Redes Neuronales: Práctico 3

María Florencia Molina

Febrero 2022

1. Introducción

En este informe se describirán los resultados obtenidos por una red neuronal feed-forward autoencoder con una capa oculta para aprender la funcion identidad, utlizando la base de datos de Fashion-MNIST, que consiste en una serie de imagenes de prendas y calzados de 28*28 pixeles. Se utilizan primero 64 neuronas para la capa oculta, y luego se utilizan sucesivamente L=128,256,512 neuronas para esta capa, con el objetivo de observar cuál de los modelos propuestos ajusta mejor la funcion identidad con esta base de datos.

2. Implementación del autoencoder

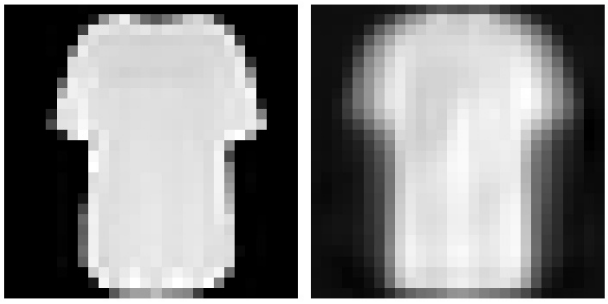
Utilizamos PyTorch para la implementación de nuestra red, mediante el comando `torch.nn.Sequential()`. La primera capa y la ultima capa son ambas de 784 neuronas, y se utilizó la función de activación ReLU para cada capa de neuronas, implementando un dropout de $p = 0{,}1$ antes de cada capa. Se usaron 60000 imagenes para el conjunto de entrenamiento y 10000 para el conjunto de testeo. Ambos conjuntos fueron separados en batchs de 1000 imagenes. El learning rate elegido fue de 0.001 y se usaron 200 épocas para el entrenamiento de la red iterando con distintos tamaños de L, mientras que para mostrar las imagenes obtenidas por el autoencoder y el error de entrenamiento y testeo de cada red por separado se usaron 50 épocas (ya que el entrenamiento para 200 épocas llevó mucho tiempo y significó un costo computacional significativo). El optimizador utilizado fue `torch.optim.Adam()`, que utiliza el método de descenso por el gradiente estocástico, siendo muy eficiente para redes con un gran numero de datos y/o parámetros, como la que se utilizó para este informe. La función `loss` utilizada fue la de error cuadrático medio.

3. Distintos tamaños para la capa intermedia con 50 épocas de entrenamiento

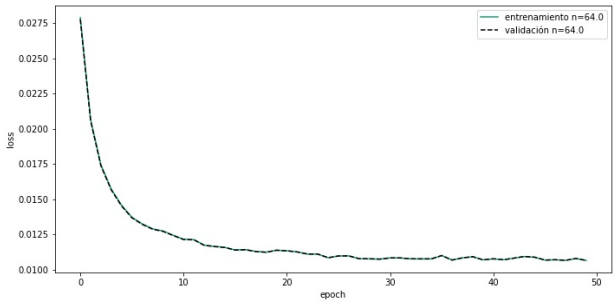
Para cada valor propuesto de L en la capa intermedia, se graficó el descenso del error cuadrático medio de los conjuntos de entrenamiento y de validación, junto con un ejemplo elegido al azar de las imagenes del conjunto de entrenamiento comparada con la predicción realizada por el autoencoder al finalizar las épocas de entrenamiento para verificar el funcionamiento del mismo. En todos los casos se obtuvieron imagenes que se aproximaban claramente a los datos correspondientes del input del conjunto de entrenamiento, por lo que se puede deducir preliminarmente que el autoencoder implementado realizaba la tarea de forma apropiada en el sentido de que arrojaba resultados pertinentes. Tambien se observaba que el error cuadrático medio descendía hasta estacionarse en un estado de estabilidad, sin experimentar fluctuaciones apreciables o significativas.

