Grupo 6

Luciana Diaz Kralj
Gonzalo Nicolás Rossin
João Nuno Diegues Vasconcelos
Mafalda Colaço Parente Morais Da Costa

TP3
Deep Learning

72.27 Sistemas de Inteligencia Artificial

1 Auto encoders

A1
Arquitectura

Resultados en función de las épocas

Resultado esperado

Resultado 15.000 épocas

Resultado 50.000 épocas

Resultado 100.000 épocas

Resultado 500.000 épocas

Arquitectura [17, 2, 17]

Resultados en función de la arquitectura

Resultado esperado

Resultado [35, 17, 2, 17, 35]

Resultado [35, 25, 17, 2, 17, 25, 35]

Resultado [35, 2, 35]

Resultado [35, 20, 10, 2, 25, 35]

1.000.000 épocas

A2
Optimizaciones

Aplicando optimizadores

Resultado esperado

Resultado sin ningún optimizador

Resultado con momentum + η adaptativo

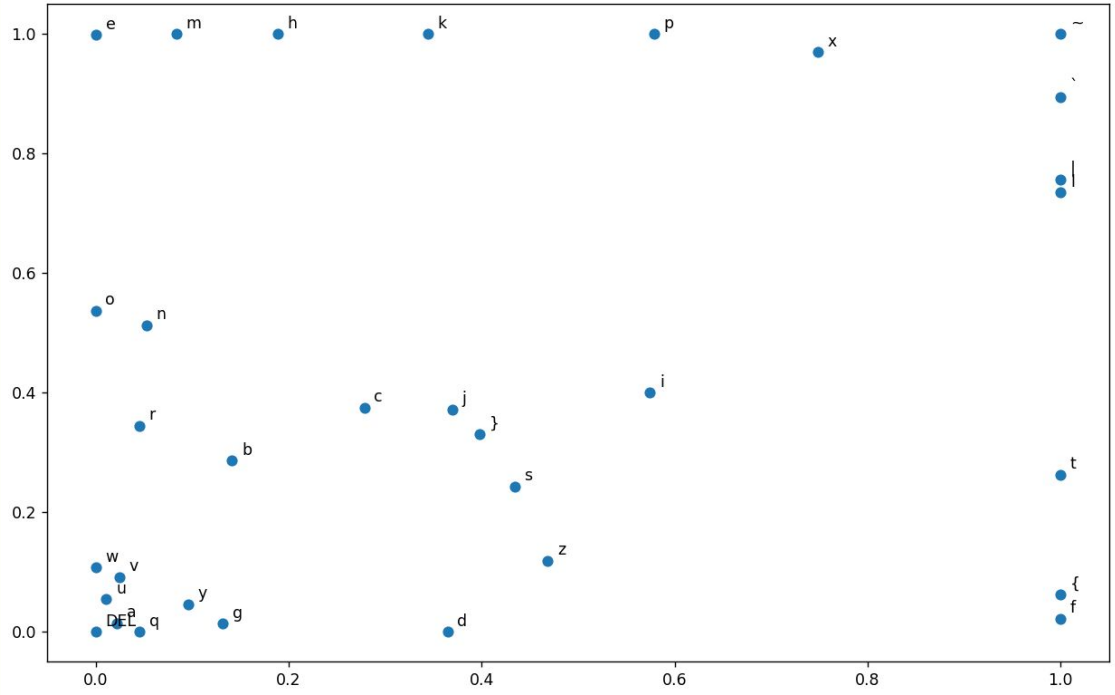
Resultado con Adam

100.000 épocas || Arquitectura [35, 25, 17, 2, 17, 25, 35]

