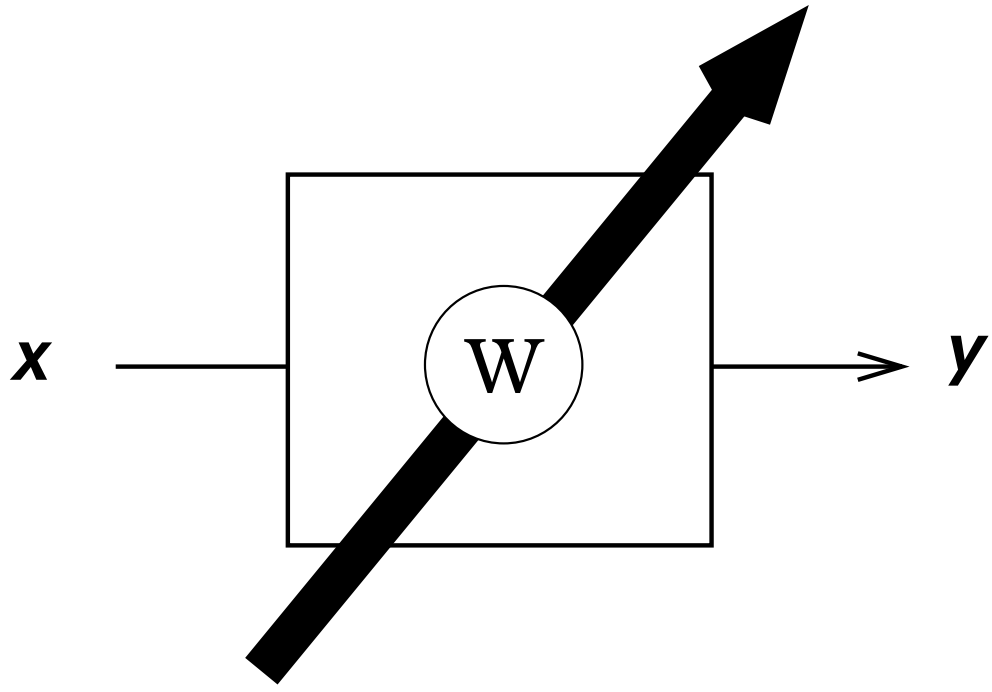
Towards hardware-friendlier neural networks

