Neural Computing and Deep Learning

SID: 2232741

Nithiyananda, Praveen (Student)
ANGLIA RUSKIN UNIVERSITY

Table of Contents

Table of Figures

Introduction

Skin lesion classification is an important task in medical field that involves identifying and categorizing various types of skin lesions. Accurate classification of skin lesions is crucial for early detection and treatment of skin diseases such as melanoma, which is a potentially deadly form of skin cancer(*Shellenberger, et al., 2016*). The HAM10000 dataset is a popular dataset that contains images of skin lesions along with clinical metadata for each image. There have been several research undertaken using this dataset to build neural network-based models, for skin lesion classification.

As of now FixCaps network, which has a larger receptive field and uses techniques like convolutional block attention and group convolution to improve accuracy, has achieved an accuracy of 96.49% on the HAM10000 dataset, outperforming other existing methods like InceptionResNetV2 (*Lan et al., 2022*).

Aims

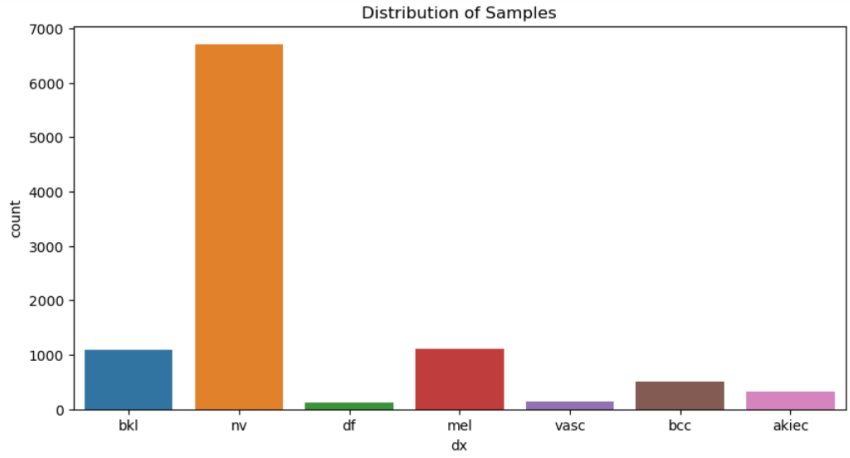
In this project, several different algorithms were tested with the images in the dataset to classify the skin lesion. Four different Convolution Neural Network (CNN) based algorithms were used to build network models for the classification; EfficientNetB0, VGG, MobileNetV2 and InceptionResNetV2. All the four models' experiments and results were then compared to determine the more accurate model.

Exploratory Data Analysis and Data visualization

This section contains an Exploratory Data Analysis (EDA) and data visualization of the HAM10000 dataset to gain insights into its characteristics and better understand the distribution of its different classes. For the analysis described below, please refer to the ***Data Analysis.ipynb*** file.

The HAM10000 dataset consists of 10,015 images of skin lesions, which are classified into 7 different types, including **Bowen's disease** (akiec), **basal cell carcinoma** (bcc), **benign keratosis-like lesions** (bkl), **dermatofibroma** (df), **melanoma** (mel), **melanocytic nevi** (nv) and **vascular lesions** (vasc). Each image is accompanied by various clinical metadata, such as age, gender, and localization of the lesion, though most of these data will not be used for this project's scope.

The distribution of the different lesion types in the dataset was first examined. *Figure 1* shows a bar plot of the frequency of each lesion type. We observe that the dataset is imbalanced, with most images belonging to the melanocytic nevi class, followed by melanoma and benign keratosis.




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

Figure 1: Lesion Types Count

Next, for some additional context of the data, the distribution of lesion localization in the dataset was examined to understand which body part the image is from. Figure 2 shows the frequency of each localization category. It can be observed that most lesions are located on the trunk, followed by the lower extremities and upper extremities. This observation could be useful in designing more effective screening model, but this project does not explore this observation.

