DEEP LEARNING

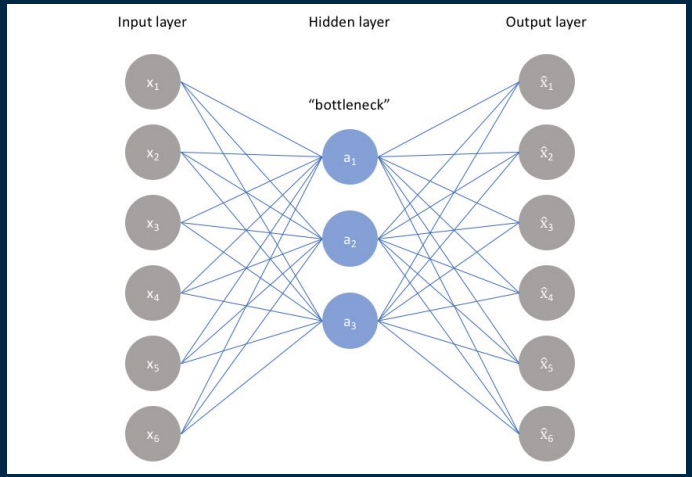
Julián Sicardi
Juan Ignacio Quintairos
Salustiano Zavalía

INTRODUCCIÓN

01

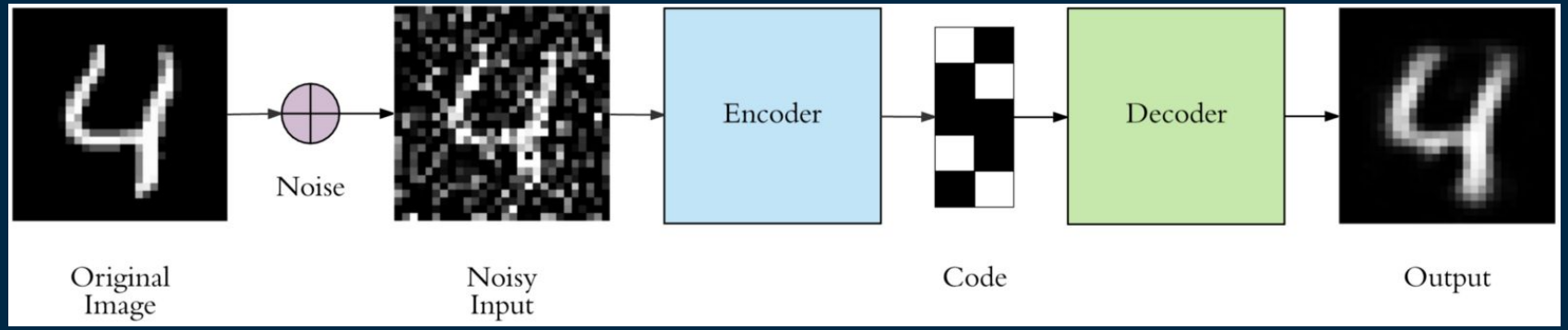
DEEP LEARNING Y AUTOENCODERS

- Def. Deep Learning: Campo de *machine learning* dedicado a algoritmos de redes neuronales artificiales. Hace énfasis en **redes neuronales grandes** que acceden a **inmensos volúmenes de datos**.
- Def. Autoencoder: Arquitectura de redes neuronales no supervisadas orientadas a la reducción de la dimensionalidad de los datos.



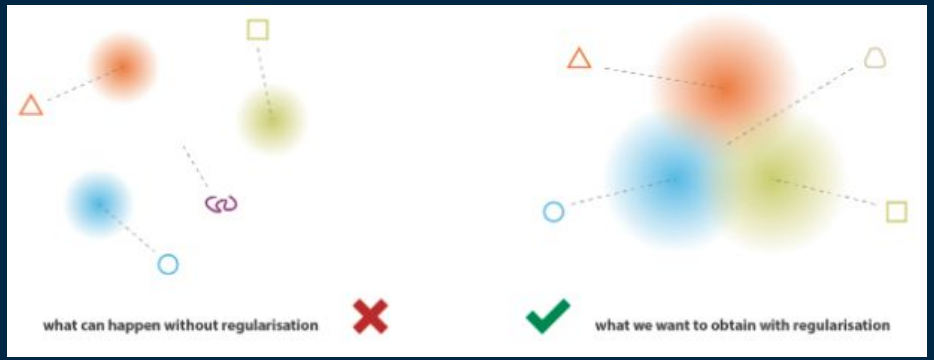
DENOISING AUTOENCODERS

- Nos ayudan a eliminar el ruido mediante la preservación de la información más importante en la capa latente.
- Una vez aprendido el conjunto de datos, se introducen nuevas muestras contaminadas con ruido.
- La salida esperada de las entradas con ruido es el dato sin ruido. Así, la red aprende a suprimir el ruido en la entrada.



AUTOENCODERS VARIACIONALES

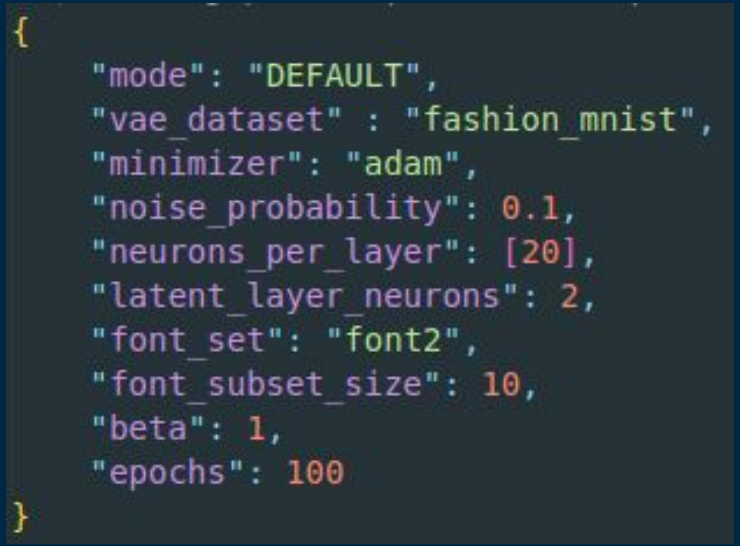
- Modelo de autoencoder generativo
- La distribución de codificaciones en el espacio latente está regularizada para asegurar:
  - Continuidad: dos puntos cercanos en el espacio latente no deben dar dos resultados completamente distintos luego de ser decodificados
  - Completitud: para una distribución dada, un punto del espacio latente debe dar contenido significativo luego de ser decodificado