A3

Espacio latente

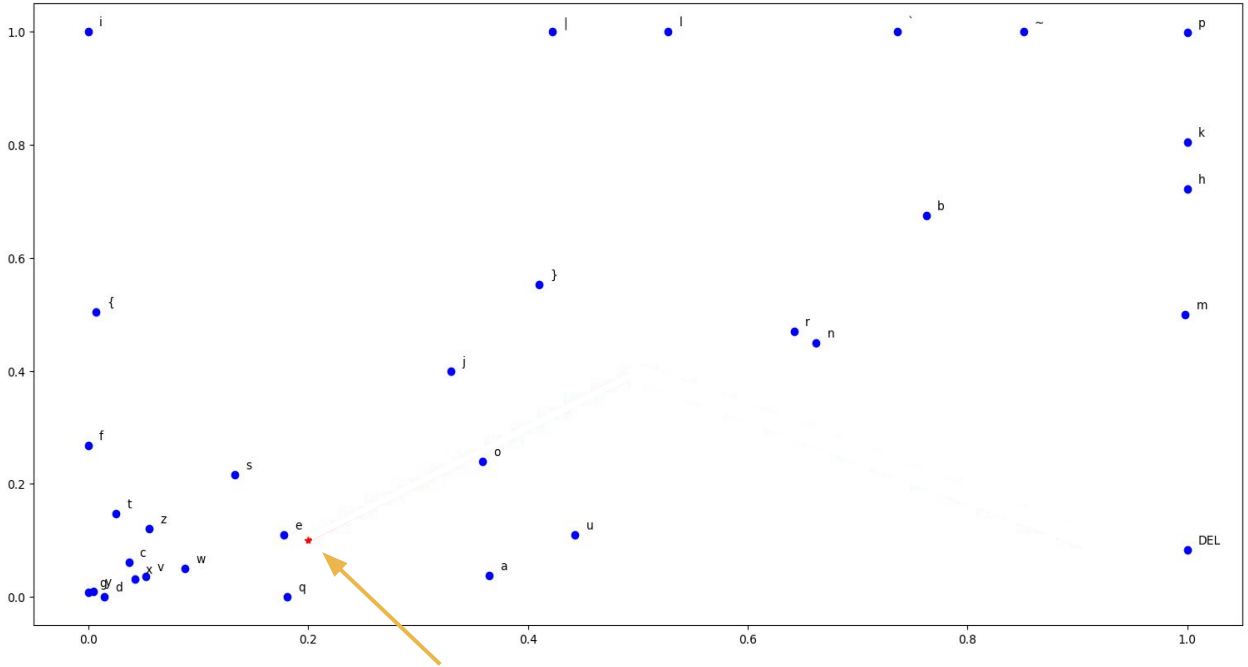
Espacio Latente

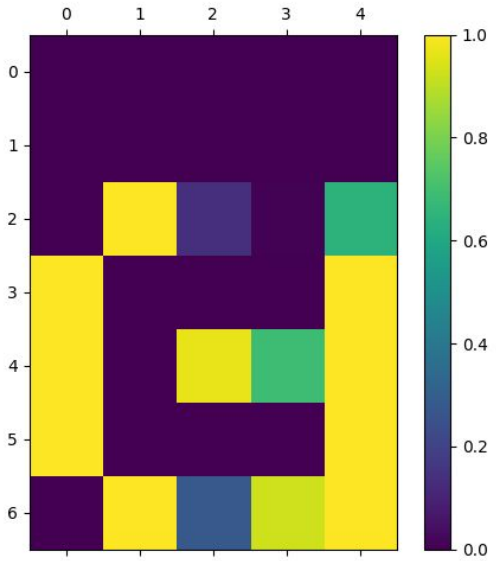


[17, 2, 17] || 500.000 épocas || η = 0.0001

A4
Generación

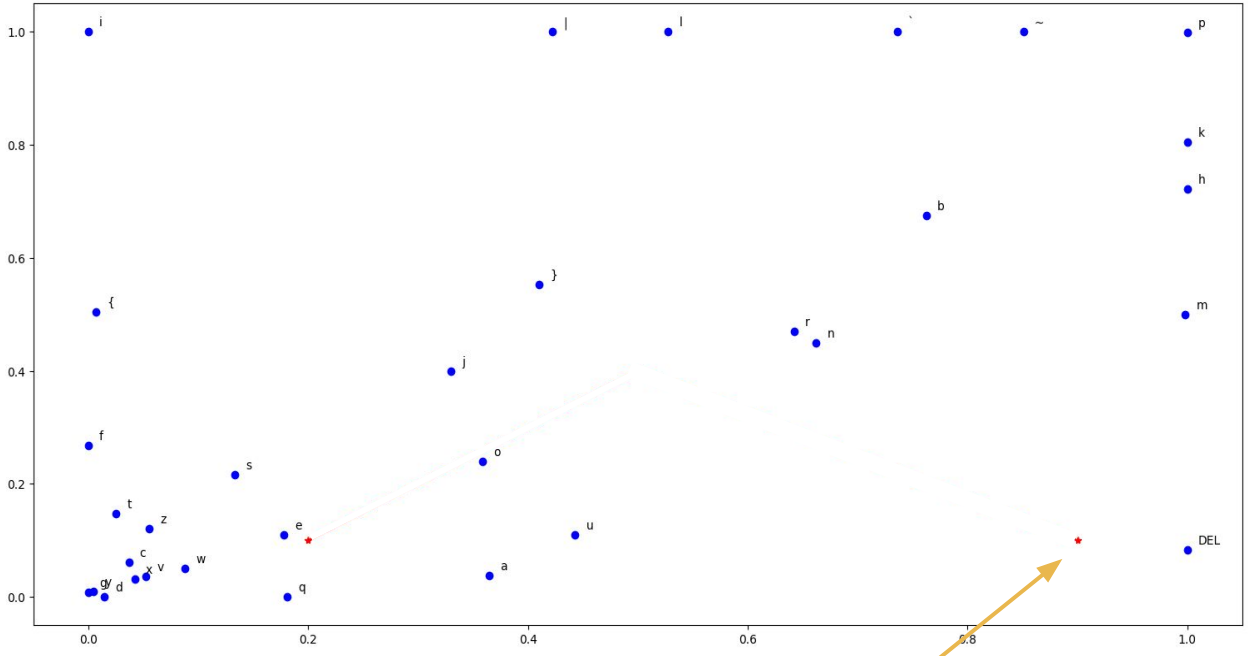
Nueva entrada [17, 2, 17] || 500.000 épocas || η = 0.0001

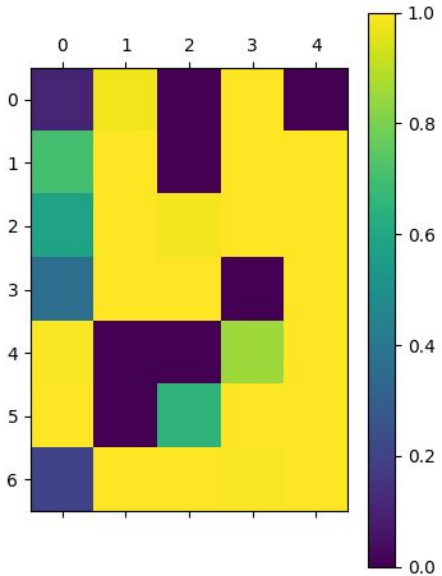




Entrada [0.2, 0.1]

Nueva entrada [17, 2, 17] || 500.000 épocas || η = 0.0001





Entrada [0.9, 0.1]

B1&2
Denoising

Binary Noise and Distribution Noise

Generación de ruido

Tenemos dos funciones diferentes para generar ruido:

1° Cambiar los valores de 0 a 1 y de 1 a 0 según cierta probabilidad.

