Лабораторна робота 8 ІАД

1.Завантаження набору даних

Почнемо із завантаження та розпакування архіву з даними:

```
import zipfile
import os

# Путь к архиву
zip_path = "https://storage.googleapis.com/ztm_tf_course/food_vision/101_food_classes_10_percent.zip"
extract_path = "food101_data"

# Распаковка архива
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)
```

2.Вибір класів для класифікації

Визначимо три класи відповідно до формули:

```
import os
import numpy as np

# Определение классов
classes = ['class1', 'class2', 'class3']  # Замените на свои классы
data_dir = extract_path

# Загрузка данных
data = []
labels = []

for class_name in classes:
    class_dir = os.path.join(data_dir, class_name)
    for img_name in os.listdir(class_dir):
        img_path = os.path.join(class_dir, img_name)
        data.append(img_path)
        labels.append(class_name)

data = np.array(data)
labels = np.array(labels)
```

3.Аугментація даних

Ми завантажимо лише вибрані класи з набору даних:

```
[ ]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Создание генератора аугментации
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

4.Побудова моделі

Використаємо **MobileNetV2** як базову модель:

```
[ ]: from tensorflow.keras.applications import MobileNetV2
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

     # Загрузка базовой модели
     base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

     # Создание модели
     model = Sequential([
         base_model,
         GlobalAveragePooling2D(),
         Dense(len(classes), activation='softmax')
     ])

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

5. Навчання моделі

```
[ ]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

     # Генераторы для обучения и валидации
     train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
     train_generator = train_datagen.flow_from_directory(
         data_dir,
         target_size=(224, 224),
         batch_size=32,
         class_mode='sparse',
         subset='training'
     )

     validation_generator = train_datagen.flow_from_directory(
         data_dir,
         target_size=(224, 224),
         batch_size=32,
         class_mode='sparse',
         subset='validation'
     )

     # Обучение модели
     history = model.fit(train_generator, validation_data=validation_generator, epochs=10)
```

6. Оцінка результатів

```
[ ]: import matplotlib.pyplot as plt

     # Графики обучения
     plt.plot(history.history['accuracy'], label='accuracy')
     plt.plot(history.history['val_accuracy'], label='val_accuracy')
     plt.xlabel('Epoch')
     plt.ylabel('Accuracy')
     plt.legend()
     plt.show()
```

7.Прогнозування для нових зображень

Завантажте зображення та виконайте прогноз:

[*]:
```python
from tensorflow.keras.preprocessing import image

def predict_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img) / 255.0  # Нормализация
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    predicted_class = classes[np.argmax(prediction)]

    return predicted_class

# Пример использования
img_path = 'path/to/new/image.jpg'
print(f'Predicted class: {predict_image(img_path)}')
```