Inference Graphs for CNN Interpretation

Anonymous ECCV submission

Paper ID 4871

Abstract. Convolutional neural networks (CNNs) have achieved superior accuracy in many visual related tasks. However, the inference process through intermediate layers is opaque, making it difficult to interpret such networks or develop trust in their operation. We propose to model the network hidden layers activity using probabilistic models. The activity patterns in layers of interest are modeled as Gaussian mixture models, and transition probabilities between clusters in consecutive modeled layers are estimated. Based on maximum-likelihood considerations, a subset of the nodes and paths relevant for network prediction are chosen, connected, and visualized as an inference graph. We show that such graphs are useful for understanding the general inference process of a class, as well as explaining decisions the network makes regarding specific images. In addition, the models provide an interesting observation regarding the highly local nature of column activities in top CNN layers.

Keywords: Deep Neural Networks, Visualization, Interpretable AI

1 Introduction

Thanks to their impressive performance, convolutional neural networks (CNNs) are the leading architecture for tasks in computer vision [1,2,3]. However, due to their complex end-to-end training and architecture, understanding of their decision-making process and task assignment across hidden layers is lacking. This turns network interpretability into a difficult problem, and undermines usage of deep networks when high reliability and inference transparency are required. Understanding CNN reasoning by decomposing it into layer-wise stages can provide insights about cases of failure, and reveal weak spots in the network architecture, training scheme, or data-collection mechanism. In turn, these insights can lead to more robust networks and develop more trust in CNN decisions.

As we see it, enabling human understanding of the deep network inference process requires facing a main challenge of transforming CNN activities into discrete representations amendable to human reasoning. Deep networks operate through a series of distributed layer representations. Human language, however, is made up of discrete symbols, i.e., words, having meaning grounded by their reference in the world of objects, predicates, and their interrelations. The question of interest in this respect is: *Can we convert distributed representations into a human-oriented language?*, more technically, can we learn a dictionary of visual words and model their interrelationships leading to an interpretable inference graph?

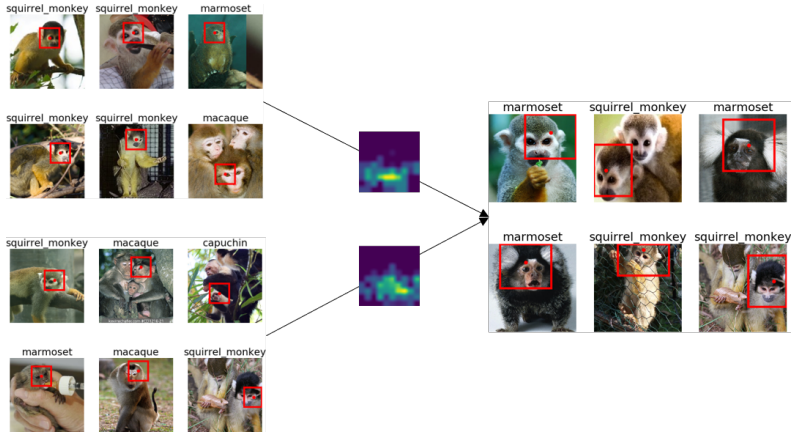

Fig. 1. Visual words connected to form an inference path. Two visual words in a lower network layer (left) (each is represented by six most typical images), representing the monkey eye (top) and face (bottom), explain a visual word in an upper network layer (right) that represents the monkey face. The heatmaps show spatial densities of lower-layer visual words in the receptive field of the higher-level word (see Section 4.3).

To address this, we suggest to describe the inference process of a network with probabilistic models. We model activity vectors in each layer as arising from a multivariate Gaussian mixture model (GMM). Layer activity in fully connected (FC) layers, or spatial location activity in convolutional layers, is associated with one of K clusters (GMM components), each representing a visual word. Connections between visual words of consecutive layers are modeled using conditional probabilities. For a multilayer perceptron (MLP) network, a full model with efficient inference can be obtained using a hidden Markov model (HMM). For the convolutional layers, each spatial location has its own hidden variable and dependencies among visual words in consecutive layers are described using conditional probability tables. Given a selected subset of images to be explained (either a specific image or images of an entire class), we describe the decision process of the network using an inference graph, representing the visual words used in different hidden layers and their probabilistic connections. As the full graph may contain thousands of visual words, a useful explanation has to find informative subgraphs, containing the most explanatory words w.r.t the network decision, and the likely paths connecting them. We suggest an algorithm for finding such graphs based on maximum-likelihood considerations.

Our contributions are: (i) a new approach for network interpretation, providing a formalism for probabilistic reasoning about inference processes in deep networks, and (ii) a graph-mining algorithm and visual tools enabling inference understanding. Our suggested inference graph provides a succinct summary of the inference process performed on a specific class of images or a single image, as inference progresses through the network layers. An example of visual nodes and the inferred connection between them is shown in Fig. 1.

2 Related Work

Network visualization. Several techniques have been suggested to visualize network behavior. In activation maximization [4], the input that maximizes the score of a given hidden unit is visualized by carrying out regularized gradient ascent optimization in the image space, as was applied to output class neurons [5] and intermediate layer neurons [6]. Another technique [5] visualizes the gradient strength in the original image space for a specific example, providing a "saliency map" showing class score sensitivity to image pixels. The idea was extended to an entire class of interest in [7]. In [8], the role of neurons in intermediate layers was visualized by an inverse de-convolution network.

