Домашняя контрольная работа №2

Цель работы

Изучение методов синтетической генерации данных с использованием OpenCV и применение нейронных сетей для задачи классификации объектов. Освоение методов дообучения существующих моделей (transfer learning) и анализа качества обученной модели.

Задачи работы

1. Создать синтетический набор данных для обучения классификатора объектов
2. Разработать или дообучить нейронную сеть для классификации объектов
3. Обосновать выбор гиперпараметров обучения (оптимизатор, функция потерь, метрики)
4. Провести анализ результатов обучения с построением графиков и вычислением метрик качества

Теоретическая справка

Синтетические данные в компьютерном зрении

Синтетические данные представляют собой искусственно созданные изображения, которые имитируют реальные сценарии. Использование синтетических данных позволяет решить несколько критических проблем: недостаток размеченных данных, увеличение разнообразия обучающей выборки, получение точной разметки без ручного труда, контроль сложности сценариев.

Transfer Learning и дообучение моделей

Transfer learning (трансферное обучение) — это метод использования предобученной модели, которая была обучена на большом наборе данных (например, ImageNet), для решения новой задачи. Существует два основных подхода:

Fine-tuning (дообучение) — инициализация модели весами предобученной сети с последующим дообучением всех или части слоев на новом наборе данных.

Feature extraction (извлечение признаков) — заморозка весов предобученной модели и обучение только нового выходного слоя.

Выбор подхода зависит от размера нового набора данных и его схожести с исходными данными.

Оптимизаторы

SGD (Stochastic Gradient Descent) — классический градиентный спуск, может требовать больше времени для сходимости, но при правильной настройке дает хорошие результаты.

RMSprop — адаптивный метод оптимизации, использующий экспоненциально взвешенное среднее квадратов градиентов, обеспечивает быструю сходимость.

Adam (Adaptive Moment Estimation) — объединяет преимущества RMSprop и momentum, обеспечивает быструю и стабильную сходимость, является наиболее популярным выбором для большинства задач.

Функции потерь

Binary Cross-Entropy — используется для бинарной классификации (два класса).

Categorical Cross-Entropy — применяется для многоклассовой классификации, когда метки представлены в формате one-hot encoding.

Sparse Categorical Cross-Entropy — аналог categorical cross-entropy, но используется, когда метки представлены в виде целых чисел.

Метрики качества

Accuracy (точность) — доля правильных предсказаний от общего числа предсказаний.

Precision (точность положительных предсказаний) — доля истинно положительных предсказаний среди всех положительных предсказаний модели.

Recall (полнота) — доля истинно положительных предсказаний среди всех действительно положительных примеров.

F1-Score — гармоническое среднее между precision и recall, особенно полезно для несбалансированных наборов данных.

ROC-кривая и AUC — ROC-кривая показывает соотношение True Positive Rate (чувствительность) и False Positive Rate при различных порогах классификации. AUC (площадь под ROC-кривой) принимает значения от 0 до 1, где 1 означает идеальный классификатор, а 0.5 — случайное угадывание.

Задание на практическую работу

Часть 1. Генерация синтетического набора данных

Требования:

1. Выберите **3-5 классов объектов** для классификации (например: кошка, собака, птица, автомобиль, велосипед)
2. Подготовьте исходные изображения:
   - Найдите 10-15 изображений объектов каждого класса с прозрачным фоном или удалите фон
   - Подготовьте 20-30 различных фоновых изображений
3. Реализуйте функцию генерации синтетических изображений с использованием OpenCV. Пример приведен ниже:

```
import cv2
import numpy as np
import random
import os

def generate_synthetic_image(object_img, background_img):
    """
    Генерирует синтетическое изображение путем наложения объекта на фон

    Args:
        object_img: изображение объекта с альфа-каналом (RGBA)
        background_img: фоновое изображение (RGB)

    Returns:
        synthetic_img: синтетическое изображение
    """
    # Случайное масштабирование объекта (0.3-0.7 от размера фона)
    scale = random.uniform(0.3, 0.7)

    # Случайный поворот объекта (-30 до +30 градусов)
    angle = random.randint(-30, 30)

    # Случайная позиция на фоне
    # Реализовать наложение объекта на фон
    # Применить аугментации (изменение яркости, контраста, размытие)

    return synthetic_img
```

4. Примените следующие техники аугментации:
   - Масштабирование объекта (0.3-0.7 от размера изображения)
   - Поворот (-30° до +30°)
   - Горизонтальное отражение
   - Изменение яркости и контраста
   - Добавление небольшого размытия или шума
   - Случайное положение объекта на фоне
5. Сгенерируйте датасет:

- Обучающая выборка: 500-800 изображений на класс
- Валидационная выборка: 100-150 изображений на класс
- Тестовая выборка: 100-150 изображений на класс

6. Рекомендуемая (но не обязательная) структура папок:

```
dataset/
├── train/
│   ├── class_1/
│   ├── class_2/
│   └── class_3/
├── validation/
│   ├── class_1/
│   ├── class_2/
│   └── class_3/
└── test/
    ├── class_1/
    ├── class_2/
    └── class_3/
```

