Data augmentation for skin pathology classification

SYS800 - Pattern Recognition
Ecole de Technologie Supérieure, Montréal

Louis CARON

December 2019

Contents

1 Introduction

Skin lesions are a common type of disease in the population. While some lesions may be benign, others can cause serious harm if not treated accordingly. Melanomas are an example of such skin lesion. Melanoma is the deadliest type of skin cancer which can be cured with a high survivability rate (98%) if detected in its early stages.

Unfortunately, many skin lesions go unnoticed and are not detected early enough to prevent serious consequences for the patient. In the case of melanoma, the survival rate drops to 63% if the cancer reach the regional state, and it drops to 22% if the tumor has metastasised [2] [1].

It is estimated that around 100 000 new melanoma cases are diagnosed every year in the USA. For most people, the detection of the cancer occurs when seeing a doctor or dermatologist for a routine examination. This need for examination can lead to late diagnostics decrease the survival rate of the patients.

For such reasons, many researches are focused on automating the detection of different skin lesions. The rise of machine learning and especially deep learning during the last decade led to many researches applying classification techniques to medical images.

Seeing a professional dermatologist can be intimidating and scary for some people, and too expensive for others. Using computer vision and pattern recognition techniques on skin lesion classification could enable millions of people to perform regular self check ups. This could potentially save millions of lives by increasing the early detection of dangerous pathologies.

Skin lesion classification is a perfect task for pattern recognition algorithms. Some recent researches [5] [3] [4] [6] have shown that some machine learning techniques can perform as well or even better than human dermatologists when classifying different skin diseases. An algorithm for efficient self diagnosis is therefore possible.

The state of the art models for skin lesion classification are all deep learning models with convolutional layers (CNN). CNN are efficient to learn complex pattern and representations on images, but they are complex and more time consuming compared to other simpler algorithms such as SVM or QDA.

Moreover, deep architectures require a lot of training samples in order to learn the complex features in the images and to accurately classify different images.

In this project for the course SYS800 - Pattern recognition and inspection of the ETS, I have the opportunity to learn about CNN and to experiment with deep CNN architectures. This course project about skin lesion classification is a chance for me to learn more about the machine learning applications and challenges in the medical field.

2 Problem statement

As mentioned previously, the state of the art techniques for medical images classification are CNNs. According to several articles [5] [8] [6] [4], different architectures of CNNs can get a better classification accuracy than human dermatologists.

One of those article especially, from Rezvantalab and al [8], claims to get more than 80% accuracy on skin lesions classifications. In the paper, the researchers compared different CNN architectures for skin lesions classification. My first intuition was therefore to validate their work and compare their models with other models.

While trying to replicate their experiments, I realized I was stumbling on a more general problem. Indeed, CNNs need a lot of data to efficiently learn the complex features of medical images. Unfortunately, annotated medical images of great quality are hard to find. The scarcity of datasets combined with the unbalancing of the data makes it really hard to properly train models.

This lack of big balanced dataset is common to all medical applications. I decided to make this project about handling such datasets.

Having unbalanced datasets can lead to poorly trained models and misleading results. If one of the class represents 90% of the dataset, then a 90% accuracy can be obtained only by classifying everything as said class, which is obviously not what to aim for.

In this project, I will compare different techniques to handle unbalanced datasets and improve the training of models with such datasets. This will be done on skin lesions images, but can later be applied to all medical datasets having unbalanced classes. This could help improve medical images classification in various fields.

In this project, I will try to find ways to efficiently deal with unbalanced datasets through data augmentation, weight-tuning...etc

Since CNNs are the state of the art paradigm for classifying complex medical images, it is more interesting to find methods for handling unbalanced datasets with CNNs. This requires a lot of data in order to properly train the neural model.

We need a lot of samples to train, test and compare different approaches in this project. For those reasons, we will use the HAM10000 dataset [7]. This recent dataset has 10015 annotated images of skin lesions and 7 lesion categories. The categories are heavily unbalanced, as shown in Fig 1.

The dataset has the following categories:

- akiec: Actinic Keratoses (Solar Keratoses) and Intraepithelial Carcinoma (Bowen's disease) are common non-invasive, variants of squamous cell carcinoma that can be treated locally without surgery.
- bcc: Basal cell carcinoma is a common variant of epithelial skin cancer that rarely metastasizes but grows destructively if untreated.
- bkl: "Benign keratosis" is a generic class that includes seborrheic keratoses ("senile wart"), solar lentigo - which can be regarded a flat variant of seborrheic keratosis - and lichen-planus like keratoses (LPLK), which corresponds to a seborrheic keratosis or a solar lentigo with inflammation and regression.

- df: Dermatofibroma is a benign skin lesion regarded as either a benign proliferation or an inflammatory reaction to minimal trauma.
- nv: Melanocytic nevi are benign neoplasms of melanocytes and appear in a myriad of variants, which all are included in our series.
- mel: Melanoma is a malignant neoplasm derived from melanocytes that may appear in different variants. If excised in an early stage it can be cured by simple surgical excision.
- vasc: Vascular skin lesions in the dataset range from cherry angiomas to angiokeratomas and pyogenic granulomas. Hemorrhage is also included in this category.

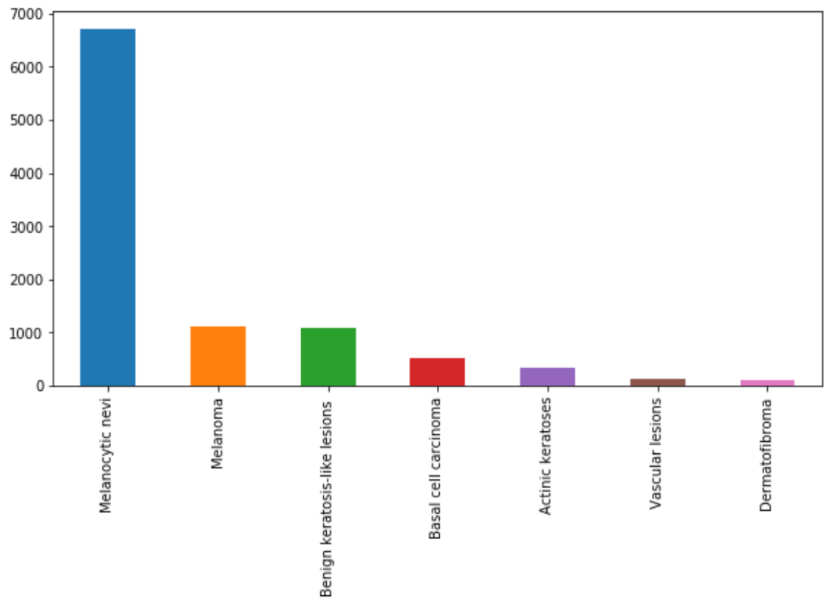

Figure 1: Number of samples per class in the HAM10000 dataset.

