Convolutional Neural Networks for Image Classification

José María Campo Viñas
Universidad de los Andes
Código: 201412002
jm.campo11@uniandes.edu.co

Cristian Amaya
Universidad de los Andes
Código: 201325578
cm.amaya10@uniandes.edu.co

Mariana Franky
Universidad de los Andes
Código: 201313944
m.franky10@uniandes.edu.co

Abstract

En este laboratorio se exploró el uso de Redes Neuronales Convolucionales (CNN) para la clasificación de imágenes. Se diseñó y entrenó una CNN a partir de una inicialización de peso al azar. Esta red contiene 14 capas, donde cinco son convolucionales, cuatro son de ReLU, cuatro de pooling y una de softmax.

1.. Introducción

Las redes convolucionales (CNN) son un tipo especial de redes neuronales que ha sido ampliamente aplicada a una variedad de problemas de reconocimiento de patrones, siendo una exitosa herramienta para solucionar problemas de procesamiento de voz y visión computacional, superando el desempeño de métodos anteriores. La idea básica de CNN es construir propiedades de invariancia en redes neuronales creando así modelos invariantes a ciertas transformaciones de entrada, las CNN tienen una arquitectura especial, esta se compone de una capa convolucional, un submuestreo que se conoce como 'pooling', una no linelidad ReLU, y un clasificador softmax.[1]

Capa convolucional: Ésta operación recibe como entrada la imagen y luego sobre la imagen se aplica un filtro o kernel que devuelve un mapa de características de la imagen original logrando reducir el tamaño de los parámetros.[2]

Capa de pooling: su utilidad principal radica en la reducción de las dimensiones espaciales (ancho x alto) del volumen de entrada para la siguiente capa convolucional. La operación que se suele utilizar en esta capa es max-pooling, que divide a la imagen de entrada en un conjunto de rectángulos y respecto de cada subregión, se va quedando con el máximo valor.[2]

Capa de softmax : las redes utilizan generalmente capas completamente conectados en la que cada pixel se considera como una neurona separada al igual que en una red neuronal regular. Esta última capa clasificadora tendrá tantas neuronas como el número de clases que se debe predecir.[2]
Este laboratorio tiene como objetivo diseñar y entrenar una red convolucional.

2.. Métodos y materiales

2.1.. Red Convolucional

La red convolucional fue diseñada para imagenes de 64x64 y se basa en 5 etapas, las primeras 4 están compuestas por una capa convolucional, una capa de ReLU y una capa de pooling de 2x2. La ultima etapa consta de una última capa convolucional y una capa softmax, que produce un output de 1x1x25 por cada batch de imágenes. Se plantearon diseños diferentes sin tantas capas de pooling, pero debido a limites de memoria, estas eran las más necesarias.

Para el entrenamiento, se usaron batches de 165 imagenes de train.

2.2.. Metodos

2.2.1. Jitter64

El método de Jitter consiste en tomar las imágenes de entrenamiento y variar su posición dentro de un fondo abarrotado, de ésta forma, aquellos filtros y respuestas que aprenda la red no se encontrarán asociados a la posición espacial del objeto, sino a las características intrínsecas de estos.

2.2.2. Test de ablación

Con el fin de explicar el funcionamiento de la red y el efecto en el rendimiento cuando se eliminan algunas de las capas de esta. Se realizaron varios experimentos:

- ***ReLU:*** Esta red se compone de cuatro capas de ReLU, para ver el efecto de estas capas en la red, se eliminó las capas de ReLU por etapa. Inicialmente se quitó la capa de la primera etapa y se entrenó de nuevo la red. Luego se agregó nuevamente esta capa y se eliminó

la capa de ReLU en la segunda etapa y se entrenó de nuevo la red, el mismo procedimiento se realizó con las etapas 3 y 4.

- ***Pooling:*** Se eliminó la capa de pooling en la etapa 2 y se entrenó de nuevo la red. Se agregó nuevamente esta capa y se eliminó la capa de pooling en la etapa 1 y se entrenó de nuevo la red.
- ***Pooling y ReLU:*** Para visualizar el efecto conjunto de las capa de pooling y ReLU, se eliminó de las etapas 2, 3 y 4 las capas correspondientes a ReLU y de la etapa 4 se eliminó la capa correspondiente a pooling.
- ***Capa de convolución:*** Se eliminó la capa de convolución y pooling de la cuarta etapa.

3.. Resultados

A continuación se muestran las gráficas correspondientes a los experimentos y el resultado final de una de nuestras redes, cabe resaltar que este resultado no es el mejor que se obtuvo. En la figura 1 se muestra el comportamiento de top1err, top5err y el objective, de la red convolucional implementada.

