Library network, a possible path to explainable neural networks

Jung Hoon Lee
Allen Institute for Brain Science
Seattle, WA 98109
jungl@alleninstitute.org

Abstract

Deep neural networks (DNNs) may outperform human brains in complex tasks, but the lack of transparency in their decision-making processes makes us question whether we could fully trust DNNs with high stakes problems. As DNNs' operations rely on a massive number of both parallel and sequential linear/nonlinear computations, predicting their mistakes is nearly impossible. Also, a line of studies suggests that DNNs can be easily deceived by adversarial attacks, indicating that DNNs' decisions can easily be corrupted by unexpected factors. Such vulnerability must be overcome if we intend to take advantage of DNNs' efficiency in high stakes problems. Here, we propose an algorithm that can help us better understand DNNs' decision-making processes. Our empirical estimates suggest that this algorithm can effectively trace DNNs' decision processes from one layer to another and detect adversarial attacks.

1 Introduction

Deep neural networks (DNNs) trained via deep learning (DL) have been adopted in a growing number of domains; see [1] for a review. DNNs are considered highly efficient in solving problems because they do not require any detailed instructions from users and can outperform humans in some domains. For instance, DL was successfully used to train 'AlphaGo' to learn a complicated board game 'Go' and defeat Sedol Lee, one of the best 'Go' players [2]. This match demonstrated DNNs/DL's efficiency, but it also revealed that DNNs' operations/decision-making processes are not transparent. Both professional Go players and AlphaGo creators still do not comprehend AlphaGo's adopted strategies against Sedol Lee.

Notably, efficiency alone cannot justify deploying DNNs into all domains. In high stakes problems, safety is more important than efficiency, and it remains unclear whether DNNs' operations could warrant safety [3, 4]. First, DNNs rely on a massive number of parallel and serial numerical operations, making diagnosis of their failures impossible. Second, DNNs' decisions can be easily corrupted by adversarial attacks [5–7]. Therefore, before we deploy DNNs into high stakes problems demanding rigorous decision-making, we need to place safety measures. To this end, it is imperative that we better understand how they reach their decisions.

A line of studies proposed that properties of hidden neurons, representing intermediate stages of DNNs' decisions, provide insights into their operating principles and decision-making processes. First, feature visualization was proposed to study hidden neurons' response characteristics [8, 9]. This approach allows users to identify optimal visual features that can stimulate target hidden neurons, which will advance our understanding of training DNNs; see also [10, 11]. Second, features of hidden layers were proposed to study DNNs' operations [12–

14]. The layer-specific features were analyzed by clustering algorithms and linear classifiers. For instance, Allain and Bengio [12] tested linear-separability of features in hidden layers and found that linear separability increases monotonically, when the selected layer is closer to the last layer. Third, a set of single neurons' responses, evoked by multiple examples, was used as feature vectors. The responses were different from those from multiple neurons evoked by the same input. Raghu et al. [15] used them to study how strongly the hidden layers are correlated with ground truth (i.e., the labels of inputs).

As hidden neurons represent intermediate states of decisions, it seems natural to assume that they encode crucial information in the neural codes underlying decisions and that these codes can help us build more explainable and safer DNNs. Then, how do we find these codes? To address this question, we propose an algorithm that can predict DNNs' answers on unseen examples (i.e., test examples) from hidden layer activity patterns (HAPs). Our motivation is as follows: if an algorithm can predict DNNs' answers, it must capture the crucial codes for their decisions. Our empirical evaluations showed that our newly proposed algorithm can reliably predict DNN's answers, supporting that it can identify the neural codes crucial for their decisions. Furthermore, it can detect adversarial attacks.

2 Results

We assumed that 1) HAPs' meanings remain relative and become apparent only when HAPs are compared with one another and 2) that HAPs evoked by the same-class inputs are clustered together; importantly, the number of HAPs' clusters is not necessarily equal to the number of classes. That is, we need effective methods to estimate similarities among HAPs. In doing so, we turn to our earlier short-term memory systems [16].

Our short-term memory systems [16] store novel input patterns (i.e., substantially different from stored input patterns), and their outputs reflect cosine similarities between a present input and stored inputs. If HAPs, evoked by the same class input patterns, are well clustered together, the stored input patterns in the memory systems can approximate input patterns' (i.e., HAPs) distribution. That is, the stored HAPs can serve as reference points for HAPs' clusters. With this possibility in mind, we use the short-term memory systems, referred to as 'library networks' hereafter, to inspect HAPs' diversity and study their links to DNNs' decisions.

