Project for the course Deep Learning
University of Piraeus - NCSR Democritos

Skin cancer detection using Deep Learning models

Iraklis Evangelinos

July 3, 2022

Abstract

The aim of this project is to explore the capability of Deep Learning models in detecting skin cancer through images. This is done by using the well established International Skin Imaging Collaboration(ISIC) 2018 dataset, which consists of dermatoscopic images, accompanied by a dermatologist's diagnosis, which is used as a ground truth. In this project a CNN model is trained and fine tuned by changing it's architecture and hyperparameters and the performance of the best version is compared to the pretrained VGG19 and ResNet50 models, when applied to the same task.

1 Introduction

Skin cancer is the most widespread form of cancer, with Melanoma specifically acounting for 75% of all skin cancer deaths, despite being one of the rarest forms of skin cancer. In the case of melanoma, early detection and treatment results in 98% recovery rate, while this statistic plummets if the cancer is allowed to progress. It is therefore imperative for means to help with early detection to be developed, as this could result in tens of thousands of lives saved each year(in the US alone 7.000 deaths from skin cancer are recorded each year).

In the future, AI solutions could be used as another tool to aid dermatologists with diagnosis, or as a means to provide readily available suggestions on whether a lesion needs medical attention or not(e.g. by implementing an AI model in a smartphone application).

An example of such a model is the one created as a product of this project, while keeping in mind that more complicated, exotic models, with more computing resources and bigger datasets can provide much better results than the CNN discussed in this report.

1.1 The Convolutional Neural Network

One of the key technologies in the field of image processing and classification as far as Deep Learning models are concerned, is that of the Convolutional Neural Network(CNN). A CNN is a type of Neural Network that specialises in tasks where the source material is images or text, due to its ability to capture context, identify patterns and encode that information in later layers, therefore drastically reducing the amount of nodes needed to process the data given.

This is especially true in the case of images where the entire image is given as an input, therefore making the use of fully connected NNs impossible due to the amount of nodes and layers needed. As opposed to that a CNN is able to filter the input through one or more convolutional layers, which are then followed by pooling layers, which are able to identify patters. Moreover, each layer

is capable of identifying more abstract information than the previous layers. For example the first layers may be able to identify the presence of a line or a curve, whereas layers 'deeper' in the network can identify the presence of the pattern 'o' present in the lower halves of the numbers 6 and 8.

A CNN model consists of one or more convolutional layers, which have a fixed size kernel (this is a hyperparameter), which is applied step by step to the entire image, as shown in Fig.1 . The weights of the kernel are parameters to be learned by the model during training. The convolutional layers are accompanied by pooling layers, where an aggregate function is applied in order to reduce the size of the grid representing the image, while also preserving any patterns recognised by the convolutional layers, so that it can be used deeper in the network.

The last part of a CNN, consists of one or more fully connected(FC) layers, the last of which is responsible for producing the output based on the information propagated by the previous layers.

1.2 Transfer learning

Transfer learning is a method to allow for large, pretrained models that have been proven capable in a certain task, to 'transfer' that knowledge and inferencing capability in a different dataset which has similar characteristics to the ones originally used during training. This allows for models that would otherwise be prohibitively resource-hungry to be used without much computational cost in the task at hand. This is achieved by removing the last layers of the pretrained model, which are adapted for the original task and dataset, and then to connect the remaining, 'headless' model to one of more layers(usually FC), so that the new layers can be trained on the representation given by the pretrained network for each input, thus adapting the original model to the new task.

In this project Transfer Learning is used to provide a baseline, i.e. a way to measure and compare our model's performance in the task at hand, by using the performance of well established, very deep models, such as ResNet50 and VGG19.

1.3 Machine learning models on image classification

A good practice when a Deep Learning solution is to be deployed on a specific task is to try and apply simpler Machine Learning models, so as to provide a baseline with which to compare the Neural Network, as well as to provide insight as to whether a Deep Learning model is necessary for the performance required, or whether the same performance can be achieved by simpler, more traditional models such as SVMs, Decision Trees etc.

