Deep Learning
and its application to CV and NLP

Fei Yan
University of Surrey
June 29, 2016
Edinburgh

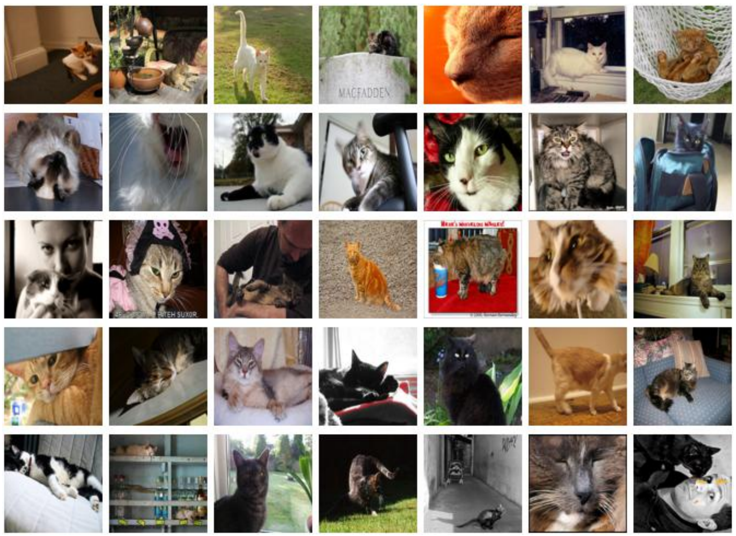
Overview

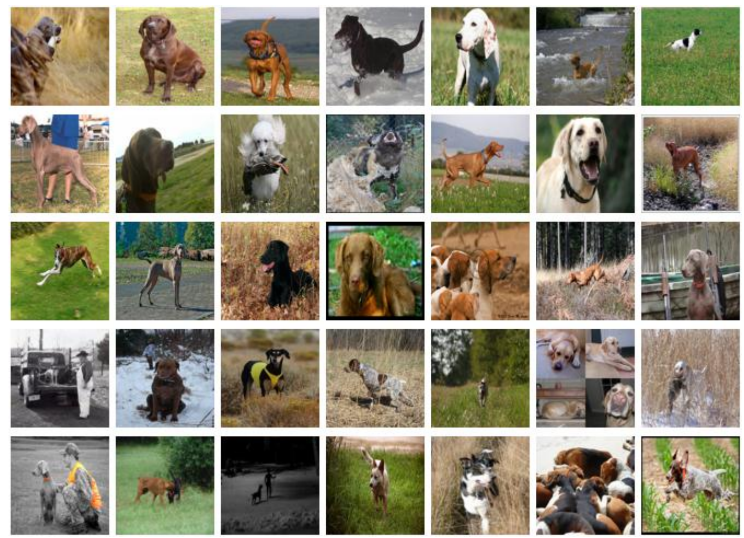
- Machine learning
- Motivation: why go deep
- Feed-forward networks: CNN
- Recurrent networks: LSTM
- An example: geo-location prediction
- Conclusions

Machine learning

- Learn without explicitly programmed
- Humans are learning machines
- Supervised, unsupervised, reinforcement, transfer, multitask …

ML for CV: image classification





ML for NLP: sentiment analysis

- "Damon has never seemed more at home than he does here, millions of miles adrift. Would any other actor have shouldered the weight of the role with such diligent grace?"

- "The warehouse deal TV we bought was faulty so had to return. However we liked the TV itself so bought elsewhere."

ML for NLP: Co-reference resolution

- "John said he would attend the meeting."

- "Barack Obama visited Flint Mich. on Wednesday since findings about the city's lead-contaminated water came to light. … The president said that …"

Overview

- Machine learning
- Motivation: why go deep
- Feed-forward networks: CNN
- Recurrent networks: LSTM
- An example: geo-location prediction
- Conclusions

Motivation: why go deep

- A shallow cat/dog recogniser:
  - Convolve with fixed filters
  - Aggregate over image
  - Apply more filters
  - SVM

$$\begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}$$

Motivation: why go deep

- A shallow sentiment analyser:
  - Bag of words
  - Part-of-speech tagging
  - Named entity recognition
  - ...
  - SVM

Motivation: why go deep

- Shallow learner eg SVM
  - Convexity -> global optimum
  - Good performance
  - Small training sets
- But features manually engineered
  - Domain knowledge required
  - Representation and learning decoupled ie not end-to-end learning

Overview

- Machine learning
- Motivation: why go deep
- Feed-forward networks: CNN
- Recurrent networks: LSTM
- An example: geo-location prediction
- Conclusions

From shallow to deep

$$\begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}$$

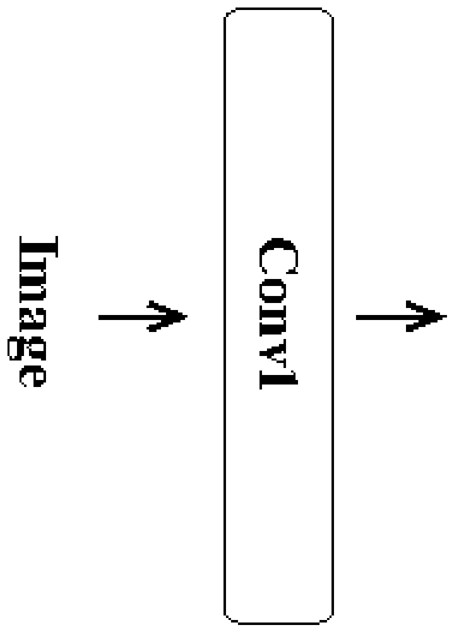
From shallow to deep

$$W^{1,1} = \begin{pmatrix} w_{11}^{1,1} & w_{12}^{1,1} & w_{13}^{1,1} \\ w_{21}^{1,1} & w_{22}^{1,1} & w_{23}^{1,1} \\ w_{31}^{1,1} & w_{32}^{1,1} & w_{33}^{1,1} \end{pmatrix}$$

$$W^{1,2} = \begin{pmatrix} w_{11}^{1,2} & w_{12}^{1,2} & w_{13}^{1,2} \\ w_{21}^{1,2} & w_{22}^{1,2} & w_{23}^{1,2} \\ w_{31}^{1,2} & w_{32}^{1,2} & w_{33}^{1,2} \end{pmatrix}$$

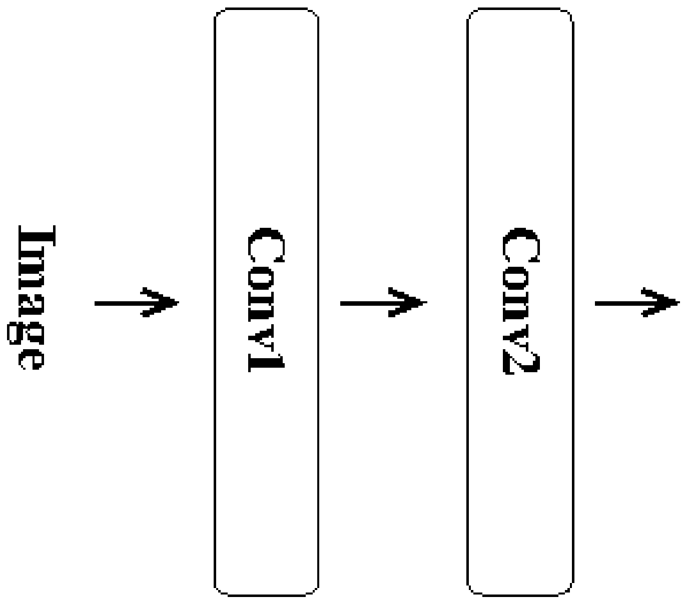
...

