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

TAREA 1: REDES NEURONALES CONVOLUCIONALES (CCNS)

Fecha de Entrega: 30 de Abril

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: GoogLeNet Inception V1 (40%).

Para ilustrar la implementación de CNNs en Keras, estudiaremos en profundidad una de las estructuras más populares del año 2014: GoogLeNet [1]. Existen diferentes versiones de esta red pero nosotros replicaremos en Keras la vesión 1. En los anexos 5.1 y 5.2 pueden ver la estructura de esta red.

Para comenzar implementaremos la primera etapa de la red, según pueden visualizar en la figura 1. Respecto a la versión oficial de la red, por simplicidad en nuestra implementación omitiremos las capas de Local Response Normalization y las ramas de clasificación auxiliar.

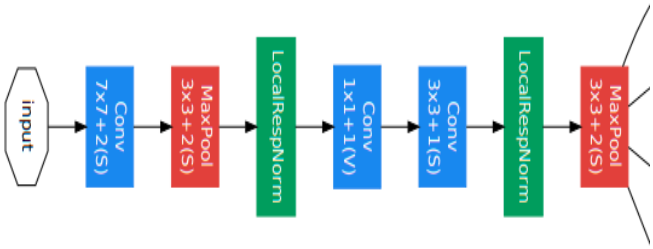

Figura 1: Primera etapa de GoogLeNet.

Descripción de las capas de la figura 1, para más detalle ver tabla en anexo 5.2:

1. Input: Tamaño (224,224,3).
2. Convolución: 64 filtros con ventana de 7x7, stride de 2 en dirección horizontal y vertical.
3. MaxPool: Ventana de 3x3, stride 2.
4. Convolución: Ventana de 1x1 con stride 1.
5. Convolución: Ventana de 3x3 con stride 5.
6. Max-Pool: Ventana de 3x3 con stride 2.

Si bien en clases se vió la forma de crear un modelo usando la función `Sequential()`, en esta tarea construiremos el modelo desde una perspectiva diferente, ya que no es un modelo secuencial en todas sus partes. Comenzamos definiendo la capa input que definirá la forma del tensor de entrada de la red. Corresponde a una imagen de 244x244 pixeles y 3 canales por pixel con representación RGB.

```
Inputs = Input(shape=(224,224,3))
```

Una vez definido el formato de input, debemos aplicar una convolución de 64 filtros de 7x7, con un stride de 2 pixeles en la dirección horizontal y 2 pixeles en la dirección vertical. Para esto usamos la función de Keras `Conv2D`, según la siguiente notación:

```
x = Conv2D(64, (7, 7), strides=(2,2), padding='same', activation="relu",
    use_bias=False,
name='Conv2d_1a_7x7_conv')(Inputs)
```

El primer parámetro indica la cantidad de filtros de la convolución, el segundo indica el tamaño de la ventana de convolución (7x7), el tercero indica los strides de 2 pixeles en cada borde (horizontal y vertical). El parámetro (`padding='same'`) agrega filas y columnas con ceros para que pueda coincidir el filtro convolucional con el tamaño de la imágen. Por otra parte, el parámetro (`activation='relu'`) agrega una capa de activación del tipo Relu que será la función de activación aplicada en toda la red. Agregamos un nombre representativo ya que nos servirá después cuando mostremos el resumen del modelo. El comando (`Inputs`) al final de la instrucción, indica sobre qué matriz (tensor) se aplicará la convolución. El resultado de la convolución quedará guardado en `x`. Es decir, `x` será el tensor que resulte de aplicar la convolución al input.

Como muestra la arquitectura luego debemos aplicar un operador de `Max-Pooling`. En este caso, el operador actúa sobre una vecindad de 3x3 utilizando un *stride* de 2 posiciones en las direcciones vertical y horizontal. Para esto usamos la función de Keras `MaxPooling2D`, según la siguiente notación:

```
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same', name='MaxPool_2a_3x3')(
    x)
```

Para visualizar el tamaño del tensor que entrega cada operación, podemos usar el comando `x.shape`.

