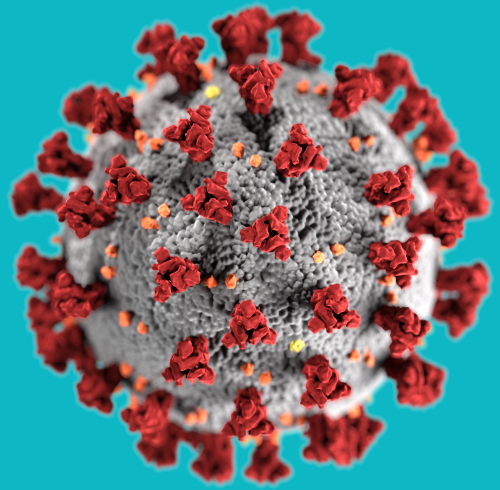
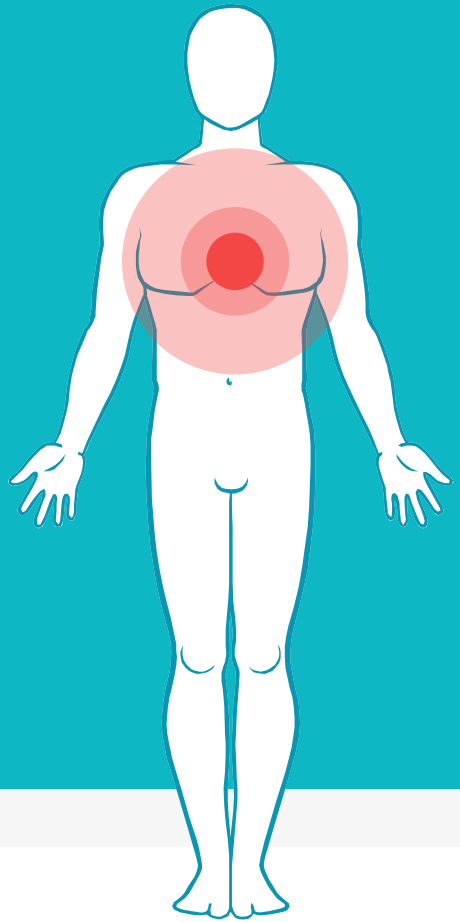


COVID-19 Detection Using Chest X-Rays



Presented by Pratik Kumar Basu (6219699)

Introduction

COVID-19

- Severe Respiratory Syndrome
- Identified first in 2019 in Wuhan, China.
- Since then it has spread and has become a global pandemic.
- Common symptoms include fever, cough and shortness of breath.

Goal

So what is the goal of this project?

- Create a CNN based deep learning model in python using *TensorFlow*, *Keras* and *OpenCV* that classifies into Covid-19 positive or Covid-19 negative based on chest x-ray images.

Project Structure

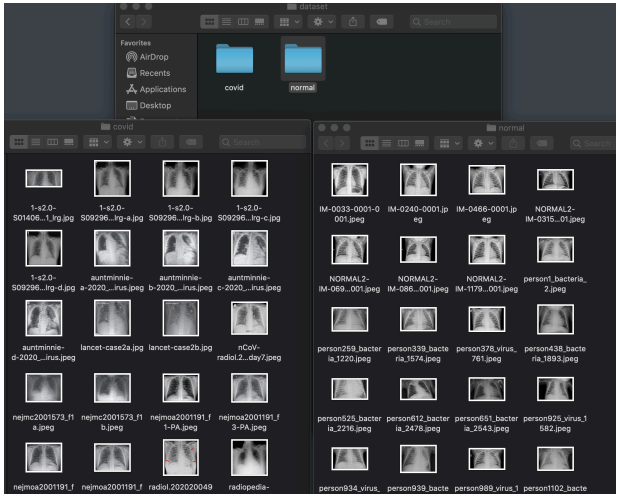
Structure of the project

- Load Dataset.
- Read & Preprocess the dataset.
- Build a CNN model and its layers to detect Covid-19 in X-ray images.
- Train the model and save its accuracy.
- Plot the accuracy and loss in as an image.

Dataset

About the dataset

- Curated by *Dr. Joseph Cohen*, PhD fellow at University of Montreal.
- Two folders
  - **Covid**: Contains 25 chest X-ray images of COVID-19 positive patients.
  - **Normal**: Contains 25 chest X-ray images of COVID-19 negative patients.



