Exploring ResNet18 Performance in Dermoscopic Image Classification

Remziye Sude Güngör Neslişah Cebeci Özgür Barış Diler

June 2, 2024

1. Introduction

In this project, we aim to address the challenge of disease classification using the HAM10000 dataset, as defined by the ISIC 2018 Challenge, specifically part 3 of the challenge. We utilized the ResNet18 convolutional neural network (CNN) architecture to perform classification tasks on The HAM10000 dataset. We experiment with different hyperparameters such as learning rate and batch size to optimize model performance. Additionally, we have preprocessed our dataset with traditional methods to experiment whether that would train a better classifier. Our goal is to evaluate and compare the performances of these different ResNet18 architectures in terms of its accuracy in this classification task.

2. Dataset Preparation

2.1. Dataset

The HAM10000 dataset is composed of 10,015 dermatoscopic images categorized into seven classes: melanocytic nevi (NV), melanoma (MEL), benign keratosis-like lesions (BKL), basal cell carcinoma (BCC), actinic keratoses (AKIEC), vascular lesions (VASC), and dermatofibroma (DF). To prepare the dataset, we addressed class imbalance problem by augmenting classes with fewer than 500 images up to 500 images and reducing any classes with more than 1000 images to 1000 images. We choose not to make each class the same size because the imbalance was drastic, between some classes there were 1:60 ratios, and making each class of equal size would make us lose too much information. We also had pre-processing in two steps: we cleaned the images by removing artifacts such as hairs and cropped images to the regions of interest to improve data quality.

2.2. Class Imbalance Problem

To oversample underrepresented classes, we used random flips of the image to preserve its size and to undersample overrepresented classes we randomly choose some images. This step

was crucial to prevent the model from becoming biased towards classes with more images. To compare, in the original dataset there were 115 images of class DF whereas there were 6705 images of class NV, but we had 500 images of class DF and 1000 images of class NV, which respects their distribution to some extent and addresses the class imbalance problem.

2.3. Preprocessing - Removing Hairs

In the HAM10000 dataset, due to the nature of the problem, there were noisy artifacts such as hairs that could've affected the model training and performance, therefore we felt the need to address this issue. To remove hairs in images we first locate them and then remove them by changing each hair pixel value to the average of nearby pixels that are not in hairs. Locating them is done in three steps, in first step we filter the image in a way that highlights the hair pixels, in second step we threshold the filter responses, in third step we drop the pixels that are wrongly chosen to be hairs. In third step, we find connected components of the thresholded segmentation resulted in the second step and disregard more circular shaped ones because we want to find more linear shapes, hairs. To measure circularity on connected components we first tried the formula below which was given in the course slides.

$$\text{Circularity} = 4\pi \frac{\text{area}}{\text{perimeter}^2} \tag{2.1}$$

That calculation was problematic because the intrusions on the connected components was common in thresholded responses which effects (2.1) a lot by increasing perimeter. Therefore we did some research and found a new measure based on image moment calculations (Žunić et al., 2010).

$$\text{Circularity} = \frac{(\mu_{0,0}(S))^2}{2\pi(\mu_{2,0}(S) + \mu_{0,2}(S))} \tag{2.2}$$

Where μ's represent central moments of shape S. This calculation provided better results as can be seen from below example, Figure 2.2 and 2.3 are the results of the third step explained above, these segmented regions will be erased from the original image. It can be seen that boundary of lesion is not considered to be circular with measure (2.1), but we see that it was considered circular with measure (2.2).

2.4. Preprocessing - Focusing on Region of Interest

Again due to the nature of problem, some skin lesions are small whereas some are large and some are on such parts of the body that in the dermoscopy images there are shades in corners or and backgrounds which are of no importance to the classification of skin lesions. Hence we process our dataset further to crop images around the skin lesions. We achieve this by, first doing a k-means clustering with k = 2 to find the approximate skin lesion region which are darker regions in images; secondly, keeping in mind that there are background pixels and shades which should be disregarded, we locate the centroid of the segmented skin lesion, then crop the image around the centroid with a suitable size that makes skin lesion area to

Figure 2.1: Original Image

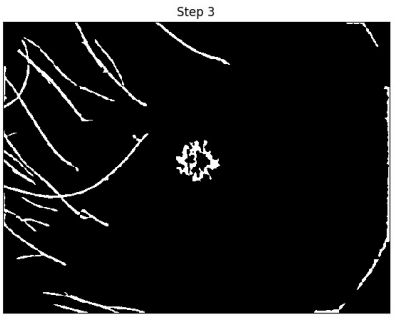

Figure 2.2: Using (2.1)

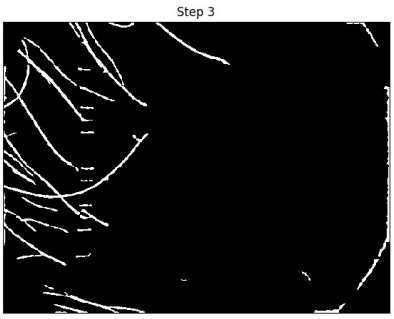

Figure 2.3: Using (2.2)

total image area larger than some threshold value. We resize the cropped image to original size by cubic interpolation. Effects of these preprocessing steps can be seen in the figures below.

