

Una breve introducción a la clasificación de imágenes

Slides: Giovanni Rescia

Motivación: clasificación de imágenes

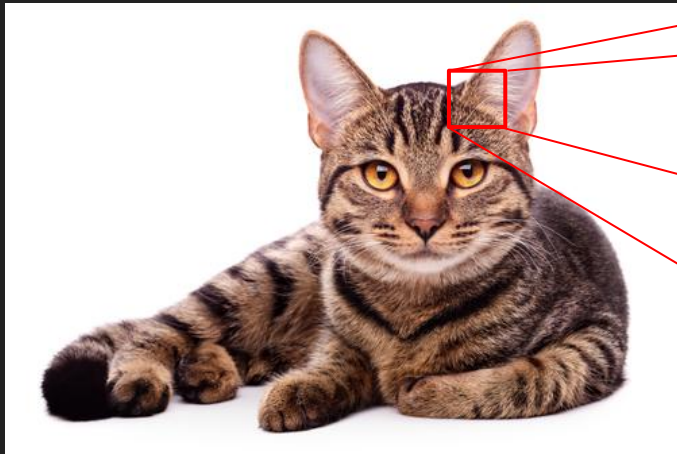
Clases fijas

{gato, avión, auto,...}



→ gato

El problema



54	58	255	8	0	
45	0	78	51	100	74
85	47	34	185	207	21
22	20	148	52	24	147
52	36	250	74	214	278
	158	0	78	51	247
		72	74	136	251

Algunos desafíos

iluminación



deformación



Algunos desafíos

oclusión



Algunos desafíos

background clutter



variación de clases



Clasificador de imágenes

```
def predict(image):  
    # ????  
    return class_label
```

- Construcción robusta
- No hay forma obvia
- Depende del dominio

Data-driven approach:

- Conseguir imágenes con su clase
- Usar machine learning para entrenar un clasificador
- Evaluar el clasificador con imágenes fuera del conjunto de entrenamiento

```
def train(train_images, train_labels):  
    # build a model for images -> labels...  
    return model  
  
def predict(model, test_images):  
    # predict test_labels using the model...  
    return test_labels
```



Primer clasificador: Nearest Neighbors

```
def train(train_images, train_labels):  
    # build a model for images -> labels...  
    return model  
  
def predict(model, test_images):  
    # predict test_labels using the model...  
    return test_labels
```

guardar todas las imágenes de
entrenamiento con su clase

Predecir la clase de la imagen
de entrenamiento más parecida

Cómo se comparan las imágenes?

L1 distance:

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

add
→ 456

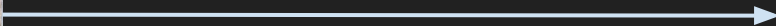
Parametric approach: clasificador lineal



[32, 32, 3]
(imagen=32x32 pixeles x 3 canales)

