Kaggle Challenge Report

Alexandre HEYMANN

May 5, 2024

1 Introduction

A skin lesion is defined as a superficial growth or patch of the skin that is visually different and/or has a different texture than its surrounding area. Skin lesions, such as moles or birthmarks, can degenerate and become cancer, with melanoma being the deadliest skin cancer. Its incidence has increased during the last decades, especially in the areas mostly populated by white people.

The most effective treatment is an early detection followed by surgical excision. This is why several approaches for skin cancer detection have been proposed in the last years (non-invasive computer-aided diagnosis (CAD)).

The goal of this challenge is to classify dermoscopic images of skin lesions among eight different diagnostic classes:

1. Melanoma
2. Melanocytic nevus
3. Basal cell carcinoma
4. Actinic keratosis
5. Benign keratosis
6. Dermatofibroma
7. Vascular lesion
8. Squamous cell carcinoma

In this report, we employ two classification methods. The first method utilizes a Random Forest Classifier, while the second method employs a pre-trained ResNet18 Neural Network (NN) model, which we have refined and adapted. In the first approach we manually extract some features which are know as ABCD for asymmetry, border irregularity, colors irregularity and diameter. All those features were combine with the one that we already had (sex, age, position) and then fed to our classifier.
The second approach extracted deep features from the images using a Neural Network. By doing so we get more nuanced features. These models used to perform very well, but we'll discussed about it later.

2 Dataset

The database provided to us is an ISIC skin lesion database. We have access to 25,331 images, with 18,998 designated for training and the remainder for testing to obtain the results of our prediction. Among these training images, only a few were provided with a mask of segmentation by a doctor. We will discuss later the advantages, but also the problems encountered with this segmentation.

Additionally, we have an Excel file with data on the training images, including for each image: its class, the age, gender of the person, and the location of the mole.

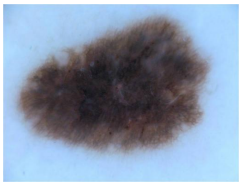

Figure 1: Exemple of a skin lesion

Figure 2: Exemple of the given segmentation mask

2.1 Analysis of the dataset

There are several issues that arise when studying the provided database; some are related to formatting, while others are more complex.
First, we notice that not all values are numeric (such as age or mole location). To perform a more efficient analysis, we must convert these values; we use One-Hot Encoding to do so.

Second, we notice that many values are missing (284 for gender, 324 for age, and 1970 for location). To fill them in, we examine the distribution.

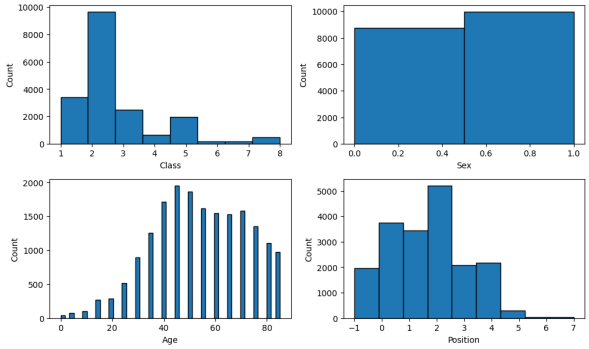

Figure 3: Initial distribution of the dataset

To replace the missing data, we examine the correlation matrix between all the features.

Table 1: Correlation between features

	CLASS	SEX	AGE	POSITION
CLASS	1.000	0.046	0.240	-0.063
SEX	0.046	1.000	0.150	0.051
AGE	0.240	0.150	1.000	0.036
POSITION	-0.063	0.051	0.036	1.000

There is no correlation between gender and the other features. As there are more men than women, we replace the missing data with women to increase their representation. For the positions, we make the same observation, and we assign the missing values randomly to the least represented classes. Finally, the correlation between age and class is the strongest (0.25). We build a simple linear model that gives us an approximation of the person's age based on the melanoma class.
With these changes, we obtain the following distribution.

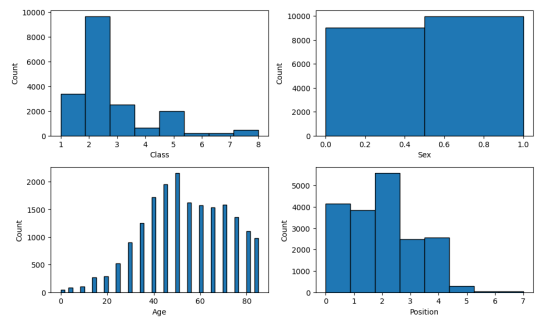

Figure 4: Distribution of the dataset after changes

This is still not balanced, and we will see later that we can assign weights to each feature to give them more or less importance.