In the domain of image classification, a method to convert the original image to a vector with reasonably low dimensionality is needed before a ML model can be used, so image extraction is a necessary step. In this specific case, NNs can be used as a way to encode the image into embeddings which are then used by the ML model to perform the task at hand. The other approach involves a painful amount of hand picking methods to extract information from the image. Both those solutions were deemed not viable, especially when the fact that CNNs need no feature extraction at all, is taken into consideration.

For this reason, transfer learning with pretrained models was chosen as a baseline and an expected 'upper bound' to the performance of our model, as opposed to the 'lower bound' which the ML models would provide.

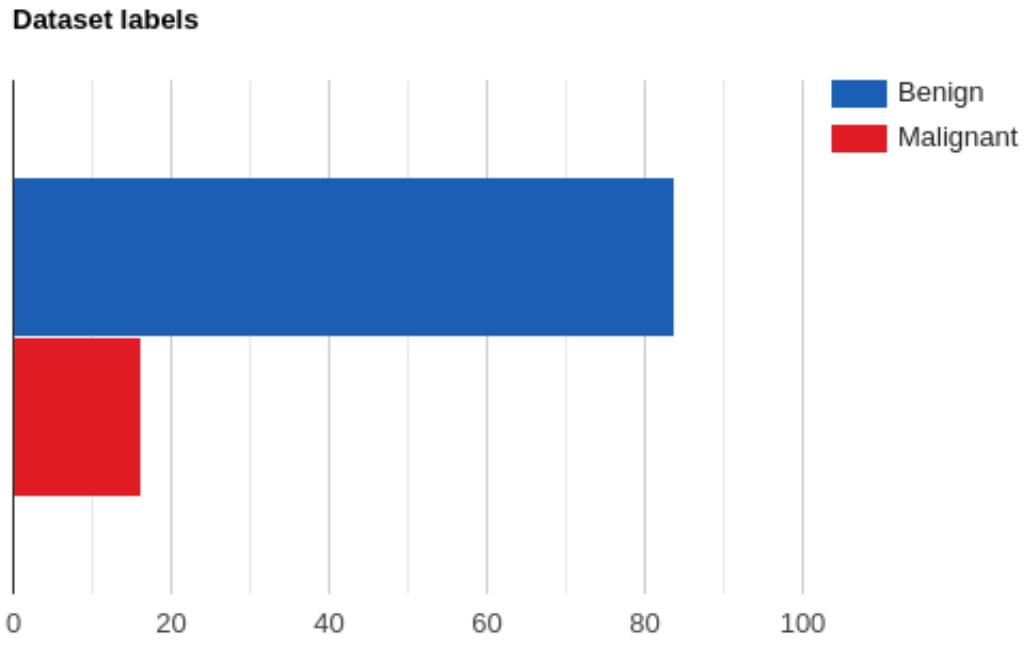


Figure 1: The distribution of the dataset's labels shows that it is heavily imbalanced

2 Proposed method and approach

2.1 Dataset

The dataset used in this project is the ISIC2018 dataset, which contains 10,015 dermatoscopic images of skin lesions. The lesions are classified among 9 different classes, 7 of which are benign and 2 of which correspond to a specific type of skin cancer. The aim of this project is to build a binary classifier capable of distinguishing between a benign and a malignant lesion, therefore the 9 classes are converted into 2, benign and malignant.

It is worth noting that the way the photos were taken ensures that the lesion is always in the center of the image and that brightness and contrast are all calibrated for the entire dataset. The dataset is split into a train and a test subset.

2.1.1 Pre-processing

As far as pre-processing goes, the fact that the dataset is already processed to some extent alevi-ates the need for some steps that would otherwise be necessary, such as cropping the images and normalising for brightness and other image features. The pre-processing applied to the dataset consists of two steps.

On the first step, the images are resized to 112 by 112 pixels, in order to reduce the unnecessary complexity of the CNN. On top of that, the fact that the test set is heavily imbalanced creates the need for data augmentation measures to be takne in order to make it as balanced as possible.

In the data augmentation step the malignant images of the train set are augmented using horizontal and vertical flip, zooming and random rotation. Note that no transformations that alter the shape and colour of the lesions are applied as this would skew the results

The final result was a test set comprised of 8059 benign and 7548 malignant images.

