Clasificación de imágenes basado en texturas utilizando Redes Neuronales Convolucionales

Rubén Darío Bohórquez Cortázar
Universidad de los Andes
Bogota, Colombia
rd.bohorquez10@uniandes.edu.co

Javier Felipe Cifuentes
Universidad de los Andes
Bogota, Colombia
jf.cifuentes10@uniandes.edu.co

Abstract

En este laboratorio se exploró el uso de redes neuronales convolucionales para la clasificación de imágenes basado en textura. Para esto, se procedió a diseñar y entrenar una CNN a partir de una inicialización de pesos al azar. Todo esto se realizó con base al toolbox Matconvnet de Matlab. Con la CNN desarrollada se obtuvo un ACA de 61.01. Adicionalmente, se encontró que utilizar más épocas en el entrenamiento de la red mejora el desempeño en la clasificación. Por otro lado, al no implementar Jitter, se disminuye el desempeño de la red. Con base a estos resultados, se puede concluir que remover capas de la red genera un fallo en el sistema, aumentando sustancialmente el error asociado. La capa ReLU, asociada a la parte no lineal de la red, fue la que generó una mayor caída en el desempeño de la CNN al ser removida, lo que expone la importancia de esta capa en la red. Finalmente, es de gran relevancia resaltar la importancia de la herramienta MatConvNet, para el rápido desarrollo y prototipado de redes neuronales convolucionales.

1. Introducción

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) han sido ampliamente estudiadas en los últimos años debido a las múltiples ventajas y beneficios que aportan a los distintos campos de la visión artificial, asociados a su capacidad de extracción de características, procesamiento de datos y adaptabilidad a las bases de datos estudiadas [1]. Las CNNs desde sus primeras implementaciones han demostrado un excelente rendimiento en la solución de problemas asociados al reconocimiento y la clasificación de imágenes. Estas arquitecturas están inspiradas en procesos biológicos como el sistema visual y su procesamiento cerebral [1]. Son diseñadas para llevar a cabo un reconocimiento de patrones a partir de los propios datos de entrada incorporando tanto la etapa de extracción automática de características como la de clasificación. Esta cualidad es tal vez la más importante debido a que confiere a esta herramienta (CNN) la capacidad de adaptarse a las imágenes de la base de datos de entrada y exponer la mejor solución partiendo de las características específicas de la misma. Debido a este aprendizaje y a la extracción automática de características, no es necesario una definición y extracción manual de las mismas, lo que reduce el problema de la clasificación y el reconocimiento al diseño óptimo, basado en prueba y error, de una red neuronal convolucional [1].Con base a esto, se plantea como el objetivo principal de este laboratorio el explorar el uso de redes neuronales convolucionales para la clasificación de imágenes basado en textura. Para esto, se procederá a diseñar y entrenar una CNN a partir de una inicialización de pesos al azar. Todo esto se realizará con base al toolbox Matconvnet de Matlab el cual implementa algoritmos para redes neuronales convolucionales con suporte GPU.

2. Materiales y métodos

2.1. MatConvNet

MatConvNet nació en el Oxford Visual Geometry Group como una plataforma educativa y de investigación para el prototipado rápido en Redes Neuronales Convolucionales [2]. Sus principales características son:

- Flexibilidad: Las capas de red neuronal se implementan de una manera directa, a menudo directamente en el código de MATLAB, de modo que son fáciles de modificar, extender o integrar con otras nueva[2].
- Poder: La implementación puede llevar a cabo modelos grandes tales como Krizhevsky et al., Incluyendo las variantes DeCAF y Caffe. Adicionalmente se proporcionan varios modelos pre-entrenados [2].
- Eficiencia: La implementación es bastante eficiente, soportando tanto la CPU como la GPU [2].

