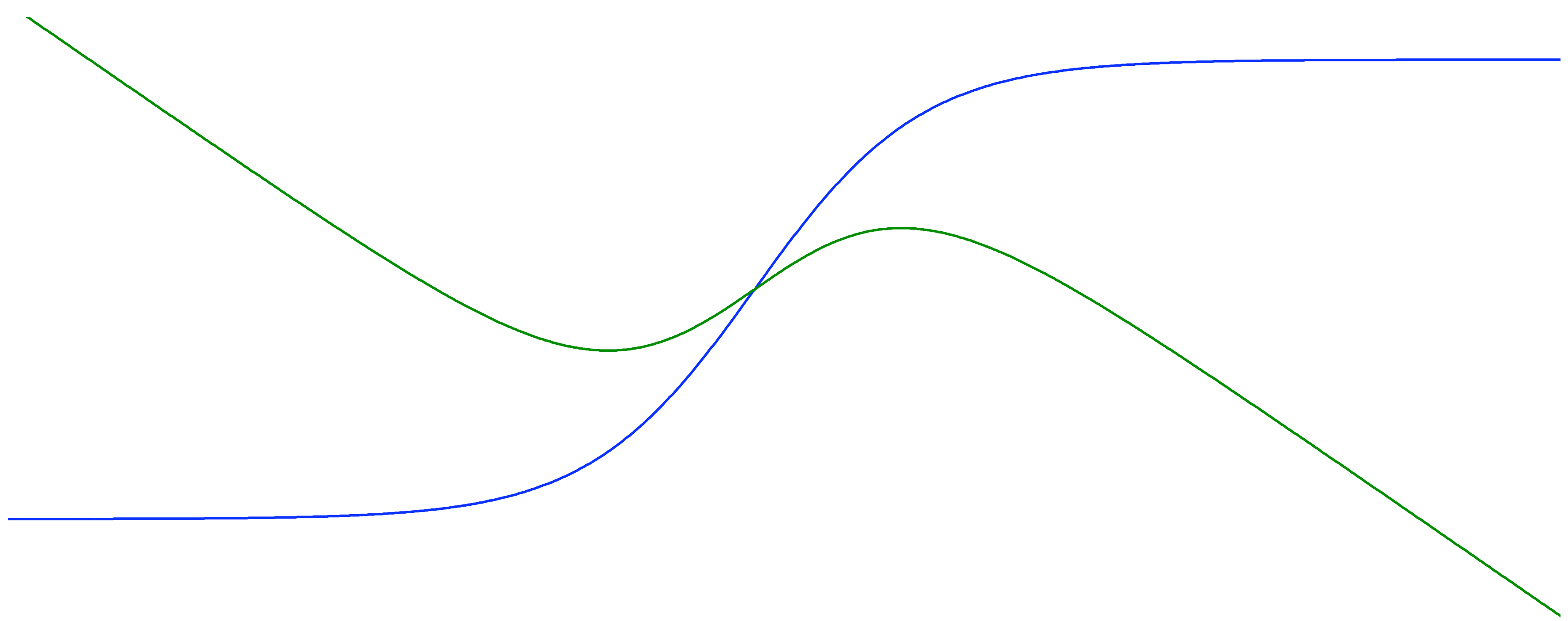
# Deep Learning Made Easier by Linear Transformations in Perceptrons


Tapani Raiko, Harri Valpola, Yann LeCun

Aalto University, New York University


AISTATS 2012



# Background

- Learning deep networks (many hidden layers) used to be difficult
- Layerwise pretraining by RBMs or denoising autoencoders helps
- Could similar performance be achieved with back-propagation?

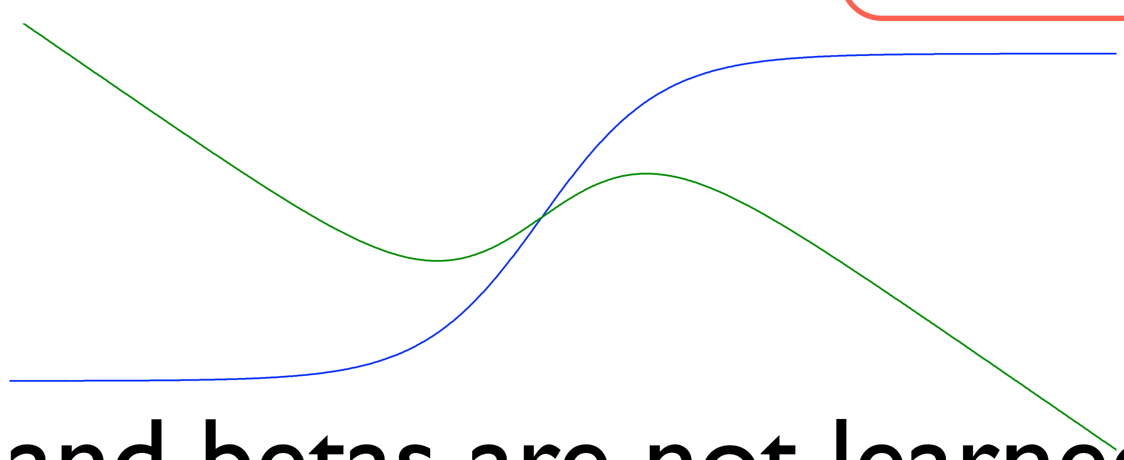
# Proposed method

- Standard MLP (only shallow shown)
- Include shortcut connections C

$$\mathbf{y}_t = \mathbf{A}\mathbf{f}\left(\mathbf{B}\mathbf{x}_t\right) + \mathbf{C}\mathbf{x}_t + \boldsymbol{\epsilon}_t$$

- Add linear transformations to nonlinearities

$$f_i(\mathbf{b}_i\mathbf{x}_t) = \tanh(\mathbf{b}_i\mathbf{x}_t) + \alpha_i\mathbf{b}_i\mathbf{x}_t + \beta_i$$



- Alphas and betas are not learned, but set to make learning the weights A,B,C easier

$$\mathbf{y}_t = \mathbf{A}\mathbf{f}\left(\mathbf{B}\mathbf{x}_t\right) + \mathbf{C}\mathbf{x}_t + \boldsymbol{\epsilon}_t$$

- Separate the nonlinear and linear problems by disabling linear dependencies from f

$$\sum_{t=1}^{T} f_i(\mathbf{b}_i \mathbf{x}_t) = 0 \qquad \sum_{t=1}^{T} f_i'(\mathbf{b}_i \mathbf{x}_t) = 0$$

by setting

$$\alpha_i = -\frac{1}{T}\sum_{t=1}^{T} \tanh'(\mathbf{b}_i \mathbf{x}_t) \qquad \beta_i = -\frac{1}{T}\sum_{t=1}^{T} \left[\tanh(\mathbf{b}_i \mathbf{x}_t) + \alpha_i \mathbf{b}_i \mathbf{x}_t\right]$$

- Compensate by changing C accordingly

$$\mathbf{C}_{\text{new}} = \mathbf{C}_{\text{old}} - \mathbf{A}(\boldsymbol{\alpha}_{\text{new}} - \boldsymbol{\alpha}_{\text{old}})\mathbf{B} - \mathbf{A}(\boldsymbol{\beta}_{\text{new}} - \boldsymbol{\beta}_{\text{old}})\,[0 \;\; 0 \ldots 1]$$

