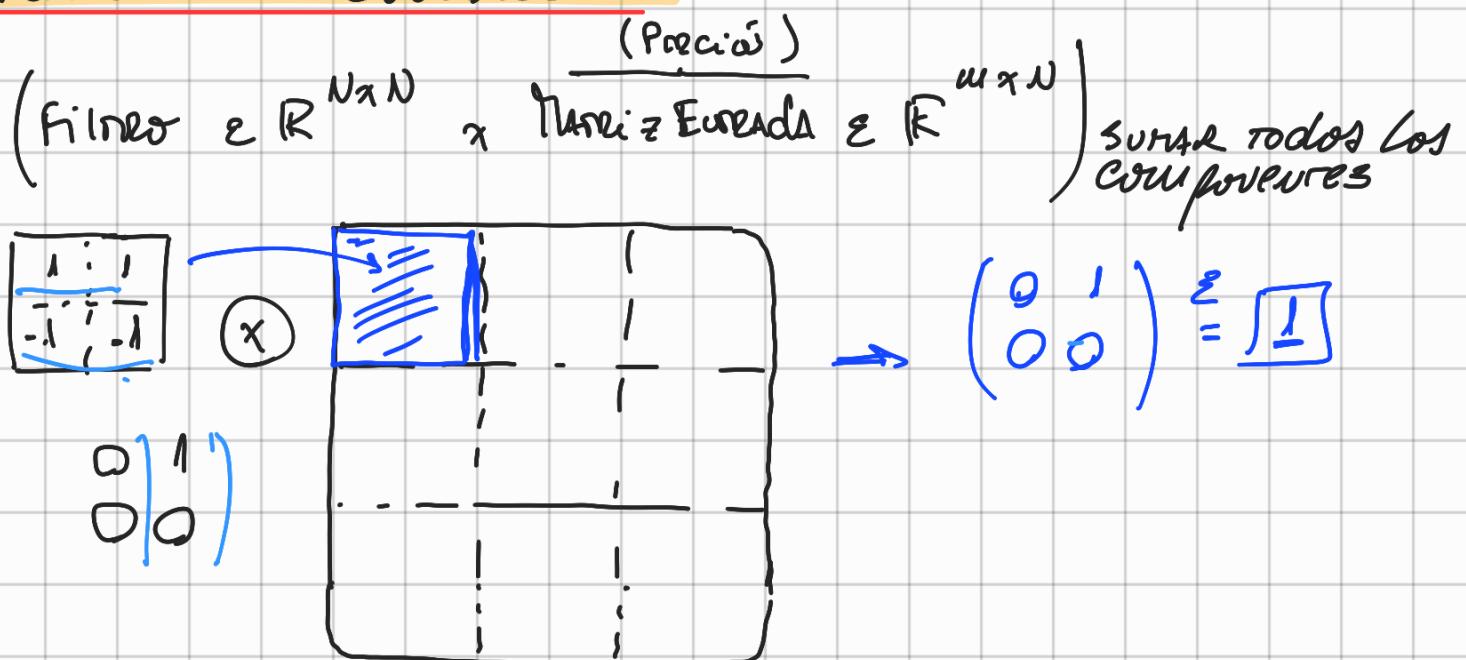


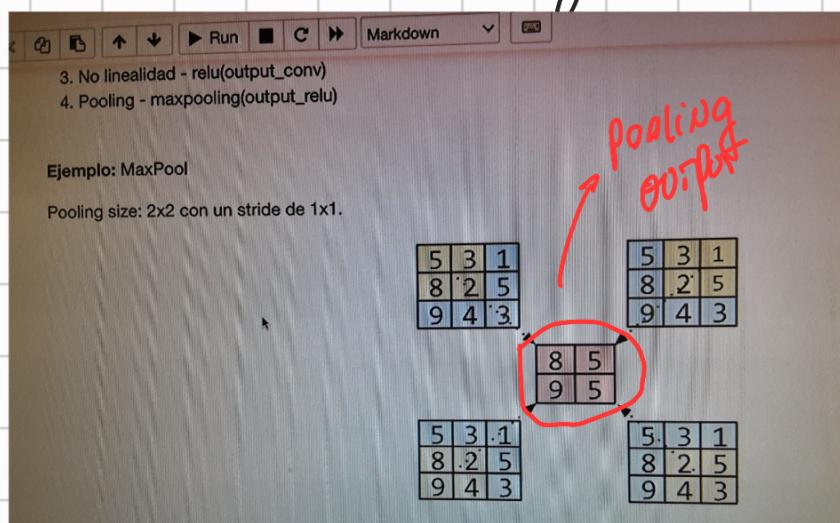
→ CNN's

OPERACIÓN de Convolución



→ Pooling ⇒ Recutar una Región mediante un Valor Significativo, para reducir el tamaño de las imágenes. (Max, Avg). SE Aplica luego de la convolución. Por lo general, el orden es

- 1) Imagen de Entrada (x)
- 2) Convolución - $\text{conv}(x)$
- 3) Nonlinearidad - $\text{ReLU}(\text{output}(\text{conv}(x)))$
- 4) Pooling ~ MaxPooling ($\text{output}(\text{ReLU})$)

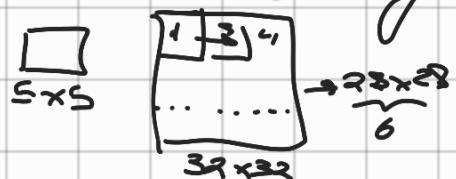


→ Ejemplo de Max Pooling.

• Padding → poneceros, le da mas espacio p/ moverse al filtro y ∴ mas buscada.

• Stride → "cuanto muesca un filtro convolucional. los 'pasos'. Cuanto mayor sea, mas se reduce la imagen capa a capa, por lo que se suele combinar con Padding.

Ejemplo de LeNet Con PyTorch



PROC. OF THE IEEE, NOVEMBER 1998

7

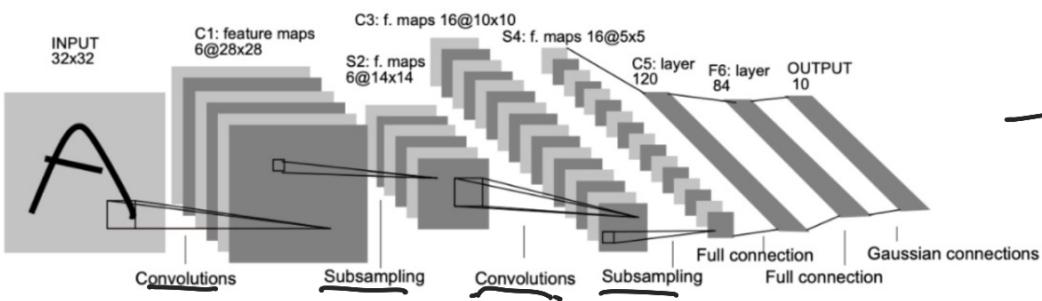


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Código

```
class LeNet5(nn.Module):
    def __init__(self, n_canales, n_clases):
        super().__init__()

        self.n_canales = n_canales
        self.n_clases = n_clases

        # le net tiene dos capas convolucionales
        self.conv1 = nn.Conv2d(self.n_canales, 6, kernel_size=5)
        self.conv2 = nn.Conv2d(6, 16, kernel_size=5)
        # que luego se conectan a 3 capas fully connected (perceptrones multi capa)
        self.fc1 = torch.nn.Linear(256, 120)
        self.fc2 = torch.nn.Linear(120, 84)
        self.fc3 = torch.nn.Linear(84, self.n_clases)

    def forward(self, x):
        # Primera capa convolucional. convolucion --> relu --> pooling
        x = self.conv1(x) # convolucion
        x = F.max_pool2d(F.relu(x), (2, 2)) # relu y luego pooling, one liner
        # segunda capa convolucional
        x = F.max_pool2d(F.relu(self.conv2(x)), (2, 2)) # todo en una misma linea pero el orden es el mismo, conv relu
        # vectorizo mis imagenes
        x = torch.flatten(x, 1)
        # capas de perceptrones 1,2 y 3
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        print(x.shape)
        # output
        return x
```

