Final Year Design Project Report

How to AI if you can't code and don't have GPU

- **Advisor: Dr. Wajahat Hussain**
- **Co-Advisor: Dr. Abid Rafique**

Talha Anwar (307793)_EE-12D

Bachelors of Electrical Engineering

School of Electrical Engineering and Computer Sciences (SEECS)

National University of Sciences and Technology (NUST)

1. Abstract

In today's technological world, machine learning (ML) is very important field used in almost every field of today's world. But it is very difficult for non-technical people to apply machine learning (ML) in their field as it requires coding skills as well as computational resources to implement. This project presents WekaLab, an innovative graphical user interface platform that makes machine learning (ML) accessible to a large number of people by allowing people to create ML models effectively without the need for advanced computational resources or programming expertise. WekaLab uses google computers processing power through Google Colab to simplify the process of building and evaluating machine learning models. WekaLab has three modules as CNN Module, RNN Module and Pre-Trained Module. A comparative analysis between WekaLab and other popular tools like WEKA and Google AutoML shows that the WekaLab significantly reduces the initial configuration and processing time needed for machine learning tasks. The findings demonstrate that WekaLab facilitates machine learning (ML) for non-technical users while simultaneously improving model training efficiency and scalability. This increases the possible uses for machine learning in almost every field of today's world.

2. Introduction:

The Machine Learning requires coding skills as well as computational resources. There are a large number of people like entrepreneurs, healthcare and from education field who need Machine Learning in their field but they don't have coding skills. For people who doesn't have coding skills can use automated Machine Learning tools like Weka, Google AutoML etc. WEKA stands as Waikato Environment for Knowledge Analysis, an open source tool developed by University of Waikato, New Zealand provides GUI to implement Machine Learning. It has a lots of options in it like data preprocessing, classification, clustering, regression etc.

WEKA also has open source deep learning module as well. WEKA is very powerful tool for small datasets as we can easily build model by just uploading dataset, selecting ML/DL algorithm, and selecting hyperparameters. But as the size of dataset increases, it becomes inefficient because it is using personal computers processing power and not everyone has good computational resources to train deep learning models on large datasets. Google AutoML developed by Google allows non-technical people who doesn't have programming skills as well as enough computational resources to implement Machine Learning in their field. AutoML only takes dataset and gives model in return. Google AutoML is easily accessible through Google Cloud and Google charges for its services. Google AutoML is an expensive tool and not everyone can pay for its services because of some financial constraints. To democratize the use of Machine Learning in today's world, we need to find the solution for those who doesn't have coding skills as well as good computational resources.

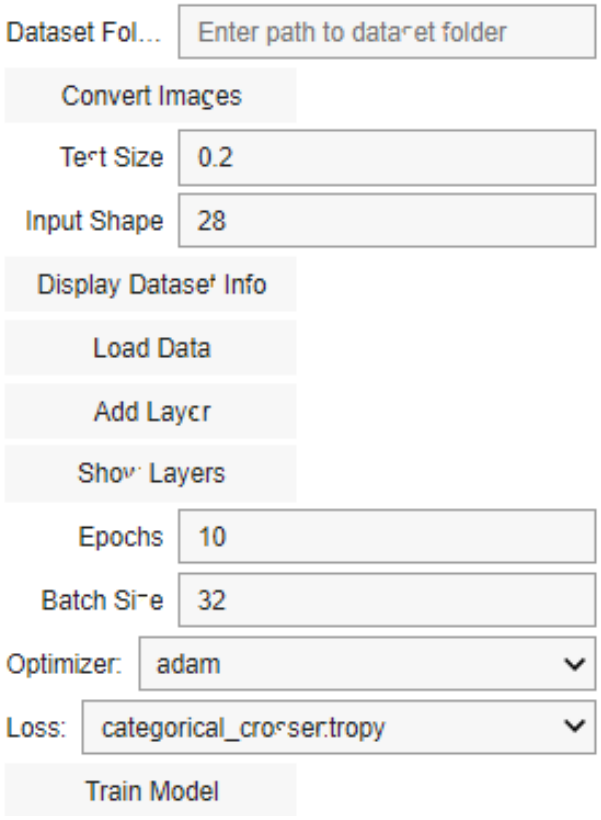
3. Methodology

As we can also access the Google computers processing power through Google Colab so there is a need to develop Weka like GUI on Google Colab. In this way, everyone can easily implement Machine Learning using Google computers processing power without paying for it. So, we developed GUI on google Colab called WekaLab. The WekaLab consists of three modules:

- CNN Module
- RNN Module
- Pre-Trained Module

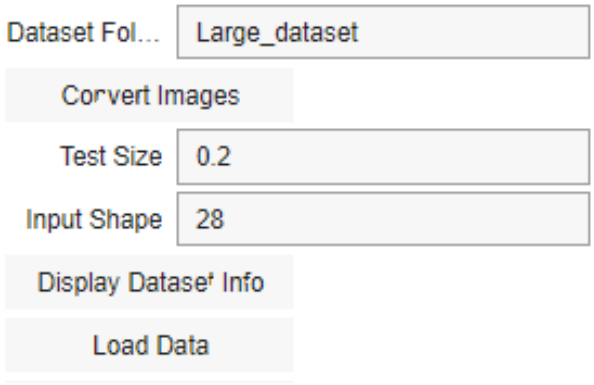
4. Project Description

- CNN Module
  - Run the CNN Module code.
  - Mount the Google Drive.
  - This display of CNN GUI will appear.



