Skin Cancer MNIST HAM10000:

A comparative Study of Different Machine Learning Approaches

Aaron Woodhouse, Mathew Kasbarian

Problem

A comparative study of different machine learning approaches on the Skin Cancer MNIST: HAM10000 database. We compare different Neural Network models to determine which has the best testing accuracy on this dataset, including:

- Linear
- MLP
- CNN
- Deep CNN
- Three Custom models
- DenseNet-121

Dataset

Skin Cancer MNIST: HAM10000

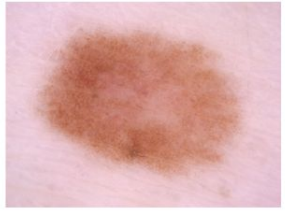
Nevus

Samples:	9958 Usable, 10015 Total
Images:	36x36 RGB images (3x36x36)
Table Data:	3 Features (Age, Sex, Localization)
Classes:	7 (Lesion Type)

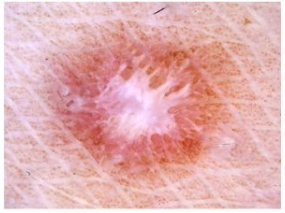
Train / Val / Test % split of dataset: 72 / 8 / 20

Classes (Lesion Types)

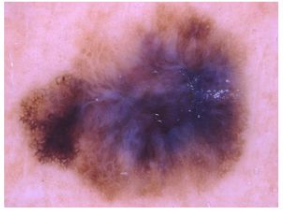
Nevus



Dermatofibroma

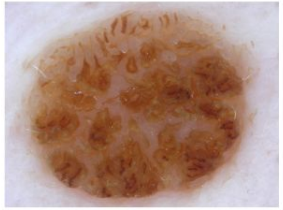


Melanoma

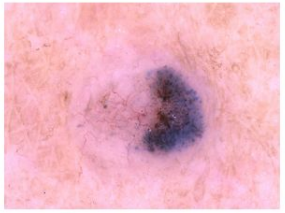


Pigmented Bowen's

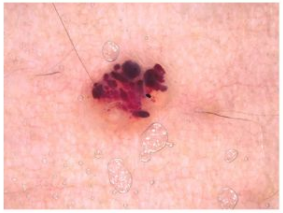
Pigmented Benign Keratoses



Basal Cell Carcinoma



Vascular



Cancerous:

Pigmented Bowen's

Basal Cell Carcinoma

Melanoma

Non-Cancerous:

Pigmented Benign keratosis

Dermatofibroma

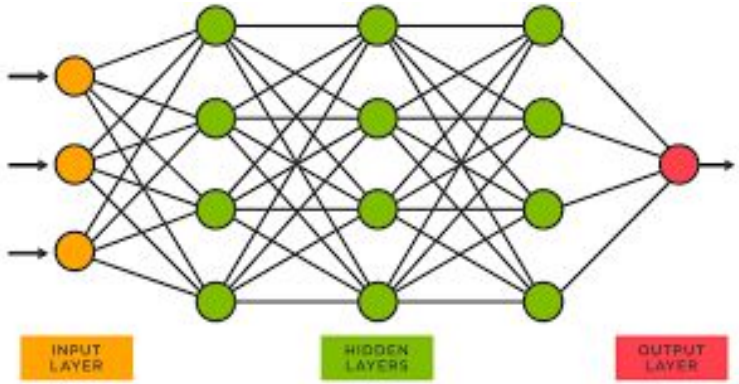
Nevus

Vascular

Neural Network Design Overview

Two data variables: image data and metadata

Epochs and batch size were determined individually for each model to maximize accuracy



Neural Network Design (Linear)

```
m = LinearModel()
xs, xs2, ys = next(iter(train_dataloader))
print(xs.shape, xs2.shape)
summary(m, xs, xs2);
```

```
torch.Size([30, 3, 36, 36]) torch.Size([30, 3])
==========================================================
         Kernel Shape Output Shape   Params Mult-Adds
Layer
0_flat              -   [30, 3888]        -         -
1_output    [3891, 7]      [30, 7]  27.244k   27.237k
----------------------------------------------------------
                        Totals
Total params          27.244k
Trainable params      27.244k
Non-trainable params      0.0
Mult-Adds             27.237k
==========================================================
```