definición de mi Req. forward prop. 😊

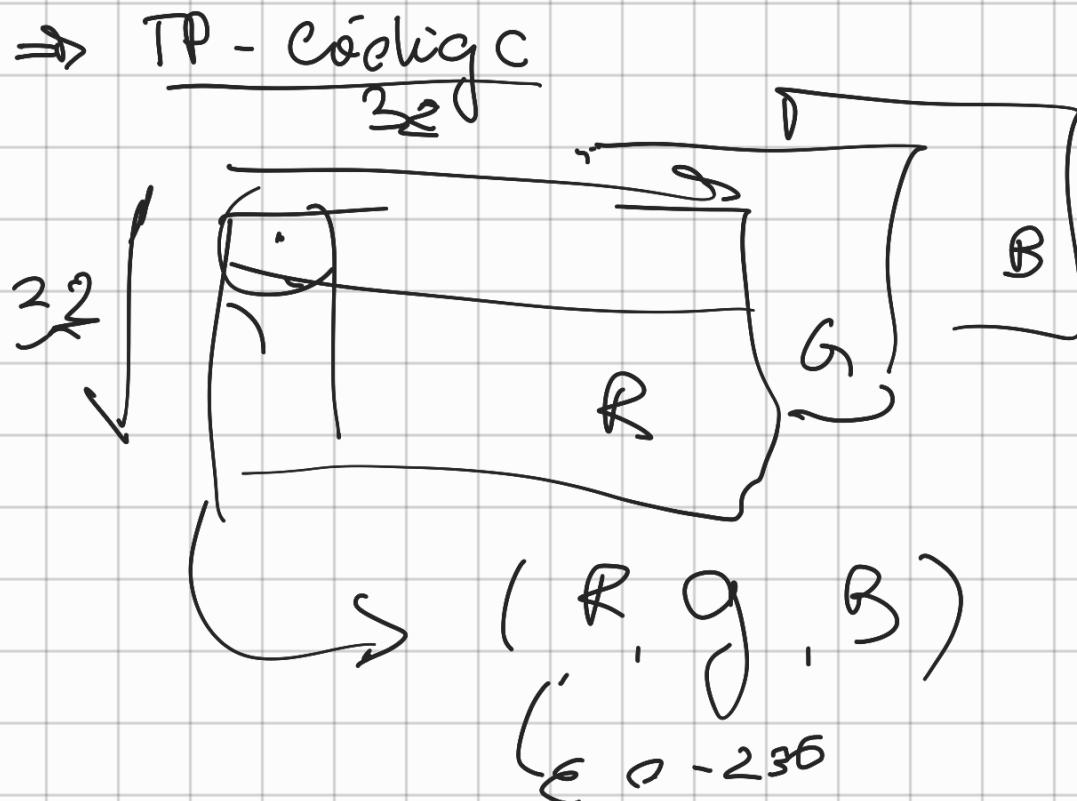
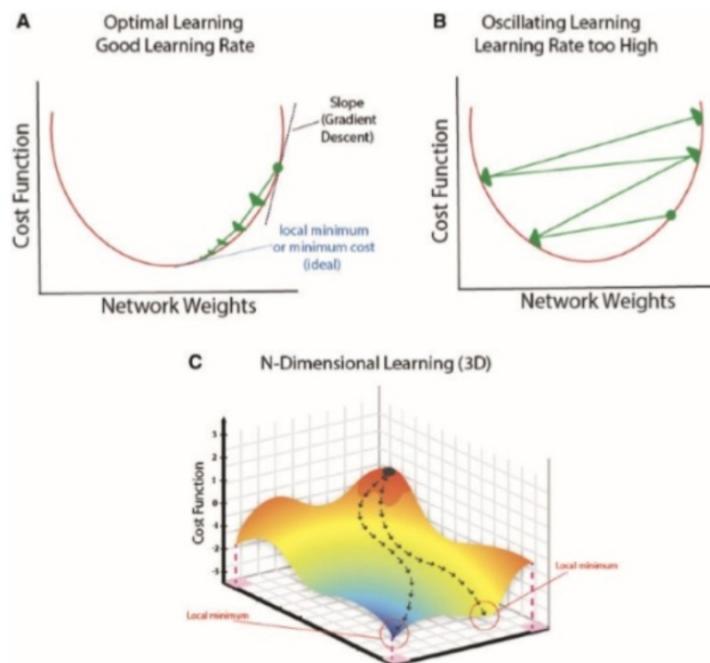
Problemas en Deep Learning

Vanishing Gradient

- La derivada se hace muy pequeña a medida que aumentamos capas por la retropropagación. Cuando se actualizan los pesos de la red, los valores son muy pequeños (> 0), por lo que detiene el entrenamiento de la red o lleva mucho tiempo..
- Ver de cambiar función de activación.

