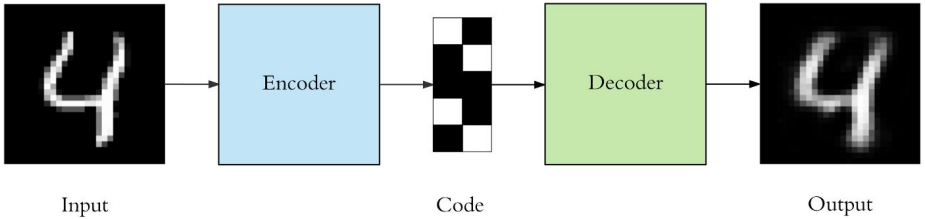
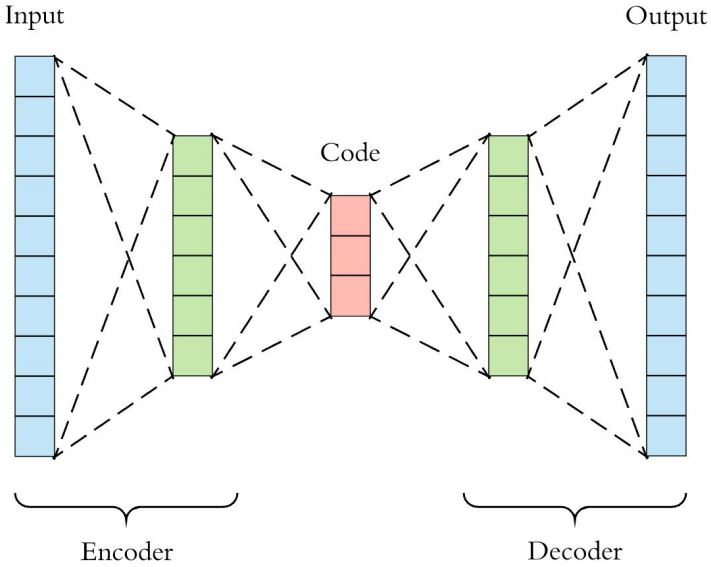
TP5 - *Deep Learning*

- De Simone, Franco - 61100
- Dizenhaus, Manuel - 61101
- Cornidez, Milagros - 61432

Introducción: Autoencoders

- Redes de tipo *Feed Forward* donde se busca que **input == Output**
- Reducción de la dimensionalidad ➜ "*Espacio latente*"
- Particularidad de los autoencoders ➜ Estructura con *encoder* y *decoder*





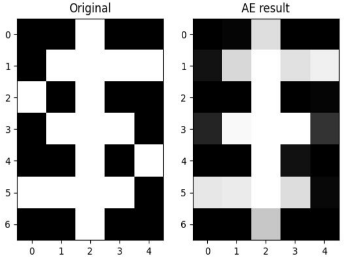
Ejercicio 1: Imágenes binarias de una *tipografía*.

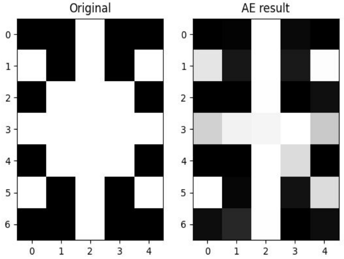
Planteo del problema

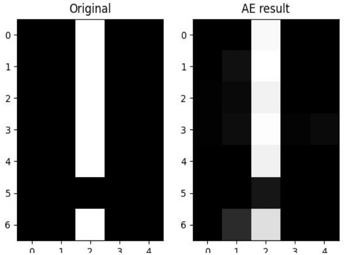
- Fuente de datos: Archivo *“fonts.h”* con letras en formato *hexadecimal*.
- Objetivo: “Diseñar un *autoencoder* que permita aprender y representar los símbolos del alfabeto propuesto.”
- Experimentación:
  - Arquitectura del autoencoder: ¿Cantidad de *capas ocultas*?
  - Tamaño del conjunto de entrenamiento: ¿Hay capacidad de generalización?
  - Exploración del espacio latente
- Implementación de un *Denoising Autoencoder*.

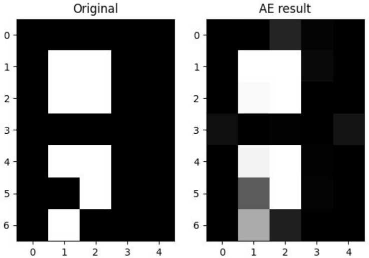
Resultados

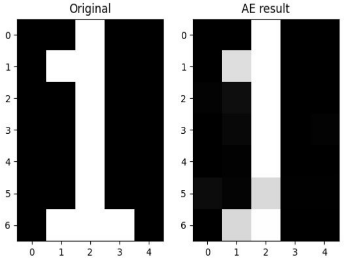
- Arquitectura [35, 20, 2, 20, 35]
- Cantidad de iteraciones: 10000
- Eta: 0,0005
- Error: 4,983

