

TRABAJO PRÁCTICO NÚMERO 5

Autoencoders

72.27 - SISTEMAS DE INTELIGENCIA ARTIFICIAL

**I N T E G R A N T E S
GRUPO 5**

Kuchukhidze, Giorgi - 67262

Madero Torres, Eduardo Federico - 59494

Ramos Marca, María Virginia - 67200

Plüss, Ramiro - 66254



Introducción

Un autoencoder es una red neuronal diseñada para aprender una representación comprimida de los datos (codificador) y reconstruirlos desde esa representación (decodificador).



Ejercicio 1A

El objetivo es entrenar un autoencoder para representar caracteres binarios en un espacio latente de **2 dimensiones**, asegurando un error máximo de reconstrucción de 1 píxel incorrecto.

Esto implica diseñar arquitecturas de codificación y decodificación adecuadas y optimizar la red para aprender todo el conjunto de datos o un subconjunto, evaluando los límites de la capacidad de representación del modelo.

Además, se busca visualizar los datos en el espacio latente y generar una nueva “letra” no incluida en el entrenamiento.



Ejercicio 01A

¿Qué *hiperarámetros* elegimos variar?

1	2	3	4	5	6	Epochs
Arquitectura	Inicialización de pesos	Learning rate	Loss function	Act. function		
Sobreparametrización Balanceada Subparametrización	Uniform Xavier He	0.01 0.001 Lr variante	Mse Cross entropy	Relu Tanh Sigmoid		6000 al 12000

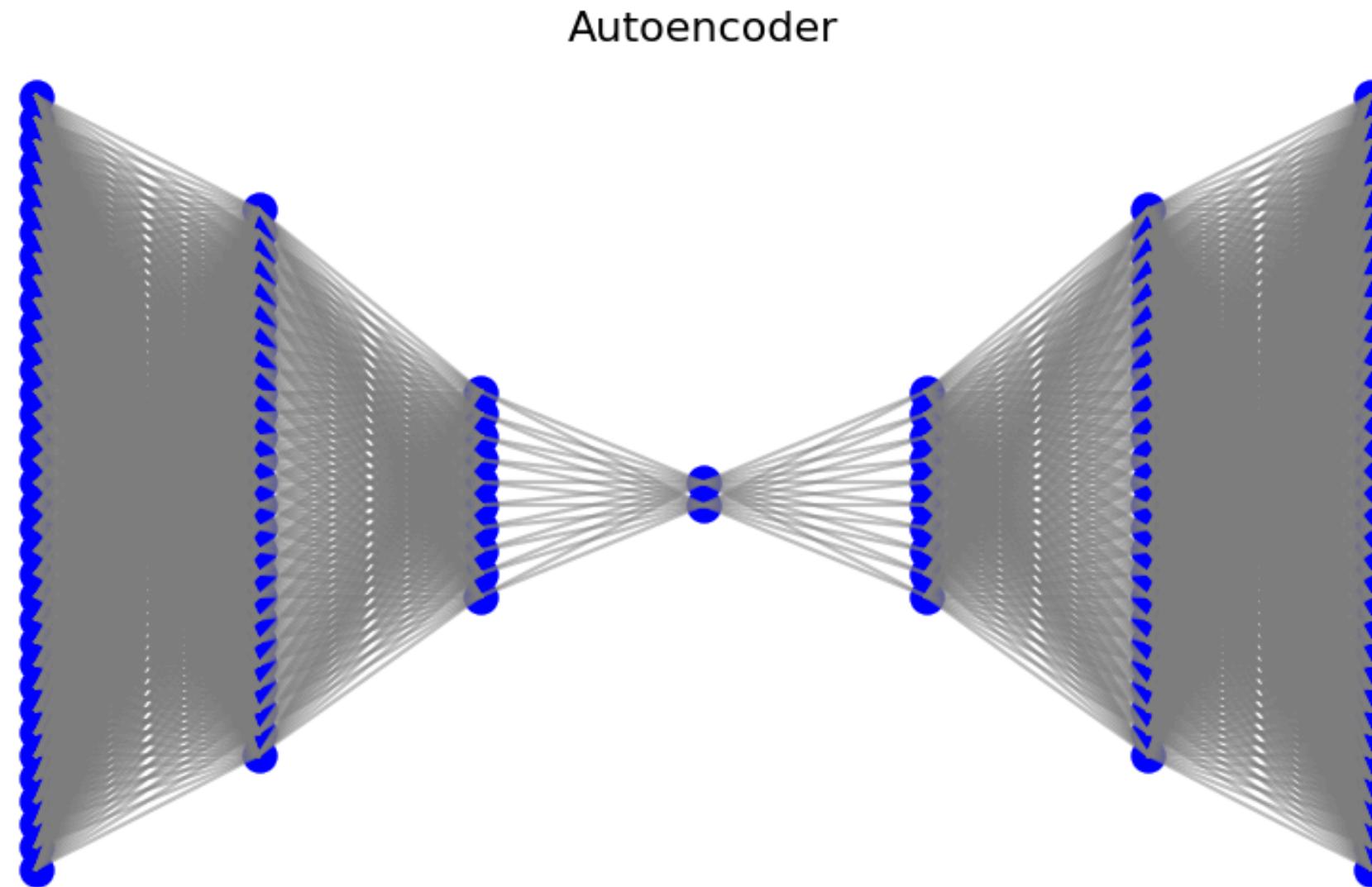
Variemos la arquitectura

Sobreparametrización
Balanceada
Subparametrización

Autoencoder

↑ Ejercicio 1A

Tipo de Arquitectura	Capas
Sobreparametrización	[35, 50, 100, 2, 100, 50, 35]
Balanceada	[35, 25, 10, 2, 10, 25, 35]
Subparametrización	[35, 10, 5, 2, 5, 10, 35]

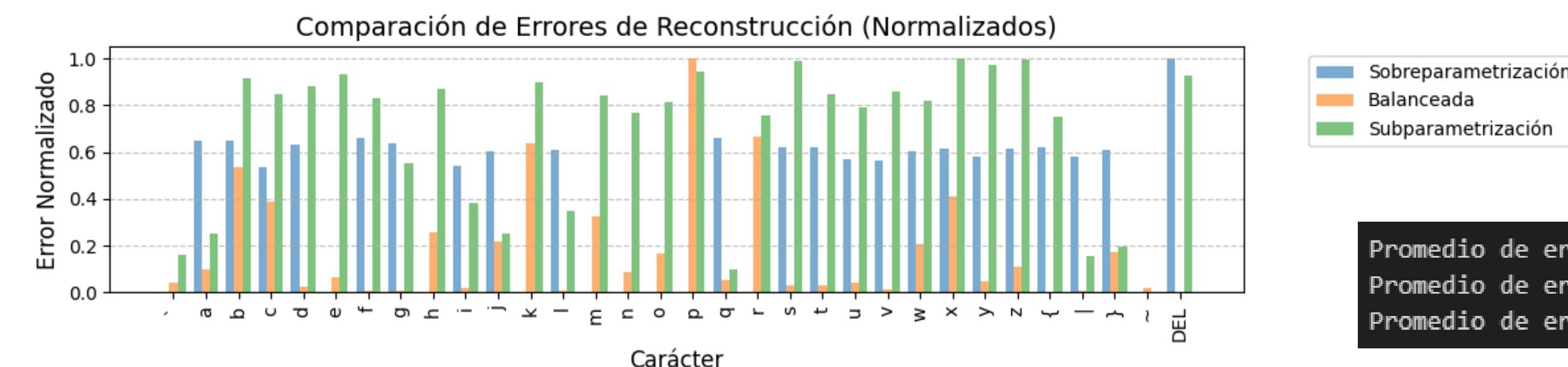
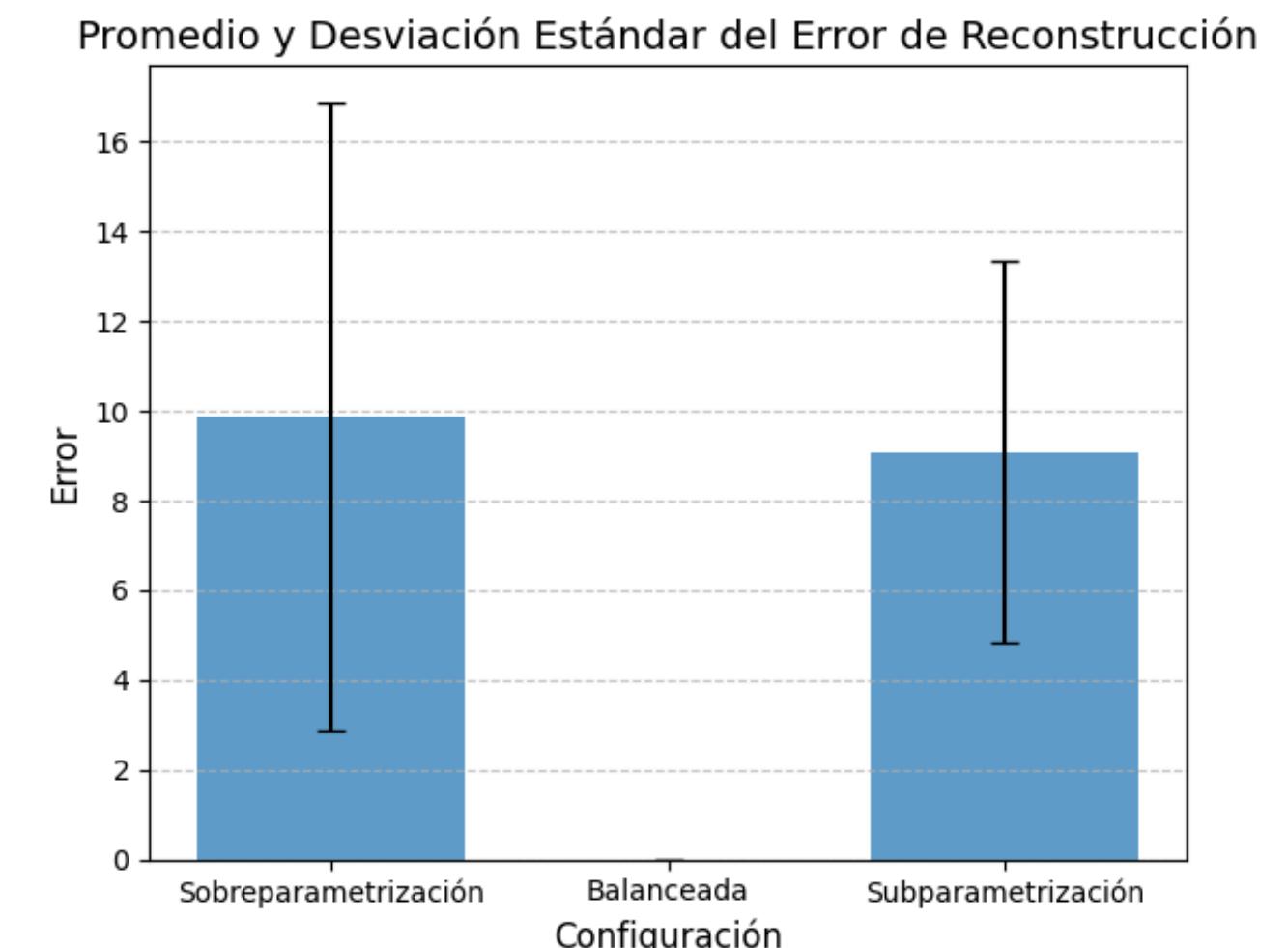
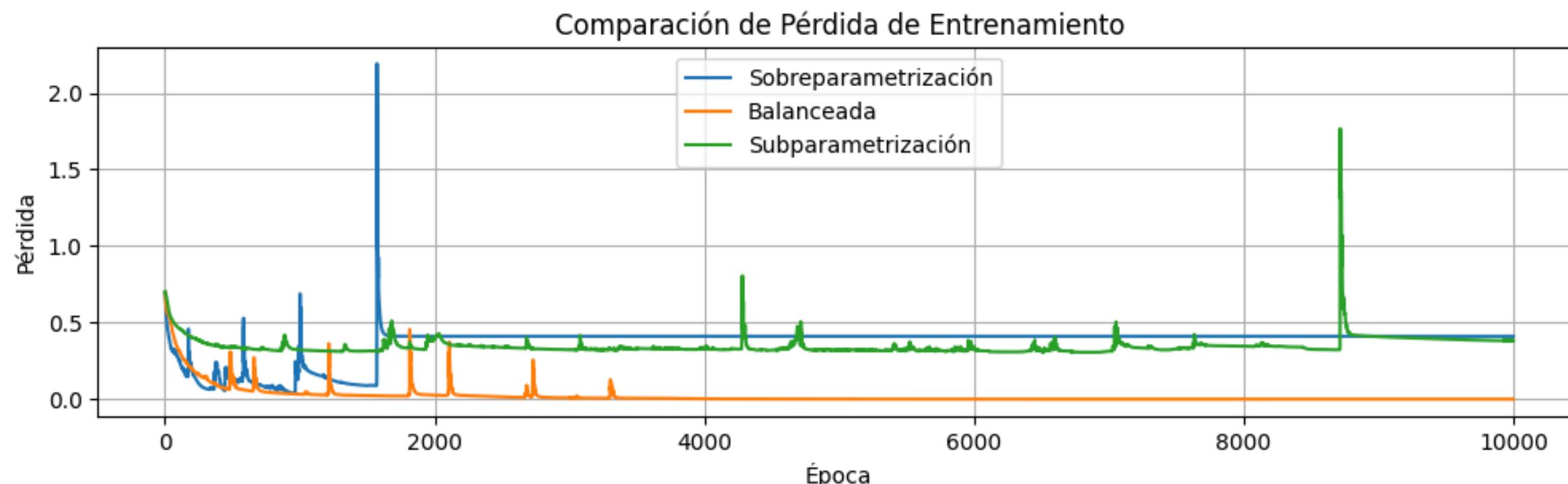


```
"learning_rate": 0.01,  
"epochs": 10000,  
"activation_fn": "relu",  
"loss_function": "cross_entropy_loss",  
"optimizer": "adam",  
"n_runs": 1,  
"seed": 42,  
"init_method": "uniform"
```

Autoencoder



Ejercicio 1A



Promedio de error (Sobreparametrización): 9.875003656250001
Promedio de error (Balanceada): 0.00028925
Promedio de error (Subparametrización): 9.07439065625

Autoencoder

↑ Ejercicio 1A

Lr: constante vs variante

Lr (cte) = 0.01

vs

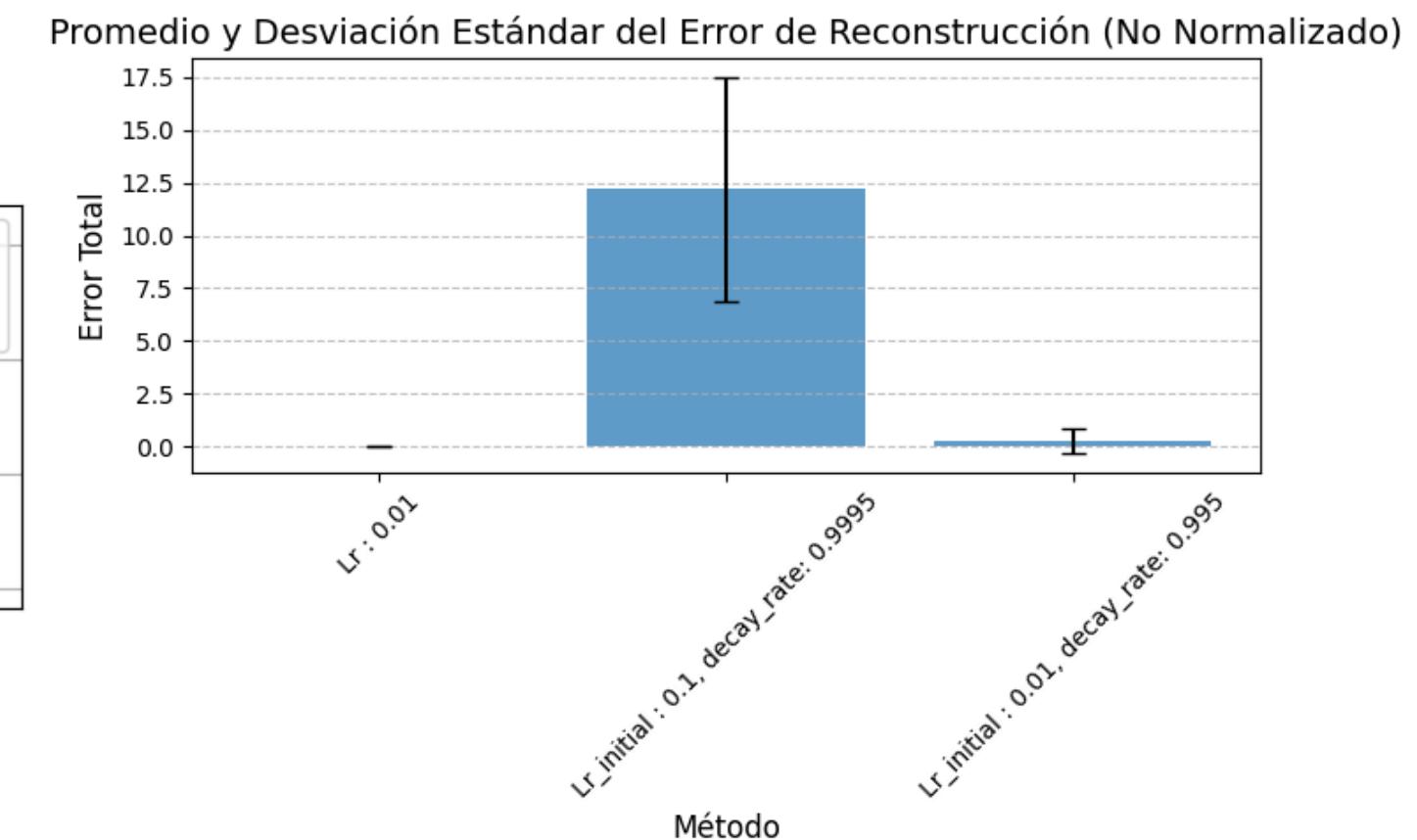
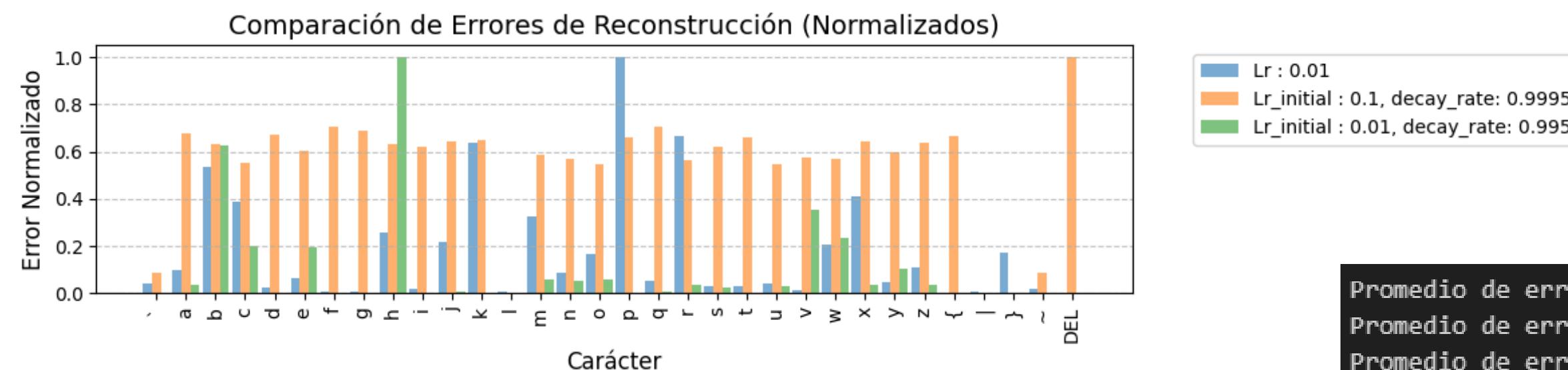
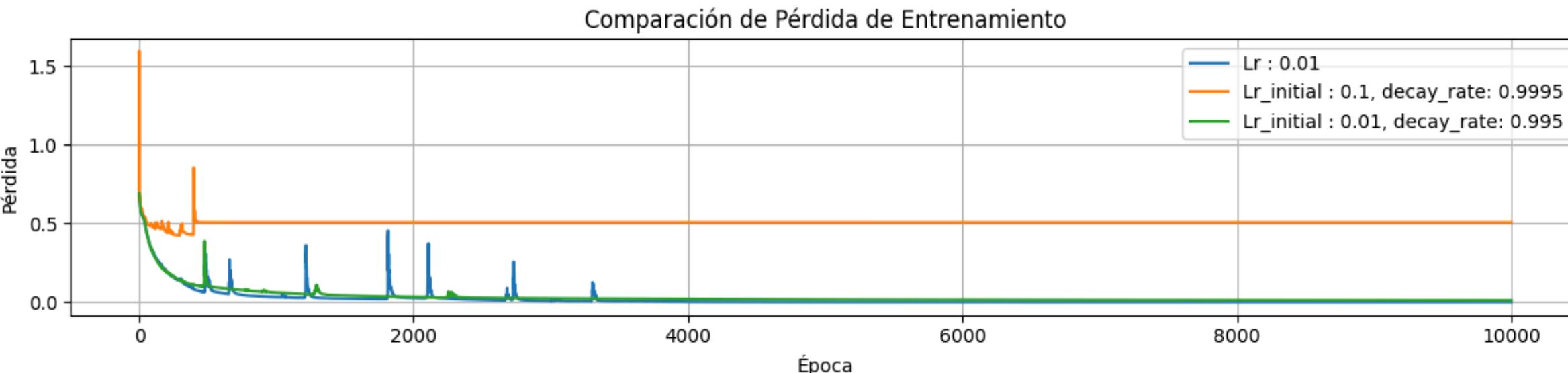
$Initial_Lr = 0.01$
 $decay_rate = 0.995$

vs

$Initial_Lr = 0.1$
 $decay_rate = 0.9995$

Autoencoder

↑ Ejercicio 1A



Promedio de error (Lr : 0.01): 0.00028925
Promedio de error (Lr_initial : 0.1, decay_rate: 0.9995): 12.18987165625
Promedio de error (Lr_initial : 0.01, decay_rate: 0.995): 0.2793526875