Exploding Gradient

Caso contrario al anterior, pero no depende de la función de activación si no de los pesos. Por lo tanto, puede no llegar a converger nunca.

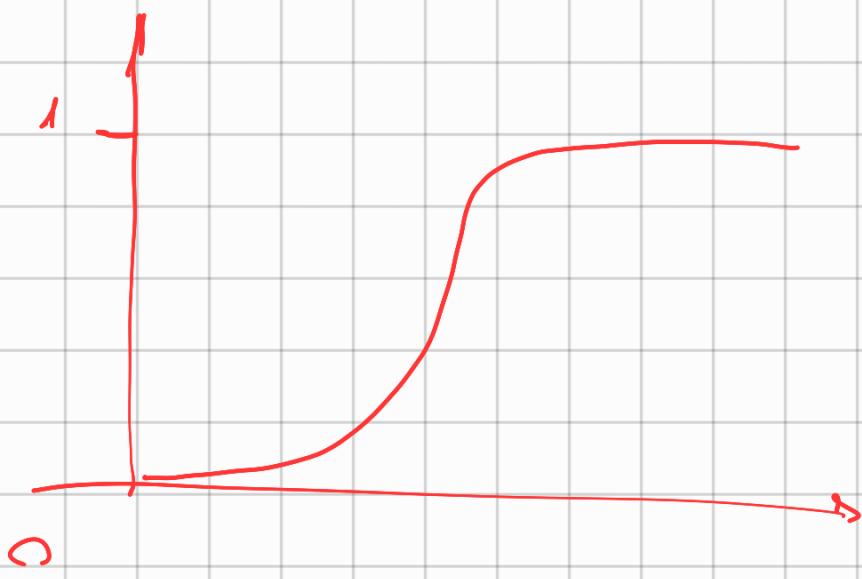


$$\begin{bmatrix} -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}$$

$$-1 \cdot 0,5 - 0,3 + 1 \cdot 0,9 = 1,1$$

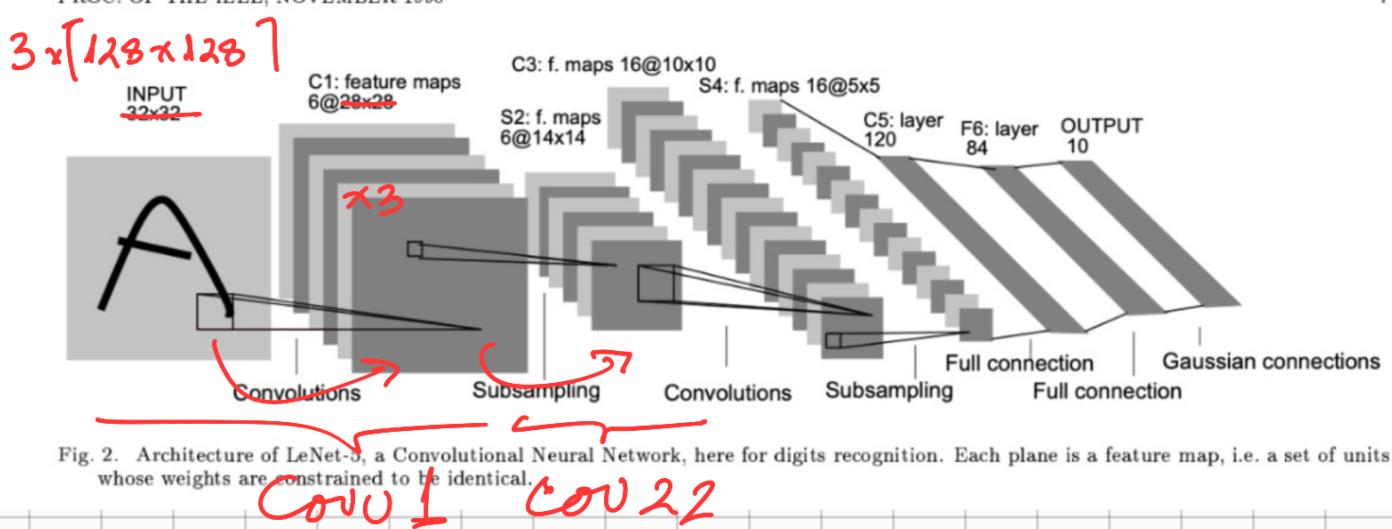
que signif ←

