Digitalizing Skin Cancer Detection with PyTorch

Sean Ashley

July 31, 2020

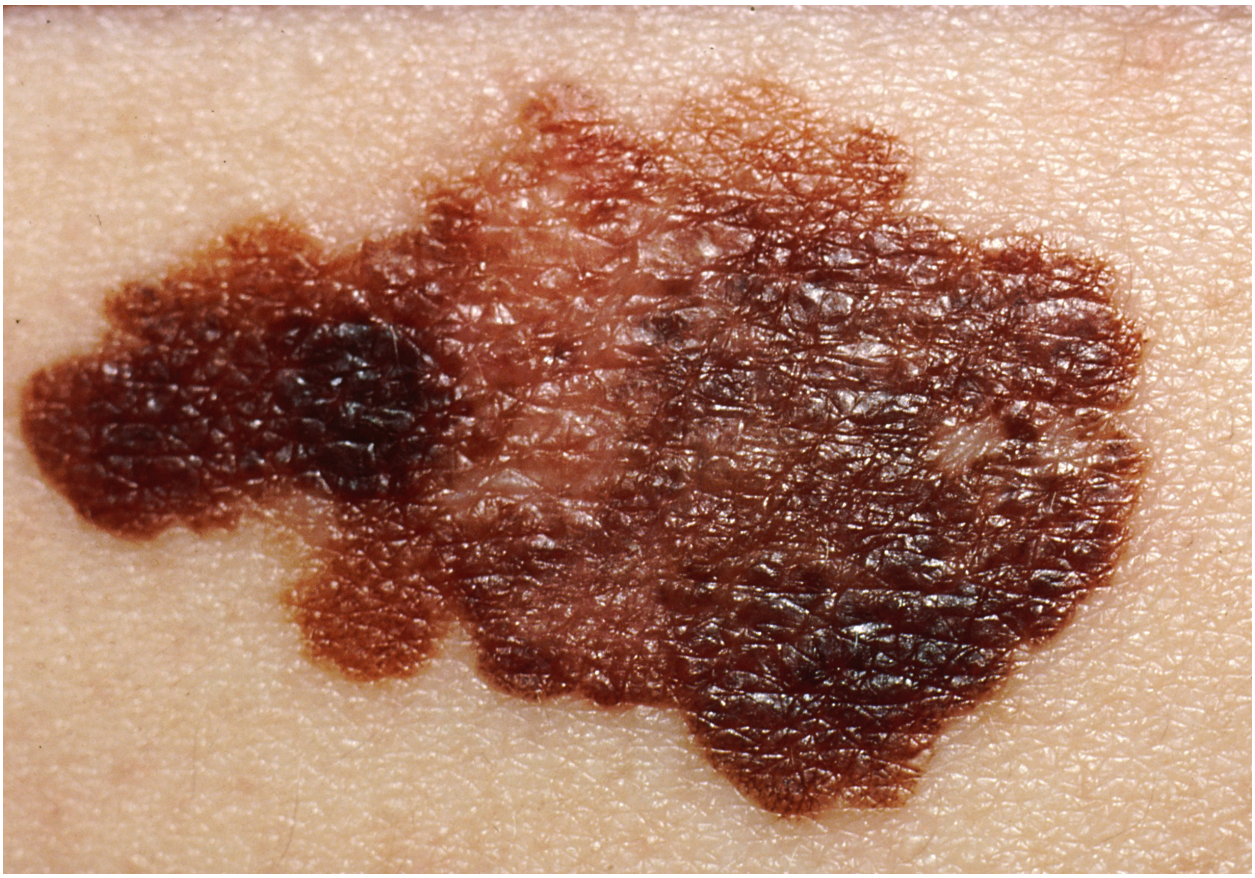

Figure 1 Picture of Melanoma.

1 Introduction

Skin cancer is the most common form of cancer, with 1 in 5 Americans developing it by the time they reach 70 years old. More then 2 people die of skin cancer in the US every hour[1]. Early detection is key to saving lives, with the early detection 5 year survival rate of skin cancer being 99%[1]. Dermatologists have to look at patients one by one, and must assess by eye whether or not a blemish is malignant or benign. Dermatologists have around a 66% accuracy rate in assessing 752 different skin diseases, while Covolutional Neural Networks, such as the one detailed in

Dermatologist-level classification of skin cancer with deep neural networks published in Nature have achieved greater accuracy levels then dermatologists, around 72.1%[2].

By converting cancer detection to easily deployable software, you could allow people to get accurate cancer testing at home, saving resources and time. By making cancer detection more accesible, people would be more likely to get tested, saving lives in the process. Below I will detail my process and results from a melanoma (malignant), nevus (benign) , and seborrheic keratosis (benign) detector model using Convolutional Neural Networks.

