Using Neural Networks to Discriminate between Benign and Malignant Moles

Michael Birkholz michael.birkholz@estudiantat.upc.edu
David Dueñas Gaviria david.duenas.gaviria@estudiantat.upc.edu
David Fernández Aldana david.fernandez.aldana@estudiantat.upc.edu
Ronald Rivera Torres ronald.rivera@estudiantat.upc.edu

Abstract

Due to recent developments in machine learning, particularly with regards to advancements in the field of convolutional neural networks (CNNs), making a fast and accurate recommendation about whether a lesion is potentially cancerous using deep learning technology may be close to becoming a reality. Our goal with this project was to utilize some of the popular techniques in use today in machine learning, namely transfer learning and data augmentation, in order to explore the possibility of developing a skin cancer classification system given a limited dataset. In addition to applying transfer learning to some well-known networks like Inception-v4, a full-featured CNN, and MobileNetV2, a rather lightweight one appropriate for mobile deployment, we have designed a rudimentary CNN from scratch. As with many things, more is not always better, and we have seen the best results come from the simplest network. However, since the literature describes many successes in applying general features learned from a large non-medical corpus to a medical application using transfer learning, further investigation into leveraging transfer learning to bootstrap a complex network is justified. The results indicate that the technology to provide a reliable system to recommend whether a lesion merits further inspection and/or a biopsy, and even to run it on a device as small as a cell phone, may be just around the corner. Finally, we propose some avenues of further research that may be undertaken, including incorporating more domain knowledge from the dermatological field, in order to progress this system to the next level and get one step closer to skin cancer diagnosis in the palm of your hand.

Keywords: Neural Networks, Convolutional Neural Networks, CNN, Residual Networks, ResNet, Inception-v4, MobileNetV2, Data augmentation, Skin cancer, Mole, Benign, Malignant, Deep Learning

1 Problem Statement and Goals

According to the Skin Cancer Foundation[1], skin cancer is the most common cancer worldwide, with 1 in 5 Americans suffering from it by the time they reach age 70. It is also an incredibly deadly disease, killing more than two Americans per hour on average. However, if it is detected early, the 5-year survival rate for melanoma, one of the deadliest skin cancers, is 99 percent. This final statistic is a major motivation for undertaking this project, which studies the feasibility of applying machine learning methods to classify skin lesions as cancerous or benign. The hope is that having an accessible, convenient, reliable, inexpensive method capable of making an initial assessment of skin lesions may significantly improve the prognosis for many skin cancer patients since the cancers could be detected much earlier in many cases.

We have obtained a labeled dataset from a Kaggle project compiled by Claudio Fanconi[1] where the original source data comes from the ISIC-Archive[2]. It contains 1800 pictures of benign moles and 1497 pictures of malignant ones. All pictures are 224x224 pixels. As this is a fairly small dataset, we have explored the use of data augmentation to synthesize additional input images from the given data.

The primary Computational Intelligence (CI) technique we have applied is using neural networks as classifiers. Since this is an image processing problem, convolutional neural networks (CNNs) are particularly well-suited to the job. For this reason, we have chosen a number of different architectures to compare, including a custom network built from scratch, Inception-v4 and MobileNetV2. Additionally, in order to overcome the lack of input data to train a complex network from scratch, we have also explored transfer learning using pre-trained networks and adding additional layers to distinguish between images of malignant or benign lesions as well as data augmentation.

2 Previous Work

Machine learning approaches have been applied to skin cancer classification problems for quite some time, but in 2016 there was a revolution. Convolutional neural networks (CNNs) came onto the scene and essentially replaced all other machine learning methods in every single entry in the International Symposium on Biomedical Imaging[3], which marked the start of a new era in machine-learning applied to medical diagnostic tasks with a visual component. Convolutional neural networks are particularly well-suited to image analysis because the convolution operation provides a convenient way to extract features in a translation-invariant manner. And, as an added benefit, it is actually practical for use due to the computational advantages provided by the sparse connectivity between layers as contrasted with a fully-connected multi-layer perceptron (MLP).

Our approach to using transfer learning, initially trained on a large dataset (ImageNet) containing general non-medical images and then subsequently refined based on images containing examples of malignant or benign marks on the skin, is supported by the literature. Specifically, in Menegola et al's 2017 paper[4], this particular approach is analyzed and they reach the conclusion that it is in fact better to use a network trained on generic images instead of one fine-tuned even for another medical application, although they leave the door open to future research in that regard. Their supposition is that the feature extractors obtained from a generic set of images are superior to the specific ones derived based on retinopathy, a very tangentially-related medical use case, in their study.

A paper by Zhang et al, Skin Cancer Diagnosis Based on Optimized Convolutional Neural Network[5], is especially interesting from a computational intelligence perspective, due to the comparison of many techniques taken from nature. The core of the paper is an algorithm used to optimize a neural network based on whale hunting techniques. This technique itself is not novel, but the the team proposed a modification to it by using the Lévy flight mechanism to avoid premature convergence and thereby provide more optimal results in identifying skin cancers. They compare their optimization algorithm against a number of other ones including a genetic algorithm, shark smell optimization, World Cup optimization, grasshopper optimization and particle swarm optimization and it achieves significantly better results on many benchmarks. Unfortunately, the paper does not provide an actual number to quantify how much better their final results are at classifying malignant vs. benign than other referenced algorithms, but instead the reader must estimate the improvement from a graph.

One of the common threads seen throughout most papers we have looked at is that in most cases the researchers have access to much larger datasets. Our dataset is on the order of 3000 images, while the research teams behind many of the papers have access to many times that amount. For example, Esteva's team[6] had access to a database with nearly 130,000 images.

