On Supervised Hybrid Models

Ana Sofia Gutierrez
Universidad EAFIT
Medellin, Colombia
asgutierrt@eafit.edu.co

Olga Lucía Quintero Montoya
Universidad EAFIT
Medellin, Colombia
oquinte1@eafit.edu.co

***Abstract*—In recent years, Artificial Intelligence has emerged as a dominant branch in the identification of intricate patterns and nonlinear relationships in input-output data. This surge of interest in implementing AI models for everyday tasks has led to innovative applications. This project explores the versatile capabilities of neural networks, including their use in classification, regression, and feature extraction. It begins by implementing autoencoders to manipulate data dimensionality and assess their impact on regression tasks. The study also investigates variations of the Multilayer Perceptron (MLP), such as convolutional neural networks (CNNs). We delve into LeNET5's application for MNIST dataset classification, and propose a validation on dataset created in the classroom. Additionally, Generative Adversarial Networks (GANs) are examined for their role in generative and discriminative modeling, once again validated using the classroom digits dataset. One last network, Style-GAN, explores image generation and modification. This comprehensive exploration encompasses a wide range of artificial intelligence models, and applies them as an education exercise.**

***Index Terms*—Autoencoding, Convolutional Neural Networks, LeNet5, Generative Adversarial Networks, Style Transfer.**

I. Introduction

In recent years, the connected networks paradigm has emerged as a dominant branch in the development of artificial intelligence models. There has been an increased interest in learning and implementing these models to streamline daily tasks, accompanied by innovative applications that bring the world closer to the realm of artificial intelligence. These models have gained recognition for their exceptional ability to identify intricate patterns and nonlinear relationships between input and output data. As a result, Neural networks have found utility in solving classification, regression and feature extraction challenges, demonstrating the extensive versatility of these models.

This work begins by implementing autoencoders to manipulate data dimensionality. We investigate their impact on solving a regression problems through the optimization of an MLP trained on original, compressed, and expanded data dimensions to determine the efficacy of dimensionality alteration. This work then focuses on the application of CNNs, aiming to comprehend their architecture, advantages, and limitations. In previous reports, we had already delved into the Multilayer Perceptron (MLP), examining the influence of hyperparameters on its architecture and gradient descent optimization. While we explored the MLP's capabilities in classification and regression, it can also serve as a fundamental component in other models, such as autoencoders and convolutional neural networks (CNNs).

A second part of this project explores the implementation of the LeNET5 architecture, designed to tackle the MNIST dataset classification problem. It demonstrates the utility of convolutional networks in solving high-dimensional challenges. COntinuing the exploration of CNNs, the Generative Adversarial Networks (GANs) provides insights into the training process of generative and discriminative models for digits generation. Lastly, we experiment with style transfer using Style-GAN, to showcase image generation and modification.

II. Methodology

A. Autoencoder

The Autoencoder is a Multi-layer perceptron where the architecture is set up to learn the input data. This approach resembles a principal component analysis, aiming to preserve maximal information from the input data while reducing dimensionality. However, an autoencoder also has the capabilities of expanding the input data. In general, autoencoders excel in mapping nonlinear relationships between the original and modified data, facilitating effective feature extraction processes. The full model is described by Quintero [1] and has already been reported on a previous class project.

The dataset comes from available data presented to the class. It is composed of 3 inputs. For the Autoencoder, the outputs presented on Figure 1 will be expanded and compacted and used to feed another MLP that decodes them.

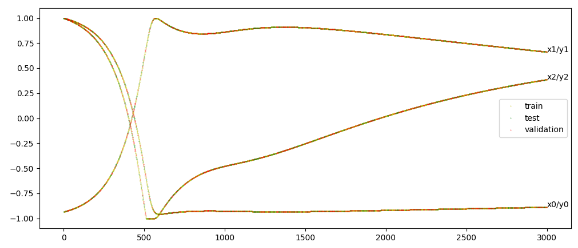


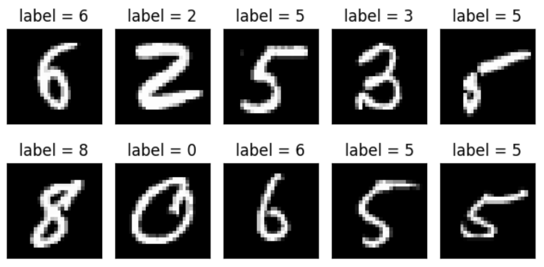
Fig. 1: Extended dataset of Bacterial Growth

B. LeNet-5 Convolutional Neural Network

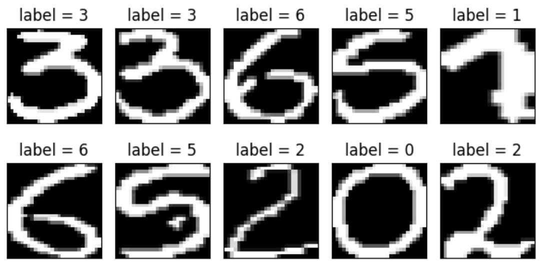
In a Convolutional Neural Network (CNN), input data goes through sequential processing in convolutional layers to extract distinct features. The CNN typically has two main sections: the first is the network's backbone, which conducts feature extraction through convolutional and pooling layers. The second section involves a fully connected neural network that receives the output from the backbone and utilizes the extracted features to make predictions or classifications.

