Practica 4:
Autoencoder Variacional Convolucional

Ana Paula Morresi

Aprendizaje profundo con aplicación a visión artificial

10 de noviembre de 2025

1. Problemas

Ejercicio 1.

Divergencia de Kullback–Leibler y Cota Variacional

a) Positividad de la Divergencia KL

La divergencia de Kullback–Leibler entre dos distribuciones de probabilidad $p(x)$ y $q(x)$ se define como

$$D_{KL}(p\|q) = \int dx\, p(x) \log \frac{p(x)}{q(x)}.$$

Usando la desigualdad $\log x \leq x - 1$ válida para todo $x > 0$, tomamos $x = \frac{q(x)}{p(x)}$, de donde

$$\log \frac{q(x)}{p(x)} \leq \frac{q(x)}{p(x)} - 1.$$

Multiplicando por $p(x) \geq 0$ e integrando sobre x:

$$\int dx\, p(x) \log \frac{q(x)}{p(x)} \leq \int dx\, (q(x) - p(x)) = 1 - 1 = 0.$$

Como

$$\int dx\, p(x) \log \frac{q(x)}{p(x)} = -D_{KL}(p\|q),$$

se obtiene

$$-D_{KL}(p\|q) \leq 0 \quad \Rightarrow \quad D_{KL}(p\|q) \geq 0.$$

b) Derivación de la Cota Variacional (ELBO)

Partimos de

$$\log p(x) = \int dz\, q(z|x) \log p(x),$$

Usando la regla de Bayes $p(z|x) = \frac{p(x,z)}{p(x)}$, tenemos:

$$\log p(x) = \int dz\, q(z|x) \log \frac{p(x,z)}{p(z|x)}.$$

Podemos escribir esto como

$$\log p(x) = \int dz\, q(z|x) \log \frac{p(x,z)q(z|x)}{p(z|x)q(z|x)}.$$

Separando los términos:

$$\log p(x) = \int dz\, q(z|x) \log \frac{q(z|x)}{p(z|x)} + \int dz\, q(z|x) \log \frac{p(x,z)}{q(z|x)}.$$

El primer término es la divergencia de Kullback–Leibler entre $q(z|x)$ y $p(z|x)$:

$$\int dz\, q(z|x) \log \frac{q(z|x)}{p(z|x)} = D_{KL}\big(q(\cdot|x) \,\|\, p(\cdot|x)\big).$$

De modo que

$$\boxed{\log p(x) = D_{KL}\big(q(\cdot|x) \,\|\, p(\cdot|x)\big) + \int dz\, q(z|x) \log \frac{p(x,z)}{q(z|x)}.}$$

Como $D_{KL} \geq 0$, se obtiene la *cota inferior variacional* o *Evidence Lower Bound (ELBO)*:

$$\log p(x) \geq \mathcal{L}(q) = \int dz\, q(z|x) \log \frac{p(x,z)}{q(z|x)}.$$

Usando $p(x,z) = p(x|z)p(z)$, la ELBO se reescribe como:

$$\mathcal{L}(q) = \int dz\, q(z|x) \log p(x|z) + \int dz\, q(z|x) \log \frac{p(z)}{q(z|x)}.$$

El segundo término es $-D_{KL}(q(\cdot|x)\|p(\cdot))$, por lo tanto:

$$\boxed{\mathcal{L}(q) = \underbrace{\int dz\, q(z|x) \log p(x|z)}_{\text{Error de reconstrucción}} - \underbrace{D_{KL}\big(q(\cdot|x) \,\|\, p(\cdot)\big)}_{\substack{\text{Divergencia KL de la posterior} \\ \text{aproximada respecto al prior}}}.}$$

c) Cálculo de D_{KL} entre Gaussianas

Sean $q(x) = \mathcal{N}(\mu_q, \sigma_q^2)$ y $p(x) = \mathcal{N}(\mu_p, \sigma_p^2)$. Las distribución gaussiana es:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Entonces:

$$\log \frac{q(x)}{p(x)} = \log \frac{\sigma_p}{\sigma_q} - \frac{(x-\mu_q)^2}{2\sigma_q^2} + \frac{(x-\mu_p)^2}{2\sigma_p^2}.$$

Tomando la esperanza bajo $q(x)$:

$$D_{KL}(q\|p) = \mathbb{E}_q\left[\log \frac{q(x)}{p(x)}\right] = \log \frac{\sigma_p}{\sigma_q} - \frac{1}{2\sigma_q^2}\,\mathbb{E}_q[(x-\mu_q)^2] + \frac{1}{2\sigma_p^2}\,\mathbb{E}_q[(x-\mu_p)^2].$$

