Feature Learning and Mapping using Deep Learning Approaches

Hamid Mohammadi

CSLU Seminar

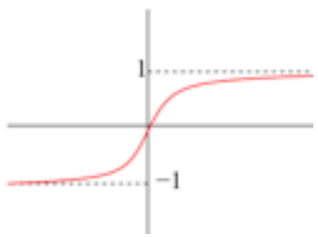
2015-02-03

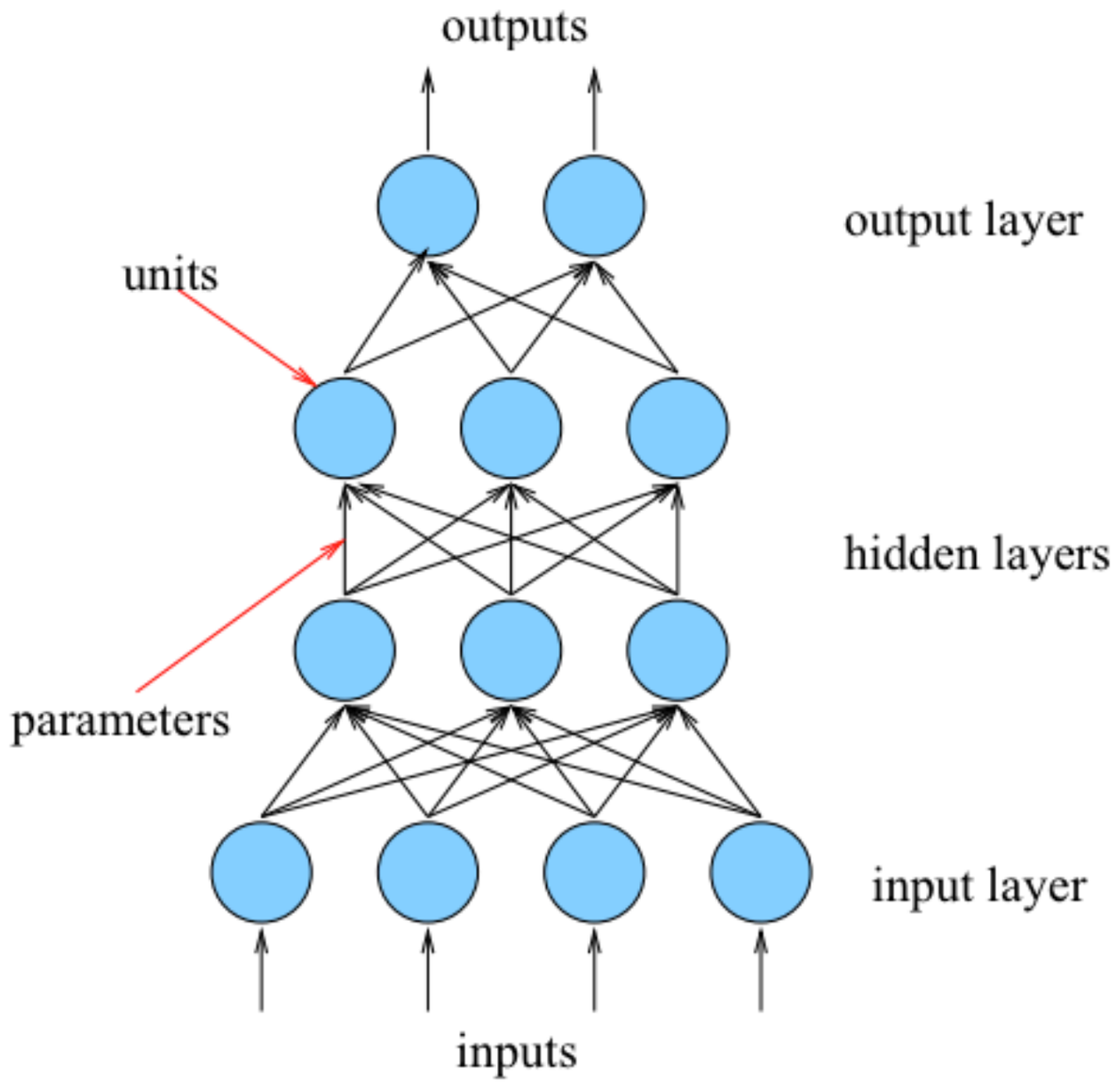
Outline

- Artificial Neural Networks (ANNs)
- MNIST classification task
- Unsupervised Feature Learning
  - various ANN architectures
  - Evaluations
- Supervised Feature Mapping
  - various ANN architectures
  - Evaluations

ANN Architecture

- ANN is composed of multiple layers
- Layers perform non-linear transformations
  - `y=g(Wx+b)`

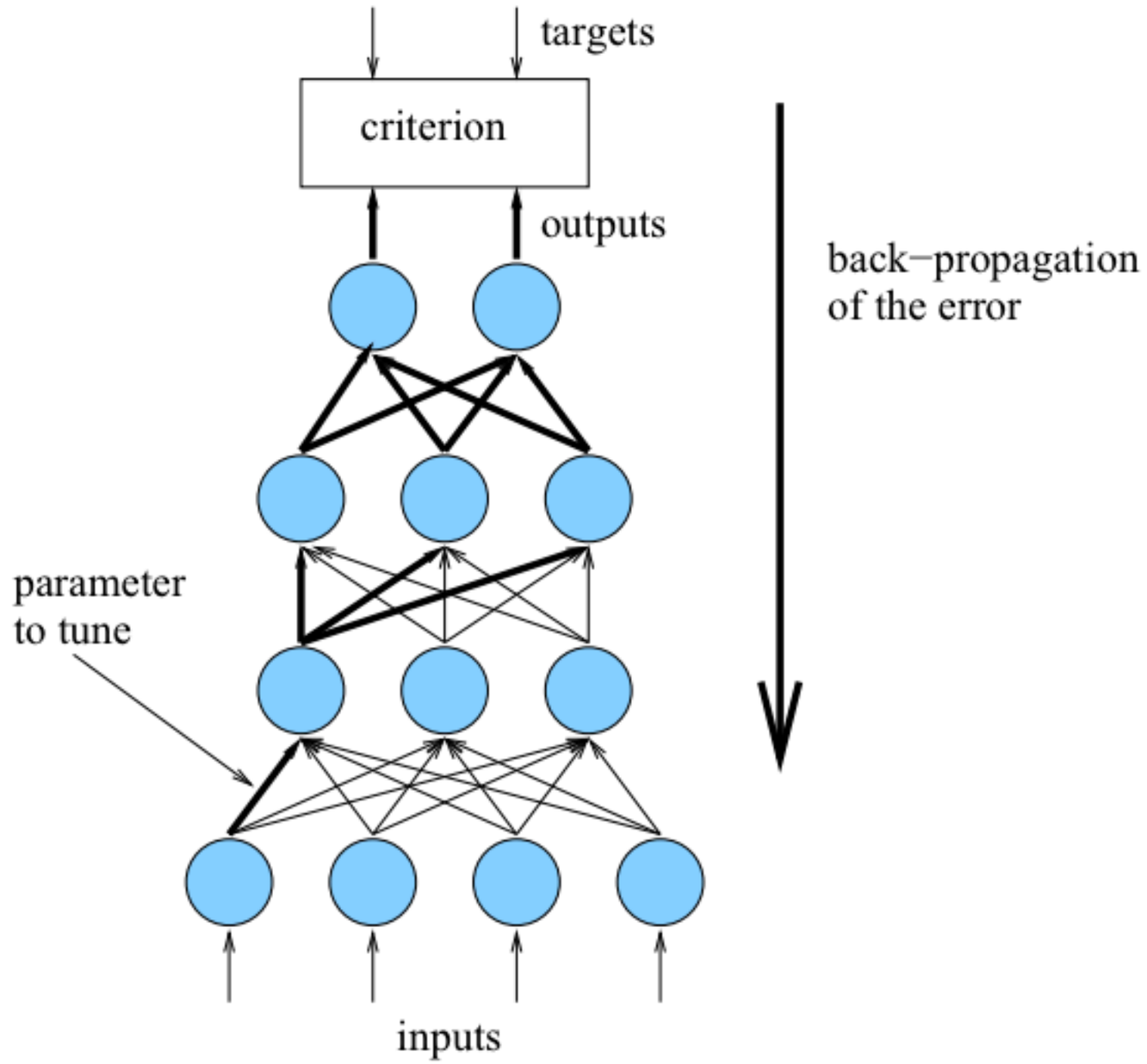




http://bengio.abracadoudou.com/lectures/old/tex_ann.pdf

Backpropagation

- Estimating model Parameters `W`s and `b`s



http://bengio.abracadoudou.com/lectures/old/tex_ann.pdf

Backpropagation

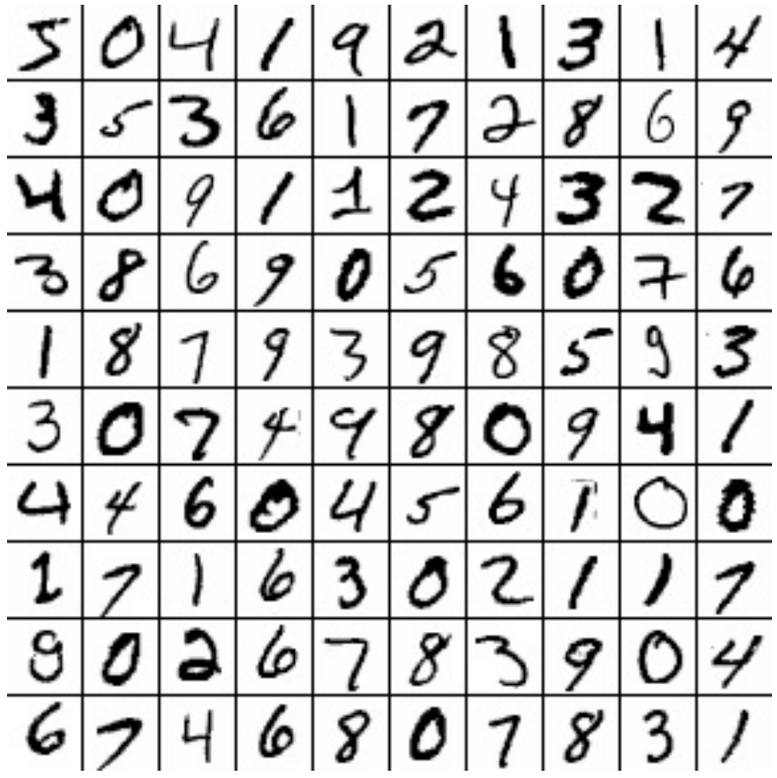
- Criterion for ANN
  - Mean Squared Error:
    - Error=(ŷ-y)^2
  - Cross-entropy
    - Error=-sum(ŷ*log*(y)+
      (1-ŷ)*log*(1-y))

Backpropagation

- Regularization:
  - L1: `Criterion = CE + |W|`
  - L2: `Criterion = CE + W^2`
  - Dropout: randomly omitting subsets of features at each iteration with probability `p=[0.0,0.5]`

