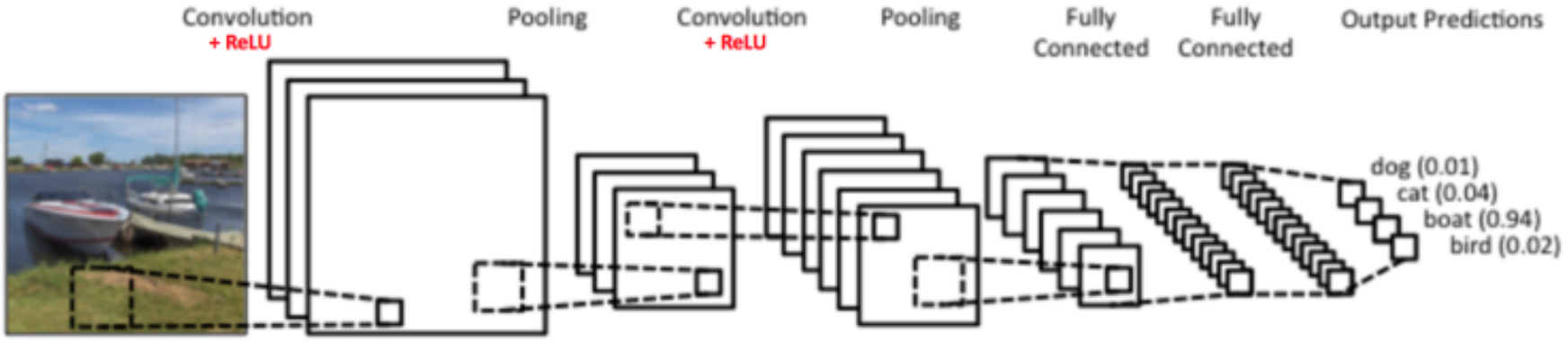
Deep Learning

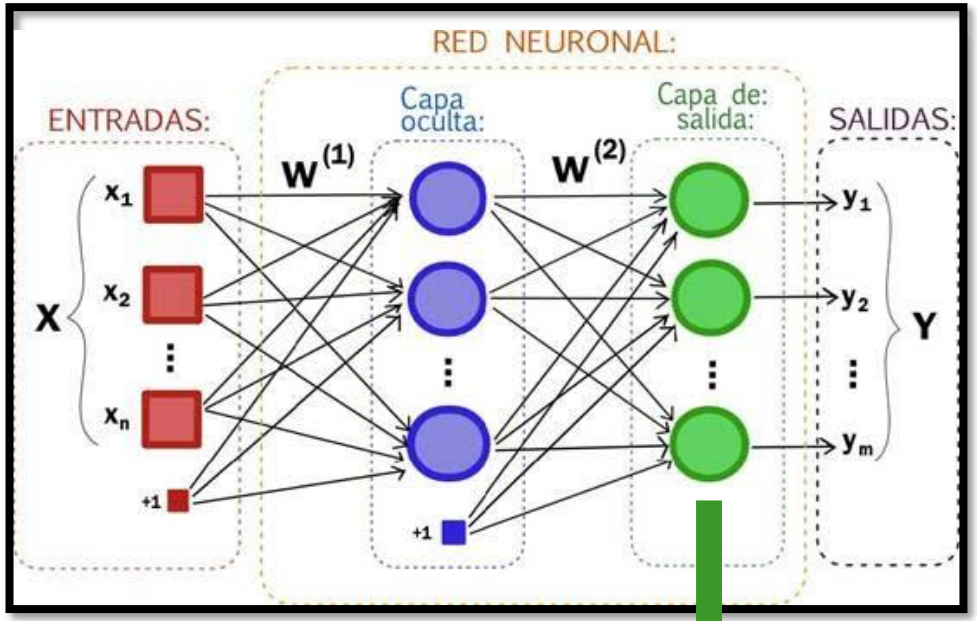
Deep Learning

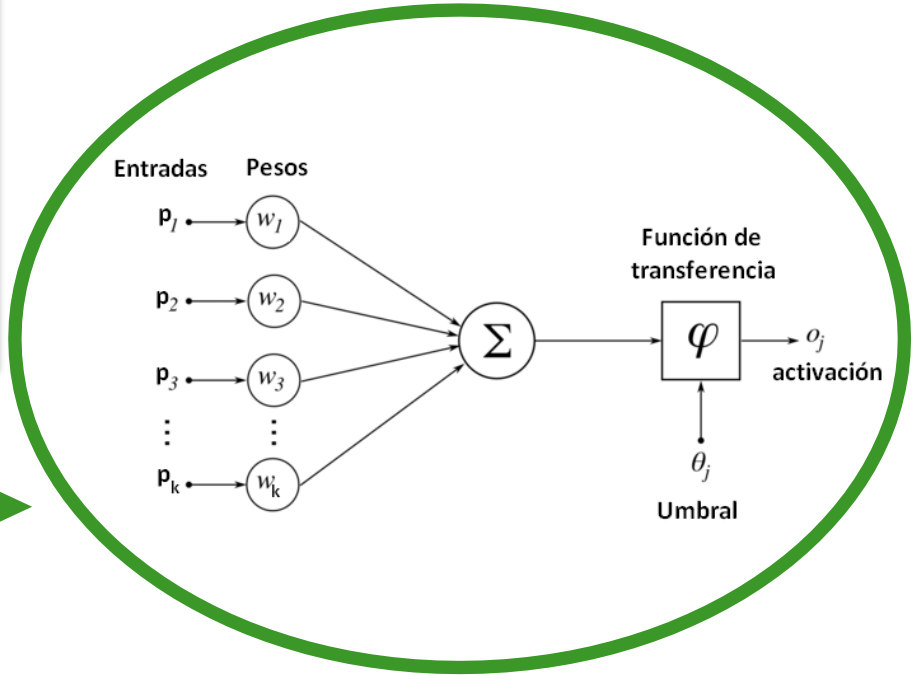
A partir del 2006 (aprox.) aparece el Deep Learning (aprendizaje profundo), que no es más que una extensión de las redes neuronales feed-forward agregándole el concepto de capas convolucionales.

Capas convolucionales y de pooling en primera instancia permiten filtrar la imagen de modo de generar un descriptor que puede ser “aprendido”. Al final suele haber una Red Neuronal Feed-Forward (tradicional) para clasificar.



Deep Learning – Capa Densa





Deep Learning – Capa de Convolución

- Es el segundo (o primer?) tipo de capa más importante junto con la Lineal
- Ideal para imágenes/audio
- Menos parámetros que las lineales, menos cómputo
- Implementan una convolución con un filtro aprendido

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

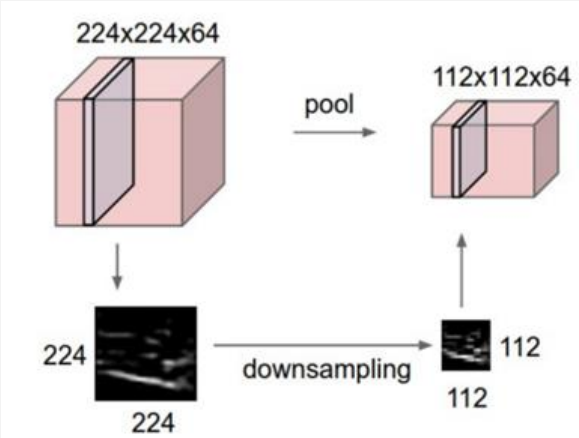
0	-1	0
-1	5	-1
0	-1	0

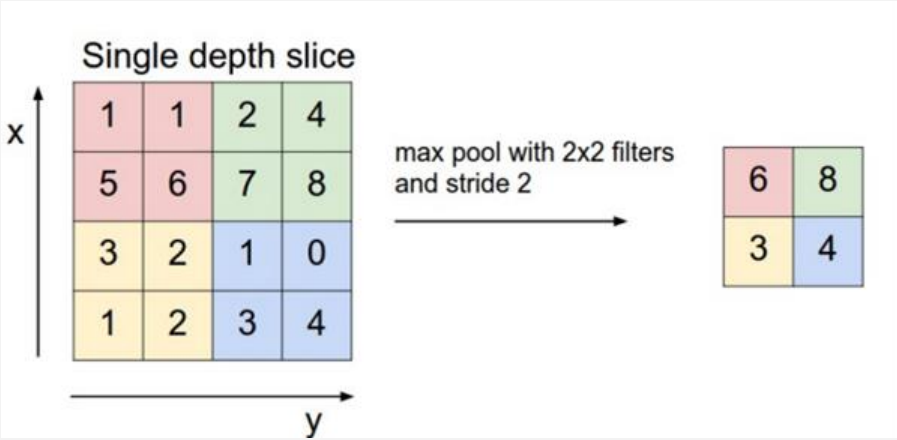
114				

Capas Pooling

Las capas Pooling ayudan a reducir la dimensionalidad espacial del *feature map* de una convolución. 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 calcular el máximo, el mínimo o el promedio.

