\_layer(vgg.layers[-2].output). Таким образом, выход предпоследнего слоя модели VGG19 передается на вход нового слоя классификации.

В конце создается новая модель model\_new с помощью функции tf.keras.Model(inp, out), которая принимает на вход входные данные и выходные данные. В этом случае, входными данными являются данные из входного слоя модели VGG19, а выходными данными - данные из нового слоя классификации.

12из30

объясни код # "заморозим" все слои, кроме последнего

for layer in model\_new.layers:

layer.trainable = False

model\_new.layers[-1].trainable = True

Этот код “замораживает” все слои новой модели, кроме последнего. Это означает, что веса этих слоев не будут обновляться во время обучения модели. Цикл for layer in model\_new.layers: используется для перебора всех слоев модели. Для каждого слоя устанавливается свойство trainable в значение False, чтобы “заморозить” этот слой.

Затем, свойство trainable последнего слоя устанавливается в значение True, чтобы разморозить этот слой. Это означает, что веса этого слоя будут обновляться во время обучения модели.

13из30

объясни код model\_new.compile(loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model\_new.summary()

Этот код компилирует новую модель и выводит информацию о ее структуре. Метод compile используется для настройки процесса обучения модели. В этом случае, функция потерь устанавливается как 'sparse\_categorical\_crossentropy', оптимизатор - как 'adam', а метрика - как ['accuracy'].

Метод summary используется для вывода информации о структуре модели, включая количество слоев, типы слоев и количество параметров в каждом слое.

14из30

объясни код model\_new.fit(train\_ds, epochs=5, validation\_data=test\_ds)

Этот код обучает новую модель на обучающем наборе данных train\_ds. Метод fit используется для обучения модели. В этом случае, модель обучается в течение 5 эпох. Параметр validation\_data=test\_ds указывает, что тестовый набор данных test\_ds будет использоваться для оценки качества модели после каждой эпохи обучения.

15из30

объясни код img = tf.keras.preprocessing.image.load\_img(

r"D:\Projects\ML\data\caltech-101\101\_ObjectCategories\MY\rhino\image\_0010.jpg", target\_size=(224, 224)

)

img\_array = tf.keras.preprocessing.image.img\_to\_array(img)

img\_array = tf.expand\_dims(img\_array, 0)

predictions = model\_new.predict(img\_array)

predictions

Этот код загружает изображение из файла, преобразует его в массив и использует обученную модель для предсказания класса этого изображения. Сначала, изображение загружается из файла r"D:\Projects\ML\data\caltech-101\101\_ObjectCategories\MY\rhino\image\_0010.jpg" с помощью функции tf.keras.preprocessing.image.load\_img и изменяется до размера (224, 224) пикселей.

Затем, изображение преобразуется в массив с помощью функции tf.keras.preprocessing.image.img\_to\_array(img). Массив расширяется с помощью функции tf.expand\_dims(img\_array, 0), чтобы добавить размер пакета (батча).

Далее, массив передается в модель с помощью метода predict, который возвращает предсказания для каждого класса. В этом случае, предсказания сохраняются в переменной predictions и выводятся на экран.

16из30

объясни код # "заморозим" все слои, кроме второй сконца

for layer in model\_new.layers:

layer.trainable = False

model\_new.layers[-2].trainable = True

model\_new.compile(loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model\_new.summary()

Этот код “замораживает” все слои новой модели, кроме второго с конца. Это означает, что веса этих слоев не будут обновляться во время обучения модели. Цикл for layer in model\_new.layers: используется для перебора всех слоев модели. Для каждого слоя устанавливается свойство trainable в значение False, чтобы “заморозить” этот слой.

Затем, свойство trainable второго с конца слоя устанавливается в значение True, чтобы разморозить этот слой. Это означает, что веса этого слоя будут обновляться во время обучения модели.

Далее, модель компилируется с помощью метода compile, который настраивает процесс обучения модели. В этом случае, функция потерь устанавливается как 'sparse\_categorical\_crossentropy', оптимизатор - как 'adam', а метрика - как ['accuracy'].

Метод summary используется для вывода информации о структуре модели, включая количество слоев, типы слоев и количество параметров в каждом слое.

17из30

объясни код model\_new.fit(train\_ds, epochs=5, validation\_data=test\_ds)

Этот код обучает новую модель на обучающем наборе данных train\_ds. Метод fit используется для обучения модели. В этом случае, модель обучается в течение 5 эпох. Параметр validation\_data=test\_ds указывает, что тестовый набор данных test\_ds будет использоваться для оценки качества модели после каждой эпохи обучения.

18из30

объясни код img = tf.keras.preprocessing.image.load\_img(

r"D:\Projects\ML\data\caltech-101\101\_ObjectCategories\MY\rhino\image\_0010.jpg", target\_size=(224, 224)

)

img\_array = tf.keras.preprocessing.image.img\_to\_array(img)

img\_array = tf.expand\_dims(img\_array, 0)

predictions = model\_new.predict(img\_array)

predictions

Этот код загружает изображение из файла, преобразует его в массив и использует обученную модель для предсказания класса этого изображения. Сначала, изображение загружается из файла r"D:\Projects\ML\data\caltech-101\101\_ObjectCategories\MY\rhino\image\_0010.jpg" с помощью функции tf.keras.preprocessing.image.load\_img и изменяется до размера (224, 224) пикселей.

Затем, изображение преобразуется в массив с помощью функции tf.keras.preprocessing.image.img\_to\_array(img). Массив расширяется с помощью функции tf.expand\_dims(img\_array, 0), чтобы добавить размер пакета (батча).

Далее, массив передается в модель с помощью метода predict, который возвращает предсказания для каждого класса. В этом случае, предсказания сохраняются в переменной predictions и выводятся на экран.