The LeNet-5 architecture, introduced by Lecun et al. [6] in 1998, is made out of seven layers categorized into: two convolutional layers responsible for feature extraction from input images, two pooling (sub-sampling) layers for downsampling feature maps and reducing spatial dimensions, and three fully connected layers for classification based on extracted features. Originally designed for handwritten digit recognition within the MNIST dataset, LeNet-5 has found applications in various computer vision tasks.

In this particular project, the focus remains on recognizing handwritten digits, utilizing the MNIST dataset for training and testing, and a separate dataset containing new handwritten digits to validate the model's performance. A sample of both datasets is shown on figure 2



(a) MNIST dataset sample



(b) Hand-written digits sample

Fig. 2: hand-written digits databases: MNIST vs. AI Class

C. Generative Adversarial Network (GAN)

The Generative Adversarial Networks (GANs) of Goodfellow et al. [4]; consist of pairs of neural networks that engage in a competitive dynamic. Specifically, one of these networks, the generator, is responsible for creating content, initially from random data. Meanwhile, the other network, the discriminator, analyzes the information generated by the generator and determines whether it is real or fake, drawing on real data for comparison. This adversarial approach drives the content generation process and has proven to be a powerful technique in various machine learning and artificial intelligence applications.

This project follows the recommended implementation by Torres [5]. During the training process, the weights and biases of the Discriminator network are updated to maximize its classification accuracy, defined as the probability of correctly identifying ”real data” as ”real” and ”fake data” as ”fake”. To elaborate on the training process of the Discriminator, it can be summarized through the following iterative steps (utilizing mini-batches):

1) Select a random mini-batch of ”real data” from the training dataset.
2) Generate a new mini-batch of random noise vectors, which are passed as input to the Generator to synthesize a mini-batch of ”fake data.”
3) The Discriminator then classifies both the ”real data” and the ”fake data.”

Classification errors are computed, and the total loss is backpropagated to the Discriminator to update its weights and biases, aiming to minimize classification errors.

Likewise, the weights and biases of the Generator are also iteratively updated. This time, to maximize the probability of the Discriminator incorrectly classifying ”fake data” as ”real.” The training process of the Generator follows the previous steps, but it focuses solely on how the Discriminator has classified the mini-batches of ”fake data” (i.e., having it classify ”fake data” as ”real”). The total loss is then backpropagated through these data points to update the weights and biases of the Generator. This training strategy aims to enhance the Generator's ability to generate data that can effectively deceive the Discriminator.

D. Style transfer GAN

Style transfer is a computer vision and image processing technique that involves applying the visual style of one image to the content of another. It utilizes deep neural networks to extract content and style information from two distinct images and combines them both preserving content and adopting style. It uses an objective function to minimize content and style loss at the same time, resulting in an adversary goals structure. To train a style transfer network, the model initially learns to recognize the style of an image based on patterns, textures, and color palettes.

There are various application algorithms like Neural Style Transfer, Fast Style Transfer, and CycleGAN. This project employs the previously trained, Fast Style Transfer implementation published by Google AI on TensorFlow Hub. This variation of style transfer can generalize to capture and transfer the artistic style of paintings never previously observed by the system [3].

III. Results

A. Autoencoder

The best autoencoder architecture to higher dimensions was that of 4 neurons and a learning rate of 0.5. Figure 3 show the results of the autoencoder on the test dataset. At the same time, figure 4 shows the best autoencoder into lower dimensions was that of 1 neuron and a learning rate of 0.9