We can see that the dataset is heavily unbalanced. Indeed, the majority class "nv" represents 6705 samples whereas the smallest class "df" has only 115 samples. This 60:1 ratio between classes is a problem if we want the accuracy for each class to be equal. Indeed, any classifier would probably just classify everything as the majority class to maximize the accuracy or loss function.

HAM10000 is a perfect example of medical images dataset that are used by researchers to classify different diseases. Being able to deal with unbalanced dataset is therefore critical for such applications. Even outside of the medical field, there are plenty of applications which encounter the same issue, such as automatic fraud detection.

With the HAM10000 dataset, it is crucial to compensate the disparity between the majority and minority classes.

Indeed, any model trying to estimate the distribution of our classes would be completely biased with such a huge difference in number of samples. Moreover, the minority classes have only a few samples, and this may even be too low for properly estimating their distribution.

But how to handle unbalanced data? And is it even necessary with CNNs?
This project aims to answer those questions. In order to do so, I will start by reviewing my main baseline for this project. This baseline compare different CNN architectures for skin lesion classification. I will then discuss the limit of the baseline and propose different methods to improve it by handling the unbalanced data. I will test those methods, compare the results and discuss their advantages and weaknesses.

3 Baselines

Our goal is to find and compare different methods of data augmentation for skin lesion images. In order to do so, we can use previous researches done on skin lesion classification [8] and data augmentation [9] to find different methods that have already been tried.

In their 2018 paper, Rezvantalab and al [8] propose a benchmark of different CNNs architectures such as Resnet152, Densenet201 or VGG16 on skin lesion classification tasks.

The team used the HAM10000 dataset, the same one I will use for this project. For each CNN architecture, they fully trained the deep neural networks and compared their respective accuracy in order to find the best suited model for this task.

According to their results, the best performing models they had tested were Resnet152 and Densenet201. Using those architectures, they managed to get 81% accuracy with Resnet152 and 85% with Densenet201 after a training period of 150 epochs with the HAM10000 dataset.

The results regarding the accuracy of various models are impressive. Indeed, the team state that the Densenet201 and Resnet152 architectures achieved better results than human dermatologists.

What is even more impressive is that the results were obtained without data augmentation. With a heavily unbalanced dataset like the HAM10000, this may hint that the results are not as good as expected since the majority class accounts for 67% of the whole dataset.

Therefore, it could be interesting to study the impact of different data augmentation techniques on the results of a CNN architecture.

Although the first article was a good start to figure out which models were the state of the art, it does not help to deal with unbalanced datasets.

Generally speaking, most methods for such problem can be described as ways to over-represent the minority classes, or under-represent the majority classes. Those methods are called oversampling and undersampling and are described in Fig 2.

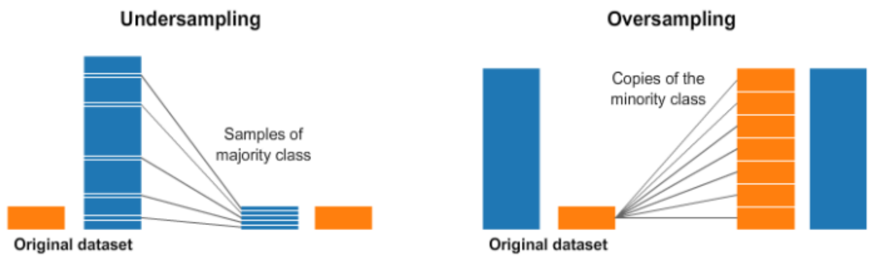

Figure 2: Description of over and under sampling methods.

In the case of undersampling, the aim is to lower each class to the same number of samples. This means discarding some data from the majority classes.

In the case of oversampling, it is the opposite. The goal is to generate new synthetic data to increase the number of samples of the minority classes.

Undersampling has the advantage to limit the resources necessary to train the model, but at the cost of a loss of information about the class. Oversampling, on the other hand, allows to keep all the information of the data. However, the new synthetic data can be correlated to the original samples which can lead to overfitting or poorly trained model. The new data must therefore be created with caution.

In order to conserve as much information as possible, oversampling is recommended for this project.

An article from Stanford University [9] provided an overview of different data augmentation techniques and image generation methods. In this article, the authors compared different methods and experimented with neural approaches and especially GAN (generative adversarial networks).

Although the neural approaches they used were efficient on simple datasets like MNIST and Tiny-imagenet-200, it is hard to say if such approach good perform well on skin lesion classification. Moreover, a far more simple approach was able to get very good results as well.

One of the methods the authors used was the combination of flipping, shifting and rotating original images. This method is one of the oldest one, and yet also one of the most efficient. Indeed, what may seem very similar to the human eye is a completely different image for a CNN.

This method proves to be extremely efficient as well. According to their own article, this traditional approach was almost as performing as neural augmentation techniques, and it was even better suited in some cases.

This simple yet robust method is the state-of-the-art for wide general image augmentation at the moment. Moreover, it corresponds perfectly to skin lesion images. Indeed, skin lesions are invariant to rotations, shifts and even flips. What characterizes a skin lesion is not its angle of rotation for instance. If you flip or rotate a skin cancer image, it stays a valid image of skin cancer that could occur in real life. Note that it is not always the case as some datasets are not suited for image generation through rotation, shift and flip, like MNIST since 6 and 9 are similar with a flip or 180^o rotation.

For vector data, several algorithms exist, like SMOTE and ADASYN [10] which are designed to create synthetic data for minority classes in a feature space. The algorithm can be used to create synthetic vectors for minority classes.

The ADASYN algorithm is particularly interesting as it create synthetic data by estimating the distribution of each class. This means that ADASYN adapt the creation of samples to the distribution of each class and generates data that are "help" classifiers to learn complex features better. Noise can also be added to make the artificial data correspond to reality.

But how does ADASYN generates samples? I will start by explaining the ADASYN algorithm.

Let's imagine a 1D dataset with unbalanced classes. ADASYN starts by calculating the degree of imbalance between each class size and the size of majority class: $d = \frac{N_{minority}}{N_{majority}}$

If the degree of imbalance d is lower than a threshold value, then the algorithm initializes and generates new samples for the class.

Once initialized, ADASYN calculates the number of synthetic samples to create with β the desired ratio between the majority and minority classes:

$$G = (N_{majority} - N_{minority})\beta$$

Since the samples are 1D vectors, they can be seen as points in a multi dimensional space. Using a constant K and for each minority sample x_i, find its K Nearest Neighbors using the euclidean distance. With Δ_i the number of majority samples among the KNN, calculate : $r_i = \frac{\Delta_i}{K}$.

Then normalize the r_i so that it becomes a distribution:

$$r_x = \frac{r_i}{\sum_{j \in N_{minority}} r_j}$$

We can calculate the number of synthetic samples to generate for each datapoint: $g_i = r_x \times G$.