Roland Memisevic

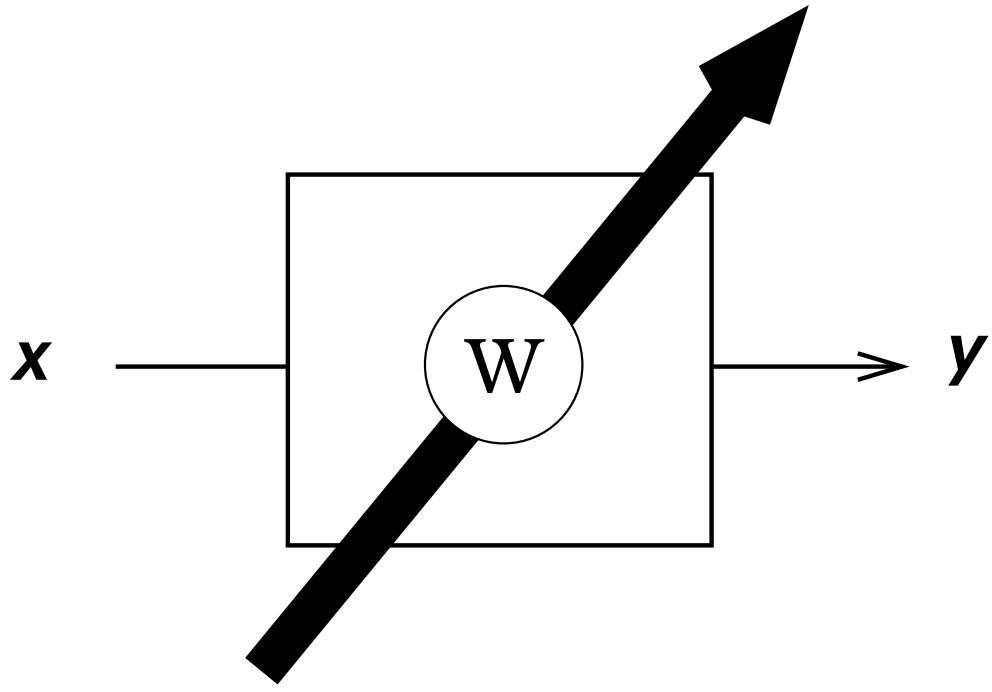
University of Montreal

December 5, 2015

Machine Learning

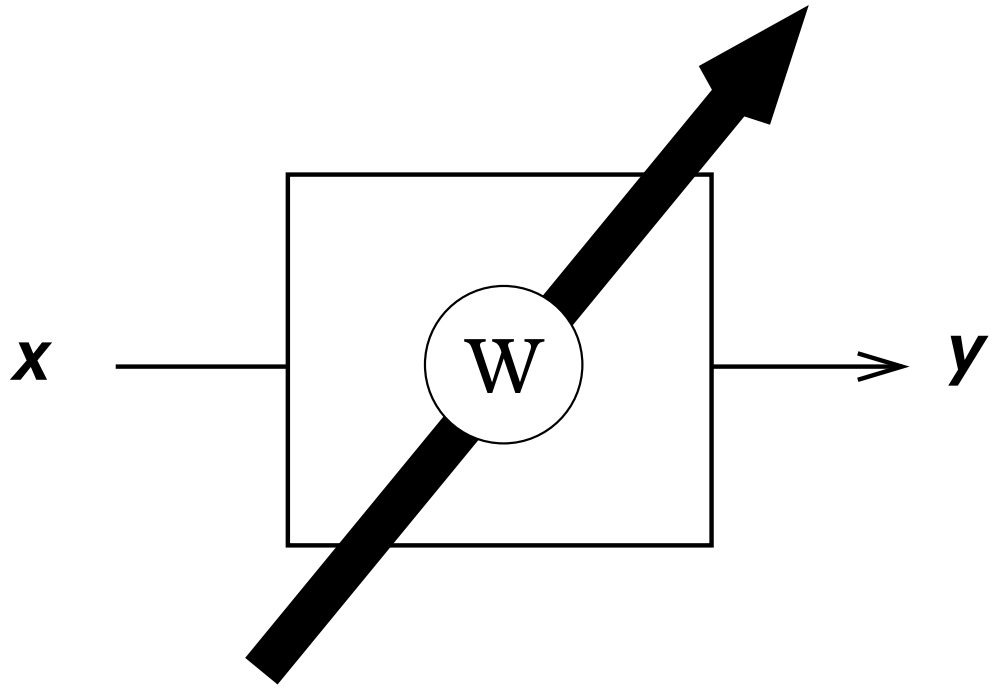


Machine Learning



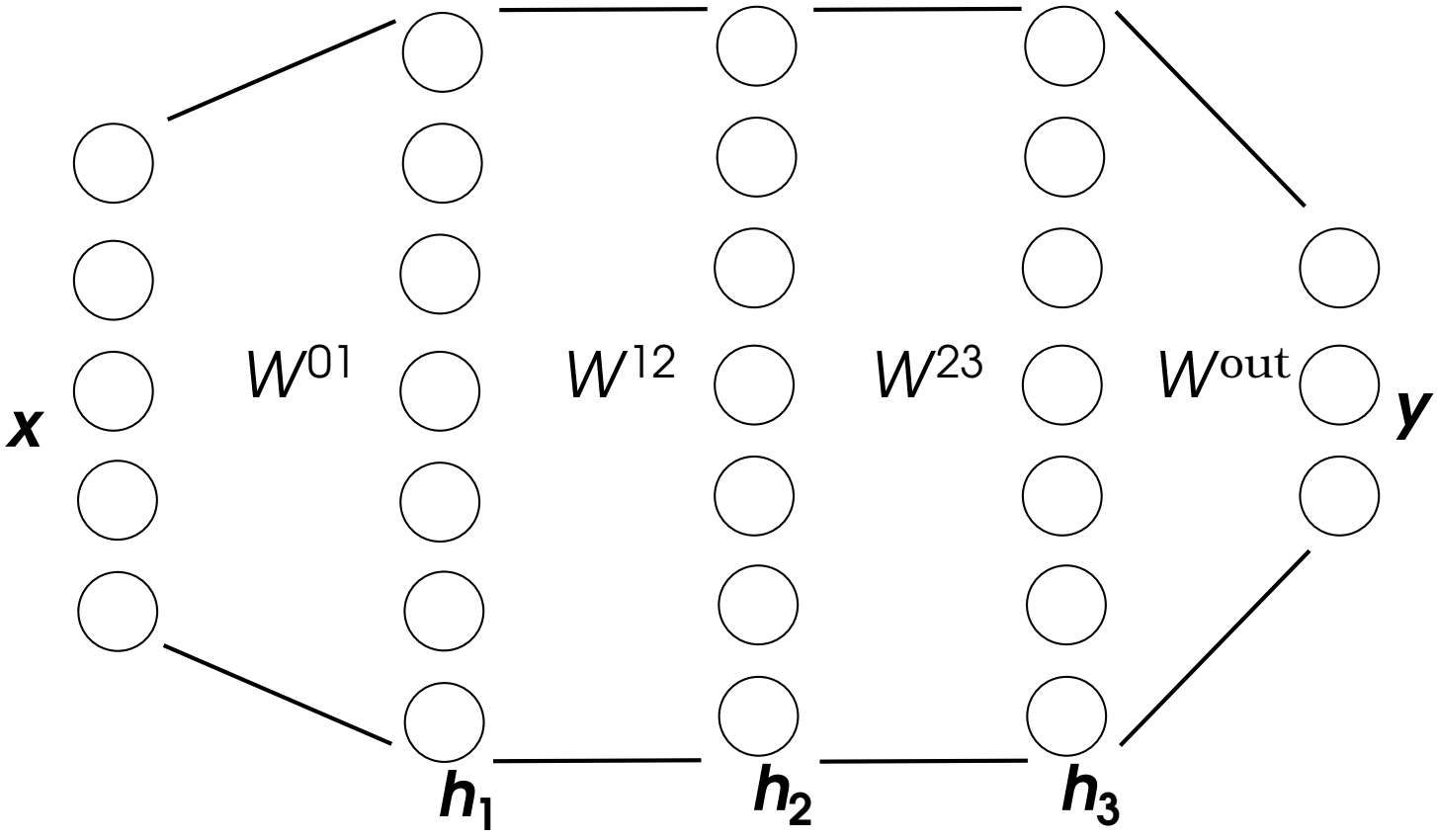
- Learning allows us to harness **training data**

Machine Learning



- Learning allows us to harness **training data**
- Learning allows us to harness **parallelization**

Neural networks



Floating point multiplication

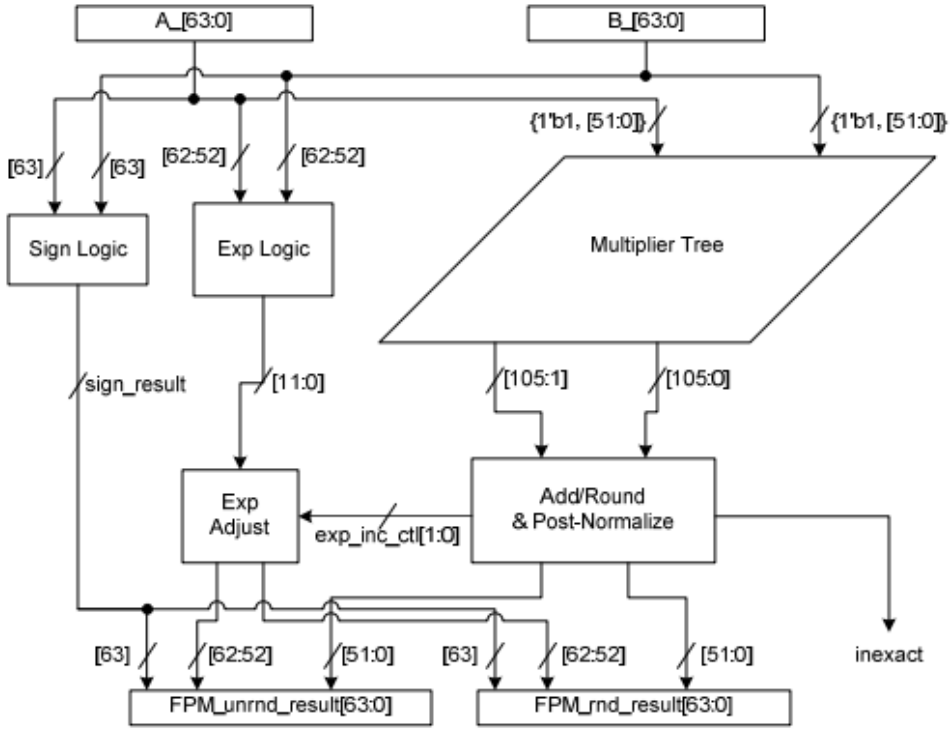


figure from:

http://www.gamasutra.com/view/news/128521/Indepth_IEEE_754_Multiplication_And_Addition.php

- Waste of circuitry?

Binary Connect

- Courbariaux, Bengio, David (2015):
- Stochastically binarize (or ternarize) weights during **propagation** (forward and backward) of activations.

$$\begin{aligned} p(W_{ij} = 1) &= \frac{W_{ij} + 1}{2} \\ p(W_{ij} = -1) &= 1 - P(W_{ij} = 1) \end{aligned}$$

- Use full resolution for weight **updates**.

Quantized Backprop

$$
\begin{aligned}
\Delta W &= \left[\eta\delta \circ h'\left(Wx + b\right)\right] x^{\mathrm{T}} \\
\Delta b &= \eta\delta \circ h'\left(Wx + b\right) \\
\delta &= \left[W^{\mathrm{T}}\delta\right] \circ h'\left(Wx + b\right)
\end{aligned}
$$

- Lin, Courbariaux, Memisevic, Bengio (2015):
- Eliminate multiplications in weight updates by quantizing activations to power of two
- see also: (Simard, Graf 1992)

Classification performance

	Full precision	Binary connect	Binary connect + Quantized backprop	Ternary connect + Quantized backprop
MNIST	1.33%	1.23%	1.29%	1.15%
CIFAR10	15.64%	12.04%	12.08%	12.01%
SVHN	2.85%	2.47%	2.48%	2.42%

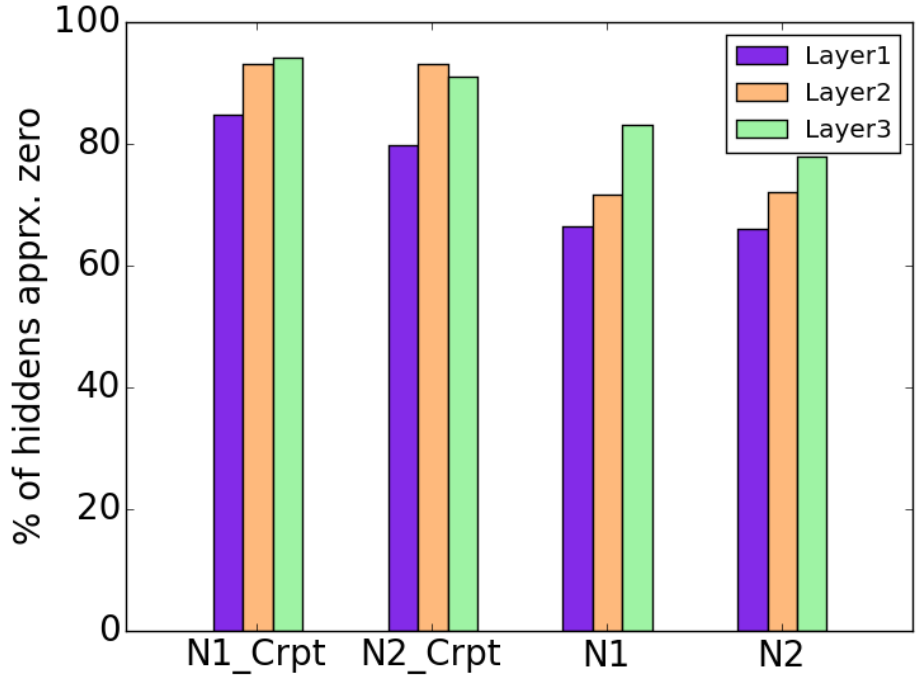
Two issues with convnets

- The number of neurons is very large: conv-nets maximize *the ratio* $\frac{\#neurons}{\#parameters}$
- During learning, updates need to be distributed across parameters, which requires long-range communication. (During inference, it is common to distribute each filter over image locations, too).

Improving fully connected nets

- Architecture (Linear bottleneck layers)
- Non-linearities (Zero-bias relus)
- Optimization (Unsupervised pre-training helps)