# Theoretical Motivation

- Fisher information matrix becomes more diagonal
- Standard gradient becomes closer to natural gradient

| | A | B | C |
|---|---|---|---|
| A | $\begin{cases} 0 & i' \neq i \\ -\frac{1}{\sigma_i^2} \sum_t f_j(\mathbf{b}_j \mathbf{x}_t) f_{j'}(\mathbf{b}_{j'} \mathbf{x}_t) & i' = i \end{cases}$ | $-\frac{1}{\sigma_i^2} a_{ij'} \sum_t f_j(\mathbf{b}_j \mathbf{x}_t) f'_{j'}(\mathbf{b}_{j'} \mathbf{x}_t) x_{kt}$ | $\begin{cases} 0 & i' \neq i \\ -\frac{1}{\sigma_i^2} \sum_t f_j(\mathbf{b}_j \mathbf{x}_t) x_{kt} & i' = i \end{cases}$ |
| B | $-\frac{1}{\sigma_i^2} a_{ij'} \sum_t f_j(\mathbf{b}_j \mathbf{x}_t) f'_{j'}(\mathbf{b}_{j'} \mathbf{x}_t) x_{kt}$ | $-\sum_i \frac{1}{\sigma_i^2} a_{ij} a_{ij'} \sum_t f'_j(\mathbf{b}_j \mathbf{x}_t) f'_{j'}(\mathbf{b}_{j'} \mathbf{x}_t) x_{kt} x_{k't}$ | $-\frac{1}{\sigma_i^2} a_{ij} \sum_t f'_j(\mathbf{b}_j \mathbf{x}_t) x_{kt} x_{k't}$ |
| C | $\begin{cases} 0 & i' \neq i \\ -\frac{1}{\sigma_i^2} \sum_t f_j(\mathbf{b}_j \mathbf{x}_t) x_{kt} & i' = i \end{cases}$ | $-\frac{1}{\sigma_i^2} a_{ij} \sum_t f'_j(\mathbf{b}_j \mathbf{x}_t) x_{kt} x_{k't}$ | $\begin{cases} 0 & i' \neq i \\ -\frac{1}{\sigma_i^2} \sum_t x_{kt} x_{k't} & i' = i \end{cases}$ |

# Implementation Details

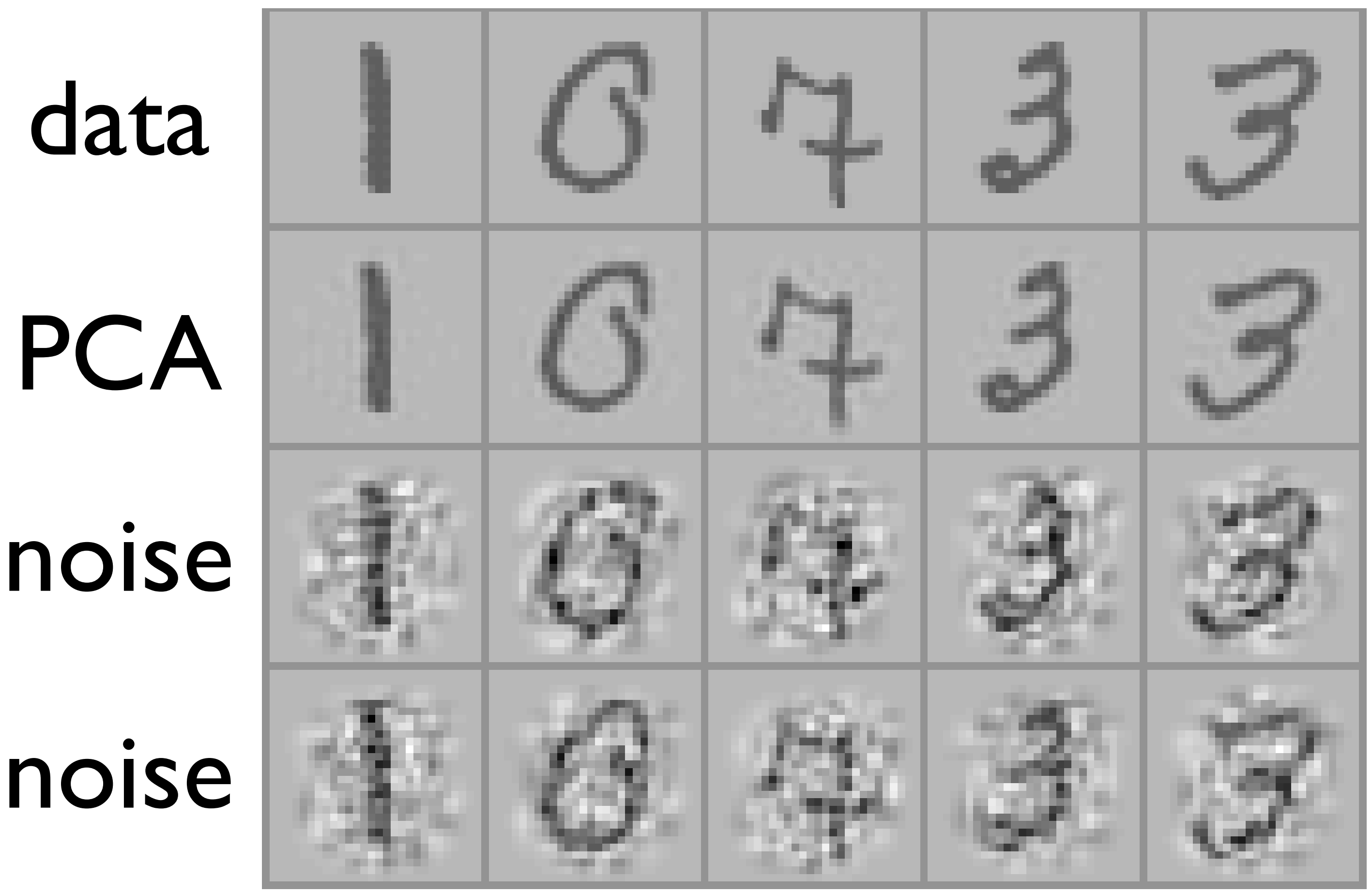
- Learning algorithm: Stochastic gradient
- Mini-batch size 1000, momentum 0.9
- Transformations done initially and after every 1000 iterations
- Soft-max for discrete outputs
- Normalized random initialization, shortcut weights to zero
- Learning rate decreased linearly in the second half of learning time
- Regularization: PCA in classification, weight decay, added noise to inputs