3 Preprocessing

The preprocessing step is crucial because it allows us to remove artifacts and defects from the image. It also simplifies the segmentation phase by removing visual clutter. The preprocessing steps are as follows: resizing, smoothing using a median filter, and removing hair.

3.1 Resizng

Dermoscopy images often contain superfluous artifacts such as skin lines or air bubbles around the lesion. These elements reduce the accuracy of contour detection and increase processing time. To mitigate the adverse effects of these artifacts, images should be preprocessed with a smoothing filter.

3.2 Smoothing

The median filter is one of the most common smoothing filters. Median filtering with an appropriate mask size can eliminate most artifacts in a dermoscopy image. According to numerous research articles, the mask size should be proportional to the image size for optimal results, and the mathematical relationship is as follows: given an image of size M by N, the mask size n is determined by:

$$n = \left\lfloor 5\sqrt{\frac{M}{768}\frac{N}{512}} \right\rfloor$$

With M=N=224, we obtain n=1

3.3 Remove hairs

Some image have hairs that may perturbs the analysis. We have to remove them. A common and useful way to do so is to use DullRazor. Indeed, it removes unnecessary parameters from the model without significantly affecting its performance. It does this by identifying and eliminating redundant connections or neurons that don't contribute much to the model's overall accuracy.
We can observe an image before and after preprocessing. This step works well.

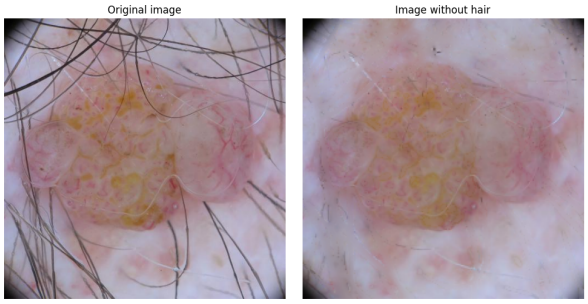

Figure 5: Before and after preprocessing

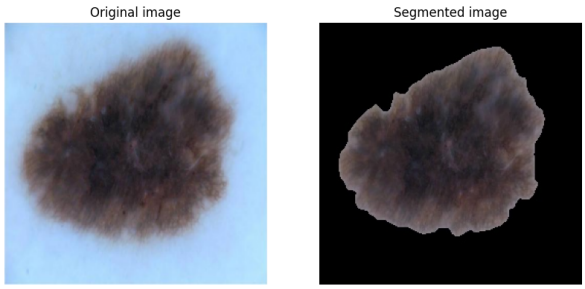

Figure 6: Great segmentation

An important thing to note are the black borders around the images. These borders should be removed, but after many attempts, I was not able to do so. I tried using Python's largestinteriorrectangle function, but it did not work, or it worked but with too much loss of image information. This is an important point to note and should be improved in a future version.

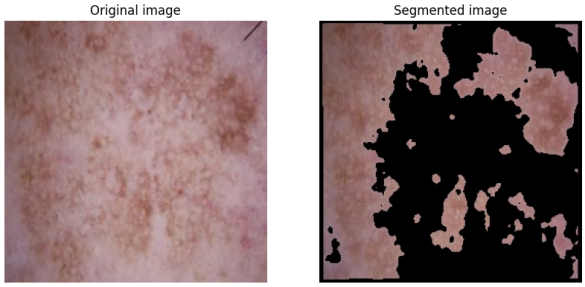

Figure 7: Poor segmentation

4 Segmentation

Segmentation is an important step in preprocessing (but a step that has caused me many problems as we will see). Indeed, it allows us to focus only on the areas of interest in the image and to efficiently calculate features such as symmetry, irregularity, etc., whose role we will see later. As we have seen, we already have segmentations from doctors, but not enough. Segmenting by hand is an option, but much too long and tedious, so we use here a machine learning algorithm, U-Net. The choice of U-Net is quite clear given that this algorithm has been specifically trained for medical tasks. Moreover, U-Net is able to capture many details and manage unbalanced databases. The U-Net architecture consists of a contracting path (encoder) and an expanding path (decoder). The contracting path is used to capture context information, while the expanding path is used to localize the object of interest at high resolution. The encoder and decoder are connected by skip connections, which allow the network to propagate low-level features from the encoder to the decoder.

