Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía, Física y Computación
Curso de Posgrado “Redes Neuronales”
Docentes: Dr. Francisco Tamarit, Dr. Juan Ignacio Perotti
Alumna: María Virginia Romero Messein

Trabajo Práctico Nº 3

Este informe tiene el objetivo de exponer el modo de implementación de una red neuronal feed-forward autoencoder para aprender la función identidad de un conjunto de imágenes. Dichas imágenes provienen de la base de datos Fashion MNIST y se dividen en 10 categorías de diversas prendas y calzados. Esta base de datos tiene 60.000 imágenes en el conjunto de entrenamiento y 10.000 imágenes en el conjunto de test. Para entrenar la red se utilizará la biblioteca de aprendizaje automático PyTorch.

En el tipo de red neuronal feed- forward las neuronas están acomodadas en capas sucesivas. Es decir, las conexiones son unidireccionales y la información siempre va hacia delante. Hay una capa de entrada, una capa intermedia y una capa de salida.

Por otra parte, autoencoder es una red neuronal artificial entrenada de manera no supervisada, que se caracteriza por aprender la función identidad. Primero, la red aprende codificando los datos de entrada (input) a través de la reducción de la información. Luego, genera datos de entrada decodificando representaciones cercanas de aquello que aprendió. De esta manera, el output del autoencoder es su predicción para la entrada. En la figura 1, se observa que el encoder comprime la información de entrada. Luego se encuentra un espacio latente en el que la información es representada dentro de la red neuronal. En tercer lugar, el decoder reconstruye la entrada lo mejor posible en base a la información obtenida.

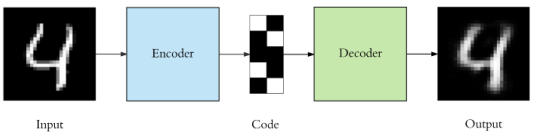

Figura 1: Autoencoder

Si bien en este trabajo el objetivo es que la red aprenda la función identidad minimizando el error, su predicción no será exactamente igual a la imagen original dado que siempre queda un error. Para alcanzar la función de error cuadrático medio se utiliza el método del descenso del gradiente estocástico (SGD).

En el proceso de entrenamiento de la red se procederá por un lado, a reducir el error sobre 50.000 imágenes de entrenamiento. Por el otro, se llevará a cabo la etapa de testeo sobre otras 10.000 imágenes que no se utilizaron para

entrenar. Aquí se comprueba si el modelo aprendió la identidad visualizando aquellas imágenes que nunca vio. Si los errores del conjunto de entrenamiento y de testeo tienen valores similares se puede considerar que el aprendizaje está siendo efectivo. Por el contrario, si el error de entrenamiento disminuye y el de testeo aumenta, se produce el overfitting. Esto implica que el modelo no está aprendiendo la identidad de las imágenes sino que las está memorizando ya que cuando se le muestran imágenes que nunca vió no es capaz de reconstruirlas.

El entrenamiento de la red cuenta con los siguientes hiperparámetros:

-La *tasa de entrenamiento,* determina la rapidez con la que la red neuronal converge a los mínimos. Cuanto más alto es su valor, más rápido avanzará el algoritmo del gradiente descendiente.

-La cantidad de *épocas,* determina cuánto tiempo andará el algoritmo de entrenamiento. En cada época los datos de entrenamiento pasan por la red neuronal para que ésta aprenda sobre ellos.

-El *batch* indica el número de datos que tiene cada iteración de una época. De esta manera, se actualizan los parámetros de la red neuronal.

-El *dropout* se utiliza para evitar el overfitting, es decir, reducir el sobreajuste en la red neuronal.

La red tiene la siguiente arquitectura: una capa de entrada con 784 unidades (cada imagen es de 28*28 píxeles en escala de grises) una capa intermedia con 64 neuronas y una capa de salida con 784 neuronas. La tasa de entrenamiento es de $1e-1$; el dropout: p =0.1 ; el batch es de 1000; las candidad de épocas es 200.

En la figura 2 se observa que la función error del entrenamiento y del testeo son similares, lo cual indica que la red está aprendiendo. El error desciende abruptamente hasta las primeras 25 épocas aproximadamente. Desde la época 50 a la 200 el error empieza a descender de manera más paulatina.

A modo de muestra en la figura 3 se presenta una imagen original elegida al azar y y la correspondiente la imagen generada por el modelo.

