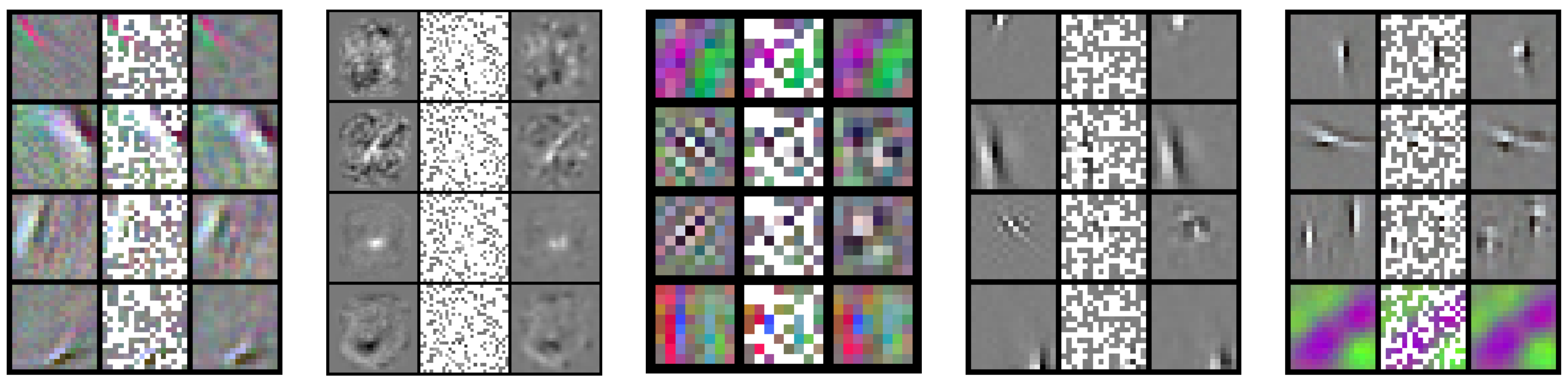
Predicting Parameters in Deep Learning

Misha Denil[1] Babak Shakibi[1] Laurent Dinh[2] Marc'Aurelio Ranzato[3] Nando de Freitas[1]

[1]University of British Columbia [2]École Centrale de Paris [3]Google Inc.

Motivation

- There is significant redundancy in the parameterization of deep learning models.

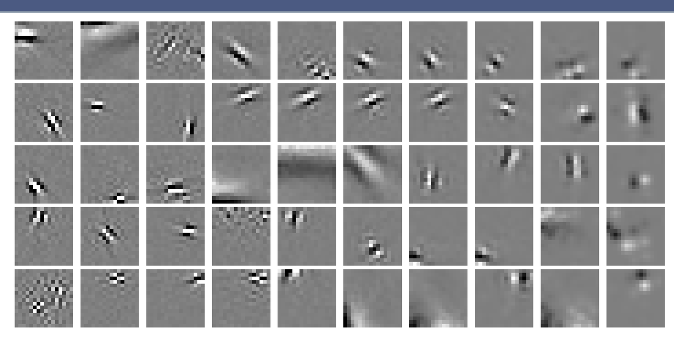


Convnet on STL-10 | MLP on MNIST | Convnet on CIFAR-10 | RICA on Hyvärinen's natural images | RICA on STL-10

- Left: Features of a trained model.
- Center: A random selection of parameters from each feature.
- Right: Predicted features based on parameters in the center column.

- Can we learn features without learning all of the parameters?

