Una breve introducción a la clasificación de imágenes

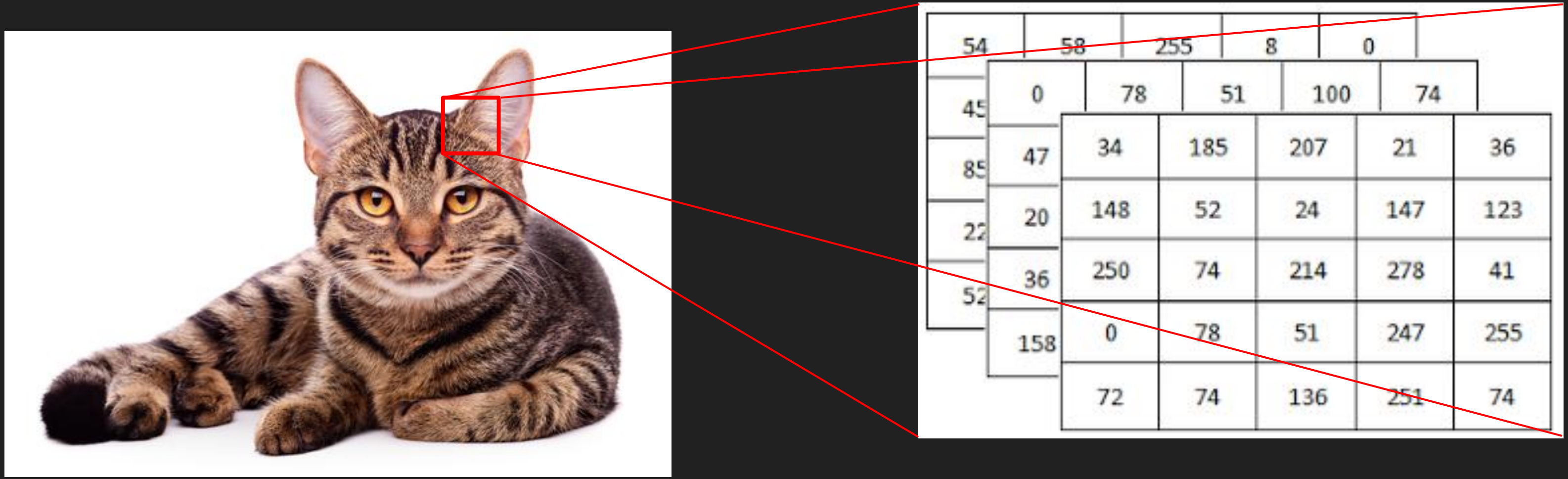
Slides: Giovanni Rescia

Motivación: clasificación de imágenes

Clases fijas
{gato, avión, auto,...}

gato

El problema


54 58 255 8 0
0 78 51 100 74
47 20 36 158
34 185 207 21 36
148 52 24 147 123
250 74 214 278 41
0 78 51 247 255
72 74 136 251 74

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

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

$$\left| \underset{\text{test image}}{\begin{array}{|c|c|c|c|}\hline 56 & 32 & 10 & 18 \\ \hline 90 & 23 & 128 & 133 \\ \hline 24 & 26 & 178 & 200 \\ \hline 2 & 0 & 255 & 220 \\ \hline\end{array}} - \underset{\text{training image}}{\begin{array}{|c|c|c|c|}\hline 10 & 20 & 24 & 17 \\ \hline 8 & 10 & 89 & 100 \\ \hline 12 & 16 & 178 & 170 \\ \hline 4 & 32 & 233 & 112 \\ \hline\end{array}} \right| = \underset{\text{pixel-wise absolute value differences}}{\begin{array}{|c|c|c|c|}\hline 46 & 12 & 14 & 1 \\ \hline 82 & 13 & 39 & 33 \\ \hline 12 & 10 & 0 & 30 \\ \hline 2 & 32 & 22 & 108 \\ \hline\end{array}} \xrightarrow{\text{add}} 456$$