2.1.2 Visualisation

Data visualisation techniques were applied to the dataset in the beginning stages of the project in order to make some sense of the data, to arrive to conclusions about distinguishing features of the lesions and to make sure that the images were handled appropriately(i.e no color distortion was applied). The following figure shows that in some cases distinguishing between malignant and benign cases can be easy even for the untrained eye, while other cases seem to be extermely hard to determine. This is in line with real world cases, where the differences are subtle enough for other forms of examination to be needed. One advantage of the CNN and NNs in general is the ability to extract that information that to the untrained human eye would go unnoticed.

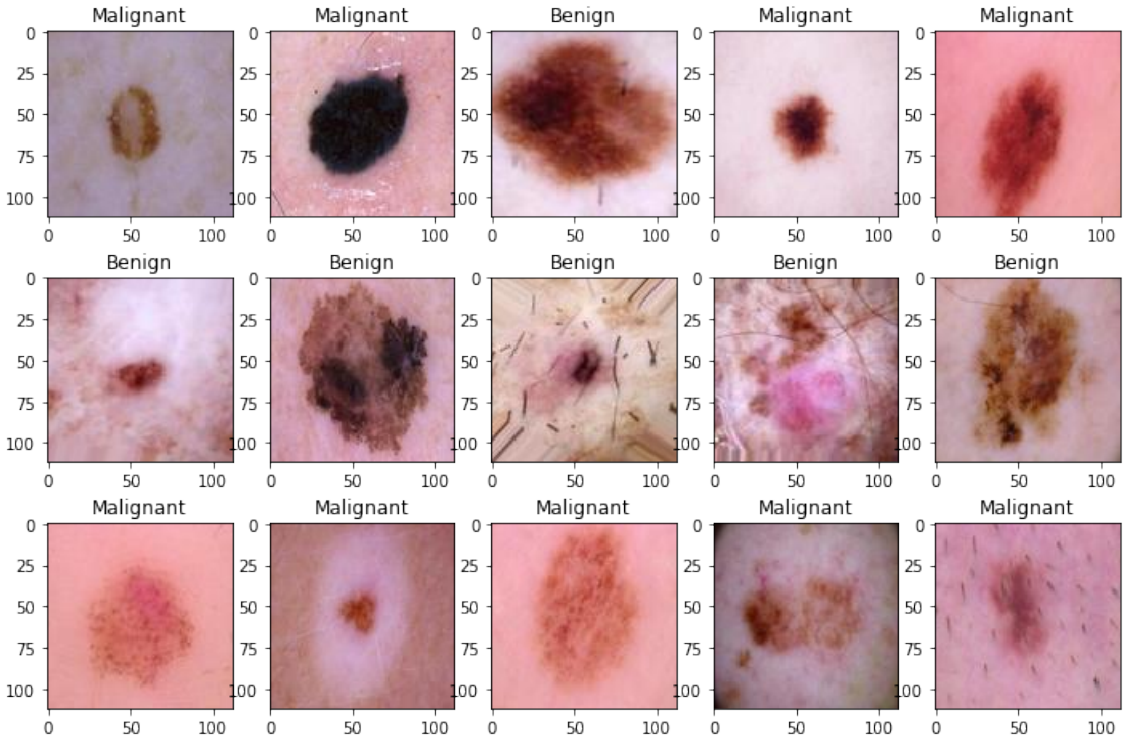


Figure 2: A small subsection of the dataset, along with each image's label.

2.2 The CNN model

The CNN model is the center piece of the entire project, the main aim is to fine tune the architecture and the hyperparameters so that optimal results can be obtained. As it will later be demonstrated, enough fine tuning puts the model on par with much larger, pretrained models.

The CNN model's architecture ranged from 2-4 convolutional and 2-4 fully connected layers, the final architecture is shown below.

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 112, 112, 64)	1792
conv2d_13 (Conv2D)	(None, 112, 112, 64)	36928
max_pooling2d_9 (MaxPooling 2D)	(None, 56, 56, 64)	0
conv2d_14 (Conv2D)	(None, 56, 56, 64)	36928
max_pooling2d_10 (MaxPoolin g2D)	(None, 28, 28, 64)	0
conv2d_15 (Conv2D)	(None, 28, 28, 64)	36928
max_pooling2d_11 (MaxPoolin g2D)	(None, 14, 14, 64)	0
dropout_3 (Dropout)	(None, 14, 14, 64)	0
flatten_3 (Flatten)	(None, 12544)	0

Figure 3: The CNN model's summary.

