CNN Model Development – 2018

# Overview

Convolutional Neural Networks are currently the most popular and effective technique for image classification. In this project, we aim to develop a CNN model for classifying Skin Lesion images into 7 different classes of lesions. There are various CNN models developed by companies like Google and researchers based on ImageNet dataset. We have explored some of these models by training them on ISIC 2018 Skin Lesion Image dataset to verify whether they provide expected accuracy. Moreover, we have developed custom models based on study of various research papers. In this report, we have described all the attempts made to develop a robust CNN model for classification.

# Data Description

The ISIC – 2018 training dataset consists of 10,015 dermatoscopic images of skin lesions belonging to different cases of skin cancers. The ground truth of these images is stored in another dataset which consists mapping of these images to their corresponding labels(skin lesion class). There are 7 cases namely :

* MEL - Melanoma
* NV - Melanocytic Nevus
* BCC -  Basal Cell Carcinoma
* AKIEC -  Actinic Keratosis
* BKL - Benign Keratosis Lesion
* DF - Dermatofibroma
* VASC - Vascular Lesions

# Pre- Processing

Apart from the pre-processing we applied to enhance quality of images of dataset, we identified 9 images that were not recoverable in a good quality form due to too much hair. We have deleted these images from the dataset as they were causing reduced accuracy.

Below are the images we removed from different classes:

***BCC:***

A picture containing domestic cat

Description automatically generatedA close up of a person's eye

Description automatically generated with medium confidenceClose-up of a person's eye

Description automatically generated with medium confidenceA close up of a dandelion

Description automatically generated with medium confidence

A picture containing pink, rodent, mammal

Description automatically generatedClose up of a person's hair

Description automatically generated with low confidenceA picture containing domestic cat, rodent, vegetable

Description automatically generated

***BKL:***

A picture containing mammal, rodent

Description automatically generated

***MEL:***

A picture containing mammal, rodent, close, domestic cat

Description automatically generated

# Model Development

We did extensive research to understand the intuition behind making a robust CNN model especially for Skin Lesion classification. Most of the research work we reviewed were performing binary classification to detect whether a Lesion is Melanoma or not. These models achieved good accuracy due to binary nature of classification. Some of the research work was for all the 7 classes of ISIC dataset however, the dataset used to train was a small subset of 10k images that are available in ISIC archives. Although the models developed in these papers achieved good accuracy, they were not generalized model since the amount of data used to train was less.

As we developed an idea about the purpose of different convolution layers, number of filters, pooling and dropout, we were able to tweak proposed and pre-implemented models to train them for our ISIC 2018 dataset.

The following is a detailed description of different attempts made to develop an efficient model along with limitations of each model.

## **Proposed Model by researchers at Telkom University, Bandung, Indonesia[[1]](#footnote-1)**

***Model Description:***

CNN model proposed by researchers at Telkom University was first utilized to develop a CNN model for ISIC 2018 images. The researchers developed model for 4 different classes of images while for our dataset, we tweaked the model to work for 7 classes of output layer. We used this model as a base reference because it was used for skin lesion classification & achieved good accuracy. Also, it is a simpler model for implementation and understanding.

***Model Architecture:***

The following is the layer-by-layer description:

Text

Description automatically generated with medium confidence

* CN Layers : x3, Kernel: 3x3

Layer 1: Filters x16

Layer 2: Filters x32

Layer 3:Filters x64

* Max Pooling Layers: 2x2
* Dropout : 50%
* Fully Connected Layer : x1
* Activation:

CNN: RELU

Output: Softmax

Optimizer: ADAM

Padding: SAME

Loss Metric: Categorical Cross

Entropy

Accuracy Metric: Accuracy

***Results:***

Number of Epochs : 50

Overall Accuracy (f1-score) : 75%

Calendar

Description automatically generated

## **Modified Proposed Model**

***Model Description:***

Since the above CNN model didn’t achieve good accuracy and was a basic model with few layers of convolution and no ANN layer, we enhanced the model by adding more layers of Convolution and ANN dense layers. This dramatically increased the number of trainable parameters. The intention was to enable model to learn more from images. Also, the accuracy achieved for NV and VASC classes was higher as compared to others. Thus, we developed model to tune hyperparameters only for rest of the 5 classes.