2 Process

2.0.1 Dataset

The dataset that was used was retrieved from Udacity's Dermatologist AI mini project. The training set contains 374 pictures of labelled images of melanoma(skin cancer), 1372 labelled image of nevus(a benign blemish), and 254 labelled images of seborrheic keratosis(a benign blemish). The validation and testing set each contain about 20% of the total images in the training set with similiar ratios of melanoma to nevus to seborrheic keratosis. The image was loaded into PyTorch using the ImageFolder class.

2.0.2 Model Architecture

The model architecture that was used was the EfficientNet model detailed in the article *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* by Mingxing Tan, and Quoc V. Le[3]. This architecture was chosen because it is among the highest ranking models in the *ImageNet* contest[4] and is extremely accurate. The model architecture was used completly unaltered except for the last dropout and linear layer. The last dropout layer was upped to 50% probability, and the last linear layer was adjusted to three output nodes to fit the classifying task. The weights were initialized as the *ImageNet* competition weights, which was provided in PyTorch by GitHub user lukemelas[5]. Gradient descent was enabled for all layers, so the weights of the layers were all altered during the training process to better fit the task of classifying various skin lesions.

Figure 2 EfficientNet Model Architecture[6].

2.0.3 Optimizer

The optimizer that was selected was the Adam optimizer, as detailed in the article *Adam: A Method for Stochastic Optimization* by Diederik P. Kingma, and Jimmy Ba[7]. The Adam optimizer was chosen because it consistently performs the highest out of all the PyTorch optimizers, providing fast and effective training.

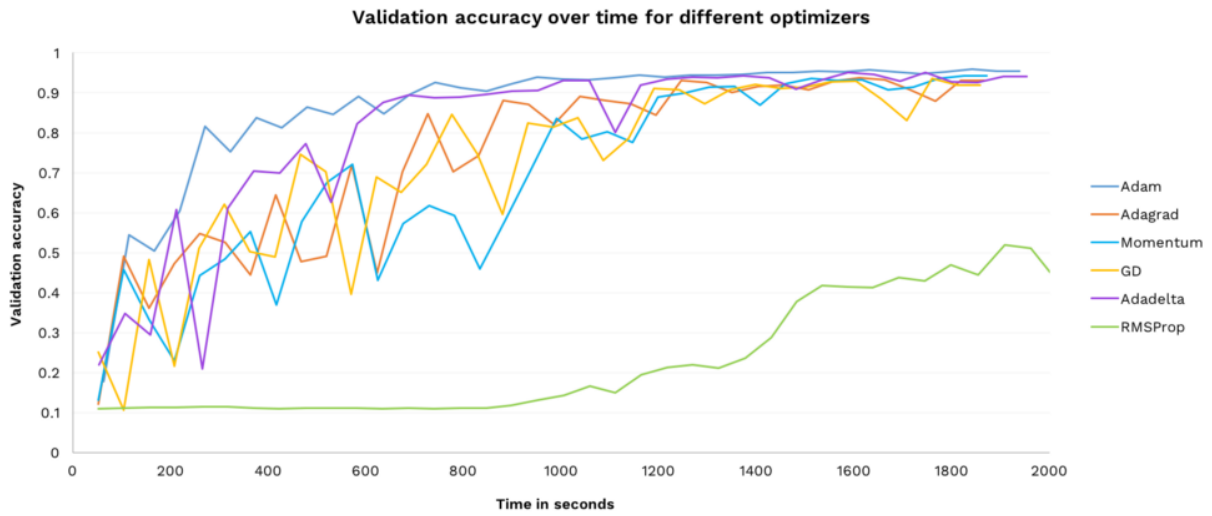

Figure 3 Performance of Adam Optimizer vs Other Common Optimizers[8].

2.0.4 Loss Function

The loss function that was chosen was Cross Entropy loss. Cross Entropy loss was chosen because it is simple to use and effective, with high performance. Data Augmentation using the PyTorch.transforms functions, all of the images were cropped to 299x299 as it allowed the model to run at a reasonable speed while attaining good results. All of the images were normalized according to the *ImageNet* values.

2.0.5 Data Augmentation

The training set was augmented using random rotations, vertical flips, horizontal flips, and crops from the default PyTorch transforms library. Other transforms were tried, such as no augmentation, random affline, gaussian blur, increased noise, and random perspective; however, all resulted in worse accuracy than the augmentation that was used.