2.2. Descripción CNN

Esta red utiliza capas de convolución, con diferentes números de filtros, así como con filtros de diferentes tamaños. Adicionalmente usa capas de ReLU, capas de pool, y una capa final de softmax. Las capas de pool, generalmente reducen las dimensiones de la imagen a la mitad, es decir, si la imagen es de 100x100, pasar a ser de 50x50. Todas las capas usan el máximo como método de pooling. La arquitectura consiste en 14 capas. Este diseño se basa en el ejemplo proporcionado por el grupo de visión de Oxford. En este se utiliza una red de 37 capas, en la cual se sigue un patrón claro. Este patrón es Convolucional-ReLU-Convolucional-ReLU-Pool. Al final una capa de softmax. Se utilizó este patrón, dado que esta red fue utilizada para clasificación en la base de datos imageNet. Se realizó más pequeña, dada el número de imágenes y de clases a utilizar. Se siguió el mismo patrón.

3. Resultados

Se obtuvieron las siguientes gráficas para cada una de las redes estudiadas:

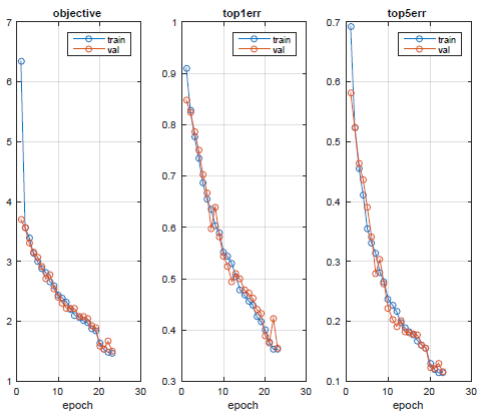

Figure 1. Red completa entrenada con 23 épocas

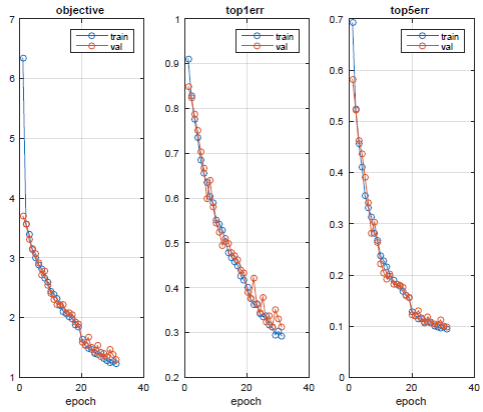

Figure 2. Red completa entrenada con 31 épocas

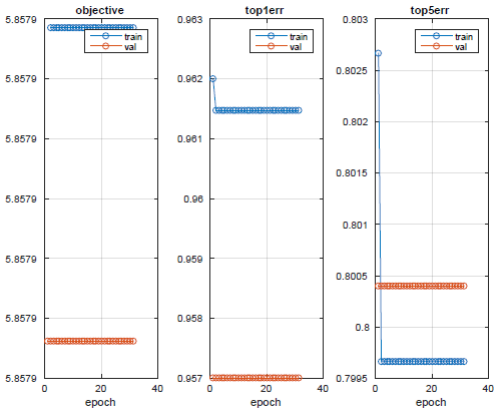

Figure 3. Red sin la ultima capa convolucional etrenada con 31 épocas

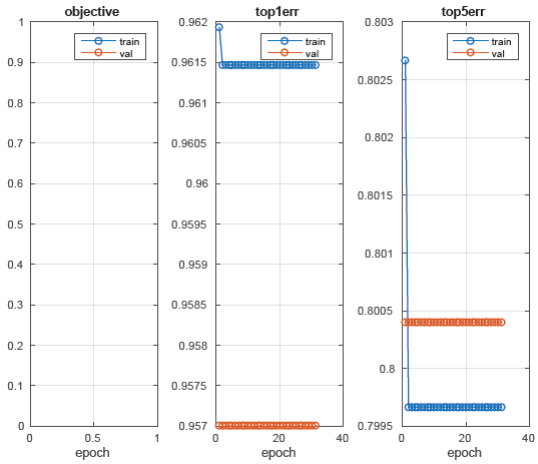

Figure 4. Red sin la ultima capa ReLU etrenada con 31 épocas

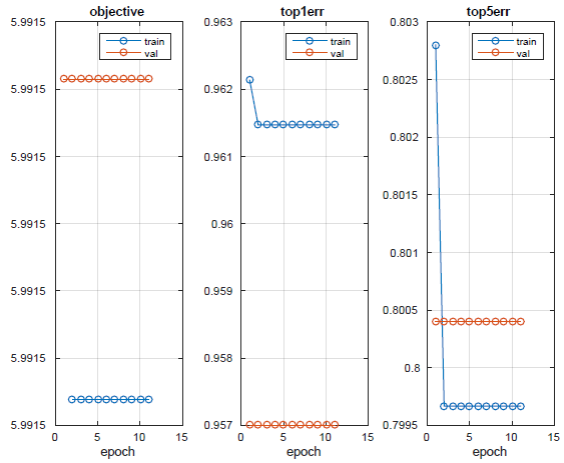

Figure 5. Red sin en el ultimo bloque de capas convolucional, ReLU,convolucional, ReLU y pool etrenada con 31 épocas

