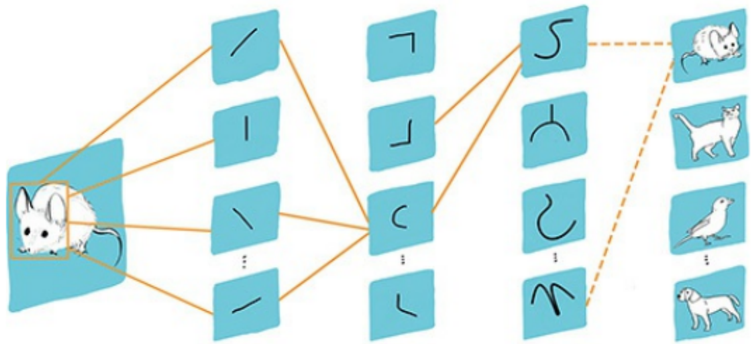
dcc CIENCIAS DE LA COMPUTACIÓN UNIVERSIDAD DE CHILE

Universidad de los Andes > FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Redes Neuronales Convolucionales

[convolutional neural networks]



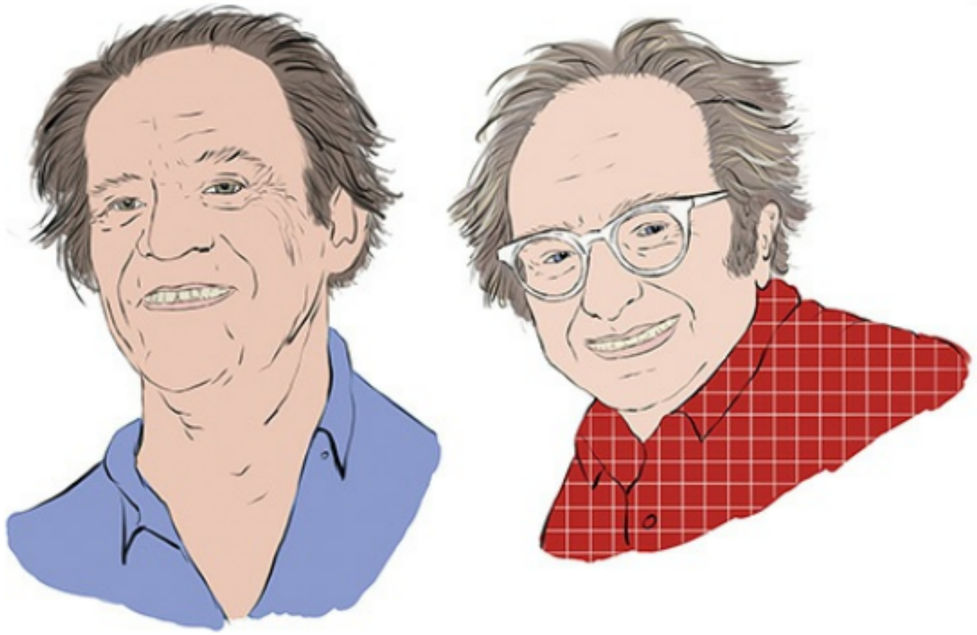
CLASE 2.2

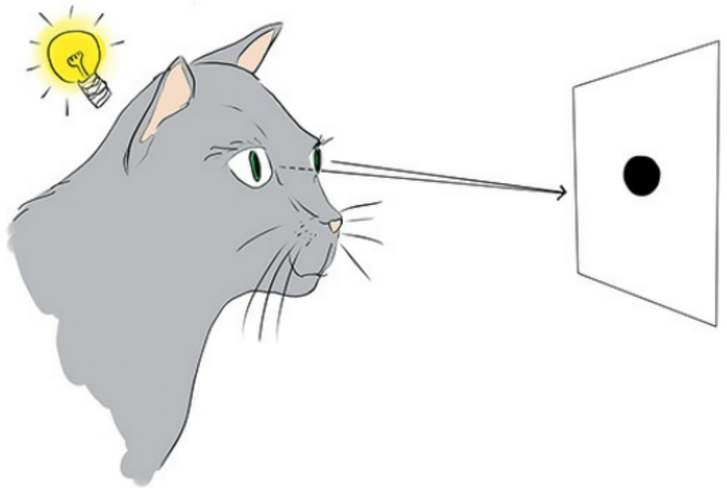
José M. Saavedra R.
Profesor Asistente

jmsaavedrar@miuandes.cl

Ed. Ingeniería - Oficina 315

Percepción Visual





https://www.youtube.com/watch?v=IOHayh06LJ4

T. Wiesel & D. Huebel y el experimento del "gato"

Visual Perception

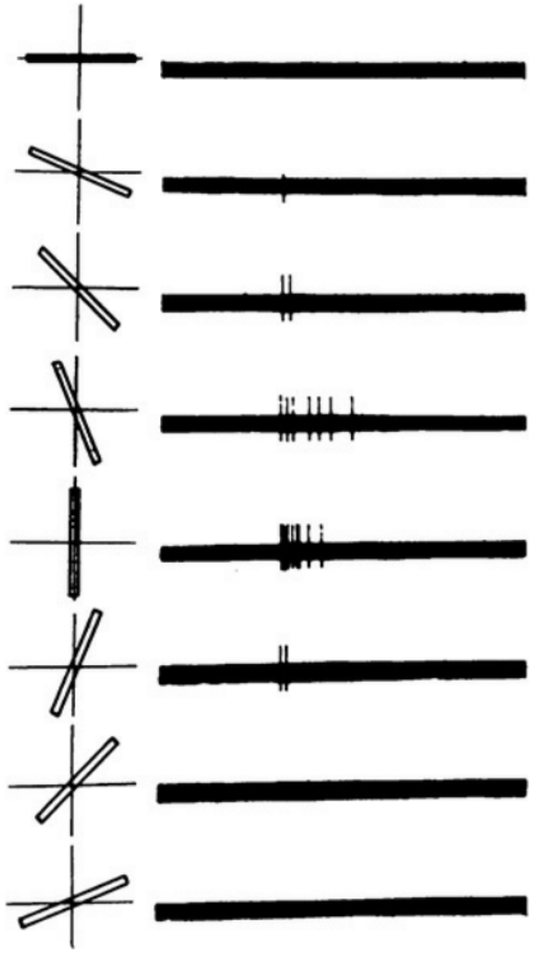
Fuente: Deep Learning Illustrated
https://www.deeplearningillustrated.com/

Percepción Visual

T. Wiesel & D. Huebel

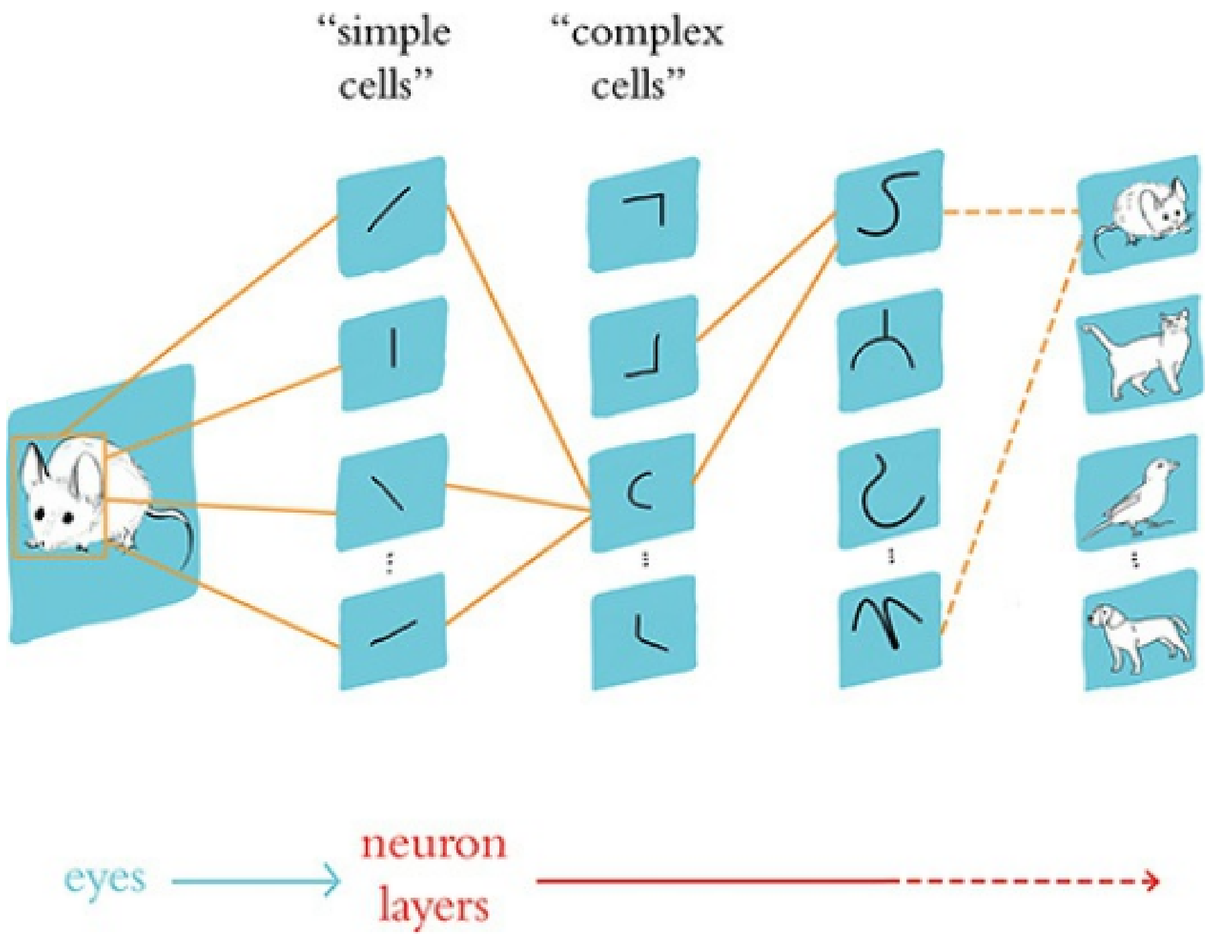
Visual Perception

Una célula simple en la corteza visual primaria de un gato responde con diferente intesidad dependiendo de la orientación de un borde.



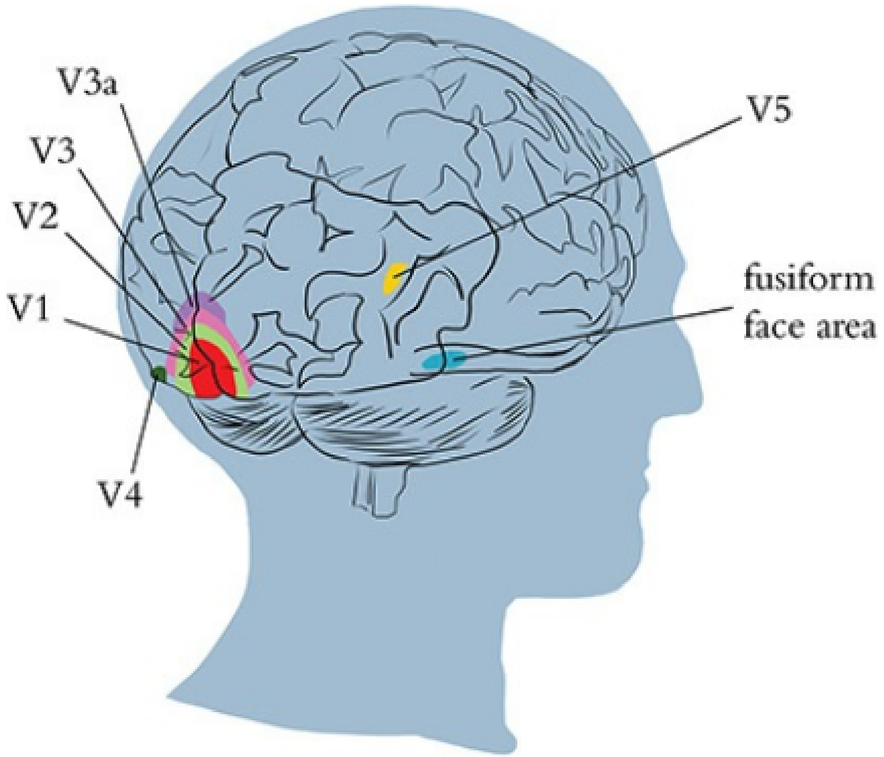
Fuente: Deep Learning Illustrated
https://www.deeplearningillustrated.com/

Percepción Visual y Redes Convolucionales



Fuente: Deep Learning Illustrated
https://www.deeplearningillustrated.com/

Percepción Visual y Redes Convolucionales



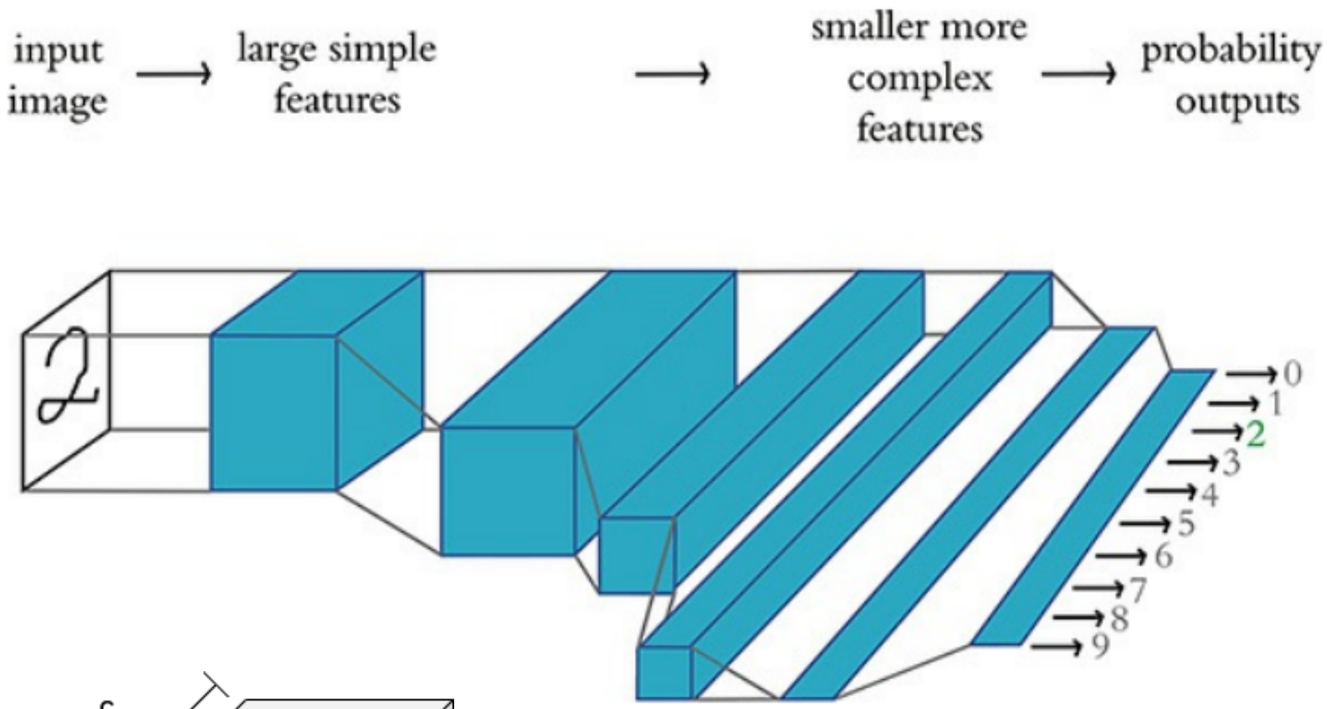
Visual Perception

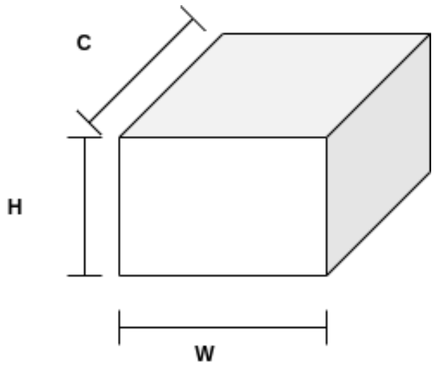
Regiones de la corteza visual. La región V1, recibe información de bajo nivel desde la retina y contiene células simples capaces de detectar bordes orientados.

