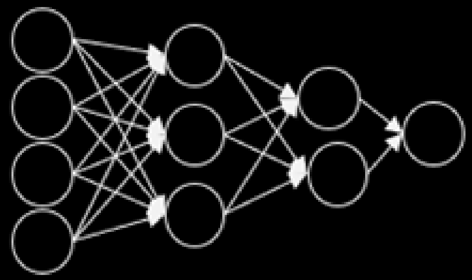
RBS - MITA - Spring 2020



Classifying Melanoma Types from Skin Lesion Images

Submitted by:

Nivetha Vedanarayanan

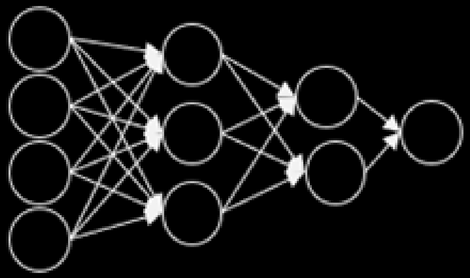
Radhika Rajeevan

Weijun Zhu

Xinran Li

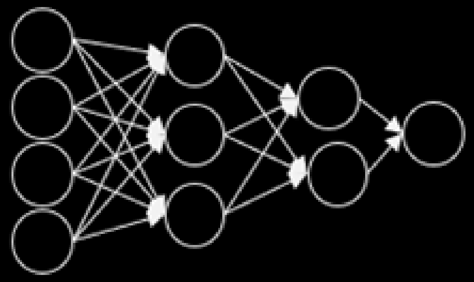
Neural Networks & Deep Learning Project

Advised By : Dr. Farid Alizadeh



Objective

- In this project, we aim to explore the application of Neural Networks in Image Classification for medical diagnosis.
- We are using dermoscopic images of pigmented skin lesions and identifying the melanoma type that it belongs to.



Motivation

- Melanoma are cancerous in nature; however the degree of malignancy differs based on each type.
- In 2015 there were a total of 350,000 cases of skin cancer with 60,000 deaths (17%)
- If diagnosed early, melanoma survival exceeds 95%

Ref- https://challenge2018.isic-archive.com/

Introduction

Data source

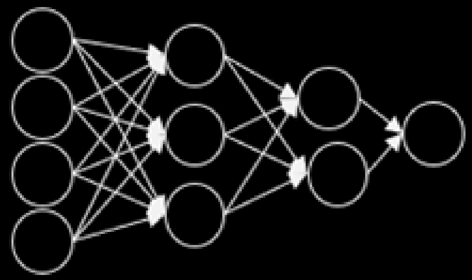
Exploratory Data Analysis

Pre-processing

Approaches

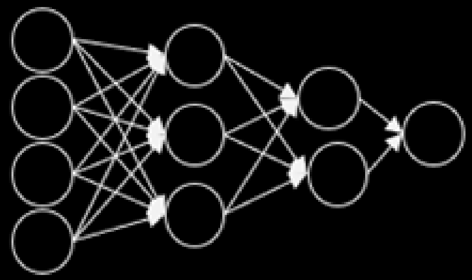
Building CNN
Feature Extraction
Pretrained Model Architecture
Reusing Pretrained Weights

Comparing Models



Data Source & Extraction

- The Skin Lesion Images were made available on Kaggle as part of a Melanoma Detection Challenge by the International Skin Imaging Collaboration (ISIC).
- There are 10,015 dermoscopic colored images classified into 7 different melanoma types.
- We have extracted the data from Kaggle into Google drive using Kaggle API.



EDA Findings

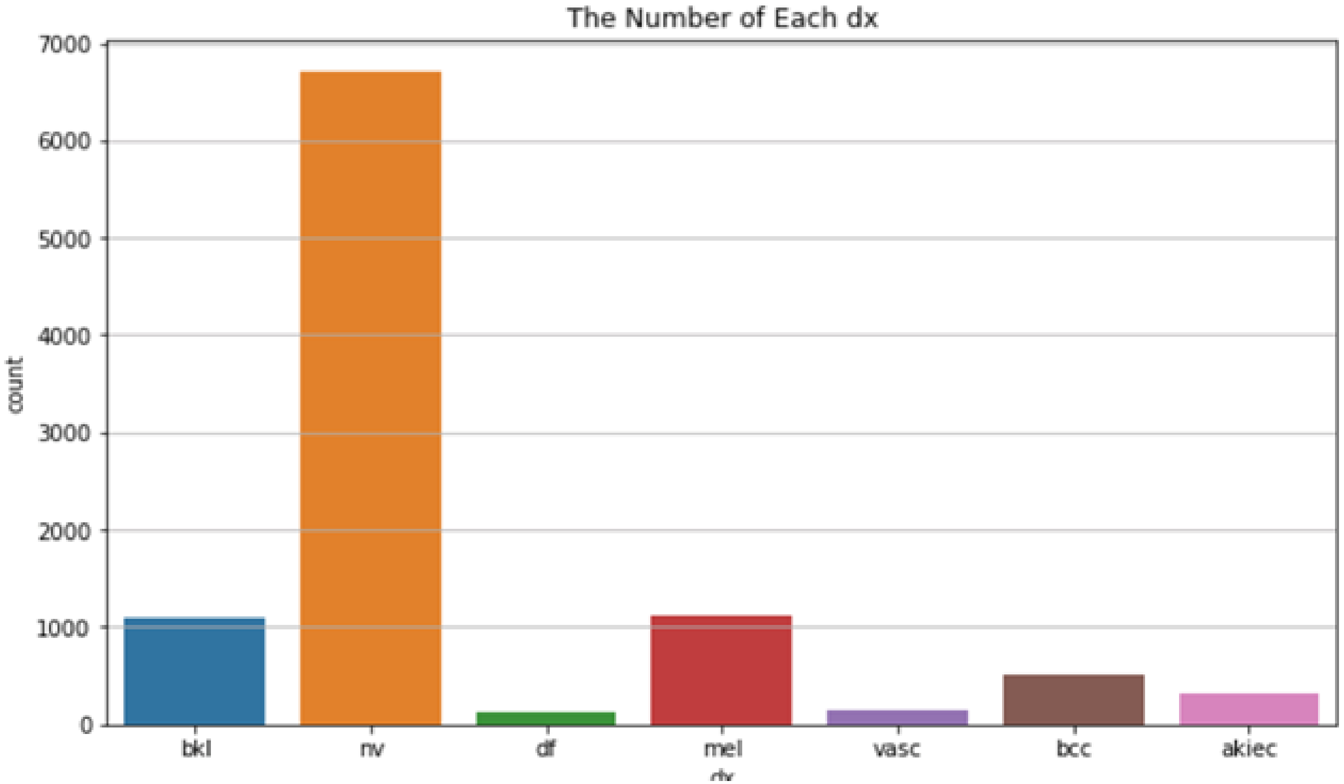
- The Skin Lesion Images were made available on Kaggle as part of a Melanoma Detection Challenge by the International Skin Imaging Collaboration (ISIC).

- There are 10,015 dermoscopic colored images classified into 7 different melanoma types.

- We have extracted the data from Kaggle into Google drive using Kaggle API.

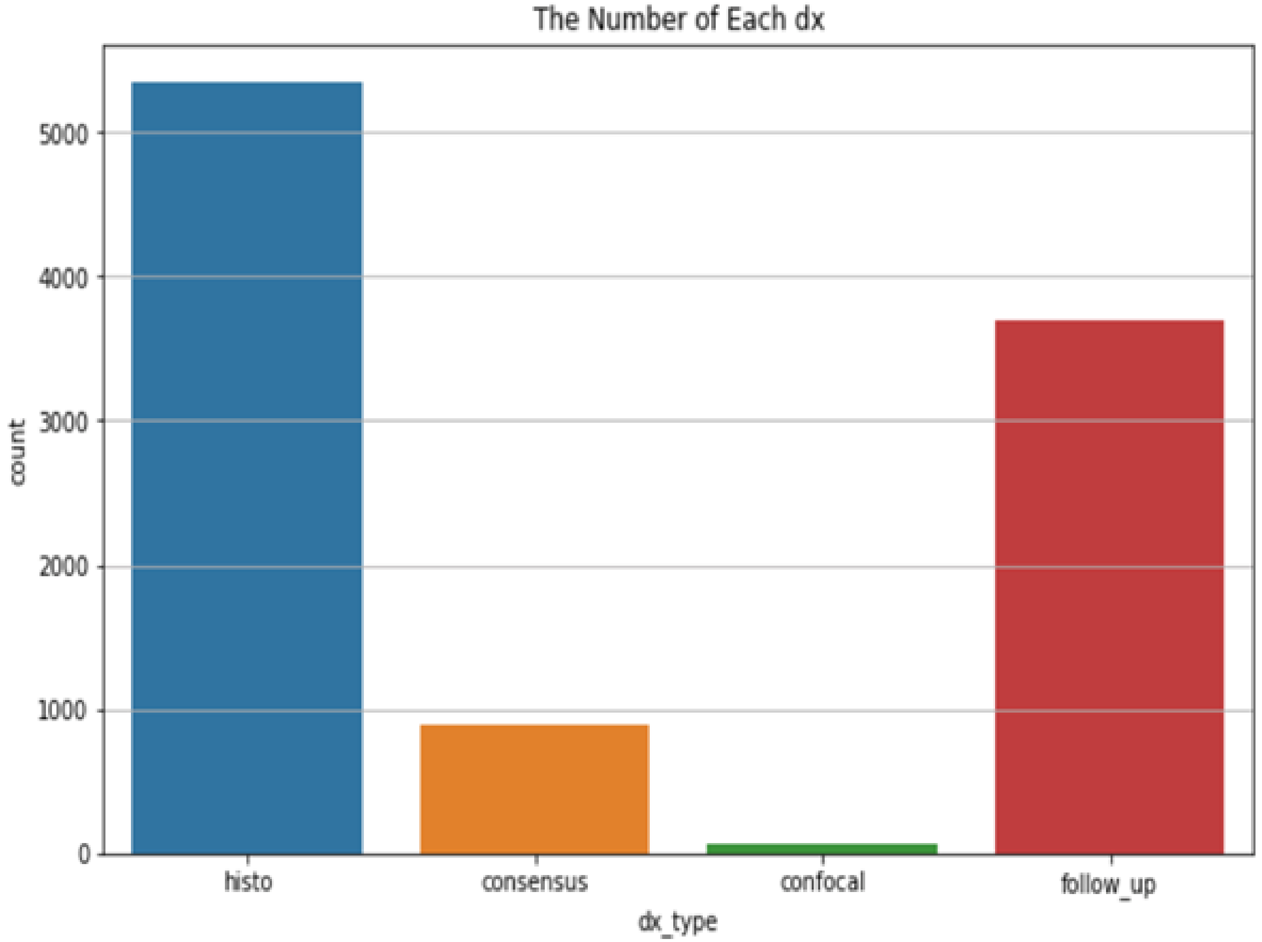
df feature

```
nv       6705
mel      1113
bkl      1099
bcc       514
akiec     327
vasc      142
df        115
Name: dx, dtype: int64
```



