Trabajo Integrados - Autoencoder y Clasificador con PyTorch

Kevin Gaston Mansilla*

(Dated: November 27, 2024)

En este trabajo se implementara un autoencoder convolucion que es una herramienta muy útil para la reducción de dimensionalidad y la extracción de características de las imágenes, ya que es capaz de capturar patrones generales de las imágenes y comprimir la información de manera eficiente. Luego se reutilizará el encoder del autoencoder previamente entrenado para entrenar un clasificador convolucional desde cero y también solo entrenando los parámetros de la capa clasificadora. Se realizaran experimentos variando los hiperparámetros de la red para ver como afectan al error de reconstrucción y al desempeño del clasificador para finalmente comparar los resultados obtenidos.

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.

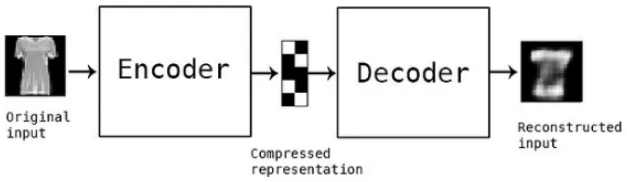

FIG. 1: Autoencoder convolucional

Entrenaremos el autoencoder base con los siguientes hiperparámetros (modelo base):

- $n = 64$ (número de neuronas en la capa lineal).
- $lr = 0.001$.
- $p = 0.2$ (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*.

No le agregamos capa de *Dropout* en la capa de salida del decoder ya que este genera la salida reconstruida, que idealmente deberia ser lo más similar posible a los datos de entrada originales, si le agregaramos esta capa introduciria ruido que afectaria directamente la calidad de reconstrucción.

Luego variaremos los hiperparámetros de la red para ver como afecta su desempeño. Primero variamos la probabildidad de dropout a 0.1, luego cambiaremos el número de neuronas en la capa lineal a 256 con el mismo dropout y finalmente cambiaremos el optimizador por SGD.

III. CLASIFICADOR CONVOLUCIONAL

Una vez entrenado el autoencoder, procederemos a reutilizar el encoder para entrenar un clasificador convolucional. Que tendra la siguiente arquitectura:

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

El clasificador sera entrenado de dos formas. Primero reentrenaremos todos los parámetros de la red y luego solo reentrenaremos los parámetros de la capa clasificadora usando el encoder del autoencoder previamente entrenado.

IV. RESULTADOS

A. Modelo sin entrenar

En la figura 2 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.

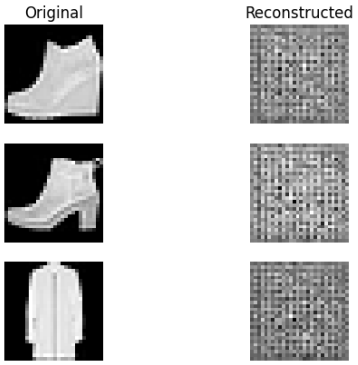

FIG. 2: Imagenea a predir vs imagenes reconstruidas por el modelo sin entrenar

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,pero a partir de la época 5 las curvas comienzan a aumentar drasticamente, lo cual podría ser un signo de que el modelo no va a tener un buen desempeño.

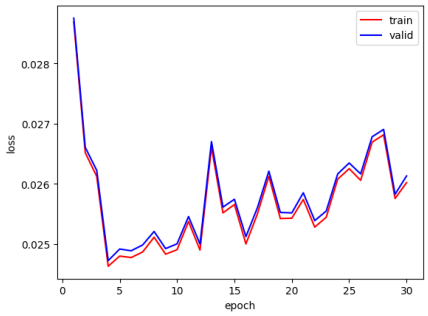

FIG. 3: Función de pérdida del modelo base entrenado

En la figura 4 se puede observar que el modelo es capaz de reconstruir la imagen original pero con una calidad no muy buena, lo cual confirma lo observado en la función de pérdida.

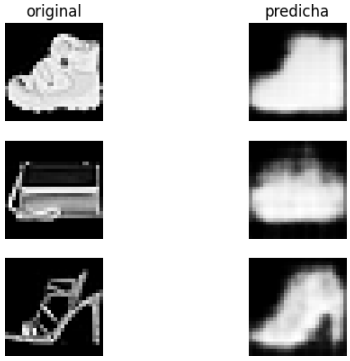

FIG. 4: Imagenea a predir vs imagenes reconstruidas por el modelo base entrenado

C. Modelo con probabilidad de dropout 0.1, $n = 64$

En la figura 5 se muestra la función de pérdida del modelo con probabilidad de dropout 0.1. Podemos identificar un comportamiento similar al modelo base, por lo que el cambio de este hiperparámetro no afecta significativamente el desempeño del modelo.

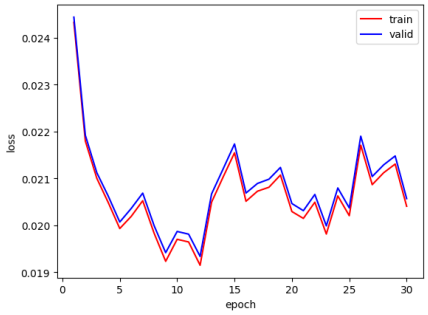

FIG. 5: Función de pérdida del modelo con probabilidad de dropout 0.1

En la figura 6 vemos que 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.

