# Hybrid Methods

Camilo Oberndorfer Mejía
*Universidad EAFIT*
Medellin, Colombia
coberndorm@eafit.edu.co

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

***Abstract*—This deliverable conducts a practical investigation of hybrid supervised learning models, including the use of Autoencoders, Convolutional Neural Networks (CNNs), and Generative Adversarial Networks (GANs).The primary goal is to gain hands-on expertise with these models. The research is divided into three major stages: first, we use Autoencoders to extract salient features from datasets; second, we investigate digit image classification using LeNet-5 CNNs, using the widely recognized MNIST dataset; and finally, we investigate generative and discriminative networks for MNIST, experimenting with style transfer GANs applied to video data. Throughout this in-depth examination, we will look into essential machine learning aspects such as feature extraction, picture categorization, and regression.**

## I. Introduction

Hybrid supervised learning models, which combine Autoencoders, Convolutional Neural Networks (CNNs), and Generative Adversarial Networks (GANs), represent a viable path forward for ML applications. CNNs excel at image recognition and classification, while GANs can generate realistic synthetic data and conduct style transfer. Hybrid supervised learning models can achieve improved performance on a wide range of tasks by using the complimentary characteristics of different models.

This research paper will realize a practical exploration of these hybrid models, driven by the pressing need to bridge the gap between theoretical knowledge and practical implementation in machine learning. While the foundations of these models are well-established in the literature, their application in real-world scenarios remains a challenging endeavor. As such, the study is motivated by the desire to gain hands-on experience with these cutting-edge models.

We understand that this hands-on experience is critical not only for expanding our grasp of machine learning techniques, but also for realizing their full potential across multiple domains. To that purpose, our research tasks include a wide range of topics. To begin, we use Autoencoders to extract discriminative features from synthetic datasets. Second, we look at digit image classification with the well-known LeNet-5 CNN architecture and the MNIST dataset. Finally, we create Generative and Discriminative networks for MNIST, which is supplemented by experiments with style transfer GANs applied to recorded movies. These phases encompass essential machine learning topics like feature extraction, image classification, and generative modeling, all of which contribute to our overall goal of developing the practical uses of machine learning techniques.

## II. Methodology

### A. Data

For the autoencoder two sets of data will be used the dataset 1 is comprised by three variables, these are COD1, t,Xv1.

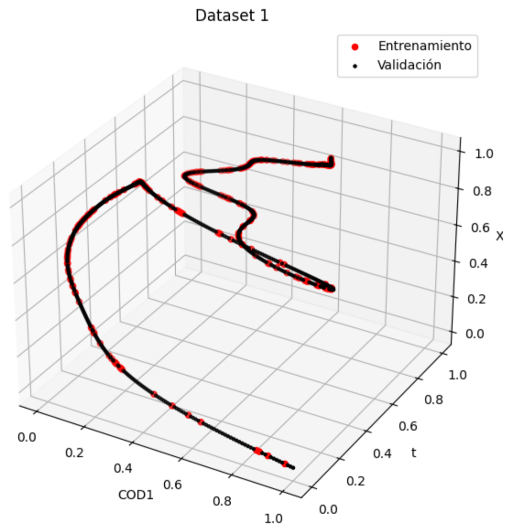


Fig. 1. 3D view of the first Dataset

Separately each of the variables has the following behavior:

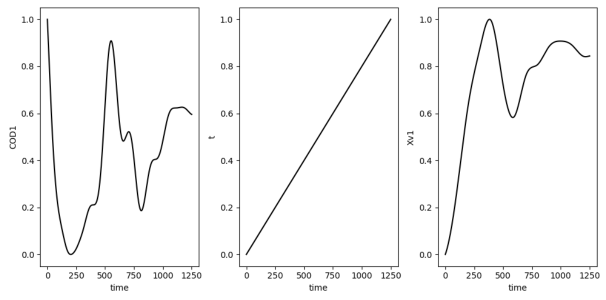


Fig. 2. Individual time plot of each variable in dataset 1

In the same way for the dataset 2

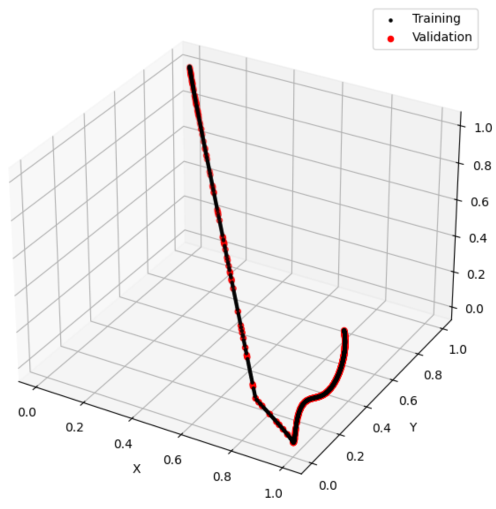


Fig. 3. 3D view of the second dataset

Separately each of the variables has the following behavior:

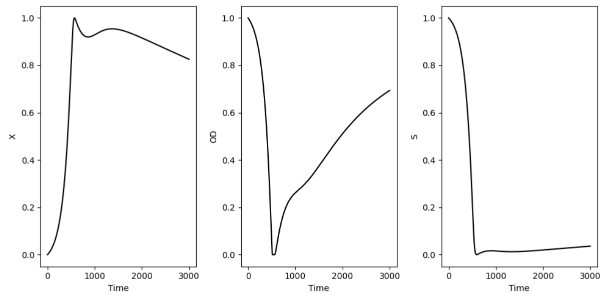


Fig. 4. Individual time plot of each variable in dataset 2

For the LeNET and GAN network there will also be two datasets used, the MNIST dataset and a dataset of numbers made by the students of the Artificial Intelligence course, a sample of how some of the digits by the course of Artificial Intelligence looks.

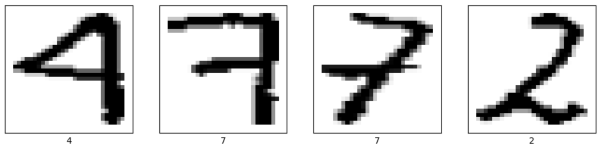


Fig. 5. Sample data of the digits from the AI course

## B. Autoencoder

An autoencoder is a type of artificial neural network used in unsupervised learning and dimensionality reduction tasks. It belongs to the family of feedforward neural networks and is primarily employed for feature learning and representation. The fundamental idea behind an autoencoder is to learn a compressed representation, or encoding, of input data and then reconstruct the original data as accurately as possible from this encoding.

Architecture: An autoencoder typically consists of two main components: an encoder and a decoder.

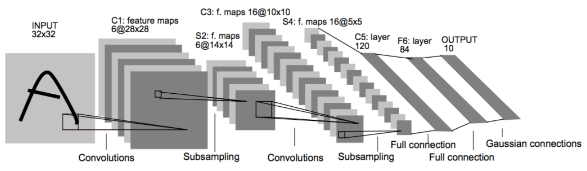
- **Encoder:** The encoder is responsible for mapping the input data into a lower-dimensional representation, often referred to as a latent space or encoding. This mapping is achieved through a series of hidden layers and activation functions. The encoder's role is to capture the most important features and patterns in the input data.
- **Decoder:** The decoder takes the encoding produced by the encoder and attempts to reconstruct the original input data from it. Like the encoder, it consists of hidden layers and activation functions, but it works in reverse. The goal of the decoder is to generate an output that is as close as possible to the input.

## C. Convolutional Neural Networks

Convolutional Neural Networks, commonly known as CNNs, are a class of deep neural networks specifically designed for tasks involving grid-like data, such as images and videos. They have proven to be highly effective in various computer vision tasks, including image classification, object detection, and image segmentation. CNNs are characterized by their ability to automatically learn and extract hierarchical features from input data, making them particularly well-suited for image-related tasks.

They consist of convolutional layers, pooling layers, activation functions and fully connected layers.

The LeNet architecture consists of:



1) *Input layer:* It consists of (32 x 32 x 1) dimension images. As the images are 28 x 28, they are padded with 0s to make them 32 x 32.
2) Convolution Layer 1 : Consists of 6 filters of size 5 x 5. Gives an output of shape (28 x 28 x 6).
3) *Pooling Layer 1:* Max pooling which reduces the input size in half. Output: (14 x 14 x 6).
4) *Convolution Layer 2:* Consists of 16 filters of size 5 x 5. Gives an output of shape (10 x 10 x 16).
5) *Pooling Layer 2:* Max pooling which reduces the input size in half. Output: (5 x 5 x 16).
6) *Fully Conected Layer:* output is then flattened and passed on to a fully connected layer of 120 neurons and then 84 neurons.
7) *Output:* The final output is a softmax output with 10 classes.