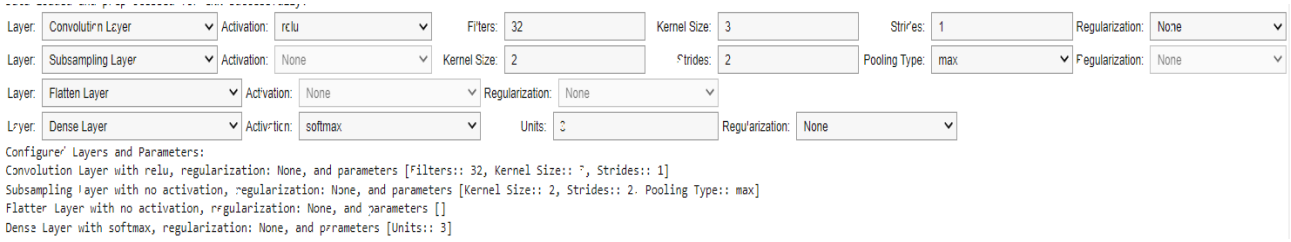
  - Enter Image Dataset folder name

- Click Convert Images button, it will convert images to csv files (images.csv and label.csv)
- Adjust the test Size and Input Shape
- Click Load data button, it will load dataset by splitting dataset into test/train, resizing images, and applying label encoding.
- Click on the Display Dataset Info button to display dataset info.

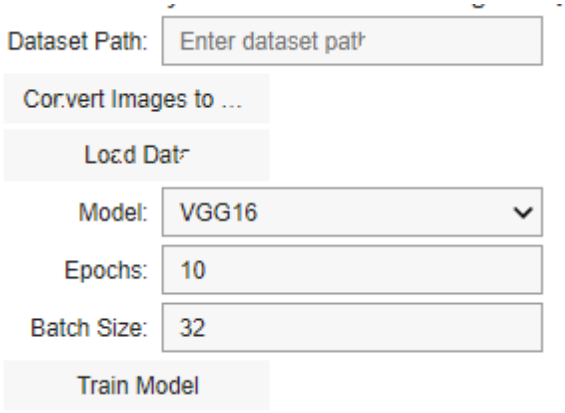



```
Images have already been converted.
Displaying Dataset Information...
Total images per class: {'BENIGN': 350, 'MALIGNANT': 350, 'NORMAL': 350}
Loading Data...
Data loaded and preprocessed for CNN successfully!
```

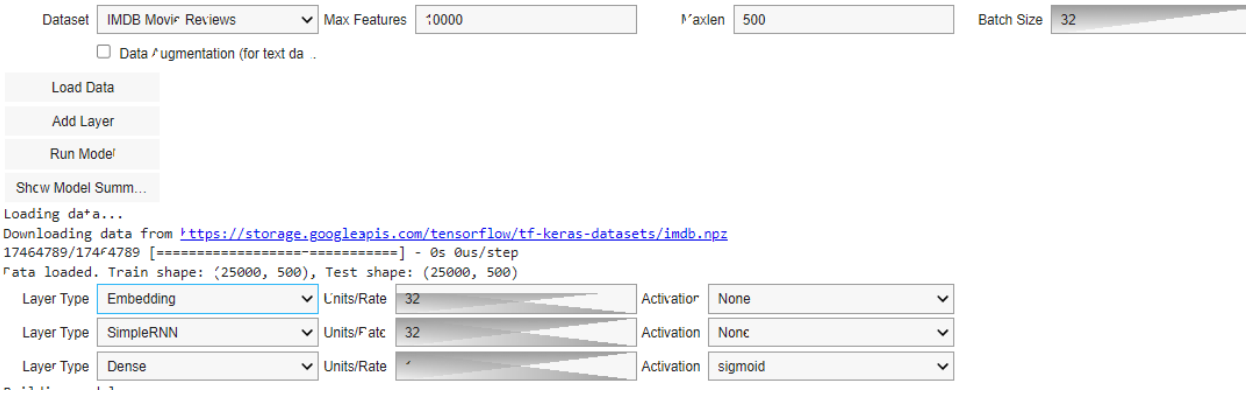
- Click on Add Layer button to add layers to the model.
- These layers can be added:
  - Convolution
  - Subsampling
  - Dense
  - Activation
  - Dropout
  - Batch Normalization
  - Flatten
- We can adjust every hyper parameter of these layers.
- Click Show Layers button to see all the added layers of model
- Adjust the Epoch and batch Size.
- Select the Optimizer and Loss Function
- Click on Train Model button to start training of the model.



