

SISTEMA DE ENCRYPTACIÓN UTILIZANDO REDES GAN Y AUTOENCODERS

EVALUACIÓN APRENDIZAJE PROFUNDO I Y II

FELIPE HIGÓN MARTÍNEZ

felipe.higon@gmail.com
fehimar@alumni.uv.es

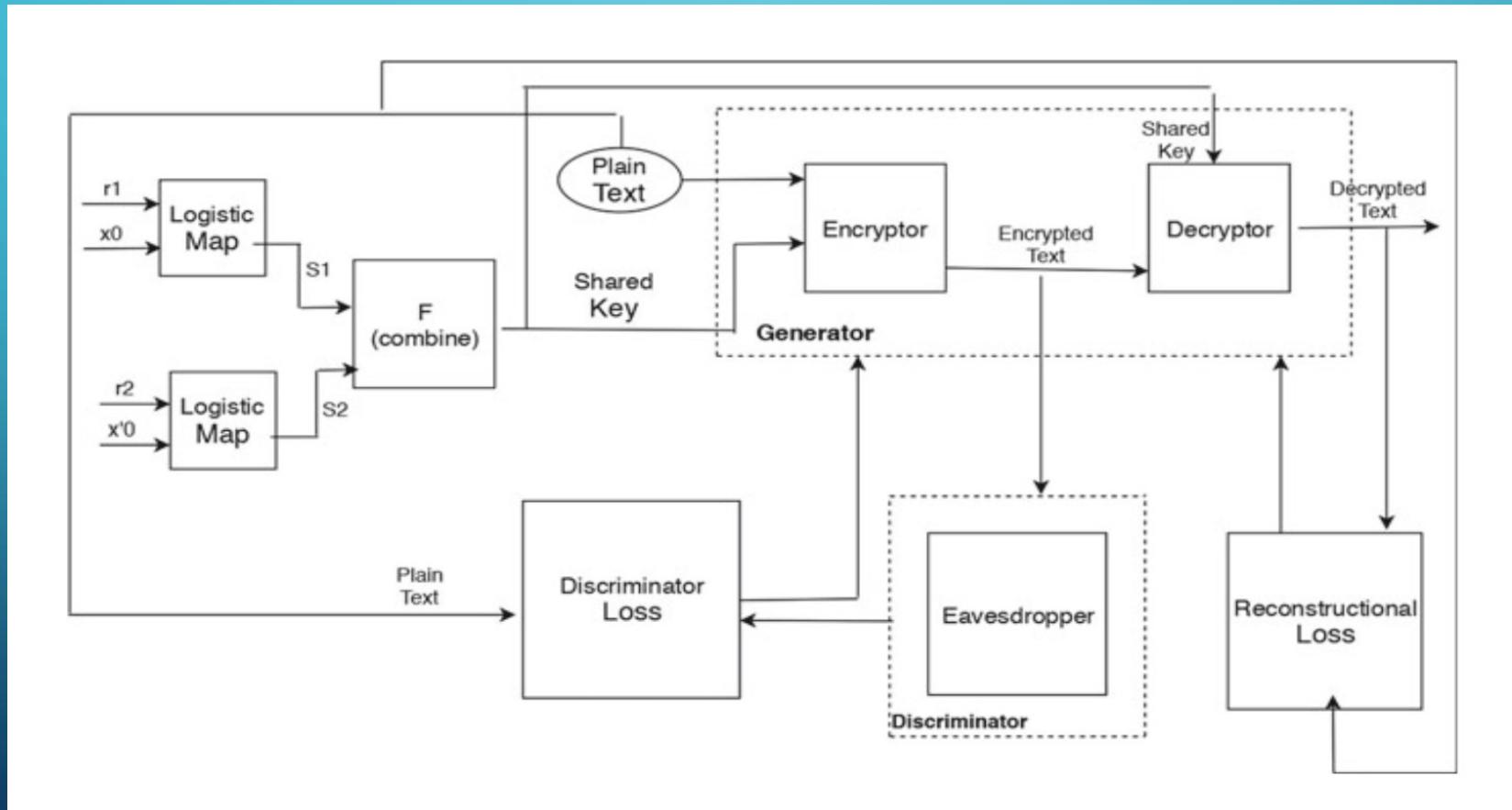
https://github.com/felipehigon/IA3_AP1_2_Encryption_GAN

CONTENIDO

- PLANTEAMIENTO DEL PROBLEMA
- SOLUCIÓN INICIAL ADOPTADA
 - Modelo GAN como generador de números.
 - Modelo AUTOENCODER como sistema de encriptación.
 - Modelo CONVOLUCIONAL como decodificador de las imágenes.
 - Modelo CONVOLUCIONAL como posible hacker al sistema.
 - Funciones Transmisión/Recepción texto.
 - Resultados.
- SOLUCIÓN FINAL ADOPTADA
 - Entrenamiento de los modelos en conjunto.
 - Resultados.
- CONCLUSIONES

PLANTEAMIENTO PROBLEMA

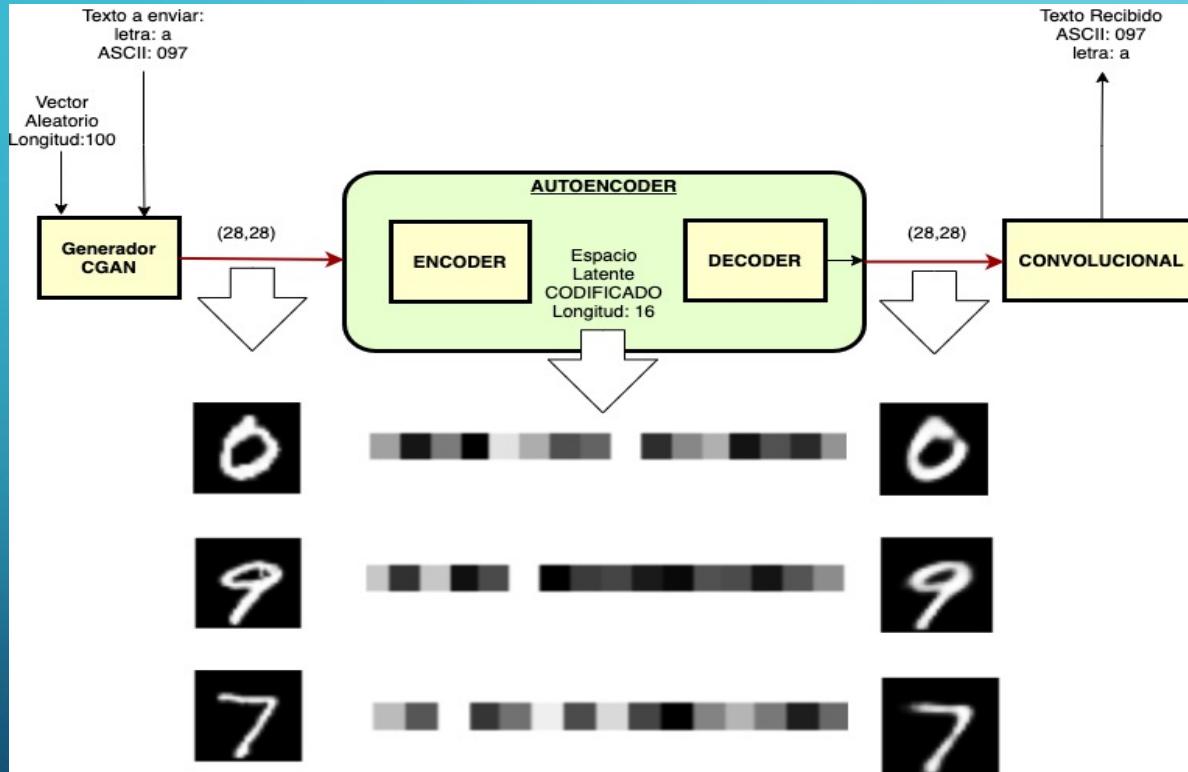
- Paper: **Chaos Theory on Generative Adversarial Networks for Encryption and Decryption of Data.** <https://www.researchgate.net/publication/336925933>



