# Inference Graphs Based on Hidden Layers Activity Interpretation

Yael Konforti†, Alon Shpigler†, Boaz Lerner, Aharon Bar-Hillel

**Abstract**—Convolutional neural networks (CNNs) have achieved superior accuracy in many visual-related tasks. However, the inference process through a CNN's intermediate layers is opaque, making it difficult to interpret such networks or develop trust in their operation. We propose to model the network's hidden layer 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 to identify paths of inference. For fully connected networks, the entire layer activity is clustered, and the resulting model is a hidden Markov model. For convolutional layers, spatial columns of activity are clustered, and a maximum likelihood model is developed for mining an explanatory inference graph. The graph describes the hierarchy of activity clusters most relevant for network prediction. We show that such inference graphs are useful for understanding the general inference process of a class, as well as explaining the (correct or incorrect) decisions the network makes about specific images. In addition, the models provide interesting observations regarding hidden layer activity in general, including the concentration of memorization in a single middle layer in fully connected networks, and a highly local nature of column activities in the top CNN layers.

**Index Terms**—Deep Neural Networks, Convolutional Neural Networks, Visualization, Interpretable AI, Explainable AI

## 1 Introduction

Thanks to their impressive performance, convolutional neural networks (CNNs) are the leading architecture for tasks in computer vision [2], [3], [4]. However, current deep-learning methods suffer from poor interpretability of the inference process conducted by their hidden layers. Due to their end-to-end training and complex architecture, the reasoning behind 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. Due to the problem's importance, a growing number of visualization techniques have been developed to promote understanding of the network's internal decision-making process. Specifically, 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 allow us to develop more trust in CNN decisions.

As we see it, enabling human understanding of the deep-network inference process requires facing several challenges. The first is transforming a distributed high dimensional representation into a discrete representation amendable to human reasoning. Deep networks operate through a series of distributed layer representations, manifested by a single activation vector in fully connected (FC) layers and a collection of spatial activation vectors (i.e., columns) in convolutional layers. Human language, however, is made up of discrete symbols, i.e., words, having meaning grounded by their reference in the world of objects 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?

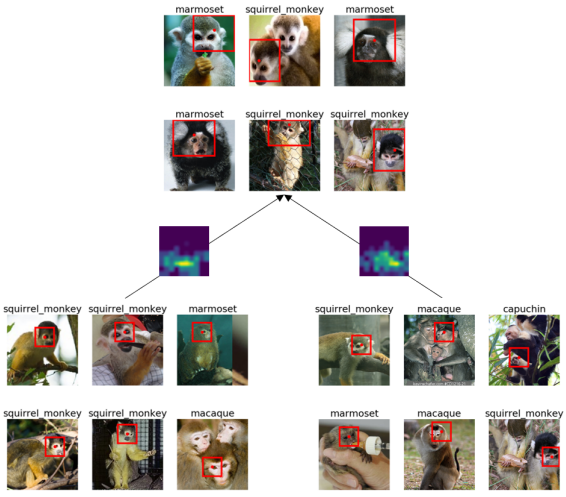



Fig. 1. **Visual words and their connections.** An example of two visual words (bottom) creating a more complex visual word (top) in an upper network layer. The bottom-left visual word represents an "eye" located right to the receptive field center. The bottom-right visual word represents the "head-face-border" of the monkey (note the red dots that represent such borders), where its eye is located to the left of the receptive field center. The aggregation of both yields the higher level visual word (top), representing most of a monkey face. A heat map annotating an arrow shows where a lower word is active in the receptive field of a upper word (see Section 3.4.3 for details).


- *Y. Konforti, A. Shpigler, B. Lerner, and A. Bar-Hillel are with the Ben-Gurion University of the Negev. E-mail: {yaelkonf, alonshp}@post.bgu.ac.il; {boaz, barhille}@bgu.ac.il.*
- †*Equal contribution.*



- *Preliminary version was published in ECCV 2020 [1].*
- *Code is released at github.com/yaelkon/GMM-CNN.*

*Manuscript received XXXX YY, ZZZZ; revised XXXX YY, ZZZZ.*

A second challenge, closely related to the first, is choosing a limited set of explanatory visual words. Given the richness of a distributed representation and the sheer size of modern networks, discretizing an internal representation may require thousands of visual words. Hence, selecting a relevant subset of visual words and connections for a specific analysis task (e.g., class-specific or image-specific analysis) is inherent to this endeavor. Finally, a third challenge is the development of a visualization system that enables understanding of the visual words and their interrelations, while taking into account the human perceptual and cognitive limitations and display capacity [5].

To address these challenges, we suggest describing the inference process of a neural network with probabilistic models. We model activity vectors in each layer as arising from a multivariate Gaussian mixture model (GMM). Layer activity in 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 multi layer 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. Full exact inference is infeasible due to the high induced width of the resulting graphical model. However, dependencies among visual words in consecutive layers can still be described using conditional probability tables, and these can be used to define weighted graphs and mine subgraphs of high explanatory value.

Given a selected subset of images to be explained (either a specific image or images of the entire class), the decision process of the network can be described using an inference graph, representing the visual words used to explain the image in different hidden layers and their probabilistic connections. As the full graph may contain thousands of visual words in all network layers, a useful explanation has to find informative subgraphs containing the most explanatory words for clarifying the network decision. We suggest a maximum likelihood (ML) based approach and an iterative algorithm for finding useful subgraphs. Visual words of the top network layer are first selected to best explain the network's decision (in a maximum likelihood sense), and words in lower layers are iteratively selected to best explain previously selected words.

For MLP networks, each node can be visualized as a "decision junction", where the network decides between multiple alternatives of consecutive layer nodes. The chain of decision junctions establishes an inference path, describing the inference process for a specific image. For CNNs, a spatial grid of visual words describes the image at each layer, and the inference process is described using a subgraph mined as mentioned above. In both cases, visual words are visualized by their most typical examples, projected onto the image space in their respective receptive field regions.

Using these models, we contribute to current knowledge by presenting:

- **A class-specific inference graph:** The class inference graph characterizes the inference process for a single class, providing a succinct summary of the inference process toward this class, as it progresses through the network layers. The connections between visual words discovered by our model in consecutive layers provide clear insights into the feature aggregation process for this class (see Fig. 1 for an example). It enables understanding how relevant low-level features (e.g., edges, textures, color, etc.) accumulate into mid-level features (small object parts like eyes) and later into conceptual high-level features that describe the class.
- **An image-specific inference graph:** The inference graph for a specific image focuses on the visual words most contributing to a class decision on this image. Such a graph is highly useful as a debugging tool to analyze network failures because it enables finding the main features leading to a wrong class decision on an image, the layer at which they appear, and their influence on consecutive spatial locations and layers.
- **Differences in layer activity behavior between MLP networks and CNNs:** Our inference graph demonstrates significant differences in activity behavior between the two network types as inference progresses through the network. For MLP networks, visual words gradually converge, from multiple input-related words to unique class-related words. In upper layers, each class is represented with an activity population highly concentrated around a single pattern. In contrast, CNN behavior remains local and diverse even at the uppermost layers, with each class represented by a combination of distinct multiple words.
- **Overfitting capacity of a single layer:** Following Zhang et al. [6], we analyze a case of extreme overfit by learning with random labels. Our model discovers that for MLP networks forced to such extreme overfit, the overfitting transformation is concentrated in a single (the middle) hidden layer. In such a case, our suggested tools enable, for the first time as far as we are aware, characterization "where" in the network overfit occurs.

