UNIVERSITÀ DEGLI STUDI DI BRESCIA

DII, COMMUNICATION TECHNOLOGY & MULTIMEDIA

REMOTE SENSING DATA ANALYSIS

PROJECT ON: CNN BASED LESION CLASSIFICATION

BY: ALEMU SISAY NIGRU,

MAT. NO.: 730159

SUBMITTED TO: PROF, ALBERTO SIGNORONI & DR. MATTIA SAVARDI

OUTLINE

- INTRODUCTION
- DATA LOADING AND PRE-PROCESSING
- DEALING WITH CLASS IMBALANCE
- TRAIN, VALIDATION AND TEST SPLIT
- DEVELOP AND TRAIN A MODEL
- MODELS EVALUATIONS
- CONCLUSION

INTRODUCTION

- THE HAM10000 (“HUMAN AGAINST MACHINE WITH 10000 TRAINING IMAGES”) DATASET WHICH CONTAINS 10,015 DERMATOSCOPIC IMAGES WAS MADE PUBLICLY AVAILABLE BY THE HARVARD DATABASE ON JUNE 2018. A METADATA FILE WITH DEMOGRAPHIC INFORMATION OF EACH LESION IS ADDITIONALLY PROVIDED.
- THE 7 CLASSES OF SKIN CANCER LESIONS INCLUDED IN THIS DATASET ARE:
  - MELANOCYTIC NEVI (NV)
  - MELANOMA (MEL)
  - BENIGN KERATOSIS-LIKE LESIONS (BKL)
  - BASAL CELL CARCINOMA (BCC)
  - ACTINIC KERATOSES (AKIEC)
  - VASCULAR LESIONS (VAS)
  - DERMATOFIBROMA (DF)

METADATA

- The metadata are all encoded within a single CSV file (comma-separated value) file, with each classification response in a row. With different columns including image id, lesion class, age, localization and dataset.

lesion_id	image_id	dx	dx_type	age	sex	localization	dataset
HAM_0000550	ISIC_0024306	nv	follow_up	45	male	trunk	vidir_molemax
HAM_0003577	ISIC_0024307	nv	follow_up	50	male	lower extremity	vidir_molemax
HAM_0001477	ISIC_0024308	nv	follow_up	55	female	trunk	vidir_molemax
HAM_0000484	ISIC_0024309	nv	follow_up	40	male	trunk	vidir_molemax
HAM_0003350	ISIC_0024310	mel	histo	60	male	chest	vidir_modern
HAM_0000981	ISIC_0024311	nv	follow_up	75	female	back	vidir_molemax
HAM_0001359	ISIC_0024312	bkl	histo	75	male	lower extremity	rosendahl
HAM_0002869	ISIC_0024313	mel	histo	50	female	back	rosendahl
HAM_0002198	ISIC_0024314	nv	histo	75	male	lower extremity	vidir_modern
HAM_0007538	ISIC_0024315	mel	histo	55	male	trunk	vidir_modern
HAM_0002718	ISIC_0024316	nv	follow_up	55	male	back	vidir_molemax
HAM_0003951	ISIC_0024317	nv	histo	35	female	abdomen	vidir_modern
HAM_0002450	ISIC_0024318	df	consensus	65	female	lower extremity	vidir_modern

DATA LOADING AND PRE-PROCESSING

After downloading the datasets from the given website, I have uploaded into the google drive folder and I tried to alter the structure of the dataset into a format which enables me to load the data more easily. In this stage the following activities are performed:

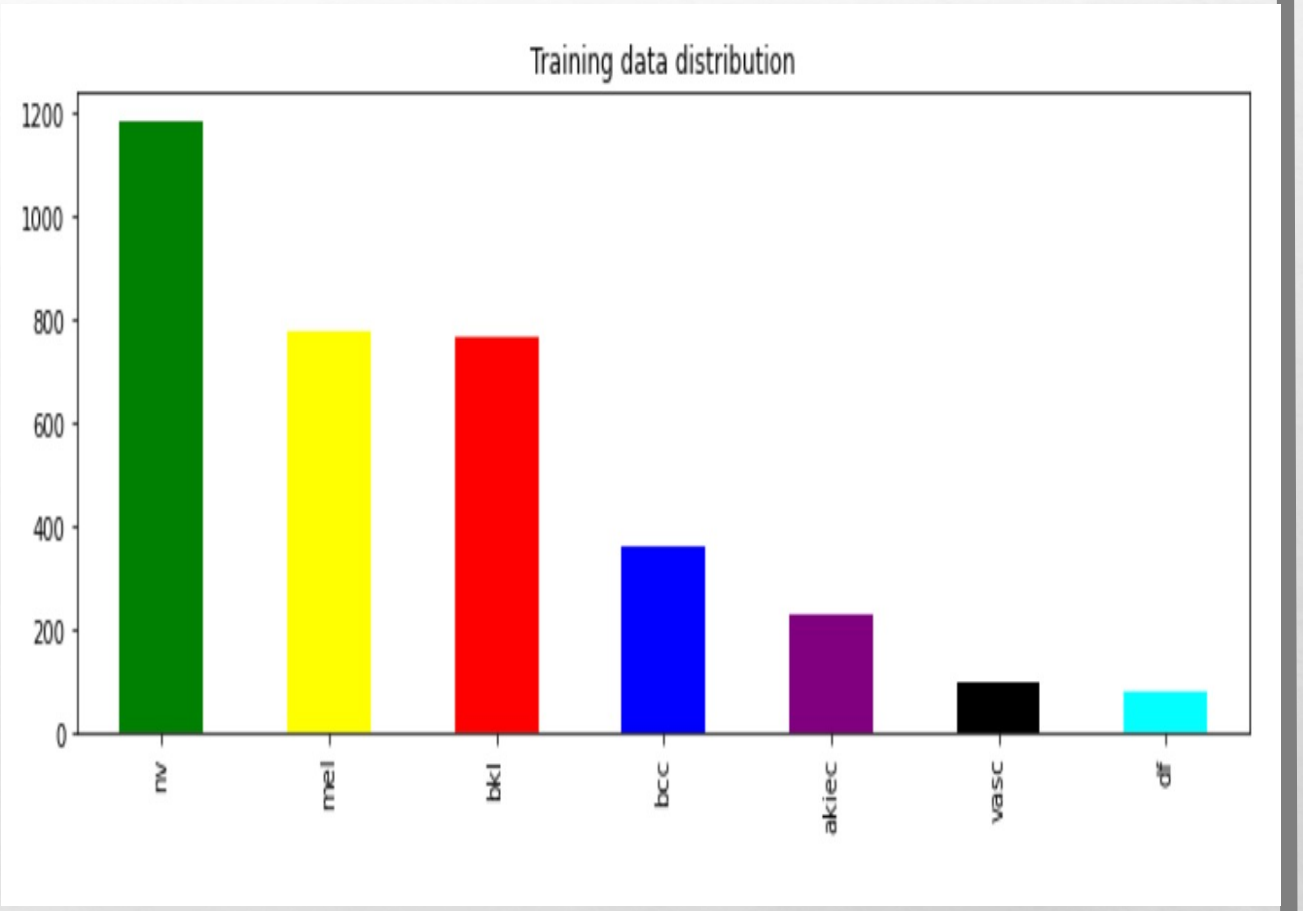
- Loading a metadata CSV file and assign it to a panda dataframe.
- Loading an image from the drive folder
- Creating a dictionary of images and their labels.
- Map an image path to the dataframe by its filename from the dataframe.
- Normalizing images to their mean and standard deviation & resizing images