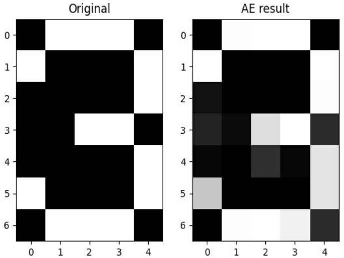


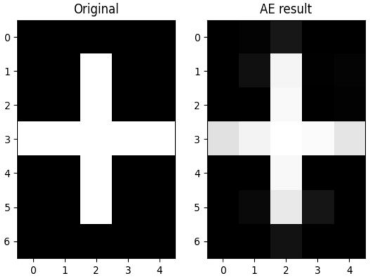


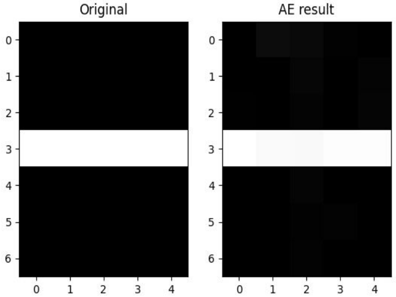






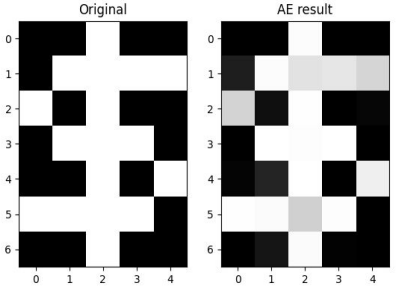


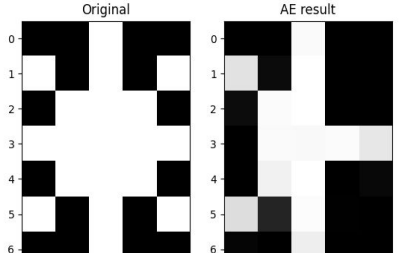


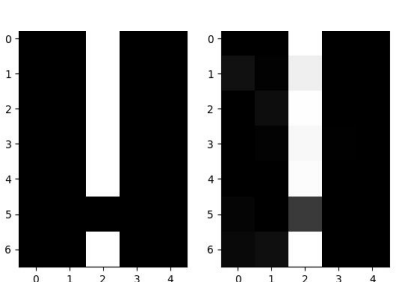


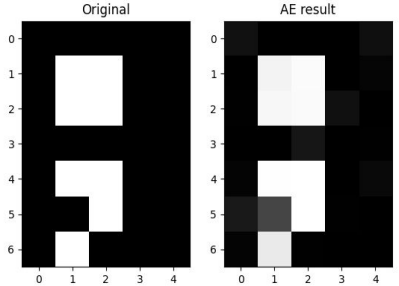
Resultados

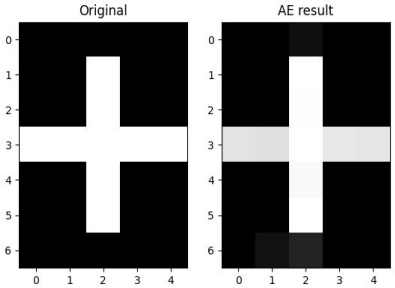
- Arquitectura [35, 15, 10, 2, 10, 15, 35]
- Cantidad de iteraciones: 10000
- Eta: 0,0005
- Error: 3,329

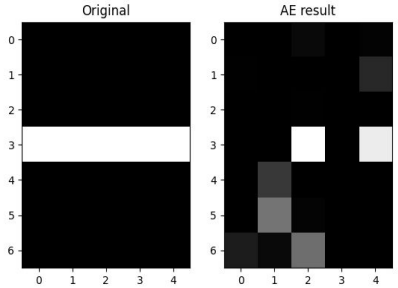


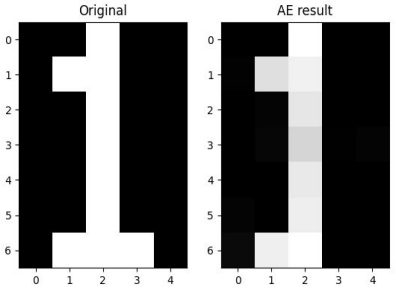


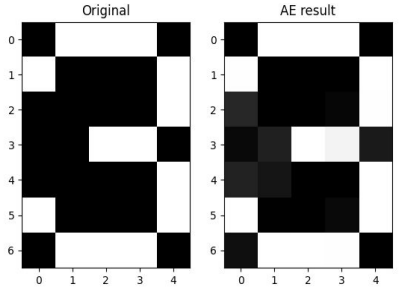






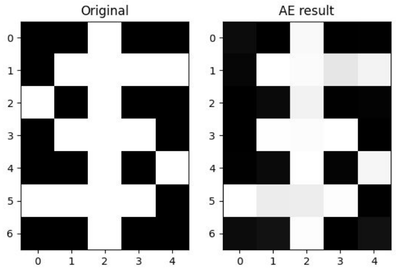


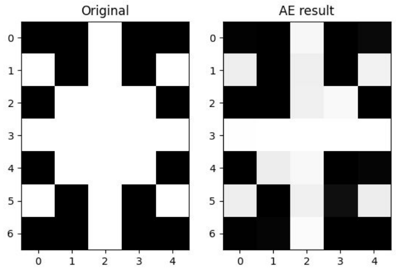


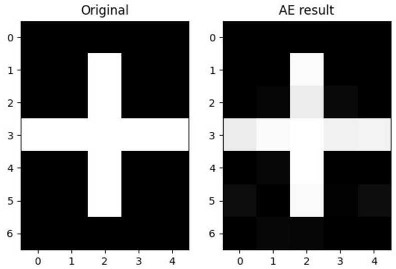


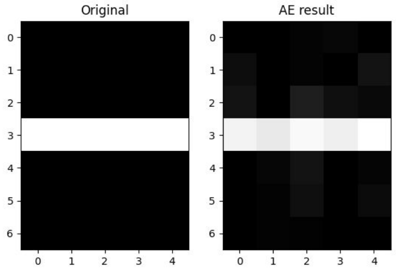
Resultados

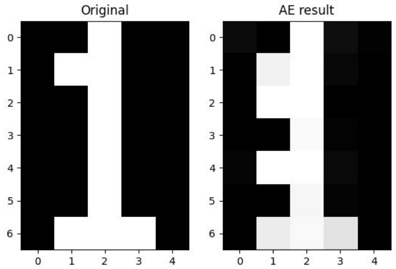
- Arquitectura [35, 25, 20, 10, 2, 10, 20, 25, 35]
- Cantidad de iteraciones: 10000
- Eta: 0,0005
- Error: 2,515

