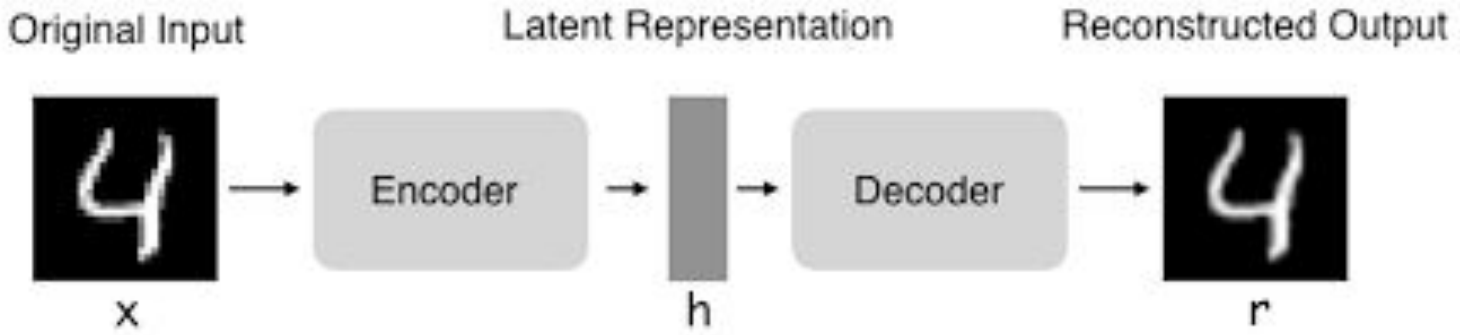
Deep Learning

Franco Baliarda, Lucía Torrusio,
Joaquín Colonnello

Introducción

- Desarrollar un autoencoder → analizar diferentes arquitecturas y parámetros
- Estudiar autoencoders variacionales
- Analizar los resultados y obtener conclusiones

