A Computational Approach to Pertinent Feature Extraction for Diagnosis of Melanom

Carlos
Sergio Vizcaino
Adriana

Table of Contents

1. Introduction

In today's day, Melanoma is one of the most common forms of skin cancer. It is classified to be a severe type of skin cancer due to its ability to spread throughout the body (3). Therefore it is essential to detect this type of skin cancer based on its features such as geometry, color, texture and the structure of skin lesions (3). In general, from a computer scientist's point of view the key steps in analyzing these skin lesions would be to use preprocessing, segmentation, feature extraction and classification (3). All these steps are essential as it will help contribute to a comprehensive understanding of the lesion, aiding in the timely and accurate identification of potential malignancies (3). Early intervention based on such diagnostic assessments plays a pivotal role in effective medical management and improved outcomes for individuals at risk (3). Figure 1 shows the sample input images of three different types of lesions. Where (a) represents a common lesion, (b) shows a suspicious lesion that may be classified as cancerous or non-cancerous and lastly (c) shows a melanoma lesion (3).

The goal of this project is to obtain the highest possible accuracy and recall value as the researchers have obtained in the paper. This project goes into details about pre-processing images and then feature engineering is used respectively to properly isolate the region of interest. Afterwards, different machine learning algorithms and techniques namely multilayer perceptron, 1D/2D conventional neural networks, batch normalization, regularization, drop out and cross validation are used to enrich the accuracy and recall values as much as possible.

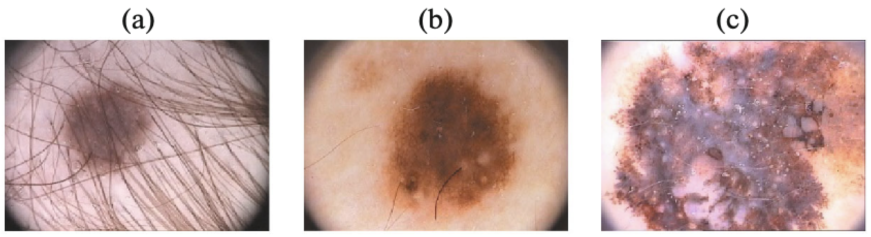


Figure 1. Sample input images: (a) common lesion, (b) suspicious lesion and (c) melanoma lesion (3)

1.1 Dataset

In the research paper, the researchers have used the PH2 database, a dataset sourced from the University of Porto (3). In this project the PH2 database is also utilized as the goal is to attempt to replicate the paper's methodology in solving the problem. The dataset is composed of 200 dermoscopic images and its corresponding binary images of the lesion, categorized into 80 non-melanoma, 80 atypical melanoma and 40 melanoma images. There is a challenge in categorizing the atypical melanoma images as the image may fall into either non-melanoma or melanoma classification. To ease implementation, the atypical melanoma and melanoma images are classified together under the melanoma classification. In the end, the data was divided into two categories, comprising 80 images classified as non-melanoma and 120 images categorized as melanoma. The original paper does not mention at all any of the features presented in the PH2 dataset txt or csv files, only the name and the label (Clinical Diagnosis) of the images are used. In order to export the image names into a CSV file named "mod_PH2_dataset" and labeling the image with binary numbers. In other words, an image classified as non-melanoma will be assigned the value 0, while an image categorized as melanoma will be assigned the value 1.

1.2 Project's Structure

Our project is divided into several folders. The *FeatureBuilders* folder contains all the builders that are used for feature extraction. Each file has 3 attributes: Feature's name, image type I will receive as input (Normal=Color or Lesion=Binary) and lastly the boolean ready attribute, False if we do not want to include that feature, True if we do. The builders will be called whenever a model needs to be trained.

Next folder is the *PreProcessing,* which contains 3 files with several preprocessing steps. They will be called in our core file, *read_images* where the class *ImageLoader* is initialized. *ImageLoader* has several useful functions such *display_image, get_one_pixel* or *get_all_pixels*. But its most important one is *read_bmps* where the .bmp files (images) are loaded and are readed into *np.arrays.* It has the parameter *target_size*, by default is 137x199 pixels, this resizes all the images to that size.
Note that : this is only because for our most complex 2D CNN, please use *target_size=(766, 575)* for MLP and 1D CNN since these models use features not the images, they need all the data available for the feature extraction.