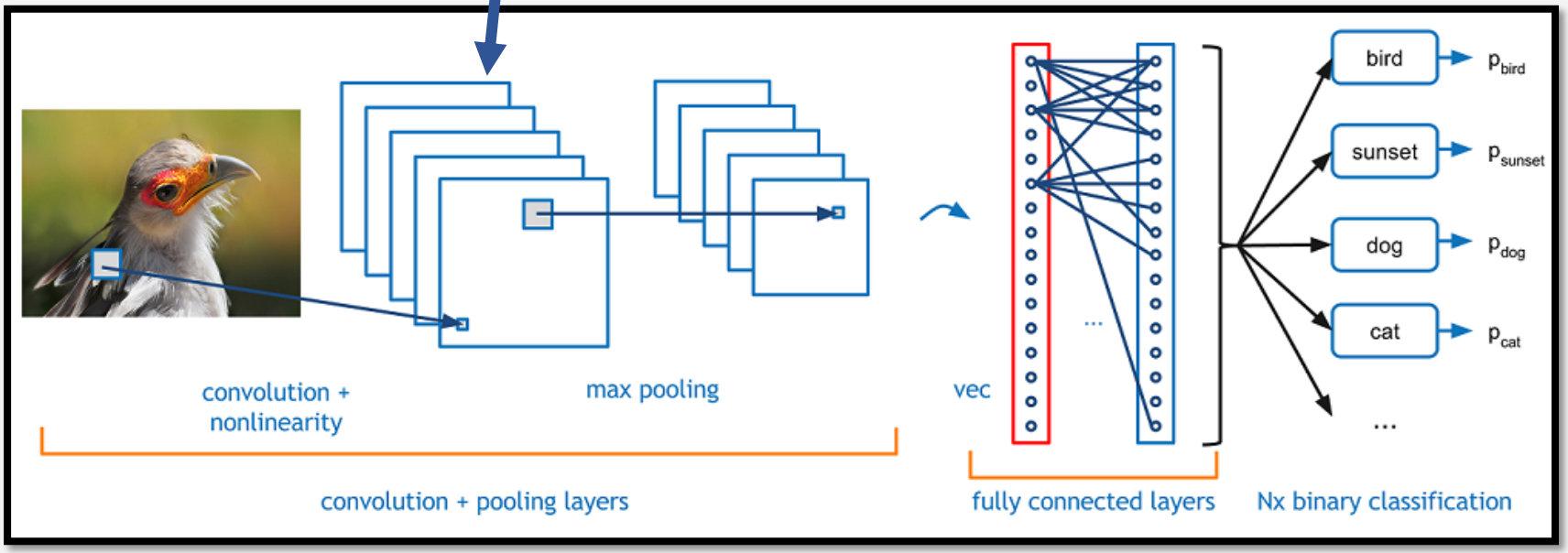
No solo reducen la dimensionalidad, sino que generalmente ayudan en la clasificación.





Capas Pooling

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

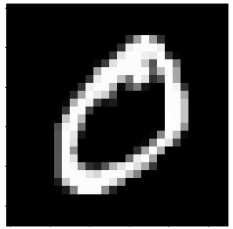


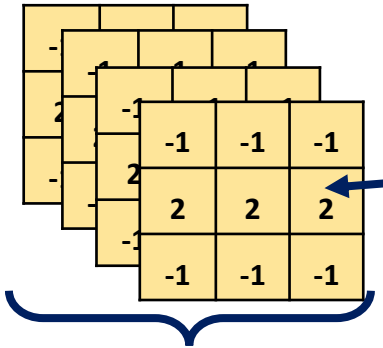
Capa convolucional en MNIST

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
```

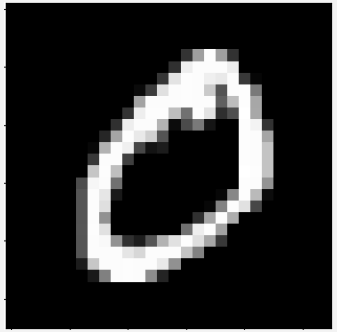
Accuracy al agregar una capa convolucional de 64 filtros.

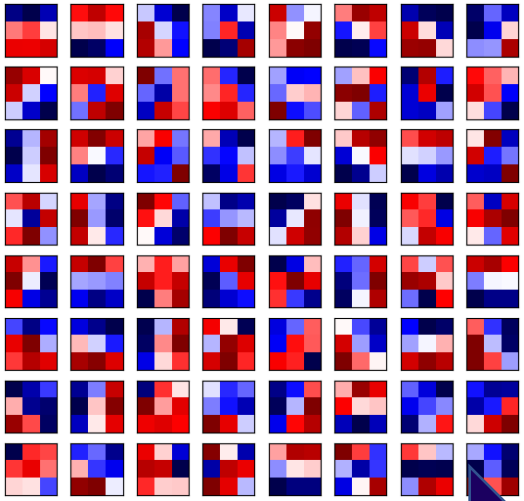
```
Train
   Accuracy: 1.00     soporte: 60000

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'))
```

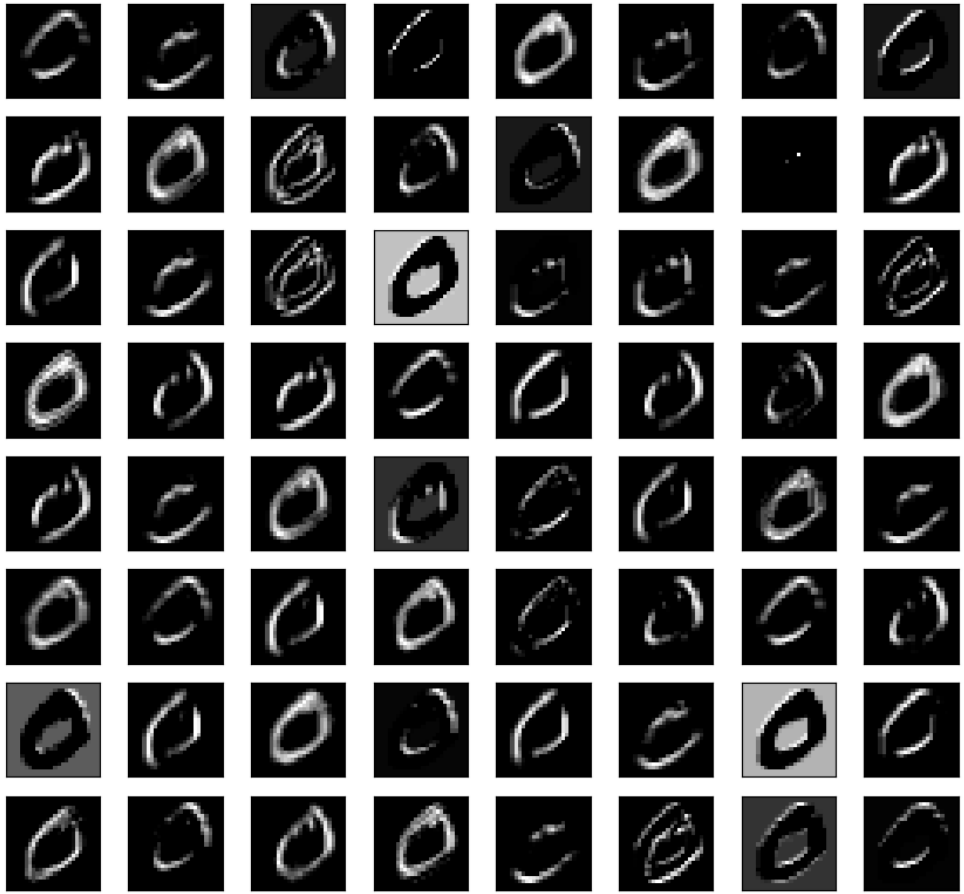
Capa convolucional

Imágenes filtradas con 64 filtros convolucionales (activation maps)

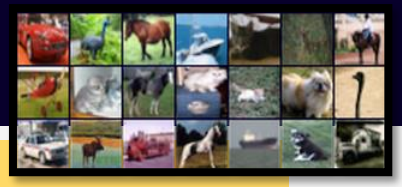




64 filtros



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.

Capa Convolucional+Pooling sobre 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'))
```