Parametric approach: clasificador lineal

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

imagen parámetros

$f(x, W)$

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

3 números, indicando el score de cada clase

Parametric approach: clasificador lineal

[32, 32, 3]

$$\boxed{f(x,W)} = \boxed{W} * \boxed{x} \; (+b)$$

3 números, indicando el score de cada clase

3x1

3x3072

parámetros o pesos

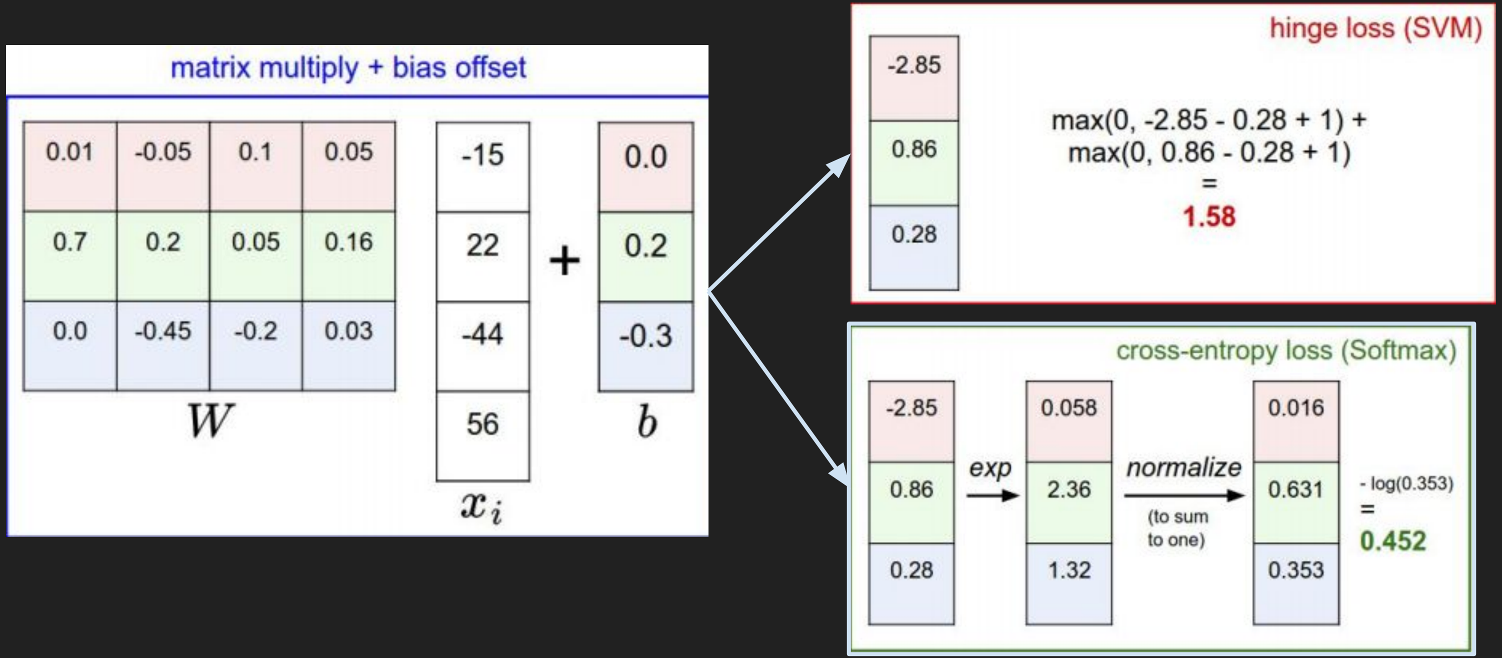
3072x1

Ejemplo

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} * \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} \longrightarrow \overset{\text{scores}}{\underbrace{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}}_{f(x;W,b)}} \begin{matrix} \text{gato} \\ \text{perro} \\ \text{auto} \end{matrix}$$

