An Analysis of Single-Layer Networks in Unsupervised Feature Learning

Adam Coates1, Honglak Lee2, Andrew Y. Ng1

Overview

- A Brief Introduction
- Unsupervised feature learning framework
- Experiments and Analysis
- Q&A

A Brief Intro

- Recent works focus on learning good feature representation
    - greedily pre-train several layers of feature
    - for each layer a set of parameters are chosen:
        - Number of features to learn
        - the location of the features
        - encoding input and output
- A Major drawback is complexity and expense

A Brief Intro

- In this paper :
  - First study the effect of these choices on single layer network
  - It turns out that there are other ingredients:
    - Whitening
    - large number of features
    - dense feature extraction

A Brief Intro

- What they did :
  - Used a simple feature learning framework that incorporates an unsupervised learning algorithm as a black box
  - Analyze performance impact of :
    - whitening
    - number of features trained
    - step size between extracted features
    - receptive field size

Unsupervised feature learning framework

- Steps to learn features :

    1. Extract random patches form unlabeled training data
    2. Apply pre-processing stage
    3. Learn a feature mapping

- After learning features :

    1. Extract features from equally spaced sub-patches
    2. Pool features together to reduce number of feature values
    3. Train a linear classifier to predict labels

Unsupervised feature learning framework

- Extract random sub-patches :
  - each patch has dimension w-by-w and d channels
  - each patch can be represented as a vector of size $w.w.d$
  - the dataset consists of m randomly sampled patches

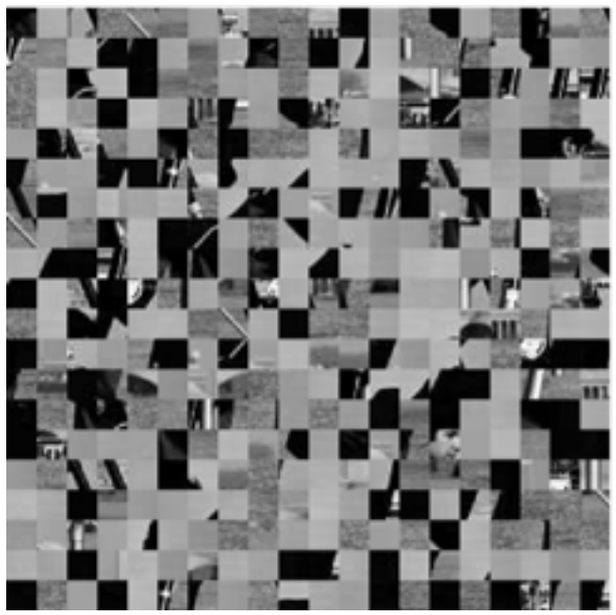
Unsupervised feature learning framework

- Pre-Processing :
  - Normalize the data by subtracting the mean and dividing by standard deviation
  - Perform whitening