IMPLEMENTACIÓN 02

CONFIGURACIÓN

- Mediante un archivo config.json
  - Mode: VAE, DAE ó DEFAULT
  - Vae dataset:
    - Fashion_mnist
    - Mnist
  - Font set: font1, font2 ó font3
  - Minimizer: adam ó powell



TÉCNICAS DE OPTIMIZACIÓN

- A la hora de entrenar a la red utilizamos:
  - El método de Powell de la librería SciPy
  - El método ADAM de Autograd
- Utilizamos operaciones matriciales siempre que fué posible.
- Empleamos distintas funciones de activación para distintas capas:
  - Las capas del *encoder* usan lineal.
  - La capa latente usa lineal.
  - Las capas del *decoder* usan la función logística.

RESULTADOS

03

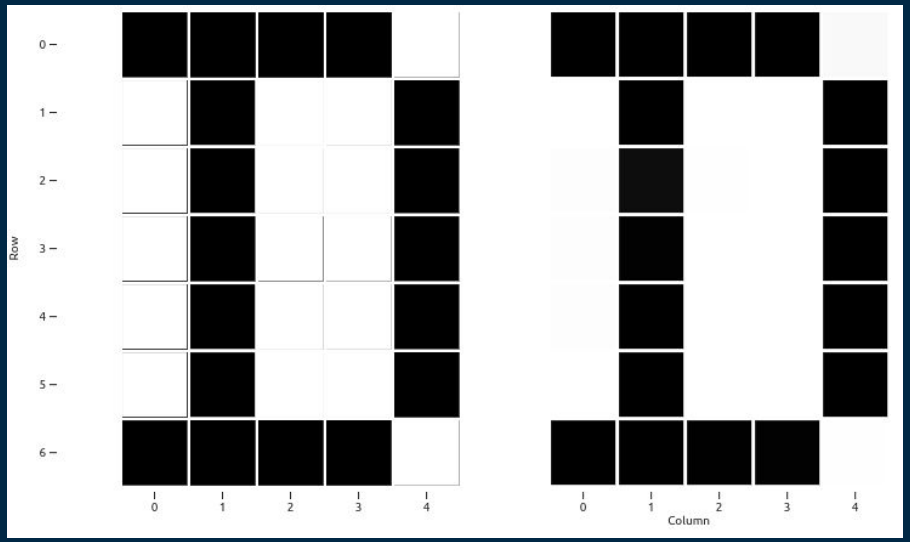
Gráficos en ObservableHQ

AUTOENCODER CLÁSICO

- El objetivo es encontrar una arquitectura que pueda aprender todos los patrones de entrada con un error aceptable
  - Consideraremos aceptable un error que nos permita identificar todas las letras luego de ser decodificadas.
- Queremos encontrar, también, una arquitectura que nos permita aprender la mayor cantidad de letras teniendo un cuello de botella de dos neuronas (capa latente):
  - Primero probamos con el conjunto completo
  - Luego intentamos con 10 letras

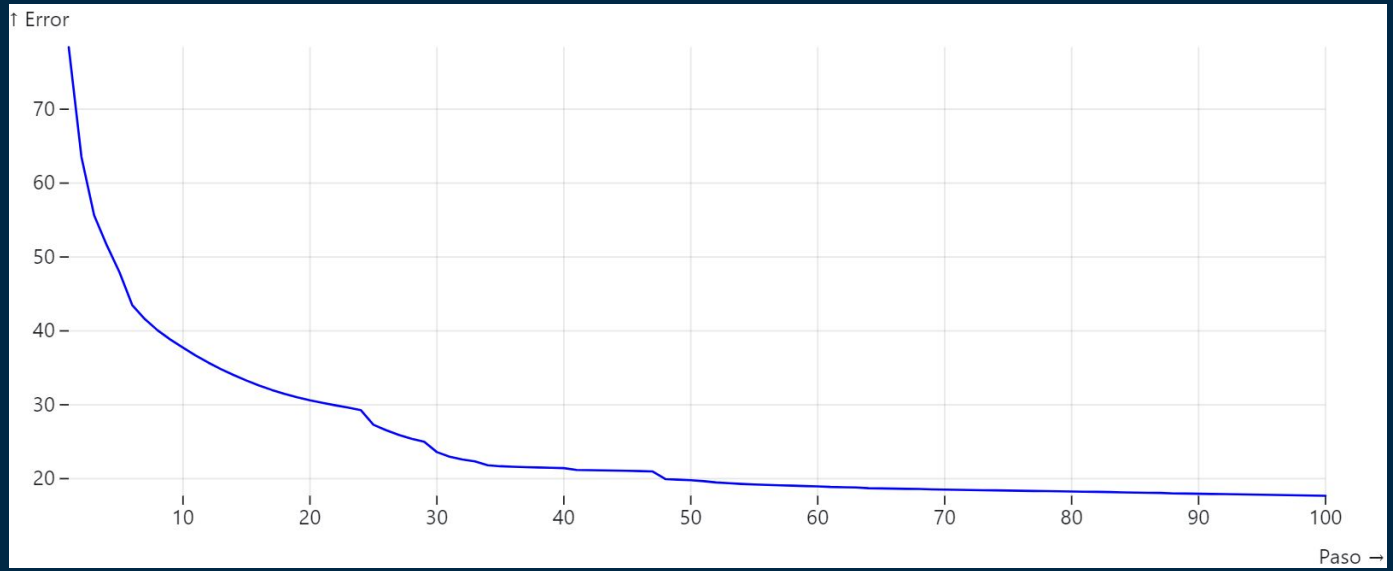
AUTOENCODER CLÁSICO

- Buscamos una arquitectura que nos permitiera aprender todas las letras de uno de los conjuntos
  - La arquitectura [35, 20, 10, 20, 35] funcionó, llegando a un error acumulado de 0.11 luego de 100 epochs

