Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

EL4106 - Inteligencia Computacional
Tarea 2: Redes Neuronales Convolucionales

Profesor: Pablo Estévez Valencia
Auxiliar: Ignacio Reyes Jainaga

Fecha de entrega: 15 de Octubre, 23:59.

Parte Teórica: Tamaños y formas de los parámetros y activaciones

La red VGGNet fue diseñada por Karen Simonyan y Andrew Zisserman en el año 2014. Su principal contribución fue mostrar que la profundidad de la red convolucional es un componente crítico para su desempeño. La arquitectura definitiva de la red (configuración D en la figura 1) consiste en 16 capas y usa una arquitectura muy homogénea, de convoluciones de 3x3 y pooling de 2x2 de principio a final. Los detalles respecto a *zero padding*, *strides*, etc. se encuentran descritos en el paper.

Demuestre que el modelo tiene 138 millones de parámetros, tal como se indica en la Tabla 2. Muestre la cantidad de parámetros capa a capa y el espacio requerido en memoria para almacenar los pesos de cada capa (usando números de punto flotante de 32 bits). No olvide considerar los biases en su análisis.

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figura 1: Descripción de las arquitecturas VGG. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Parte Práctica

1. Introducción

Esta tarea es para familiarizarse de forma práctica con la estructura de una red convolucional en un ejemplo real. Para esta tarea ustedes tendrán que resolver el problema de clasificación utilizando la base de datos CIFAR-10, la cual contiene 60000 imágenes de 32x32 a color, 50000 de entrenamiento y 10000 de prueba, 10 clases de imágenes con cada clase balanceada. La primera vez que se ejecuta el código se realiza la descarga del dataset.

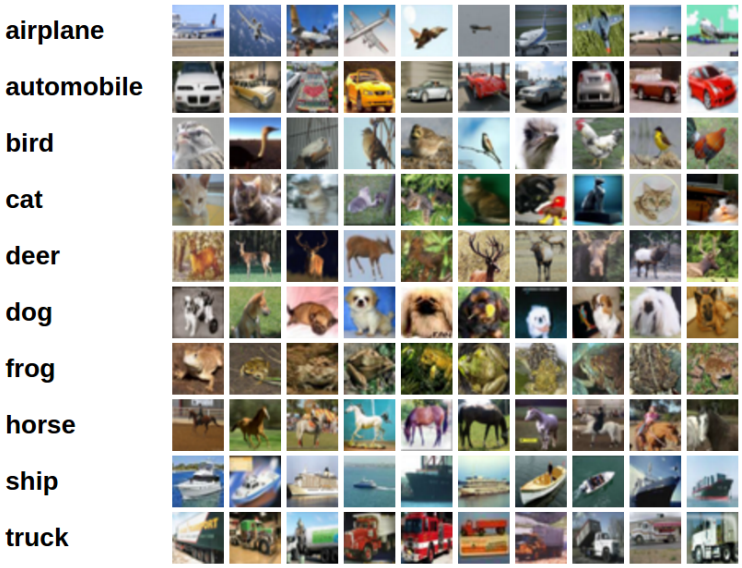

Figura 2: Clases e imágenes de ejemplo de CIFAR 10

El código de la tarea se encuentra disponible en U-Cursos. Es una implementación en Python de una red convolucional y un perceptrón multicapa. Utiliza las mismas dependencias de la Tarea 1, tales como Tensorflow, Matplotlib, Numpy, scikit-learn, etc. Para ejecutar el código se debe proceder de la misma forma que para la tarea 1. Puede utilizar Google Colab para aprovechar el cálculo en GPUs que ofrece.

2. Experimentos

Si va a utilizar Colab suba el archivo "cifar10.py" usando el primer bloque de código del Jupyter notebook. La primera vez que construya el objeto CIFAR10 puede que ocurra un error. En ese caso simplemente ejecute de nuevo dicho bloque.

2.1. Dropout

Ejecute el entrenamiento de la red convolucional dada (dos capas convolucionales, dos capas de pooling y dos capas fully-connected) utilizando 0.0 y 0.5 como probabilidad de *Dropout* (probabilidad de apagar la unidad). Elija un valor de la probabilidad de *Dropout*, justifique su elección a partir de los resultados y manténgalo fijo para las siguientes pruebas.

2.2. Impacto del número de capas

Modifique la red convolucional para que tenga solo una capa de convolución y de *pooling* (mantenga el número de capas *fully connected*) y entrénela. Luego, modifique la red para que tenga 3 capas de convolución y de *pooling* manteniendo el número de capas *fully connected* y entrénela. Compare los resultados obtenidos entre estos dos casos y la red con 2 capas convolucionales de la parte anterior, explicando las diferencias observadas.

2.3. Comparación con MLP

Entrene un perceptrón multicapa con una capa oculta de 100 neuronas. Compare los resultados obtenidos por las distintas redes convolucionales entrenadas y la MLP en términos de error de clasificación, velocidad en el entrenamiento y sobreajuste.

2.4. Data Augmentation

Usando la mejor red convolucional de los experimentos anteriores, active el *flag* de *data augmentation* en el objeto CIFAR10 del código y entrene. Compare los resultados obtenidos al usar esta técnica versus no hacerlo, considerando el desempeño conseguido luego del entrenamiento, épocas requeridas para entrenar y sobreajuste.

3. Parte de programación: ¡¿No hay fully-connected?!

El objetivo de la tercera parte de la tarea es construir una red convolucional que no tenga capas *fully-connected*. Para ello deberá modificar la red convolucional de la tarea, eliminando las capas *fully-connected* y reemplazándolas por capas convolucionales. Elija como punto de partida la arquitectura de tres capas convolucionales utilizada anteriormente.

Las capas *fully-connected* del modelo deberán ser reemplazadas por una capa convolucional con filtros de 1x1 que dé lugar a 10 *feature maps*. Esta nueva capa no debe tener una función de activación (sin ReLU ni sigmoides) ni debe estar seguida de una operación de *pooling*. Para construir el vector de salida de la red reduzca las dimensiones espaciales de los *feature maps* a la salida del modelo calculando el promedio de los valores de cada *feature map*. Luego, cada ejemplo a la entrada de la red dará lugar a un vector de salida de dimensión 10, el que puede ser utilizado junto a una función de costos típica como la entropía cruzada.

¿Puede el modelo construido resolver el problema de clasificación? ¿cómo se comparan los resultados obtenidos con el desempeño de la convnet que sí tiene capas *fully-connected*? Compare el número de parámetros en ambos modelos y el espacio que ocupan en memoria (cada parámetro pesa 4 bytes).

4. Observaciones

- Se recomienda que la extensión del informe no supere las 10 páginas (límite flexible).
- El código provisto está diseñado para visualizar el entrenamiento a través de tensorboard. Aproveche las figuras y funcionalidades que éste provee, le serán de gran ayuda.
- Considerando que deben ejecutar varios entrenamientos, se recomienda elegir directorios distintos para almacenar los registros de tensorboard. Esto se realiza simplemente escogiendo un nuevo nombre para la variable `SUMMARIES_DIR` del tercer bloque de los notebooks. Si se entrena un modelo guardando

los eventos en un directorio previamente utilizado, tensorboard mostrará una visualización errónea con ambas figuras superpuestas, de modo que deben siempre utilizarse directorios limpios.