Welcome to Theories of Deep Learning (DL)

This course will introduce you to Deep Learning and various theories being developed to explain partial answers to the remarkable efficacy of DL.

Health Warning:

- This is not your typical course in mathematics, the material presented is new (this century) and has not been polished to the level normally seen in a mathematics course.
- You will not be presented with a unifying theory of DL.
- Few, if any, questions here will have definitive answers.
- Assessment will require reading and understanding recent, mostly 2018-current, research articles.

Outline for today

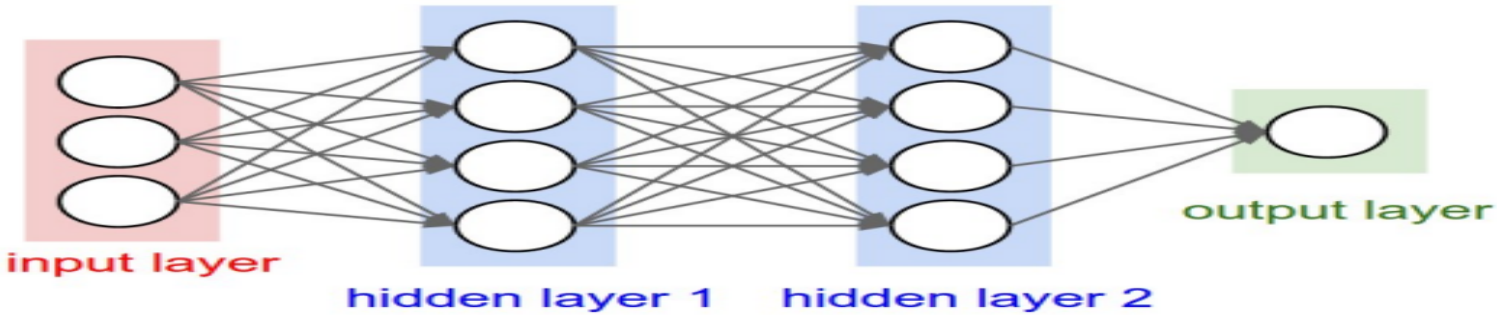
- Introduction to the ingredients that make up deep learning:
  - architectures including Feedforward, convolutional, and other nets
  - examples of data sets commonly used in deep learning
  - raising the issue of how such networks are training
- A few examples of current applications to motivate why DL has become of such interest recently.
- Outline of some of the theoretical issues we may discuss.

Example of a Feedforward network / multilayer perceptron

A feedforward net is a nonlinear function composed of repeated affine transformation followed by a nonlinear action:

$$h_{i+1} = \sigma_j \left(W^{(i)} h_i + b^{(i)} \right) \qquad \text{for} \quad i = 1, \ldots, N-1$$

where $W^{(i)} \in \mathbb{R}^{n_{i+1} \times n_i}$ and $b^{(i)} \in \mathbb{R}^{n_{i+1}}$ and $\sigma(\cdot)$ is a nonlinear activation such as ReLU, $\sigma(z) := max(0, z) = z_+$.
The depth, or number of layers, of this network is N and if $N > 1$ it is referred to as "deep."
The input to the net is h_1, the output is h_N, and h_i for intermediate $i = 2, \cdot, N-1$ are referred to as "hidden" layers.



This Feedforward is an example of a network "architecture."
https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/feedforward.html

Deep learning requires a deep net, loss function, and data

The network "Weights" $W^{(i)}$ (and shifts $b^{(i)}$) are learnt to fit a task for a particular data set; the collection of all network weights are normally summarised as a variable θ.

A "labeled" data set is a collection of input, $x(j) = h_1(j)$, and desired output $y(j)$, pairs $\{(x(j), y(j))\}_{j=1}^{m}$.

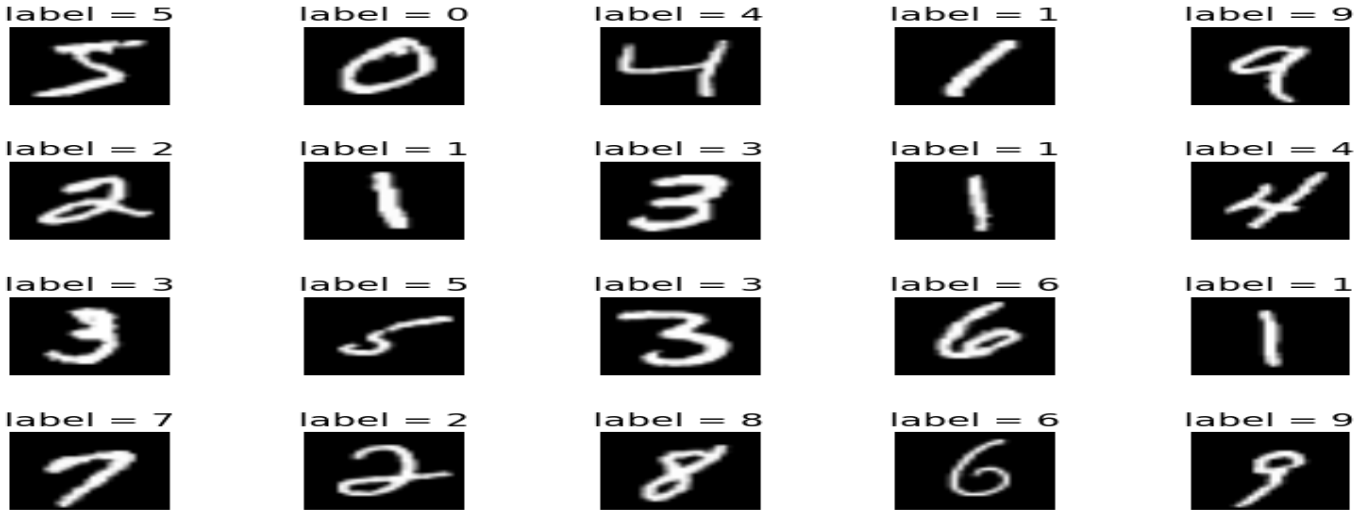
The net is trained by minimising a loss function $L(x(i), y(i); \theta)$ summed over all data pairs; that is

$$\min_{\theta} \sum_{i=1}^{m} L(x(i), y(i); \theta).$$

and the resulting learned net is, $H(\cdot; \theta)$.