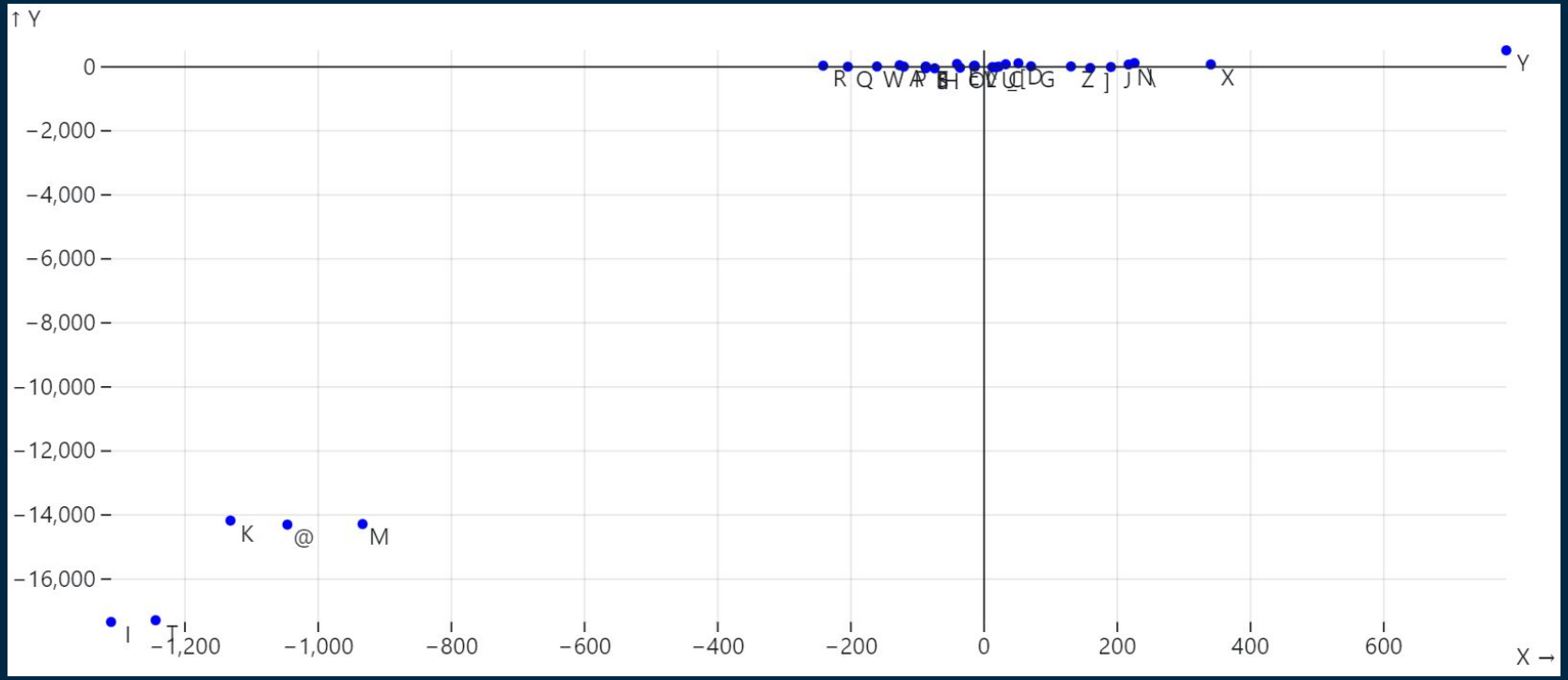


AUTOENCODER CLÁSICO

- No logramos encontrar una arquitectura que pudiera aprender todas las letras pero que al mismo tiempo logre reducir la dimensionalidad a 2.
  - La arquitectura [35, 20, 8, 2, 8, 20, 35] se acercó, con un error acumulado de 17.68 luego de 100 épocas

