Application of ResNet-50, Convolutional Neural Network and Multi-class Logistic Regression in Skin Cancer Classification and Analysis of Back Propagation

Ana Peçini[1], **Bengi Dönmez**[2], **Cazibe Kavalcı**[3], **Emre Batuhan Baloğlu**[4], **and Ahmet Talha Şen**[5]

[1]21600103, Computer Science
[2]21602237, Mathematics
[3]21501185, Mathematics
[4]21502554, Mathematics
[5]21502663, Mathematics

ABSTRACT

Skin cancer classification has been an important study in recent research and machine learning industry. In this project we aim at using two neural network architectures to classify skin cancer: Residual Network(ResNet) and a traditional Convolutional Neural Network (CNN). We experimentally observed and compared the performance between these two models. Furthermore, we understood the mathematics behind the architecture of ResNet, back-propagation algorithm and Adam optimization. The experimental results show that traditional CNN has a slightly better performance than ResNet, but due to the restrictions of the computational power, we cannot train ResNet with a large amount of epochs, because it is a deep network.

1 Introduction

Skin cancer is the most common human disease where averagely 40% to 50% of fair-skinned people will develop at least one type until the age of 65[1]. More than 40% of melanomas are found by patients themselves by visually examining their skin and spotting suspicious lesions[2]. The disease can be cured if it's found and treated early. Because of these reasons, automated classification of skin lesions can remarkably help the doctors, dermatologists and patients themselves diagnose skin cancer in the early stages and increase the chances of successful treatments.
There is research in recent years that studies and investigates the development of skin cancer classification algorithms[3–5]. One of the most common methods to perform automated classification is using neural networks. Practice has shown that shallow neural networks produce overfitting. However, increasing the depth of the network avoid overfitting but increases testing and training errors if weights are not properly initialized. That happens because of the vanishing gradient problem. As the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient extremely small. Consequently, as the network becomes deeper, the accuracy gets saturated and the error remains higher than shallower networks. Residual Network(ResNet) is a deep neural network which gives a solution to the problem of vanishing gradients by introducing the concept of skip connection or shortcuts. In the following sections, we will analyze the back propagation process of the traditional CNN model, illustrate the skip connection and architecture of ResNet, analyze the back propagation formula of this network and compare it with traditional CNN and explain how this model helps with the vanishing gradient problem. Lastly, we will present and discuss the results and demonstrate the methods we used in our experiments.

2 Back Propagation on CNN

With back propagation, the gradient descent algorithm adjusts the weights of the network through a backward pass. It updates each weight in the direction of the gradient of this weight and using chain rule it finds the gradient for the weights of shallower layers. In this project we are going to analyze the back propagation process with ReLu as the activation function, adam optimizer and cross - entropy loss function.

2.1 Rectified Linear Unit (ReLU)

Rectifier function is an activation function defined as:

$$ReLU(x) = max(x,0)$$

which gives

$$ReLU'(x) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x > 0, \end{cases}$$

2.2 Cross Entropy Function

In this algorithm we use cross entropy function as a cost function. Which is defined as:

$$CE(y_i, p_i) = \sum_{i=1}^{C} y_i log(p_i)$$

where C defines number of classes,p_i is the classifier prediction and y_i is the ground-truth labels.

2.3 Calculation of the Gradient

We defined the variables as the following:

- x^ℓ the input vector of the layer ℓ
- x_j^ℓ the input feature of the node j in layer ℓ
- a^ℓ :the output vector of the layer ℓ
- a_j^ℓ :the output feature of the node j in layer ℓ
- w^ℓ the weight matrix for neurons between layer $\ell - 1$ and ℓ
- w_{jk}^ℓ :the weight feature applied to the k^{th} element of $a^{\ell-1}$ to feed into the node j in layer ℓ
- m^ℓ :the number of nodes in the layer ℓ

We calculate back propagation algorithm, first we need:

$$\frac{\partial CE}{\partial x_j^\ell} = \frac{\partial CE}{\partial a_j^\ell} \frac{\partial a_j^\ell}{\partial x_j^\ell} \tag{1}$$

We know that $a_j^\ell = ReLu(x_j^\ell)$. Hence partial derivative is $\omega = ReLu'(x_j^\ell)$ which is 1 if $x_j^\ell > 0$, otherwise it gives 0. So equation becomes:

$$\frac{\partial CE}{\partial x_j^\ell} = \frac{\partial CE}{\partial a_j^\ell} \omega = \delta_j^\ell \tag{2}$$