TRAIN, VALIDATION AND TEST SPLIT

- We first split the dataset to 75% training data and 25% testing data. We then take the 25% testing data and split it into 50% validation data and 50% testing data, to form our 75% training – 12.5% validation – 12.5% testing. Note that the split will be applied across each class individually to ensure there is enough samples from each class in each split for accurate modeling, meaning each class will be split 75:12.5:12.5 as well. This is done by setting '**stratify**' in train_test_split function to our target.
  - Training: = 75%
  - Validation: 50% of 25% = 12.5%
  - Testing: 50% of 25% = 12.5%

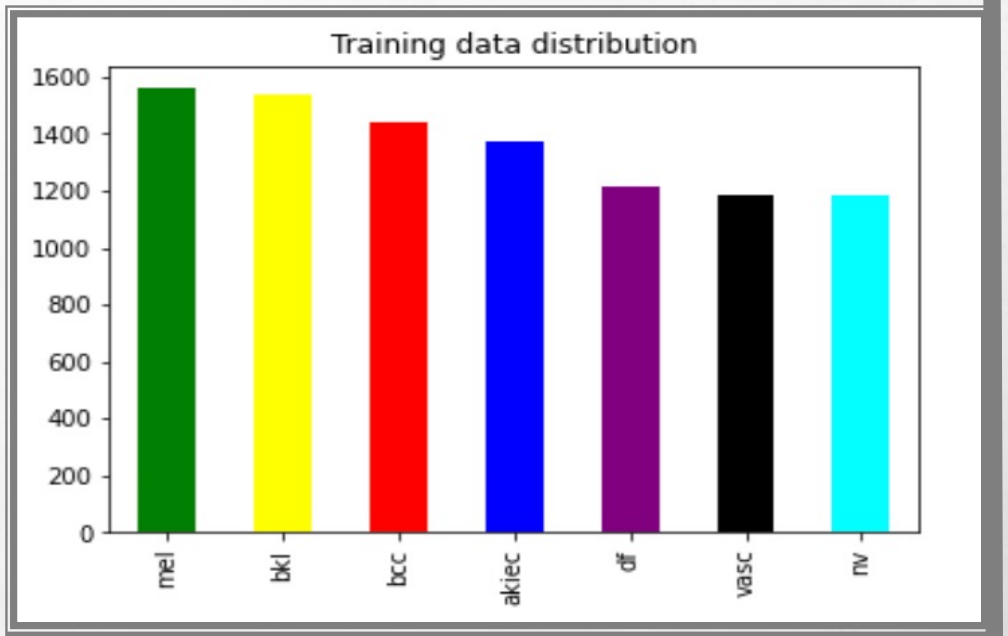
DEALING WITH CLASS IMBALANCE

- After selecting 5000 images from the total of 10,000 sets, the train, validation and test split is performed and the trainind data distribution is shown on the below graph.
- As it can be seen from the graph the data distribution is highly imbalanced.



...CONT'D

- Now, After doing data augmentation on the training data frame, the training distribution becomes balanced as it can be shown on the graph.




```
mel      1558
bkl      1538
bcc      1440
akiec    1374
df       1215
vasc     1188
nv       1183
Name: dx, dtype: int64
```

