

# Deep Learning

Volker Tresp  
Summer 2018

## Neural Network Winter and Revival

- While Machine Learning was flourishing, there was a Neural Network winter (late 1990's until late 2000's)
- Around 2010 there was a revival which made neural networks again extremely popular; it was restarted by Geoffrey Hinton, Yann LeCun, and Yoshua Bengio
- Yann LeCun (New York University and Facebook Artificial Intelligence Research) and Yoshua Bengio (Université de Montréal) are two world-class researchers who never stopped working with neural networks. Geoffrey Hinton (co-inventor of the MLP, Boltzmann machines and others) (Google and University of Toronto ) says it all got re-started with the 2006 paper “A fast learning algorithm for deep belief nets” by Hinton, Osindero, and Teh
- In Europe: Jürgen Schmidhuber at IDSIA
- Deep networks achieved best results on many tasks/datasets

## Schmidhuber: “Deep Learning Conspiracy”



Geoffrey Hinton  
(Toronto, Google)



Yann LeCun  
(New York, Facebook)



Yoshua Bengio  
(Montreal)

## Jürgen Schmidhuber



## Kai Yu



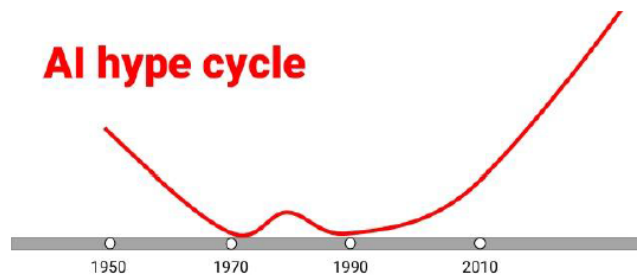
Yu Kai, head of Baidu's Institute of Deep Learning (IDL), demonstrates the smart bike project, DuBike, at the company's headquarters in Beijing. Photo: Simon Song

## What Belongs to Deep Learning

1. In general: Multi Layer Perceptrons (Neural Networks) with many large hidden layers
2. Any Recurrent Neural Network (RNN) network, in particular LSTMs and GRUs (separate slides)
3. Convolutional Neural Networks (CNNs)
4. Deep Reinforcement Learning (separate slides)
5. Representation Learning, including representations for words, entities, predicates, samples
6. Deep Generative Models: VAEs, GANs (separate slides)

# Artificial Intelligence

- *Creating machines that perform functions that require intelligence when performed by people (Kurzweil, 1990)*



## Games



## Q&A



## Auton. Driving



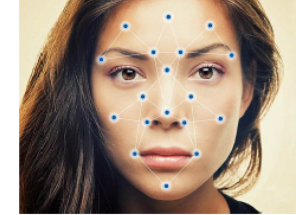
## Drones, Robots



## Translation



## Face Recognition



## Speech Recognition



?

## Deep Learning

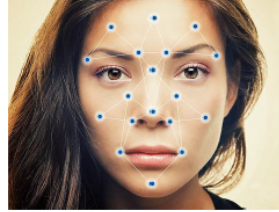
- Deep Learning is the reason for the emerging huge interest in AI

- Convolutional DL

- Recurrent DL

- Reinforcement DL

Face Recognition



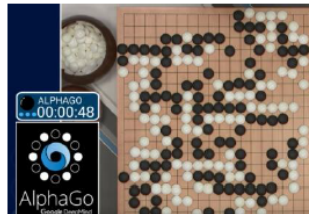
Translation



Speech Recognition




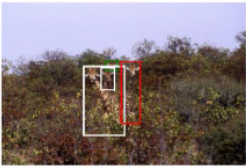
Games





## Student Magic: Visual Q&A

*"I spy with my little eye ..."*

Image	Policy Gradient	Tempered Policy Gradient		
	Is it in left?	No	Is it a person?	No
	Is it in front?	No	Is it a <b>vehicle</b> ?	Yes
	Is it in right?	Yes	Is it a <b>truck</b> ?	Yes
	Is it in middle?	Yes	Is it in front of photo?	No
	Is it person?	No	In the left half?	No
	Is it ball?	No	In the middle of photo?	Yes
	Is it bat?	No	Is it to the right photo?	Yes
	Is it <b>car</b> ?	Yes	Is it in the middle of photo?	Yes
	Status:	Failure	Status:	Success
		Is it in <b>left</b> ?	No	Is it a giraffe?
Is it in front?		Yes	In front of photo?	Yes
Is it in right?		No	In the <b>left half</b> ?	Yes
Is it in middle?		Yes	Is it in the middle of photo?	Yes
Is it person?		No	Is it <b>to the left</b> of photo?	Yes
Is it giraffe?		Yes	Is it to the right photo?	No
Is in middle?		Yes	<b>In the left</b> in photo?	No
Is in middle?		Yes	In the middle of photo?	Yes
Status:		Failure	Status:	Success

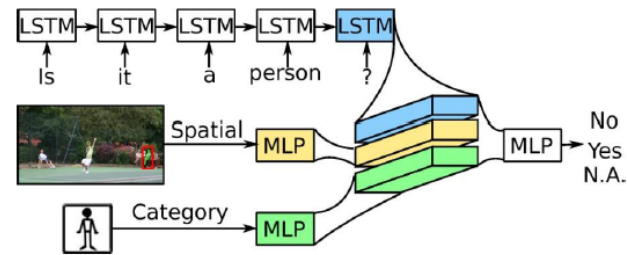
Convolutional DL

+ Recurrent DL

+ Reinforcement DL

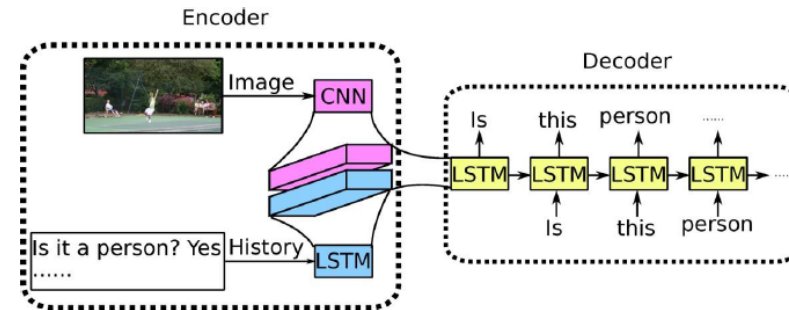
*Talents, Talents Talents!*

### The Oracle Model

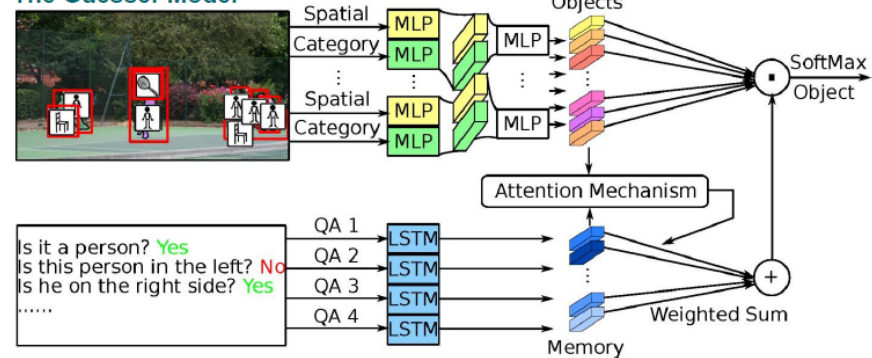


Rui Zhao, 2018

### The Question-Generator Model



### The Guesser Model



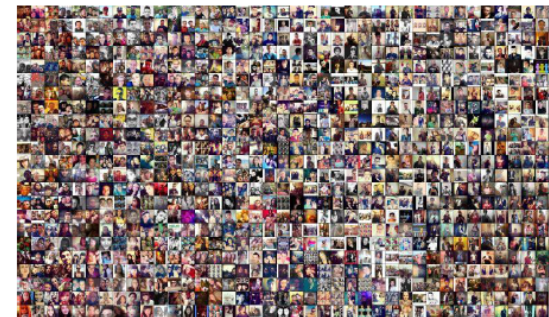
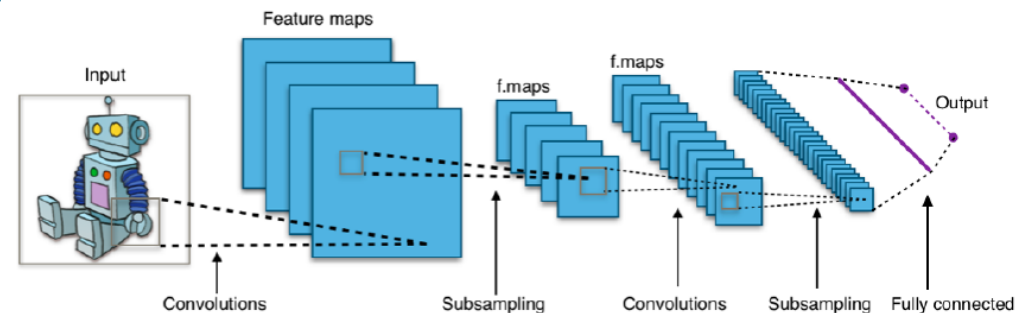
## Deep X Technologies behind Artificial Intelligence

