Universitá degli Studi di Cassino e del Lazio Meridionale

Image Processing and Analysis
&
Machine Learning and Deep Learning

Professors:
Alessandro Bria
Claudio Marrocco

Skin Lesion Segmentation and Classification Report

Leticia Itzel Rivera Contreras
Nisma Amjad

E-mails:

nisma.amjad@studentmail.unicas.it

leticiaitzel.riveracontreras@studentmail.unicas.it

Abstract:

Skin cancer is a rapidly growing disease and melanoma is one of its types which can be treated if diagnosed in early stages. For a dermatologist it is a lot of effort and time consuming to diagnose and inspect the dermoscopic images. Therefore, we designed a system to classify three types using many experiments through conventional machine learning and deep learning methods to classify melanoma, begin and seborrheic keratosis.

Introduction:

Skin is the largest organ in the whole body. Skin Cancer is one of the most fastly growing cancer maladies worldwide[1]. Melanoma is a type of skin cancer happening to have 76,389 new cases and over 100,00 deaths in the United States in 2016 [2]therefore early diagnosis is not only important but very critical, as study showed survival rate for Melanoma increased over 90% if detected in the early stage [1]. Since skin cancer occurs at the surface of skin, visual inspection by a dermatologist using Dermoscopy is the common way for diagnosis. To add on, it requires a lot of time and effort for the Inspection of the dermoscopic images for dermatologists. To aid the dermatologist and improve the accuracy of the diagnosis computer-aided diagnosis systems have been developed. Skin lesion segmentation and classification is very important to predict and give diagnosis in a CAD system. However, automatic skin lesion segmentation and classification of skin lesions is very challenging due to the large variety of appearance in color, texture, and size for different patients. In addition to these hair, veins, medical gauze and light reflections makes it a more difficult task.

Celebi at el. [3] presented a classification technique to classify pigmented skin lesions from dermoscopy images using color, texture and shape features. For color and texture features, the image is divided into a set of regions presenting significant clinical properties of the lesions. For shape features, the method performs lesion border detection to separate lesions from background skin.

Barata et al., 2014 [4] presented two systems for melanoma detection for dermoscopic images. Global features were used by the first system to classify skin lesions while the second system required local features and the bag-of-features classifier to categorize skin lesions. The results showed that color features obtained better performance than texture features when used alone for the problem of skin lesion classification.

Since there are many methods in the literature to perform melanoma classification of skin cancer, we designed a system using many experiments through conventional machine learning and deep learning methods to classify melanoma using multiple techniques, including a set of 3 preprocessing methods, 4 feature extraction methods and 3 different classifiers ISIC 2017 data set melanoma dataset and summarize our findings from this in-depth evaluation.

Dataset:

We have worked on a dermoscopy dataset which is called the IEEE International Symposium on Biomedical Imaging (ISBI) 2017 segmentation challenge dataset given by the ISIC archive[5]. This dataset includes

8-bit RGB dermoscopy images having multiple image sizes ranging from 540 × 722 to 4,499 × 6,748 pixels. Furthermore, this dataset has 2,000 training images along with validation that is 150 and testing datasets of 600 images, respectively. These images include skin lesions labeled as benign, seborrheic keratosis or melanoma. [5]. We have used 2000 training images for machine and deep learning while 200 images for image processing from this dataset.

Distribution of the ISIC 2017 Challenge Dataset

Data Type	Benign	SK	Melanoma	Total
Training Data	1,372	254	374	**2,000**
Validation Data	78	42	30	**150**
Test Data	393	90	117	**600**
Total	**1,843**	**386**	**521**	**2,750**

Table 1: Has the distribution of the dataset. Moreover, this dataset also contains the ground-truth segmented lesion boundaries in the form of binary masks.

Advanced image analysis

Methodology

We proposed a semi-supervised learning technique to automatically segment skin lesions in a given image. The method includes two major steps, preprocessing and segmentation. The preprocessing step is applied to filter out the artifacts present in an image and this filtered image is further used for segmenting the skin lesion. The segmentation is done with GrabCut and k-means clustering algorithms. Due to the multiple sizes of image (i.e., the majority ratio of height to width of 3:4), we resized images into 192 × 256 pixels utilizing bilinear interpolation according to the report in [5], where the optimal segmentation achievement was made with the image size of 192 × 256 among other different image sizes. Python used with OpenCv and other multiple libraries was used.

The workflow of the proposed method is depicted in Fig. 1 and is explained below:

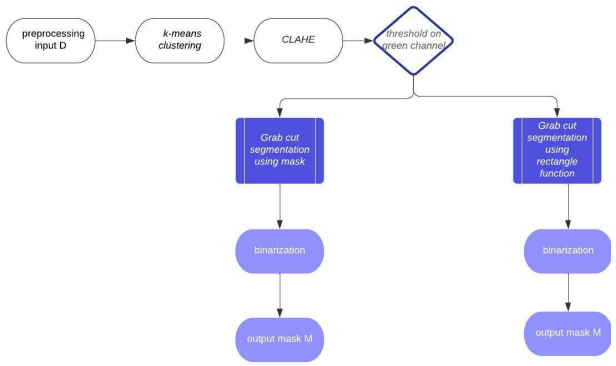
- Image Input: Dermoscopic image D.
- Output: Binary mask of the skin lesion M.

A. Preprocessed image was taken as the input of the segmentation pipeline.
B. Afterwards k-means clustering was applied to cluster the pixels and the BGR image was converted to HSV.
C. Then the green channel was extracted from the image using thresholding and channels were merged.
D. Following that thresholding strategy was applied whether to apply a mask or not for applying GrabCut Segmentation.
E. GrabCut is used for background and foreground extraction techniques that uses minimum graph cut for segmentation. Authors in [15] presented the algorithm and a

modified version of GrabCut algorithm was implemented for skin lesion segmentation.

