

VQVAE via RoundTrip prior

Proyecto Final

Integrantes: Felipe Méndez
Diego Kauer
Profesor: Felipe Tobar
Auxiliar: Camilo Carvajal
Cristóbal Alcazar
Ayudantes: Roberto Barceló

Fecha de realización: 18 de diciembre de 2023
Fecha de entrega: 18 de diciembre de 2023
Santiago de Chile

1. Introducción

El modelo *Vector Quantized Variational AutoEncoder* (VQ-VAE) es un modelo propuesto [1] que mejora las capacidades de un *Variational AutoEncoder* (VAE) [2] que incorpora técnicas para discretizar el espacio latente y resuelve algunos de los problemas del modelo original ademas de obtener reducciones de menor dimensionalidad.

1.1. VAE

Un VAE tradicional esencialmente funciona con dos secciones, la primera consiste en un **encoder** que codifica los datos de entrada (x), y los representa en un vector de menor dimensionalidad, llamado vector latente (z). Un VAE tradicional, hace esto modelando la distribución posterior ($q_\phi(z|x)$) asumiendo una distribución normal. Este modelo, además incluye un **decoder** que a partir del vector latente es capaz de reconstruir la imagen original. Esta parte modela la distribución $p_\theta(x|z)$. Para entrenar estos modelos se calcula el error entre la imagen original y aquella reconstruida, minimizando esta diferencia. Esto es equivalente a maximizar la *cota inferior de la evidencia* (ELBO) [3].

$$ELBO = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x)] - D_{KL}(q_\phi(z|x)||p_\theta(z)) \quad (1)$$

Como se puede ver, al maximizar la ELBO se está minimizando implícitamente el término de la divergencia KL , la cual mide la diferencia entre dos distribuciones de probabilidad. Un fenómeno llamado *posterior collapse* que puede ocurrir al ir acercando $q_\phi(z|x)$ a $p_\theta(z)$ es que el decoder ($q_\phi(z|x)$), en vez de aprender una representación de baja dimensionalidad (z) dado un dato x , el decoder aprenda la distribución marginal del espacio latente $p_\theta(z)$. Esto suele ocurrir cuando los modelos no se entrena lo suficiente. Como veremos más adelante, VQ-VAE evita este problema al modificar la función de perdida.

1.2. VQ-VAE

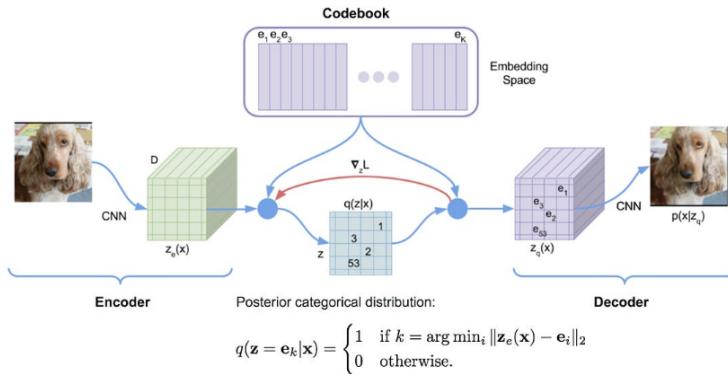


Figura 1: Arquitectura VQ-VAE

La arquitectura de los VAE generan representaciones continuas de los datos de origen, sin embargo es más natural construir representaciones discretas. Por otra parte, VQ-VAE resuelve este desafío al implementar técnicas de cuantificación vectorial para discretizar la representación de los datos tras ser codificadas por el encoder. Como se puede observar en la figura 1 se utiliza un *codebook* para encontrar los estados latentes más cercanos y reemplazar estos códigos discretos en la representación latente. Al revisar la ecuación (2), se puede notar que no se incorpora la divergencia KL evitando así el fenómeno de *posterior collapse*.

$$\mathcal{L} = \log p(x|z_q(x)) + \|\mathbf{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \mathbf{sg}[e]\|_2^2 \quad (2)$$

La función de pérdida consta de tres términos para entrenar el modelo. En primer lugar, se tiene la pérdida de reconstrucción, que optimiza el encoder y decoder. Luego, el segundo término actualiza el *codebook* usando la norma L2 entre el espacio latente y las salidas del encoder. Finalmente, se agrega un término de pérdida de compromiso para evitar un crecimiento excesivo del espacio de latente.