Program Code – Command Line Argument Handling

```
# Argument parser to parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
    help="path to input dataset")
ap.add_argument("-p", "--plot", type=str, default="plot.png",
    help="path to output loss/accuracy plot")
ap.add_argument("-m", "--model", type=str, default="covid19.model",
    help="path to output loss/accuracy plot")
args = vars(ap.parse_args())

# initialize the initial learning rate, number of epochs to train for, and batch size
INIT_LR = 1e-3
EPOCHS = 25
BS = 8
```

Program Code – Loading and Processing Data

```
37  # grab the list of images in our dataset directory, then initialize the list of data (i.e., images) and class images
38  print("Starting Program: Loading Images...")
39  imagePaths = list(paths.list_images(args["dataset"]))
40  data = []
41  labels = []
42
43
44  # loop over the image paths
45  for imagePath in imagePaths:
46      # extract the class label from the filename
47      label = imagePath.split(os.path.sep)[-2]
48
49      # load the image, swap color channels, and resize it to be a fixed
50      # 224x224 pixels while ignoring aspect ratio
51      image = cv2.imread(imagePath)
52      image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
53      image = cv2.resize(image, (224, 224))
54
55      # update the data and labels lists, respectively
56      data.append(image)
57      labels.append(label)
58
59
60  # convert the data and labels to NumPy arrays while scaling the pixel
61  # intensities to the range [0, 255]
62  data = np.array(data) / 255.0
63  labels = np.array(labels)
64
```

Covid / Normal

BGR in OpenCV to RGB ordering

Resize

Update labels likewise

numpy array conversion of data & labels

Program Code – One Hot Encoding to Split Data Into Test & Train

```
65  # perform one-hot encoding on the labels
66  lb = LabelBinarizer()
67  labels = lb.fit_transform(labels)
68  labels = to_categorical(labels)
69
70  # partition the data into training and testing splits using 80% of
71  # the data for training and the remaining 20% for testing
72  (trainX, testX, trainY, testY) = train_test_split(data, labels,
73      test_size=0.20, stratify=labels, random_state=42)
74
75  # initialize the training data augmentation object
76  trainAug = ImageDataGenerator(
77      rotation_range=15,
78      fill_mode="nearest")
```

one hot encoding binary classification [0.1.][0.1.][0.1.].....[1.0.][1.0.][1.0.] since + or -

Split dataset to train & test 20% test & 80% train

data augmentation : pre-processing technique to improve model's classification capacity

Program Code – Building the Model

```
80  # load the VGG16 network, ensuring the head FC layer sets are left off
81  baseModel = VGG16(weights="imagenet", include_top=False,
82      input_tensor=Input(shape=(224, 224, 3)))
83
84  # construct the head of the model that will be placed on top of the the base model
85  headModel = baseModel.output
86  headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
87  headModel = Flatten(name="flatten")(headModel)
88  headModel = Dense(64, activation="relu")(headModel)
89  headModel = Dropout(0.5)(headModel)
90  headModel = Dense(2, activation="softmax")(headModel)
91
92  # place the head FC model on top of the base model (this will become the actual model we will train)
93  model = Model(inputs=baseModel.input, outputs=headModel)
94
95  # loop over all layers in the base model and freeze them so they will *not* be updated during the first training process
96  for layer in baseModel.layers:
97      layer.trainable = False
```

Instantiate VGG16 model – CNN architecture used in Imagenet competition 2014

Construct head layers & append them

Freeze every layer in our model so they will not be updated during the first training

Program Code – Train the CNN

```
 99  # compile our model
100  print("[INFO] compiling model...")
101  opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
102  model.compile(loss="binary_crossentropy", optimizer=opt,
103      metrics=["accuracy"])
104
105  # train the head of the network
106  print("[INFO] training head...")
107  H = model.fit_generator(
108      trainAug.flow(trainX, trainY, batch_size=BS),
109      steps_per_epoch=len(trainX) // BS,
110      validation_data=(testX, testY),
111      validation_steps=len(testX) // BS,
112      epochs=EPOCHS)
113
```

Compile CNN with Adam optimizer using binary_crossentropy

Call Keras's fit_generator method, while passing x-ray data via data augmentation object

Adam optimizer: adaptive learning algorithm

fit_generator: used in large dataset or when data augmentation is applied

Program Code – Evaluating the Model

```
114  # make predictions on the testing set
115  print("[INFO] evaluating network...")
116  predIdxs = model.predict(testX, batch_size=BS)
117
118  # for each image in the testing set we need to find the index of the label with corresponding largest predicted probability
119  predIdxs = np.argmax(predIdxs, axis=1)
120
121  # show a nicely formatted classification report
122  print(classification_report(testY.argmax(axis=1), predIdxs,
123      target_names=lb.classes_))
```

