UNIVERSITY OF
OXFORD

Mathematical
Institute

A few things we missed and summary of theories of deep learning material covered

Theories of Deep Learning: C6.5,
Lecture / Video 15
Prof. Jared Tanner
Mathematical Institute
University of Oxford

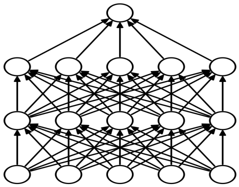
Oxford
Mathematics

A bit more Deep Learning you might like to know: dropout, resnets, attention, transformers, sparsifying nets, and we end with a summary.

Dropout (Srivastava et al. 14')

Setting hidden layer entries to zero at random

Dropout is a method by which, during training, the activations are set to zero with some probability. Note, dropout is only used in the training phase, not in testing.


(a) Standard Neural Net

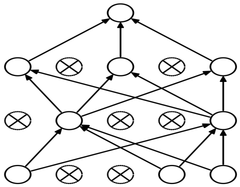

(b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Dropout has a number of valuable consequences: reducing correlation in training, inducing sparsity, avoiding overfitting, and can be used to evaluate uncertainty in a deep net.

`http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf`

Dropout (Srivastava et al. 14')

Improved test error

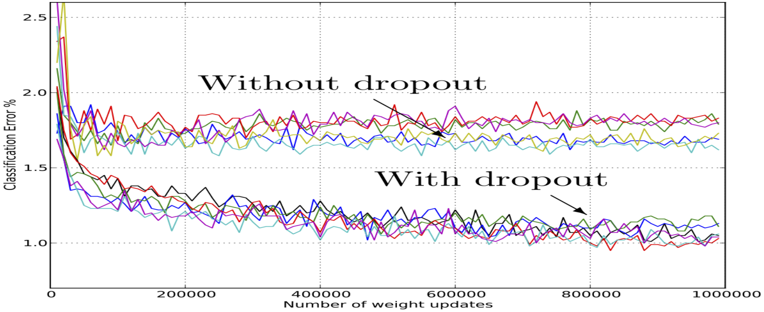
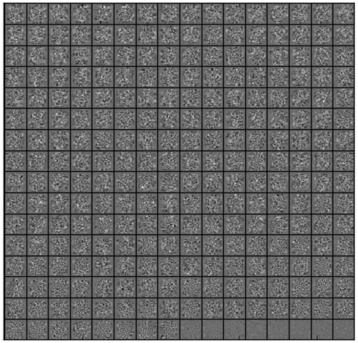


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

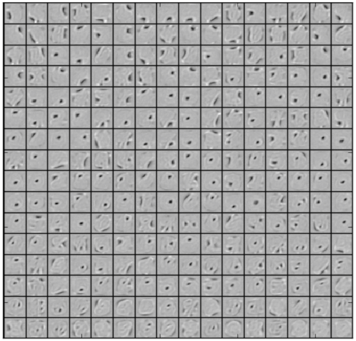
http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Dropout (Srivastava et al. 14')

Sparse features



(a) Without dropout



(b) Dropout with $p = 0.5$.

Figure 7: Features learned on MNIST with one hidden layer autoencoders having 256 rectified linear units.

`http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf`

Residual Networks (He 15')

The skip connection building block

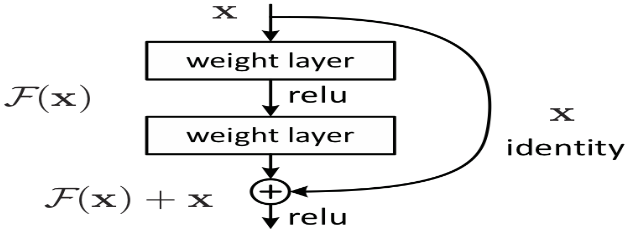


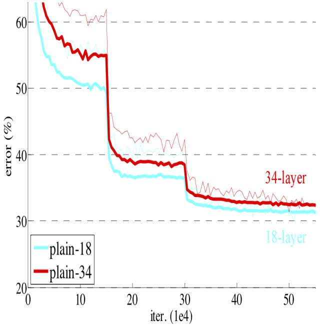
Figure 2. Residual learning: a building block.