Another outcome and/or goal of many of the papers reviewed was to be able to properly segment the lesion and separate it from the surrounding skin. In some cases the shape of the segmentation was used as an input to the classification algorithm[7]. In other cases, it appeared to simply be a byproduct of using the CNN[5]. In fact, on the Kaggle website from which the dataset was obtained, there is an open task to create an unsupervised model that can learn to segment moles from the surrounding skin, but no submissions have been attempted for that task to date.

3 The CI Methods

CI methods are computational techniques whose inspiration is found in natural phenomena. Of the major branches of CI, we have primarily focused on neural computation for this project. An assortment of the CI methods we have used are outlined in this section.

3.1 Activation Functions

Activation functions are intended to mimic neuronal activation in living cells and allow neurons to fire more or less strongly. We have bisected our network into two sets of layers, where a different activation function was chosen for each set. In the intermediate layers, ReLU was used since modern research shows that it generally has favorable performance, particularly towards training, than either the Sigmoid or tanh functions, despite being more dissimilar to biological neural activation[8]. The second set of layers, or rather just one layer, is the output layer, where we have selected the Sigmoid function. The cancer classification problem disallows outputs from being both positive and negative at the same time. In other words, the output classes are not fuzzy, but rather mutually exclusive. The scope of this work has been limited to a two-class classification problem instead of a multi-class problem, like, for instance, an image of a lung that may show both cancer and pneumonia at the same time. Usually, the Softmax activation function is used in the classification layer for classification problems because it distributes the probability per class across each output node. The Sigmoid function is the two-class case of the Softmax activation function and is an even more efficient choice than two Softmax nodes because in a crisp binary classification problem each input must be assigned a class. Therefore, we can simply test if an input likely belongs to the class "Benign," or if not, assign it to class "Malignant," or vice-versa. This saves an extra node, connections and calculations in the final layer.

3.2 Data Augmentation

The purpose of applying data augmentation is to increase the model's ability to generalize. In an ideal world, there is enough training data to represent every possible variation in input data. In this case, that variation may come about as a result of intrinsic characteristics of the subjects represented in the input images, such as skin tone, lesion pigmentation, hair thickness, presence/absence of hair, etc. Or, it may be the result of lack of control over the conditions under which each image is taken: lighting conditions, camera resolutions, focus, rotation, centeredness, zoom, etc. Since the size of the dataset was fairly limited, particularly for training models of higher complexity, we decided to employ data augmentation techniques to manufacture some additional training images to create some additional variation in the training data in the hope of improving the model's ability to generalize.

Data augmentation encompasses a wide range of techniques used to generate training samples taken from the original set of images by applying various random perturbations without changing the class labels of the images. Some examples of techniques we used were rotation/flipping, shear warping, zooming and adjusting the brightness.

As part of this exercise is to investigate the effect of applying data augmentation and evaluating whether it is even helpful, we will first present the results of our network trained on the original data with no augmentation. Then, we will compare with the results of using augmented training data. Note that the final evaluation of the network is always performed on unmodified test data and no data augmentation techniques have been applied to any of the test data.

We have used Keras's built-in module for image data augmentation: `ImageDataGenerator`. It is a very convenient approach to quickly generate new training data to alleviate problems with overfitting due to a lack of input data, but the downside of using this generator is that it does not provide as full control over the augmentation process as a homegrown approach might. Therefore, if the parameters

are not combined carefully, it might lead to an unrealistic generation of images which will only add noise to the model.

The idea of generating augmented data also helps to balance the training data equally among the classes represented in the dataset. Essentially, the purpose of generating these modified versions of training images is, first, to make sure that the model gets fed with sufficient data for proper generalization and, second, for robustness in terms of becoming invariant to small changes like rotation or camera angles when applied in a real world scenario.

3.3 Transfer Learning

In some scenarios, it can be difficult and expensive to obtain training data that fits the feature space and the expected characteristics of the test data. This is how the need to create a high-performance learner for the target domain comes about[9]. This learner consists of, in essence, taking advantage of a large amount of information related to solving a specific problem and using it on a different one, but the problems must have some characteristics in common. In other words, this approach attempts to modify or add on to patterns already learned (such as trained neural networks) to be able to recognize similar ones. This is the motivation for transfer learning, which is used to improve learners by transferring information from related fields. The networks used were previously trained on the ImageNet dataset, which bring some general image recognition features to the table, and we have subsequently applied transfer learning to them in order to make them specialized in recognizing benign vs. malignant moles.

3.4 Convolutional Neural Networks

Neural networks are clearly patterned after biological structures and were in fact born from the fields of psychology and neurophysiology rather than computer science. While they are much simpler than biological neurons, in large groups and when exposed to enough training data, behavior emerges which mirrors that of organic neurons. The CNNs are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. The inspiration for Convolutional Neural Networks was somewhat indirectly found in nature as well. They are based on the neocognitron[10] which was intended to mimic the behavior of cells in the visual cortex of cats and monkeys.

3.4.1 Convolutional Neural Network from Scratch

Our team created a convolutional neural network to attempt to solve the binary classification problem of benign vs. malignant when presented with an image of a lesion. We use the terminology "from scratch" to signify that only the basic building blocks available in the 'Keras Sequential API' were used instead of relying on an existing deep learning architecture. As this was a custom design, importing existing weights and training via transfer learning was not an option for this network and trial and error was required. Despite not being a tried-and-true model, the network described below actually performed reasonably well.

