Auto-encoders y Clasificadores con redes convolucionales

Lautaro Ochotorena*
Facultad de Ciencias Exactas, Universidad Nacional de La Plata (Estudiante de la Licenciatura en Matemática)
(Dated: 19 de diciembre de 2023)

I. INTRODUCCIÓN

Se trabajará con el dataset Fashion-MNIST el cual consiste de 70000 imágenes de 28×28 píxeles (en escala de grises) de ropa y calzado que están clasificadas en 10 categorías.

Se implementarán varias redes neuronales, primero se crearán los autoencoders para luego utilizar los encoders en redes de clasificación para predecir las etiquetas de las prendas.

Dividiremos el dataset en dos subconjuntos, uno de *entrenamiento* y uno de *validación* de la siguiente manera: partiendo del total de imágenes, utilizaremos 60000 para entrenar el modelo y 10000 para realizar la validación.

El objetivo principal es adentrarse en el funcionamiento de las redes convolucionales e ir probando diferentes técnicas e hiperparámetros para mejorar el desempeño de los modelos.

II. AUTOENCODERS

Se crearán 4 modelos utilizando como loss function el Error Cuadrático Medio. Todas consistirán de la misma arquitectura

Para la parte del **encoder** se seguirá el siguiente orden:.

- Una capa convolucional 2D de entrada de dimensiones (1, 28, 28) y salida (16, 26, 26).
- Función de activación ReLU.
- DropOut con probabilidad p.
- Una capa MaxPool que mapea dimensiones (16, 26, 26) a dimensiones (16, 13, 13).
- Una capa convolucional 2D de entrada de dimensiones (16, 13, 13) y salida (32, 11, 11).
- ReLU.
- DropOut con probabilidad p.
- Una capa MaxPool que mapea dimensiones (32, 11, 11) a dimensiones (32, 5, 5).
- Una capa Flatten que mapea dimensiones (32, 5, 5) a un vector de $32 * 5 * 5$.
- Una capa linear que mapea un vector de $32 * 5 * 5$ a uno de n elementos.
- ReLU.
- DropOut con probabilidad p.

Ahora, para el **decoder** se seguirá el siguiente orden:.

- Una capa linear que mapea un vector de n elementos a uno de $35 * 5 * 5$.
- ReLU.
- DropOut con probabilidad p.
- Una capa Unflatten que mapea un vector de $32*5*5$ a dimensiones (32, 5, 5).
- Una capa convolucional 2D traspuesta que mapea dimensiones (32, 5, 5) a dimensiones (16, 13, 13).
- ReLU.
- DropOut con probabilidad p.
- Una capa convolucional 2D traspuesta que mapea dimensiones (16, 13, 13) a dimensiones (1, 28, 28).
- Función de activación Sigmoid

Para más detalle de la construcción del mismo ver [1].

Los distintos modelos poseen hiperparámetros diferentes, éstos son:

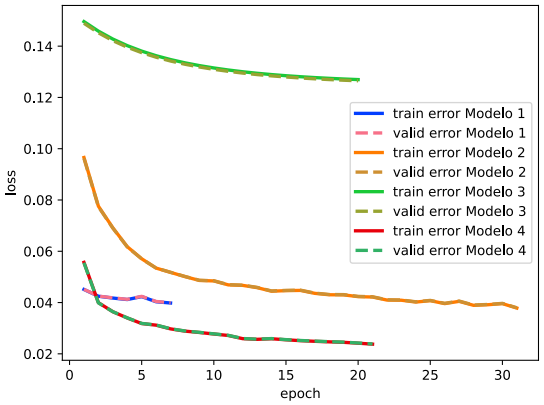
	Modelo 1	**Modelo 2**	**Modelo 3**	**Modelo 4**
n	64	256	64	128
p	0.2	0.3	0.3	0.3
batch size	100	1000	1000	1000
optimizador	Adam	Adam	SGD	Adam
learing rate	0.001	0.001	0.01	0.001

Cuadro I: Los modelos con optimizador Adam poseen un parámetro *eps* con valor 10^{-8}

Hay una sutileza en el Modelo 4 con respecto al resto, al mismo se le eliminó el parámetro *output_padding* de las convolucionales traspuestas y se le agregó (para mantener dimensionalidad) *kernel_size*.

A la hora de entrenar se implementó un Early Stopper que parará la ejecución de las épocas si el error de validación no baja (o baja muy poco) en las últimas 5 épocas. Se restauran los pesos óptimos antes de que se produzca ese estancamiento.

