CS221 Final Project: Classifying skin cancer within a dermatoscopic image dataset

Dawn Finzi and Mona Rosenke

I. Introduction and task definition

Skin cancer is the most common form of cancer in the United States with one estimate putting the number of cases at 5.4 million per year [1]. While the large majority of these cases are non-lethal, early and accurate diagnosis is crucial for improving outcomes, especially for more aggressive skin cancers such as melanomas. In this project, we will be attempting to classify seven categories of pigmented lesions from dermatoscopic images, a task that closely matches the real-world clinical scenario, as all of the main categories of pigmented lesions are represented. We proposed a two-pronged approach to this problem, where we first attempted to classify the network using different convolutional neural network architectures and then delved into the features that the network learned, with the hope of either using the model to inform human performance or incorporating the heuristics and features dermatologists use to discriminate between the different lesion types to improve and simplify our models.

II. Infrastructure

A. Dataset

The dataset we used is an image dataset from Kaggle consisting of 10015 dermatoscopic images of pigmented lesions (Skin Cancer MNIST: HAM10000). Cases include seven main categories of pigmented lesions

- Actinic keratoses and intraepithelial carcinoma / Bowen's disease
- Basal cell carcinoma
- Benign keratosis-like lesions
- Dermatofibroma
- Melanoma
- Melanocytic nevi
- Vascular lesions

where three of the seven are precancerous or cancerous (actinic keratoses, basal cell carcinoma and melanoma) and the other four are largely benign (keratosis-like lesions, dermatofibroma, melanocytic nevi and vascular lesions).

All of the images are labeled based on histopathology (50% of cases), follow-up examination, expert consensus or confirmation by in-vivo confocal microscopy.

B. Concrete Example of Input and Output

For inputs, the system will take dermatoscopic images such as these (Fig. 1) and return one of seven labels corresponding to the seven main categories of pigmented lesions outlined above.

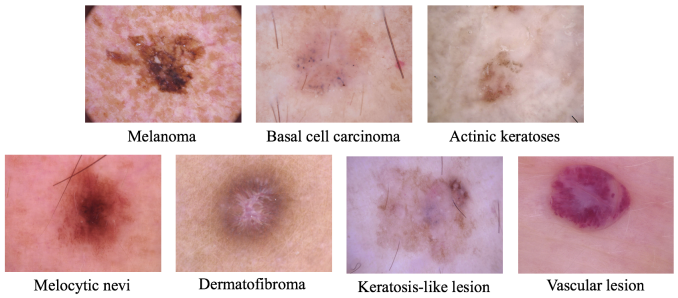


Fig. 1: Example input images for each category and their corresponding labels. Cancerous lesions are on the top row and benign lesions are on the bottom row

C. Benchmarks

1) Baseline: For our baselines, we used the RBG values of 28 pixel x 28 pixel versions of the images and implemented three out of the box models from scikit learn. For all three classifiers we used 80% of the data as training data (8012 images) and 20% as test data (2003 images). We first implemented a linear support vector classifier (LinearSVC) which classified test data with 60.0% accuracy. We then tried logistic regression using the "lbfgs" solver for our multiclass problem. This performed better with 69.7% accuracy on the test data. Finally, we used the default scikit-learn instantiation of k-nearest neighbors classifier with five neighbors. The algorithm also performed decently on the test data, with classification accuracy at 70.1%. Note that chance on

this dataset is 14%, which means that fairly decent performance can be achieved even with these more basic models.

2) Oracle: For our oracle, we initially thought to recruit a board-certified dermatologist who frequently diagnoses and treats melanomas and other skin cancer. We provided the dermatologist with 100 of the images and asked him to label the dematoscopic images based on the seven possible label categories. However, our proposed oracle only achieved 35% accuracy! While he did manage 89% accuracy when diagnosing the lesions as cancerous or benign, his 7-label categorization was far below our baseline performance. We speculate that this may be in part due to many of these lesions needing to be confirmed by histopathology, and also as the dermatologist we were able to recruit does not use dermatoscopy in his daily practice. Regardless, we were forced to explore other options for our oracle.

We then looked at the literature for recent state-of-the-art work that we could use as a benchmark. We discovered that an ISIC Challenge was held for this dataset in the summer of 2018 [3] and the winner of the challenge managed to achieve 89% on test data [4]. We are using this performance as our oracle. This approach uses an ensemble of over 19 models, including many convolution neural networks (CNN), rendering it essentially impossible to use in a real-world clinical setting. Thus it serves well as an upper bound on our simpler attempts.