A través de la recombinación de información, vía un gran número de capas de neuronas subsecuentes , una mayor abstracción es representada (mayor semántica).

Fuente: Deep Learning Illustrated
https://www.deeplearningillustrated.com/

Redes Convolucionales





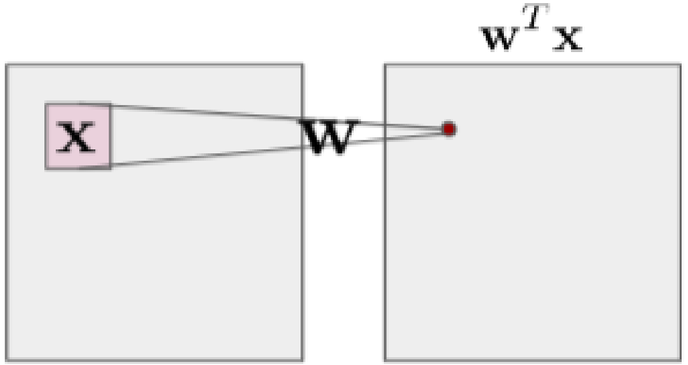
Aquí se muestra una simple arquitectura de red convolucional (CNN) compuesta por capas convolucionales (al inicio) y capas densas (al final).

En el proceso (fordward) cada capa recibe un tensor (generalmente de 3 dimensiones) y devuelve otro tensor con la misma dimensionalidad.

- Un tensor T es representado por 3 dimensiones : H x W x C
- Cuando procesamos en batch (SGD) añadimos una cuarta dimensión al inicio así, el tensor T queda representado como B x H x W x C

Redes Convolucionales

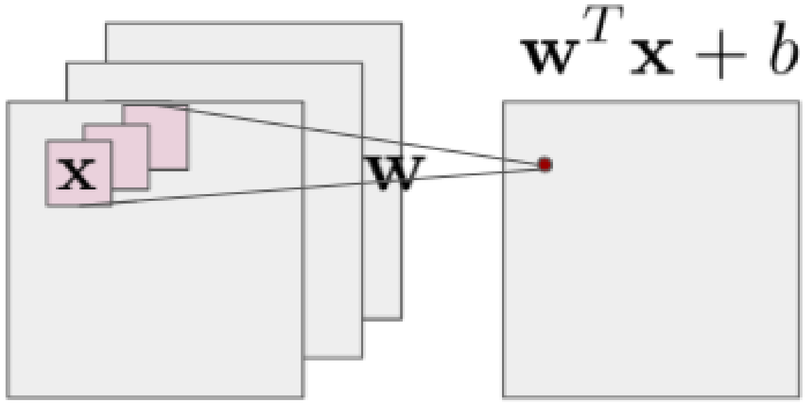
Convolutional Layer



regular convolution
[correlation]

Filter size: $k \times k$

Here, we suppose a square filter, but we can also use a non-square one.



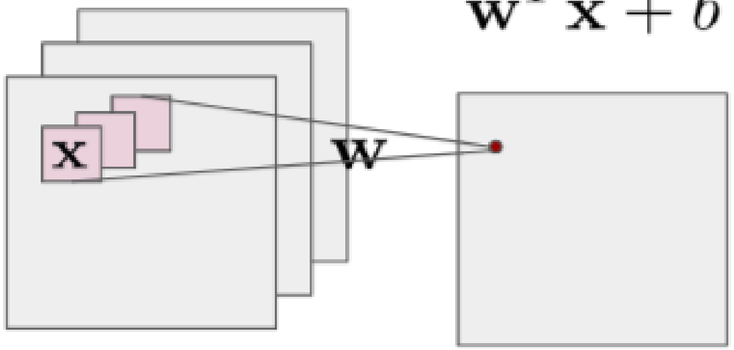
convolution layer

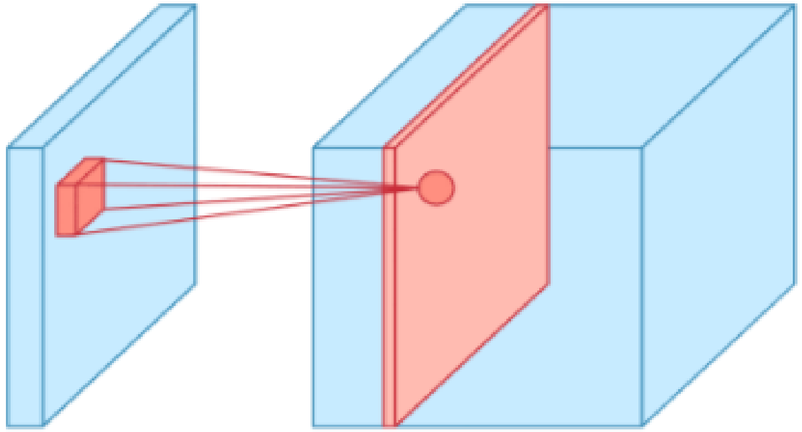
Filter size: $n_{in} \times k \times k$

number of channels in the input layer

Redes Convolucionales

Convolutional Layer

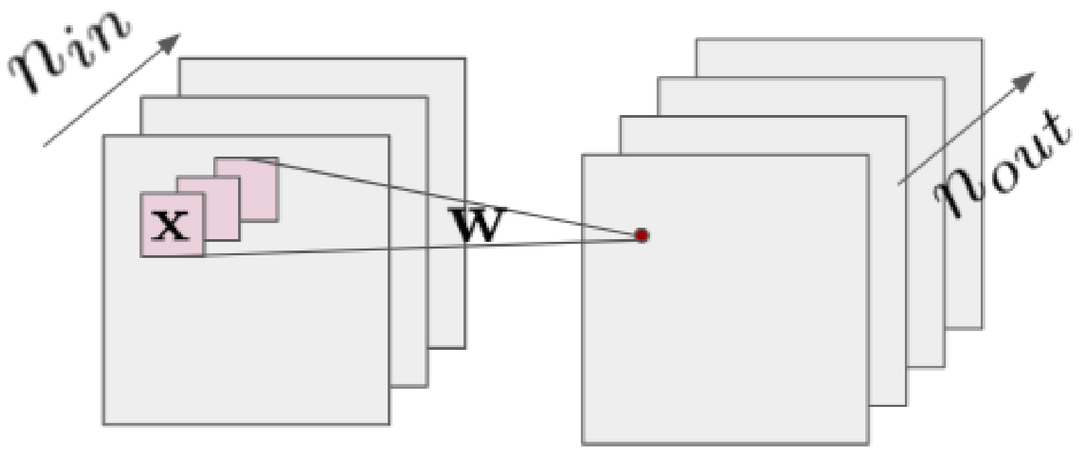


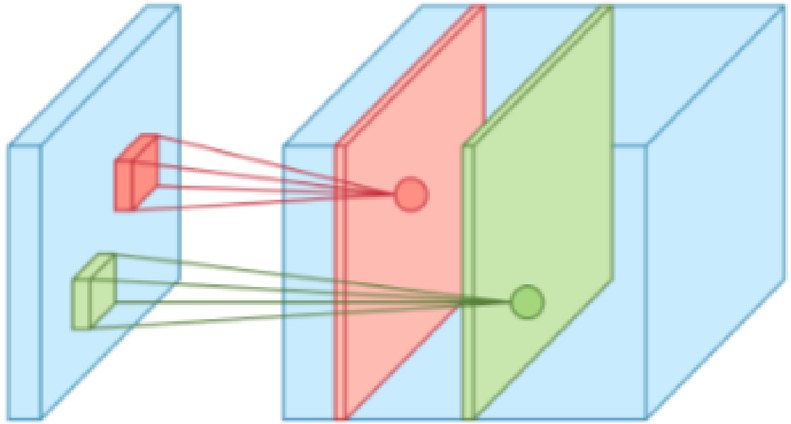


Number of learnable parameters $n_{in} \times k \times k + 1$

Redes Convolucionales

Convolutional Layer





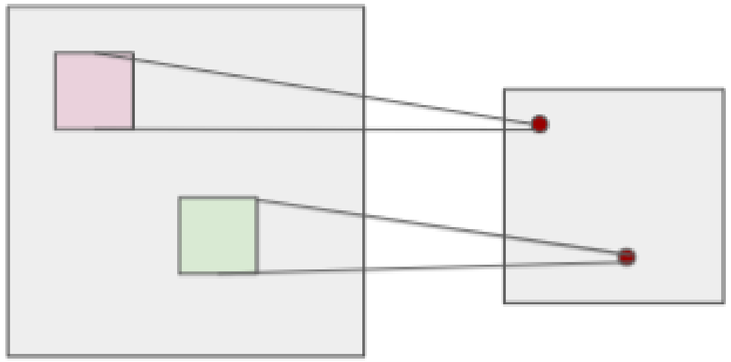
$$\mathbf{w}^T\mathbf{x} + b$$

number of channels in the output layer

Number of learnable parameters $(n_{in} \times k \times k + 1) \times n_{out}$

Redes Convolucionales

Pooling Layer



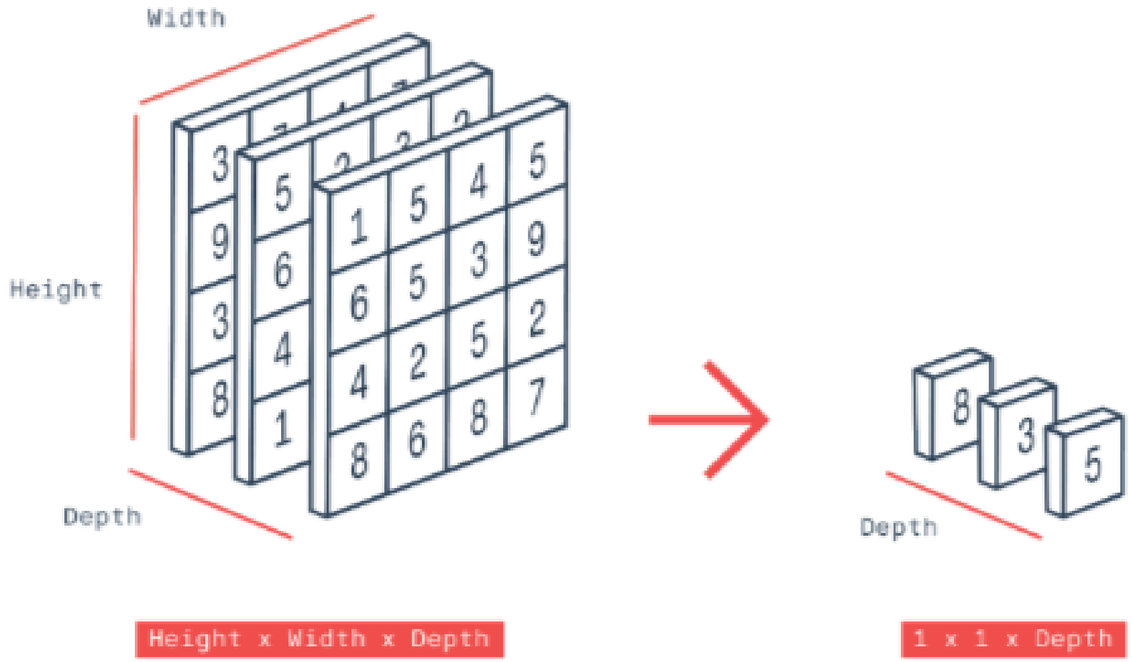
pooling = It locally aggregates the network's responses.