One last folder called *results* contains the models' performances. The rest of the files are model implementations that use all the previous files we just talked about.

1.3 Pre-processing Images

Hair is a normal aspect of human skin. Many of the images from the database contained hair on the area of study which brings the importance of pre-processing images before further studies. This is because hair can lead to noisy results and misclassification of images.

In order to overcome this problem, hair removing algorithms were used. The researchers in the paper have suggested using the Dull Razor Algorithm and Median Filter to overcome this problem (3). Unfortunately the implementation of Dull Razor Algorithm was not compatible with the code implemented for this project. As an alternative, the Morphological Black Hat Transformation algorithm was used to remove the hair and the Media Filter was then used to remove any remaining noise (3,10). The Morphological Black Hat Transformation algorithm was inspired by a researcher in Github (10). Something to note is removing hair can possibly remove other meaningful information for the research like loss of color, texture, etc which are important for the classification.

2. Feature Extraction

One of the most important aspects of this project is feature engineering. Although for models such as the two dimensional Convolutional NN these features are not needed. For other models, MLP and one dimensional CNN, they are crucial. In the majority of runs of our models we use 12 features: two asymmetry indexes, ratio between area and perimeter, six binary color indicators, compactness index and two diameters formulas. We have found that this set performs the best and boosts the recall of the models up to 5% when compared to other sets. All the files where the features are extracted are inside the *FeatureBuilders* folder. Each one has an individual file that will be imported and called whenever is necessary.

But first, it is important to know the data structure we have chosen to load the images. We decided to store them in *np.arrays* that contained the rows of the image.
Example of a dermoscopic color image:

```
array([[[ 48,  51,  35],[ 58,  59,  46], [ 60,  59,  49], ...,
            [ 64,  60,  51], [ 60,  56,  48], [ 58,  53,  46]],
      [[ 56,  54,  49], [ 62,  60,  56], [ 62,  58,  57],  ...,
           [ 62,  55,  46],  [ 62,  56,  46], [ 64,  60,  51]],....
```

Each list of RGB values is a pixel that is contained in a *np.array* that represents a row.

Example of a lesion binary image:

```
array([[False, False, True, ..., False, False, False],
      [False, False, True, ..., True, False, False], …,
      [False, False, True, ..., True, True, False],
      [False, False, True, ..., False, False, False]])
```

Same structure but now each pixel is not a RGB list is just a boolean.

Note that all these features have to be extracted using the full image, that is 575x766 pixels. This means that whenever running the MLP and 1D CNN, must ensure that *read_images.py* has the full pixel image as target_size().

Also, it is important that after the extraction of the features, normalization is applied to all of them (binary color features also, but they do not suffer any changes because binary is immune to normalization). This is done with the *sklearn* library using the *MinMaxScaler,* which subtracts the minimum and divides by the range. $(X - X_{min})/(X_{max} - X_{min})$. Normalization helps a lot to the models since converts all the features to 0-1 range. *StandardScaler()* was also tried, but in our case, models did not benefit from having gaussian data as much as normalized data.

2.1 Area and Perimeter

Starting with the simplest feature, area is computed by the is determined by zeroth-order moment M_{00} of the lesion. Which really is only counting all the white pixels of the binary image. This is computed with the *np.sum(image)* function that, since the images are *np.arrays* of *np.arrays* we can easily count all the True pixels with this built-in function.

Our first approach of the perimeter's computation was a double nested *for* loop that iterated through all the images and checked all the True pixels' neighborhoods, if the True pixel had at least one False neighborhood, it counted as perimeter. This approach was computationally expensive and inefficient as well as unstable with some errors. We discarded and opted for the following implementation.

If a reduction of complexity was needed our first idea was to look for built-in numpy functions. *np.diff()* performs subtractions of all the rows (axis = 1) and columns (axis = 0). With these subtractions only the True pixels that had False neighborhoods will remain as 1 (True-True=0, True-False=1, False-True=-1, False-False=0). Obtaining these values using both rows and columns axis, now we could count them and obtain the perimeter value. Secondly, the True pixels that were at the borders of the image (1st and last row and column) also had to be counted as perimeter. They are counted right after the *np.diff()* computation.