Calculando las esperanzas:

$$\mathbb{E}_q[(x-\mu_q)^2] = \sigma_q^2,$$

y usando $x - \mu_p = (x-\mu_q) + (\mu_q - \mu_p)$:

$$\mathbb{E}_q[(x-\mu_p)^2] = \sigma_q^2 + (\mu_q - \mu_p)^2.$$

Sustituyendo:

$$\begin{aligned} D_{KL}(q\|p) &= \log \frac{\sigma_p}{\sigma_q} - \frac{1}{2\sigma_q^2}\,\sigma_q^2 + \frac{1}{2\sigma_p^2}\big(\sigma_q^2 + (\mu_q - \mu_p)^2\big) \\ &= \log \frac{\sigma_p}{\sigma_q} - \frac{1}{2} + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2}. \end{aligned}$$

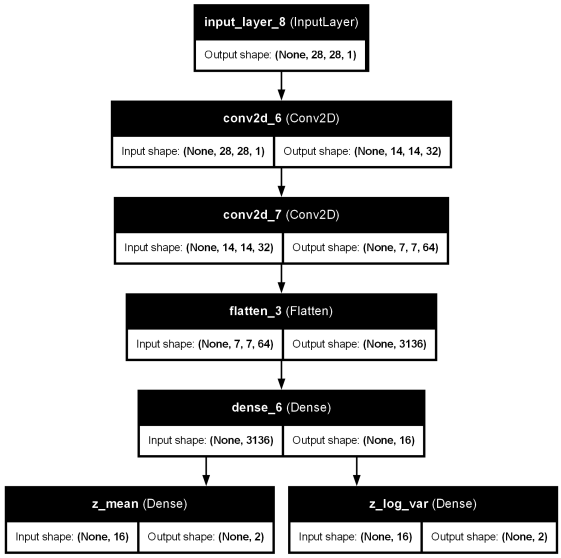
Por lo tanto:

$$\boxed{D_{KL}\big(\mathcal{N}(\mu_q, \sigma_q^2) \,\|\, \mathcal{N}(\mu_p, \sigma_p^2)\big) = \frac{1}{2}\left(\frac{(\mu_q - \mu_p)^2}{\sigma_p^2} + \frac{\sigma_q^2}{\sigma_p^2} - 1 + 2\ln\frac{\sigma_p}{\sigma_q}\right).}$$

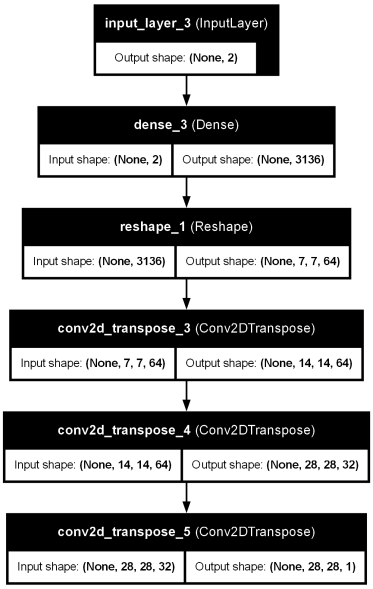
Ejercicio 2.

Se implemento un autoencoder variacional convolucional que aprenda a generar dígitos manuscritos a partir de la base de datos del MNIST.

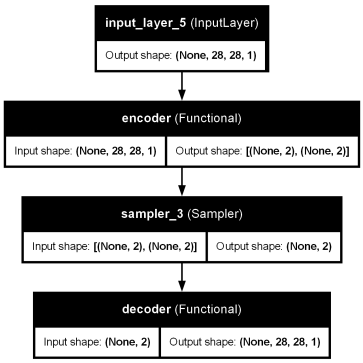
La arquitectura de la red encoder implementada se puede observar en la Fig. 1(a), y la de la red decoder en la Fig. 1(b).



(a) Arquitectura del encoder.



(b) Arquitectura del decoder.



(c) Arquitectura del VAE.

