SIA - Trabajo Práctico 5
Autoencoder

Grupo 4
Tomás Dorado
Tomás Dallas

Variables del Autoencoder

- ETA
- Arquitectura de la red
- Inicialización de los pesos
  - Distribución normal uniforme (Desvío estándar)
- Iteraciones
- Solver
- Frecuencia de minimización de error
- Función de activación
- Dataset (*)

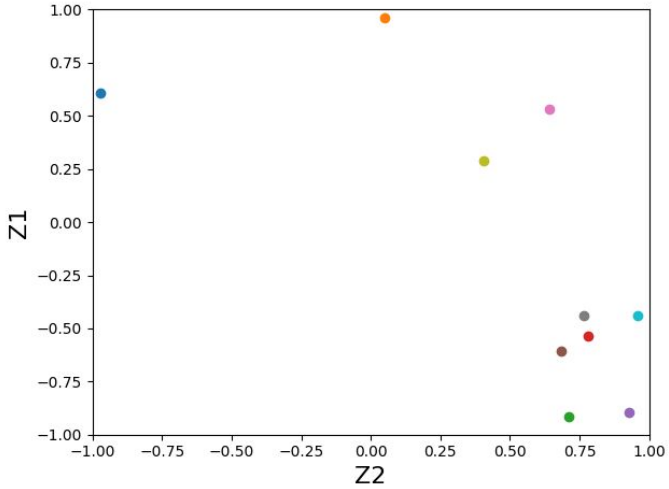
Espacio latente de 2 dimensiones

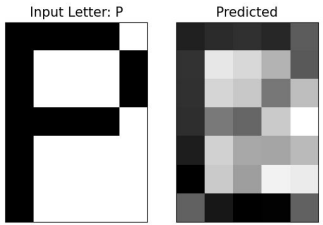
- La arquitectura planteada es:
  - **En el encoder:**
    i. Una capa de entrada con 35 neuronas
    ii. Una capa siguiente de 29 neuronas
    iii. Una capa siguiente de 23 neuronas
  - **Espacio latente:**
    i. Una capa con 2 neuronas (2 dimensiones)
  - **En el decoder:**
    i. Una capa de entrada de 23 neuronas
    ii. Una capa de 29 neuronas
    iii. Una capa de salida de 35 neuronas

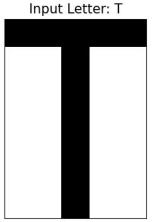
Pruebas iniciales

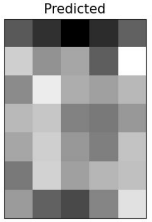
Se utilizó el dataset de fuentes 1
ETA 0.01
300 iteraciones
Desvío estándar = 0.6
Frecuencia de minimización = cada 2 iteraciones

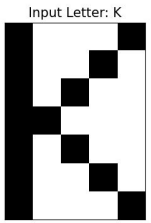
ESPACIO LATENTE

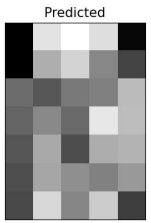


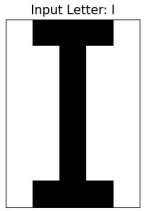


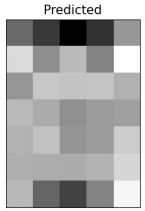












Los resultados no parecen ser muy exactos

Problemas encontrados en las pruebas iniciales

- La cantidad de iteraciones es chica
- Los patrones muy definidos y separables parecieran estar bien definidos en las predicciones
- Hay mucho ruido en las predicciones
- Se entrenó con el dataset completo
    - Esto no generará mucho ruido al reducir a dos dimensiones?
- El desvío estándar será el correcto?

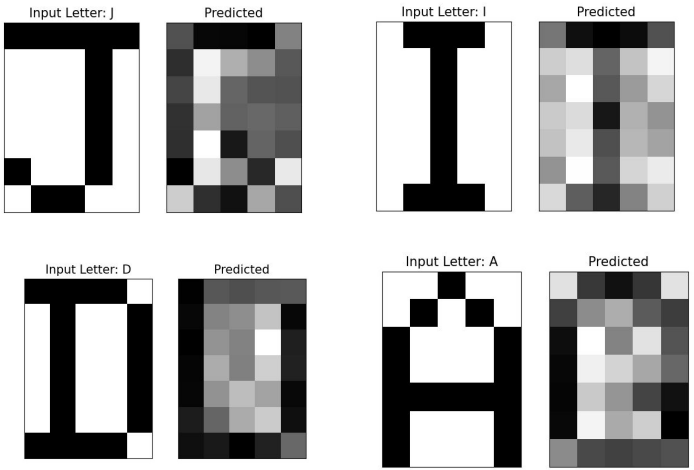
Plan de acción

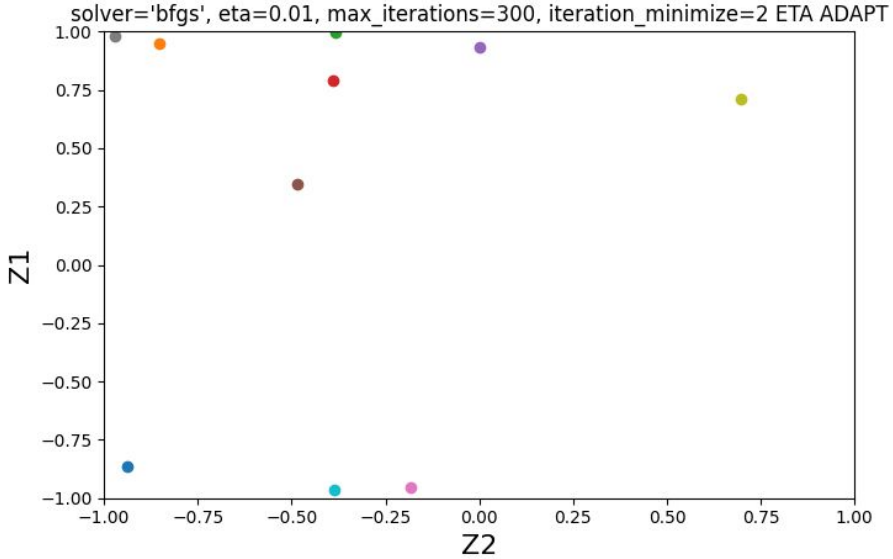
- Variar un parámetro del autoencoder dejando los otros fijos.
- La elección de los parámetros fijos fueron elegidos de manera arbitraria después de correr muchas simulaciones, y viendo la adaptación de cada uno.
- Lo que nos interesa ver es **cómo** se comporta el autoencoder tanto en la capa latente como en las resultados de las predicciones modificando un parámetro y dejando los otros fijos para encontrar el mejor match entre todos.

Variando el ETA

- Variamos el ETA con valores pertenecientes a **{0.0005, 0.005, 0.05, 0.1, 0.15, 0.45}**
- Utilizamos un ETA de 0.01

ETA adaptativo

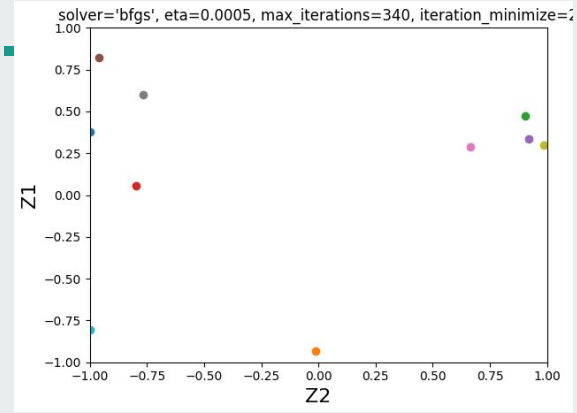


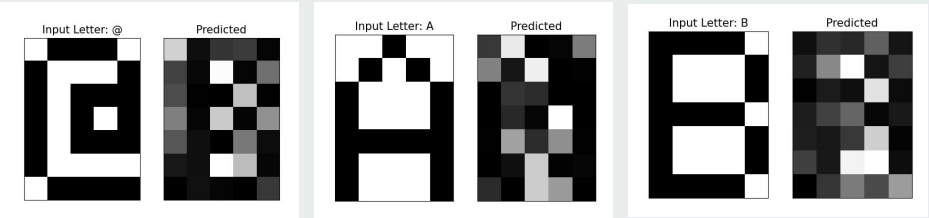


- **Hay mucho ruido en las predicciones**
- **El espacio latente pareciera estar bien distribuido pero muy en los extremos**
- **El ETA está creciendo mucho en la adaptación?**

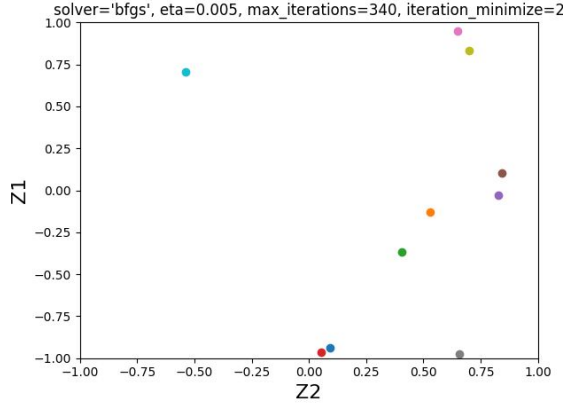
- Una sola prueba utilizando ETA = 0.01
- No llegamos con el tiempo a probarlo de manera correcta, creemos que por un problema de implementación y que no está muy optimizado no obtuvimos buenos resultados

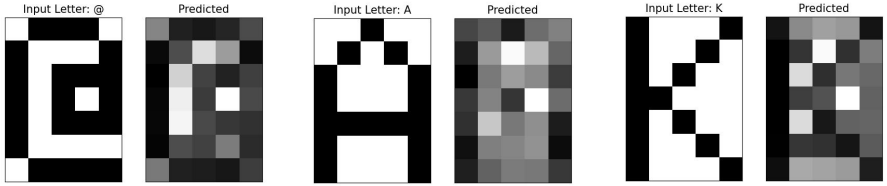
ETA 0.0005





ETA 0.005



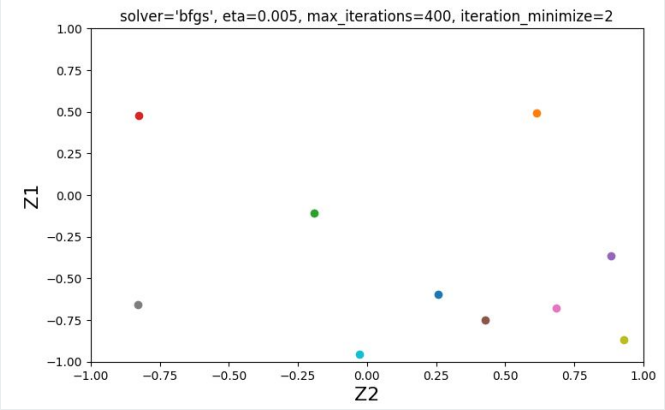


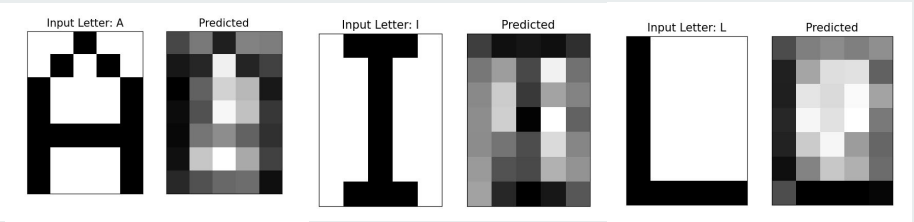
Un ETA chico pareciera dar

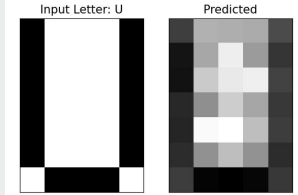
- Precisión en las predicciones sobre el patrón fuerte, aunque con mucho ruido (pocas iteraciones respecto del ETA?)
- Buena distribución en la capa latente