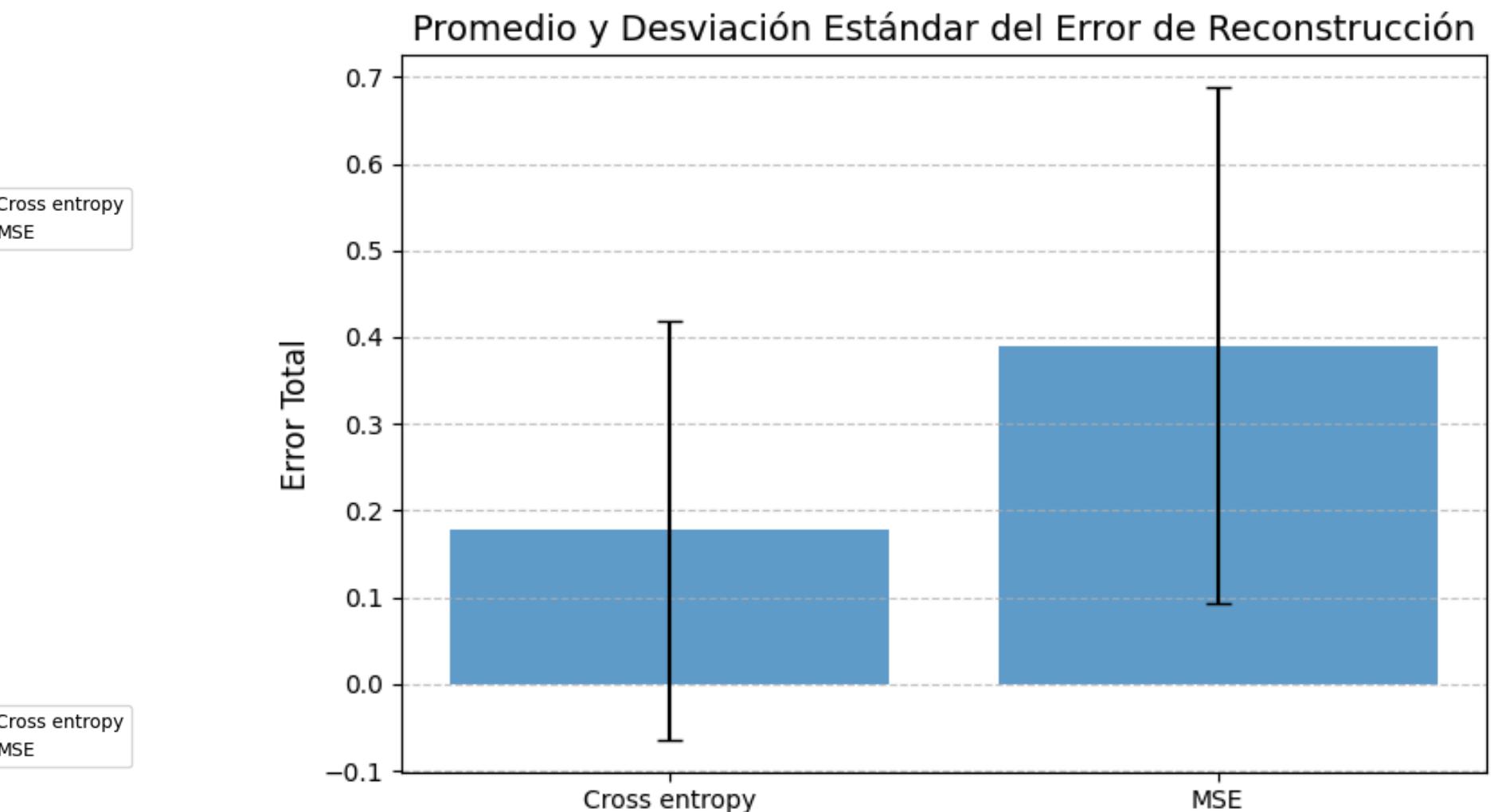
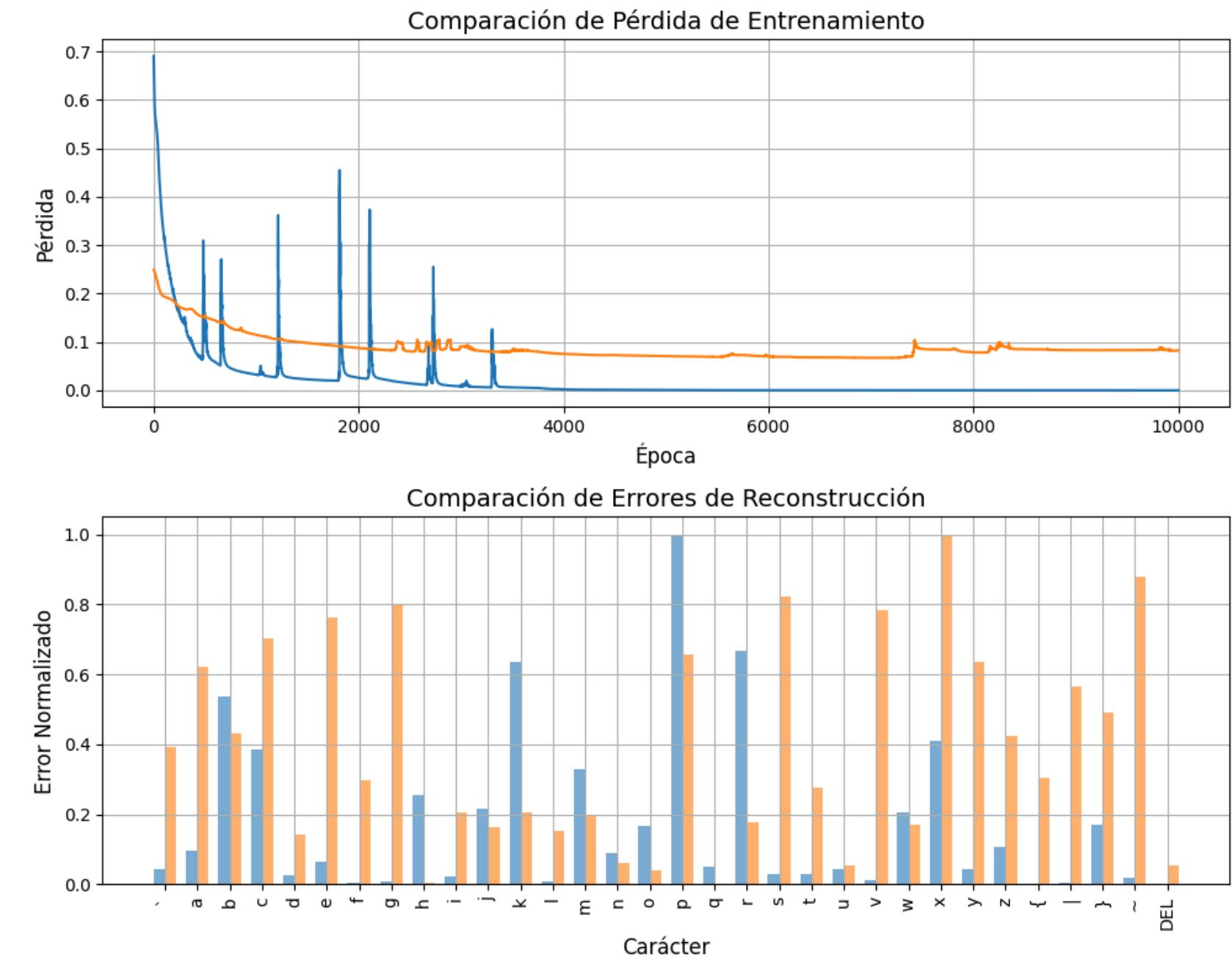
Loss Function

MSE

Cross Entropy

Autoencoder

↑ Ejercicio 1A



Promedio de error (Cross entropy): 0.17779503105590058
Promedio de error (MSE): 0.39037808756776793

Variemos epochs

6000

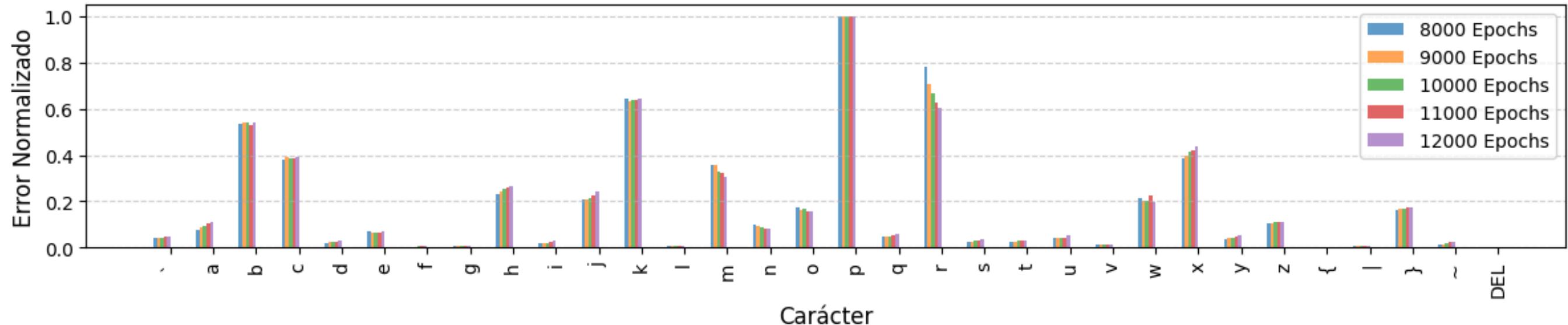
al

12000

Autoencoder

↑ Ejercicio 1A

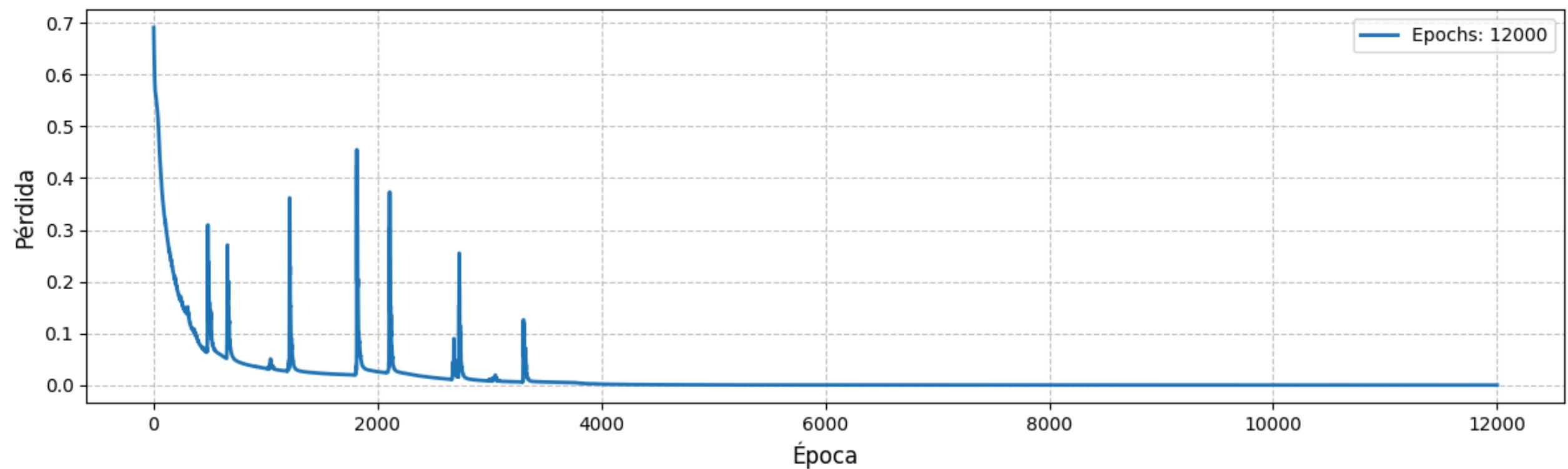
Comparación de Errores de Reconstrucción Normalizados por Configuración



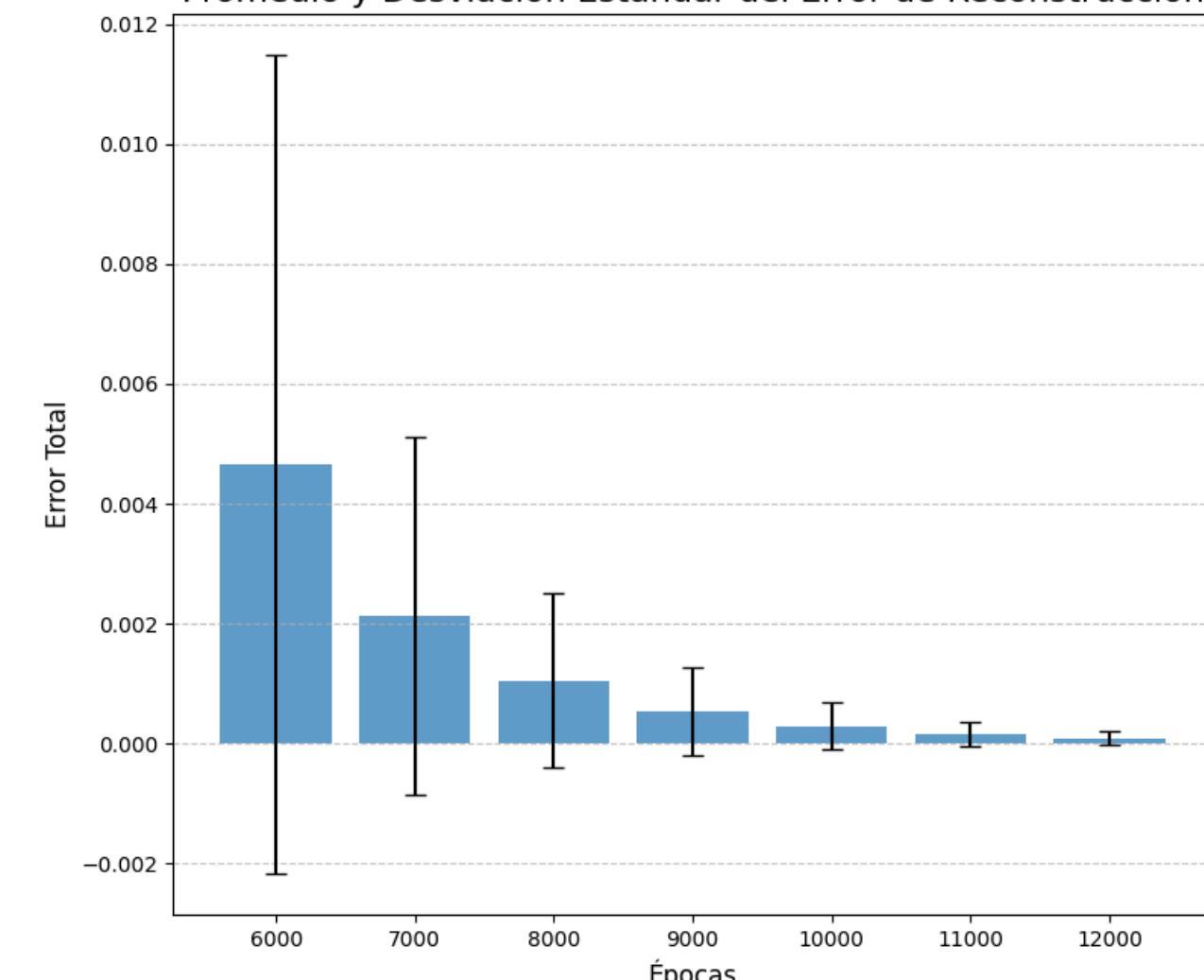
Valores medios:

Épocas 6000: 0.0047
Épocas 7000: 0.0021
Épocas 8000: 0.0011
Épocas 9000: 0.0005
Épocas 10000: 0.0003
Épocas 11000: 0.0002
Épocas 12000: 0.0001

Entrenamiento



Promedio y Desviación Estándar del Error de Reconstrucción





Ejercicio 01A

¿Qué *hiperarámetros* elegimos variar?

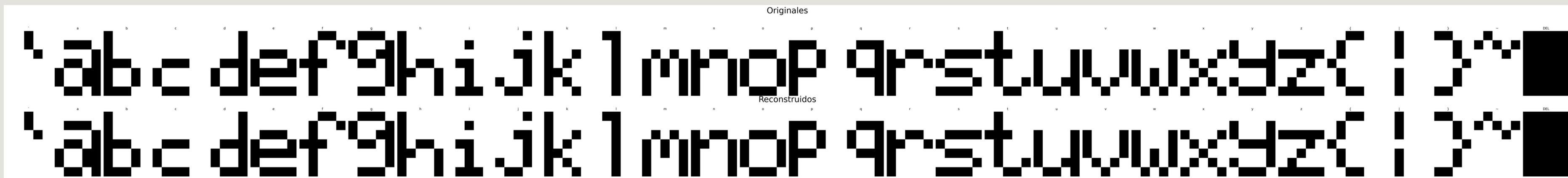
1	2	3	4	5	6	Epochs
Arquitectura	Inicialización de pesos	Learning rate	Loss function	Act. function		12000
Sobreparametrización Balanceada	Uniform	0.01	Mse	Relu		
Subparametrización	Xavier	0.001	Cross entropy	Tanh		
	He	lr variante		Sigmoid		

```
"layers": [35, 25, 10, 2, 10, 25, 35],  
"learning_rate": 0.01,  
"epochs": 12000,  
"activation_fn": "relu",  
"loss_function": "cross_entropy_loss",  
"optimizer": "adam",  
"n_runs": 1,  
"seed": 42,  
"init_method": "uniform"
```

Autoencoder

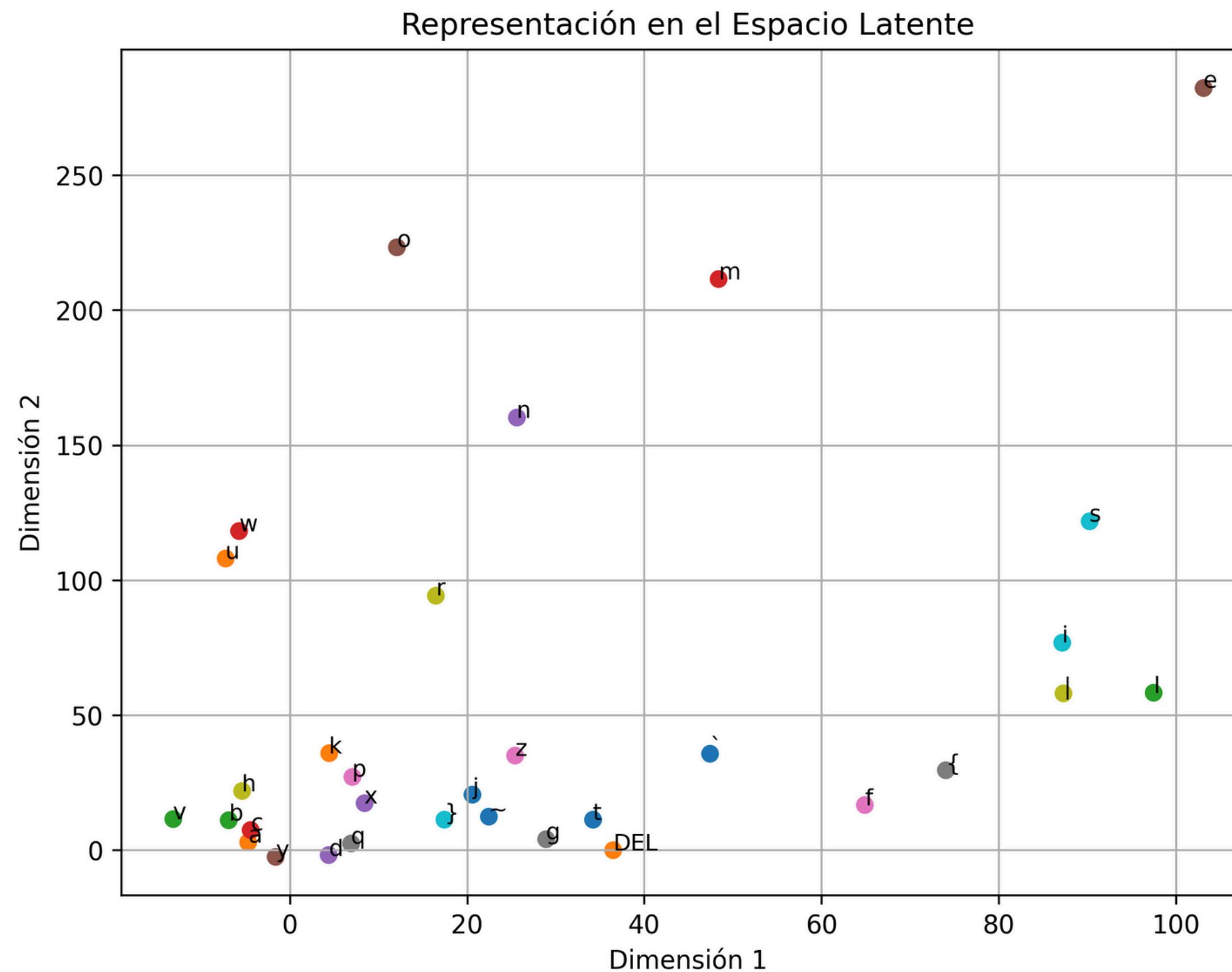
↑ Ejercicio 1A

Reconstrucción



Autoencoder

↑ Ejercicio 1A



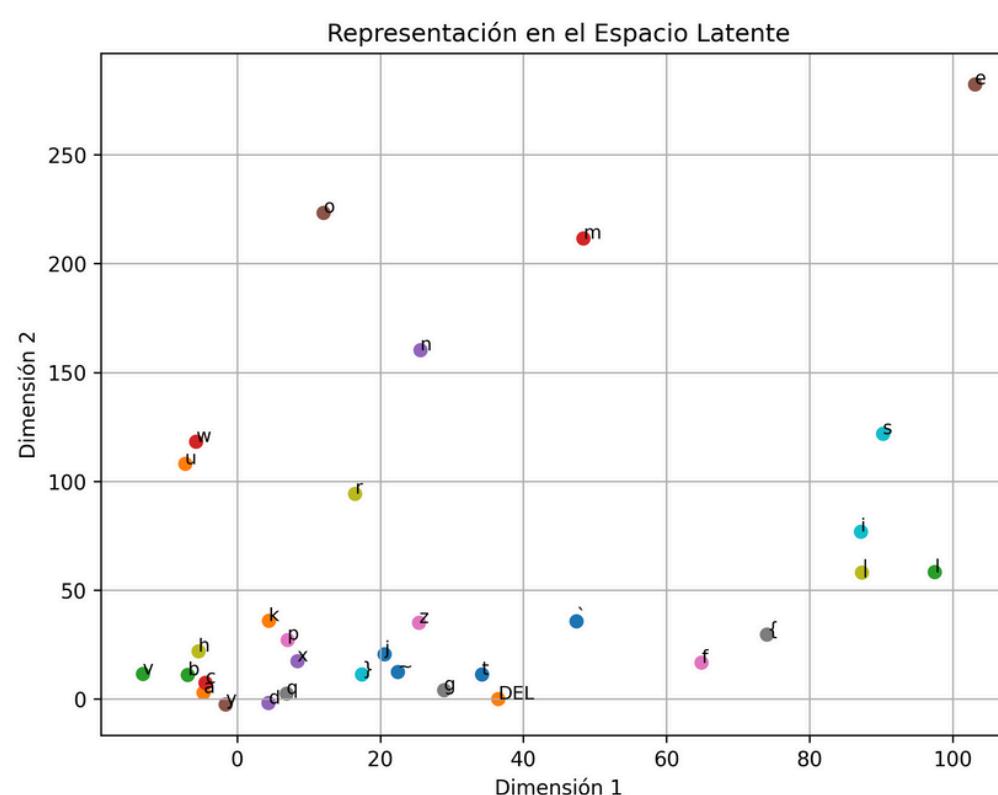
Autoencoder



Ejercicio 1A

¿Que pasa si queremos construir una letra que no esta en el conjunto de entrenamiento?