Simplifying network representation. Similar to our work, several works looked for categorizing features through clustering [9,10]. Liao et al. [9] added a regularization term encouraging the network representation to form three kinds of clusters governed by examples, spatial locations, and channels. A similar approach was introduced for learning class discriminative clusters of spatial columns [10]. However, these approaches influence the trained network and trade accuracy for explainability, whereas ours finds meaningful inference explanations without interfering with the network learning process.

Modeling relationships between consecutive layers. In CNNVis [11], neurons in each layer are clustered to form groups having similar activity patterns. For the clustering, a neuron was described using a C-dimensional vector of its average activity on each class $1, .., C$. A graph between clusters of subsequent layers was then formed based on the average weight strengths between cluster neurons. Note that this method clusters neurons, while we cluster activity vectors (of neuronal columns) across examples. Olah et al. [12] proposed a tool for visualizing the network path for a single image. They decomposed each layer's activations into groups using matrix factorization, and connected groups from consecutive layers into a graph structure similar to [11].

3 Method

Inference graphs for an MLP, for which a full graphical model can be suggested, are presented in Section 3.1. The more general case of a CNN is discussed in Section 3.2, and its related graph-mining algorithm in Section 3.3. Models can be trained on the full set of network layers or on a subset, indexed by $l \in \{1, \ldots, L\}$.

3.1 Inference Graphs for MLPs

The hidden layer activity of an MLP network composed of FC layers can be modeled by a single probabilistic graphical model with an HMM structure, enabling closed-form inference. The model structure is shown in Fig. 2.

The activation vector of the lth FC layer with D^l neurons, $x^l = (x^l[1], .., x^l[D^l]) \in \mathbb{R}^{D^l}$, results from a mixture of K^l hidden states (i.e., clusters) with a discrete hidden variable $h^l \in \{1, ..., K^l\}$ denoting the cluster index. To model the ReLU

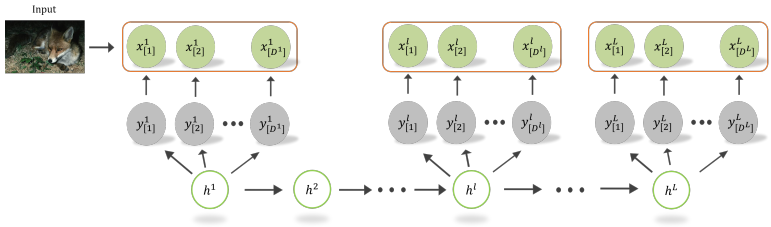

Fig. 2. HMM for MLP networks. Orange rectangles represent post-ReLU layer activity. Activation $x^l[d]$ of neuron d in layer l is generated from a rectified Gaussian density, where $x^l[d]$'s parent, $y^l[d]$, is a Gaussian density before rectification, and h^l is a hidden variable generating the hidden vector of multivariate Gaussians.

operation, each activation $x^l[d]$ is generated from a rectified Gaussian distribution [13]. The conditional probability $P(x^l|h^l)$ is hence assumed to be a rectified multivariate Gaussian distribution with a diagonal covariance matrix. Connections between hidden variables in consecutive layers are modeled by a conditional probability table (CPT) $P(h^l|h^{l-1})$.

Using this generative model, an activity pattern for the network is sampled by three steps. First, a path $(h^1, \ldots, h^L)$ of hidden states is generated according to the transition probabilities $P(h^l = k|h^{l-1} = k') = t^l_{k,k'}$, where $t^l \in \mathbb{R}^{K^l \times K^{l-1}}$ is a learned CPT. For notation simplicity, we define $h^0 = \{\}$, so $P(h^1|h^0)$ is actually $P(h^1)$ parametrized by $P(h^1 = k) = t^1_k$. After path generation, "pre-ReLU" Gaussian vectors $(y^1, \ldots, y^L)$, with $y^l \in \mathbb{R}^{D^l}$, are generated based on the chosen hidden variables. A single variable $y^l[d]$ is formed according to

$$P(y^l[d]|h^l = k) \sim \mathcal{N}(y^l[d]|\mu^l_{d,k}, \sigma^l_{d,k}), \tag{1}$$

where $\mu^l_{d,k}$ and $\sigma^l_{d,k}$ are the mean and standard deviation of the dth element in the kth cluster of layer l. Since the observed activity $x^l[d]$, generated as $x^l[d] = max(y^l[d], 0)$, is a deterministic function of $y^l[d]$, its conditional probability $P(x^l[d]|y^l[d])$ can be written as

$$P(x^l[d]|y^l[d]) = \begin{cases} \delta_{x^l[d]=y^l[d]} & , y^l[d] > 0 \\ \delta_{x^l[d]=0} & , y^l[d] \leq 0 \end{cases}, \tag{2}$$

with $\delta_{(x=c)}$ as the Dirac delta function concentrating the distribution mass at c.

The full likelihood of the model is given by

$$P(X, Y, H|\Theta) = \prod_{l=1}^{L} P(h^l|h^{l-1}) P(y^l|h^l) P(x^l|y^l), \tag{3}$$

where Y, H, X are tuples representing their respective variables across all layers (e.g., $H = \{h_l\}_{l=1}^L$), and the equation's components are stated above.

