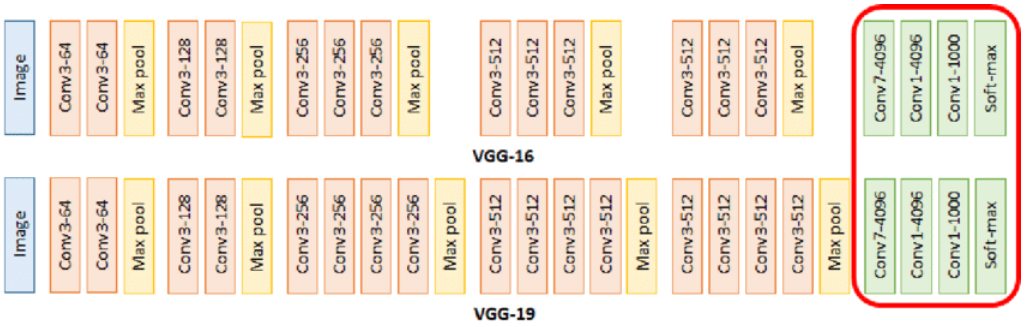
Практическая работа №2

Сверточные нейронные сети. Модели VGG

Цель работы: изучение работы искусственных сверточных нейронный сетей, в частности моделей VGG.

Инструментарий: Язык Python (с использованием IDE (PyCharm, Jupyter, Spyder и т.п.), а также библиотек, содержащих которые нейросетевые модели (Tensorflow, Theano, Keras и т.п.). *//опционально, можно согласовать иной инструмент*



Задача №1.

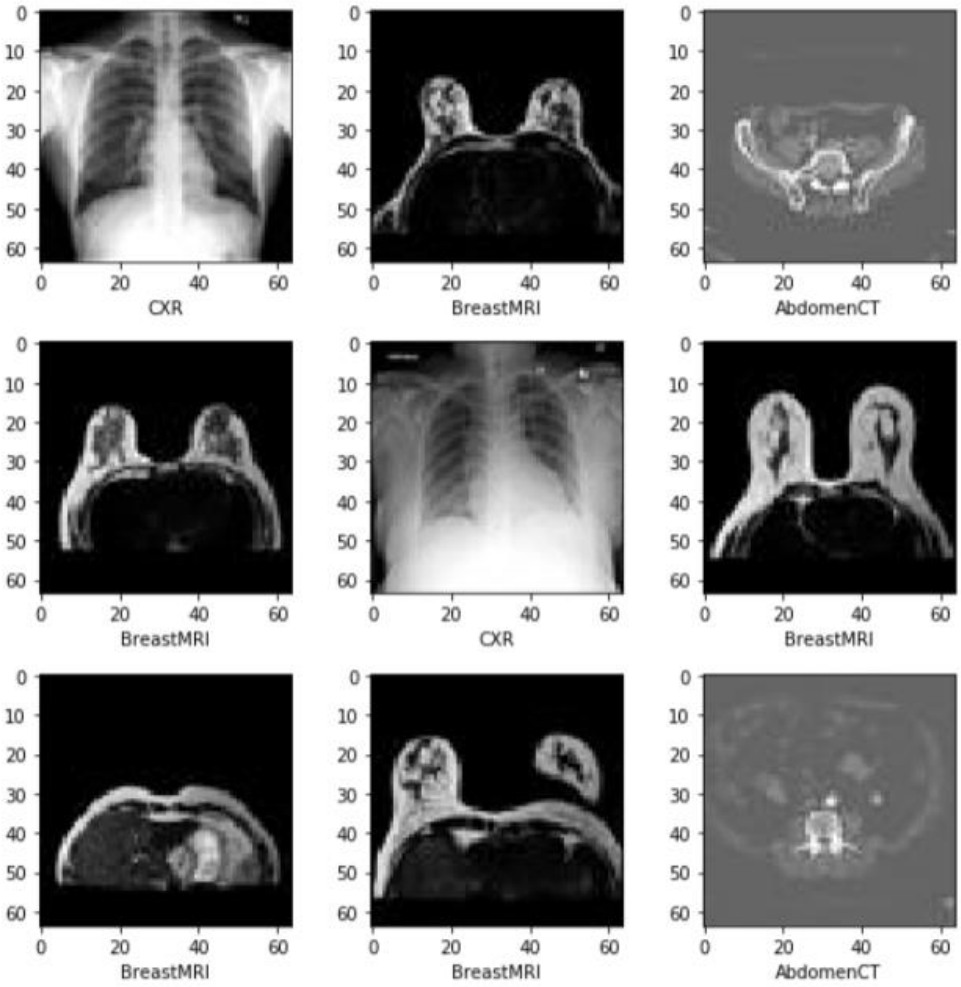
Обучить и использовать модели VGG16 и VGG19 для задач классификации по 6 заданным классам.