Función de pérdida / Optimización

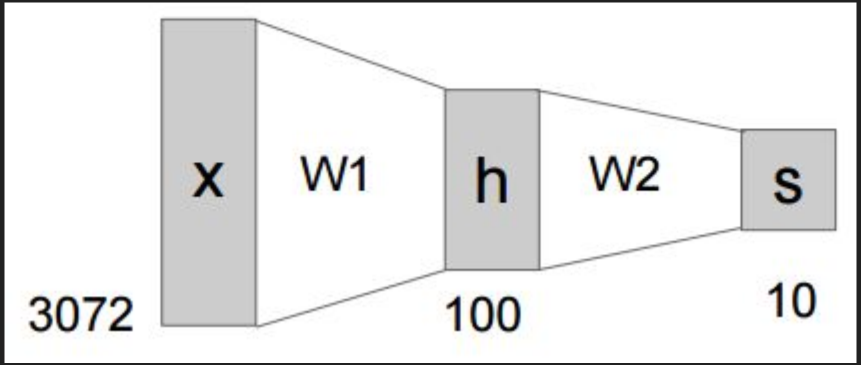
- Definimos una una función de loss para cuantificar el error
- Ajustamos nuestros pesos W para minimizar la loss



Redes neuronales

- Función lineal: $f = W * x$

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

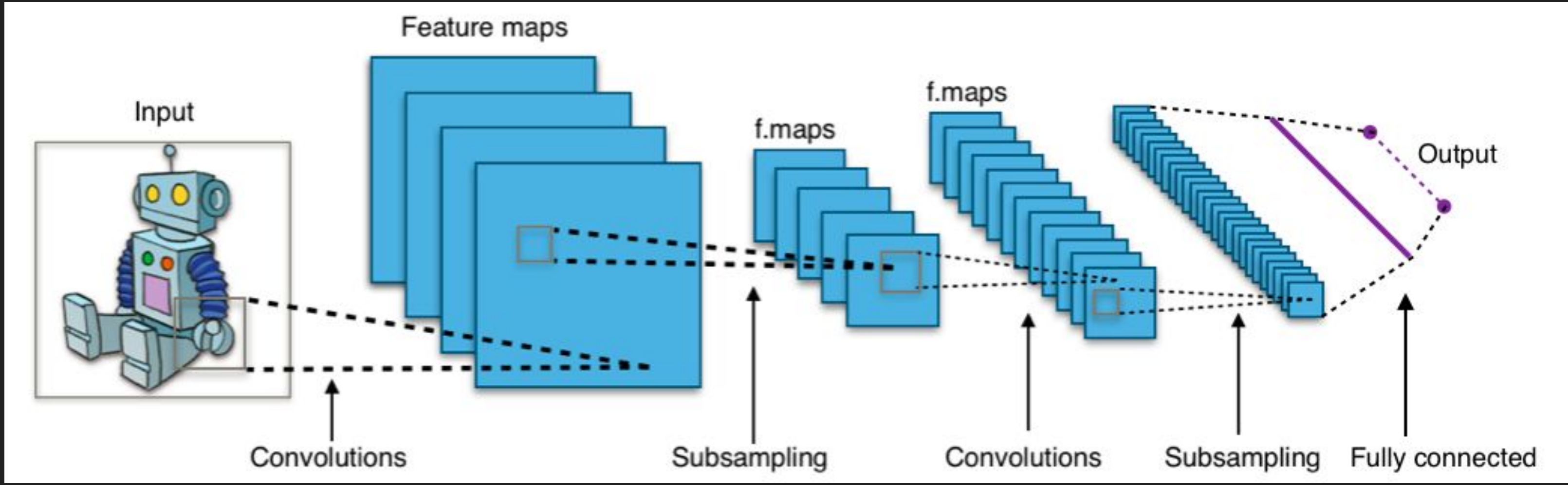


- Red neuronal de tres capas: $f = W'' * \tanh(W' * \max(0, W * x))$

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

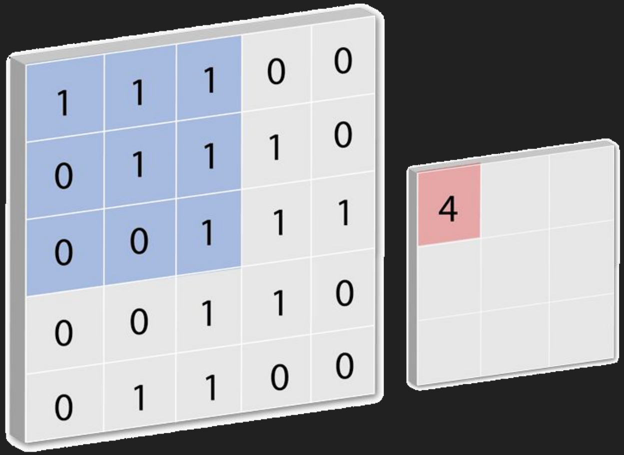
Redes convolucionales (CNN)

- 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)