Feature Prediction



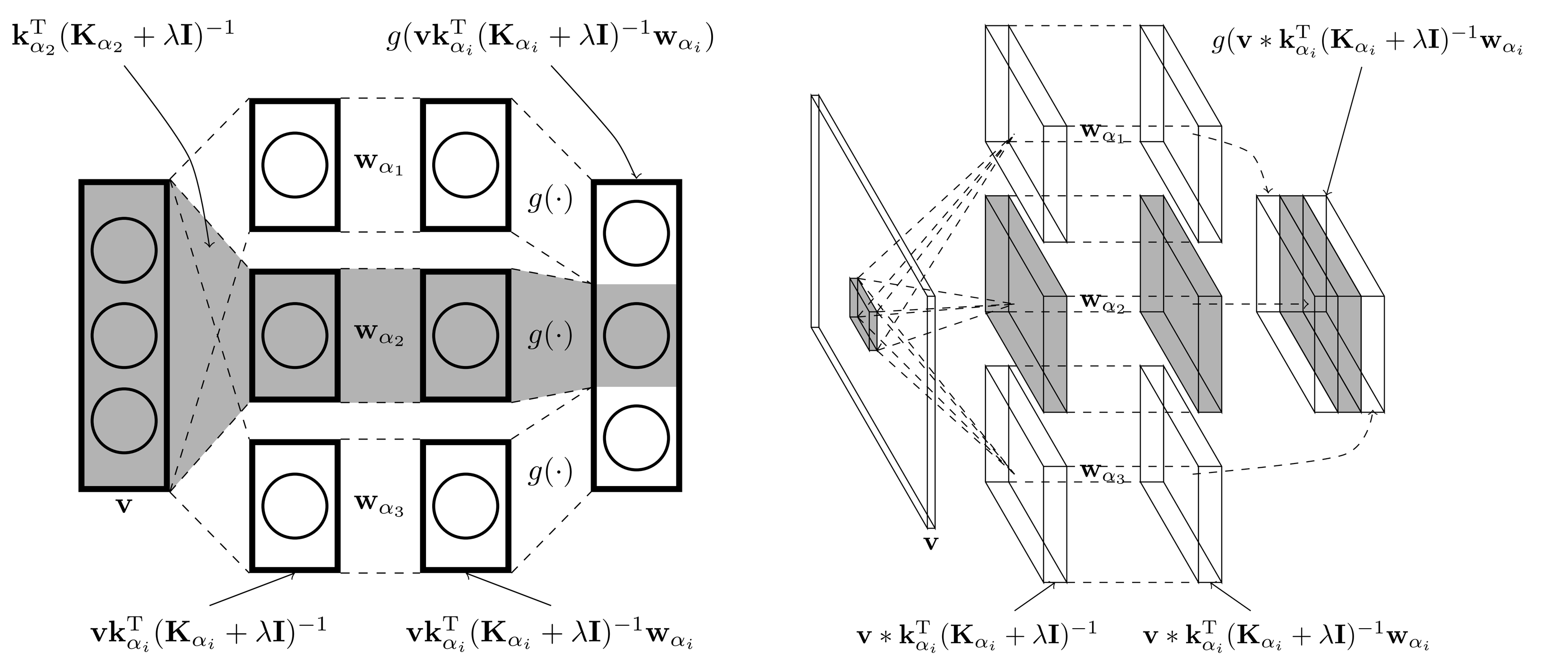
- Instead of learning a full weight matrix learn a low rank factorization $\mathbf{W} = \mathbf{UV}$.
- Optimize both factors: doesn't work well.
- Factorization is still redundant: $\mathbf{W} = (\mathbf{UQ})(\mathbf{Q}^{-1}\mathbf{V})$.
- Remove redundancy by fixing $\mathbf{U}$. Learn only $\mathbf{V}$.
- Many options for $\mathbf{U}$:
  - Easy choices: Random projections (Gaussian IID); Random connections (Columns of identity).
    - These still don't work very well.
- Better choice: Kernel ridge regression: $\mathbf{W} = \mathbf{k}_\alpha^{\mathrm{T}}(\mathbf{K}_\alpha + \lambda\mathbf{I})^{-1}\mathbf{W}_\alpha$; $(\mathbf{U} = \mathbf{k}_\alpha^{\mathrm{T}}(\mathbf{K}_\alpha + \lambda\mathbf{I})^{-1})$
- For images use the squared exponential kernel.
- When no topology is available use the empirical covariance, or the empirical squared covariance.
  - Pre-train layerwise as an autoencoder to initialize the kernels.

Two perspectives

- Feature completion: $g^{-1}(\mathbf{h}) = \mathbf{v}(\mathbf{UV}) = \mathbf{vW}$
  - Predict $\mathbf{W}$ using ridge regression, evaluate the network like normal.
- Spatial pooling: $g^{-1}(\mathbf{h}) = (\mathbf{vU})\mathbf{V} = \tilde{\mathbf{v}}\mathbf{V}$
  - $\mathbf{U}$ matrix applies a linear pooling operator to $\mathbf{v}$.
  - $\mathbf{V}$ contains the weights of an ordinary layer with the pooled representation as input.