From shallow to deep



- 100x100x1 input
- 10 3x3x1 filters
- # of params:
  - 10x3x3x1=90
- Size of output:
  - 100x100x10 with padding and stride=1

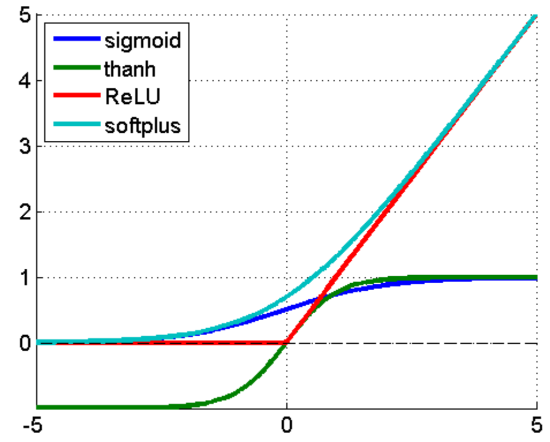
From shallow to deep



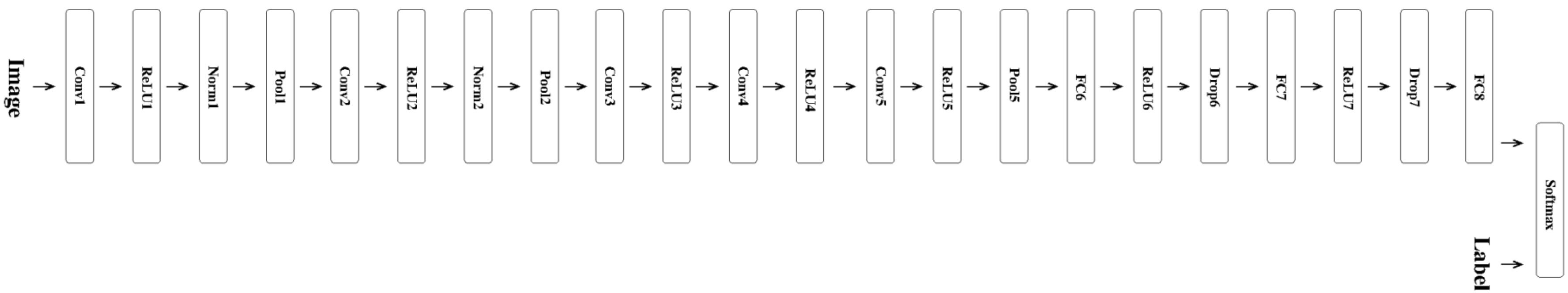
- 100x100x10 input
- 8 3x3x10 filters
- # of params:
  - 8x3x3x10=720
- Size of output:
  - 100x100x8 with padding and stride=1

Other layers

- Rectified linear unit (ReLU)
- Max pooling
  - Location invariance
- Dropout
  - Effective regularisation
- Fully-connected (FC)



Complete network



- Loss:
  - Softmax loss for problem
  - How wrong current prediction is
  - How to change FC8 output to reduce error

Chain rule

- If y if a function of u, and u is a function of x

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

- DNNs are nested functions
  - Output of one layer is input of next

Back-propagation

- If a layer has parameters
  - Convolution, FC
  - O is function of Input I and parameters W

$$\frac{\partial L}{\partial W} = \frac{dL}{dO} \cdot \frac{\partial O}{\partial W} \qquad \frac{\partial L}{\partial I} = \frac{dL}{dO} \cdot \frac{\partial O}{\partial I}$$

- If a layer doesn't have parameters
  - Pooling, ReLU, Dropout
  - O is a function of input I only

$$\frac{dL}{dI} = \frac{dL}{dO} \cdot \frac{dO}{dI}$$

Stochastic gradient descent (SGD)

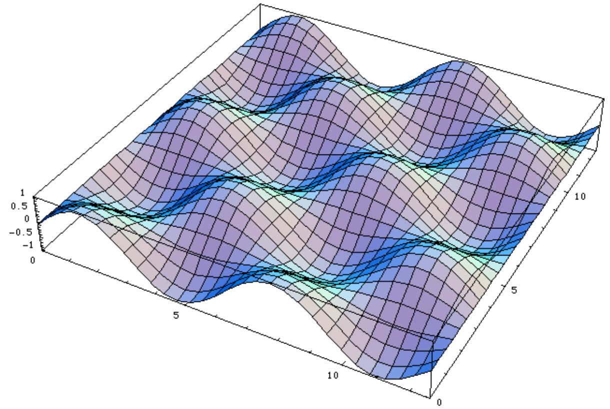
- Stochastic: random mini-batch
- Weight update: linear combination of
  - Negative gradient of current batch
  - Previous weight update

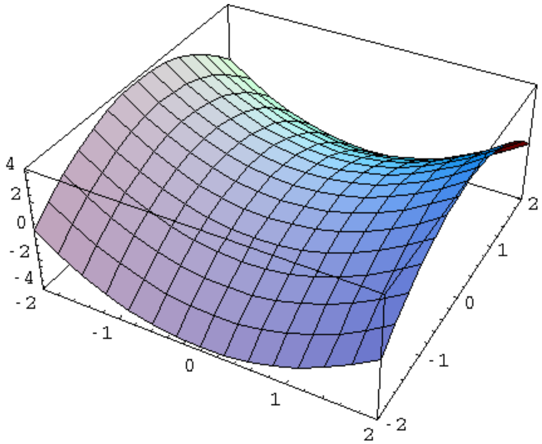
$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t), \quad W_{t+1} = W_t + V_{t+1}$$

- α : learning rate; μ : momentum
- Other variants
  - Adadelta, AdaGrad, etc.

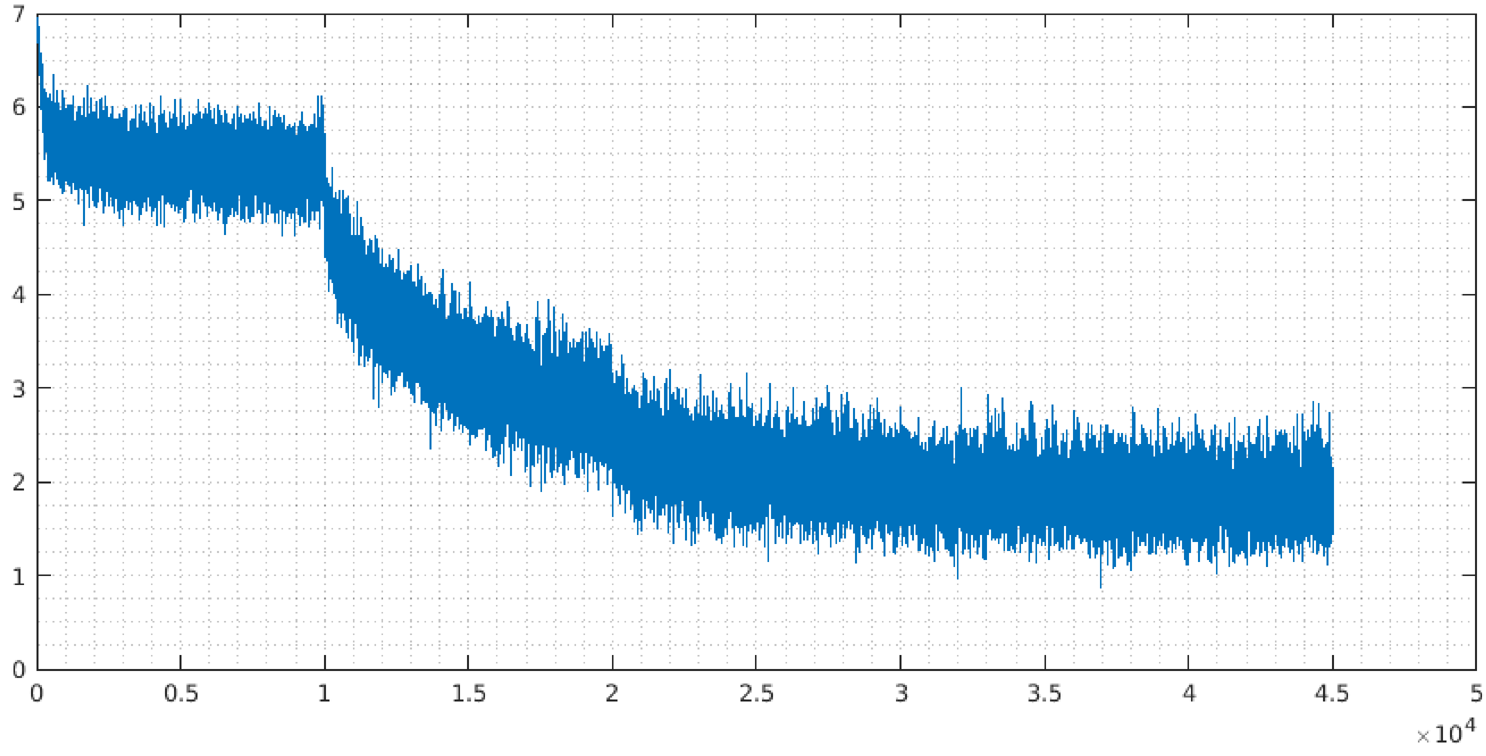
Why SGD works

- Deep NNs are non-convex
- Most critical points in high dimensional functions are saddle points
- SGD can escape from saddle points





