Neural Networks

Jan Kościałkowski

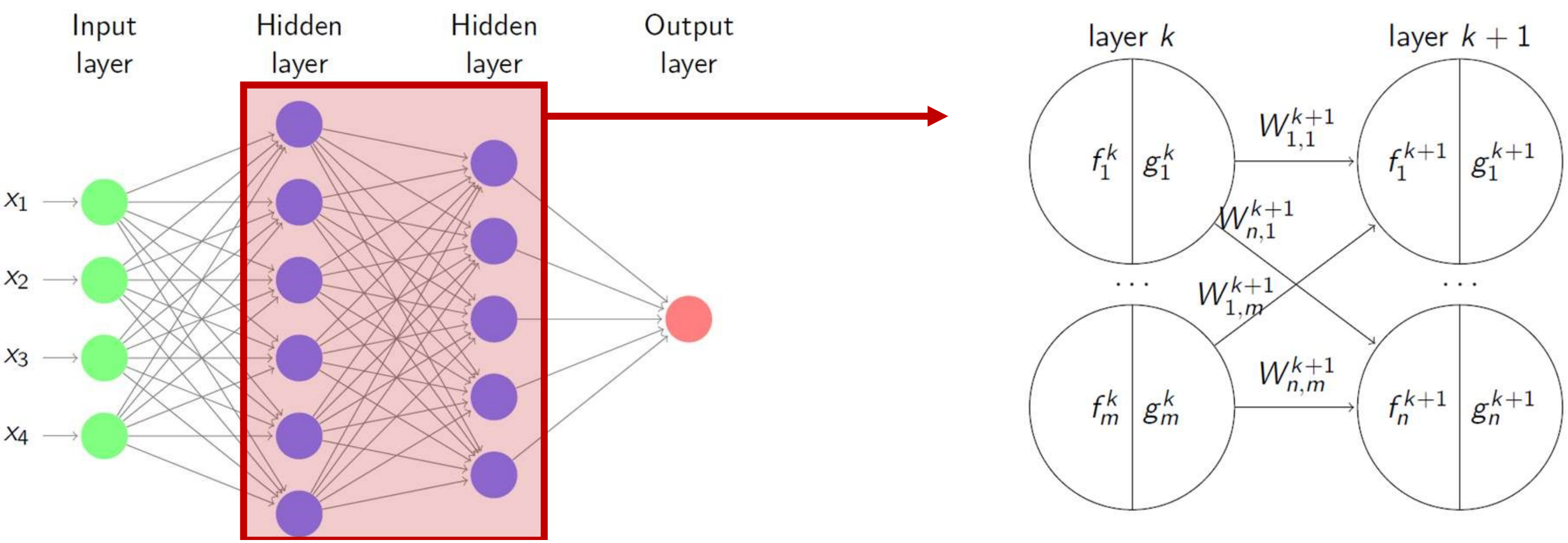
18 May 2020

Idea: approximate some function

- Examples: regression function, conditional probability
- Can be much more complicated
- Define some loss function (discrepancy between true values and predictions) and try to minimize it
- Example: number of claims, Poisson loss, $\mathrm{y} = \exp(f(x_1, \dots, x_p))$

Multi-Layer Perceptron (MLP)

- Nodes – apply activation function
- Edges – matrix multiplication
- Universal Approximation Theorem (Cybenko 1989):
  Approximates any continuous function on a compact subset of $\mathbb{R}^d$ arbitrarily well

