SIA - TP5

Grupo 7

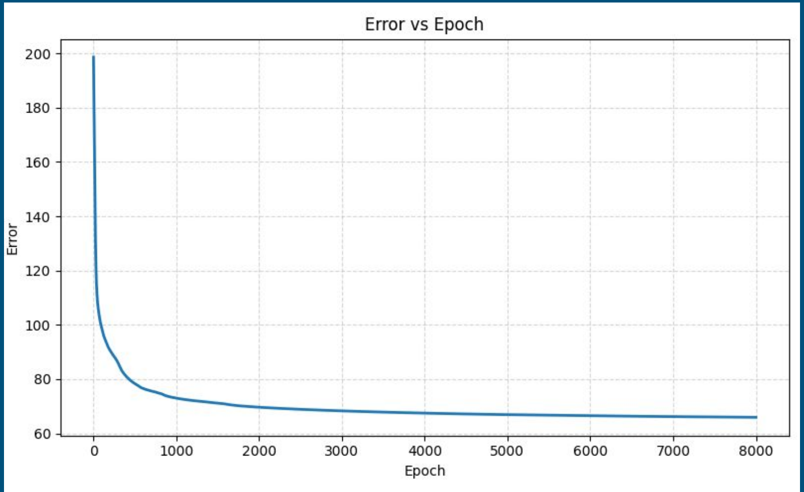
Tomás Scheffer - 63393
Matías Sapino - 61067
Tobías Pugliano - 62180
Luca Bloise - 63004

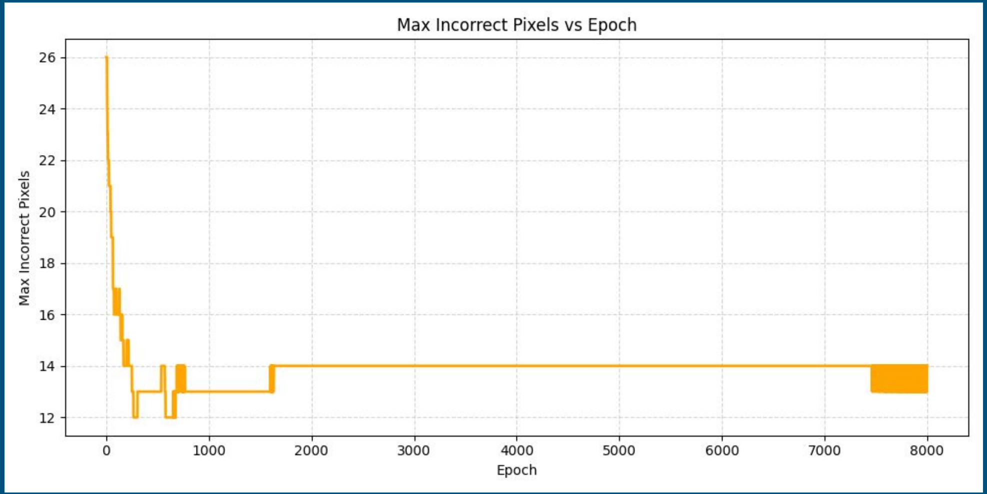
Autoencoder

Arquitectura de red e hiperparametros

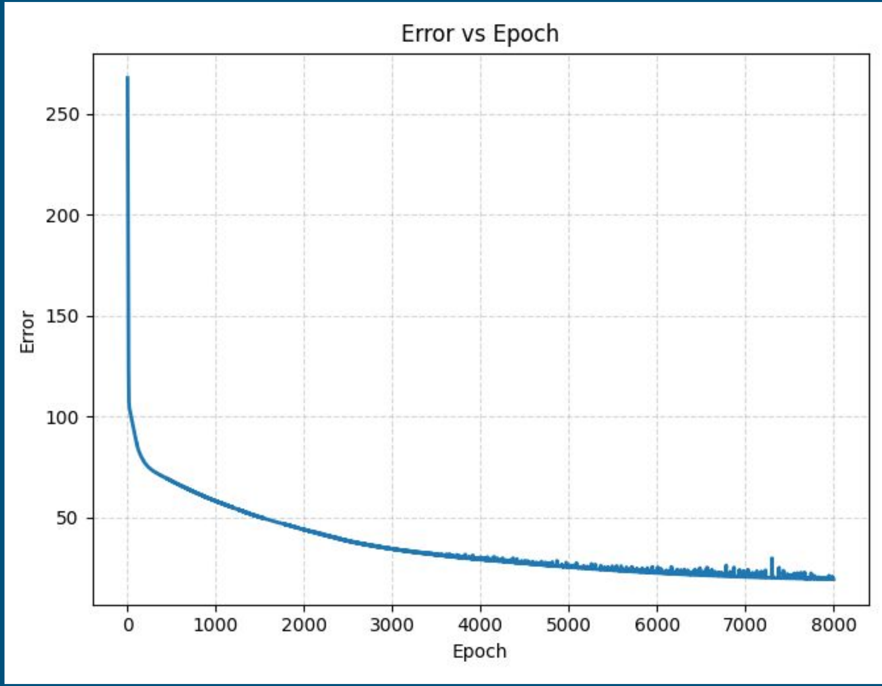
- Tomamos como base la implementación del perceptrón multicapa realizada en el TP3
- En dicho TP se obtuvo que el mejor optimizador para el perceptrón es Adam por lo cual optamos por utilizar el mismo
- Comenzamos con una arquitectura pequeña y vamos aumentando la cantidad de capas intermedias del encoder y decoder, ya que en el TP3 notamos que a mayor cantidad de capas intermedias menor resultaba el error
- Las siguientes pruebas se realizaron con un learning rate $\eta = 0.000985$

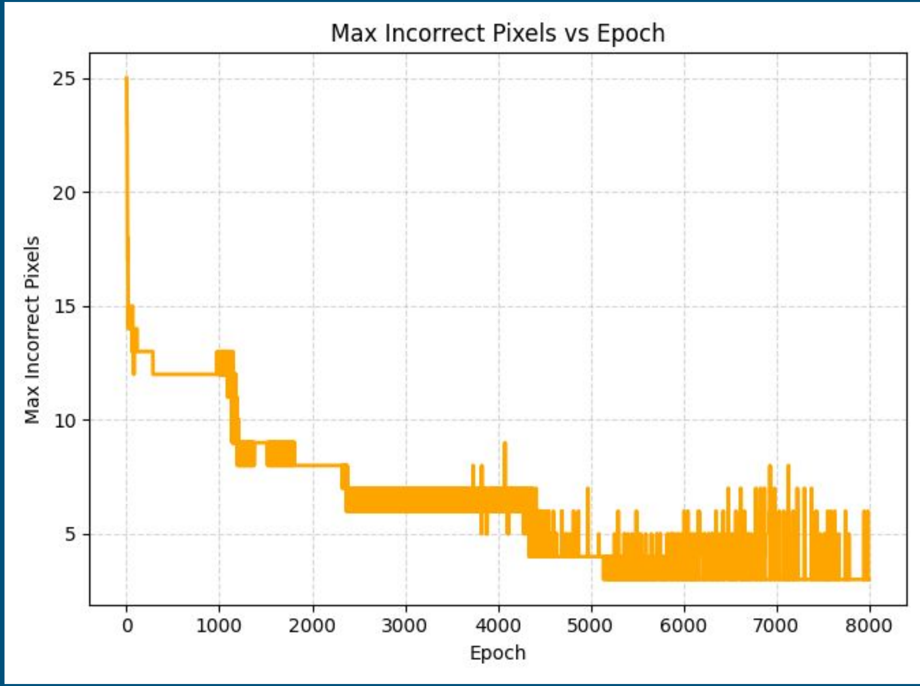
Modificando la estructura: [35, 2, 35]



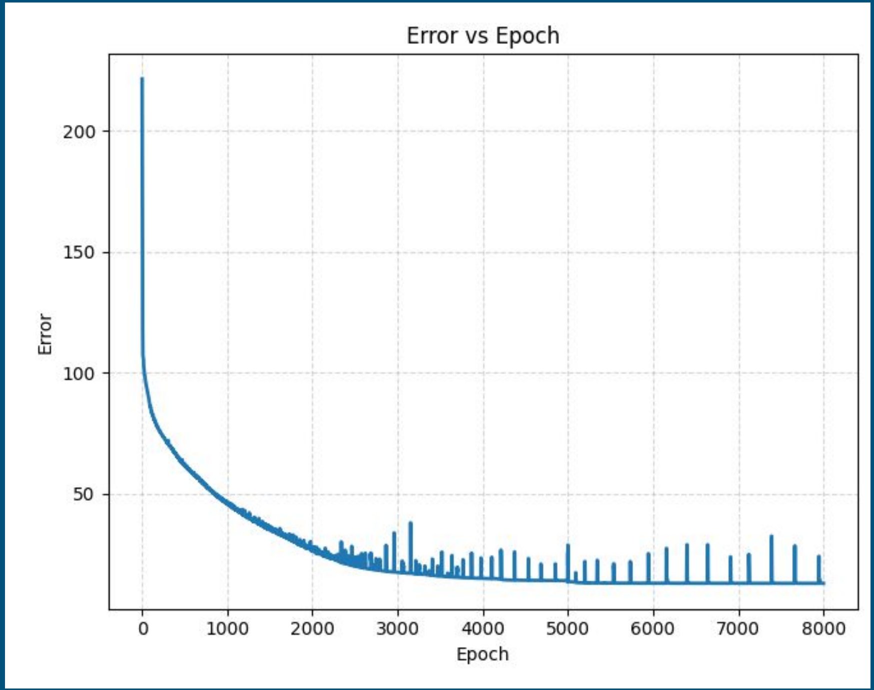


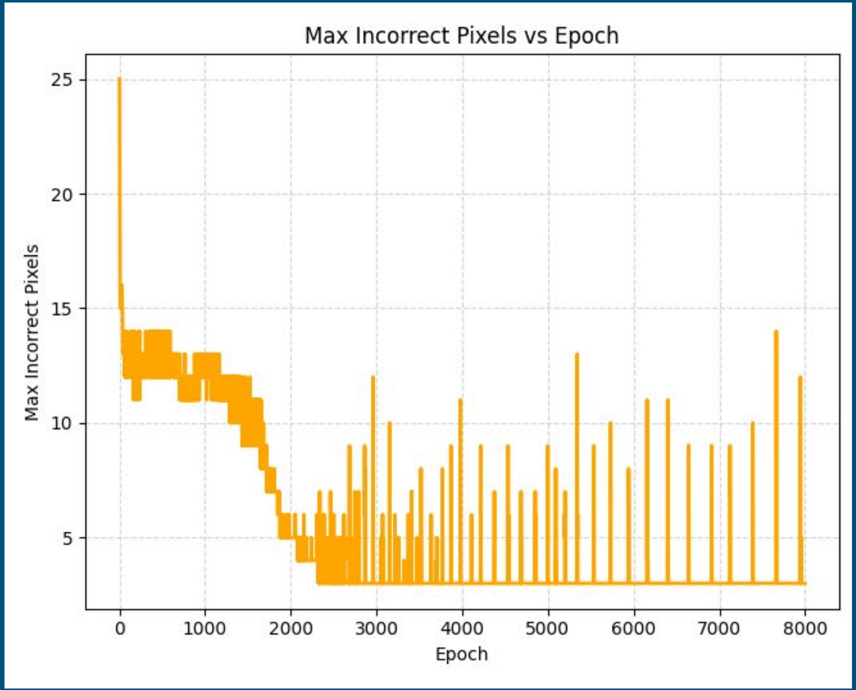
Modificando la estructura: [35, 16, 2, 16, 35]



