Laboratory 11: Texton CNN

Fabian Martínez
Universidad de los Andes
Bogota
fe.martinez10@uniandes.edu.co

Juan Felipe Pérez
Universidad de los Andes
Bogota
jf.perez10@uniandes.edu.co

Abstract

En el siguiente informe se presentan los procesos y resultados obtenidos al realizar la creación de una red neuronal convolucional con el objetivo de clasificar texturas.

1. Introducción

Las redes neuronales han tenido un gran auge en los últimos años, ya que estas constituyen un fuerte mecanismo para la solución de diversas clases de problemas, que van desde aproximaciones de funciones, clasificación e identificación de información, y con una gran cantidad de aplicaciones en áreas como por ejemplo la economía, medicina, optimización e investigación.

Estas redes originalmente fueron una simulación abstracta de lo que son los sistemas nerviosos de los seres vivos, tomando el concepto de estar constituidos por un conjunto de unidades llamadas neuronas y quienes están conectadas por medio de lazos o dendritas unas con otras.

Las redes neuronales utilizadas para la solución de problemas de visión son conocidas como redes neuronales convolucionales. Estas redes están basadas en el funcionamiento que presenta el sistema visual humano, en especial atención a los campos receptivos que se presentan en las células de la corteza visual primaria.

Las redes neuronales convolucionales están compuestas de capas de diferentes tipos, como por ejemplo las neuronas convolucionales o las de submuestreo, en las que esta arquitectura las hace óptimas para tareas de visión artificial, especialmente en la que corresponde a la clasificación de imágenes.

1.1. Objetivos

1. Crear una red neuronal con diferentes tipos de capas con el objetivo de clasificar diversos tipos de texturas.

2. Modificar los diferentes parámetros involucrados en una red neuronal convolucional con el objetivo de observar las diferencias en cada uno de los resultados.

3. Obtener las gráficas correspondientes al error de entrenamiento y validación de una red neuronal para la clasificación de texturas.

2. Materiales y métodos

Para el siguiente laboratorio se utilizaron las funciones de Matlab de MatConvNet, con las cuales se pueden crear redes neuronales convolucionales.

Como recursos computacionales se utiliza una GPU NVIDIA GEFORCE GTX 950M, sobre la cual es posible entrenar y validar la red neuronal convolucional en un tiempo corto.

2.1. Red neuronal convolucional

Para el desarrollo del laboratorio se implementaron varias redes neuronales convolucionales, en las que se varían algunos de los parámetros para evaluar la eficiencia y el desempeño que presentaban en el momento de clasificar las diferentes texturas. Las capas que hacen parte de las redes neuronales convolucionales son:

- Capa convolucional: encargada de realizar convoluciones con matrices de pesos aleatorios (inicialmente). A lo largo de la red, estas capas cambian el tamaño de la matriz de convolución. La función de estas capas convolucionales radica en extraer los rasgos característicos de las imágenes por medio de un banco de filtros sobre la imagen (o respuestas anteriores) de entrada a la capa.

- Capa relu: encargada de hacer una función de máximo entre 0 y los valores de la capa anterior. Fundamentalmente, es una capa no lineal que se encarga de eliminar las respuestas negativas de la capa anterior. Entre las principales ventajas que presenta el uso de Relu como una capa de no linealidad a comparación de otras

(como por ejemplo la tanh o la sigmoide) es la capacidad de acelerar los procesos de entrenamiento e igualmente evita los problemas de desvanecimiento de gradiente a medida que aumenta el número de capas utilizadas en la red neuronal. En general, la función Relu esta definida como:

$$f(x) = max(0, x)$$

- Capa de Pooling (maxPooling): encargada de realizar una operación no lineal que reduce dimensionalidad. Se divide la imagen en ventanas de un tamaño determinado, y que no se sobrelapen entre ellas, y en cada ventana se determina el máximo. Cabe destacarse que existen otros tipos de pooling como por ejemplo el promedio, sin embargo este no presenta tan buenos resultados. Este valor obtenido pasa a reemplazar a todos los de su ventana. La ventaja del uso de la capas de pooling es que disminuye la cantidad de procesamiento que se necesitan realizar (en especial cuando la red es muy profunda) y evita que la red tenga un sobreajuste a los datos de entrenamiento.

- Capa de softmax: función exponencial realizada con el ánimo de encontrar el máximo valor de la capa anterior, pero que a su vez pueda ser diferenciable con el objetivo de que se puedan clasificar las imágenes en las diferentes categorías.

