Deep Learning Autoencoders

Indhira Ramirez
UTEC
Lima, Perú
indhira.ramirez@utec.edu.pe

Mauricio Nieto
UTEC
Lima, Perú
mauricio.nieto@utec.edu.pe

I. Objetivos

- Eliminar el ruido en imágenes de texto a través del uso de autoencoders.
- Desarrollar un modelo basado en mlps como autoencoders.
- Desarrollar un modelo basado en CNNs como autoencoders
- Utilizar $k\ fold\ cross\ validation$ en la etapa de entrenamiento de cada modelo
- Comparar resultados de accuracy, error y tiempo

II. Experimentación

A. Datos

- Todos las fotos del train contienen el mismo texto, con distinto tipo de fuente y diferente tipo de ruido. El test contiene un texto diferente del train. Esto es importante pues la similitud de los datos en train podría provocar un overfitting en el modelo.

B. MLP

1) Arquitectura:

Esta red neuronal presenta un encoder y decoder que poseen una simetría en su estructura. Además, esta estructura consta de 2 capas lineales en el encoder y decoder que realizan una reducción de dimensionalidad hasta 1800 dimensiones. Debido al tiempo de computo y necesidad de memoria para correr esta arquitectura se redujo bastante las dimensiones de la imagen a un formato de 180 * 100 pixeles en la imagen.

se intento mantener lo más sencillo la arquitectura para así evitar el overfitting causado por la naturaleza de los datos.

C. CNN

Se desarrollaron dos arquitecturas de CNN para resolver el objetivo principal de eliminación de ruido. Y aunque ambas sean similares por su naturaleza de CNN, se diferencian en la forma en que codifican la información y como logran decodificarla.

1) Arquitectura 1: La primera red neuronal cuenta con un encoder y un decoder, pero tiene la particularidad de que luego del decoder implementa una serie de convoluciones encapsuladas en la clase Process, la cual aplica convoluciones extra al resultado del output del decoder.

La parte del encoder posee 3 capas de convolución a las que procede de una capa lineal, no obstante esta capa lineal no realiza un flatten a las imágenes que obtiene, tan solo reduce su dimensionalidad. De igual modo, cada capa de convolución aplica una convolución con valores sencillos que son: kernel en 3, stride en 1 y padding en 1. Igualmente, cada capa de convolución aumenta el número de outputs que emite para ello esta la capa lineal del final, que actúa como contrapeso y evita que la dimensionalidad del tensor resultante del encoder sea demasiado alto.También es importante resaltar que en las 2 primeras capas de convolución se aplica maxpool y los índices resultantes son guardados para usarlos posteriormente en el decoder.

Al igual que en el encoder, el decoder consta de una capa lineal opuesta a la del encoder para recuperar la dimensionalidad perdida, seguido de de 3 capas de convolución transpuesta que contienen los kernels, strides y paddings necesarios para aumentar la dimensionalidad de forma opuesta al encoder. Además, se aplica un unpool en las dos últimas capas de convolución, donde se utilizan los índices anteriormente mencionados para así ejecutar su tarea de forma correcta.

Después encontraremos el Process el cual es un conjunto de 4 capas de convolución con valores de kernels, strides y paddings en 3, 1 y 1. El número de outputs por convolución es también bajo no pasando de 5. Estas capas de procesamiento extra mejoran bastante el resultado del decoder a pesar de ser un agregado relativamente sencillo. De igual manera, al ser un conjunto de capas sencillas no aumentan de forma notable el tiempo de ejecución.

2) Arquitectura 2: En la arquitectura 1 la reducción de dimensionalidad no es tan grande como podría parecer en un inicio, pues mientras que los pool reducen el tamaño de las imágenes el output de cada convolución aumenta, manteniendo en cierta medida la dimensionalidad del tensor entrante. De igual manera, no se aplica un flatten dentro de la arquitectura aunque es posible aplicar una entre encoder y decoder. Son esos aspectos que ignora la primera arquitectura en los que se enfoca la segunda. En cuestión de capas y sus especificaciones, esta arquitectura es muy similar a la anterior. No obstante, se caracteriza por aplicar un flatten en la capa lineal del encoder y la operación inversa en la capa lineal del decoder. Asimismo, este flatten ayuda a reducir la dimensionalidad hasta tan solo 20 dimensiones numéricas y los índices producidos por los dos pools aplicados en el encoder.

III. Resultados

A. MLP

Con la arquitectura MLP no se encontró buenos resultados, pero observamos que si se diferenciaban las figuras de la Tabla 1.

