Entrenamiento de Modelos Generativos con Dataset y Recursos Limitados

Álvaro Felipe Pérez

Technical University of Madrid, ETSISI, Ctra. de Valencia Km. 7, Madrid, Spain

Abstract: Este trabajo aborda el desafío de generar nuevos objetos en el contexto de las texturas de pixel art de Minecraft (32×32), un dominio caracterizado por un conjunto de datos limitado y un estilo único. Se ha buscado optimizar la coherencia visual y la variedad de las muestras generadas de tres arquitecturas generativos: la Conditional Variational Autoencoder (CVAE), la Conditional Generative Adversarial Network (CGAN), y el Denoising Diffusion Probabilistic Model (DDPM).Estos modelos reciben un prompt de texto para generar objetos nuevos, típicamente la mezcla semántica de dos ya existentes. Finalmente, se presentan y comparan los resultados obtenidos por los tres modelos, evaluando sus respectivas fortalezas y debilidades al operar sobre imágenes de baja resolución en entornos con datos limitados.

Keywords: Generative Adversarial Networks, Variational Autoencoders, Denoising Diffusion Probabilistic Models, Image Generation.

Introducción

El estado del arte en modelos generativos ha demostrado capacidades impresionantes para sintetizar imágenes de alta fidelidad, impulsado principalmente por el acceso a conjuntos de datos masivos y diversos. Sin embargo, la aplicación de estas arquitecturas en dominios con escasez de datos (limited data regimes) sigue siendo un desafío significativo.
Este desafío se exacerba cuando se trabaja con imágenes de baja resolución o pixel art (como texturas de 32×32 de minecraft Faithful [0]). Dada la cantidad limitada de objetos oficiales de Minecraft y la inexistencia de otro videojuego con un estilo de arte similar, no es viable recurrir a conjuntos de datos externos.
No hemos utilizado la calidad original del videojuego (16×16), ya que un error mínimo en estas imágenes de baja resolución puede resultar en una distorsión muy significativa en la generación final, a diferencia de la fotografía de alta resolución, donde un poco de ruido puede pasar desapercibido, en imágenes de pocos píxeles la precisión es crítica: un error de un solo píxel o color se percibe como una aberración visual que estropea el objeto.

Para cada modelo (CVAE, CGAN y DDPM) analizamos las limitaciones al operar sobre baja resolución intentamos aumentar la coherencia visual y la variedad de las muestras generadas.

Trabajos Relacionados

Los modelos utilizados se pueden categorizar de la siguiente manera:

- VAEs: Los autoencoders variacionales modelan la distribución de los datos a través de una representación latente gaussiana. Estas redes utilizan una pérdida basada en la evidencia inferior (ELBO) para aprender una codificación probabilística.
- GANs: Las redes generativas adversariales consisten en un generador y un discriminador que compiten en un juego minimax. El generador crea muestras sintéticas y el discriminador aprende a distinguirlas de las reales. Las versiones condicionadas (cGAN) inyectan información auxiliar en ambas redes.
- Modelos de difusión: Últimamente, los modelos de difusión han emergido como una familia poderosa de generadores con rendimientos récord en síntesis de imágenes. Estos modelos generan datos a través de un proceso de difusión reversible.

Cada enfoque presenta fortalezas y limitaciones particulares: los GAN suelen generar muestras de alta fidelidad pero pueden ser difíciles de entrenar, mientras que los VAE garantizan diversidad de muestras a través del espacio latente, aunque a veces producen imágenes más borrosas. Los modelos de difusión, por su parte, ofrecen alta fidelidad y diversidad a cambio de un mayor costo computacional.

Metodología

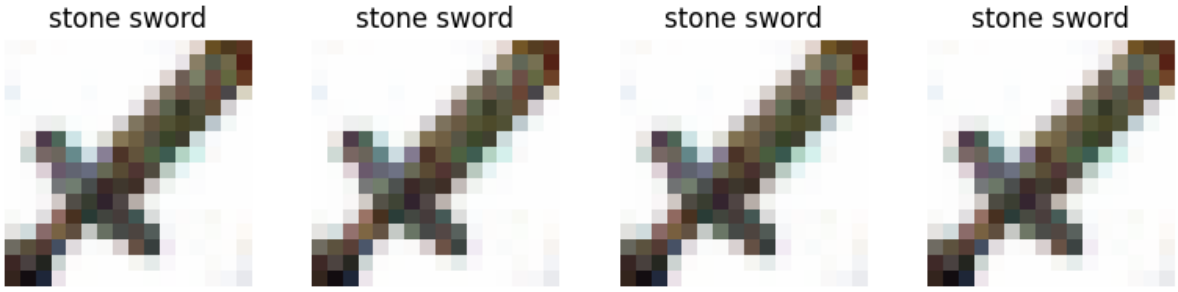
Consideraciones Generales sobre la Arquitectura

Para este estudio, se excluyó la consideración de arquitecturas como los Visual Transformers o sistemas generativos basados en transformers debido a su alta dependencia de grandes volúmenes de datos para el entrenamiento (data hungry). Dada la naturaleza del dataset de objetos de Minecraft Faithful, la implementación de tales modelos resultaría ineficiente.