ETA 0.05

Para este caso se decidió aumentar un poco la cantidad de iteraciones para intentar confirmar hipótesis anterior

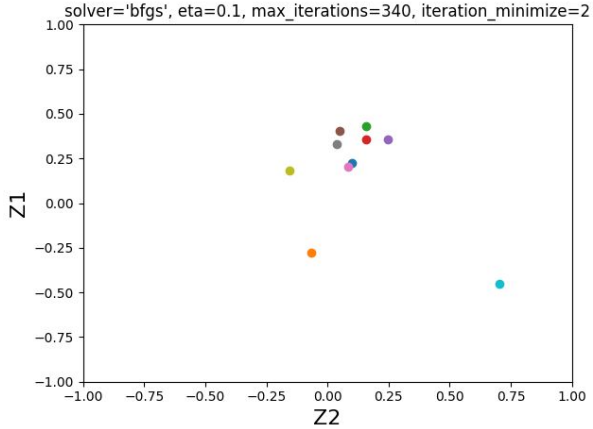


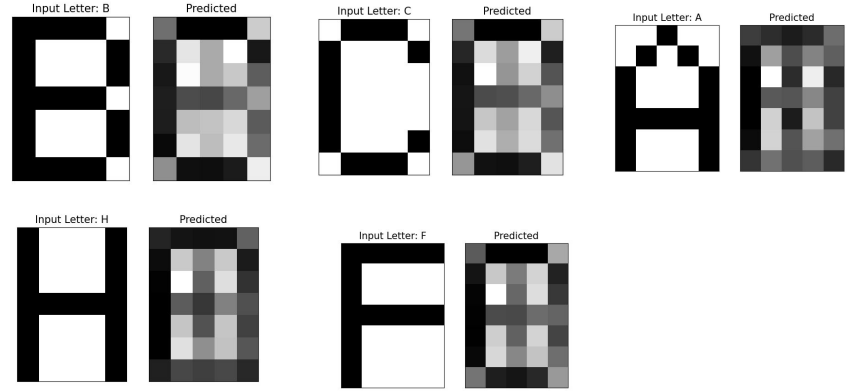




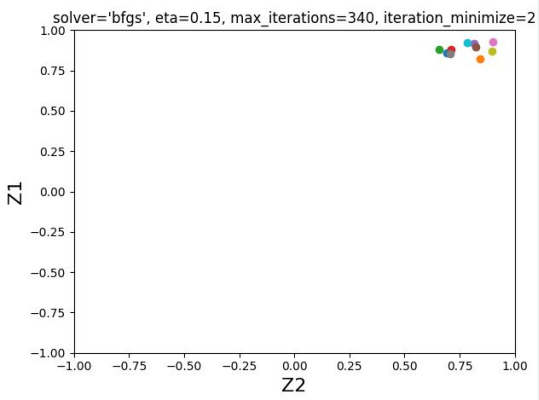
Las predicciones son más precisas
La capa latente está mejor distribuida

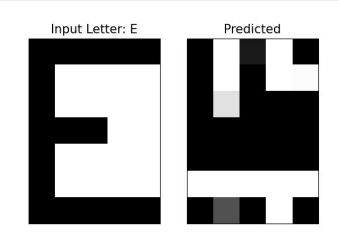
ETA 0.1

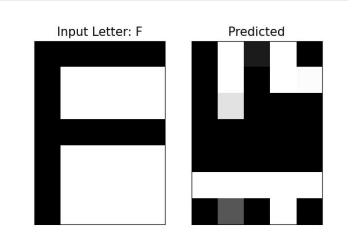




ETA 0.15







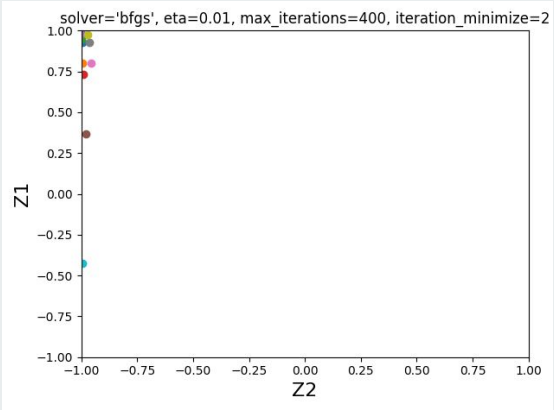
CONCLUSIONES

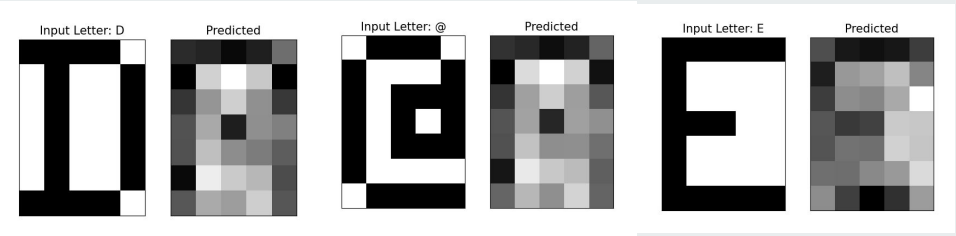
- Un ETA muy grande quita cualquier tipo de precisión deseada
- Un eta entre 0.01 y 0.001 pareciera ser el punto medio ideal
  - Mayor precisión de los patrones
  - Sólo faltaría afirmar esa precisión y reducir el ruido

Variando la cantidad de iteraciones

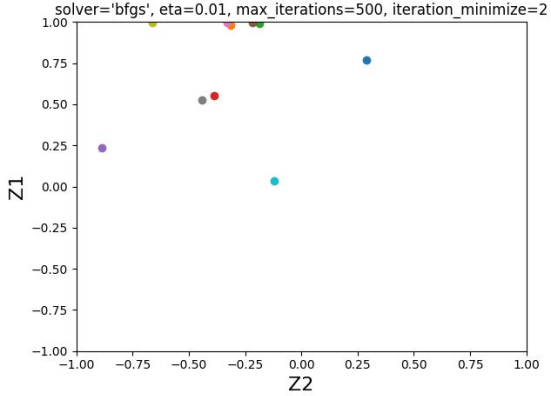
- Buscamos mayor precisión con un ETA entre 0.01 y 0.001
- Utilizamos 400, 500, 600 y 1500 iteraciones

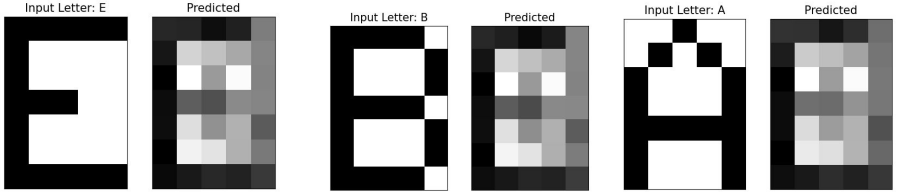
400 iteraciones





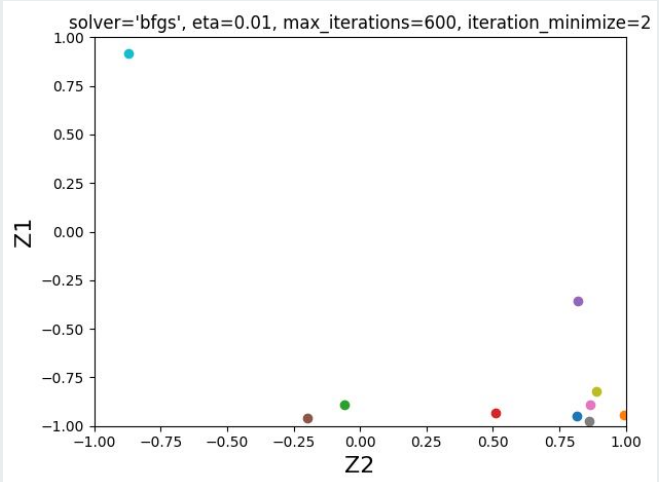
500 iteraciones

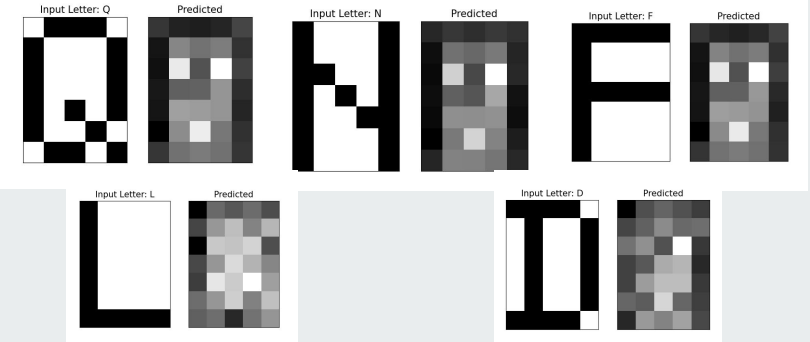




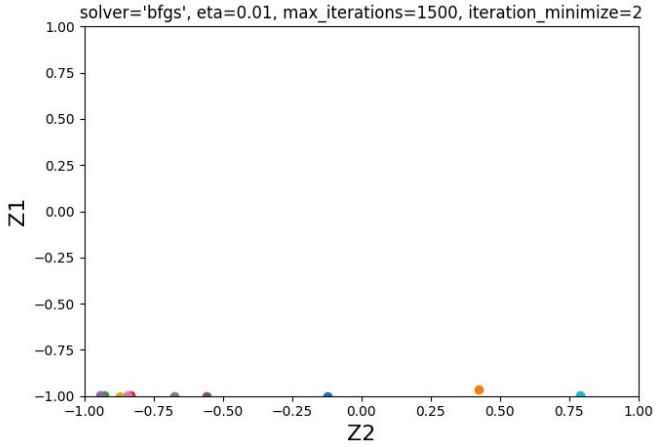
Pareciera que cuanto más iteraciones, más dispersión en el espacio latente.. ¿Pero hay sobreentrenamiento?