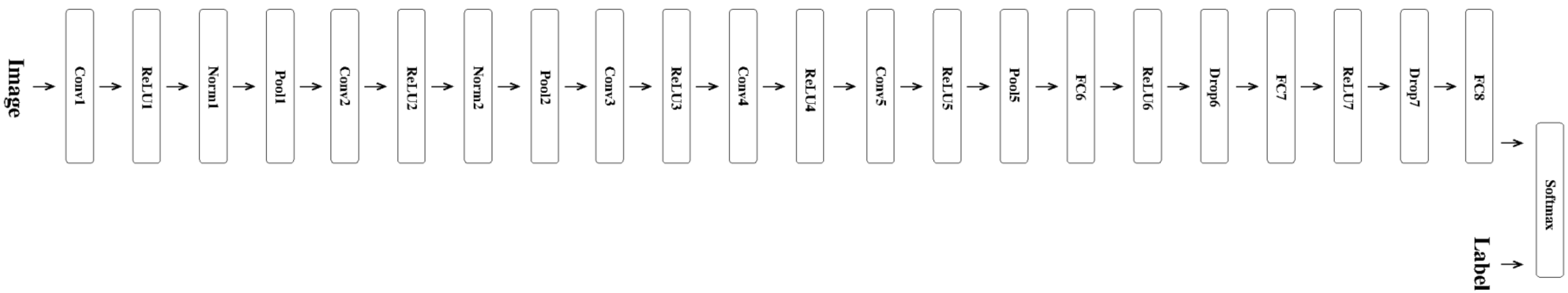
Loss vs. iteration



ImageNet and ILSVRC

- ImageNet
  - # of images: 14,197,122, labelled
  - # of classes: 21,841
- ILSVRC 2012
  - # of classes: 1,000
  - # of train image: ~1,200,000, labelled
  - # of test image: 50,000

AlexNet

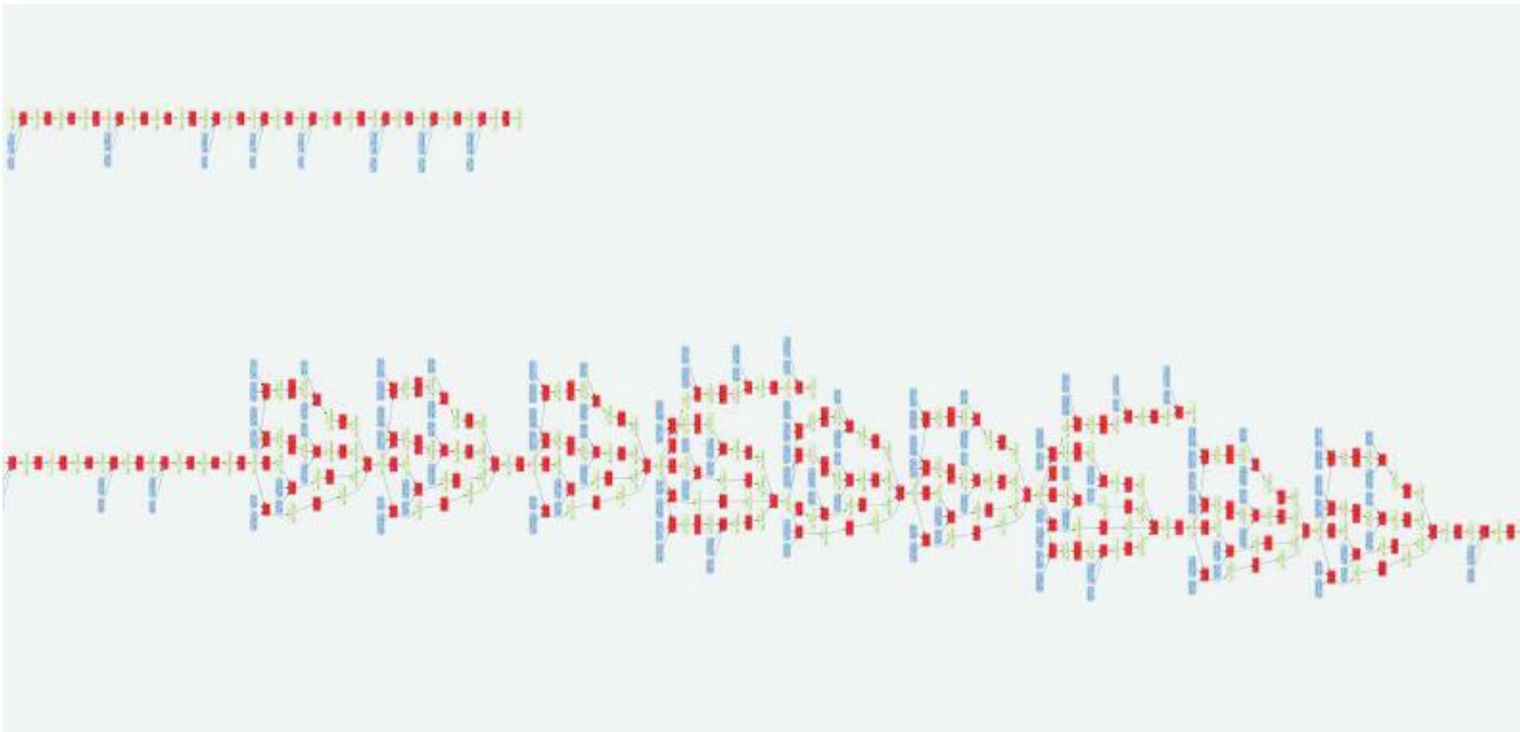


- [Krizhevsky et al. 2012]
- Conv1: 96 11x11x3 filters, stride=4
- Conv3: 384 3x3x256 filters, stride=1
- FC7: 4096 channels
- FC8: 1000 channels

