КУРСОВОЙ ПРОЕКТ

по дисциплине «Глубокое обучение»

|  |  |
| --- | --- |
| Студенты | Дудина А.Н.  Забродина Е.С.  Русаков Е.С. |
| Группа | 3530904/70103 |
| Преподаватель | Малеев О.Г. |

Оглавление

[Введение 3](#_Toc69134423)

[Используемые датасеты 3](#_Toc69134424)

[Архитектура нейронной сети 4](#_Toc69134425)

[Архитектура нейронной сети представлена на картинке: 4](#_Toc69134426)

[Результаты обучения 5](#_Toc69134427)

[Работа программы 5](#_Toc69134428)

# **Введение**

Использование глубокого обучения в медицине активно развивается благодаря большому количеству размеченных снимков, возросшей вычислительной мощности и появлению облачных хранилищ данных. Нейронные сети влияют на состояние медицины на трех уровнях:

· помогают врачам быстро и точно интерпретировать изображения;

· уменьшают количество врачебных ошибок;

· помогают пациентам самостоятельно анализировать данные с помощью датчиков, чтобы контролировать свое состояние.

Поэтому темой курсового проекта было выбрано определение заболеваний при помощи нейронных сетей.

# **Используемые датасеты**

Для обучения и тестирования нейронной сети мы использовали следующие датасеты:

Breast Histopathology Images

Этот датасет состоит из 162 отсканированных при 40-кратном увеличении изображений образцов гистологии рака молочной железы. Из них получены 277 524 изображения размером 50 х 50 (198 738 отрицательных и 78 786 положительных). Имя файла каждого изображения имеет формат: u\_xX\_yY\_classC.png — > пример 10253\_idx5\_x1351\_y1101\_class0.png . Где u-идентификатор пациента (10253\_idx5), X-координата x того места, откуда было вырезано это изображение, Y-координата y того места, откуда было вырезано это изображение, а C указывает класс 0(рак не выявлен) и 1(рак выявлен).

Ссылка на данный датасет: https://www.kaggle.com/paultimothymooney/breast-histopathology-images

Chest X-rays (Pneumonia)

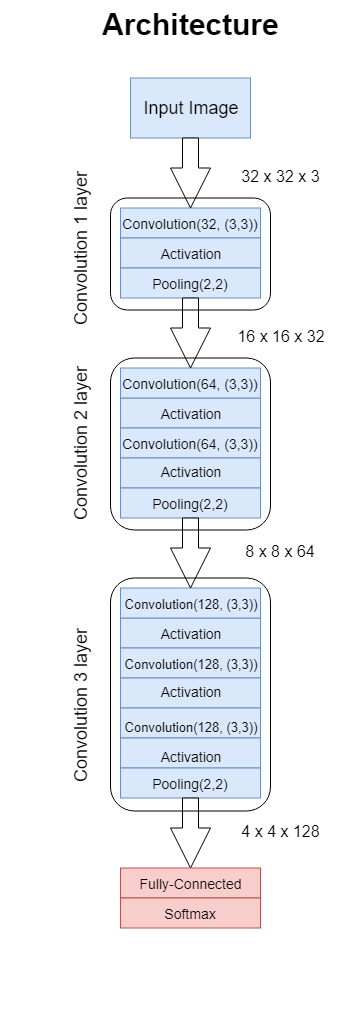
Набор данных разделен на обучающую и тестовую части, каждая из которых содержит рентгеновские снимки, где заболевание выявлено (папка PNEUMONIA) и где его нет (папка NORMAL).

Ссылка на данный датасет: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

# **Архитектура нейронной сети**

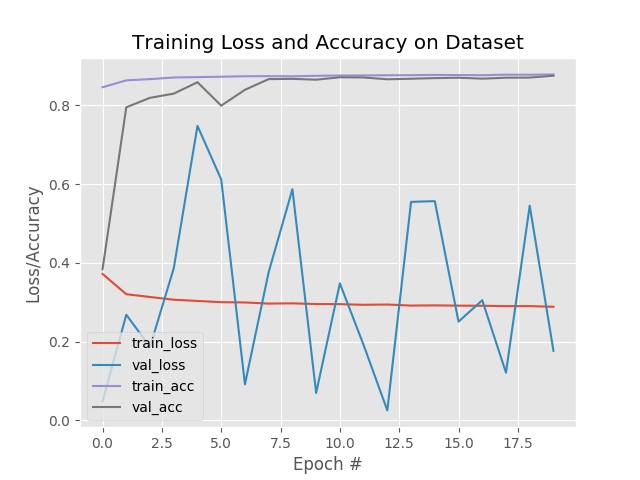
Данный проект содержит сверточную нейронную сеть, способную определять рак груди на изображениях гистологии и пневмонию на рентгеновских снимках легких. Он был основан на упрощенной модели VGGNet с 3 сверточными слоями. Проект включает обучение, сами модели и функции прогнозирования.