The input layer of the network matches the shape of the images, $(224, 224, 3)$, for 224×224 pixel images with 3 color channels. Immediately following, there is a series of convolution layers. The first convolution layer with a set of 32 learnable filters is set up to use a kernel size of 3×3, stride 1. The common ReLU was chosen as the activation function for all intermediate layers since current research indicates that it provides favorable characteristics to facilitate training. This same set of convolution plus activation plus max pooling, with all of the same parameters is repeated once more. The third and last convolution layer was set up to learn 64 filters instead of 32. Each of the 2×2 max pooling layers is intended to reduce sensitivity to local variations and reduce the volume of data that needs to be fed forward in the network in favor of reducing computational cost. Next, the 3-D tensor output from the convolution layers is flattened to 1-D. At this point, we found empirically that adding a 64-neuron

dense layer plus a 0.5 dropout layer provided good results in helping to reduce overfitting. To further improve this architecture, and, in fact, the other pre-trained architectures as well, a set of optional configurations were incorporated into the model immediately before the final layers. For example, in Table 1, there is an optional dropout layer included. Another pattern that was tested was including a fully-connected layer before a final dropout layer. Finally, the last two layers include a fully-connected layer that aggregates all of the outputs of the previous layers and feeds the Sigmoid activation function ultimately used to classify the input image as benign or malignant.

Table 1: Summary of CNN from Scratch Model with one Optional Layer

Layer (type)	**Output Shape**	**Param #**
conv2d (Conv2D)	(None, 222, 222, 32)	896
activation (Activation)	(None, 222, 222, 32)	0
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 32)	9248
activation_1 (Activation)	(None, 109, 109, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 32)	0
conv2d_2 (Conv2D)	(None, 52, 52, 64)	18496
activation_2 (Activation)	(None, 52, 52, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)	0
flatten (Flatten)	(None, 43264)	0
dense (Dense)	(None, 64)	2768960
dropout (Dropout)	(None, 64)	0
Optional Layer: dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 1)	65

3.4.2 MobileNetV2

The first pre-trained network that will be tested is MobileNetV2, a type of residual network (ResNet). It has promoted the development of lightweight models by significantly reducing the number of operations and amount of memory required while maintaining the same accuracy as the previously state-of-the-art MobileNetV1[11]. The architecture of MobileNetV2 contains the initial full convolution layer with 32 filters, followed by 19 residual bottleneck layers, uses ReLU as the non-linear activation and a kernel size 3×3 and utilizes dropout and batch normalization during training. This module takes a low-dimensional compressed representation as input which is first expanded to a higher dimensionality and filtered with a lightweight depth-wise convolution. Overall, this network uses memory very efficiently when it makes inferences, which means that it is particularly suitable for mobile applications.

3.4.3 InceptionV4

Inception v4, developed in 2016, is an evolution of GoogLeNet (Inception v1) and Inception v3. This new version is a simplified architecture and contains more inception modules than the third version of Inception. Before proceeding with the explanation of the fourth version of Inception we first need to explain the earlier iterations of this model since it builds on top of them. The first version of

Inception proposed using a wider neural network instead of a deeper one. The idea was to run multiple filter sizes at the same level and then combine the results. The first version comprised nine of these inception modules stacked linearly and 22 layers in total. In the third version the main focus was to simplify the inception model further. The research team noticed that the auxiliary classifiers hardly contributed until the end of the process, when the accuracies were almost saturated. In addition, to improve computational speed they factorized the 7x7 convolution to two 5x5 convolution that themselves would in turn be factorized to two 3x3 convolutions. Lastly, reduction blocks were also introduced to change the height and width of the grids of the model. As with previous improvements, the goal of these changes was even better model performance.

Our implementation, deemed "BinaryInceptionV4", makes use of InceptionV4 (based on [12] updated to work in Tensorflow 2.4) without the top layers, adding a Flatten layer, an optional 128-unit dense layer and dropout layer in the middle (depending on each particular experiment) and finally a single dense layer with Sigmoid activation to perform the binary classification. All the original layers were frozen except for the last 22 layers so that our network could retrain those high level weights to work better with our cancer image dataset.

4 Results and Discussion

4.1 General Methodology

4.1.1 Data Preprocessing

The pixel values in each image were normalized between 0 and 1. Also, the folds were created and saved to disk in advance to allow for better performance and complete determinism of the cross-validation procedure as explained in the next section.

4.1.2 Validation

Stratified 5-fold validation was used to validate each model performance. The number of folds was chosen to be a compromise between the runtime cost that would be incurred by using more folds and the training-validation splits, since:

$$split_{validation} = \frac{1}{|folds|} = \frac{1}{5} = .2$$

and thus:

$$split_{training} = 1 - split_{validation} = 1 - .2 = .8$$

Fewer folds would result in a less-representative result because they use less training data (generalization issue) and runs (more variance) while a higher number would not only take a considerable amount of time to train but also, because of the lower validation split, would end up being less representative. This stratified approach was used since there is a small class imbalance problem in our dataset, where about 55% are 'benign' and about 45% are 'malignant.' The folding approach was chosen to mitigate the fact that we do not have many samples to work with.

4.1.3 Performance Metrics

With respect to the performance scores used to evaluate and improve the models[13], in this binary classification problem the metrics considered were:

- **F1 Score**: This is the harmonic mean value of "Precision" and "Recall", which can better measure misclassification cases than the classic "Accuracy" metric.