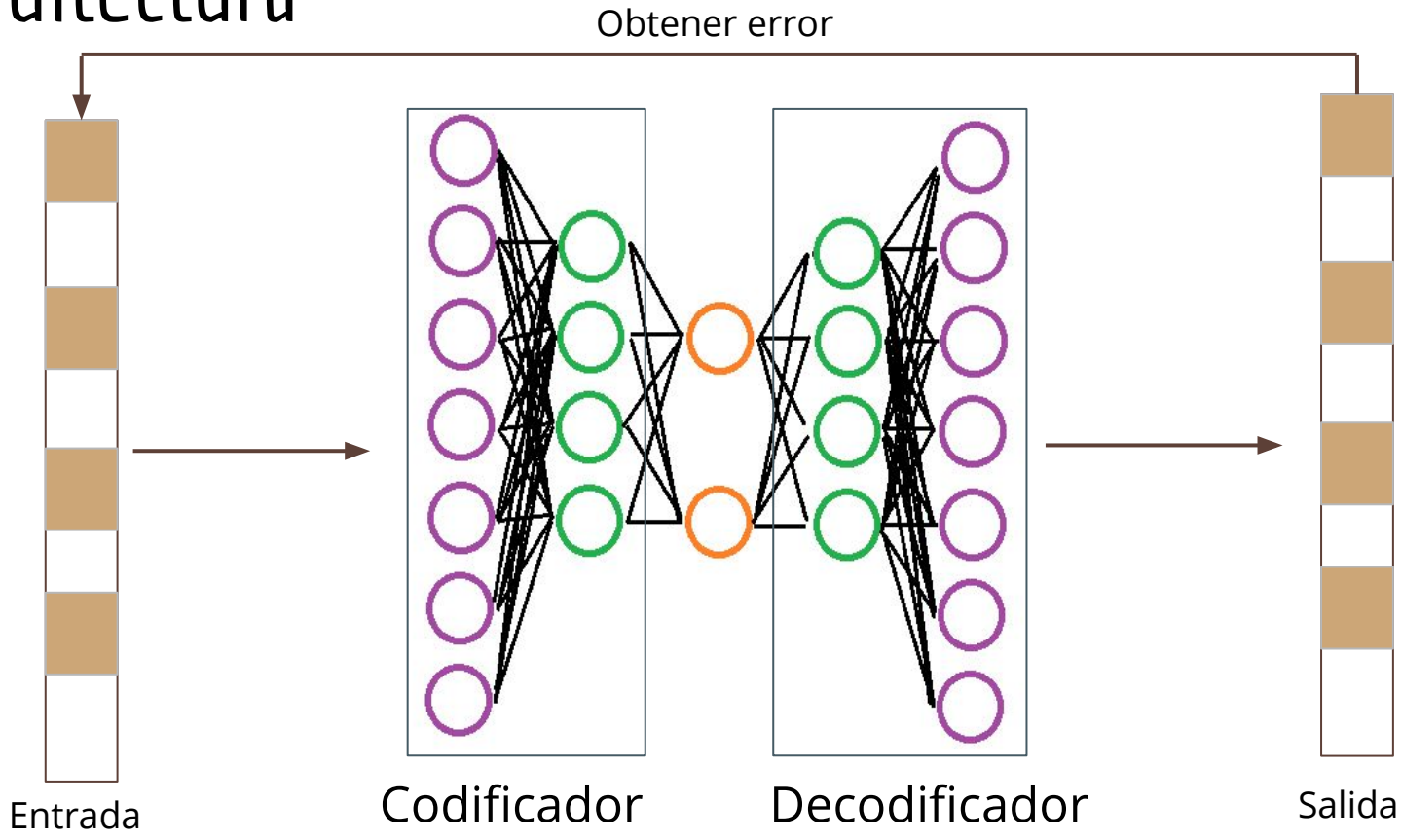


Ejercicio 1

Autoencoder

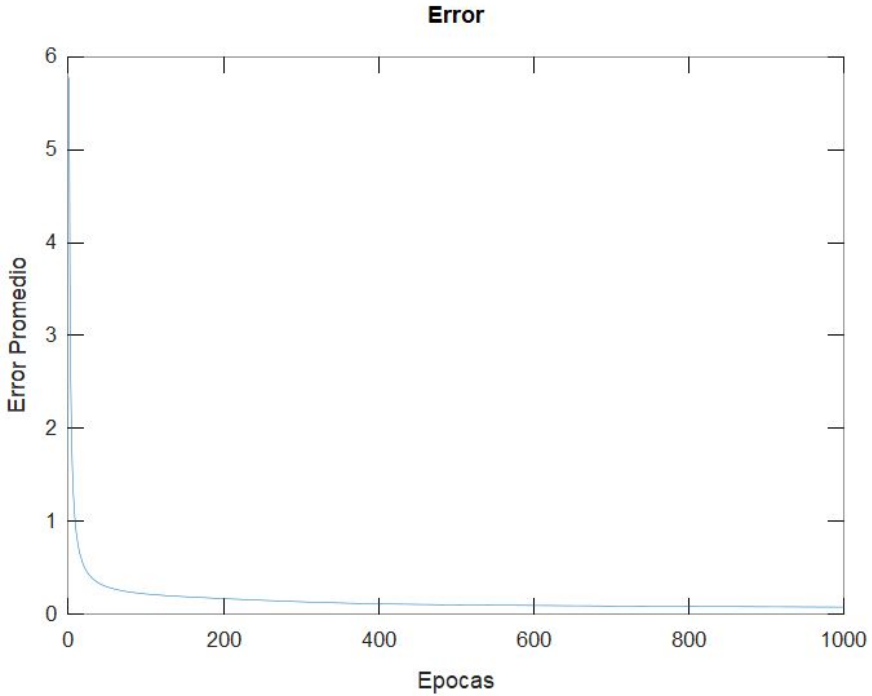
Caracteres elegidos:

Arquitectura

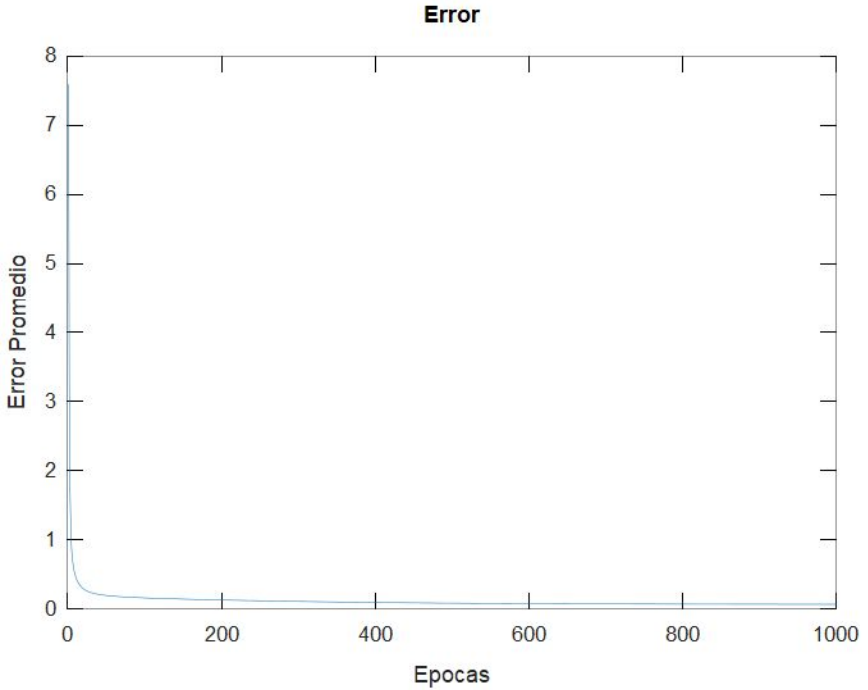


Autoencoder - Arquitectura

Capa Intermedia 4D → Error promedio mín: **0.072**

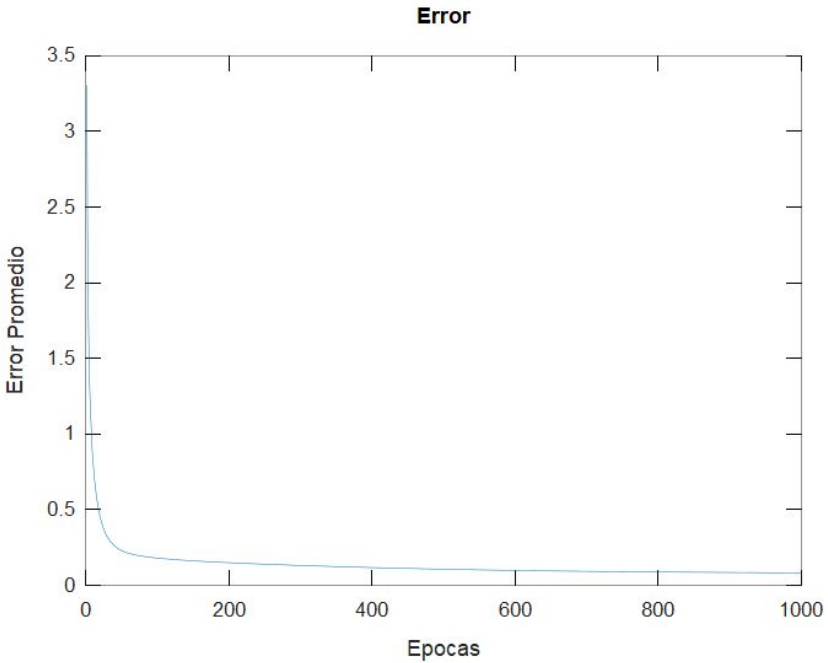


Capa Intermedia 6D → Error promedio mín: **0.064**

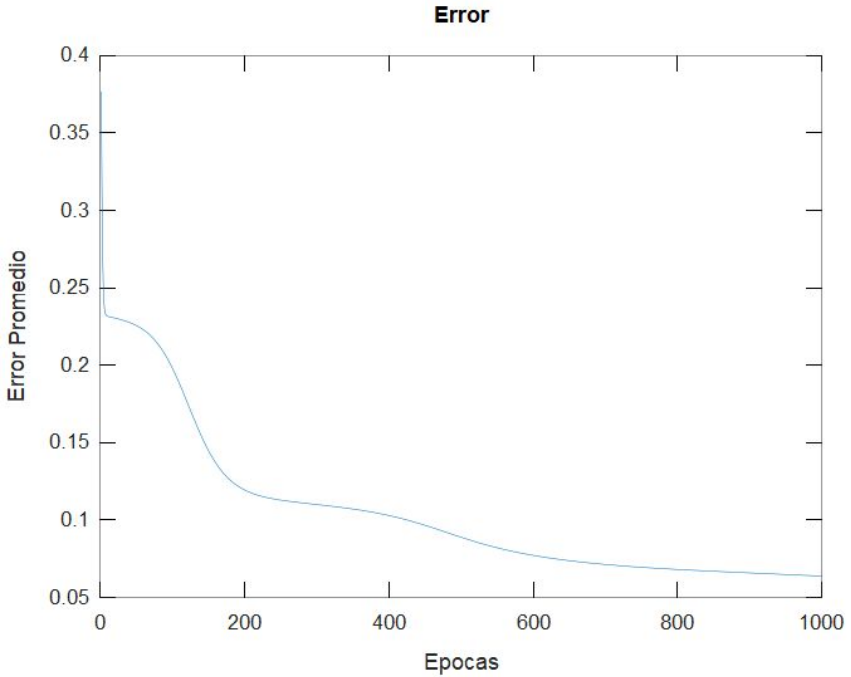


Autoencoder - Activación

Lineal → Error promedio mín: **0.082**



No Lineal → Error promedio mín: **0.063**



Optimizaciones

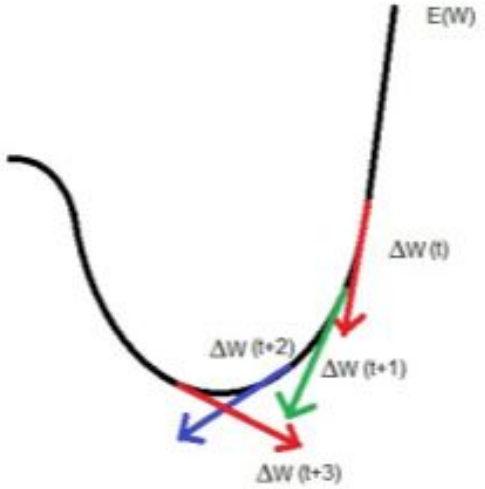
Tasa de aprendizaje adaptativo: BrentSearch

$$\alpha_k = \arg\min_{\alpha>0} g(\alpha) = f(x_k + \alpha d_k)$$

$$\eta_k = arg\, min_{\eta>0}\ g(\eta) = f(x_k + \eta d_k)$$

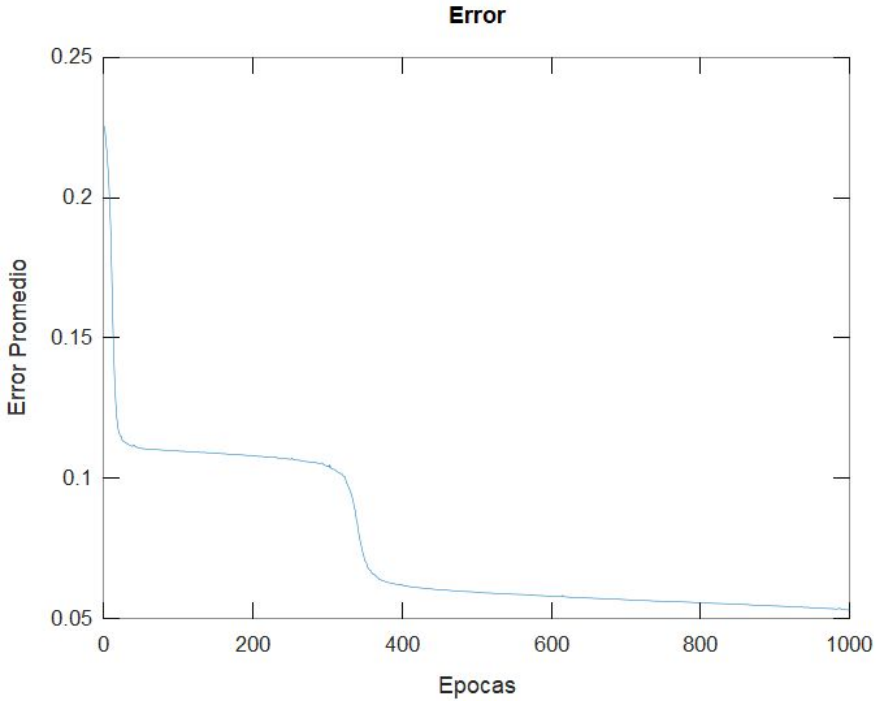
Momentum: $\alpha = 0.8$

$$\Delta w_{ij}(t+1) = -\eta \frac{\delta E}{\delta w_{ij}} + \alpha \Delta w_{ij}(t)$$

