JMLR: Workshop and Conference Proceedings 44 (2015) 196-212 NIPS 2015

The 1st International Workshop "Feature Extraction: Modern Questions and Challenges"

# Convergent Learning: Do different neural networks learn the same representations?

**Yixuan Li** YLI@CS.CORNELL.EDU
**Jason Yosinski** YOSINSKI@CS.CORNELL.EDU
*Cornell University*

**Jeff Clune** JEFFCLUNE@UWYO.EDU
*University of Wyoming*

**Hod Lipson** HOD.LIPSON@COLUMBIA.EDU
*Columbia University*

**John Hopcroft** JEH@CS.CORNELL.EDU
*Cornell University*

**Editor:** Afshin Rostamizadeh

## Abstract

Recent successes in training large, deep neural networks (DNNs) have prompted active investigation into the underlying representations learned on their intermediate layers. Such research is difficult because it requires making sense of non-linear computations performed by millions of learned parameters. However, despite the difficulty, such research is valuable because it increases our ability to understand current models and training algorithms and thus create improved versions of them. We argue for the value of investigating whether neural networks exhibit what we call *convergent learning*, which is when separately trained DNNs learn features that converge to span similar spaces. We further begin research into this question by introducing two techniques to approximately align neurons from two networks: a bipartite matching approach that makes one-to-one assignments between neurons and a spectral clustering approach that finds many-to-many mappings. Our initial approach to answering this question reveals many interesting, previously unknown properties of neural networks, and we argue that future research into the question of convergent learning will yield many more. The insights described here include (1) that some features are learned reliably in multiple networks, yet other features are not consistently learned; and (2) that units learn to span low-dimensional subspaces and, while these subspaces are common to multiple networks, the specific basis vectors learned are not; (3) that the average activation values of neurons vary considerably within a network, yet the mean activation values across different networks converge to an almost identical distribution.

## 1. Introduction

Many recent studies have focused on understanding deep neural networks from both a theoretical perspective (Arora et al., 2014; Neyshabur and Panigrahy, 2013; Montavon et al., 2011; Paul and Venkatasubramanian, 2014; Goodfellow et al., 2014) and from an empirical perspective (Eigen et al., 2013; Szegedy et al., 2013; Simonyan et al., 2013; Zeiler and Fergus, 2014; Nguyen et al., 2014; Yosinski et al., 2014; Mahendran and Vedaldi, 2014; Yosinski et al., 2015). In this paper we continue this trajectory toward attaining a deeper understanding

of neural net training by proposing a new approach. We begin by noting that modern deep neural networks (DNNs) exhibit an interesting phenomenon: networks trained starting at different random initializations frequently converge to solutions with similar performance (see Dauphin et al. (2014) and Section 2 below). Such similar performances by different networks raises the question of to what extent the learned internal representations differ: Do the networks learn radically different sets of features that happen to perform similarly, or do they exhibit *convergent learning*, meaning that their learned feature representations are largely the same?

This paper makes a first attempt at asking and answering these questions. Any improved understanding of what neural networks learn should improve our ability to design better architectures, learning algorithms, and hyperparameters, ultimately enabling more capable models. Specifically, we investigate the similarities and differences between the representations learned by neural networks with the same architecture trained from different random initializations. We employ an architecture derived from AlexNet (Krizhevsky et al., 2012) and train multiple networks on the ImageNet dataset (Deng et al., 2009) (details in Section 2). We then compare the representations learned across different networks. Our specific contributions are asking and shedding light on the following questions:

1. By defining a measure of similarity between units[1] in different neural networks, can we come up with a permutation for the units of one network to bring it into a one-to-one alignment with the units of another network trained on the same task? Is this matching or alignment close, because features learned by one network are learned nearly identically somewhere on the same layer of the second network, or is the approach ill-fated, because the representations of each network are unique? (Answer: a core representation is shared, but some rare features are learned in one network but not another; see Section 3).
2. Are the above one-to-one alignment results robust with respect to different measures of neuron similarity? (Answer: yes, under both linear correlation and estimated mutual information metrics; see Section 3.2).
3. When an accurate one-to-one neuron alignment is not possible, can we cluster groups of neurons from one network with a similar group from another network? (Answer: yes. To approximately match clusters, we introduce a hierarchical clustering algorithm that enables partial matches to be found between networks. We demonstrate the effectiveness of this method by both visually and quantitatively showing that the features learned by some neuron clusters in one network can be quite similar to those learned by neuron clusters in an independently trained neural network. See Section 4).

## 2. Experimental Setup

All networks in this study follow the basic architecture laid out by Krizhevsky et al. (2012), with parameters learned in five convolutional layers (conv1 – conv5) followed by three fully connected layers (fc6 – fc8). The structure is modified slightly in two ways. First, Krizhevsky et al. (2012) employed limited connectivity between certain pairs of layers to

1. Note that we use the words "filters", "channels", "neurons", and "units" interchangeably to mean channels for a convolutional layer or individual units in a fully connected layer.

enable splitting the model across two GPUs.[2] Here we remove this artificial group structure and allow all channels on each layer to connect to all channels on the preceding layer, as we wish to study only the group structure, if any, that arises naturally, not that which is created by architectural choices. Second, we place the local response normalization layers after the pooling layers following the defaults released with the Caffe framework, which does not significantly impact performance (Jia et al., 2014). Networks are trained using Caffe on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 dataset (Deng et al., 2009).

We trained four networks in the above manner using four different random initializations. We refer to these as Net1, Net2, Net3, and Net4. The four networks perform very similarly on the validation set, achieving top-1 accuracies of 58.65%, 58.73%, 58.79%, and 58.84%, which are similar to the top-1 performance of 59.3% reported in the original study (Krizhevsky et al., 2012).