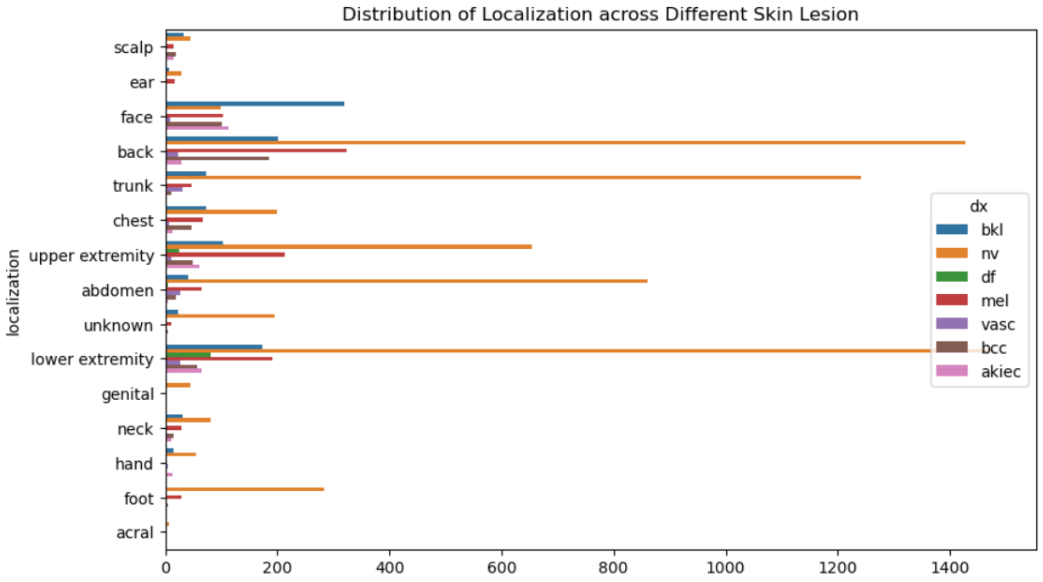


Figure 2: Localization Distribution

Then, the images in the dataset were analysed. Initially, random images were fetched and displayed along with its features. It was noticed that all the images have the same dimensions, 450px height and 600px width.

```
Image ID: ISIC_0031295
Lesion Class: mel
Age: 50.0
Gender: male
Localisation: upper extremity
Image Dimensions: (450, 600, 3)
```

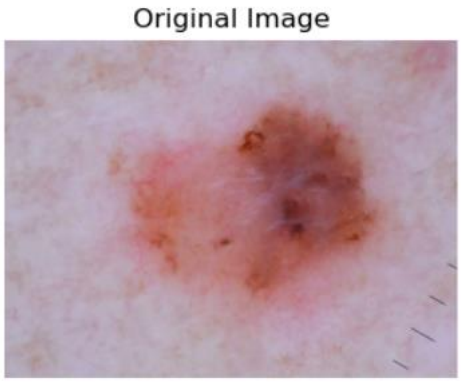


Figure 3: Sample Random Image Fetched

To confirm this, a bar chart was created by processing all the images in the dataset as shown in *Figure 4*.