For each minority sample x_i, ADASYN generates g_i synthetic samples. For each sample to create, ADASYN randomly select a minority sample x_u from the KNN of x_i.

Then, the synthetic sample s is generated :

$$s = x_i + (x_u - x_i) \times \lambda \ \ , \ \ \lambda \in [0, 1]$$

The Fig 3 shows ADASYN creating new samples for the minority class in orange.

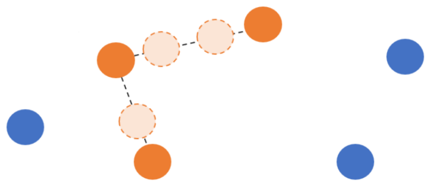

Figure 3: Representation of ADASYN creating synthetic minority (orange) samples.

Note that by using r_x to calculate the neighborhood in which to create the synthetic data, ADASYN generates new samples in the neighborhoods where the majority class is more present. This means that ADASYN creates samples in areas that are "hard to learn" for a model because of the high presence of majority sample. The localization of the new samples makes it easier for the model to learn in such areas. However, this also changes the distribution of the minority class.

The problem is that ADASYN is only suitable in a feature space, and it may seem that it is not suited for image generation.

Despite this, ADASYN remains a very famous algorithm for synthetic data generation. It seems like applying ADASYN to images had never really been attempted before. It could prove interesting to see if such algorithm can be adapted to generate images.

Another mean of treating unbalanced data with CNN is to tune the weights of the neural network. It is indeed possible to give more weight to the samples of the minority classes.

During the optimization of the CNN, it will give more importance to those sample and optimize the network accordingly. This method do not actually create new samples, but it gives more importance to existing ones depending on the proportion of each class in the whole dataset.

There are many ways to deal with unbalanced datasets of images. In this project, I will only focus on the 3 methods mentioned above : traditional image generation, image generation using ADASYN and weight-tuning.

4 Methodology

As mentioned in the baselines, there are plenty of different ways to deal with unbalanced dataset. For this project, the goal is to compare and study the impact of some data augmentation techniques on the performance of state of the art models.

Due to resource and time limitation, I will use only one model in this project. This model is the Resnet152 that was used in the CNN benchmark [8]. According to the researchers, this model is one of the best for skin lesion classification.

Resnet152 [11] is a CNN architecture composed of 152 convolutional layers as described in Fig 4, with a fully connected layer at the end for classification.

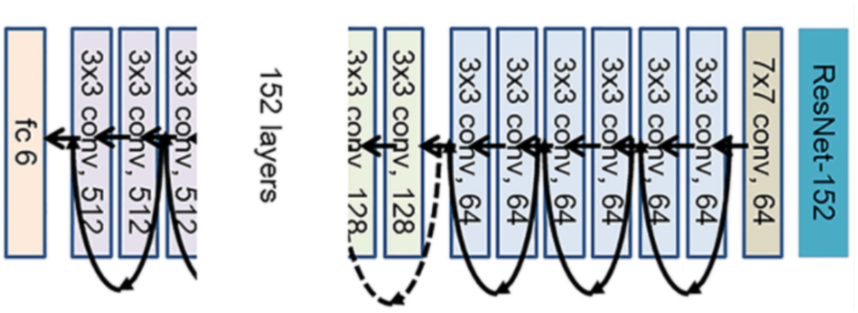

Figure 4: Architecture of Resnet152.

The specificity of a Resnet architecture compared to other CNN is the fact that it has "shortcuts". Shortcuts are links between a layer and another layer which is not located directly next to it. The idea behind it is that the representation of an input can shortcut blocks.

This residual learning does not increase the number of parameters, but it does allow for a greater variety of representation of the information. It was shown to improve the performance of CNNs, and to work well with deep architectures.

In this project, the final layer is removed and replaced with an average pooling, followed by a fully connected layer of 512 nodes and finally the 7 output nodes with softmax activation function. The model will be trained with an adaptive SGD using categorical cross entropy as loss function. In total, the model has almost 60 000 000 parameters.

I will use this model for every experiments in order to obtain comparable results. To study the impact of data augmentation techniques on the model accuracy, I will test each of the aforementioned techniques - weight tuning, traditional image generation and generation using ADASYN - with the model.

The HAM10000 dataset used for this project will be separated into 3 different sets. The training set, used to train the model, will account for 60% of the original dataset. The validation set, used to track the evolution of the model during training and the selection of the hyperparameters, will consist of 10% of the original dataset. Finally, the testing set will contain the remaining 30% of the original dataset.

The data augmentation techniques will be applied on the training set only. Indeed, the goal is to see if they can improve the performance of the model on real data. Therefore, only the training set will be augmented, the validation and testing sets will remain unmodified.

If data augmentation was used on each set, the potential weakness of each technique would affect the test set as well and there would be a risk of overfitting our data. In such case, the result of the experimentation would not represent the performance of the model with real-life data. To avoid this issue, only the training set must be augmented, and the validation and testing set must be kept intact.

The training set will be augmented by the following data augmentation techniques : weight tuning, traditional image generation and image generation using ADASYN. Then, the modified Resnet152 model described before will be trained with the augmented training set, using the validation set to keep track of the loss function and accuracy. Finally, after selecting the best hyper-parameters and verifying that the model is not overfitting, the performance of the model will be calculated on the testing set.

If this methodology is applied, it should provide an efficient, fair and "scientific" comparison between the data augmentation techniques on the HAM10000 dataset. The metrics measured for the comparison will be the global accuracy of the model on the testing set, as well as the loss function with categorical cross entropy.

However, the global accuracy could be misleading when dealing with a unbalanced dataset. Indeed, the testing set will not be balanced and the majority class will therefore represent around 67% of its samples. The model could have a 67% accuracy by classifying everything in the testing set as the majority class.

To avoid this, the accuracy for each class must be compared. This can be done using a confusion matrix to display the accuracy for each class and also display the distribution of the predictions of the model. It would enable the reader to compare the data augmentation techniques class per class, and to calculate the one with the best accuracy all classes included.

To compare my different data augmentation techniques, I will start by validating the result of the baseline and by measuring the performance of the model without data augmentation. This will provide the base on which to iterate from, and allow to compare the data augmentation techniques impact on the model performance.

Then, I will apply weight tuning on the CNN to increase the importance of the minority classes and measure its effects on the accuracy of each class. I will study the impact of this method on the loss function, the model optimization and the overall accuracy.

I will then generate synthetic data using the traditional method of rotating, flipping and shifting images. The model will be trained with this augmented dataset and the accuracy for each class measured, as well as the loss.

The same methodology will be applied for an augmented dataset using ADASYN. The ADASYN algorithm will first be explained in details. Then, a self-made process allowing to use this algorithm for image generation will be created and made explicit. The results of this method, the loss and the accuracy for each class, will be included.