***Model Architecture:***

The following is the layer-by-layer description:

A picture containing graphical user interface

Description automatically generated

* CN Layers : x6, Kernel: 3x3

Layer 1,2 : Filters x32

Layer 3,4 : Filters x64

Layer 5,6 : Filters x128

* Max Pooling Layers: 2x2
* Dropout : 50%
* Fully Connected Layer : x4

3 Layers: 512 Nodes

Output Layer: 5 nodes

* Activation:

CNN: RELU

Dense: RELU

Output: Softmax

Optimizer: ADAM (lr=0.001)

Padding: SAME

***Results:***

The results below show that the model accuracy is too low. Also, it started decreasing after 10 Epoch, so we stopped the training.

Text

Description automatically generated with medium confidence

## **RESNET50 Model**

***Model Description:***

RESNET50 is one of the most popular models used for image classification. It is a model which is originally trained on ImageNet dataset where it achieved 92.1% Top-5 accuracy. We utilized its architecture and trained it using ISIC dataset. We didn’t use the pre-trained weights of ImageNet since they are not relevant to Skin Lesion images.

***Model Architecture:***

The model architecture is huge as it contains 50 DCNN layers with Max Pool, Average Pool and Batch Normalization layers etc.

***Results:*** Number of Epochs = 20

Calendar

Description automatically generated

## **InceptionV3**

***Model Description:***

Like RESNET50, InceptionV3 is another popular model developed by GoogleLeNet used in Transfer learning. Various research papers described the usage of this model for skin lesion multiclass classification with sufficiently good accuracies.

***Model Architecture:***

The model architecture is huge and is 189 layers deep. We did not use pre trained weights from ImageNet Datasets. Instead, we trained the model based on our ISIC 2018 dataset.

***Results:***

Number of Epochs = 20

Calendar

Description automatically generated

## **Ensemble 1: Basic Proposed Model Ensemble**

***Model Description:***

Based on few research papers, we got an intuition about using Ensemble of CNN’s to develop our classification model. Thus, we applied the ensemble learning technique to develop a robust model. Similar to ensemble learning techniques of Machine Learning like Random Forest where each Random Forest is an ensemble of decision trees trained on randomly picked different train-validation split of dataset, we utilized Proposed Model as specified in ‘1’ above and developed an ensemble of 10 such CNN models each of which was trained on a randomly picked Train-Validation Split (based on different random seed). The ensemble was used with Max Voting technique to classify the images while testing. Below is a step-by-step approach of building ensemble of CNN models:

* 1. Step 1: Create Base CNN Model:  
     We used the model proposed in research paper[[2]](#footnote-2) which was a good model for Skin Lesion classification as suggested in paper. The model architecture is described above in [Model 1](#_Proposed_Model_by)
  2. Step 2: Create a List of 10 CNN Base Models and compile them using metrics described in [Model 1](#_Proposed_Model_by). This list is named ‘ensemble’ in the code.
  3. Set the values of following hyperparameters to be used for model development
* TRAIN\_DIR : The directory on system containing pre-processed images
* Epochs: The number of epochs to run for training the ensemble. We chose this value as 50 based on our research review where most of the CNN models were trained for at least 50 epochs. This helps in better trained model development when the number of images is huge and variation among them is significant.
* Batch Size: The model is trained on batches of images. This is the number of images used to train a single forward & backward pass(epoch). If this value is too high, it can make the CNN take too long to achieve convergence(no more gain in accuracy). However, if it is too low, it will take more time for accuracy to stabilize (the accuracy will bounce up and down in subsequent epochs.). Therefore, based on several attempts, we fixed the value to be 64 for training the network better.  
  1. Run a for loop for 10 iterations. In each iteration, the following processes are executed:
     1. Pick a random seed value which is different in every iteration
     2. Shuffle the pre-processed images using this random seed value such that different set of images are picked in every iteration.
     3. Split the data with 70% images as training set and 30% as validation set.
     4. Perform Augmentations with an objective to balance the images in each class. The functions to augment images are separately written based on the number of images for each class.
     5. Keep the augmented training images and validation images in separate directories with following structure:

|  |  |
| --- | --- |
| augmented|  |-akiec  |-bcc  |-bkl  |-df  |-mel  |-vasc  |-nv | test |  |-akiec  |-bcc  |-bkl  |-df  |-mel  |-vasc  |-nv |