$$\begin{bmatrix} 0,3 & 0,5 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots \\ 0,3 & 1 & 0,9 & \dots \end{bmatrix} = \boxed{1,1, y}$$

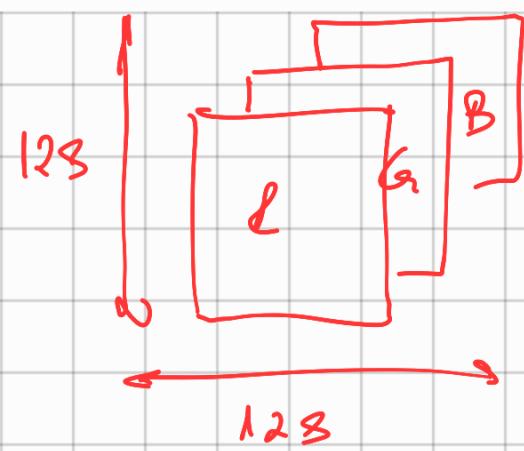


$$G = \frac{1}{1 + e^{-W^T x}}$$

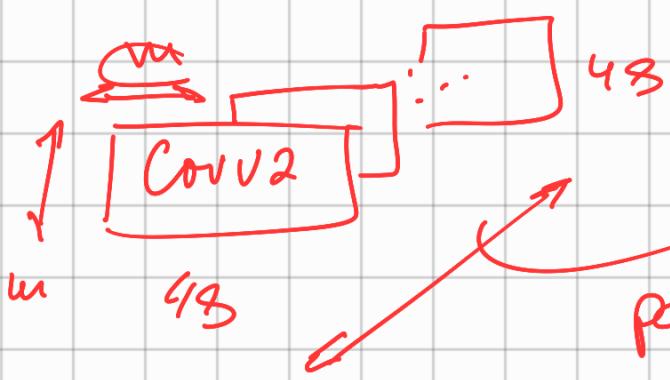
PROC. OF THE IEEE, NOVEMBER 1998



Conv2d (input channels, output channels, kernel size)
3? 6 5



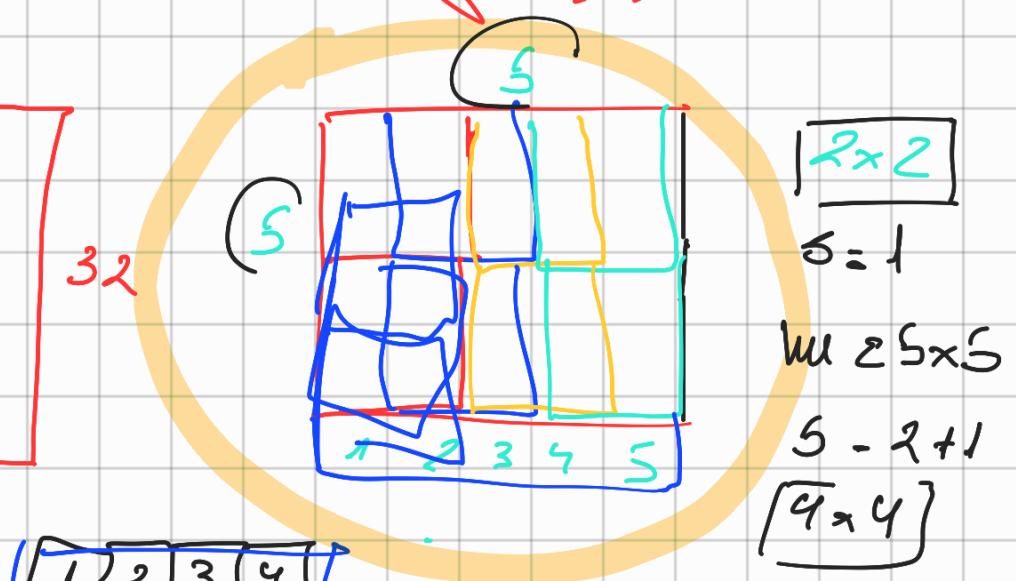
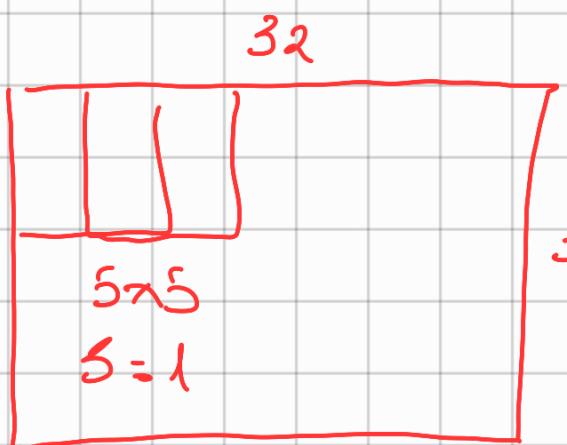
Cov output channels