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- **Precision**: This is a measure of correctly identified positive cases from all predicted positive cases.

$$Precision = \frac{True\,Positives}{True\,Positives + False\,Positives}$$

- **Recall**: This is a measure of correctly identified positive cases from all real positive cases. This is important when the cost of false negatives is high, which is the case here.

$$Recall = \frac{True\,Positives}{True\,Positives + False\,Negatives}$$

- **Accuracy**: One of the more obvious indicators is the measurement of all correctly identified cases. This is most commonly used when all classes are equally important, while the F1-score is used when both False Positives and False Negatives are important.

$$Accuracy = \frac{True\,Positives + True\,Negatives}{True\,Positives + True\,Negatives + False\,Positives + False\,Negatives}$$

4.1.4 Static Parameters

Every experiment in the report (with or without augmentation) has the following configuration:

- **Maximum number of epochs**: 50. This number is chosen as a compromise between the runtime duration and classification performance.
- **Early stopping**: Monitoring validation loss with a patience of 10 epochs restoring the best epoch weights. This is used both to improve runtime and classification performance.
- **Adaptive Learning Rate**: Monitoring validation loss with a patience of 5 epochs with a factor of .5 and a minimum LR of 10^{-7}. This is used to complement some old algorithms like SGD that do not have this capability.
- **Loss function**: Binary cross-entropy. This is used since we are doing a binary classification in conjunction with a sigmoid last layer activation.
- **Classification layer**: 1-unit dense layer with sigmoid activation.

4.2 Neural Networks without Data Augmentation

The results from the three neural network architectures trained without data augmentation can be found in the following sections. The results of five-fold cross-validation are presented first. These results were used to select the best additional layers to add after the core layers of each network and the best optimization parameters. Since the primary goal of this exercise is to compare network architectures and data augmentation techniques, we have limited the number and range of varied parameters that we present here to a set of five different configurations.

Table 2: **Results for Scratch Model with no augmentation**

Dropout	Dense Layer units	Optimizer	Epochs	Learning Rate	Accuracy	Recall	Precision	F1 score
.7	0	Adam	50	10^{-5}	.7788	.8570	.7126	**.7778**
0	128	Adam	50	10^{-4}	.8092	.8319	.7712	**.7986**
0	0	SGD	50	5×10^{-3}	.7603	.7810	.7178	**.7466**
0	128	RMSprop	50	10^{-4}	.8221	.8562	.7770	**.8121**
.1	128	Adam	50	10^{-5}	.8024	.8712	.7418	**.8001**

4.2.1 From Scratch with No Augmentation

The network developed from scratch with no transfer learning achieved accuracy and F1 scores from approximately 0.75 to 0.80, as can be observed in Table 2.

As expected, the newer algorithms designed to improve upon SGD have effectively outperformed it. To be fair, there may be some ambiguity about whether the relatively lower performance is a result of the optimization algorithm or the lack of an extra fully-connected layer as compared with the other networks. But, the overall result is not surprising since both algorithms were devised to extend and/or improve upon standard SGD. The Recall statistic is also a bit low for actual practical use, despite being the strongest metric of the model. The cost of missing a potentially-cancerous lesion is simply too high, so some improvement on that metric would be desirable.

4.2.2 MobileNetV2 with No Augmentation

Among the MobileNetV2 models, a slight improvement was achieved with the first pre-trained network, coming from the lack of the Dropout Regularization technique in the case of the second model. It achieved an F1 score of 0.7225, after the same number of epochs as almost all the other models, which can be seen in Table 3.

Table 3: **Results for MobileNetV2 model with no augmentation**

Dropout	Dense Layer units	Optimizer	Epochs	Learning Rate	Accuracy	Recall	Precision	F1 score
.7	0	Adam	50	10^{-5}	.7295	.5259	.8213	**.6365**
0	128	Adam	50	10^{-4}	.7803	.6405	.8405	**.7225**
0	0	SGD	50	5×10^{-3}	.7515	.5627	.8347	**.671**
0	128	RMSprop	50	10^{-4}	.7326	.5736	.7832	**.6608**
.1	128	Adam	48	10^{-5}	.7363	.6313	.7505	**.6853**

The aggressive Dropout used in the first model had the worst performance with an F1 score of 0.6365, although this is to be expected as the aggressive Dropout of 0.7, which was intended to prevent overfitting, seemed to hurt the Recall score significantly with 0.5259, although the Precision score was just slightly lower than the top result from this network. The RMSprop had a fair performance since it also uses adaptive methods for convergence. However, the SGD and the RMSprop methods were in fact outperformed by the Adam when Dropout was not applied aggressively. Omitting or using a very small Dropout rate and instead using a Dense layer of 128 with Adam seem to have an improvement in the performance of the best network which can be noticed with second and fourth models, achieving the highest F1 scores but with slightly lower recall of 0.6405 as compared to 0.6313.

4.2.3 InceptionV4 with No Augmentation

The best results, as measured by the F1 score, were obtained by the fourth combination that uses RMSprop with a standard Learning Rate (LR) without any extra dropout layers and with a 128-unit dense layer.

Table 4: **Results for InceptionV4 model with no augmentation**

Dropout	Dense Layer units	Optimizer	Epochs	Learning Rate	Accuracy	Recall	Precision	F1 score
.7	0	Adam	49	10^{-5}	.7769	.7300	.7681	**.7482**
0	128	Adam	50	10^{-4}	.8114	.7809	.7998	**.7900**
0	0	SGD	50	5×10^{-3}	.8027	.7726	.7900	**.7803**
0	128	RMSprop	50	10^{-4}	.8141	.7734	.8099	**.7908**
.1	128	Adam	50	10^{-5}	.8054	.7659	.7926	**.7812**

With a slightly lower F1-score we find the second combination which is the same as the best except for using the Adam optimizer, which scored better in recall but trailed in accuracy. As can be seen, adding dropout, especially in the first case with a 70% dropout rate, proved to be detrimental since underfitting occurred. In the middle of the scores, we find the third combination with Stochastic Gradient Descent (SGD) which performed poorly in the precision metric probably due to being compared to more modern optimizers as well as missing and additional dense layer.

