# Subspace Clustering Based Analysis of Neural Networks

Uday Singh Saini ✉[1], Pravallika Devineni[2], and Evangelos E. Papalexakis[1]

[1] University of California Riverside, 900 University Avenue, Riverside, CA, USA
usain001@ucr.edu, epapalex@cs.ucr.edu
[2] Oak Ridge National Laboratory, Oak Ridge, TN, USA
devinenip@ornl.gov

arXiv:2107.01296v1 [cs.LG] 2 Jul 2021

**Abstract.** Tools to analyze the latent space of deep neural networks provide a step towards better understanding them. In this work, we motivate sparse subspace clustering (SSC) with an aim to learn affinity graphs from the latent structure of a given neural network layer trained over a set of inputs. We then use tools from Community Detection to quantify structures present in the input. These experiments reveal that as we go deeper in a network, inputs tend to have an increasing affinity to other inputs of the same class. Subsequently, we utilise matrix similarity measures to perform layer-wise comparisons between affinity graphs. In doing so we first demonstrate that when comparing a given layer currently under training to its final state, the shallower the layer of the network, the quicker it is to converge than the deeper layers. When performing a pairwise analysis of the entire network architecture, we observe that, as the network increases in size, it reorganises from a state where each layer is moderately similar to its neighbours, to a state where layers within a block have high similarity than to layers in other blocks. Finally, we analyze the learned affinity graphs of the final convolutional layer of the network and demonstrate how an input's local neighbourhood affects its classification by the network.

## 1 Introduction

With the emergence of deep neural networks in a variety of domains, there is a need for understanding and characterising the inner workings and operations of these models. With critical applications such as autonomous vehicles, healthcare, and criminal justice relying on neural network-based models, model interpretability has become an important and necessary aspect of machine learning. In the domain of theoretical analysis of neural networks, previous works like [13], [7], and [14] focus on the approximation capability of neural networks, while works like [12] and [19] focus on the interpretations of neural networks as kernels. In the domain of experimental analysis, works like [25], [21] and [2] focus on visualising inputs, filters and neurons of a network.

Our work focuses on interpreting the behaviour of neural networks by analyzing subspaces of latent representations learned by the layers of a neural network.

Most closely related to our approach are works like SVCCA [20], PWCCA [15] and Linear-CKA [10], [18]. SVCCA and PWCCA operate directly on the activations of a neural network for a given set of inputs and allows comparison between two such sources of activations over the same inputs, thereby facilitating a comparison between different layers of the same network, or even different networks. On the other hand, [10] and [18] take the same activations over a set of inputs and use it to construct a pairwise similarity matrix with the help of a kernel. Taking these pairwise similarity matrices obtained via the response of two different layers to a set of inputs, they then compare the two kernel matrices using Centered Kernel Alignment (CKA) [4]. This lets them directly compare any two layers from different sources. For most of their experiments, [10] and [18] motivate the theory and use of linear kernels, henceforth we call their method Linear-CKA. All these methods offer architecture agnostic ways to analyze neural networks.

We investigate Sparse Subspace Clustering (SSC) [5] as an alternative to Kernel-based pairwise Similarity matrices over inputs. Methods like SSC help us learn a connectivity graph over data points that are assumed to belong to an underlying union of subspaces embedded in a given space. Such methods are typically based on the **Self-Expressiveness** property in a union of subspaces, where each data point can be represented as a sparse linear combination of points from its own subspace. Learning such a graph helps impart transparency to the learning dynamics of the neural network and also presents us with a way to represent each layer of a neural network in an architecture agnostic manner. This facilitates the comparison of layers within a network and across various architectures.

We combine SSC with CKA to provide an alternate similarity measurement tool for latent representations learned by neural networks. It is akin to translating a layer's activations over a set of inputs into a connectivity graph and then using CKA to compute the similarity between graphs arising from two different neural network representations.

Our main contributions are the following:

1. We utilise SSC to learn a connectivity graph over inputs in the latent space. We then employ graph modularity [16] to characterise the community structure of each input's neighbourhood. We demonstrate that as we go deeper into the layers of the network, the community around each input becomes more homogeneous, i.e., a higher proportion of an input's neighbourhood in the Affinity Graph is of the same class as the input itself. Additionally, we also demonstrate the utility of SSC-CKA to capture network training dynamics. We observe that when compared to shallower layers, the deeper layers of the network tend to take longer to converge to their final state. This is a corroboration of similar observations made in SVCCA [20].
2. We demonstrate the ability of SSC-CKA to visualize and analyze entire network architectures by capitalising on its ability to perform pairwise comparison of two different layers of a network (or two different networks). These

experiments set along the lines of Linear-CKA [10], demonstrate the effects of depth, width, epochs and training data size on the network architecture.
3. We demonstrate the ability of SSC to interpret the latent spaces of a given layer of the network. SSC allows us to represent each input as a weighted and sparse linear combination of other inputs in the same subspace. Here we demonstrate a strong correlation between classes of network's prediction for the input and the class having the highest weightage in the local neighbourhood of the input. This hints towards neural networks separating out classes into disjoint subspaces and presents interesting opportunities and directions going forward for creating tools to analyze neural networks.