2.2 Area to Perimeter Ratio and Compactness Index

However, neither Area and Perimeter are included as features. Instead, they are used for multiple formulas for more complex functions. Starting with the simplest ones, Area to

Perimeter Ratio is just $Area/Perimeter$. This ratio captures valuable information of a lesion, since melanomas tend to have much greater ratios than benignant lesions.

An alternative Builder for the Ratio is in *CAreaPerimeterRatioBuilder.py,* this idea uses *masks* data structures to compute perimeter and give similar results. However, we decided to not use this builder because it is more complex than the previous one.

Another feature extracted from Area and Perimeter is Compactness Index whose formula is $4\pi A/P^2$ Compactness is a measure of closeness of the pixels in the shape to the center of the shape. The most compact figure is the circle with a compactness of 1. Malignant lesions tend to deviate from 1 and close to 0, since they are irregular and uneven shapes.

2.3 Colors

The presence of colors is one of our most vital features. First, we considered 6 colors: White, Red, Light Brown, Dark Brown, Blue-gray and Black. And using the following thresholds proposed in the original paper we determined the color of each pixel.

Color	**Red (R) | Green (G) | Blue (B)**
White	| $R \geq 0.8$ | $G \geq 0.8$ | $B \geq 0.8$ |
Red	| $R \geq 0.588$ | $0 \leq G < 0.2$ | $0 \leq B < 0.2$ |
Light brown	| $0.588 \leq R \leq 0.94$ | $0.196 < G \leq 0.588$ | $0 < B < 0.392$|
Dark brown	| $0.243 < R < 0.56$ | $0 \leq G < 0.392$ | $0 < B < 0.392$ |
Blue-gray	| $0 \leq R \leq 0.588$ | $0.392 \leq G \leq 0.588$ | $0.490 \leq B \leq 0.588$ |
Black	| $R \leq 0.243$ | $G \leq 0.243$ | $B \leq 0.243$ |

After counting all the pixels of each color, if there were more than 5% of pixels of that color we consider that color as present (present = 1, not present = 0). We tried with 1%, 10% and 15%, neither of them gave greater results than 5%. The original paper summed all of the binary variables to obtain a color score that ranges from 0 to 6. However, our models performed better with 6 binary features than only the numerical score. That is why we decide to keep the binaries and discard the score

2.4 Asymmetry Indexes

Asymmetry Index 1 (Major Axis) was calculated by aligning the lesion's major axis with the x-axis of the coordinate system. After rotating the lesion image, it was flipped vertically (upside-down). The non-overlapping region between the original and flipped images was computed, and the asymmetry index was determined as the ratio of the area of this non-overlapping region to the total area of the lesion.

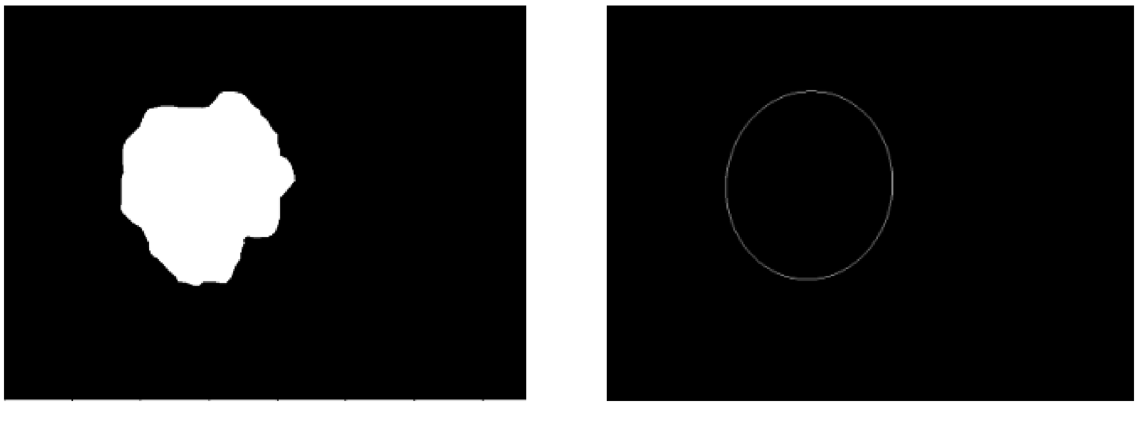
Asymmetry Index 2 (Minor Axis) followed a similar approach but focused on the minor axis. After aligning the lesion's major axis with the x-axis and rotating the image, the lesion was flipped horizontally (left to right) instead of vertically. The asymmetry index was then calculated based on the non-overlapping region between the rotated and flipped images, similar to Index 1.