2.3 The ResNet50 model

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It is a very big model, which is used for such tasks. The model uses residual mappings, meaning that connections between layers can skip the ones in between. The model was chosen to provide a baseline for what extremely large models can produce on this specific dataset. The model was imported headless and 3 fully connected layers were stacked on top.

2.4 The VGG19 model

The significantly more simple, in terms of architecture and depth, VGG19 model was also used, as its complexity lies between the ResNet and the CNN model, while also being renowned for the spectacular results it produces in image classification among other tasks. The model was imported headless and 1 fully connected layer was stacked on top.

3 Experimental results

3.1 Evaluation mehods

Because the task at hand was essentially a heavily imbalanced, binary classification one, this needed to be taken into consideration when the models were evaluated, so that the metrics and the results would be as comprehensive as possible. For this reason the loss function was defined to be binary crossentropy.

At the same time the dataset's imbalance made metrics such accuracy not useful to evaluate the models. The choice was finally made to use the AUC metric to evaluate the models. On top of that other metrics were provided, such as Precission, Recall and F1 measure, which were taken into account to the extent that they provided some insight to the model's performance.

3.2 Results

The CNN model was allowed to train for 30 epochs, while the two pretrained ones were trained for 20 epochs with all but the topmost layers frozen. Learning rate decay with a patience of 5 epochs with respect to validation AUC performance was implemented to help with the model's convergence to a loss minimum.

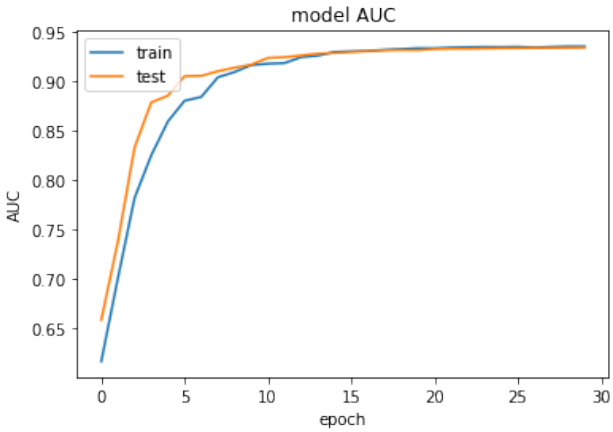
4 Conclusion

From the testing section and the accompanying graphs we can draw the following conclusions:

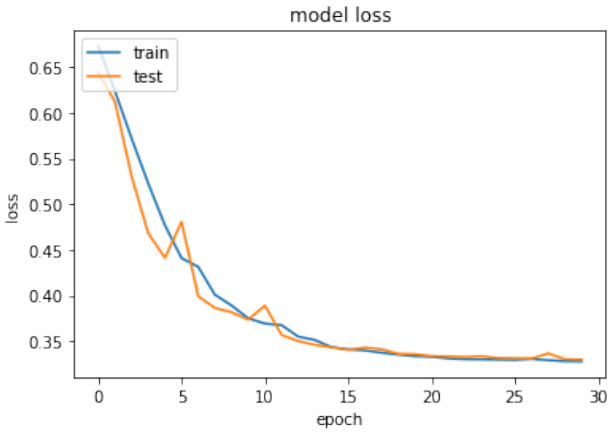
- The CNN model's performance increased dramatically through fine tuning the parameters.
- Compared to the pretrained state of the art models it was able to perform fairly well, given the size of the parameters of the final model was 3.3million.
- Through fine tuning, the CNN model managed to match the VGG19 model, and to outclass the ResNet50 model. An important note is that even though all models underwent fine tuning, none compared to the extent the CNN model did. Despite that, it is a remarkable accomplishment for a much simpler model.
- The author speculates that through more fine tuning of the parameters the VGG19 model, as well as the ResNet50 one can outperform the CNN model. To prove the hypothesis more layers would need to be unfrozen, which requires more processing resources than the ones available at the moment.
- An interesting note was that fine tuning the final layer gave VGG19 1% improvement during testing by adjusting the FC layer and 1% by unfreezing the top 4 layers.