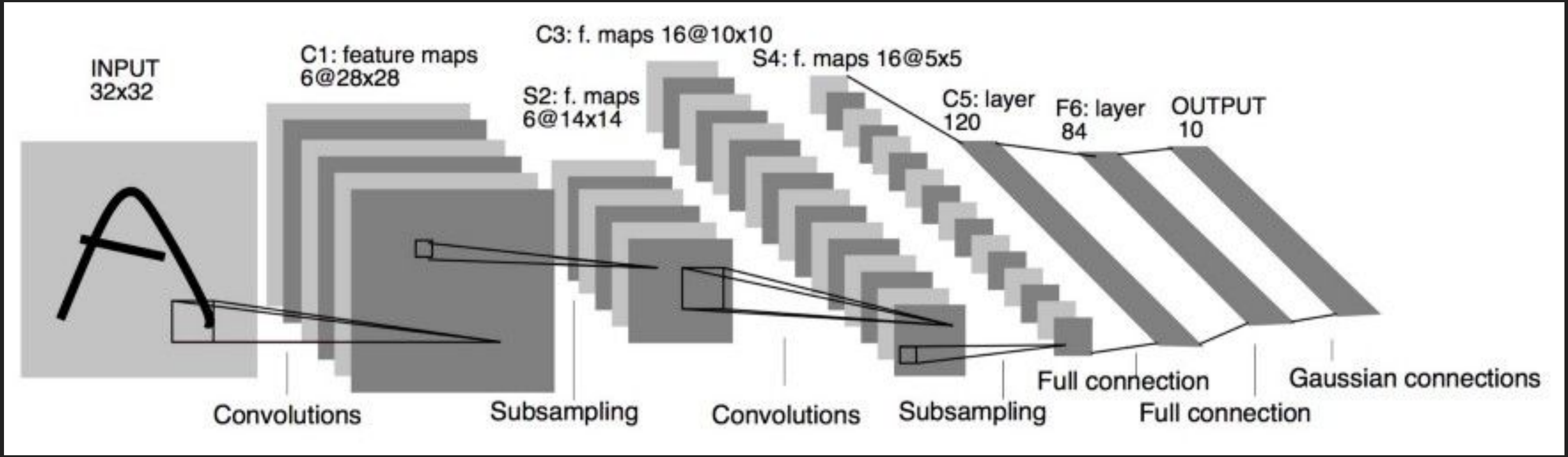
CNN

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



CNN: LeNet

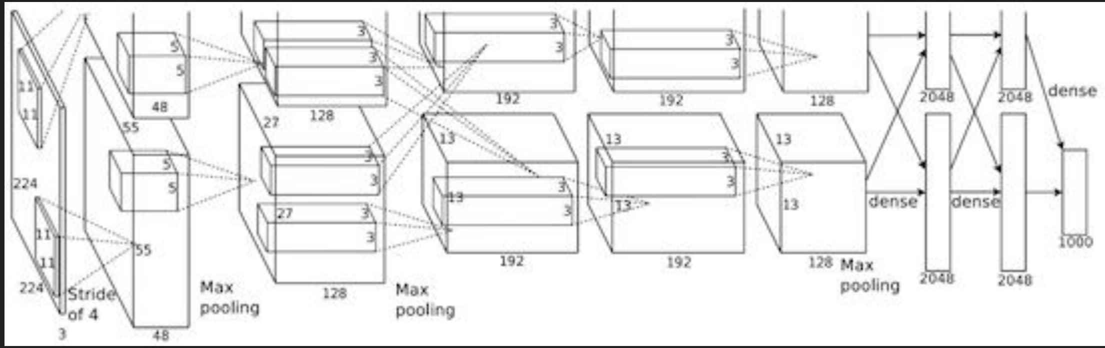