Now we need:

$$\frac{\partial CE}{\partial a_j^\ell} = \sum_{k=1}^{m^\ell} \frac{\partial CE}{\partial x_k^{\ell+1}} \frac{\partial x_k^{\ell+1}}{\partial a_j^\ell} \tag{3}$$

We know that $x_k^{\ell+1} = (a_j^\ell)^T w_k^\ell$. This gives $\frac{\partial x_k^{\ell+1}}{\partial a_j^\ell} = w_{kj}^\ell$. So equation becomes

$$\frac{\partial CE}{\partial a_j^\ell} = \sum_{k=1}^{m^\ell} \frac{\partial CE}{\partial x_k^{\ell+1}} \frac{\partial x_k^{\ell+1}}{\partial a_j^\ell} = \sum_{k=1}^{m^\ell} \frac{\partial CE}{\partial x_k^{\ell+1}} w_{kj}^\ell = \sum_{k=1}^{m^\ell} \delta_k^{\ell+1} w_{kj}^\ell = [(w^\ell)^T \delta^{\ell+1}]_j \tag{4}$$

Finally we get,

$$\frac{\partial CE}{\partial x_j^\ell} = \delta_j^\ell = [(w^\ell)^T \delta^{\ell+1}]_j \cdot \omega \tag{5}$$

We also need:

$$\frac{\partial CE}{\partial w_{jk}^\ell} = \frac{\partial CE}{\partial x_j^\ell} \frac{\partial x_j^\ell}{\partial w_{jk}^\ell} \tag{6}$$

We know that $x_j^\ell = w^\ell a^{\ell-1}$ which gives $\frac{\partial x_j^\ell}{\partial w_{jk}^\ell} = a_k^{\ell-1}$ So:

$$\frac{\partial CE}{\partial w_{jk}^\ell} = \frac{\partial CE}{\partial x_j^\ell} \frac{\partial x_j^\ell}{\partial w_{jk}^\ell} = \delta^\ell a_k^{\ell-1} \tag{7}$$

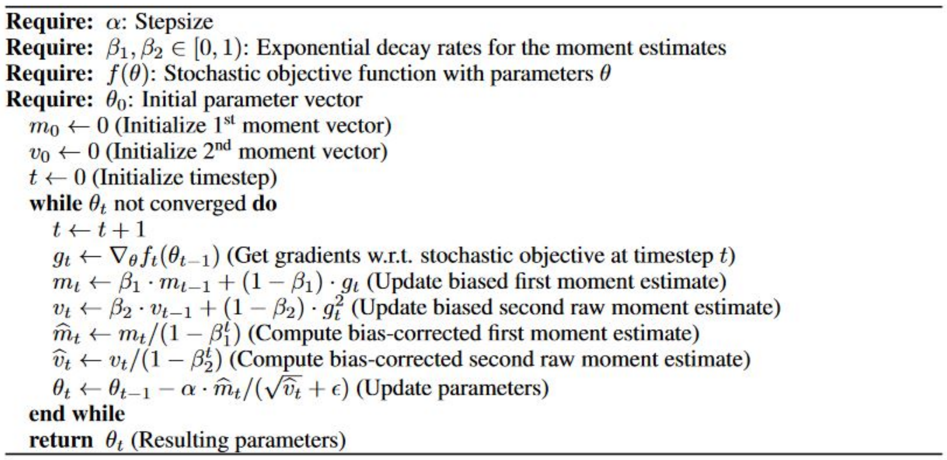
Which gives the gradient:

$$\nabla CE_i(w^\ell) = \delta^\ell (a^{\ell-1})^T$$

According to this gradient Adam Optimization Algorithm does the updating the weights.