Apendix: Performance metrics

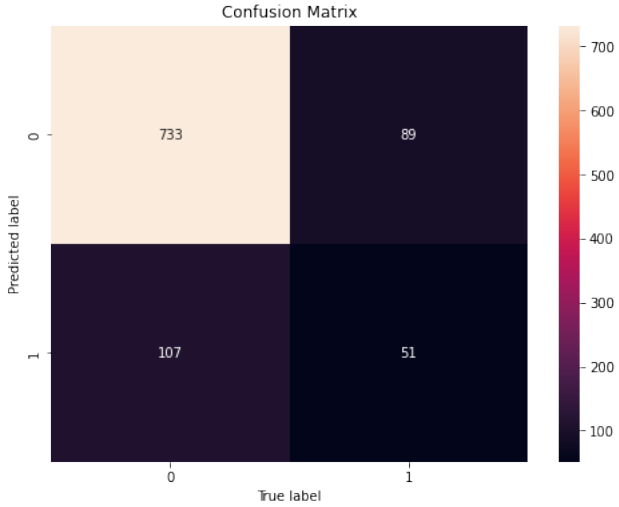
CNN model



(a) CNN training AUC graph



(b) CNN training loss graph



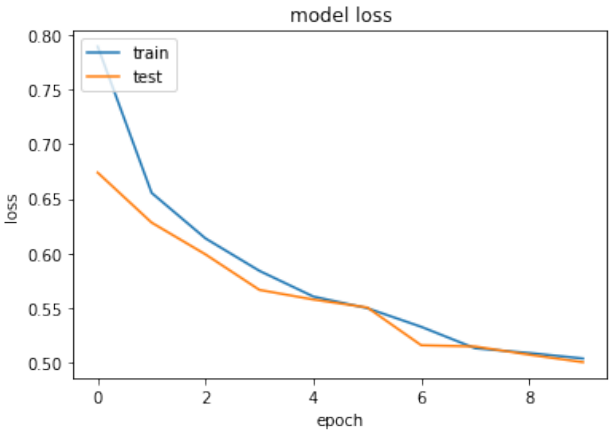
(c) The CNN's confusion matrix

	0.0	1.0	accuracy	macro avg	weighted avg
precision	0.89	0.32	0.8	0.61	0.81
recall	0.87	0.36	0.8	0.62	0.80
f1-score	0.88	0.34	0.8	0.61	0.80
support	840.00	140.00	0.8	980.00	980.00

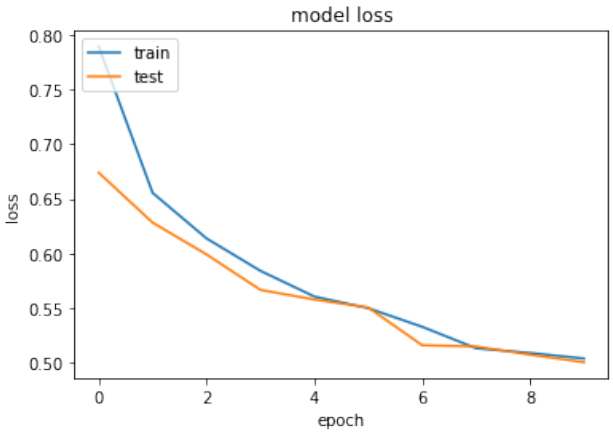
(d) CNN accuracy, recall, precision and F1 score

Figure 4: The performance metrics of the CNN model.

