Classification of Benign and Malignant Melanoma Skin Cancer using Deep Neural Network

CS 7150 - Minh Phuong

Introduction

- Melanoma is a special type of skin cancer. It develops in the cell that produce melanin which creates the color of our skin.
  - One of the main causes for melanoma is UV exposure.
  - Risk of getting melanoma seems to increase in middle-aged people.If detected early, it could be treated well. (C. Halpern et al., 2021).

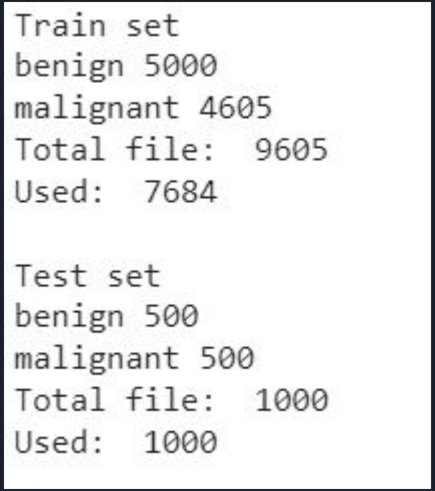
Introduction

- However, it is a challenge to detect melanoma just by normal eyes. It is often developed around a mole on one’s skin and is better observed over time.
- If the mole develops irregularities such as strange border, invasive color or diameter growth, it is highly melanoma.
- Risk factors include fair skin, history of sunburn or excessive UV exposure. Populations residing closer to the equator or higher elevation (hence closer to the sun) are also susceptible. (AIM at Melanoma, 2021)

Dataset and Approach

Dataset

- The dataset was adapted from Kaggle: https://www.kaggle.com/datasets/hasnainjaved/melanoma-skin-cancer-dataset-of-10000-images
- This dataset contains 10000 images of benign and malignant melanoma skin cancer photographs.
- There are 9605 files for training and 1000 for testing. The train-validation split is 80-20



```
Train set
benign 5000
malignant 4605
Total file:  9605
Used:  7684

Test set
benign 500
malignant 500
Total file:  1000
Used:  1000
```

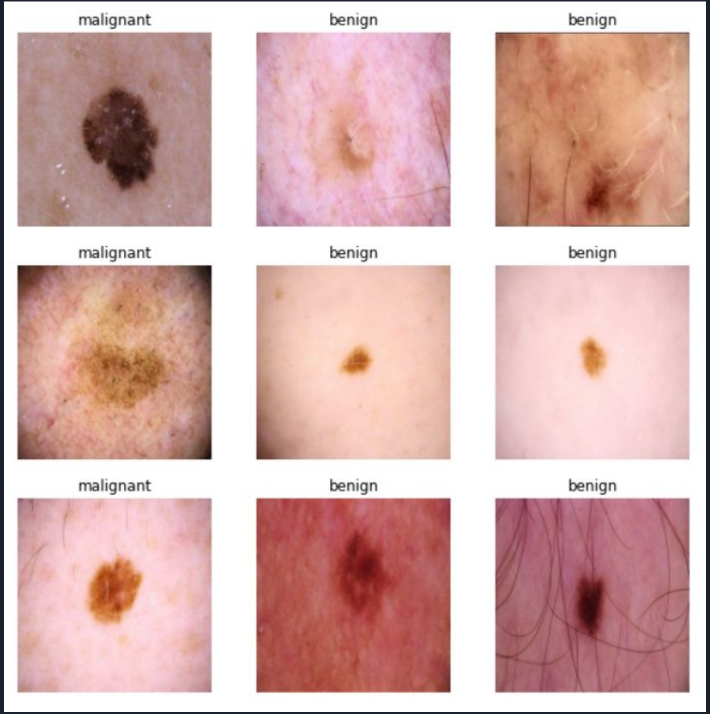
Approach

- Data analysis and augmentation
- Experimented with fully connected layer (baseline), convolutional network
- Experimented with transfer learning: VGG16, ResNet50

Data Analysis

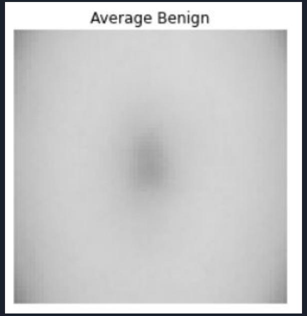
Benign vs Malignant

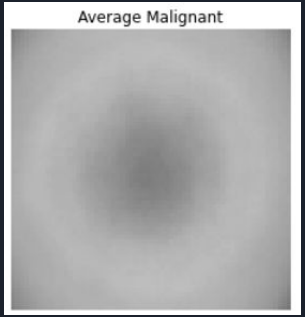
- After several rounds of inspections, it seemed that the malignant moles are darker in color, have raised surface as well as a more "blooming" pattern.

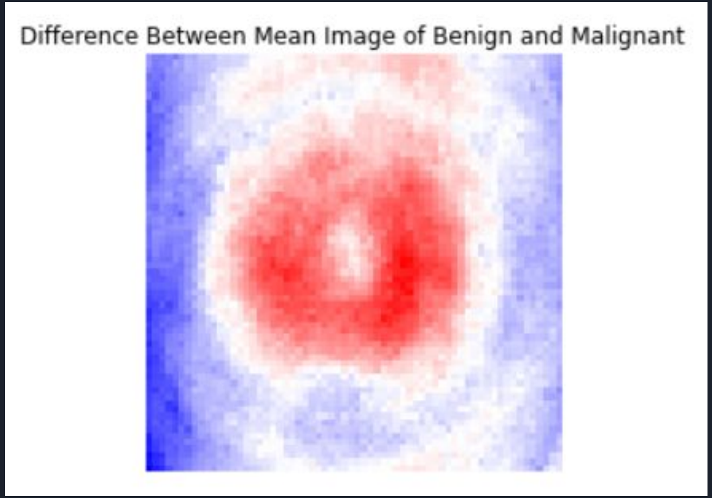


Benign vs Malignant

- We created an “averageimage” of the melanomas
- Hypothesis: blooming pattern and larger area of coverage could be the tell-tale signs of malignant melanoma, ,as compared to the vertically grown patch of benign melanoma.



