DOI 10.37882/2223-2966.2023.9-2.12

ДИАГНОСТИКА ПНЕВМОНИИ НА РЕНТГЕНОЛОГИЧЕСКИХ СНИМКАХ С ИСПОЛЬЗОВАНИЕМ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ

PNEUMONIA DETECTION IN CHEST X-RAY IMAGES USING A CONVOLUTIONAL NEURAL NETWORK

S. Kasyuk
G. Didenko
O. Stepanova

Summary. The article considers a modern technology of X-ray diagnostics based on the use of a convolutional neural network. A publicly available pneumonia X-ray dataset, provided by Kermany et al., for image-based deep learning is described. A configuration of convolutional neural network built on the GoogLeNet architecture is outlined. The Python program implementing neural network's learning process, calculation of evaluation metrics and pneumonia detection is described. The process of training the convolutional neural network is outlined. The comparison of the obtained evaluation metrics of the binary classifier and results of published international researches is given.

Keywords: X-ray diagnostics, data classification, deep learning, convolutional neural network, Python programming.

Касюк Сергей Тимурович
К.т.н., доцент, ФГБОУ ВО «Южно-Уральский государственный медицинский университет» Министерства здравоохранения Российской Федерации (г. Челябинск)
sergey.kasyk@gmail.com
Диденко Галина Александровна
К.п.н., доцент, ФГБОУ ВО «Южно-Уральский государственный медицинский университет» Министерства здравоохранения Российской Федерации (г. Челябинск)
pga80@mail.ru
Степанова Оксана Александровна
К.п.н., доцент, ФГБОУ ВО «Южно-Уральский государственный медицинский университет» Министерства здравоохранения Российской Федерации (г. Челябинск)
okalst@mail.ru

Аннотация. В статье рассматривается современная технология рентгенодиагностики пневмонии, основанная на использовании сверточной нейронной сети. Описывается набор рентгенологических снимков грудной клетки Д. Кермани, К. Чжана и М. Голдбаума (Kermany dataset), используемый для глубокого обучения. Приводится конфигурация сверточной нейронной сети, построенной на архитектуре GoogLeNet. Описывается программа на языке Python, реализующая обучение нейронной сети, расчет метрик классификации и диагностику пневмонии. Рассматривается процесс обучения сверточной нейронной сети. Дается сравнение полученных метрик бинарной классификации с опубликованными результатами международных исследований.

Ключевые слова: рентгенодиагностика, классификация данных, глубокое обучение, сверточная нейронная сеть, метрики классификации, программирование на Python.

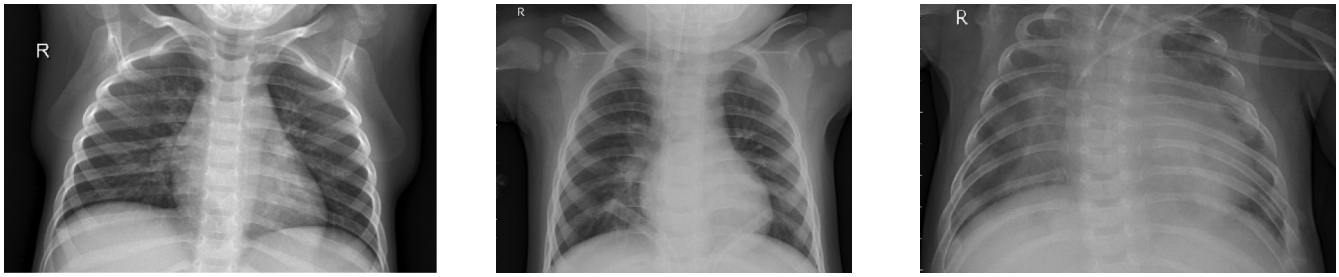
Введение

Эта статья посвящена диагностике пневмонии на рентгенологических снимках грудной клетки с использованием сверточной нейронной сети, построенной на архитектуре GoogLeNet. Глубокое обучение нейронной сети проводилось на общедоступном наборе рентгенограмм Д. Кермани, К. Чжана и М. Голдбаума (Kermany dataset) [1]. Программная реализация сверточной нейронной сети была осуществлена на языке программирования Python с применением библиотек машинного обучения TensorFlow и Keras.

Цель статьи — показать использование технологии сверточных нейронных сетей и глубокого обучения для построения классификатора пневмонии и сравнить полученные результаты с данными опубликованных международных исследований.

