Dermoscopic Images Challenge - IMA205

Cristian Chávez (Alejandro Chávez on kaggle leaderboard)

Télécom Paris
91120 Palaiseau
chavez@telecom-paris.fr

Abstract. We had a set of images corresponding to skin lesions of 8 different types, which were used to train models that could then predict the classes of another set of test images. Different models of different natures were implemented such as convolutional networks (ResNet, EleNet, DenseNet, ResNext, Inception), nonlinear Support Vector Machines and a combination of both methods. It was found that the best solution was given by the convolutional network model ResNext thanks to its complex architecture that allowed not only to have a good percentage of accuracy but also to make a good generalization that was what allowed to maintain the same percentage of accuracy in the public and private leaderboard.

1 Introduction

Dermoscopy is a non-invasive method that allows in vivo evaluation of colors and microstructures of the epidermis, dermoepidermal junction and papillary dermis not visible to the naked eye. These structures are specifically correlated with histological features. [1] Despite the quality of dermoscopic images that facilitate their study for subsequent classification, it is not a simple task for the human eye because different classes have very similar characteristics, which makes the process difficult. Among our dataset there were images that belonged to 8 different specific 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

We had different number of images of each class as can be seen in the Figure 1.

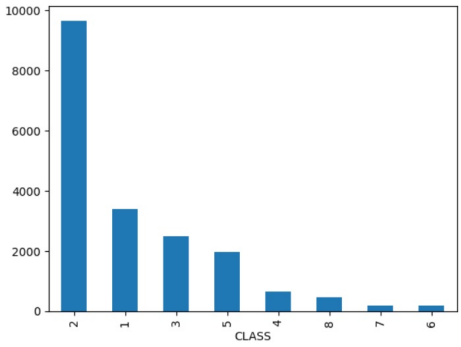


Fig. 1: Number of images per class

This was taken into account in the loss function of the models using the weights given for each class that can be seen in table 1

Class	Weight
1	0.7005531
2	0.24592265
3	0.95261733
4	3.64804147
5	1.20674543
6	13.19375
7	12.56547619
8	5.04219745

Table 1: Weights for each class

In addition, there were also segmentations of some images, as well as metadata correponding to the age, sex and anatomical position of the skin lesion in question that allowed us to have more information in order to find patterns with this data.

For pre-processing we used 2 approaches for segmentation: Otsu and UNet. These 2 approaches were evaluated using the dice-score (comparing the segmentations given by our algorithms with the ground truth initially given for certain images) which showed that the best segmentation was obtained with UNet.

The models applied were Support Vector Machines linear and nonlinear, Random Forest, K-Neighbors and different convolutional networks such as EleNet, ResNet (ResNet101, ResNet152), DenseNet201 and ResNext101 which were trained in some cases with images without segmentation and in other cases with images without segmentation. We also tried to use the output of convolutional networks as input for the Support Vector Machine along with metadata and ABCD

(Asymmetry, the Border irregularity, the Colour and the Dimension of the lesio) features calculated in order to take into account both information

The calculated ABCD features were the asymmetry, compactness, mean and standard deviation of the 3 color channels in the cielab space, area and aspect ratio

All this was done using the Python programming language given its ease to implement these methods using different libraries such as sklearn, tensorflow. It was also decided to use this language to be able to use the pytorch library that allowed the parallelization of convolutional networks and thus be able to train the models with more speed, using the Télécom Paris equipment equipped gpu's that allowed us to use the Compute Unified Device Architecture (CUDA) which is a proprietary parallel computing platform and application programming interface (API) that allows software to use certain types of graphics processing units (GPUs) for accelerated general-purpose processing, an approach called general-purpose computing on GPUs. [2]

2 Pre Processing

2.1 Segmentation

We implemented the Otsu and UNet segmentations that were later evaluated using the metric says score which were calculated using the ground truth masks given for some images, the matching percentage was around 60% and 80% respectively. It was therefore decided to use UNet.

I think it was better to have used UNet segmentations not only for the dice score but also because while Otsu is a quick technique compared to UNet to do segmentation, is also a fairly simple technique considering the variety of shades and shapes of skin lessions. This complexity of the images to be segmented may be better worked by UNet than by Otsu since Olaf Ronneberger and his team developed U-Net in 2015 for their work on biomedical images. It won the ISBI challenge by outperforming the sliding window technique by using fewer images and data augmentation to increase the model performance. [3]

U-Net gets its name from its architecture. The "U" shaped model comprises convolutional layers and two networks.