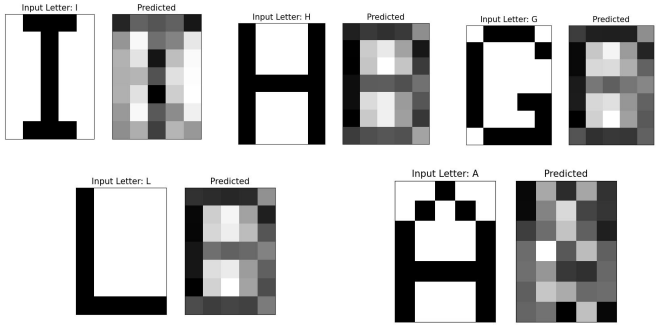
600 iteraciones





1500 iteraciones





Conclusiones variando la cantidad de iteraciones

- Cuanto más iteraciones se realizan, la capa latente se distribuye más.
- Pero esa distribución no siempre es la ideal (todas tiendan a un mismo *y* cuando las iteraciones tienden a ser mayores)
- Al estar minimizando el error cada 2 iteraciones, más iteraciones implican un aprendizaje mejor para el autoencoder, lo que implica un sobreentrenamiento
    - Esto creemos notarlo cuando para letras con patrones distintos obtenemos una predicción igual o similar

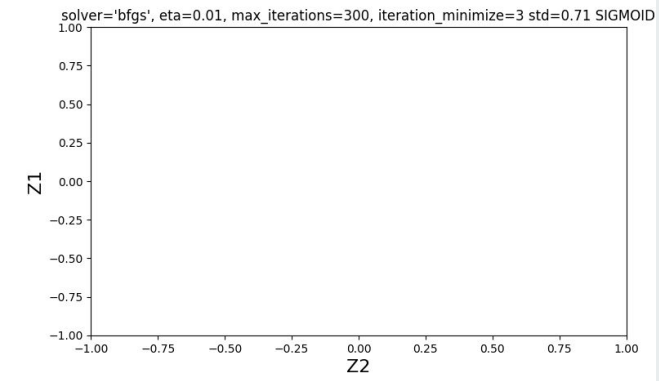
Variando función de activación

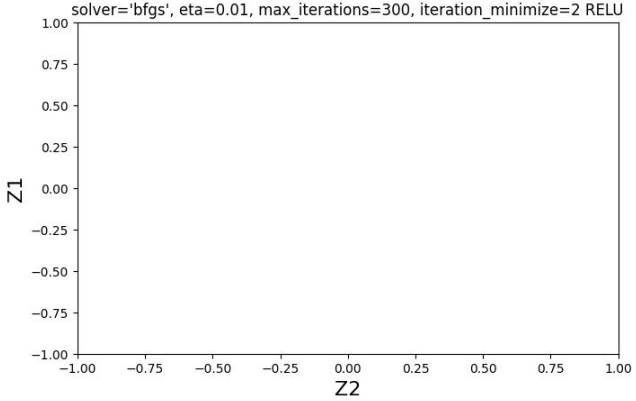
- Buscamos ver si la función de activación tiene algún efecto muy grande en el ajuste de los pesos
- Cómo todas las simulaciones de prueba se realizaron con ***tanh*****,** solo se muestran resultados con función ***sigmoide*** y función ***relu***
- **Se decidió no variar los parámetros de cada función en particular porque se buscó observar qué función (a nivel básico) se ajusta más a nuestro autoencoder y el dataset elegido.**

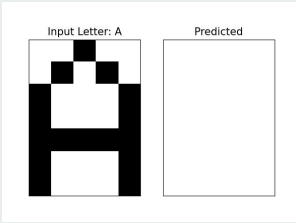
Sigmoide

No hay puntos entre -1 y 1 en la capa latente. Podría ser un error de implementación o que no ajustamos lo suficiente el resto de los parámetros a estas funciones.

Relu

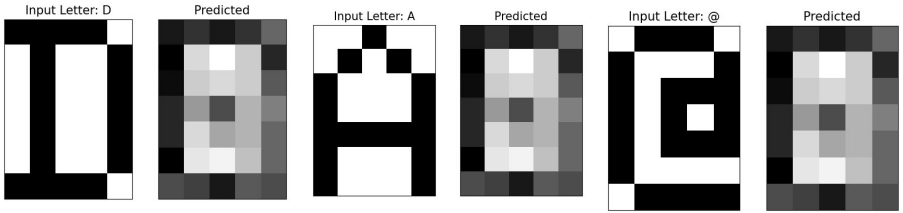






Las predicciones, además, no son exactas para ambos casos.

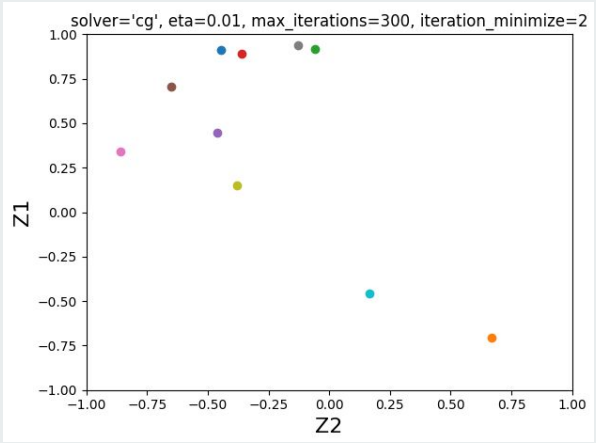
Se decide utilizar *tanh* y desestimar futuras pruebas variando estas funciones

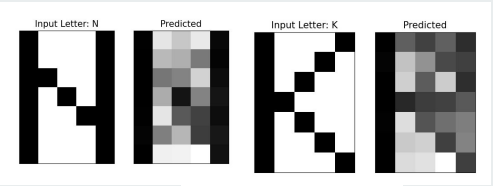


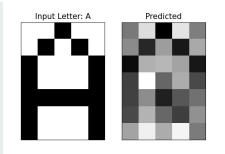
Variando función de minimización del error

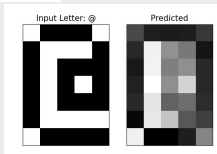
- Buscamos optimizar el ajuste de los pesos cada **x** iteraciones, de manera tal que el autoencoder no se entrene de más, mantener las optimizaciones observadas, y agregar más precisión a las predicciones
- Cómo todas las simulaciones se realizaron utilizando la función ***bfgs*** *(Broyden–Fletcher–Goldfarb–Shanno)*, sólo se van a mostrar resultados para ***Powell*** y ***CG*** *(Conjugate gradient)*

CG



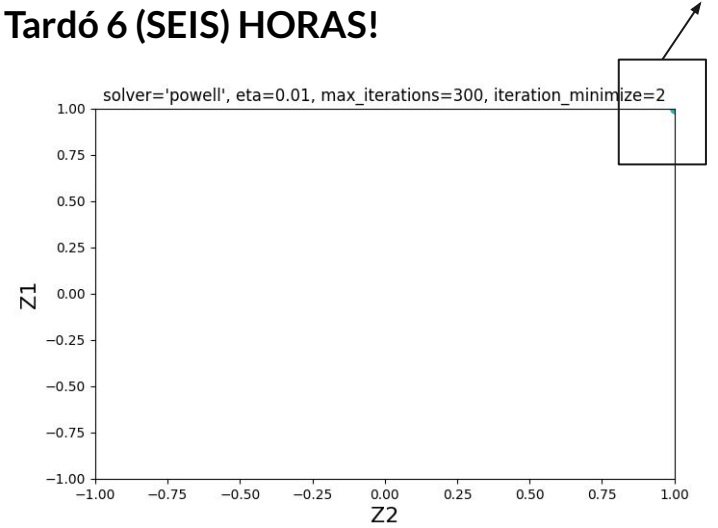


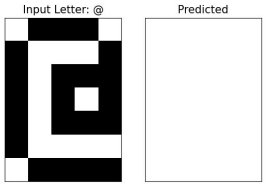




Powell

Tardó 6 (SEIS) HORAS!





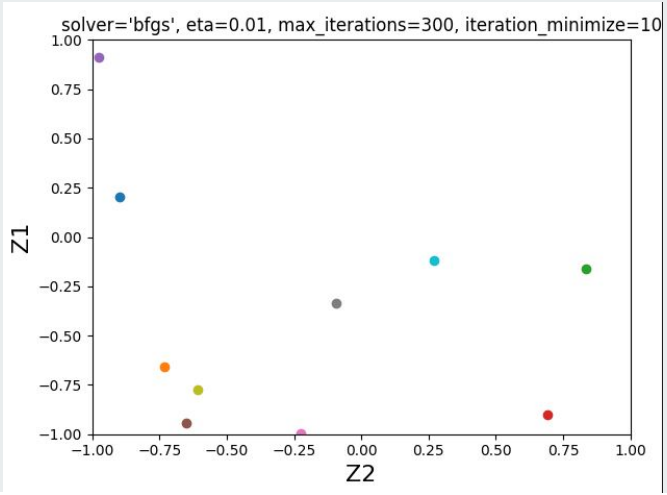
Conclusiones variando función de minimización del error

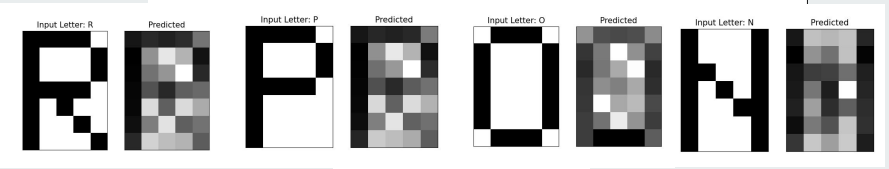
- Decidimos descartar utilizar Powell debido a que la simulación tardó alrededor de 6 horas y los resultados no eran alentadores.
- CG parece una buena alternativa, da buenos resultados.
    - Capa latente distribuida
    - Predicciones precisas y con poco ruido
    - Creemos que ajustando los parámetros podría servir de alternativa a ***BFGS***
    - Poco tiempo de ejecución

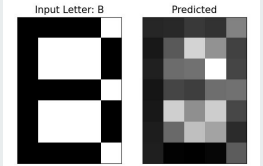
Variando la frecuencia que se optimizan los errores

- Eligiendo la mejor función para optimizar los errores, luego buscamos la frecuencia ideal que maximice los resultados
- Cada:
    - 10 iteraciones
    - 30 iteraciones
    - 50 iteraciones
    - 150 iteraciones
    - Todas las iteraciones
    - Ninguna iteración