1. Особенности диагностики пневмонии

Рентгенография грудной клетки является наиболее часто используемым неинвазивным методом диагностики пневмонии. Белые пятна на рентгенограммах, называемые инфильтратами, отличают пневмонию от здоровых легких. Нормальная рентгенограмма грудной клетки показывает чистые легкие без каких-либо областей аномального помутнения на изображении. Бактериальная пневмония обычно характеризуется фокальной долевой консолидацией. Вирусная пневмо-



а) здодовые легкие б) бактериальная пневмония в) вирусная пневмония

Рис. 1. Примеры рентгенограмм грудной клетки

ния проявляется более диффузным интерстициальным рисунком в легких [1].

Таким образом, рентгенологическая диагностика пневмонии являются сложной задачей, которую решает врач-специалист. На сегодняшний день обученные на тысячах изображений сверточные нейронные сети способны быстро классифицировать большое количество рентгенограмм и могут оказать консультационную помощь специалисту, уменьшив количество ошибок диагностики.

2. Исходные данные для машинного обучения

Исходными данными для обучения сверточной нейронной сети послужил общедоступный набор данных Д. Кермани, К. Чжана и М. Голдбаума (*Kermany dataset*), содержащий 5863 рентгенограммы грудной клетки. Все снимки были сделаны в *Женском и детском медицинском центре г. Гуанчжоу (КНР)* в рамках обзорной рентгенографии органов грудной клетки (передняя и задняя прямая проекция), проводимой у педиатрических пациентов в возрасте от одного до пяти лет. Обследования выполнялись в рамках оказания медицинской помощи пациентам. Примеры рентгенограмм из этого набора для здоровых легких, бактериальной пневмонии и вирусной пневмонии приведены на рис. 1 [1].

Используемый набор данных прошел контроль качества — все низкокачественные и неразборчивые рентгенограммы были отбракованы. Диагнозы бактериальной и вирусной пневмонии были поставлены двумя врачами-экспертами; во избежание ошибок этот набор данных был проверен третьим экспертом.

Рентгенограммы в формате *JPEG* расположены в трех папках (*train*, *test* и *val — обучающая*, *тестовая*, *проверочная выборки* соответственно) и содержит подпапки *PNEUMONIA* и *NORMA* для каждой категории изображений. Этот набор является несбалансированным, поскольку в *обучающей выборке* содержится в три раза больше рентгенограмм с пневмонией, чем с легкими без особенностей. Диаграмма распределения рентгенограмм в *обучающей выборке* представлена на рис. 2.

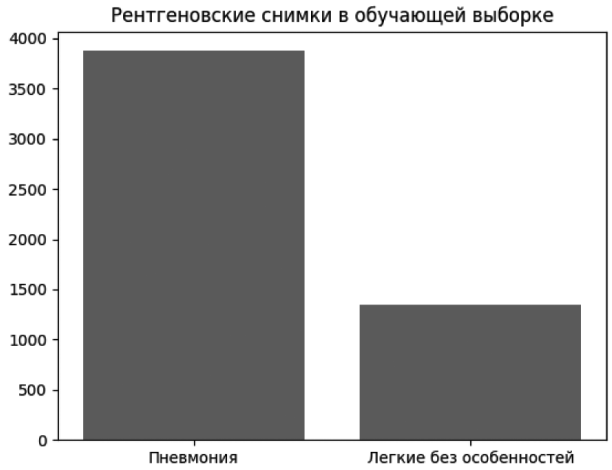


Рис. 2. Диаграмма распределения рентгенограмм в обучающей выборке

3. Использованная архитектура сверточной нейронной сети GoogLeNet

Для решения задачи диагностики пневмонии была использована классическая *22-слойная архитектура GoogleNet (Inception-v1)*, представленная в работе Кристиана Сегеди, Вэй Лю, Янцина Цзя и др. «Углубление с помощью свертки» («Going deeper with convolutions») [2]. Отличительной чертой этой архитектуры является улучшенное использование вычислительных ресурсов внутри сети за счет применения модуля Inception [3].

На рис. 3 показана архитектура модуля Inception-v1, включающая: 1) *сверточные слои (Convolution)* с функцией активации *ReLU (Rectified Linear Unit)*, 2) *слой объединения по максимуму (Max Pooling)* и 3) *слой конкатенации в глубину (Concatenate)*. *Сверточные слои* содержат: *f — фильтры (filters); k — ядра (kernels); s — страйды (strides); p — дополнения нулями (padding = SAME)* [3, 4].

В табл. 1 представлена архитектура GoogLeNet, включающая девять модулей Inception. Сеть имеет глубину 22 слоя, если считать только слои с параметрами. Общее количество слоев (самостоятельных строительных блоков), используемых для создания сети, составляет около 100. Однако это число зависит от параметров машинного обучения [2].