Una vez que la descripción del modelo está lista, es necesario instanciarlo. Éste se intancia con la función `Model()`, donde el primer parámetro corresponde al input de la red y el segundo al output de ésta, que contiene todos los filtros por los cuales pasará el input. La función `summary()` expande la arquitectura de la red, permitiendo visualizar sus filtros, dimensiones de salida y otros datos relevantes.

```
model = Model(Inputs, x)
model.summary()
```

Uniendo todo lo anterior y agregando la definición de las librería necesarias, obtenemos el siguiente código:

```
#Definicion de librerias con las funciones que seran utilizadas por Keras.
import keras
from keras.models import Model
from keras.layers import Input, Conv2D, MaxPooling2D, Dense, Activation, Dropout
    , Flatten,
concatenate, AveragePooling2D

#Definicion de input y secuencia de capas de la red
Inputs = Input(shape=(224,224,3))

x=Conv2D(64, (7, 7), strides=(2,2), padding='same', activation="relu", use_bias=
    False,
name='Conv2d_1a_7x7_conv')(Inputs)
x=MaxPooling2D((3, 3), strides=(2, 2), padding='same', name='MaxPool_2a_3x3')(x)
```

```
#Instanciar el modelo e imprimir su resumen
model = Model([Inputs], x)
model.summary()
```

El archivo `GoogLeNetPrimerasCapas.py`, disponible en el sitio web del curso, contiene el código anterior.

Actividad 1

Ejecute el código en `GoogLeNetPrimerasCapas.py` y verifique que las dimensiones de salida de la capa del primer MaxPooling corresponden a las indicadas en la tabla del anexo 5.2.

Actividad 2

A continuación implemente las otras capas de la primera parte de la red, las cuales corresponden a:

1. Convolución de 64 filtros, con ventana de 1x1 y un stride de 1. Sin Bias.
2. Convolución de 192 filtros, con ventana de 3x3, stride de 1. Sin Bias
3. MaxPooling de ventana de 3x3 y stride de 2.

Estos detalles se pueden visualizar en la tabla del anexo 5.2.

En su informe de laboratorio reporte el código generado. Adicionalmente, utilice la función `shape` para verificar que la salida del segundo `MaxPooling` corresponde al que se indica en la tabla.

La red incorpora convoluciones de 1x1 seguidas de una convolución de ventanas más grandes. Investigue y explique para qué sirve esta convolución de 1x1.

Actividad 3

GoogleNet incorpora lo que se denomina módulos de incepción (inception), según muestra la figura 2.

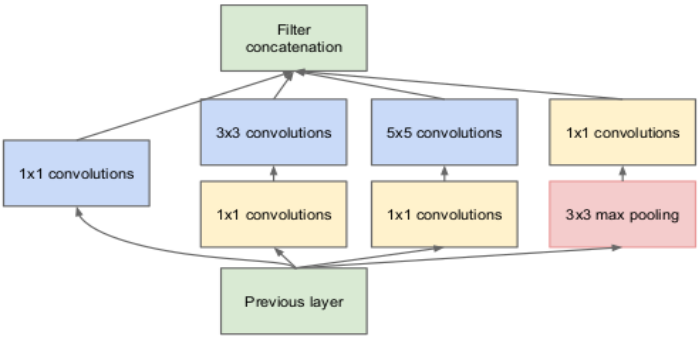

Figura 2: Modulo de incepción (inception).

Investigue e indique el funcionamiento y utilidad de estos módulos, específicamente respecto al uso de filtros con diferentes tamaños de convolución que son concatenados. ¿Qué condición deben cumplir los tensores que se concatenan para poder ser concatenados?

Actividad 4

Ahora modelaremos un módulo de incepción con Keras. A continuación se muestra la modelación de 2 ramas del primer módulo y su concatenación.

```
branch_0 = Conv2D(64, (1, 1), strides=(1,1), activation="relu", padding='same',
    use_bias=False,
name='Mixed_3b_Branch_0_a_1x1')(x)
```

```
branch_2 = Conv2D(16, (1, 1), strides=(1,1), padding='same', activation='relu',
    use_bias=False,
name='Mixed_3b_Branch_2_a_1x1'@)(x)
 branch_2 = Conv2D(@32, (5, 5), strides=(1,1), padding='same', activation='relu'
     , use_bias=False,
name='Mixed_3b_Branch_2_b_5x5'@)(branch_2)

x = concatenate([branch_0,branch_2],
        axis=3,
        name='Mixed_3b_Concatenated')
```

Todas las convoluciones dentro de un módulo de incepción tienen stride 1 y no utilizan bias. La cantidad de filtros que genera cada convolución varía, estas cantidades están descritas en la tabla del anexo 5.2. Desarrolle una función genérica para crear los módulos de inception. Esta función debe recibir como parámetros un tensor de una capa anterior y las cantidades de filtros que aplicarán cada convolución. Luego aplique las operaciones necesarias sobre este tensor y las concatenaciones respectivas según lo requiera el módulo de incepción modelado.