For a more visual understanding, images generated using each method will be shown and compared. This will ensure that the created images correspond to real life images and show the readers the impact of each technique.

Finally, I will discuss the results and provide a final comparison of the data augmentation techniques based on the results of my contribution.

5 Contributions

In this section, I will explain in details my contributions for this project. My contributions include implementing the data augmentation techniques mentioned before, testing them and comparing them, as well as creating a new pipeline for synthetic images generation.

5.1 Validation of the baseline

In order to measure how efficient the data augmentation techniques are, we must first measure the performance of the model without any augmentation. This will allow for a fair and repeatable comparison.

In their 2018 paper, Rezvantalab and al [8] showed that Resnet152 could achieve 81% accuracy on the HAM10000 dataset. After a short discussion with the main author, I learned that this result was achieved in 150 epochs using SGD and categorical cross entropy as loss function.

However, CNNs (Resnet included) are usually heavy and complex models. Although they are described as been the state of the art for image recognition and classification in the literature, they are very demanding in terms of processing power, memory consumption and time. In the case of the HAM10000 dataset, whether or not other classifiers could obtain similar results is not referenced.

In order to verify if the use of CNN is appropriate, I started by testing simpler classifiers with the data.

I experimented with SVM (support vector machine), QDA (quadratic discriminant analysis) and a Resnet152 model from the article [8]. For SVM, I used cross-validation to get the optimum C and γ values.

The results are shown in Table 1.

Classifier	SVM	QDA	Resnet152
Global accuracy	66%	62%	80%

Table 1: Global performance of different classifiers

We can see that the complexity of the data does not allow traditional classifiers like SVM and QDA to get good results. Moreover, the unbalancing of the data greatly affects those classifiers, which are not able to efficiently handle it. Indeed, both classify most of the samples in the majority class, no matter their true label.

The article was therefore right to use CNN in this case, as the overall results of Resnet152 are way better than those of the SVM and QDA.

Using a Resnet152 with a layer of 512 fully connected nodes and 7 output nodes, as explained before, I trained the model. The model was trained on 40 epochs only, as I did not had the processing power necessary for computing 150 epochs quickly enough.

SGD was used as optimizer, and categorical cross entropy as loss function. After 40 epochs, the validation showed increasingly slow improvement. Due to my processing power limitation and the fact the the loss function was starting to slowly oscillate, I decided to only compute 40 epochs.

When predicting the classes of the images in the testing set using the model, we get the confusion matrix shown in Table 2.

Predicted True labels	akiec	bcc	bkl	df	mel	nv	vasc
akiec	0.37	0.21	0.20	0.01	0.05	0.15	0.00
bcc	0.13	0.59	0.12	0.00	0.03	0.12	0.01
bkl	0.03	0.03	0.56	0.00	0.09	0.28	0.00
df	0.11	0.15	0.11	0.19	0.07	0.33	0.04
mel	0.02	0.02	0.10	0.00	0.41	0.45	0.01
nv	0.00	0.01	0.03	0.00	0.02	0.94	0.00
vasc	0.00	0.02	0.00	0.00	0.10	0.15	0.73

Table 2: Confusion matrix of the model without data augmentation, with predicted labels on the horizontal axis and true labels on the vertical axis.

The overall accuracy on the testing set is 80%, and the loss value is 0.55. Those results are coherent with the ones from the article [8], as they are almost identical.

Getting such accuracy without data augmentation questions the importance of such techniques. Indeed, why go through so much trouble to generate new data if we already have 80% accuracy without it?

If we take a closer look at our confusion matrix, we can see that most classes are actually poorly recognised. Indeed, the model classify most of the samples as the majority class.

The model classify the samples belonging to the ”nv” class with 94% accuracy, but at the cost of many false positive for this class. Unfortunately, it also means that the other classes are are not well classified.

Since ”nv” samples account for roughly 67% of the dataset, the model is unbalanced in favor of this class. This 94% accuracy on the ”nv” samples represents 63% of accuracy on the whole testing set. This is why the model gets an accuracy of 80% even though the classification is not performing well in practice. The average accuracy by class is 54%.

Without data augmentation, it is therefore possible to obtain a good overall accuracy on an unbalanced dataset, but the model will not be able to properly classify the samples of each class. Here for instance, the ”df” samples are only correctly classified in 0.19% of the cases, and the model actually label those samples as being ”nv” in 33% of the cases.

It is obvious that such model cannot be deployed for real life applications. This unbalancing of the model must be corrected.

5.2 Weight tuning

In order to counter the negative effects of having an unbalanced dataset, it is possible to improve how the model handle each sample depending on its class. More specifically, it is possible to give more importance to some samples than others during training by *weighting* them in the neural network.

During the training process, samples of the training set are fed in the neural architecture. This single feed forward, provide a loss value for the model. The neural network then applies backpropagation to minimize the loss value. The idea behind it to minimize the loss value by propagating the gradient of the output layer back to the input layer. Gradient descent, or stochastic gradient descent is used to minimize the loss value and update all weights in the neural architecture.

The backpropagation and SGD (stochastic gradient descent) are used to optimize the model, which reduces its loss and at the same time usually improves its accuracy. During this process, each sample has the same importance for the network, meaning that it will try to minimize the loss function equally for all samples.

However, it is possible to decide that some samples are worth more than other during training. By weighting each sample depending on its class, the neural net can give more importance to one sample or another. During training, the network will update its weights, but the update will be weighted depending on the class of the sample.

In a formal way, the loss function assign a greater value to some instances than others. Hence, the loss becomes a weighted average, where the weight of each sample is specified by the weight of its class.

By minimizing the loss function, the optimization of the network grants more importance to some samples. This is useful when handling unbalanced data. In such case, each sample of a minority class can be worth several samples of the majority class for the network.

This can be use to balance the optimization of the model, but it also means that any eventual variance in the few samples of the minority class will have a greater impact on the model. If a sample is of poor quality or has wrong values, the importance given to the samples could increase the variance representation. Moreover, this approach does not generate noise, nor does it create new data, which could lead to overfitting. Indeed, we still have the same samples but each of them counts as several samples of the majority class. We can picture it as having several time the same samples.

This could be problematic when dealing with a dataset such as HAM10000 which has a 60 to 1 ratio between some classes.

To see the impact of weight-tuning on the Resnet152 model, the weight for each class was defined as to represent the ratio between said class and the majority class. This means that the majority class was given a weight of 1, and for each minority class the weight was calculated as :

$$w_i = \frac{N_{majority}}{N_i}$$

with $N_{majority}$ the number of samples of the majority class and N_i the number of samples of the minority class i.

This tuning of the weights should, in theory, ensure that all class have the same importance with regards to the loss function.

With the weights define, the model can be trained on the training set with the chosen weights. Note that the weights are only used during training, and not during validation nor testing.

The Table 3 shows the results on the testing set after training with the defined weights.