imagen parámetros

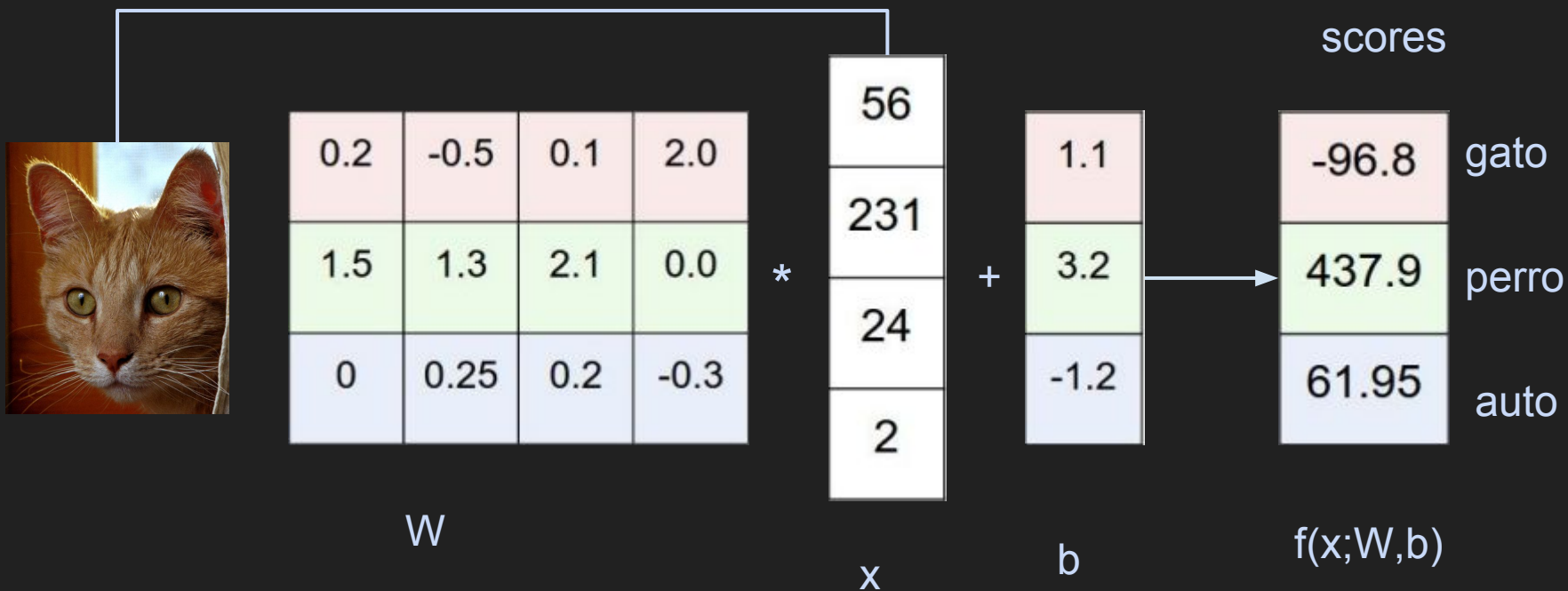
$$f(\mathbf{x}, \mathbf{W})$$



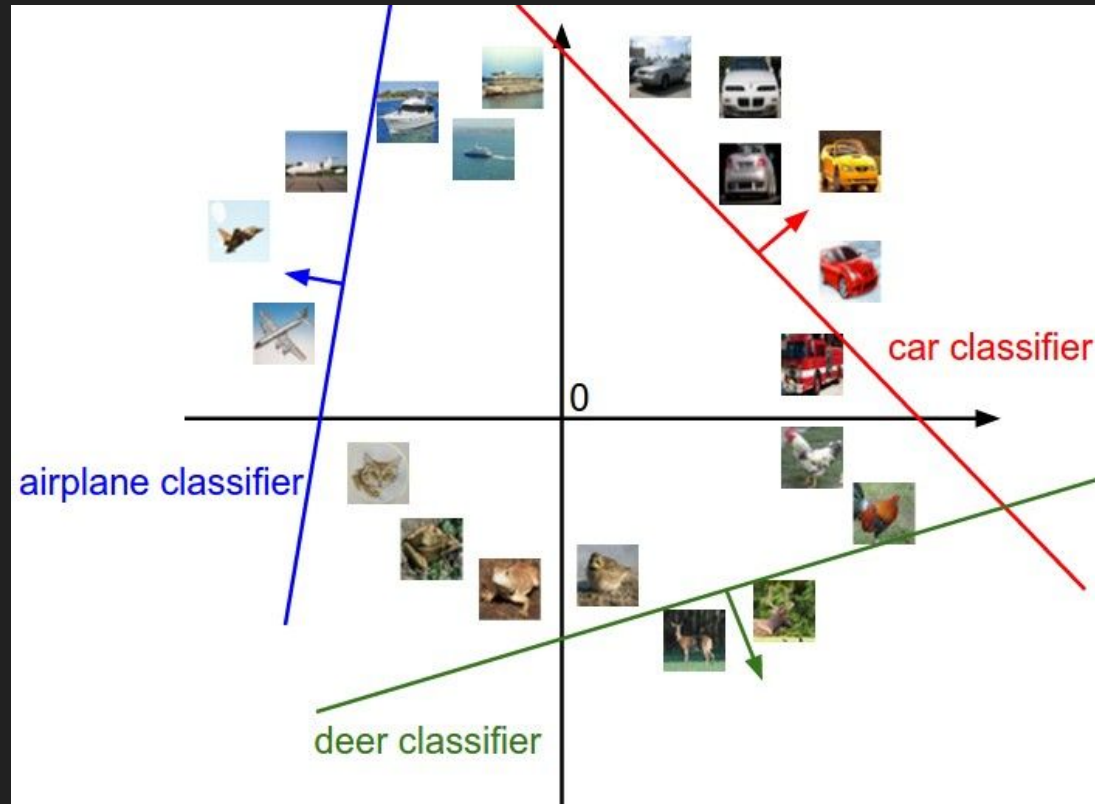
Asumimos 3
clases (eg, gato,
perro, auto)