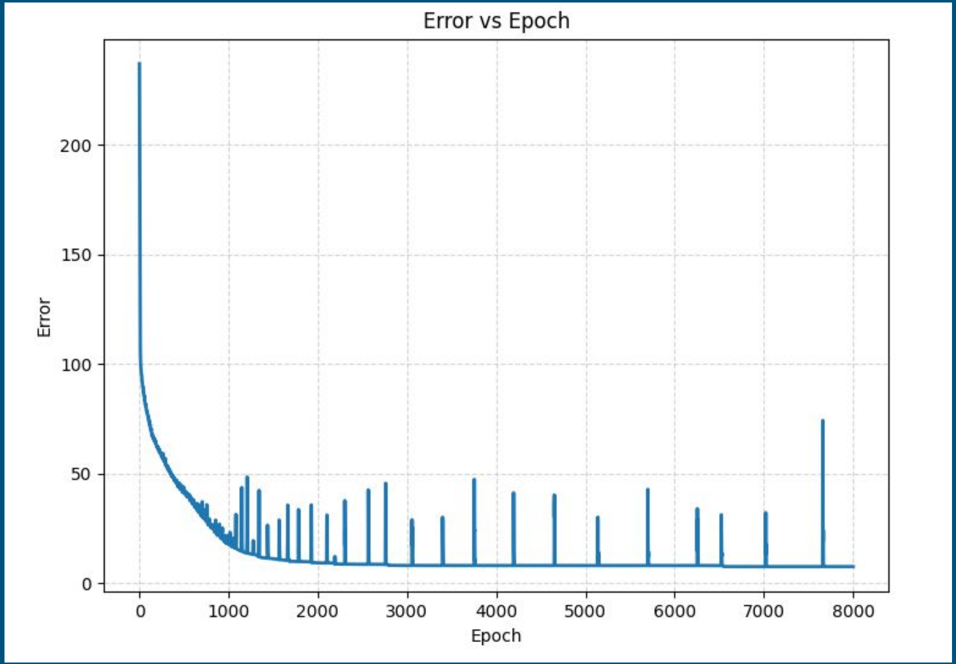


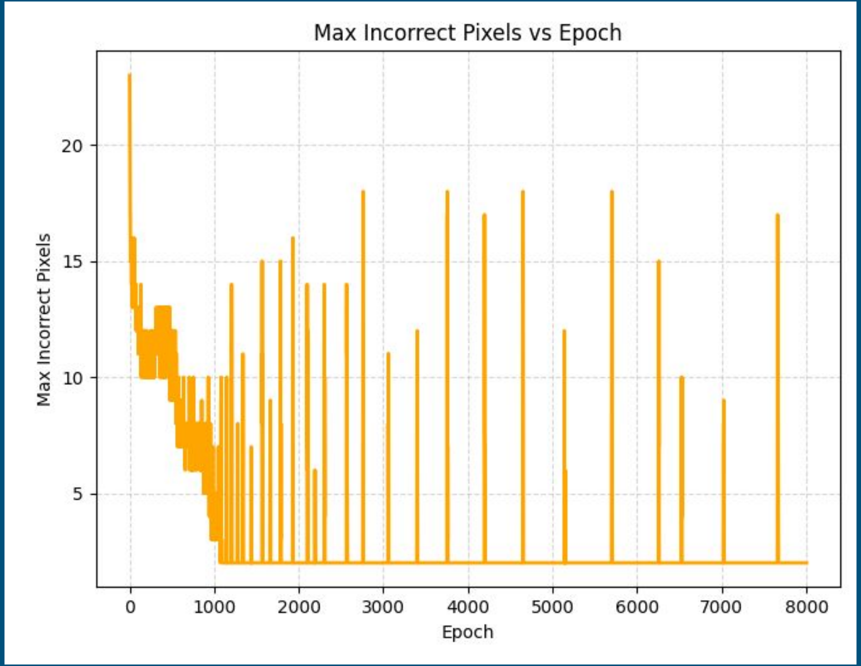
Modificando la estructura: [35, 24, 16, 2, 16, 24, 35]



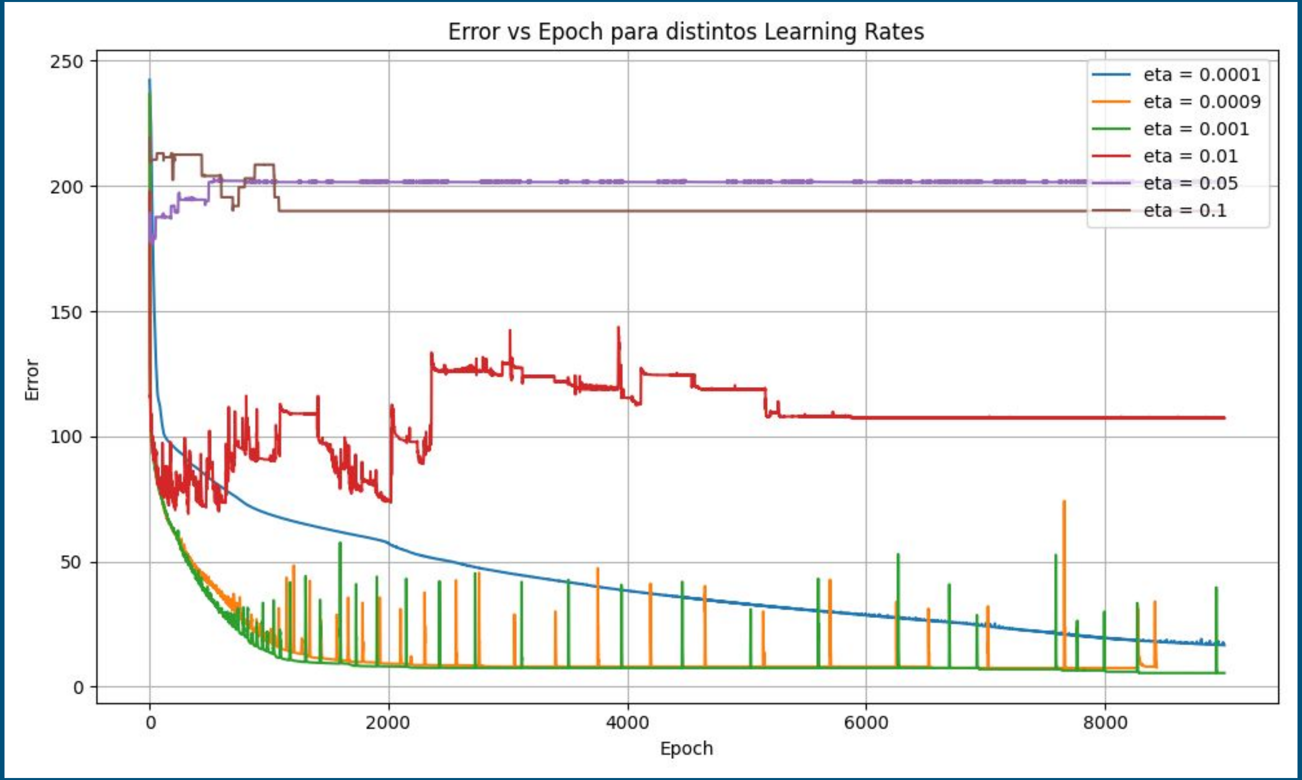


Modificando la estructura: [35, 32, 24, 16, 2, 16, 24, 32, 35]





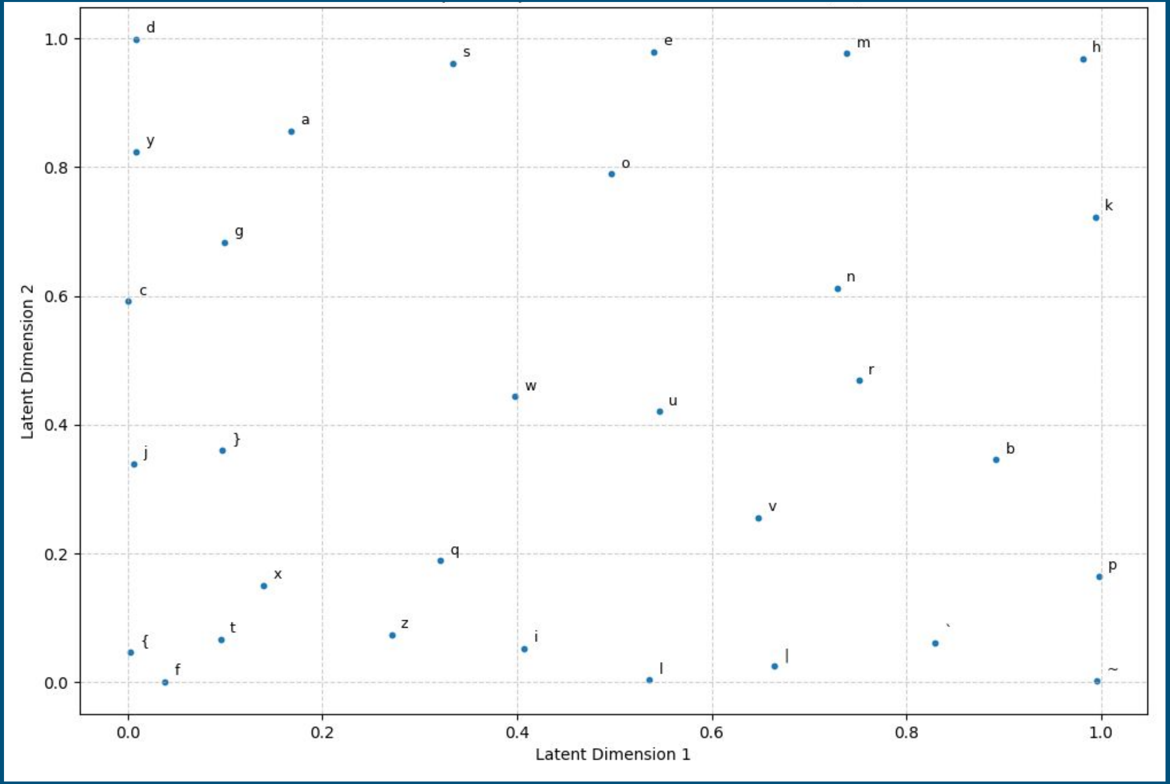
Modificando el learning rate



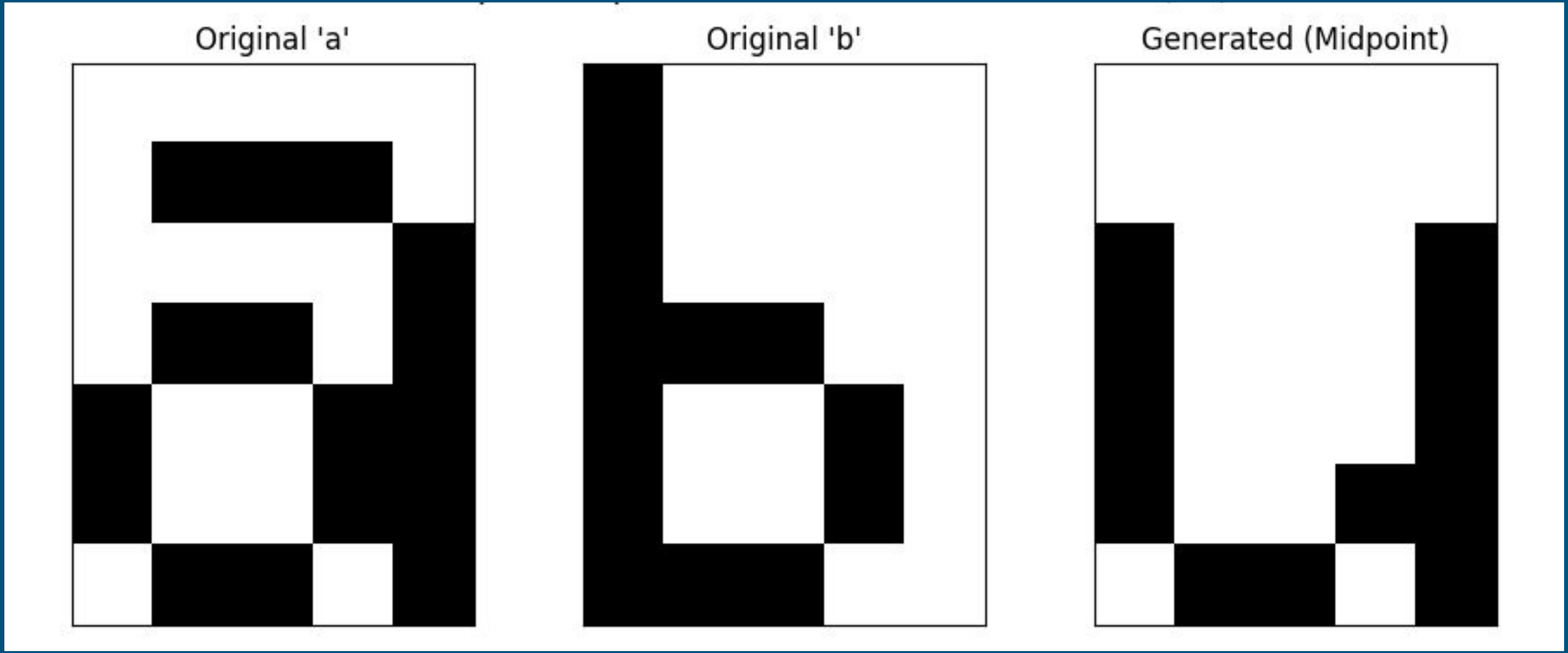
Arquitectura de red e hiperparametros

- Encoder y Decoder de 4 capas conectados por un espacio latente de 2 neuronas ([35, 32, 24, 16, 2, 16, 24, 32, 35])
- Como función de activación se utilizó la sigmoide
- Como learning rate se optó por η = 0.0009