Training algorithm: In [14], the EM formulation was suggested for training a mixture of rectified Gaussians. We extended this idea to the HMM formulation

in an online setting. Following [15], the online EM algorithm tracks the sufficient statistics using running averages, and updates the model parameters using these statistics. Explicit update formulas in terms of the tracked sufficient statistics are presented in the Supplementary Material.

3.2 Inference Graphs for CNNs

Layer dictionaries: In a CNN, the activation output of the lth convolutional layer is a tensor $X^l \in R^{H^l \times W^l \times D^l}$, where H^l, W^l, and D^l correspond to the height, width, and number of maps, respectively. We consider the activation tensor as consisting of $H^l \times W^l$ spatial column examples, $x_p^l \in R^{D^l}$, located at $p = (i, j) \in \{\{1, \ldots, H^l\} \times \{1, \ldots W^l\}\}$, and wish to model each such location as containing a separate visual word from a dictionary shared by all locations.

The number of hidden variables (one per location) is much larger than that in an FC layer (where a single hidden variable per layer was used), and their connectivity pattern across layers is dense, leading to a graphical model with high induced width, but with infeasible exact inference. Hence, we turn to simpler model and training techniques that are scalable to the size and complexity of CNNs. In this model, spatial column x_p^l is described as arising from a GMM of K^l clusters, regarded as visual words forming the layer dictionary. Using a training image set $S_T = \{(I_n, y_n)\}_{n=1}^{N_T}$, the GMM is trained independently for each layer of interest. While each location p in layer l has a separate hidden random variable, h_p^l, the GMM parameters are shared across all the spatial locations of that layer. After model training, the activity tensor of layer l for an example I can be mapped into a tensor $P \in R^{H^l \times W^l \times K^l}$, holding $P(h_p^l(I) = k)$. With slight abuse of notation, we say that $h_p^l(I) = k^*$ (an activation column of image I in position p is assigned to cluster k^*) iff $k^* = \text{argmax}_k P(h_p^l(I) = k)$. Accordingly, visual word k in layer l is the cluster $C_k^l = \{(I, p), I \in S_T : h_p^l(I) = k\}$ containing activations over all positions for all images in the training set S_T, where cluster k has the highest $P(h_p^l(I) = k)$.

When the CNN also contains global layers, these can be modeled using a GMM trained on the layer's activity vectors. This can be regarded as a degenerate case of convolutional layer modeling, where the number of spatial locations is one. Specifically, the output layer of the network, X^L, containing the M class of predicted probabilities, is modeled using a GMM of M components. This GMM is not trained, and instead is fixed such that $\mu_{d,m} = 1$ for $d = m$ and 0 otherwise, with a constant standard deviation of $\sigma_{d,m} = 0.1$. In this setting, cluster m of the output layer contains images that the network predicts to be of class m.

Probabilistic connections between layer dictionaries: Transition probabilities between visual words in consecutive layers are modeled a-posteriori. For two consecutive modeled layers l and l' ($l' < l$), the receptive field $R(p)$ of location p in layer l is defined as the set of locations $\{q = p + o : o \in O\}$ in layer l' used in the computation of x_p^l. O is a set of $\{(\Delta x, \Delta y)\}$ integer offsets. Using a validation sample $S_V = \{I_n\}_{n=1}^{N_V}$, we compute the co-occurrence matrix

$N \in M^{K^l \times K^{l'}}$ between the visual words contained in the dictionaries of layers l and l',

$$N(k,k') = \left|\{(I_n,p,q) : h_p^l(I_n) = k, h_q^{l'}(I_n) = k', q \in R(p)\}\right|. \quad (4)$$

Using N, we can obtain the following first and second order statistics:

$$\hat{P}(h^l = k) = \frac{\sum_j N(k,j)}{\sum_{i,j} N(i,j)} \quad (5)$$

$$\hat{P}\left(h_q^{l'} = k' | h_p^l = k, q \in R(p)\right) = \frac{N(k,k')}{\sum_j N(k,j)} = \quad (6)$$

$$\frac{\left|\left\{(I_n,p,q) : h_p^l(I_n) = k, h_q^{l'}(I_n) = k', q \in R(p)\right\}\right|}{|O| \cdot \left|\left\{(I_n,p) : h_p^l(I_n) = k\right\}\right|} = \frac{1}{|O|} \sum_{o \in O} \hat{P}(h_{p+o}^{l'} = k' | h_p^l = k).$$

The transition probabilities as defined above are abbreviated in the following discussion to $\hat{P}(h^{l'} = k' | h^l = k)$. These probabilities are averaged over specific positions in the receptive field, since modeling of position-specific transition probabilities separately would lead to proliferation in the parameter number.

Training algorithm: The GMM parameters Θ^l of layer l are trained by associating a GMM layer to each modeled layer of the network. Since we do not wish to alter the network's behavior, the GMM gradients do not propagate towards lower layers of the network. We considered two optimization approaches for training Θ^l:

(1) *Generative loss* — The optimization objective is to minimize the negative log-likelihood function:

$$\mathcal{L}_G(X^l(I_n), \Theta^l) = -\sum_{p \in (H^l \times W^l)} \log \sum_{k=1}^{K^l} \pi_k^l G(x_p^l(I_n) | \mu_k^l, \Sigma_k^l), \quad (7)$$

where G is the Gaussian distribution function, and π_k^l is the mixture probability of the k'th component in layer l.