## 2 Background and Method

In this section, we lay the background on the Sparse Subspace Clustering, its optimization through ADMM [3] and motivate its fitness for the purpose of interpreting subspaces learned by neural networks.

### 2.1 Sparse Spectral Clustering

Subspace clustering refers to the task of clustering the data into their original subspaces and uncovering the underlying structure of the data. Given some high-dimensional data, subspace clustering attempts to model the data as samples drawn from a union of multiple low-dimensional linear subspaces [5]. First, we present Sparse Spectral Clustering (SSC) for the case of uncorrupted data. As mentioned earlier, SSC typically works well when the underlying data follows Self-Expressive Property, i.e., each data point can be represented by a linear combination of points that belong to the same subspace.

Let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, such that $X \in \mathbf{R}^{d \times N}$ represents the data matrix. Let $C = [\mathbf{c}_1, \ldots, \mathbf{c}_N]$, such that $C \in \mathbf{R}^{N \times N}$ and $\mathbf{c}_i \in \mathbf{R}^N$, where the entry $(i, j)$ of the matrix $C$, given by $c_{ij}$ represents the weight of data point $\mathbf{x}_j \in \mathbf{R}^d$ in the linear combination to reconstruct $\mathbf{x}_i \in \mathbf{R}^d$. Mathematically, a noiseless model for SSC is equivalent to Equation 1. However, as shown in [1], this problem is NP-hard.

$$\min_{\mathbf{c}_i} ||\mathbf{c}_i||_0 \quad \text{s.t.} \quad \mathbf{x}_i = X\mathbf{c}_i,\, c_{ii} = 0 \quad \forall i \in \{1, \ldots, N\} \tag{1}$$

We therefore focus on Equation 2, a convex relaxation of Equation 1 which is also robust to noise and solve this optimization problem using Algorithm 1 as shown in [23].

$$\min_{C,Z} ||C||_1 + \frac{\tau}{2}||X - XZ||_F^2 \quad \text{s.t.} \quad Z = C - diag(C) \tag{2}$$

### 2.2 Centered Kernel Alignment

Centered Kernel Alignment [4], as defined in Equation 3, between two similarity matrices $X$ and $Y$ is an isotropic invariant similarity index that relies upon

Hilbert-Schmidt Independence Criterion (HSIC) [6] to determine statistical dependence between two sets of variables.

$$CKA(X,Y) = \frac{HSIC(X,Y)}{\sqrt{HSIC(X,X)HSIC(Y,Y)}} \tag{3}$$

where HSIC between a pair of $N \times N$ matrices is defined in Equation 4

$$HSIC(X,Y) = \frac{trace(HXHHYH)}{(N-1)^2} \tag{4}$$

where H is a Centering matrix given by $H = I - \frac{1}{N}\mathbf{1}\mathbf{1}^T$.

### 2.3 Meta Algorithm

Our goal is to take a matrix of neural activations $X \in \mathbf{R}^{d_1 \times N}$, where $d_1$ is the number of neurons in the given layer and $N$ is the number of examples for which we obtain the activations, and construct an affinity matrix $C_X \in \mathbf{R}^{N \times N}$ , via Sparse Subspace Clustering [5] , where each non diagonal entry $(i, j)$ of $C_X$ denotes the affinity between input samples $i$ and $j$. This gives us the ability to compare 2 matrices of neural activations, say $X \in \mathbf{R}^{d_1 \times N}$ and $Y \in \mathbf{R}^{d_2 \times N}$ by representing them as 2 Affinity Matrices $\in \mathbf{R}^{N \times N}$, namely $C_X$ and $C_Y$ and then using Centered Kernel Alignment[4] to compare the similarity of the 2 Affinity Matrices. This procedure helps us compare 2 different layers of a network, or 2 different layers of 2 architecturally different neural networks. We would like to point out that $C_X = |C| + |C^T|$ where the matrix $C$ is obtained from Algorithm 1, where $|C|$ represents the absolute value function applied element-wise to $C$.

**Algorithm 1:** Matrix LASSO Minimization by ADMM

**Data:** Data Matrix $X$
**Result:** Sparse Representation $C$
**initialization:** $C^0 = \mathbf{0}$, $\Lambda_2^0 = \mathbf{0}$, $\mu_2 > 0$;
**while** *not converged* **do**
  $Z^{k+1} = (\tau X^T X + \mu_2 I)^{-1}(\tau X^T X + \mu_2(C^k - \frac{\Lambda_2^k}{\mu_2}))$;
  $C^{k+1} = \mathbf{S}_{\frac{1}{\mu_2}}(Z^{k+1} + \frac{\Lambda_2^k}{\mu_2})$; $\mathbf{S}$ : Shrinkage operator
  $C^{k+1} = C^{K+1} - diag(C^{k+1})$;
  $\Lambda_2^{k+1} = \Lambda_2^k + \mu_2(Z^{K+1} - C^{k+1})$;
**end**

## 3 Problem and Experimental Setup

In this paper, we experiment with VGGs [22], ResNets [8], Wide-ResNets [24] and DenseNets [9] trained on CIFAR-10 and CIFAR-100 datasets [11]. Our approach

relies upon having access to activations of various hidden layers in the network for each input instance. As an example, for the first set of experiments in this study, we take outputs from a few pooling layers at the end of each block of a DenseNet to study the community structure of its subspace and also to learn the layer-wise dynamics of training progression. In another setting, we take focus on all the convolutional layers in the network and get their activations for an input.