4.3 Neural Networks Using Data Augmentation

We evaluated a number of combinations of data augmentation techniques. These combinations are enumerated in Table 5. To identify the best combination of data augmentation parameters we used

Table 5: **Augmentation parameters used per tested model**

Parameters	Model 1	Model 2	Model 3	Model 4	Model 5
rotation range	-	.1	.1	-	40
width shift range	-	-	-	-	.2
height shift range	-	-	-	-	.2
brightness range	(.9, 1)	-	(.9, 1.1)	-	(.9, 1)
horizontal flip	True	True	True	False	True
vertical flip	False	True	True	False	True
zoom range	-	-	30	.2	0.2
shear range	-	.1	225	-	0.2
fill mode	-	-	-	-	nearest

the NN that performed best without data augmentation. That is, we used the network developed from scratch with a 128-unit dense layer and RMSprop optimizer. We can see the results of these combinations in Table 6 with a 50-epoch run. Using the F1 score as our fitness metric, we observe that our performance is in the range of 75% to 78%.

The model with the best performance is the first one where only the brightness range and horizontal flip were adjusted/applied randomly. This was expected since the brightness conditions may improve or degrade the categorization of skin marks. On the other hand, the worst model, model 3, used a high zoom value and shear range. In this case, we assume that the zoom and shear caused the loss of relevant features in the images. It is worth mentioning that the difference between each model is small and increasing the number of epochs could provide different results in the ranking of the F1 scores.

Table 6: **Augmentation Parameter Results**

Param Model	Epochs	Accuracy	Precision	Recall	F1 score
1	50	.7879	.7396	.8235	**.7788**
2	50	.7944	.7688	.7868	**.7764**
3	50	.7705	.7357	.7783	**.7529**
4	50	.7739	.7419	.7758	**.7545**
5	50	.7686	.7055	.8420	**.7672**

4.3.1 From Scratch with Augmentation

Using data augmentation provided a modest improvement over simply using the original dataset. These results also show that either SGD is the least effective optimizer or that the extra dense layer provides an advantage, consistent with the non-augmented results. Most of the models have a slightly better F1 score compared to the same model using the original data. One of the most interesting parts is that the model that performed best with the non-augmented data and was used to select the most appropriate data augmentation parameters was also the model that performed worst with the augmented data. We surmise that this may be the case simply because training was set to stop after 50 epochs due to compute resource limitations, but the models may have still been in a somewhat noisy descent process when training was stopped. Additionally, since the from-scratch network was used to select the best augmentation parameters, this made us suspicious that the performance improvement expected from using augmented data might be biased towards the model used to select the augmentation parameters, but these results seem to refute that idea. Recall is notably higher, which is of particular importance as that indicates that it has managed to suppress false negatives, the most dangerous mistake in this classification problem.

Table 7: **Results for Scratch model with Augmentation**

Dropout	Dense Layer units	Optimizer	Epochs	Learning Rate	Accuracy	Recall	Precision	F1 score
.7	0	Adam	50	10^{-5}	.7800	.8955	.7026	**.7870**
0	128	Adam	50	10^{-4}	.8133	.9064	.7429	**.8155**
0	0	SGD	50	5×10^{-4}	.7553	.8629	.6839	**.7612**
0	128	RMSprop	50	10^{-4}	.7876	.8653	.7233	**.7857**
.1	128	Adam	50	10^{-5}	.8001	.8905	.7298	**.8013**

For simpler comparison, the best results from the no data augmentation and data augmentation have been reproduced in Table 8.

Table 8: **Comparison of augmentation vs. no augmentation**

Aug.	Dropout	Dense Layer units	Optimizer	Learning Rate	Accuracy	Recall	Precision	F1 score
No	0	128	RMSprop	10^{-4}	.8221	.8562	.7770	**.8121**
Yes	0	128	Adam	10^{-4}	.8133	.9064	.7429	**.8155**

4.3.2 MobileNetV2 with Augmentation

Once more, the use of augmentation seemed to provide a slight improvement for the pre-trained network, which applied fine-tuning techniques to improve the overall performance. The results can be found in Table 9.

Table 9: **Results for MobileNetV2 model with augmentation**

Dropout	Dense Layer units	Optimizer	Epochs	Learning Rate	Accuracy	Recall	Precision	F1 score
.7	0	Adam	48	10^{-5}	.7428	.6546	.741	**.6939**
0	128	Adam	50	10^{-4}	.8077	.7767	.7971	**.7864**
0	0	SGD	50	5×10^{-4}	.8054	.7684	.7991	**.7824**
0	128	RMSprop	50	10^{-4}	.8153	.7709	.8166	**.7912**
.1	128	Adam	50	10^{-5}	.8092	.7726	.8011	**.7854**

In this case, interestingly, with just a small delta from the other models, the fourth model which used the RMSprop achieved the highest F1 score of 0.7912 for the network. Also, using the 128 Dense layer with no Dropout was able to score the highest precision of 0.8166 although the recall was slightly lower than the other approaches using Adam. The worst model was the first, whose performance seemed to suffer due to the the aggressive Dropout.

For simpler comparison, the best results from the no data augmentation and data augmentation have been reproduced in Table 10.