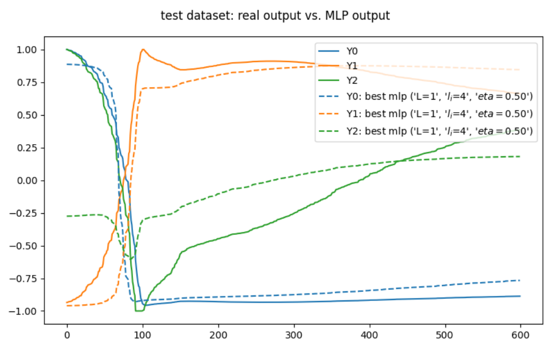


Fig. 3: Autocoder expansion of Bacterial Growth data

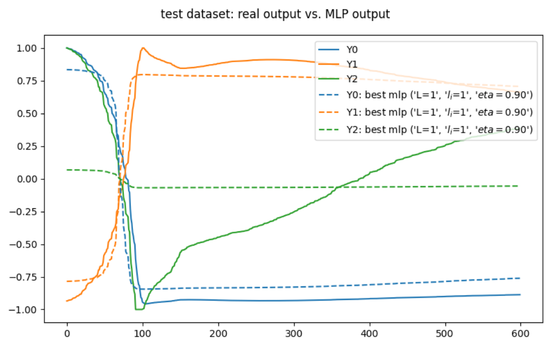


Fig. 4: Autocoder compresion of Bacterial Growth data

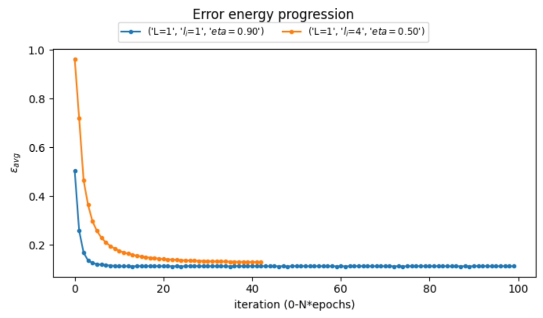


Fig. 5: Error energy progression on the autoencoder training process

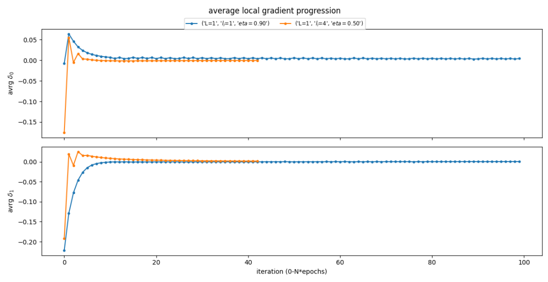


Fig. 6: Average local gradient progression on the autoencoder training process

The training algorithm of both selected autoencoders converges as shown in the average local gradients progression on figure 6. After finishing the training of both autoencoders, the code output at the hidden layer is used to train an MLP from the library sklearn with the goal of decoding the data. Table I shows the training and test errors of both models.

model input variation	train error	test_error
expanded input	0.002466	0.00215
compresed input	0.066213	0.064332

TABLE I: Error: sklearn MLP decoder trained with different inputs

Finally, Figures 8 and 7 show the output of the sklearn MLP models.

B. LeNet-5 convolutional networks

Figure 10 displays the accuracy matrix on validation data. The model was trained on 5 epochs, a learning rate of 0.0001, batch size of 64 and with data normalized to the space $[-1, 1]$. The general performance of the model evaluated on the MNIST dataset was of 0.97. The performance evaluated on the class's handwritten numbers was 0.25 and Table II shows the accuracy of each digit.

As the classroom created a dataset thas was then label by the same students, some of the digits were incorrectly written. The model was applied to the dataset and Figure 9 shows a sample of the class's written dataset and its classification by the LeNet5 model. Additionally, Figure 15 on the appendix section, show the classification of the model made on a sample of the incorrectly written digits.

Digit label	accuracy
0	0.08
1	0.22
2	0.34
3	0.33
4	0.32
5	0.28
6	0.0
7	0.03
8	0.68
9	0.2

TABLE II: digit accuracy on class' handwritten dataset

C. Generative Adversarial Network

Figure 16 shows generated numbers through the training process. It was executed on 100 training epochs, with data normalized to the space $[-1, 1]$ and a learning rate of 0.0001 for both the generator and the discriminator. In addition Figure 11 presents the binary cross-validation cost function of both the generator and discriminator through the training epochs.

As an additional validation step, the Discriminator was employed to assess the class's hand written dataset, and outputs are included in Figures 12b and 12c. Similarly, Figure 12a illustrates the Discriminator's outputs when presented with digits generated after training. In this context, "Real" inputs are designated with values close to 1, while "Fake" inputs are assigned values closer to 0. By applying a threshold of 0.5, the