## D. Generative Adverserial Networks (GANs)

GANs are a class of deep learning models composed of two neural networks: a generator and a discriminator. The

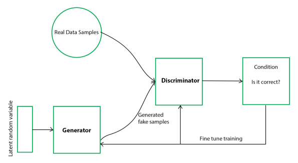


Fig. 6. GANs Architecture

generator aims to create realistic data, such as images, while the discriminator's role is to distinguish between genuine data and data generated by the generator. They engage in a training process where the generator tries to produce data that is indistinguishable from real data, and the discriminator tries to become better at distinguishing real from fake data. This adversarial training process leads to the generator learning to create increasingly realistic data.

GANs architecture is show in 6

### E. *StyleGANs*

StyleGANs are a specific variant of GANs designed for image generation tasks. They are known for their ability to generate high-quality images with fine-grained control over the style and content. StyleGANs introduce the concept of "style" in addition to the traditional GAN framework. The style controls various aspects of an image, such as its color palette, texture, and other artistic elements. This separation of style and content allows for the generation of highly customizable and diverse images.

## III. Experiments

### A. *Autoencoder*

In this investigation, we used two autoencoders to extract important characteristics from our datasets. One autoencoder was created from scratch and self-implemented, while the other was sourced from the TensorFlow Keras module, reflecting a commonly used prebuilt model.

The custom-designed autoencoder allowed us to explore various architectural configurations tailored to our specific dataset and objectives. We experimented with different layer sizes, activation functions, and regularization techniques to optimize feature extraction performance. Our custom autoencoder was tested with a range of encoding and decoding layer sizes, allowing us to assess the impact of different dimensionalities on feature representation.

In addition to our custom implementation, we also employed an autoencoder available through the TensorFlow Keras module. This prebuilt autoencoder provided a baseline for comparison and evaluation. We tested this model under the same experimental conditions to assess its performance relative to our custom-designed autoencoder.

### B. *LeNet*

In our experiments, we utilized the LeNet-5 neural network for digit image classification using the MNIST dataset as a benchmark. First, we trained LeNet-5 on the MNIST dataset to establish a baseline accuracy in classifying handwritten digits (0-9). Subsequently, we assessed its classification performance on a dataset generated by students as part of our research, aiming to test its generalization capability beyond MNIST.

In a reverse experiment, we trained LeNet-5 on the student-created dataset and evaluated its adaptability by testing it on the MNIST dataset. These experiments allowed us to investigate both the model's ability to classify digits in a novel dataset and its adaptability when trained on synthetic data. The results provide insights into LeNet-5's performance across different datasets and highlight the potential of student-generated data for training deep learning models.

### C. *GAN*

In our experiments, we employed Generative Adversarial Networks (GANs) and specifically StyleGAN to generate synthetic data and explore style transfer techniques. Initially, we trained StyleGAN on the MNIST dataset to generate handwritten digit images, demonstrating its ability to create realistic synthetic data.

In a complementary experiment, we applied StyleGAN to recorded videos, exploring its potential for style transfer. This process involved manipulating the artistic style of the generated content while preserving the underlying structure. These experiments showcase the versatility of StyleGAN in generating diverse and creatively modified visual data, emphasizing its potential for both image synthesis and transformation tasks.

## IV. Results

### A. *Autoencoder*

For the self-coded autoencoder, tested with different numbers of neurons, the results obtained are presented in Table I. The autoencoder was configured with ReLU, ReLU, Linear activation functions to maintain a semblance of comparison with the Keras autoencoder.

TABLE I
Results for Self-Coded Autoencoder with Different Numbers of Neurons

| Number of Neurons | Energy Test | Validation Error |
|---|---|---|
| 1 | 0.0580 | 0.2574 |
| 2 | 0.1579 | 0.3661 |
| 5 | 0.0524 | 0.2477 |
| 7 | 0.0511 | 0.2253 |
| 9 | 0.0499 | 0.2385 |
| 12 | 0.0533 | 0.2508 |

The best results were observed when the autoencoder had either 7 or 9 neurons. These configurations demonstrated the lowest energy test values and validation errors, indicating their effectiveness in capturing important features from the datasets. As a visual representation of the autoencoder's performance, we provide 3D predictions of both datasets using the best-performing neuron configuration, as shown in Figure IV-A and Figure IV-A.