Imgs and data are combined after layer 0

Training: 30 Epochs, Batches of 100

Avg Epoch: 0.16s **Total Training Time:** 4.79s

Final Results:

train_loss=0.87, train_acc=0.70, val_loss=0.85, val_acc=0.70

Testing Accuracy: 68.71 %

Neural Network Design (MLP)

```
m = MLPModel()
xs, xs2, ys = next(iter(train_dataloader))
print(xs.shape, xs2.shape)
summary(m, xs, xs2);
```

```
torch.Size([30, 3, 36, 36]) torch.Size([30, 3])
==========================================================
         Kernel Shape Output Shape  Params Mult-Adds
Layer
0_flat              -   [30, 3888]       -         -
1_ll      [3891, 200]    [30, 200]  778.4k    778.2k
2_relu              -    [30, 200]       -         -
3_output     [200, 7]      [30, 7]  1.407k      1.4k
----------------------------------------------------------
                          Totals
Total params           779.807k
Trainable params       779.807k
Non-trainable params        0.0
Mult-Adds               779.6k
==========================================================
```

Imgs and data are combined after layer 0

Training: 30 Epochs, Batches of 100

Avg Epoch: 0.50s **Total Training Time:** 15.06s

Final Results:

train_loss=0.83, train_acc=0.71, val_loss=0.78, val_acc=0.73

Testing Accuracy: 72.07 %

Neural Network Design (CNN)

```
m = CNNModel()
xs, xs2, ys = next(iter(train_dataloader))
print(xs.shape, xs2.shape)
summary(m, xs, xs2);
```

```
torch.Size([30, 3, 36, 36]) torch.Size([30, 3])
==========================================================================
                    Kernel Shape     Output Shape  Params Mult-Adds
Layer
0_conv.Conv2d_0     [3, 5, 3, 3]  [30, 5, 36, 36]   140.0   174.96k
1_conv.MaxPool2d_1             -  [30, 5, 18, 18]       -         -
2_relu                         -  [30, 5, 18, 18]       -         -
3_flat                         -       [30, 1620]       -         -
4_ll                [1623, 200]        [30, 200]  324.8k    324.6k
5_relu                         -        [30, 200]       -         -
6_output              [200, 7]          [30, 7]  1.407k      1.4k
--------------------------------------------------------------------------
                         Totals
Total params           326.347k
Trainable params       326.347k
Non-trainable params        0.0
Mult-Adds              500.96k
==========================================================================
```

Imgs and data are combined after layer 3

Training: 30 Epochs, Batches of 100

Avg Epoch: 0.49s **Total Training Time:** 14.74s

Final Results:

train_loss=0.61, train_acc=0.78, val_loss=0.68, val_acc=0.74

Testing Accuracy: 75.99 %

Neural Network Design (Deep CNN)

```
m = DCNNModel()
xs, xs2, ys = next(iter(train_dataloader))
print(xs.shape, xs2.shape)
summary(m, xs, xs2);
```

```
torch.Size([30, 3, 36, 36]) torch.Size([30, 3])
==========================================================================
                      Kernel Shape      Output Shape  Params Mult-Adds
Layer
0_conv.Conv2d_0      [3, 10, 5, 5]  [30, 10, 36, 36]   760.0    972.0k
1_conv.MaxPool2d_1               -  [30, 10, 18, 18]       -         -
2_conv.Conv2d_2     [10, 20, 3, 3]  [30, 20, 18, 18]   1.82k    583.2k
3_conv.MaxPool2d_3               -    [30, 20, 9, 9]       -         -
4_relu                           -    [30, 20, 9, 9]       -         -
5_flat                           -        [30, 1620]       -         -
6_ll                  [1623, 200]         [30, 200]  324.8k    324.6k
7_relu                           -         [30, 200]       -         -
8_output                [200, 7]           [30, 7]  1.407k      1.4k
--------------------------------------------------------------------------
                         Totals
Total params           328.787k
Trainable params       328.787k
Non-trainable params        0.0
Mult-Adds              1.8812M
==========================================================================
```