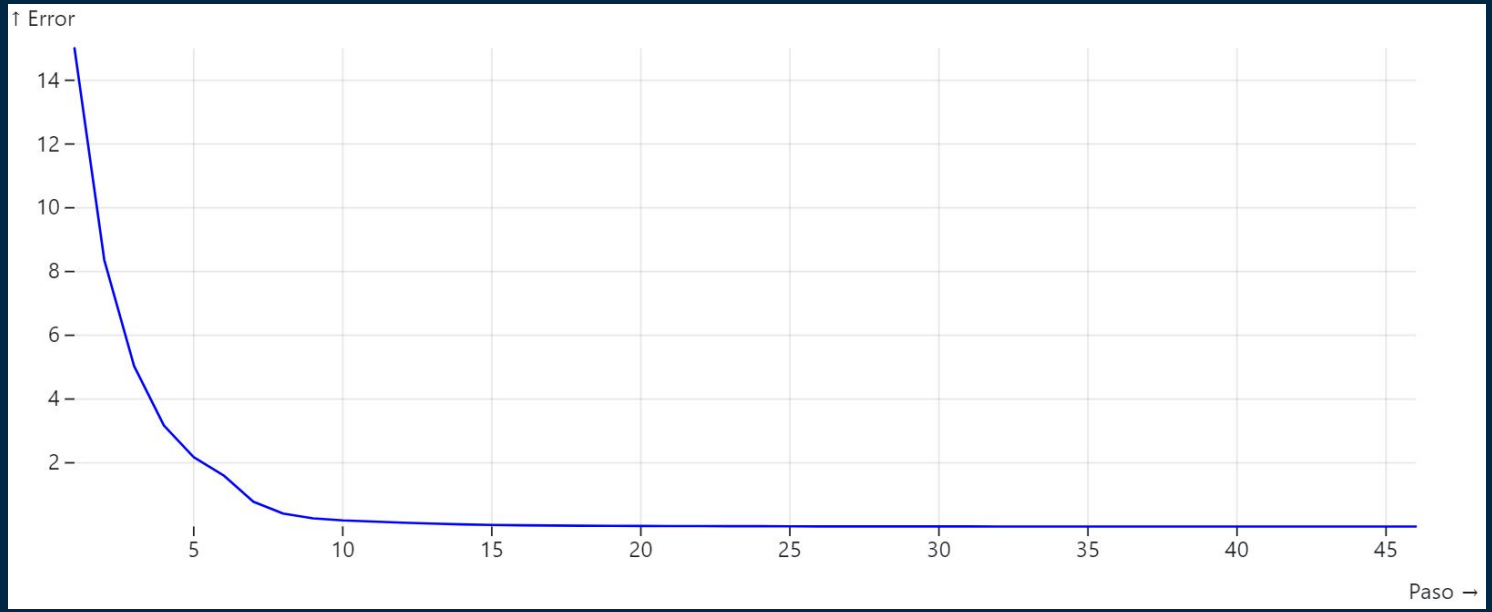


AUTOENCODER CLÁSICO: ESPACIO LATENTE

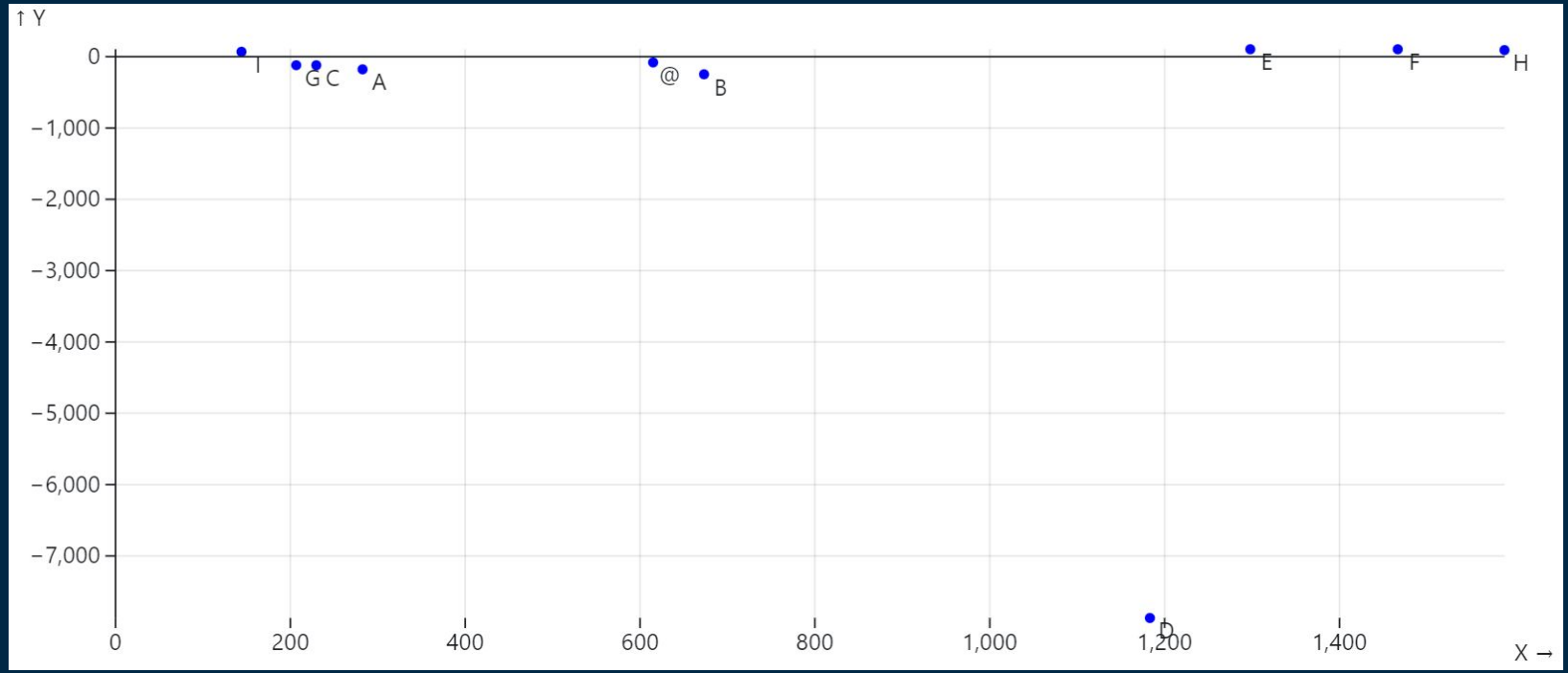


AUTOENCODER CLÁSICO: MUESTRA REDUCIDA

- Al reducir el conjunto de entrada a solo 10 letras la arquitectura [35, 10, 5, 2, 5, 10, 35] logra aprender, llegando a un error acumulado de e-22 luego de 100 épocas