3 números,
indicando el
score de cada
clase

Ejemplo

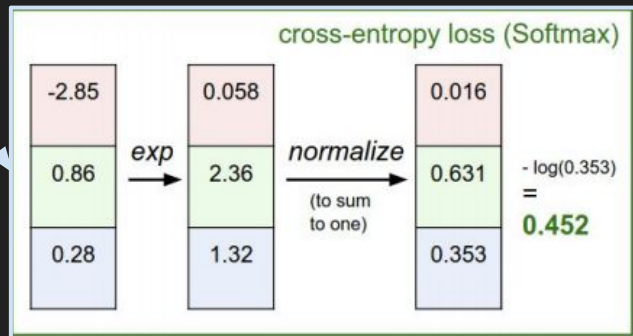
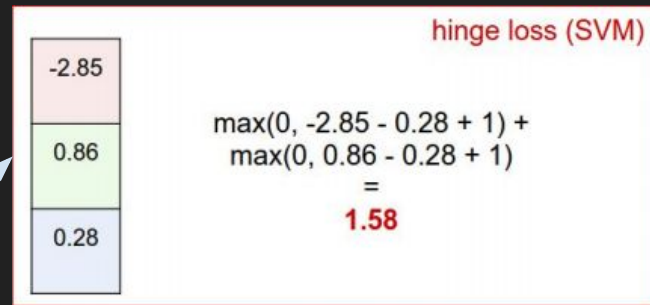
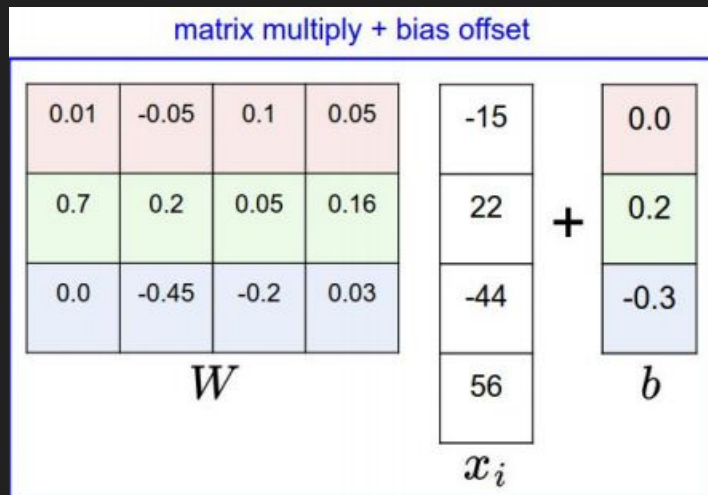


Ejemplo



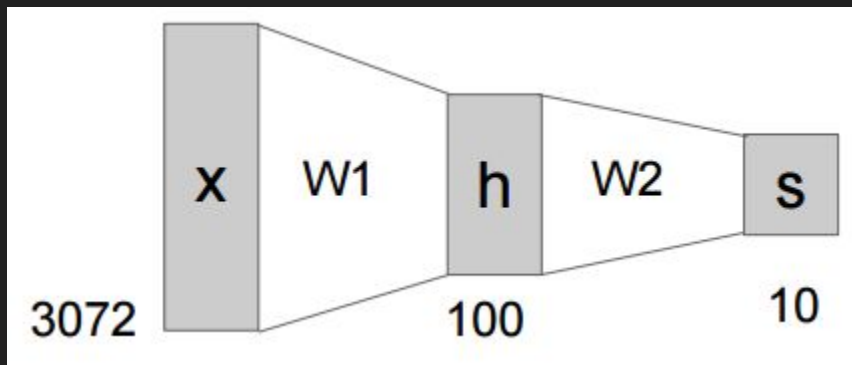
Función de pérdida / Optimización

- Definimos una función de loss para cuantificar el error
- Ajustamos nuestros pesos W para minimizar la loss



Redes neuronales

- Función lineal: $f = W * x$
- Red neuronal de dos capas: $f = W' * \max(0, W * x)$