- Pre-Trained Module
  - Enter image dataset folder name to access from Google Drive.
  - Click on Convert Images button to convert images into csv files
  - Select Models from drop down menu
    - VGG19, VGG16
    - ResNet50
    - DenseNet
    - Inception
    - EfficientNet
    - and more
  - Adjust Epochs and Batch sizes.
  - Click on the Train Model button to start training the model



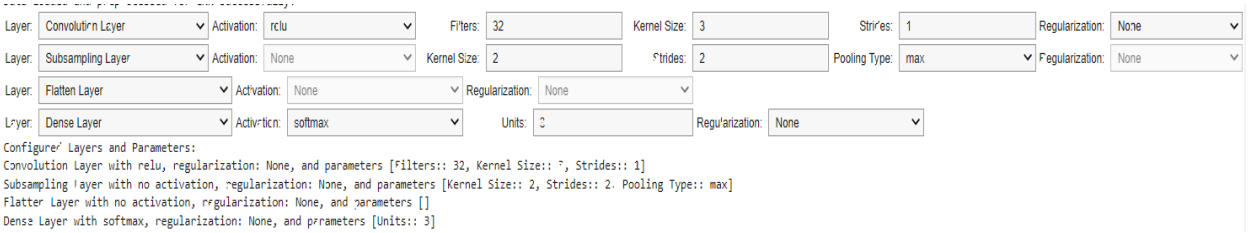
- RNN Module
  - Select dataset from the available datasets dropdown menu.
  - Click on Load Data button to load dataset
  - Click on Add Layer button to add Layers
  - Click on Run Model button to start training
  - Click on Show Model Summary button to see details of RNN model.



5. Results

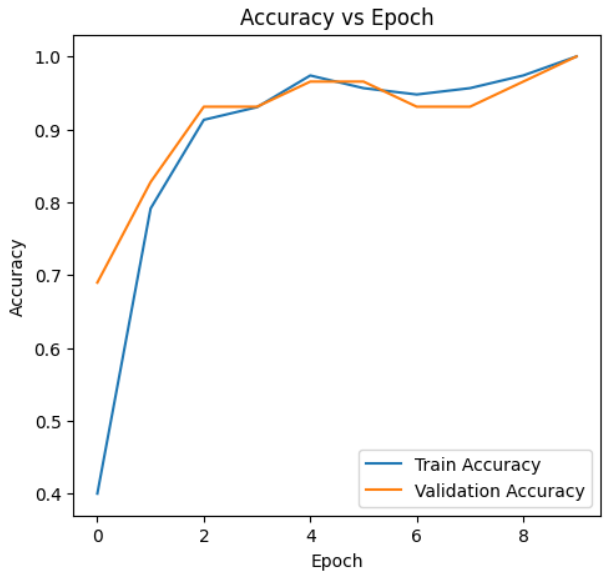
- CNN Module

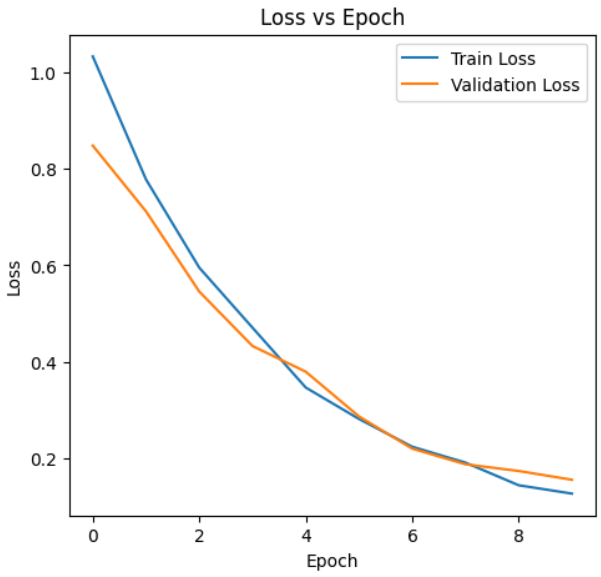
The cancer classification problem has been solved using CNN Module of WekaLab. The CNN model The used consists of four layers as Convolution Layer, Subsampling Layer, Flatten Layer, Dense Layer. Below are the detailed screenshots of the results of this model. We can see that cancer images have been correctly classified using WekaLab.