### 3.1 Problem Formulation

We attempt to interpret and analyze neural networks by taking a matrix of its activations (layer-wise) $X \in \mathbf{R}^{d \times N}$, where $d$ is the number of neurons in the subject layer and $N$ is the number of inputs used for analysis. We first aim to learn a connectivity graph between these inputs based on their affinity scores obtained from SSC. Then we use this connectivity matrix to perform analysis pertaining to the community structure of the graph in Section 4, and in Section 6 we analyze instance based neighbourhoods of various inputs to better reconcile network predictions with the need to human oriented explanations

### 3.2 Experimental Details

All networks, unless otherwise stated, were trained with a learning rate of 0.1 and a weight decay of $5 \times 10^{-4}$ with a learning rate step multiplier of 0.2 applied after every 30 epochs. All networks were trained for 100 epochs since that was sufficient to achieve optimal performance, with the exception of VGG-29, which required 160 epochs. For SSC computations, both $\tau$ and $\mu$ were set to 10, with $\mu$ being adaptive based on equation 3.13 in [3]. We recommend choosing $\tau$ and $\mu$ between 10 and 100. Our implementation is publicly available on GitHub [1].

## 4 Analysis of Network Training Dynamics

In this section, we analyze behaviour of the network as its training progresses. To demonstrate the training dynamics, we train the following networks[2] - ResNet [8], Wide ResNet [24] and DenseNet [9] on the datasets CIFAR-10 and CIFAR-100 [11]. Both ResNets and DenseNets are constructed by stacking and joining different residual blocks with multiple convolution layers residing in each block. For brevity and scalability of this experiment, we use the output of residual blocks instead of every convolution layer and the final classification layers. For a given combination of the network and the dataset, we present three results - layer-wise modularity of the sparse subspace affinity graphs, SSC-CKA based layer-wise similarity to understand training dynamics, and analogous training dynamic analysis with Linear-CKA.

[1] URL - `https://github.com/23Uday/Subspace-Clustering-based-analysis-of-Neural-Networks`

[2] Networks used from: `https://github.com/kuangliu/pytorch-cifar` and `https://github.com/meliketoy/wide-resnet.pytorch`

### 4.1 Community Structure via Graph Modularity

We analyze the community structure of the sparse subspace affinity graph for various layers at each epoch. We do this by calculating the modularity [16] of the learned subspace affinity graph. Modularity is a measure of the structure of a graph, measuring the density of connections within a module or community. High modularity in a graph indicates dense connectivity between nodes of the same type, while low modularity indicates dense connectivity between nodes of different types, where a node type is its class label. Mathematically, modularity of a graph can be represented as follows

$$Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{5}$$

where $m$ is the number of edges in the graph, $A_{ij}$ is the weight of the edge between nodes $i$ and $j$, $k_i$ is the degree of node $i$, $c_i$ is the class label of node $i$ and $\delta(c_i, c_j)$ is 1 if node $i$ and node $j$ are of the same class and 0 otherwise.

We present this analysis in Figure 1a and Figure 2a. We observe that as we go deeper in the network, the modularity of the learned sparse subspace affinity graph increases, implying that earlier layers of the network cluster examples of different classes together in the same subspace and deeper layers of the network tend to separate out classes into different disjoint subspaces. Another noteworthy observation is that the modularity scores of subspace affinity graph learned from earlier layers tends to saturate earlier in training when compared to the modularity of the subspace affinity graphs of deeper layers. This phenomenon is consistent across different architectures and different datasets. A similar observation regarding earlier saturation of shallower layers in training dynamics is also made in [20], albeit in context of representational similarity, which we address next.

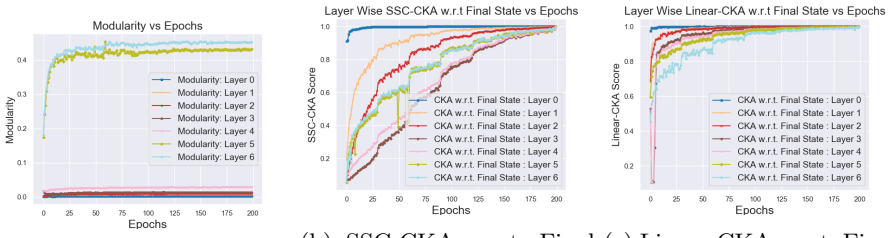
### 4.2 Layer-wise Training Dynamics and Convergence