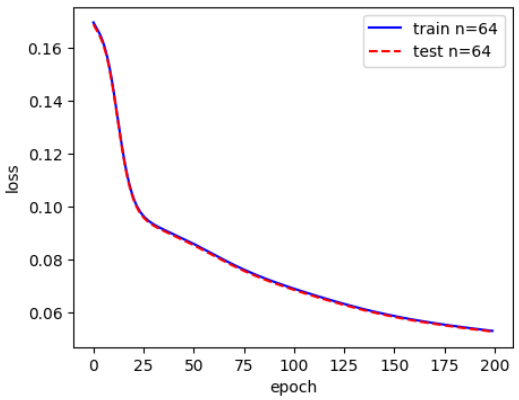

Figura 2: n= 64 - Época 199→ train_loss: 0.053142761687437694 test_loss: 0.05289321728050709

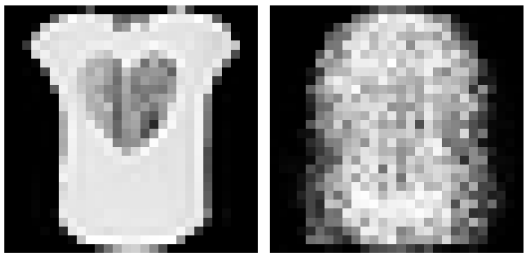

Figura 3: Izquierda= imagen original. Derecha = imagen de la predicción

A continuación se mostrarán los resultados obtenidos modificando el tamaño de la capa intermedia (128, 256 y 512) para observar cómo se comporta la red a medida que el tamaño de la capa aumenta. En cada caso la red mantiene los mismos hiperparámetros que en el primer caso.

En la figura 4 el valor de capa intermedia es de 128. Se presenta la función error del entrenamiento y del testeo. La curva de entrenamiento es levemente mayor a la de testeo. El error desciende abruptamente hasta las primeras 20 épocas aproximadamente y luego el decaimiento es más paulatino. Además, ambos errores disminuyen con respecto al caso anterior.

En la figura 5 se presenta la imagen original y la correspondiente generada por el modelo.

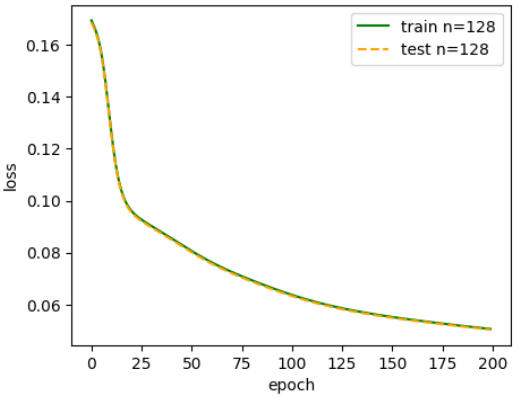

Figura 4: n=128 - Época 199 → train_loss: 0.0506463635712862 test_loss: 0.05042565241456032

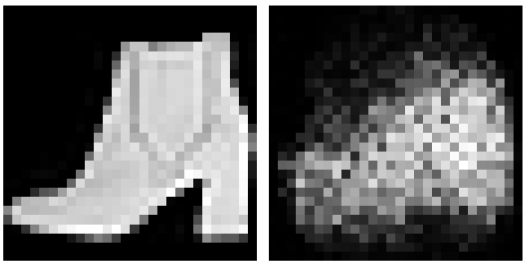

Figura 5: Izquierda= imagen original. Derecha = imagen de la predicción

En la figura 6 el tamaño de la capa intermedia es de 256. Se observa que la curva de entrenamiento es levemente mayor a la de testeo. El error desciende abruptamente hasta las primeras 15 épocas aproximadamente y luego el decaimiento es más paulatino. El error disminuye con respecto a los dos casos anteriores.

En la figura 7 se presenta la imagen original y la imagen de la predicción.

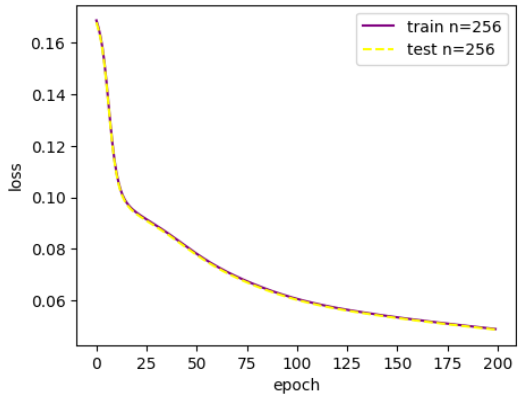

Figura 6: n=256 - Época=199→ train_loss: 0.0489689360683163 test_loss: 0.04876345060765743