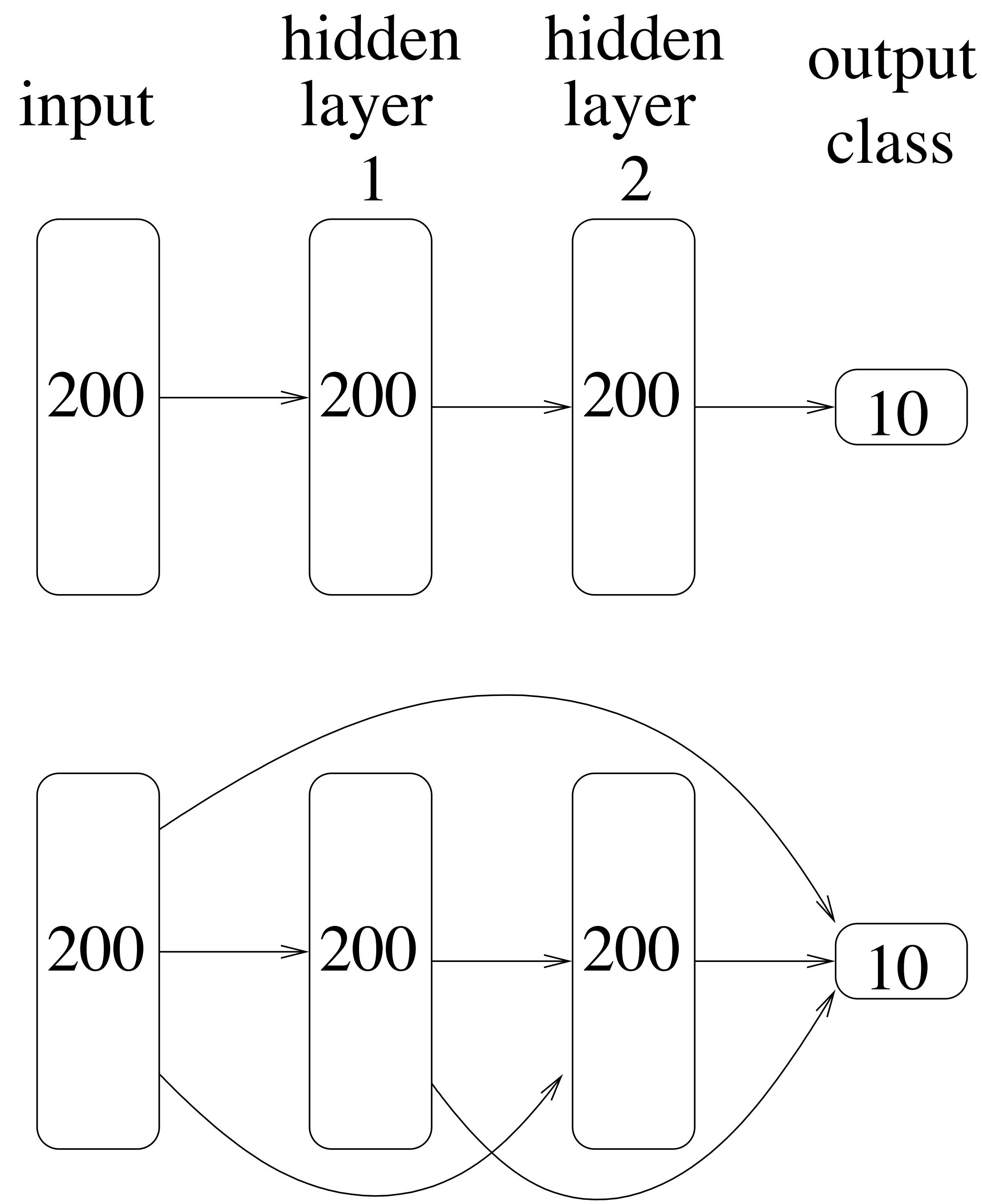
# Experiments

- MNIST Classification
- CIFAR-10 Classification
- MNIST Autoencoder

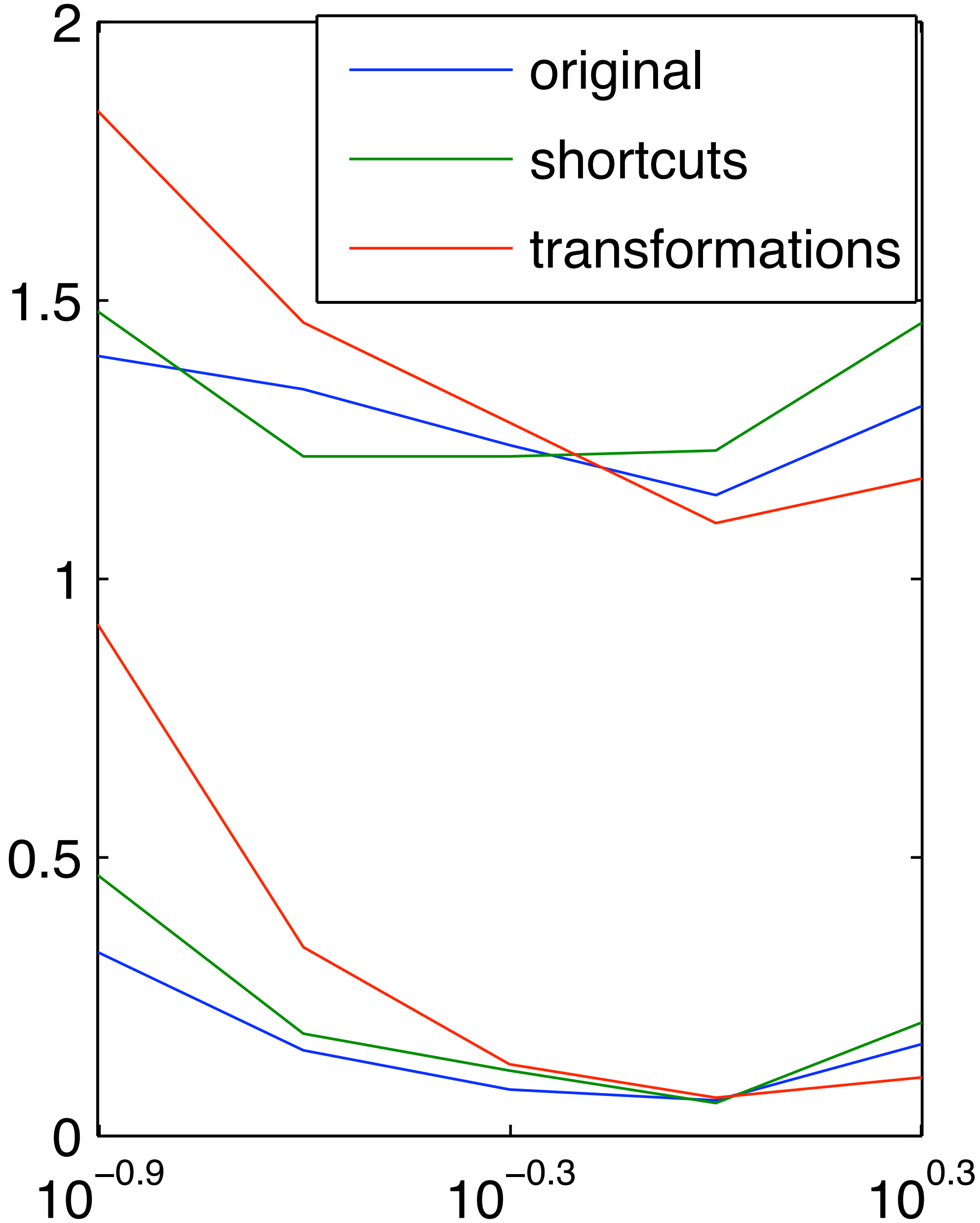
- Image data, but nothing image-specific