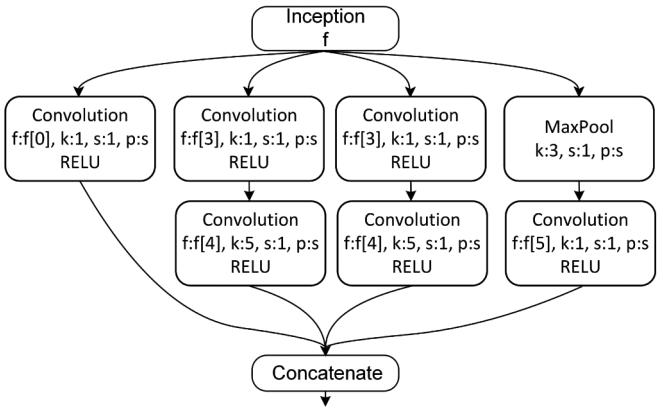


Рис. 3. Схема архитектуры Inception

Во всех сверточных слоях применяется функция активации *ReLU*. Столбцы таблицы «#3×3 reduce (сжатие)» и «#5×5 reduce (сжатие)» означают количество фильтров 1×1 в слое сжатия, расположенном до сверток 3×3 и 5×5. В столбце «pool proj» располагается количество *фильтров 1×1 в слое проекции (projection layer)* после *слоя объединения по максимуму (MaxPool)* [2].

На вход этой сети поступает RGB изображение размером 224×224 пикселей, проходит через последовательность *сверточных слоев (Convolution), слоев объединения по максимуму (MaxPool)*, модулей *Inception, слой объединения по среднему (AvgPool), слой отключения (DropOut)*, используемый для регуляризации, и заключительный *полносвязный слой (Flatten)* с функцией активации *Softmax*, используемый для получения оценочных вероятностей двух классов [2–4].

4. Реализация сети GoogLeNet (Inception-v1) на языке программирования Python

Программная реализация модуля Inception-v1 на языке Python представлена ниже [4]:

```
def inception_block(x, filters):
t1 = Conv2D(filters=filters[0], kernel_size=1,
activation='relu')(x)
t2 = Conv2D(filters=filters[1], kernel_size=1,
 activation='relu')(x)
t2 = Conv2D(filters=filters[2], kernel_size=3,
padding='same', activation='relu')(t2)
t3 = Conv2D(filters=filters[3], kernel_size=1,
activation='relu')(x)
t3 = Conv2D(filters=filters[4], kernel_size=5,
padding='same', activation='relu')(t3)
t4 = MaxPool2D(pool_size=3, strides=1, padding='same')(x)
t4 = Conv2D(filters=filters[5], kernel_size=1,
activation='relu')(t4)
 output = Concatenate()([t1, t2, t3, t4])
 return output
```

Таблица 1.

Архитектура сети GoogLeNet

Type	Patch size/ Stride	Output size	Depth	#I×I	#3×3 reduce	#3×3	#5×5 reduce	#5×5	Pool proj
Convolution	7×7/2	112×112×64	1						
Max pool	3×3/2	56×56×64	0						
Convolution	3×3/1	56×56×192	2		64	192			
Max pool	3×3/2	28×28×192	0						
Inception (3a)		28×28×256	2	64	96	128	16	32	32
Inception (3b)		28×28×480	2	128	128	192	32	96	64
Max pool	3×3/2	14×14×480	0						
Inception (4a)		14×14×512	2	192	96	208	16	48	64
Inception (4b)		14×14×512	2	160	112	224	24	64	64
Inception (4c)		14×14×512	2	128	128	256	24	64	64
Inception (4d)		14×14×528	2	112	144	288	32	64	64
Inception (4e)		14×14×832	2	256	160	320	32	128	128
Max pool	3×3/2	7×7×832	0						
Inception (5a)		7×7×832	2	256	160	320	32	128	128
Inception (5b)		7×7×1024	2	384	192	384	48	128	128
Avg pool	7×7/1	1×1×1024	0						
Dropout (40 %)		1×1×1024	0						
Linear		1×1×2	1						
Softmax		1×1×2	0						

