Unsupervised Learning of Invariances in Deep Networks

Michael Harris
David Kamm
Jaehyun Park
Jeffrey Wang
Motivation and Goals

• Build a neural network which can learn complex invariances without hardcoding them
• Biologically-inspired model of simple and complex cells
• Want network to generalize to different types of data (e.g. audio/video)
Network Structure

\[
\sqrt{} \quad h_1(x) \quad h_2(x) \quad h_3(x)
\]

\[
v \quad (\cdot)^2 \quad w
\]

\[
x_1 \quad x_2 \quad x_3
\]

Sublayer 1

Sublayer 2
Learning the Weights

• We originally tried to learn V, but we did not quite see the “peaks” we wanted (maybe due to the data or regularization), so we set V to be a neighborhood matrix to mimic topographical layout of visual cortex
Learning the Weights

• We learn $W$ using the sparse cost objective, i.e. minimizing activations at the output, as this produces very localized gabor filters (also constrain rows of $W$ to be far apart)
Using CUDA

- All training done on the GPU, which gives us an estimated 25x speedup (with just 1600 hidden units); training on CPU would have been infeasible
- Mini-batch approach to maximize GPU effectiveness
Vision Task: Classification

- CIFAR 10 dataset: tiny 32x32 images belonging to 10 different classes
- Training set: 50,000 images
- Test set: 10,000 images
- Added a softmax layer on top of network
Attempted Methods

• CIFAR data is RGB, i.e. there are three channels for each image; we can combine channels to improve classification (sort of)
• Combine output of both W and V layers before softmax
• We originally trained the W-V layers on data independent of CIFAR, but we found that training on CIFAR itself produced different bases
Results and Future Work

• Unfortunately, none of the attempted methods really improved our classification results on the CIFAR test set (note that our bases are trained in an unsupervised setting)
  – 56% using 1600 hidden units
  – 59% using 10,000 hidden units
  – State-of-the-art: 65% using RBM (Krizhevsky)

• Ideas for future work: better features, stacking more layers, different datasets
SparseCudaMatrix Design

- Motivation: sparse weight matrices, RBMs, etc.
- Structure: row-major and column-major entries arrays, index into each array (supports fast multiplication and transpose)
- Key Assumption: locations of (potentially) nonzero entries do not change
- Performance Assumption: number of entries per row/per col are relatively well balanced
SparseCudaMatrix Features

- Component-wise arithmetic: assume nonzero entries are the same and blaze through arithmetic (see key assumption!)
- Sparse * Sparse $\rightarrow$ Sparse
- Sparse * Dense $\rightarrow$ (Dense or Sparse)
- (Dense * Dense) .* Sparse
  - Implemented by dotting necessary row/col of dense matrices
- Full matrix reductions (sum, norm, min, max)
Acknowledgements

• Prof. Andrew Ng
• Quoc Le
• Andrew Saxe