Что включить в отчет:

- Описание выбранных классов и источников данных
- Код функции генерации синтетических изображений
- Примеры сгенерированных изображений (5-10 изображений)
- Статистику датасета (количество изображений по классам)

Часть 2. Создание и дообучение нейронной сети

Требования:

Реализуйте **один из двух подходов**:

Вариант А: Создание собственной CNN архитектуры

Пример:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

def create_custom_cnn(input_shape, num_classes):
    """
    Создает кастомную CNN архитектуру
    """
    model = keras.Sequential([
        # Входной слой
        layers.Input(shape=input_shape),

        # Блок 1: Conv + MaxPool
        layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        # Блок 2: Conv + MaxPool
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
```

```
        # Блок 3: Conv + MaxPool
        # Добавить еще слои

        # Полносвязные слои
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])

    return model
```

Вариант Б: Transfer Learning с предобученной моделью (не допускается использование YOLO)

Пример:

```
from tensorflow.keras.applications import ResNet50, VGG16, MobileNetV2

def create_transfer_learning_model(input_shape, num_classes,
base_model_name='ResNet50'):
    """
    Создает модель на основе предобученной сети
    """
    # Загрузка предобученной модели без верхних слоев
    if base_model_name == 'ResNet50':
        base_model = ResNet50(weights='imagenet',
                              include_top=False,
                              input_shape=input_shape)
    elif base_model_name == 'VGG16':
        base_model = VGG16(weights='imagenet',
                           include_top=False,
                           input_shape=input_shape)

    # Заморозка слоев базовой модели
    base_model.trainable = False

    # Добавление новых слоев для классификации
    model = keras.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])

    return model
```

Что включить в отчет:

- Описание выбранной архитектуры модели
- Визуализацию архитектуры (model.summary() или схема)
- Количество обучаемых параметров
- Обоснование выбора архитектуры

Часть 3. Обоснование выбора гиперпараметров (2-3 балла)

Требования:

Для обучения модели необходимо выбрать и **обосновать** следующие компоненты:

1. **Оптимизатор**
   - Выберите один из: SGD, RMSprop, Adam, Adamax, Nadam
   - Обоснуйте выбор на основе характеристик задачи и данных
   - Укажите learning rate и другие параметры
2. **Функция потерь**
   - Categorical Cross-Entropy или Sparse Categorical Cross-Entropy
   - Обоснуйте выбор на основе формата меток классов
3. **Метрики качества**
   - Обязательные: accuracy, precision, recall, F1-score
   - Дополнительно: AUC-ROC для каждого класса
   - Обоснуйте важность каждой метрики для вашей задачи
4. **Параметры обучения**
   - Batch size
   - Количество эпох
   - Стратегия обучения (callbacks: EarlyStopping, ReduceLROnPlateau, ModelCheckpoint)

Пример кода:

```
# Компиляция модели
optimizer = keras.optimizers.Adam(learning_rate=0.001)
loss = 'categorical_crossentropy'  # или 'sparse_categorical_crossentropy'
metrics = ['accuracy',
          keras.metrics.Precision(name='precision'),
          keras.metrics.Recall(name='recall'),
          keras.metrics.AUC(name='auc')]

model.compile(optimizer=optimizer,
             loss=loss,
             metrics=metrics)

# Callbacks
callbacks = [
    keras.callbacks.EarlyStopping(monitor='val_loss',
                                  patience=10,
                                  restore_best_weights=True),
    keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                      factor=0.5,
                                      patience=5),
    keras.callbacks.ModelCheckpoint('best_model.h5',
                                    save_best_only=True,
                                    monitor='val_accuracy')
]

# Обучение
history = model.fit(
    train_dataset,
    validation_data=val_dataset,
    epochs=50,
```

```
    batch_size=32,
    callbacks=callbacks
)
```

Что включить в отчет:

Таблицу с выбранными гиперпараметрами и их обоснованием:

Параметр	Значение	Обоснование выбора
Оптимизатор	Adam	Обеспечивает быструю сходимость, адаптивный learning rate для каждого параметра
Learning rate	0.001	Стандартное значение для Adam, обеспечивает стабильное обучение
Функция потерь	Categorical Cross-Entropy	Метки представлены в one-hot формате, подходит для многоклассовой классификации
Batch size	32	Баланс между скоростью обучения и стабильностью градиентов
Метрики	Accuracy, Precision, Recall, F1, AUC	Комплексная оценка качества модели

Часть 4. Графики обучения и метрики

Требования:

1. **Графики обучения**

Постройте следующие графики:

```
import matplotlib.pyplot as plt

def plot_training_history(history):
    """
    Строит графики обучения модели
    """
    fig, axes = plt.subplots(2, 2, figsize=(15, 10))

