深度學習於生醫資料分析 作業一

生資所碩一 310112019 黃上瑄

1. Tool Introduction

使用語言	Python 3.9
使用框架	Pytorch 1.10.1
使用套件	Matplotlib, pandas, tqdm, imgaug, numpy, seaborn, sklearn
是否使用 GPU	是(GTX1080)
參考別人的 github	是，https://github.com/eclique/pytorch-gradcam/blob/master/gradcam.ipynb

Table 1. Tool Introduction

2. Dataset Introduction:

a. MURA dataset[1] were used in this report, the original dataset include 7 different body part and bone fraction or not, however, in this report, only 3 body parts' x-ray image were included.

b. Image source: https://stanfordmlgroup.github.io/competitions/mura/

c. Original image size: (Each image size are different, Channel numbers = 3)

d. Detail of image number and distribution:

12941 images were used in this report, distribution were shown in table below

	Training	Validation	Test	Total
Elbow	2048	292	585	2925
Shoulder	2948	421	842	4211
Wrist	4036	576	1153	5765
Total	9032	1289	2580	12941

Table 2. Detail of data distribution

3. workflow:

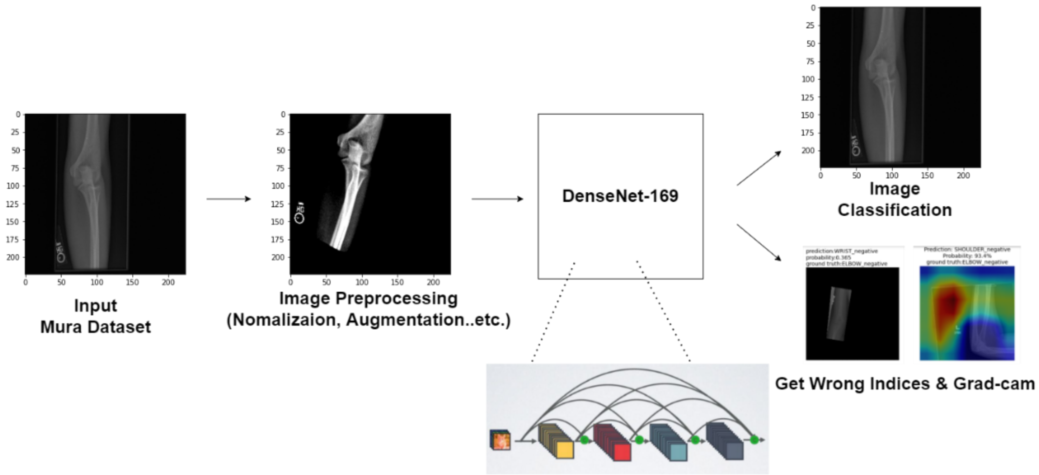

Fig 1. Workflow

In this report, all the data were taken as input at the beginning, after loading the dataset, all the images were resized to 224x224 for matching the convolution channel and decrease the computational complexity for the device.

Image Preprocessing were applied on training data after the images were resized and distributed properly, In image preprocessing part, normalization were take as the first part to increase the performance of model, furthermore, package **imgaug** was used to do data augmentation, in order to decrease the chance of overfitting and increase number of data, for augmentation used in the report, see **Table 3.**

Augmentation	**Parameter**
Flip (horizontal)	0.5
Crop	(0, 0.1)
Gaussian Blur	0.2
Linear Contrast	(0.75, 1,5)
Gaussian Noise	Scale = 0.05* 255 Per_channel = 0.5
Multiply channel (Darker or Brighter)	(0.8, 1.2)
Affine Transformation	Scale: (0.8, 1.2) Translate_percent(0.2, 0.2) Rotate: (-25, 25) Shear: (-8,8)

Table 3. Detail of data augmentation

Once the augmentation part was finished, Dataset were taken into dataloader as a input that the model need. To prevent overfitting, training and validation dataset have been shuffled. In this report, Dense-169[2] was chosen as the main model, for Densenet model structure, see **Fig 2.**

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 6$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 6$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 6$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 12$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 12$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 12$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 24$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 32$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 48$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 16$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 32$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 32$	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Fig 2. Densenet model structure

