Tarea 2

CNN

Profesor: Pablo Estévez
Auxiliar: Juan Urrutia

Instrucciones generales:

- **Importante:** Todos los plots realizados **deben** tener una etiqueta en el eje X y en el eje Y, además de un título. Las etiquetas y los títulos son libres a ser determinados por cada uno, sin embargo, deben estar relacionados con el contexto del gráfico.
- Por favor comentar los códigos. Si hay subsecciones se recomienda comentar los lugares donde comienza cada subsección.
- Utilicen nombres significativos para variables, funciones y clases. Esto les ayudará principalmente a ustedes cuando lean sus códigos en un futuro y también al resto de la gente con la que trabajen luego.

Instrucciones específicas:

- La tarea consta de tres secciones, una teórica, una práctica y una de programación. Usted deberá entregar un informe con las respuestas de las tres partes. En la parte práctica y de programación, usted debe ejecutar los experimentos pedidos, mostrar los resultados, realizar el análisis respectivo y las conclusiones respondiendo las preguntas presentes en el enunciado. El informe debe ser conciso, evitando extenderse más allá de las 10 páginas (límite de páginas flexible).
- El código entregado en el notebook de la parte práctica es la base para realizar los experimentos pero no está ordenado en base a las preguntas. Ustedes deben generar bloques del notebook (utilizando los bloques de tipo markdown para denotar las distintas preguntas) y copiar y pegar lo necesario en cada uno de estos bloques además de modificarlos en base a lo que se pregunta.
- El informe no debe contener introducción ni conclusión general, sólo mostrar resultados, análisis y conclusiones de cada pregunta, recordando agregar también los gráficos de entrenamiento.

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 CIFAR10, la cual contiene 60000 imágenes de 32x32 a color, 50000 de entrenamiento y 10000 de prueba, 10 clases de imágenes con cada clase balanceada.

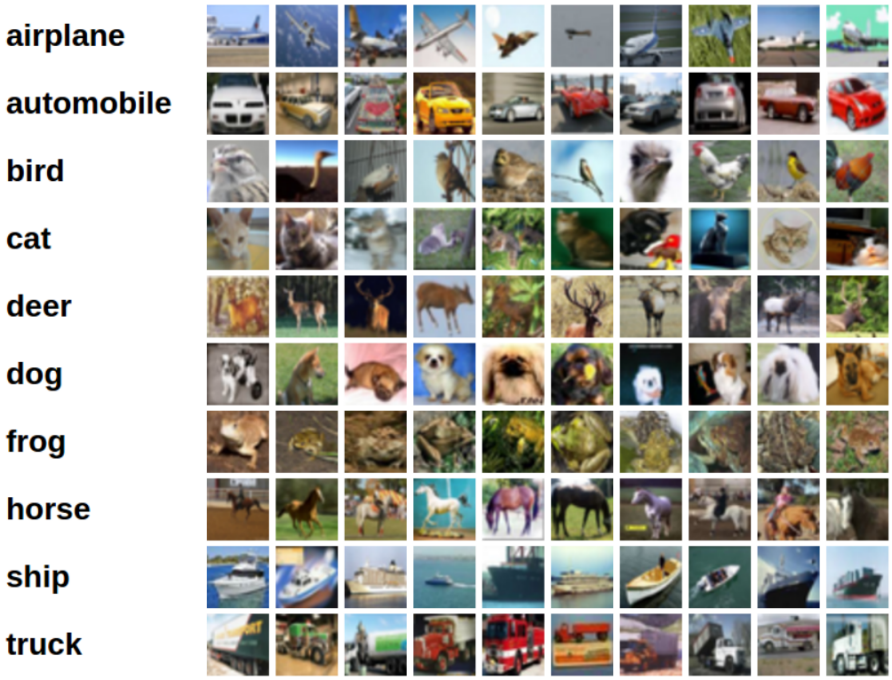

Figura 2: Clases e imágenes de ejemplo de CIFAR 10

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

2. Experimentos

En el código se entrega una clase 'CNNModel' que contiene una red convolucional con tres bloques. Cada bloque está compuesto de una capa convolucional, una activación ReLU y un max pooling. Cada bloque en el código está etiquetado (con comentarios) como bloque 1, 2 y 3. En los siguientes experimentos se les puede pedir utilizar sólo el primer bloque (tienen que programarlo en la clase 'SingleBlockCNNModel'), el primero y el segundo (tienen que programarlo en la clase 'TwoBlockCNNModel'), o los tres bloques (se entrega listo en la clase 'CNNModel'). En todos los casos la red MLP se mantiene igual, con una diferencia que se explica a continuación:

Al cambiar el número de capas convolucionales y de filtros, también cambia el tamaño del vector final que se le entrega a la primera capa lineal de la MLP que está luego de los bloques convolucionales. Ustedes deben encontrar el tamaño del vector que se le entrega a esta capa lineal para cada caso.

Para esto pueden comentar todas las capas de la MLP excepto la capa Flatten y entregarle a ese modelo una imagen de entrada. Con esto obtendrán a la salida un vector al cual ustedes pueden extraerle su tamaño y utilizar este tamaño como el número de neuronas de entrada de la primera capa lineal de la MLP. El resto de la MLP se mantiene igual.

2.1. Dropout

Ejecute el entrenamiento de la red convolucional dada utilizando los dos primeros bloques (clase 'TwoBlockCNNModel') 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. Utilice 30 épocas para su entrenamiento.

2.2. Impacto del número de capas

Entrene utilizando sólo el primer bloque (clase 'SingleBlockCNNModel'). Luego entrene la red 'CNNModel'. Compare los resultados obtenidos entre estos dos casos y la red con 2 bloques de la parte anterior, explicando las diferencias observadas. Utilice 30 épocas para su entrenamiento.

2.3. Comparación con MLP

Entrene un perceptrón multicapa con una capa oculta de 100 neuronas (clase 'MLPModel'). 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. En el entrenamiento de la red MLP aumente el número de épocas a 50.

2.4. Data Augmentation

Usando la mejor red convolucional de los experimentos anteriores, fije el argumento 'data_augmentation' de la función 'train_model' en True 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. Utilice 30 épocas para su entrenamiento.

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 10 filtros de 1x1 que resultarán en 10 feature maps. Esta nueva capa no debe tener una función de activación (sin ReLU ni sigmoides) ni debe estar seguida de un max 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 (ver función 'nn.AvgPool2d'). Luego, cada ejemplo a la entrada de la red dará lugar a un vector de salida de dimensión 10, el que puede ser utilizado como los logits a evaluar en 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). Utilice 30 épocas para su entrenamiento.