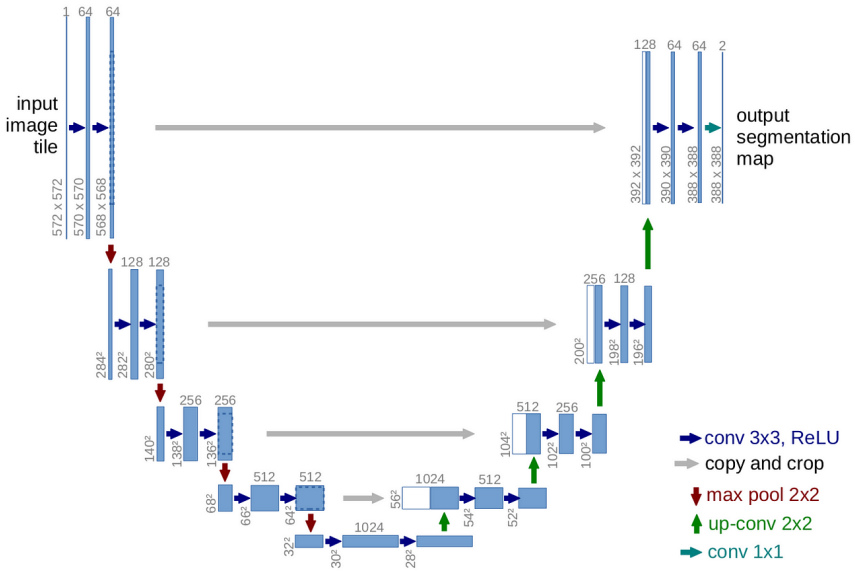


Fig. 2: UNet U architecture

This architecture is basically composed of an Encoder and a Decoder since we want an output of the same size as the input but we want to change the input in the process.

Segmentation tries to response to the answer of 'what' and 'where' of objects to be segmentated, the encoder is in charge of the 'what' that is to 'recognize' the objectwe want to segmentate.

This encoder network consists of 4 encoder blocks. Each block contains two convolutional layers with a kernel size of 3*3 and valid padding, followed by a Relu activation function. This is inputted to a max pooling layer with a kernel size of 2*2.

The decoder network tries to solve our second question-"where" is the object in the image? It consists also of 4 decoder blocks.

What makes U-Net so good at image segmentation is skip connections and decoder networks that differentiate this architecture from the other convolutional neural networks. Skip connections help us use the contextual feature information collected in the encoder blocks to generate our segmentation map. [3]

The source code for this implementation was inspired from the analytics vidhya post about UNet on segmentation for biomedical images.

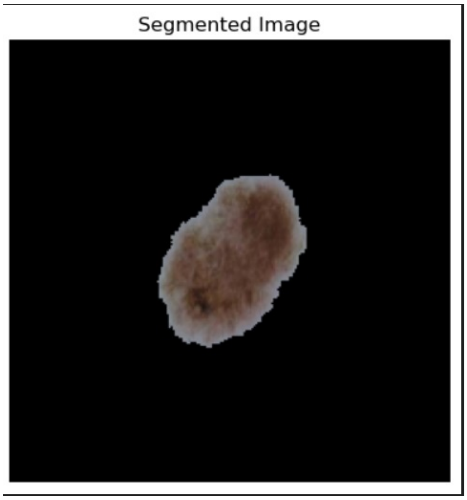


Fig. 3: Segmentations example

3 Features Extraction

The ABCD rule is the first dermoscopy algorithm created to help differentiate benign from malignant melanocytic lesions. This algorithm, described by Stolz, is based on multivariate analysis of four criteria (asymmetry, border, colors and different structures) with a score system [4]

In total 10 features were extracted corresponding to the asymmetry, compactness, mean and standard deviation of the 3 color channels in the cielab space, area and aspect ratio.

The assymetry of the lesion was calculated by mirroring the left half of the image and computing the Structural Similarity Index (SSIM) between the original and mirrored halves. The SSIM measures the similarity between two images, with a value of 1 indicating perfect symmetry.

The border irregularity was calculated computing the compactness of the lesion's border. First converting the image to grayscale and applying Canny edge detection to detect the lesion's border. Then, the contours of the lesion were found and the perimeter and area of the largest contour was calculated. The compactness is calculated as the square of the perimeter divided by the area.

The mean and standard deviation of the channels are statistics that capture the color variation within the lesions.