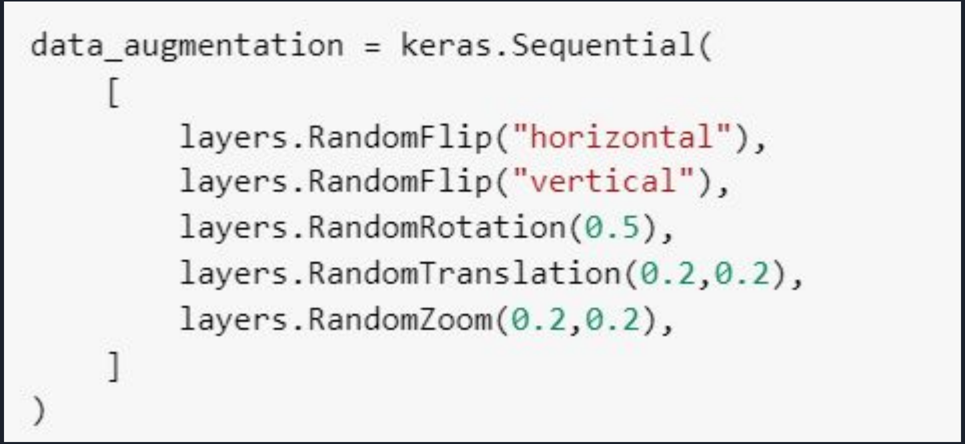




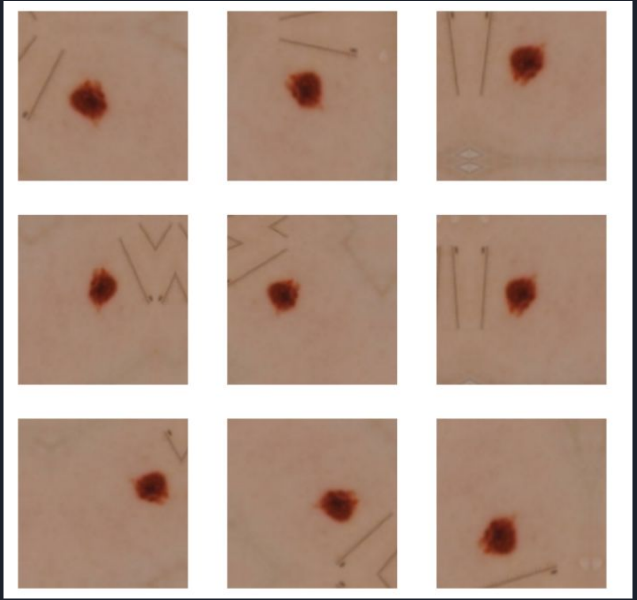
Process

Data Augmentation

For data augmentation, we want to make sure that the images are manipulated in several ways. From our exploration, we see that the moles could be off centered, flattened or skew



```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomFlip("vertical"),
        layers.RandomRotation(0.5),
        layers.RandomTranslation(0.2,0.2),
        layers.RandomZoom(0.2,0.2),
    ]
)
```



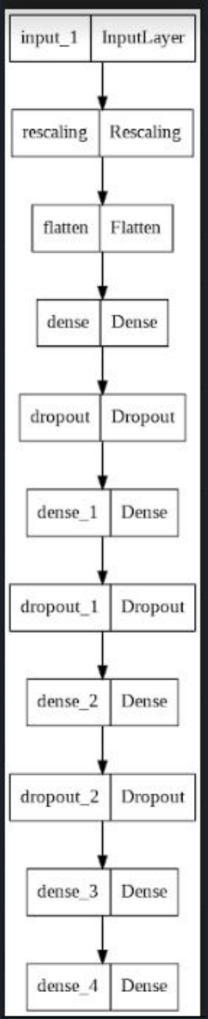
Baseline: Fully connected

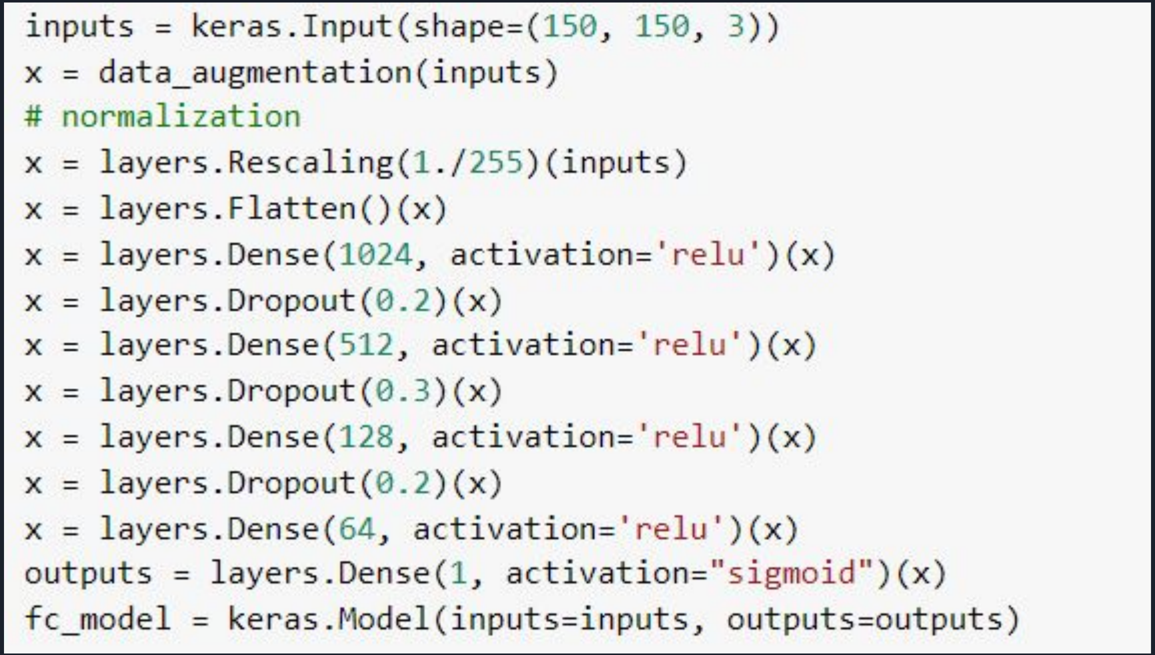
We built a simple fully connected layers for the baseline