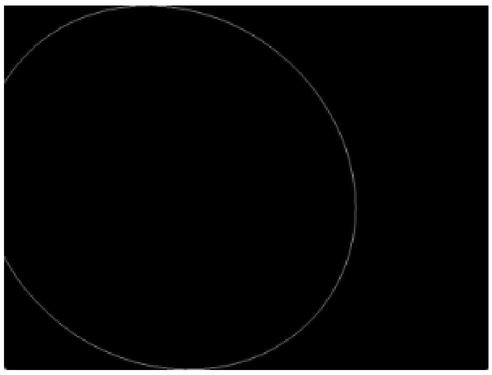
The asymmetry index (AS1) across the major axis defines the ratio of the areas of ΔBx and B. (The non overlapping region (ΔBx) between segmented image (B) and flipped image across major axis (Bx).) $AS1 = \Delta Bx/Bx$. Similarly, AS2 with the minor axis, $AS2 = \Delta By/By$. Note that: $\Delta Bx = B \oplus Bx$ and $\Delta By = B \oplus By$.

2.5 Best Fit Ellipse, Diameters (D1 and D2).

Another essential feature is the best fit ellipse of the lesion binary image. Thanks to the well known image preprocessing library *OpenCV cv2* this can be done quite easily. We use the findContours function to identify contours in the binary image. Contours are continuous curves that form the boundary of an object. The function returns several pieces of information but we are only interested in the contours. The *RETR_EXTERNAL* mode retrieves only the external contours, while *CHAIN_APPROX_SIMPLE* compresses the information, saving memory.

Then, we only are interested in the largest contour and thanks to the *fitEllipse* function we compute the best fit ellipse for the selected contour. Now we have access to the center coordinates of the ellipse, its axis and its angle. Example of several applications:





Now we are able to compute the Diameters features proposed in the original paper.
Average Diameter of the lesion, D1, is computed using the following formula:
$D1 = (d1 + d2)/2$ where $d1 = \sqrt{4A/\pi}$ and $d2 = (2a + 2b)/2$.
Note that: major (2a) and minor (2b) axes lengths of its best-fit ellipse.
D2, Difference between Principal Axes Lengths. $D2 = 2(a - b)$. Melanoma are characterized for having greater diameter values than benignant lesions.

3. Machine Learning Models and Techniques

In this research, three different machine learning models were used for the study. These include Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN) to understand the distinct output results for each model and the varying impacts each model has on the final results. Furthermore,

3.1 Multilayer Perceptron

In machine learning, Multilayer Perceptron (MLP) is a type of artificial neural network (NN) architecture where the information is transmitted in one direction (1). In other words, this architecture is also called feed forward (1). As shown in figure 2 below, the NN is composed of three components, these include an input layer, hidden layer and output layer (1). The input layer takes the input data (1). For the case of this project the image features were taken as inputs and were then sorted into an array. The hidden layer is the intermediate layer between the input and output layer (1). The hidden layer is responsible for applying weights to the input data and using activation functions to then introduce non linearity to the model (1). Having hidden layers is important as it enables the network to learn patterns from the input data (1). Lastly, the output layer is responsible for producing the network's output (1). The amount of nodes in the output layer depends on the problems being solved. For the

case of this study, the output layer is composed of one node labeled as “binary decision” as a binary classification to classify melanoma vs non melanoma.
In this project, MLP has extracted features from the images. The MLP model was composed of one hidden layer with 100 neurons. The features extracted are obtained as an array using NumPy, a Python library. Each value was then calculated to represent a specific value. For instance, the perimeter feature, NumPy will compute and return a real number for this feature. This applies to all the other characteristics. Every image in the dataset is processed to generate a list of features and these lists of features then serve as the input data for further analysis. In our case 12 features are used, although the number of features can be adjusted at any time.

