GROUP 25

IMAGE CLASSIFICATION PROJECT

1ST DECEMBER 2023

Group Members

JACKLINE

MOURINE MWANGI

LESLEY WANJIKU

Overview

Image Classification with Deep Learning

Based on the dataset provided, we are to build a deep neural network kind of model that speaks to image classification

Process

1.Business Understanding

2.Data Understanding

3.Data Preprocessing

4. Model Selection

5. Model Fine Tuning

Image Augmentation

Transfer Learning

Next Steps

6. Conclusion

Business Understanding

For this project, we are required to build a model that trains on a large dataset for classification purposes.

Our task :

Identify whether or not pediatric patients have pneumonia using the X-ray images

Data Understanding

The dataset comes from Kermany, its divided into 3 categories:

- Test
- Train
- Value

This will in turn enable us to perform the modellig and fine tuning that will aid with the classification discussion.

For this process,

- We create our paths to our data, then double check the distribution of 'normal' vs. 'pneumonia' xrays in the train, test, and val directories to ensure all our files loaded properly

```python
# Define paths to train, test and val sets
train_folder = '/content/chest_xray/train'
train_pneu = '/content/chest_xray/train/PNEUMONIA'
train_norm = '/content/chest_xray/train/NORMAL'

test_folder = '/content/chest_xray/test'
test_pneu = '/content/chest_xray/test/PNEUMONIA'
test_norm = '/content/chest_xray/test/NORMAL'

val_folder = '/content/chest_xray/val'
val_pneu = '/content/chest_xray/val/PNEUMONIA'
val_norm = '/content/chest_xray/val/NORMAL'

# Print distribution
print('Train PNEUMONIA= ', len(os.listdir(train_pneu)))
print('Train NORMAL= ', len(os.listdir(train_norm)))

print('Test PNEUMONIA= ', len(os.listdir(test_pneu)))
print('Test NORMAL= ', len(os.listdir(test_norm)))

print('Val PNEUMONIA= ', len(os.listdir(val_pneu)))
print('Val NORMAL= ', len(os.listdir(val_norm)))
```

Data Preprocessing

This process of preprocessing data involves:

- Read in our data in batches
- Normalizing pixel output
- Scaling images
- Setting to the binary class mode for our target variable, y

```python
# Batch feed and join images/labels to create data sets
X_train_aug, y_train_aug = next(train_generator)
X_test, y_test = next(test_generator)
X_val, y_val = next(val_generator)
```

```python
# Explore dataset again to ensure all files are there and in desired shape
print ("Train_images shape: " + str(X_train_aug.shape))
print ("Train_labels shape: " + str(y_train_aug.shape))
print ("Test_images shape: " + str(X_test.shape))
print ("Test_labels shape: " + str(y_test.shape))
print ("Val_images shape: " + str(X_val.shape))
print ("Val_labels shape: " + str(y_val.shape))
```

```
Train_images shape: (5216, 224, 224, 3)
Train_labels shape: (5216,)
Test_images shape: (624, 224, 224, 3)
Test_labels shape: (624,)
Val_images shape: (16, 224, 224, 3)
Val_labels shape: (16,)
```

```
# Create Image Data Generator for Train Set to augment normal cases
image_gen = ImageDataGenerator(
        rotation_range = 45,
        rescale = 1./255)

# Create Image Data Generator for Test/Validation Set
test_data_gen = ImageDataGenerator(rescale = 1./255)
val_data_gen = ImageDataGenerator(rescale=1./255)

train_generator = image_gen.flow_from_directory(
        train_folder,
        target_size=(224, 224), batch_size=5216,
        class_mode='binary',
        shuffle=False,
        seed=18
        )

# get all the data in the "test" directory (624 images), rescale and reshape them
test_generator = test_data_gen.flow_from_directory(
        test_folder,
        target_size=(224, 224), batch_size = 624,
        class_mode='binary',
```

Model Selection

- For our Model Classification we use the technique: CNN (Convutional Neural Network)
- A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology
- The CNN is used to detect and classify objects in an image

Model Fine Tuning

In deep learning, fine-tuning is an approach to transfer learning in which the weights of a pre-trained model are trained on new data.

It involves taking a pre-trained model, which has been trained on a large dataset for a general task such as image recognition

Image Augumentation

Transfer Learning

There's several paths we could take to improve our model's precision and recall.

We'll try two, both address our class imbalance.