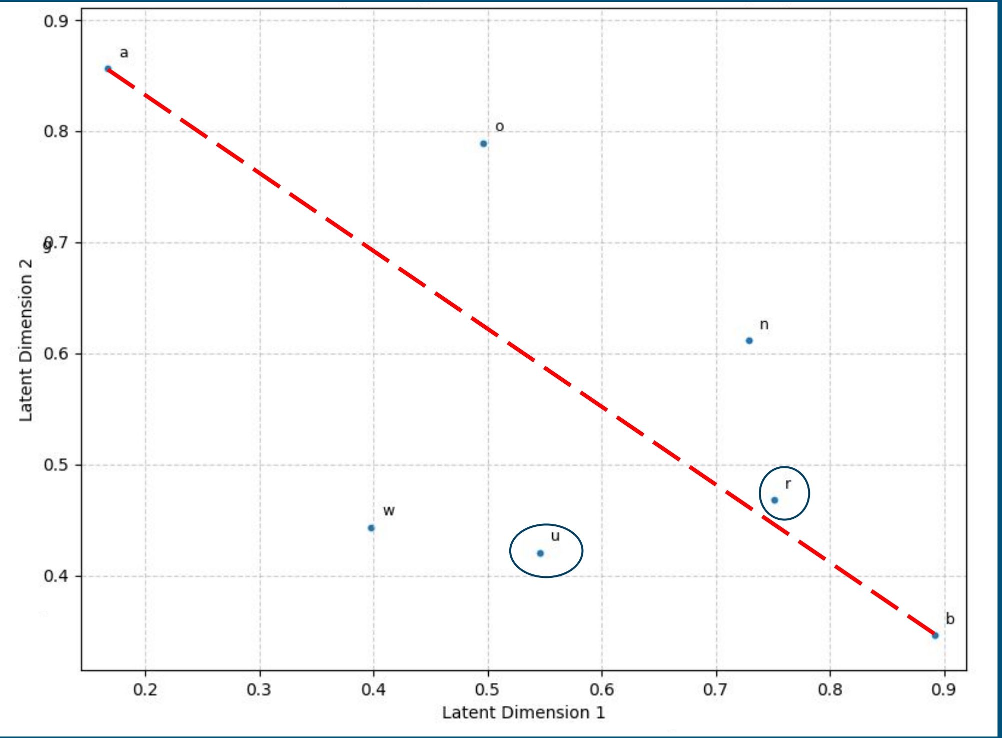
Espacio Latente resultante

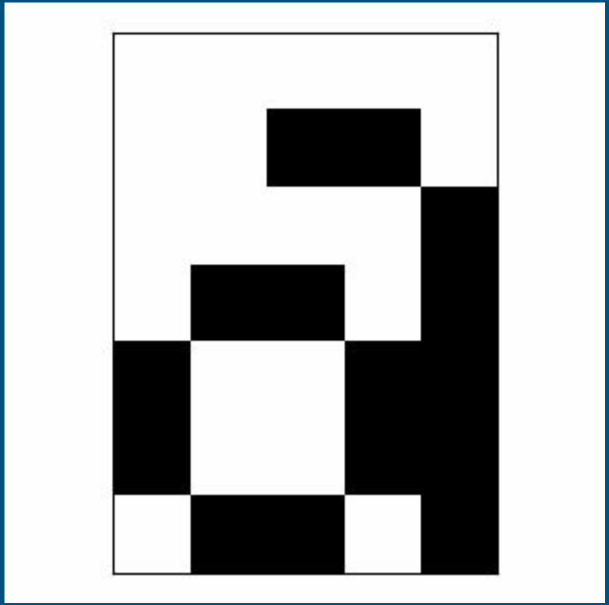


Generación de una nueva letra



Generación de una nueva letra

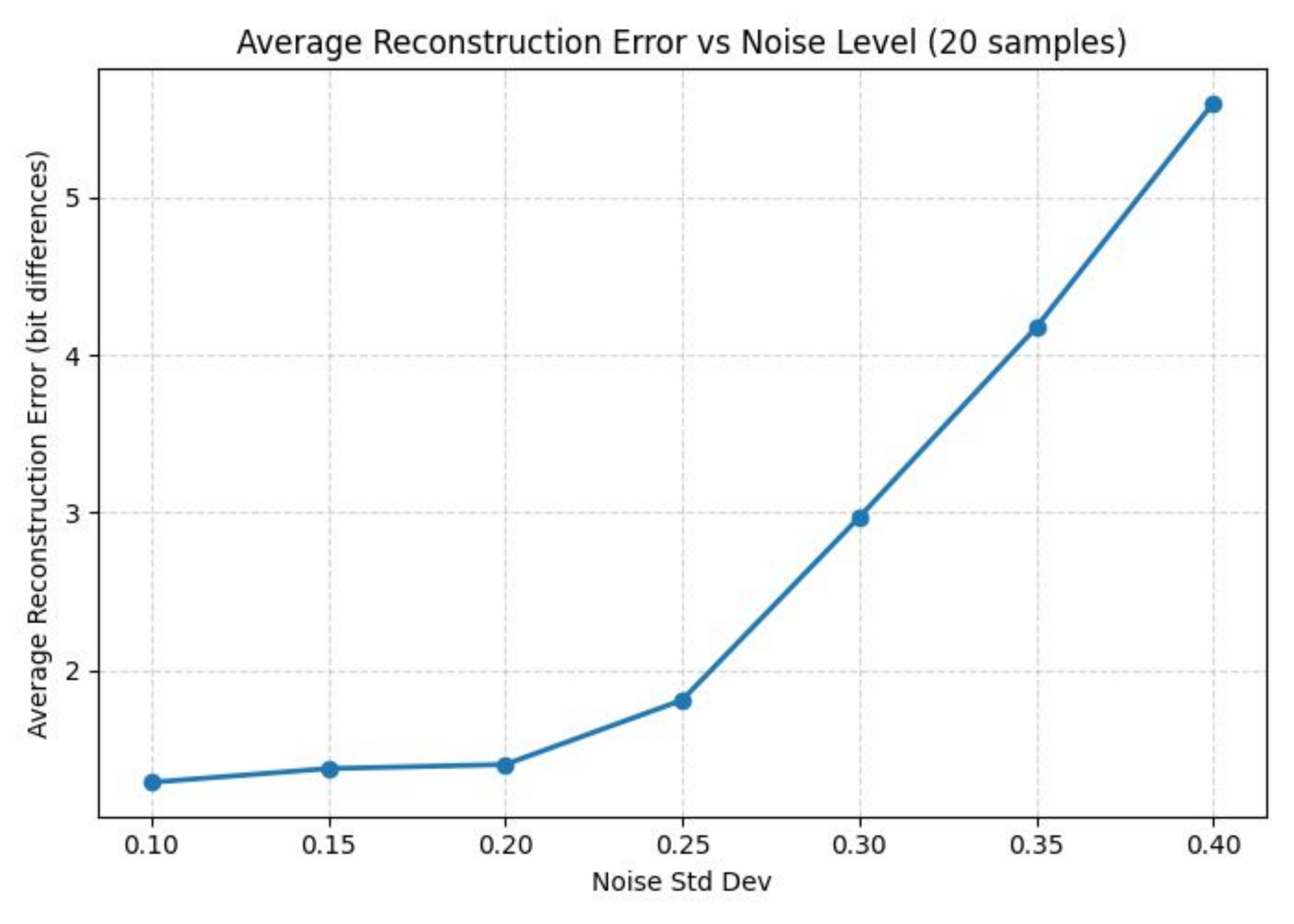


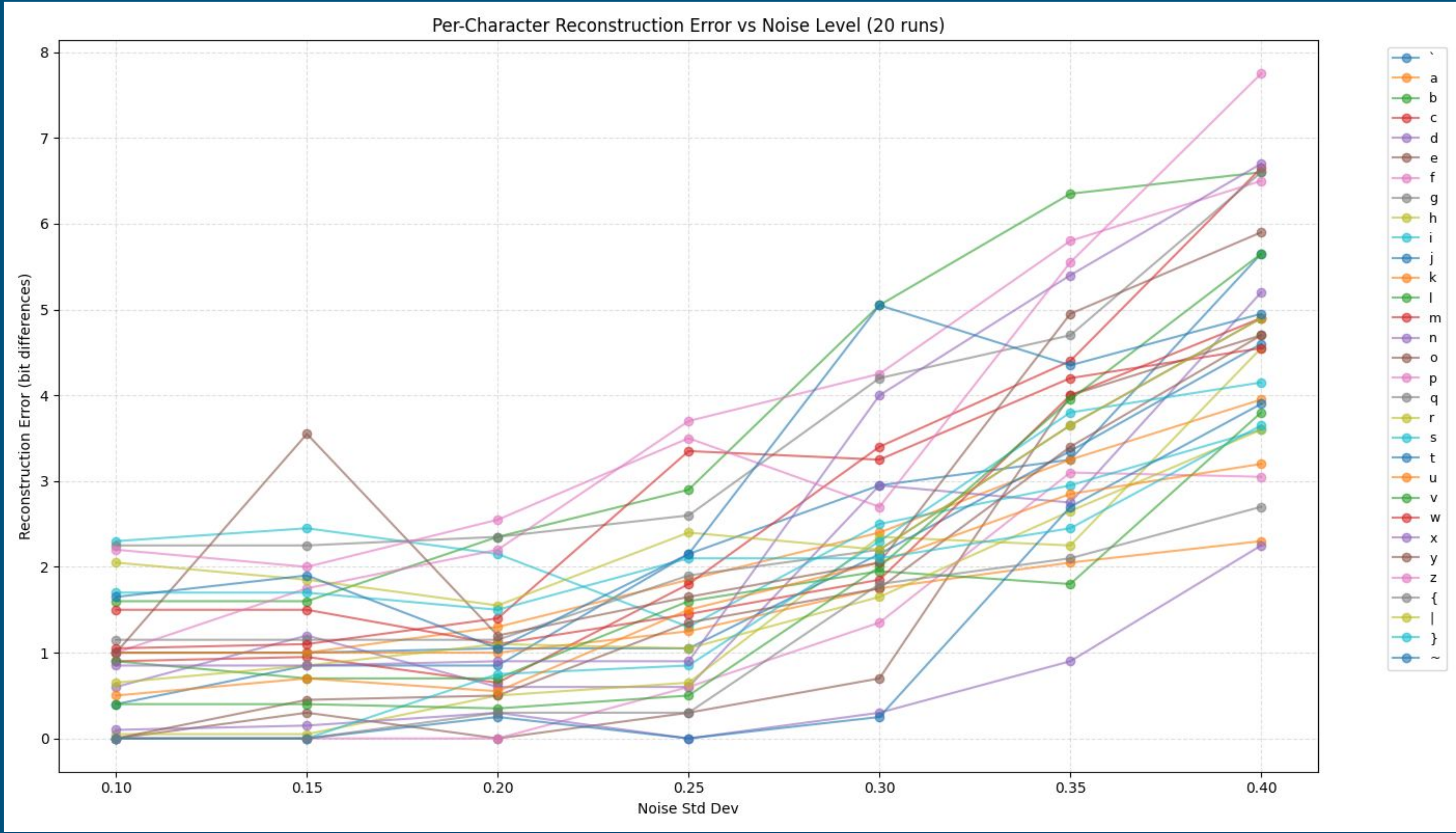


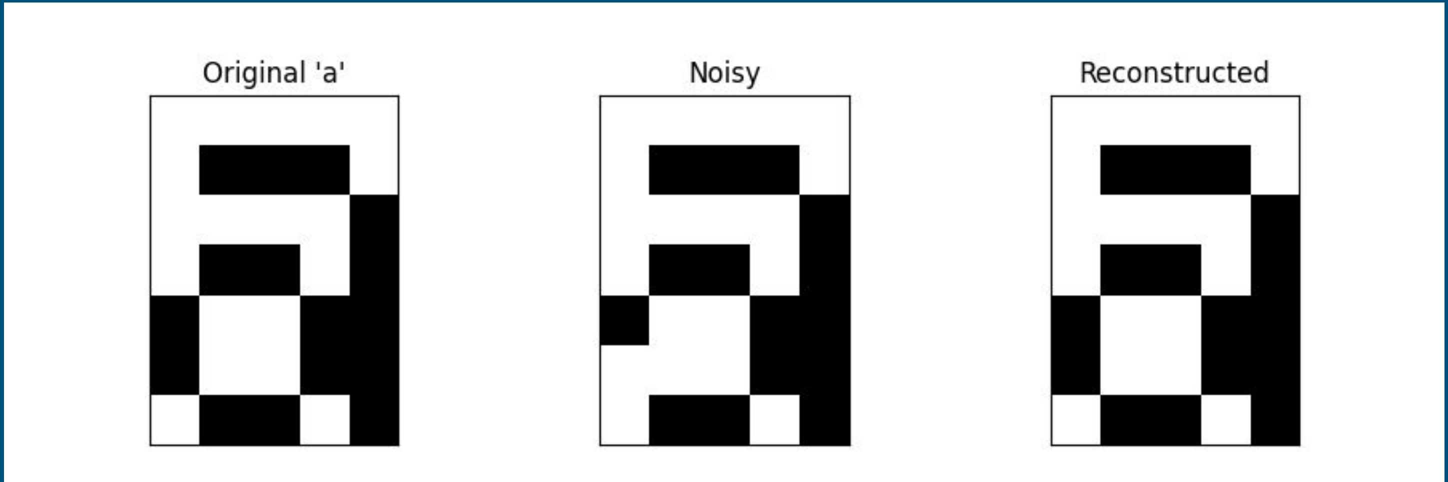
Denoising Autoencoder

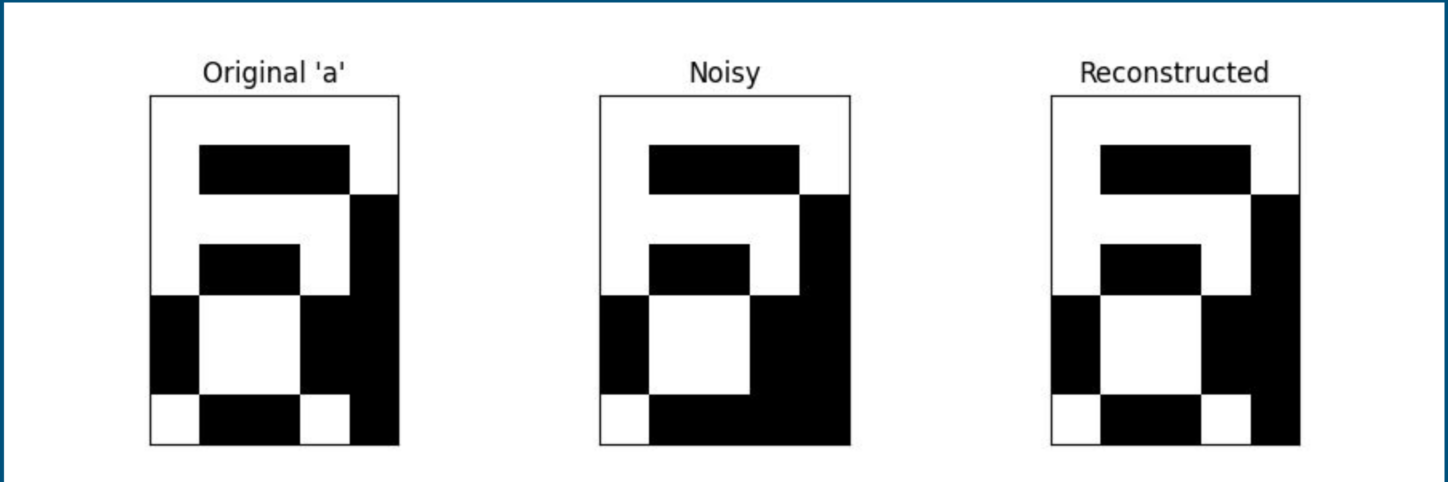
Configuración y entrenamiento

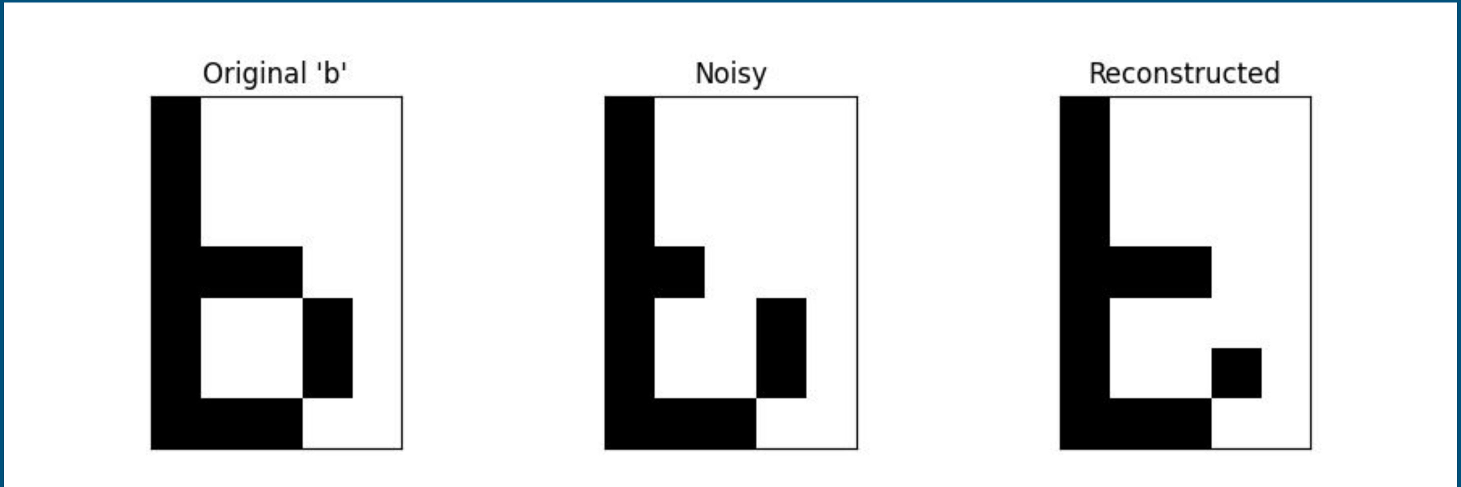
- Encoder y Decoder de 4 capas conectados por un espacio latente de 2 neuronas ([35, 32, 24, 16, 2, 16, 24, 32, 35])
- Función de activación sigmoide, learning rate η = 0.000985, 2000 Epochs