For the dimension the area and aspect ratio of the lesion was calculated. The area of the lesion was computed by summing the number of non-zero pixels in the binary image representation. Then, the bounding box of the lesion was estimated and also the aspect ratio as the width divided by the height.

4 Classification Methods

4.1 Support Vector Machines

A support vector machine (SVM) is a machine learning algorithm that uses supervised learning models to solve complex classification problems by performing optimal data transformations that determine boundaries between data points based on predefined classes, labels, or outputs. SVMs are widely adopted across disciplines such as healthcare, natural language processing, signal processing applications, and speech and image recognition fields. [4] That is why it was implemented giving it as input the abcd features calculated.

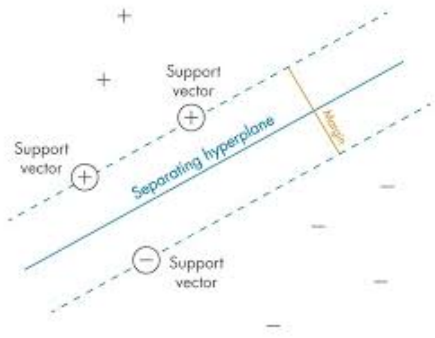


Fig. 4: Support Vector Machine

An implementation of Grid Search Cross Validation was performed with the following different values for the parameters

Parameter	Values
C	0.1, 1, 10, 100, 1000
γ	0.001, 0.01, 0.1, 1, 10

Table 2: Parameter Grid for SVM

Where C is the regularization parameter and gamma the kernel coefficient for the non-linear SVM

At the beginning i wanted to test how well a linear SVM could work with the features. It did not give a good result using the accuracy metric since it only arrived to 10% of accuracy meaning that the feature space was more complex, a hyperplane in the feature space could not perform properly the task, so that, the implementation of a non-linear SVM was performed since it allowed us to

calculated a hyperplane of a feature space with higher dimensions, being able to deal with more complex behaviour.

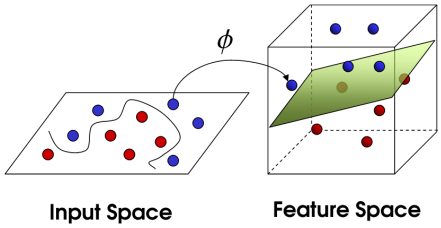


Fig. 5: Non linear SVM

The Radial Basis Function kernel was used

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

The best parameters found using the this kernel where C=10 and gamma=1

Parameter	Value
C	10
γ	1

Table 3: Best Parameters for RBF Kernel

Fitting 5 folds for each of 25 candidates. 125 folds were fit in total to find this hyperparameters. To deal with the unbalanced dataset towards the class of label 2, the weights given for each class were passed to SVM library of sklearn so that it was taken into account in the loss function.

However, the accuracy on the leaderboard was not that good achieving 23,1% in the public score and 22,4% in the private score.

Score Type	Accuracy (%)
Public	23.1
Private	22.4

Table 4: Accuracy Scores on Leaderboard of SVM

This is possibly explained to the necessity of more features required from the SVM, since the images are too complex to differentiate since they rely on small changes in their appearance and shape to be differentiated.

4.2 Random Forest

A similar procedure was done to implement Random Forest, however, it got worst results in the leaderboard achieving 20,9% in the public score and 20,6% in the private score.

Score Type	Accuracy (%)
Public	20.9
Private	20.6

Table 5: Accuracy Scores on Leaderboard of Random Forest

Again, a similar explanation is given, probably the lack of more information did not allow to have a better score of this method

4.3 Convolutional Neural Networks models

Among all the different architectures of Convolutional Neural Networks implemented there are common characteristics:

They were implemented using the pytorch libraries that allowed their implementation and subsequent parallelization to use the CUDA capabilities given by the Télécom Paris gpu's.

All architectures used different transformations to be able to do data augmentation such as: Horizontal Flip, Vertical Flip and Color Jitter.

All images were applied a gaussian blur with mean 5 and a standard deviation between 0.1 and 2.0 in order to make the model more general

A 260x260 resize was applied to all images as networks need a fixed number of inputs

All methods were applied drop out with different proportions, different optimizers such as SGD, Adam and Adamax were used, two different types of stepsize, constant and dynamic (SetpLR and Step Plateau), weight decay was applied.

The defined loss function selected was the Cross Entropy since we were facing a method of classification of multiple labels. The weights given for each class were used in the loss function in order to also take into account the imbalance between classes.

