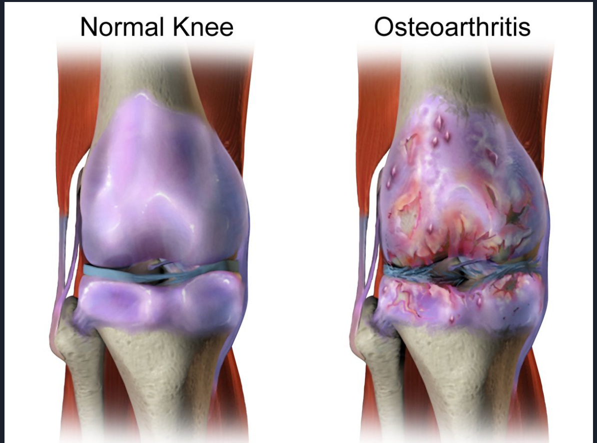
Identifying the Severity of Knee Osteoarthritis Utilizing Machine Learning

By Aayush Balaji

Introduction to Experiment

Knee osteoarthritis is the gradual, progressive process in the loss of articular cartilage, and is a common occurrence among the elderly. Machine learning is a great concept to allude to in terms of experimental design. This is why I decided to experiment and tinker with the hyperparameters of popular image recognition CNN(Convolutional Neural Network) algorithms, to detect potential sources of osteoarthritis in the x-ray image of a knee.



Hypothesis

I hypothesize that training and experimenting with convolutional neural network(CNN)) algorithms that are popularly known for image recognition and image processing such as MobilenetV2, ResNet50, and VGG 16 will result in an efficient machine learning- based model that can be utilized for identifying the severity of knee osteoarthritis in an x-ray.

This is because these CNN algorithms are known for success in image recognition when corresponded with the best hyperparameters.

Variables

Independent Variable:

Hyperparameters(Epochs, Learning rate, Batch size)*

Dependent Variable:

Accuracy of model with all dataset images

*Hyperparameters are sets of guidelines provided to the algorithm that control it’s actions throughout the dataset analysis of it.

(E.g Epochs = Amount of times the CNN algorithm analyzes the dataset content)

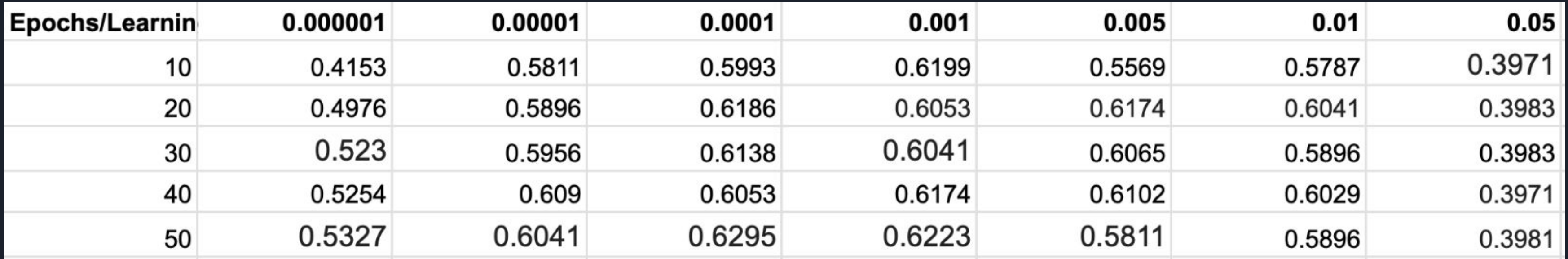
Materials

Computer consisting of the following resources:

- ➢ TensorFlow - Machine learning platform
- ➢ Google Colab - Python coding platform
- ➢ Google Suite for documentation and data storage
- ➢ Mendely.com - Data source
- ➢ Matplotlib.pyplot - For generating graphs

Procedure

1. Download the data from the following dataset, and upload it on google drive: Knee Osteoarthritis Dataset
2. Split the data into train, test, and validation. The split should be 80% train, 10% test, and 10% validation. Modify the data into non-Kellgren-Lawrence grading scale format* only if necessary. An elaboration on this is given at the end of the procedure.
3. As the advent of the model training, start experimenting with the MobilnetV2 algorithm against the train and validation datasets, constantly modifying the hyperparameters of epochs, base learning rate, and batch size. When each google colab run is completed, record the experimental results on a google spreadsheet. One example of this corresponds to the image below.
4. Run the code for a wide hyperparameter spectrum of your choosing. It should be in the following range:
   - ➢ Epochs: 10-50
   - ➢ Learning rate: 0.000001 - 0.05
   - ➢ Batch size: 32
   - ➢ You could experiment with a larger variety of hyperparameters as well, for more potential best models. This process will require an extensive duration of time.