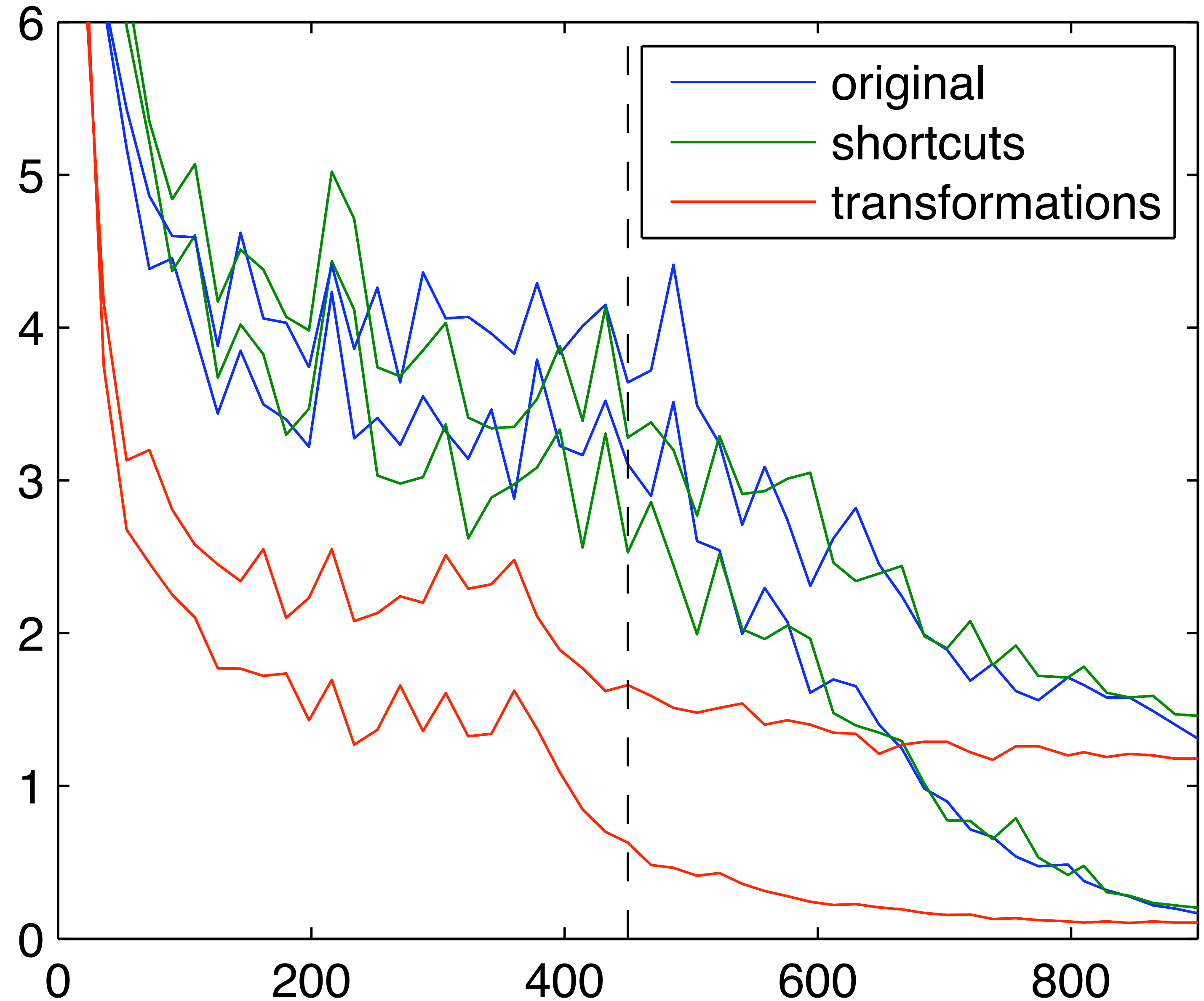
# MNIST Classification



# MNIST Classification



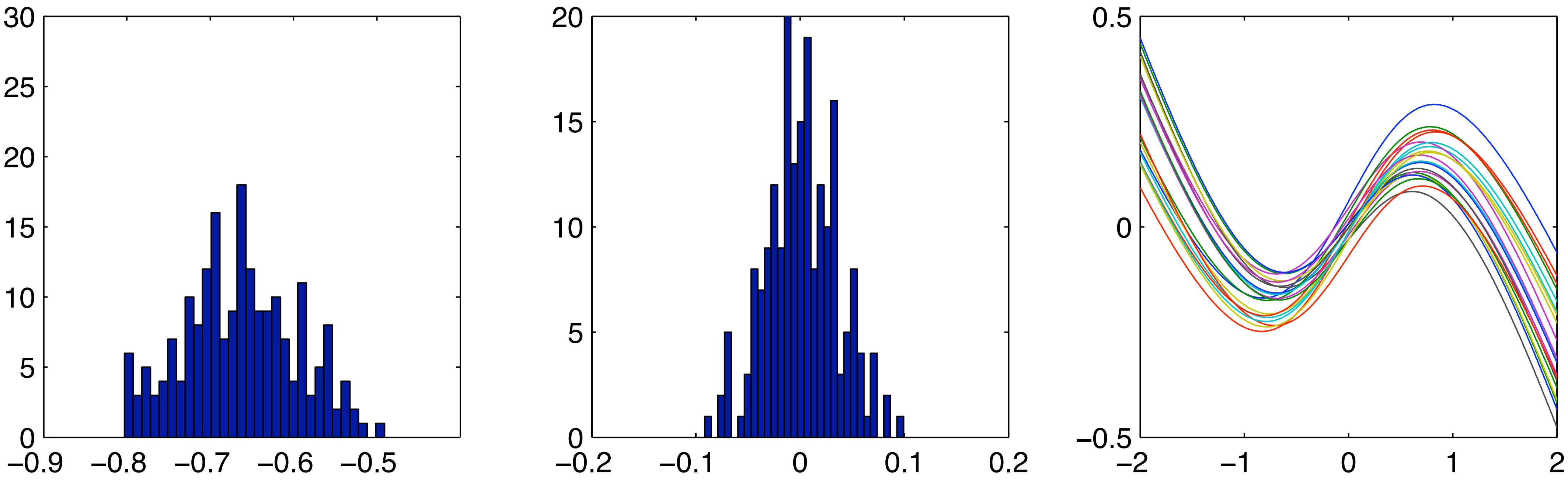

Error against learning rate Error against learning time

Training (lower) and test errors (higher)

# MNIST Classification

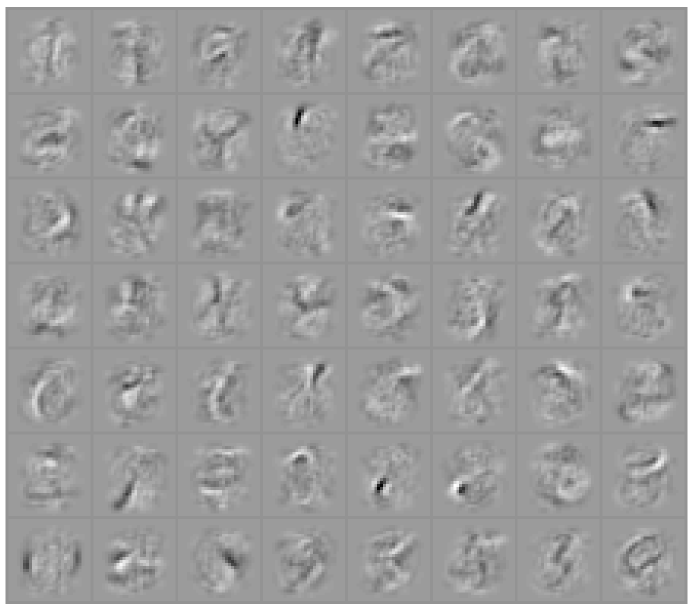
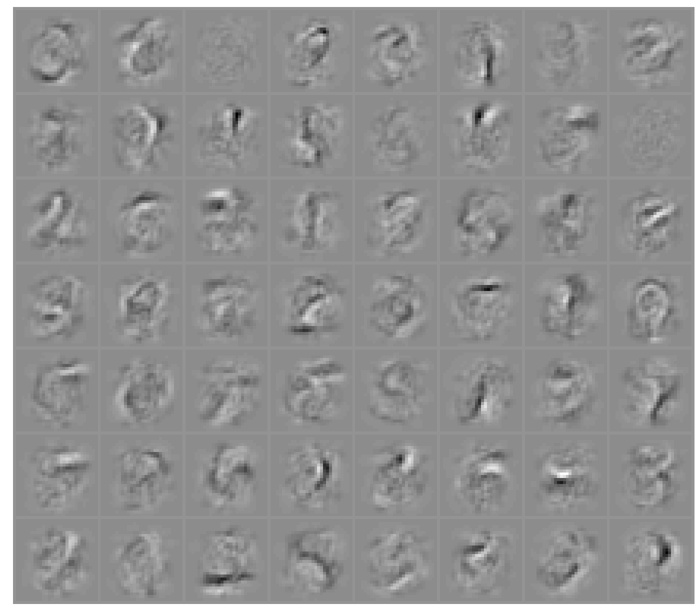
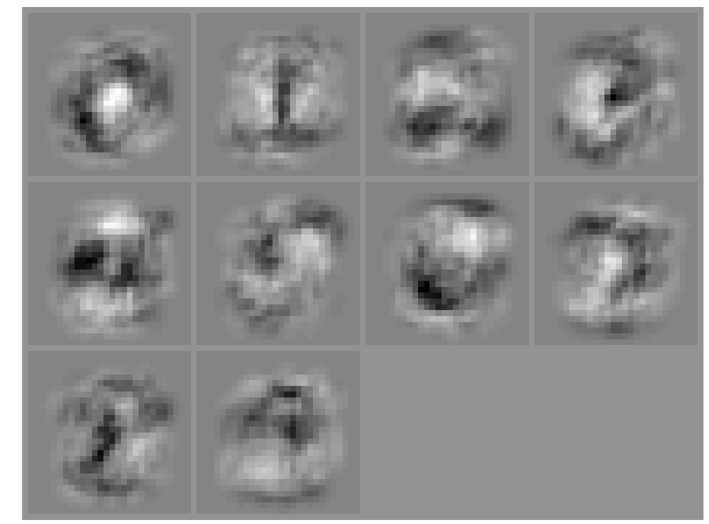
- Test errors after 15 minutes as regularization methods are included:

| regularization | none | weight decay | PCA | noise | (150 minutes) |
|---|---|---|---|---|---|
| original | 1.87 | 1.85 | 1.62 | 1.15 | 1.03 |
| shortcuts | 2.02 | 1.77 | 1.59 | 1.23 | 1.17 |
| transform. | 1.63 | 1.56 | 1.56 | 1.10 | **1.02** |




