Convolutional Networks

Susana Alvarez Zuluaga
salvarezz1@eafit.edu.co
EAFIT University
Mathematical Engineering
Medellín, Colombia

Mariajose Franco Orozco
mfrancoo@eafit.edu.co
EAFIT University
Mathematical Engineering
Medellín, Colombia

***Abstract*—Convolutional Neural Networks, also known as CNN, are an important branch of Deep Learning and are widely used in image processing. It consists of an Artificial Neural Network that uses convolutional layers in order to apply filters to the input for extracting some features and pooling layers which allows for reducing dimensionality. A classic example of a CNN is LeNet-5, which was designed for handwritten digit recognition and consists of different convolutional and pooling layers followed by fully connected layers. Another interesting example of CNN is the Generative Adversarial Network (GAN), which consists of 2 neural networks that use convolutional layers to learn the input and the output in order to generate realistic images and classified them correctly. In this paper, we performed a LeNet-5 using the MNIST dataset, a GAN with convolutional layers which was also trained with the same dataset as the LeNet-5, and finally, a Style GAN trained with a Monet dataset which was then applied to videos to transfer to them the style of the Monet paintings.**

I. Introduction

Deep Learning or Deep Neural Networks refers to Artificial Neural Networks that have multiple layers. In recent decades, this has been regarded as one of the most powerful tools because it can process large amounts of data [1]. One of the most popular and significant networks in the deep learning field is the Convolutional Neural Network (CNN), named after the linear mathematical operation between matrices known as convolution [2].

The earliest form of CNNs is known as Neocognitron, which was developed by Kunihiko Fukushima in the decade of the 1980s. This model was inspired by the visual cortex in the brain and used a series of convolutional layers to extract features from input images [3]. In the decade of the 1990s, Yann LeCun developed the LeNet-5 architecture, which was one of the first CNNs to be applied to real-world problems, such as handwritten digit recognition. LeNet-5 incorporated a trainable convolutional layer, which improved the network's ability to learn from data [4].

The CNN consists of multiple layers, including convolutional, non-linearity, pooling, and fully-connected layers. Their ability to learn from the input data in a hierarchical manner is what makes the performance of the CNNs outstanding. Each layer of the CNN learns increasingly abstract representations of the input data. It starts with low-level features and starts to increase allowing it to capture high-level features such as object shapes and semantic meaning [5].

Convolutional Neural Networks have become very popular for computer vision tasks such as image classification and object detection [6]. CNNs can effectively handle the high-dimensional nature of image data, while also capturing local spatial information through the use of convolutional layers. These layers apply filters to input images and learn to detect features such as edges, corners, and textures. In addition to image-related tasks, CNNs have also been successfully applied to a wide range of other applications, such as natural language processing [7], speech recognition [8], and drug discovery [9]. As deep learning methods continue to advance, CNNs are expected to remain a powerful tool for analyzing and making sense of complex data in a variety of fields.

In this project, we will explore the use of different CNN models applied to image problems. First, a LeNet-5 will be trained for the problem of handwritten digit recognition. Then a Generative Adversarial Network (GAN), which was first proposed by the authors in [10] in 2014 will be trained for the same handwritten digit recognition problem. Finally, a StyleGan, which is a model proposed in 2019 by the authors in [11] will be trained and then used to perform style transfer on some videos. The outline of this paper is as follows. In Section II a theoretical framework about Convolutional Neural Networks, specifically about the LeNet-5, GAN, and StyleGan architectures will be presented. Then in Section III an explanation of the implementation of each CNN will be addressed. In Section IV the results obtained will be presented and finally, in Section V some conclusions and future work will be exposed.

II. Theoretical Framework

A. LeNet-5

As previously mentioned, LeNet-5 is a type of CNN developed by Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner in the 1990s. It was designed for handwritten digit recognition. In Figure 1 the LeNet-5 architecture is presented. LeNet-5 is made up of seven layers, including two convolutional layers, two pooling layers, and three fully connected layers.

The LeNet-5 takes as input a grayscale image of 32x32 pixels. This image is then passed through 2 convolutional layers which are in charge of extracting features from the input image. These layers use a set of filters known as Kernels to extract features from the input image. The filters slide across the input image to generate a feature map. Then these feature maps are passed through two pooling layers, also known as sub-sampling layers. These pooling layers are in charge of reducing the dimensionality of the feature maps while preserving important information. Subsampling layers help to make the model more robust to small variations in the input image and reduce the number of parameters in the network. [6].