In the first section 2.1, we explain how library networks are constructed and how we use them to explain DNN's answers. In the second section 2.2, we describe our empirical evaluations of the library networks.

2.1 Network structure

2.1.1 Generating libraries of activity patterns in hidden layers

As shown in Fig. 1, a library network has a single synaptic weight layer and accepts normalized HAPs (Eq. 1) as inputs.

$$h_i^k = \sum_j w_{ij}^L \frac{f_j^k}{\left\| \vec{f^k} \right\|}, \text{where } w_{mn}^L = \frac{f_n^m}{\left\| \vec{f^m} \right\|} \tag{1}$$

, where h_i^k represents a synaptic input to output node i of library networks evoked by the input pattern $\vec{f^k}$.

It performs two tasks mainly. First, it estimates activations for inputs and finds maximal activations. When maximal activations are below the predefined threshold value (θ), the inputs are labeled as novel. Second, it stores novel inputs by adding output nodes and imprinting inputs to synaptic weights targeting newly added nodes (Eq. 1); that is, a newly added node effectively 'stores' a present input pattern. As output nodes are continuously added, its size is determined by characteristics of HAPs and the threshold value; naturally, the higher the chosen threshold value is θ, the bigger the constructed library networks are.

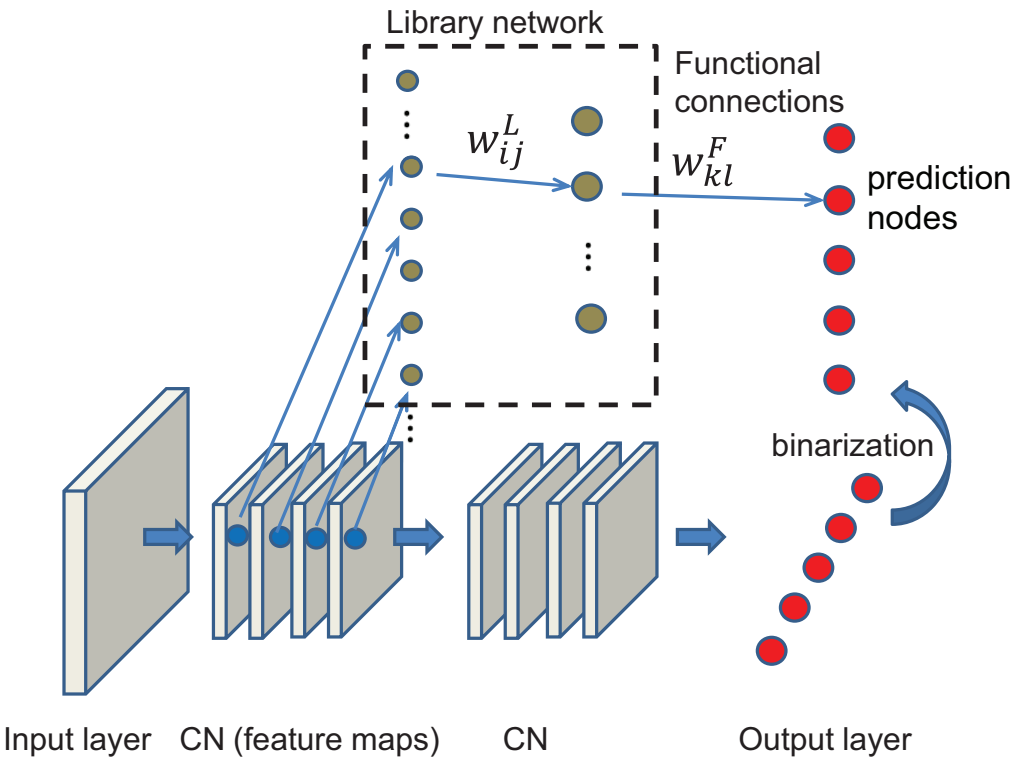

Figure 1: The structure of library networks. The top part of the figure shows the schematics of a library network, whereas the bottom part shows a DNN consisting of multiple convolutional layers (CNs) and a fully-connected (FC) layer (chosen for illustration). For clarification, a single library network is displayed. CN layers produce multiple sheets of feature maps, and all nodes within them are mapped onto input nodes of the library network (indicated by a dashed box). Similarly, all outputs of FC layers are introduced to the library networks. The output nodes of the library networks are determined by the properties of input patterns (outputs of hidden neurons). Then, the outputs of the library networks are correlated with DNNs' outputs (shown in red circles). In the experiments, we also construct the library networks for blocks of layers used in ResNet. The outputs of these blocks are modulated with batch normalization before introduction to the library networks.