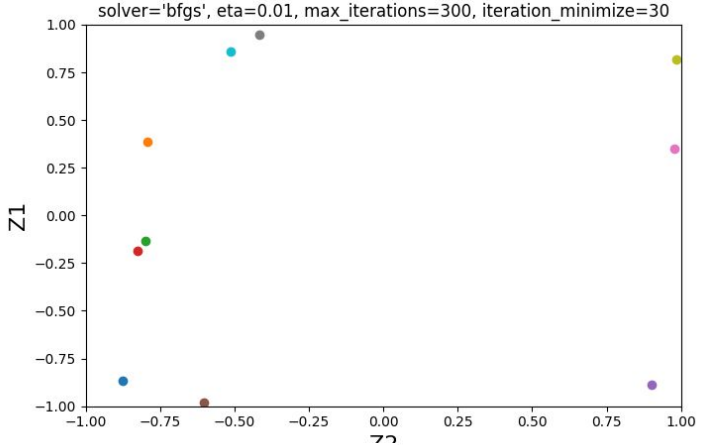
Cada 10 iteraciones

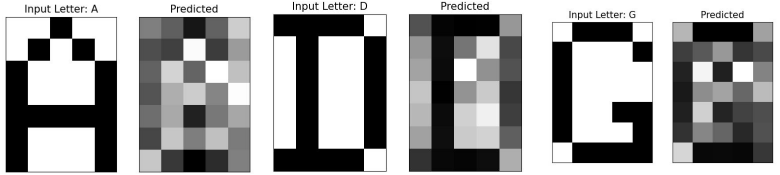


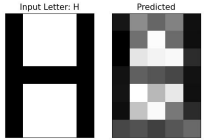




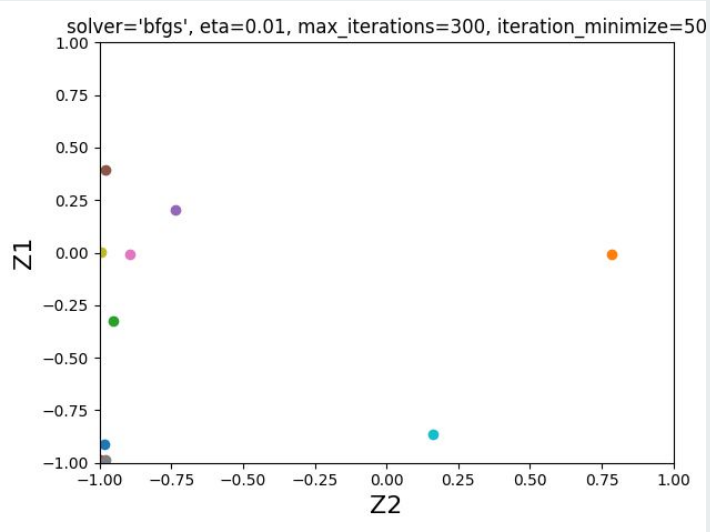
Cada 30 iteraciones

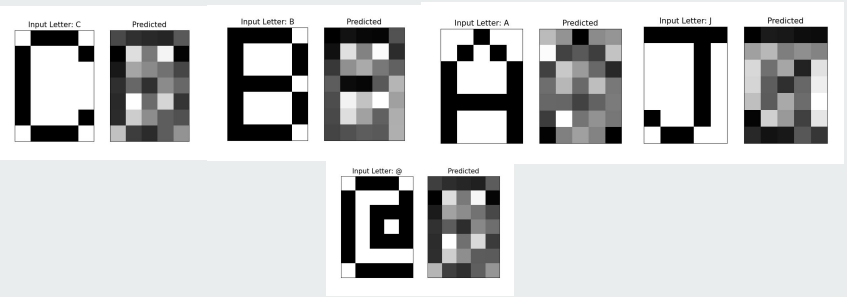




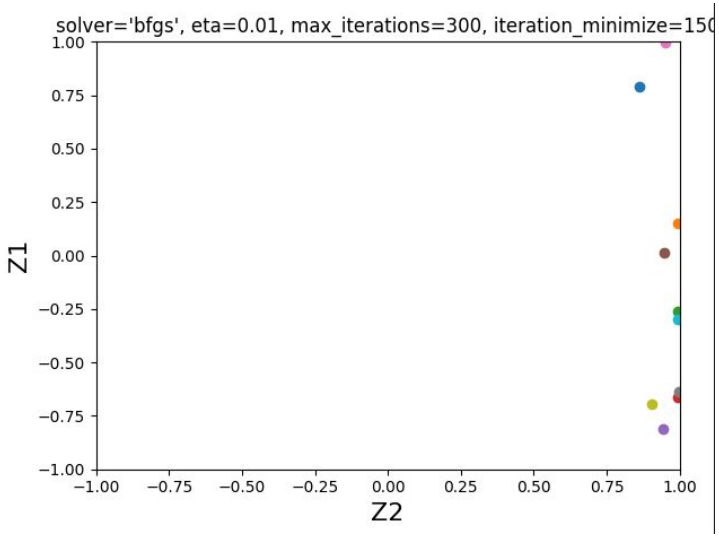


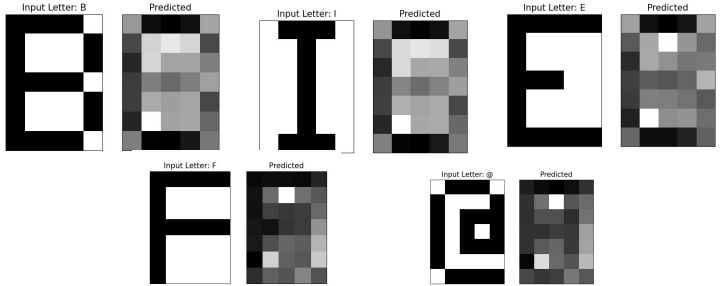
Cada 50 iteraciones



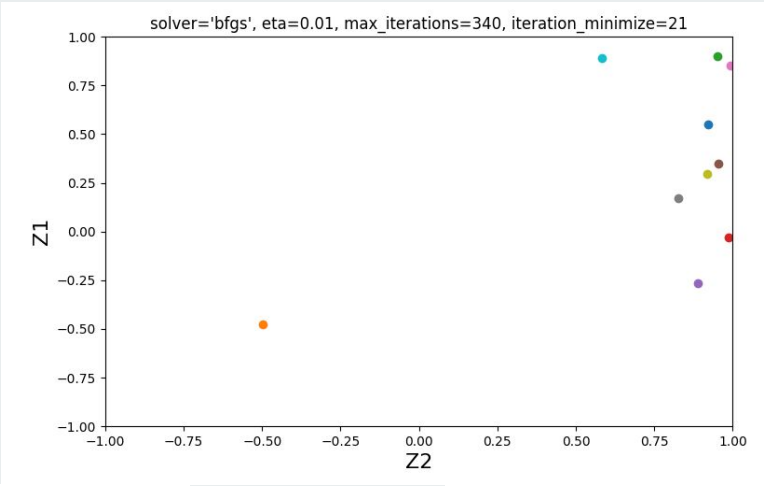


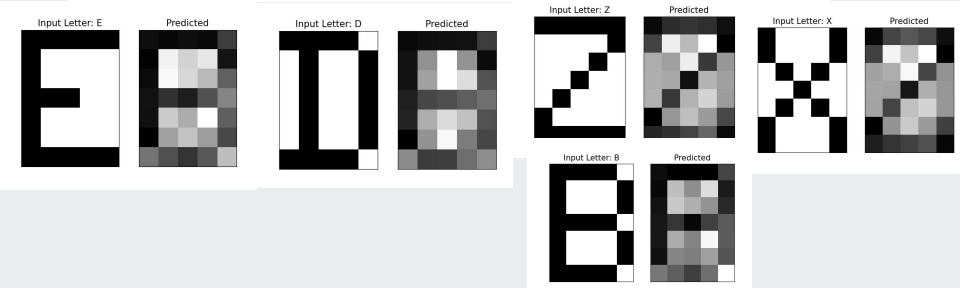
Cada 150 iteraciones



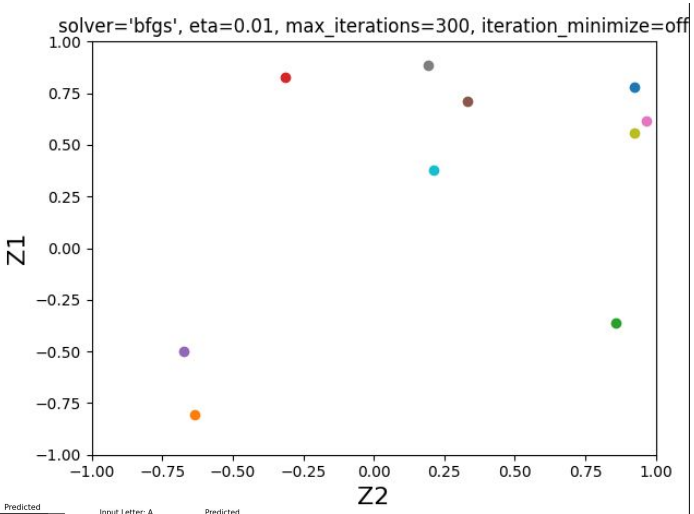


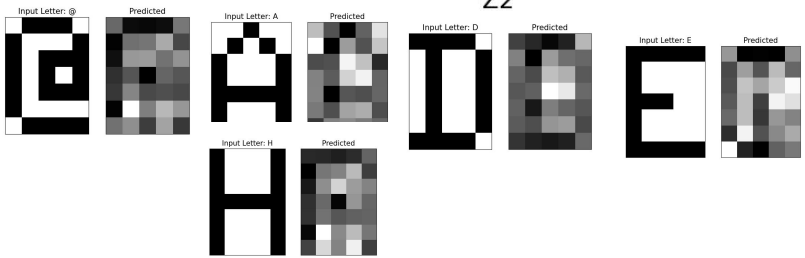
Todas las iteraciones





Ninguna iteración





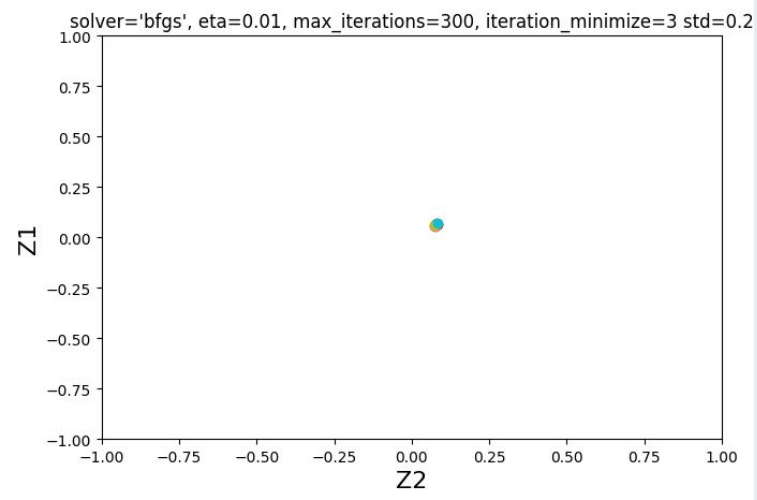
Conclusiones variando la frecuencia que se optimizan los errores

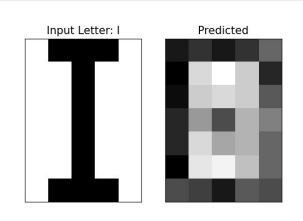
- Cuanto cada más iteraciones (en los extremos) las predicciones de los patrones menos marcados producen mucho ruido y todo el espacio latente se distribuye sólo en un sector.
- Cuando no hay minimización del error la capa latente está bien distribuida, pero incorrectamente. Las predicciones NO son exactas, tienen mucho ruido.
- El punto ideal, para este conjunto de parámetros, incluido el dataset, lo encontramos entre minimizar entre 1 iteración y 6/7 iteraciones manteniendo 300 iteraciones.