For our second analysis, we compute the Centered Kernel Alignment (CKA) scores between the sparse subspace affinity graph of a layer at a given epoch and the same layer after the final epoch. This allows us to observe the rates at which layers converge to their final states. These results are shown in Figure 1b and Figure 2b. We observe a similar pattern of shallower layers converging to their final representation much earlier, when compared to deeper layers of the network. Another key observation that we make pertains to the behaviour of representations when the learning rate of the optimiser is reduced to improve convergence. For training of all the networks, we start with a learning rate of 0.1 and reduce it by a factor of 0.2 after every 30 epochs using SGD with learning rate decay. At each step size decay, in addition to an improvement in network accuracy, we also observe a jump in the CKA similarity scores of the epoch's subspace affinity graph towards the CKA scores of final epoch's subspace affinity graph. We note that this jump is less prominent in shallower layers of the network

when compared to the deeper layers as shallower layers converge to their final representations much earlier in training when compared to deeper layers. This bottom-up convergence observation is similar to observations made in [15], and [20] as described earlier. This jump also signifies a marked deviation from the representations of the layer during it's previous state - before the step decay, and current state - after the step decay. These observations highlight the role of step decay in Stochastic Gradient Descent based training of Neural networks to escape saddle points and other spurious local minima.

### 4.3 Comparison with Linear-CKA:

As a comparison with related works in the area in Figure 1c and Figure 2c we present the same training dynamics as previous paragraph, this time by analyzing the Linear Kernel-CKA [10]. We observe a much cleaner bottom-up convergence of layers when compared with our method SSC-CKA, but the rates of convergence to final state is much faster when compared with the rates from SSC-CKA.

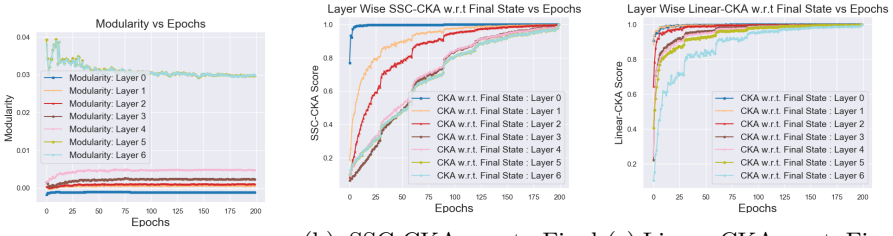



(a) Modularity vs Epochs (b) SSC-CKA w.r.t. Final State (c) Linear-CKA w.r.t. Final State

Fig. 1: Analysis of SSC Affinity Graphs using Modularity and CKA and it's comparison with a Linear Kernel Affinity Graph on DenseNet-121 network using CIFAR-10 dataset.

## 5 Analysis of Network Architecture

In this section, we utilise SSC-CKA to visualize network architectures. We do this by using CKA to compare the Subspace Affinity Graphs obtained by applying SSC on the activations of two different layers of the network. We try to focus on various inter-layer dynamics in different scenarios so as to learn intrinsic behaviours of the network and the architecture class it belongs to. In pursuit of these goals, we evaluate and demonstrate the architectural behaviour of various networks as a function of network architectural depth, network width, training duration in terms of epochs and finally as a function of quantity of training data.




(a) Modularity vs Epochs (b) SSC-CKA w.r.t. Final State (c) Linear-CKA w.r.t. Final State

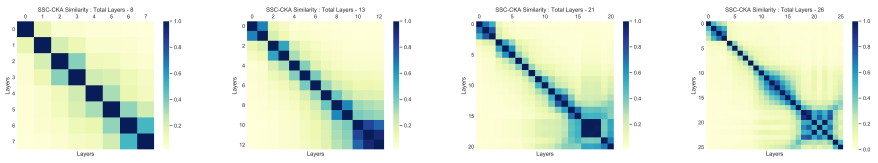
Fig. 2: Analysis of SSC Affinity Graphs using Modularity and CKA and it's comparison with a Linear Kernel Affinity Graph on DenseNet-121 network using CIFAR-100 dataset.

The motivation behind these experiments is to decipher how the latent space structure evolves through different layers of the network and how does a given set of layers add to the modeling power of the network. In doing so, we observe that deeper networks, wider networks, prematurely trained networks (networks trained for fewer epochs than optimal) and malnourished networks (networks trained on less training data but trained till saturation of performance) tend to develop prominent and mostly non-overlapping block diagonal structures in their layer-wise SSC-CKA heatmaps. This indicates that as the network's modeling capacity increases, a given amount training data is unable to exhaust the spare modeling bandwidth available due to additional layers, thus, a network tends to reorganise itself into blocks of layers with a high intra-block similarity among layers and a low inter-block similarity between layers. The networks used in these set of experiments are VGGs [22], ResNets [8], Wide-ResNets [24], all experiments conducted with CIFAR-10 [11] as the dataset.