Inicialmente, se probaron algunos modelos utilizando la resolución original de las texturas de Minecraft, que es de 16×16 píxeles. Sin embargo, se observó que, debido a la reducida definición, cualquier error resultaba en un impacto visual grande en el objeto final. Para mejorar la calidad del resultado sin perder de vista el objetivo de centrarse en el minecraft, se optó por emplear el pack de texturas de Faithful32x32 [0] que contenía los mismos objetos pero con una resolución ligeramente

aumentada. Esta resolución se utilizó en todos los métodos generativos implementados para el trabajo.

```
Imágenes generadas 16x16
```



En los modelos generativos que requieren la combinación de un vector de ruido base y otro de embeddings del texto, se aplicó un proceso de normalización a los vectores. El objetivo de esta normalización es asegurar que ninguna de las entradas domine excesivamente la generación del objeto y facilitando al modelo a aprender una representación más equilibrada, permitiendo que las condiciones de texto guíen la composición del objeto sin que la variación estocástica del ruido se vea anulada.

Preproceso de datos

Durante el preprocesamiento de datos se tomaron todos los objetos presentes en la carpeta de items del pack Faithful y se les añadió un fondo blanco para sustituir la transparencia original, de modo que todas las imágenes quedaron en formato RGB. A partir del nombre de cada archivo se generó también una etiqueta, eliminando la extensión y reemplazando los guiones bajos por espacios. Con estas dos piezas de información se construyó un dataframe que reúne cada objeto en una fila, dejando el conjunto listo para su uso. Dado que el tamaño del dataset es reducido, lo habitual sería aplicar técnicas de data augmentation [1]; sin embargo, debido a la baja resolución del pixel art, muchas transformaciones clásicas como giros, recortes o cambios de color introducen distorsiones visibles que alteran la semántica del objeto. Por tanto la única transformación que podría resultar viable fue un horizontal flip, pero para nigún modelo funcionó,ya que los sitemas generadores producían las imágenes del objeto resultado superpuesto entre sí en las 2 orientaciones posibles.

Text encoder

Para la codificación textual se implementó una clase TextEmbedding que configura la dimensionalidad del espacio latente destinado a representar los embeddings de texto [2]. Tras analizar las labels asociadas a los objetos (las mismas que se usan como prompts durante el entrenamiento de los modelos generativos) se determinó que el conjunto de palabras únicas era relativamente reducido y

por lo tanto se consideró que la codificación de cada prompt con 100 valores proporcionaba una capacidad de representación suficiente.

El vocabulario se construyó aplicando una limpieza sencilla convirtiendo a minúsculas y eliminando dígitos por que ciertos objetos de Minecraft incluyen números en sus nombres debido posibles representaciones alternativas en función de determinadas condiciones, pero dichas variantes no representan diferencias conceptuales para el proceso de generación. Finalmente, la codificación de cada prompt se obtiene tokenizando cada palabra del texto de entrada y calculando el promedio de los embedding correspondientes.

Modelo cVAE

El CVAE empleado está compuesto por un encoder y un decoder condicionados. El encoder recibe la imagen de entrada, aplica convoluciones para obtener su representación latente y la concatena con el embedding del prompt textual; a partir de este vector se estiman los parámetros μ y σ^2 que definen la distribución normal del espacio codificado [3]. El decoder, construido con una profundidad simétrica al encoder, para deshacer sus compresiones, recibe la muestra latente generada mediante el truco de reparametrización para el entrenamiento, o ruido durante la inferencia, y la concatena junto con el embedding del prompt, aplicando operaciones de deconvolución para reconstruir la imagen. Para el entrenamiento de los pesos de los modelos se implementó una clase CVAE que encapsula el proceso de reparametrización [4] y la propagación completa del encoder y decoder condicionados. A diferencia de un autoencoder convencional, el CVAE hace continua y estocástica la distribución latente, por lo que la función de pérdida combina el término de divergencia KL, que empuja la distribución inferida hacia una normal estándar [3], [12]. No obstante, en nuestro caso se observó que el término KL podía dominar la optimización debido al reducido tamaño del dataset, por lo que se introdujo un coeficiente de ponderación β para reducir su influencia. La función de pérdida modificada queda así:

$$\mathcal{L} = \mathbb{E}_{q_\phi(z|x,c)} \left[\|x - \hat{x}\|^2 \right] + \beta \, D_{\mathrm{KL}} \left(q_\phi(z \mid x, c) \,\|\, p(z) \right)$$