Histograms of $\alpha_i$ and $\beta_i$ in the first hidden layer. Examples of $f_i(\cdot)$.

# MNIST Classification

- Visualization of learned weights to randomly chosen hidden units on layers 1 and 2, and to the class outputs 0,1,...,9

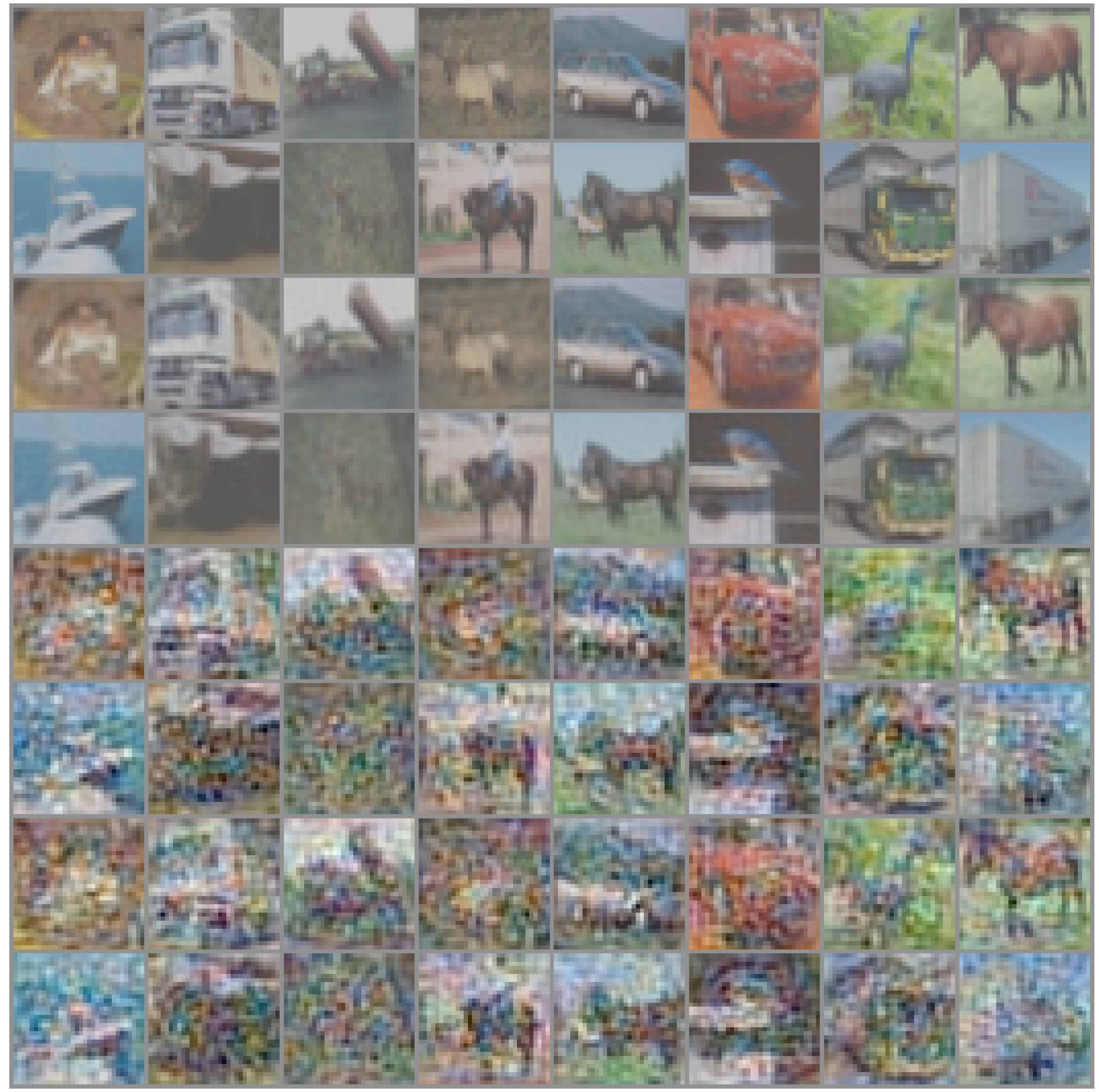
# CIFAR-10 Classification

original data

after PCA to 500

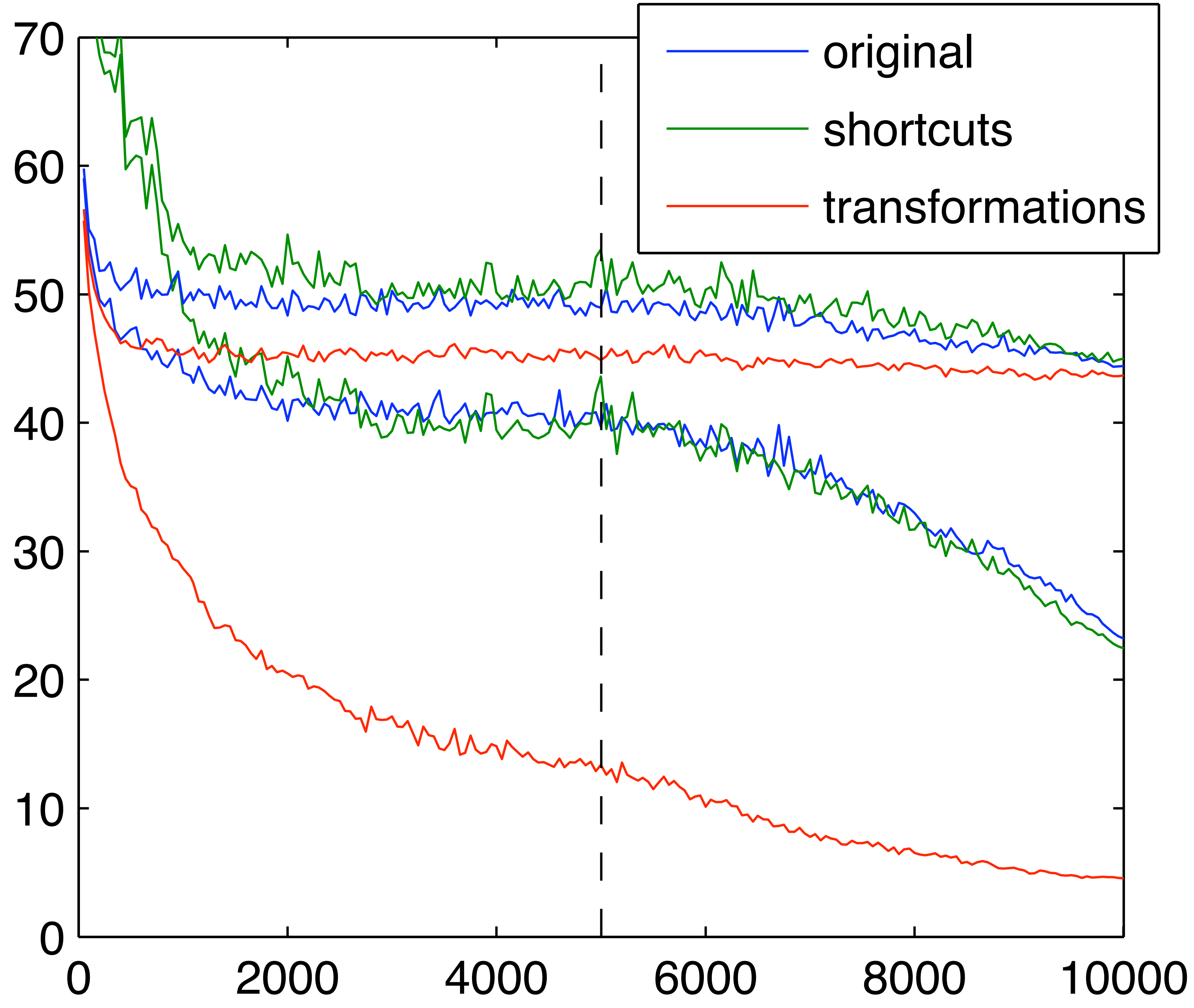
with noise

with noise



- 500-500-500-10 network

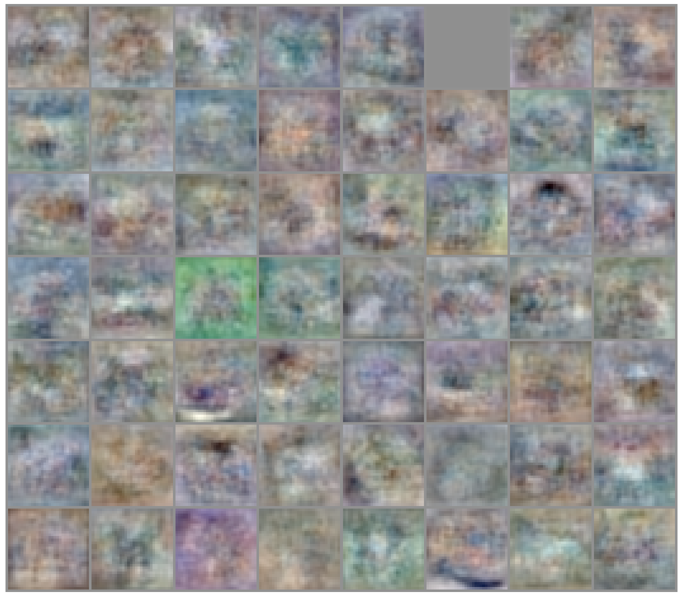