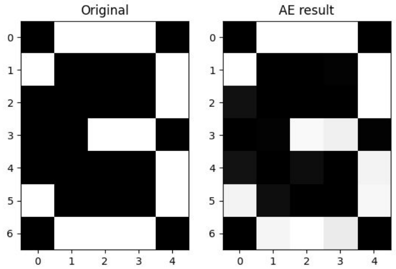






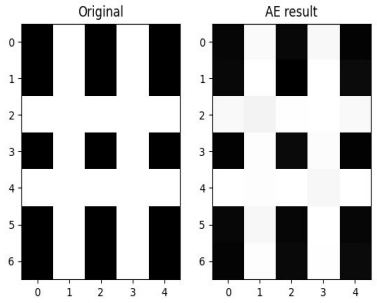




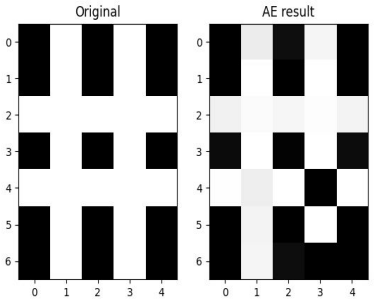


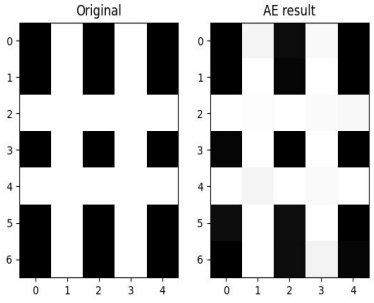
¿Y probando distintos subconjuntos?

Train set: 4

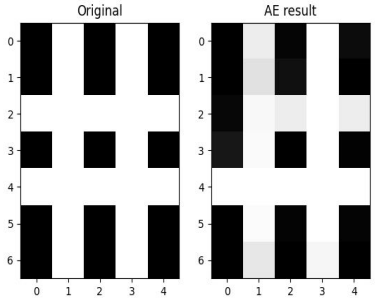


Train set: 16





Train set: 8

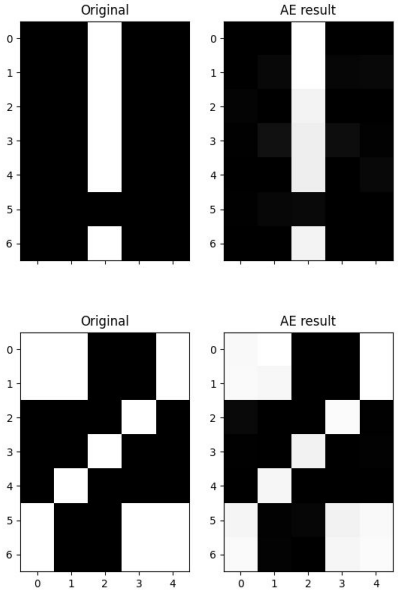


Train set: 32

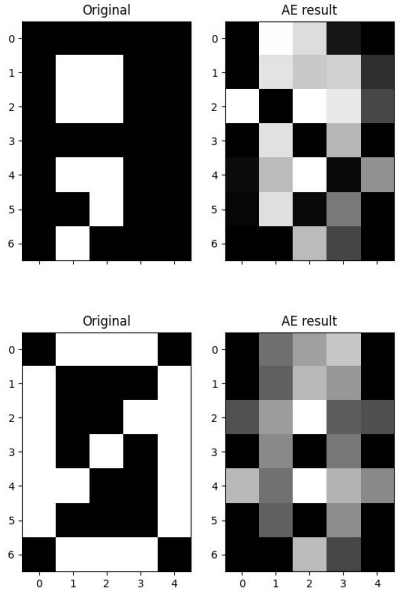
Capacidad de generalización

Tamaño conjunto train: 8

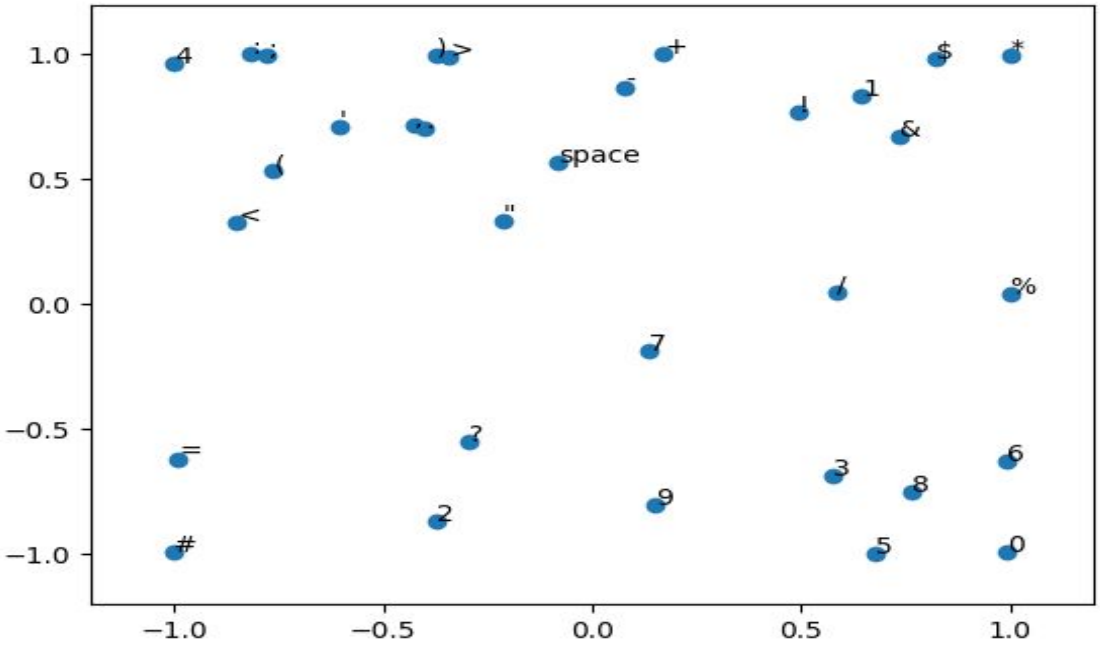
Train:



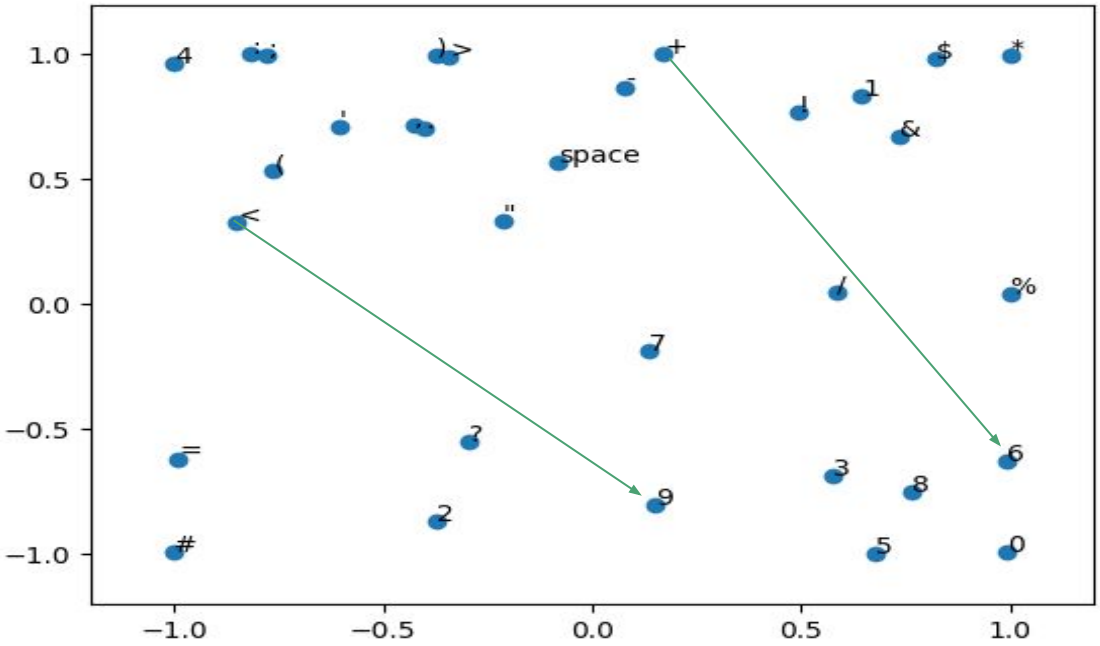
Test:



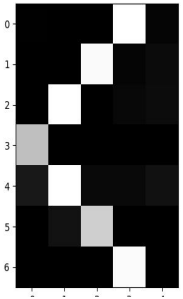
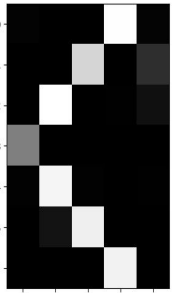
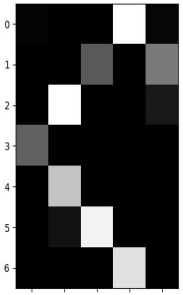
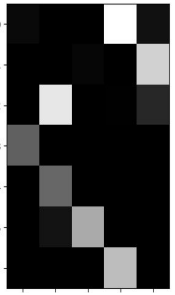
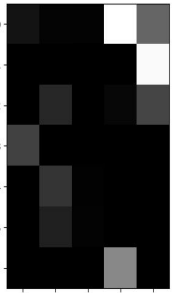
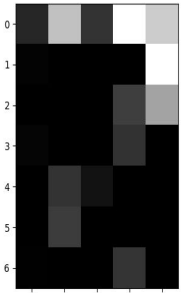
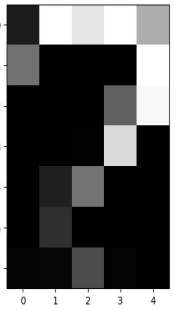
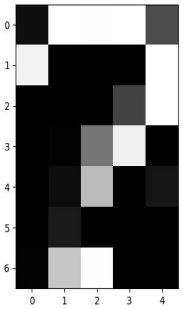
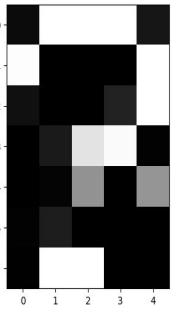
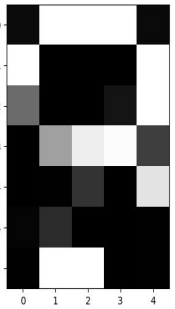
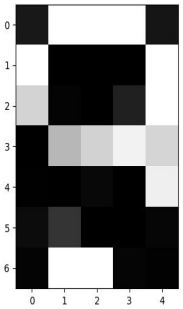
Espacio latente: 2 dimensiones



