Autoencoder-based Dimensionality Reduction for Classification

Howell Xia – Nathan Lau – Tara Gill

Project Introduction

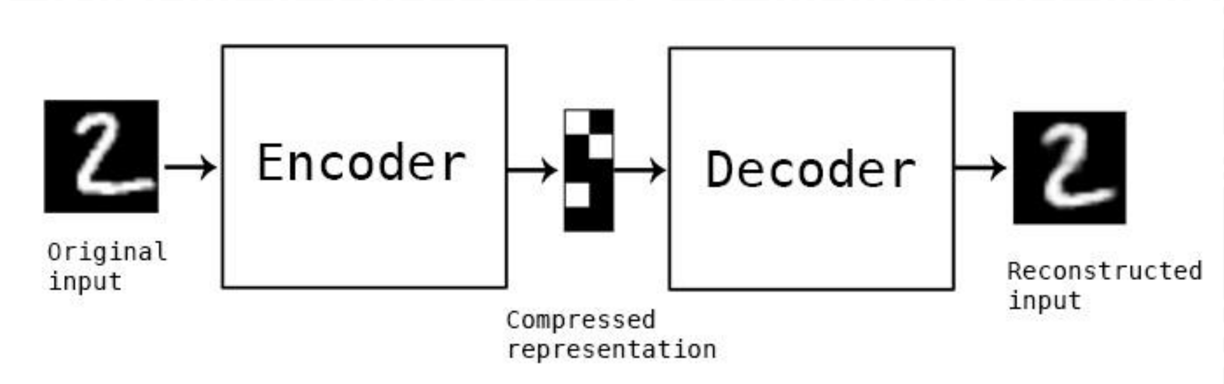
Background

Compression algorithms are used to reduce computational load, increase transfer rates, and optimize storage on devices…

…seeks to address the

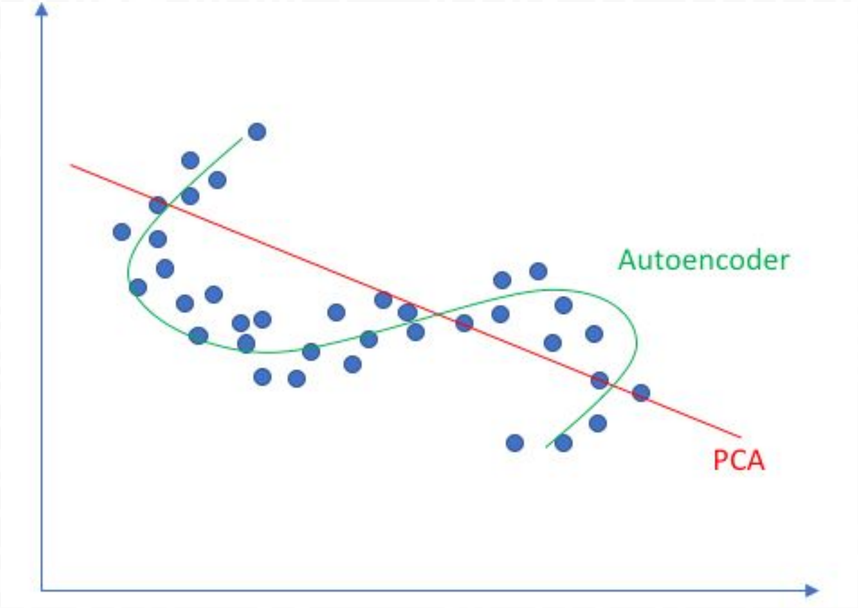
Curse of Dimensionality

Solution Approach: Autoencoders



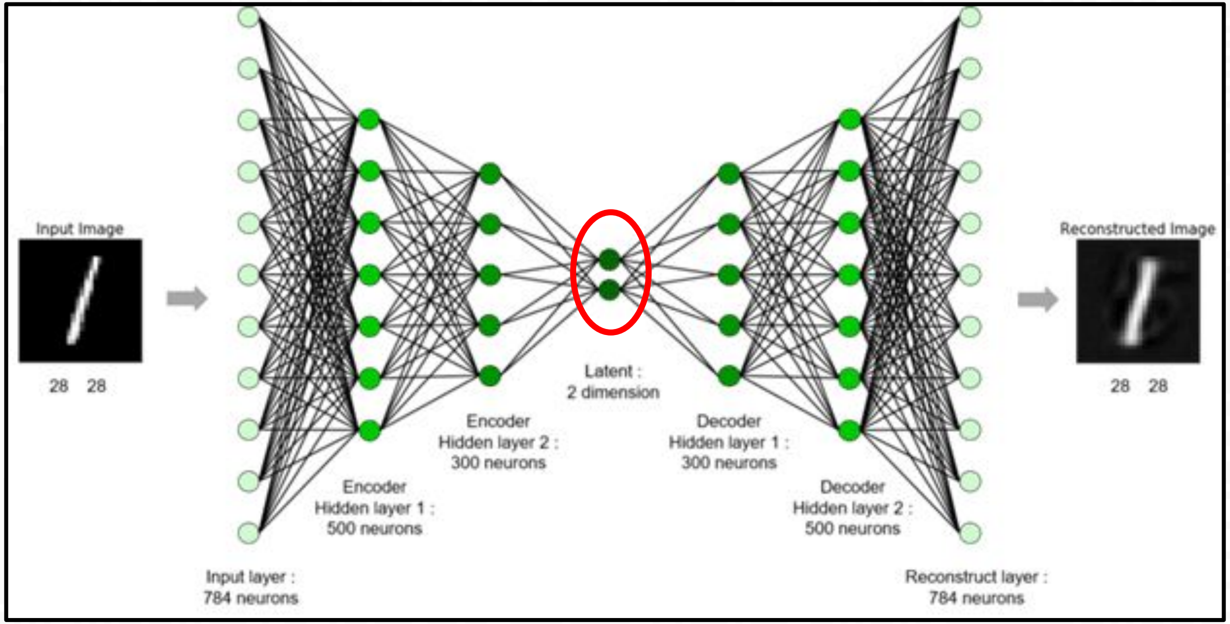
- Training assigns weights and biases to the neural network in cycles (epochs)
- The algorithm learns to reduce then reconstruct the image
- Typically: number of neurons in layer 1 = number of pixels