We then aggregate certain statistics of the activations within the networks. Given a network Net$n$ trained in this manner, we use the scalar random variable $X_{l,i}^{(n)}$ to denote the series of activation values produced over the entire ILSVRC validation dataset by unit $i$ on layer $l \in \{\text{conv1}, \text{conv2}, \text{conv3}, \text{conv4}, \text{conv5}, \text{fc6}, \text{fc7}\}$.[3] We collect the following statistics by aggregating over the validation set (and in the case of convolutional layers also over spatial positions):

$$\begin{aligned}
\text{Mean:} \quad & \mu_{l,i}^{(n)} &&= \mathbb{E}[X_{l,i}^{(n)}] \\
\text{Standard deviation:} \quad & \sigma_{l,i}^{(n)} &&= \sqrt{(}\mathbb{E}[(X_{l,i}^{(n)} - \mu_{l,i}^{(n)})^2]) \\
\text{Within-net correlation:} \quad & c_{l,i,j}^{(n)} &&= \mathbb{E}[(X_{l,i}^{(n)} - \mu_{l,i}^{(n)})(X_{l,j}^{(n)} - \mu_{l,j}^{(n)})]/\sigma_{l,i}^{(n)}\sigma_{l,j}^{(n)} \\
\text{Between-net correlation:} \quad & c_{l,i,j}^{(n,m)} &&= \mathbb{E}[(X_{l,i}^{(n)} - \mu_{l,i}^{(n)})(X_{l,j}^{(m)} - \mu_{l,j}^{(m)})]/\sigma_{l,i}^{(n)}\sigma_{l,j}^{(m)}
\end{aligned}$$

Intuitively, we compute the mean and standard deviation of the activation of each unit in the network over the validation set. For convolutional layers, we compute the mean and standard deviation of each channel. The mean and standard deviation for a given network and layer is a vector with length equal to the number of channels (for convolutional layers) or units (for fully connected layers).[4] The within-net correlation values for each layer can be considered as a symmetric square matrix with side length equal to the number of units in that layer (e.g. a $96 \times 96$ matrix for conv1 as in Figure 1a,b). For a pair of networks, the between-net correlation values also form a square matrix, in this case non-symmetric (Figure 1c,d).

2. In Krizhevsky et al. (2012) the conv2, conv4, and conv5 layers were only connected to half of the preceding layer's channels.

3. For the fully connected layers, the random variable $X_{l,i}^{(n)}$ has one specific value for each input image; for the convolutional layers, the value of $X_{l,i}^{(n)}$ takes on different values at each spatial position. In other words, to sample an $X_{l,i}^{(n)}$ for an FC layer, we pick a random image from the validation set; to sample $X_{l,i}^{(n)}$ for a conv layer, we sample a random image and a random position within the conv layer.

4. For reference, the number of channels for conv1 to fc8 is given by: $\mathcal{S} = \{96, 256, 384, 384, 256, 4096, 4096, 1000\}$. The corresponding size of the correlation matrix in each layer is: $\{s^2 \mid \forall s \in \mathcal{S}\}$. Furthermore, the size of each channel in each convolutional layer is given by: $\{\text{conv1} : 55 \times 55, \text{conv2} : 27 \times 27, \text{conv3} : 13 \times 13, \text{conv4} : 13 \times 13, \text{conv5} : 13 \times 13\}$

We use these correlation values as a way of measuring how related the activations of one unit are to another unit, either within the network or between networks. We use correlation to measure similarity because it is independent of the scale of the activations of units Within-net correlation quantifies the similarity between two neurons in the same network; whereas the between-correlation matrix quantifies the similarity of two neurons from different neural networks. Note that the units compared are always on the same layer on the network; we do not compare units between different layers. In the Supplementary Information (see Figure 7), we plot the activation values for several example high correlation and low correlation pairs of units from conv1 and conv2 layers; the simplicity of the distribution of values suggests that the correlation measurement is an adequate indicator of the similarity between two neurons. Furthermore, we adopt *mutual information* as a complementary measurement of similarity between the representations of two neurons (see Section 3.2). Our results have shown that using mutual information yields qualitatively the same results as using correlation statistics.

## 3. Is There a One-to-One Alignment Between Features Learned by Different Neural Networks?

We would like to investigate the similarities and differences between multiple training runs of same network architecture. Due to symmetries in the architecture and weight initialization procedures, for any given parameter vector that is found, one could create many equivalent solutions simply by permuting the unit orders within a layer (and permuting the outgoing weights accordingly). Thus, as a first step toward analyzing the similarities and differences between different networks, we ask the following question: if we allow ourselves to permute the units of one network, to what extent can we bring it into alignment with another? To do so requires finding equivalent or nearly-equivalent units across networks, and for this task we adopt the magnitude independent measures of correlation and mutual information. We primarily give results with the simpler but faster to calculate measure of correlation (Section 3.1) but then confirm that using the complete mutual information does not significantly change the results (Section 3.2).

### 3.1 Alignment via Correlation

As discussed in Section 2, we compute within-net and between-net unit correlations. Figure 1 shows the within-net correlation values computed between units on a network and other units on the same network (panes a,b) as well as the between-net correlations between two different networks (pane c). We find matching units between a pair of networks — here Net1 and Net2 — in two ways. In the first approach, for each unit in Net1, we simply find the unit in Net2 with maximum correlation, or the max along each row of Figure 1c. This type of assignment is known as a bipartite *semi-matching* in graph theory (Lawler, 1976), and we adopt the same nomenclature here. This procedure can result in multiple units of Net1 being paired with the same unit in Net2. Figure 2 shows the eight highest correlation matched features and eight lowest correlation matched features using the semi-matching approach (corresponding to the leftmost eight and rightmost eight points in Figure 3). The filters on the left have nearly perfect counterparts in each network, but the filters on the right are unique to one network or the other.

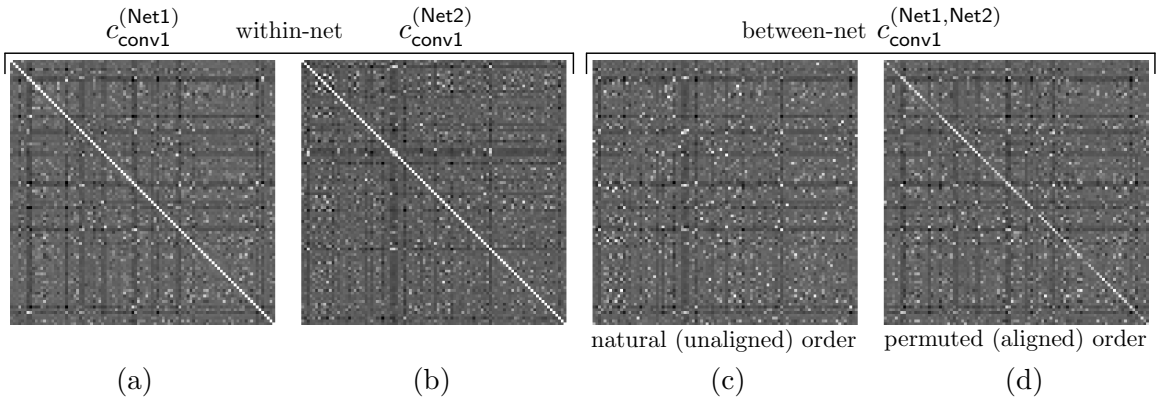



Figure 1: Correlation matrices for the conv1 layer, displayed as images with minimum value at black and maximum at white. **(a,b)** Within-net correlation matrices for Net1 and Net2, respectively. **(c)** Between-net correlation for Net1 vs. Net2. **(d)** Between-net correlation for Net1 vs. a version of Net2 that has been permuted to approximate Net1's feature order. The partially white diagonal of this final matrix shows the extent to which the alignment is successful; see Figure 3 for a plot of the values along this diagonal and further discussion.