PLANTEAMIENTO PROBLEMA

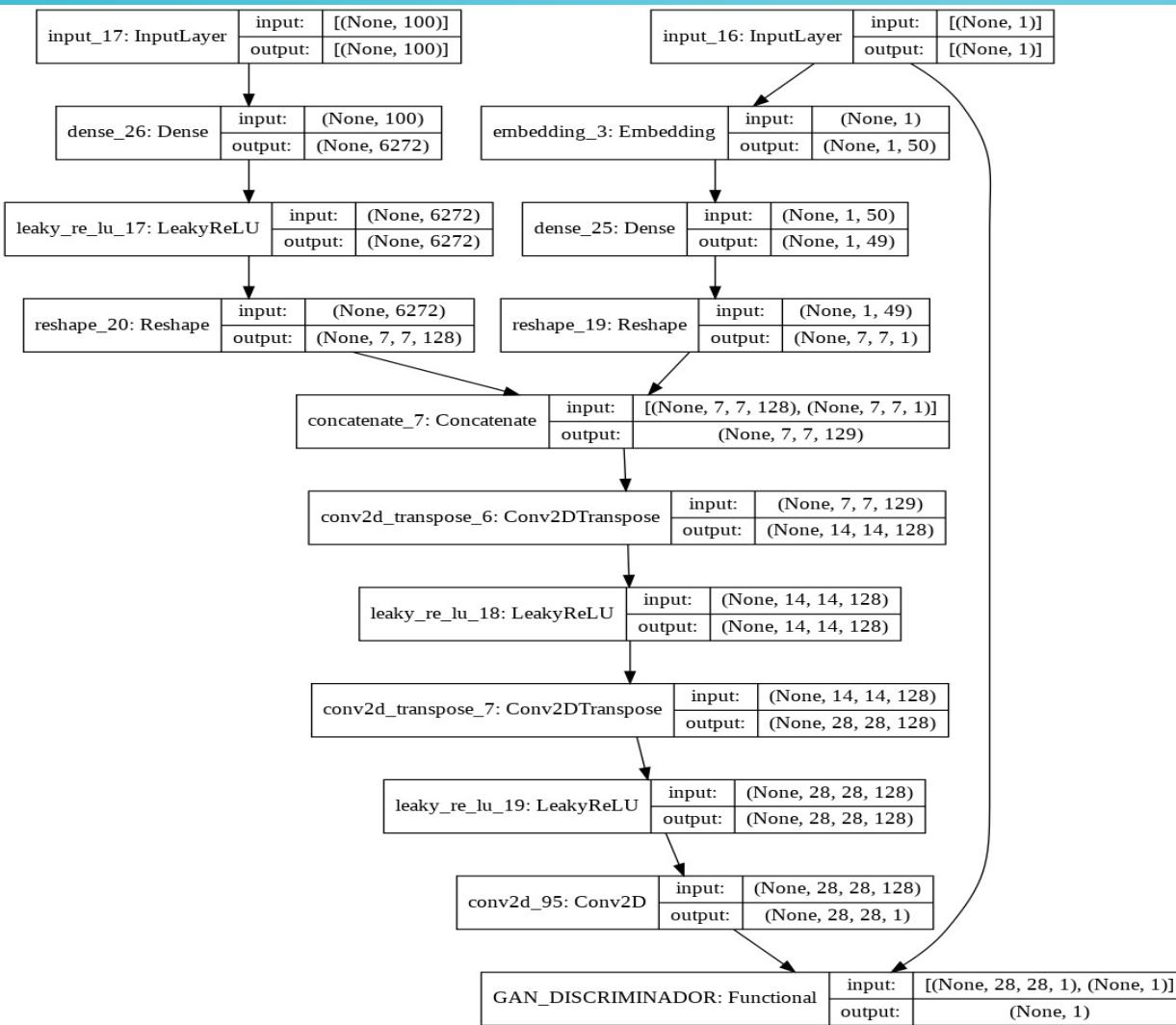
- Generar chat encriptado “Sin Clave de Encriptación”.
- El texto se pasa a numérico con código ASCII.
- Usar modelo GAN con el dataset MNIST números como generador de números.
- Usar modelo AUTOENCODER para generar espacio latente para transmitir.
- Usar modelo CONVOLUCIONAL para pasar de imagen a texto.
- Realizar simulaciones y generar datasets encriptados para con otro modelo convolucional intentar hackear la conexión.

SOLUCIÓN INICIAL ADOPTADA

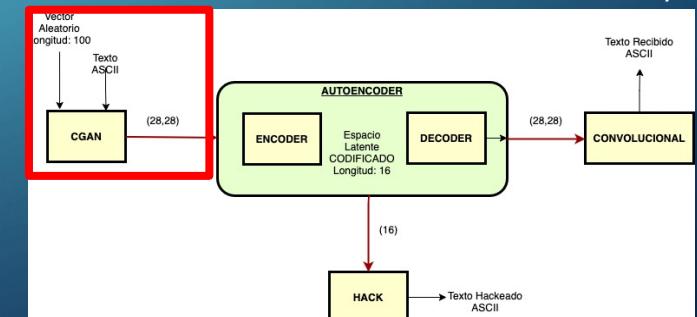


- Generar un espacio latente codificado lo mas caótico posible.
- Evitar usar “claves de encriptación”.
- Conseguir que el modelo genere su propia encriptación.