Этот набор данных представляет собой простое медицинское изображение в стиле MNIST размером 64x64.;

Первоначально оно было взято из других наборов данных и обработано в таком стиле. Имеется 58954 медицинских изображения, относящихся к 6 классам.

Набор данных содержит 58954 медицинских изображения, относящихся к 6 классам – КТ грудной клетки (10000 изображений), МРТ молочной железы (8954 изображения), CXR (10000 изображений), руки (10000 изображений), КТ головы (10000 изображений), брюшной полости (10000 изображений). Изображения имеют размеры 64 × 64 пикселя.

//Задачей является корректное определение класса изображения.



Пример изображений

Заполнить таблицу для каждой из нейросетевых моделей, в которой включить наборы параметров настройки, характерные для моделей, для каждого из примеров обучения и использования сети (эксперимента).

Таблица 1 –VGG16 сеть (пример)

№ п/п	Функции активации слоя свертки Блока №1	Функции активации слоя свертки Блока №2	…	Функции активации слоя свертки Блока №5	Оптими-затор	Loss функция	метрика
1	relu'	relu'		relu'	Adam (lr=0.0001)	categorical_ crossentropy	accuracy
…	…	…		…	…	…	…

//параметры могут быть выбраны опционально

// значения метрик, полученные в результате компиляции и работы модели привести в отдельных таблицах/логах

Аналогичные действия произвести с моделью VGG19.

Разбиение выборки считать стандартным 80% - обучающая, 20% - тестовая.

Сделать выводы относительно каждой из моделей. Сравнить модели.

Примеры VGG сетей на Python:

VGG16 состоит из нескольких блоков сверток (Convolutional layers), каждый блок включает в себя от двух до четырех слоев свертки (Conv2D) с размером ядра 3x3, за которыми следуют слои нормализации (BatchNormalization), активации ReLU и слой max-pooling (MaxPooling2D). После последнего блока сверток идет последовательность полносвязных слоев (Dense), завершающаяся слоем Softmax для предсказания классов.

Input Layer → Conv Block x 5 → Flatten → Dense x 3 → Output Layer

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
BatchNormalization, Dropout
from tensorflow.keras.optimizers import Adam

# Создаем последовательную модель
model = Sequential()

# Блок 1: два слоя свёртки + pooling
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same',
input_shape=(224, 224, 3)))
model.add(BatchNormalization())
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# Блок 2: два слоя свёртки + pooling
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
```

```
# Блок 3: три слоя свёртки + pooling
model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# Блок 4: три слоя свёртки + pooling
model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# Блок 5: три слоя свёртки + pooling
model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# Полносвязная сеть
model.add(Flatten()) # Превращаем выход предыдущего слоя в вектор
model.add(Dense(units=4096, activation='relu')) # Первый скрытый слой
model.add(Dropout(rate=0.5)) # Регуляризация dropout
model.add(Dense(units=4096, activation='relu')) # Второй скрытый слой
model.add(Dropout(rate=0.5))
model.add(Dense(units=1000, activation='softmax')) # Выходной слой (для ImageNet)

# Компиляция модели
optimizer = Adam(lr=0.0001)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Выведем сводку архитектуры модели
model.summary()
```

Чтобы натренировать модель, вам потребуется набор данных (например, датасет ImageNet или другой подходящий набор данных).