An alternative approach is to find the one-to-one assignment between units in Net1 and Net2 without replacement, such that every unit in each network is paired with a unique unit in the other network. This the known as the more common bipartite *matching*.[5] A matching that maximizes the sum of the chosen correlation values may be found efficiently by turning the between-net correlation matrix into a weighted bipartite graph and using the Hopcroft-Karp algorithm (Hopcroft and Karp, 1973) to find the max weighted matching. Figure 1c shows an example between-net correlation matrix; the max weighted matching can be thought of as a path through the matrix such that each row and each column are selected exactly once, and the sum along the path is maximized. Once such a path is found, we can permute the units of Net2 to bring it into the best possible alignment with Net1, so that the first channel of Net2 approximately matches (has high correlation with) the first channel of Net1, the second channels of each also approximately match, and so on. The correlation matrix of Net1 with the permuted version of Net2 is shown in Figure 1d. Whereas the diagonal of the self correlation matrices are exactly one, the diagonal of the permuted between-net correlation matrix contains values that are generally less than one. Note that the diagonal of the permuted between-net correlation matrix is bright (close to white) in many places, which shows that for many units in Net1 it is possible to find a unique unit in Net2 with highly correlated activation values.

Figure 3 shows a comparison of assignments produced by both methods, semi-matching and matching, for the conv1 layer; see Figure 8 – Figure 11 in the Supplementary Information for results for other layers. In Figure 3 both the semi-matching and matching are found for each unit, and then the units are sorted in order of decreasing semi-matching value and both correlation values are plotted. Insights into the differing representations learned

5. Note that the *semi-matching* is "row-wise greedy" and will always have equal or better sum of correlation than the *matching*, which maximizes the same objective but must also satisfy global constraints.

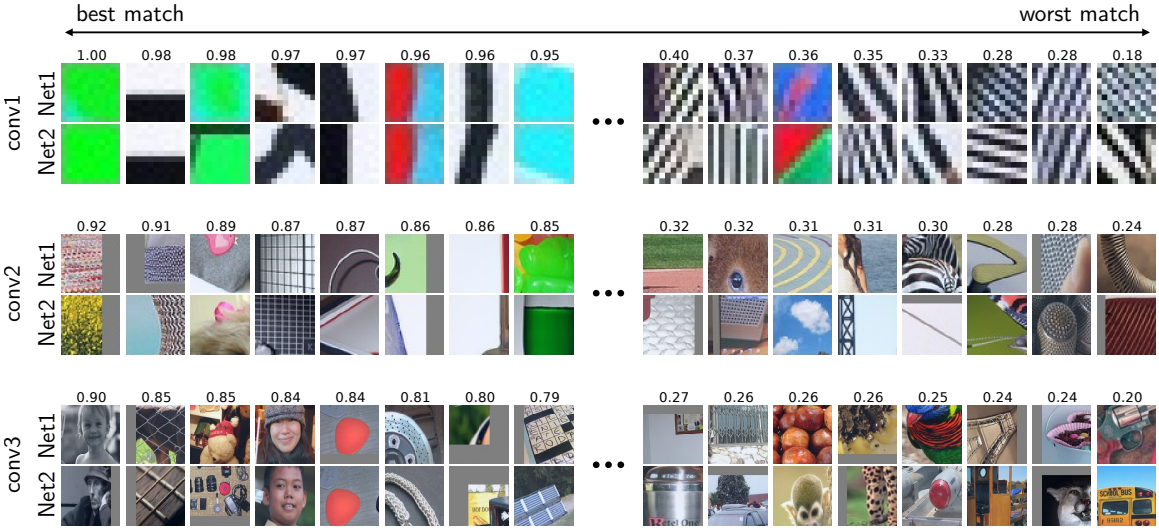



Figure 2: With assignments chosen by semi-matching, the eight best (highest correlation, left) and eight worst (lowest correlation, right) matched features between Net1 and Net2 for the conv1 – conv3 layers.

can be gained from both types of assignment. The first conclusion is that for many units, particularly those with the higher greedy match correlations on the left, the greedy and max match assignments coincide. Such correspondence between the two matching methods shows that for many units a one-to-one assignment is possible. The third column of Table 1 gives the ratio of units that overlap: in conv1 the overlap is 72.9%, but the extent of one-to-one neuron assignment varies from layer to layer — with the highest matching capability in conv1 and conv5 a lower values in conv2 – conv4. The results of average correlation between the paired neurons in different layers also implies that the path from (perfectly matchable) three-channel pixels and a relatively matchable conv1 representation to the relatively matchable conv5 representation passes through an intermediate middle region where matching is more difficult. This may be related to previously observed greater complexity in the intermediate layers as measured through the lens of optimization difficulty (Yosinski et al., 2014).

Next, we can see that where the semi-matching and matching differ, the matching is often much worse. One hypothesis for why this occurs is that the two networks learn different numbers of units to span certain subspaces. For example, Net1 might learn a representation that uses six filters to span a subspace of human faces, but Net2 learns to span the same subspace with five filters. In the max matching, five out of the six filters from Net1 may be matched to their nearest counterpart in Net2, but the sixth Net1 unit will be left without a counterpart and will end up paired with an almost unrelated filter.

Finally, with reference to Figure 3 (but similarly observable in Figure 8 and Figure 9), another salient observation is that the correlation of the greedy match falls significantly from the best-matched unit (correlations near 1) to the lowest-matched (correlations near 0.3). This indicates that some filters in Net1 can be paired up with filters in Net2 with high correlation, but other filters in Net1 and Net2 are network-specific and have no high-

| | Semi-matching | Matching | Overlap ratio |
|---|---|---|---|
| conv1 | 0.703 | 0.663 | 72.9% |
| conv2 | 0.577 | 0.527 | 58.6% |
| conv3 | 0.483 | 0.445 | 61.2% |
| conv4 | 0.374 | 0.348 | 54.7% |
| conv5 | 0.577 | 0.527 | 68.4% |

Table 1: The average correlation between neurons paired using a semi-matching (first column) or a matching (second column). The third column shows the fraction of pairs where the greedy and max match coincide.

