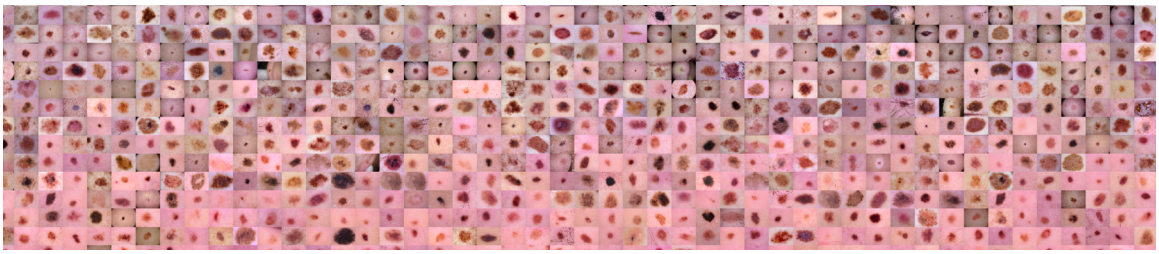

Figure 1: Skin Cancer MNIST: HAM10000

Dermoscopy Image Analysis

Andrei Bunu, Tudor Cocaină, Georgiana Loba, Cristian Pop

January 3, 2024

Abstract

As skin cancer is the most common type of cancer globally, non-invasive dermoscopy tools are becoming more and more essential for early detection and treatment. Our project aims to build an ensemble of neural networks model that can classify skin lesions as either malignant or benign. The ensemble will conatin to two models which use transfer learning and two which are built from the ground up.

1 Introduction

Skin cancer is by far the most common type of cancer. The good news is that, for the most part, it is very treatable especially in the case of early detection. Our dataset includes two types of skin cancers: Melanoma and Basal cell carcioma (abreviated as BCC). BCC is the most common type of skin cancer. BCCs grow slowly, most are curable and cause minimal damage when caught and treated early. On the other hand, Melanoma is the most aggressive and serious type of skin cancer. It is also highly treatable if it‘s detected early in its development.

Dermatolgists have managed to point out characteristics of skin cancers such as shape, size, symmetry and color. Although skin cancer can usually be diagnosed by using a dermatoscope or even by the naked eye, there are cases when skin lesions can become hard to differentiate. Our model aims to predict if a skin lesion has the potential of being malignant, so that it can further be investigated if necessary.

2 State of the Art

Due to the widespread of the disease there exist a lot of machine learning approaches that try to solve the problem of categorizing skin lesions. In the following we will present two approaches, a simpler, and closer one to our implementation and another one which is one of the more complex algorithm that has been proposed for the problem in hand. Both are deep learning approaches. We decided to focus on deep learning since the methods showed remarkable accuracy, compared to the old ones, which were relying on handcrafted features that were fed to KNNs, ANNs and SVMs.

I. The simpler approach, Romero López et al. [2017] is benchmarked on an ISIC dataset of skin lessions. To make the predictions 3 aproaches were used, everyone one of them has as base network the VGGNet, for which the last FC layer was switched into one that does binary classification:

- Train the whole network bottom to top
- Train only the new FC layer
- Train the last Conv layer as well as the FC layer

Results for the methods are as follow:

	Loss	Accuracy	Sensitivity	Precision
M1	0.6743	66.00%	0.5799	0.6777
M2	1.0306	68.67%	0.3311	0.4958
M3	**0.4337**	**81.33%**	**0.7866**	**0.7974**

Figure 2: Adrià Romero et al results

As expected, the transfer learning method with a trainable conv layer achieved the best results. A **strength** of this algorithm would be the fact that it achieves better performance than a specialized human can (about 65-70%) and than the state of the art at that point in time (50.7% and 64 %). A **weakness** would be the fact that the models were trained on a particularly small dataset of 2000 images, the effect of this can be seen in the fact that the second method overfitted.

II. The second approach Esteva et al. [2017], is a very ambitious one. It manages to successfully classify classify 757 skin deseases, without restrictions for images, as in the previous case where they had to be dermoscopy images. The model can even predict diseases from pictures taken with a smartphone camera. Transfer learning is again used, this time the base network being Google's Inception pretrained again on Imagenet. The interesting part of the algorithm is the fact that it introduces a taxonomy composed of 2.032 individual diseases arranged in a tree structure with three root nodes representing general disease classes: benign lesions, malignant lesions and non-neoplastic lesions. Along with this, it introduces a diesease partioning algorithm designed to leverage the

taxonomy to generate training classes whose individual diseases are clinically and visually similar in order to take advantage of fine-grained information extracted from the images. Results of the algorithm were compared against 21 board ceritified dermatologists on multiple different questions like: would you biopsy/treat the lesion or reassure the patient (an actual in-clinic task) or do you believe a lesion is malignant or benign. In all cases the model achieved a better results than the ones of the specialized personal. The most significant one being the classification of skin lesions in one of the 3 main classes, in which the model achieved 72.1% overall accuracy compared to 65-66% accuracy for humans. One **strength** of this algorithm is the fact that it has a high accuracy even on smartphone taken images. Another one that is that it was trained on a very large dataset so it is able to generalize well on test sets as well. One potential **weakness** would be the fact that the CNN has initialized weights from Imagenet. Maybe it would be worth training the whole model?

