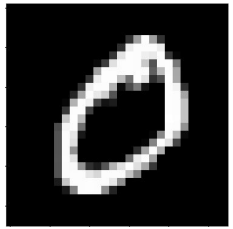
Redes Neuronales Convolucionales

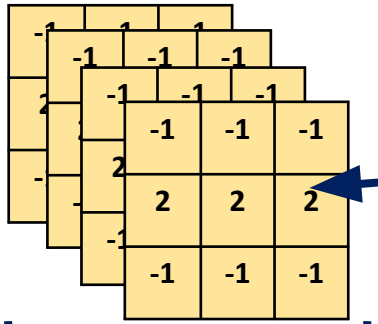
Capa convolucional

Una capa convolucional no es más que muchos filtros convolucionales de tamaño **K*K*C** (**K** se debe definir y **C** = canales de la imagen).

```
Conv2D(cant_filtros,
     kernel_size= k,
     strides= (n,m)
     activation='relu',
     padding = 'same')
```

En Keras





Estos valores son parte de los “pesos” de nuestra red.

Capa convolucional de 4 filtros de 3x3x1

Capa convolucional en MNIST

Veamos cómo sería aplicar una simple capa convolucional en el dataset MNIST.

Accuracy alimentando a la red Feed-Forward con la imagen cruda.

```
Train
   Accuracy: 0.93     soporte: 60000

Test
   Accuracy: 0.93     soporte: 10000
```

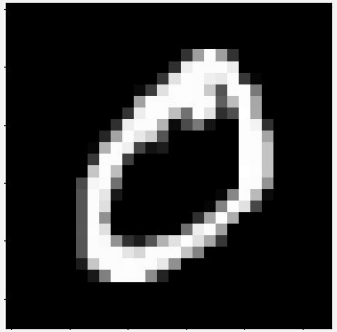
Accuracy al agregar una capa convolucional de 64 filtros.

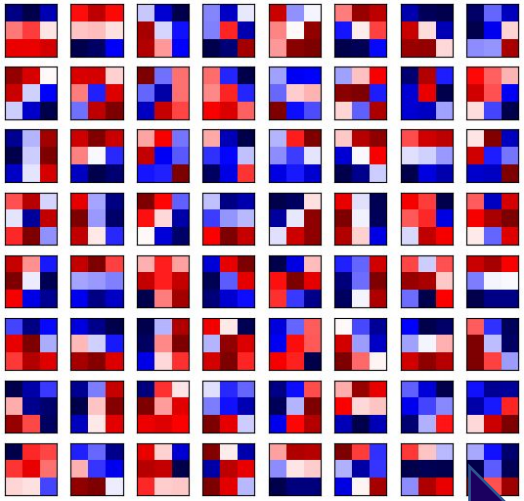
```
Train
   Accuracy: 1.00     soporte: 60000

Test
   Accuracy: 0.98     soporte: 10000
```

```
model = Sequential()
model.add(Conv2D( 64, kernel_size=3,
                  activation='relu',
                  input_shape= INPUT_SHAPE))

model.add(Flatten())
model.add(Dense(n_clases, activation= 'softmax'))
```

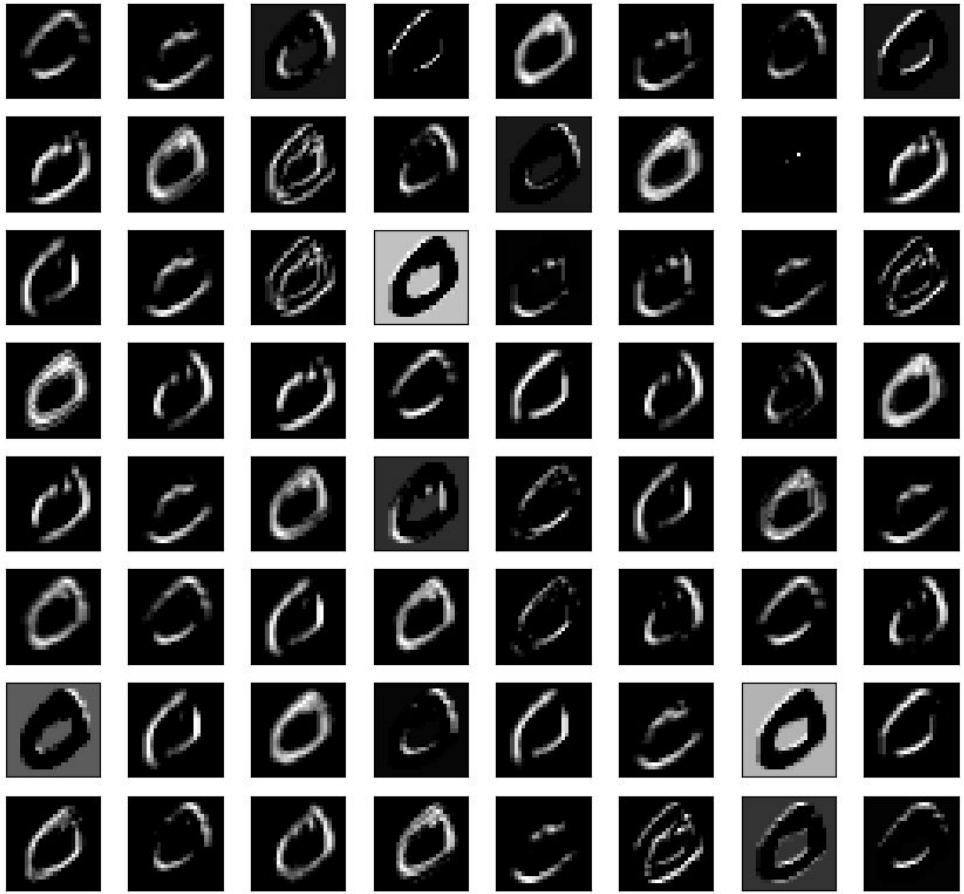
Capa convolucional

Imágenes filtradas con 64 filtros convolucionales (activation maps)