AlexNet

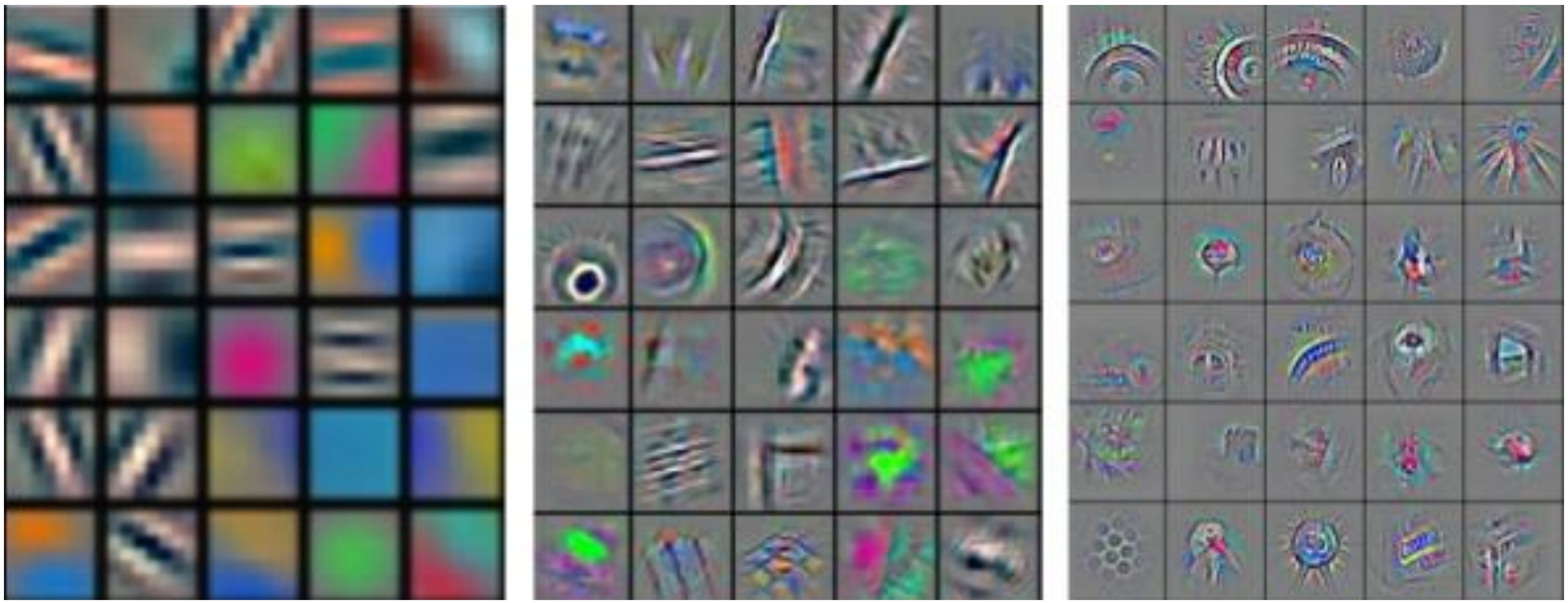
- Total # of params: ~60,000,000
- Data augmentation
  - Translation, reflections, RGB shifting
- 5 days, 2 x Nvidia GTX 580 GPUs
- Significantly improves state-of-the-art
- Breakthrough in computer vision

More recent nets



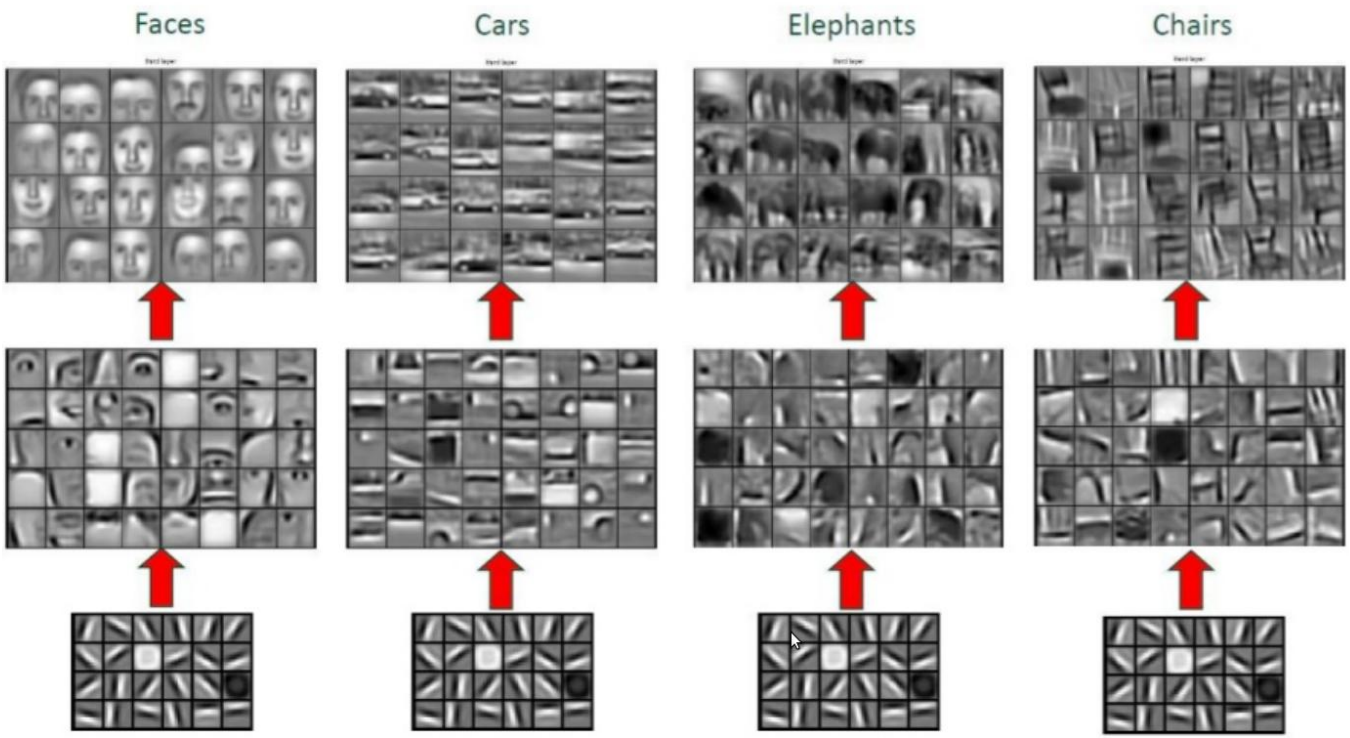
AlexNet 2012 vs GoogleNet 2014

Hierarchical representation



Visualisation of learnt filters. [Zeiler & Fergus 2013]

Hierarchical representation



Visualisation of learnt filters. [Lee et al. 2012]

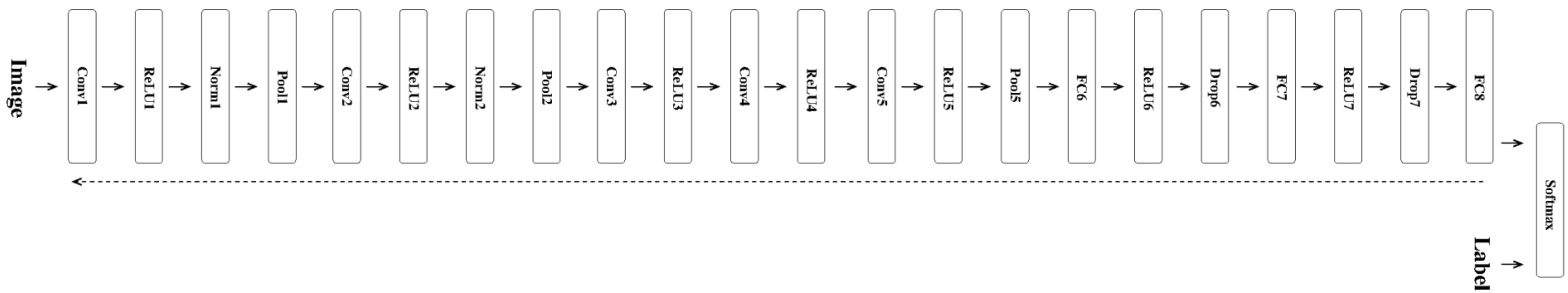
CNN as generic feature extractor

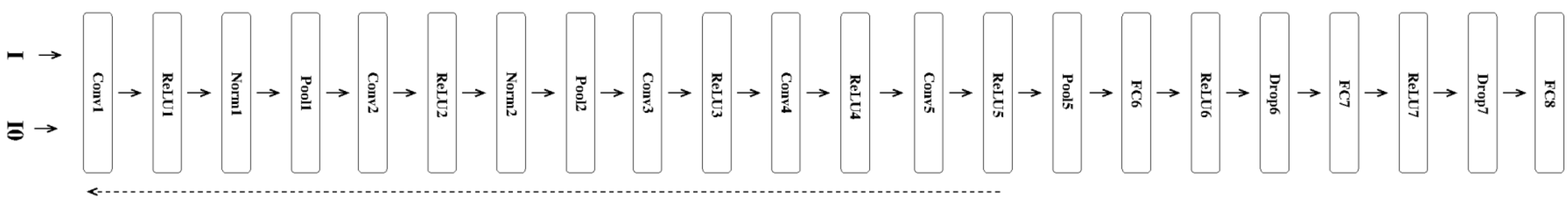
- Given:
  - CNN trained with eg ImageNet
  - A new recognition task/dataset
- Simply:
  - Forward pass, take FC7/ReLU7 output
  - SVM
- Often outperform hand crafted features

