Skin Cancer Detection Using CNN (December 2019)

Maryam Vazirabad

***Abstract*— With the desire to research precision medicine, this paper explores biomedical data science with an image classification problem: automating skin diagnoses. The dataset Skin Cancer MNIST: HAM10000 features over 10,000 publicly available images of skin lesions. Using a convolution neural network with Keras TensorFlow in backend, this paper attempts to detect seven different classes of skin cancer using the dataset, and then interprets the results to determine the worth of the model in a practical scenario. The model's validation set reached a level of 75% accuracy, suggesting that while it does relatively well at prediction, further improvements are possible.**

I. INTRODUCTION

Today, data is growing in many different sectors around the world at an exponential rate. The field which has one of the fastest data growths happens to be healthcare, which also arguably holds one of the most worthwhile data in existence. Harnessing this data is incredibly important for things like medical imaging and clinical decision-making. Image processing is a field of study that needs a lot of attention in order to improve patient outcomes and achieve higher quality healthcare. The dermatological image dataset Skin Cancer MNIST: HAM10000 is a large collection of skin lesions. There are over 10,000 images of skin lesions, all of which can be categorized into seven different classifications of skin cancer. The purpose of the dataset is to practice with creating machine learning models to automate the diagnosis of these skin lesions. Because skin cancer is the most common cancer in the United States, precision medicine for dermatology should be one of our focuses. Datasets like this are useful for developing machine learning algorithms that can be beneficial in the medical field and have a significant impact on our world. With this dataset a convolution neural network was built to try and predict seven classifications of skin cancer: Melanocytic nevi, Melanoma, Benign keratosis-like lesions, Basal cell carcinoma, Actinic keratoses, Vascular lesions and Dermatofibroma.

II. RELATED WORK

While researching models to apply to the skin cancer dataset, I came across an article that talked about implementing a convolution neural network to classify histopathology images from the dataset PatchCamelyon (PCam) [1]. The author was attempting to predict cancer via machine learning algorithms, so I wanted to read more about their techniques. They chose a ResNet CNN because they are known to be one of the most powerful types of architecture today for medical image classification. The paper sounded exciting because the author was able to achieve 98.6% training accuracy and they were training with over 300,000 images, so it seemed like a robust algorithm even though their validation loss was high. After their first run of their model, they tuned the model for optimization using discriminant learning rates and 1cycle. They used confusion matrices in order to better visualize the effectiveness of the model, while I simply plotted the training/validation accuracies and losses.

Another paper I came across, "Superior skin cancer classification by the combination of human and artificial intelligence", used the same skin cancer dataset as my project and implemented a ResNet CNN as well [2]. This was an interesting article because it gave statistics of how convolutional neural networks are significantly outperforming dermatologists when it came to diagnosing skin cancer. In their study of detecting skin cancer in five different categories (my project consisted of seven categories of skin diagnosis), they were able to achieve an accuracy score of 81.59%, which vastly surpassed the dermatologist's accuracy score of only 42.94%. It was clever that they not only compared the accuracy scores of the physician and the machine, but that they combined the diagnoses of both to achieve a superior accuracy score of 82.95%. ResNet CNNs are known to be very powerful due to their use of residual blocks, but I decided to see how regular deep neural networks worked on this dataset.

III. DATASET AND FEATURES

For preprocessing of the dataset, I had to load the data and merge the skin lesion images from folders into one dictionary. This was done with the help of pandas and glob libraries [3] [4]. The dictionary will help with one-hot encoding, which will convert categorical data into a form that is best for machine learning algorithms. A unique binary value will be added for each integer encoded variable. I created a key to map the seven skin cancer diagnoses to their shorthand, which was found in a paper discussing this dataset in depth [5]. Afterwards, I made sure to remove all null values in the dataset so that model performance would not be affected.

Before delving into the processing of the data, I had to first

investigate the type of data I was working with. Data exploration is an important part of learning what makes up the dataset [6]. In doing so, I learned some interesting statistics. Looking at a snapshot of the data, I could see that the columns were made up of features such as gender, sex, age, location of lesion, etc. In order to learn more about these features and how they related to each other, I created some plots using the library seaborn that compared things such as gender or age versus each type of skin lesion [7]. One thing that I noticed straight away was that the dataset had an extreme bias for the skin diagnosis, melanocytic nevi. Figure 1 displays the plot of the distribution of skin diagnoses.