F. Finally binarization is used for converting into binary images

Fig.1: flow diagram of the process



A). Preprocessing:

The first step for this project was to download the dataset of the ISIC 2017 challenge explained before. The training set contains 2000 images of skin lesion segmentation with three different diagnoses, 1372 images labeled as benign, 374 labeled as melanoma, and the other 254 images labeled as seborrheic keratosis.

a). Class Imbalance

It means that the dataset was not class-balanced. To solve the problem, we had to split the dataset class-wise. Each dataset should have the same amount of diagnoses, and 50% of the images had to be part of the training set, and the other 50% to the test set.

To make this arrangement. First, we added two columns to the original CSV, one labeled as benign to have the three classes and the other one labeled as final labels to have the final diagnosis. It should be noted that we used the encoding labels, giving them the following values:

```
labels: 0 benign, 1 melanoma,2 keratosis
```

After that, we used the *sklearn* library to import *the stratified kfold* function, allowing us to divide the data equally. Moreover, with the *shutil* library, we copied the training and test images to the corresponding folder.

b). Hair removal

The first pre-process we applied to the images was hair removal to eliminate the significant amount of hair. We created our function called *the smplextract_hair function.* The proposed method was based on contrast enhancement and morphological operations to identify the hair pixels. The *Inpaint function* was used for image interpolation to replace hair pixels with the neighboring pixels.

The function receives as input the original image, and then it is converted to grayscale. After that, we applied CLAHE to get an illumination corrected image[6]. In order to

avoid noise amplification, we also applied the contrast limit parameter.

In the second step, we performed a sum of the black hats of the image. The black hat is the difference between the closing and the original image. It points out the dark object against the white background; in this case, the skin lesion and the hairs are darker than the skin.

We used the *sum_operation_se_different_directions function* to create the number of structural elements that almost cover the 360 degrees of the space using different functions of the *open cv* library. These elements allow us to find all the possible directions of the hair to create the mask we want to eliminate at the end. We utilized the *line function* to draw the structure element. The transformations are done using a warpAffine function that remaps routines and then the RotationMatrix2D to rotate the elements. Then, we performed the sum of the elements with a morphology operation that we will define in the main code later (in this case, black hats). The final step of the function is the normalization of pixel intensities.

Returning to the main function, we defined the width, height, and number of elements as input to the *sum_operation_se_different_directions* function. This technique is employed to find the contours of the hairs concerning the direction. This morphology operation is also used to enhance the darker components in the image. We tried with diverse sizes and numbers of structure elements. The last configuration that gave us better results was the following one:

```
width= 29
height=2
n_se=15
```

We tried with more than 30 elements to have better space coverage, but the computational cost was so high.

The third step is Otsu's binarization for the mask. First, a binary threshold is administered to raise the intensity of the regions where the hair was identified [7].

After an Otsu's threshold is applied to the sum of black hats to detect the dark corners that may have the same pixel intensities as the lesion and can influence the performance. This type of threshold is used because it sets an automatic threshold that should be the optimum one.

The fourth step is to make a dilation on the image to prepare the mask for the next step.

The last step is applying the *Inpaint function* to the hair regions detected in the previous steps (mask). This function replaces the hair pixels with background pixels to recover the image.

Fig.2: This shows the results of 1). Original image, 2) sum of black hats, 3) threshold, 4) dilation, 5) inpaint and 6) structuring element

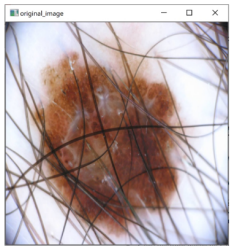


Figure 1. The original image (input)

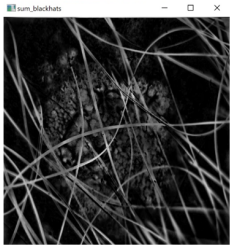


Figure 2. Sum of black hats

Figure 3. Threshold

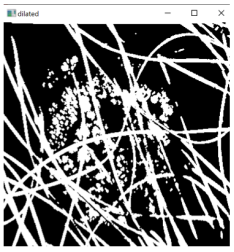


Figure 4. Dilation

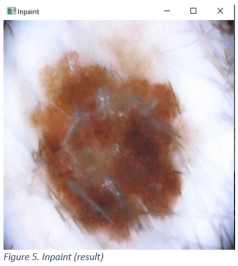


Figure 5. Inpaint (result)

Figure 6. Structure element

c). median blur:

Furthermore, we applied a median filter which is a common way to reduce noise and smooth the blur images since the dataset images are from several sources. We use a standard kernel size 5 × 5 median filter.

2) Color quantization:

We have used the k-means clustering algorithm to group the pixels of the obtained foreground image in RGB colour space. The process that minimizes the number of colours given an image maintaining the quality of image and information is called color quantization. Here we used k-means for that.

a) k-means clustering:

The method performs clustering by choosing the first cluster centre randomly from the input pixels. The cluster centres are known from the rest of the pixels on the basis of probability proportional to its squared distance from the pixels' closest existing cluster centre. As cluster centres are far apart this is done to reduce the probability of bad initialization as proposed by Arthur and Vassilvitskii [9][8]. This helps input pixels to grouped into:

1. interior skin lesion,
2. lesion boundary,
3. lesion background (normal skin),
4. background image.

From the obtained clusters, the cluster with pixel colour which is similar to skin lesion colour is retained.

b) CLAHE (correction technique adaptive histogram equalization):

Then the BGR image was converted to HSV color space. In HSV color space most of the skin lesion appears to be green so we take the green channel for thresholding. Illumination correction technique CLAHE algorithm was applied on each channel separately to remove uneven illumination[8]. Afterward all the color channels were merged to get the enhanced image.