64 filtros

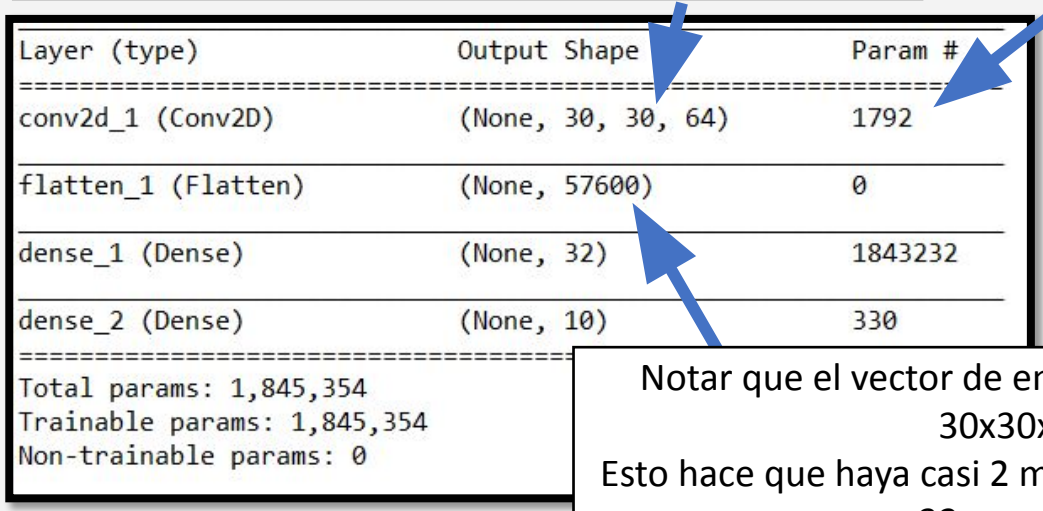


Capa Convolucional sobre CIFAR10

```
model = Sequential()
model.add(Conv2D( 64,   kernel_size=3,activation='relu',input_shape= INPUT_SHAPE))

model.add(Flatten())
model.add(Dense(32, activation= 'relu'))
model.add(Dense(n_clases, activation= 'softmax'))
```

¿Por qué la imagen resultante tiene 30x30?

¿Por la capa convolucional tiene 1792 parámetros (pesos)?



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 30, 30, 64)	1792
flatten_1 (Flatten)	(None, 57600)	0
dense_1 (Dense)	(None, 32)	1843232
dense_2 (Dense)	(None, 10)	330

Total params: 1,845,354
Trainable params: 1,845,354
Non-trainable params: 0

64 x 3 x 3 x 3 + 64

Cant filtros

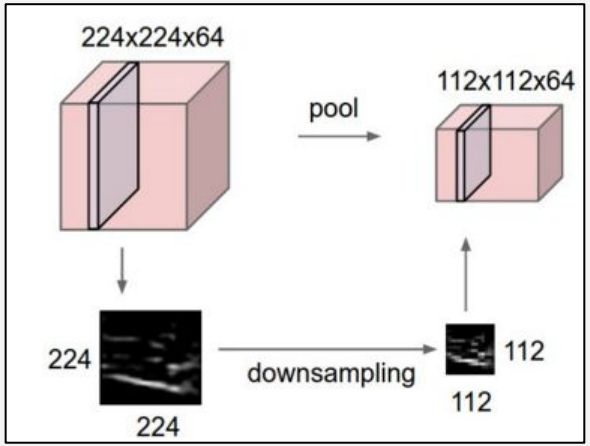
Tamaño del filtro

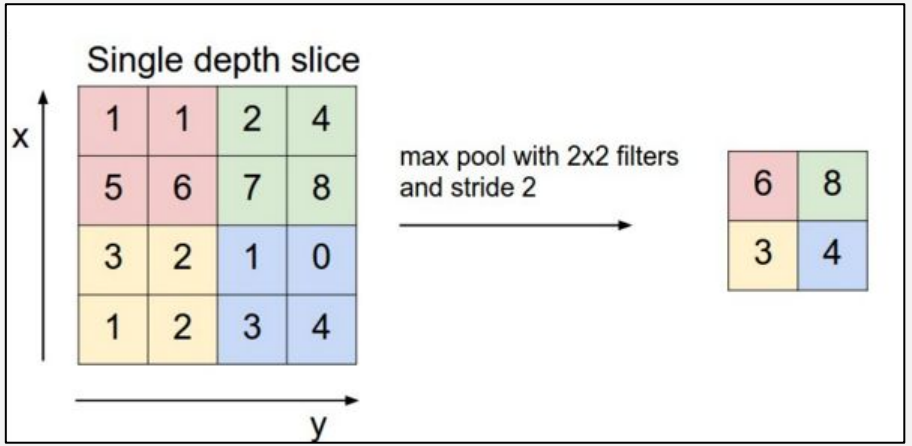
Bias

Notar que el vector de entrada a la Feed-Forward es de 30x30x64= 57600.
Esto hace que haya casi 2 millones de parámetros! solo para 32 neuronas ocultas.

