Deep Learning

Medical Image Multi-Label Classification

Tomeh Lara 507

Introduction:

This paper outlines a project aimed to train deep learning models in order to classify medical x-ray images obtained from an MRI machine. It is worth noting that these images have multiple labels which mean that the image can belong to more than one class or one category. 12,000 image files make up the training set, and the validation set has 3,000 image files, while the test set includes 5,000 image files.

In order to design a deep learning model, we need some libraries that are provided by TensorFlow in the first place. We also need some other libraries such as Pandas, Numpy, Matplotlib, and OS. At first, a dedicated environment was created for TensorFlow projects, using Anaconda Navigator or Anaconda Prompt, TensorFlow was loaded and the environment was used to set up the project. Then, to read the files, the listdir function provided by OS module was used, and by using the for loop to read all the images and convert them into arrays and into float type, the images are of the same size and are in black and white. Validation and Testing images were read in the same way. In order to document the ineffectiveness, I made the image division by 255 after converting it to the float type, but without this operation the model's scores were rising. As for labels, it is simply using the read_csv function from Pandas. Since this project is a multi-Label image classification, therefore, each image may belong to more than one label at the same time, and it may also the image has no label at all. Then a function for calculating precision, recall, and f1 score values was defined.

Convolutional Neural Network (CNN):

It is a neural network that learns from input and is effective for categorizing or identifying objects in images. It has been utilized in this study. The basic layers in a CNN network are the input layer and the output layer, as well as the many hidden layers between them. These layers perform various operations to modify the data and learn its various characteristics.

The most popular layers are:

1- **Convolution layer:** which applies a filter with a specific value to the input images to activate and extract the various characteristics in the images. It gives the feature map which contains details about the image, including its corners and edges. This feature map is later supplied to further layers to teach them additional features about the input image. In this layer many parameters are set including filters, kernel size, padding, and activation function like ReLU, SoftMax, sigmoid, and tanh, in addition to those parameters, input shape, which is specified exclusively in the first layer, and its value depends on the size of the images.
2- **Pooling layer:** used to cut down on the number of parameters the network needs to learn, the pooling layer employs down sampling. There are three types of pooling:
    - max pooling which chooses the batch's highest pixel value.
    - min pooling, which chooses its lowest value.
    - average pooling, which chooses the batch's average pixel value.

    Pooling layer has parameters like pool size and strides which is the number of pixels moves over the input matrix. Since the images' backgrounds are black, the max pooling is used to reduce the computation costs.

3- **ReLU layer:** which trains more quickly and effectively by maintaining positive values while shifting negative values to zero. It gets its name because it only sends active properties to the following layer, it takes maximum value, negative slope, and threshold parameters.
4- **Dropout layer:** in order to avoid overfitting which happens when a model performs good on training data that it has a negative influence on the model's performance when applied to validation data, and the dropout is based according to a specific dropout rate.
5- **Flatten layer:** which flattens the output of the preceding layers into a single vector that can be used as an input for the following layer.
6- **Dense layer:** takes the feature analysis inputs and applies weights to anticipate the proper label, and returns the final probability for each label, it has several parameters, units and activation were used. The last dense layer should take a value of 3 for the unit because we have three labels and

sigmoid activation function, given that we are doing a binary classification for a multi-label classification, if we were doing a multi-class classification, we would have chosen SoftMax activation function.

Implementation:

Five different models were designed, each time a certain number of parameters were changed (considering that hyperparameter tuning is obtained manually in such type of models) in order to evaluate the final performance of the models. In each model, something changes such as the number of filters in the convolution layer, the size of the kernel, the size of the pool in the max pooling layer, the number of strides, or the percentage of drops out, the number of dense layers, and flatten layers, the number of convolutional blocks, or the size of the patch, the number of epochs, and optimizer's parameters.

Then, for compiling the model, Adam optimizer was used with binary cross entropy loss since this is a multi-label image classification, while categorical cross entropy is for multi-class classification when each sample belongs to a single class. Then the accuracy of the model was calculated using evaluate function on both the training data and the validation data, with the value of loss, accuracy, F1_Score, Recall and Precision calculated. The classification report for the training data and the validation data was extracted to clarify the metrics for each label. Plotting was done for the previously mentioned four metrics in both the training set and the validation set. In the end, the model was saved, and predictions were made for the testing image labels, since the outputs of the CNN models are float numbers, and we need zeros and ones, threshold was specified with a certain value in order to convert these numbers to match the required file format and then they stored in a csv file.
There was a problem during the implementation, as it was noticed that there were several fluctuations in accuracy and other parameters when plotting, so kernel regularizer and bias regularizer were added in the convolution and dense layers in model 3.