Network activation sparsity

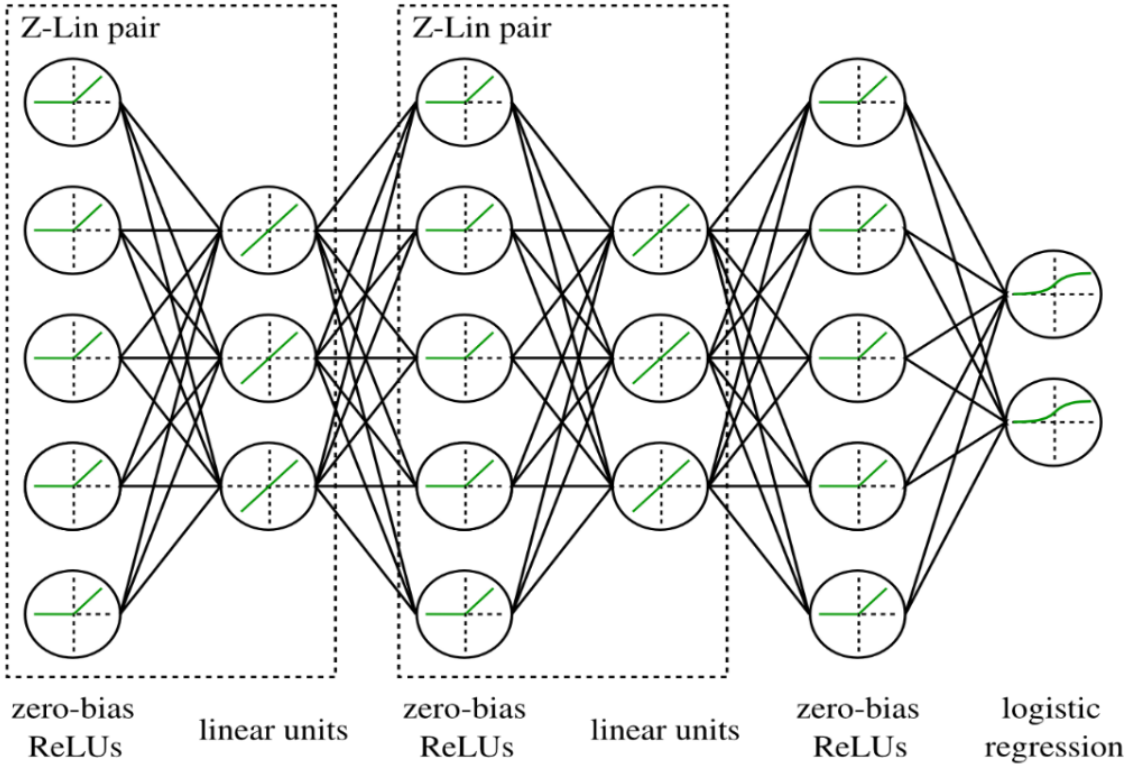


Sparsity levels in two networks trained on CIFAR-10. N1=(1000-2000-3000), N2=(2000-2000-2000 units). (N1_Crpt, N2_Crpt trained with dropout).

figure by Kishore Konda

- Unfortunately, sparsity leads to "update scarcity".

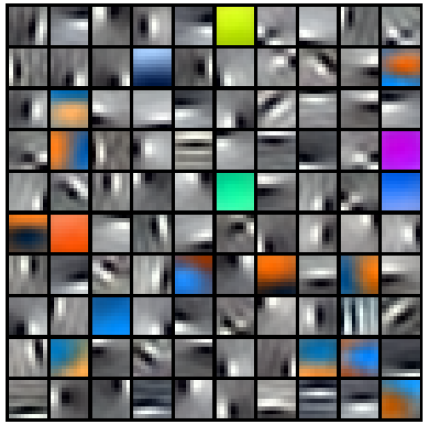
Interleaved linear layers

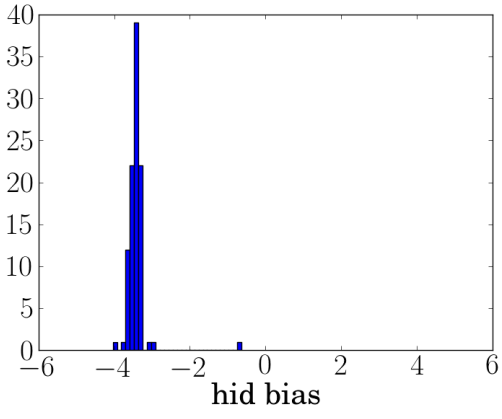


- Reduce parameters, increase gradient density.

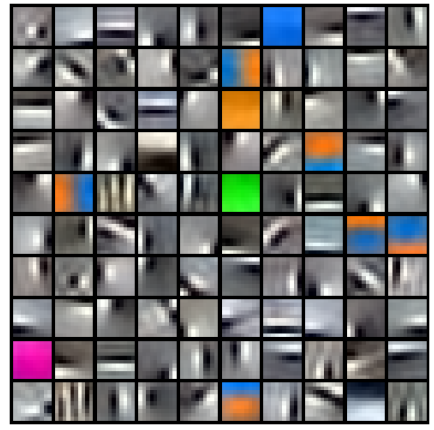
Autoencoders learn negative biases

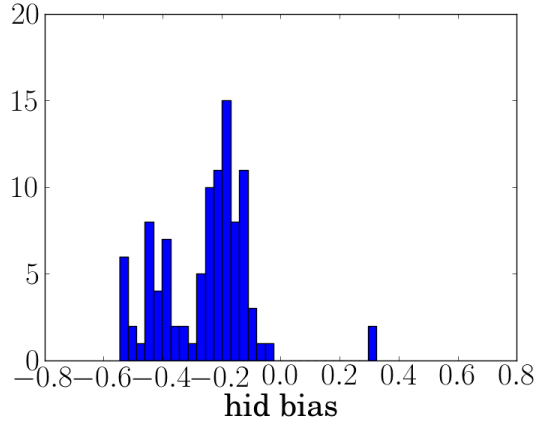
contractive AE (sigmoid)





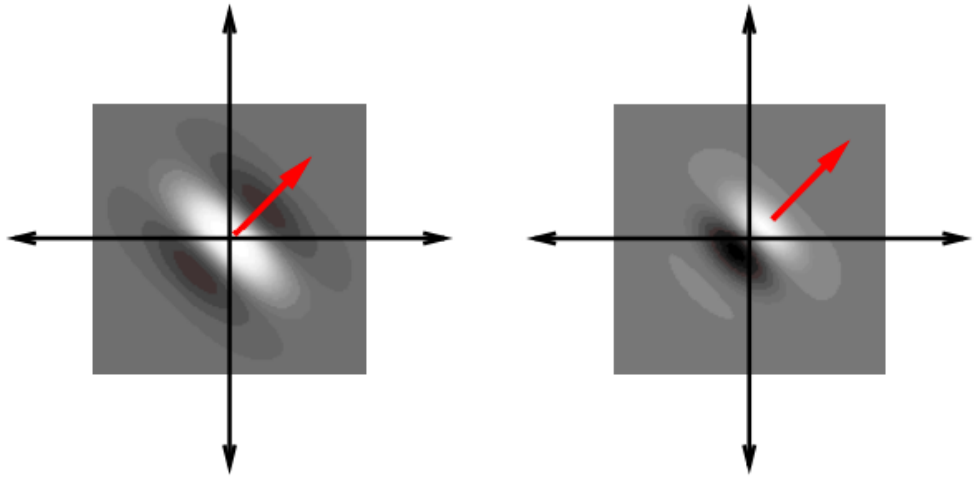
denoising AE (ReLU)





- see also M. Ranzato, et al. 2007, K. Kavukcuoglu, et al., 2008.