MNIST Corpus

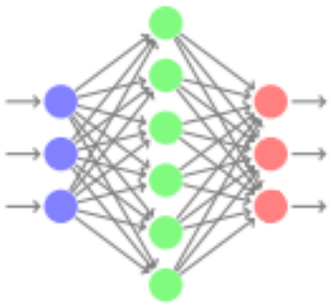
- 28x28 pixels, pixel values range from 0 to 1
- Contains 70,000 images
  - 50,000 training set
  - 10,000 validation set
  - 10,000 test set

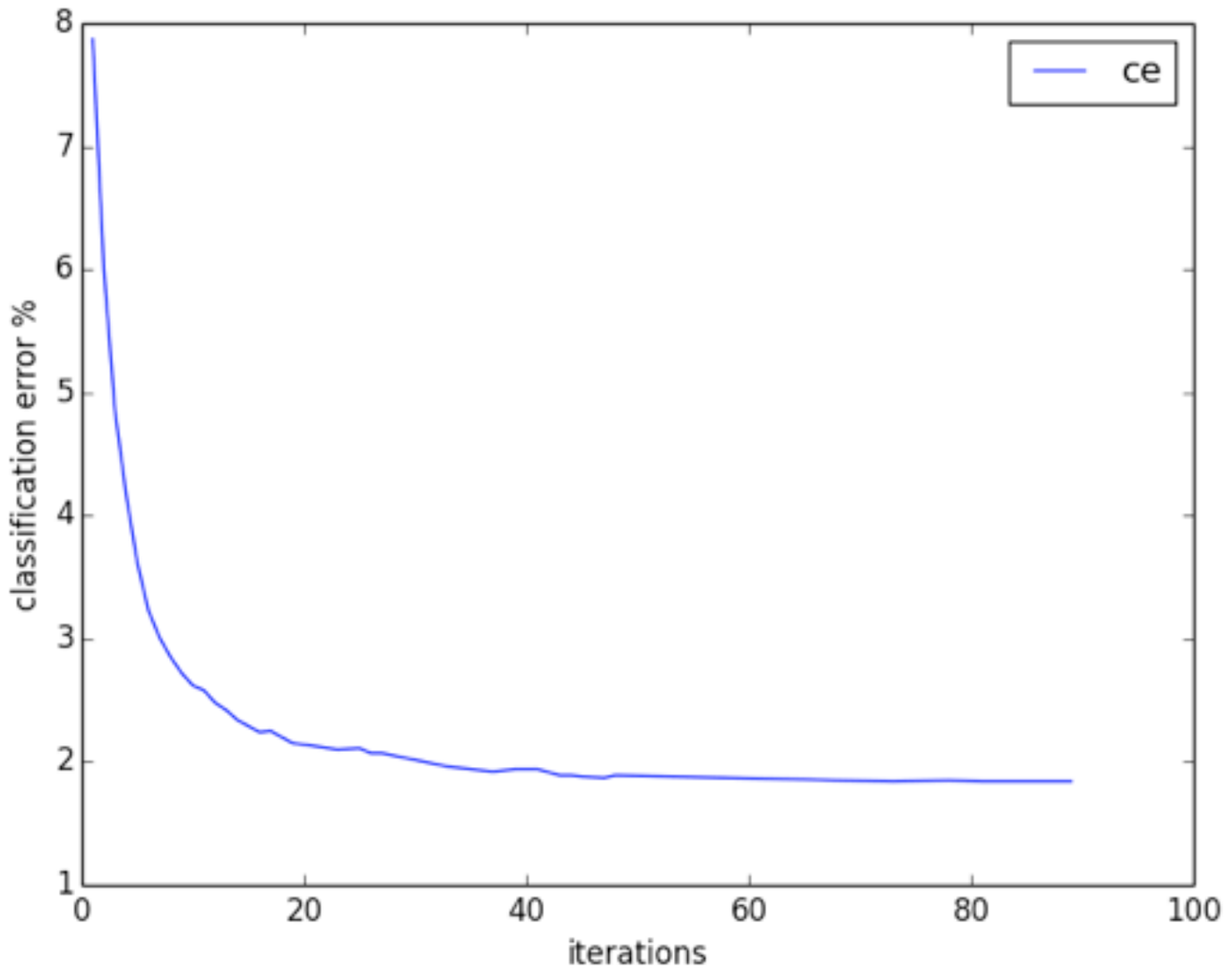


- Task: Classify 10 digit classes

ANN for MNIST

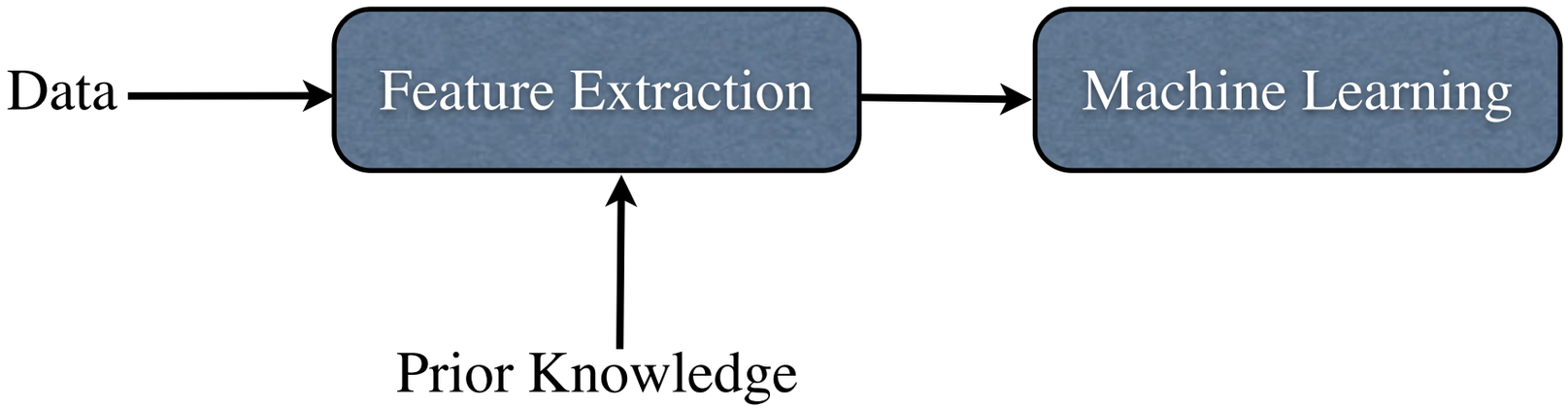
- Two-layer ANN with 100 hidden units





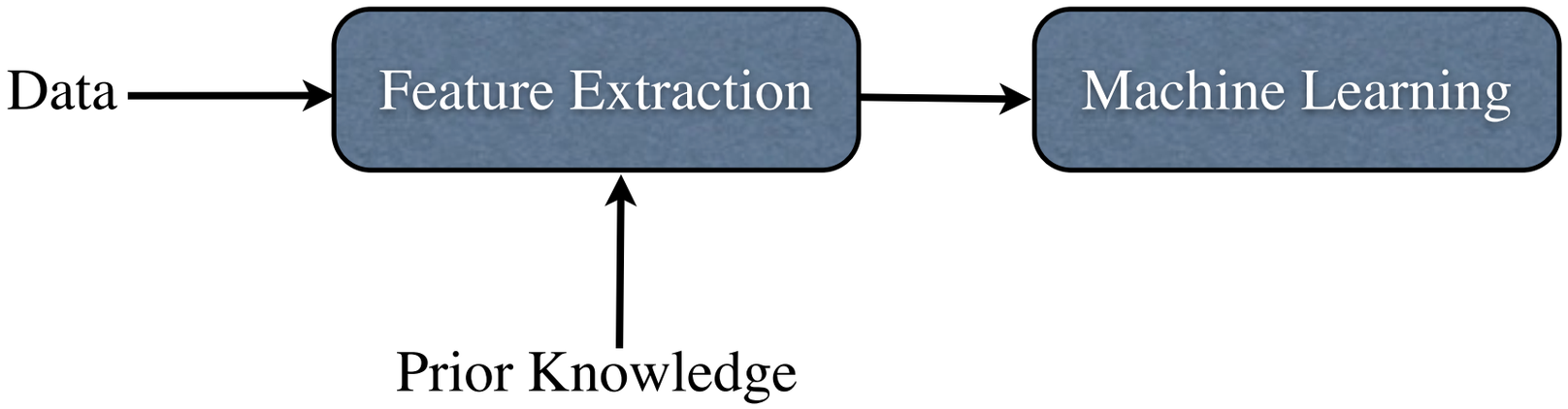
Feature Learning

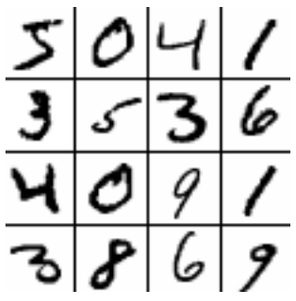
- Usual Machine Learning applications have two steps



Feature Learning

- Usual Machine Learning applications have two steps



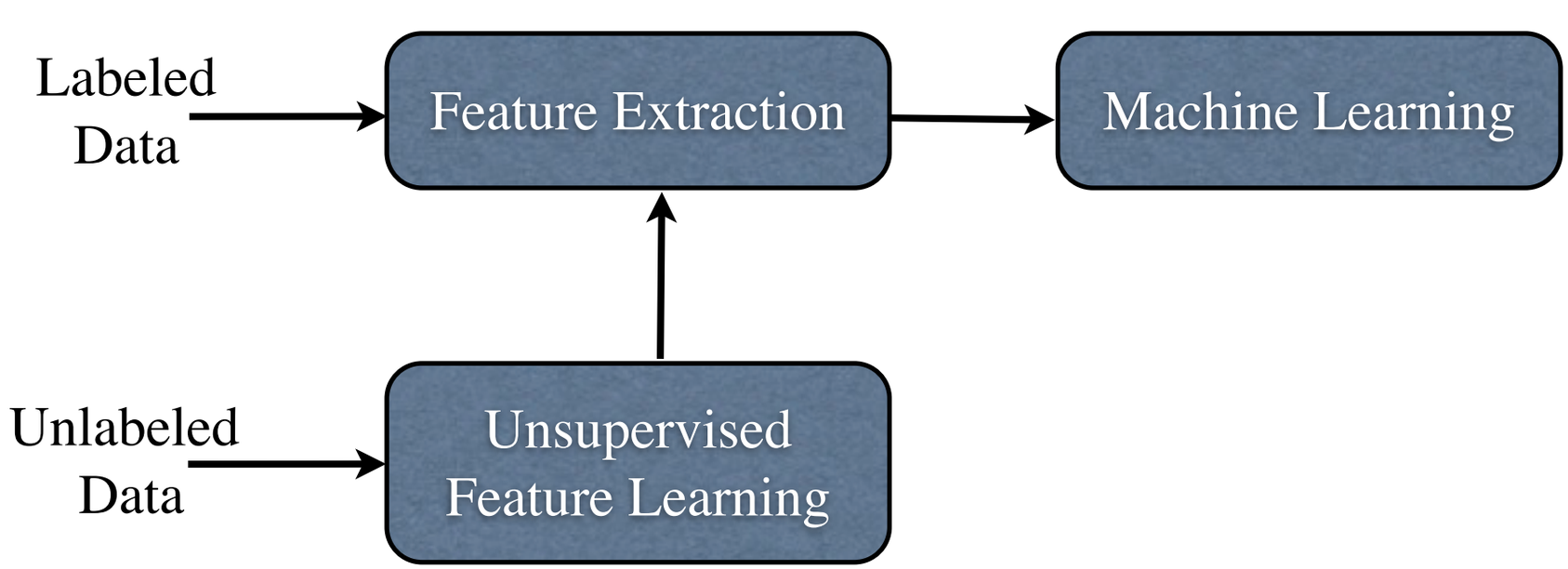


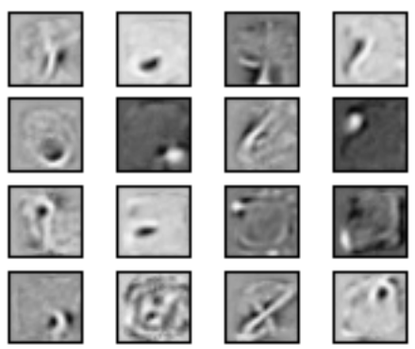
how many straight vertical lines? how long?

how many circles?

how many straight horizontal lines? how long?