The different tested values for the different parameters and hyperparameters were:

Parameter or Hyperparameter	Values
Learning Rate	1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1}
Batch Size	16, 32, 64, 128
Split	0.15, 0.2, 0.25, 0.3
Weight Decay	1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1}
Dropout Percentage	0.2, 0.3, 0.4, 0.5

Table 6: Parameters and Hyperparameters for Model Training

In each model implemented, the best values of these parameters and hyperparameters will be given according to your score in the private leaderboard

DenseNet The two most recent versions of DenseNet where implemented, namely, DenseNet101 and DenseNet152 through their respective library¿ies from pytorch. Densely Connected Convolutional Networks also known as DenseNet are Convolutional Neural Networks (CNN) architecture whose principle is based in the linking of each layer with every other layer as shown in Figure 5:

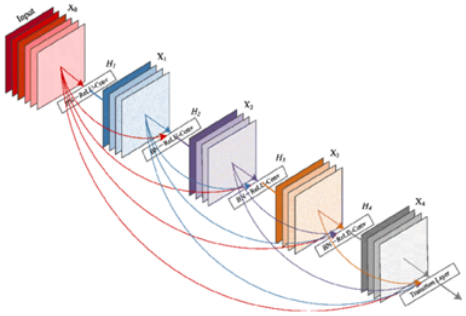


Fig. 6: DenseNet network architecture

Its architecture is made by transition layers and dense blocks. Convolutional layers are inside the dense blocks, which allows each layer to access the information of the previous layers by connecting the output of the layer with the input of the following layer generating a “shortcut” link. There are some transition layers that minimize the size of the feature maps between dense blocks.

Each architecture consists of four DenseBlocks with a varying number of layers. For example, the DenseNet-121 has [6,12,24,16] layers in the four dense blocks whereas DenseNet-169 has [6, 12, 32, 32] layers. We can see that the first part of the DenseNet architecture consists of a 7x7 stride 2 Conv Layer followed by a 3x3 stride-2 MaxPooling layer. And the fourth dense block is followed by a Classification Layer that accepts the feature maps of all layers of the network

to perform the classification. Also, the convolution operations inside each of the architectures are the Bottle Neck layers. What this means is that the 1x1 Conv reduces the number of channels in the input and the 3x3 Conv performs the convolution operation on the transformed version of the input with a reduced number of channels rather than the input.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Fig. 7: DenseNet layers

This connection makes the network more efficient as it reuses features, reducing the numbers of parameters and improving the gradient flow. DenseNet's state-of-the-art performance can be observed in a range of computer vision tasks including picture classification, object recognition, and semantic segmentation.

The DenseNet design successfully tackles overfitting by lowering the number of parameters and enabling feature reuse, enhancing the model's capacity to generalize to unknown data.

We could see an improvement in the score of leaderboard having now score scores above the 50% for different combinations of the parameters

The best values for this model were:

Parameter or Hyperparameter	Value
Learning Rate	1×10^{-2}
Batch Size	64
Split	0.2
Weight Decay	1×10^{-4}
Dropout Percentage	0.5

Table 7: Values for DenseNet

Achieving the following scores:

Score Type	Accuracy (%)
Public	61.7
Private	58.4

Table 8: Accuracy Scores on Leaderboard of Dense Net

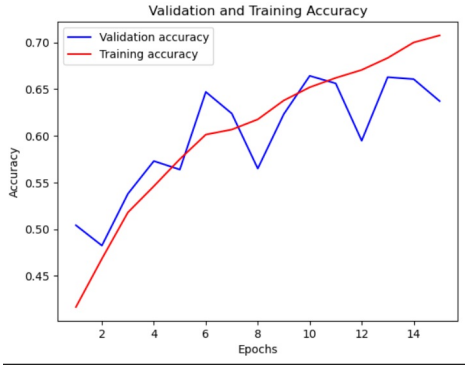


Fig. 8: DenseNet validation and training accuracy over epochs

DenseNet got better results than ResNet probably because is an architecture desgined to handle more complex data, this behaviour was expected

ResNet The last version of ResNet, namely ResNet101 (the number is given by the number of layers used in the architecture) was implemented through the its library from pytorch.

The bigger the number of layers in the architecture of a CNN, the bigger the problem of vanishing/exploding gradients.