```
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2,
horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory('dataset/training_set',
target_size=(224, 224), batch_size=32, class_mode='categorical')
test_set = test_datagen.flow_from_directory('dataset/test_set', target_size=(224,
224), batch_size=32, class_mode='categorical')

history = model.fit(training_set, epochs=10, validation_data=test_set)
```

Важные моменты реализации:

Классический размер входных изображений для VGG16 — 224×224 пикселей.

Все изображения масштабируются путем деления на 255, чтобы нормализовать значения цветов между 0 и 1.

Регуляризация (Dropout) применяется для предотвращения переобучения.

Каждый последующий блок увеличивает количество фильтров вдвое.

Кросс-энтропия используется в качестве функции потерь для многоклассовой классификации.

Задача №2 (дополнительная)

С использованием простой CNN модели или любой из VGG моделей произвести прогноз смерти пациента от COVID-19.

Обучающая выборка имеет следующие поля:

USMER указывает, лечился ли пациент в медицинских учреждениях первого, второго или третьего уровня.

MEDICAL_UNIT тип учреждения Национальной системы здравоохранения, которое оказывало медицинскую помощь.

ПОЛ 1 - женщина. 2 – мужчина

PATIENT_TYPE — тип ухода, который пациент получал в отделении.

ДАТА СМЕРТИ Если пациент умер, дата смерти, в противном случае - 9999-99-99.

INTUBED, был ли пациент подключен к аппарату искусственной вентиляции легких.

ПНЕВМОНИЯ, в зависимости от того, есть ли у пациента воспаление лёгочной ткани.

ВОЗРАСТ пациента.

БЕРЕМЕННА независимо от того, беременна пациентка или нет.

ДИАБЕТ - независимо от того, есть ли у пациента диабет или нет

Выборка **нуждается** в предобработке и расширении. Необходимо добавить поле – **факт смерти**. Если пациент умер (в наличии дата смерти) – 2, в противном случае – 1. Самостоятельно заполнить поле для текущей выборки.

Прогнозируемым параметром является **факт смерти.**

Привести результаты обучения, тестирования и прогноза при различных наборах параметров, заполнить таблицы аналогичные таблицам в задаче №1.

Сделать выводы относительно возможности использования сетей данного типа в прогнозировании. Какие нейросетевые модели было бы оптимальнее использовать и почему?

Отчет должен содержать:

- титульный лист

- краткие теоретические сведения, в том числе архитектуру ИНС

- таблицы, со сравнением нейросетевых моделей

- экранные формы работы моделей

- исходный код

- датасет (в частности демонстрация разделения на обучающий и тестовый, задачи 2 и 3)

- выводы.

Список используемых источников:

1. VGG16 – Convolutional Network for Classification and Detection URL: https://neurohive.io/en/popular-networks/vgg16/
2. Применение предобученной модели VGG16 для рекомендаций на основе изображений товаров URL: https://habr.com/ru/companies/skillfactory/articles/545384/
3. VGG-16 | CNN model URL: https://www.geeksforgeeks.org/vgg-16-cnn-model/?ysclid=macvkx3att285121563
4. Ghada A. Shadeed, Mohammed A. Tawfeeq and Sawsan M. Mahmoud Automatic Medical Images Segmentation Based on Deep Learning Networks OP Conf. Series: Materials Science and Engineering 870 (2020) 012117 IOP Publishing doi:10.1088/1757-899X/870/1/012117
5. Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni MedMNIST v2- A large-scale lightweight benchmark for 2D and 3D biomedical image classification arXiv:2110.14795v2 [cs.CV] 25 Sep 202