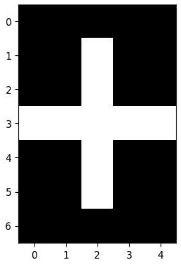
Exploración del Espacio Latente

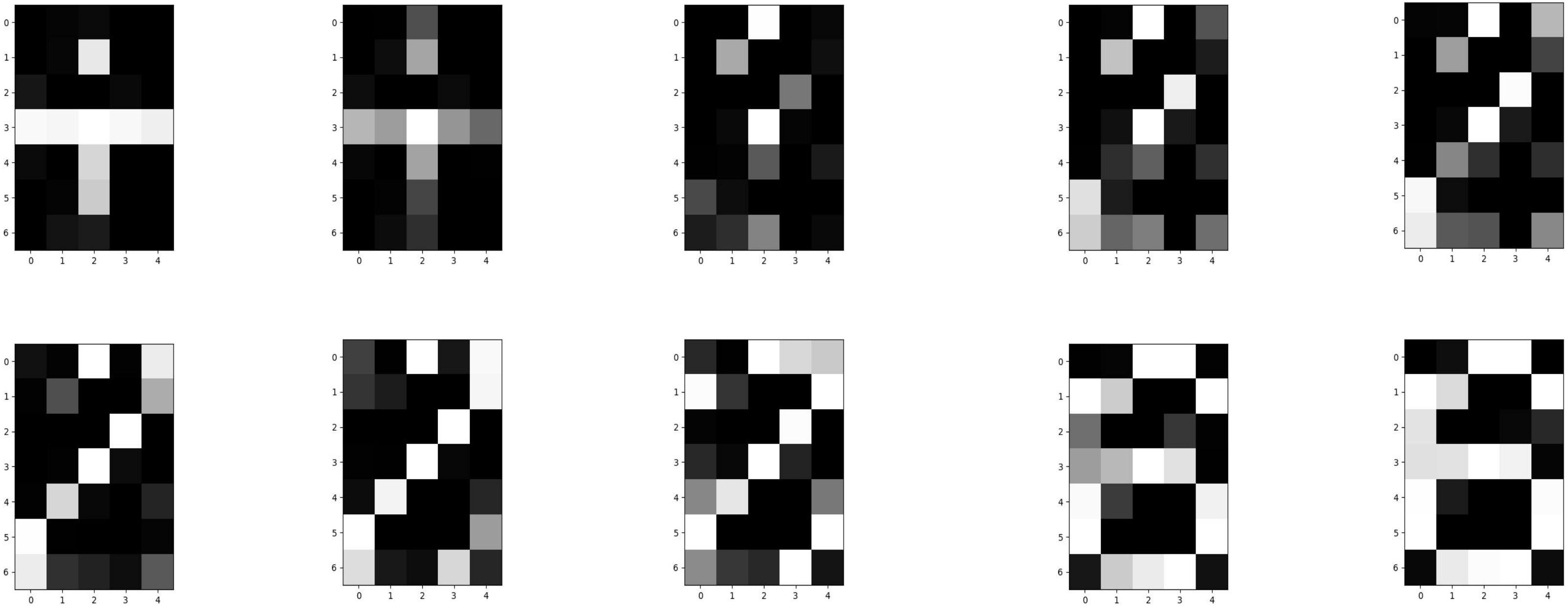


Exploración del Espacio Latente

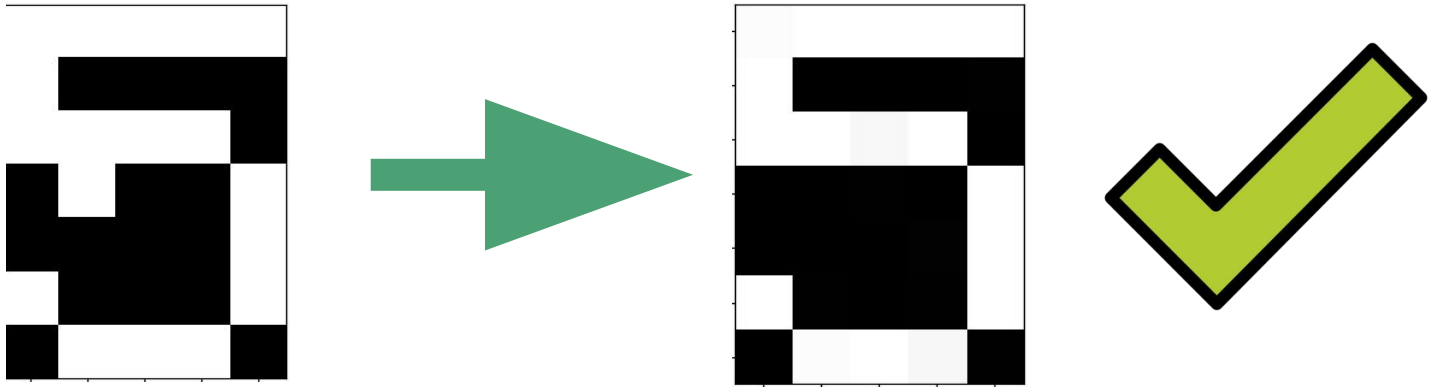
Exploración del Espacio Latente



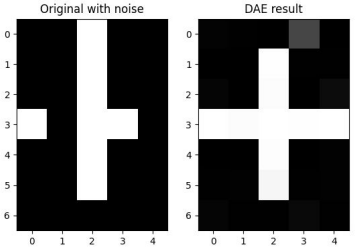


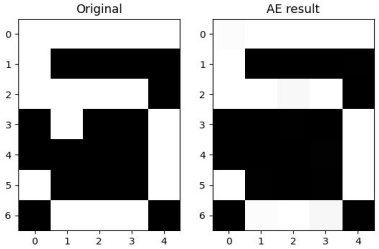
Denoising Autoencoder

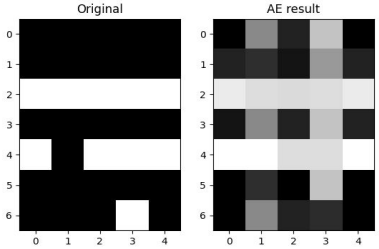
El *Denoising Autoencoder* se utiliza para quitar el “ruido” (perturbaciones indeseables) de un conjunto de datos. En este caso, se entrenará a la red para que, ante la aparición de una letra con una pequeña perturbación, sepa devolver la letra original con el menor ruido posible.