- ‘88 -> ‘94



(Error rate top 5)

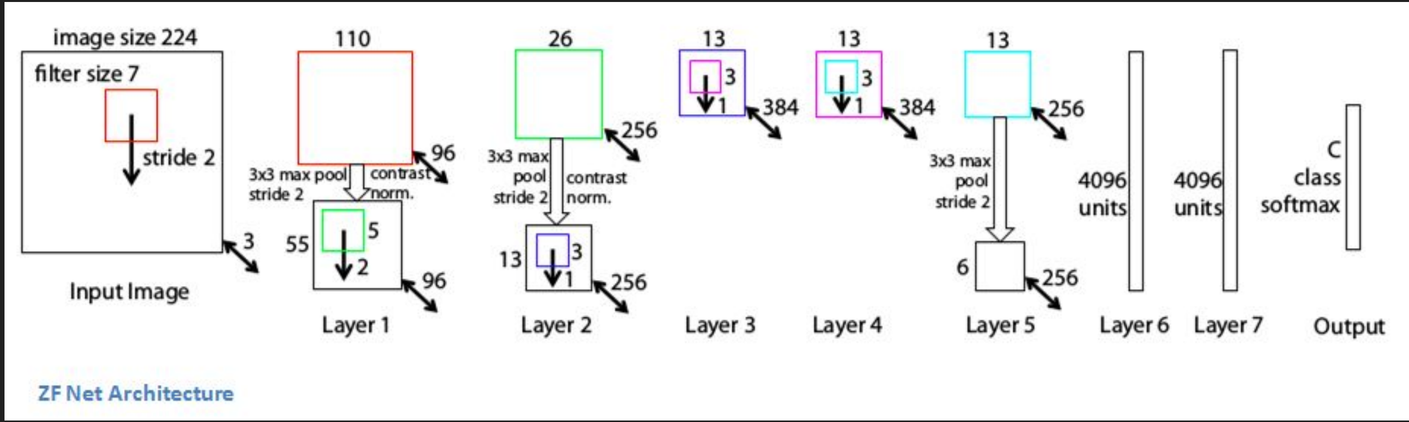
CNN: AlexNet

- 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)

