Лабораторна робота 10 ІАД

Transfer Learning з використанням TensorFlow

Мета : Навчитися застосовувати transfer learning для класифікації зображень, використовуючи попередньо натреновані моделі у TensorFlow.
Ознайомитися з основами fine-tuning моделі для покращення точності на
нових даних.

Крок 1: Імпорт бібліотек

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Крок 2: Завантаження та підготовка даних

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Путь к директории с данными
data_dir = 'path/to/your/dataset'

# Подготовка генераторов данных
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'
)
```

Крок 3: Візуалізація даних

```
[ ]: # Визуализация нескольких изображений
def visualize_data(generator):
    batch = next(generator)
    images, labels = batch[0], batch[1]
    plt.figure(figsize=(10, 10))
    for i in range(9):
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[i])
        plt.title(f'Label: {labels[i]}')
        plt.axis('off')
    plt.show()

visualize_data(train_generator)
```

Крок 4: Завантаження попередньо натренованої моделі

```
[ ]: # Загрузка модели MobileNetV2 без верхнего слоя
base_model = tf.keras.applications.MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))


# Замораживание слоев базовой модели
base_model.trainable = False
```

Крок 5: Тренування моделі

```
[ ]: # Создание модели
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dense(train_generator.num_classes, activation='softmax')  # Классы
])

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

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

Етап 6:Оцінка моделі

```
# Оценка модели
loss, accuracy = model.evaluate(validation_generator)
print(f'Точность модели: {accuracy * 100:.2f}%')
```

Етап 7: Матриця похибок

```
# Получение предсказаний
y_true = validation_generator.classes
y_pred = np.argmax(model.predict(validation_generator), axis=-1)

# Создание матрицы ошибок
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=validation_generator.class_indices.keys())
disp.plot(cmap=plt.cm.Blues)
plt.show()
```

Етап 8: Візуалізація результатів передбачення

```
def plot_predictions(generator):
    batch = next(generator)
    images, labels = batch[0], batch[1]
    predictions = np.argmax(model.predict(images), axis=-1)

    plt.figure(figsize=(10, 10))
    for i in range(9):
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[i])
        plt.title(f'True: {labels[i]}, Pred: {predictions[i]}')
        plt.axis('off')
    plt.show()

plot_predictions(validation_generator)
```

Етап9: Fine-tuning (Донастроювання моделі)

```
# Разморозка последних 20 слоев базовой модели
base_model.trainable = True
for layer in base_model.layers[:-20]:
    layer.trainable = False

# Компиляция модели с низким значением скорости обучения
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Донастройка модели
history_fine = model.fit(train_generator, validation_data=validation_generator, epochs=5)
```