Skoltech
NLA 2020 Final Project

Active Subspace of Neural Networks: Structural Analysis

Team: Passive Subspaces

Daniil Cherniavskii, Vanessa Skliarova, Aleksei Panchenko, Saveliy Galochkin

Problem statement

- **What?** - Neural network compression, using active subspace structural analysis
- **Why?** - Less memory, higher processing speed, more simple architecture
- **Hypothesis:** check that we can cut off network tail at some middle layer and approximate tail with a couple of much simpler layers with good compression rate and reasonable quality.
- **Applications** - use compressed models in applications.
- **How to measure quality?** - Compression rate - how much nnz parameters are compressed in ASNet + accuracy w/ fine tuning and w/o

Active Subspace of Neural Networks: Structural Analysis and Universal Attacks, Chunfeng Cui, Kaiqi Zhang, Talgat Daulbaev, Julia Gusak, Ivan Oseledets, Zheng Zhang

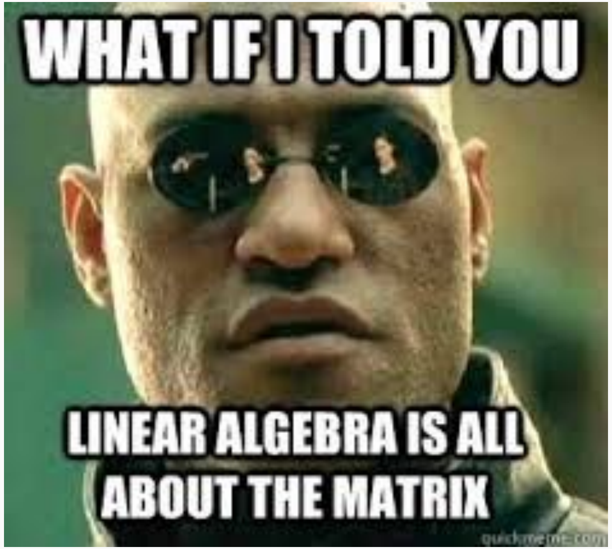
Neural Network Compression Methods

- Weight Sharing
- Network Pruning
- Low Rank Matrix & Tensor Decompositions
- Knowledge Distillation
- Quantization

An Survey of Neural Network Compression https://arxiv.org/pdf/2006.03669.pdf

Neural Network Compression Techniques

- Weight Sharing
- Network Pruning
- **Low Rank Matrix & Tensor Decompositions**
- Knowledge Distillation
- Quantization



An Survey of Neural Network Compression https://arxiv.org/pdf/2006.03669.pdf

Major Points

- Activate Subspace
- Structural analysis and compression of deep neural networks
- Active subspace network (ASNet)
  - The active subspace layer
  - Polynomial chaos expansion layer
- The training procedure of the ASNet
- Experimental Results
- Conclusion

Active Subspace

c(x) - continuous function

$$C = \mathbb{E}[\nabla c(x) \nabla c(x)^T]$$

$$C = V \Lambda V^T$$

$$\Lambda = diag(\lambda_1, ..., \lambda_n), \lambda_1 \geq \cdots \geq \lambda_n \geq 0$$

$$V = [V_1, V_2], \text{where } V_1 \in \mathbb{R}^{n \times r} \text{ and } V_2 \in \mathbb{R}^{n \times (n-r)}$$

Active Subspace

c(x) - continuous function

$$C = \mathbb{E}[\nabla c(x) \nabla c(x)^T]$$

$$C = V \Lambda V^T$$

$$\Lambda = diag(\lambda_1, \ldots, \lambda_n), \lambda_1 \geq \cdots \geq \lambda_n \geq 0$$

$$V = [V_1, V_2], \text{where } V_1 \in \mathbb{R}^{n \times r} \text{ and } V_2 \in \mathbb{R}^{n \times (n-r)}$$

Active Subspace

Lemma 2.2 (see [10]). *Suppose $c(\mathbf{x})$ is a continuous function and $\mathbf{C}$ is obtained from (2.1). For the matrices $\mathbf{V}_1$ and $\mathbf{V}_2$ generated by (2.3), and the reduced vector*

$$\mathbf{z} = \mathbf{V}_1^T\mathbf{x} \text{ and } \tilde{\mathbf{z}} = \mathbf{V}_2^T\mathbf{x}, \tag{2.5}$$

it holds that

$$\begin{aligned} \mathbb{E}_{\mathbf{x}}[\nabla_{\mathbf{z}} c(\mathbf{x})^T \nabla_{\mathbf{z}} c(\mathbf{x})] &= \lambda_1 + \cdots + \lambda_r, \\ \mathbb{E}_{\mathbf{x}}[\nabla_{\tilde{\mathbf{z}}} c(\mathbf{x})^T \nabla_{\tilde{\mathbf{z}}} c(\mathbf{x})] &= \lambda_{r+1} + \cdots + \lambda_n. \end{aligned} \tag{2.6}$$