Epochs/Learnin	0.000001	0.00001	0.0001	0.001	0.005	0.01	0.05
10	0.4153	0.5811	0.5993	0.6199	0.5569	0.5787	0.3971
20	0.4976	0.5896	0.6186	0.6053	0.6174	0.6041	0.3983
30	0.523	0.5956	0.6138	0.6041	0.6065	0.5896	0.3983
40	0.5254	0.609	0.6053	0.6174	0.6102	0.6029	0.3971
50	0.5327	0.6041	0.6295	0.6223	0.5811	0.5896	0.3981

Procedure(Continued)

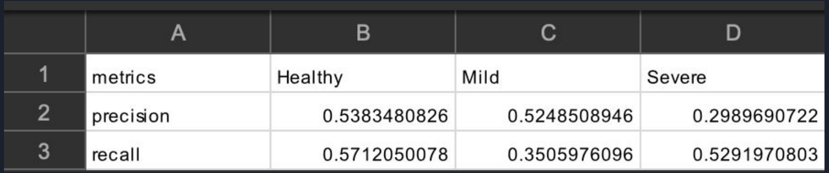
5. With the results, generate a multi-line plot graph utilizing matplotlib. This helps with the further evaluation as well as depiction of the data. Examples are provided in the next set of slides.
6. Along with your graph, generate various forms of your data, displaying the accuracy. Examples include a confusion matrix, classification report, and precision/recall score. All of this should be executed with the test dataset, and a saved best model in your google drive for valid, viable results.
7. Switch the algorithm to ResNet50, and repeat steps #3 - 6
8. Switch the algorithm to VGG 16, and repeat steps #3 - 6
9. Compare and contrast the best models and their corresponding algorithms, then determine the best model among the algorithms.
10. Convert the best algorithm's model (.h5) into an .onnx file.
11. Lastly, deploy this model as a REST API so that anyone in the world with an internet connection can access it. I used AWS lambda and AWS Gateway to achieve this.

*The Kellgren-Lawrence grading scale evaluates osteophytes and joint space narrowing to assign a score between 0 (no ROA) to 4 (severe ROA) (1, 2, 3). In other words, it evaluates a specified skeletal image and provides it with a designated grade, a number ranging from 0 to 4, in order of severity. 0 would signify no sign of osteoarthritis, while 4 would indicate and represent a severe source of osteoarthritis in the x-ray. I originally planned on experimenting with this utilizing machine learning, however the inaccurate results caused me to revert to a generic categorization.