Actividad 5

Basándose en la arquitectura de la red, modele el resto de las capas, concatenando los diferentes módulos de incepción. Recuerde que cada módulo de incepción utiliza diferente cantidad de filtros para sus convoluciones. Recuerde también incluir las capas de MaxPooling que aparecen entre algunos módulos de incepción. Debe llegar hasta el último módulo de incepción previo al `AveragePool`. Refleje el código en su informe y revise si la dimensión del output de este último modulo corresponde al indicado en la tabla del anexo 5.2.

Actividad 6

Ha sido una labor ardua pero estamos cerca de la gloria y completar la implementación de GoogLeNet Inception V1. Resta la definición de las últimos operadores: average pool, capa de conexión densa (multilayer perceptron o MLP) y activación softmax. Las funciones `AveragePooling2D`, `Dense`, `Flatten` y `Activation` en la librería de Keras nos ayudarán en esto.

Instancie el modelo completo y utilice la función `model.summary()` para visualizar la estructura de la red. En su informe de laboratorio reporte el código generado. Realice un análisis general del número de capas y parámetros de la red final. ¿Cuál es la dimensión de salida de la red? ¿Cuál es el objetivo de usar una función de activación del tipo `softmax`?.

2 Parte 2: Reconocimiento de Visual (60%).

Una de las aplicaciones donde los modelos profundos convolucionales han tenido mayor éxito es el reconocimiento visual. En esta parte de la tarea explorarán algunas arquitecturas que han surgido en este ámbito. Específicamente, tendrán la oportunidad de experimentar con 3 populares arquitecturas convolucionales: GoogLeNet [1], AlexNet [2] y ResNet-50 [3]. Estos modelos se encuentran ya implementados en el módulo `applications` de la librería Keras.

Actividad 7

Investigue sobre las redes AlexNet y ResNet–50, luego realice un breve cuadro que indique las principales diferencias entre estas redes y GoogLeNet (Inception V1). En particular, compare profundidad, número de parámetros y el rendimiento que obtienen estas redes en el set de datos ImageNet (ILSVRC-2010). En su informe, comente **brevemente** sus principales observaciones.

2.1 CIFAR-10

Para esta parte de la tarea deberán usar el set de datos CIFAR [4]. Específicamente, el conjunto CIFAR-10, que contiene 60.000 imágenes RGB de 32 x 32 pixeles pertenecientes a 10 clases. Cada clase tiene 6000 imágenes. En el sitio web del curso se publicará un enlace para descargar este dataset. También es posible importar este dataset directamente desde Keras usando funciones de su librería.

Seleccione alguna de las redes que hemos estado discutiendo, vale decir AlexNet, GoogLeNet o Resnet-50, y entrénela sobre el conjunto de datos CIFAR-10. Para el entrenamiento de la red, defina y comente alguna estrategia de validación a fin de ajustar hiperparámetros, como el número de épocas a entrenar, optimizador y tasa de aprendizaje.

Actividad 8

Detalle el conjunto de datos utilizados. Inserte un par de imágenes de ejemplo con sus respectivas etiquetas. Entrene la red utilizando diferentes hiperparámetros. Comente acerca de la estrategia de validación, tiempo de entrenamiento, ajuste de parámetros, tamaño de mini-batch .

Actividad 9

Usando la información de entrenamiento, realice gráficos que muestren:

- Evolución de la función de pérdida respecto de las épocas de entrenamiento.
- Evolución de la exactitud de clasificación sobre el set de entrenamiento y validación.
- Analice y comente los gráficos anteriores.

Actividad 10

Pruebe el modelo resultante en el set de test.

- Indique la exactitud promedio. Compare las exactitudes en el set de test, validación y entrenamiento. Comente.
- Seleccione algunas imágenes en que su modelo entrega una clasificación errónea, muéstrelas y discuta posibles razones.

2.2 Comparación según set de datos

Realice los ajustes necesarios y vuelva a utilizar la red anterior pero esta vez sobre el set de datos CIFAR-100.

Actividad 11

Detalle la estructura del dataset. Vuelva a realizar el entrenamiento. Refleje en su informe los cambios/ajustes que realizó a su código.

Usando la información de entrenamiento, realice gráficos que muestren:

- Evolución de la función de pérdida respecto de las épocas de entrenamiento.
- Evolución de la exactitud de clasificación sobre el set de entrenamiento y validación.
- Analice y comente los gráficos anteriores.

Actividad 12

Pruebe el modelo resultante en el set de test.

- Indique la exactitud promedio. Compare las exactitudes en el set de test, validación y entrenamiento. Comente.
- ¿Cuál es la clase con el mejor y cuál tiene el peor rendimiento?. Seleccione alguna imágenes de las clases con mejor y peor clasificación, inclúyalas en su informe y comente.