Solution Approach: Autoencoders



- Training assigns weights and biases to the neural network in cycles (epochs)
- The algorithm learns to reduce then reconstruct the image
- Typically: number of neurons in layer 1 = number of pixels

Solution Approach: Autoencoders



- Training assigns weights and biases to the neural network in cycles (epochs)
- The algorithm learns to reduce then reconstruct the image
- Typically: number of neurons in layer 1 = number of pixels

Project Goals

"How does appending a label matrix affect the **correct classification rate** (CCR) and **mean squared error** (MSE)"

"To what degree can we reduce an image to a **(k×1)** representational vector that will still maintain a **high correct classification rate** (CCR) and **low mean squared error** (MSE) between the following classes:"

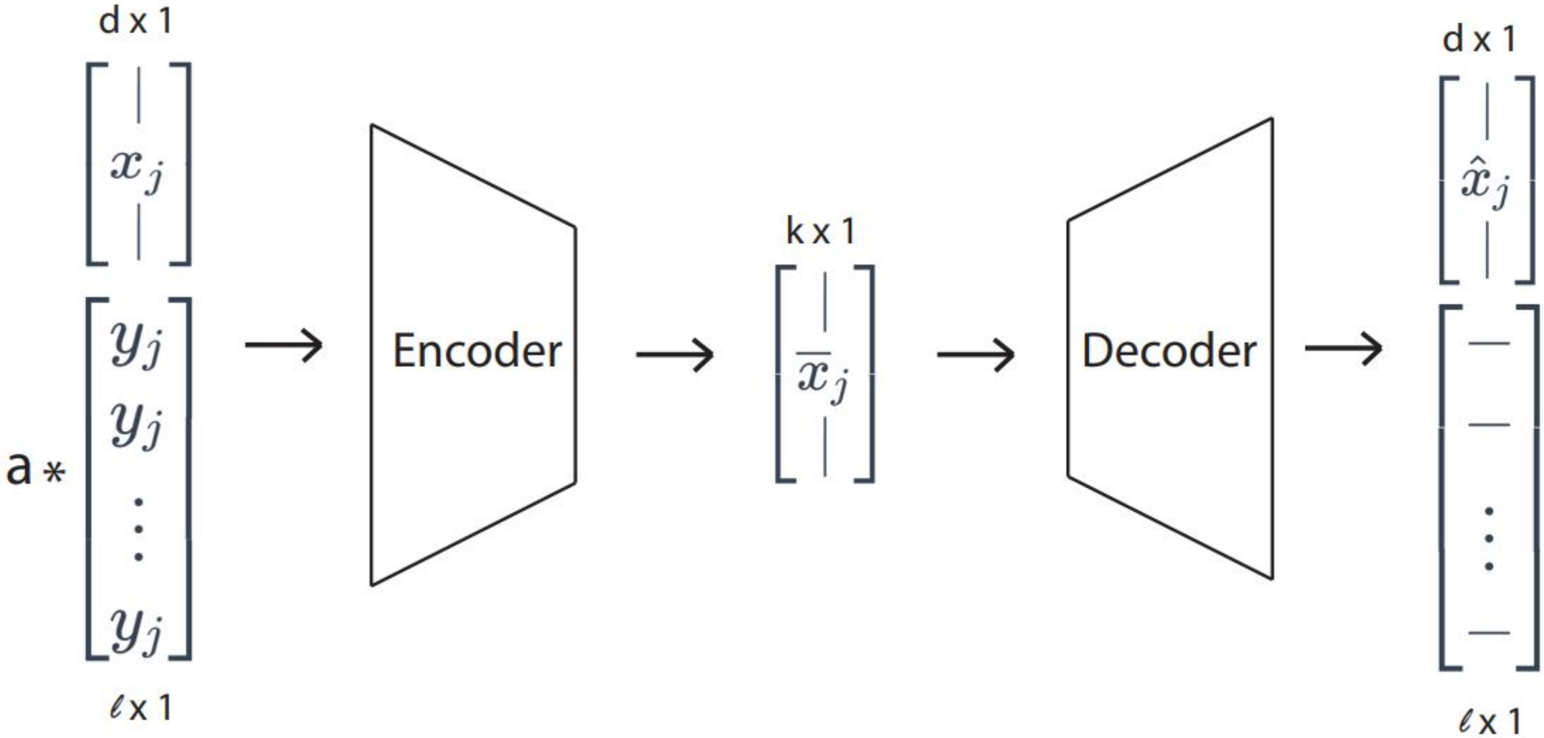
Dataset Feature Vector Characteristics

- **MNIST** Numerical Handwriting Data
  - Dimensions: 28 x 28px
  - Different digits have different labels