In our experiments, we construct library networks for multiple hidden layers while introducing all training examples. Once the library networks are fully constructed with the training set, they can estimate cosine similarities between a present HAP and stored ones. If the previously stored HAP is presented again to the library networks, the activation of output node, added when presented, will be 1, but all other activations are smaller than 1.

2.1.2 Functional connections between library networks and decisions

The motivation of using library networks can be easily explained with an extreme case, in which HAPs' clusters are well separated according to their classes (i.e., desired labels). Suppose that HAPs in the same clusters are extremely similar to each other but significantly different from those in different clusters. Naturally we can assume 1) that the size of the constructed library network (i.e., the number of output nodes) is the same as the number of HAPs' clusters and 2) each output node stores a single example from each cluster. With these well separated HAPs' clusters, outputs of library networks will become binary vectors, in which most components are 0 and a single component is ≈ 1. As each output node estimates the similarity between the present and stored HAPs, the non-zero output indicates that they belong to the same class. That is, in this extreme example, the outputs of library

networks can effectively classify HAPs. In DNNs, such an extreme case may not hold, but we assume that training can enforce HAPs to cluster together depending on classes of input patterns, and thus the library networks can still predict DNNs' answers. Conversely, we can use these hypothetical links between the library networks and DNNs' answers to evaluate how well the hidden layers are trained to find correct answers.

To address this possibility, DNNs' answers are transformed into binary vectors and then introduced into (linear) prediction nodes (Fig. 1), and we establish correlations between the library networks' output nodes and prediction nodes using the Hebbian-rule (Eq. 2):

$$W^F_{mn} = \sum_k g(h^k_n) \times O^k_m, \text{where } g(x) = \exp\left(\frac{-(1-x)}{0.01}\right) \tag{2}$$

, where W^F_{mn} denotes the functional connection from the library network node n to prediction node m; where O^k_m and h^k_n represent the input to prediction node m (depending on DNNs' answers) and the input to output node n of the library network when kth input pattern is presented. During the construction of the functional links, the input of prediction node corresponding to the DNN' s predicted class is 1, and the rest of the inputs are -1; that is, if the DNN predicts that an input pattern k belongs to a class c, O^k_c=1 and $O^k_{j\neq c} = -1$. It should be noted that W^F_{mn} depends on nonlinear kernels (Eq. 2) to render sharper correlations. The training examples are used to establish the functional connections between library networks and DNNs' answers (Fig. 1). With these functional connections, we test the library networks' ability to predict DNNs' decisions on unseen (training) examples by estimating forward inputs to prediction nodes. In the experiment, we calculate the likelihood of answer being m in response to kth input by using three or eight maximally activated output nodes of the library networks (Eq. 3)

$$P^k_m = \sum_{a=1}^{a=3,8} W^F_{ma} \times \text{sort}(h^k)_a \tag{3}$$

, where $\text{sort}(h^k)_a = (h^k_{argmax(h^k)[0]}, h^k_{argmax(h^k)[1]}, ...)$; $argmax(V)$ represents indices of vector V components sorted in descending order.

2.2 Empirical evaluations of utilities of library networks

In this study, we test the library networks' utility with two DNNs, 1) a convolutional network (CNN) trained with MNIST dataset [17] and 2) a ResNet trained with CIFAR10 dataset [18]. MNIST includes 60,000 training examples and 10,000 test images of handwritten digits (0–9), and CIFAR-10 is the collection of 10 classes of items ranging from animals to man-made objects. See section 4 for more details on network implementations and databases.

