22MAT230-Mathematics for Computing 3

Low-Rank Training of Neural Networks using SVD-Based Parameterization

Nimmagadda Kesav Satya Sai	CB.SC.U4AIE24236
Nakka Saampotth Maddileti	CB.SC.U4AIE24233
Gnana Vikas Sai Pabbathi	CB.SC.U4AIE24238
Vemula Poornachandra	CB.SC.U4AIE24258

Introduction

- Modern deep neural networks are memory heavy ,computationally expensive and difficult to deploy on edge devices, mobile / embedded systems
- Weight matrices are already low-rank but the rank of the wieght matrix is not controlled during the training itself
- In this project we will be training the neural networks in such a way that directly singular vectors and singular values will eb trained directly and pruned after every step such that only valuable eigen directions stay in the weight matrix. This helps reduce parameters

Existing Approaches

Post-training SVD

- Decompose pretrained weights
- Accuracy loss at high compression

Nuclear norm regularization

- Requires SVD every iteration

Filter / channel pruning

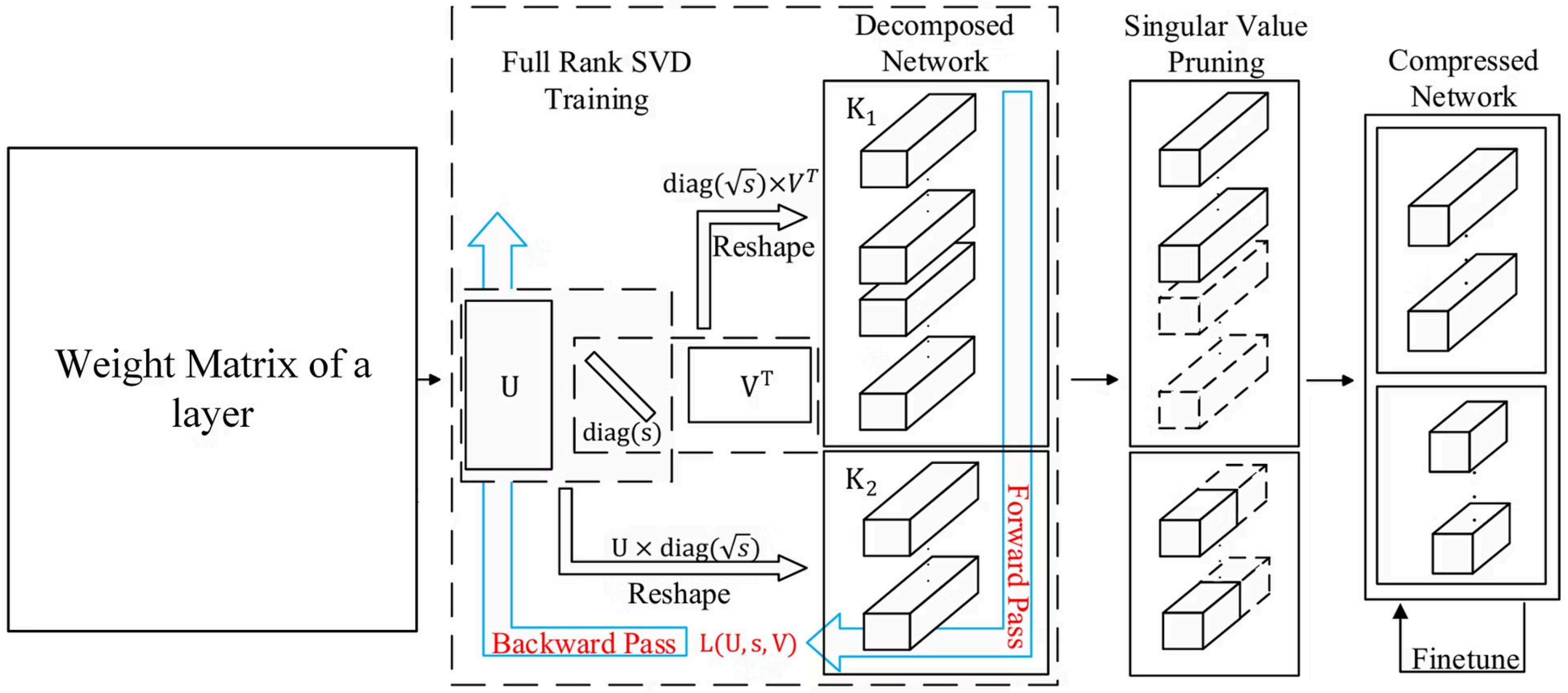
- Architecture constraints are very important for this, if the network is rnn or resnet kind then it is very hard

Our approach

- Represent each layer directly in SVD form
- Train singular vectors, singular values directly
- Enforce orthogonality on singular vectors and sparsity on singular values using regularizers to avoid exploding gradients and SVD functionalities

Main Idea

For a DNN, the process goes like this



Dataset Information

Name: MNIST

Total Images: 70000

Training Images: 60000 (already separated by the publishers)

Testing Images: 10000

Number of classes: 10(0-9 digits)