Example of a data set: MNIST[2]

MNIST is a collection of 70,000 digitised hand written digits from 0 to 9 in the form of a gray scale image of size 28×28 along with a label indicating which digit each image is of.



[1]

Vectorising each image gives $x(j) \in \mathbb{R}^{784}$ and with ten output classes we set $y(j) \in \mathbb{R}^{10}$ where the index of $y(j)$ denotes the index; that is for an input $x(j)$ corresponding to digit 4 we set $y(j)(\ell) = 1$ for $\ell = 5$ and $y(j)(\ell) = 0$ for $\ell \neq 5$.

[1] https://corochann.com/mnist-dataset-introduction-1138.html

[2] http://yann.lecun.com/exdb/mnist/

Example of a data set, architecture, and loss functions[3]

LeCun et. al. considered, amongst others, a 2 layer feedforward net for this data set and loss function with architecture $W^{(1)} \in \mathbb{R}^{100\times 728}$ and $W^{(2)} \in \mathbb{R}^{10\times 100}$ and sigmoid activation $\sigma(z) = \frac{1}{1+e^{-z}}$.

They used individual loss function of the sum of squares, e.g.

$$L(x(i), y(i); \theta) := (y(i) - H(x(i); \theta))^2$$

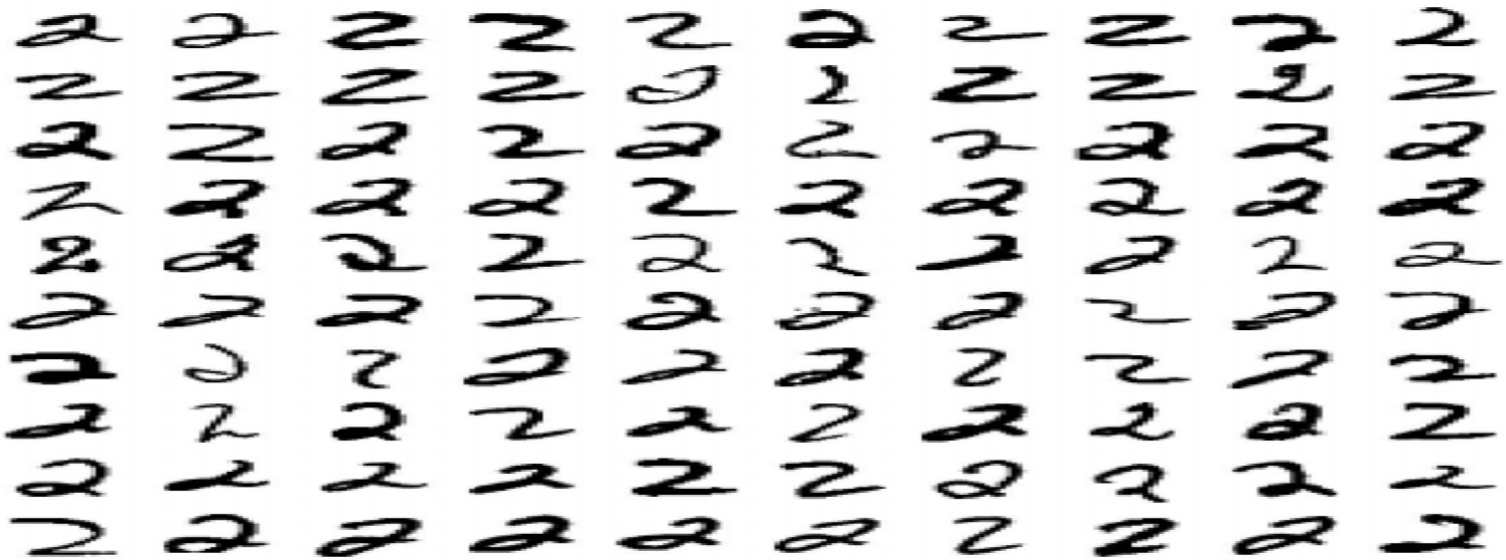
An optimisation algorithm, such as backprop, is applied to learn the network weights so as to minimize the loss function on a "training" set and then tested on a "test" set. For MNIST 50,000 images are classed as training and 10,000 as test.

With the above architecture, having $73,910$ parameters, they achieved a 4.7% classification accuracy on the test set.

[3] `http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf`

Considering the LeCun et. al. 2 layer feedforward net[4]

The resulting net $H(x;\theta)$ is a function from $\mathbb{R}^{728}$ to $\mathbb{R}^{10}$ whose goal is to map points from a given class to a single point; that is, all images of the digit 4 should be mapped to the single point $y(2)$ whose 3^{rd} entry is 1 and all other entries are zero.

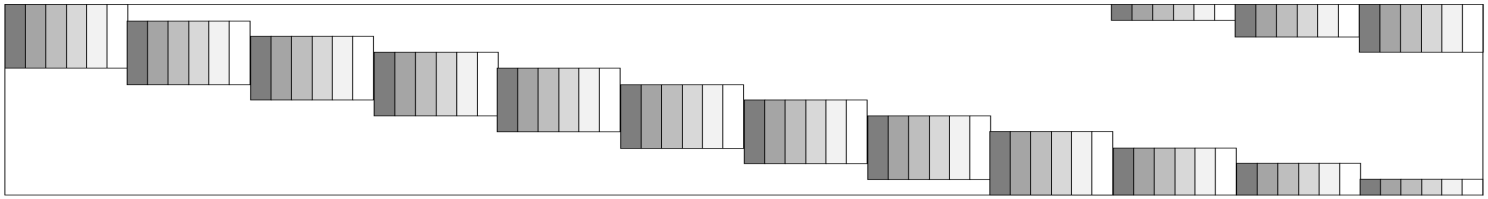


The function $H(x;\theta)$ had its, $\theta \in \mathbb{R}^{73,910}$ learned, based on 50,000 examples so as to achieve 4.7% error rate. Also training on artificially distorted examples decreased the error to 2.5%.