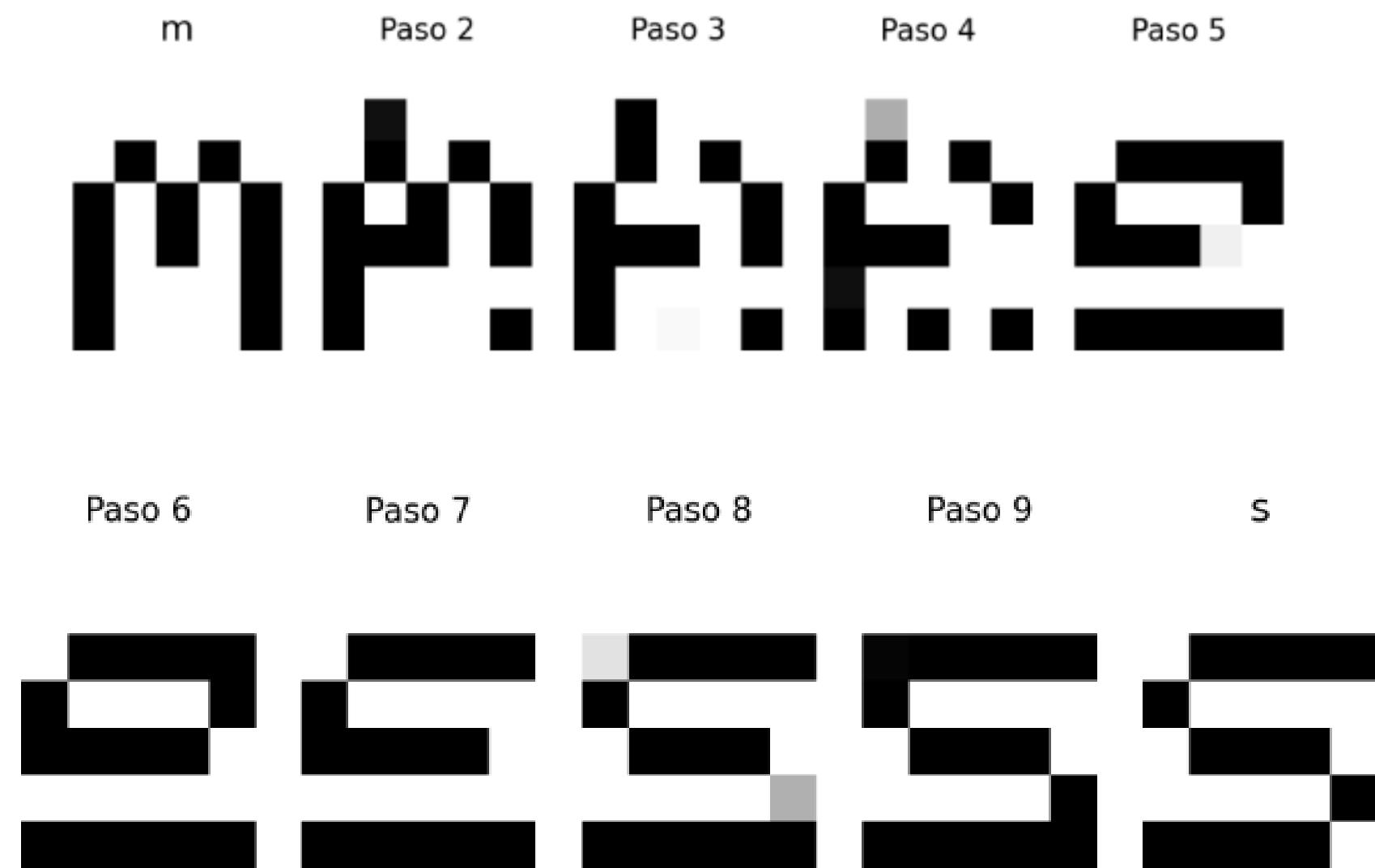
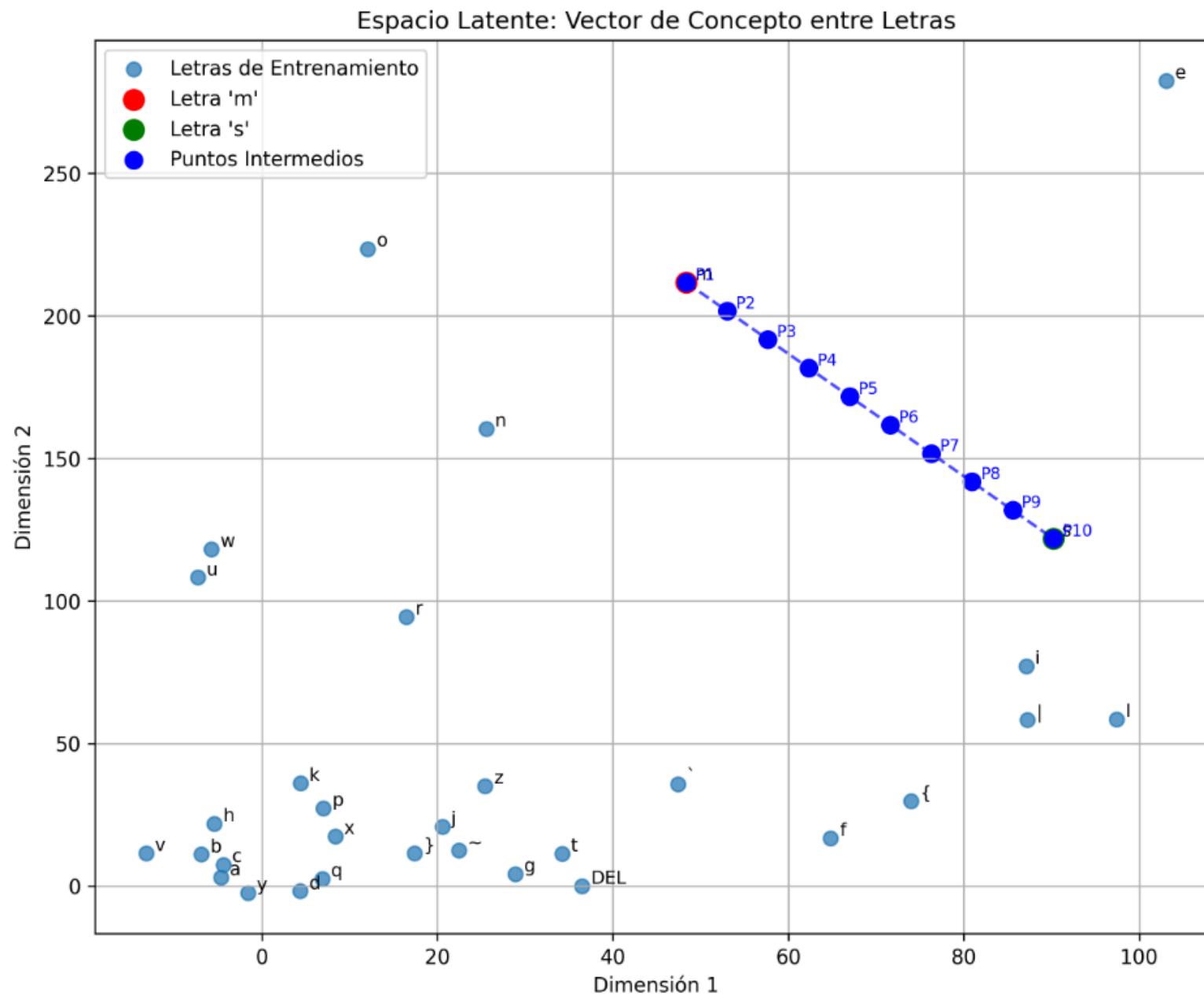
Podemos utilizar el espacio latente en orden de reconstruir una imagen dado un punto aleatorio en dicho espacio.



Autoencoder

↑ Ejercicio 1A

Concept vector desde “m” a “s”

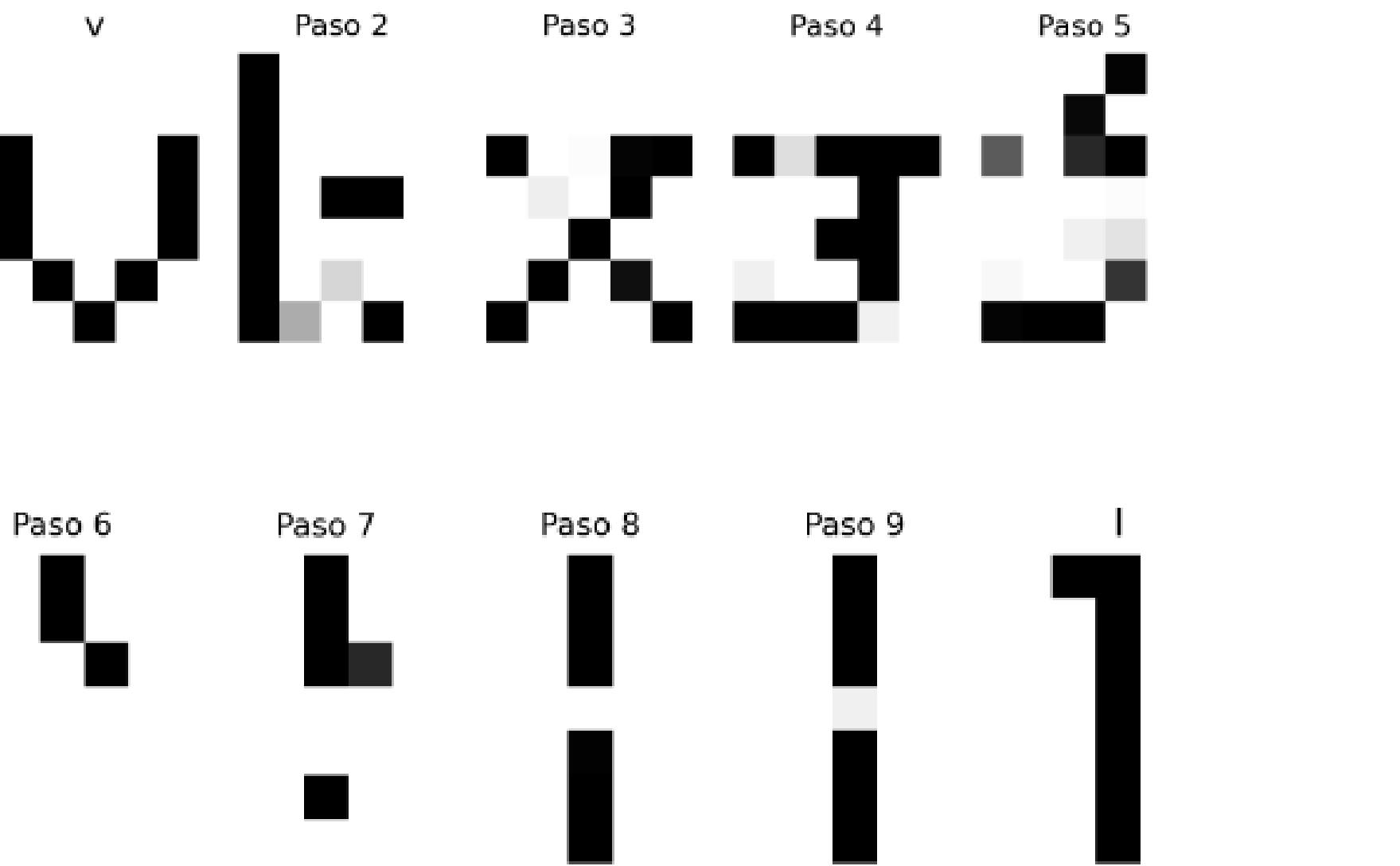
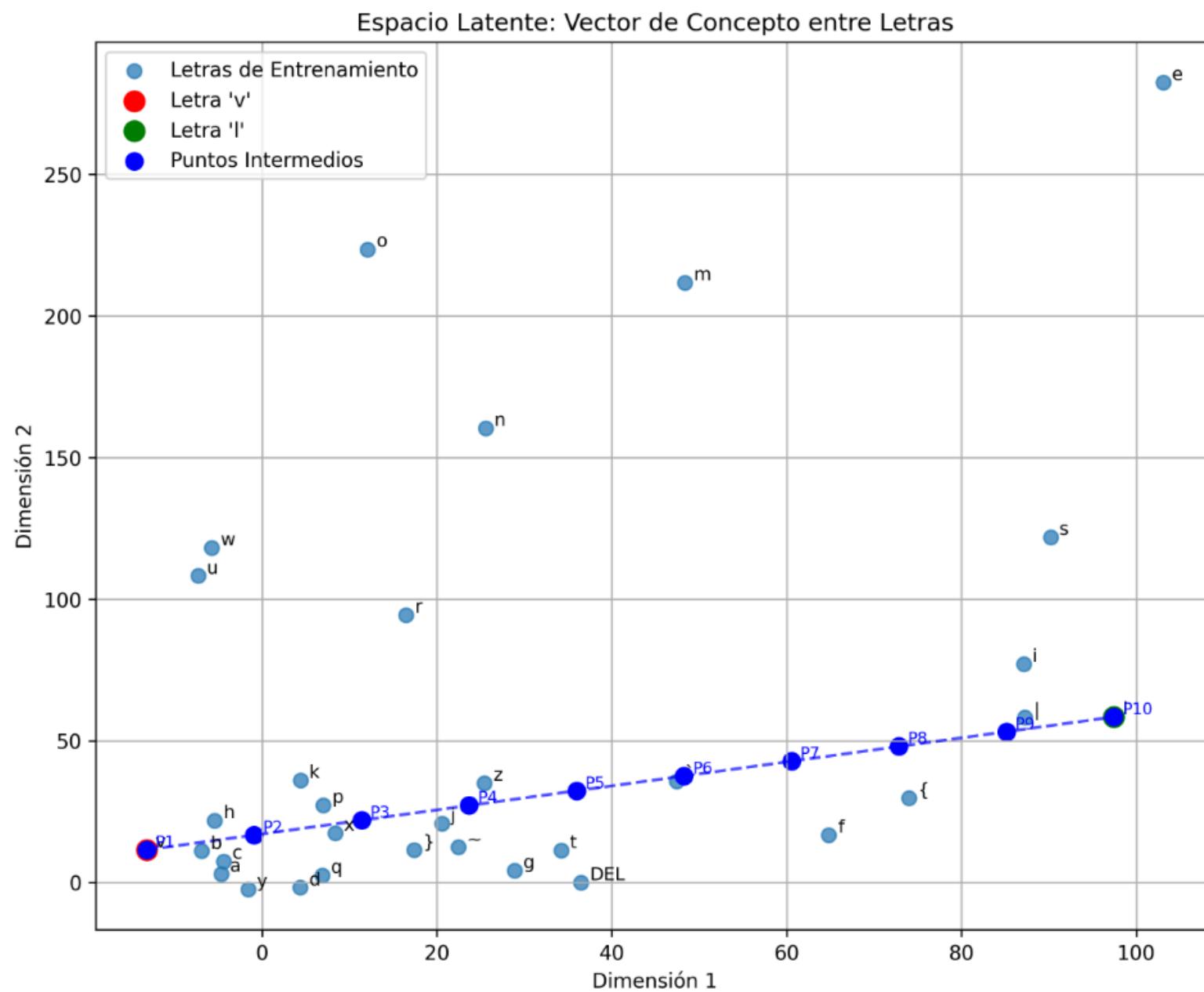


Autoencoder



Ejercicio 1A

Concept vector desde “v” a “l”



Conclusiones ejercicio 1A

Encontrar la arquitectura de red que mejor se ajusta a la necesidad **requiere de mucha prueba y error.**

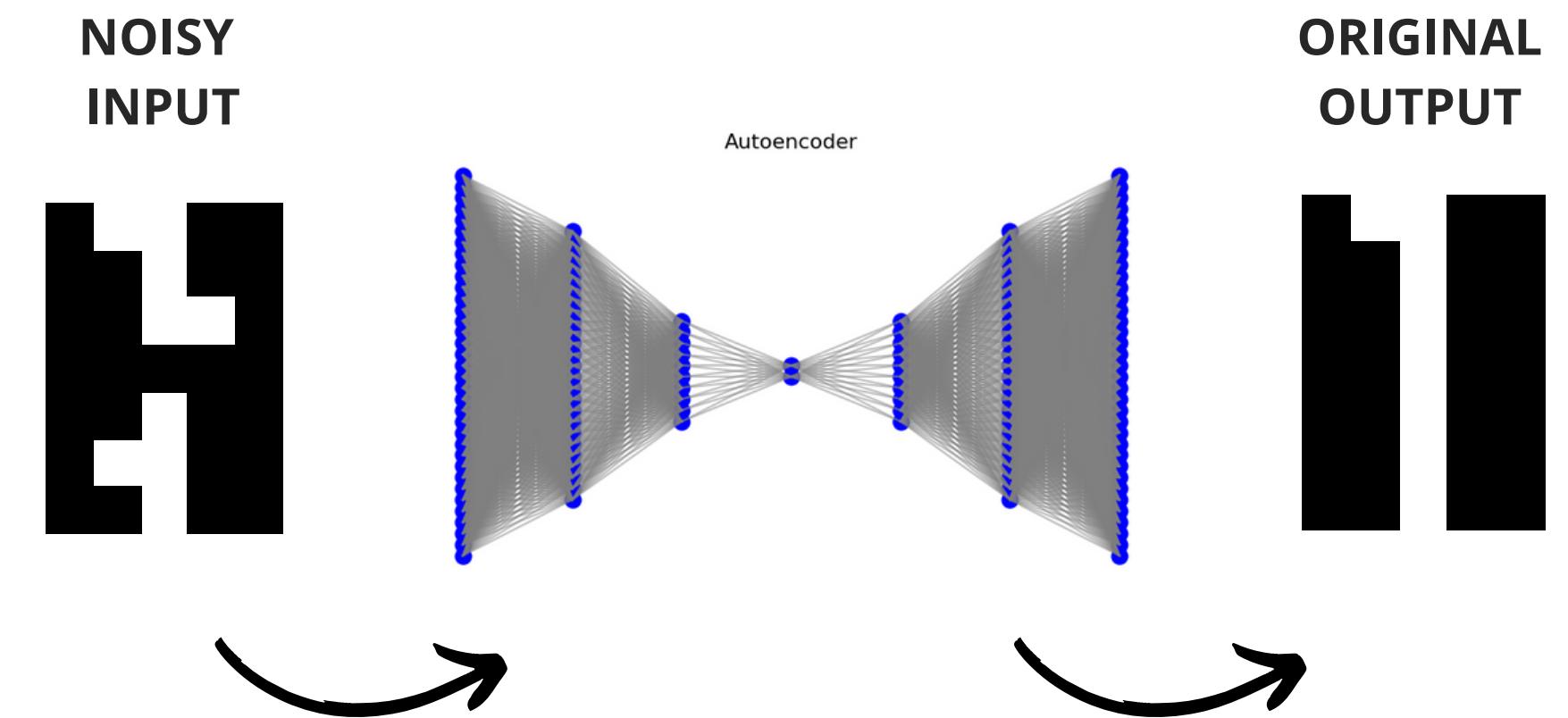
Se logró una **excelente reconstrucción** del conjunto de datos.

El autoencoder básico **logró mapear caracteres** en el espacio latente, pero la separación entre ellos no es completamente clara. Algunas letras quedaron acumuladas en ciertas regiones del espacio latente, lo que sugiere que **el modelo no logró representar de forma totalmente diferenciada todas las letras.**

El modelo no genera caracteres nuevos, el modelo tiende a crear deformaciones de los caracteres que ya ha aprendido, o genera caracteres que están cerca de las representaciones latentes de los caracteres conocidos.