Conclusion:

In conclusion, the first and fourth models achieved better results than the rest of models as the F1 score in validation sets was 0.86 and 0.87 in both of the previously mentioned models, respectively. They have the size of the kernel five, the size of the pool three, and the percentage of the drop out 0.5 or less than that. Model 1 achieved 0.84 precision score for class 3. Also, in model 3 I used kernel regularizer and bias regularizer to reduce fluctuating and this is shown in the plotting of that model, but this operation did not work will for the final results, it did not improve the metrics as well, except for accuracy score for validation dataset. In model 5 I got very high scores for precision in training data, it achieved 1 for the second class and 0.99 for the two other classes as it shown in the figures of the fifth model.

The following table show the results for each model on training and validation sets both.

Models' metrics and parameters										
Parameters	Model 1		Model 2		Model 3		Model 4		Model 5	
Epochs	20		20		15		20		20	
Batch size	64		64		128		128		128	
Dataset	TD	VD	TD	VD	TD	VD	TD	VD	TD	VD
Loss	0.1667	0.2708	0.1969	0.2617	0.4443	0.4665	0.1783	**0.2572**	**0.0504**	0.6018
Accuracy	0.5027	0.5520	0.5322	0.5623	**0.9764**	**0.9660**	0.8782	0.8857	0.8521	0.8637
Recall	**0.9323**	0.8656	0.9294	**0.8921**	0.8161	0.8348	0.9374	0.8920	0.9676	0.7918
Precision	0.9210	0.8631	0.8852	0.8537	0.8035	0.7960	0.9093	**0.8666**	**0.9943**	0.8652
F1 Score	0.9253	0.8622	0.9054	0.8706	0.8065	0.8122	0.9218	**0.8770**	**0.9803**	0.8241

TD: training data, VT: validation data

The following table represents the average Precision per class on the validation set for five models.

	Model 1	Model 2	Model 3	Model 4	Model 5
	Precision per class				
Class 1	0.89	0.89	0.88	**0.90**	0.89
Class 2	0.85	0.85	0.81	**0.87**	0.85
Class 3	**0.84**	0.78	0.61	0.81	0.81

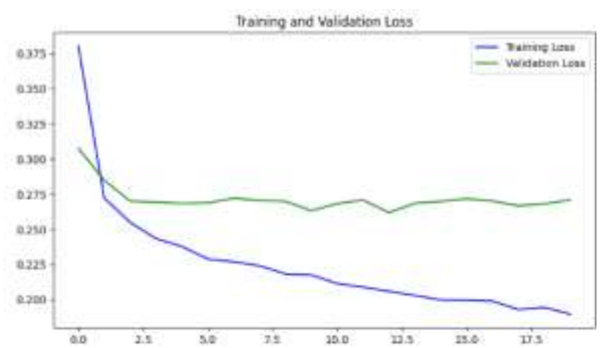
Classification Report for Training data and Validation data, respectively (Model 1)

Classification report of Training data

	precision	recall	f1-score	support
0	0.94	0.96	0.95	5564
1	0.90	0.94	0.92	4334
2	0.92	0.86	0.89	2720
micro avg	0.92	0.93	0.93	12618
macro avg	0.92	0.92	0.92	12618
weighted avg	0.92	0.93	0.93	12618
samples avg	0.43	0.45	0.44	12618

Classification report of Validating data

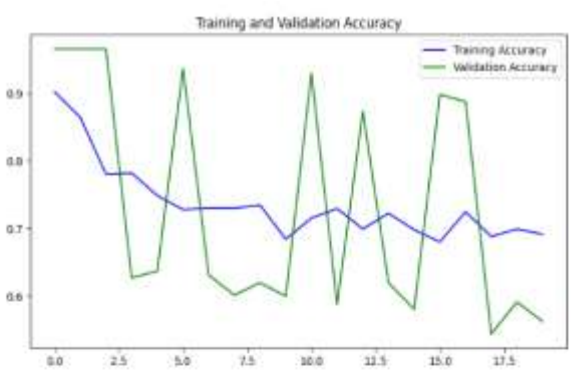
	precision	recall	f1-score	support
0	0.89	0.94	0.91	1447
1	0.85	0.86	0.85	1153
2	0.84	0.72	0.78	684
micro avg	0.86	0.87	0.87	3284
macro avg	0.86	0.84	0.85	3284
weighted avg	0.86	0.87	0.86	3284
samples avg	0.43	0.44	0.42	3284