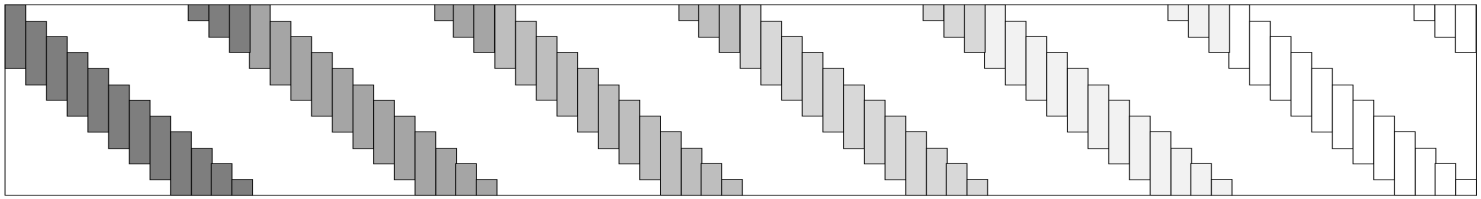
[4] http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

Many architecture choices beyond fully connected: CNNs

Convolutional neural network layers impose a structure on $W^{(i)}$:



(a) A convolutional matrix.



(b) A concatenation of banded and Circulant matrices.

[5]

$W^{(i)}$ is composed of a "mask" (usually of compact support, say just living on 9 pixels) translated by some amount which is referred to as "stride." These "masks" are sometimes referred to as "features." Additional nonlinearities include "max pooling."

[5] https://arxiv.org/pdf/1607.08194.pdf

Revisiting MNIST classification by LeCun et. al. 1998

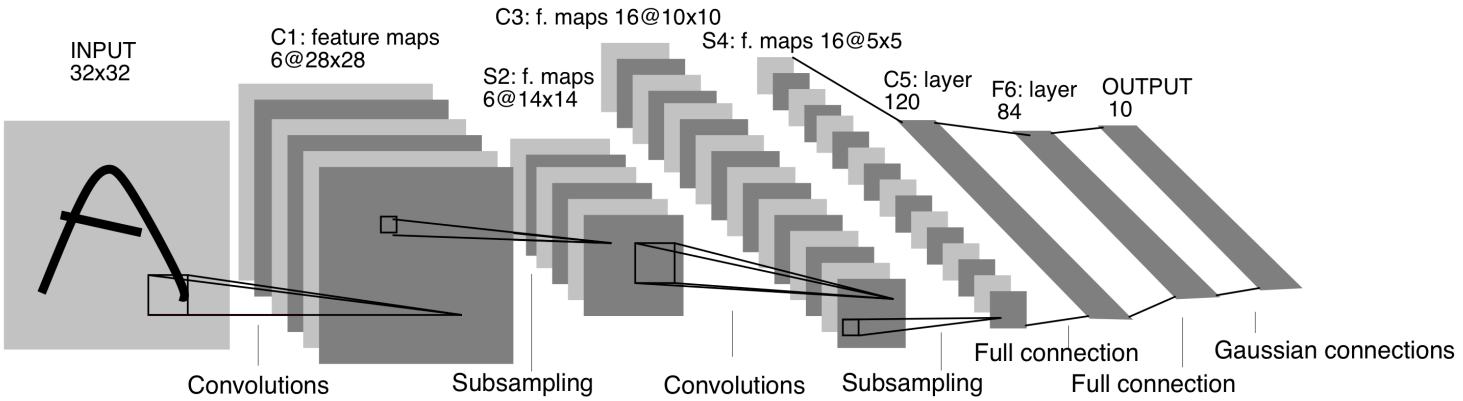


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

6

C1: conv. layer with 6 feature maps, 5 by 5 support, stride 1.

S2 (and S4): non-overlapping 2 by 2 blocks which equally sum values, mult by weight and add bias.

C3: conv. layer with 16 features, 5 by 5 support, partial connected.

C5: 120 features, 5 by 5 support, no stride; i.e. fully connected.

F6: fully connected, $W \in \mathbb{R}^{84\times 120}$.

Final layer is more complex, gaussian RBF.

Classification error rate from 4.5% 2 layer FFN to 0.95% error; or 0.8% when trained with artificially distorted data.

[6] `http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf`

Since 1998 the data sets have grown: ImageNet (2009+)