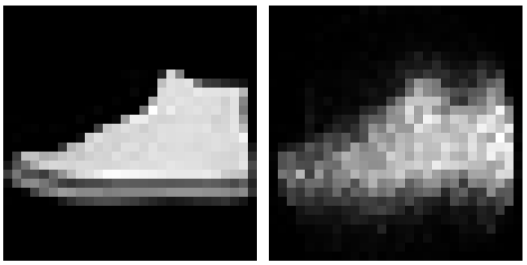

Figura 7: Izquierda= imagen original. Derecha = imagen de la predicción

En la figura 8 el tamaño de la capa intermedia es de 512. Al igual que en los casos anteriores la curva de entrenamiento es levemente mayor a la de testeo. Ambos errores bajan abruptamente hasta la época 10 aproximadamente y luego el decaimiento es más paulatino. El error disminuye con respecto a los tres casos mostrados anteriormente.

En la figura 9 se presenta la imagen original y la imagen de la predicción.

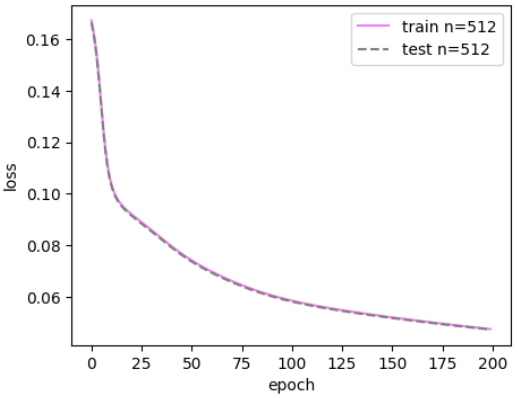

Figura 8: n= 512 - Época=199 →train_loss: 0.047447525523602964 test_loss: 0.04725775420665741

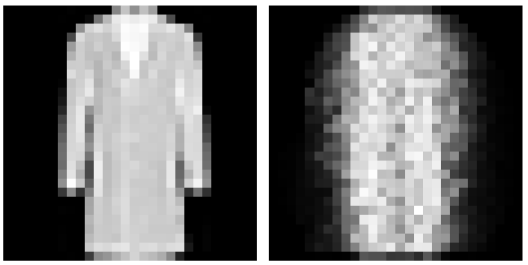

Figura 9: Izquierda= imagen original. Derecha = imagen de la predicción

En la figura 10 se observan las curvas superpuestas para todos los valores de la capa intermedia. Esto demuestra que a medida que aumenta el tamaño de la capa intermedia el error va disminuyendo y aumenta la capacidad de predecir la imagen.

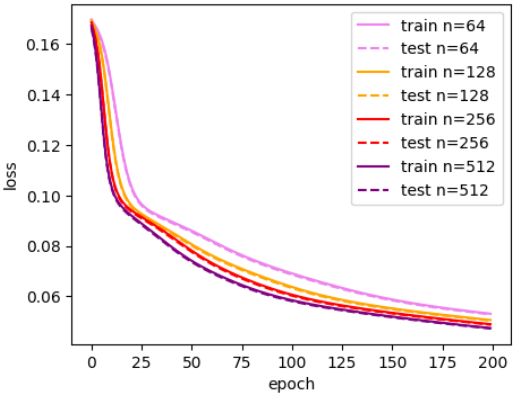

Figura 10: gráfico de curvas 1

Por último, para lograr una mayor reducción del error, se exploró el espacio de los hiperparámetros para todos los valores de la capa intermedia. En este caso, la tasa de entrenamiento es $1e-1$, no hay dropout, el batch es de 100 y la cantidad de épocas aumenta a 300.

La figura 11 demuestra que al ajustar algunos hiperparámetros el error disminuye aún más.

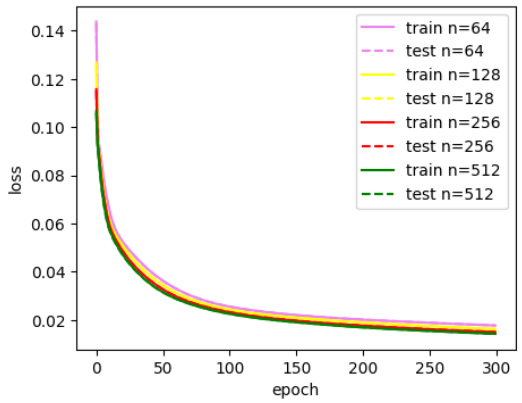

Figura 11: gráfico de curvas 2