This techniques include:

- Image Augumentation
- Transfer Learning

```
[ ]  model = models.Sequential()
     model.add(Conv2D(32 , (3,3) , strides = 1, padding = 'same',
                      activation = 'relu', input_shape = (224, 224, 3)))
     model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

     model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
     model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

     model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
     model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

     model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
     model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

     model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
     model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

     model.add(Flatten())
     model.add(Dense(128, activation = 'relu'))
     model.add(Dense(64, activation = 'relu'))
     model.add(Dense(64, activation = 'relu'))
```

```
callbacks = [EarlyStopping(monitor='val_loss', patience=4),
             ModelCheckpoint(filepath='best_model.h5', monitor='val_loss', save_best_only=True)]
```

```
weights = class_weight.compute_class_weight(class_weight='balanced',
                                            classes= np.unique(y_train_aug),
                                            y= y_train_aug)
weights= dict(zip(np.unique(y_train_aug), weights))
weights
```

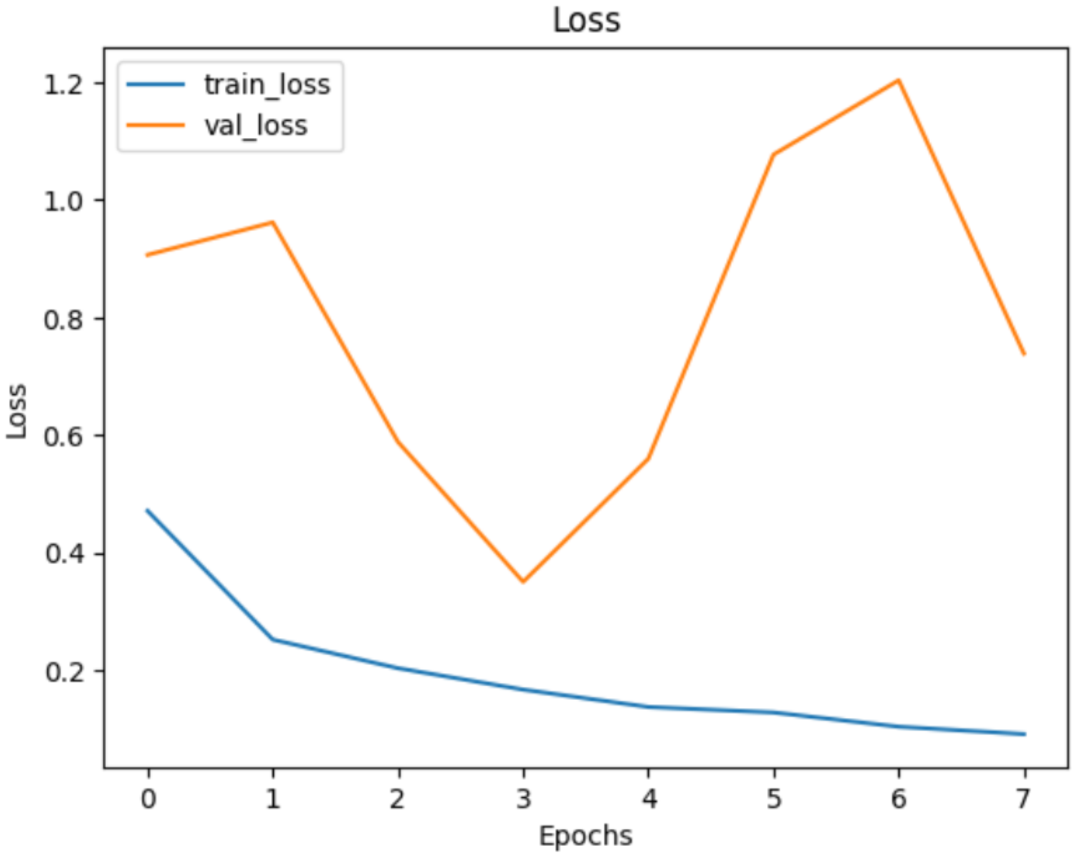
```
{0.0: 1.9448173005219984, 1.0: 0.6730322580645162}
```