Columnar architecture

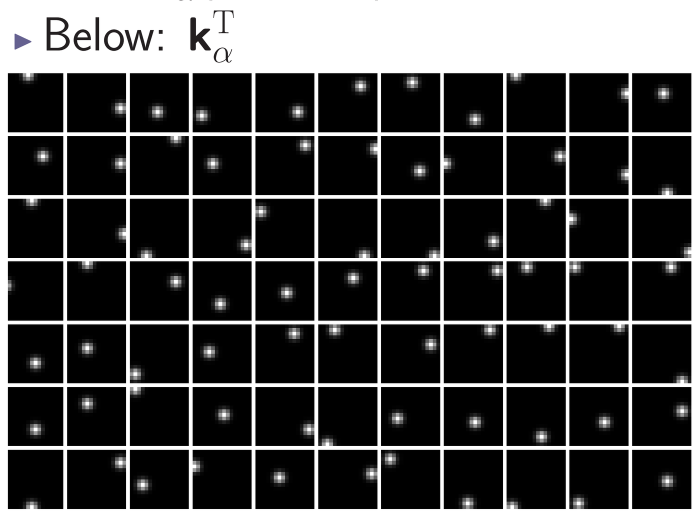
- Split $\mathbf{W}$ into blocks: $\mathbf{W} = [\mathbf{W}_1 \ldots \mathbf{W}_J]$, factor each $\mathbf{W}_j = \mathbf{U}_j\mathbf{V}_j$ separately.
- Several columns of representation, each with different preprocessing.

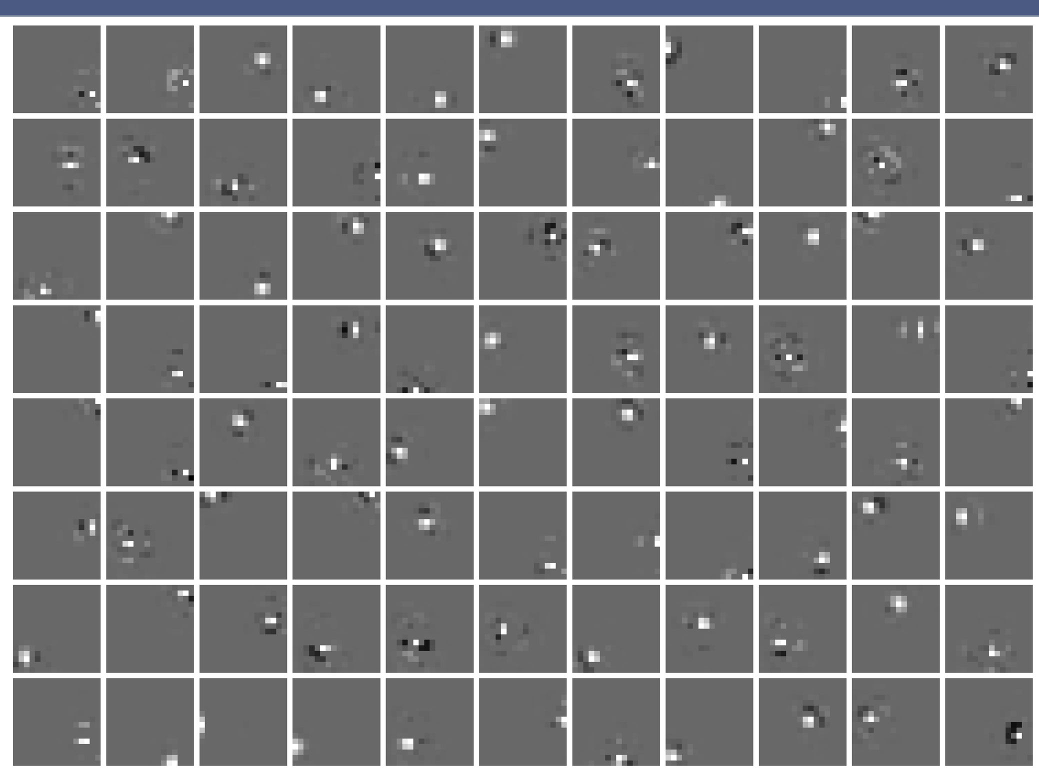


- In densely connected layers this corresponds to different pooling operators, each followed by a densely connected layer.
- In a convolutional network this corresponds to convolution with a fixed filter bank (defined by the $\mathbf{U}$ matrix) followed by a densely connected layer that takes linear combinations of these feature maps.
- In both cases the layer output is the concatenation of each column output.