Parameters (Hyperparameters):

- Pooling Type : AVG, MAX
- Regions Size (kernel size) : Size of the local region
- Stride: It will affect the output size. A stride = 2 will produce an output reduced to half (spatially).
- No learnable parameters

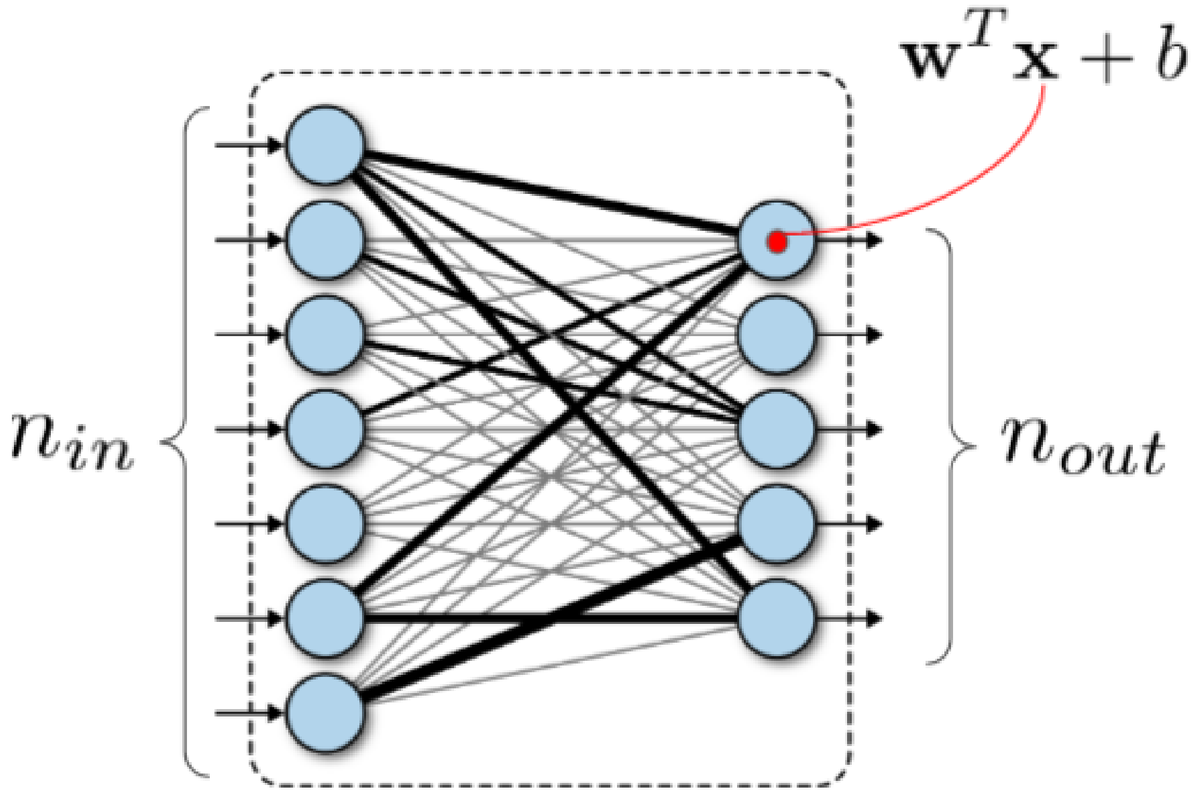
Redes Convolucionales

Global Average Pooling (GAP)



Redes Convolucionales

Fully-Connected Layer (Dense Layer)



Parameters :

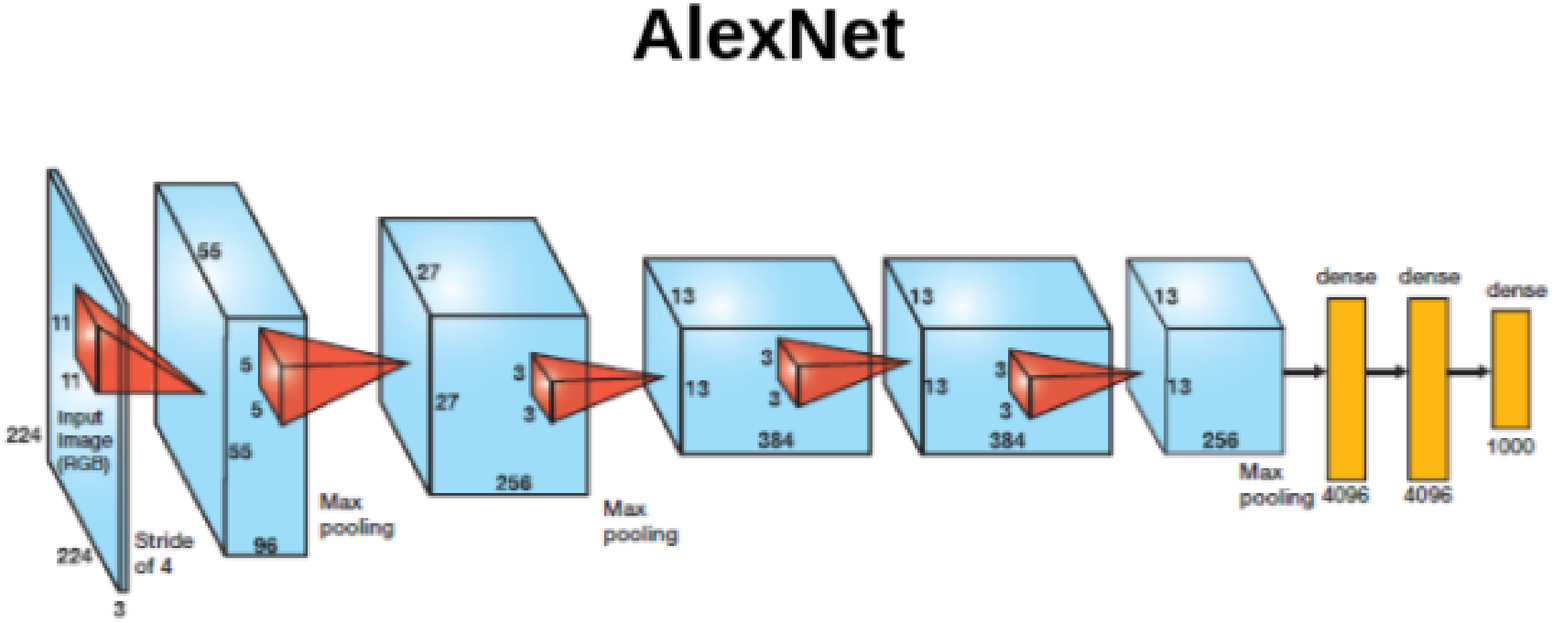
- **Size :** Number of neurons n_{out}

Number of learnable parameters:

$$(n_{in} + 1) \times n_{out}$$

Redes Convolucionales

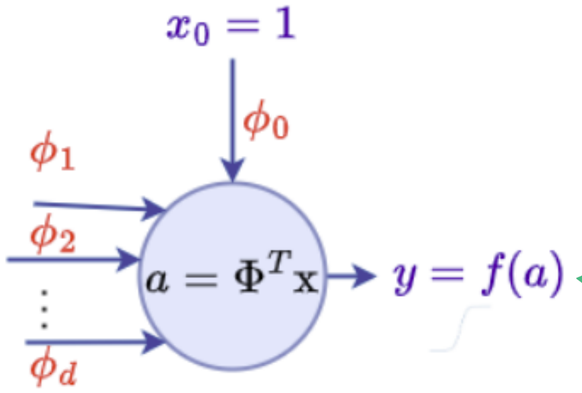
Colocando todos lo componentes en acción



ImageNet Classification with Deep Convolutional Neural Networks
[Alex Krizhevsky 2012]

Redes Convolucionales

Para generar un modelo no-lineal y favorecer la convergencia, se agregan funciones de activación al final de cada capa (sigmoid, ReLU, leakyReLU, softplut, etc.).



función de activación

Name	Plot	Function, $g(x)$	Derivative of g, $g'(x)$	Range	Order of continuity
Identity		x	1	$(-\infty, \infty)$	C^∞
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	0	$\{0, 1\}$	C^{-1}
Logistic, sigmoid, or soft step		$\sigma(x) \doteq \frac{1}{1+e^{-x}}$	$g(x)(1-g(x))$	$(0, 1)$	C^∞
Hyperbolic tangent (tanh)		$\tanh(x) \doteq \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - g(x)^2$	$(-1, 1)$	C^∞
Rectified linear unit (ReLU)[8]		$(x)^+ \doteq \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max(0, x) = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$	C^0
Gaussian Error Linear Unit (GELU)[5]		$\frac{1}{2}x\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$ $= x\Phi(x)$	$\Phi(x) + x\phi(x)$	$(-0.17\ldots, \infty)$	C^∞
Softplus[9]		$\ln(1 + e^x)$	$\frac{1}{1+e^{-x}}$	$(0, \infty)$	C^∞
Exponential linear unit (ELU)[10]		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter α	$\begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \text{ and } \alpha = 1 \end{cases}$	$(-\alpha, \infty)$	$\begin{cases} C^1 & \text{if } \alpha = 1 \\ C^0 & \text{otherwise} \end{cases}$
Scaled exponential linear unit (SELU)[11]		$\lambda\begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameters $\lambda = 1.0507$ and $\alpha = 1.67326$	$\lambda\begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$	C^0
Leaky rectified linear unit (Leaky ReLU)[12]		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0.01 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$(-\infty, \infty)$	C^0
Parametric rectified linear unit (PReLU)[13]		$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameter α	$\begin{cases} \alpha & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0

https://github.com/jmsaavedrar/convnet2

Redes Convolucionales

Normalización de Capas

Normalizar la salida de una capa es tan importante como normalizar la entrada. Recordemos que normalizar la entrada de un modelo significa hacer que las diferentes características tengan el mismo rango de variación. En una red neuronal, las salidas de una capa también pueden mostrar variaciones que afectan el entrenamiento. Para manejar tales variaciones se han propuesto estrategias que permiten normalizar las salidas de una capa de una red neuronal.

En general, la idea es estimar la media μ y la desviación estándar σ, y normalizar los datos bajo la siguiente expresión:

$$\hat{X} = \frac{X - \mu}{\sigma} \tag{3.44}$$

$$\text{norm(X)} = \gamma \hat{X} + \beta \tag{3.45}$$

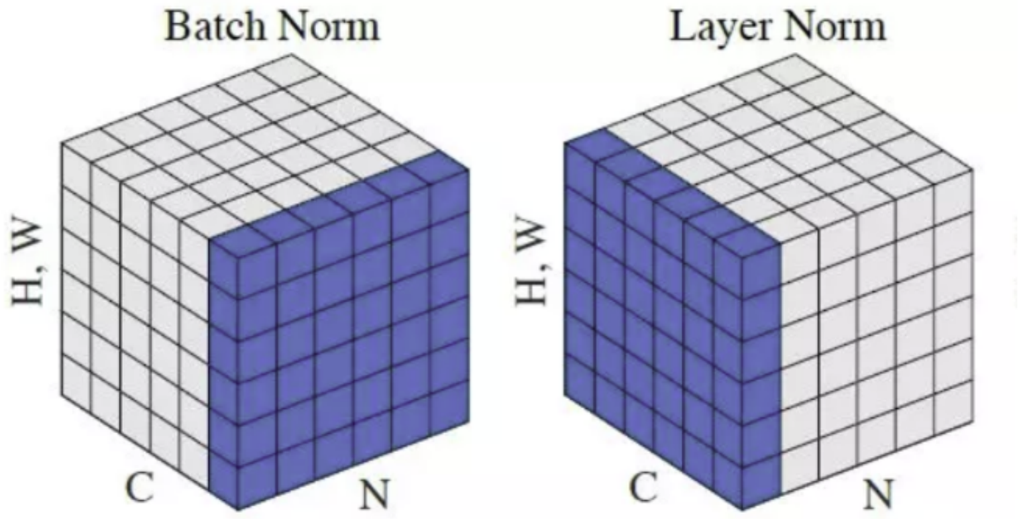
$$\tag{3.46}$$

donde γ y β son parámetros aprendibles, y agregan flexibilidad a la normalización.

Cap 3.5.7

https://github.com/jmsaavedrar/convnet2

Redes Convolucionales



BatchNorm requiere 2 parámetros (γ y β) aprendibles por cada canal de la capa subyacente. Además, es necesario mantener los estadísticos μ y σ para la fase de predicción.

LayerNorm requiere 2 parámetros (γ y β) aprendibles por cada canal de la capa subyacente. A diferencia de BatchNorm, LayerNorm no necesita mantener los estadísticos μ y σ, ya que estos dependen de la entrada.

https://github.com/jmsaavedrar/convnet2

Redes Convolucionales

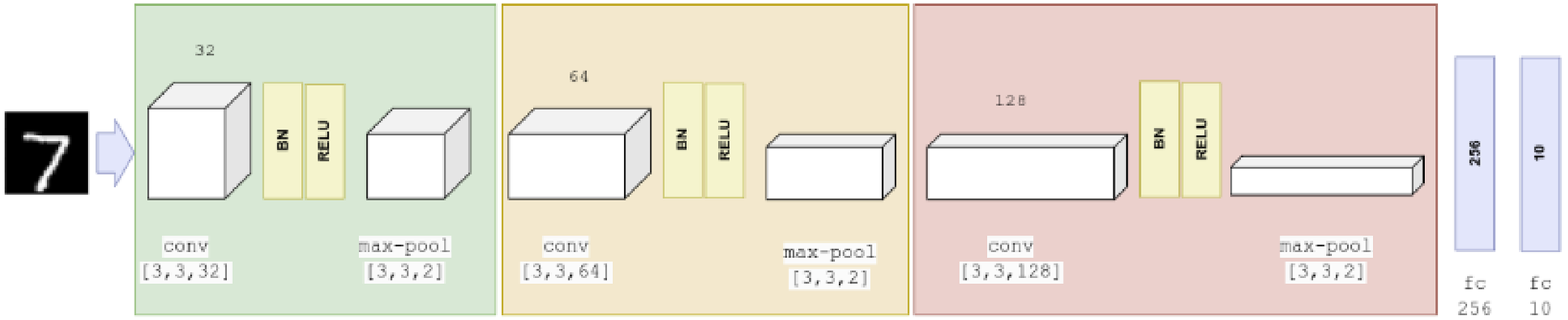


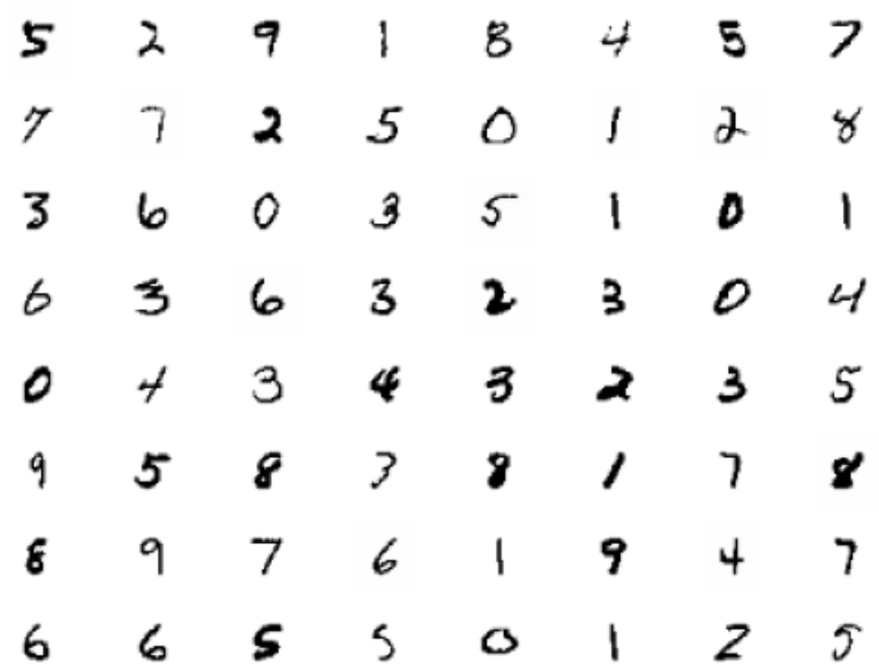
Figure 3.21: Una simple arquitectura convolucional para MNIST.