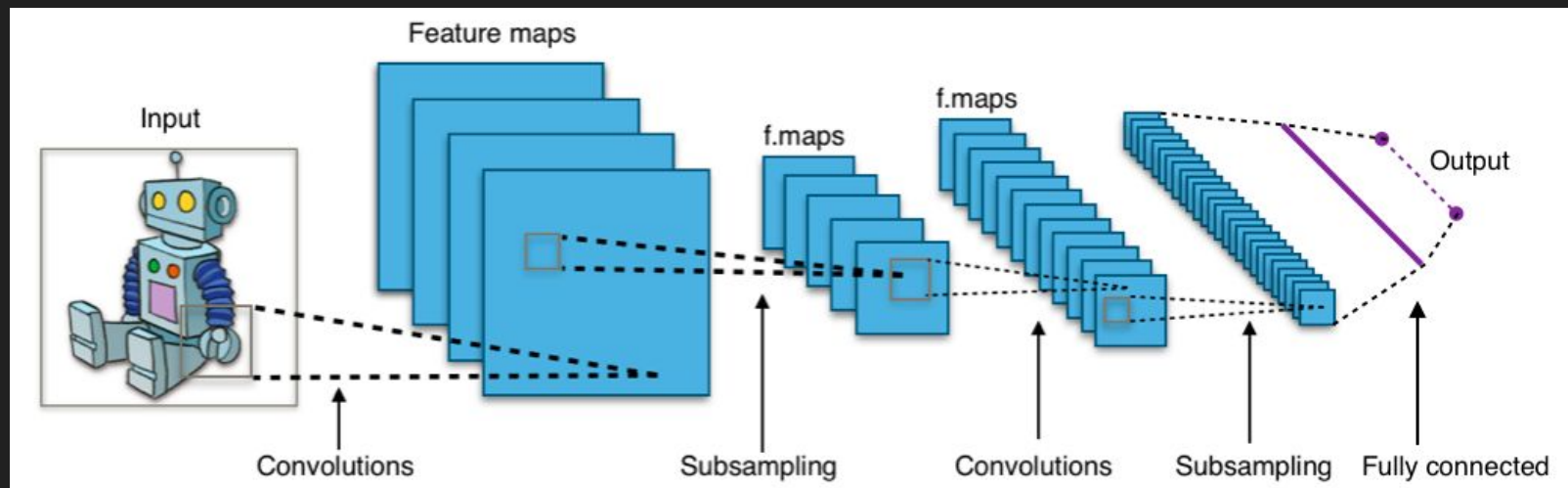


- Red neuronal de tres capas: $f = W'' * \tanh(W' * \max(0, W * x))$

Red neuronal (NN): composición de funciones lineales con funciones no lineales en el medio

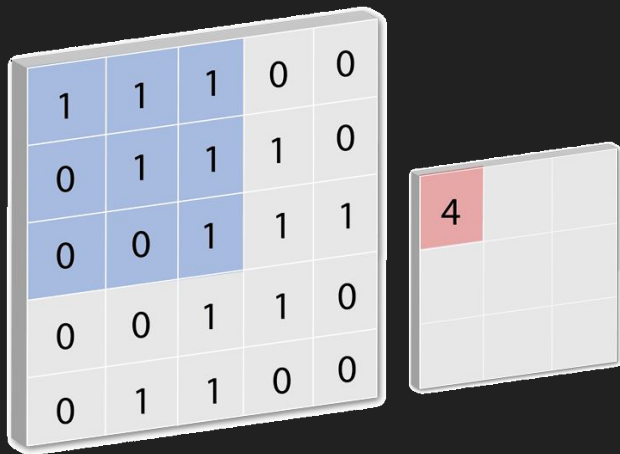
Redes convolucionales (CNN)

- Approach red neuronal: trabajar con toda la imagen a la vez
- Approach convolucional: inspeccionar la imagen de a pequeñas partes (correlación espacial / no independencia de pixels)

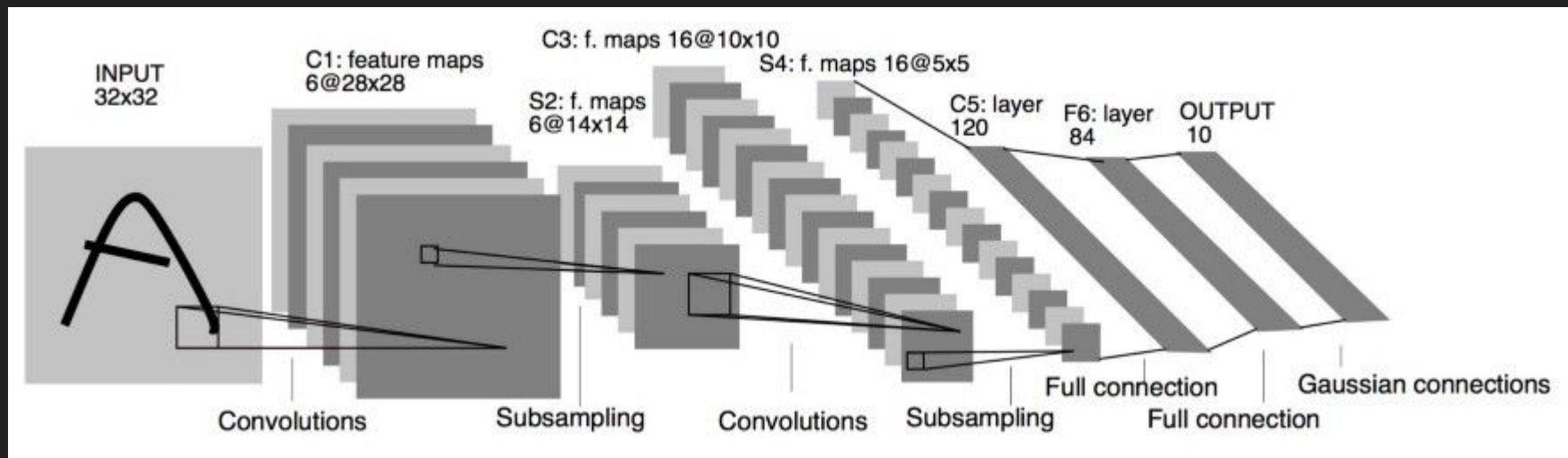


CNN

- Muchas multiplicaciones y sumas (convoluciones), funciones no lineales
- Entrenadas en datasets grandes
- Mismos principios de propagación del error
- GPUs