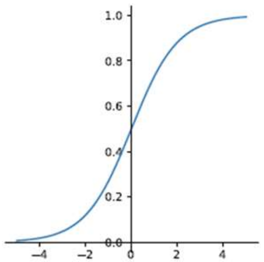


$$f^k = W^k g^{k-1}$$

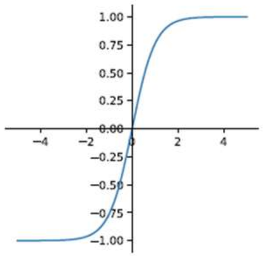
$$g^k = \sigma(f^k)$$

Activation Functions

Issue: saturation

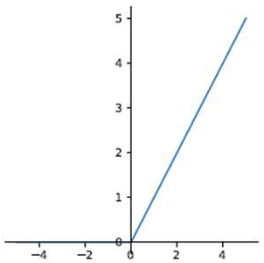


sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$

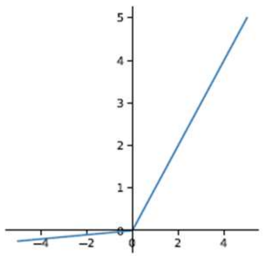


tanh: $\tanh(x) = \frac{e^x+e^{-x}}{e^x+e^{-x}}$

Issue: dead ReLU



ReLU: $\mathrm{ReLU}(x) = \max(0, x)$



leaky ReLU: $\mathrm{LReLU}(x) = \max(ax, x)$

Mini-batch Learning

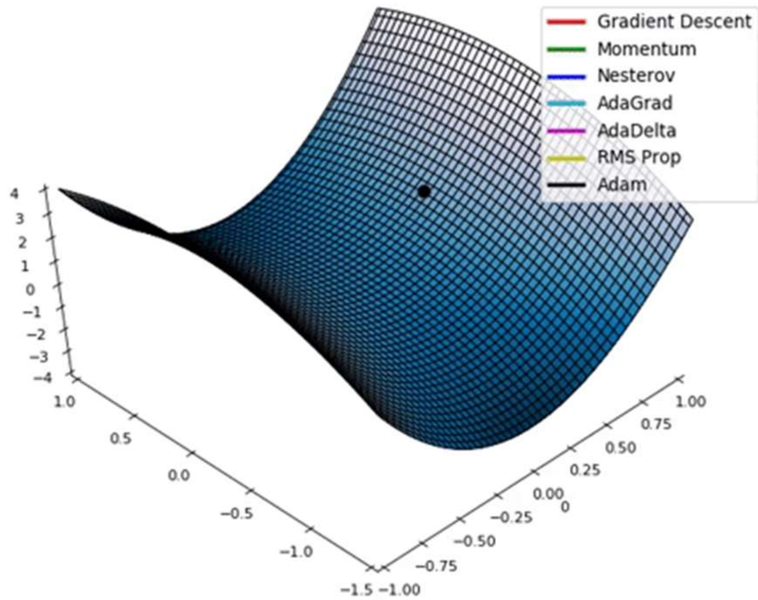
https://www.researchgate.net/figure/Example-of-mini-batch-learning-on-CNN_fig41_339054847

Optimization

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

Neural networks are optimized using (mini-batch) SGD and its variants:

- Momentum – exponential moving average of past gradients
- Learning rate decay – decrease learning rate e.g. according to an exponential schedule
- Adagrad – for each weight w decay learning rate by $\sqrt{G}$ where G is the sum of squares of past gradients for w
- RMSProp – as in Adagrad, but G is exponential moving average of G and squared gradients
- Adam – combines EMA of squared gradients and EMA of gradients, usually the easiest to use