Программная реализация архитектуры GoogLeNet на языке Python, работающей с двумя классами изображений, представлена ниже [4]:

```
INPUT_SHAPE = 224, 224, 3
input = Input(INPUT_SHAPE)
x = Conv2D(filters=64, kernel_size=7, strides=2,
padding='same', activation='relu')(input)
x = MaxPool2D(pool_size=3, strides=2, padding='same')(x)
x = Conv2D(filters=64, kernel_size=1, activation='relu')(x)
x = Conv2D(filters=192 (128), kernel_size=3,
padding='same',
activation='relu')(x)
x = MaxPool2D(pool_size=3, strides=2)(x)
x = inception_block(x, filters=[64, 96, 128, 16, 32, 32])
x = inception_block(x, filters=[128, 128, 192, 32, 96, 64])
x = MaxPool2D(pool_size=3, strides=2, padding='same')(x)
x = inception_block(x, filters=[192, 96, 208, 16, 48, 64])
x = inception_block(x, filters=[160, 112, 224, 24, 64, 64])
x = inception_block(x, filters=[128, 128, 256, 24, 64, 64])
x = inception_block(x, filters=[112, 144, 288, 32, 64, 64])
x = inception_block(x, filters=[256, 160, 320, 32, 128, 128])
x = MaxPool2D(pool_size=3, strides=2, padding='same')(x)
x = inception_block(x, filters=[256, 160, 320, 32, 128, 128])
x = inception_block(x, filters=[384, 192, 384, 48, 128, 128])
x = AvgPool2D(pool_size=7, strides=1)(x)
x = Dropout(rate=0.4)(x)
x = Flatten()(x)
output = Dense(units=2, activation='softmax')(x)
from tensorflow.keras import Model
model = Model(inputs=input, outputs=output)
```

5. Разработка программы диагностики пневмонии на рентгенологических снимках грудной клетки

Разработка программы диагностики пневмонии осуществлялась на языке Python с использованием открытых библиотек машинного обучения *TensorFlow* и *Keras*.

Схема программы представлена на рис. 4. Основные операции, выполняемые программой, следующие:

1) загрузка рентгенограмм в списки *train_data, test_data, val_data*, применяемые для обучения, теста и проверки (валидации);

2) конфигурирование сверточной нейронной сети GoooLeNet (Inception-v1);

3) инициализация параметров обучения: функции потерь (*loss function*), оптимизатора (*optimizer*), ранней остановки (*early_stop*), количества эпох обучения (*epochs*) и размера пакета (*batch_size*);

4) обучение сверточной нейронной сети с использованием метода *fit*;

5) сохранение и загрузка обученной сети в формате *h5* с использованием метода *save*;

6) расчет метрик классификации на тестовом наборе данных с использованием функций *classification_report* и *confusion_matrix*;

7) расчет вероятностей принадлежности к классам «норма» и «пневмония» для выбранного файла;

8) вывод диагноза.

Разработанная программа прошла *государственную регистрацию программы для ЭВМ* под названием «Программа для диагностики пневмонии на рентгенограммах грудной клетки с помощью сверточной нейронной сети» (номер регистрации 2023619898) [5].

6. Обучение сверточной нейронной сети

Обучение нейронной сети требует выполнения следующих шагов:

1. Вначале в модель необходимо подать *обучающие данные* (*train_data*).
2. Затем модель *учится ассоциировать изображения* с правильными классами «норма» и «пневмония».
3. В конце модель *проверяется на тестовых данных* (*test_data*) на соответствие предсказанных классов.

Перед началом обучением сети, на этапе компиляции модели (*compile*), были использованы следующие параметры:

1) *минимизируемая во время обучения функция потерь* (*loss function*) — *кросс-энтропия* для бинарной классификации (*binary_crossentropy*);

2) *оптимизатор* (*optimizer*) *с коэффициентом скорости обучения lr* (*learning_rate*), равным 0,0001, и *регуляризацией весов decay* (*weight decay*), равной 0,00001;

3) *метрика классификации* (*metrics*) — *точность* (*accuracy*), равная доле правильно классифицированных изображений.

Эти *параметры компиляции* задавались в программе следующим образом:

```
opt = Adam(lr=0.0001, decay=1e-5)
model.compile(loss='binary_crossentropy',
metrics=['accuracy'], optimizer=opt)
```

Количество эпох обучения (*nb_epochs*) задавалось равным 30; *размер пакета* (*batch_size*) — 256.

Для остановки обучения нейронной сети при достижении заданной точности были использованы следующие параметры:

1) *показатель производительности для мониторинга* (*monitor*) — *потери* (*loss*);

2) *количество эпох без улучшения* (*patience*), после которых обучение останавливается, равно 3.