## 2 Related Work

**Image/class-specific visualization.** Several techniques have been suggested to explain a network's behavior with respect to a specific input image or a class of interest. With activation maximization [7], a learned artificial input image, which maximizes the score of a given hidden unit, was visualized. Optimization was done using gradient ascent to the image space. In [8], an image of the pixels with significant gradients with respect to an output class neuron was produced (a "saliency" map), highlighting the features that most affect the class score. The technique was later applied to intermediate feature maps in convolutional layers [9], optimized using deconvolution layers. The authors showed how lower level neurons correspond to semantic parts, with complexity growing with layer index. Visualization using deconvolution was later enhanced by [10]. Another visualization technique for creating class-related activation maps was proposed by [11]. This approach aggregates maps of the final convolutional layer with a class-specific weighted sum. The weights used are those connecting the globally pooled maps to the output class neurons. Heat maps produced by this technique are shown to have remarkable object localization capabilities.

**Identifying semantics in hidden layers.** Recent work has

tried to measure the semantic content in hidden layers. Bau et al. [12] defined six types of semantic patterns (colors, textures, materials, parts, objects, and scenes) identified by CNNs, and labeled image pixels accordingly. They used this to analyze what type of patterns are associated with activations in intermediate convolutional layers. A later work [13] aimed to increase interpretability of high convolutional layers by enforcing the filters to represent object parts. This was done by requiring that filters fire only for specific classes and specific compact regions. Networks trained indeed had more interpretable filters, but often at the cost of some accuracy decrease.

**Simplifying network representations.** Several works looked for categorizing features through clustering [14], [15], [16]. Liao et al. [14] proposed a unified framework for enforcing clustering during network training, including clustering of samples and spatial columns as specific cases. They show that clustering enforcement may improve generalization and interpretability. Another approach for training interpretable networks was introduced by Chen et al. [15]. The spatial columns of the topmost convolutional layer were encouraged to represent part prototypes of specific classes. Each such prototype is equated with a a spatial region of a specific image, which is used to visualize it. Loss terms are added to enforce that class inference will be mediated through prototype existence, and that the prototypes will be class-specific. Yang et al. [16] introduced a recurrent procedure for joint unsupervised learning of deep representations and image clusters. The joint procedure provides improved image clustering and some generalization of the representation across data sets. In contrast to all these approaches which use clustering ideas to change the network model and its training procedure, our approach is applicable to any network, as we do not alter an existing model but use our clustering-based model distinctively only for network interpretation.

**Modeling relationships between consecutive layer representations.** In CNNVis [17], neurons in each layer are clustered to form groups having similar activity patterns. For the clustering, a neuron is described using a $C$-dimensional vector of its average activity on each class $1,..,C$. A graph between neuron clusters of subsequent layers is then formed based on the average weight strengths over the cluster's neurons. While this method clusters neurons, in our approach, spatial activity vectors (neuronal columns) are clustered across examples. Olah et al. [18] proposed a tool for visualizing the network path for a single image. They decomposed each layer's activations into neuron groups using matrix factorization, and connected groups from consecutive layers into a graph structure similar to [17].

## 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 $l \in \{1,\dots,L\}$.

### 3.1 Inference Graphs for MLPs

A network composed of FC layers can be modeled by a single probabilistic graphical model based on the following assumptions: (a) The activity of a layer can be modeled by a single mixture model. (b) Conditional independence holds between the activity of layer $l$ and activities of layers preceding $l-1$ given the activity of layer $l-1$. (c) Layer activity is generated by a rectified normal distribution [19], censored at zero according to ReLU operation. For such a network, the activity of hidden layers is modeled by an HMM structure, enabling closed-form inference. The model structure is shown in Fig. 2.

For the $l$th FC layer with $D^l$ neurons, denote the activation vector as $x^l = (x^l[1],..,x^l[D_l]) \in \mathbb{R}^{D^l}$. The distribution of $x^l$ is modeled using 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 operation, each neuron activation $x^l[d]$ is generated from a rectified Gaussian distribution. 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 $(h^1,\dots,h^L)$ of hidden states is generated according to the transition probabilities

$$P(h^l = k|h^{l-1} = k') = t^l_{k,k'} \tag{1}$$

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 $(y^1,\dots,y^L)$, 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

$$P(y^l[d]|h^l = k) \sim \mathcal{N}(y^l[d]|\mu^l_{d,k}, \sigma^l_{d,k}), \tag{2}$$

where $\mu^l_{d,k}$ and $\sigma^l_{d,k}$ are the mean and standard deviation of the $d$th element in the $k$th component 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

$$P(x^l[d]|y^l[d]) = \begin{cases} \delta_{x^l[d]=y^l[d]} & , y^l[d] > 0 \\ \delta_{x^l[d]=0} & , y^l[d] \le 0 \end{cases}, \tag{3}$$

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

The full likelihood of the model is given by

$$P(X,Y,H|\Theta) = \prod_{l=1}^{L} P(h^l|h^{l-1})P(y^l|h^l)P(x^l|y^l), \tag{4}$$

where $P(h^l|h^{l-1})$, $P(y^l|h^l)$, and $P(x^l|y^l)$ are stated in (1), (2), and (3), respectively. $Y,H,X$ are tuples representing their respective variables across all layers, e.g., $H = \{h_l\}_{l=1}^L$. The set of parameters $\Theta$ learned to optimize the model likelihood is

$$\Theta = \left\{ \{t^l_{k,k'}\}^{L,K^l,K^{l-1}}_{l=1,k=1,k'=1}, \{\mu^l_{d,k}\}^{L,D^l,K^l}_{l=1,d=1,k=1}, \{\sigma^l_{d,k}\}^{L,D^l,K^l}_{l=1,d=1,k=1} \right\}. \tag{5}$$

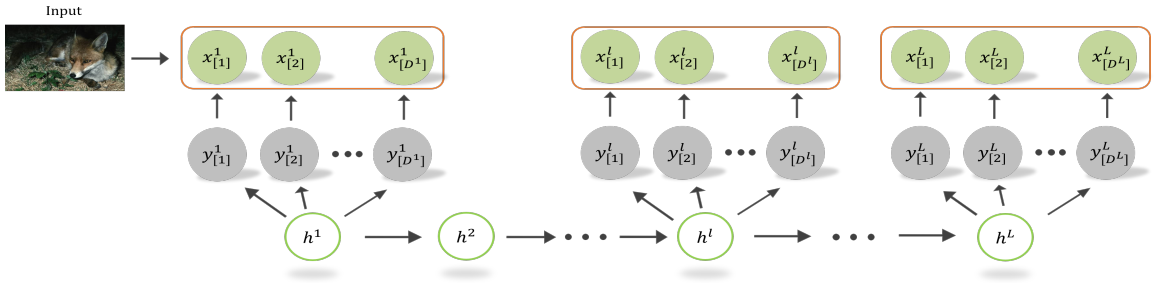



Fig. 2. **A graphical model for MLP networks.** Each orange rectangle is a layer activation vector after the ReLU operation. Activation $x^l[d]$ of neuron $d$ in layer $l$ is assumed to be generated from a rectified Gaussian density, resetting values lower than zero to zero. $y^l[d]$ is the parent of $x^l[d]$, describing the original Gaussian density before it was rectified. $h^l$ is a hidden variable generating the hidden vector of multivariate Gaussians $Y^l$. $h^l$ can be in $K^l$ different states, creating a mixture of multivariate Gaussians for layer $l$.

**Training algorithm:** In [20], the EM formulation was suggested for training a mixture of censored Gaussians. We extended this idea to the HMM formulation in an online setting. Following [21], the online EM algorithm tracks the sufficient statistics using running averages, and updates the model parameters using these statistics. We train with mini-batches, which enables scalable learning for large-scale networks. Training iterates between updating the running averages of the sufficient statistics (an online approximation of the E-step), and updating the parameters based on these averages (the M-step). This procedure is shown [21] to be consistent (i.e., finding a stationary point of the data log likelihood with probability 1) and asymptotically efficient.