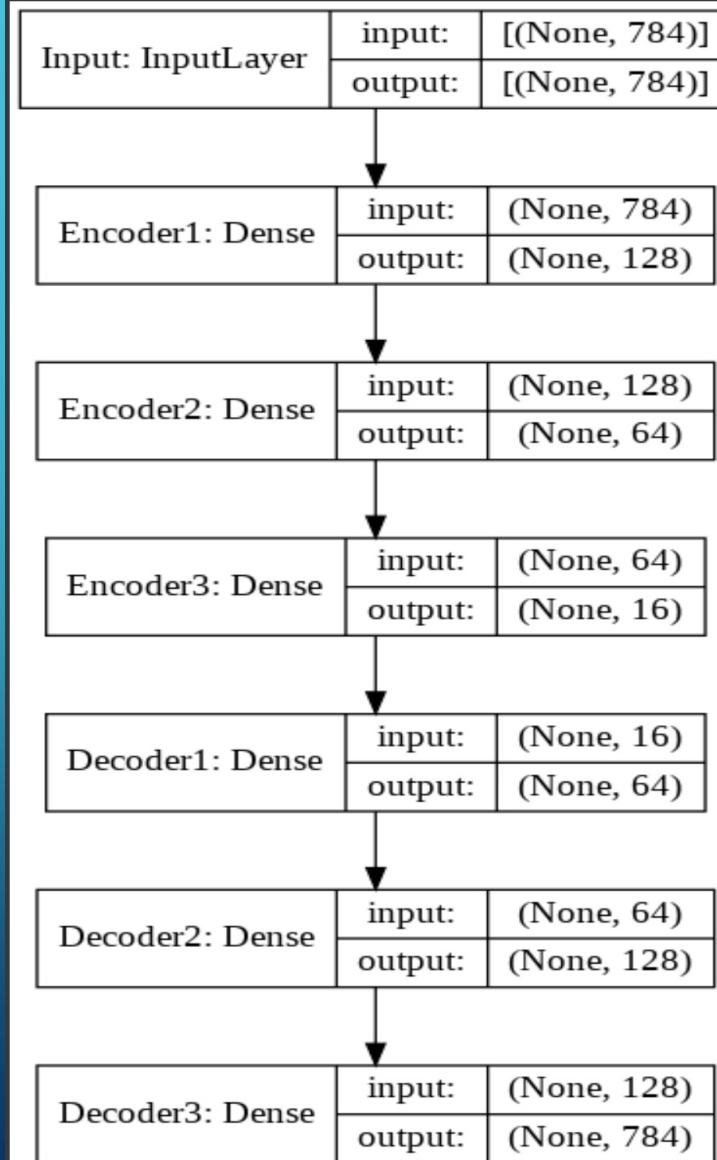
GAN



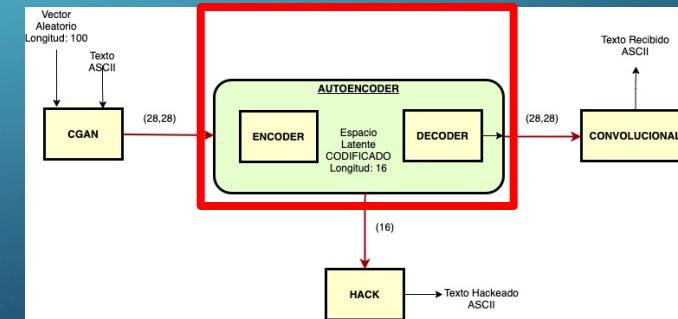
- Entradas: vector aleatorio de dimensión 100 y numero de 0 a 9.
- Se entrena con dataset MNIST.
- Salida imagen de 28x28 (784) con cierta variabilidad para el autoencoder.
- Para este proyecto solo usare el generador GAN.
- Parámetros: 1169336
- Modelo esta basado en:
<https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/>



AUTOENCODER

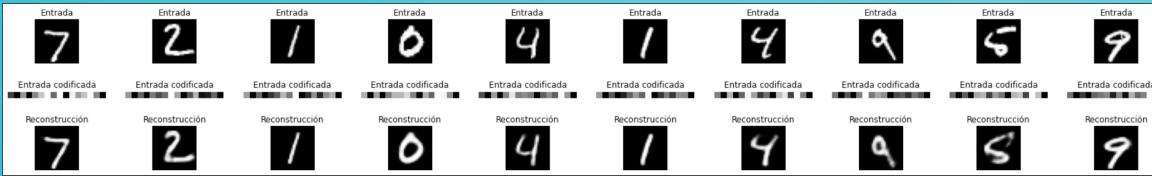


- Entradas: Imagen del generador GAN (784)
- Se entrena con dataset MNIST.
- Encoder: espacio latente de dimensión 16.
- Salida del encoder texto codificado, transmite por internet.
- Decoder: recibe vector de 16 y vuelve a generar imagen de dimensión 784.
- Conseguir que este espacio latente sea lo mas caótico posible.
- Se generan dos autoencoders y los encoders y decoders por separado para poder obtener el espacio latente.
- Parámetros: 220320