After training, test dataset was used to check model's performance, in order to have a detail result, a package that can plot confusion matrix was import from **sklearn**, also, in this report, information was extracted from the previous layer before classify layer followed with Grad-Cam, which give a chance to understand what the model learned during training

4. Parameter:

Parameter	
epoch	15
Image Size	224 x 224 x3
Hyperparameter	
Learning Rate	1e-4, divide by 2 for every 5 epochs
Optimizer	Adam
Loss Function	Cross Entropy Loss

Table 4. Training implement

In this report, model was trained for 15 epochs, to get a better result, learning rate was set as 1e-4 at the beginning, divided by 2 for every 5 epochs, decrease the learning rate might help the model to approach the best answer that it can learned.

For the optimizer part, the reason why Adam was choosing is because it is one of the state-of-art optimizers that are often used in classification problem. The reason that Cross Entropy Loss was choosing is the same reason as optimizer.

5. Result:

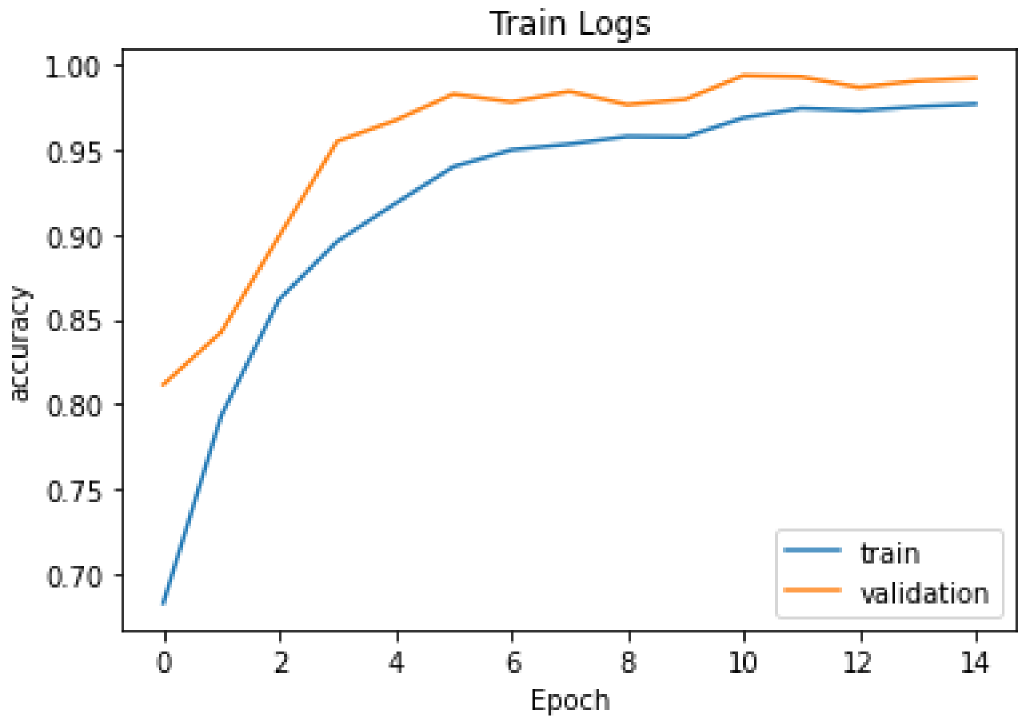

Fig 3. Training and Validation Loss of DenseNet-169

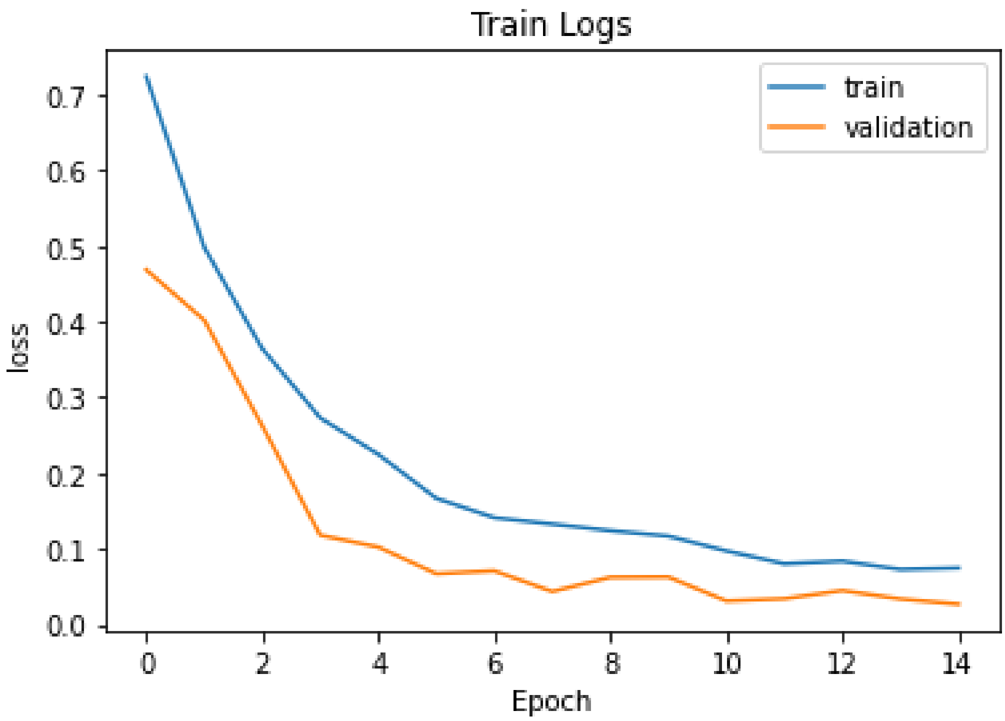

Fig 4. Training and Validation loss of DenseNet-169

Densenet seems to perform well on this classification problem, by looking into **Fig 3.** and **Fig 4.**, we can notice that the accuracy has a high score, 97.71% accuracy for training dataset and 99.92% accuracy for validation dataset.

However, well performance in training and validation dataset doesn't mean the model has learned the feature properly, to check if the model face overfitting or not, test dataset was used to check the model's performance, result of test dataset was shown in **Fig 5.** and the confusion matrix of test dataset was shown in **Fig 6.**

```
testing loss: 0.06210914945316522 testing acc: 0.9810077519379845
```