3 Proposed Approach

3.1 Architecture

Our initial idea was to build a simple model from scratch and train it on the dataset. The initial results were promising but, the model started to overfit on the validation set at about 76% accuracy, so we thought of a way to improve our approach. We settled on creating an ensemble of networks, out of which 2 use transfer learning and the other are 2 of our best models built from scratch.

The first one is built over **VGG19** to which we changed the last FC layer so it outputs a binary result, and then trained the entire network on our dataset. We chose VGG19 as it showed very good results on the Imagenet dataset so there was no reason it couldn't do the same on what we consider to be an 'easier' dataset.

As our second model, we chose **Inception V3** as the base since it's the culmination of many ideas developed by researchers over the years and it attained great results in practice on the Imagenet dataset. For personalising it to suit our needs and skin cancer images, we then added some Dense layers paired with Dropout on top.

We considered **Transfer Learning** to be the best approach since we wanted something efficient and highly beneficial to our goal. By using an already trained model as out starting point, we spared training time and resources and only focused on adapting and perfecting it to our dataset. The picking of the two pre-trained models out of the numerous models out there was a difficult task since the Keras api offers 18 of such models trained on "Imagenet", yet our picking was done taking into consideration numerous papers such as Nivrito et al. [2016] and Marcelino [2018].

We picked the other two models after a **random grid search** in which we tried to find the optimal configuration of weight initialization, number of conv and dense layers, number of filters, kernel sizes, dropout rates, type and size

of pooling layers and lastly, optimizers. After the search we kept the 2 best models:

- 3 Conv layers with relu activation of 32 filters and kernel size of 3, after each an MaxPool layer of pool size 3, then 1 dense layer of 64 units with dropout of rate 0.5 and an output layer, compiled with Adam optimizer.
- 2 Conv layers with relu activation of 16 filters and kernel size of 3, after each an MaxPool layer of pool size 3, then 1 dense layer of 64 units with dropout of rate 0.3 and an output layer, with uniform distribution weight initialization and RMSProp optimizer.

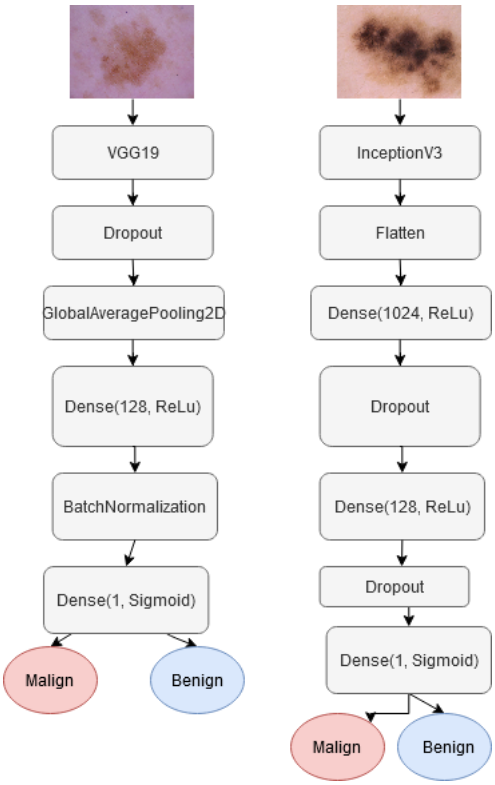

Figure 3: Transfer Learning Models

3.2 Data Preprocessing

File preparation. The dataset was originally spread across 2 folders. We wrote a script that merged it into a single folder, then looks over the data in the csv and copies every image in its coresponding directory (malign or benign), in order to be able to use **Keras's ImageDataGenerator**, which is a class that

offers lots of functionalities like augmenting the data, resizing it or generating a dataset that yields batches of an inputted size of images from the subdirectories malign and benign, together with labels 0 (benign) and 1 (malign).

Data Augmentation. The dataset was heavily unbalanced initially. Out of the 10,000 pictures, 8,000 were of benign lesions and only the remaining 2000 of malignant, so we decided to augment the malign class in order to balance it. Data augmentation is a means of balancing the data, by increasing the data set. It is common way of preventing overfitting, especially when working with images. By making alterations to an existing image, the model will treat it as a new different one. There are several ways to apply data augmentation, out of which we used the following:

- Flipping the image (both vertically and horizontally)
- Rotating the image 180 degrees
- Scaling
- Cropping
- Translating the image (in order to force the model to look everywhere in the image)

The last steps in data preparation were **resizing** and **normalization**. Before feeding the data to our model, we resize it to 224 pixels and then we normalize it by dividing every pixel of every image with 255 so we have only values between 0 and 1.

3.3 Training Process

Our model is implemented using **Keras**, which is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation library TensorFlow.

Models in Keras are defined as a sequence of layers. A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor

We started out with a basic model, with one Conv layer with Relu activation and maxPooling and then a FC layer with dropout in front of the output layer. This approach overfitted quickly, training accuracy reaching high 90s while test accuracy never breaking the 75% mark. As we had already tried to solve the overfitting concern by augmenting the data, we decided to try finding a better model. Every one of our models uses as loss function the binary cross-entropy function which is the standard for binary classification problems, as we have here. Cross-entropy will calculate a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1. The score is minimized and a perfect cross-entropy value is 0.