- Manageable Processing
  - Included only digits **3** and **8** for binary classification problem
  - ~10,000 training, ~2000 testing

Code Implementation

Code



Main Parameters: *L* and a

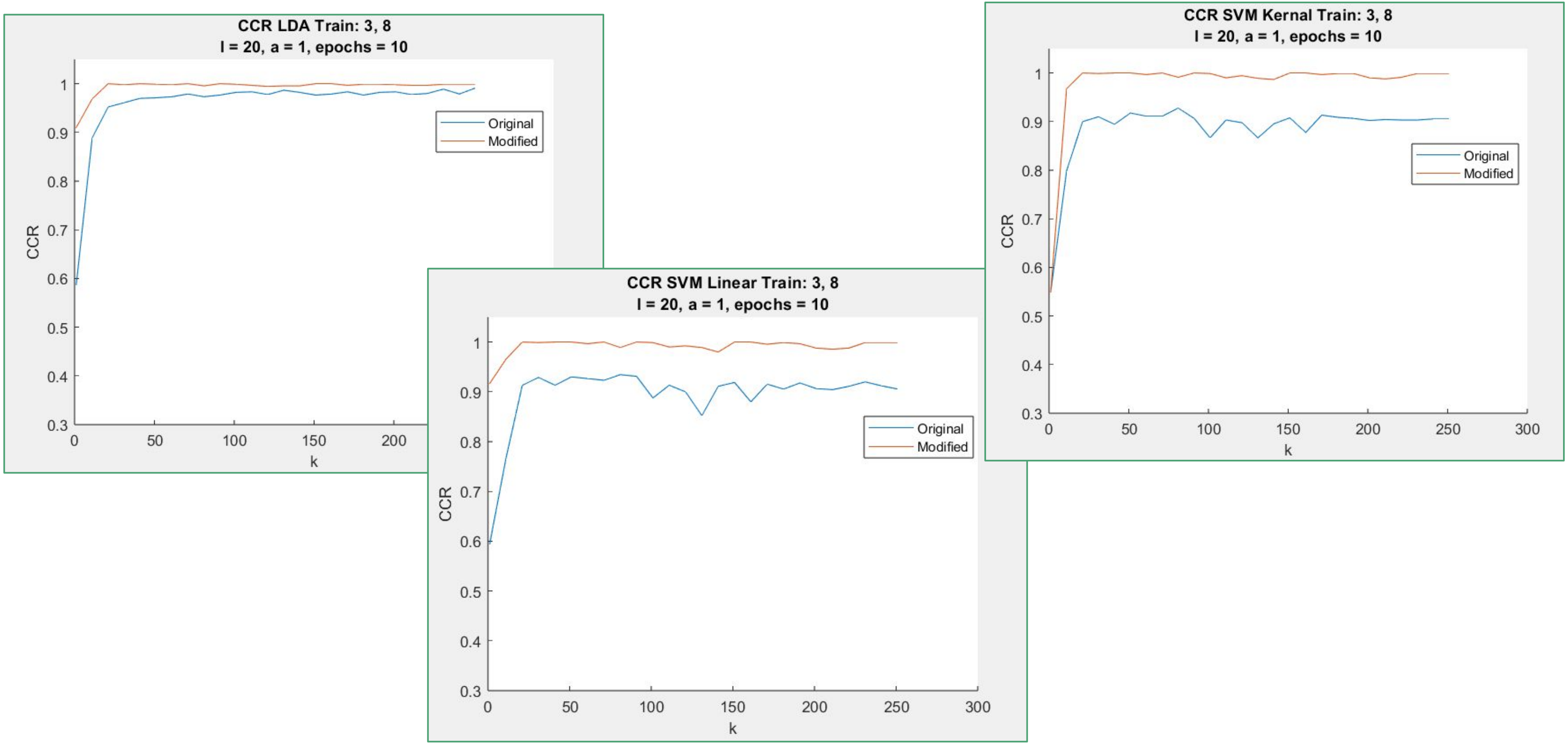
Code

$$
\overset{\text{k x (d + }\ell\text{)}}{\begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1,d+l} \\ W_{21} & W_{22} & \cdots & W_{2,d+l} \\ \vdots & \vdots & \ddots & \vdots \\ W_{k1} & W_{k2} & \cdots & W_{k,(d+l)} \end{bmatrix}}
*
\overset{\text{(d + }\ell\text{) x 1}}{\begin{bmatrix} | \\ x_{\text{test},j} \\ | \end{bmatrix}}
\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
=
\overset{\text{k x 1}}{\begin{bmatrix} | \\ \overline{x}_{\text{test},j} \\ | \end{bmatrix}}
$$

