Convolutional Neural Networks for image processing

Daniel Palacios Hidalgo

Abstract— With the evolution of Artificial Intelligence, Machine Learning algorithms and Computer Vision techniques, several researchs have focused on the application of well known algorithms to obtain non-common results. In this paper related to the Final Degree Project of Daniel Palacios Hidalgo under the supervision of Ramón Baldrich and with the support of the Autonomous University of Barcelona, the original Style Transfer Algorithm, a Denoising Convolutional Network and a Deep Dreaming network will be exposed in depth. The main purpose of this project is the understanding of how they work, how to implement them and finally, evaluate their results to compare them against the original implementations of their respective authors using Pytorch framework.

Index Terms— Artificial Intelligence, Computer vision, Machine Learning, Convolutional Neural Networks, Deconvolutional Neural Networks, Style Transfer, Image Denoising, Deep Dreaming, Torch, PyTorch.

1 Introduction

Image processing considers the manipulation and analysis of images. This processing refers to apply transformations and restauration techniques to improve the quality of the images. The analysis consists in the extraction of features and properties of the images to classify, identify or recognize patterns. With the recent evolution and improvement in hardware architectures, nowadays computers can execute algorithms and complex computation structures that were impossible to execute 20 years ago in a personal computer. These facts brings us out the chance to apply those complex algorithms in our home computers to make study cases and understand the way they work in such a deep way.

1. 2 Objectives

This project has two groups of objectives. The first group of objectives are related to the professional skills set that a Computation Engineer should achieve, those objectives are:

- C3: The skill to avaluate the computational complexity of a problem, know algorithmic strategies to solve the problem and implement them.
- C4: The skill set to know the fundamentals, paradigms and techniques of artificial intelligence systems and how to build them.
- C6: The skill set to build interactive systems to present the information.
- C7: Develop the necessary skill set to design and implement systems that use machine learning techniques.

The second group of objectives of this project are related to the understanding of how to interpretate and to implement different research papers around computer vision, machine learning and convolutional neural networks. Those objectives are the following:

- Implement Deep Learning algorithms from zero using the original papers.
- Implement a Style Transfer network.
- Implement a Denoising Network.
- Implement a Deep Dreaming Network.
- Implement a user interface to handle easily the execution of the previous algorithms.

These objectives are chosen to allow to the student to acquire general knowledge in leading artificial intelligence technologies and image processing and at the same time, acquire autonomy to interpretate previous researches performed in the chosen area.

1.3 Methodology

This project is developed under an Agile methodology. There have been two stages, in the first stage all the efforts were focused in the research, design and implementation of the Style Transfer algorithm and the Denoising Network. In the second stage, the objectives were centered in developing and implementing the deep dreaming algorithm and the user interface.

In the beginning of every stage, an analysis of the to do tasks is done to planify the temporal resources. At the end of every stage, there will be an analysis to understand the key points that worked or didn't work to plan the next stage in a more efficient way.

1.4 Document structure

This document begins with a brief introduction to the project, its objectives and the methodology used along the project. Following the introduction section, the image

- *Contact e-mail:Daniel.PalaciosH@e-campus.uab.cat*
- *Major: Computation*
- *Treball tutoritzat per: Dr. Ramon Baldrich (CVC)*
- *Course: 2017/18*

processing section contains several subsections with an in-depth view of the different algorithms implemented with their related results. To close this report, a conclusions section will cover the main points in a summarized way, suggestions to the reader to follow a path from this project on, the aknowledgements and the bibliograpgy where this project relies on.

2 IMAGE PROCESSING

2.0 State of Art

The algorithms exposed along this project are not new algorithms. Some of them were developed in 2014 and have been optimized along these years. This is the case of Style Transfer which was originally designed by Leon A. Gatys in his paper [1]. Along these years, several optimizations [2] have improved the performance of this algorithm getting speed ups up to 1000x. Those improvements allowed bringing this algorithm to mobile devices in applications such as Prisma and getting quasi-instant results.

