

CSCI 569/ Math 569 Image Analysis and Recognition with Learning
Project Report: Melanoma Classification from Skin Lesion Images using
Convolutional Neural Network (CNN)

By

Subroto Singha (CWID: 50158985)

Emran Hossen (CWID : 50251820)

Supervisor: Dr. Nikolay Sirakov



Abstract

Convolutional Neural Network (CNN) classifies the images with high precision. Building a highly accurate CNN requires various hyper parameters tuning and optimal neural network architecture. In this project, we develop a CNN architecture such that it classifies the skin lesion images with high accuracy. Using three loss minimizing algorithm, namely SGD, RMSProp, ADAM our CNN classifies the images into two classes namely benign & malignant and it achieves Training accuracy: 98.15% and Validation accuracy: 89.53%. Further, we use five more evaluation metrics namely Sensitivity, Specificity, Precision, Loss and Validation loss to evaluate our models performance.



Contents

1 Introduction	6-7
2 Background	7-8
3 Mechanism behind CNNs.....	9
3.1 Forward Propagation Summary.....	9-10
3.2 Backward Propagation	10
3.3 Optimizing loss using Gradient Descent	10-11
3.3.1 Batch Gradient Descent	11
3.3.2 Stochastic Gradient Descent (SGD).....	11
3.3.3 Mini_batch Gradient Descent.....	11
3.3.3.1 Adaptive Moment Estimation (ADAM).....	11
3.3.3.2 Root Mean Square Propagation (RMSProp).....	12
4 Implementation	12-13
4.1 Dataset Description	13-15
5 Experimental Results	15-22
6 Conclusion	22-23
6.1 Future works.....	23
7 References.....	23-24

List of Figures



1 Rectified Linear Unit (ReLu).....	9
2 Sigmoid Function	10
3 Model Summary	13
4 Benign and Malignant Classes	13
5 Benign and Malignant Images (Random)	14
6 Training& Validation Accuracy and Training & Validation Loss using RMSProp (601:232).....	18
7 Training& Validation Accuracy and Training & Validation Loss using SGD (601:232).....	18
8 Training& Validation Accuracy and Training & Validation Loss using ADAM (601:232).....	19
9 Training& Validation Accuracy and Training & Validation Loss using RMSProp (2416:633).....	19
10 Training& Validation Accuracy and Training & Validation Loss using SGD (2416:633).....	20
11 Training& Validation Accuracy and Training & Validation Loss using ADAM (2416:633)..	20
12 Training& Validation Accuracy and Training & Validation Loss using RMSProp (3416:1133)..	21
13 Training& Validation Accuracy and Training & Validation Loss using SGD (3416:1133)...	21
14 Training& Validation Accuracy and Training & Validation Loss using ADAM (3416:1133).....	22



List of Tables


1 Confusion matrix.....	16
2 Model evaluation using (RMSProp).....	16
3 Model evaluation using Stochastic Gradient Decent (SGD).....	17
4 Model evaluation using (ADAM).....	17

1 Introduction

Skin cancer is one of the most common type of cancer. According to American Cancer Society, 75% of skin cancer deaths are due to melanoma. Thus, dermatologists evaluate every patient's moles and mostly the ugly ducklings are likely to be melanoma. Therefore artificial intelligence can play a vital role to aid the medical professional by correctly detecting the melanoma.

Exposures to UV radiations damage the melanocyte cells in human body and this causes a type of skin cancer called melanoma. According to World Health Organization (WHO), skin cancer is most prevalent cancer and it accounts for one third of all types of cancers around the world. Basal cell carcinoma (BCC), squamous cell carcinoma (SCC) and malignant melanoma (MM) are most common ones. In fact, BCC and SCC are responsible for non-melanocytic cancer and the majority of skin cancers are non-melanocytic but melanoma is the most dangerous because it rapidly metastasizes if it does not get diagnosed and treated at an early stage [3]. Thus diagnosis with Melanoma involves various organs in human body such as cancerous genes, hair color, elevated number of benign melanocytic nevi and dysplastic nevi.

Currently, skin cancer is a major public health problem with over 123,000 newly diagnosed cases worldwide every year. Surprisingly, 75% skin cancer deaths are due to melanoma. In addition, the American Cancer Society forecasts that there will be approximately 100,000 new melanoma will be diagnosed by end of 2020 and 7,000 people may die from this devastating cancer. In fact, melanoma is responsible for over 9000 deaths in the United States every year [8]. But if this cancer cannot be detected early the cost of treatment with later detection is high and about \$134,000 in its IV stage [2]. Among all available treatment strategies, dermatologists first detect melanoma by looking at images of the cancer and moles. Further they find the outlier lesions or “ugly ducklings” around the moles which may be a melanoma. If they are lucky, findings would be precise and accurate.

Dermatological accuracy can be increased using artificial intelligence (AI) based detection techniques. Existing AI approaches have not adequately considered this clinical frame of reference. Dermatologists could enhance their diagnostic accuracy if detection algorithms take into account “contextual” images within the same patient  determine which images represent a melanoma. If successful, classifiers would be more accurate and could better support dermatological clinic work.

Historically CNN have done tremendous job to do image classification such as breast cancer image classification [14], hyperspectral image classification [7], pap smear image classification [4], skin cancer and lesion classification [5],[6],[13] and so on. In fact, convolutional neural networks (CNNs) are almost accurate in classifying medical images while provided with sufficient images. In our case, skin lesion images suffice this task of melanoma classification in two binary classes namely benign and malignant.