Capas Pooling

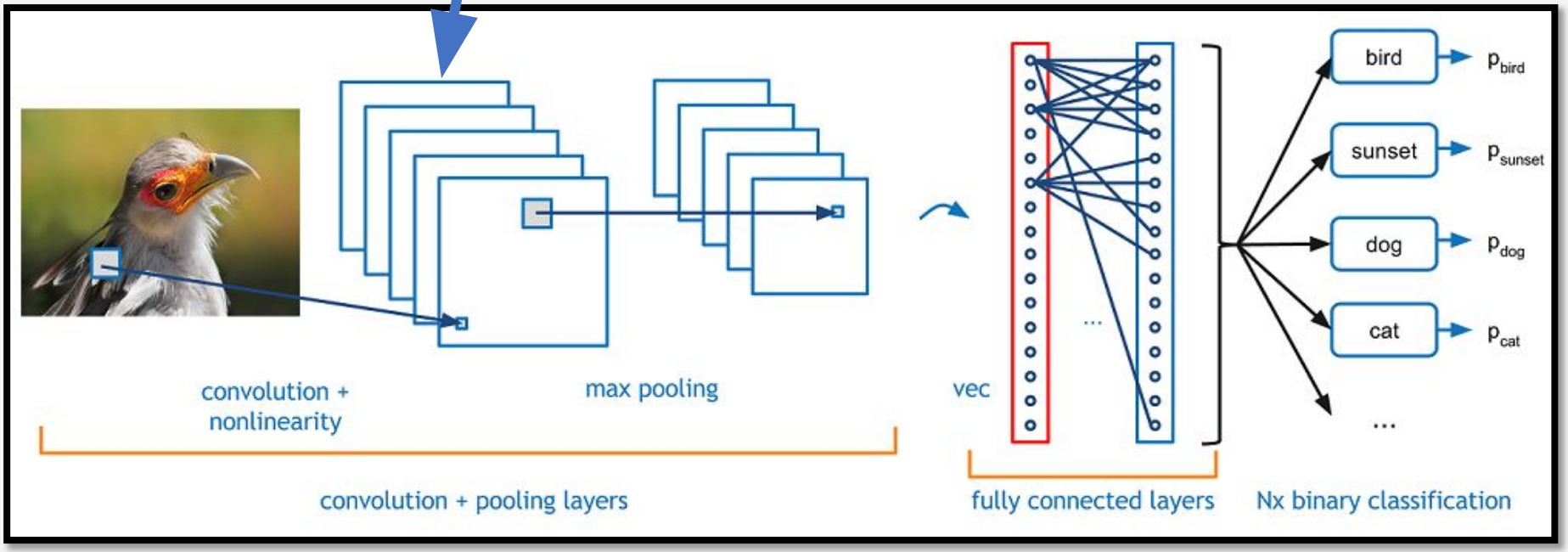
Las capas Pooling ayudan a reducir la dimensionalidad espacial del feature map. Básicamente son convoluciones con un stride igual al tamaño del kernel y donde se calcula alguna función sobre todos los píxeles. Lo más usual es el máximo, el mínimo o el promedio.
No solo reducen la dimensionalidad, sino que generalmente ayudan en la clasificación.





Capas Pooling

Generalmente se grafican los Feature maps, no los kernels.
Ya que esto se da como entrada a la próxima capa.



Capa Convolucional + Pooling - CIFAR10

```
model = Sequential()
model.add(Conv2D( 64,   kernel_size=3,activation='relu',input_shape= INPUT_SHAPE))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(32, activation= 'relu'))
model.add(Dense(n_clases, activation= 'softmax'))
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 30, 30, 64)	1792
max_pooling2d_1 (MaxPooling2	(None, 15, 15, 64)	0
flatten_1 (Flatten)	(None, 14400)	0
dense_1 (Dense)	(None, 32)	460832
dense_2 (Dense)	(None, 10)	330

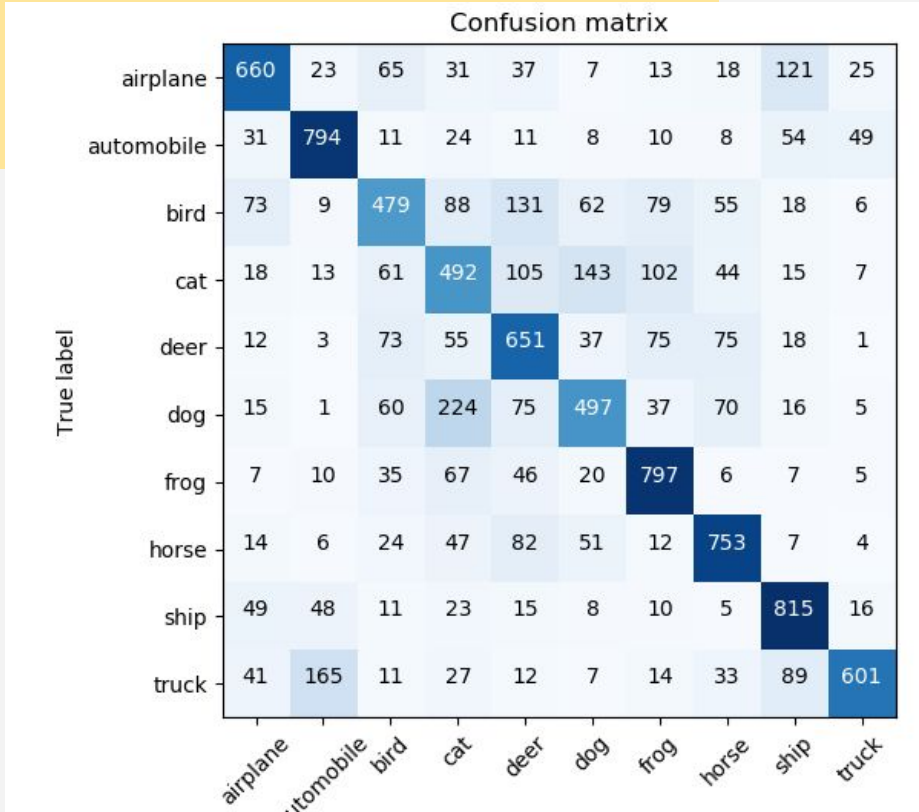
Total params: 462,954
Trainable params: 462,954
Non-trainable params: 0

Agregando la capa Pooling la cantidad de parámetros se redujo a un cuarto

Capa Convolucional + Pooling - CIFAR10

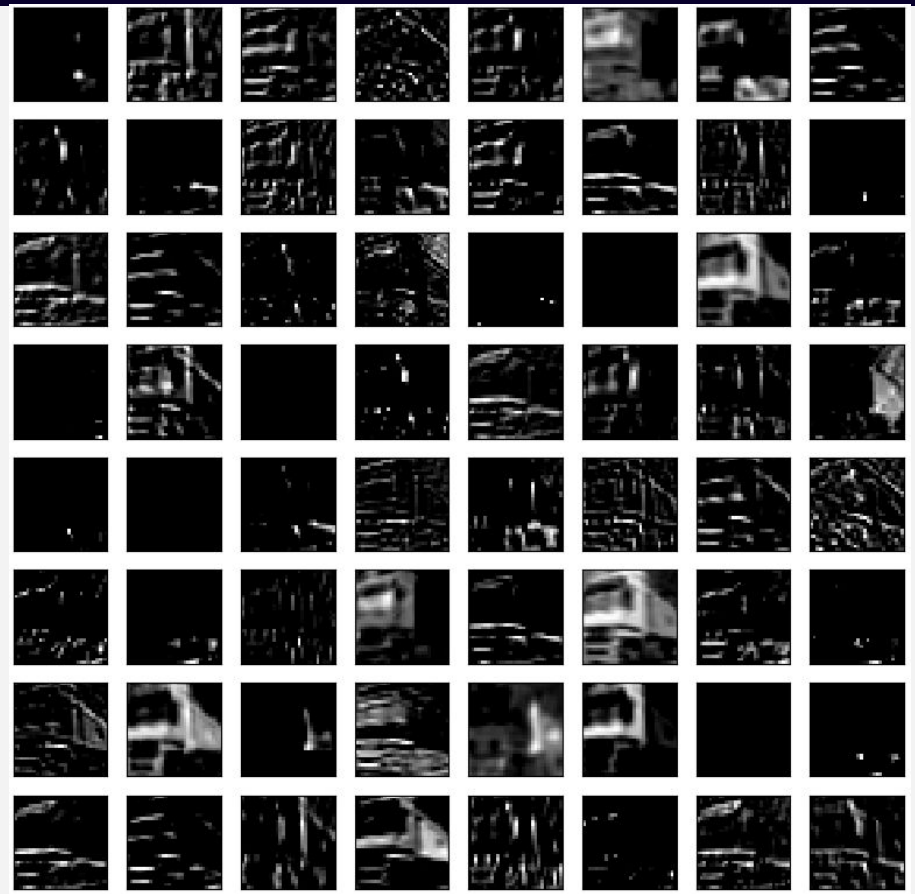
```
model = Sequential()
model.add(Conv2D( 64,   kernel_size=3,activation='relu',input_shape= INPUT_SHAPE))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(32, activation= 'relu'))
model.add(Dense(n_clases, activation= 'softmax'))
```