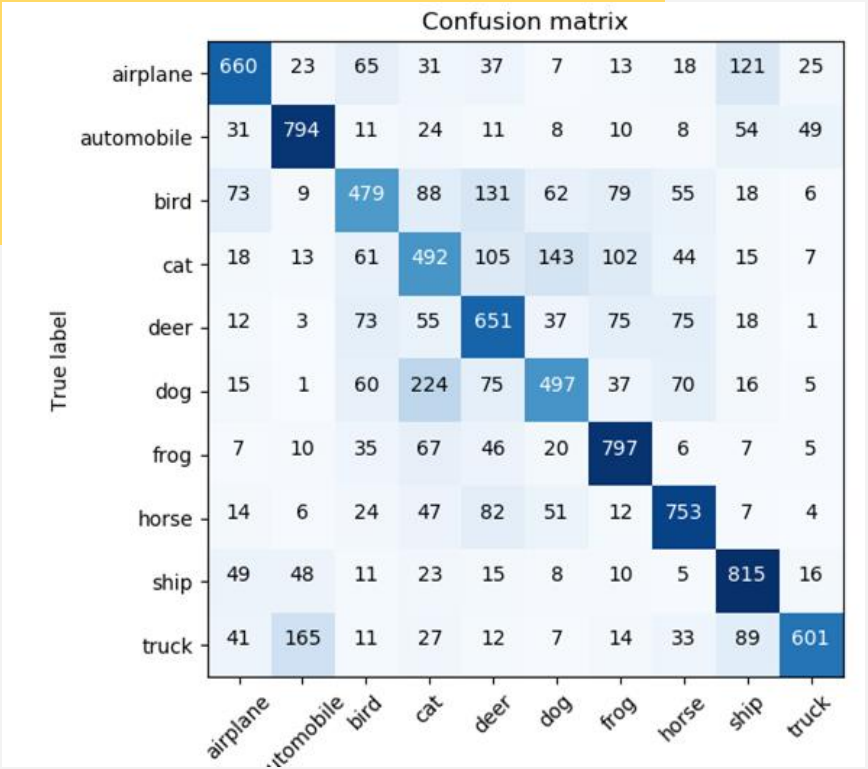
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 64)        1792
_________________________________________________________________
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 sobre CIFAR10