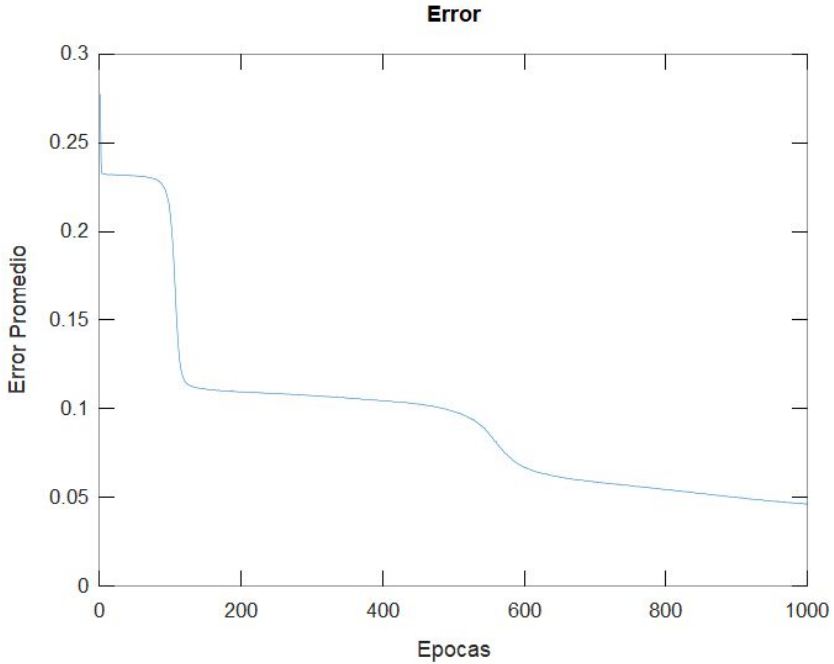


Autoencoder - Optimizaciones

Lr adaptativo → Error promedio mín: **0.053**

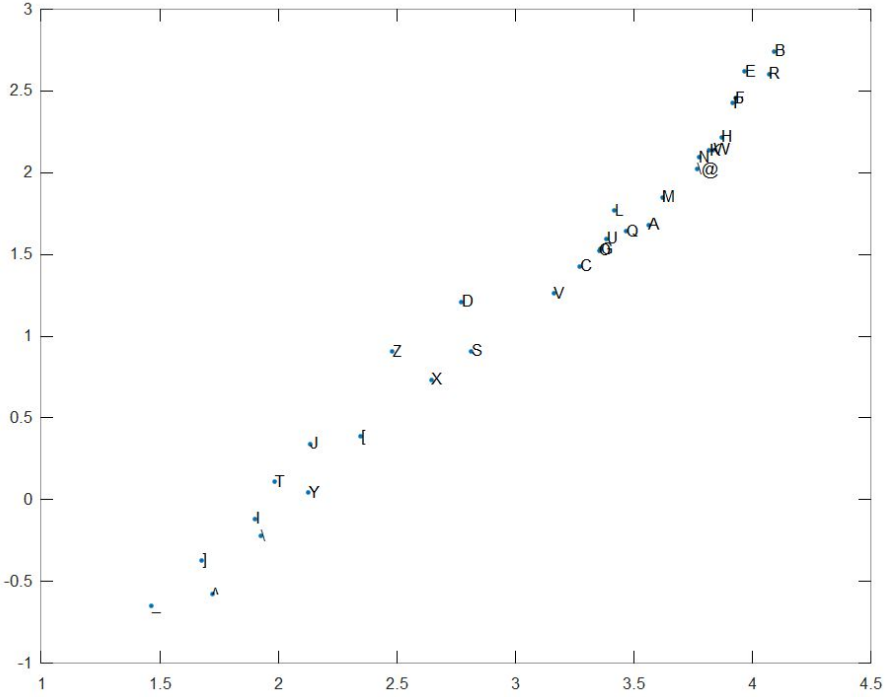


Momentum → Error promedio mín: **0.046**

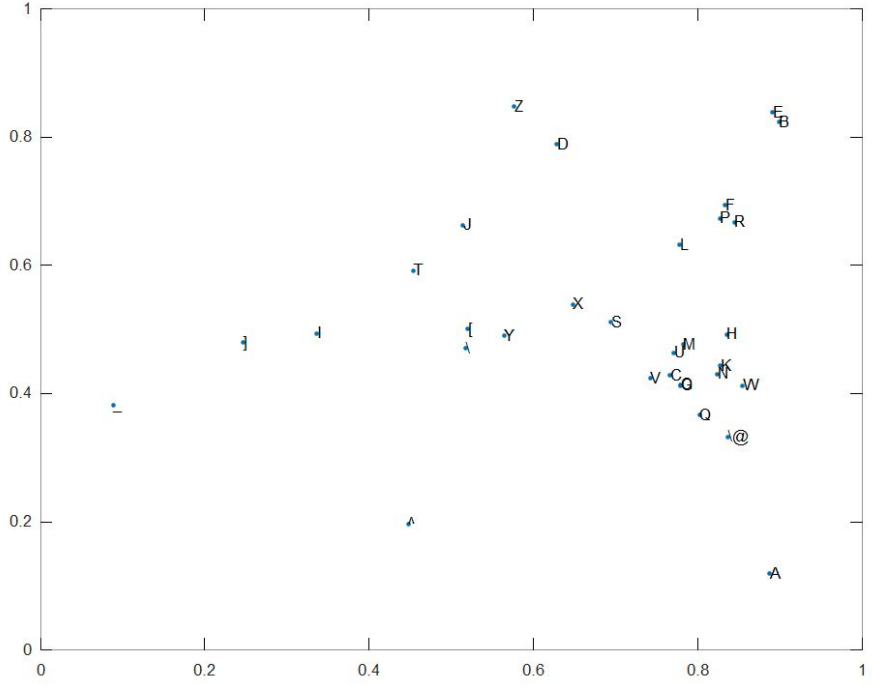


Autoencoder - Espacio Latente

Lineal

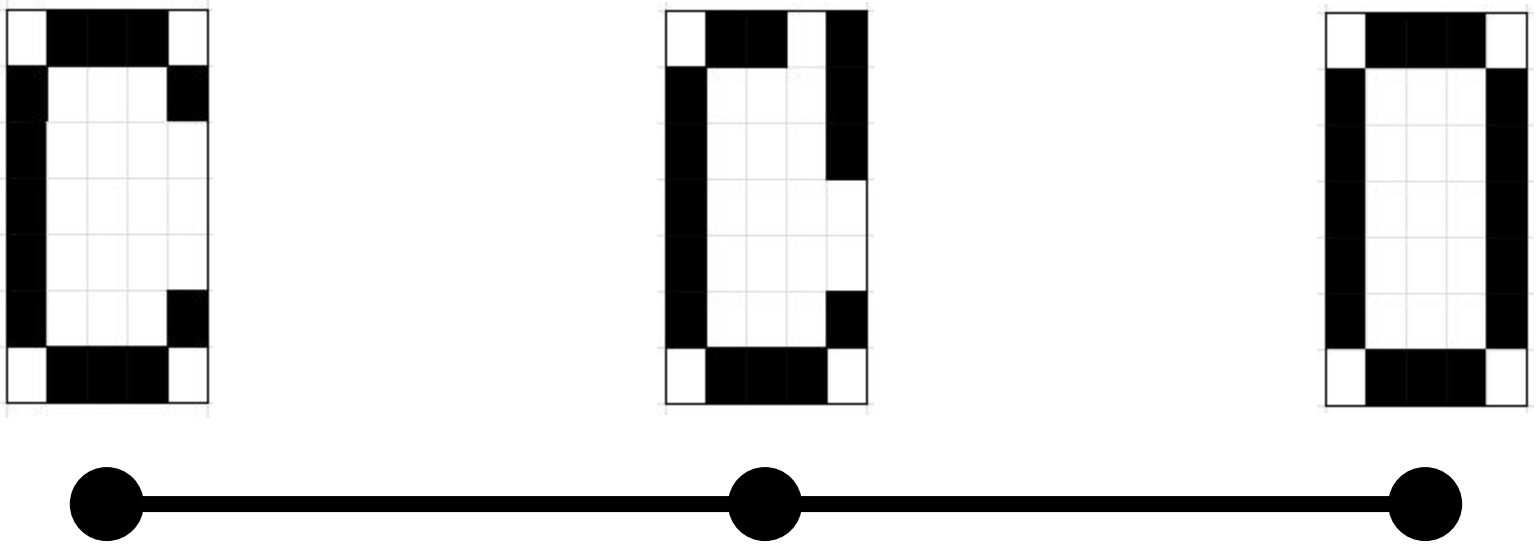