Data

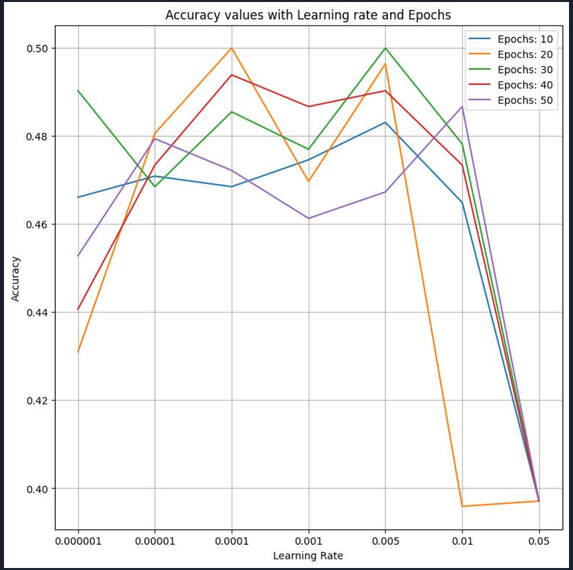
Data - MobileNetV2

Precision/Recall score

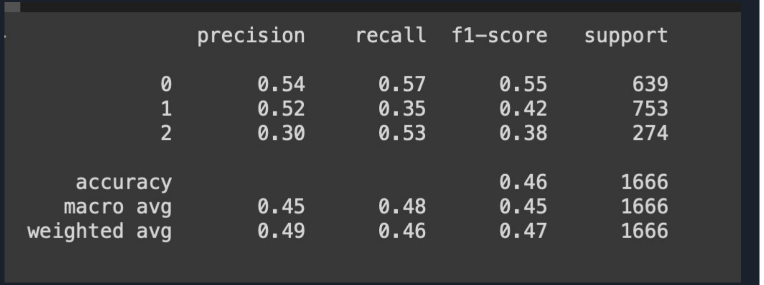


	A	B	C	D
1	metrics	Healthy	Mild	Severe
2	precision	0.5383480826	0.5248508946	0.2989690722
3	recall	0.5712050078	0.3505976096	0.5291970803

Graph



Classification Report

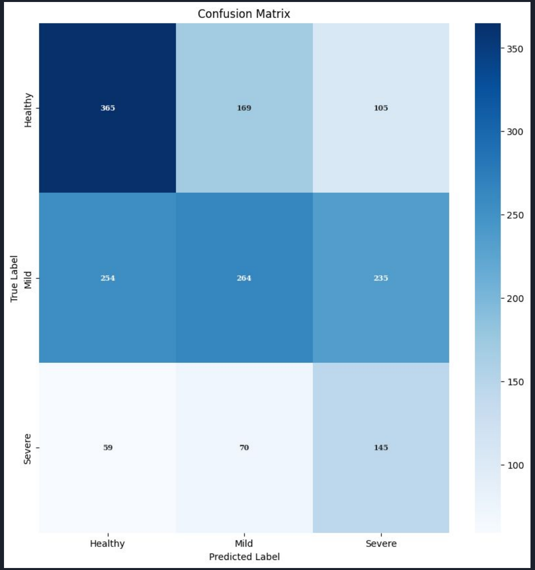


```
              precision    recall  f1-score   support

           0       0.54      0.57      0.55       639
           1       0.52      0.35      0.42       753
           2       0.30      0.53      0.38       274

    accuracy                           0.46      1666
   macro avg       0.45      0.48      0.45      1666
weighted avg       0.49      0.46      0.47      1666
```

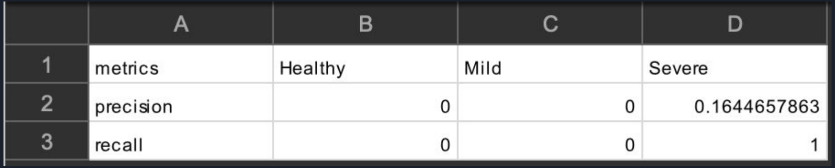
Confusion Matrix



[Link to Model Training Experiments](): Recording hyperparameter accuracy(Google Spreadsheet)