For the fully connected layer, we experimented with 2, 3, 4 layers respectively, with the number of hidden units divided by half as the layer increased.

After that, we introduced drop out layers with different ratio from 0.1 to 0.5.

We experimented with 20 epochs and keep the highest accuracy.

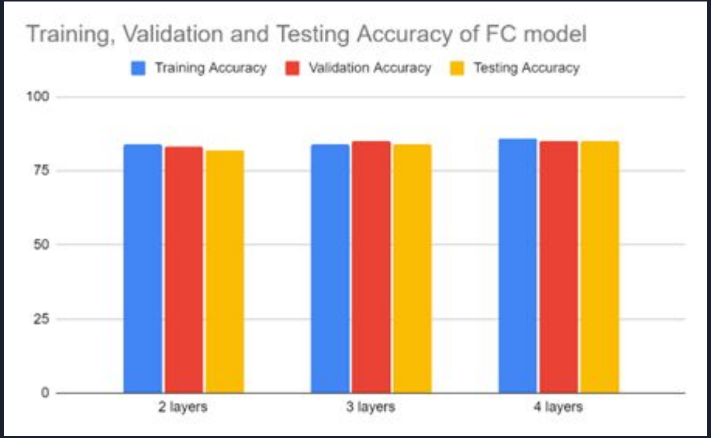


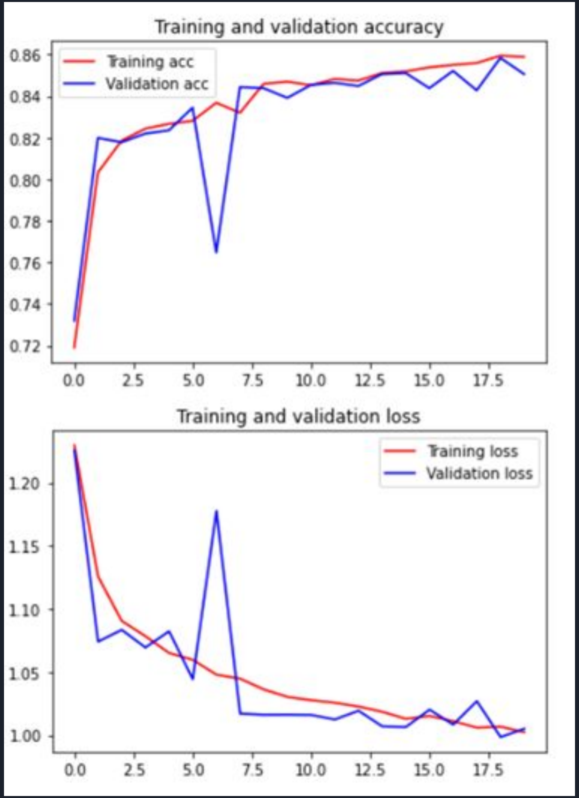


```
inputs = keras.Input(shape=(150, 150, 3))
x = data_augmentation(inputs)
# normalization
x = layers.Rescaling(1./255)(inputs)
x = layers.Flatten()(x)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.3)(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense(64, activation='relu')(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
fc_model = keras.Model(inputs=inputs, outputs=outputs)
```

Baseline: Fully connected

The highest testing accuracy is 87% while the training accuracy is 85%. Dropout rate of 0.2 was the best. However, the loss curve was not stable and overfitting was observed at times.



