Aprendizaje Automático Profundo (Deep Learning)

Dr. Facundo Quiroga - Dr. Franco Ronchetti

Visualización de Filtros

Visualización de Filtros

```
from keras.applications.vgg16 import VGG16
model = VGG16()
print(model.summary())
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)
```

```
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
```

Acceso a los filtros. Verificación de su tamaño

```
or layer in model.layers:
  if 'conv' not in layer.name:
    continue
  filters, biases = layer.get_weights()
  print(layer.name,filters.shape)
```

- 64 filtros de 3x3x3
- 64 filtros de 3x3x64
- 128 filtros de 3x3x64
- 128 filtros de 3x3x128
- …
- 512 filtros de 3x3x512

```
Filter (h,w,cin,cout)
block1_conv1 (3, 3, 3, 64)
block1_conv2 (3, 3, 64, 64)
block2_conv1 (3, 3, 64, 128)
block2_conv2 (3, 3, 128, 128)
block3_conv1 (3, 3, 128, 256)
block3_conv2 (3, 3, 256, 256)
block3_conv3 (3, 3, 256, 256)
```

```
Filter (h,w,cin,cout)
block4_conv1 (3, 3, 256, 512)
block4_conv2 (3, 3, 512, 512)
block4_conv3 (3, 3, 512, 512)
block5_conv1 (3, 3, 512, 512)
block5_conv2 (3, 3, 512, 512)
block5_conv3 (3, 3, 512, 512)
```

Selección de la capa en base al nombre

```
def layer_by_name(model,layer_name):
 for layer in model.layers:
   if layer.name==layer_name:
     return layer
 raise ValueError(f"Invalid layer name {layer_name}")

#Elegir capa convolucional y nro de filtro
layer_name="block1_conv2"
filter_index=0
#dibujar filtro
layer = layer_by_name(model,layer_name)
filters, biases = layer.get_weights()
plot_conv_weight(layer.name,filters[:,:,:,filter_index])
```

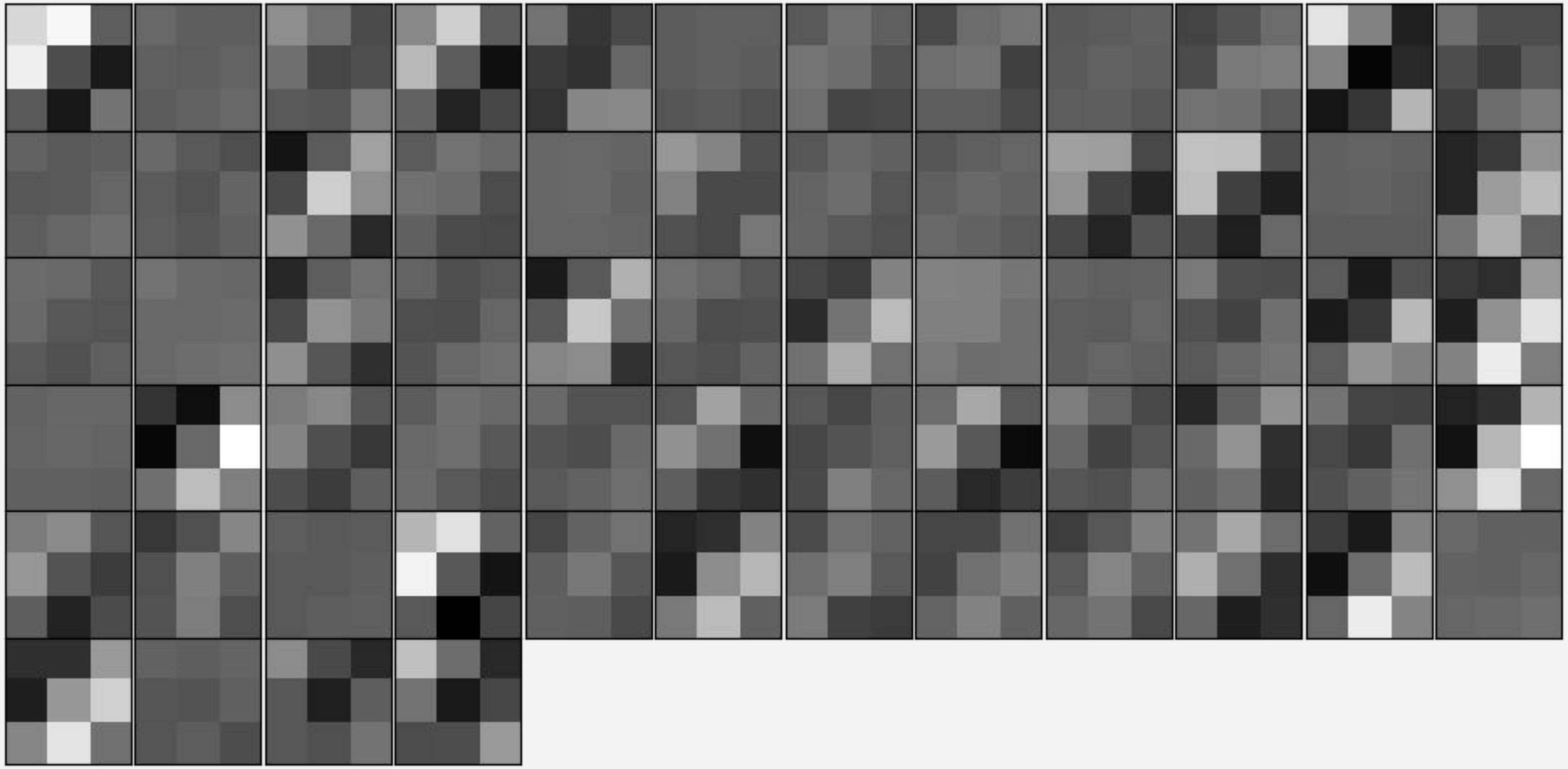
Visualización de Filtros

```
def plot_conv_weight(layer_name,filters,cols=32):
 h,w,c=filters.shape
 rows=( c // cols ) + int((c % cols)>0)
 gs = gridspec.GridSpec(rows, cols,wspace=0.1,hspace=0.1)
 # maximo y minimo para dibujar todos en la misma escala
 mi,ma=filters.min(),filters.max()
 for i in range(c):
     ax = plt.subplot(gs[i])
     ax.imshow(filters[:,:,i],cmap="gray",vmin=mi,vmax=ma)
     ax.set_xticks([])
     ax.set_yticks([])
 # poner en blanco los ax que sobran
 for i in range(c,cols*rows):
   ax = plt.subplot(gs[i])
   ax.axis("off")
```

Ejemplo Completo

Verificación de aprendizaje correcto

- Capa **`block1_conv2`**
- block1_conv2 (3, 3, 64, 64)
- Primer filtro (**`filter_index=0`**)
- shape = (3,3,64)
- 64 cuadrados de 3x3
- c/u: pesos sobre el Feature Map anterior



Importante: No son todos 0 (filtros muertos)