AUTOENCODER CLÁSICO: ESPACIO LATENTE REDUCIDO

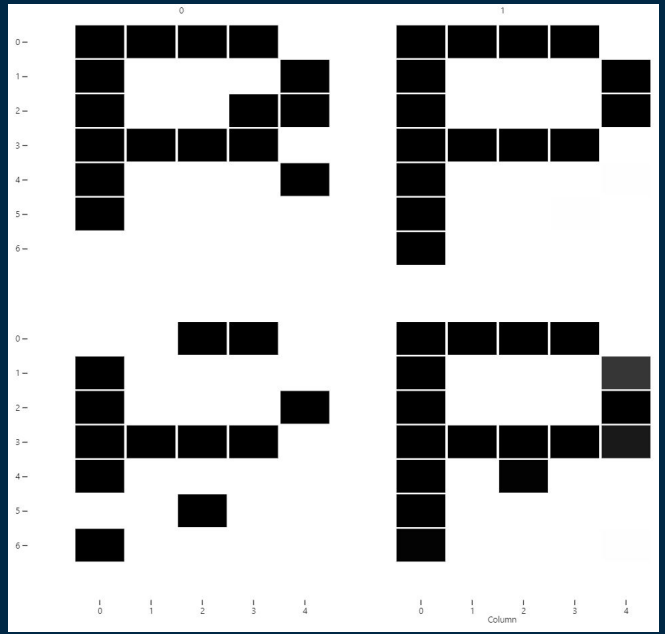


DENOISING AUTOENCODER

- Elegimos una arquitectura de [35, 40, 35]. Este modelo puede ser calificado como un SAE, también usado para feature extraction.
- Utilizamos una probabilidad de ruido $p = 0.1$ y $p = 0.02$.
- Variamos el tamaño del subconjunto de entrada entre: el conjunto completo y 10 letras.
- Para entrenar se generan 5 variantes de cada carácter con ruido aplicado.
- Lectura de los recuadros:
  - 1ra fila a la izquierda: la entrada en entrenamiento.
  - 1ra fila a la derecha: la salida en entrenamiento.
  - 2da fila a la izquierda: la entrada para test.
  - 2da fila a la derecha: la salida para test.

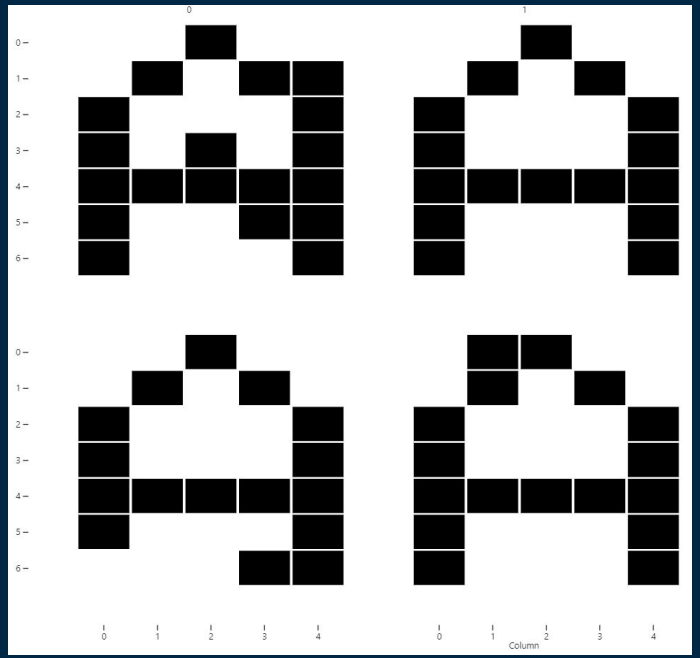
DENOISING AUTOENCODER: CONJUNTO COMPLETO

- Obtuvimos un error acumulado de 40.32 luego de 100 epochs.
  - Nos da un error promedio de 1.26



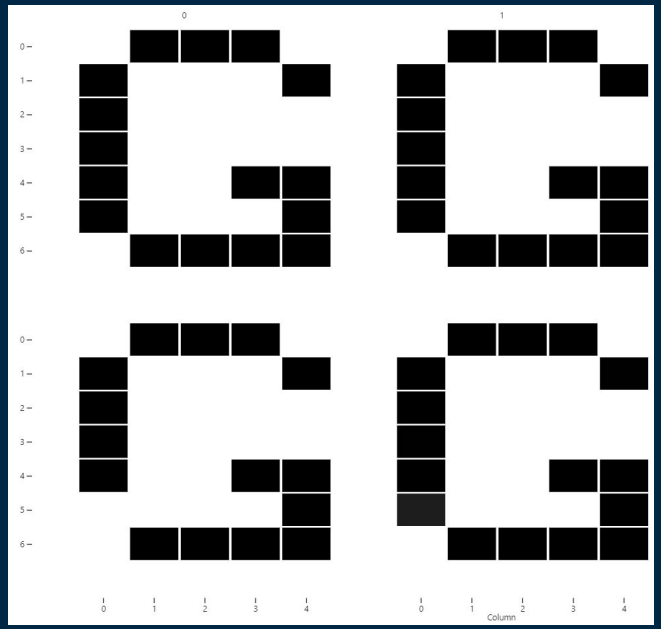
DENOISING AUTOENCODER: 10 LETRAS

- Obtuvimos un error acumulado de 9.5 luego de 100 epochs
  - Nos da un error promedio de 0.95



DENOISING AUTOENCODER: *p = 0.02*

- Obtuvimos un error de 18.37 luego de 100 epochs, corrimos con el conjunto de entrada
  - Nos lleva a un error promedio de 0.57



Muy poco ruido.