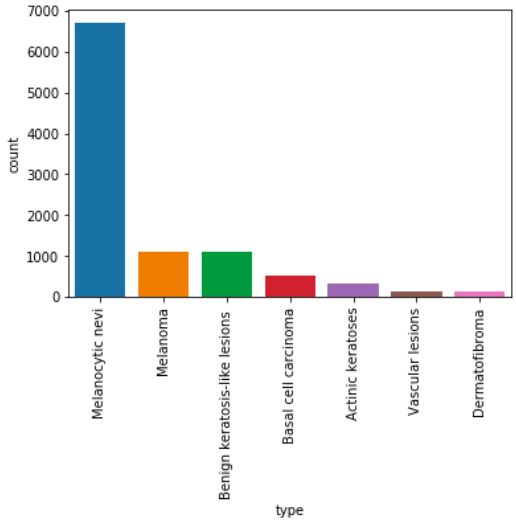

Figure 1: Distribution of skin diagnoses, showing bias for melanocytic nevi

I considered dropping some of the data for this type of diagnosis to try and even it out amongst the other features but ultimately kept it. I also created a box plot to show the distribution of skin cancer for age and sex, displayed in Figure 2.

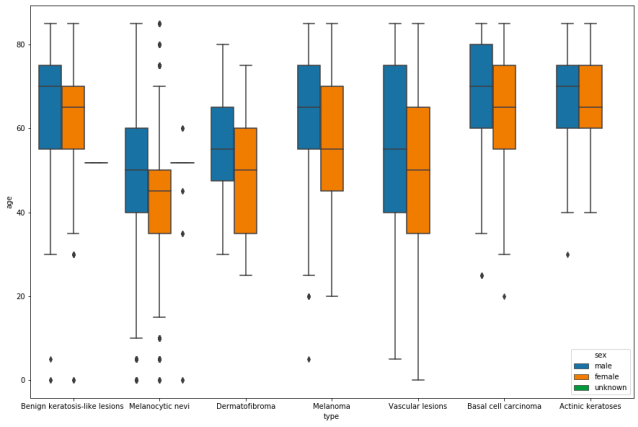

Figure 2: Box plot showing distribution versus age and sex

The images were resized from 600x450 pixels to 32x32 pixels using the library NumPy [8]. I considered only shrinking them down to 100x100 but shrinking them to one third of that value allowed me to more quickly run my model. Figure 3 displays the resulting skin lesion images after resizing:

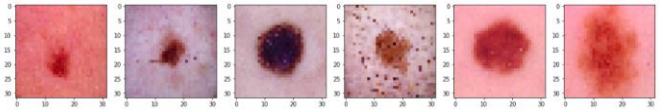

Figure 3: Skin lesion images resized to 32x32 pixels

Afterwards, I split the dataset into test and train sets using the library scikit-learn, with 80% going towards train and the rest towards testing [9]. This resulted in 8,012 images in the train and 2,003 in testing. I then normalized the data so that it can converge faster (accelerates learning). Information on how to implement the preprocessing, including the reshaping of images and normalization was found online [10].

IV. Methods

After processing, I built my convolution neural network using Keras [11]. I decided to use a simple architecture that we saw in class to see how it performed. I added the first input convolution layer, which consisted of 32 filters using a 3x3 window. I used the ReLU activation function. The input shape is the size of the image, 32x32 (and the color channel is a depth of three). I then added a dropout of .2, which is used to prevent overfitting by adding a penalty to the loss function. I added another convolution layer (this time not needing to add the input shape) and then added a pooling layer with a 2x2 pixel filter to get the maximum element from the feature maps. This reduces the feature maps by half and summarizes the presence of features in patches of the feature map. I stacked several more layers with more dropouts and pooling layers, and I also added a flattening layer to reduce the image to a linear array. The very last layer of this neural network has seven neurons for each of the seven skin cancer categories, and uses the softmax function because this is the output layer. The softmax function is used to give probabilities of the categories occurring. After seeing the summary of the model, I compiled it with a loss function of categorical crossentropy because I am using a single label categorization and an Adam optimization (common with deep neural networks). I first ran the model with a batch size of 256, a validation split of 0.3, and 10 epochs. After evaluating the model, it gave an accuracy score of 67% and a loss of 1.0. I was not impressed with these results so I decided to run it with more epochs to see if this would improve the learning. There was a small improvement; my accuracy came to 69%. However, it clearly plateaued and began to overfit. The training sets came out to significantly higher in accuracy than with validation. This concerned me because my dataset is relatively large.

In order to improve my results, I decided to change the overall architecture of the model. I attempted to do this by reducing the number of layers., thereby simplifying the network so it can possibly learn better and improve performance. I came across a few other simple neural network architectures and developed a model that I hoped would have a higher accuracy score. Since I had more than enough examples from which to train on, I did not need to add more. Instead, reducing the complexity of the model seemed like a

good option. In doing so, I reduced the structure and the parameters. A summary of the new, simplified model is shown in Figure 4:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               524416
_________________________________________________________________
dropout_9 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 7)                 903
=================================================================
Total params: 590,887
Trainable params: 590,887
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Summary of model, showing a reduction of the number of parameters in order to mitigate overfitting

After running 50 epochs with the new model, the accuracy came to be about 75% and the loss came to be about 0.75.