The loss value is 1.12 and the global accuracy is 52%. Despite the weight tuning, the performance of the model seem quite disappointing. Indeed, the confusion matrix clearly shows that most samples are not labeled correctly.

Predicted / True label	akiec	bcc	bkl	df	mel	nv	vasc
akiec	0.56	0.02	0.17	0.18	0.03	0.00	0.03
bcc	0.40	0.05	0.06	0.45	0.01	0.01	0.03
bkl	0.22	0.01	0.47	0.07	0.20	0.01	0.01
df	0.44	0.00	0.04	0.41	0.04	0.04	0.04
mel	0.09	0.01	0.30	0.01	0.53	0.04	0.03
nv	0.06	0.00	0.08	0.04	0.24	0.55	0.02
vasc	0.00	0.02	0.00	0.02	0.06	0.00	0.90

Table 3: Confusion matrix of the model with weight tuning, with predicted labels on the horizontal axis and true labels on the vertical axis.

Most classes are below 60% of correct detection, except for the ”vasc” class, but this class already had a good detection rate without weight tuning. The average accuracy by class is 50%, which is even lower than without weight tuning. This technique actually decrease the overall performance of the model.

Since the loss value of the model is quite high, it is safe to assume that the model was overfitting the samples that were given the highest weights. The disparity between classes was so big that some samples were given too much importance. In return, the model over-fitted those samples from the minority classes. The model was unable to estimate the distribution of the classes correctly due to overfitting, which is why the results are so bad.

Weigh tuning is prone to overfitting and high variance. It gets poor results in our case. I would not recommend it for HAM10000, nor for any dataset with a great class disparity and high complexity.

5.3 Traditional image generation

Since the model is overfitting due to the lack of diversity among the samples when using weight tuning, a solution could be to generate synthetic data which would be slightly different from the original ones. This would bring more diversity among our samples and improve the generalization of the model, thus preventing overfitting.

Although this idea seem great, generating data can be tricky. Indeed, the new samples must belong to the minority classes, but be diverse enough to not create overfitting.

Another danger is to generate samples that are too diverse, and therefore cannot effectively represent the distribution of each class. Indeed, if 2 classes are quite similar in reality, then the generation of new synthetic samples may ”overlap” on the other class.

Imagine an image of a zero and an image of the letter ”O”. Both are quite close and their distribution are probably overlapping a lot. If we try to generate a new sample slightly different from an existing one, this sample may end up in the overlapping region of the distributions and confuse the classifier. For instance, my synthetic sample of the letter ”O” may actually be closer to a real-life zero.

The technique used to generate samples must therefore ensure that the generated samples are following the original distribution of their class, or that the samples are representatives of what could be found in real life for their class.

A popular way of generating new samples is simply by shifting, rotating and flipping the original images. Although those can be seen as independent techniques, they are usually applied together [9] and are regarded as one of the most efficient way to generate new data.

Indeed, this method could create an infinity of potential synthetic samples. To the human eye, a rotation of 1^o may go unseen, but to a CNN it is a completely new sample. By rotating, shifting and flipping images, we generate new samples that are different enough of the original ones to prevent overfitting, and at the same time are a coherent representation of their class.

Those synthetic samples are actually generating new information about the distribution of each class. Indeed, the new samples are not necessary following the original distribution of their class. Although generating too diverse samples may be a problem, the new samples are actually representing their class perfectly.

In the case of skin lesion images, rotating, shifting or flipping has no impact. In real life, what characterizes a skin lesion is not its rotation, shift or flip. Applying those techniques to a skin image classification therefore generates a new perfectly valid sample that belongs to the same class as the original image. Skin lesion images validity is invariant to shift, rotation or flips.

This traditional approach can generate new samples that do not follow the original distribution of their class, while guaranteeing that said samples are valid and that the skin lesions they depict could occur in real life. This actually creates new valid information and add it to our data, which modifies the distribution of each class and make it more robust.

This technique also conserve any outliers in our data, as we draw each image following a uniform distribution to perform the augmentation. It means that every image in a minority class is, on average, replicated the same number of times. So the outliers in the data would be conserved.

In this project, the range of the rotation will be of 20^o in any direction (left or right). When oversampling an original image, the original image will be rotated of r degrees, with r following a uniform distribution in the aforementioned range.

The range for the horizontal and vertical shifts will be 20% of the image and the shift value follow a uniform distribution in the selected range. Finally, the images will be randomly flipped vertically or horizontally half of the time.

For every original image, new synthetic samples can be generated by flipping the image with a 0.5 probability of flip vertically and horizontally, then by shifting the image in the range of 20% of its size in any direction, and finally by randomly rotating it in the limit of 20^o left or right.

This enables the creation of a very wide and diverse range of potential new samples. Examples of generated images are shown in Fig 5.

Please note that the purple taint is due to the normalization of the images. Notice how the rotation and shifting distort the image. You can clearly see that the top right image has been rotated and shifted, whereas the top left has only been rotated.

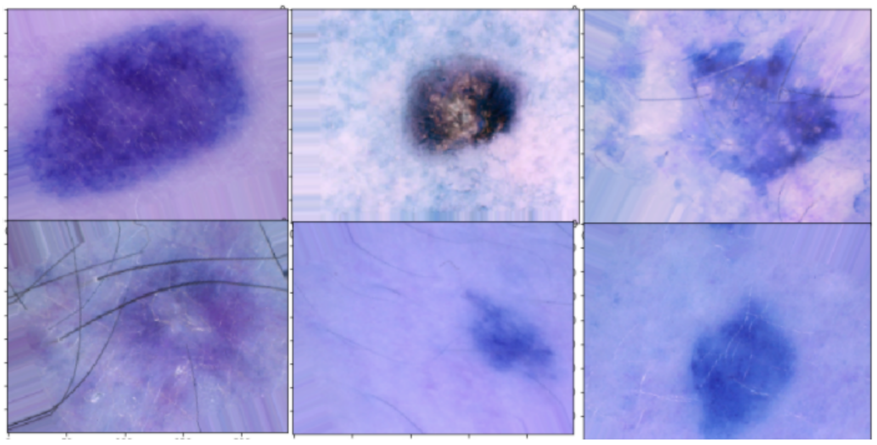

Figure 5: Examples of synthetic images generated with the traditional method.

The data augmentation is performed on the training set, which is then used to train the model. The evolution of the validation loss and accuracy are shown in Fig 6. We can see that the model begins to overfit the data quite quickly. The best validation loss value is 0.79 and the best validation accuracy is 0.79.

