# Entreposage et fouille de données (STA211)
# Neural Networks and Deep Learning

**Nicolas Thome**
Conservatoire Nationnal des Arts et Métiers (CNAM)
Laboratoire CEDRIC - équipe Vertigo

le cnam

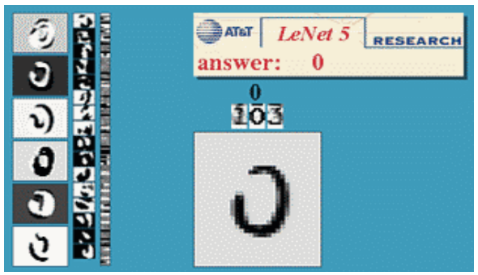
Cédric

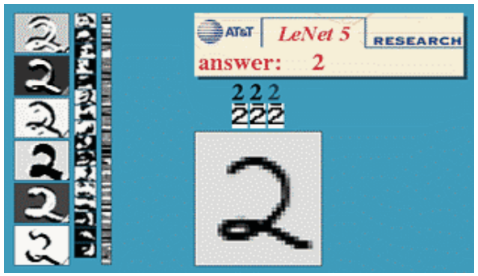
# Outline

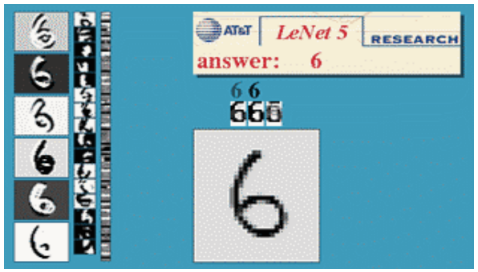
# 80's: LeNet 5 Model

- Evaluation on MNIST
- Total # parameters ~ 60000
  - 60,000 original datasets: test error: 0.95%
  - 540,000 artificial distortions + 60,000 original: Test error: 0.8%

- **Successful deployment for postal code reading in the US**



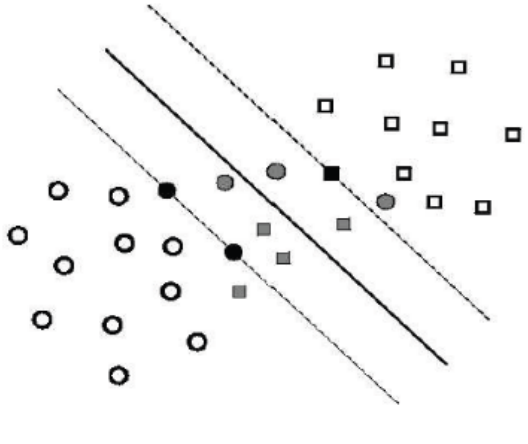



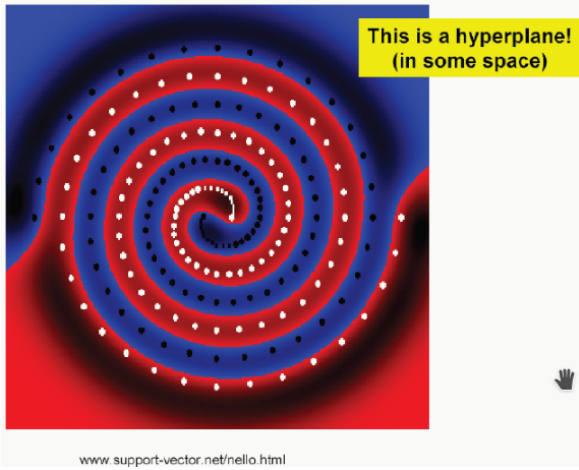



# Deep Learning: Trends and methods in the last four decades

## 90's: start of winter for deep learning

- Deep neural nets =' black magic', black boxes
  - Lack of interpretability
  - Optimization issues for highly non-convex objective function
- **Golden age of kernel methods**
  - Generalization theory with Support Vector Machines
  - Extension to non-linear modes: kernel trick
    - Kernel encode prior knowledge (structure) on data
  - Convex optimization problem

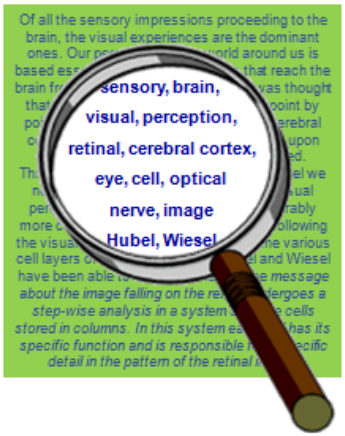




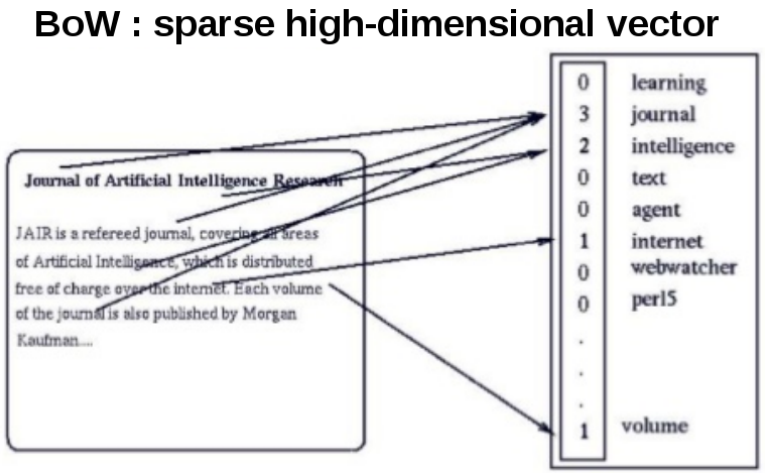
# Deep Learning: Trends and methods in the last four decades

## 2000's: Bag of Words Model (BoW)

- Started from the Information Retrieval (IR) community
- Text classification : document as a histogram of word occurrences





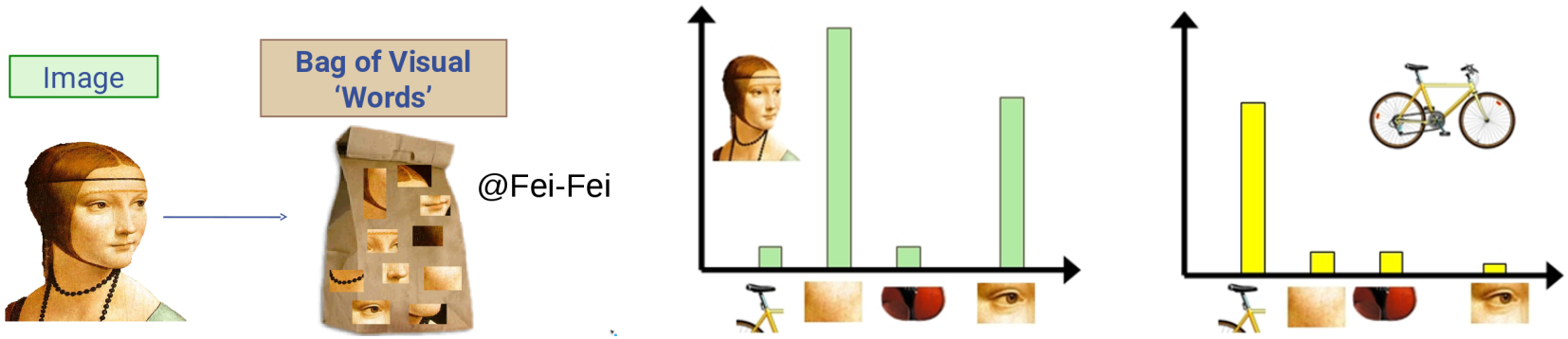



- Bow representation as input for powerful classifiers, *e.g.* SVM

# Deep Learning: Trends and methods in the last four decades

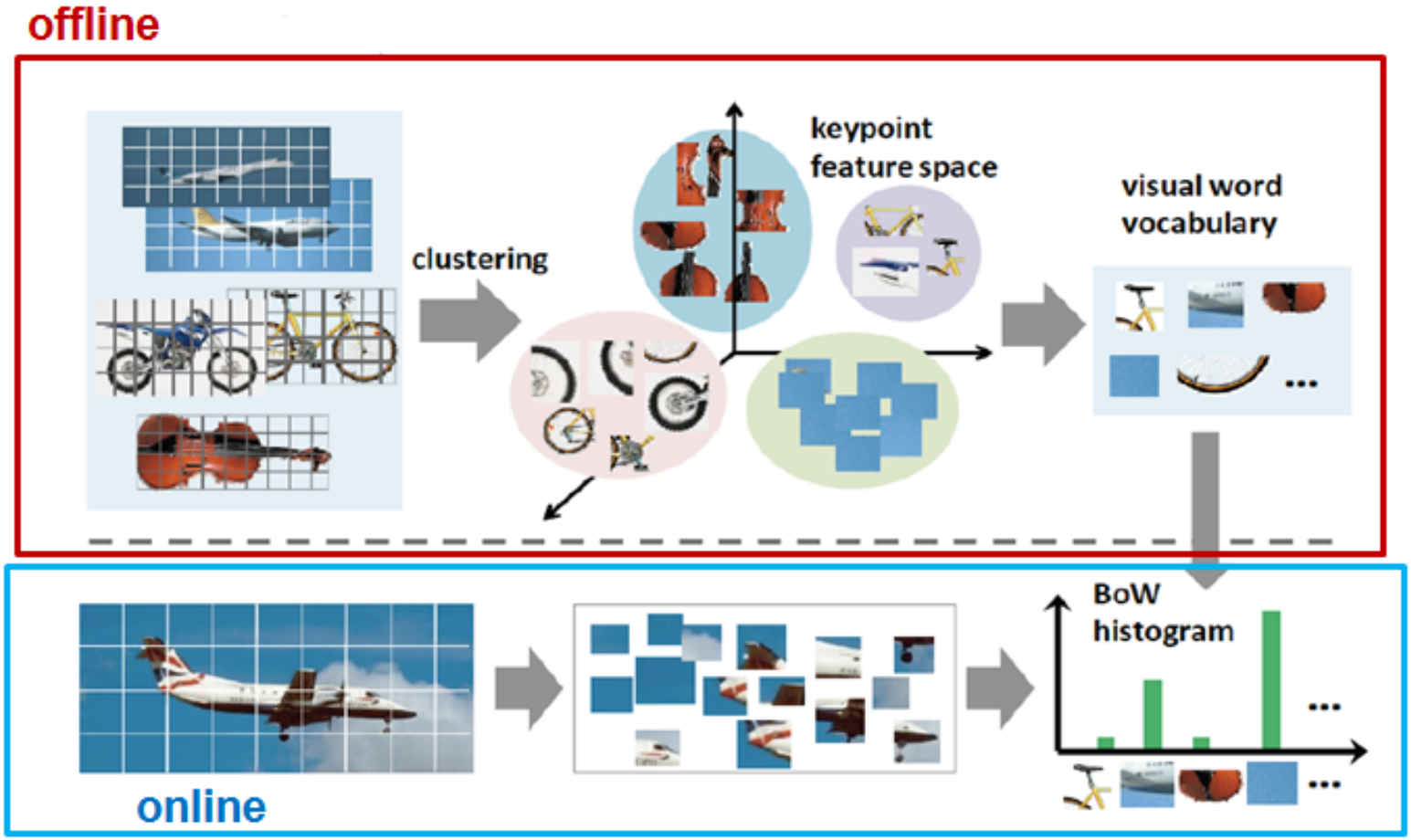
## 2000's: Bag of Words Model

- Adapting the BoW model for visual recognition ?
  ⇒ Bag of Visual Word (BoV)
- Main challenge: definition of visual words unclear!




- Solution: compute a dictionary on local image regions (clustering)
  - Local regions represented by handcrafted descriptors, *e.g.* SIFT

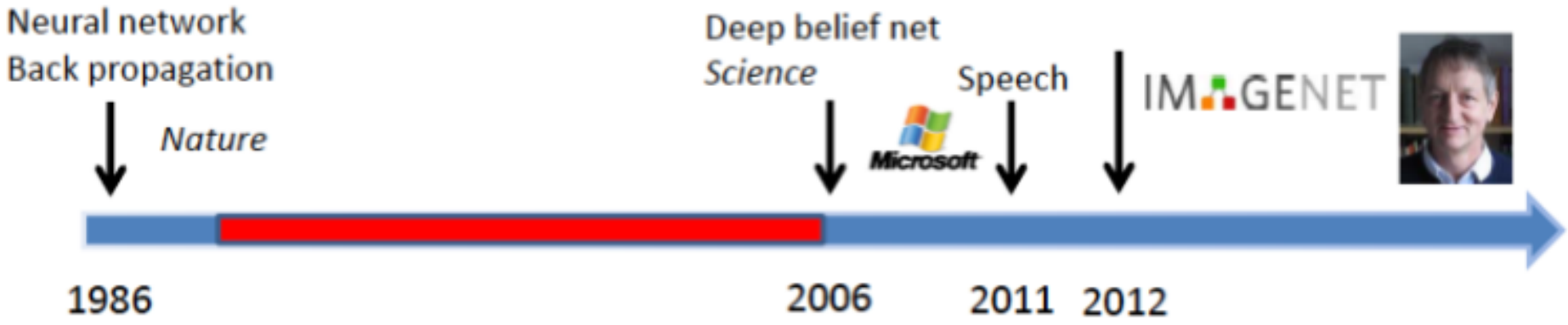
# 2000's: Bag of Visual Words Model




- 2000's: BoW + SVM state-of-the-art
- Many works on kernel on BoW, coding & pooling → 2012

# Deep Learning: Trends and methods in the last four decades

## Deep Learning renewal since 2006

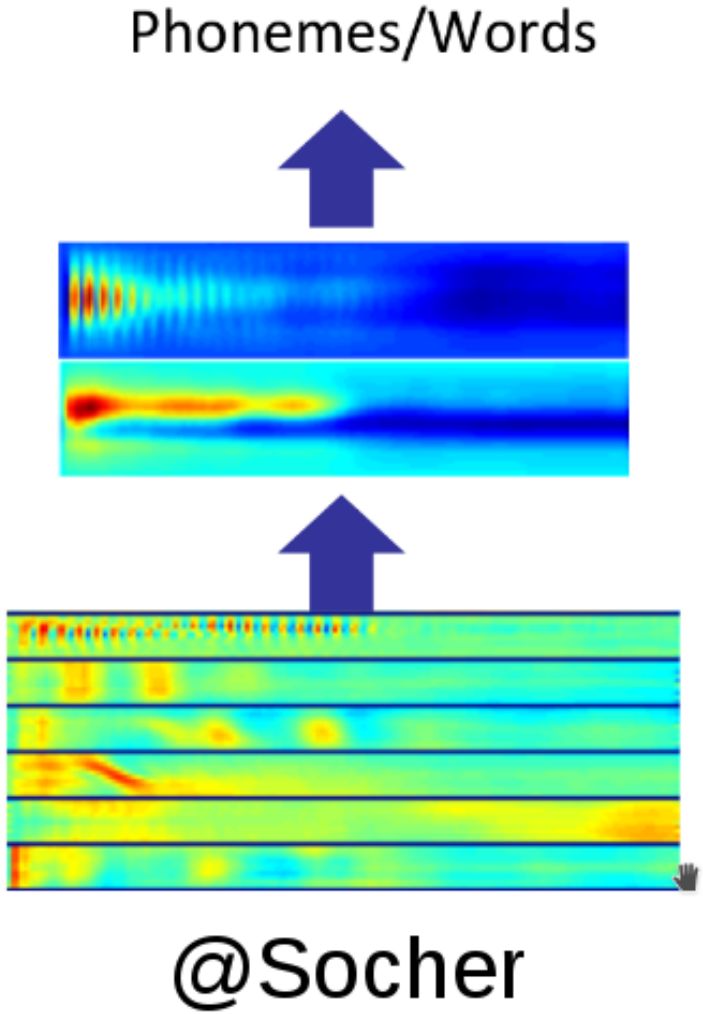



- 2006: new unsupervised learning for Deep Belief Nets (DBN) [HOT06]
- Theoretical results for improving model quality with depth
- Unsupervised training used as init for supervised learning with back-prop