CNN: LeNet

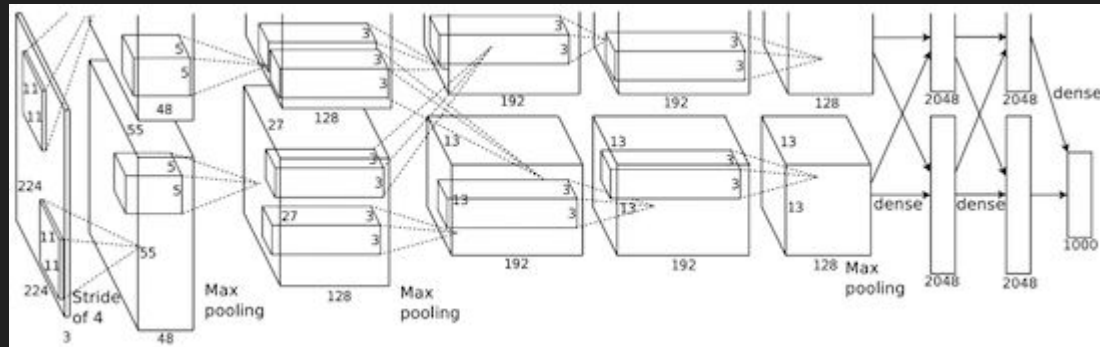




(Error rate top 5)

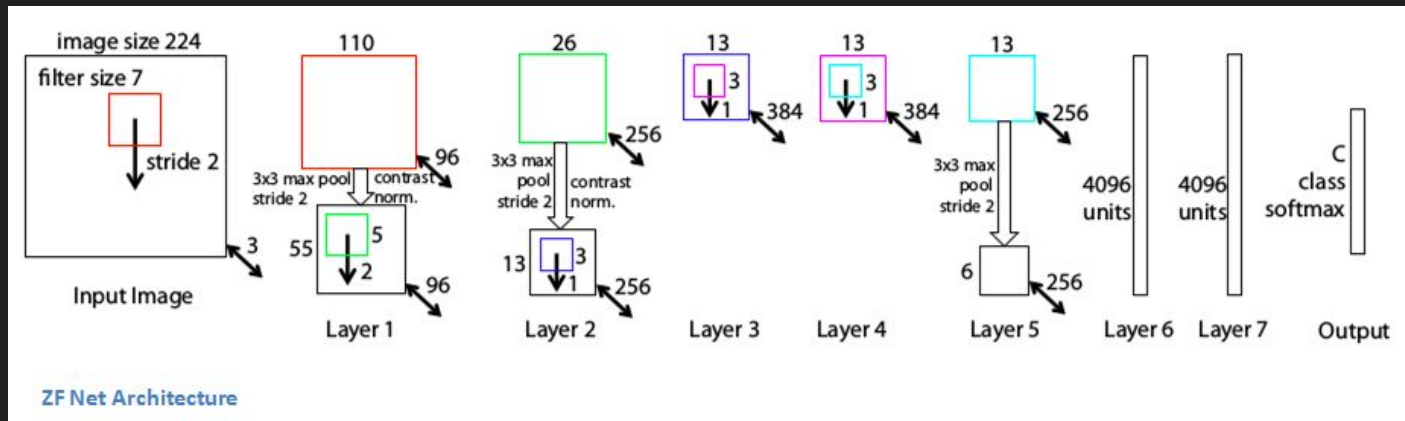
CNN: AlexNet

- 16.4% Error rate
- Entrenada con 15 millones de imágenes
- Entrenada en dos GTX 580 GPUs (5/6 días)



CNN: ZF Net

- 14.8% Error rate
- AlexNet tuneada
- Entrenada con 1.3 millones de imágenes
- GTX 580 GPU (12 días)