It is worth noting that after each convolutional and sub-sampling layer, a hyperbolic tangent function is used as the activation function and it is applied to the resulting feature maps. After the pooling layers, come three fully connected layers which are used to map the feature maps from the previous layers to the output class probabilities. Finally, the output of the last fully connected layer is passed through a Softmax function which outputs the predicted probabilities for each class. [6].

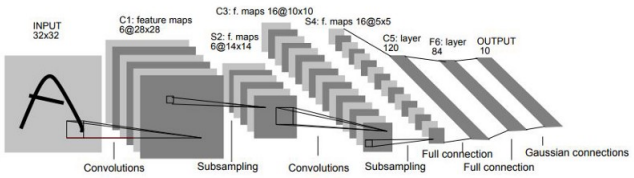

Fig. 1. LeNet-5 Architecture

B. GAN

A Generative Adversarial Network (GAN) is a type of neural network architecture that consists of two components: a generator and a discriminator [10]. As mentioned before, the first Generative Adversarial Network was introduced by [10] in 2014. The paper proposed a new approach that used two neural networks working together to generate synthetic data. The idea behind it was inspired by the zero-sum game, where one player's game is balanced with the other player's loss [12]. It consists of 2 networks that are commonly formulated through a minimax optimization problem as follows:

$$\min_{g \in G} \max_{d \in D} V(g, d) \tag{1}$$

The generator receives random noise as input in order to produce synthetic data samples that are meant to resemble real data from a given distribution, while the discriminator is trained to distinguish between the synthetic data and the real data [10]. The two components are trained together in a competitive process, where the generator tries to produce better synthetic data that can fool the discriminator, while the discriminator tries to improve its ability to distinguish between the real and synthetic data [10]. This process is shown in Figure 2

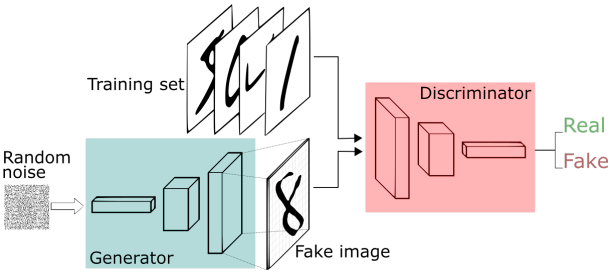

Fig. 2. GANs Architecture

Through this process of adversarial training, the generator gradually learns to produce synthetic data that is indistinguishable from real data, while the discriminator becomes better at identifying the differences between the two [10]. Since the introduction of GANs, there have been many variations and extensions proposed, including conditional GANs, cycle-consistent GANs, and more. It has also found applications in a variety of fields, including image and video generation, text generation, and data augmentation [10], [13], [14].

C. StyleGAN

A Style Generative Adversarial Network, also known as StyleGAN is a type of GAN that was introduced by Nvidia researchers in 2019. In a StyleGAN, the generator network is designed to generate images with a specific style. The GAN achieves this by separating the latent space into two parts: the style space and the content space. The style space controls the overall style of the image which includes textures and colors. On the other hand, the content space controls the specific features of the image which includes features such as shape and structure. While training, the StyleGAN is capable of manipulating the style vectors of the images in the latent space to produce images with varying styles, while keeping the content vectors unchanged. This enables the network to create a wide range of images with consistent styles, making them a powerful tool for performing style transfer [11].

Style transfer is a technique used to apply the style of one image to the content of another image. One way to perform style transfer is to use the pre-trained generator network of a StyleGAN to generate images in the desired style, and then combine the generated images with the content of the target image. Style transfer is a versatile tool for creating visually appealing and engaging content, making it very useful in many fields such as photography, film, art, advertising, fashion, beauty, and virtual and augmented reality [15]. In this project, we will transfer style to some videos using a StyleGan trained on Monet's paintings.

III. Methods

This section provides an overview of the works conducted. As previously mentioned, three distinct Convolutional Neural Network models were trained and tested. The following sub-sections will present an explanation of the methods used to implement each model. Firstly, the methods used to implement

the LeNet-5 will be outlined. Secondly, the methods used to implement the GAN will be described. Finally, an explanation of how StyleGAN was implemented and used to perform style transfer will be provided.

A. LeNet-5

To implement LeNet-5, we used and adapted the code presented in here. This code implemented a LeNet-5 architecture using Python and the TensorFlow library. Our objective was to train a LeNet-5 using the MNIST dataset and test it using a dataset recollected by ourselves. A more detailed explanation of how this process was conducted will be described below.