- By visualizing data, we can tell that the “nv” type is the most common melanoma type. The quantity of “nv” exceed over ½ of the overall collected images.
- The other melanoma types are: “bkl”, “df”, “mel”, “vasc”, “bcc”, “akiec”

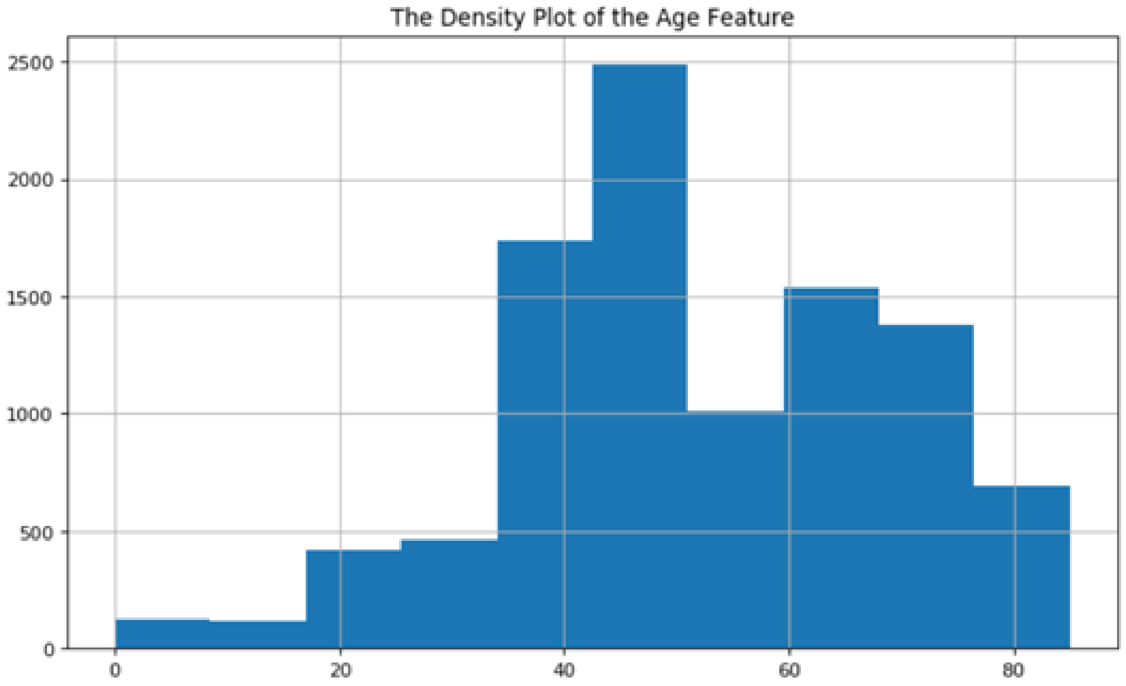
```
histo          5340
follow_up      3704
consensus       902
confocal         69
Name: dx_type, dtype: int64
```