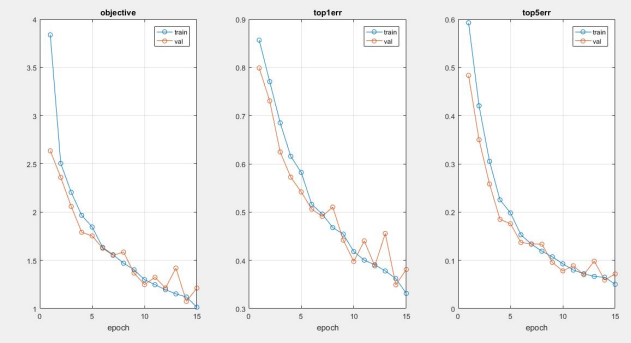

Figura 1. Resultado CNN 15 épocas

Las figuras 2 y 3 muestran la incidencia en los resultados al quitar las capas de ReLU en ciertas etapas en la red.

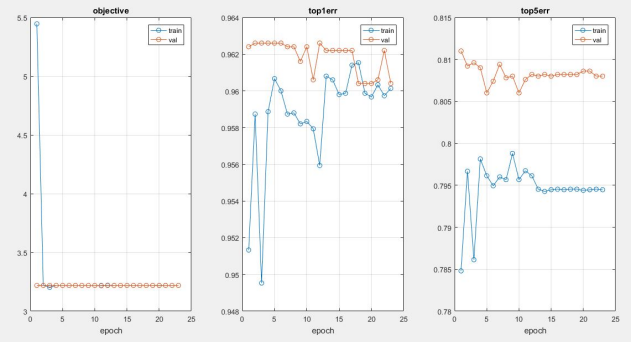

Figura 2. Resultado CNN quitando ReLU en la primera etapa.

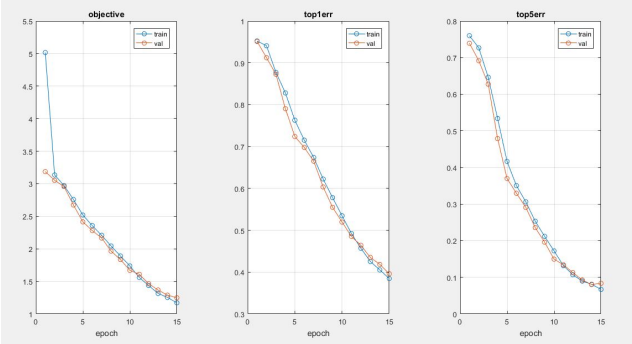

Figura 3. Resultado CNN quitando ReLU en la tercera etapa

La figura 4 muestra la incidencia en los resultados al quitar la capa de pooling en la segunda etapa en la red.

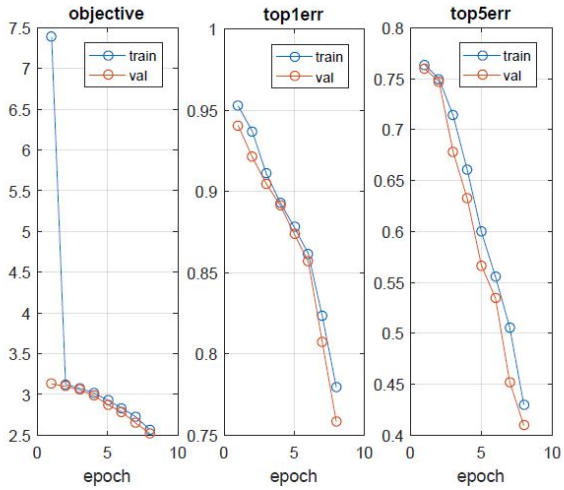

Figura 4. Resultado CNN quitando Capa pooling en la segunda etapa

La figura 5 muestra la incidencia en los resultados al quitar la capa de pooling en la cuarta etapa y quitar ReLU en la etapa 2, 3 y 4 en la red.

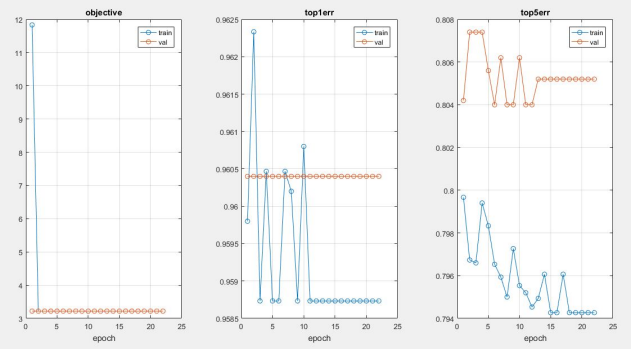

Figura 5. Resultado CNN quitando Capa pooling en la cuarta etapa y ReLU en segunda, tercera y cuarta etapa

Al quitar la capa de convolución y pooling en la cuarta etapa la red no entrenó.