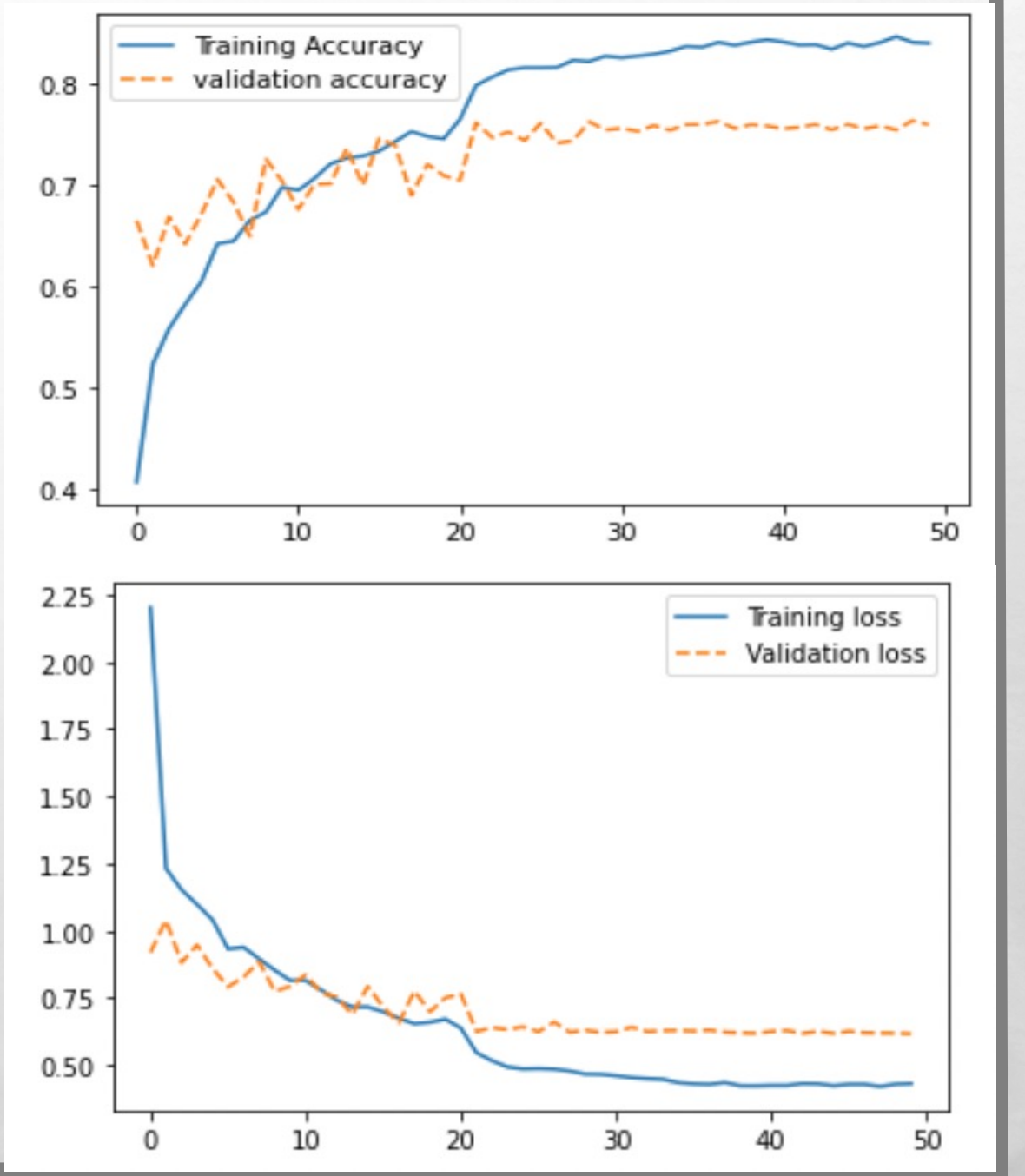
DEVELOP AND TRAIN A MODEL

- I have created a 4 convolutional neural network with ReLU activation, each CNN is followed by Batch normalization and Max pooling layers with some dropouts in between.
- I added five flatten fully connected layers at the output of the last convolutional layer.
- The last fully connected layer has 7 outputs each output corresponds to each class.
- The model has a total of around 3.09 million parameters