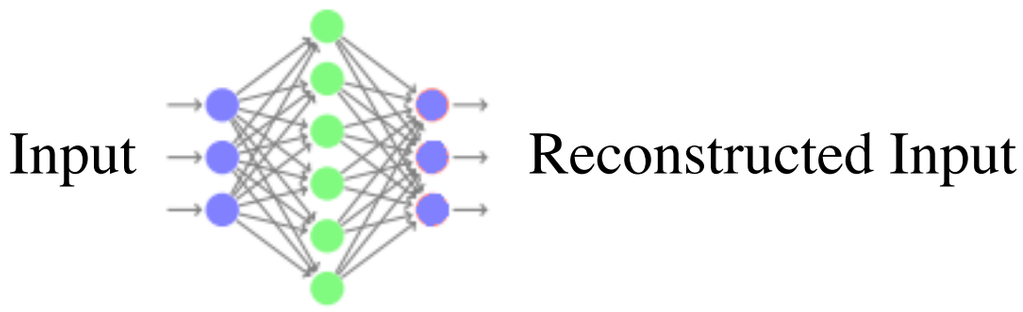
Feature Learning





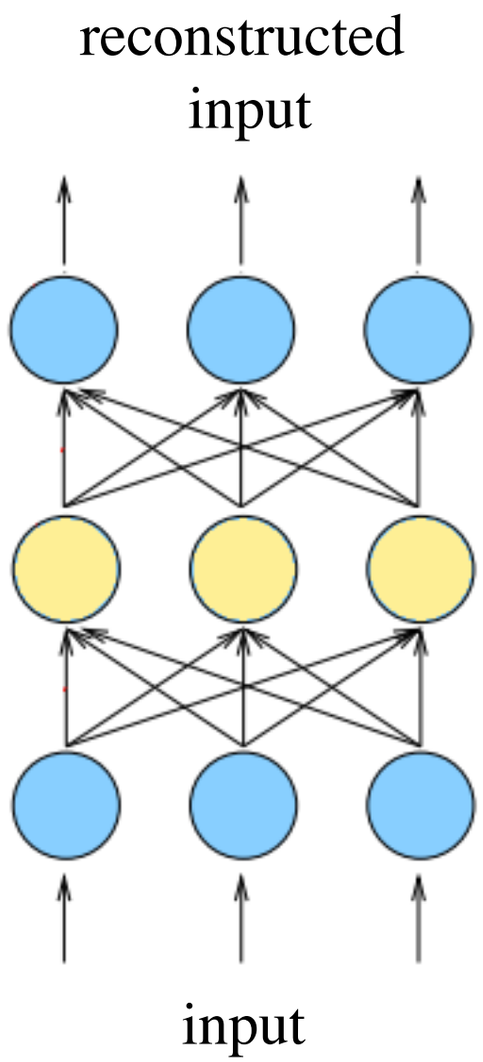
ANN for Feature Learning

- How to use ANNs for Feature Learning?
- ONE solution is to use networks called Autoencoders
- These networks try to reconstruct the input
- They are unsupervised, no labels needed



Autoencoders

- These networks try to reconstruct the input
- The model's equation:
  `x_rec=g(W'g(Wx+b)+b')`
- where `W'=transpose(W)` (tied weights)

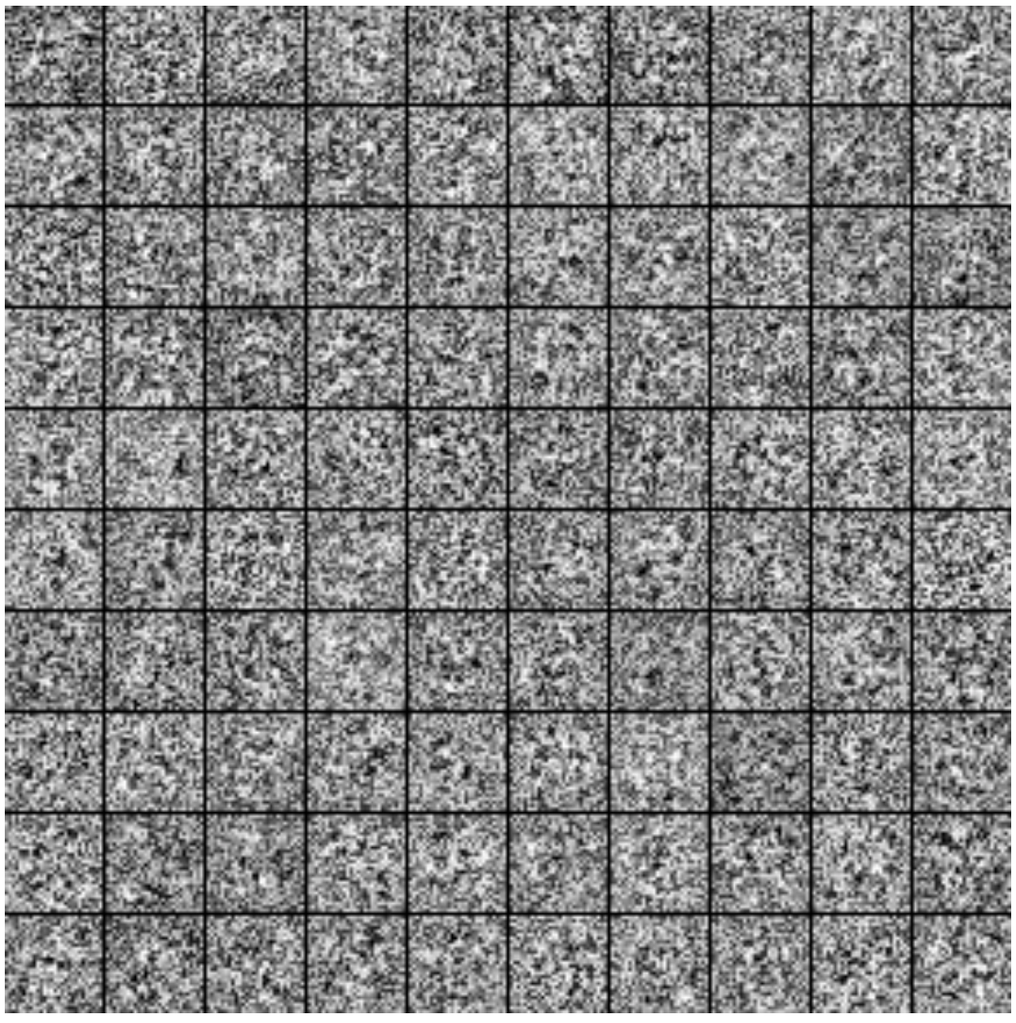


Autoencoders

- Implementation:
  - Python 2.7
  - Theano 0.6

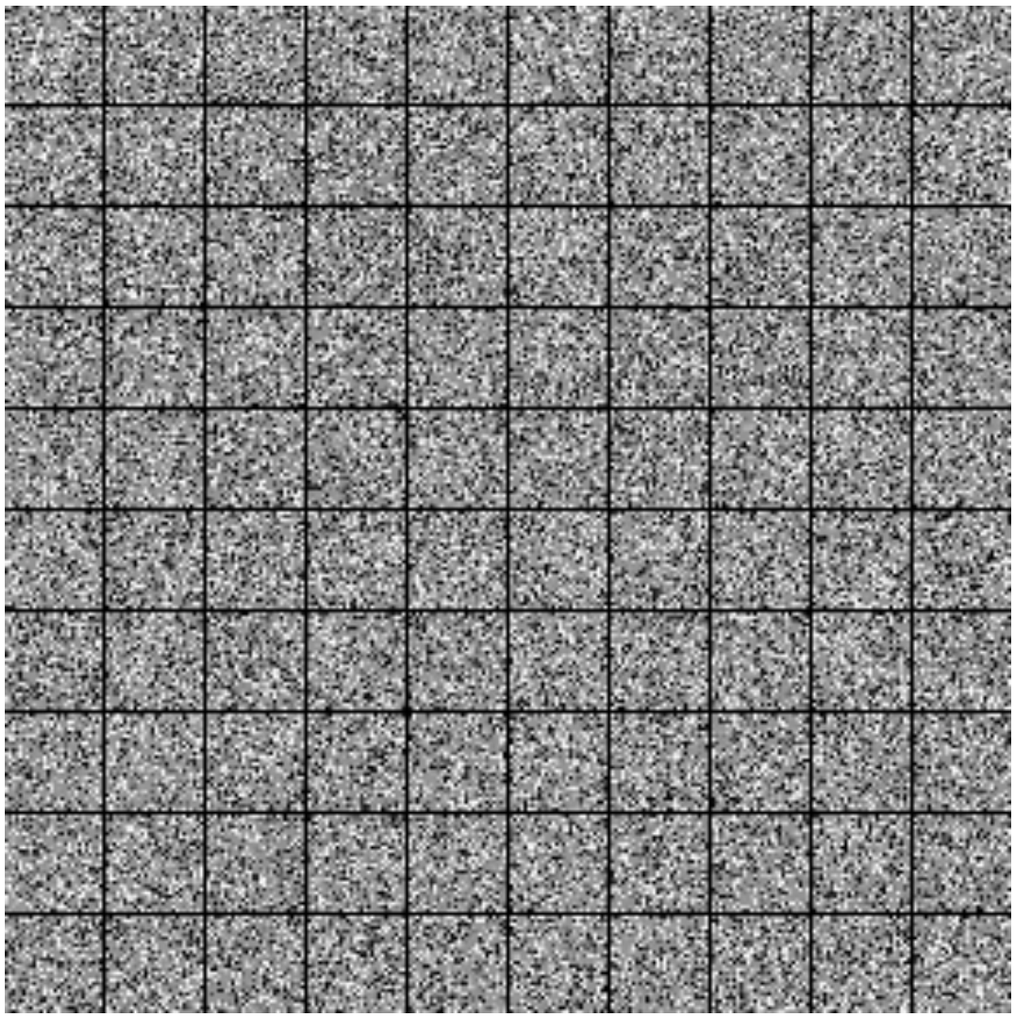
Autoencoders

- AE-1000 weights (1000 hidden units)



