Lecture 8a

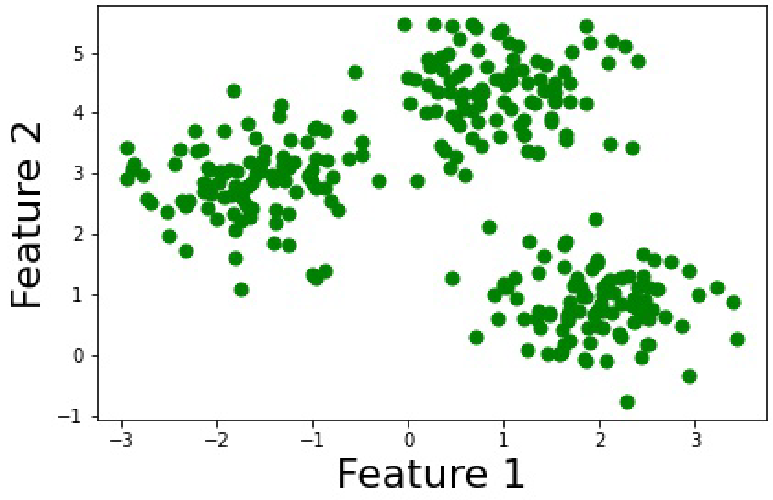
Learning without labels

The Rosetta stone

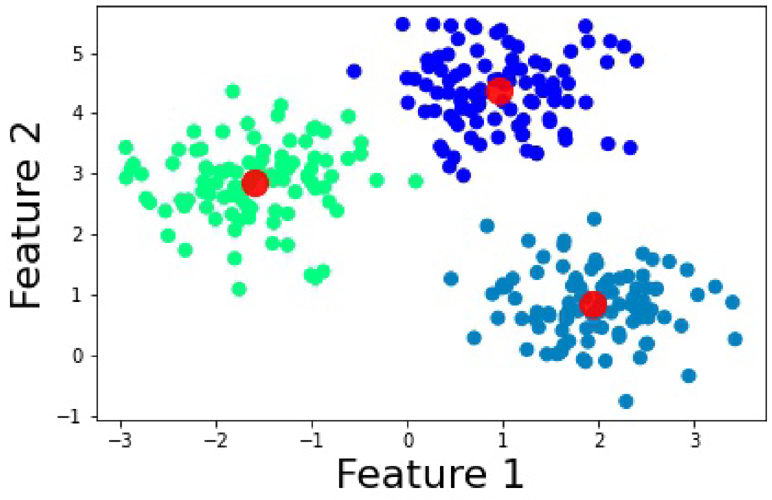
K-means

K-Means

- A clustering method
- The algorithm assign each data point to one of the K groups
- Data points in the same groups are more similar



Applying K-means



Red points are the centroids of the clusters

K-means

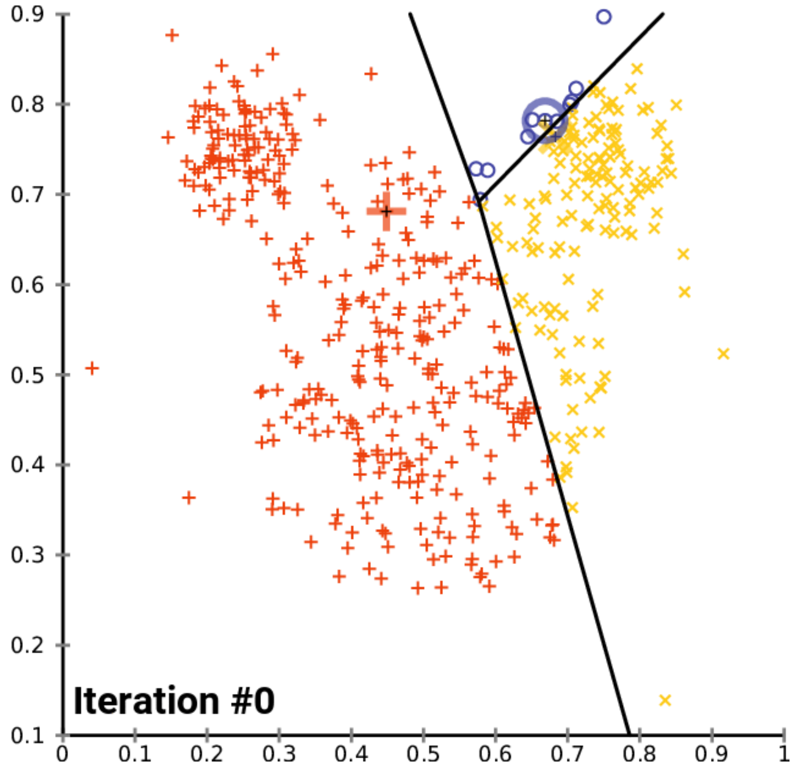
K-means iterates over the following steps until there is no change to the centroids:

1. Compute the sum of the squared distance between each data point and all centroids.
2. Assign each data point to the closest cluster (centroid).
3. Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.
4. If the calculated means don't change then STOP. Otherwise Go to 1

Cost function:

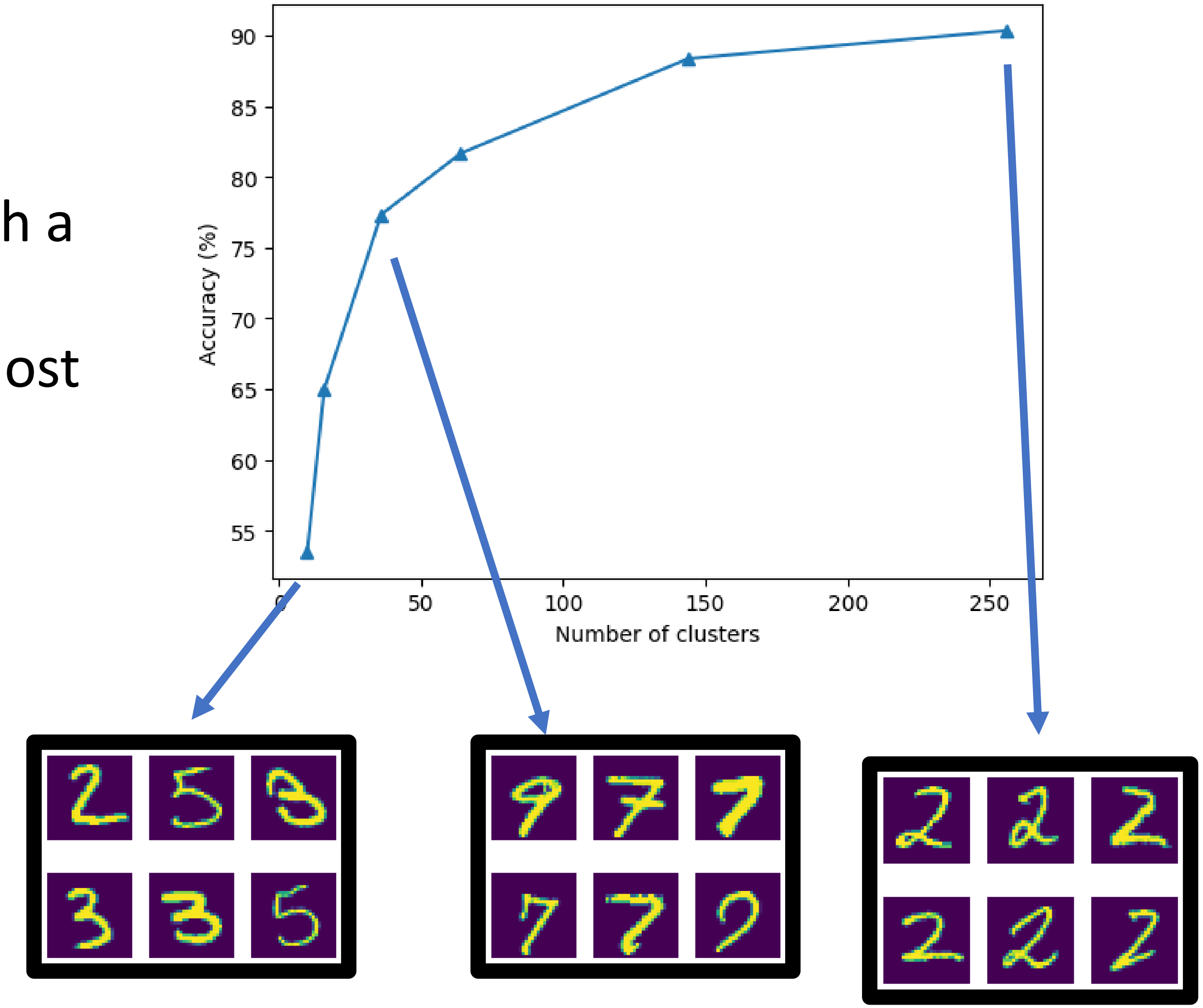
$$C = \sum_{i=1}^{k} \sum_{x \in s_i} \left\| x - \mu_i \right\|^2$$

- s_i is cluster i
- μ_i is the mean of points in cluster i
- k is the number of clusters



