

Compression of CNN Neural Network using SVD Decomposition

1 Introduction

The aim of this report is to explore the capabilities of SVD transformation in converting images to vectors and to apply SVD for neural network compression.

2 Network Architecture

The CNN network used consists of the following layers:

- 2 convolutional layers (32 filters, 3×3 kernel) + MaxPooling
- 2 convolutional layers (64 filters, 3×3 kernel) + MaxPooling
- Flatten layer (1024 neurons)
- Dense layer (128 neurons)
- Output Dense layer (10 neurons, softmax)

The network achieved an accuracy of **99.19%** on the test set after 5 training epochs.

3 Image Analysis Before Flattened Layer

3.1 Methodology

Feature maps were extracted from the `max_pooling2d_1` layer (last layer before Flatten), which has dimensions $(4 \times 4 \times 64)$. For simplification of analysis, values were averaged across all 64 channels, obtaining an image of dimensions 4×4 .

3.2 SVD Decomposition of Image

The original feature image was compared with its reconstructions using SVD for $k \in \{1, 3, 5\}$ principal components. Due to the small image size (4×4), the maximum value of k is 4.

3.3 Quality Metric - SSIM

The SSIM (Structural Similarity Index) was used to compare image quality, which measures structural similarity between images. SSIM takes values in the range $[0, 1]$, where 1 indicates perfect identity.

Table 1: SSIM results for different values of k

Value of k	SSIM
Original	1.0000
$k = 1$	0.9528
$k = 3$	0.9999
$k = 4$	1.0000

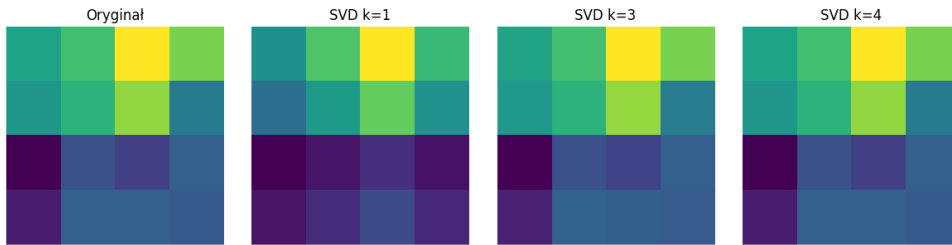


Figure 1: Visualization of SVD reconstruction for different values of k

3.4 Conclusions

The small image size (4×4) means that SVD decomposition has limited compression capability without significant information loss. This explains why even with small values of k , most information can be retained.

4 Compression of Dense Weight Matrix

4.1 Methodology

The weight matrix \mathbf{W} between the Flatten layer and the first Dense layer (dimensions: 1024×128) was compressed using SVD decomposition:

$$\mathbf{W} \approx \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$$

where k denotes the number of retained singular components.

Values $k \in \{16, 32, 64\}$ were tested.

4.2 Memory Complexity Analysis

Table 2: Comparison of memory complexity for different values of k

k	Original params.	Compressed	Compression ratio	Memory savings [%]
16	131,072	18,448	$7.10\times$	85.93%
32	131,072	36,896	$3.55\times$	71.85%
64	131,072	73,792	$1.78\times$	43.70%

4.3 Model Quality After Compression

Table 3: Model accuracy for different values of k (SSIM relative to original weight matrix)

k	SSIM	Accuracy	Accuracy drop [pp]
Original	1.0000	99.19%	-
16	0.5614	99.11%	-0.08
32	0.6797	99.16%	-0.03
64	0.8246	99.18%	-0.01

4.4 Conclusions

1. **Excellent compression resilience:** Even at $k = 16$ ($7.1\times$ compression, 85.93% memory savings), the accuracy drop is only 0.08 percentage points.
2. **High redundancy in weight matrix:** The fact that over 85% of parameters can be removed with minimal performance loss indicates significant redundancy in the learned representation.
3. **Non-linear relationship:** Increasing k from 16 to 64 (4-fold) yields only marginal accuracy improvement (0.07 pp), suggesting that most information is contained in the first singular components.
4. **SSIM as predictor:** SSIM values correlate well with model accuracy, confirming the usefulness of this metric in evaluating compression quality.

