**Justificación código GAN con Transformer encoder**

**Preparacion de Datos**

Se configuró el entorno importando las bibliotecas PyTorch, TorchVision y NumPy. Se cargó el dataset FashionMNIST que contiene 60,000 imágenes de 10 categorías de ropa. Se implementó una clase Dataset para convertir las imágenes a tensores de PyTorch, en este caso el dispositivo es CPU, asegurando que los datos estuvieran normalizados para el entrenamiento.

**Arquitectura**

Como toda GAN el modelo tiene dos partes que son :

Generador con Transformer: Usé un generador que utiliza la arquitectura Transformer en lugar de una red convolucional. Este generador toma vectores de ruido de 100 dimensiones y los transforma en secuencias de parches de 7x7 píxeles. Implementé multihead-attention que permite al modelo aprender relaciones entre diferentes partes de la imagen, junto con normalización por capas y conexiones residuales para que el modelo no olvide lo que aprendió en capas anteriores.

Discriminador MLP: Utilicé una red neuronal fully connected como discriminador, compuesta por tres capas lineales con funciones de activación ReLU. Su función es distinguir entre imágenes reales del dataset e imágenes generadas por el transformer.

**Entrenamiento**

Se entrena el discriminador con lotes que tienen imágenes reales y también generadas

Se actualiza el generador para crear imágenes que puedan engañar al discriminador

Se utiliza la función de pérdida BCEWithLogitsLoss para las perdidas o de las dos partes

Incluí barras de progreso para ver las perdidas del generador y discriminador al mismo tiempo

**Evaluación y generacion**

Finalmente, pasamos con la visualización de los resultados. Se genera un gráfico de las pérdidas durante el entrenamiento para ver la convergencia del modelo, y se muestran ejemplos de imágenes generadas después del entrenamiento.