Random Grid Search. While searching for a better model configuration we decided to automate our work and use a random grid search. The principle

of it is simple: have a list of possible values for every parameter that you want to change and from the permutations of those values select some at random, train the model to see some intermidiate results and keep the best one(s). The parameters we tried to search for are: dropout rate, number of hidden layers (conv and FC), number of filters in the hidden layers, size of the kernel and the padding, type of pooling (avg or max), weight initialization and finally, the optimizer. Though as expected, the performance of the models still wasn't very good. We decided to keep only the two best ones that we mentioned in the Introduction.

Transfer Learning. As the results were not satisfactory enough, we started to dig into transfer learning. This technique is an optimization, allowing rapid progress, consisting of taking a well-trained model for a task and re-purposing it as a starting point for another. Papers such as Caleb [2020] present the blooming of this technique in recent years and the reasoning behind it. Due to the vast amount of data, training from scratch a model is a tedious job, taking a large amount of computational power and time. All this considered, we also jumped abroad the trend and used the technique of transfering the learning for relevant usage on our problem.

The first try at transfer learning was starting with VGG16 as base model with weight initialization from the Imagenet dataset, on top of which we add added a fully connected then an output layer with sigmoid activation to make a prediction, that we then fully retrained on our dataset. The results were much better than anything we managed before, but after about 20 epochs of training, that took a lot of time since we were training the whole network, the model started to overfit, train accuracy going up to 97% while test accuracy staying at 85%.

The second approach was made with Inception V3. We added on top Dense Layers paired with Dropout. This pre-trained model by Google promised faster computing and performance, however, we encountered the issue of **exploding gradients** right from the beginning of the training, the loss value jumping to inf and then to nan. This was a problem that we tackled by clipping the gradients (if the gradients ever surpassed the value of 1.0) and adding weights regularization. This changes brought the gradients under control, and thus, the model started performing properly. Further on, we experimented on different optimizers from Keras, such as **SGD**, **Adam**, **Nadam** and **Adamax**. We trained the inception model 10 epochs using each and the results were interesting, showing the need of an adaptive gradient loss.

InceptionV3 with different optimizers				
	SGD	Adam	Adamax	Nadam
in training	55.15%	83.99%	81.90%	**84.21%**
in testing	58.68%	77.24%	75.84%	**77.54%**

As seen in the table above, SGD gave the worst results since the model is deep and the optimizer cannot escape the local optima and converging slowly. Adam and Nadam gave the best results both in training and in testing, yet we decided to continue with Nadam since the Nesterov Momentum added to the

Adam optimizer gave slightly better results.

We trained the model for 15 epochs using Nadam and binary cross-entropy as the loss function and the results were promising, the loss function decreasing towards 0 and the accuracy increasing. However, we had a 20% difference between train and test accuracy due to overfitting our model as it wasn't generalizing well from our training data to unseen data (the inception model being extremely complex). We tried numerous regularization techniques to tackle the issue, adding dropout and tuning parameters in experimental ways. Afterwards, in training, we managed to reach a **97.32%** accuracy and in testing, the accuracy moved towards **86.62%**.

Although the results were satisfactory, we decided that the test accuracy was still not high enough and we researched on methods to improve it. The inception model had its layers frozen in our first training attempts, we were thus only training the layers added on top of it. Upon research, we decided to experimentally unfreeze the last 150 layers of the model and see how the accuracy in train and test will pan out. The training took a long time, since we were attempting to train large, complex layers, however, the results were impressive, as seen in figure 4 with 10 extra training epochs.

```
Epoch 00005: val_accuracy did not improve from 0.87226
393/393 [==============================] - 3371s 9s/step - loss: 0.2162 - accuracy: 0.9967 - val_loss: 0.6011 - val_accuracy: 0.8653
Epoch 6/10
393/393 [==============================] - ETA: 0s - loss: 0.1676 - accuracy: 0.9963
Epoch 00006: val_accuracy did not improve from 0.87226
393/393 [==============================] - 3318s 8s/step - loss: 0.1676 - accuracy: 0.9963 - val_loss: 0.5511 - val_accuracy: 0.8723
Epoch 7/10
393/393 [==============================] - ETA: 0s - loss: 0.1297 - accuracy: 0.9977
Epoch 00007: val_accuracy improved from 0.87226 to 0.88523, saving model to 2_1
393/393 [==============================] - 3374s 9s/step - loss: 0.1297 - accuracy: 0.9977 - val_loss: 0.5056 - val_accuracy: 0.8852
Epoch 8/10
393/393 [==============================] - ETA: 0s - loss: 0.1030 - accuracy: 0.9967
Epoch 00008: val_accuracy did not improve from 0.88523
393/393 [==============================] - 4960s 13s/step - loss: 0.1030 - accuracy: 0.9967 - val_loss: 0.4894 - val_accuracy: 0.8792
Epoch 9/10
```

Figure 4: Inception with last 150 layers unfrozen in training

We acquired accuracy of **99.78%** in training and when it came to test, the accuracy reached **88.53%**.

Ensemble Model. Ensemble methods are a useful technique that use several different models in order to achieve a better result.

Having more than one model predict on output generally helps minimize noise and variance. It can also help reduce the possibility of overfitting. We created one such model, varying the combinations and methods of combining the models. We used pre-trained saved models, as training each one of them would have been time consuming, and we needed to try different options.

There are several ways to combine the results. The most straightforward method is to count the number of times each possible output was predicted. For example, if 5 models predicted 1, while only 3 predicted 0, you would take the most common output. However, the results were not satisfactory, because this method is more appropriate for multiclass classification problems.