III. Related Work

Using artificial intelligence to diagnose skin cancer has been a hot topic lately with a highly publicized study pitting 'man against machine' published in the August 2018 issue of Annals of Oncology [6]. In practice, this study compared the performance of a CNN to 58 dermatologists on classifying dermatoscopic melanoma. Using Google's Inception V4 CNN architecture, the authors showed that the neural network model outperformed most dermatologists as measured by sensitivity, specificity and area under the curve (AUC) of receiver operating characteristics (ROC) for classification. Here the classification problem was binary (melanoma vs. non-melanoma), allowing for computation of more sensitive performance metrics based on signal detection theory. However, this research further supports the idea that CNNs can acheive high performance on skin cancer classification problems, as well as illustrating the broad interest in methods for solving such a diagnostic problem.

A second main source of related work stems from the ISIC Challenge [3]. Task 3 of this challenge specifically looked at 7-class classification accuracy on this very dataset. We are using the winner of the contest as our oracle performance.

IV. Implementation

A. Approach 1: Getting familiar with neural networks and the dataset

For our first approach, we used a number of different convolutional neural network architectures to try and achieve high accuracy classification performance on this dermatoscopic dataset.

1) Shallow 3-layer CNN: We started familiarizing ourselves with the dataset and CNNs by implementing a shallow 3-layer CNN using the open source neural network library Keras. Due to the nature of our dataset, which consists only of images of skin lesions, we have a high risk of overfitting a model to our data as the different images do not have a large degree of variability. Consequently, we thought a shallow 3-layer network would not only provide a good starting point, but could potentially provide similar performance to a more complicated network as it less likely than larger models to learn the data too well.

To begin with, we resized the dermatoscopic images, as the original images are fairly high resolution at 450 x 600 pixels, and this level of resolution is both unnecessary and computationally expensive. We reduced the image size to 100 x 75 pixels. We then divided the 10015 images into our train and test subsets, using an 80/20 split. We further subdivided this training set into training and validation sets, using a 90/10 split.

In order to help avoid overfitting, which we had already identified as a potential problem with this dataset, we used data augmentation to artificially expand our limited dataset. We chose to randomly 1) rotate some images by 10 degrees, 2) zoom in on some images by 10%, 3) shift some images vertically by 10% of the height and 4) shift some images horizontally by 10% of the width.

For our 3-layer architecture, we created a linear stack of layers. The first layer is comprised of: two Conv2D layers with 32 filters, 3x3 kernel sizes, and relu activation, one pooling layer (MaxPool2D) with a 2x2 pool size and 25% dropout for regularization. The second layer is similar with two Conv2D layers (though here with 64 filters each), one pooling layer and 40% dropout. The final layer begins with a 'flatten' layer to convert the final feature maps into one one-dimensional vector.

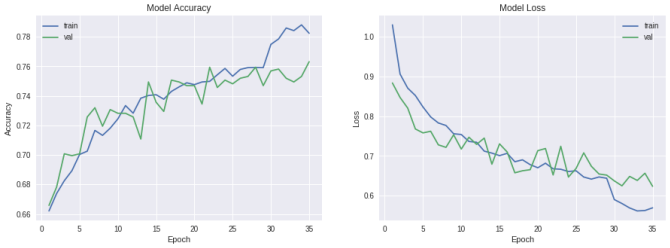


Fig. 2: Model accuracy and loss across epochs for the basic 3-layer network

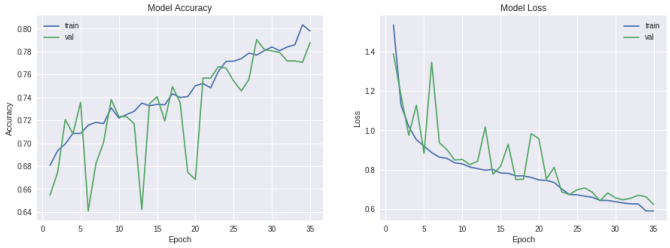


Fig. 3: Model accuracy and loss across epochs for the Resnet18

It then consists of two fully connected layers, separated by 50% dropout, where the last fully connected layer outputs the probability distribution for each of our seven diagnostic classes.