# Архитектура нейронной сети представлена на картинке:



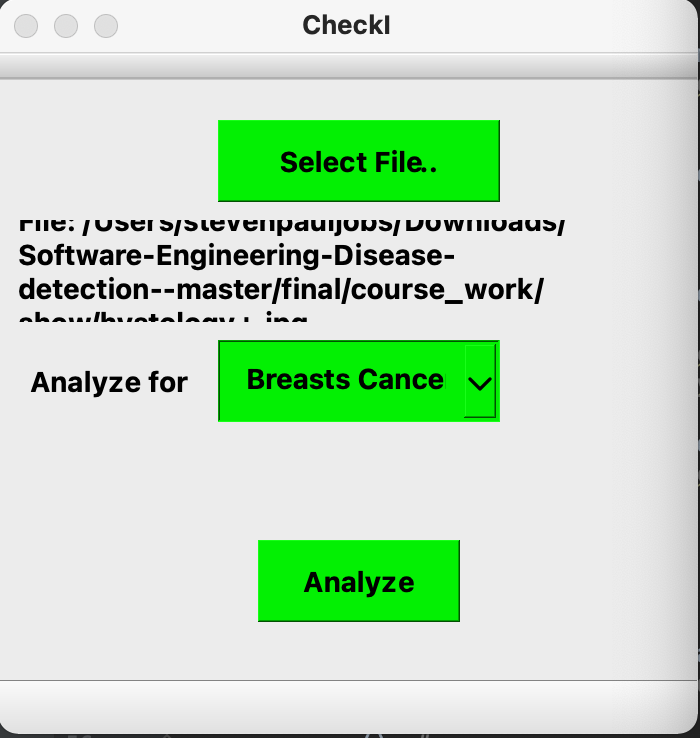
# **Результаты обучения**

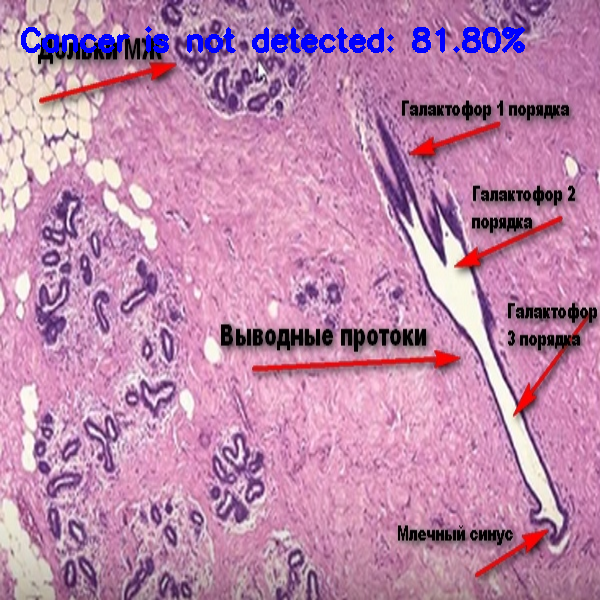
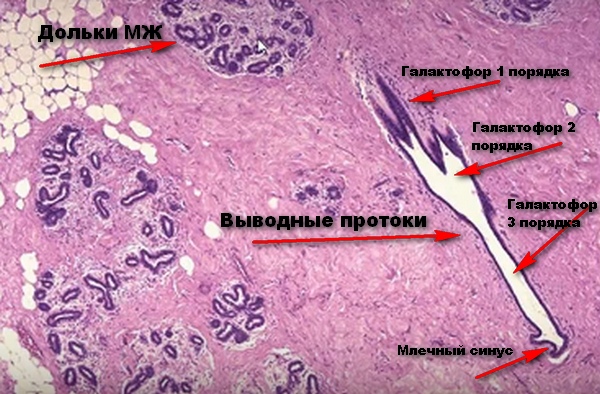
Эта нейронная сеть была обучена на 277524 изображениях гистологии груди и 5863 изображениях рентгеновских снимков легких. В процессе обучения мы достигли точности около 95% процентов:

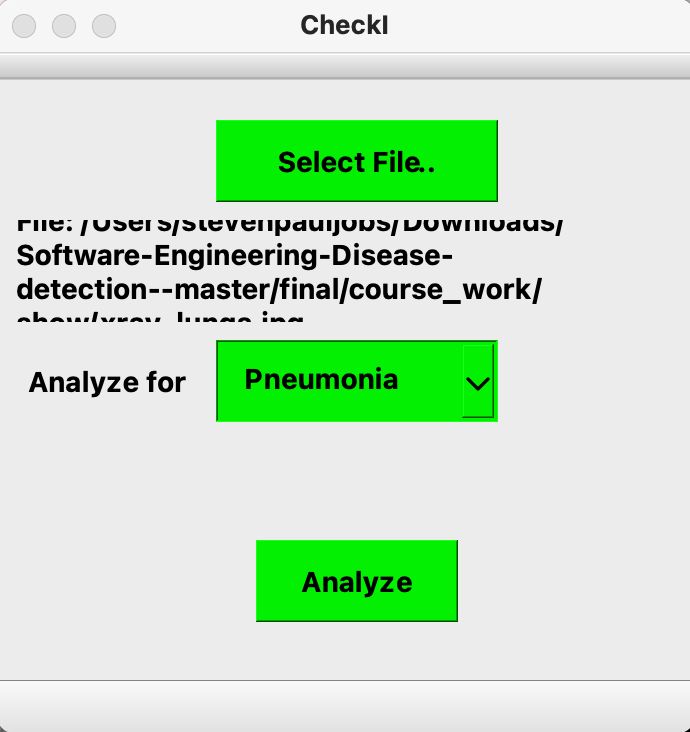


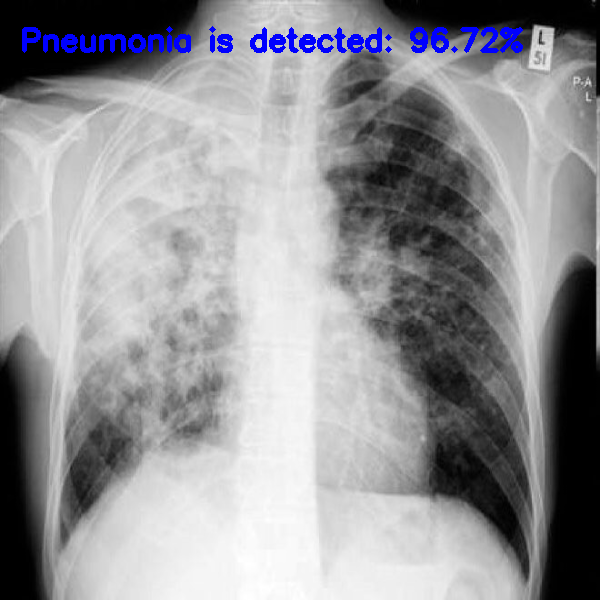
# **Работа программы**

Чтобы использовать эту нейронную сеть, необходимо изображение рентгеновского снимка легких или гистологии в формате png / jpeg. Используя графический интерфейс, мы можем загрузить свое изображение, и результат работы программы появится в новом окне.









**Приложение**

Листинг net.py на языке python

|  |
| --- |
| from keras.models import Sequential from keras.layers.normalization import BatchNormalization from keras.layers.convolutional import SeparableConv2D from keras.layers.convolutional import MaxPooling2D from keras.layers.core import Activation from keras.layers.core import Flatten from keras.layers.core import Dropout from keras.layers.core import Dense from keras import backend as K  class net:  @staticmethod  def build(width, height, depth, classes):  model = Sequential()  inputShape = (height, width, depth)  chanDim = -1    if K.image\_data\_format() == "channels\_first":  inputShape = (depth, height, width)  chanDim = 1   model.add(SeparableConv2D(32, (3, 3), padding="same",  input\_shape=inputShape))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(MaxPooling2D(pool\_size=(2, 2)))  model.add(Dropout(0.25))   model.add(SeparableConv2D(64, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(SeparableConv2D(64, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(MaxPooling2D(pool\_size=(2, 2)))  model.add(Dropout(0.25))   model.add(SeparableConv2D(128, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(SeparableConv2D(128, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(SeparableConv2D(128, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(MaxPooling2D(pool\_size=(2, 2)))  model.add(Dropout(0.25))   model.add(Flatten())  model.add(Dense(256))  model.add(Activation("relu"))  model.add(BatchNormalization())  model.add(Dropout(0.5))   model.add(Dense(classes))  model.add(Activation("softmax"))   return model |