- **Deep Learning;** Machine Learning;  
Data Mining; Statistics

- More (Labeled) Data
- Deeper Models
- New Algorithms
- End-to-End Training; Differentiable-Computing
- Computational Power
- Community

- **Deep Knowledge**

- Huge Document Repositories with Rapid IE / QA (IBM Watson)
- Maps with GPS for Autonomous Driving
- Ubiquitous Big Data in Industry
- Detailed (Patient) Profiles
- Web Content, Wikipedia for Humans
- **Knowledge Graphs for Machines**



## Deep Learning Recipe: What Are the Reasons? (Hinton 2013)

1. Take a large data set
2. Take a Neuronal Network with many (e.g., 7) large (z.B. 1000 nodes/layer) layers
3. Optional: Use GPUs
4. Train with Stochastic Gradient Decent (SGD)
5. Except for the output layer use *rectified linear units*:  $\max(0, h)$
6. Regularize with *drop-out*
7. Optional: Initialize weights with unsupervised learning
8. If the input is spatial (e.g., a picture), use convolutional networks (*weight sharing*) with *max-pooling*

## Important Benefits

- A deep network learns complex application-specific features trained on **many data points (large  $N$ )**
- Data are given in some feature space (can be raw pixel images); **no additional feature engineering or basis function design is necessary**. Work with the data as they are, also with **large  $M$**
- A deep architecture can achieve an efficient representation with fewer resources in a hierarchical layered structure
- Compositionality: In a classifier, an image is analysed by composing low level representations formed in the first processing layers, to generate high level features in the higher layers; a deep generative models composes an image from hierarchical representations

# 1: Large Data Set

- When decision boundaries are complex, a large data set describes the details of those boundaries
- Details can be captured with a complex (multi-layer) neural network
- “As of 2016, a rough rule of thumb is that a supervised deep learning algorithm will generally achieve acceptable performance with around 5,000 labeled examples per category, and will match or exceed human performance when trained with a dataset containing at least 10 million labeled examples.” (Deep Learning, Goodfellow et al.)

## 2: Large Networks

- It has been possible to train small to medium size problems since the early 1990s
- In deep learning, people work with really large Neural Networks. Example: 10 layers, 1000 neurons/layer
- ResNet from 2015 had 152 layers

### 3: Graphical Processing Units (GPUs)

- GPUs are highly suited for the kind of number crunching, matrix/vector math involved in deep Neural Networks. GPUs have been shown to speed up training algorithms by orders of magnitude
- Their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel
- General-Purpose Computing on Graphics Processing Units (GPGPU) is the utilization of a graphics processing unit (GPU), which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the central processing unit (CPU)

## 4: Stochastic Gradient Descent SGD

- Often regular SGD is used where the gradient is calculated on a single training pattern
- “Minibatch SGD” works identically to SGD, except that more than one training example is used to make each estimate of the gradient
- Gradient clipping (to avoid huge update steps):  
if  $\|\mathbf{g}\| > v$  then  $\mathbf{g} \leftarrow \mathbf{g}v/\|\mathbf{g}\|$ .  $v > 0$  is the norm threshold
- Local optima do not appear to be a major problem: current thinking is that there are many local optima, but that they are all very good



## Adaptive Learning Rates

- AdaGrad (adaptive gradient algorithm) is often used for learning rates to be adaptively altered
- Let  $g_j$  be the gradient for weight  $w_j$  accumulated over a minibatch at “time”  $t$

$$w_j := w_j - \frac{\eta}{\sqrt{G_{j,j}}} g_j$$

Here

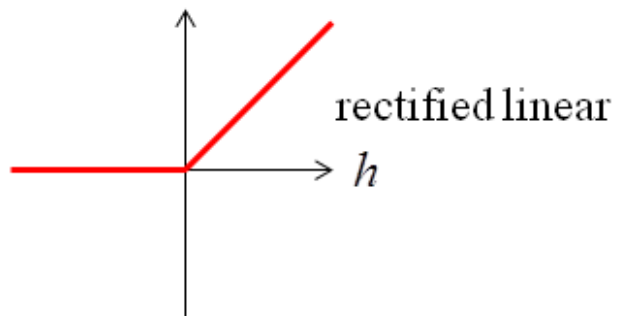
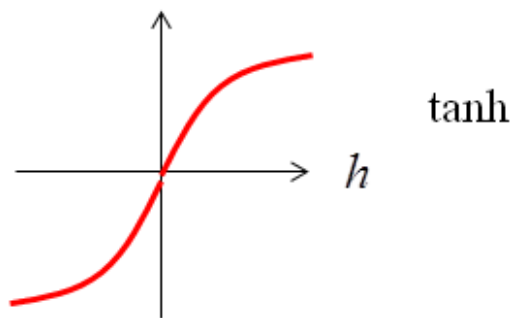
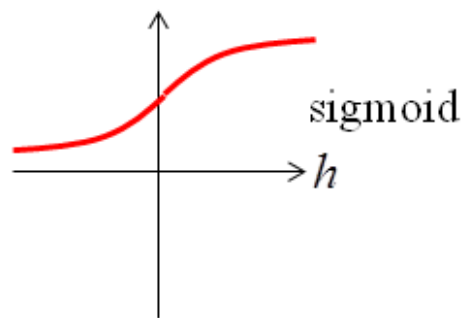
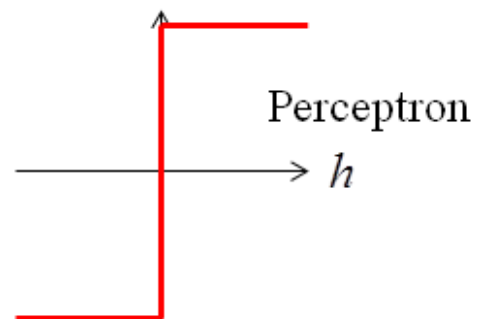
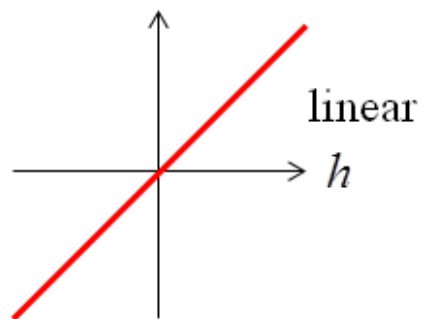
$$G_{j,j} = \sum_{\tau=1}^t g_{\tau,j}^2$$

is the accumulated squared gradient from the start of the epoch

- Extreme parameter updates get dampened, while parameters that get few or small updates receive higher learning rates
- The accumulation of the gradient from the start of the epoch can be problematic. Other popular variants: AdaDelta, Adam, RMSProp, Hessian-Free Optimization

## 5: Rectified Linear Function

- The transfer function of a Rectified Linear Unit (ReLU) is  $\max(0, h)$
- ReLU can be motivated in the following way: summing up the response of identical neurons (same input and output weights) where only the threshold/bias is varying. This becomes similar to a rectified linear neuron
- Reduces the effects of the vanishing gradient problem with sigmoid neurons! They learn much faster!
- Seems odd since some neurons become insensitive to the error, but a sufficient number stays active
- Leads to sparse gradients and to a sparse solution
- For training classification tasks, the output layer has sigmoidal activation functions and the cross-entropy cost function is used



**common neural  
transfer functions**

## 6A: Drop-Out Regularization

- For each training instance: first remove 50% of all hidden units, randomly chosen. Only calculate error and do adaptation on the remaining network
- For testing (prediction): use all hidden units but multiply all outgoing weights by  $1/2$  (gives you same expectation but no variance)
- This is like a committee machine, where each architecture is a committee member, but committee member share weights. It supposedly works like calculating the geometric mean: average the log of the predictions (and then take the exponential over the average)
- Works better than stopped training! No stopping rule required!
- Can even do drop-out in the input layer, thus different committee members see different inputs!
- Hinton: *use a large enough neural network so that it overfits on your data and then regularize using drop out*

- Goodfellow (DL): *Dropout provides an inexpensive approximation to training and evaluating a bagged ensemble of exponentially many neural networks*
- Variant: DropConnect (dropout of single connections)