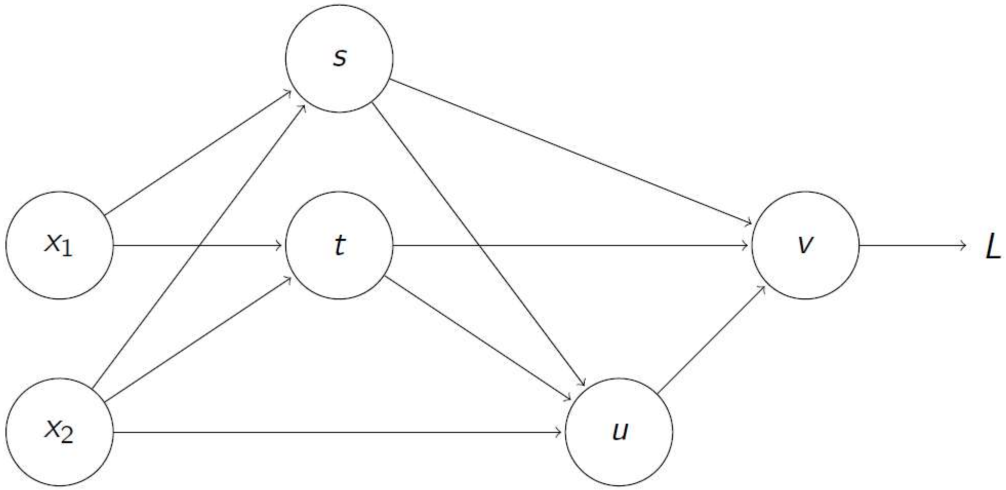


https://rnrahman.com/blog/visualising-stochastic-optimisers/

Backpropagation

- We need derivatives of loss function w.r.t. weights for SGD
- Use multi-dimensional chain rule to decompose derivative computations into products of step-by-step derivatives
- In general, works for DAGs of differentiable operations (including MLP)
- Idea – for each mini-batch do the following:
    - Forward pass: run the input data through the network and compute all intermediate values
    - Compute the value of the loss function
    - Compute the single-step derivatives going backwards from the loss function
    - Multiply them to get the desired derivatives

$$\frac{\partial L}{\partial z} = \sum_{z=z_0\to\ldots\to z_l=L} \prod_{i=1}^{l-1} \frac{\partial z_{i+1}}{\partial z_i}.$$

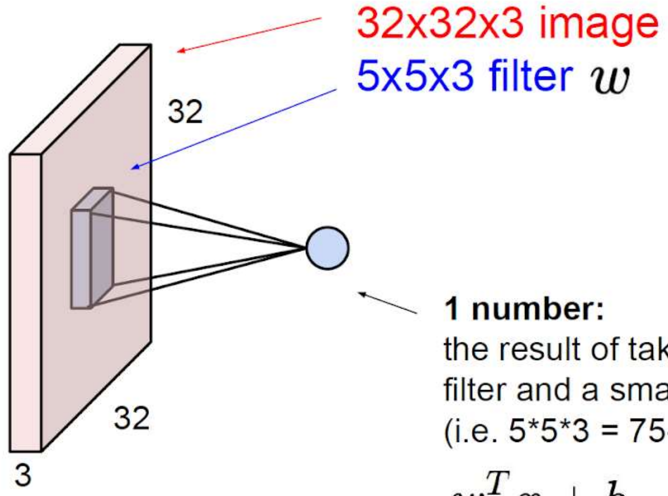


Regularization

NNs are usually too complex for tabular data, heavy regularization needed:

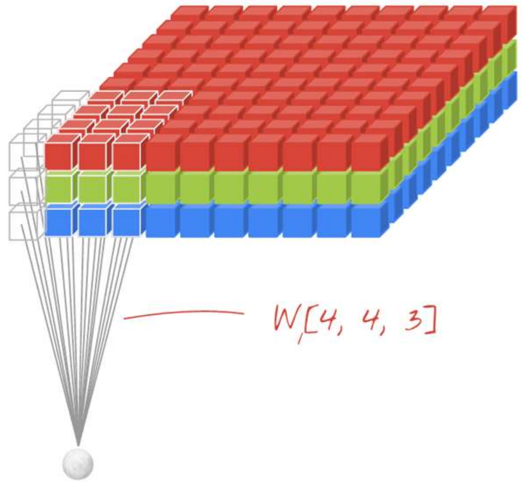
- L1, L2
- Data augmentation – perturb the real data (add noise, offset, distort etc.)
- Dropout – randomly omit network edges during training
- Batch, layer normalization – normalize across different dimensions of the batch to try to make the distributions more normal
- Early stopping – monitor validation set metrics and stop training when they stop improving

