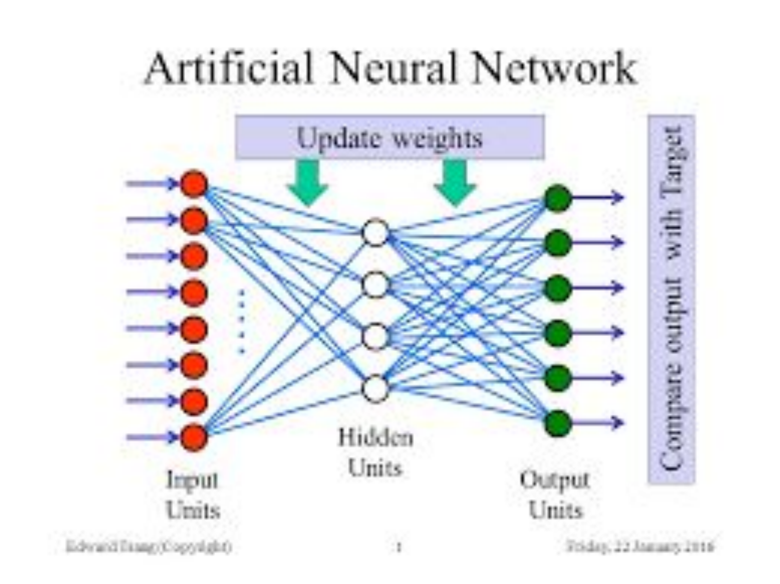
Pruning Neural Networks with Matrix Methods

Yajvan Ravan, Ben Ebanks

May 9, 2024

Background

- Neural Networks are the state-of-the-art models for tasks such as *classification*, *regression*, *generative modeling*, etc



- Most networks are very large
  - AlexNet (240Mb, 21 M parameters)
  - ChatGPT (500Gb, 175 B parameters)

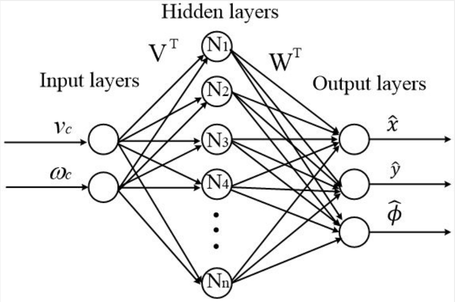
- Cons: High Inference Time, Need lots of compute, Limited applications in edge computing

- Pruning aims to reduce redundant parameters

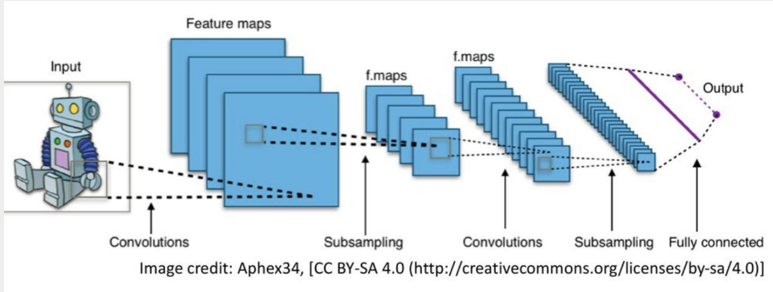
Methods

Fast Post-training Pruning

- Linear layer == Matrix Multiplication,
- Convolutional Layer ~ Matrix Multiplication
    - Not quite the same but it is a linear operation
- We can approximate each operation with techniques from this class
    - **PCA** for dimension reduction
    - **Low-rank approximation** for fast multiplication
    - **Randomized linear algebra** for fast multiplication