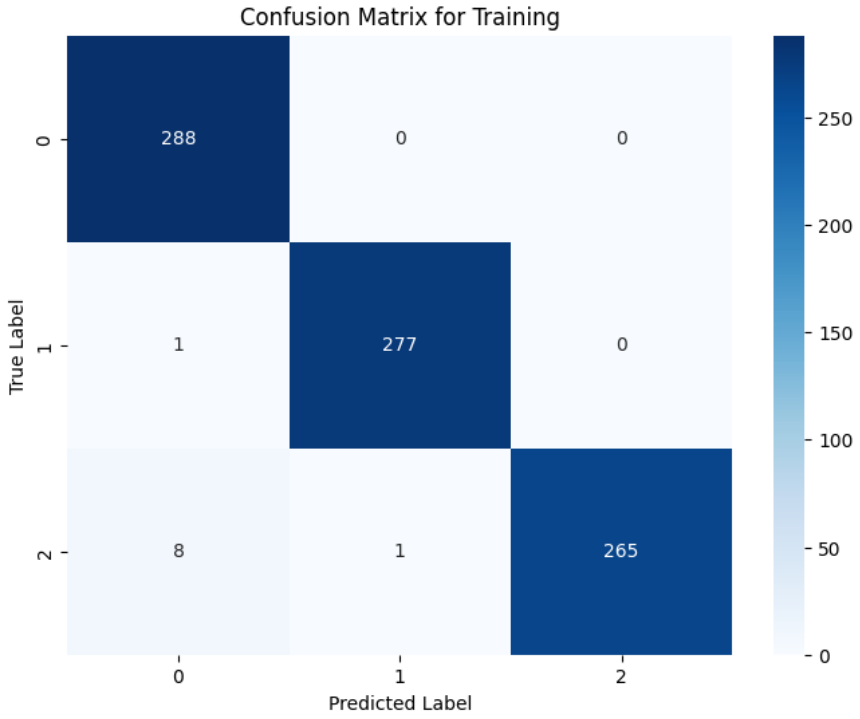



```
Epoch 1/10
27/27 [==============================] - 1s 23ms/step - loss: 0.9522 - accuracy: 0.5357 - val_loss: 0.7906 - val_accuracy: 0.6905
Epoch 2/10
27/27 [==============================] - 0s 16ms/step - loss: 0.7109 - accuracy: 0.7440 - val_loss: 0.6365 - val_accuracy: 0.7429
Epoch 3/10
27/27 [==============================] - 0s 17ms/step - loss: 0.5455 - accuracy: 0.8274 - val_loss: 0.5045 - val_accuracy: 0.8524
Epoch 4/10
27/27 [==============================] - 0s 16ms/step - loss: 0.4118 - accuracy: 0.8798 - val_loss: 0.4135 - val_accuracy: 0.8429
Epoch 5/10
27/27 [==============================] - 0s 15ms/step - loss: 0.3247 - accuracy: 0.9119 - val_loss: 0.3495 - val_accuracy: 0.8619
Epoch 6/10
27/27 [==============================] - 0s 15ms/step - loss: 0.2656 - accuracy: 0.9357 - val_loss: 0.3035 - val_accuracy: 0.8905
Epoch 7/10
27/27 [==============================] - 0s 15ms/step - loss: 0.2073 - accuracy: 0.9571 - val_loss: 0.2868 - val_accuracy: 0.9143
Epoch 8/10
27/27 [==============================] - 0s 16ms/step - loss: 0.1770 - accuracy: 0.9667 - val_loss: 0.2595 - val_accuracy: 0.9143
Epoch 9/10
27/27 [==============================] - 0s 16ms/step - loss: 0.1729 - accuracy: 0.9524 - val_loss: 0.2269 - val_accuracy: 0.9190
Epoch 10/10
27/27 [==============================] - 1s 24ms/step - loss: 0.1316 - accuracy: 0.9750 - val_loss: 0.2092 - val_accuracy: 0.9238
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 5408)              0

 dense_2 (Dense)             (None, 3)                 16227

=================================================================
Total params: 16547 (64.64 KB)
Trainable params: 16547 (64.64 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```