If the block is attempting to learn a map $\mathcal{H}(x)$ the ResNet instead attempts to learn $\mathcal{F}(x) := \mathcal{H}(x) - x$ which is the residual. One can speculate this is easier to learn if $H(x)$ is approximately an identity map.

`https://arxiv.org/pdf/1512.03385.pdf`

Residual Networks (He 15')

Impact on training loss function vs without skip connections



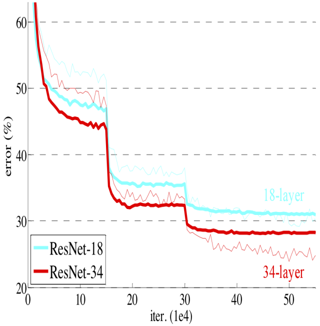


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

`https://arxiv.org/pdf/1512.03385.pdf`

Residual Networks (He 15')

Examples of ResNet architectures in more detail

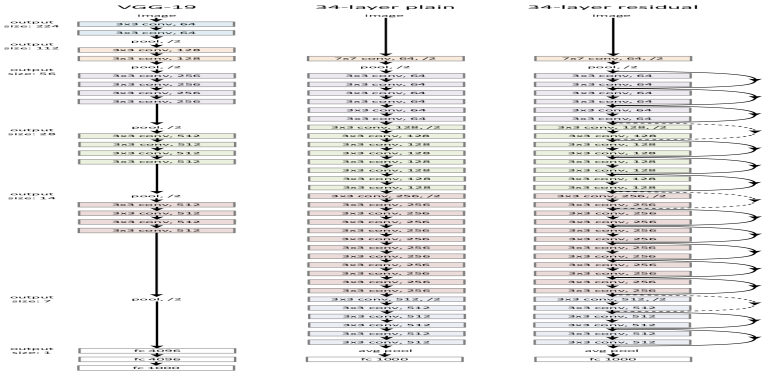


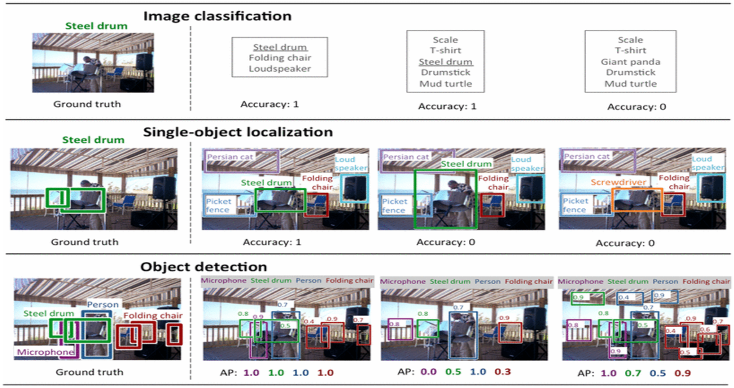
Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

https://arxiv.org/pdf/1512.03385.pdf

Recall ILSVRC

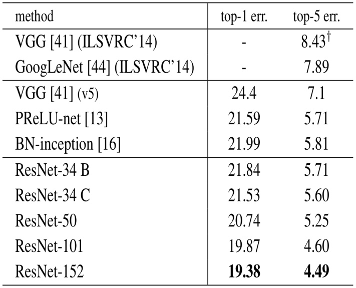
ImageNet Large Scale Visual Recognition Challenge, Russakovsky et al. 2015.

Mathematical Institute



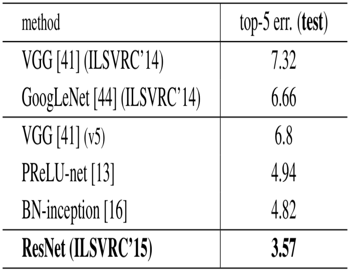
Residual Networks (He 15')

Improved accuracy for ILSVRC



method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).



method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

See also Highway nets by Srivastava et al. 15 which vary the amount of x being passed through a skip connection.