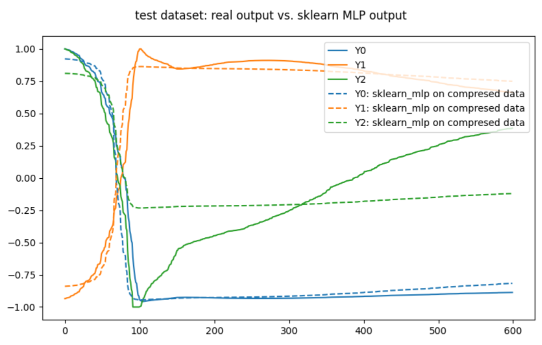


Fig. 7: Sklearn MLP output when input is expanded autoencoder data

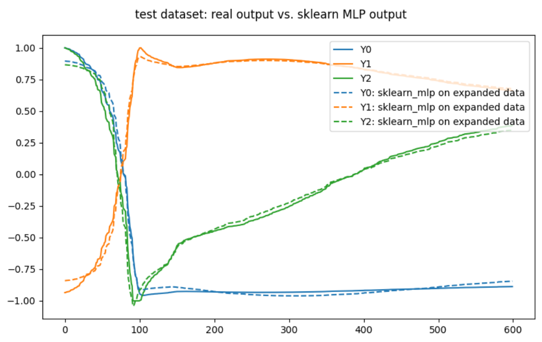


Fig. 8: Sklearn MLP output when input is compressed autoencoder data

Discriminator model categorized 98.8% of the classes' handwritten data as "fake". The only images classified as "real" are depicted on figure 13. This classification indicates that the input does not originate from the MNIST dataset, reinforcing the discriminative capability of the model.

D. Style transfer with GANs

The full style transfer applied to a brief video can be accessed with the project's deliverables. As an illustrative example, figure 17 showcases the implementation of the selected artistic style onto a single image. This exemplifies the aesthetic transformation achieved through the project's style transfer technique onto the video's content.

IV. Discussion and Conclusion

The autoencoder configuration of a Multilayer Perceptron (MLP) proves to be a valuable tool for encoding data into either reduced or expanded dimensions. Through our project, we have confirmed that an MLP model trained on this encoded data yields different outcomes compared to its performance on the original information. This highlights the significance of data dimensionality alteration through autoencoders, effectively removing not relevant data that allows for a simpler input that can still carry the necessary information for an appropriate decoding process. It reflects the importance of this technique in various machine learning and artificial intelligence applications.

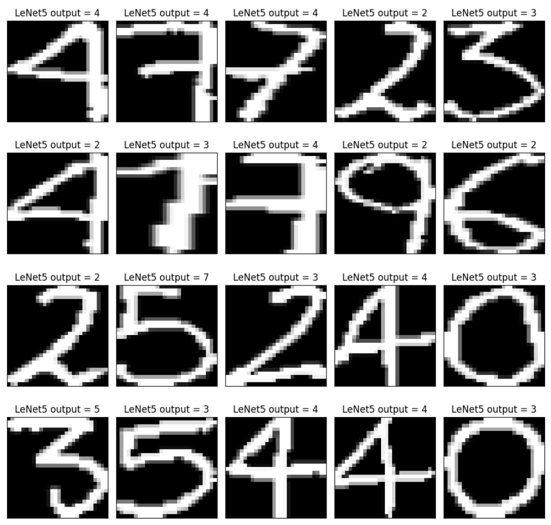


Fig. 9: LeNet generated label for a sample of the class' hand written dataset

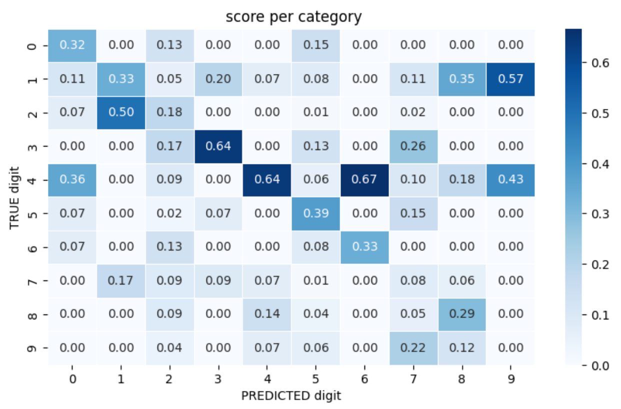
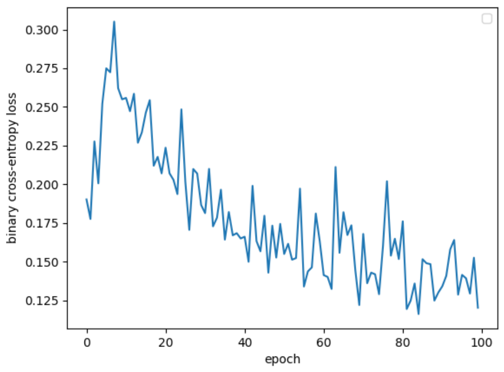


