Redes neuronales: Autoencoder sobre FashionMNIST

Daniel A. Arias [1]

[1] *Universidad de Buenos Aires, FCEyN*
max.daniels77@gmail.com

ABSTRACT

Los *autoencoders* son un tipo de red neuronal cuyo objetivo es regenerar a la salida un determinado *input* de entrada. En el presente trabajo, se generaron dos sobre el dataset *Fashion-MNIST*, con distintos optimizadores, adam y SGD, para los cuales se vario el *learning rate* y el tamaño de la capa oculta en $64, 128, 256$ y 512 neuronas. Finalmente se utilizaron las redes mejores entrenadas con ambos optimizadores para filtrar imágenes del dataset a las cuales se les añadió de manera artificial un ruido gaussiano.

Keywords: Redes Neuronales — Autoencoders — Modelado Numérico — *Fashion-MNIST*

1. INTRODUCCIÓN

Los *autoencoders* son un tipo de red neuronal utilizada para aprender de manera eficiente problemas de aprendizaje no supervisado. De acuerdo con la figura 1, un autoencoder consiste en dos partes principales, un *encoder* que mapea la información de entrada a código y un *decoder* que decodifica esta información. Un *autoencoder* ideal realizaría una reconstrucción lo más cercana posible a la perfección. La forma más sencilla de realizar este proceso de manera perfecta, sería duplicar la señal de entrada. Para suprimir este comportamiento (ya que de lo contrario nuestro modelo estaría aprendiendo de los errores), el espacio de código, en nuestro caso, la dimensión de la o las capas ocultas debe ser menor que la dimensión de la información de entrada, generando un cuello de botella. De acuerdo con Hinton & Zemel (1994) este tipo de *autoencoders* se denomina subcompleto y puede interpretarse como una compresión del mensaje, o una reducción de su dimensionalidad .

Si el espacio de código o dimensión de la o las capas ocultas tiene una dimensión mayor, o igual, que el espacio de la dimensión de entrada, o las capas ocultas tienen suficiente capacidad, un *autoencoder* de este tipo, llamado sobrecompleto puede aprender la función de identidad y volverse inútil. Sin embargo, los resultados experimentales han demostrado que los *autoencoders* sobrecompletos pueden aprender características útiles Bengio (2009).

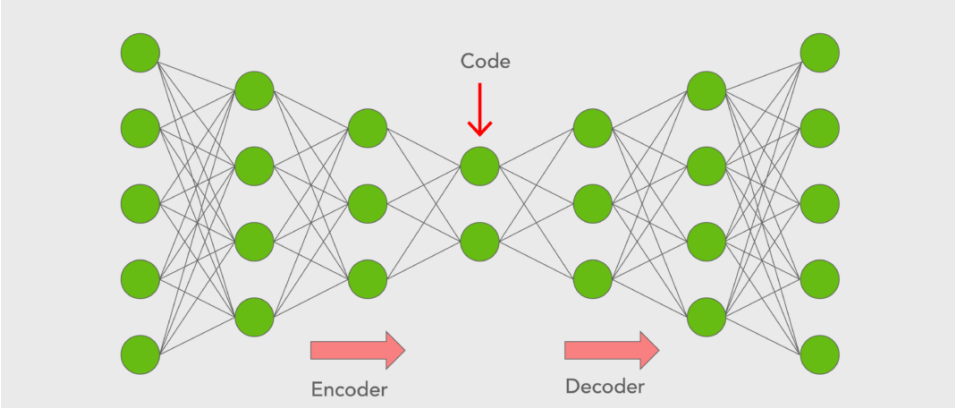
En este trabajo se generarán dos redes *feed-forward autoencoders* (en adelante, solo *autoencoders*), con una única capa interna, utilizando como optimizadores adam y *SGD* sobre el dataset *Fashion-MNIST*. En ambos casos, por medio de *grid search* se variaran *learning rates* η y el número de neuronas de la capa interna para 100 épocas, con el objetivo de hallar los mejores hiperparámetros de la red. Finalmente, se comparará el resultado de ambos modelos, usando tanto adam como SGD para los mejores parámetros de redes hallados con el objetivo de filtrar ruido gaussiano generado de manera artificial.