2.2.1 Peeking into CNN through library networks

CNN used here is an implementation of a variation of LeNet-5 [17] consisting of 2 convolutional (CN) layers and 2 fully-connected (FC) layers. As the size of the library networks reflects HAPs' homogeneity, we first measure the sizes of 4 library networks for CN1, CN2, FC1 and FC2 layers in the CNN, depending on threshold values (θ). When the same threshold value is used for all layers, the library networks of the earlier hidden layers are bigger than those of the later layers (Fig. 2A); for instance, the library network for FC1 is bigger than the others when a threshold value is chosen for all layers. That is, HAPs in the earlier layers are more heterogeneous, suggesting that the input vectors are transformed into homogeneous ones (possibly, homogeneous clusters), as information propagates through the CNN.

Next, using the Hebbian rule (Eq. 2), we build functional connections (i.e., correlations) between 4 library networks and CNN's answers. Once the functional connections are established, we test how well the library networks can predict DNNs' answers/decisions (Section 2.1.2); to estimate P^k_m in Eq. 3, we use 3 maximally activated output nodes of the library networks for CNN. Fig. 2B shows the prediction accuracy from all 4 library networks for

CN1, CN2, FC1 and FC2 depending on θ. As shown in the figure, the library networks can reliably predict CNN's answers on test examples, especially when the higher threshold values are chosen. We further test the predictive power of the library networks by allowing them to provide three best answers. If CNN's decision coincides with one of the three best answers proposed by the library networks, we count it as a correct answer. As shown in Fig. 2C, the library networks exhibit dramatically enhanced predictive power.

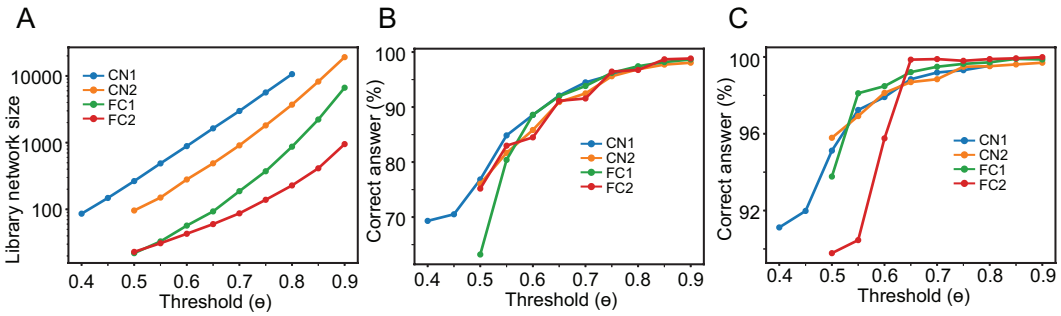

Figure 2: Empirical evaluations using the CNN trained for MNIST. (A), The sizes of the library networks (i.e., the number of output nodes) depending on threshold values for novelty detection (see the text). The color codes are used to specify target layers. For instance, CN1 represents a library network's size using HAPs from the first CN layer. (B), The fraction of correct predictions of the library networks using single best answers. (C), The same as (B) but the three best answers are used for predictions.

We note that the predictive power of the library networks for the later layers are higher than those for the earlier layers. This is consistent with the notion that input vectors are transformed progressively to networks' decisions through layers (see Alain and Bengio [12], for instance). As the library networks seem to represent information regarding CNN's decisions, we further track changes in predictions from one layer to another by establishing the confusion index (CI) between two digits using the inputs to prediction nodes in each layer. $CI(d_1, d_2)$ is formally defined in (Eq. 4).

$$CI(d_1, d_2) = \frac{CA(d_1, d_2)}{CA(d_1, d_1)}, \text{where } CA(d_1, d_2) = \frac{1}{(\sum_{i=1}^{i=10} \exp(P_i^k))^2} \sum_k \exp(P_{d_1}^k) \times \exp(P_{d_2}^k) \tag{4}$$

, where P_i^k represents input to ith prediction node in response to kth input pattern (eq. 3); for instance, the first and last prediction nodes represent digits 0 and 9, respectively.

We use trials (i.e., forward passes) to estimate CI, in which a maximally activated prediction node correctly predicts the digit (d_1) presented to the CNN. As such, CI can estimate the probability of the library networks' misidentifying digits d_1 as d_2, on average. As $CI(d_1, d_1)$ is always 1, we do not report them below.