**Update equations:** In a batch EM formulation, model parameters are updated based on sample statistics of interest. Each statistic is defined as the sample average $\frac{1}{N}\sum_{i=1}^{N} f(X_i)$ for a function $f(x)$ of interest. In the online setting, for each such function $f(x)$, an online sample estimator is kept, denoted here by $< f(X) >$. Given a batch of examples $\{X_i\}_{i=1}^{B}$ and adaptation parameter $\alpha > 0$, $< f(X) >$ is updated in iteration $q+1$ by

$$< f(X) >_{q+1} = (1-\alpha) < f(X) >_q + \frac{\alpha}{B}\sum_{i=1}^{B} f(X_i) \tag{6}$$

The tracked sufficient statistics are used in the update of the model parameters as follows:

- The transition probability between hidden states $t^l_{k,k'}$ update is given by

$$t^l_{k,k'} = \frac{< P(h^l = k, h^{l-1} = k'|X,\Theta) >}{\sum_{k=1}^{K^l} < P(h^l = k, h^{l-1} = k'|X,\Theta) >}. \tag{7}$$

The average joint distribution of clusters from consecutive layers $< P(h^l = k, h^{l-1} = k'|X,\Theta) >$ is a tracked statistic, computed for each example using the forward-backward algorithm [22]. For the first layer, $t^1_k$ is updated analogously using $t^1_k =< P(h^1 = k|X,\Theta) >$.

- The mean $\mu^l_{d,k}$ for an estimated Gaussian before rectification (dropping the $l$ index for notation convenience) is

$$\mu_{d,k} = \frac{< P(h = k|x[d],\Theta)\cdot \hat{y}[d] >}{< P(h = k|x[d],\Theta) >} \tag{8}$$

with $\hat{y}[d]$ defined by

$$\hat{y}[d] = \begin{cases} x[d], & x[d] > 0 \\ M_1(\mu_{d,k},\sigma_{d,k}), & x[d] = 0 \end{cases}$$

and $M_1(\mu_{d,k},\sigma_{d,k})$, defined below (Eq. 9), is a statistic measured with $\mu_{d,k}$ and $\sigma_{d,k}$ values from the previous iteration. The new mean is a weighted average of all examples' $y$ activities, with each example contributing based on its probability to belong to the cluster. When $x[d] = 0$, the value of the activity prior to the ReLU operation is estimated using the first moment of a rectified Gaussian $M_1(\mu_{d,k},\sigma_{d,k})$, which has a closed form solution [20] for known mean and variance:

$$M_1(\mu,\sigma) = \int_{-\infty}^{0} x\cdot G(x|\mu,\sigma)dx = \mu - \sigma\frac{(G(-\frac{\mu}{\sigma}|0,1))}{(C(-\frac{\mu}{\sigma}|0,1))} \tag{9}$$

where $G(-\frac{\mu}{\sigma}|0,1)$ and $C(-\frac{\mu}{\sigma}|0,1)$ are the density and cumulative values of the normal distribution at $-\frac{\mu}{\sigma}$. Since $\hat{y}[d]$ has two cases, two statistics are tracked for the computation of the nominator in Eq. 8: $< P(h = k|x[d],\Theta)\cdot x[d]1_{x[d]>0} >$ and $< P(h = k|x[d],\Theta)\cdot 1_{x[d]==0} >$.

- The std of an estimated Gaussian density before rectification $\sigma^l_{d,k}$, dropping the $l$ index, is updated by

$$\sigma^2_{d,k} = \frac{< P(h = k|X,\Theta)(\hat{y}[d] - \mu_{d,k})^2 > + R_{d,k}}{< h = k|X,\Theta >}. \tag{10}$$

This formula can be seen as a weighted sum-of-squares and a correction factor

$$\begin{aligned} R_{d,k} =& < P(h = k|x[d],\Theta)\cdot 1_{x[d]==0} > \\ &\cdot\left(M_2(\mu_{d,k},\sigma_{d,k}) - M_1(\mu_{d,k},\sigma_{d,k})^2\right). \end{aligned} \tag{11}$$

The term $M_2(\mu_{d,k},\sigma_{d,k})$ is the second moment of a censored Gaussian distribution, which also has a closed form solution [20]:

$$M_2(\mu,\sigma) = \int_{-\infty}^{0} x^2 G(x|\mu,\sigma)dx = \mu^2 + \sigma^2 - \sigma\mu\frac{(G(-\frac{\mu}{\sigma}|0,1))}{(C(-\frac{\mu}{\sigma}|0,1))}. \tag{12}$$

### 3.2 Inference Graphs for CNN

In a CNN, the activation output of the $l$-th 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, $x^l_p \in R^{D^l}$, located at $p = (i,j) \in \{\{1,\ldots,H^l\}\times\{1,\ldots W^l\}\} = A$, 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 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 [23]. Hence, we turn to simpler model and training techniques, that are scalable to the size and complexity of CNNs. In this model, activities of different layers are modeled independently, each using a Gaussian mixture model, and transition probabilities between clusters in consecutive layers are modeled a-posteriori (i.e., they are not part of the generative model).

**Layer dictionaries**: Similarly to the full activity vector in FC layers, the activity of a 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 $S_T = \{(I_n, y_n)\}_{n=1}^{N_T}$, a GMM is trained independently for each layer of interest. While each location 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, i.e., a single GMM is trained using all spatial columns of layer $l$ and all training images. Training is done using a gradient descent procedure on mini-batches, enabling straightforward GPU implementation. After model training, the activity tensor of layer $l$ for a new example $I$ can be mapped into a tensor $P \in R^{H^l \times W^l \times K^l}$ holding $P(h_p^l(I) = k)$. We say that the visual word $h_p^l(I) = k^*$ (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 ($X^L$) of the network, containing the $M$ class pseudo probabilities (output neurons after softmax), 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, and a constant variance parameter 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 for each two consecutive modeled layers $l$ and $l'$ the co-occurrence matrix $N \in M^{K^l \times K^{l'}}$ between the visual words these two dictionaries contain,

$$N(k,k') = \left|\{(I_n, p, q) : h_p^l(I_n) = k, h_q^{l'}(I_n) = k', q \in R(p)\}\right|. \quad (13)$$

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

$$\hat{P}(h^l = k) = \frac{\sum_j N(k,j)}{\sum_{i,j} N(i,j)} \quad (14)$$

$$\begin{aligned} \hat{P}\left(h_q^{l'}\right) &= k' | h_p^l = k, q \in R(p)) = \frac{N(k,k')}{\sum_j N(k,j)} \quad (15) \\ &= \frac{\left|\left\{(I_n, p, q) : h_p^l(I_n) = k, h_q^{l'}(I_n) = k', q \in R(p)\right\}\right|}{|O| \cdot \left|\left\{(I_n, p) : h_p^l(I_n) = k\right\}\right|} \\ &= \frac{1}{|O|} \sum_{o \in O} \hat{P}(h_{p+o}^{l'} = k' | h_p^l = k) \end{aligned}$$

The transition probabilities as defined above are abbreviated in the following discussion to $\hat{P}(h^{l'} = k' | h^l = k)$. As defined, 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 $\Theta^l$ 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 $\Theta^l$:

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

$$\mathcal{L}_G(X^l(I_n), \Theta^l) = -\sum_{p \in A} \log \sum_{k=1}^{K^l} \pi_k^l G(x_p^l(I_n) | \mu_k^l, \Sigma_k^l) \quad (16)$$

  with

$$\Sigma_k^l = \begin{bmatrix} \sigma_{1,k}^l & & \\ & \ddots & \\ & & \sigma_{D^l,k}^l \end{bmatrix}$$

  where $G$ is the Gaussian distribution function, and $\pi_k^l$ is the mixture probability of the $k$'th component in layer $l$.
- *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 $\mathcal{W} \cdot Hist^l(I_n)$ is formed and optimized by minimizing a cross entropy loss, where $\mathcal{W}$ is the classifier weights vector

$$\mathcal{L}_D(X^l(I_n), \Theta^l, y_n) = -\log P(\hat{y_n} = y_n | \mathcal{W} \cdot Hist^l(X^l(I_n), \Theta^l)). \quad (17)$$

  Here, $y_n$ is the true label of image $I_n$ and $\hat{y_n}$ is the predicted output after a softmax transformation.