https://github.com/jmsaavedrar/convnet2

Redes Convolucionales

Práctica

https://github.com/jmsaavedrar/convnet2



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	multiple	320
max_pooling2d (MaxPooling2D)	multiple	0
re_lu (ReLU)	multiple	0
batch_normalization (BatchNo	multiple	128
conv2d_1 (Conv2D)	multiple	18496
batch_normalization_1 (Batch	multiple	256
conv2d_2 (Conv2D)	multiple	73856
batch_normalization_2 (Batch	multiple	512
dense (Dense)	multiple	524544
batch_normalization_3 (Batch	multiple	1024
dense_1 (Dense)	multiple	2570

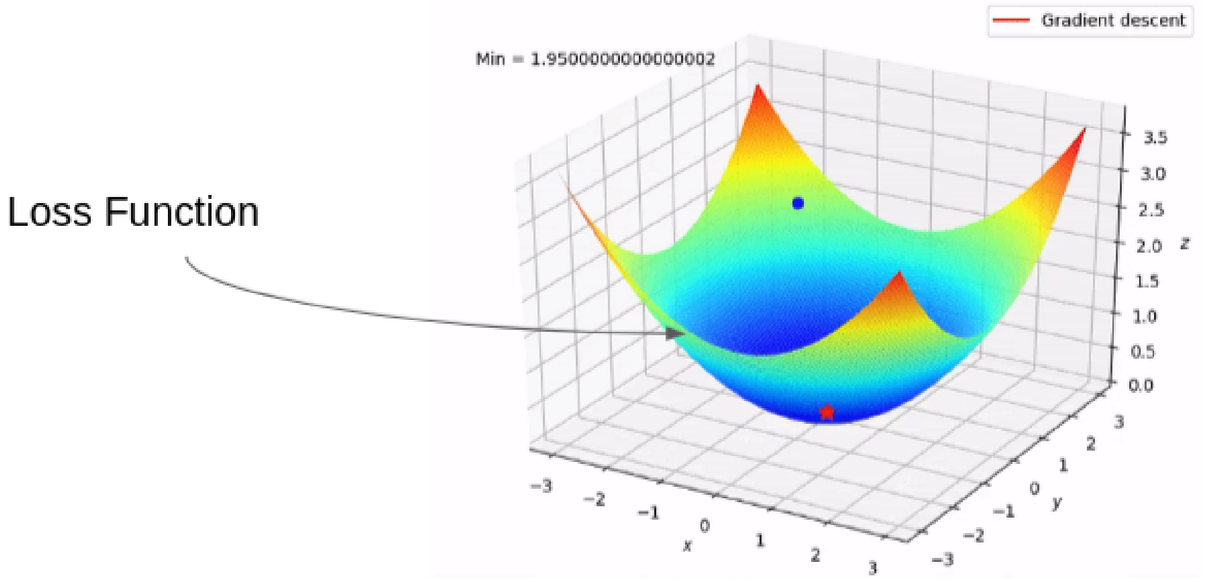
Total params: 621,706
Trainable params: 620,746
Non-trainable params: 960

https://impresee.com/train-your-own-convolutional-network/

Redes Convolucionales

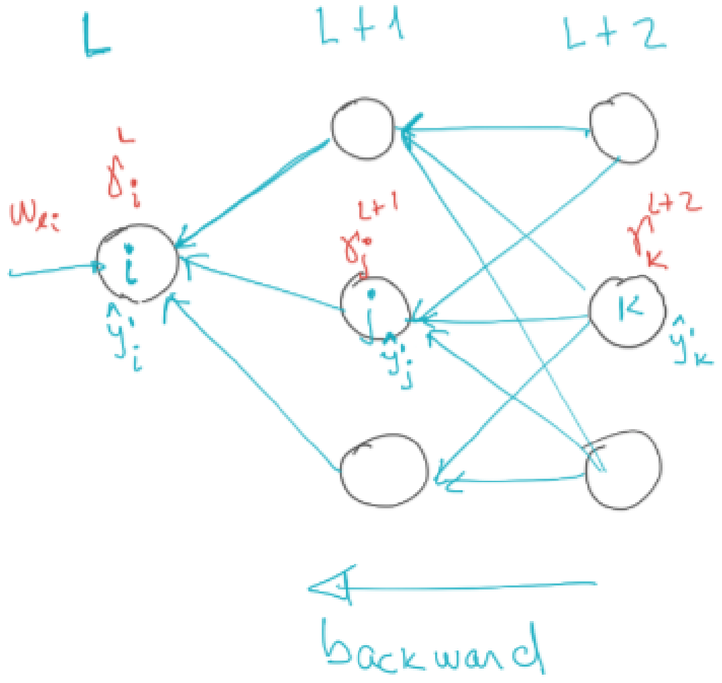
Gradient Descent

$$\mathbf{w} = \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{L}$$



Redes Convolucionales

Vanishing Gradient Problem [backpropagation]



$$\gamma_i^L = \left[\sum_j \gamma_j^{L+1} w_{ij}^{L+1} \right] \hat{y}'^L_i$$

$$\left[\sum_k \gamma_k^{L+2} w_{jk}^{L+2} \right] \hat{y}'^{L+1}_j$$

product of derivatives

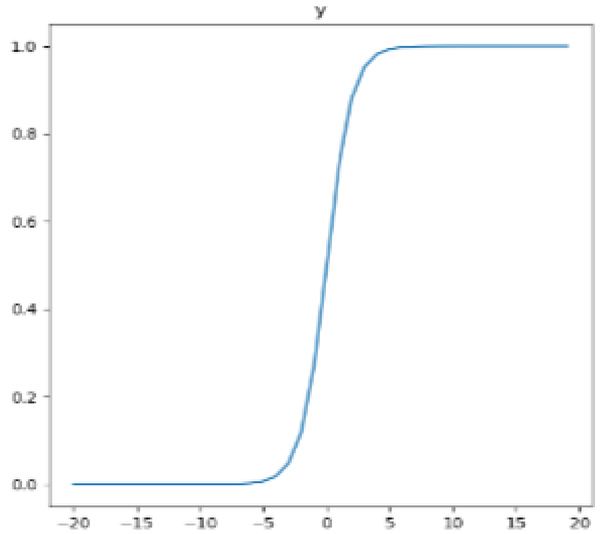
$$\gamma_i^L = \hat{y}'^L_i \, \hat{y}'^{L+1} \, \hat{y}'^{L+2} \ldots$$

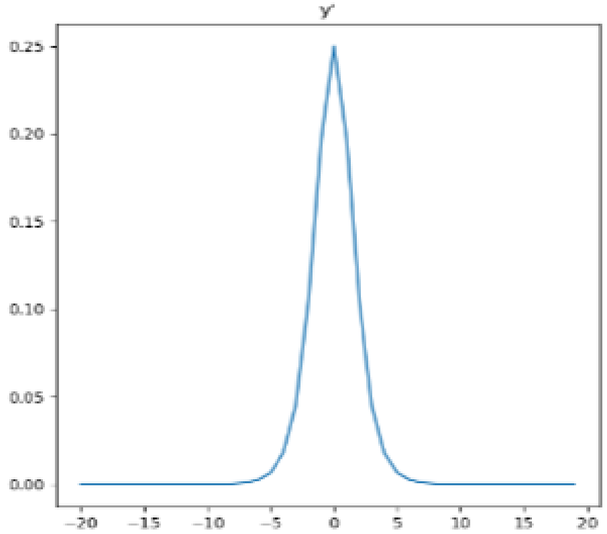
https://www.youtube.com/watch?v=e0dlpYaJ9k0&list=PLUh8MMo7TfsPLI0QtwXMb59lBDsPAYsHk

Redes Convolucionales

Vanishing Gradient Problem

What is the effect of the sigmoid activation function in deep networks?





depth

$$0,25^{10} = 0,000000954$$

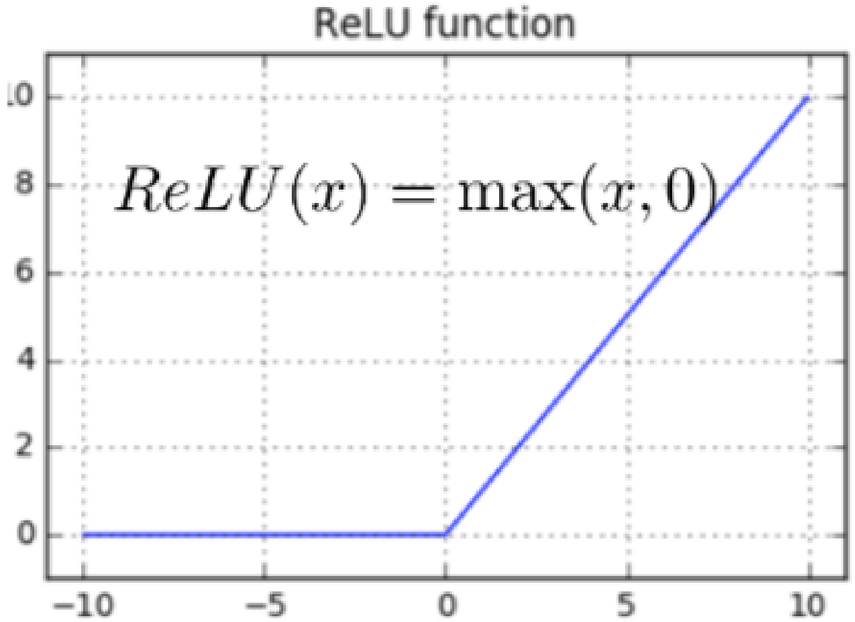
$$0,25^{30} = 8.673617379e-19$$

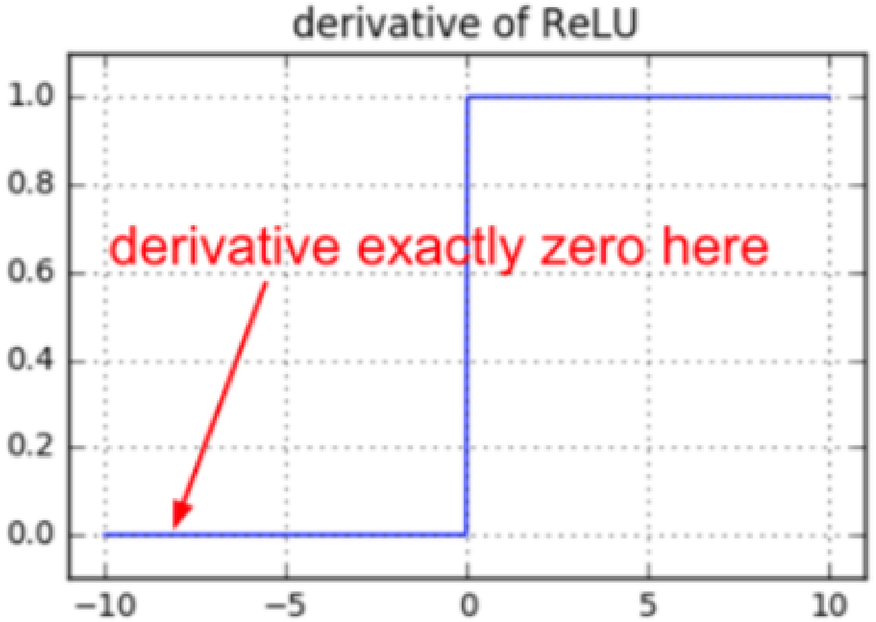
vanishing gradient problem on deep networks

https://www.youtube.com/watch?v=e0dlpYaJ9k0&list=PLUh8MMo7TfsPLI0QtwXMb59lBDsPAYsHk