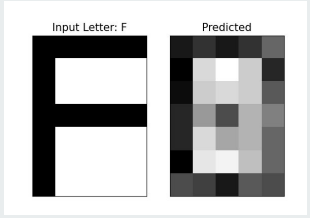
Variando la generación de pesos

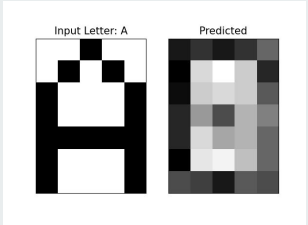
- Dado que se utiliza una función con distribución normal gaussiana para la generación de pesos randoms, se puede variar el desvío estándar y la media
- Creemos que el desvío estándar es el valor que más nos va a modificar el comportamiento, por lo que decidimos variar entre:
    - 0.2
    - 0.6
    - 0.9
    - 1.3

Desvío = 0.2

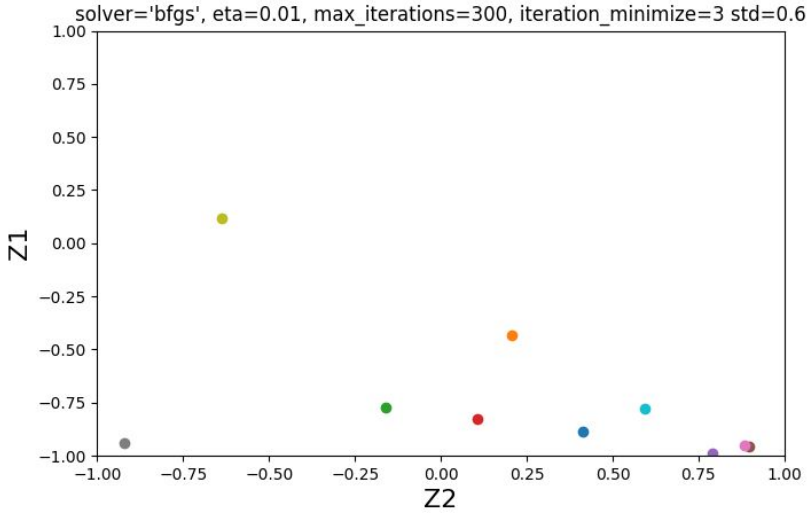


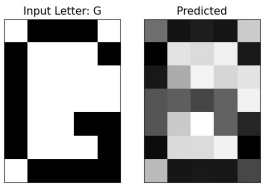


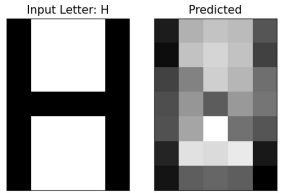


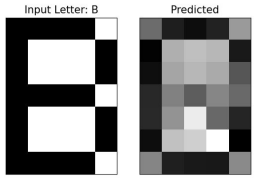


Desvío = 0.6

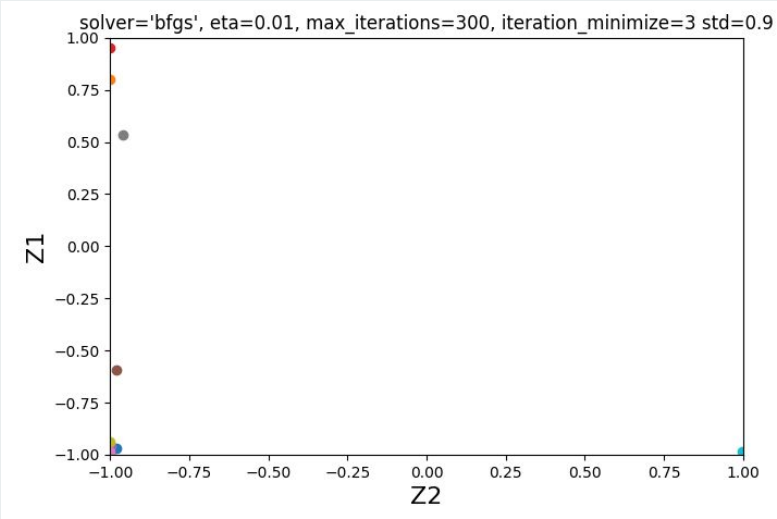


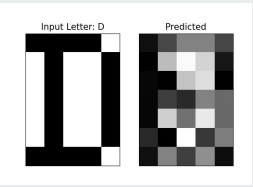


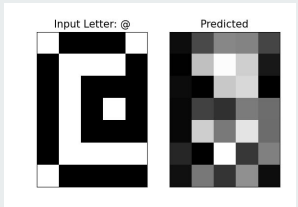


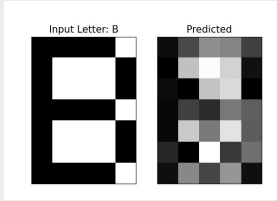


Desvío = 0.9

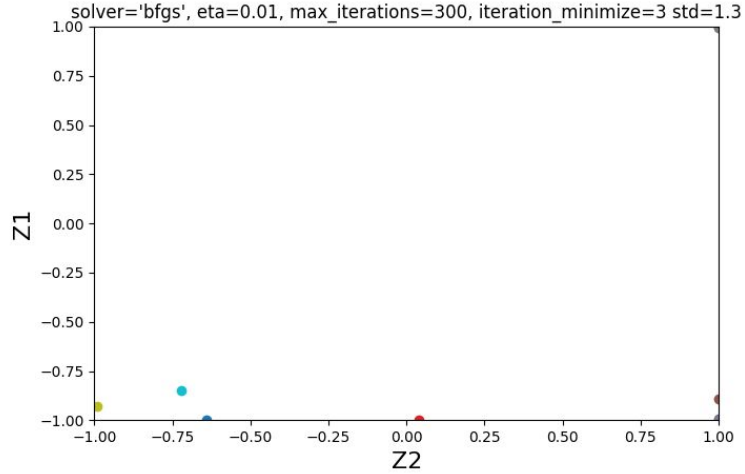


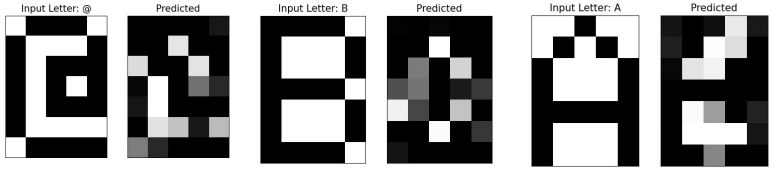






Desvío = 1.3





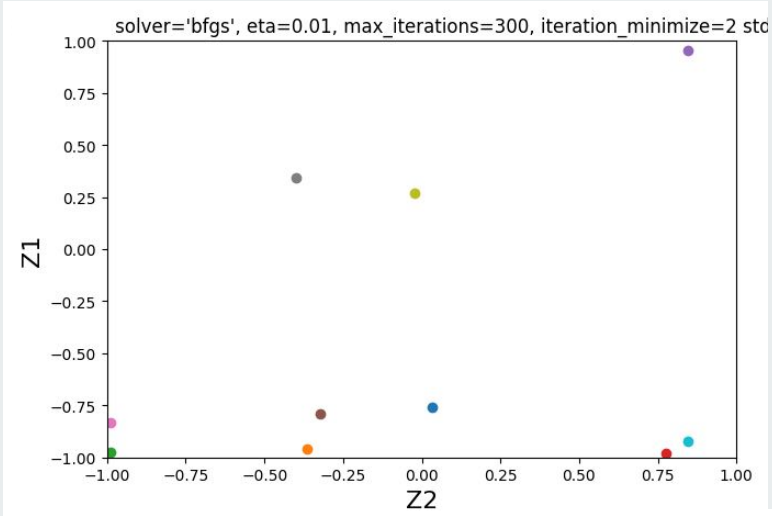
Conclusiones variando la generación de pesos

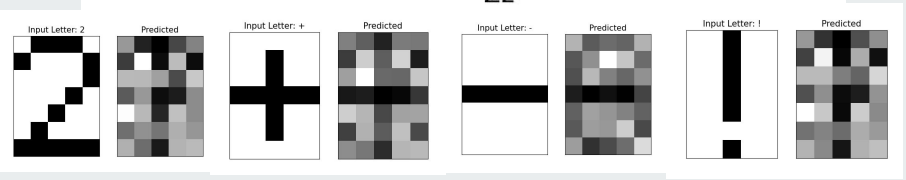
- Con un desvío estándar muy chico o muy grande, el espacio latente se distribuye de manera *errónea* y las predicciones no son precisas. Incluso llegan a ser **MUY** erróneas
- El punto justo del desvío estándar lo encontramos entre 0.6 y 0.75.

Variando el dataset

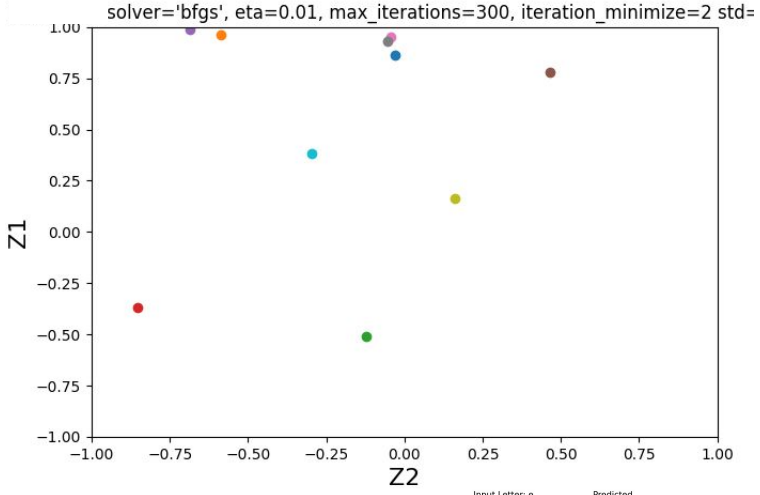
Por último, nos quedaba testear todos estos parámetros variando los 3 datasets provistos. Si bien creemos que no es un parámetro en sí del autoencoder, tenemos que tener en cuenta que todas las pruebas se realizaron sobre un dataset en particular, que elegimos específicamente por los patrones bien marcados, y con los datasets que dejamos de lado el autoencoder podría no comportarse de la misma manera

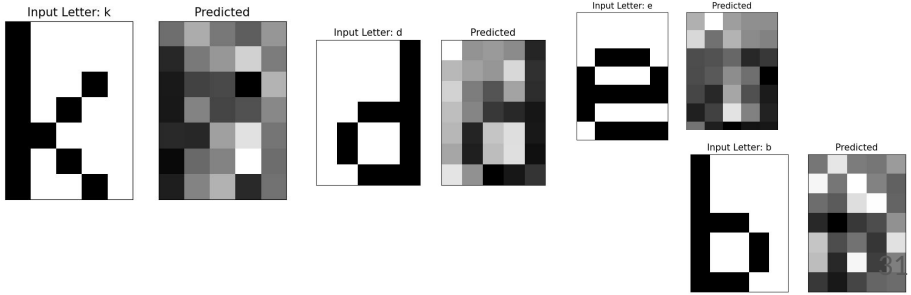
Dataset 1





Dataset 3





Conclusiones variando el dataset

- En una primera instancia, los resultados obtenidos son positivos para con todos los datasets.
- Lo que notamos con el datasets que tiene números y signos es que los caracteres no tienen un patrón muy marcado, y suelen confundirse en algunas predicciones.
- Con el dataset de letras minúsculas notamos que hay muchos 0s o espacios en blanco, que agregan mucho ruido a las predicciones. Los patrones no están marcados.