2. MODELADO NUMÉRICO

2.1. *Dinámicas utilizando SGD*

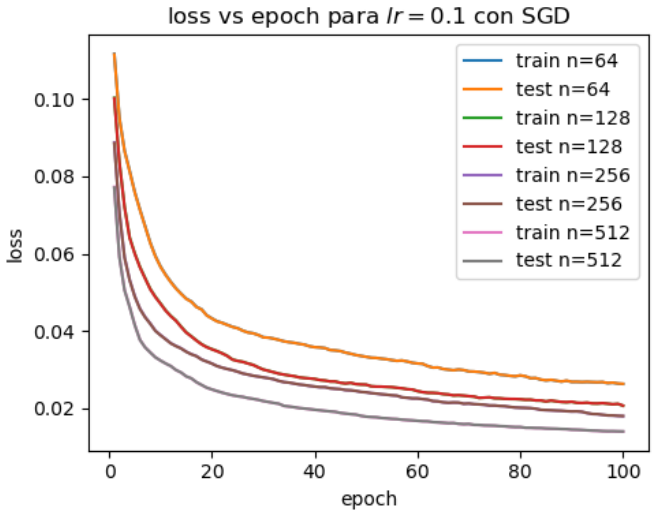
La primera etapa del trabajo se realizó en partir del setup básico provisto, el cual consistía en utilizar como función de pérdida (o *loss*), el error cuadrático medio, SGD (*stocastic gradient descent*) como optimizador, una capa oculta de 64 neuronas, un *minibatch* de 1000 y un *dropout* con $p = 0.1$, el cual consiste en apagar de manera aleatoria un porcentaje p de las neuronas en cada época del entrenamiento para introducir aleatoriedad en el modelo y como función de activación ReLu. Tanto la función de perdida, el *dropout*, la función de activación, como el numero de épocas (fijado en 100), permanecerán invariantes para el resto del experimento. Además, en esta primera instancia, se fijo un *learning rate* $\eta_1 = 0.001$. Estos valores no arrojaron los resultados esperados[1].

[1] Se omiten las correspondientes imágenes para no superar las 4 paginas en el documento

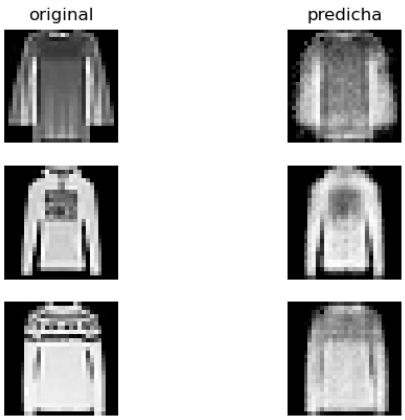


(a)

Figure 1. En esta figura, se puede apreciar el esquema básico de un autoencoder, donde la cantidad de neuronas a la entrada, es la misma que a la salida. En ella se aprecian 5 capas ocultas las cuales conforman las dos partes principales, tanto *encoder* como *decoder*. Además de puede observar el cuello de botella antes mencionado generado por las reducción y posterior ampliación de la cantidad de neuronas de las capas ocultas.



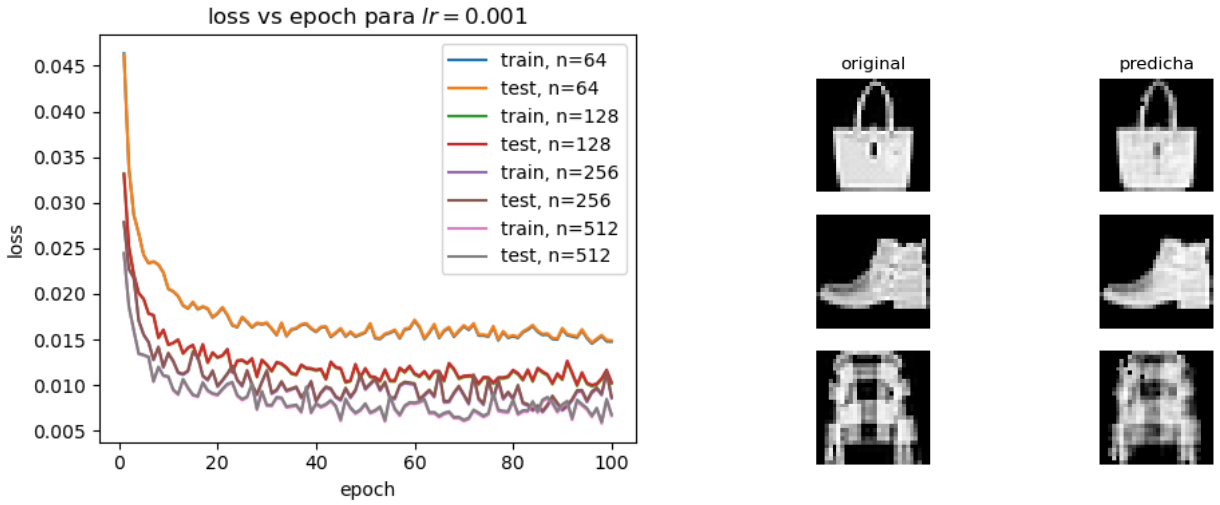
(a)