## 6B: Weight Regularization

- Weight decay works
- But even better: for each neuron, normalize the incoming weight vector to have the same maximum length. Thus if  $\|\mathbf{w}\| > \alpha$

$$\mathbf{w} \rightarrow \alpha \frac{1}{\|\mathbf{w}\|} \mathbf{w}$$

- Backpropagation is performed through the normalization

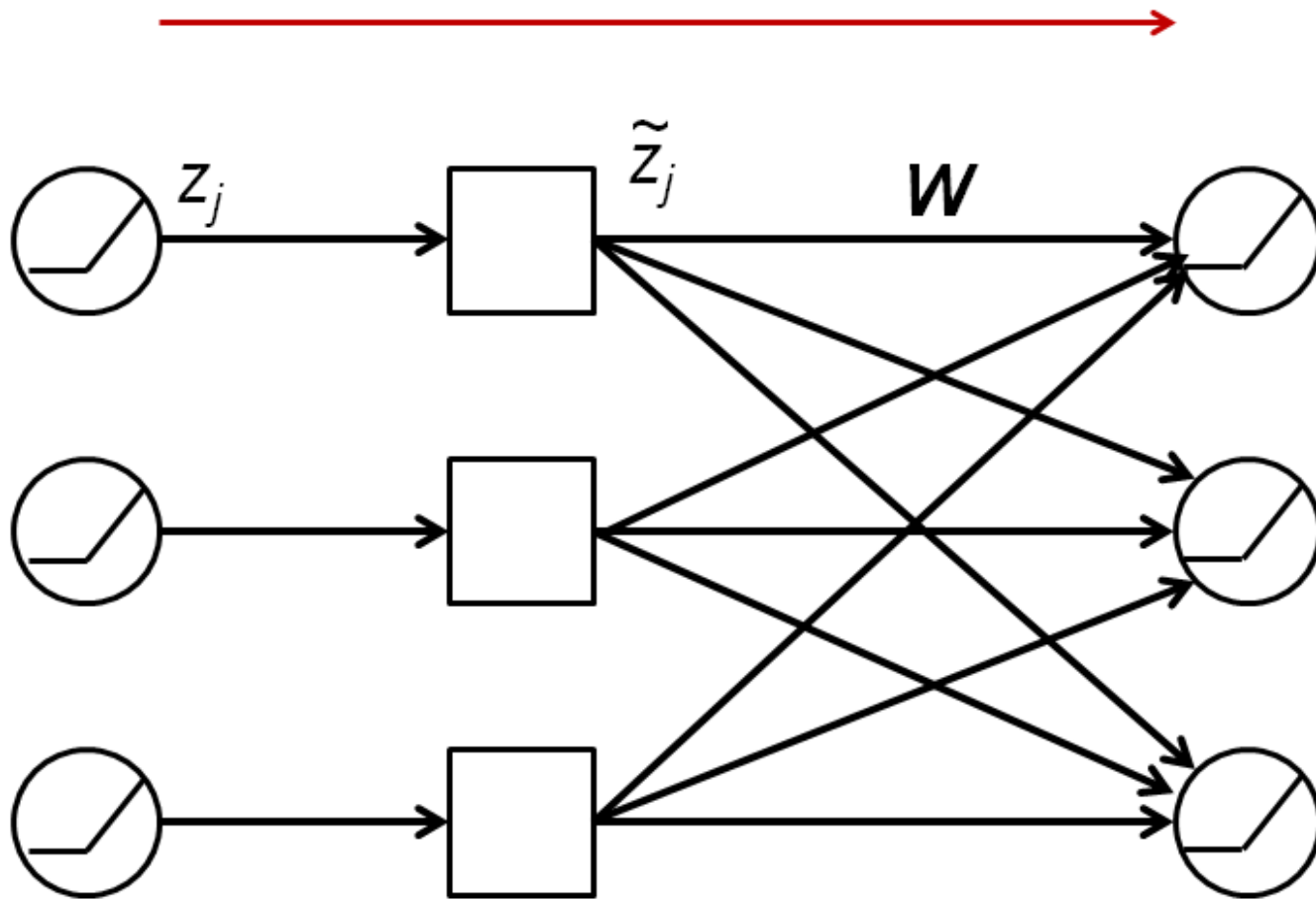
## 6C: Batch Normalization

- Batch normalization: Let  $\mathbf{z}$  be the vector of outputs of the previous layer. Then each dimension  $z_j$  is normalized as

$$\tilde{z}_j = \frac{z_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

- Mean  $\mu_j$  and variance  $\sigma^2$  are calculated over a small batch. Backpropagation is performed through these operations.  $\epsilon > 0$  is a small number, to ensure stability
- Batch normalization can greatly improve the training of deep networks!

## Batch Normalization



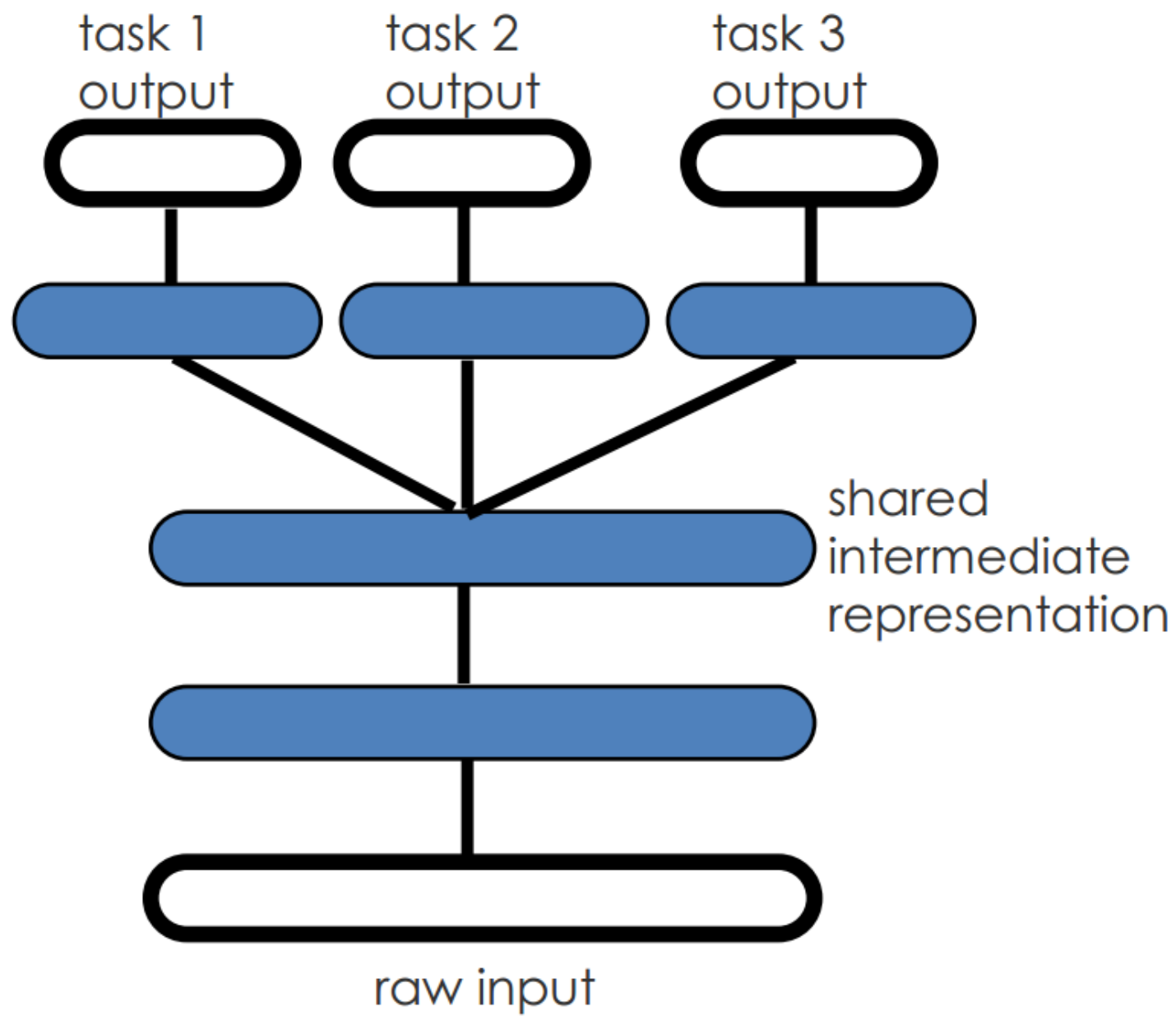


## 7: Initialize Weights with Unsupervised Learning

- The first layer is initialized with the encoder part of an autoencoder (separate slides)
- Alternatively: Several layers are initialized by a stacked autoencoder
- Not as popular anymore: Restricted Boltzmann Machine (RBM) for Deep Boltzmann Machines (DBMs) and Deep Belief Networks (DBNs)

## Multitask Learning

- In the autoencoder approach, the same representation is used to train both the auto-encoder and the classification task. This idea can be generalized in many ways. The idea is to learn several tasks by sharing common representations
- Another advantage is that a new task can be learned much faster!
- Current research: few-shot learning, one-shot learning, zero-shot learning



## Facebook's Deep Face: Face Recognition as Multi-Task Learning

- Build a deep learning NN to classify many face images from 4000 persons. Thus there are 4000 outputs, one for each person
- The next to last layer is used as a representation for any face image (also for faces and persons not in the training set)
- How do I use this net for new persons and faces?
- Calculate the distance between a new face and labeled faces from your Facebook friends based on the representation in the next to last layer
- Note that here, the representation is close to the output layer
- Much effort is spent in the input layers to normalize the facial images
- $C$ : convolutional layer.  $M$ : max-pooling layer. The subsequent layers (L4, L5 and L6) are locally connected, like a convolutional layer they apply a filter bank, but every location in the feature map learns a different set of filters.