Figure 3 A-C show the confusion matrices whose components are CIs, when $\theta = 0.65$. We do not display CI for FC2 because they are extremely low. The y-axis represents d_1 (correct answer), and the x-axis, d_2, respectively. As shown in the figure, digit 9 confuses CN1 layer (i.e., the first CN). Interestingly, CN2 also shows relatively high CI values when digit 9 is presented (i.e., $d_1 = 9$). These confusions are alleviated in FC1 layer. To determine whether this trend is a consequence of low θ, we increased θ from 0.65 to 0.75 and found equivalent results (Fig. 3 D-F). The high CI values in response to digit 9 ($d_1 = 9$) can be explained by the fact that digit 9 has multiple visual features similar to those in other digits. Thus, CN layers, which rely on limited spatial filter, may have difficulty differentiating 9 from others. FC layers, fully connected to previous layers, can use global features (e.g., locations of features) to precisely recognize digits. We also note that $CI(4, 9)$ is high in all layers, which seem natural given the similarity between 4 and 9. These results raise the possibility that CN layers are optimized to identify local features, while FC layers are optimized to

utilize the (relative) locations of the features detected by CN layers. Thus, we propose that the library networks and confusion matrices help us infer the workflow of DNNs consisting of CN and FC layers.

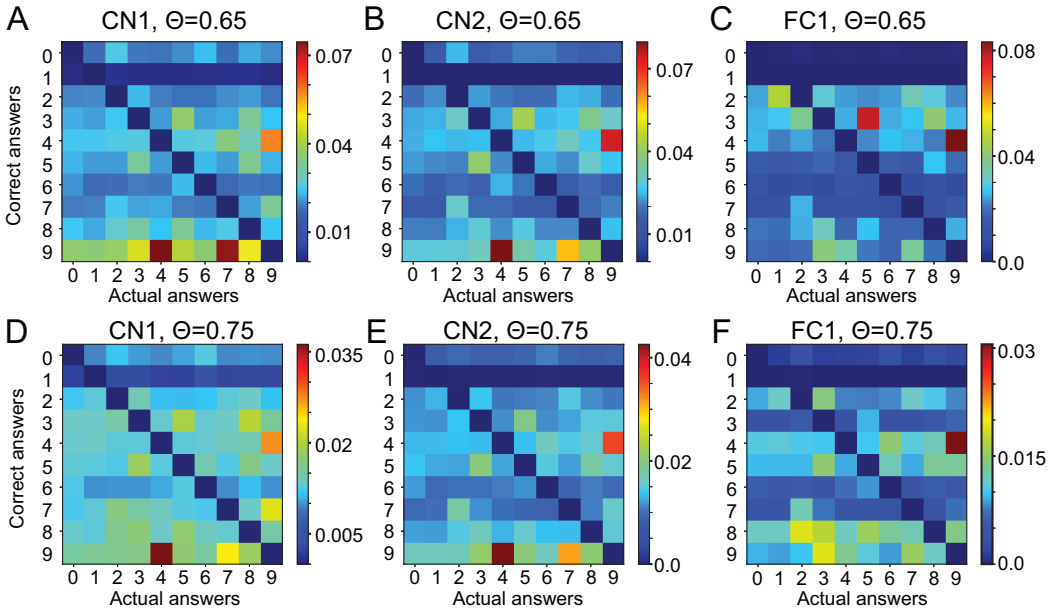

Figure 3: Confusion matrix. (A)-(C), The confusion indices estimated using the library networks' predictions constructed for CN1, CN2, FC1, respectively. y-axis represents the correct digit (d_1), and x-axis represents a possible answer (d_2). For instance, the values shown in the 10th row and 5th column represent the probability of the library networks' reporting digit '4' in response to digit '9'. (D)-(F), The same as (A)-(C), but the threshold value is 0.75 instead of 0.65.

2.2.2 Peeking into ResNet through library networks

We further test the library networks' utilities via a ResNet trained with CIFAR 10 dataset. Specifically, we use the pretrained ResNet44 network from the public github repository (section4). Although there are 44 layers in the ResNet, it is organized with 5 functional blocks. The three blocks, composite layers (CL) 1, 2 and 3, include multiple layers, and the rest (CN1 and FC) are single layers. The library networks are constructed to inspect the 5 blocks instead of all hidden layers, and we repeat the same experiment. In the ResNet, we used the 8 highest output activations of the library networks to calculate the input to prediction nodes P_k^m (Eq. 3).