We first obtained the MNIST dataset which was obtained from here This dataset contains a large collection of grayscale images sized at 28x28 pixels, with each one containing a handwritten digit between 0 and 9. The dataset consists of 60,000 training images and 10,000 test images which were used as validation images. After downloading the data, the input data was normalized and it was used to train the model. After training the model, the images in the testing dataset were transformed so that they would have the same characteristics as the images in the MNIST dataset (have a size of 28x28 pixels and be gray scale). These images were then used to test the model and evaluate the models' performance

B. GAN

For the GAN implementation, we used the code presented in here where they perform a deep convolutional GAN implementation and train it with celebrities' faces. We adapt that code and train it for the MNIST dataset used before in the LeNet-5. The first step is to transform the dataset. Then, the training will be performed on the generator and discriminator at the same time with the training dataset. The generator starts with random noise as an input, which produces an image without any form, but the higher the epochs (or iterations), the better the image generated by the generator network. Once the training is done, the testing can be performed. The output of the discriminator is a probability from 0 to 1, meaning that, if the probability is < 0.5, the discriminator has determined that the image received is an image generated with the generator network, and, if the probability is > 0.5, then the image is real. The GAN was trained for 20 epochs, with a learning rate of 0.0002 and 1 GPU.

C. StyleGAN

To implement THE StyleGAN, we utilized NV labs' PyTorch implementation, which is available here. Our objective was to train a StyleGAN using a selection of Monet's paintings and then use the trained model to perform style transfer on some landscape videos. A more detailed explanation of how this process was conducted will be described below.

We first obtained a dataset of 300 images of Monet's drawings from Kaggle. After that, the training data was prepared by transforming all the images to a size of 256x256 pixels and the same color depth. To do so, a function provided in the package mentioned previously was used. After this, we proceeded to train the model. The StyleGAN was trained for approximately 48 hours in Google Colab using 1 GPU.

After the training process was done we were left with a resulting trained StyleGAN model which could be used to perform style transfer on any set of images we wanted to apply the style of Monet to. We proceeded to perform style transfer on some landscape videos. To do so, we first converted these videos into a sequence of images. Then these images along with the trained model were used to perform the style transfer. After the images were transferred, then they were then put together to make up the videos again.

IV. Results

A. LeNet

The loss function used in the LeNet-5 implementation was the the cross-entropy loss. In simple terms, the cross-entropy loss measures the difference between the predicted probability distribution and the actual probability distribution of the data. Another important thing to highlight is that the optimization algorithm used was the Adam optimizer which is similar to the Stochastic Gradient Descent, but performs better.

Using this cost function and optimization algorithm, the model was trained for different amounts of epochs and different batch sizes. The best result was obtained when the model was trained for 5 epochs and a batch size of 250 images. When it was tested for a larger amount of epochs or batch size, overfitting was evidenced. In Figure 3 the evolution of the loss and accuracy when training the model can be seen. It is important to notice that the loss and accuracy shown in this Figure correspond to the validation dataset. From 3 it can be evidenced that this model is very effective since it takes a small amount of iterations for the model to obtain very low values for the cost function and high values for the accuracy.

This model was tested on a dataset which was recollected by ourselves. With this dataset the model had an accuracy of 52%. It can clearly be seen that the model performed much better with the MNIST dataset than with our own dataset. This difference between the results obtained with the validation dataset could have been caused by difference between the MNIST dataset and out dataset. The MNIST dataset contains images that are relatively clean and easy to classify. On the other hand, our own dataset was more complex, it contained more noise and variations, it had different image resolutions, lighting conditions, and image distortions. This means that the LeNet model trained on MNIST was not be able to generalize well to our own dataset.

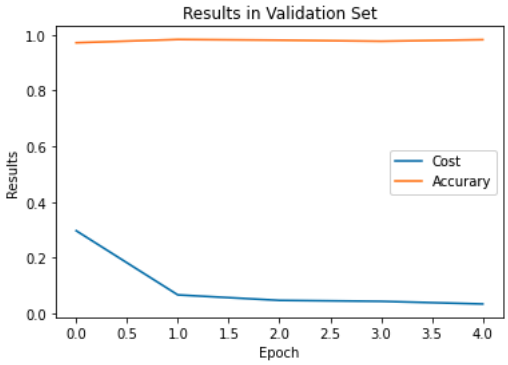

Fig. 3. Accuracy and lost on validation set

B. GAN

The loss function considered in the GAN implementation was the Binary Cross Entropy function (BCE). The loss for the discriminator and the generator was tracked for the different batches in the training and are presented in Figure 4.