Fig 5. Testing Loss and Accuracy

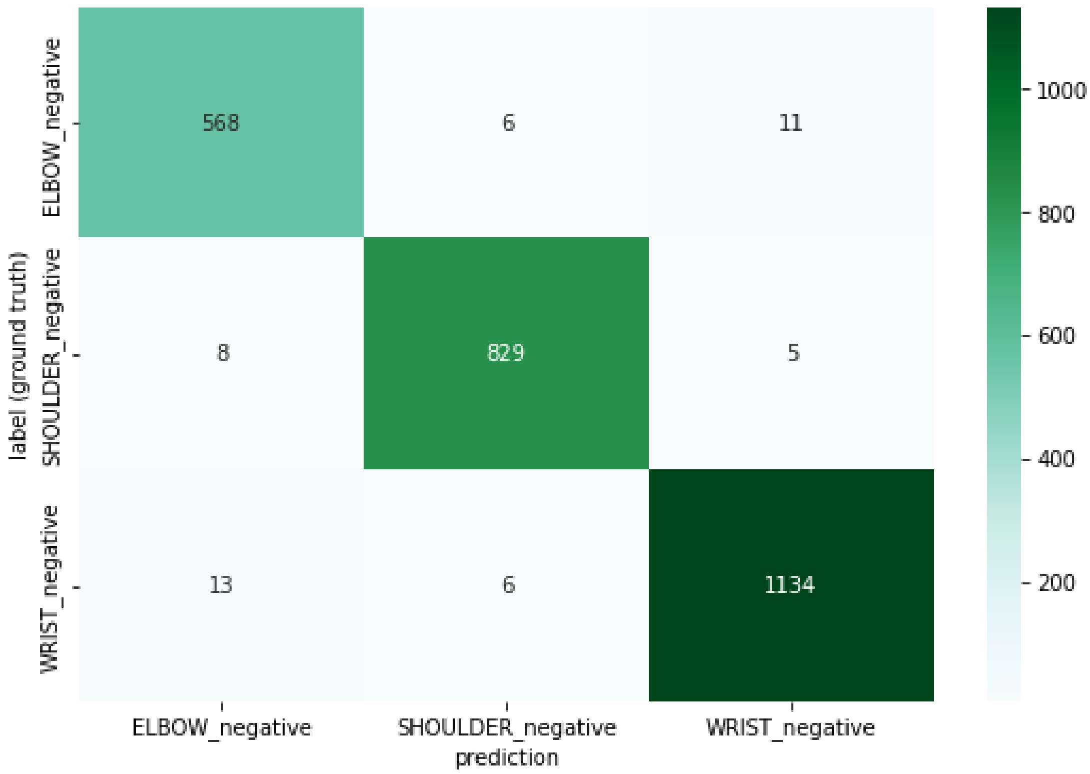

Fig 6. Confusion Matrix of Test dataset

After testing model's performance, we would like to understand the detail of what the model has learned, to approach this goal, first we show some of the wrong indices, images that the model has miss-predicted (**Fig 7.)**, information was then extracted from the previous layer of classify layer. Grad-cam was applied on the image to give chance for us to