(2) *Discriminative loss* — The probability tensor P is summarized into a histogram of visual words $Hist^l(X^l(I_n)) \in R^{K^l}$ using a global pooling operation. A linear classifier $\mathcal{W} \cdot Hist^l(X^l(I_n))$ is formed and optimized by minimizing a cross entropy loss, where $\mathcal{W}$ is the classifier weights vector,

$$\mathcal{L}_D(X^l(I_n), \Theta^l) = -\log P(\hat{y_n} = y_n | \mathcal{W} \cdot Hist^l(X^l(I_n), \Theta^l)), \quad (8)$$

and $\hat{y_n}$ is the predicted output after a softmax transformation.

Comparison between these two approaches is given in Section 4.3.

For ImageNet-scale networks, full modeling of the entire network at once may require thousands of visual words per layer. Training such large dictionaries is not feasible with current GPU memory limitations (12GB for a TitanX). Our solution is to train a class-specific model, explaining network behavior for a specific class m and its "neighboring" classes, i.e., all classes erroneously predicted by the network for images of class m. The set of neighboring classes is chosen based on the network's confusion matrix computed on the validation set. The model is trained on all training images of class m and its neighbors.

3.3 Maximum-Likelihood-Based Node Selection

Consider a graph in which column activity clusters (i.e., visual words) $\{C_k^l\}_{l=1,k=1}^{L,K^l}$ are the nodes, and transition probabilities between clusters of consecutive layers quantify edges between the nodes. Typically, this graph contains thousands of nodes and, thus, is not feasible for human interpretation. However, specific subgraphs may have high explanatory value. Specifically, nodes (clusters) of the final layer C_k^L in this graph represent images for which the network predicted a class k. To understand this decision, we evaluate clusters in the previous layer $C_{k'}^{L-1}$ using a score based on the transition probabilities $P(h^L = k|h^{L-1} = k')$. The step of finding such a set of "explanatory" clusters in layer $L-1$ is repeated to lower layers. Below, we develop an iterative algorithm that using a validation subset of images $\Omega = \{I_n\}_{n=1}^N$ outputs a subgraph of the nodes that most "explain" the network decisions on Ω, where "explanation" is defined in the maximum-likelihood sense. We first explain node selection for a single visual word in a single image, and then extend this notion to a full algorithm operating on multiple visual words and images.

Explaining a single visual word: Consider an instance of a single visual word $h_p^l(I) = s$, derived from a column activity location p in layer l for image I. For this visual word, we look for the visual words in $R(p)$ most contributing to its likelihood (omitting the image notation I in $h_p^l(I)$ for brevity):

$$P\left(h_p^l = s \middle| \left\{h_q^{l'} : q \in R(p)\right\}\right) = \frac{P\left(\left\{h_q^{l'} : q \in R(p)\right\} \middle| h_p^l = s\right) \cdot P(h_p^l = s)}{P\left(\left\{h_q^{l'} : q \in R(p)\right\}\right)} \tag{9}$$

$$\approx \frac{\prod_{q\in R(p)} P(h_q^{l'}|h_p^l = s) \cdot P(h_p^l = s)}{\prod_{q\in R(p)} P(h_q^{l'})}.$$

In the last step, two simplifying assumptions were made: conditional independence over locations in the receptive field (nominator) and independence of locations (denominator). Taking the logarithm, we decompose the expression:

$$\underbrace{\log P(h_p^l = s)}_{\text{constant } A} + \textstyle\sum_{q\in R(p)} \log \frac{P(h_q^{l'}|h_p^l=s)}{P(h_q^{l'})} = \tag{10}$$

$$A + \textstyle\sum_{t=1,\ldots,K^{l'}} \left|\left\{q : h_q^{l'} = t, q \in R(p)\right\}\right| \log \frac{P\left(h_q^{l'}=t|h_p^l=s, q\in R(p)\right)}{P(h_q^{l'}=t)}.$$

Denote $Q_t^{l'}(I,p) = \left|\left\{q : h_q^{l'} = t, q \in R(p)\right\}\right|$ as the number of times visual word t appears in the receptive field of location p. We look for a subset of words $T \subset \{1,\ldots,K^{l'}\}$, which contribute the most to the likelihood of $h_p^l = s$. Thus, the problem we solve is

$$\max_{\substack{T \\ |T|=Z}} \left\{ \sum_{t\in T} Q_t^{l'}(I,p) \log \frac{P\left(h_q^{l'} = t|h_p^l = s, q \in R(p)\right)}{P(h_q^{l'} = t)} \right\}, \tag{11}$$

Algorithm 1 Inference graph building

Input: CNN CN, a set Ω of validation images predicted by CN to class m, Z nodes per layer, network model $\{\Theta^l, \hat{P}(h^l = k), \hat{P}(h^l = k|h^{l'} = k')\}_{l=1,k=1,k'=1}^{L,K^l,K^{l'}}$

Output: An inference graph $G = (N, E)$, where N and E hold clusters (nodes) and their weighted connections (edges) in the graph, respectively