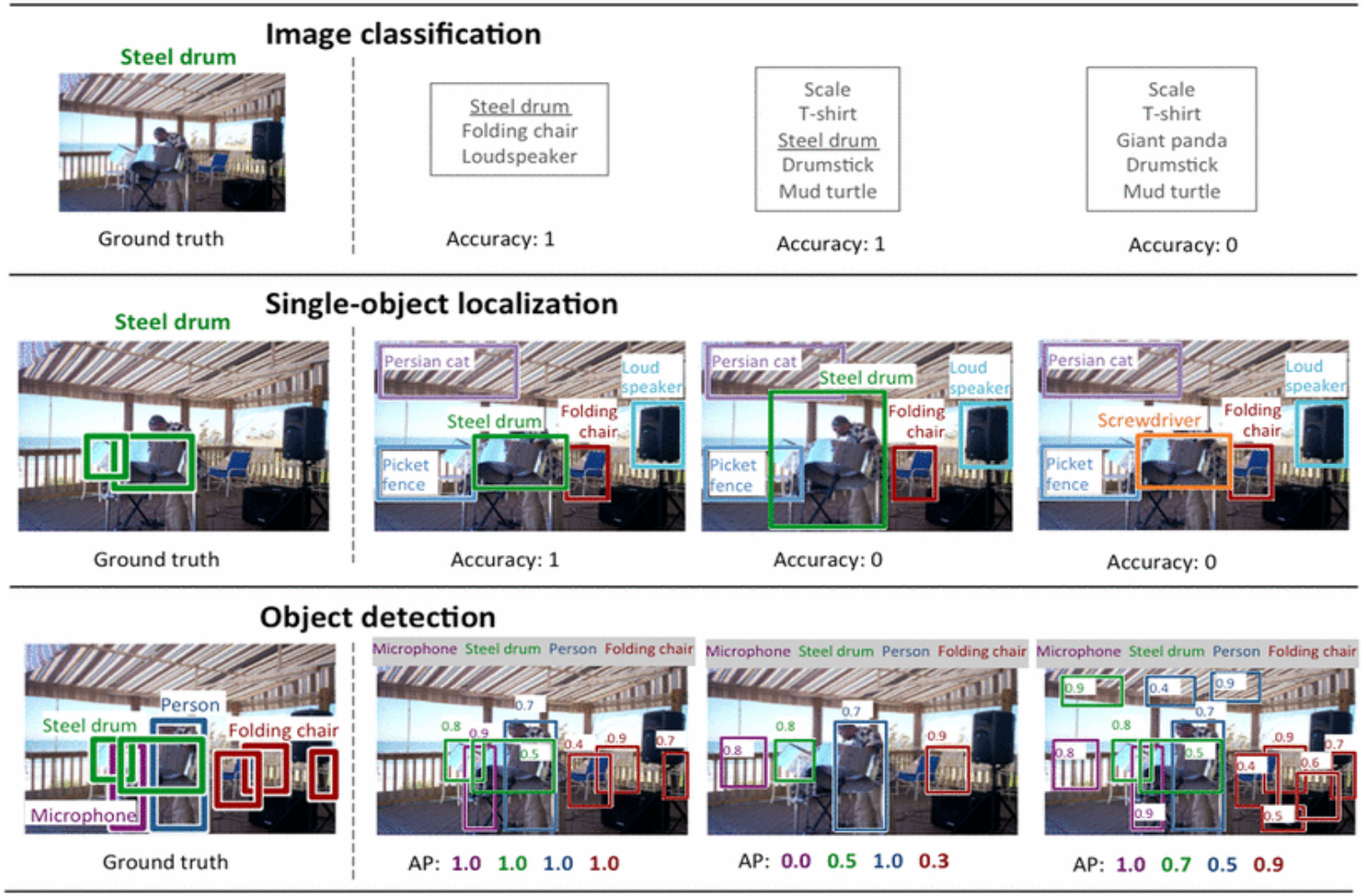
[7]

ImageNet was first presented in 2009 and was central to the development of image classification methods through the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

2010-14: Image classification; 1.2 million training labeled images
2011-14: Single object localisation; 524,000 training labeled bbox
2013-14: All object classification per scene; 456,000 training set

[7] http://image-net.org

Example of ILSVRC localisation and object detection

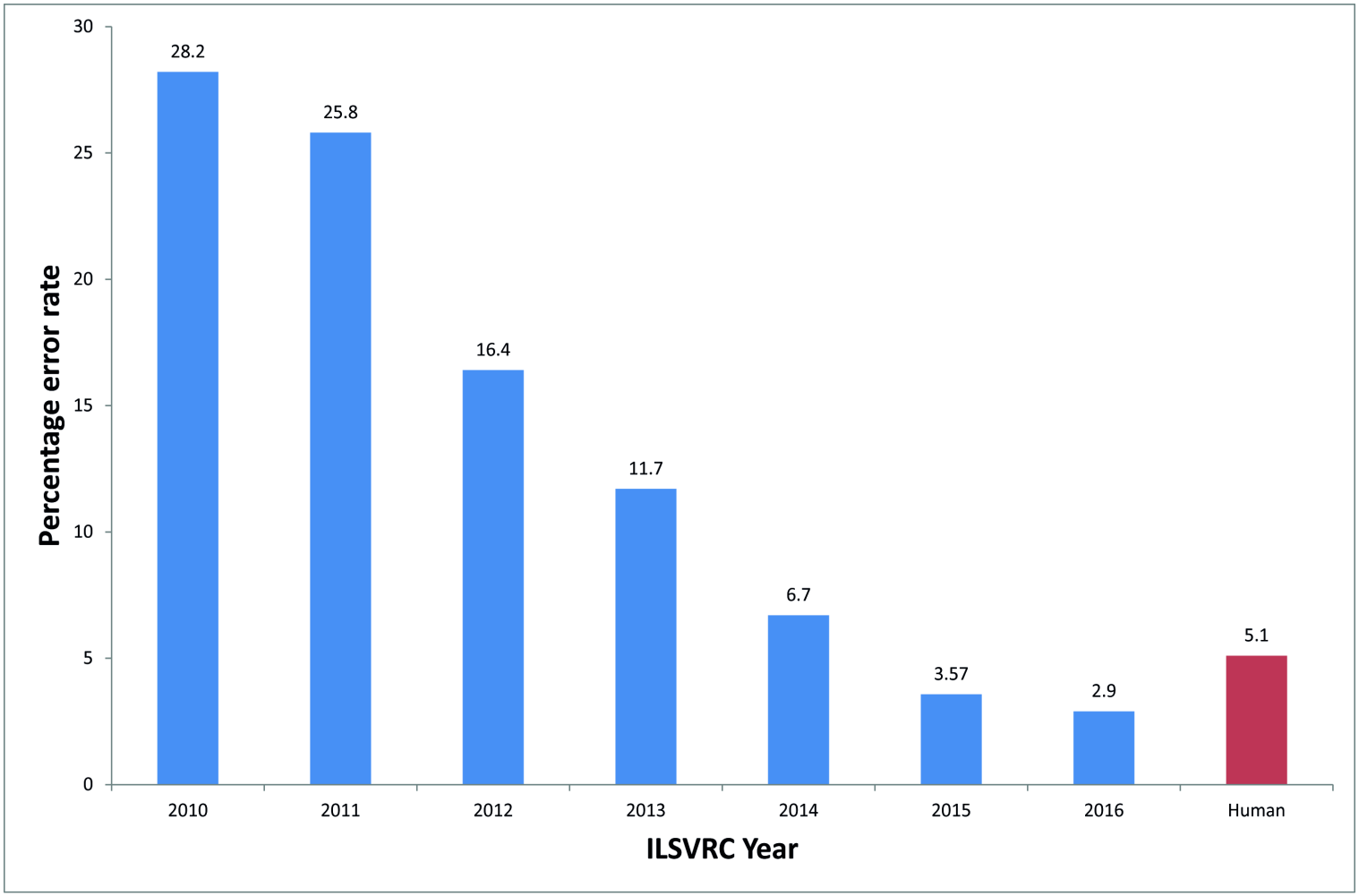


8

2013-14: All object classification per scene; 456,000 training set

[8] ImageNet Large Scale Visual Recognition Challenge, Russakovsky et al. 2015.

