ДОДАТОК А

Вихідний код програми

```
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow.keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D,
Flatten, Dropout, BatchNormalization
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import
classification_report,confusion_matrix
from tensorflow.keras.callbacks import ReduceLROnPlateau
import cv2
import os
import numpy as np
import pandas as pd

labels = ['PNEUMONIA', 'NORMAL']
img_size = 150
def get_training_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img),
cv2.IMREAD_COLOR)
                resized_arr = cv2.resize(img_arr, (img_size,
img_size))
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)

train = get_training_data('/content/drive/My
Drive/pneumonia_dataset_new/chest_xray/train')
test = get_training_data('/content/drive/My
Drive/pneumonia_dataset_new/chest_xray/test')
val = get_training_data('/content/drive/My
Drive/pneumonia_dataset_new/chest_xray/val')

l = []
```

```
for i in train:
    if(i[1] == 0):
        l.append("Pneumonia")
    else:
        l.append("Normal")
sns.set_style('darkgrid')
sns.countplot(l)

plt.figure(figsize = (5,5))
plt.imshow(train[0][0], cmap='gray')
plt.title(labels[train[0][1]])

plt.figure(figsize = (5,5))
plt.imshow(train[-1][0], cmap='gray')
plt.title(labels[train[-1][1]])

x_train = []
y_train = []

x_val = []
y_val = []

x_test = []
y_test = []

for feature, label in train:
    x_train.append(feature)
    y_train.append(label)

for feature, label in test:
    x_test.append(feature)
    y_test.append(label)

for feature, label in val:
    x_val.append(feature)
    y_val.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255

# resize data for deep learning
x_train = x_train.reshape(-1, img_size, img_size, 3)
y_train = np.array(y_train)

x_val = x_val.reshape(-1, img_size, img_size, 3)
y_val = np.array(y_val)

x_test = x_test.reshape(-1, img_size, img_size, 3)
```

```
y_test = np.array(y_test)

datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over
the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs
by std of the dataset
        samplewise_std_normalization=False,  # divide each inp
ut by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range = 30,  # randomly rotate images in the
range (degrees, 0 to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizo
ntally (fraction of total width)
        height_shift_range=0.1,  # randomly shift images verti
cally (fraction of total height)
        horizontal_flip = True,  # randomly flip images
        vertical_flip=False)  # randomly flip images

datagen.fit(x_train)

model = Sequential()
model.add(Conv2D(32 , (3, 3) , strides = 1 , padding = 'same'
, activation = 'relu' , input_shape = (150, 150, 3)))
model.add(Conv2D(32 , (3, 3) , strides = 1 , padding = 'same'
, activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2, 2) , strides = 2 , padding = 'same'))
model.add(Dropout(0.2))
model.add(Conv2D(64 , (3, 3) , strides = 1 , padding = 'same'
, activation = 'relu'))
model.add(Conv2D(64 , (3, 3) , strides = 1 , padding = 'same'
, activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Dropout(0.2))
model.add(Conv2D(128 , (3, 3) , strides = 1 , padding = 'same'
 , activation = 'relu'))
model.add(Conv2D(128 , (3, 3) , strides = 1 , padding = 'same'
 , activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2, 2) , strides = 2 , padding = 'same'))
model.add(Dropout(0.2))
model.add(Conv2D(256 , (3, 3) , strides = 1 , padding = 'same'
 , activation = 'relu'))
model.add(Conv2D(256 , (3, 3) , strides = 1 , padding = 'same'
 , activation = 'relu'))
model.add(BatchNormalization())
```

```
model.add(MaxPool2D((2, 2) , strides = 2 , padding = 'same'))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(units = 512 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 1 , activation = 'sigmoid'))
model.compile(optimizer = "rmsprop" , loss = 'binary_crossentr
opy' , metrics = ['accuracy'])
model.summary()

learning_rate_reduction = ReduceLROnPlateau(monitor='val_accur
acy', patience = 2, verbose = 1, factor = 0.3, min_lr = 0.0000
01)

history = model.fit(datagen.flow(x_train, y_train, batch_size
= 32), epochs = 25, validation_data = datagen.flow(x_val, y_va
l), callbacks = [learning_rate_reduction])

print("Loss of the model is - " , model.evaluate(x_test,y_test
)[0])
print("Accuracy of the model is - " , model.evaluate(x_test,y_
test)[1]*100 , "%")

epochs = [i for i in range(25)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20,10)

ax[0].plot(epochs , train_acc , 'go-
' , label = 'Training Accuracy')
ax[0].plot(epochs , val_acc , 'ro-
' , label = 'Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")

ax[1].plot(epochs , train_loss , 'g-
o' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'r-
o' , label = 'Validation Loss')
ax[1].set_title('Testing Accuracy & Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Training & Validation Loss")
```

```
plt.show()

predictions = model.predict_classes(x_test)
predictions = predictions.reshape(1,-1)[0]
predictions[:15]

print(classification_report(y_test, predictions, target_names
= ['Pneumonia (Class 0)','Normal (Class 1)']))

cm = confusion_matrix(y_test,predictions)

cm = pd.DataFrame(cm , index = ['0','1'] , columns = ['0','1']
)

plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth
= 1 , annot = True, fmt='',xticklabels = labels,yticklabels =
labels)
correct = np.nonzero(predictions == y_test)[0]
incorrect = np.nonzero(predictions != y_test)[0]

i = 0
for c in correct[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[c].reshape(150,150,3), cmap="gray", inte
rpolation='none')
    plt.title("Predicted Class {},Actual Class {}".format(pred
ictions[c], y_test[c]))
    plt.tight_layout()
    i += 1

i = 0
for c in incorrect[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[c].reshape(150,150, 3), cmap="gray", int
erpolation='none')
    plt.title("Predicted Class {} ,Actual Class {}".format(pre
dictions[c], y_test[c]))
    plt.tight_layout()
    i += 1
```

ДОДАТОК Б

Відомість атестаційної роботи магістра

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	80 с.
	Інші документи	
2.	Презентаційні матеріали	плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	

Розробив		Легенька А. В.			Дослідження штучних нейронних мереж в задачі діагностики захворювань за рентгенівськими зображеннями	Шифр групи	Код напр./спец.
Перевірив		Магдаліна І. В.				СШІм-19-1	122
Н.контр.		Малєєва І. А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					