Table 10: **Comparison of augmentation vs. no augmentation**

Aug.	Dropout	Dense Layer units	Optimizer	Learning Rate	Accuracy	Recall	Precision	F1 score
No	0	128	Adam	10^{-5}	.7803	.6405	.8405	**.7225**
Yes	0	128	RMSprop	10^{-4}	.8153	.7709	.8166	**.7912**

4.3.3 InceptionV4 with Augmentation

With InceptionV4, we also find an improvement using augmentation as can be seen in Table 11.

Table 11: **Results for InceptionV4 model with augmentation**

Dropout	Dense Layer units	Optimizer	Epochs	Learning Rate	Accuracy	Recall	Precision	F1 score
.7	0	Adam	50	10^{-5}	.8202	.7792	.8171	**.7973**
0	128	Adam	50	10^{-4}	.8084	.7600	.8098	**.7825**
0	0	SGD	50	5×10^{-4}	.7959	.7450	.7938	**.7683**
0	128	RMSprop	50	10^{-4}	.8209	.7751	.8219	**.7972**
.1	128	Adam	50	10^{-5}	.7940	.7442	.7911	**.7665**

Surprisingly, the results which were, by a large margin, the worst with no augmentation, with an F1 score of just 0.7482 became the best with augmentation with an F1 score of 0.7973, barely beating out the fourth combination which had an F1 score of 0.7972 (the first combination with no augmentation). This phenomenon can be explained by the fact that, since we are augmenting the images, we have more input images so more regularization is needed to avoid overfitting. We can also observe that the

other remaining three had worse performance, especially the last combination which had a high drop in recall and around a 0.1 drop in accuracy which could be attributed to the variability in initializing the weights of the network or the image augmentation itself.

For simpler comparison, the best results from the no data augmentation and data augmentation have been reproduced in Table 12.

Table 12: **Comparison of augmentation vs. no augmentation**

Aug.	Dropout	Dense Layer units	Optimizer	Learning Rate	Accuracy	Recall	Precision	F1 score
No	0	128	RMSprop	10^{-4}	.8141	.7734	.8099	**.7908**
Yes	0.7	128	Adam	10^{-5}	.8202	.7792	.8171	**.7973**

4.4 Test Results of Best Model Overall

After evaluating all of the networks with and without image augmentation, we scrutinized the cross-validation results for each model in order to select the best one. The best F1 score, 0.8155, was obtained by the from-scratch network with a configuration of 128-unit dense layer and the Adam optimizer as can be seen in Table 13.

Table 13: **Model comparison**

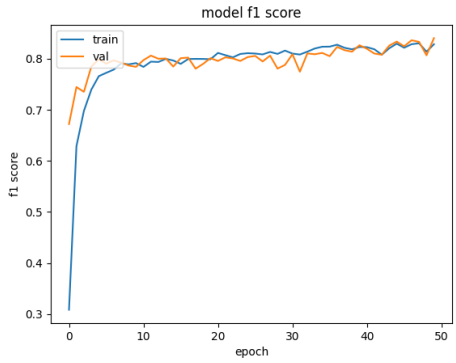
Models	Optimizer	Epochs	Accuracy	Recall	Precision	F1 score
Scratch	Adam	50	.8133	.9064	.7429	**.8155**
MobileNetV2	RMSprop	50	.8153	.7790	.8166	**.7912**
InceptionV4	Adam	50	.8202	.7792	.8171	**.7973**

To evaluate the networks we use the augmentation parameters that performed better in the train section; brightness range (0.9, 1) and horizontal flip (True). Now, training this network using all of the training data plus augmentation, after 50 epochs we achieved the following results on the holdout test set of 660 images: an accuracy of 0.8439, recall of 0.9033, precision of 0.7855 and a F1 score of 0.8403. In Figure 1 we can observe the additional results obtained running this network with augmentation. We see that in the first iteration we quickly achieve a F1 score above 0.75 and it continues to increase steadily as we run more iterations. On the loss we have similar behavior where it continues to decrease as the epochs increase. It appears that the model may be able to improve further according to the loss and F1 score charts, which do not indicate either a plateau or stagnation within the first 50 epochs. According to the confusion matrix after training for 50 epochs, we have correctly classified 79.4% benign and 90.3% of the malignant skin growths.

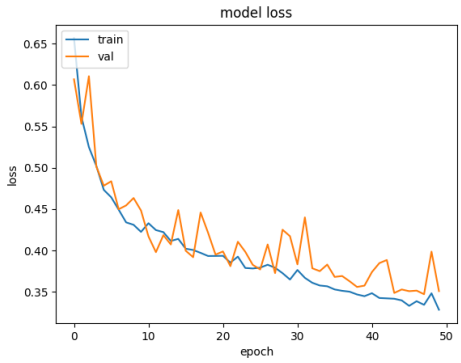
5 Strengths and Weaknesses

Our system developed from scratch performed very well even compared to other more well-known architectures. Due to its small size, it had a much smaller memory footprint and was trainable much faster than larger networks like InceptionV4. It achieved above a 90% recall rate on the test data, which is important since that is indicative of a low rate of false negatives, the parameter we want to minimize above all others. We can also observe that the validation loss does not follow the expected decreasing curve but, while it has a negative trend there are a lot spikes where it grows and shrinks which might be indicative that we are using higher than needed Learning Rate.