# CIFAR-10 Classification

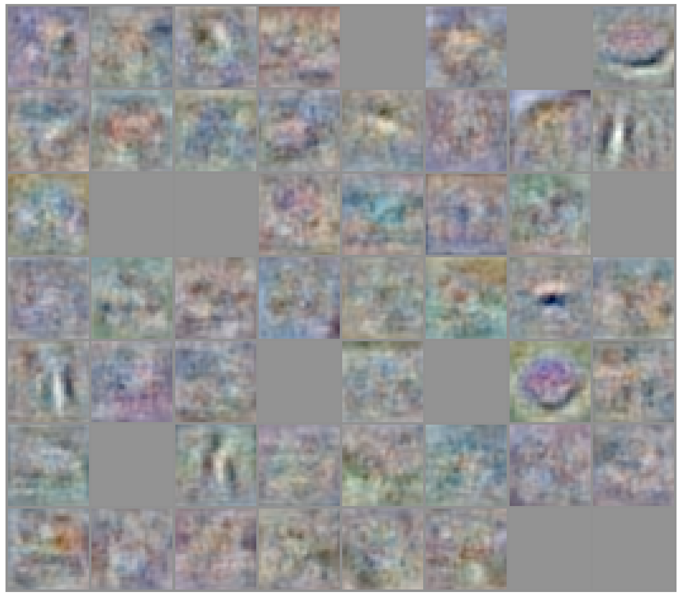


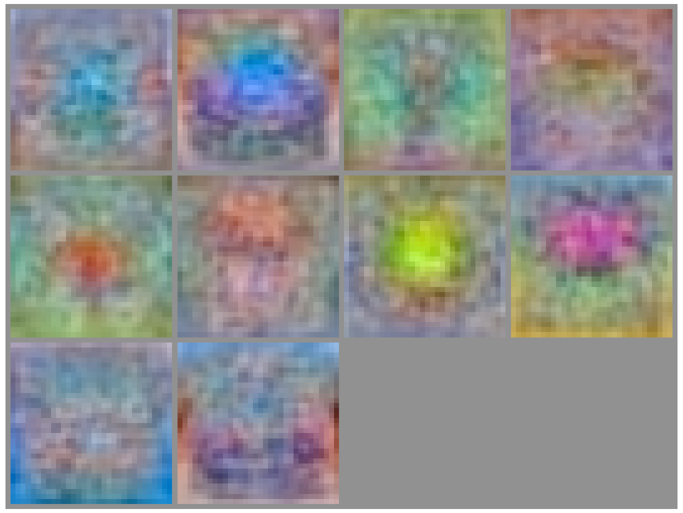

Classification error against learning time

# CIFAR-10 Classification

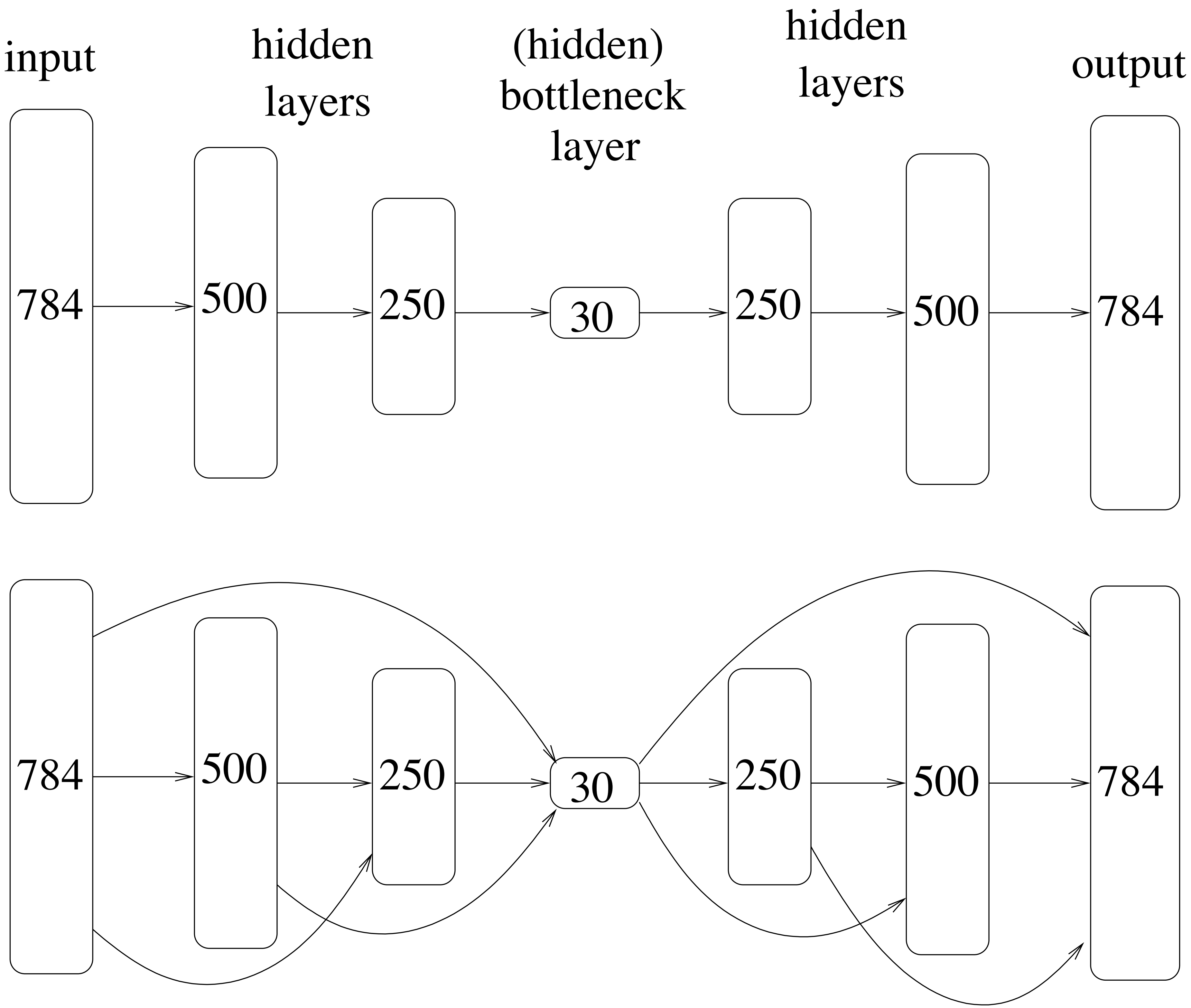
| **Classification %** | linear | original | shortcuts | transf. | Krizhevsky (2009) |
|---|---|---|---|---|---|
| Training error | 58.07 | 23.21 | 22.46 | 4.56 | |
| Test error | 59.09 | 44.42 | 44.99 | **43.70** | 48.47 |