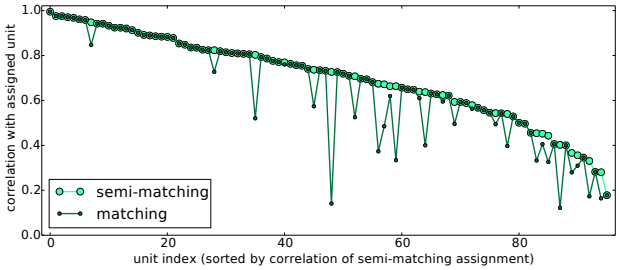



Figure 3: Correlations between conv1 units in Net1 and their paired conv1 unit in Net2, where pairings are made via semi-matching (large light green circles) or matching (small dark green dots). See text for discussion.

correlation pairing in the alternate network, implying that those filters are rare and not always learned. This holds across at least the conv1 – conv3 layers.

### 3.2 Alignment via Mutual Information

Because the correlation used in the previous section is a simple metric that may miss some forms of statistical dependence, we also find alignments using *mutual information* as a measurement of mutual dependence between the activations of two neurons. Mutual information between two variables essentially measures how much knowledge one gains about one variable by knowing the value of the other. The within-net mutual information matrix and between-net mutual information matrix are similarly shaped as the correlation matrices.

As a sanity check, we apply the same matching technique described in Section 3.1 to the between-net mutual information matrices.[6] Figure 4 shows the eight highest mutual information matched features and eight lowest mutual information matched features. Compared to the results using correlation matrices in Figure 2, we find that observations from using mutual information estimator is largely similar to that of using correlation similarity. For example, seven out of eight highest matched pairs in the conv1 layer stay the same. This suggests that the correlation is an adequate measurement of the similarity between two neurons.

## 4. Does Relaxing the One-to-One Constraint to Find Many-to-Many Groupings Reveal More Similarities Between what Different Networks Learn?

Since the preceding sections have shown that neurons may not necessarily correspond via a globally consistent one-to-one matching between networks, we now seek to find many-to-many matchings between networks using a spectral clustering approach (Ng et al., 2001).

6. The mutual information between each pair of neurons is estimated using 1D and 2D histograms of paired activation values over 60,000 random activation samples. We discretize the activation value distribution using percentile bins along each dimension, each of which captures 5% of the marginal distribution mass. We also add a special bin with range $(-\inf, 10^{-6}]$ in order to capture the significant mass around 0.

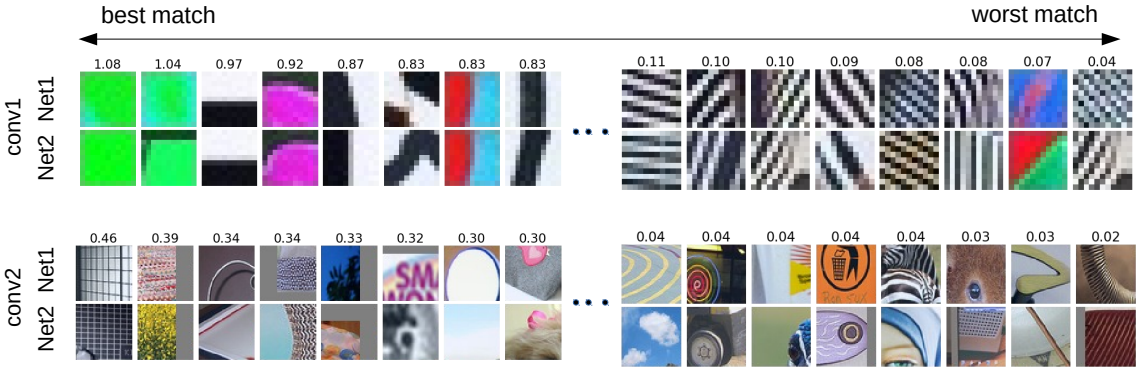



Figure 4: The eight best (highest mutual information, left) and eight worst (lowest mutual information, right) features in the semi-matching between Net1 and Net2 for the conv1 and conv2 layers.

### 4.1 Spectral Clustering and Metrics for Neuron Clusters

We define three types of similarity graphs based on the correlation matrices obtained above (see Section 6.5 for definition). Define $W_l \in \mathbb{R}^{2\mathcal{S}_l \times 2\mathcal{S}_l}$ to be the combined correlation matrices between two DNNs, $X$ and $Y$ in layer $l$, where $w_{jk}$ is the entry at $j$th row and $k$th column of that matrix. $\mathcal{S}_l$ is the number of channels (units) in layer $l$. $W_l$ is given by

$$W_l = \begin{bmatrix} \text{corr}(X_l, X_l) & \text{corr}(X_l, Y_l) \\ \text{corr}(X_l, Y_l)^\top & \text{corr}(Y_l, Y_l) \end{bmatrix}.$$

The unnormalized Laplacian matrix is defined as $L_l = D_l - W_l$, where the degree matrix $D_l$ is the diagonal matrix with the degrees $d_j = \sum_{k=1}^{2\mathcal{S}_l} w_{jk}$. The unnormalized Laplacian matrix and its eigenvalues and eigenvectors can be used to effectively embed points in a lower-dimension representation without losing too information about spatial relationships. If neuron clusters can be identified, then the Laplacian $L_l$ is approximately block-diagonal, with each block corresponding a cluster. Assuming there are $k$ clusters in the graph, spectral clustering would then take the first $k$ eigenvectors[7], $\mathcal{U} \in \mathbb{R}^{2\mathcal{S}_l \times k}$, corresponding to the $k$ smallest eigenvalues, and partition neurons in the eigenspace with the $k$-means algorithm.

### 4.2 Spectral Clustering Results

We use Net1 and Net2 as an example for showing the results of matching neurons between DNNs. Figure 5 displays the top 8 neuron clusters with highest between-net similarity measurement (See Section 6.6 for definition) in conv2 layer. The matching results imply that there exists many-to-many correspondence of the feature maps between two fully trained networks with different random initializations, and the number of neurons learning the same feature can be different between networks. For example, the four units of {89, 90, 134, 226} in Net1 and three units of {2, 39, 83} in Net2 are learning the features about green objects.

We also computed and visualized the matching neurons in other layers as well. The results of conv1 are appended in the supplementary material Figure 12. As noted earlier,

7. In practice, when the number of clusters is unknown, the best value of $k$ to choose is where where the eigenvalue shows a relatively abrupt change.