# Deep Learning and ConvNet for Speech Recognition

- First DL breakthrough on large datasets: speech recognition
- Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition, Dahl et al. (2010)

| Acoustic model | Recog \ WER | RT03S FSH | Hub5 SWB |
|---|---|---|---|
| Traditional features | 1-pass –adapt | **27.4** | **23.6** |
| Deep Learning | 1-pass –adapt | **18.5** (–33%) | **16.1** (–32%) |



# Deep Learning and ConvNet for Image Classification

- ImageNet ILSVRC Challenge (Stanford):
  - $1,200,000$ training images, $1,000$ classes, mono-label
  - Based on WordNet hierarchy (ontology)
  - Evaluation: top-5 error
- Up to 2012, leading approaches: BoW + SVM
- ILSVRC'12: the deep revolution $\Rightarrow$ outstanding success of ConvNets [KSH12]

| Rank | Name | Error rate | Description |
|---|---|---|---|
| 1 | **U. Toronto** | 0.15315 | Deep learning |
| 2 | U. Tokyo | 0.26172 | Hand-crafted features and learning models. Bottleneck. |
| 3 | U. Oxford | 0.26979 | |
| 4 | Xerox/INRIA | 0.27058 | |

# 2012: the deep revolution

## Deep ConvNet success at ILSVRC'12

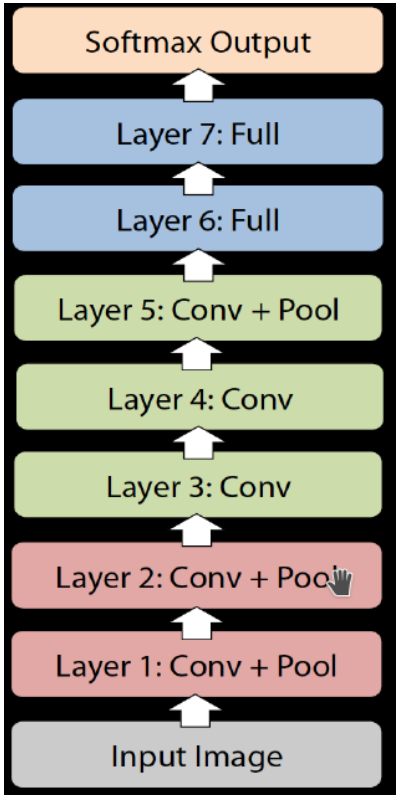
**Two main practical reasons:**

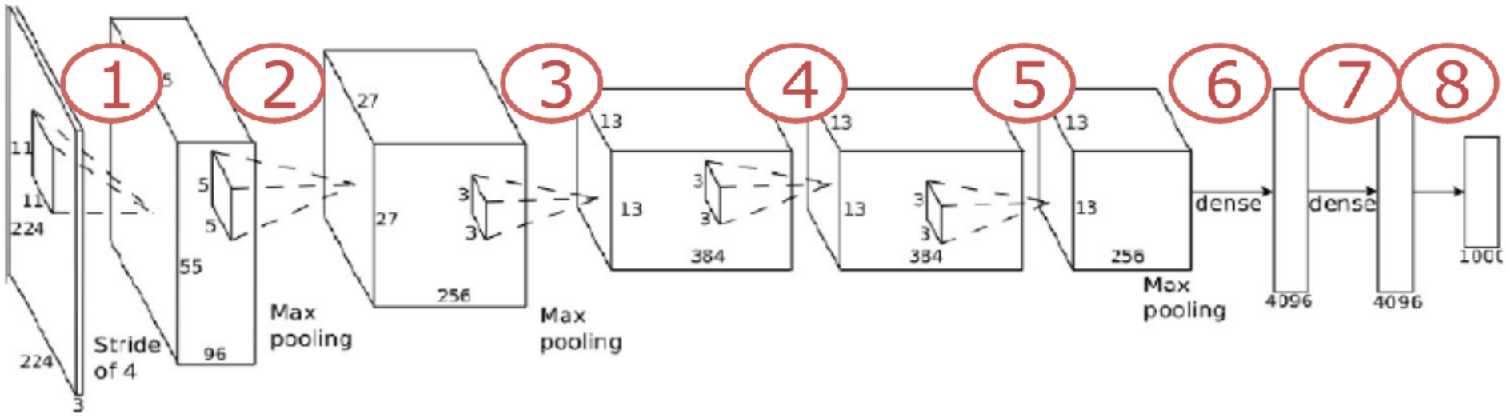
1. Huge number of labeled images ($10^6$ images)
   - Possible to train very large models without over-fitting
   - Larger models enables to learn rich (semantic) features hierarchies
2. GPU implementation for training
   - Relatively cheap and fast GPU
   - Training time reduced to 1-2 weeks (up to 50x speed up)

IMAGENET

# AlexNet [KSH12] in ILSVRC'12

- 60,000,000 parameters
- 650,000 neurons - 630,000,000 connections
- 5 convolutional layers, 3 Fully Connected (FC)
  - Convolution layer: Convolution + non linearity (ReLU) + pooling
  - Full= FC + non linearity - Final FC: 4096-dim
- Trained on 2 GPUs for a week

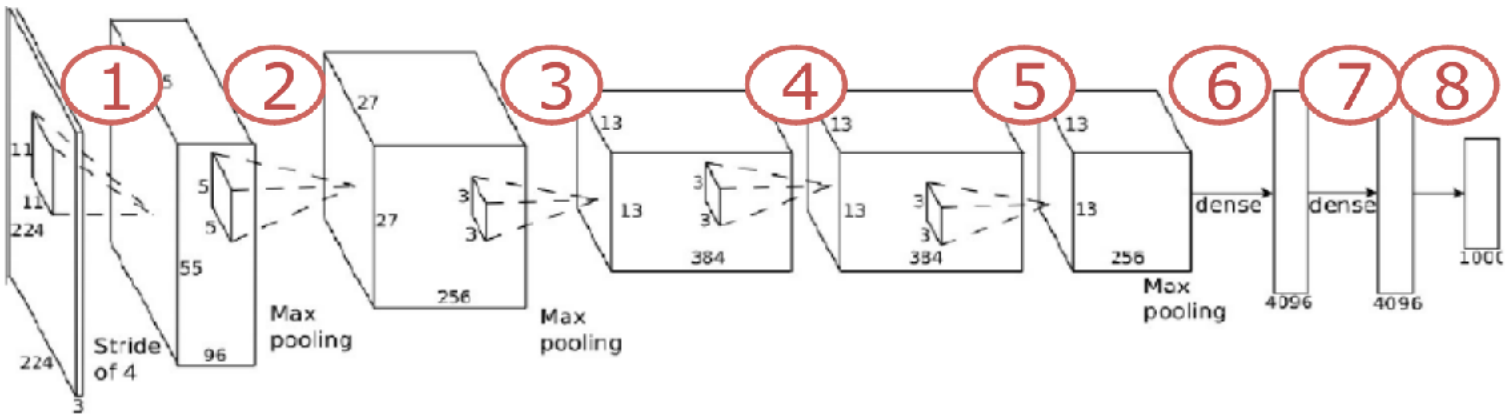


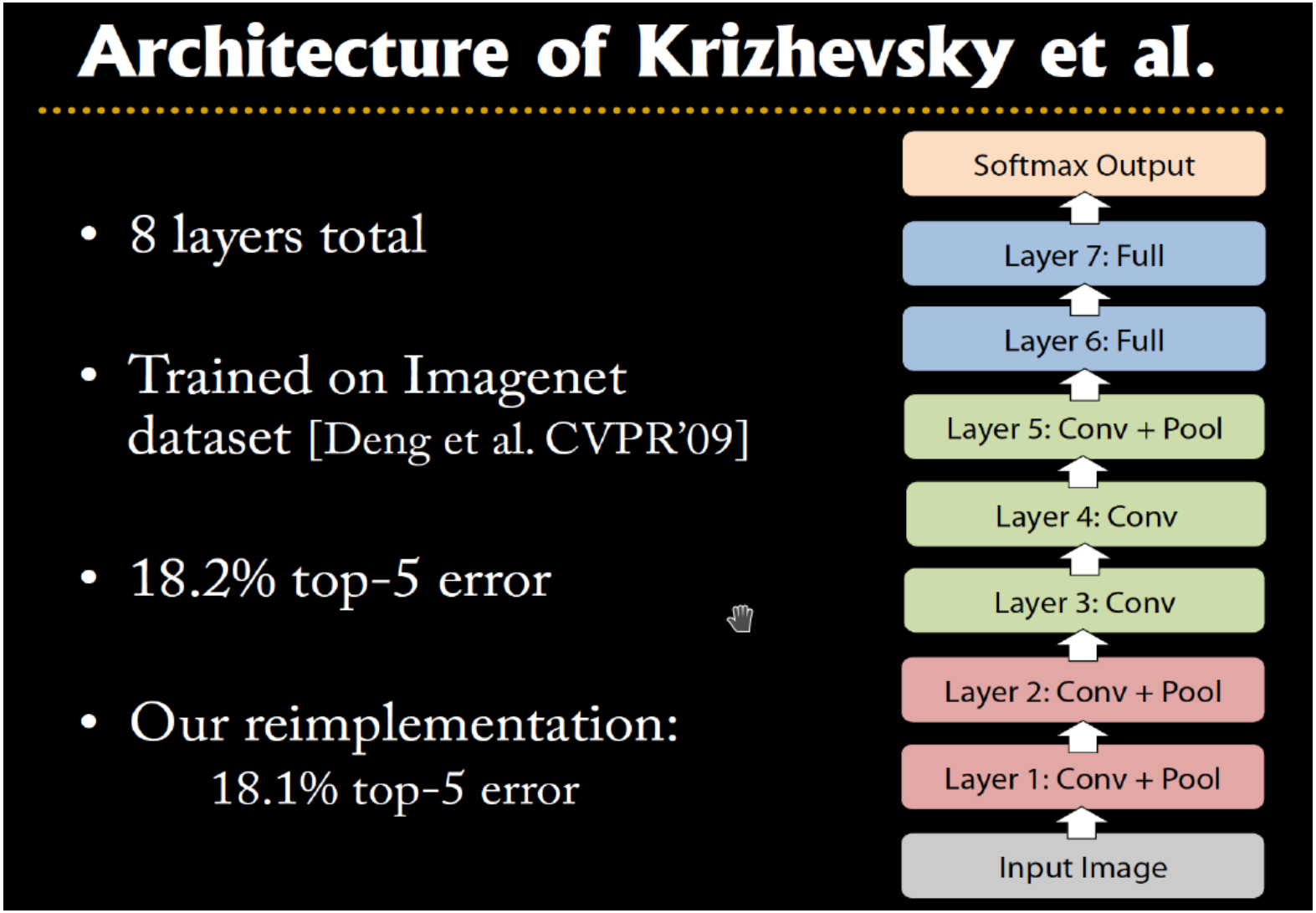
# AlexNet [KSH12] in ILSVRC'12

## First Convolutionnal Layer

- Input: Images: 227x227x3
- Filter (receptive field) size F: 11, S (stride) = 4
- 96 filters ⇒ output size 55*55*96 = 290,400 neurons
- Each Filter: 11*11*3 = 363 weights + 1 bias = 364 params
  - N.B.: Convolution in whole feature map depth (*cf* LeNet 5 discussion)
- # parms: 96 * 364 = 34, 944