	Original	Noise
1º	1	0
2º	0	0
3º	1	0
4º	0	0

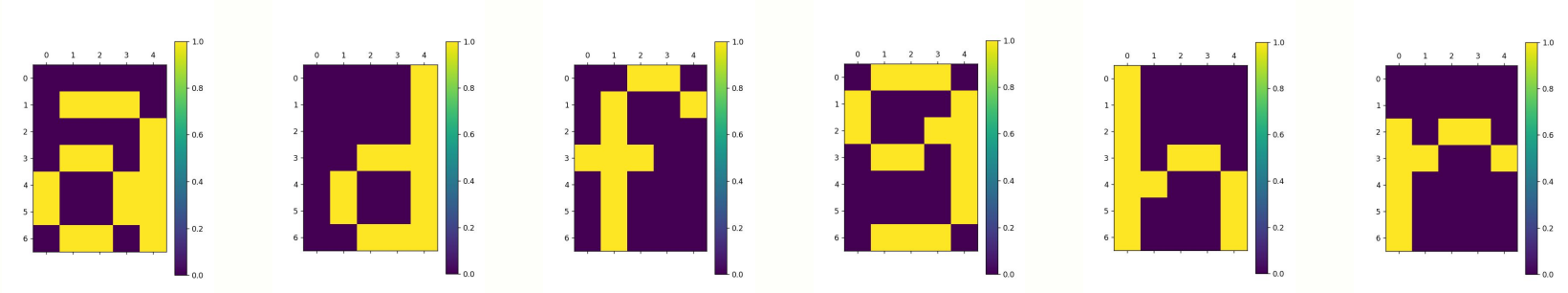
Generación de ruido

Tenemos dos funciones diferentes para generar ruido:

2° Añadimos incrementos de 0,1 o más a los valores según cierta probabilidad.

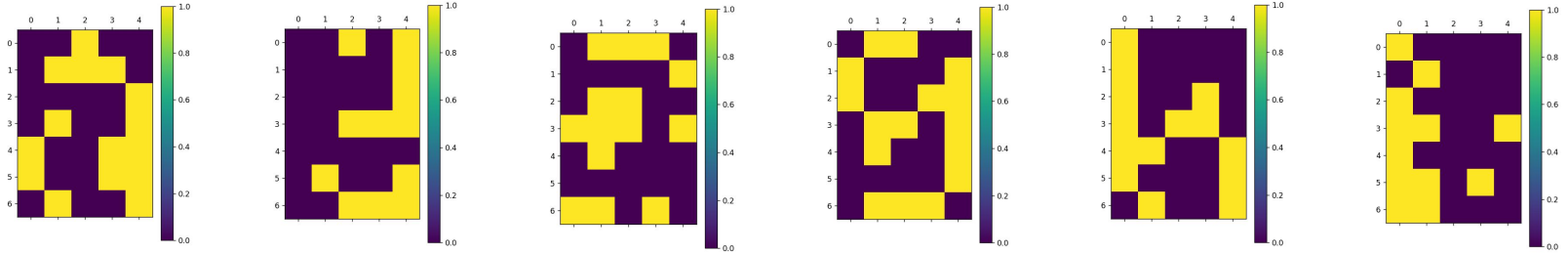
	Original	Noise
1º	1	0.2
2º	0	0.3
3º	1	1
4º	0	0.1