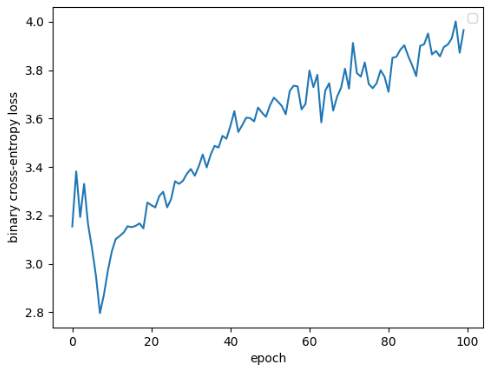
Fig. 10: LeNet-5 confusion matrix on classroom validation dataset

The LeNET-5 architecture, a blend of convolutional models with multi-layer perceptrons, proves highly effective for solving the recognition of handwritten digits in the MNIST dataset. However, the model exhibited subpar performance on the course dataset, because of disparities in dataset origins and characteristic distributions. This project resolves the well-known handwritten digit recognition challenge using the LeNET-5 architecture on the original MNIST dataset. It also underscores the architecture's sensitivity to variations in digit shapes.

The Generative Adversarial Networks (GAN) training process can be naively assessed by inspecting the generated images at each epoch and evaluating, from a human perspective, whether the generative model is improving. At the same time, the results reveal that the discriminator struggles to accurately distinguish which of our class's written digits correspond to real numbers. This discrepancy arises from the discriminator being trained on the MNIST dataset. The previous training of LNET5 for classification had already indicated significant

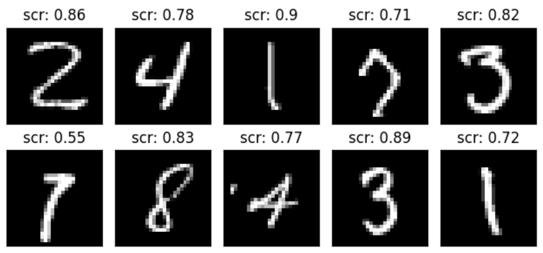


(a) Discriminator loss progression during training

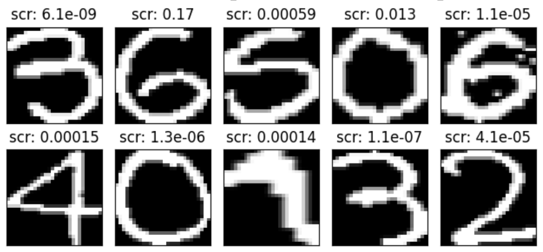


(b) Generator loss progression during training

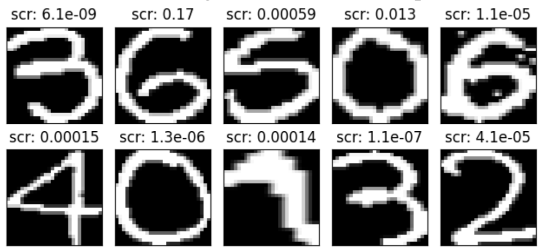
Fig. 11: Trained models binary cross-entropy loss on each epoch



(a) MNIST sample discriminated output



(b) Classroom digits discriminated output value



(c) Worst classroom digits discriminated output value

Fig. 12: Datasets' discriminated output values

dissimilarities between these datasets. Consequently, it is inappropriate to employ the discriminator trained on MNIST to identify whether the digits written in the classroom truly represent numbers.

Finally, the utilization of pre-trained models proves to be useful in gaining insight into the operations of various convolutional networks. This approach saves time and computational resources. In particular, the implementation developed by the Google AI team stands out due to its ability to accept input images that were not part of its training dataset. It can generate accurate styled results even when confronted with previously unseen reference styles.

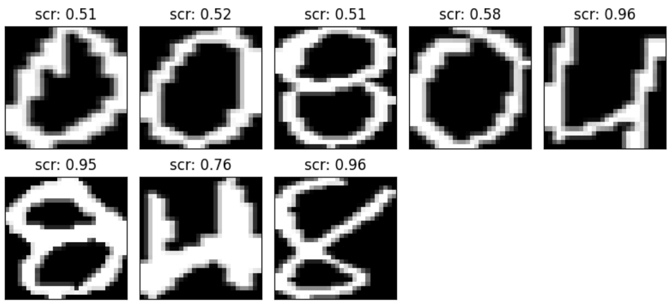


Fig. 13: only classroom digits discriminated as ”Real”