La arquitectura del modelo se puede ver con más detalle en el anexo (A.1).

1.3. RoundTrip

En el paper original de VQ-VAE [1], se utiliza PixelCNN para aprender la distribución *prior* ($p(z)$), y así generar muestras. Para explorar otros generadores, se eligió **RoundTrip**, dado que este modelo está dentro del estado del arte en la estimación de densidades de probabilidad. La idea clave de Roundtrip es aproximar la distribución objetivo como una convolución de una gaussiana con una distribución inducida al transformar una distribución base, donde la transformación se aprende mediante el entrenamiento conjunto de dos modelos GAN (Figura 2) [4]. En donde un par de *generator-discriminator* genera muestras en el espacio latente, y el otro *generator-discriminator* genera vectores aleatorios. Se utilizó como base el código del siguiente [repositorio](#).

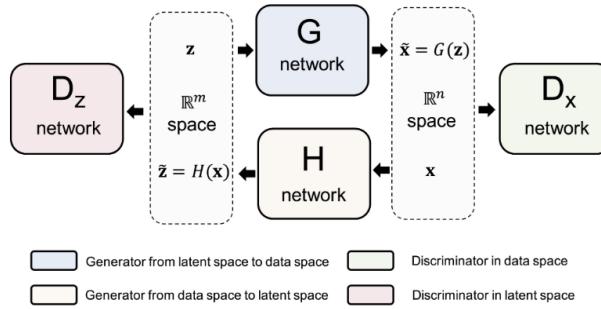


Figura 2: Arquitectura del modelo Roundtrip.

2. Desarrollo

Se comenzó por entrenar el modelo VQ-VAE utilizando el conjunto de datos de CIFAR-10. Para esto se utilizó un *codebook* de tamaño 256 con una dimensión de 512 y un β de 0.25. Para el entrenamiento se entrenó durante 200 épocas con un *batch size* de 128 y una tasa de aprendizaje de $1e^{-3}$. El entrenamiento duró aproximadamente 4 horas. Luego de haber construido un espacio latente satisfactorio, se entrenó el modelo *RoundTrip* unas 60 épocas debido a una falta de tiempo. Se utilizó un vector aleatorio de tamaño 128, tasa de aprendizaje $2e^{-4}$ y *batchsize* de 2048. Esto duró aproximadamente 2 horas.

3. Resultados

A continuación se puede ver los resultados del entrenamiento de VQ-VAE. En la figura (3) se puede ver la reconstrucción obtenida de este modelo. Cabe mencionar que se llegó a una loss final de 0.037. En el anexo (B.1) se puede apreciar la compresión obtenida utilizando este modelo, llegando a un compresión del 98 %.

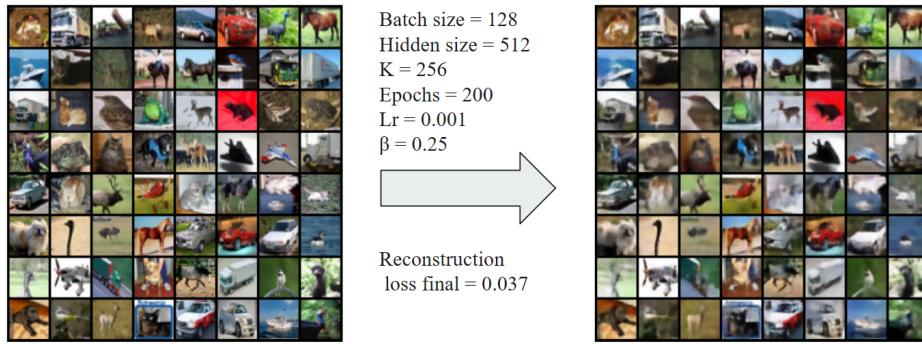


Figura 3: Reconstrucción de las imágenes luego del entrenamiento del modelo VQ-VAE.