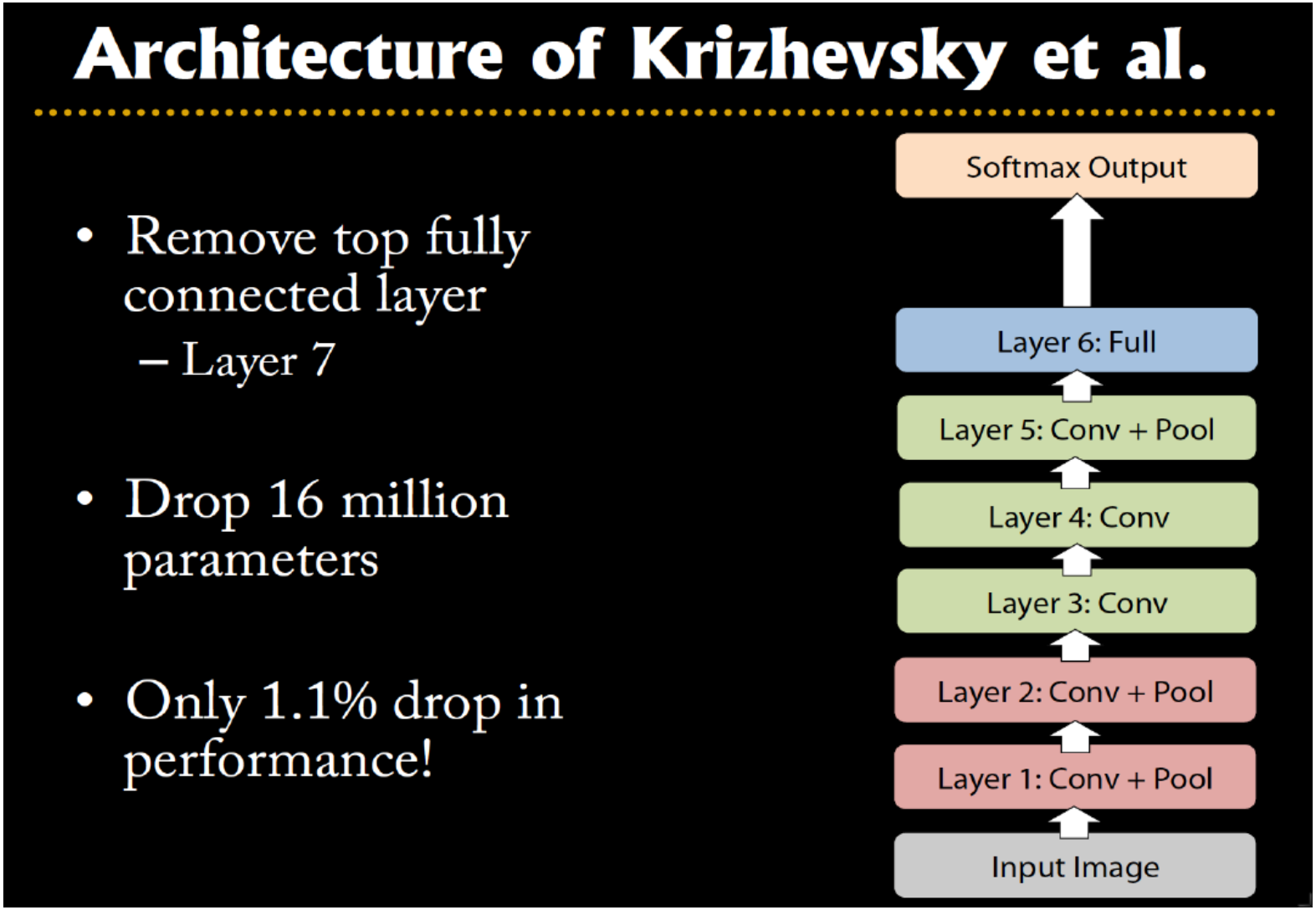


# AlexNet [KSH12] in ILSVRC'12




Credit: R. Fergus

# AlexNet [KSH12] in ILSVRC'12

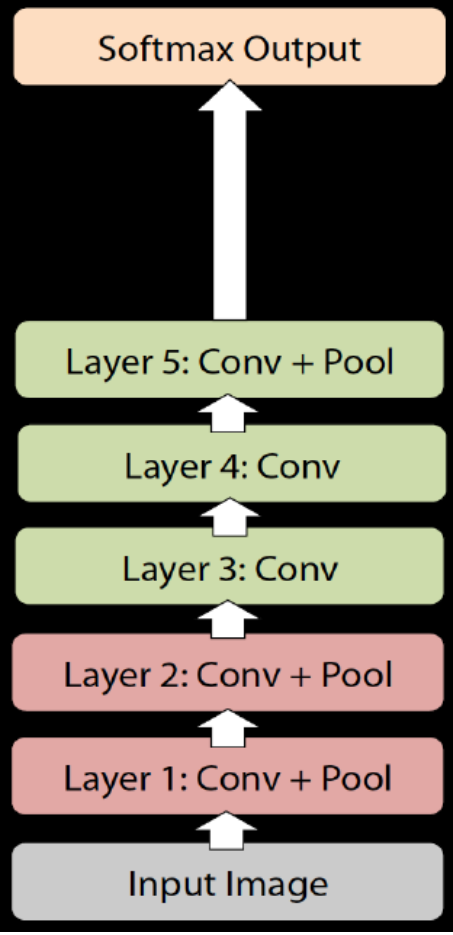



Credit: R. Fergus

# AlexNet [KSH12] in ILSVRC'12

## Architecture of Krizhevsky et al.

- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance



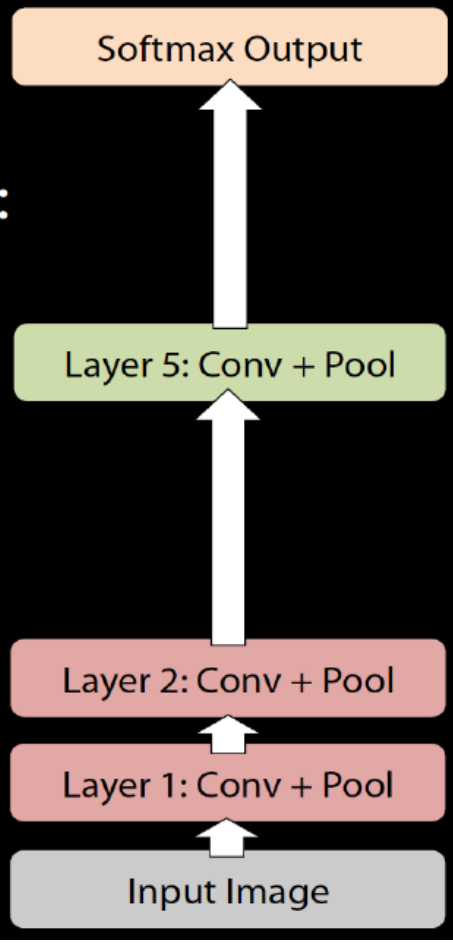

Credit: R. Fergus

# AlexNet [KSH12] in ILSVRC'12

## Architecture of Krizhevsky et al.

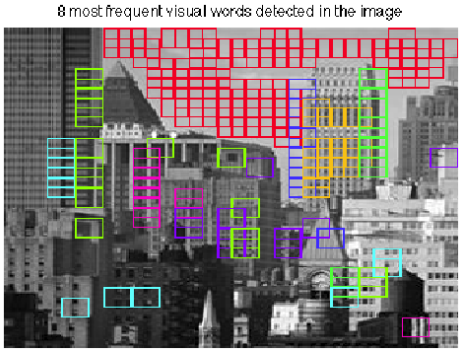
- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6 ,7
- Now only 4 layers
- 33.5% drop in performance

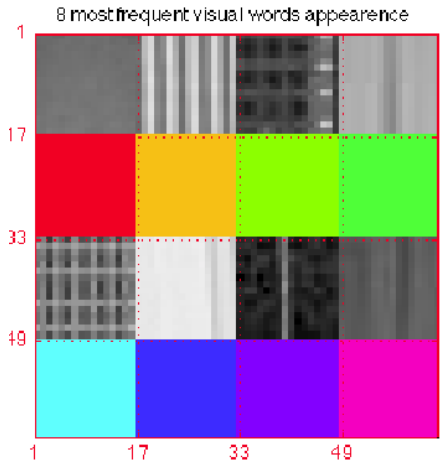
→Depth of network is key



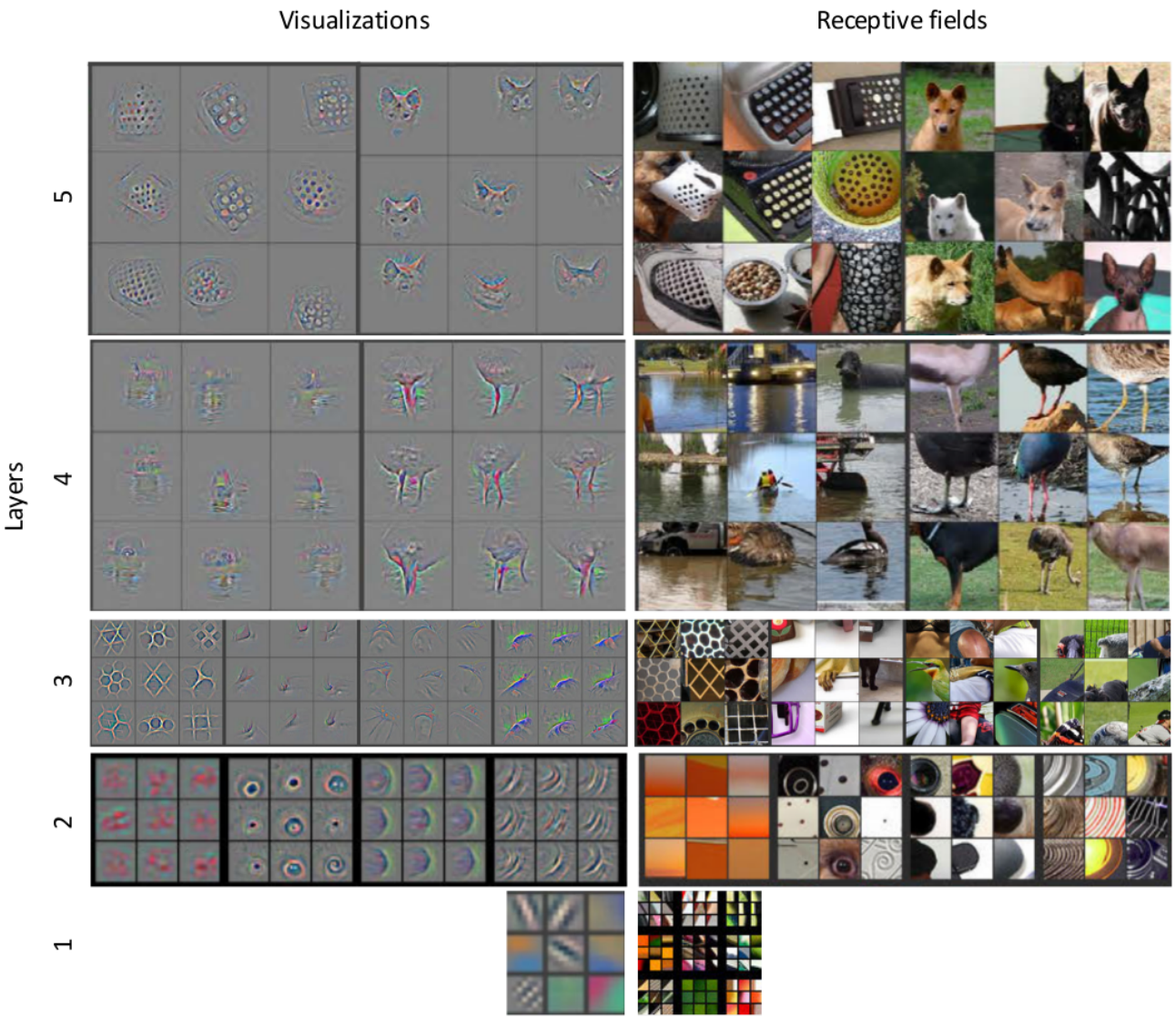

Credit: R. Fergus

# Deep Learning in 2012: Representation Learning

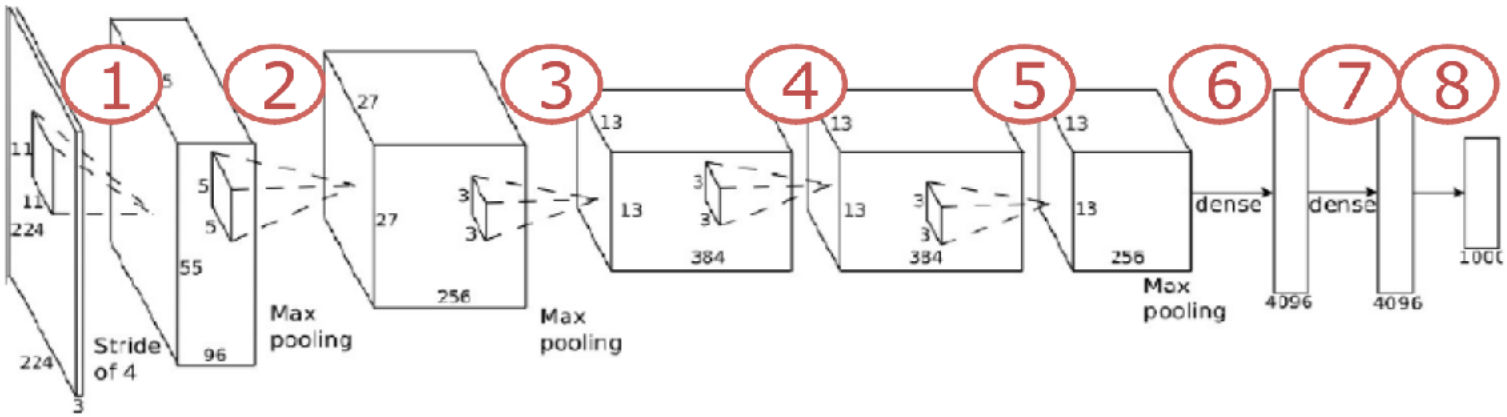




## Deep: more semantic features



# AlexNet [KSH12] in ILSVRC'12

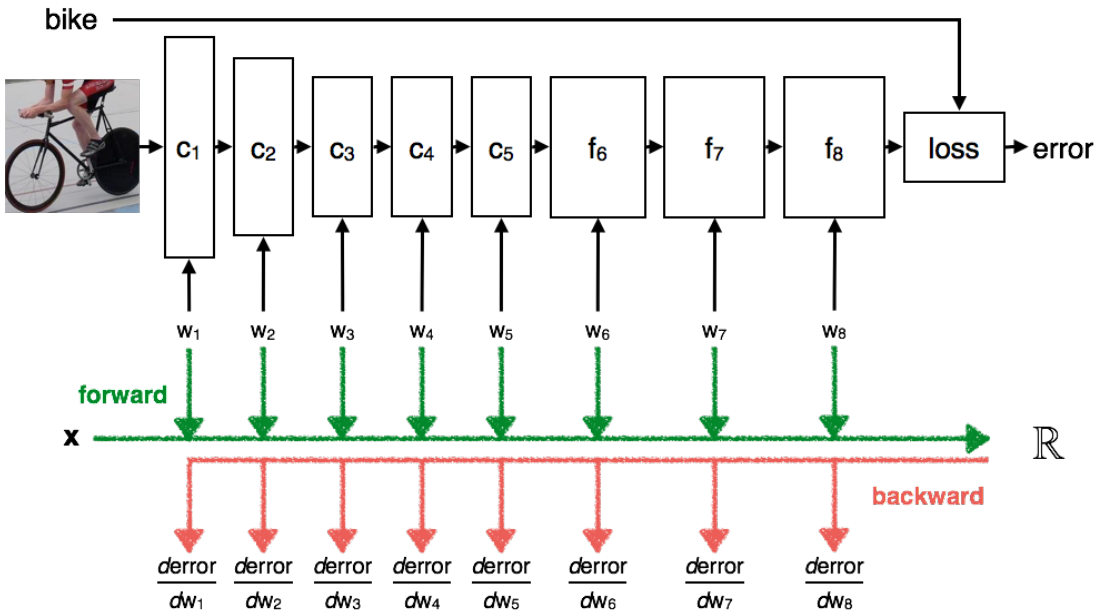
- Same global architecture as older nets, *e.g.* LeNet
  - Trained with back-prop and stochastic gradient descent
- But bigger (deeper and wider): 60 $10^6$ parameters *vs* 60 $10^3$
  - Needs more data ($10^6$ *vs* $10^4$)
  - GPU implementation for fast training
- Also some architectural and optim improvements (see next course):
  - Non-linearity: ReLU *vs* sigmoid
  - Regularization: data augmentation, dropout
  - Overlapping pooling (Local Response Normalisation, LRN)



# Outline

# Deep Learning: resources for the community

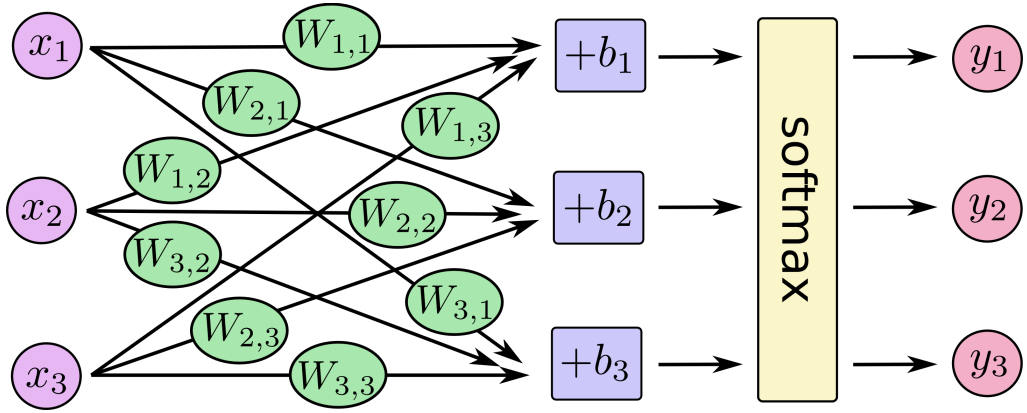
- Formal training of deep CNNs straightforward (backprop)
- Efficient CNN implementation far from trivial, especially convolution
- Libraries made available in the community:
  - Caffe / Decaf : script, non modular
  - MatConvNet (matlab): easy, Torch (Lua): efficiency, modularity
  - TensorFlow / Theano / PyTorch (python): auto-differentiation
  - Keras: wrapper on top of TensorFlow / Theano



# Keras: simple example

## Logistic Regression

$$p(\hat{y_{c,i}}|x_i)) = \frac{e^{\langle x_i; w_c \rangle + b_c}}{\sum_{c'=1}^{K} e^{\langle x_i; w_{c'} \rangle + b_{c'}}}$$




- Example in MNSIT: $K = 10$ classes
- # parameters: $784 * 10 + 10 = 7850$

# Keras: simple example

## Logistic Regression

- Define an (empty) feedfoward network

```
from keras.models import Sequential
model = Sequential()
```

- Add fully connected layer (size 10) + softmax activation

```
from keras.layers import Dense, Activation
model.add(Dense(10, input_dim=784, name='fc1'))
model.add(Activation('softmax'))
```

# Keras: simple example

## Logistic Regression

- Compile model with cross-entropy loss

```
from keras.optimizers import SGD
learning_rate = 0.5
sgd = SGD(learning_rate)
model.compile(loss='categorical_crossentropy',optimizer=sgd,metrics=['accuracy'])
```

- Optimize model parameters to fit training data (*e.g.* MNIST)

```
from keras.datasets import mnist
# MNIST data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# + some pre-processing ... and fit model to data
model.fit(X_train, y_train,batch_size=128, epochs=20,verbose=1)
```

# Keras: more complex examples

- Design more complex model by adding layers : fully connected, convolution, non-linearity, pooling, *etc*
- Code for training remains unchanged (back-prop does the job)

```
s=(5,5)
ish=(28,28,1)
model = Sequential()
model.add(Conv2D(32,kernel_size=s,activation='sigmoid',input_shape=ish,padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='sigmoid', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(100, activation='sigmoid'))
model.add(Dense(nb_classes, activation='softmax'))
```

# Deep Learning Modules

## Non-linearities

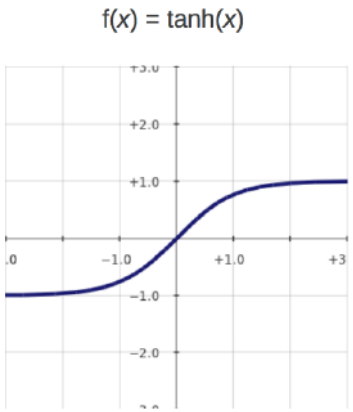
- Rectified Linear Unit (ReLU) [KSH12] : $y = 0$ if $x < 0$, $y = x$ otherwise
- Solving vanishing gradients problems $\Rightarrow$ faster learning / convergence

