UNIVERSITY OF OXFORD
Mathematical Institute

Three ingredients of deep learning: LeNet-5, MNIST, and backprop.

Theories of Deep Learning: C6.5,
Lecture / Video1

Prof. Jared Tanner
Mathematical Institute
University of Oxford

Oxford
Mathematics

Three ingredients to deep learning

Architecture, data, and training

- Architecture: There are a number of network architectures, with the most classical being *fully connected* and *convolutional* networks which are composed of *layers with non-linear activations of affine transformations*. There are exponential gains in *expressivity with increased numbers of layers, depth.*
- *Data: sufficient amount of data to learn the isometries contained within the data, such as translation or rotation invariant in images, to learn context in text, to learn accents in speech, to learn styles of paintings, etc...*
- *Training Algorithms: Networks need to be trained, to learn* many parameters; *this requires proper initialisation, effective optimisation algorithms, and computational hardware.*

Example of a fully connected DNN:

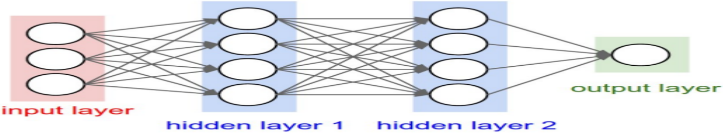
Two layer fully connected neural net

Repeated affine transformation followed by a nonlinear action:

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

where $W^{(i)} \in \mathbb{R}^{n_{i+1} \times n_i}$ and $b^{(i)} \in \mathbb{R}^{n_{i+1}}$ and $\phi(\cdot)$ is a nonlinear activation such as ReLU, $\phi(z) := max(0, z) = z_+$. The input is $h^{(1)}$, the output is $h^{(N)}$, and $h^{(i)}$ for intermediate $i = 2, \cdot, N-1$ are referred to as “hidden” layers.

The number of layers N is the depth, $N \gg 1$ is called “deep.”



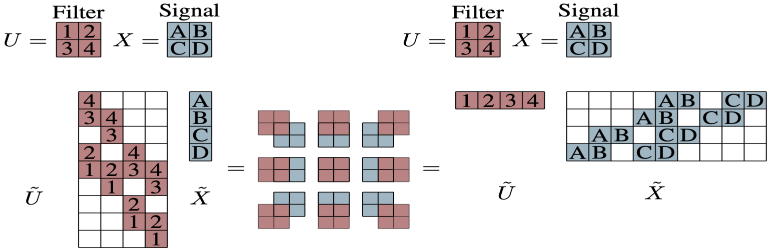
https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/feedforward.html

Convolutional neural net (CNN):

Convolutional nets use local information

Convolutional neural network use local information from the input which also imposes structure on the weight matrix $W^{(i)}$.

In its simplest form, consider a two-dimensional input X and a single filter U, we can view the action of convolving the filter with the zero-padded version of X with stride (shift) one as:



1	2
3	4

A	B
C	D

4			
3	4		
	3		
2		4	
1	2	3	4
	1		3
		2	
		1	2
			1

A
B
C
D

1	2	3	4

				A	B		C	D
			A	B		C	D	
	A	B		C	D			
A	B		C	D				

Image complements of Thziri Nait Saada.

Convolutional neural net (CNN):

Multiple input and output channels

The input to a CNN, and the hidden layers, are typically tensor; say $h^{(l)} \in \mathbb{R}^{m \times n \times c_l}$. Defining a filter spatial two-dimensional size $n_x \times n_y$ and stride s, we can then enumerate the locations where the convolution will occur by $k = 1, \cdot, ..., p$.

The p patches from $h^{(l)}$ to be acted on can be extracted and their entries stored in vectors; say $h_k^{(l)} \in \mathbb{R}^{n_x \cdot n_y \cdot c_l}$. Then form the matrix $\tilde{h}^{(l)} = \left(h_1^{(l)} \cdots h_p^{(l)}\right) \in \mathbb{R}^{n_x \cdot n_y \cdot c_l \times p}$.

The action of the convolution can be computed as $W^{(l)} \tilde{h}^{(l)}$ where the c_{l+1} rows of $W^{(l)}$ are its filters and the data tensor $h^{(l+1)}$ then computed by re-ordering the p locations as two-dimensional locations.

LeNET-5, an early Image processing DNN:

Network architectures often include fully connected and convolutional layers

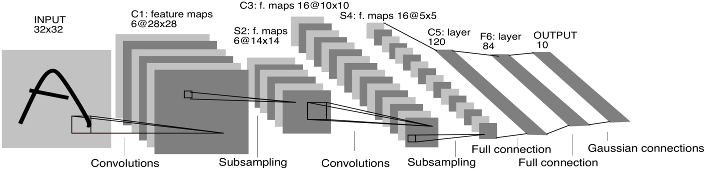


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

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.