We note that HAPs in the ResNet strongly vary from one block to another, and thus widely different threshold values θ are necessary to describe them properly. Specifically, we used 8 sets of θs for CN1, CL1-3 and FC, respectively; see Table 1 for the actual values. Figure 4 shows the accuracies of 5 library networks' predictions depending on the 8 sets of θs. Although the accuracies are lower than those of MNIST case (Fig. 2), the library networks can still reliably predict the ResNet's decisions on test examples (Fig. 4B and C), which indicates that even HAPs in the ResNet are forced to cluster during training. We also note that HAPs in CL1 (e.g., the first composite layer) are more homogeneous than those in any other blocks/layers in the ResNet (Fig. 4A), which is distinct from the gradual increase in HAPs' homogeneity in the CNN (Fig. 2A). This suggests that the main function of the RenNet's first block (CL1) is to transform feature vectors of CN1 into homogeneous vectors and that the function of subsequent blocks/layers is to find optimal ways to map these highly homogeneous vectors (from the first block) into proper clusters to reflect the classes of input patterns.

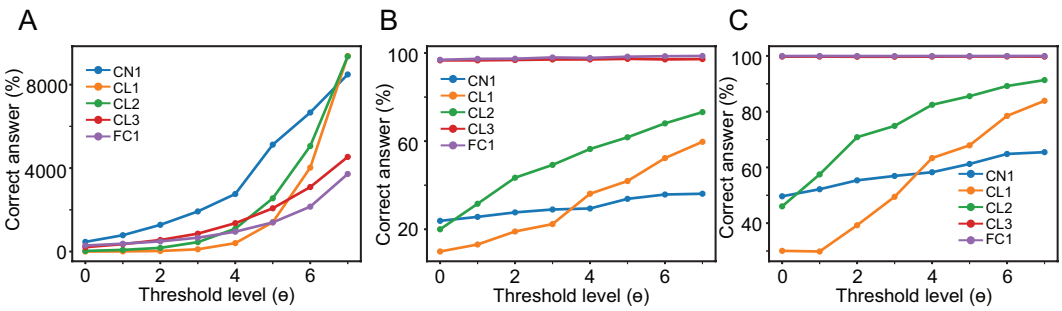

Figure 4: Empirical evaluations using the ResNet trained for CIFAR 10. (A), The sizes of the library networks (i.e., the number of output nodes) depending on threshold values θ for novelty detection. Since HAPs' homogeneity varies significantly from one layer to another, a widely different set of threshold values is chosen for each layer. The actual values are listed in Table 1. (B), The fraction of correct predictions of library networks using single best answers. (C), The same as (B), but the three best answers are used for predictions.

2.2.3 Adversarial attacks reduce consistency among library networks' predictions

The results above indicate that the library networks can predict DNNs' answers and that multiple library networks provide multiple predictions. In our experiments, we note that the library networks' predictions are largely consistent with each other and thus assume that consistency among predictions of library networks (CPL, hereafter) is the result of training; if DNNs are well trained, all parts of the networks are consistent with one another. This leads us to speculate that CPL could decrease when inputs are drawn from 'out-of-train' domains (i.e., novel domains). We address this hypothesis by introducing adversarially manipulated inputs and measuring CPL using the correlations between prediction node outputs of the library networks (Eq. 5).

$$CPL(k) = \frac{1}{N} \sum_{i>j} \frac{\vec{P_i^k} \cdot \vec{P_j^k}}{\left\|\vec{P_i^k}\right\| \left\|\vec{P_j^k}\right\|}, \tag{5}$$

, where $\vec{P_i^k}$ represents inputs to prediction nodes from RestNet's block i or CNN's hidden layer i elicited by kth input pattern, and N is the number of all possible pairs of blocks/layers. By definition, we get a single CPL value for an input pattern.

In our study, we use the routine 'LinfPGDAttack' included in the 'advertorch' [19], which implements the projected gradient descent attack [20], to generate 200 adversarial images from MNIST and CIFAR 10 datasets, respectively. We fix the iteration number at 40 and step size at 0.01, while we test multiple perturbation sizes ϵ. CPLs are estimated according to Eq. 4 for 200 normal and adversarial images. To evaluate CPLs' dependence on a threshold value θ of each layer, we test 7 sets of threshold values (see Table 1). Fig. 5A shows a example of CPL distributions calculated with normal and adversarial images of MNIST dataset. As shown in the figure, CPLs are distinct and well separated between normal and adversarial inputs. To quantify how well they can be separated by an ideal observer, we calculate the area of the receiver operating characteristic curve (AUROC); see [21] for details. Fig. 5B shows the changes in AUROC depending on the degree of adversarial attacks (ϵ). The color codes represent the selected set of threshold values θs. We also test CPLs of the ResNet between normal and adversarial inputs. As shown in Fig. 5C, the AUROC values can be higher than 0.8, indicating that the library networks can also detect adversarial images of CIFAR-10. Based on these results, we propose that CPL can be used to detect adversarial attacks or more broadly the out-of-training examples (see below).