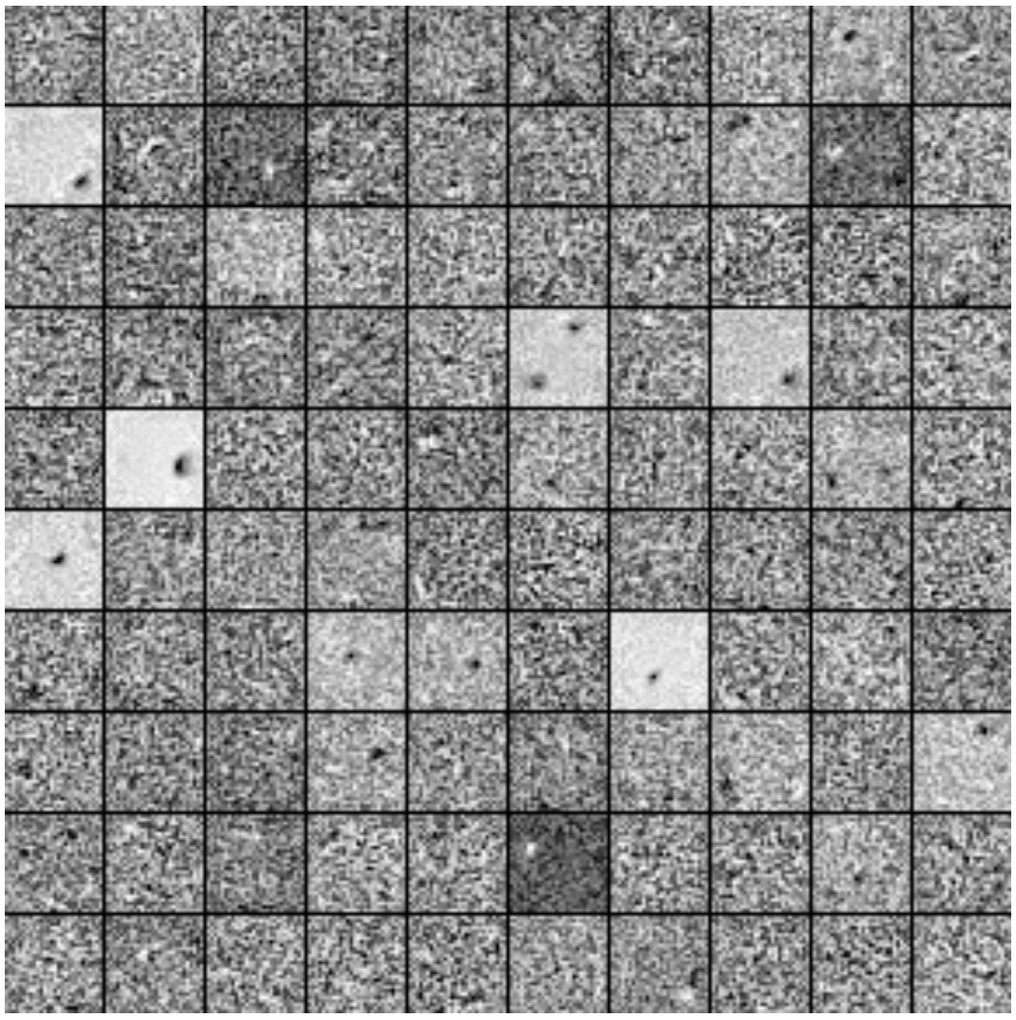
AE Regularization

- AE-1000-L1 weights



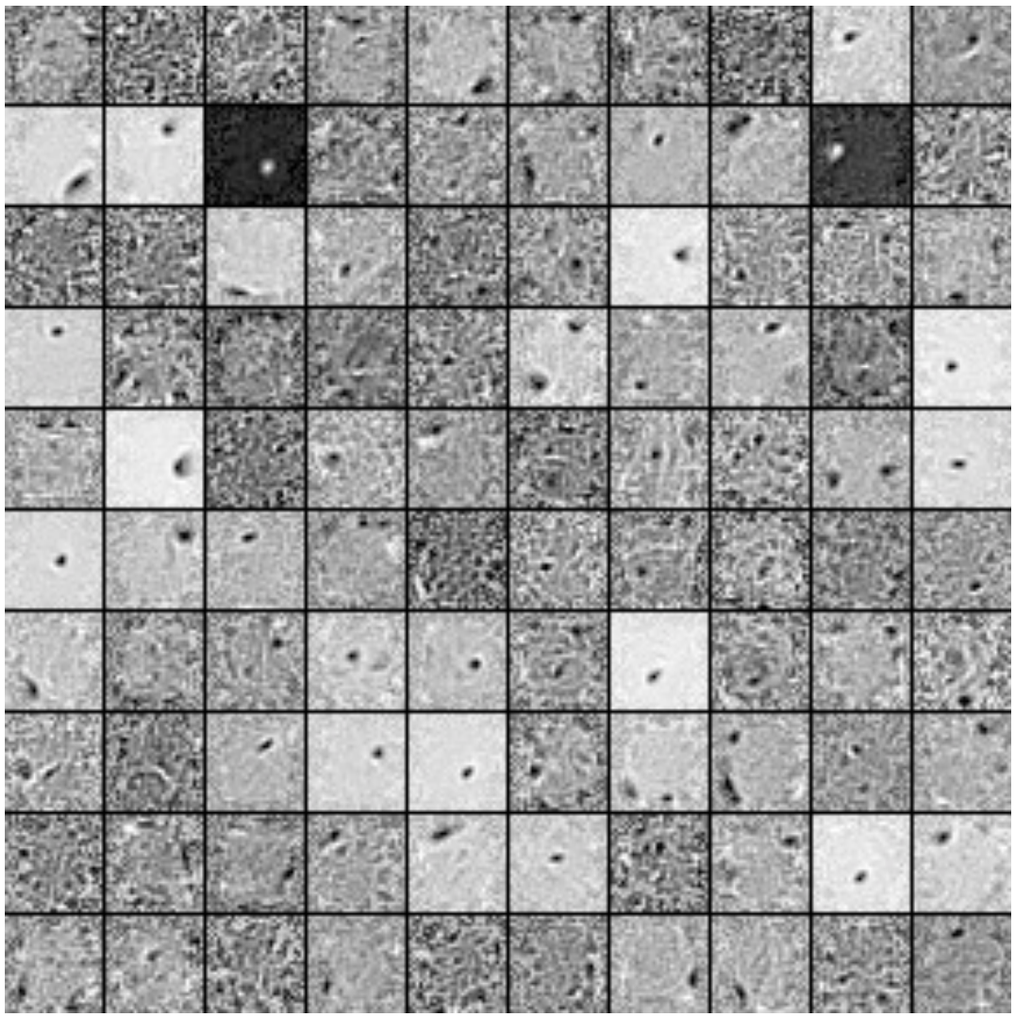
AE Regularization

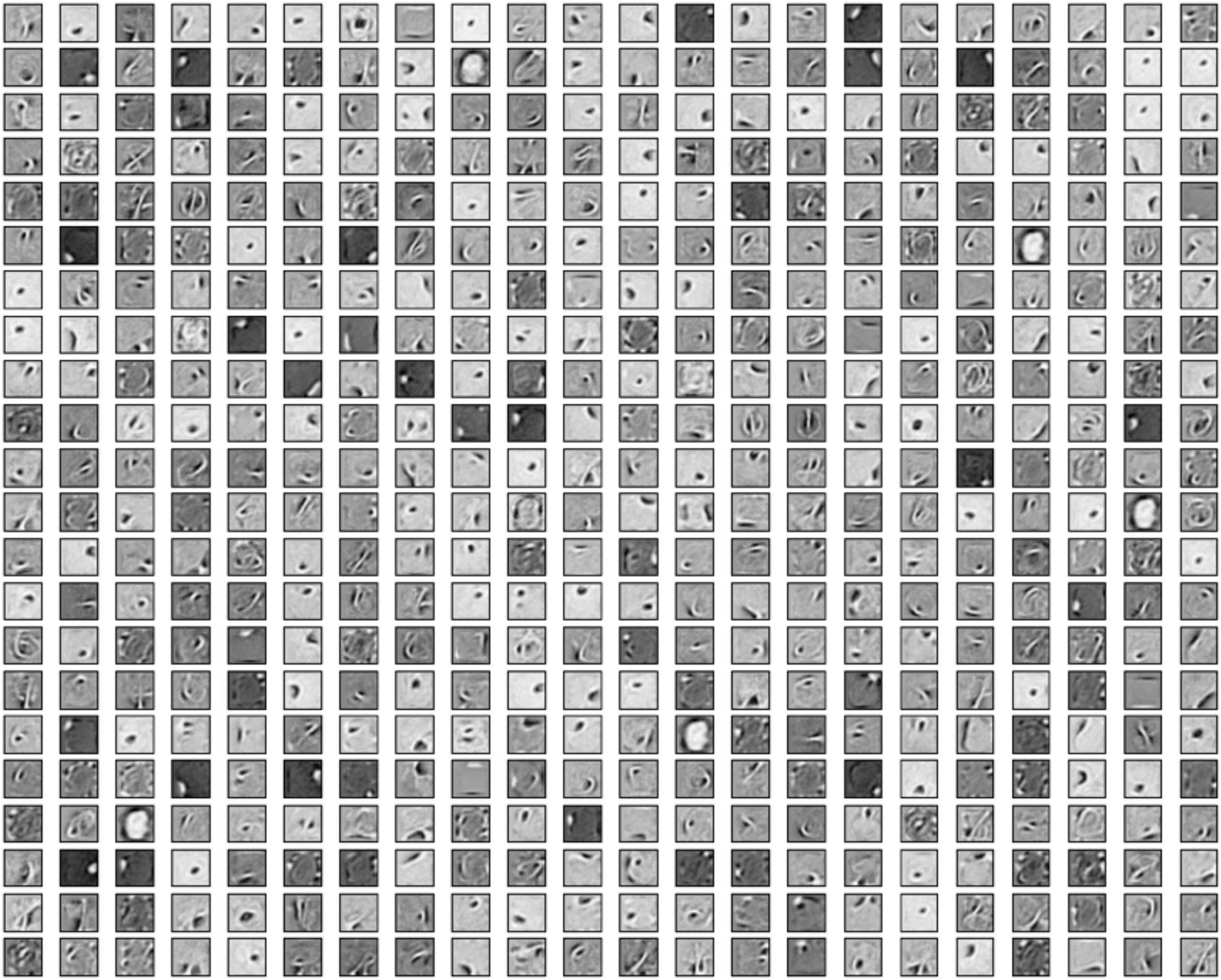
- AE-1000-L2 weights



AE Regularization

- AE-1000-Dropout-0.1

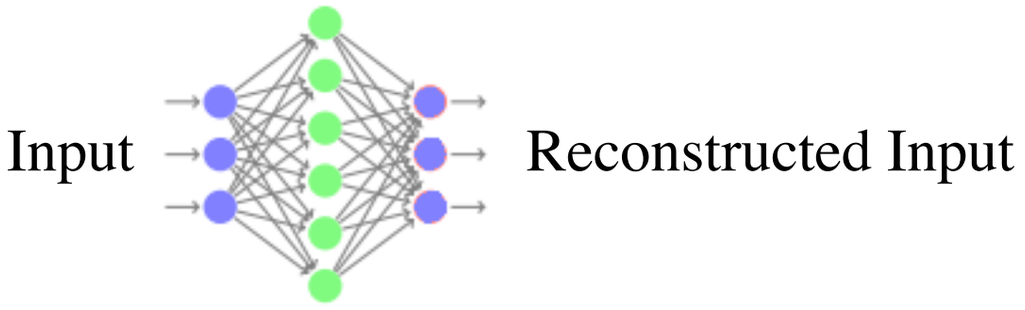




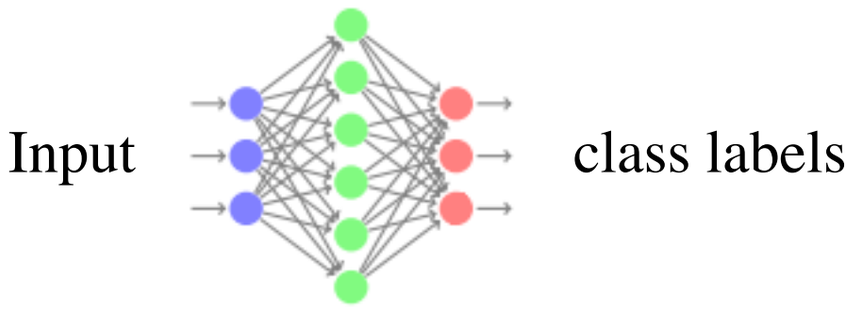
http://r9y9.github.io/blog/2014/03/06/restricted-boltzmann-machines-mnist/