### 5.1 Observing the Effects of Depth

In this section we demonstrate the effect of depth on neural network architecture by analyzing the SSC-CKA maps of different CNNs with varying depths. Through Figure 3a - Figure 3d, we demonstrate the effects of increasing depths in VGG architecture based networks. In Figure 3a, we train a VGG-11 and reach around 92% accuracy on the evaluation set. Applying SSC-CKA to this network, we observe a prominently diagonal similarity matrix indicating that most layers, especially the earlier ones learn unique representations and as we go deeper in the architecture, some inter layer similarity appears. Upon increasing the depth in the architecture to a VGG-16 as shown in Figure 3b, we observe a slight improvement in accuracy 94%, but we begin to observe a diagonally dominant similarity matrix where shallower layers have some degree of similarity with their neighbours, but deeper layers, especially towards the end tend to form blocks of layers which are very similar to each other indicating that the network doesn't

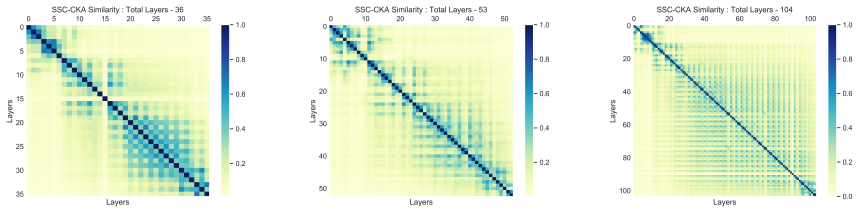
need to utilize the additional bandwidth to learn newer features in order to better learn and generalise. In Figure 3c, we show a similar observation for VGG-24 that attained an accuracy of 93%. However, as we increase the number of convolutional layers to 29 (VGG-29) Figure 3d, the performance of the network drops to around 62%, which is symptomatic of overfitting. We also observe a reinforcement in the prominence of the block diagonal structure signifying network over-parametrization relative to the data.



(a) VGG-11 (b) VGG-16 (c) VGG-24 (d) VGG-29

Fig. 3: Left to Right : Pairwise SSC-CKA between all convolution layers of VGG-11: (accuracy 92%), VGG-16: (accuracy 94%), VGG-24: (accuracy 93%) and VGG-29: (accuracy 62%) respectively on CIFAR-10 dataset
.

Next we demonstrate a similar set of results for ResNets of various depths, Figure 4a - Figure 4c. In case of ResNets, in addition to the block diagonal structure we also notice a chess-board pattern inside the blocks themselves, where every layer is similar to every alternate layer in a block. This is a consequence of skip-connections present inside resnet blocks, which allow inputs from one layer to propagate much deeper into the network. The observations in this experiment are in-line with the ones made in [10] and [18].

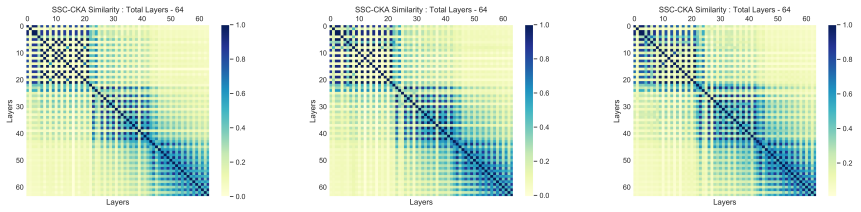


(a) ResNet-36 (b) ResNet-53 (c) ResNet-104

Fig. 4: Left to Right : Pairwise SSC-CKA between all convolution layers of ResNet-36: (accuracy 94%), ResNet-53: (accuracy 94%), ResNet-104: (accuracy 93%) respectively on CIFAR 10 dataset.

### 5.2 Observing the Effects of Width

In this experiment, we aim to study the effects of increasing the width of an architecture for a given depth. We use Wide-ResNet-64 network architecture with varying depth configurations (width 2x, 6x and 10x) and train them to their peak performance. The results of SSC-CKA are presented in Figure 5a - Figure 5c. In case of Wide-ResNets for a given depth, we observe that the architectural structure of various width configurations is very similar. The network can be divided into three distinct and disjoint blocks of layers, each having a high intra-block similarity and low inter-block similarity. Furthermore, we also notice the network wide presence of the chess-board pattern inherent to ResNets where even and odd layers in a layer block are dissimilar to each other, especially in the shallower layers of the network. These results present a contrast to those in [18], as the authors there observe a marked increase in presence of a block structure as the width of a network increases.



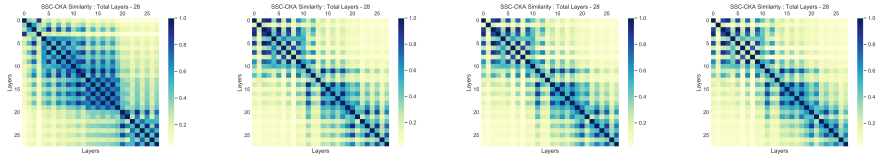

(a) WR-64x2 (b) WR-64x6 (c) WR-64x10

Fig. 5: Left to Right : Pairwise SSC-CKA between all convolution layers of Wide ResNet-64 with width 2x(Accuracy-95%), 6x(Accuracy-95%) and 10x(Accuracy-96%), respectively on CIFAR-10 dataset.