Linear Layer



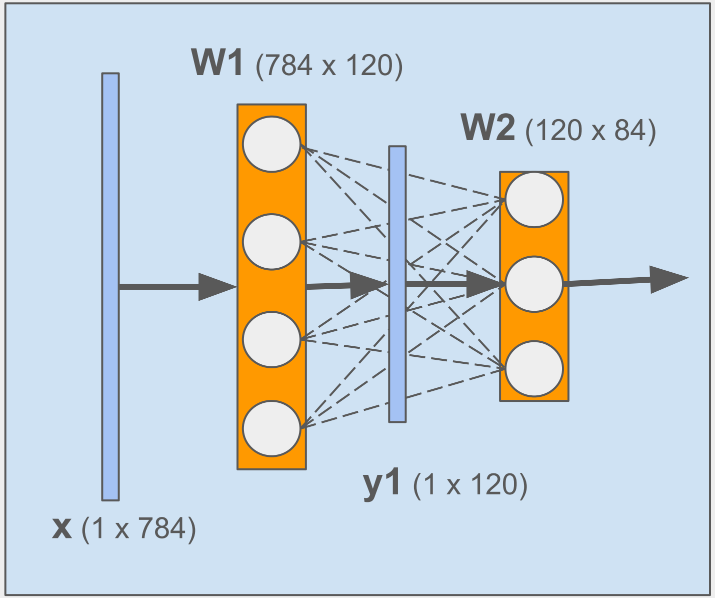
Convolutional Layer

PCA For Dimension Reduction

- **Goal**: Reduce the dimensionality of intermediate features

- **Expected Result:**
  - Applicable for taking high dimensional input to low dimensional output
  - Ex. Image Encoders, Image/Text Classification
  - Reduce the Size & Inference Time of the network

- **Procedure**:
  - PCA to reproject **W1** to a smaller output feature space
  - Preserve most of the variance between the rows of **W1**
  - Projection matrix: PCA(**W1**) => **P** (120 x 80)
  - Reproject
    - **W1* = W1 * P** (784 x 80)
    - **W2* = $\mathbf{P^T}$ * W2** (80 x 74)
  - Essentially, we have reprojected **y1* = y1 * P**



Neural Network

*Input: **x***

*Layer 1 output: **y1***

*Layer 1 Weight: **W1***

*Layer 2 Weight: **W2***

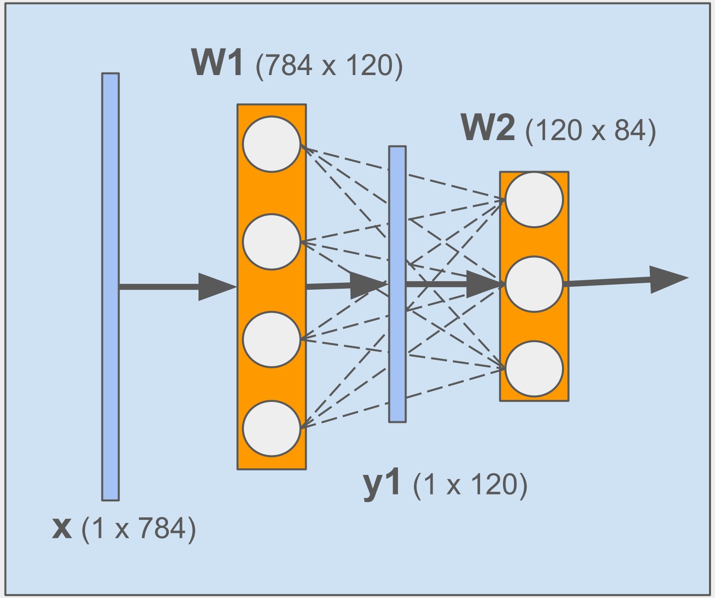
y1 = x * W1

Low-Rank Approximation

- **Goal**: Make Matrix Multiplication Faster

- **Expected Result:**
    - Inference time should be drastically reduced for networks with many redundant parameters

- **Procedure**:
    - Use SVD to compute a low rank approximation of **W1**
    - **W1 = U * V** where **U** (784 x k) & **V** (k x 120), *k << 120*
    - **y1 = x * W1 ===> y1 = x * U * V**
    - Computation time: 784 * 120 ===> 884 * k