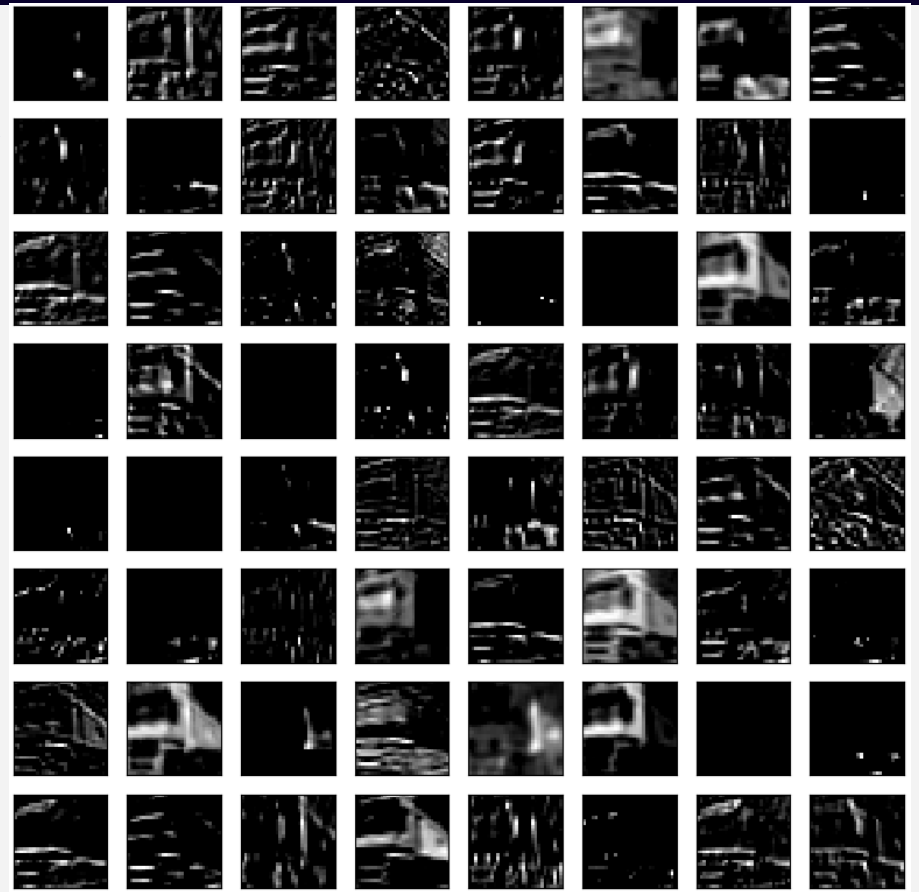
```
model = Sequential()
model.add(Conv2D( 64, kernel_size=3,activation='relu', input_shape= INPUT_SHAPE))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(32, activation= 'relu'))
model.add(Dense(n_clases, activation= 'softmax'))
```

```
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.

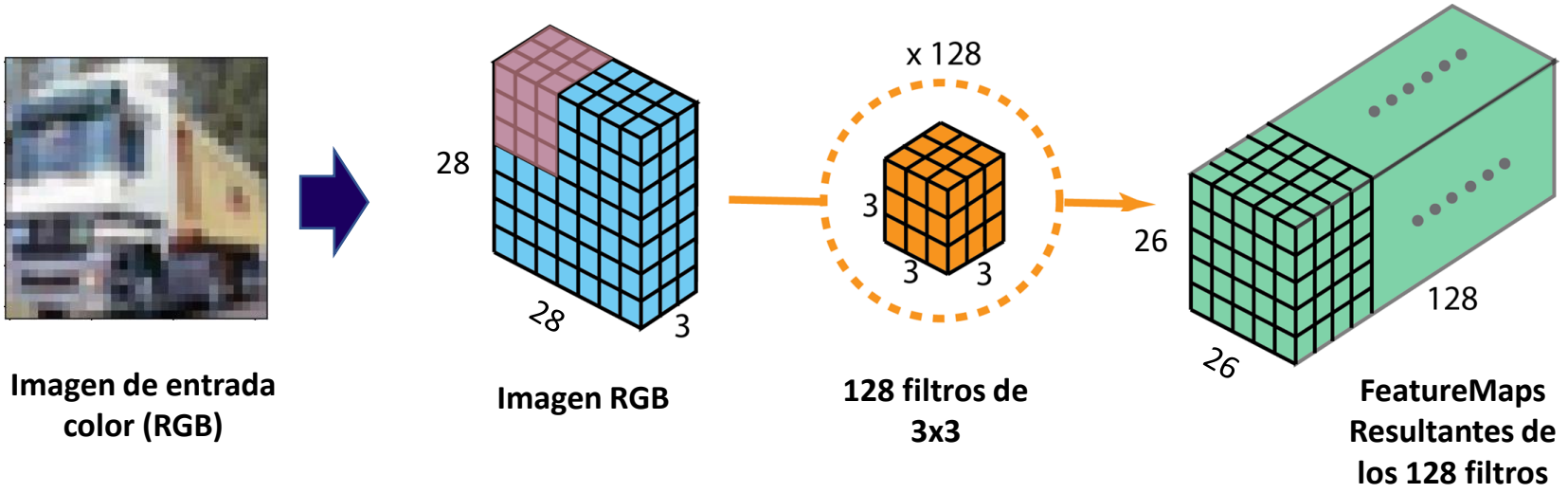


Imagen de entrada color (RGB)

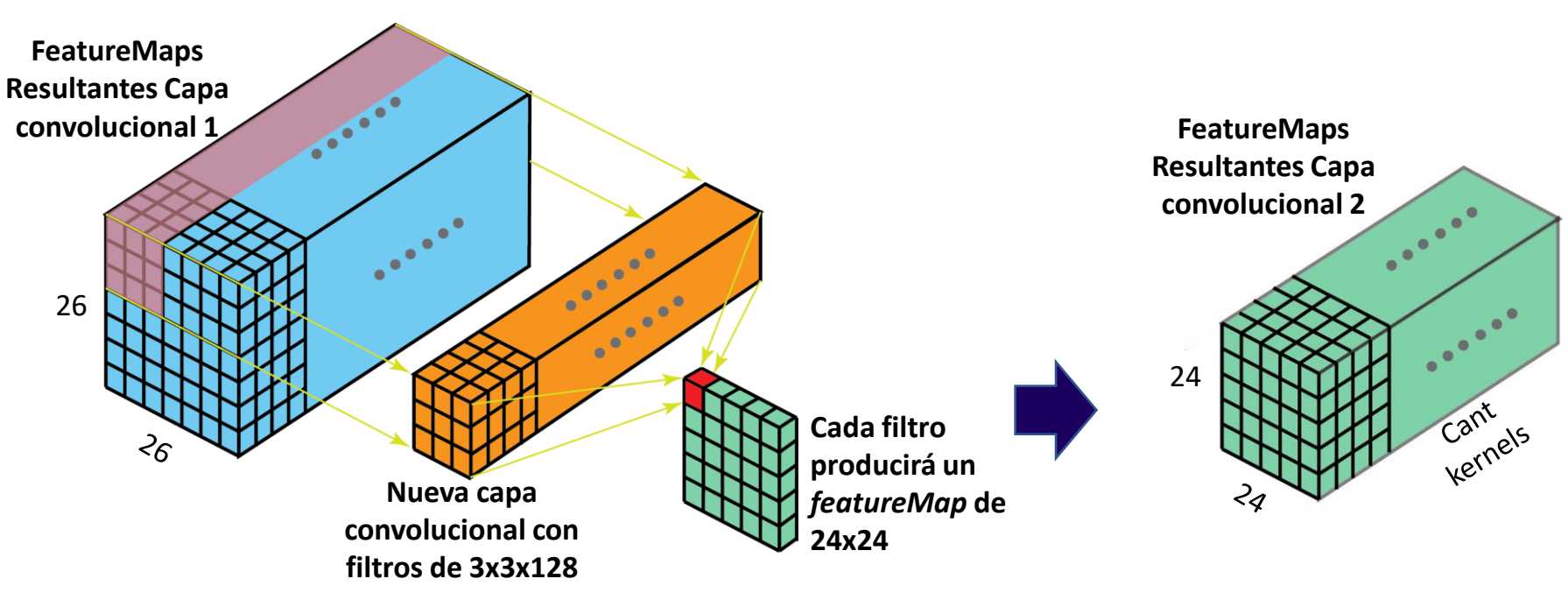
Imagen RGB

128 filtros de 3x3

FeatureMaps Resultantes de los 128 filtros

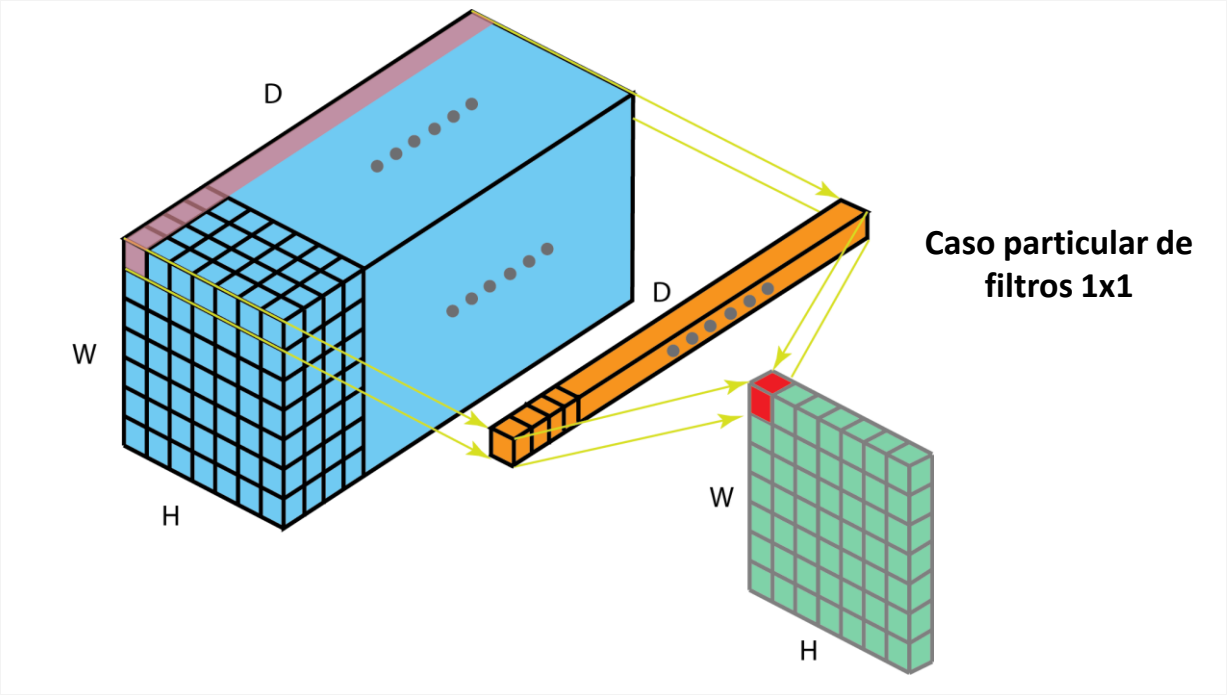
Capas convolucionales

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



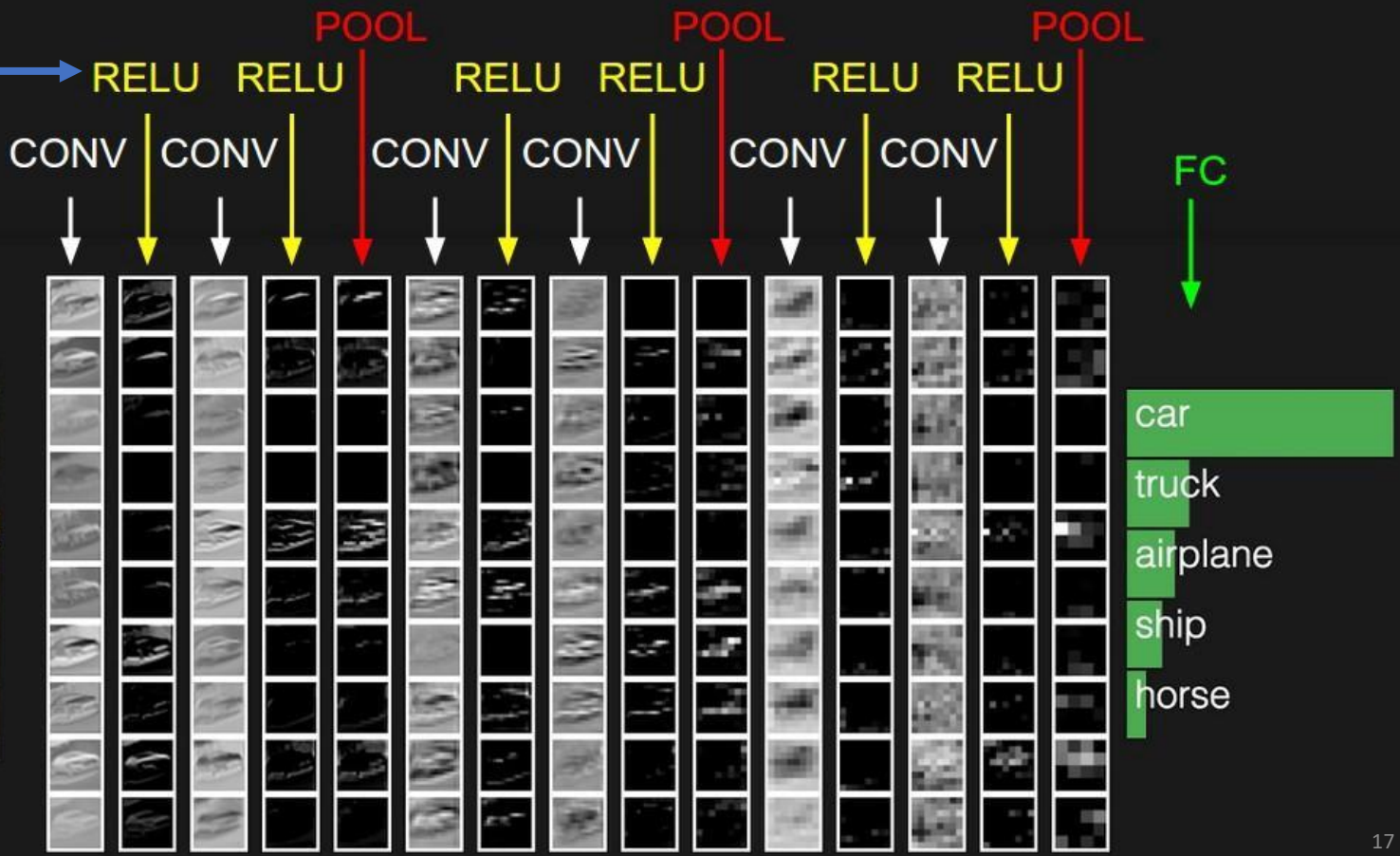
Kernel sizes