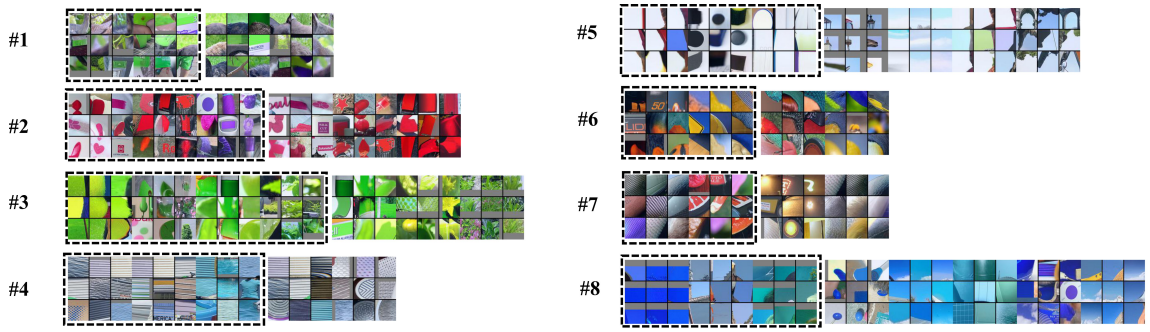



Figure 5: The neuron matchings between two DNNs: Net1 and Net2. Each of the $3 \times 3$ block displays the top 9 image patches that cause the highest activations to each neuron. Each labeled half-row corresponds to one cluster, where the filter visualizations with dashed boxes represent neurons from Net1 and those without are from Net2. For example, there are 7 neurons learning similar features in cluster #3, where the left four neurons are in Net1 and the right three are from Net2. Best viewed in electronic form with zoom.

the conv1 layer tends to learn more general features like Gabor filters (edge detectors) and blobs of color. Our approach finds many matching Gabor filters (e.g., clusters #5, #6, #10, #11 and #12), and also some matching color blobs (e.g., clusters #1, #3 and #4).

### 4.3 Hierarchical Spectral Clustering Results

Due to the stochastic effects of randomly initializing centroids in $k$-means clustering, some of the initial clusters contains more neurons than others. To get more fine-grained cluster structure, we recurrently apply $k$-means clustering on any clusters with size $> 2\alpha \cdot \mathcal{S}_l$, where $\alpha$ is a tunable parameter for adjusting the maximum size of the leaf clusters. Figure 6 shows the partial hierarchical structure of neuron matchings in the conv2 layer. The cluster at the root of the tree is a first-level cluster that contains many similar units from both DNNs. Here we adopt $\alpha = 0.025$ for the conv2 layer, resulting in a hierarchical neuron cluster tree structure with leaf clusters containing less than 6 neurons from each network. The bold box of each subcluster contains neurons from Net1 and the remaining neurons are from Net2. For example, in subcluster #3, which shows conv2 features, units {62, 137, 148} from Net1 learned similar features as units {33, 64, 230} from Net2, namely, red and magenta objects.

## 5. Conclusions

We have demonstrated a method for quantifying the feature similarity between different deep neural networks. We show how insight may be gained by approximately aligning units from two networks either via a matching approach or a softer spectral clustering approach borrowed from research in community detection in large graphs. We find that some features are learned repeatedly in multiple networks, but other rare features are not always learned. Our findings may shed light on several future research directions, for example:

1. Model compression. Would removing low-correlation, rare filters affect performance?

2. Optimizing ensemble formation. The results show some features (and subspaces) are shared between independently trained DNNs, and some are not. This suggests

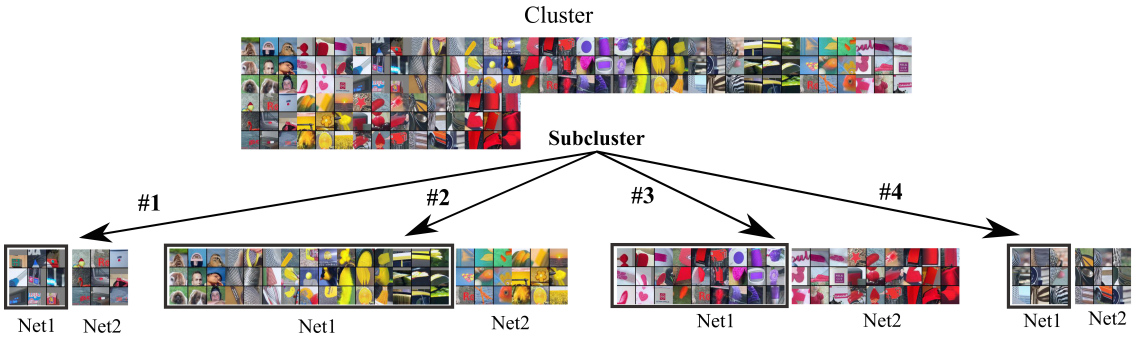



Figure 6: The hierarchical structure of neuron matchings between two DNNs: Net1 and Net2 (conv2 layer). The initial clusters are obtained using spectral clustering with the number of clusters $k = 100$ and threshold $\tau = 0.2$.

testing how feature correlation among different DNNs in an ensemble affects ensemble performance. Networks' "shared cores" could be deduplicated, but unique features in the tails of the feature distribution could be kept.

3. Similarly, one could (a) post-hoc assemble ensembles with greater diversity, or even (b) directly tune for it during training.
4. Certain visualization techniques (max-patch, deconv, deepvis) have revealed neurons with multiple functions (e.g. detectors firing for wheels and faces). Our matching methods could reveal more about why these arise. Are these units consistently learned because they are helpful or just noisy, imperfect features found in local optima?
5. Model combination: can multiple models be combined by concatenating their features, deleting those with high overlap, and fine-tuning?
6. Employing the same matching analysis on networks trained for different tasks could provide a quick check for how similar or different the various features are.

Acknowledgments: The authors are grateful to the NASA Space Technology Research Fellowship (JY) for funding, Wendy Shang for conversations and initial ideas, and Anh Nguyen and Kilian Weinberger for helpful comments and edits. This work was supported in part by US Army Research Office W911NF-14-1-0477, NSF grant 1527232. Jeff Clune was supported by an NSF CAREER award (CAREER: 1453549).

## References

Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *ICML*, pages 584–592, 2014.

Yann Dauphin, Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *CoRR*, abs/1406.2572, 2014. URL `http://arxiv.org/abs/1406.2572`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

David Eigen, Jason Rolfe, Rob Fergus, and Yann LeCun. Understanding deep architectures using a recursive convolutional network. *arXiv preprint arXiv:1312.1847*, 2013.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, December 2014.