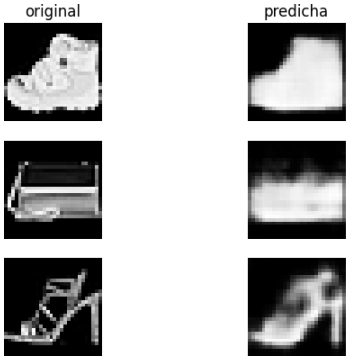

FIG. 6: Imagenea a predir vs imagenes reconstruidas por el modelo con probabilidad de dropout 0.1

D. Modelo con probabilidad de dropout 0.1, $n = 256$

Ahora con un dropout de 0.1 y $n = 256$ como vemos en la figura 5 la funciones de perdida de ambos conjuntos de datos disminuyen constantemente, lo que indica el autoencoder está logrando comprimir y reconstruir los datos de entrada de manera eficiente. La proximidad entre las curvas muestra que el autoencoder esta funcionando correctamente, es decir, que las representaciones latentes que aprende son aplicables tanto al conjunto de entrenamiento como al de validación.

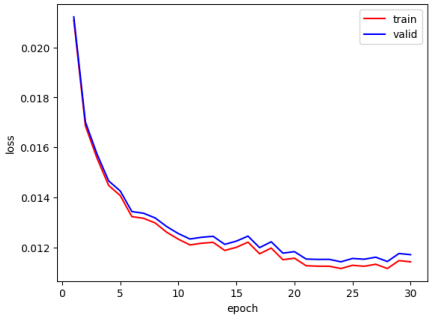

FIG. 7: Función de pérdida del modelo con probabilidad de dropout 0.1 y $n = 256$

En la figura 6 aunque no son perfectas, el modelo es capaz de capturar más detalles de las imágenes originales.

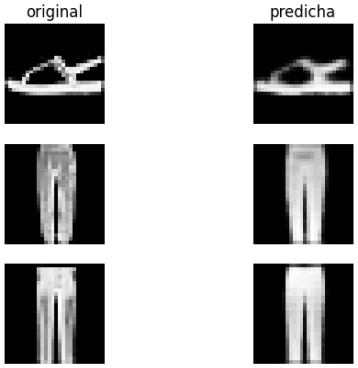

FIG. 8: Imagenea a predir vs imagenes reconstruidas por el modelo con probabilidad de dropout 0.1 y $n = 256$

E. Modelo con optimizador SGD

Los resultados obtenidos para el modelo con optimizador SGD se presentan en la figura 9 vemos que la función de pérdida del modelo disminuyen en ambos conjuntos de datos, lo cual podría indicar un buen comportamiento del modelo. Sin embargo, si observamos el eje de loss vemos que los valores son mucho mayores que en los modelos anteriores, lo cual indica que el modelo no esta aprendiendo correctamente.

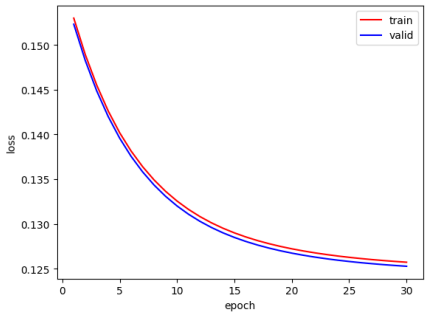

FIG. 9: Función de pérdida del modelo con optimizador SGD

En la figura 10. Se puede observar que el modelo no es capaz de reconstruir la imagen original.

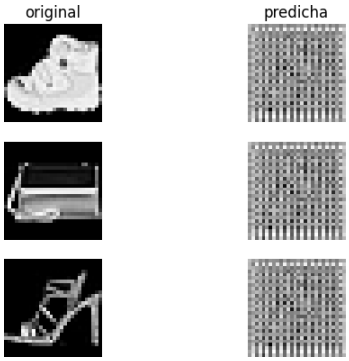

FIG. 10: Imagenea a predir vs imagenes reconstruidas por el modelo SDG

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 11 se muestran las funciones de perdidas tomando como partida el modelo base. 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 con un error relativamente bajo, lo cual indica que le modelo mejora su capacidad de ajuste a los datos.

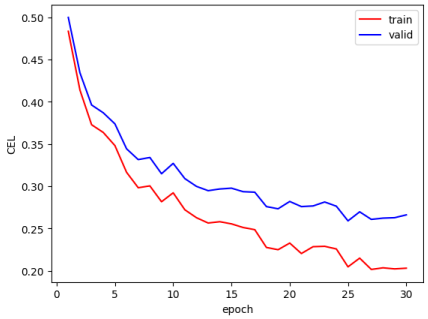

FIG. 11: Función de pérdida del clasificador

La precisión aumenta en ambos conjuntos de datos segun se muestra en la figura 12, lo cual indica que el modelo es capaz

de clasificar correctamente las imágenes de entrada.