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



Red Convolucional estándar

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



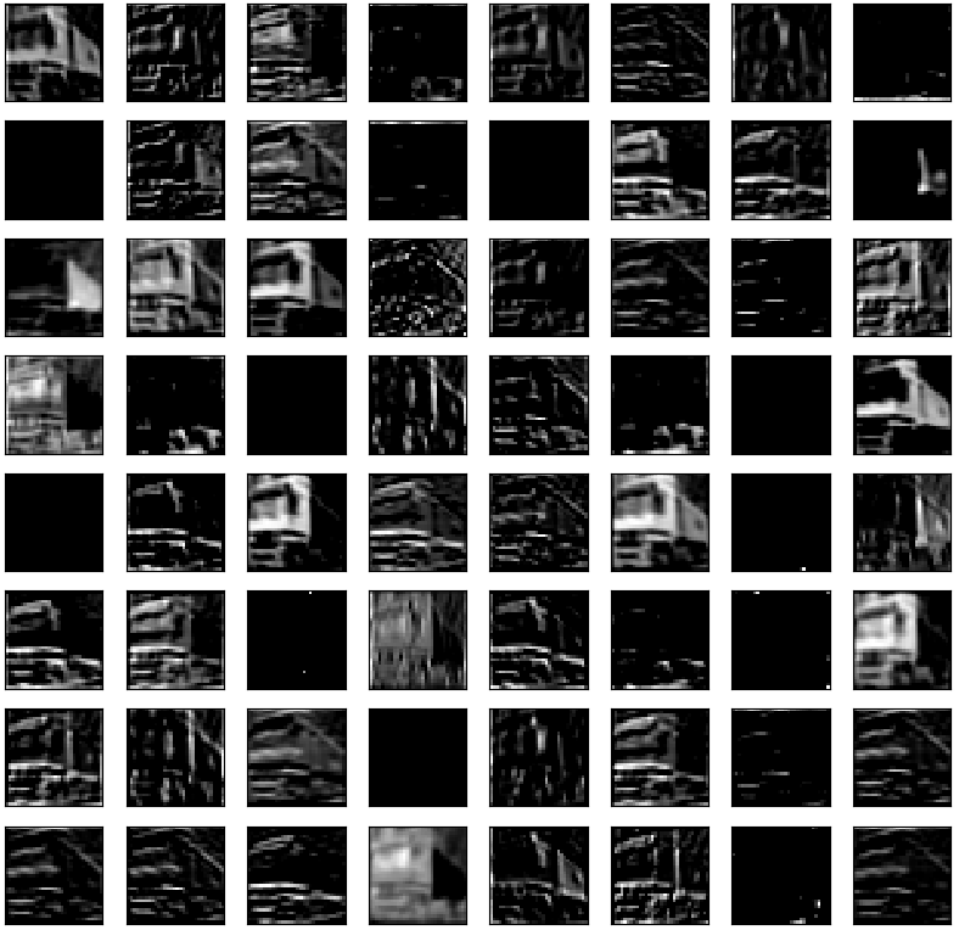
Varias capas sobre CIFAR10

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

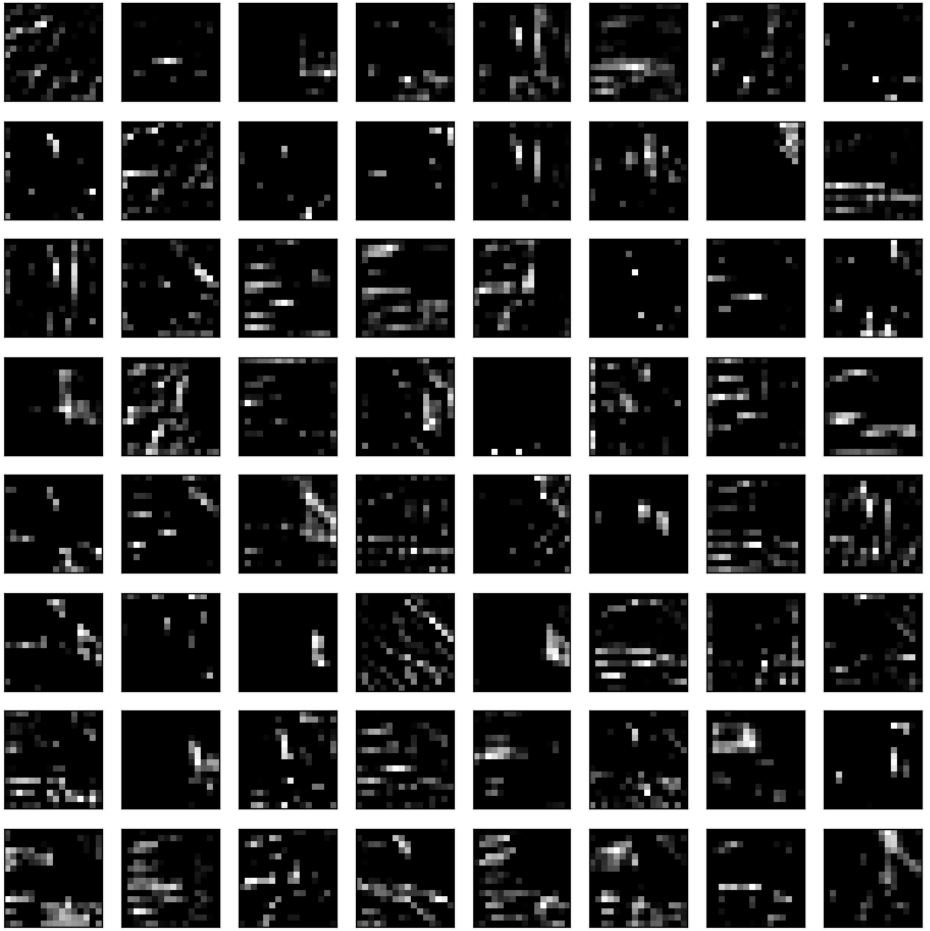
Después de Max Pooling 2x2

Feature Map= 64x16x16

Varias capas sobre CIFAR10

Después de segunda capa convolucional.

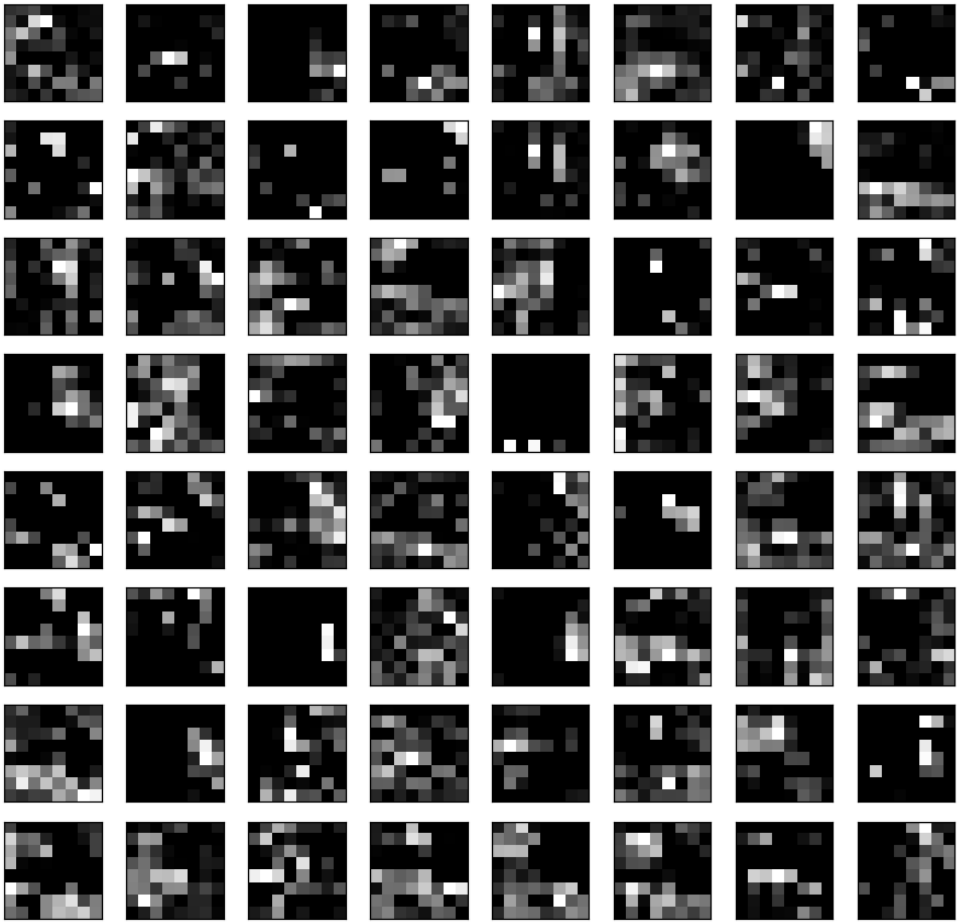
Feature Map= 64x16x16



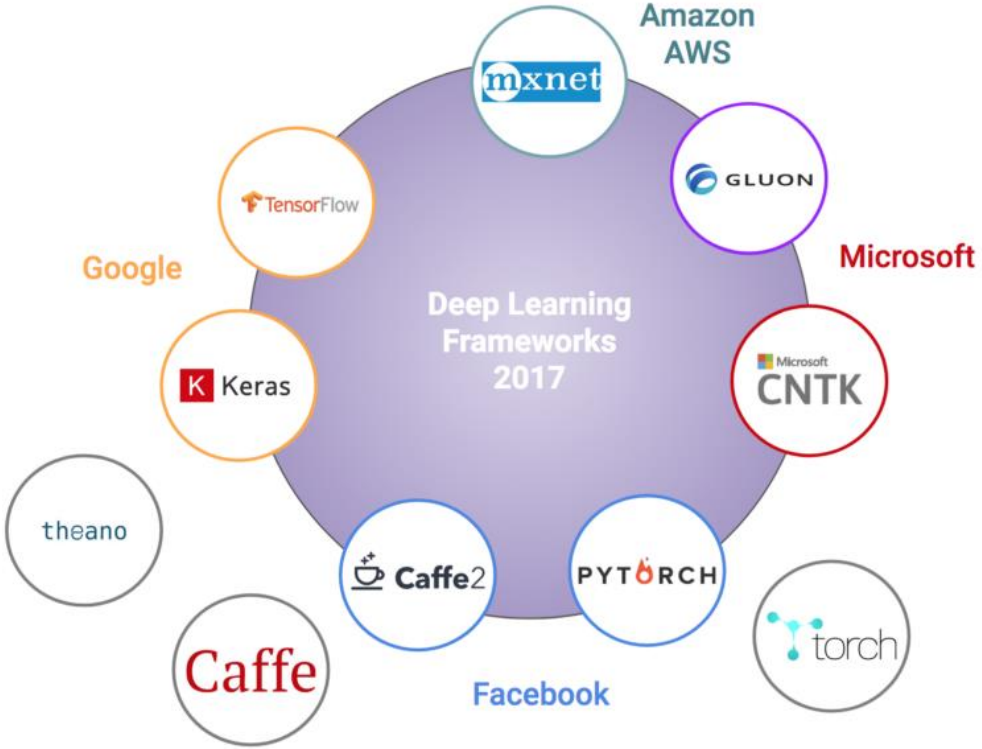
Varias capas sobre CIFAR10

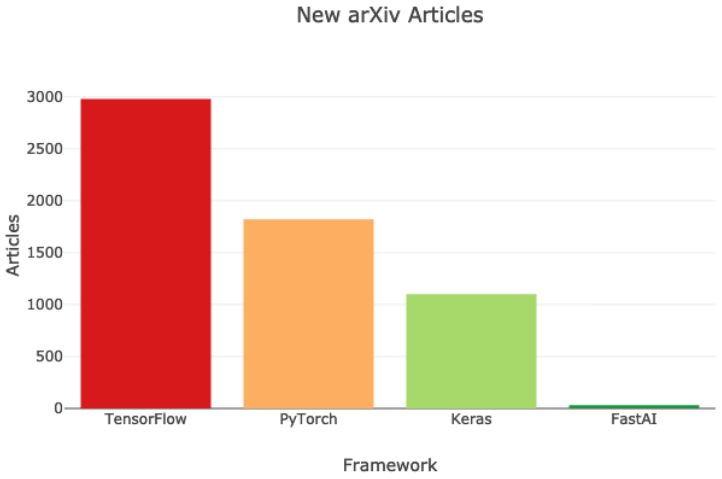
Después de Max Pooling 2x2

Feature Map= 64x8x8



Frameworks/Librerías





python

C++ Ftn R

Modelos actuales - VGG