John E Hopcroft and Richard M Karp. An n^5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 1976.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. *arXiv preprint arXiv:1412.0035*, 2014.

Grégoire Montavon, Mikio L Braun, and Klaus-Robert Müller. Kernel analysis of deep networks. *The Journal of Machine Learning Research*, 12:2563–2581, 2011.

Behnam Neyshabur and Rina Panigrahy. Sparse matrix factorization. *arXiv preprint arXiv:1311.3315*, 2013.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *arXiv preprint arXiv:1412.1897*, 2014.

Arnab Paul and Suresh Venkatasubramanian. Why does deep learning work?-a perspective from group theory. *arXiv preprint arXiv:1412.6621*, 2014.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034, presented at ICLR Workshop 2014*, 2013.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

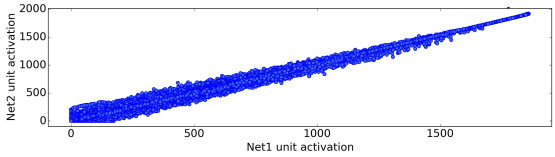
J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., December 2014.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

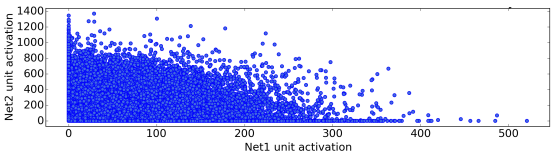
Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.

# 6. Appendix

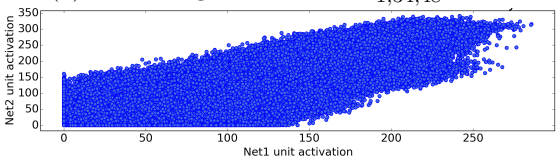
## 6.1 Activation Values: High Correlation vs. Low Correlation



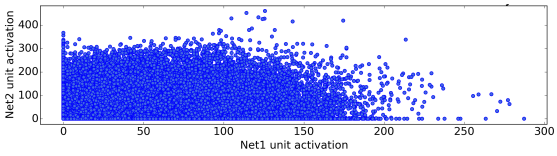

(a) conv1, high correlation $c^{(1,2)}_{1,34,48}$: 0.995




(b) conv1, low correlation $c^{(1,2)}_{1,2,74}$: 0.178




(c) conv2, high correlation $c^{(1,2)}_{2,122,236}$: 0.92




(d) conv2, low correlation $c^{(1,2)}_{2,30,26}$: 0.24

Figure 7: Activation values are computed across 5,000 randomly sampled images and all spatial positions ($55 \times 55$ and $27 \times 27$ for layers conv1 and conv2, respectively). The joint distributions appear simple enough to suggest that a correlation measure is sufficient to find matching units between networks.

## 6.2 Additional Hard Match Results

Figure 8 – Figure 11 shows additional results of comparison of assignments produced by greedy-match and max-match methods in conv2 – conv5. For each unit, both the greedy match and max match are found, then the units are sorted in order of decreasing greedy match value and both correlation values are plotted.

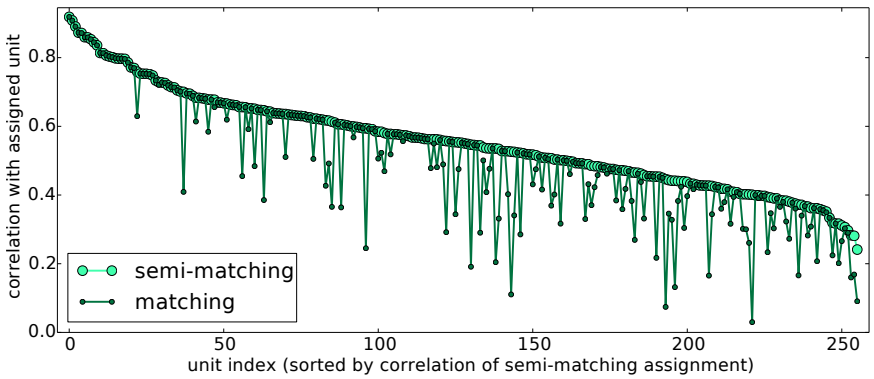



Figure 8: Match results for conv2 in the manner of Figure 3.

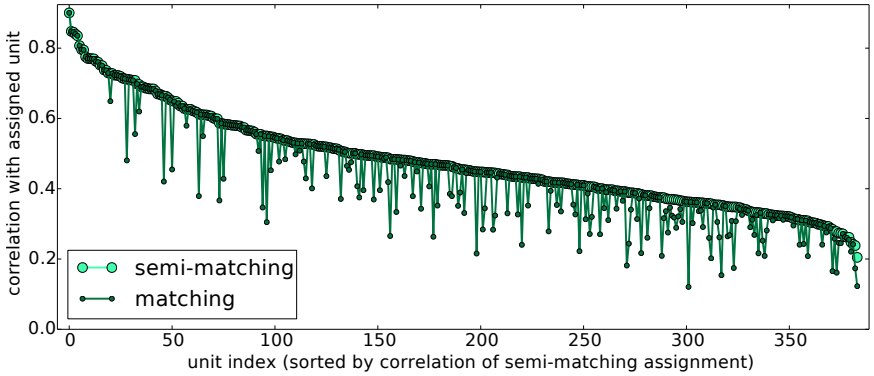



Figure 9: Match results for conv3 in the manner of Figure 3.

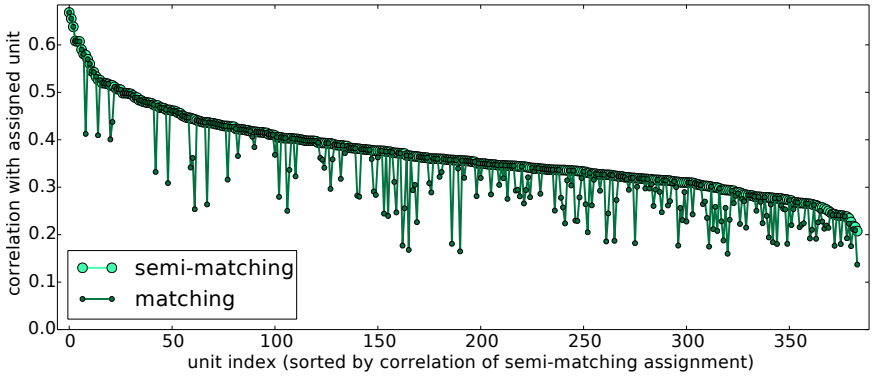



Figure 10: Match results for conv4 in the manner of Figure 3.