Caracteres esperados

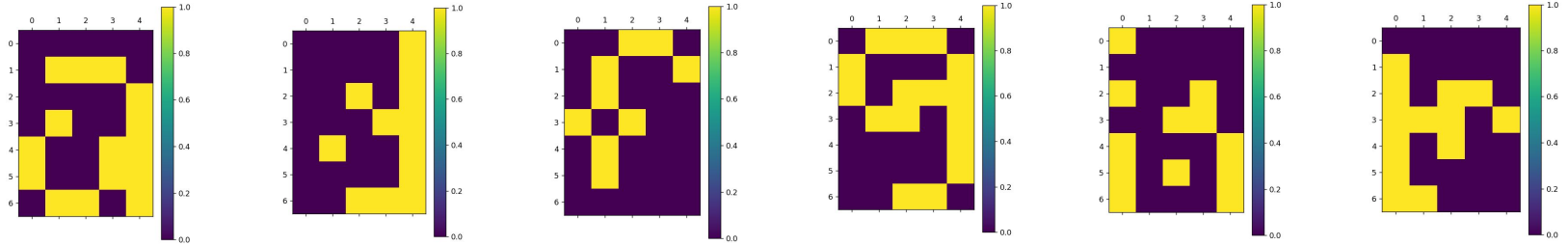


Caracteres con 10% de Ruido Binario

Modelo con un red de 3 layers: 17-8-2-8-17



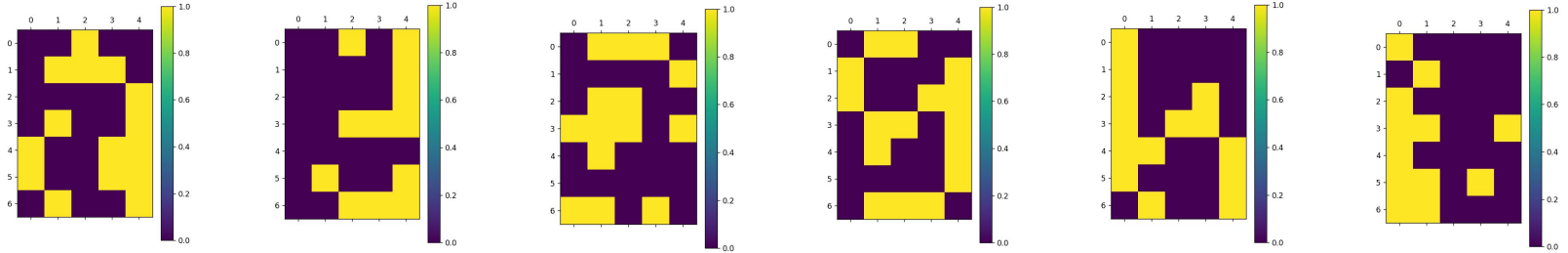
Caracteres antes del entrenamiento después de introducir 10% de ruido binario



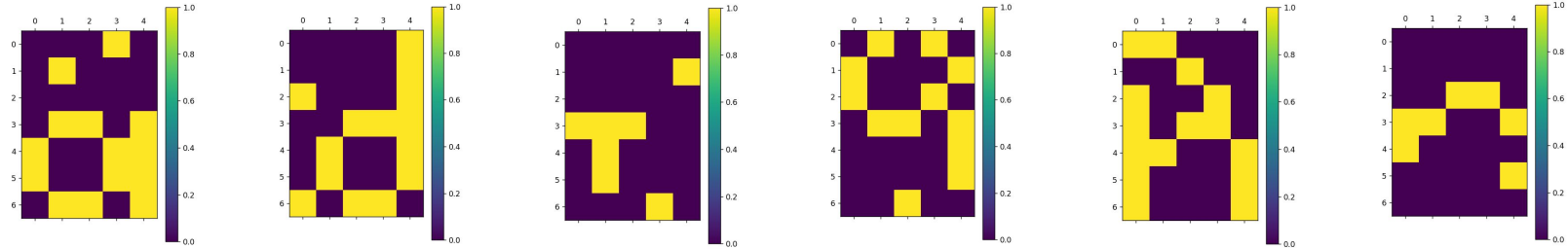
Caracteres después del entrenamiento

Caracteres con 10% de Ruido Binario

Modelo con un red de 4 layers: 35-30-20-2-20-30-35



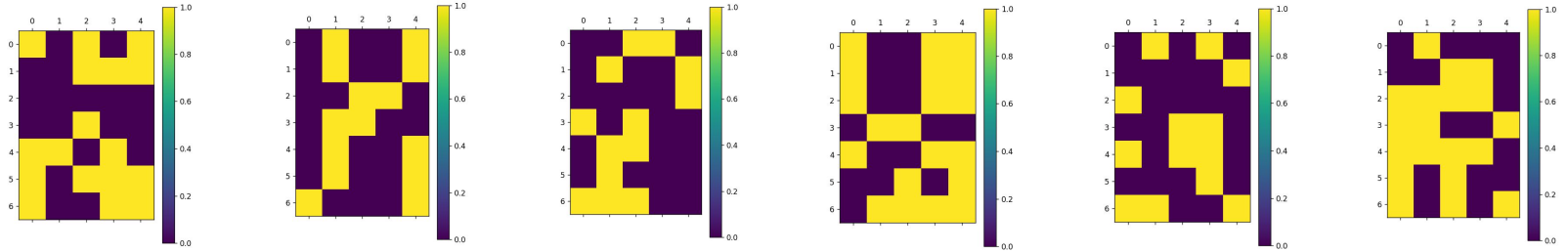
Caracteres antes del entrenamiento después de introducir 10% de ruido binario