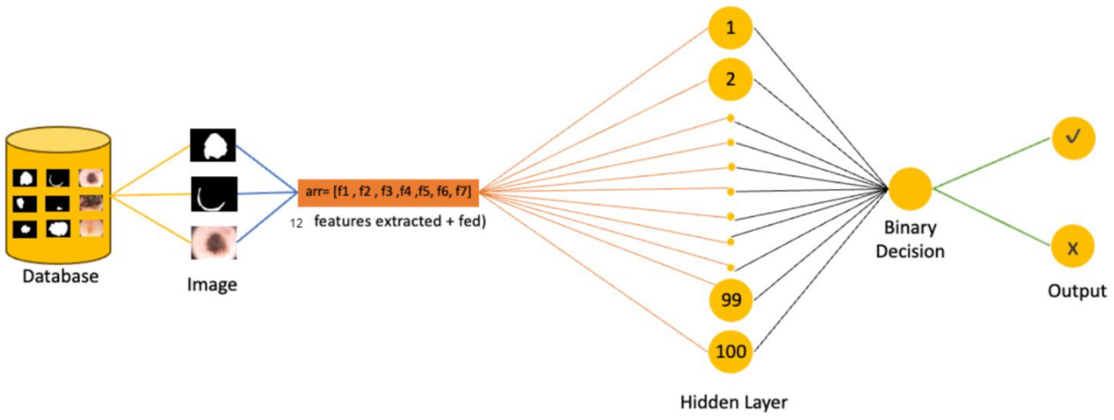

Database
Image
arr= [f1 , f2 , f3 ,f4 ,f5, f6, f7]
12 features extracted + fed)
1
2
99
100
Hidden Layer
Binary Decision
✓
X
Output

Figure 2. Overview of the Multilayer Perceptron Neural Network Architecture

3.2 Two Dimension CNN

In the second CNN model, it is composed of three filter layers where two of the layers are convolution layers, one is the dense layer. In the implementation for the project, layer 1 contains 16 filters, layer 2 contains 32 filters and layer 3 is composed of 64 filters. This implementation was chosen to progressively capture more complex features from the input images. This is because in the earlier layer, the CNN can learn from those low level features. There is also a fourth layer which is a fully connected layer. Where the convolution layer is responsible for applying a filter or kernel over the input through operations (5). The fully connected layer on the other hand is a dense layer where each node in the NN is fully connected to every other node in the previous and subsequent layers which allow the NN to learn a variety of patterns and relationships in the inputs (5).
The second CNN model was implemented in a 2D format in order to aim to improve the overall performance and achieve superior results. As CNN models are becoming more

complex it is important to ensure computation time and results remain adequate enough. Thus batch normalization, regularization and drop out techniques were used to help maintain computation time and enrich the results.

3.3 One Dimension CNN

Another model was the conversion of the previous CNN to one dimension. This will simplify the model and reduce the complexity. This implies that the input will no longer be the raw images, but the 12 features as the MLP.

Going from 2D to 1D implies that pooling is not possible anymore and the kernel size has to be downgraded to 1 dimension. Sizes of 2, 3, 4 and 6 were tried, being 4 the one with better results. The rest of the structure remained intact except for the *Conv2D* function that was replaced by its minor *Conv1D.* Also since this model is less computationally expensive, we could afford a 10-fold CV.

4. Machine Learning Techniques

In addition to the three different machine learning models used, several different machine learning techniques are implemented to help learn patterns, make predictions and extract meaningful information in order to make informed decisions. The different machine learning techniques introduced for this project are batch normalization, regularization, dropout and cross validation which will be discussed in detail in the following paragraphs.