F1 Score



Model Loss



Model Confusion Matrix

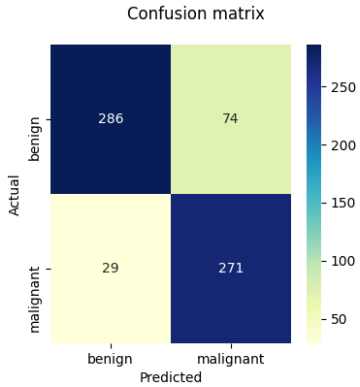


Figure 1: Results for Best Model Overall

We made great efforts to separate training, validation and test data in order not to corrupt any evaluation results. A stratified 5-fold cross-validation was performed in order to maximize the amount of data we could train our model with while also getting an accurate picture of its performance and knowing when to stop training to avoid overfitting. This was all done with great care to reserve all of the test data only for the final model evaluation. Only cross-validation results were used during the model selection process in order to avoid selecting a model that is biased towards the test data.

Our initial attempts at image augmentation with a home-grown script had mixed results. The transformations (various filters, gamma, translations, rotations) applied to the images were too heavy-handed and it was found that the CNNs were overly sensitive to the changes performed. Luckily, a there was already an appropriate and robust function built into Keras that performed state-of-the-art augmentation with either subtle or drastic changes depending on the parameters used, which overall allowed the Network to learn the underlying patterns in the data.

Our team worked effectively in parallel to determine the best set of model parameters and hyper-parameters given the limited computing resources at our disposal vs. the length of time each set of training/cross-validation took. Our primary tool was Google Colaboratory (commonly know as Co-lab), but the team frequently surpassed the computing limits of that system. Additionally, we took advantage of a cluster with some GPU resources but this system took significantly longer to execute tests than Colab. Given our limited time and computing resources, the decision was made to prioritize comparison of a small set of varied model parameters and hyperparameters rather than performing a semi-exhaustive search of the entire parameter space for our models. For this reason, we have chosen five unique sets of optional layers, optimizers and learning rates to apply to our models and compare results. This provides us a scientific benchmark to know how each model performs against each other one under the same conditions, but at the same time it is nearly guaranteed not to have found the global optimum for any of the models. If the true goal of this research was to find the absolute best performance of each model, more variation and a more exhaustive search would need to be done, or even a random search of an appropriate size[14].

A particular weakness of our system was not including domain knowledge. As none of the team is a dermatologist nor has any dermatological training, we were limited to general machine-learning techniques. One of the papers that had a very impressive accuracy in identification of benign vs. malignant moles of above 96%[7] adapted some techniques used by dermatologists to decide whether a lesion merits a further look. In particular, they used the ABCD rules (A - asymmetry, B - border, C - color and D - diameter) in order to extract features to then be used as inputs to an artificial neural

network. There are certainly many more examples of other such details that could be incorporated and any serious follow-up work that seeks to advance the current state-of-the-art should consult with experts in the field in order to ensure that applicable domain knowledge is applied if appropriate.

6 Conclusions

We have explored the application of Neural Networks in the medical field, focusing on the classification of cancerous and non-cancerous skin conditions using image processing. The data used to carry out and evaluate this classification task consisted of 3,297 images, 1,800 benign and 1,497 malignant. This dataset was divided into training and test sets along an 80% and 20% split, respectively. In addition, we evaluated if applying data augmentation techniques to the training data increased performance for the same neural network models. Image augmentation was conducted primarily by normalizing the images and modifying image properties (rotation, brightness, zoom, etc.). The networks tested were a traditional CNN (scratch), MobileNetV2 (from the ResNet family) and InceptionV4. To evaluate the performance of each network we use the F1 score metric after 50 epochs.

After testing the previously-mentioned neural networks with no augmentation, in the first section, we obtained that the traditional convolutional network with a configuration of 128 dense layer and and RMSprop achieve the best performance with 0.8121 F1 score. The Scratch network outperformed the InceptionV4 and MobileNetV2, the last one having the worst performance in almost all configurations with the lowest F1 score of 0.6365. Using the best-performing network with no augmentation data, we tested different combinations of parameters to identify the best augmentation parameters. The identified parameters were brightness range (0.9, 1) and horizontal flip, with a F1 score of 0.7788. We evaluated each of our three networks with this combination of parameters and observed an improvement in each of the networks. The best performance was achieved in the from-scratch network with a F1 score of 0.8155 using 128 dense layer unit and an Adam optimizer configuration. Once all of the above process was concluded we selected the top performing model to evaluate it against the test data. Using the from-scratch CNN model with the augmented data we obtained a 0.8439 F1 score with 79.4% correctly classified benign lesions and and 90.3% correctly classified malignant growths after a maximum of 50 epochs.

As expected, data augmentation provided improved results by allowing the network to generalize better and recognize unimportant features as noise. When evaluating the models, the best performing model was on both occasion the traditional CNN (scratch) network. This result was unexpected by the team but it could be explained by the more complex architectures tending to overfit the small dataset. With this evaluation we have proven the potential of neural network models in medical applications, considering the fact that dermatologists are rarely able to surpass the 80% threshold in skin cancer diagnosis from visual inspection alone[3]. The results we have obtained are very encouraging for the widespread adoption of CNNs as an aid to improve skin cancer detection rates in the future. With the help of a dermatoscopic camera, dermatologists are able to improve their accuracy a bit further and may reach approximately 84%, a number our models have approximated. However, our models still have too high of a false negative rate to truly be comparable to a trained dermatologist. The next section includes some ideas on how that might be addressed.