Empirical 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 Graph Node Selection Algorithm

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 a suitable iterative algorithm. Given a validation subset of images $\Omega = \{I_n\}_{n=1}^N$, it outputs a subgraph of the nodes that most "explain" the network decisions on $\Omega$, 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.

### 3.3.1 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 $I$. Given this visual word, we look for the visual words in $R(p)$ most contributing to its likelihood, given by (omitting the image notation $I$ in $h_p^l(I)$ for brevity):

$$\begin{aligned} P\left(h_p^l = s\middle|\left\{h_q^{l'} : q \in R(p)\right\}\right) = \\ \frac{P\left(\left\{h_q^{l'} : q \in R(p)\right\}\middle|h_p^l = s\right) \cdot P(h_p^l = s)}{P\left(\left\{h_q^{l'} : q \in R(p)\right\}\right)} \\ \approx \frac{\prod_{q\in R(p)} P(h_q^{l'}|h_p^l = s) \cdot P(h_p^l = s)}{\prod_{q\in R(p)} P(h_q^{l'})}. \end{aligned} \tag{18}$$

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 and see the contribution of visual words to the likelihood:

$$\begin{aligned} \underbrace{\log P(h_p^l = s)}_{\text{constant } A} + \textstyle\sum_{q\in R(p)} \log \frac{P(h_q^{l'}|h_p^l=s)}{P(h_q^{l'})} = \\ A + \textstyle\sum_{t=1,\ldots,K^{l'}} \left|\left\{q : h_q^{l'} = t, q \in R(p)\right\}\right| \log \frac{P\left(h_q^{l'}=t|h_p^l=s,q\in R(p)\right)}{P(h_q^{l'}=t)}. \end{aligned} \tag{19}$$

Denote by $Q_t^{l'}(I,p) = \left|\left\{q : h_q^{l'} = t, q \in R(p)\right\}\right|$ 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

$$\max_{|T|=Z} \left\{\sum_{t\in T} Q_t^{l'}(I,p) \log \frac{P\left(h_q^{l'} = t|h_p^l = s, q \in R(p)\right)}{P(h_q^{l'} = t)}\right\}. \tag{20}$$

The solution is obtained by choosing the first $Z$ words for which the score

$$S^{l'}(I,s,t) = Q_t^{l'}(I,p) \log \frac{P\left(h_q^{l'} = t|h_p^l = s, q \in R(p)\right)}{P(h_q^{l'} = t)} \tag{21}$$

is the highest. Intuitively, the score of visual word $t$ 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 $\hat{P}(h^{l'} = k'|h^l = k)$ given by Eqs. 14 and 15, respectively.

**Algorithm 1** Inference graph building

**Input:** CNN $CN$, a set $\Omega$ of images predicted by $CN$ to class $m$, network model $\{\Theta^l, \hat{P}(h^l = k), \hat{P}(h^l = k|h^{l'} = k')\}_{l=1,k=1,k'=1}^{L,K^l,K^{l'}}$, $Z$ - number of allowed nodes per layer.
**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 $\Omega$ through the network model to get $\{Q_{t,s}^l(\Omega)\}_{l=1}^{L-1}$ (Eq. 24) 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. 25)
  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 $e_{i,j}^l = S^l(\Omega,z_i^{l+1},z_j^l), \quad \forall i,j = 1,\ldots Z$
  Set $N = N \cup \{C_{z_i^l}^l\}_{i=1}^Z$ and $E = E \cup \{e_{i,j}^l\}_{i=1,j=1}^{Z,Z}$

### 3.3.2 Explaining Multiple Words and Images

The optimization problem presented in Eq. 20 can be extended to multiple visual words in multiple images using column position and image independence assumptions. Assume a set of validation images $\Omega$ 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. 18:

$$\begin{aligned} \log P\left(\left\{h_p^l(I_n) : h_p^l(I_n) \in S, I_n \in \Omega\right\}\middle|\left\{h_q^{l'}(I_n) : I_n \in \Omega\right\}\right) = \\ \sum_{n=1}^N \sum_{s\in S} \sum_{\{p:h_p^l(I_n)=s\}} \log P\left(h_p^l(I_n)\middle|h_q^{l'}(I_n), q \in R(p)\right). \end{aligned} \tag{22}$$

Repeating the derivation also given in Eqs. 18, 19, and 20 for this expression, we get a similar optimization problem,

$$\max_{|T|=Z} \left\{\sum_{t\in T} \sum_{s\in S} Q_{t,s}^{l'}(\Omega) \log \frac{P\left(h_q^{l'} = t|h_p^l = s, q \in R(p)\right)}{P(h_q^{l'} = t)}\right\}, \tag{23}$$

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

$$Q_{t,s}^{l'}(\Omega) = \sum_{n=1}^N \sum_{\{p:h_p^l(I_n)=s\}} Q_t^{l'}(I_n,p). \tag{24}$$

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

$$S^{l'}(\Omega,S,t) = \sum_{s\in S} Q_{t,s}^{l'}(\Omega) \log \frac{P\left(h_q^{l'} = t|h_p^l = s, q \in R(p)\right)}{P(h_q^{l'} = t)} \tag{25}$$

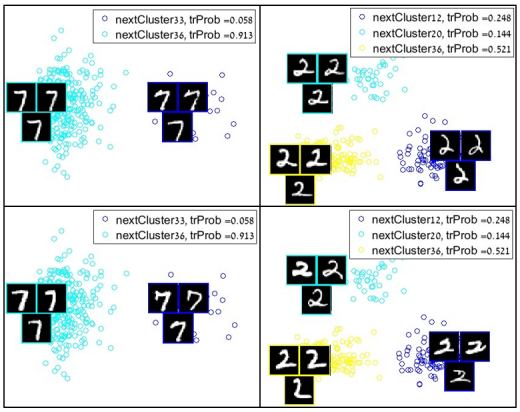



Fig. 3. **Decision junctions**. Two clusters from a six-layer MLP network trained on the MNIST data. The two images on the left visualize a cluster from layer 1, and the images on the right a cluster from layer 2. The top images show sub-cluster representatives chosen by the generative $l_2$ method, and the bottom images by the discriminative $LLR$ method. Each image represents a cluster according to the decision it makes among consecutive cluster alternatives colored in different shades. Transition probabilities to consecutive clusters are stated in the legend. In the left cluster, a decision is made between the curvy and straight "seven" digits. In the right cluster, a decision is made mainly based on the "two" digits' widths and lower part rounds. In both cases, selecting representatives with the $LLR$ criterion provides more extreme examples, enabling easier understanding of the sub-cluster nature.

is maximized. Like the score in Eq. 21, the contribution of word $t$ to the explanation of a single word $s$ is a product of its frequency in the relevant receptive fields and its discriminative value term.

The inference graph is generated by going over the layers backwards, from the top layer, for which the decision has to be explained, and downwards towards the input layer, selecting the explaining nodes using the score of Eq. 25. See Algorithm 1 for details.

## 3.4 Visualization Techniques

We consider visualization at several levels starting from simple visual word to path inference, and relating to the two types of networks.

### 3.4.1 Simple Cluster

We visualize a cluster $C_k^l$ by showing the $m$ examples ($m = 6$) with the highest $P(h_p^l(I) = k)$ across $(I, p) \in S_v$. For FC layers, for each representative example, the full image is shown. For convolutional layers, visual word examples are typically sub-regions of the input image (if the receptive field does not cover the image entirely). For receptive fields larger than 5% of the image size, the visual word occurrence is visualized by drawing the relevant image with a red rectangle around the relevant receptive field (see Fig. 1). When the receptive field is smaller, only the region of the receptive field is shown instead of the entire image.