Entre una de las principales características que se logran observar es que el mejor funcionamiento de la capa de pooling es cuando esta se realiza justamente después de cada una de las capas de convolución.

Las principales ideas (cambios de parámetros) que se buscaron cambiar en el diseño de las redes fueron:

- Número de capas: Se busca encontrar el número de capas óptimo para llevar a cabo la clasificación de las imágenes. Con el fin de tener un desempeño óptimo se inicia con un número pequeño de capas, sin embargo la clasificación presenta una gran cantidad de error, por lo tanto se aumenta poco a poco el número de capas,sin embargo se tiene en cuenta que agregar demasiadas implica un costo computacional bastante alto y que a pesar de que el error puede bajar, los cambios empiezan a ser poco considerables a comparación al costo de recursos.

- Valor de la tasa de aprendizaje: Un valor óptimo de la tasa de aprendizaje implica una correcta clasificación. Si este valor es demasiado alto no disminuye el error y se comienza a diverger en la clasificación, mientras que una tasa demasiado baja implica una gran cantidad de épocas de entrenamiento aumenta para poder disminuir el error.

- Número de filtros: La cantidad de filtros utilizados en las capas convolucionales indican el número de características que es posible obtener de una imagen de entrada.

- Tipo de pooling: Se busca cambiar el tipo de pooling o submuestro de las imágenes.

- Valores iniciales: Se busca cambiar los valores iniciales de los diferentes pesos, lo cual se realiza por medio de números aleatorios entre diferentes valores.

- Tamaños de ventana: Se cambian el tamaño tanto de los filtros utilizados como en las capas de pooling (tamaños de ventana de 2, 3, 4 etc.) con el fin de que se observen características diferentes en las diferentes redes.

- Número de épocas: La cantidad de entrenamientos en las redes implica una dismunición del error en el proceso de clasificación, sin embargo llega un momento en el que el error nodisminuye (o disminuye muy poco) a medida que se aumenta el número de épocas.

2.2. Uso del jitter

Por lo observado durante el desarrollo de la etapa de entrenamiento, el uso del jitter es muy útil a la hora de reducir el tiempo de procesamiento, sin reducir en gran medida los resultados del aprendizaje. Este jitter lo que hace es encontrar rápidamente los parches, en este caso de 128x128 píxeles, que definen una textura; utilizando menor cantidad de instrucciones que con un getBatch.

2.3. Arquitectura de la red, retos y problemas presentados

En un principio, entrenamos una red de solo 3 capas convolucionales, otras 3 de maxpooling, y finalmente una de softmax. Luego de las 23 épocas de entrenamiento y validación, el top 5 error era de 0,59, y para que la red aprendiera, era necesario ingresarle una tasa de aprendizaje del orden de $1 * 10^{-6}$, por lo que en las ultimas épocas se podía evidenciar que los cambios en el error eran casi nulos de una imagen a la siguiente.

De esta manera, fue posible observar que, entre mayor el numero de capas, y a su vez, entre menor el tamaño de las matrices para la convolución, mayor el aprendizaje logrado y menores las tasas de error.

Sin embargo, cabe resaltar que durante el diseño de la arquitectura ocurrieron varios problemas. Sin lugar a dudas, el problema más común fueron los tamaños de los resultados de las capas. Más específicamente, que no cuadraran los tamaños de la penúltima capa con la última, en las que tendrían que ser capas "fully-connected"; o que se realizaban tantas convoluciones que las matrices de pesos para

la convolución en una capa resultaban ser mayores que las matrices resultantes de la capa anterior. Se tuvieron que realizar muchos cambios hasta obtener una red convolucional en la que se lograra un aprendizaje óptimo en comparación con redes anteriores, y en la que las etapas de entrenamiento y validación funcionaran a la perfección.

Uno de los principales retos que implica la construcción de redes neuronales radica en encontrar los valores óptimos para cada uno de los parámetros existentes con el fin de encontrar el mejor desempeño. En teoría, la cantidad de posibles modelos para la solución de un es infinito, sin embargo debe limitarse el problema dependiendo de los recursos computacionales, el tiempo y los cambios que se presentan al configurar los valores de los diferentes criterios de las redes neuronales.