ILSVRC classification error rate by year



[9]

2012 ILSVRC classification won using 7 layer CNN by Krizhevsky, Sutskever, and Hinton; users in widespread use of ConvNets.

[9] `http://image-net.org/challenges/LSVRC/2012/results.html`

Optimisation algorithms and initialisations allow depth

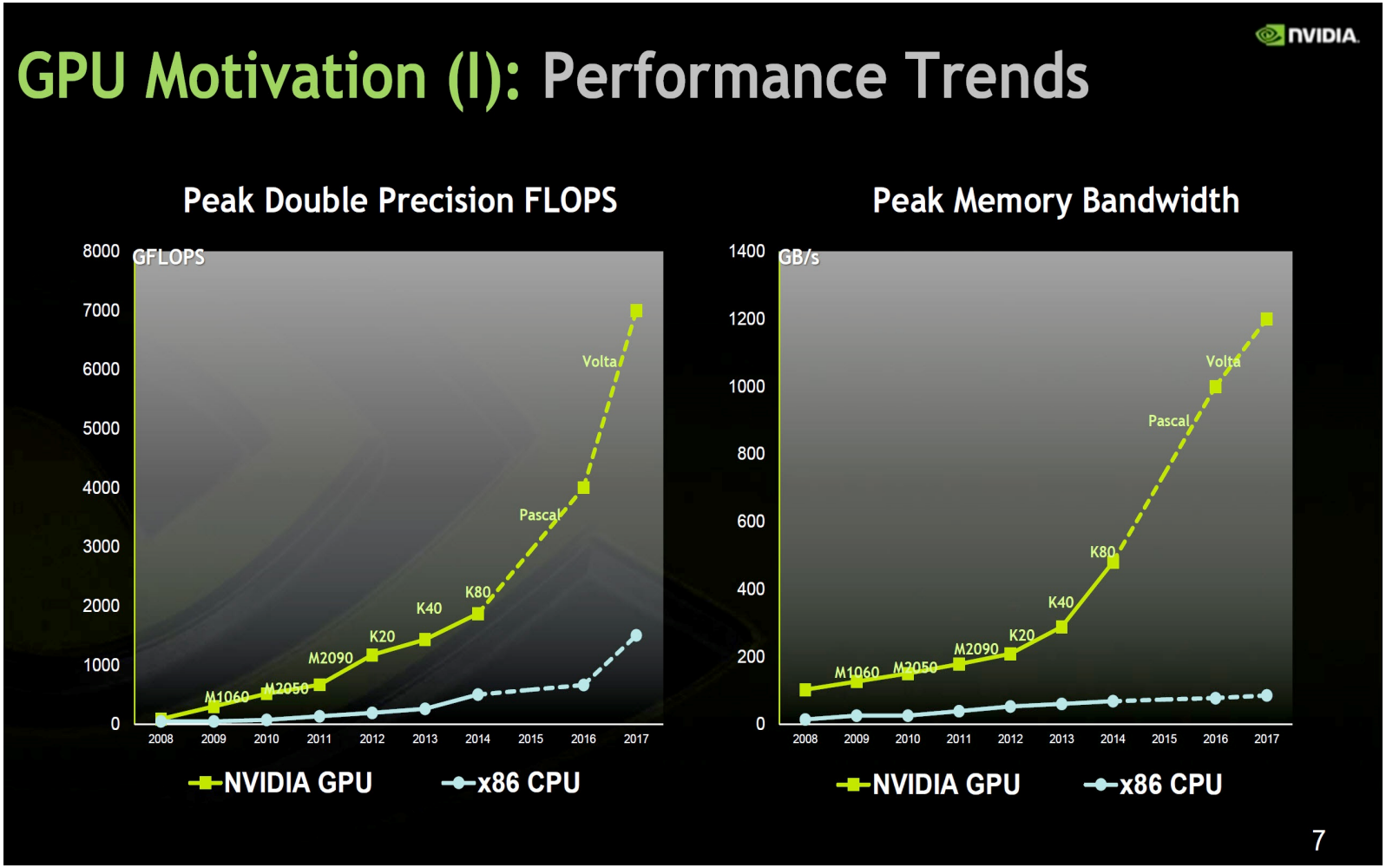
Training ever larger number of parameters and larger data sets is possible in large part to improvements in optimisation algorithms:

- Backpropagation is an efficient way to compute the gradient of the training loss function
- Stochastic gradient descent, and advanced variants, such as Adagrad and Adam, have been key to reducing the computational time and ability to train networks.

Understanding of network initialisation and heuristics have allowed training deep networks; without these advances deeper networks typically perform worse then shallow networks:

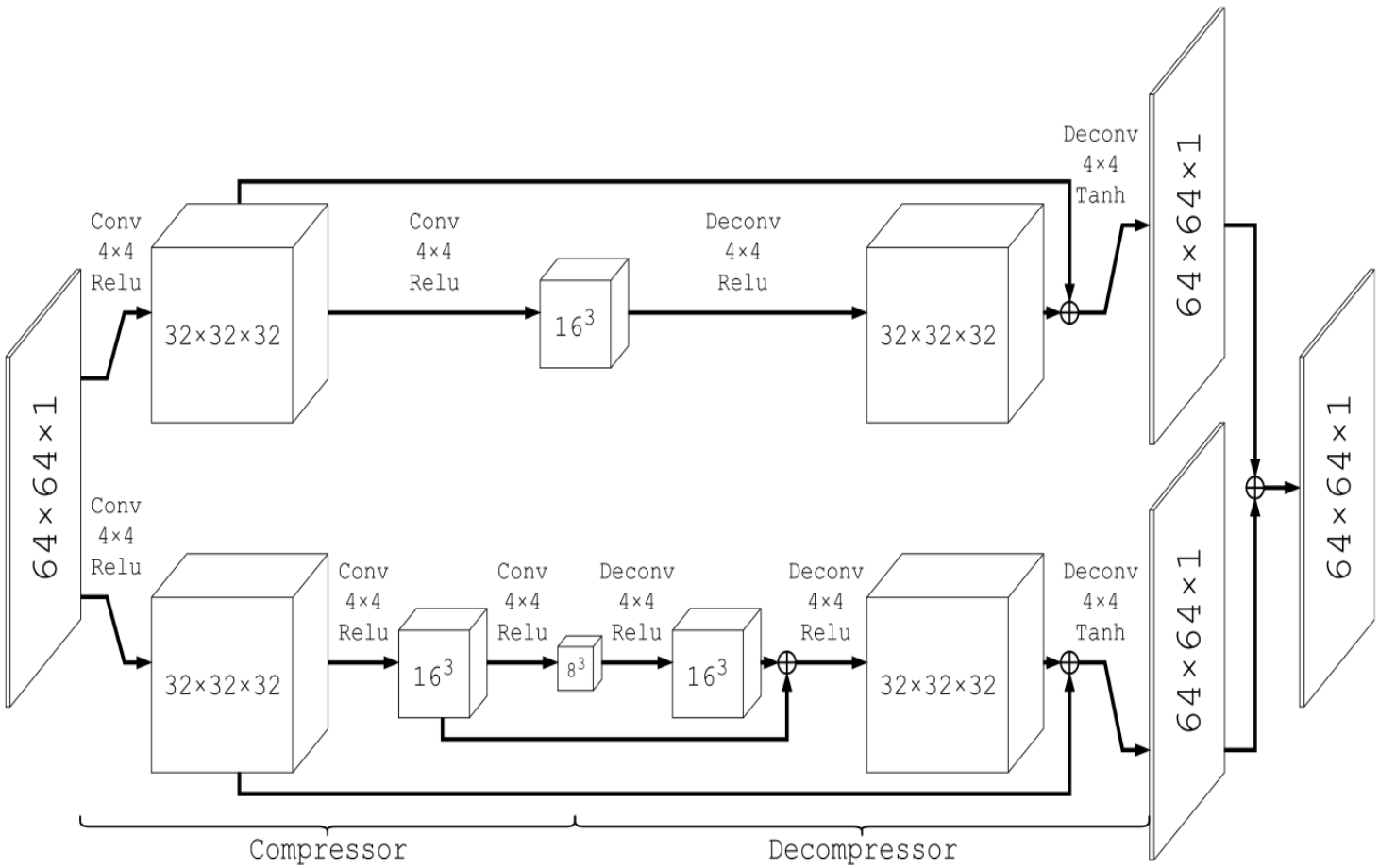
- Deep networks can suffer from gradients that either diverge or converge to zero with depth (vanishing or exploding),,
- Careful choice of network weight initialisation can greatly reduce this effect, allowing training deeper networks,
- Heuristics and new architectures, such as batch-normalisation and residual networks, greatly aided training with depth.

Training of deep nets aided by increased compute



Cloud computing and software such as PyTorch and TensorFlow have made training the deep net parameters computationally tractable.

A few deep learning applications: denoising



10

Deep learning tasks go beyond classification. Can make use of vast data that is "corrupted" and learn the inverse process.