Ejercicio 1B

Denoising autoencoder

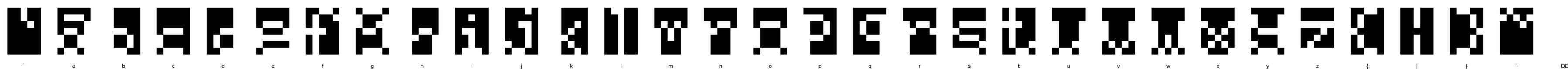


↑ Ejercicio 1B

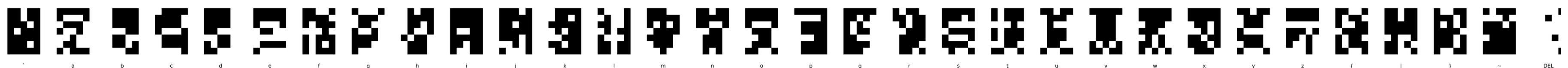
DATOS DE ENTRADA

Ruido salt and pepper

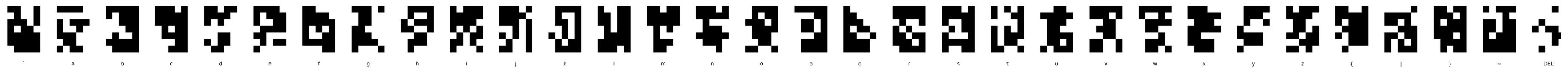
0%



20%



50%



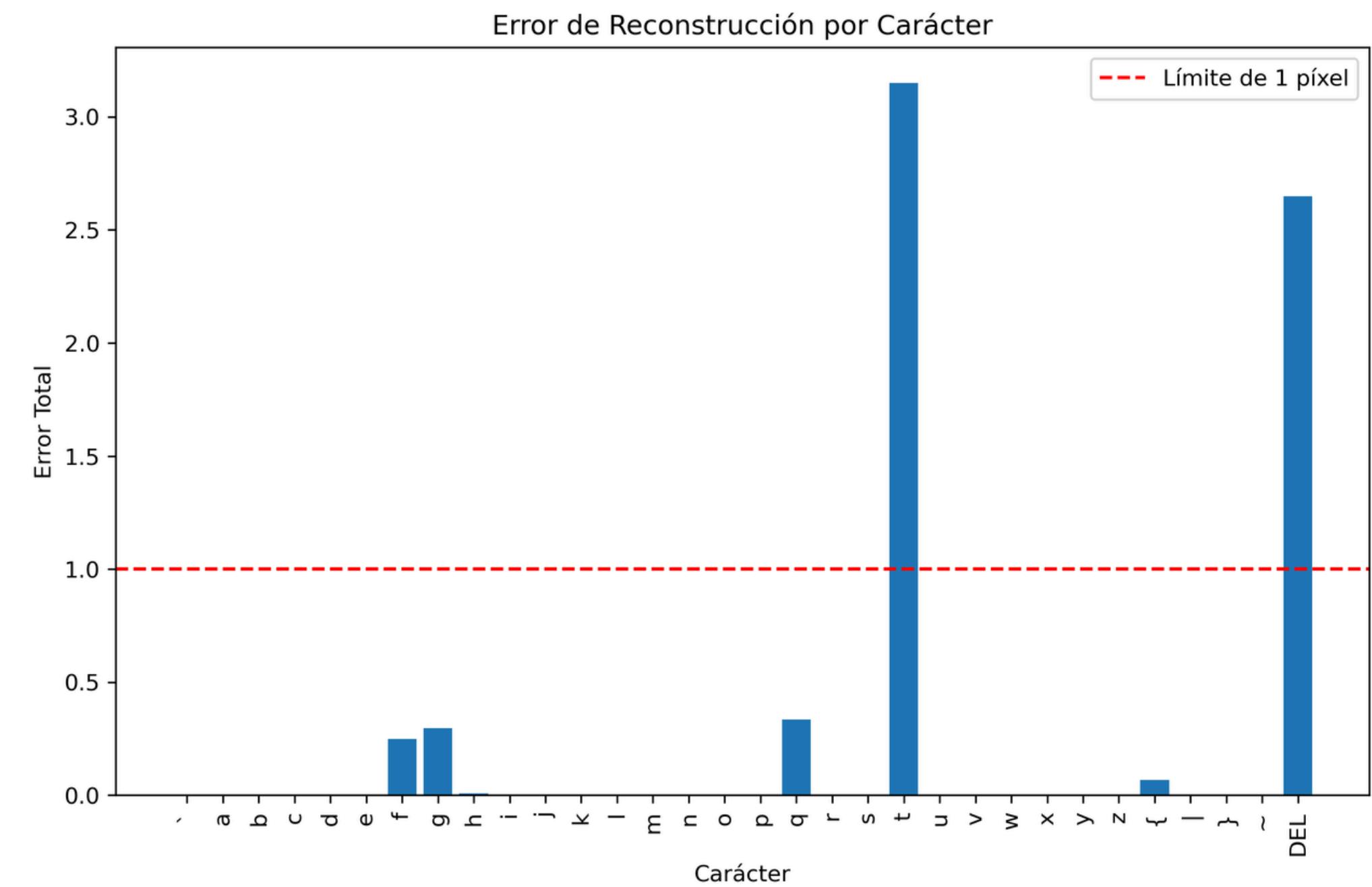
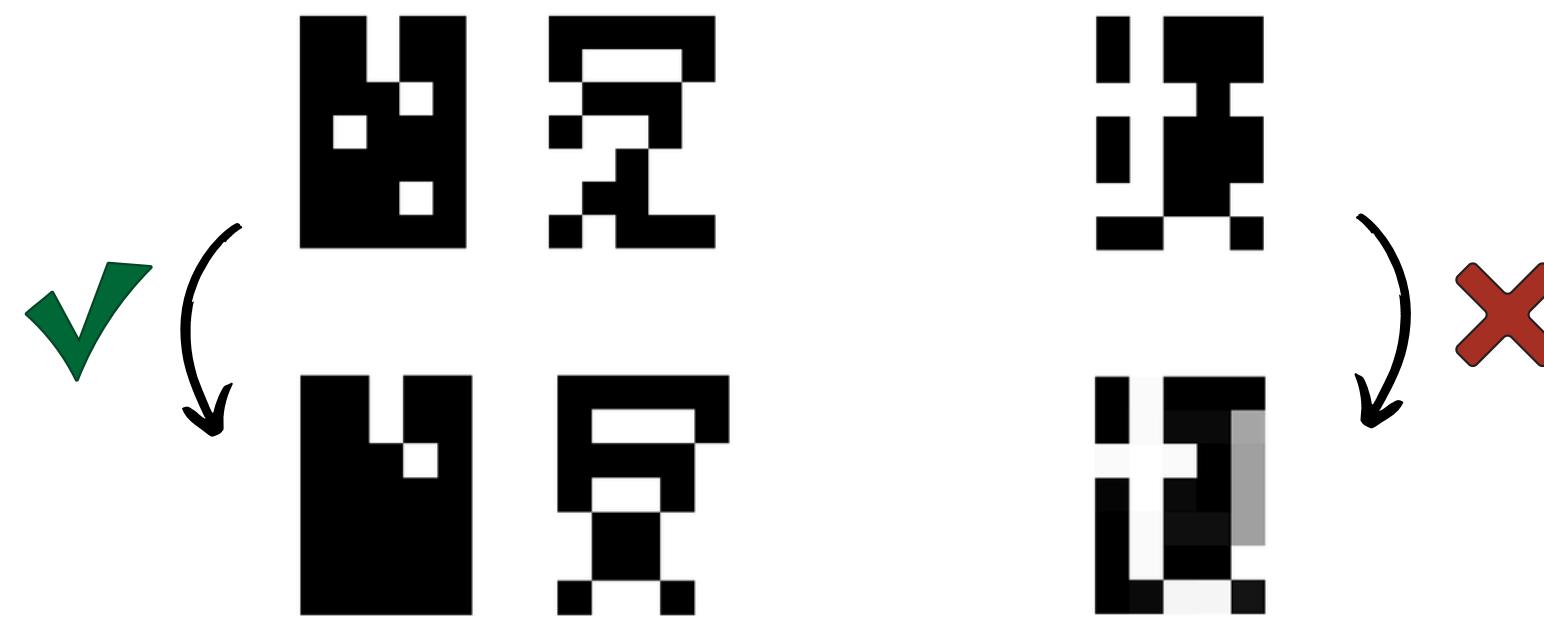
90%



↑ Ejercicio 1B

20% Salt and Pepper

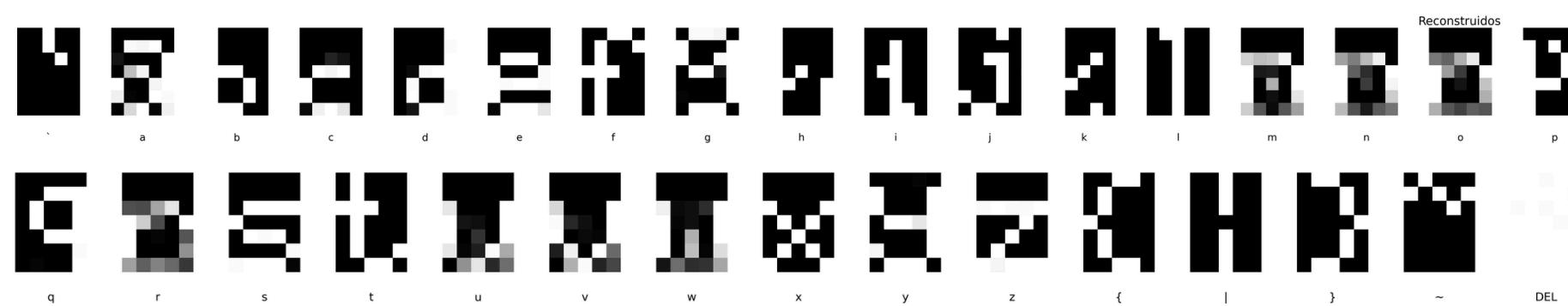
Arquitectura:
35 25 10 2 10 25 35
Epochs: 12000
LR: 0.01