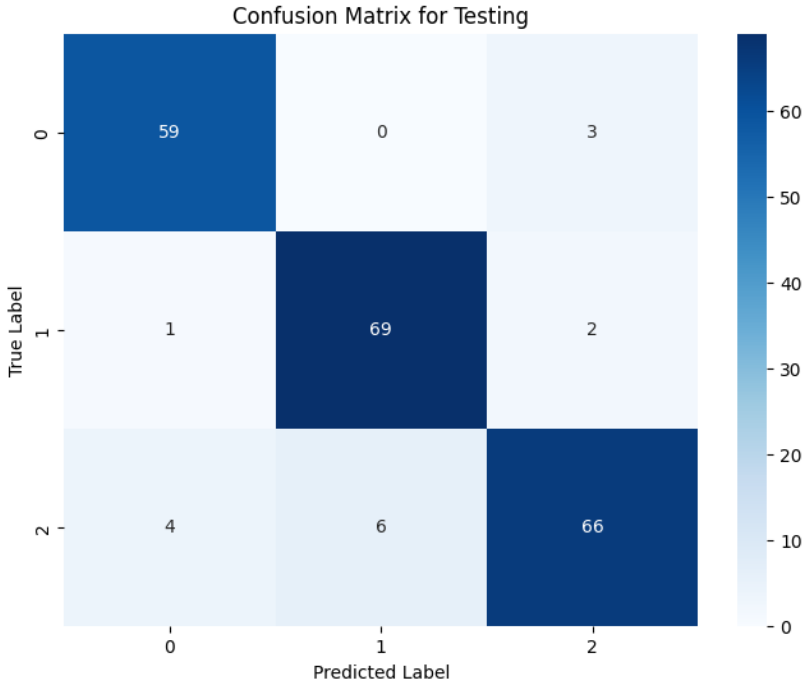


Classification Report for Training Data:

	precision	recall	f1-score	support
0	0.969697	1.000000	0.984615	288.000000
1	0.996403	0.996403	0.996403	278.000000
2	1.000000	0.967153	0.983302	274.000000
accuracy	0.988095	0.988095	0.988095	0.988095
macro avg	0.988700	0.987852	0.988107	840.000000
weighted avg	0.988420	0.988095	0.988088	840.000000

Metrics for Training Data:

	Accuracy	Error Rate
Training	0.988095	0.011905



Classification Report for Testing Data:

	precision	recall	f1-score	support
0	0.921875	0.951613	0.936508	62.00000
1	0.920000	0.958333	0.938776	72.00000
2	0.929577	0.868421	0.897959	76.00000
accuracy	0.923810	0.923810	0.923810	0.92381
macro avg	0.923317	0.926122	0.924414	210.00000
weighted avg	0.924020	0.923810	0.923334	210.00000

Metrics for Testing Data:

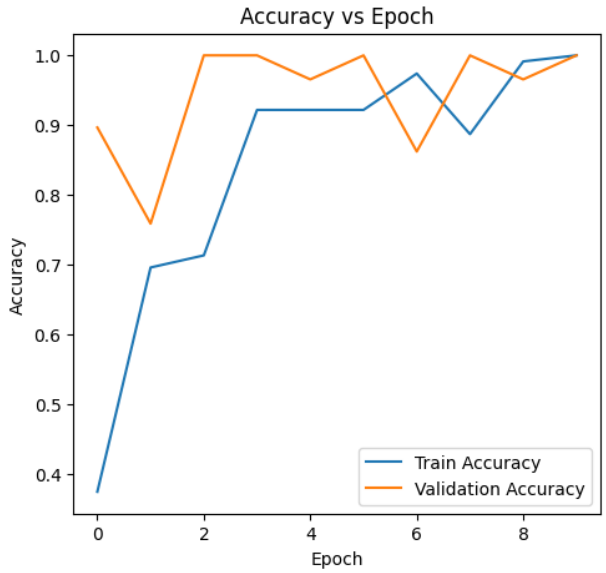
	Accuracy	Error Rate
Testing	0.92381	0.07619

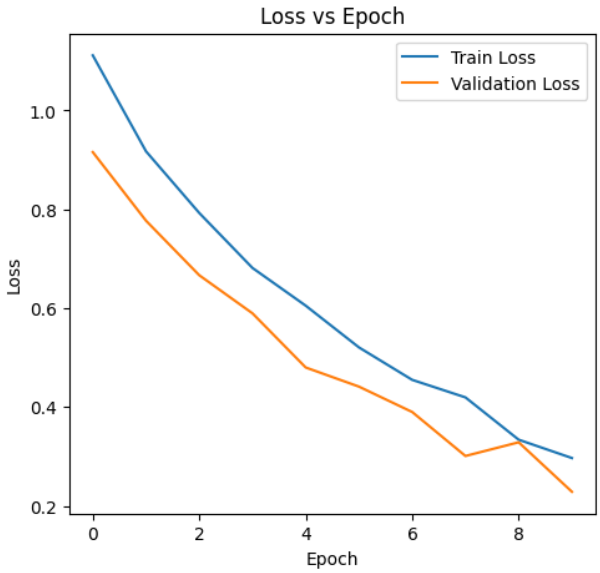
By using WekaLab, this model trained within few seconds but the same model takes around 4 minutes on Weka. So, we can say that WekaLab is clearly time efficient as well as using Google Computers processing power effectively. Similarly, we can solve any kind of image classification problem using WekaLab providing an innovative solution to democratize the use of Machine Learning.

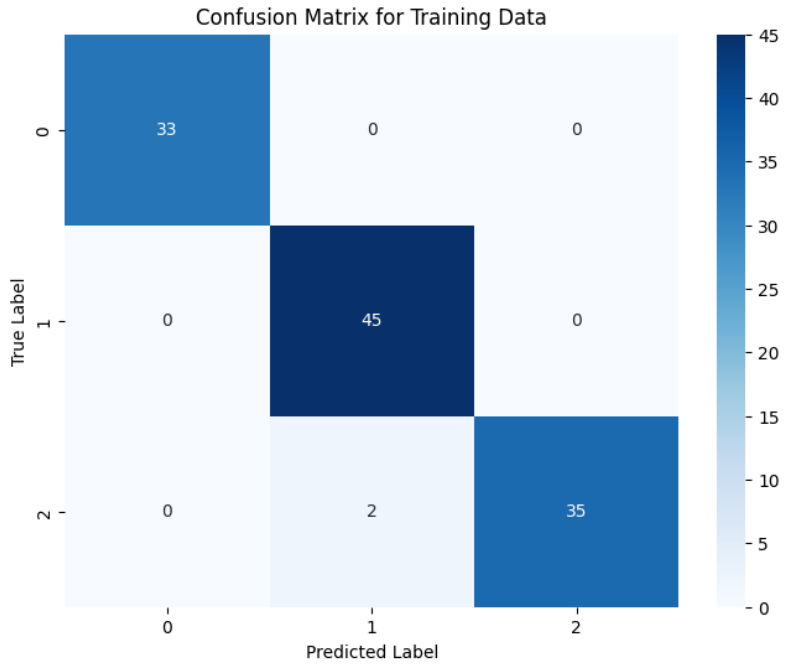
- Pre-Trained Module

As we know that every pre-trained model on Google Colab can load within seconds but weka takes a lot of time to just load pre-trained model like it takes around 30min to load VGG19 model. Therefore, WekaLab Pre-Trained Module has clear advantage over Weka in terms of time efficiency and over Google AutoML in terms of its accuracy and computational resources. Let's see Pre-Trained Module results of cancer image classification as:

```
Epoch 1/10
4/4 [==============================] - 81s 21s/step - loss: 1.1116 - accuracy: 0.3739 - val_loss: 0.9159 - val_accuracy: 0.8967
Epoch 2/10
4/4 [==============================] - 30s 21s/step - loss: 0.9175 - accuracy: 0.6957 - val_loss: 0.7771 - val_accuracy: 0.7586
Epoch 3/10
4/4 [==============================] - 79s 21s/step - loss: 0.7928 - accuracy: 0.7130 - val_loss: 0.6668 - val_accuracy: 1.0000
Epoch 4/10
4/4 [==============================] - 79s 21s/step - loss: 0.6816 - accuracy: 0.9217 - val_loss: 0.5897 - val_accuracy: 1.0000
Epoch 5/10
4/4 [==============================] - 83s 22s/step - loss: 0.6053 - accuracy: 0.9217 - val_loss: 0.4803 - val_accuracy: 0.9655
Epoch 6/10
4/4 [==============================] - 73s 19s/step - loss: 0.5209 - accuracy: 0.9217 - val_loss: 0.4415 - val_accuracy: 1.0000
Epoch 7/10
4/4 [==============================] - 79s 21s/step - loss: 0.4552 - accuracy: 0.9739 - val_loss: 0.3902 - val_accuracy: 0.8621
Epoch 8/10
4/4 [==============================] - 79s 21s/step - loss: 0.4198 - accuracy: 0.8870 - val_loss: 0.3013 - val_accuracy: 1.0000
Epoch 9/10
4/4 [==============================] - 80s 21s/step - loss: 0.3342 - accuracy: 0.9913 - val_loss: 0.2290 - val_accuracy: 0.9655