[10] https://arxiv.org/pdf/1809.10410.pdf

A few deep learning applications: super-resolution

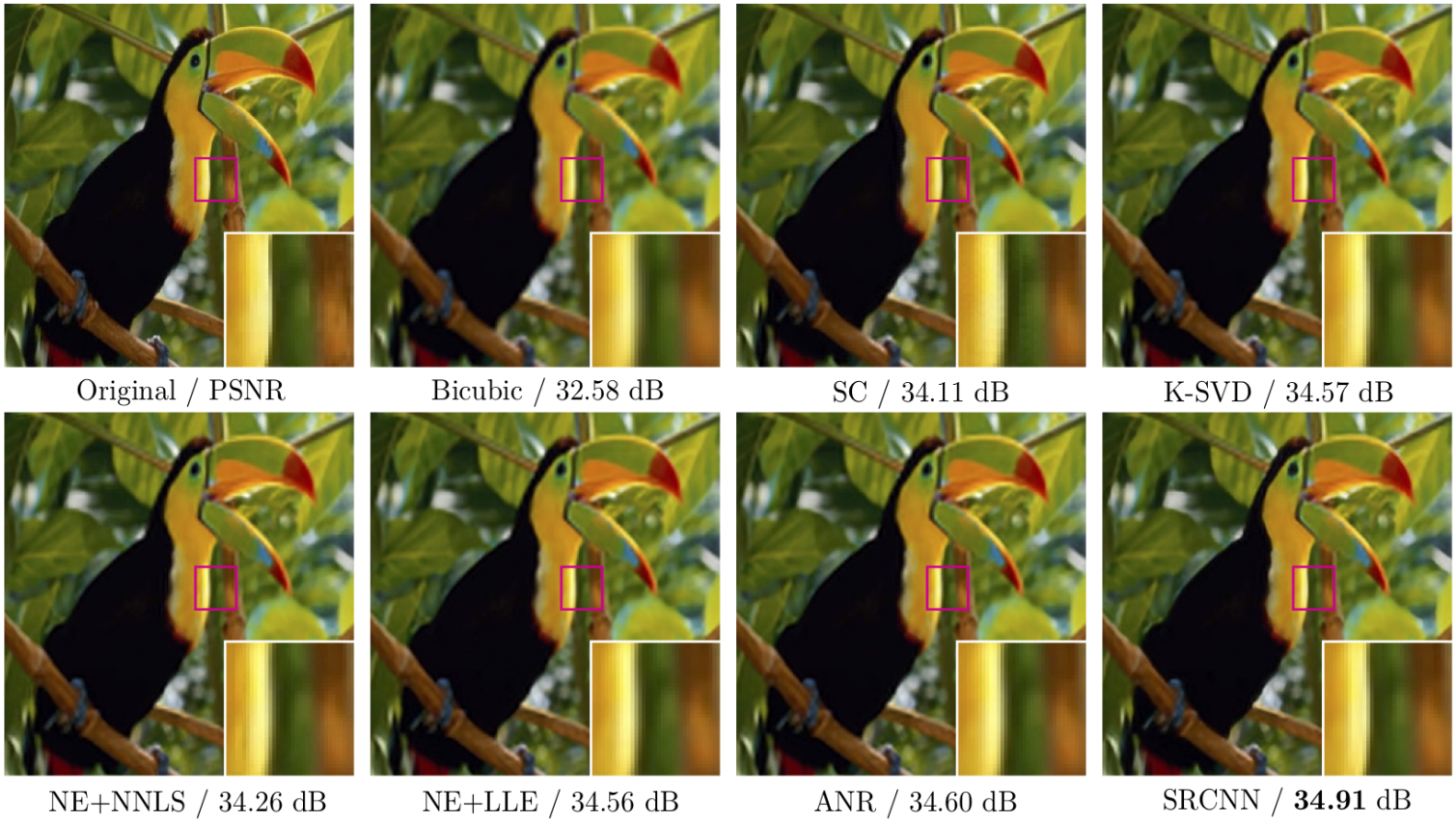


Fig. 9. "Bird" image from Set5 with an upscaling factor 3

11

Deep learning tasks go beyond classification. Can make use of vast data that is "corrupted" and learn the inverse process.