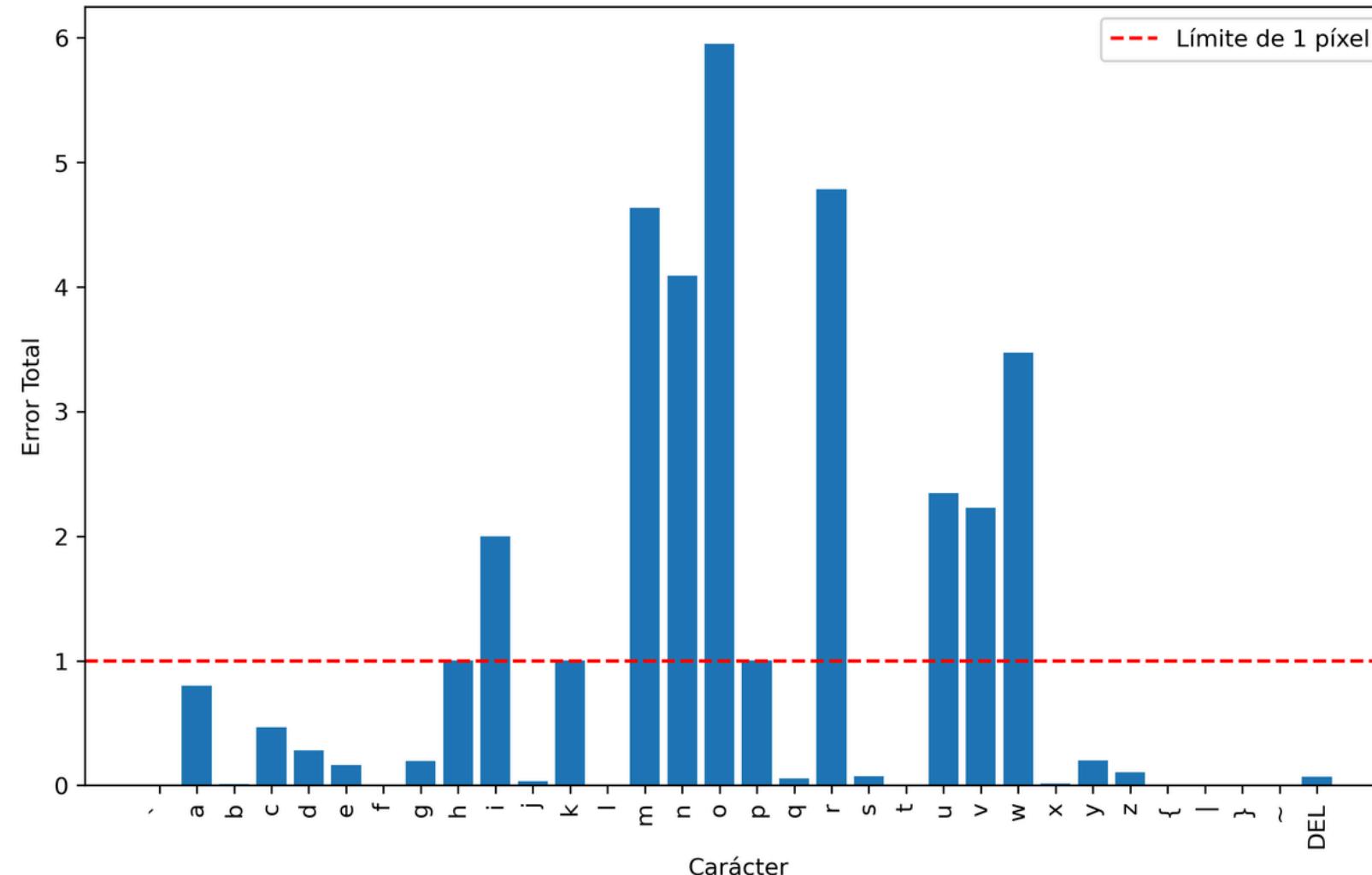
Denoising Autoencoder

↑ Ejercicio 1B

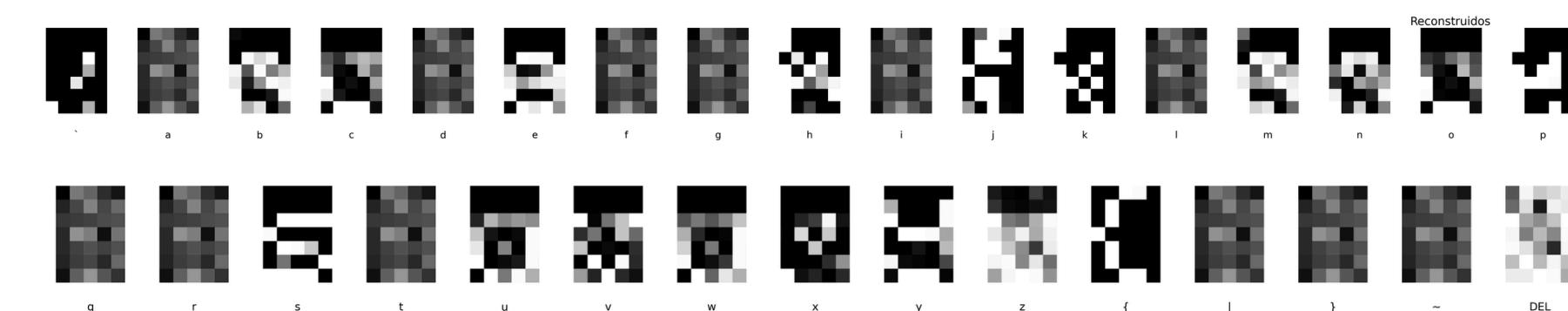
50% Salt and Pepper



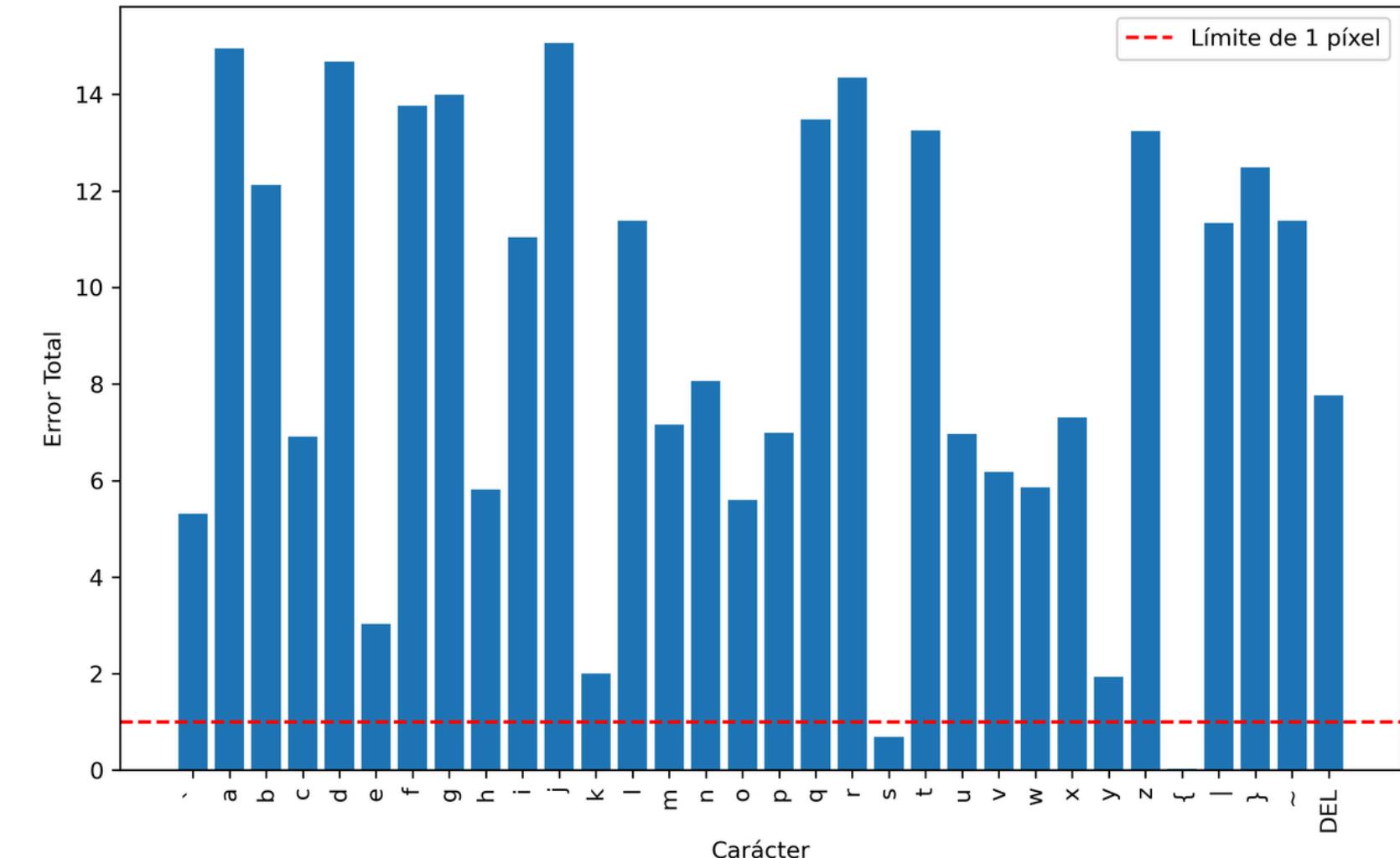
Error de Reconstrucción por Carácter



90% Salt and Pepper



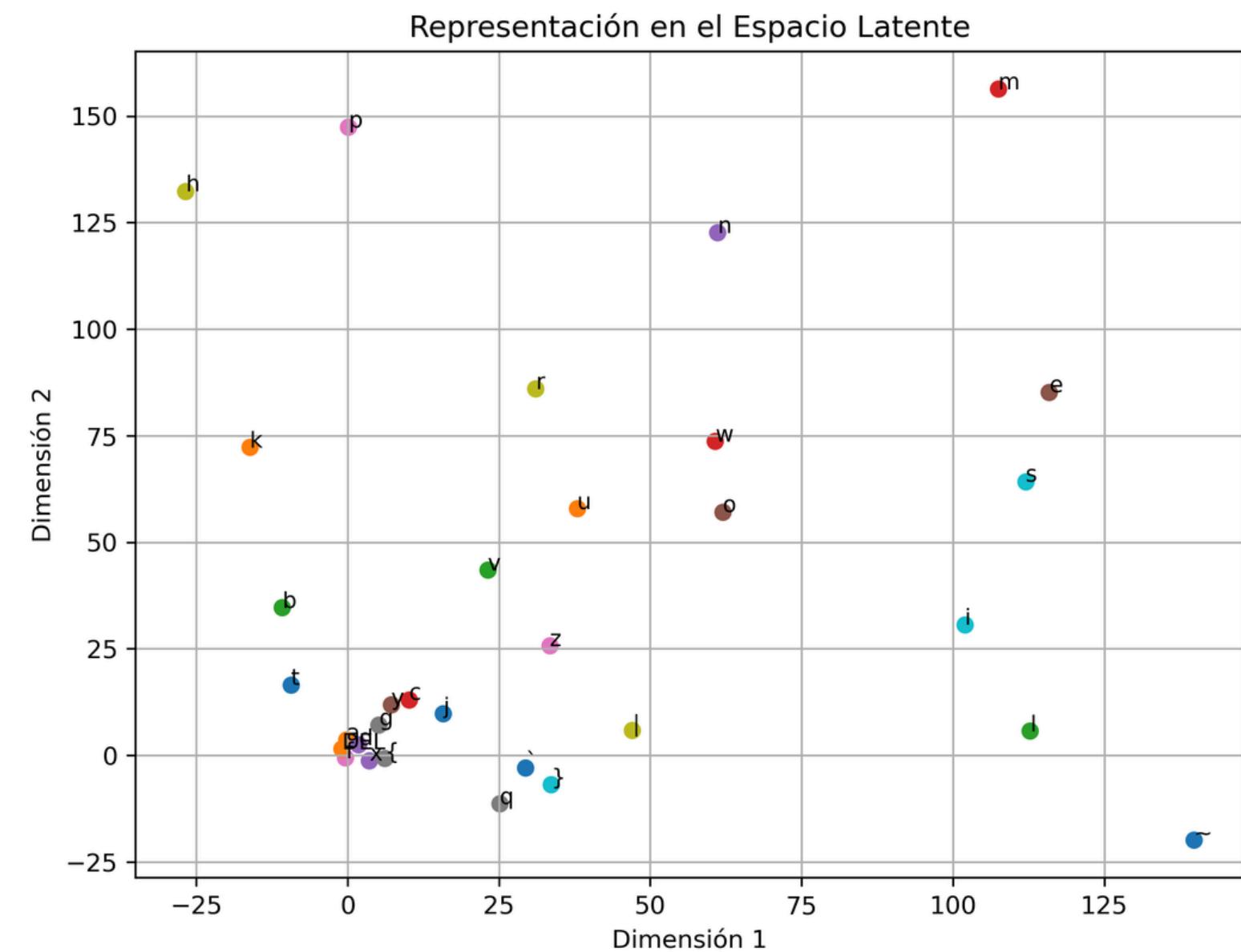
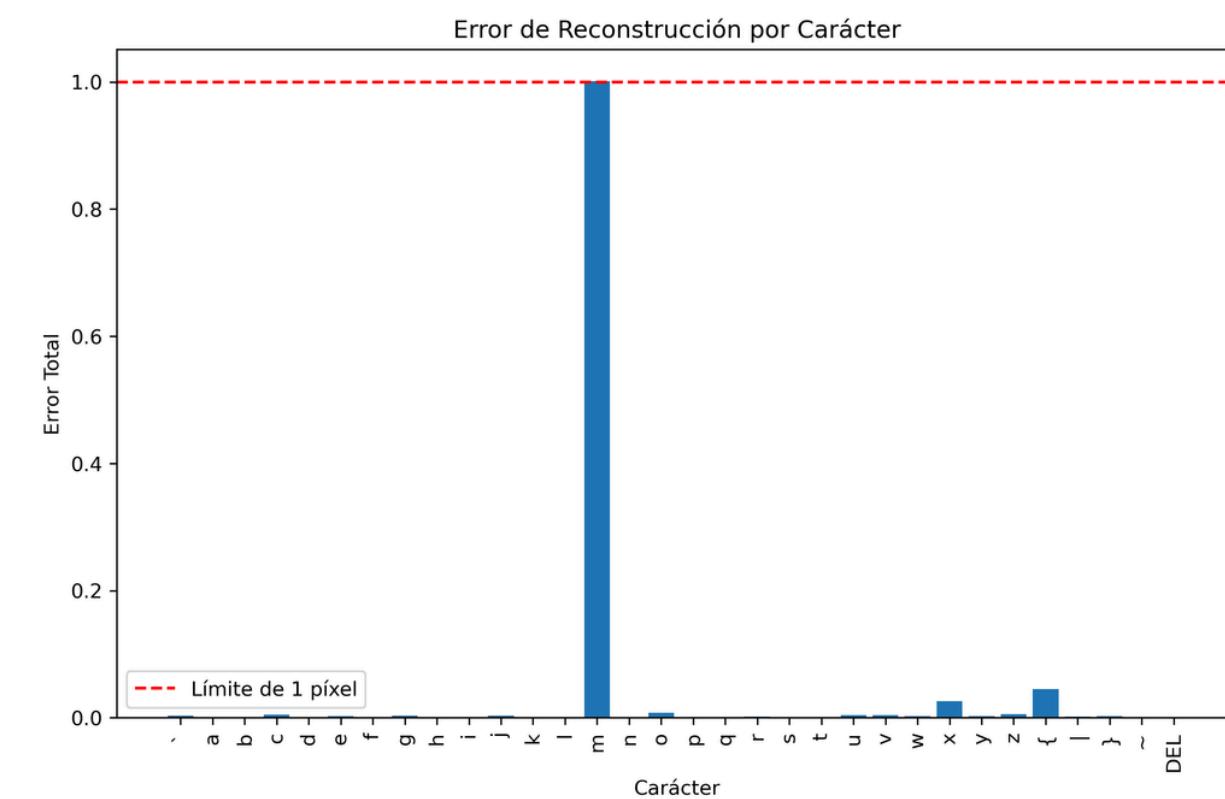
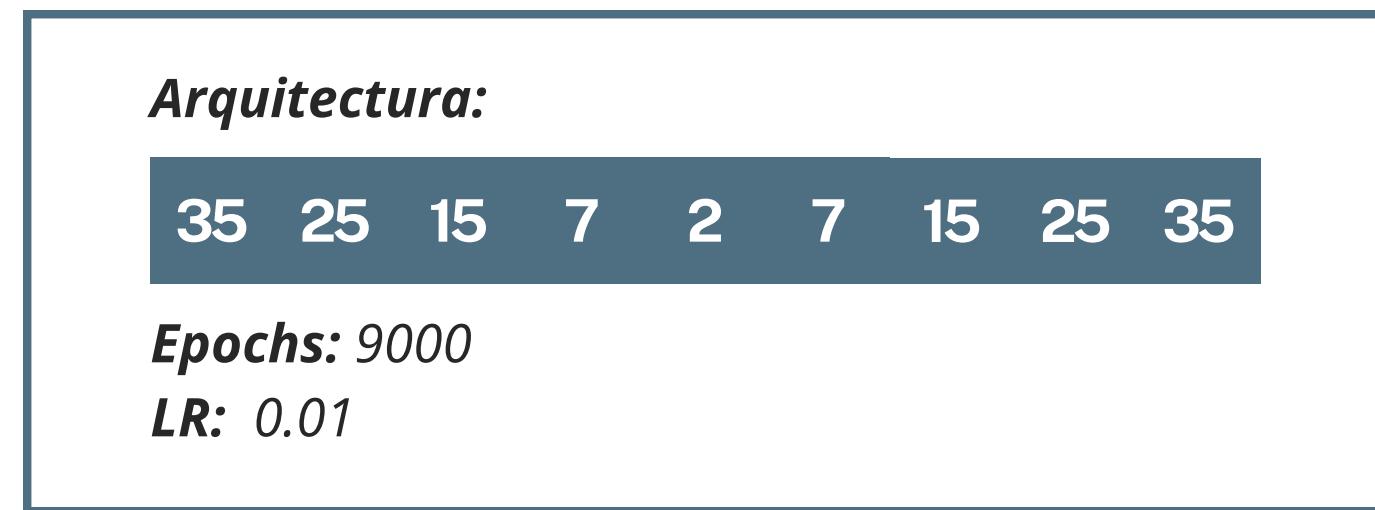
Error de Reconstrucción por Carácter



Denoising Autoencoder

↑ Ejercicio 1B

20% Salt and Pepper

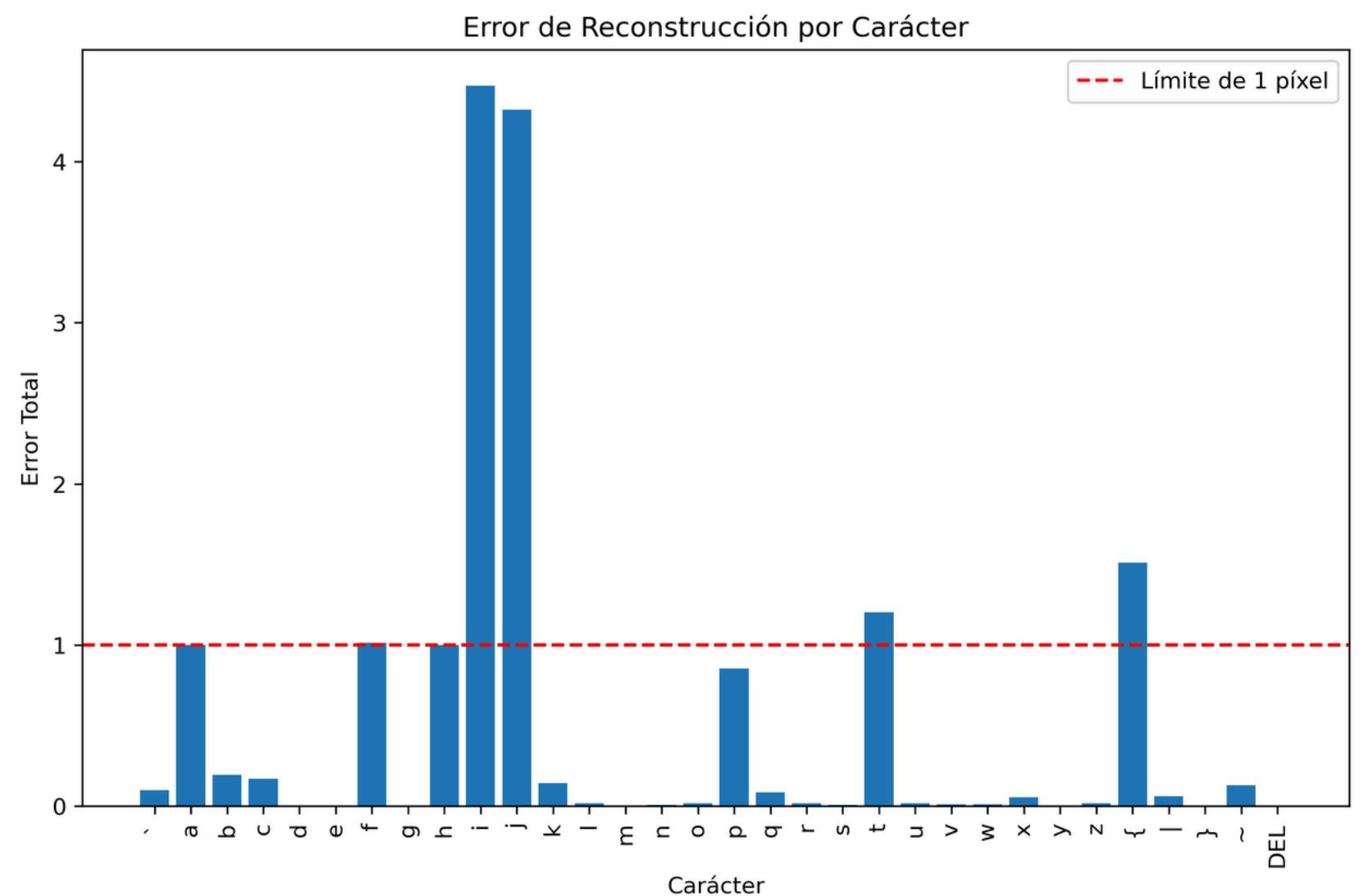
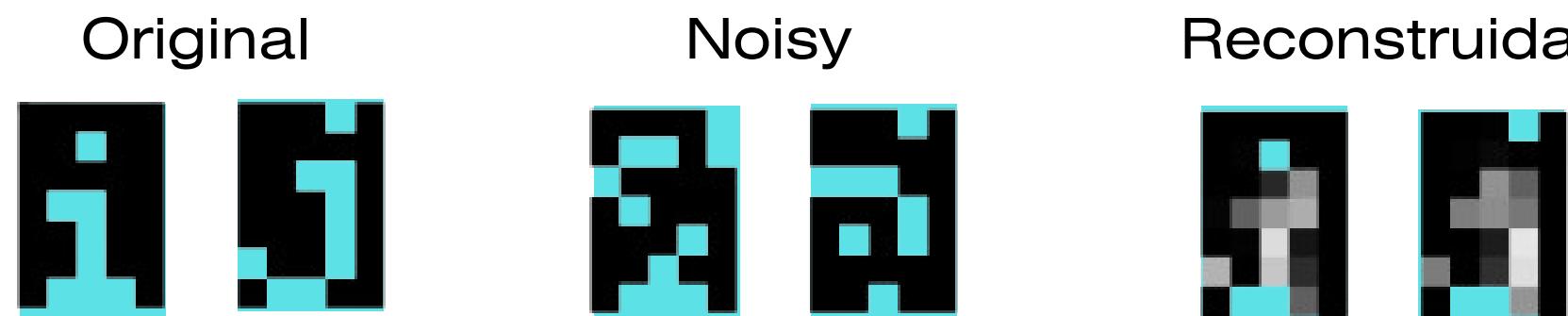


Denoising Autoencoder

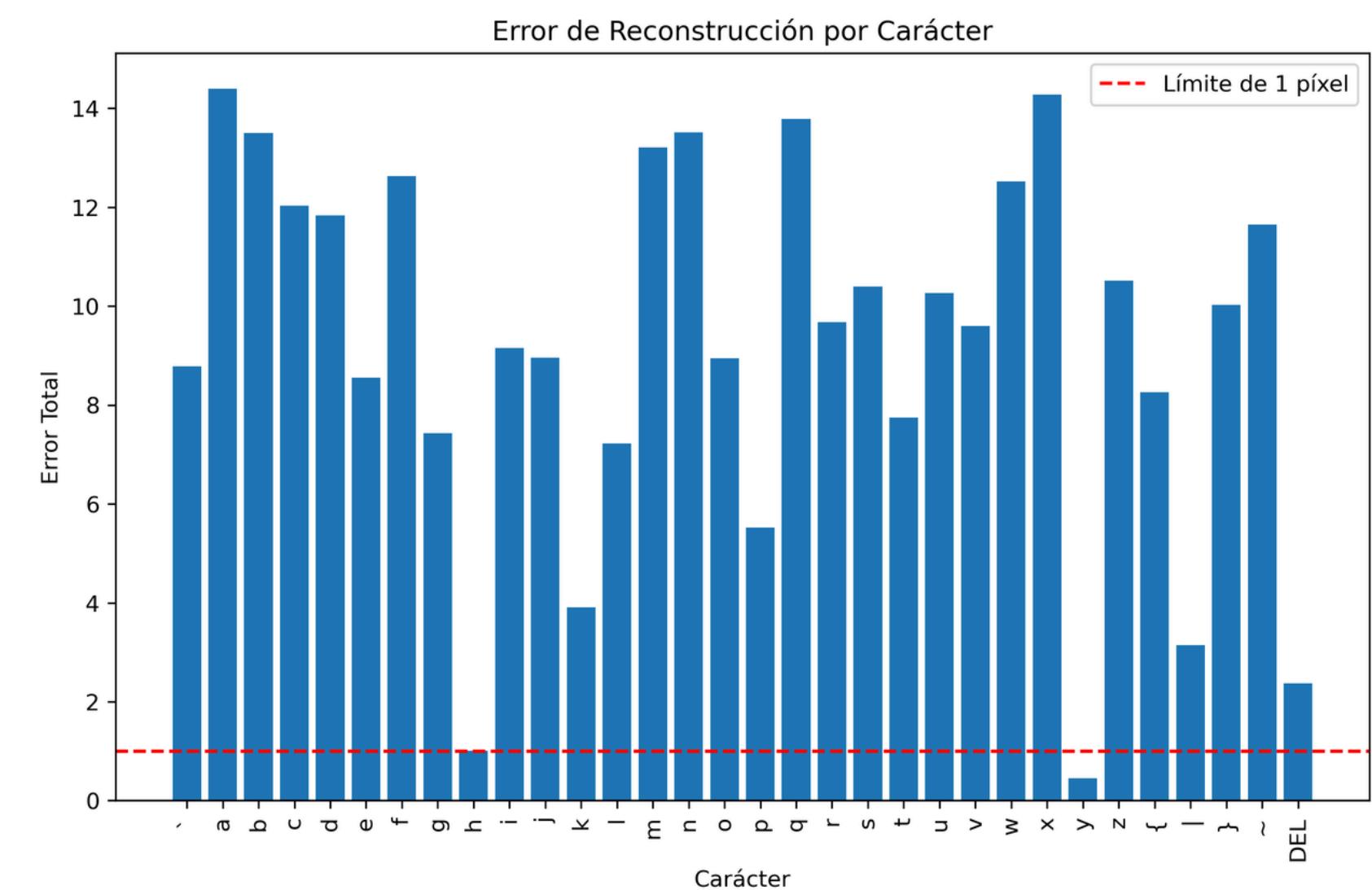


Ejercicio 1B

50% Salt and Pepper



90% Salt and Pepper





Conclusiones ejercicio 1

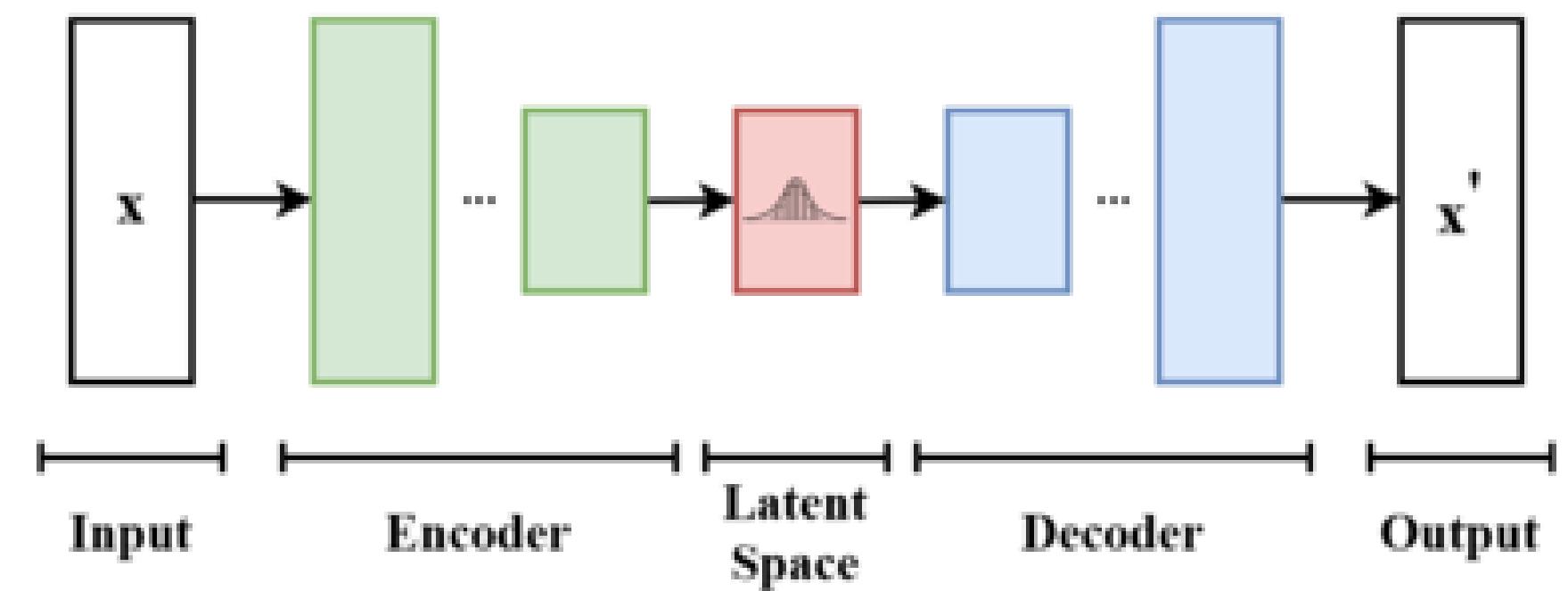
La arquitectura del ejercicio 1A no pudo manejar niveles de ruido (20%, 50% y 90%) eficazmente debido a su capacidad limitada para aprender representaciones complejas, lo que resultó en errores superiores a 1 píxel.

Más capas mejoraron la reconstrucción en niveles de ruido moderado (20% y 50%), ya que tiene mayor capacidad de modelado y puede aprender representaciones más complejas y filtrar mejor el ruido.

Redes más complejas aumentan la capacidad del modelo para aprender características complejas, lo que les permite manejar mejor el ruido moderado, pero **el ruido extremo (90%) sigue siendo un desafío** incluso para arquitecturas grandes.