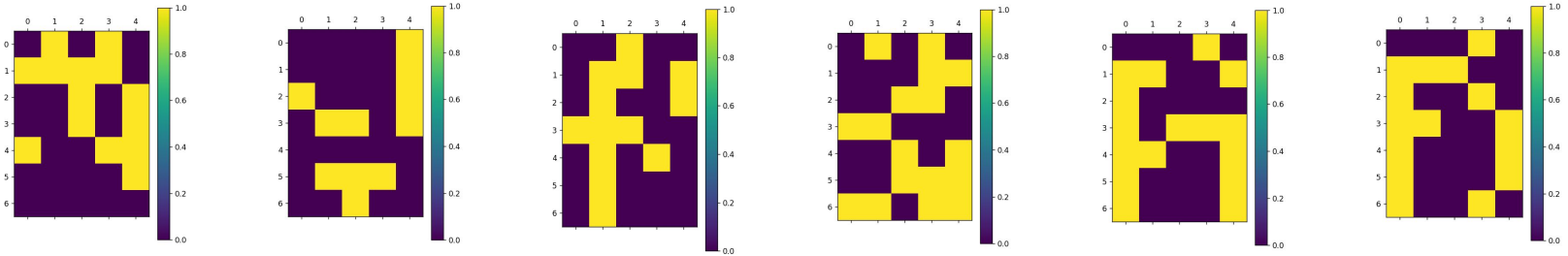
Caracteres después del entrenamiento

Caracteres con 30% de Ruido Binario

Modelo con un red de 3 layers: 17-8-2-8-17



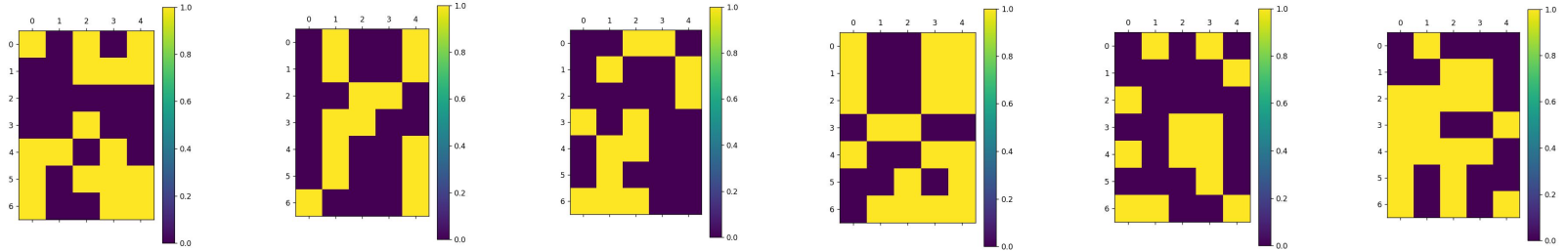
Caracteres antes del entrenamiento después de introducir 30% de ruido binario



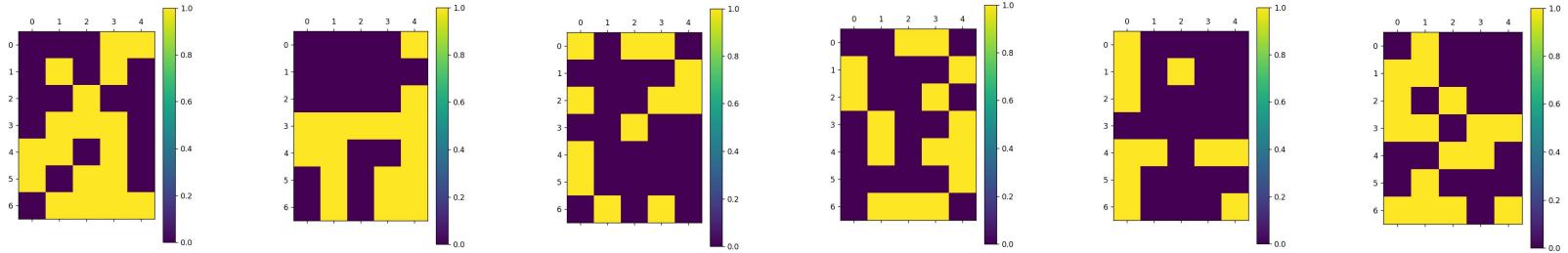
Caracteres después del entrenamiento

Caracteres con 30% de Ruido Binario

Modelo con un red de 4 layers: 35-30-20-2-20-30-35



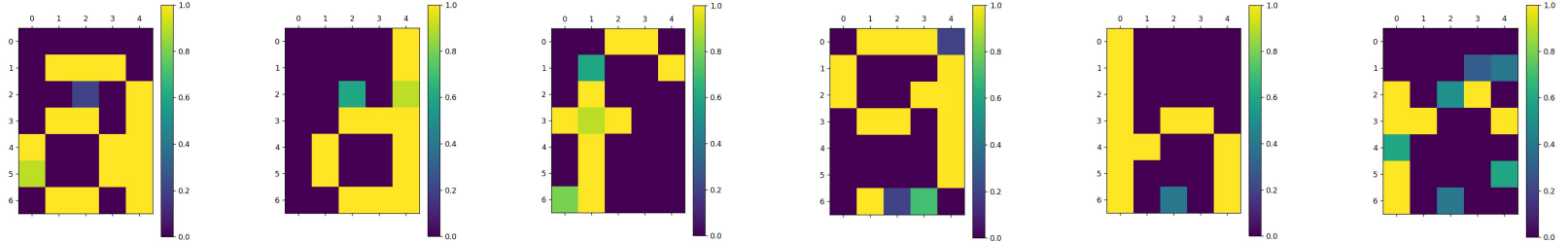
Caracteres antes del entrenamiento después de introducir 30% de ruido binario