3) Grabcut:

The GrabCut algorithm works in a way to simply apply a rectangle around the object and then define all pixels inside the rectangle as foreground and all pixels outside the rectangle as the initial background pixels. In other terms, the opacity of the pixels inside the applied rectangle is α=1 and outside it is α=0. To add on, the author[10] have used a Gaussian mixture model (GMM) to generate 2 RGB (three-channel) GMMs, one for background and one for foreground. Furthermore, k-means algorithm was used to perform clustering into k classes as k gaussian components of GMMs according to their RGB values. A graph was initiated depending on the GMMs distribution. Pixels were the nodes of the graph, all the pixels belonging to the foreground connected to the source node while all the pixels at background connected to the sink node. Next, was initialization of each gaussian component of GMMs, and calculate the mean and covariance values of parameters from the sample set of pixels, and each weight is the ratio of pixels' number of the gaussian component to total number of pixels. Afterward probabilities of each pixel inside the rectangle belonging to each gaussian component of the GMMs was estimated. Then a mincut algorithm was used to segment the graph used to cut the graph into source node and sink node with minimum cost function. Here the cost function is the sum of all weights of the edges that are cut. After the cut, all the pixels connected to the source node become foreground and those connected to the sink node become background. The process is continued until the classification converges.

Proposed GrabCut algorithm:

We extracted green color from the image and applied thresholding to obtain the final mask having value 1 as foreground and wherever 0 was considered as background. To perform GrabCut two approaches were used. First was using a mask and the second was using a rectangle. A Threshold was applied which was calculated on the basis of the mask dimension and intensity value of pixels, if the summation of intensity value exceeds a rectangle is generated and used for grabcut otherwise the extracted mask performs. Finally binarization is performed after applying a median filter.

Results:

Jaccard score:

The final part of our code is the obtention of the metric. In this case, we are getting the Jaccard score through the *function Jaccard Score* from the *sklearn library* in a set of the first 200 images.

Our first Jaccard was 0.59. After some modifications in the threshold parameter and the rectangle size we obtained a

Jaccard of 0.62. Also, with the removal of the field of view our Jaccard was not improving, so we decided not to implement

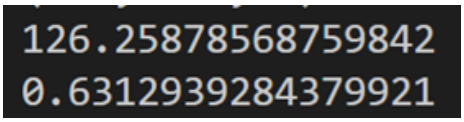
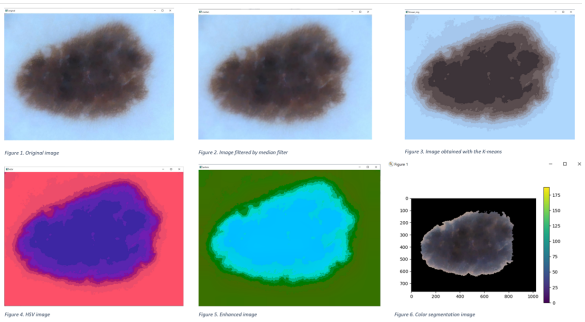
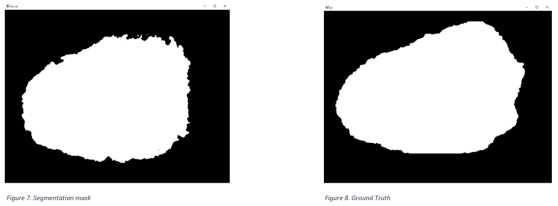


Fig.3: Jaccard score.

the field of view function. The last modification that we did was adjusting the threshold again achieving a Jaccard of 0.63

Fig.4: All of the steps of the segmentation function are shown in the images below.





Machine Learning:

Methodology:

Our classification evaluation procedure is done in the following steps:

- Data Preprocessing
- Feature Extraction
- Classification
- Result Analysis

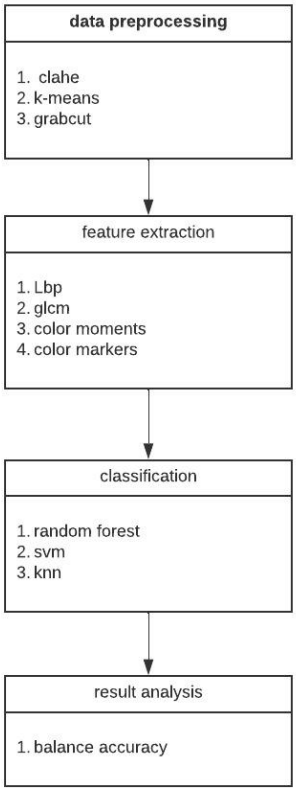


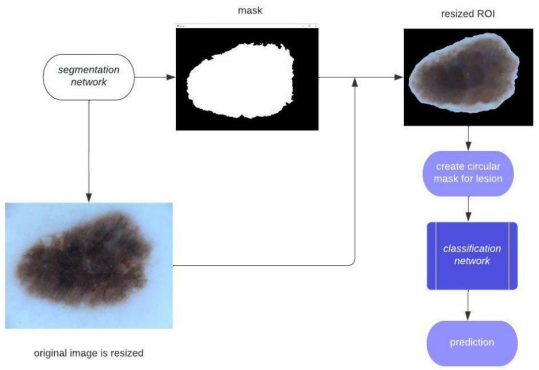
Fig.4: shows the flow followed for machine learning

1). Data preprocessing:

It is mainly done during the image segmentation. After attaining the segmentation masks we apply a combination of bit and operation mask to take out the characters of the original images and then the and afterwards we take the ROI(range of interest). The obtained image that is the ROI is then upscaled to the original size of the input image. The trade-off here is the staircase effect created.