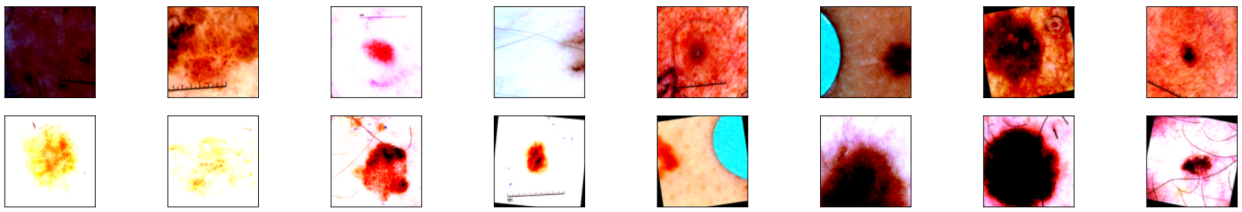

Figure 4 Resulting Images from Applied Data Augmentation.

2.0.6 Number of Epochs and Early Stopping

I implemented early stopping functionality into my model to ensure that no time was wasted during training. By stopping the training loop after 6 epochs of no improvment on validation loss, I ensured that I was not wasting time training my model just to overfit the data set. Additionally,

the model was only saved if it achieved a new best score in validation loss. Implementing early stopping also allowed me to set a large amount of epochs and not have to worry about my code running for too long, as I knew it would just keep going until it was not improving anymore.

3 Results

3.0.1 Final Validation Loss and Test Accuracy

The final model ended up training for 10 epochs and achieved a validation loss of 0.513393, which was the lowest validation loss of any model tested. The testing accuracy of the model ended up being 76%, correctly classifying 456/600 images. This is notably higher then the dermatologist rating of 66%[1], though with notably fewer classes (752 vs 3).

3.0.2 ROC Curve

The ROC Curve of the model points out a glaring fault. The model does exceedingly well at correctly classfying benign lesions from melanoma as a binary task (task_1) but struggles with classifying melanoma out of the 3 possible classes, i.w. melanoma, seborrheic keratosis, and nevus (task_2). This is concerning considering malignant lesions pose the greatest harm to health, and may results in death if missed. The reasons for this discrepancy are most likely due to the abudance of pictures of benign lesions compared to malignant lesions in the training data. There are only 372 pictures of melanoma in the training data, while there are over 1500 pictures of benign lesions (nevus and seborrheic keratosis). This most likely lead to the convulational layer weights becoming overfitted to classifying benign lesions, while doing a poor job at classifying malignant ones. I attempted to add more dropout layers to the EfficientNet model, however it lead to a drastic drop in accuracy. This leads me to believe that the best solution would be to implement more data into the experiment. There are millions of labelled images of Melanoma available from sources such as the *Kaggle SIIM-ISIC Melanoma Classification* competition[9]; however my current equipment setup is not powerful enough to deal with that amount of data. This will be something I will have to look to implement in the future, when I have more resources available.

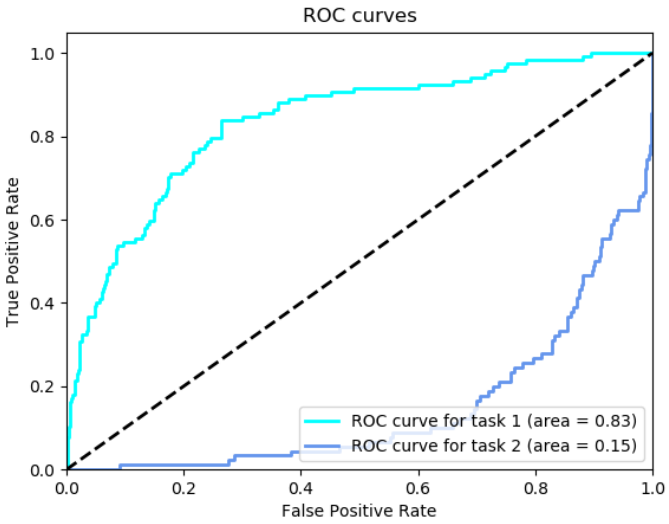

Figure 5 ROC Curve Generated by the Model (Task_1 = classfying benign lesions from melanoma as a binary task, Task_2 = classifying melanoma out of the 3 possible classes, melanoma, nevus, and seborrheic keratosis).

3.0.3 Confusion Matrix

A confusion matrix represents the amount of correctly labelled postives vs correctly labelled negatives vs incorrectly labelled positives vs incorrectly labelled negatives. Positives in this case means the patient has melanoma and negative means the patient has a benign lesion. I set the threshold for considering a skin lesion to be melanoma at 30% (as in if the model believes with a 30% probability the skin lesion is melanoma, it counts it as positive). I chose this because in general, it is safer to misdiagnose someone with melanoma, then to miss a melanoma diagnosis completly, as the latter could lead to death while the former might slightly inconvenience someone. With those parameters, I received 71% correctly labelled positives (correctly labelled as melanoma) and 81% correct labelled negatives (correctly labelled as benign skin lesions).