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 126, 126]             448
       BatchNorm2d-2         [-1, 16, 126, 126]              32
              ReLU-3         [-1, 16, 126, 126]               0
         MaxPool2d-4           [-1, 16, 63, 63]               0
            Conv2d-5           [-1, 32, 61, 61]           4,640
       BatchNorm2d-6           [-1, 32, 61, 61]              64
         MaxPool2d-7           [-1, 32, 30, 30]               0
              ReLU-8           [-1, 32, 30, 30]               0
            Conv2d-9           [-1, 64, 28, 28]          18,496
      BatchNorm2d-10           [-1, 64, 28, 28]             128
             ReLU-11           [-1, 64, 28, 28]               0
           Conv2d-12           [-1, 64, 26, 26]          36,928
      BatchNorm2d-13           [-1, 64, 26, 26]             128
        MaxPool2d-14           [-1, 64, 13, 13]               0
             ReLU-15           [-1, 64, 13, 13]               0
           Linear-16                 [-1, 140]       1,514,380
           Linear-17                 [-1, 140]       1,514,380
           Linear-18                  [-1, 32]           4,512
           Linear-19                  [-1, 32]           4,512
           Linear-20                   [-1, 7]             231
================================================================
Total params: 3,098,879
Trainable params: 3,098,879
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 10.53
Params size (MB): 11.82
Estimated Total Size (MB): 22.54
----------------------------------------------------------------
```

MODEL EVALUATIONS

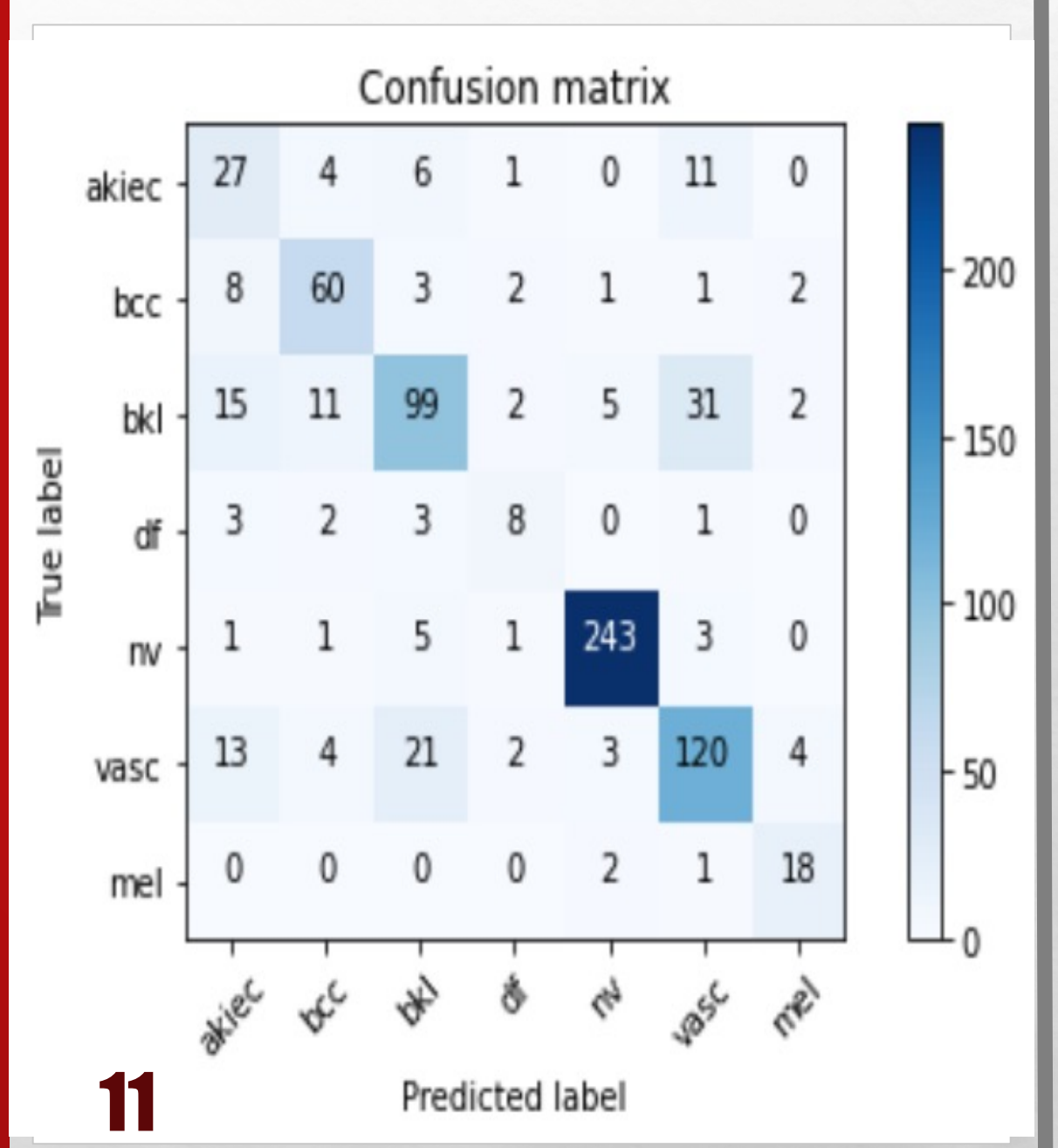
- After 50 epochs, the model achieves accuracy of 83.99% for the training set, 75.94% for the validation set and 76.60% for test set.
- As it can be seen from the graph, as the number of epoch increases, our trained model will experience overfitting problem.



…CONT'D

	precision	recall	f1-score	support
akiec	0.40	0.55	0.47	49
bcc	0.73	0.78	0.75	77
bkl	0.72	0.60	0.66	165
df	0.50	0.47	0.48	17
nv	0.96	0.96	0.96	254
vasc	0.71	0.72	0.72	167
mel	0.69	0.86	0.77	21
accuracy			0.77	750
macro avg	0.67	0.70	0.69	750
weighted avg	0.77	0.77	0.77	750