donde la divergencia KL [12] se calcula como :

$$D_{\mathrm{KL}} \left(q_\phi(z \mid x, c) \,\|\, p(z) \right) = -\frac{1}{2} \sum_{i=1}^{d} \left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2 \right)$$

y la reparametrización se define como:

$$z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

Modelo cGAN

En el caso de la CGAN se empleó una arquitectura formada por un generador y un discriminador condicionados [5]. El generador recibe el vector de ruido y el embedding del prompt y sintetiza la imagen mediante un proceso de expansión progresiva. El discriminador evalúa conjuntamente la imagen y el prompt, y determina si la muestra corresponde o no a la condición indicada. Durante el entrenamiento se aplicó label smoothing [6] en las etiquetas reales para mejorar la estabilidad adversarial.

Para evitar que las imágenes generadas presentaran saltos abruptos de color, se introdujo una función de pérdida adicional basada en la variación de píxeles adyacentes, penalizando así la falta de continuidad espacial. Además, se añadió un término de reconstrucción que compara directamente la imagen generada con la real condicionada. La función de pérdida total del generador combina la pérdida adversarial con estos dos términos adicionales [5], [6], regulados mediante los factores de ponderación λ_{tv} y λ_{rec}:

Definimos la pérdida del discriminador como:

$$\mathcal{L}_D = \frac{1}{2}\Big[\text{BCE}\big(D(x,y),\,0.9\big) + \text{BCE}\big(D(G(z,y),y),\,0.0\big)\Big]$$

Y la pérdida del generador como:

$$\mathcal{L}_G = \mathcal{L}_{\text{adv}} + \lambda_{\text{tv}}\,\mathcal{L}_{\text{tv}} + \lambda_{\text{rec}}\,\mathcal{L}_{\text{rec}}$$

Donde la la imagen real se denota como x, y la imagen generada como $\hat{x}$. Para la pérdida de variación total $v_{i,j}$ representa el valor del píxel situado en la posición (i,j).

$$\mathcal{L}_{\text{tv}}(v) = \sum_{i,j} \sqrt{(v_{i,j+1} - v_{i,j})^2 + (v_{i+1,j} - v_{i,j})^2}$$

$$\mathcal{L}_{\text{rec}} = \|x - \hat{x}\|_2^2$$

Modelo DDPM

EL Denoising Diffusion Probabilistic Model (DDPM) [7] es modelo generativo basado en un proceso de difusión estocástico que transforma progresivamente una imagen real en ruido gaussiano, definiendo una cadena de Markov hacia una distribución conocida. El modelo debe aprender la revertir esos pasos y en la fase de generación, invertir el proceso mediante denoising iterativo, reconstruyendo gradualmente imágenes limpias a partir de muestras de ruido, condicionadas a un embedding textual. Cada paso supone una pequeña perturbación gaussiana, lo que permite al

modelo aprender a predecir el ruido introducido en cada instante y generar imágenes mediante la eliminación de este ruido en cada iteración.

La actualización en cada paso se expresa como:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t, c)\right) + \sqrt{\beta_t}\,z, \quad z \sim \mathcal{N}(0, I)$$

Donde, x_t representa la imagen en el paso t, $\epsilon_\theta(x_t, t, c)$ es la predicción del ruido realizada por el modelo condicionado al embedding textual c, y β_t corresponde al coeficiente de ruido para ese paso. La inclusión del término aleatorio z asegura diversidad en la generación, ya que evita un proceso determinista para reconstruir la imagen. En ausencia de este ruido adicional en cada paso, se obtendría un Denoising Diffusion Implicit Models (DDIM) [8], que permite generar imágenes con muchos menos pasos y mayor rapidez, pero a costa de una pérdida de diversidad en las muestras generadas.

La arquitectura empleada es un MiniUNet [10] condicional, compuesto por bloques residuales [9] que integran la información del tiempo t y el embedding textual c. El modelo se organiza en un encoder-decoder y un bloque central, usando conexiones residuales para la propagación de gradientes. Cada bloque residual incluye convoluciones, normalización y proyecciones lineales de las condiciones y del tiempo, sumadas a la activación principal, lo que permite que la información condicional influya directamente en la generación en cada capa.

En comparación con los modelos anteriores, los DDPM requieren más capas convolucionales con más canales cada una para aprender la distribución del ruido en múltiples pasos y la dinámica temporal del proceso de difusión, y no reconstruir la imagen de una sola ejecución como hacen el cVAE y la cGAN.

Para entrenar usamos de funcion de perdia el MSE entre el ruido real y el ruido predicho. Esta estrategia, junto con el diseño residual permite generar imágenes de alta fidelidad incluso con un número limitado de muestras por clase, maximizando la capacidad de generar nuevos objetos de Minecraft coherentes.