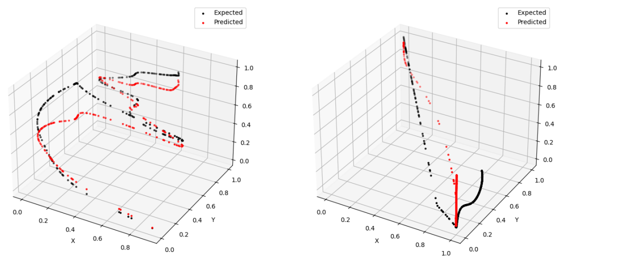


Fig. 7. 3D Predictions of Both Datasets using the Self-Coded Autoencoder

To gain insights into the learning process and the network's ability to capture gradients in the data, we also present visualizations of the gradients for the autoencoder applied to both datasets. These gradient visualizations, displayed in Figure IV-A and Figure IV-A, provide a deeper understanding of how the autoencoder extracts features and information from the input data.

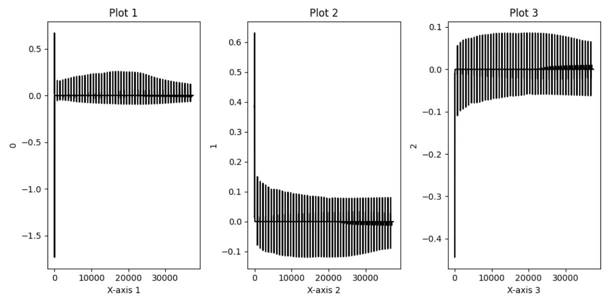


Fig. 8. Gradients for the Self-Coded Autoencoder in the First Dataset

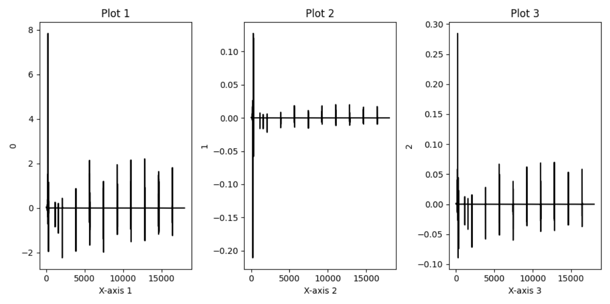


Fig. 9. Gradients for the Self-Coded Autoencoder in the Second Dataset

In parallel, we conducted similar experiments with the Keras module autoencoder, varying the number of neurons in the hidden layers. The results of these experiments are summarized in Table II. Remarkably, the Keras autoencoder demonstrated remarkable performance even with a minimal configuration of 9 neurons, as evidenced by negligible training, testing, and validation losses.

TABLE II
RESULTS FOR KERAS MODULE AUTOENCODER WITH DIFFERENT NUMBERS OF NEURONS

| Hidden Neurons | Training Loss | Testing Loss | Validation Loss |
|---|---|---|---|
| 1 | 0.028 | 0.026 | 0.029 |
| 2 | 0.004 | 0.004 | 0.004 |
| 5 | 0.025 | 0.023 | 0.025 |
| 7 | 0.002 | 0.003 | 0.003 |
| 9 | 2.455e-07 | 4.111e-07 | 2.094e-07 |

To provide a visual representation of the Keras autoencoder's performance, we include 3D predictions of both datasets using the best-performing neuron configuration in Figure IV-A and Figure IV-A.

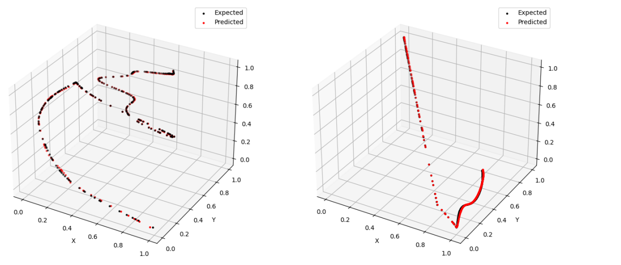


Fig. 10. 3D Predictions of Both Datasets using the Keras Module Autoencoder

## B. LeNet

We begin by visualizing the testing accuracy of the LeNet model during its training process, as depicted in Figure 11. This figure showcases how the model's testing accuracy evolves over the training epochs, providing insight into its learning behavior.