### 5.3 Observing the Effects of Epochs

Here, we try to observe the behaviour of a network during the process of its training to better understand network training dynamics and visualize through SSC-CKA the process with which layers of a neural network organise themselves. We choose a Wide ResNet-28-by-2 and observe the SSC-CKA heatmaps after $1^{st}$ Epoch (35% - Figure 6a), $31^{st}$ Epoch (84% - Figure 6b), $61^{st}$ Epoch (91% - Figure 6c) and $100^{th}$ Epoch (95% - Figure 6d) of training where the network accuracy at that stage is indicated in parenthesis. After the first epoch when the network has an accuracy of around 35% on the evaluation set, we observe an extremely large block diagonal structure encapsulating about the first two-third of the network. Subsequently as the network trains and improves its performance, the large global structure present in the earlier layers makes way for a smaller and more contained local block diagonal structures. This observation into network training

dynamics can provide us with alternate ways to determine the maturation of the training process, one that doesn't necessarily need any labeled data. This experiment can be seen as an extension of the procedure demonstrated in Figure 1b and Figure 2b, but instead of comparing a few layers of the network per epoch to their final state, we compare all convolutional layers after every few epochs.

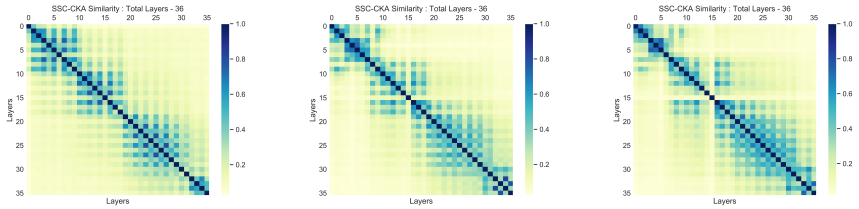


(a) WR-28x2 After $1^{st}$ Epoch. (b) WR-28x2 After $31^{st}$ Epoch. (c) WR-28x2 After $61^{st}$ Epoch. (d) WR-28x2 After $100^{th}$ Epoch.

Fig. 6: Left to Right : Pairwise SSC-CKA between all convolution layers of Various Wide ResNet-28x2 After $1^{st}$, $31^{st}$, $61^{st}$ and $100^{th}$ Epoch in Training on CIFAR-10 dataset.

### 5.4 Observing the effects of Quantity of Training Data

Next, along the lines of [18] we setup an experiment to demonstrate the emergence of the block diagonal structure in SSC-CKA heatmaps in networks that are over-parameterized relative to the training data. To simulate network size relative to data we train the network on 3 different dataset configurations (of CIFAR-10), the first configuration uses only 5% of the original CIFAR-10 training set, the second uses 50% of the training set and the third uses the entire set. The network used in these sets of experiments is a ResNet-36 trained till saturation of performance on the evaluation set. The network achieves an accuracy of 56% when trained with only 5% of the training data - Figure 7a, 92% when trained with half the training set - Figure 7b and 94% when trained with the entire training data - Figure 7c. Comparing the network that was trained on only 5% of the dataset to the other two configurations we observe a more pronounced block-diagonal structure consisting of two distinct and disjointed blocks comprising the first half of the network in the former configuration. As the network is less starved for data, the block diagonal structure that was prevalent in the first half starts to become less recognisable until it breaks down into much more localised structures as seen in Figure 7c. These observations through SSC-CKA re-affirm the assertions made in Linear-CKA [10], [18] that over-parametrized networks are prone to developing a block diagonal structure as shown in their heat-maps for pairwise similarity. Please note that the second half of all the networks still retains a block diagonal structure. This is in accordance with the earlier observations made by increasing the depths of neural networks.




(a) ResNet-36 - 5% Training Data (b) ResNet-36 - 50% Training Data (c) ResNet-36 - 100% Training Data

Fig. 7: Left to Right : Pairwise SSC-CKA between all convolution layers of Various ResNet-36 when trained on 5%, 50% and 100% Training Data on CIFAR-10 dataset.

## 6 Analysis of Inputs

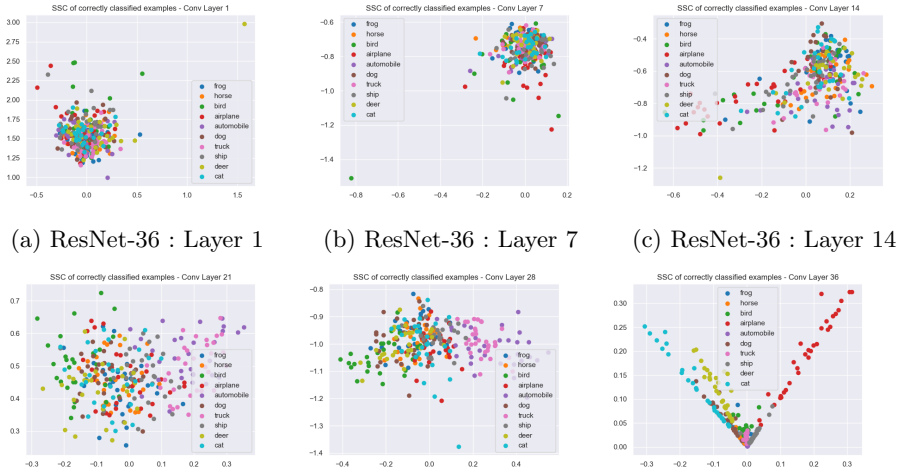
In this section, we focus our analysis on individual inputs fed to the network as a part of the study. The goal is to highlight the ability of the model to interpret each input in terms of it's affinity to it's neighbours, and demonstrate experimentally that the neural network tries to separate out input of different classes into different mostly disjoint subspaces.