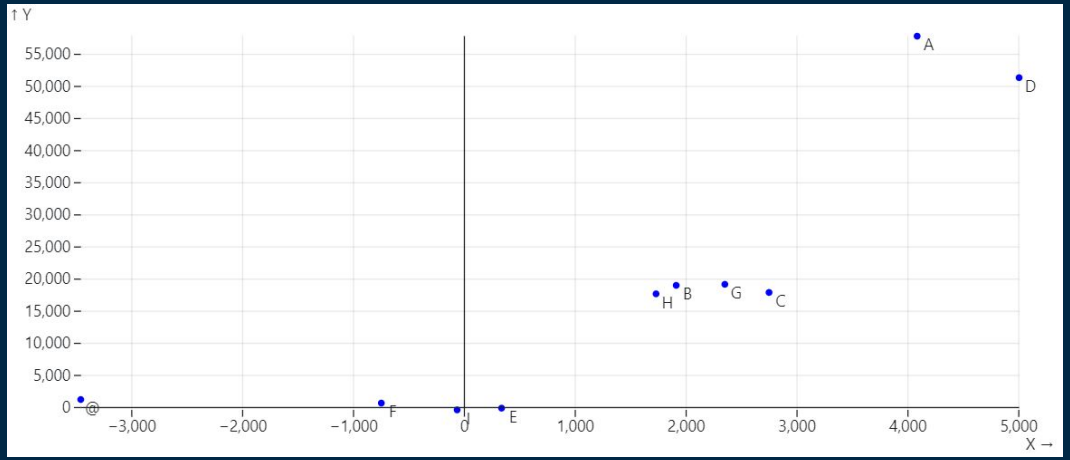
No aprendió bien al no entrenar con una cantidad decente de ruido

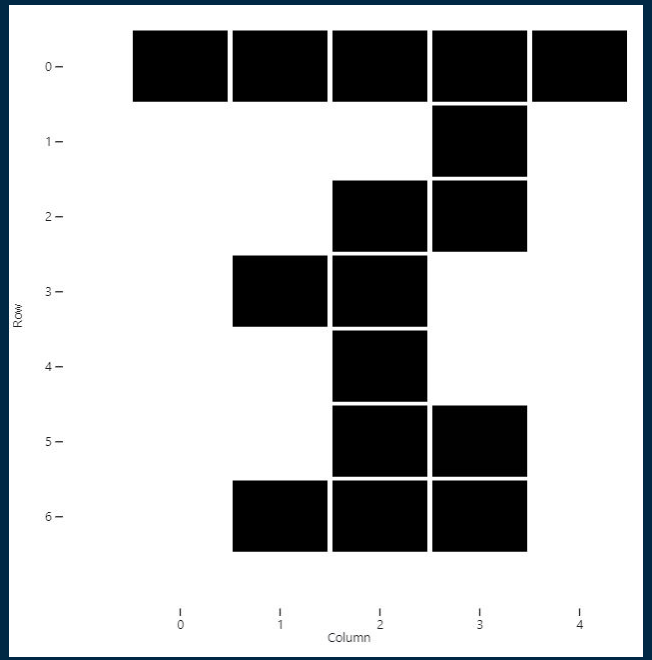
EXPLORACIÓN DEL ESPACIO LATENTE

- Una vez que se tiene al autoencoder entrenado se deja a un lado al encoder.
- Se insertan valores (que son una suerte de combinación de valores reales) en la capa latente y se estudia el output del decoder:
  - Se generan valores para los caracteres con mayor y menor distancia en el espacio latente, y para caracteres con distancia intermedia.
  - Se toma la recta de unión a estos puntos y se toman 5 puntos equiespaciados en esta.
- Estudiamos para todo el set y para un subset de tamaño 10.
- Sin embargo, las salidas no se corresponden con una combinación de las salidas de entradas reales, sino con lo que pareciera ser un output sin mucho sentido.

EXPLORACIÓN DEL ESPACIO LATENTE

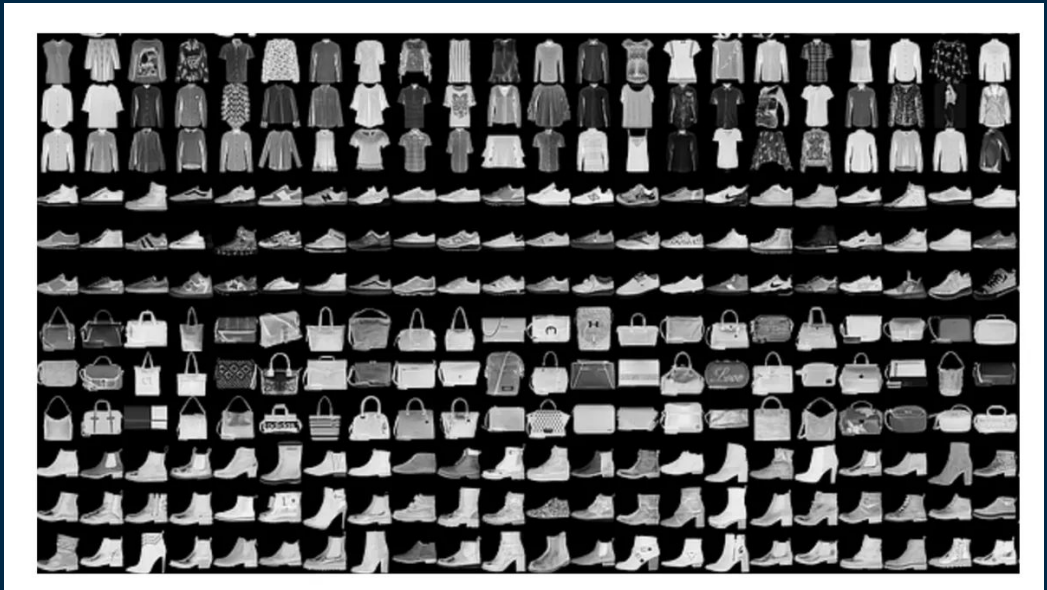
- Para un subset de 10, arquitectura [35, 10, 5, 2, 5, 10, 35]
  - Obtenemos por ejemplo esto como combinación entre @ y B (distancia intermedia)