Redes Convolucionales

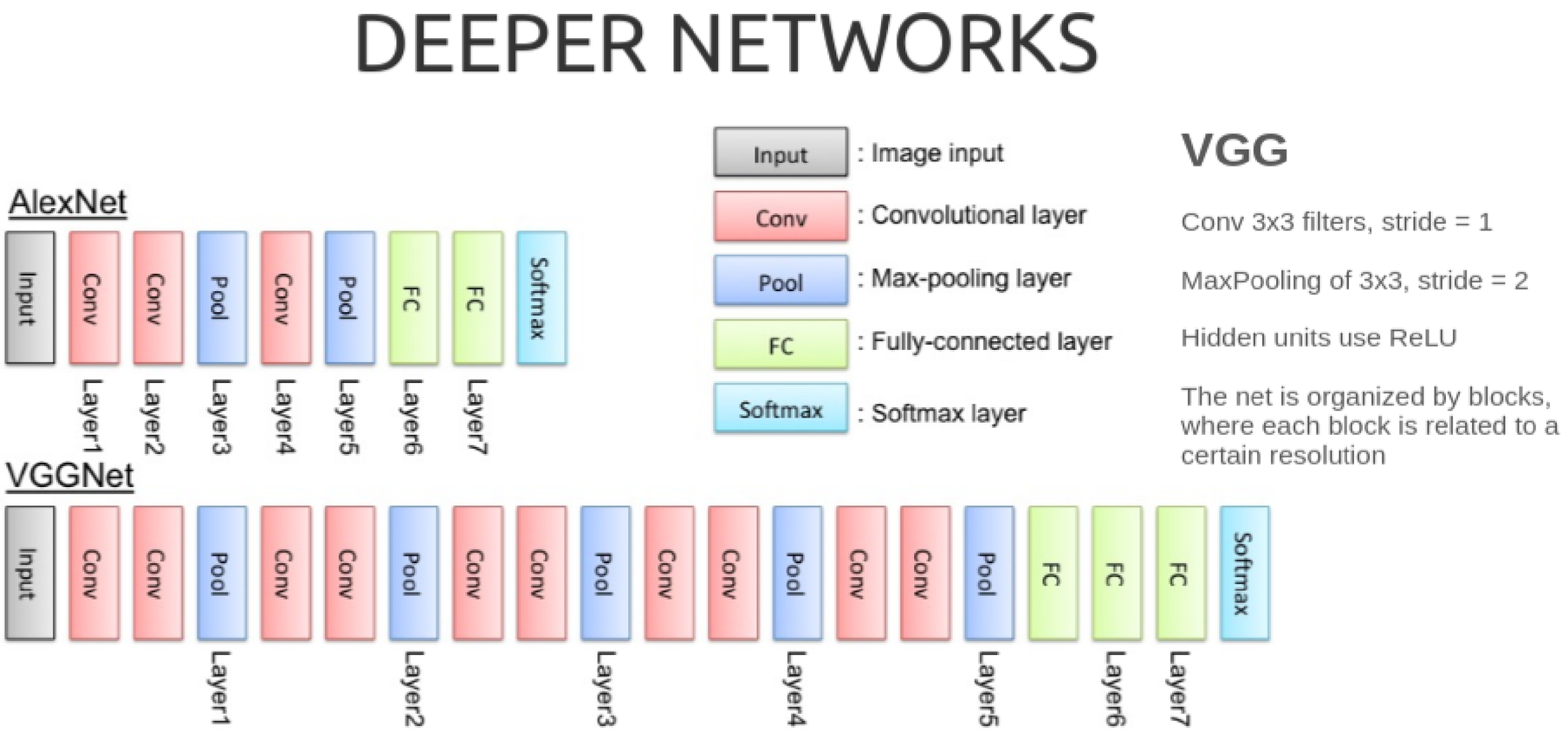
Vanishing Gradient Problem





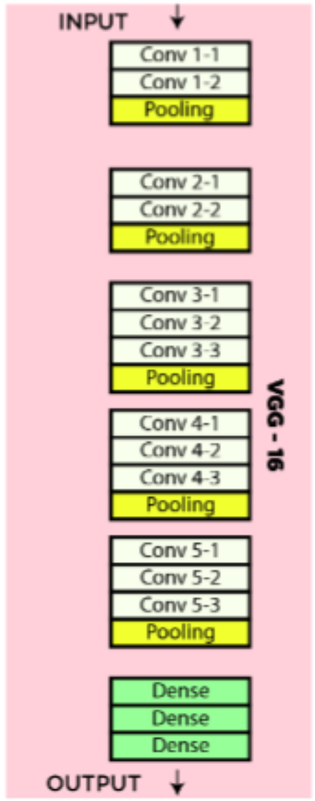
https://www.youtube.com/watch?v=e0dlpYaJ9k0&list=PLUh8MMo7TfsPLI0QtwXMb59lBDsPAYsHk

Redes Convolucionales



Karen Simonyan, Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". ICLR-2015

Redes Convolucionales



VGG

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**
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**
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**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG-16

Redes Convolucionales

VGG

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

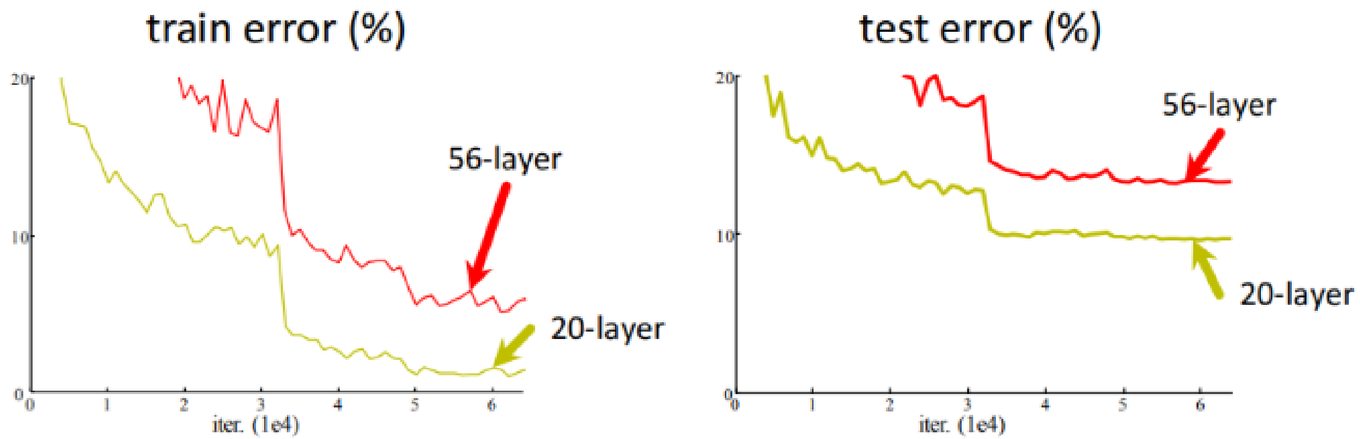
The input is randomly cropped to 224x224

AlexNet (16.4%)

Redes Convolucionales

Simply stacking layers?

CIFAR-10

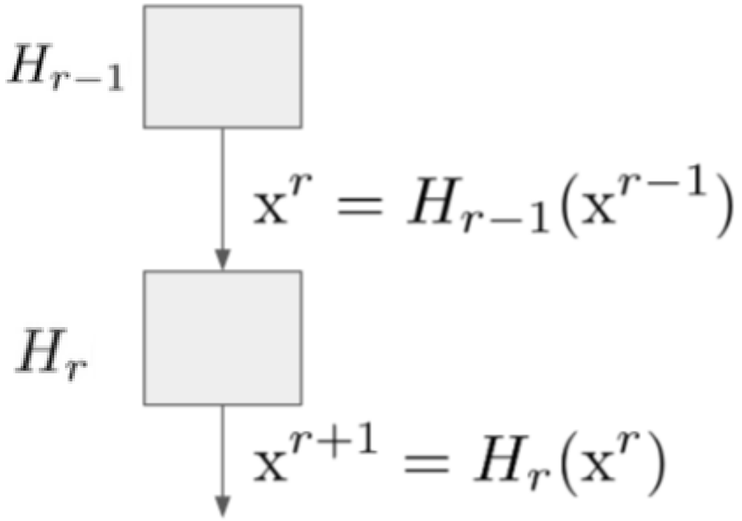


- *Plain* nets: stacking 3x3 conv layers…
- 56-layer net has **higher training error** and test error than 20-layer net

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep Residual Learning for Image Recognition". 2015

Redes Convolucionales

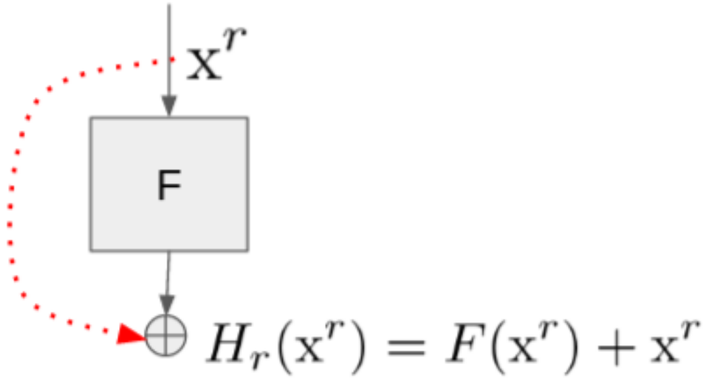
Residual Networks (ResNet)



Machine Learning: A Bayesian and Optimization Perspective
Sergio Theodoridis

If x^r provides enough information, H_r should learn the identity mapping. That is, the extra layer adds no information and simply copy the input to the output. However, once the network starts becoming very deep, the optimizer experiments difficulties to learn such a solution.

ResNet: A shortcut is the solution



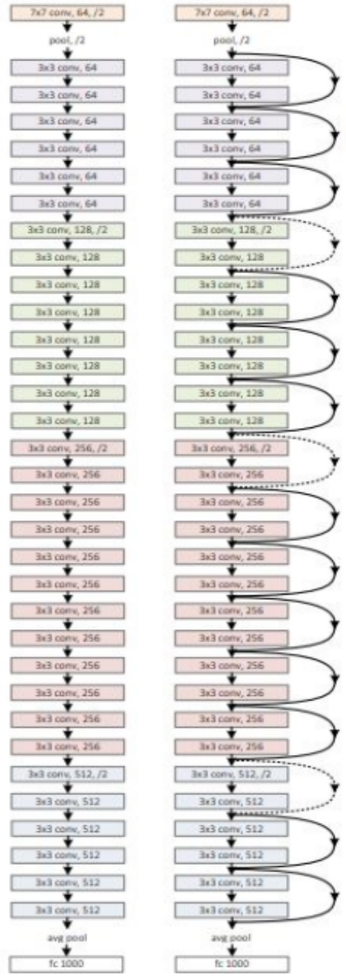
Redes Convolucionales

Residual Networks (ResNet)

Network “Design”

- Keep it simple
- Our basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2 (~same complexity per layer)
  - Simple design; just deep!
- Other remarks:
  - no hidden fc
  - no dropout

plain net



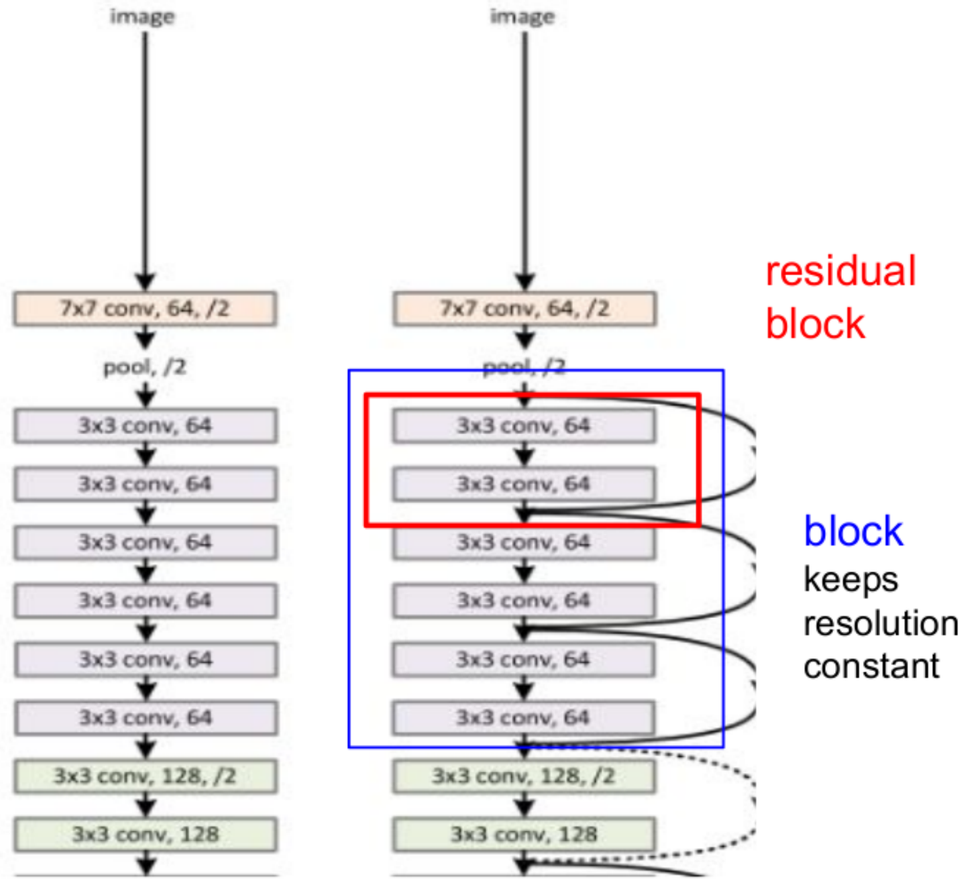
ResNet

Redes Convolucionales

Residual Networks (ResNet)

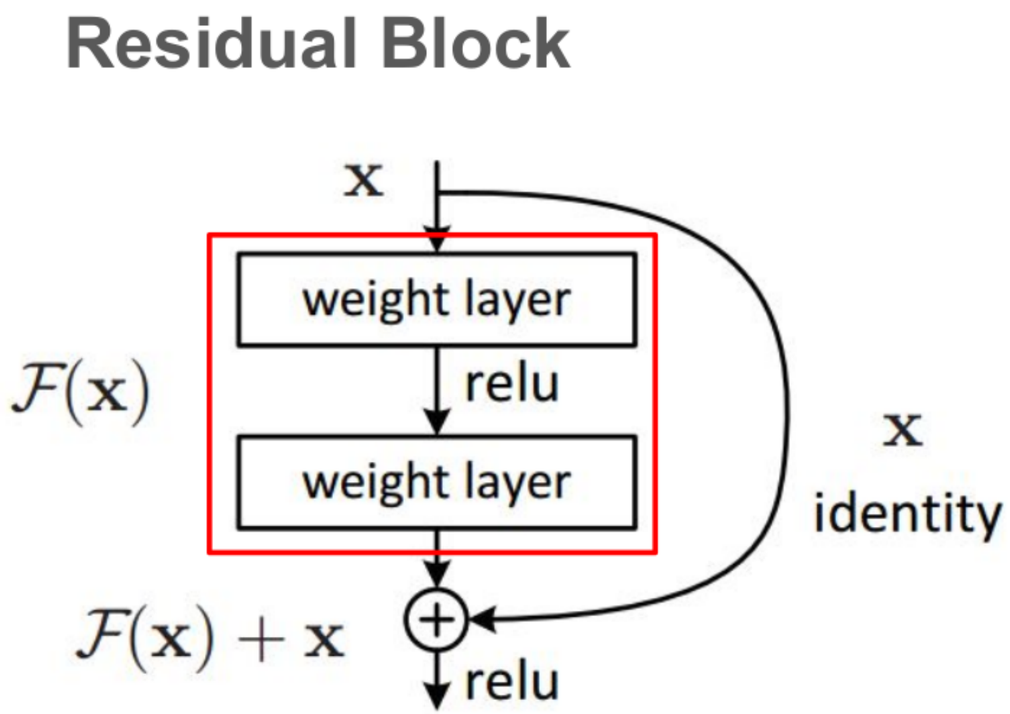
Network "Design"