### 3.4.2 Cluster as a Decision Junction

In MLP networks, the entire layer activity is assigned to a single cluster. We consider such a cluster as a "decision junction", where a decision regarding the consecutive layer cluster is made. For the visualization of such a decision, the activity vectors assigned to $C_k^l$ are labeled according to their cluster index *in the consecutive layer*, thereby forming sub-clusters. We use linear discriminant analysis (LDA) [24], [25] to find a two-dimensional projection of the activities that maximizes the separation of the examples with respect to their sub-cluster labels. To understand the semantics of the sub-clusters, we draw three representative examples from each sub-cluster near the sub-cluster centroid. Two methods were considered for selecting the representative examples:

1) **Most typical examples ($l_2$):** The $m$ closest images to the sub-cluster center according to the $l_2$ distance.
2) **Most discriminative examples ($LLR$):** Assume $C_k^l$ contains $Z$ sub-clusters corresponding to cluster indices $(a_1, ..., a_Z)$ in the consecutive layer, i.e., an example $I$ belongs to sub-cluster $j$ iff $X^{l+1}(I) \in C_{a_j}^{l+1}$. For such an example, the log-likelihood ratio (LLR) w.r.t sub-cluster $a_z$, $z \neq j$ is given by

$$LLR_I(j,z) = \log P\left(X^{l+1}[I] \mid h^{l+1} = a_j\right) - \log P\left(x^{l+1}[I] \mid h^{l+1} = a_z\right). \quad (26)$$

This is a natural margin, stating the confidence we have in assigning $X^{l+1}[I]$ to cluster $a_j$ rather than $a_z$. Since we want the representative examples of sub-cluster $a_j$ to be discriminative w.r.t *every* other sub-cluster $a_z$, we choose them according to

$$\max_{\{I:h^l(I)=k, h^{l+1}(I)=a_j\}} \min_{z \neq j} LLR_I(j,z). \quad (27)$$

The $l_2$ method chooses examples most characteristic of the sub-cluster, while the LLR method focuses on more discriminative, hence, extreme examples. See Fig. 3 for decision junction examples.

### 3.4.3 Inference Graphs

For MLP networks, the inference path of a specific example $I$ contains a single visual word at each layer. It can be defined by the maximum a-posterior (MAP) cluster sequence, i.e., the sequence $H = (h^1, ..., h^L)$ satisfying

$$\max_{h^1,...,h^L} \log P(h^1, ..., h^L | X(I)). \quad (28)$$

$H$ can be found using the Viterbi algorithm [26]. The path nodes are visualized using the decision junction technique of Section 3.4.2.

For CNNs, multiple spatial words are active at each layer. We use the technique explained in Section 3.3.2 (Algorithm 1) to generate inference graphs highlighting the main active words. Such graphs can be built for an entire class by choosing the input $\Omega$ of Algorithm 1 to be the set of all images predicted to the class, or for a single example. In the visualization of such graphs, each visual word is displayed using its $m$ most representative spatial examples in the validation set. Connections between words are characterized by their contribution to the maximum likelihood (Eq. 25) and using a heat map showing the spatial distribution of explaining words in the receptive field of the target word. For inference graphs of a specific image $I$, node visualization also shows which spatial locations in $I$ belong to the node.

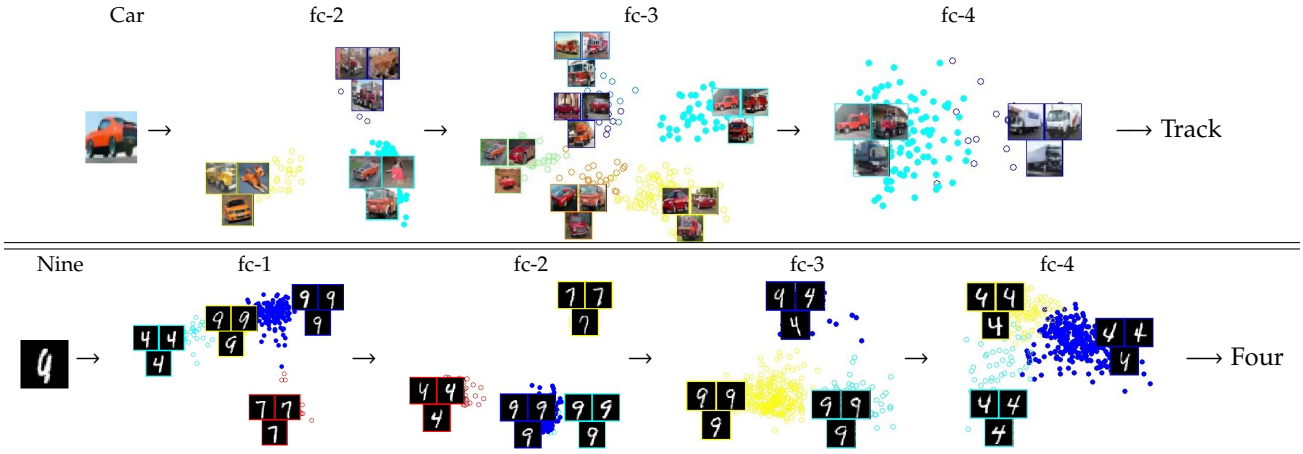



Fig. 4. **Two inference paths in a MLP**. We show the main decision junctions for six layer MLP networks. The analyzed examples are presented on the left. Cluster representatives are chosen using the $l_2$ metric techniques. In each decision junction, the points of the sub-cluster chosen by the example are marked with full circles in cyan (top) and blue (bottom) **Top:** The path of a misclassified CIFAR10 car example is shown. The main decision points occurred in layers fc-2 – fc-4, with the critical decision made in layer fc-3 where the abnormal appearance of the example misled the network to consider the car as a truck. **Bottom:** The path of a misclassified MNist "nine" digit example is shown. The input image traversing main decision points through layers fc-1–fc-4, where the main flawed decision is made in layer fc-3, where the open head of the image misleads the network into considering it as a "four" digit.

## 4 Results

### 4.1 Implementation Details

The HMM for MLP formalism was tested by training fully connected networks on the MNIST [27] and CIFAR10 [28] datasets, containing 10 classes each. The networks included six layers with the first five containing 1,000 neurons each (the last layer has 10 neurons according to the number of classes). Based on a preliminary evaluation, the number of visual words $K^l$ was set at 40 for all layers.

CNN models included ResNet20 [3] trained on CIFAR10, and VGG-16 [29] and ResNet50 [3] trained on the ILSVRC 2012 dataset [30]. For ResNet20, the output of all add-layers after each skip connection were modeled, as these outputs are expected to contain aggregated information. For ResNet50, there are 16 add-layers and the output of add-layers 3, 7, 13, and 16 were modeled. For VGG-16, the first convolutional layers at each block were modeled (four layers in total). The numbers of visual words were set at 100, 200, 450, and 1,500 for layers 1–4, 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 $\frac{1}{K^l}$.

### 4.2 Inference modeling in MLP networks

**Sequential path.** Inference paths drawn as sequences of decision nodes are useful for error diagnosis. In Fig. 4 (top), a path of an erroneous "car" example in the CIFAR10 network is partially presented. The sub-clusters containing the example are marked with full cyan circles. While layers fc-2 and fc-4 primarily make decisions based on color, the wrong decision leading to the mis-classification of the car as "truck" is made at layer fc-3. The example's cluster in layer fc-3 contains six sub-clusters, leading to car and truck clusters in the consecutive layer. At this point, the example was wrongly associated with the sub-cluster representing "truck" due to its exceptional rear appearance, resembling the appearance of a truck front. From this point onwards, the path is associated with "truck" clusters, up until the classification layer. In Fig. 4 (bottom), a path of an erroneous "nine" example in the MNIST network is partially presented with full blue clusters. Correct network decisions are made in layers fc-1 and fc-2, where the network associates the example with primary "nine" sub-clusters. The wrong decision of the network is made in layer fc-3, where it decided to send the example to a "four" cluster in layer fc-4, continuing with this pattern up until the classification layer.