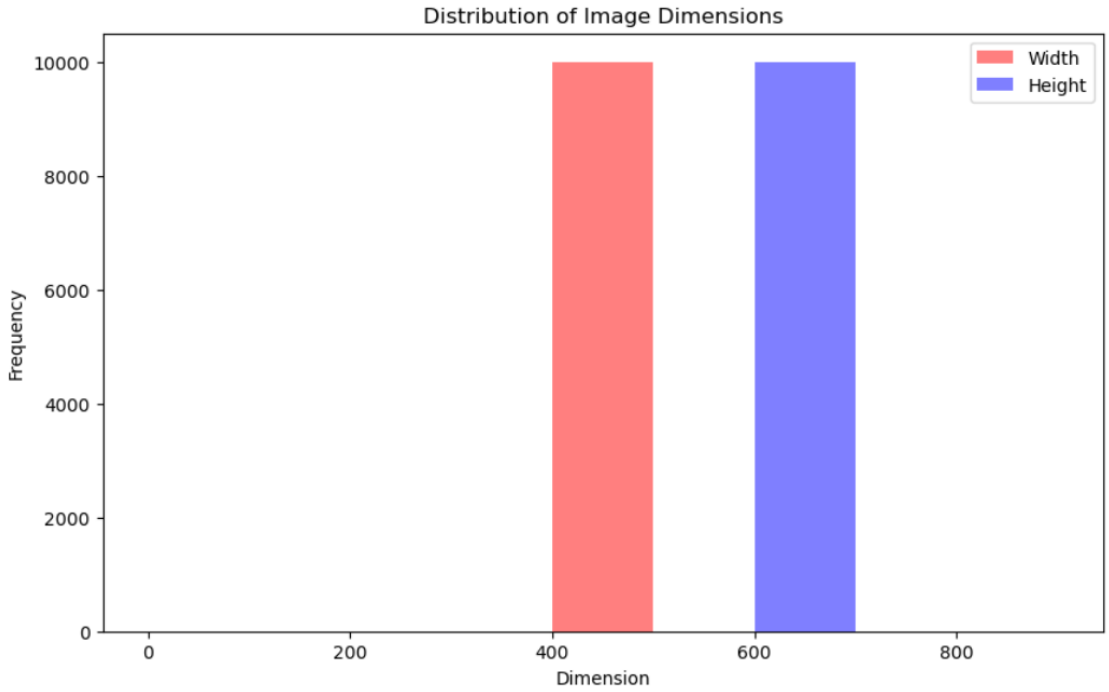


Figure 4: Image Dimension Distribution

For additional information, distribution of colour and pixel intensities of random 100 images were also obtained and visualised as histograms as shown in *Figure 5* and *Figure 6*.