- As it can be seen from the classification report, some of the classes are not well classified specifically, akiec and df classes.
- So, it can be optimized using large training dataset and deeper network with more learnable parameters.
- So, to get rid of these overfitting problem, I decided to train my model with Kfold cross validation.

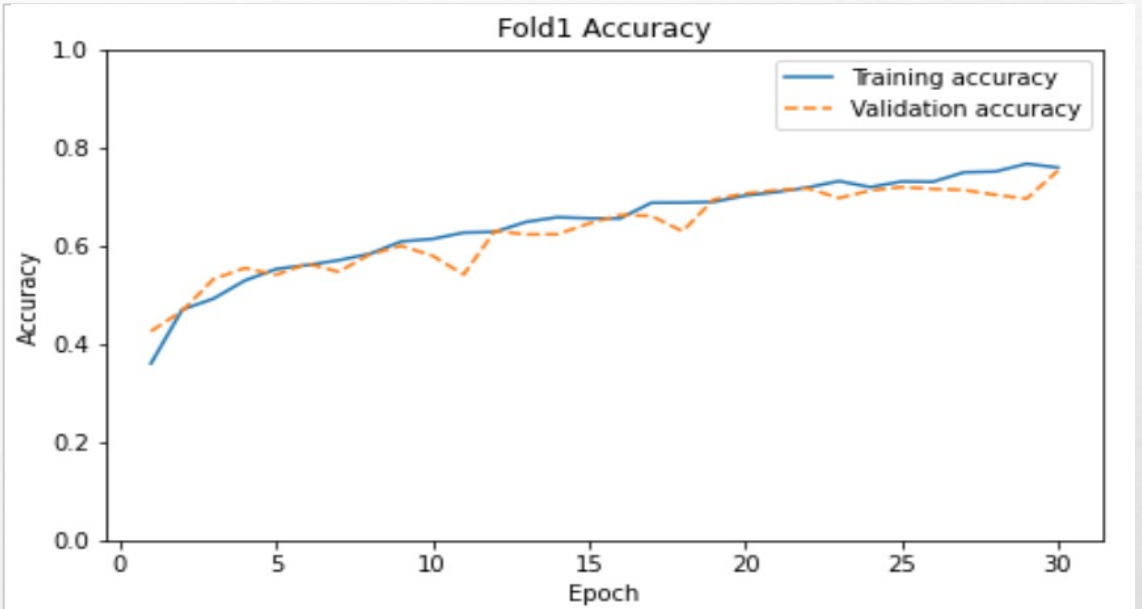


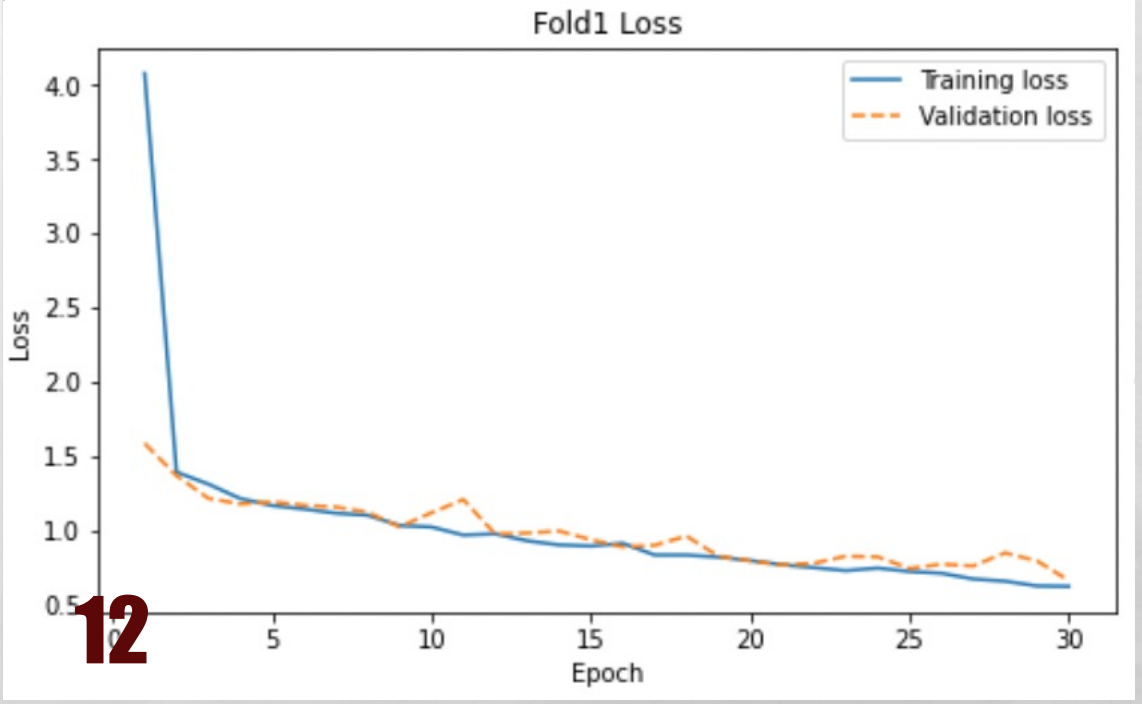
KFOLD CROSS VALIDATION, K=4

- It was good to use k value more than or equivalent to the total number of class, but due to computational complexity (Computation time), I select 4 as a k value, just to study the effect of Kfold cross validation.

- So, I trained the model with Kfold cross validation techniques and the resulting measurement of accuracy and loss are plotted as follows for each fold.

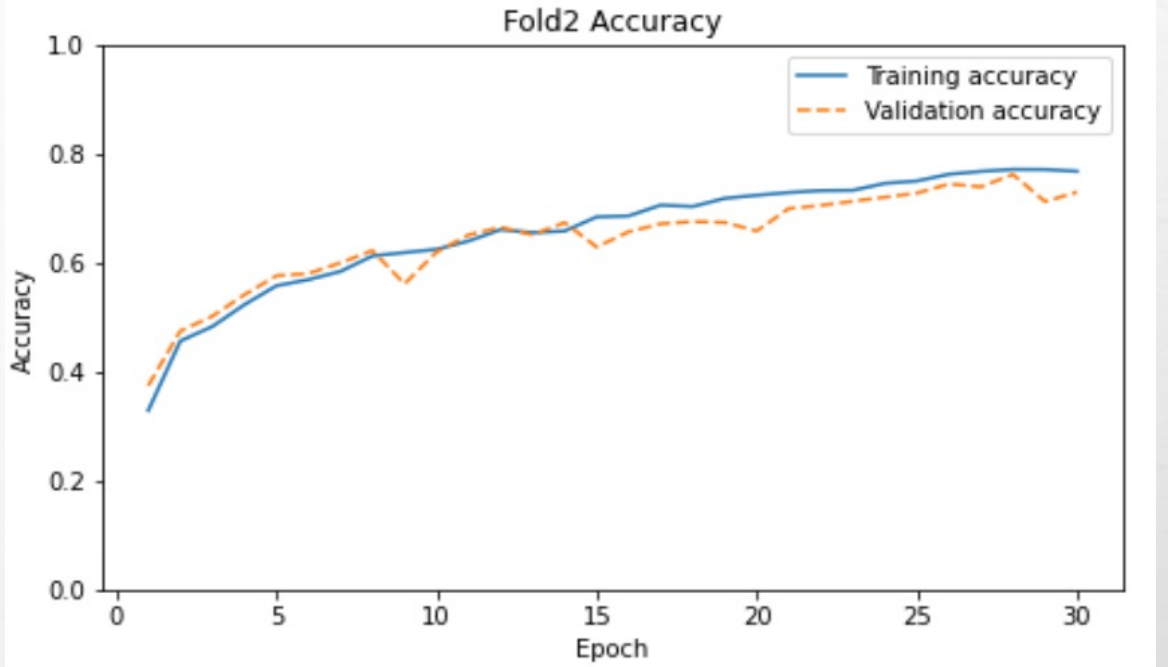
FOLD 1 ➔

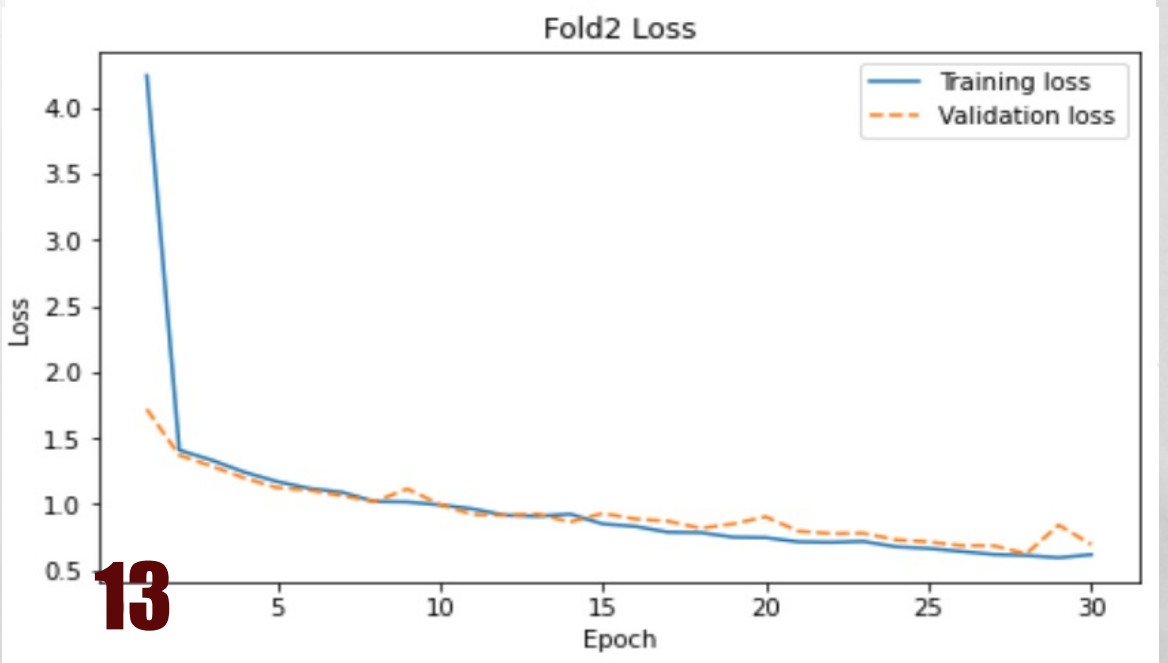




...CONT'D

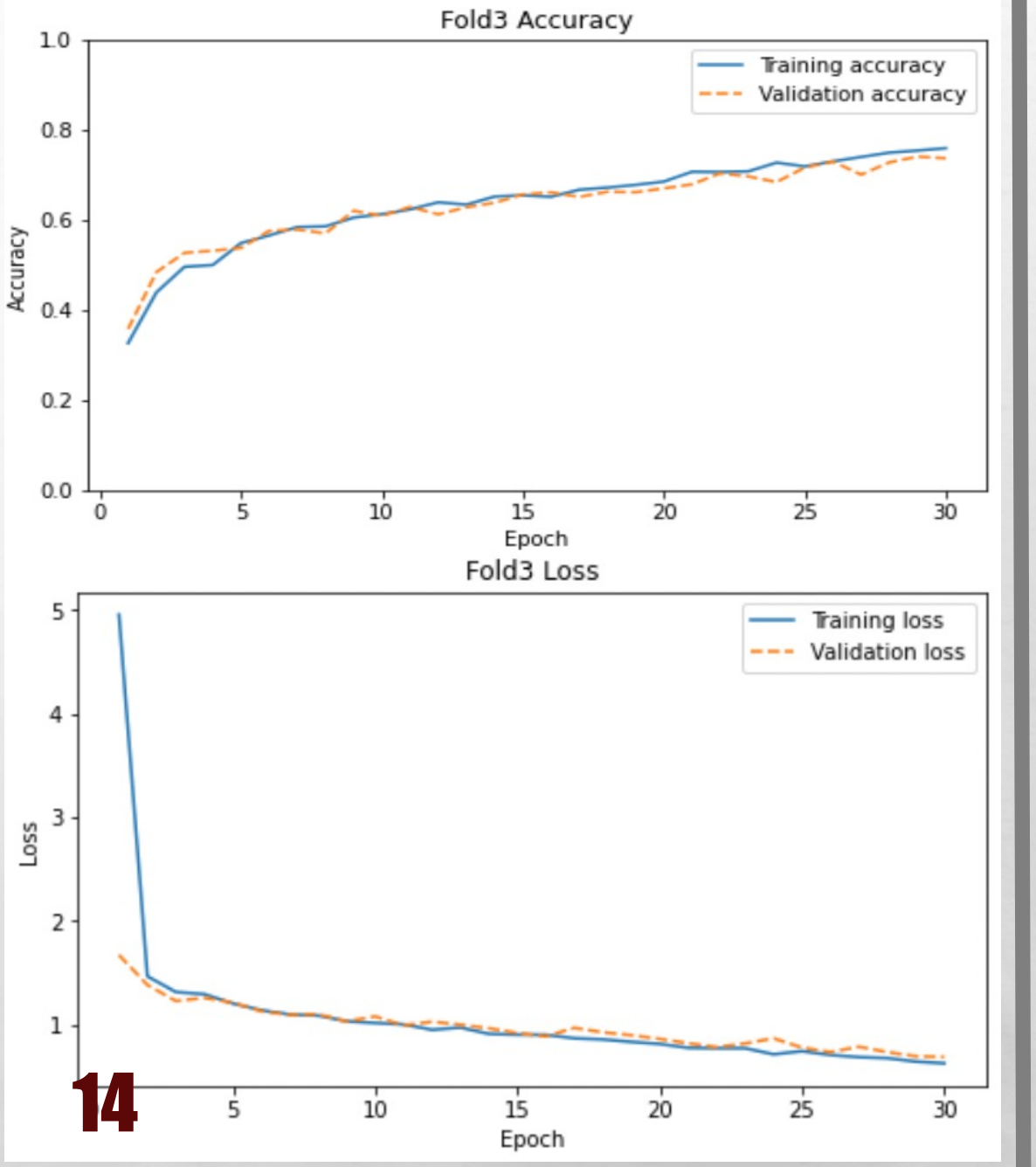
FOLD 2 ➔


Fold2 Accuracy
Training accuracy
Validation accuracy
Accuracy
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30

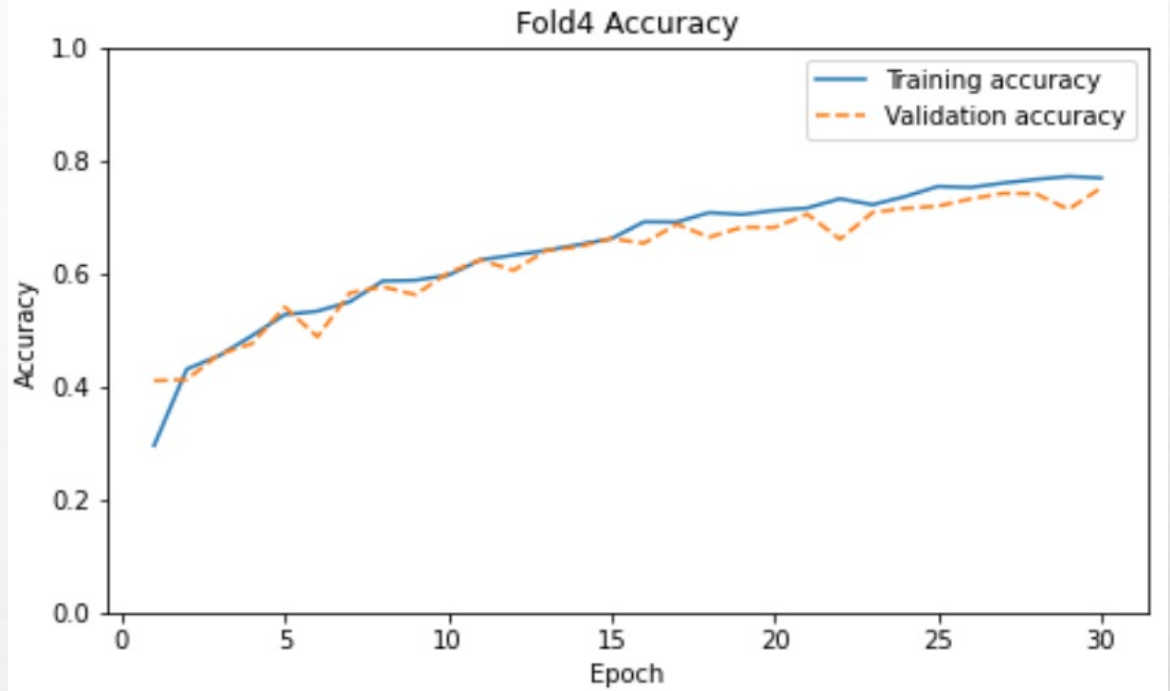

Fold2 Loss
Training loss