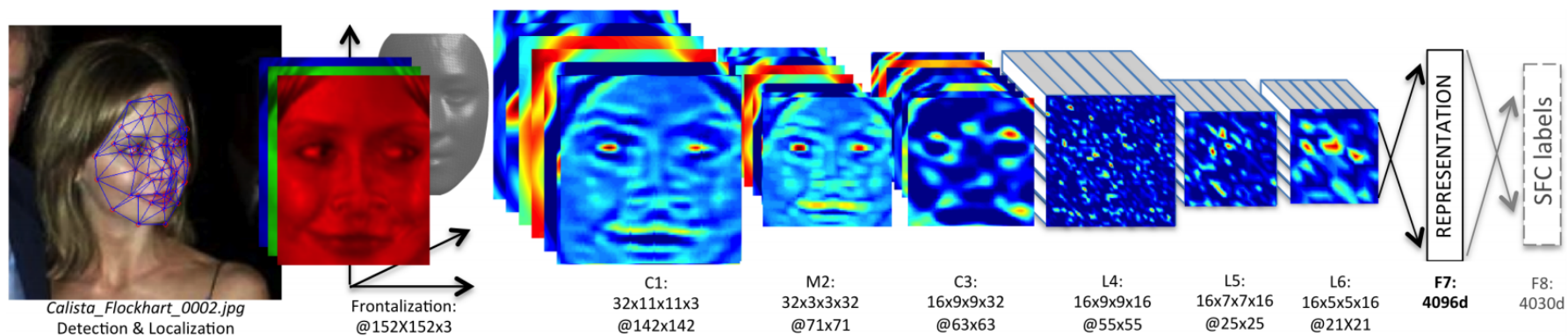
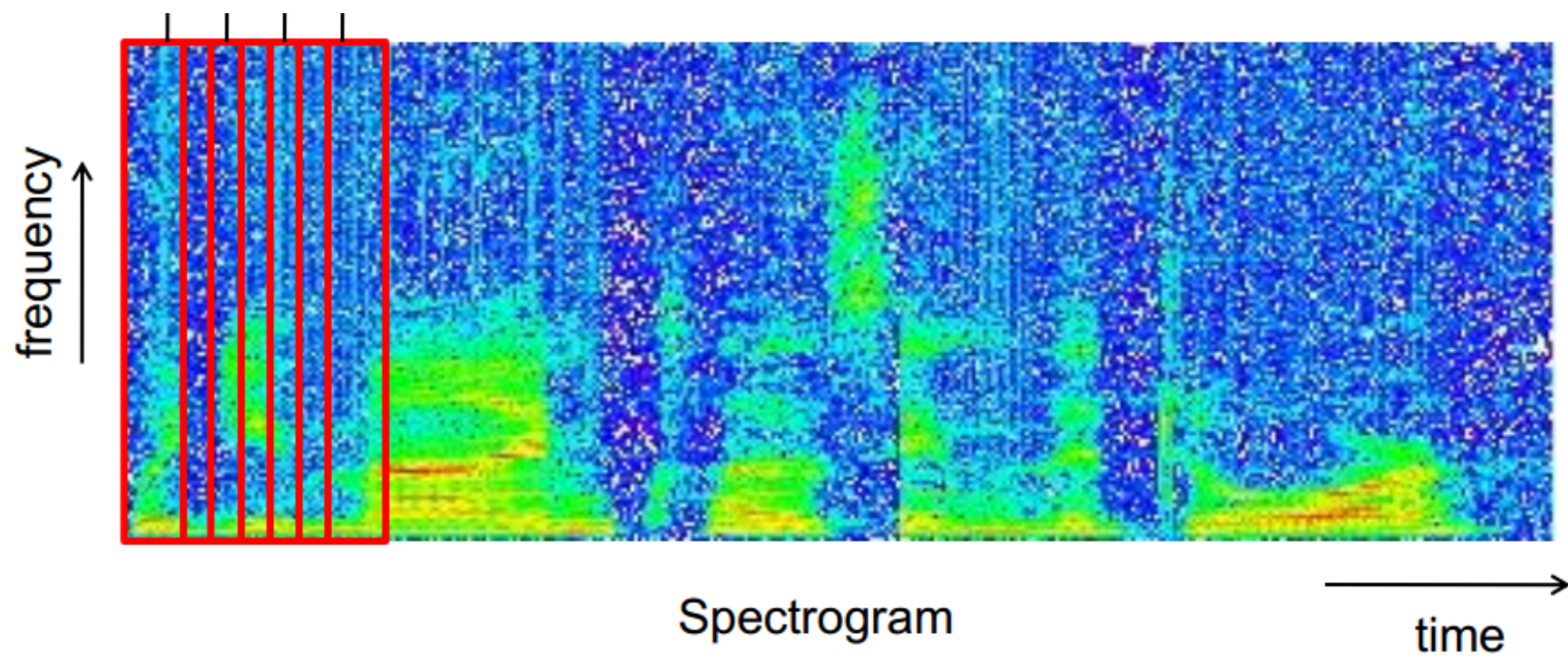


Figure 2. **Outline of the *DeepFace* architecture.** A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally-connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers.

## Android Server Architecture for Speech Recognition (2013)

- Part of speech recognition with Hidden Markov Models (HMMs): predict a state in the HMM (State) using a frequency representation of the acoustic signal in a time window (Frame)
- The Neural Network is trained to learn  $P(State|Frame)$
- 4-10 layers, 1000-3000 nodes / layer, no pre-training
- Rectified linear activations:  $y=\max(0,x)$
- Full connectivity between layers,
- Softmax output (cross-entropy cost function) (see lecture on linear classifiers)
- Features:
  - 25ms window of audio, extracted every 10ms.
  - log-energy of 40 Mel-scale filterbanks, stacked for 10-30 frames.

- Training time: 2-3 weeks using GPUs!
- Online: Android uses the server solution. Offline: Small Neural Network on the Smart Phone
- Advantage: Speaker independent! Now used by Google, Microsoft, IBM, replacing Gaussian mixture models (30% reduction in error)
- Even more improvement on the task of object recognition in images (from 26% error to 16% error)) using 1.2 million training images. With convolutional neural networks.

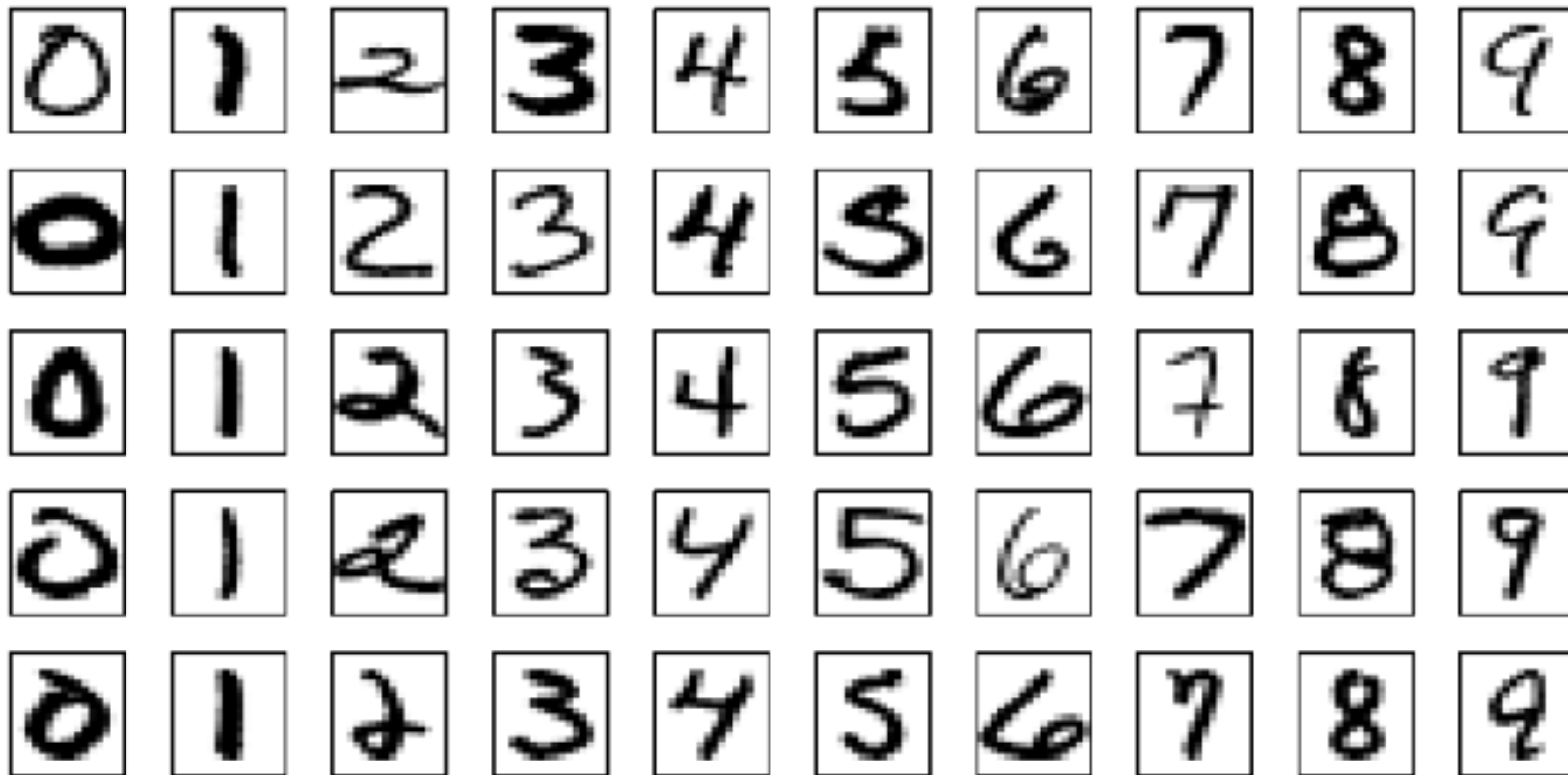




task	Hours of training data	Deep net+HMM	GMM+HMM same data	GMM+HMM more data
Switchboard	309	16.1	23.6	17.1 (2k hours)
English Broadcast news	50	17.5	18.8	
Bing voice search	24	30.4	36.2	
Google voice input	5870	12.3		16.0 (lots more)
Youtube	1400	47.6	52.3	