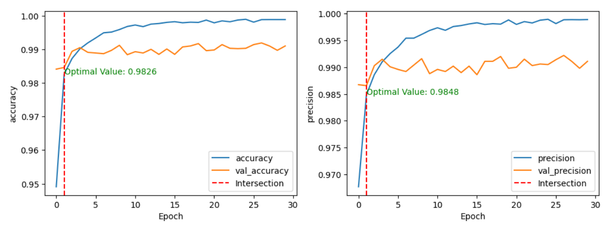


Fig. 11. Testing Accuracy of the LeNet Model During Training

To further evaluate the model's performance, we present the confusion matrices for two datasets: the MNIST dataset (Figure 12) and the numbers dataset provided by the AI course (Figure 13). These matrices provide a comprehensive view of the model's classification results, illustrating its ability to correctly classify digits while also highlighting any misclassifications.

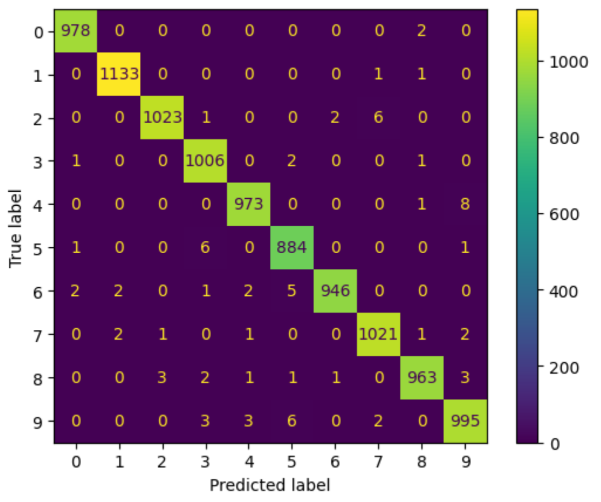


Fig. 12. Confusion Matrix for Validation Data (MNIST Dataset)

The first confusion matrix provides valuable insights into the LeNet model's performance when applied to the MNIST dataset's validation data. Each row in the matrix represents the actual digit classes, while each column represents the predicted classes.

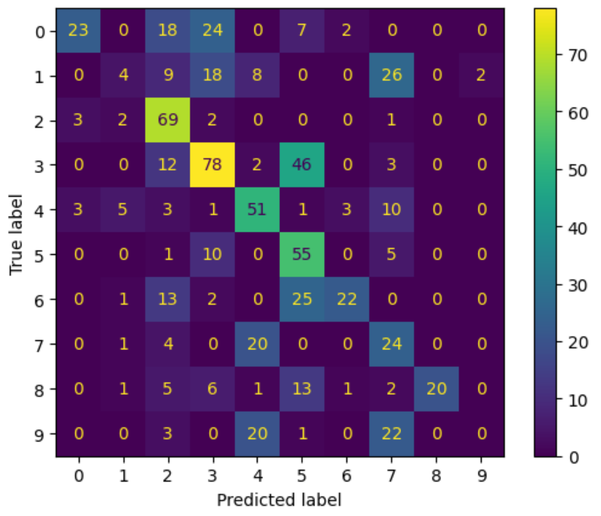


Fig. 13. Confusion Matrix for AI Course Numbers Dataset

Here, we can observe how well the model performs when classifying digits from the AI course dataset. Similar to the previous matrix, it allows us to identify areas where the model excels and potential challenges it encounters. Comparing this matrix to the one for the MNIST dataset can help us understand how well the model generalizes across different datasets and whether any dataset-specific biases exist. We can clearly see how the AI diagonal for this confusion matrix isnt very strong. Indicating a lack of model generalization or a lack of ability in the students handwriting.

Subsequently, we delve into the predictions made by the LeNet model when applied to the AI course digits. Figure 14 showcases a sample of the model's predictions for the AI course digits when it was originally trained on the MNIST dataset. This visualization allows us to assess the model's performance on a different dataset.