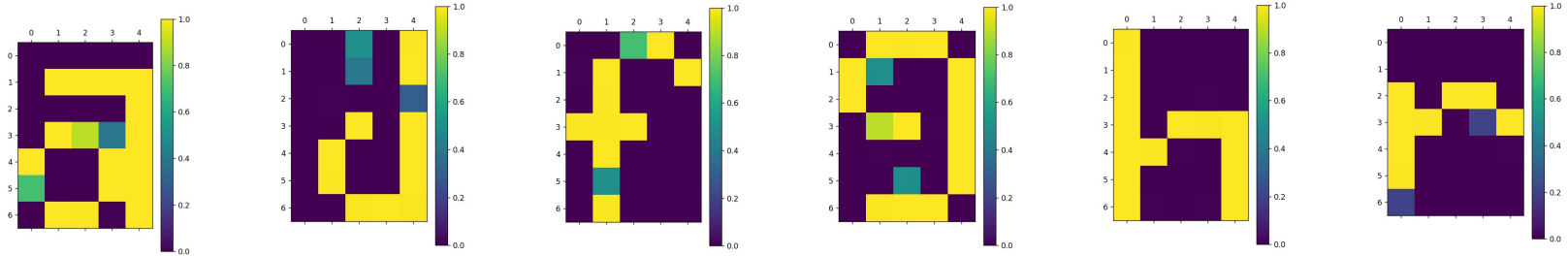
Caracteres después del entrenamiento

Caracteres con 10% de Ruido Decimal

Modelo con un red de 3 layers: 17-8-2-8-17



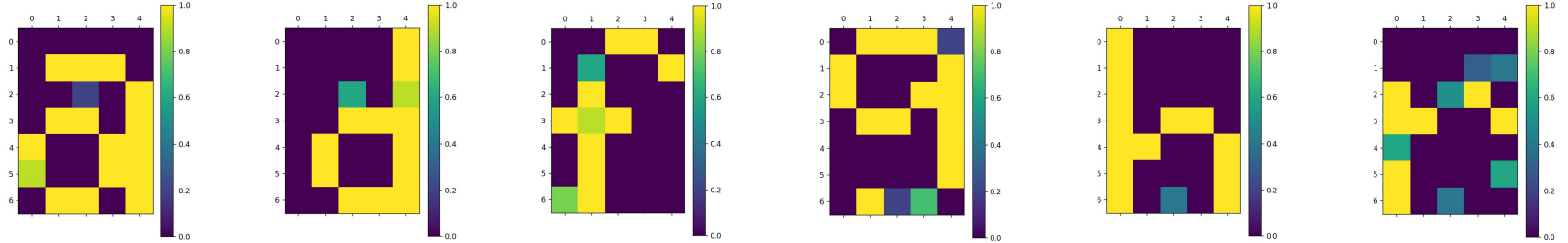
Caracteres antes del entrenamiento después de introducir 10% de ruido decimal



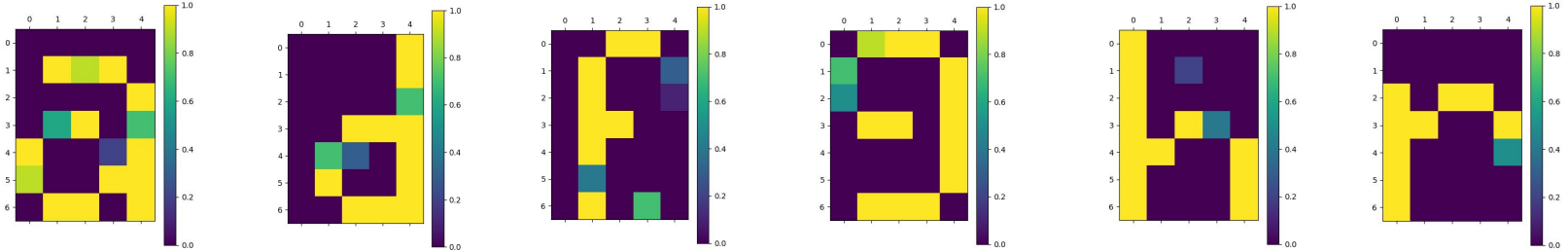
Caracteres después del entrenamiento

Caracteres con 10% de Ruido Decimal

Modelo con un red de 4 layers: 35-30-20-2-20-30-35



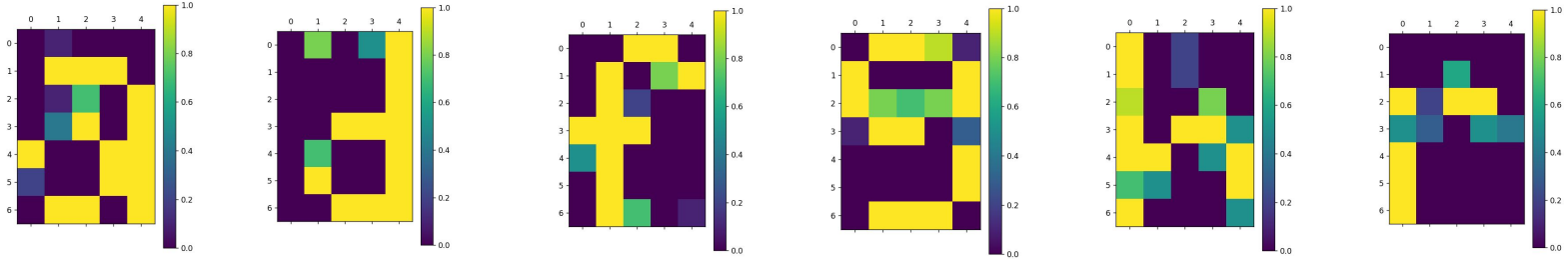
Caracteres antes del entrenamiento después de introducir 10% de ruido decimal



