Trabajo Integrados - Autoencoder y Clasificador con PyTorch

Kevin Gaston Mansilla*

(Dated: November 25, 2024)

I. INTRODUCCIÓN

Tomando como punto de partida la base de datos de Fashion MNIST [1], el objetivo de este trabajo es implementar un autoencoder convolucional que es basicamente un sistema de codificación y decodificación de imágenes.

Luego de entrenar el autoencoder, procederemos a variar los hiperparámetros de la red para ver como afectan al error de reconstrucción. A su vez, definiremos y entrenaremos un clasificador convolucional reutilizando el encoder del autoencoder previamente entrenado.

Por último, experimentaremos solo reentrenando los parametros de la capa clasificadora, dejando los parámetros de la capa codificadora tal como vienen entrenados del autoencoder convolucional y compararemos los resultados obtenidos.

II. AUTOENCODER CONVOLUCIONAL

En un autoencoder convolucional, se aplican convoluciones o filtros entre sucesivas capas de la red para reducir la dimensionalidad del problema. Estos filtros van a captar la información importante con la que realizaremos el encoder de la red para luego proceder a realizar el procedo inverso de la codificación, es decir, la decodificación.

Como se muestra en la figura 1, la principal funcion es entrenarlo para que reconstruya una imagen a partir de los datos normales con un error de reconstrucción menor posible.

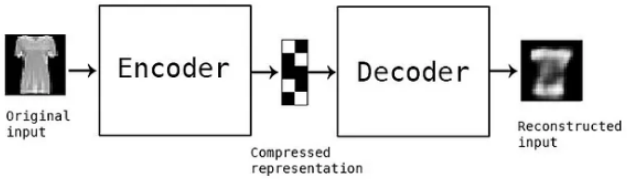

FIG. 1: Red feed forward

Entrenaremos el autoencoder base con los siguientes hiperparámetros:

- $n = 64$.
- $lr = 0.001$.
- $p = 0.2$ (probabilidad de dropout).
- $epochs = 15$.
- $batch\_size = 100$.
- $loss = MSELoss$.
- $optimizer = Adam$.

La arquitectura del encoder es la siguiente:

- Una capa convolucional 2D compuesta por:
  - un capa $Conv2d$ de entrada de dimensiones $(1, 28, 28)$ y de salida $(16, 26, 26)$.
  - una capa de activación $ReLU$.
  - una capa $Dropout$.
  - una capa $MaxPool2d$, la cual mapea dimensiones $(16, 26, 26)$ a $(16, 13, 13)$.
- Otra capa convolucional 2D compuesta por:
  - un capa $Conv2d$ de entrada de dimensiones $(16, 13, 13)$ y de salida $(32, 11, 11)$
  - una capa de activación $ReLU$.
  - una capa $Dropout$.
  - una capa $MaxPool2d$, la cual mapea dimensiones $(32, 11, 11)$ a $(32, 5, 5)$.
- Una capa lineal compuesta por:
  - una capa $Flatten$ que mapea dimensiones $(32, 5, 5)$ a (800).
  - una capa $Linear$ de entrada (800) y salida n.
  - una capa de activación $ReLU$.
  - una capa $Dropout$.

El decoder con las siguientes capas:

- Una capa lineal compuesta por:
  - una capa $Linear$ de entrada n y salida (800).
  - una capa de activación $ReLU$.
  - una capa $Dropout$.
  - una capa $Unflatten$ que mapea dimensiones (800) a $(32, 5, 5)$.
- Una capa convolucional 2D compuesta por:
  - un capa $ConvTranspose2d$ que mapea dimensiones $(32, 5, 5)$ a $(16, 13, 13)$.
  - una capa de activación $ReLU$.
  - una capa $Dropout$.
- Otra capa convolucional 2D compuesta por:
  - un capa $ConvTranspose2d$ que mapea dimensione $(16, 13, 13)$ a $(1, 28, 28)$.
  - una capa de activación $Sigmoid$.
  - una capa $Dropout$.

Luego variaremos los hiperparámetros de la red para ver como afecta su desempeño. Primero cambiaremos el optimizador por SGD, luego la probabilidad de dropout a 0.5 y 0.1. Siempre manteniendo el resto de los hiperparámetros constantes para poder comparar los resultados.

III. CLASIFICADOR CONVOLUCIONAL

Una vez entrenado el autoencoder, procederemos a reutilizar el encoder para entrenar un clasificador convolucional. La idea es que el encoder ya tiene una buena representación de las imágenes, por lo que no es necesario volver a entrenarlo.

El clasificador tendrá la siguiente arquitectura:

- Una capa lineal compuesta por:
  - una capa *Linear* de entrada n y salida 10.
  - una capa de activación *ReLU*.
  - una capa *Dropout*.