Imgs and data are combined after layer 3

Training: 30 Epochs, Batches of 100

Avg Epoch: 0.85s **Total Training Time:** 25.59s

Final Results:

train_loss=0.50, train_acc=0.81, val_loss=0.63, val_acc=0.77

Testing Accuracy: 77.90 %

Neural Network Design (Custom 1)

```
m = CustomModel1()
xs, xs2, ys = next(iter(train_dataloader))
print(xs.shape, xs2.shape)
summary(m, xs, xs2);
```

```
torch.Size([30, 3, 36, 36]) torch.Size([30, 3])
==========================================================================
                      Kernel Shape      Output Shape  Params Mult-Adds
Layer
0_conv.Conv2d_0      [3, 10, 5, 5]  [30, 10, 36, 36]   760.0    972.0k
1_conv.MaxPool2d_1               -  [30, 10, 18, 18]       -         -
2_conv.Conv2d_2     [10, 20, 5, 5]  [30, 20, 18, 18]   5.02k     1.62M
3_conv.MaxPool2d_3               -    [30, 20, 9, 9]       -         -
4_conv.Conv2d_4     [20, 40, 5, 5]    [30, 40, 9, 9]  20.04k     1.62M
5_conv.MaxPool2d_5               -    [30, 40, 3, 3]       -         -
6_relu                           -    [30, 40, 3, 3]       -         -
7_flat                           -         [30, 360]       -         -
8_ll                  [363, 200]          [30, 200]   72.8k     72.6k
9_mlp.Linear_0        [200, 100]          [30, 100]   20.1k     20.0k
10_mlp.ReLU_1                    -          [30, 100]       -         -
11_mlp.Linear_2       [100, 200]          [30, 200]   20.2k     20.0k
12_mlp.Linear_0       [200, 100]          [30, 100]       -     20.0k
13_mlp.ReLU_1                    -          [30, 100]       -         -
14_mlp.Linear_2       [100, 200]          [30, 200]       -     20.0k
15_relu                          -          [30, 200]       -         -
16_output               [200, 7]            [30, 7]  1.407k      1.4k
--------------------------------------------------------------------------
                        Totals
Total params          140.327k
Trainable params      140.327k
Non-trainable params       0.0
Mult-Adds             4.366M
==========================================================================
```

Imgs and data are combined after layer 7

Training: 30 Epochs, Batches of 100

Avg Epoch: 1.06s **Total Training Time:** 31.79s

Final Results:

train_loss=0.54, train_acc=0.79, val_loss=0.64, val_acc=0.77

Testing Accuracy: 78.30 %

Neural Network Design (Custom 2)

```
m = CustomModel2()
xs, xs2, ys = next(iter(train_dataloader))
print(xs.shape, xs2.shape)
summary(m, xs, xs2);
```

```
torch.Size([30, 3, 36, 36]) torch.Size([30, 3])
==========================================================================
                      Kernel Shape      Output Shape  Params Mult-Adds
Layer
0_conv.Conv2d_0      [3, 25, 5, 5]  [30, 25, 36, 36]    1.9k     2.43M
1_conv.MaxPool2d_1               -  [30, 25, 12, 12]       -         -
2_conv.ReLU_2                    -  [30, 25, 12, 12]       -         -
3_conv.Conv2d_3     [25, 50, 3, 3]  [30, 50, 12, 12]   11.3k     1.62M
4_conv.MaxPool2d_4               -    [30, 50, 4, 4]       -         -
5_conv.ReLU_5                    -    [30, 50, 4, 4]       -         -
6_flat                           -          [30, 800]       -         -
7_ll                   [803, 200]          [30, 200]  160.8k    160.6k
8_mlp.Linear_0         [200, 100]          [30, 100]   20.1k     20.0k
9_mlp.ReLU_1                     -          [30, 100]       -         -
10_mlp.Linear_2        [100, 200]          [30, 200]   20.2k     20.0k
11_mlp.Linear_0        [200, 100]          [30, 100]       -     20.0k
12_mlp.ReLU_1                    -          [30, 100]       -         -
13_mlp.Linear_2        [100, 200]          [30, 200]       -     20.0k
14_output                [200, 7]            [30, 7]  1.407k      1.4k
--------------------------------------------------------------------------
                        Totals
Total params          215.707k
Trainable params      215.707k
Non-trainable params       0.0
Mult-Adds              4.292M
==========================================================================
```