- Requirió varias pruebas encontrar cómo entrenar la red para conseguir resultados consistentes: usamos ruido binarizado y 5 versiones de cada carácter (stddev de 0.2)

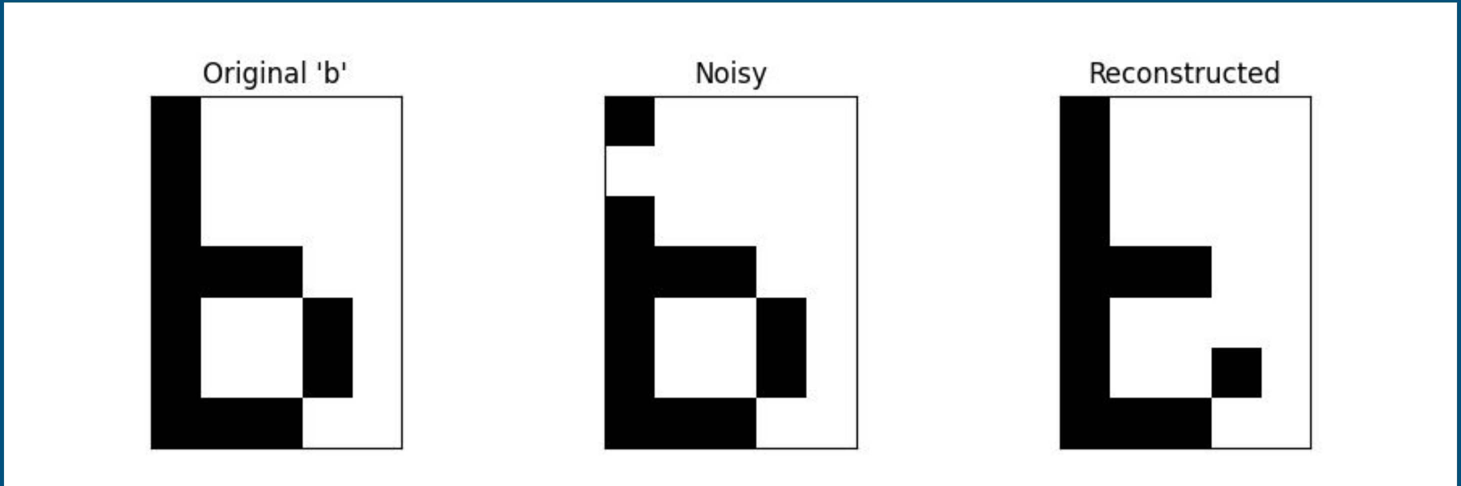

Average Reconstruction Error vs Noise Level (20 samples)
Average Reconstruction Error (bit differences)
2
3
4
5
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Noise Std Dev

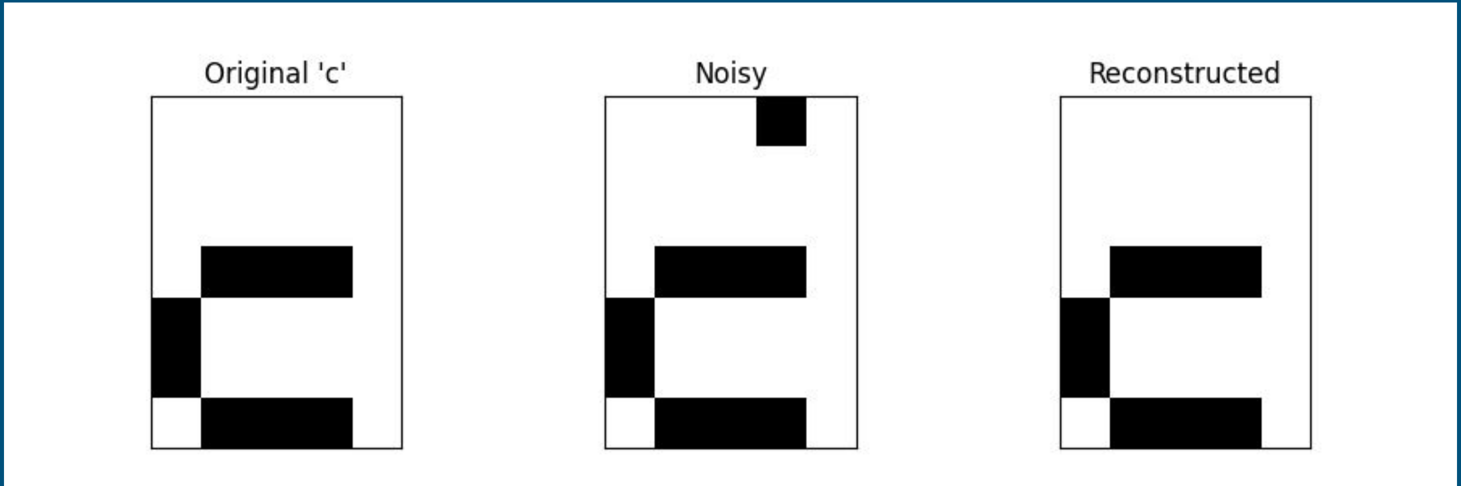

Per-Character Reconstruction Error vs Noise Level (20 runs)
Reconstruction Error (bit differences)
Noise Std Dev
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
1
2
3
4
5
6
7
8
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~