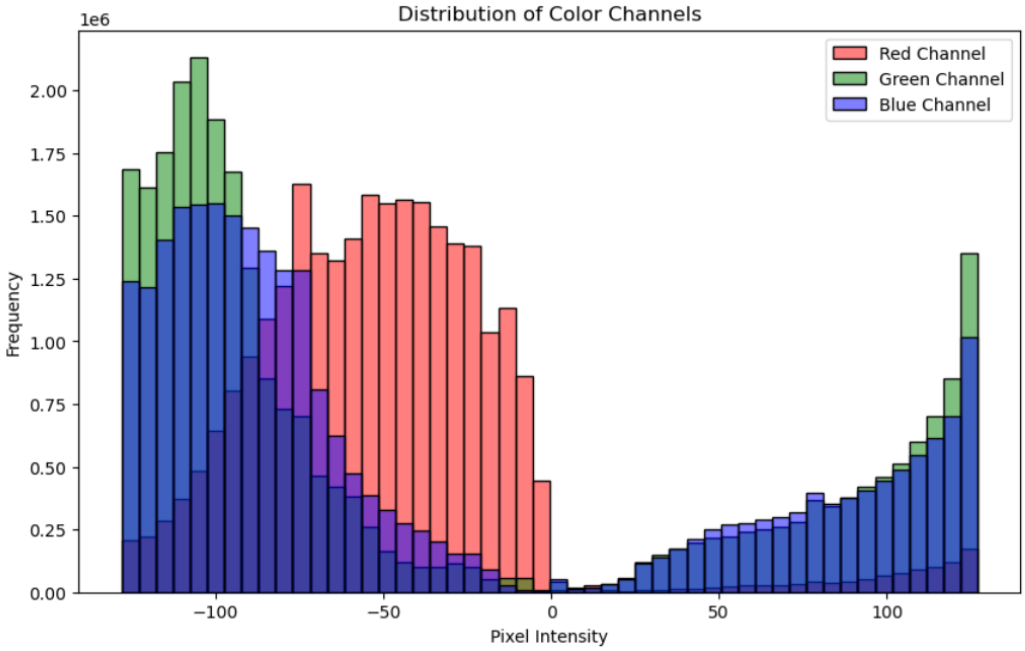


Figure 5: Colour distribution against Pixel Intensity

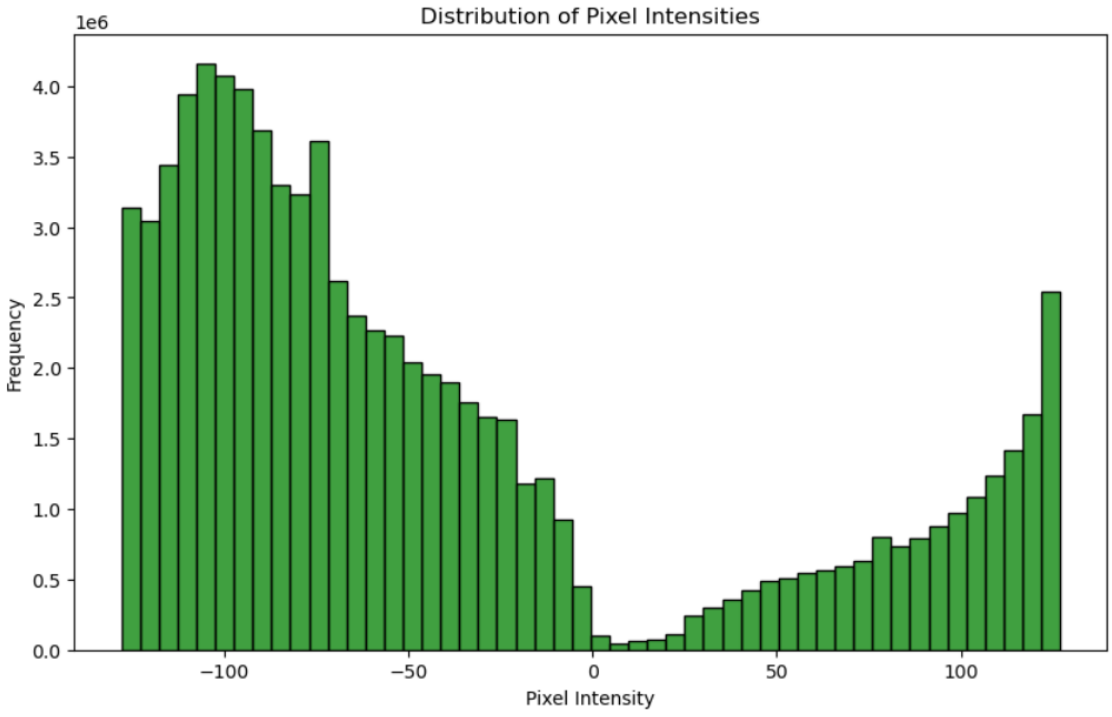


Figure 6: Frequency of Pixel Intensity

Limitations and Issues

There are several issues in the outcome of this project, mainly due to the limitations of the system used to process the dataset and build the network. Image processing and training the neural network models required high-performance computing systems and sufficient memory and storage resources. This is mainly due to the size of the dataset, which posed a challenge in terms of processing time and storage requirements. Also extracting large number of usable features from it required more memory.

Next section discusses the pre-processing and feature selection of the images, but unfortunately the extracted features were not used to train the model, mainly because the features were unable to fit into the CNN architectures used in this project. Therefore, the images were only resized, and colour space were converted from BGR image to RGB using OpenCV before fitting the data and training the model. The process behind extracting features is described in the next section only to express the effort put into it.

Implementation, Training and Results

Data Pre-Processing and Feature Selection

For the implementation described below, please refer to the ***Data Preprocessing.ipynb*** file.

Initially, metadata for the dataset and the target image were mapped and then splits the dataset into training and testing data 80:20 ratio. Then the minority classes were unsampled to balance the classes in the training dataset.

Next, using skimage library, the images' features were extracted and concatenated them into a feature vector. Then feature selection was performed on the concatenated images using the chi-squared test, but unfortunately due to memory constraints of the machine, only 100 most relevant features were selected.

Finally, the extracted train and test data were saved as *.npy* (numpy) files, to be easily loaded and reused in other files where different architecture models were trained. But, as mentioned in the Limitation section, these extracted features were unable to be used to train the algorithms as there were multiple issues while trying to fit the extracted features with the models, and reperforming these steps were not practical due to the resource limitations and time constraints. Hence, the images were simply resized to 128, 128 pixels and their colour space were converted from BGR image to RGB using OpenCV.

Training Overview

Models based on the four algorithms; EfficientNetB0, VGG, MobileNetV2 and InceptionResNetV2, were trained and tested with pre-processed images and the results for each of the algorithm were captured. To understand more about the implementation, refer to the Jupyter Notebook (***.ipynb)*** files of the respective algorithms, where all the graphs and values could be found.

Figure 7 shows the overall results for each algorithm, and it is pretty evident that MobileNetV2 and EfficientNetB0 received better accuracies in comparison to other models. Each model and their architecture are described below in detail.