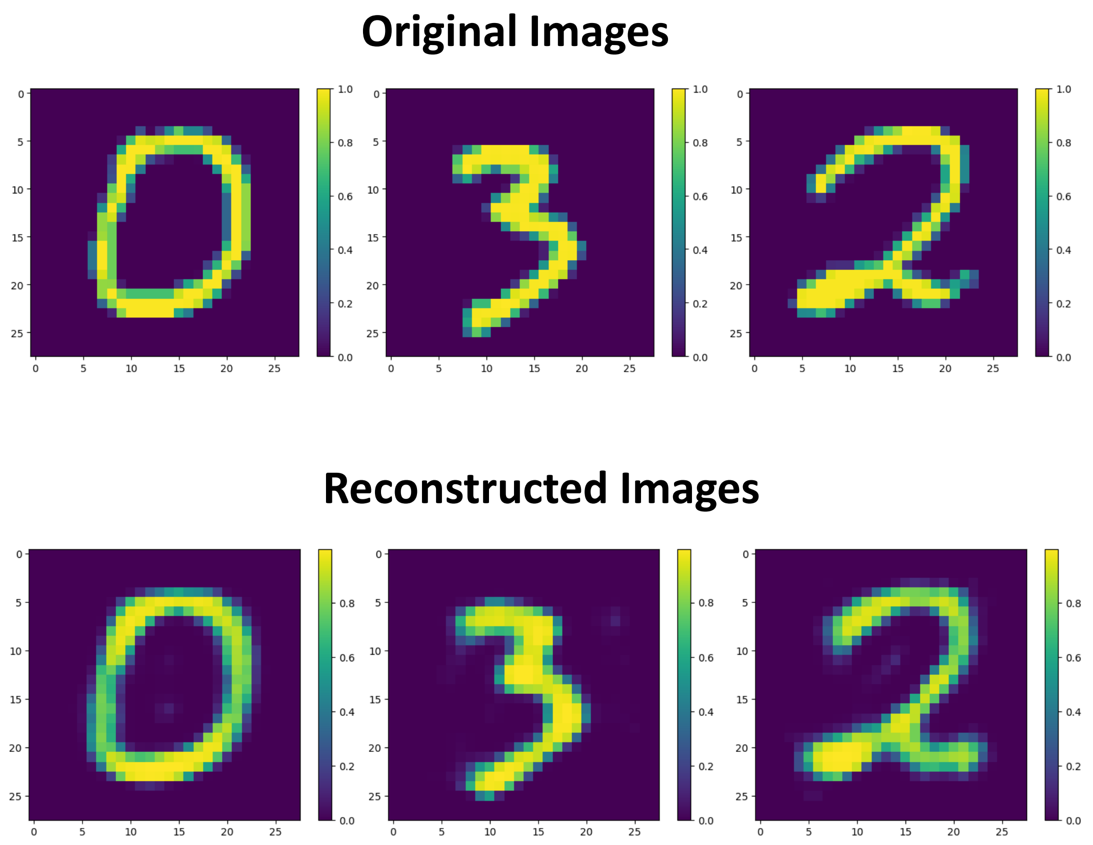
K-Means Clustering on MNIST-Digits

- Algorithm:
  - K-means clustering of training set with a given number of means
  - Assign each cluster the label of the most common class
  - For test samples, find the closest mean/label



Autoencoder on MNIST

- Dataset: MNIST-Digits training set (60000 samples, 28 x 28)
- Network: 28 hidden units with ReLU, 784 output units with sigmoid, trained with Adam and MSE loss