Эти *параметры остановки* задавались в программе следующим образом:

```
early_stop = EarlyStopping(monitor='loss',
patience=3, verbose=1)
```

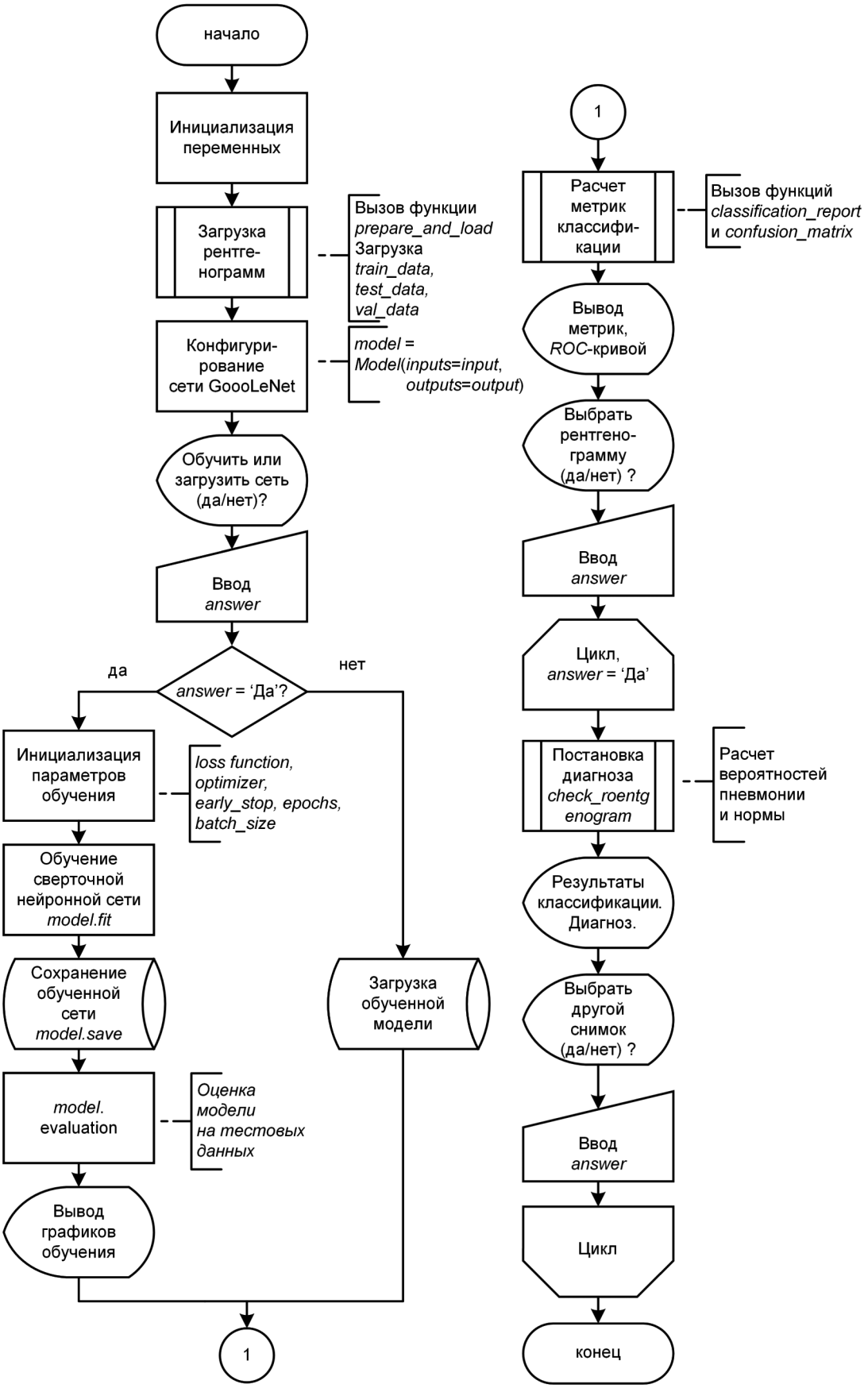


Рис. 4. Схема программы диагностики пневмонии на рентгенологических снимках грудной клетки

Обучение нейронной сети производилось с использованием метода *fit*. Аргумент *class_weight* при этом учитывал несбалансированность обучающих данных.

```
history = model.fit(train_data_gen, epochs=nb_epochs,
steps_per_epoch=nb_train_steps,
validation_data=(val_data,val_labels),
callbacks=[early_stop],
class_weight ={0:1.0, 1:0.4})
```

Количество эпох обучения составило 21. Количество обученных параметров сети GoogLeNet — 5975602. Время обучения сети на ПК с процессором AMD Ryzen 5 3600 и 16 Гб оперативной памяти, работающими частоте 3200 МГц, заняло около двух часов.