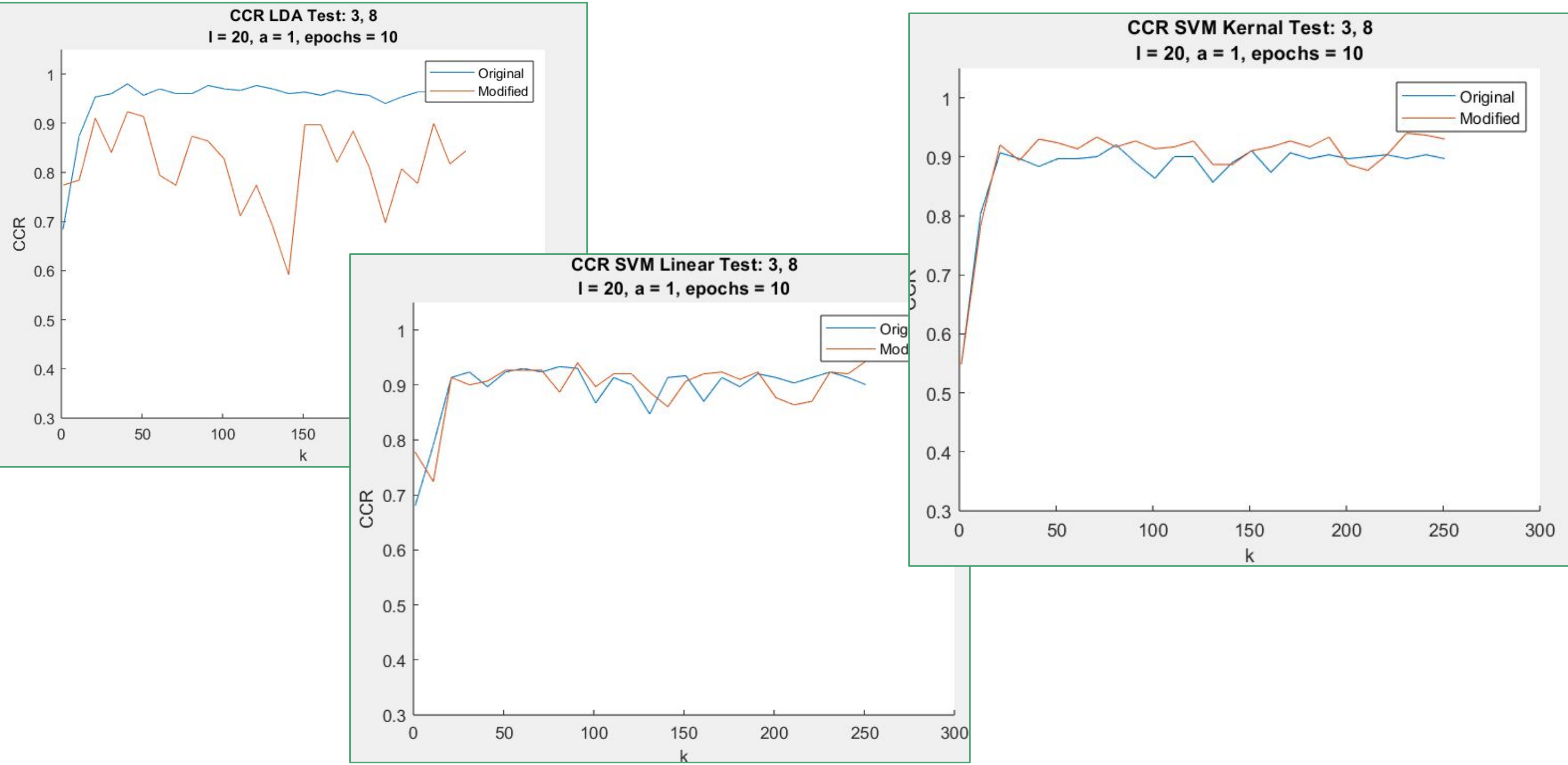
$$
\overset{\text{(d + }\ell\text{) x k}}{\begin{bmatrix} V_{11} & V_{12} & \cdots & V_{1k} \\ V_{21} & V_{22} & \cdots & V_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ V_{(d+l)1} & V_{(d+l)2} & \cdots & V_{(d+l)k} \end{bmatrix}}
*
\overset{\text{k x 1}}{\begin{bmatrix} | \\ \overline{x}_{\text{test},j} \\ | \end{bmatrix}}
=
\overset{\text{(d + }\ell\text{) x 1}}{\begin{bmatrix} | \\ \hat{x}_{\text{test},j} \\ | \end{bmatrix}}
\begin{bmatrix} - \\ - \\ \vdots \\ - \end{bmatrix}
$$

Experimental Results

Training Results with Different Binary Classifiers

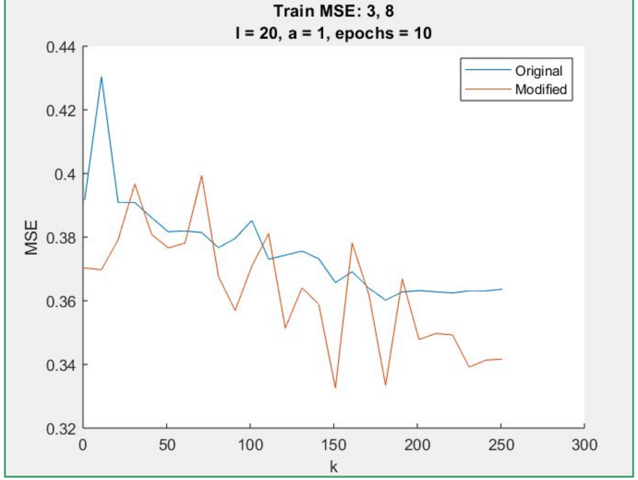


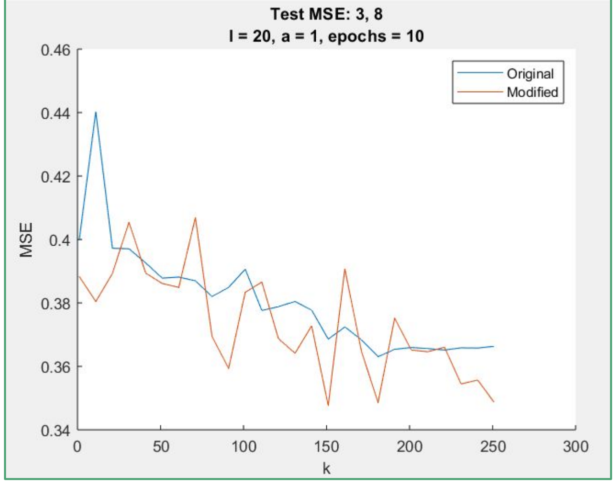
Testing CCR with Different Binary Classifiers