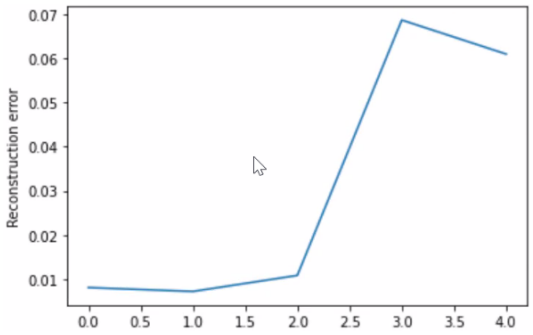

Fig. 1. Loss por k-fold, con k = 10, en la MLP

B. CNN

En el desarrollo de los experimentos se encontraron bastantes observaciones que variaron para cada arquitectura de CNN.

1) Arquitectura 1: Una de las cosas más importantes que encontramos para ambas arquitecturas es que el learning rate es muy influyente a la hora de obtener resultados. Pues, el loss function tiende a estancarse en valores como 0.06 y 0.07, si se usa el modelo para hacer una predicción en esa situación se obtendrá una imagen totalmente blanca, esto se debe a la naturaleza de los datos que contienen una gran superficie totalmente blanca restándole importancia a la superficie que contiene el texto. En caso de que el loss function aparentará estancarse entre estos valores, la mejor opción sería volver a comenzar el entrenamiento variando ligeramente el learning rate. Asimismo encontramos que para esta primera arquitectura el decoder daba mucha importancia a los índices generados por el pool, especialmente los del primer pool. Igualmente, también había fuerte importancia para el decoder en el resultado comprimido que emitía el encoder. Por otro lado, al momento de aplicar un k-fold cross validation se encontró que requería de mucho tiempo de ejecución. Por ello se decidió que el k del k-fold sea de 3, tal como se ve en la figura 2. Igualmente se puede ver como hay gran variación entre loss functions alcanzados, esto debido a la alta influencia que tiene el learning rate y que en algunos casos puede estancar el loss function en valores de 0.06, tal como muestra a figura 2. Los resultados se pueden observar en la Tabla 2.

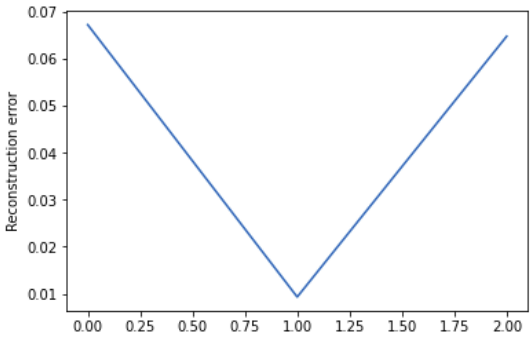

Fig. 2. Loss por k-fold, con k = 3, el arquitectura 1 de CNN

2) Arquitectura 2: En esta arquitectura, debido a la reducción extrema de dimensionalidad, los índices producidos por el pool han tomado demasiada importancia dejando casi en obsoleto el output comprimido del encoder. No obstante, el hecho de que el encoder tuviera 20 dimensiones como output y no el mínimo, que sería 1, ayudo al modelo a generalizar y poder limpiar el ruido con mayor facilidad. Por otro lado, esta arquitectura es muy susceptible al overfitting, esto se refleja fuertemente en el loss function que presenta mucha variación entre resultados conforme avanza el entrenamiento. Por ello es recomendable detener el entrenamiento en cuanto empiece a generar esta variación. Los resultados se pueden observar en la Tabla 3. Igualmente, en la figura 3 se vuelve a ver que hay mucha variación de resultado esto también se debe a la alta dependencia del learning rate para reducir el loss function.

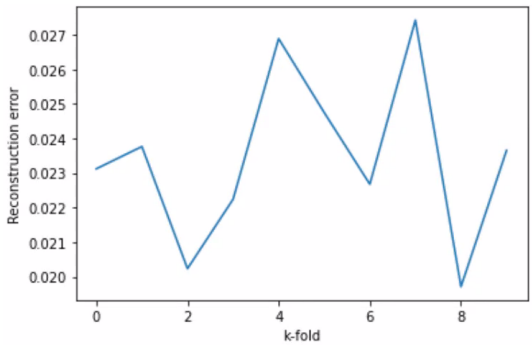

Fig. 3. Loss por k-fold, con k = 5, en la arquitectura 2 de CNN

IV. Conclusión

- Las CNN son mucho mas efectivas para quitar el ruido que las MLP.
- Fue mas complicado entrenar las redes porque la data de entrenamiento eran las misas, y las de test eran diferentes.