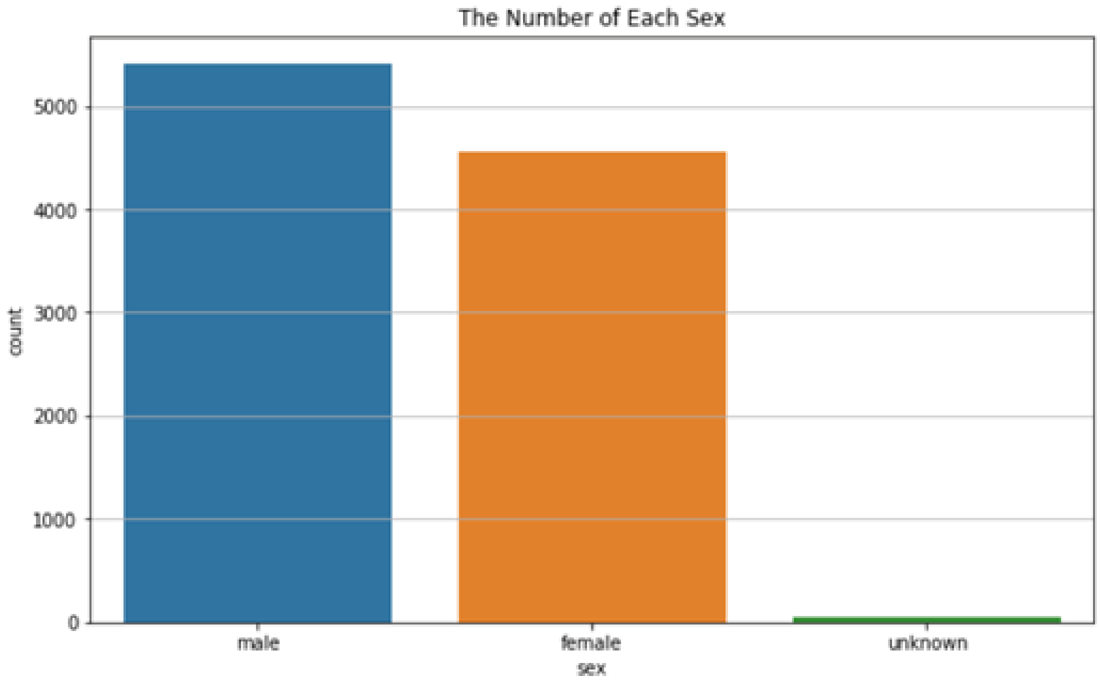


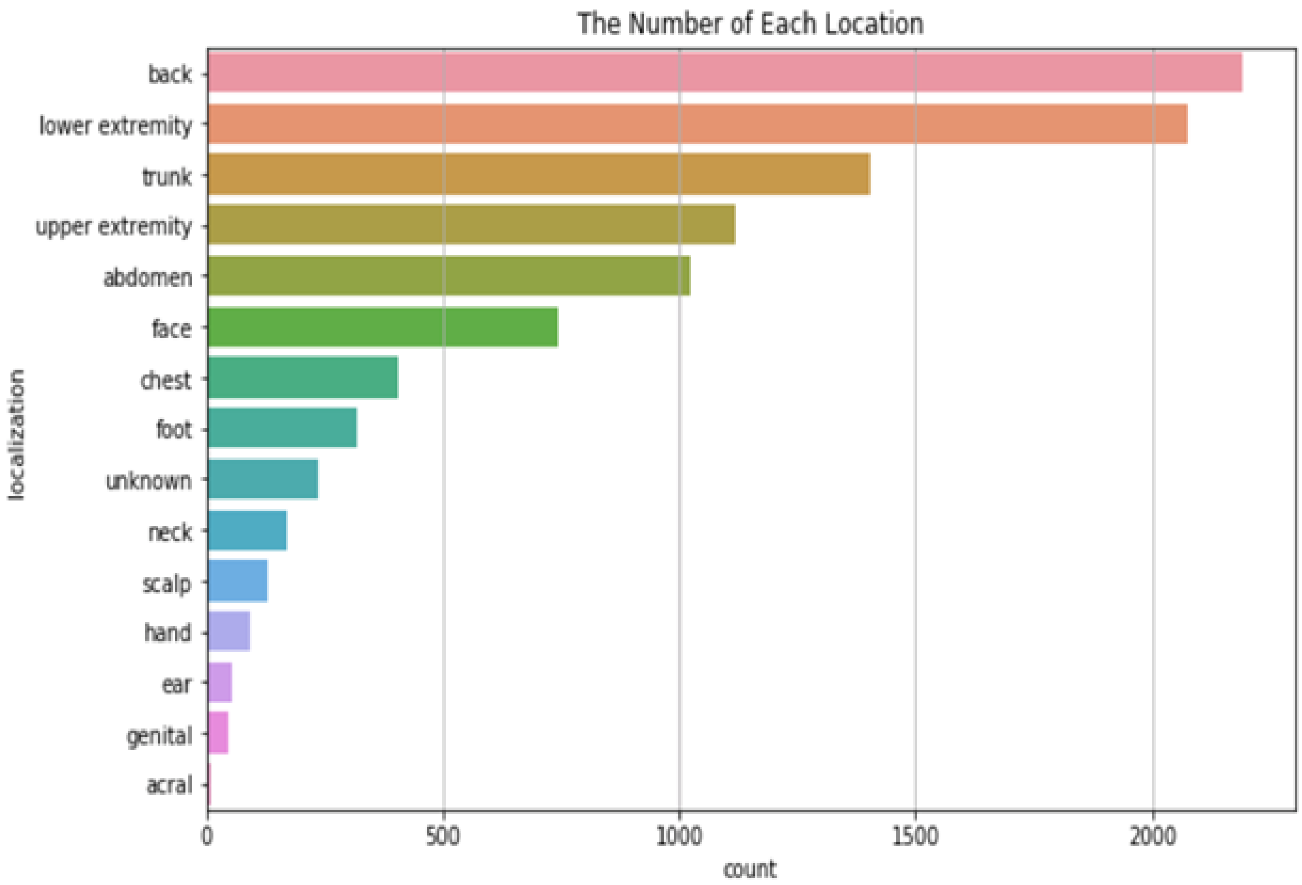
dx type feature

- We notice from the visualization that over 50% of lesions are confirmed through “histo”.
- While “follow_up” play another major role in confirming cases, we could also confirmed by “consensus” and “confocal”.