In this project, we implement a CNN for binary classification and our contributions in this project are as follows:

1. We construct a neural network architecture that consists of four convolutional layers, four max pooling layers and two dense layers that give us highest accuracy.
2. We collected a huge dataset ISCI 2020 database 48.9 GB that has 33,126 jpeg images in 1024x1024 pixels where 533 are malignant images and rest are benign images.
3. Using SGD, ADAM and RMSProp optimizing algorithms, we compare our CNN considering various data splits.
4. We use dropout layer to overcome the over fitting issues.
5. We evaluate our CNN using five metrics namely Sensitivity, Specificity, Precision, loss and accuracy. We achieve highest 98.15% accuracy using data split 601:232 and RMSProp optimizing algorithm.

We divide entire project report into five sections. Section-1, discusses about the introduction and contribution of our project. Section-2 describes the related background. Section-3 presents mechanism behind CNNs. Section-4 shows our implementation and Section-5 shows our projects results. Finally Section-6 discusses the conclusion and future works.

2 Background

By looking at a glance via human eyes was the process of detecting skin cancer in earlier 20th centuries. Banerjee *et al.* (2020) discussed few examples of such strategies such as size, bleeding or ulceration [3]. But this strategy solely based on physicians' visual examination rather than any technical or advanced processes. Another detection process called Dermoscopy of epiluminescence microscopy was able to achieve 75% to 85% accuracy [15]. Biopsy was another process of detecting melanoma if a physician was not able to detect a mole as melanoma or not [3].

Recently, CNN based various classifiers came into play that helped the dermatologists to better detect the melanoma. In fact, Astudillo *et al.* (2020) achieved 95.23% accuracy to classify skin lesion images into two classes i.e. benign and malignant. Their CNN was constructed using 4 convolutional layers, ReLu activation function and a softmax classifier. They used ADAM, SGD algorithms to optimize the loss of the neural network. They added noise with SGD and ADAM to achieve more accuracy. ISIC 2018 skin lesion images was used to train and evaluate their proposed classifier. Albahar (2019) proposed a new model that groups skin lesions into benign or malignant lesions depending on a novel regularizer method [1]. Their binary classifier distinguished images between benign or malignant lesions. Considering cases such as nevus against melanoma lesion, seborrheic keratosis versus basal cell carcinoma lesion, seborrheic keratosis versus melanoma lesion, solar lentigo versus melanoma lesion they found Area under Curve (AUC) evaluation metrics were 0.77, 0.93, 0.85, and 0.86, respectively. Their model attained an average precision of 97.49%. Their model helped medical practitioners in grouping various skin lesions. Harangi (2018) presented a group of framework of CNNs for skin lesion grouping to attain a high group precision of dermoscopy images [8]. In their technique, they put together the outputs of the group layers of four different deep neural network architectures. Their final findings were that their technique exceeded the accuracy of the individual CNNs. In this work, they showed a new CNN model constructed on a novel regularizer. This CNN model carried out a better result than the state-of-the-art methods in terms of accuracy, the AUC & ROC curve, and sensitivity. Furthermore, Esteva *et al.* (2017) analyzed the performance of their CNN to 21 board-certified dermatologists with biopsy-proven clinical images including two critical binary grouping use cases [6]. Their deep learning CNN was better successful than the dermatologists when classifying the skin cancer via dermoscopy and photographic imaging. In addition Mahbod *et al.* (2017) presented an automatic computerized method that received great results for a dermatologist, and their technique employed learning and pre-trained CNN for the classification of skin lesions [10]. They used 2000 images from ISIC 2017 and reached an accuracy of 93.6%. Finally, analyzing the CNN models from literature, it is clear that the CNN models showed lesser computational complexity and better generalization capability than other models.

3 Mechanism behind CNNs

The entire CNN does the image classification very smoothly but under the hood there is basically heavy mathematics are involved. First the convolution layer is responsible for detecting edges in an image since we can differentiate an image from another image easily as the pixel value changes sharply at the edges. So, convolutional layer basically detects the features of an image to classify it. This first layer produces a linear function. We pass this information to the non-linear function such as ReLu (Rectified Linear Unit) function as to capture the high complex relationships. Then the max pooling layer brings the most dominating feature from an image. Then the flatten layer is used to make the 2D array into a 1D array. Then the fully connected layer is used to determine the classes. This layer performs two operation first linear transformation and then a non-linear transformation such as sigmoid. This whole process is known as forward pass then the backward pass is done to minimize the error.

3.1 Forward Propagation Summary

Step 1: Load the input images in a variable (say X)

Step 2: Define (randomly initialize) a filter matrix. Images are convolved with the filter

$$Z1 = X * f$$

Step 3: Apply the Sigmoid activation function on the result

$$A = \text{sigmoid}(Z1)$$

We may choose different non-linear function such as Rectified Linear Unit (ReLU) or Leaky ReLu. The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So, it can be written as:

$$f(x) = \max(0, x)$$

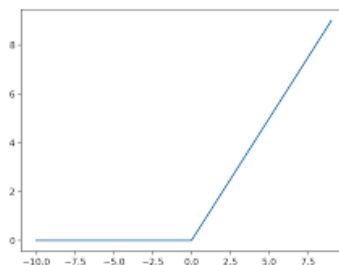


Figure-1: Rectified Linear Unit (ReLU)