[11] https://link.springer.com/chapter/10.1007/978-3-319-10593-2_13

A few deep learning applications: style transfer

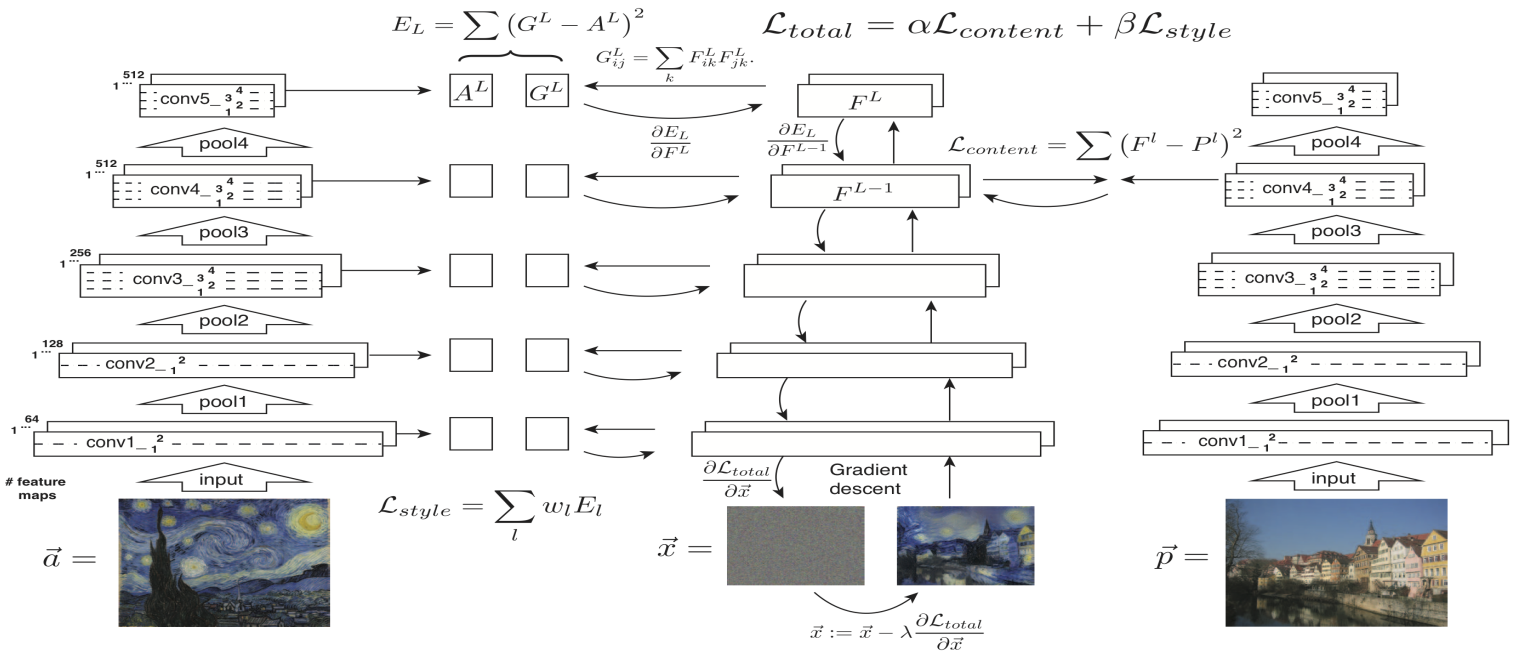


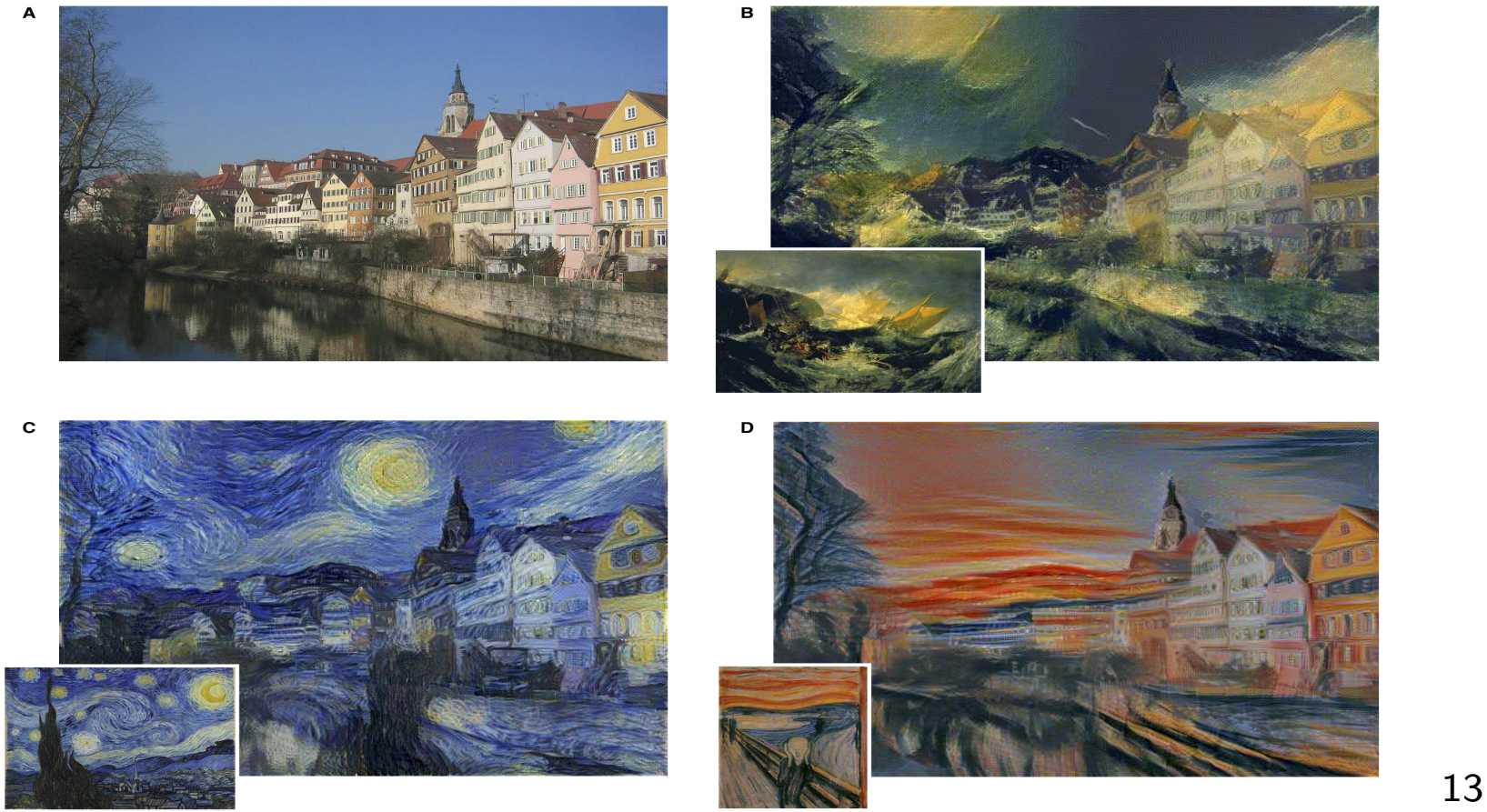
Figure 2. Style transfer algorithm. First content and style features are extracted and stored. The style image $\vec{a}$ is passed through the network and its style representation A^l on all layers included are computed and stored (left). The content image $\vec{p}$ is passed through the network and the content representation P^l in one layer is stored (right). Then a random white noise image $\vec{x}$ is passed through the network and its style features G^l and content features F^l are computed. On each layer included in the style representation, the element-wise mean squared difference between G^l and A^l is computed to give the style loss $\mathcal{L}_{style}$ (left). Also the mean squared difference between F^l and P^l is computed to give the content loss $\mathcal{L}_{content}$ (right). The total loss $\mathcal{L}_{total}$ is then a linear combination between the content and the style loss. Its derivative with respect to the pixel values can be computed using error back-propagation (middle). This gradient is used to iteratively update the image $\vec{x}$ until it simultaneously matches the style features of the style image $\vec{a}$ and the content features of the content image $\vec{p}$ (middle, bottom).

12

Deep learning tasks go beyond classification. Deep nets can be combined and loss functions blended.

[12] https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

A few deep learning applications: style transfer



13

Deep learning tasks go beyond classification. Deep nets can be combined and loss functions blended.

[13] https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

A few deep learning applications: DeepLens (CMU)

For many science applications the quantity of data is beyond human inspection, use DL: Large Synoptic Survey Telescope [14]

One approach to scale the visual inspection effort to the size of these surveys is to use crowdsourcing. This is the idea behind the Space Warps project (Marshall et al. 2015; More et al. 2015), which crowdsourced the visual inspection of a sample of 430 000 images from the CHFTLS to a crowd of 37 000 citizen scientists, yielding a new sample of gravitational lens candidates. The authors further estimate that a similar crowdsourcing effort can be scaled up to LSST sizes, where a considerable crowd of 10^6 *volunteers could visually inspect* 10^6 *LSST targets in a matter of weeks.*

[14] `https://academic.oup.com/mnras/article/473/3/3895/3930852`

A few deep learning applications: Go

Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

15

Deep learning tasks go beyond classification. Deep nets can be combined with tree search to play games.

[15] https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf

What are the ingredients making Deep Learning work:

Data:

- real data lives on a lower dimensional manifold,
- much of the variation we observe are invariance which should be in the net null space; e.g. translation, rotation, dilation,
- real data is compressible, what is the role of sparsity,

Architecture:

- how does the data inform the type of architecture to be used,
- what are the ingredients of modern deep nets, and why,
- to what, if any degree, are networks robust to adversaries,

Ability to train:

- how to train large numbers of networks parameters, θ,
- methods to train them must be scalable and are not run to global convergence,
- what are the impacts of choices in the optimisation method, such as batch size as an implicit regulariser.

Theories for Deep Learning:

- Expressivity of deep networks and the depth vs. width tradeoff?
- Convergence properties of stochastic gradient descent and other alternatives?
- What can we say about a single layer?
- Are there models of data for which we can guarantee activations are correct?
- What is the topology of the net parameters, θ,
- Are there variants of deep networks that don't require learning, easier to analyse
- Random matrix theory as a model if few network weights are changed

A few practicalities:

The course will be assessed through mini-projects analogous to that of the Networks (C5.4) course.

You will read recent conference proceedings / journal articles on theories for deep learning

You will write/modify code to perform experiments similar to those in the articles you read

You will write a report to discuss the topic you selected; details release in week 8 with the report due on Dec. 19th.

Tutorials: Fridays of weeks 3,5,7,8 will be lead by Dr. Abrol by Mr. Ughi. The tutorials are to aid you in how you should go about the assessment, and in so doing will introduce you to some practise of deep leaning and some details beyond lecture.