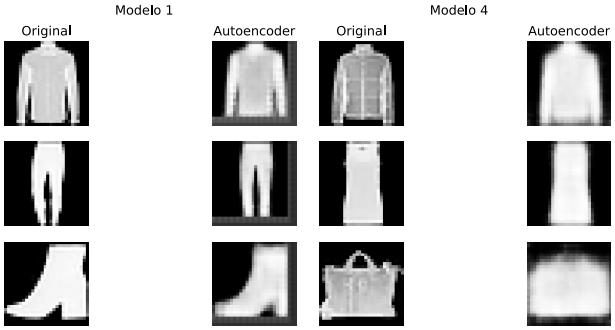
Los errores obtenidos son representados por la siguiente figura



En todos los casos la época óptima es la última.

El mejor resultado se obtuvo con el Modelo 4 que con diferencia logra el mejor resultado. Por otro lado, el peor es el Modelo 3 probablemente por el optimizador del Gradiente Estocástico.

Si ponemos en comparativa los dos mejores modelos autoencoders tenemos



De hecho, la sutileza anteriormente mencionada del Modelo 4 surgió en base a que los modelos sin esa sutileza presentan franjas mal clasificadas en la parte inferior y derecha de las imágenes.

III. CLASIFICADORES

Se crearán 4 modelos utilizando como loss function la Cross Entropy. También se implementó un Early Stopper para no sobreajustar a los modelos.

La figura 1. es la arquitectura del modelo de clasificación con $n = 64$, el resto de modelos tienen una arquitectura análoga.

Las variaciones vienen dadas por hiperparámetros diferentes y quedan expresados en el cuadro II.

Se entrenaron 40 épocas por cada modelo. En algunos casos, para evitar el overfitting, se decidió una época óptima y se restauran los pesos sinápticos del modelo a los de dicha época.

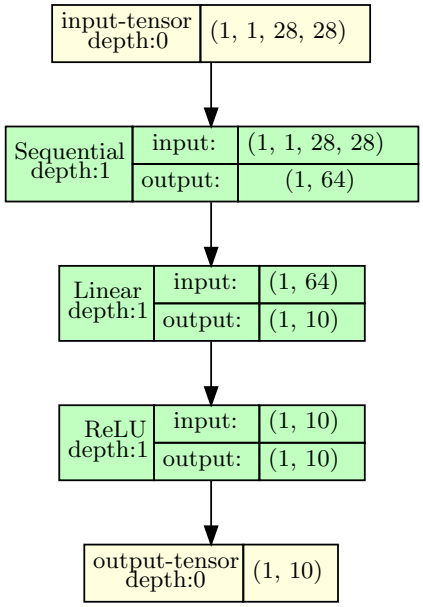


Figura 1: La capa Sequential viene a representar el encoder que se utilizará de los autoencoders ya entrenados.

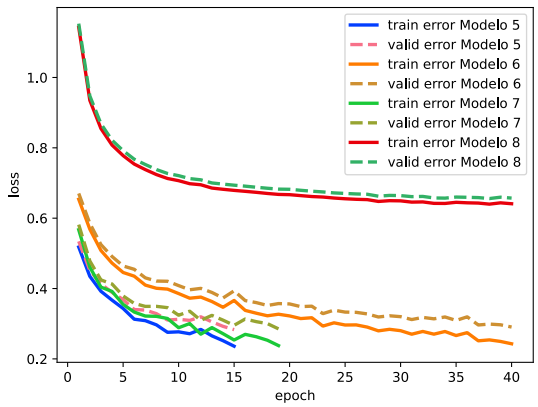
	Modelo 5	**Modelo 6**	**Modelo 7**	**Modelo 8**
n	64	256	128	128
p	0.2	0.3	0.3	0.3
batch size	100	1000	100	1000
optimizador	Adam	Adam	Adam	Adam
learing rate	0.001	0.001	0.001	0.001
encoder	Modelo 1	Modelo 2	Modelo 4	Modelo 4
final act func	ReLU	ReLU	ReLU	–

Cuadro II: Los modelos con optimizador Adam poseen un parámetro *eps* con valor 10^{-8}. La CrossEntropyLoss cuenta con la Softmax implementada y por ello no se indica función de activación en el modelo 8. Además, dicho modelo sólo entrenará los pesos del clasificador, deja fijos los del encoder.

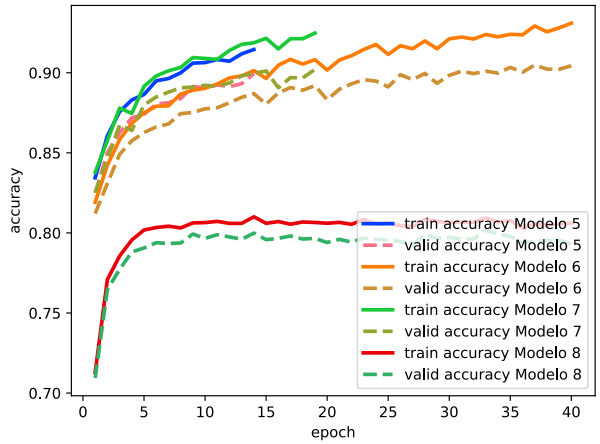
Los resultados obtenidos indican que:

- El modelo 5 entrenó hasta la época 38 ya que se activó el Early Stopper y se decidió como la época óptima la 15 por el sobreajuste visto.
- El modelo 6 entrenó las 40 épocas, se decidió que la época óptima sea la última.
- El modelo 7 entrenó hasta la época 38 ya que se activó el Early Stopper y se decidió como la época óptima la 19 por el sobreajuste visto.
- El modelo 8 entrenó las 40 épocas, no hubo overfitting, la época óptima es la última. De hecho hay underfitting.