Architecture	Number of Epochs	Accuracy	F1 Score	Precision	Recall
MobileNetV2	50	0.794807792	0.78968326	0.8276766	0.794807788
VGG	10	0.667998016	0.53503821	0.44622133	0.667998003
EfficientNet	50	0.797304034	0.78471141	0.78195455	0.797304044
InceptionResNet	50	0.667498767	0.5347984	0.44611055	0.667498752

Figure 7: Results Overview

EfficientNetB0

EfficientNetB0 is a state-of-the-art CNN architecture that achieves high accuracy while being computationally efficient. It uses a compound scaling method that jointly scales up the depth, width, and resolution of the network, while balancing the trade-off between accuracy and computational resources. The network is composed of several layers of convolutional and pooling operations, as well as a novel mobile inverted bottleneck convolutional block that uses shortcut connections and squeeze-and-excitation modules to enhance feature representation *(Tan and Le, 2019).*

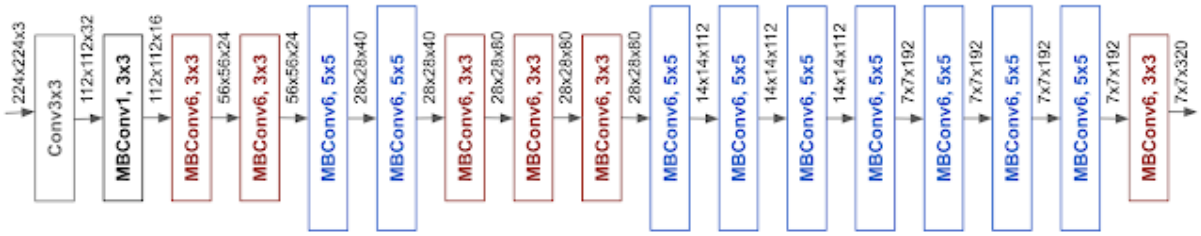


Figure 8: EfficientNetB0 architecture (Google AI Blog, n.d.)

EfficientNetB0's architecture was obtained from the keras library, and the model was trained for 50 epochs, and it uses the categorical cross-entropy loss function to calculate the error between the predicted and actual labels, and the Adam optimizer to update the model weights, with a batch size of 32 and a learning rate of 1e-4. This model produced one of the best accuracies of 0.794807791, when being run with the test dataset.

```
Test accuracy: 0.7973040342330933
F1 score: 0.7847114086605549
Precision: 0.7819545495434883
Recall: 0.7973040439340988
Confusion matrix: [[ 122   64    0   23    0   11    8]
 [  23 1270    0   34    2    7    2]
 [   3    9    8    3    0    3    2]
 [  30   88    0  100    0    3    5]
 [   1    7    0    0   13    0    0]
 [  12   14    1    4    0   57    5]
 [  14   13    1    6    1    7   27]]
```

Figure 9: EfficientNetB0 results

VGG

VGG (Visual Geometry Group) is a popular CNN architecture for image classification, which consists of several convolutional and pooling layers followed by fully connected layers. VGG16 uses small 3x3 convolutional filters, which allows it to learn more complex features while keeping the number of parameters manageable and be easily adapted to different image sizes and datasets (*Simonyan and Zisserman, 2014*).

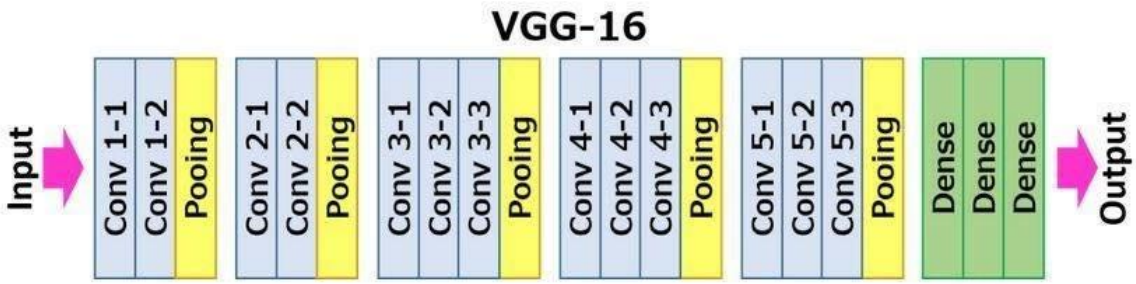


Figure 10: VGG16 Architecture (Rohini G, 2021)