Neural Network

Input: ***x***

Layer 1 output: ***y1***

Layer 1 Weight: ***W1***

Layer 2 Weight: ***W2***

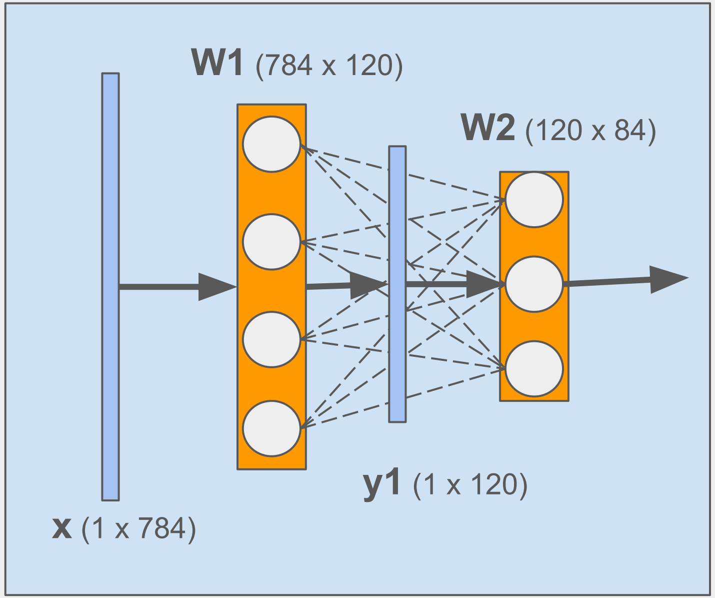
y1 = x * W1

Randomized Linear Algebra

- **Goal**: Make Matrix Multiplication Faster

- **Expected Result:**
    - Inference time should be drastically reduced for networks with many equivalent parameters

- **Procedure**:
    - Use Random Matrix Multiplication to do multiplication
    - **y1 = x * W1**
    - Computation: 784 * 120 ===> 120 * num_samples



<u>Neural Network</u>

*Input: **x***

*Layer 1 output: **y1***

*Layer 1 Weight: **W1***

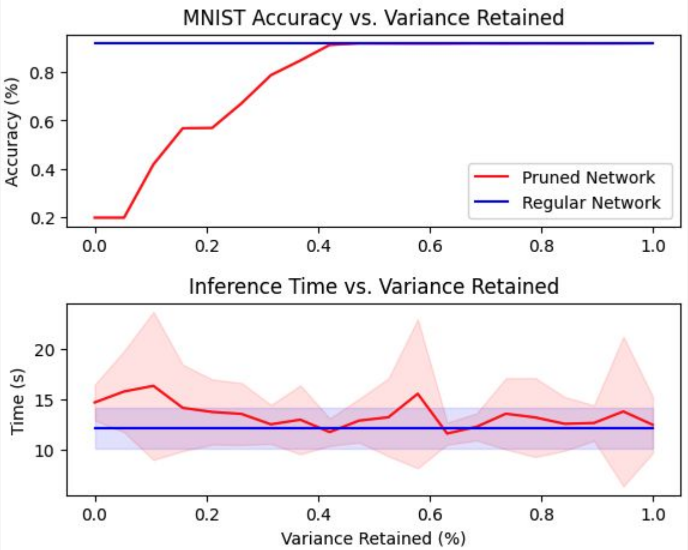
*Layer 2 Weight: **W2***

y1 = x * W1

Preliminary Results

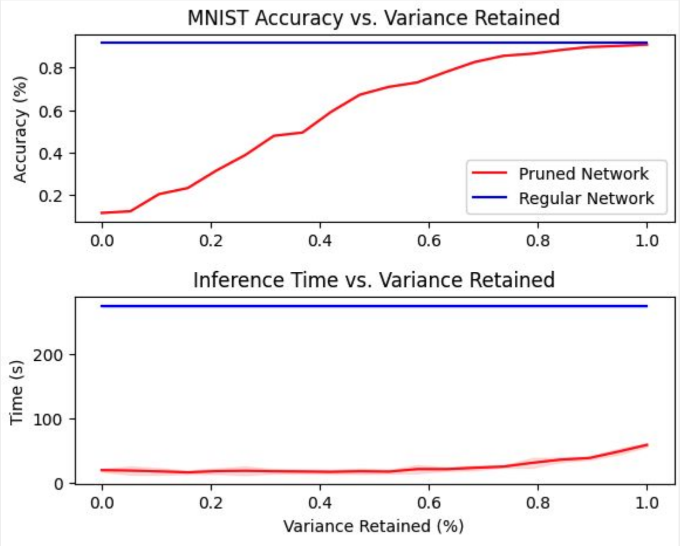
PCA:

Small Linear Network



Hidden Units: 120, 84

Large Linear Network:

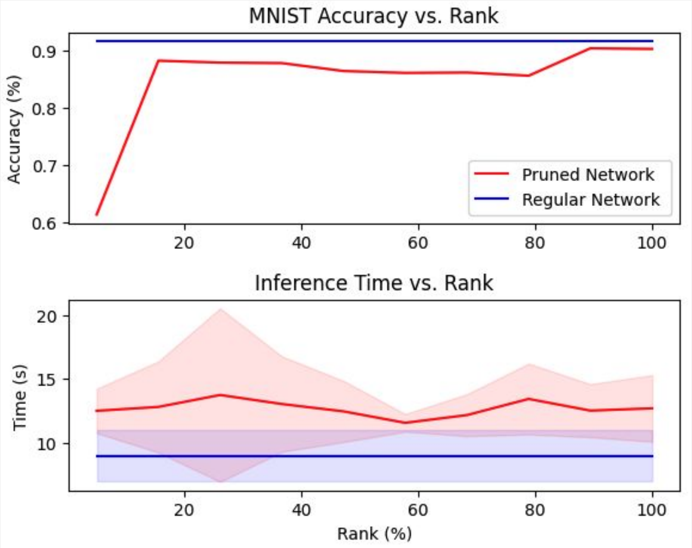


Hidden Units: 128, 512, 2048, 8192, 2048, 512, 128

Preliminary Results

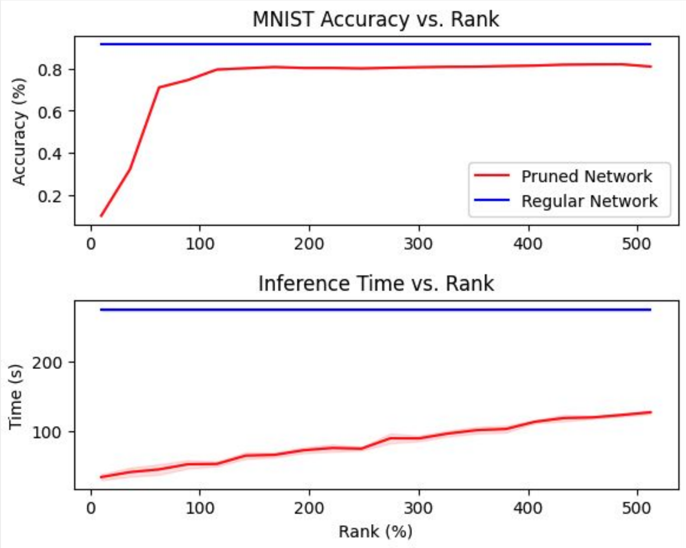
Low-Rank Approximation:

Small Linear Network



Hidden Units: 120, 84

Large Linear Network:



Hidden Units: 128, 512, 2048, 8192, 2048, 512, 128

References

1. Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'14). MIT Press, Cambridge, MA, USA, 1269–1277.
2. Garg, I., Panda, P. and Roy, K. (2020) ‘A low effort approach to structured CNN design using PCA’, IEEE Access, 8, pp. 1347–1360. doi:10.1109/access.2019.2961960.
3. Jaderberg, M., Vedaldi, A. and Zisserman, A. (2014) ‘Speeding up convolutional neural networks with low rank expansions’, Proceedings of the British Machine Vision Conference 2014 [Preprint]. doi:10.5244/c.28.88.
4. Levin, Asriel, Todd Leen, and John Moody. "Fast pruning using principal components." Advances in neural information processing systems 6 (1993).