Active Subspace

2.1. Response surface. For a fixed $\mathbf{z}$, the best guess for g is the conditional expectation of c given $\mathbf{z}$, i.e.,

$$g(\mathbf{z}) = \mathbb{E}_{\tilde{\mathbf{z}}}[c(\mathbf{x})|\mathbf{z}] = \int c(\mathbf{V}_1\mathbf{z} + \mathbf{V}_2\tilde{\mathbf{z}})\rho(\tilde{\mathbf{z}}|\mathbf{z})d\tilde{\mathbf{z}}. \tag{2.7}$$

Based on the Poincaré inequality, the following approximation error bound is obtained [10].

Lemma 2.3. *Assume that $c(\mathbf{x})$ is absolutely continuous and square integrable with respect to the probability density function $\rho(\mathbf{x})$; then the approximation function $g(\mathbf{z})$ in (2.7) satisfies*

$$\mathbb{E}[(c(\mathbf{x}) - g(\mathbf{z}))^2] \leq O(\lambda_{r+1} + \cdots + \lambda_n). \tag{2.8}$$

Structural analysis and compression of deep neural networks

$$f(x_0) = f_L(f_{L-1}\dots(f_1(x_0)))$$

$$f(x_0) = f^l_{post}(f^l_{pre}(x_0))$$

Definition 3.1. *Suppose* $\mathbf{\Lambda}$ *is computed by* (2.2). *For any layer index* $1 \leq l \leq L$, *we define the number of active neurons* $n_{l,AS}$ *as follows:*

$$n_{l,AS} = \arg\min \left\{ i : \frac{\lambda_1 + \ldots + \lambda_i}{\lambda_1 + \ldots + \lambda_{n_l}} \geq 1 - \epsilon \right\}, \tag{3.4}$$

where $\epsilon > 0$ *is a user-defined threshold.*

Active subspace network (ASNet)

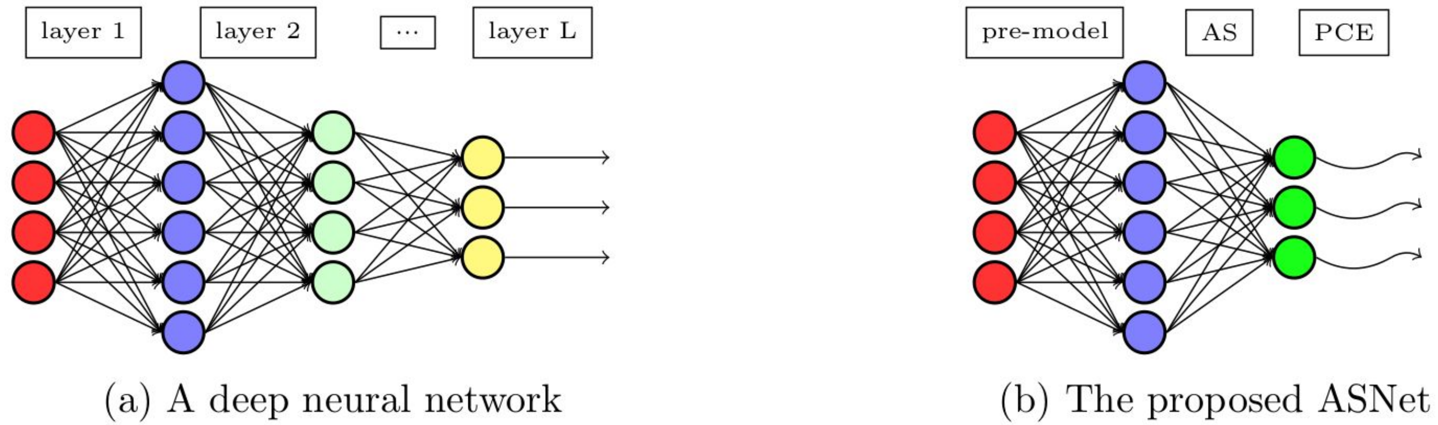


Figure 2. (a) *The original deep neural network.* (b) *The proposed ASNet with three parts: a pre-model, an active subspace (AS) layer, and a polynomial chaos expansion (PCE) layer.*

The active subspace layer

Algorithm 3.2. The frequent direction algorithm for computing the active subspace.