The probability is more useful in a binary classification. After using the average probability for the ensemble output, we could saw an improvement. The final step was taking into consideration the accuracy performance for each pre-trained model, as the prediction of a more precise model is more important than a less performant one. Therefore we calculated their weighted average, using as weights the accuracy percentage from each model. We selected our best models and try different combinations. Figure 5 shows the evolution of the ensemble predictions for every added model. We can observe that after the first three models the ensemble is more accurate than any of the used models, but the final accuracy is still less than the model with 88.5%.

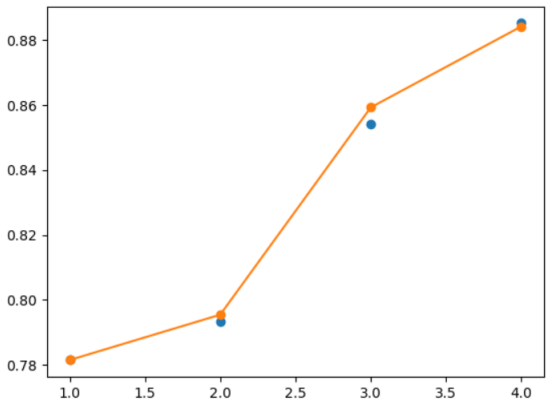

Figure 5: Ensemble method with the top 4 models

Although the the predictions are used together with the appropriate weights, we concluded that using a larger number of models is difficult to control. After removing the first model, our final result improved, as shown in the figure 6.

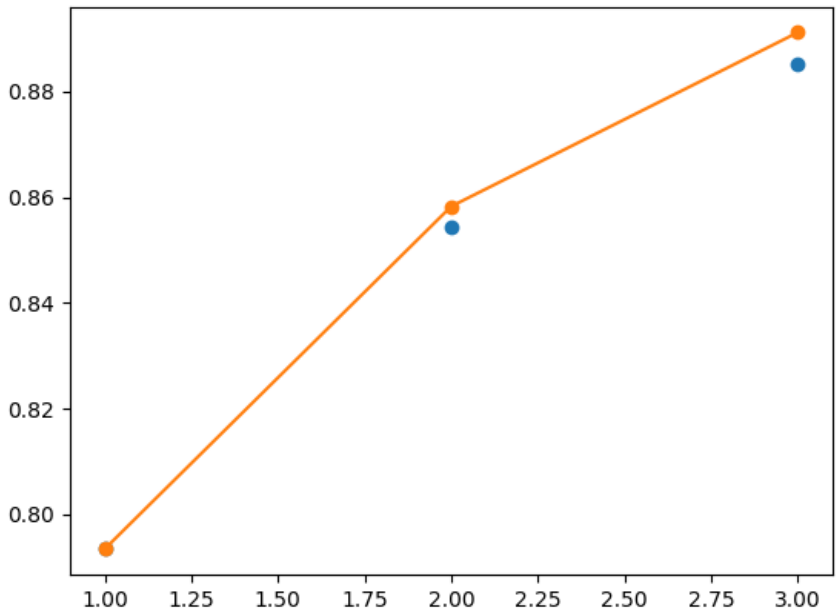

Figure 6: Ensemble method with the top 3 models

Our final model is shown in the figure 7, explaining each of our best models used in our ensemble.