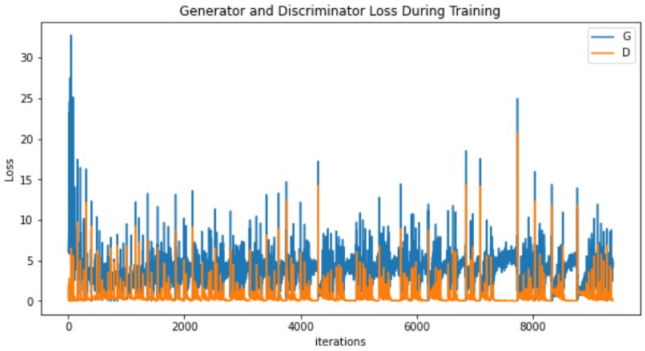

Fig. 4. Discriminator and Generator Loss

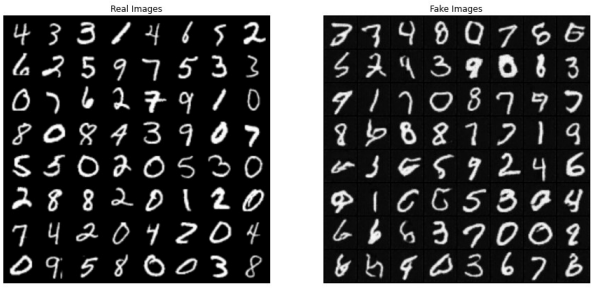

Fig. 5. MNIST images vs Generated images

The images generated with the generator network after the training can be seen in Figure 5. It can be seen that the generator learned from the training and it reproduces understandable images similar to the MNIST data.

Finally, the validation was performed with the dataset that we recollect, which consists of 300 images of numbers written by humans as it is presented in Figure.

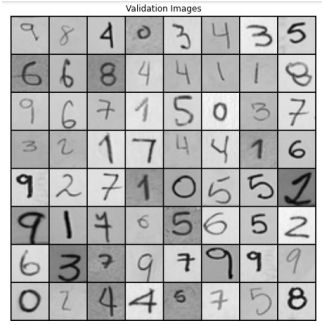

Fig. 6. Images of the numbers written by humans

This dataset was received by the discriminator in order to classify it as real or fake data. The results obtained are presented in Table I.

TABLE I
PERCENTAGE OF OUTPUTS FROM THE DISCRIMINATOR

Probability	Generated Images	MNIST Testing Images	Human Images
<0.5	100%	1.56%	35.31%
>0.5	0%	98.44%	64.69%

C. StyleGAN

As stated before, the StyleGAN was trained for about 48 hours. The learning process of the model was monitored continuously and the scores obtained in each epoch are presented in Figure 7. This Figure contains the value obtained for the FID metric at different epochs of the training process. The Fréchet Inception Distance (FID) is a metric used to measure the similarity between two sets of images. In the context of StyleGANs, it is commonly used to evaluate the quality of the generated images generated by the generator model by comparing them to the set of real images. The FID measures the similarity between the distributions of real images and generated images by comparing their features extracted from an Inception network. A lower FID score indicates a higher quality output [16]. As seen in Figure 7, the lowest FID obtained was of 82.

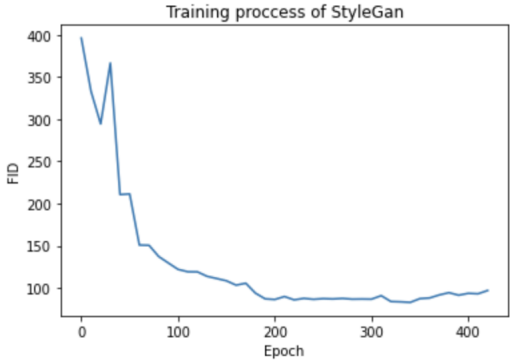

Fig. 7. Training process of StyleGAN

Figure 8 presents the images used to train the StyleGAN. It can be seen that they all are artworks created by Monet. On the other hand, Figure 9 presents the images generated by the StyleGAN when it was fed with random inputs. By comparing these images to real images the StyleGAN was trained with, it can be seen that the generator model in the StyleGAN was able to generate images with Monet's style. The trained StyleGAN was then used to perform style transfer on some videos. These videos were recorded by ourselves. The original videos and the videos obtained when performing the style transfer are presented in Table I.