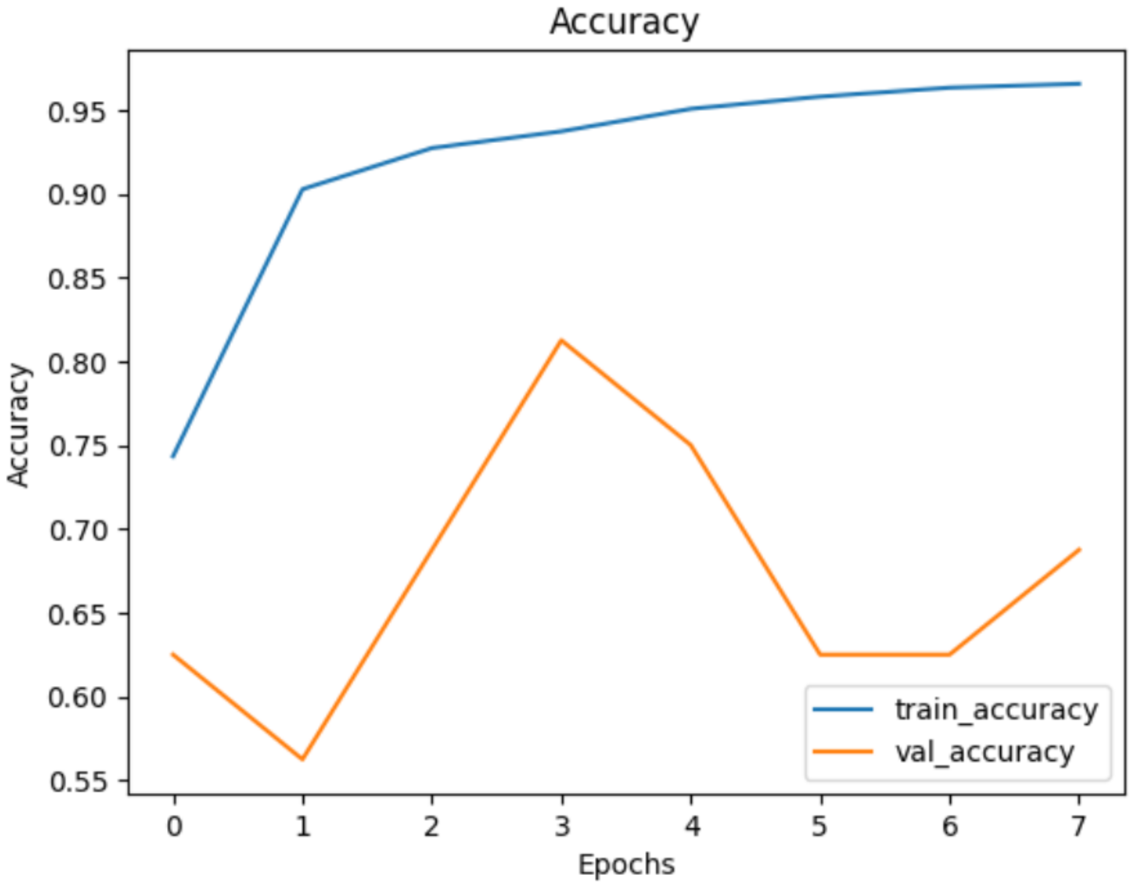
```
history = model.fit(X_train_aug,
                    y_train_aug,
                    epochs=20,
                    batch_size=20,
                    callbacks= [callbacks],
                    class_weight= weights,
                    validation_data=(X_val, y_val))
```