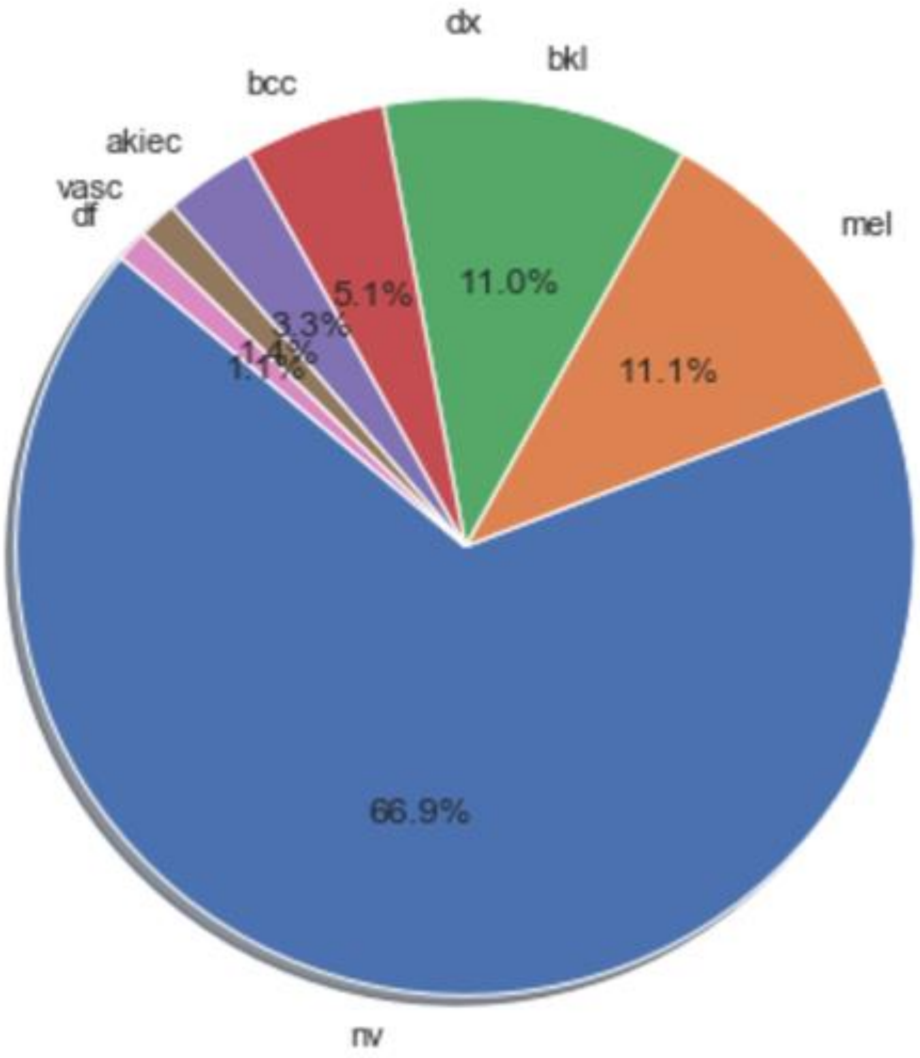
Some other features… …

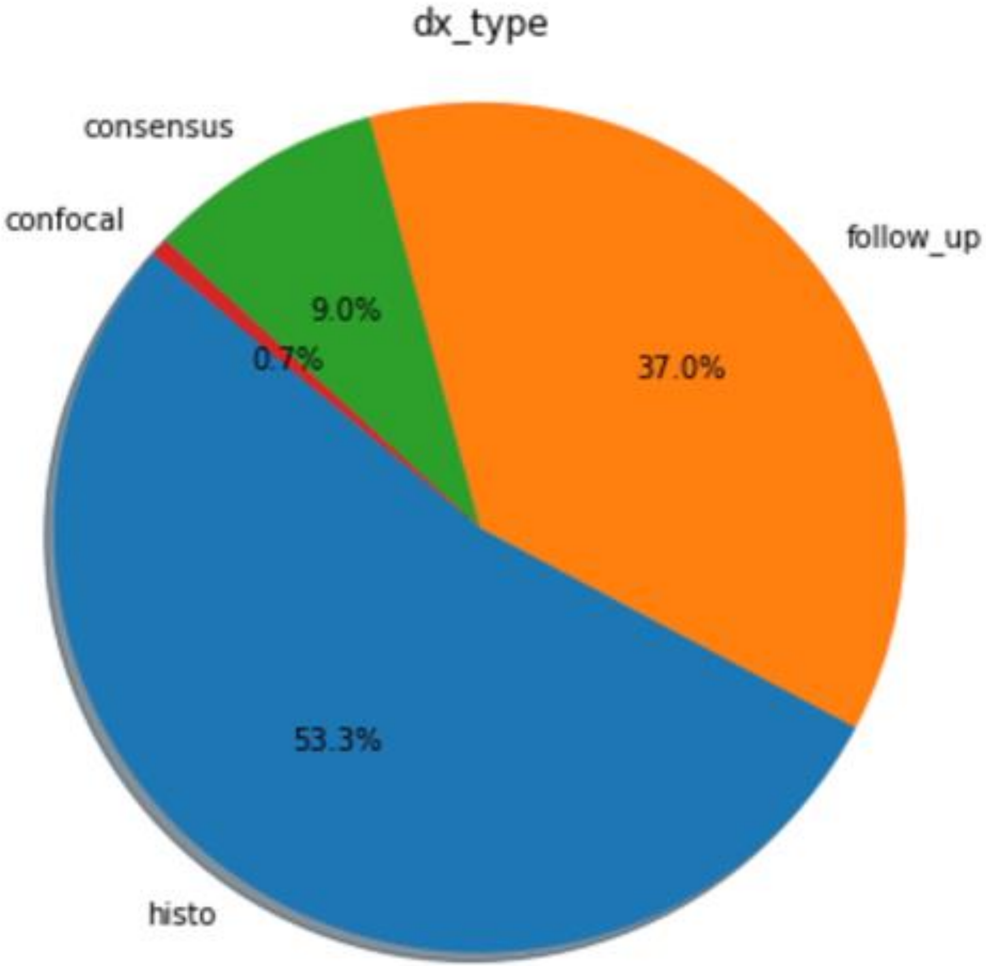




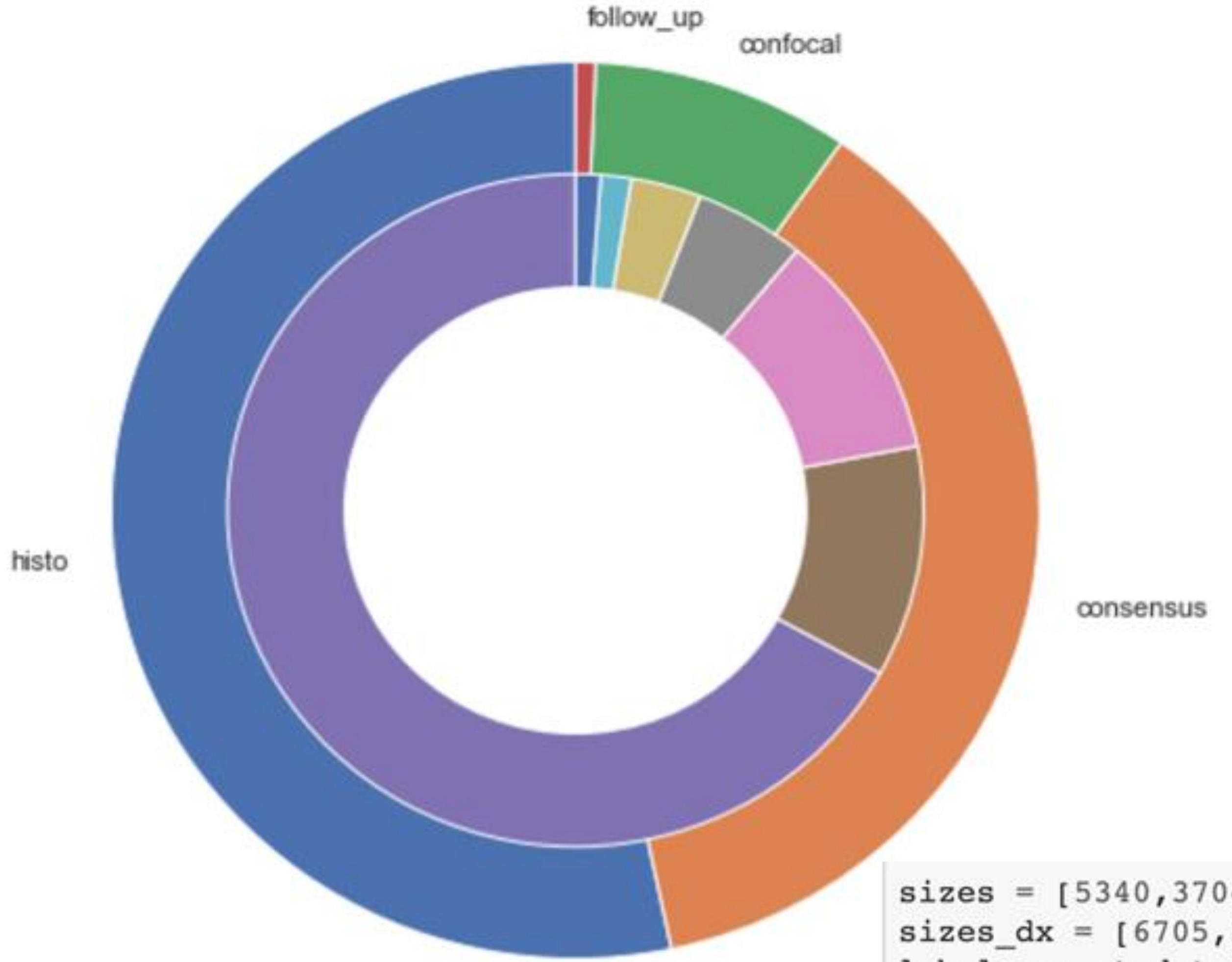


Another view for the percentage between dx and dx_type



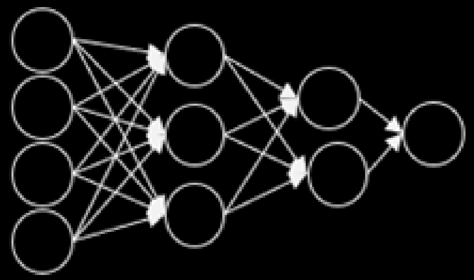


Relationship between dx and dx_type (How & Where)



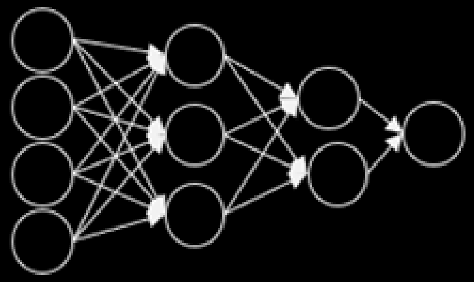

```
sizes = [5340,3704,902,69]
sizes_dx = [6705,1113,1099,514,327,142,115]
labels = metadata["dx_type"].unique()

plt.figure(figsize=(9,6))
plt.pie(sizes, labels=labels, startangle=90,frame=True)
plt.pie(sizes_dx,radius=0.75,startangle=90)
centre_circle = plt.Circle((0,0),0.5,color='black', fc='white',linewidth=0)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
```



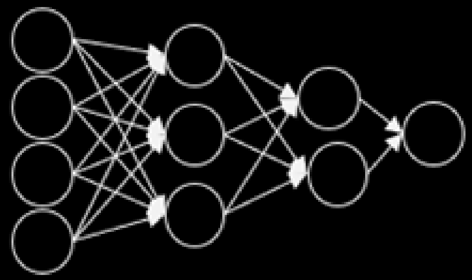
Preprocessing

- Scaling Data
  - Images were converted to numeric arrays and scaled between 0 and 1 and saved as npy files.
  - We have tried models with 3 different dimensions:
    - 128 x 128 x 3
    - 150 x 150 x 3
    - 224 x 224 x 3
- Class Imbalance
  - Since class proportions are highly skewed, we have used Data Augmentation to generate rotated images for classes with less training data for a class-wise balance.



MobileNet

- MobileNet with 128 x 128 x 3 arrays
  - Train-Val-Test Split
  - Model Architecture
  - Learning Rate
  - Accuracy & Loss
  - Performance on Test Data
  - Confusion Matrix
  - Precision & Recall



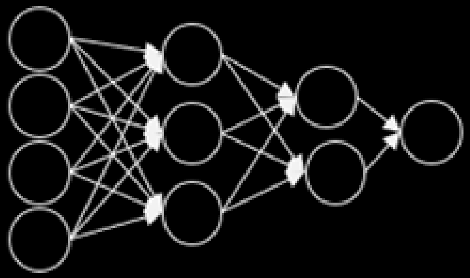
MobileNet Architecture