(b)

Figure 2. En la figura a, se puede apreciar el resultado del *grid search* utilizando como optimizador SGD, en ella se puede apreciar que el mejor lr obtenido fue $\eta_1 = 0.1$ asi como las curvas de loss para los distintos conjuntos de neuronas. En la figura b se puede apreciar el desempeño del modelo con menor loss (512 neuronas), al comparar la imagen original vs la predicha por el *autoencoder*, en ella se aprecia que si bien el modelo representa relativamente bien el *input* de entrada, tiene problemas ya que las predicciones son ligeramente ruidosas así como en algunos casos muy suavizadas.

Con el objetivo de solventar el problema antes comentado, se procedió en primera instancia a realizar *grid search* sobre otros dos *learning rates*, $\eta_2 = 0.1$ y $\eta_2 = 0.01$ así como variar el numero de neuronas de la capa interna a fin de seleccionar el modelo que tenga menor perdida (se añadieron 128, 256 y 512 neuronas a la búsqueda en grilla), esto puede ser apreciado en la figura 2a, en la cual se puede apreciar el *loss* o pérdida para distintos números de neuronas con el mejor η obtenido, mientras que en la figura 2b, se puede apreciar su funcionamiento prediciendo sobre 3 valores tomados de manera aleatoria del conjunto de testeo.



(a) (b)

Figure 3. En la figura a, se puede apreciar el resultado del *grid search* utilizando como optimizador adam, en ella se puede apreciar que el mejor lr obtenido fue $\eta_1 = 0.001$ así como las curvas de loss para los distintos conjuntos de neuronas. En la figura b se puede apreciar el desempeño del modelo con menor loss (512 neuronas), al comparar la imagen original vs la predicha por el *autoencoder*, en ella se aprecia que el modelo representa de manera adecuada el *input* de entrada.

2.2. *Dinámicas utilizando adam*

En esta etapa, se decidió utilizar otro optimizador dado que se buscaba mejorar los resultados obtenidos por lo que se paso a utilizar el algoritmo adam Brownlee (2020), el cual es una implementación en *pytorch* que calcula *learning rates* o tasas de aprendizaje adaptativas individuales para diferentes parámetros a partir de estimaciones del primer y segundo momento de los gradientes. Este método permitió obtener mejores resultados de manera rápida ya que en la primera época obtiene resultados similares a SGD para ciertas configuraciones de hiperparámetros. Esto permite aplicar algún criterio, por ejemplo el del codo o evaluar la ganancia porcentual de *loss* con el objetivo de limitar el numero de épocas necesarias para entrenar el modelo dado que el loss a partir de la época 30 se mantiene de manera relativamente constante. Este resultado se puede apreciar en la figura 3a la cual al igual que en el caso anterior fue obtenida al realizar *grid search* variando η y el numero de neuronas donde los mejores resultados se obtuvieron para $\eta = 0.001$, en ella, se puede apreciar los valores de *loss* para distintas épocas y numero de neuronas n. Por otro lado en la figura 3b, se puede apreciar la mejora del modelo al cambiar el optimizador utilizado.

2.3. *Implementación del modelo como filtro de ruido gaussiano*

Una de las ventajas de utilizar un autoencoder, es que permite obtener una salida robusta frente a errores, esto puede ser explicado bien por el cuello de botella generado al bajar la dimensión de las capas internas como también por el dropout el cual permite reducir la varianza al introducir aleatoriedad al modelo. Estas características permiten utilizar el *autoencoder* como filtro el cual limpie imágenes ruidosas.

Se partió de las imágenes del *dataset Fashion-MNIST* a las cuales se les sumó de manera arbitraria un ruido gaussiano con valor medio y desviación, las propias de cada imagen. Dicho ruido fue modulado por medio de una constante para poder variar la intensidad del ruido. Luego, utilizando los 2 mejores modelos entrenados previamente, SGD con $\eta = 0.1$ y 512 neuronas y adam con $\eta = 0.001$ y 512 neuronas. se procedió a comparar en la figura 4 el desempeño de ambos modelos filtrando el ruido. En ellas se aprecia que el desempeño del modelo que implementa adam como optimizador fue mejor al filtrar el ruido.