No Lineal



Autencoder

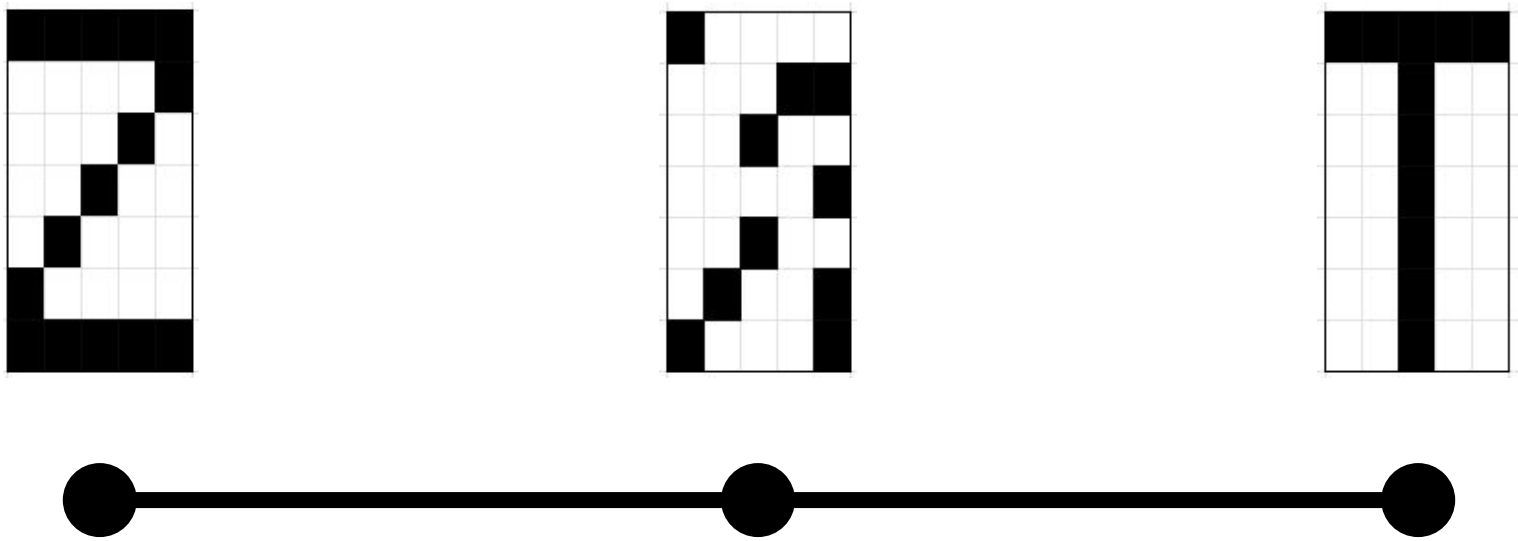
Generación de nuevas letras



Punto intermedio
entre ***c*** y ***o***

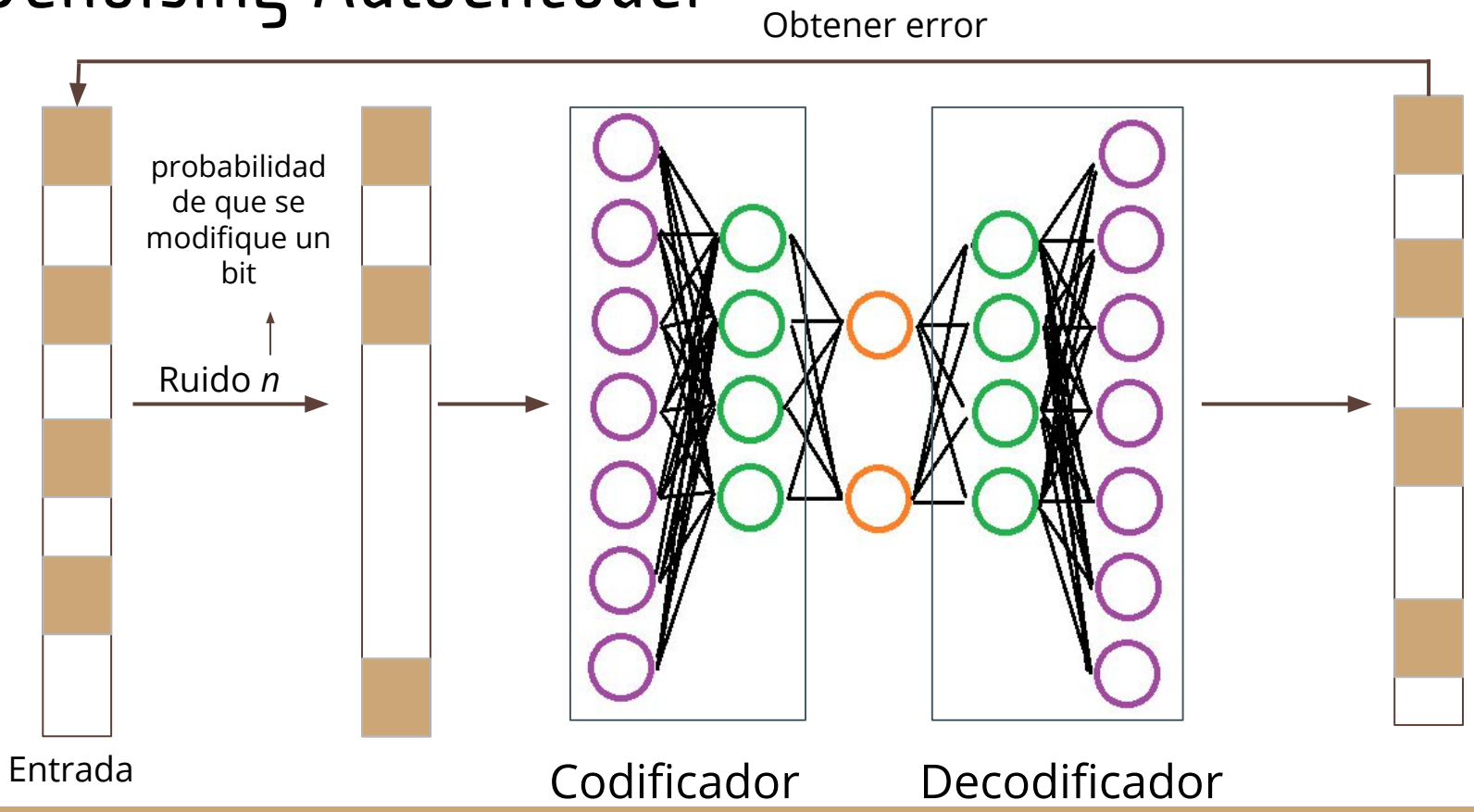
Autencoder

Generación de nuevas letras



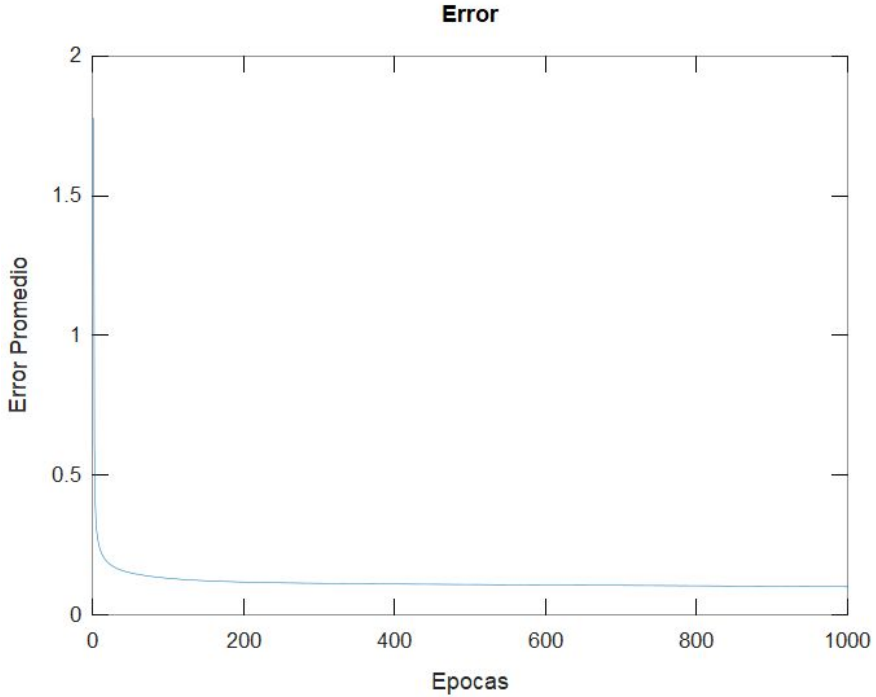
Punto intermedio
entre ***z*** y ***t***