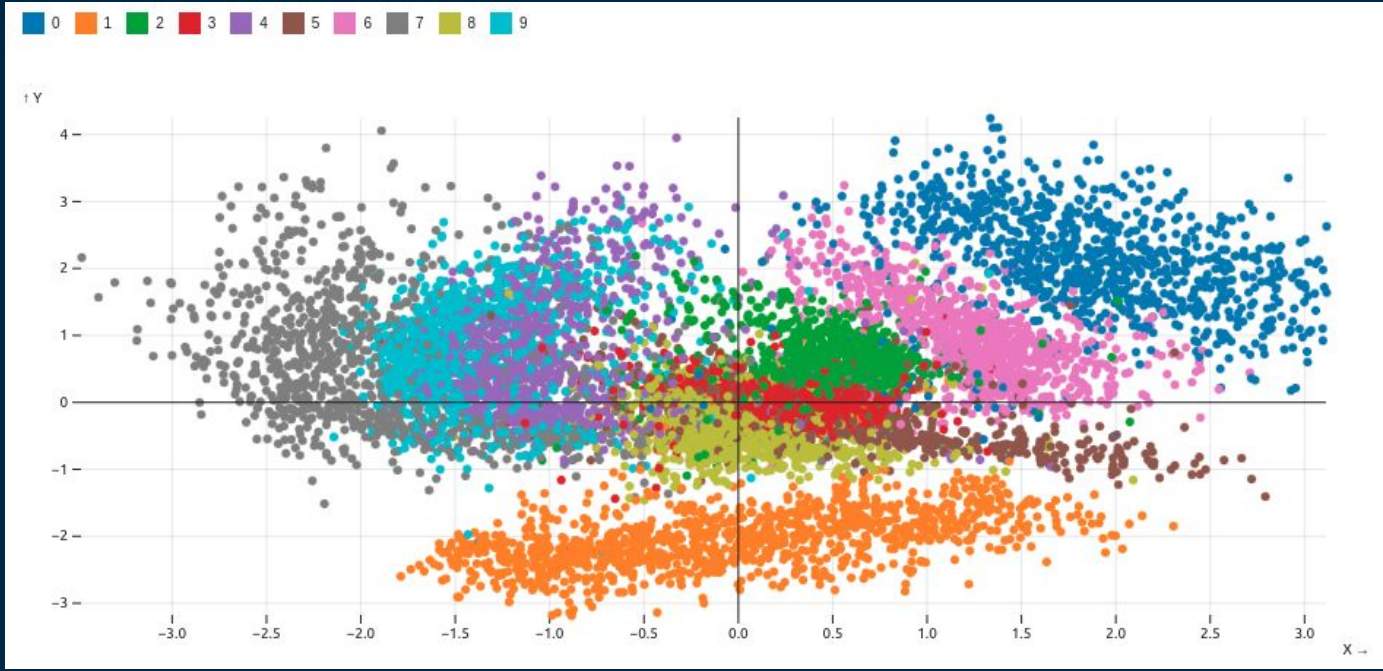
VARIATIONAL AUTOENCODER

- Elegimos un dataset de números manuscritos y diseños de indumentaria
- Generamos una muestra nueva utilizando un autoencoder variacional y exploramos el espacio latente



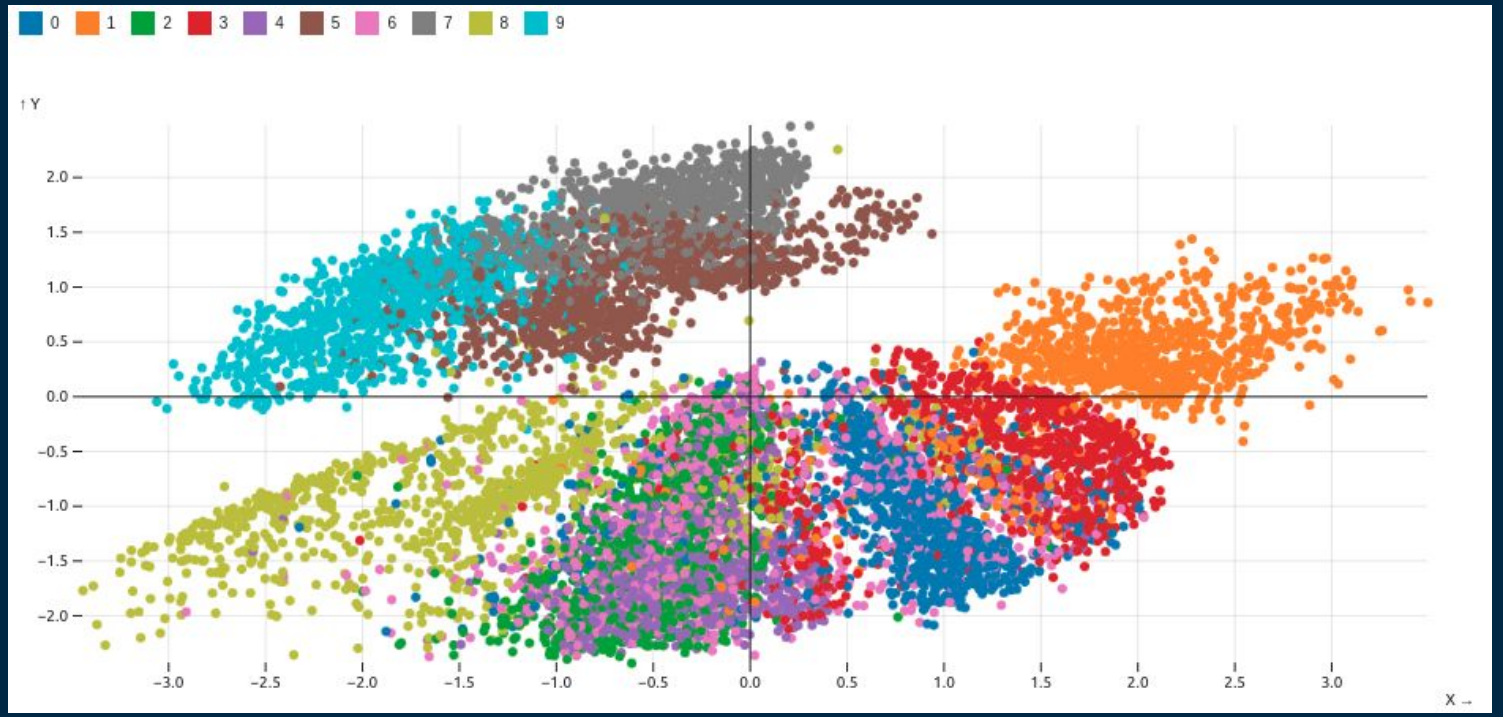
AUTOENCODER VARIACIONAL: ESPACIO LATENTE