### 6.1 Layer-wise Latent Space Visualisation

To begin the analysis, we first present a selective layer-by-layer two-dimensional embedding of all correctly classified inputs to the network. We obtain the embeddings by taking the top two eigenvectors obtained from the decomposition of the respective layer's affinity matrix (Normalised Laplacian Matrix can also be used), which we do along the lines of [17] and [5]. In Figure 8a - Figure 8f, we present the layer-wise analysis for six layers within the network, namely layer 1, 7, 14, 21, 28 and 36 respectively. In accordance with the observation made in Figure 4 regarding the layer-by-layer modularity scores of the subspace affinity graph as we go deeper in the network, we observe a similar and perhaps a corroborating phenomenon, where inputs belonging to the same or similar classes get clustered closer to each other in a section of the latent space, and the inputs that belong to different classes get pushed into disjoint sections of the latent space, as we go deeper in the network.

### 6.2 Model Explanation by Instance Neighbourhood Visualisation

Here, we analyze the failure cases and focus on the inputs where the network failed to classify the input correctly and chose inputs which had the highest output softmax scores among the incorrectly classified inputs, i.e. the network was confidently wrong for those inputs. For the purpose of this subsequent analysis, we take the SSC-based affinity of the network's final convolution layer.




(a) ResNet-36 : Layer 1 (b) ResNet-36 : Layer 7 (c) ResNet-36 : Layer 14

(d) ResNet-36 : Layer 21 (e) ResNet-36 : Layer 28 (f) ResNet-36 : Layer 36

Fig. 8: Left to Right : Spectral Embeddings of Layer-wise Affinity graphs learned via a ResNet-36 - Data Set : CIFAR 10

In Figure 9a we show the input image of a cat that was incorrectly classified as a bird with a softmax score of 0.97. Figure 9b shows the normalised class-wise distribution of affinity scores for this input summed over all the images. This plot clearly shows the three strongest classes for which the given input has a strong affinity for, namely, bird, airplane and cat, in that order. At this juncture, we make a note that the predicted label, though incorrectly, of this input by the network seems to be the class for which this input has the highest affinity for as determined by Sparse Subspace Clustering Algorithm. Figure 9c- Figure 9j show the top-8 images, in descending order, for which the given input of cat has the highest affinity for.

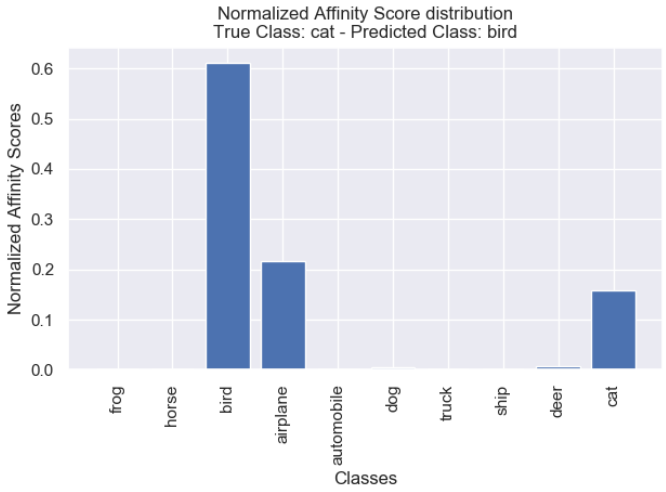
Continuing discussion on the previous observation, we further expand on that observation in Table 1, where we present the accuracy of the network on the testing set in two scenarios. The first scenario assigns an output label to a given input based on the highest aggregate class-wise affinity score for that input and the second scenario assigns the same output label to the input as the networks prediction through it's classification layers. The results of those are presented in columns 2 and 3 labeled 'SSC Label' and 'Network Prediction' respectively, in Table 1. We observe that both these 'accuracies' turn out to be 95.8%, Please be advised that the equality of values is a coincidence. Thus, given the activations of the final convolutional layer for the entire testing set we observe that the SSC based Affinity scoring assigns labels which are competitive with what the classification layers of the network could learn. This is further demonstrated in column 4 of Table 1 which shows a 98.3% agreement between one to one comparisons of network assigned labels with SSC derived labels. Row

3 and row 4 show the same metrics for other 2 networks, on CIFAR-10. Such a high annotation agreement between SSC and network prediction indicates a tendency of neural networks to separate out data points belonging to different classes into disjoint sections of the latent embedding space.