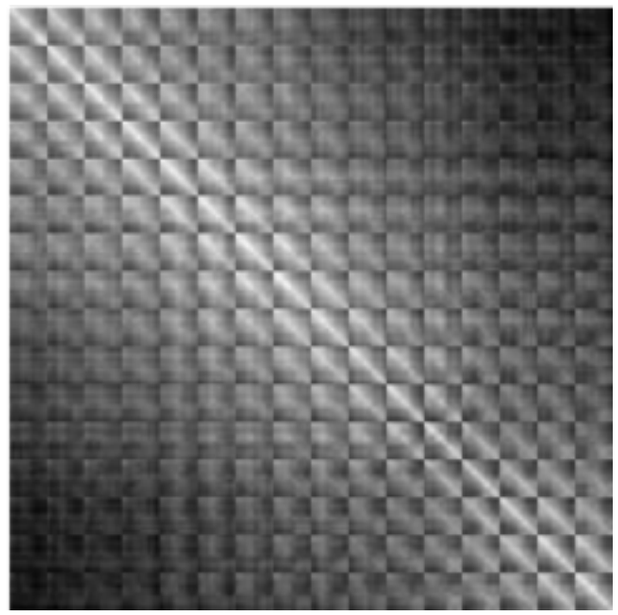
Whitening

Base Image

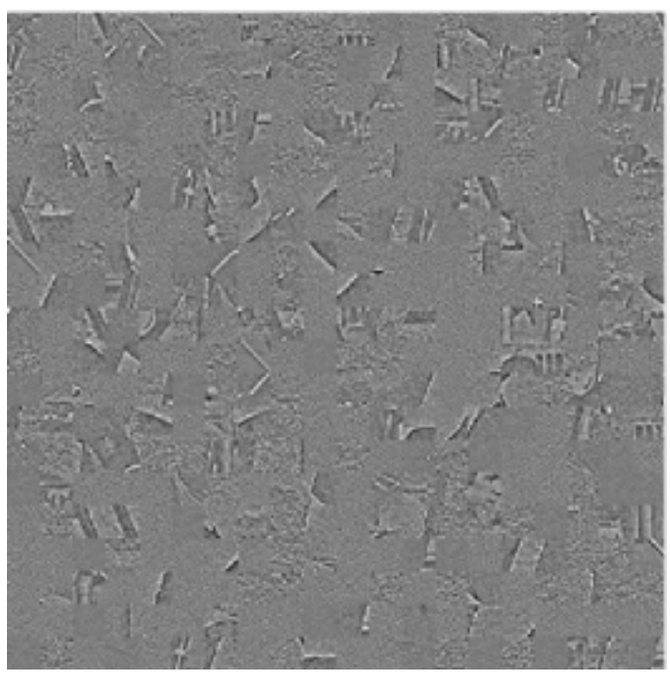
Extracted Patches



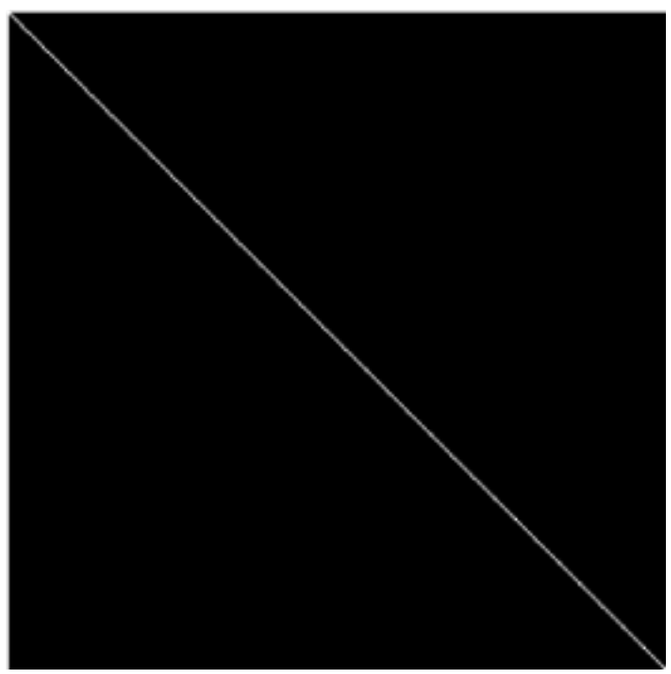
Extracted Patches Covariance



Whitened Patches



Whitened Patches Covariance



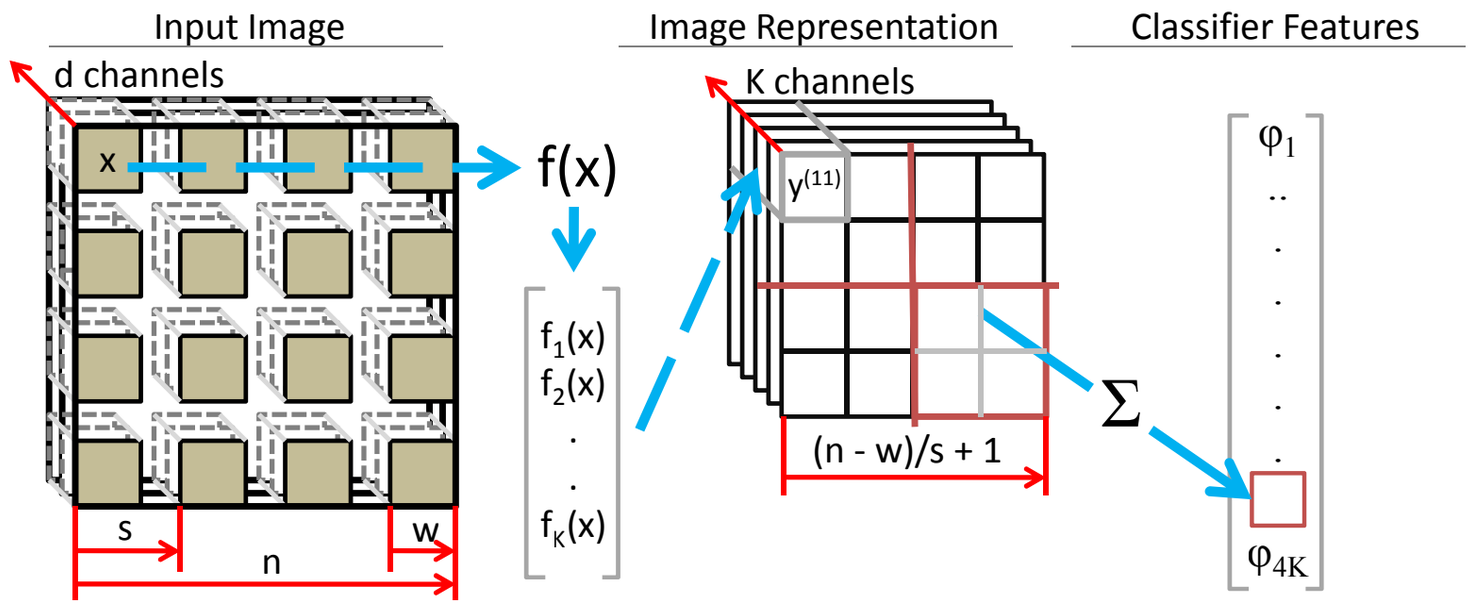
Unsupervised feature learning framework

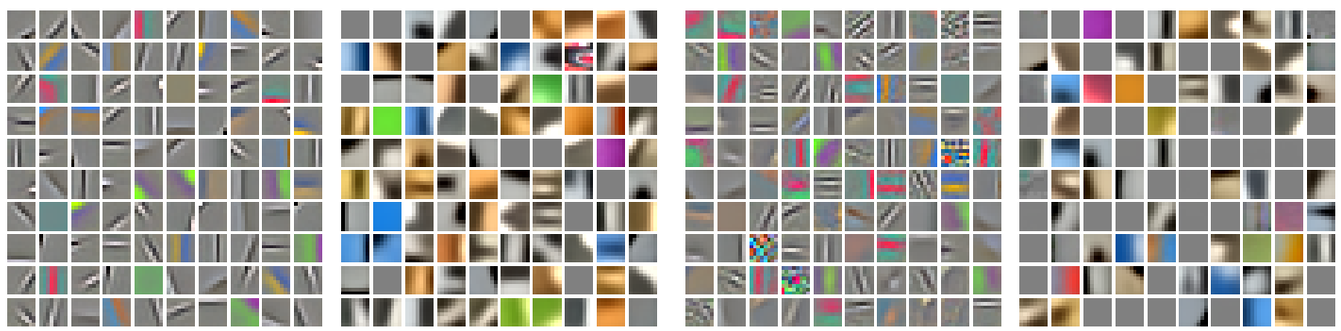
- Unsupervised Learning :
  - the “black box” takes dataset X and outputs a function $f : \mathbb{R}^N \rightarrow \mathbb{R}^K$ that maps input vector to a feature vector of size K