In the case of image denoising, the first approach purposed by ZHAO, Aojia was done in 2013. Although this algorithm had good results, in January of 2018 a new research in image restoration field using deconvolutional neural networks DNNs [5] was performed with highly relevant results that showed promising results for various image restoration tasks.

Finally, Deep Dreaming networks were firstly presented by Google Developing Group in 2015 [7]. The efforts since its presentation have focused in understanding in depth how does artificial neural networks store the information in its layers and why that information [8].

Regarding the technologies and frameworks that eases Deep Learning developments, there are several elections that might be considered:

TensorFlow
TensorFlow is a blend between lower computation libraries like Theano and higher level computation libraries such as Blocks or Lasagne. Although it is one of the newest frameworks, it has earned a huge reputation because of being backed by Google Brain team. Its pros fall in have a large active community, low level and high level interfaces to network training, faster model compilation and cleaner multi-GPU support. Its cons fall in being slower in the benchmarks than Theano and its hardness in its learning curve.

Theano
Theano is a numerical computing framework that powers many other deep learning frameworks. Theano works at low level. In its pros, we can consider its flexibility to work in almost any deep learning project and his high performance. In its cons, we must consider its substantial learning curve, his low speed at compiling graphs and the need of a substantial learning expertise to write effective code.

Keras
Keras is probably the highest level, must user friendly library. It allows to the developer to choose whether the models they build are executed on Theano's or TensorFlow's. Keras community is quite large and active and TensorFlow team announced a future integration of Keras as a subset of TensorFlow project. Regarding its pros, Keras grants a easier learning curve and an intuitive high level interface. Its main coin is its lower flexibility than other frameworks because of its high-level interface.

PyTorch
Pytorch is a port of Lua's Torch library to Python. Pytorch provides tensor computation with strong GPU acceleration and Deep Neural Networks built in. It is backed by Facebook Artificial Intelligence Research Team. In its cons, we might consider its philosophy "Python First", making it easy to integrate into Python Ecosystem. Also, it is a blend of high level and low level features, allowing to perform fast built-in functionalities or customize needed functionalities. In its cons, it is one of the newest frameworks and it is less mature than other alternatives. Also, references outside the official documentation are very limited.

In this project, PyTorch will be used because of its capacity to perform good at low level and high level computation. Also, because of the image processing nature of the project and its capacity to integrate easily with other Python libraries such as NumPy or PIL. Moreover, taking into account that this project is an academic project to understand and to learn Deep Learning, its learning-curve and difficulty its more suitable than other possible elections.

2.1 Style transfer

Style transferring algorithm was first proposed by Leon A. Gatys [1]. This algorithm consists in transferring the style of an image (S) to another image (B).

2.1.1 Algorithm explanation

To perform this operation, we should use a distance measure to know in every step of the algorithm how different is the result image from the style image S and the base image B. The original algorithm, the implementation here[3] and the implementation performed in this project, uses as a distance mesure the Mean Squared Error formula (MSE) such that:

$$d(I,B) \rightarrow \delta_1 \; AND \; d(I,S) \rightarrow \delta_2$$

Where:
d is the distance measure,
I is the result image,
B is the base image,
S is the style image,
δ_i *is a threshold*

Style Transferring is a 2-variable optimization problem. Because of the process takes in consideration two imputs, we must be able to regulate the weight of those inputs to control how much style or base should be applied to the result image. This can be performed calculating a weighted total MSE such that:

$$MSE_{total} = \alpha MSE_{style} + \beta MSE_{base}$$

Where:

α is the style weight,
β is the base weight

Where alpha and beta are the weight regulators.

In specific terms, the behaviour of the algorithm[Fig. 1] consists in, assuming that both input images are NxM pixels with 3 RGB color channels and given a pretrained convolutional network, both images must be forwarded through the network and its output must be captured to perform a weighted loss calculation on and an optimization on this loss to accomplish the transference of style effect.

To capture style features independently from the content features, it is convenient to apply a Gram Matrix[4]. The terms of this matrix are proportional to the covariances of corresponding sets of features, and thus captures information about which features tend to activate together. By only capturing these aggregate statistics across the image, they are blind to the specific arrangement of objects inside the image. This is what allows to capture information about style independent of content.