Imgs and data are combined after layer 7

Training: 45 Epochs, Batches of 130

Avg Epoch: 1.44s **Total Training Time:** 64.96s

Final Results:

train_loss=0.51, train_acc=0.81, val_loss=0.65, val_acc=0.77

Testing Accuracy: 78.20 %

Neural Network Design (Custom 3)

```
m = CustomModel3()
xs, xs2, ys = next(iter(train_dataloader))
print(xs.shape, xs2.shape)
summary(m, xs, xs2);
```

```
torch.Size([30, 3, 36, 36]) torch.Size([30, 3])
==========================================================================
                      Kernel Shape      Output Shape   Params Mult-Adds
Layer
0_conv.Conv2d_0      [3, 25, 5, 5]  [30, 25, 36, 36]     1.9k     2.43M
1_conv.MaxPool2d_1               -  [30, 25, 12, 12]        -         -
2_conv.ReLU_2                    -  [30, 25, 12, 12]        -         -
3_conv.Conv2d_3     [25, 50, 3, 3]  [30, 50, 12, 12]    11.3k     1.62M
4_conv.MaxPool2d_4               -    [30, 50, 4, 4]        -         -
5_conv.ReLU_5                    -    [30, 50, 4, 4]        -         -
6_flat                           -         [30, 800]        -         -
7_lstm                           -         [30, 200]  1.4448M     1.44M
8_ll                  [203, 200]         [30, 200]    40.8k     40.6k
9_mlp.Linear_0        [200, 100]         [30, 100]    20.1k     20.0k
10_mlp.ReLU_1                    -         [30, 100]        -         -
11_mlp.Linear_2       [100, 200]         [30, 200]    20.2k     20.0k
12_mlp.Linear_0       [200, 100]         [30, 100]        -     20.0k
13_mlp.ReLU_1                    -         [30, 100]        -         -
14_mlp.Linear_2       [100, 200]         [30, 200]        -     20.0k
15_output               [200, 7]           [30, 7]   1.407k      1.4k
--------------------------------------------------------------------------
                         Totals
Total params          1.540507M
Trainable params      1.540507M
Non-trainable params        0.0
Mult-Adds               5.612M
==========================================================================
```

Imgs and data are combined after layer 7

Training: 45 Epochs, Batches of 150

Avg Epoch: 3.00s **Total Training Time:** 135.06s

Final Results:

train_loss=0.45, train_acc=0.83, val_loss=0.73, val_acc=0.77

Testing Accuracy: 75.94 %

Neural Network Design (DenseNet-121)

DenseNet uses a combination of many layers.