3 Adam Optimization Algorithm

Adam Optimization Algorithm (Adaptive Moment Estimation Algorithm) provides a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement.[6] The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. This method is designed to combine the advantages of two recently popular methods: AdaGrad, which works well with sparse gradients, and RMSProp, which works well in on-line and non-stationary settings. Some of Adam's advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradient, its stepsizes are approximately bounded by the stepsize hyperparameter,it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing.



```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Figure 1. Adam Optimization Algorithm

Therefore, this method has advantages since it is easy to implement as it can be seen from pseudo-code, computationally efficient with little memory space requirement, suitable for non-stationary objectives, works on problems with noisy gradients

properly and so on.
Here is the update rule for Adam Optimization:

$$\theta_{t+1} = \theta_t - \frac{n.\tilde{m}_t}{\sqrt{\tilde{v}_t} + \varepsilon}$$

where

$$\begin{aligned}
\tilde{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\tilde{v}_t &= \frac{v_t}{1-\beta_2^t} \\
m_t &= (1-\beta_1)g_t + \beta_1 m_{t-1} \\
v_t &= (1-\beta_2)g_t^2 + \beta_2 v_{t-1}
\end{aligned}$$

We generally take $\varepsilon = 10^{-8}, n = 0.001$ as gradient learning rate, $g = 0.99$ and $\beta = 0.9$ as momentum terms.

4 Back Propagation on ResNet and the Vanishing Gradient Problem

Figure 1 illustrates the concept of skip connection. In the diagram to the left there are layers with neurons stacked after one another. Each layer propagates its activation values to the next layer. On the other hand, in the diagram to the right, layers are stacked in the same way but there is an additional connection from a previous layer to a next one by jumping over one or more layers in between.

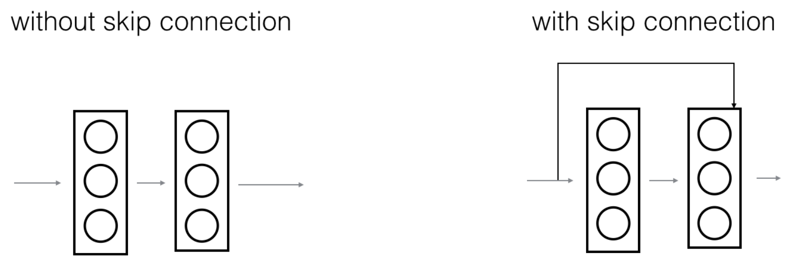


Figure 2. Skip connection

ResNet model consists of such convolutional blocks with skip connections which then are loaded to an activation function (e.g. ReLU). In this case, the skip connections perform identity mapping and their outputs are added to the outputs of the stacked layers. These are called residual blocks and are shown in figure 2[7].

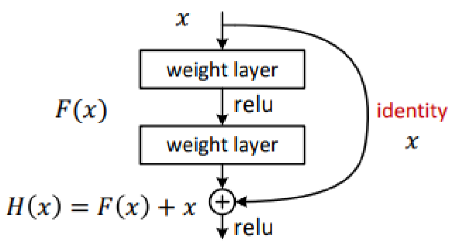


Figure 3. Residual block

To explain the process of back propagation through residual blocks, we will use the following notation, same as in figure 2:

- x^ℓ the input feature to the layer ℓ

- F the residual function, for example a stack of convolutional layers.