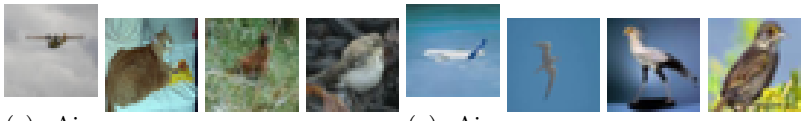
Table 1: **SSC Labels vs. Network Labels**: Comparison of Accuracy and Correlation on the test set

| Accuracy of Labels when compared to ground truth (CIFAR-10) | | | |
|---|---|---|---|
| Network | SSC Label | Network Prediction | Correlation |
| ResNet-34 | 95.8% | 95.8% | 98.3% |
| ResNet-18 | 95.4% | 96.1% | 97% |
| ResNet-50 | 90.6% | 94.8% | 93.5% |

(a) Input




(b) Neighbourhood Distribution (Normalised)



(c) Airplane (d) Cat (e) Bird (f) Bird (g) Airplane (h) Bird (i) Bird (j) Bird

Fig. 9: 9(a): A Cat, classified as a bird with a softmax score of 0.97. 9(b): Distribution of it's Neighbourhood Affinity Scores (Normalised). 9(c): Airplane. 9(d): Cat. 9(e): Bird. 9(f): Bird. 9(g): Airplane. 9(h): Bird. 9(i): Bird. 9(j): Bird. Network: ResNet-36 - Data Set : CIFAR 10

## 7 Conclusions

Our work proposes a two-step framework to analyze deep neural networks. This framework combines Sparse Subspace Clustering and Centered Kernel Alignment to provide the ability to analyze the network on a macro and micro level. The macro analysis helps in visualising network architectures as a function of network depth, width, training epochs and training data quantity and provides an insight into network architecture and training dynamics of the network. It also provides a framework to compare two architecturally different neural network based on a common set of inputs. The framework also provides the ability to micro analyze the network in the form of instance based interpretability by providing a measure of a degree of closeness each input has to decision boundaries for different classes in the loss landscape of a network.

## Acknowledgements

This research was supported by National Science Foundation Grant No. 1901379, the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Scientific Discovery through Advanced Computing (SciDAC) program, specifically the RAPIDS-2 SciDAC institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties. The authors would like to thank Tushar Nagarajan for helpful discussions and anonymous reviewers for critical feedback.

## References

1. Amaldi, E., Kann, V.: On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. Theor. Comput. Sci. **209**(1–2), 237–260 (Dec 1998). https://doi.org/10.1016/S0304-3975(97)00115-1, https://doi.org/10.1016/S0304-3975(97)00115-1
2. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations (2017)
3. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (Jan 2011). https://doi.org/10.1561/2200000016, https://doi.org/10.1561/2200000016
4. Cortes, C., Mohri, M., Rostamizadeh, A.: Algorithms for learning kernels based on centered alignment. CoRR **abs/1203.0550** (2012), http://arxiv.org/abs/1203.0550
5. Elhamifar, E., Vidal, R.: Sparse subspace clustering: Algorithm, theory, and applications. IEEE transactions on pattern analysis and machine intelligence **35**(11), 2765–2781 (2013)
6. Gretton, A., Bousquet, O., Smola, A., Scholkopf, B.: Measuring statistical dependence with hilbert-schmidt norms. In: Proceedings of the 16th International Conference on Algorithmic Learning Theory. p. 63–77. ALT'05, Springer-Verlag, Berlin, Heidelberg (2005). https://doi.org/10.1007/11564089_7, https://doi.org/10.1007/11564089_7

7. Hanin, B., Sellke, M.: Approximating continuous functions by relu nets of minimal width (2018)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
9. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2018)
10. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited (2019)
11. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep. (2009)
12. Lee, J., Bahri, Y., Novak, R., Schoenholz, S.S., Pennington, J., Sohl-Dickstein, J.: Deep neural networks as gaussian processes (2018)
13. Lin, H., Jegelka, S.: Resnet with one-neuron hidden layers is a universal approximator (2018)
14. Lu, Z., Pu, H., Wang, F., Hu, Z., Wang, L.: The expressive power of neural networks: A view from the width (2017)
15. Morcos, A.S., Raghu, M., Bengio, S.: Insights on representational similarity in neural networks with canonical correlation (2018)
16. Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences **103**(23), 8577–8582 (2006). https://doi.org/10.1073/pnas.0601602103, https://www.pnas.org/content/103/23/8577
17. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems. vol. 14. MIT Press (2002), https://proceedings.neurips.cc/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf
18. Nguyen, T., Raghu, M., Kornblith, S.: Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth (2020)
19. Novak, R., Xiao, L., Lee, J., Bahri, Y., Yang, G., Hron, J., Abolafia, D.A., Pennington, J., Sohl-Dickstein, J.: Bayesian deep convolutional networks with many channels are gaussian processes (2020)
20. Raghu, M., Gilmer, J., Yosinski, J., Sohl-Dickstein, J.: Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability (2017)
21. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps (2014)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015)
23. Vidal, R., Ma, Y., Sastry, S.S.: Generalized Principal Component Analysis. Springer Publishing Company, Incorporated, 1st edn. (2016)
24. Zagoruyko, S., Komodakis, N.: Wide residual networks (2017)
25. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks (2013)