    - sparse auto-encoders
    - sparse RBMs
    - K-means clustering
    - Gaussian mixtures

Unsupervised feature learning framework

- Feature Extraction and Classification
  - using the learned feature extractor, given any image patch
    - compute a representation for the patch
    - do this for many sub-patches of images

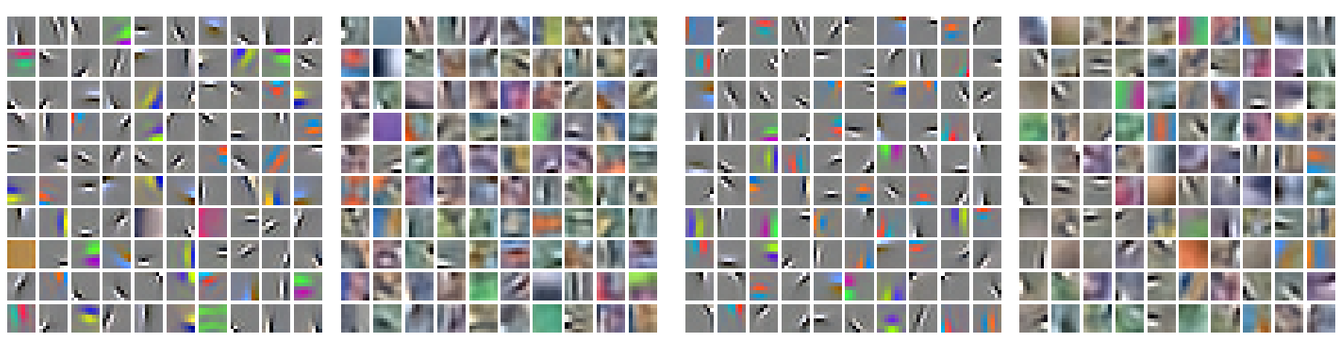
Unsupervised feature learning framework





(a) K-means (with and without whitening)