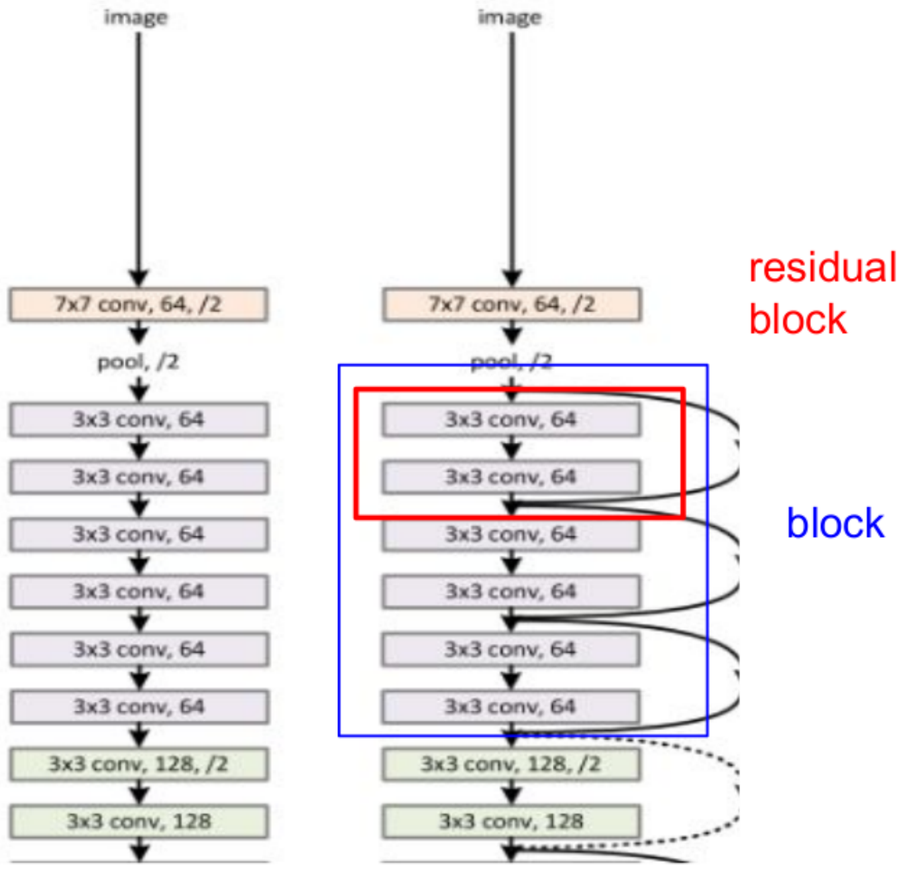
- Keep it simple
- Our basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2 (~same complexity per layer
  - Simple design; just deep!
- Other remarks:
  - no hidden fc
  - no dropout



Redes Convolucionales

Residual Networks (ResNet)

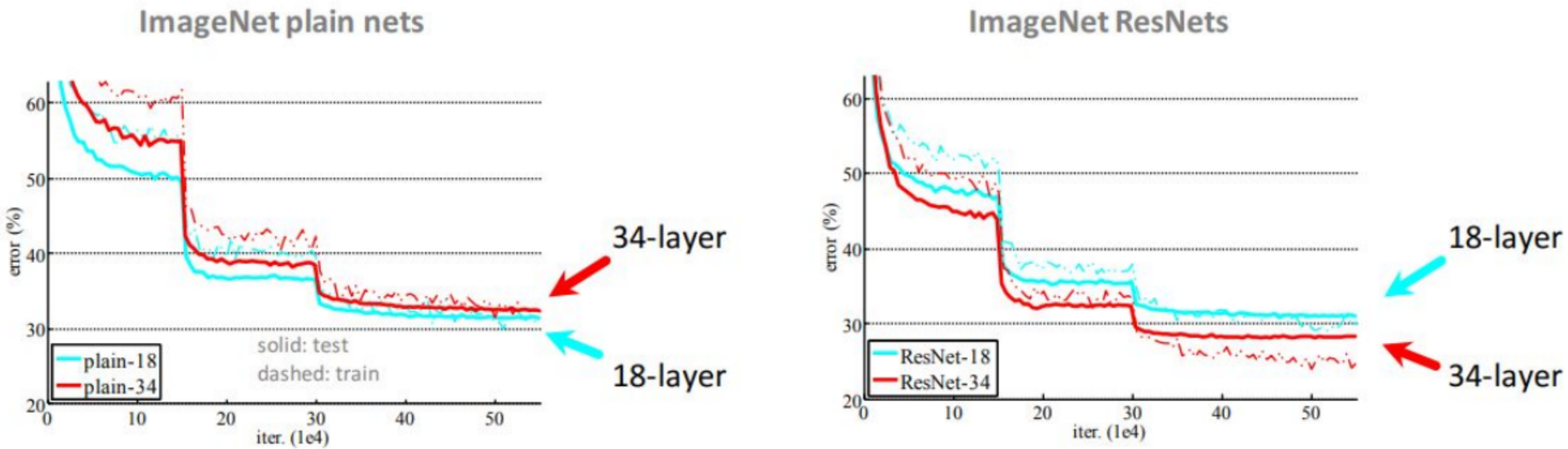




Redes Convolucionales

Residual Networks (ResNet)

ImageNet experiments



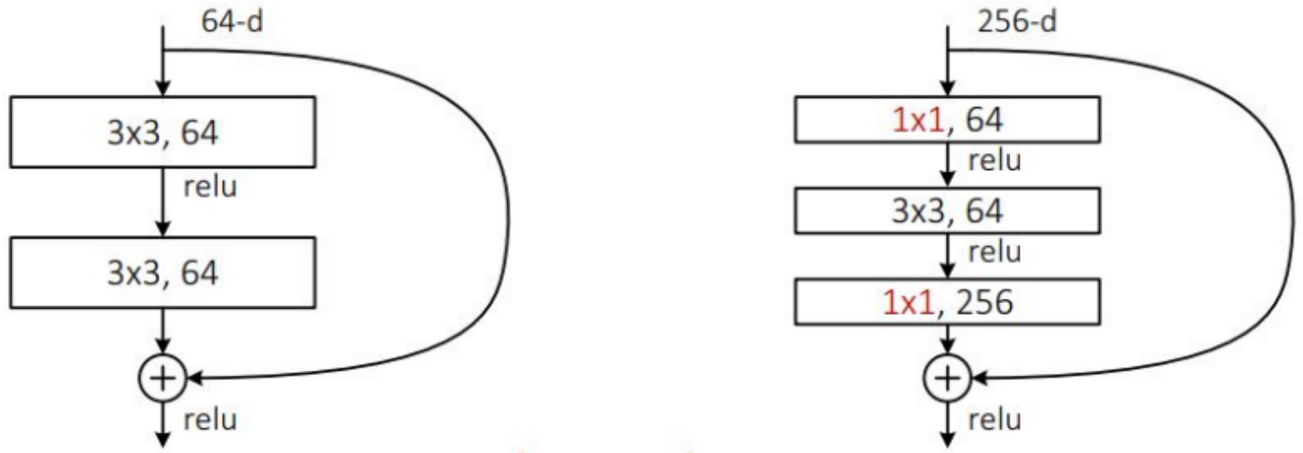
- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Redes Convolucionales

Residual Networks (ResNet)

ImageNet experiments

- A practical design of going deeper



Redes Convolucionales

Residual Networks (ResNet)

A implementation of a ResNet block

https://github.com/jmsaavedrar/convnet2/blob/master/models/resnet.py

```
class ResidualBlock(tf.keras.layers.Layer):
    """
    residual block implementated in a full preactivation mode
    input bn-relu-conv1-bn-relu-conv2->y------------------
      |                                                  |+
      ------------------(projection if necessary)-->shortcut--> y + shortcut

    """
    def __init__(self, filters, stride, use_projection = False, se_factor = 0,  **kwargs):
        super(ResidualBlock, self).__init__(**kwargs)
        self.bn_0 = tf.keras.layers.BatchNormalization(name = 'bn_0')
        self.conv_1 = conv3x3(filters, stride, name = 'conv_1', use_bias = False)
        self.bn_1 = tf.keras.layers.BatchNormalization(name = 'bn_1', )
        self.conv_2 = conv3x3(filters, 1, name = 'conv_2', use_bias = False)
        self.use_projection = use_projection;
        self.projection = 0
        if self.use_projection :
            self.projection = conv1x1(filters, stride, name = 'projection', use_bias = False)

        self.se = 0
        self.use_se_block = False
        if se_factor > 0 :
            self.se = SEBlock(filters, filters / se_factor)
            self.use_se_block = True

    #using full pre-activation mode
    def call(self, inputs, training = True):
        y = self.bn_0(inputs)
        y = tf.keras.activations.relu(y)
        if self.use_projection :
            shortcut = self.projection(y)
        else :
            shortcut = inputs
        y = self.conv_1(y)
        y = self.bn_1(y, training)
        y = tf.keras.activations.relu(y)
        y = self.conv_2(y)
        if self.use_se_block :
            y = self.se(y)
        y = shortcut + y # residual function
        return y
```

https://github.com/jmsaavedrar/convnet2/blob/master/models/resnet.py

Redes Convolucionales

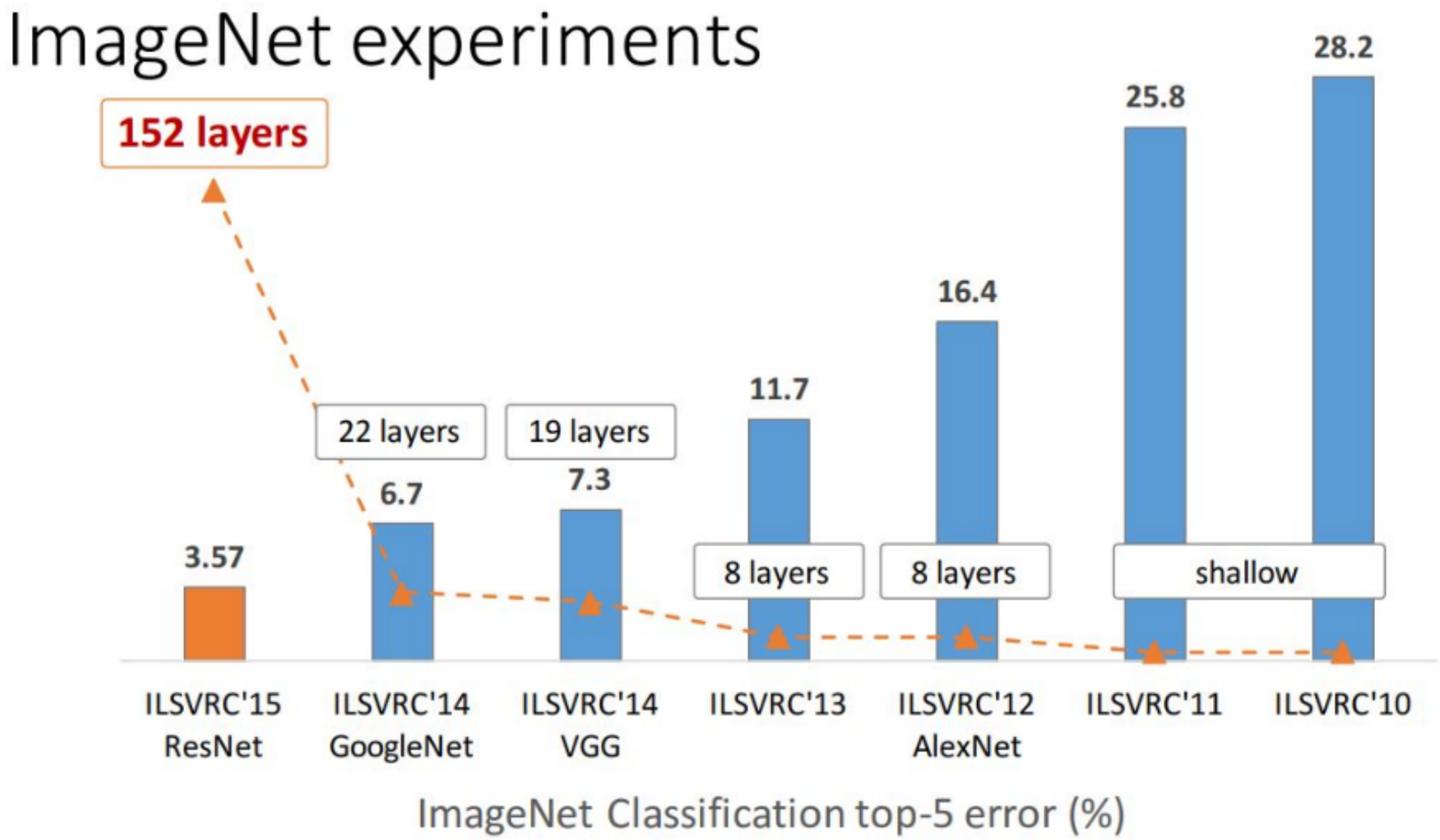
Residual Networks (ResNet)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

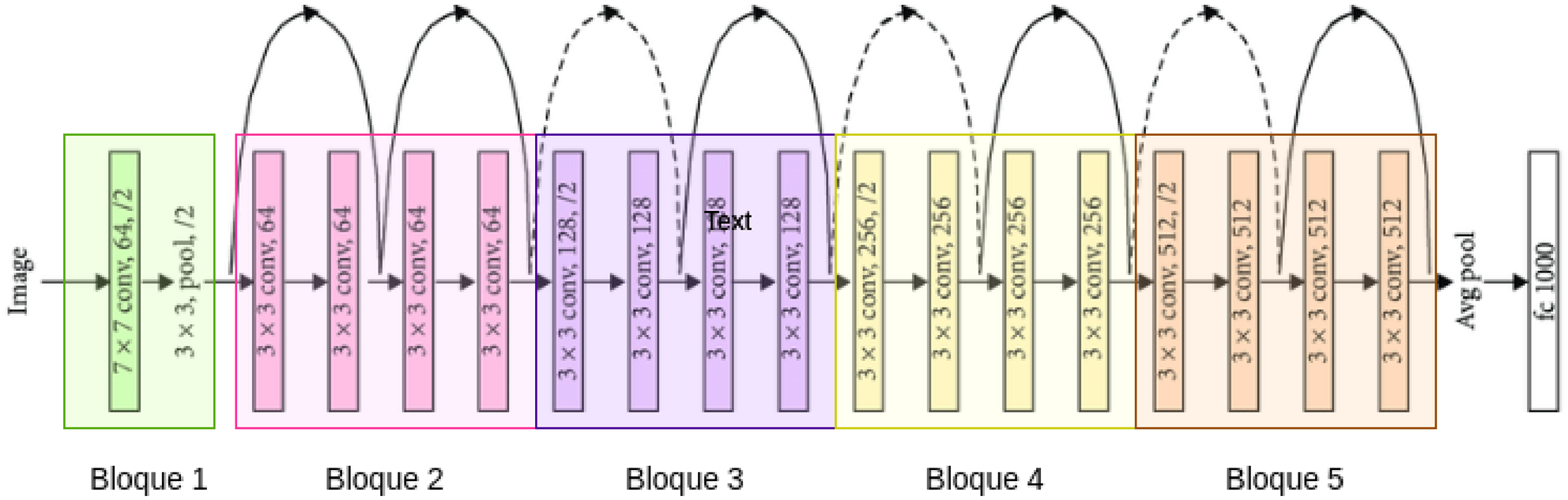
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep Residual Learning for Image Recognition". 2015