    # График 1: Loss
    axes[0, 0].plot(history.history['loss'], label='Train Loss')
    axes[0, 0].plot(history.history['val_loss'], label='Val Loss')
    axes[0, 0].set_title('Model Loss')
    axes[0, 0].set_xlabel('Epoch')
    axes[0, 0].set_ylabel('Loss')
    axes[0, 0].legend()
    axes[0, 0].grid(True)

    # График 2: Accuracy
```

```
    axes[0, 1].plot(history.history['accuracy'], label='Train Accuracy')
    axes[0, 1].plot(history.history['val_accuracy'], label='Val Accuracy')
    axes[0, 1].set_title('Model Accuracy')
    axes[0, 1].set_xlabel('Epoch')
    axes[0, 1].set_ylabel('Accuracy')
    axes[0, 1].legend()
    axes[0, 1].grid(True)

    # График 3: Precision
    axes[1, 0].plot(history.history['precision'], label='Train Precision')
    axes[1, 0].plot(history.history['val_precision'], label='Val Precision')
    axes[1, 0].set_title('Model Precision')
    axes[1, 0].set_xlabel('Epoch')
    axes[1, 0].set_ylabel('Precision')
    axes[1, 0].legend()
    axes[1, 0].grid(True)

    # График 4: Recall
    axes[1, 1].plot(history.history['recall'], label='Train Recall')
    axes[1, 1].plot(history.history['val_recall'], label='Val Recall')
    axes[1, 1].set_title('Model Recall')
    axes[1, 1].set_xlabel('Epoch')
    axes[1, 1].set_ylabel('Recall')
    axes[1, 1].legend()
    axes[1, 1].grid(True)

    plt.tight_layout()
    plt.show()
```

Проанализируйте графики на предмет переобучения и недообучения:

- Если обучающий loss снижается, а валидационный растет — переобучение
- Если оба loss высокие и не улучшаются — недообучение
- Оптимальная модель: оба loss снижаются, небольшой разрыв между ними

2. **Матрица ошибок (Confusion Matrix)**

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np

# Получение предсказаний на тестовой выборке
y_pred = model.predict(test_dataset)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = test_labels  # истинные метки

# Построение матрицы ошибок
cm = confusion_matrix(y_true, y_pred_classes)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=class_names)
disp.plot(cmap='Blues', values_format='d')
plt.title('Confusion Matrix')
plt.show()
```

3. **ROC-кривые для каждого класса**

```
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize

# Бинаризация меток для ROC-кривой
y_test_bin = label_binarize(y_true, classes=range(num_classes))
```

```
# Вычисление ROC-кривой для каждого класса
plt.figure(figsize=(10, 8))
for i in range(num_classes):
    fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_pred[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for Multi-Class Classification')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

4. **Таблица метрик по классам**

```
from sklearn.metrics import classification_report

# Вычисление метрик для каждого класса
report = classification_report(y_true, y_pred_classes,
                               target_names=class_names,
                               output_dict=True)

# Создание таблицы
import pandas as pd
df_report = pd.DataFrame(report).transpose()
print(df_report)
```

Что включить в отчет:

- Все графики обучения с подробным анализом
- Матрицу ошибок с интерпретацией
- ROC-кривые для всех классов с значениями AUC
- Таблицу с метриками (precision, recall, F1-score) для каждого класса
- Анализ: какие классы классифицируются лучше/хуже и почему

Часть 5. Выводы

Требования:

Напишите развернутые выводы по работе, включающие:

1. **Оценка качества синтетических данных**
    - Насколько синтетические данные подходят для обучения модели
    - Какие проблемы возникли при генерации данных
    - Предложения по улучшению генерации
2. **Анализ результатов обучения**
    - Итоговые метрики качества модели на тестовой выборке
    - Сравнение результатов на обучающей, валидационной и тестовой выборках

- Наличие/отсутствие переобучения
- Какие классы классифицируются лучше и почему

3. **Эффективность выбранного подхода**
   - Насколько удачным был выбор архитектуры модели
   - Эффективность выбранных гиперпараметров
   - Что можно было сделать по-другому
4. **Практическая применимость**
   - Можно ли использовать полученную модель на реальных данных
   - Какие ограничения имеет модель
   - Направления для дальнейшего улучшения

Требования к оформлению отчета

1. **Титульный лист** с указанием названия работы, ФИО студента, группы
2. **Структура отчета:**
   - Введение (цель и задачи работы)
   - Теоретическая часть (краткое описание методов)
   - Практическая часть (выполнение всех 5 частей задания)
   - Выводы
   - Список использованных источников (если имеются)
3. **Технические требования:**
   - Весь код должен быть оформлен в виде Jupyter Notebook или Python скриптов
   - Код должен быть прокомментирован
   - Все графики должны иметь подписи осей и легенду
   - Отчет должен содержать скриншоты результатов работы
4. **Формат сдачи:**
   - PDF-файл отчета
   - Архив с кодом и примерами сгенерированных данных
   - Обученная модель (в edu не влезет, поэтому продемонстрировать работоспособность на паре)