CNN as generic feature extractor

Image retrieval with trained CNN. [Krizhevsky et al. 2012]

Neural artistic style





Neural artistic style

- Key idea
  - Hierarchical representation
    => content and style are separable
  - Content: filter responses
  - Style: correlations of filter responses

Neural artistic style

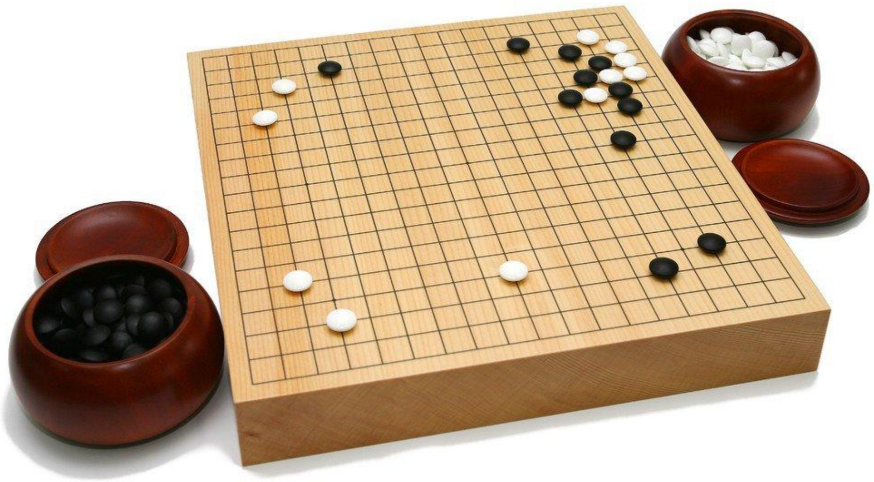
- Input
  - Natural image: content
  - Image of artwork: style
  - Random noise image
- Define content loss and style loss
- Update a random image with BP to minimise:

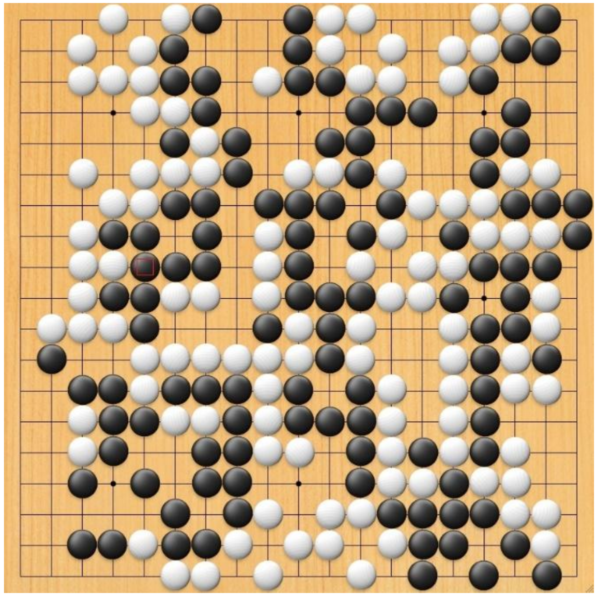
$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Neural artistic style

[Gatys et al. 2015]

Go game





CNN for Go game

- Treated as 19x19 image
- Convolution with zero-padding
- ReLU nonlinearity
- Softmax loss of size 361 (19x19)
- SGD as solver
- No Pooling

AlphaGo

- Policy CNN
  - Configuration -> choice of professional players
  - Trained with 30K+ professional games
- Simulate till end to get binary labels
- Value CNN
  - Configuration -> win/loss
  - Trained with 30M+ simulated games
- Reinforcement learning, Monte-Carlo tree search
- 1202 CPUs + 176 GPUs
- Beating 18 times world champion

Why it didn't work

- Ingredients available in 80s
  - (Deep) Neural networks
  - Convolutional filters
  - Back-propagation
- But
  - Dataset thousands times smaller
  - Computers millions times slower
- Recent techniques/heuristics help
  - Dropout, ReLU

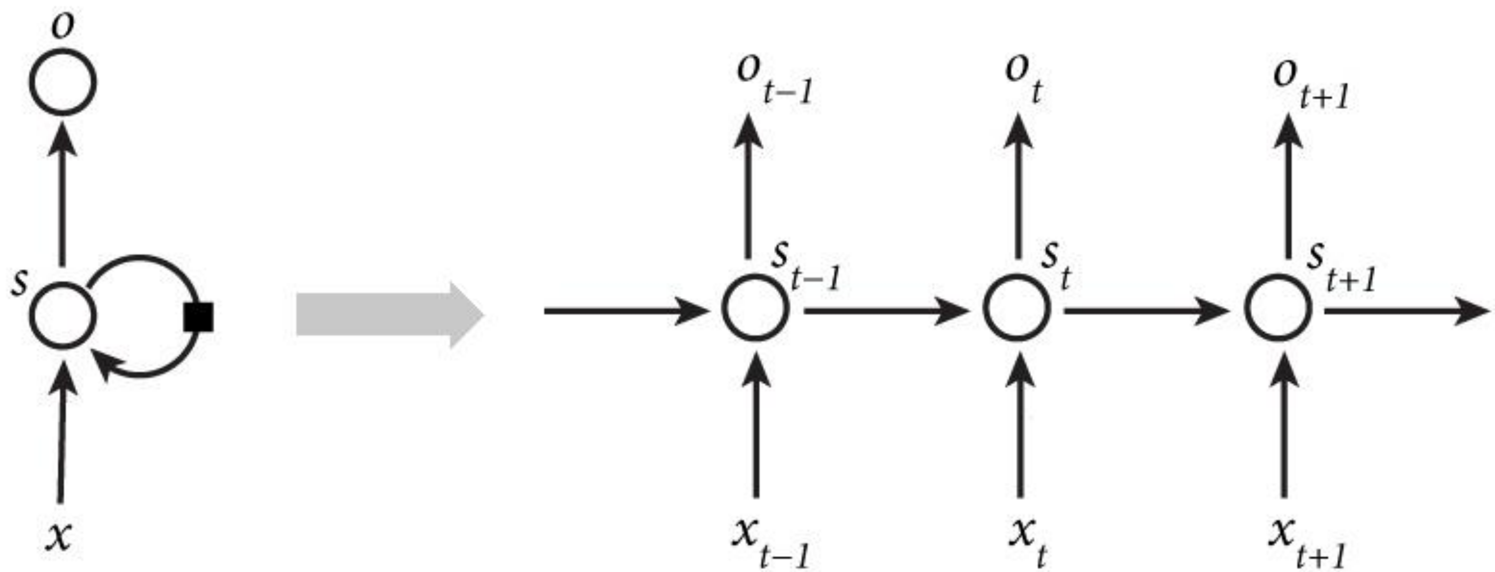
Overview

- Machine learning
- Motivation: why go deep
- Feed-forward networks: CNN
- Recurrent networks: LSTM
- An example: geo-location prediction
- Conclusions

Why recurrent nets

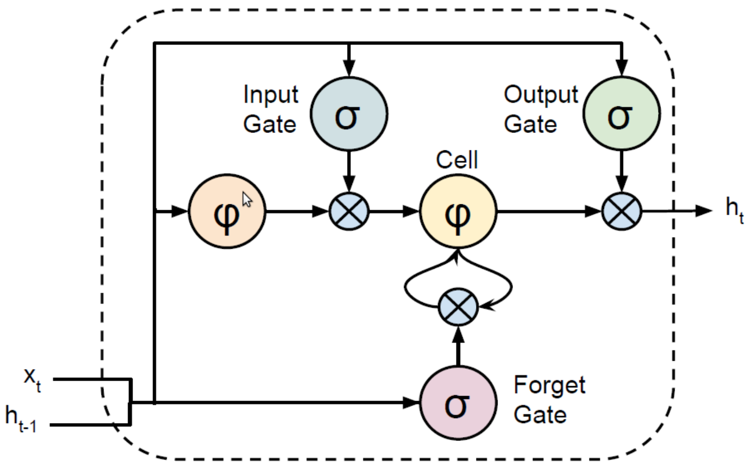
- Feed-forward nets
  - Process independent vectors
  - Optimise over functions
- Recurrent nets
  - Process sequences of vectors
  - Internal state, or “memory”
  - Dynamic behaviour
  - Optimise over programs, much more powerful