```
Train
   Accuracy: 0.78     soporte: 50000

Test
   Accuracy: 0.65     soporte: 10000
```



Filtros convolucionales sobre CIFAR10

Imágenes filtradas con los 64 filtros convolucionales (activation map)



64 filtros de 3x3x3

Capas convolucionales

A medida que las capas se apilan, los filtros convolucionales se aplican sobre los feature maps de las capas anteriores.

Imagen de entrada color (RGB)

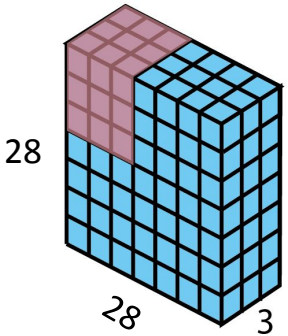
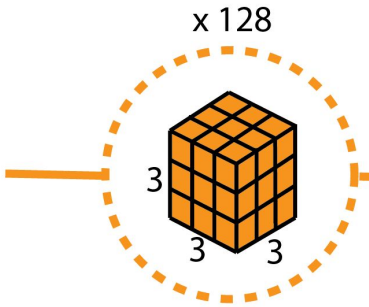
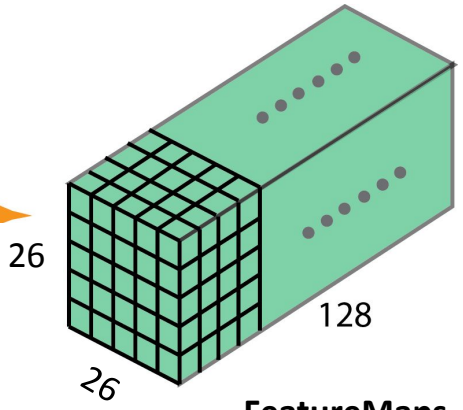


Imagen RGB



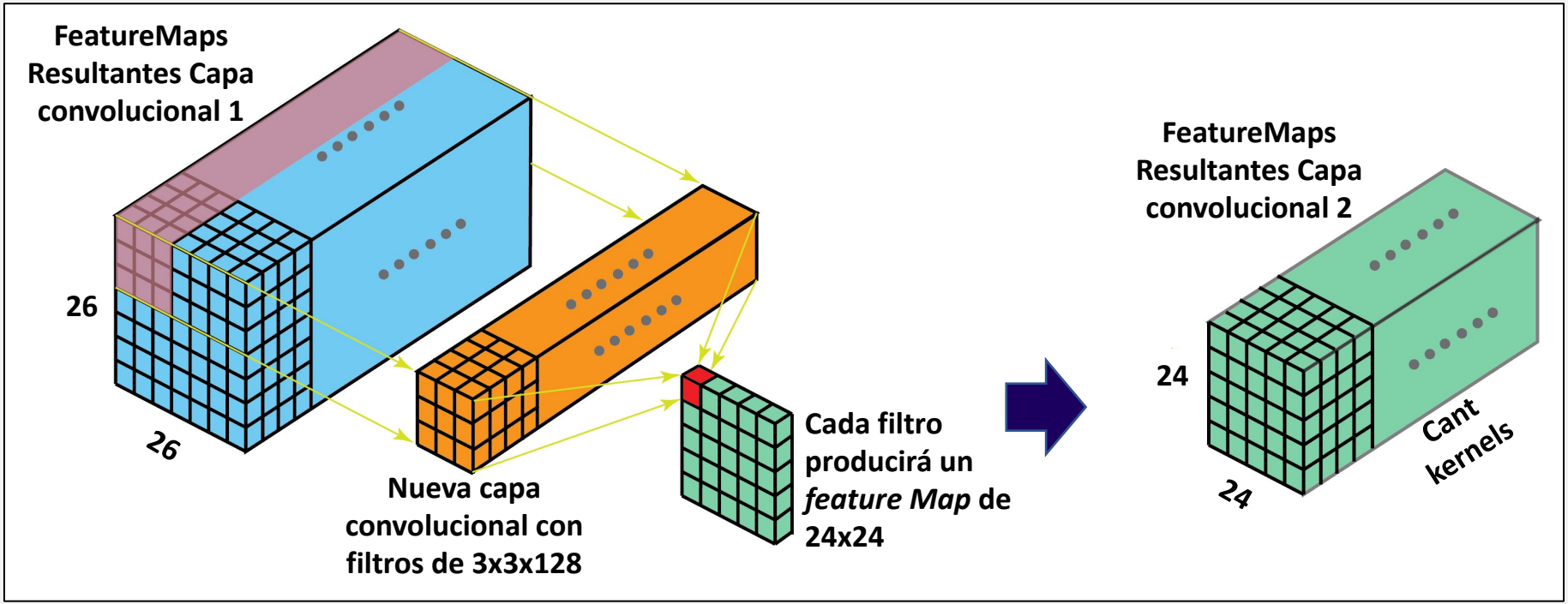
128 filtros de 3x3



FeatureMaps Resultantes de los 128 filtros

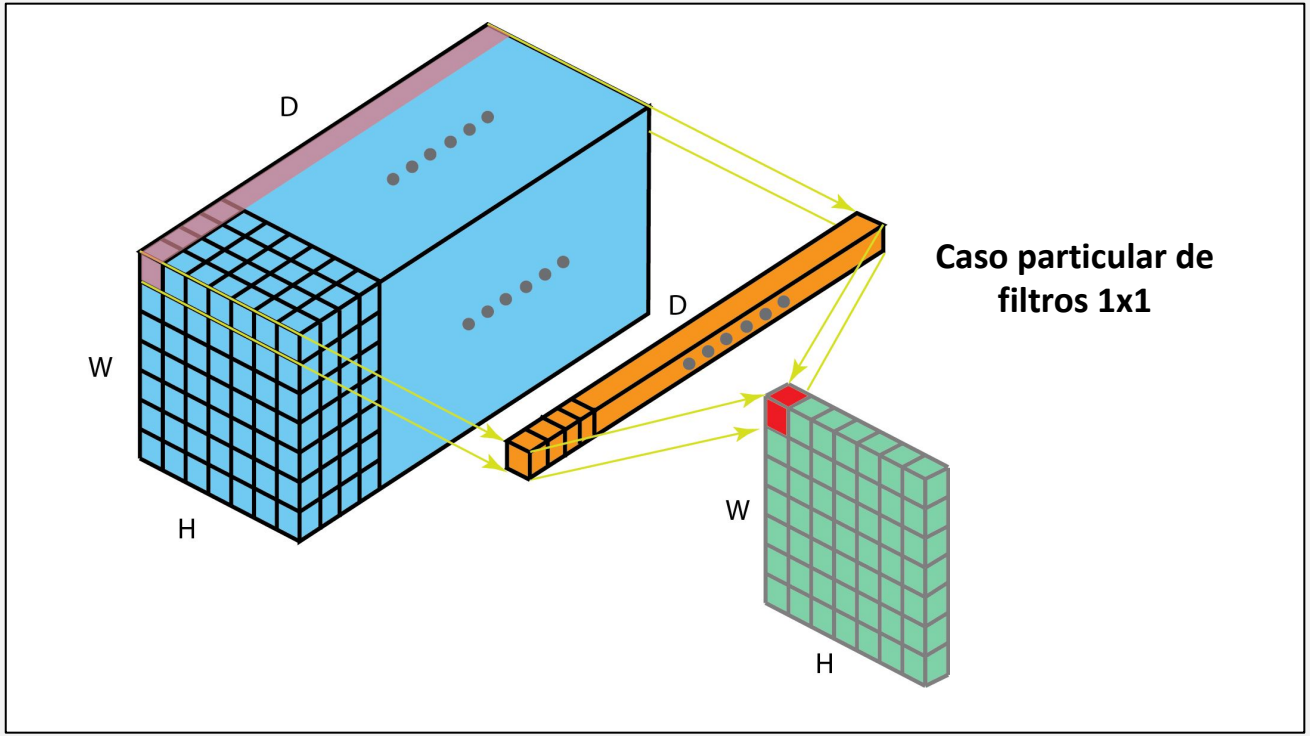
Capas convolucionales

A medida que las capas se apilan, los filtros convolucionales se aplican sobre los feature maps de las capas anteriores.



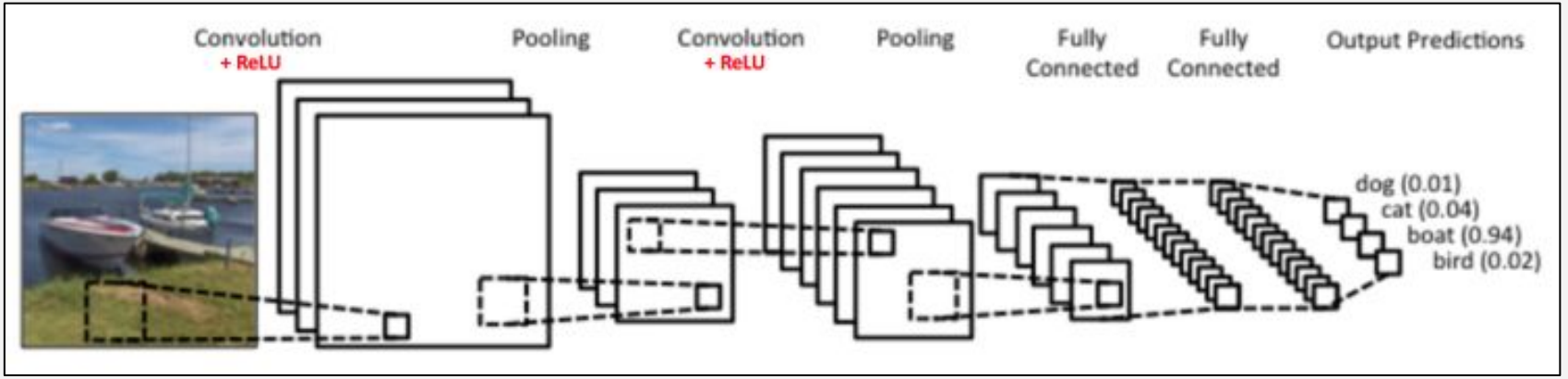
Kernel sizes