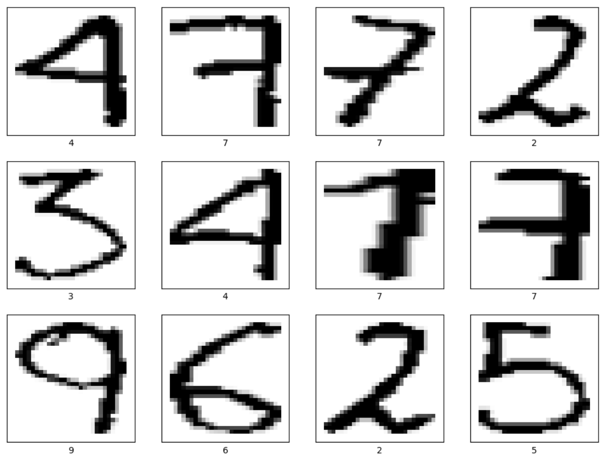


Fig. 14. Predictions for AI Course Digits (MNIST-trained Network)

In this specific section, it becomes apparent that training the LeNet model with the course dataset resulted in a significant lack of generalization capabilities. This limitation can likely be attributed to several factors, including the limited diversity of digits, the presence of unusual or outlier digits, and variations in students' handwriting styles.

The decision to train the LeNet model on samples from the AI course dataset and subsequently evaluate its performance on the MNIST dataset aimed to assess the model's adaptability to new, diverse data sources. However, the results, as depicted in the confusion matrix in Figure 15, illuminate the challenges encountered when attempting to generalize the model trained on the AI course dataset. These challenges manifest as suboptimal performance, likely due to the unique characteristics of the AI course dataset that differ significantly from MNIST.

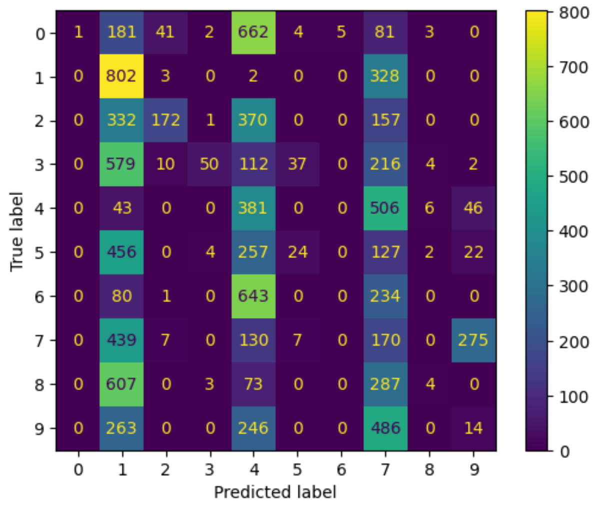


Fig. 15. Confusion Matrix for Validation Data (AI Course-trained Network)

Furthermore, the predictions made by the AI Course-trained LeNet model when applied to digits from the MNIST dataset (Figure 16) underscore the model's inability to adapt seamlessly. These predictions offer valuable insights into the model's performance when faced with the standardized MNIST dataset, where it struggles to achieve the high accuracy it exhibited with the original training dataset.

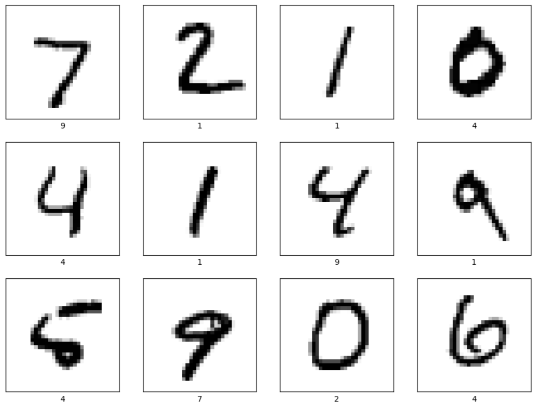


Fig. 16. Predictions for MNIST Digits (AI Course-trained Network)

## C. GAN

To assess the performance of our Generative Adversarial Network (GAN), we begin with a visual inspection of the synthetic digits generated by the GAN model trained on the MNIST dataset. Figure 17 displays a selection of these generated digits, providing an overview of the model's ability to produce realistic handwritten digits.