VGG-16 model's architecture was defined in the code along with the correct input shape. Unfortunately, due to its complexity, only 10 epochs were managed to run using the available hardware resources without the kernel being interrupted itself. Hence, the results obtained for this algorithm could be biased when compared with other algorithms. And as expected, this model only produced an accuracy of 0.66799802.

```
Test accuracy: 0.6679980158805847
F1 score: 0.535038208924267
Precision: 0.446221332005985
Recall: 0.6679980029955067
Confusion matrix: [[   0  228    0    0    0    0    0]
 [   0 1338    0    0    0    0    0]
 [   0   28    0    0    0    0    0]
 [   0  226    0    0    0    0    0]
 [   0   21    0    0    0    0    0]
 [   0   93    0    0    0    0    0]
 [   0   69    0    0    0    0    0]]
```

Figure 11: VGG16 results

MobileNetV2

MobileNetV2 is a CNN architecture that is designed to be lightweight and efficient for mobile and embedded devices. The network consists of a sequence of convolutional and depth wise separable convolutional layers, which reduce the computational complexity and memory requirements of the network. This is done by inverted residuals and linear bottleneck layers to reduce the number of parameters while maintaining high accuracy (*Sandler et al., 2018*).

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 12: MobileNetV2 architecture (Tsang, 2019)

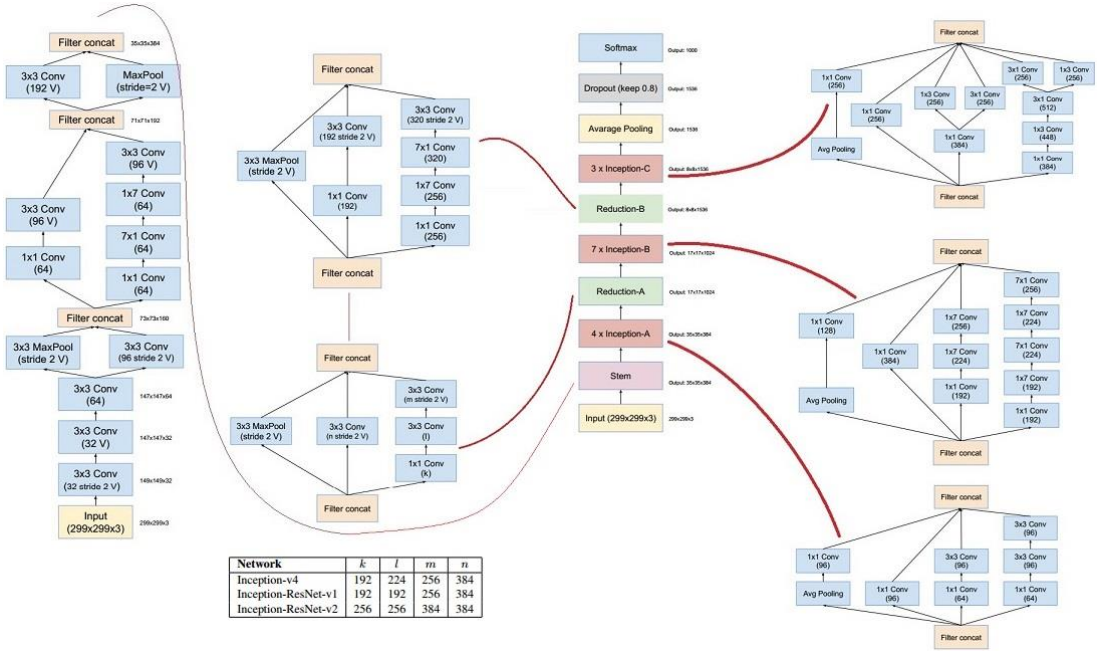
This model's architecture was obtained from keras library, and was trained for 50 epochs, and it uses the categorical cross-entropy loss function to calculate the error between the predicted and actual labels, and the Adam optimizer to update the model weights, with a batch size of 32 and a learning rate of 0.001. This model produced one of the best accuracies with the test data, of 0.7948077917.

```
Test accuracy: 0.7948077917098999
F1 score: 0.7896832622317308
Precision: 0.8276765960121745
Recall: 0.7948077883175237
Confusion matrix: [[ 186   32    0    4    0    6    0]
 [  85 1225    2   18    2    6    0]
 [  14    4    8    0    0    2    0]
 [  66   69    2   86    0    3    0]
 [   2    1    0    0   17    0    1]
 [  30   11    0    0    0   51    1]
 [  35    9    0    1    0    5   19]]
```

Figure 13: MobileNetV2 results

InceptionResNetV2

InceptionResNetV2 is a deep CNN architecture that combines the Inception and ResNet architectures to achieve high accuracy while being computationally efficient. The Inception module uses multiple parallel convolutional filters with different sizes to capture features at different scales, while the ResNet module uses shortcut connections to facilitate gradient flow and prevent vanishing gradients. The network also uses batch normalization and dropout regularization to improve generalization performance (*Szegedy et al., 2016*).