understand how the model classified image or what did the model has learned during training (**Fig 8.)**.

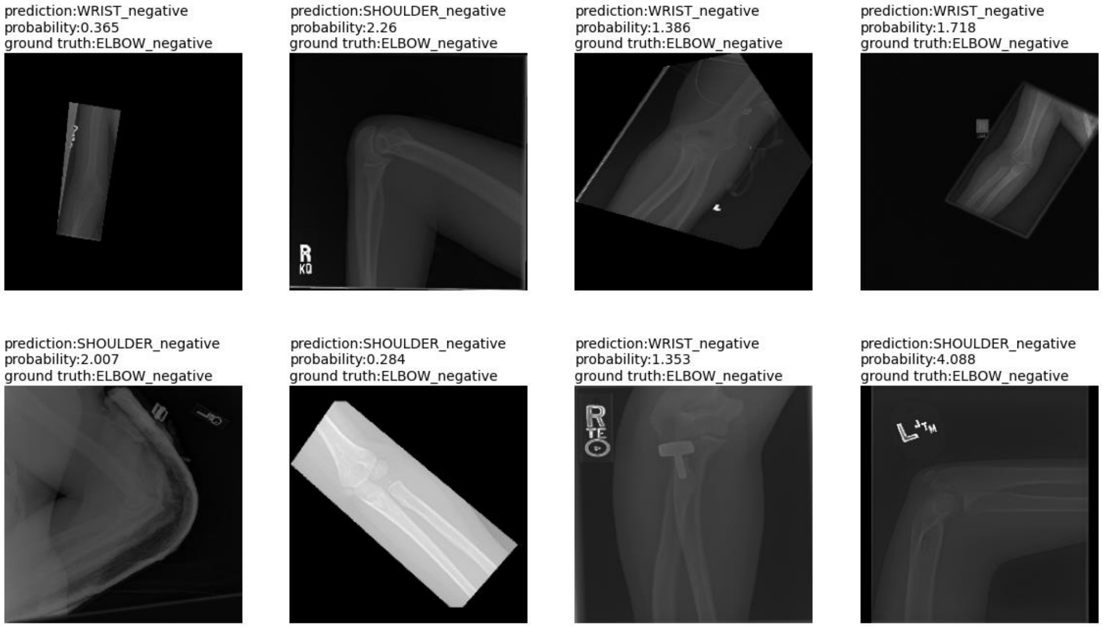

Fig 7. Wrong Indices

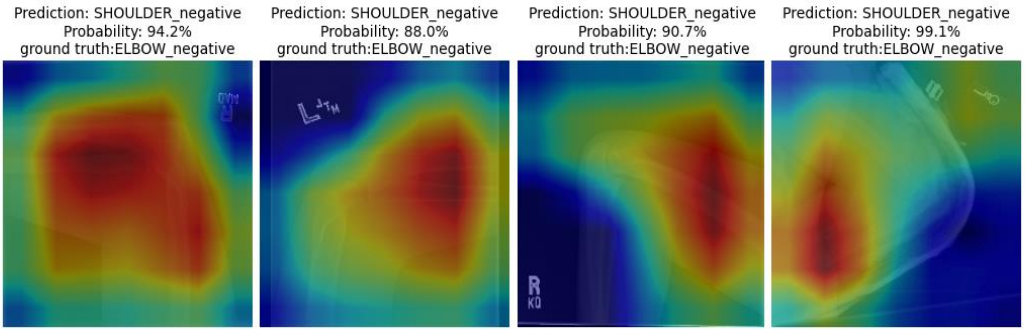

Fig 8. Grad-Cam

While looking into the result of Grad-cam, we can notice that sometimes the model cannot focus on the bone part properly, it might make its prediction by images edge of images or the word in the images, for these kinds of cases, see **Fig 9.**