Epoch 10/10
4/4 [==============================] - 76s 20s/step - loss: 0.2971 - accuracy: 1.0000 - val_loss: 0.2289 - val_accuracy: 1.0000
Model trained successfully!
```





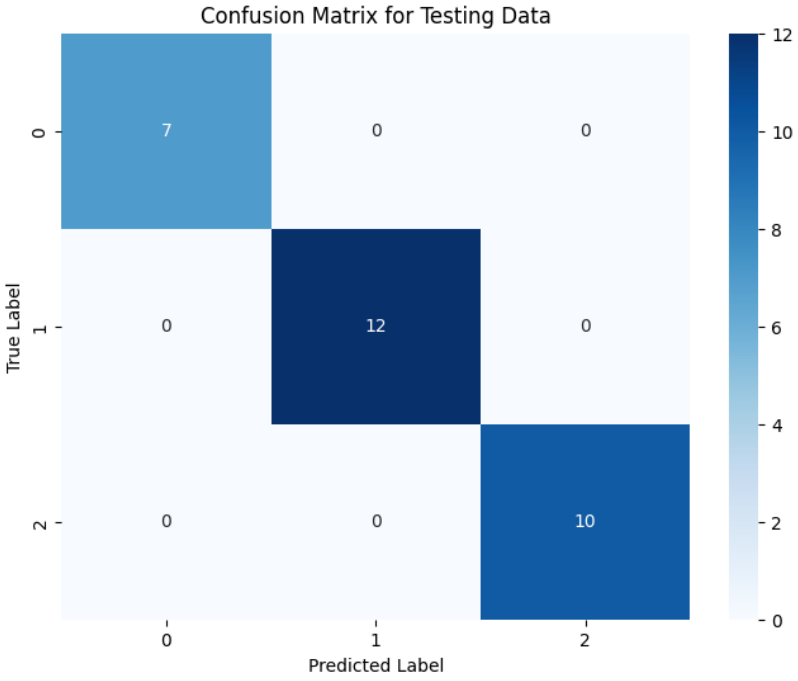


Classification Report for Training Data:

	precision	recall	f1-score	support
0	1.000000	1.000000	1.000000	33.000000
1	0.957447	1.000000	0.978261	45.000000
2	1.000000	0.945946	0.972222	37.000000
accuracy	0.982609	0.982609	0.982609	0.982609
macro avg	0.985816	0.981982	0.983494	115.000000
weighted avg	0.983349	0.982609	0.982556	115.000000

Metrics for Training Data:

	Accuracy	Error Rate
Training	0.982609	0.017391



Classification Report for Testing Data:

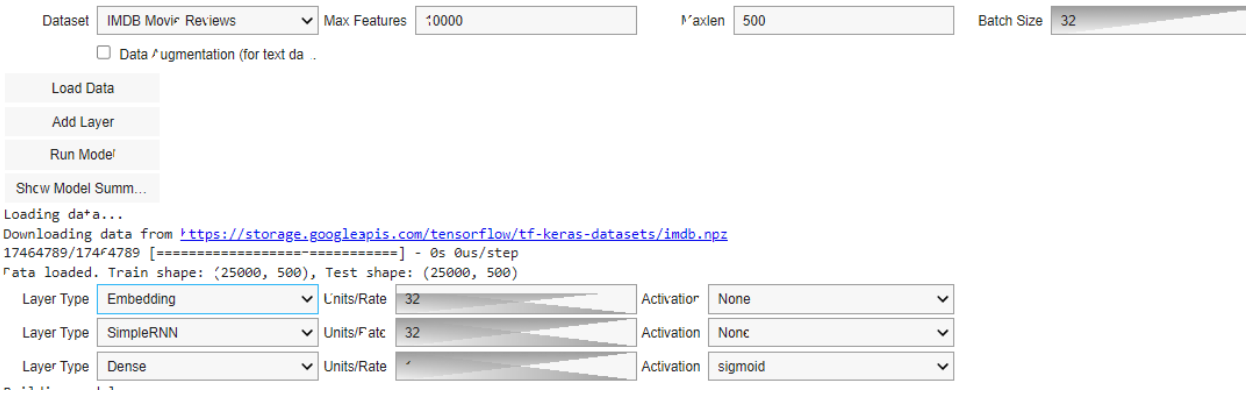
	precision	recall	f1-score	support
0	1.0	1.0	1.0	7.0
1	1.0	1.0	1.0	12.0
2	1.0	1.0	1.0	10.0
accuracy	1.0	1.0	1.0	1.0
macro avg	1.0	1.0	1.0	29.0
weighted avg	1.0	1.0	1.0	29.0

Metrics for Testing Data:

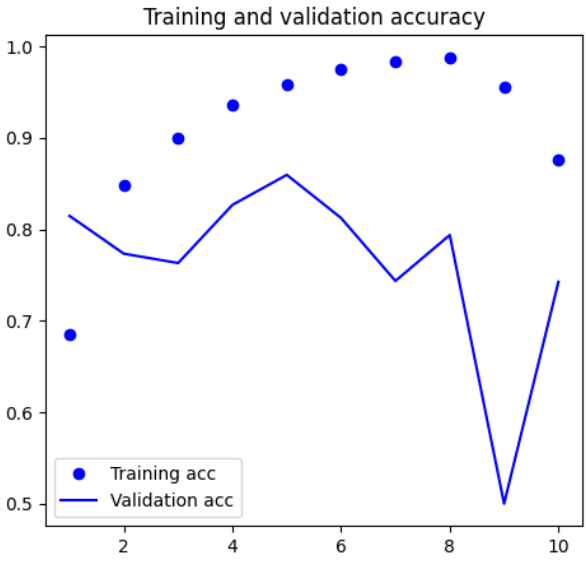
	Accuracy	Error Rate
Testing	1.0	0.0

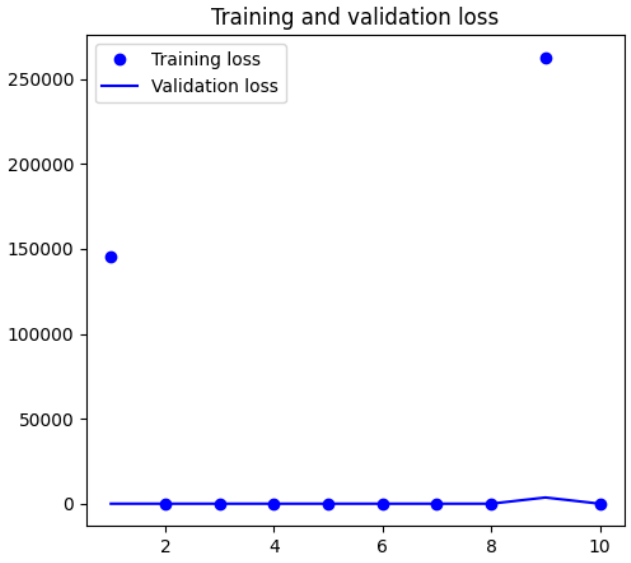
- RNN Module

Similarly, we have RNN module to implement Machine Learning for sequential dataset. Its features are very limited and it can be further developed by adding more options in it.