Initialization: Push Ω through the network model to get $\{Q_{t,s}^l(\Omega)\}_{l=1}^{L-1}$ (Eq. 15) and clusters $\{C_i^l\}_{l=1,i=1}^{L,K^l}$. Set $S = m$, $N = C_m^L$, and $E = \emptyset$

For $l = L-1, \ldots, 1$

  For $t = 1, \ldots, K^l$, compute $S^l(\Omega, S, t)$ (Eq. 16)

  Choose $(z_1^l, \ldots, z_Z^l)$ to be the Z clusters indices with the largest scores $S^l(\Omega, S, t)$

  Set $S = (z_1^l, \ldots, z_Z^l)$ and $e_{i,j}^l = S^l(\Omega, z_i^{l+1}, z_j^l)$, $\forall i, j = 1, \ldots Z$

  Set $N = N \cup \{C_{z_i^l}^l\}_{i=1}^{Z}$ and $E = E \cup \{e_{i,j}^l\}_{i=1,j=1}^{Z,Z}$ // nodes and edges update

by choosing the first Z words for which the explanatory score term

$$S^{l'}(I, s, t) = Q_t^{l'}(I, p) \log \frac{P\big(h_q^{l'} = t | h_p^l = s, q \in R(p)\big)}{P(h_q^{l'} = t)} \tag{12}$$

is the highest. Intuitively, the score of the tth visual word is the product of two terms, $Q_t^{l'}(I, p)$, which measures the word frequency in the receptive field, and $\log \frac{P\left(h_q^{l'}=t|h_p^l=s,q\in R(p)\right)}{P(h_q^{l'}=t)}$, which measures how likely it is to see word t in the receptive field compared to seeing it in general. To compute the probabilities in the log term of the score, we use the estimations $\hat{P}(h^l = k)$ and $\hat{P}(h^{l'} = k'|h^l = k)$ made using Eqs. 5 and 6, respectively.

Explaining multiple words and images: The optimization problem presented in Eq. 11 can be extended to multiple visual words in multiple images using column position and image independence assumptions. Assume a set of validation images Ω is being analyzed, and a set of words $S \subset \{1, \ldots, K^l\}$ from layer l has to be explained by lower layer words for these images. We would like to maximize the likelihood of the set of all column activities $\{h_p^l(I_n) : h_p^l \in S, I_n \in \Omega\}$, in which a word from S appears. Assuming column position independence, this likelihood decomposes into terms similar to Eq. 9:

$$\log P\Big(\big\{h_p^l(I_n) : h_p^l(I_n) \in S, I_n \in \Omega\big\} \big| \big\{h_q^{l'}(I_n) : I_n \in \Omega\big\}\Big) = \tag{13}$$

$$\sum_{n=1}^{N} \sum_{s \in S} \sum_{\{p: h_p^l(I_n) = s\}} \log P\big(h_p^l(I_n) \big| h_q^{l'}(I_n), q \in R(p)\big).$$

Repeating the derivation also given in Eqs. 9, 10, and 11 for this expression, we get a similar optimization problem,

$$\max_{\substack{T \\ |T|=Z}} \left\{ \sum_{t \in T} \sum_{s \in S} Q_{t,s}^{l'}(\Omega) \log \frac{P\big(h_q^{l'} = t | h_p^l = s, q \in R(p)\big)}{P(h_q^{l'} = t)} \right\}, \tag{14}$$

where $Q^{l'}_{t,s}(\Omega)$ is the aggregation of $Q^{l'}_t(I,p)$ over multiple positions and images

$$Q^{l'}_{t,s}(\Omega) = \sum_{n=1}^{N} \sum_{\{p:h^l_p(I_n)=s\}} Q^{l'}_t(I_n,p). \tag{15}$$

That is, $Q^{l'}_{t,s}(\Omega)$ is the number of occurrences of word s with word t in its receptive field in all the images in Ω. The solution is given by choosing the Z words in layer l' for which the score

$$S^{l'}(\Omega,S,t) = \sum_{s\in S} Q^{l'}_{t,s}(\Omega) \log \frac{P\big(h^{l'}_q = t | h^l_p = s, q \in R(p)\big)}{P(h^{l'}_q = t)} \tag{16}$$

is maximized. The inference graph is generated by going over the layers from the top layer—for which the decision has to be explained—and downwards, selecting the explaining nodes using the score of Eq. 16. See Algorithm 1 for details.

4 Results

Fully connected networks were trained on the MNIST [16] and CIFAR10 [17] datasets, containing 10 classes each. The networks include six layers with the first five containing $1,000$ neurons each. Based on a preliminary evaluation, the number of visual words K^l was set at 40 for all layers.

CNN models included ResNet20 [2] trained on CIFAR10, and VGG-16 [18] and ResNet50 [2] trained on the ILSVRC 2012 dataset [19]. For ResNet20, the output of add-layers after each skip connection were modeled, as these outputs are expected to contain aggregated information. For ResNet50, the output of add-layers 1, 3, 7, 13, and 16 were modeled. For VGG-16, the first convolutional layers at each block were modeled (five layers in total). The numbers of visual words were set at $60, 100, 200, 450$, and $1,500$ for Layers 1-5, respectively, according to the GPU memory limitation.