Листинг train\_model.py на языке python

|  |
| --- |
| import matplotlib matplotlib.use("Agg")  from keras.preprocessing.image import ImageDataGenerator from keras.optimizers import Adagrad from keras.utils import np\_utils from sklearn.metrics import classification\_report from sklearn.metrics import confusion\_matrix import net from configuration import config from imutils import paths import matplotlib.pyplot as plt import numpy as np import argparse import os   ap = argparse.ArgumentParser() ap.add\_argument("-p", "--plot", type=str, default="plot.png",  help="path to output loss/accuracy plot") args = vars(ap.parse\_args())   NUM\_EPOCHS = 20 INIT\_LR = 1e-2 BS = 64  trainPaths = list(paths.list\_images(config.TRAIN\_PATH)) totalTrain = len(trainPaths) totalVal = len(list(paths.list\_images(config.VAL\_PATH))) totalTest = len(list(paths.list\_images(config.TEST\_PATH)))   trainLabels = [int(p.split(os.path.sep)[-2]) for p in trainPaths] trainLabels = np\_utils.to\_categorical(trainLabels) classTotals = trainLabels.sum(axis=0) classWeight = classTotals.max() / classTotals  trainAug = ImageDataGenerator(  rescale=1 / 255.0,  rotation\_range=20,  zoom\_range=0.05,  width\_shift\_range=0.1,  height\_shift\_range=0.1,  shear\_range=0.05,  horizontal\_flip=True,  vertical\_flip=True,  fill\_mode="nearest")  valAug = ImageDataGenerator(rescale=1 / 255.0)  trainGen = trainAug.flow\_from\_directory(  config.TRAIN\_PATH,  class\_mode="categorical",  target\_size=(48, 48),  color\_mode="rgb",  shuffle=True,  batch\_size=BS)  valGen = valAug.flow\_from\_directory(  config.VAL\_PATH,  class\_mode="categorical",  target\_size=(48, 48),  color\_mode="rgb",  shuffle=False,  batch\_size=BS)  testGen = valAug.flow\_from\_directory(  config.TEST\_PATH,  class\_mode="categorical",  target\_size=(48, 48),  color\_mode="rgb",  shuffle=False,  batch\_size=BS)  model = CancerNet.build(width=48, height=48, depth=3,  classes=2) opt = Adagrad(lr=INIT\_LR, decay=INIT\_LR / NUM\_EPOCHS) model.compile(loss="binary\_crossentropy", optimizer=opt,  metrics=["accuracy"])  H = model.fit\_generator(  trainGen,  steps\_per\_epoch=totalTrain // BS,  validation\_data=valGen,  validation\_steps=totalVal // BS,  class\_weight=classWeight,  epochs=NUM\_EPOCHS)   print("[INFO] evaluating network...") testGen.reset() predIdxs = model.predict\_generator(testGen,  steps=(totalTest // BS) + 1)  predIdxs = np.argmax(predIdxs, axis=1)  print(classification\_report(testGen.classes, predIdxs,  target\_names=testGen.class\_indices.keys()))   cm = confusion\_matrix(testGen.classes, predIdxs) total = sum(sum(cm)) acc = (cm[0, 0] + cm[1, 1]) / total sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1]) specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])  print(cm) print("acc: {:.4f}".format(acc)) print("sensitivity: {:.4f}".format(sensitivity)) print("specificity: {:.4f}".format(specificity)) model.save('cancer.model')  N = NUM\_EPOCHS plt.style.use("ggplot") plt.figure() plt.plot(np.arange(0, N), H.history["loss"], label="train\_loss") plt.plot(np.arange(0, N), H.history["val\_loss"], label="val\_loss") plt.plot(np.arange(0, N), H.history["accuracy"], label="train\_acc") plt.plot(np.arange(0, N), H.history["val\_accuracy"], label="val\_acc") plt.title("Training Loss and Accuracy on Dataset") plt.xlabel("Epoch #") plt.ylabel("Loss/Accuracy") plt.legend(loc="lower left") plt.savefig(args["plot"]) |