3. Resultados

Luego de correr el algoritmo de entrenamiento, obtuvimos los siguientes resultados en la arquitectura propuesta:

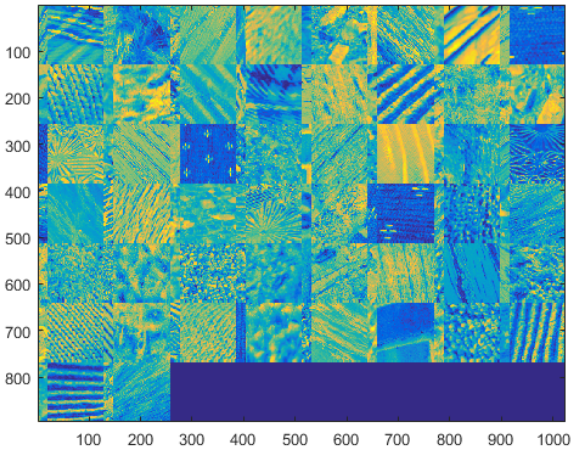

Fig. 1. Imagen de texturas

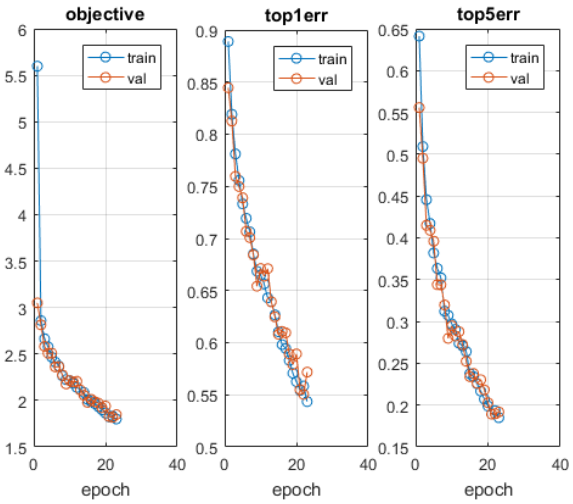

Fig. 2. Imagen correspondiente a los errores a lo largo del proceso de entrenamiento

En la figura anterior es posible observar la evolución que presenta la red neuronal convolucional a lo largo de las diferentes épocas de entrenamiento y de validación. Se puede observar que inicialmente el error obtenido decae de una manera bastante rápida, sin embargo en las etapas mas avanzadas la disminución del error es menor, lo que implica que los valores de los pesos no tienen un cambio tan drástico. Por otra parte puede apreciarse que en la red planteada los errores de entrenamiento y validación son bastante cercanos, lo que deja como conclusión de que no se presenta un sobreajuste de la red a los datos de entrenamiento.

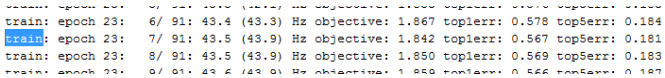

Fig. 3. Mejor resultado en entrenamiento

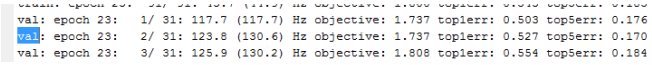

Fig. 4. Mejor resultado en validación

En resumen, se llegó a un top 5 error de 0,17; lo cual es un error bastante bajo teniendo en cuenta el tamaño de la red y el hecho de que para esta arquitectura no se usó ninguna capa ReLu. Esto se logró luego de 23 épocas tanto de entrenamiento como de validación, proceso que tomó un tiempo aproximado de 2 horas.

4. Conclusiones

1. Por lo observado en el desarrollo de la arquitectura de la red, y su posterior desempeño en la etapas de entrenamiento y validación, a mayor cantidad de capas en

la red, menor la cantidad de error obtenido luego de varias iteraciones.

2. El tamaño de los filtros de pesos determina en gran medida que tan buena es la arquitectura. Por un lado, con matrices más pequeñas los cambios y operaciones aplicados en una capa son más específicos. Por otro, el hecho de que sean de menor tamaño permite posteriormente mayor numero de capas convolucionales, pues la reducción del tamaño de la respuesta es menor.

References

[1] Raúl Lopez (2016). Redes neuronales convolucionales con TensorFlow. Matemáticas, análisis de datos y python. Tomado de $http://relopezbriega.github.io$

[2] Erick Zamora. Redes neuronales convolucionales. Tomado de $https://es.scribd.com/doc/295974900/Redes-Neuronales-Convolucionales$