Mejores parámetros elegidos

Parámetros 1:

ETA = 0.0085

Iteraciones = 220

Minimizaciones = cada 4 iteraciones

Arquitectura = [35, 28, 23, **2**, 23, 28, 35]

Solver = *BFGS*

Activación = *tanh*

Std = 0.73

Dataset de entrenamiento: 10 elementos

Parámetros 2:

ETA = 0.0073

Iteraciones = 220

Minimizaciones = cada 2 iteraciones

Arquitectura = [35, 29, 21, **2**, 21, 29, 35]

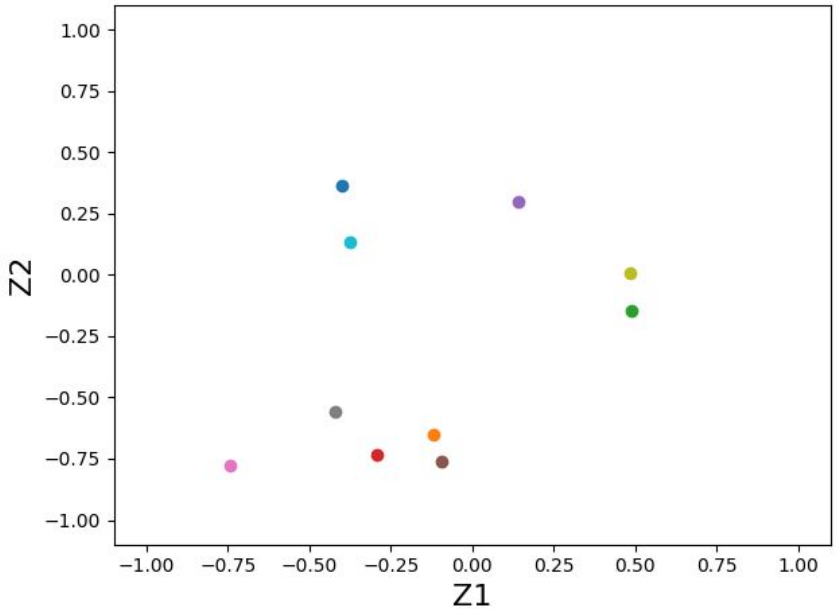
Solver = *BFGS*

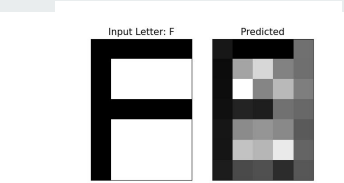
Activación = *tanh*

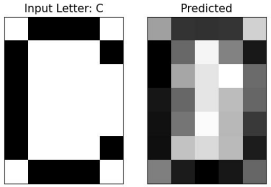
Std = 0.68

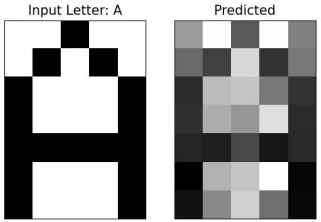
Dataset de entrenamiento: 10 elementos

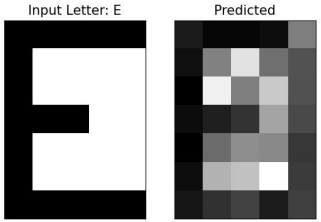
Parámetros 1

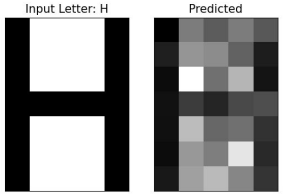


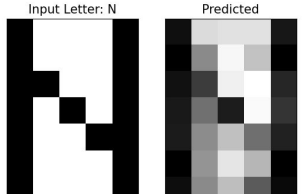


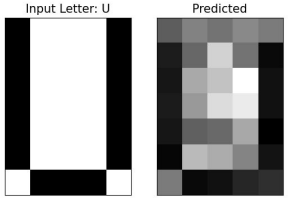


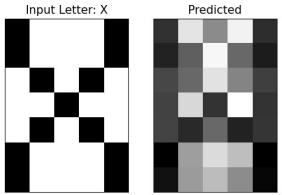


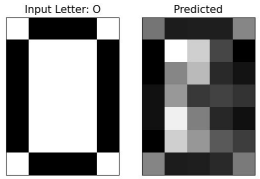


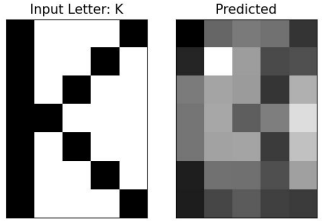




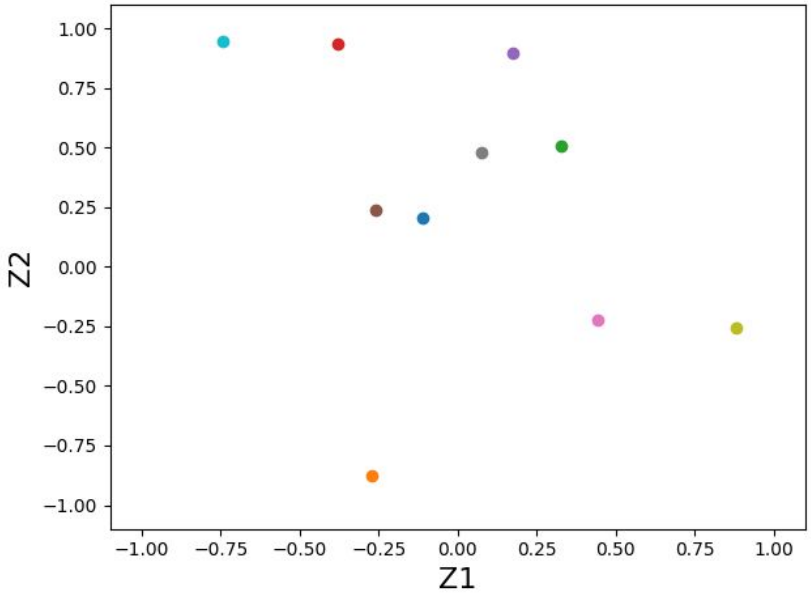


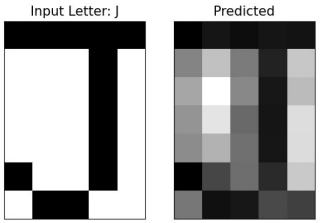


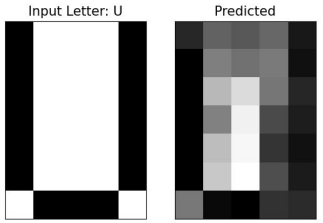


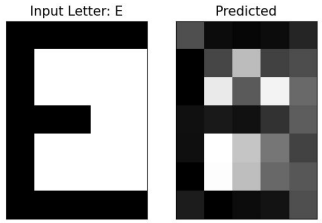


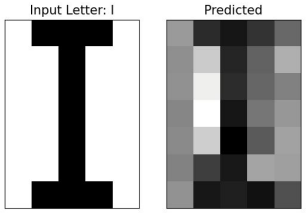
Parámetros 2

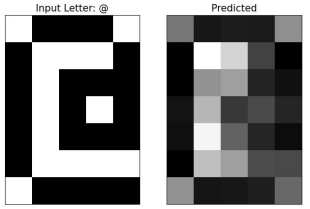


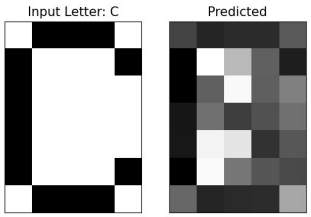




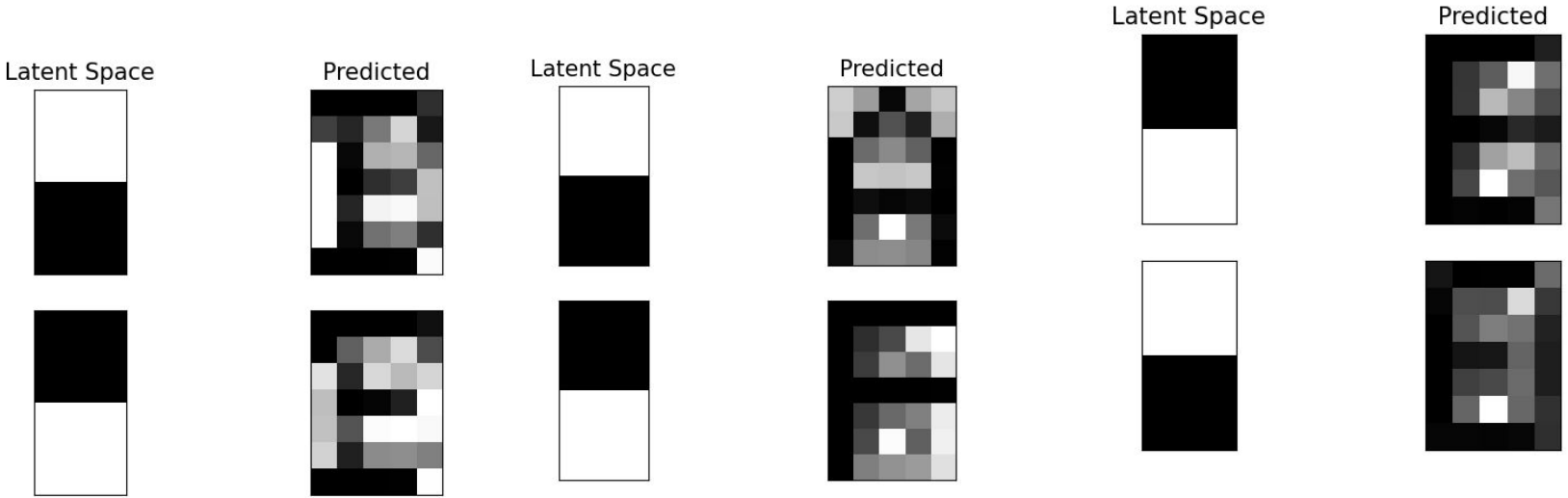


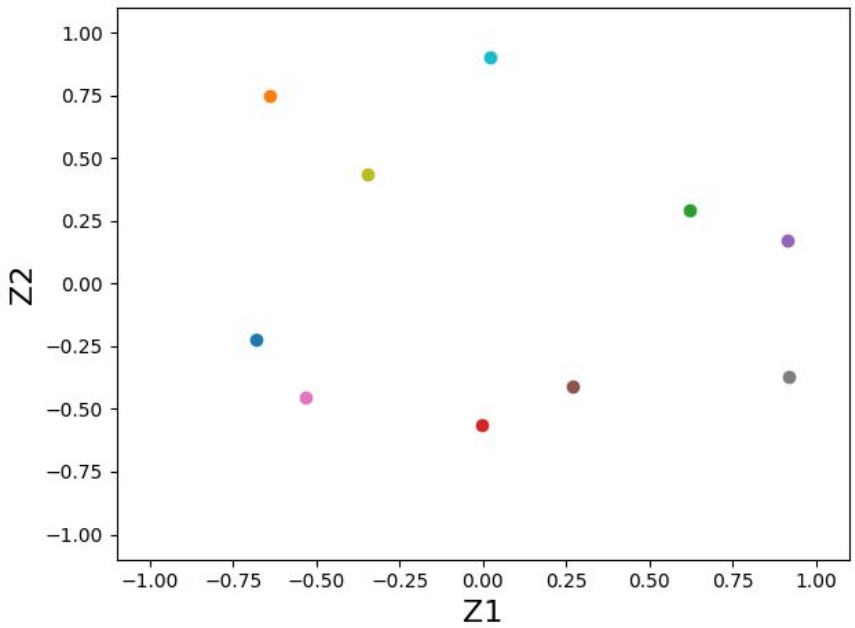


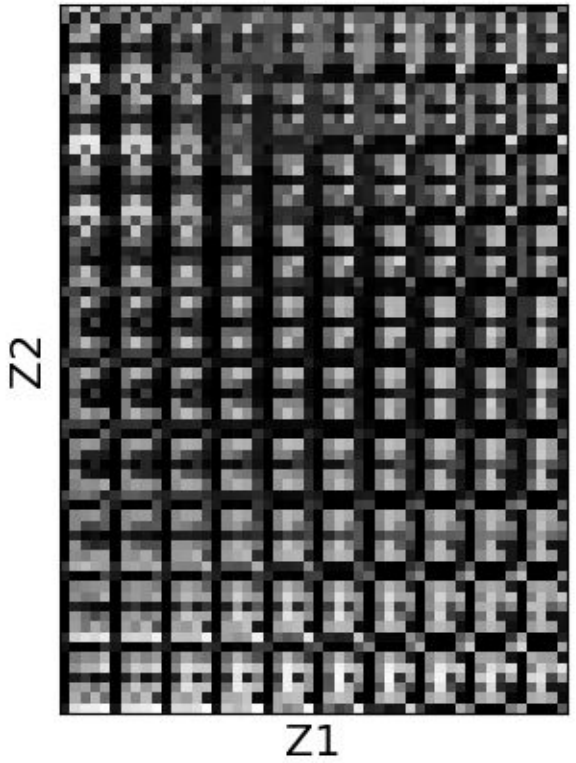




Generación de una nueva letra que no pertenece al dataset



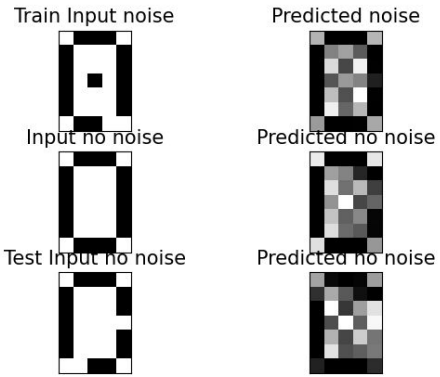

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Z2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Z1

