Dimensionality Compression and Expansion in Deep Neural Networks

Stefano Recanatesi, Madhu Advani, Guillaume Lajoie, Amitai Armon, Matthew Farrell, Timothy Moore, Eric Shea-Brown

October 27, 2019

Amit Gabay

1. Introduction

Deep learning data, such as images and text, typically lie on a lower-dimensional manifold in a high-dimensional ambient space. This paper considers in which ways deep networks identify and learn intrinsic lower-dimensional structure of such data. The study shows that deep networks learn representations whose intrinsic dimensionalities are drastically lower than each of their respective numbers of neurons. Working with high-dimensional data comes with significant challenges, often referred to as 'The curse of dimensionality'. As the number of dimensions (features) in a dataset increases, data points tend to be far apart from each other, making distance-based methods less effective. A model that works well in low dimensions might overfit in high dimensions because it tries to fit noise rather than true patterns, which makes it difficult to sample data points that are truly representative of the distribution.
This paper investigates how deep neural networks (DNNs) achieve high classification accuracy on high-dimensional data, generalizing well despite the challenges of the curse of dimensionality and being highly overparameterized. The neural networks operate in two phases:

i. Dimensionality Expansion – Early levels increase intrinsic dimension by generating new representations of attributes.
ii. Dimensionality Compression – Later layers selectively reduce dimensionality by retaining only task-relevant features.

Through theoretical and empirical analysis, the paper explores how stochastic gradient descent (SGD) influences these processes, balancing the need for feature expansion with the necessity of reducing representation complexity.

2. Feature Expansion and the Kernel Trick: Preliminary and Motivation

DNNs process data by progressively transforming and extracting useful representations. A key concept that helps explain this process is feature expansion, where data is mapped into a higher-dimensional space to make patterns more distinguishable. This section introduces the fundamental idea of feature expansion, a concept that is crucial to understanding how deep networks transform input data.

2.1 The Need for Feature Expansion

Many real-world datasets are not linearly separable in their original feature space. That is, there may not be a simple linear function that correctly distinguishes different categories. A common solution is to map the data into a higher-dimensional space where a simpler decision boundary can be found. However, explicitly computing this transformation can be computationally expensive.

2.2 The Kernel Trick: Implicit Feature Expansion

The kernel trick provides a computationally efficient way to perform feature expansion without explicitly computing the transformed features. Given a dataset with input vectors $x \in \mathbb{R}^d$, we can define a mapping: $\phi: \mathbb{R}^d \to \mathbb{R}^m$, where $m \gg d$:

$$\phi(x) = \big(f_1(x), f_2(x), \dots, f_m(x)\big)$$

that transforms each data point into a higher-dimensional feature space.
Now, instead of computing $\phi(x)$ directly, the kernel trick relies on defining a kernel function: $K: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, that computes the inner product in the high-dimensional space without explicitly mapping the points:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle$$

Several commonly used kernel functions include mapping features to polynomial combinations ($K(x, x') = (x \cdot x' + c)^p$) and to an infinite-dimensional space ($K(x, x') = e^{\gamma \|x - x'\|^2}$).

2.3 Feature Expansion in Deep Neural Networks

Deep neural networks achieve feature expansion in a different but conceptually related way.
Unlike the kernel trick, where the feature expansion is fixed and predefined, deep networks learn the optimal feature space through training and achieve a similar expansion via successive nonlinear transformations:

$$h^{(l)} = \phi_l\left(h^{(l-1)}\right)$$

where $h^{(l)}$ is the activation at layer l.

2.4 Link to Dimensionality Expansion and Compression

The paper extends these ideas by the two phases. This dynamic process allows neural networks to effectively learn lower-dimensional representations while still benefiting from the expressive power of high-dimensional feature spaces.

3. Mathematical Analysis of Dimensionality Evolution in Deep Networks

3.1 Intrinsic Dimensionality and Layer-Wise Evolution

A dataset with input vectors $\{x_i\} \subset \mathbb{R}^d$ can be mapped in the high-dimensional space $\mathbb{R}^d$, but effectively lies on a manifold $\mathcal{M}$ of much lower dimension $m \ll d$ (*the intrinsic dimensionality*). Hence the intrinsic dimensionality of a dataset refers to the minimum number of independent variables required to represent its structure effectively. if data points x lie on a manifold $\mathcal{M}$, its intrinsic dimension d is given by:

$$d = \min\{k \mid x \in \mathbb{R}^k \; \forall x \in M\}$$

Given an input dataset with a high ambient dimensionality (e.g., images with 3072 pixels in RGB space), the data is often constrained to a lower-dimensional manifold.