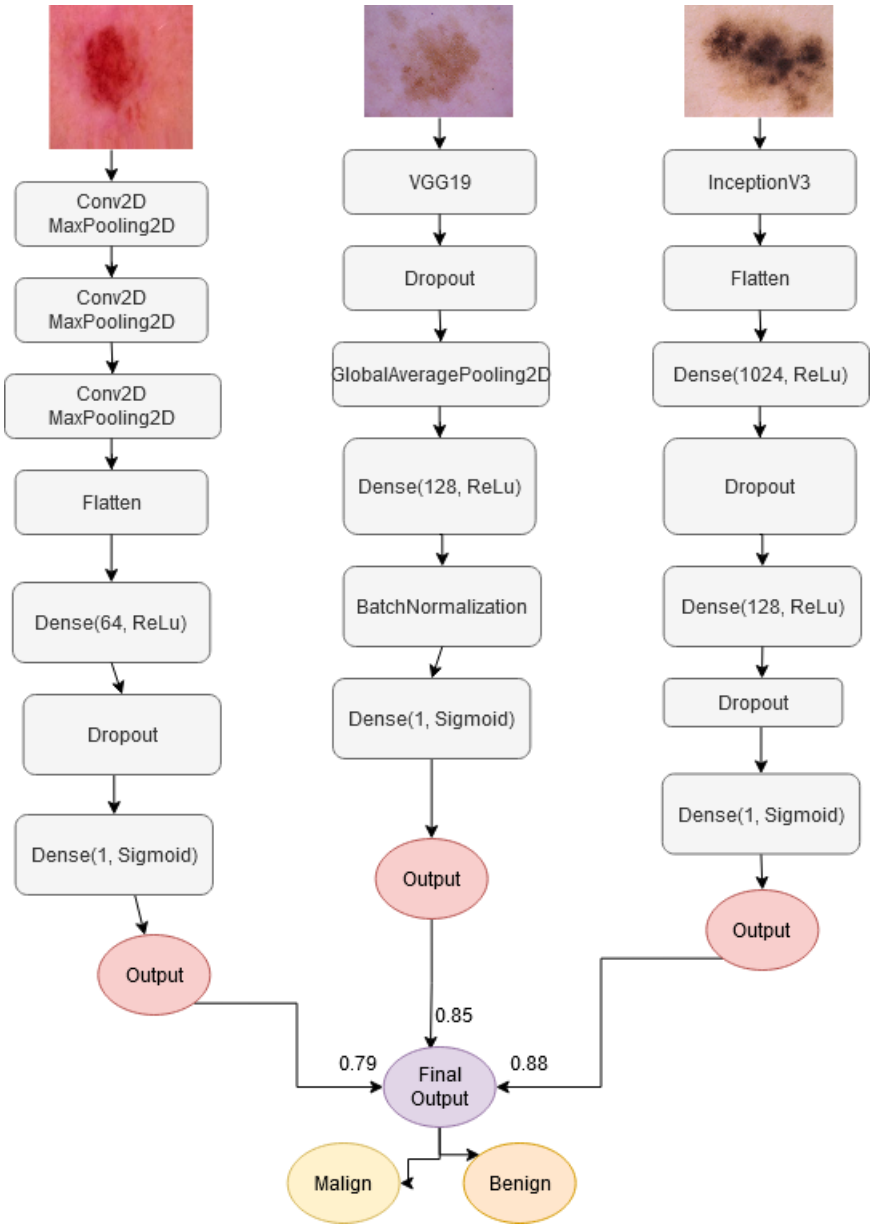

Figure 7: Ensemble architecture

4 Experimental Results

4.1 Dataset

The dataset we selected is ISIC's HAM10000 ("Human Against Machine with 10000 training images") that is composed of dermatoscopic images from different populations. The final dataset consists of 10015 dermatoscopic images which can serve as a training set for academic machine learning purposes. 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). A big challenge that rose was the fact that the data is very unbalanced, Out of the 7 classes, the melanocytic nevi makes up 67% of the data, while the melanoma and basal cell carcinoma that we wanted to target were only 11% respectively. At a closer inspection we can see that the dataset has several inherent sources of noise:

- Hair present in various densities
- Variation in ambient lighting
- Variation in skin lighting
- Variation in focal length (i.e. scale is unknown)
- Variation in image aspect ratio and resolution

Another problem that we predicted may arise is the fact that out two classes of interest mel and bcc are very different in aspect, the mel class being very easily mistaken for other benign classes by the naked eye, while bcc looks different than most of the other types.

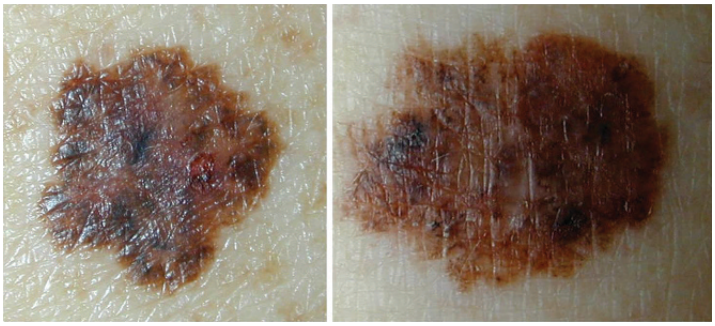

Figure 8: NV and Melanoma

Figure 9: BCC

4.2 Not Working Setup

As described in the paper Wen et al. [2016], **center loss** has shown remarkable results when paired with softmax as the supervision signal. The center loss works by simultaneously learning a center for deep features of each class and penalizes the distances between the deep features and their corresponding class centers. This approach steered our interests and we decided upon trying it. However, we quickly discovered a setback. In our project we decided to only output if the image fed to the model is either benign or malign instead of classifying the image to the numerous types of skin lesions. This being said, we were using as a supervision signal the **sigmoid** activation as it worked best for our binary classification. Upon implementing the center loss and trying to train the model using it, the losses were not being computed properly, jumping large values and reaching infinity quickly. By conducting other tests and doing more research, we reached the conclusion that in order for the center loss to properly achieve discriminative feature learning, it is necessary to combine the softmax loss with the center loss for jointly supervising the model. However, given our problem at hand, the binary classification of malignity and benignity, we couldn't add softmax to our model for proper results so we decided to drop the center loss altogether.

5 Evaluation Metrics

5.1 Confusion Matrix

The currently used model shows a **99.78%** accuracy in training. Using only classification accuracy alone, limits our understanding of the model quality. Thus, a confusion matrix is used further. The confusion matrix (figure 10) shows the ways in which your classification model is confused when it makes predictions. It gives you insight not only into the errors being made by your classifier but more importantly the types of errors that are being made. From our splitting of the database, we obtained 1002 test batches and we computed our confusion matrix over them. We obtain the following results, when the picked threshold is 0.5.