Unfolding recurrent nets in time



LSTM

- **LSTM**
  - Input, forget and output gates: i, f, o
  - Internal state: c



[Donahue et al. 2014]

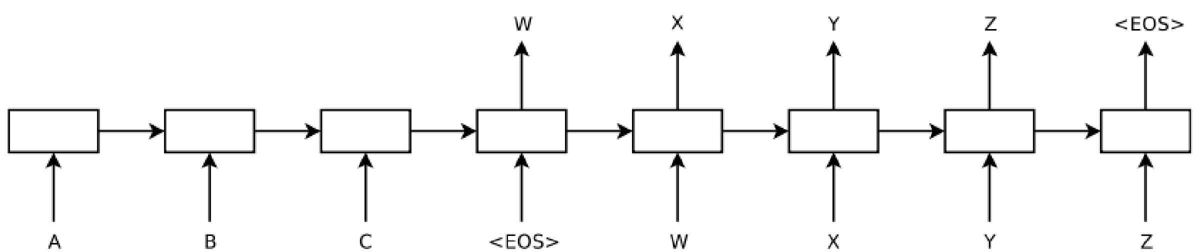
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1})$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1})$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1})$$
$$g_t = \phi(W_{xc}x_t + W_{hc}h_{t-1})$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \phi(c_t)$$

Machine translation

- Sequence to sequence mapping
  - ABC<E> => WXYZ<E>
- Traditional MT:
  - Hand-crafted intermediate semantic space
  - Hand-crafted features

Machine translation

- LSTM based MT:
  - Maximise prob. of output given input
  - Update weights in LSTM by BP in time
  - End-to-end, no feature-engineering
  - Semantic information in LSTM cell



[Sutskever et al. 2014]

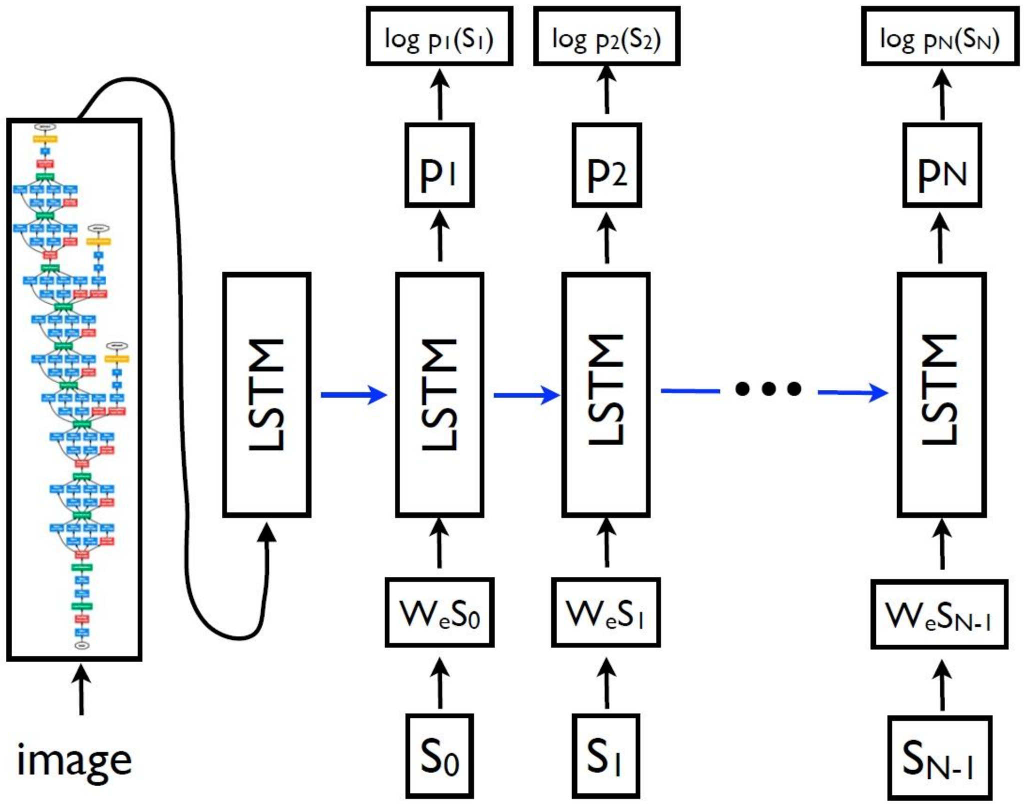
Image captioning

- Image classification
  - Girl/child, tree, grass, flower
- Image captioning
  - Girl in pink dress is jumping in the air
  - A girl jumps on the grass

Image captioning

- Traditional methods
  - Object detector
  - Surface realiser: objects => sentence
- LSTM
  - Inspired by neural machine translation
  - Translate image into sentence

Image captioning



[Vinyals et al. 2014]

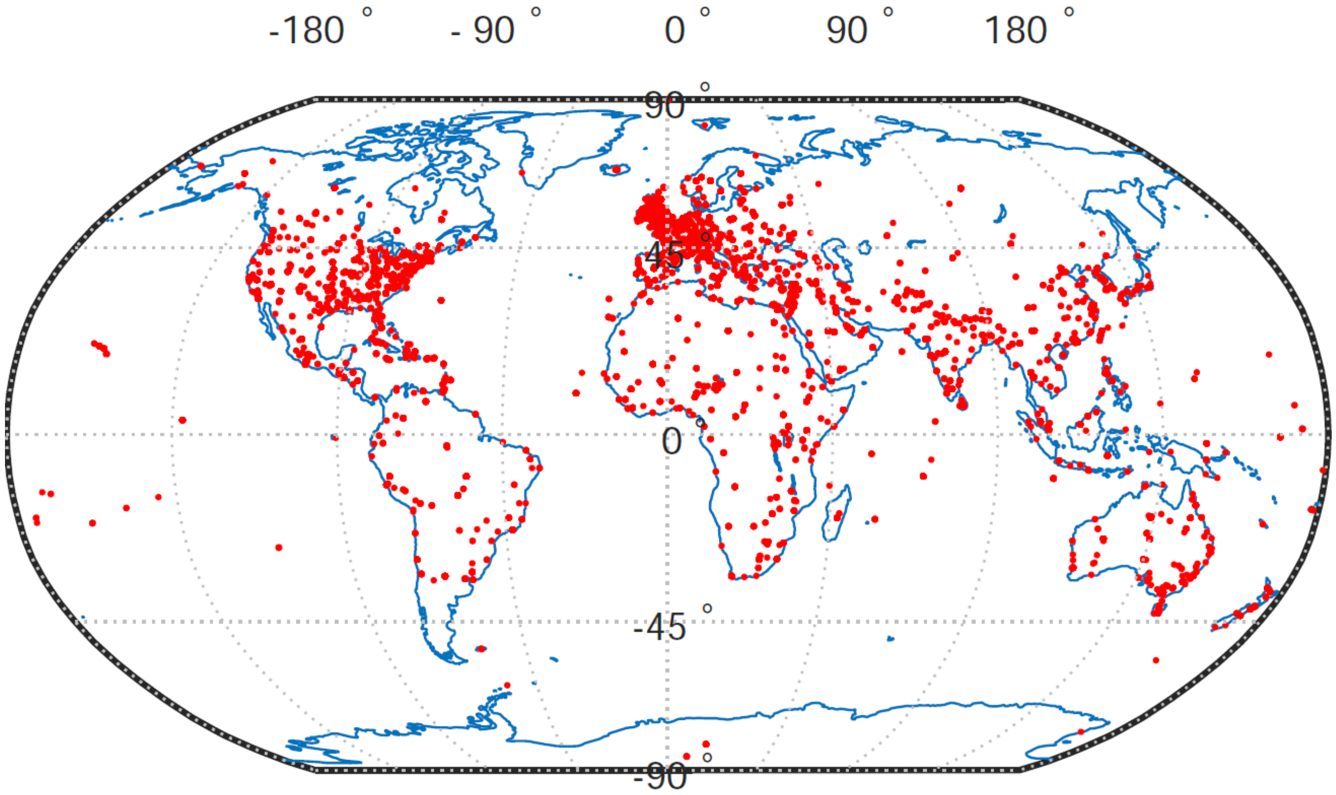
Overview

- Machine learning
- Motivation: why go deep
- Feed-forward networks: CNN
- Recurrent networks: LSTM
- An example: geo-location prediction
- Conclusions