3.2 Methods for Estimating Intrinsic Dimensionality

The paper employs dimensionality estimation techniques that are robust to the high-dimensional nature of neural representations. The two main approaches are:

i. Local Intrinsic estimates the intrinsic dimensionality of data locally by analyzing the distribution of distances to a point's nearest neighbors: for a point x in a dataset with a distance distribution $P_{NN}(r)$, describing the probability of observing a neighbor at distance r, the LID is given by:

$$LID(x) = \lim_{r \to 0} \frac{\log\left(P_{NN}(r)\right)}{\log(r)}$$

This can be approximated using the maximum likelihood estimator (MLE):

$$LID(x) \approx \left(\frac{1}{k} \sum_{i=1}^{k} \log\left(\frac{r_i(x)}{r_k(x)} \right) \right)^{-1}$$

where $r_i(x)$ is the distance from x to its i-th nearest neighbor, k is the number of nearest neighbors used for estimation, $r_k(x)$ is the distance to the farthest of the k nearest neighbors, and:

$$P_{NN}(r) \propto r^d$$

where d is estimated based on the slope $\frac{\log(P_{NN}(r))}{\log(r)}$ in small neighborhoods.
This method helps understand fine-grained geometric structure of the data manifold.

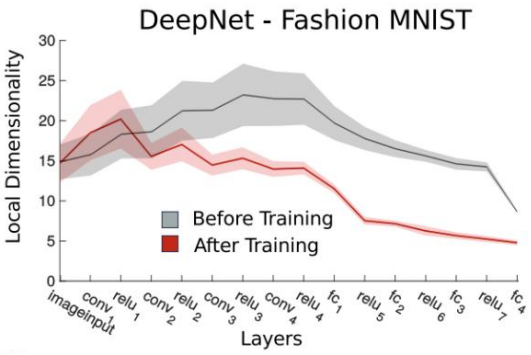

Figure 1: Local Intrinsic Dimensionality Evolution in Deep Networks
LID evolution across layers in DeepNet trained on the Fashion-MNIST dataset. The black curve represents LID before training, while the red curve represents LID after training. Early layers increase dimensionality, while later layers compress the representation.

ii. Global Intrinsic Dimensionality (GID): Uses geodesic distances computed on a k-nearest neighbor graph to estimates the overall manifold structure rather than local variations.
Given a dataset X where points lie on a manifold $\mathcal{M}$, the GID is estimated by comparing the empirical distribution of geodesic distances $P_G(r)$ to the expected distance distribution of a hypersphere of dimension d:

$$GID = argmin_d \sum_{r \in R} \left(P_G(r) - P_d(r)\right)^2$$

where $P_G(r)$ is the empirical distribution of geodesic distances between data points, $P_d(r)$ is the theoretical distance distribution for a d-dimensional hypersphere, and r is a selected range of distances used for comparison.

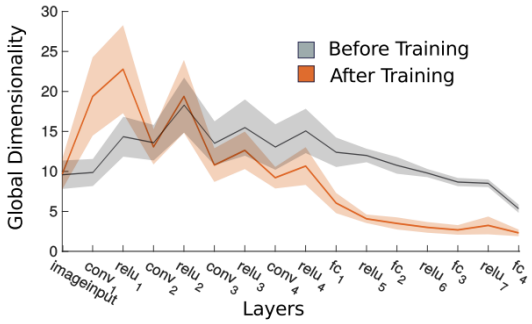

Figure 2: Global Intrinsic Dimensionality Across Layers in DeepNet
GID evolution across layers in DeepNet trained on the Fashion-MNIST dataset. The gray curve represents GID before training, while the orange curve represents GID after training. Early layers increase dimensionality, while later layers compress it.

These approaches reveal that deep networks first expand dimensionality and later compress it, balancing feature extraction and selection.

3.3 Intrinsic Dimensionality of Learned Representations

The paper examines how intrinsic dimensionality evolves across layers in deep networks, revealing a two-phase transformation. Each layer of a deep network can be represented as a function
$f: \mathbb{R}^d \to \mathbb{R}^m$. The change in dimensionality depends on the structure of f:

i. Convolutional layers introduce local feature expansion.
ii. Early convolutional layers significantly increase intrinsic dimensionality. The authors found that ReLU activations consistently increase dimensionality by a measured factor of approximately 1.16× per layer.
iii. Final layers reduce dimensionality, selecting only task-relevant features. Compression is driven by fully connected layers and the optimization process itself.

The balance between expansion and compression is automatically regulated by SGD, which introduces an effective regularization term favoring compact representations.
Experimentation on Convnet trained on Fashion-MNIST and CIFAR-10 shows that first few layers increase dimensions, while last layers reduce it drastically. The last hidden representation typically has only 10-50 effective dimensions despite a total of thousands of neurons, which indicates many neurons are silent or redundant.