Fig.5: The images after segmentation

Fig.6: shows the flow of steps performed for machine learning preprocess



2). Feature Extraction:

We require a special feature selection step during classification achieved by removing redundant, noisy and unimportant features. This step has several benefits including reducing feature extraction time, reducing complexity for the next classification step, improving prediction results, reducing training and testing time. [11] Each extracted feature is a vector representing an input image. We propose to use below features for different aspects of an input image:

a). LBP (Local Binary Pattern) is a visual descriptor and is the most powerful and straightforward technique for feature extraction. It thresholded neighboring pixels and handles the result as a binary number. It calculates the histogram of each cell in neighboring pixels, combines all together and performs normalization.[11] In the LBP, we received 60 bins of features using the three channels of color space RGB and HSV that is 10 bins for each channel of colorspace.

b). GLCM (Gray-Level Co-Occurrence Matrix) is a statistical method analyzing texture related properties of an image which relies on the spatial dependency of pixels in the gray-level co-occurrence matrix[12]. The matrix calculates the frequency of pairs of pixels in the spatial domain and extracts statistical values. After deploying it we focused on features of contrast, correlation, homogeneity and dissimilarity using the three cancels of color space RGB and HSV. Contrast statistics computed variation between neighboring pixels and weights increase exponentially. Homogeneity estimates the density of distribution of elements relating to the GLCM diagonal. Correlation measures the dependency of gray levels on those of neighboring pixels. Dissimilarity is variation between pixels but here the weights increase linearly.

Altogether there were 24 features extracted from GLCM.

c). Color Moments:

Color constancy is the ability of the human eye that ensures the perception of the color of objects to remain relatively constant under varying illumination conditions. We achieved color constancy by detecting the color of the image independent of the light source and the amount of varying illumination and image acquisition methods. it is based on low-level image features, such as max-RGB and gray-world. [13] These low-level methods are used widely due to their very low computational costs, i.e. taking the maximum (max-RGB) or average pixel values (Grey-World). Max-RGB is a simple and fast color constancy algorithm which estimates the light source color from the maximum response of the different color channels[13]. Another well-known simple color constancy method is based on the Grey-World hypothesis [14], which assumes that the average reflectance in the scene is achromatic[13][14]. We implemented 2 statistics-based methods Grey world and MAX-RGB and calculated the first order statistics mean, skew, standard deviation and got 3 illuminations for each image making it 36 features. Furthermore, we calculated the first order statistics mean, skew, standard deviation for RGB and HSV of the image getting 18 features. Total number was 54 features.

d). Color Markers:

We used the function to extract the color markers of colors by setting different thresholds for the image in colors black, red, blue gray, white, light brown, dark brown.

As a result, we received 24 GLCM, 60 LBP and 54 color descriptors, 6 color markers. All features were saved into a feature vector and stored in a csv file called features.csv.

3) Classification:

The third step to solve the classification problem was to perform classification, using the extracted features from the previous step. This was a supervised learning task having three labels benign, malignant and seborrheic keratosis: for each image, its extracted features with their corresponding label are fed to the classification model so that it can learn from the dataset. Implementing author [15] and due to their popularity, results on various datasets we applied following three models:

1. SVM 2. KNN 3. random forest

a). Scaling Data:

Firstly, we implemented standardscalar, feature scaling to rescale the data in order to get every feature column to have mean zero standard deviation. As most machine learning algorithms require scaling the data to work better.

b). Feature Selection:

Secondly, we applied *f*eature selection to select important features in the dataset. It was done by meta-transformer with an estimator that assigns importance to each feature through a specific attribute after fitting. It selects the feature with most importance according to threshold. Here we use Random forest as an embedded method and it provides good predictive performance, low overfitting, and easy interpretability. We had a total of 139 features out of which 64 were used during the training.

c) Models applied:

Here we applied the three models:

1) SVM: (Support Vector Machine) is defined by a separating hyperplane [15]. The labeled training data are divided by an optimal hyperplane which could be used for categorizing new, untrained data.

2) RF: (Random Forest) is a classification technique using multiple decision trees, each containing leaves representing class labels and branches representing conjunctions of features that lead to those labels. The trained decision trees are then used in a randomized fashion to classify an untrained data.[15]

3)KNN: (k-nearest neighbor) is a classification technique and It is a supervised machine learning algorithm.

4) Comparison Metrics:

We applied two approaches using all the features and using just the important ones. The performance measure used was the balance accuracy. The results of statistics are listed in Table.3 and Table.4 for both approaches. In our case the best classifier was SVM with highest balance accuracy for using important features and knn worked better for other ones. Although the balance accuracy was not much we tried scaling the data and trying again. This helped to achieve better results. After the result analysis we came to know that the balance accuracy can further be improved by extracting more visual descriptors features like HOG and SIFT it improves accuracy dramatically and represents texture features. [11]

Table.2: here we can see the used parameters used for each model

name	SVM	RF	KNN
Used parameters	C = [0.01, 0.1, 1, 10] K = ['linear', 'rbf'] param_grid = {'C':C, 'kernel':K} gamma='scale', cv=5	n_estimators = [int(x) for x in np.linspace(start=70, stop=250, num=25)] max_features = ['auto', 'sqrt'] max_depth = [int(x) for x in np.linspace(start=5, stop=100, num=10)] min_samples_leaf = [1, 2, 5, 10] min_samples_split = [2, 5, 10, 20] n_iter=100, cv=5,	param_grid=7,8,9,10,12,15,20, cv=5

Table.3: here we can see the balance accuracy of three model with the parameters used without feature extraction

Name	SVM	RF	KNN
Best parameters	'C': 1, 'kernel': 'rbf'	'n_estimators': 152, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 36, 'bootstrap': False	para_grid =20
Balance accuracy	0.495	0.46	0.409

Table.4: here we can see the balance accuracy of three model with the parameters used with feature extraction

name	SVM	RF	KNN
Best parameters	'C': 10, 'kernel': 'linear'	'n_estimators': 205, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 89, 'bootstrap': False	param_grid=12, cv=5
Balance accuracy	0.53	0.49	0.408

Deep Learning:

Methodology for the Segmentation Network:

Our segmentation procedure is done in the following steps:

- Data Preprocessing
- Network architecture
- Training
- Testing
- Results

1). Data preprocessing:

The dataset used for deep learning was the original one from the ISIC 2017 challenge. For the validation, 150 images, and for the test, 600 images from the original sets. For the training, the 2000 images from the training set without splitting.