The highest computation costs relies on the optimization process. A normal hill climbing algorithm to find out the minimum values would be computationally-expensive due to the size of the images. To minimize the impact on the total time execution, the chosen algorithm is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. This algorithm is an iterative quasi-Newton method for solving unconstrained nonlinear optimization problems. PyTorch includes a variant of this algorithm called Limited Memory BFGS.

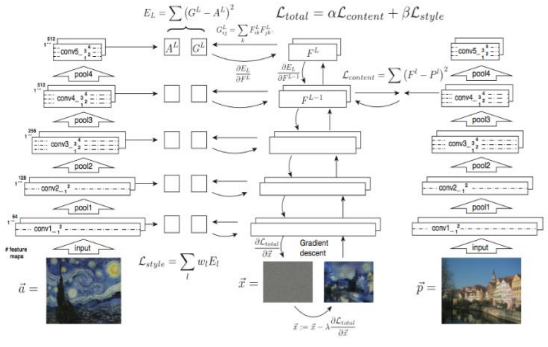


Fig. 1: Style Transfer algorithm scheme.

2.1.2 Convolutional models

Style Transferring algorithm works on an image generated by a convolutional network. There are several configurations of convolutional networks used for this task based on VGG[Fig. 2], Inception and Xception, but according to the original research paper where Leon A. Gatys uses a VGG-19 architecture with 16 convolutional layers, in this implementation we will use a VGG-16 model D where all the operations are performed over 3D convolutions. The differences between a VGG-16 and a VGG-19 are subtle and non-recognizable for a human eye. For solve this task, we only need the upper layers of the network to gather all the information of the input related to general colours and/or shapes. The lower layers can perform better for classifying and detection tasks which are not related or necessary for Style Transferring process.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Fig. 2: Different VGG architectures scheme.

2.1.3 Results

Due to the arguable nature of the interpretation of the results, it is not possible to cuantify metrics but, the results itself. In the first approach[Fig. 3], Style Transferring was applied using a weight relation of 100:1. The MSE-Steps relation shows off a stable point over 10 steps where the image is no longer updated because of its similitude to the original and the superation of the threshold imposed.

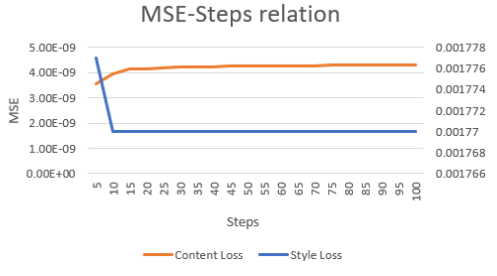


Fig. 3: Graphical style transfer results when applying a heavy weight to the base image.

The result images [Fig. 4] for this approach were the expected. A high similitude to the base image and no appretiation of the style image.

Fig. 4: Image generated from the input images when applying a heavy weight to the base image.

In the second approach, Style Transferring was applied with a configuration of 1:100. The expected result[Fig. 5] is to have an image with the style applied but still having some similitudes to the original base image[Fig. 6]. The MSE graphic states that over time, style and base were both optimized to achieve a stable point. In the case of the content loss, the stable point was close to 0 and for the style around 4.5.

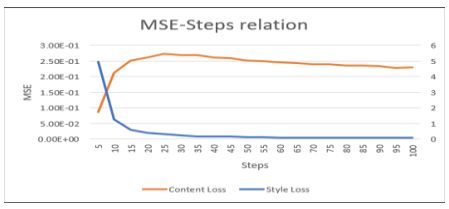


Fig. 5: Graphical results when applying a mixed weight.

Fig. 6: Results applying a mixed weight.

The final experiment was performed applying a high style weight, in particular a ratio of 1:5000.

As we could expect[Fig. 7], the base image had a high loss around 3000 units of MSE loss while style has been optimized around 0. The image that the algorithm provides as result is more similar to the Style image than to the Base image[Fig. 8].