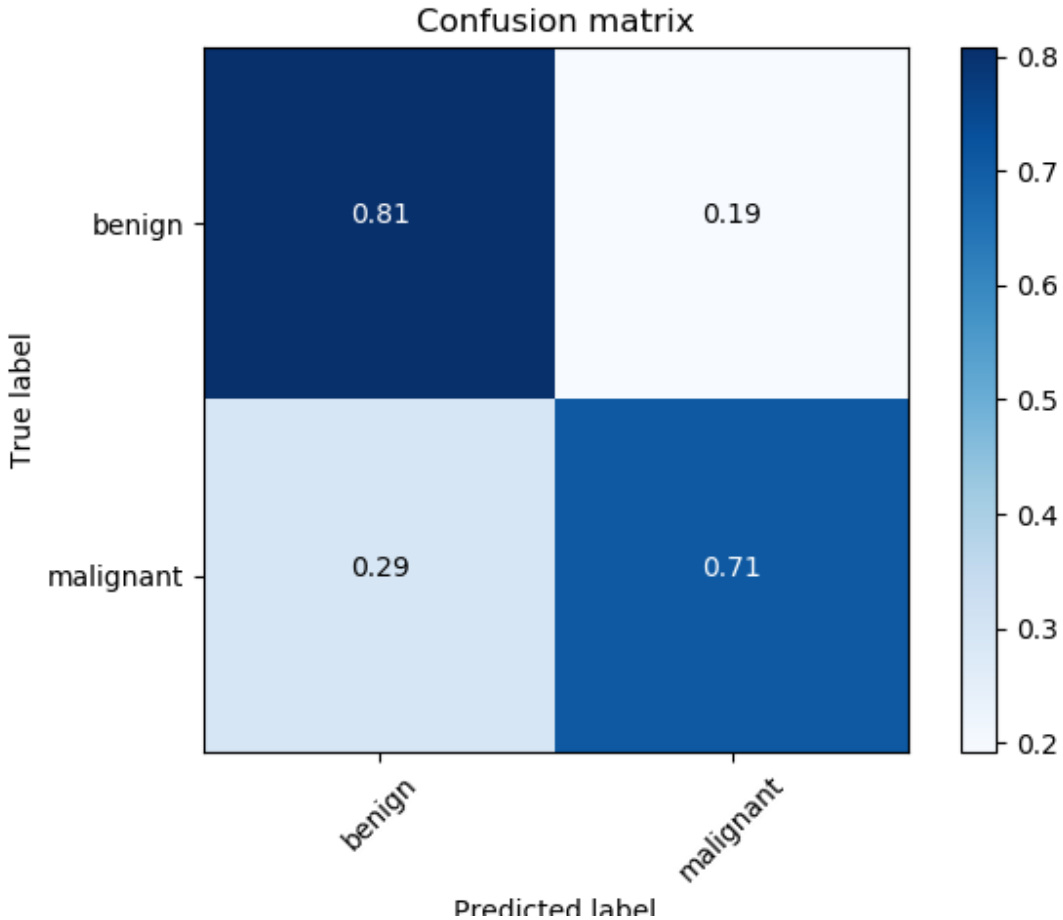

Figure 6 Shows the Fraction of Correctly Labelled Malignant Lesions and Benign Lesions.

4 Reflection

This project was an extremely rewarding experience. It allowed me to see the potential impact that Neural Networks and AI could have on the future. Digitilizing patient care could help speed up the process, make it more accurate, reduce the costs, and make top quality healthcare accessible to everyone. While the results of my trained model were not perfect, they are certainly promising, and I am extremely excited to continue working with Machine Learning. My passion is AI, and I will continue working and improving my skills to help make the world a better place.

5 Bibliography

[1] Skin Cancer Facts & Statistics. (2020, July 10). Retrieved July 15, 2020, from https://www.skincancer.org/skin-cancer-information/skin-cancer-facts/

[2] 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. doi:10.1038/nature21056

[3] Tan, M., Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946 [cs.LG]

[4] Melas, L. (n.d.). Lukemelas/EfficientNet-PyTorch. Retrieved July 19, 2020, from https://github.com/lukemelas/EfficientNet-PyTorch

[5] Papers with Code - ImageNet Leaderboard. (n.d.). Retrieved July 19, 2020, from https://paperswithcode.com/sota/image-classification-on-imagenet

[6] Tan, M., & Le, Q. V. (2019, May 29). EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling. Retrieved July 20, 2020, from https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

[7] Kingma, D. P., Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv:1412.6980v9 [cs.LG]

[8] Mack, D. (2018, April 10). How to pick the best learning rate for your machine learning project. Retrieved July 20, 2020, from https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2cs.LG

[9]SIIM-ISIC Melanoma Classification. (2020). Retrieved July 20, 2020, from https://www.kaggle.com/c/siim-isic-melanoma-classification