Por otra parte, en la figura (4) se puede ver el muestreo generado por RoundTrip. Como se puede apreciar las muestras generadas no son de muy alta calidad. En el anexo (C.1) se puede ver aquellas generadas por PixelCNN, tal como lo hacen en el paper original. Se puede apreciar claramente que se obtienen resultados sobresalientes, considerando que solo se entrenó durante 20 épocas.

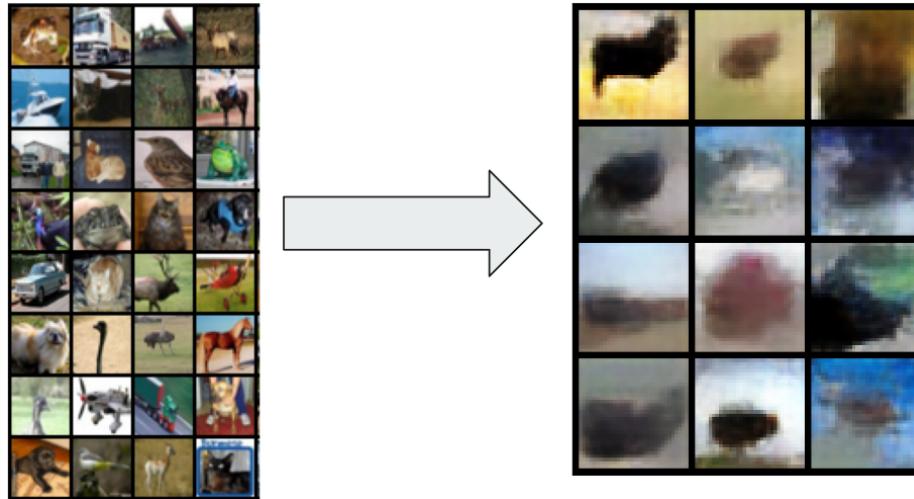


Figura 4: Sampleo de imágenes utilizando como prior un modelo Roundtrip.

4. Conclusiones

En conclusión, los resultados obtenidos de los experimentos muestran un rendimiento positivo de VQ-VAE en términos de entrenamiento y su habilidad para representar las imágenes de buena manera en baja dimensionalidad. Por otro lado, aunque Rountrip logró aprender de manera parcial la distribución del espacio latente, se cree que faltó entrenamiento o que se cometió algún error al reproducir el código del repositorio para lograr buenos resultados pues no samplea bien las imágenes de las clases. Finalmente, se destaca los resultados utilizando PixelCNN como prior, que a pesar de ser entrenado por solo 20 épocas demostró una buena capacidad para generar imágenes y aprender la distribución del espacio latente.

Referencias

- [1] van den Oord, A., Vinyals, O., y Kavukcuoglu, K., “Neural discrete representation learning”, CoRR, vol. abs/1711.00937, 2017, <http://arxiv.org/abs/1711.00937>.
- [2] Kingma, D. P. y Welling, M., “Auto-encoding variational bayes”, arXiv preprint arXiv:1312.6114, 2013.
- [3] Luo, C., “Understanding diffusion models: A unified perspective”, 2022.
- [4] Liu, Q., Xu, J., Jiang, R., y Wong, W. H., “Roundtrip: A deep generative neural density estimator”, CoRR, vol. abs/2004.09017, 2020, <https://arxiv.org/abs/2004.09017>.