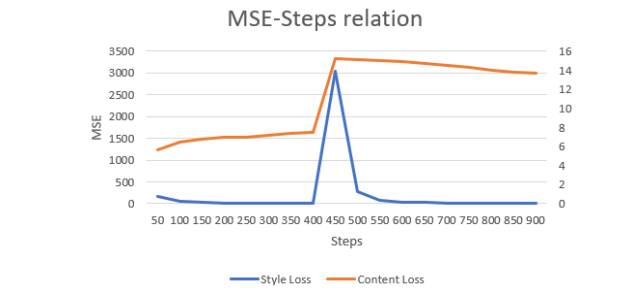
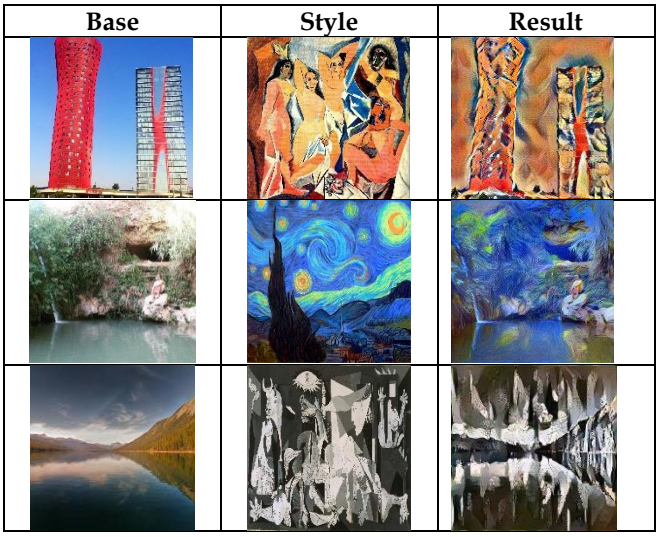


Fig. 7: Graphical results when applying a heavy weight to the style image.

Fig. 8: Image generated from the input images when applying a heavy weight to the style.

The algorithm perfoms correctly when random images are provided [Fig. 9].



Performed evaluations demonstrates a correct execution of the algorithm according to the original formula exposed in the explanation section and in the paper of Leon A. Gatys.

2.2 Denoising network

The denoising networks consists in a deep convolutional network specialized in removing noise from images. The model architecture implemented in this project is the proposed in the former paper [5] published by Aojia Zhao.

2.2.1 Algorithm Explanation

This architecture is based in a convolutional-deconvolutional model that interpretates a result image I as the addition of an image result of a degradative function D plus an additive noise h.

$$I' = D(I) + h$$

Where:

I' is an image with noise,
I is an image with noise,
$D(x)$ is the degradative function,
h is an additive noise (i.e. by the channel)

Performing consecutive convolutional and deconvolutional operations the noise can be erased progressively, and the image can be reconstructed according to the original image.

2.2.2 Model architecture

The model used in the original paper and the implemented in this project is a stacked convolutional-deconvolutional model of 10 layers [Fig. 10]. The first 5 layers are convolutional, and the 5 consecutives are deconvolutional layers. Those layers are connected using 4 direct and symmetrically connections[6].

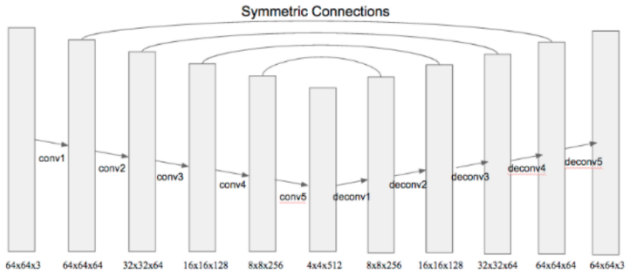


Fig. 10: Model architecture exposed by Aojia Zhao.

The main purpose of the model during the scale-rescale process is to extract the feature vectors of the training dataset that, according to the correct weights of the delta regulation parameters of the connections, will allow us to reconstruct the original image.