In all experiments and modeled layers, the GMM's mean parameters were initialized using K^l randomly selected examples. The variance parameters were initialized as the variances computed from $1,000$ random examples. Prior probabilities were uniformly initialized to be $\frac{1}{K^l}$.

4.1 MLP Inference Path

In MLP networks, the entire layer activity is assigned to a single cluster. To visualize such a cluster, we consider it as a "decision junction", where a decision regarding the consecutive layer cluster is made. For this visualization, the activity vectors in the C^l_k cluster are labeled according to their consecutive layer clusters, forming subclusters for this cluster. We use linear discriminant analysis (LDA) [20] to find a 2D projection of the activity vectors that maximize the

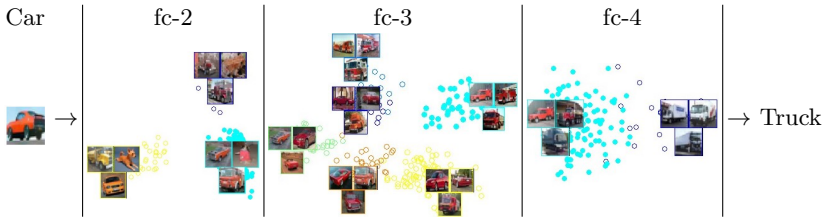

Fig. 3. MLP inference path. Three main decision junctions of a misclassified example in a 6-layer network. Subclusters are visualized by the three most representative examples. Points from the subcluster chosen by this example are marked by full circles.

separation of the examples with respect to their subcluster labels. Each subcluster is visualized using the three examples with the minimal l_2 distance to the subcluster center.

The inference path for an example I is defined to be the maximum a-posteriori (MAP) cluster sequence, i.e., $H = (h^1, ..., h^L)$, satisfying

$$\max_{h^1,...,h^L} \log P(h^1, ..., h^L | X(I)), \tag{17}$$

where H can be found using the Viterbi algorithm [21].

Such inference paths are useful for error diagnosis. In Fig. 3, a partial path containing three decision junctions of an erroneous "car" example in the CIFAR10 network is presented. It can be seen that the example's likely "decisions" in layer fc-3 leads to car and truck clusters in the consecutive layer. At this point, due to its unconventional rear appearance, resembling a truck front, this car example was wrongly associated with a "truck" subcluster, an association that remained until the classification layer.

4.2 Cluster Similarity Across Layers

A plausible assumption about deep network representations is that early representations are input dominated, while representations in late layers are more class-related. This phenomena can be observed in our framework by considering how distance between clusters changes as a function of layer index. In Fig. 4 (left), cluster similarity matrices are presented. Each matrix shows Euclidean distances between centers of the clusters from a single layer. Clusters are ordered by their dominant class index, defined as the class whose examples are the most frequent in the cluster. For an MLP network, presented in Fig. 4 (top left), the progression toward class-related representation is evident from the emerging block structure in Layers 3-5, indicating increasing similarity between clusters representing the same class.

In contrast, as seen in Fig. 4 (bottom left), CNN clusters (representing column activities) stay local and diverse, even at the uppermost layers where their receptive fields cover the entire input space. This phenomenon is demonstrated

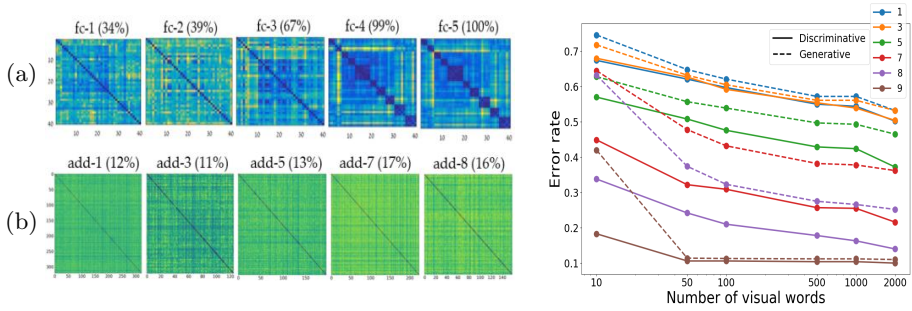

Fig. 4. Left: Cluster similarity matrices for increasing layer indices in a 6-layer MLP (a) and ResNet20 (b) both trained on CIFAR10. The average percentage of dominant class examples (across clusters) is stated above each matrix. **Right:** Error rates of a linear classifier trained over word histograms taken from six ResNet20 conv-layers (1, 3, 5, 7, 8, and 9) trained with either a generative or discriminative loss (Section 3.2).

by the lack of block structure, as well as the relatively low frequency of the dominant class in a cluster. This indicates that the final CNN classification is based on several class-specific words, which are not similar, and appear simultaneously in different image regions.

4.3 CNN Inference Graphs

Loss and dictionary size. Fig. 4 (right) shows the errors obtained for linear classification using dictionary histograms (Eq. 8), as a function of dictionary size. Graphs are shown for dictionaries obtained using losses $\mathcal{L}_G$ (7) and $\mathcal{L}_D$ (8), for several intermediate convolutional layers of the CIFAR10 network. For all layers, except the final one (indexed 9), larger dictionaries provide better classification. However, for the final layer, dictionaries larger than 50 clusters do not increase accuracy, which approaches the original network error of 0.088. As expected, the discriminative loss $\mathcal{L}_D$ leads to smaller errors than the generatively-optimized loss. Therefore, all models presented below were trained with the $\mathcal{L}_D$ loss.