Validation loss
Loss
Epoch
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
5
10
15
20
25
30

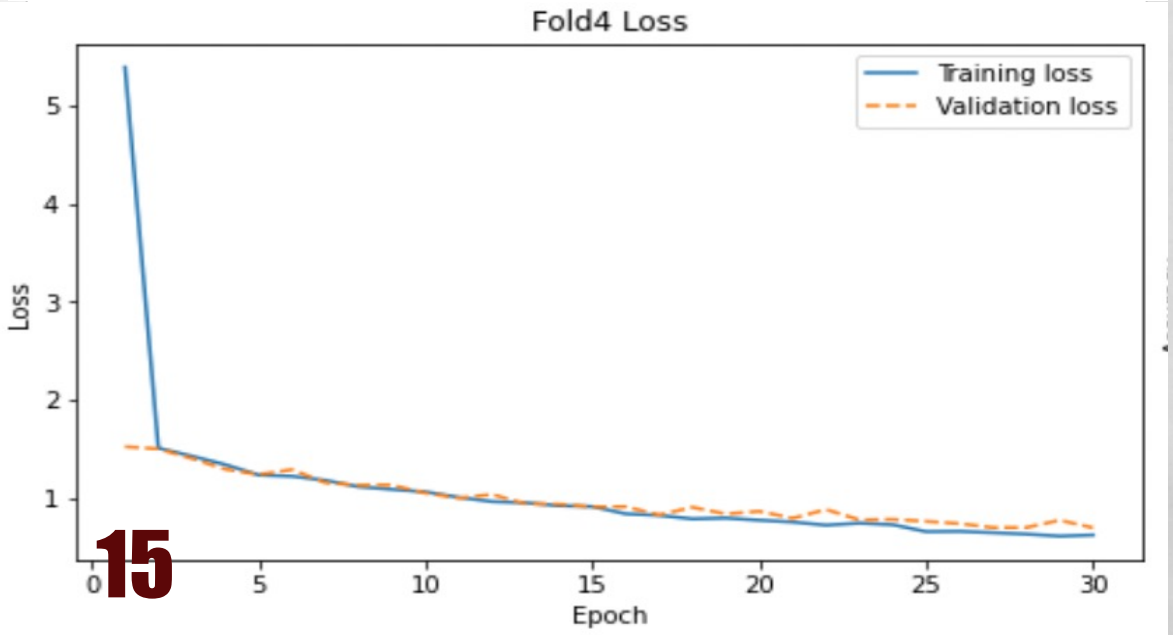
...CONT'D

FOLD 3 ➔


Fold3 Accuracy
Training accuracy
Validation accuracy
Accuracy
Epoch
Fold3 Loss
Training loss
Validation loss
Loss
Epoch

...CONT'D

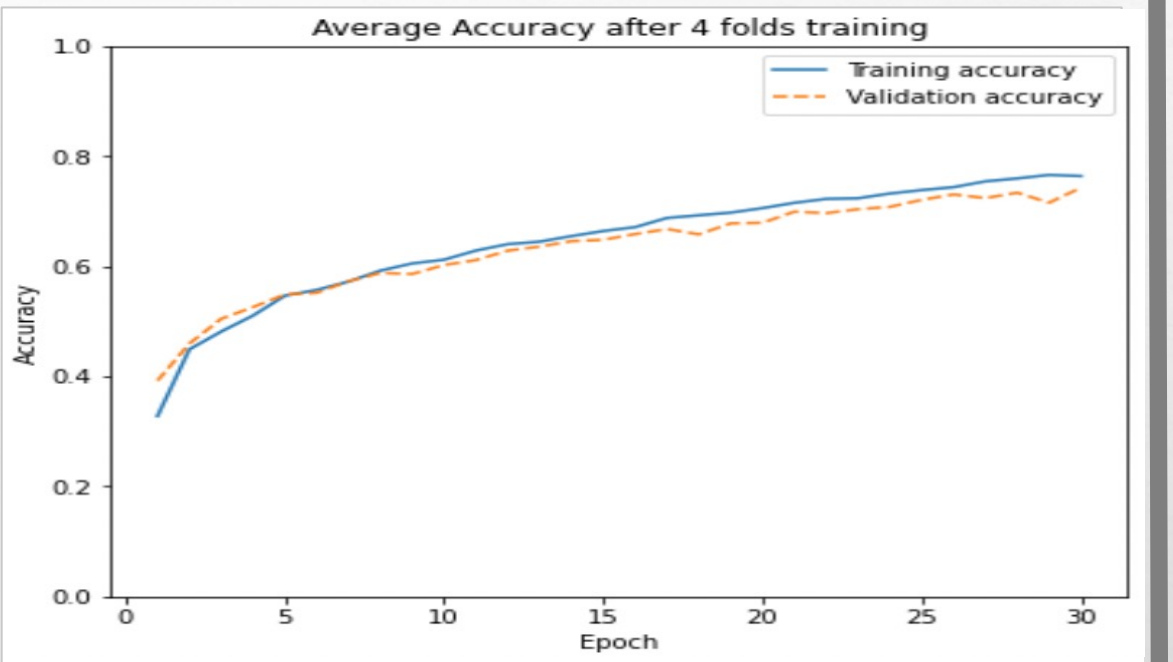

Fold4 Accuracy
Training accuracy
Validation accuracy
Accuracy
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30

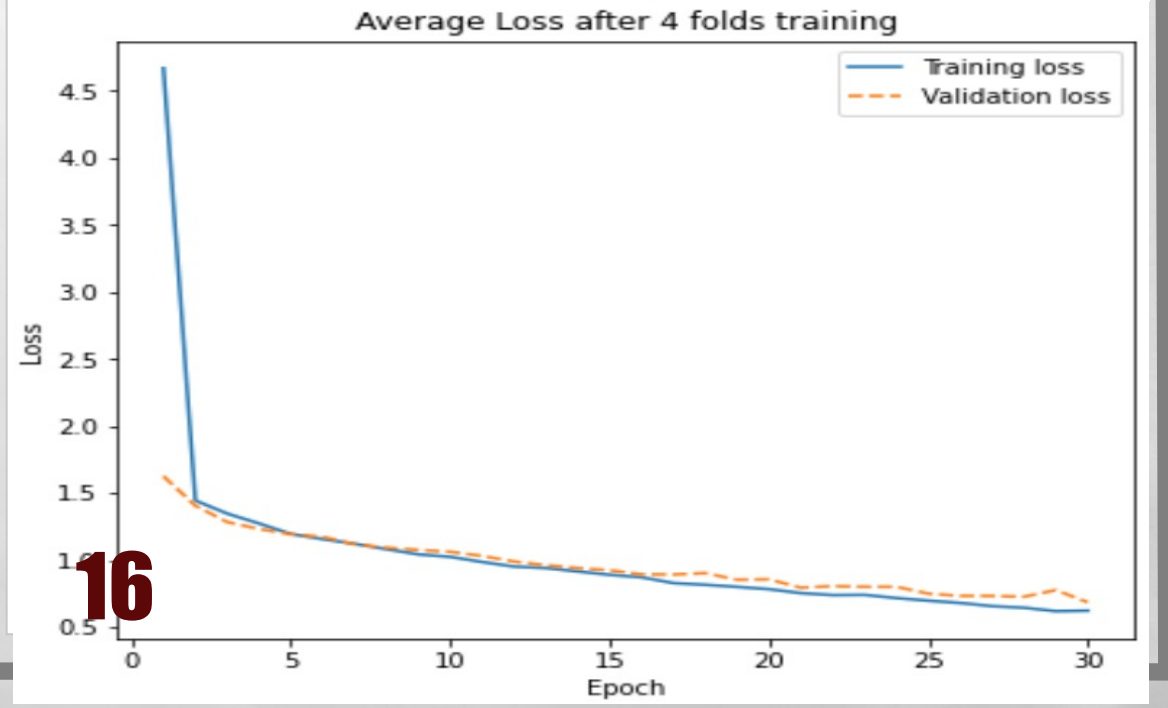

Fold4 Loss
Training loss
Validation loss
Loss
Epoch
5
4
3
2
1
0
5
10
15
20
25
30

FOLD 4 ➔

AVERAGING ALL FOLD LOSS AND ACCURACY

- After averaging all fold loss and accuracy records, finally we got 76.5% train accuracy and 74.5% validation accuracy after 30 epochs training.
- This way we can get rid of Overfitting problem.

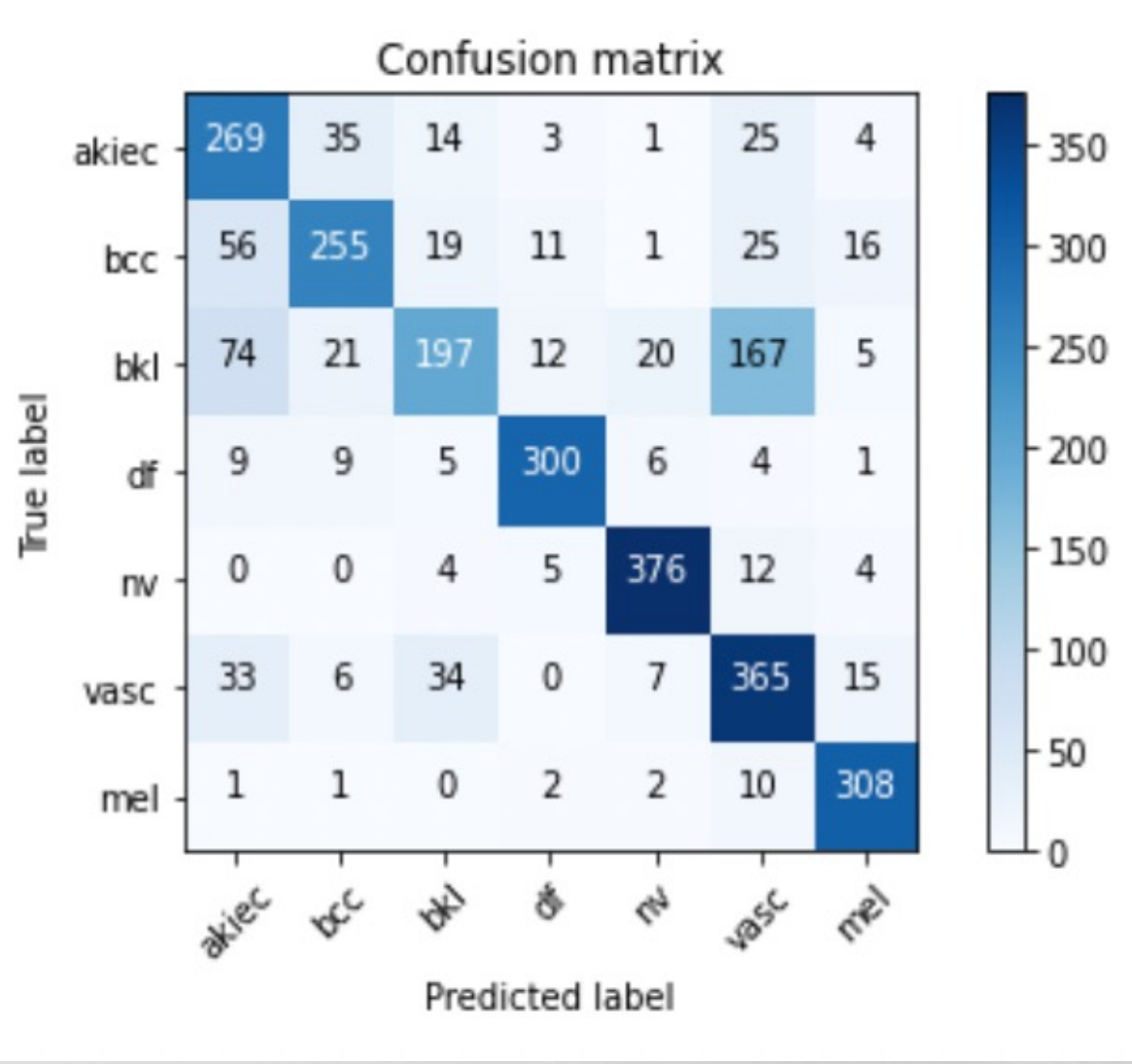




CLASSIFICATION REPORT AND CONFUSION MATRIX

	precision	recall	f1-score	support
akiec	0.61	0.77	0.68	351
bcc	0.78	0.67	0.72	383
bkl	0.72	0.40	0.51	496
df	0.90	0.90	0.90	334
nv	0.91	0.94	0.92	401
vasc	0.60	0.79	0.68	460
mel	0.87	0.95	0.91	324
accuracy			0.75	2749
macro avg	0.77	0.77	0.76	2749
weighted avg	0.76	0.75	0.74	2749

- After trained the model 4 times with different train and test collection, the confusion matrix optimizes. The number of wrongly classified images are becomes less as compared with the result without Kfold. And also, per class score of all class is getting better now.



CONCLUSION

- In this project I am able to understand ability of convolutional neural networks in the classification of skin lesion types.
- From the first trained model result, we have face a problem of overfitting as the number of epochs gets increasing.
- And a new cross validation method called Kfold (with k=4) was performed on the data grouping, that means We perform 4 different training by dividing our data into 4 different parts and using one of those four sets of images as a validation set at a time and use the remaining three sets as a training sets.
- After performing these kfold cross validation techniques, the performance of our model is getting improved and also we can get rid of the overfitting problem.

ANY QUESTION?

THANK YOU!
GRAZIE MILLE!
አመሰግናለሁ!