It has:

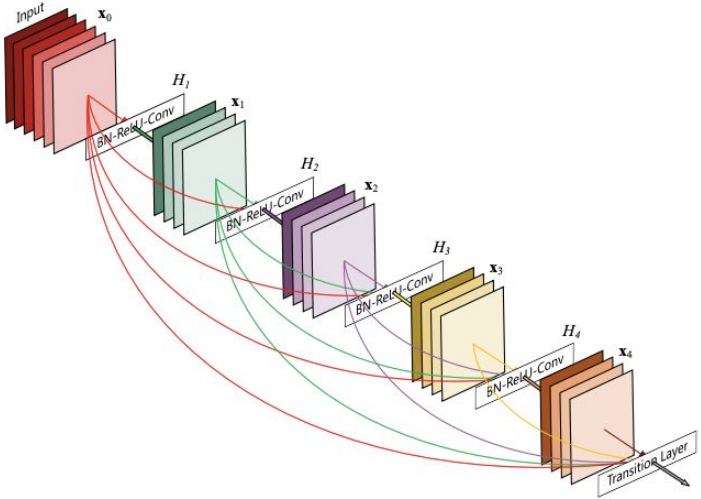
1 7x7 Convolutions,

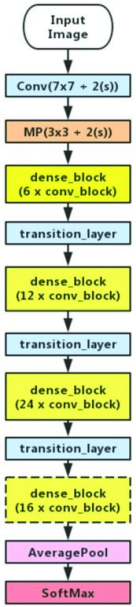
58 3x3 Convolutions,

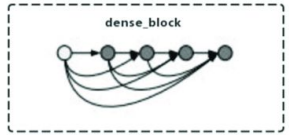
61 1x1 Convolutions,

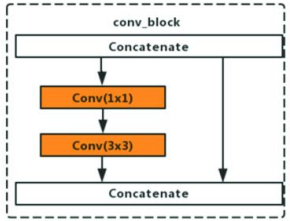
4 AvgPools,

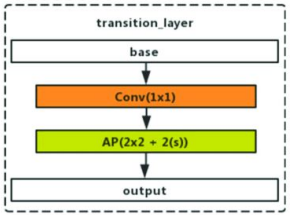
and 1 fully connected layer



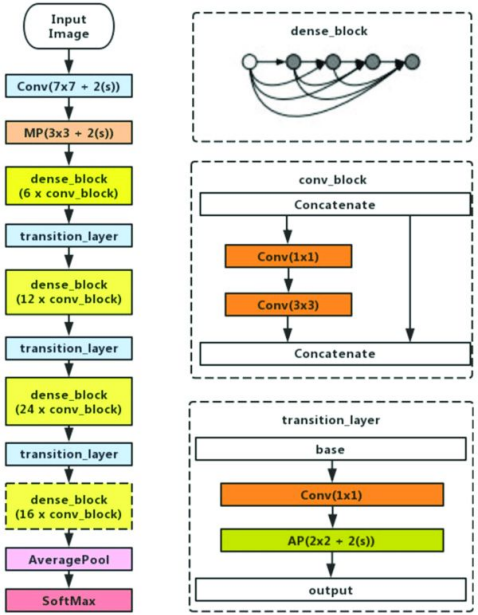








Neural Network Design (DenseNet-121)



	Totals
Total params	6.961116M
Trainable params	6.961116M
Non-trainable params	0.0
Mult-Adds	64.705141M

Imgs and data are combined after layer 7

365 layers

Training: 45 Epochs, Batches of 150

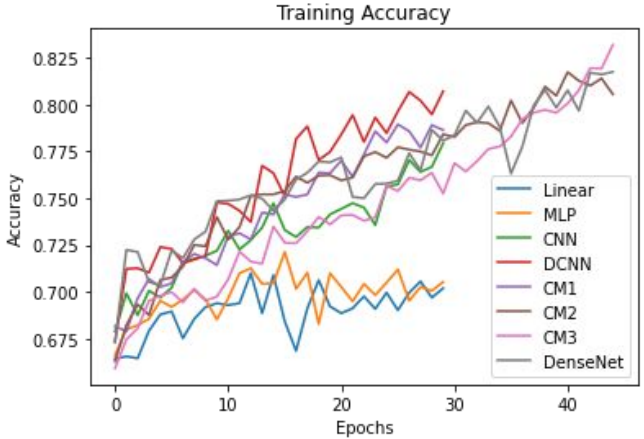
Avg Epoch: 35.43s **Total Training Time:** 1594.15s