Class inference graph. An example of a class inference graph for the class "pineapple" in VGG-16 is presented in Fig. 5. The graph shows that the most influential words in the top convolution layer can be roughly characterized as "grassy head" (left word), "pineapple body" (middle), and "body textured with (lower) round edge". The origin of these words can be traced back to lower layers. For example, "grassy head" is composed of words capturing mostly "vegetation" in the layer below, which are in turn generated from words describing green texture and multiple diagonal lines (middle layer, left and middle words). Similarly, the origin of "pineapple body" can be traced back to texture words in lower layers and until the green edge filters in Layer 1.

Image inference graphs. Fig. 6 shows an image inference graph for a pineapple image wrongly classified to the "swing" class. Using the inference graph, we can analyze the dominant (representative) visual words that have

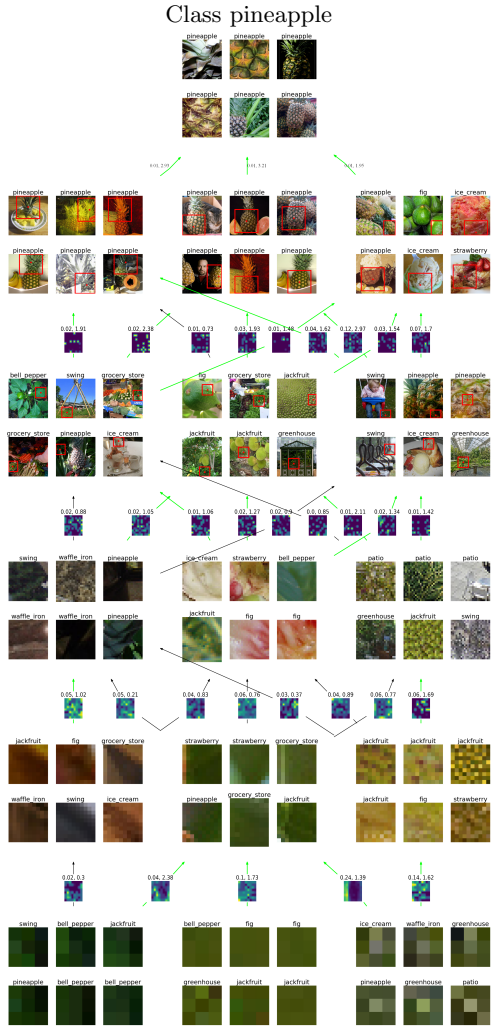

Fig. 5. Pineapple inference graph. The graph is generated by training a model on "pineapple" class and its neighboring classes. The top node is a visual word of the output layer, representing the predicted class "pineapple". The lower levels in the graph show the three most influential words in preceding modeled layers. Visual words are manifested by the six representative examples for which $P(h^l = k|x_p^l)$ is the highest. For the two highest layers, examples are presented by showing the example image with a rectangle highlighting the receptive field of the word's location. For lower layers, the receptive field patches themselves are shown. Images are annotated by their true label. Arrows are shown when the log-ratio term is positive, colored green for significant connections in which the term is higher than 1. They are annotated by the frequency of the lower word in the receptive field (left) and the log ratio (right) (the two components of the score in Eq. 16). In addition, for each arrow, a heatmap is shown indicating the frequent locations of the lower-level visual words in the receptive field of the higher-level word. The figure is best inspected by zooming in on clusters of interest.