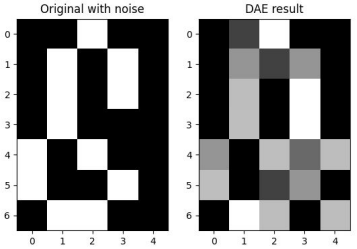


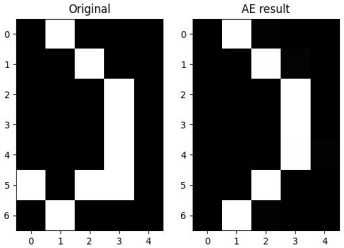
Denoising Autoencoder

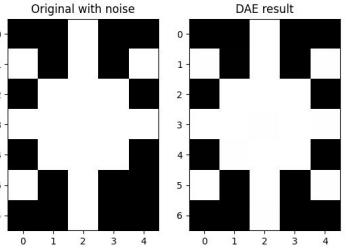


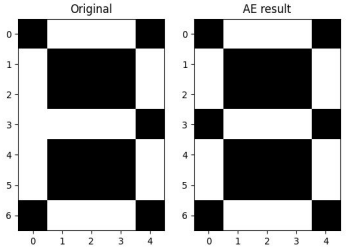


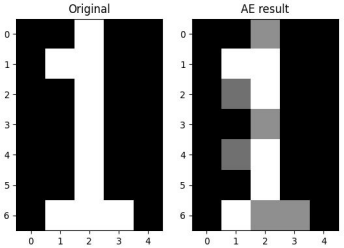








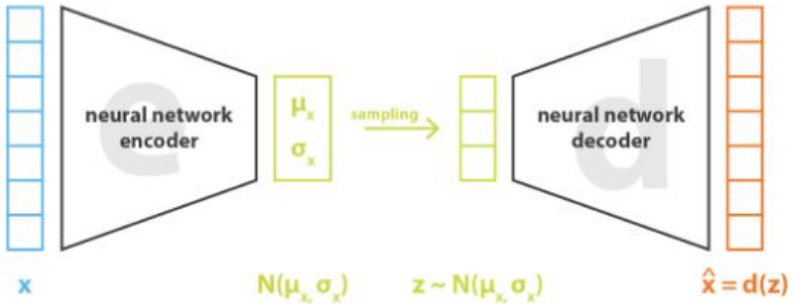




Ejercicio 2: Implementación de un Autoencoder Variacional Simple

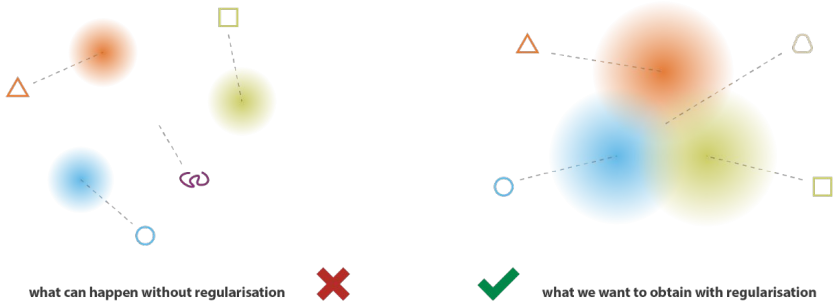
¿Qué son los autoencoders variacionales?

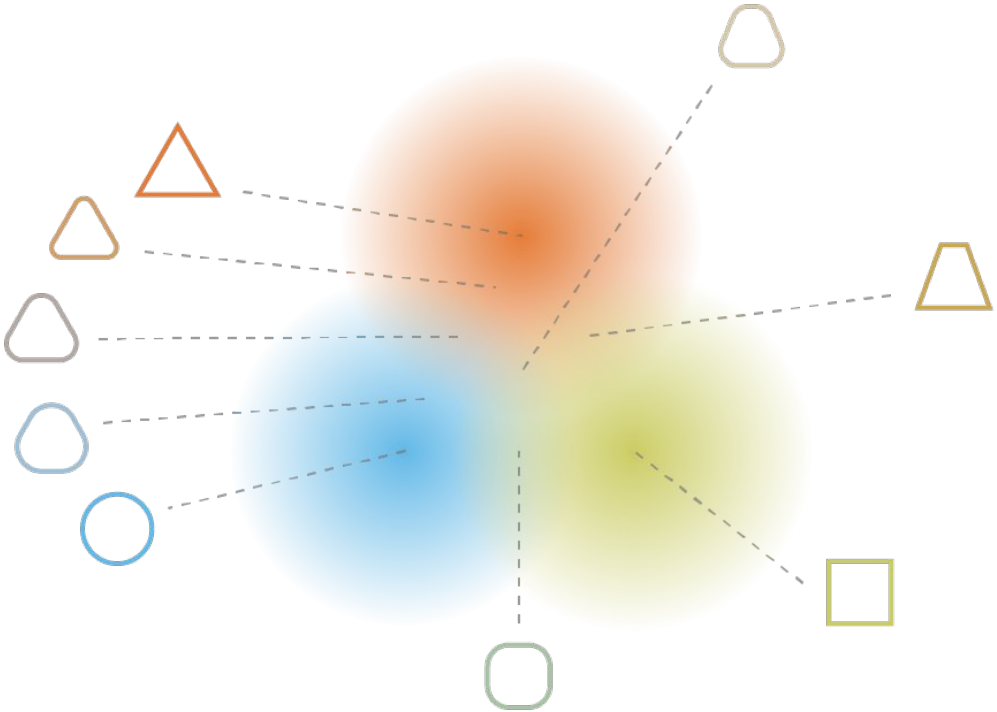
- Autoencoders ➜ **Peligro de overfitting**
- VAEs ➜ Buscan *regularizar* el entrenamiento para evitar *overfitting* y asegurar que el espacio latente tenga buenas propiedades generativas.
- En vez de codificar el input como un punto, se codifica como una distribución normal a lo largo del espacio latente.
- La regularidad esperada del espacio latente tiene dos propiedades centrales:
  - *Continuidad:* Dos puntos cercanos se deben decodificar de manera similar
  - *Completitud:* Un punto sampleado debe dar información valiosa del contenido