Denoising Autoencoder

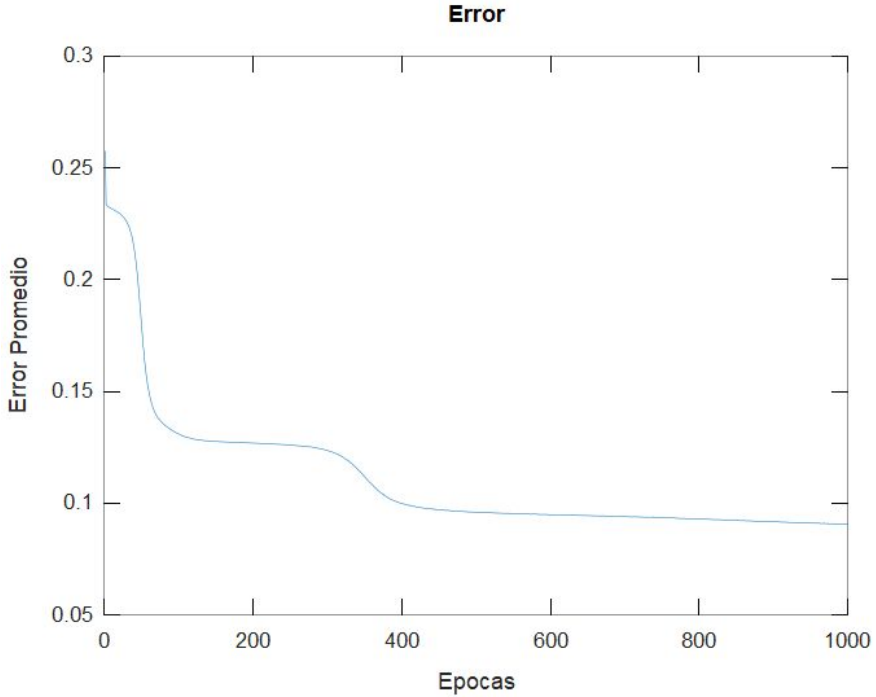


Denoising Autoencoder

Lineal → Error promedio mín: **0.101**

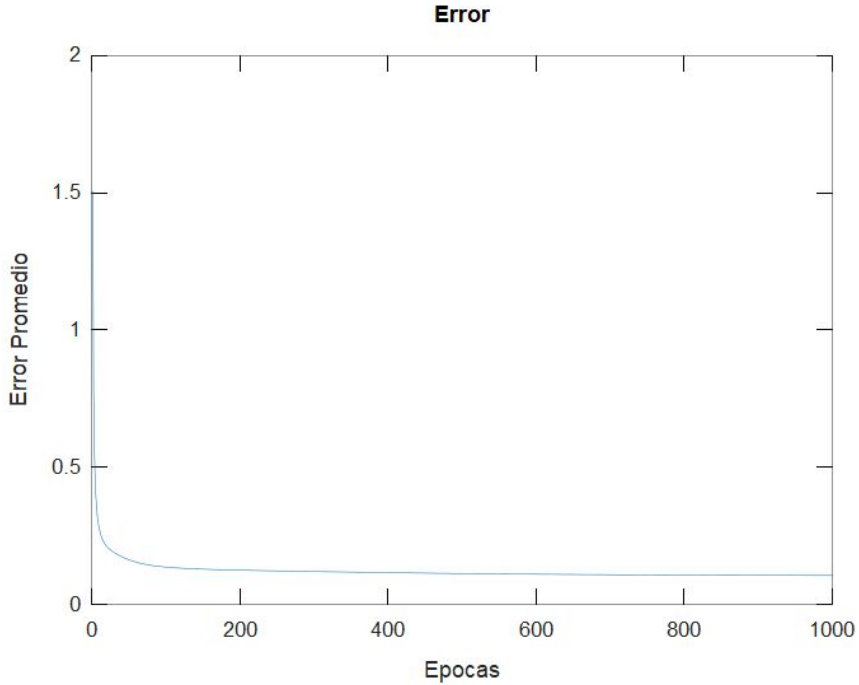


No Lineal → Error promedio mín: **0.0905**

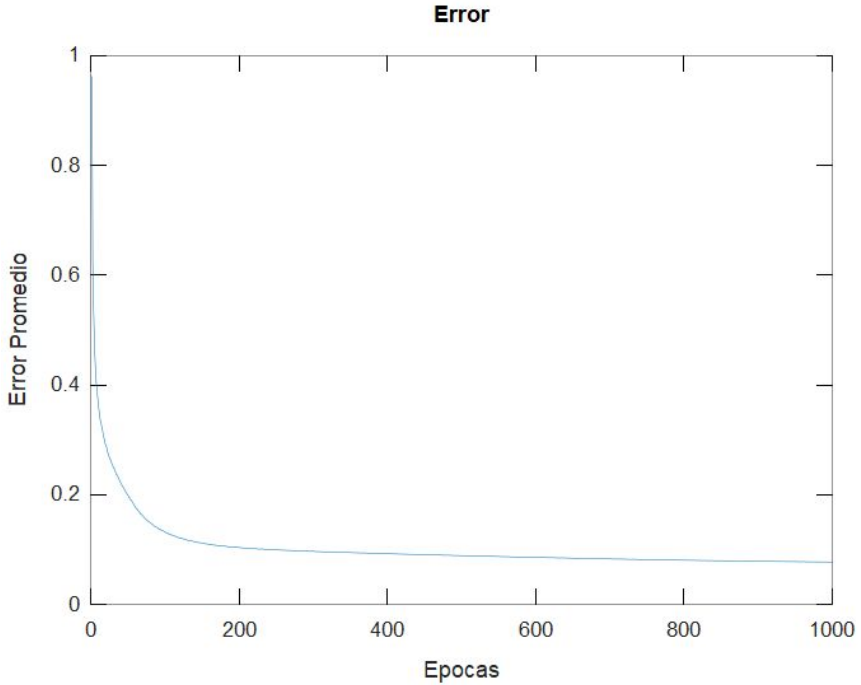


Denoising Autoencoder

Esp. Latente 2D → Error promedio mín: **0.105**

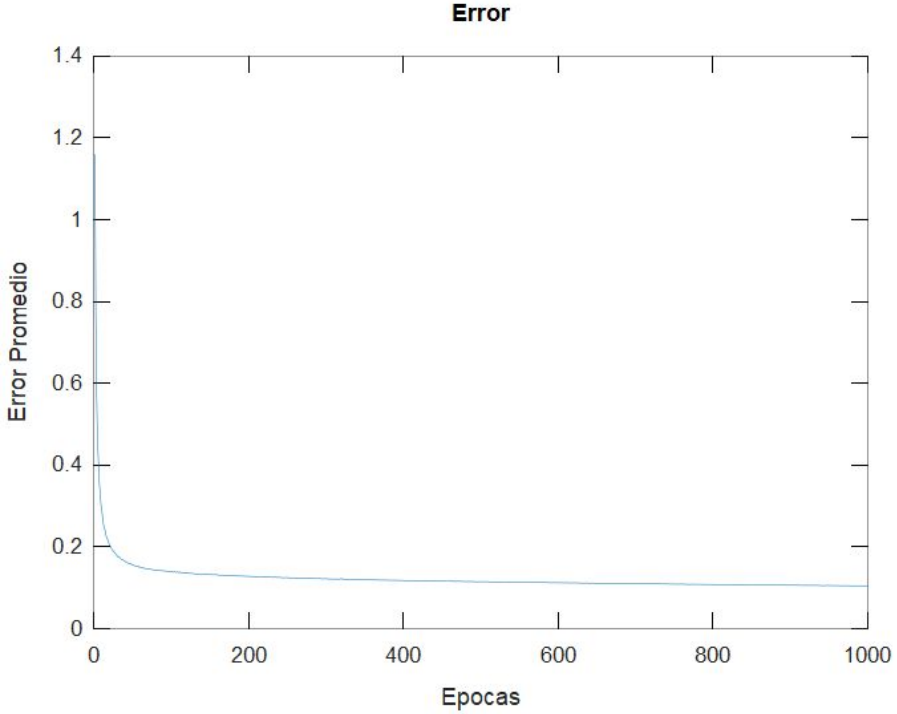


Espacio Latente 4D → Error promedio mín: **0.077**