4.1 Batch Normalization

Batch normalization is a known technique in machine learning and deep learning to help improve the training for neural networks (4). Batch normalization is created to deal with covariate shift which in other words sees the change in how the NN hidden layers are spread out while in the learning phase (4). Thus, batch normalization is implemented in this project as it helps keep track of this change which makes the training process more stable and effective in projects (4).
During the project, the inputs to each layer may change as the parameters are constantly being updated which may lead to a slow down in the training process (4). Thus, using batch normalization the inputs of a layer are adjusted and scaled during the training process (4). It was noticed that after implementing batch normalization, the training was accelerated significantly. For instance, initially the project research was focused on a fixed image size of 100x100. Using batch normalization, it does not operate on this fixed size as it has been

observed that altering the image size influences the results. Instead, it adapts to the specific image dimensions of the batch, such as 199x137.

4.2 Regularization

Another commonly used machine learning technique is regularization. It is a technique used to prevent overfitting to avoid the model becoming too complex and performing low on new data (6). The goal is to have good generalization where the model performs well on the current features training set and also on any new features. In short, regularization has a penalty term in the model which discourages complex patterns and moderates the risk of overfitting (6).

In this project, it was seen that employing the regularization techniques has improved results. The process has involved experimenting with various configurations which has included the number of filters in each layer and different dropout values.

Initially the project had dropout values of 30, 30 and 50 for layers 1, 2 and 3 respectively. The image size for the maximum target is 766x575, however unfortunately this is not sufficient as it would significantly delay the training time. Therefore, when the image size was reduced to 199x137 the values were optimized respectively resulting in a more refined setting of 0.05, 0.05 and 0.035 for the three layers. These lower dropout values were proved to be more effective and reduce unnecessary connections in the neurons (9). These regularization adjustments significantly contributed to achieving an accuracy rate of 0.82.

4.3 Dropout

In machine learning, “dropout” is part of the regularization method where it is used during training to prevent overfitting (6,7). As the name states, the dropout technique works by randomly dropping out and ignoring some nodes in each layer during each iteration of the training (7). This is a useful technique to implement in the project as it helps with the generalization of the model from relying on certain nodes or overfitting the training data (7).

Applying the dropout technique into this project helps to temporarily deactivate some random connections during the training. In the CNN model, early stopping was implemented. Early stopping works by monitoring the performance during the training and then stopping the training when specific criteria is met (8). In other words, early stopping will stop the training when the model outputs the same results or shows minimal changes after several epoch iterations (8).

4.4 5-Fold Cross Validation

In both MLP and CNN models, the cross validation technique was used. Cross validation is a useful technique to evaluate and learn algorithms by dividing the data into segments (11). One segment used to train the model and the other segment to validate the model (11).

In this project, the dataset is balanced where there are 80 images classified as non-melanoma and 120 images classified as melanoma. The cross validation was applied by splitting all the data into 5 folds. In the next step, the training is done with one batch and then the other segment is used to validate the results. After training all the epochs and measuring the loss, accuracy and recall, the process is restarted again from the beginning however the training will not be done using the same batch instead the training is going to be done with another batch. Such an approach involves training, validating and assessing metrics such as loss, accuracy and recall with different training folds up to five times.

The training time for each fold takes 5 minutes. Therefore, doing the training for all 5 folds will take a total of 25 minutes. Unfortunately, this technique is computationally expensive however it helps to validate the data being selected for training is not biased for the model.

Overall the cross validation reveals the same results obtained at the beginning of this experiment. It confirms there was no significant change on the classificator depending on the data randomly selected. In the end of this experiment, it showed that the initial data selected does not influence the final result of the accuracy and recall. The accuracy and recall for MLP remained to be 0.65 and for CNN it remained to be 0.82.

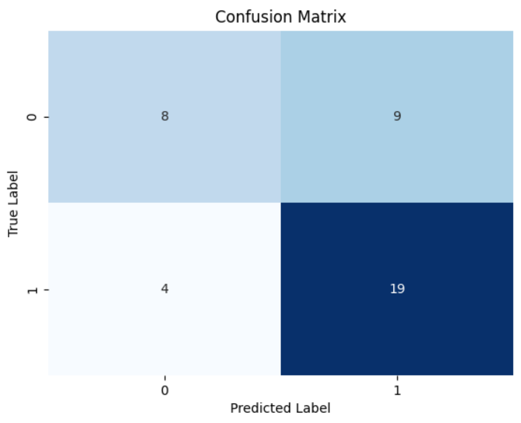


Figure 3. MLP results (0.65 accuracy)