V. Experiments/Results/Discussion

After trial and error, my final convolution neural network seemed to be an improvement compared to my previous trials, confirming that reducing the structure and parameters of the model helped in overcoming some of the overfitting, although it still clearly exhibited overfitting. Adding more regularization also seemed to help with the issue of overfitting. I tuned the values of dropout between the layers. Also, in order to increase the accuracy of the model, I attempted to reshape the images to 100x100 pixels. I assumed that perhaps in order to determine the type of skin cancer, it was necessary to have a clearer view of the skin lesion (therefore needing more pixels and overall detail). While this experiment is not shown in the implementation of my project because my report would end up becoming far too long if I included all my experimentation, increasing the size of the images did not have an impact on the accuracy of the model. I even increased the size of the images to 200x200 to see if this had any effect. There was still seemingly no improvement to the performance of the model. This surprised me, but I was happy that I no longer had to run multiple hour-long epoch trials and could return the image sizes to 32x32 pixels. I also continued to play with the number of epochs I ran to see if there would be an improvement in performance. I ran the model with 10, 20, 30, 40, and 50 epochs to see which would be best for my model.

Running my final model with 50 epochs ended up giving me the highest validation accuracy score of 75% and lowest validation loss score of 0.89. It was still clear that the model continued to learn and improve with the training set and so there was still overfitting because the validation set did not do the same.

VI. Conclusion

This paper describes the application of convolution neural networks to the dermatological image dataset Skin Cancer MNIST: HAM10000, a large collection of skin lesions. With a dataset of over 10,000 images of skin lesions, there were a large sum of examples for which to train on. The goal of the deep neural network was to categorize the images into seven different classifications of skin cancer: Melanocytic nevi, Melanoma, Benign keratosis-like lesions, Basal cell carcinoma, Actinic keratoses, Vascular lesions and Dermatofibroma. After trial and error of different types of CNN architectures and the tuning of hyperparameters such as epoch count, I was able to achieve an accuracy score of 75% on the validation set and a loss of 0.75. While I was hoping to achieve a higher accuracy score, I am reminded that real, human dermatologists were only able to accurately diagnose five categories of skin cancer about 43% of the time according to the paper which mostly used the same dataset I worked with [2].

With more time or team members, it would be possible to further change the architecture of the convolution neural network or perhaps try a different classifier altogether. Since there was still clear overfitting, applying more regularization tools or overall simplification of the model could have been beneficial. Because many of the same image classification problems I read about used CNNs and were successful in developing a robust algorithm, I decided that using one myself was a good plan. I do not know a lot about the common types of CNNs such as ResNet, which is very popular, so this could have been a helpful tool that would have potentially improved my accuracy scores.

References

[1] J. W. Wei, L. J. Tafe, Y. A. Linnik, L. J. Vaickus, N. Tomita, and S. Hassanpour, “Pathologist-level classification of histologic patterns on resected lung adenocarcinoma slides with deep neural networks,” Nature News, 04-Mar-2019.H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[2] A. Hekler, J. S. Utikal, A. H. Enk, A. Hauschild, M. Weichenthal, R. C. Maron, C. Berking, S. Haferkamp, J. Klode, D. Schadendorf, B. Schilling, T. Holland-Letz, B. Izar, C. von Kalle, S. Fröhling, and T. J. Brinker, “Superior skin cancer classification by the combination of human and artificial intelligence,” European Journal of Cancer, 10-Sep-2019.

[3] “powerful Python data analysis toolkit,” pandas. [Online].

[4] F. Lundh, “Python Standard Library - glob,” O'Reilly | Safari. [Online].

[5] P. Tschandl, C. Rosendahl, and H. Kittler, “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,” Nature News, 14-Aug-2018.
[6] Pmarcelino, “Comprehensive data exploration with Python,” Kaggle, 23-Aug-2019M. Young, *The Techincal Writers Handbook.* Mill Valley, CA: University Science, 1989.
[7] “statistical data visualization,” seaborn. [Online].
[8] “NumPy,” NumPy. [Online].
[9] “scikit-learn,” scikit. [Online].
[10] A. Nag, “Image Classification using Deep Learning & PyTorch: A Case Study with Flower Image Data,” Medium, 14-Aug-2019.C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.
[11] “Keras: The Python Deep Learning library,” Home - Keras Documentation. [Online].