Aprendizaje de máquina - LeNet5

Bruno, César

November 25, 2019

LeNet-5

- LeNet-5 es una arquitectura de red convolucional desarrollada por Yann LeCun, de las primeras en emplearse para el reconocimiento de caracteres escritos a mano.
- Consta de 7 capas, determinadas por parámetros de entrenamiento.
- Se alimenta de imágenes 32×32 pixeles.
- En total cuenta con 410,000 conexiones pero tiene 90,000 parámetros libres.

Arquitectura LeNet-5

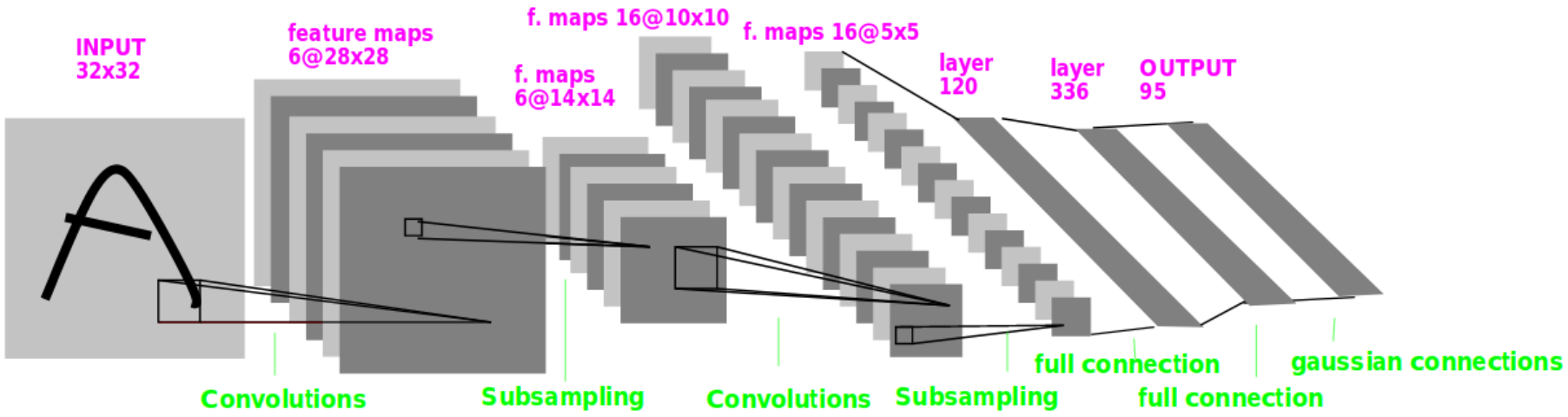

Figure: Arquitectura de la Red Convolucional LeNet-5

Arquitectura LeNet-5

- **input** Es una imagen de 32 X 32 en escala de grises
- **1a capa:** Capa convolucional que consta de 6 filtros de tamaño 5 X 5 y un stride de 1. La dimensión cambia a 28 X 28 X 6
- **2a capa:** capa de *average pooling* de tamaño 2 X 2 y un stride de 2 (sub-sampling)[1] La dimensión baja a 14 X 14 X 6
- **3a capa:** capa convolucional que consta de 16 filtros de tamaño 5 X 5 y un stride de 1. En esta capa solo 10 de los 16 filtros están conectados La dimensión cambia a 10 X 10 X 16

[1] Reduce dimensiones de c/feature map retienendo información más importante

Arquitectura LeNet-5

- **4a capa:** nuevamente capa de *average pooling* de tamaño 2 X 2 y de 2 strides. La dimensión cambia a 5 X 5 X 16
- **5a capa:** es una capa convolucional que conecta la salida de la cuarta capa (400 parámetros) a una capa completamente conectada de 120 nodos de de tamaño 5 X 5 La dimensión resultante es de 120
- **6a capa:** una capa igualmente completamente conectada que consta de 84 nodos, de la que se derivan de las salidas de los 120 nodos de la quinta capa. La dimensión reslutante es de 84.

Arquitectura LeNet-5

- **7a capa:** consiste en clasificar la salida de la última capa en 10 clases relacionadas con los 10 dígitos que se entrenó principalmente para clasificar.
- La función de activación que se usa es una tangente hiperbólica.

```python
class LeNet5(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 6, kernel_size=5, stride=1)
        self.average1 = nn.AvgPool2d(2, stride=2)
        self.conv2 = nn.Conv2d(6, 16, kernel_size=5, stride=1)
        self.average2 = nn.AvgPool2d(2, stride=2)
        self.conv3 = nn.Conv2d(16, 120, kernel_size=4, stride=1)
        self.flatten = Flatten()
        self.fc1 = nn.Linear(120, 82)
        self.fc2 = nn.Linear(82,10)

    def forward(self, xb):
        xb = xb.view(-1, 1, 28, 28)
        xb = F.tanh(self.conv1(xb))
        xb = self.average1(xb)
        xb = F.tanh(self.conv2(xb))
        xb = self.average2(xb)
        xb = F.tanh(self.conv3(xb))
        xb = xb.view(-1, xb.shape[1])
        xb = F.relu(self.fc1(xb))
        xb = F.relu(self.fc2(xb))
        return xb
```

Referencias

- **Gradient-Based Learning applied to document recognition**. Y. Lecun; L. Bottou; Y. Bengio; P. Haffner See http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
- **Página electrónica de Y. Lecun dedicada a la red LeNet** http://yann.lecun.com/exdb/lenet/