3. DISCUSIÓN

Si bien, para el caso donde se utilizo el algoritmo adam, el resultado fue satisfactorio, hay algunos detalles a tener en cuenta, de acuerdo con la figura 3a, no es necesario entrenar el modelo por 100 épocas para obtener resultados similares ya que por simple inspección visual se puede apreciar que a partir de le época 30, los valores de loss se mantienen de

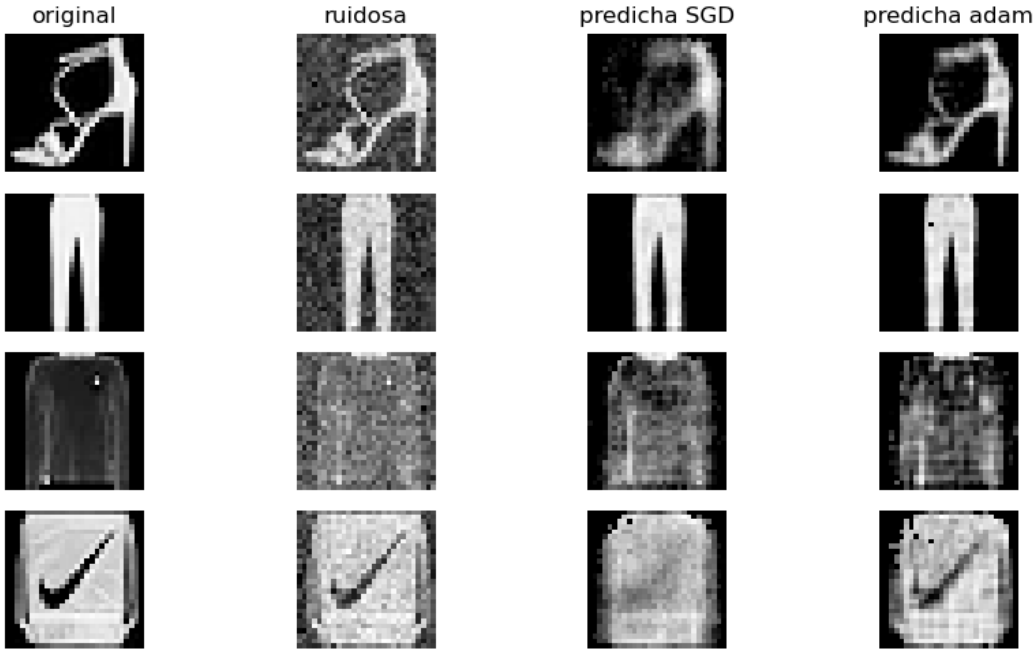


Figure 4. En la figura, se puede apreciar la comparativa en el desempeño como filtro de ruido gaussiano, para los modelos entrenados con SGD y adam. En la primera fila, se puede apreciar un caso donde el autoencoder con adam representa mejor la imagen original. En la segunda una donde el autoencoder con SGD funciona mejor. En la tercera una donde ambos fallan y en la cuarta, una donde el modelo con SGD no logra captar la reconocida pipeta mientras que el modelo con adam logra captarla. Es necesario aclarar que las imágenes fueron tomadas de manera arbitraria del conjunto de testeo para ejemplificar casos en donde se vean las virtudes y falencias de ambos casos.

manera relativamente constante, lo cual hace que la mejora real frente al tiempo de computo sea leve. Esto no sucede para el caso en que se utiliza como optimizador a SGD ya que en ese caso añadir mas épocas probablemente generaría una mejora, pero difícilmente supere a adam por características propias de construcción. Respecto del *denoising* si bien el modelo regenera las imágenes en un contexto ruidoso seria bueno observar como responde el modelo a otro tipo de ruidos.

4. CONCLUSIONES

Se logró implementar el *autoencoder*, se comprobó que el uso del adam como optimizador es mas eficiente frente a SGD ya que permite obtener mejores resultados en menor cantidad de épocas, esto se traduce en menor tiempo y costo computacional. Se verifico el uso del *autoencoder* como filtro de ruido.

BIBLIOGRAFÍA

Bengio, Y. 2009, Found. Trends Mach. Learn., 2, 1, doi: 10.1561/2200000006

Brownlee, J. 2020. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Hinton, G. E., & Zemel, R. S. 1994, in Advances in Neural Information Processing Systems 6, ed. J. D. Cowan, G. Tesauro, & J. Alspector (Morgan Kaufmann), 3–10