```
Epoch 1/10
625/625 [==============================] - 67s 107ms/step - loss: 145345.7500 - accuracy: 0.6856 - val_loss: 0.4366 - val_accuracy: 0.8148
Epoch 2/10
625/625 [==============================] - 68s 108ms/step - loss: 4.4958 - accuracy: 0.8482 - val_loss: 0.4657 - val_accuracy: 0.7734
Epoch 3/10
625/625 [==============================] - 65s 103ms/step - loss: 0.9001 - accuracy: 0.8997 - val_loss: 0.4817 - val_accuracy: 0.7632
Epoch 4/10
625/625 [==============================] - 68s 109ms/step - loss: 4.1175 - accuracy: 0.9364 - val_loss: 0.3989 - val_accuracy: 0.8268
Epoch 5/10
625/625 [==============================] - 64s 103ms/step - loss: 0.1483 - accuracy: 0.9587 - val_loss: 0.4266 - val_accuracy: 0.8596
Epoch 6/10
625/625 [==============================] - 67s 107ms/step - loss: 0.2310 - accuracy: 0.9746 - val_loss: 0.5319 - val_accuracy: 0.8124
Epoch 7/10
625/625 [==============================] - 65s 103ms/step - loss: 0.1151 - accuracy: 0.9836 - val_loss: 0.7375 - val_accuracy: 0.7436
Epoch 8/10
625/625 [==============================] - 66s 105ms/step - loss: 6.5215 - accuracy: 0.9880 - val_loss: 0.7129 - val_accuracy: 0.7938
Epoch 9/10
625/625 [==============================] - 65s 103ms/step - loss: 262698.8125 - accuracy: 0.9558 - val_loss: 3646.4121 - val_accuracy: 0.5000
Epoch 10/10
625/625 [==============================] - 66s 105ms/step - loss: 184.5802 - accuracy: 0.8762 - val_loss: 0.9476 - val_accuracy: 0.7426
782/782 [==============================] - 19s 24ms/step - loss: 0.9560 - accuracy: 0.7402
Test loss: 0.9530114145278931
Test accuracy: 0.7402399778366089
```





6. Future Recommendations:

These three modules of WekaLab can be combined in a website or application interface to use it anywhere. We can also develop a separate module of pre-processing that includes all the necessary pre-processing steps before solving any classification problem. The CNN Module of WekaLab can solve only image classification problem now. We can develop it more to solve other image related problems like image segmentation, image detection etc. The RNN Module of WekaLab has very limited options in it. We can develop it more to handle all the necessary steps needed to apply RNN on any dataset.

7. Conclusion:

First of all, we analyze different automated Machine Learning tools available for non-technical people like Weka and Google AutoML. We see that Weka use personal computers processing power that can lead to computational resources constraints for large datasets. On the other hand, Google AutoML is a paid tool that can lead to financial constraints as not everyone can pay for it. To make Machine Learning accessible to everyone, we develop Weka like GUI on Google Colab called WekaLab to access google computers processing power without paying for it. We have seen that WekaLab accuracy is very good as compared to Weka and Google AutoML in terms of time efficiency and effectively using computational resources. We are getting image classification with high accuracy making it the most powerful tool as we are not paying anything but accessing Google computers processing power to train Machine Learning or Deep Learning models.

8. References:

- https://waikato.github.io/weka-site/index.html
- https://cloud.google.com/automl
- https://cloud.google.com/