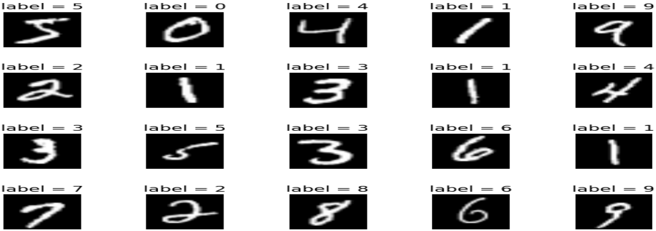
C5: 120 features, 5 by 5 support, no stride; i.e. fully connected.

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

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

Dataset MNIST, simple exemplar dataset for classification:

MNIST collection of 70,000 labeled digitised gray scale hand written digits 0 to 9



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

Vectorising each 28×28 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 $\ell \neq 5$.

http://yann.lecun.com/exdb/mnist/

Data characteristics and isometries:

DNNs are learned functions for specific data types

- DNN parameters $\theta := \left\{W^{(i)}, b^{(i)}\right\}_{i=1}^{N}$ are learned so that the DNN is the desired function from the data input ambient dimension to the task output dimension.
- Data generally has high correlations such as local smoothness in images, low-rank in consumer data, or phonemes for audio data. Though the data lives in a high-dimensional ambient dimension, the correlation typically causes the data to be approximately low dimensional; e.g. each MNIST digit class is contained on a locally less than 15 dimensional space.
- For classification tasks, much of the variation we observe are invariants which should be in the nullspace of the DNN; e.g. translation and rotation. Classifiers can be viewed largely as nullspace maps.

Data set size and complexity

The quantity of and richness of datasets has been central to the use of DNNs

http://image-net.org

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). In 2010 ImageNet included more than 1.2 million labeled images with each image class having approximately 1000 examples.

Datasets as random realisations

Datasets are

- The *training data* used to learn the network parmaters $\theta := \left\{W^{(i)}, b^{(i)}\right\}_{i=1}^{N}$ is only a small set of the data of the data class it is used to represent.
- The value of the DNN is its ability to *generalize* to unseen *testing data*. The ability to generalize demonstrates that the DNN is approximating the data manifold beyond those seen data, and has learned invariants which are considered to be unimportant for the task.
- Typically a dataset is "randomly" partitioned into disjoint training and testing sets. This and many other aspects of training a DNN introduce randomness in the DNN obtained and its performance; for these reasons DNNs are not typically trained to zero training accuracy.

Training loss function:

Training is formulated as optimizing a loss function

The network "Weights" $W^{(i)}$ (and biases $b^{(i)}$) are learnt to fit a task for a particular data set.

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

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

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

where $\theta := \left\{W^{(i)}, b^{(i)}\right\}_{i=1}^{N}$and the resulting learned net is, $H(\cdot;\theta)$. The learned network depends on the choice of function $L(\cdot,\cdot;\theta)$.

Optimisation algorithms:

Loss function minimised through gradient descent

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

- Back-propagation is key advancement, giving an efficient way to compute the gradient of the training loss function, $\nabla_\theta L(X, Y; \theta)$, and θ is updated along the gradient

  $$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla_\theta L(X, Y; \theta^{(t)})$$

  with a learning rate η_t that determines how far to move at each iteration.

- Stochastic gradient descent (SGD), and advanced variants, such as Adagrad and Adam, have reduced the computational time and ability to train large networks; typically using only a subset of the training data at each iteration.

Optimisation landscapes and initialisations:

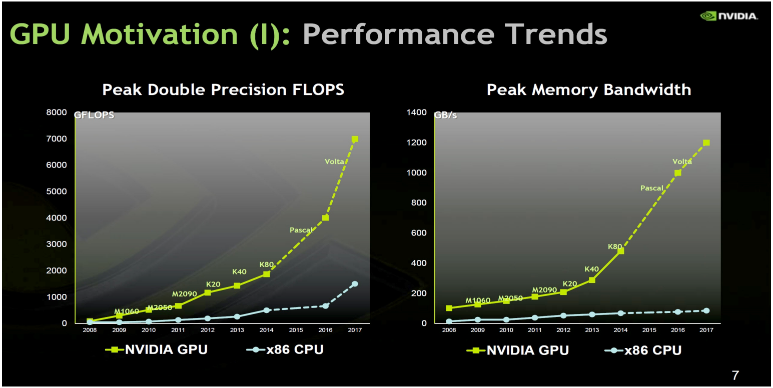
The training objective dependence

DNNs are applied to increasingly challenging tasks, which have required increased depth.

- The network hyperparameters, width and depth, as well as training loss function impact the shape of the *typically non-convex landscape* to be minimised. Non-convex landscapes often have many local minima which can result in wide variation in learned networks.
- In some cases, network parameters or architectures can be proven to result in *convex* landscapes which are inherently easier to train.
- Unless network weight initialisation is chosen appropriately, deep networks can suffer from gradients that either diverge or converge to zero with depth (vanishing or exploding).