Step 4: Define (randomly initialize) weight and bias matrix. Apply linear transformation on the values

$$Z2 = W^T \cdot A + b$$

Step 5: Apply the Sigmoid function on the data. This will be the final output

$$\hat{y} = \text{sigmoid}(Z2)$$

The equation for the Sigmoid function

$$f(x) = 1/(1 + e^{-x})$$

So, we pass Z2 to this sigmoid function that maps the output between 0 to 1:

$$f(z2) = 1/(1 + e^{-z2})$$

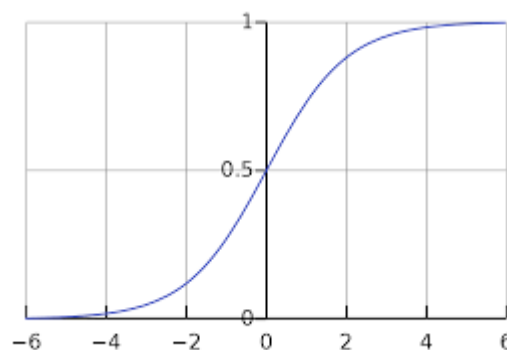


Figure-2: Sigmoid Function

3.2 Backward Propagation

In the backward propagation process, the model tries to update the parameters such that the overall predictions are more accurate. For updating these parameters, we use the gradient descent technique (RMSProp, Stochastic Gradient Descent (SGD), ADAM). Here is a generic equation for updating the parameter values:

$$W = W_0 - lr * \frac{dE}{dW}; lr = \text{learning rate}$$

3.3 Optimizing loss using Gradient Descent

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize the neural networks. To minimize the objective function say $J(\theta)$, where θ is model's parameter and $\theta \in R^d$, by updating the parameters in the opposite direction of the gradient of the objective function with respect to the parameters. We use a learning rate η to determine the step size to reach to the global minimum of the objective

function. Depending on the size of data usage, there are three types of gradient descent approaches.

3.3.1 Batch Gradient Descent

In brief, if we update the parameter θ using the entire training dataset, this process is known as batch gradient descent. The equation is as below:

$$\theta = \theta - \eta \nabla J(\theta)$$

3.3.2 Stochastic Gradient Descent (SGD)

In brief, if we update the parameter θ using each training example $x^{(i)}$ and $y^{(i)}$:



$$\theta = \theta - \eta \nabla J(\theta; x^{(i)}; y^{(i)})$$

3.3.3 Mini-Batch Gradient Descent

In brief, if we split the entire training dataset into mini-batches and update the parameter for every mini-batch of n training examples:

$$\theta = \theta - \eta \nabla J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

Now, comparing these three types of gradient descents, batch gradient descent is computationally expensive but it guarantees to reach to global minima for convex loss function and local minima to non-convex loss function. On the other hand, SGD reaches to global minimum based on optimum learning rate (lr) but may have overshooting problem for high lr value. Finally mini-batch gradient descent overcomes both problems and minimize the loss function by reaching to the global minima. Typically, we refer to SGD meaning mini-batch gradient descent and the size of various mini-batches are ranges from 50 to 256 based on computational device capability. In our project, we use this SGD and its variants such as ADAM and RMSProp and we keep the batch size 20 depending on the computational capability of our device.

3.3.3.1 Adaptive Moment Estimation (ADAM)

As we mentioned earlier, if there is no optimum learning rate (lr), we may face overshooting problem to reach to global minima of loss function. At the same time if lr is too small we may face longer training time. Momentum helps accelerate the SGD optimization.

ADAM computes adaptive learning rates for each parameter. Kingma *et al.* (2014) describe ADAM in more details [9].

3.3.3.2 Root Mean Square Propagation (RMSProp)

Similar to ADAM, RMSProp is basically an unpublished strategy but first introduced by Geoff Hinton in one of the Coursera classes. But this is another method for adaptive learning rate based on the dataset. Though later some papers have used this method such as in Zou *et al.* (2019)[16].

4 Implementation

Implementing CNNs require an architecture that best suits the given problem but there is no fixed architecture on hand that best fits the data. Thus it is a common practice to start following the architecture from the literature similar to the problems. In our case, our goal is to construct a CNN for image binary classification thus VGG like architectures are the automatic choice to start with. Since VGGs are the proven architecture in classifications such in Mateen et al. (2019) [11], Oraibi et al. (2018) [12] and so on. In VGG like architectures, there are basically sequence of convolutional layers (conv layers) and max pooling layers with ReLu activation functions and the number of neurons are increased by twice the previous neurons. Thus in our case, we randomly start with 32 neurons in belief of highest accuracy and increase to 64, 128 neurons simultaneously.

To implement our classifier, we have used the followings:

- Python
- Tensorflow 2.0
- Keras
- Jupyter Notebook
- Open CV

We have constructed our baseline CNN model for image classification. The architecture of our model has four Convolutional layers, four MaxPooling layers, one Flatten layer and two Dense layers. First layer of Convolution has 32 neurons and next three consecutive Convolution layers have 64, 128 and 128 neurons, respectively. Before the last dense layer, we have used 512 neurons. Our MaxPooling layer is 2x2 and all the activation functions are ReLu (Linear Rectified Unit). Final Dense layer has 'sigmoid' activation function to map the probabilities between 0 to 1 for predicting the images into two classes. Though in this project we are not going in details for overfitting and under fitting issues but we used a dropout layer of 0.5 to overcome the overfitting problems. The overall model summary is as follows:

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_5 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_6 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_6 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_7 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_7 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_8 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_8 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_2 (Flatten)	(None, 6272)	0
dense_3 (Dense)	(None, 512)	3211776
dense_4 (Dense)	(None, 1)	513
Total params: 3,453,121		
Trainable params: 3,453,121		
Non-trainable params: 0		



Figure-3: Model Summary

4.1 Dataset Description

For the CNN training and testing purposes, we consider using International Skin Imaging Collaboration (ISIC2020) database commonly known as ISIC2020 dataset. The dataset contains 33,126 dermoscopic training images of unique benign and malignant skin lesions from over 2,000 patients. All malignant diagnoses have been confirmed via histopathology, and benign diagnoses have been confirmed using either expert agreement, longitudinal follow-up, or histopathology. In whole dataset two classes are as follows:

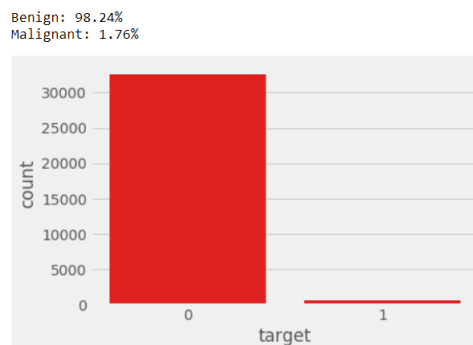


Figure-4: Benign and Malignant Classes

Few benign and malignant images are shown below:

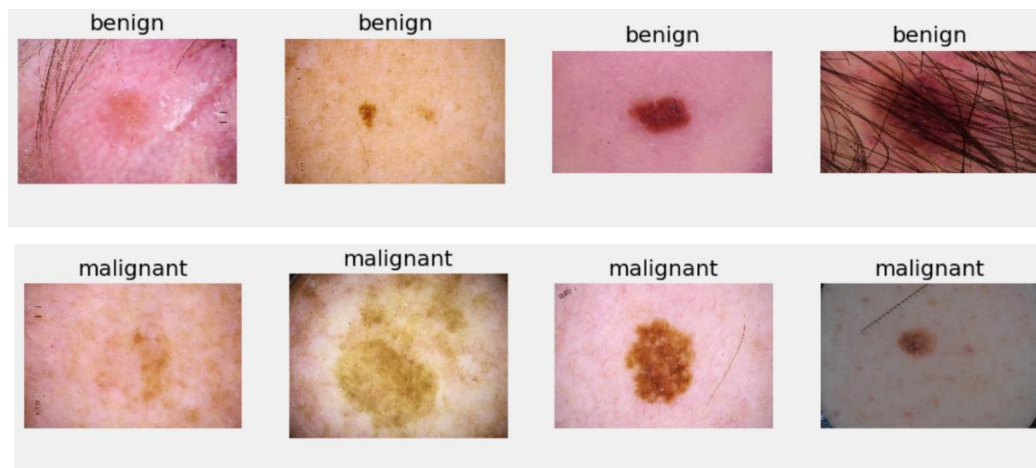


Figure-5: Benign and Malignant Images (Random)

To minimize the loss, we have used RMSProp, SGD and ADAM optimizers which are variant of gradient descents. Then we normalized the images between 0 to 1 for faster training. We used total 4549 images, and split the dataset into three datasets. We started using datasets from low to medium to high. Our three datasets are in form of (number of training images: number of testing images) 601:232, 2416:633, 3416:1133. We used these three datasets to check our models performance in order to achieve highest accuracy. For each dataset we trained our constructed CNN and kept the image size as 150x150 pixels for models best performance and introduced a validation data split of 10% and batch size 20.

In dataset 601:232:

- total training benign images: 401
- total training malignant images: 200
- total testing benign images: 132
- total testing malignant images: 100

In dataset 2416:633:

- total training benign images: 2016
- total training malignant images: 400
- total testing benign images: 500
- total testing malignant images: 133

In dataset 3416:1133:

- total training benign images: 3016
- total training malignant images: 400
- total testing benign images: 1000
- total testing malignant images: 133



During training the hyperparameters are as follows:

- steps_per_epoch=100,
- epochs=30,
- validation_steps=50

5 Experimental Results

We train our constructed CNN for three datasets and used seven evaluation metrics to evaluate our neural network. The metrics are:

- Validation Accuracy
- Validation loss
- Precision
- Specificity
- Sensitivity
- Loss
- Accuracy



Brief description of these metrics will helpful before presenting our final results.

Accuracy: Accuracy is a method for measuring a classification model's performance. It is typically expressed as a percentage. Accuracy is the count of predictions where the predicted value is equal to the true value. It is binary (true/false) for a particular sample.

Loss: A loss function, also known as a cost function, takes into account the probabilities or uncertainty of a prediction based on how much the prediction varies from the true value. This gives us a more nuanced view into how well the model is performing.


The most common loss functions are log loss and cross-entropy loss (which yield the same result when calculating error rates between 0 and 1), as well as mean squared error, and likelihood loss.

Validation accuracy: Models accuracy on validation data.

Validation loss: Models loss on validation data.

To define Precision, Sensitivity and Specificity we need the following table:

Table-1: Confusion matrix

	Positive Prediction	Negative Prediction
Positive Class	True Positive (TP) 	False Negative (FN)
Negative Class	False Positive (FP)	True Negative(TN)

Precision: Precision is a metric that quantifies the number of correct positive predictions made.

$$\text{Precision} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalsePositives}}$$

Sensitivity: Sensitivity is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made.

$$\text{Sensitivity} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalseNegatives}}$$

Specificity: Specificity of a classifier is the ratio between how much were correctly classified as negative to how much was actually negative.

$$\text{Specificity} = \frac{\text{TrueNegatives}}{\text{FalsePositives} + \text{TrueNegative}}$$

Below three tables' present CNNs performance:

Table-2: Model evaluation using (RMSProp):

	<i>Data Split</i>		
<i>Evaluation Metrics</i>	601:232	2416: 633	3416:1133
Validation Accuracy	0.7063	0.8419	0.8912
Validation Loss	0.4451	0.2178	0.1045
Precision	0.5000	0.5000	0.5000
Sensitivity	0.4000	1.0000	1.0000
Specificity	0.6667	0.3333	0.7368
Loss	0.0647	0.3156	0.2928
Accuracy	0.9815	0.8692	0.8935

Table-3: Model evaluation using Stochastic Gradient Decent (SGD):

Evaluation Metrics	Data Split		
	601:232	2416: 633	3416:1133
Validation Accuracy	0.6498	0.8022	0.8953
Validation Loss	0.3427	0.1978	0.1447
Precision	0.7500	0.4545	0.5000
Sensitivity	0.8571	0.6667	0.5000
Specificity	0.5385	0.5882	1.0000
Loss	0.5165	0.3974	0.3191
Accuracy	0.7313	0.8455	0.8938

Table-4: Model evaluation using (ADAM):