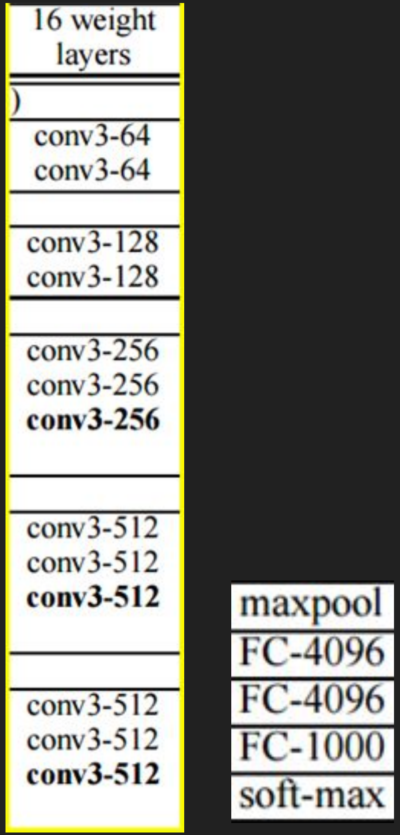


CNN: VGG Net

- 7.3% Error rate
- Punto fuerte: profundidad

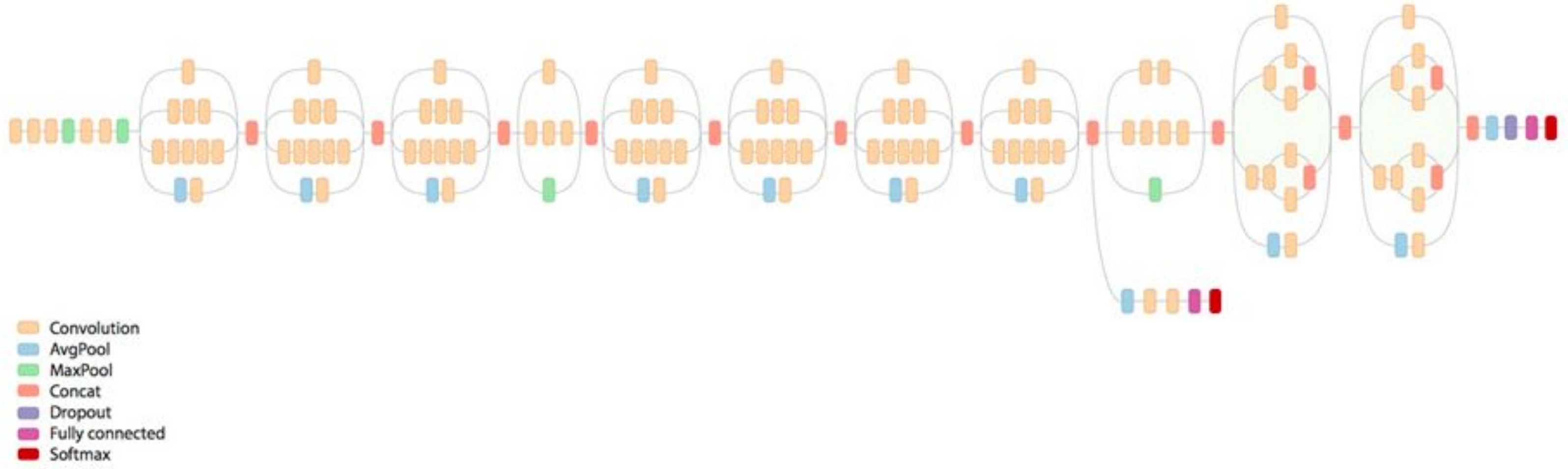
- Convoluciones más chicas

- 4 Nvidia Titan Black GPUs (2->3 semanas)



CNN: Google Net

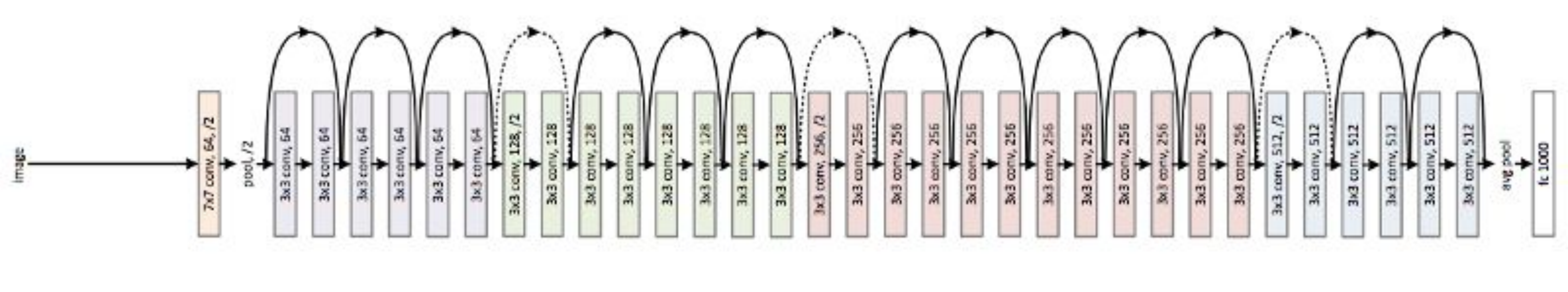
- 6.67% Error Rate
- > 100 capas
- Paralelización
- Entrenada en “algunas pocas GPUs” en una semana