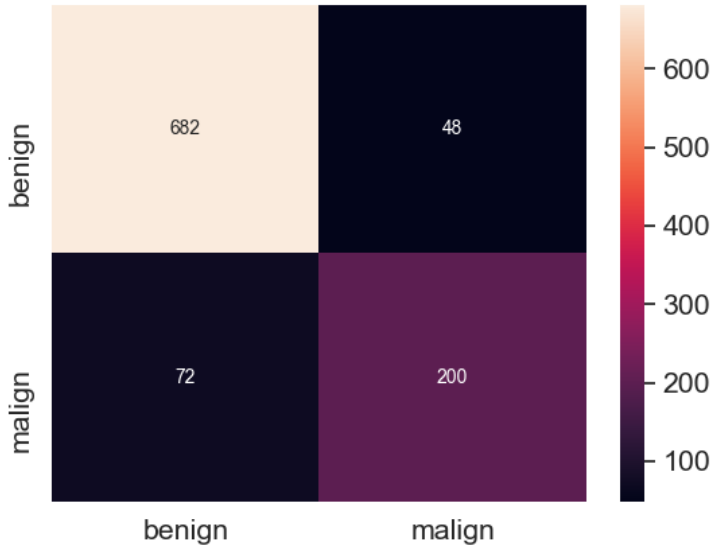

Figure 10: Confusion Matrix, threshold: 0.5

From our confusion matrix, we can deduct a large amount of information:

- Looking at columns, we can see the total amount of test images present for each category, benign and malign. In total, we have 730 benign test cases and 272 malign cases.

- On the first diagonal, we can see the correctly predicted cases: 682 benign and 200 malign correctly predicted images, summing up to 882 correct predictions.

- Looking as rows, we see the way the model labeled our data. The first row concerns the benign label. Out of 730 test images labeled as benign, there have been 682 correctly labeled and 48 were mislabeled as malign. On the second row, we see that the model labeled 200 malign cases correctly and misslabeled 72 of them as benign.

Our project refers to a special case of classification, namely a two-class classification. Since we are interested in whether a presented case is either benign ("true" case) or malign ("false" case), we can summarize the confusion matrix as following:

- True Positives **TPs**: predicted malign and actually malign

- False Positives **FPs**: predicted benign and actually malign
- False Negatives **FNs**: predicted malign and actually benign
- True Negatives **TNs**: predicted benign and actually benign

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

By looking at the above formula, we can calculate the accuracy of our model in the following way:

$$Accuracy = \frac{200 + 682}{200 + 48 + 72 + 682} = \frac{882}{1002} = 88.0239520958083\%$$

5.2 Precision and Recall

By looking at the confusion matrix in figure 10, we can also obtain two relevant evaluation metrics, namely: **precision** and **recall**. To fully evaluate the effectiveness of a model, you must examine both precision and recall. Unfortunately, precision and recall are often in tension. That is, improving precision typically reduces recall and vice versa.

Precision attempts to answer the following question: *what proportion of positive identifications was actually correct?* and is defined as:

$$Precision = \frac{TP}{TP + FP}$$

Recall attempts to answer the following question: *what proportion of actual positives was identified correctly?* and is defined as:

$$Recall = \frac{TP}{TP + FN}$$

For our test cases, we obtain the following, concerning the precision and recall for the malign label:

$$Precision = \frac{200}{200 + 48} = \frac{200}{248} = 80.64253\%$$
$$Recall = \frac{200}{200 + 78} = \frac{204}{278} = 73.52941\%$$

By increasing classification threshold, the number of false positives decreases, but false negatives increase. As a result, precision increases, while recall decreases. By decreasing classification threshold our false positives increase, and false negatives decrease. As a result, this time, precision decreases and recall increases. In the cancer prediction models such as ours, **recall is more important**, because misplaced false negatives (malign cases not identified correctly) result in serious health issues for the subjected individual.

5.3 ROC - Receiver Operating Characteristics

The ROC curve measures the performance of the the classification model using the TPR (True Positive Rate, also called "Recall") and FPR (False Positive Rate). Smaller values on the x-axis indicate lower false positives and higher true negative, and larger values on the y-axis of the plot indicate higher true positives and lower false negatives.

Therefore, a precise model will generate the ROC curve tending to the top left of the figure. If the curve would be identical to the diagonal line in the figure, it means the model has no precision, being completely random, with probability of success 0.5 (as it is a binary classification).