In the convolution stage, the formula used in every layer to extract the data is:

$$X_i = Conv(\delta * X_{i-1})$$

Where:

X_i *is the result of a convolution stage*
δ *is the gating factor,*
X_{i-1} *is the result of the previous convolution stage*

In the other hand, in the deconvolutional stage the formula used to reconstruct is the following:

$$X_i' = Deconv(X_{i-1}' + (1 - \delta) * X_i)$$

Where:

X_i' *is the result of a deconvolution stage*
δ *is the gating factor,*
X_{i-1}' *is the result of the previous deconvolution stage*

The delta factor regulates the information flow between layers. To minify bad visual effects that may occur, it is convenient to apply a Rectified Linear Unit to rectify possible negative values that can appear in the process.

2.2.3 Data and training

Dataset choices are restricted because of data labelling. In image denoising it is not necessary to classify, the objective is to perform regression on the images. In the original paper, Zhao used the STL-10 dataset. In this project the choice has been the CIFAR-10 dataset. In the original paper, the researchers used images of atleast 64x64 pixels, but in this project implementation, all images will be resized to 64x64 pixels. The training will be performed 2000 epochs over 150 images of the first batch of the CIFAR-10 dataset and the conclusions will be analysed over random images of the evaluation batch due to the lack of computational power.

In the former paper, the author performs a training over the dataset using MSE pixel loss and the Stochastic Gradient Descent with minibatches. In this implementation, the training has been performed using Stochastic Gradient Descent with minibatches.

2.2.4 Results

The results [Fig. 11] obtained in the evaluation period shows off results very similar to the exposed in the original paper of Aojia Zhao.

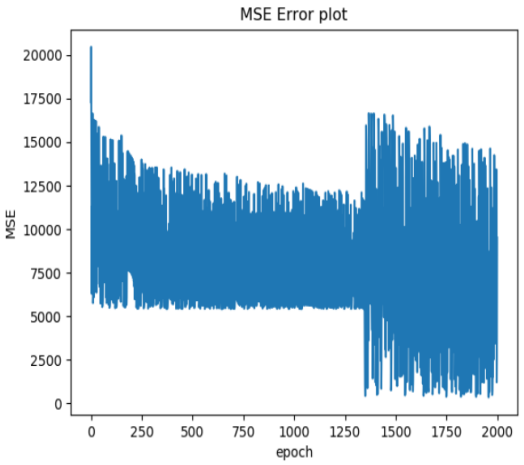


Fig. 11: Results of the training stage.

Comparing the results with the results of the original paper, we can observe that the upgrade in the MSE loss is obtained after 1200 epochs, while in the original paper it happens after 5000 epochs. Although the curve is very similar in both trainings, this may happen because of the small size of the dataset used in this project that turned up into an overfitted model. The mean MSE for the first 1000 iterations is around 10000 units, over this point, in some cases, the MSE can draw down until 2500 units. The total time training for 2000 epochs has been 37.43 minutes.

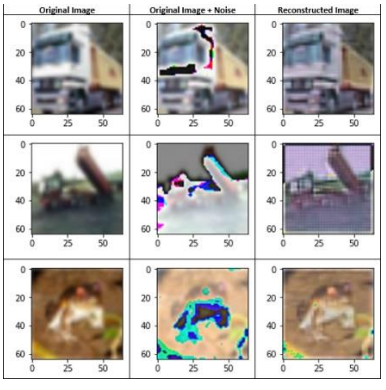


Fig. 12: Graphical results of reconstructions

2.3 Deep Dreaming

A deep dreaming algorithm can emulate the process of generating dreams or hallucinations on an image the same way that a living being can do.

2.3.1 Algorithm explanation

This process of generating dreams relies on the fact that neural networks layers store different level information in different level layers. Recent study cases demonstrate that, in general, information stored in the layers responds to:

- **First layers:** Sensible to basic functions such as borders or orientations
- **Intermediate layers:** Sensible to general components and shapes such as leafs or doors.
- **Deep layers:** those layers are sensible to very complex groupings such as trees or buildings.