Lo entrenaremos usando como base el modelo original y posteriormente solo reentrenaremos los parámetros de la capa clasificadora dejando los parámetros de la capa codificadora tal como vienen entrenados del autoencoder convolucional.

IV. RESULTADOS

A. Modelo sin entrenar

En la figura **??** se muestra la imagen a predecir y las imagenes reconstruidas por el modelo sin entrenar. Se puede observar que el modelo no es capaz de reconstruir la imagen original.

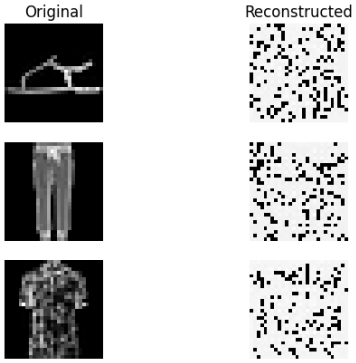

FIG. 2: Imagenea a predir vs imagenes reconstruidas por el modelo sin entrenar

B. Modelo base entrenado

Los resultados obtenidos para el modelo base entrenado se presentan en la figura 3. Se ve que las curvas muestran una disminución en la función de pérdida a medida que avanzan las épocas, lo cual indica que le modelo mejora su capacidad de ajuste a los datos. Que las curvas sean casi identicas sugiere que no hay overfitting.

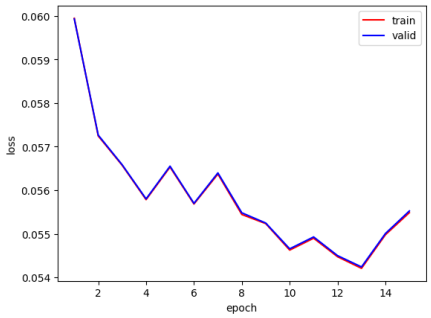

FIG. 3: Función de pérdida del modelo base entrenado

También se muestra en la figura 4 la imagen a predecir y las imagenes reconstruidas por el modelo base entrenado. Se puede observar que el modelo es capaz de reconstruir la imagen original con un error de reconstrucción menor.

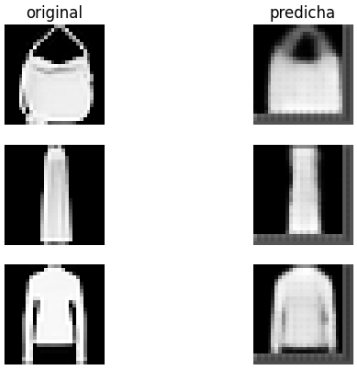

FIG. 4: Imagenea a predir vs imagenes reconstruidas por el modelo base entrenado

C. Modelo con optimizador SGD

Los resultados obtenidos para el modelo con optimizador SGD se presentan en la figura 5 Se ve que las curvas muestran una gran disminución al inicio (epocas 0-10), con una posterior estabilización y un ligero aumento hacia el final despues de la epoca 20. Esto podria ser un signo de sobre ajuste, donde el modelo comienza a ajustarse demaciado a los datos y pierde su capacidad de generalización.

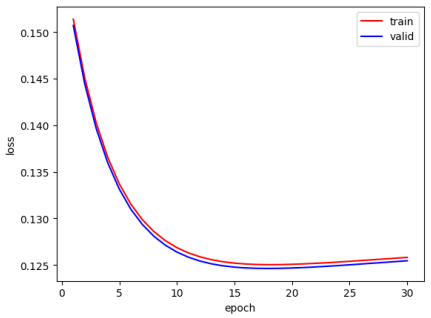

FIG. 5: Función de pérdida del modelo con optimizador SGD

En el caso de la imagen a predecir y las imagenes reconstruidas por el modelo con optimizador SGD se muestra en la figura 6. Se puede observar que el modelo no es capaz de reconstruir la imagen original.

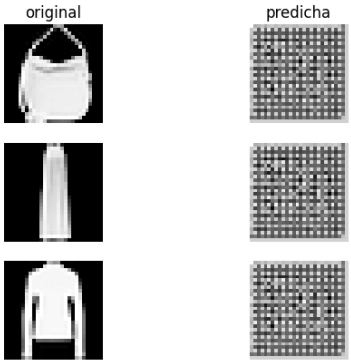

FIG. 6: Imagenea a predir vs imagenes reconstruidas por el modelo SDG

D. Modelo con probabilidad de dropout 0.5, optimizador Adam

En la figura 7 se muestra la función de pérdida del modelo con probabilidad de dropout 0.5. Se ve que las curvas muestran una oscilación en la función de pérdida a medida que avanzan las épocas, lo cual indica que le modelo no mejora su capacidad de ajuste a los datos ya que la tasa de aprendizaje es muy alta produciendo ruido en el ajuste.