Experimentos y Resultados

Muestras de Evaluación

Para la evaluación del rendimiento de los modelos se van a usar un conjunto de prompts de prueba que cubren distintos escenarios de escasez de datos y dificultad en la síntesis composicional. El

objetivo es comparar la capacidad de cada arquitectura para generar objetos donde la información es limitada.

Prompt de Prueba	Objeto	Frecuencia
rotten flesh	Real (x1)	One-Shot Learning
emerald pickaxe	Sintético (x2)	few-shot learning
redstone ingot	Sintético (x2)	few-Shot Learning
golden door	Sintético (x2)	Many-Shot Learning
oak chestplate	Sintético (x2)	Many-Shot Learning
slime shears	Sintético (x2)	One-Shot Learning
blue slime dye	Sintético (x3)	One/Few-Shot Learning

Modelo cVAE

El modelo presenta una arquitectura encoder–decoder mediante bloques convolucionales con downsampling en el encoder y su correspondiente inversión en el decoder. Cada bloque se compone de dos capas convolucionales (solo una realiza el downsampling) y utiliza Batch Normalization para estabilizar la distribución de activaciones durante el entrenamiento y facilitar una convergencia más estable.

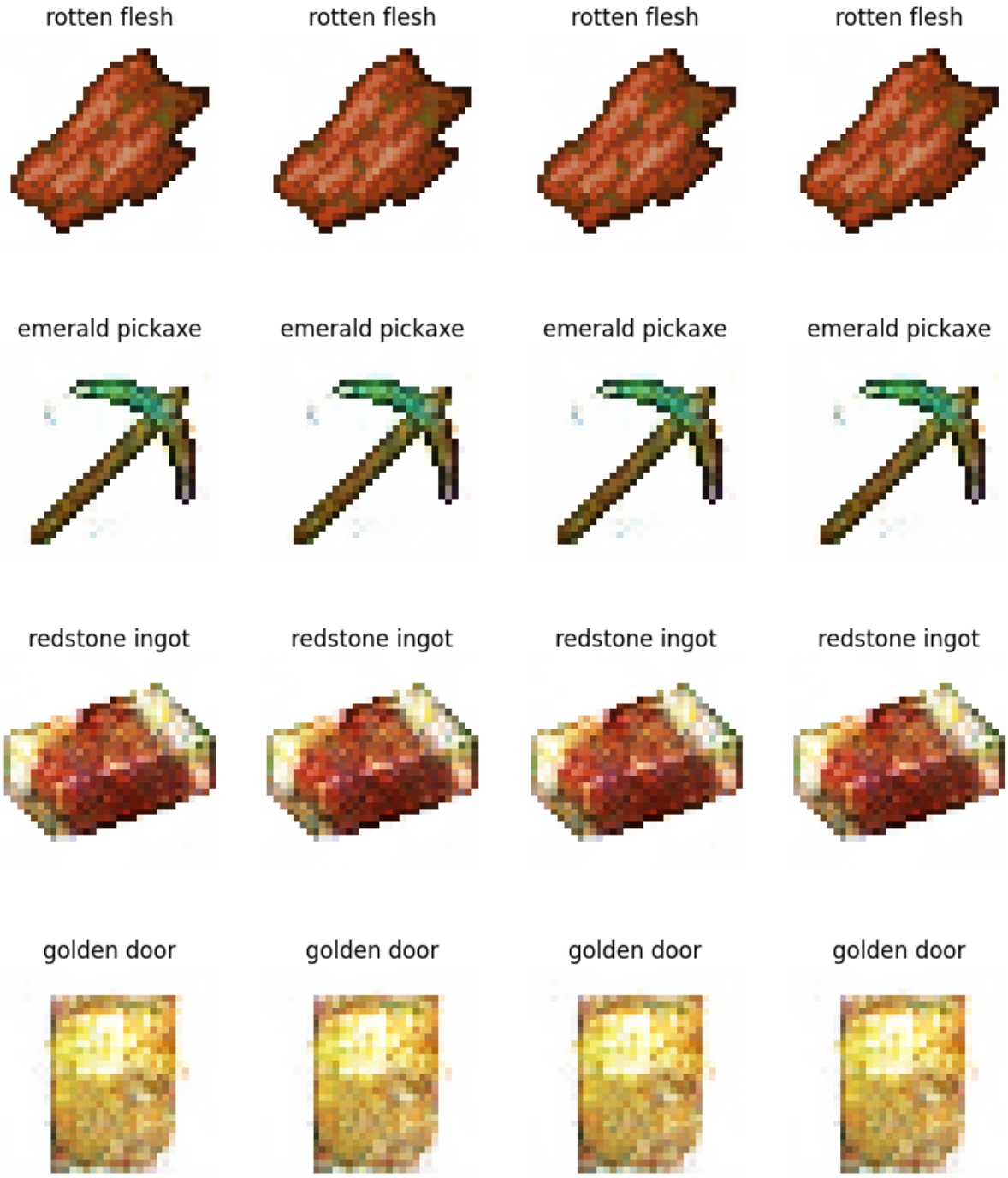
Desde el punto de vista funcional, el encoder genera los dos parámetros μ y σ^2 tras concatenar la representación visual comprimida con el embedding textual. El decoder recibe como entrada dos vectores, el vector de ruido (en la inferencia) o el reparametrizado (en el entrenamiento) y el de condicionamiento, que concatena y pasa por la red para ampliar y reconstruir la imagen. Como se ha explicado antes en la metodología, para entrenar se utiliza una clase CVAE que combina ambos modelos y realiza la reparametrización.