Image Size: 28×28

Standard DNN Architecture

For MNIST Dataset,

- **Input layer:**
  28 × 28 grayscale image will be flattened to a vector x (x belongs to R^784)
- **Hidden Layer 1:**
  Weight matrix W1 (W1 belongs to R^128×784)
  Activation: ReLU
- **Hidden Layer 2:**
  Weight matrix W2 (W2 belongs to R^64×128)
  Activation: ReLU
- **Output Layer:**
  Weight matrix W3 (W3 belongs to R^10×64)
  Activation: Softmax

Forward Pass

$$\begin{aligned} z_1 &= W_1 x + b_1, \quad a_1 = \mathrm{ReLU}(z_1) \\ z_2 &= W_2 a_1 + b_2, \quad a_2 = \mathrm{ReLU}(z_2) \\ z_3 &= W_3 a_2 + b_3, \quad \hat{y} = \mathrm{Softmax}(z_3) \end{aligned}$$

Backpropagation in Standard DNN

Loss Function used:
Cross-Entropy Loss

$$\mathcal{L} = -\sum_{k=1}^{10} y_k \log(\hat{y}_k)$$

Weights are updated using gradient descent

$$W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(l)}}$$

Limitations

- Large number of parameters
- High computational cost
- Redundant weights in fully connected layers

SVD modified DNN Architecture

Fully connected weight matrices are often low-rank

Singular Value Decomposition (SVD) enables:

  - Parameter reduction
  - FLOPs reduction
  - Model compression
- Instead of learning a full weight matrix W, each layer is parameterized directly using SVD components
- No SVD is performed during training
- U,S,V are initialized at the beginning and trained directly. But during forward pass we just multiply them in 2 layers

$$W = U\sqrt{\Sigma} \cdot \sqrt{\Sigma}V^T$$

Instead of Z=wx+b, we will be doing,

h=sqrt(S)*(V^T)*x

z=U*sqrt(S)*h + b

Loss Function in modified DNN

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_{orth}\left(\|U^TU - I\|_F^2 + \|V^TV - I\|_F^2\right) + \lambda_{sparse}\frac{\|\Sigma\|_1}{\|\Sigma\|_2}$$

Overall Loss Function= Cross entropy loss + orthogonality loss + Hoyer Regularizer

Cross Entropy is the normal training loss

Orthogonality Loss is introduced to preserve the SVD properties

Sparsity is forced in this method using Hoyer regularizer so that rank will become less which leads to low parameters

Backpropagation in SVD modified DNN

Weights are never trained directly, we backpropagate and train the U,S,V matrices directly from the feedback of loss function

The network trains:

- Left singular vectors, Singular values, Right singular vectors
- This enforces low-rank structure during training itself

Gradients are computed for loss function with respect to U,S,V(Respectively) and updated using gradient descent

Rank Pruning

After convergence, effective rank is selected using energy criterion

$$\sum_{j \in \mathbb{K}} s_j^2 \leq e \sum_{i=1}^{r} s_i^2$$

Corresponding columns of U and V are pruned accordingly

In the experiment performed, e=0.99

Progress

Current Progress~80%

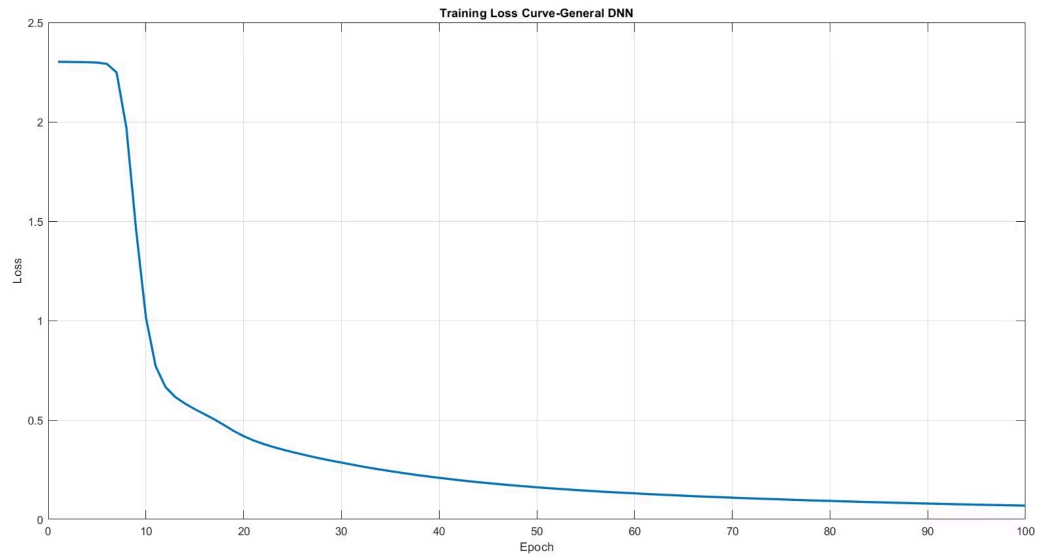
Will be applied on CNN for the next review