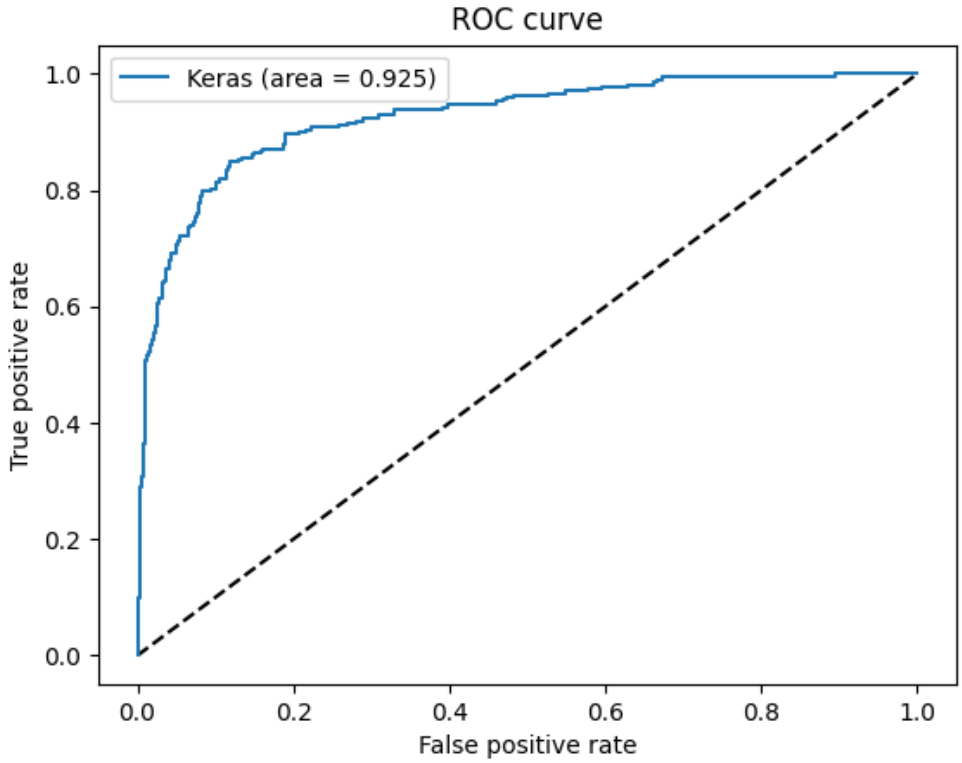

Figure 11: ROC curve

.

5.4 Activation Maps

An activation map, or feature map, is the output activations for a given filter. It can give us more insight into the descision making process of the CNN by visualizing how its neurons are activated by the input. We can take a look at how the network activates when trying to predict the malignity of the mole from Figure 10. We can see that this particular input has some noise too, consisting of the hair. The red color represents a highly activated area of the image, while the blue color represents low activation.

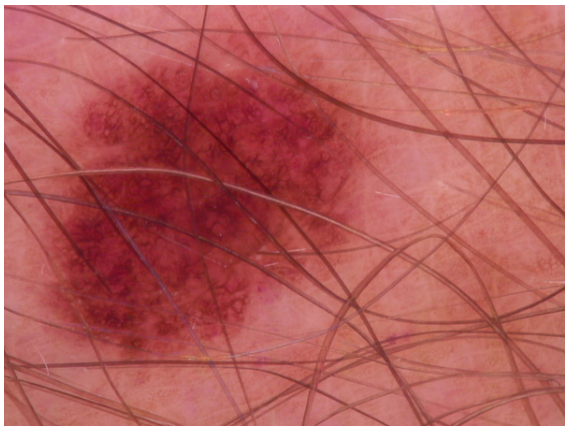

Figure 12: Input image

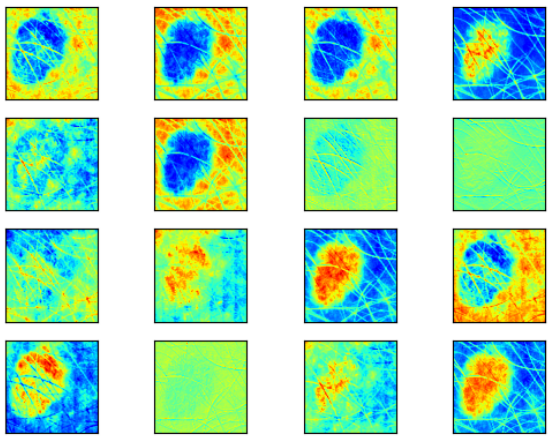

Figure 13: Activations for the 1st layer

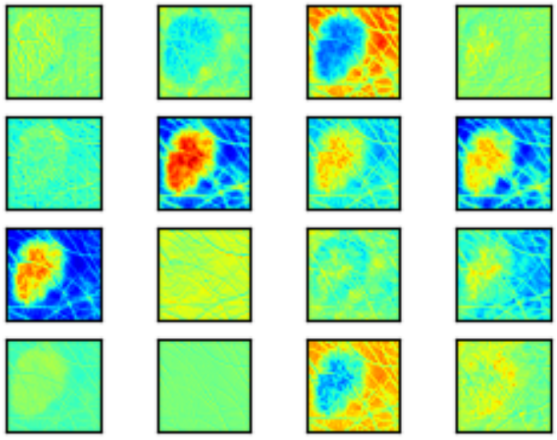

Figure 14: Activations for the 4th layer

One feature of the activations is that they get more abstract as we progress deeper into the network, the first layers processing finer details. This is a good thing, because the network tends to make its own abstractions in order to generalize the model.