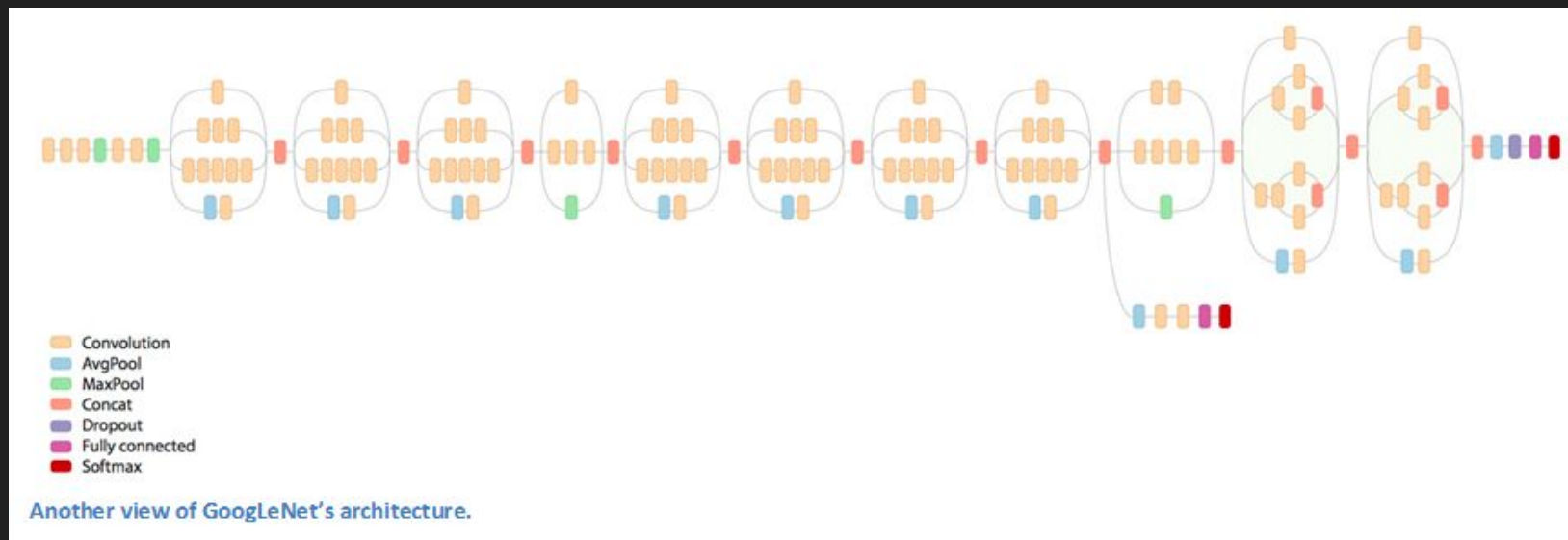
CNN: VGG Net

- 7.3% Error rate
- Punto fuerte: profundidad
- Convoluciones más chicas
- 4 Nvidia Titan Black GPUs (2->3 semanas)

16 weight layers	
)	
conv3-64 conv3-64	
conv3-128 conv3-128	
conv3-256 conv3-256 conv3-256	
conv3-512 conv3-512 conv3-512	
conv3-512 conv3-512 conv3-512	
	maxpool
	FC-4096
	FC-4096
	FC-1000
	soft-max

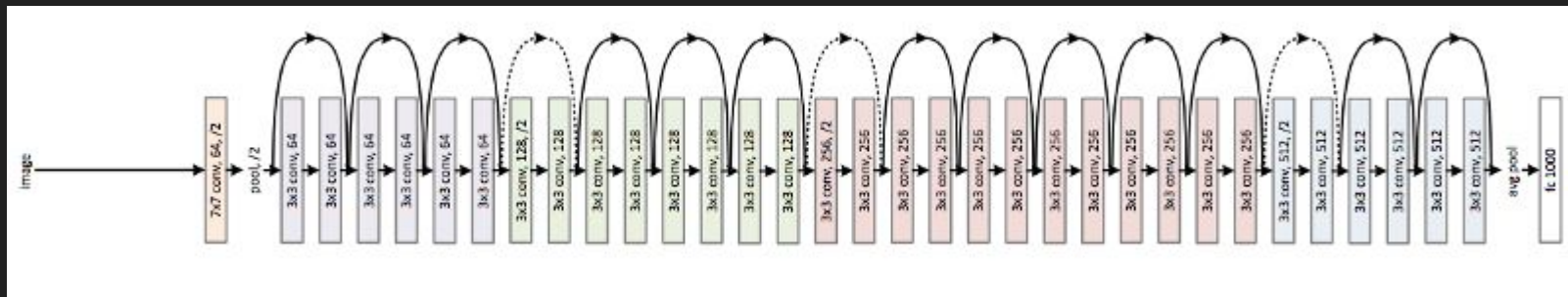
CNN: Google Net

- 6.67% Error Rate
- ~3x más capas que AlexNet (pero 12x menos parámetros)
- Paralelización
- Entrenada en “algunas pocas GPUs” en una semana



CNN: ResNet

- 3.57% Error rate (humano: 5-10%)
- 152 capas ("ultra-deep")
- "mantener" el input original
- 8 GPU (2->3 semanas)



CNN: ResNet

- 3.57% Error rate (humano: 5-10%)
- 152 capas (“ultra-deep”)
- “mantener” el input original
- 8 GPU (2->3 semanas)