- Freeze the pretrained MobileNet weights

```
Model: "mobilenet_1.00_128"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         (None, 128, 128, 3)       0
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 129, 129, 3)       0
_________________________________________________________________
conv1 (Conv2D)               (None, 64, 64, 32)        864
_________________________________________________________________
conv1_bn (BatchNormalization (None, 64, 64, 32)        128
_________________________________________________________________
conv1_relu (ReLU)            (None, 64, 64, 32)        0
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, 64, 64, 32)        288
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, 64, 64, 32)        128
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, 64, 64, 32)        0
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, 64, 64, 64)        2048
_________________________________________________________________
conv_pw_1_bn (BatchNormaliza (None, 64, 64, 64)        256
_________________________________________________________________
conv_pw_1_relu (ReLU)        (None, 64, 64, 64)        0
_________________________________________________________________
```

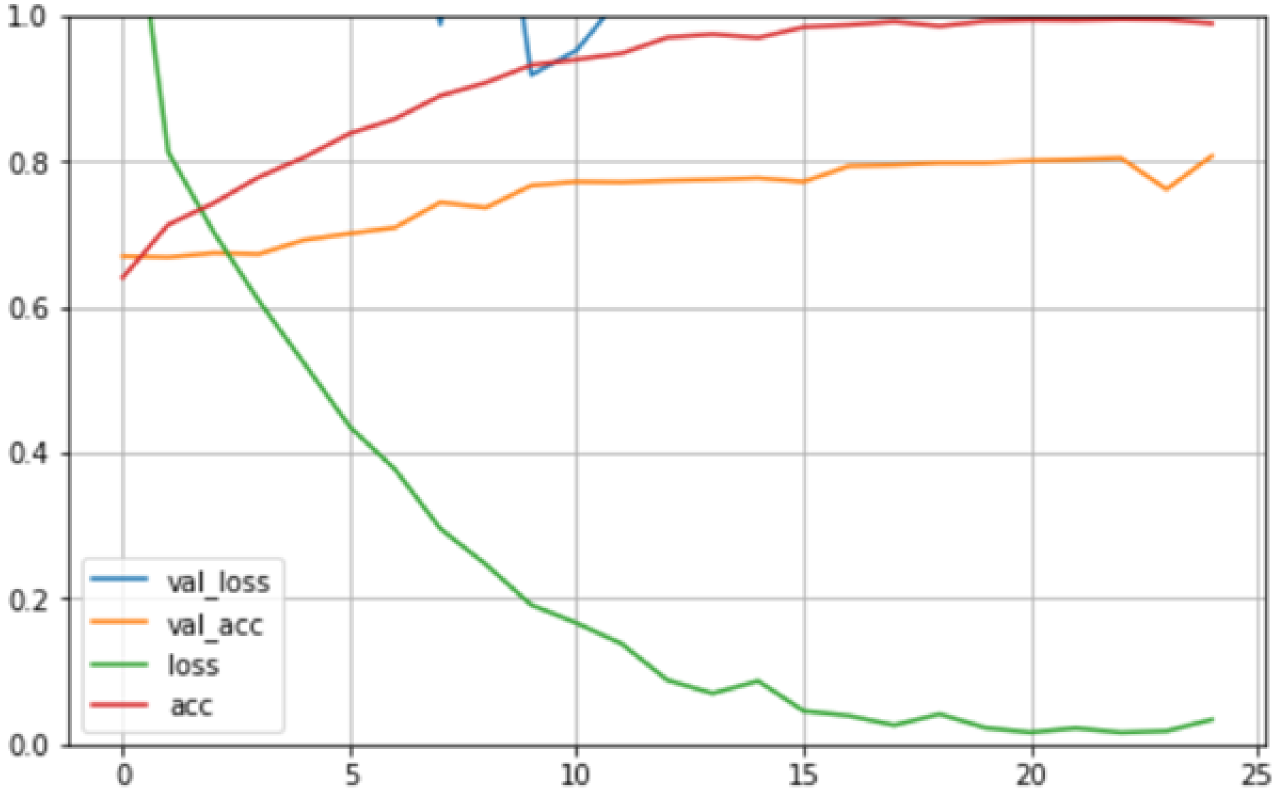
```python
print('Total weights : ',len(model_mobilenet.weights))
print('Trainable weights : ',len(model_mobilenet.trainable_weights))
```

```
Total weights :  143
Trainable weights :  8
```