Convolutional Networks



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

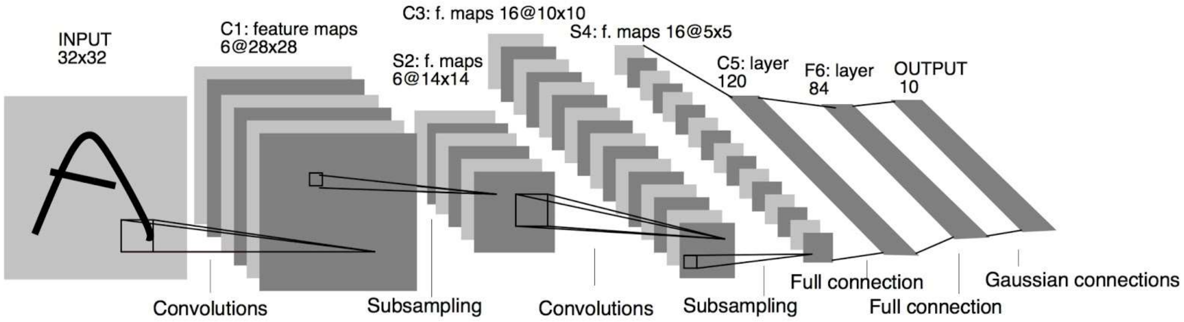


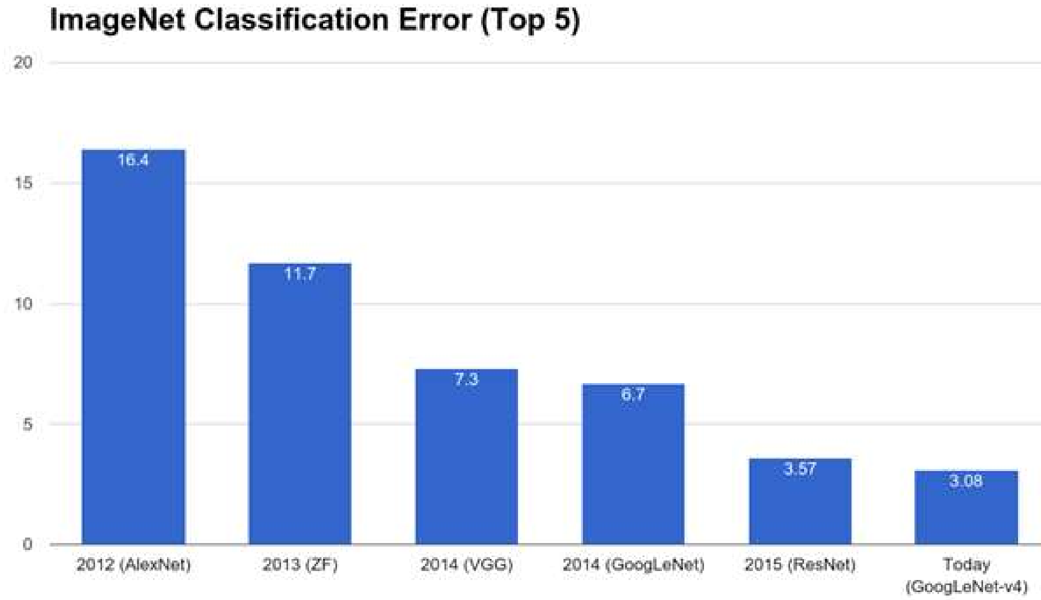
https://cs231n.github.io/convolutional-networks/