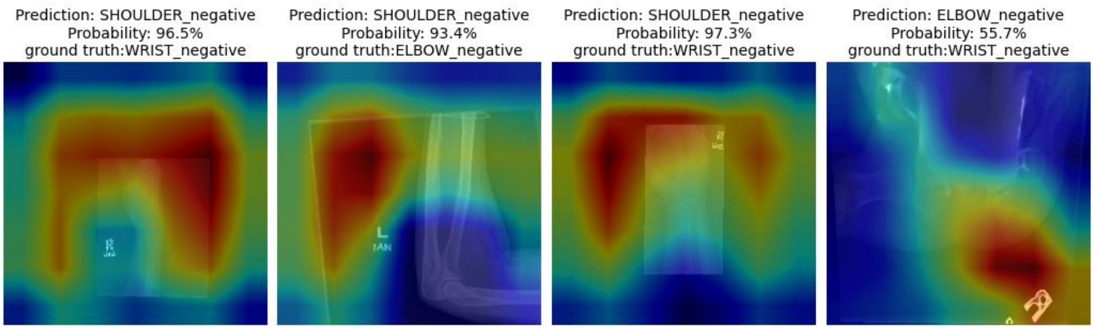

Fig 9. Special Cases

6. Reference:

[1] Mura Dataset:
https://stanfordmlgroup.github.io/competitions/mura/
[2] Densenet:
https://arxiv.org/abs/1608.06993
[3] Deep Residual Learning for Image Recognition:
https://arxiv.org/pdf/1512.03385
[4] Number of training parameters in millions for VGG ResNet and DenseNet model:
https://www.researchgate.net/figure/Number-of-training-parameters-in-millionsM-for-VGG-ResNet-and-DenseNet-models_tbl1_338552250

7. Additional Part:

In additional part, we compare Densenet-169 with another popular model, ResNet50[3], all the training implements were the same.

First, we compare training and validation's training and loss by looking to plot (**Fig 10.** and **Fig 11.**). After the comparison on training and validation dataset were done, test dataset was applied to check how these 2 models perform on the same data (**Table 5.)**, we can notice that their results are roughly the same, however, by the parameter shown in **Fig 12.**[4], we notice that DenseNet-169 only has half of the parameter than ResNet-50, which means DenseNet-169 can reach same result with less resources.