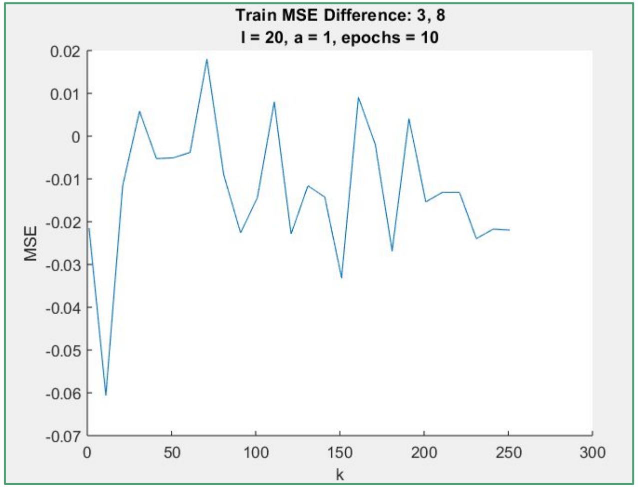


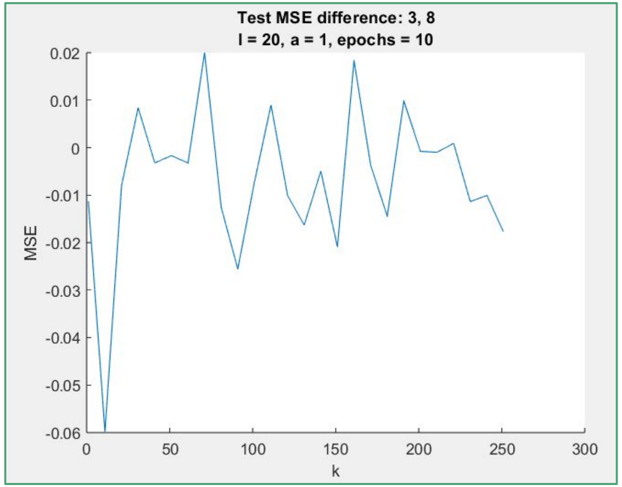
MSE

- Chaotic MSE due to low epochs
- Modified performs better than original
  - Low epochs
- Original has smoother pattern
  - Modified more chaotic due to appended dimensions