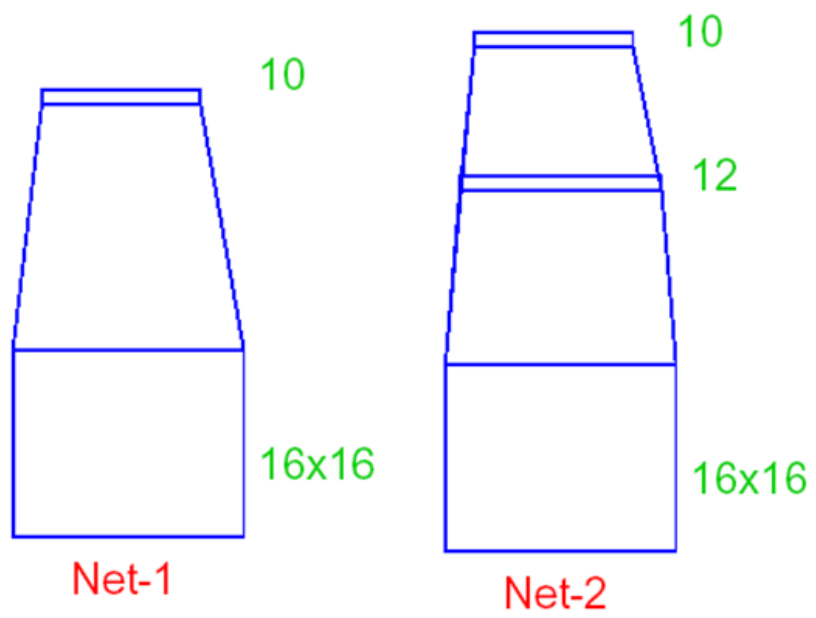
# 8: Convolutional Neural Networks (CNNs)

## Recognition of Handwritten Digits



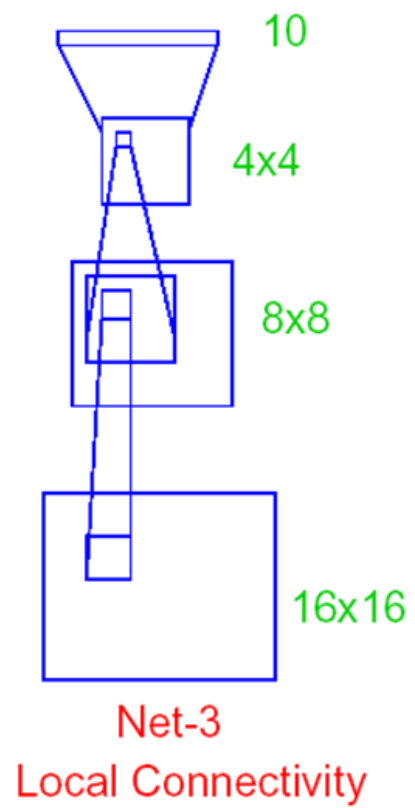
## Recognition of Handwritten Digits using Neuronal Networks

- Example:  $16 \times 16$  grey-valued pictures; 320 training images, 160 test images
- Net-1: No hidden layer: corresponds to 10 Perceptrons, one for each digit
- Net-2: One hidden layer with 12 nodes; fully connected (“normal MLP”)



## Neuronal Network with local connectivity: Net-3

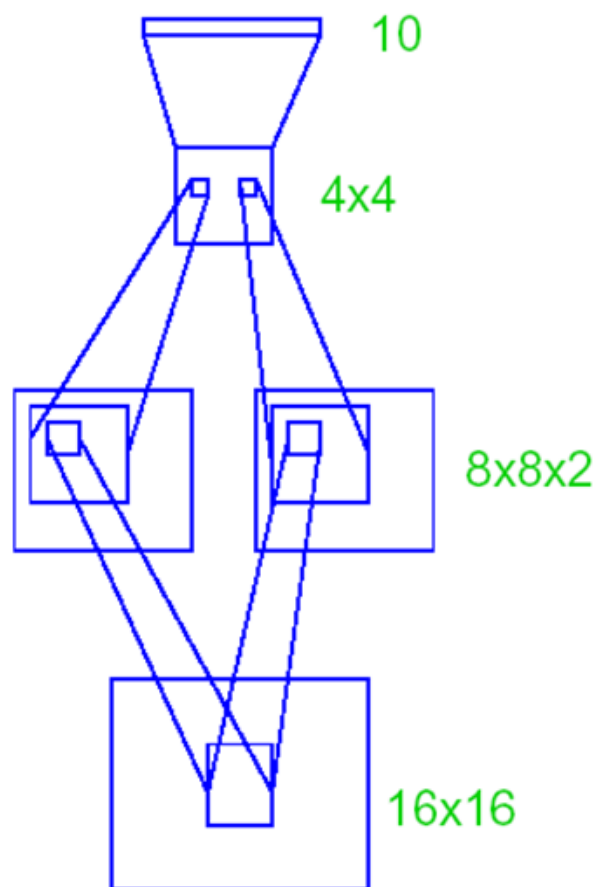
- In the following variants, the complexity was reduced
- Net-3: Two hidden layers with local connectivity (but no weight sharing yet): motivated by the local receptive fields in the brain
  - Each of the  $8 \times 8$  neurons in the first hidden layer is only connected to  $3 \times 3$  input neurons from a receptive field
  - In the second hidden layer, each of the  $4 \times 4$  neurons is connected to  $5 \times 5$  neurons in the first hidden layer
  - Net-3 has less than 50% of the weights of Net-2, but more neurons



## Neuronal Networks with Weight-Sharing (Net-4)

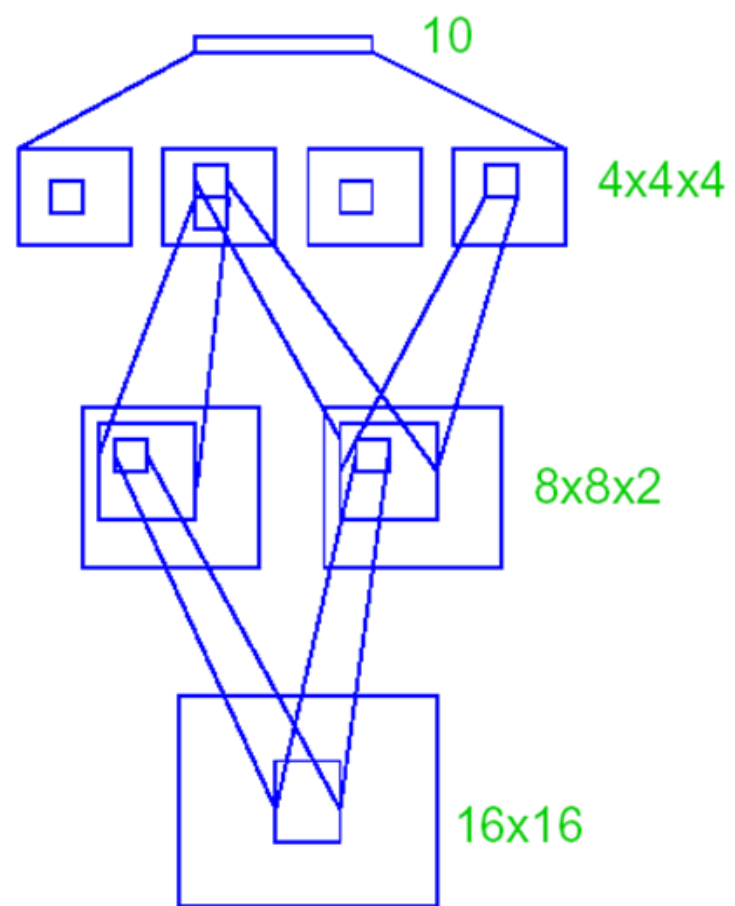
- Net-4: Two hidden layers with local connectivity and *weight-sharing*
- All receptive fields in the left  $8 \times 8$  block have the same weights; the same is true for all neurons in the right  $8 \times 8$  block
- The  $4 \times 4$  block in the second hidden layer, as before