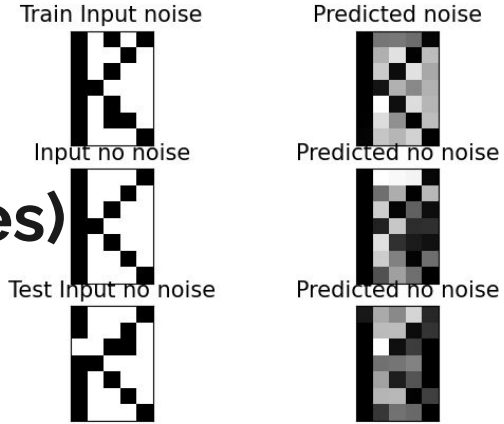

Z2
Z1

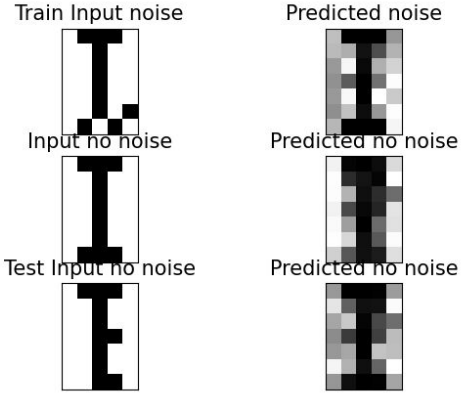
Denoising Autoencoder

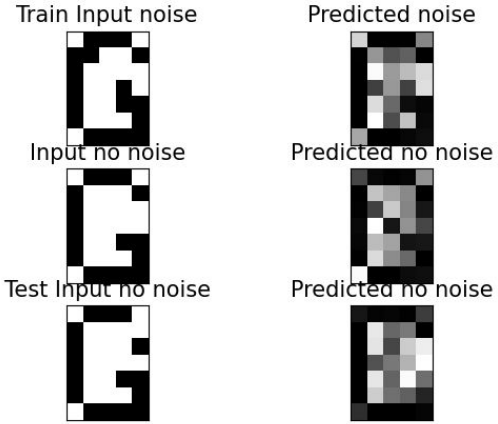
- Se planteó una arquitectura de [35, 25, **20**, 25, 35]
- Un máximo de 200 iteraciones
- Se agregó ruido modificando 2 y 3 *pixeles* de cada letra de manera random

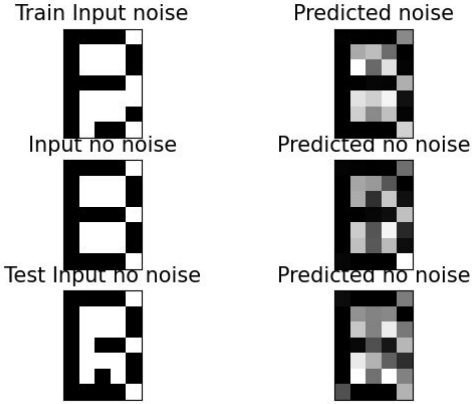
Resultados (2 píxeles)



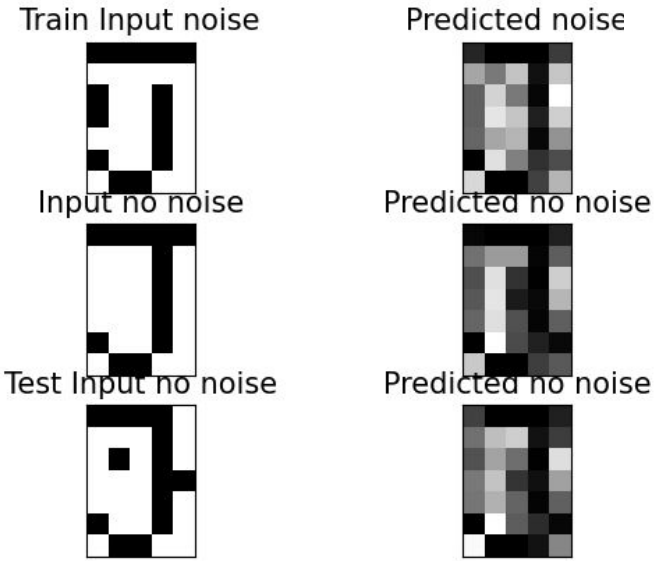


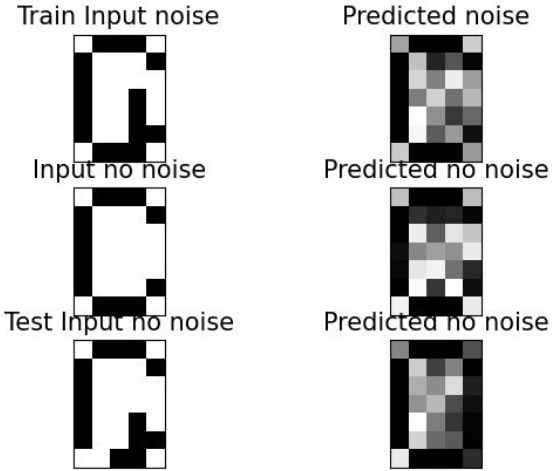


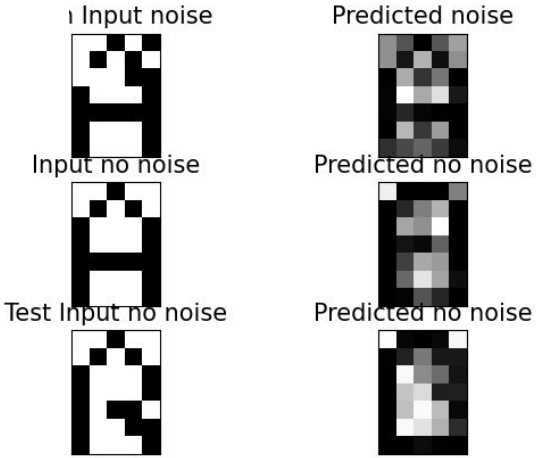




Resultados (3 píxeles)





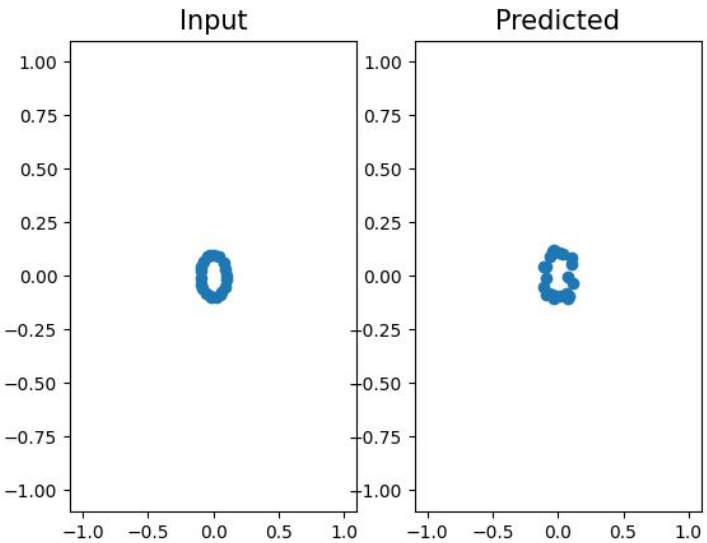


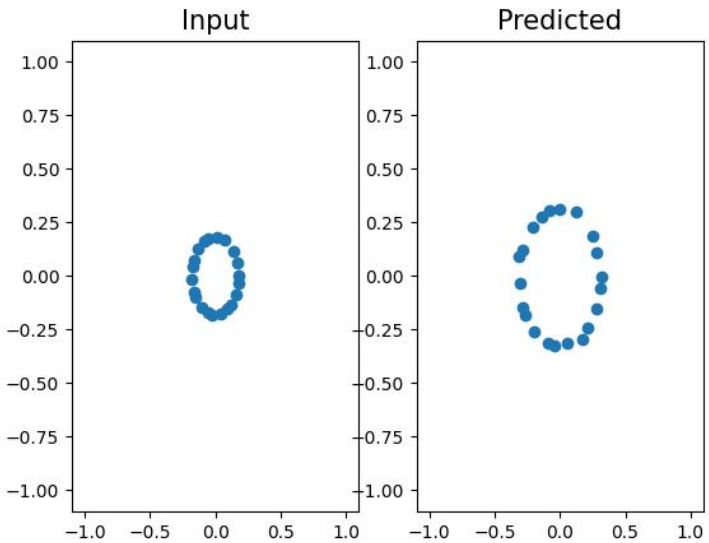
Capacidad generativa

- Se decidió entrenar el autoencoder con circunferencias con centro en (0, 0)
- Se utilizaron 100 círculos variando el radio entre 0.1 y 0.9 con 21 puntos, ambos variados equidistantes, de dataset de entrenamiento para la red.
- La arquitectura utilizada fue de [42, 25, 15, 9, 3, 2, 3, 9, 15, 25, 42]
- Con 1000 iteraciones y un eta de 0.01