Ejercicio 2

Variational Autoencoder



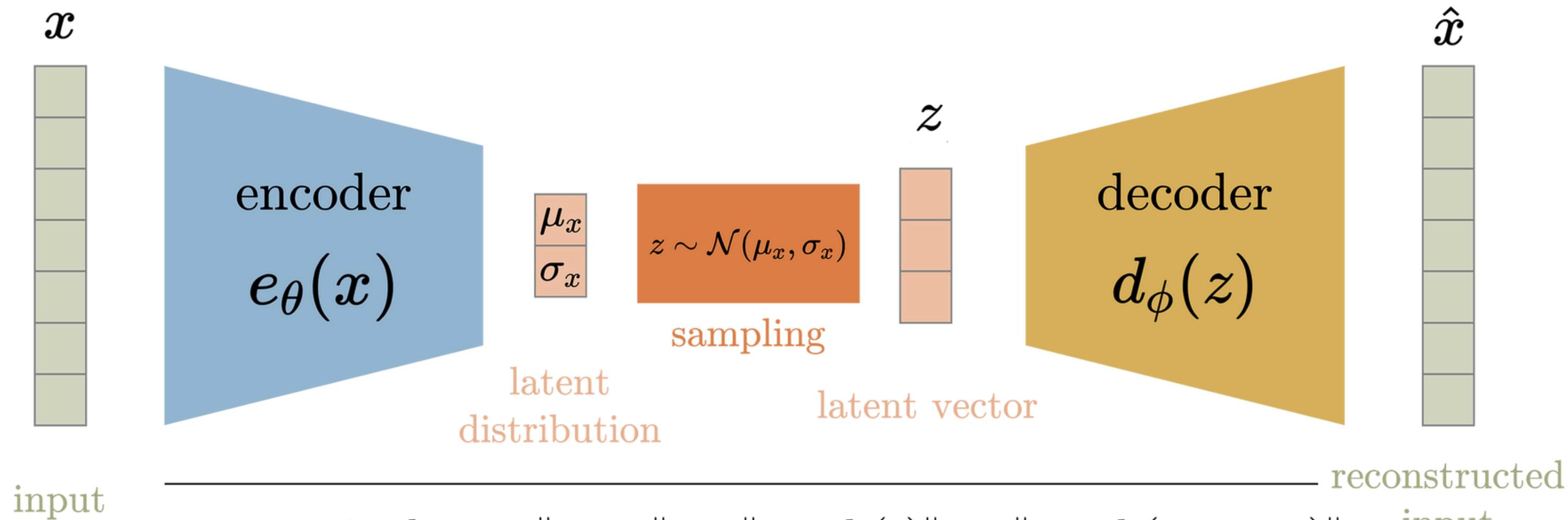
VAE

↑ Ejercicio 2

Autoencoder vs VAE

	Autoencoder	VAE
Representación del espacio latente	determinista	probabilístico, aproximando una distribución Gaussiana
Variabilidad en la salida	deterministas y fijas para la misma entrada	varían ligeramente debido al muestreo en el espacio latente
Capacidad Generativa	Modelo generativo deficiente	Diseñado para generación
Aplicaciones	Enfocado en reducción de dimensionalidad y tareas de reconstrucción	Usado para tareas generativas, aumento de datos y aprendizaje no supervisado

Autoencoder vs VAE



input

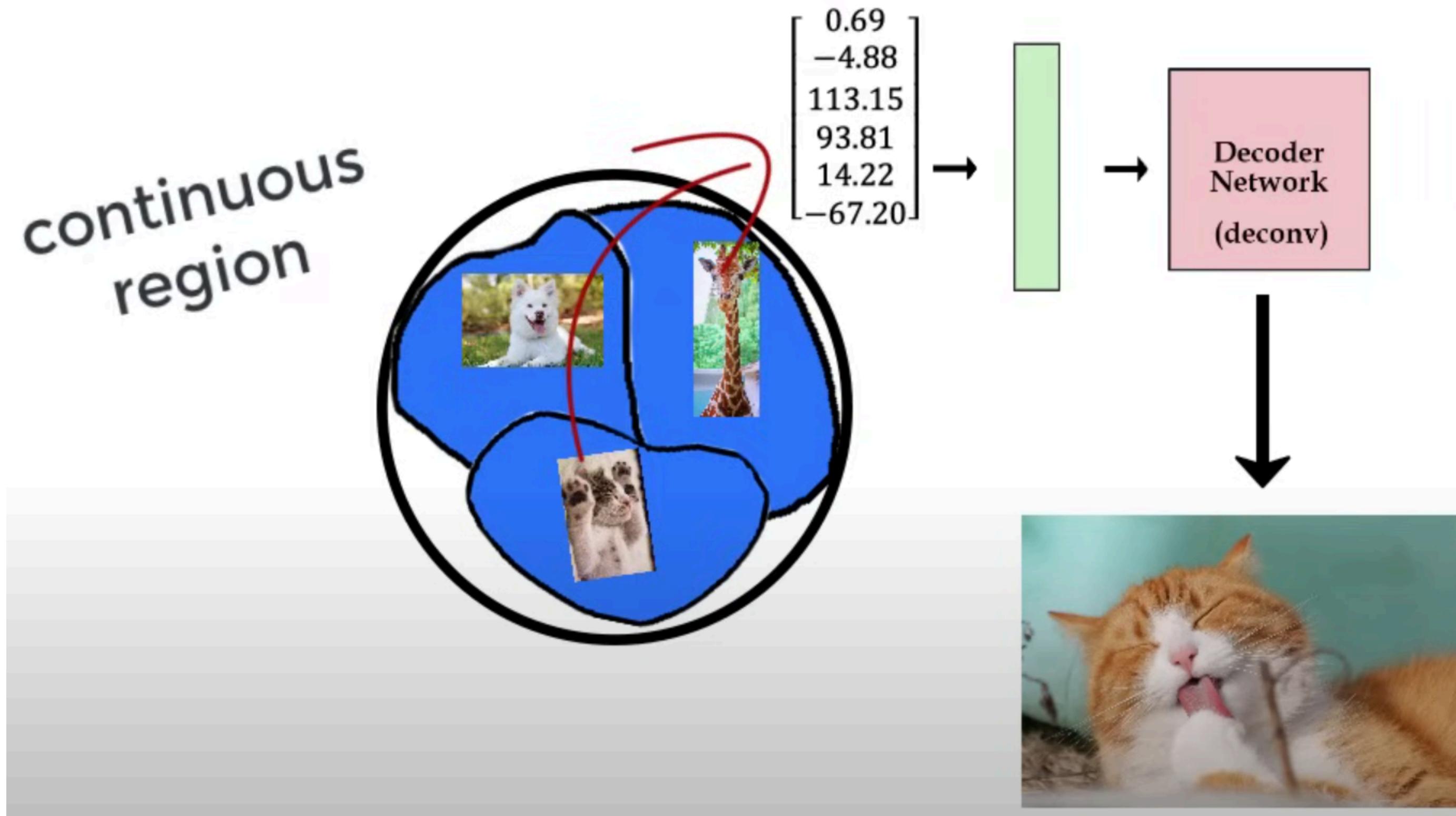
$$\text{reconstruction loss} = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(\mu_x + \sigma_x \epsilon)\|_2$$

$$\mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\text{similarity loss} = \text{KL Divergence} = D_{KL}(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}))$$

$$\text{loss} = \text{reconstruction loss} + \text{similarity loss}$$

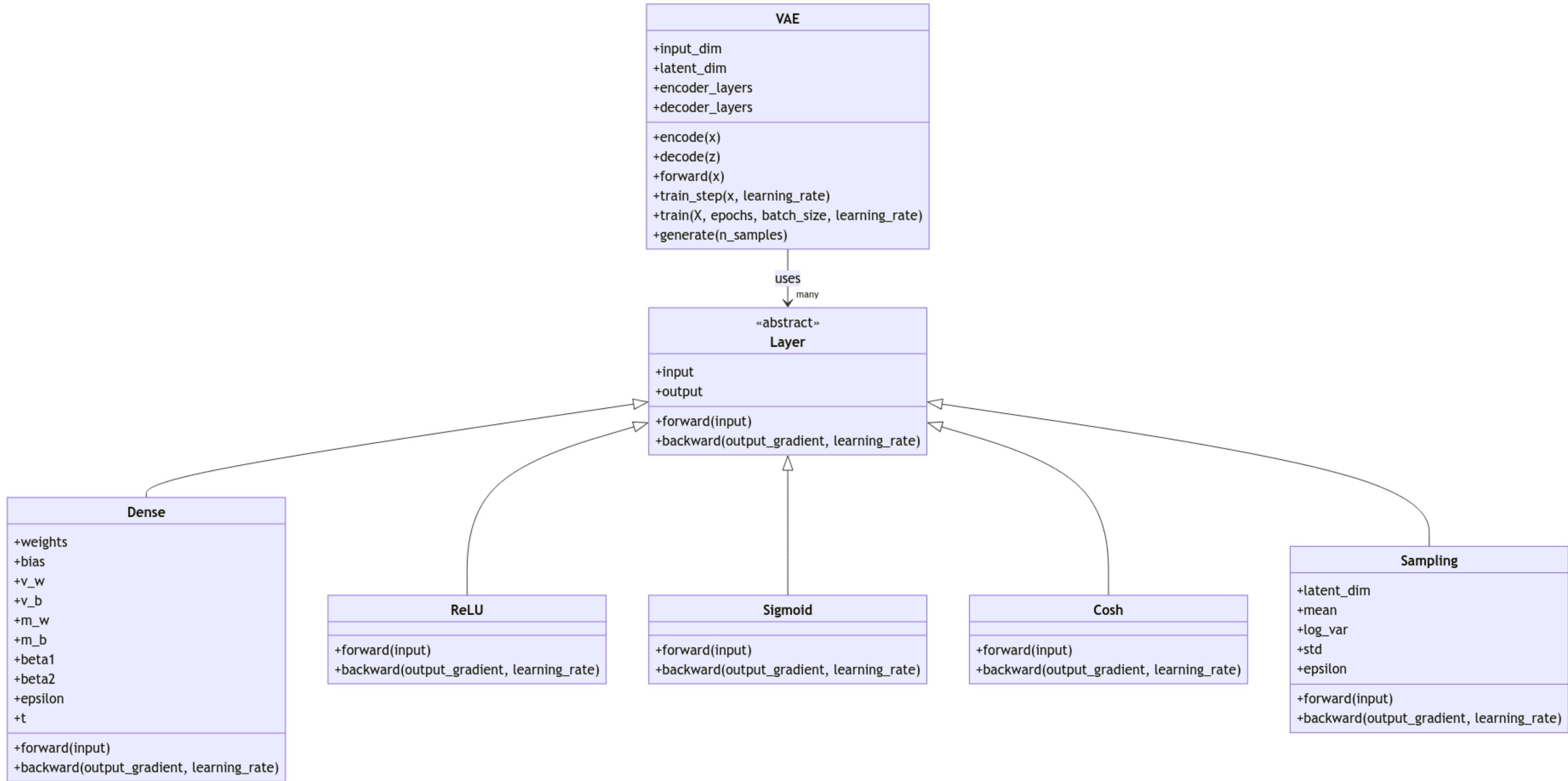
Sampling for data generation



VAE

↑ Ejercicio 2

Implementación VAE



VAE

↑ Ejercicio 2

Dataset - 64x64 pixels



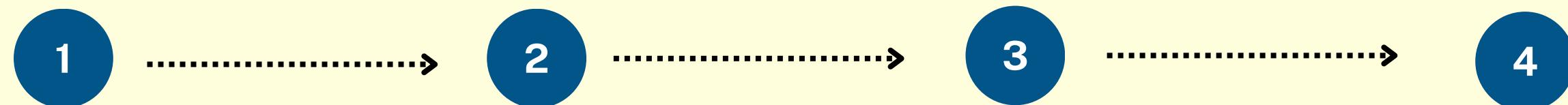
↑ Ejercicio 2





Ejercicio 02

¿Qué *experimentos* decidimos realizar?



*Variación de
Espacio Latente*

1
64
128
4

*Variación de
Learning Rate*

0.0001
0.01

*Variación de
Arquitectura*

shallow
deep

*Variación de F.
de Activación*

Relu
Cosh
Sigmoid



¿Podremos
configurar una
“mala
combinación?”



Ejercicio 02

¿Qué *vamos a mostrar*?

1

Reconstrucción
(reconstrucion.png)

2

Loss (loss.png)

3

Traversal del Espacio
Latente
(latent_traversal.png)

Variación de Espacio Latente

1

1

64

128

4

```
{  
    "data": {  
        "input_size": [64, 64],  
        "channels": 3,  
        "batch_size": 2,  
        "validation_split": 0.2  
    },  
    "model": {  
        "latent_dim": 4,  
        "encoder_layers": [  
            {"units": 256, "activation": "relu"},  
            {"units": 128, "activation": "relu"}  
        ],  
        "decoder_layers": [  
            {"units": 128, "activation": "relu"},  
            {"units": 256, "activation": "relu"}  
        ]  
    },  
    "training": {  
        "epochs": 200,  
        "learning_rate": 0.001,  
        "batch_size": 2  
    },  
    "paths": {  
        "data_dir": "input/emojis/",  
        "save_dir": "output/models/",  
        "log_dir": "logs/"  
    }  
}
```

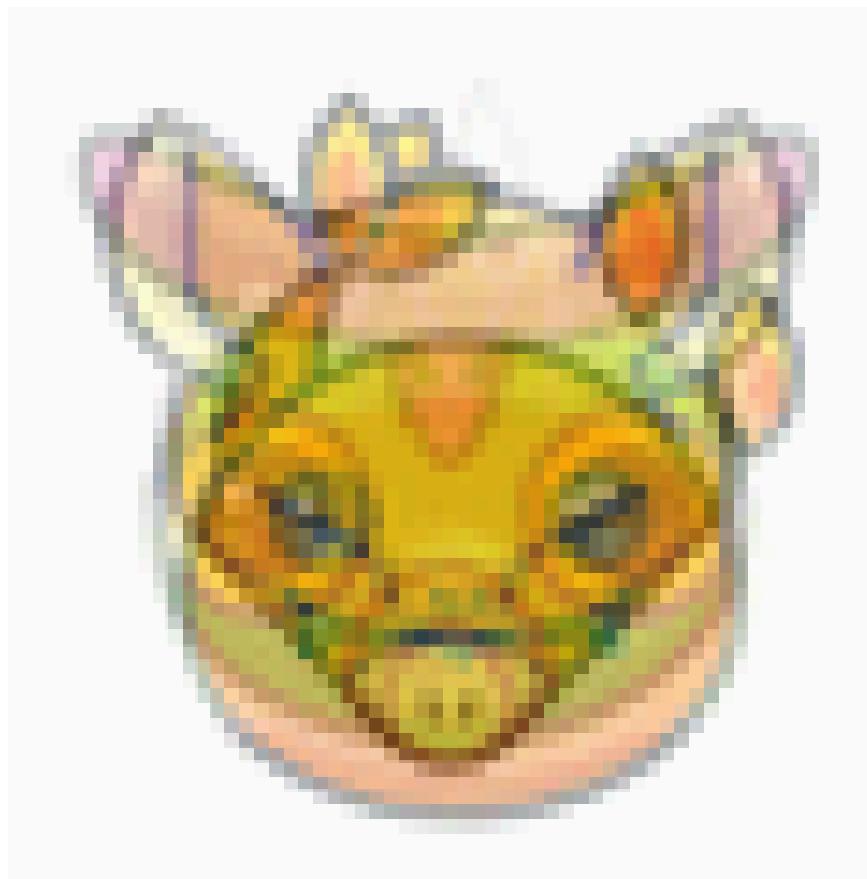


Latent space dim=1

Original



Reconstruida



- **Sobre-simplificación:** Una sola dimensión latente no puede capturar la complejidad de los datos de entrada de alta dimensión.
- **Alta Pérdida de Reconstrucción:** El decoder recibe información insuficiente, haciendo imposible reconstruir con precisión.

VAE

↑ Ejercicio 2

Latent space dim=64

Original



Reconstruida



Latent space dim=128

Original



Reconstruida



Dimensiones altas

- **Sobreajuste con dimensiones altas:** El modelo prioriza la memorización sobre la generalización, degradando la estructura del espacio latente.
- **Dispersión del espacio latente:** Un espacio latente de alta dimensión se vuelve disperso, ya que muchas dimensiones no se utilizan completamente. Los puntos codificados se agrupan en regiones pequeñas, dejando el resto del espacio mal representado.



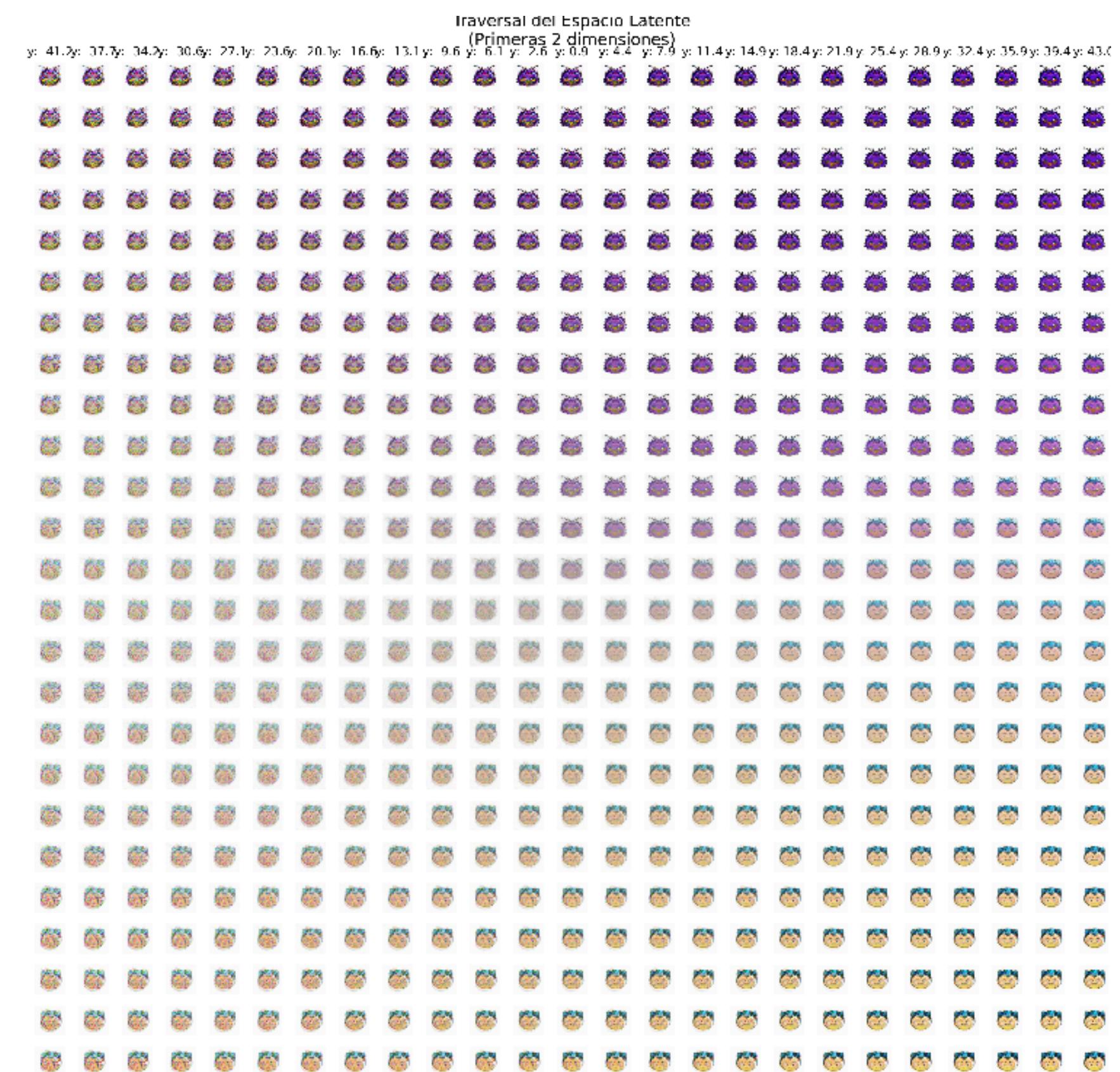
↑ Ejercicio 2

Latent space dim=4

Original

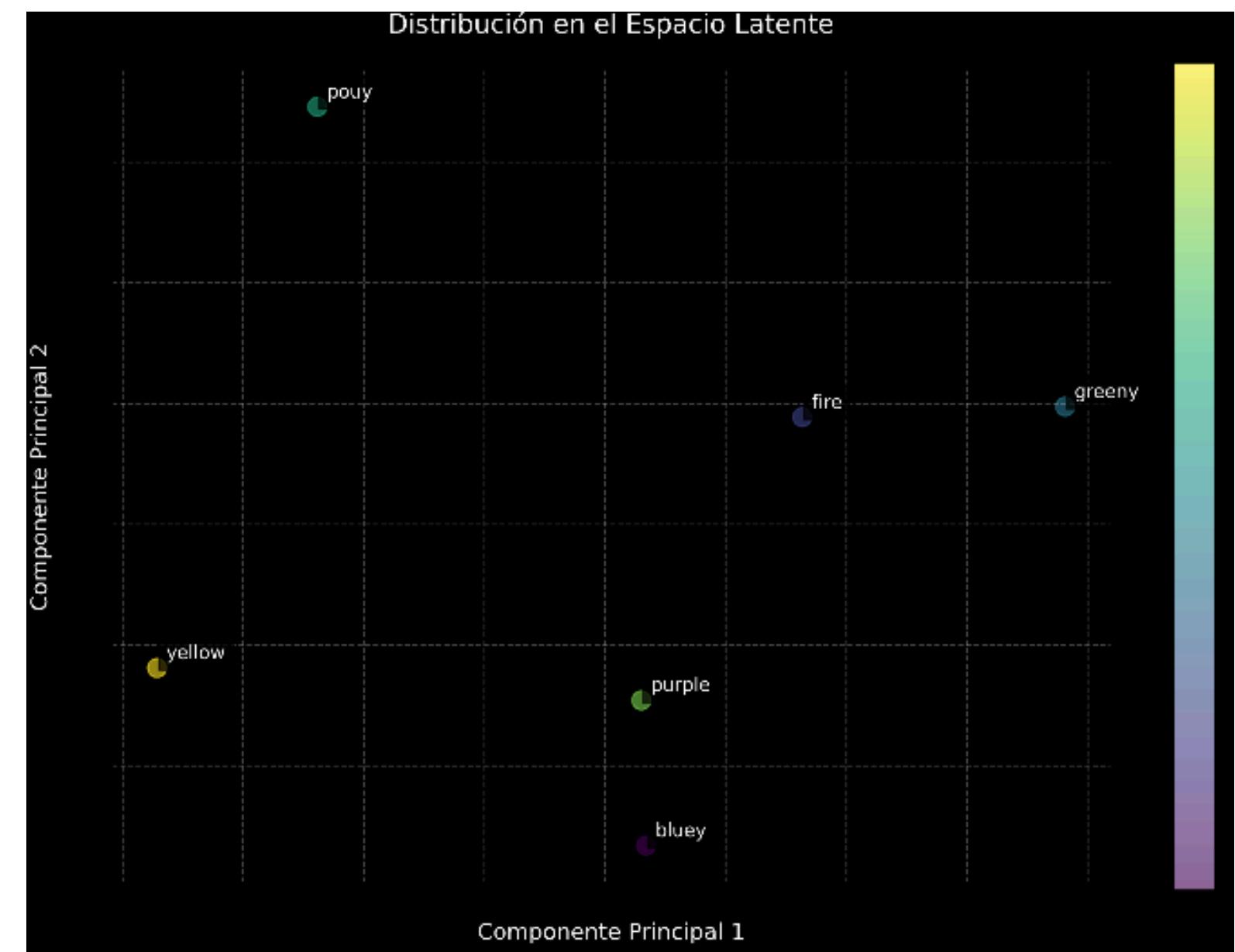
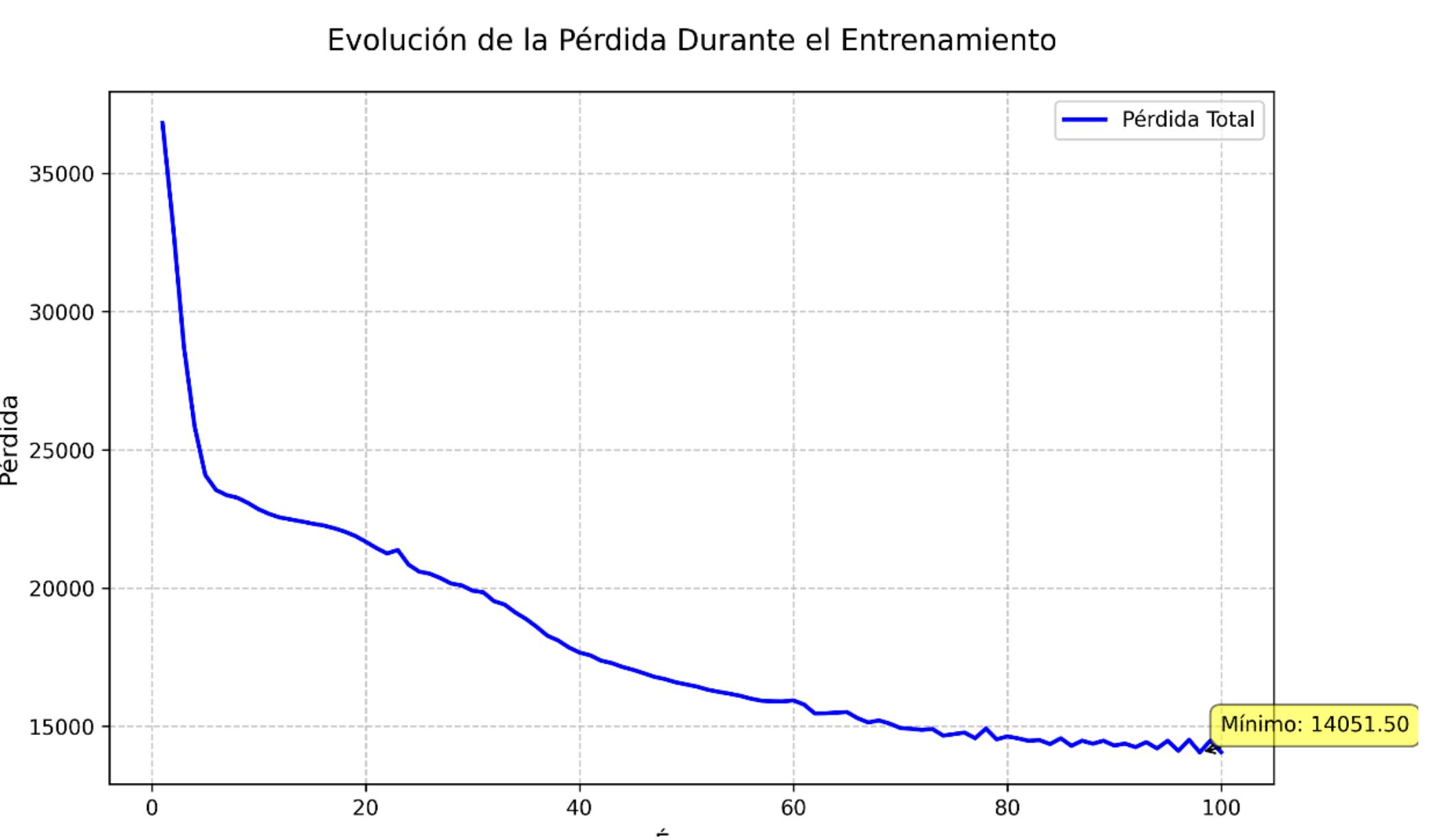