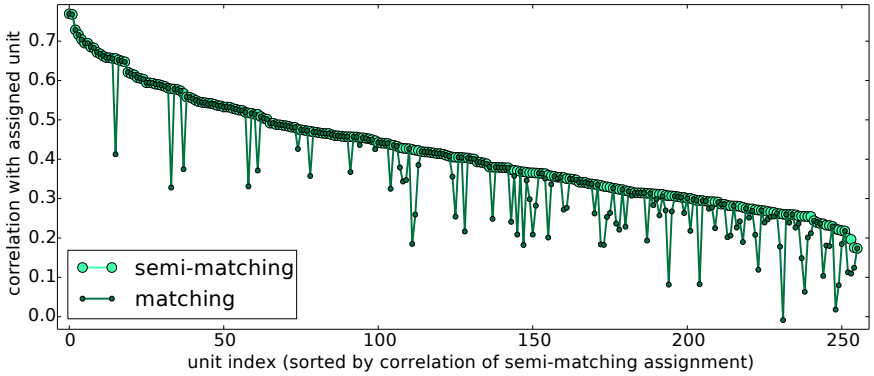



Figure 11: Match results for conv5 in the manner of Figure 3.

### 6.3 Matching Neurons Between DNNs

Figure 12 displays 12 neuron clusters with high between-net similarity measurement in conv1 layer.

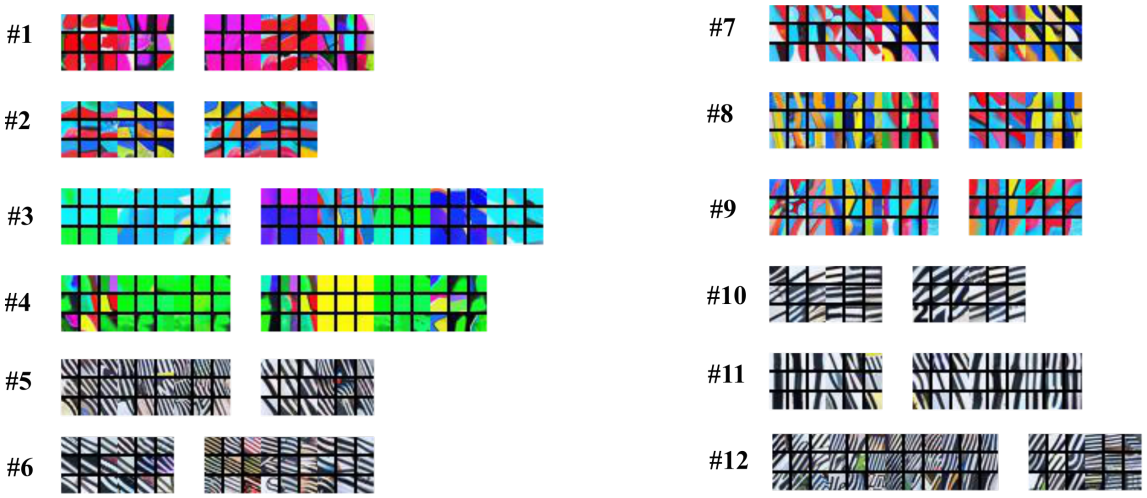



Figure 12: The neuron matchings between two DNNs (Net1 and Net2) in conv1 layer. Here we display the 12 neuron clusters with relatively high between-net similarity measurement. Each labeled half-row corresponds to one cluster, where the filter visualizations for neurons from Net1 and Net2 are separated by white space slot. The matching results imply that there exists many-to-many correspondence of the feature maps between two fully trained networks with different random initializations. For instance, in cluster #6, neurons from Net1 and Net2 are both learning 135° diagonal edges; and neurons in cluster #10 and #12 are learning 45° diagonal edges.

### 6.4 Diagonal Structure of Correlation Matrix After Clustering

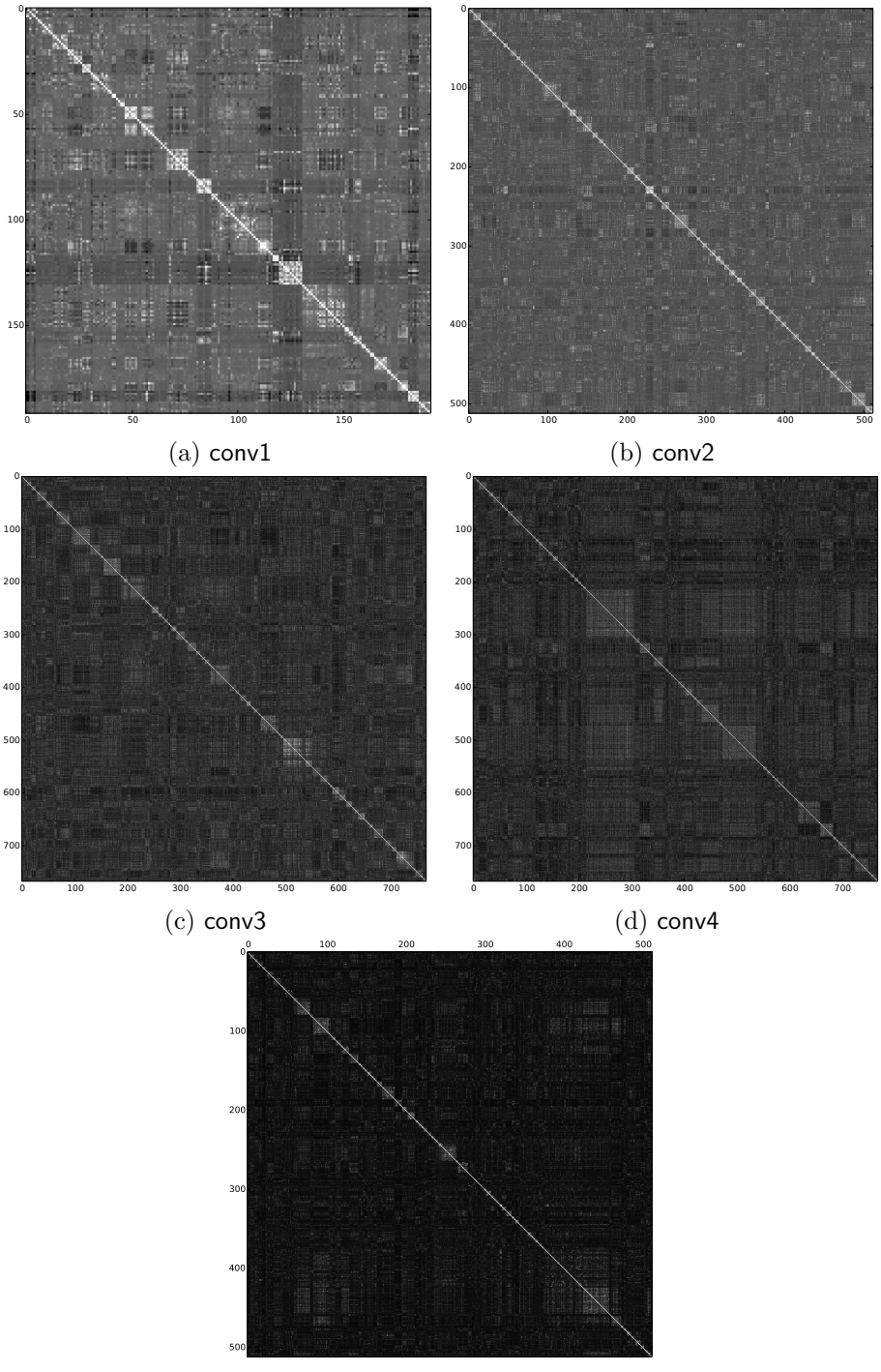
Figure 13 shows the permuted combined correlation matrix after apply the spectral clustering algorithm for conv1 – conv5.