Input: A dataset with m_{AS} input samples $\{\mathbf{x}_0^j\}_{j=1}^{m_{AS}}$, a pre-model $f_{\text{pre}}^l(\cdot)$, a subroutine for computing $\nabla c_l(\mathbf{x})$, and the dimension of truncated singular value decomposition r.

Select r samples $\mathbf{x}_0^i$, compute $\mathbf{x}^i = f_{\text{pre}}^l(\mathbf{x}_0^i)$, and construct an initial matrix $\mathbf{S} \leftarrow [\nabla c_l(\mathbf{x}^1), \ldots, \nabla c_l(\mathbf{x}^r)]$.

for t=1, 2, ..., **do**

Compute the singular value decomposition $\mathbf{V\Sigma U}^T \leftarrow \text{svd}(\mathbf{S})$, where $\mathbf{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_r)$.

If the maximal number of samples m_{AS} is reached, stop.

$$\mathbf{S} \leftarrow \mathbf{V}\sqrt{\mathbf{\Sigma}^2 - \sigma_r^2}$$

Update $\mathbf{S}$ by the soft-thresholding (3.8).

Get a new sample $\mathbf{x}_0^{\text{new}}$, compute $\mathbf{x}^{\text{new}} = f_{\text{pre}}^l(\mathbf{x}_0^{\text{new}})$, and replace the last column of $\mathbf{S}$ (now all zeros) by the gradient vector $\mathbf{S}(:, r) \leftarrow \nabla c_l(\mathbf{x}^{\text{new}})$.

end for

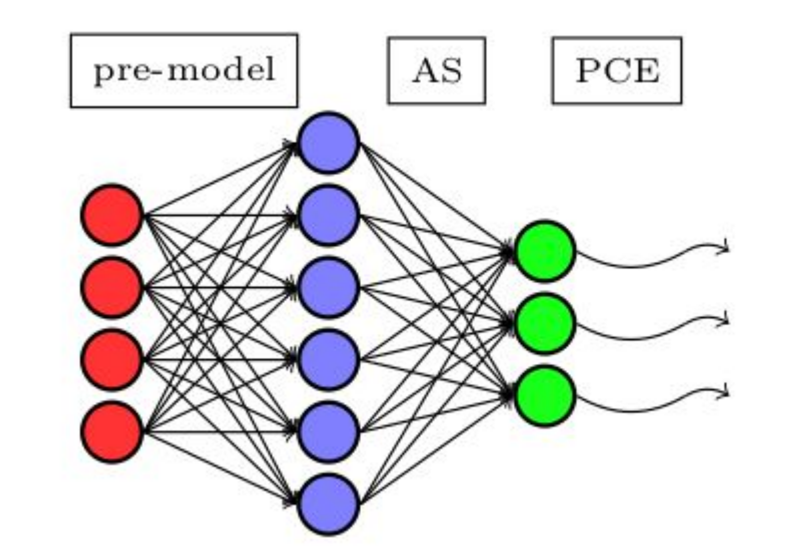
Output: The projection matrix $\mathbf{V} \in \mathbb{R}^{n_l \times r}$ and the singular values $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$.

$$\boxed{\hat{n}_{l,AS} = \arg\min \left\{ i : \frac{\sqrt{\sigma_1^2 + \cdots + \sigma_i^2}}{\sqrt{\sigma_1^2 + \cdots + \sigma_r^2}} \geq 1 - \epsilon \right\}.}$$

Polynomial Chaos Expansion Layer

$$\hat{\mathbf{y}} \approx \sum_{|\boldsymbol{\alpha}|=0}^{p} \mathbf{c}_{\boldsymbol{\alpha}} \phi_{\boldsymbol{\alpha}}(\mathbf{z}), \text{ where } |\boldsymbol{\alpha}| = \alpha_1 + \cdots + \alpha_d.$$

$$\min_{\{\mathbf{c}_{\boldsymbol{\alpha}}\}} \quad \frac{1}{m_{\text{PCE}}} \sum_{j=1}^{m_{\text{PCE}}} \|\mathbf{y}^j - \sum_{|\boldsymbol{\alpha}|=0}^{p} \mathbf{c}_{\boldsymbol{\alpha}} \phi_{\boldsymbol{\alpha}}(\mathbf{z}^j)\|^2.$$



The training procedure of the ASNet

Algorithm 3.1. The training procedure of the ASNet.