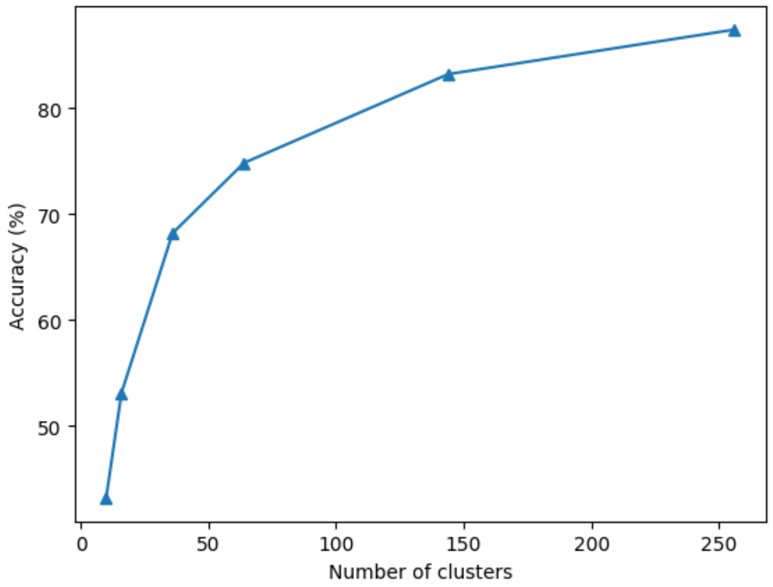


Original Images

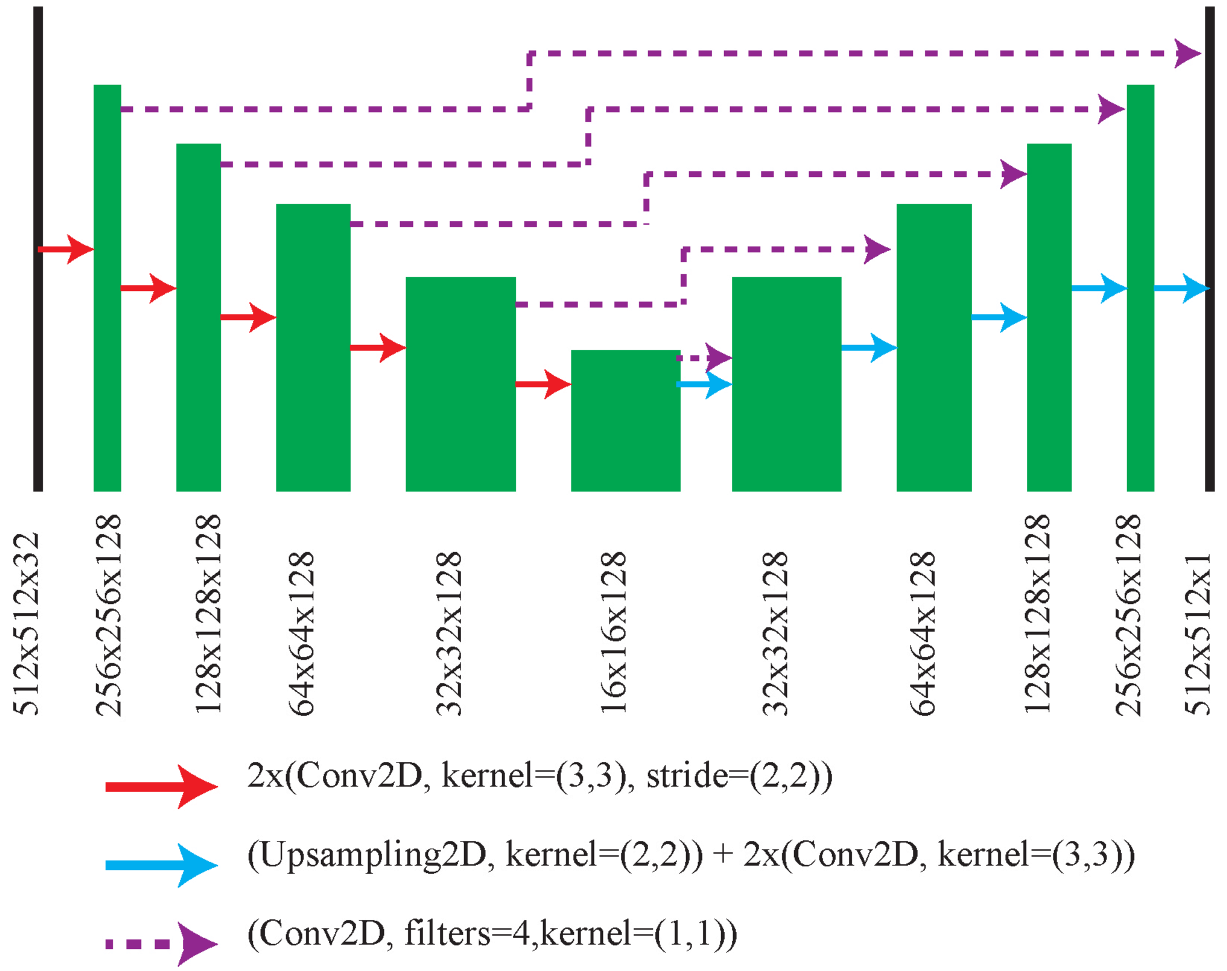
Reconstructed Images

Clustering Hidden Layer Activations

- In the autoencoder architecture, hidden unit activations (28 features) of test samples are clustered with K-means.
- Despite the smaller dimensions, accuracy is similar with when 784 dimensional input is used.



Deep Image Prior



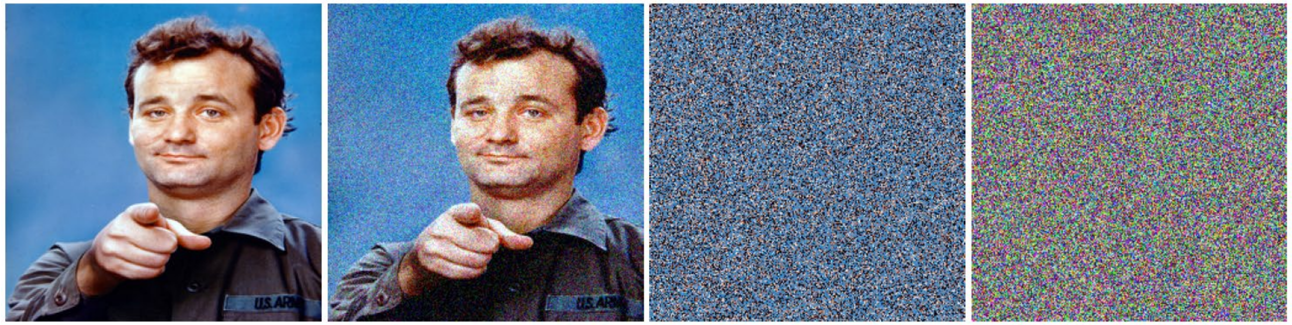
→ 2x(Conv2D, kernel=(3,3), stride=(2,2))