Pooling Filters

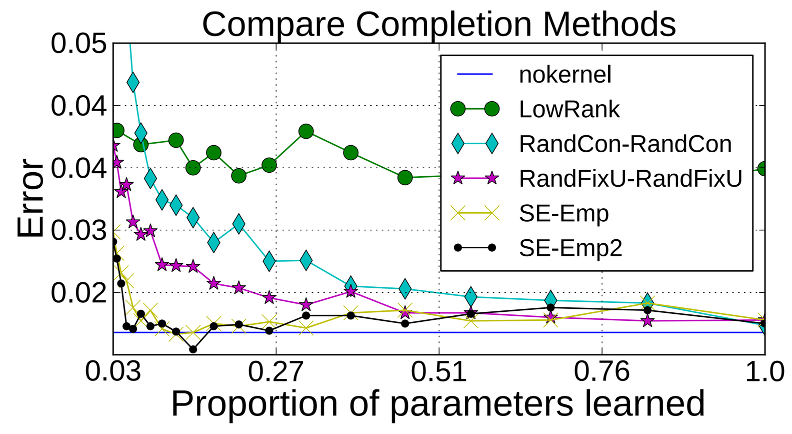
- Examples of fixed filters in the expander matrix (squared exponential kernel).
- Left: $\mathbf{k}_\alpha^{\mathrm{T}}(\mathbf{K}_\alpha + \lambda\mathbf{I})^{-1}$
- Below: $\mathbf{k}_\alpha^{\mathrm{T}}$

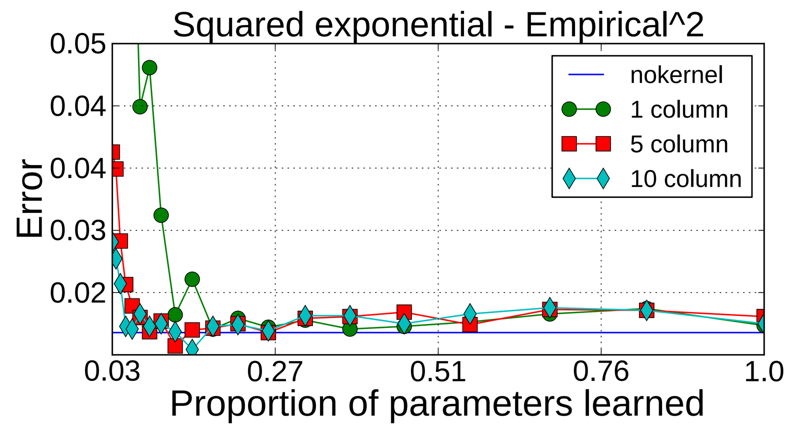




Multilayer Perceptrons

- Some experiments with a 784–500–500–10 MLP on MNIST.



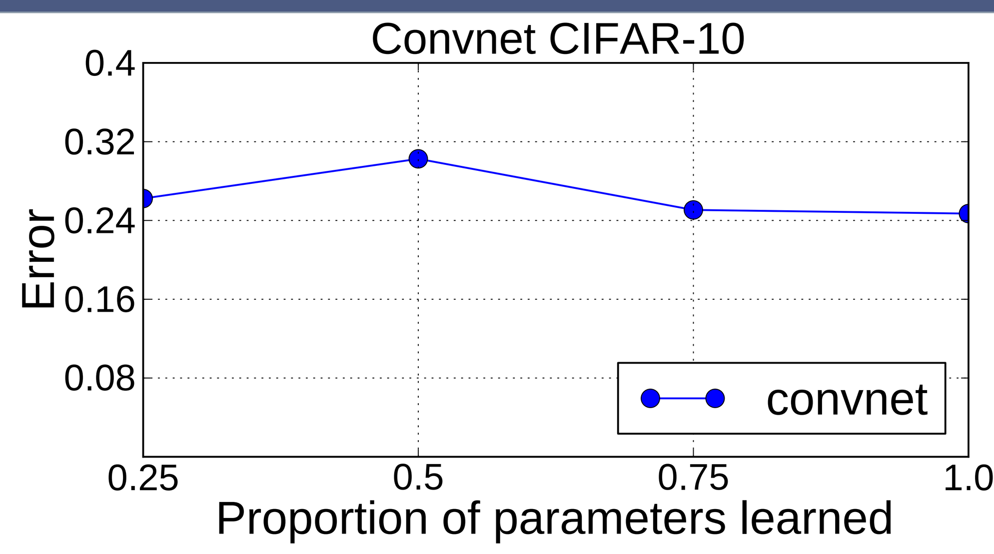


- Left: Comparing different expander construction methods to predict the weights in the first two layers of the MLP network.
- Right: The effect of changing the number of columns. With 10 columns we are able to predict more than 95% of the dynamic parameters from the first two layers of the network without any substantial drop in accuracy.