Input: A pretrained deep neural network, the layer index l, and the number of active neurons r.

Step 1 **Initialize the active subspace layer.** The active subspace layer is a linear projection where the projection matrix $\mathbf{V}_1 \in \mathbb{R}^{n \times r}$ is computed by Algorithm 3.2.
If r is not given, we use $r = n_{\text{AS}}$ defined in (3.4) by default.

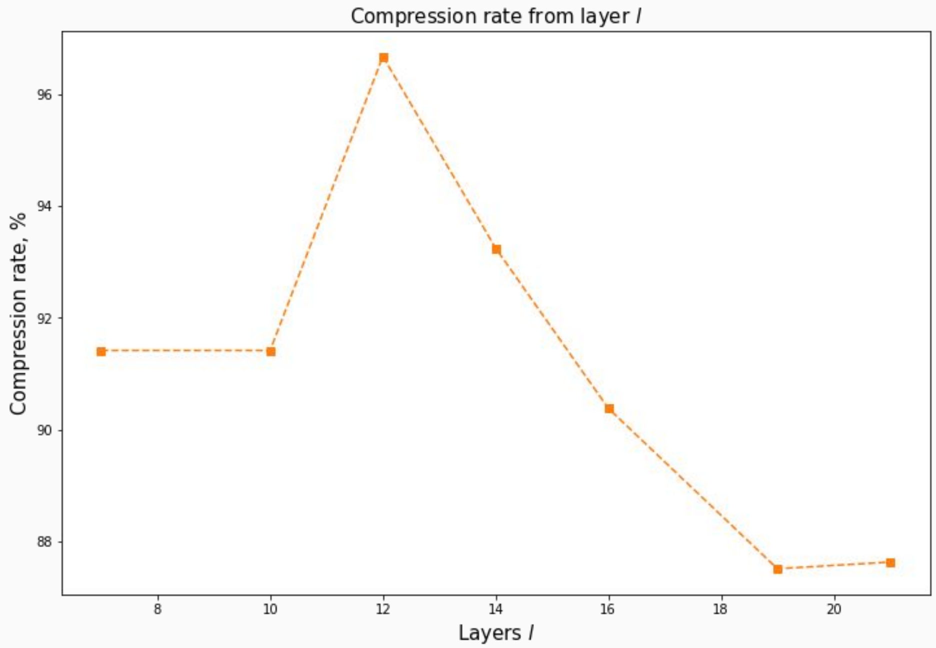
Step 2 **Initialize the polynomial chaos expansion layer.** The polynomial chaos expansion layer is a nonlinear mapping from the reduced active subspace to the outputs, as shown in (3.10). The weights $\mathbf{c}_{\boldsymbol{\alpha}}$ is computed by (3.12).

Step 3 **Construct the ASNet.** Combine the pre-model (the first l layers of the deep neural network) with the active subspace and polynomial chaos expansion layers as a new network, referred to as ASNet.

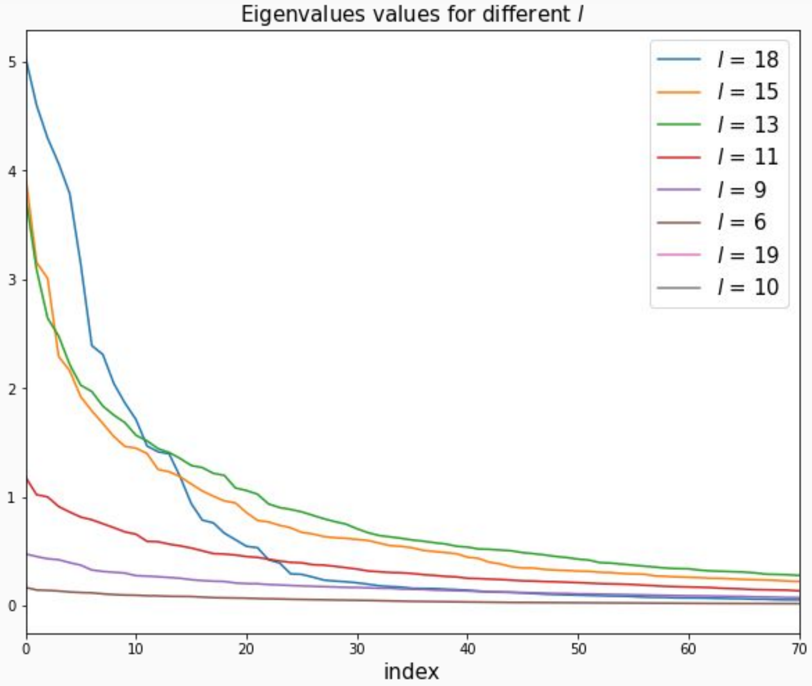
Step 4 **Fine-tuning.** Retrain all the parameters in pre-model, active subspace layer, and polynomial chaos expansion layer in ASNet for several epochs by a proper (variant of) the stochastic gradient descent algorithm.