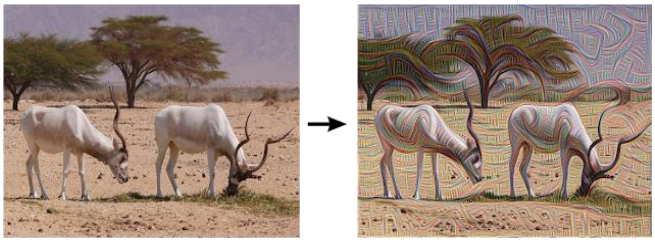


Fig. 13: Deep dreaming result by Google

In the algorithm, a random layer is chosen to optimize whatever it detects through an descent gradient. From this point, a relative shift of X pixels is performed over the original image to make an effect of dream or hallucination. In this implementation, an Stochastic Gradient Descent is used to perform the minimization of the signal.

To improve the results and make a more pleasant resultant image, the process is performed in a recurrent rescaling procedure using octaves and octaves factors[9]. Every octave is a depth level rescaled to a scale factor, that merged with the previous image, gives a fractal effect of hallucination.

2.3.2 Model

The model used for deep dreaming in the original paper is a GoogLe Net trained on ImageNet dataset. This implementation will be performed over a VGG-19 using the pretrained weights of PyTorch Model Zoo. This model has been trained for image classification purposes and can perform accurately in pattern and object recognition. After slicing the network, the selected layer to perform a hook is the 34th. This layer is deep enough to general object recognitions, which are a key of success in generating relevant high-level patterns in the input image.

2.3.3 Results

To analyse the results, three random images have been selected and processed through the network [Fig. 14]. The layer selected to analyse is the last convolutional layer of the VGG19 model. In general, the selected layer is specialized in high level feature detection. If we check for the first image, we can identify several eyes. In the second and third image, several random shapes can be found such as mountains or animals.

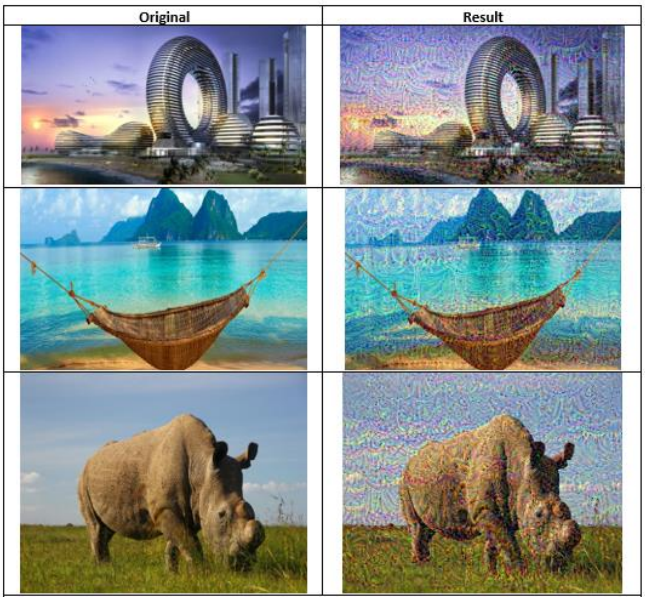


Fig. 14: Results of deep dreaming.

Due to the recursive nature of the algorithm, a fractal image [Fig. 15] has been forwarded through the network to figure out how this network could interpretate its information. Apparently, the algorithm replicates the geometric shape of the image randomly over the fractal.

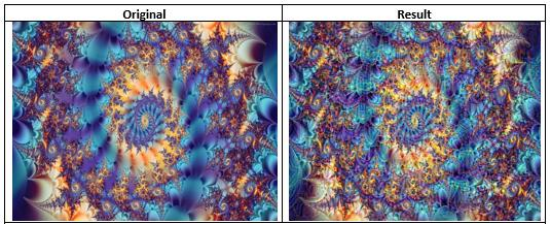


Fig. 15: Results on a Fractal image.

For analysis purposes of figuring out which information is stored in every layer, the third previous image has been processed by the first convolution layer and the last convolutional layer of the network [Fig. 16]. As stated in the theory behind neural networks, every layer interpretates the information in such a different way.