We compiled the model to use a standard Adam optimizer and score based on accuracy. We used categorical crossentropy (i.e. log) loss, defined as:

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbb{1}y_i \in C_c \log p_{model}[y_i \in C_c]$$

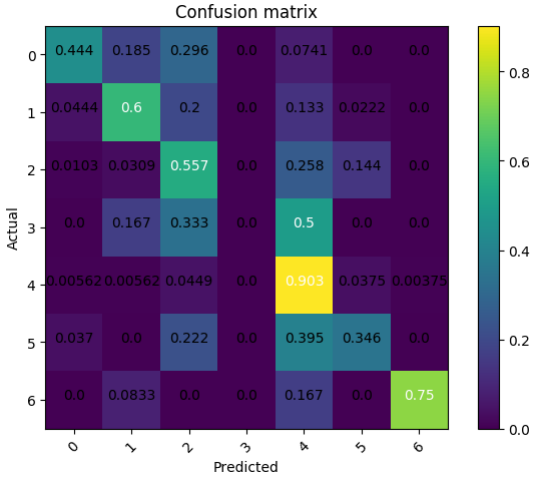
where i corresponds to each instance and c corresponds to each of the classes (in this case, $C = 7$). The indicator term evaluates to 1 if the class label is correct for the observation and $\log p_{model}[y_i \in C_c]$ represents the natural log of the predicted probability that observation y_i is of class c.

We then fit the model, including a learning rate annealer and an early stopper, to save computational resources in the case that validation loss plateaus. We then used batch sizes of 16 and ran the model for 35 epochs.

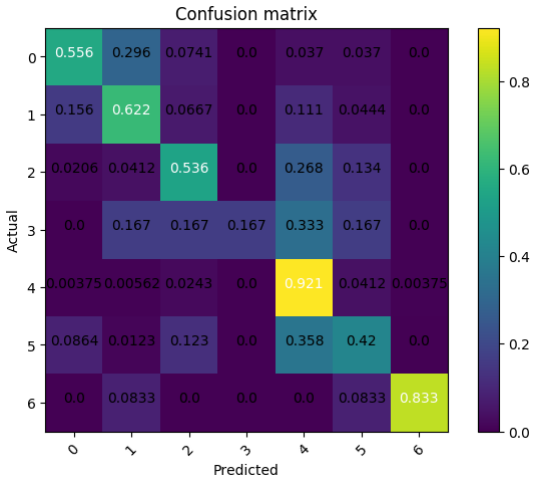
We found that this simple 3-layer CNN actually performed fairly impressively, with an average accuracy of 76% on the validation data. This already outperforms our highest baseline, a k-nearest neighbors classifier which reached 70% accuracy. Additionally, this network has not appeared to overfit the data too badly, as validation accuracy generally tracked training accuracy fairly closely across epochs as shown in Figure 1.

2) ResNet 18: We then decided to try implementing a well-known architecture. We chose ResNet 18 as the ResNet models have been widely used in the field since ResNet won 1st place in the ILSVRC classification competition in 2015, and substantial performance improvements have been demonstrated with ResNet over VGG-16 [5]. We decided to start with the smaller, 18 layer version of the network, ResNet 18.

After importing and building the ResNet 18 architecture, we compiled the model again using categorical



(a) The 3 layer CNN



(b) The ResNet 18 model

Fig. 4: Confusion matrices for the first two models

crossentropy loss and an Adam optimizer. We also used the same train/test/validation splits as for the 3-layer network and kept the batch size at 16 and the number of epochs at 35. This model also performed well on the validation dataset, achieving 79% accuracy. However, despite being a much larger network, this was only 3% performance improvement over our previous model and the confusion matrices for both were fairly similar (see