TABLE I
RESULTADOS CON MLP

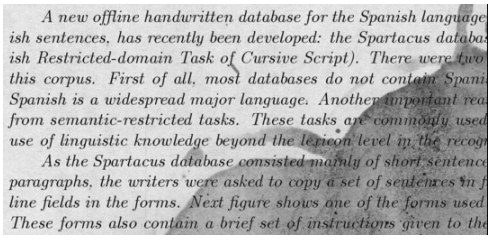

Fig. 4. Foto original

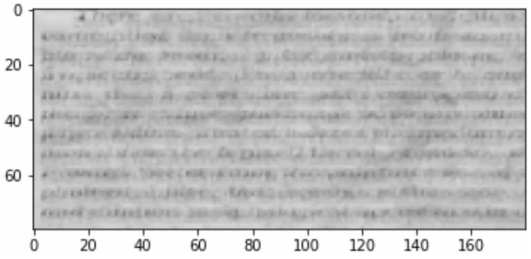

Fig. 5. Foto después del encoder

TABLE II
RESULTADOS CON CNN (ARQUITECURA 1)

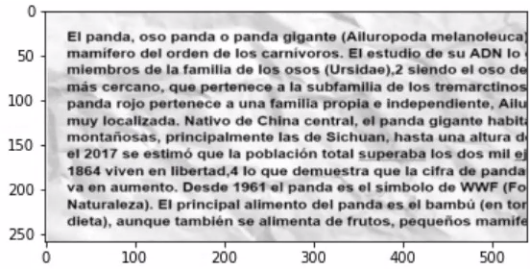

Fig. 6. Foto original

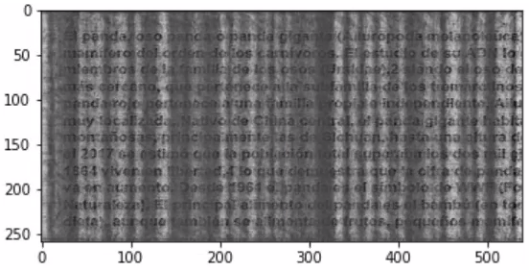

Fig. 7. Foto después del encoder

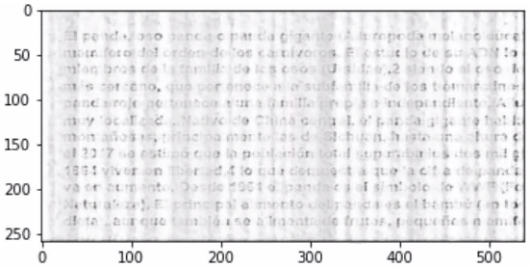

Fig. 8. Foto después del decoder

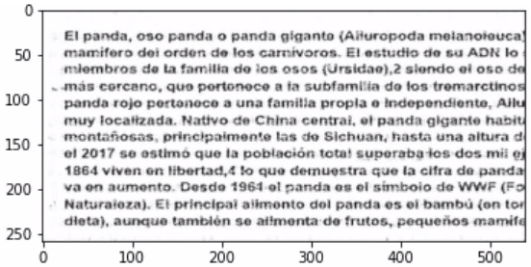

Fig. 9. Foto después de process

TABLE III
RESULTADOS CON CNN (ARQUITECURA 2)

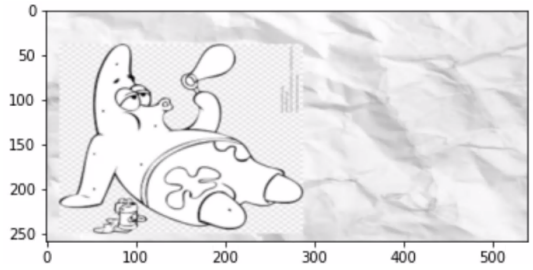

Fig. 10. Foto original

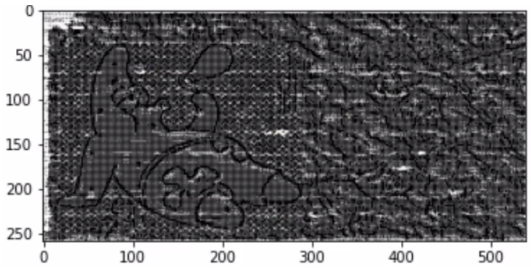

Fig. 11. Foto después del encoder

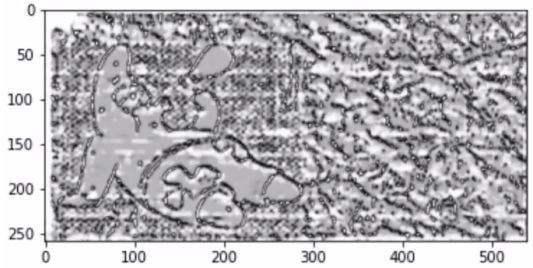

Fig. 12. Foto después del decoder

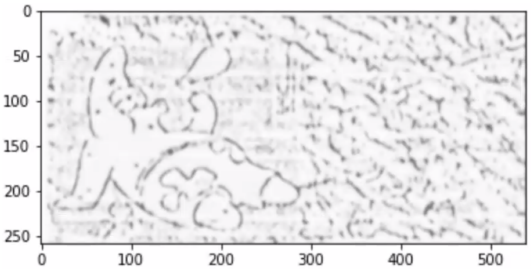

Fig. 13. Foto después de process