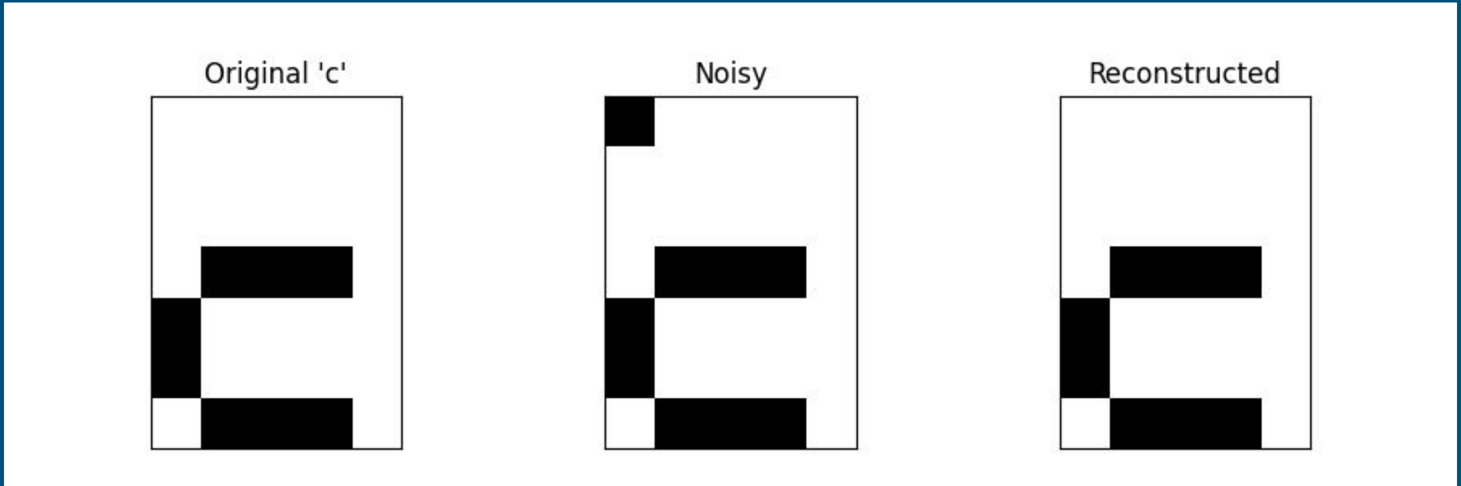

Original 'a'
Noisy
Reconstructed

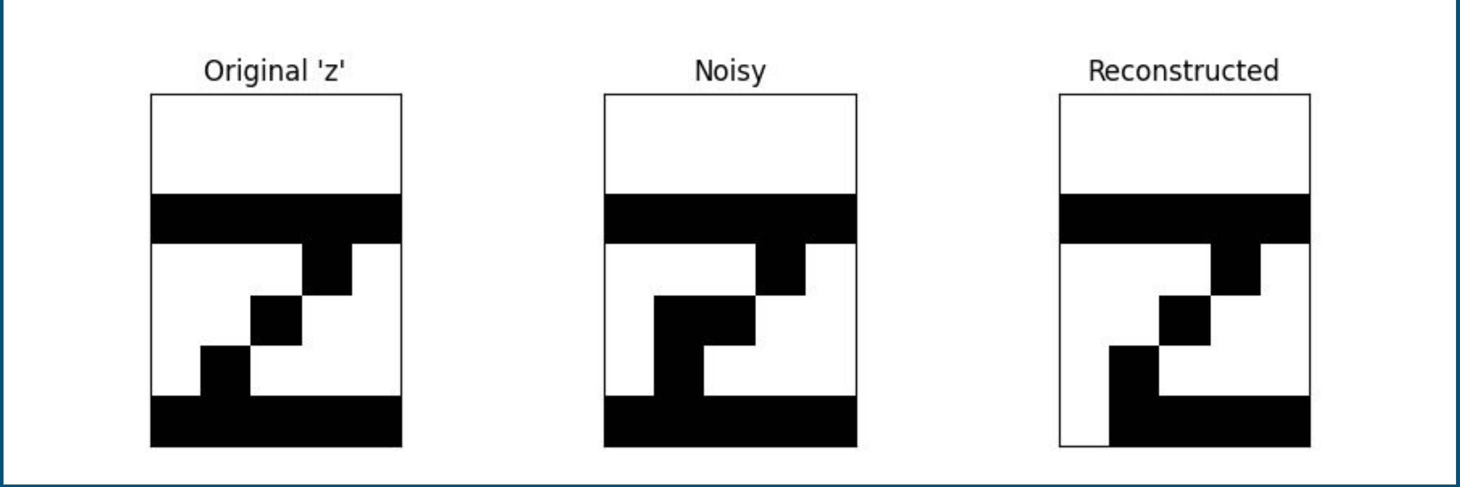

Original 'a'
Noisy
Reconstructed

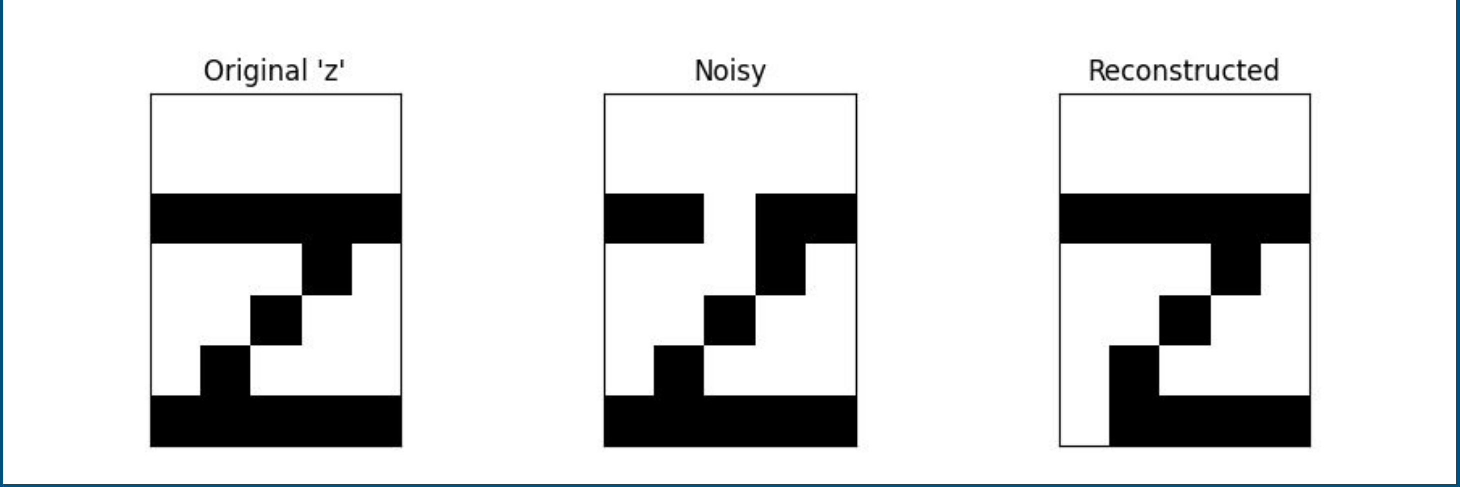

Original 'b'
Noisy
Reconstructed


Original 'b'
Noisy
Reconstructed


Original 'c'
Noisy
Reconstructed


Original 'c'
Noisy
Reconstructed

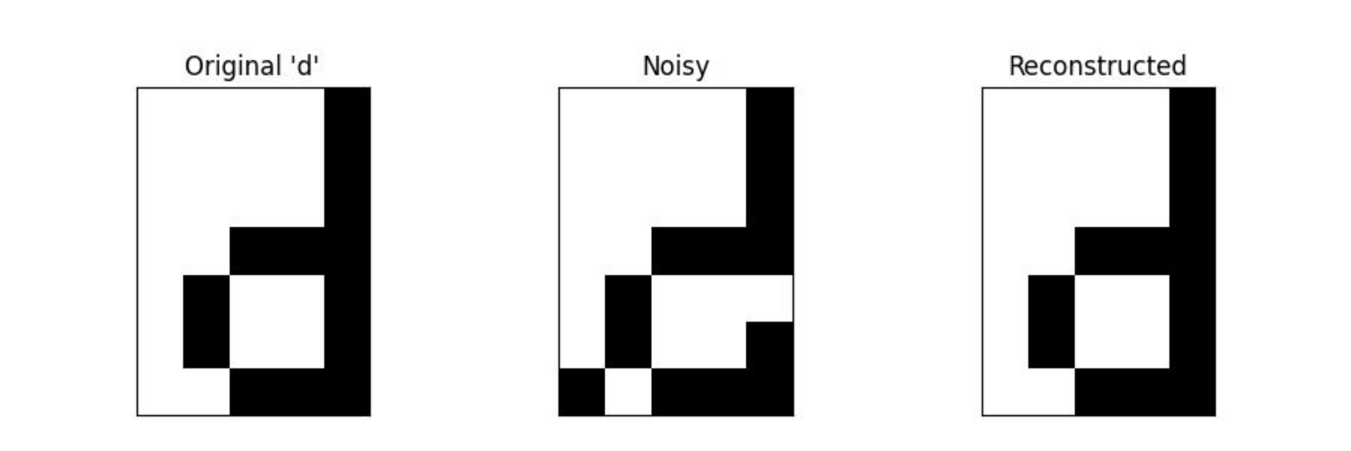

Original 'z'
Noisy
Reconstructed

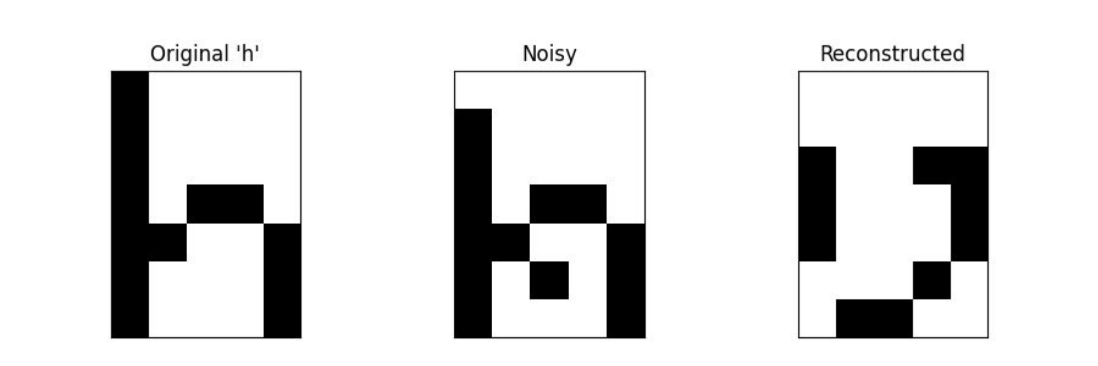

Original 'z'
Noisy
Reconstructed

Casos con más ruido

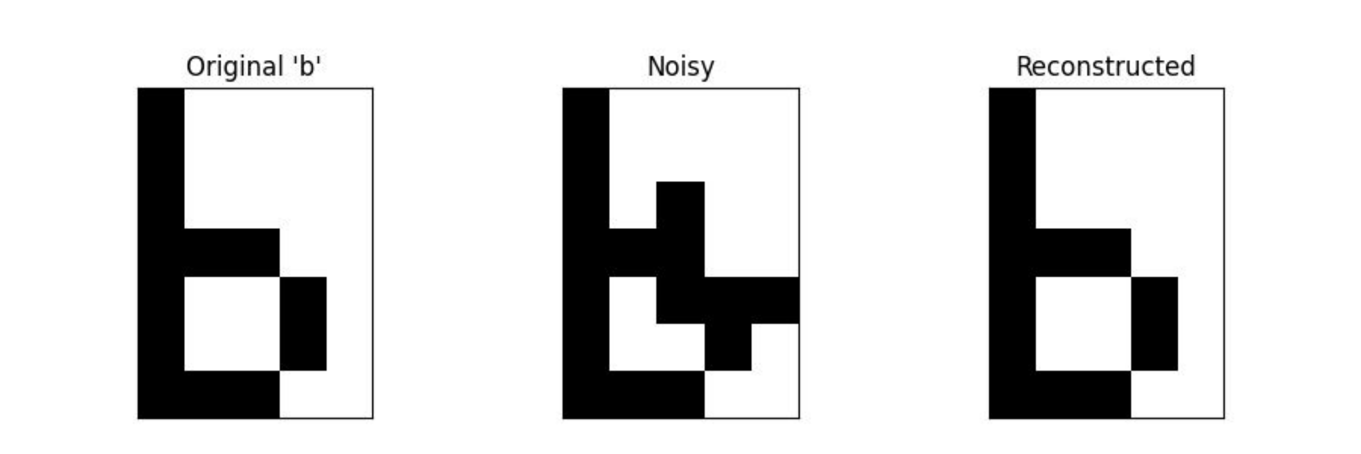
- Con más de un bit de diferencia es más propenso a fallar catastróficamente en la reconstrucción para ciertos caracteres
- Mientras que otros caracteres resultaron más “inmunes” al ruido
- Como se vió en el gráfico de Errores vs Noise Level por caracteres obtuvimos una distribución amplia