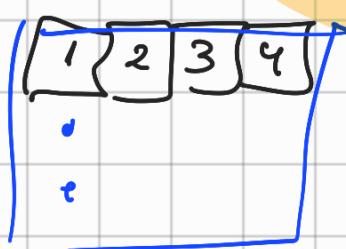
for $\delta \rightarrow 0$

$$40368 = f^2 \times 25$$

$$\frac{1030}{75} = 8$$



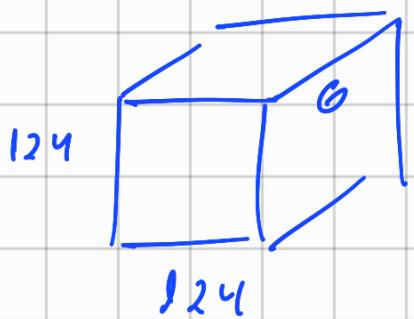
128×128



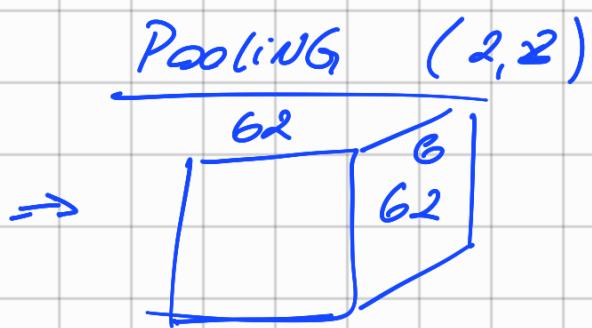
$$\Rightarrow 5 - 2 + 1 = 4$$

$$123 - 5 + 1 = \underline{124}$$

Tauño Inaugurales = Tauño Icy - Tauño + Sonda
post conv례urs - filtro

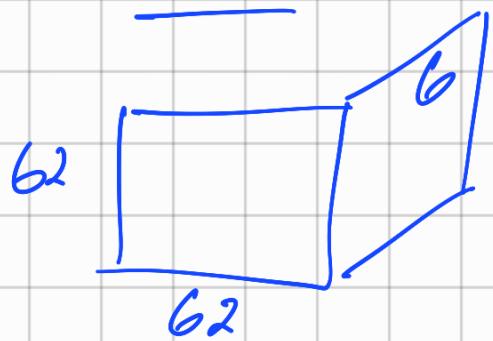


$$\Rightarrow 124 \times 124 \times 6 = \text{parameters}$$



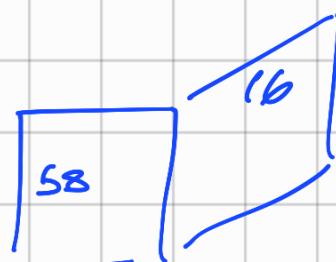
| Conv + Pool + ReLU

\Rightarrow Conv



Filter
[5x5]

16



$$62 - 5 + 1 = 58$$

\Rightarrow Pooling (2,2)



\Rightarrow Pooling

A la final del MLP $\Rightarrow 29 \times 29 \times 16$

$$= \boxed{13456}$$