Fig. 8. Real images used to train the StyleGAN

Fig. 9. Images generated by the StyleGAN

TABLE II
ORIGINAL VIDEOS VS VIDEOS WITH STYLE TRANSFER

Original video URL	Video with style transfer URL
Nazare	Nazare with style transfer
Portugal Beach	Portugal Beach with style transfer
La Guajira	La Guajira with style transfer
Alhambra	Alhambra with style transfer
Portugal Cliffs	Portugal Cliffs with style transfer

After watching the original videos and the ones with style transfer, it is evident that the frames exhibit a clear resemblance to Monet's distinctive style. While certain characteristics of the original video may have been left out in the process, this outcome can be attributed to the emulation of Monet's artistic style. Monet's style was characterized by a focus on light, color, and the natural world, with loose brushwork and an emphasis on capturing the essence of a scene rather than a realistic representation. This stylistic approach is reflected in all the videos, where various details from the original ones are omitted during style transfer, resulting in a video that captures the essence of the scene, rather than its details.

V. CONCLUSIONS

Overall, the results from all three models were satisfactory, providing valuable insights into the application of deep learning to image-related problems. Through this project, we have gained a strong understanding of the principles and techniques involved in deep learning, and their potential to address a wide range of challenges. Moving forward, we will discuss our conclusions based on the results, and highlight potential avenues for future research.

During the training of the LeNet-5 model, we observed that the accuracy on the testing dataset was not very satisfactory. This is primarily because the MNIST dataset, on which the model was trained, mainly consists of images that are relatively simple and easy to classify. However, our own dataset contained more complex images with added noise, variations, different image resolutions, and lighting conditions, making it significantly more challenging. As a result, the LeNet model trained on the MNIST dataset was not able to generalize well to our own dataset. In future works, we aim to improve the performance of our model by collecting more data, and training, validating, and testing the model on our own dataset. We believe that this approach could yield better results and help the model to better adapt to the complexity and diversity of our data.

As it was mentioned above, after training the discriminator of the GAN model, the output will be a probability. If it is less than 0.5, it means that the data is classified as fake data, and 0.5 and above means that is classified as real data. It can be seen in Table I that, for images generated with the generator network, the discriminator could classify 100% as fake images. For the MNIST data, the discriminator classified 98.44% of it as real data. And finally, for the data recollected from real people, the discriminator could classify 64.69% of it correctly. Even though the results for the validation dataset were not the desired outputs, with these results, we can conclude that the discriminator is trained correctly and can accurately identify between real and fake data.

On the other hand, when testing the StyleGAN, the model yielded an FID score of 82, which isn't a very good result. This may be attributed to the training time and the number of images used to train the model. However, given a large

amount of time required for training such models, it was not feasible to train the model for a longer duration. Also, the training dataset only included 300 images, which may have restricted the model's ability to generate more accurate style transfer outputs. In future works, we aim to extend the training duration of the StyleGAN model and utilize a larger amount of images during training to potentially enhance its performance. Additionally, it is worth exploring the suitability of other models, such as CycleGAN, for style transfer tasks, and comparing their efficacy against that of StyleGAN.

REFERENCES

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[2] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022.

[3] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, 1980.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha, "LeNet," *Proceedings of the IEEE*, no. November, pp. 1–46, 1998.

[5] A. Saxena, "An Introduction to Convolutional Neural Networks," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, no. 12, pp. 943–947, 2022.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[8] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4277–4280.

[9] E. Gawehn, J. A. Hiss, and G. Schneider, "Deep learning in drug discovery," *Molecular informatics*, vol. 35, no. 1, pp. 3–14, 2016.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, pp. 2672–2680, 2014.

[11] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4401–4410.

[12] F. Farnia and A. Ozdaglar, "GANs May Have No Nash Equilibria," pp. 1–38, 2020. [Online]. Available: http://arxiv.org/abs/2002.09124

[13] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations*, 2018.

[14] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Wang, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *IEEE International Conference on Computer Vision*. IEEE, 2017, pp. 5907–5915.

[15] R. Abdal, Y. Qin, and P. Wonka, "Image2stylegan: How to embed images into the stylegan latent space?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4432–4441.

[16] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2018, pp. 6626–6637.