Evaluation Metrics	Data Split		
	601:232	2416: 633	3416:1133
Validation Accuracy	0.7510	0.8337	0.8993
Validation Loss	0.4790	0.1887	0.0789
Precision	0.6250	0.5000	0.3333
Sensitivity	0.2222	1.0000	0.3158
Specificity	0.8182	0.0588	1.0000
Loss	0.0726	0.3114	0.2786
Accuracy	0.9740	0.8725	0.9013

Using three set of datasets and analyzing the evaluation metrics from the above three table, we achieved maximum accuracy 98.15% from the dataset 601:232 using RMSprop optimizer and maximum validation accuracy from the dataset 89.93% using 3416:1133 dataset & ADAM optimizer. Maximum Precision 0.75 using 601:232 dataset and SGD optimizer. Similarly, we got maximum Sensitivity 1.000 using 2416:633, 3416:1133 dataset and ADAM & SGD optimizers respectively. Next we got maximum Specificity 1.00 for 3416:1133 dataset using ADAM optimizer.

To better understand the CNNs behavior and performance, we further plotted graphs considering training and validation accuracy, training and validation loss. These graphs are basically used to check models overfitting and under fitting behavior. Though in our project we have not considered this issue in depth but just utilized a dropout layer .Graphs are given below:

Dataset Split: 601:232

Using RMSProp:

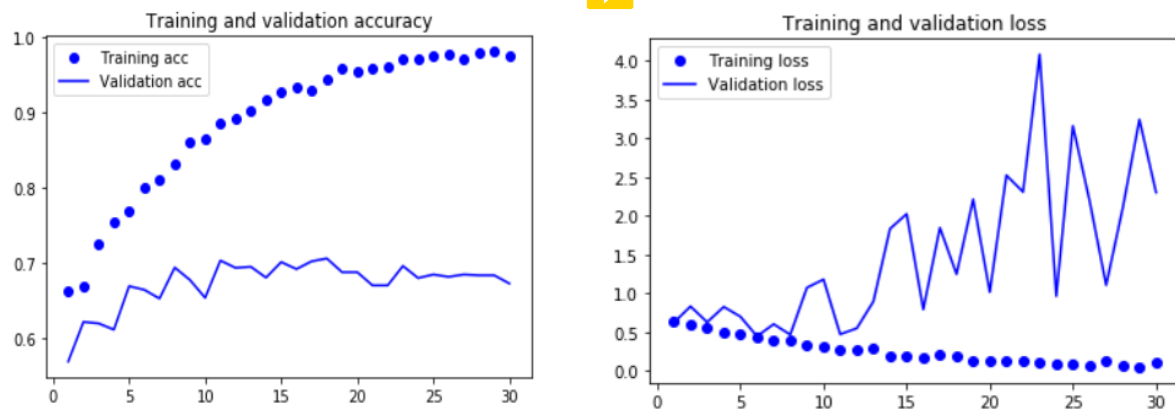


Figure-6: Training& Validation Accuracy and Training & Validation Loss using RMSProp (601:232)

Using SGD:

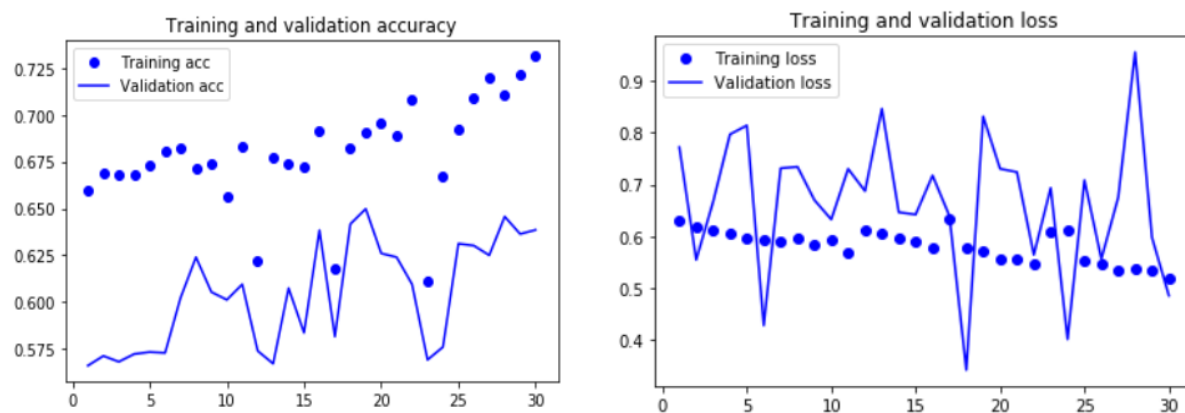


Figure-7: Training& Validation Accuracy and Training & Validation Loss using SGD (601:232)

Using ADAM:

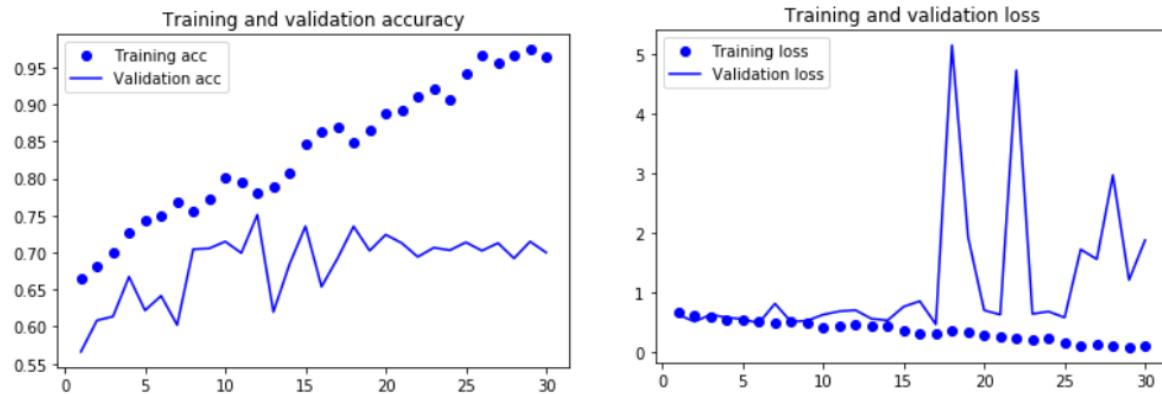


Figure-8: Training& Validation Accuracy and Training & Validation Loss using ADAM (601:232)