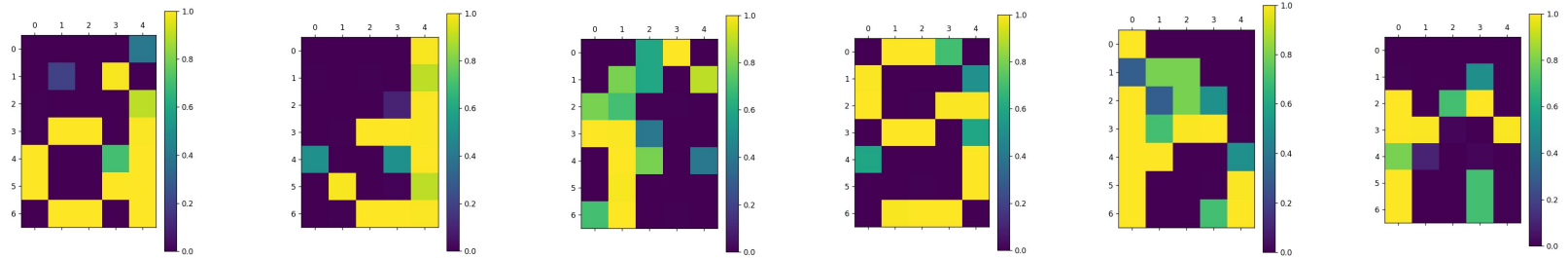
Caracteres después del entrenamiento

Caracteres con 30% de Ruido Decimal

Modelo con un red de 3 layers: 17-8-2-8-17



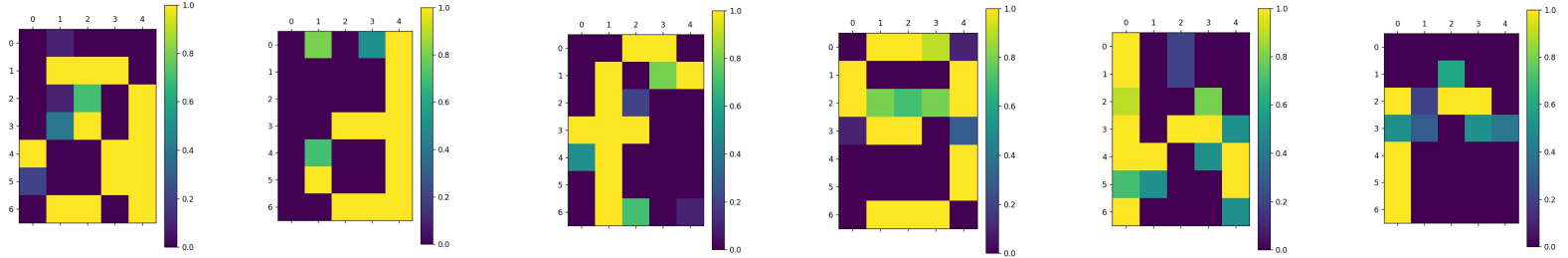
Caracteres antes del entrenamiento después de introducir 30% de ruido decimal