Another view of GoogLeNet's architecture.

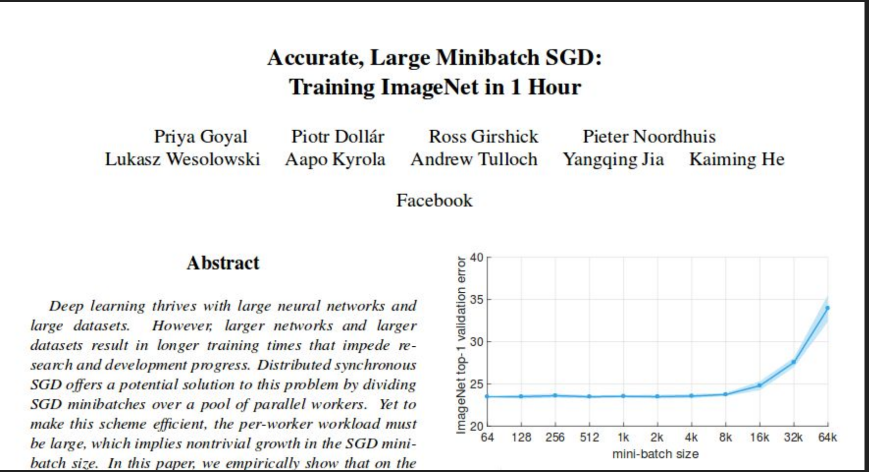
CNN: ResNet

- 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

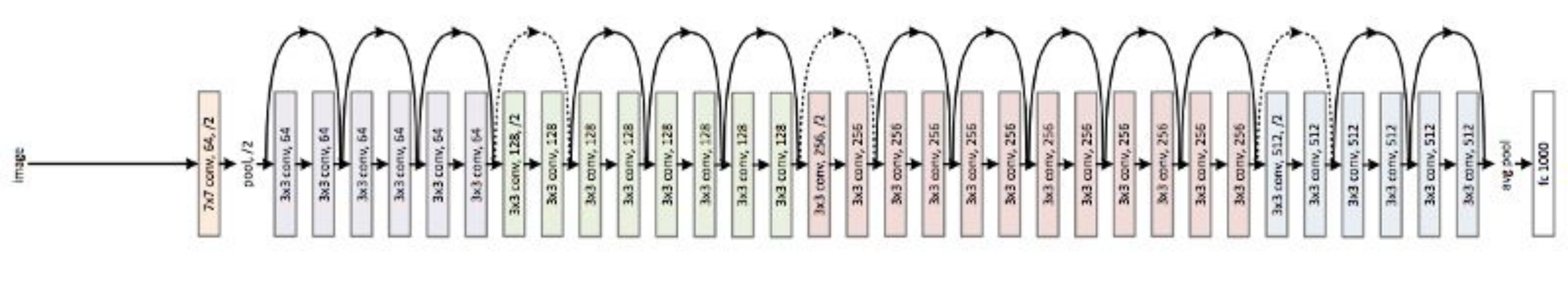
Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis
Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