Resultados

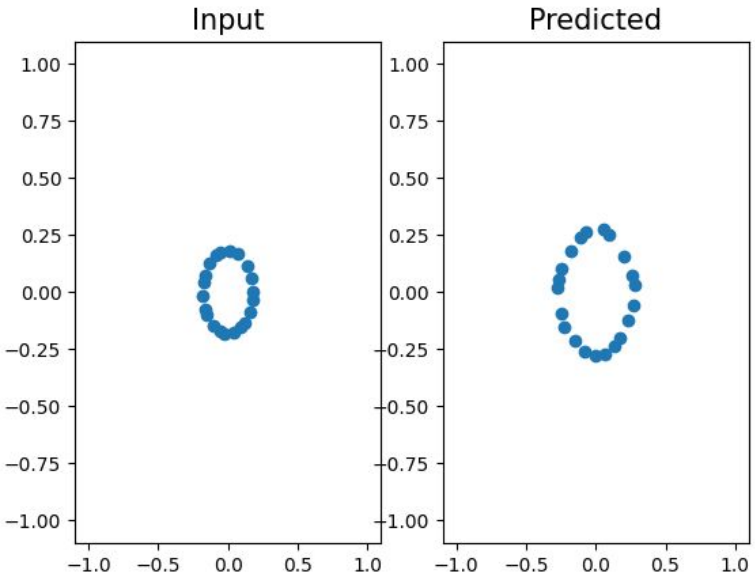
Comparación de entrada con predicción

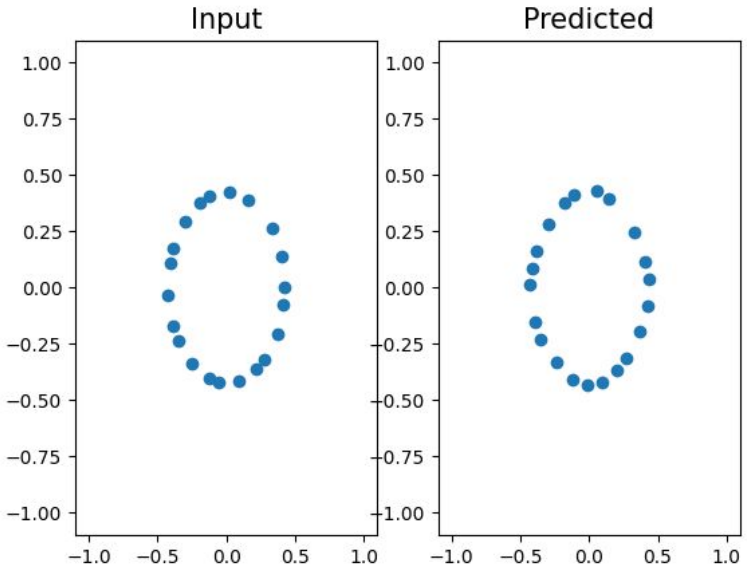




Resultados

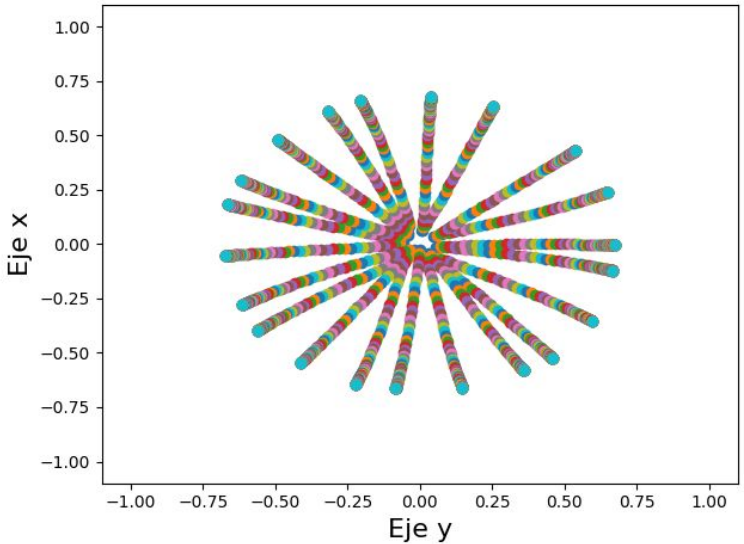
Comparación de entrada con predicción

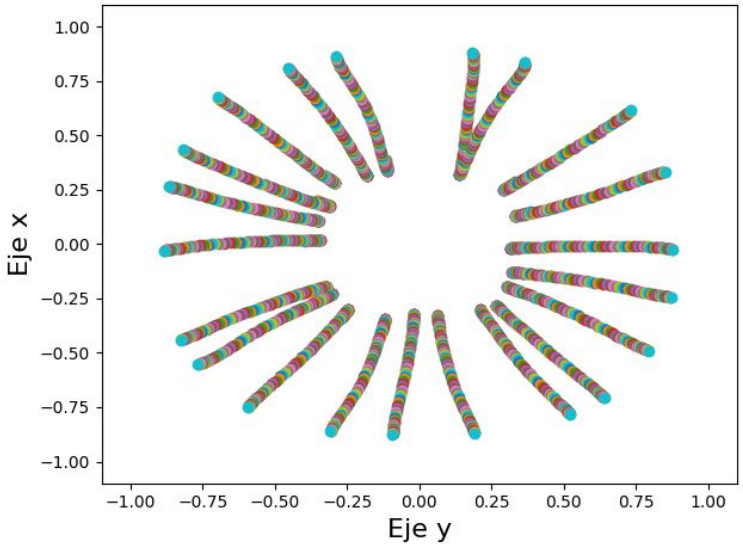




Resultados

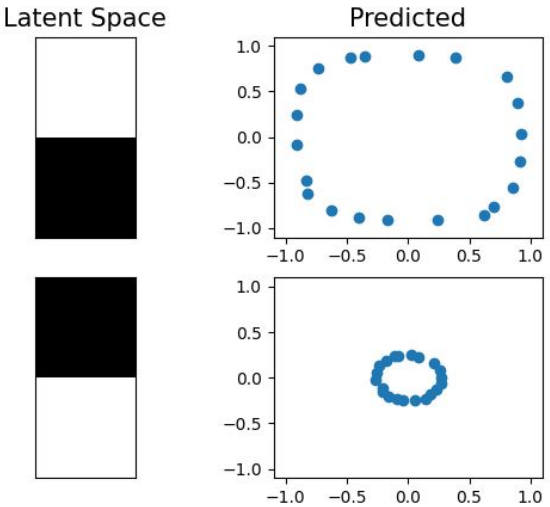
Comparación de predicciones con todas las entradas

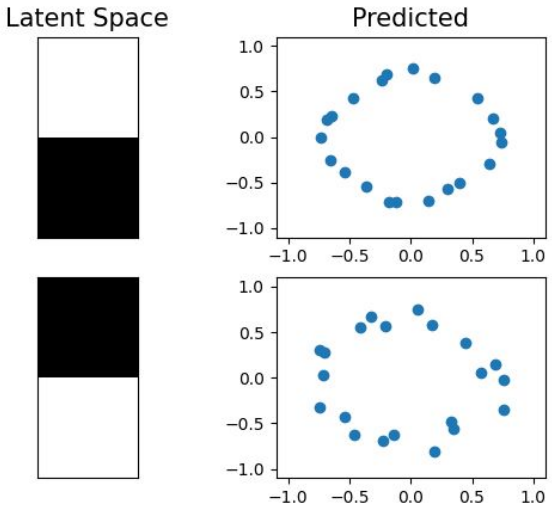




Resultados

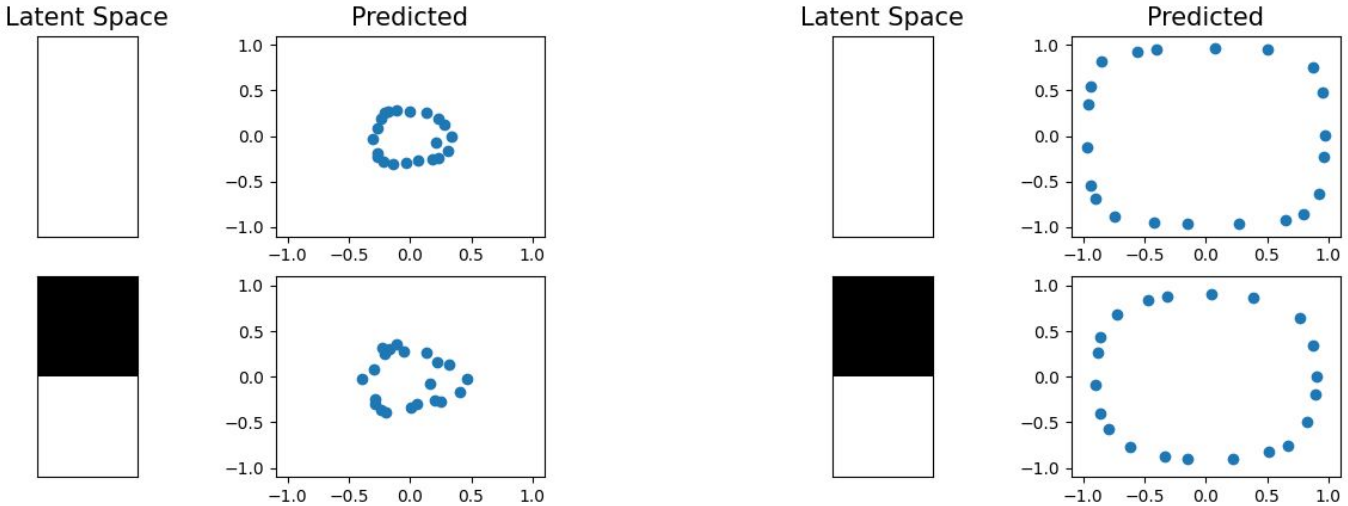
Generando nuevos círculos a partir del espacio latente





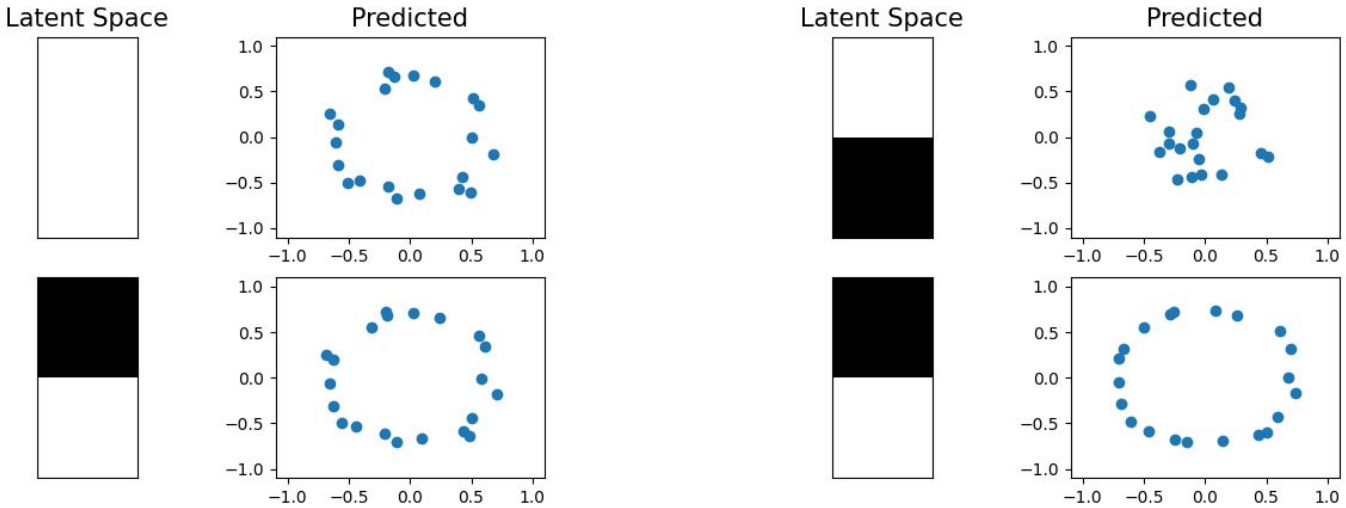
Resultados

Generando nuevos círculos a partir del espacio latente



Resultados

Generando nuevos círculos a partir del espacio latente



¡Muchas gracias por su atención!