(b) GMM (with and without whitening)



(c) Sparse Autoencoder (with and without whitening)

(d) Sparse RBM (with and without whitening)

Experiments

- The above framework includes number of parameters :
  - whitening
  - number of features
  - the step size
  - receptive field

Experiments

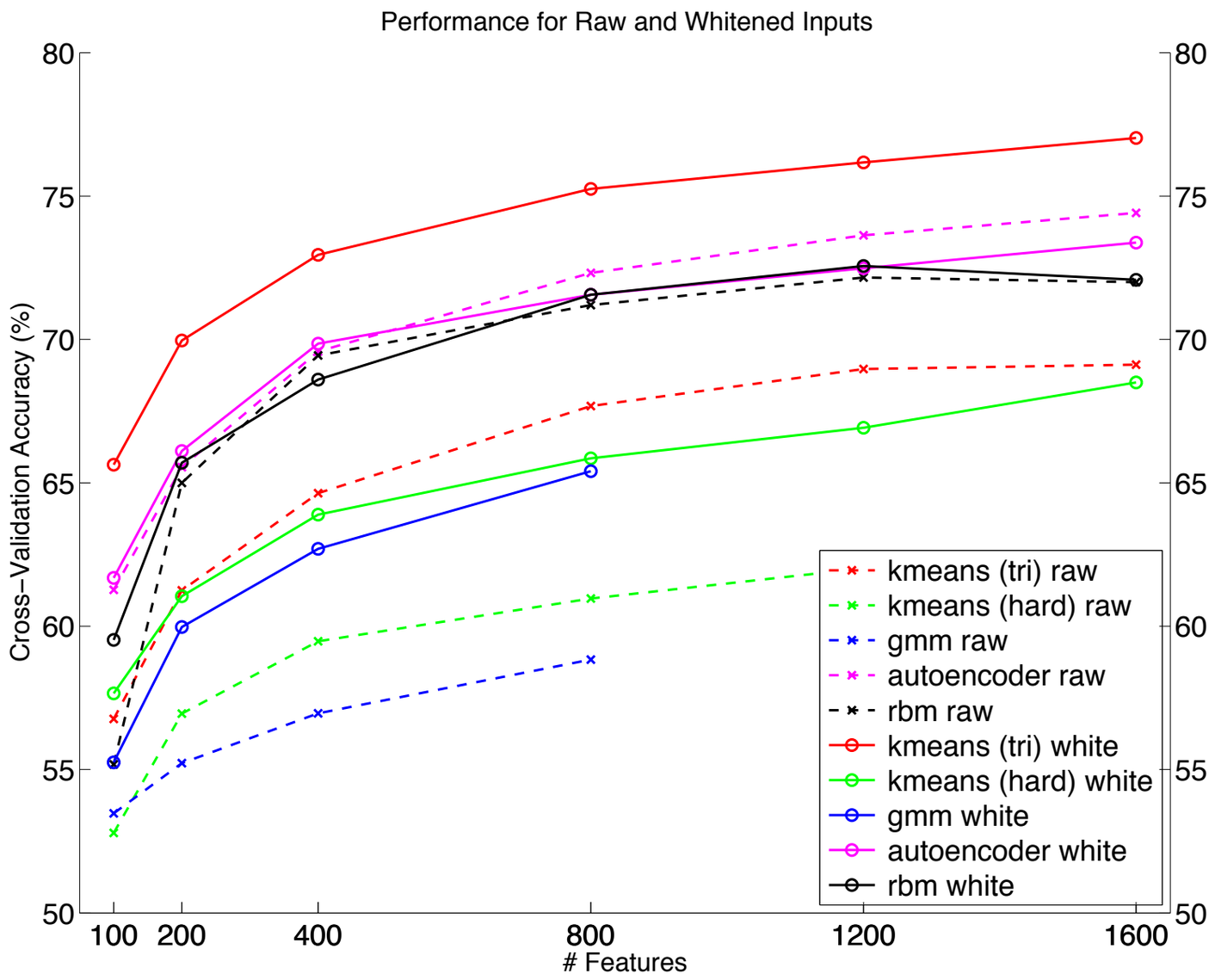
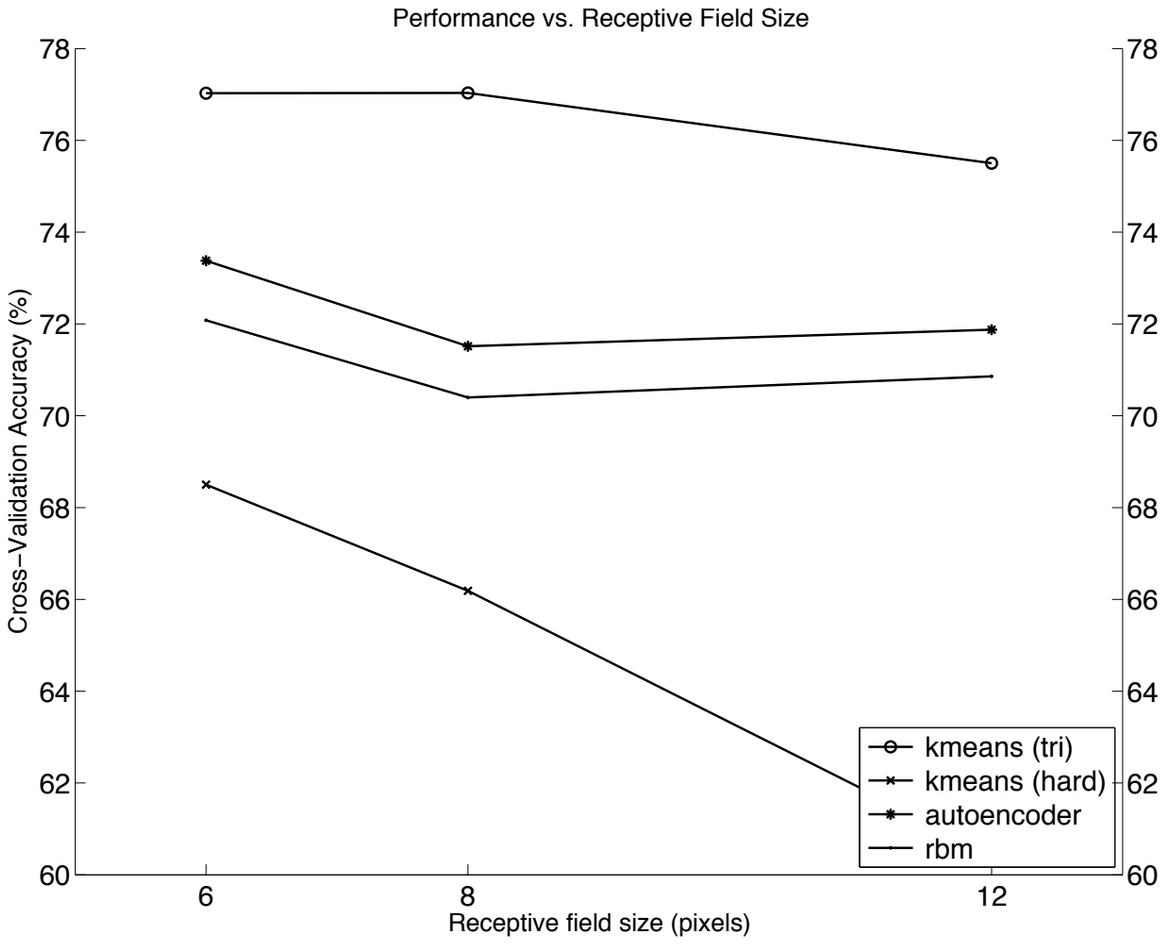
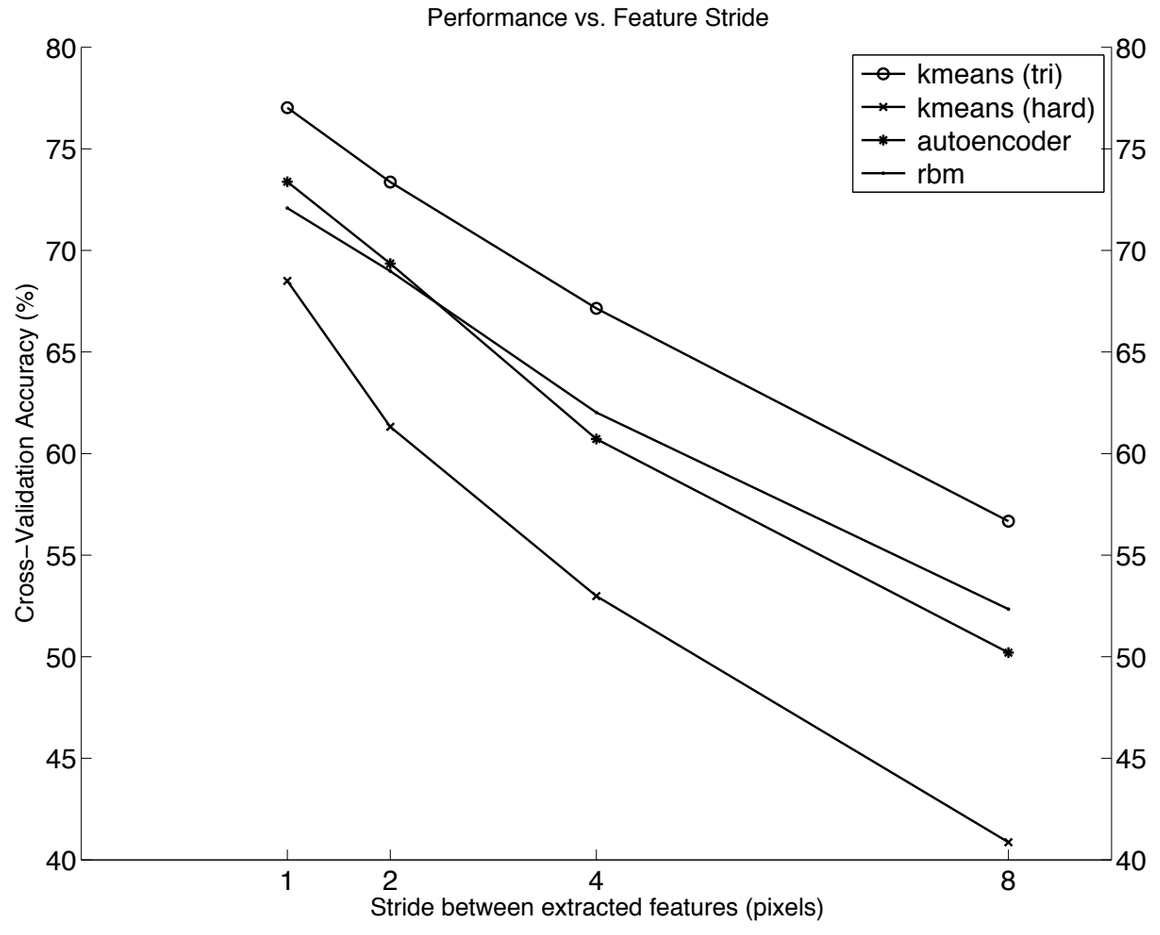