Pretraining ANN

- AE training

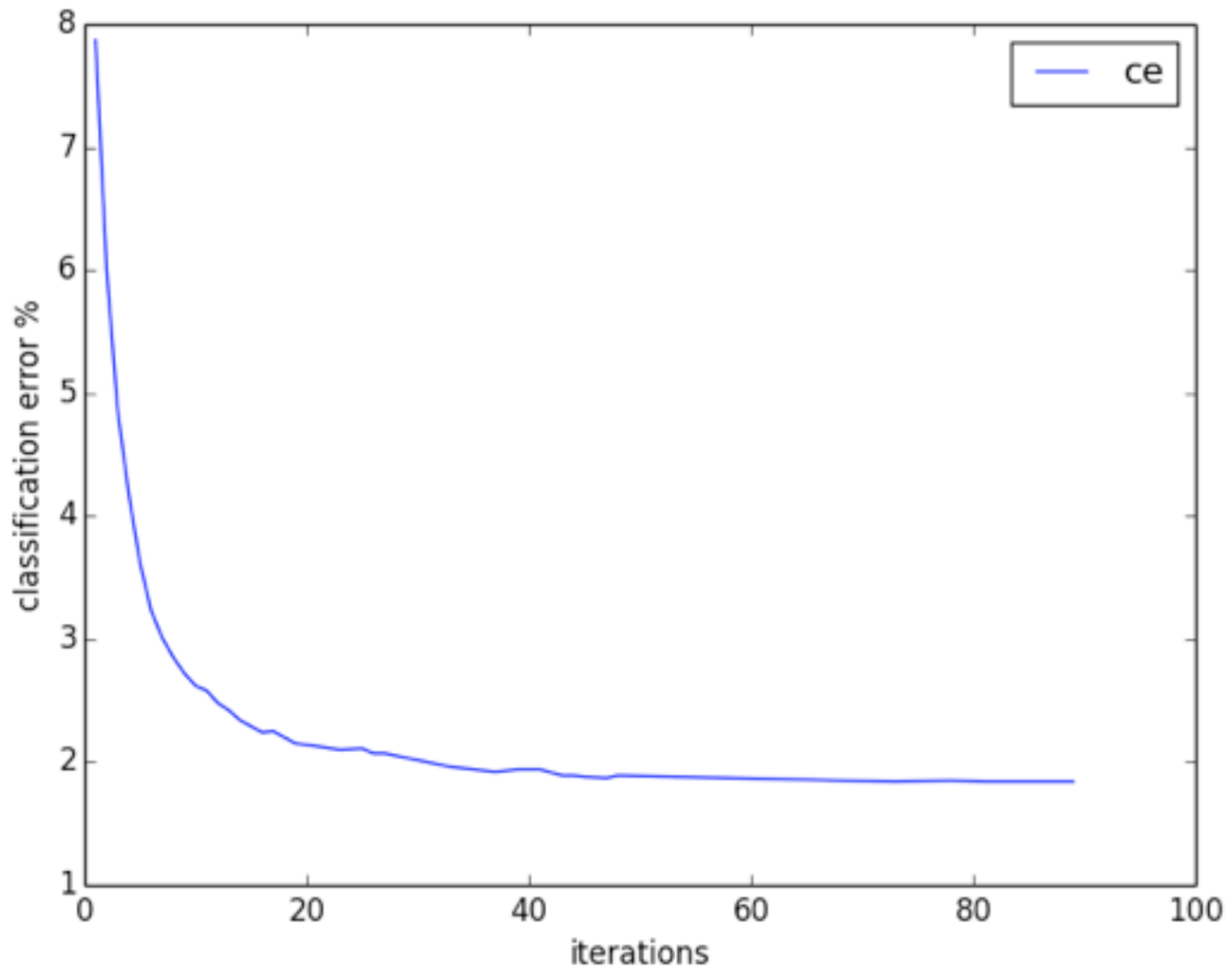


- No pre-training: Initializing ANN weights from random numbers
- Pre-training: Initializing ANN weights from AE weights