### RELU Nonlinearity

- Standard way to model a neuron

  $f(x) = tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$

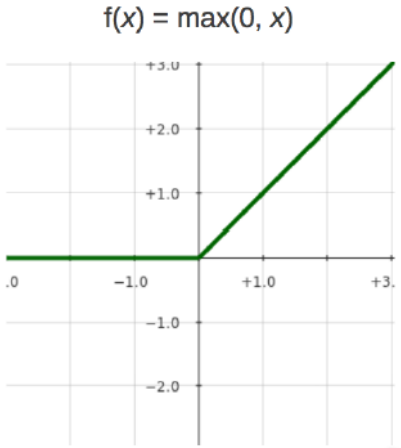
  Very slow to train




- Non-saturating nonlinearity (RELU)

  $f(x) = max(0, x)$

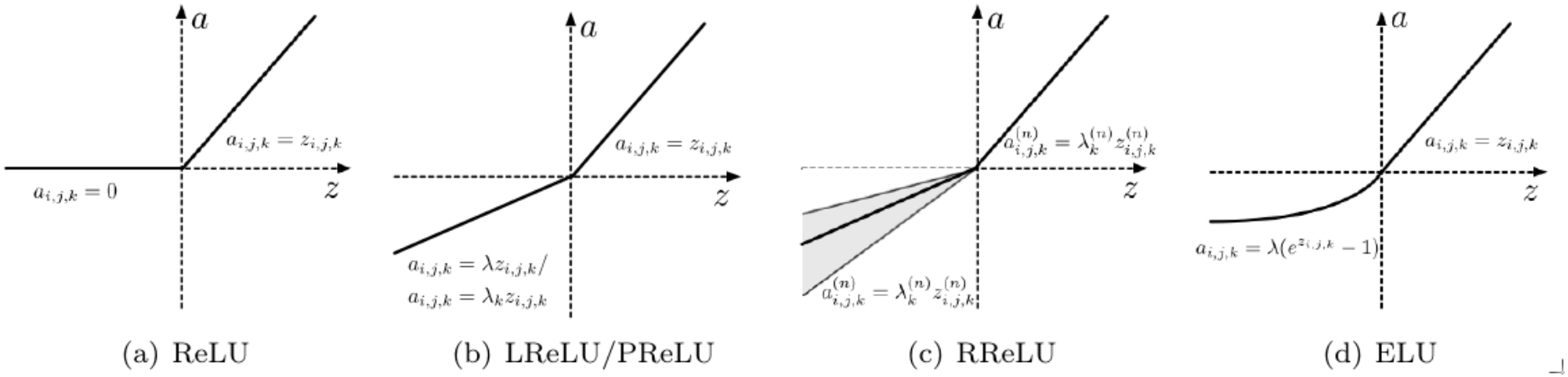
  Quick to train



# Deep Learning Modules

## Non-linearities, ReLU variants

- Leaky ReLU (LReLU): $\lambda$ is empirically predefined
- Parametric ReLU (PReLU) : $\lambda_k$ is learned from training data
- Randomized ReLU (RReLU): $\lambda_k^n$ is a random variable which is sampled from a given uniform distribution in training and keeps fixed in testing
- Exponential Linear Unit (ELU): $\lambda$ fixed



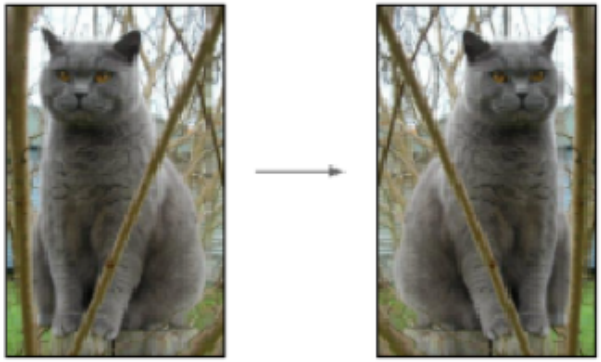

(a) ReLU (b) LReLU/PReLU (c) RReLU (d) ELU

From Gu *et. al.* [GWK+15]

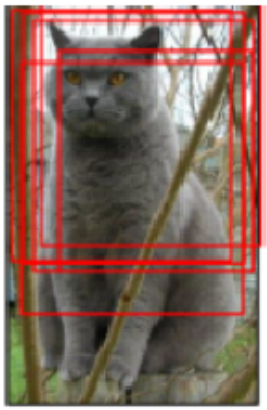
# Deep Learning Modules

## Training: data-augmentation

- Jittering, mirroring, color perturbation, rotation, stretching, shearing, lens distortions, *etc* of the original images
- Increases # training samples, adds robustness to irrelevant variations
- **Done in train AND in test**



Flip horizontally



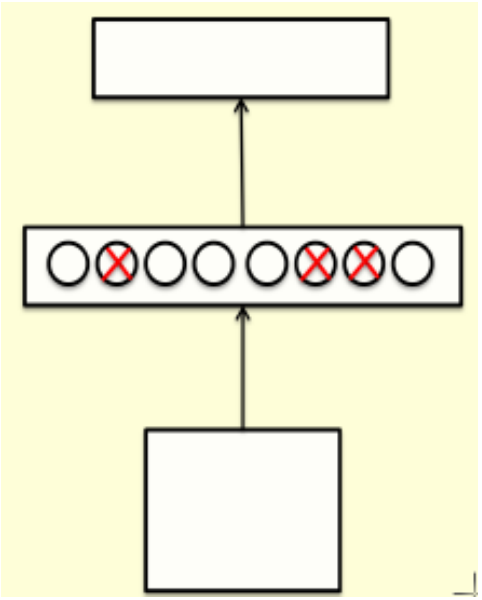
Random crops/scales

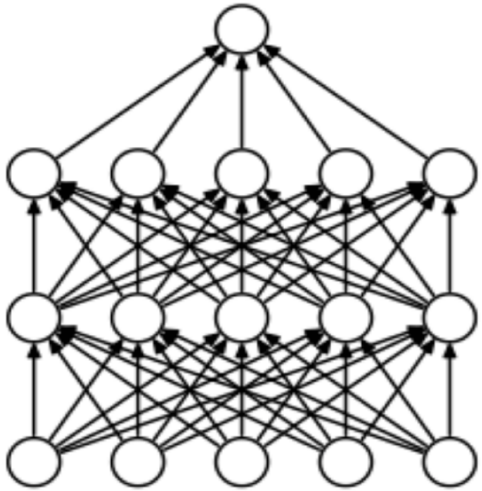
Color jittering

# Deep Learning Modules

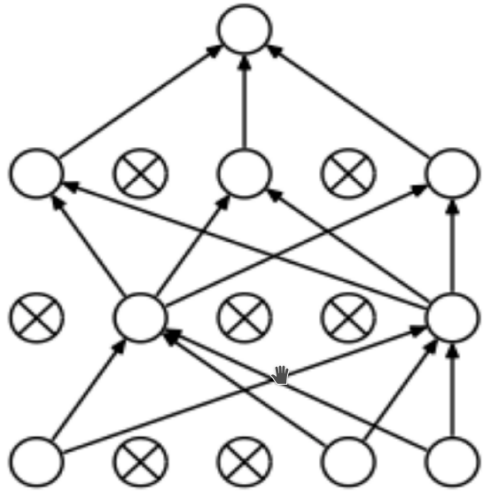
## Training: droupout [HSK+12]

- Randomly omit each hidden unit with probability 0.5
- **Regularization technique**, limits over-fitting (better generalization)
  - Pulls the weights towards what other models want, useful to prevent co-adaptation (feature only helpful when other specific features present)
  - May be viewed as averaging over many NN
  - Slower convergence





Standard Neural Net



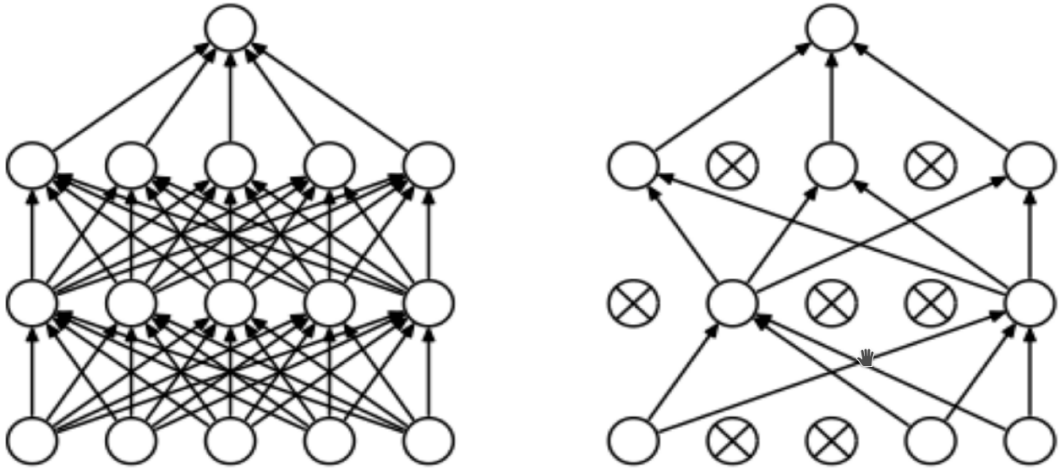
After applying dropout.

Credits: Geoffrey E. Hinton, NIPS 2012

# Deep Learning Modules

## Training: droupout

- What to do at test time ?
  - Sample many different architectures and take the geometric mean of their output distributions
  - Faster alternative: use all hidden units (but after halving their outgoing weights)
  - Equivalent to the geometric mean in case of single hidden layer
  - Pretty good approximation for multiple layers



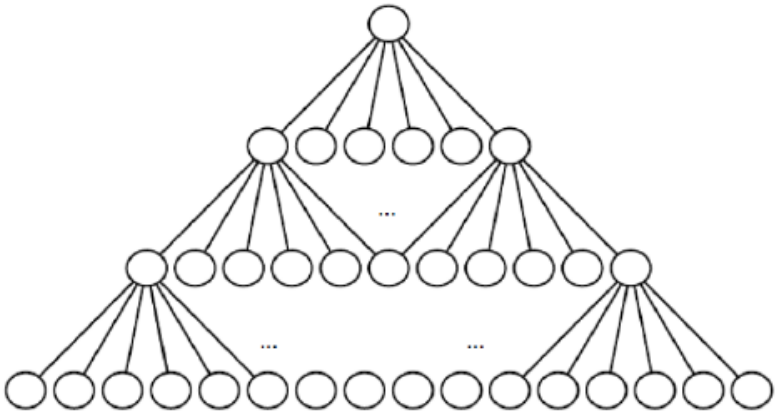
Standard Neural Net | After applying dropout.