Результаты обучения на последней эпохе следующие:

```
Epoch 21/50
20/20 [==============================] — 383s
19s/step — loss: 0.0677 — accuracy: 0.9504 — val_loss:
0.2728 — val_accuracy: 0.9375
```

Полученная сверточная нейронная сеть достигла на *обучающих данных* точности, равной 0,9504 (95,04 %), на *данных для валидации* — точности, равной 0,9375 (93,75 %).

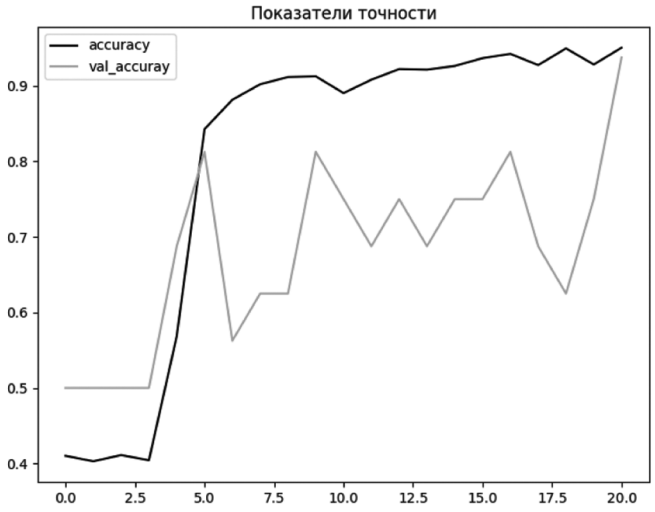


Рис. 5. Графики *точности классификации* на тестовой (*accuracy*) и проверочной (*val_accuracy*) выборках в зависимости от эпох обучения

Вывод графиков *точности классификации* на *обучающей* (*accuracy*) и *проверочной* (*val_accuracy*) выборках в зависимости от циклов обучения (рис. 5) был осуществлен с помощью следующего фрагмента кода:

```
plt.title('Показатели точности')
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['accuracy', 'val_accuray'])
plt.show()
```

Очевидно, что на *двадцать первой эпохе обучения* значения точности сети на *обучающей* и *проверочной* выборках практически сравнялись, то есть была обеспечена сходимость модели.

7. Полученные метрики классификации

Вывод текстового отчета, содержащего основные метрики классификации для тестовой выборки, был осуществлен следующим образом:

```
pred = model.predict(test_data, batch_size=16)
pred = np.argmax(pred, axis=-1)
labels = np.argmax(test_labels, axis=-1)
from sklearn.metrics import classification_report
print(classification_report(labels, pred))
```

Рассчитанные *метрики классификации* приведены ниже:

```
precision recall f1-score support
0 0.89 0.80 0.84 234
1 0.89 0.94 0.91 390
accuracy 0.89 624
macro avg 0.89 0.87 0.88 624
weighted avg 0.89 0.89 0.89 624
```

В колонках этого отчета расположены [3, 6]:

1) *precision* (*точность*):

$$precision = TP / (TP + FP), \quad (1)$$

где *TP* — это количество истинно положительных классификаций, *FP* — количество ложноположительных классификаций;

2) *recall* (*полнота*):

$$Recall = TP / (TP + FN), \quad (2)$$

где *FN* — это количество ложноотрицательных классификаций;

3) *F*1-*score* (оценки *F*1) — средневзвешенное гармоническое значение *precision* и *recall*:

$$F1 - score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}; \quad (3)$$

4) *support* (*поддержка*) — количество правильной классификации для каждого класса объектов.

В строках этого отчета располагаются [7]:

1) *классы объектов*: 0 — «норма»; 1 — «пневмония»;

2) значения *усредненных метрик*: *точность* (*accuracy*), *макроусреднение* (*macro avg*) и *взвешенное усреднение* (*weighted avg*).

Нейронная сеть показала одинаковую *точность* (*precision*), равную 0,89, на рентгенограммах классов «норма» и «пневмония». Значение *полноты* (*recall*) для класса «пневмония», равное 0,94, оказалось выше. Таким образом, классификатор лучше диагностирует класс «пневмония».