## Neural Networks with Weight Sharing (Net-5)

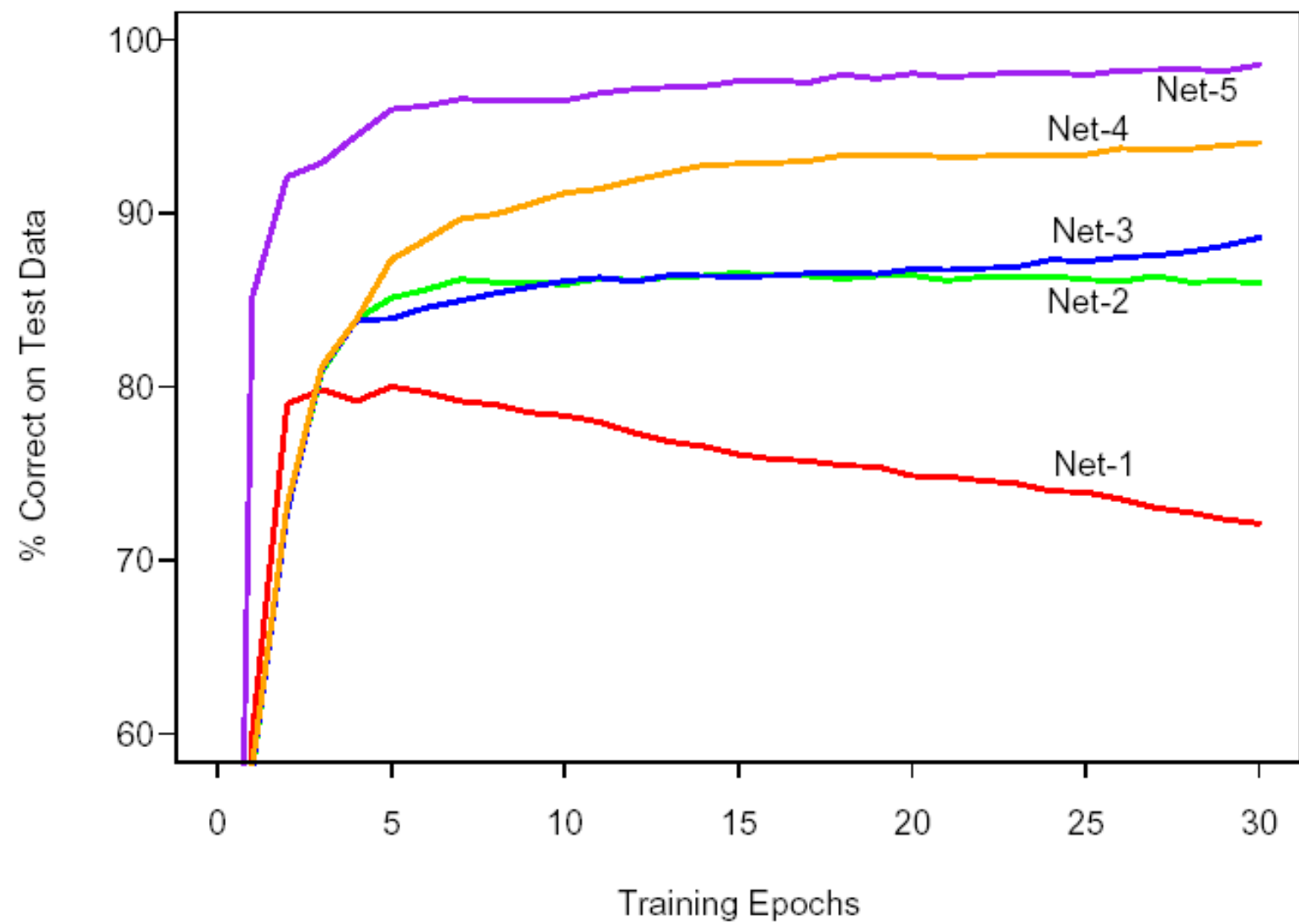
- Net-5: Two hidden layers with local connectivity and two layers of *weight-sharing*



Net-5

## Learning Curves

- One training epoch is one pass through all data
- The following figure shoes the performance on the test set
- Net-1: One sees overfitting with increasing epochs
- Net-5: Shows best results without overfitting



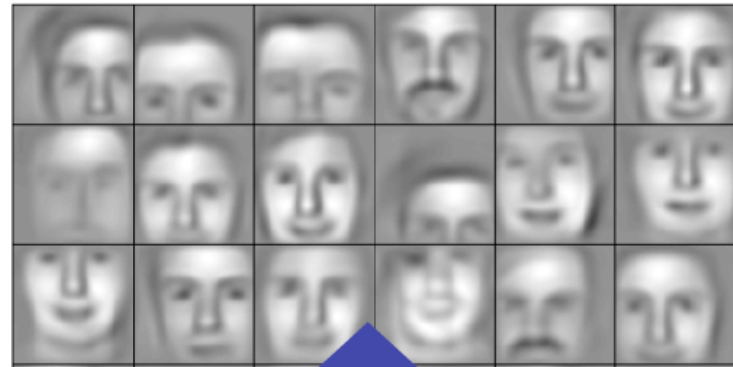
## Statistics

- Net-5 has best performance. The number of free parameters (1060) is much smaller than the total number of parameters (5194)

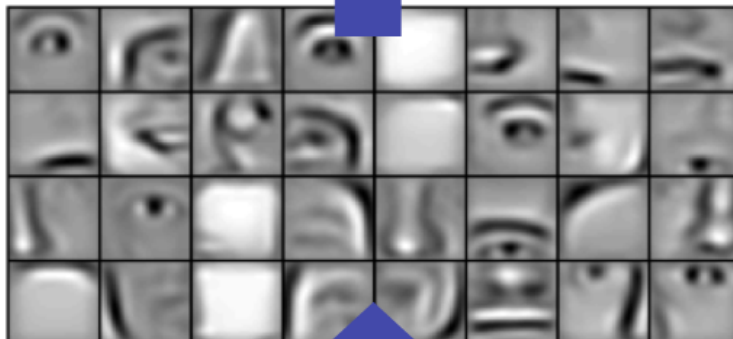
**TABLE 11.1.** *Test set performance of five different neural networks on a hand-written digit classification example (Le Cun, 1989).*

	Network Architecture	Links	Weights	% Correct
Net-1:	Single layer network	2570	2570	80.0%
Net-2:	Two layer network	3214	3214	87.0%
Net-3:	Locally connected	1226	1226	88.5%
Net-4:	Constrained network 1	2266	1132	94.0%
Net-5:	Constrained network 2	5194	1060	98.4%