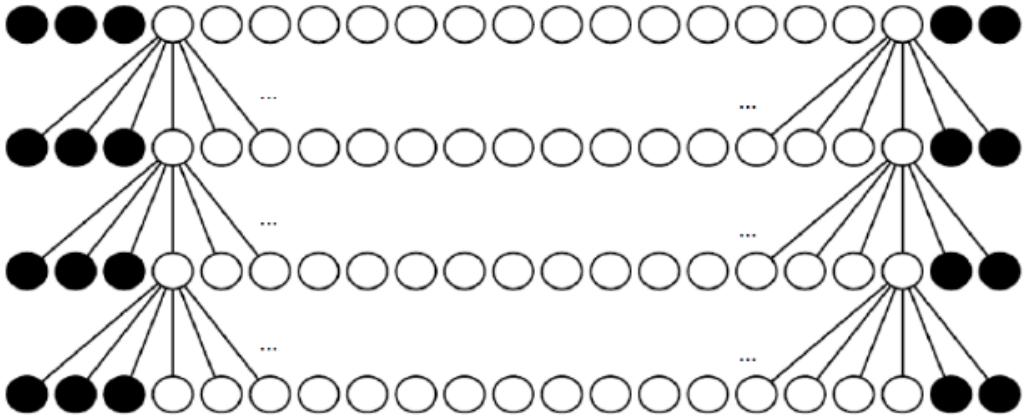
Credits: Geoffrey E. Hinton, NIPS 2012

# Deep Learning Modules

## Padding, *e.g.* zero-padding

- To avoid shrinking the spatial extent of the network rapidly

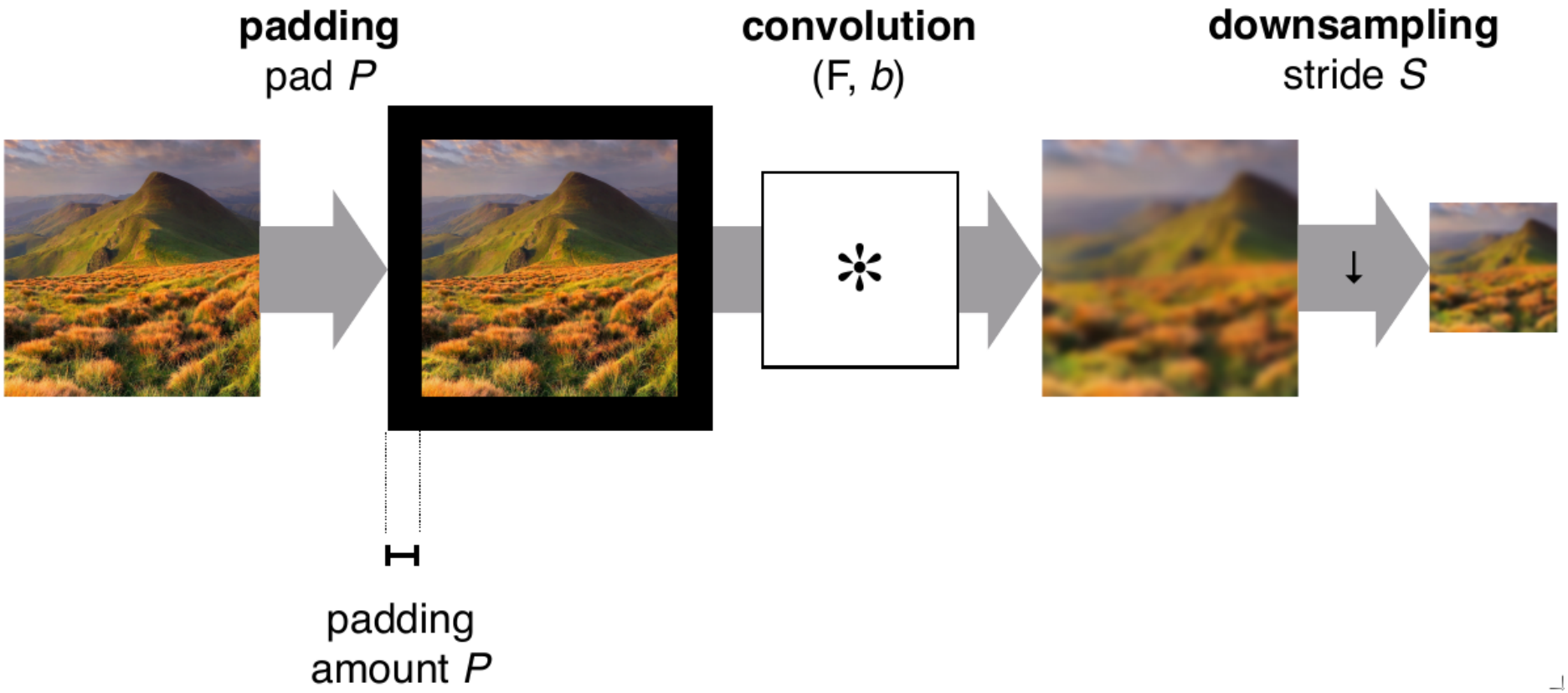




# Deep Learning Modules

## Padding, *e.g.* zero-padding

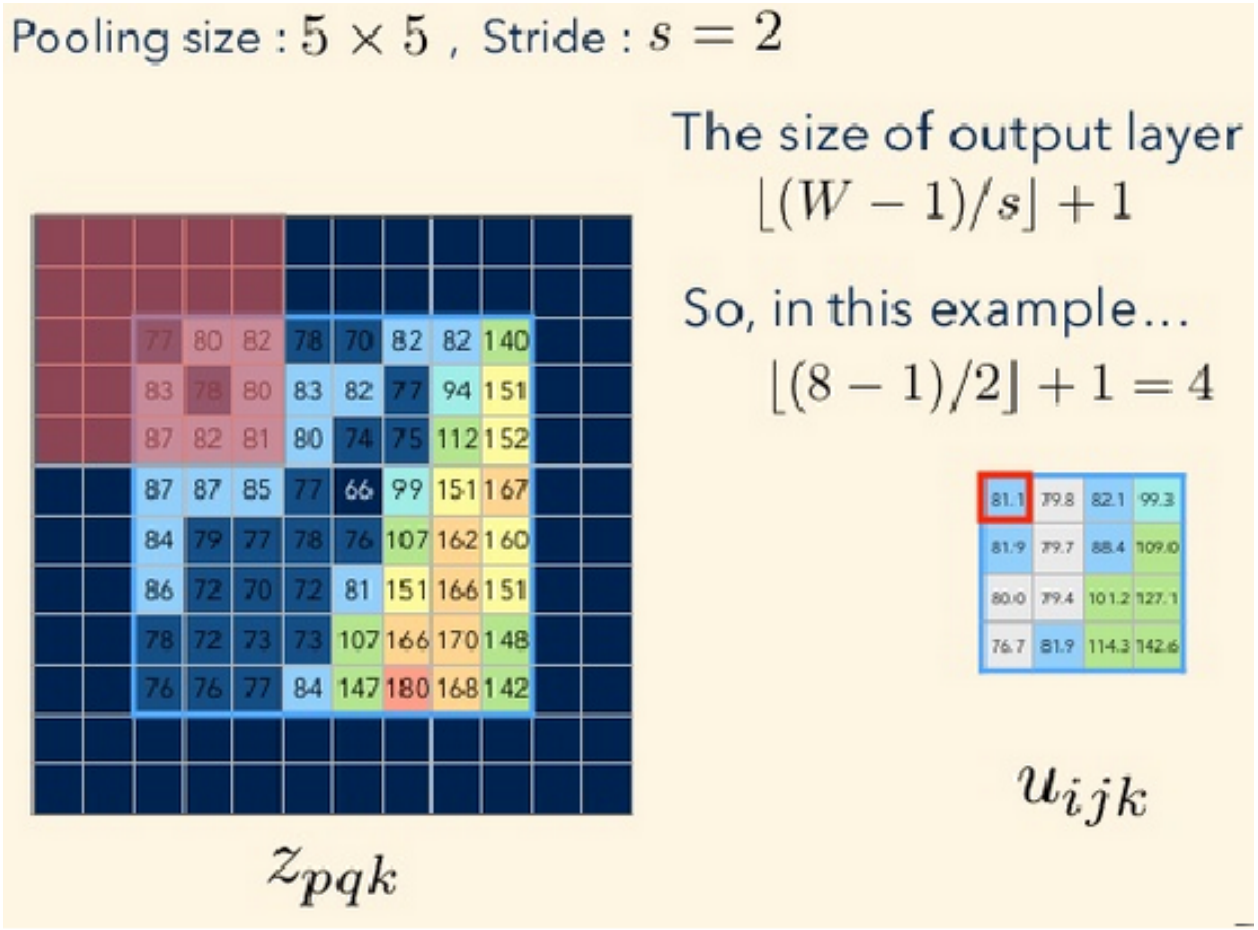
- Ex for images:




Credit: A. Vedaldi

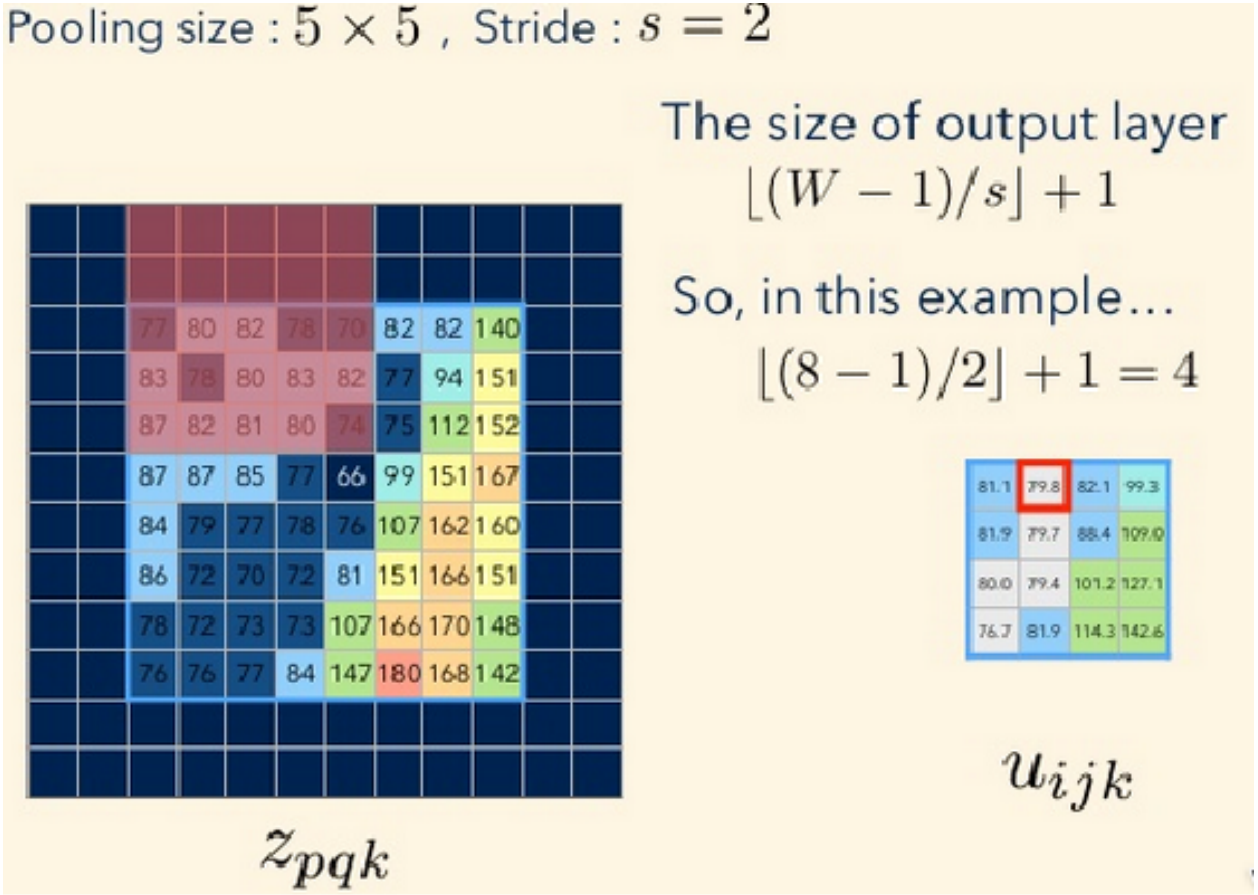
# Deep Learning Modules

## Overlapping Pooling



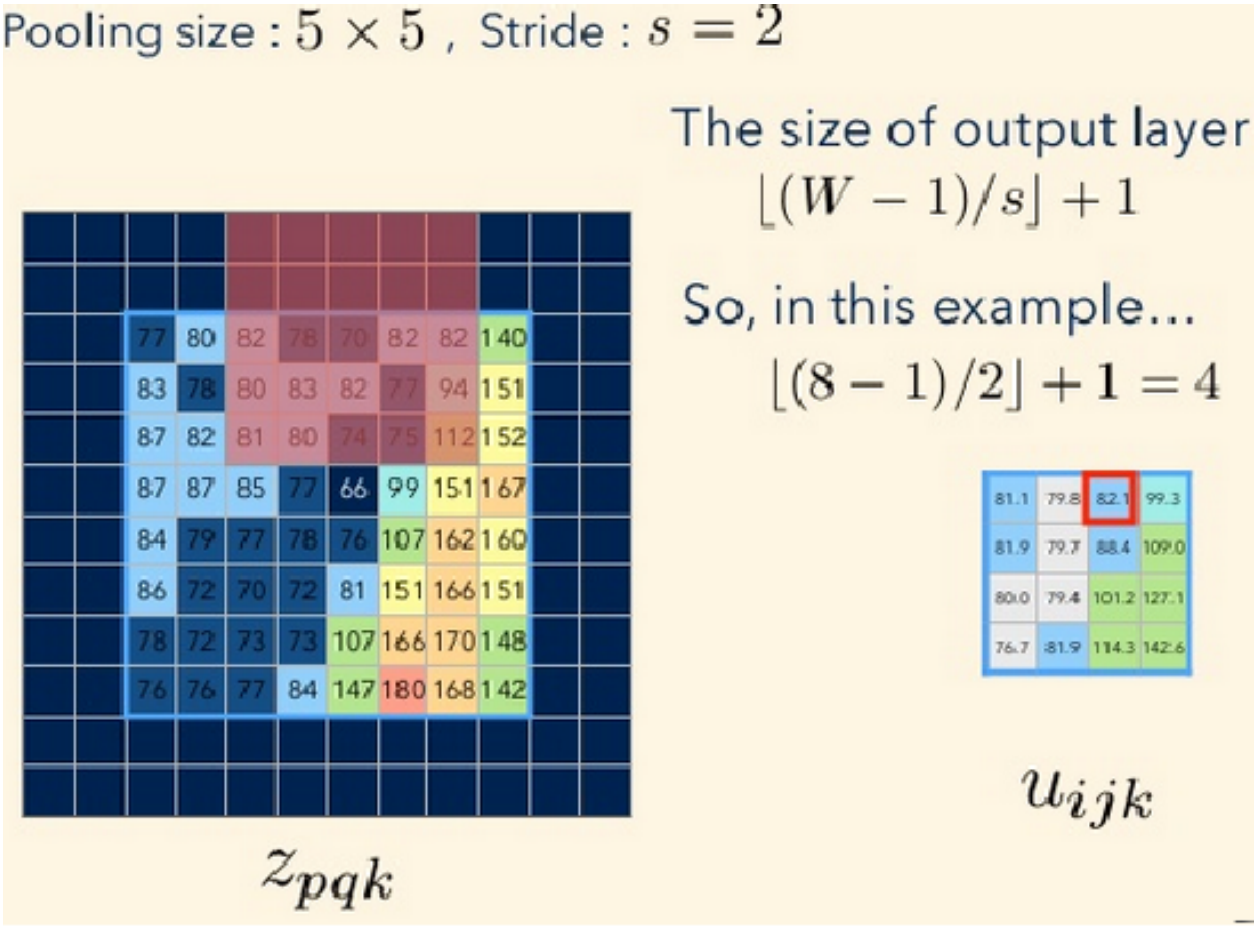
# Deep Learning Modules

## Overlapping Pooling



# Deep Learning Modules

## Overlapping Pooling

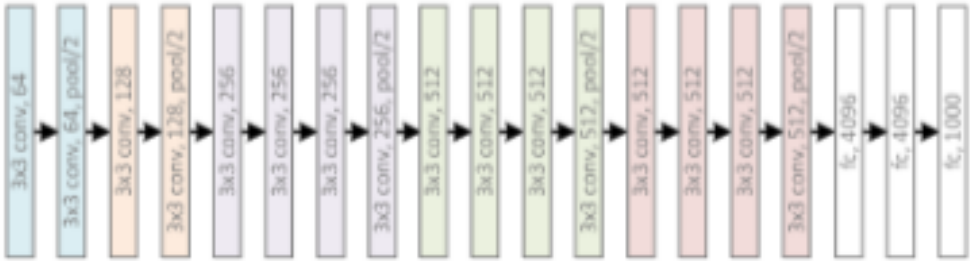


# Outline

# Deep Learning since 2012

## More & more data (Facebook $10^9$ images / day), larger & larger networks

VGG, 16/19 layers, 2014



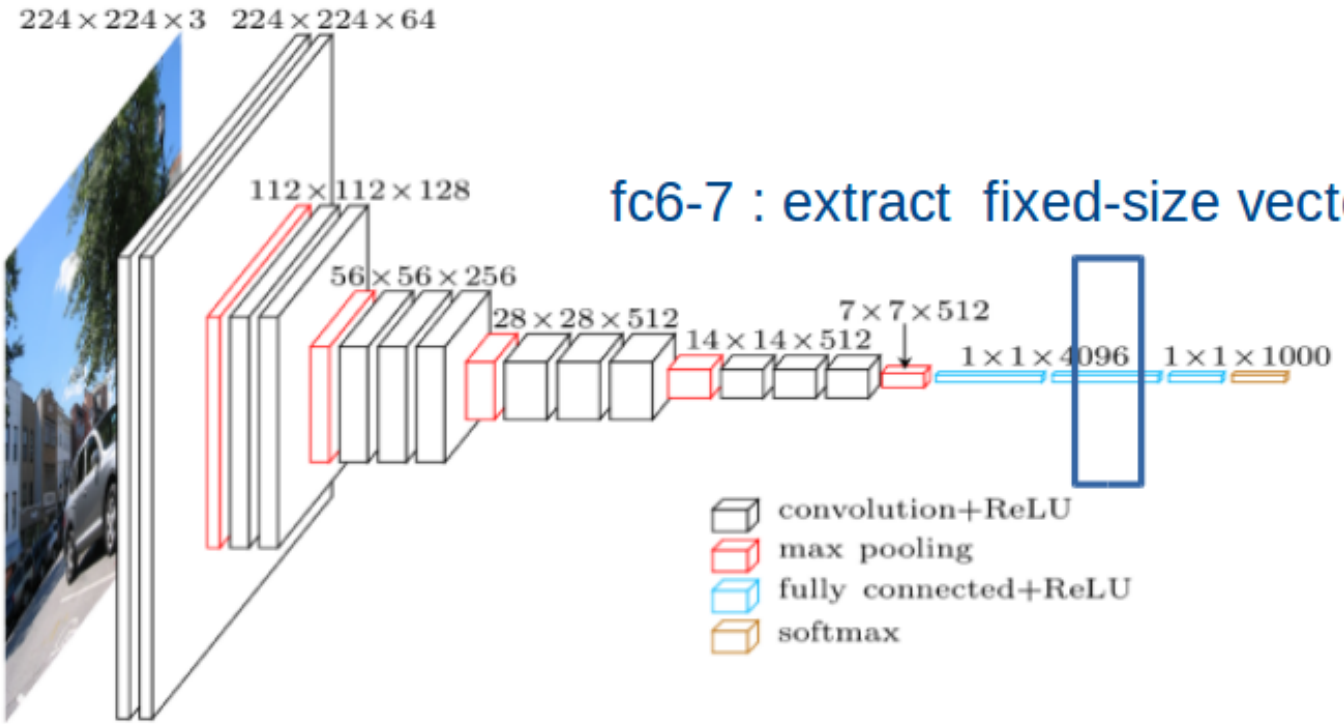

GoogleNet, 22 layers, 2014

ResNet, 152 layers, 2015

# Deep Learning since 2012

## Transferring Representations learned from ImageNet

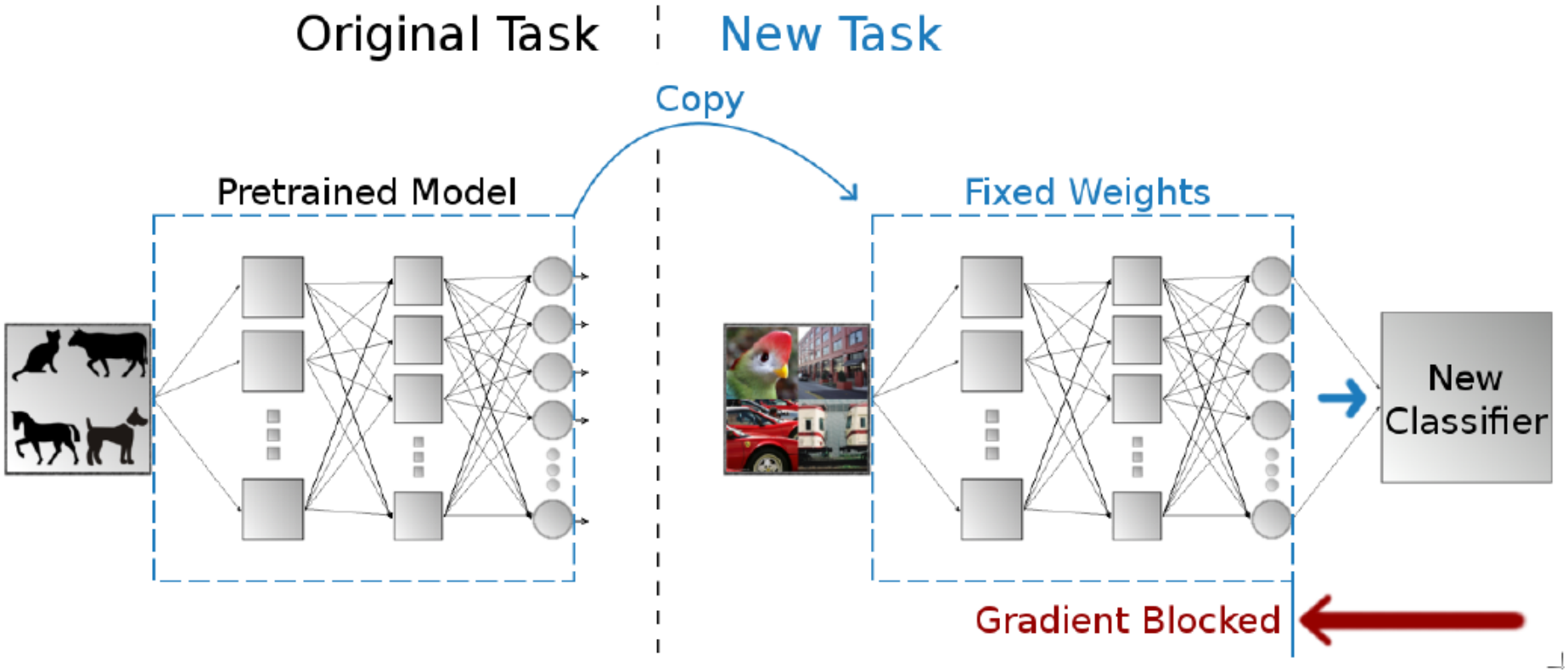
- Deep ConvNets require large-scale annotated datasets
  - Huge # params ⇒ difficult to train from scratch on "small datasets"
- **BUT**: Extract layer ⇒ fixed-size vector: **"Deep Features" (DF)**




- **Now state-of-the-art for any visual recognition task**

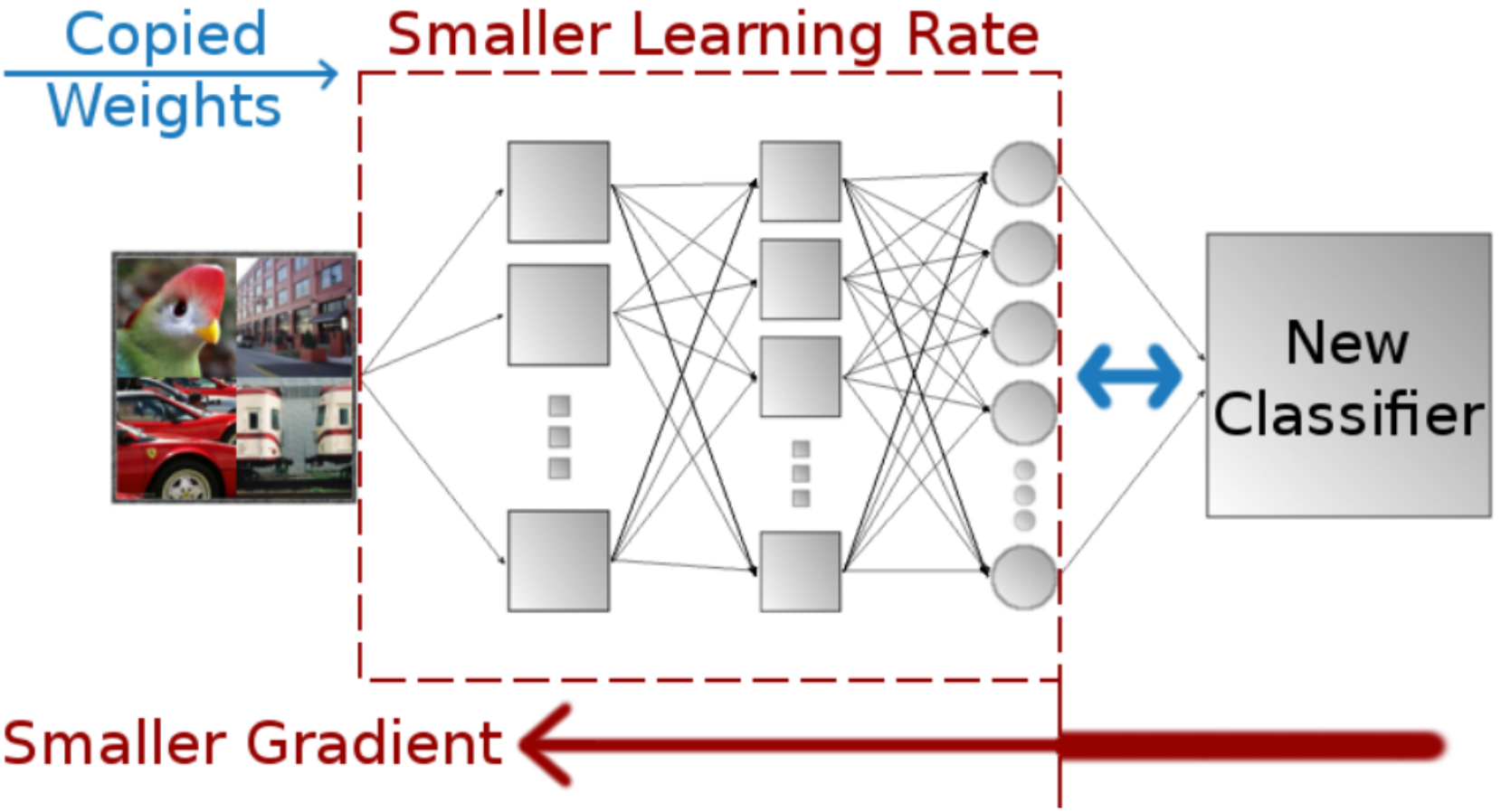
# Deep Features (DF) and Domain Adaptation

## DF: off-the-shelf descriptors (pure transfer)



# Deep Features (DF) and Domain Adaptation

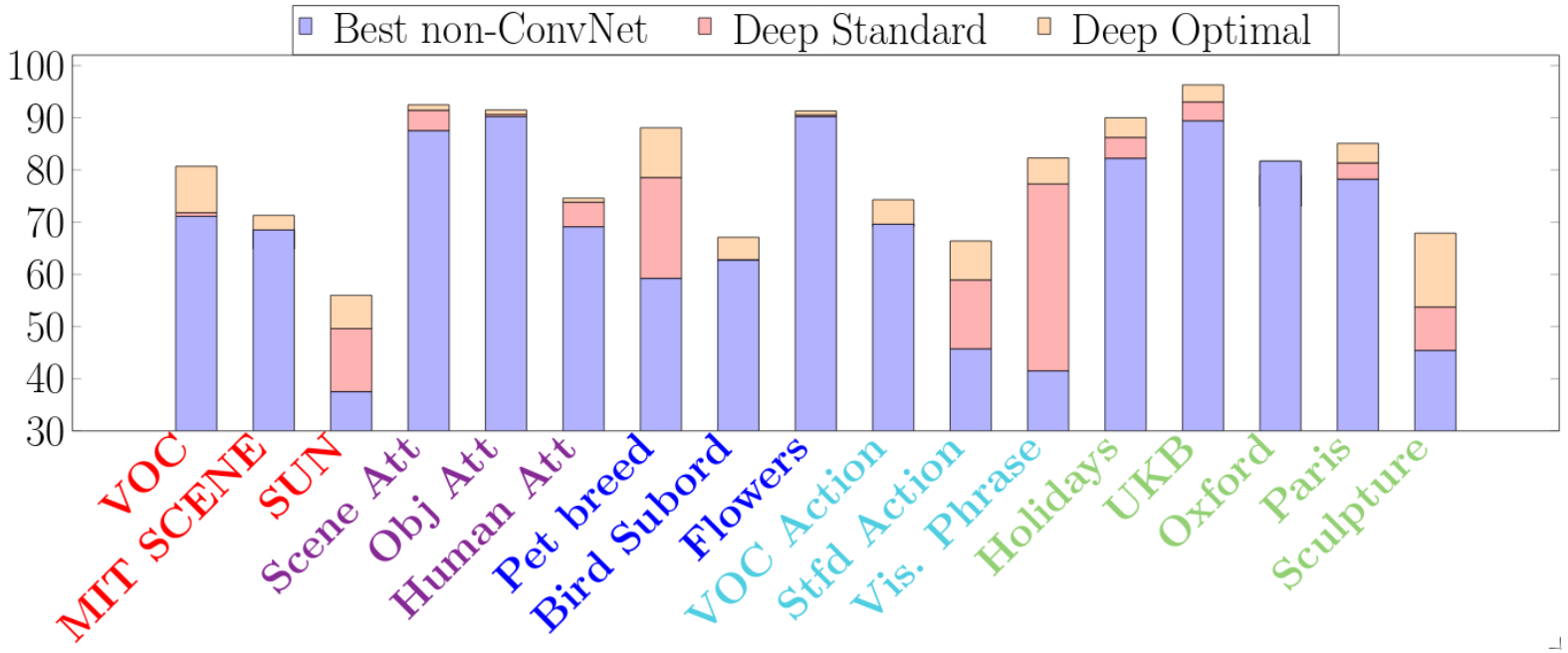
## DF: fine-tuning



# Deep Features (DF) and Domain Adaptation

**Increasing distance from ImageNet** →

| Image Classification | Attribute Detection | Fine-grained Recognition | Compositional | Instance Retrieval |
|---|---|---|---|---|
| PASCAL VOC Object [9] | H3D human attributes [6] | Cat&Dog breeds [29] | VOC Human Action [9] | Holiday scenes [17] |
| MIT 67 Indoor Scenes [33] | Object attributes [10] | Bird subordinate [43] | Stanford 40 Actions [46] | Paris buildings [31] |
| SUN 397 Scene [45] | SUN scene attributes [30] | 102 Flowers [27] | Visual Phrases [34] | Sculptures [4] |

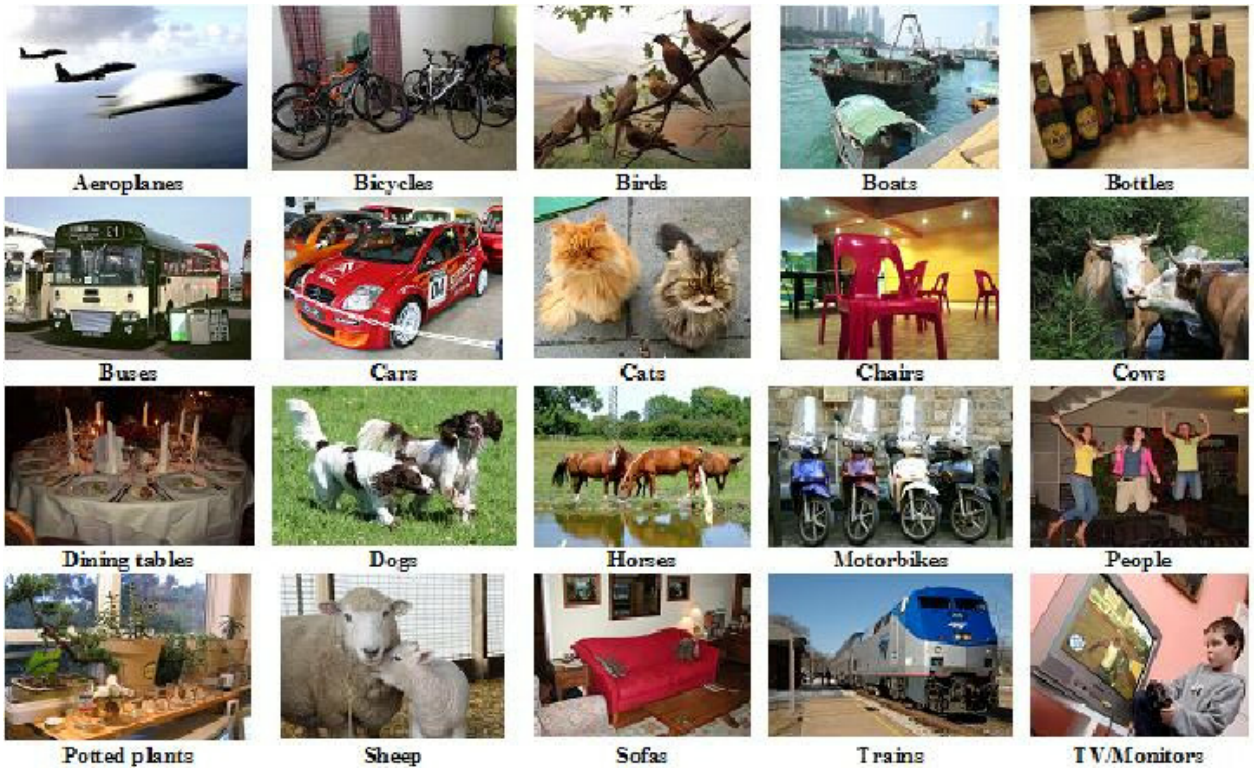



Credit: Razavian *et. al.* [ARS+16]

# Deep Features (DF) and Domain Adaptation

## DF for Image classification on other datasets

- **Small size datasets**: $\sim 10^3 - 10^4$ ex, *e.g.* VOC'07 (20 classes, 5000 ex)




| Model type | Test mAP |
|---|---|
| From Scratch | 39.79 |
| BoW [ATC+13] | 61.6 |
| Transfer | 83.22 |
| Fine Tuning | **85.70** |

Table: Mean Average Precision.