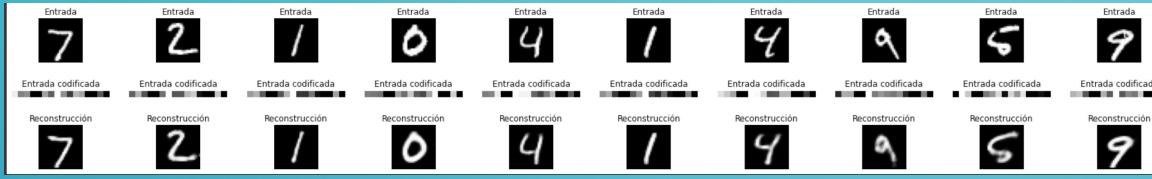


AUTOENCODER

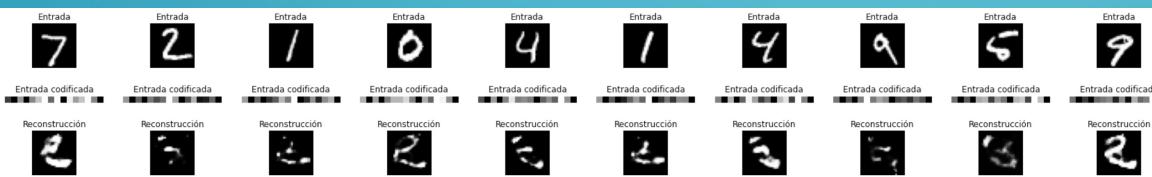
Encoder 1 – Decoder 1



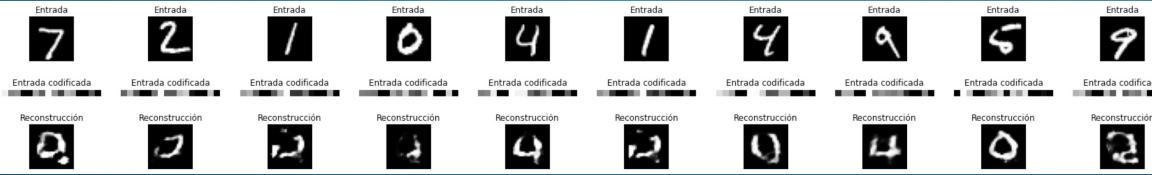
Encoder 2 – Decoder 2



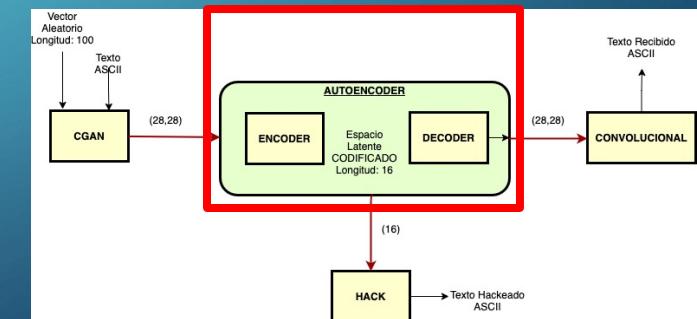
Encoder 1 – Decoder 2



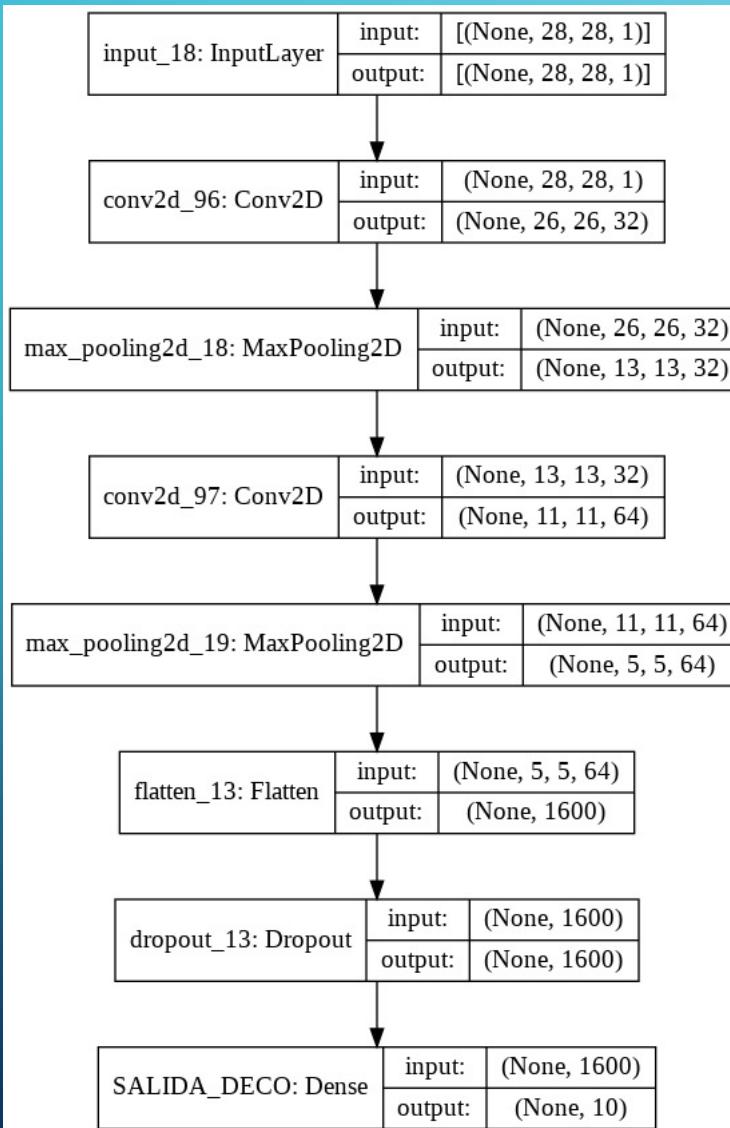
Encoder 2 – Decoder 1



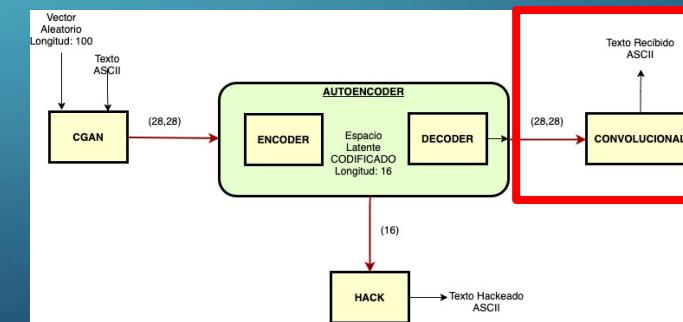
- Se comprueba que el espacio latente es único a cada encoder.
- Inicialización de pesos es aleatoria.



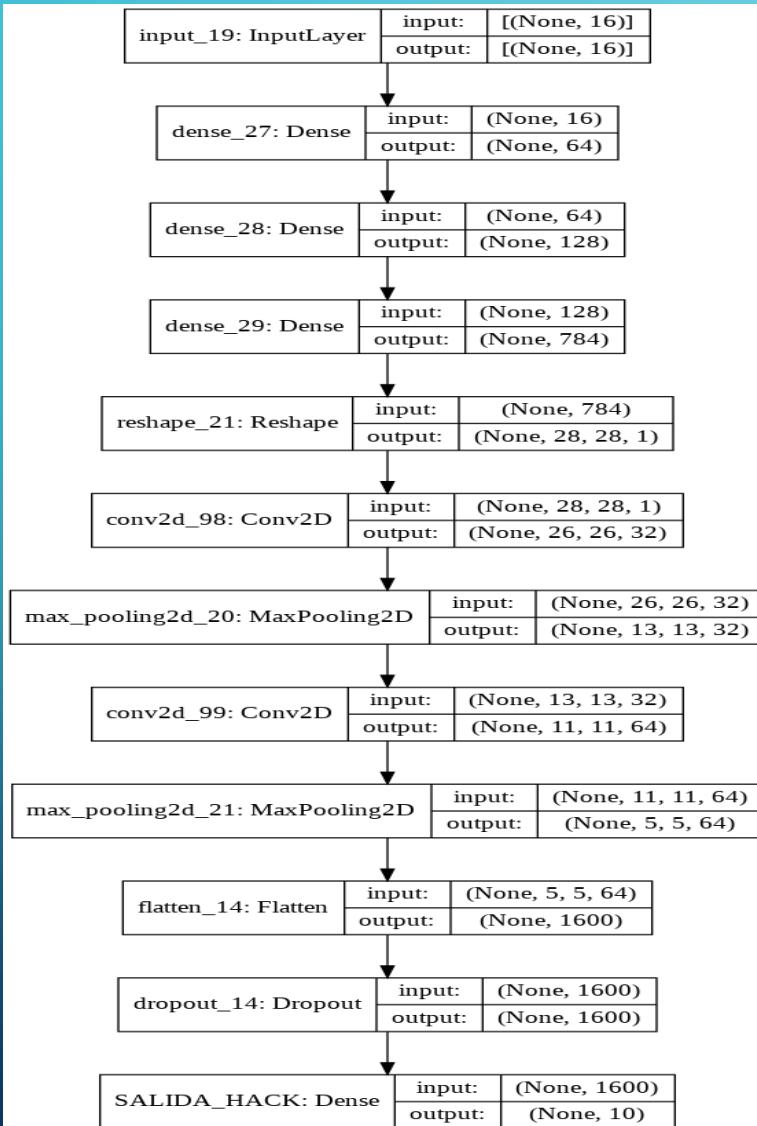
CONVOLUCIONAL



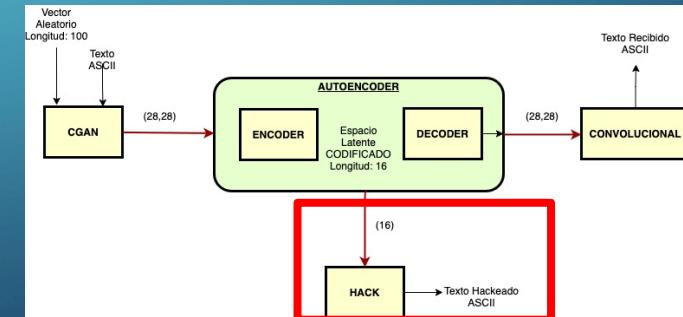
- Entradas: Imagen del decoder (28x28)
- Se entrena con dataset MNIST.
- Salida son los números del 0 al 9.
- Posteriormente se pasan de ASCII a texto.
- Resultados en test:
 - Test loss: 0.024375958368182182
 - Test accuracy: 0.9926000237464905
- Resultados del decoder (prueba de 10 números):
 - Test loss: 0.0027047712355852127
 - Test accuracy: 1.0
- Parámetros: 34826



CONVOLUCIONAL HACK

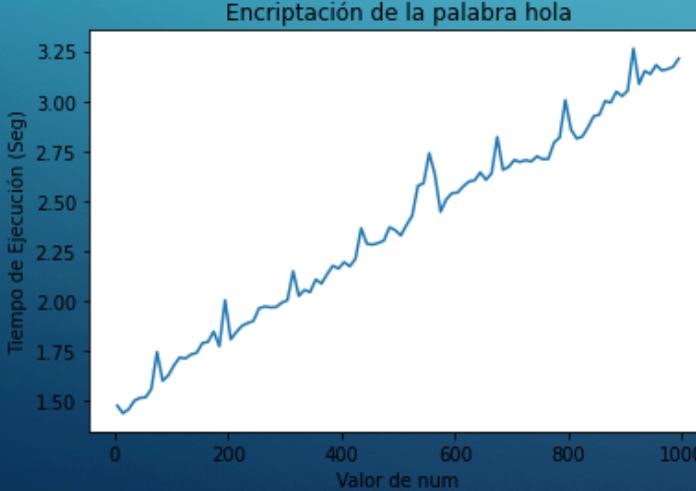
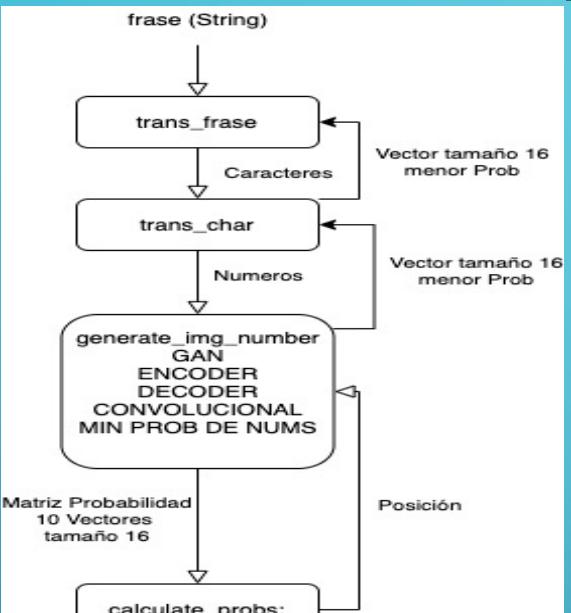