Dataset Split: 2416:633

Using RMSProp:

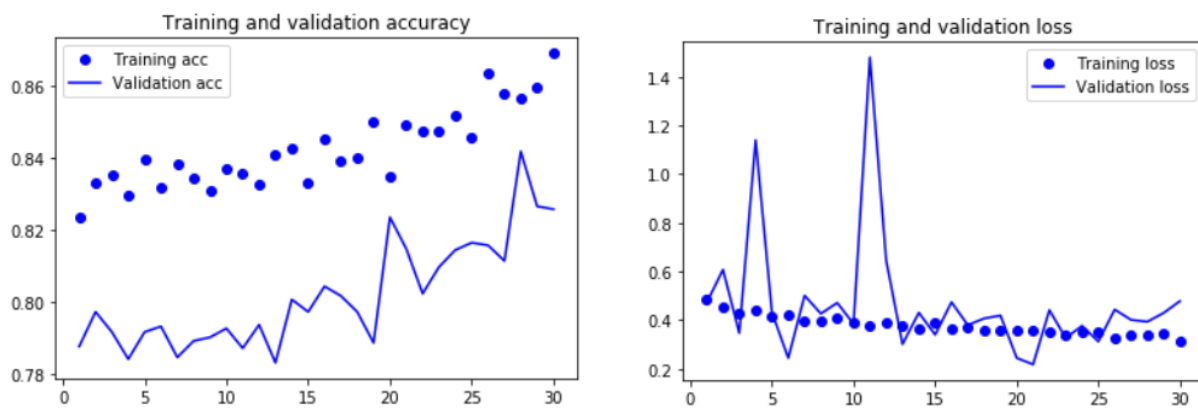


Figure-9: Training& Validation Accuracy and Training & Validation Loss using RMSProp (2416:633)

Using SGD:

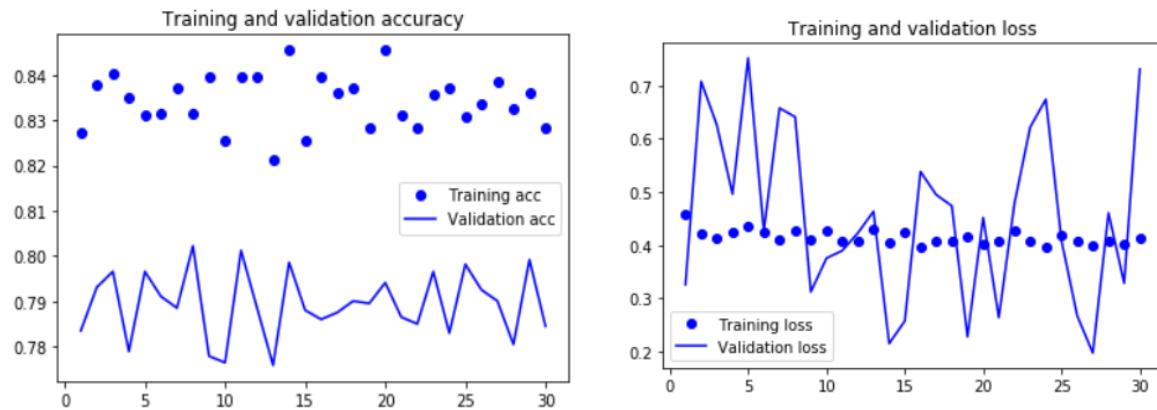


Figure-10: Training& Validation Accuracy and Training & Validation Loss using SGD (2416:633)

Using ADAM:

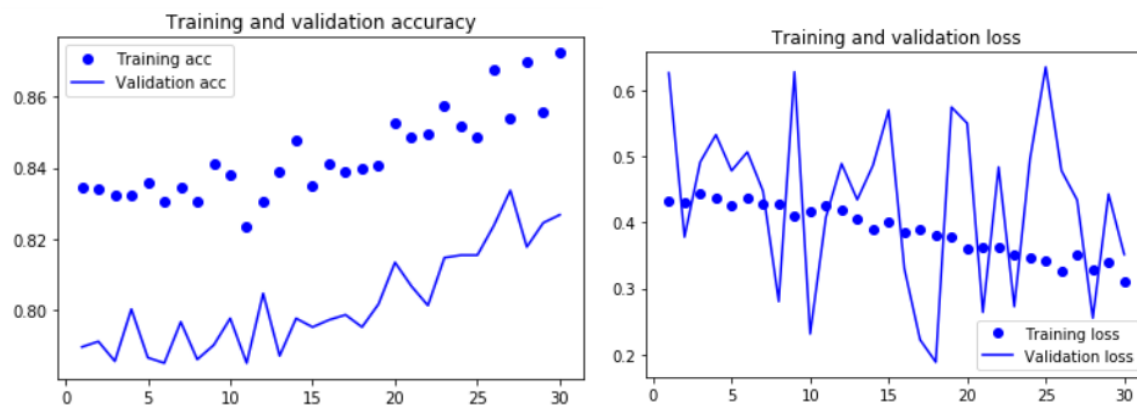


Figure-11: Training& Validation Accuracy and Training & Validation Loss using ADAM (2416:633)

Dataset Split: 3416:1133

Using RMSProp:

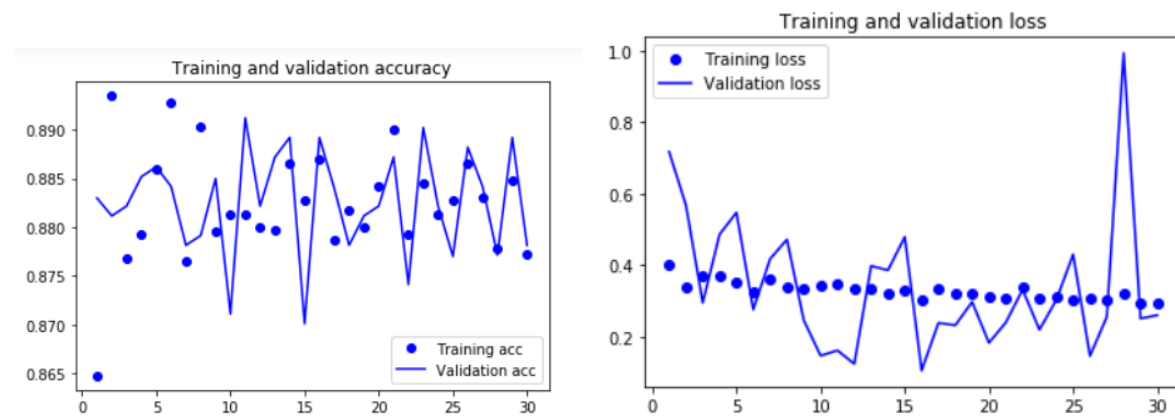


Figure-12: Training& Validation Accuracy and Training & Validation Loss using RMSProp (3416:1133)

Using SGD:

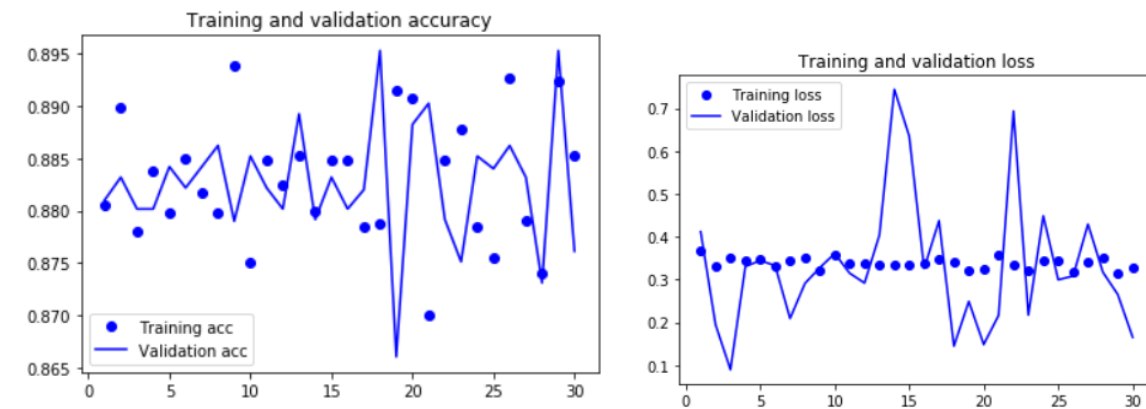


Figure-13: Training& Validation Accuracy and Training & Validation Loss using SGD (3416:1133)

Using ADAM:

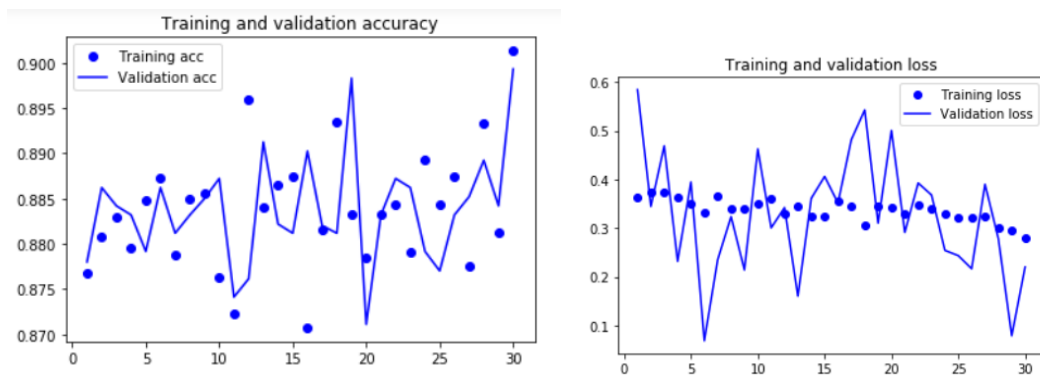


Figure-14: Training & Validation Accuracy and Training & Validation Loss using ADAM (3416:1133)

Analyzing all the graphs, it is clear that for small dataset 601:232 the training accuracy is increasing but validation accuracy is decreasing. In addition, training loss is decreasing but validation loss is increasing. Thus it's a sign of overfitting. For dataset 2416:633, the previous issue has improved since the validation accuracy is almost following the training accuracy curve and validation loss is decreasing with training loss but there is still a gap between training and validation accuracy. Thus there is still chance to improve the models performance. Thus we added more data to train. For dataset 3416:1133, it is clear that validation and training accuracy are improving together and validation accuracy is constantly improving with training accuracy. In addition, for this dataset, training and validation loss are decreasing together. Thus increasing dataset size helps reduce overfitting problem.