Lo más usual es tener tamaños de kernel de 3x3, 5x5 y 1x1.



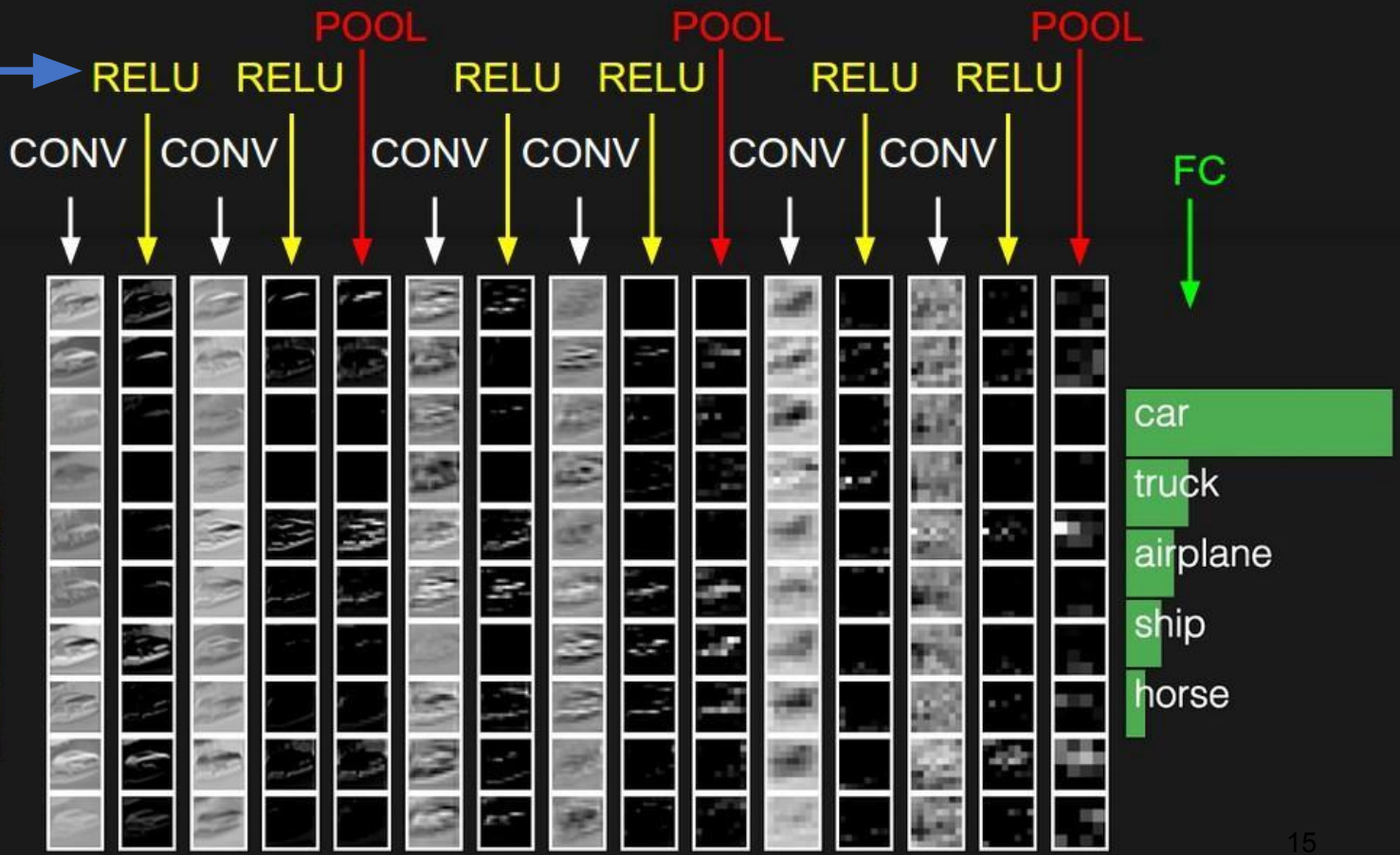
Red Convolucional estándar

Generalmente se suelen utilizar varias capas convolucionales seguidas de capas pooling.



Red Convolucional estándar

La función de activación es muy importante para generar una transformación no lineal en la salida de las neuronas.



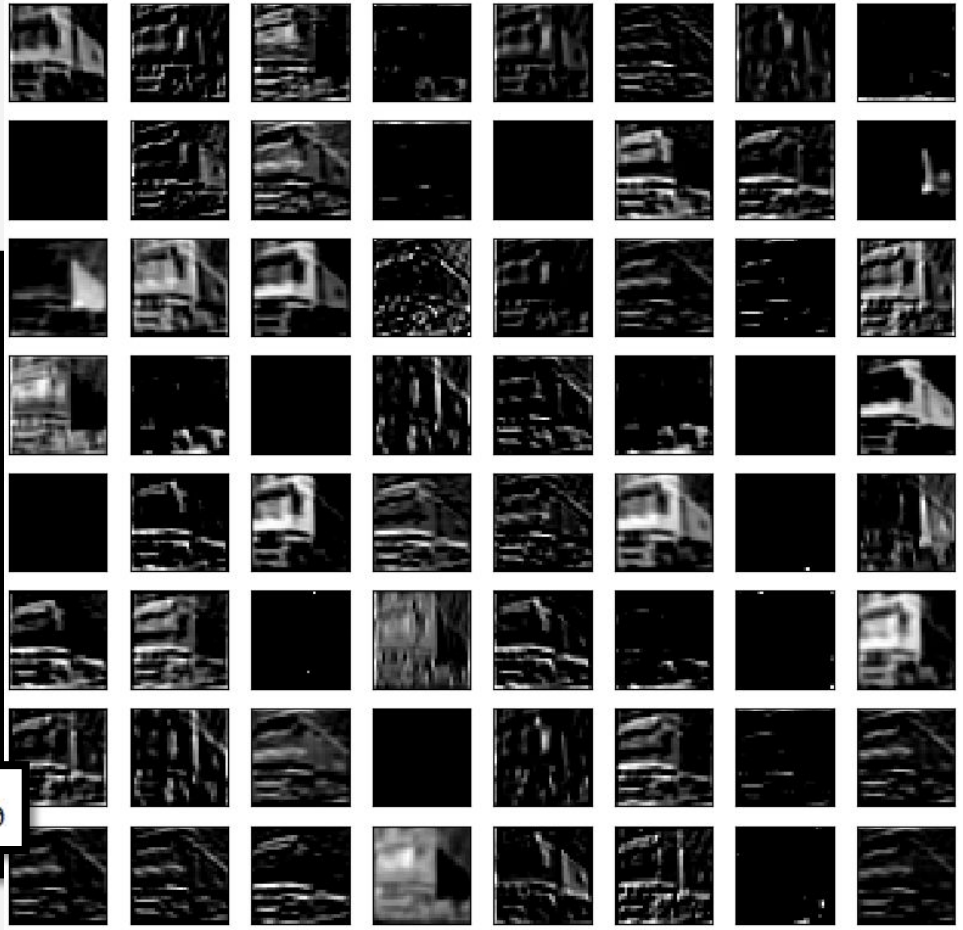
Varias capas sobre CIFAR10

Modelo más “profundo” para clasificar CIFAR10