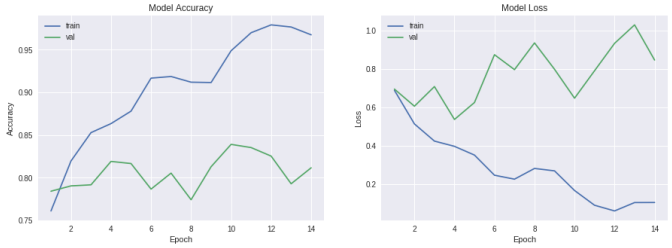


Fig. 5: Model accuracy and loss across epochs for the ResNet50 architecture

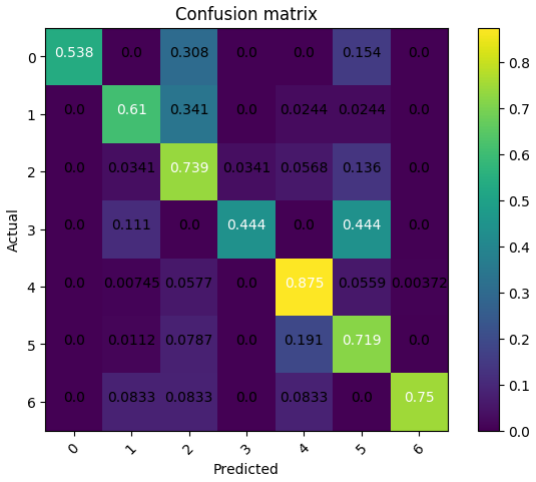
Figure 3).

One thing we noticed while working with the dataset for the 3-layer CNN and ResNet 18 was that the dataset is very unbalanced, with certain lesion categories highly over-represented. Specifically, melanocytic nevi consisted of 6705 out of our 10015 images, with dermatofibroma comprising only 115 examples. This is not something we took into account when creating our train and test splits in these first two models, but do address in our ResNet 50 instantiation (see below).

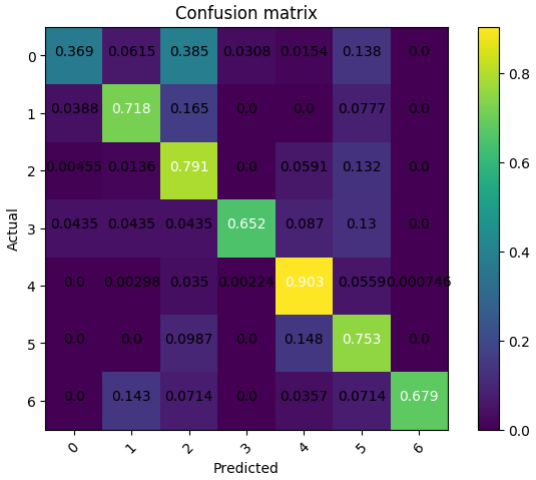
3) ResNet 50: Lastly, we used ResNet 50, a larger version of the ResNet 18 architecture detailed above. We were able to import the main architecture using a Keras application, and we added custom fully connected final layers after the last activation layer to suit our needs. First, we flattened the ResNet 50 output. We then added three fully connected ('dense') layers with relu activation, separated by 20% dropout. As in the simple 3-layer network, the last fully connected layer is designed to output the probability distribution for each of our seven diagnostic classes.

Due to the above mentioned imbalance in images with one category representing nearly 70% of the dataset, we decided to sample the training and validation set proportionally in this instantiation. Specifically, when selecting the 80% of the data for training the model, we selected the relative amount of examples for each labeled category (e.g. if one category makes up 70% of the data, it will also make up 70% of the training and validation sets).

As in the previous models, we used data augmentation to help with overfitting and kept the batch size at 16 and the number of epochs at 35. We again compiled the model to use categorical crossentropy loss and used an Adam optimizer with a learning rate annealer, starting the learning rate this time at 0.0001, lower than the default parameter of 0.001 provided in the original paper in order not to overshoot [7]. As ResNet50 is a large and powerful architecture, and our dataset is fairly limited



(a) Validation data



(b) Test data

Fig. 6: Confusion matrices for the ResNet50 model

compared to more complex and expansive datasets such as ImageNet, we were less concerned that the model would take too long to descend. We then fit the model, again including an early stopper to avoid overfitting and unnecessary use of computational resources. This ended up being useful as model performance plateaued and stopped after only 14 epochs (see Figure 5).

This model performed well, achieving 81% on the validation dataset and 84% on the left out test dataset! We are pleased with this performance considering that the oracle against which we are benchmarking our performance, an ensemble of over 19 models, achieved 89%, only a 5% performance increase.