Successive model layers learn deeper intermediate representations



Layer 3

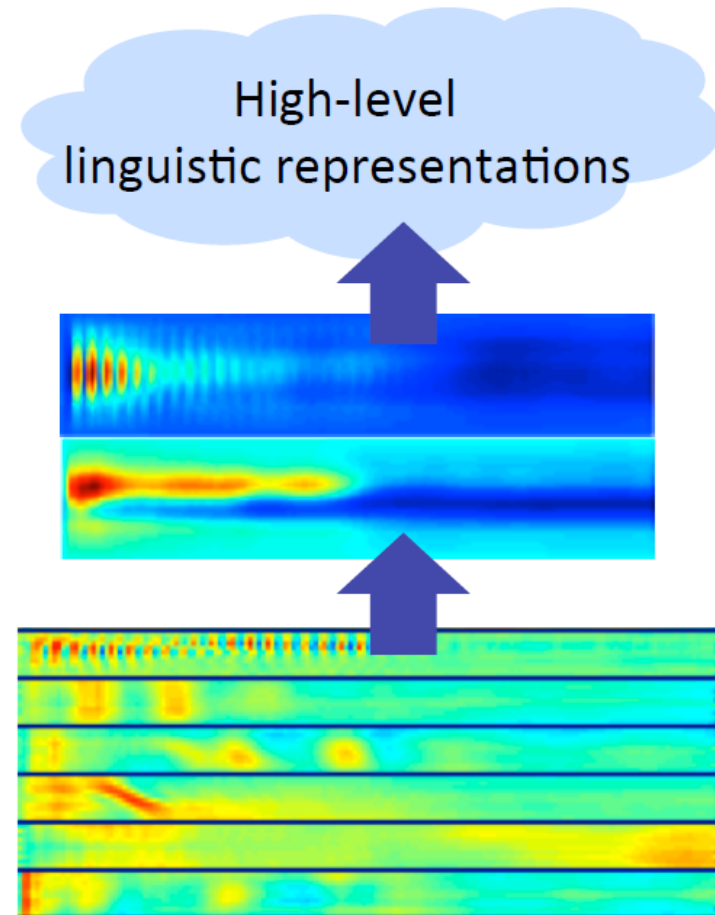


Parts combine  
to form objects

Layer 2



Layer 1



## Pooling

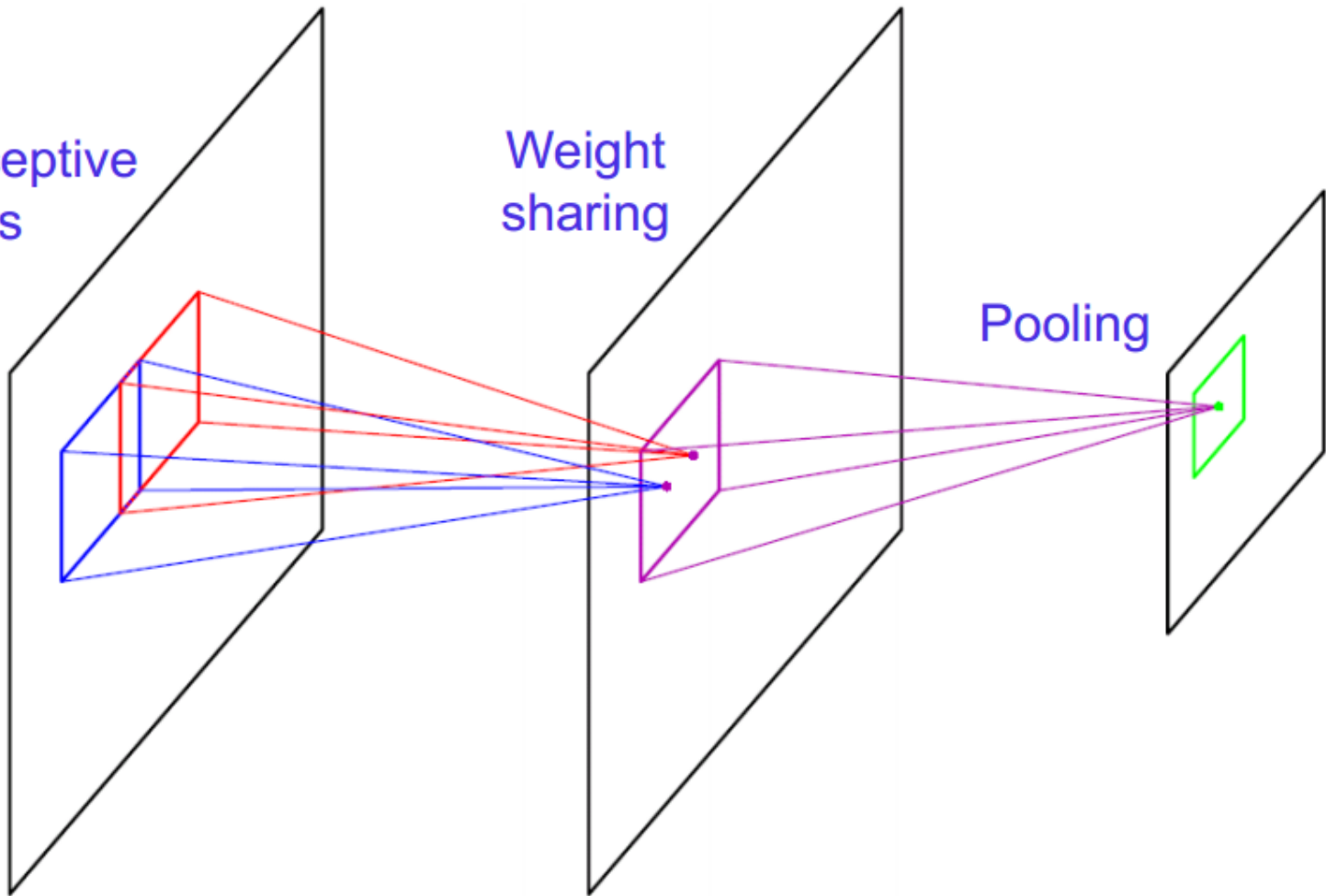
- For example, one could compute the mean (or max) value of a **particular feature** over a **region of the image**. These summary statistics are much lower in dimension (compared to using all of the extracted features) and can also improve results (less over-fitting). We aggregation operation is called this operation pooling, or sometimes **mean pooling** or **max pooling** (depending on the pooling operation applied).
- Max-pooling is useful in vision for two reasons: (1) it reduces the computational complexity for upper layers and (2) it provides a form of translation invariance
- Since it provides additional robustness to position, max-pooling is thus a “smart” way of reducing the dimensionality of intermediate representations.



Local Receptive  
Fields

Weight  
sharing

Pooling



Input image

Convolutional layer

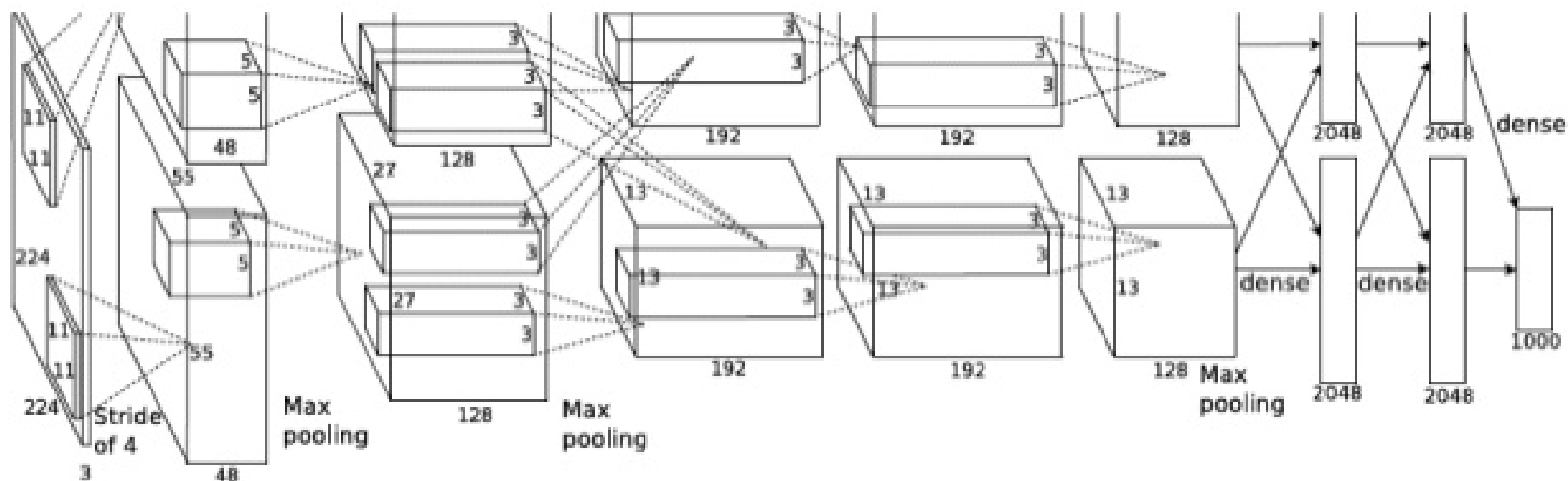
Sub-sampling  
layer

## Architectures

- The next slide shows AlexNet
- Alternatives: Inception (Google), Visual Geometry Group (VGG) (Oxford)

# AlexNet

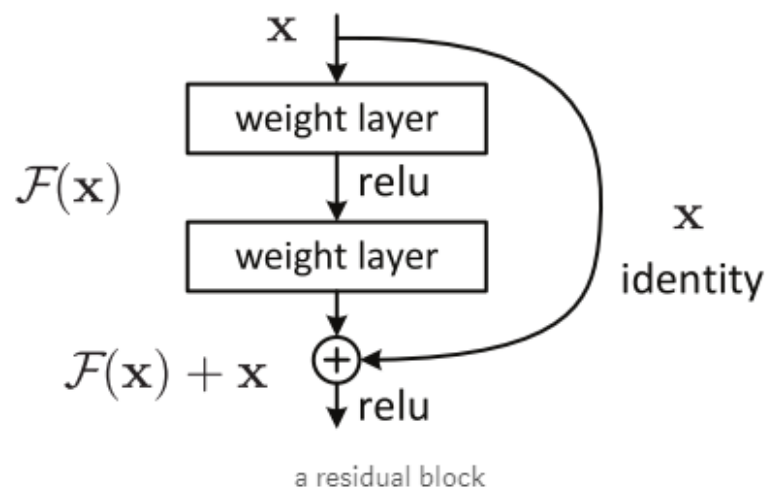
- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data ( $10^6$  vs.  $10^3$  images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week



A. Krizhevsky, I. Sutskever, and G. Hinton,  
[ImageNet Classification with Deep Convolutional Neural Networks](#), NIPS 2012

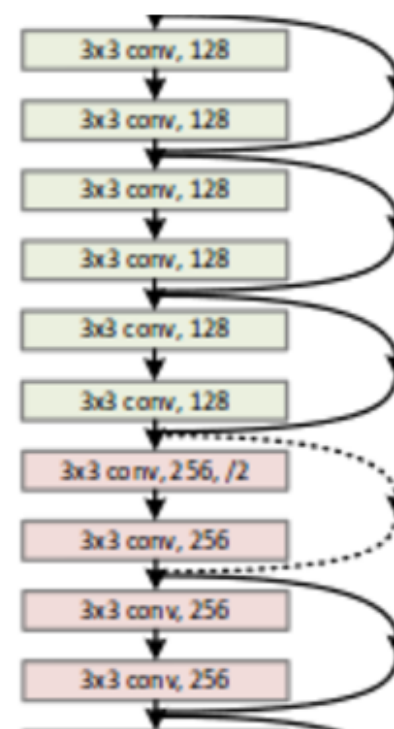
## Deep Residual Network

- The next figure shows a Deep Residual Network (ResNet) (2015)
- A ResNet of 152 layers became world champion in the ImageNet data base
- Other special architectures used in image classification: AlexNet (5 convolutional layers) (2012), VGG network (19 convolutional layers) (2014), GoogleNet (Inception) (22 convolutional layers) (2015), and many variants



Input Image

.....