5. Results and Discussion

After experimenting with different machine learning models, techniques and adjusting parameters, there was a tradeoff between conserving information and accelerating the training time. In the research paper, the researchers have discussed using image pixels of

size 768x560 (3). In this project, the target size of the image was initially set to 766x575 pixels. Unfortunately, the time to train this model required long delays. Therefore having used batch normalization has accelerated the training and has also been found to use image pixels of 199x137 was the most optimal and used for all three models.
First off the MLP model was implemented and executed. The accuracy results obtained for this model was 65% as shown in Figure 3. Next, the two dimension CNN model was executed. The CNN models for both 1D/2D differ to that of MLP as different combinations were tried in each layer to define the correct one. One of the parameters is the dropout value. In the beginning the dropout value was 30,30,50 for layers 1, 2 and 3 respectively in the second CNN model. But finally, in this project with the reduction of the size of the image it was a good tradeoff, as the size of the image was reduced from 30,30,50 to 0.05, 0.05, 0.35. Which is good as with the new value only 5%, 5% and 35% is dropped out from the connections. Thus combination of regularization and the dropout helped to reach the accuracy and recall of 82%. Using the same techniques for the first CNN model, the accuracy and recall results were 72%, which is still good enough. It shows that going from the first CNN model (70% accuracy and 76% recall) to the second CNN model (82% accuracy and recall) (Figure 4) yields approximately 10% better in terms of results as shown in Image 5.

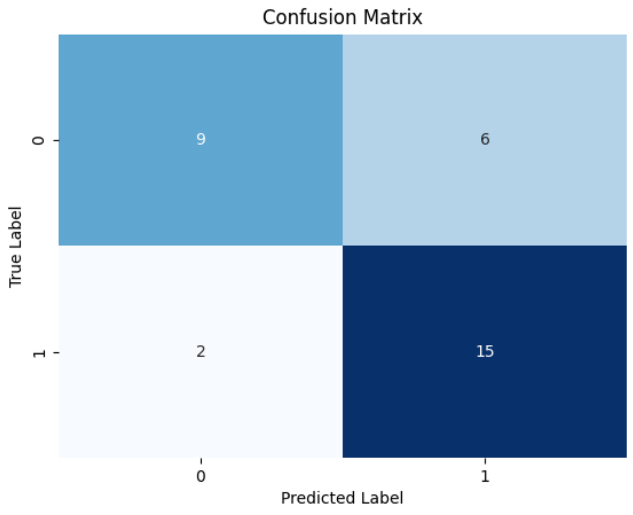
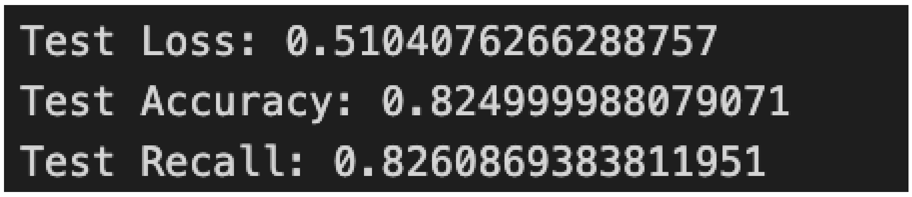


Figure 4. CNN results (0.82 accuracy)

After undergoing a lot of research and many experiments, a lot was learned in this project. Image processing tools and insights were gained as well as tons of hands-on practice on feature extraction. We learned that MLP has lower accuracy results in comparison to the CNN models. This is because the CNN models contain machine learning techniques that help improve the models to output efficient results. Furthermore, we learned that the model's performance sometimes is influenced by the stochastic nature of weight initialization. Consequently, when executing our code sometimes there are variations in the results and this is due to the randomness in weight initialization.

Overall our implementation strategy did work, despite not being able to implement one algorithm named Dull-Razor hair removal algorithm, another alternative named Morphological Black Hat Transformation algorithm was found and implemented. Other than this, all other implementations were successfully implemented, besides the specifics that were missing from the paper. With many different machine learning techniques there definitely can be other ways to implement this research. Future work can include looking into different machine learning techniques into this research and see how the accuracy may change.



```
Test Loss: 0.5104076266288757
Test Accuracy: 0.824999988079071
Test Recall: 0.8260869383811951
```

Image 5. Success file of the second CNN model showing the results of loss, accuracy and recall values.