Actividad 13

Compare los resultados obtenidos con ambos set de datos entre redes entrenadas con hiperparámetros iguales.

- Compare las exactitudes promedio para cada caso (utilizando CIFAR 10 y CIFAR 100) en el set de test. Comente las diferencias.

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 Anexos

5.1 Arquitectura Visual GoogLeNet Inception V1

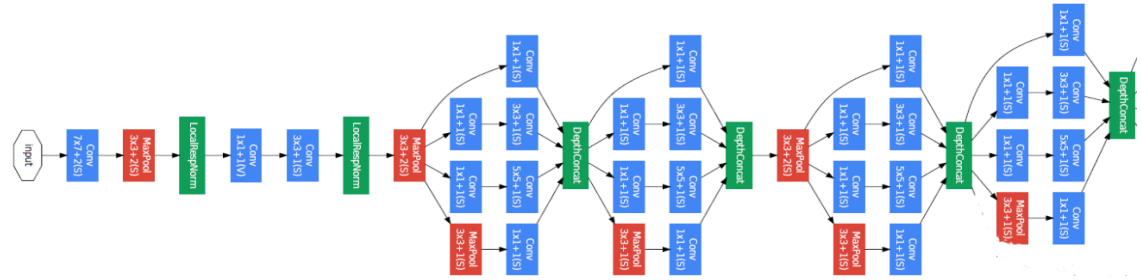

Figura 3: Diagrama de la arquitectura de la red convolucional GoogLeNet. Primera Mitad.

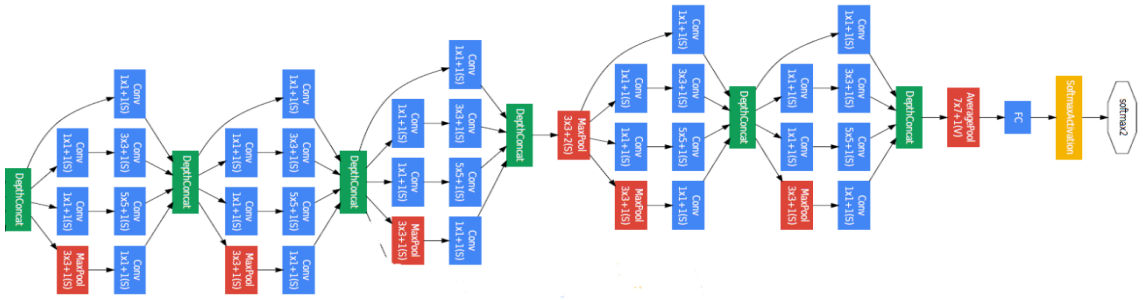

Figura 4: Diagrama de la arquitectura de la red convolucional GoogLeNet. Segunda Mitad.

5.2 Detalle de la Arquitectura de GoogLeNet Inception V1

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj
convolution	7×7/2	112×112×64	1						
max pool	3×3/2	56×56×64	0						
convolution	3×3/1	56×56×192	2		64	192			
max pool	3×3/2	28×28×192	0						
inception (3a)		28×28×256	2	64	96	128	16	32	32
inception (3b)		28×28×480	2	128	128	192	32	96	64
max pool	3×3/2	14×14×480	0						
inception (4a)		14×14×512	2	192	96	208	16	48	64
inception (4b)		14×14×512	2	160	112	224	24	64	64
inception (4c)		14×14×512	2	128	128	256	24	64	64
inception (4d)		14×14×528	2	112	144	288	32	64	64
inception (4e)		14×14×832	2	256	160	320	32	128	128
max pool	3×3/2	7×7×832	0						
inception (5a)		7×7×832	2	256	160	320	32	128	128
inception (5b)		7×7×1024	2	384	192	384	48	128	128
avg pool	7×7/1	1×1×1024	0						
dropout (40%)		1×1×1024	0						
linear		1×1×1000	1						
softmax		1×1×1000	0						

Figura 5: Cada fila corresponde a filtros de la red. Las columnas con NxN indican la cantidad de filtros que aplica la convolución respectiva. La columna *reduce* corresponde a la convolución de 1x1 que aplica antes de la siguiente convolución indicada por la columna siguiente. El número indica la cantidad de filtros que aplica esa convolución. La columna *pool proj* indica la cantidad de filtros que aplica la convolución de 1x1 sobre el MaxPooling dentro de un módulo de inception.

6 Bibliografía

[1] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. Going deeper with convolutions. In CVPR 2015.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR 2015.

[4] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.