Figure 3: Effect of whitening and number of bases (or centroids).

Experiments





Experiments

Table 1: Test recognition accuracy (and error) for NORB (normalized-uniform

Algorithm	Test accuracy (and error)
Convolutional Neural Networks [14]	93.4% (6.6%)
Deep Boltzmann Machines [25]	92.8% (7.2%)
Deep Belief Networks [18]	95.0% (5.0%)
(Best result of [10])	94.4% (5.6%)
K-means (Triangle)	**97.0% (3.0%)**
K-means (Hard)	96.9% (3.1%)
Sparse auto-encoder	96.9% (3.1%)
Sparse RBM	96.2% (3.8%)

Table 2: Test recognition accuracy on CIFAR-10

Algorithm	Test accuracy
Raw pixels (reported in [11])	37.3%
RBM with backpropagation [11]	64.8%
3-Way Factored RBM + ZCA (3 layers) [23]	65.3%
Mean-covariance RBM (3 layers) [22]	71.0%
Improved Local Coordinate Coding [31]	74.5%
Convolutional RBM [12]	78.9%
K-means (Triangle)	**77.9%**
K-means (Hard)	68.6%
Sparse auto-encoder	73.4%
Sparse RBM	72.4%
K-means (Triangle, 4k features)	**79.6%**

Questions?