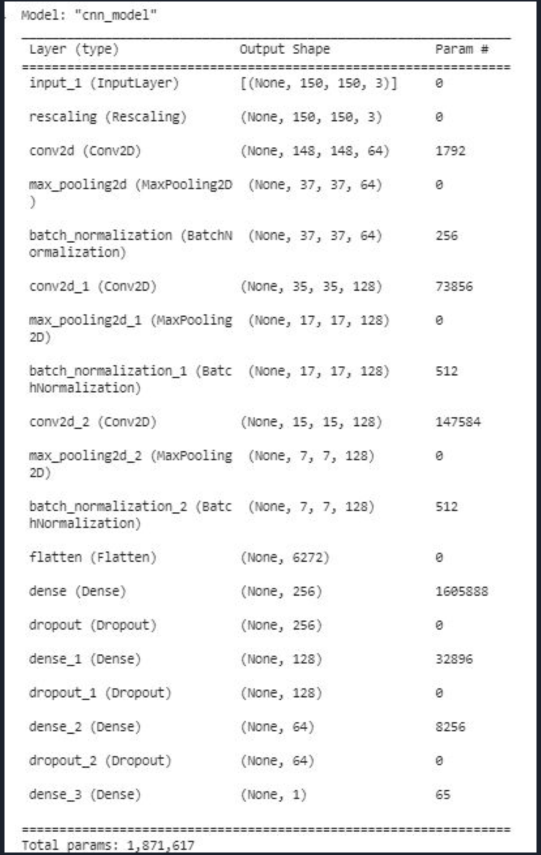


Convolutional Network

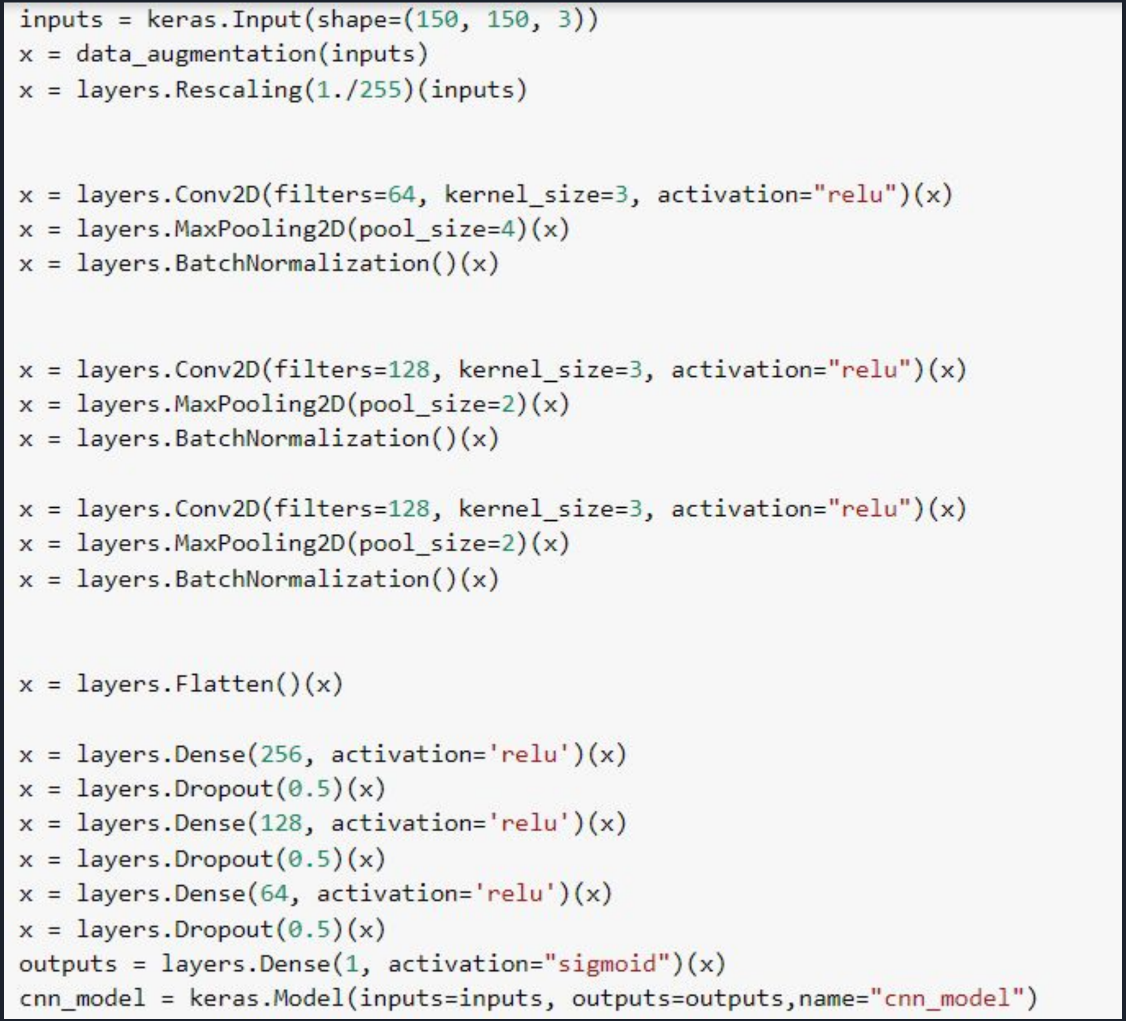
We slowly increase each convolutional block while experiment with different regularization parameters.

- We also introduced batch normalization at each iteration.
- We observed that without dropout layers, the network performed better than with dropout.
- Additionally, global average pooling made the model worse off, hence, flattening layer was used instead.

We also experimented with 2, 3 and 4 blocks of convolutional layers, with increasing number of hidden units. 4 blocks of convolutional was the best performing



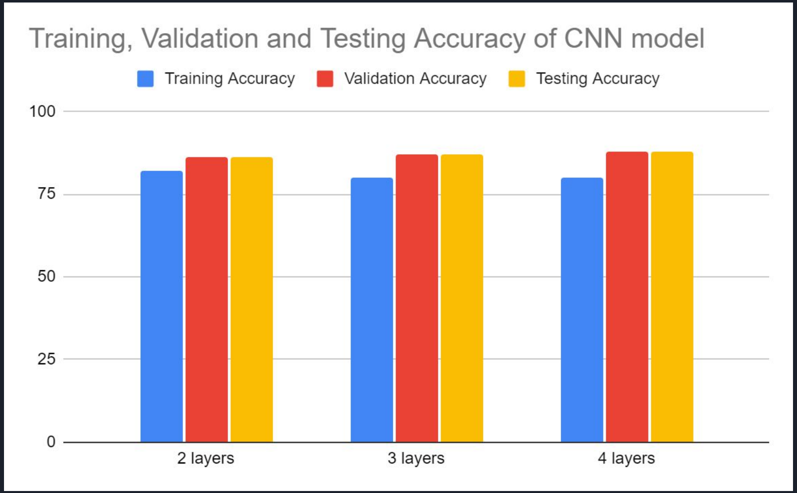
```
Model: "cnn_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 150, 150, 3)]     0

 rescaling (Rescaling)       (None, 150, 150, 3)       0

 conv2d (Conv2D)             (None, 148, 148, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 37, 37, 64)       0
 )

 batch_normalization (BatchN  (None, 37, 37, 64)       256
 ormalization)

 conv2d_1 (Conv2D)           (None, 35, 35, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 batch_normalization_1 (Batc  (None, 17, 17, 128)      512
 hNormalization)

 conv2d_2 (Conv2D)           (None, 15, 15, 128)       147584

 max_pooling2d_2 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 batch_normalization_2 (Batc  (None, 7, 7, 128)        512
 hNormalization)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 256)               1605888

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 dropout_2 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 1,871,617
```