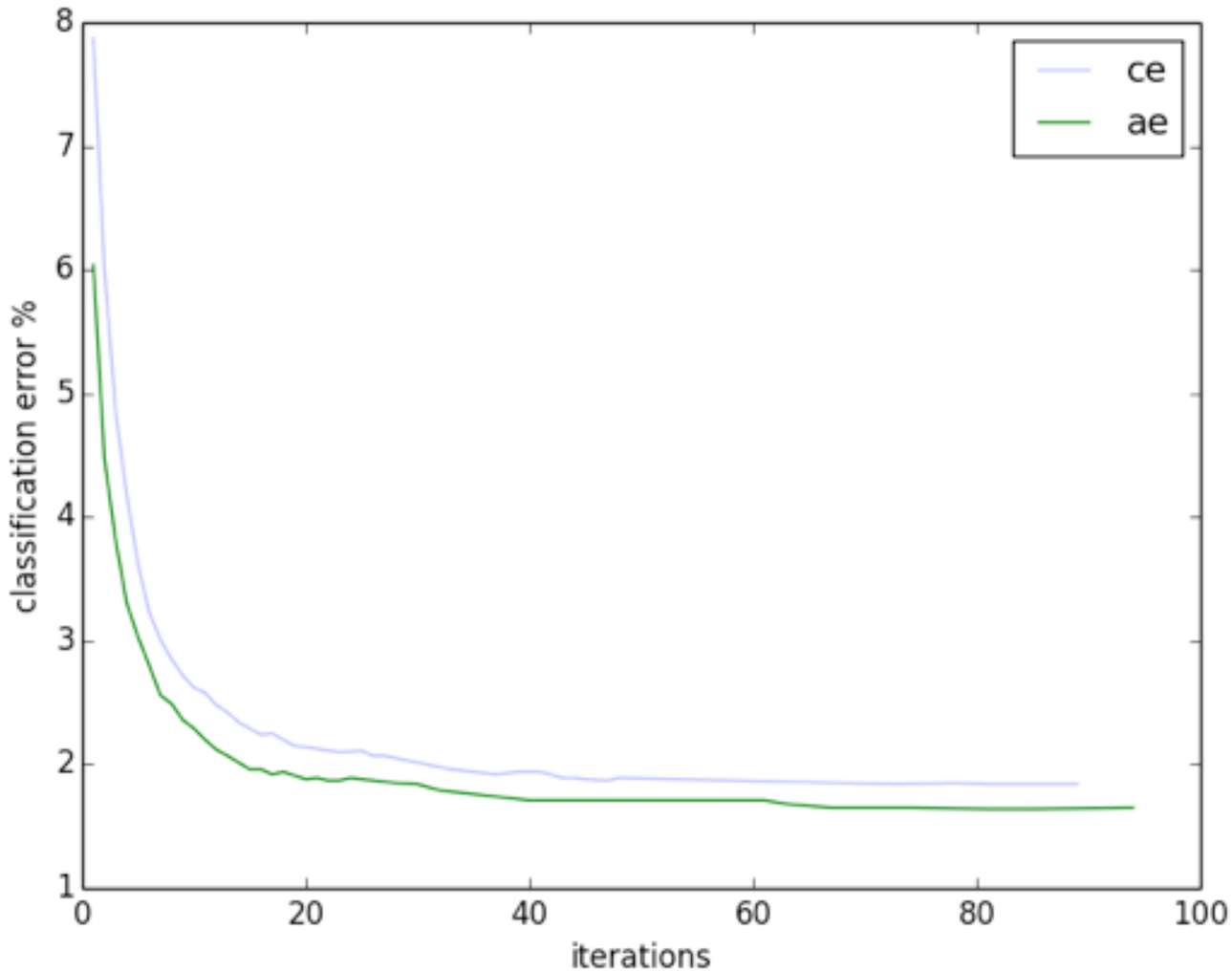
ANN for MNIST

- ANN with 1000 hidden units



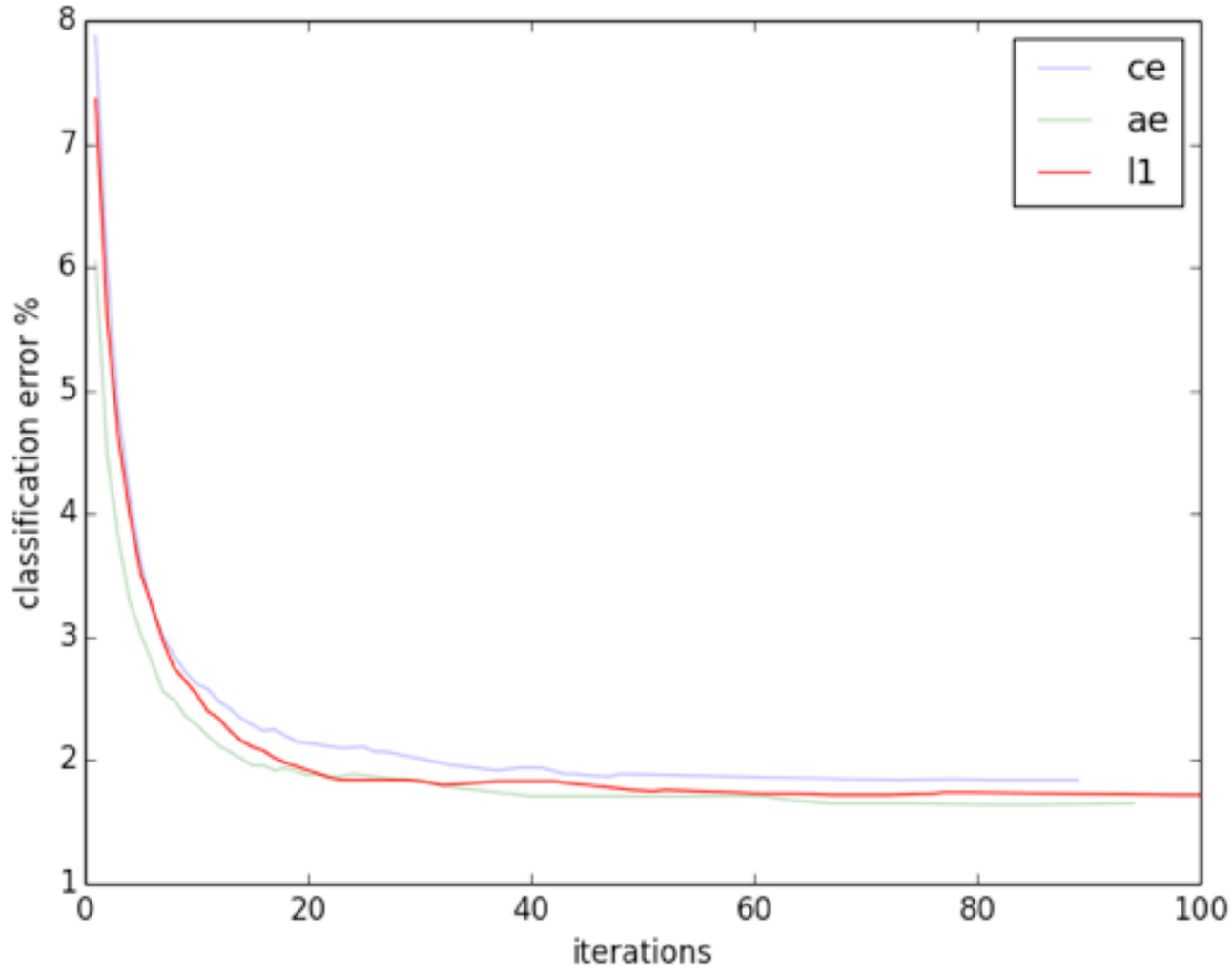
Autoencoders

- ANN pre-trained with regular AE



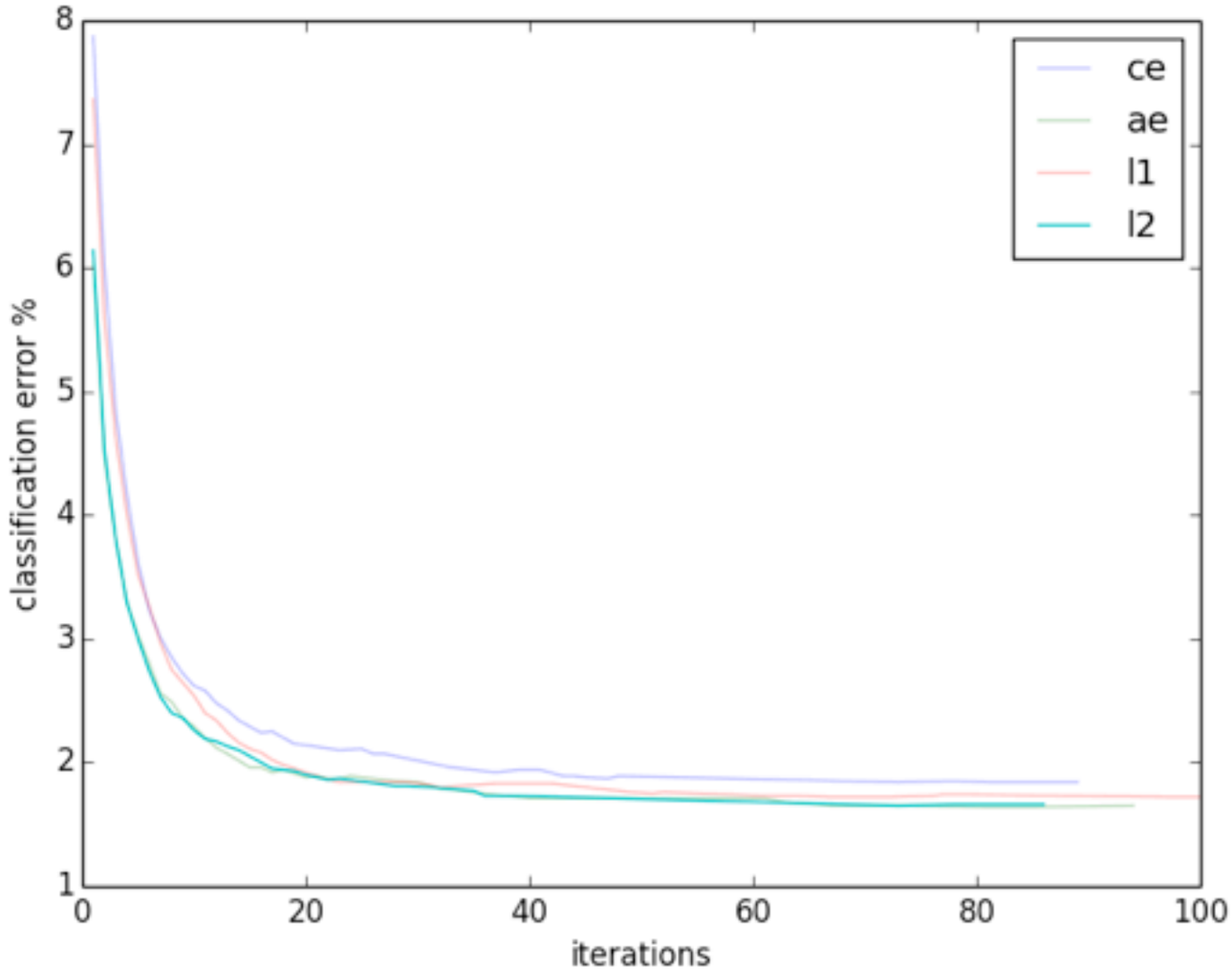
AE Regularization

- ANN pre-trained with regular AE-L1-0.001



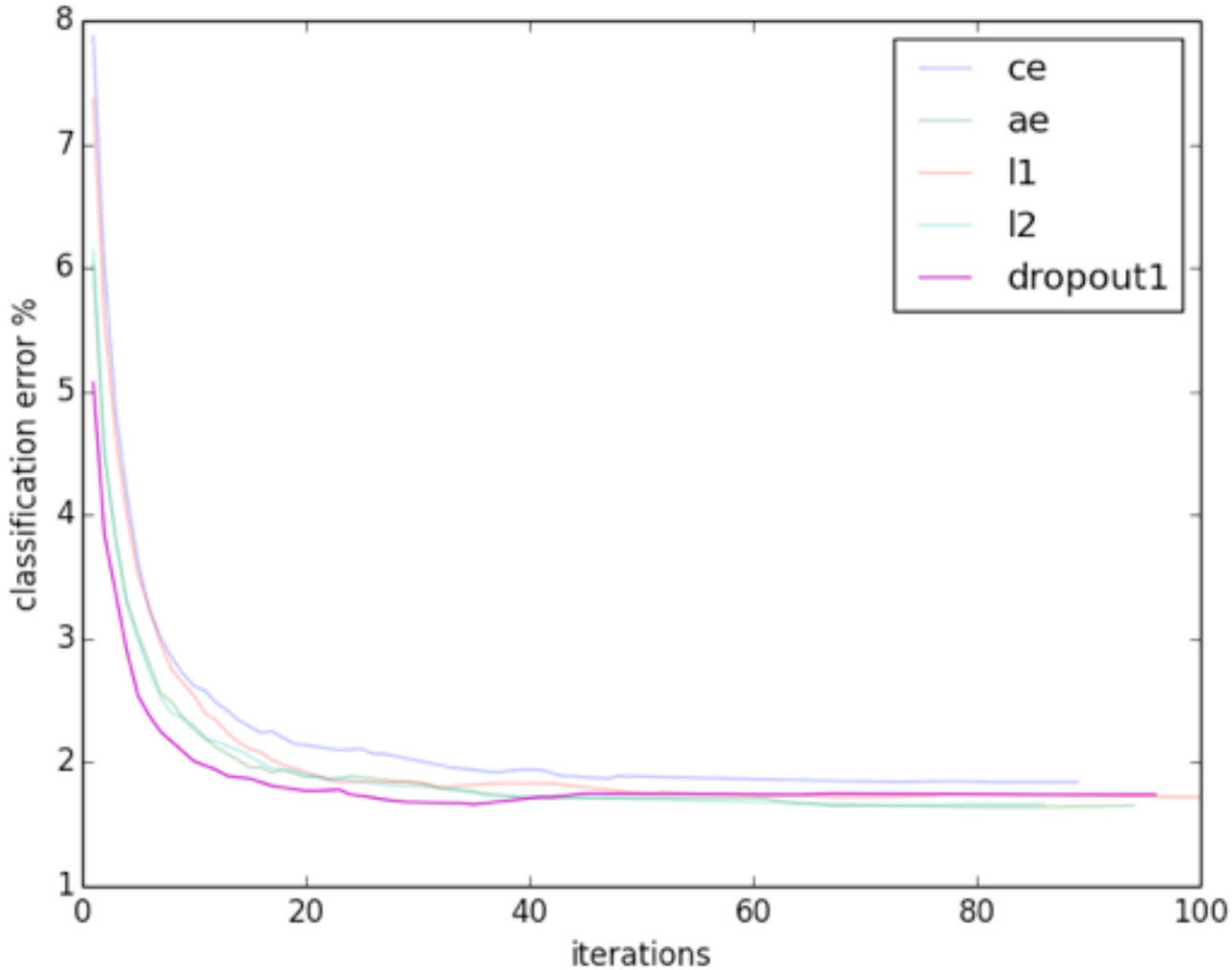
AE Regularization

- ANN pre-trained with regular AE-L2-0.001



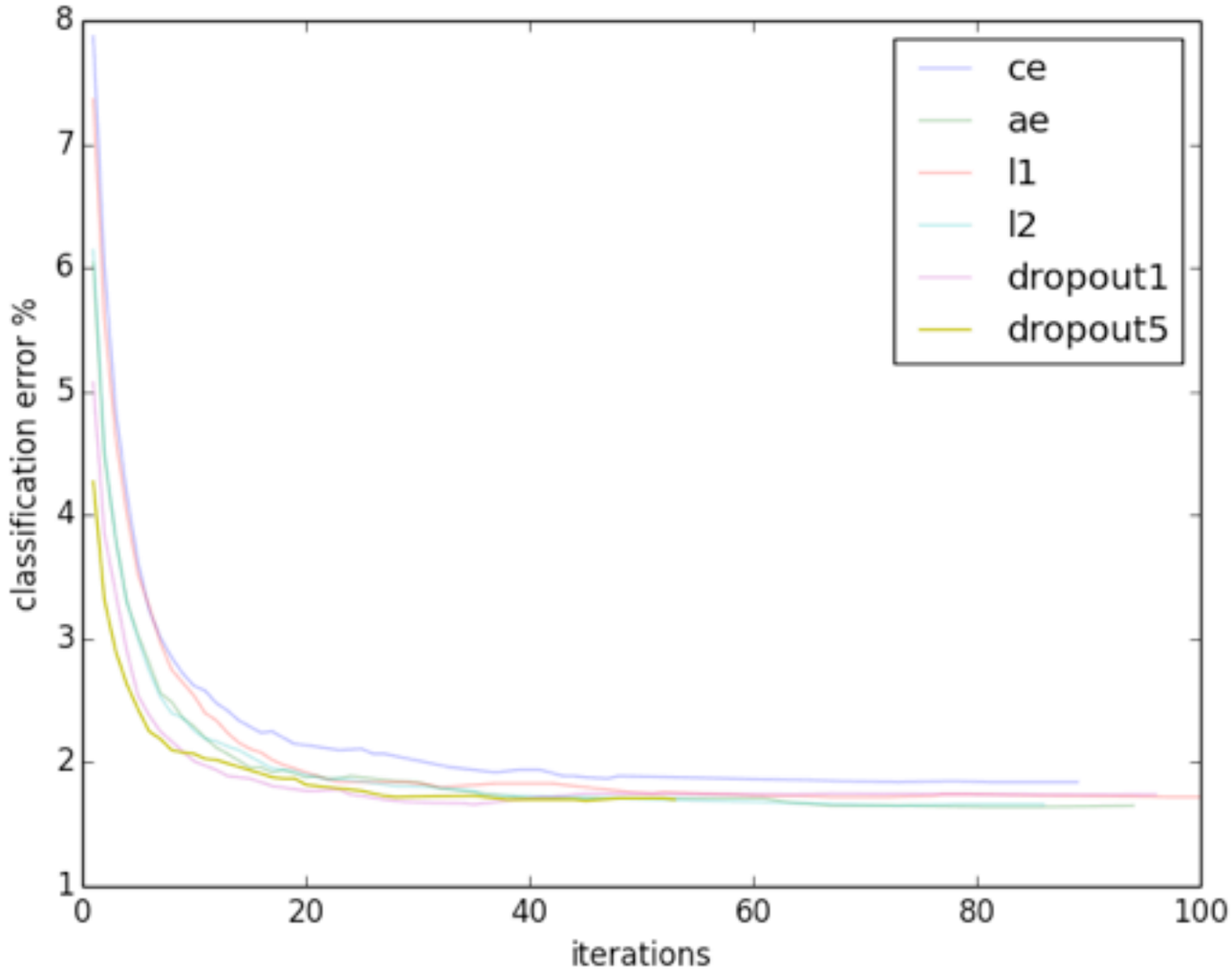
AE Regularization

- ANN pre-trained with regular AE-Dropout-0.1



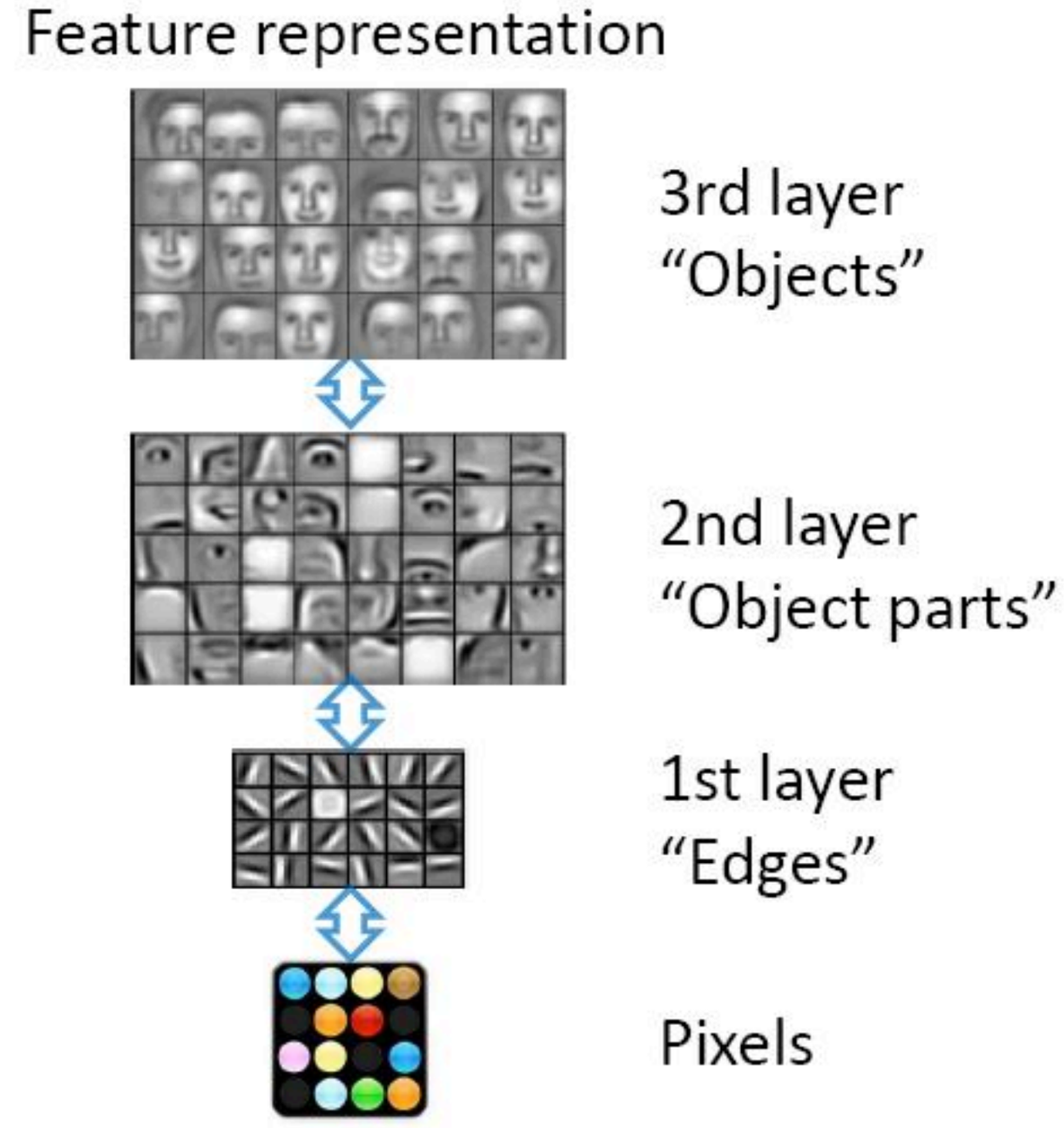
AE Regularization

- ANN pre-trained with regular AE-Dropout-0.5



Stacked Autoencoders

Different Levels of Abstraction



https://deeplearningworkshopnips2010.files.wordpress.com/2010/09/nips10-workshop-tutorial-final.pdf

Evaluation

Model	Notes	Reconstruction	Classification Error %
ANN 1000	-	-	1.84%
ANN 1000	Pre-trained AE	0.00101	1.65%
ANN 1000	Pre-trained AE-L1-0.001	0.00131	1.72%
ANN 1000	Pre-trained AE-L2--0.001	0.00194	1.66%
ANN 1000	Pre-trained AE-Dropout-0.1	0.00256	1.66%
ANN 1000	Pre-trained AE-Dropout-0.5	0.01556	1.70%
