PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
IIC-3697 APRENDIZAJE PROFUNDO

TAREA 1: REDES NEURONALES CONVOLUCIONALES (CCNS)

Fecha de Entrega: 4 de Mayo

Objetivo

Estimad@s, en esta tarea tendrán la oportunidad de poner en práctica sus conocimientos sobre aprendizaje profundo (deep learning). En particular, podrán experimentar con las técnicas que discutimos en clases para implementar Redes Neuronales Convolucionales (Convolutional Neural Nets o CNNs). Adicionalmente, podrán experimentar con Google-Colab y observar el poder de las GPUs para acelerar el entrenamiento de redes de aprendizaje profundo.

1 Parte 1: YOLOv1 en Pytorch (40%).

Para comenzar, vamos a hacer una implementación de la primera versión de YOLO[1] (You Only Look Once) usando la librería Pytorch. YOLO es un modelo para detección de objetos, por lo tanto no sólo clasifica imágenes, sino que hace una segmentación y puede entregar múltiples detecciones para una misma imágen. YOLO, como su nombre sugiere, tiene la particularidad de que tiene un único pipeline donde hace todo el procesamiento.

Actividad 1

Investigue cómo es el output de YOLOv1 y explique cómo se traduce ese tensor a las bounding boxes y clasificaciones.

Actividad 2

Implemente una versión de YOLOv1 *from scratch*(sin entrenar) usando las herramientas que proporciona Pytorch. Para esto, escriba una clase llamada `YOLO_v1` que herede de la clase `torch.nn.Module` e implemente las capas que correspondan, en el orden que correspondan, utilizando elementos de `torch.nn`. No olviden agregar los leaky ReLU que correspondan.

Su clase, como mínimo debe implementar también un método de clase llamado `forward`, que es donde usted debe armar la arquitectura de la red cuando ésta hace una predicción para que entregue un resultado.

Al momento de implementar, verifique que su feature map tenga las dimensiones que correspondan en las capas que corresponda. Para esto les aconsejo observar en detalle la figura 1 y ver, por ejemplo, qué cosas hacen que la imagen disminuya de área.

Por último, entréguele a su modelo una imágen de 448×448 como input y verifique que la salida es un tensor de $7 \times 7 \times 30$

Actividad 3

Si quisieran que su modelo tenga más opciones de bounding boxes, ¿qué tendrían que cambiar? Si quisieran entrenar una versión de YOLO que clasifique entre 10 clases, ¿qué tendrían que cambiar?

Cree otra instancia de su clase, pero esta vez que el output esté pensado para 3 bounding boxes por celda y con 10 clases posibles para la detección. Si ya generalizó su clase de la Actividad 2 para estas variables, en esta actividad sólo necesita volver a instanciarla con los nuevos parámetros.

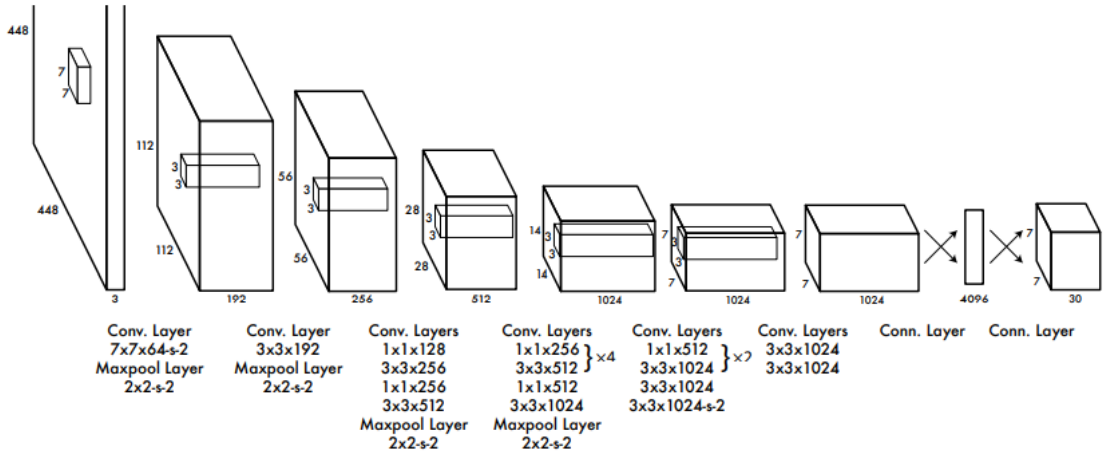

Figura 1: Diagrama de la arquitectura de YOLOv1. 24 capas convolucionales en total, 4 capas de Maxpooling y 2 capas fully conected al final.

Verifique que realizó los cambios correctamente ingresando como input la misma imágen de la actividad anterior y revise que el output tiene las dimensiones que corresponden.

2 Parte 2: Fine-tuning con un modelo pre-entrenado (60%).

En esta parte vamos a usar una red pre-entrenada, vamos a cambiarle el head y vamos a entrenar este último para una tarea distinta a la que fue entrenada la red original. En particular, tienen la opción de escoger entr AlexNet[2], VGG[3], ResNet50[4] e Inception[5], todas ya implementadas en torchvision.

Actividad 4

Escoja una de las redes mencionadas. Inicialice una instancia pre-entrenada y sustituya su última capa por la última capa de una YOLOv1 con una salida en una grilla de 5×5 celdas, con 2 clases y 2 bounding boxes por celda.

Actividad 5

Haga fine-tuning de su modelo sobre el Raccoon Detector Dataset[6]. Entrene su red usando la loss function del paper original de YOLO[1]. Haga un gráfico de el valor de la función de pérdida, true positives y false negatives con respecto a las épocas de entrenamiento. Considere las detecciones como "correctas" cuando el IOU (intersection over union) sea mayor a 0.5.

Actividad 6

Programe una función que tome como input la imagen, la predicción de su modelo y un umbral de confianza. Grafique las bounding boxes predichas que tengan una confianza mayor al umbral de input, junto con la clase que está prediciendo y el valor de confianza de la predicción.

3 Google Colaboratory

En la página web del curso podrán encontrar un documento con instrucciones para utilizar `Colaboratory`, un proyecto de Google orientado a diseminar la educación e investigación en aprendizaje de máquina, que nos permitirá utilizar GPUs para ejecutar las tareas del curso.

4 Consideraciones y Formato de Entrega

La tarea deberá ser entregada via SIDING en un cuestionario que se habilitará oportunamente. Se deberá desarrollar la tarea en un Jupyter Notebook con todas las celdas ejecutadas, es decir, no se debe borrar el resultado de las celdas antes de entregar. Si las celdas se encuentran vacías, se asumirá que la celda no fue ejecutada. Es importante que todas las actividades tengan respuestas explícitas, es decir, no basta con el output de una celda para responder.

5 Bibliografía

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. doi:10.48550/ARXIV.1506.02640

[2] Krizhevsky, A. (2014). One weird trick for parallelizing convolutional neural networks. doi:10.48550/ARXIV.1404.5997

[3] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. doi:10.48550/ARXIV.1409.1556

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. doi:10.48550/ARXIV.1512.03385

[5] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. doi:10.48550/ARXIV.1512.00567

[6] `https://github.com/datitran/raccoon_dataset`