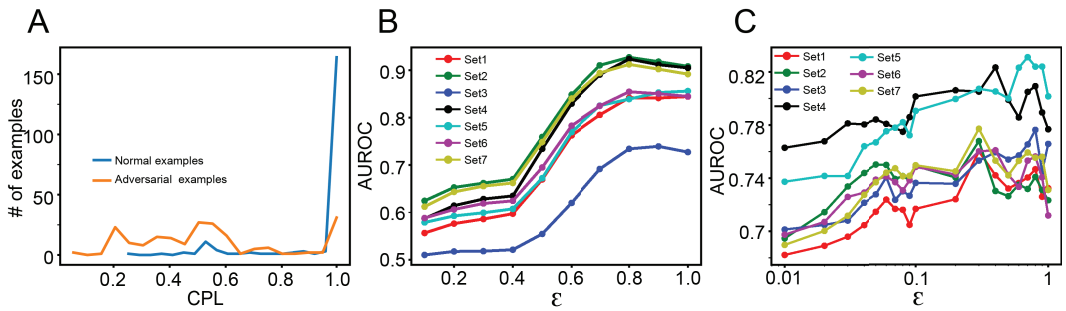

Figure 5: Detecting adversarial attacks: (A), The histogram of comparing CPL values between normal and adversarial examples. (B), AUROC calculated using CPL values from the CNN trained for MNIST, depending on the ϵ. The colors represent a set of threshold values chosen for the library networks; see Table 1 for the actual values. (C), The same as (B), but AUROC calculated from the ResNet.

3 Discussion

To probe DNNs' decision-making processes, we studied the functional links between HAPs and DNN's decisions using the library networks. Our empirical evaluations show that library networks can reliably predict DNN's answers, suggesting that they can identify (internal) neural codes crucial for their decisions. We further found 1) that library networks can allow us to infer functions of hidden layers and 2) that they can detect adversarial attacks. Thus, we propose that library networks can help us build more explainable and safer DNNs.

While in line with the earlier studies focusing on the properties of hidden neurons to explore DNNs' operating principles, our study differs in that 1) it emphasizes the importance of similarities among HAPs in understanding DNNs' decision-making processes, 2) it proposes the library networks that can estimate similarities to analyze DNNs' decision-making processes and 3) it proposes potential links between CPL (consistency of predictions of library networks) and the detection of adversarial attacks. Below we discuss the implications of our study in detail.

3.1 Transparency of DNNs' decisions and library networks

The library networks allow us to inspect how well the hidden layer activity is correlated with DNNs' answers. While the estimated correlations may not fully explain DNNs' operations, they do allow us to look into their decision-making processes. In our CNN experiments, the confusion matrices suggest that 1) CN1 is trained to detect necessary visual features, 2) CN2 is trained to amplify differences among the features, and 3) FC1 and 2 are trained to find ideal ways to utilize global features like locations to make decisions. With the hidden layers' potential functions, we can make an effective flowchart explaining DNNs' decision processes.

Further, we note that it is possible to use the library networks to estimate individual layers' performance and then selectively improve a under-performing layer. If a library network suggests that two classes cannot be clearly separated from each other in one of the CN layers (or a functional block of DNNs assigned to feature-detection), we can conclude that more filters are needed for the layer (or blocks) and provide additional filters to improve its resolution power. We could selectively train the added filters while keeping the old filters intact. Alternatively, if a library network suggests that a FC layer performs poorly, we can retrain it while keeping the other parts intact or add more FC layers to the networks to enhance DNNs' performance.