- Entradas: Espacio latente codificado dimensión 16.
- Salida son los números del 0 al 9.
- Posteriormente se pasan de ASCII a texto.
- Modelo idéntico si juntamos el decoder + convolucional.
- Genero Dataset artificial con generador GAN + ENCODER. Dataset de 10000 muestras 1000 de cada número.
- Resultados: loss: 0.0264 - accuracy: 0.9941 (100 Épocas).
- Parámetros: 145370



FUNCIONES TRANSMISIÓN

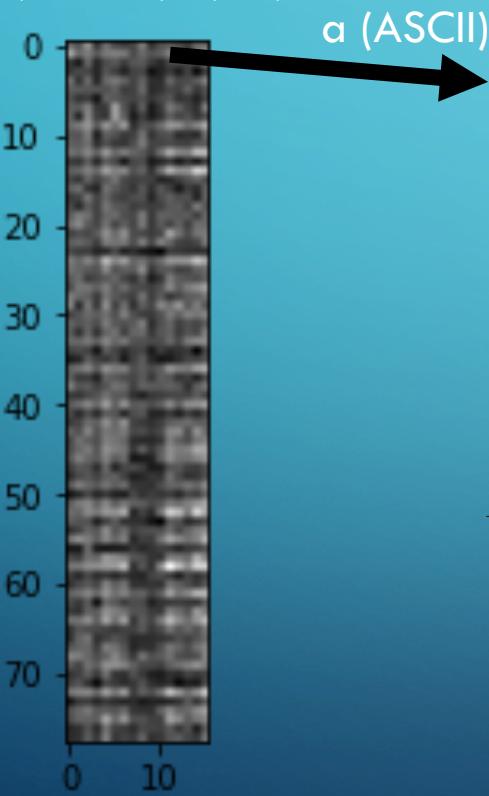
RECEPCIÓN TEXTO



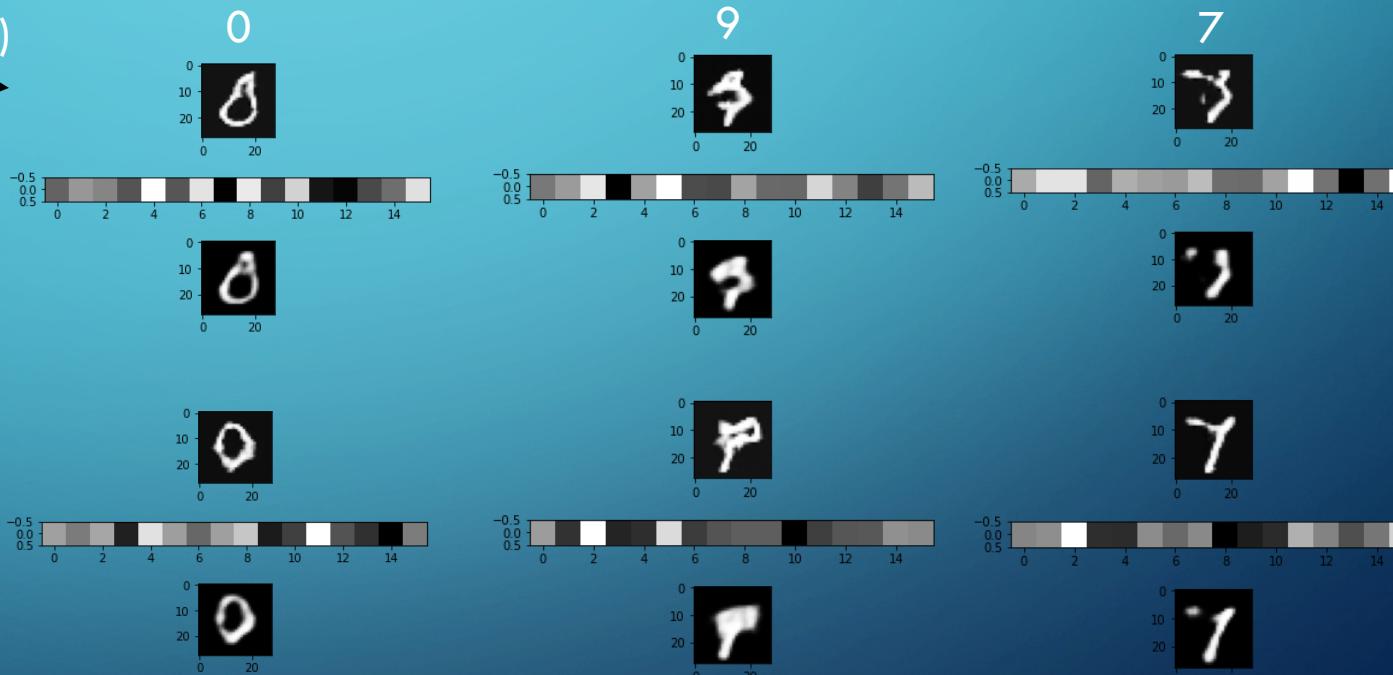
- Transmisión:
 - Entrada: Texto a enviar.
 - Salida: vector de dimensiones: $(\text{len}(\text{texto}) \times 3,16)$
 - Genera proceso completo de encriptación-desencriptación.
 - Devuelve la menor probabilidad de 10 elementos.
- Ventajas:
 - Se asegura la correcta recepción del mensaje.
 - Se asegura la máxima variabilidad posible.
- Desventajas:
 - Tiempo para encriptar.
 - Longitud del mensaje, por cada carácter (3,16).
- Recepción:
 - Entrada: vector codificado de dimensiones: $(\text{len}(\text{texto}) \times 3,16)$.
 - Salida: Texto decodificado.

FUNCIONES TRANSMISIÓN RECEPCIÓN TEXTO

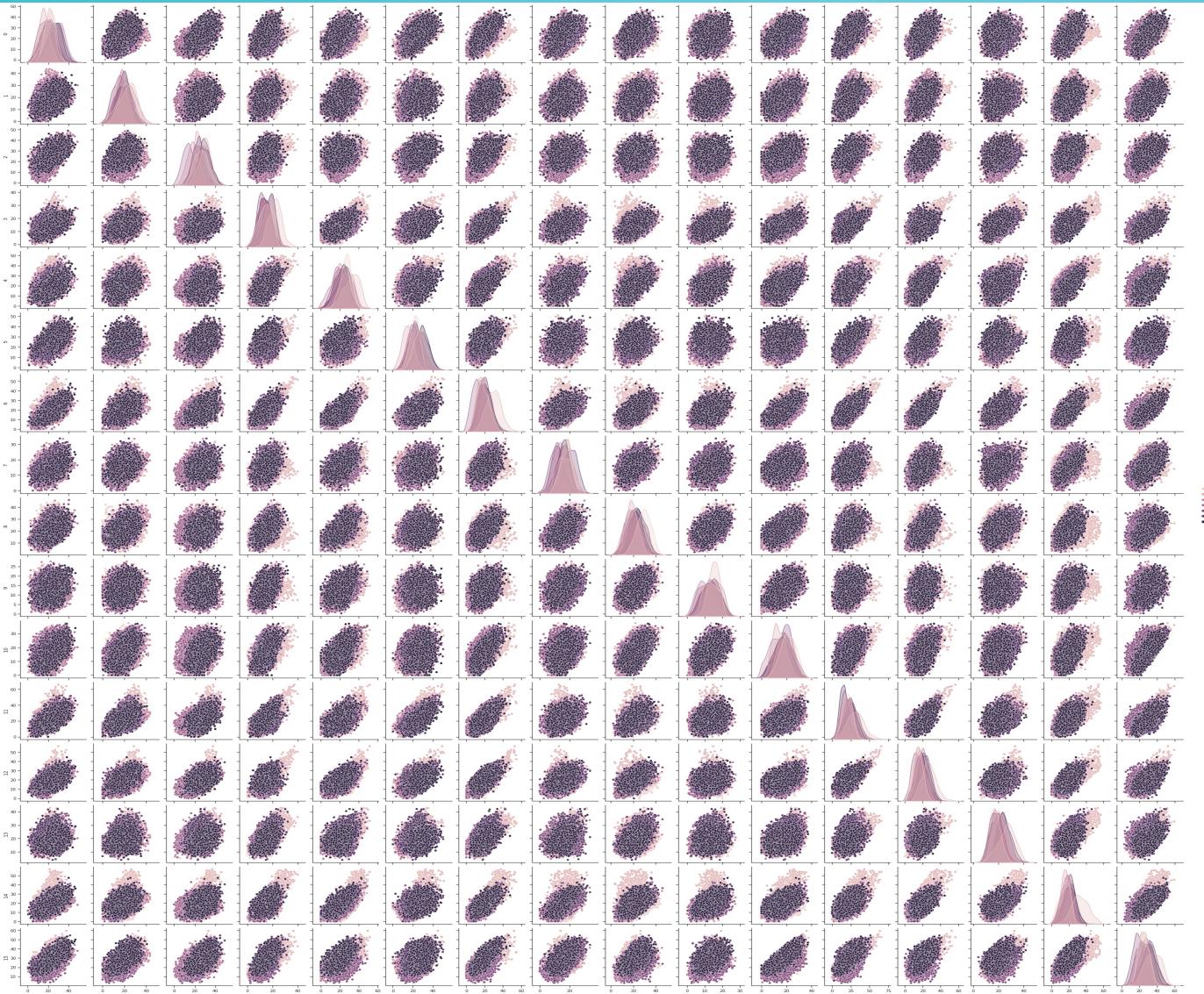
- Para la cadena "abcdefghijklmnopqrstuvwxyz", se genera el siguiente vector: $(26 \times 3), (16)$.