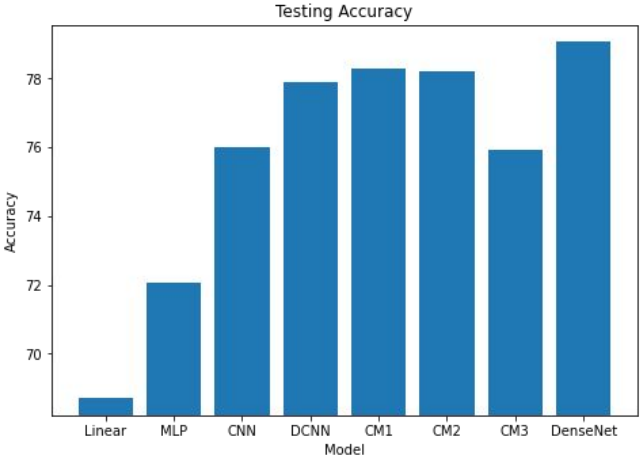
Final Results:

train_loss=0.48, train_acc=0.82, val_loss=0.63, val_acc=0.78

Testing Accuracy: 79.06 %

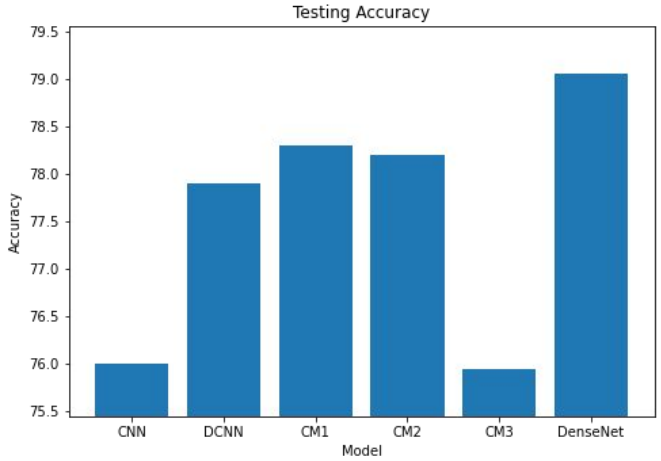
Neural Network Designs Compared





The DenseNet Model performed the best, with 79.1% test accuracy

The Linear model meanwhile performed the worst, with only 68.7% test accuracy



Real-life Applications

- Dermatologists could use the application to quickly classify large quantities of lesion images as cancerous or non-cancerous. Though due to inaccuracies they would likely be restricted to using this as a helper tool in diagnosing lesions.

- Medical Practitioners could also use the application to decide how urgently they may need to refer a patient to a dermatologist if a given image is predicted as cancerous or non-cancerous.

Deployment Demo

Deployment Application

```
PS C:\Users\[redacted]\project> python CancerPrediction.py
CSV file path: CancerData/deployment_data.csv
Images folder path: C:/Users/[redacted]/project/CancerData/deployment_images

Loading from dcnn_model.pt
Img     Predicted
0       bkl
1       bcc
2       nv
3       nv
4       akiec
5       bkl
6       bcc
7       bcc
8       vasc
9       mel
10      akiec
11      bcc
12      nv
13      nv
14      nv
15      mel
16      nv
17      bkl
18      akiec
19      bkl
20      bkl

PS C:\Users\[redacted]\project>
```