Вывод *матрицы ошибок* (рис. 6) был произведен с помощью программного кода, представленного ниже:

```
cm = confusion_matrix(labels, pred)
plot_confusion_matrix(cm,figsize=(12,8), hide_ticks = True,
 cmap = plt.cm.Blues)
plt.title('Матрица ошибок')
plt.xticks(range(2), ['Normal', 'Pneumonia'], fontsize = 16)
plt.yticks(range(2), ['Normal', 'Pneumonia'], fontsize = 16)
plt.show()
```

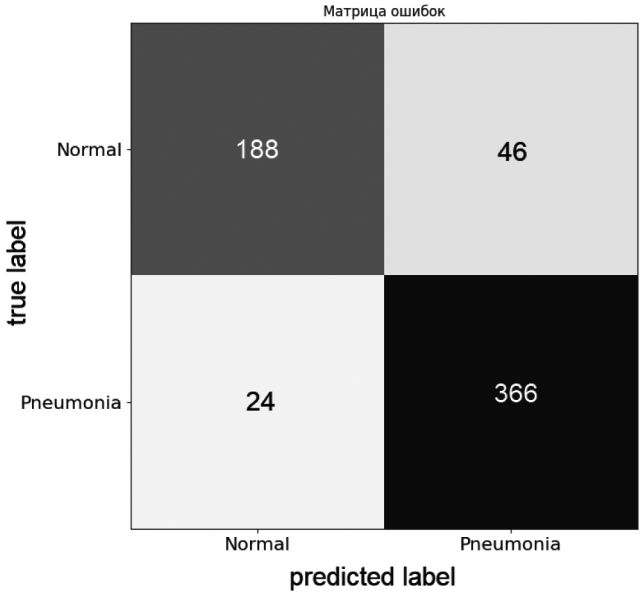


Рис. 6. Матрица ошибок

Используя полученную матрицу ошибок, рассчитаем следующие показатели:

1) *точность* (*accuracy*) — долю правильных классификаций:

$$accuracy = (TP + TN)/(TP + FP + FN + TN) =$$
$$= (188 + 366) /(188 + 46 +24 +366) = 0{,}8878\ (88{,}78\ \%); \quad (4)$$

2) *чувствительность* (*sensitivity*) — долю положительных образцов, которые корректно обнаружены классификатором:

$$sensitivity = TP / (TP + FN) =$$
$$= 188 / (188 + 24) = 0{,}8868\ (88{,}68\ \%). \quad (5)$$

3) *специфичность* (*specificity*) — долю отрицательных образцов, которые корректно обнаружены классификатором:

$$specificity = TN / (TN + FP) =$$
$$= 366 / (366 + 46) = 0{,}8883\ (88{,}83\%). \quad (6)$$

Вывод *ROC*-кривой (рис. 7) был произведен с помощью следующего программного кода:

```
import matplotlib.pyplot as plt
plt.title('ROC-кривая')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('Доля верных положительных классификаций')
plt.xlabel('Доля ложных положительных классификаций')
plt.show()
```

Значение *площади под кривой* (*AUC*), равное 0,87, говорит о *хорошем* качестве классификации нейронной сети, достигнутом на тестовых данных.

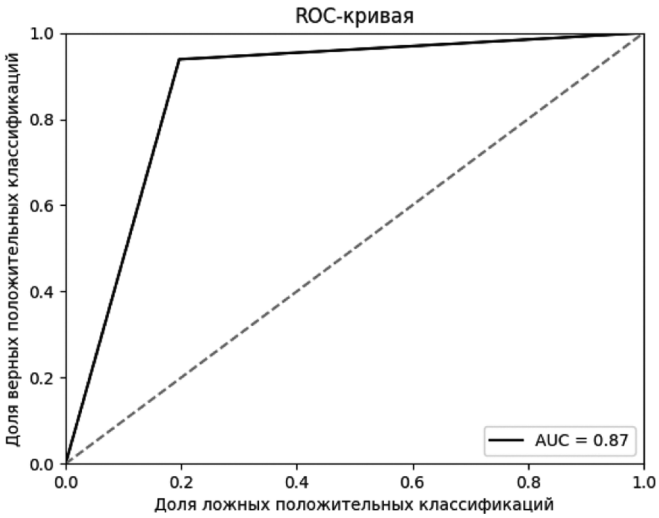


Рис. 7. ROC-кривая

8. Классификация рентгенологических снимков с помощью обученной сверточной нейронной сети

Классификация рентгенологических снимков осуществлялась с помощью разработанной функции *check_roentgenogram:*

```
answer = mb.askyesno(title="Вопрос",
message=»Выбрать рентгенологический снимок для
классификации?»)
while answer:
file_name = fd.askopenfilename()
check_roentgenogram(file_name)
answer = mb.askyesno(title="Вопрос",
message=»Выбрать другой рентгенологический снимок
для классификации?»)
```

Оператор, отвечая на вопрос на рис. 8, выбирает конкретный снимок.