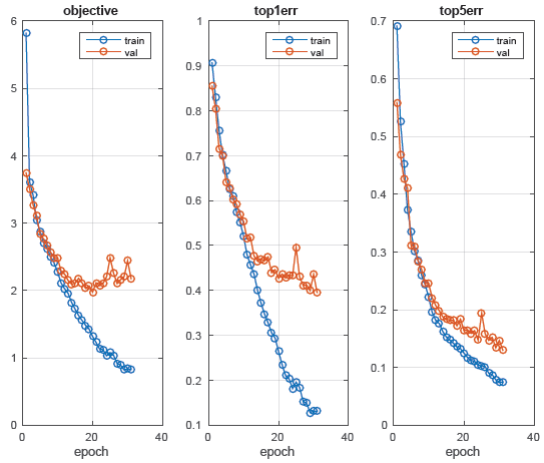

Figure 6. Red completa entrenada con 31 épocas sin Jitter

CNN	ACA
Entrenada con 23 épocas	56.84
Entrenada con 31 épocas	61.01
Entrenada con 31 épocas sin Jitter	59.22

Table 1. Valores de los ACA para cada CNN estudiada

4. Análisis de resultados

De las figuras 2 y 6, se puede ver el efecto que tiene el uso de Jitter en el desempeño de la red estudiada para la clasificación de texturas. Se puede observar que al no utilizar Jitter, figura 6, se logran mejores resultados en la fase de training, es decir errores más bajos y un objetivo más pequeño, sin embargo, la validación toma valores de error y de objetivo superiores, con una diferencia de casi 0.2 por época. Por otro lado, en la red entrenada con Jitter, 2, se observa que los errores obtenidos en la fase de training son un poco más altos, sin embargo, en la fase de validación, los errores son muy parecidos a la fase de training. Adicionalmente, se debe resaltar que el error en la validación al utilizar Jitter es mucho menor que el error en la validación sin utilizar Jitter, lo que implica que utilizar Jitter mejora el desempeño de la CNN. Debido a esto, se puede afirmar que implementar Jitter si ayuda a obtener mejores resultados.Esto se puede corroborar observando la tabla 1, en la cual se muestra que el ACA para la clasificación con la red con Jitter es de 61.01, mientras que para la clasificación con la red sin Jitter es de 59.22. Esto muestra claramente que implementar Jitter en el desarrollo de la red aumenta el desempeño de la clasificación.