- Fine Tuning > Transfer >> Handcrafted (BoW)
- From scratch does not work (not enough training data)

# Deep Features (DF) and Domain Adaptation

## DF for Image classification on other datasets

- **Medium size datasets**, *e.g.* Food (LIP6) : 101 classes, $10^5$ ex

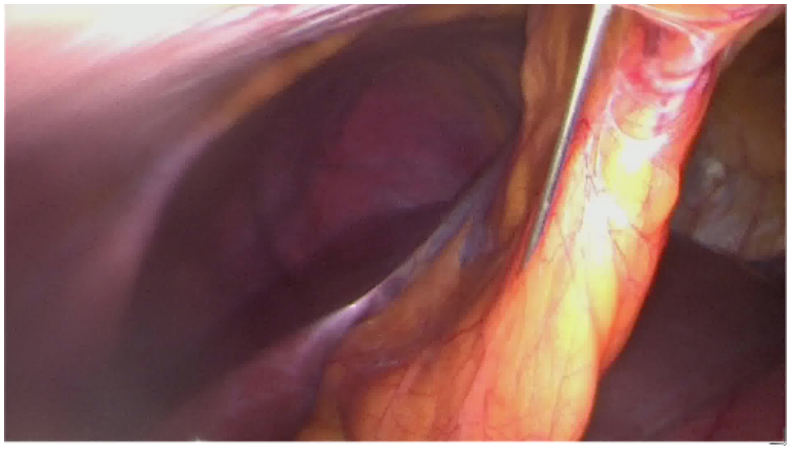
| Modèle | Test top 1 (%) |
|---|---|
| (a) Bag of visual Words | 23.96 |
| Overfeat & Extraction | 33.91 |
| Overfeat & From Scratch | 47.46 |
| Overfeat & Fine Tuning | 57.98 |
| (b) Vgg16 & Extraction | 40.21 |
| Vgg16 & From Scratch | 53.62 |
| Vgg16 & Fine Tuning | **65.71** |

- From scratch DOES work (well !)
- Fine Tuning >> From scratch >> Transfer >> Handcrafted (BoW)

# Deep Features (DF) and Domain Adaptation

## DF for Image classification on other datasets

- Another ex: M2CAI'16 challenge - large domain shift (medical images)
  - Medium-size: 22 videos, $\sim 60 \cdot 10^4$ images, 8 classes

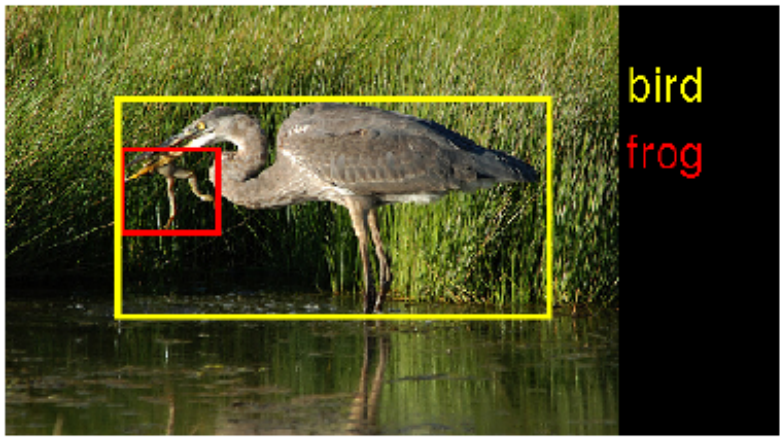


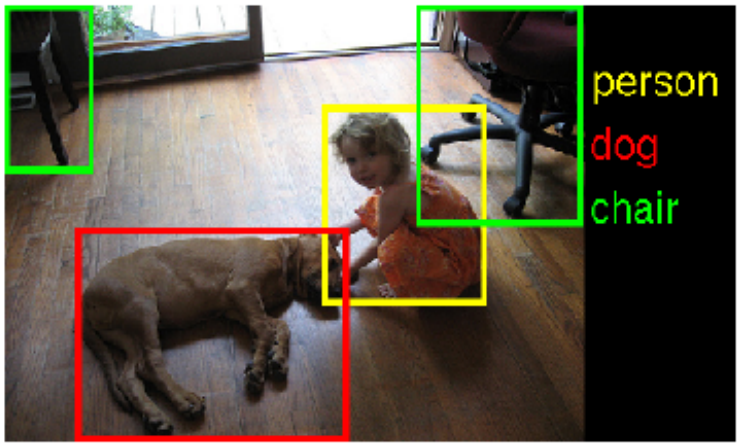
| Model | Acc Top1(%) |
|---|---|
| Transfer | 59.27 |
| From Scratch | 69.13 |
| Fine Tuning | 79.06 |

- Fine Tuning >> From scratch >> Transfer
- Transfer already good baseline despite big visual content shift

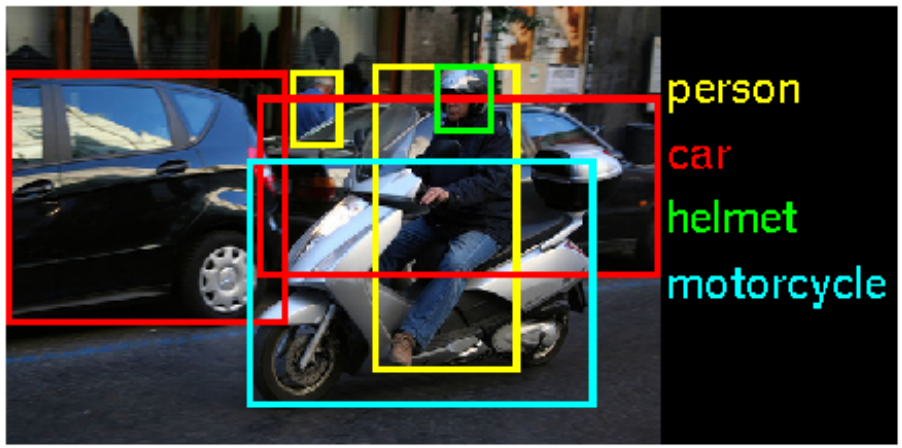
# Deep Features (DF) and Domain Adaptation

## Task Adaptation: Localization



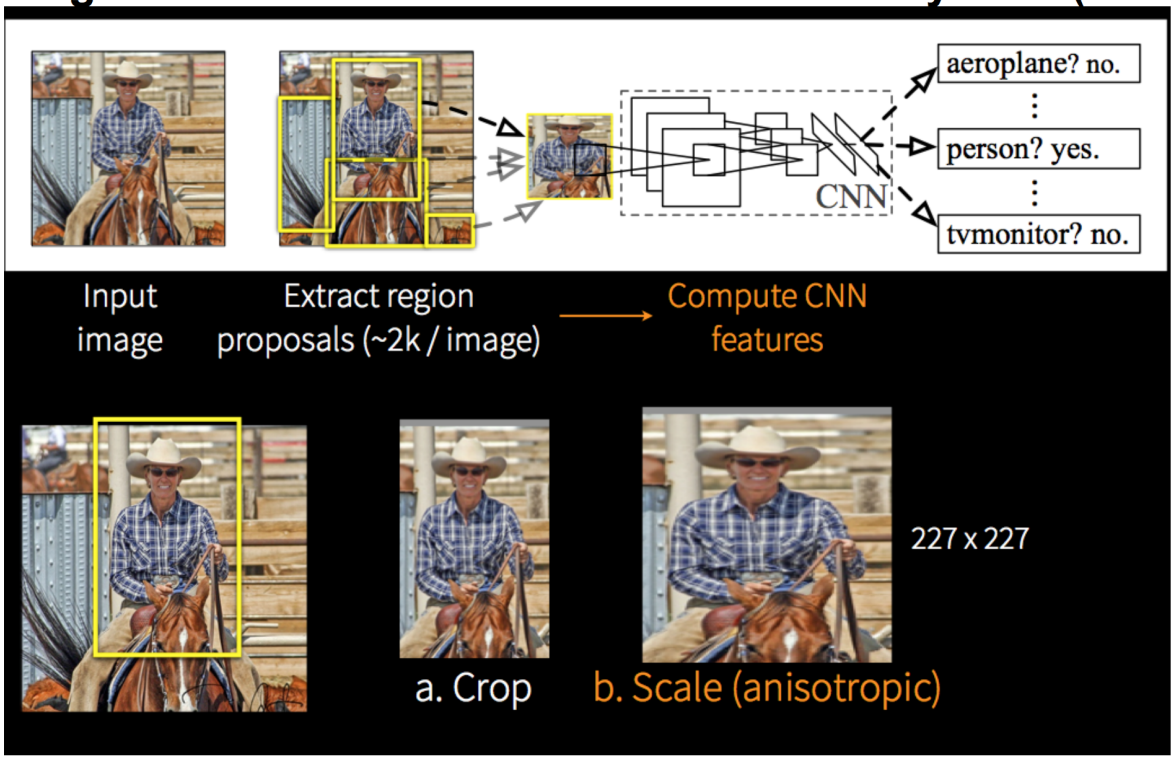




# Deep Features (DF) and Domain Adaptation

## Task Adaptation: Localization

**Regions with Convolutional Neural Net.s system (RCNN)**




R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR* 14

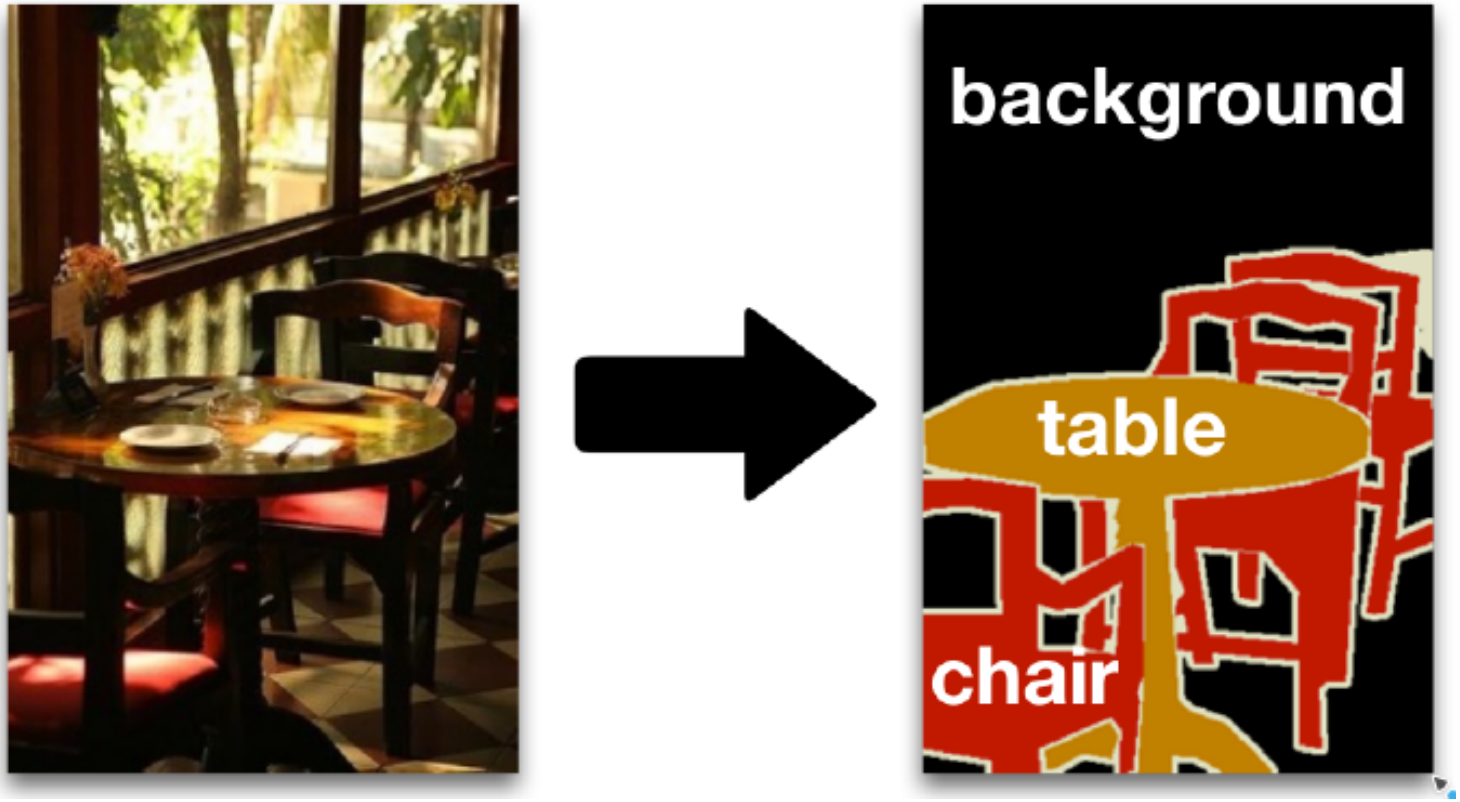
Eval on VOC'07:

| | |
|---|---|
| R-CNN | **58.5** |
| DPM HoG | 34.3 |

- R-CNN ⇒ region proposals ⇒ Deep Features ⇒ classify
- Significantly outperformed previous models (DPM on HoG features)

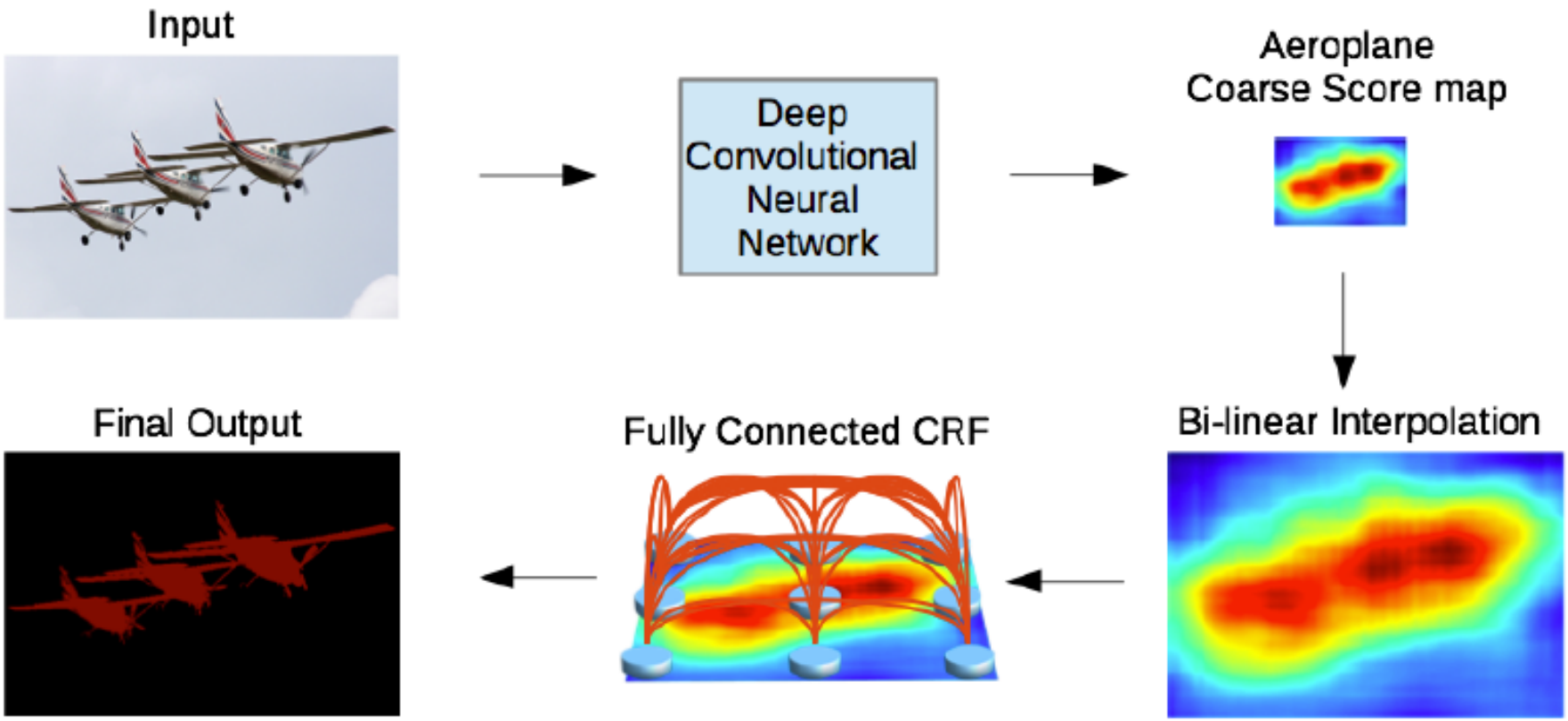
# Deep Features (DF) and Domain Adaptation