→ (Upsampling2D, kernel=(2,2)) + 2x(Conv2D, kernel=(3,3))

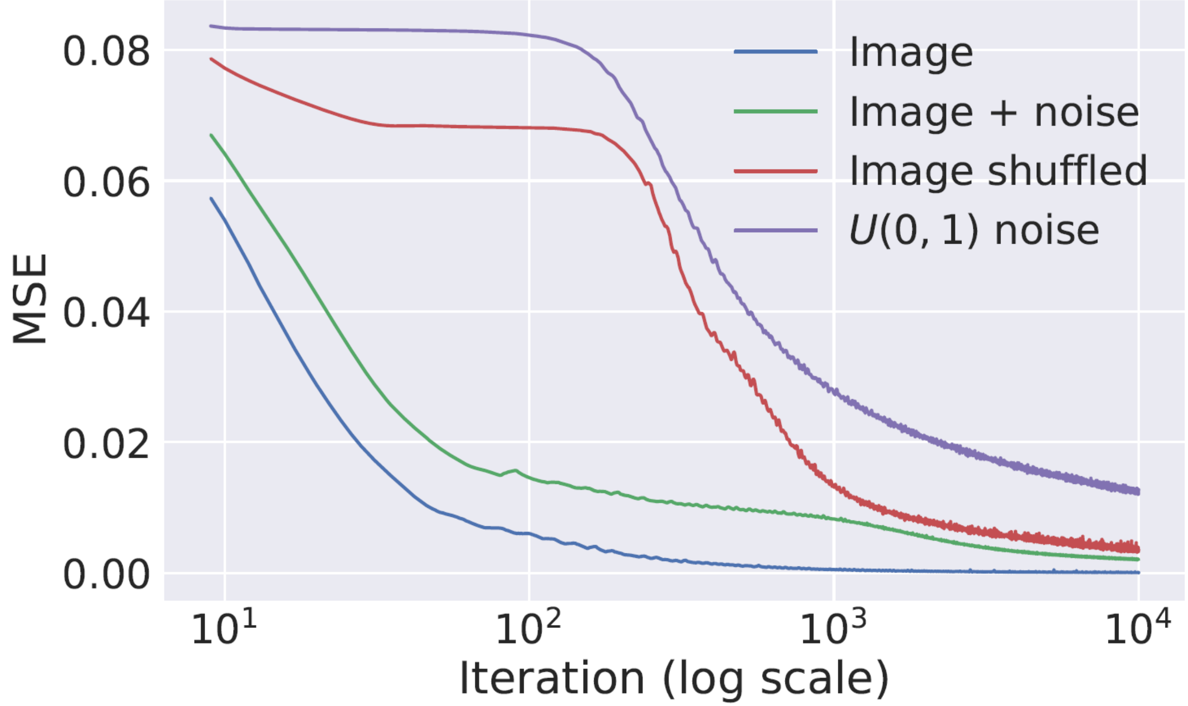
⇢ (Conv2D, filters=4,kernel=(1,1))

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2018). Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9446-9454).

Deep Image Prior

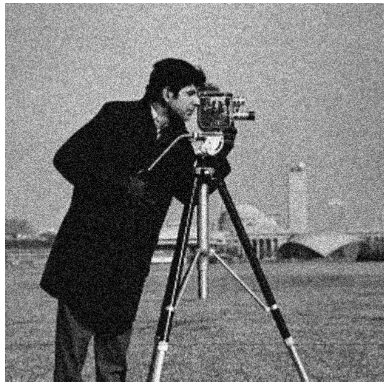


(a) Image (b) Image + noise (c) Image shuffled (d) $U(0,1)$ noise

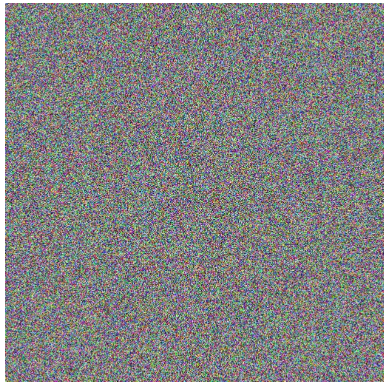


Deep Image Prior: Denoising

Clean image



Noisy image

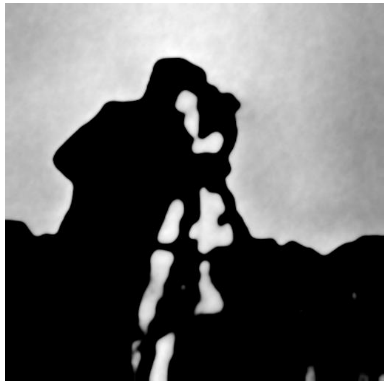


Network input

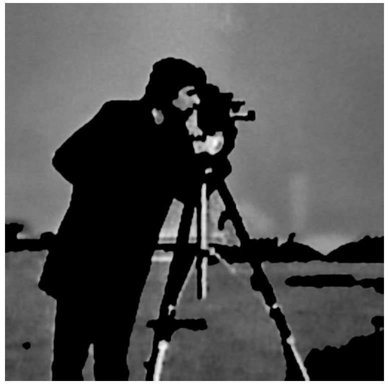
Network structure:

- 5 down-sampling
  - Kernel size=3
  - Number of filters=128
- 5 up-sampling
  - Kernel size=3
  - Number of filters=128
- 5 skip connection
  - Kernel size=1
  - Number of filters=4
- Leaky ReLU activation function

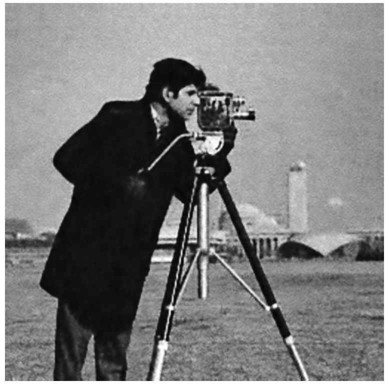
Cost function: $C = \left\| network_{output} - image_{noisy} \right\|^2$



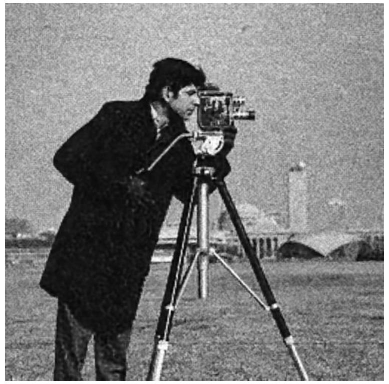
Epoch 100



Epoch 300



Epoch 3000

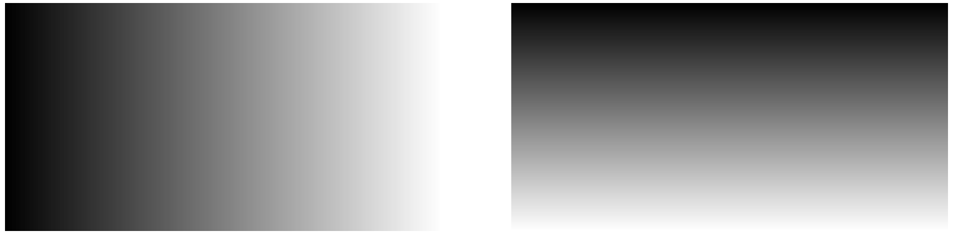


Epoch 3900

The network reaches to the denoised image at epoch=3000.

Deep Image Prior: Inpainting

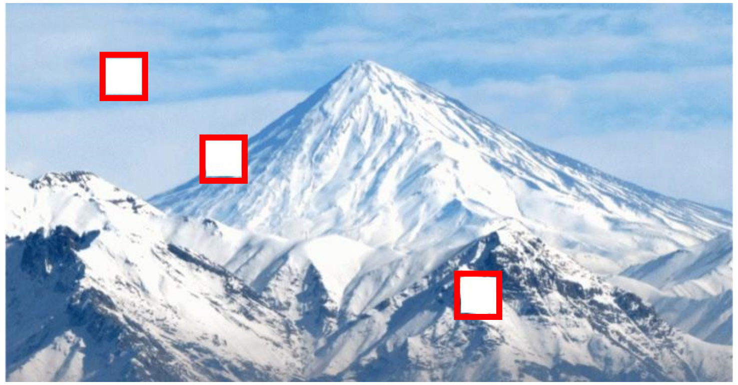
Network input



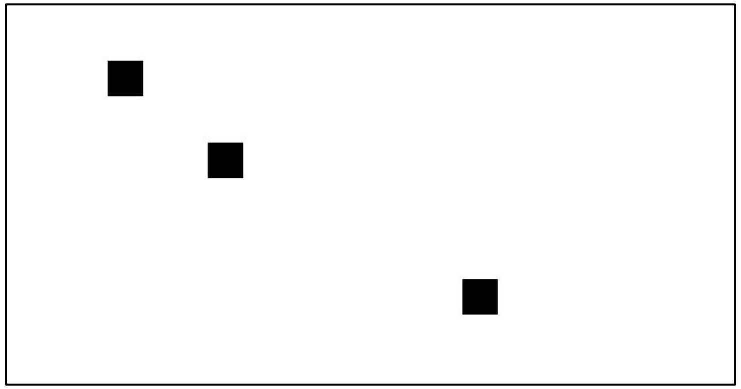
Network structure:

- 5 down-sampling
  - Kernel size=3x3
  - Number of filters in each layer=128
- 5 up-sampling
  - Kernel size=3x3
  - Number of filters in each layer=128
- No skip connection
- Leaky ReLU activation function