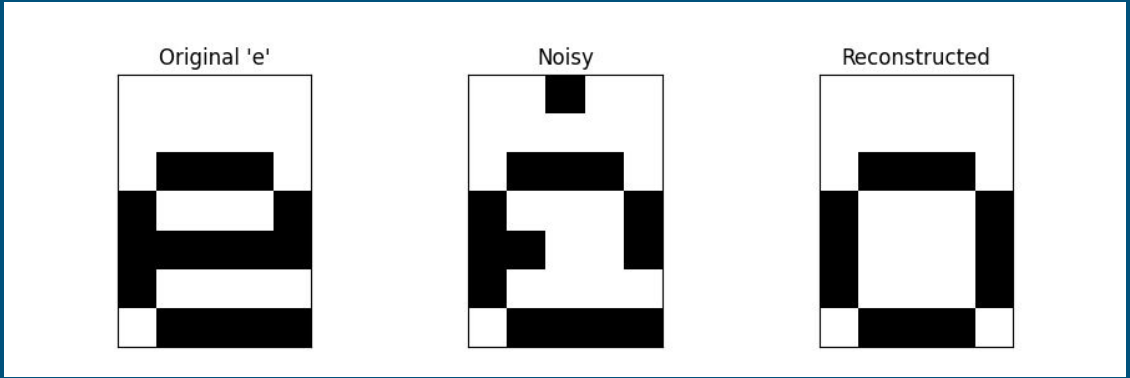
2 Flips


Original 'd'
Noisy
Reconstructed

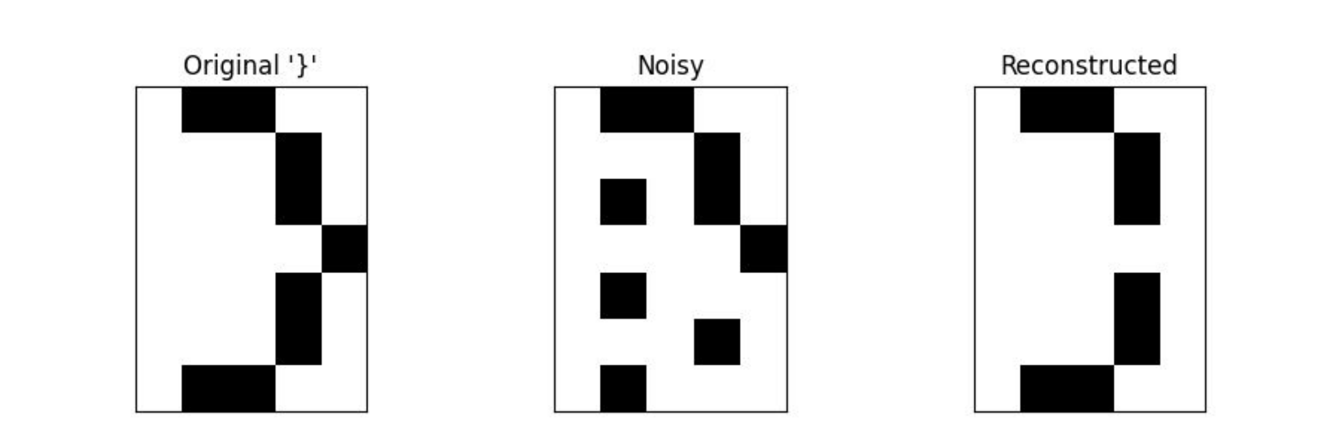

Original 'h'
Noisy
Reconstructed

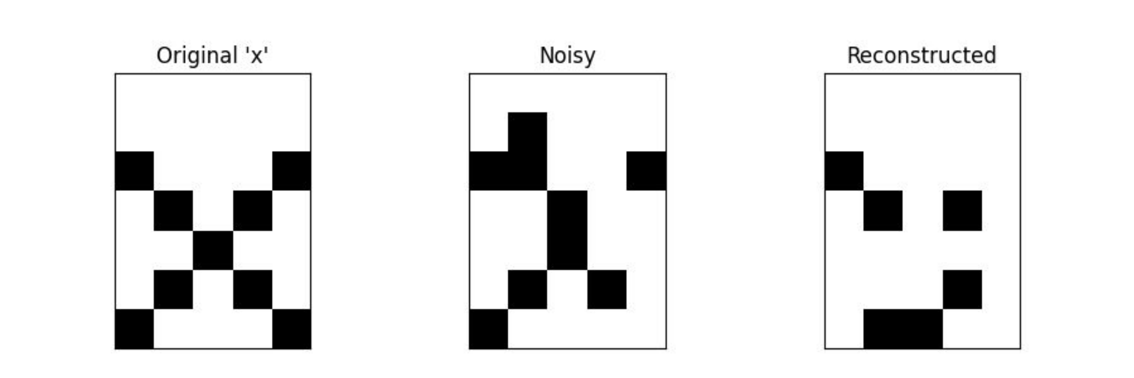
3 Flips


Original 'b'
Noisy
Reconstructed

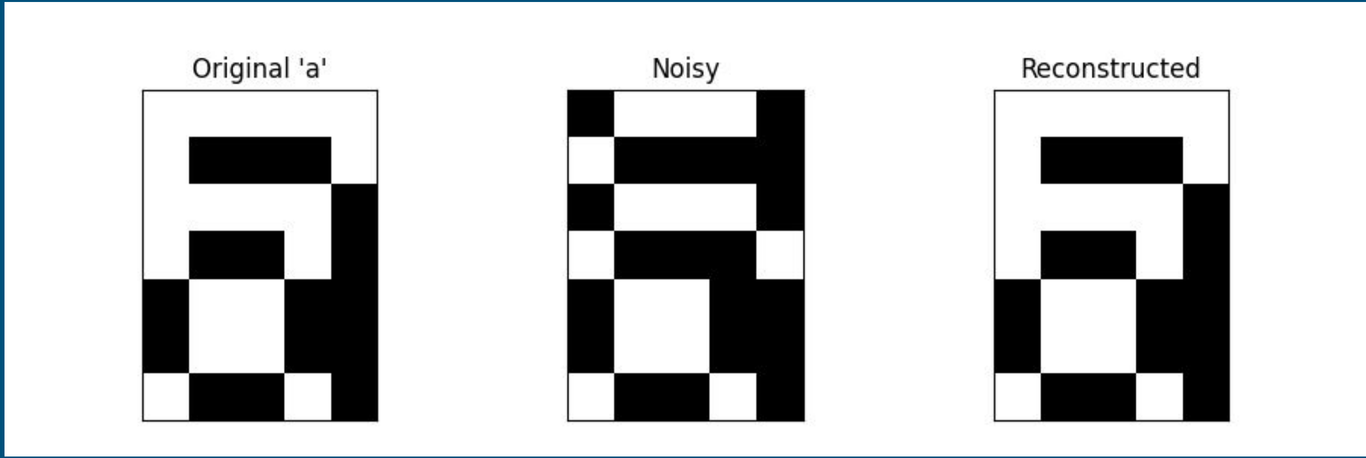

Original 'e'
Noisy
Reconstructed

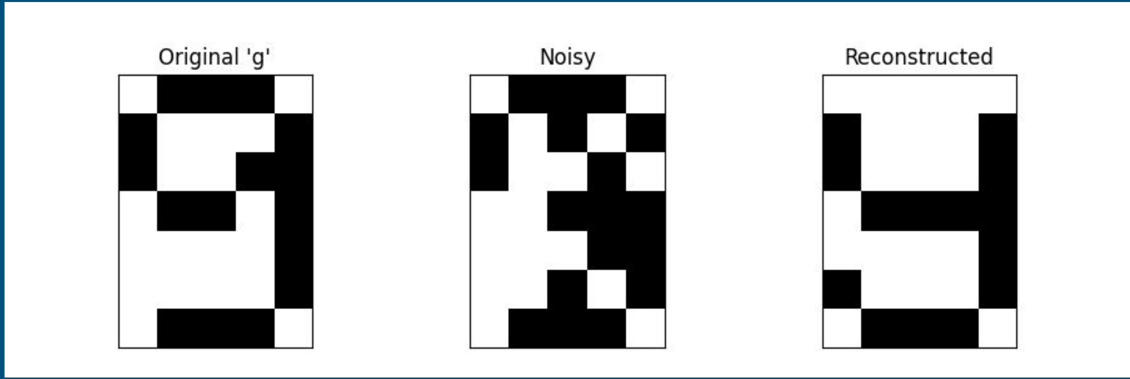
4 Flips


Original '}'
Noisy
Reconstructed

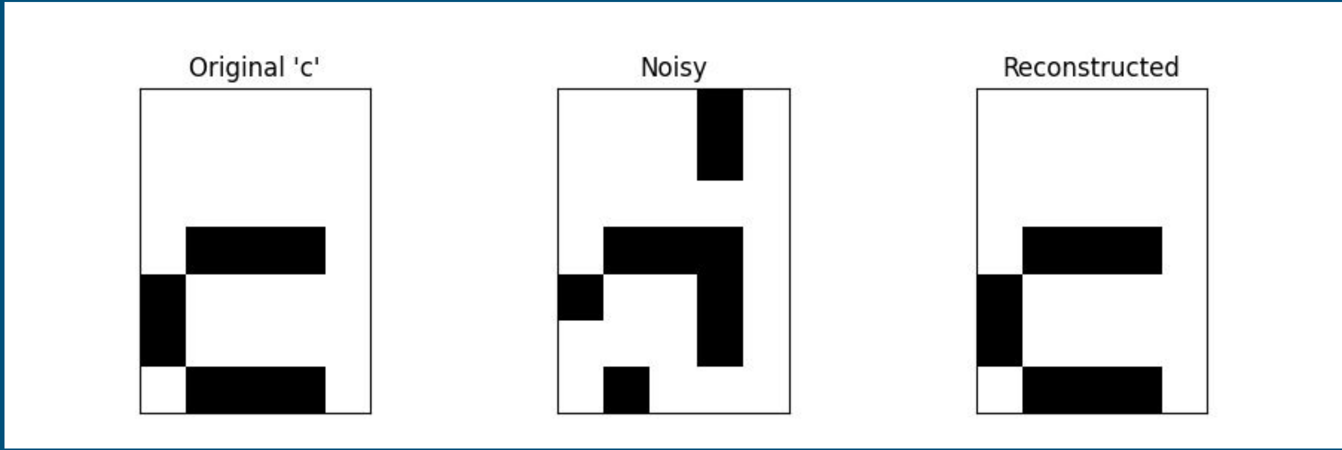

Original 'x'
Noisy
Reconstructed

5 Flips

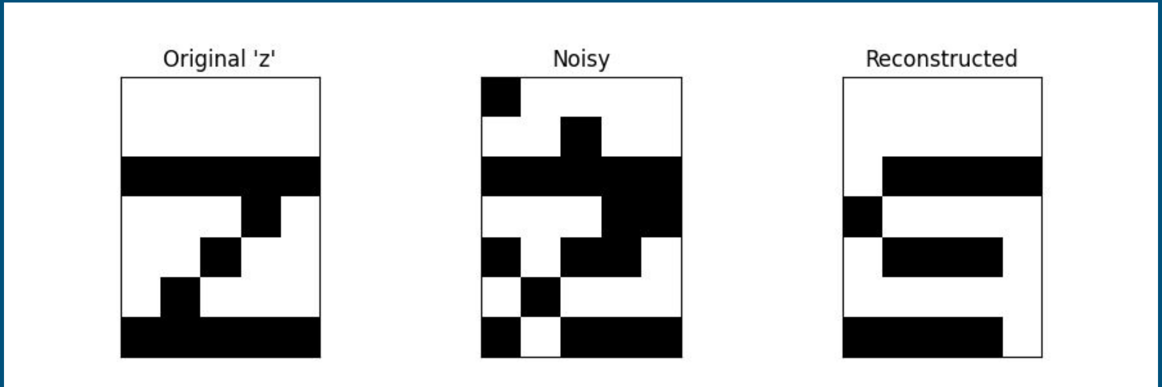

Original 'a'
Noisy
Reconstructed


Original 'g'
Noisy
Reconstructed

6 Flips



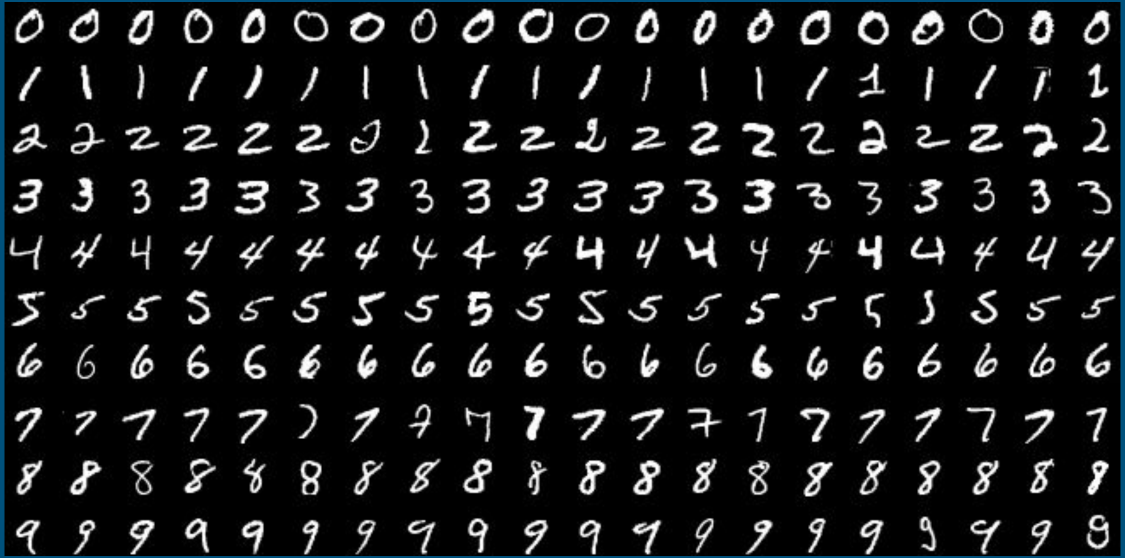
Original 'c'
Noisy
Reconstructed



Original 'z'
Noisy
Reconstructed