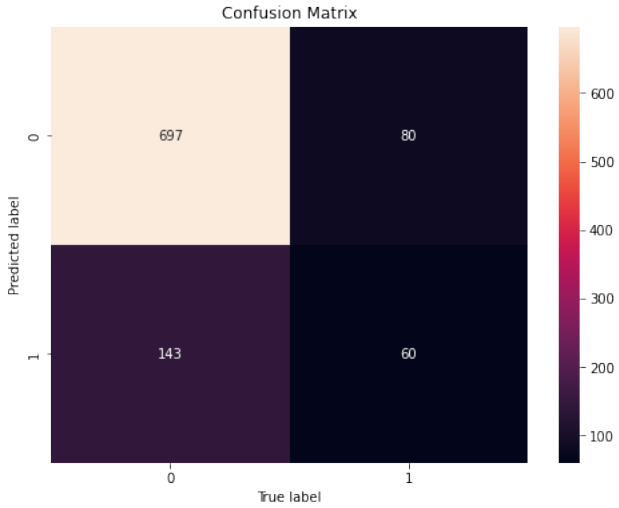
ResNet50 model



(a) ResNet50 training AUC graph



(b) ResNet50 training loss graph



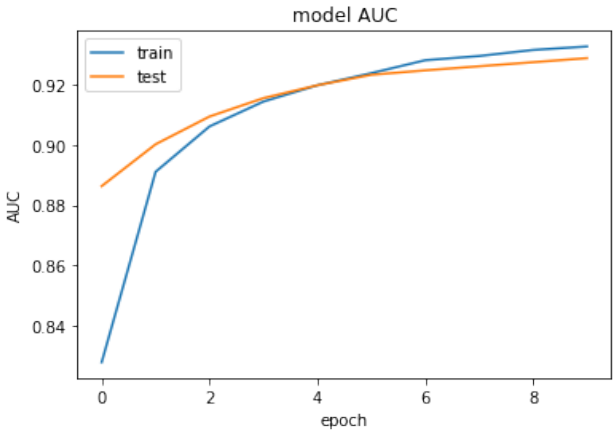
(c) The ResNet50's confusion matrix

	0.0	1.0	accuracy	macro avg	weighted avg
precision	0.90	0.30	0.77	0.60	0.81
recall	0.83	0.43	0.77	0.63	0.77
f1-score	0.86	0.35	0.77	0.61	0.79
support	840.00	140.00	0.77	980.00	980.00

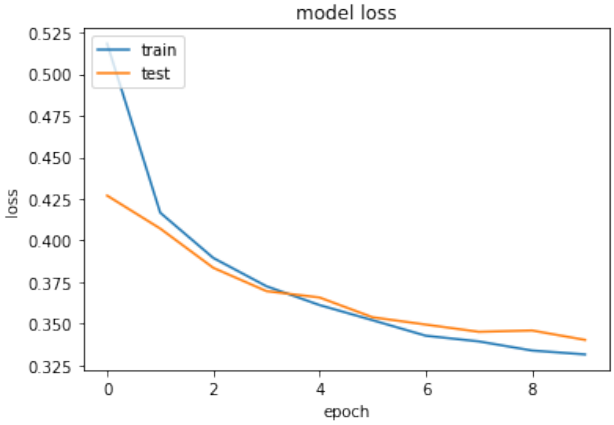
(d) ResNet50 accuracy, recall, precision and F1 score

Figure 5: The performance metrics of the ResNet50 model.

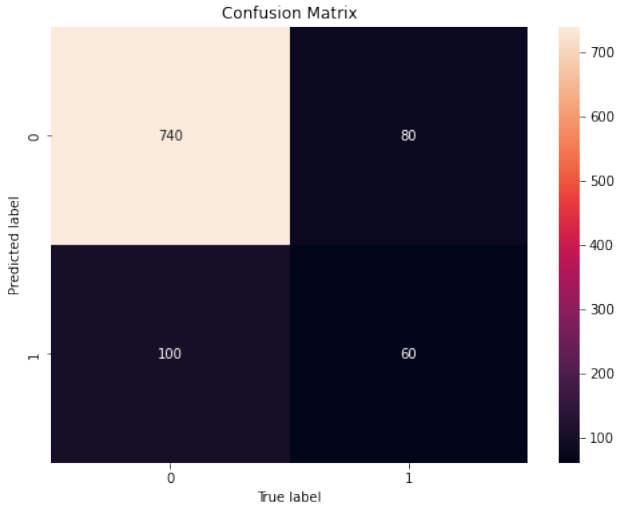
VGG19 model



(a) VGG19 training AUC graph



(b) VGG19 training loss graph



(c) The VGG19's confusion matrix

	0.0	1.0	accuracy	macro avg	weighted avg
precision	0.90	0.38	0.82	0.64	0.83
recall	0.88	0.43	0.82	0.65	0.82
f1-score	0.89	0.40	0.82	0.65	0.82
support	840.00	140.00	0.82	980.00	980.00

(d) VGG19 accuracy, recall, precision and F1 score

Figure 6: The performance metrics of the VGG19 model.