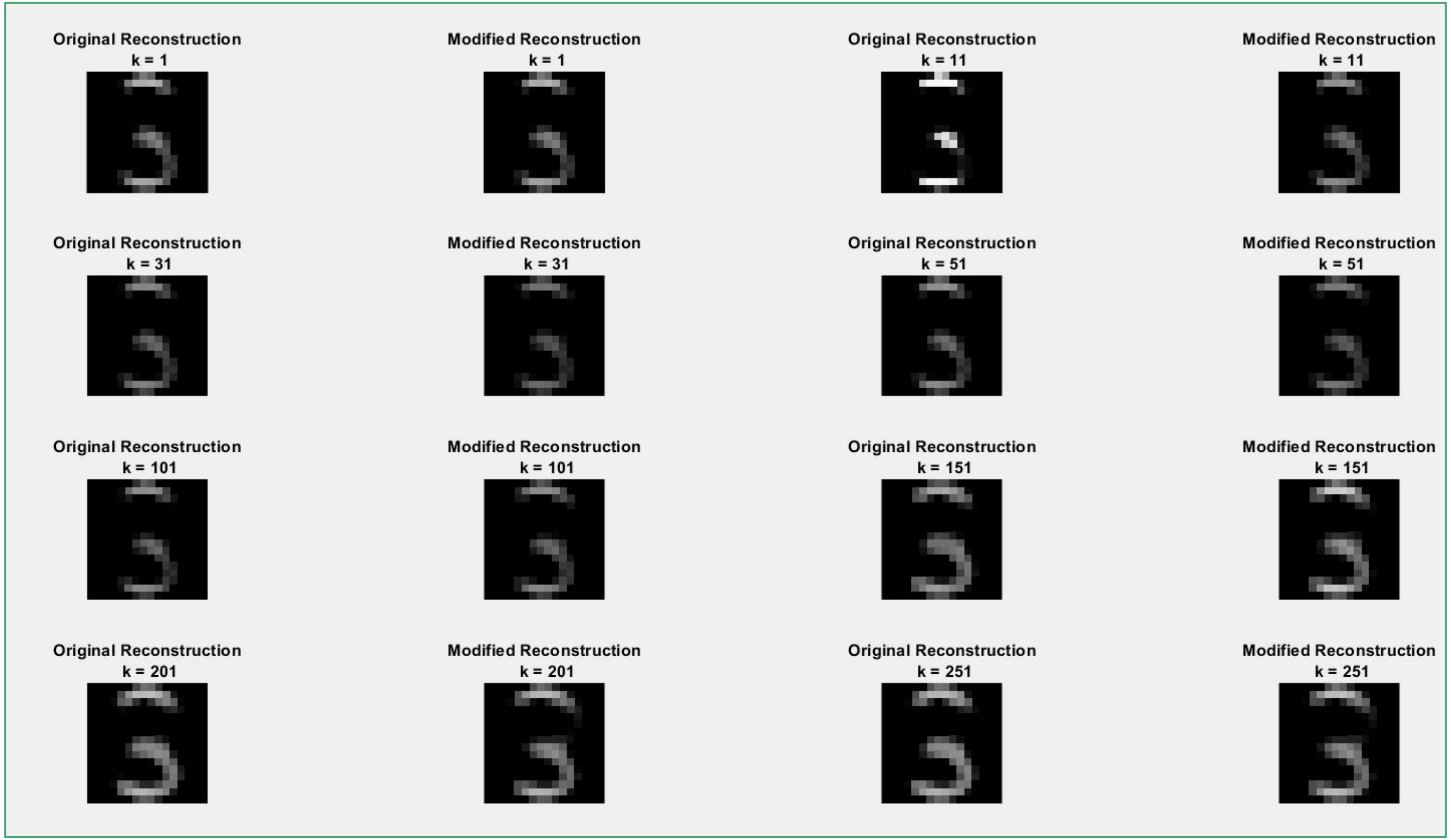


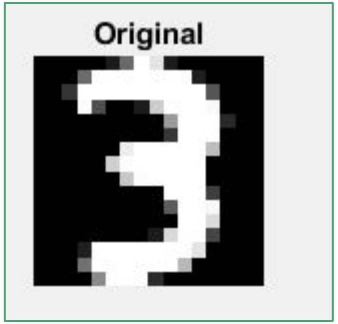




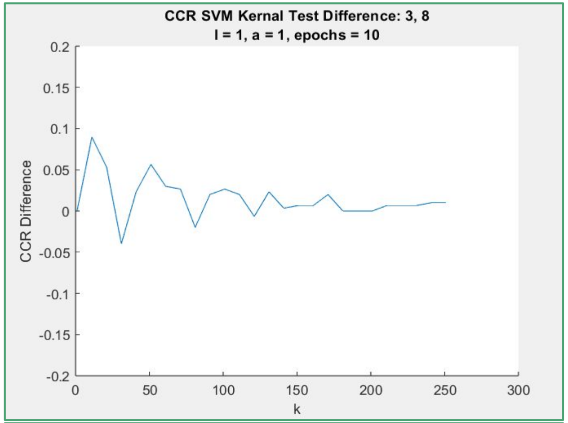


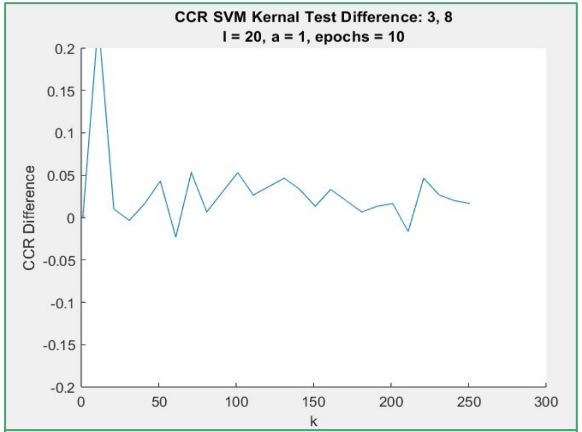
Reconstructed Images

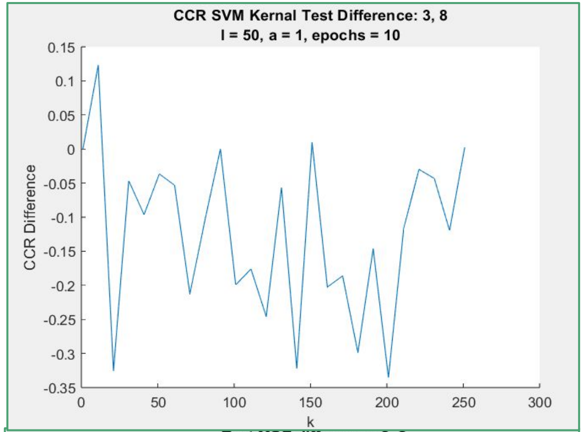


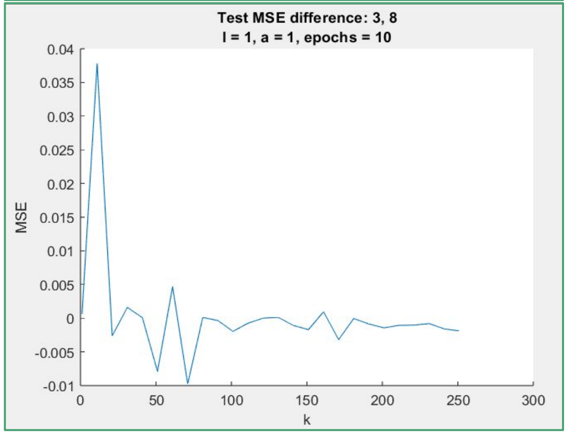


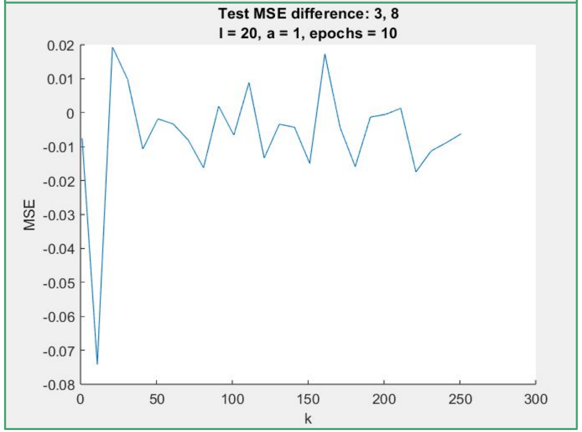
Adjusting *L*

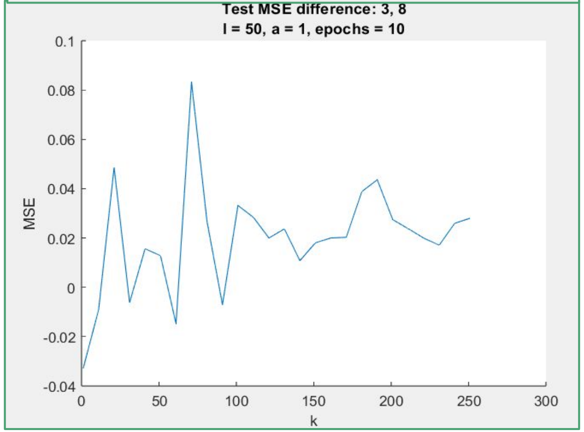

CCR SVM Kernal Test Difference: 3, 8
l = 1, a = 1, epochs = 10
CCR Difference
k


CCR SVM Kernal Test Difference: 3, 8
l = 20, a = 1, epochs = 10
CCR Difference
k

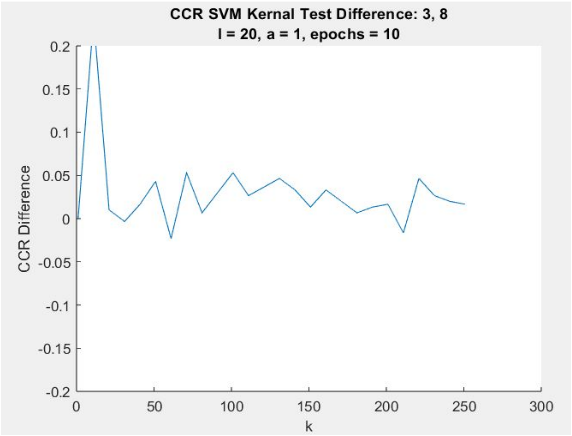

CCR SVM Kernal Test Difference: 3, 8
l = 50, a = 1, epochs = 10
CCR Difference
k

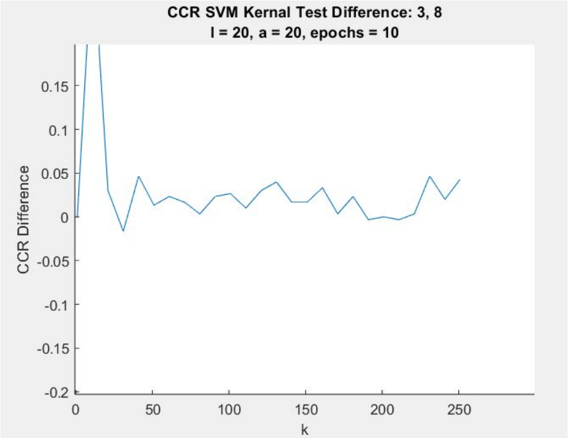

Test MSE difference: 3, 8
