Red neuronal profunda y convolucional para clasificar imágenes

María Florencia Molina[1]
[1]FAMAF-UNC – Córdoba, Argentina
e-mail: florencia.molina.756@mi.unc.edu.ar

***Abstract* – El objetivo de este trabajo es crear un calsificador convolucional para la base de datos Fashion-MNIST, variando distintos parámetros de la red, a saber, la función de activación, el dropout y el optimizador proveído por Pytorch, permitiendo seleccionar los parametros que mejoran el rendimiento de la red. Finalmente se compara su rendimiento con un clasificador de una sola capa oculta no convolucional. Se descubre que su rendimiento es mejor a esta, pero no lo suficiente como para justificar el costo computacional. Se concluye que deberían optimizarse otros hiperparámetros.**

***Keywords* – Red neuronal convolucional (CNN), Función de activación, Dropout, Optimizador**

NOMENCLATURA

CNN Red neuronal convolucional (convolutional neural network)
ANN Red neuronal lineal (artificial neural network)

I. INTRODUCCIÓN

Las CNN son utilizadas ampliamente en tareas tales como reconocimiento de imágenes o procesamiento de lenguaje natural por ser altamente competitivas, logrando mejores resultados que las ANN. Su arquitectura consiste de una serie de operaciones de convolución y "pooling", seguido de una serie de capas lineales conectadas. Estas capas de convolución permiten reconocer características cada vez mas complejas a medida que la red se vuelve mas profunda. En el caso de una imagen empiezan reconociendo bordes y formas simples, y a medida que se le agregan mas capas y filtros puede empezar a reconocer formas mas complejas pertinentes a los datos que se le proporcione, por ejemplo ojos u orejas (si por ejemplo debe reconocer fotos de animales). Este aspecto de las CNN es lo que las hace idóneas para la tarea de reconocimiento de imagenes [1].

Para este trabajo, se eligió una arquitectura consistente en dos capas convolucionales, ambas con un kernel de 5*5 y stride 2 y dos capas lineales conectadas [2]. La arquitectura no varió durante la optimización del modelo. Se eligió optimizar el modelo sobre la variacion de diferentes dropouts para las capas lineales, sobre dos funciones de activación distintas y sobre dos optimizadores. Se utilizo PyTorch para el armado, entrenamiento y validación de la red.

II. VARIACIÓN DE HIPERPARÁMETROS

A. Dropout

Se eligió variar el dropout de la red entre los valores 0.1, 0.3 y 0.5. Los resultados obtenidos para los valores de loss y accuracy por cada modelo se plasman en el siguiente gráfico.

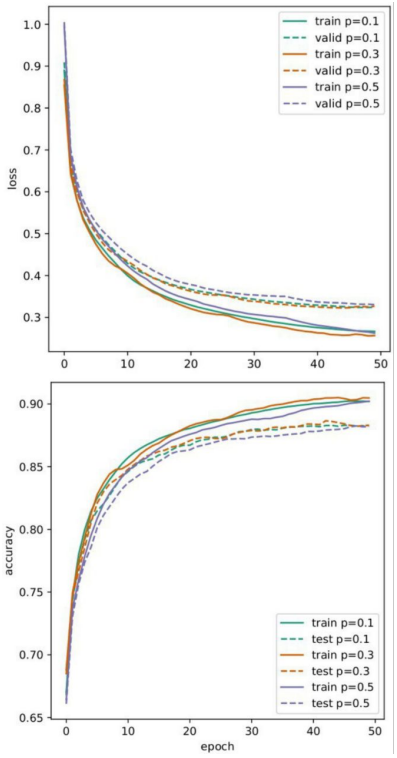

Fig. 1. Loss y accuracy para diferentes valores de dropout

Se observa que para $p = 0.3$ se consiguen mejores resultados tanto en la loss como en la accuracy, siendo la diferencia entre validación y entrenamiento similar para los tres valores elegidos.

1) Función de activación: A continuación, habiendo optimizado el dropout, creamos dos modelos similares excepto que en uno utilizamos una función de activación ReLU y en el otro la Leaky ReLU, tanto en las capas lineales como en las convolucionales. Los resultados obtenidos son los siguientes.

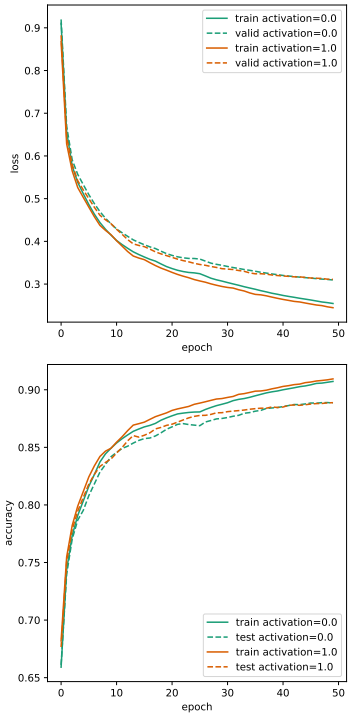

Fig. 2. Loss y accuracy para ReLU=0.0 y Leaky ReLU=1.0

Se observa que con la Leaky ReLU el error desciende mas rápido y la accuracy llega a valores ligeramente mayores, mientras que la accuracy y loss de validación alcanzan valores mas altos y bajos, respectivamente.