- Espacio latente para el conjunto de números

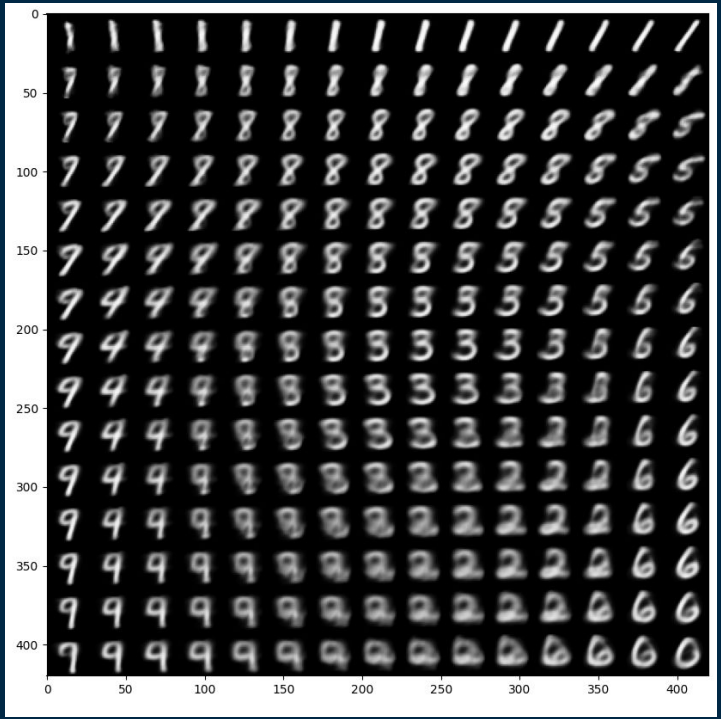


AUTOENCODER VARIACIONAL: ESPACIO LATENTE

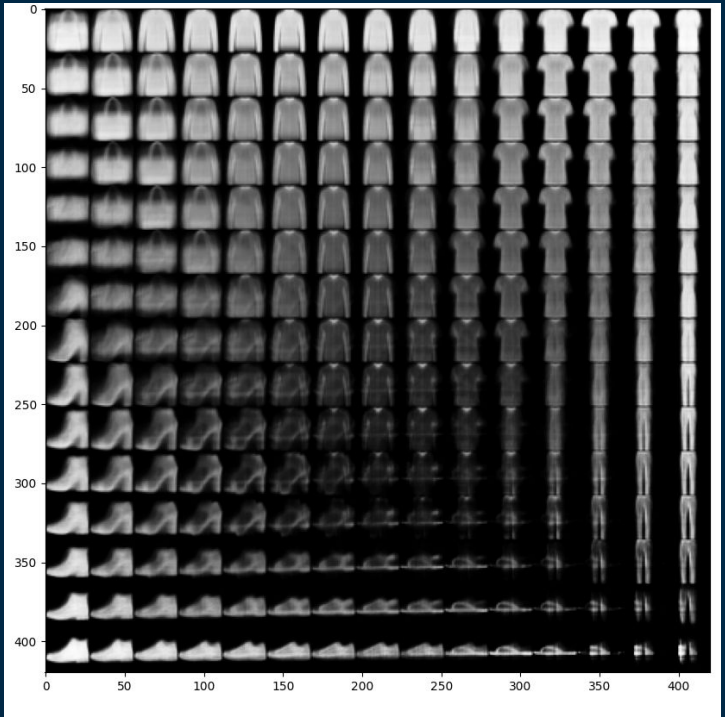
- Espacio latente para el conjunto de indumentaria



VARIATIONAL AUTOENCODER: ESPACIO LATENTE



Números manuscritos



Diseños de indumentaria

CONCLUSIONES

Autoencoder Tradicional:

- El conjunto de letras no puede ser representado por completo por las arquitecturas propuestas:
  - La reducción a 2 dimensiones es demasiado agresiva.
  - Reducir el espacio de datos reduce significativamente el error.
- El error decae de manera desacelerada a lo largo de las épocas.
- Utilizar ADAM llevó a una reducción en la performance y enlentece la convergencia respecto al método de Powell.

CONCLUSIONES

Denoising Autoencoder

- Reducir el ruido lleva a un menor error pero si es muy bajo no funciona como DAE con nuevas muestras pues no aprendió a identificar el ruido.
- Reducir el tamaño del espacio de datos llevó a un menor error promedio

Autoencoder Variacional

- El espacio latente puede utilizarse de manera más efectiva en la generación de muestras que en los autoencoders tradicionales.
- El uso de Keras permitió trabajar con datasets de dimensiones mayores.

MUCHAS GRACIAS