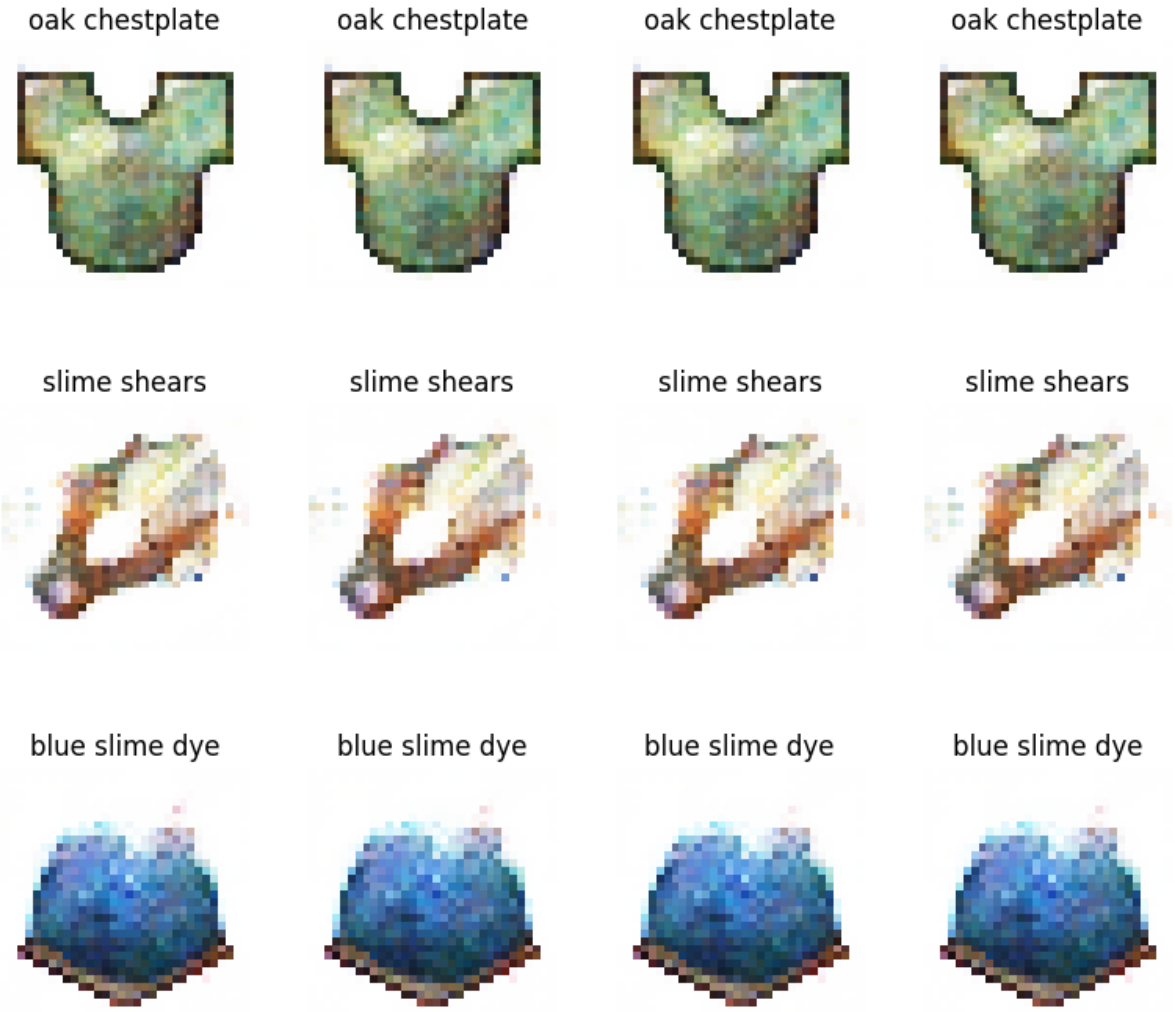
El CVAE ha resultado ser el mejor modelo para eliminar el ruido y distinguir el objeto del fondo blanco, y aunque reconstruye imágenes reales con fidelidad, como vemos en el caso de rotten flesh, el modelo presenta dificultades para combinar objetos distintos. Para intentar solucionar esto se decidió ampliar la información disponible en el espacio latente (para lograr contener más información relevante a los objetos) y aumentar el peso de la pérdida de divergencia KL al entrenar para lograr más continuidad en el espacio, pero aun así no logró mezclar bien los objetos.