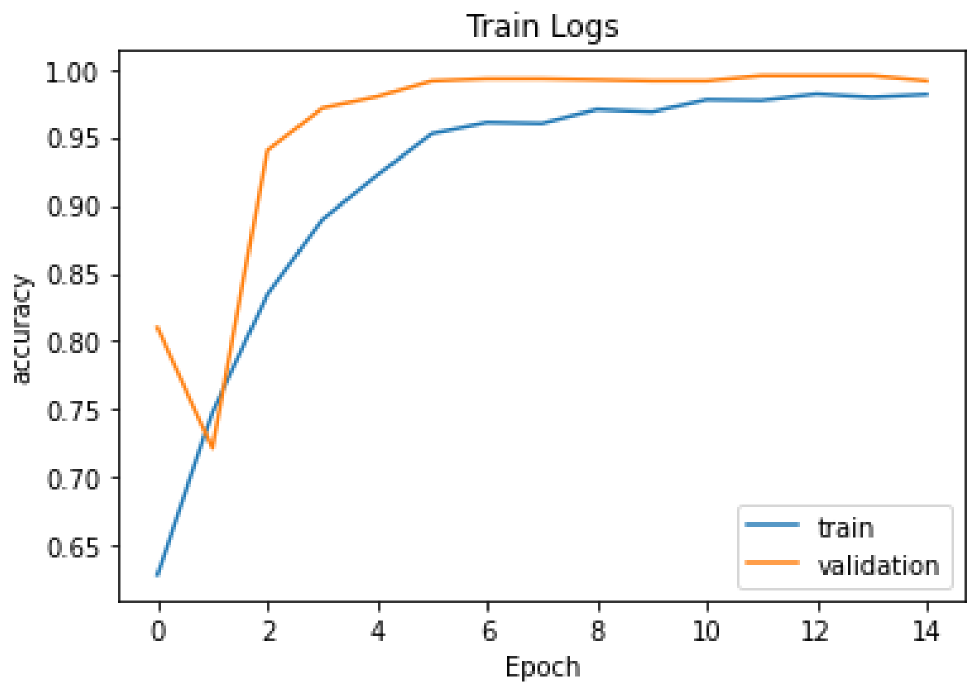

Fig 10. Training and Validation accuracy of ResNet-50

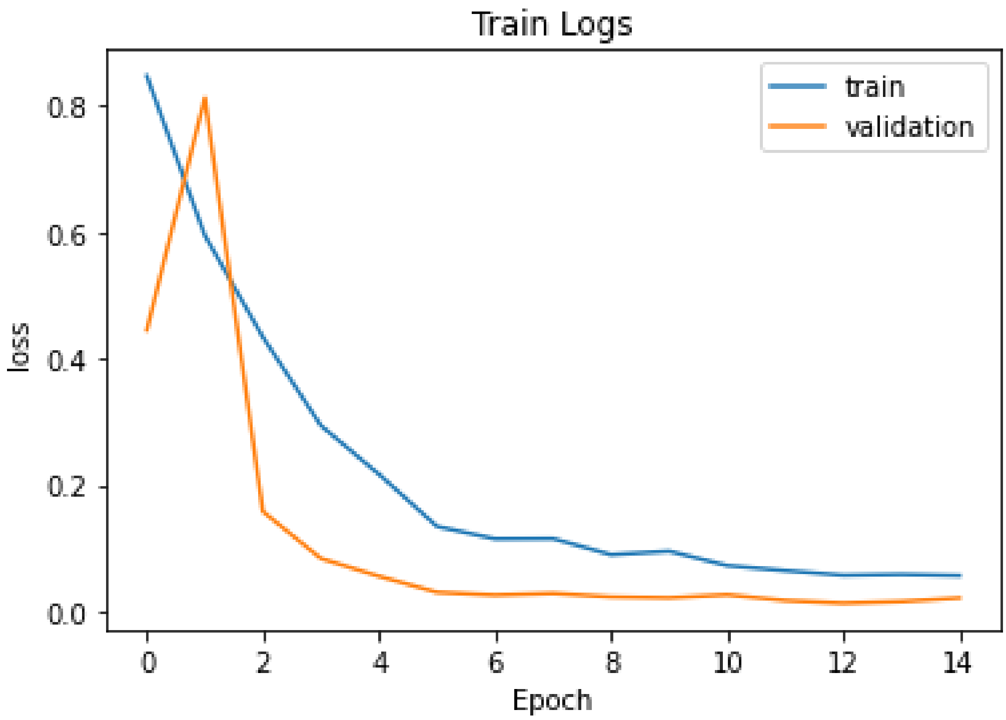

Fig 11. Training and Validation loss of ResNet-50

	Accuracy	Loss
DenseNet-169	0.98100	0.06210
ResNet-50	0.98372	0.04789

Table 5. Comparison between DenseNet-169 and ResNet-50 on test dataset

Model	2D-CNN	3D-CNN	Semi-CNN	
	Params	Params	Pre-Trained Params	Total Params
VGG-16	134.7 M	179.1 M	5.3 M	82.2 M
ResNet-18	11.4 M	33.3 M	0.4 M	31.7 M
ResNet-34	21.5 M	63.6 M	0.8 M	60.5 M
ResNet-50	23.9 M	46.4 M	0.9 M	45.8 M
ResNet-101	42.8 M	85.5 M	0.9 M	84.8 M
ResNet-152	58.5 M	117.6 M	1.4 M	115.6 M
DenseNet-121	7.2 M	11.4 M	0.8 M	10.4 M
DenseNet-169	12.8 M	18.8 M	0.8 M	17.9 M

Fig 12. Parameters Comparison