Masked image



Mask

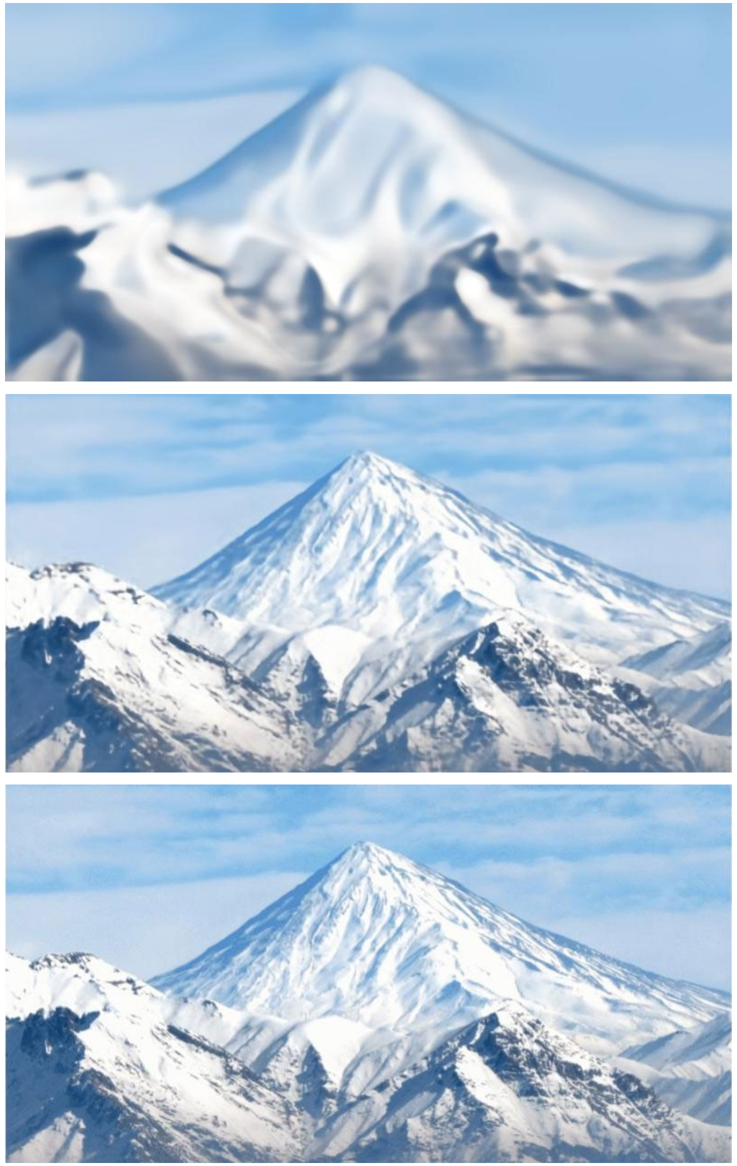


Cost function:

$$C = \left\| (network_{output} - image_{masked}) \square \ mask \right\|^2$$

INNER PRODUCT

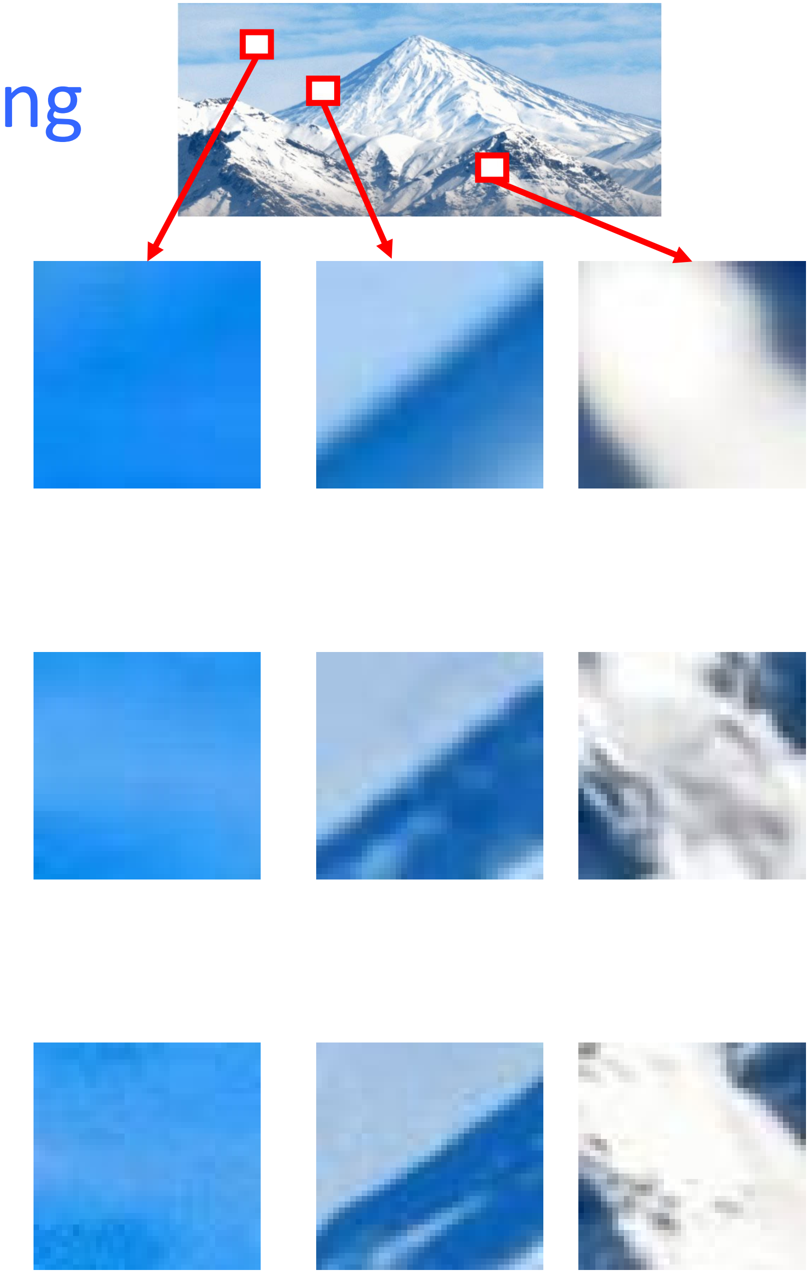
Deep Image Prior: Inpainting



Epoch 1000

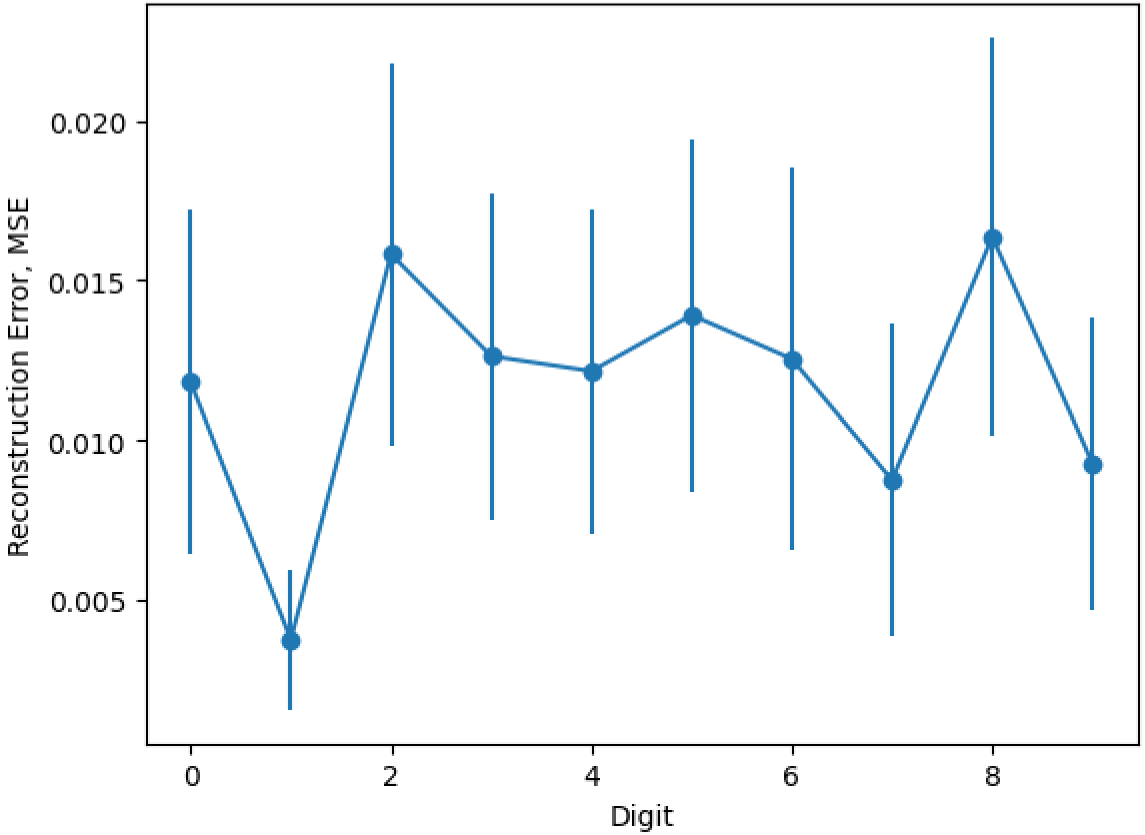
Epoch 10000

Original image



Reconstruction Error Distribution over Classes

- On test set (10000 samples), average and standard deviation of reconstruction error over different classes.
- This error can be used for classification with K-means
  - For classifying between 1 and 2, 89.6% accuracy
  - For 3 and 4, 51.5 %


Reconstruction Error, MSE
Digit