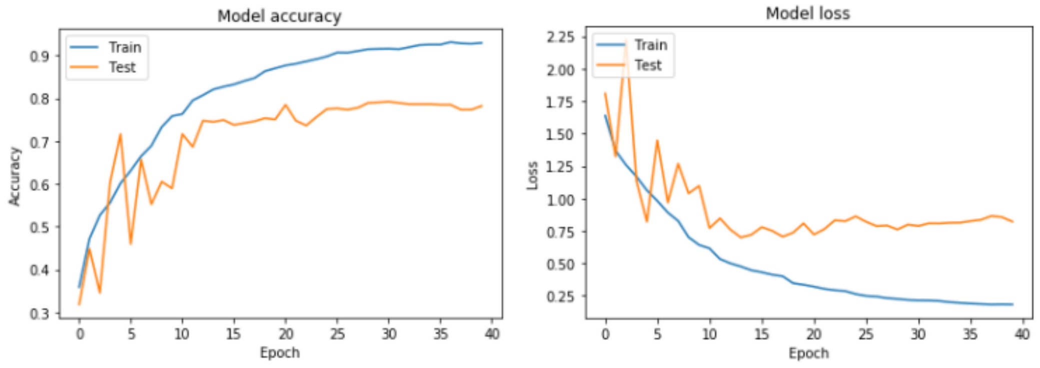

Figure 6: Evolution of the accuracy and loss on the training and validation set during training with the traditional data augmentation.

The model is trained using the best number of epochs according to the validation, and tested on the testing set. The global accuracy is 75% and the loss value is 0.78. Those results are not as good as the ones obtained without data augmentation (80% accuracy).

However, the global accuracy may once again be tricky. As shown by the confusion matrix in Table 4, the predictions of the model seem way less unbalanced than without data augmentation.

The average accuracy by class is 73%, and most classes have an accuracy above 60%. This results are better than the ones obtained through weight tuning or without data augmentation. Indeed,

Predicted / True label	akiec	bcc	bkl	df	mel	nv	vasc
akiec	0.61	0.08	0.17	0.02	0.11	0.01	0.00
bcc	0.10	0.69	0.09	0.02	0.03	0.06	0.01
bkl	0.05	0.03	0.70	0.01	0.15	0.07	0.00
df	0.07	0.04	0.00	0.78	0.00	0.11	0.00
mel	0.01	0.01	0.20	0.00	0.59	0.18	0.01
nv	0.01	0.01	0.07	0.01	0.10	0.79	0.01
vasc	0.00	0.03	0.00	0.00	0.00	0.03	0.94

Table 4: Confusion matrix of the model with traditional data augmentation.

the decrease of the global accuracy from 80% to 75% is due to the unbalancing of the testing set, which contains around 67% of ”nv”. Without data augmentation, the model classifies everything as ”nv” making it right in most of the cases. Here, the testing set remains unbalanced but the model does not blindly label everything as ”nv”, thus the majority class only has a 74% accuracy, which is why the global accuracy decreases even though all other classes are better predicted.

For a patient, it would be way better to be diagnosed by this model than by the previous ones, which are not actually recognising any disease except ”nv”.

Oversampling with this simple yet efficient method can present impressive results. This is probably because the generation of new samples actually improves the dataset by creating samples that are valid yet not necessarily correlated to each others from the CNN perspective. This basically create new correct images. The model can use those images efficiently to become more balanced. The confusion matrix shows that, despite the fact that the global accuracy decreased, the model performs better class by class.

The model does not label everything as the majority class. We can see that besides the ”mel” class, most samples that are not ”nv” are not classified as ”nv”. We can also see that ”vasc” seem to perform extremely well once again. This hints that the ”vasc” class is easy to differentiate from the other class.

The traditional data augmentation performs well in practice on the HAM10000 dataset, reducing the unbalancing effects and allowing for a better trained and more general classifier which treat each class with the same importance.

5.4 Image generation using ADASYN

ADASYN is an algorithm developed in 2008 to generate synthetic samples by adapting to distribution of the majority class in the dataset. ADASYN creates new samples that are different from the original ones, and help make the minority class more visible to the model.

The creation of new samples help to avoid overfitting while at the same time improving the results of the classifier. Unlike weight tuning, it does not not increase the importance given to individual samples, and unlike the traditional data augmentation, it does not generate data that could be unrelated to the original distribution of the classes.

ADASYN is popular for data augmentation. It is a successor of SMOTE, another famous data oversampling algorithm. However, ADASYN, like SMOTE, works for vector (1D) data, and cannot be used on images (multidimensional) data.

In this part of the project, I would like to present a pipeline that uses ADASYN to generate synthetic images. This pipeline is my own creation and it is not based on any existing paper or implementation.

How can ADASYN be applied to images? Since it only works with 1D vectors, the images must be flatten into 1D vectors before going any further. With our images of shape $240 \times 180 \times 3$, we get a vector of length 129 600. Since ADASYN has a complexity of $O(n \times d \times k \times n_m)$ with d the number of dimensions, n_m the number of samples from the minority class and n the global number of samples, it is not wise regarding performance to apply ADASYN right now, as it would take too much memory and time to process the new samples.

To solve this issue, dimensionality reduction is applied on the vectors. Using PCA, we can reduce the length of the vectors to 3000 while keeping 99.8% of the inertia, as shown in Fig 7.

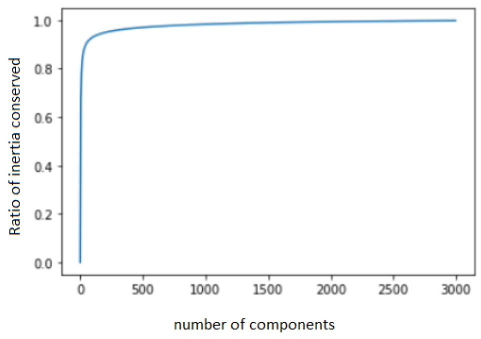

Figure 7: Ratio of conserved inertia per number of components using PCA.

This enables the utilisation of ADASYN without having performance issues. Once ADASYN has generated samples, they must be transformed back into the original space. This is done by using the eigenvectors matrix V and the PCA scores P of the original vectors:

$$X_{reconstructed} = PV^T = X_{original}VV^T$$

This reconstruction looses a little bit of information due to the discarded principal components, but works extremely well as shown in Fig 8.

The dimensionality reduction does not appear to affect the images. It is therefore safe to use and will not loose interesting information about the images.

Once the original images and synthetic samples generated by ADASYN are reconstructed into 1D vectors of 129 600 dimensions, they can then be unflattened into images with the original dimensions.

The Fig 9 display the architecture and organization of the pipeline and how ADASYN is used to create synthetic images.

Using this process, new synthetic samples can be created by ADASYN. It is interesting to see if such synthetic images are looking ”real” or not. Indeed, nothing guarantees that the new samples

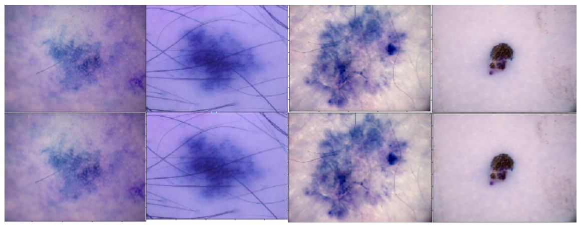

Figure 8: Original images (top row) and reconstructed (bottom row) after PCA of 3000 components.