4.. Discusión y Conclusiones

Inicialmente para diseñar la red se tomó como base la arquitectura propuesta en el ejemplo del laboratorio, esta red cuenta con 8 capas de las cuales tres son convolucionales, dos son de ReLU, dos de pooling y una de softmaxloss. Antes de lograr que una red entrenara, se diseñaron 4 redes, uno de los problemas que se presentaron fue que muchas veces al entrenar la red, el equipo en donde se entrenaba se quedaba sin memoria, por esta razón se comenzó a diseñar la arquitectura de la red con varias capas de pooling para disminuir el tiempo computacional. Otra dificultad que se presentó al inicio fue calcular la siguiente capa de convolución, dado que no se entendía el input que entraba a la otra capa, una vez que se entendió se logró modificar la red original. De la red original, se tomó de la arquitectura que por cada etapa se realizaba una convolución, un ReLU y un pooling, dando como resultado 14 capas de las cuales cinco son convolucionales, cuatro son de ReLU, cuatro son de pooling y una es de Softmax.

Los resultados de la red creada al llegar a la época 15 fueron un valor de objetivo de 1.2, top1err de 0.35 y top5err de 0.17 aproximadamente. Como era de esperarse, todas las pruebas de ablación poseen un error mayor a la red creada.

Ahora bien, una vez que se logró entrenar la red, se visualizó el efecto que tenía quitar algunas capas y entrenar el modelo de nuevo. En los resultados se observa que muchas de las redes modificadas no llegan a la última época, esto se debe a que después de una época su comportamiento no variaba o era menor al de la red sin modificar.

Se puede denotar la importancia de la capa ReLU en las primeras etapas al observar en la Figura 2 que el error no varía al pasar de épocas, mientras que en las últimas etapas, éste no es de gran importancia como lo demuestra la Figura 3 al compararlo con la Figura 1. Una de las posibles razones por las cuales afectó en gran medida el resultado, puede deberse a que gran parte de la información se encuentra denotada en las variaciones de alta frecuencia como se observa en algunas imágenes dentro de la Figura 6.

Sabiendo que las primeras capas tienen en cuenta una sección pequeña de la imagen y las altas una grande, al no crear una no linealidad entre dos capas contiguas, en teoría ambas convoluciones podrían ser vistas como una sola e ignorarían elementos de alta frecuencia al no poder generar respuestas a características pequeñas, mientras que la no linealidad no afectó las capas altas al no haber grupos de elementos de baja frecuencia que generasen un objeto completo.

Para efectos prácticos se realizó una pirámide gaussiana de la imagen de entrenamiento mostrado en la Figura 6, con el fin de mostrarle al lector que si bien es posible diferenciar estructuras en las imágenes de la izquierda, al reducir ésta, se perderán dichas diferencias denotables entre imágenes como se observa en la imagen de la derecha. La anterior observación es un referente a cuando no se crean filtros para remediar con respuestas pequeñas, como ocurre durante la unión de dos capas convolucionales al ser éstas resueltas por medio de una operación lineal.

En cuento a la remoción de las capas para pooling, en la Figura 4 lo primero que se denota es que dicha prueba no llegó hasta la última etapa, debido a que se demoró alrededor de 5 horas entrenando desde la primera época hasta la octava. Al cabo de 5 horas el equipo se sobrecalentó, por lo que se optó por finalizar el entrenamiento, esto es un indicio sobre la exigencia de memoria y procesamiento que maneja cada etapa al no dejar de lado ninguna respuesta de las neuronas en cada grupo. De otra manera, al comparar sólo las primeras etapas resultantes, el error es mucho mayor en la etapa 8 de la prueba (con un top1err de 0.76) al de la red creada (con un top1err de 0.48), cuya posible causa sea el overfitting entre etapas, ya que la idea del pooling es obviar ruidos que no generan una respuesta óptima entre capas.

Un ejemplo de lo anterior puede ser demostrado en algunas de las imágenes en la Figura 6, si enfatizamos en aquellas con bordes definidos y de baja frecuencia, se puede llegar a observar ruido normal en ellos. Dicho ruido normal será captado posiblemente por alguna neurona, pero al no ser consistente tendrá una respuesta baja, por lo que la respuesta máxima se presentará en la respuesta al borde y será aquella tenida en cuenta en la siguiente convolución.

Por último, la idea del escalamiento entre etapas de convolución resulta ser un factor fundamental para la obtención de características de la imagen, esto se notó al eliminar la penúltima etapa, en la cual se pasa de una matriz de respuestas de tamaño 5x5 con 500 características, a una matriz

de tamaño 1x1 con 25 características, la cual culmina en la etapa de clasificador softmax. En ella se observa que el algoritmo no arroja valores numéricos definidos, es decir, se obtuvieron algunos valores NaN (Not a Number) resultantes de divergencias en las operaciones.

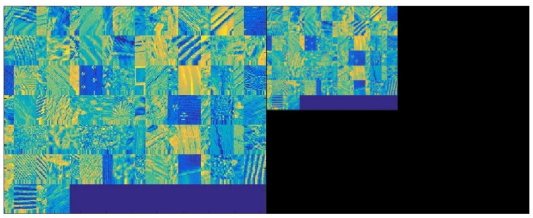

Figura 6. Muestra de texturas de entrenamiento.

Referencias

[1] S. Hijazi, R. Kumar, and C. Rowen, "Using convolutional neural networks for image recognition," 2015.

[2] R. E. L. Briega, "Redes neuronales convolucionales con tensorflow," 2016.