Red feed-forward auto-encoder

Manuela Fredriks, Fernanda Micucci, Lautaro Ochotorena
Facultad de Ciencias Exactas, Universidad Nacional de La Plata (Estudiantes de la Licenciatura en Matemática)
(Dated: 7 de diciembre de 2023)

I. INTRODUCCIÓN

El objetivo de este trabajo es implementar una red neuronal del tipo feed-forward auto-encoder aplicada sobre el conjunto de imágenes del dataset Fashion-MNIST. Este contiene 70000 imágenes de 28×28 píxeles (en escala de grises) de prendas y calzados que están clasificadas en 10 categorías.

Esta red incluirá una capa oculta en principio de 64 neuronas, y luego iremos modificando el tamaño de esta capa para así comparar el desempeño de los modelos resultantes.

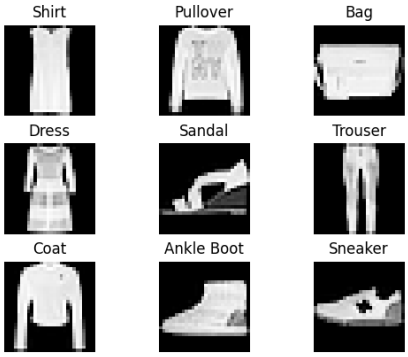
II. CONSTRUYENDO LA RED

Dividiremos el dataset en tres subconjuntos, uno de *entrenamiento*, uno de *validación*, y otro de *testeo* de la siguiente manera: partiendo del total de imágenes, utilizaremos 60000 para entrenar el modelo, y 10000 para realizar el testeo. En la etapa de entrenamiento, tomaremos estas 60000 imágenes y las dividiremos, de forma aleatoria, en subconjuntos de 50000 para entrenar la red, y las 10000 restantes para la validación.

Para el entrenamiento de la red, se utilizarán:

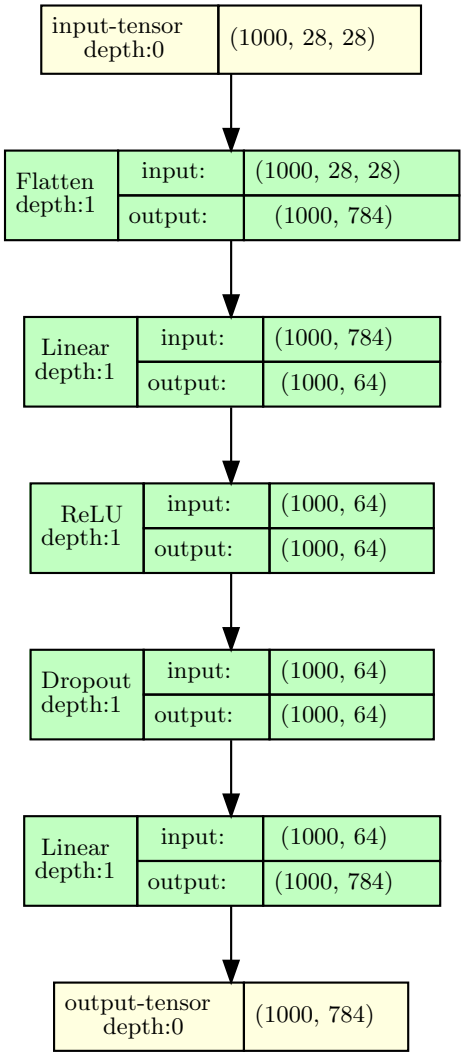
- la función del Error Cuadrático Medio
- el Método del descenso por el Gradiente Estocástico
- dropout con $p = 0{,}1$
- minibatchs de tamaño 1000

La siguiente figura ilustra la clasificación de prendas y calzados del dataset Fashion-MNIST.



Para cada imagen, la red tomará 784 entradas que corresponden a los 28×28 píxeles de la imagen, luego pasará por una capa oculta de neuronas y finalmente una capa de salida también de 784 neuronas que se reorganizaran de nuevo para formar una figura de 28×28.

Implementaremos la red para distintos tamaños de la capa oculta, considerando $n = 64, 128, 256, 512$. Para la comparación del desempeño de los distintos modelos, decidimos fijar una tasa de aprendizaje (learning rate) de 0,5. Como el tamaño del minibatch es 1000, la red ajustará sus pesos cada 1000 imágenes. La siguiente figura representa la arquitectura de la red con $n = 64$.