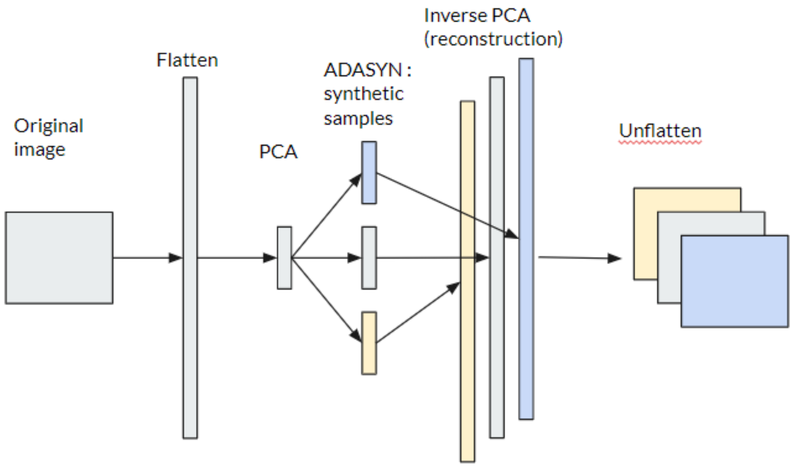

Figure 9: Architecture of the pipeline using ADASYN to generate synthetic samples.

of ADASYN can be transform back into the original space and still make sense and be coherent as images.

We can see in Fig 10 that the generated images seem to be valid, as in they depict skin lesions that could occur naturally in real life. However, this would need to be confirmed by a dermatologist.

It is interesting to see the effect on ADASYN on the generated images. Fig 10 shows that they look just like the other images. Fig 11 shows that the synthetic data are created from an original image which is then "merged" with another image it seems. This correspond exactly to what ADASYN does : creating samples that are an weighted average of two original samples.

After generating synthetic samples so that every class has the same number of samples ($\beta = 1$), the Resnet152 model is trained on the augmented, and balanced, training set. The validation set, which is not augmented, gives us the evolution of the loss and accuracy in Fig 12. We can see that, despite a bad initialization, the model seem to perform well as the validation accuracy and validation loss reach 0.79 and 0.98 respectively, after 22 epochs.

Using the optimal number of epochs, the model gets a global accuracy of 77% and a loss value of 0.86 on the testing set. At first sight, it may seem like a good result, but the accuracy class per

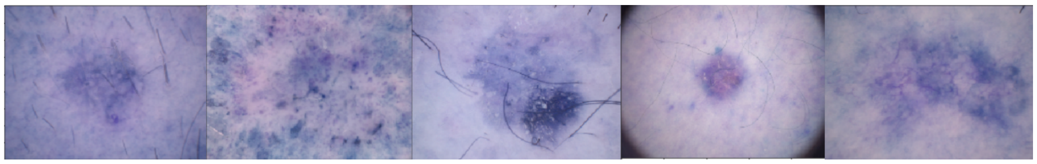

Figure 10: Examples of synthetic images generated using ADASYN with the pipeline.

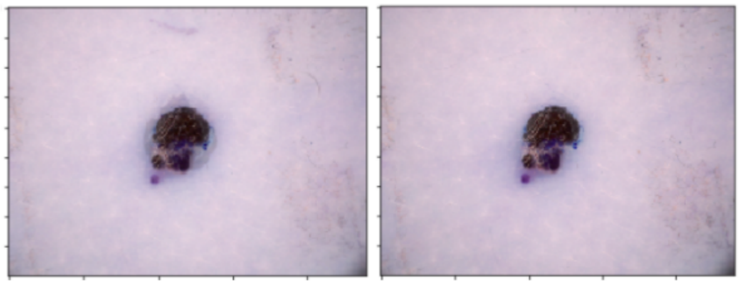

Figure 11: Synthetic image generated with the pipeline (left), and the original image (right).

class is more interesting to study.

The confusion matrix in Table 5 shows that the average accuracy by class is 65%, and that most class have more than 50% accuracy. This is better than the results obtained without data augmentation (54% accuracy by class on average) but not as good as the result obtained with the traditional data augmentation (average accuracy by class of 73%).

Predicted / True label	akiec	bcc	bkl	df	mel	nv	vasc
akiec	0.48	0.08	0.21	0.01	0.16	0.07	0.00
bcc	0.07	0.59	0.14	0.03	0.04	0.14	0.00
bkl	0.01	0.02	0.63	0.01	0.12	0.20	0.00
df	0.04	0.00	0.15	0.63	0.04	0.15	0.00
mel	0.01	0.01	0.10	0.00	0.54	0.34	0.01
nv	0.00	0.01	0.04	0.00	0.08	0.86	0.00
vasc	0.00	0.09	0.00	0.03	0.03	0.06	0.79

Table 5: Confusion matrix of the model with the ADASYN pipeline data augmentation.

We can see in the confusion matrix (Table 5) that some classes are more predicted than others (especially mel, bkl and nv). This decreases the average accuracy of each class. Even the ”vasc” class, which was very well predicted with the traditional data augmentation, only has a 80% accuracy.

The class ”nv” has quite a high accuracy score, which can explain why the model has a high

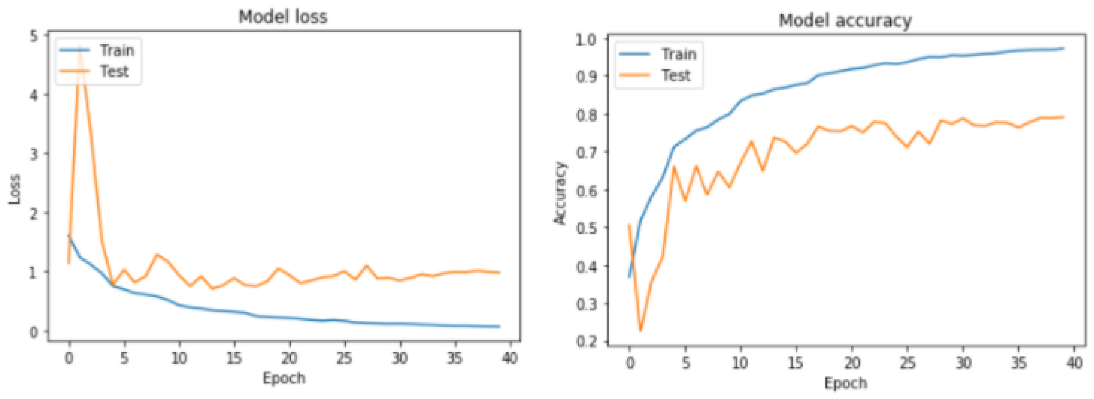

Figure 12: Evolution of the accuracy and loss on the training and validation set during training with the ADASYN pipeline data augmentation.

global accuracy score while not having a great average accuracy by class. The majority of samples in the testing set are ”nv”, which is inflating the importance given to this class on the global accuracy.