Los resultados en escenarios few-shot y many-shot tienden a mostrar superposición de los objetos (como se aprecia claramente en el redstone ingot), en lugar de integrarse de forma coherente.

Además, en los casos donde un objeto es predominante en el dataset frente al otro, el modelo tiende a generar únicamente el objeto con más apariciones, incorporando solo un pequeño ruido con rasgos como el color del otro (como puede ser la oak chestplate o las slime shears).

`Resultados modelo cvae`

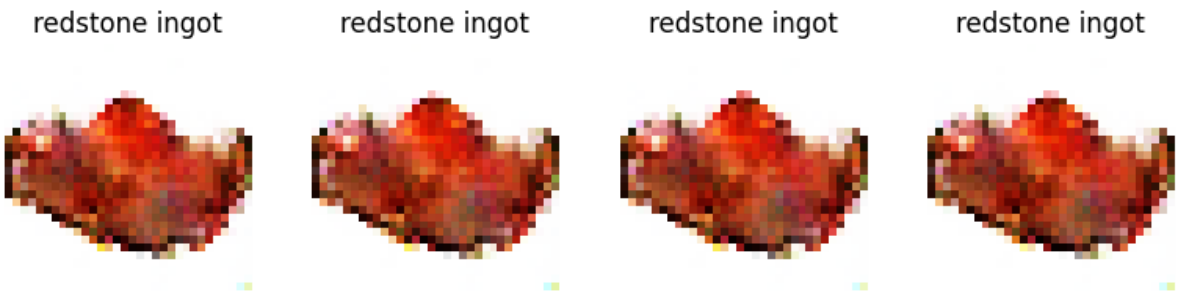




Modelo cGAN

El entrenamiento de las GAN ha resultado especialmente complejo debido a la estructura limitada del conjunto de datos y a las restricciones de capacidad de los modelos. En una primera etapa se utilizaron arquitecturas muy simples, con únicamente tres capas convolucionales tanto en el generador como en el discriminador. Estos modelos eran capaces de aprender la distribución básica, pero el rendimiento era pobre: aparecía un claro underfitting porque la capacidad del generador era insuficiente para capturar la variabilidad necesaria, causando imágenes poco detalladas y de baja calidad.

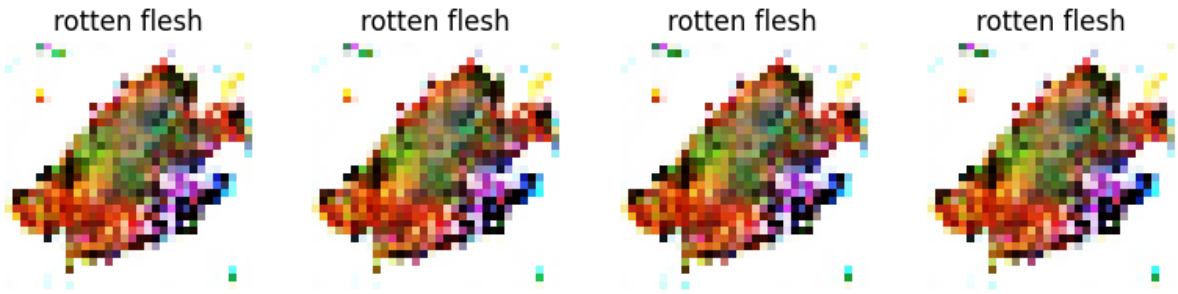
`Resultados modelo gan tres capas`



Sin embargo, al incrementar la profundidad de ambos modelos para mejorar la capacidad representativa, surgió el problema opuesto: con un conjunto de datos tan reducido, el discriminador tendía rápidamente al sobreajuste, aprendiendo casi de memoria las muestras reales. Esto provocaba un desfase en la dinámica de entrenamiento (training imbalance), donde el discriminador se volvía demasiado fuerte frente al generador. Como consecuencia, el generador dejaba de recibir gradientes útiles, pues sus muestras eran clasificadas como falsas siempre, lo que llevaba a un colapso del entrenamiento y a la incapacidad de producir nuevas muestras quedándose con resultados degenerados.