**Cluster similarity development across layers**. Progressing through FC layers, activity clusters tend to become more class oriented, i.e., dominated by examples from a single class. Furthermore, these clusters become increasingly similar with layer index progression, indicating convergence of class examples toward a single class-specific representation. For each cluster, we define its class naturally as the class whose examples are the most frequent among the cluster examples, and the class dominance index is the percentage of dominant class examples. In Fig. 5 (top) cluster purity and inter-cluster distances are shown for the CIFAR10 FC network layers. The similarity of clusters representing the same class gradually increases in layers 3-5, as evident from the emerging block structure. It can be observed that a significant portion of the training occurs in the middle layer of the network, specifically in the transition between the third representation (last layer of the first half) and the fourth (first in the second half). This phenomenon was enhanced in the case of severe overfit discussed below.

**Overfitting capacity in a single layer.** It is known that networks can be trained to an extreme overfit condition by using randomly generated labels in training [6]. The resulting classifier has a training error of 0, meaning that it had successfully memorized a mapping between all training

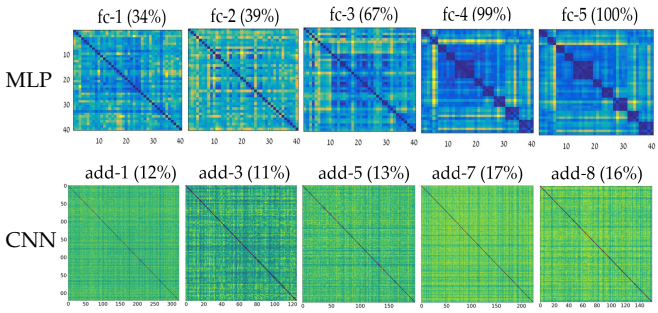



Fig. 5. **Cluster development across layers.** Cluster similarity matrices for increasing layer indices in MLP and CNN networks trained for CIFAR10 classification. In each matrix, Euclidean distances between cluster centers for clusters of a single layer are shown. Clusters are ordered by their dominant class (clusters with dominant class 1, followed by all clusters with dominant class 2, etc.). Layer index and average cluster purity (%) are shown above each matrix. **Top: MLP activity.** Growing similarity between clusters with the same dominant class can be seen by the appearance of a block diagonal structure, from layer 3 onwards. This indicates convergence toward a single class-specific representation as layers progress. **Bottom: CNN activity**. Clusters of add-layers of ResNet blocks 1, 3, 5, 7 and 8 are presented. Here, no similarity is formed with layer growth; each class is represented by multiple localized visual words, which have unrelated activity patterns.

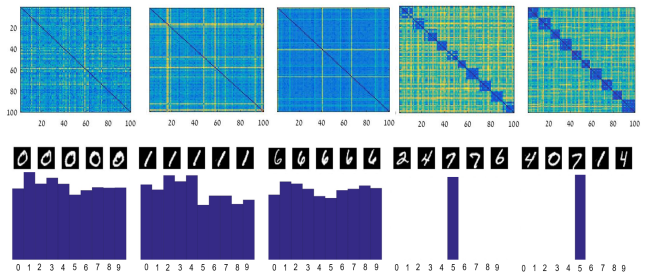


Fig. 6. **Cluster development for an extreme overfit case**. Cluster distances and average purity for a network trained with random labels are shown. **Top:** Cluster similarity matrices for layers 1-5. **Bottom:** Typical clusters at these layers. For each cluster the (pseudo) label histogram is shown, as well as some representative cluster images. An abrupt transition from input-dominated to output-dominated representation occurs in the transformation between the third and fourth layers.

images to their pseudo labels, but its generalization error is of chance level. We utilize our model to understand how this memorization mapping is formed. A six FC-layer network was trained on the 60,000 examples of the MNIST data with random labels, until reaching a zero training error. In Fig. 6, the similarity matrices between cluster centers for layers 1–5 are shown (top), with typical clusters at each layer (bottom), presented using their label histograms and representative images.

Surprisingly, one can observe a concentration of the network overfit behavior in a single layer transformation between the third and fourth layers. In layers $1-3$, clusters are input related. They contain images with similar appearance, hence, with similar true labels (and uniformly distributed pseudo labels). These clusters are not close in the Euclidean sense. In Layer 4, there is a sharp transition to clusters which are completely (pseudo) label dominated, as indicated by their class purity (histogram) and block structure in the

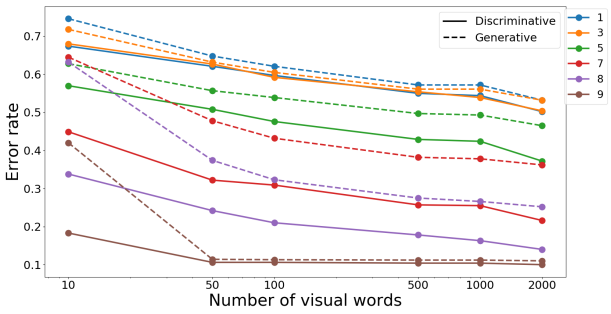



Fig. 7. **Classification accuracy as a function of dictionary properties.** Error rates of a linear classifier trained over word histograms for multiple dictionaries (different numbers of visual words), six convolutional layers (between layers 1 and 9), and the generative and discriminative training losses (Eqs. 16 and 17, respectively).

similarity matrix. The fact that the memorization of the full data (60,000 samples) concentrates almost completely at a single transformation in the middle of the network (between layers 3 and 4) is a novel unexpected observation, for which we do not currently have a good explanation.

### 4.3 Inference modeling in CNNs

**Loss and dictionary sizes.** The discriminative quality of a visual dictionary can be quantified by using it to form word histograms, then checking the error of a linear classifier on this representation (a bag-of-words methodology). Fig. 7 shows the errors obtained for dictionaries trained with losses $\mathcal{L}_G$ (16) and $\mathcal{L}_D$ (17) on intermediate convolutional layers of a ResNet20 network on CIFAR10. The graphs show error rates as a function of the layer index and dictionary size. It can be seen that errors decrease with the layer index (from 1 to 9), approaching the original network error of 0.088. As expected, the discriminative loss $\mathcal{L}_D$, whose minimization directly decreases the error, leads to a higher accuracy than the generatively-optimized loss. Therefore, all models presented below were trained with the $\mathcal{L}_D$ loss. In addition, the error rate decreases monotonically with the dictionary size for all layers, but the gain from dictionary sizes larger than 500 is minor in most cases.

**Cluster development across layers.** Fig. 5 (bottom) shows the cluster distance matrix for several convolutional layers of a ResNet20 network trained on CIFAR10. Unlike in MLPs, learned clusters represent activity at specific spatial locations. Even though receptive fields of clusters from advance layers cover the entire input space, there is no apparent similarity between clusters with the same dominant class. This indicates that activity columns of advanced layers remain local, an observation also supported by the inference graph visualizations shown below. While clusters do become more class specific, they are less specific than in the MLP network, and the final classification is based on several class-specific words appearing simultaneously in different image regions.