Redes Convolucionales

Residual Networks (ResNet)



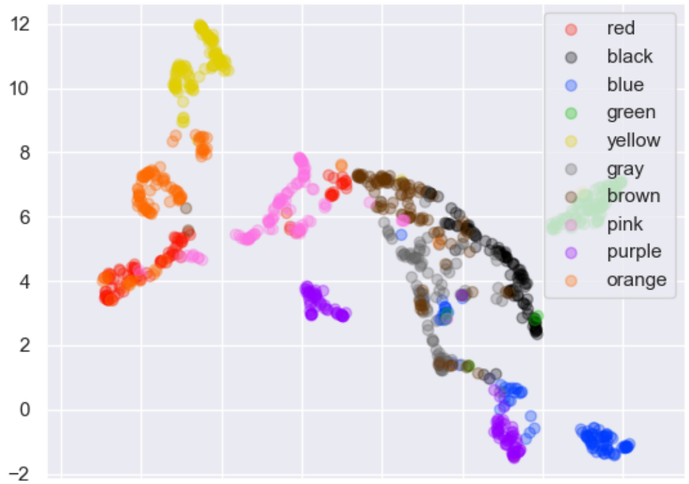
Análisis de un Modelo ResNet



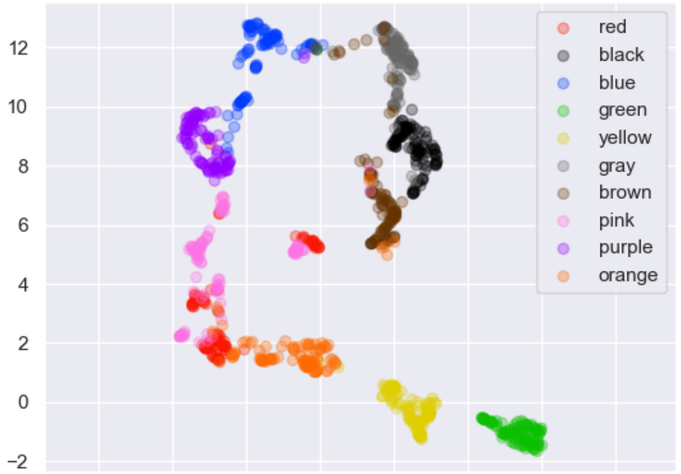
A. Baloian, N. Murrugarra, J. Saavedra. Scalable Visual Attribute Extraction through Hidden Layers of a Residual ConvNet. LatinXinCV research workshop, CVPR 202

Análisis de un Modelo ResNet [atributo = color]

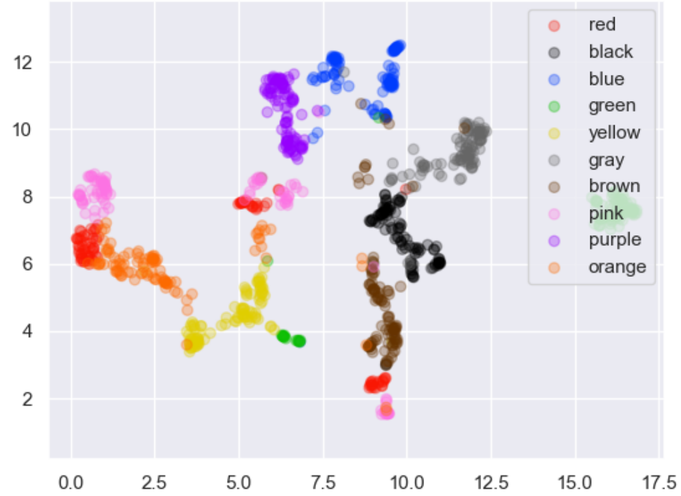
Bloque 1



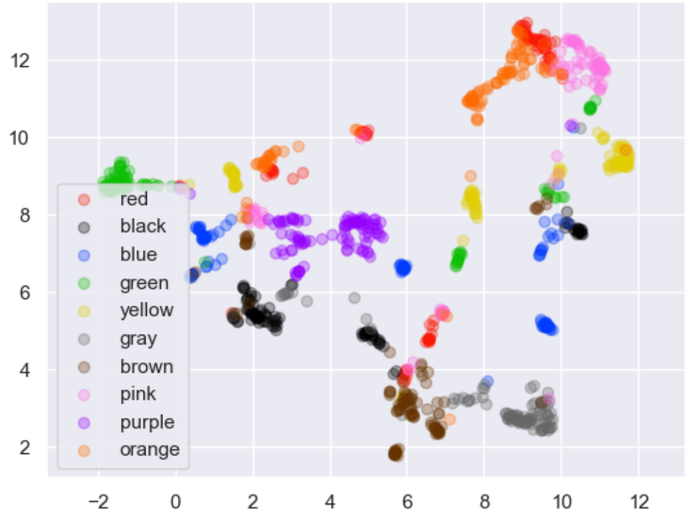
Bloque 2



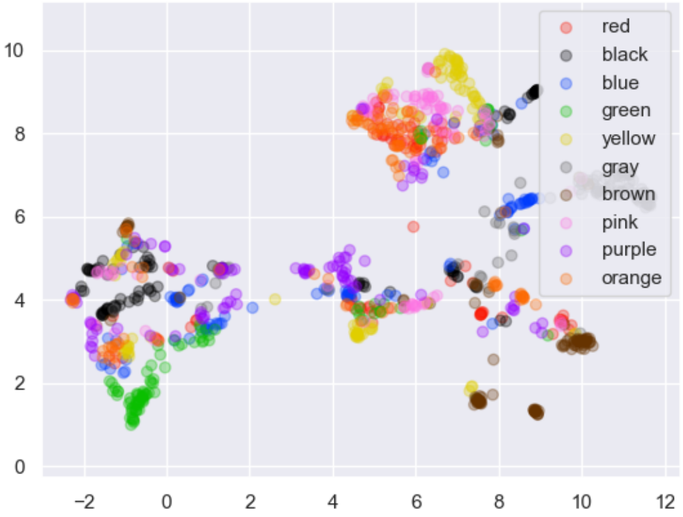
Bloque 3

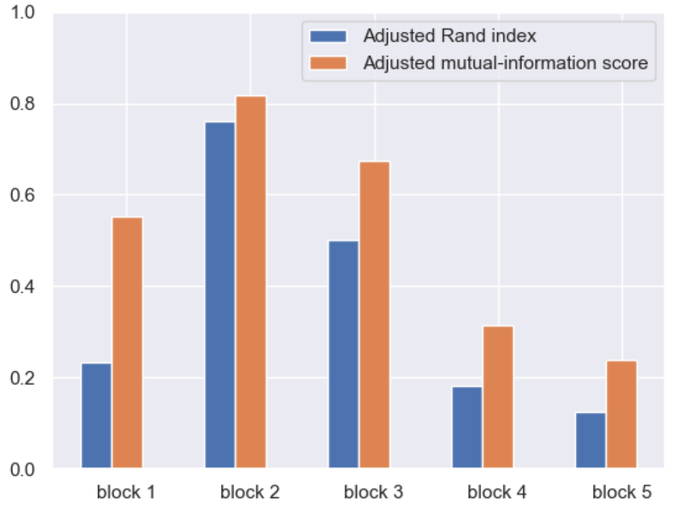


Bloque 4

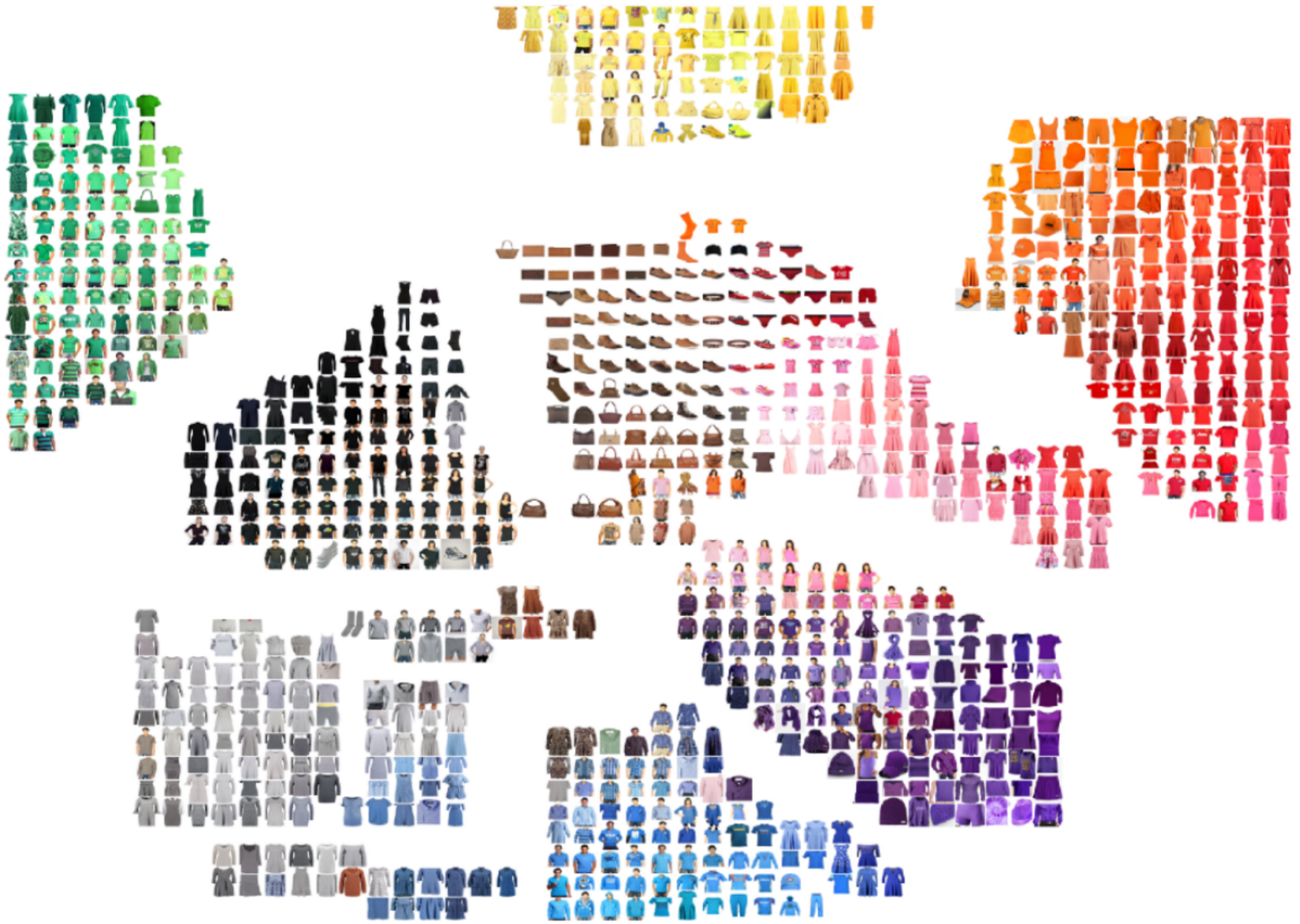


Bloque 5





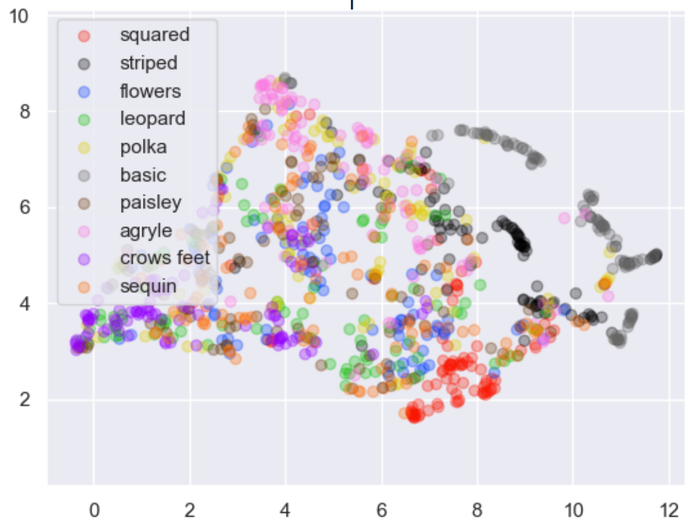
Análisis de un Modelo ResNet [atributo = color]



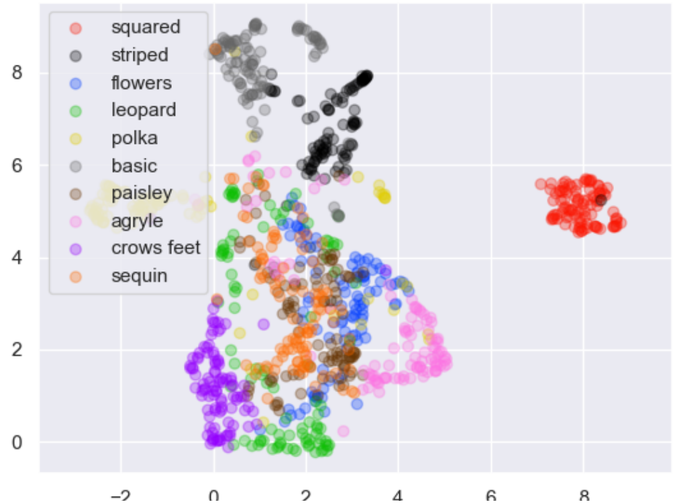
Jose M. Saavedra

Análisis de un Modelo ResNet [atributo = textura]