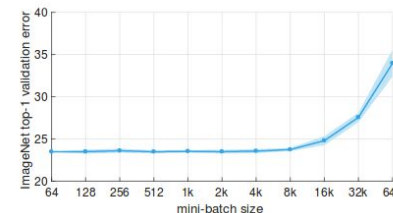
Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis
Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

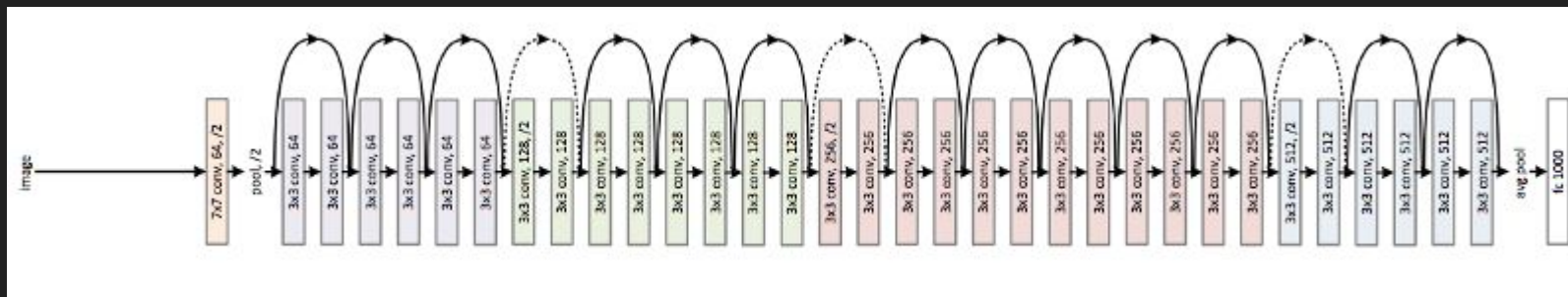
Facebook

Abstract

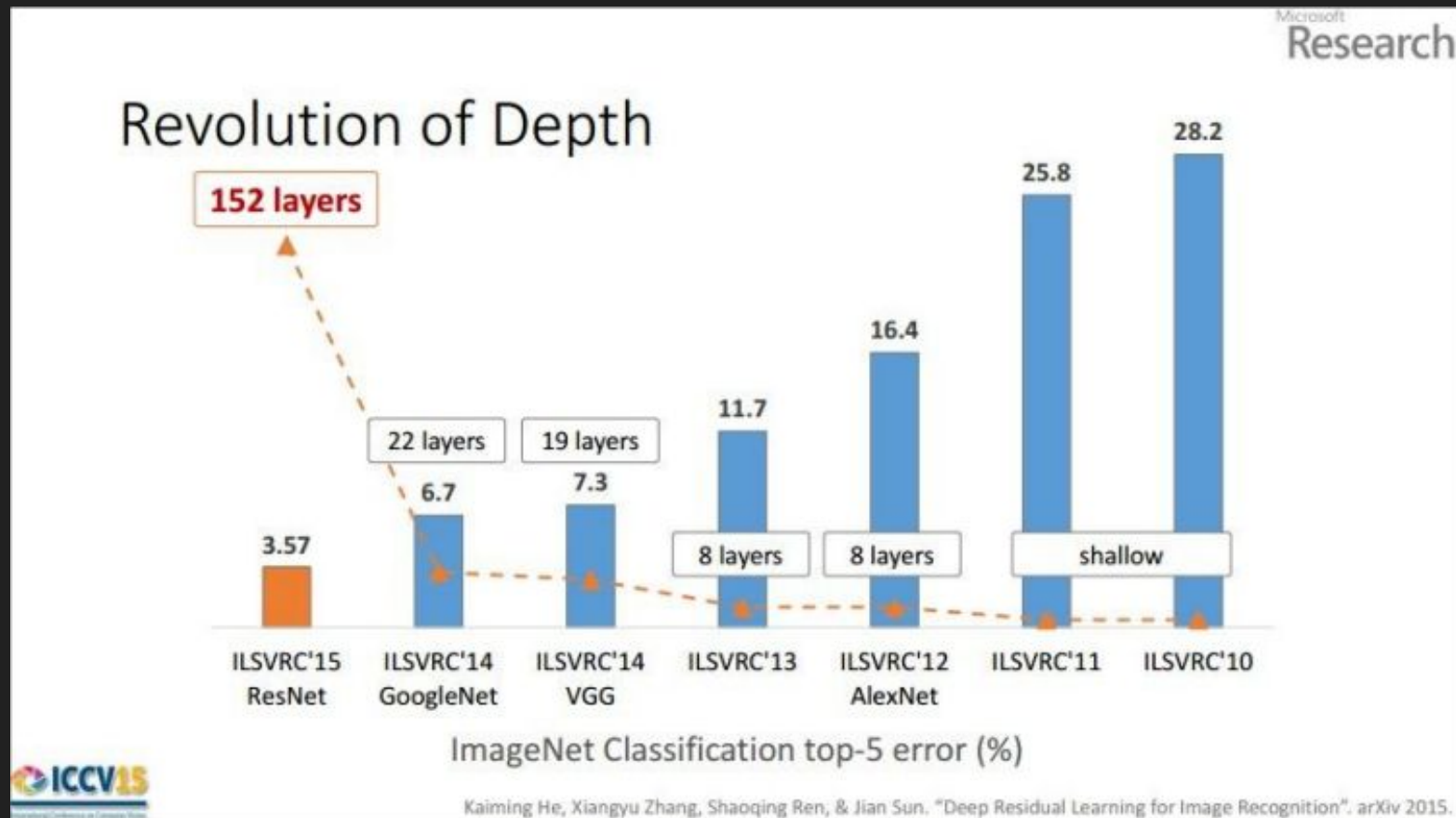
Deep learning thrives with large neural networks and large datasets. However, larger networks and larger datasets result in longer training times that impede research and development progress. Distributed synchronous SGD offers a potential solution to this problem by dividing SGD minibatches over a pool of parallel workers. Yet to make this scheme efficient, the per-worker workload must be large, which implies nontrivial growth in the SGD minibatch size. In this paper, we empirically show that on the