The 2-d subspaces for images



$\boldsymbol{w}_k$ $\boldsymbol{w}_\ell$

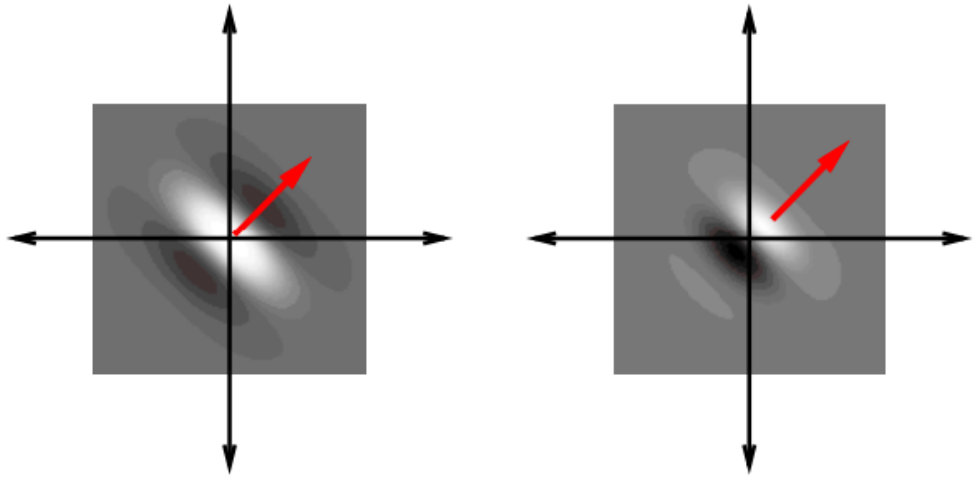
For orthogonal features, which coefficients minimize reconstruction error?

$$\boldsymbol{x} \approx a_k \boldsymbol{w}_k + a_\ell \boldsymbol{w}_\ell$$

$$a_k = ?, \quad a_\ell = ?$$

figures by Javier Movellan

The 2-d subspaces for images



$\boldsymbol{w}_k$ $\boldsymbol{w}_\ell$

For orthogonal features, which coefficients minimize reconstruction error?

$$\boldsymbol{x} \approx a_k \boldsymbol{w}_k + a_\ell \boldsymbol{w}_\ell$$

$$a_k = \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}, \quad a_\ell = \boldsymbol{w}_\ell^{\mathrm{T}} \boldsymbol{x}$$

figures by Javier Movellan

Do autoencoders orthogonalize weights?

- Autoencoders minimize

$$\left(\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{x}\right)^2$$

  using as the reconstruction:

$$\boldsymbol{r}(\boldsymbol{x}) = W\boldsymbol{h}(\boldsymbol{x}) = \sum_{k:h_k \neq 0} h_k \boldsymbol{w}_k$$

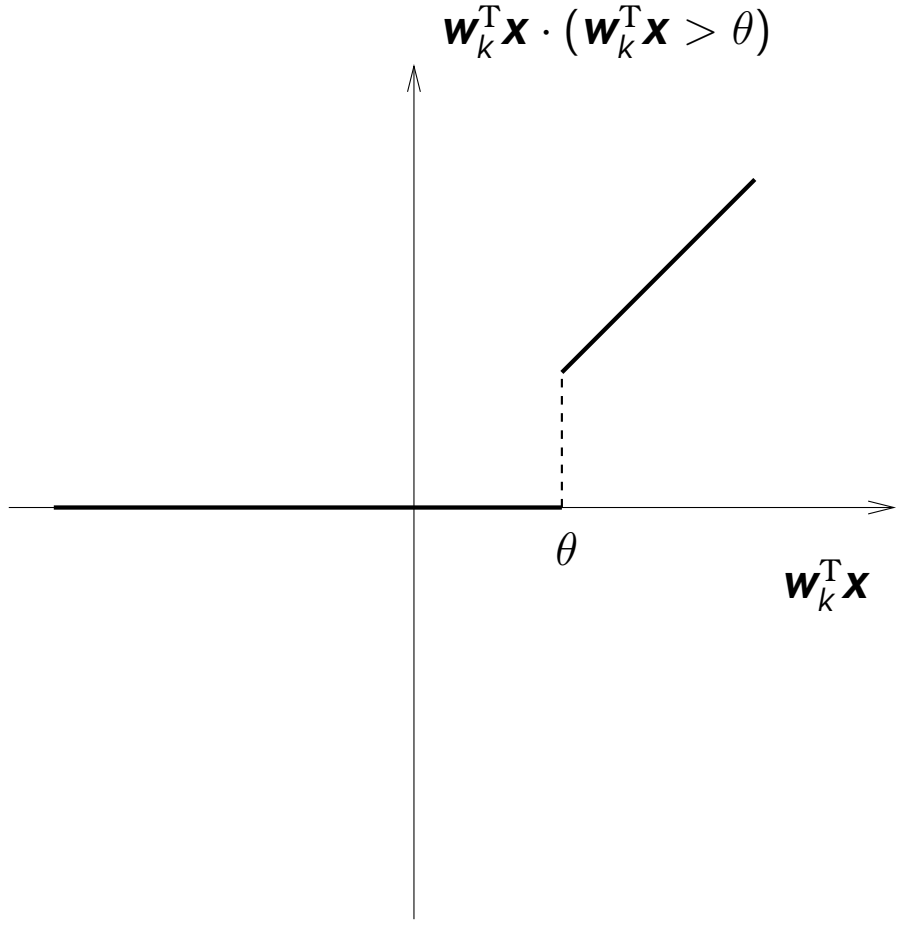
  where h_k is the output of hidden unit k
- For orthonormal active weights the optimal coefficients would be:

$$h_k = \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}$$

- In reality, a ReLU autoencoder uses

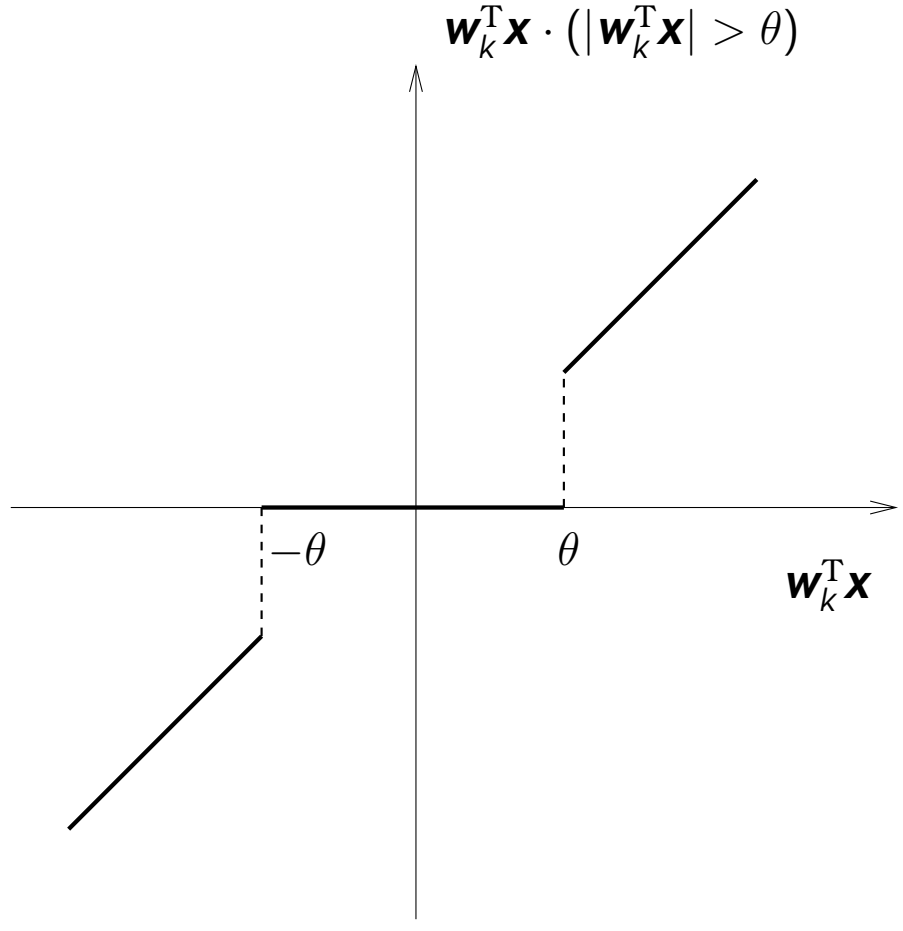
$$h_k = \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x} + b_k$$

Truncated rectified unit (Trec)



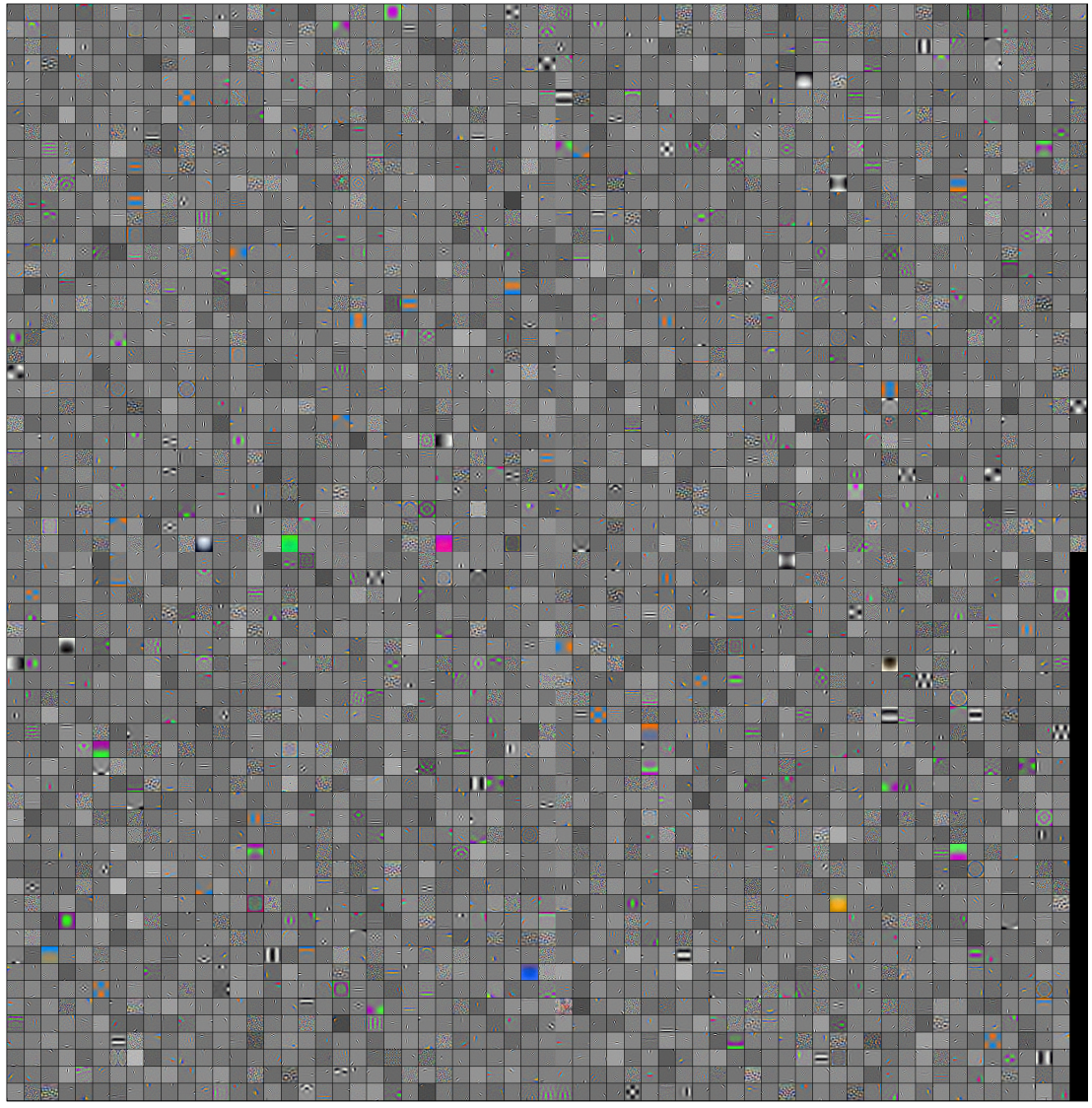
- Like spike-and-slab, hard-threshold, "coring"