6 Conclusion

Therefore, considering the major issue in healthcare which is skin cancer, AI can play a vital role alongside the dermatologists to quickly diagnose the skin cancer. In this project, we just touch a small portion of this huge problem to resolve by introducing a CNN architecture that achieved accuracy 98.15%. We used popular optimizing algorithms to minimize the loss of our neural network. By using ADAM, RMSProp and SGD algorithms, we attained 89.53% validation accuracy. Considering other evaluation metrics such as Precision, Sensitivity and Specificity, our model performed better. By plotting the training and validation accuracy, training and validation loss graphs, we showed the comparisons of various algorithms performance considering three datasets. We successfully trained and validated our model by using seven evaluation metrics. In addition, our collection of a huge skin lesion images dataset

took ample amount of time. Therefore, preprocessing the dataset was a huge task to accomplish. Using a small subset of this huge dataset showed great promise regarding image classification. Therefore, it leaves a huge future improvement on this CNN model.

6.1 Future Works

- Other variants of the optimizers such as NADAM, Adadelata, AdaMax and so on could be tried in future to minimize the loss function.
- Increasing the number of training and validation images may produce different results and improvement.
- Different set of hyperparameters with different values such as learning rate, momentum, decay etc. may improve the performance significantly.
- Modifying the neural network architecture with different number of convolutional layers and max_pooling layers may produce good results as well.
- We may implement transfer learning such as train the architecture on VGG-16 or ResNet or LeNet.
- We have not done much for overfitting and under fitting issue in this project that can be tried in future works.

7 References

- [1] Albahar, M. A. (2019). Skin lesion classification using convolutional neural network with novel regularizer. *IEEE Access*, 7, 38306-38313.
- [2] Astudillo, N. M., Bolman, R., & Sirakov, N. M. (2020). Classification with Stochastic Learning Methods and Convolutional Neural Networks. *SN Computer Science*, 1(3), 1-9.
- [3] Banerjee, S., Singh, S. K., Chakraborty, A., Das, A., & Bag, R. (2020). Melanoma Diagnosis Using Deep Learning and Fuzzy Logic. *Diagnostics*, 10(8), 577.
- [4] Bora, K., Chowdhury, M., Mahanta, L. B., Kundu, M. K., & Das, A. K. (2016, December). Pap smear image classification using convolutional neural network. In *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing* (pp. 1-8).
- [5] Brinker, T. J., Hekler, A., Utikal, J. S., Grabe, N., Schadendorf, D., Klode, J., ... & von Kalle, C. (2018). Skin cancer classification using convolutional neural networks: systematic review. *Journal of medical Internet research*, 20(10), e11936.

- [6] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639), 115-118.
- [7] Gao, Q., Lim, S., & Jia, X. (2018). Hyperspectral image classification using convolutional neural networks and multiple feature learning. *Remote Sensing*, 10(2), 299.
- [8] Harangi, B. (2018). Skin lesion classification with ensembles of deep convolutional neural networks. *Journal of biomedical informatics*, 86, 25-32.
- [9] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
- [10] Mahbod, A., Schaefer, G., Wang, C., Ecker, R., & Ellinge, I. (2019, May). Skin lesion classification using hybrid deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1229-1233). IEEE.
- [11] Mateen, M., Wen, J., Song, S., & Huang, Z. (2019). Fundus image classification using VGG-19 architecture with PCA and SVD. *Symmetry*, 11(1), 1.
- [12] Oraibi, Z. A., Yousif, H., Hafiane, A., Seetharaman, G., & Palaniappan, K. (2018, October). Learning local and deep features for efficient cell image classification using random forests. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (pp. 2446-2450). IEEE.
- [13] Pham, T. C., Luong, C. M., Visani, M., & Hoang, V. D. (2018, March). Deep CNN and data augmentation for skin lesion classification. In *Asian Conference on Intelligent Information and Database Systems* (pp. 573-582). Springer, Cham.
- [14] Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2016, July). Breast cancer histopathological image classification using convolutional neural networks. In *2016 international joint conference on neural networks (IJCNN)* (pp. 2560-2567). IEEE.
- [15] Winkelmann, R. R., Farberg, A. S., Tucker, N., White, R., & Rigel, D. S. (2016). Enhancement of International Dermatologists' Pigmented Skin Lesion Biopsy Decisions Following Dermoscopy with Subsequent Integration of Multispectral Digital Skin Lesion Analysis. *The Journal of clinical and aesthetic dermatology*, 9(7), 53.

- [16] Zou, F., Shen, L., Jie, Z., Zhang, W., & Liu, W. (2019). A sufficient condition for convergences of adam and rmsprop. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 11127-11135).

APPENDIX

Our constructed neural network is implemented using Tensorflow, Keras and Python. We use Jupyter notebook to write our code. Our code is like follows:

#Libraries

```
from keras import layers
from keras import models
```

```
from keras import optimizers
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
from keras.preprocessing.image import ImageDataGenerator
```

#Dataset

```
dir='/Dataset/'
```

#Model

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

#Image Preprocessing and Scaling

#scaling all images between 0-1

```
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

#splitting images into training and validation set

```
train_generator = train_datagen.flow_from_directory(train_dir,target_size=(150, 150),batch_size=20,class_mode='binary')
validation_generator = test_datagen.flow_from_directory(test_dir,target_size=(150, 150),batch_size=20,class_mode='binary')
```

#Model Compiling

```
model.compile(loss='binary_crossentropy', optimizer='RMSProp',metrics=['acc'])
model.compile(loss='binary_crossentropy', optimizer=ADAM,metrics=['acc'])
model.compile(loss='binary_crossentropy', optimizer=SGD,metrics=['acc'])
```

#Training

```
history=model.fit(train_generator,steps_per_epoch=100,epochs=30,validation_data=validation_generator,validation_steps=50)
```

#Precison

```
tf.keras.metrics.Precision()
```

#Sensitivity and Specificity

```
from keras import backend as K
```

```
def sensitivity(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    return true_positives / (possible_positives + K.epsilon())
```

```
def specificity(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1-y_true) * (1-y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1-y_true, 0, 1)))
    return true_negatives / (possible_negatives + K.epsilon())
```

#Evaluation

```
model.evaluate()
```

#Prediction

```
model.predict()
```