```
Layer (type)                 Output Shape              Param #
=================================================================
block1
block1
block1
block2
block2
block2
block3
block3
block3
block4
block4
block4
block5
block5
block5
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
flatten_1 (Flatten)          (None, 512)               0
dense_1 (Dense)              (None, 4096)              2101248
dense_2 (Dense)              (None, 4096)              16781312
dense_3 (Dense)              (None, 10)                40970
Total params: 33,638,218
```

Visual Geometry Group – Oxford

- Ganador del ILSVRC 2012
- Muchas convoluciones 3x3
- Diseño en bloques
- La más popular es la versión de 16 capas

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Modelos actuales - AllConvolutional

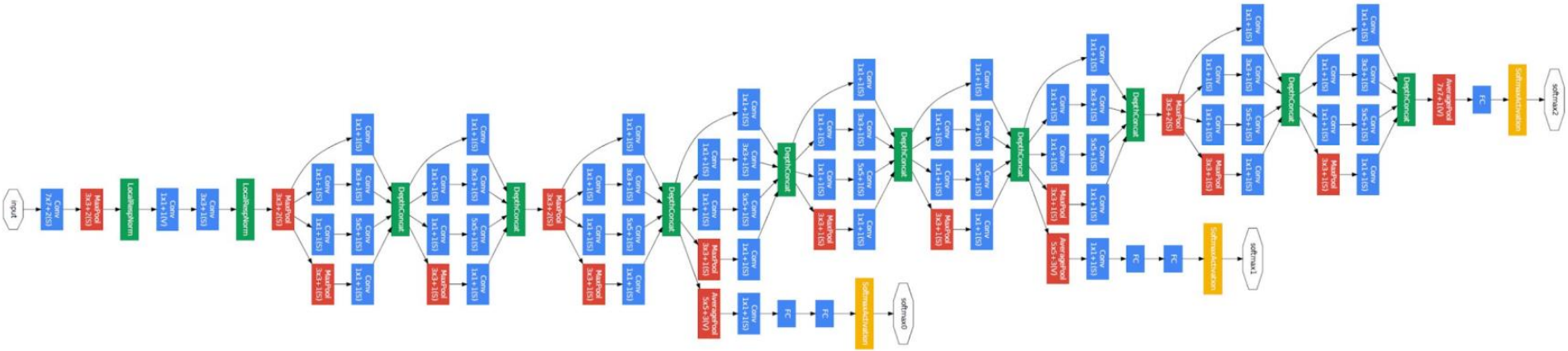
- No hay capas lineales (densas), sólo convoluciones.
- Tiene una capa especial llamada “global average pooling”.
- Más pequeña que VGG.
- Fue diseñada para Cifar10, que es un dataset mas chico que ImageNet.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 32, 32, 96)        2688
conv2d_20 (Conv2D)           (None, 32, 32, 96)        83040
conv2d_21 (Conv2D)           (None, 16, 16, 96)        83040
conv2d_22 (Conv2D)           (None, 16, 16, 192)       166080
conv2d_23 (Conv2D)           (None, 16, 16, 192)       331968
conv2d_24 (Conv2D)           (None, 8, 8, 192)         331968
conv2d_25 (Conv2D)           (None, 8, 8, 192)         331968
conv2d_26 (Conv2D)           (None, 8, 8, 192)         37056
conv2d_27 (Conv2D)           (None, 8, 8, 10)          1930
global_average_pooling2d_1 ( (None, 10)                0
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 1,369,738
```