MobileNet Performance

- Learning Rate of 0.0001
- Optimizer - RMSProp



- The validation loss is > 1 and does not drop
- Validation accuracy does not improve beyond 80%

MobileNet Test Data

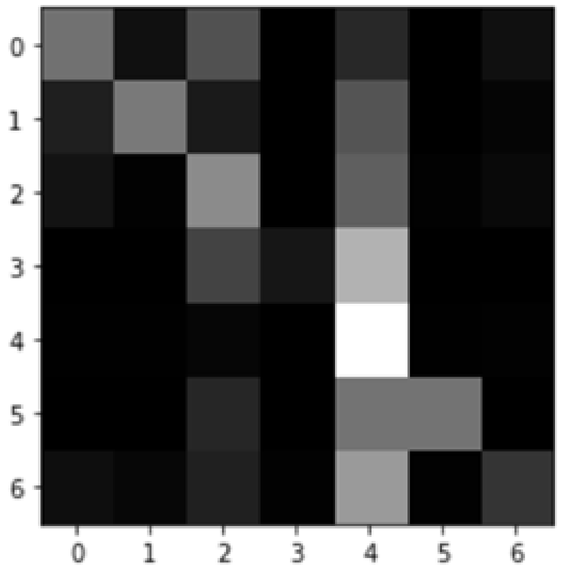
```
# test accuracy
test_loss, test_acc = model_mobilenet.evaluate(mat_data_test,mat_label_test)
print('Test Loss : ',test_loss,' Test Acc : ',test_acc)
```

```
1002/1002 [==============================] - 3s 3ms/step
Test Loss :  0.9709897623239401  Test Acc :  0.7614770531654358
```

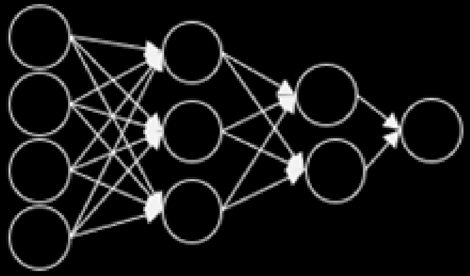
	Class	Acc
0	akiec	0.424242
1	bcc	0.450980
2	bkl	0.518182
3	df	0.083333
4	nv	0.953800
5	vasc	0.428571
6	mel	0.198198

	akiec	bcc	bkl	df	nv	vasc	mel
akiec	14	2	10	0	5	0	2
bcc	6	23	5	0	16	0	1
bkl	8	1	57	0	39	1	4
df	0	0	3	1	8	0	0
nv	3	3	17	0	640	3	5
vasc	0	0	2	0	6	6	0
mel	6	3	14	1	64	1	22

<matplotlib.image.AxesImage at 0x7

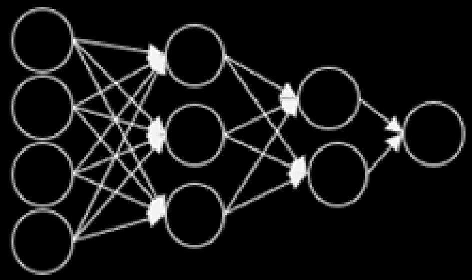


- Identification of 'df' is extremely low : 8%
- Except 'nv' rest of classes have an accuracy of <= 50%



MobileNet with Data Aug

- MobileNet with Data Augmentation
  - Images are Generated on Imbalanced class images
  - Model Architecture
  - Learning Rate
  - Accuracy & Loss
  - Performance on Test Data
  - Confusion Matrix
  - Precision & Recall



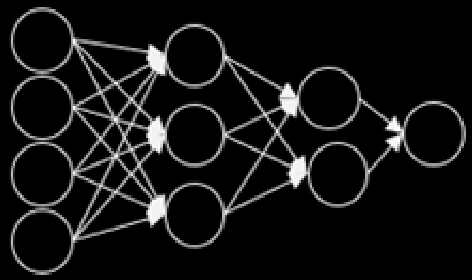
MobileNet-DA Architecture

- Freeze the pretrained MobileNet weights

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_128 (Model)   (None, 4, 4, 1024)        3228864
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16384)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              16778240
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               131200
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 455
=================================================================
Total params: 20,147,015
Trainable params: 20,125,127
Non-trainable params: 21,888
_________________________________________________________________
```

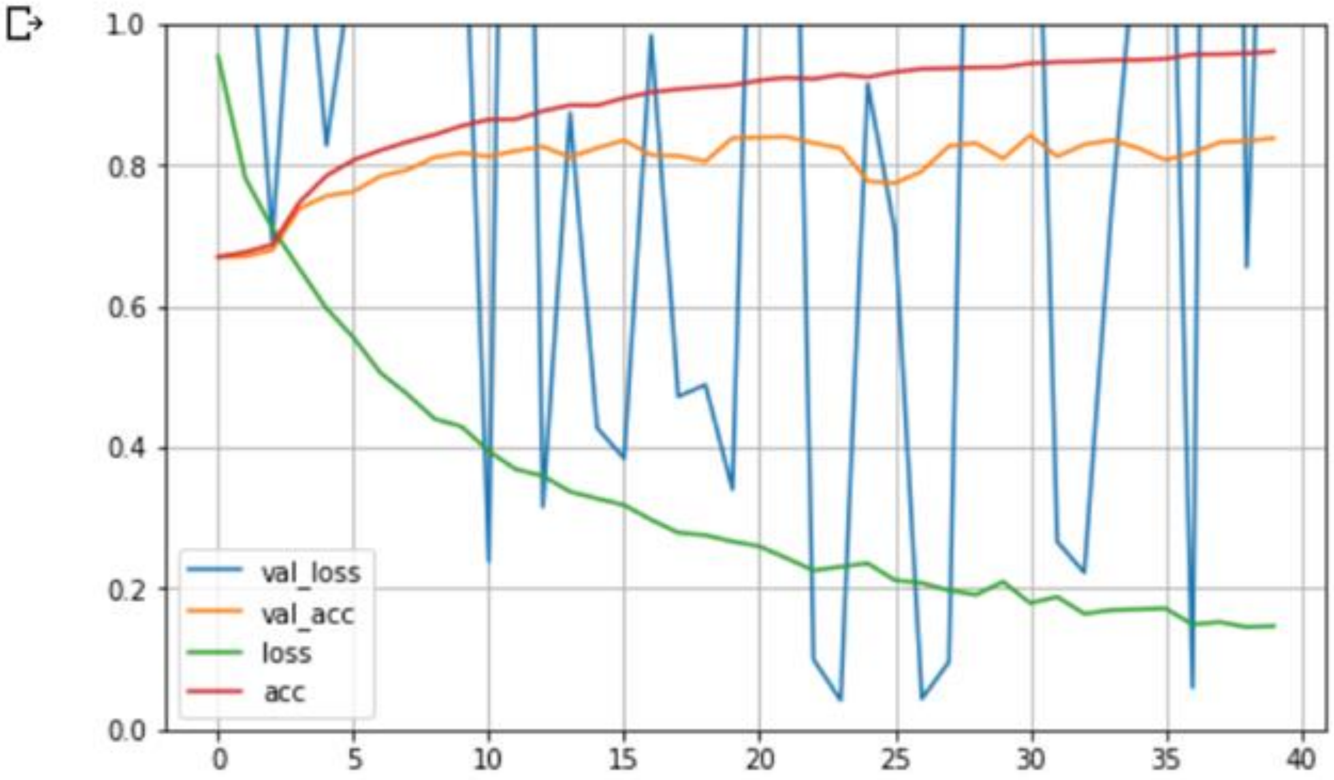
```
[ ] print('Total weights : ',len(model_mobilenet.weights))
    print('Trainable weights : ',len(model_mobilenet.trainable_weights))
    print('Non Trainable weights : ',len(model_mobilenet.non_trainable_weights))
```

```
Total weights :  143
Trainable weights :  89
Non Trainable weights :  54
```



MobileNet-DA Performance

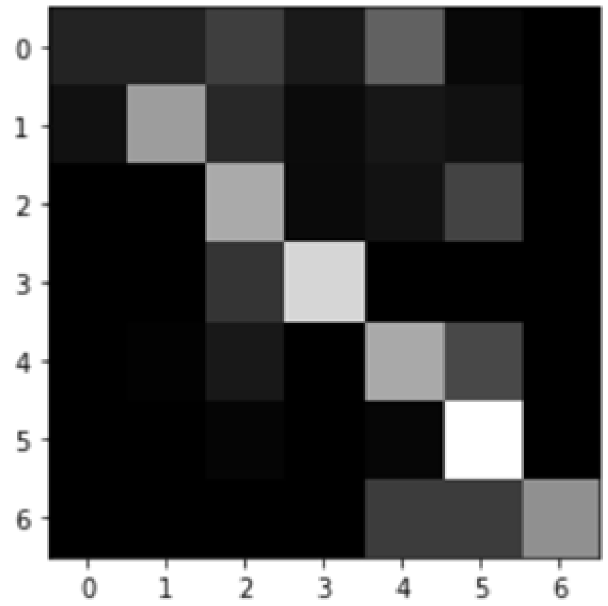
- Learning Rate of 0.0001
- Optimizer – RMSProp
- steps_per_epoch=(train_size/batch_size)



- High fluctuation in validation loss
- Validation accuracy improves slightly beyond 80%

MobileNet-DA Test Data

<matplotlib.image.AxesImage at 0x