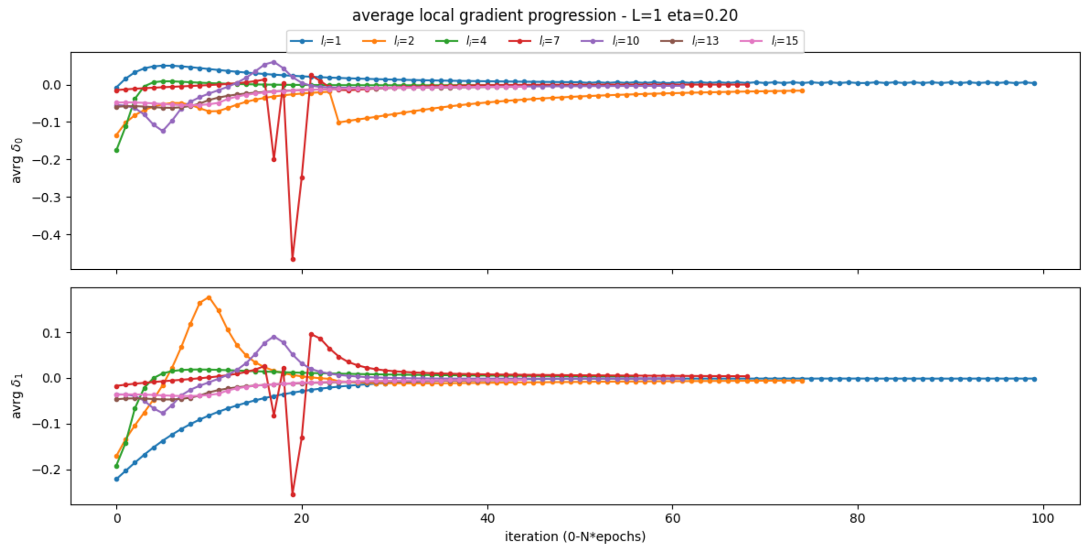
REFERENCES

[1] Quintero, Olga Lucia. 2019. Machine Intelligence for Human Decision Making.
[2] World Tourism Organization. Centro de Medellin.
[3] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, Jonathon Shlens. 2017. Exploring the structure of a real-time, arbitrary neural artistic stylization network. Proceedings of the British Machine Vision Conference (BMVC)
[4] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. 2014. Generative Adversarial Networks.
[5] Torres, J. 2020. Python Deep Learning: Introducción práctica con Keras y TensorFlow 2. Marcombo.
[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324.

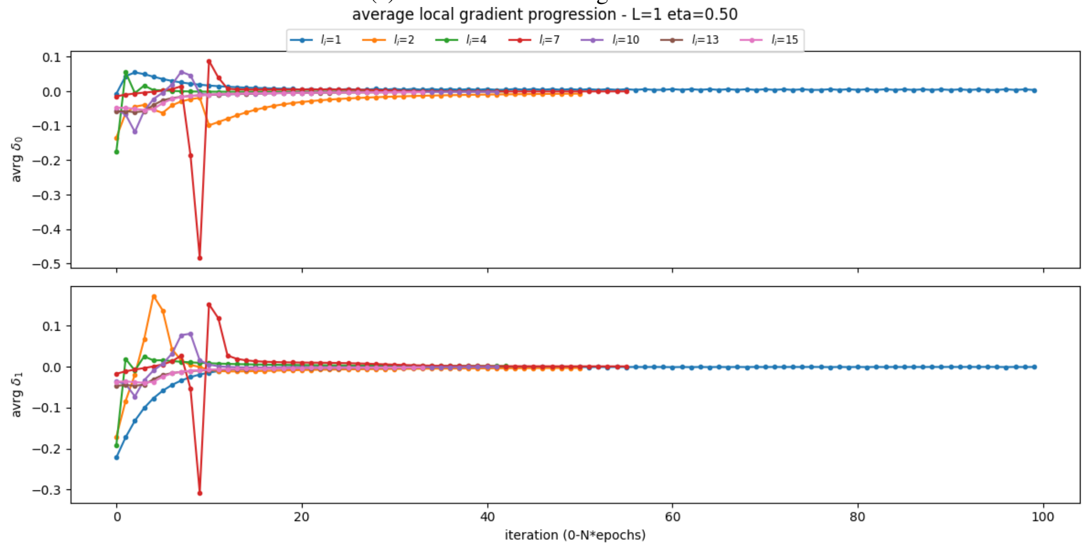
Appendix

neurons	**eta**	**train error**	**test error**
1	0.9	0.1199	0.1077
2		3.5113	3.7475
4		0.1901	0.1242
7		0.5615	0.2613
10		3.3502	3.7141
13		1.9705	2.0115
15		1.9738	2.0145
1	0.5	0.1239	0.1087
2		3.4436	3.7399
4		0.1991	0.1211
7		0.6056	0.2607
10		3.2306	3.6893
13		1.9642	2.0087
15		1.9680	2.0123
1	0.2	0.1375	0.1083
2		3.2975	3.7217
4		0.2451	0.1297
7		0.7677	0.3125
10		3.0080	3.6381
13		1.9515	2.0040
15		1.9560	2.0087