Incluso después de experimentar con arquitecturas más profundas, el modelo no logró generalizar adecuadamente. La GAN terminó siendo visualmente el peor modelo en cuanto a calidad y variedad de las imágenes sintetizadas. Esto se debe a que la ausencia de un conjunto de datos diverso afecta a los 2 modelos utilizados. El discriminador, con pocos ejemplos, memoriza fácilmente las imágenes reales, mientras que el generador, carente de suficientes muestras de entrenamiento y no puede producir variaciones sustanciales para engañar al discriminador. En los resultados se observa claramente este efecto: las muestras generadas presentan una definición de los objetos deficiente y muy poca diversidad independientemente del ruido inicial, un comportamiento característico del colapso modal.

`Resultados modelo gan`



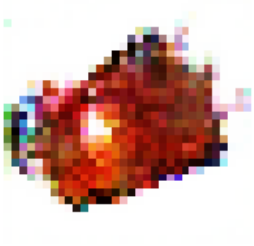
emerald pickaxe

emerald pickaxe

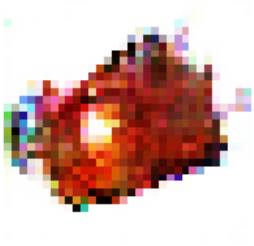
emerald pickaxe

emerald pickaxe

redstone ingot



redstone ingot

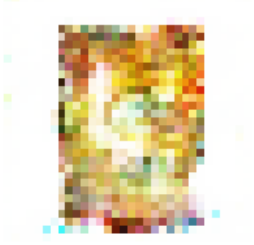


redstone ingot

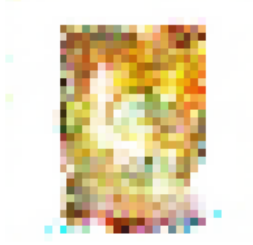
redstone ingot

golden door

golden door



golden door

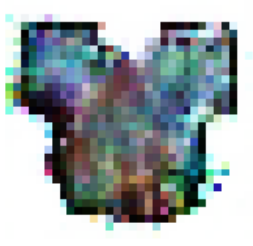


golden door

oak chestplate

oak chestplate

oak chestplate



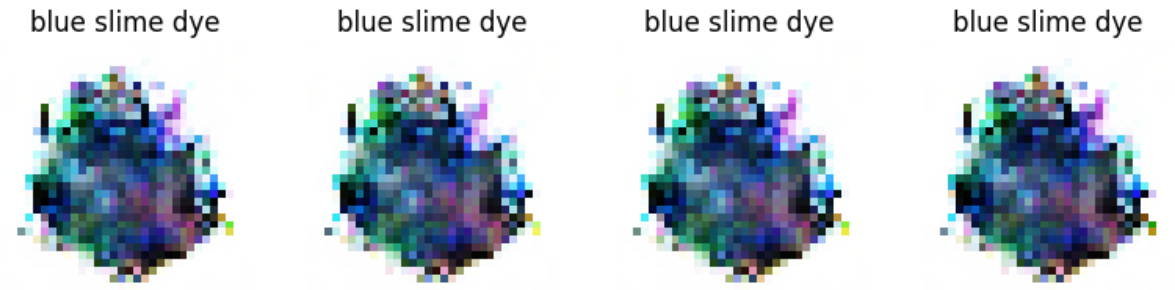
oak chestplate

slime shears

slime shears

slime shears

slime shears



Modelo DDPM

El modelo presenta una arquitectura tipo UNet [10] compuesto por bloques residuales convolucionales [9]. Cada bloque integra el timestep y el embedding textual a las activaciones internas, para condicionar la predicción del modelo en cada paso. Hay un tramo intermedio, con dos bloques residuales que operan sobre la representación más comprimida, para enriquecer la información latente antes de iniciar el proceso de reconstrucción.

La cantidad de neuronas y capas usadas para este modelo es mucho mayor que en el resto, ya que se basa en proceso más complejos y requiere una mayor capacidad de aprendizaje, para eliminar el ruido iterativamente en N pasos, algo que el resto de modelos realiza en una única iteración. Se probó a reducir de 200 a 20 los timesteps, pero como vemos en los resultados, no logra nada debido a que los saltos grandes suponen alteraciones muy pesadas [11] que consigo arrastran posibles errores de inferencia, esto junto con la introducción de ruido en cada predicción para evitar un modelo determinista, suponen un modelo incapaz de cumplir su función de construcción.