Training of DNNs aided by hardware advances:

Hardware is now specifically designed to training DNNs



GPU Motivation (I): Performance Trends

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

Returning to MNIST classification: training

Exemplifying the three ingredients of deep learning: training loss function

LeCun et. al. considered, amongst others, a 2 layer fully connected net with architecture $W^{(1)} \in \mathbb{R}^{100\times 728}$ and $W^{(2)} \in \mathbb{R}^{10\times 100}$ and sigmoid activation $\phi(z) = \frac{1}{1+e^{-z}}$, applied to MNIST classification using the sum of squares training loss objective:

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

Backprop was introduced in this article, and gradient descent applied. The MNIST dataset is partitioned into 50,000 training images and 10,000 test images, each in $\mathbb{R}^{728}$.

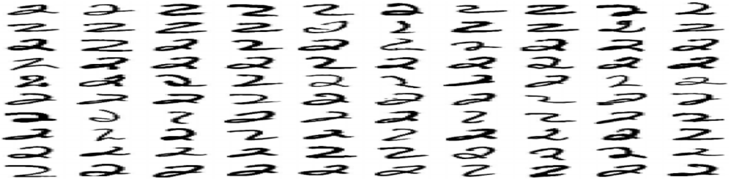
The above two layer net has $73,910$ parameters and achieved a 4.7% classification error rate on the test set.

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

Returning to MNIST classification: data

Exemplifying the three ingredients of deep learning: data augmentation

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 $H(x;\theta)$ trained on 50,000 examples achieves 4.7% error rate; augmenting the data by including artificially distorted examples decreased the error to 2.5%.

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

Returning to LeNet-5 for MNIST classification: architecture

Exemplifying the three ingredients of deep learning: architecture

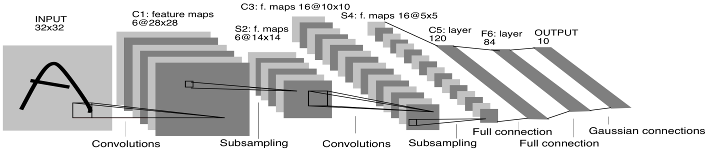


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

C1: conv. layer with 6 feature maps, 5 by 5 support, stride 1.
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, $W \in \mathbb{R}^{84\times 120}$.

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

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

Examples of topics in this course:

Example course content

- Approximation theory perspectives, characteristics of data, and generalisation error.
- Role of initialisation on information flow and ability to train very DNNs.
- Training landscape: non-convex and convexification.
- Optimisation algorithms used for training DNNs.
- Interpretability and structure in DNN weights
- Robustness of DNNs and adversarial perturbations.
- More recent architectures and advances.

Course practicalities:

Reading, tutorials, and assessment

- We will not be following a textbook, but you might find Deep Learning by Goodfellow, Bengio, and Courville useful: `http://www.deeplearningbook.org`
- Lectures / videos will typically include links to conference proceedings or journal articles. You are encouraged to read some but not all of these; this week read: lecun-98.pdf. See the weekly announcements.
- Tutorials will include some pen and paper questions, along with computational experiments. This course is focused on theoretical aspects of deep learning. Computational experiments help explore the theory, and are a valuable skill.
- The course is assessed entirely by individual mini-projects. You will have significant scope to select a topic of interest to you, but there are some restrictions; details to follow.

Course practicalities:

Mini-project assessment for OMMS and Part C 1 of 3

Select one of the following general areas and write a report on a specific subtopic within one of the areas:

1. List to be released later in the term, tentatively "Friday of MT Week 8, due by 12pm Wednesday of HT Week -1."

Your report must include a discussion of some theoretical portion of deep learning along with numerical simulations selected to highlight the issue being investigated and a discussion of how the numerical simulations reflect on the issue. Your numerical simulations should be novel in some regard; e.g. data set, architecture parameter choices, training algorithm, etc... Ensure that your report has as its focus a theoretical issue rather than an application of deep learning.

Course practicalities:

Mini-project assessment for OMMS and Part C 2 of 3

Deep learning is a rich topic under intense investigation. Two of the main venues for original research in this topic are the Neural Information Processing Systems (NeurIPS) conference and the International Conference on Machine Learning (ICML). Proceedings for these conferences are available at `https://papers.nips.cc` for NeurIPS and `http://proceedings.mlr.press` for ICML (Volume 97 for ICML 2019) as well as other conferences on machine learning. You are encouraged to select a topic and starting point for your report by perusing these proceedings papers and selecting an article to your liking.

Course practicalities:

Mini-project assessment for OMMS and Part C 3 of 3

Your report should be in the format and style of a NeurIPS Proceedings, abridged to not exceed 5 pages of text and graphics and 1 page of references (for a total length of not more than 6 pages). LaTeX style files and an example template are provided on the course page. Clearly indicate any ideas in your report which are your own and give appropriate attributions for all references (including research articles, software, data sets, etc....). Your report need not contain original research results, though you must use some original research articles as references (not just review articles or books). You should include a high level description of the code used to generate your numerical simulations, but should not submit the entire code; description of the code should not exceed one page of the report.