### 6.5 Neuron Similarity Graphs

We define three types of similarity graphs based on the correlation matrices obtained above.

**Single-net neuron similarity graphs.** Given a fully trained DNN $X$ and a specified layer $l$, we first construct the *single-net neuron similarity graph* $G_{X,l} = (V, E)$. Each vertex $v_p$ in this graph represents a unit $p$ in layer $l$. Two vertices are connected by an edge of weight $c_{pq}$ if the correlation value $c_{pq}$ in the self-correlation matrix $\text{corr}(X_l, X_l)$ between unit $p$ and unit $q$ is greater than a certain threshold $\tau$.

**Between-net neuron similarity graphs.** Given a pair of fully trained DNNs $X$ and $Y$, the *between-net neuron similarity graph* $G_{XY,l} = (V, E)$ can be constructed in a similar manner. Note that $G_{XY,l}$ is a bipartite graph and contains twice as many vertices as that in $G_{X,l}$ since it incorporates units from both networks. Two vertices are connected by an edge



(a) conv1 (b) conv2

(c) conv3 (d) conv4

(e) conv5

Figure 13: The permuted combined correlation matrix after apply spectral clustering method (conv1 – conv5). The diagonal block structure represents the groups of neurons that are clustered together. The value of $k$ adopted for these five layers are: {40,100,100,100,100}, which is consistent with the parameter setting for other experiments in this paper.

of weight $c_{pq}$ if the correlation value $c_{pq}$ in the between-net correlation matrix $\text{corr}(X_l, Y_l)$ between unit $p$ in $X$ and unit $q$ in $Y$ is greater than a certain threshold $\tau$.

**Combined similarity graphs.** The problem of matching neurons in different networks can now be reformulated as finding a partition in the combined neuron similarity graphs $G_{X+Y,l} = G_{X,l} + G_{Y,l} + G_{XY,l}$, such that the edges between different groups have very low weights and the edges within a group have relatively high weights.

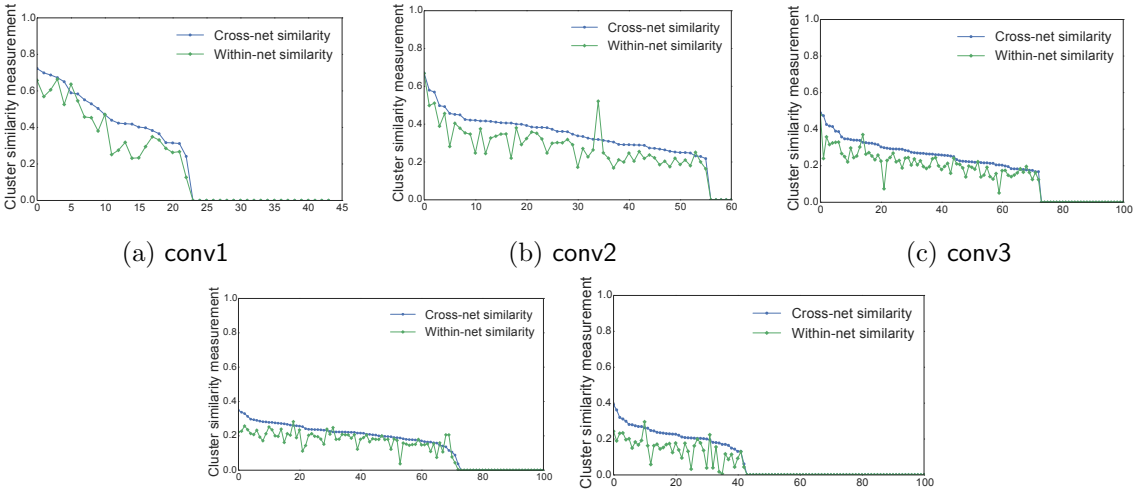
### 6.6 Neuron Cluster Similarity Measurement

Here we introduce two metrics for quantifying the similarity among neurons grouped together after applying the clustering algorithm above.

$$\text{Between-net similarity:} \quad \text{Sim}_{X_l \to Y_l} = (\sum_{p=1}^{\mathcal{S}_l} \sum_{q=1}^{\mathcal{S}_l} \text{corr}(X_l, Y_l)_{pq})/\mathcal{S}_l^2$$

$$\text{Within-net similarity:} \quad \text{Sim}_{X_l, Y_l} = (\text{Sim}_{X_l \to X_l} + \text{Sim}_{Y_l \to Y_l})/2$$

We further performed experiments in quantifying the similarity among neurons that are clustered together. Figure 14 shows the between-net and within-net similarity measurement for conv1 – conv5. The value of $k$ for initial clustering is set to be 40 for conv1 layer and 100 for all the other layers. In our experiments, the number of final clusters obtained after further hierarchical branching is {43, 113, 130, 155, 131}. The tail in those curves with value 0 is due to the non-existence of between-net similarity for the clusters containing neurons from only one of the two DNNs. To better capture the distribution of non-zero similarity values, we leave out the tail after 100 in the plot for conv3 - conv5 layers.




(a) conv1 (b) conv2 (c) conv3

(d) conv4 (e) conv5

Figure 14: The distribution of between-net and within-net similarity measurement after clustering neurons (conv1 – conv5). The x-axis represents obtained clusters, which is reshuffled according to the sorted between-net similarity value.