`https://arxiv.org/pdf/1505.00387.pdf`

Natural Language Processing

Tokenization and word embeddings

Natural language processing (NLP) involves decomposing text into a finite set of tokens.

"Tokens can be thought of as pieces of words. Before the API processes the request, the input is broken down into tokens. These tokens are not cut up exactly where the words start or end - tokens can include trailing spaces and even sub-words. Here are some helpful rules of thumb for understanding tokens in terms of lengths:"

1 token = 4 chars in English

1 token = 3/4 words

100 tokens = 75 words"

`https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them`

GPT3 architecture (Brown 20')

GPT: Generative Pre-Trained Transformers, dimensions

Each token is then embedded in a word embedding dimension and sequentially each token in the context window is embedded as rows in a matrix $X \in \mathbb{R}^{d_{ctx} \times d_e mb}$.
NLP networks can have tasks such as "next token prediction" where text is grown, or "translation" from one language or sentiment to another, or combined in multi-modal systems with other networks (e.g. vision) for such tasks as automatic caption generation.
By far the dominant method for NPL are attention / transformer modules.
`https://arxiv.org/abs/2005.14165`

Attention mechanism (Vaswani 17'), equations

Key and Query quadratic form to highlight relations

Patches of inputs of a fixed dimension are extracted and arranged into a matrix X, similar to CNNs. They queries, keys, and values are then computed with matrix-products $Q^T = W_Q X^T$, $K^T = W_K X^T$, and $V^T = W_V X^T$ then the re-activation attention layer is

$$H = \phi_1 \left(QK^T d^{-\alpha} \right) V$$

and another nonlinear activation, $\phi_2(\cdot)$ is then applied to either H or maybe with a skip to $H + X$. Typically $\phi_1(\cdot)$ is softmax and ϕ_2 is ReLU or similar. The scaling α is typically $1/2$, but also sometimes 1, and generally Q and V have layer-norm applied to enforce fixed mean and variance. Multi-head attention concatenates many of these and multiply by a weight matrix.

https://arxiv.org/abs/1706.03762

https://arxiv.org/abs/1607.06450

Attention mechanism (Vaswani 17'), diagram

Key and Query quadratic form to highlight relations

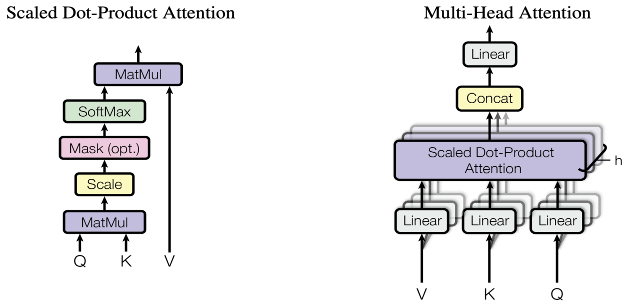


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

https://arxiv.org/abs/1706.03762

Transformer mechanism (Vaswani 17'), diagram

Encoder-decoder structure using attention

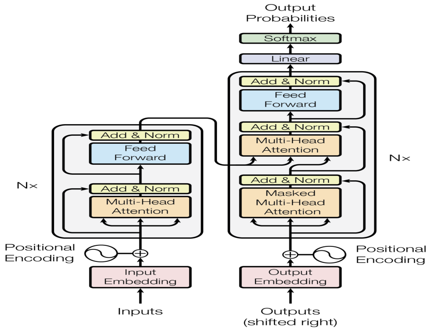


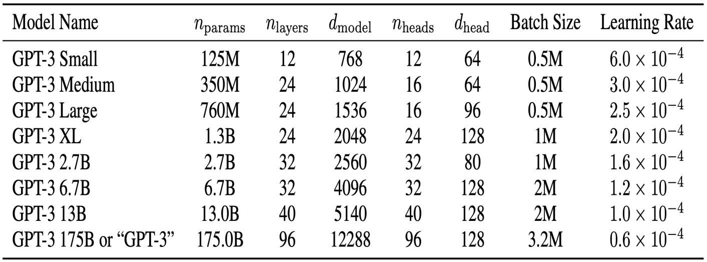
Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

https://arxiv.org/abs/1706.03762

GPT3 architecture (Brown 20'), diagram

GPT: Generative Pre-Trained Transformers



Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Each layer of GPT3 (and earlier variants) is a transformer block.
Energy usage to train and apply such large models is very costly.
`https://arxiv.org/abs/2005.14165`