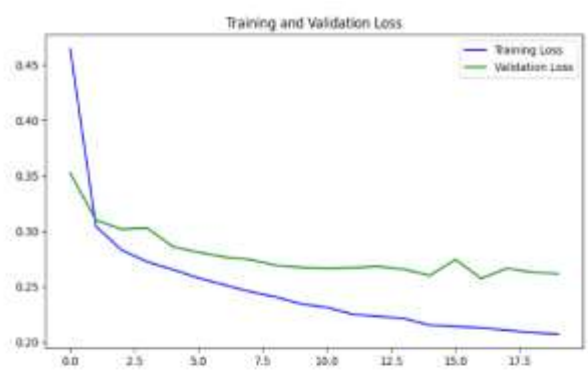


Classification Report for Training data and Validation data, respectively (Model 2)

```
Classification report of Training data
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      5564
           1       0.87      0.93      0.90      4334
           2       0.83      0.87      0.85      2720

   micro avg       0.89      0.93      0.91     12618
   macro avg       0.88      0.92      0.90     12618
weighted avg       0.89      0.93      0.91     12618
 samples avg       0.42      0.44      0.42     12618
```

```
Classification report of Validating data
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      1447
           1       0.85      0.89      0.87      1153
           2       0.78      0.80      0.79       684

   micro avg       0.86      0.89      0.87      3284
   macro avg       0.84      0.88      0.86      3284
weighted avg       0.86      0.89      0.87      3284
 samples avg       0.42      0.45      0.43      3284
```





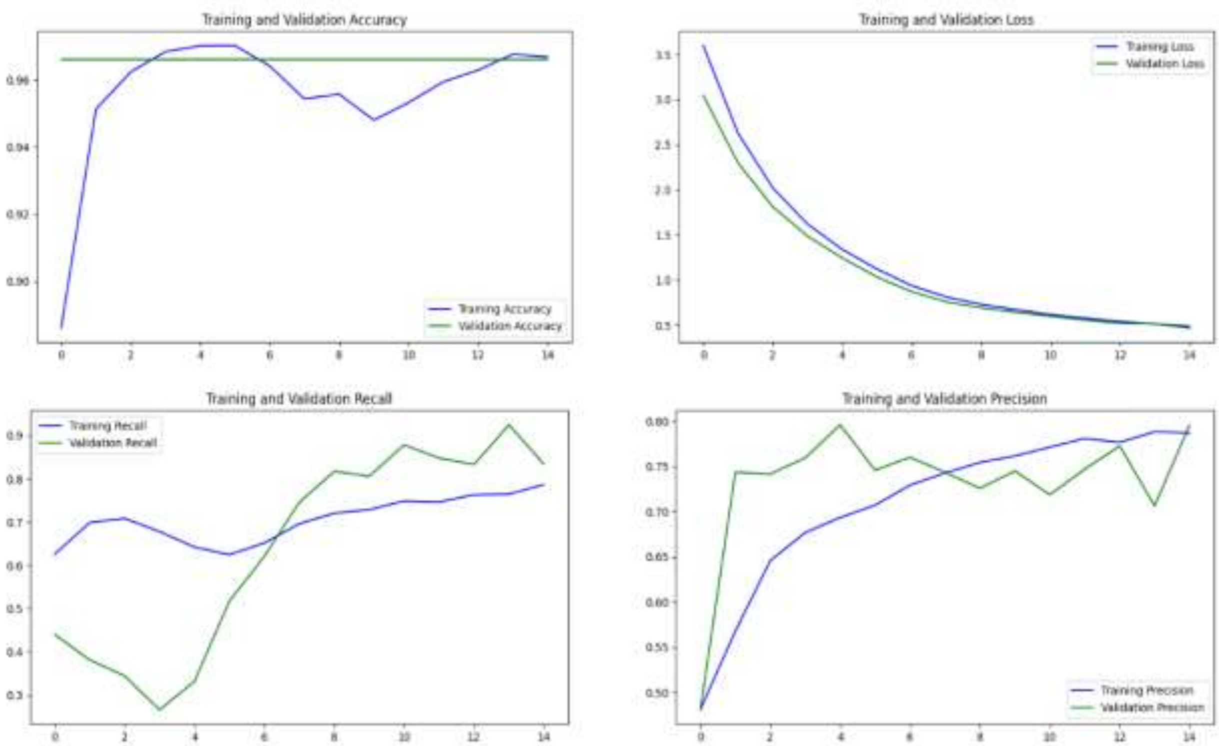
Classification Report for Training data and Validation data, respectively (Model 3)

Classification report of Training data

	precision	recall	f1-score	support
0	0.91	0.87	0.89	5564
1	0.80	0.83	0.81	4334
2	0.62	0.69	0.65	2720
micro avg	0.80	0.82	0.81	12618
macro avg	0.78	0.80	0.79	12618
weighted avg	0.81	0.82	0.81	12618
samples avg	0.35	0.38	0.35	12618