In our case, we use an 18-layer U-Net (each pair consisting of a convolution layer followed by another convolution layer). This choice was made for computational complexity reasons but could have been extended for more accurate results. We can see the results of the segmentation on a succesfull one, and on a bad one.

We can see that the segmentation algorithm works very well on images without artifacts, but can quickly deteriorate. I tried to solve this problem, without success. I think this is largely due to the black borders (remaining on some segmentations) and the too low complexity of the neural network that I used.

5 Post processing

As a post-processing operation, we are using edge refinement to try to get rid of some edges that may disturb the algorithm. However, after many attempts, we lose too much contrast and textures to gain just a little refinement. So I choose not to keep it.

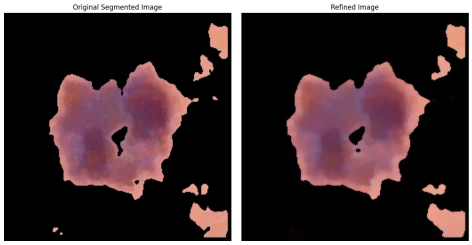

Figure 8: Post processing step

6 Extract features

The given csv file gives us informations about the sex, position and age. Those are important data but are not enough to determine the class of a new image. We thus need more features to extract.
We use the widely used ABCD rule of dermatologists (asymmetry, border irregularity, color irregularity, diameter). These features are crucial for the determination of a melanoma.

6.1 Asymmetry

For the asymmetry measure we use the following relations

$$\begin{aligned} h, w &= \text{image shape} \\ \text{vertical asym} &= \sum_{i=1}^{w//2} |\text{image}[:, i] - \text{image}[:, w-i]| \\ \text{horizontal asym} &= \sum_{i=1}^{h//2} |\text{image}[i, :] - \text{image}[h-i, :]| \\ \text{total asym} &= \frac{\text{vertical asym} + \text{horizontal asym}}{h \times w} \end{aligned}$$

6.2 Border irregularity

To calculate border irregularity, we use the findContours function from the cv2 package. In this package, we use the cv2.RETR_EXTERNAL method to detect only the outer contours and cv2.CHAIN_APPROX_SIMPLE to reduce their number of points. Among the contours found, we select the largest one and then calculate the perimeter and area still with the cv2 package. Finally, we calculate the compactness by dividing the square of the perimeter by the area by the formula :

$$compacity = \frac{perimeter^2}{area}$$

6.3 Color

To calculate color contrasts, we use the K-Means algorithm. We group pixels into clusters and count the number of clusters. The higher the number of clusters, the greater the risk of melanoma.

6.4 Diameter

To calculate the diameter, after finding the contour using cv2, we calculate the convex hull which provides an approximate representation of the shape of the object. We then calculate the diameter of this convex hull.

We used the following formula with C the matrix of the coordinates of the pixels of the border of the image

$$diameter = max_{i,j}||C_i - C_j||$$

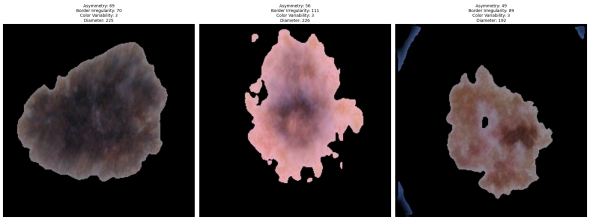

Figure 9: ABCD measures

The measurements make sense, although irregularity and color detection do not work completely. This is probably due to a too restrictive cluster limitation.

7 Random Forest Classifier

We use a Random Forest as our first method. Indeed, these models can handle multi-dimensional problems, which is our case here, and meet the kernel requirements that an SVM may pose. Random Forests are also more easily usable in the case of larger databases and are more versatile than SVMs.
In our case (see the Jupyter Notebook), we obtain 0.47 of correct predictions, which is correct but not convincing. We can note the fact that it is difficult to manage the weights that we can attribute to the classes in the case of a Random Forest (which is not the case for the NN that we will use later). Depending on the number of trees, the depth of the trees, and the number of features, the results can also vary, and these are parameters to take into account.