Autoencoder Variacional

Conjunto de datos para entrenar

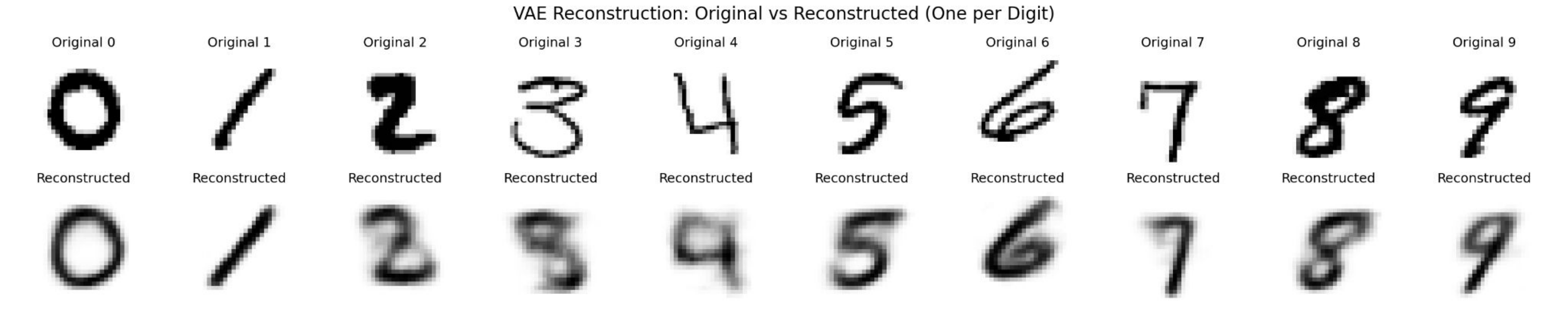


20K elementos del conjunto de datos de MNIST (28x28)

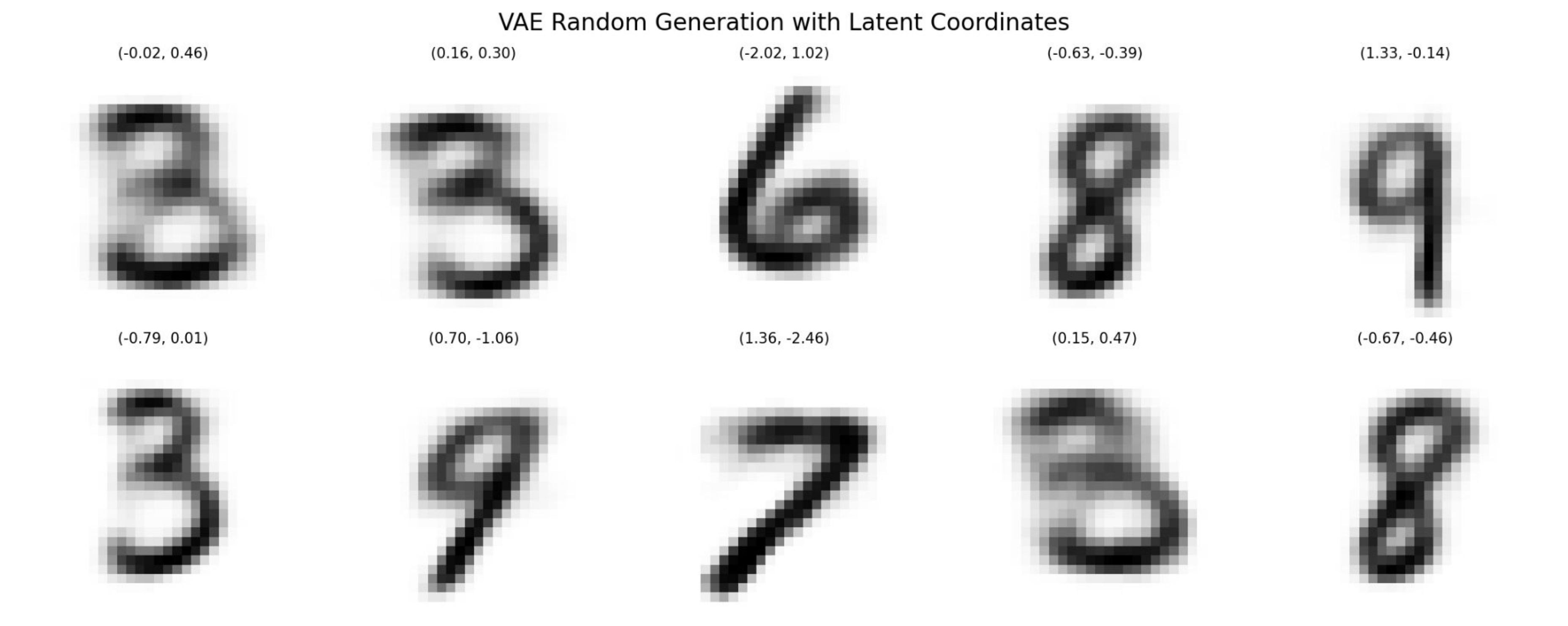
Arquitectura de red e hiperparametros

- Arquitectura:
  - Encoder: 784 -> 128 -> 64
  - Espacio Latente: 2
  - Decoder: 64 -> 128 -> 784
- Optimizador: ADAM
- Learning Rate: 0.001
- Funciones de activación:
  - ReLU (Capas ocultas)
  - Sigmoidea (Última capa)
- Epochs: 200
- Batch Size: 64 (Mini-batch)

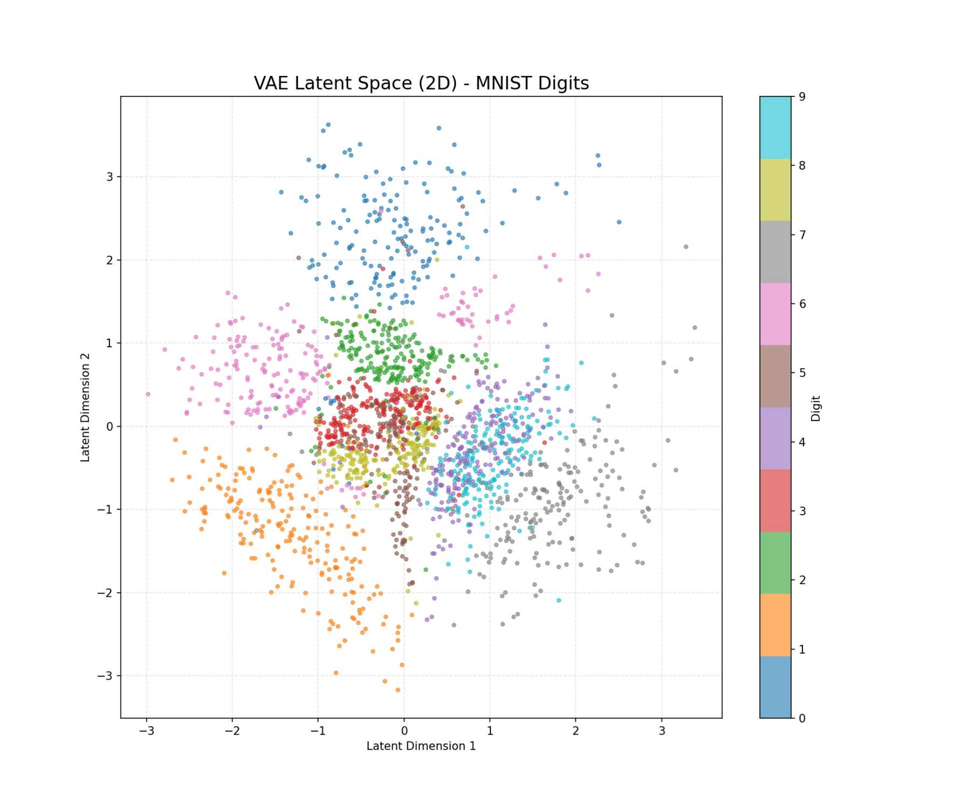
Reconstrucción (E = 0.175)

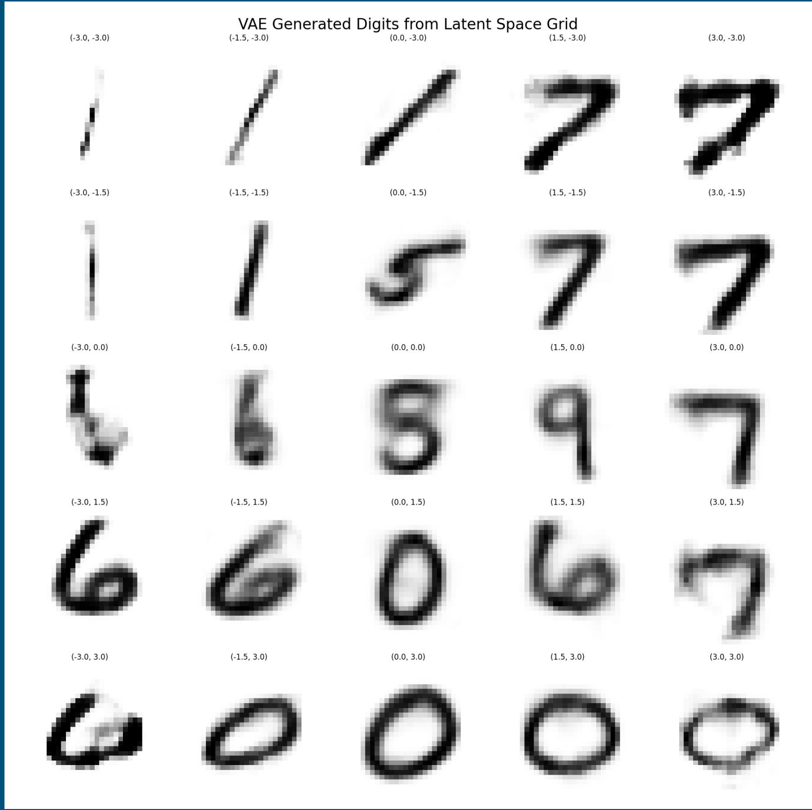


Generación (E = 0.175)



Espacio Latente (E = 0.175)



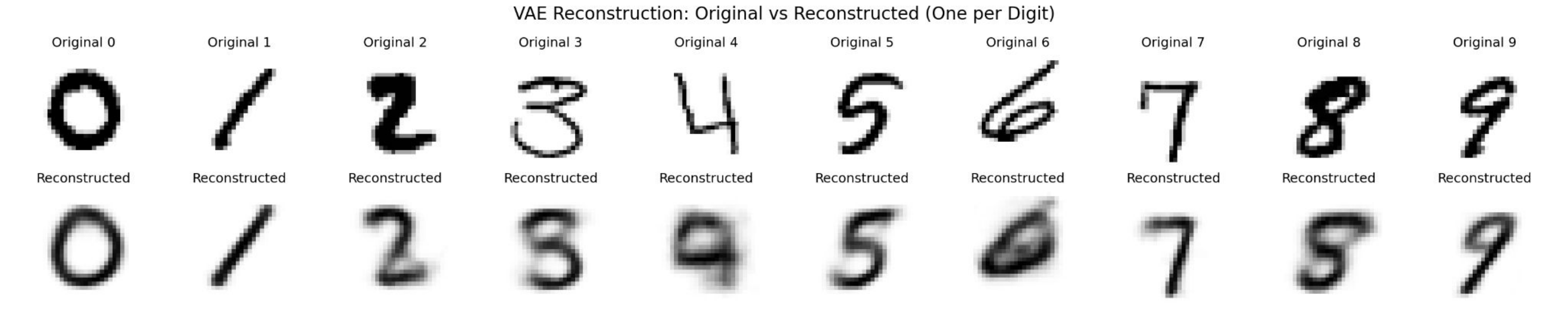


Si modificamos la arquitectura….