6.1 Future Work

Naturally, since our goal is to combine the use of a CNN with ubiquitous technology in order to provide medical advice to patients from the comfort of their own home, a compromise was made to train a lighter weight Convolutional Neural Network, MobileNetV2, in order to evaluate the feasibility of developing a system like this for use on mobile devices. It was trained with the same augmentation techniques and the best parameters found above and was compiled into a simple android .apk. Its performance was

not the best given the limitations of mobile platforms, but it achieved reasonable prediction results on images taken from the test dataset. Ideally, this technology would be integrated into an online diagnostic tool or a mobile app that would provide usable, fast and accurate guidance about whether the patient should seek professional advice regarding a suspicious lesion. The demo installer for android devices along with additional information can be found at: `https://skin-moles.web.app`.

The idea of using additional inspection techniques and even other information leads to some interesting ideas for follow-on work. If a dataset could be created with the images, or possibly even videos, from dermatoscopic cameras, it may provide significantly more information to train neural networks and improve their results even further past the state-of-the-art today. There may also be valuable information about skin lesions outside the visible spectrum. The inspiration for this idea comes from a study about insect vision[15] where pictures of flowers taken without a UV filter are displayed and oftentimes have strong differentiating features in the UV spectrum that are invisible to the naked eye. Gathering a dataset of images of cancerous vs. non-cancerous lesions' responses to many wavelengths outside of the range of visible light may lead to the discovery of features unobservable by the human eye. Another missing component in the kaggle dataset, and in fact many of the datasets used in other studies, is metadata providing more context for each image, such as the age, race, location on body, medical history of patient/relatives, etc., which are all pieces of information a dermatologist has access to when making a recommendation. It is unclear how exactly this extra metadata could be injected directly into a convolutional neural network, but perhaps there could be a hybrid architecture that treats part of the input as an image-processing task and then combines the output of the image processing with the metadata as inputs to a classification system.

One final very important consideration is that skin cancer is not a single entity. It comes in many forms, including melanoma and various types of carcinomas. We treated this problem as a simple binary classification problem when in reality a multi-class approach may be warranted. Presumably, each different type of cancer would present somewhat different features and if they can be distinguished visually, this may be a more appropriate way to formulate the problem than simply treating it as a binary classification. Even if the final output to the app user could ultimately be presented as benign or malignant, behind the scenes it may be advantageous to take advantage of features presented by specific types of cancer.

A Running the Code

In order to run the code and make some sample predictions with the trained model, you first need to clone the project's repository:

```
git clone https://github.com/mbirkholzupc/mai_ci_ddmr
```

Once you have cloned it, you may simply run the following notebook (inside the repository):
`https://github.com/mbirkholzupc/mai_ci_ddmr/blob/main/scratch_model_prediction.ipynb`

A requirements.txt file is provided in the repository to aid in python (virtual) environment setup if necessary. As this is just a prediction task and not training the model, a normal CPU runtime should be sufficient and there are no extraordinary memory requirements.

References

[1] Skin Cancer Foundation. *Skin Cancer Facts & Statistics: What You Need to Know*. URL: https://www.skincancer.org/skin-cancer-information/skin-cancer-facts/. (accessed: 30.12.2020).

[2] International Skin Imaging Collaboration. *International Skin Imaging Collaboration*. URL: https://www.isic-archive.com/. (accessed: 30.12.2020).

[3] Titus Josef Brinker et al. "Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review". In: *J Med Internet Res* 20.10 (Oct. 2018), e11936. ISSN: 1438-8871. DOI: 10.2196/11936. URL: http://www.ncbi.nlm.nih.gov/pubmed/30333097.

[4] Afonso Menegola et al. "Knowledge Transfer for Melanoma Screening with Deep Learning". In: *CoRR* abs/1703.07479 (2017). arXiv: 1703.07479. URL: http://arxiv.org/abs/1703.07479.

[5] Ni Zhang et al. "Skin Cancer Diagnosis Based on Optimized Convolutional Neural Network". In: *Artificial Intelligence in Medicine* 102 (Nov. 2019), p. 101756. DOI: 10.1016/j.artmed.2019.101756.

[6] Andre Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542 (Jan. 2017). DOI: 10.1038/nature21056.

[7] T. Kanimozhi and Dr. A. Murthi. "COMPUTER AIDED MELANOMA SKIN CANCER DETECTION USING ARTIFICIAL NEURAL NETWORK CLASSIFIER". In: 2016.

[8] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[9] Taghi M. Khoshgoftaar Karl Weiss and DingDing Wang. "A survey of transfer learning". In: *Journal of Big Data* 3.1 (2016), p. 1717. DOI: 10.1186/s40537-016-0043-6.

[10] K. Fukushima. "Neocognitron". In: *Scholarpedia* 2.1 (2007). revision #91558, p. 1717. DOI: 10.4249/scholarpedia.1717.

[11] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

[12] Kent Sommer. *keras-inceptionV4*. URL: https://github.com/kentsommer/keras-inceptionV4. (accessed: 30.12.2020).

[13] Taghi M. Khoshgoftaar Karl Weiss and DingDing Wang. "Beyond Accuracy, F-score and ROC: a Family of Discriminant Measures for Performance Evaluation". In: *Advances in Artificial Intelligence, Lecture Notes in Computer Science* 4304 (2006). DOI: 10.1007/11941439_114.

[14] A. Zheng. *Evaluating Machine Learning Models: A Beginner's Guide to Key Concepts and Pitfalls*. O'Reilly Media, 2015. ISBN: 9781491932469. URL: https://books.google.es/books?id=OFhauwEACAAJ.

[15] James Lincoln and Andrew Davidhazy. "Ultraviolet photography and insect vision". In: *The Physics Teacher* 57 (Mar. 2019), pp. 204–205. DOI: 10.1119/1.5092494.