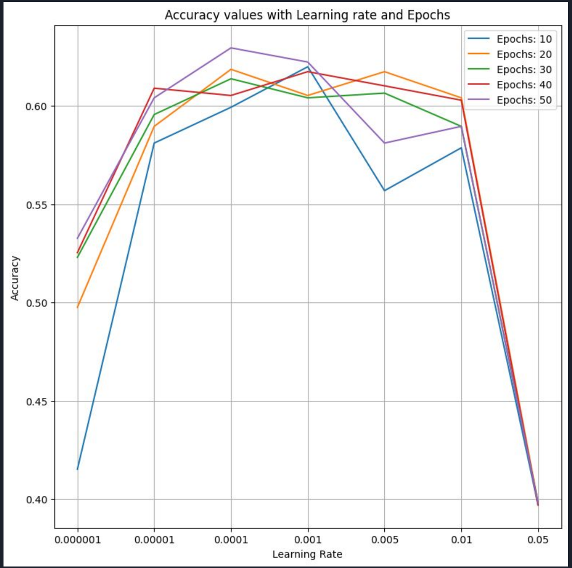
Data - ResNet50

Precision/Recall score

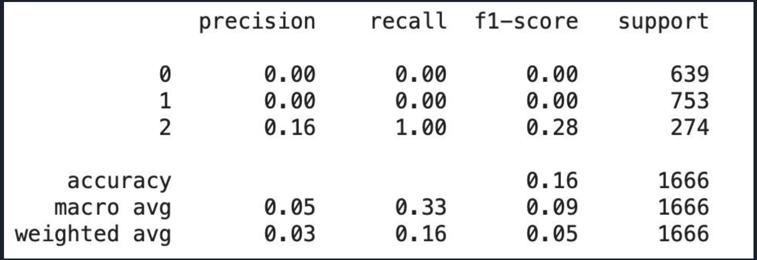


	A	B	C	D
1	metrics	Healthy	Mild	Severe
2	precision	0	0	0.1644657863
3	recall	0	0	1

Graph



Classification Report

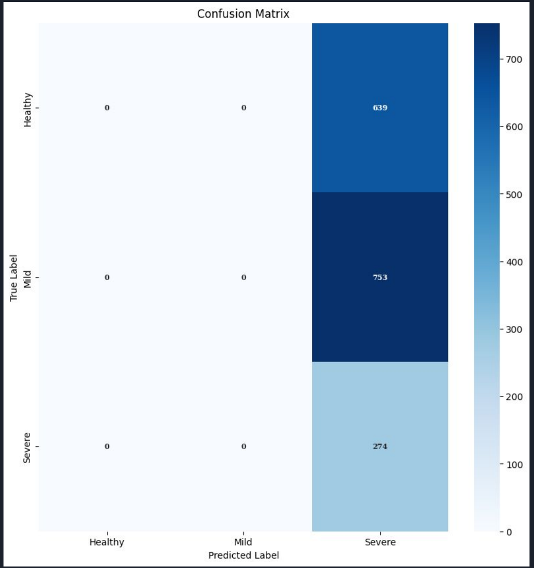


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       639
           1       0.00      0.00      0.00       753
           2       0.16      1.00      0.28       274

    accuracy                           0.16      1666
   macro avg       0.05      0.33      0.09      1666
weighted avg       0.03      0.16      0.05      1666
```

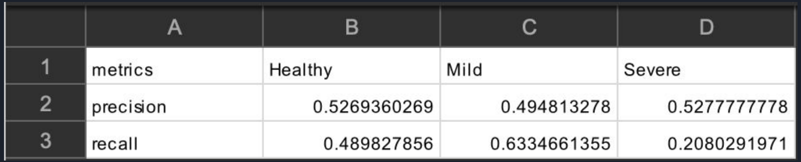
Confusion Matrix



Link to Model Training Experiments:
Recording hyperparameter accuracy(Google Spreadsheet)

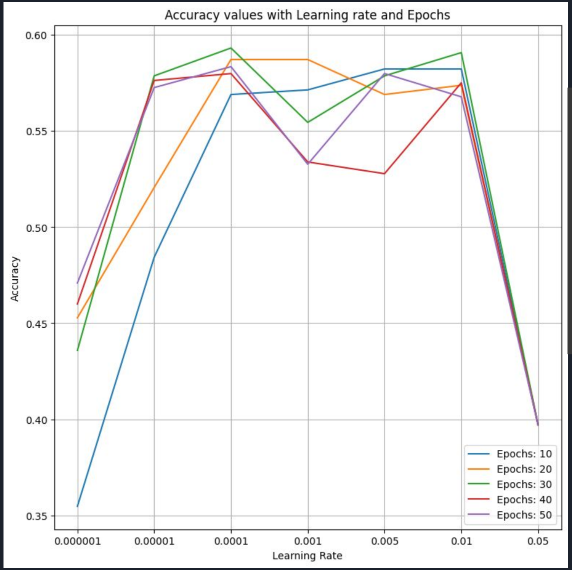
Data - VGG16

Precision/Recall score

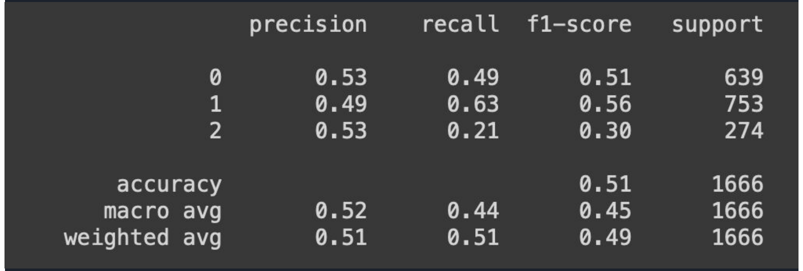


	A	B	C	D
1	metrics	Healthy	Mild	Severe
2	precision	0.5269360269	0.494813278	0.5277777778
3	recall	0.489827856	0.6334661355	0.2080291971

Graph



Classification Report

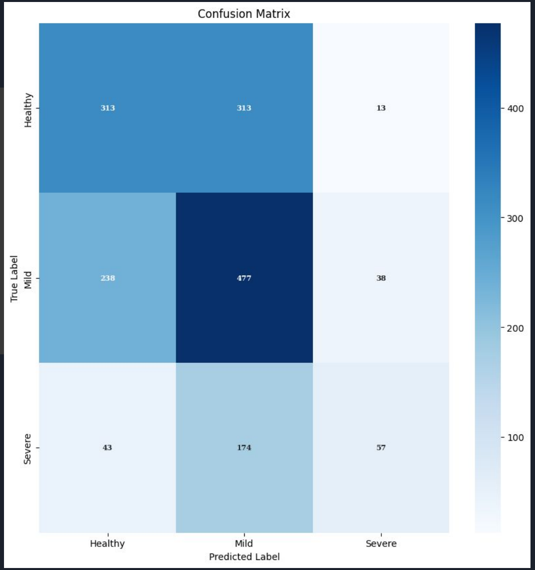