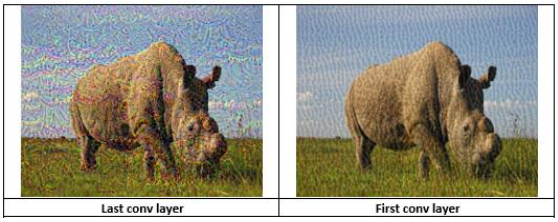


Fig. 16: Results by first and last layer

The information that every layer is capable of identify depends on the depth of itself. While the first conv layer seems to be sensitive to edge detection, the last convolutional can detect complex objects on its image.

3 PROJECT CONCLUSIONS

Despite the difficulty of implementing those algorithms from zero for a novice future computer engineer, the project has been accomplished satisfactory.

One of the biggest issues while developing the project was setting up the environment to bring the algorithms to a mobile platform. Initially, this objective was reflected into the initial report as a primary objective but, because of the incompatibility with required dependencies, a lot of time and effort was spended trying to solve it and definitely, with the support of the supervisor teacher the decision was to replanify the project to ensure the correct development of the algorithms, which in fact, are the core of this project. Fortunately, the time schedule was accomplished correctly, and a basic design of a GUI was implemented.

Another issue that has been intrinsic in the project is the basic knowledge set in deep learning. In the first implementation of Style Transferring algorithm the major problem was learning to implement the first paper and, at the same time, learn to use PyTorch as framework. Progressively, this issue has been solved. The second implementation has been easier than the first because of the accumulated experience in understanding articles and the experience earned using PyTorch, and the third implementation despite of the difficulty of finding relevant documentation, has been easier than the previous ones in terms of using PyTorch and understanding the articles.

This work has definitely met its goals, not only towards the familiarisiation with the state of art of deep learning for Image Processing which is a constant evolution field, but the learning to implement image processing papers. Facing common problems in this area such as choosing the most suitable deep learning framework, choosing the most suitable optimization learning method, treating the data before and after processing, trying to figure out the best analysis method to analyse the results and trying to repair execution problems derivated of malfunctioning or optimize it, are valuable skillsets that are useful for any specialist in deep learning and are difficult to acquire in a closed academic environment where the guidelines are well defined.

3.1 FUTURE STEPS

Although the main requirements have been solved, there are no new contributions for the community and for the current artificial intelligence panorama due to the high complexity that it implies and the shortage of time. However, this project can contribute to help future students to understand how those algorithms work, and, at the same time, can help to scientific communities to understand how different neuronal levels interact in cognitive and creative processes and the interpretation of how and when neural networks store information in their data structure.

AKNOWLEDGEMENTS

This project could not be completed without the support and guidance of BALDRICH, Ramon and the special support of my closest family and friends.

BIBLIOGRAPHY

[1] GATYS, Leon A.; ECKER, Alexander S.; BETHGE, Matthias. Image style transfer using convolutional neural networks. In Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on. IEEE, 2016. p. 2414-2423.

[2] JOHNSON, Justin; ALAHI, Alexandre; FEI-FEI, Li. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision. Springer, Cham, 2016. p. 694-711.

[3] NARANAYAN, Harish. Convolutional neural networks for artistic style transfer. In personal blog.

[4] BERGER, Guillaume; MEMISEVIC, Roland. Incorporating long-range consistency in CNN-based texture generation. arXiv preprint arXiv:1606.01286, 2016.

[5] ZHAO, Aojia. Image Denoising with Deep Convolutional Neural Networks.Etc.

[6] MAO, Xiaojiao; SHEN, Chunhua; YANG, Yu-Bin. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. En Advances in neural information processing systems. 2016. p. 2802-2810.

[7] MORDVINTSEV, Alexander; OLAH, Christopher, TYKA, Mike. Inceptionism: Going Deeper into Neural Networks.

[8] BROWNE, Kieran; SWIFT, Ben; GARDNER, Henry. Critical Challenges for the Visual Representation of Deep Neural Networks. En Human and Machine Learning. Springer, Cham, 2018. p. 119-136.

[9] PEDERSEN, Magnus Erik Hvass. Deep Dreams implementation.