Deep Neural Networks Applications in Bioinformatics

Yuzhen Ye
School of Informatics and Computing, Indiana University

Contents

- From shallow to deep
- Vision & Alpha Go
- Applications in Bioinformatics

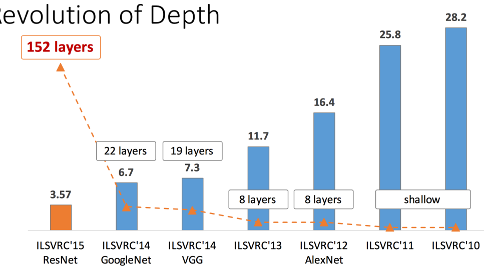
ResNets @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

*improvements are relative numbers

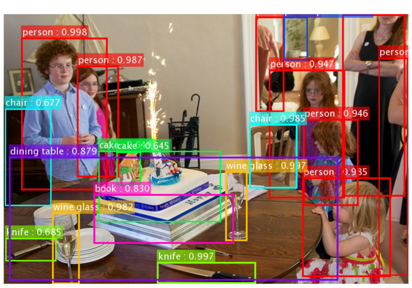
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Revolution of Depth



ImageNet Classification top-5 error (%)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

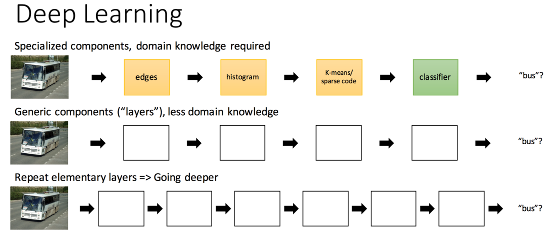


ResNet's object detection result on COCO

*the original image is from the COCO dataset

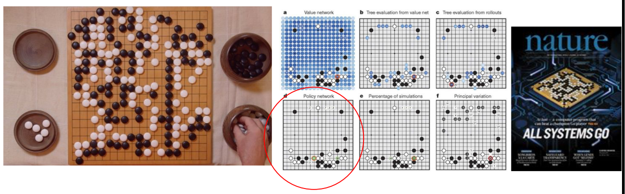
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Deep Learning



- End-to-end learning
- Richer solution space

Case Study Bonus: DeepMind's AlphaGo



The input to the policy network is a 19 × 19 × 48 image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23 × 23 image, then convolves *k* filters of kernel size 5 × 5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21 × 21 image, then convolves *k* filters of kernel size 3 × 3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1 × 1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used *k* = 192 filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with *k* = 128, 256 and 384 filters.

policy network:
[19x19x48] Input
CONV1: 192 5x5 filters , stride 1, pad 2 => [19x19x192]
CONV2..12: 192 3x3 filters, stride 1, pad 1 => [19x19x192]
CONV: 1 1x1 filter, stride 1, pad 0 => [19x19] *(probability map of promising moves)*

Potential problems with going deep

- Decay of gradients
  - When sigmoid activation function is used, the gradient decays to 0.25 of the previous layer
  - Use ReLU instead of sigmoid
- Local minimum

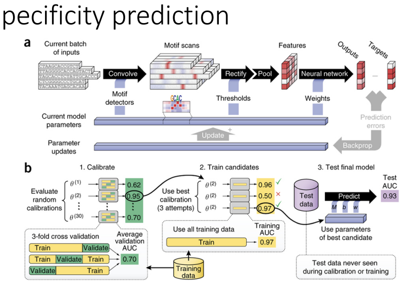
Reducing the dimensionality of data with neural networks.

- High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.
- Ref: Science. 2006 Jul 28;313(5786):504-7.

Deep learning for bioinformatics

- **Review: Deep learning for computational biology**
- **The human splicing code reveals new insights into the genetic determinants of disease**
- **Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning** (DeepBind)
- **Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks**
- **Predicting effects of noncoding variants with deep learning–based sequence model** (DeepSEA)

DeepBind for DNA- and RNA-binding protein specificity prediction



http://www.nature.com/nbt/journal/v33/n8/full/nbt.3300.html

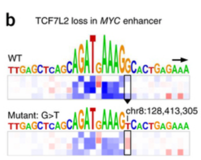
DeepBind: Training

- Training dataset
  - DeepBind uses a set of sequences and, for each sequence, an experimentally determined binding score. Sequences can have varying lengths (14–101 nt in our experiments), and binding scores can be real-valued measurements or binary class labels.
- Training: For a sequence *s*, DeepBind computes a binding score *f*(*s*) using four stages: $f(s) = \text{net}_W(\text{pool}(\text{rect}_b(\text{conv}_M(s))))$
  - The convolution stage (conv_M) scans a set of motif detectors with parameters *M* across the sequence. Motif detector M_k is a $4 \times m$ matrix, much like a PWM of length *m* but without requiring coefficients to be probabilities or log odds ratios.
  - The rectification stage isolates positions with a good pattern match by shifting the response of detector M_k by b_k and clamping all negative values to zero.
  - The pooling stage computes the maximum and average of each motif detector's rectified response across the sequence; maximizing helps to identify the presence of longer motifs, whereas averaging helps to identify cumulative effects of short motifs, and the contribution of each is determined automatically by learning.
  - These values are fed into a nonlinear neural network with weights *W*, which combines the responses to produce a score

More on training datasets & DeepMind models

- DeepBind models were trained on a combined 12 terabases of sequence data, spanning thousands of public PBM, RNAcompete, ChIP-seq and HT-SELEX experiments.
- the source code for DeepBind together with an online repository (http://tools.genes.toronto.edu/deepbind/) of 927 DeepBind models representing 538 distinct transcription factors and 194 distinct RBPs, each of which was trained on high-quality data and can be applied to score new sequences using an easily installed executable file with no hardware or software requirements.

DeepBind mutation maps for understanding disease-causing SNVs associated with transcription factor binding.



A cancer risk variant in a *MYC* enhancer weakens a TCF7L2 binding site.