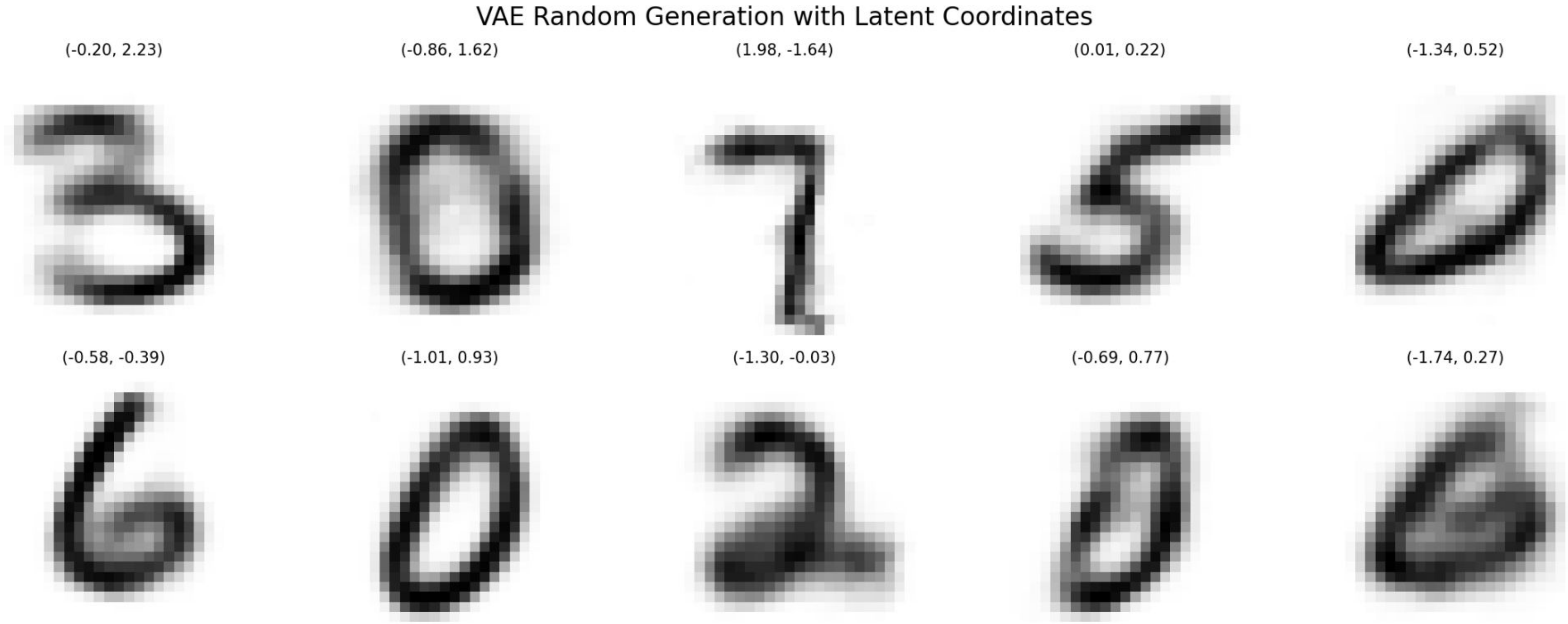
- Arquitectura:
  - Encoder: 784 -> 256 -> 128
  - Espacio Latente: 2
  - Decoder: 128 -> 256 -> 784
- Optimizador: ADAM
- Learning Rate: 0.001

- Funciones de activación:
  - ReLU (Capas ocultas)
  - Sigmoidea (Última capa)
- Epochs: 200
- Batch Size: 64 (Mini-batch)

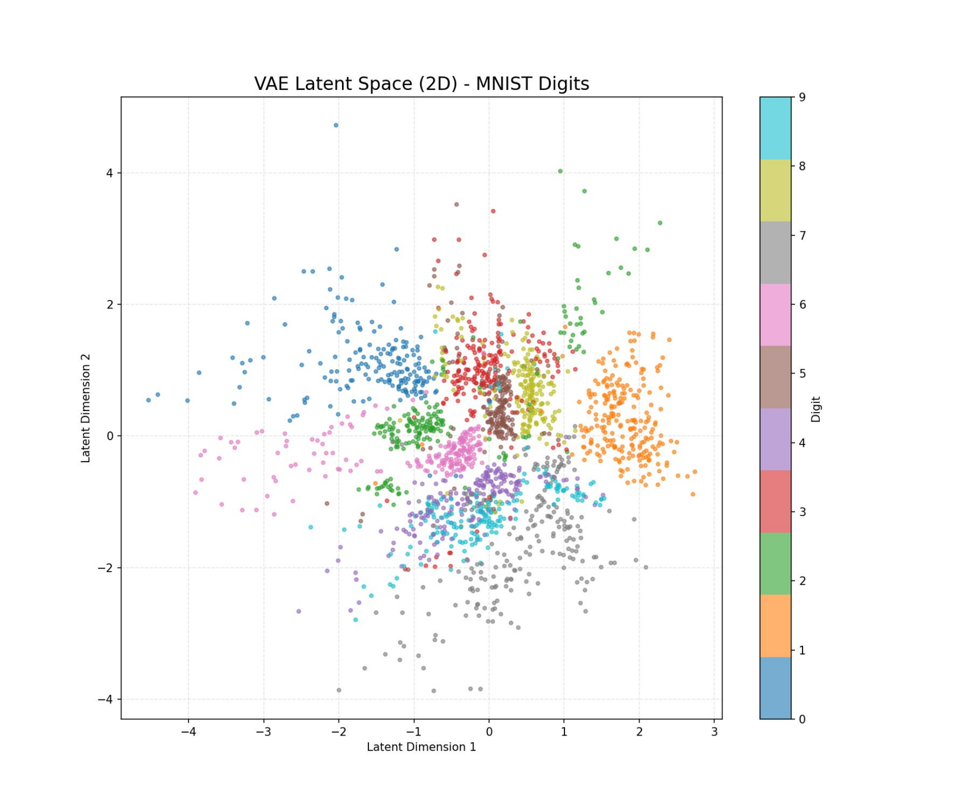
Reconstrucción (E = 0.170)

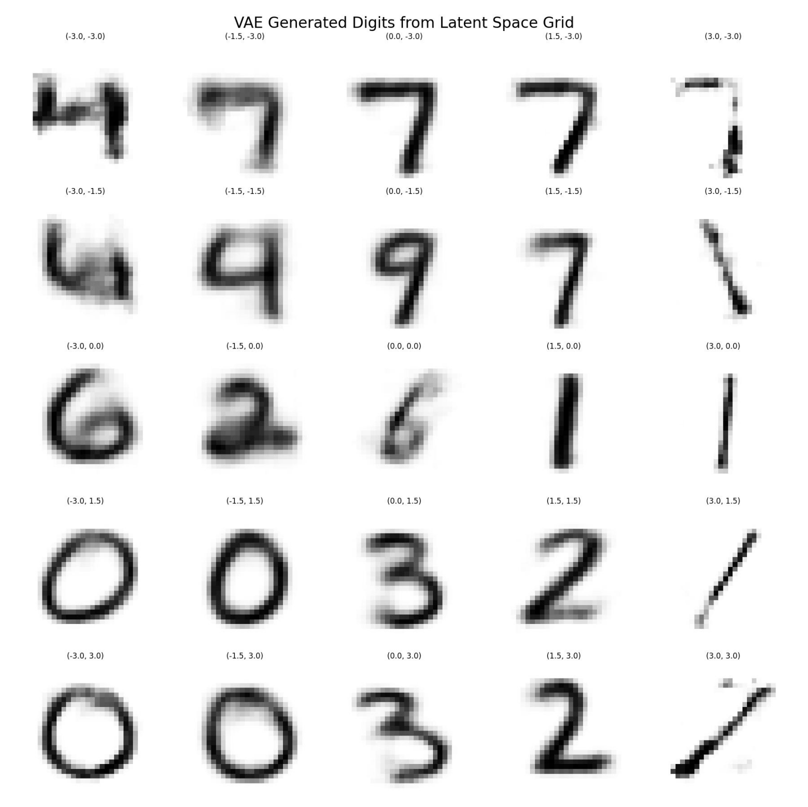


Generación (E = 0.170)



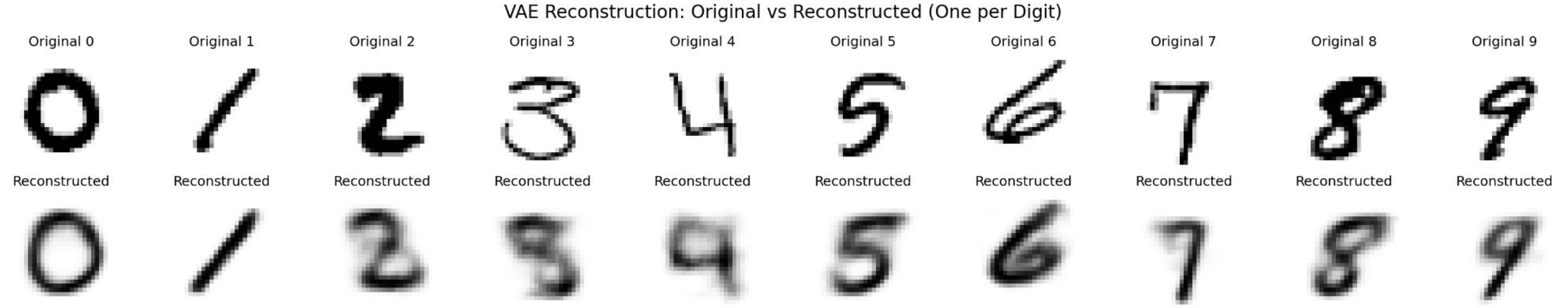
Espacio Latente (E = 0.170)



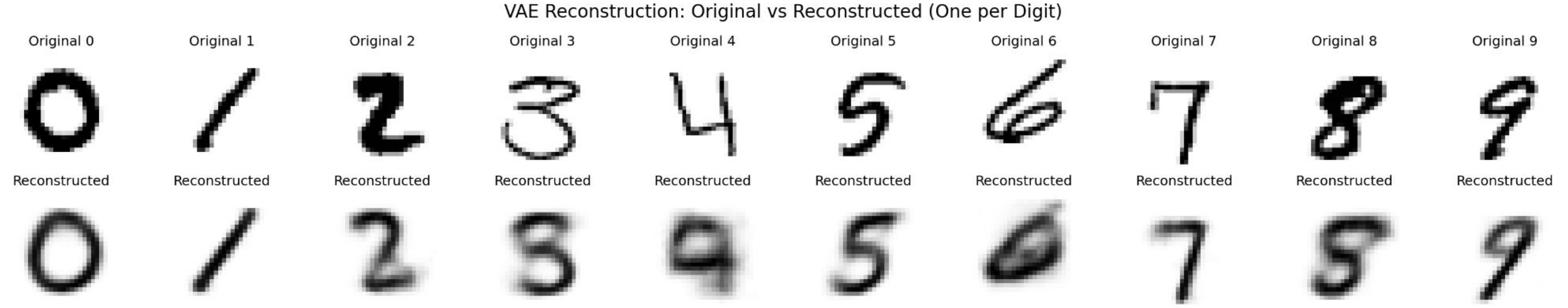


Comparación

Reconstrucción 1 (E = 0.175)



Reconstrucción 2 (E = 0.170)



Conclusiones

- El DAE necesita un mínimo de copias con ruido en el dataset para aprender a corregir errores y memorizar patrones base. Aún así es muy difícil arreglar errores producto de patrones similares.
- En el VAE, la arquitectura 128→64 demostró ser suficiente para modelar dígitos MNIST, con una mejora marginal al duplicar parámetros (256→128: 0.170 vs 0.175)
- El VAE logró separar automáticamente las 10 clases de dígitos en clusters diferenciados en un espacio 2D, sin supervisión de etiquetas durante el entrenamiento