## Task Adaptation: Semantic Segmentation



# Deep Features (DF) and Domain Adaptation

## Task Adaptation: Semantic Segmentation

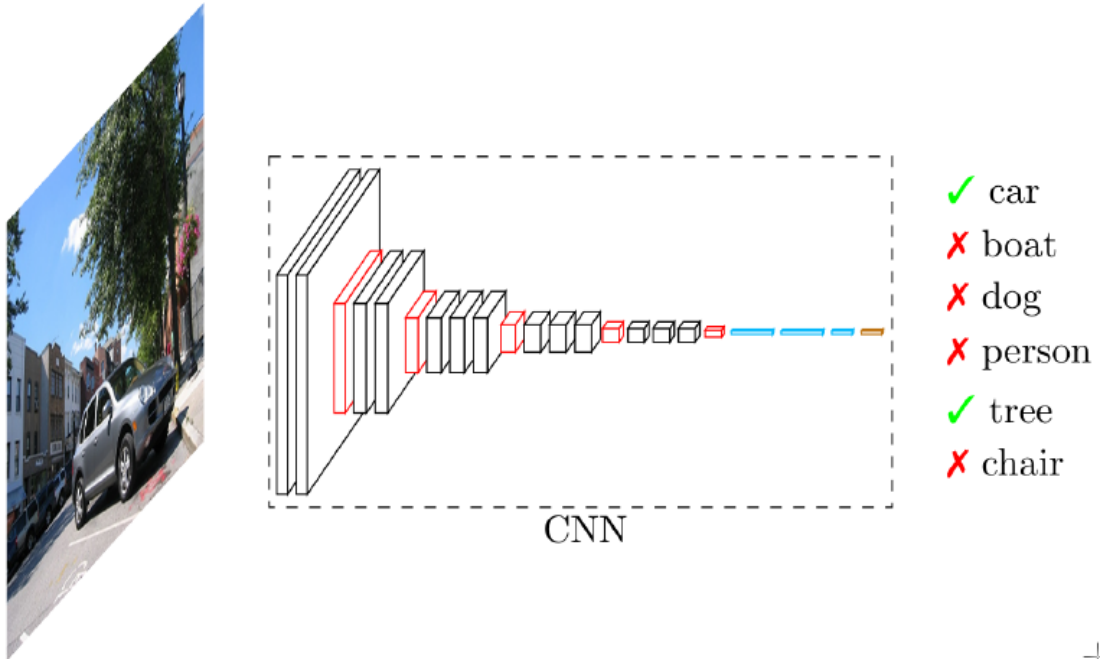



Chen *et.al.* ICLR'15

# Deep Features (DF) and Domain Adaptation

## Conclusion

- Deep Feature in transfer mode
  - Very good baseline
  - Ofter > descriptors based on expert knowledge
  - The solution for small databases
- From medium-size: from scratch possible and very competitive
- Fine-tuning always improves performances (small → large datasets)

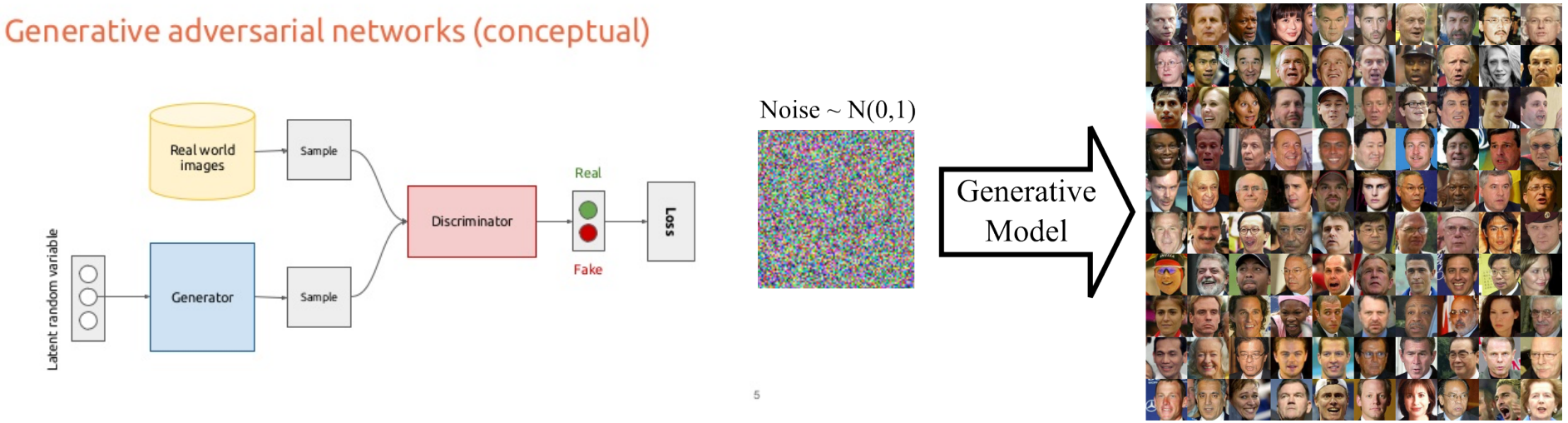


# Outline

# Ongoing Issues in Deep Learning

## Unsupervised Training

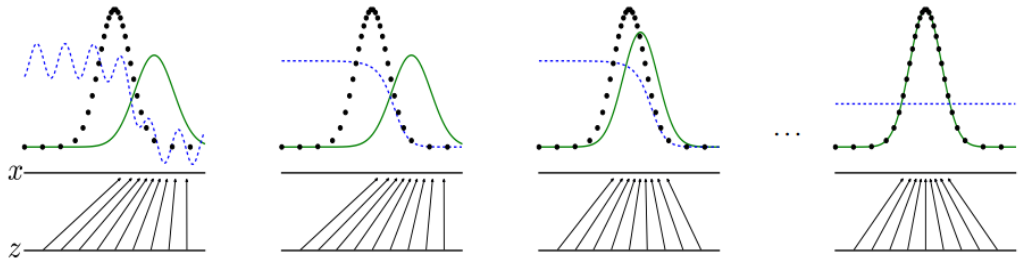
- Standard ways to perform unsupervised: learning representations fitting data well, *e.g.* Maximum likelihood, reconstruction error, *etc*
- Success of deep learning essentially for supervised problem
- Solution: cast unsupervised problem as a supervised one
  $\Rightarrow$ **auto-supervision**
  - Trendy example: Generative Adversarial Networks (GAN) [GPAM+14]



# Ongoing Issues in Deep Learning

## Unsupervised Training: GAN

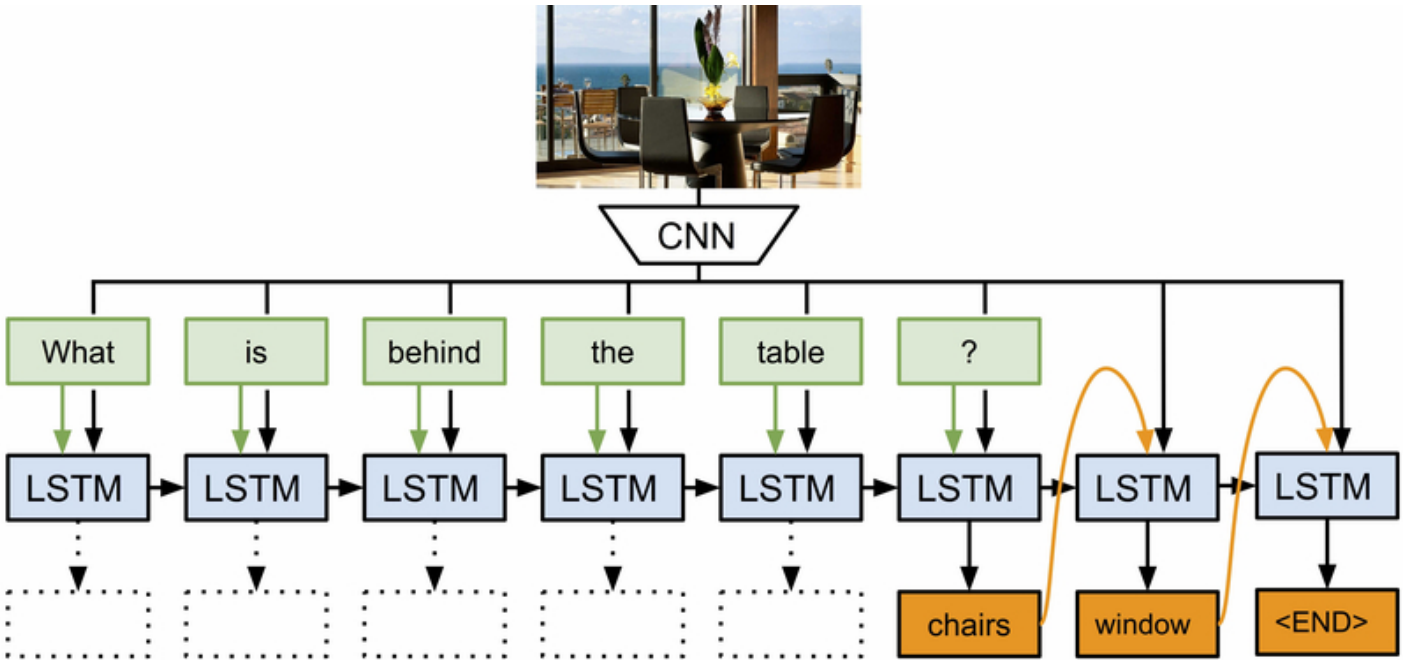
- Unsupervised problem ⇒ 2-player game theory problem
- Interesting results: optimal generator learns data distribution




# Ongoing Issues in Deep Learning

## New Tasks in Artificial Intelligence

- Vision and language, Visual Question Answering (VQA)

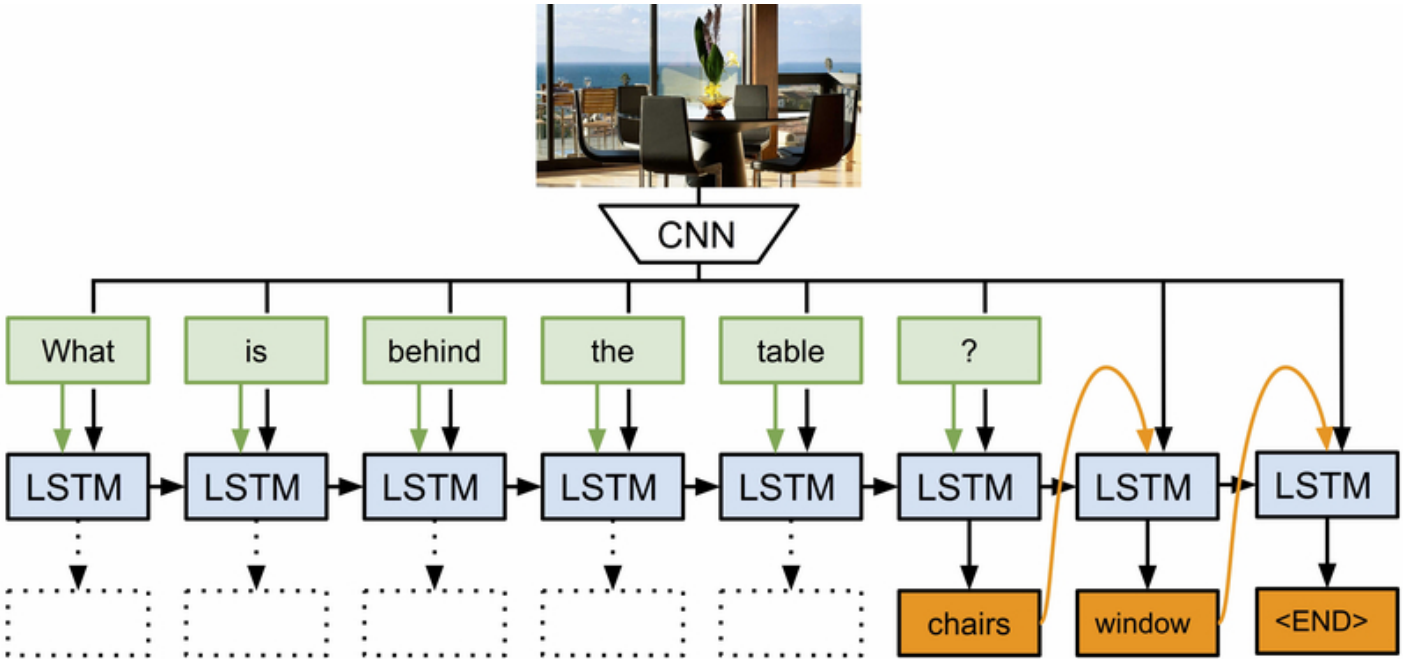



Credit: M. Malinowski [MRF15]

# Ongoing Issues in Deep Learning

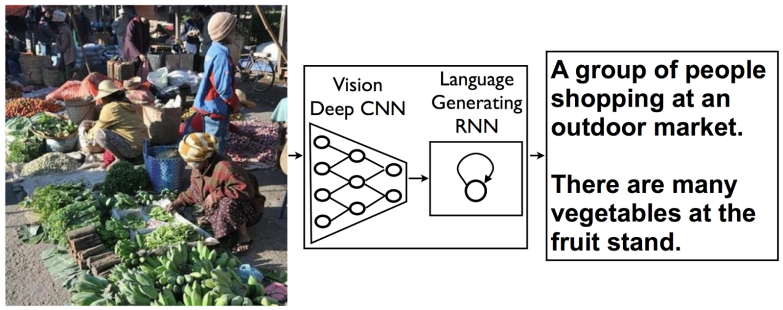
## New Tasks in Artificial Intelligence

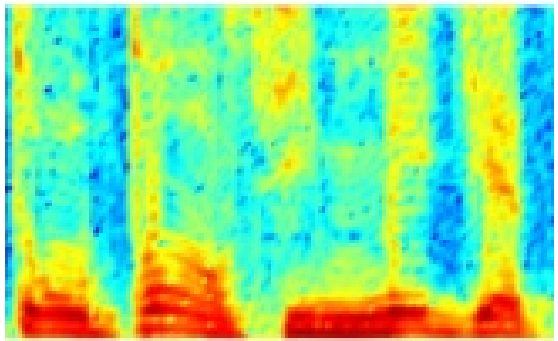
- But still a long way to go toward real AI ...

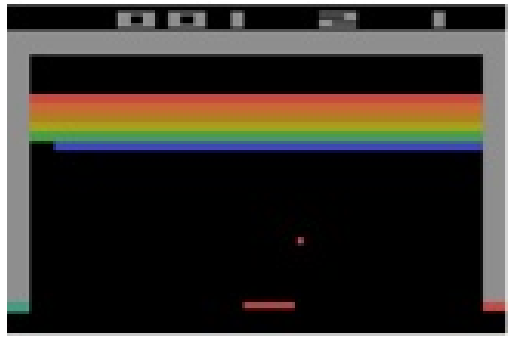



Credit: M. Malinowski [MRF15]

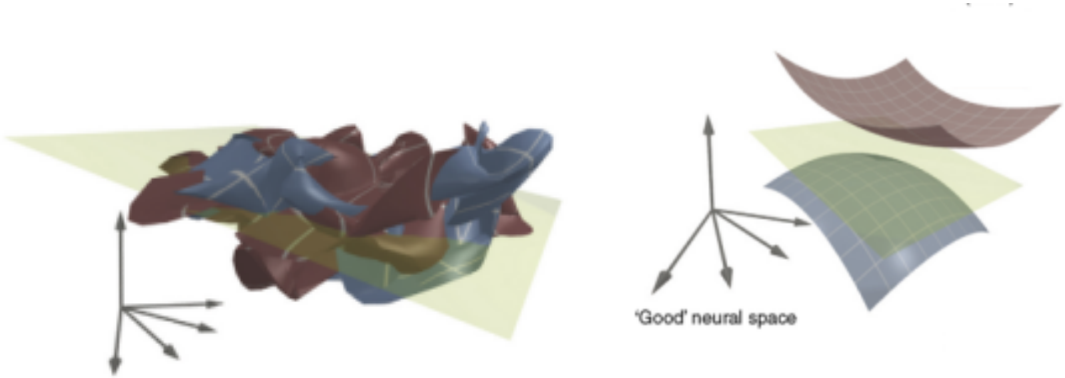
# Conclusion





- Deep Learning: huge impact in terms of experimental results
- BUT: formal understanding still limited,
  - Optimization: non-convex problem
  - Model: ability to untangle manifold
  - Robustness to over-fitting & generalization



# References I



Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson, *Factors of transferability for a generic convnet representation*, IEEE Trans. Pattern Anal. Mach. Intell. **38** (2016), no. 9, 1790–1802.


Bearman, Russakovsky, Ferrari, and Fei-Fei, *What's the Point: Semantic Segmentation with Point Supervision*, ECCV (2016).


Thibaut Durand, Nicolas Thome, and Matthieu Cord, *WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks*, Computer Vision and Pattern Recognition (CVPR), 2016.


Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in Neural Information Processing Systems 27 (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), Curran Associates, Inc., 2014, pp. 2672–2680.


Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang, *Recent advances in convolutional neural networks*, CoRR **abs/1512.07108** (2015).


Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, *A fast learning algorithm for deep belief nets*, Neural Comput. **18** (2006), no. 7, 1527–1554.


Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, CoRR **abs/1207.0580** (2012).

# References II


Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 2012, pp. 1097–1105.


Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz, *Ask your neurons: A neural-based approach to answering questions about images*, 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, 2015, pp. 1–9.