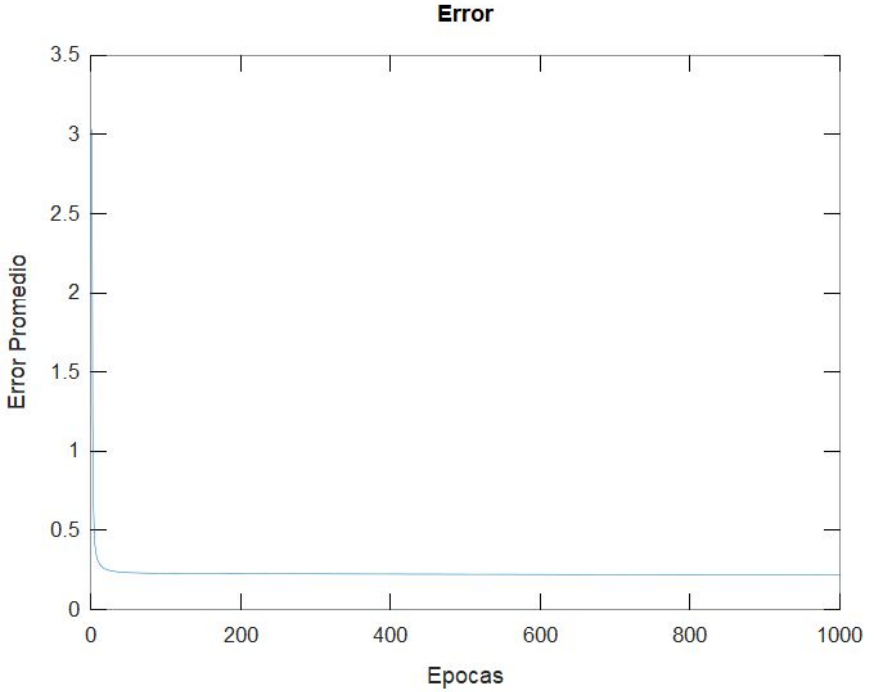


Denoising Autoencoder

Ruido n = 0.1 → Error promedio mín: **0.105**



Ruido n = 0.4 → Error promedio mín: **0.219**



Ejercicio 2

Conjunto elegido

- 92 Emojis de caras usados por Microsoft
- Imágenes 36x36x3, reescaladas de 72x72
- Valores RGB normalizados de 0.0 a 1.0

Autoencoder

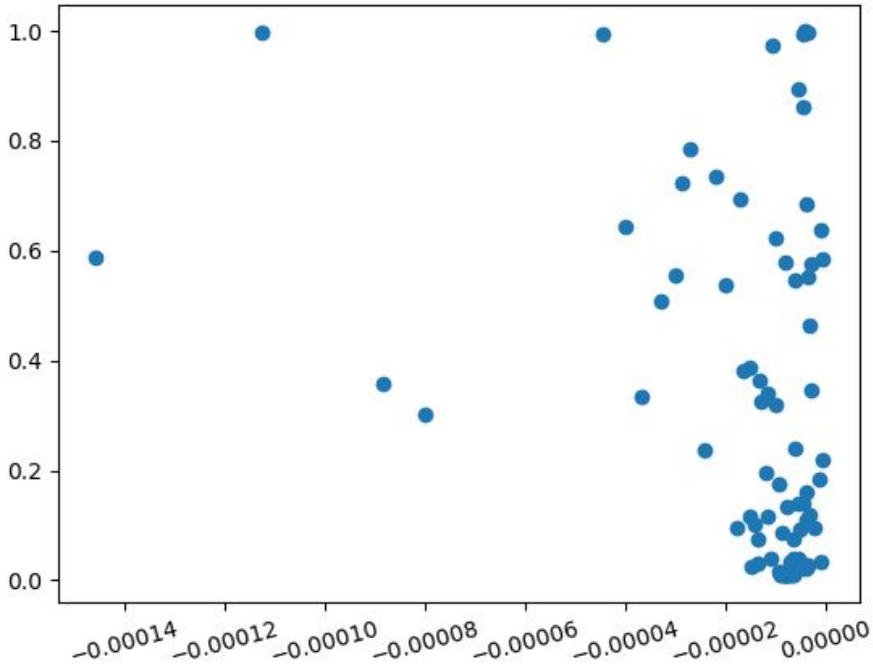
Capas:

- Entrada: 3888 (36x36x3)
- 128
- 2
- 128
- Salida: 3888

Espacio latente de 2 dimensiones

Función de activación: Logística

Autoencoder - Espacio latente



Autoencoder - Resultados

Nuevas Imágenes ⟶ a partir de valores en el espacio latente

[1.0, 0.39]

Variational Autoencoder

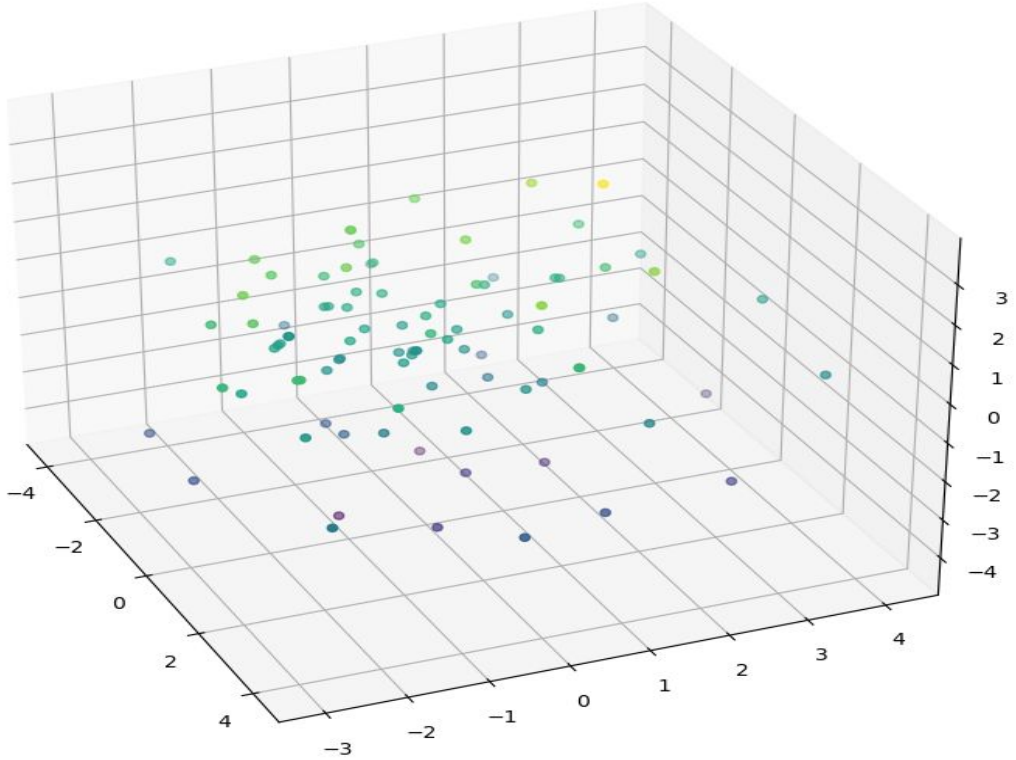
Capas:

- Entrada: 3888 (36x36x3)
- 128
- 3
- 128
- Salida: 3888

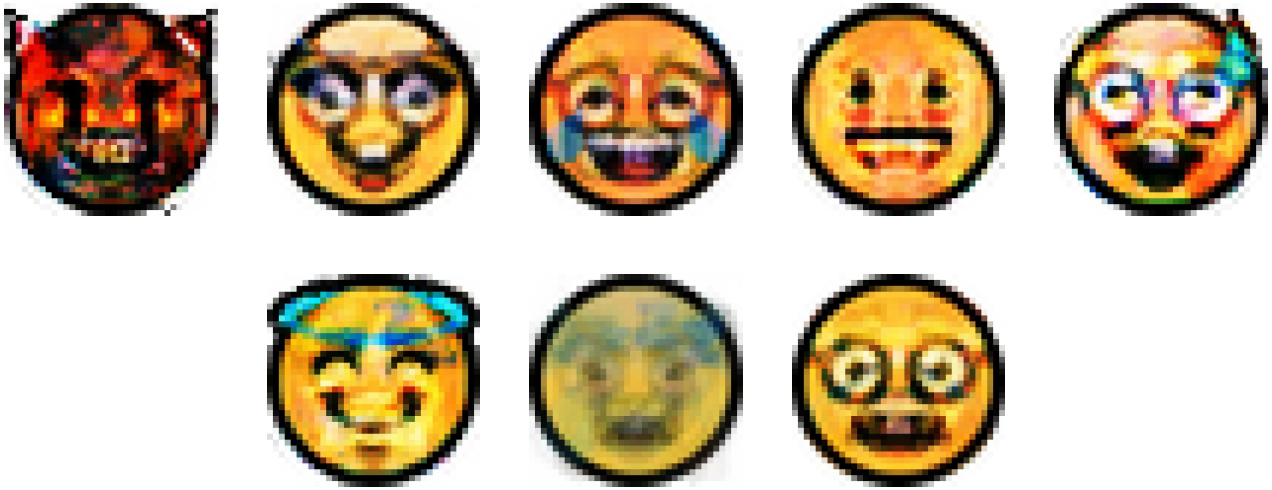
Espacio latente de 3 dimensiones

Función de activación: Logística

Variational Autoencoder - Espacio latente



Variational Autoencoder - Resultados



Conclusiones

Conclusiones

- El metodo de optimizacion que llevó a mejores resultados fue el *momentum* → Simple de implementar

- El autoencoder común no genera una estructura en el espacio latente

↓

Pueden generarse nuevas “letras” que son similares a las conocidas como no

Conclusiones

- El autoencoder común no logra generar nuevas entradas muy diferenciables del resto
- Las codificaciones de los elementos del dataset se encuentran concentradas en una región pequeña del espacio latente, salvo algunos outliers
- Los mejores resultados se lograron con codificaciones de 2 a 4 dimensiones
- Las codificaciones en el VAE están dispersas
- Con el VAE se logran generar nuevos elementos mucho más interesantes