l = 1, a = 1, epochs = 10
MSE
k

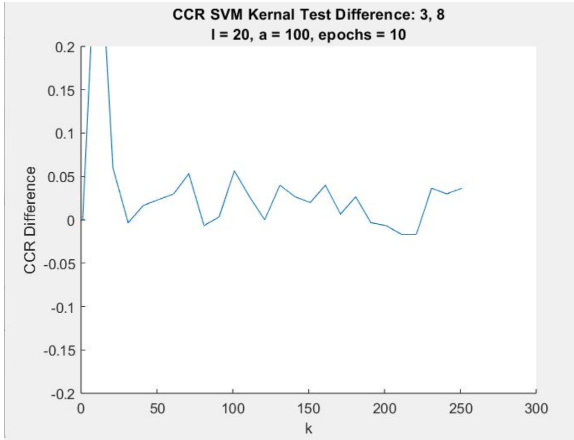

Test MSE difference: 3, 8
l = 20, a = 1, epochs = 10
MSE
k

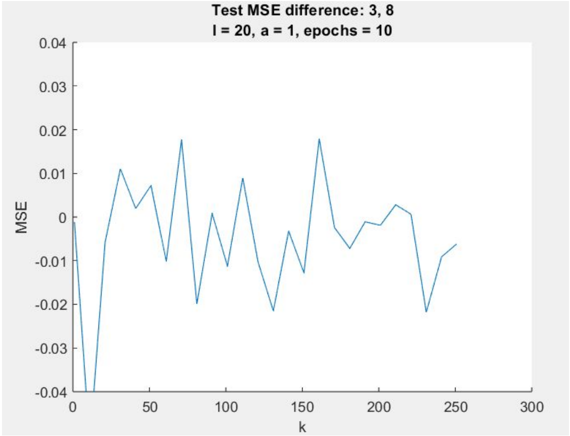

Test MSE difference: 3, 8
l = 50, a = 1, epochs = 10
MSE
k

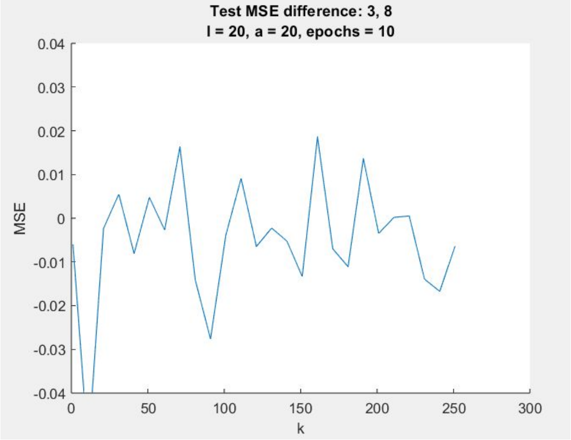
Adjusting *a*

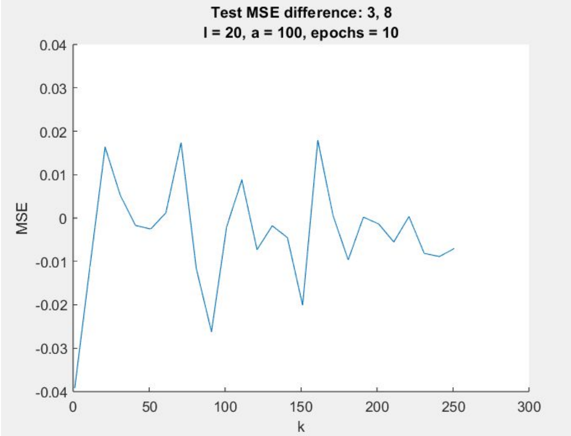

CCR SVM Kernal Test Difference: 3, 8
l = 20, a = 1, epochs = 10
CCR Difference
k


CCR SVM Kernal Test Difference: 3, 8
l = 20, a = 20, epochs = 10
CCR Difference
k


CCR SVM Kernal Test Difference: 3, 8
l = 20, a = 100, epochs = 10
CCR Difference
k

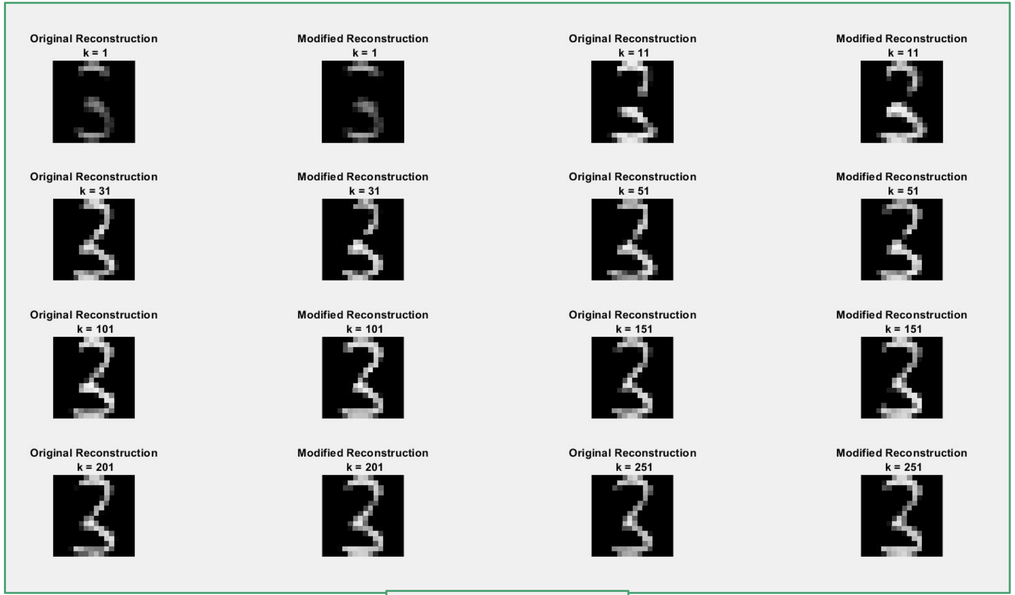

Test MSE difference: 3, 8
l = 20, a = 1, epochs = 10
MSE
k

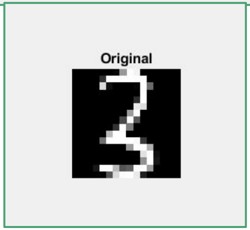

Test MSE difference: 3, 8
l = 20, a = 20, epochs = 10
MSE
k


Test MSE difference: 3, 8
l = 20, a = 100, epochs = 10
MSE
k

Some other parameters...

- Including 0's as labels in the training set
  - No effect
- Epochs
  - High Epochs -> Great MSE but complicated CCR behavior
- Data choice
  - Affects the results of different length and epochs





Conclusion

In Summary

- Only somewhat productive for certain length and epoch selections
  - trial and error
  - could be developed/better way to find ideal epoch and length values
- MSE results chaotic because low epochs
- Trade-off: higher epochs give better MSE but generally worse CCR
- Some of the odd/bad CCR behavior may be due to SVM kernel