3.1. L=64

Implementando un total de 64 neuronas de capa intermedia se obtuvieron los siguientes resultados



(a) Imagen elegida al azar de la base de datos junto con su predicción arrojada por el autoencoder

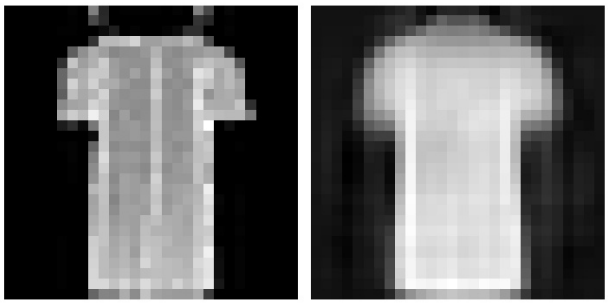


(b) Descenso del error cuadrático medio para los conjuntos de validación y de testeo

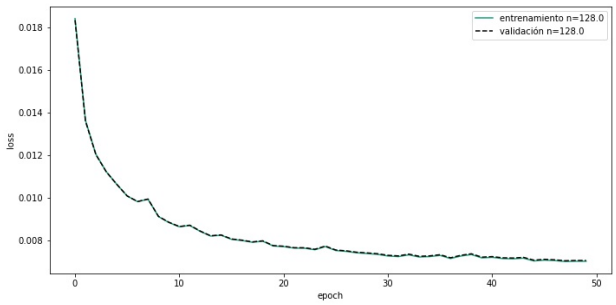
Figura 1: L=64

3.2. L=128

Implementando un total de 128 neuronas de capa intermedia se obtuvieron los siguientes resultados



(a) Imagen elegida al azar de la base de datos junto con su predicción arrojada por el autoencoder



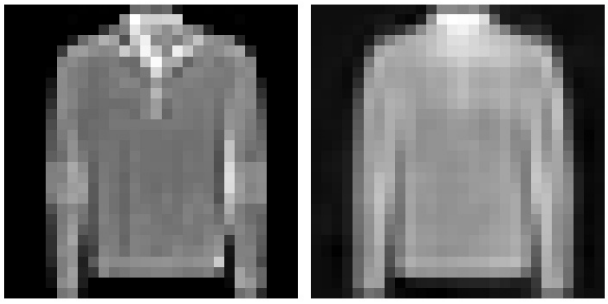
(b) Descenso del error cuadrático medio para los conjuntos de validación y de testeo

Figura 2: L=64

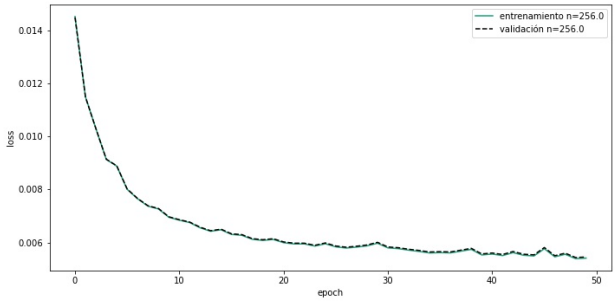
Se observa que la predicción de la imagen arrojada por el modelo se ha vuelto mas definida que para el número anterior de neuronas de capa oculta.

3.3. L=256

Implementando un total de 256 neuronas de capa intermedia se obtuvieron los siguientes resultados



(a) Imagen elegida al azar de la base de datos junto con su predicción arrojada por el autoencoder



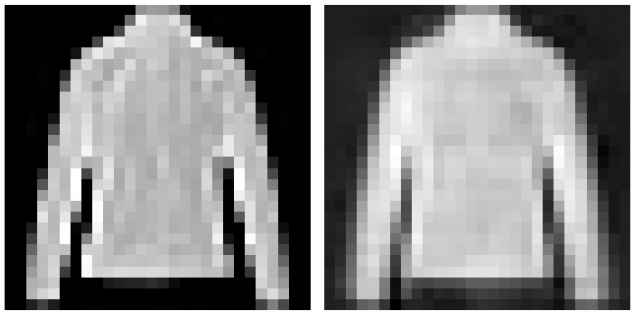
(b) Descenso del error cuadrático medio para los conjuntos de validación y de testeo

Figura 3: L=64

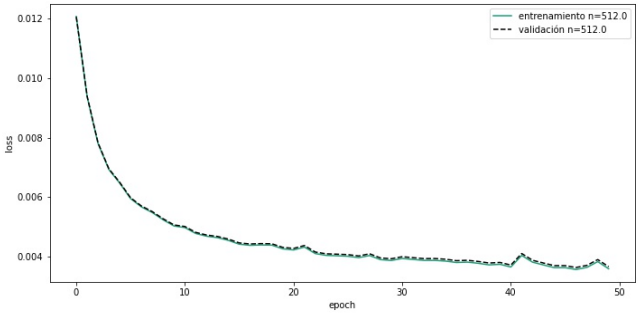
Se observa, al igual que antes, cómo mejora la predicción arrojada por el autoencoder.