- W^ℓ the weight matrix for neurons between layer $\ell - 1$ and ℓ
- f the activation function, for example, ReLU.

Then the output of the ℓ residual block through forward propagation is given by

$$x^{\ell+1} = f(x^\ell + F(x^\ell, W^\ell)) \tag{8}$$

For simplicity we assume f is an identity mapping. Then equation (1) is transformed to

$$x^{\ell+1} = x^\ell + F(x^\ell, W^\ell) \tag{9}$$

and by substituting recursively we have

$$x^{\ell+2} = x^{\ell+1} + F(x^{\ell+1}, W^{\ell+1}) = x^\ell + F(x^\ell, W^\ell) + F(x^{\ell+1}, W^{\ell+1}) \tag{10}$$

Thu, the equation (2) can be generalized by the following equation

$$x^L = x^\ell + \sum_{i=\ell}^{L-1} F(x^i, W^i) \tag{11}$$

for any deeper layer L and shallower ℓ as in figure 3.

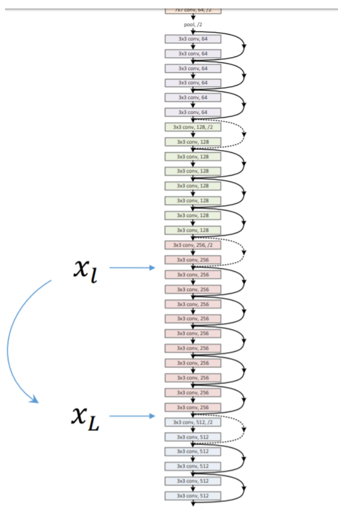


Figure 4. Forward Propagation

In the training process with back propagation, the weights are updating through a backward pass. Let E denote the loss function. $\frac{\partial E}{\partial x^L}$ propagates information backwards to adjust the weights of x^ℓ. The back propagation algorithm is given by

$$\frac{\partial E}{\partial x^\ell} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial x^\ell} = \frac{\partial E}{\partial x^L}(1 + \frac{\partial}{\partial x^\ell}\sum_{i=\ell}^{L-1} F(x^i, W^i)) \tag{12}$$

Equation (5) shows that there are two additive terms being propagated to shallower layers. The term $\frac{\partial E}{\partial x^L}$ is propagated directly and the term $\frac{\partial E}{\partial x^L}(\frac{\partial}{\partial x^\ell}\sum_{i=\ell}^{L-1} F(x^i, W^i))$ propagated through weight layers. Hence, these identity shortcuts operate as gradient gateways, where the gradient flows straightforward.
In contrast, the equivalent forward propagation formula in traditional CNN is

$$x^{\ell+1} = W^\ell x^\ell \tag{13}$$

which gives the general equation

$$x^L = \prod_{i=\ell}^{L-1} W^i x^\ell \tag{14}$$

The gradient $\frac{\partial E}{\partial x^\ell}$ in this case is given by

$$\frac{\partial E}{\partial x^\ell} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial x^\ell} = \frac{\partial E}{\partial x^L}\prod_{i=\ell}^{L-1} W^i \tag{15}$$

In traditional CNN case, if the weights are relatively small, the multiplicative terms may make the gradient cancel out(vanish) and become 0.
However, in the ResNet case, for the gradient $\frac{\partial E}{\partial x^\ell}$ to be 0, then $\frac{\partial}{\partial x^\ell}\sum_{i=\ell}^{L-1} F(x^i, W^i) = -1$ for all x^ℓ. Since this is very unlikely, it follows that the network does not allow the vanishing gradient problem to occur, even when the weights are relatively small.

Methods