Truncated linear unit (TLin)

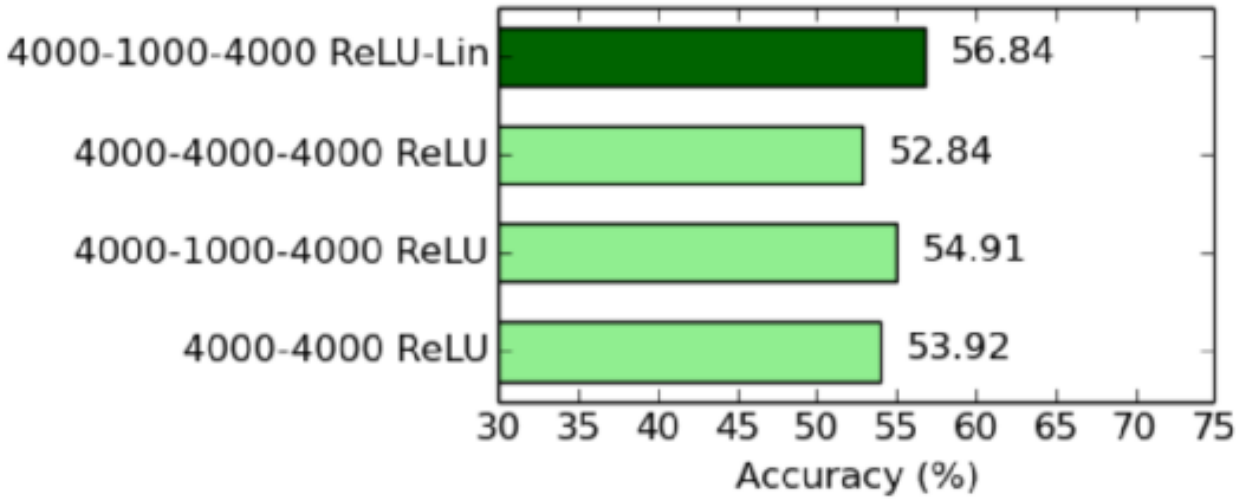


- Like spike-and-slab, hard-threshold, "coring"

ZAE features from tiny images (Torralba et al.)

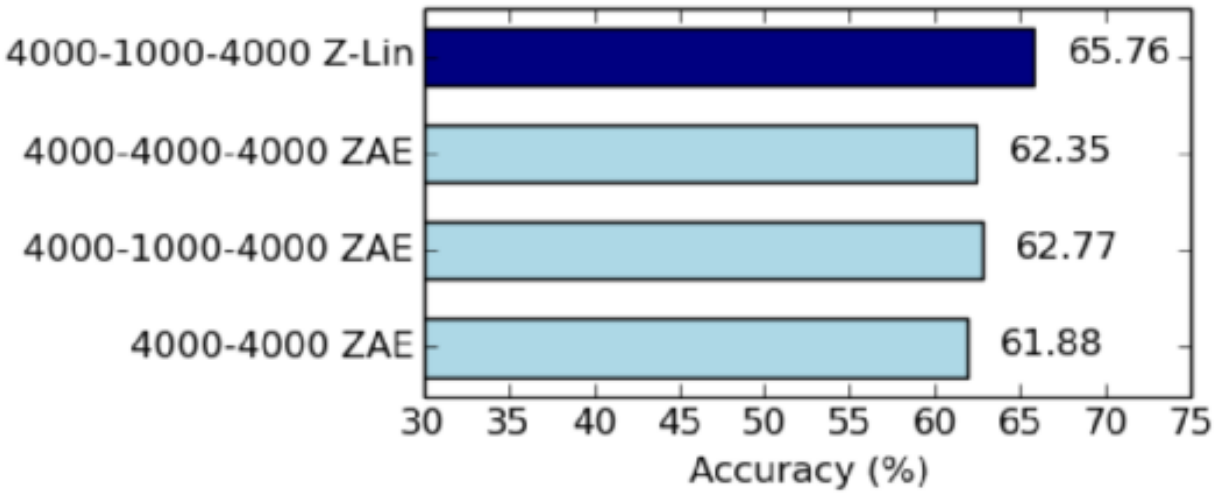


Deep fully-connected CIFAR-10



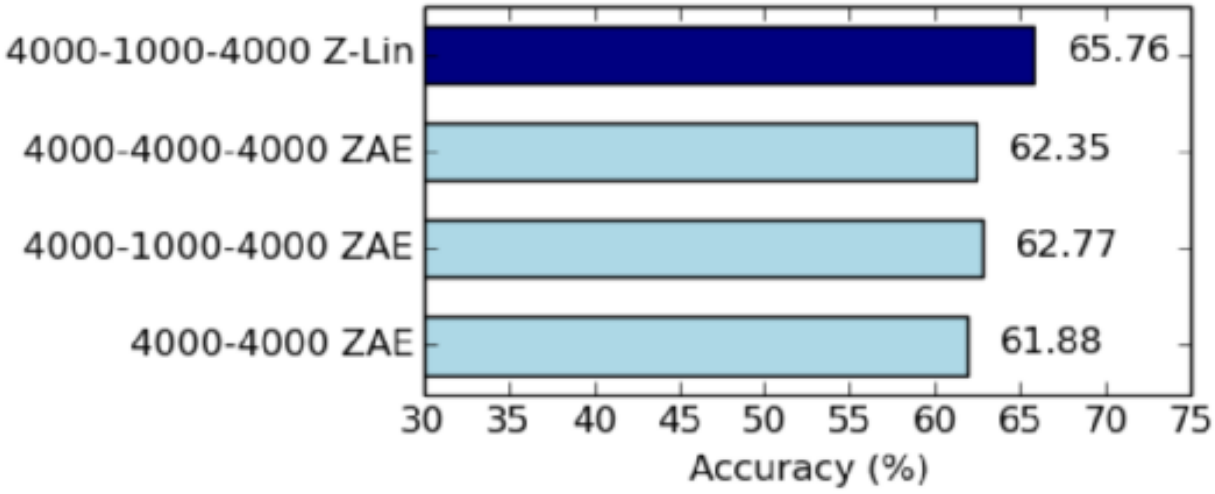
(Zhouhan Lin)