The input for the network was the 2000 images of the training set resized to 400 x 300 (width x height).

We created additional images (data augmentation) through the *albumentations library,* applying flip and randomly resize methods for the training. Then, we normalized them with the values of the average mean and standard deviation of the dataset. We obtained the following:

Fig.7 shows the normalization values

```
mean=[0.71, 0.59, 0.54],
std=[0.09, 0.11, 0.12],
max_pixel_value=255.0,
```

Figure 1. Normalization values

For the validation, we just applied the resize and the normalization, but not randomly.

2). Model Architecture

It should be mentioned that for the deep learning segmentation network, we took inspiration from the code of Mahmoud and ElBanna. We modified their model in a way that it functioned for us.

A UNET architecture was used to perform the semantic segmentation. It generated an estimated probability for each pixel in the original image [17]. The UNET took an input image of 400 by 300 pixels and produced a probability map with the exact dimensions. Three down-sampling layers, an utterly connected layer at the bottom of the "U," and three up-sampling layers comprised the UNET. The output channels are equal to the number of filters.

3). Training

The network was trained with the following configuration of hyperparameters:

Batch size= 10
Number of epochs= 42
Number of workers=2
Learning rate=variable

The network was trained using the ADAM optimizer, the IOU loss function, and diverse learning rates initialized to 0.01, followed by 0.001 and 0.0001 posteriorly. We built three models with different learning rates to determine which one we obtained the best accuracy. After training with these first three models, we selected the best one to train the network again with the same hyperparameters and observe if we obtained the same result. After training this model, we executed the training again but added the batch normalization to see if the performance increased.

In the beginning, we started with the following configuration:

- Batch size= 32
- Number of epochs= 65
- Number of workers=2
- Learning rate=variable

In the model trained with a learning rate of 0.0001, we got an accuracy on the validation set of 0.6535 and a loss of 0.3464.

In the model trained with the same learning rate but adding the batch normalization, we reached an accuracy of 0.6396 and a loss of 0.3603, similar results to the one without batch normalization. However, we realized, observing the graphs below, that 65 epochs were a high number because around epoch 35, the line did not have a significant change, so we decided to train with a smaller number of epochs. Also, it seems

that the model is overfitting because the training accuracy is higher than the validation, and the training loss is decreasing significantly with respect to the validation loss.

Fig.8,9 demonstrate the loss and the accuracy of training vs validation

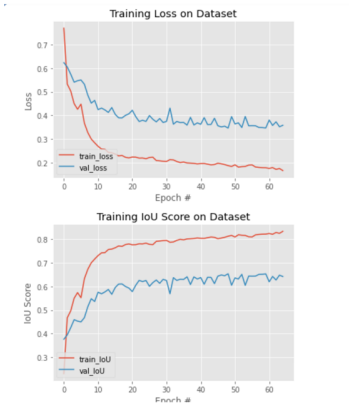


Figure 2. Plot of the best model obtained after the training with 65 epochs

Figure 3. Plot of the best model with batch normalization obtained after the training with 65 epochs

Another problem we found was that the RAM was not supporting the full training, so we reduced the batch size to avoid memory saturation.

After this first trial, we trained with the following settings:

- Batch size= 10
- Number of epochs= 15
- Number of workers=2
- Learning rate=variable

Here we found that our three models did not train. We obtained an accuracy of 0.4187 and an average loss of 0.5812 in the model with a learning rate of 0.0001.

We got an accuracy of 0.6735 and a loss of 0.3265 in the model with batch normalization. The results were very inconsistent; apparently, they suggested increasing the number of epochs to reach higher accuracy.

Figure 4. Plot of the best model with batch normalization obtained after the training with 15 epochs

After this failed training, we determined to increase the number of epochs to 25. We set the hyperparameters with the previous configuration. We earned an accuracy in both models of 0.4246 and a loss of 0.5753. The other models with a more considerable learning rate were not trained because their accuracy and loss were the same in all epochs. Regardless of the value, we knew

something was wrong with the training, so we agreed to increase the number of epochs again.

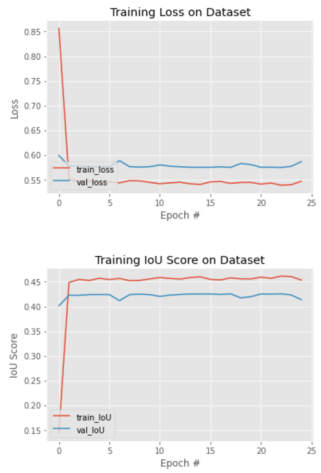


Figure 5. Plot of the best model obtained after the training with 25 epochs

It is worth mentioning that in the above training, we were setting in the optimizer a weight decay to 0.01. Then, for the subsequent trial, we changed this parameter to a lower value very close to zero. Also, we raised the number of epochs to 35. The results in the best model were: an accuracy of 0.7770 and a loss of 0.2229. With batch normalization: accuracy of 0.7928 and loss of 0.2071. We can observe these outcomes in the following graphs:

Figure 6. Plot of the best model obtained after the training with 35 epochs

Figure 7. Plot of the best model with batch normalization ol

Nevertheless, we noticed that some models were not trained because we got an accuracy of 0 and a loss of 1. With this terrible result, we reviewed other parameters in the network to have better results and to fix the problem of not training.