ResNet architecture is different in its implementation of residual blocks. The Residual Blocks idea was created to address the issue of the vanishing/exploding gradient. We apply a method known as skip connections in this network. The skip connection bypasses some levels in between to link-layer activations to subsequent layers. This creates a leftover block. These leftover blocks are stacked to create resnets. The strategy behind this network is to let the network fit the residual mapping rather than have layers learn the underlying mapping. Thus, let the network fit instead of using, say, the initial mapping of H(x)

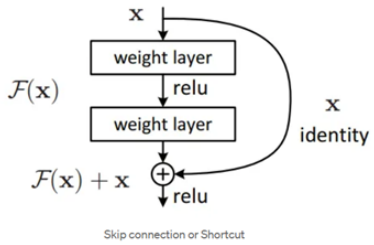


Fig. 9: ResNet skip connection idea

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 10: ResNet architecture

Through combining multiple-sized convolutional filters, ResNet can be faster in the training process, making it a suitable model for training with huge datasets.

ResNet becomes the Winner of ILSVRC 2015 in image classification, detection, and localization, as well as Winner of MS COCO 2015 detection, and segmentation. This is a 2016 CVPR paper with more than 19000 citations. [5]

Given these two main features of ResNet, its residual blocks idea and its faster training process, this model was implemented. [6]

The best values for this model were:

Parameter or Hyperparameter	Value
Learning Rate	1×10^{-3}
Batch Size	64
Split	0.25
Weight Decay	1×10^{-2}
Dropout Percentage	0.3

Table 9: Values for ResNet

Achieving the following scores:

Score Type	Accuracy (%)
Public	59.9
Private	65.1

Table 10: Accuracy Scores on Leaderboard of ResNet

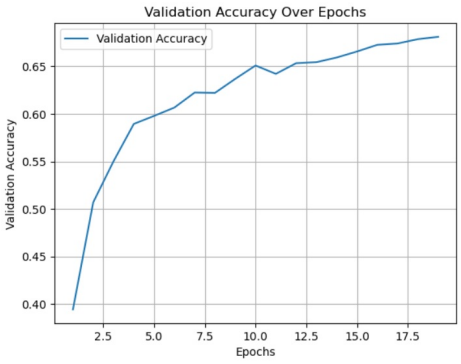


Fig. 11: ResNet validation accuracy over epochs

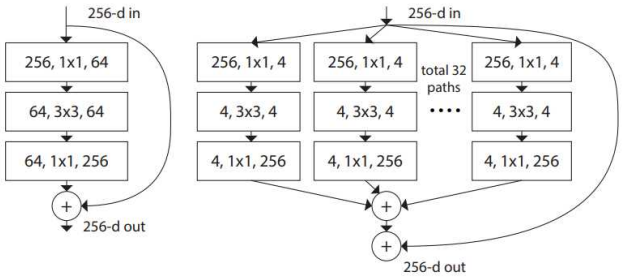
ResNeXt ResNeXt stands for Residual Networks with Aggregated Transformations. It's a type of CNN that was introduced to address some of the limitations found in traditional CNN architectures.

Like ResNet, ResNeXt applies the concept of residual connection to avoid the problem of vbanishing gradient. The key innovation in ResNeXt is its use of "cardinality" [7]

The main goal of developing ResNeXt was to improve the performance without needing to increase the number of computations. Cardinality refers to the

number refers to the number of parallel transformation paths within a network layer making it a different strategy from the traditionals where we wheter increase the number of layers or the number of units per layer which allow to handle more complex data without making the network more complex to handle.

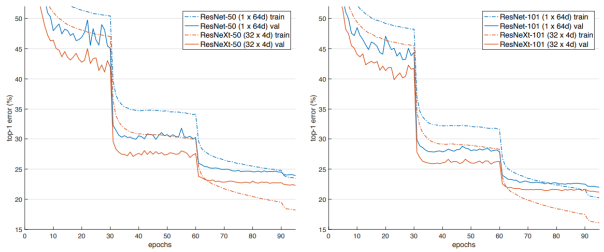
The input is first split into different paths that are gonna represent different transformations but with the same parameters. Being able to extract different information without making it much more complex.



Left: A block of ResNet. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

Fig. 12: ResNeXt input split

We can see that the cardinality concept is more efficient than width (number of neurons in the same layer)



Training curves on ImageNet-1K. (**Left**): ResNet/ResNeXt-50 with preserved complexity (∼4.1 billion FLOPs, ∼25 million parameters); (**Right**): ResNet/ResNeXt-101 with preserved complexity (∼7.8 billion FLOPs, ∼44 million parameters).

