Deep Learning con Pytorch

Juan Pablo Morales
@juanpamf

Espacio de modelos en Deep Learning

Espacio de modelos

```
[ ] class DummyNet(nn.Module):
      def __init__(self):
        super(DummyNet, self).__init__()
        self.conv1 = nn.Conv2d(3,9,3)
        self.relu = nn.ReLU(True)
        self.fc = nn.Linear(225, 10)

      def forward(self, x):
        x = self.relu(self.conv1(x))
        x = x.view(-1, 225)
        x = self.fc(x)
        return x
```

```
[ ] net1 = DummyNet()
    net2 = DummyNet()
```

```
[ ] net1.named_parameters() == net2.named_parameters()
```

```
False
```

Arquitectura: Capas

Espacio de modelos

```
[ ] x = torch.randn((1,3,7,7))
    (net1(x),net2(x))
```

```
(tensor([[-6.7621e-02, -3.6932e-01, -1.2229e-02, -1.6261e-01,  2.0797e-01,
          -1.2283e-02,  2.2240e-01,  4.1544e-01, -1.7868e-04,  2.5359e-01]],
        grad_fn=<AddmmBackward>),
 tensor([[ 0.0623, -0.0934,  0.0330,  0.3020,  0.3081,  0.2868, -0.1110, -0.1643,
           0.4227, -0.0915]], grad_fn=<AddmmBackward>))
```

```
[ ] from operator import mul
    import functools

    n = 0
    for t in list(net1.parameters()):
      n += functools.reduce(mul,t.size(),1)
    n
```

```
2512
```

$$H = \{f_p, p \in \mathbb{R}^{2512}\}$$

Espacio de modelos

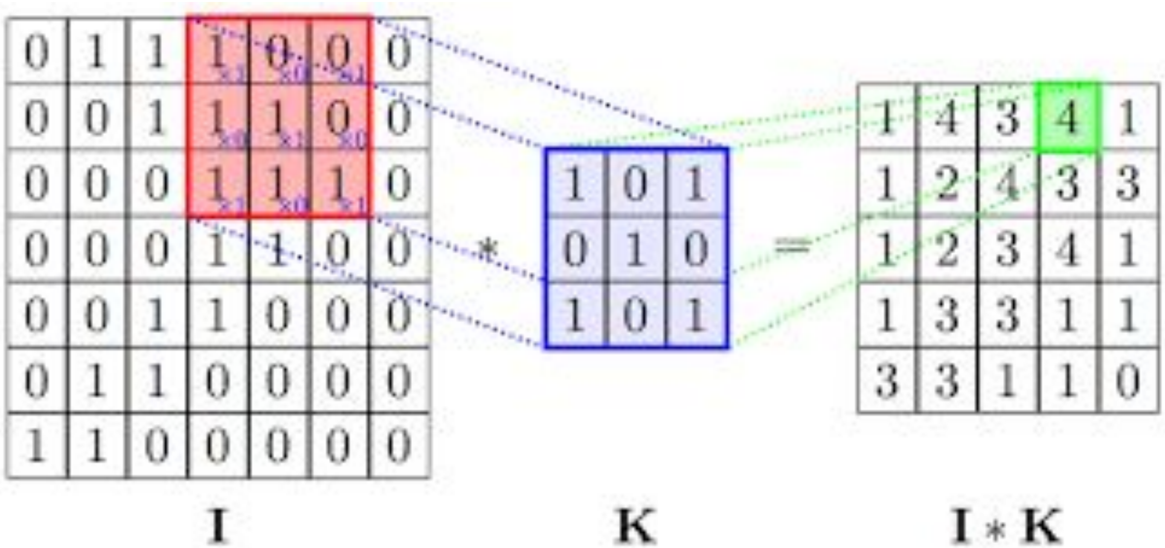
- Dada una arquitectura, para cada valor de los parámetros obtenemos una red neuronal distinta.
- Por ende una arquitectura define un espacio de modelos.
- Buscamos los parámetros óptimos

Espacio de modelos y capacidad

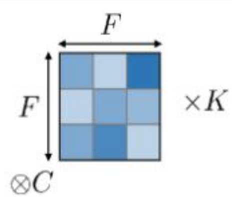
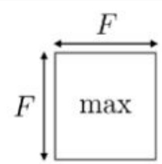
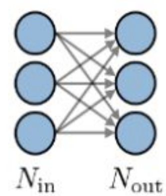
- Mientras más grande el espacio de modelos decimos que tiene más capacidad
- Mientras más capacidad más potencial de aprender, pero más difícil de enseñar
- Red profunda pequeña = 1M parámetros

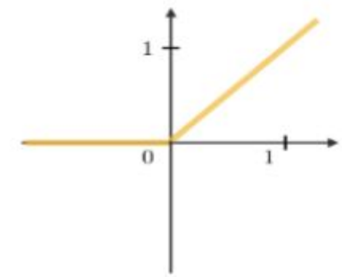
Capas Convolucionales

- Conectadas localmente
- Comparten parámetros
- ¡Muy eficientes sin tener un costo alto!



Arquitectura convolucional

	CONV	POOL	FC
Illustration			
Input size	$I \times I \times C$	$I \times I \times C$	N_{in}
Output size	$O \times O \times K$	$O \times O \times C$	N_{out}
Number of parameters	$(F \times F \times C + 1) \cdot K$	0	$(N_{\text{in}} + 1) \times N_{\text{out}}$
Remarks	- One bias parameter per filter - In most cases, $S < F$ - A common choice for K is $2C$	- Pooling operation done channel-wise - In most cases, $S = F$	- Input is flattened - One bias parameter per neuron - The number of FC neurons is free of structural constraints

ReLU
$g(z) = \max(0,z)$

Non-linearity complexities biologically interpretable

- Arquitectura clásica:

(Conv - ReLU - Pool)* - FC

Fórmula para convoluciones

- La cantidad de canales de entrada C
- El tamaña del filtro/kernel F
- La cantidad de filtros K
- El padding P
- El stride S

$$C \times I \times I \rightarrow K \times O \times O$$

$$O = \frac{I - F + 2P}{S} + 1$$