```
#create model
model = Sequential()
#add model layers
model.add(Conv2D(64, kernel_size=3, activation='relu’,
                 input_shape= INPUT_SHAPE, padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=3, activation='relu’,
                 input_shape= INPUT_SHAPE, padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128,input_dim=d_in, activation= 'relu'))
model.add(Dense(n_clases, activation= 'softmax))
```

Varias capas sobre CIFAR10

Salida primer capa convolucional



Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 32, 32, 64)	1792
max_pooling2d_3 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_5 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_4 (MaxPooling2	(None, 8, 8, 64)	0
flatten_2 (Flatten)	(None, 4096)	0
dense_3 (Dense)	(None, 128)	524416
dense_4 (Dense)	(None, 10)	1290

Total params: 564,426
Trainable params: 564,426
Non-trainable params: 0

Test
Accuracy: 0.70 soporte: 10000

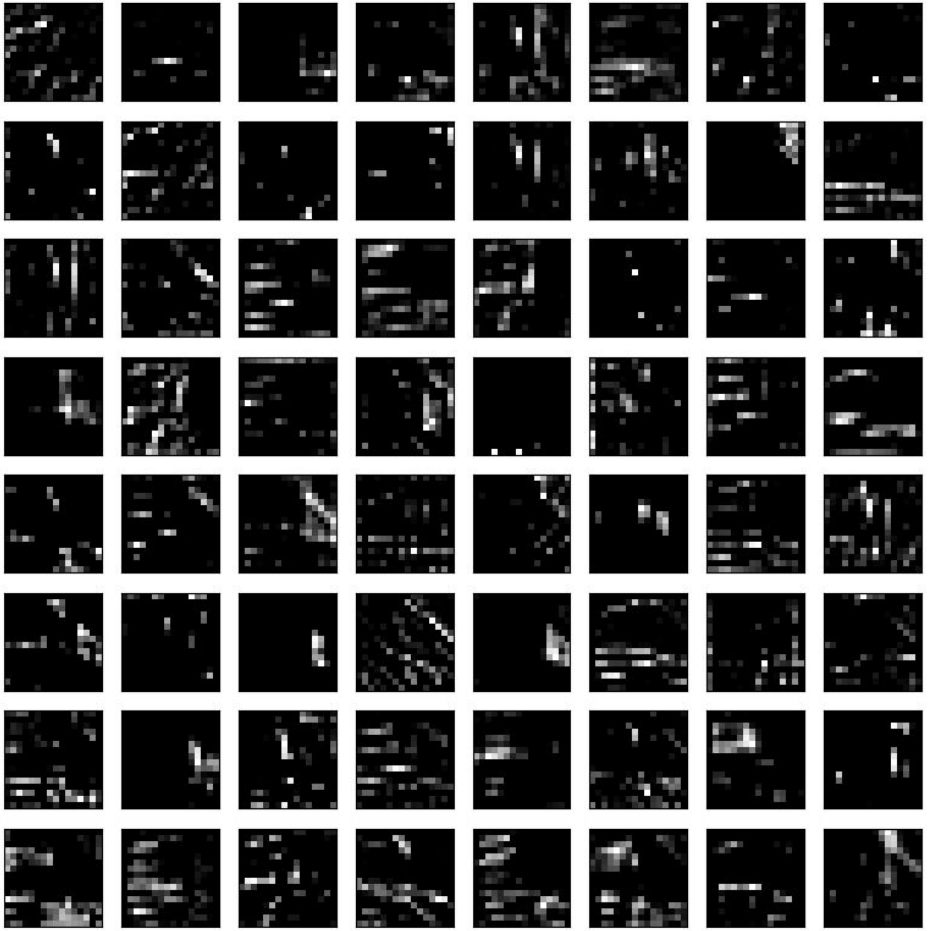
Varias capas sobre CIFAR10

Después de Max Pooling 2x2

Feature Map= 64x16x16

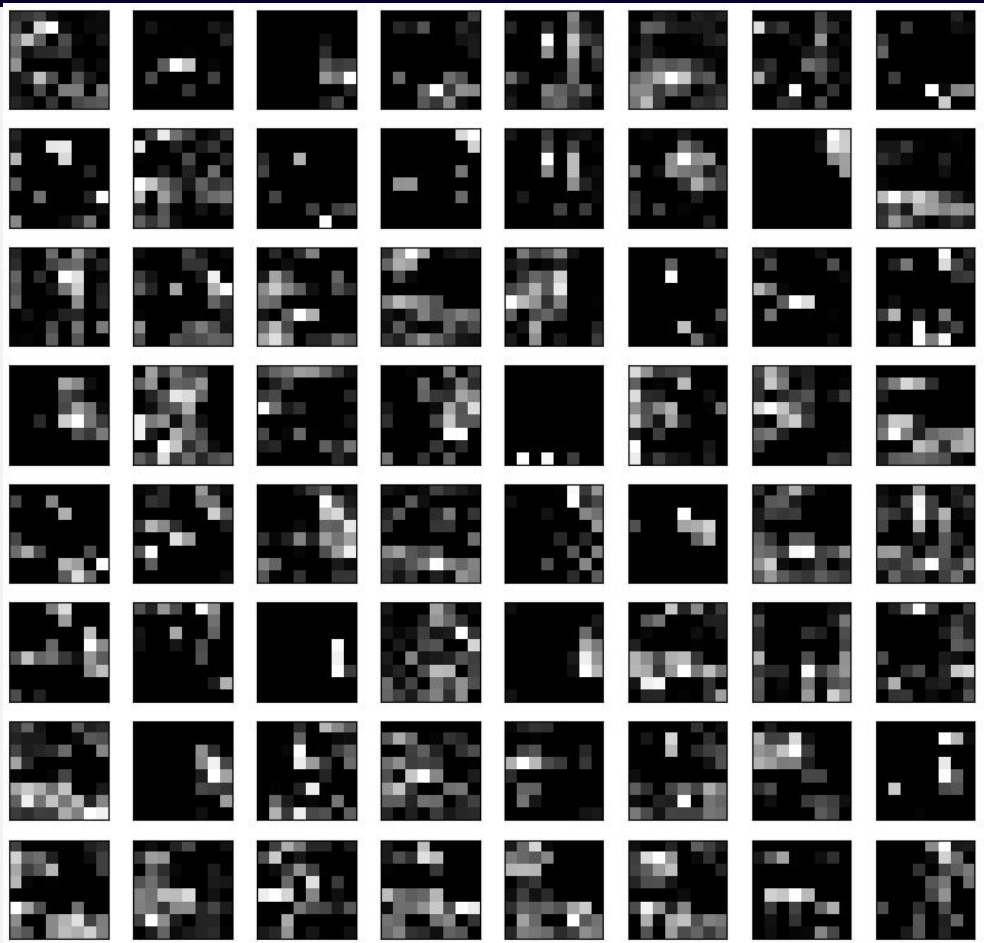
Varias capas sobre CIFAR10

