Computer Vision Assignment 2

Muhammad Alfi Ramadhan - 21/472839/PA/20345

Create a Convolutional Neural Network (CNN) model to classify X-Ray images by classifying them into two categories: COVID and non-COVID

Google Colab Link

I. Introduction

A convolutional neural network (CNN) is a type of neural network for working with images which extracts features from raw image data and provides learnable parameters to efficiently perform tasks such classification, detection and a variety of other tasks. In computer vision CNN is highly important and widely used, especially for object classification.

II. Problem

We are given a dataset of 156 X-Ray images divided into two classes: positive and negative. In this report, a CNN model is created to classify this dataset of X-Ray images into COVID or non-COVID classes.

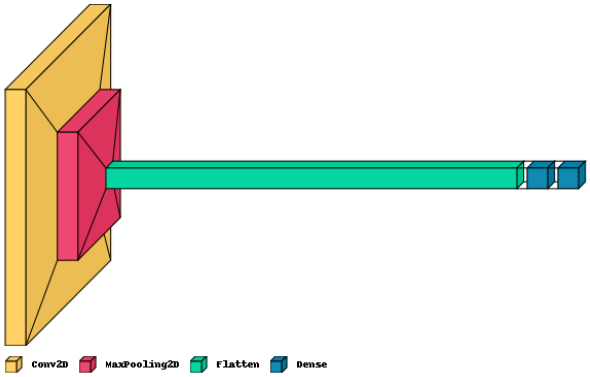
III. Splitting the dataset

The dataset consists of 156 X-Ray images that are divided into two folders: positive and negative which comprise 98 and 58 images respectively. The dataset is split into training and testing sets which are separate from each other with a ratio of 8:2. This results in a training set comprising 78 negative and 46 positive images, summing up to 124 images, and a testing comprising 20 negative and 12 positive images, summing up to 32 images.

IV. Building the CNN

The following code and diagram shows how the CNN model is structured:

```
classifier = Sequential()
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Flatten())
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```



This is a basic CNN classifier, defined as a sequential model, meaning that each layer is added after one another in a sequential manner.

The first layer is a convolution layer with 32 filters of size 3x3 and an input shape of 64x64x3. The filters are applied to the input image and produce feature maps that capture patterns in the image. Rectified Linear Unit acts as the activation function in this layer which increases the non-linearity of the output, which in turn allows the neural network to better learn complex patterns in the input data.

The convolution layer is followed by a MaxPooling layer with a pool size of 2x2. This layer makes the representation smaller and manageable by reducing the spatial dimensions of the resulting feature maps, which is done through taking the largest value in each 2x2 block of the map.

Subsequently, the Flatten layer converts the 2D feature maps produced by the previous layer into a 1D feature vector.

Next is a dense layer with 128 units and ReLU as activation function. It is a fully connected layer to the previous one which learns to combine the features extracted from previous layers in order to make a prediction.

Finally, the CNN is concluded with a dense layer containing 1 unit and Sigmoid activation function. The sigmoid activation function is used here because it exists between 0-1, which is suitable for our CNN where our prediction is in the form of probability in

order to classify the image into either positive or negative class.

The optimization algorithm for training the CNN is the Adam optimizer. Binary cross-entropy is used as the loss function, since it is suited to binary classification problems, such as this one. The accuracy metric is used to evaluate the performance of the model in order to measure the percentage images that are classified correctly (true positives and true negatives).

Overall, the structure of this CNN model is designed to be relatively simple, comprising a small number of layers and parameters as a preventative measure to avoid overfitting on the small dataset of 156 images being used.

V. Data Augmentation

Data augmentation is a technique used to artificially add variety to existing data which will allow the model to train a larger number of network parameters without overfitting or scarcity of the data.

```
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('/content/drive/MyDr
ive/UNI/SEMESTER 4/comvis/COVID-19/Training',
target_size = (64,64), batch_size=31,
class_mode='binary')
test_set =
test_datagen.flow_from_directory('/content/drive/MyDri
ve/UNI/SEMESTER 4/comvis/COVID-19/Testing',
target_size=(64,64), batch_size=16, class_mode =
'binary')
```

The ImageDataGenerator class is used to implement data augmentation to our data. We use the rescale parameter to divide the pixel values of the input images by 255 to yield values within an interval of [0,1]. This normalized range of values will come in favor for the model during the training process. Other than the rescaling, we also use shear_range, zoom_range and horizontal flips in an effort to increase the variety of our dataset. For both training and testing, the images will be resized into 64x64 pixels before being fed into the CNN. For the training process, the model will take in batches of 31 images at a time and for the testing process, the model will take in batches of 16 images at a time.

VI. Training

We train our CNN model by using the fit function below:

```
classifier.fit(training_set, steps_per_epoch = 4,
epochs = 100, validation_data = test_set,
validation_steps = 2, callbacks =
[checkpoint_callback])
```

The steps_per_epoch parameter denotes the number of batches that will be processed by the model in each epoch during the training process. Since our training set consists of 124 images, and our batch size is 31, we can obtain the number of batches that will be processed each epoch by dividing the total number of images by the batch size (124/31) which in this case results in 4 batches.

Our CNN will go through 100 epochs in order to improve its accuracy over time. The ModelCheckpoint callback is used to save and update the best val_accuracy achieved by the model after each epoch, which helps prevent overfitting and save the best model.