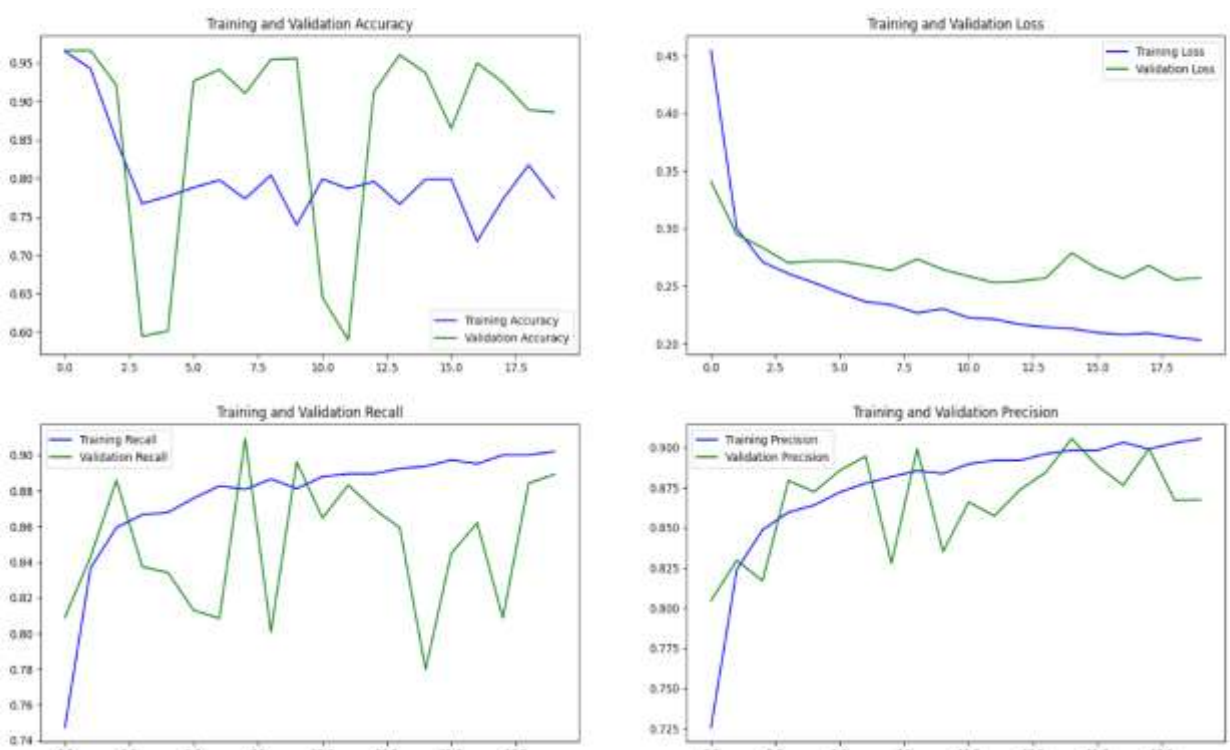
Classification report of Validating data

	precision	recall	f1-score	support
0	0.88	0.90	0.89	1447
1	0.81	0.86	0.83	1153
2	0.61	0.68	0.64	684
micro avg	0.80	0.84	0.82	3284
macro avg	0.77	0.81	0.79	3284
weighted avg	0.80	0.84	0.82	3284
samples avg	0.39	0.42	0.39	3284



Classification Report for Training data and Validation data, respectively (Model 4)

```
Classification report of Training data
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      5564
           1       0.89      0.94      0.92      4334
           2       0.87      0.89      0.88      2720

   micro avg       0.91      0.94      0.92     12618
   macro avg       0.90      0.93      0.92     12618
weighted avg       0.91      0.94      0.92     12618
 samples avg       0.42      0.45      0.43     12618
Classification report of Validating data
              precision    recall  f1-score   support

           0       0.90      0.94      0.92      1447
           1       0.87      0.88      0.87      1153
           2       0.81      0.83      0.82       684

   micro avg       0.87      0.89      0.88      3284
   macro avg       0.86      0.88      0.87      3284
weighted avg       0.87      0.89      0.88      3284
 samples avg       0.43      0.45      0.43      3284
```



Classification Report for Training data and Validation data, respectively (Model 5)

```
375/375 [==============================] - 40s 105ms/step
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      5564
           1       1.00      0.97      0.98      4334
           2       0.99      0.95      0.97      2720

   micro avg       0.99      0.97      0.98     12618
   macro avg       0.99      0.96      0.98     12618
weighted avg       0.99      0.97      0.98     12618
 samples avg       0.47      0.47      0.47     12618

94/94 [==============================] - 11s 119ms/step
              precision    recall  f1-score   support

           0       0.89      0.90      0.90      1447
           1       0.85      0.76      0.80      1153
           2       0.81      0.63      0.71       684

   micro avg       0.87      0.79      0.83      3284
   macro avg       0.85      0.76      0.80      3284
weighted avg       0.86      0.79      0.82      3284
 samples avg       0.42      0.40      0.40      3284
```