```
# test accuracy
test_loss, test_acc = model_mobilenet.evaluate_generator(test_generator, steps=50)
print('Test Loss : ',test_loss,' Test Acc : ',test_acc)
```

```
Test Loss :  2.1438980102539062  Test Acc :  0.8329097628593445
```

	Class	Acc
0	akiec	0.133333
1	bcc	0.586957
2	bkl	0.636364
3	df	0.800000
4	nv	0.630000
5	vasc	0.953642
6	mel	0.538462

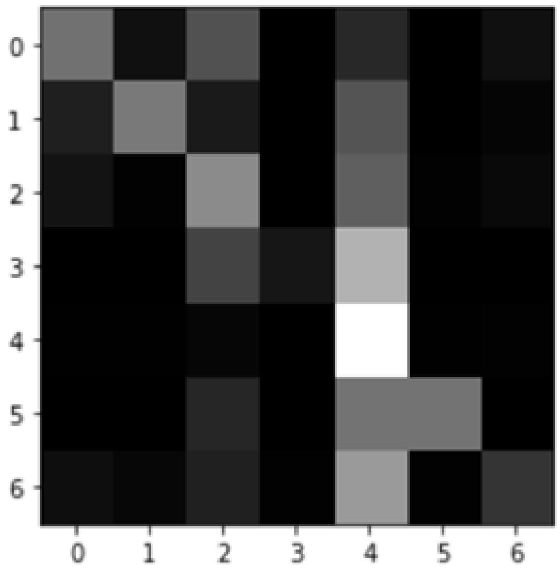
	akiec	bcc	bkl	df	nv	vasc	mel
akiec	4	4	7	3	11	1	0
bcc	3	27	7	2	4	3	0
bkl	0	0	63	4	7	25	0
df	0	0	2	8	0	0	0
nv	0	1	9	0	63	27	0
vasc	0	1	12	1	14	576	0
mel	0	0	0	0	3	3	7

- Identification of ‘df’ is extremely low : 8%
- Except ‘nv’ rest of classes have an accuracy of <= 50%

Downgrades

- Accuracy for 'akiec' has dropped from 42% to 13%
- 'nv' has dropped from 95% to 63%

<matplotlib.image.AxesImage at 0x7



	Class	Acc
0	akiec	0.424242
1	bcc	0.450980
2	bkl	0.518182
3	df	0.083333
4	nv	0.953800
5	vasc	0.428571
6	mel	0.198198

	akiec	bcc	bkl	df	nv	vasc	mel
akiec	14	2	10	0	5	0	2
bcc	6	23	5	0	16	0	1
bkl	8	1	57	0	39	1	4
df	0	0	3	1	8	0	0
nv	3	3	17	0	640	3	5
vasc	0	0	2	0	6	6	0
mel	6	3	14	1	64	1	22

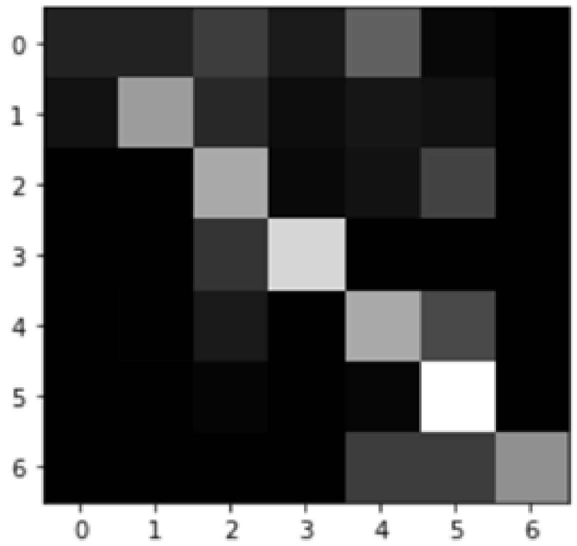
Improvements

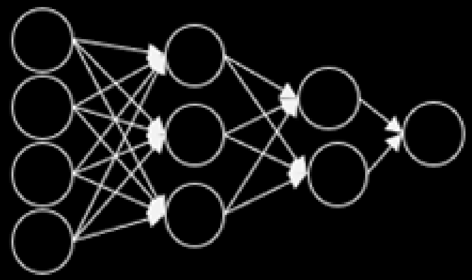
- Identification of 'df' has improved from 8% to 80%
- Fewer misclassifications

	Class	Acc
0	akiec	0.133333
1	bcc	0.586957
2	bkl	0.636364
3	df	0.800000
4	nv	0.630000
5	vasc	0.953642
6	mel	0.538462

	akiec	bcc	bkl	df	nv	vasc	mel
akiec	4	4	7	3	11	1	0
bcc	3	27	7	2	4	3	0
bkl	0	0	63	4	7	25	0
df	0	0	2	8	0	0	0
nv	0	1	9	0	63	27	0
vasc	0	1	12	1	14	576	0
mel	0	0	0	0	3	3	7

<matplotlib.image.AxesImage at 0x





MobileNet-DA with Balanced Class

- MobileNet with Data Augmentation
  - Images are Generated on Imbalanced class images
  - Model Architecture
  - Learning Rate
  - Accuracy & Loss
  - Performance on Test Data
  - Confusion Matrix
  - Precision & Recall

PART IV
ResNet50

Weijun Zhu

Residual Neural Network

Deeper neural networks are more difficult to train, and sometimes researchers find the deeper neural network get the worse results than the shallower one.

ResNet was found by Kaimin in 2015, and it is good to fit the model in the deeper neural network.

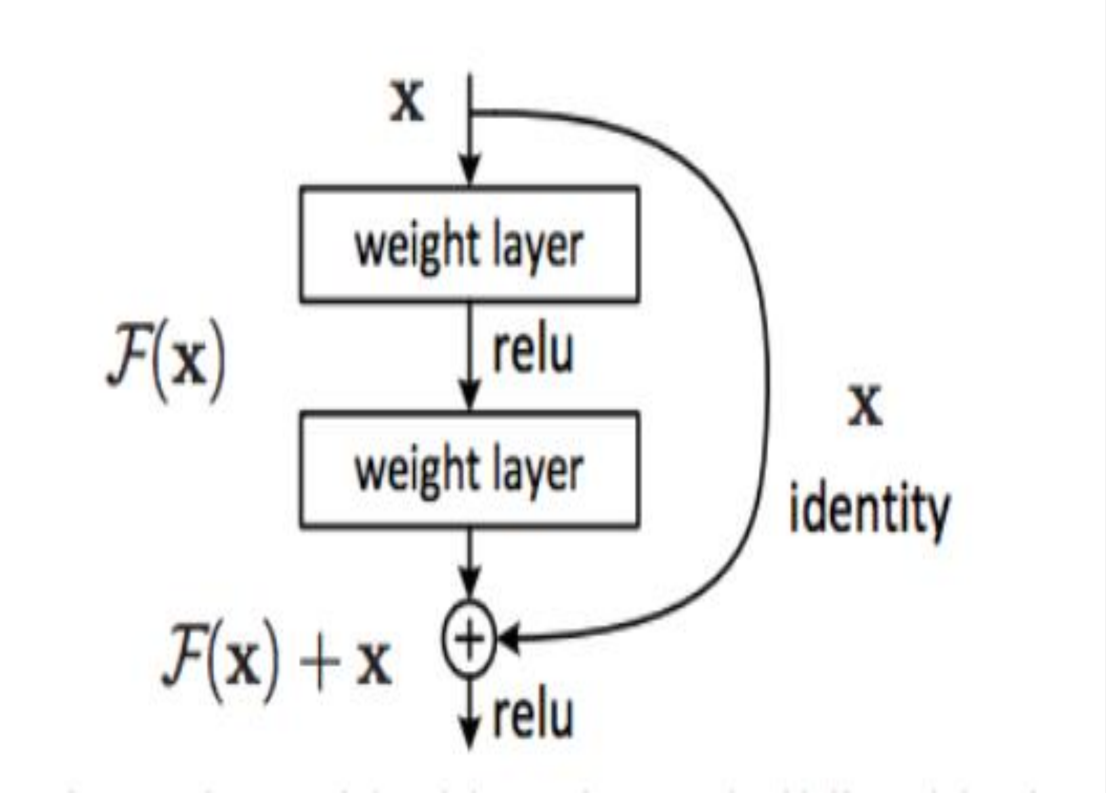
They explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions.

Residual Neural Network

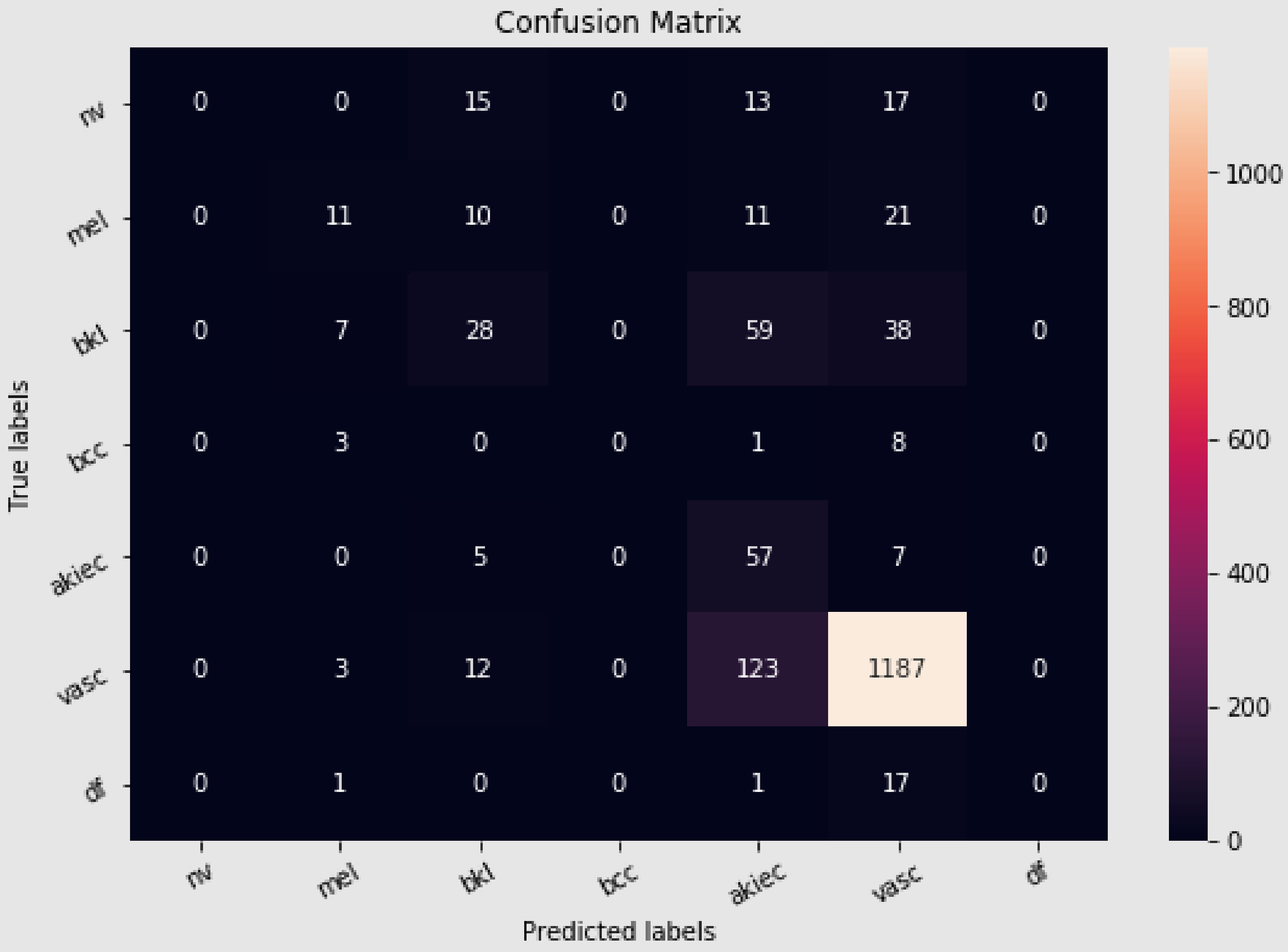
```python
import torch.nn as nn
import torch
from torch.nn.init import kaiming_normal, constant