* + 1. Use ImageDataGenerator class of keras module in python to develop an efficient structure to provide training and validation data for model training. It also scales the images so that pixels are in range of 0 to 1 instead of 0 to 255.
    2. Train the model using the hyperparameters set above and save the model in the list ‘ensemble’
    3. Delete the augmented and validation images at end of each iteration for new set to be generated in the next iteration.
  1. Predict the class labels of test images. The working of the ensemble prediction is explained below:

**IMAGE INPUT**

Model 1

Model 10

Model 9

Model 8

Model 7

Model 6

Model 5

Model 4

Model 3

Model 2

**df**

**nv**

**nv**

**nv**

**nv**

**MAX VOTING**

**df**

**df**

**nv**

**nv**

**nv**

**nv**

***Results:***

Number of Epochs : 50 for each model in Ensemble  
 Overall Accuracy: 90%  
 Improvement from Base Model: +15%

Calendar

Description automatically generated

## **Ensemble 2: Proposed Model with 4 layers and 0.25 dropout**

***Model Description:***

The Ensemble developed with basic model provided with substantial improvement in Accuracy. However, the recall for some of the classes i.e., bkl, df and mel were low. Thus, to improve the recall of these classes, we applied certain modifications explained below.

***Modifications:***

1. Switch to Average Pooling from Max Pooling: We modified the pooling technique used after the first layer of CNN from Max pooling to Average pooling. This affects the pixel value being taken from portions of images to develop feature map. Average pooling helps in extracting overall features such as image contrast, whereas max pooling is useful for edge detection.
2. Added one more convolution layer of 128 filters. In CNN, an addition of a CNN layer results in extraction of more features from the images. In a dataset with large number of images with significant variations, increasing the convolution layers increases the accuracy and more details from the images are extracted. We did not increase more as it will increase the computational power usage and memory during training. It is a best practice to increase the number of filters in powers of 2 so we used 128 filters.
3. Dropping dropout to 0.25: We decreased the dropout after the third CNN +Pooling Layer from 50% to 25%. The dropout is responsible for deactivating neurons in layers such that model doesn’t overfit. But with a high value of 0.5 it may result in underfitting as we have large number of images with wide variations. Thus, we reduced the value to 0.25.
4. Decreasing Number of Epochs: From the training of Ensemble 1, we observed that the model doesn’t improve significantly after 30 epochs but consumes resources. Thus, we reduced the number of epochs for Ensemble 2.

***Model Architecture:***

The Ensemble architecture used here is same as that of Base Ensemble Model. The only difference is that the Base CNN model used is modified as given below:

Graphical user interface

Description automatically generated with medium confidence

***Results:***

Number of Epochs for each Model : 30  
 Overall Accuracy: 97%  
 Improvement from Ensemble 1: +7%

Calendar

Description automatically generated

## **Ensemble 3: Proposed Model with 4 layers and 0.4 dropout**

***Model Description:***

The overall accuracy in the previous model was very high with great precision and recall scores as well. This created a doubt regarding model being a bit overfitted due to low percentage of dropout. Thus, we developed yet another Ensemble model with 0.4 or 40% drop out.

***Modifications:***

1. Increase Dropout: In order to make sure that model is not overfitting, we increased the dropout to 40%. Increasing dropout results in lowering the chances of overfitting.

***Model Architecture:***

The model is same as before just with a difference in the dropout ratio.

***Results:***

Number of Epochs for each Model : 30  
 Overall Accuracy: 96%  
 Improvement from Ensemble 1: +6%  
 Change in Accuracy from Ensemble 2 after changing dropout: -1%

Calendar

Description automatically generated

1. https://iopscience.iop.org/article/10.1088/1757-899X/982/1/012005/pdf [↑](#footnote-ref-1)
2. https://iopscience.iop.org/article/10.1088/1757-899X/982/1/012005/pdf [↑](#footnote-ref-2)