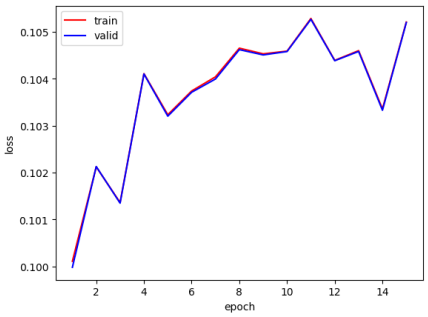

FIG. 7: Función de pérdida del modelo con probabilidad de dropout 0.5

En la figura 8 las imágenes en la columna "predicha" parecen tener menos detalles en comparación con las originales, indicando que el modelo tiene dificultades para capturar todas las características de las imágenes originales. Esto puede deberse a un modelo subentrenado, la regularización por dropout muy alta, o una capacidad limitada del modelo. El modelo es capaz de captar los patrones generales de las imágenes, pero pierde información de alto nivel, como los bordes o los detalles más finos.

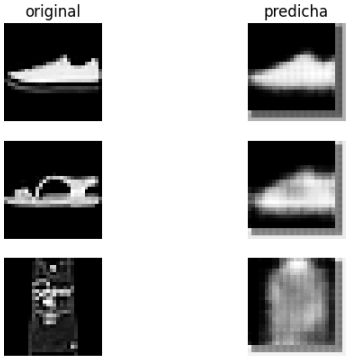

FIG. 8: Imagenea a predir vs imagenes reconstruidas por el modelo con probabilidad de dropout 0.5

V. CLASIFICADOR CONVOLUCIONAL

En esta sección se presentan los resultados obtenidos para el clasificador convolucional que reutiliza el encoder del autoencoder previamente entrenado y en este caso se reentrenan todos los parámetros de la red por completo.

En la figura 9 se muestra la función de pérdida del modelo tomando como partida el modelo base, se observa que las curvas muestran una disminución en la función de pérdida tanto en el conjunto de entrenamiento como en el de validación a medida que avanzan las épocas, lo cual indica que le modelo mejora su capacidad de ajuste a los datos, aun que hay fluctiaciones no se presentan signos de overfitting.

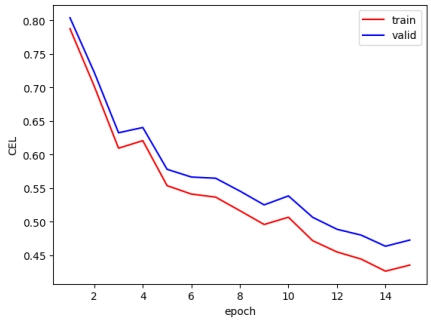

FIG. 9: Función de pérdida del clasificador

La precisión aumenta en ambos conjuntos de datos segun se muestra en la figura 10, lo cual indica que el modelo es capaz de generalizar bien a partir de los datos de entrenamiento.

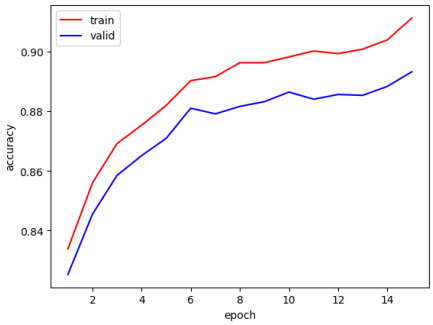

FIG. 10: Presición del clasificador

Y la matriz de confución se muestra en la figura 11, donde se puede observar un buen desempeño del clasificador. En la diagonal principal se encuentran los valores más altos, lo cual indica que la mayoría de las predicciones son correctas. Tiene un buen desempeño en clases como Sneaker, Bag, Trouser y Ankle Boot ya que sus valores de la diagonal principal esta en al valor mas alto de la escala cromatica. Pero tiene dificultades en distinguier entre las clases Shirt y Coat, ejemplo tomando la fila Shirt se ve que solo acierta 400 veces y 235 veces se equivoca prediciendo T-shirt y 245 como Pullover, algo similar ocurre con Coat, esto se debe a la similitud entre las categorías, ya que todas son prendas de vestir superiores.

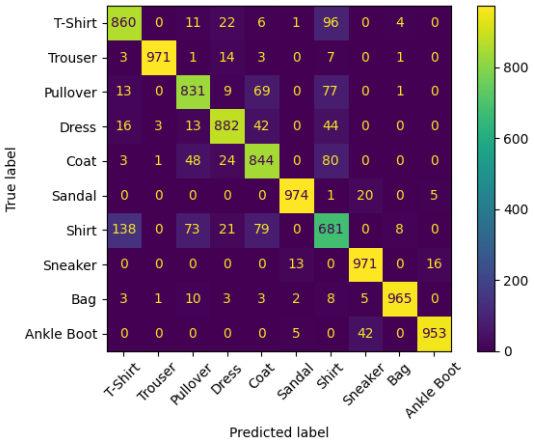

FIG. 11: Matriz de confución del clasificador

A. Clasificador solo entrenando la capa clasificadora

Ahora analizamos los resultados obtenidos para el clasificador convolucional reutilizando el encoder del autoencoder previamente entrenado y en este caso solo reentrenando los parámetros de la capa clasificadora. Es decir, que tomando como punto de partida el modelo base, solo se reentrenan los parámetros de la capa clasificadora.