4.1 Data Preprocessing

Several processing methods are used on the data images. Initially, we process the images as $(64 \times 64 \times 3)$, $(32 \times 32 \times 3)$ matrices, each varying in resolution where it represents each image as a square frame with red-green-blue color values of each pixel. Our dataset consists of 10015 dermatoscopic images. Cases include a representative collection of all important diagnostic categories in the realm of pigmented lesions: Actinic keratoses and intraepithelial carcinoma / Bowen's disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratoses, bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage, vasc).
On the other hand, we see an imbalanced distribution among labels for our data. It can be observed that, among 10015 images, almost half of them consist of images which has diagnosis melanocytic nevi (nv), and this imbalance reduces the performance of our models, which we will discuss in the results section.

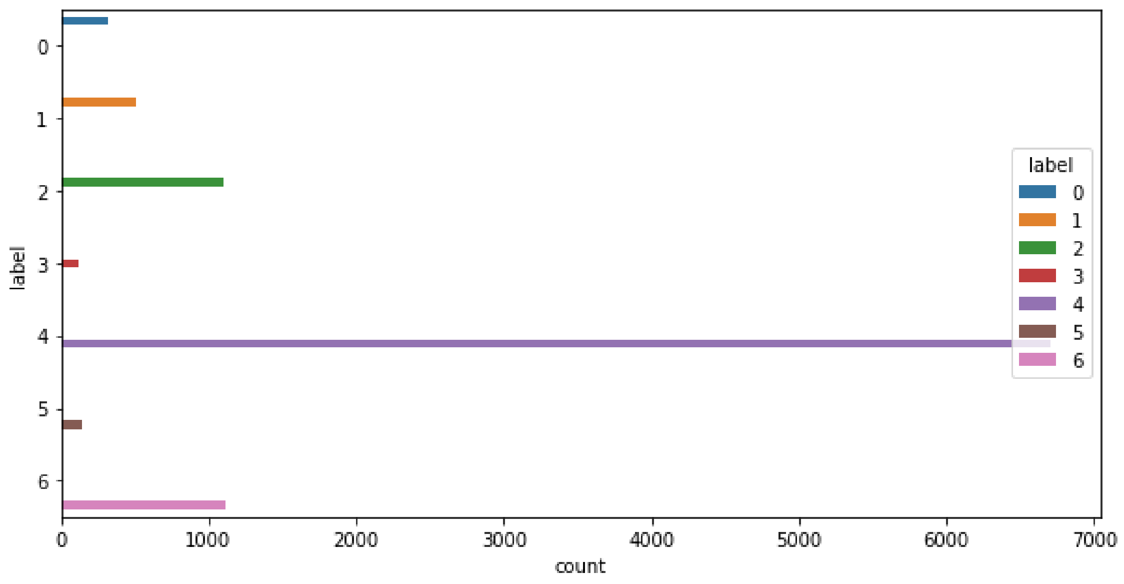


Figure 5. Distribution of classes among the data

As it is seen above, our data is heavily imbalanced. Therefore, taking accuracy into consideration is nothing but illusion. Any model can easily orient itself to predict 4th category which would result in a very high accuracy. However, the model would be very unsuccessful at differentiating classes. One possible solution is over-sampling the minority classes. Through our experiments, over-sampling failed to improve our models because some classes, namely 3 and 5, contain few samples. Generating new data points from these have no significant outcome. Another possible solution is under-sampling the classes with too many data points. In our case reducing data points of 4th label resulted in a slight enhancement in the model. Combining both under-sampling and over-sampling slightly increased the quality of prediction ability of our designs. In all of our models, we split the data into training and test sets by 80% and 20% respectively. The reason for this is to measure the models' performance on both of them and observe bias and variance of our models. Observing the training set and test set performance of the models help us reduce overfitting to the data by using regularization.

4.2 ResNet-50

The first model architecture we used for this classification task is ResNet-50, as it is described by He et. al.[7]

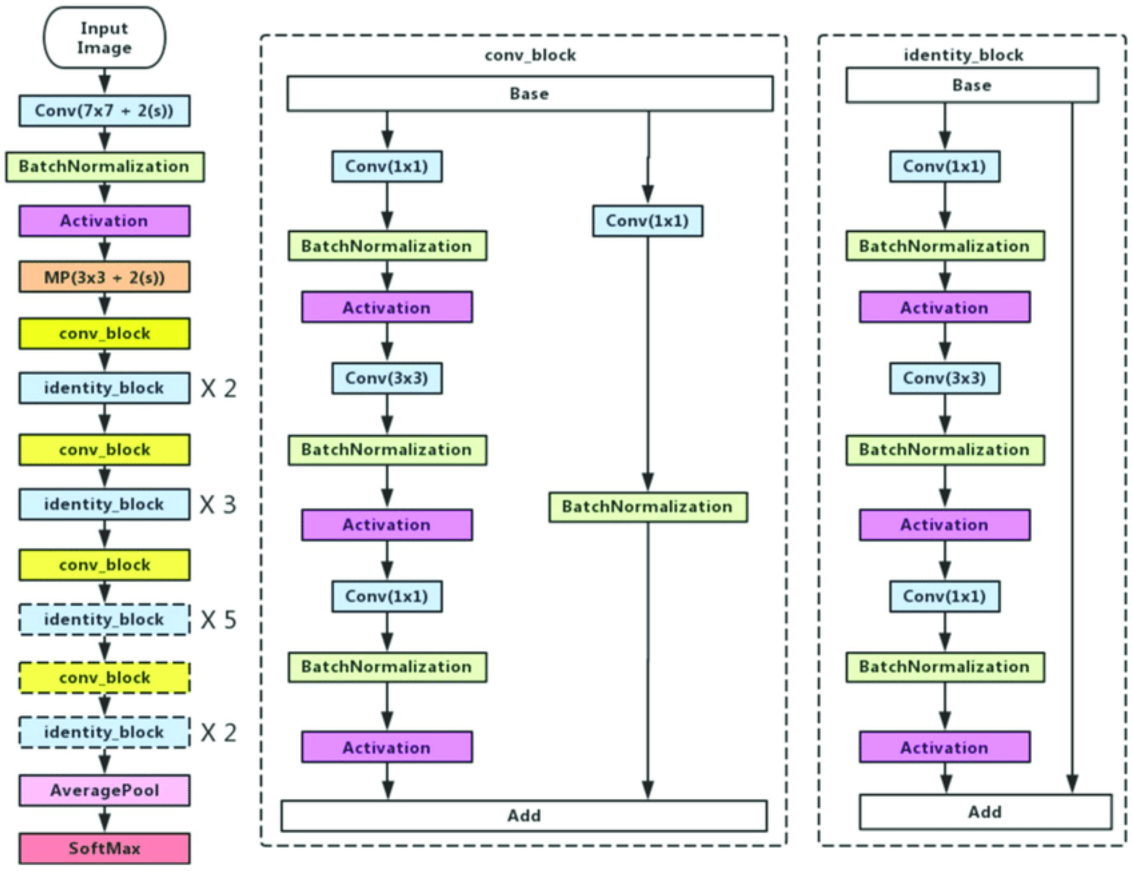


Figure 6. Architecture of the ResNet-50

As we discussed in the previous sections, ResNet-50 is a deep neural network which benefits from the skip connections between hidden layers. In order to get a better convergence rate, we also used Adam Optimization algorithm with learning rate 0.01.