**Class inference graph.** An example of a class inference graph for the class "pineapple" in VGG-16 is presented in Fig. 8. This graph is generated by training a clustering model on the "pineapple" and related classes (see Section 3.2), then applying the node selection algorithm (Section 3.3) to the set $\Omega$ of pineapple images included in the validation

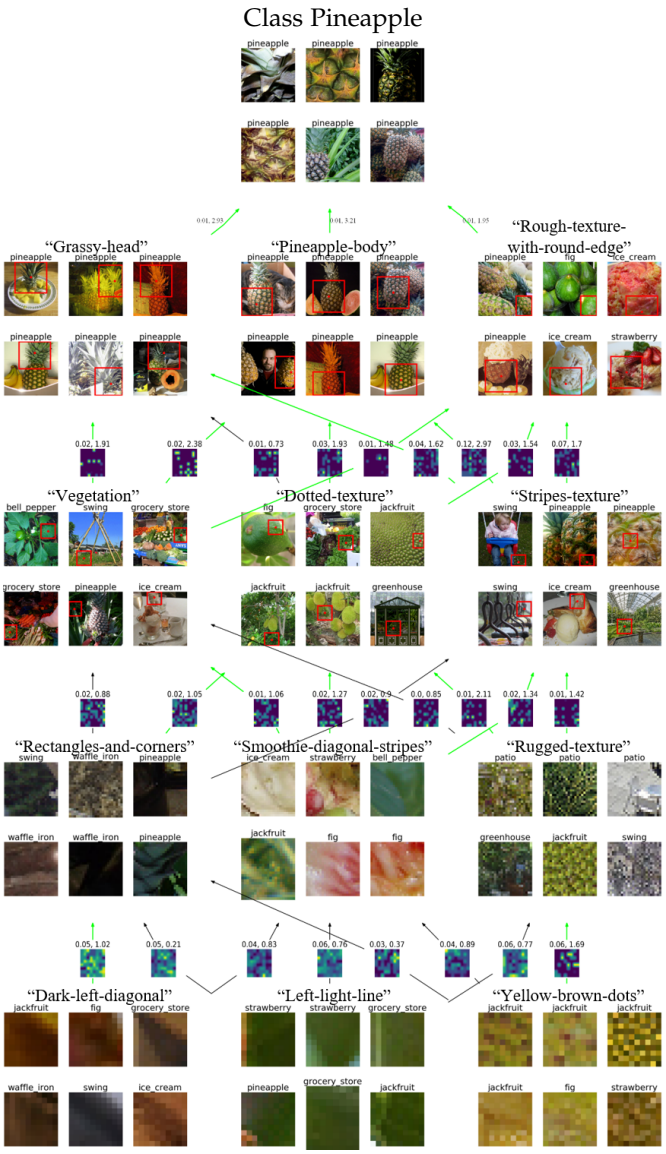



Fig. 8. **Pineapple inference graph**. The graph is generated by training a model on the "pineapple" class and its neighboring classes. The top node is a visual word of the output layer, representing the predicted class "pineapple". The lower levels in the graph show the three most influential words 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. 25). In addition, for each arrow, a heat map is shown indicating the frequent locations of the lower-level visual words in the receptive field of the higher-level word. A tag above each visual word was added by the authors for figure explanation convenience. The figure is best inspected by zooming in on clusters of interest.

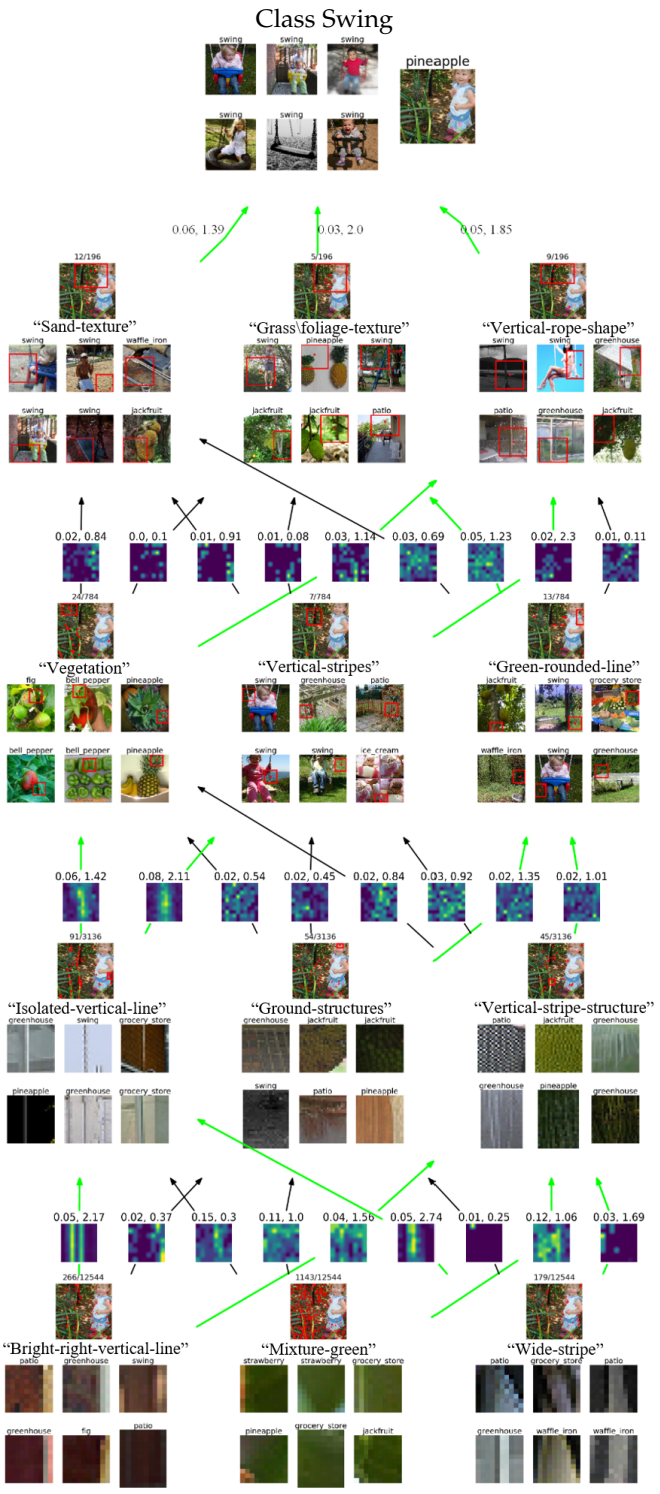



Fig. 9. **An image inference graph of an erroneous image**. An image inference graph for a pineapple image wrongly classified to the class "swing". The model is trained using "pineapple" and its neighboring classes (same as in Fig. 8), where the neighbor class "swing" is included. The graph is generated by applying the node selection algorithm (Section 3.3) to a set $\Omega$ containing this single erroneous image. The analyzed image is shown on the top 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. 10. **Partial image inference graph of a correctly classified zebra image**. The figure presents only the main contributing visual words from the three bottom modelled layers.

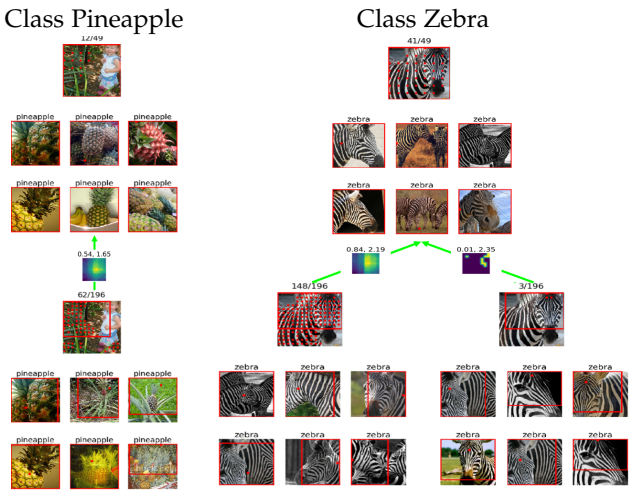



Fig. 11. **Sub-graph inference for ResNet50. Left:** The pineapple image wrongly classified to class "swing" in VGG-16, presented in Fig. 9, is correctly classified to "pineapple" class in ResNet50. The sub-graph presents the visual word from the top layer (add-16) connected to a visual word from the lower layer (add-13). **Right:** The zebra image from Fig. 10, correctly classified in VGG-16, is correctly classified in ResNet50 as well. The sub-graph includes a visual word from the top layer (add-16) aggregated from two visual words from the lower layer (add-13).

set. The graph shows that the most influential words in the top convolution layer can be roughly characterized as "Grassy-head", "Pineapple-body", and "Rough-textured-with-(lower)-round-edge". The origins 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 (in layer 2). Similarly, the origins of "Pineapple body" can be traced back to yellow and brown texture words in lower layers.