News article analysis

- BreakingNews dataset
  - 100k+ news articles
  - 7 sources: BBC, Yahoo, WP, Guardian, …
  - Image + caption
  - Metadata: comments, geo-location, …
- Tasks
  - Article illustration
  - Caption generation
  - Popularity prediction
  - Source prediction
  - Geo-location prediction

Geo-location prediction



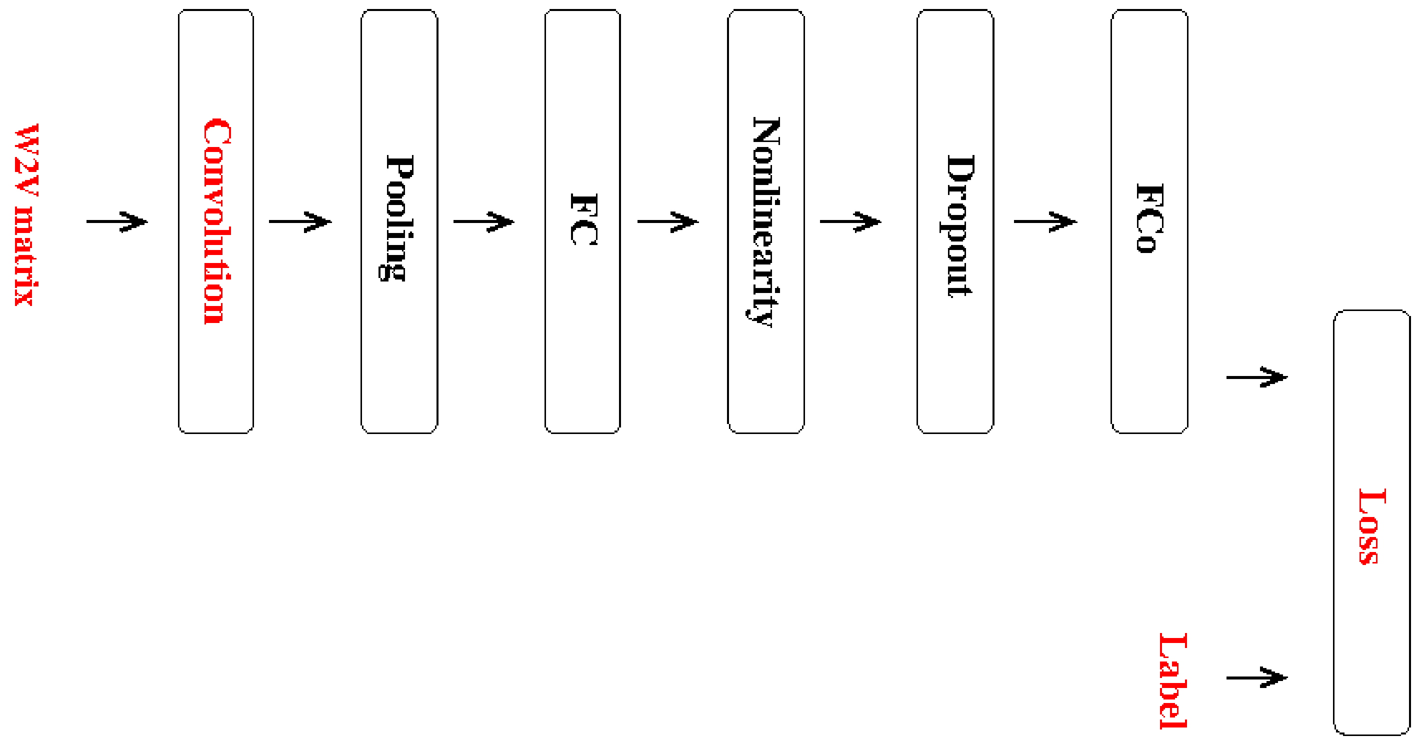
Word2Vec embedding

- Word embedding
  - Words to vectors
  - Low dim. compared to vocabulary size
- Word2Vec
  - Unsupervised, neural networks [Mikolov et al. 2015]
  - Trained on large corpus eg 100+ billion words
  - Vectors close if similar context

Word2Vec embedding

- W2V arithmetic
  - King - Queen ~= man - woman
  - knee - leg ~= elbow - arm
  - China - Beijing ~= France - Paris
  - human - animal ~= ethics
  - library - book ~= hall
  - president - power ~= prime minister

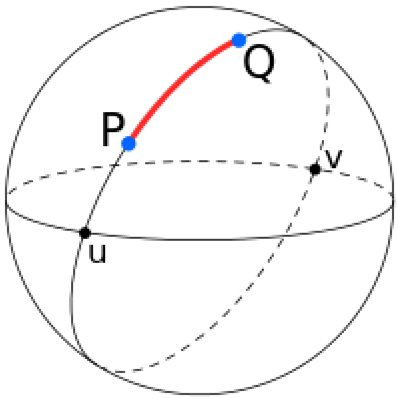
Network



W2V matrix
Convolution
Pooling
FC
Nonlinearity
Dropout
FCo
Loss
Label

Geoloc loss

- Great circle
  - Circle on sphere with same centre as the sphere
- Great circle distance (GCD)
  - Distance along great circle
  - Shortest distance on sphere



Geoloc loss

- Given two (lat, long) pairs

$$\mathbf{y} = [y_1, y_2]^\top \quad \mathbf{z} = [z_1, z_2]^\top$$

- A good approximation to GCD

$$\mathrm{GCD} = R \cdot \arccos(\sin y_1 \sin z_1 + \cos y_1 \cos z_1 \cos \delta)$$