Reconstruida



↑ Ejercicio 2

Latent space dim=2



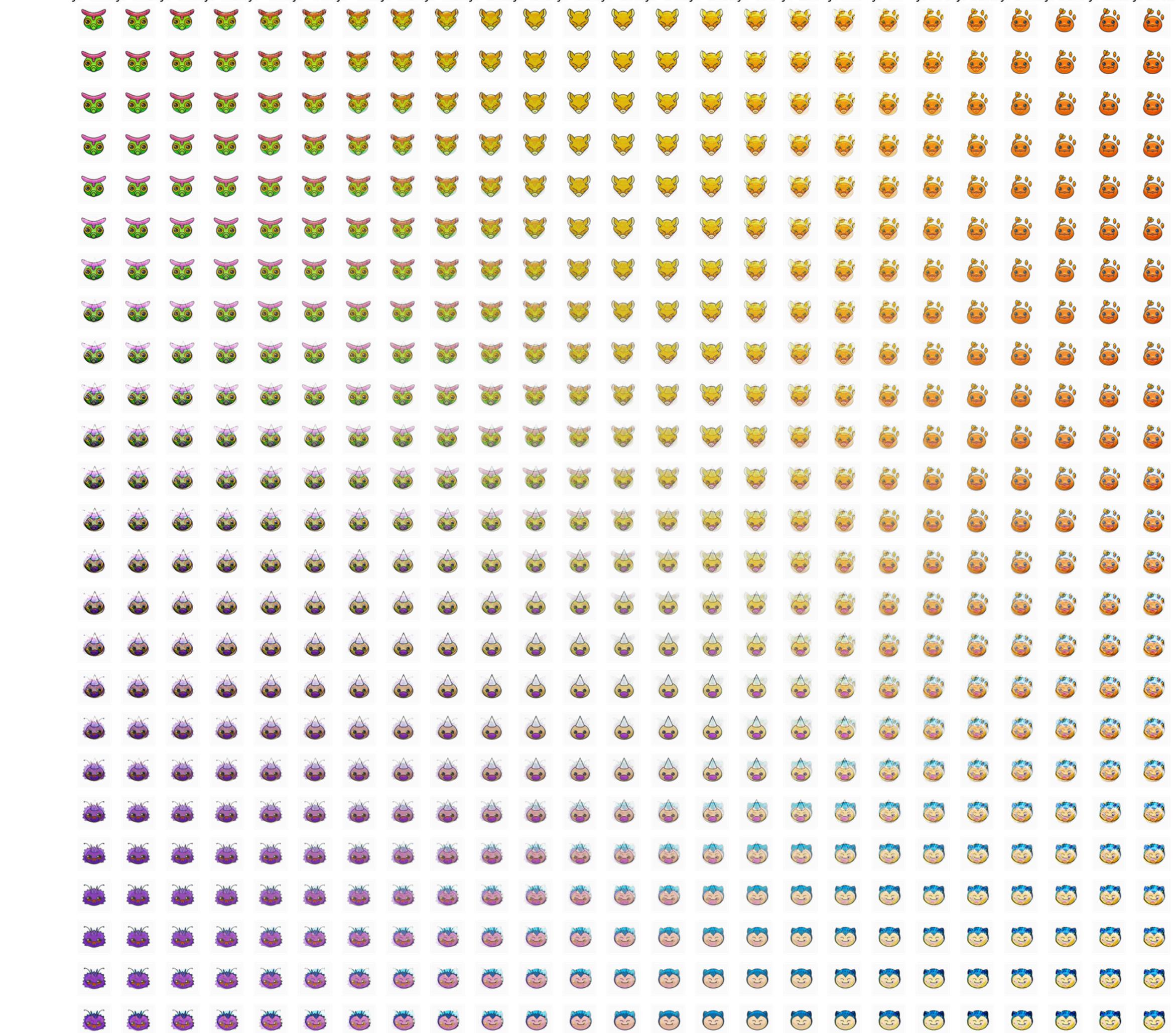
VAE

Traverso del Espacio Latente
(Primeras 2 dimensiones)

y: -63.0 y: -59.3 y: -55.5 y: -51.7 y: -48.0 y: -44.2 y: -40.5 y: -36.7 y: -32.9 y: -29.2 y: -25.4 y: -21.7 y: -17.9 y: -14.1 y: -10.4 y: -6.6 y: -2.9 y: 0.9 y: 4.7 y: 8.4 y: 12.2 y: 15.9 y: 19.7 y: 23.5 y: 27.2



Ejercicio 2



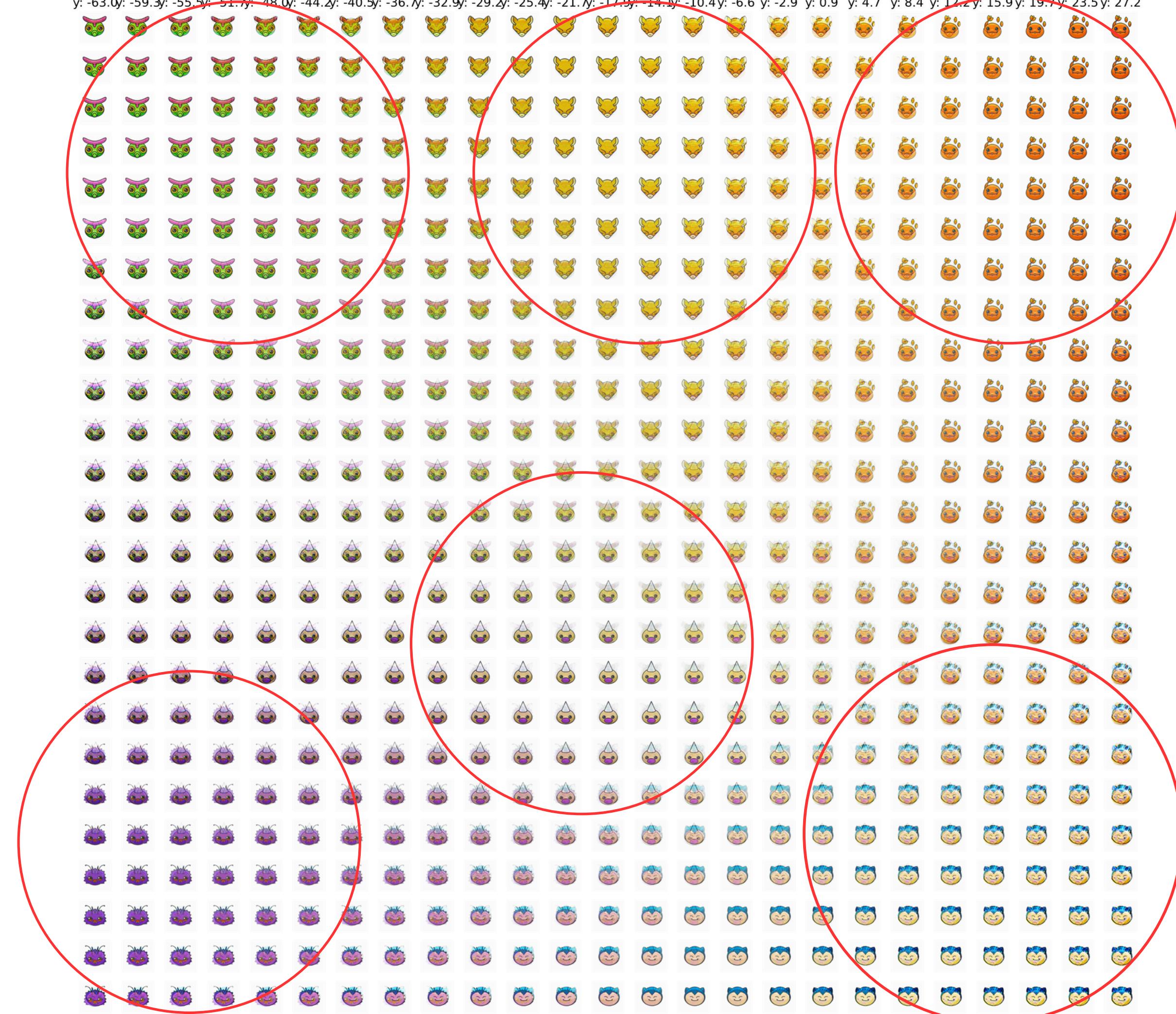
VAE



Ejercicio 2

Traverso del Espacio Latente
(Primeras 2 dimensiones)

y: -63.0: -59.3y: -55.5y: -51.7y: -48.0y: -44.2y: -40.5y: -36.7y: -32.9y: -29.2y: -25.4y: -21.7y: -17.9y: -14.1y: -10.4y: -6.6 y: -2.9 y: 0.9 y: 4.7 y: 8.4 y: 12.2y: 15.9y: 19.7y: 23.5y: 27.2



2

Variación de Learning Rate

0.0001
0.01

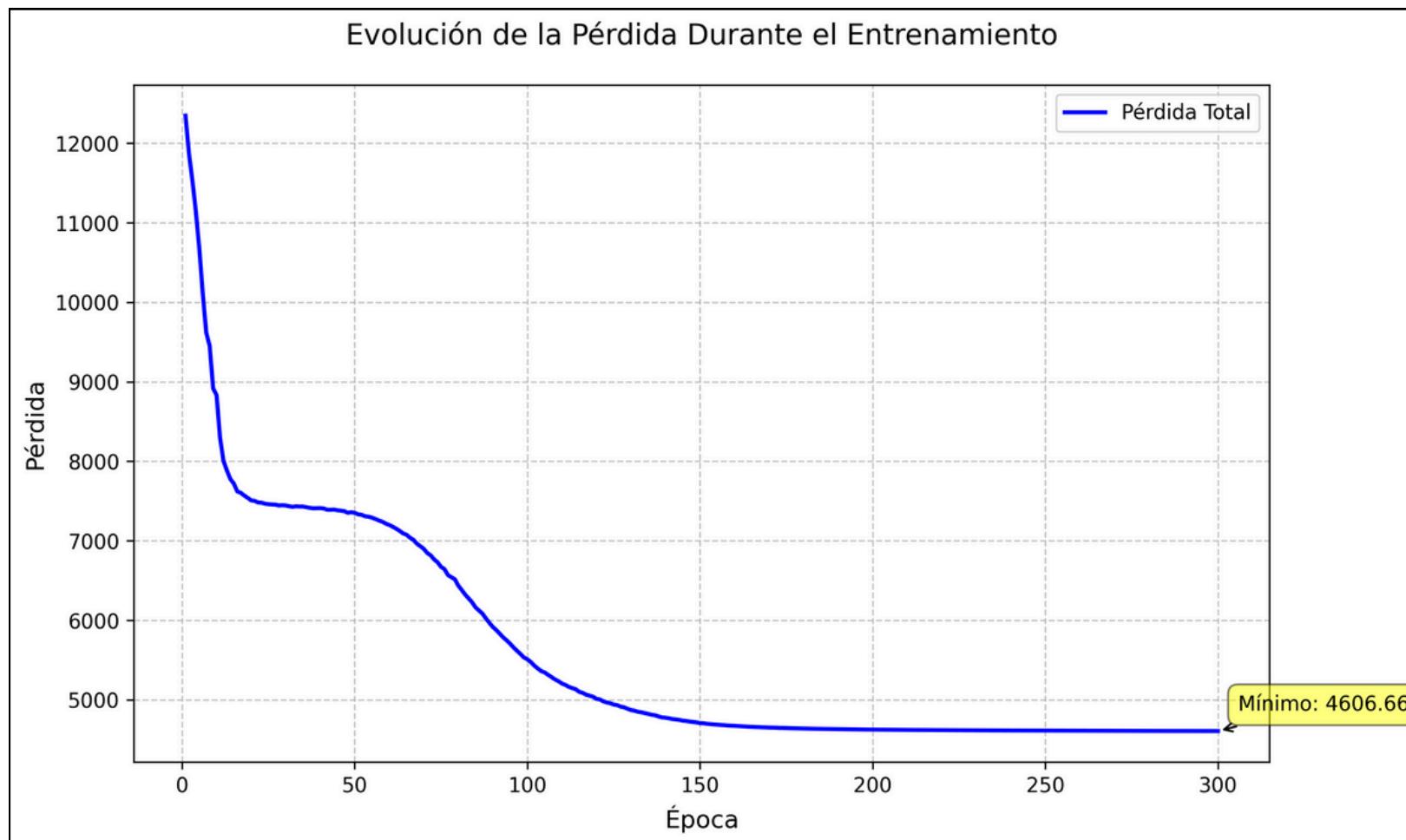


```
{
  "data": {
    "input_size": [64, 64],
    "channels": 3,
    "batch_size": 2,
    "validation_split": 0.2
  },
  "model": {
    "latent_dim": 16,
    "encoder_layers": [
      {"units": 256, "activation": "relu"},
      {"units": 128, "activation": "relu"}
    ],
    "decoder_layers": [
      {"units": 128, "activation": "relu"},
      {"units": 256, "activation": "relu"}
    ]
  },
  "training": {
    "epochs": 100,
    "learning_rate": 0.0001,
    "batch_size": 2
  },
  "paths": {
    "data_dir": "input/emojis/",
    "save_dir": "output/models/",
    "log_dir": "logs/"
  }
}
```

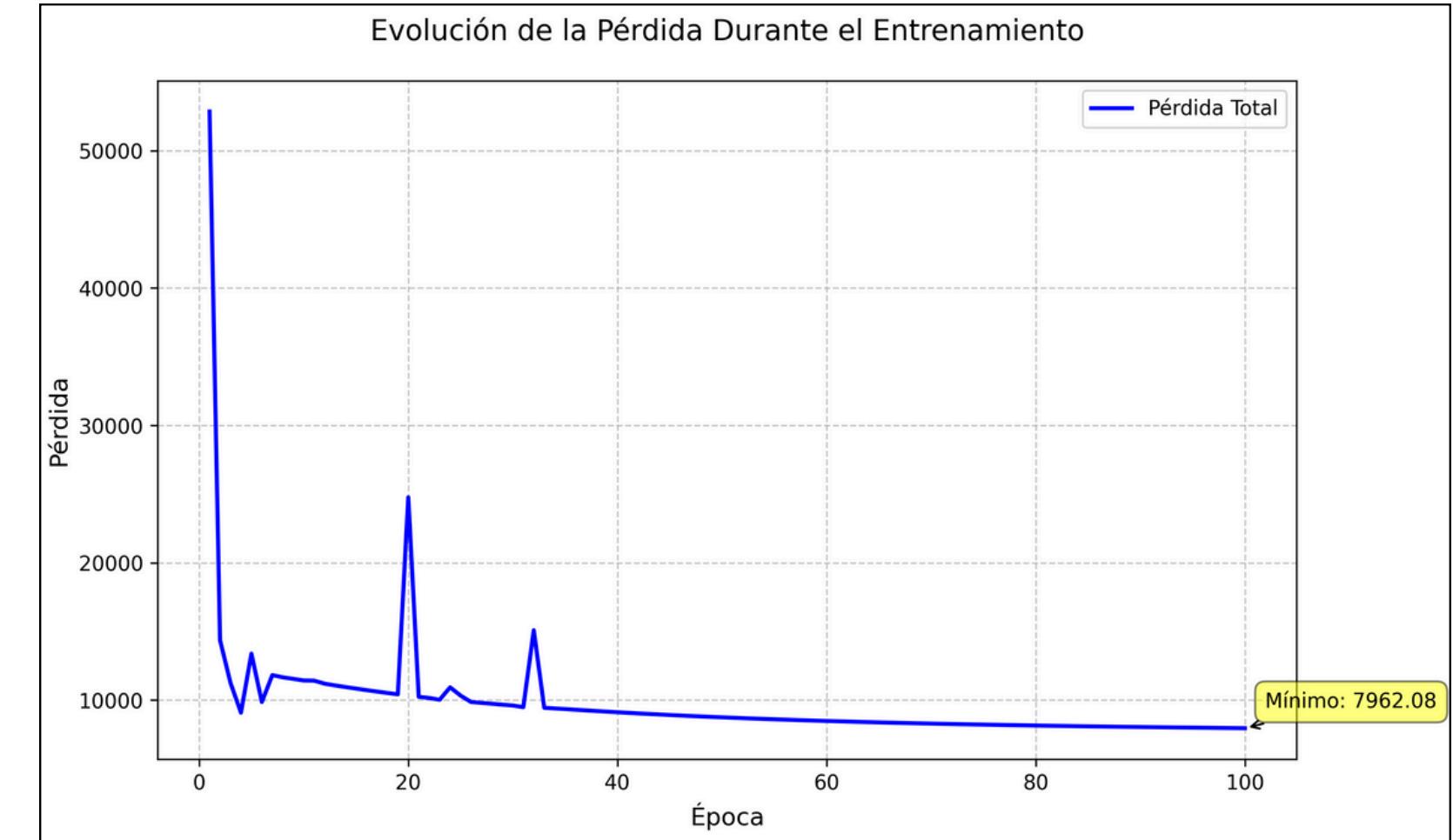


Ejercicio 2

Learning Rate 0.0001



Learning Rate 0.01



- Aprendizaje muy cauteloso
- Convergencia muy lenta
- Curva de loss suave pero con progreso lento
- Más estable pero puede necesitar muchas épocas.

- Aprendizaje muy agresivo
- Convergencia rápida o divergencia
- Curva de pérdida con oscilaciones
- Posible inestabilidad en el entrenamiento

↑ Ejercicio 2

Learning Rate 0.0001



Learning Rate 0.01



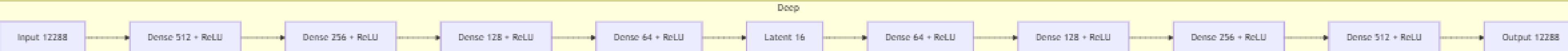
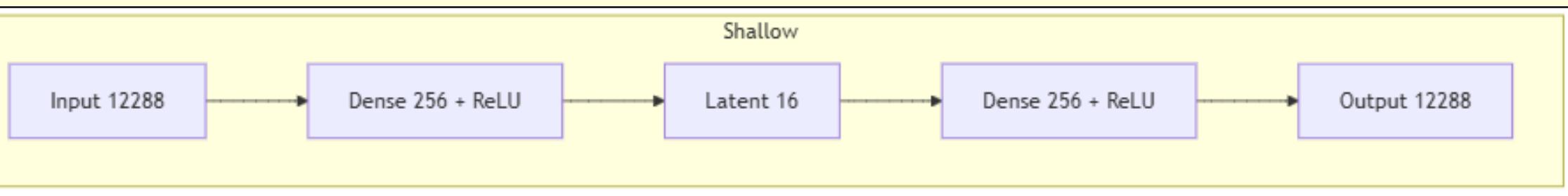
- Aprendizaje muy cauteloso
- Convergencia muy lenta
- Curva de loss suave pero con progreso lento
- Más estable pero puede necesitar muchas épocas.