[paper](#)



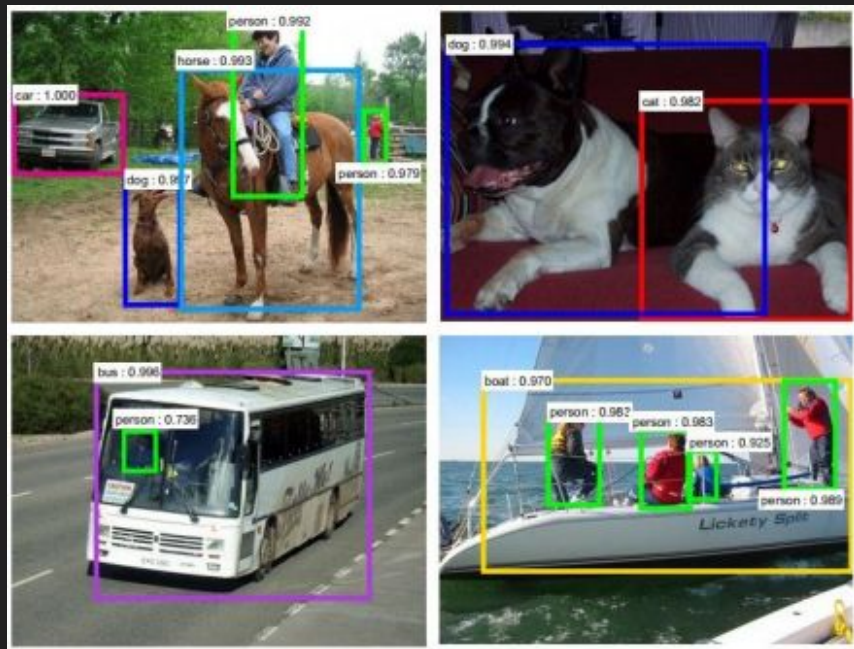
CNN: Resumen



ConvNets, ConvNets everywhere

Retrieval

Detección



ConvNets, ConvNets everywhere

Segmentación



Self-driving cars



ConvNets, ConvNets everywhere

Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
 <p>A person riding a motorcycle on a dirt road.</p>	 <p>Two dogs play in the grass.</p>	 <p>A skateboarder does a trick on a ramp.</p>	 <p>A dog is jumping to catch a frisbee.</p>
 <p>A group of young people playing a game of frisbee.</p>	 <p>Two hockey players are fighting over the puck.</p>	 <p>A little girl in a pink hat is blowing bubbles.</p>	 <p>A refrigerator filled with lots of food and drinks.</p>
 <p>A herd of elephants walking across a dry grass field.</p>	 <p>A close up of a cat laying on a couch.</p>	 <p>A red motorcycle parked on the side of the road.</p>	 <p>A yellow school bus parked in a parking lot.</p>

ConvNets, ConvNets everywhere



(<https://github.com/junyanz/CycleGAN>)

Info recomendada

- Intro a CNNs
(<http://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>)
- Andrej Karpathy's Blog (<http://karpathy.github.io/>)
- CS231n (<http://cs231n.stanford.edu/>)
 - https://www.youtube.com/playlist?list=PL16j5WbGpaM0_Tj8CRmurZ8Kk1qEBc7fq

Muchas gracias!

Preguntas?