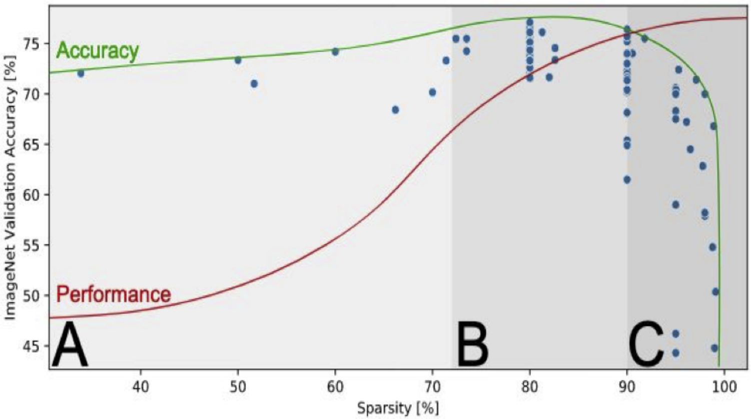
Sparsifying deep networks

Severe overparameterisation and scale of modern nets

- Deep nets for some state of the art tasks have vast numbers of trainable parameters:
  - ResNet101, image classification - 45 million parameters
  - GPT-3, text generation - 175 billion parameters
  - T5-XXL, language model - 1.6 trillion parameters
- An approximation theory viewpoint suggest this isn't necessary at inference, see Optimal Approximation with Sparsely Connected Deep Neural Networks, Bolcskei et al. 2019. `https://arxiv.org/abs/1705.01714`
- Practise tell us the number of parameters can be reduced to 5% or fewer parameters without loss of accuracy.

Sparsifying and the lottery ticket hypothesis

Reducing the number of parameters initially improves accuracy



https://arxiv.org/pdf/2102.00554.pdf

How to create sparse networks

At initialisation and dynamic pruning

- Starting with a sparse network: pruning at initialisation:
  - From any network, random or trained, determine a measure of importance for a parameter in the network and set the parameter to zero if below that threshold.
  - Examples include: magnitude of the parameters, gradient of the loss with respect to that parameter, or measure of information flow.
- Dynamic sparsifying the network:
  - Prune as above, but allow some entries to be reintroduced and then pruned again during training.
  - Most successful is to prune based on magnitude and re-introduce based on training gradient magnitude.
- This is an increasingly active area due to the every growing size of networks.

Summary of the material covered.

Lecture 1: Three ingredients to deep learning

Architecture, data, and training

- Structure of a deep net as repeated affine transforms and non-linear activations.
- Introduction to LeNet-5 with convolutional and fully connected layers.
- MNIST as an example of small dataset, along with the more complex imagenet dataset.
- Discussion of availability of computational resources and optimisation algorithms.

Lecture 2-4: Exponential expressivity with depth

Why depth from a function approximation perspective

- Telgarsky 15' sawtooth function giving example function with exponentially many maxima as a function of depth, but linear in width.
- Yarotsky 16' extension of the sawtooth to generate local polynomial approximations within ϵ needing $\log(1/\epsilon)$ depth.
- Poggio et al. 17' tree structure for approximation rate at the low effective dimensionality.
- Hein et al. 05' showing that MNIST, for example, has low effective dimension.