Dataset will be changed such that there will be images of bigger dimensions

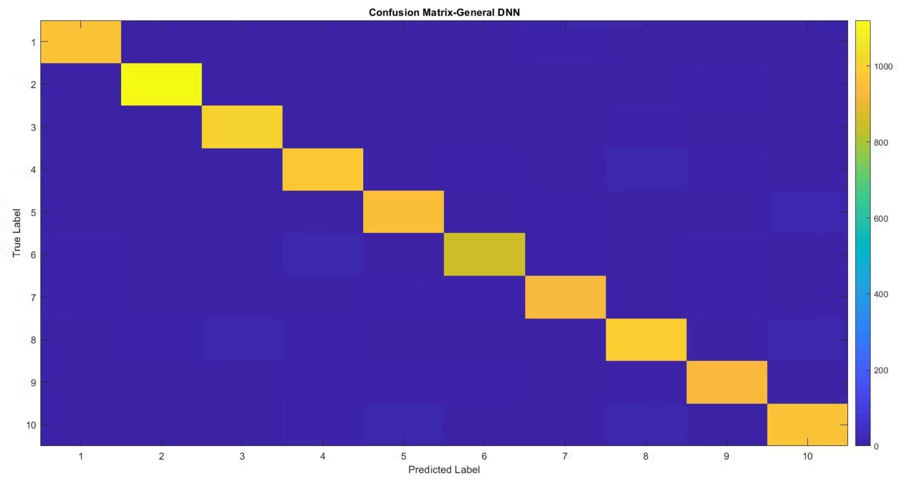
In-depth analysis will be done

Standard DNN Results

Accuracy: 97.10%



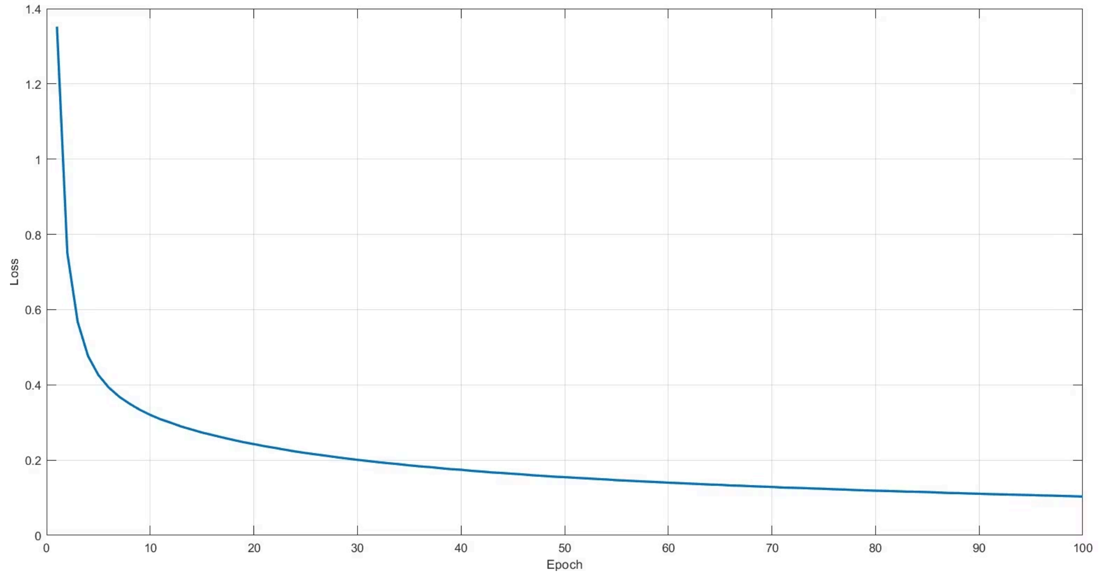
Training Loss Curve



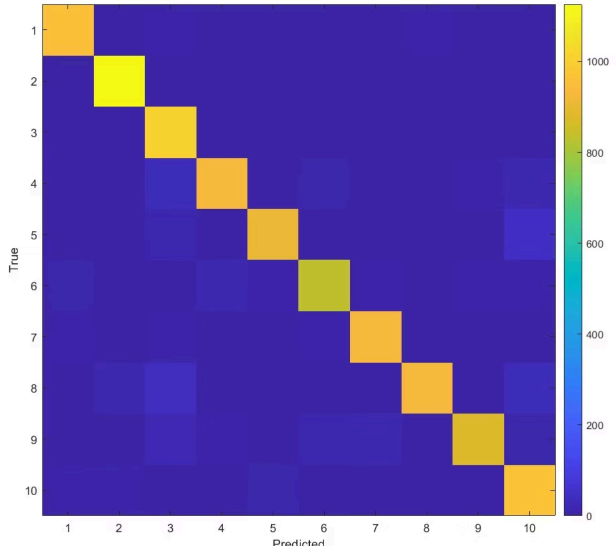
Confusion Matrix for 10 classes

SVD modified DNN result

Accuracy: 94.86%



Training Loss curve



Confusion matrix

Thank you