4) Error Analysis: As stated above, the ResNet50 model that we trained on the dermatoscopic data reached 81% on the validation dataset and 84% on the held out test dataset. As we can see in the confusion matrices for both the validation and test datasets (see Figure 6), per-

Fig. 7: Example misclassified images within the test dataset

formance is fairly consistent across the lesion categories, at least in comparison to the extremely skewed performance we saw in our first two approaches where we did not stratify our train and test splits. Interesting, the profile of misclassifications is slightly different for the validation and test data. For the validation dataset, actinic keratoses and basal cell carcinoma are commonly misclassified as benign keratosis-like lesions and dermatofibromas are frequently misclassified as melanocytic nevi. In the test dataset, however, the main misclassification proportionally appears to be confusing actinic keratoses for benign keratosis-like lesions (a few examples of test misclassifications can be seen in Figure 7). Note that for all confusion matrices in this paper, classifications are normalized by category based on the number of actual instances in each category.

In order to get a more nuanced view of our model performance, we also looked at the precision, recall and f1-scores for each of the seven lesion categories. Precision is defined as the number of true positives over the sum of the true positives and the false positives, while recall is instead the number of true positives over the sum of true positives and false negatives. An f1-score seeks to balance precision and recall and is defined as:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

This is an important measure for our dataset as we have highly imbalanced classes and want to minimize both false positives (such as misdiagnosing a benign lesion as cancerous) and false negatives (such as missing a dangerous melanoma). Luckily, the weighted average f1-score for the test data was 0.85, on par with our basic accuracy measure. However, there was a lot of variation by lesion class, with the maximum f1-score at 0.93 for melanocytic nevi, which is also the class with the largest number of instances in the dataset. The lowest f1-score was 0.51 for actinic keratoses. Future work may want to focus on increased precision and recall for certain classes over others, as while the difference between two benign classes (such as benign keratosis-like lesions and melanocytic nevi) is not very important clinically, the distinction between a benign and cancerous tumor is crucial.

B. Approach 2: Understanding the network through network visualization tools

For our second approach, we started by use different network visualization tools to try and get a sense of ”how” the network was classifying the images.

1) Feature Visualization: As we were initially interested in the differences between the heuristics humans use to tackle this problem and the features neural networks rely on, we decided to visualize the filters at different layers of the network (also known as feature visualization). The general idea is to create an input image that maximizes the filter activations using gradient ascent [8], [9]. Based on [10], [11], we used keras backend functions to maximize the activations of specific layers in our network. We normalized the gradient of the pixels of the input image to smooth gradient ascent and then ran gradient ascent within the input space for 500 iterations.

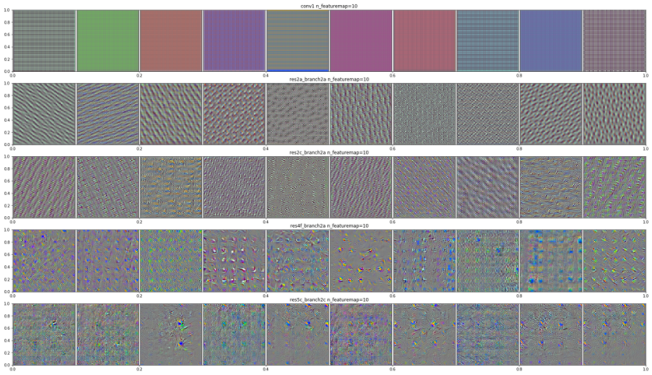
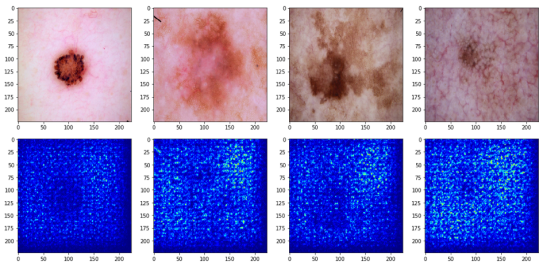


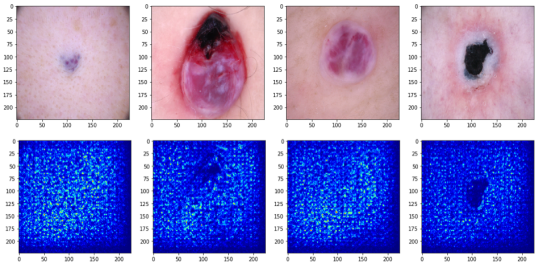
Fig. 8: Feature maps for different convolutional layers in the network (starting at conv layer 1 and ended at the last conv layer)

As you can see in Figure 8, the first convolutional layer seems to encode direction and color, as we would expect. As we move along the network more complex patterns begin to emerge, although the filters still seem fairly dot and grid-like, even at the last convolutional layer, which may reflect the fairly uniform nature of images within the lesion dataset.