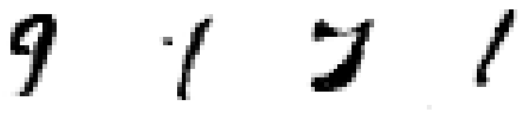


Fig. 17. Sample of Digits Generated by the GAN Model (MNIST Dataset)

Next, we delve into the training process by examining the losses incurred by both the discriminator and the generator throughout the epochs. Figure 18 illustrates the evolution of these loss functions, shedding light on the GAN's training dynamics and its convergence towards a stable equilibrium.

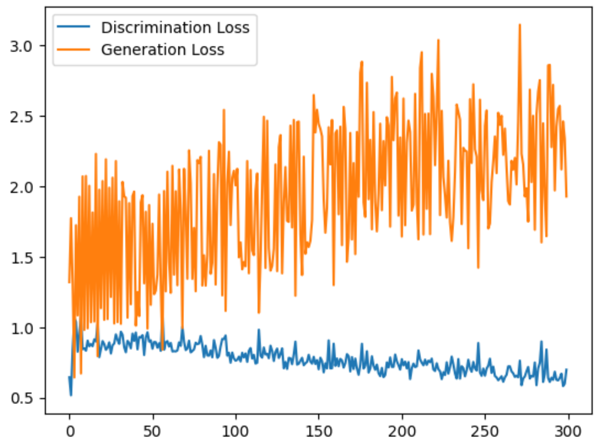


Fig. 18. Discriminator and Generator Loss Evolution During Training

Additionally, we conducted a comprehensive evaluation of the GAN's discriminator's ability to discern the authenticity of digits from the AI course dataset. The table in Table III provides a probabilistic assessment of how "real" each of the numbers from the AI course dataset was determined to be by the GAN's discriminator. Entries with a probability greater than 0.5 indicate that the discriminator classified the digit as "real," while entries with probabilities less than 0.5 represent digits classified as "fake."

TABLE III
DISCRIMINATOR'S CLASSIFICATION OF AI COURSE DIGITS

| Probability | AI Course Dataset Classification |
|---|---|
| $> 0.5$ | 0.67 (Classified as "Real") |
| $< 0.5$ | 0.33 (Classified as "Fake") |

## D. StyleGAN

For the StyleGAN we used Tensorflow Hub module called "Arbitrary Image Stylization v1-256/2", a pre-trained StyleGAN model. With this model, the artwork taken as base was Edvard Munch "The Scream", Figure 19 displays this artwork.

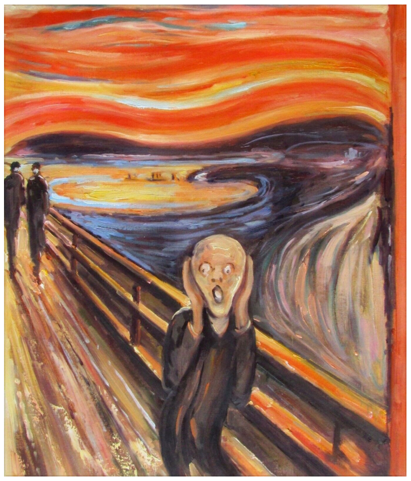


Fig. 19. Artistic Style Applied to an Image Using StyleGAN

Next, we showcase the model's results. Figure 20 provides a comparison between an original frame from a video and a frame generated by our StyleGAN model.