4.3 Convolutional Neural Network

We used a Convolutional Neural Network model together with two fully connected layers and a softmax layer when training the images. We build three convolution layers. In the first layer, we have 10 3×3 convolution filters, with same padding. Following this, we apply Max-Pooling with a 2×2 pooling size. In the second convolution layer, we use 10 5×5 convolution filters. Then we apply Max-Pooling with 3×3 pooling size. In the last convolution layer, we use 10 7×7 convolution filters. Then we apply Max-Pooling with 4×4 pooling size. Then we obtain a three dimensional output which we flatten and feed into a fully connected neural network with layers having 120, 64 and 7 units. We use ReLU activation in each of these units including the convolution layers except the output layer, in which we use softmax activation.

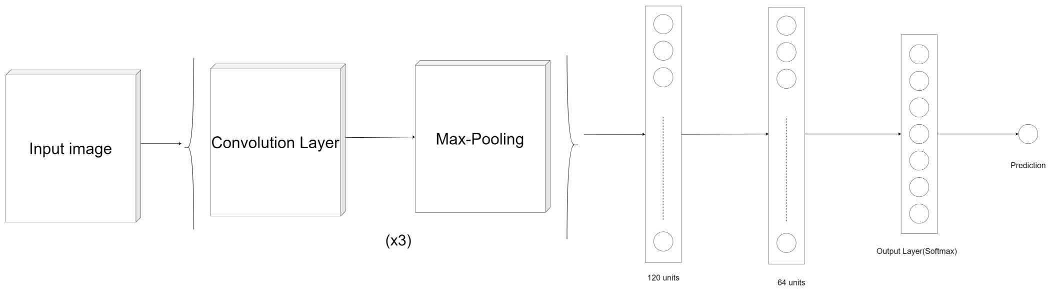


Figure 7. Architecture of our CNN model

4.4 Multi-Class Logistic Regression

We also used a simple neural network architecture to simulate Multi-Class Logistic Regression. Multi-Class Logistic regression is a widely used Machine Learning model for classification purposes. This approach uses the logistic function to estimate the probability of the desired category where the logistic function of probabilistic distribution is:

$$S(x) = \frac{e^x}{e^x + 1} \tag{16}$$

We feed the model with the flattened vectors we obtained from three dimensinoal matrices generated in data preprocessing stage. In order to improve performance, we applied batch normalization in the model. We used a softmax activation layer so as to generate the predictions of the model, which provides a prediction for every class.

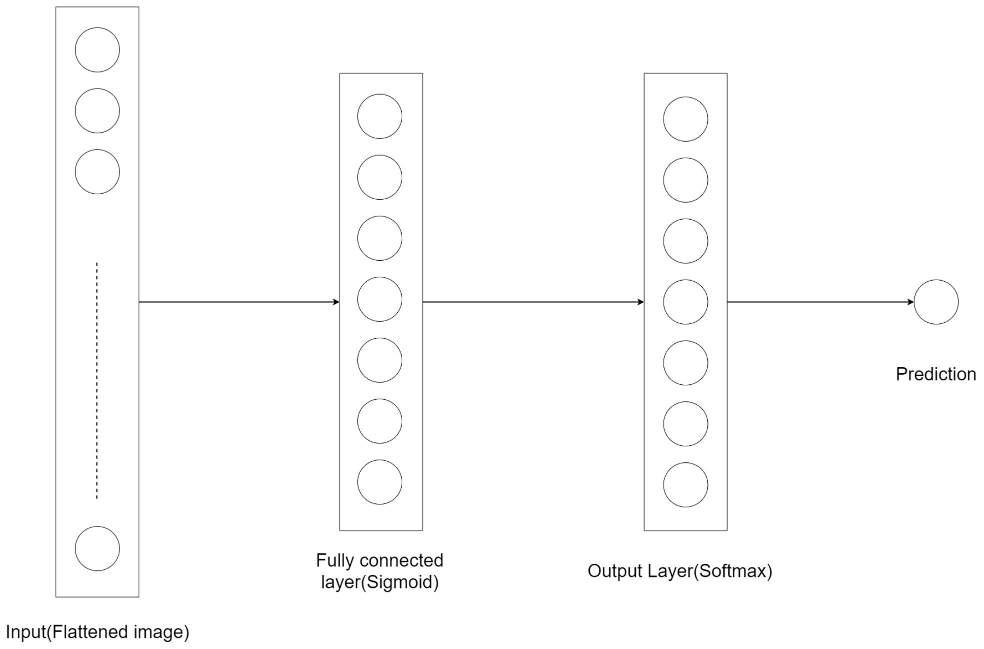


Figure 8. Diagram of our Multi-Class Logistic Regression Model

Discussion

As we discussed earlier, accuracy is not a good measure on imbalanced data set. So we will use recall, precision and F1 score to measure viability of our models. Let's first demonstrate the confusion matrix for 2 classes which are negative and positive.

	Negative	Positive
Negative	True Negative	False Positive
Positive	False Negative	True Negative

$$Precision = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{17}$$

$$Recall = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{18}$$

Precision shows how successfully our model is predicting whereas Recall demonstrates the portion of successful predictions over total instances of that class. $F1$ score combines those in order to lessen the effect of imbalance.

$$F1 = 2 \text{ x } \frac{\text{Precision x Recall}}{\text{Precision + Recall}} \tag{19}$$