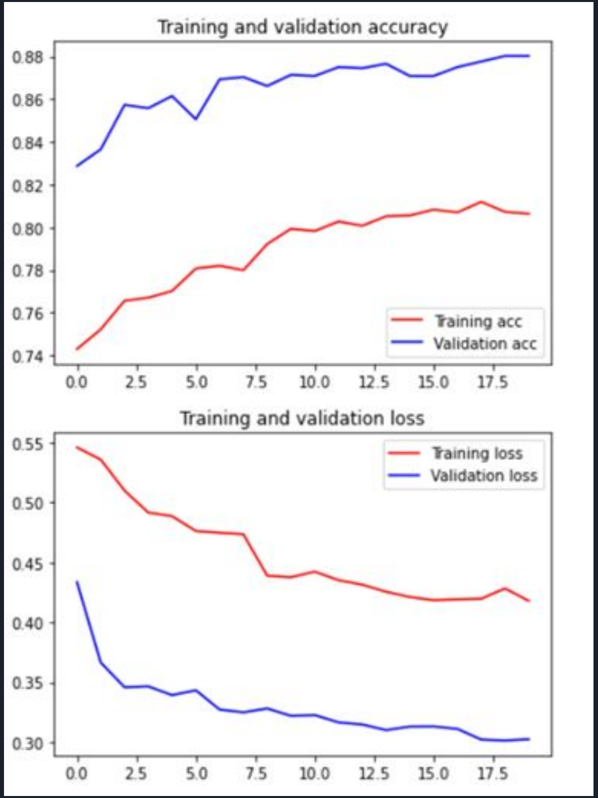


```python
inputs = keras.Input(shape=(150, 150, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(inputs)


x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=4)(x)
x = layers.BatchNormalization()(x)


x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.BatchNormalization()(x)

x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.BatchNormalization()(x)


x = layers.Flatten()(x)

x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(64, activation='relu')(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
cnn_model = keras.Model(inputs=inputs, outputs=outputs,name="cnn_model")
```

Convolutional Network

The highest testing accuracy is 88% while the training accuracy is 80% and validation accuracy is 88%. We can see that the model is not overfitting and the loss curves are more stabilized.



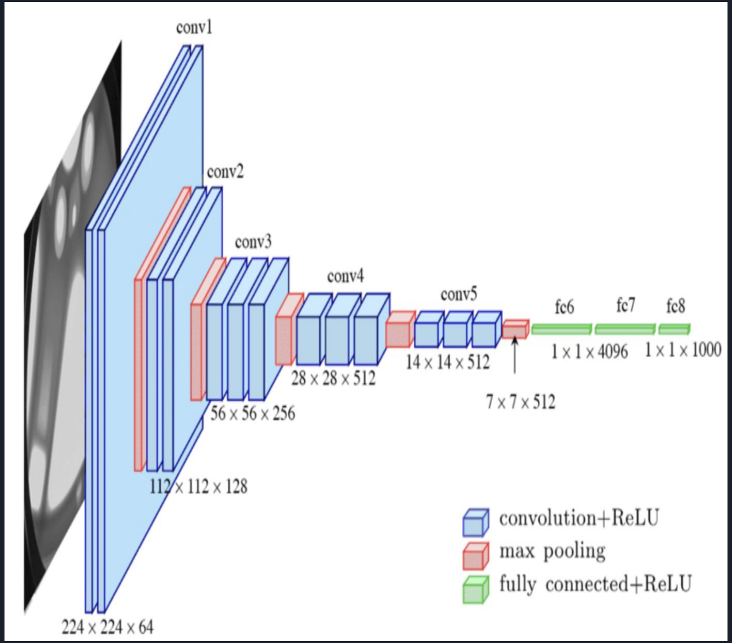


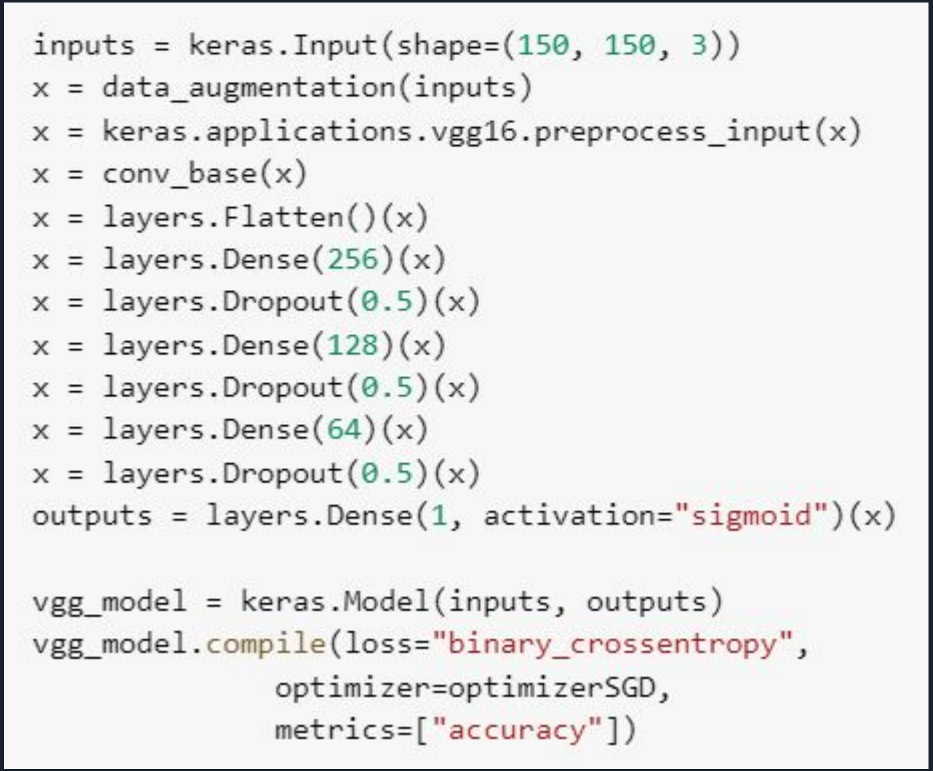
VGG16

For VGG16, we first experimented with freezing everything but the last layer. Then we implemented our own fully connected block, referenced from the fully connected architecture.