Листинг predict\_pneumonia.py на языке python

|  |
| --- |
| from keras.preprocessing.image import img\_to\_array  from keras.models import load\_model  from imutils import build\_montages  from imutils import paths  import numpy as np  import argparse  import random  import cv2  def predict\_image(imagePaths):  print("[INFO] loading pre-trained network...")  model = load\_model("pneumonia.model")  results = []  orig = cv2.imread(imagePaths)    image = cv2.cvtColor(orig, cv2.COLOR\_BGR2RGB)  image = cv2.resize(image, (48, 48))  image = image.astype("float") / 255.0  image = img\_to\_array(image)  image = np.expand\_dims(image, axis=0)    pred = model.predict(image)    i = pred.argmax(axis=1)[0]      label = "Pneumonia is detected" if i == 1 else "Pneumonia is not detected"  color = (0, 0, 255) if i == 0 else (255, 0, 0)  text = "{}: {:.2f}%".format(label, pred[0][i]\*100)  orig = cv2.resize(orig, (600, 600))  cv2.putText(orig, text, (20, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1,  color, 3)  results.append(orig)  cv2.imshow("Results", orig)  cv2.waitKey(0) |

Листинг predict\_cancer.py на языке python

|  |
| --- |
| from keras.preprocessing.image import img\_to\_array  from keras.models import load\_model  from imutils import build\_montages  from imutils import paths  import numpy as np  import argparse  import random  import cv2  def predict\_image(imagePaths):  print("[INFO] loading pre-trained network...")  model = load\_model("cancer.model")    results = []  orig = cv2.imread(imagePaths)  image = cv2.cvtColor(orig, cv2.COLOR\_BGR2RGB)  image = cv2.resize(image, (48, 48))  image = image.astype("float") / 255.0  image = img\_to\_array(image)  image = np.expand\_dims(image, axis=0)  pred = model.predict(image)    i = pred.argmax(axis=1)[0]      label = "Cancer is detected" if i == 0 else "Cancer is not detected"  color = (0, 0, 255) if i == 0 else (255, 0, 0)  text = "{}: {:.2f}%".format(label, pred[0][i]\*100)  orig = cv2.resize(orig, (600, 600))  cv2.putText(orig, text, (20, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1,  color, 3)  results.append(orig)  cv2.imshow("Results", orig)  cv2.waitKey(0) |

Листинг app.py на языке python

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | import sys | | import os # sys нужен для передачи argv в QApplication | | from PyQt5 import QtWidgets, QtGui | | from PyQt5.QtWidgets import (QMainWindow, QTextEdit, QAction, QFileDialog, QApplication) | |  | |  | |  | | import MainWindow as design | |  | |  | | import predict\_cancer as pc | | import predict\_pneumonia as pp | |  | | globFilename = "NULL" | |  | | class ExampleApp(QtWidgets.QMainWindow, design.Ui\_MainWindow): | | def \_\_init\_\_(self): | | super().\_\_init\_\_() | | self.filename = "NULL" | | self.setupUi(self) | | self.select\_button.clicked.connect(self.getfiles) | | self.analyze\_button.clicked.connect(self.execscript) | |  | |  | |  | | def getfiles(self): | |  | | globFilename, \_ = QtWidgets.QFileDialog.getOpenFileName(self, 'Single File') | | self.filename = globFilename | | self.path\_label.setText("File: " + self.filename) | | print("Name = " + self.filename) | |  | | def execscript(self): | | currText = self.comboBox.currentText() | | currText.replace(" ", "") | | print(currText) | | if currText == "Pneumonia": | | pp.predict\_image(self.filename) | |  | | else: | | pc.predict\_image(self.filename) | |  | |  | |  | |  | | def main(): | | app = QtWidgets.QApplication(sys.argv) QApplication | | window = ExampleApp() | | window.show() | | app.exec\_() | |  | |  | |  | |  | |  | | if \_\_name\_\_ == '\_\_main\_\_': | | main() | |