The reason for such mitigated performance can be explained by the ADASYN algorithm. By creating synthetic samples that are a weighted average of two other samples, the algorithm may create samples that are not representing what can be found in real life. Indeed, taking the average of some samples of a class does not guarantee that the new samples will be valid.

The ADASYN algorithm can disturb the original distribution of the samples, which can lead the classifier to poorly estimate each class density function. Moreover, the new samples are related to the original ones in their neighborhood, which does not help when estimating their distribution.

In the case of really scarce datapoints, ADASYN creates samples that may not have any relevance. For really complex features, which is probably the case with the HAM10000, ADASYN may not be the best choice as its generation algorithm is quite simple. Using a complex function to generate samples, such as a GAN, may prove wiser and increase the performance of the model compared to a simple algorithm like ADASYN.

6 Conclusion

During this project for the course SYS800 - Pattern Recognition and Inspection of the ETS, I had the opportunity to discover and experiment on new technologies like convolutional neural networks (CNN) and data augmentation techniques.

This project was the occasion for me to compare different data augmentation techniques regarding their performance for skin pathology classification using a Resnet152 architecture.

By replicating the results of my first baseline [8], I discovered that unbalanced data could have a significant negative impact on the performance of models. This showed the need for data augmentation and oversampling techniques.

The global accuracy of a model is indeed misleading when dealing with unbalanced data, and the average accuracy per class can prove to be a better metric.

Data augmentation technique	Raw data	Weight tuning	Traditional data augmentation	Augmentation with ADASYN
Global accuracy	0.80	0.52	0.75	0.77
Average acc. by class	0.54	0.50	0.73	0.65

Table 6: Summary of the results of my experiments.

The Table 6 shows the results for the different techniques tested in this project. Although the raw data are getting a better global accuracy, the average by class is very low and this classifier cannot be used for real life applications. This is due to the unbalancing of the dataset which pushes the classifier to label everything as the majority class.

Weight tuning gets even worse results. Indeed, the number of samples from the minority classes are so low that the model overfit those samples when they are given too much weight. This method is therefore not suited for a heavily unbalanced dataset like HAM10000.

This problem is solved by using oversampling. The traditional state of the art method for oversampling consist in rotating, flipping and shifting original images. This works well on skin lesion images, since their validity is invariant to all these operations. Moreover, the new valid samples do not necessarily follow the distribution of the existing ones, and they can actually bring new information for the model to learn. Traditional data augmentation therefore help to improve the dataset and the estimation the model has of the classes.

Another oversampling method is ADASYN. This algorithm can be used on images by manipulating the images through a pipeline which flatten, reduces, oversamples and then reconstructs the images. This pipeline create new samples that are the weighted mean of 2 other samples from the same class. The new samples are not proven to be valid, and a dermatologist is required to verify this. However, experiments showed that it increased the performance of the model, but not as much as the traditional oversampling technique.

Overall, the best data augmentation technique I have tried remains the traditional method, which can create new valid samples that improves the estimation of the distribution by the model.

Indeed, ADASYN may generate samples that are not valid. Moreover, the new samples are related to the original ones, which can lead to poor classification. Indeed, ADASYN appears not to be complex enough for the task of skin lesion classification.

The weight tuning is not suited for such unbalanced dataset and pushes the model to overfit the few samples of the minority class. Each sample has too much importance, and since no synthetic samples are generated, those few samples are rapidly overfitted.

The traditional method allows to create new sample simply and efficiently, by using the fact that our data's validity (the fact that it may be a real image) is invariant to shifts, rotations and flips. The new images can augment the dataset, and be seen as completely new for a neural architecture. This greatly improves the performance of the model as it can estimate each distribution easily.

For future work, It may be interesting to keep experimenting with the pipeline presented in this project. Indeed, while ADASYN was not suited for such task, using a low dimension representation of an image to create synthetic samples can be achieve through other means, like with SMOTE. Moreover, one could imagine using an autoencoder instead of a PCA to represent the images as vectors.

In order to properly create new data using a fully neural architecture, it may also be interesting to experiment with GAN in order to generate new valid samples.

Overall, this project was a great opportunity for me to learn about computer vision and pattern recognition. When I started the course, I did not know what a CNN was, and I had never worked on computer vision in my life.

During the project, I had the opportunity to learn about CNN, and how to use them. I also learned about how to handle images for pattern recognition applications, and how to apply data augmentation for unbalanced dataset. This knowledge and experience will certainly remain useful to me throughout the rest of my studies and for my future career.

References

[1] American Cancer Society, available at *https://www.cancer.org/cancer/melanoma-skin-cancer/about/key-statistics.html*, last accessed in september 2019.

[2] Melanoma Reasearch Alliance, available at *https://www.curemelanoma.org/about-melanoma/melanoma-staging/melanoma-survival-rates/*, last accessed in september 2019.

[3] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau and Sebastian Thrun. ”Dermatologist-level classification of skin cancer with deep neural networks”. Stanford University, Stanford, California, USA, 2017.

[4] Nabin K. Mishra, M. Emre Celebi. ”An Overview of Melanoma Detection in Dermoscopy Images Using Image Processing and Machine Learning”. Stoecker & Associates, Rolla, MO, USA & Department of Computer Science, Louisiana State University, Shreveport, LA, USA, 2016.

[5] Mahmoud Elgamal. ”AUTOMATIC SKIN CANCER IMAGES CLASSIFICATION”. Institute of Hajii and umra., Um Alqura University, Saudi Arabia, 2013.

[6] Philipp Tschandl, Noel Codella, Bengü Nisa Akay, Prof Giuseppe Argenziano, Ralph P Braun, Prof Horacio Cabo and al. ”Comparison of the accuracy of human readers versus machine-learning algorithms for pigmented skin lesion classification: an open, web-based, international, diagnostic study”. 2019.

[7] Tschandl, Philipp, 2018, ”The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions”, https://doi.org/10.7910/DVN/DBW86T, Harvard Dataverse, V1, UNF:6:IQTf5Cb+3EzwZ95U5r0hnQ== [fileUNF]

[8] Rezvantalab and al, ”Dermatologist Level Dermoscopy Skin Cancer Classification Using Different Deep Learning Convolutional Neural Networks Algorithms”, Azad University, Iran, 2018.

[9] Jason Wang, Luis Perez, ”The Effectiveness of Data Augmentation in Image Classification using Deep Learning”, Stanford University, 2017.

[10] Haibo He and al, ”ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning”, IEEE World Congress on Computational Intelligence, 2008.

[11] K. He, X. Zhang, S. Ren, and J. Sun, ”Deep residual learning for image recognition”, Microsoft Research, 2015.