Pequeña observación del autoencoder variacional simple

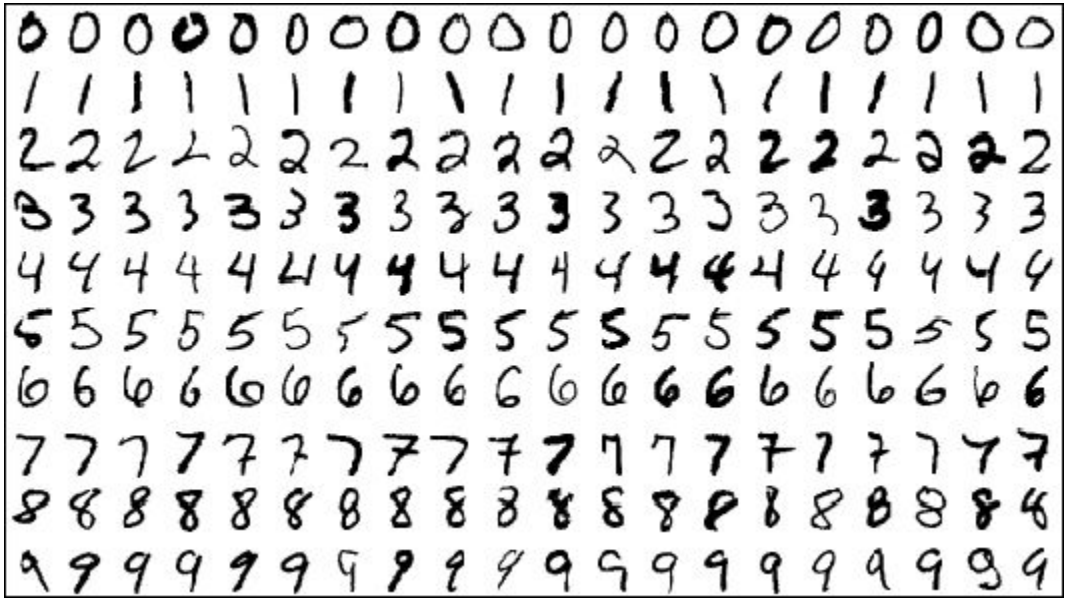
- Definimos que la distribución del espacio latente era *normal*
- Puede ocurrir que, si el término regularizador no está bien definido, el modelo puede tender a una *varianza* muy chica (acercándose al overfitting) como también devolver distribuciones con *medias* que estén muy lejanas en el espacio latente.
- Para evitar esto, hay que regularizar la matriz de covarianza y la media de las distribuciones retornadas por el encoder, tratando que se acerquen a una distribución normal



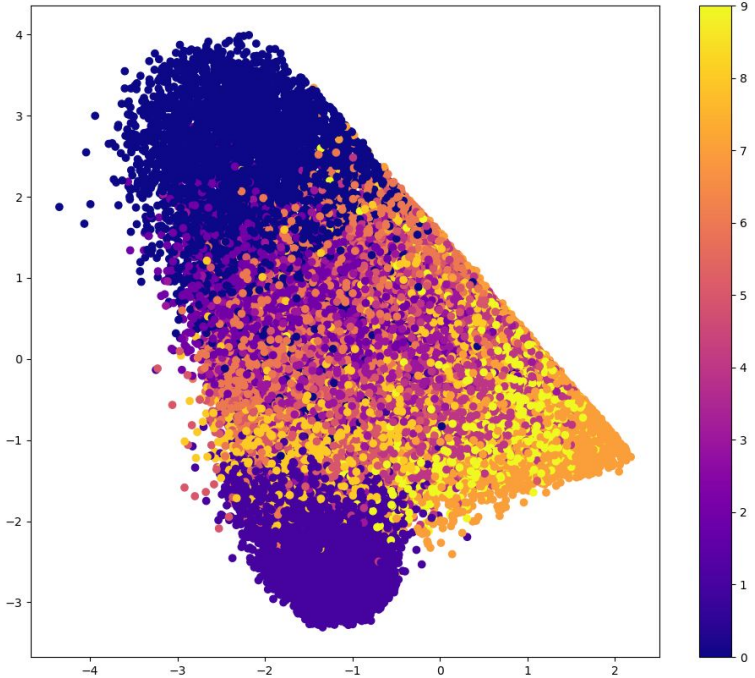


Dataset escogido: MNIST font

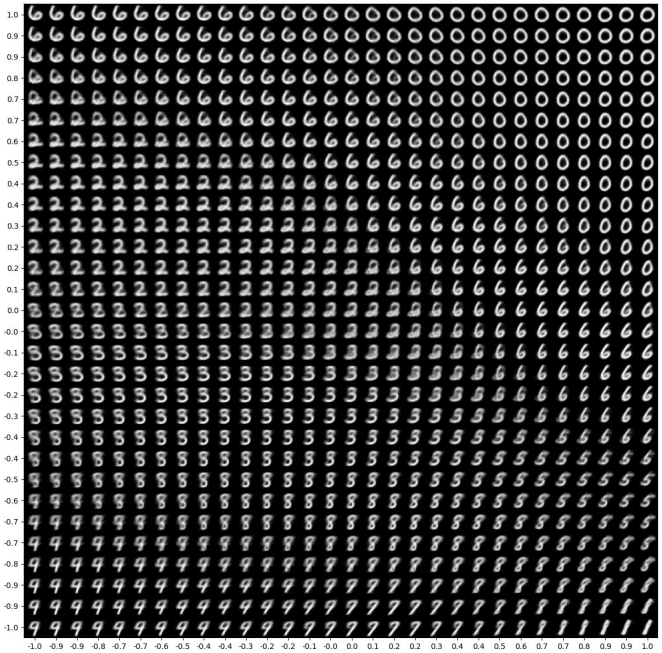
- Conjunto de 60.000 dígitos, en una matriz de 28x28, escritos a mano