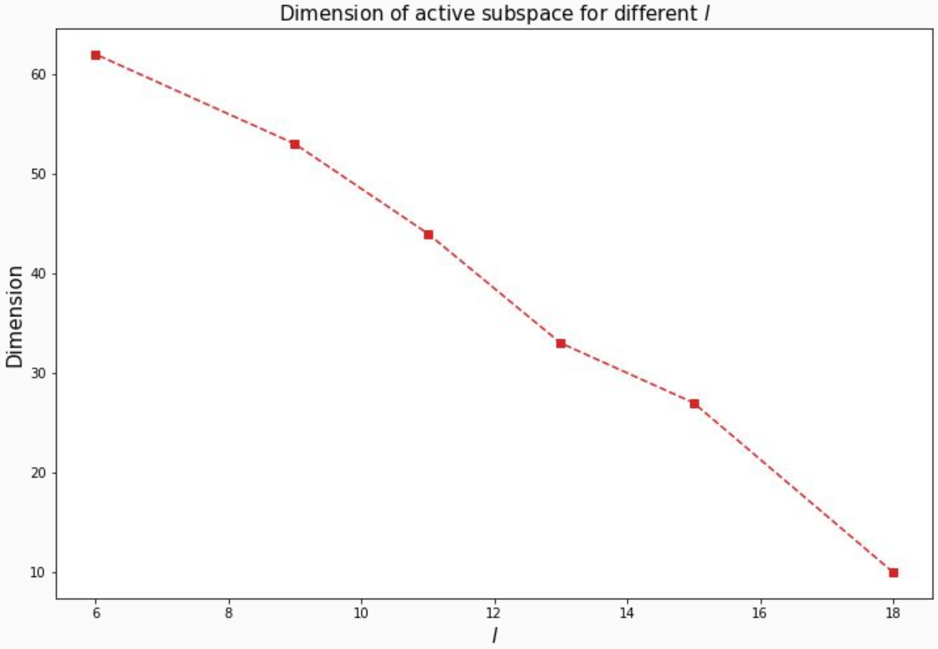
Output: A new network ASNet

Experimental Results: VGG19

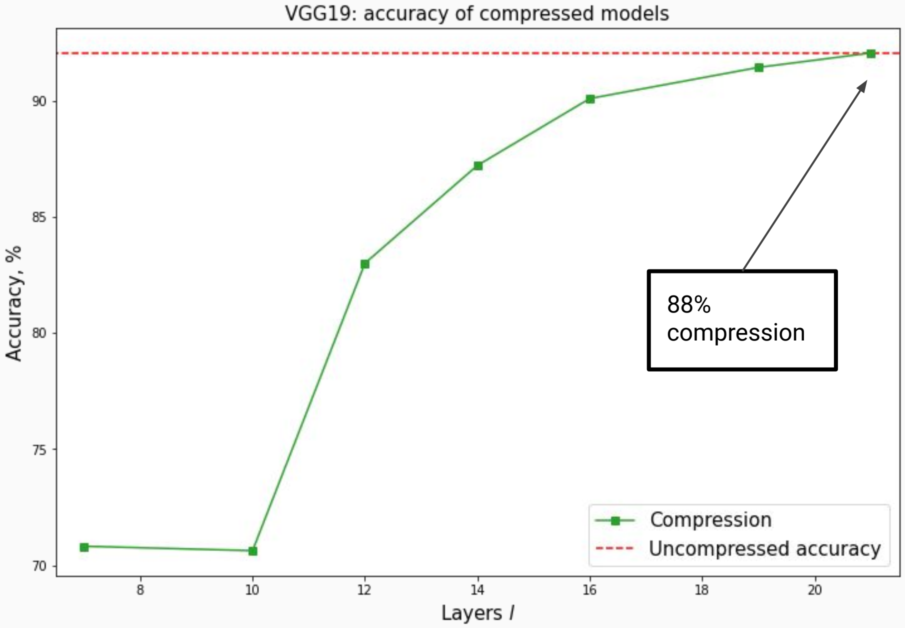


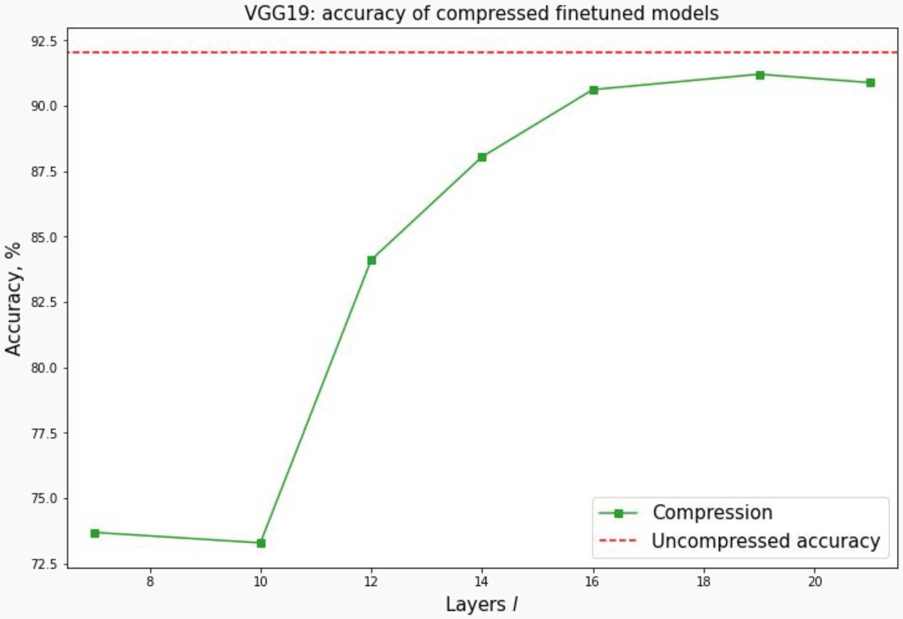
Experimental Results: VGG19



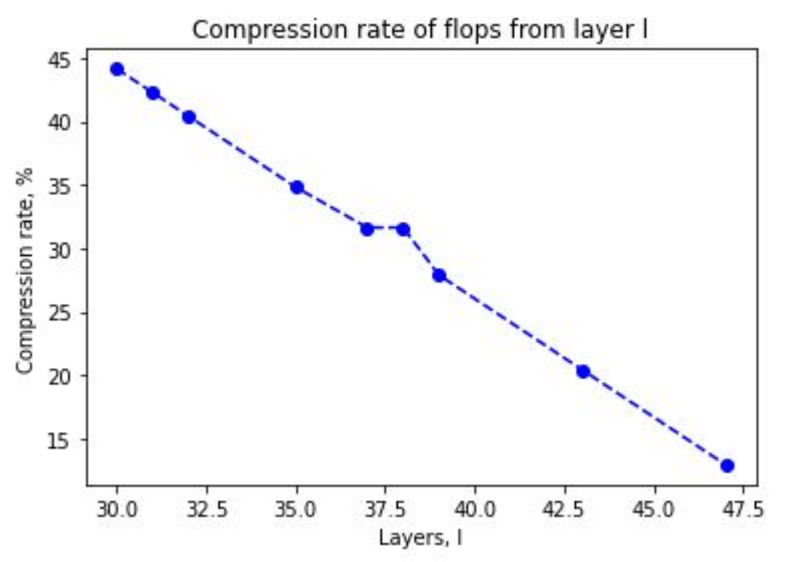


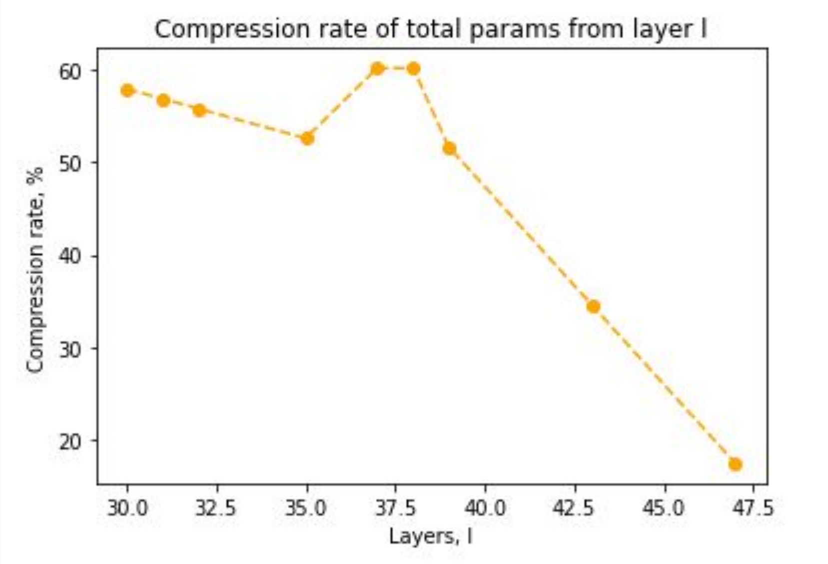