`Resultados ddpm con pocas iteraciones`

En los resultados con un modelo con hipeparámetros bien ajustados podemos ver que no es muy capaz de lograr un fondo blanco absoluto como el CVAE ni la reconstrucción de objetos reales ya existentes en el dataset con fidelidad, pero muestra una clara superioridad a la hora de combinar distintos objetos. Tanto en few-shot como en many-shot es capaz de entender cada objeto y fusionarlos adecuadamente en la representación visual, mostrando solo algunos artefactos en ciertas generaciones. No presenta problemas al juntar tres objetos o más (como vemos en el blue slime dye). En cambio, cuando genera objetos que solo ha visto una vez, encuentra más dificultades y no logra combinarlos bien, resultando en una superposición de características; en el ejemplo de slime shears

vemos como la generación conserva los filos de las tijeras y el color verde con forma redondeada del slime en el centro.

```
Resultados modelo ddpm
```

Conclusiones

En conjunto, el CVAE demuestra una gran capacidad para eliminar ruido, reconstruir detalles finos y recuperar la textura de objetos individuales. Sin embargo, persisten limitaciones relacionadas con la representación simultánea de múltiples conceptos. Esto se debe a que su espacio latente continuo no logra capturar mezclas semánticas complejas.

Por otro lado, las GAN han sido el modelo con peor rendimiento. Esto se debe a la escasez de datos y a la naturaleza de su entrenamiento inestable, que a diferencia de modelos auto-supervisados con pérdidas explícitas de reconstrucción, las GAN aprenden mediante un proceso competitivo que se asemeja más a un entrenamiento por refuerzo, donde el generador recibe una señal de recompensa proveniente de un discriminador que también se está entrenando simultáneamente. En escenarios con pocos datos, esto conduce rápidamente a desajustes entre redes, colapso de modos y se refleja en la escasa variedad de las imágenes generadas.

En contraste, el DDPM ha demostrado claramente por qué es el estado del arte actual en generación de imágenes. Su proceso de difusión ofrece una optimización mucho más estable (aunque costosa) y una señal de aprendizaje densa en todos los niveles de ruido, lo que le permite aprender gradientes precisos incluso a partir de conjuntos de datos reducidos. Además, la naturaleza probabilística del proceso inverso, donde cada paso introduce una ligera perturbación estocástica controlada, evita el colapso típico de las GAN. En nuestro caso, el DDPM ha mostrado la mejor capacidad para preservar estructura, respetar el condicionamiento textual y sintetizar imágenes con la mayor variedad y fidelidad entre todos los modelos evaluados.

En cuanto al problema del tamaño del dataset, una ventaja de este entorno es que la tarea permite utilizar modelos con menos neuronas y menos parámetros entrenables, ya que las imágenes poseen una resolución reducida. A diferencia de entrenar modelos multipropósito con resoluciones altas, especializar arquitecturas compactas para un dominio concreto reduce significativamente el tiempo tanto de entrenamiento como de inferencia. No obstante, los resultados obtenidos indican que, incluso en este contexto favorable, el rendimiento óptimo sigue estando limitado por la cantidad y la diversidad de los datos disponibles. Ampliar el dataset con más muestras reales, especialmente aumentando el número de instancias por clase, sería la estrategia más efectiva para mejorar la generalización de cualquiera de los modelos estudiados sobre imágenes pixel-art.

Referencias

[0] Faithful resource — Faithful Texture Pack. Disponible en: https://faithfulpack.net/downloads

[1] Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. & Aila, T.
Training Generative Adversarial Networks with Limited Data. NeurIPS, 2020.

[2] Mikolov, T., Chen, K., Corrado, G. & Dean, J.
Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781, 2013.

[3] Kingma, D. P. & Welling, M.
Auto-Encoding Variational Bayes. ICLR, 2014.

[4] Rumelhart, D. E., Hinton, G. E. & Williams, R. J.
Learning representations by back-propagating errors. Nature, 323(6088), 533–536, 1986.

[5] Goodfellow, I., Pouget-Abadie, J., Mirza, M. et al.
Generative Adversarial Nets. NeurIPS, 2014.

[6] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. & Chen, X.
Improved Techniques for Training GANs. NeurIPS, 2016.

[7] Ho, J., Jain, A. & Abbeel, P.
Denoising Diffusion Probabilistic Models. arXiv:2006.11239, 2020.

[8] Song, J., Meng, C. & Ermon, S.
Denoising Diffusion Implicit Models. arXiv:2010.02502, 2020.

[9] He, K., Zhang, X., Ren, S. & Sun, J.
Deep Residual Learning for Image Recognition. CVPR, 2016.

[10] Ronneberger, O., Fischer, P. & Brox, T.
U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI, 2015.

[11] Nichol, A. & Dhariwal, P.
Improved Denoising Diffusion Probabilistic Models. arXiv:2102.09672, 2021.

[12] Kullback, S. & Leibler, R.
On Information and Sufficiency. Annals of Mathematical Statistics, 22(1), 79–86, 1951.

Autor

Álvaro Felipe Pérez, Universidad Politécnica de Madrid, Grado en Ciencia de Datos e Inteligencia Artificial, Proyecto de Ciencia de datos.

El código de este trabajo se puede encontrar en el repositorio de GitHub: PCD

```
https://github.com/Alvarofp537/PCD.git
```