We also experimented with 2, 3 and 4 fully connected layers at the end. In our experiments, we noticed that increasing dropout rate to 0.5 was helping the model performing better.

Increasing number of fully connected layers at the end improved the result overall.

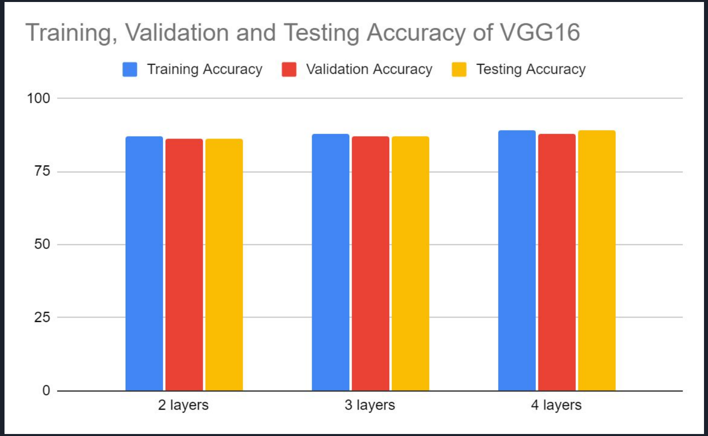


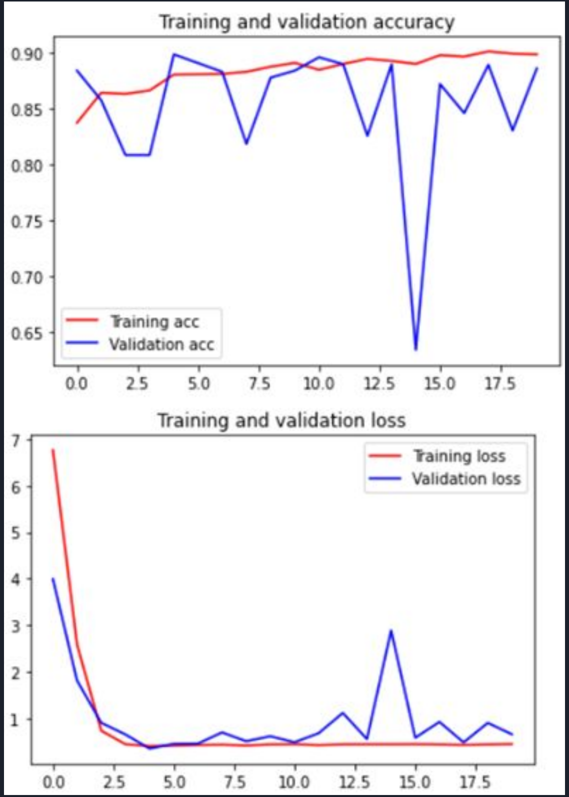


```
inputs = keras.Input(shape=(150, 150, 3))
x = data_augmentation(inputs)
x = keras.applications.vgg16.preprocess_input(x)
x = conv_base(x)
x = layers.Flatten()(x)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(128)(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(64)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)

vgg_model = keras.Model(inputs, outputs)
vgg_model.compile(loss="binary_crossentropy",
              optimizer=optimizerSGD,
              metrics=["accuracy"])
```

VGG16

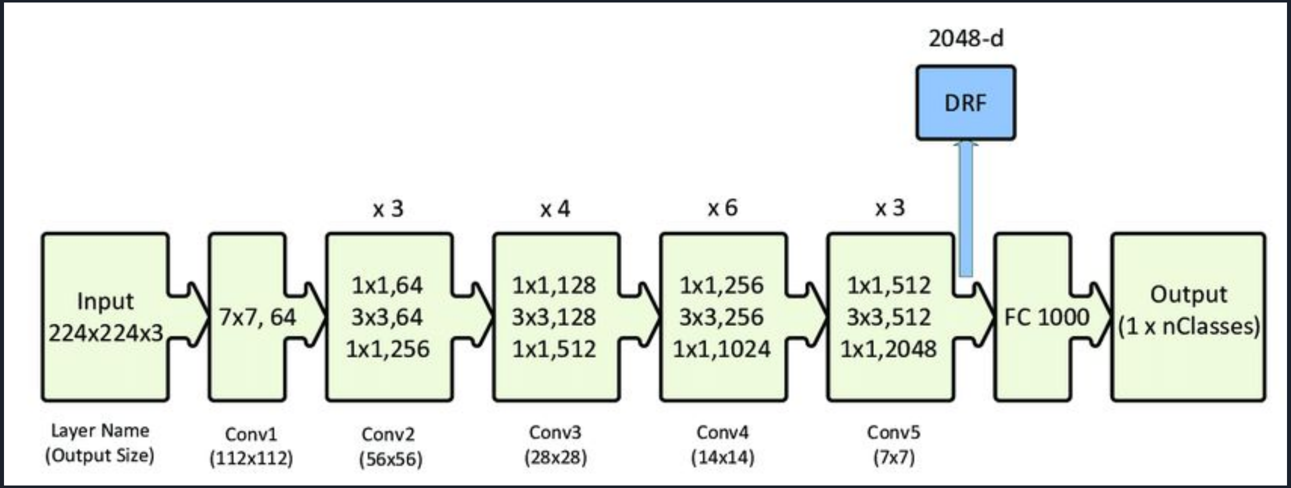
Test accuracy is 89%, an improvement from our CNN Model. The training accuracy is 88% while validation accuracy is 89%. However, the loss curve was once again not stable. Some low dips were visible and overfitting could be a potential issue.