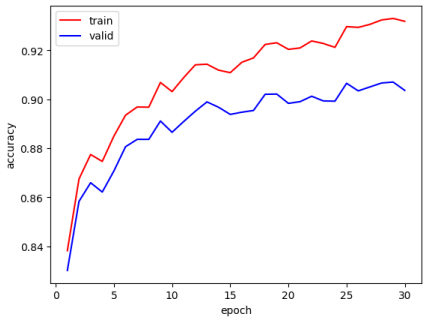

FIG. 12: Presición del clasificador

Y la matriz de confución se muestra en la figura 13, 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.

Se tiene un buen desempeño en clases como Trouser, Sandal, Sneaker, Bag y Ankle Boot. Pero tiene dificultades en distingir entre las clases Shirt y Coat, ejemplo tomando la fila Shirt se ve que solo acierta 695 veces y 138 veces se equivoca prediciendo T-shirt y 83 como Pullover, algo similar ocurre con Coat, esto se debe a la similitud entre las categorías, ya que todas son prendas de vestir superiores.

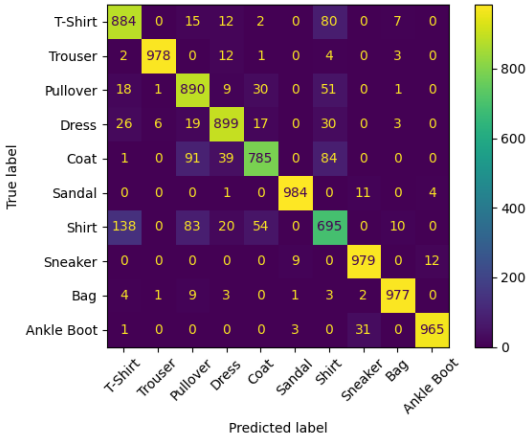

FIG. 13: Matriz de confución del clasificador

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 14 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, pero con un error mucho mayor en la función de pérdida.

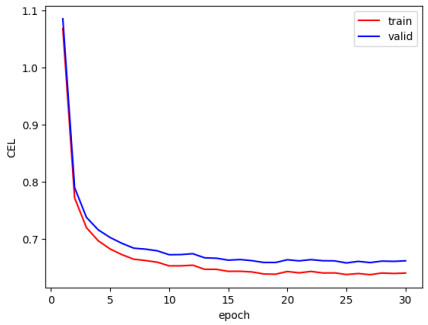

FIG. 14: Función de pérdida del clasificador - solo reentrenando la capa clasificadora

En la figura 15 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.

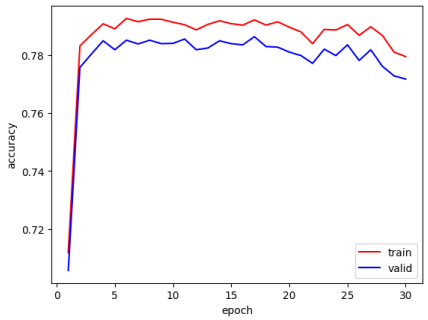

FIG. 15: Presición del clasificador - solo reentrenando la capa clasificadora

A fin de comparar el desempeño de ambos clasificadores, se muestra en la figura 16 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.

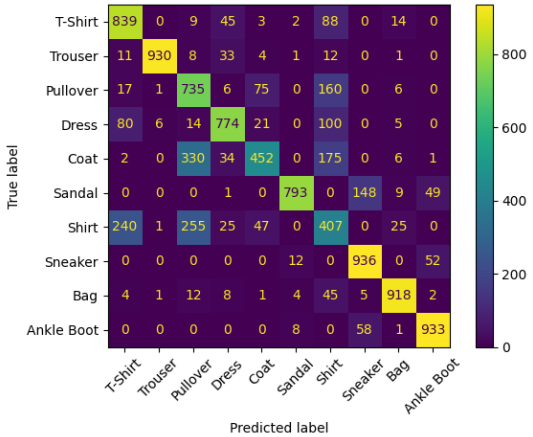

FIG. 16: matrix de confución - solo reentrenando la capa clasificadora

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 compararon los resultados obtenidos al reentrenar 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 signos de un desempeño no tan bueno.

El modelo base su desempeño al reconstruir las imágenes no fue muy bueno, aun que tenia un error relativamente bajo. Al variar la probabilidad de dropout a 0.1 no se observaron cambios significativos en el desempeño del modelo. Pero al aumentar el número de neuronas en la capa lineal a 256 se observó una mejora en la calidad de las imágenes reconstruidas.

El modelo con optimizador SGD no tuvo un buen desempeño, ya que el error de reconstrucción fue muy alto. Por otro lado, el clasificador convolucional reutilizando el encoder del autoencoder previamente entrenado tuvo un buen desempeño, ya que al observar la matriz de confución se ve que la mayoría de las predicciones son correctas. Por último, el clasificador solo reentrenando la capa clasificadora tuvo un desempeño peor que el modelo anterior.

Algo muy importante a resaltar es que es muy dificil encontrar una combinación de hiperparámetros que permita obtener un modelo con un desempeño óptimo, ya que requiere de mucho tiempo y experiencia encontrar la combinación correcta.

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

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