Hence, we set the threshold of the IOU function to 0.1 instead of 0.5. We trained with 42 epochs with the same hyperparameters as before, and we achieved the following results: accuracy of 0.7996 and loss of 0.2003 and batch

normalization: accuracy of 0.8059 and loss of 0.1940. The last one was the best result obtained during the experiments.

We can observe these outcomes in the following graphs:

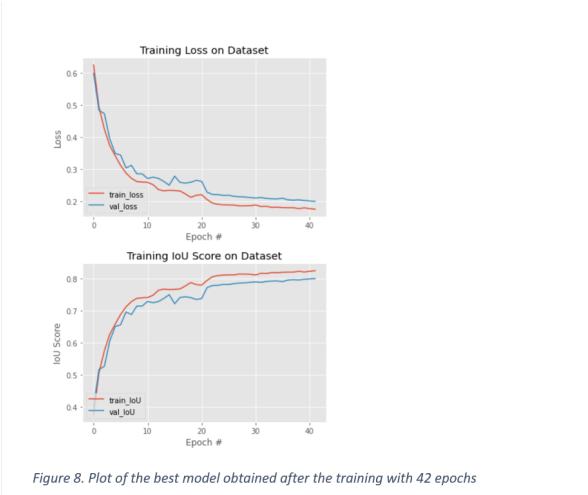


Figure 8. Plot of the best model obtained after the training with 42 epochs

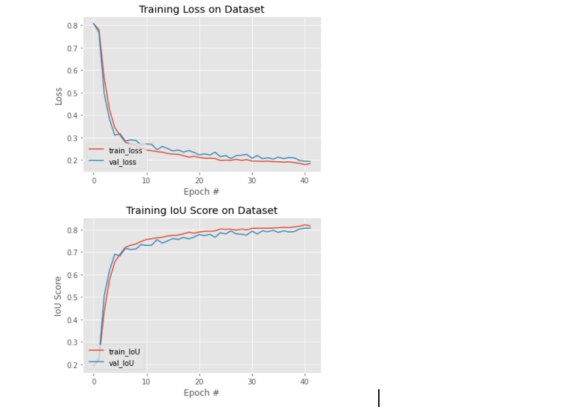


Figure 9. Plot of the best model with batch normalization obtained after the training with 42 epochs

Every model took around 6 and a half to 7 hours of training. The total hours of training were around 35.

4). Testing

For the testing we chose the best model between the best model without or with batch normalization. The network selected the best one according to the results.

We can see the final outcomes of every experiment in the next table.

5). Results

Models	Number of epochs	Accuracy	Loss
Best model on validation set	65	0.6535	0.3464
Best model w/Batch Normalization on validation set	65	0.6396	0.3603
Test	65	0.6424	0.3576
Best model on validation set	15	0.4187	0.5812
Best model w/Batch Normalization on validation set	15	0.6735	0.3265
Test	15	0.4239	0.5857
Best model on validation set	25	0.4246	0.5753
Best model w/Batch Normalization on validation set	25	0.4246	0.5753
Test	25	0.5093	0.5038
Best model on validation set	35	0.7770	0.2229
Best model w/Batch Normalization on validation set	35	0.7928	0.2071
Test	35	0.7864	0.2136
Best model on validation set	42	0.7996	0.2003
Best model w/Batch Normalization on validation set	42	0.8059	0.1940
Test	42	0.8113	0.1880

Table 1. Segmentation results

Methodology for the Classification Network:

Our classification procedure is done in the following steps:

- Dataset preparation
- Data Preprocessing

- Network architecture
- Training
- Testing
- Results

1.)Dataset Preparation

The dataset used for deep learning was the original one from the ISIC 2017 challenge. For the training, the 2000 images from the training set without splitting. For the validation, 150 images, and for the test, 600 images from the corresponding original sets.

We created a directory structure as follows:

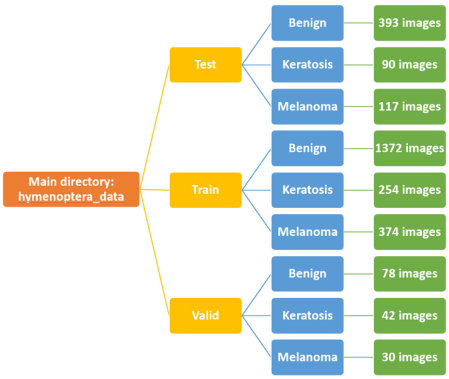


Figure 10. Structure of the Image Directory

We based on the example of the hymenoptera dataset. For that reason, the name of the main directory is hymenoptera. If we changed that name, the network made a mistake, so we decided to conserve it.

The following directories were labeled as Test, Train, and Valid. Inside each directory, we constructed three sub-directories named with the three different classes: benign, keratosis, and melanoma. Inside are the corresponding number of images according to the diagnosis and set.

It is necessary to structure the data as in the image above to assure the excellent performance of the pre-trained network.

2.) Preprocessing

The input for the network was the 2000 images of the training set resized to 256 x 256 (width x height) within a scale range of 0.8 to 1.0.

We created additional images (data augmentation) by randomly applying flip and rotation transformations for the training set. For the validation and test sets, we did not do the randomization. However, each image was rotated 15 degrees and then resized to 256 x 256 and cropped out the center 224 x 224 to be able to be used with the pre-trained model. After that, the image was converted into a tensor and normalized using the mean and standard deviation of all ImageNet images.

```
transforms.Normalize([0.485, 0.456, 0.406],
                     [0.229, 0.224, 0.225])
```

Figure 11. Normalization values of the ImageNet

For the classification, we started with a pre-trained network based on ResNet18, but it was not performing well. Therefore, we implemented another pre-trained network for multi-class classification based on ResNet50.

3.) Model Architecture

Using pre-trained networks helps us get good results even if the dataset is small. A pre-trained ResNet50 model was used for the skin lesion classification. In our case, we selected this network because, in his article, Canziani mentions that ResNet50 is the model that strikes a decent balance between accuracy and inference time [16].

