Практическая работа №1

**Классификация медицинских изображений**

**1. Постановка задачи**

**Цель**: Научить модель отличать рентгеновские снимки пациентов с пневмонией от нормальных.

**Датасет**: [Chest X-Ray Images (Pneumonia)](https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia) (Kaggle).

* **Классы**: 2 класса — NORMAL (норма) и PNEUMONIA (пневмония).
* **Данные**:
  + Тренировочная выборка: 5216 изображений (1341 норма, 3875 пневмония).
  + Тестовая выборка: 624 изображения.

**2. Подготовка данных**

**Шаги**:

1. Загрузка данных и проверка структуры папок.
2. Разделение данных на тренировочную, валидационную и тестовую выборки.
3. Нормализация изображений (приведение к диапазону [0, 1]).
4. Аугментация данных для борьбы с переобучением:
   * Поворот на 20°,
   * Горизонтальное отражение,
   * Сдвиг по высоте/ширине на 10%,
   * Заливка (fill\_mode="nearest").

**Пример кода (Python, TensorFlow/Keras):**

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Пути к данным

train\_dir = 'chest\_xray/train'

test\_dir = 'chest\_xray/test'

# Аугментация для тренировочных данных

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=20,

width\_shift\_range=0.1,

height\_shift\_range=0.1,

horizontal\_flip=True,

fill\_mode='nearest',

validation\_split=0.2 # 20% данных для валидации

)

# Загрузка данных

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(150, 150),

batch\_size=32,

class\_mode='binary',

subset='training'

)

val\_generator = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(150, 150),

batch\_size=32,

class\_mode='binary',

subset='validation'

)

test\_datagen = ImageDataGenerator(rescale=1./255)

test\_generator = test\_datagen.flow\_from\_directory(

test\_dir,

target\_size=(150, 150),

batch\_size=32,

class\_mode='binary'

)

**3. Построение модели**

**Архитектура CNN**:

* Используется сверточная нейронная сеть (CNN) с 3-4 слоями.
* Пример архитектуры:

model = tf.keras.Sequential([

tf.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(150, 150, 3)),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(128, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(512, activation='relu'),

tf.keras.layers.Dropout(0.5),

tf.keras.layers.Dense(1, activation='sigmoid')

])

**Компиляция модели**:

model.compile(

optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy', tf.keras.metrics.Precision(), tf.keras.metrics.Recall()]

)

**4. Обучение модели**

**Параметры обучения**:

* Эпохи: 10-15.
* Callbacks: EarlyStopping (остановка при отсутствии улучшений), ModelCheckpoint (сохранение лучшей модели).

**Пример кода**:

early\_stop = tf.keras.callbacks.EarlyStopping(monitor='val\_loss', patience=3)

checkpoint = tf.keras.callbacks.ModelCheckpoint('best\_model.h5', save\_best\_only=True)

history = model.fit(

train\_generator,

epochs=15,

validation\_data=val\_generator,

callbacks=[early\_stop, checkpoint]

)

**5. Оценка модели**

**Метрики**:

* Точность (Accuracy),
* Чувствительность (Recall),
* Специфичность (Specificity),
* AUC-ROC.

**Пример кода для тестирования**:

# Загрузка лучшей модели

model = tf.keras.models.load\_model('best\_model.h5')

# Оценка на тестовых данных

test\_loss, test\_acc, test\_precision, test\_recall = model.evaluate(test\_generator)

print(f'Test Accuracy: {test\_acc:.2f}, Recall: {test\_recall:.2f}')

# Построение ROC-кривой

from sklearn.metrics import roc\_curve, auc

import matplotlib.pyplot as plt

y\_pred = model.predict(test\_generator)

fpr, tpr, thresholds = roc\_curve(test\_generator.classes, y\_pred)

roc\_auc = auc(fpr, tpr)

plt.figure()

plt.plot(fpr, tpr, label=f'AUC = {roc\_auc:.2f}')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.legend()

plt.show()

**6. Интерпретация результатов**

**Пример вывода**:

* **Accuracy**: 92% на тестовых данных.
* **Recall (чувствительность)**: 95% (модель хорошо обнаруживает пневмонию).
* **Precision**: 90% (меньше ложноположительных срабатываний).
* **AUC-ROC**: 0.97 (высокое качество классификации).

**7. Анализ ошибок**

* Ложноположительные случаи: нормальные снимки, ошибочно классифицированные как пневмония.
* Ложноотрицательные случаи: пропущенные случаи пневмонии.

**8. Пример улучшения модели**

**Transfer Learning с ResNet50**:

base\_model = tf.keras.applications.ResNet50(

weights='imagenet',

include\_top=False,

input\_shape=(150, 150, 3)

)

# Заморозка слоев ResNet

base\_model.trainable = False

model = tf.keras.Sequential([

base\_model,

tf.keras.layers.GlobalAveragePooling2D(),

tf.keras.layers.Dense(1, activation='sigmoid')

])

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

**9. Выводы**

* Классификация медицинских изображений требует тщательной обработки данных и учета дисбаланса классов.
* Transfer Learning значительно улучшает качество моделей.
* Интерпретируемость решений критична для доверия врачей.