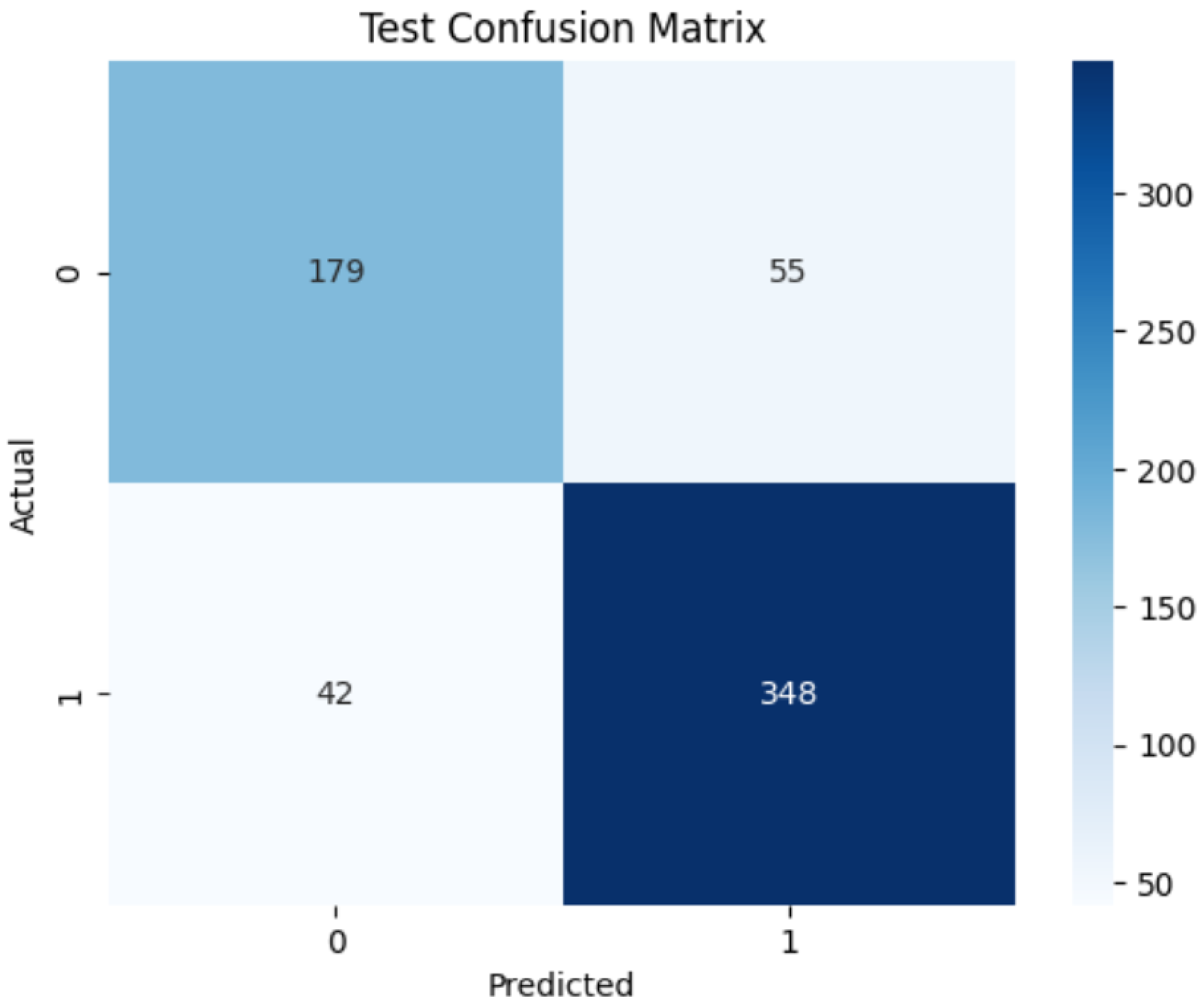
- Visualizing the CNN Results

```
[ ]  def visualize_CNN_results(results, y_test=y_test):
       """
       input results of model fitting.
       output loss and accuracy curves, and confusion matrix
       """
       # Plot Train and Val Loss
       history = results.history
       plt.figure()
       plt.plot(history['loss'])
       plt.plot(history['val_loss'])
       plt.legend(['train_loss', 'val_loss'])
       plt.title('Loss')
       plt.xlabel('Epochs')
       plt.ylabel('Loss')
       plt.show()

       # Plot Train and Val Accuracy
       plt.figure()
       plt.plot(history['acc'])
       plt.plot(history['val_acc'])
       plt.legend(['train_accuracy', 'val_accuracy'])
       plt.title('Accuracy')
```


Loss
train_loss
val_loss
Loss
Epochs


Accuracy
train_accuracy
val_accuracy
Accuracy
Epochs


Test Confusion Matrix
179
55
42
348
Actual
0
1
Predicted
0
1
300
250
200
150
100
50

Model Selection

```
model = models.Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1, padding = 'same',
                 activation = 'relu', input_shape = (224, 224, 3)))
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))

model.add(Dropout(0.2))
```

Conclusion

Based on our cm numbers, it looks like our PNet with image augmentation and weights was our best model so far, with 179 TP (actual normal) and 55 FP (predicted normal), and 42 FN(predicted pneumonia), and 348 TN(actual pneumonia). ?

Next Steps

- In order to employ our model at scale with confidence, we'd want to improve our models detection of pneumonia by continuing to fine tune our cutoff points and image augmentation.

THANK YOU