Modelos actuales - Inception

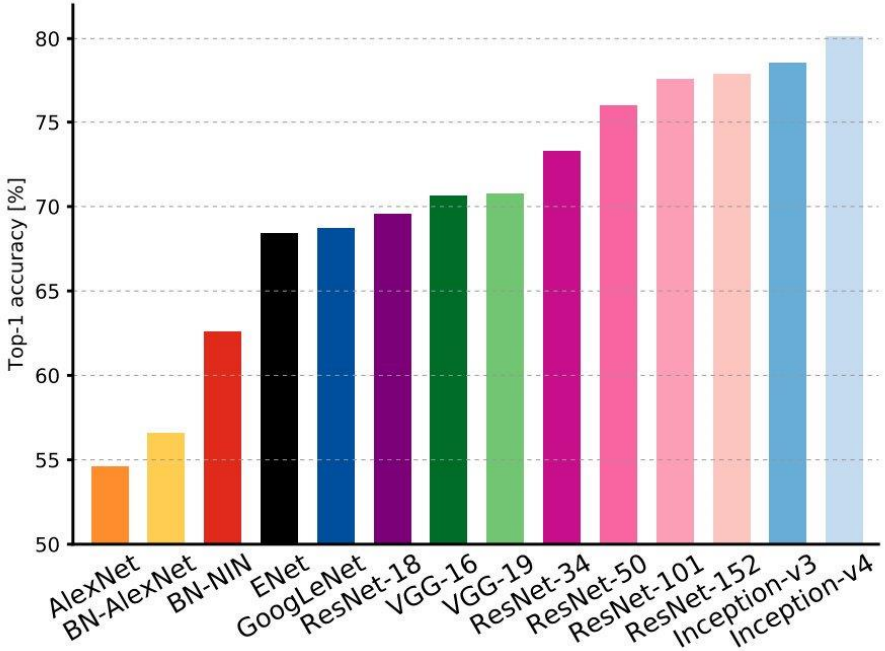
Creado por Google

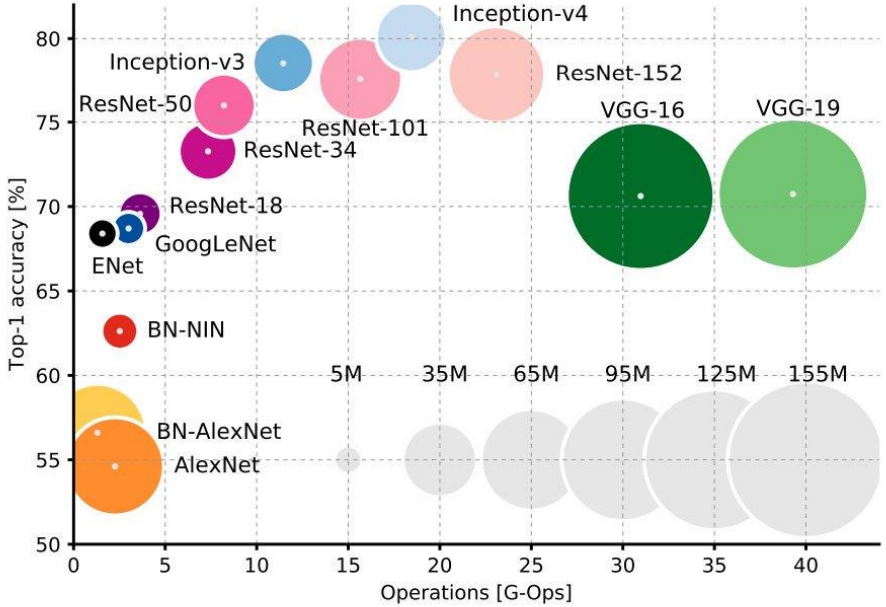
- Ganador del ILSVRC 2014
- Introdujo bloques Inception (en vez de ser más profunda, la red es más ancha).