We replaced the final layer of the ResNet50 with a small collection of Sequential layers. A linear layer receives the inputs from ResNet50's last fully linked layer. It contains 256 outputs which are fed into the ReLU and Dropout layers. The 256x3 linear layer is then applied, with three outputs corresponding to our three classes.

4.) Training

The network was trained with the following configuration (the one proposed in the original model):

- Batch size= 32
- Number of epochs= 65
- Learning rate=0.001 in the ADAM optimizer
- Loss function: NLLLoss

After the training, we got an accuracy of 63% on the validation set and 62% on the test set. Without modifying the network, we did not reach good results, so we wanted to increase the number of epochs to have higher accuracy.

In the next experiment, we changed the configuration to:

- Batch size= 32
- Number of epochs= 25
- Learning rate=0.001 in the ADAM optimizer
- Loss function: NLLLoss

We obtained an accuracy of 66.6% on the validation set and 69.83% on the test set. The plots of this training are shown below.

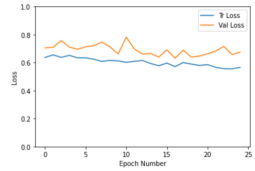


Figure 12. Plot of the train and validation loss at 25 epochs

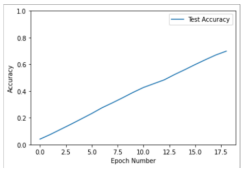


Figure 14. Plot of the test accuracy at 25 epochs

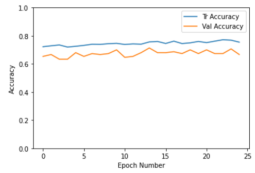


Figure 13. Plot of the train and validation accuracy at 25 epochs

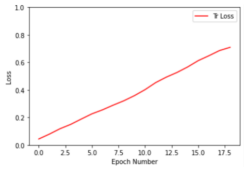


Figure 15. Plot of the test loss at 25 epochs

We were not satisfied with the previous results, so we tried again but with different parameters:

- Batch size= 32
- Number of epochs= 40
- Learning rate=0.0001 in the ADAM optimizer
- Loss function: Cross Entropy

Also, we removed the Softmax layer because we used Cross-Entropy as a loss function instead of the NLLLoss function proposed in the past models.

Here we achieved an accuracy of 68.66% on the validation set and 68.99% on the test set. The following graphs describe these results.

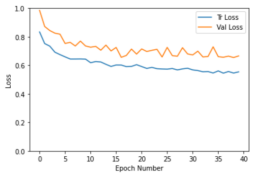


Figure 16. Plot of the train and validation loss at 40 epochs

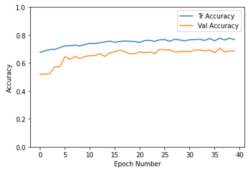


Figure 17. Plot of the train and validation accuracy at 40 epochs

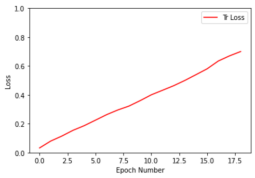


Figure 18. Plot of the test loss at 40 epochs

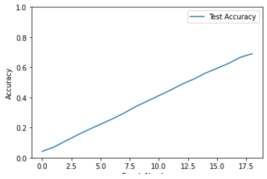


Figure 19. Plot of the test accuracy at 40 epochs

After training the network with 40 epochs, we retrained the network with the same configuration. However, we added the balanced accuracy parameter to measure the absolute accuracy since we are dealing with the issue of imbalanced data. It means that one class appears more than the others.

The outcomes were the same, just with the difference in the balanced accuracy. We obtained on the test a BA of 69.32%.

In order to increment the BA value, we did two more training with the same parameters but with different numbers of epochs. The reached results are reported in the table.

5.) Testing

We had two ways to make the final test. The first one is on a specific image we select from the directory or wherever we want, and the network makes the last prediction.

The code is the following.

```
dataset = '/content/drive/MyDrive/Machinelearning&image/hymenoptera_data'
model = torch.load("{}_model_{}.pt".format(dataset, best_epoch))
predict(model, dataset +'/test/melanoma/ISIC_0014982.jpg')
```

Moreover, the prediction of the chosen image is the next. In this case, we are predicting a melanoma image that reaches a high probability between the other two classes, so this prediction is correct.