- Aprendizaje muy agresivo
- Convergencia rápida o divergencia
- Curva de pérdida con oscilaciones
- Posible inestabilidad en el entrenamiento

3

Variación de Arquitectura

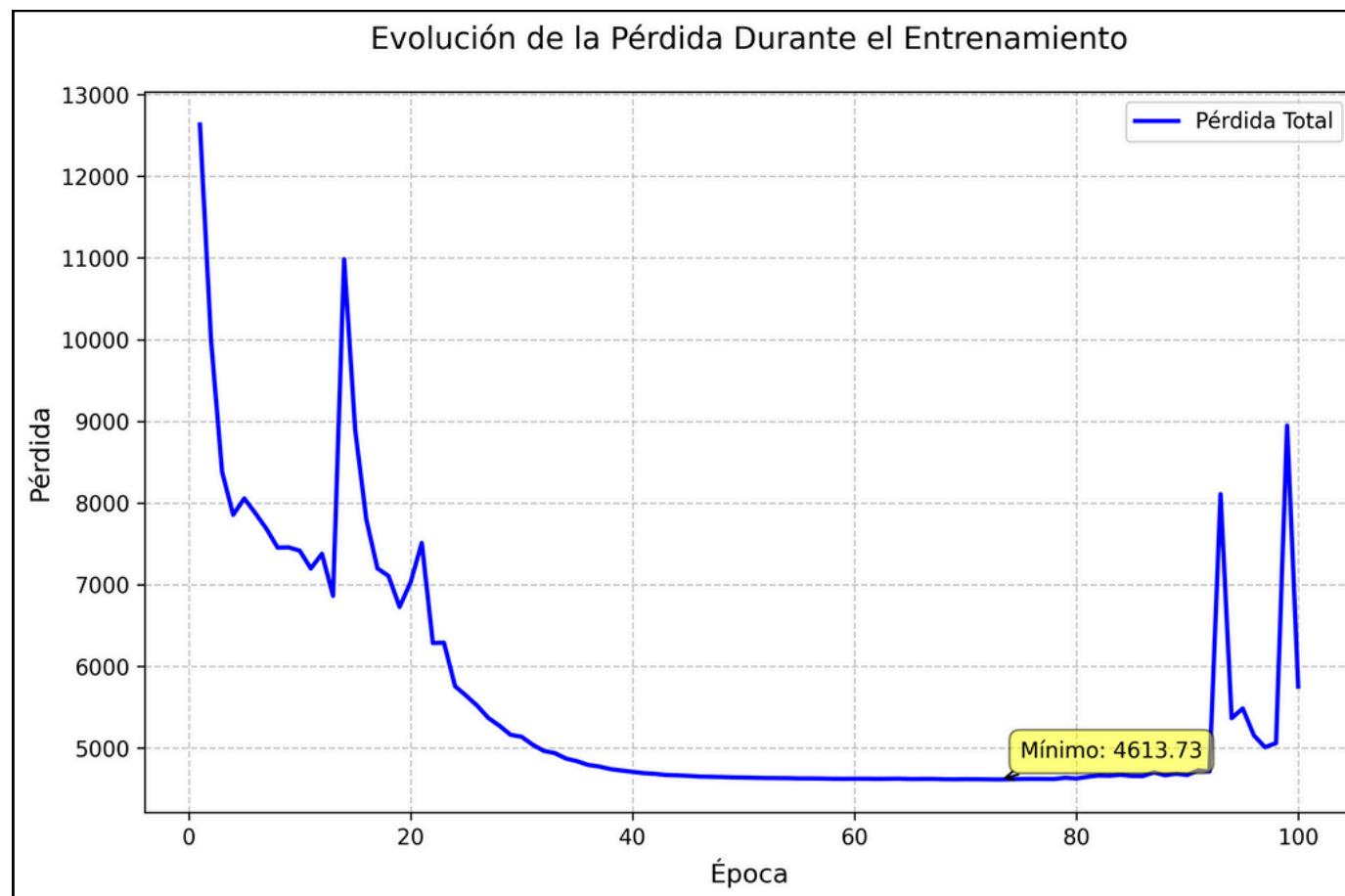
Shallow
Deep



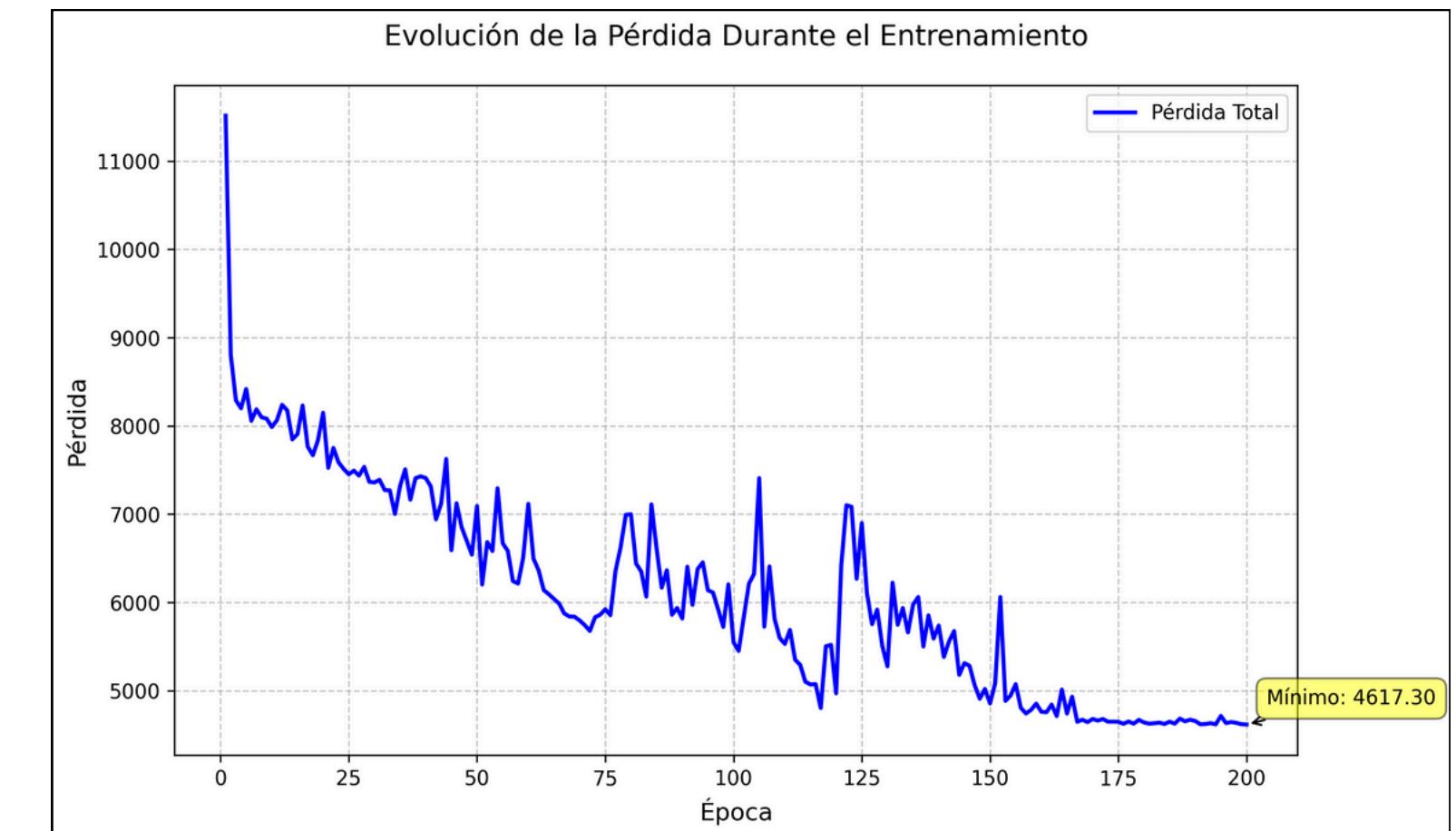
```
[{"data": {"input_size": [64, 64], "channels": 3, "batch_size": 2, "validation_split": 0.2}, "model": {"latent_dim": 16, "encoder_layers": [{"units": 256, "activation": "relu"}], "decoder_layers": [{"units": 256, "activation": "relu"}]}, "training": {"epochs": 200, "learning_rate": 0.001, "batch_size": 2}, "paths": {"data_dir": "input/emojis/", "save_dir": "output/models/", "log_dir": "logs/"}}]
```

↑ Ejercicio 2

Shallow



Deep



- Red simple
- Menos capacidad de aprendizaje
- Reconstrucciones más generales/básicas
- Entrenamiento más rápido pero resultados limitados

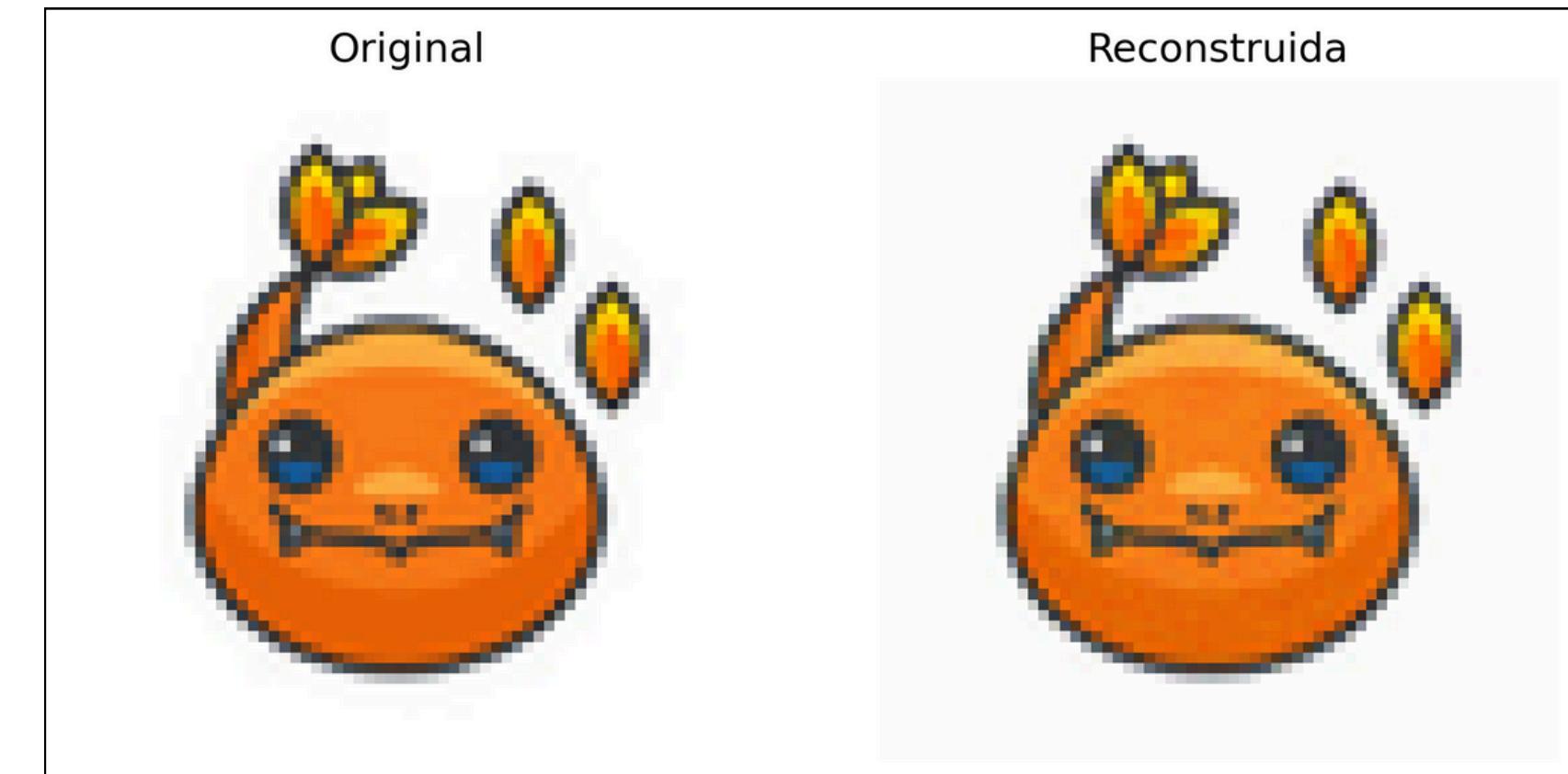
- Red más compleja
- Mayor capacidad de aprendizaje
- Reconstrucciones más detalladas
- Entrenamiento más lento pero mejores resultados

↑ Ejercicio 2

Shallow



Deep



- Red simple
- Menos capacidad de aprendizaje
- Reconstrucciones más generales/básicas
- Entrenamiento más rápido pero resultados limitados

- Red más compleja
- Mayor capacidad de aprendizaje
- Reconstrucciones más detalladas
- Entrenamiento más lento pero mejores resultados

Variación de Función de Activación

Relu

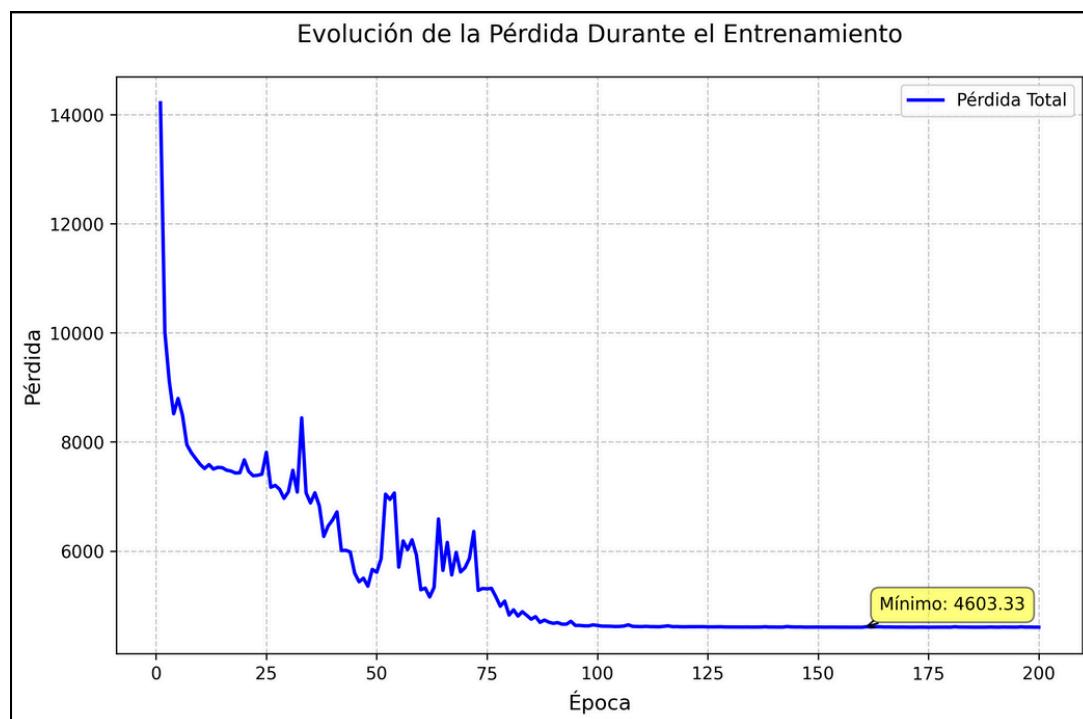
Cosh

Sigmoid

```
{  
    "data": {  
        "input_size": [64, 64],  
        "channels": 3,  
        "batch_size": 2,  
        "validation_split": 0.2  
    },  
    "model": {  
        "latent_dim": 16,  
        "encoder_layers": [  
            {"units": 256, "activation": "relu"},  
            {"units": 128, "activation": "relu"},  
            {"units": 64, "activation": "relu"}  
        ],  
        "decoder_layers": [  
            {"units": 64, "activation": "relu"},  
            {"units": 128, "activation": "relu"},  
            {"units": 256, "activation": "relu"}  
        ]  
    },  
    "training": {  
        "epochs": 200,  
        "learning_rate": 0.001,  
        "batch_size": 2  
    },  
    "paths": {  
        "data_dir": "input/emojis/",  
        "save_dir": "output/models/",  
        "log_dir": "logs/"  
    }  
}
```

↑ Ejercicio 2

ReLU



Sigmoid



Cosh



- Mejor preservación de detalles finos
- Puede perder algunos detalles en regiones oscuras (por el corte en 0)
- Reconstrucciones más nítidas
- Convergencia más rápida al inicio
- Curva de pérdida suave al inicio y descendente

- Mejor manejo de transiciones suaves
- Puede perder contraste en los extremos
- Reconstrucciones más suavizadas/difuminadas
- Convergencia inicial lenta
- Curva de pérdida con plateaus (mesetas)

- Balance entre preservación de detalles y suavizado
- Mejor manejo de valores extremos que sigmoid
- Convergencia moderada
- Curva de pérdida más estable que ReLU



5

**¿Podremos configurar
una “mala combinación?**



¿Por qué esta configuración es mala?

Dimensión latente = 2

- Demasiado pequeña para capturar la complejidad de imágenes RGB
- Pérdida de detalles.

Learning rate muy alto (0.1)

- Causará oscilaciones grandes en el entrenamiento
- Probablemente nunca converja

Pocas épocas (50):

- Insuficiente para aprender con una arquitectura tan limitada

Arquitectura muy simple (32 unidades)

- Capacidad insuficiente para procesar imágenes 64x64x3
- No puede aprender características importantes

Batch size = 6

- Igual al tamaño total del dataset
- No hay aleatoriedad en el entrenamiento



Ejercicio 2

¿Por qué esta configuración es mala?

Original

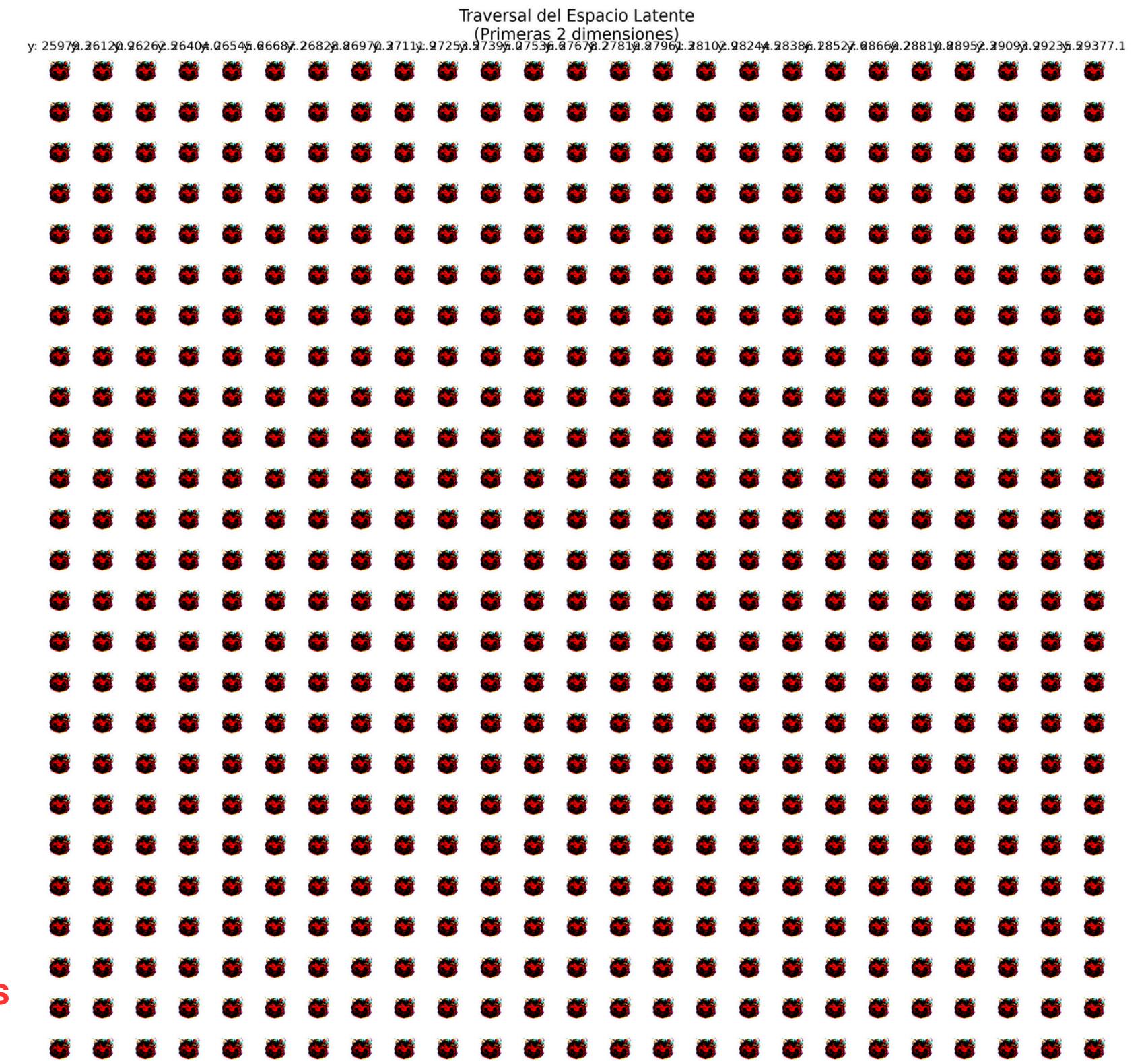


Reconstruida



- Los que no converge o fluctúa muchísimo
- Reconstrucciones irreconocibles
- Espacio latente sin estructura clara
- Posiblemente "modo colapso" donde todas las reconstrucciones son similares

La reconstrucción depende de manera significativa en una elección precisa de los hiperámetros.



Conclusiones



Ventajas de los VAEs

- **Capacidad Generativa:** Los VAEs pueden generar nuevos puntos de datos muestreando el espacio latente, mientras que los autoencoders tradicionales no.



Límites de los VAEs

- **Realismo limitado:** Los VAEs tienen dificultades para generar salidas altamente realistas en comparación con los GANs, ya que su marco probabilístico sacrifica detalles por suavidad.
- **Dependencia de los datos de entrenamiento:** Los datos generados están limitados a variaciones dentro de la distribución del conjunto de entrenamiento; no pueden crear muestras verdaderamente nuevas o fuera de distribución.



¿Preguntas?

¡Gracias por la atención!