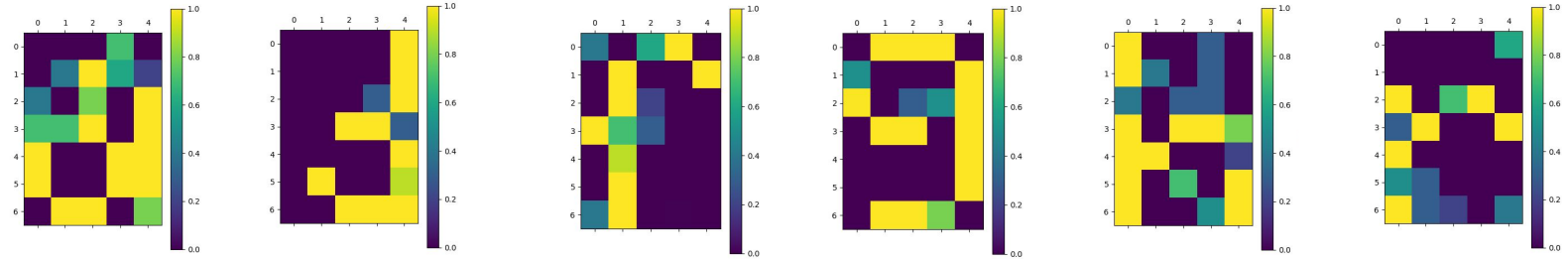
Caracteres después del entrenamiento

Caracteres con 30% de Ruido Decimal

Modelo con un red de 4 layers: 35-30-20-2-20-30-35



Caracteres antes del entrenamiento después de introducir 30% de ruido decimal

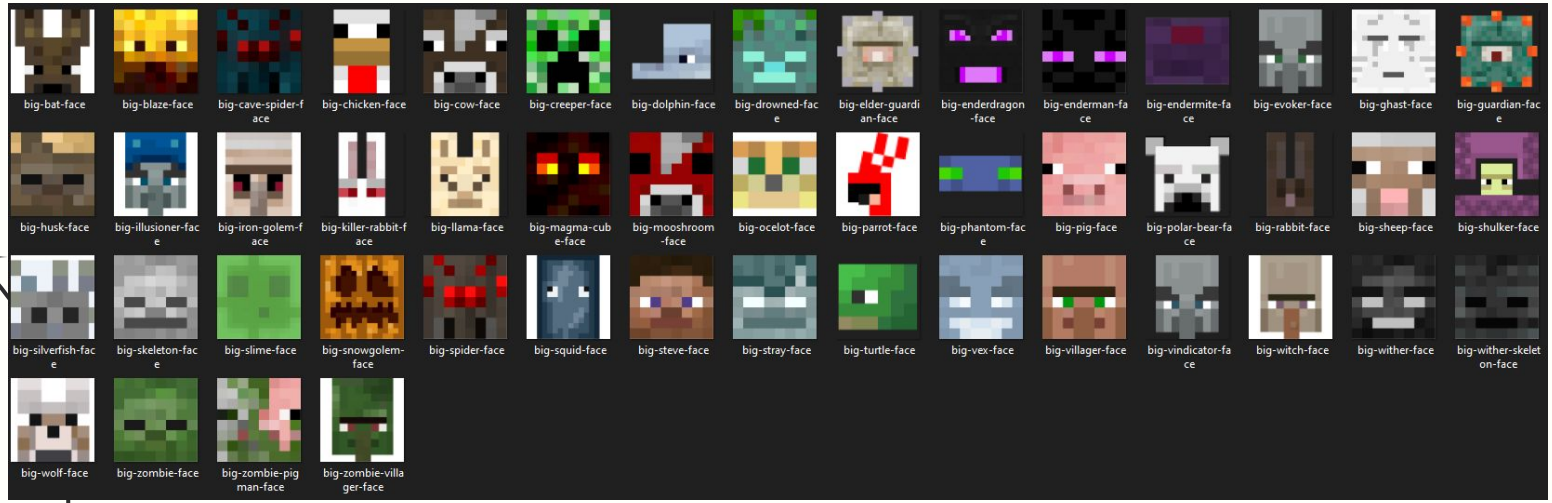


Caracteres después del entrenamiento

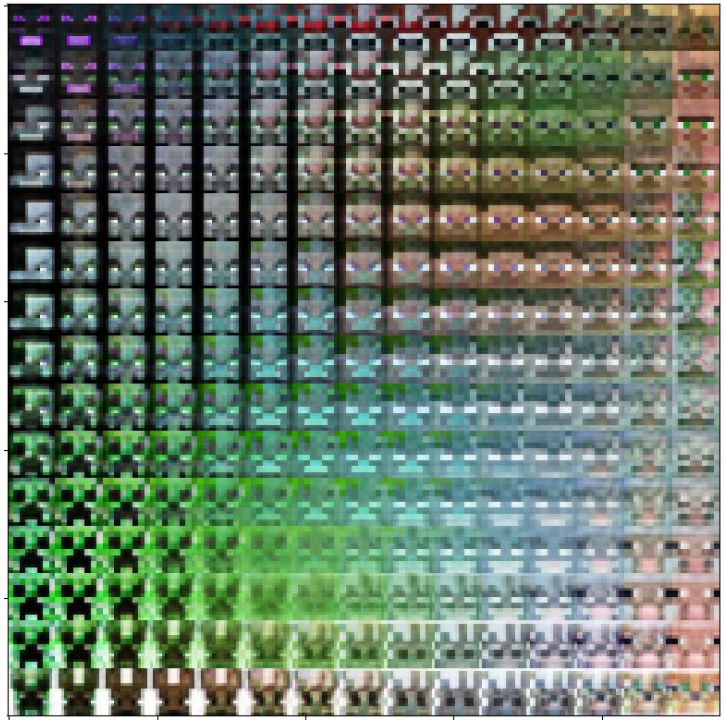
2

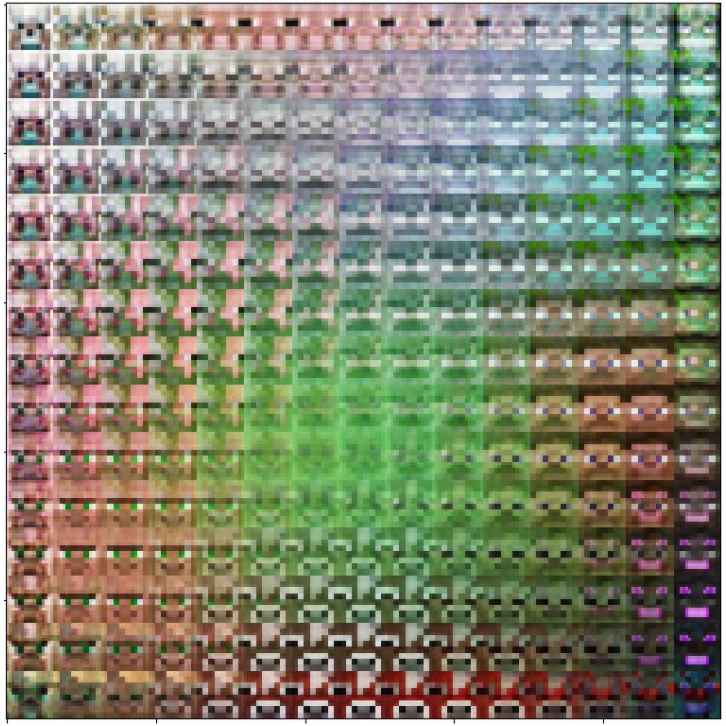
Variational autoencoder

Dataset: caras de Minecraft

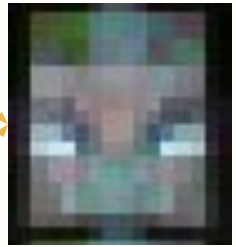


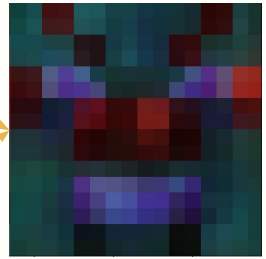
Espacio latente





Caras que no existían





Dataset: caras de personas

Caras que no existían

=

Conclusiones

- Se pudo ver una diferencia importante entre escalar gradualmente la dimensionalidad del input y comprimirlo abruptamente.

- Es curioso que a pesar de elegir un punto alejado del espacio latente, el autoencoder intenta generar a la salida un carácter parecido al carácter más cercano que sí conoce.

- El ruido decimal tiene mejores resultados porque el cambio es menos brusco. Como se puede observar con el ruido binario para obtener resultados aceptables el porcentaje de ruido a aplicar tiene que ser muy bajo. En general consideramos que la red de 3 layers tiene una mejor aprendizaje.

Conclusiones

- Observamos que si las dimensiones de las imágenes eran muy grandes, rápidamente ocurría un overflow, con lo cual era importante reducir la dimensionalidad del problema para un mejor procesamiento.

- Por otra parte, observamos que en ciertos casos, lo que aprendía la red era una distribución que permitía un morphing entre imágenes similar a desvanecer una imagen sobre la otra.