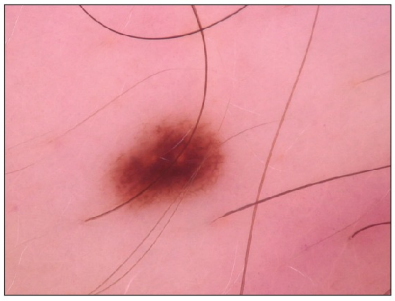

Figure 2.4: Original Image

Figure 2.5: Removing Hairs

Figure 2.6: Lesion Focus

3. Model Architecture

3.1. Model

We chose the ResNet18 neural network due to its effectiveness in image classification tasks and it suitability in limited computational resources. The network architecture consists of 18 layers, including convolutional layers, pooling layers, and fully connected layers and residual blocks in its core. We have modified its final fully connected layer to output seven classes, corresponding to the categories in the HAM10000 dataset. We use torchvision library implementation of ResNet18 and use pre-trained version with default weights at the start and then finetune it gradually.

3.2. Criterion, Optimizer and Scheduler

We train the model with a specific setup and finetune the parameters in training. The criterion we use is CrossEntropyLoss function, our optimizer is Stochastic Gradient Descent

with learning rate and momentum parameter to be chosen, our learning rate scheduler is reducing the learning rate when we do not see improvements in validation metrics. The metrics that we consider throughout the training is the criterion values and accuracy.

4. Model Training and Hyperparameter Tuning

4.1. Model Training Experiments

We trained multiple models, experimenting with various batch sizes, learning rates and momentum factors. Throughout the training process, we closely monitored the loss and accuracy on both the training and validation sets and we stopped training if validation accuracy does not improve for several epochs. In deciding which parameters are better for effective learning experience, we have used the validation set performances. After we found the parameter values for an effective learning, we also train our model on the preprocessed dataset to see whether they improve our results.

4.2. Batch Size Decision

We experimented with different batch sizes while keeping learning rate and momentum fixed, and found that a batch size of 30 provided a good balance between computational efficiency and model performance. The values of accuracy pertaining to different batch sizes can be seen from table below.

Batch Size	10	20	30	40
Validation Accuracy	0.8601	0.8652	**0.8705**	0.8445
Training Accuracy	0.9934	0.9732	0.9876	0.9992

Table 4.1: Learning rate fixed to 0.001 and momentum fixed to 0.9

4.3. Learning Rate Decision

In another experiment, various learning rates were tested while keeping momentum fixed and batch size fixed, and a learning rate of 0.001 was found to be optimal for this task. The learning rate scheduler helped easing the learning process by reducing the rate by a factor of 0.1 when the validation accuracy stopped improving for 3 successive epochs. The values of accuracy pertaining to different learning rates can be seen from table below.

Learning Rate	0.00005	0.0001	0.0005	0.001
Validation Accuracy	0.7046	0.8031	0.8497	**0.8705**
Training Accuracy	0.7084	0.8120	0.9713	0.9876

Table 4.2: Batch size fixed to 30 and momentum fixed to 0.9

4.4. Momentum Decision

Finally we tried to find a best momentum factor value, this time keeping batch size and learning rate fixed, we got the result that momentum factor 0.96 is performing the best among them. The values of accuracy pertaining to different momentum factors can be seen from table below.

Momentum	0.9	0.92	0.94	0.96
Validation Accuracy	0.8705	0.8653	0.8705	**0.8756**
Training Accuracy	0.9876	0.9659	0.9651	0.9926

Table 4.3: Batch size fixed to 30 and learning rate fixed to 0.001

5. Results and Conclusion

	Validation Accuracy	Test Accuracy
Model 1	0.8756	**0.7946**
Model 2	**0.8860**	0.7762

Table 5.1: Model 1 trained on original, Model 2 trained on preprocessed dataset.

We used our previous experiments in fixing batch size, learning rate and momentum hyperparameters to make a comparison between effects our preprocessing. We trained another model with the same values but on the preprocessed dataset. From their performance results, which are shown on the table above, we have concluded that our preprocessing was not promising because even though there was a slight improvement in the validation accuracy that was compensated by the difference in test accuracies. Further investigations regarding the effects of preprocessing of dataset in deep learning models can be made because it may be the case that deep learning does not require such preprocessing at all.

References

- Žunić, J., Hirota, K., & Rosin, P. L. (2010). A Hu moment invariant as a shape circularity measure. *Pattern Recognition, 43*(1), 47–57. https://doi.org/10.1016/j.patcog.2009.06.017