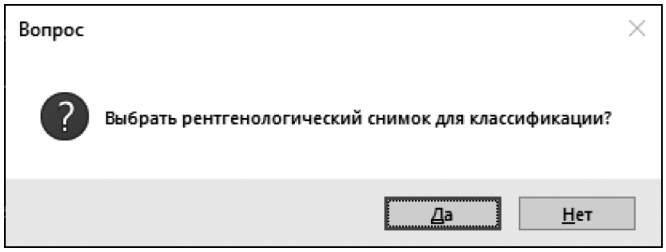


Рис. 8. Окно выбора

Программа проводит классификацию рентгенограммы и выводит на экран значения вероятностей принадлежности снимка к классам «норма» и «пневмония». Рентгенограмма принадлежит к тому классу, для которого вероятность выше.

Пример классификации рентгенограммы приведен ниже:

Вероятность нормы и пневмонии соответственно равны:
[[5.4182870e-05 9.9994576e-01]]
На рентгеновском снимке обнаружена пневмония.

Результат классификации приведен на рис. 9.

Рис. 9. Результат классификации рентгенологического снимка

9. Сравнение полученных метрик классификации с результатами опубликованных исследований

Рентгенологические снимки грудной клетки Д. Кермани, К. Чжана и М. Голдбаума (Kermany dataset) являются популярным медицинским набором данных, используемым международными научными коллективами для исследований в области глубокого обучения. Так, в работе [8] приведен обзор статей, повещённых этому набору: различные группы исследователей, применяя архитектуры сверточных нейронных сетей GoogLeNet, ResNet-18 и DenseNet-121, достигли на этом наборе значений accuracy и AUC, превышающих 0,97[1].

Авторы этой статьи достигли значений accuracy, равного 0,89, и AUC, равного 0,87, что на 10 % ниже лучших опубликованных результатов зарубежных исследований. Повышение точности классификации возможно осуществить за счет подбора параметров обучения сети, а также изменения её архитектуры.

Заключение

Диагностика пневмонии на рентгенологических снимках может эффективно осуществляться с использованием сверточной нейронной сети. Авторы статьи разработали программу на языке Python, осуществляющую обучение нейронной сети на базе архитектуры GoogLeNet, расчет метрик классификации и диагностику пневмонии. Было получено свидетельство о государственной регистрации программы для ЭВМ №2023619898. Метрики классификации, достигнутые на наборе рентгенологических снимков Д. Кермани, К. Чжана и М. Голдбаума (Kermany dataset), показали значения accuracy, равное 0,89, и AUC, равное 0,87, что на 10 % ниже лучших результатов опубликованных зарубежных исследований. Повышение точности классификации возможно осуществить за счет подбора параметров обучения сети, а также изменения её архитектуры. Дальнейшие работы в этом направлении требуют сбора десятков тысяч рентгенограмм с поставленными диагнозами и проведения надлежащей клинической валидации.

[1] https://doi.org/10.1371/journal.pone.0256630.t009

ЛИТЕРАТУРА

1. Chest X-Ray Images (Pneumonia) [Электронный ресурс]. — Режим доступа: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia (дата обращения: 06.06.2023).
2. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., et al. Going deeper with convolutions. Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition. pp. 1–9 (2015).
3. Жерон О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем / О. Жерон; пер. с англ. и ред. Ю.Н. Артеменко. — СпБ.: ООО «Альфа-книга»: 2018. — 688 с.
4. Implementation of GoogLeNet [Электронный ресурс]. — Режим доступа: https://github.com/Machine-Learning-Tokyo/CNN-Architectures/tree/master/Implementations/GoogLeNet (дата обращения: 06.06.2023).
5. Федеральная служба по интеллектуальной собственности. Номер регистрации (свидетельства): 2023619898. Название программы для ЭВМ: Программа для диагностики пневмонии на рентгенограммах грудной клетки с помощью сверточной нейронной сети. — Режим доступа: https://www.fips.ru/iiss/document.xhtml?faces-redirect=true&id=078a814e2920f1549ec728f09adba938 (дата обращения: 06.06.2023).

6. Sklearn.metrics.precision_recall_fscore_support. — Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision _recall_ fscore_support.html (дата обращения: 06.06.2023).
7. https://scikit-learn.org/stable/modules/classes.html — module-sklearn.metricsSklearn.metrics.classification_report. — Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html (дата обращения: 03.06.2023).
8. Kundu R, Das R, Geem ZW, Han G-T, Sarkar R (2021) Pneumonia detection in chest Xray images using an ensemble of deep learning models. PLoS ONE 16(9): e0256630. https://doi.org/10.1371/journal.pone.0256630.