https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#10

Convolutional Networks: History

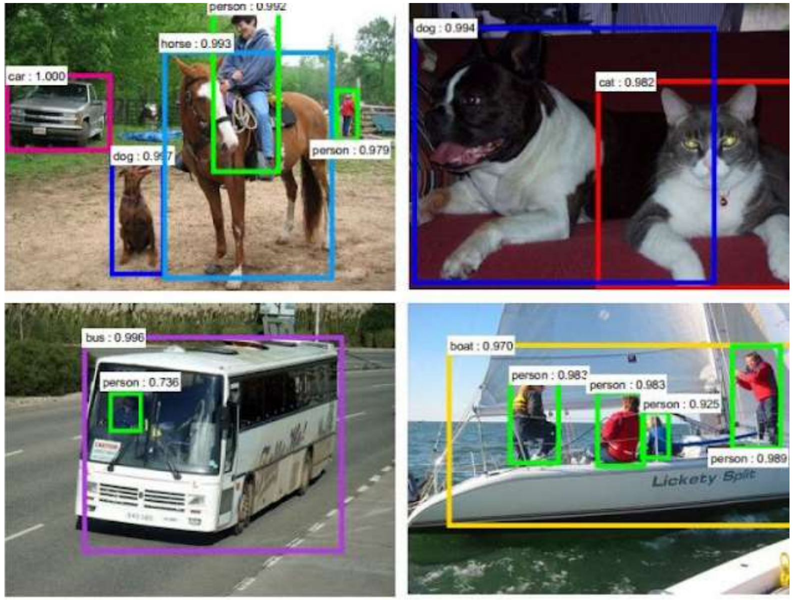
- Around since 1980s
- LeNet-5 (LeCun et al. 1998) – handwirtten digit recognition
- Due to GPU utilization, massive advances since 2012 + switch from SVMs to NNs





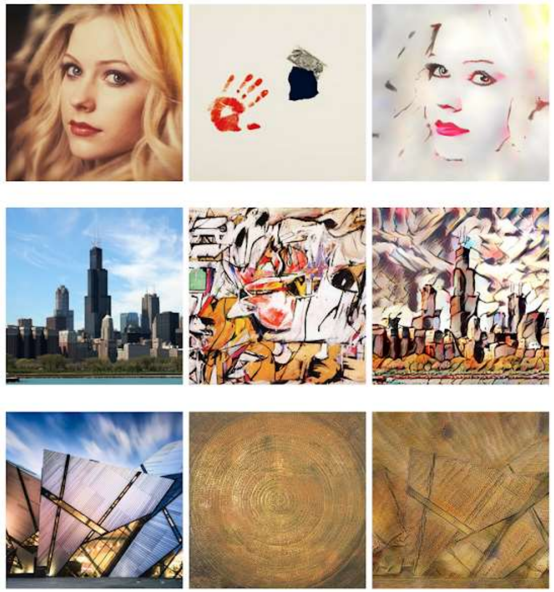
Convolutional Networks: Use Cases

Detection



Segmentation

Style Transfer



https://pjreddie.com/darknet/yolo/

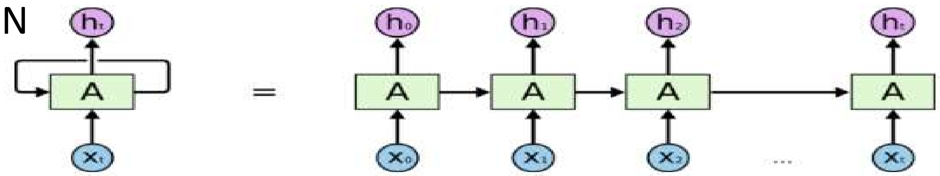
https://github.com/xunhuang1995/AdaIN-style