En la figura 12 se muestran las funciones de perdidas de ambos conjuntos de datos y se observa que las curvas muestran una disminución bastante pronunciada en ambos casos. El comportamiento es similar al modelo anterior.

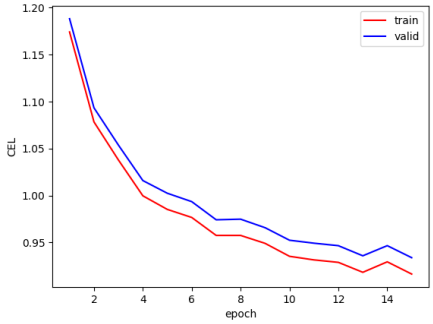

FIG. 12: Función de pérdida del clasificador - solo reentrenando la capa clasificadora

En la figura 13 se muestra la precisión del modelo en ambos conjuntos de datos. Se observa un comportamiento diferente al modelo anterior. Inicialmente ambas curvas aumentan, pero luego se estabilizan y comienzan a oscilar, esto puede ser un signo de sobre ajuste en el modelo.

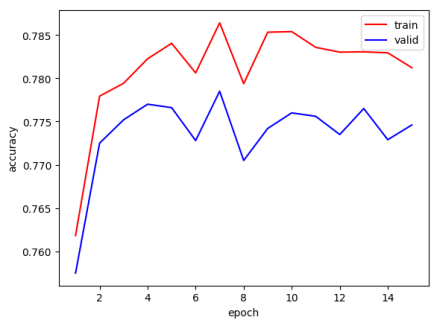

FIG. 13: Presición del clasificador - solo reentrenando la capa clasificadora

A fin de comparar el desempeño de ambos clasificadores, se muestra en la figura 14 la matriz de confución donde efectivamente se comprueba que el desempeño del clasificador solo reentrenando la capa clasificadora es peor que el modelo anterior. Ya que los valores de la diagonal principal son menores en todos las categorias. Y además empeora mucho el desempeño en las categorias Coat y Shirt.

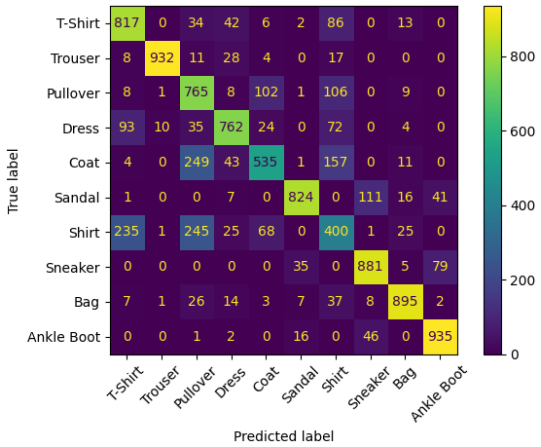

FIG. 14: matrix de confución - solo reentrenando la capa clasificadora

Entonces podemos concluir que el modelo base donde se reentrenan todos los parámetros de la red tiene un mejor desempeño que el modelo donde solo se reentrenan los parámetros de la capa clasificadora.

VI. CONCLUSIONES

A lo largo de este trabajo se fueron entrenando diferentes modelos de redes neuronales convolucionales. Se comenzó con un autoencoder convolucional, luego se reutilizó el encoder para entrenar un clasificador convolucional y finalmente se reentrenó solo la capa clasificadora.

A medida que se fue aumentado la complejidad de los modelos los resultados fueron mejorando, sin embargo se observó que las distancias entre las curvas de entrenamiento y validación se fueron agrandando, lo cual indica que los modelos estaban sobreajustando los datos.

El modelos bases generaron buenos resultados tanto en el caso del autoencoder como en el clasificador. Y se pudo notar que al variar hiperparametros como el optimizador, dropout hay que ser bastante cuidadoso ya que los resultados pueden variar mucho y no siempre para mejor. Asi como también en el caso de reentrenar solo la capa clasificadora, se observó que el modelo no mejoró su desempeño sino que empeoró.

Algo muy importante a resaltar es que no es necesario entrenar los modelos durante muchas épocas, ya que con 15 épocas se obtuvieron resultados satisfactorios.

* kevin.mansilla@mi.unc.edu.ar

[1] Https://github.com/zalandoresearch/fashion-mnist.