DNN 1000-1000	-	-	1.74%
DNN 1000-1000	AE-Dropout-0.1-0.2	0.00411	**1.46%**
DNN 1000-1000-1000	-	-	1.72%
DNN 1000-1000-1000	AE-Dropout-0.1-0.2-0.3	0.00952	**1.40%**

100 supervised training iterations,
lr=0.1, batch=10, SGD

15 unsupervised training iterations,
lr=0.05, batch=10, SGD

Evaluation

- 49,000 images for unsupervised training
- 1000 image for supervised training

Model	Notes	Classification Error %
ANN 100	-	12.92%
ANN 1000	-	13.47%
DNN 1000-1000-1000	-	12.71%
ANN 1000	Pre-trained AE-Dropout-0.1	10.36%
DNN 1000-1000-1000	AE-Dropout-0.1-0.2-0.3	**8.75%**

100 supervised training iterations,
lr=0.1, batch=10, SGD

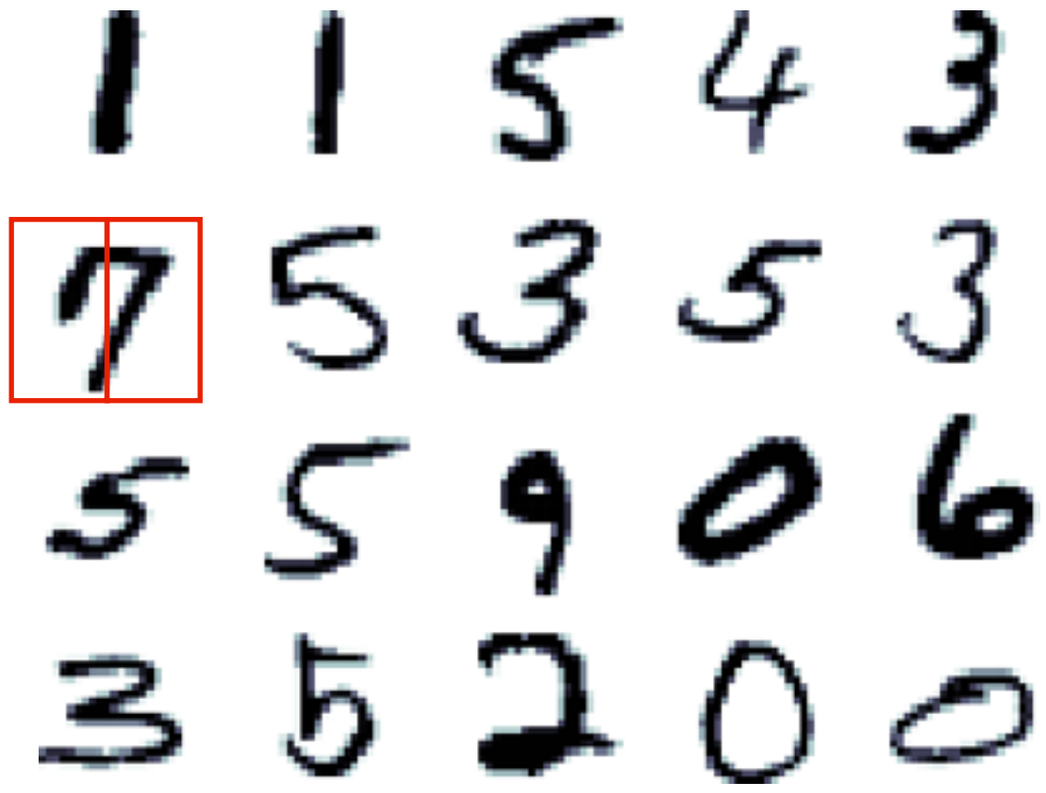
15 unsupervised training iterations,
lr=0.05, batch=10, SGD

Feature Mapping

- The usual application of DNN is classification
- We will propose some training methods for mapping (regression)
- These have the potential to be used in voice conversion

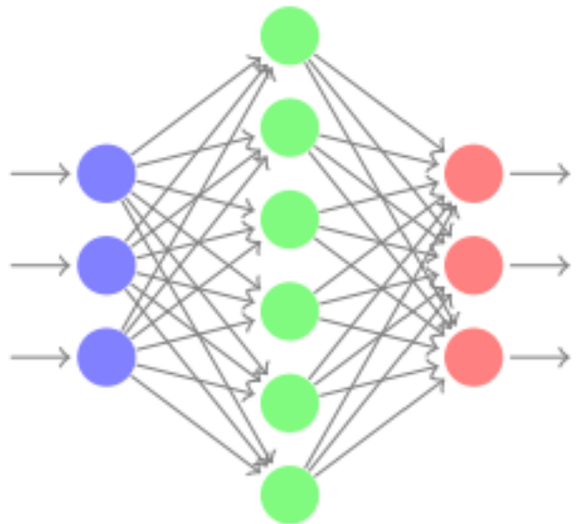
Feature Mapping

- In this study, we will perform our evaluations on MNIST
  - x: half left images
  - y: half right images