3.4. L=512

Implementando un total de 512 neuronas de capa intermedia se obtuvieron los siguientes resultados



(a) Imagen elegida al azar de la base de datos junto con su predicción arrojada por el autoencoder



(b) Descenso del error cuadrático medio para los conjuntos de validación y de testeo

Figura 4: L=64

4. Distintos tamaños de L para la capa intermedia con 200 épocas de entrenamiento

Se modificó el código de forma que la implementación itere entre los distintos tamaños elegidos para la capa intermedia de forma de obtener una comparación entre los distintos modelos para el descenso del error cuadrático medio. Se obtuvo lo siguiente

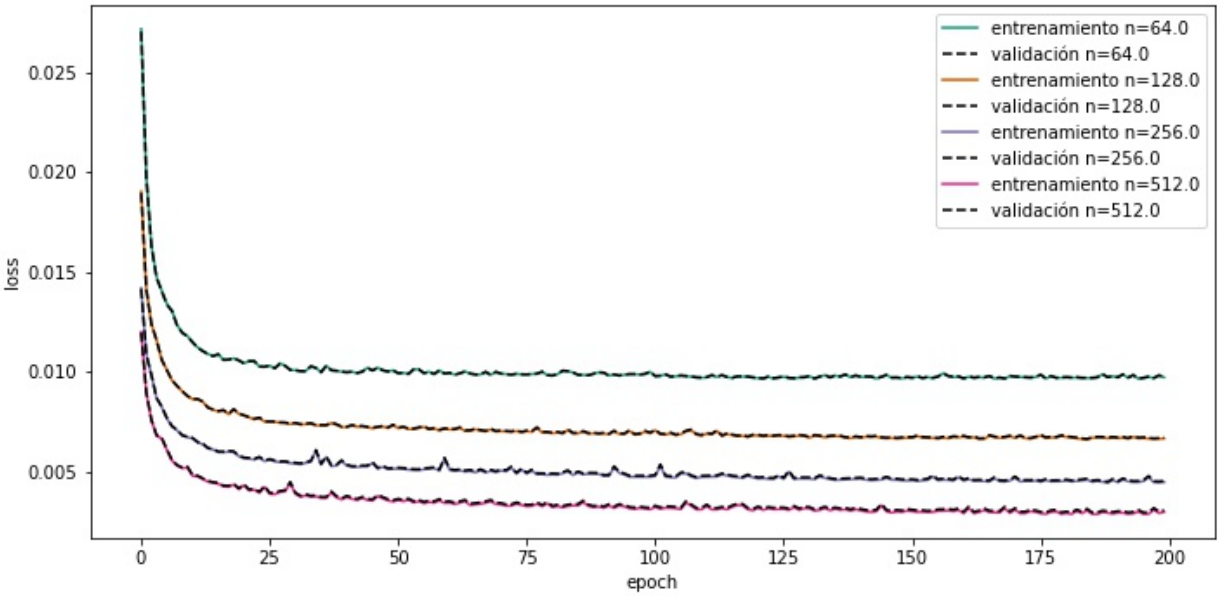


Figura 5: Errores de validación y de entrenamiento para distintos tamaños de capa oculta

Se observa que cuanto mayor sea el número de neuronas de la capa oculta, mas rápido es el descenso del error cuadrático medio, y estaciona a valores estables sucesivamente mas bajos. Esto coincide con las imágenes obtenidas por cada modelo, de los cuales hemos plasmado ejemplos en este informe, pues es apreciable como mejora la predicción del autoencoder en los ejemplos que hemos puesto para cada L. También se observa que pasadas aproximadamente entre 50 y 70 épocas el error no desciende significativamente para ninguno de los cuatro modelos, por lo que realizar mas épocas no es conveniente considerando el costo computacional. En todos los gráficos se observa que los errores de validación y de testeo estan muy próximos uno de otro, siendo el error de testeo apenas menor que el de validación (esto es mas notorio en el caso de L=512). Esto indica que el modelo no sufre de overfitting ni underfitting.

5. Conclusión

Se logró implementar exitosamente un autoencoder con los parámetros requeridos para distintos tamaños de capa oculta. Se observa que el modelo ajusta mejor la función identidad con 512 neuronas en la capa oculta.