Anexo A. Arquitectura VQ-VAE

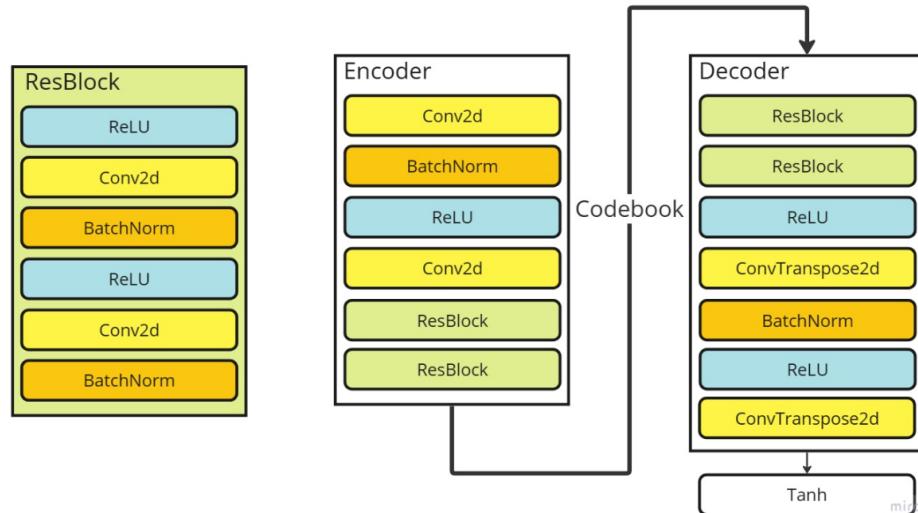


Figura A.1: Arquitectura de VQ-VAE utilizada.

Anexo B. Resultados de Codificación

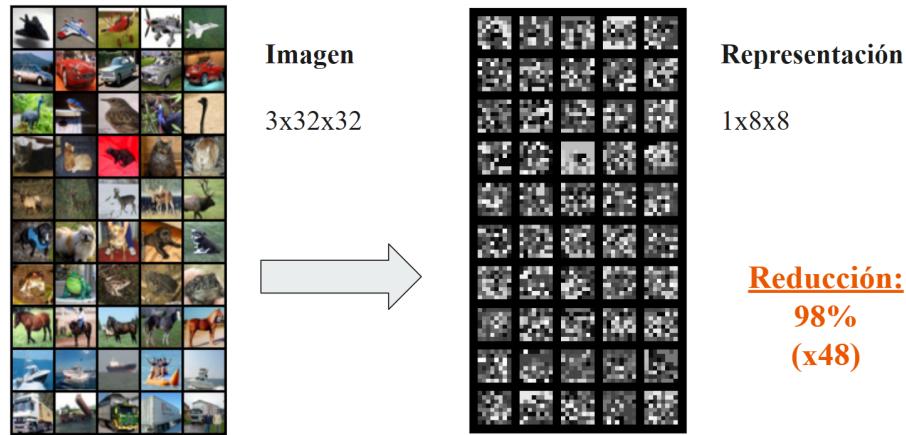


Figura B.1: Transformación de las imágenes al espacio latente discretizado.

Anexo C. Resultados VQ-VAE + PixelCNN

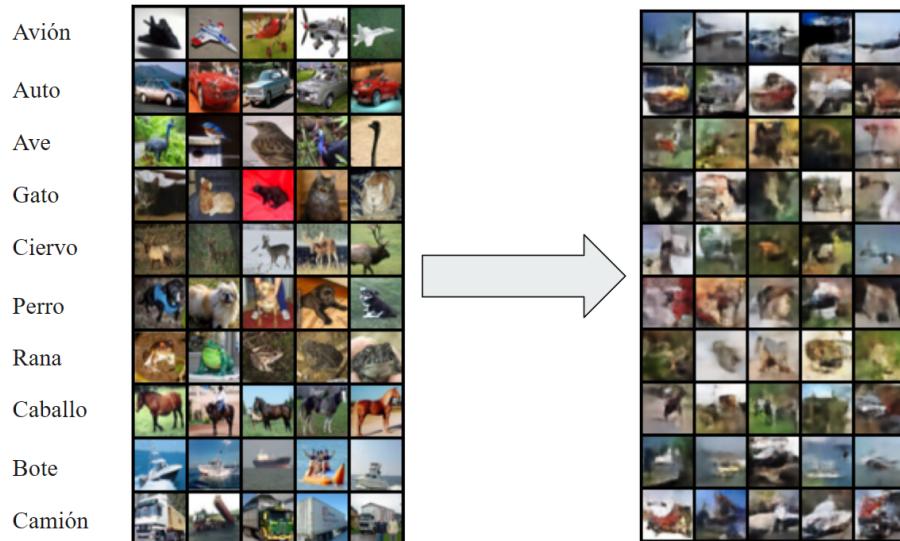


Figura C.1: Sampleo de imágenes para cada clase utilizando como prior un modelo PixelCNN.