2) Saliency Maps: We next wanted to see if we could visualize where, on specific input images, the network was ”attending”, with the hope that this would provide a sense of what features or attributes of an image were helpful for classification. Practically, this involves computing the gradient of the output class with respect the input image and using this to highlight input regions that cause the most change in the output. This technique



(a) Melanoma



(b) Vascular lesions

Fig. 9: Example saliency maps for two lesion classes

is called "saliency mapping", established by Simonyan and colleagues in 2013 [12]. To do this, we used the keras-vis package [13]. We ran saliency mapping on example images from two lesion categories, one cancerous (melanoma) and one benign (vascular lesions). We tried both guided saliency and rectified saliency. While rectified saliency didn't appear to work, guided saliency seemed to be picking up on certain aspects of the image (see Figure 9), particularly boundaries.

C. Approach 3: Human feature integration

Lastly, we were interested in incorporating human-like features by preprocessing the images and re-training the last layer of ResNet50. Our rational was that it would be interesting to see how the model does when we include certain features, in order to evaluate how effective human feature selection is. For that means, we asked our first oracle, the dermatologist, to give us some more details on how he classifies different skin lesions. He mentioned that one of the things he looks for is the border of a lesion and to what degree it changes.

1) The 'to-go' model: In order to make this approach more feasible for real-life clinical settings, we tried to develop a "to-go" model by only used a subset of the data for ResNet50 re-training purposes. Our rational was that the ideal case would be for a clinician to be able to use pre-trained models quickly by applying a filter to the image and classifying the images in minutes. The original dataset consists of 10015 images, of which we used only 1% of each label as a training set and 0.4% as a validation set. Importantly, due to the above mentioned imbalance in images with one category representing 80% of the dataset, we started to sample training and validation set proportionally. Specifically, when selecting the 1% of the data for training the model, we stratified the images proportionally (e.g. if one category makes up 80% of the data, it will also make up 80% of the training and validation sets). Using the to-go model, we reached a classification accuracy across the 7 classes of 58%, using 6 epochs and 701 iterations per epoch. This will be used as our baseline to assess how much influence preprocessing has on the models accuracy.

2) Edge filter: In order to test how sharp edges in lesions influence the models prediction, we used an edge filter applying the Canny method to find edges in each lesion image. We applied a threshold of 0.3 for the higher threshold, which the filter uses to look for local maxima in the gradient. An example can be seen in Figure 10. We re-trained the model after freezing the first 10 layers and updated the remaining layers. We choose a slow learning rate (3e-4), to not drastically adapt the model's layer to our images, and reduced it with advancing layers.

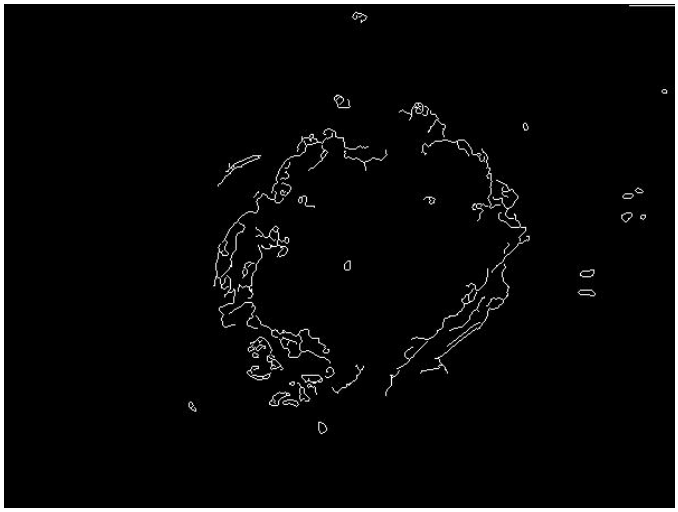


Fig. 10: Example lesion image after edge filtering

Using our to-go approach and the pre-trained ResNet50, we achieved an accuracy of 62%, which 4% better than using the same approach with the raw images.