```
              precision    recall  f1-score   support

           0       0.53      0.49      0.51       639
           1       0.49      0.63      0.56       753
           2       0.53      0.21      0.30       274

    accuracy                           0.51      1666
   macro avg       0.52      0.44      0.45      1666
weighted avg       0.51      0.51      0.49      1666
```

Confusion Matrix



Link to Model Training Experiments: Recording hyperparameter accuracy(Google Spreadsheet)

Analysis/Interpretation of Data

The data provided has clearly indicated valid and crucial information. Analyzing the data corresponding to the CNN algorithm of MobilenetV2, the algorithm seemed to have fair predictions in terms of accuracy. It's best hyperparameters were an epochs of 30, and a learning rate of 0.001. Overall, it's evaluation of the test dataset was similar in comparison to the model training results. Secondly, the algorithm of ResNet50 displayed a very high accuracy in comparison to the other models when depicted on the graph in terms of hyperparameters, with a best epochs as 50, best base learning rate as 0.0001, and a total accuracy of 0.6295. Generally, this dataset tends to be more on the complex side, and it is tough to obtain perfection in terms of accuracy. A link to it is given by the beginning of my procedure. Although it is certified by many medical associations to be accurate, there have been many complaints on the data source website of which include accuracy errors. As the ResNet50 model proceeded to the evaluation of the test images, it began to have a sudden, unexpected downfall in the accuracy to the best model. It seemed to be only predicting the category of "Severe"(See ResNet50 confusion matrix in data). Lastly, the algorithm of VGG16 contained results that were similar to the test image evaluation in comparison to the model training, just like MobilenetV2. However, it seems to perform poorly corresponded with the best hyperparameters when faced with a non-severe image, which I have observed is a flowing pattern throughout the experiment. It's best hyperparameters are an epochs of 30, and a base learning rate of 0.0001. Overall, my hypothesis was partially approved because there were still initially unexpected accuracies due to the dataset complexity.

Potential Experimental Errors

The dataset used is fairly complex and there has been many complaints about it on it’s website. Furthermore, it was first providing me with an accuracy below 50% each time I modified the hyperparameters. I drifted to utilizing various techniques such as data augmentation, different algorithms, and eventually making the data more concise and relying on categorizing the images into healthy, mild, and severe, because I was originally using the **Kellgren-Lawrence grading scale** for knee osteoarthritis classification.