class BasicConvResBlock(nn.Module):
    def __init__(self, input_dim=128, n_filters=256, kernel_size=3, padding=1,
                 stride=1, shortcut=False, downsample=None):
        super(BasicConvResBlock, self).__init__()
        self.downsample = downsample
        self.shortcut = shortcut
        self.conv1 = nn.Conv1d(input_dim, n_filters, kernel_size=kernel_size,
                               padding=padding, stride=stride)
        self.bn1 = nn.BatchNorm1d(n_filters)
        self.relu = nn.ReLU()
        self.conv2 = nn.Conv1d(n_filters, n_filters, kernel_size=kernel_size,
                               padding=padding, stride=stride)
        self.bn2 = nn.BatchNorm1d(n_filters)
    def forward(self, x):
        residual = x
        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)
        out += residual
        out = self.relu(out)
        return out
```

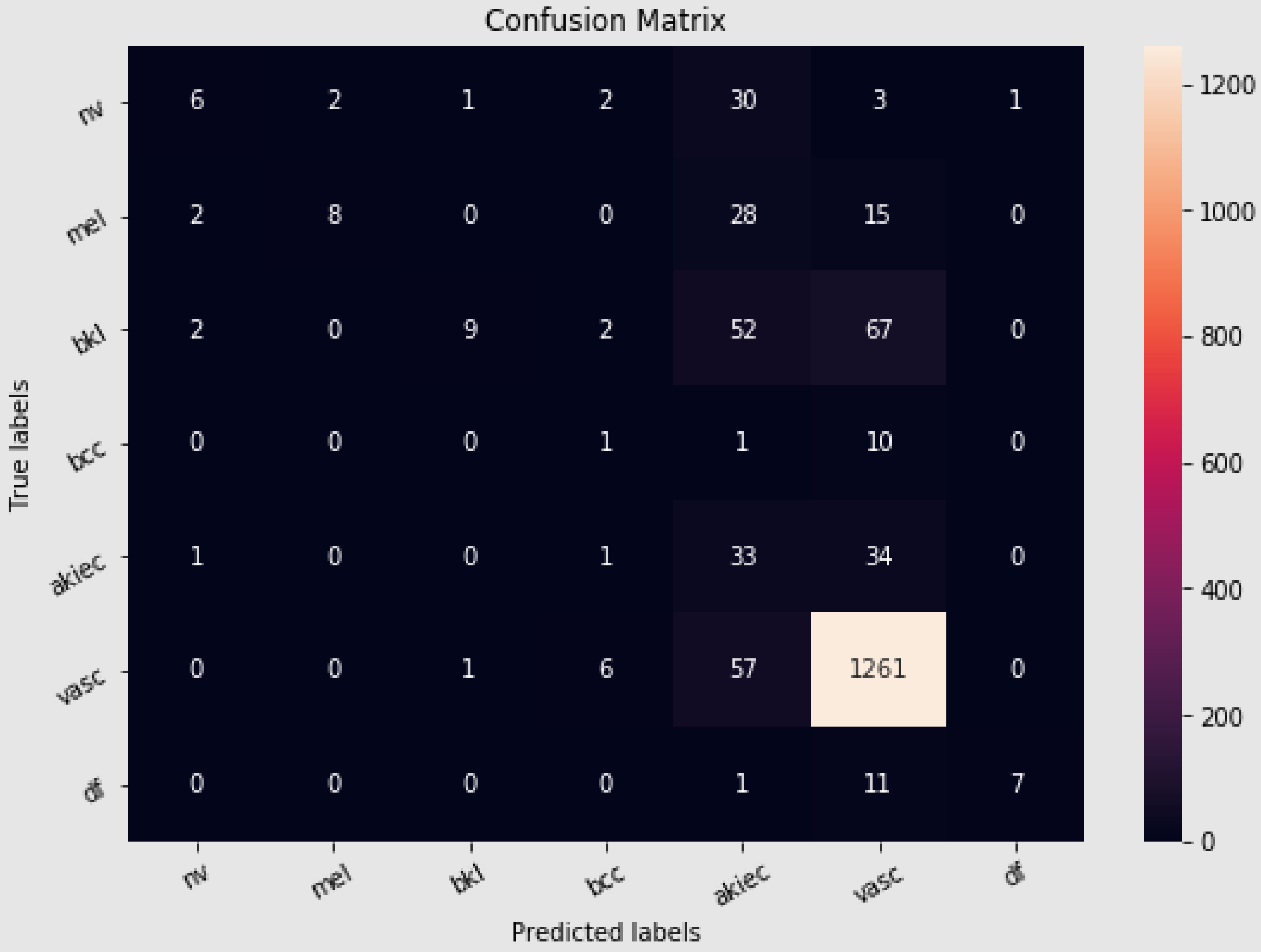


ResNet 50



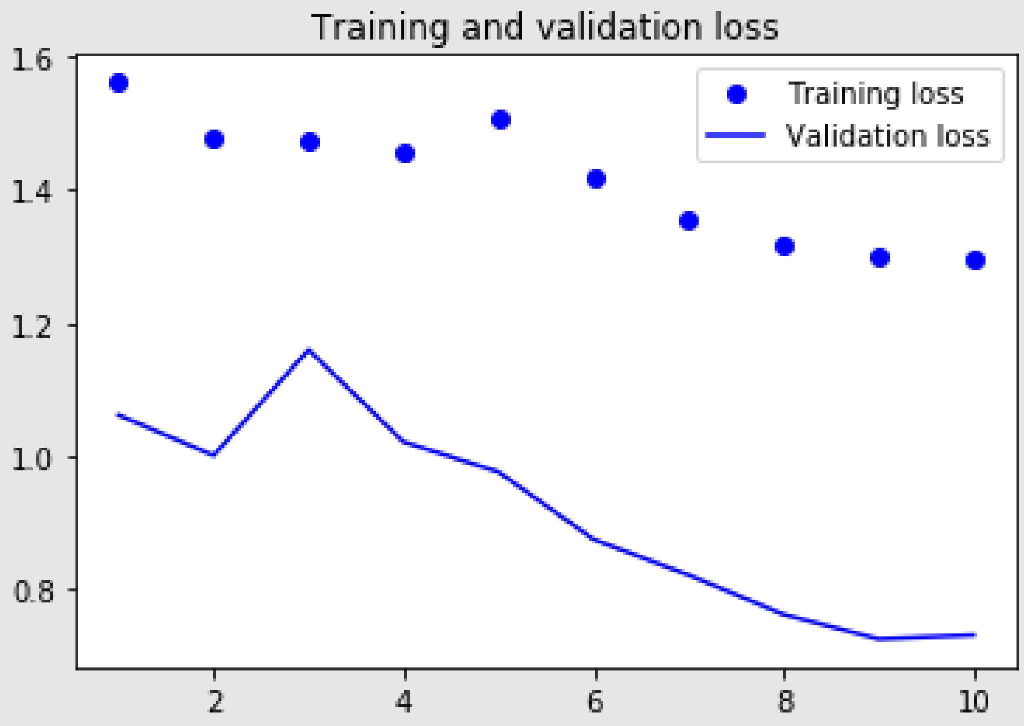
	precision	recall	f1-score	support
akiec	0.00	0.00	0.00	45
bcc	0.44	0.21	0.28	53
bkl	0.40	0.21	0.28	132
df	0.00	0.00	0.00	12
mel	0.22	0.83	0.34	69
nv	0.92	0.90	0.91	1325
vasc	0.00	0.00	0.00	19
accuracy			0.78	1655
macro avg	0.28	0.31	0.26	1655
weighted avg	0.79	0.78	0.77	1655

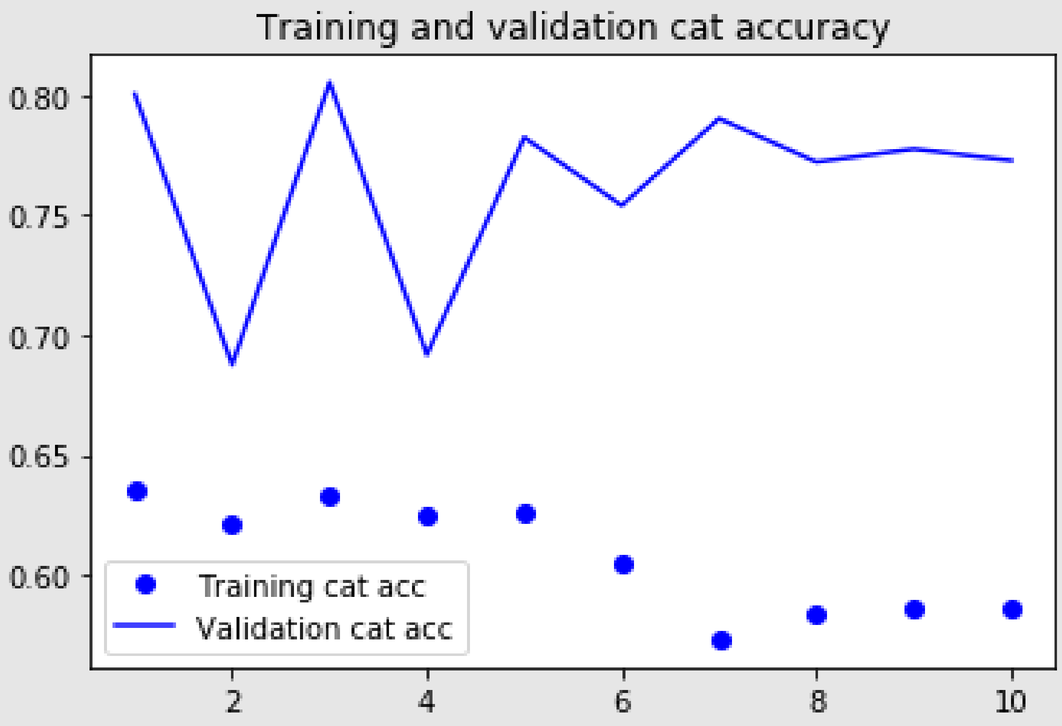
MobileNet



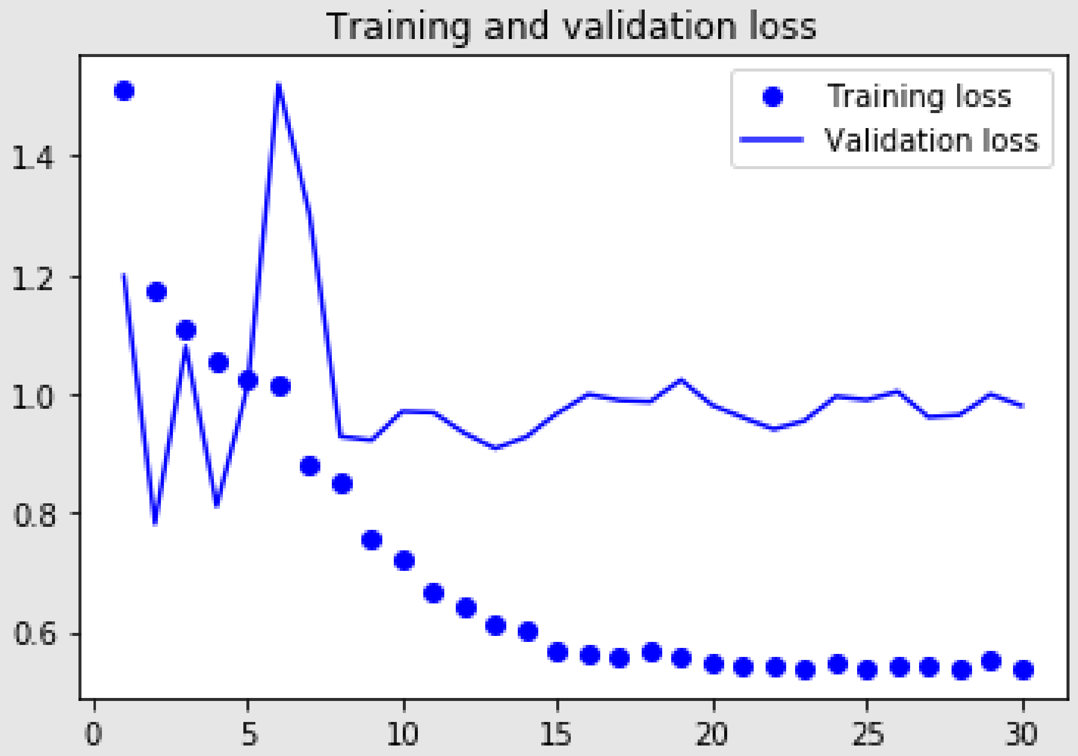
	precision	recall	f1-score	support
akiec	0.55	0.13	0.21	45
bcc	0.80	0.15	0.25	53
bkl	0.82	0.07	0.13	132
df	0.08	0.08	0.08	12
mel	0.16	0.48	0.24	69
nv	0.90	0.95	0.93	1325
vasc	0.88	0.37	0.52	19
accuracy			0.80	1655
macro avg	0.60	0.32	0.34	1655
weighted avg	0.84	0.80	0.78	1655

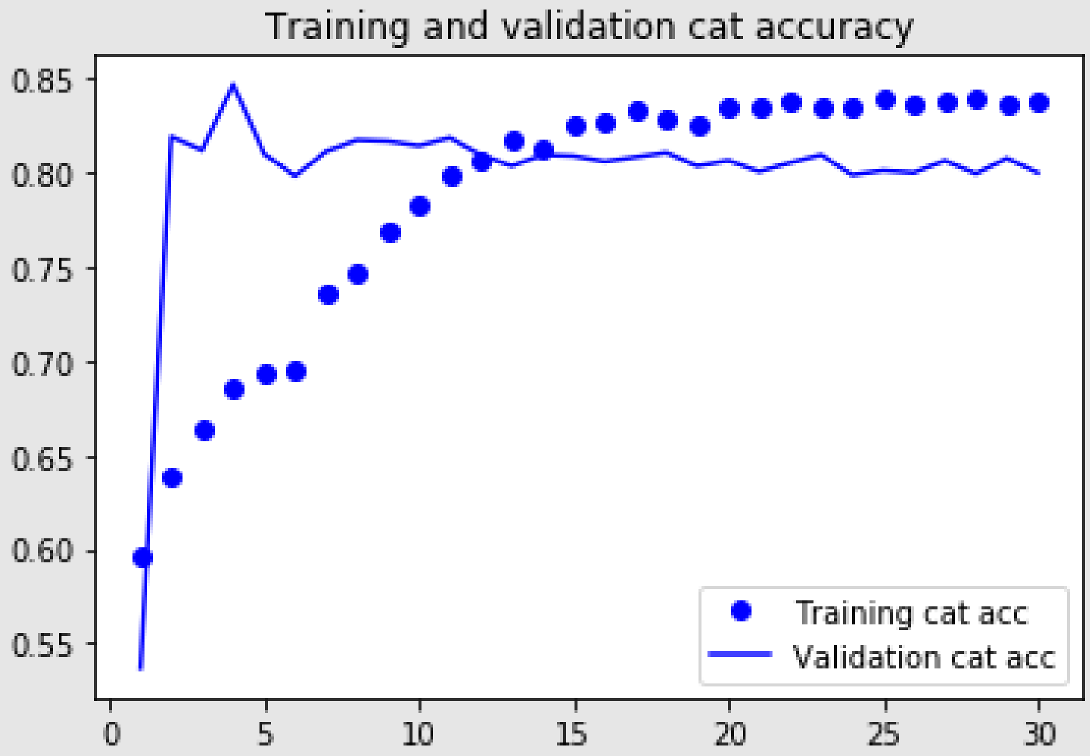
ResNet 50; epoch=10





MobileNet; epoch=30





Reference

1. Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
2. Identity Mappings in Deep Residual Networks, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
3. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Andrew G. Joward, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam
4. https://www.zhihu.com/search?type=content&q=%E6%AE%8B%E5%B7%AE%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C
5. https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4

THANK YOU