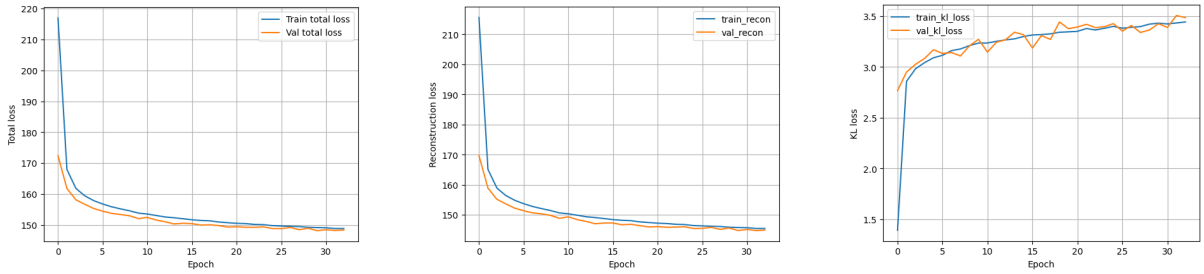
Figura 1: Arquitectura completa del Variational Autoencoder (VAE).

Se entreno el modelo usando el optimizador *Adam*, con un *early stopping* sobre la loss total en validation y paciencia de 3 épocas. En las Fig. 2 se puede observar la evolución de las perdidas en el train y validation datasets. Se obtuvo de valores de loss finales:

Loss total en train: 148.548 || Loss total en validation: 147.244
Loss total en test: 149.556

Error reconstrucción en train: 145.116 || Error reconstrucción en validation: 143.821
Error reconstrucción en test: 146.138

KL loss en train: 3.432 || KL loss en validation: 3.423
KL loss en test: 3.418



(a) Error total. (b) Error de reconstrucción. (c) KL loss.

Figura 2: Evolución de errores a través de las épocas.

Se puede ver que en la loss total la contribución del termino *KL loss* no es muy significativa respecto al error de reconstrucción. En este caso el error de reconstrucción se calculo usando *binary_crossentropy*.

A continuación se pueden ver ejemplos de imágenes originales y las reconstrucción obtenida con el VAE, Fig. 3. Las imágenes reconstruidas mantienen los rasgos principales de los dígitos originales (en general), aunque presentan un leve suavizado.

Figura 3: Ejemplo de reconstrucción de imágenes con el VAE.

Luego lo que se hizo fue samplear el espacio latente y examinar las salidas del decodificador, esto se hizo con valores de z entre -2 y 2. En la Fig. 4(a) se puede observar esto. En la Fig. 4(b) se puede ver la representación del espacio latente del conjunto de test. Se pueden observar conjuntos por clase, habiendo superposición entre algunos de estos grupos. Esto ultimo es esperable si se tiene características compartidas entre números diferentes, como por ejemplo ocurre con el 9 y 4, los cuales tiene nubes que se solapan en gran parte.

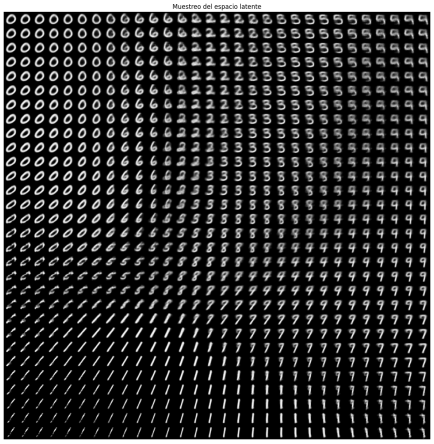
Luego se entreno el modelo usando como error de reconstrucción *mse*. En las Fig. 5 se puede observar la evolución de las perdidas en el train y validation datasets. Se obtuvo de valores de loss finales:

Loss total en train: 35.103 || Loss total en validation: 34.575
Loss total en test: 35.366

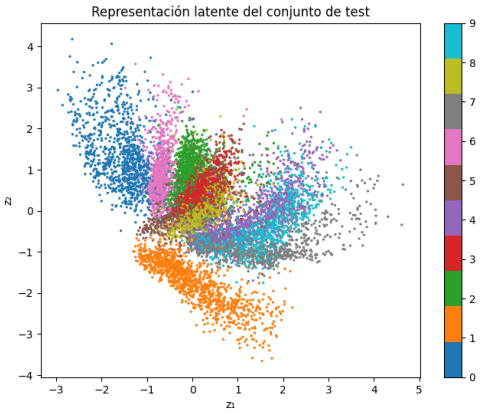
Error reconstrucción en train: 32.216 || Error reconstrucción en validation: 31.682
Error reconstrucción en test: 32.482

KL loss en train: 2.887 || KL loss en validation: 2.893
KL loss en test: 2.884

A continuación se pueden ver ejemplos de imágenes originales y las reconstruccion obtenida con el VAE, Fig. 6. Se puede ver como antes que las imágenes reconstruidas presentan un leve suavizado, no tan intenso como con la loss BCE.