```
Epoch 100/100
4/4 [==============================] - ETA: 0s - loss: 0.4876 - accuracy: 0.7419
Epoch 100: val_accuracy did not improve from 0.75000
4/4 [==============================] - 12s 4s/step - loss: 0.4876 - accuracy: 0.7419 -
<keras.callbacks.History at 0x7f74b2e09c90>
```

After 100 epochs, the highest val_accuracy that the model achieved was 0.75.

VII. Testing

After the training process is complete, it is appropriate to evaluate the model on the testing set.

```
classifier.evaluate(test_set)
```

```
2/2 [==============================] - 14s 12s/step - loss: 0.6861 - accuracy: 0.6562
[0.6860675811767578, 0.65625]
```

The model yields a loss of 0.6861 and an accuracy of 0.6562 after evaluation.

Now we test the model using the testing set to predict whether the X-Ray images it is being fed belong to the COVID or Non-COVID class.

A. Testing the model on Positive Images

Each image in the 'positive' class of the testing set is fed into the CNN model for testing. Below are the results:

```
covid-19-pneumonia-8.jpg
0.0
negative

covid-19-caso-70-1-PA.jpg
1.0
positive

covid-19-pneumonia-40.jpg
1.0
positive

covid-19-pneumonia-49-day4.jpg
1.0
positive

covid-19-rapidly-progressive-acute-respiratory-distress-syndrome-ards-day-1.j
9.7046446e-05
negative

covid-19-pneumonia-34.png
4.0087702e-06
negative

covid-19-pneumonia-15-L.jpg
1.0
positive

covid-19-pneumonia-22-day1-l.png
1.0
positive

covid-19-pneumonia-53.jpg
1.3557449e-24
negative

covid-19-pneumonia-rapidly-progressive-admission.jpg
1.0
positive

covid-19-pneumonia-43-day0.jpeg
1.0
positive

covid-19-pneumonia-14-L.png
1.0
positive
```

After testing the model using images belonging to the 'positive' class of the testing set, it managed to correctly predict 8/12 X-Ray images as COVID.

B. Testing the model on Negative Images

Each image in the 'negative' class of the testing set is fed into the CNN model for testing. Below are the results:

```
7E335538-2F86-424E-A0AB-6397783A38D0.jpeg
1.365866e-11
negative

2C10A413-AABE-4807-8CCE-6A2025594067.jpeg
0.0022676727
negative

B59DD164-51D5-40DF-A926-6A42DD52EBE8.jpeg
0.3297448
negative

7D2CF6CE-F529-4470-8356-D33FFAF98600.jpeg
2.1131723e-18
negative

kjr-21-e24-g003-1-a.jpg
0.49480408
negative

53EC07C9-5CC6-4BE4-9B6F-D7B0D72AAA7E.jpeg
2.3972872e-12
negative

nejmoa2001191_f3-PA.jpeg
3.1712516e-05
negative

446B2CB6-B572-40AB-B01F-1910CA07086A.jpeg
1.2351362e-12
negative

nejmoa2001191_f1-PA.jpeg
3.1582073e-09
negative

kjr-21-e24-g002-1-a.jpg
0.63549113
positive

39EE8E69-5801-48DE-B6E3-BE7D1BCF3092.jpeg
1.446824e-06
negative

7C69C012-7479-493F-8722-ABC29C60A2DD.jpeg
5.3669073e-08
negative

figure1-5e7c1b8d98c29ab001275405-98.jpeg
0.27719507
negative
```

```
aspiration-pneumonia-5-day27.jpg
0.9997481
positive

legionella-pneumonia-2.jpg
0.00055214996
negative

353889E0-A1E8-4F9E-A0B8-F24F36BCFBFB.jpeg
0.22214429
negative

2B8649B2-00C4-4233-85D5-1CE240CF233B.jpeg
0.34913865
negative

figure1-5e7c1b8d98c29ab001275405-98-later.jpeg
1.905336e-20
negative

F63AB6CE-1968-4154-A70F-913AF154F53D.jpeg
1.0113161e-09
negative

acute-respiratory-distress-syndrome-ards.jpg
1.5180256e-10
negative
```

After testing the model using the images belonging to the 'negative' class of the testing set, it managed to correctly predict 18/20 X-Ray images as Non-COVID.