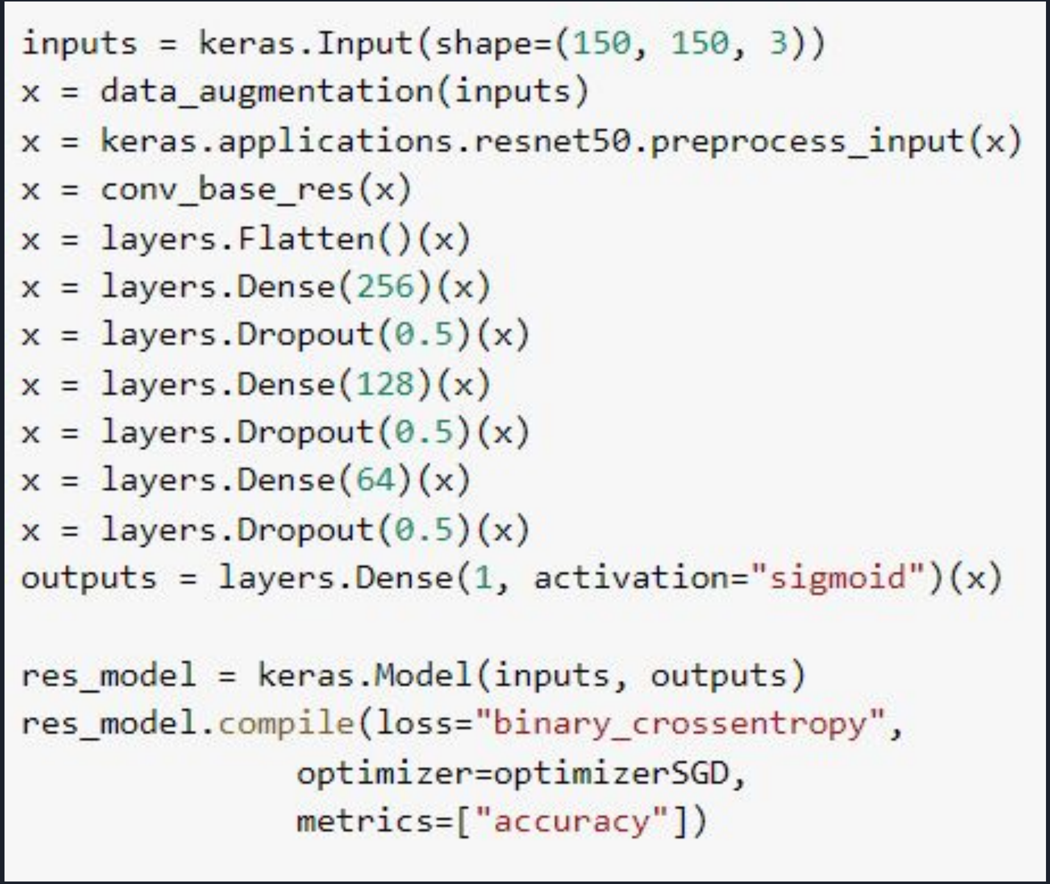




ResNet50

Finally, we experimented with transfer learning on ResNet50. We also froze everything but the last layer. Implemented our own fully connected block, referenced from the fully connected architecture in 2,3 and 4 layers respectively.

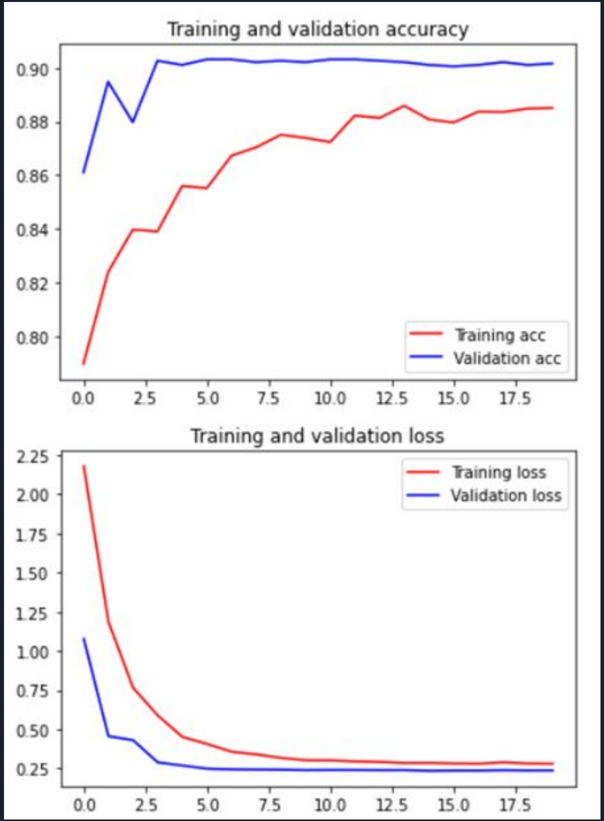


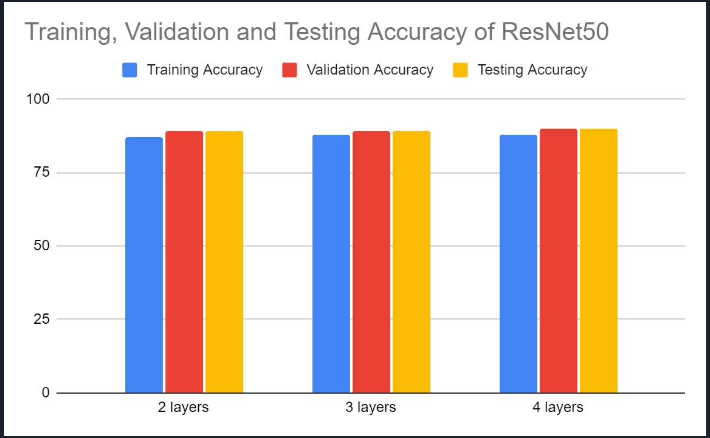


```python
inputs = keras.Input(shape=(150, 150, 3))
x = data_augmentation(inputs)
x = keras.applications.resnet50.preprocess_input(x)
x = conv_base_res(x)
x = layers.Flatten()(x)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(128)(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(64)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)

res_model = keras.Model(inputs, outputs)
res_model.compile(loss="binary_crossentropy",
              optimizer=optimizerSGD,
              metrics=["accuracy"])
```

ResNet50

With transfer learning on ResNet50, the test accuracy is 90%, an improvement from VGG16. The loss curves were much more stable and overfitting was not an issue. Thus, this model combined the best performance from all the previous ones. Similar to before, more fully connected layers at the end improved the result overall.