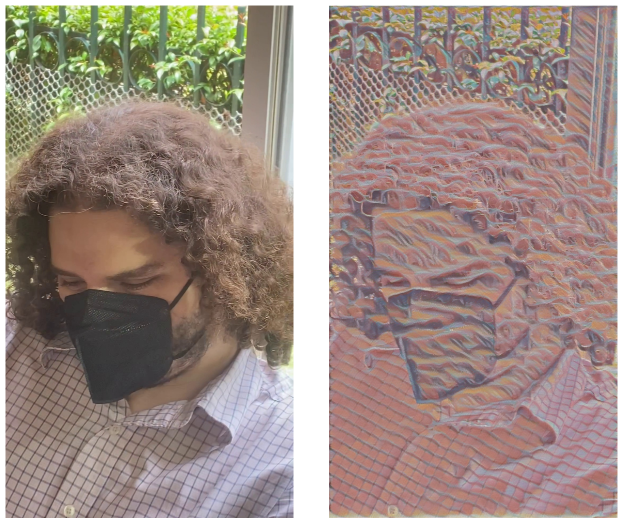


Fig. 20. Comparison of an Original Video Frame and a StyleGAN-Generated Frame

## V. Discussion

### A. Autoencoder

For the self coded autoencoder we can see that the best model was the one with seven neurons even though there wasnt a considerable difference with the model with 9 neurons. Furthermore, when exploring dimensionality reduction autoencoders or compression autoencoders, the input dimensional space and amount of hidden layers was too low to be able to capture the shape of characteristcs of the data, thus a higher dimensional in the hidden layer or interpolation model was required, this meant that it would be harder for the model to be able to reduce the noise, however from the lack of jagged lines and "noise" shown in the data plots for both datasets in II-A and 3. Thus a higher dimensional model cas ideal in this case to be able to capture the interaction of the data. In further analysis it would be an interesting experiment testing a dataset with more noise to be trained in a compression autoencoder as to diminish the noise and learn the hidden characteristics of the data. The self coded autoencoder had a remarkably hard time trying to learn the characteristics of the data, this is probably due to three things, an optimization function as to better optimize the learning process, since it tended to go a lot of times settle in local optima that did not represent the model well. The bias gradient calculation presented issues sometimes. Finally the activation functions were chosen to be high energy function as to not lead to "gradient death" in training, however from what we can see in IV-A and IV-A the gradients tended to die quickly (the spikes are showing the shape of the data, spiking only with the higher values and then inmediately dying). The combination of the problem in the bias calculation and the activation functions were the probable cause of this, since the bias should have helped keep the "gradients alive".

Finally we can see the keras autoencoder learns much more accurately the shape and patterns of the data much more accurately. However the amount of neurons in the hidden layers in II showed a similar result to what what was obtained with the self coded autoencoder in I.

### B. LeNet

In the training and testing phases of our LeNet5 CNN using the MNIST database, the model exhibited exceptional performance, achieving consistently high accuracy and precision throughout the training epochs. The confusion matrix (12) further underscores this excellence. However, the model faced challenges when classifying digits from our custom course database as seen in 13. This performance discrepancy can be attributed to several factors.

One key factor is the substantial dissimilarity between the digits in the MNIST dataset and those in our custom testing dataset. Notably, the writing styles and digit representations have evolved over time. Moreover, our custom dataset's zoomed-in format introduced specific challenges, potentially altering the spatial correlations between pixels and influencing model errors. It is observed that the model has a specifically hard time classifying the nines and the one, for the nine seems to have been confused with the fours and sevens most likely due to the similarity in the shape. The ones and the sevens also tended to be confused, between them.

In the inverse experiment however we can see the lack of generalization capabilities from a model trained with the Artificial Intelligence Dataset (15). The model was capable of identifying well the test dataset after it finished training, however when it was tested against the test set of the MNIST dataset, we can see how the complete lacked generalization capabilities. Though it is worth noting how the activation space for one was quite ample, leading many of the predictions to be one.

### C. GAN

17 shows how the digits were generated in the last epoch of the model (300). Even thoush not all the digits looked like discombobulated lines, it was clear the model had much more improvement to go through still. Though some specific number patterns seem to have been caught, meaning you can recognize the number well enough to determine which number it is. Further improvement of the GAN model ould involve setting up a different metric more specialized in vector pixel differentiation. furthermore, in 18, we can see the discriminator and generator loss. We can see that their behavior is wildly oscillating, so it could be thought that it may stabilize in a more significant number of epochs. We can see that despite oscillating, both decrease, i.e., both the generator and the discriminator are meeting their objectives of reducing the discriminator error when evaluating the generated samples and reducing its error in the classification of the samples, respectively.

We can think of the experiment of using the discriminator to determine if the digits by the AI course were ”real” or ”fake” as incomplete as we would want first a model that consistently generates more digit-like outputs to then understand a bit more of the handwriting capabilities of the students in the Artificial Intelligence and be able to say whether or not an AI thinks the students are capable of actual writing.

### *D. StyleGAN*

The use of the pretrained model to change each frame of the video to that style showed some patterns of how the actual network was trained. First we can see the network was trained more with human faces in te fact that the hairs had extreme detail in the video compared to other things, you could observe how the network had the activation space focused more in the persons. Another remarkable thing was that teh model was trained in more pastel and red colors, because through different experiments with different styles the red hue dominated in almost every case, thus indicating the activation space also having a preference towards pastel red hues.