- All models (except for LowRank) are pretrained layerwise as autoencoders.

Convolutional Networks on CIFAR-10

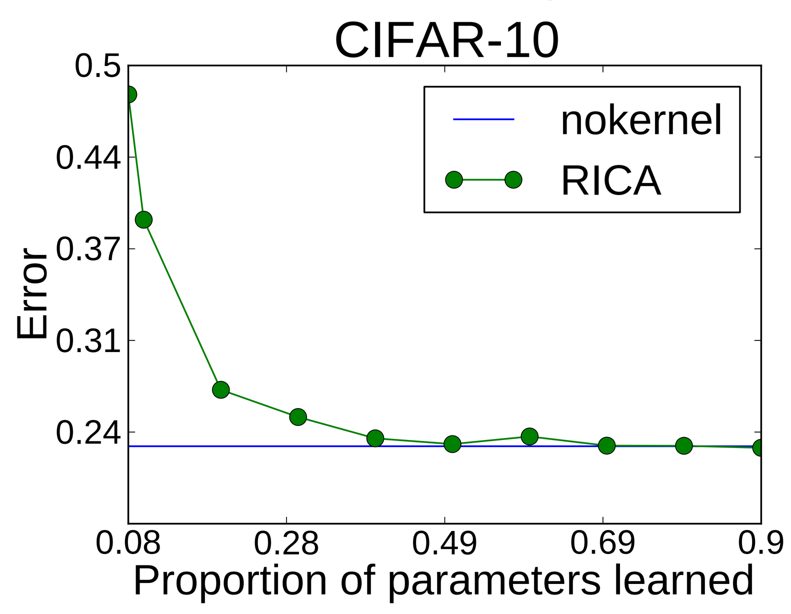
- Input: $32 \times 32 \times 3$
- Layer 1 (convolution): 48 filters of size $8 \times 8 \times 3$.
- Layer 2 (convolution): 64 filters of size $8 \times 8 \times 48$.
- Layer 3 (convolution): 64 filters of size $5 \times 5 \times 64$.
- Layer 4 (dense): 500 hidden units.
- Layer 5 (softmax): 10 classes.

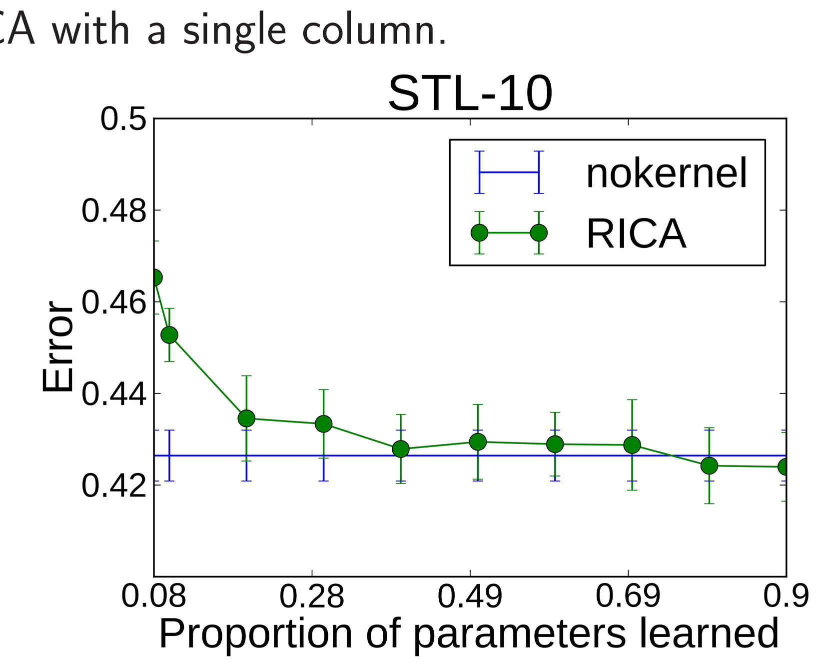


- The convolutional layers each have one column and the fully connected layer has five columns.
- Convolutional layers have a natural topological structure, so we use an expander matrix constructed with the squared exponential kernel in each convolutional layer. The input to the fully connected layer at the top of the network comes from a convolutional layer so we use ridge regression with the squared exponential kernel to predict parameters in this layer as well.