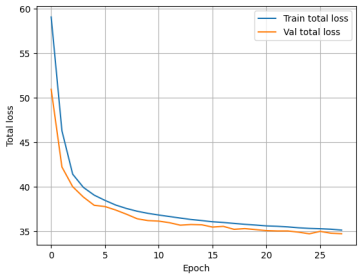


(a) Sampleo del espacio latente con el decoder.

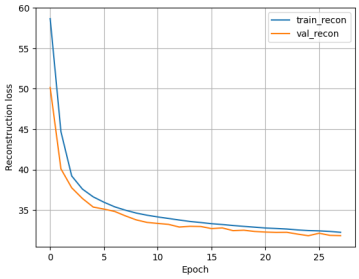


(b) Representación latente del conjunto de test.

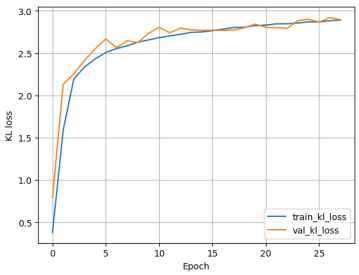
Figura 4: Análisis del espacio latente.



(a) Error total.



(b) Error de reconstrucción con mse.



(c) KL loss.

Figura 5: Evolución de errores a través de las épocas.

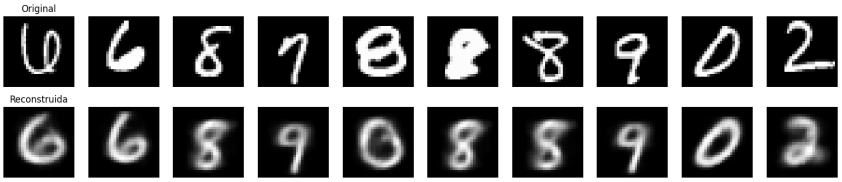
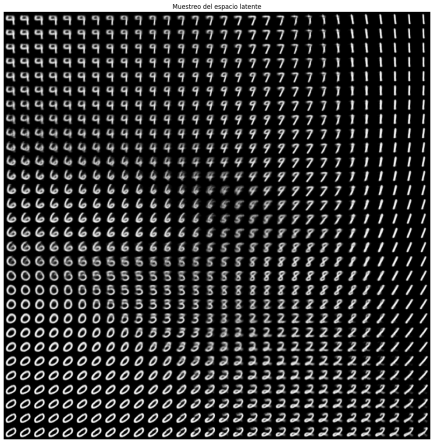


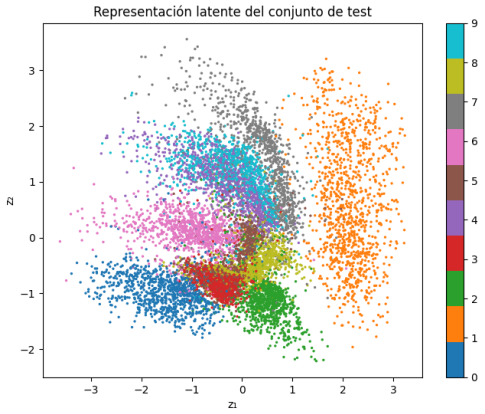
Figura 6: Ejemplo de reconstrucción de imágenes con el VAE.

Luego lo que se hizo fue samplear el espacio latente y examinar las salidas del decodificador, esto se hizo con valores de z entre -2 y 2. En la Fig. 7(a) se puede observar esto. En la Fig. 7(b) se puede ver la representación del espacio latente del conjunto de test.

En este caso también se pueden observar grupos correspondientes a los diferentes números, habiendo superposición entre números con características similares como el 4 y 9.



(a) Sampleo del espacio latente con el decoder.



(b) Representación latente del conjunto de test.

Figura 7: Análisis del espacio latente.

Se puede ver que a pesar de cambiar la función que usamos para el error de reconstrucción, obtuvimos buenos resultados. En este caso las imágenes reconstruidas están menos difuminadas que usando BCE. En ambos casos se observa en el espacio latente grupos correspondiente a las diferentes clases, para el caso del MSE, al dispersión de los valores de z son menores que con BCE.

2. Apéndice

Los códigos con los ejercicios resueltos se encuentran en el repositorio: `https://github.com/AnaMorresi/DeepLearning`.