Facebook

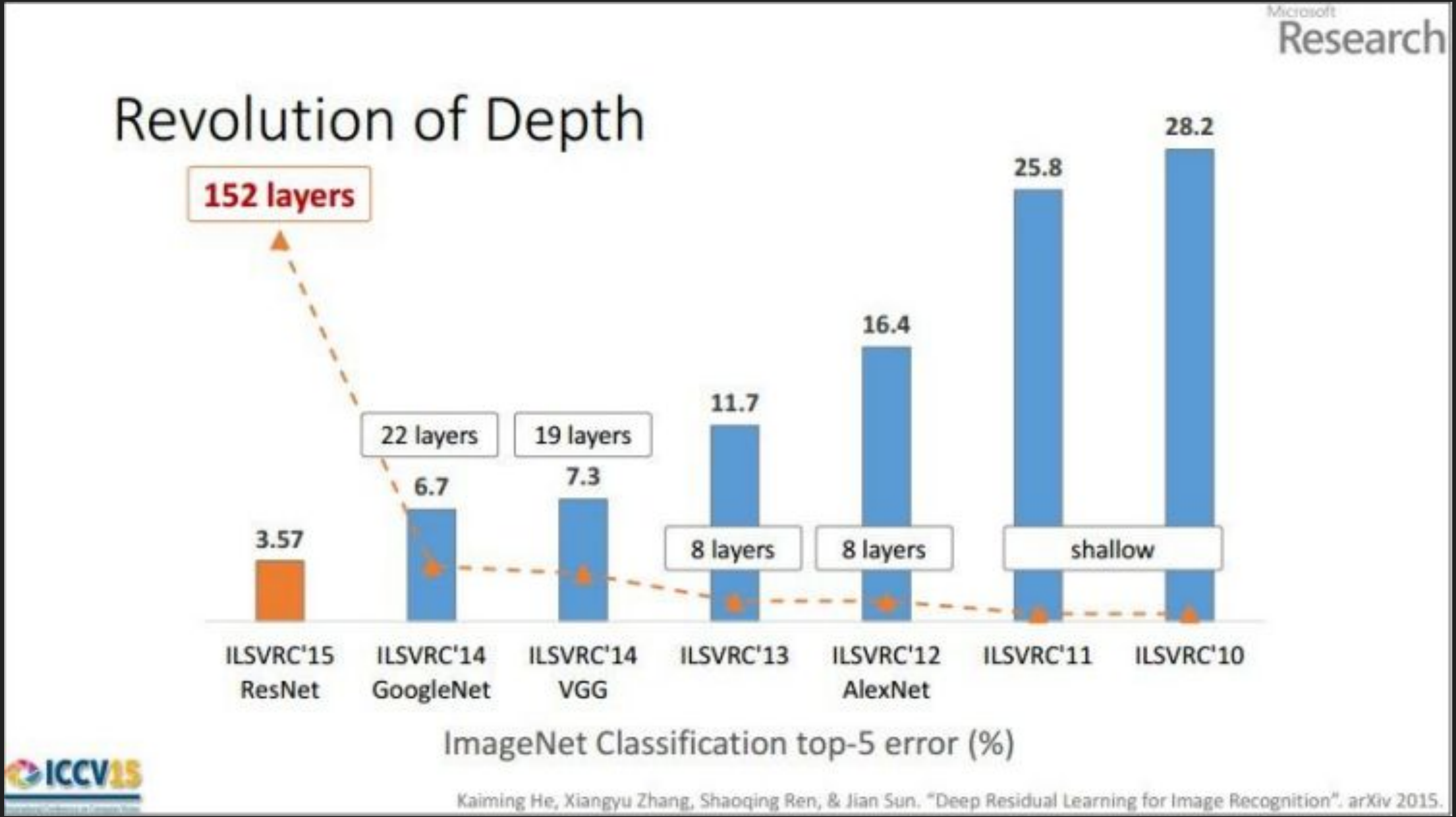
Abstract

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


Microsoft
Research
Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)
ICCV15
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

ConvNets, ConvNets everywhere

Retrieval

Detección

ConvNets, ConvNets everywhere

Segmentación

Self-driving cars

ConvNets, ConvNets everywhere

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_Tj8CRmurZ8Kk1gEBc7fg

Muchas gracias!

Preguntas?