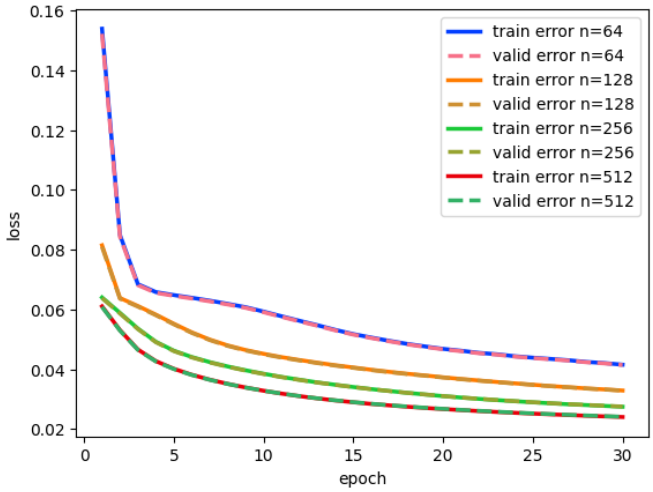


Figura 1: Desempeño de modelos. La curva train error representa el error del conjunto de entrenamiento una vez finalizada la época.

A. Fase de aprendizaje y validación

Observemos en la Figura 1, que en cada modelo la curva de entrenamiento y de validación son casi idénticas y, en consecuencia, no necesitamos establecer un criterio de parada (no hay underfitting ni overfitting) al correr las épocas. De hecho se obtiene el mejor resultado en la última época.

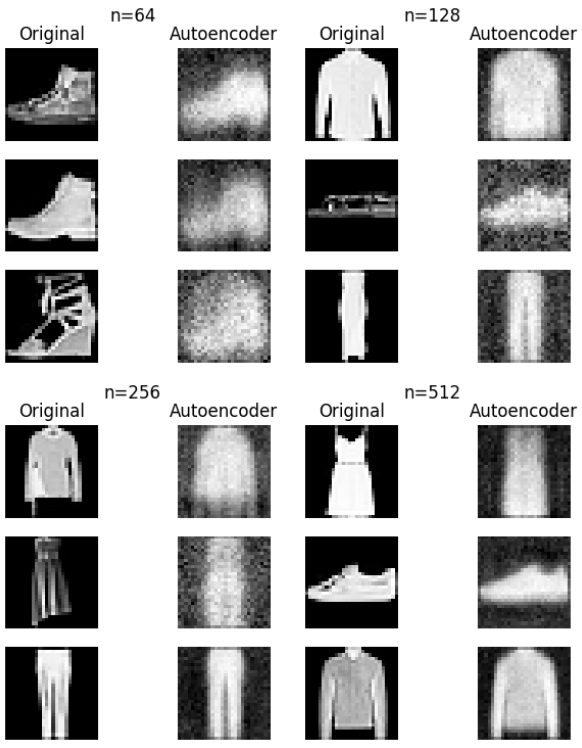


Figura 2: Ejemplos de desempeño según cada modelo

B. Fase de prueba y tiempo de ejecución

Los resultados del error en el conjunto de prueba y el tiempo de ejecución se representan en la siguiente tabla:

n	64	128	256	512
test error	0.0414	0.0328	0.0274	0.0240
tiempo[s]	417.2	412.9	416.8	420.3

Los tiempos de ejecución obtenidos fueron dados con una iteración para cada modelo, sería conveniente repetirlo con varias iteraciones y luego usar una medida de centralidad para comparar. En ese caso esperaríamos ver mayores tiempos de ejecución al aumentar la cantidad de neuronas en la capa oculta.

III. CONCLUSIÓN

Observando la Figura 2, podemos visualizar cómo reconstruye cada modelo las diversas imágenes. Para el caso de $n = 64$ no se obtiene un buen resultado ya que el autoencoder las genera con poca claridad y de manera difusa.

A partir del caso $n = 128$ se nota una mejora a medida que el n aumenta. Si bien la mejor performance de error se obtiene con el modelo de $n = 512$, notemos que la complejidad computacional aumenta y, por lo tanto, en teoría el tiempo de ejecución. Además, no hay una diferencia de error ni de testeo muy significativa entre el modelo $n = 256$ y $n = 512$ pero se ve una mejora en la reconstrucción de las imágenes en el último caso.

¿Cómo podríamos mejorar el desempeño de la red? Considerando que para estos cuatro modelos fijamos el learning rate y el optimizador, sería útil probar con otros hiperparámetros, por ejemplo con el Adam, buscar el learning rate óptimo para cada n fijo y jugar con el dropout con valores como 0,1, 0,2 y 0,3.

IV. E-MAIL DE CONTACTO

manufredriks@gmail.com
micuccifernanda97@gmail.com
lau_sansimon@hotmail.com