Los errores de los modelos fueron los siguientes



Mientras que la precisión



Resumiendo el desempeño de los modelos en una tabla quedan:

	Modelo 5	**Modelo 6**	**Modelo 7**	**Modelo 8**
train error	0.236	0.242	0.237	0.641
valid error	0.283	0.290	0.285	0.657
train accuracy	91.8 %	93.1 %	92.5 %	80.6 %
valid accuracy	90.0 %	90.4 %	90.2 %	79.3 %

Cuadro III: Resultados en la época óptima

El modelo 8 logra un desempeño muy malo en comparación al resto, probablemete a causa de sólo entrenar los pesos del clasificador y no los del encoder también. El resto de modelos tienen un resultado bueno y similar entre ellos.

Tomando el modelo 6 como referente se verá cómo clasifica algunos ejemplos del conjunto de validación (ver figura 2).

Y para ver en forma general el desempeño en todo el conjunto de validación se muestra la matriz de confusión (ver figura 3).

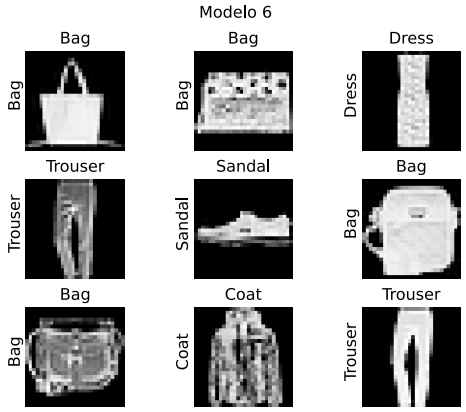


Figura 2: La etiqueta de la izquierda de cada imagen es la verdadera y la de arriba la predicha por el modelo

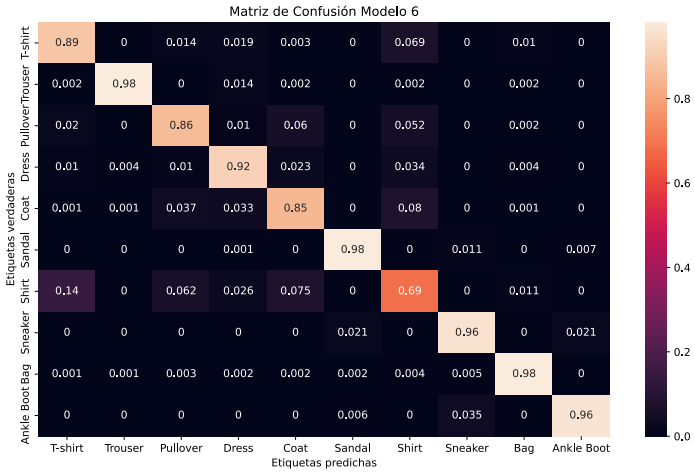


Figura 3: Matriz de confusión normalizada

Lo que más le complica clasificar son las camisas que a veces las confunde con remeras. Por otro lado, el clasificador se destaca con los pantalones, bolsas, sandalias y botas.

IV. CONCLUSIÓN

Este apartado son conclusiones que he sacado a lo largo de experimentar y jugar con los modelos:

Los hiperparámetros óptimos dependen de cada modelo pero por lo general el optimizador Adam es más eficiente en comparación con el gradiente estocástico.

Se recomienda mantener los pesos del autoencoder sólo como inicialización, no dejarlos fijos a la hora de entrenar la red de clasificación.

El Early Stopper es muy necesario para evitar desperdiciar tiempo de cálculo en modelos que ya no logran bajar el error de validación.

Si bien no adjunto sus matrices de confusión en es-

te trabajo, los modelos 5 y 7 logran clasificar mejor las camisas que con el modelo 6.

Para las redes de clasificaicón, la utilización de los pesos de los encoders no significó una diferencia notable al cambiar de autoencoders.

* Mail: lau_sansimon@hotmail.com

[1] Lautaro Ochotorena. Collab de los autoencoders y clasificadores. https://colab.research.google.com/drive/1Y1MiSmAzOqyNJeSE6MwHcs0qlPP3tBLk?usp=sharing, 2023.