Make predictions on testing set and grab prediction indices

generate & print classification report

Program Code – Compute Confusion Matrix

```
125  # compute the confusion matrix and and use it to derive the raw accuracy, sensitivity, and specificity
126  cm = confusion_matrix(testY.argmax(axis=1), predIdxs)
127  total = sum(sum(cm))
128  acc = (cm[0, 0] + cm[1, 1]) / total
129  sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
130  specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
131
132  # show the confusion matrix, accuracy, sensitivity, and specificity
133  print(cm)
134  print("acc: {:.4f}".format(acc))
135  print("sensitivity: {:.4f}".format(sensitivity))
136  print("specificity: {:.4f}".format(specificity))
```

Program Code – Plot the Model's Accuracy / Loss

```
138   # plot the training loss and accuracy
139   N = EPOCHS
140   plt.style.use("ggplot")
141   plt.figure()
142   plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
143   plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
144   plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
145   plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
146   plt.title("Training Loss and Accuracy on COVID-19 Dataset")
147   plt.xlabel("Epoch #")
148   plt.ylabel("Loss/Accuracy")
149   plt.legend(loc="lower left")
150   plt.savefig(args["plot"])
151
152   # serialize the model to disk
153   print("[INFO] saving COVID-19 detector model...")
154   model.save(args["model"], save_format="h5")
155
```

Program Code - Result

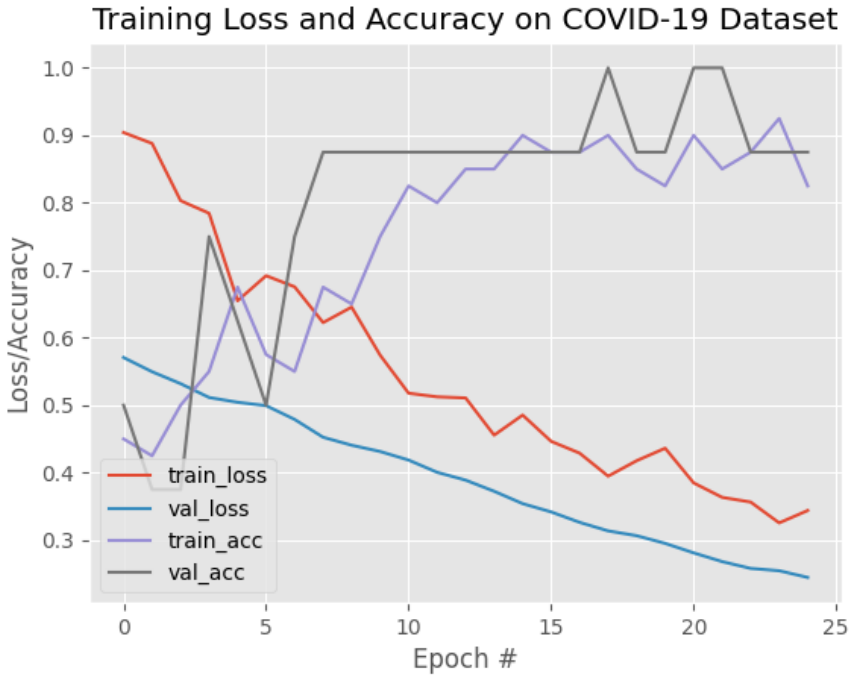
Not under/over fitted

Loss decreases with increase in EPOCH

```
Evaluating network...
              precision    recall  f1-score   support

       covid       0.80      0.80      0.80         5
      normal       0.80      0.80      0.80         5

    accuracy                           0.80        10
   macro avg       0.80      0.80      0.80        10
weighted avg       0.80      0.80      0.80        10

[[4 1]
 [1 4]]
acc: 0.8000
sensitivity: 0.8000
specificity: 0.8000
```

80% accuracy due to :
- maybe there is better model for classification
- Small dataset due to few Covid-19 + images
-

Program Code – Result



Program Code - Testing

```
22  CATEGORIES = ["Covid Positive", "Covid Negative"]
23
24
25  # Testing on unknown data
26  image = cv2.imread('normal.jpg')
27  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
28  image = cv2.resize(image, (224, 224))
29  image = image.reshape((-3 ,224, 224,3))
30
31  model = tf.keras.models.load_model("covid19.model")
32
33
34  # Result to our prediction
35  prediction = model.predict(image)
36  print("\nThe X-ray result is: ")
37  print(CATEGORIES[int(prediction[0][1])])
38  print("\n")
39  #print(prediction)
40
```

Random chest x-ray image of a COVID-19 negative patient was tested, with our model created.

Program Code – Testing Result

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CON

xtaticxeolite@Pratiks-MBP Deep Learning
2020-04-30 10:55:29.786925: I tensorflow
2020-04-30 10:55:29.801924: I tensorflow
Devices:
2020-04-30 10:55:29.801945: I tensorflow

The X-ray result is:
Covid Negative
```

The result came out to be as expected

‘Negative”