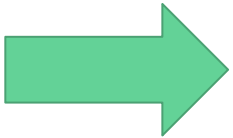


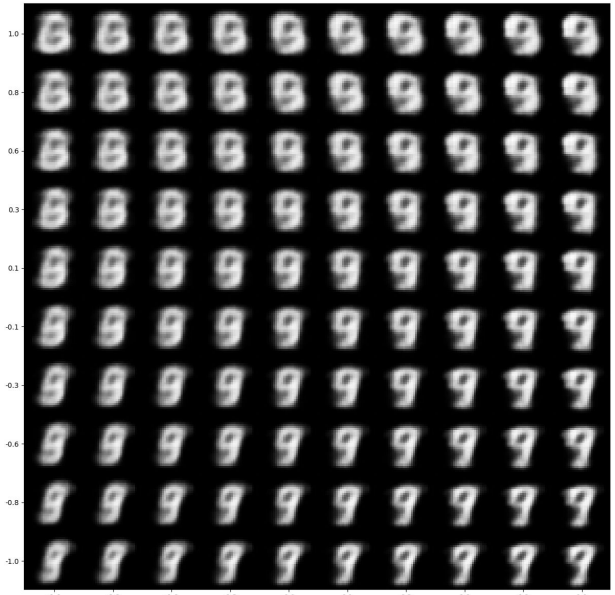
Representación del espacio latente



Dígitos generados por el VAE



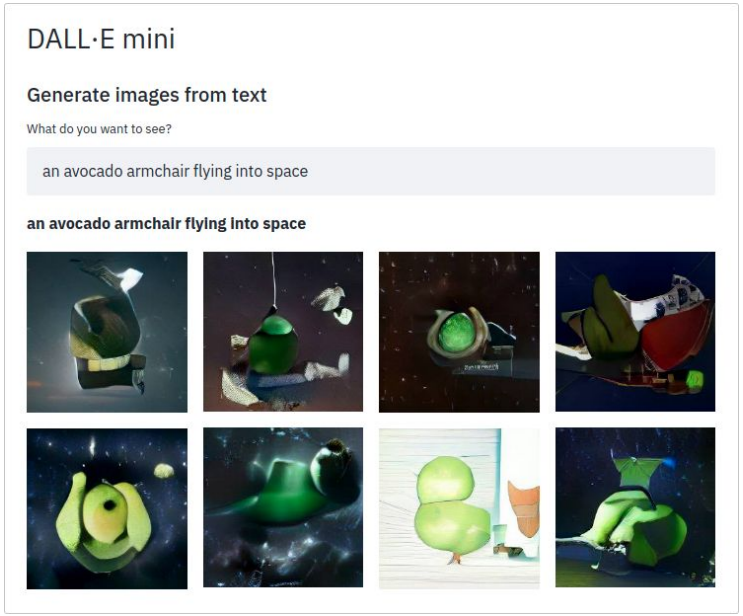


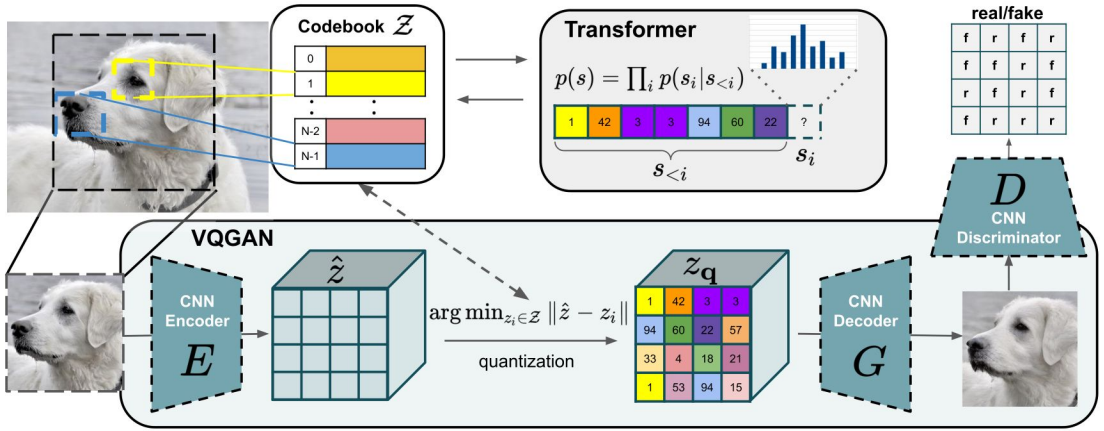


Información adicional

Información interesante: Tendencias actuales

- **Dall-e mini** ➜ Tecnología que es furor en las redes sociales actualmente, está basada en estructuras que estudiamos





Información interesante: Tendencias actuales

This person does not exist ➜ Genera, con una Generative Adversarial Network (GAN), rostros humanos de personas que no existen

- StyleGAN: Presentado por NVIDIA en 2018, y lanzado al público en 2019



Conclusiones

Conclusiones

- El VAE logra reconstruir la entrada de manera efectiva
- Tiempos de ejecución extremadamente altos para pocas iteraciones
- Cercanía en espacio latente