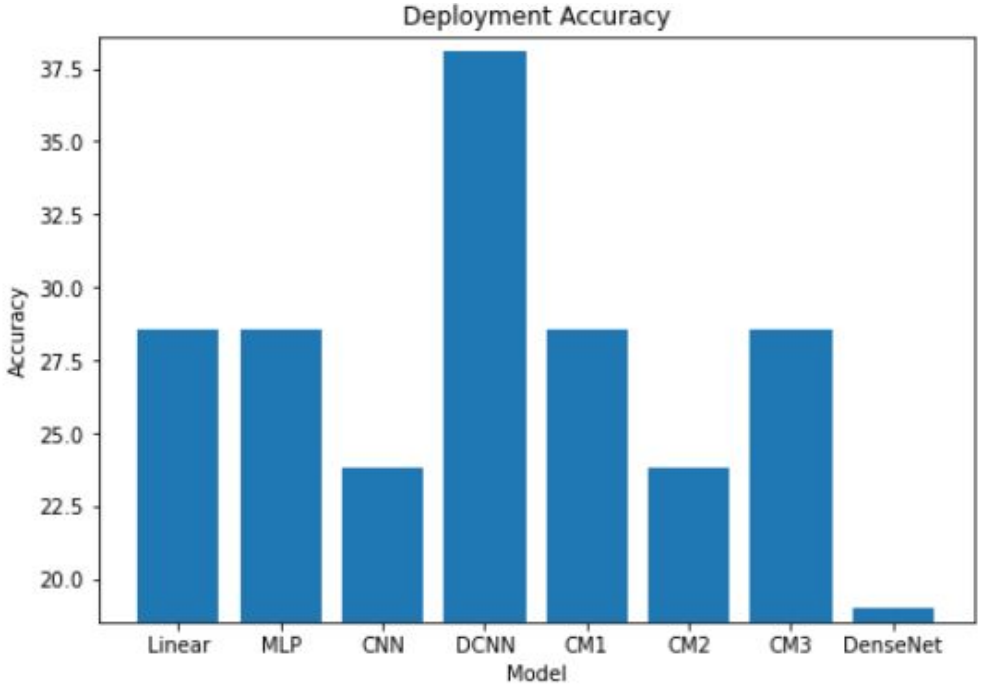
The user specifies a path for the .csv file containing image metadata, and then the folder path containing the images

Deployment Overview

Melanoma

- Using 21 samples with 3 for each class of lesion
- Each sample has an 36x36 RGB image (3x36x36), and
- data containing the age, sex, and localization for each sample
- Each model was tested on the deployment data, which was collected from images found online

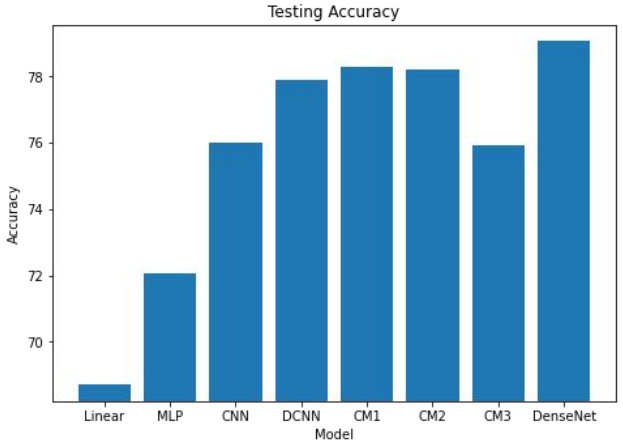
Deployment Results



Significant differences in accuracy between deployment and testing

The DCNN Model performed the best, with 38% accuracy

DenseNet meanwhile performed the worst, with only 19% accuracy



Deployment Problems

- Looking at the true positives and false positives of each class we noticed a large amount of false positives of Nevus lesions using the deployment data, and many of the predictions in testing showed a similar result, where most of the accuracy came from the many nevus samples, while predicting other kinds of lesions incorrectly.

- We found the majority of samples in the MNIST Skin Cancer dataset are Nevus lesions. This, combined with how similar Nevus lesions may look to other types of lesions, the models often predict most images as Nevus.

- This causes the training and testing accuracy to be significantly higher than they should be, as shown by the deployment accuracy, where there are an equal number of samples of each class.

References

Skin Cancer MNIST: HAM10000 | Kaggle

DenseNet121 | pytorch | Kaggle

Architecture of DenseNet-121 (opengenus.org)