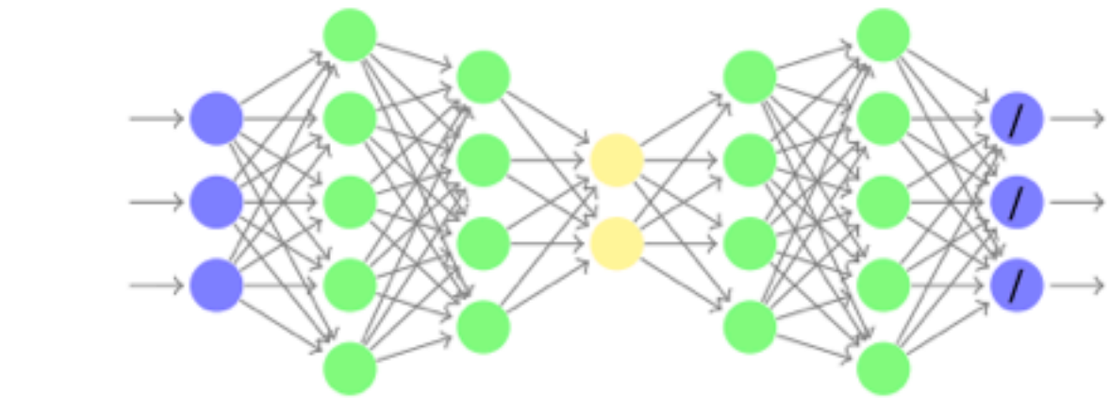
Feature Mapping

- Simple ANN mapping

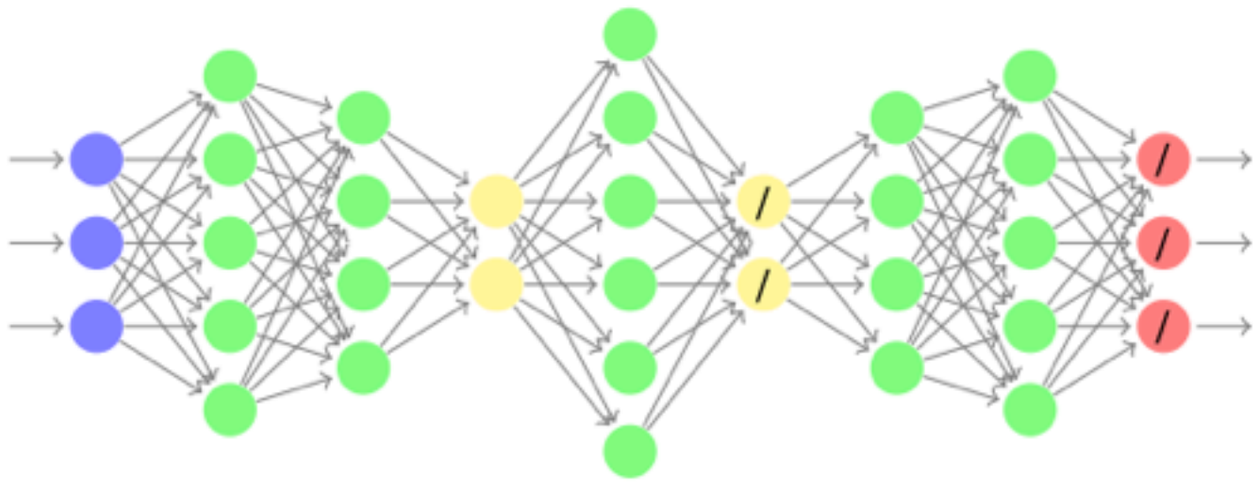
source  target

Feature Mapping

- The previously proposed approach for VC



Deep Autoencoder



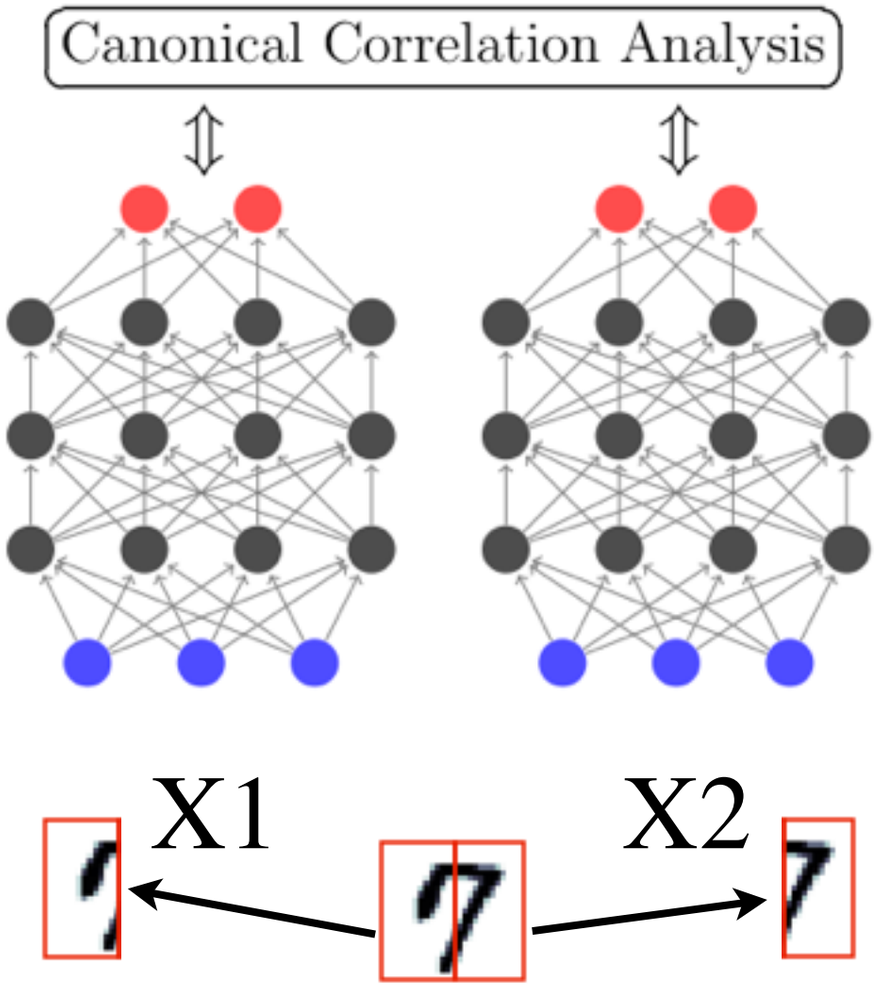
Deep Neural Network

Feature Mapping

- We propose Correlated AEs
- Background
- Canonical Correlation Analysis
- Two random variables `X1` and `X2`
- find `W1` and `W2` such that `W1.X` and `W2.X2` are maximally correlated

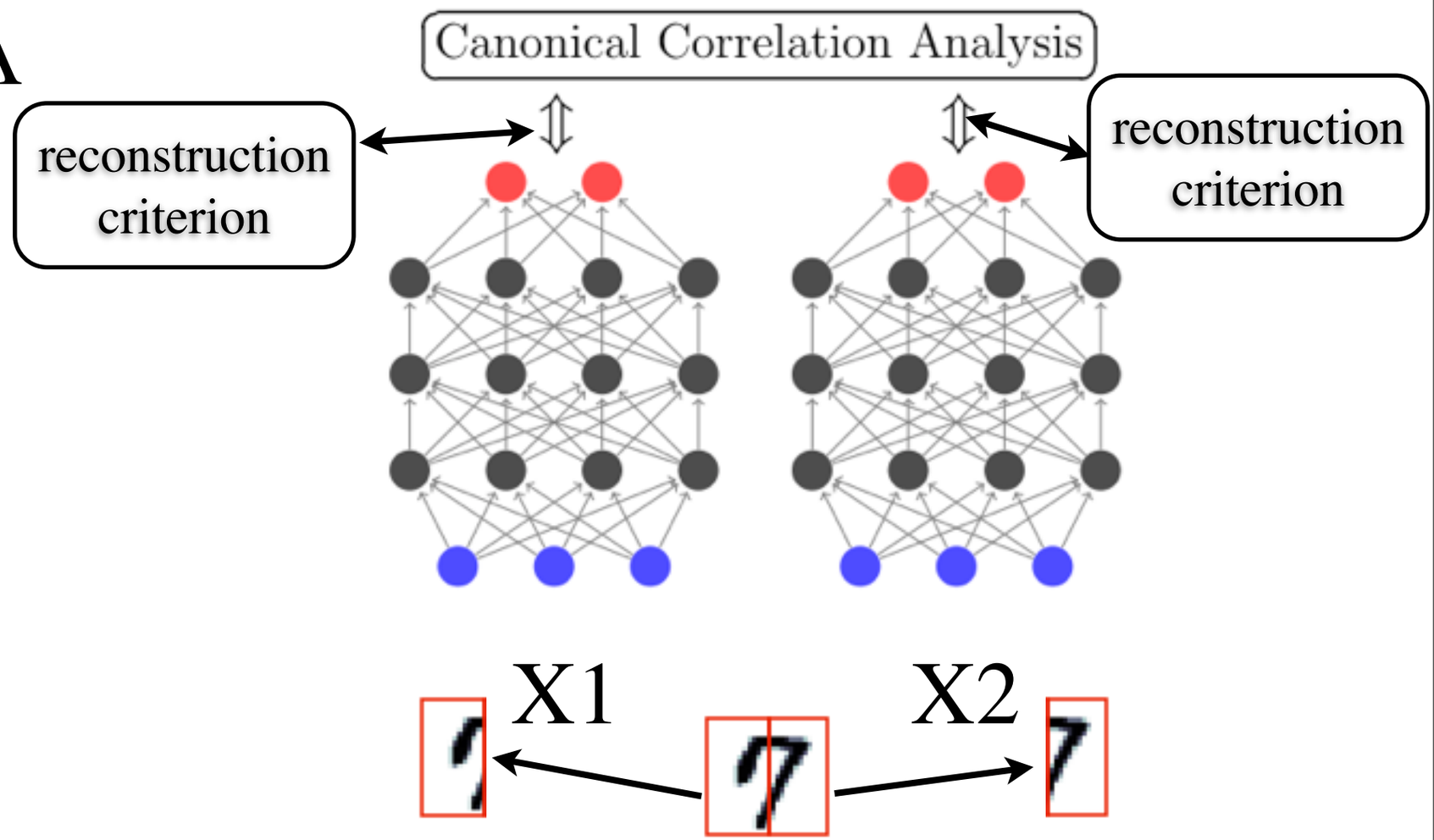
Feature Mapping

- Deep CCA
- The hidden layer outputs have the highest correlation



Feature Mapping

- DCCA does not have good reconstruction
- We propose join the cost of reconstruction and DCCA



Feature Mapping

- 1000 labeled, 49,000 unlabeled

Model	Notes	Reconstruction
ANN 100	-	0.0724
ANN 100	AE-Dropout-0.1	0.0554
ANN 100	AE-Dropout-0.1+DCCA	0.0463

400 supervised training iterations,
lr=0.1, batch=20, SGD

15 unsupervised training iterations,
lr=0.1, batch=20, SGD