Se puede ver el efecto que produce el generador GAN en el espacio latente al generar varias veces el mismo número y la ventaja que esto supone para criptografía:

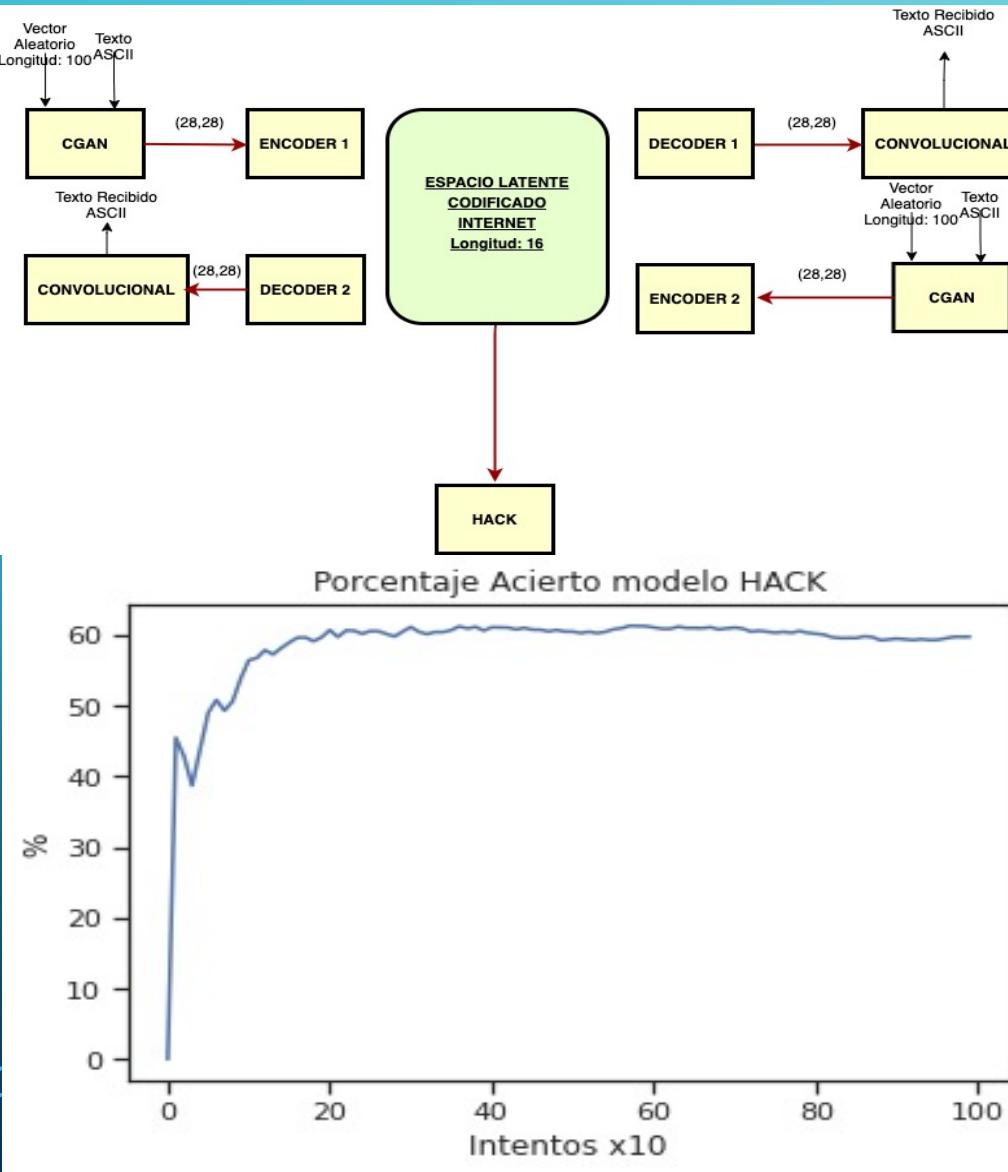


RESULTADOS



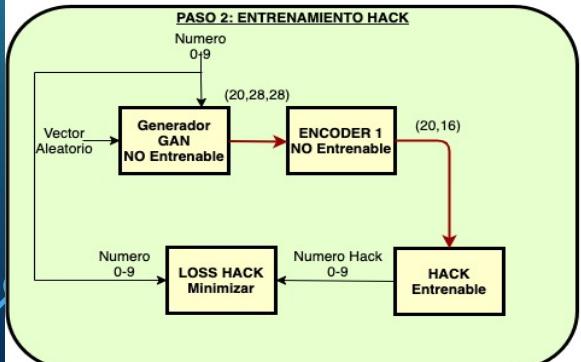
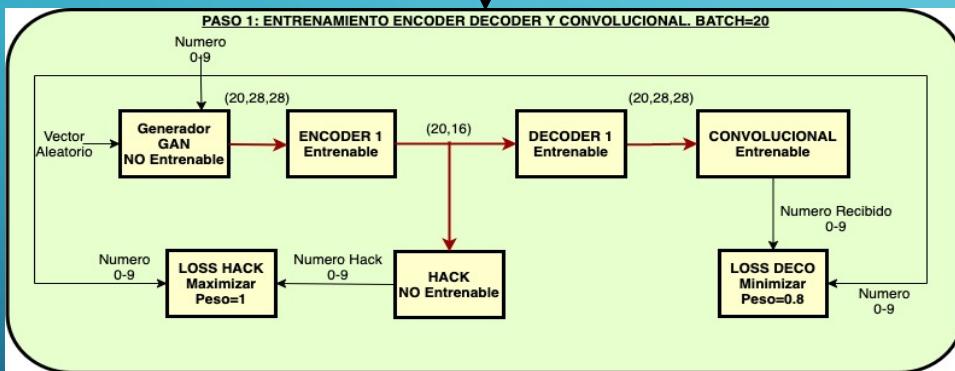
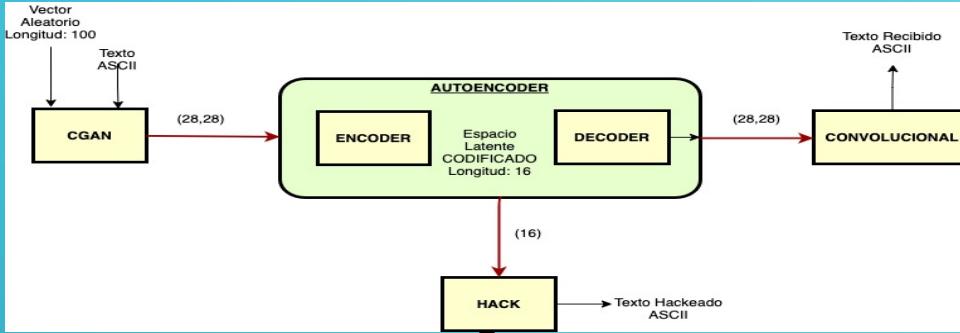
- Análisis estadístico del dataset generado (números del 0 al 9).
- Vector de dimensión 16 que genera el encoder.
- Se puede ver que no hay un patrón aparente.