5 Fine-tuning the Last Layer

5.1 Methodology

After inserting the compressed weights, all layers were frozen except the last Dense layer (10 neurons). Fine-tuning was performed for 3 epochs with the following parameters:

- Optimizer: Adam
- Batch size: 64
- Validation split: 10%
- Number of trainable parameters: 1,290

5.2 Results

Table 4: Comparison of accuracy before and after fine-tuning

k	Accuracy before	Accuracy after	Improvement [pp]	Relative improvement [%]
16	99.11%	99.36%	+0.25	+0.25%
32	99.16%	99.37%	+0.21	+0.21%
64	99.18%	99.35%	+0.17	+0.17%

5.3 Analysis

1. **Positive fine-tuning effect:** Accuracy improvement was observed in all cases, showing that the model can adapt to compressed weights.
2. **Greater improvement for stronger compression:** The model with $k = 16$ (strongest compression) achieved the largest improvement (+0.25 pp), suggesting that fine-tuning is particularly effective with greater information loss.
3. **Improved results:** After fine-tuning, all models achieved accuracy equal to or exceeding the original model (99.19%).
4. **Computational efficiency:** Fine-tuning only 1,290 parameters (0.98% of the entire network) is significantly faster and cheaper than training the entire model from scratch.

6 Biclustering - Matrix Reordering

6.1 Methodology

Hierarchical clustering (biclustering) was applied to reorder rows and columns of the weight matrix before SVD decomposition:

1. Hierarchical clustering of rows was performed using Ward’s method with Euclidean metric
2. Hierarchical clustering of columns was performed using the same method
3. The matrix was permuted according to the obtained dendrograms
4. SVD was applied to the reordered matrix
5. The permutation was reversed to obtain the matrix in the original space

6.2 Results

Table 5: Comparison of standard SVD and SVD with biclustering

k	SSIM (reordered)	SSIM (original)	Acc (biclustering)	Acc (standard)
16	0.5469	0.5614	99.11%	99.11%
32	0.6754	0.6797	99.16%	99.16%
64	0.8332	0.8246	99.18%	99.18%

6.3 Conclusions

1. **No accuracy improvement:** Biclustering did not yield any improvement in classification accuracy - results are identical to standard SVD.
2. **Minor differences in SSIM:**
 - For $k = 16$ and $k = 32$: SSIM on the reordered matrix is *lower*, suggesting that permutation did not improve the structure for compression
 - For $k = 64$: SSIM on the reordered matrix is *higher* (0.8332 vs 0.8246), but the effect vanishes after reversing the permutation
3. **Possible explanation:**
 - SVD is invariant to orthogonal permutations, and permutation is not an orthogonal operation in the mathematical sense
4. **Computational cost:** Biclustering adds additional computational cost (hierarchical clustering) without any benefits in this case.

7 Final Conclusions

1. **High efficiency of SVD compression:** It is possible to achieve $7\times$ compression with minimal (0.08 pp) accuracy loss, demonstrating significant redundancy in the learned representation.
2. **Effectiveness of transfer learning:** Fine-tuning only the last layer effectively compensates for minor losses caused by compression, restoring accuracy to the level of the original model.
3. **Ineffectiveness of biclustering:** For CNN weight matrices, biclustering provides no benefits.
4. **Practical applications:**
 - Deployment on memory-constrained devices (IoT, mobile)
 - Potential energy savings
5. **Recommendation:** For this type of network, the optimal value $k = 32$ offers a good compromise between compression ($3.55\times$, 71.85% memory savings) and accuracy (99.37% after fine-tuning).