Para el estudio del efecto de diferentes capas sobre la red neuronal, se realizó un procedimiento en el cual se removían capas específicas del final de la red y se evaluaba nuevamente su desempeño. Se removieron específicamente, la última capa convolucional, la última capa de ReLU y el último bloque de capas convolucional-ReLU-convolucional-ReLU-pool. Estos resultados se pueden ver en las figuras 3, 4 y 5 respectivamente. Como se puede observar en estas figuras, la remoción de estas capas género que el sistema dejara de funcionar correctamente, ya que, al remover estas capas, tanto el error top1 así como el error top5 se mantuvieron constantes en valores muy cercanos a 1. Adicionalmente el objetivo se mantuvo en valores bastante altos en los tres casos. Se puede notar que estas son redes que no aprenden, dado que no baja el error, así como el objetivo. Esto demuestra la importancia de cada una de estas capas. Específicamente la capa de ReLU, género que los valores de objetivo fueran no numéricos, específicamente NaN. En los otros casos los objetivos se mantuvieron como doubles de valor constante. Esto implica la importancia de la capa de ReLU, lo cual tiene coherencia, dado que esta capa implementa la parte no lineal de la red.
Al realizar el entrenamiento de la red con 23 y 31 épocas, como se muestra en las figuras 1 y 2 , se pudo observar que el error asociado tanto al entrenamiento como a la validación disminuía conforme aumentaban las épocas. Resultado que se refleja en un mejor desempeño para la clasificación utilizando la red completa entrenada con 31 épocas. Esto se puede corroborar observando la tabla 1, en la cual se expone los resultados de los ACA para cada red. En esta se muestra que para la red con 23 épocas, el ACA fue de 56.84 mientras que para la de 31 épocas, fue de 61.01.

5. Discusión

El reto más importante a la hora del diseño de la arquitectura es al momento de calcular las salidas y entradas de cada capa. Estos valores tienen que encontrarse correcta-

mente establecidos, dado que, la red no entrena si estos no coinciden. Uno de los fenómenos que se pueden notar al no realizar correctamente este cálculo, es que en algunos puntos los filtros terminan siendo más grandes que las imágenes a evaluar. Fue necesario probar los tamaños varias veces, hasta encontrar unos valores de tamaños de filtros que fueran adecuados para que la red funcionara. Fue necesario algún tiempo, para encontrar que la capa de softmax requiere vectores como parámetros, de modo que, al ingresar matrices a esta capa, varias veces no funcionaba. También fue necesario encontrar una secuencia de capas que generará un resultado óptimo. El orden en el que se colocan las capas convolucionales, de ReLU y pool afecta de manera significativa el resultado. Se encontró secuencias en las cuales la red no entrenaba, manteniendo el error siempre constante. Finalmente, el tamaño fue el último factor a tener en cuenta. Se encontró que redes demasiado pequeñas no entrenan, mientras que redes demasiado grandes pueden ser difíciles de construir así como de entrenar.

6. Conclusiones

- Se logró diseñar, desarrollar e implementar una red neuronal convolucional para la clasificación de imágenes basado en la textura. Se obtuvo un ACA de 61.01.

- Implementar Jitter en la CNN permite disminuir el error asociado a la clasificación, aumentando el ACA y por ende, mejorando el desempeño de la red.

- Se puede afirmar que remover capas de la CNN, genera una falla en el sistema, lo que se ve reflejado en un incorrecto funcionamiento del mismo. Esto demuestra la importancia de cada una de estas capas. Específicamente, al remover la capa de ReLU, se obtuvo una caída importante del desempeño, lo cual expone de la parte no lineal de la red en el desempeño de la misma.

- Se puede afirmar que implementar más épocas en el entrenamiento de la CNN, genera un aumento considerable en el desempeño de la red en la clasificación de imágenes.

- Finalmente, es de gran relevancia resaltar la importancia de la herramienta MatConvNet, para el rápido desarrollo y prototipado de redes neuronales convolucionales.

References

[1] Paris, J. L. G. S., Nakano-Miyatake, M., Robles-Camarillo, D. Comparación de Arquitecturas de Redes Neuronales Convolucionales para

[2] About MatConvNet. (n.d.). Retrieved May 11, 2017, from http://www.vlfeat.org/matconvnet/about/