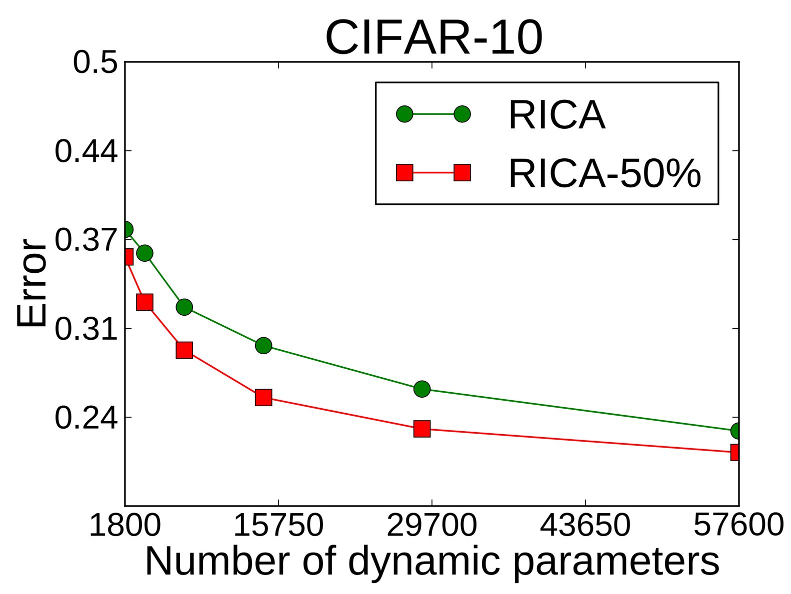
Reconstruction ICA

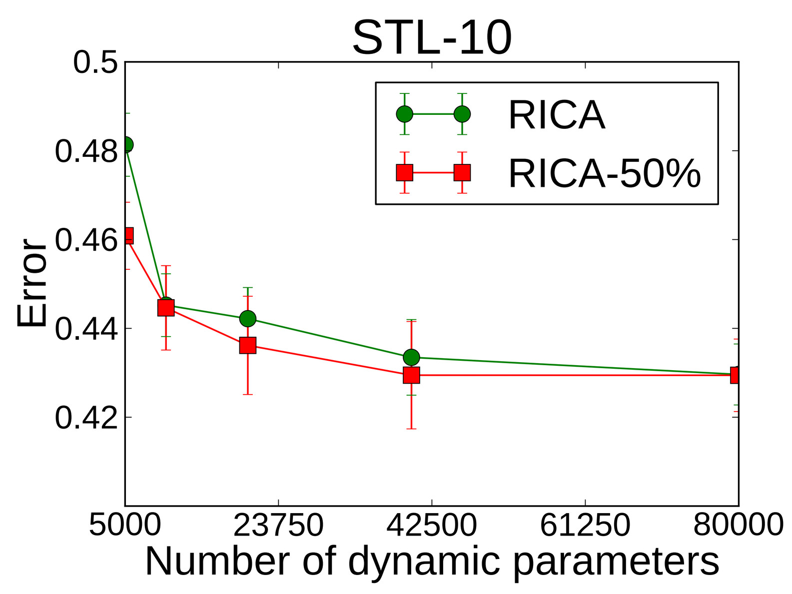
- Comparison of the performance of RICA with and without parameter prediction on CIFAR-10 and STL-10. Both plots use RICA with a single column.





- Comparison of RICA, and RICA with 50% parameter prediction using the same number of dynamic parameters (i.e. RICA-50% has twice as many features). There is a substantial gain in accuracy with the same number of dynamic parameters using our technique.





Future Work

- Better kernels, especially when no natural topology exists.
- Deep columns.
- Better spacial sampling techniques: locally dense?

Predicting Parameters in Deep Learning

Misha Denil[1] Babak Shakibi[1] Laurent Dinh[2] Marc'Aurelio Ranzato[3] Nando de Freitas[1]
[1]University of British Columbia [2]École Centrale de Paris [3]Google Inc.

Predicting Parameters in Deep Learning

Misha Denil[1] Babak Shakibi[1] Laurent Dinh[2] Marc'Aurelio Ranzato[3] Nando de Freitas[1]
[1]University of British Columbia [2]École Centrale de Paris [3]Google Inc.