Deep fully-connected CIFAR-10



(Zhouhan Lin)

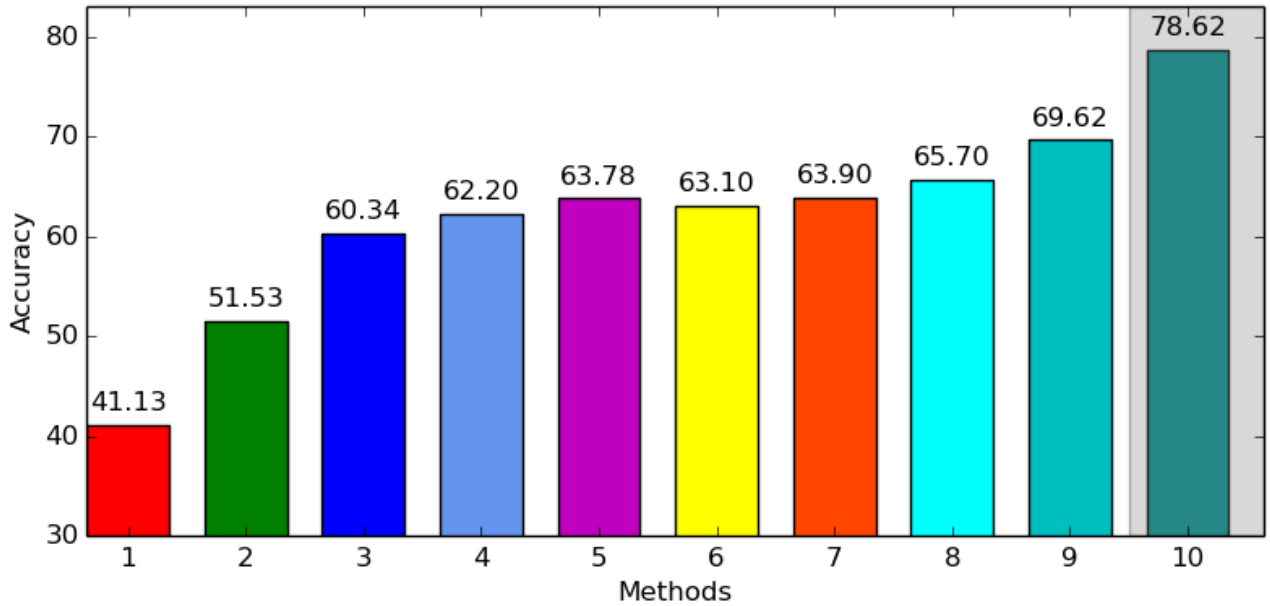
Deep fully-connected CIFAR-10



(Zhouhan Lin)

8 layers and dropout: **69.62%**
Training with deformations (not perm-invariant):
78.62%

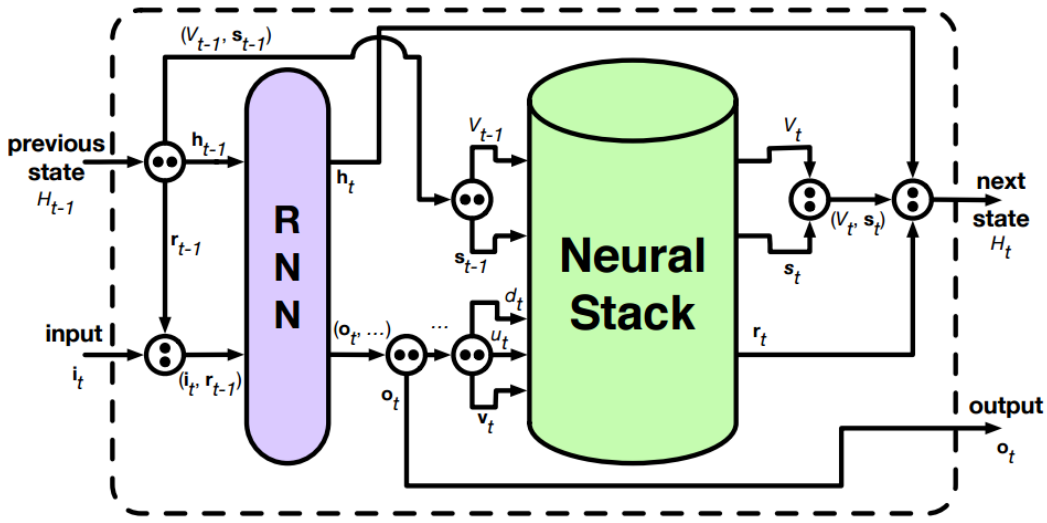
Deep fully-connected CIFAR-10



1. Logistic Regression on whitened data (Krishevsky);
2. Pure backprop on a 782-10000-10 network (Krishevsky);
3. Pure backprop on a 782-10000-10000-10 network (Krishevsky);
4. RBM with 2 hidden layers of 10000 hidden units each, plus alogistic regression (Krishevsky);
5. RBM with 10000 hiddens plus logistic regression (Krishevsky);
6. Fastfood FFT model (13);
7. Zerobias autoencoder of 4000 hidden units with logistic regression (10);
8. 782-4000-1000-4000-10 Z-Lin network trained without dropout;
9. 782-4000-1000-4000-1000-4000-1000-4000-10 Z-Lin network, trained with dropout
10. Z-Lin network the same as (8) but trained with dropout and data augmentation

Differentiable models of computation

- (Bahdanau et al. 2014), (Graves et al, 2014), (Weston et al, 2014), (Grefenstette et al. 2015), etc.



Other neural program applications

- Distilling complicated models or functions (Bucila et al 2006)
- Speeding up routines (eg. Esmaeilzadeh et al. 2012)
- Generating, *then editing*, text (eg. for translation)?
- Generating mocap sequences?

Deep Learning as compute paradigm

- Simulating neural nets on classic hardware never really worked
- Simulating classic hardware on neural nets works quite well
- The reason is that learning is a way to harness any kind of hardware, even the kind of hardware that would be hard to program