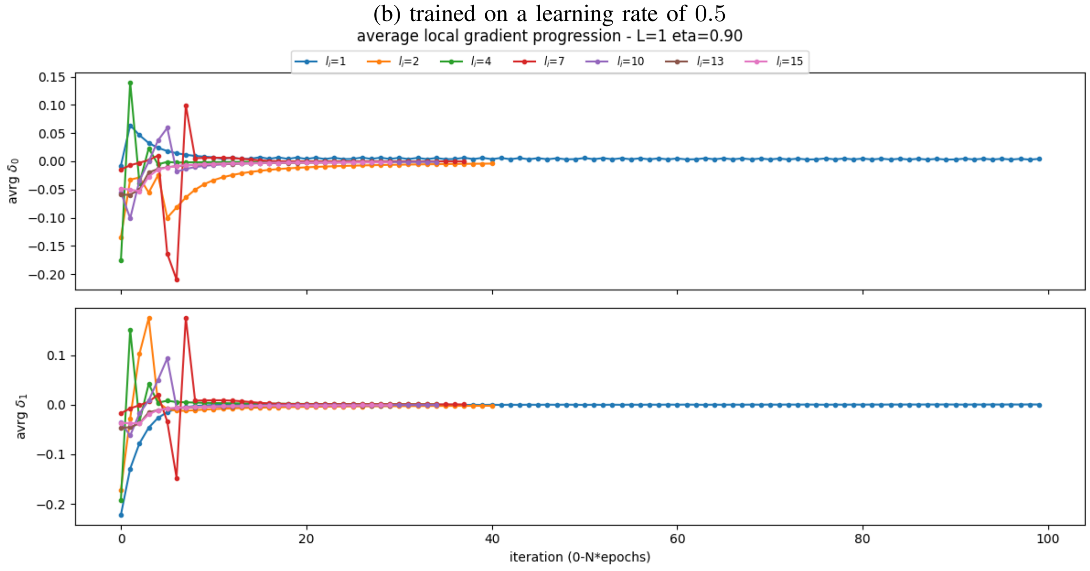
TABLE III: Autoencoder Training Results



(a) trained on a learning rate of 0.2



(b) trained on a learning rate of 0.5



(c) trained on a learning rate of 0.9

Fig. 14: autoencoder models comparison: average local gradient progression

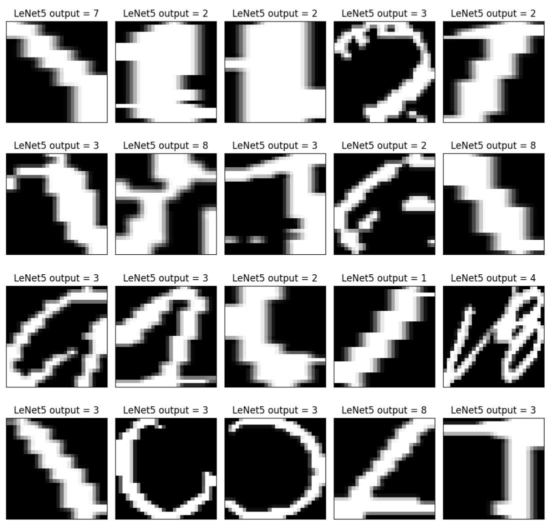
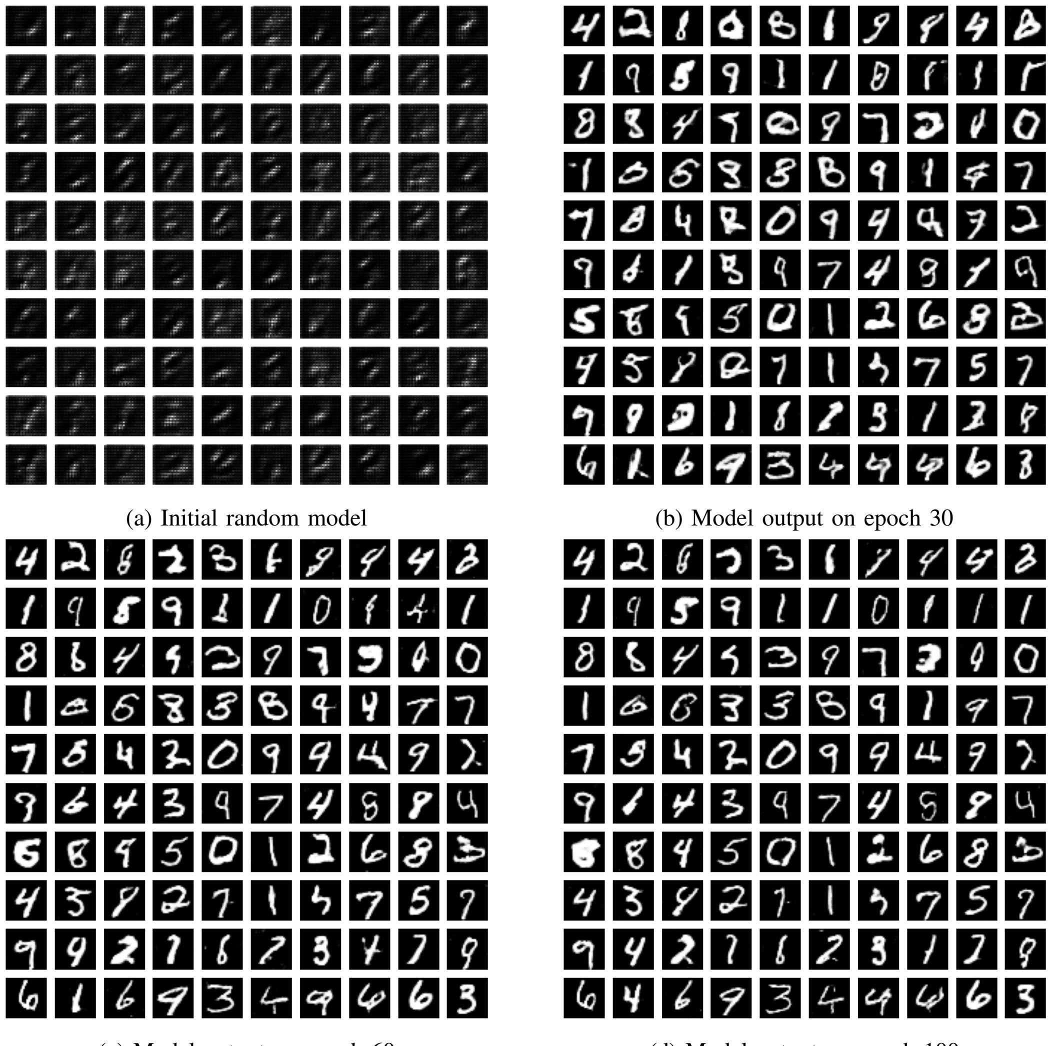


Fig. 15: LeNet generated label for a sample of the classrooms' worst hand written digits