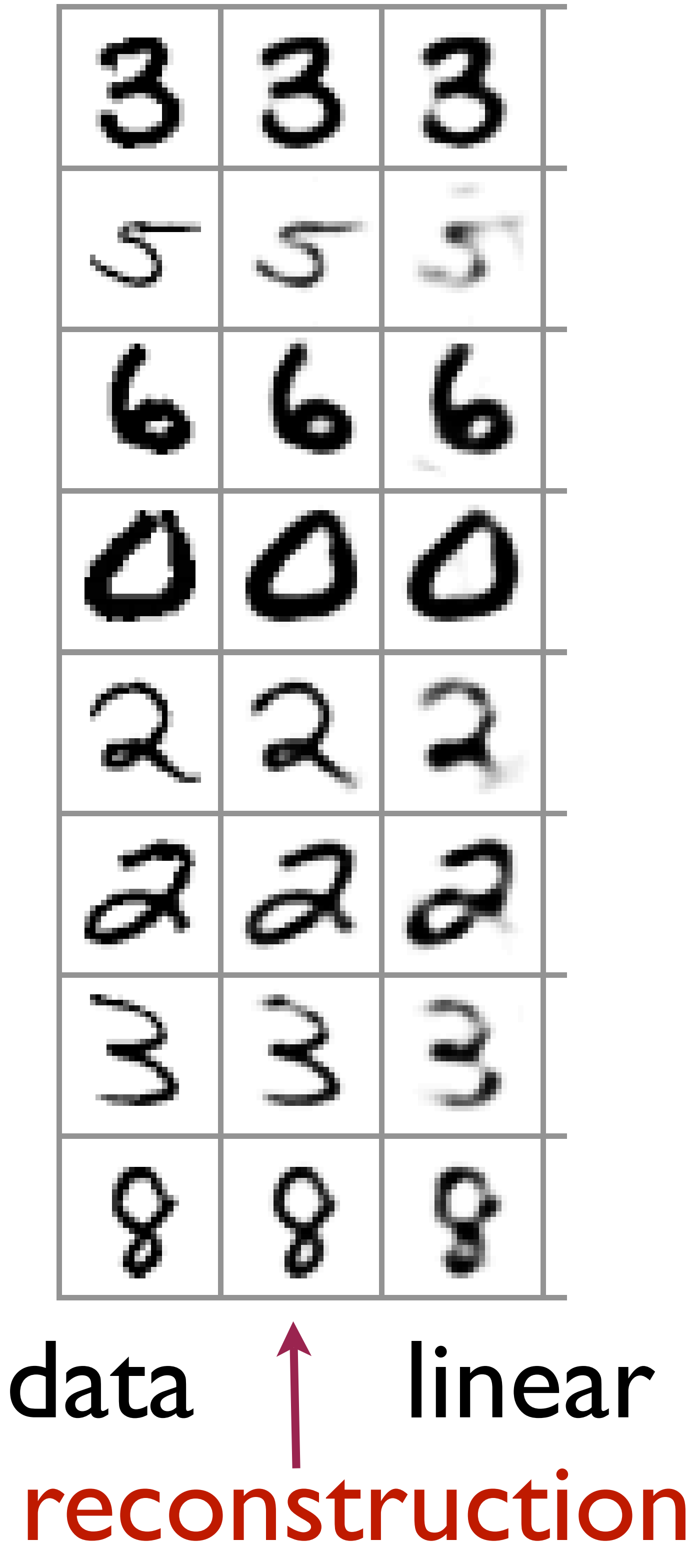




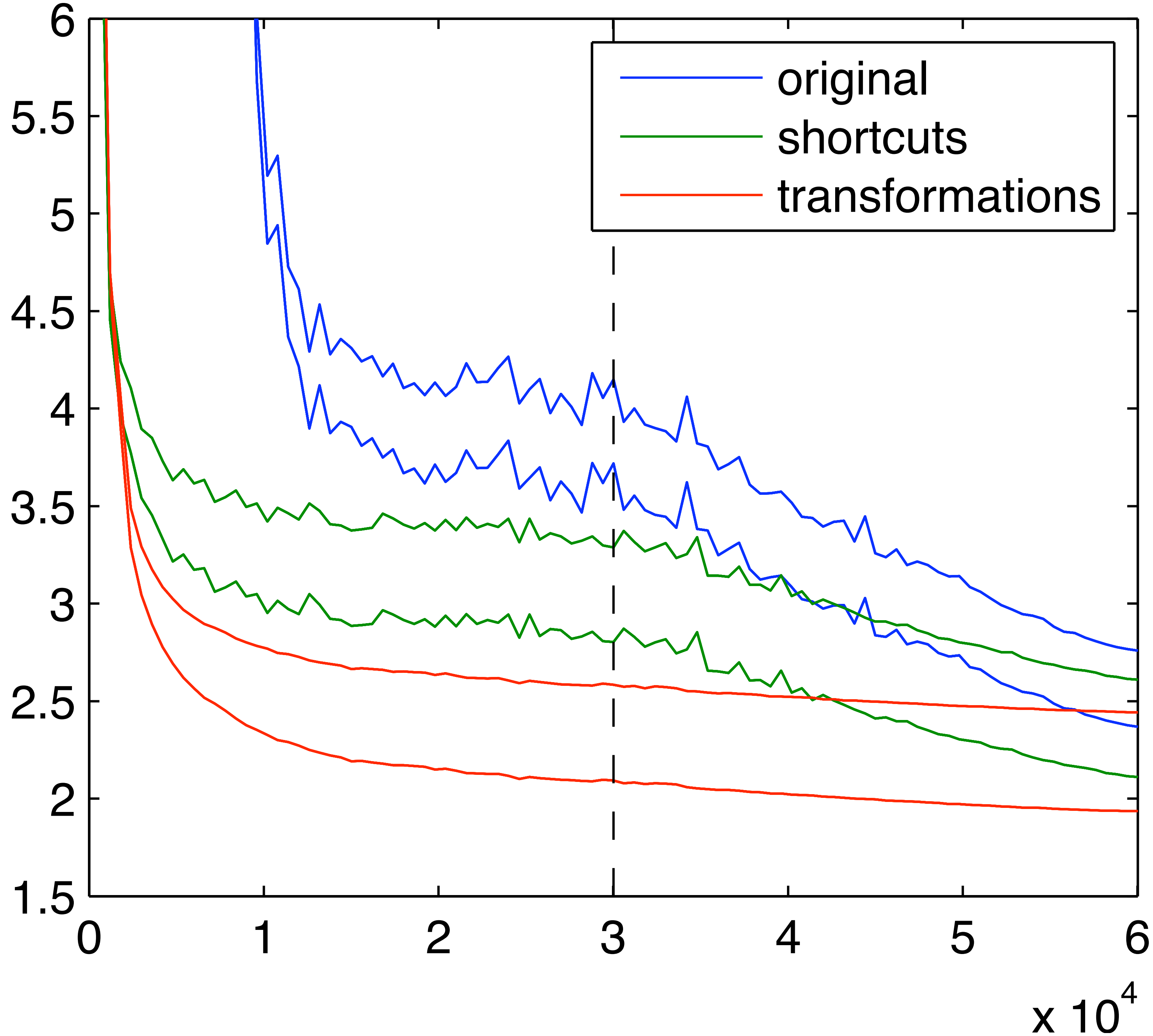


# MNIST Autoencoder



# MNIST Autoencoder

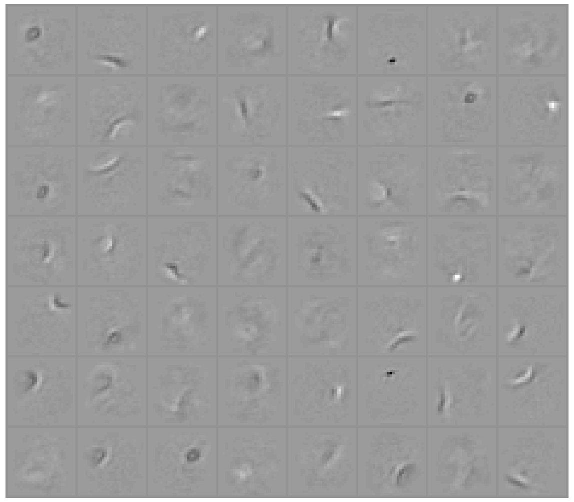



Reconstruction error against learning time

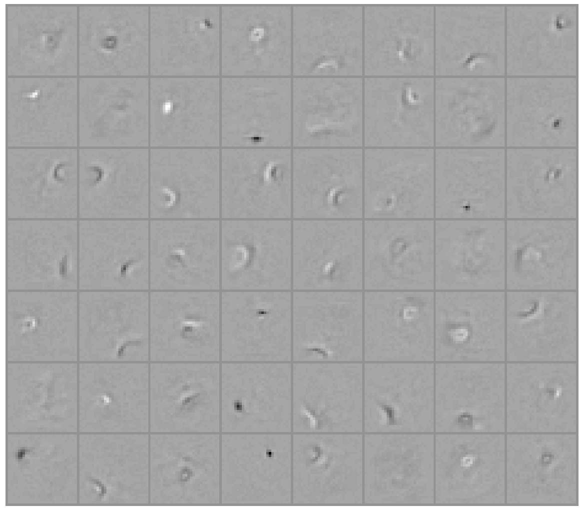
# MNIST Autoencoder

| | linear | original | shortcuts | transf. | Martens (2010) |
|---|---|---|---|---|---|
| training error | 8.11 | 2.37 | 2.11 | 1.94 | 1.75 |
| test error | 7.85 | 2.76 | 2.61 | **2.44** | 2.55 |
| # of iterations | 92k | 49k | 38k | 37k | ? |