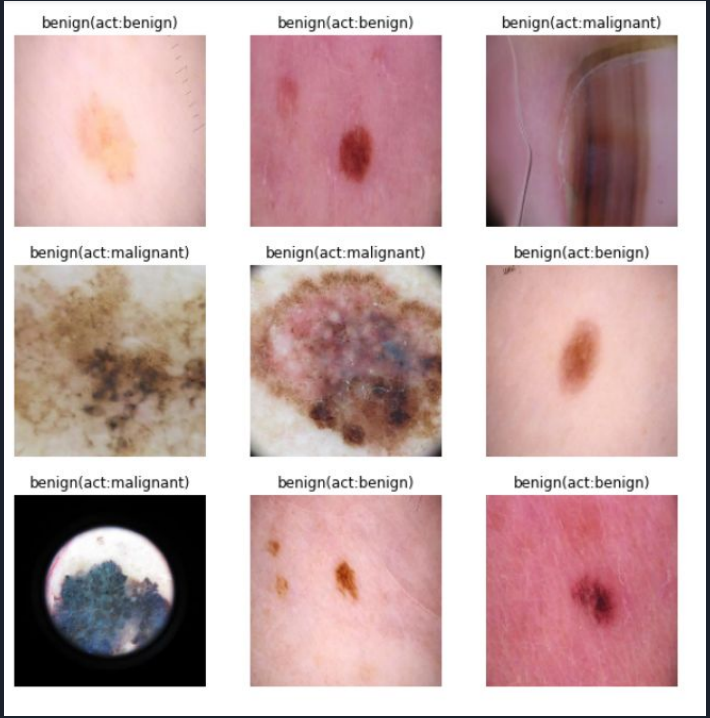


Error Analysis & Conclusion

Error Analysis

By manually looking at the results, we could see that some misclassification happened when malignant cells are classified as benign. This could be due to their various shapes that are not just in the “blooming” forms. Some images are different from the rest as well, with a strong vignette outline.

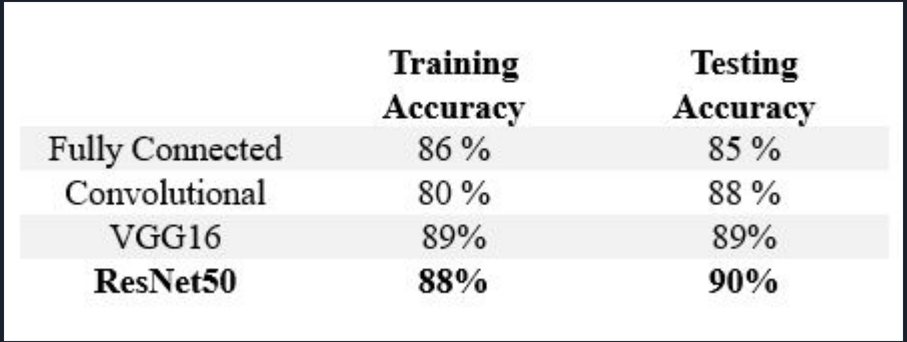
Thus, future works could explore on how to make the images more uniform through data manipulation. Oversampling of the malignant class could also be another alternative.



Conclusion

While in practice, diagnosing melanoma skin cancer requires multi-faceted examination, not just from images but also from physical inspection, a preliminary methodology for classifying benign and malignant moles could be highly useful.

In this experiment, we have seen that Pre-Trained Deep Learning frameworks achieved superior results compared to normal handcrafted fully connected or convolutional networks.



	Training Accuracy	**Testing Accuracy**
Fully Connected	86 %	85 %
Convolutional	80 %	88 %
VGG16	89%	89%
ResNet50	**88%**	**90%**

Conclusion

For many of our models, the testing accuracy is higher than the training accuracy, thus we managed to combat overfitting.

The pretrained model converges faster, the ResNet50 model has the best performance and loss curves.

However, it's worth exploring other state-of-the-art models.

Next step and future works

- Possible next steps can include training on more epochs, looking at more regularization method, experiment with different optimizers and kernel regularizers.
- Experiment with oversampling
- The model can certainly still improve, and it’s best to predict on other datasets beside the one provided.
- Future work could expand on data manipulation and feature extraction before training for more efficient computation.

References

AIM at Melanoma. (2021, January 5). Melanoma Risk Factors. AIM at Melanoma Foundation. Retrieved May 1, 2022, from https://www.aimatmelanoma.org/melanoma101/understanding-melanoma/melanoma-riskfactors/

Byeon, E. (2021, December 15). Exploratory Data Analysis Ideas for Image Classification. Medium. Retrieved May 1, 2022, from https://towardsdatascience.com/exploratory-dataanalysis-ideas-for-image-classificationd3fc6bbfb2d2

C. Halpern, A., A. Marghoob, A., & Reiter, O. (2021, June 1). Melanoma risk factors. SkinCancer.Org. Retrieved May 1, 2022, from https://www.skincancer.org/skin-cancerinformation/melanoma/melanoma-causes-and-riskfactors

Thank you