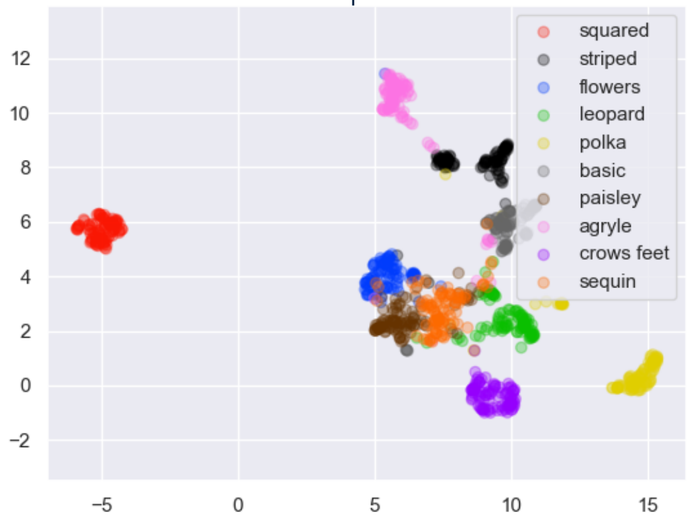
Bloque 1



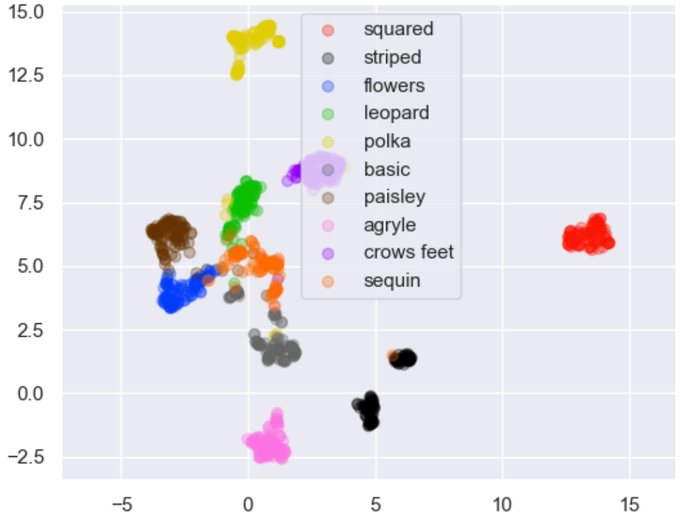
Bloque 2



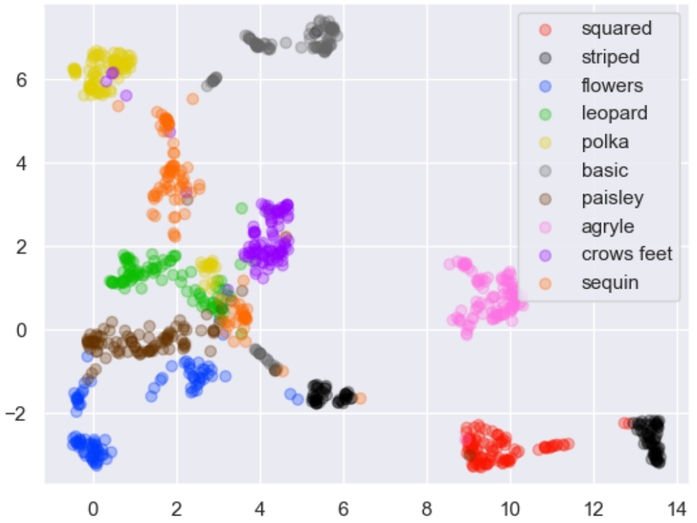
Bloque 3

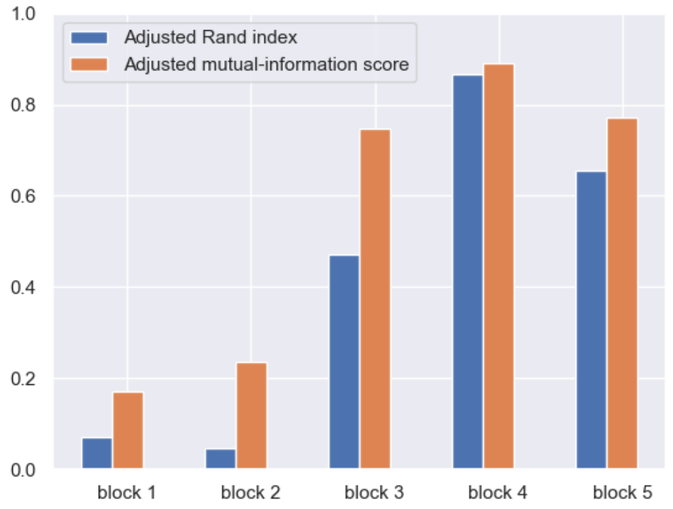


Bloque 4

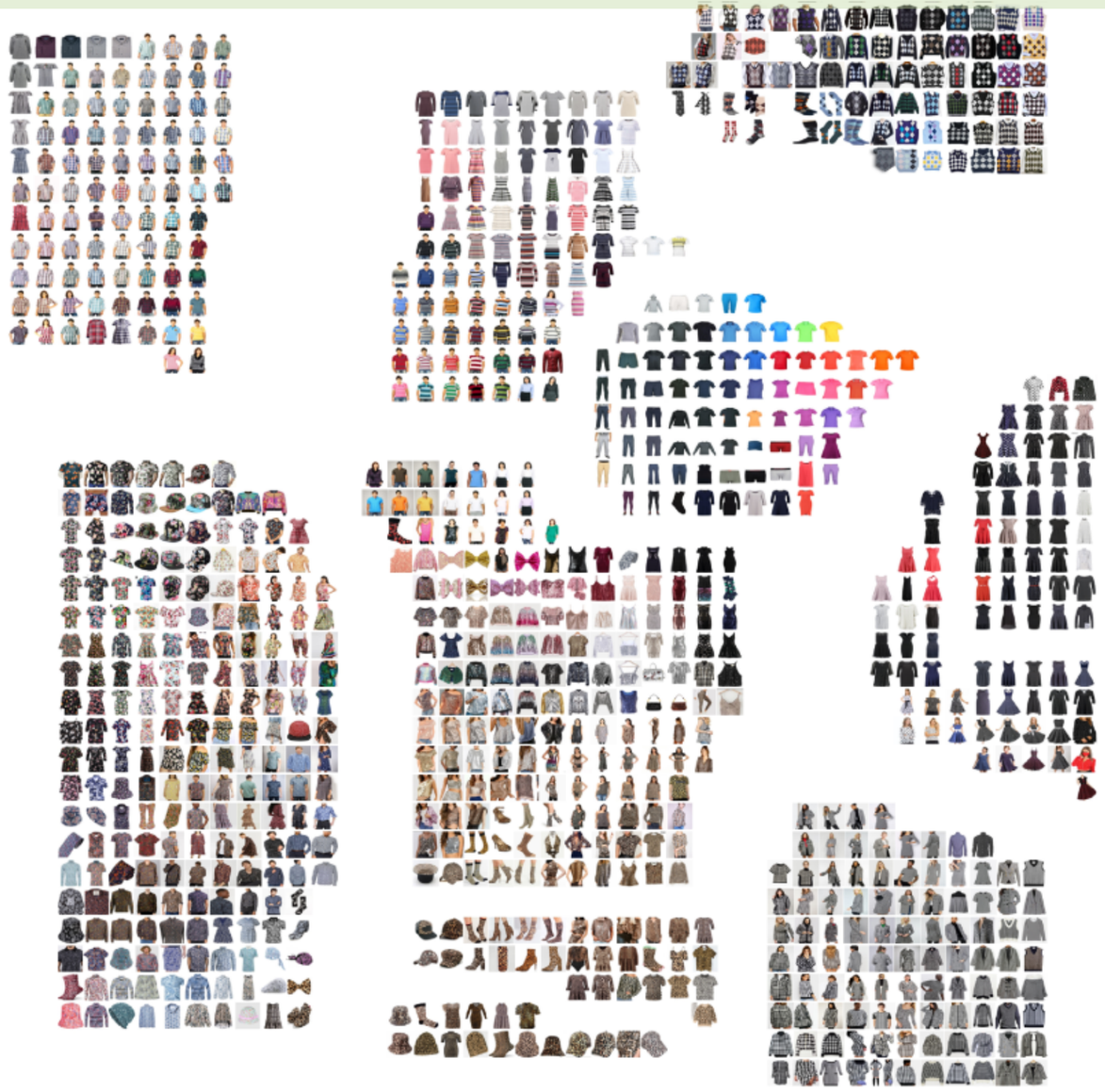


Bloque 5





Análisis de un Modelo ResNet [atributo = textura]



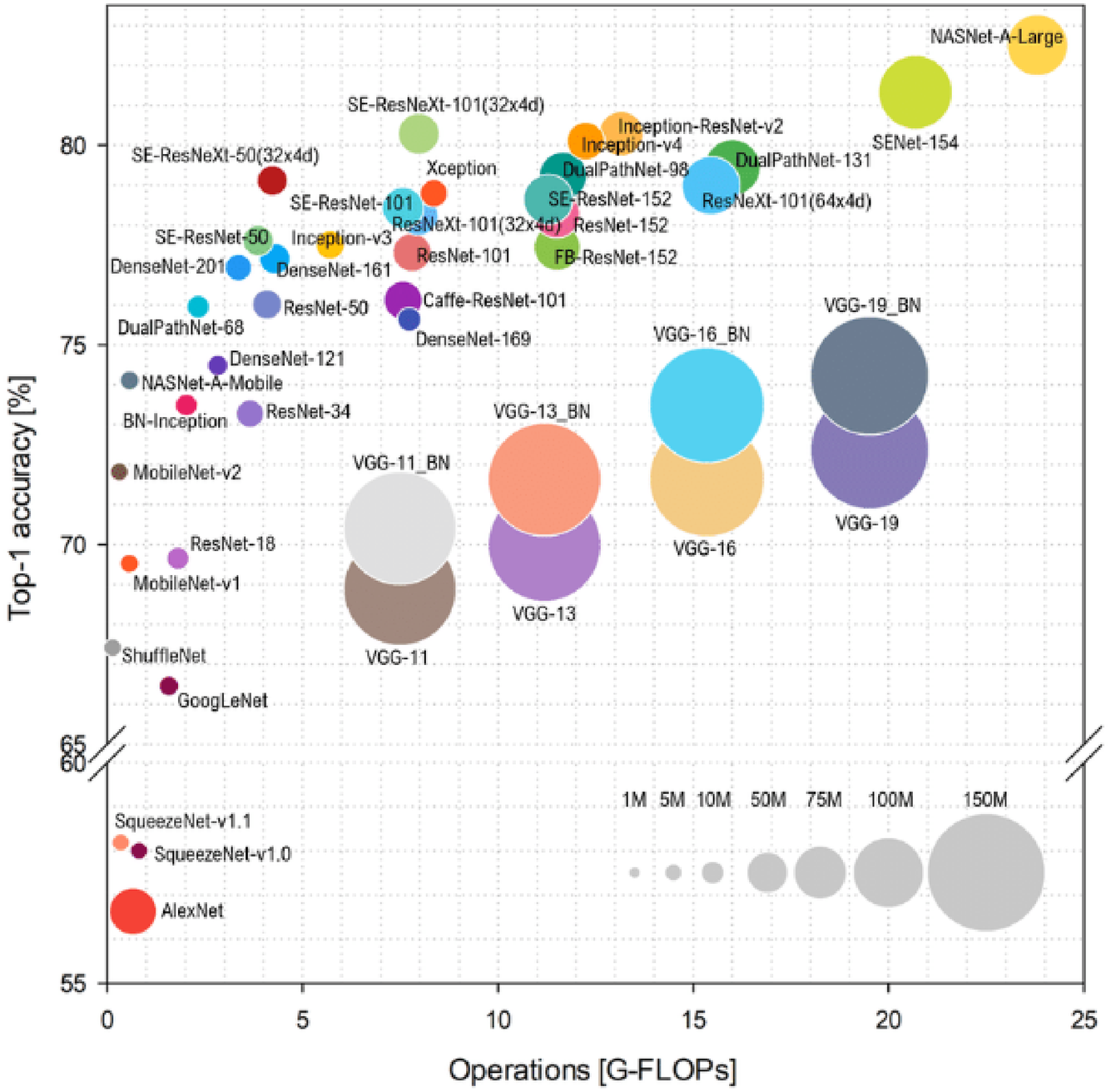
Jose M. Saavedra

Análisis de un Modelo ResNet [atributo = textura]

Práctica

- Evaluar el comportamiento del bloque-2 de un modelo ResNet-50 en el contexto de recuperación de imágenes basada en color.
- Evaluar el comportamiento del bloque-4 de un modelo ResNet-50 en el contexto de recuperación de imágenes basada en textura.

https://github.com/jmsaavedrar/visual_attributes


NASNet-A-Large
SE-ResNeXt-101(32x4d)
Inception-ResNet-v2
Inception-v4
SENet-154
SE-ResNeXt-50(32x4d)
Xception
DualPathNet-98
DualPathNet-131
SE-ResNet-101
SE-ResNet-152
ResNeXt-101(64x4d)
ResNeXt-101(32x4d)
ResNet-152
SE-ResNet-50
Inception-v3
ResNet-101
FB-ResNet-152
DenseNet-201
DenseNet-161
Caffe-ResNet-101
ResNet-50
VGG-19_BN
DualPathNet-68
DenseNet-169
VGG-16_BN
DenseNet-121
NASNet-A-Mobile
BN-Inception
ResNet-34
VGG-13_BN
VGG-11_BN
MobileNet-v2
VGG-19
VGG-16
ResNet-18
MobileNet-v1
VGG-13
VGG-11
ShuffleNet
GoogLeNet
1M
5M
10M
50M
75M
100M
150M
SqueezeNet-v1.1
SqueezeNet-v1.0
AlexNet
Top-1 accuracy [%]
Operations [G-FLOPs]
80
75
70
65
60
55
0
5
10
15
20
25

Clase 3

- Modelos Atencionales
- Tranformers
- Vision-Transformers