2) Optimizador: Variamos entre los optimizadores SGD (stochastic gradient descent) y Adam proveídos por Pytorch. Los resultados obtenidos son los siguientes

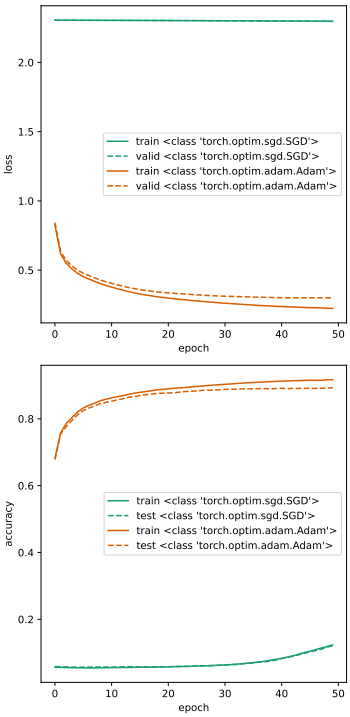

Fig. 3. Loss y accuracy para diferentes optimizadores

Se observa que claramente el mejor optimizador para esta red es Adam, el error casi no disminuye ni en el entrenamiento ni en la validación con SGD y en la accuracy llega a resultados en absoluto satisfactorios.

III. ENTRENAMIENTO DE LA RED CON HIPERPARÁMETROS OPTIMIZADOS

Una vez optimizados los tres hiperparámetros que variamos anteriormente, entrenamos a la red sobre 100 épocas de entrenamiento y le pedimos que nos dé las predicciones que obtiene sobre la base de datos de Fashion-MNIST. Las curvas para loss y accuracy quedan como se muestra a continuación.

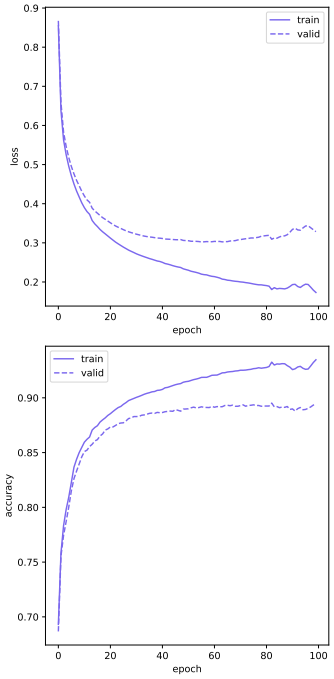

Fig. 4. Loss y accuracy para 100 épocas de entrenamiento

Observamos que las curvas de validacion se estancan en alrededor de 0.33 para el loss y 0.88 para la accuracy a partir de aproximadamente 40 épocas, mientras que las curvas de entrenamiento siguen optimizando el aprendizaje sobre el conjunto de entrenamiento. Estimamos que mas alla de 40 épocas la red incurre en overfitting.

Se incluyen algunas imágenes elegidas al azar del dataset de testeo de Fashion-MNIST junto con las predicciones realizadas por la red.

IV. COMPARACIÓN EN UN CONJUNTO DE TESTEO CON UNA ANN

Por ultimo, procedemos a comparar el desempeño de nuestra red CNN con el mejor rendimiento posible de una red neuronal simple de una sola capa oculta. Se ha visto [0] que una red de este tipo consigue su mejor desempeño con 512 neuronas en la capa oculta y 36 épocas de entrenamiento. Lo que hacemos a continuación es dividir el conjunto de entrenamiento de Fashion-MNIST en dos conjuntos: uno de 50000 imagenes que usaremos como entrenamiento y otro de 10000 imágenes que usaremos con validación, para luego utilizar en ambas redes el conjunto de testeo para juzgar su

Fig. 5. Coat

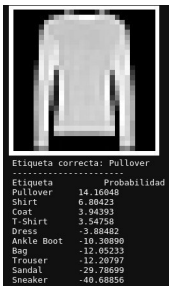

Fig. 6. Pullover

Fig. 7. Sneaker

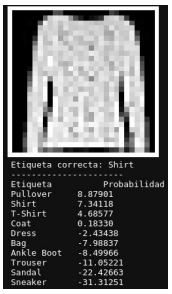

Fig. 8. Shirt

Fig. 9. T-shirt

Fig. 10. Sandal

desempeño. Entrenamos y validamos a cada red con estos dos conjuntos, guardamos el mejor modelo conseguido de cada una y a ese modelo lo testeamos. Los resultados obtenidos son:

```
test_loss ANN =  0.3300893396139145
test_accu ANN =  0.8852

test_loss CNN =  0.31221417188644407
test_accu CNN =  0.8879
```

Se observa que los resultados fueron mejores en la CNN que en la ANN pero no lo suficiente como para justificar el costo computacional de una frente a la otra. Como las redes convolucionales son de acuerdo a la bibliografía mejores que las ANN especialmente en el reconocimiento de imágenes, se deduce que tal vez la arquitectura elegida para esta red no fue la apropiada y debieron haberse mejorado otros hiperparámetros.

V. CONCLUSIONES

Se logró implementar una CNN profunda con dos capas convolucionales, optimizando sobre tres hiperparámetros elegidos. En la comparación sobre un conjunto de testeo con una ANN se desempeño mejor que esta pero no lo suficiente como para justificar el costo computacional. Se debería intentar optimizar sobre otros hiperparámetros de la red o considerar otra arquitectura para mejorar su desempeño.

REFERENCES

[1] https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[2] https://towardsdatascience.com/build-a-fashion-mnist-cnn-pytorch-style-efb297e22582

https://github.com/jipphysics/curso-redes-neuronales-famaf-2021/ blob/master/practicos/practico0/pytorch/entrenamiento-validacion-prueba.ipynb.

Dubey, A. K., Jain, V. (2019). Comparative Study of Convolution Neural Network's Relu and Leaky-Relu Activation Functions. Applications of Computing, Automation and Wireless Systems in Electrical Engineering, 873–880. doi:10.1007/978-981-13-6772-4 76