Листинг Net.py на языке python

|  |
| --- |
| from keras.models import Sequential  from keras.layers.normalization import BatchNormalization  from keras.layers.convolutional import SeparableConv2D  from keras.layers.convolutional import MaxPooling2D  from keras.layers.core import Activation  from keras.layers.core import Flatten  from keras.layers.core import Dropout  from keras.layers.core import Dense  from keras import backend as K  class Net:  @staticmethod  def build(width, height, depth, classes):    model = Sequential()  inputShape = (height, width, depth)  chanDim = -1    if K.image\_data\_format() == "channels\_first":  inputShape = (depth, height, width)  chanDim = 1    model.add(SeparableConv2D(32, (3, 3), padding="same",  input\_shape=inputShape))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(MaxPooling2D(pool\_size=(2, 2)))  model.add(Dropout(0.25))  model.add(SeparableConv2D(64, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(SeparableConv2D(64, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(MaxPooling2D(pool\_size=(2, 2)))  model.add(Dropout(0.25))    model.add(SeparableConv2D(128, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(SeparableConv2D(128, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(SeparableConv2D(128, (3, 3), padding="same"))  model.add(Activation("relu"))  model.add(BatchNormalization(axis=chanDim))  model.add(MaxPooling2D(pool\_size=(2, 2)))  model.add(Dropout(0.25))  model.add(Flatten())  model.add(Dense(256))  model.add(Activation("relu"))  model.add(BatchNormalization())  model.add(Dropout(0.5))  model.add(Dense(classes))  model.add(Activation("softmax"))  return model |

Листинг MainWindow.py на языке python

|  |
| --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_MainWindow(object):  def setupUi(self, MainWindow):  MainWindow.setObjectName("MainWindow")  MainWindow.resize(350, 340)  MainWindow.setMinimumSize(QtCore.QSize(300, 300))  MainWindow.setMaximumSize(QtCore.QSize(512, 512))  MainWindow.setStyleSheet("QPushButton, QComboBox {\n"  " background-color: #03F003;\n"  " color: black;\n"  " \n"  " \n"  "}\n"  "\n"  "menubar, toolbar {\n"  " height: 1px;\n"  "}\n"  "\n"  "QWidget {\n"  " color: #white\n"  " background: #011001;\n"  " font: bold 14px;\n"  "}")  self.centralwidget = QtWidgets.QWidget(MainWindow)  self.centralwidget.setObjectName("centralwidget")  self.select\_button = QtWidgets.QPushButton(self.centralwidget)  self.select\_button.setGeometry(QtCore.QRect(110, 20, 141, 41))  self.select\_button.setObjectName("select\_button")  self.analyze\_button = QtWidgets.QPushButton(self.centralwidget)  self.analyze\_button.setGeometry(QtCore.QRect(130, 230, 101, 41))  self.analyze\_button.setObjectName("analyze\_button")  self.comboBox = QtWidgets.QComboBox(self.centralwidget)  self.comboBox.setGeometry(QtCore.QRect(110, 130, 141, 41))  self.comboBox.setObjectName("comboBox")  self.comboBox.addItem("")  self.comboBox.addItem("")  self.label = QtWidgets.QLabel(self.centralwidget)  self.label.setGeometry(QtCore.QRect(10, 130, 91, 41))  self.label.setAlignment(QtCore.Qt.AlignCenter)  self.label.setObjectName("label")  self.path\_label = QtWidgets.QLabel(self.centralwidget)  self.path\_label.setGeometry(QtCore.QRect(10, 70, 291, 51))  self.path\_label.setWordWrap(True)  self.path\_label.setObjectName("path\_label")  MainWindow.setCentralWidget(self.centralwidget)  self.menubar = QtWidgets.QMenuBar(MainWindow)  self.menubar.setGeometry(QtCore.QRect(0, 0, 350, 25))  self.menubar.setObjectName("menubar")  MainWindow.setMenuBar(self.menubar)  self.statusbar = QtWidgets.QStatusBar(MainWindow)  self.statusbar.setObjectName("statusbar")  MainWindow.setStatusBar(self.statusbar)  self.toolBar = QtWidgets.QToolBar(MainWindow)  self.toolBar.setObjectName("toolBar")  MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)  self.retranslateUi(MainWindow)  QtCore.QMetaObject.connectSlotsByName(MainWindow)  def retranslateUi(self, MainWindow):  \_translate = QtCore.QCoreApplication.translate  MainWindow.setWindowTitle(\_translate("MainWindow", "CheckI"))  self.select\_button.setText(\_translate("MainWindow", "Select File.."))  self.analyze\_button.setText(\_translate("MainWindow", "Analyze"))  self.comboBox.setItemText(0, \_translate("MainWindow", "Breasts Cancer"))  self.comboBox.setItemText(1, \_translate("MainWindow", "Pneumonia"))  self.label.setText(\_translate("MainWindow", "Analyze for"))  self.path\_label.setText(\_translate("MainWindow", "File: "))  self.toolBar.setWindowTitle(\_translate("MainWindow", "toolBar")) |