3) Gradient filter in x-direction: Next to edge filters, another feature in the image that can capture sharp edges in images is the gradient of the image. This is best captured by applying a directional gradient filter to the image, and more specifically to the depth of the image

(x-direction). We implemented the filter by using the Sobel method (see the imagePreprocessing.m script in our github repository for details). The same image shown in Figure 10 can be seen in Figure 11 after applying the directional gradient filter.

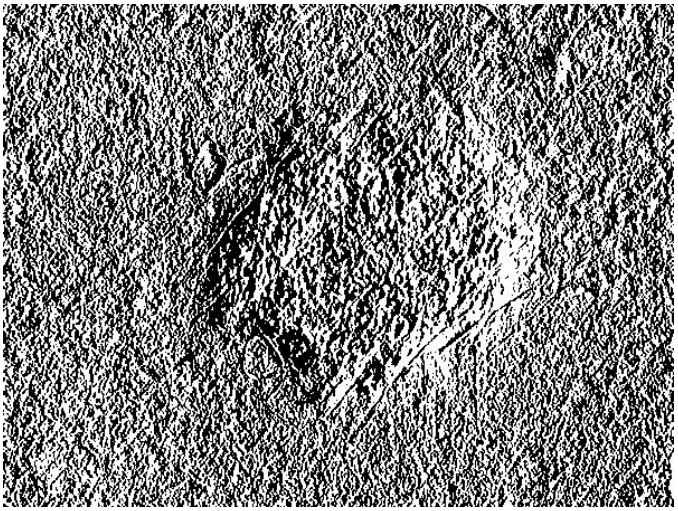


Fig. 11: Example lesion image after directional gradient filtering

Using our to-go approach and the pre-trained ResNet50, we achieved an accuracy of 64%, which 6% better than using the same approach with the raw images, and 2% better than the same approach after edge filtering.

V. Conclusion

This project explored using convolutional neural networks for medical diagnosis, specifically for differentiating between different classes of benign and cancerous lesions. All code for this project can be found at `https://github.com/monarosenke/CS221project`. We showed that even a simple three layer network could achieve 76% accuracy on this 7-class task, while the more complicated ResNet50 model reached 84% accuracy on a held-out test dataset.

We then tried to gain insight into how the network was achieving this using network visualization tools, namely feature visualization and saliency maps. While both were interesting, and the feature maps provided a good sanity check, they were difficult to interpret. Borders and texture changes appeared particularly important but this was not wholly surprising. Future work could potentially extend this by using saliency maps to systematically identify which parts of the dermatoscopic images are most relevant for classification.

Ultimately, we tried to integrate human ratings and neural network learning by preprocessing the dermatoscopic images according to features described by a practicing dermatologist. In addition, we explored how to make this approach more feasible for use in daily life by practitioners by dramatically reducing the amount of images used for retraining. Our analyses show that we improved the network's classification ability by preprocessing the image according to our human feature. However, the reduced-retraining approach predictability suffers a loss in overall prediction accuracy. Future work can try to disentangle the improvements of human features and data set reduction by trying to maximize both approaches before combining them. Furthermore, collecting more human features by interviewing more practicing dermatologists could further advance our understanding of (a) the usefulness of human feature integration into neural network classification and (b) potentially inform dermatologists about optimal feature selection.

References

[1] `https://www.cancer.org`

[2] Nithin D. Reddy, *Classification of Dermoscopy Images using Deep Learning.* arXiv preprint, arXiv:1808.01607, 2018

[3] `https://challenge2018.isic-archive.com/leaderboards/`

[4] Aleksey Nozdryn-Plotnicki, Jordan Yap, and William Yolland, *Ensembling Convolutional Neural Networks for Skin Cancer Classification.*

[5] `https://medium.com/@14prakash`

[6] Haenssle HA, Fink C, Schneiderbauer R, et al., *Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists.* Annals of Oncology, 2018.

[7] Kingma, Diederik P., and Jimmy Ba. *Adam: A method for stochastic optimization.* arXiv preprint, arXiv:1412.6980, 2014.

[8] `https://raghakot.github.io/keras-vis/visualizations/activation_maximization/`

[9] `https://distill.pub/2017/feature-visualization`

[10] `https://fairyonice.github.io/`

[11] `https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html`

[12] Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. *Deep inside convolutional networks: Visualising image classification models and saliency maps.* arXiv preprint, arXiv:1312.6034, 2013.

[13] Kotikalapudi, Raghavendra and contributors. *Keras-vis.* `https://github.com/raghakot/keras-vis`.