RESULTADOS

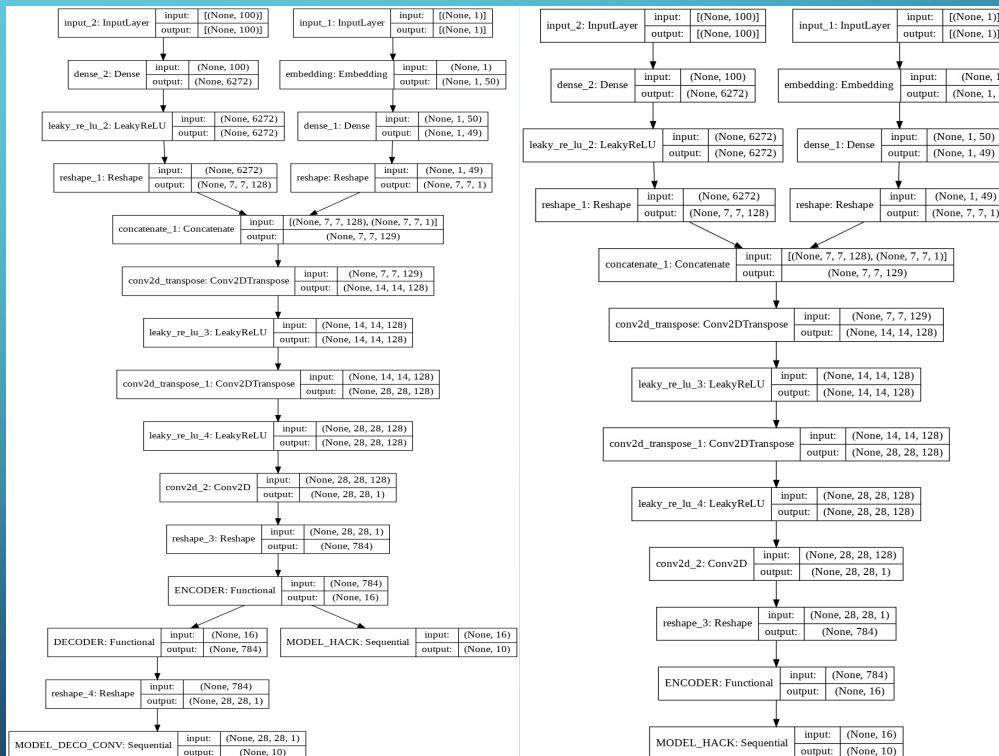


- Modelo bidireccional.
- Entradas: Imagen del decoder (28x28)
- Se entrena con dataset MNIST.
- Salida son los números del 0 al 9.
- Posteriormente se pasan de ASCII a texto.
- Se pasa el texto “Hola me llamo Felipe.”
- Resultados en el modelo HACK:
 - HolG me g\x8aaev Fe\x8aioe.
 - \\\ol\x11Hm\(\x18egamo\x16Felipe.
 - Holc\x1fmÉÓíDcgo FÉliple.
- Se genera 1000 strings de prueba compuestos por dos caracteres aleatorios cada uno, el modelo HACK saca una precisión aproximada del 60%.

SOLUCION FINAL ADOPTADA



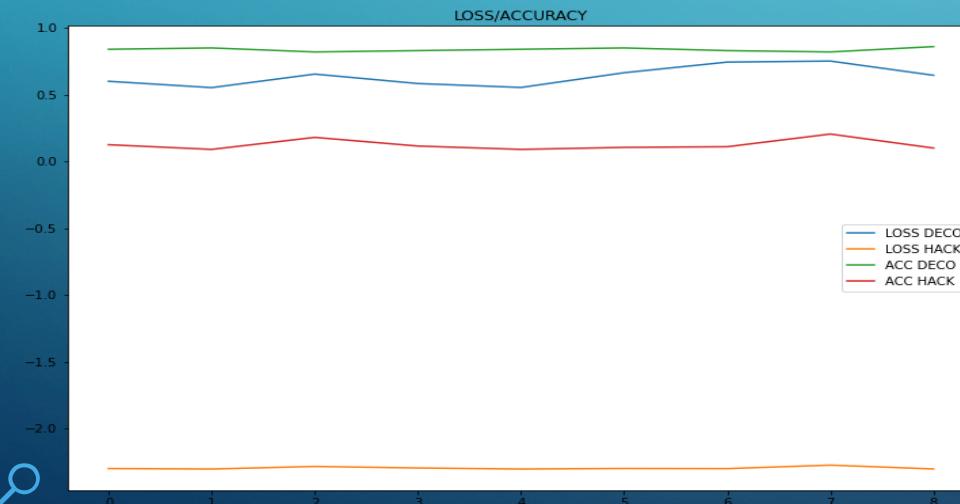
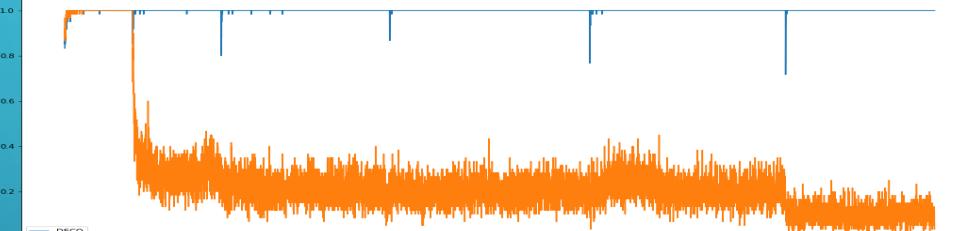
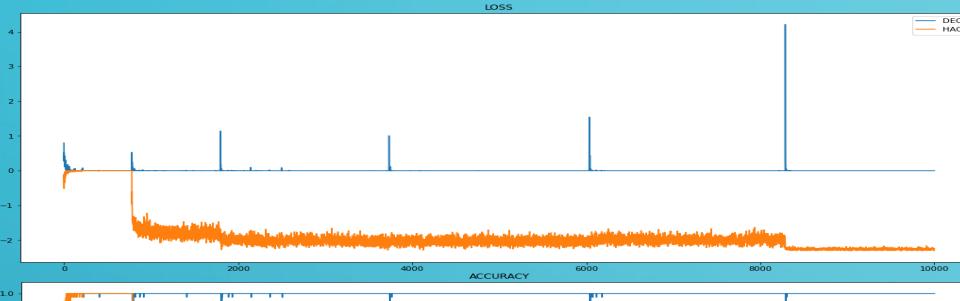
- Generar una estructura GAN para entrenar todos los modelos a la vez.
- Entradas: Vector aleatorio y Número (0-9).
- Salidas: Número Recibido (0-9) y Número Hack (0-9).
- Se crean dos modelos partiendo de los ya existentes.