Modelos actuales

Resumen

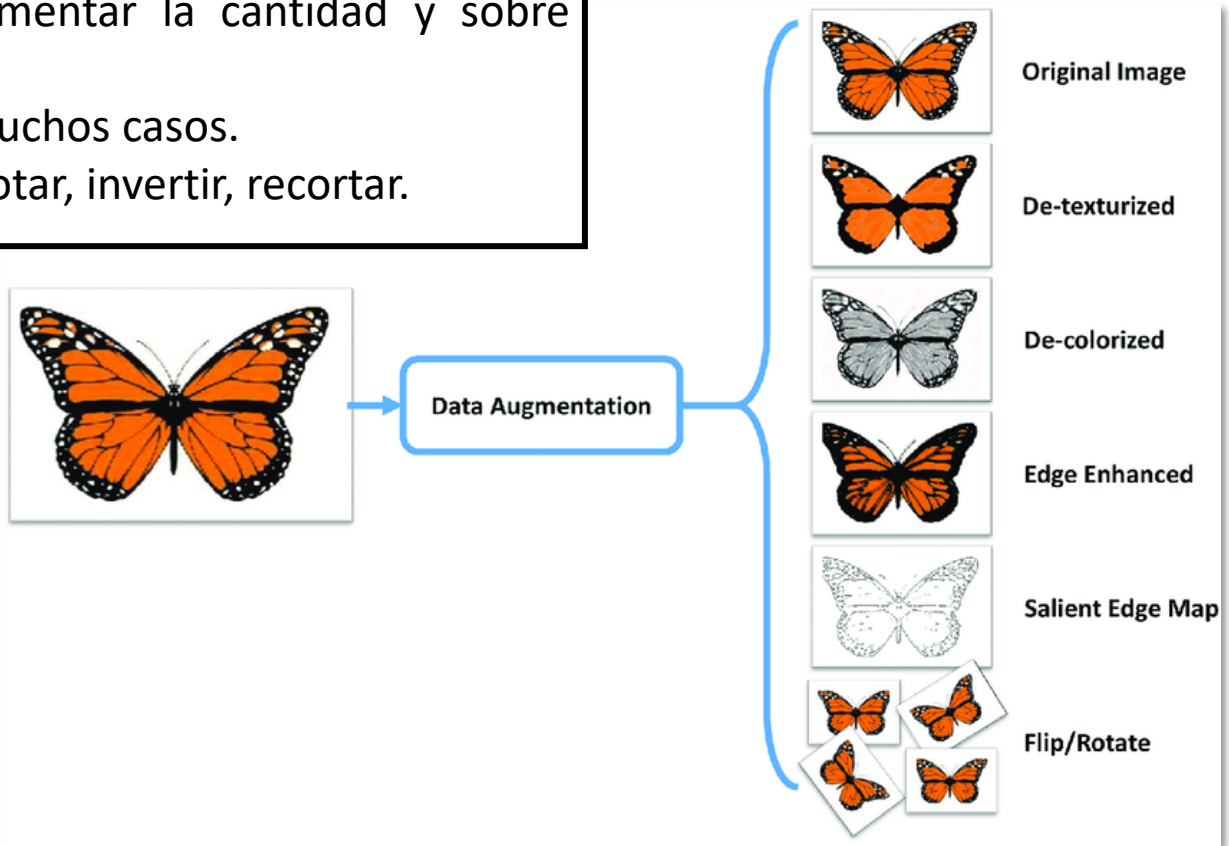




Data augmentation

Técnica utilizada para aumentar la cantidad y sobre todo la variedad de datos

- Baja 1-5% el error en muchos casos.
- Las más comunes son rotar, invertir, recortar.



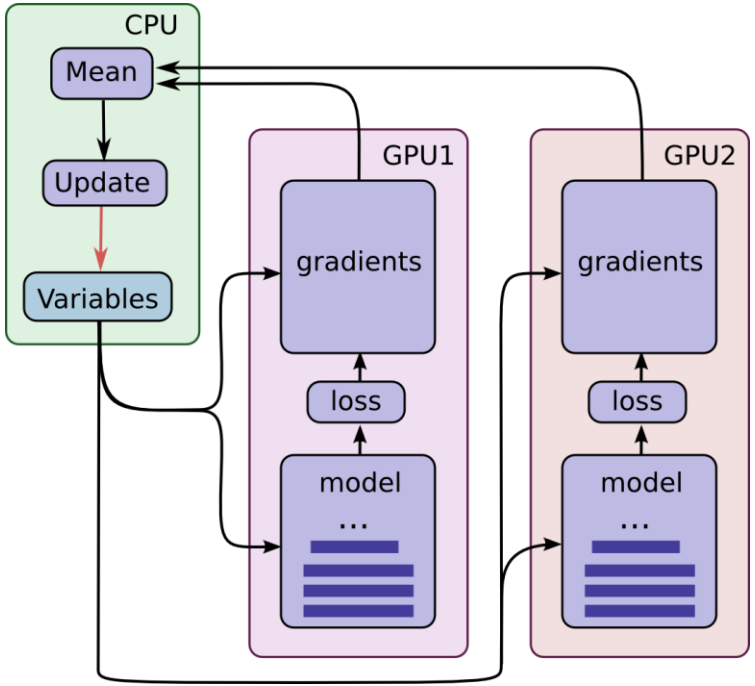
Deep Learning – CPU/GPU

Al tener procesamiento masivo en paralelo, lo usual es utilizar una GPU para llevar a cabo los entrenamientos de las redes.

Nvidia GTX TITAN Z

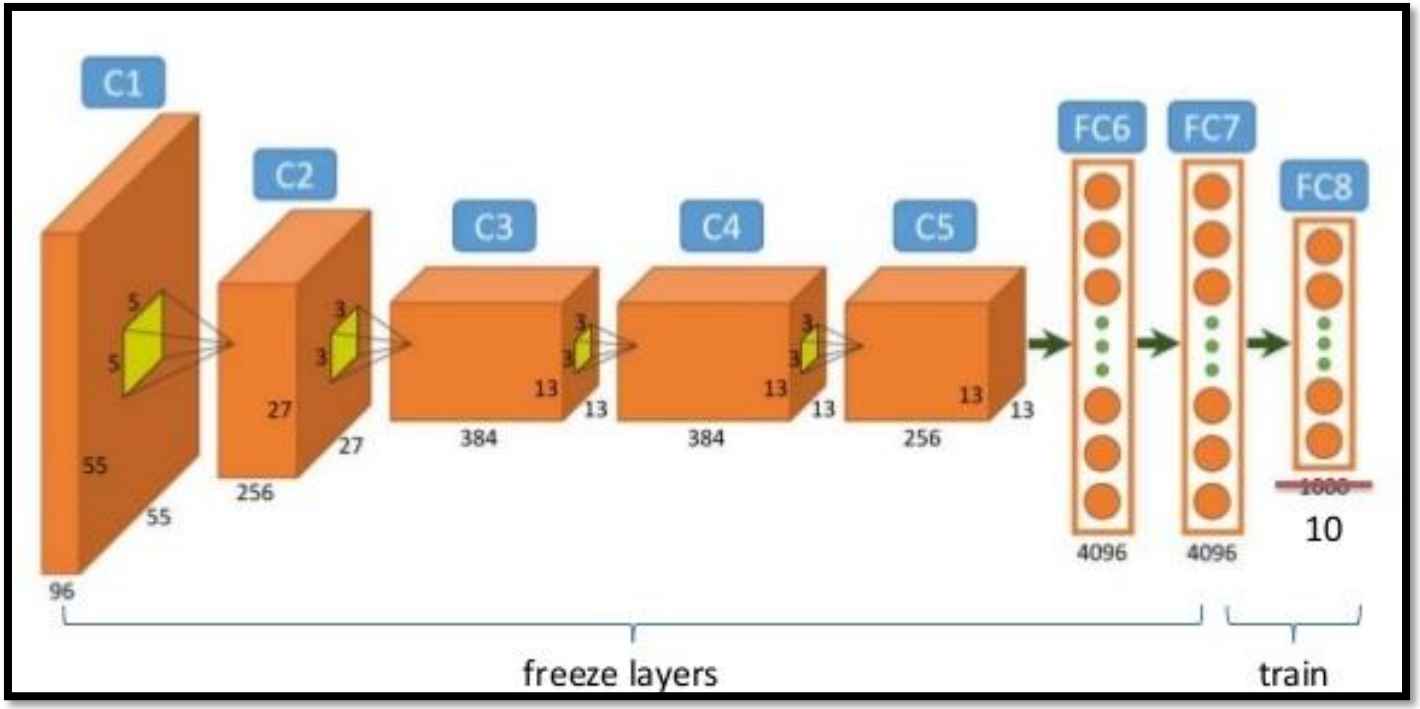
- 12 Gb Gddr5 – interfaz de 384 Bit posee un ancho de banda de 336.5 GB / s
- 3072 procesadores.
- Pci Express 3.0 X16, 7.0 GB/s
- Consumo 250 Watt
- 6,691 gflops

U$S 1.100 en Amazon



Transfer Learning

Un concepto muy usado hoy en día es el de utilizar redes pre-entrenadas en otro dominio. Esto ahorra mucho tiempo de entrenamiento con resultados muy buenos.



Recursos Deep Learning / ML

Libros:

- Libro gratis de Michael Nielsen: https://neuralnetworksanddeeplearning.com/
- Libro gratis de Jason Brownlee: https://machinelearningmastery.com/

Noticias, tendencias, cursos:

- https://deepai.org/
- https://www.deeplearning.ai/ cursos carreras
- https://www.latinxinai.org/
- https://openai.com/
- https://www.xataka.com/
- https://www.kdnuggets.com/

Canales de youtube:

- Dot CSV (español): https://www.youtube.com/channel/UCy5znSnfMsDwaLlROnZ7Qbg
- Henry AI Labs: https://www.youtube.com/channel/UCHB9VepY6kYvZjj0Bgxnpbw
- Two Minute Papers: https://www.youtube.com/user/keeroyz
- Yannic Kilcher: https://www.youtube.com/channel/UCZHmQk67mSJgfCCTn7xBfew
- Zak Jost: https://www.youtube.com/channel/UCxw9_WYmLqlj5PyXu2AWU_g/videos