Network	k	l	m	n
Inception-v4	192	224	256	384
Inception-ResNet-v1	192	192	256	384
Inception-ResNet-v2	256	256	384	384

Figure 14: InceptionResNetV2 architecture (yeephycho, n.d.)

InceptionResNetV2 from keras library, was trained for 50 epochs, and it uses the categorical cross-entropy loss function to calculate the error between the predicted and actual labels, and the Adam optimizer to update the model weights, with a batch size of 32 and a learning rate of 1e-4. Unexpectedly, this produced the least accuracy, 0.66749876, even though existing research mention that this produces high accuracy (*Lan et al., 2022*). This could be due to the limitations with image processing in this project.

```
Test accuracy: 0.667498767375946
F1 score: 0.5347984012005943
Precision: 0.4461105544480482
Recall: 0.6674987518721918
Confusion matrix: [[   0  228    0    0    0    0    0]
 [   0 1337    0    0    0    1    0]
 [   0   28    0    0    0    0    0]
 [   0  226    0    0    0    0    0]
 [   0   21    0    0    0    0    0]
 [   0   93    0    0    0    0    0]
 [   0   69    0    0    0    0    0]]
```

Figure 15: InceptionResNetV2 results

Conclusions

In conclusion, the HAM10000 dataset has been a valuable resource for skin lesion classification research. CNN based models, have shown promising results in accurately classifying skin lesions using image data. There are multiple ways this research could be improved, mostly by the availability of computing resources, access to high-performance computing systems and sufficient memory and storage resources. Even though the extracted features were not used to train any models, the selected features itself would not have promised a more accurate model, mainly since only 100 features were obtained, while with a high performing system, at least 5000 important features could have been extracted. Another major issue was the fact that all the images had to be resized to (128,128) pixels due to memory constraints, which could have possibly attributed for the low accuracies produced by the models, since the input shape was small. Nevertheless, EfficientNet and MobileNetV2 model managed to produce very high accuracies of almost 80% even with all the issues and limitations with the pre-processed images, proving that these approaches could be very efficient when the concerns faced in this project are addressed.

References

Shellenberger, R., Nabhan, M. and Kakaraparthi, S. (2016). Melanoma screening: A plan for improving early detection. Annals of Medicine, 48(3), pp.142–148. doi:https://doi.org/10.3109/07853890.2016.1145795.

Sandler, M., Howard, A.W., Zhu, M., Andrey Zhmoginov and Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv (Cornell University). doi:https://doi.org/10.48550/arxiv.1801.04381.

Tsang, S.-H. (2019). Review: MobileNetV2 — Light Weight Model (Image Classification). [online] Medium. Available at: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c.

Tan, M. and Le, Q.V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. [online] arXiv.org. Available at: https://arxiv.org/abs/1905.11946.

Google AI Blog. (n.d.). EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling. [online] Available at: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html.

Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. [online] arXiv.org. Available at: https://arxiv.org/abs/1409.1556.

G, R. (2021). Everything you need to know about VGG16. [online] Medium. Available at: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918.

yeephycho. (n.d.). A Note to Techniques in Convolutional Neural Networks and Their Influences III (paper summary). [online] Available at: https://yeephycho.github.io/2016/08/31/A-reminder-of-algorithms-in-Convolutional-Neural-Networks-and-their-influences-III/.

Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv:1602.07261 [cs]. [online] Available at: https://arxiv.org/abs/1602.07261v2.

Lan, Z., Cai, S., He, X. and Wen, X. (2022). FixCaps: An Improved Capsules Network for Diagnosis of Skin Cancer. IEEE Access, [online] 10, pp.76261–76267. doi:https://doi.org/10.1109/ACCESS.2022.3181225.