Fig. 13: Cardinality vs Width

Parameter or Hyperparameter	Value
Learning Rate	1×10^{-3}
Batch Size	32
Split	0.15
Weight Decay	1×10^{-2}
Dropout Percentage	0.2

Table 11: Values for ResNeXt

Achieving the following scores:

Score Type	Accuracy (%)
Public	73.1
Private	73.1

Table 12: Accuracy Scores on Leaderboard of ResNeXt

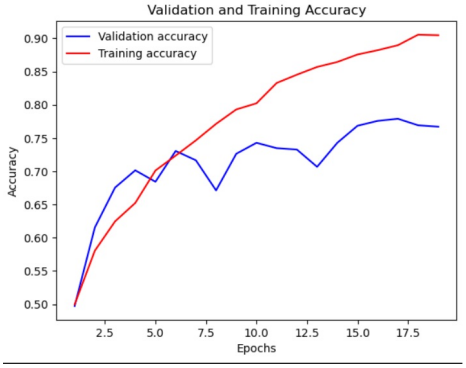


Fig. 14: ResNeXt validation and training accuracy over epochs

ResNeXt got better results than the the 2 other CNN's because it has the best of both models and also adds the concept of cardinality, being able to take more attention to the details.

The batch size was important, if it was too small, it paid too much attention to the details but not to the image in general, if it was too big then it would not pay attention to the image in general, that's why it was chosen a middle term

5 Conclusions

Implementing traditional methods such as Random Forest and Support Vector Machines (linear and non-linear) along with models based on convolutional neural network architecture, We can see firsthand how history repeats itself that has been said so much since CNN appeared: Convolutional neural networks are generally better than our traditional methods before them.

I consider that this improvement of CNN is thanks to the universal approximation theorem: any continuous function f : [0, 1]n [0, 1] can be approximated arbitrarily well by a neural network with at least 1 hidden layer with a finite number of weights. Although the theorem does not specifically indicate how we should fix our architectures depending on the tasks, for now, it is more a task of trial and error, of using models based on their results in the face of equal or similar problems.

This is a problem since the training time is not small even using the parallelization tools given by pytorch using the opportunities given by CUDA. So you have a great feature space to be able to test the different parameters and hyperparameters of the different CNN's models.

However, thanks to advances in supervised and unsupervised learning theory we know different techniques for different problems that may arise, for example, the effectiveness of the drop out method, the weight decay method was evidenced, method of regularization when there is an overfitting in the model or also, in cases where we have an inbalanced dataset (which was our case) we could apply data augmentation techniques in such a way that the dataset is less inbalanced, we were able to include the weights of each class in the loss function so that these imbalances were taken into account in the training process itself.

ResNext proved to be the best choice among the different architectures created, using different techniques to improve its prediction generalization capability such as early stopping, drop out, weight decay and regularization along with data augmentation. This is thanks to its concept of 'cardinality' that allowed to extract more information without the need to complicate the calculation processes and at the same time allowing the possibility of parallelizing the calculations and thus to test more combinations of parameters and hyperparasites.

The good generalization of this model is evidenced in that the model did not change its accuracy percentage between the public and private leaderboard score: 0.731 in both cases. Which allowed it to climb to 7 position in the leaderboard of the private score.

Model	Public Score (%)	Private Score (%)
SVM (RBF Kernel)	23.1	22.4
Random Forest	20.9	20.6
DenseNet	61.7	58.4
ResNet	59.9	65.1
ResNeXt	73.1	73.1

Table 13: Comparison of Implemented Models

6 Bibliography

References

1. Medscape Reference (2023). Dermoscopy: Overview, technical procedures and equipment, color. Retrieved from [Medscape eMedicine](https://emedicine.medscape.com/article/1130783-overview)
2. Nvidia. (n.d.). CUDA Toolkit. Retrieved from https://developer.nvidia.com/cuda-toolkit
3. Analytics Vidhya. (2022, October 10). Image segmentation with U-Net. Retrieved from https://www.analyticsvidhya.com/blog/2022/05/introduction-to-image-segmentation/
4. Dermoscopedia. (n.d.). ABCD rule. Retrieved from https://dermoscopedia.org/Melanoma
5. Towards Data Science. (2019, June 17). Review: ResNet - Winner of ILSVRC 2015 (Image Classification, Localization, Detection). Retrieved from https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8
6. National Institutes of Health. (n.d.). A deep learning framework for automatic segmentation of dermoscopic melanoma lesions. [PMC](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8880650/)
7. Ikomia. (2021, March 17). ResNeXt CNN: Cardinality and efficiency explained. Retrieved from https://www.ikomia.ai/blog/resnext-cnn-cardinality-efficiency-explained