(a) Initial random model

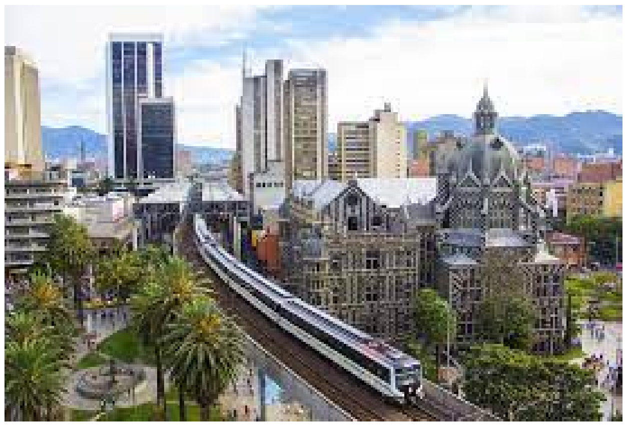
(b) Model output on epoch 30

(c) Model output on epoch 60

(d) Model output on epoch 100

Fig. 16: Generative adversarial network training on MNIST database

(a) Style reference: Tucked away by Mike Kowalski


(b) Original image [2]

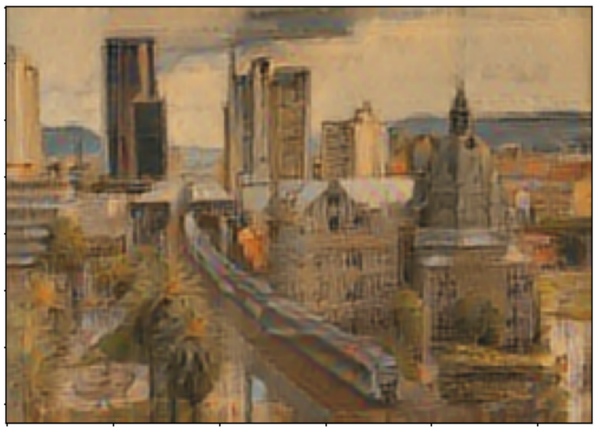

(c) Stylized image

Fig. 17: Style Transfer Network: Single image implementation