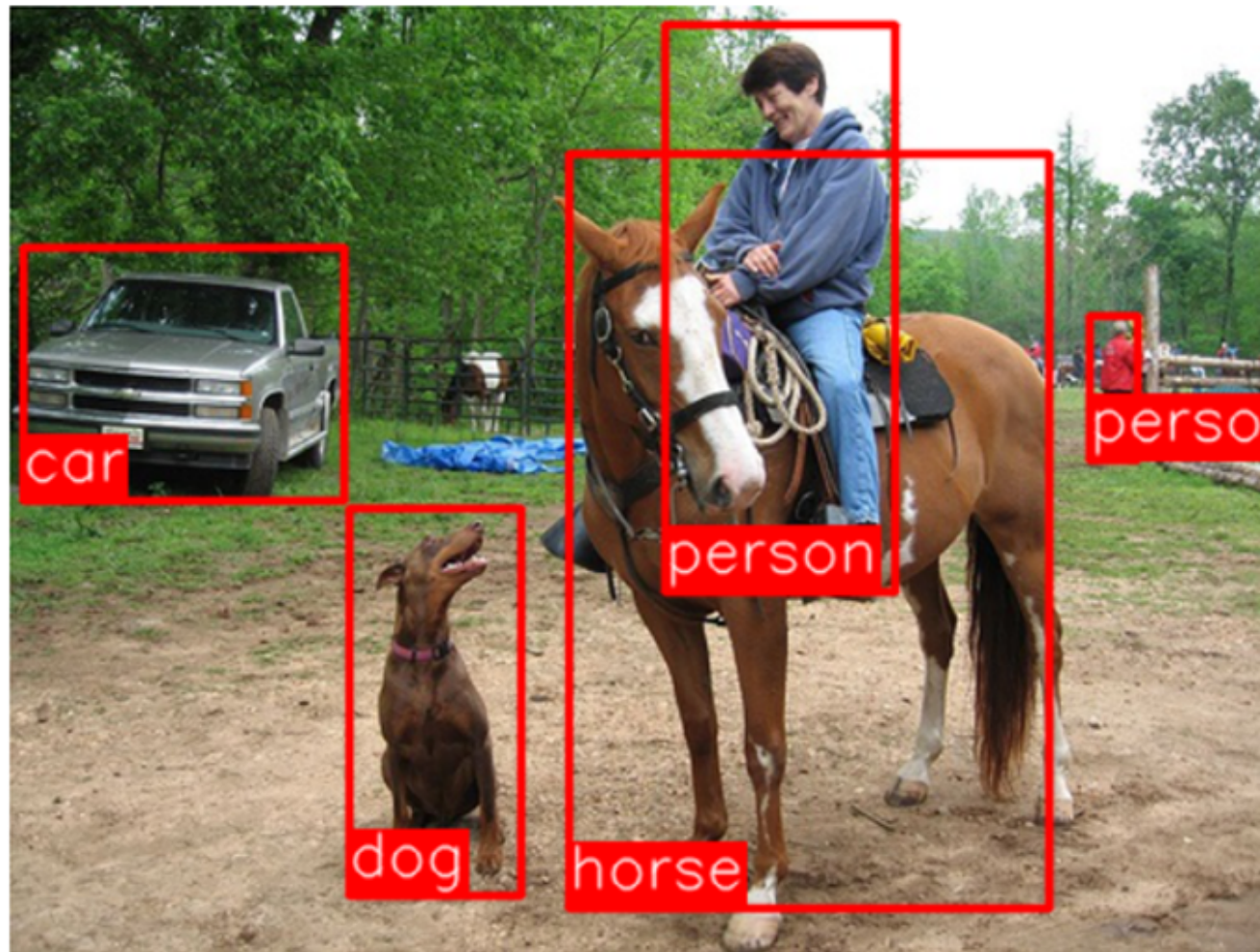
152 layers in total

.....

Output Classes

## Regional CNN

- Task: Finding bounding boxes in images (object detection, object segmentation )
- R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN

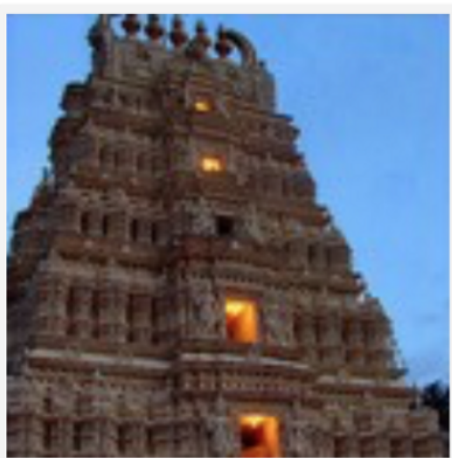


Object detection algorithms such as R-CNN take in an image and identify the locations and classifications of the main objects in the image. Source: <https://arxiv.org/abs/1311.2524>.

## Adversarial Examples

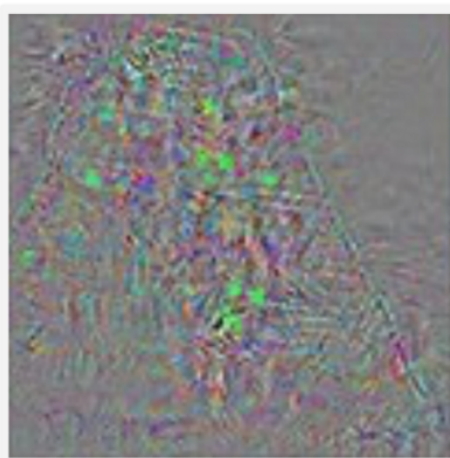
- Deep high-performing DNNs can be fooled by examples intentionally constructed by using an optimization procedure to search for an input that is very close to a real data point and produces a very different label
- Adversarial training are attempts to make DNNs less prone to adversarial examples (active research area)



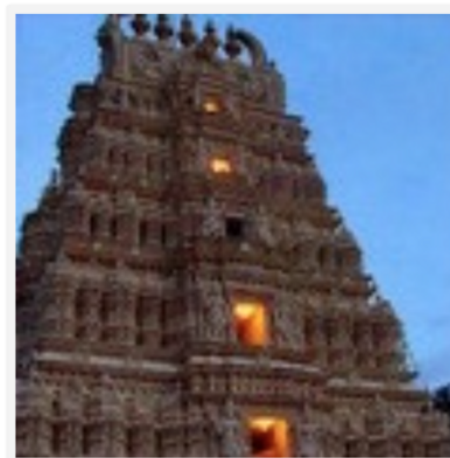


**Original image**

Temple (97%)



**Perturbations**



**Adversarial example**

Ostrich (98%)

## Where from here?

- There will never be enough labelled data to learn it all
- The Google cat recognizer sees more cat images than any child and is not as good
- If one assumes that cat features are not encoded genetically, then unsupervised learning. i.e., understanding the world's statistics might do the job! First attempts: RBM, all sorts of Clustering, autoencoders, ...

## Tools

- **Torch7** is used at facebook, Deep Mind and several groups at Google (based on LuaJIT which is similar to Python)
- **GP-GPU-CUDA:** Facebook, NYU, and Google/Deep Mind all have custom CUDA back-ends for fast/parallel convolutional network training (CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model implemented by the graphics processing units (GPUs) that they produce. CUDA gives program developers direct access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs)
- **Theano:** Python library. Popular in the deep learning community. Theano family:
  - **Blocks + Fuel:** Blocks and Fuel are machine learning frameworks for Python developed by the Montreal Institute of Learning Algorithms (MILA) at the University of Montreal. Blocks is built upon Theano (also by MILA) and allows for rapid prototyping of neural network models. Fuel serves as a data processing pipeline and data interface for Blocks.

- **Keras:** Keras is a minimalist, highly modular neural networks library, written in Python and capable of running on top of either TensorFlow or Theano.
- **Lasagne:** Lasagne is a lightweight library to build and train neural networks in Theano.
- **PyLearn2:** Pylearn2 is a machine learning library.
- **TensorFlow:** TensorFlow is an open source software library for machine learning in various kinds of perceptual and language understanding tasks. Under development: **Tensor Processing Unit (TPU) custom chip**
- **Deeplearning4j** is an open source deep learning library for Java and the Java Virtual Machine
- **Caffe** is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. Yangqing Jia created the project during his PhD at UC Berkeley.

## Successes

- Microsoft Speech (2012): Chief Research Officer Rick Rashid demonstrates a speech recognition breakthrough via machine translation that converts his spoken English words into computer-generated Chinese language.
- Google: Android Speech Recognition: Maps; Image+ (cats etc. ); improve Google translate (ongoing project); Google used the (deep learning) program to update its Google+ photo-search software in May 2013; Apple: SIRI (the iPhone's voice-activated digital assistant, Siri, relies on deep learning.)
- In September 2016, a research team at Google announced the development of the Google Neural Machine Translation system (GNMT) and by November Google Translate began using neural machine translation (NMT)
- Facebook: DeepFace (2014), of the steps detect-align-represent-classify, the representation step is done by a DNNs. Asked whether two unfamiliar photos of faces show the same person, a human being will get it right 97.53 percent of the time. New software developed by researchers at Facebook can score 97.25 percent on the same

challenge, regardless of variations in lighting or whether the person in the picture is directly facing the camera.

- AlexNet is the name of a convolutional neural network, originally written with CUDA to run with GPU support, which competed in the ImageNet Large Scale Visual Recognition Challenge in 2012. The network achieved a top-5 error of 15.3%, more than 10.8 percentage points ahead of the runner up. AlexNet was designed by the SuperVision group, consisting of Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever
- ResNet (2015)
- GANs have been used to produce samples of photorealistic images for the purposes of visualizing new interior/industrial design, shoes, bags and clothing items or items for computer games' scenes (2014)
- Spectacular success of AlphaGo (2015) and AlphaZero (2017) in Go, Chess, and other games
- AutoML, Automatic Statistician, ...
- Self-driving cars

## Conclusions

- Why is this a lecture on Machine Learning and not Deep Learning?
- “If you only know deep learning, you’re pretty shallow” (VT)