8 Neural Network

We obtained results of 0.47 with RandomForest, it may be relevant to test with a neural network. After many tests by creating a neural network "from scratch", the results are lower than a RandomForest (around 0.3). So I chose to use a pre-trained model. ResNet18 is a relevant choice due to its relatively moderate depth, which offers an effective compromise between accuracy and computational complexity, making this model particularly suitable for environments with limited hardware resources.

The ResNet18 model was employed with predefined parameters, notably the use of pretraining (pretrained) to leverage the generic features learned on large datasets such as ImageNet. This is advantageous because it significantly reduces the time and data required to reach high performance. Moreover, the integration of a 0.5 dropout layer just before the last fully-connected layer aims to reduce overfitting by randomly ignoring a portion of the features during learning, which should help generalize the results on new unseen data.

The CustomDataset class was set up to facilitate data processing and manipulation, ensuring appropriate conversion of images to RGB and applying a set of transformations. These transformations, including random cropping and rotation, are crucial to increase the diversity of the training data and improve the robustness of the model. The approach adopted for the division into training and validation sets uses a random permutation method to ensure that the model is not biased by a particular order in the data.

The model is configured to adjust dynamically in response to performance on the validation set through the implementation of an early stopping mechanism, which interrupts training if the validation loss does not improve beyond a minimal threshold for a given number of cycles. This prevents unnecessary resource usage and overfitting.Also, I tested the model under different configurations. By changing the size of the images, adding other post-processing steps or using the pre-processed and segmented images (segmentation which were not completely correct so this must have an impact!), the results are better when we let the algorithm "figure it out".

However, the method has some limitations. Although ResNet18 is a good starting point for image classification problems, it may be insufficient to capture finer details in more complex or specific datasets. In this context, deeper or specialized architectures may be necessary. Moreover, the class weighting method used to compensate for class imbalances may not be optimal if class distributions are extremely unbalanced.

To further improve the model, one could consider experimenting with more advanced data augmentation techniques and testing different neural network architectures to compare their performance. Furthermore, a more in-depth exploration of hyperparameters, such as learning rate and decay weight, could provide significant performance gains. Finally, the integration of semi-supervised learning or transfer learning techniques could also be considered to leverage unlabeled datasets or adapt the model to slightly different tasks, thus increasing its flexibility and applicability.

9 Conclusion

We used two main approaches for modeling: Random Forest and a neural network based on ResNet18. Comparing the performances, the neural network achieves a superior precision score of 0.66 against 0.47 for Random Forest. This difference may marks the advantage of deep neural networks in handling complex image processing tasks, where they excel particularly thanks to their ability to capture hierarchical features and adapt to the nuances of image data.

9.1 Limitations of Random Forest

Scalability: Despite its advantages in terms of ease of use and interpretation, Random Forest showed relatively low performance. This may be due to its limitation in processing high dimensionality and interdependent features inherent to images.

Adaptability: Unlike neural networks, Random Forest has limited capabilities to adapt to the complex and varied features of medical images, which may restrict its efficiency in contexts where precision is critical.

9.2 Limitations of Neural Network

Computational Resources: Neural networks, particularly ResNet18, require significant computational resources, which can be a challenge for large-scale deployments or in resource-limited environments.

Overfitting: Although more performant, these models are also prone to overfitting, especially when trained on limited datasets without adequate augmentation or regularization.

9.3 Limitations and improvements

We have already seen many points to improve throughout this report (failing segmentation, not deep enough neural networks, preprocessing steps not performant enough). We can mention a few more.

1. Exploration of Hybrid Models

   Combination of Approaches: A promising avenue could be the exploration of hybrid models that combine the strengths of Random Forests (e.g., their robustness to outliers and ease of interpretation) with the ability of neural networks to handle complex data structures.

2. Data Augmentation

   Diversification: To combat limitations related to overfitting and improve generalization, data augmentation could be intensified, using more advanced techniques to simulate a wider variety of pathological cases.

3. Hyperparameter Optimization

   Fine-Tuning: A finer tuning of hyperparameters for each model could potentially improve their performance. Techniques such as grid search or Bayesian optimization could be used to optimize these parameters in a more systematic way.

4. Use of Advanced Deep Learning Techniques

   Deeper Architectures: Testing deeper network architectures or recent variants of ResNet could also provide performance improvements.