Deep Learning: Transfer Learning № 6

Daniela Pinto Veizaga, dpintove@itam.mx
Diego Villa Lizárraga, dvillali@itam.mx

1/03/2020

Introducción

En esta sección revisaremos el concepto de **aprendizaje por transferencia** como método de aprendizaje automático en el que un modelo desarrollado para una tarea se reutiliza como punto de partida para un modelo en una segunda tarea.

Pregunta 1

Investiga de qué manera se almacenan las capas (layers) de los modelos de tf.keras.

De acuerdo con la documentación Tensor Flow Keras las capas de los modelos de keras tienen los siguientes métodos:

- get_weights
- set_weights
- get_config

  Con estos métodos accedemos a los valores numéricos de los parámetros (pesos y sesgos, entre otros).

Además, sabemos que se puede utilizar *model.save(filepath)* para guardar un modelo Keras en un archivo único HDF5 que contendra:

- La arquitectura de un modelo (incluyendo detalle de cada capa en formato de lista, propiedad model.layers), permitiendo recrear el modelo.
- Los pesos del modelo.
- La configuración del entrenamiento (funcion de perdida, optimizador).
- El estado del optimizador, permitiendo retomar el entrenamiento exactamente donde lo dejaste.

Pregunta 2

¿Cómo imprimirías información sobre el número de capas que tiene un modelo?

La información de las capas se guarda en *model.layers*, contando el número de elementos en esta lista podemos obtener el total de capas del modelo. Además, se puede obtener el tipo y output shape de cada capa con *model.output* como se muestra en la imagen de abajo.

```
outputs = [layer.output for layer in model.layers]
```

```
print("Total de capas del Modelo: ",len(outputs))
```

```
Total de capas del Modelo:  6
```

```
for i in range(len(model.layers)):
    layer = model.layers[i]
    print(i, layer.name, layer.output.shape)
```

```
0 vgg16 (None, 4, 4, 512)
1 conv2d (None, 2, 2, 128)
2 max_pooling2d (None, 1, 1, 128)
3 flatten (None, 128)
4 dense (None, 128)
5 dense_1 (None, 1)
```

Finalmente, la forma convencional empleada hasta el momento para visualizar el contenido los modelos es *model.summary()*.

Pregunta 3

Para el caso particular de transferencia de conocimiento, en el cual usas un modelo preentrenado como segmento de un modelo más grande, ¿cambia la forma en que se almacenan las capas?

Sí, para el caso particular de transferencia de conocimiento, en el cual usas un modelo preentrenado como segmento de un modelo más grande, el modelo pre-entrenado se guarda como una sola capa de tipo "Model" del nuevo modelo.

En esta capa se concentran todos los par ámetros de todas las capas de la red preentrenada; para acceder a ellas, se procede de la misma manera que a cualquier capa normal. La diferencia radica en la forma en la que presenta la información de la capa: en formato anidado.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
conv2d (Conv2D)              (None, 2, 2, 128)         589952
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 128)               16512
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 15,321,281
Trainable params: 606,593
Non-trainable params: 14,714,688
_________________________________________________________________
```

Pregunta 4

Explica el procedimiento para visualizar los kernels entrenados de alguna capa intermedia (puedes ayudarte de líneas de código, pero la importancia de esta pregunta radica en que puedas explicar qué hace cada línea de código).

Listing 1: Get all kernels from the 5-th hidden layer

```
1
2  W, b = vgg_model.layers[5].get_weights()
3  print("Weights:", W.shape)
4  print("bias:", b.shape)
5
6  # Visualize a few examples
7  ind_plot = 1
8  plt.figure(figsize=(15, 8))
9  for ind_filter in range(10):
10   for ind_chann in range(20):
11     img = W[:, :, ind_chann, ind_filter].copy()
12     img -= img.min()
13     img /= img.max()
14     plt.subplot(10, 20, ind_plot)
15     ind_plot += 1
16     plt.imshow(img, cmap='gray')
17     plt.axis('off')
18 plt.tight_layout()
19 plt.show()
```

A continuación, se explica las líneas de código del *Listing 1*:

1. Obtenemos las capas con la propiedad *model.layers* y los pesos y los sesgos con get_weights.
2. Filtramos por alguna de las capas intermedias (capa 5 en este caso).
3. Normalizamos los pesos para que esten entre 0 y 1 para que podamos visualizarlos.
4. Usamos matplotlib y graficamos cada peso como una nueva fila de sub-graficas.
5. Se genera una figura con 10 filas de 20 imágenes, una fila para cada peso y una columna para cada canal.

Pregunta 5

De manera similar, explica el procedimiento para visualizar las salidas de las capas intermedias (feature map o image response).

Listing 2: Feature map or image response

```
1  # Predict
2  feat_map = model2.predict(sample_x)
3  print(feat_map.shape)
4
5  # Show a few feature maps (kernel responses)
6  a_sample = feat_map[0]
7  plt.figure(figsize=(10, 10))
8  for ind in range(100):
9    img = a_sample[:, :, ind].copy()
10   img -= img.min()
11   img /= img.max()
12   plt.subplot(10, 10, ind+1)
13   plt.imshow(img, cmap='gray')
14   plt.axis('off')
15 plt.tight_layout()
16 plt.show()
```

En referencia al código anterior, se explica línea por línea cómo visualizar las salidas de las capas intermedias:

1. *Líneas 1-3*: Definimos que la predicción se realizará usando la red previamente entrenada.

2. *Líneas 6 y 7*: Con una imagen (0) se extrae los feature maps y se define el número de gráficas y cuadrículas.

3. *Líneas 8-14*: A través de un loop de tipo **for**, se extrae la acción de los primeros 100 filtros, mediante la ultima entrada del arreglo que tiene cada imagen. Se toman todas las posiciones de los $64x64$ pixeles que tiene la imagen.

   Luego, con el fin de normalizar y centrar cada pixel, se resta el mínimo y divide entre el máximo.

   Finalmente, los pixeles mapean a una escala de grises e imprimen.

4. *Línea 15 y 16:* Imprimen la cuadrícula.