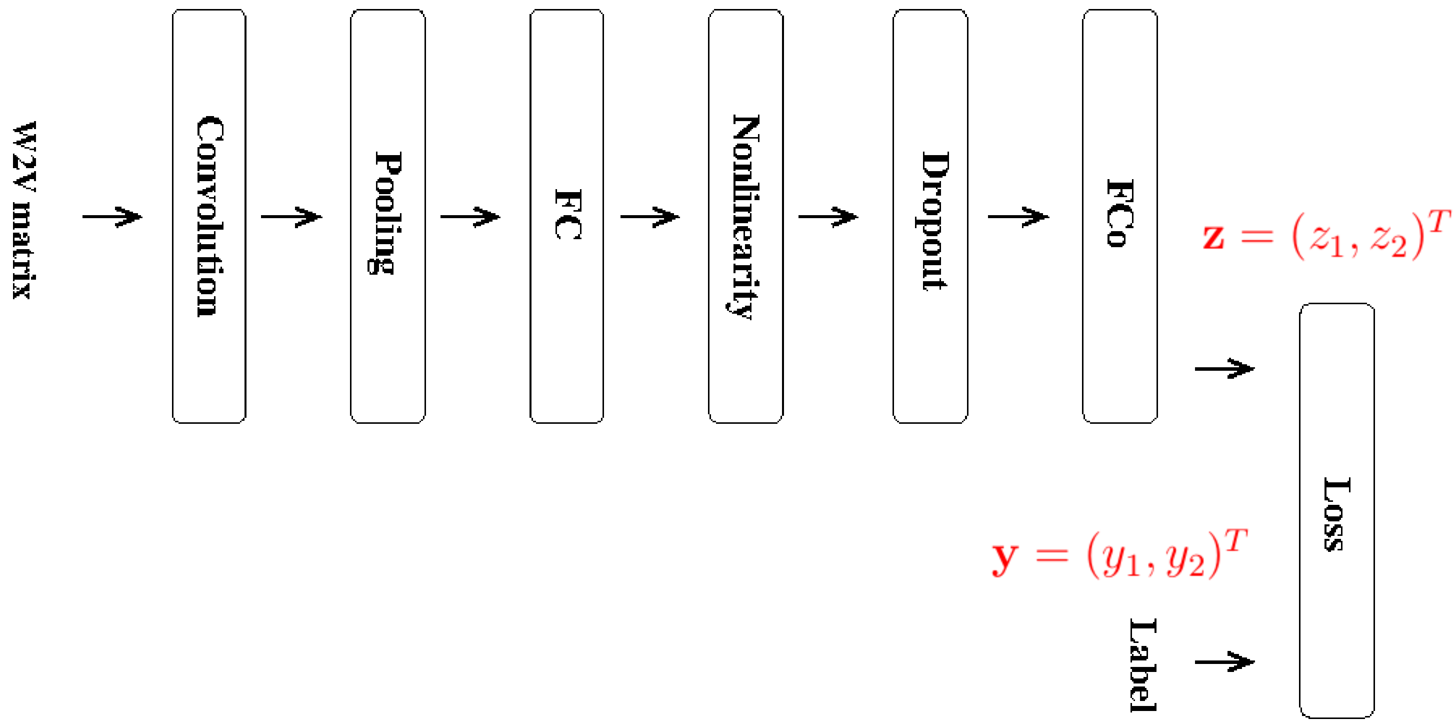
where R is radius of Earth, and

$$\delta = z_2 - y_2$$

- Geoloc loss

$$L = \arccos(\sin y_1 \sin z_1 + \cos y_1 \cos z_1 \cos \delta)$$

Geoloc loss


W2V matrix
Convolution
Pooling
FC
Nonlinearity
Dropout
FCo
$\mathbf{z} = (z_1, z_2)^T$
$\mathbf{y} = (y_1, y_2)^T$
Label
Loss

Geoloc loss

- Gradient w.r.t. z

$$\frac{\partial L}{\partial \mathbf{z}} = \begin{pmatrix} -\frac{1}{\sqrt{1-\phi^2}}(\sin y_1 \cos z_1 - \cos y_1 \sin z_1 \cos \delta) \\ -\frac{1}{\sqrt{1-\phi^2}}(-\cos y_1 \cos z_1 \sin \delta) \end{pmatrix}$$

where

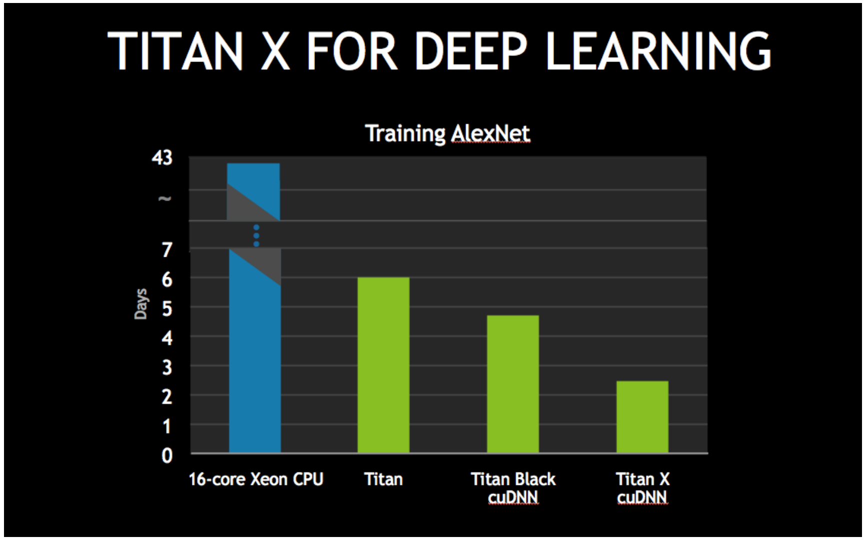
$$\phi = \sin y_1 \sin z_1 + \cos y_1 \cos z_1 \cos \delta$$

- All other layers are standard
- Chain rule, back-propagation, etc.

Practical issues

- Hardware
  - Get a powerful GPU
- Software
  - Choose a library
- What code do I need to write?
  - Solver def. and net def.
  - Optionally: your own layer(s)

GPU



TITAN X FOR DEEP LEARNING
Training AlexNet
Days
43
~
7
6
5
4
3
2
1
0
16-core Xeon CPU
Titan
Titan Black cuDNN
Titan X cuDNN

Libraries

Software	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Has pretrained models	Recurrent Nets	Convolutional Nets	RBM/DBNs
Caffe	C++, Python[5]	C++, command line, Python, MATLAB[6]	No	Branch,[7] pull request,[8] third party implementation[9]	Yes	Yes[10]	Yes	Yes	No[11]
CNTK	C++	Command line;[14] C++, Python and .NET interfaces coming[15]	Yes[16]	No	Yes	No	Yes[17]	Yes[17]	?[18]
Deeplearning4j	Java, Scala, C	Java, Scala, Clojure	?	No[19]	Yes[20]	Yes[21]	Yes	Yes	Yes
MXNet	C++, Python, Julia, Matlab, Go, R, Scala	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala	Yes	On roadmap[26]	Yes	Yes[27]	Yes	Yes	Yes
Neural Designer	C++	Graphical user interface	Yes	No	No	?	No	No	No
OpenNN	C++	C++	Yes	No	No	?	No	No	No
SINGA[28]	C++, Python	Python, C++	No	No	Yes	No	Yes	Yes	Yes
SystemML[29]	Java, R	?	?	?	?	?	?	?	?
TensorFlow	C++, Python	Python, C/C++	No	On roadmap[31][33]	Yes	No	Yes	Yes	Yes
Theano	Python	Python	Yes	Under development[34]	Yes	Through Lasagne's model zoo[35]	Yes	Yes	Yes
Torch	C, Lua	Torch, C, utility library for C++/OpenCL[37]	Yes	Third party implementations[38]	Third party implementations[39]	Yes[40]	Yes	Yes	Yes

Wikipedia: comparison of deep learning software

What you need to code

- solver.prototxt
  - Solver hyper-params
- train.prototxt
  - Network architecture
  - Layer hyper-params
- Layer implementation C++/CUDA
  - Forward pass
  - Backward propagation
  - Efficient GPU programming, CUDA kernel

solver.prototxt & train.prototxt

```
net: "train.prototxt"

solver_mode: GPU
device_id: 0

type: "SGD"
momentum: 0.9
weight_decay: 0.0005

base_lr: 0.1
lr_policy: "step"
gamma: 0.1
stepsize: 1000

max_iter: 1000000

display: 20
test_iter: 1000
test_interval: 500

snapshot: 500
snapshot_prefix: "geolo"

```

```

layer {
  name: "FCo"
  type: "InnerProduct"
  bottom: "DO"
  top: "FCo"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}

layer {
  name: "Geo_loss"
  type: "GeoLoss"
  bottom: "FCo"
  bottom: "Geolo"
  top: "Geo_loss"
  loss_weight: 1
  geo_loss_param {
    verbosity: 0
  }
}

```

Overview

- Machine learning
- Motivation: why go deep
- Feed-forward networks: CNN
- Recurrent networks: LSTM
- An example: geo-location prediction
- Conclusions

Conclusions

- Why go deep
- CNN and LSTM
- Example: geo-location prediction
- Apply DL to my problem:
  - CNN or LSTM?
  - Network architecture, loss
  - Library and GPU
  - (Little) Coding

What's not covered

- Unsupervised learning
  - Auto-encoder, restricted Boltzmann machine (RBM)
- Reinforcement learning
  - Actions in an environment that maximise cumulative reward
- Transfer learning, Multitask learning
- Application to audio signal processing