Lecture 5-6: Randomly initialized DNN exponential behaviour

Controlling the exponential with depth behaviour: correlation and gradients

- Glorot et al. 10' observation of pre-activation hidden layers being approximately Gaussian and normalizing to have constant variance with depth.
- Poole et al. 16' quantifying the convergence of the Gaussian variance through a computable recursion relation and developing a theory for the correlation between inputs dependence on $(\sigma_w, \sigma_b, \phi(\cdot))$. Controlling geometric collapse or instability through selecting $(\sigma_w, \sigma_b, \phi(\cdot))$ according to derived formulae.
- Pennington et al. 18' introduced the edge of chaos and connecting the Poole et al. work with exploding and vanishing gradients. Random matrix theory to derive moments of the spectra of the DNN.

Lecture 7-8: Optimisation algorithms for DNNs

Character of the loss landscape and algorithms to minimize it

University of Oxford Mathematical Institute

- Foundational theory on stochastic gradient descent (SGD), making use of back-propagation and mini-batches to improve scalability of the algorithms.
- Reducing the gap between the value of a global minimizer and SGD through decreasing stepsize, increasing batchsize, or other variance reduction methods.
- Accelerating through momentum and diagonal scaling.
- Ward et al. AdaGrad scalar diagonal scaling to have reliable training over a wide range of stepsize initializations.

Lecture 9-10: Loss landscape properties and improvements

Simply connected landscapes and architecture choices to aid them

- Venturi et al. 16' showed the minimizers of the loss landscape can have paths between them where the loss landscape remains of a similar value, nearly simply connected.
- Pennington et al. 17' showed how the distribution of eigenvalues of the loss landscape's Hessian can be computed as a function of the number of trainable parameters compared to the amount of data available. With enough data, and when close to a minimizer, then the landscape is locally convex.
- Li et al. 18' illustrated how the loss landscape width near minimizers is impacted by training batch-size, and how ResNets can improve the landscape characteristics.
- Loffe 15' introduced batchnorm to have trainable bulk scaling to aid optimisation.

Lecture 11-12: Visualizing and understanding CNN filters

CNN filters learned on natural images are interpretable and predictable

Mathematical Institute

- Filters for early layers of CNNs show characteristic high-dimensional wavelet like structure.
- Deeper layers combine such filters and give greatest response for more structured inputs, leading to "memory" where some units in the CNN are maximized by objects within training classes.
- Such structure helps explain the efficacy of transfer learning.
- Mallat 12' introduced the Scattering Transform, which is a deep transform that only learns the final layers; activations are hand crafted to encourage desired invariants such as translation.

Lecture 13-15: Generative models and lack of robustness 1 of 2

How DNNs can be reversed to generate data

- Goodfellow et al. 14' introduced the generative adversarial network (GAN) structure using reversed DNNs to generate data characteristic of a training dataset.
- Bau et al. 20' Filters analogous to those in CNNs on natural images are learned for generating natural images. They can be modified in order to influence expected properties of the generated data, such as colour or frequency of objects such as trees or doors in buildings.
- Moosavi-Dezfooli et al. 16' introduced DeepFool to effectively generate adversarial misclassification.

Lecture 13-15: Generative models and lack of robustness 2 of 2

How DNNs can be reversed to generate data

- Engstrom et al. 18' showed that natural actions on objects, such as rotation or translation can also be used to generate misclassification; showing inherent lack of robustness in typical DNNs.
- Wong et al. 17' demonstrated that one can adapt the training to have certificates which ensure that the network is provably robust in some circumstances.
- Gopalakrishnan et al 18' showed how sparsifying a DNN can improve robustness, explainable by reducing an upper bound on the DNN's Lipshitz constant.
- Autoencoders and VAEs as alternative architectures that can be built from DNNs for tasks such as denoising or explainable generative data.

Major omissions and challenges

Generalization error, benefit of depth

- Exponential approximation ability with depth is well understood.
- Algorithms exist to effectively train DNNs and their variants, overcoming disadvantages that accompany the exponential nature of depth.
- DNNs are observed to generalize well, but theory is lacking to show a benefit of depth in terms of generalization; such a result may not exist, or may require a DNN that is regularized to show that with a fixed amount of trainable parameters, that depth has generalization benefits over width.
- This leads us to consider the amount of trainable parameters needed; width and depth are needed, but generate undesirably large amounts of trainable parameters.