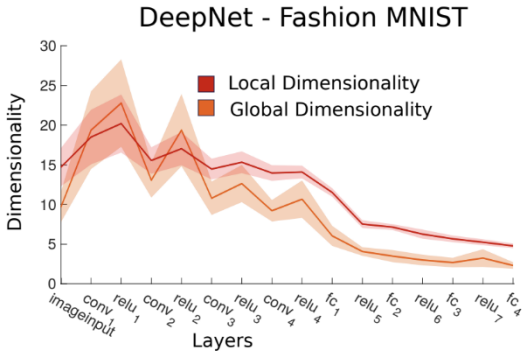

Figure 3: Evolution of Local and Global Intrinsic Dimensionality in DeepNet (Fashion-MNIST)
Comparison of Local Intrinsic Dimensionality (LID) and Global Intrinsic Dimensionality (GID) evolution across layers in DeepNet trained on the Fashion-MNIST dataset. The red curve represents local dimensionality, while the orange curve represents global dimensionality. Both measures initially increase in early convolutional layers due to feature expansion, followed by a gradual compression in deeper fully connected layers.

3.4 Stochastic Gradient Descent (SGD) as a Dimensionality Regularizer

3.4.1 Formulating the Neural Network Model

SGD introduces stochasticity into weight updates, which naturally results in an effective regularization that compresses feature spaces. Consider a two-layer neural network where an input $x \in \mathbb{R}^d$ is transformed into a hidden representation $h \in \mathbb{R}^m$, followed by a linear readout producing the final output $y \in \mathbb{R}^c$:

$$h_i = \sum_j \phi\left(\omega_{i,j}^{(1)} x_j\right), \hat{y}_i = \sum_j \omega_{i,j}^{(2)} h_j$$

where $\omega^{(1)}$ represents the input-to-hidden weights, $\omega^{(2)}$ represents the hidden-to-output weights, ϕ is a nonlinear activation function (e.g., ReLU), h represents the hidden-layer activations, and $\hat{y}$ is the network's predicted output.
The network is trained to minimize a loss function:

$$L(\mathrm{W}) = \sum_{\mu=1}^{P} \|y^\mu - \hat{y}^\mu\|^2$$

where $W = \left[\omega^{(1)}, \omega^{(2)}\right]$, $\{(x^\mu, y^\mu)\}_{\mu=1}^P$ are the training samples, $\hat{y}^\mu = f(W, x^\mu)$, and f represents the function (a deep neural network) that takes x^μ as input and produces $\hat{y}^\mu$ as output.

3.4.2 Effect of SGD Noise on Weight Updates

SGD updates the parameters using a noisy estimate of the gradient:

$$\Delta W = -\eta \nabla_w L(W) + \sigma\xi$$

where η is the learning rate, σ is the noise amplitude, which depends on batch size, and ξ represents stochastic noise arising from mini-batch updates. This stochastic noise can be modeled as Gaussian with zero mean: $\xi = \mathcal{N}(0, I)$.
Because of the noisy updates, SGD does not converge exactly to the minimum of $L(W)$, but rather fluctuates around it, leading to an implicit regularization effect.

3.4.3 Effective Loss Function Induced by SGD

SGD automatically suppresses task-irrelevant dimensions by adding a regularization penalty proportional to the variance of hidden activations. Due to the stochastic noise in weight updates, an effective loss function emerges:

$$L_{eff}(W, \sigma) = L(W) + R(W, \sigma)$$

where the regularization term $R(W, \sigma)$ takes the form:

$$R(W, \sigma) = \sigma^2 Tr(C)$$

where C is the covariance matrix of the hidden-layer activations:

$$C_{jj'} = \sum_\mu h_j^\mu h_{j'}^\mu$$

This means that SGD penalizes large hidden representations, leading to dimensionality compression.

3.4.4 Compression of Task-Irrelevant Directions

The goal of learning is to minimize the effective loss L_{eff}.
To achieve this, the network adapts its representation by expanding the dimensionality in earlier layers to generate features that help separate classes, and compressing the dimensionality in later layers to remove redundant task-irrelevant dimensions. Thus the optimal hidden representation h^* satisfies:

$$h^* = argmin_h \left\{ L(W) + \sigma^2 \sum_j \|h_j\|^2 \right\}$$

which shows that SGD inherently favors lower-dimensional representations.

3.4.5 Role of Activation Functions in Dimensionality Compressio

- Linear case – $\phi(x) = x$:
  the hidden layer is given by:

  $$h = \omega^{(1)} x$$

  and the output is:

  $$\hat{y} = \omega^{(2)} h$$

  the Moore-Penrose pseudo-inverse provides the optimal solution:

  $$\omega^{(1)} = \left(\left(\omega^{(2)}\right)^T \omega^{(2)} + \sigma^2 I\right)^\dagger \left(W^{(2)}\right)^T Y^T X (X^T X)^\dagger$$

  where X is a $P \times d$ matrix of input samples ,Y is the matrix of target outputs for the training data (if we have P training samples and c output classes, then $Y \in \mathbb{R}^{P \times c}$), and each row y^μ corresponds to the true label of a training sample.