6. Challenges and Future Work

The research paper lacks details on various aspects of the project. The researchers are experts in the domain of melanoma and skin cancer. The research paper seems to appear more of a medical focused paper rather than a machine learning paper. Although the authors mention information regarding the neural network, they provide limited details on the implementation, such as the number of epochs, iterations, specifics about the weights and activation functions. Additionally, the authors did not provide any clarifications on how the features were constructed which has resulted in a lot of assumptions for the project. The gaps represented challenges in replicating accurate results as presented in the paper as the necessary information was not sufficiently provided by the authors.

Moving forward, the attention should be directed towards obtaining additional details from the authors to help improve the methods used for preprocessing images, extracting features and optimizing the architecture models in this project. With the possibility of also having a skin cancer specialist to help guide the project with valuable insights and suggestions.

7. Conclusions

In summary, the goal on the project was to build a model using the paper as a reference while having a high accuracy and recall value. The project aimed to enhance the diagnosis of melanoma skin lesions through computational approaches. These include pre processing, feature extraction and machine learning models and techniques. Although many challenges were faced due to limited details from the reference paper, the project has successfully implemented techniques like batch normalization and regularization to improve the accuracy. Furthermore, feature extraction has included important components like area, perimeter, color presence, asymmetry indexes and best fit ellipse properties. The machine learning models including Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN), were implemented to understand the diverse patterns in data. The 5-fold cross-validation technique validated the robustness of the models. Moving forward in terms of future work, there is still potential to further improve this project. This can be done with collaboration of domain experts and obtaining additional insights from the original authors to further refine the results. Details on project execution can be found in the provided readme.txt file, offering a comprehensive guide for reproducing the results in this project.

8. References

(1) Popescu.M.C, Balas. V.E, Popescu. L & Masorakis.N. *Multilayer perceptron and neural networks*. (2009). https://www.researchgate.net/publication/228340819_Multilayer_perceptron_and_neural_networks

(2) ADDI Project. *Automatic Computer-Based Diagnosis System for Dermoscopy Images.* https://www.fc.up.pt/addi/ph2%20database.html

(3) S.Majumdera & M.A. Ullaha. *A Computational Approach to Pertinent Feature Extraction for Diagnosis of Melanoma Skin Lesion.* (2019). Pattern Recognition and Image Analysis, Vol. 29, No. 3, pp. 503–514

(4) Ioffe.S & Szegedy.C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* (2015). Vol. 37https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43442.pdf

(5) Indolia.S, Goswami. A.K, Mishra. S.P & Asopa.P. Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. (2018). Vol. 132 https://www.sciencedirect.com/science/article/pii/S1877050918308019

(6) Tian.Y & Zhang.Y. *A comprehensive survey on regularization strategies in machine learning.* (2022) Vol.80 p146-166. https://www.sciencedirect.com/science/article/abs/pii/S156625352100230X

(7) Marimuthu. P. *Dropout Regularization in Deep Learning*. Analytics Vidhya. (2023) https://www.analyticsvidhya.com/blog/2022/08/dropout-regularization-in-deep-learning/#:~:text=In%20machine%20learning%2C%20“dropout”,are%20codependent%20with%20one%20another.

(8) Brownlee.J. *A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks*. Machine Learning Mastery. (2019) https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/

(9). Baldi.P & Sadowski.P. *The dropout learning algorithm*. (2014). Vol 210. Pg78-122. https://www.sciencedirect.com/science/article/pii/S0004370214000216

(10) *Digital Hair Removal - GitHub Code Source*. (2017). https://github.com/sunnyshah2894/DigitalHairRemoval/blob/master/DigitalHairRemoval.py

(11) Refaeilzadeh.P, Tang.L & Liu.H. *Cross-Validation*. Encyclopedia of Database Systems. pp532-538. https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_565#:~:text=Definition,used%20to%20validate%20the%20model.