Después de segunda capa convolucional.
Feature Map= 64x16x16



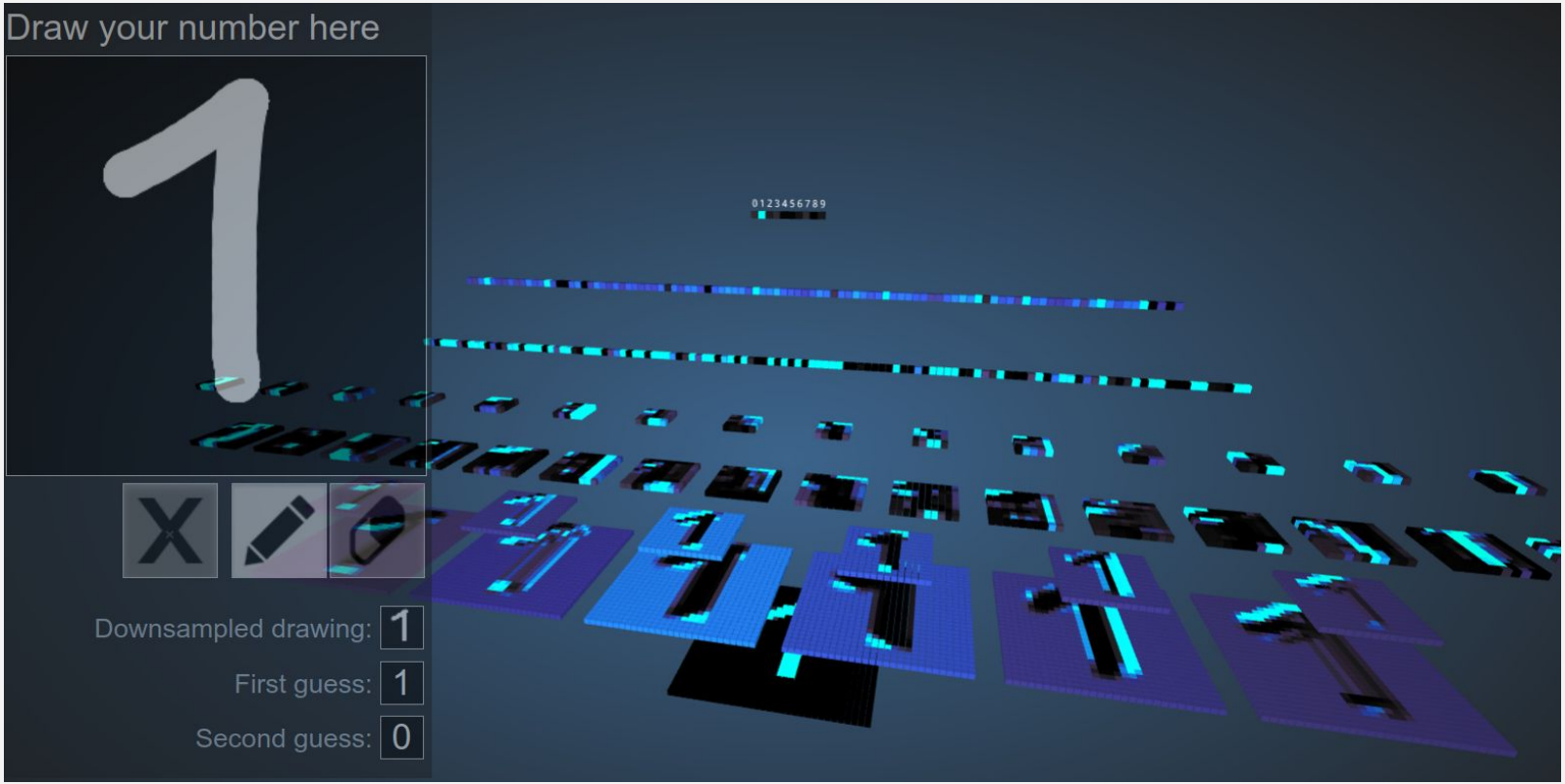
Varias capas sobre CIFAR10

Después de Max Pooling 2x2
Feature Map= 64x8x8



Visualización de Redes Convolucionales

- https://projects.trinadh.com/conv-neural-net/



Resumen

- Las Capas convolucionales 2D tienen filtros (kernels) que se entrenan para detectar diferentes características.
- Cada filtro genera un Feature Map de NxNx1. Donde N dependerá del tamaño original de la imagen, del filtro, el padding y el stride que usemos.
- Todos los Feature Maps de los distintos filtros de una capa convolucional se apilan, generando una nueva “imagen” (Activation Map) de NxNxF, siendo F la cantidad de filtros.
- Las capas Pooling permiten reducir la dimensionalidad del problema, haciendo no solo más rápido el entrenamiento sino más eficaz al momento de generalizar.
- El modo más ordenado de realizar una arquitectura ConvNet es intercalar capas Convolucionales con capas Pooling hasta llegar a las capas Dense (Feed-Forward) que discriminarán las características aprendidas por las capas anteriores.