**Image inference graphs**. Fig. 9 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 led to this erroneous classification:

- Top layer (layer 4): The visual words connected directly to "swing" class, hence, causing the error, can be characterized as "Sand-texture", "Grass/foliage-texture", and "Vertical-rope-shape". Such words are indeed statistically related to swing presence in images, and many map locations in the inspected image are assigned to them.
- layers 3 and 2: The "Vertical-rope-shape" (of layer 4) originates from a similar visual word, "Vertical-stripes", of layer 3, and this in turn depends strongly on the "Isolated-vertical-line" word in layer 2. The foliage word (in layer 4) mainly originates from the "Vegetation" word in layer 3, which in turn heavily depends on the two brown/green "vertical-stripe-structure" and "Ground-structure" words in layer 2.
- layer 1: The main explanatory words in layer 1 are green and bright vertical edges and lines, which are combined to construct the "Isolated-vertical-line" and "Vertical-stripe-structure" words in layer 2.

In Fig. 10, we show part of the inference graph for a successfully classified zebra image, focusing on the bottom three layers. The gradual development of discriminative stripe-based features can be seen. Visual words in layer 3 (top) are each characterized by a single orientation: vertical (left), leaning to the right (middle), or leaning to the left (right). These words are less sensitive to spatial frequency. They abstract over the spatial frequency by combining words from layer 2 (middle) that mostly differ w.r.t their line spatial frequency and edge patterns. In layer 1 (bottom), we encounter the edge feature patterns composed to create the words of layer 2.

In Fig. 11, we show partial inference graphs for ResNet-50. On the left, upper nodes from the inference graph of "pineapple" image from Fig. 9 are shown. Unlike VGG-16, ResNet50 successfully classifies this image. 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. On the right, we show upper nodes of the ResNet-50 inference graph for the "zebra" image from Fig. 10, successfully classified also by ResNet-50. The top word shown usually captures the zebra's head, with its receptive field center located on the neck. This "head" visual word is formed from the bottom-left word representing a near-head stripes texture, and the bottom-right word which captures a diamond shape located between the zebra's eyes. Both words are discriminative, as indicated by their log-odds score, but the left is frequent and the right is rare, capturing a singular position on the zebra forehead.

## 5 Conclusion

In this paper, we introduced a new approach for interpreting hidden layers activity of deep neural networks based on learning activity cluster dictionaries and transition probabilities between clusters of consecutive modeled layers. We formalized a maximum-likelihood criterion for mining explanatory clusters, and an algorithm for the construction of inference graphs with manageable sizes. Inference graphs can be constructed for 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 here can be used to verify the soundness of the network reasoning and to better understand the network's hidden mechanisms, or conversely, reveal weaknesses and main error causes. Network debugging is currently a difficult and daunting task, and we believe the suggested tools may be a useful component in a developer's debugging toolbox. Beyond its utility in network interpretation and debugging, the suggested approach and tools revealed several surprising network behavior patterns, such as the extreme locality of activity columns in top CNN layers, and the concentration of memorization in a single intermediate middle layer in fully connected networks.

Several interesting avenues are open for future work. One such avenue may be re-training more explainable networks by enforcing, during training, only activity of clusters and connections with high explanatory value. Another direction is to use the models in a task-transfer scenario. Network refinement for a new task may be constrained to use only relevant activation clusters, as determined by a human observer. In this way, a human may use the suggested tool to define a relevant prior for the new task. Finally, we may try to design explanatory capability that goes beyond statistical analysis and maximum-likelihood justification, and into causal analysis based on intervention.

## Acknowledgments

This work was supported by the Israeli Ministry of Science and Technology and Israel Innovation Authority through the Phenomics consortium.

## References

[1] Y. Konforti, A. Shpigler, B. Lerner, and A. Bar-Hillel, "Inference graphs for CNN interpretation," in *16th European Conference on Computer Vision*, 2020.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[5] M. Sedlmair, M. Meyer, and T. Munzner, "Design study methodology: Reflections from the trenches and the stacks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431–2440, 2012.

[6] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[7] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[8] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.

[10] J. Yosinski, J. Clune, A. M. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *CoRR*, vol. abs/1506.06579, 2015.

[11] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.

[12] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6541–6549.

[13] Q. Zhang, Y. Nian Wu, and S.-C. Zhu, "Interpretable convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8827–8836.

[14] R. Liao, A. Schwing, R. Zemel, and R. Urtasun, "Learning deep parsimonious representations," in *Advances in Neural Information Processing Systems*, 2016, pp. 5076–5084.

[15] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, "This looks like that: deep learning for interpretable image recognition," in *Advances in Neural Information Processing Systems*, 2019, pp. 8928–8939.

[16] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

[17] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 91–100, 2016.

[18] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, vol. 3, no. 3, p. e10, 2018.

[19] N. D. Socci, D. D. Lee, and H. S. Seung, "The rectified Gaussian distribution," in *Advances in Neural Information Processing Systems*, 1998, pp. 350–356.

[20] G. Lee and C. Scott, "EM algorithms for multivariate Gaussian mixture models with truncated and censored data," *Computational Statistics & Data Analysis*, vol. 56, no. 9, 2012.

[21] O. Cappé and E. Moulines, "On-line Expectation–Maximization algorithm for latent data models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, no. 3, pp. 593–613, 2009.

[22] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, p. 4–15, January 1986.

[23] R. J. Lipton and R. E. Tarjan, "A separator theorem for planar graphs," *SIAM Journal on Applied Mathematics*, vol. 36(2), p. 177–189, 1979.

[24] A. J. Izenman, "Linear discriminant analysis," in *Modern Multivariate Statistical Techniques*. Springer, 2013, pp. 237–280.

[25] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[26] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

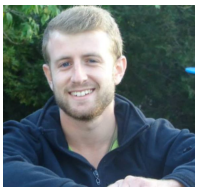
[27] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[28] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

**Yael Konforti** was born in Tel-Aviv, Israel, in 1990. She received her B.Sc (Hons.) degree in Industrial Engineering and Management from Ben-Gurion University, Beer-Sheva, Israel, in 2018 and a M.Sc.(tech.) degree in Computer Vision, Statistics and Quantitative Methods from Ben-Gurion University, Beer-Sheva, Israel, in 2020.



**Alon Shpigler** received his B.Sc. degree in Industrial Engineering and Management from the Ben Gurion University, Israel, in 2017, and a M.Sc in Computer Vision, Statistics and Quantitative methods at Ben-Gurion University, Israel, in 2018. He is working toward his Ph.D. degree in the Ben Gurion University, focusing on utilizing deep networks for Ultrasound and CT imaging applications.

**Boaz Lerner** received a B.A. degree in Physics and Mathematics from the Hebrew University, Israel, in 1982 and a Ph.D. degree in Computer Engineering from Ben-Gurion University, Israel, in 1996. He was a researcher at the Neural Computing Research Group at Aston University, Birmingham, UK and the Computer Laboratory of Cambridge University, Cambridge, UK, and now is an Associate Professor in the Department of Industrial Engineering and Management at Ben-Gurion University, Israel, where he established in 2007 and has since headed the Machine Learning and Data Mining Lab. His current interests include machine learning and data mining approaches to data analysis and their application to real-world problems especially in precision medicine and precision agriculture. Lerner has supervised more than 50 graduate students and has published more than 100 papers in journals and conference proceedings.

**Aharon Bar-Hillel** was born in 1971 in Beer-Sheva, Israel. He received his B.A in math and philosophy from the Tel-Aviv University in 1999 and Ph.D. in neural computation from the Hebrew University of Jerusalem in 2006. He was a machine learning and computer vision researcher in Intel Research (2006-2008), GM Research (2009-2012), and Microsoft Research (2013-2016). Currently, he is a senior lecturer at the Department of Industrial Engineering and Management in Ben-Gurion University, Israel. His research interests include interpretation, acceleration, and decomposition of deep networks, as well as applications of computer vision and machine learning for agricultural phenotyping, imaging for nondestructive testing, visual tracking, and more.