This reveals that the range of $\omega^{(1)}$ is restricted to the span of $\omega^{(2)}$, and task-irrelevant directions are removed, reducing the intrinsic dimensionality.

- ReLU case – $\phi(x) = \max(x, 0)$:
  For ReLU networks, the constraint $h \geq 0$ introduces additional dimensionality constraints:
  $$h = \max\left(0, W^{(1)}x\right)$$
  This causes higher-dimensional activations in early layers, since ReLU can introduce sparsity, and more compression in deeper layers, as SGD further reduces redundancy.

In both linear and ReLU networks, SGD automatically balances expansion and compression, ensuring the learned representation is as low-dimensional as possible while preserving task-relevant information.

3.5 Experimental Evidence of SGD-Induced Compression

Simulation Setup: a 7-layer fully connected network was trained on Fashion-MNIST with ReLU activations.
Two scenarios were compared: without SGD noise ($\sigma = 0$) and with it ($\sigma > 0$).
Key Observations:

i. When noise is injected into weight updates, the dimensionality of representations is lower across layers.
ii. Smaller mini-batch sizes (which increase SGD noise) lead to stronger compression.
iii. Doubling layer width does not increase the dimensionality of learned representations—suggesting that SGD finds a minimal intrinsic dimensionality regardless of network size.

Experimental Conclusion: SGD-driven noise acts as a dimensionality regularizer, enforcing compact representations that improve generalization.

4. Implications for Generalization and Network Design and Open Questions

Why Does Compression Improve Generalization?

i. Overfitting Prevention: Lower-dimensional representations reduce the model's ability to memorize noise.
ii. Feature Selection: The network automatically removes unimportant dimensions.
iii. Generalization Bounds: Theoretical work suggests that low-dimensional manifolds require fewer samples to generalize well.
iv. The paper suggests new training strategies: Instead of just minimizing classification error, one could explicitly optimize network dimensionality.

5. Final Takeaways

This paper provides a mathematical framework explaining why and how deep networks expand and compress dimensionality during learning. By leveraging intrinsic dimensionality estimation and SGD theory, the authors show that dimensionality balancing is a fundamental property of deep learning:

i. SGD acts as an implicit regularizer, reducing task-irrelevant dimensions.
ii. Dimensionality expansion in early layers improves feature separability.
iii. Dimensionality compression in later layers enhances generalization.
iv. The optimal network representation balances these two forces.

These insights could lead to better network architectures, improved training strategies, and a deeper understanding of generalization

6. Practical Applications and Potential Future Directions

A clearer understanding of mechanisms of dimensional compression and expansion can lead to enhanced deep learning architectures, which will help develop a more powerful neural networks with enhanced capacity for generalization, with enhanced performances in real-world applications ranging from scientific simulations, natural language processing, and computer vision:

i. Building more powerful architectures, improving optimization algorithms, and new regularization methods.
ii. The outcomes of this study have additional applicability in other areas, such as in neuroscience and data-driven scientific modeling.
iii. Additional study of dimensions in neural networks can lead to a greater depth understanding of theories and more interpretable and resilient machine learning.
iv. The results suggest that optimal architectures should encourage early expansion followed by compression. Adding explicit noise (dropout, weight noise) can further enhance compression.
v. Since SGD naturally enforces dimensionality compression, future work could explore explicitly controlling dimensionality during training and finding methods to control and optimize this behavior.

vi. Neuroscientists could investigate whether brain regions responsible for different cognitive tasks exhibit similar expansion-compression dynamics. This could provide insights into how biological learning optimally balances efficiency and expressivity, just like deep networks

7. Opinion on the Paper and Its Significance

In my opinion, this paper presents a fundamentally important contribution to understanding how and why deep neural networks generalize well despite their overparameterization. The authors provide both empirical and theoretical evidence that deep networks naturally expand and then compress intrinsic dimensionality, a process that balances expressiveness and efficiency.

I think one of the most compelling aspects of the paper is its rigorous mathematical analysis of how SGD acts as an implicit dimensionality regularizer. This perspective helps bridge gaps between:

i. Optimization theory: how SGD minimizes loss while enforcing structure in representations.
ii. Representation learning: how deep networks organize information across layers.
iii. Generalization theory: why overparameterized models don't necessarily overfit.

The main result, stating deep learning models learn low-dimensional manifolds from high-dimensional data, which improves generalization by eliminating redundant dimensions, aligns well with the Information Bottleneck Hypothesis and the broader theme of efficient representation learning in both artificial and biological systems.

From my perspective, the paper offers a deep theoretical insight into why deep learning works so well, going beyond traditional explanations. The mathematical connection between SGD, intrinsic dimensionality, and generalization is especially compelling.