Parámetros: 1569852

Parámetros: 1424482

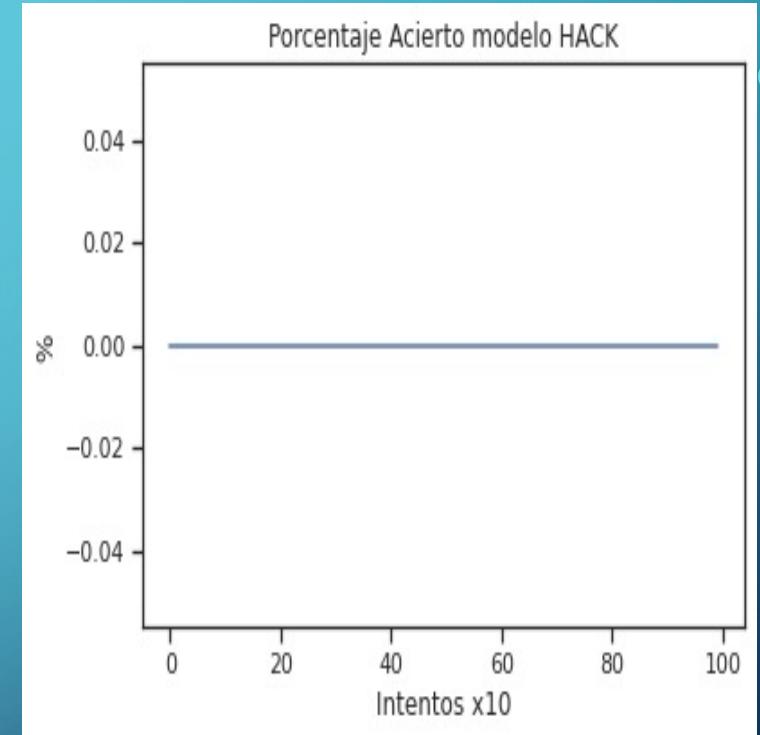
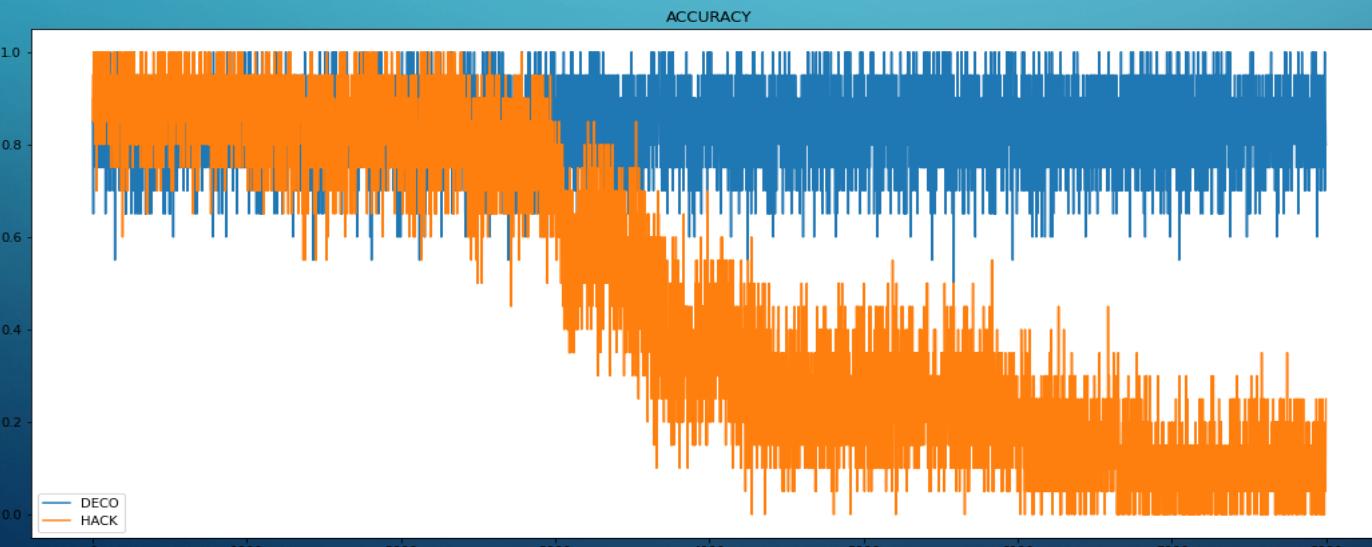
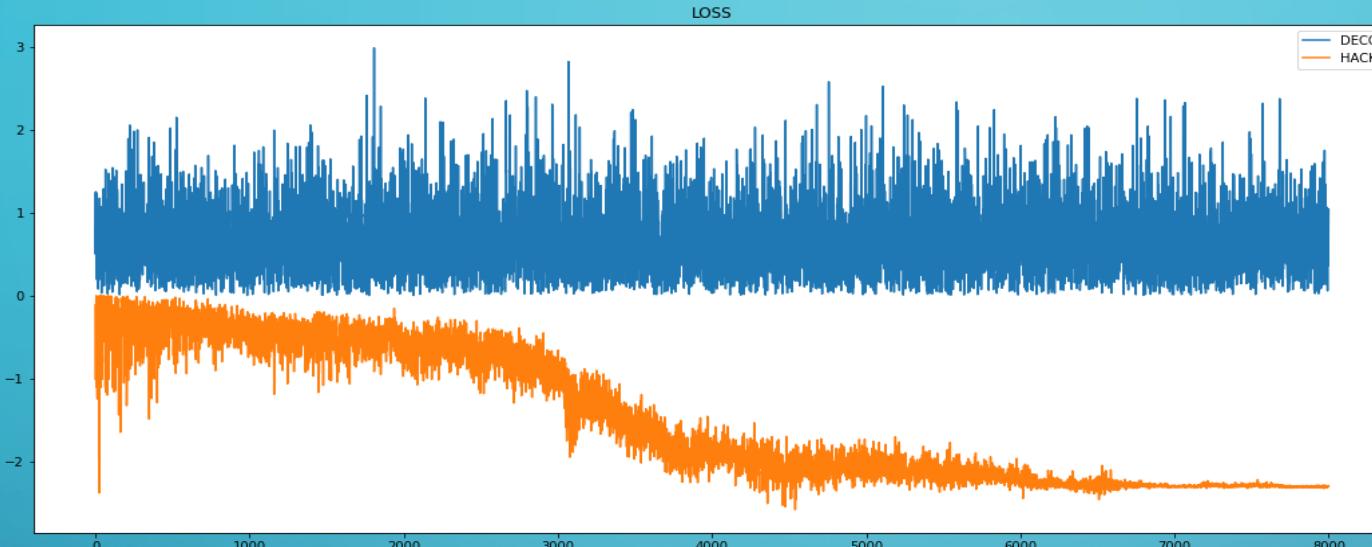
ENTRENAMIENTO DE LOS MODELOS EN CONJUNTO



- Se crea función de perdida personalizada para maximizar la perdida.
- Generador GAN no se entrena. Se realiza prueba entrenando los pesos del generador GAN sin resultados satisfactorios.
- Partimos de los modelos ya pre-entrenados.

- Búsqueda Hiper-parámetros, peso LOSS DECO.
- 0,1 – 0,9. 7000 épocas.
- No se encuentra variación significativa, se fija en 0,8.

RESULTADOS



- Modelos Pre-entrenados
- 8000 épocas.
- LOSS DECO = 0,8.

CONCLUSIONES

- Aplicando técnicas GAN y estadísticas se puede generar encriptaciones sin necesidad de aplicar algoritmos de criptografía.
- Enfrentando los modelos se puede entrenar un conjunto de redes neuronales para generar una encriptación mas fuerte.
- Posibles mejoras:
 - Reducir la transmisión de información a un solo vector de tamaño 16.
 - Convertir el código a TensorFlow Lite, para en inferencia, usar herramientas tipo Coral.
 - Probar otro tipo de arquitecturas para los modelos.