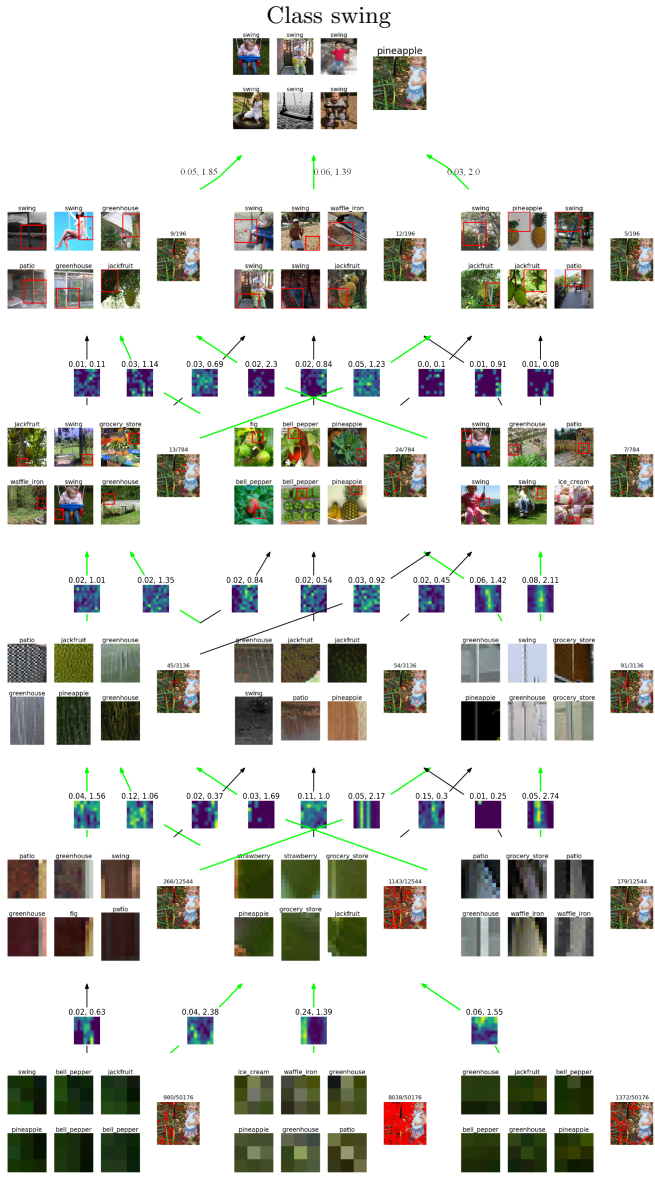

Fig. 6. An image inference graph of an erroneous image. An image inference graph for a pineapple image wrongly classified to the class "swing". The model is generated using "pineapple" and its neighboring classes (same as in Fig 5), where the neighbor class "swing" is included. The graph is generated by applying the node selection algorithm (Section 3.3) to a set Ω containing this single erroneous image. The analyzed image is shown on the right side of each cluster node, with red dots marking spatial locations assigned to the cluster. The fraction of spatial examples belonging to the cluster (in this image) appears in the title.

Fig. 7. Subgraph of the same image as in Fig. 6 classified by ResNet50. A visual word in layer add-13 (left), leading to a visual word in layer add-16 (middle), and to correct classification of the pineapple (right).

led to this erroneous classification: **(1)** Top layer (Layer 5): The visual words voting for the swing class focus on strong vertical rope-like shapes (left), sand texture (middle), and grass/foliage texture (right). **(2)** Layers 4 and 3: The strong vertical line (left of Layer 5) originates from a similar visual word of Layer 4 (right), and this in turn depends strongly on the "isolated-vertical-line" word in Layer 3 (right). The foliage word (right in Layer 5) mainly originates from the vegetation left word in Layer 4, which in turn heavily depends on the two green "vertical-stripe-structure" words in Layer 3 (left and middle). **(3)** Layers 2 and 1: the main explanatory words in Layer 2 are green and dark vertical edges and lines, which are combined to construct the "isolated-vertical-line" word in Layer 3 (right) and the "vertical-stripe-structure" word in Layer 3 (left). These in turn are created from earlier "green-edge" words in Layer 1.

In Fig. 7, we show a partial inference graph of the same "pineapple" image wrongly classified to class "swing" by VGG-16 (Fig. 6), which is successfully classified by ResNet50. As can be seen in the top layer of the graph (add-16, middle), Resnet50 successfully detects the pineapple location in the image, where both visual words presented contain strong "pineapple" features. Additional examples of class and image inference graphs are given in the Supplementary Material.

5 Conclusion

We introduced a new approach for interpreting hidden layers activity of deep neural networks by learning dictionaries of activity clusters and transition probabilities between clusters of consecutive modeled layers. We formalized a maximum-likelihood criterion for mining clusters relevant for network prediction, which enable building explanatory inference graphs of manageable size. Inference graphs can be constructed for an entire classes, to understand the general network reasoning for this class, or for specific images, for which error analysis may specifically be sought. The tools developed can be used to verify the soundness of the network reasoning and to understand its hidden inference mechanisms, or conversely to reveal network weaknesses. The code for the framework algorithms is available at https://bit.ly/38pcatH.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. (2012) 1097–1105
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 770–778
3. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 4700–4708
4. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. University of Montreal **1341**(3) (2009) 1
5. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
6. Yosinski, J., Clune, J., Nguyen, A.M., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. CoRR **abs/1506.06579** (2015)
7. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2921–2929
8. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision, Springer (2014) 818–833
9. Liao, R., Schwing, A., Zemel, R., Urtasun, R.: Learning deep parsimonious representations. In: Advances in Neural Information Processing Systems. (2016) 5076–5084
10. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.K.: This looks like that: deep learning for interpretable image recognition. In: Advances in Neural Information Processing Systems. (2019) 8928–8939
11. Liu, M., Shi, J., Li, Z., Li, C., Zhu, J., Liu, S.: Towards better analysis of deep convolutional neural networks. IEEE Transactions on Visualization and Computer Graphics **23**(1) (2016) 91–100
12. Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., Mordvintsev, A.: The building blocks of interpretability. Distill **3**(3) (2018) e10
13. Socci, N.D., Lee, D.D., Seung, H.S.: The rectified Gaussian distribution. In: Advances in Neural Information Processing Systems. (1998) 350–356
14. Lee, G., Scott, C.: EM algorithms for multivariate Gaussian mixture models with truncated and censored data. Computational Statistics & Data Analysis **56**(9) (2012)
15. Cappé, O., Moulines, E.: On-line Expectation–Maximization algorithm for latent data models. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **71**(3) (2009) 593–613
16. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist **2** (2010)
17. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009)
18. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
19. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large

scale visual recognition challenge. International Journal of Computer Vision **115**(3) (2015) 211–252
20. Fukunaga, K.: Introduction to statistical pattern recognition. Academic Press (1990)
21. Forney, G.D.: The Viterbi algorithm. Proceedings of the IEEE **61**(3) (1973) 268–278