Experimental Results: VGG19



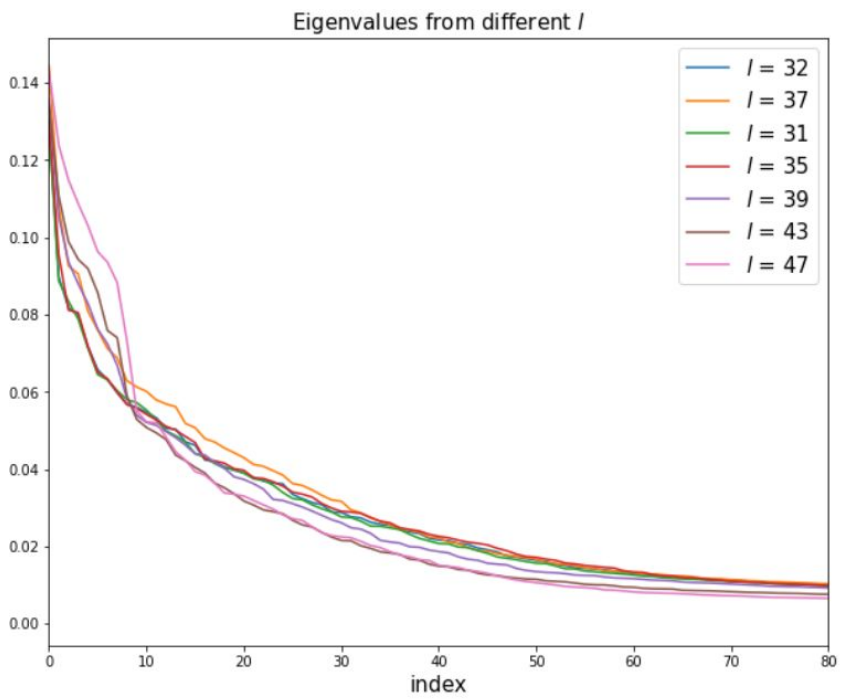


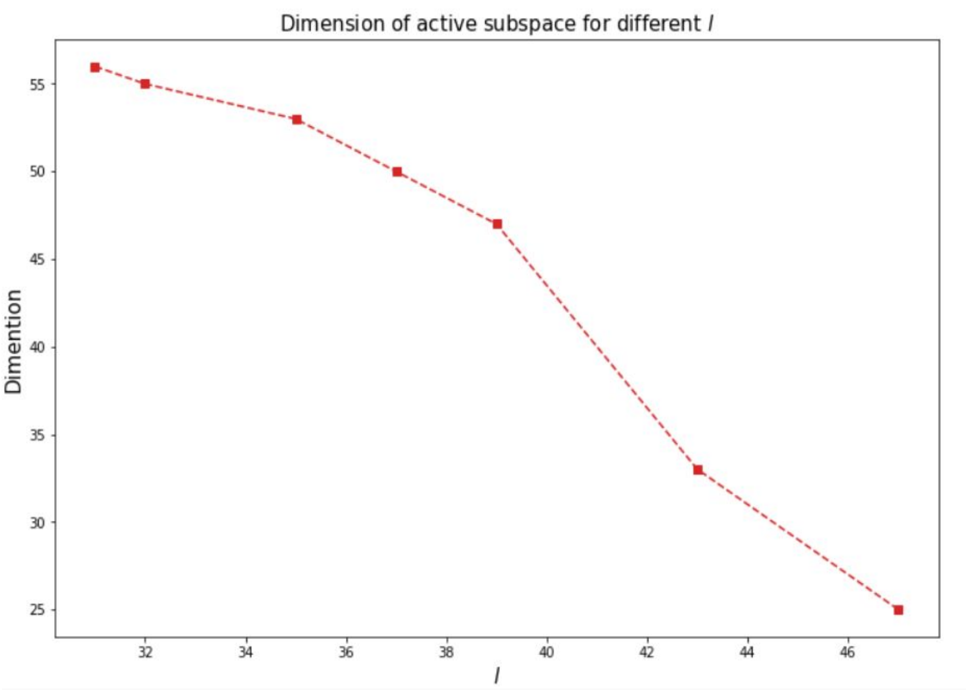
Experimental Results: Resnet110



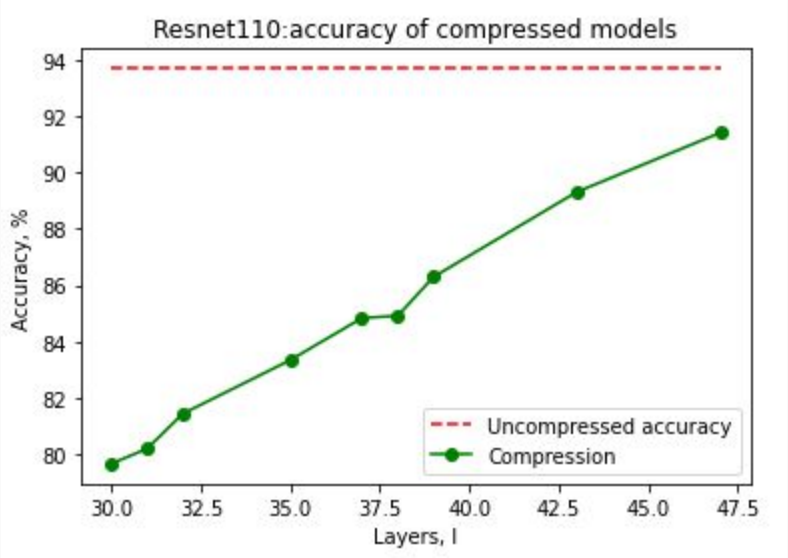


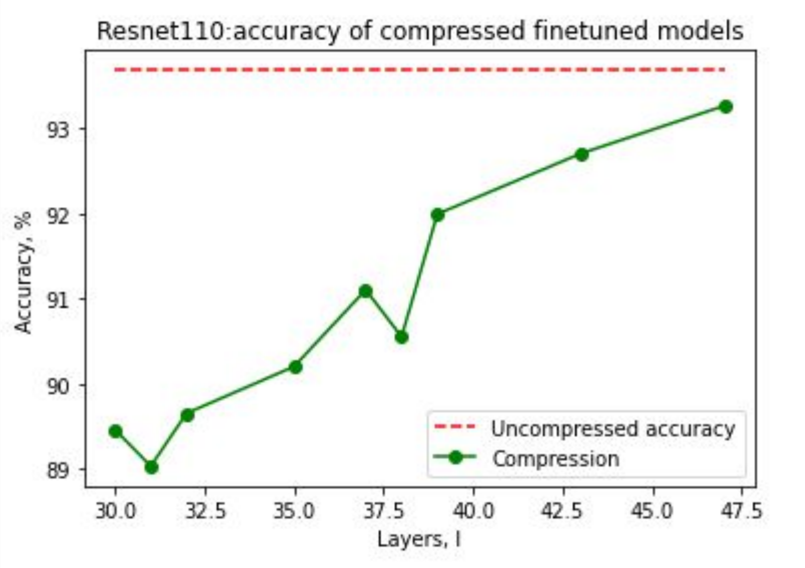
Experimental Results: Resnet110





Experimental Results: Resnet110





Summary

- Compression approach from original paper was reproduced for **VGG19** and **Resnet110** architectures
- Compression worked differently for different network topologies, cut-off layer selection is important
- Minor code errors from the original repository were fixed

References

- Active Subspace of Neural Networks: Structural Analysis and Universal Attacks, Chunfeng Cui, Kaiqi Zhang, Talgat Daulbaev, Julia Gusak, Ivan Oseledets, Zheng Zhang
- ResNet repo: https://github.com/Kaktusava/NLA_ResNet

Questions?