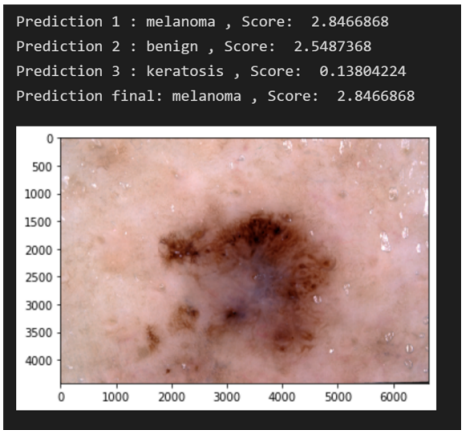


Figure20. Prediction on a specific image

The second way is testing the whole dataset and getting a list of three predictions for each image like the one shown in the following figure.

```
ISIC_0012510.jpg
Prediction 1 : melanoma , Score:  1.4385726
Prediction 2 : benign , Score:  1.2352039
Prediction 3 : keratosis , Score:  0.30611023
Prediction final: melanoma , Score:  1.4385726
ISIC_0013281.jpg
Prediction 1 : keratosis , Score:  4.332807
Prediction 2 : melanoma , Score:  0.45203128
Prediction 3 : benign , Score:  0.39231676
Prediction final: keratosis , Score:  4.332807
ISIC_0012248.jpg
Prediction 1 : keratosis , Score:  1.3537599
Prediction 2 : melanoma , Score:  1.0012969
Prediction 3 : benign , Score:  0.55385697
Prediction final: keratosis , Score:  1.3537599
ISIC_0014503.jpg
Prediction 1 : benign , Score:  7.6077766
Prediction 2 : keratosis , Score:  1.6466873
Prediction 3 : melanoma , Score:  0.05053511
Prediction final: benign , Score:  7.6077766
ISIC_0014278.jpg
Prediction 1 : benign , Score:  2.3397527
Prediction 2 : keratosis , Score:  1.0081471
Prediction 3 : melanoma , Score:  0.16992435
Prediction final: benign , Score:  2.3397527
ISIC_0012548.jpg
Prediction 1 : keratosis , Score:  2.9963717
Prediction 2 : benign , Score:  2.2068903
Prediction 3 : melanoma , Score:  0.12239441
Prediction final: keratosis , Score:  2.9963717
```

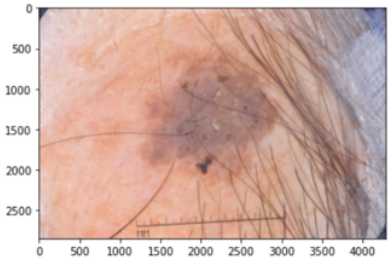


Figure 21. Prediction on the whole test dataset

6.) Results

All the results are reported in the table below.

Models	Number of epochs	Accuracy	Balanced Accuracy	Loss
Validation set	65	63.00%	---	0.6440
Test	65	62.00%	---	0.6368
Validation set	25	66.66%	---	0.6743
Test	25	69.83%	---	0.7093
Validation set	40	68.66%	---	0.6657
Test	40	68.99%	---	0.7003
Validation set	40	71.33%	---	0.6481
Test	40	68.66%	69.32%	0.7096
Validation set	50	72.00%	---	0.6746
Test	50	69.19%	66.74%	0.7037
Validation set	60	70.66%	---	0.6580
Test	60	69.33%	69.79%	0.7121

Table 2. Classification results

References:

1] G. Zongyuan, S. Demyanov, R. Chakravorti, and R. Garnavi, “Skin dis-ease recognition using deep 178 saliency features and multimodal learning of dermoscopy and Clinical Images,” International Conference 179 on MedicalImage Computing and Computer-Assisted Intervention, Springer, Cham, pp. 250-258, 2017 180

2]. Merican Cancer Society, “Key Statistics for Melanoma Skin Cancer,” Available: 181 https://www.cancer.org/cancer/melanoma-skin-cancer/about/key-statistics.html. [Accessed: 18-Sep182 2018]

[3] M. E. Celebi, H. A. Kingravi, B. Uddin, H. Iyatomi, Y. A. Aslandogan, W. V. Stoecker, and R. H. Moss, “A methodological approach to the classification of dermoscopy images,” Computerized Medical imaging and graphics, vol. 31, no. 6, pp. 362–373, 2007.

[4] C. Barata, M. Ruela, M. Francisco, T. Mendonc¸a, and J. S. Marques, “Two systems for the detection of melanomas in dermoscopy images using texture and color features,” IEEE Systems Journal, vol. 8, no. 3, pp. 965–979, 2014.

[5] Y. Yuan, M. Chao and Y. Lo, "Automatic Skin Lesion Segmentation Using Deep Fully Convolutional Networks With Jaccard Distance," in *IEEE Transactions on Medical Imaging*, vol. 36, no. 9, pp. 1876-1886, Sept. 2017, doi: 10.1109/TMI.2017.2695227.

[6] Jaisakthi, S., Mirunalini, P., & Aravindan, C. (2018). Automated skin lesion segmentation of dermoscopic images using GrabCut and k-means algorithms. IET Computer Vision, 12(8), 1088-1095. doi: 10.1049/iet-cvi.2018.5289

[7] Khan, A., Iskandar, D., Al-Asad, J., & El-Nakla, S. (2021). Classification of Skin Lesion

with Hair and Artifacts Removal using Black-hat Morphology and Total Variation. International Journal Of Computing And Digital Systems, 10(1), 597-604. doi: 10.12785/ijcds/100157

[9,8]Jaisakthi, Seetharani Murugaiyan; Mirunalini, Palaniappan; Aravindan, Chandrabose (2018). *Automated skin lesion segmentation of dermoscopic images using GrabCut and k-means algorithms. IET Computer Vision, (), –.* doi:10.1049/iet-cvi.2018.5289

[10]Rother, Carsten; Kolmogorov, Vladimir; Blake, Andrew (2004). *"GrabCut". ACM Transactions on Graphics, 23(3), 309–.* doi:10.1145/1015706.1015720

[11]A comparative study for classification of skin cancer

[12]Comparative analysis of classification techniques for brain magnetic resonance imaging images

D. Bhargava, ... Ayushi Bansal, in Advances in Computational Techniques for Biomedical Image Analysis, 2020

[13] Edge-Based Color Constancy Joost van de Weijer, Theo Gevers, and Arjan Gi

[14] J.-q. Ma, “Content-based image retrieval with hsv color space and texture features,” in Web Information Systems and Mining, 2009. WISM 2009. International Conference on. IEEE, 2009, pp. 61–63

[15] A Comparative Study for Classification of Skin Cancer Tri Cong Pham, Giang Son Tran, Thi Phuong Nghiem, Antoine Doucet, Chi Mai Luong, Van-Dung Hoang

[16] Canziani, A. (2022). Analysis of deep neural networks. Retrieved 14 June 2022, from https://culurciello.medium.com/analysis-of-deep-neural-networks-dcf398e71aae

[17] Berseth, M. (2022). ISIC 2017 - Skin Lesion Analysis Towards Melanoma Detection. Retrieved 14 June 2022, from https://arxiv.org/abs/1703.00523