Results

Image Size $(64 \times 64 \times 3)$	Logistic Regression	CNN	ResNet-50
Accuracy	0.680	0.757	0.713
F1 Score	0.629	0.688	0.644
Precision	0.311	0.351	0.336

Table 1. Scores of the models with image size $(64 \times 64 \times 3)$.

Image Size $(32 \times 32 \times 3)$	Logistic Regression	CNN	ResNet-50
Accuracy	0.649	0.708	0.698
F1 Score	0.651	0.687	0.626
Precision	0.250	0.387	0.287

Table 2. Scores of the models with image size $(32 \times 32 \times 3)$.

It turns out that our CNN model performs better than the ResNet-50 model in both of the images sizes. The reason for this may be a result of epoch sizes of these models. We trained the CNN model with 200 epochs, and ResNet-50 model with 20 epochs. Since training even a single epoch takes a lot of time for ResNet-50 because it is a deep network with 50 layers, hence we needed to keep the number of epochs small in this model.

For better results, we should train the data set on deeper and more complex machine learning models. Moreover, these results base on few training samples, for more accurate results, we should do cross validation for each model. Unfortunately, the lack of computing power does not allow us to conduct such experiments. Further, in order to compensate the imbalance effect of minority classes, one should gather more images from patients so that our models will have better understanding of those classes.

References

1. Skin cancer photos: What skin cancer precancerous lesions look like.
2. Tsao, H. *et al.* Early detection of melanoma: Reviewing the abcdes. *J. Am. Acad. Dermatol.* **72**, 717–723, DOI: 10.1016/j.jaad.2015.01.025 (2015).
3. de Carvalho, T. M., Noels, E., Wakkee, M., Udrea, A. & Nijsten, T. Development of smartphone apps for skin cancer risk assessment: Progress and promise. *JMIR Dermatol* **2**, e13376, DOI: 10.2196/13376 (2019).
4. Codella, N. C. F. *et al.* Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic) (2017). 1710.05006.
5. Esteva, A. *et al.* Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **542**, 115–118, DOI: 10.1038/nature21056 (2017).
6. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. (2014). 1412.6980.
7. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *ICLRV & COCO* (2015). 1512.03385.

Author contributions statement

E. B., A. P. and T. Ş. worked on training machine learning models. A. P. analysed the ResNet architecture, the back propagation on this network and the vanishing gradient problem. E. B. and T. Ş. clarified the methods, results and discussion. C. K. and B. D. analysed Back Propagation on CNN and Adam Optimization Algorithm. All authors reviewed the manuscript.