Recurrent Networks

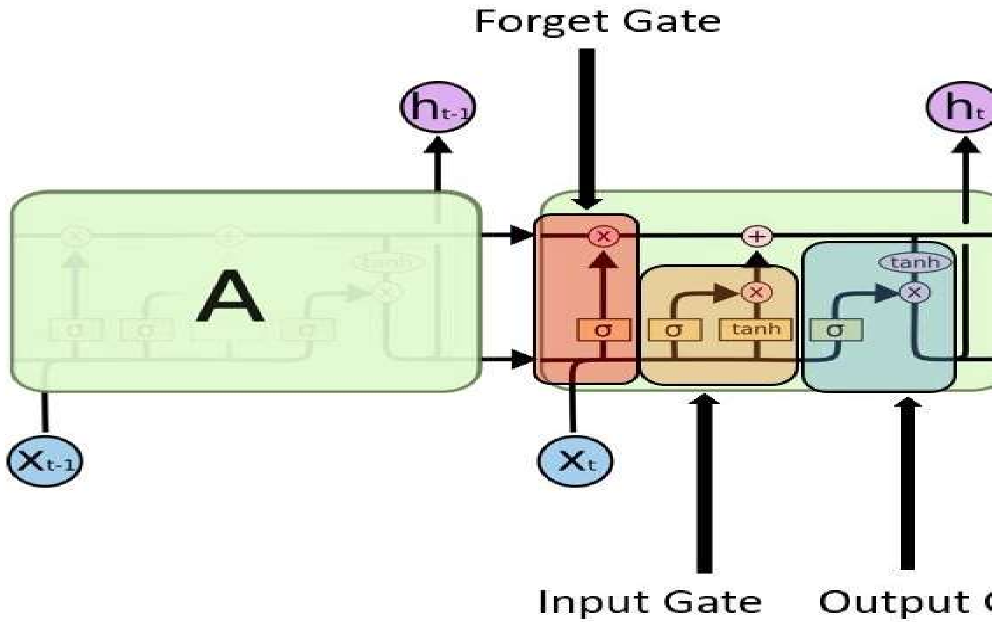
Used for modeling sequential data: text, voice, time series etc.

Operations exactly the same as for MLP, but uses the concept of weight sharing and backpropagation through time.

RNN



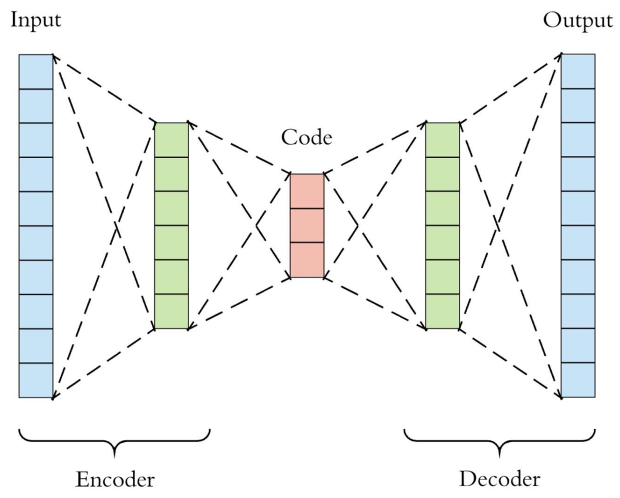
LSTM



https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e

Representation Learning

- NNs very good at learning well-performing low-dimensional representations of complex objects (words, images etc.)
- Usually works by trying to reproduce the initial object or the context, the representations are taken to be outputs from some intermediate layer
- Example: autoencoders



https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

Other Important Classes

- Generative Adversarial Networks (generating fake photos/videos)
- Reinforcement Learning (learning by interaction with environment)
- Transformers (NLP, new models have billions of learnable weights)
- TBC…

Toolkit

Python is the language of choice for Deep Learning research and prototyping, but compiled languages are often used for production uses (e.g. C++, Go)

- Pillow
- NLTK
- PyTorch
    - Pytorch Lightning/Ignite
- Tensorflow (has R API)
    - Keras
- ONNX

Resources

- http://www.deeplearningbook.org/
- https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
- https://www.tensorflow.org/tutorials
- StackOverflow
- Elements of Statistical Learning, Chapter 11