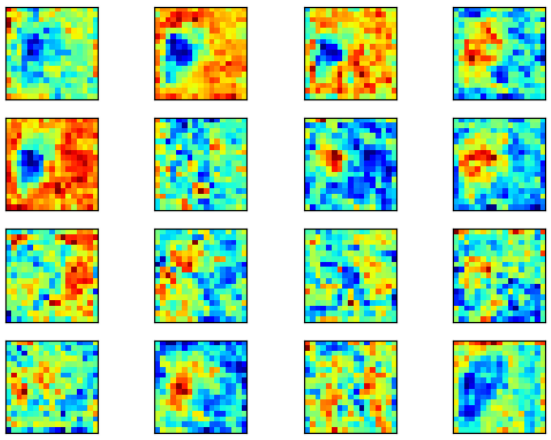

Figure 15: Activations for the 101th layer

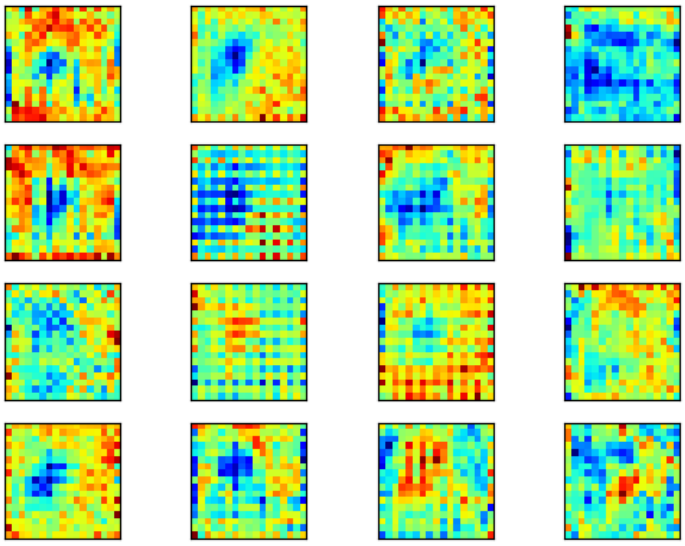

Figure 16: Activations for the 200th layer

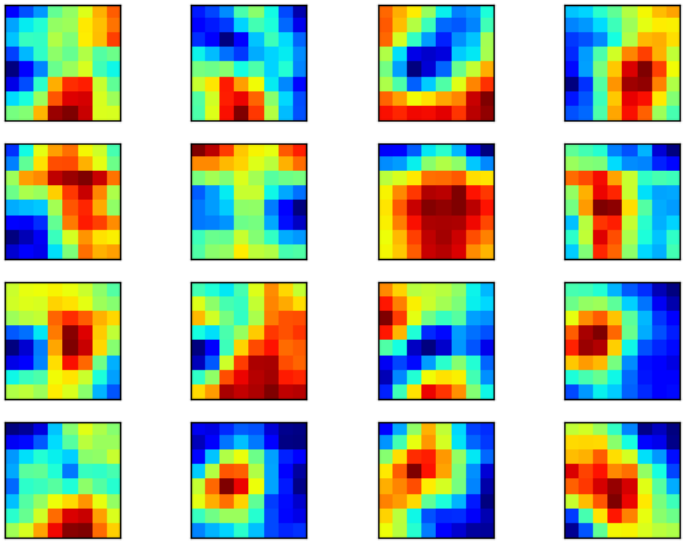

Figure 17: Activations for the 299th layer

5.5 Wrongly classified images

The last thing we took a look at were some examples of malignant signs that the model predicted wrong. From figure **??** we can see that the wrongly classified examples resemble benign lesions, specifically nv (the usual skin lession), so understandably the model has a tough time deciding their malignity.

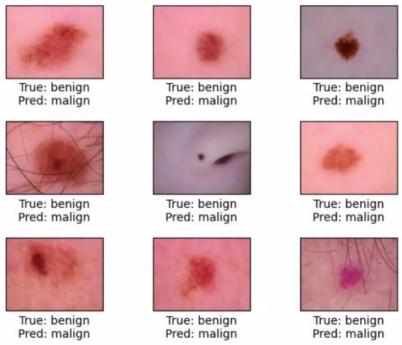

Figure 18: Wrong examples predicted by our model

5.6 Comparison

We decided to use the Romero López et al. [2017] approach as the comparison term since their approach and ours have ours have similarities in terms of the used methods. The differences are in the model's architectures and in the fact that we combine the results through ensembleing. Our results, as saw it the previous sections, are significantly better than theirs on the test set, their best sensitivity being 0.78, ours reaching 0.82, accuracy for them 81.3%, for us 89% and finally precision 0.79 for them and 0.81 for us. Our intuition is that the difference in results is coming probably from the fact that the data-set they are using is considerably smaller than ours.

6 Conclusions

Our starting point was the classification of images that present skin lesions into a diagnosis: **malign** or **benign**. We used numerous approaches for tackling this problem in order to get a final accuracy as high as possible considering the fact that diagnosing correctly a skin lesion is of a considerable relevance and might make a big difference into treating it. By using the Keras Api with techniques of transfer learning and random grid search, we managed to reach the accuracy of over 99% in training and over 88% in testing.

7 Future work

- Tackle the still present over-fitting: noise cancelling of the data-set(?), increase the data-set, train more layers of the pre-trained models(?)
- Try more approaches: detecting the malignant features of a sign, classify the images of the data-set into all the 7 different skin lesions.

8 Authors Contributions

- Andrei Bunu, 932: ensambling, over-fitting;
- Tudor Cocaină, 932: data pre-processing, transfer learning VGG19, grid search for model;
- Georgiana Loba, 934: transfer learning InceptionV3, optimizers, center loss;
- Cristian Cătălin Pop, 936: evaluation metrics, over-fitting;

We have numerous experiments and attempts of models, optimizers, evaluations and so on that did not end up being used or in the documentation, yet helped us learn and reach a well-performing model at the same time! :)

References

K. Caleb. Why everyone uses transfer learning. *Towards Data Science*, 2020.

A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017. doi: 10.1038/nature21056. URL `https://app.dimensions.ai/details/publication/pub.1074217286`.

P. Marcelino. Transfer learning from pre-trained models. *Towards Data Science*, 2018.

A. Nivrito, M. Wahed, R. Bin, et al. *Comparative analysis between Inception-v3 and other learning systems using facial expressions detection.* PhD thesis, BRAC University, 2016.

A. Romero López, X. Giró-i Nieto, J. Burdick, and O. Marques. Skin lesion classification from dermoscopic images using deep learning techniques. 01 2017. doi: 10.2316/P.2017.852-053.

Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016.