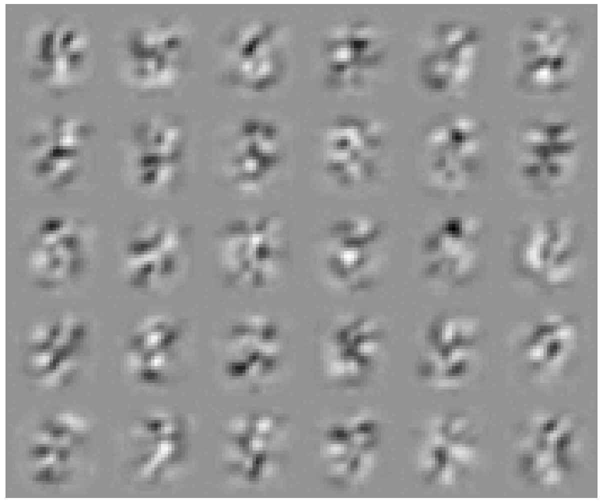
x-h1



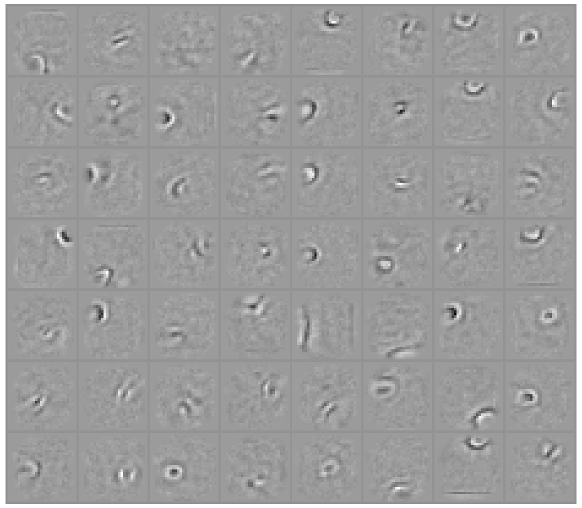
x-h2



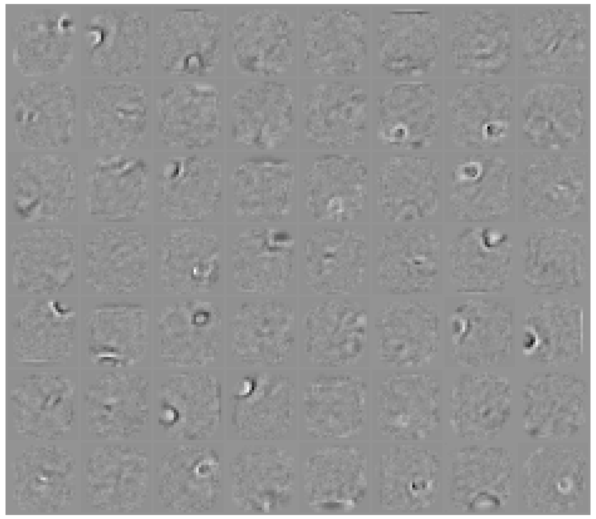
x-h3



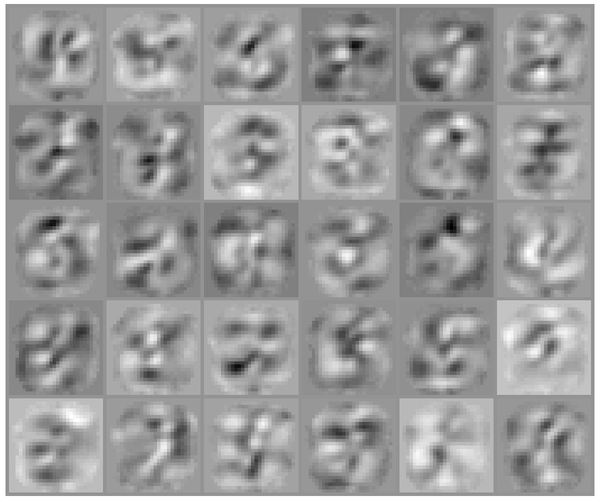
h5-y



h4-y



h3-y



# Discussion

- Simple transformations make basic gradient competitive with state-of-the-art
- Making parameters more independent will also help variational Bayes and MCMC
- Could be initialized with unsupervised pre-training for further improvement
- How about doing classification and autoencoder as a multitask?