3.2 Detecting out-of-training tasks via library networks

DNNs can work properly only when test inputs and training examples are drawn from the same domain. More importantly, after being trained, DNNs implicitly assume that the current inputs originate from the domain, for which they are trained. For instance, once DNNs are trained with MNIST, they will always report numerical digits as the best answers even when the inputs are alphabet letters. These indiscriminate answers to 'out-of-training' examples may result in critical errors under particular circumstances. Thus, DNNs need automatic systems to detect out-of-training examples to prevent catastrophic failures. We note that adversarially manipulated inputs can be considered as out-of-training examples. Noting that CPL is reduced when adversarially manipulated inputs are introduced, we propose that the library networks and CPL can be used to determine whether the inputs originate from the proper domains; when CPL values are too low, we should seek a second opinion (i.e., alternative DNNs or human opinion) or retrain the networks.

3.3 Future directions

In this study, we maintain the structure of the library networks and learning algorithms for functional connections (Eq. 2) as simple as possible. Instead of refining them, we focused on addressing the potential functional links between hidden layer activity patterns and DNNs' decisions. Importantly, even though the library networks and functional connections are not highly optimized, they can successfully predict DNN's answers on test examples, suggesting that HAPs are effectively clustered according to their classes during training. We believe that these functional clusters of HAPs can shed light on DNNs' decision-making processes. Thus, in the future, we will extend the library networks to further study functional clusters of HAPs in two ways. First, we will test the capability of new library networks which take a subset of hidden neurons (chosen sparsely and randomly) from the same hidden layers. This 'sparse sampling' can reduce the library networks' sizes, which may be necessary for a large-scale dataset or networks, and we will study the predictive powers of the 'sparse sampling' library networks. Second, we will construct library networks that sparsely sample hidden neurons across layers to determine whether DNNs can develop internal concepts necessary for tasks. If a library network detects a crucial set of hidden neurons across hidden layers, it suggests that DNNs can develop certain task-specific 'concepts' and use them to perform tasks. We believe that these hypothetical concepts, if they exist, can shed light on DNNs' operating principles and help us build truly explainable DL/DNNs.

Table 1: We list the threshold (θ) values that are used to build the library networks for the ResNet. The top rows show the ranges of θ tested for all 5 layers. The bottom rows show the θ chosen for calculating CPLs.

Level	0	1	2	3	4	5	6	7
CN1	0.18	0.2	0.22	0.24	0.26	0.3	0.32	0.34
L1	0.64	0.66	0.68	0.7	0.72	0.74	0.76	0.78
L2	0.48	0.5	0.52	0.54	0.56	0.58	0.6	0.62
L3	0.5	0.52	0.54	0.56	0.58	0.6	0.62	0.64
FC	0.8	0.82	0.84	0.86	0.88	0.9	0.92	0.94

	CN1	L1	L2	L3	FC
Set1	0.24	0.72	0.58	0.62	0.94
Set2	0.26	0.72	0.56	0.58	0.88
Set3	0.3	0.74	0.58	0.6	0.9
Set4	0.32	0.76	0.6	0.62	0.92
Set5	0.34	0.78	0.62	0.64	0.94
Set6	0.26	0.72	0.6	0.62	0.92
Set7	0.26	0.72	0.62	0.64	0.94

4 Methods

Both CNN and ResNet were implemented using 'Pytorch', the open-source python machine learning libraries [22]. CNN used here is an implementation of a variation of LeNet-5 [17]

consisting of 2 convolutional (CN) layers and 2 fully-connected (FC) layers. We adopted the official implementation of pytorch [23] and trained it with default parameters from the released example. For the ResNet, we adopted pre-trained networks and implementation of ResNet44 available in the public github repository [24]; ResNet44 is referred to as ResNet.

MNIST includes 60,000 training and 10,000 test images of handwritten digits (0–9). Each image consists of 28-by-28 8-bit gray pixels. CIFAR-10 is the collection of 10 classes of items ranging from animals to man-made devices. Each class has 500 training and 100 test examples, each of which is a 32-by-32 color image. MNIST and CIFAR-10 can be obtained from http://yann.lecun.com/exdb/mnist/ and https://www.cs.toronto.edu/ kriz/cifar.html, respectively.

Using these datasets and networks, we tested the predictive power of the library networks constructed for CNN and ResNet. All codes used in this study are freely available in the public github repository [25] without restriction.

References

[1] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] Wikipedia contributors. Alphago — Wikipedia, the free encyclopedia, 2019. [Online; accessed 8-November-2019].

[3] Zachary C. Lipton. The Mythos of Model Interpretability. In *ICML WHI*, 2016.

[4] Cynthia Rudin. Please Stop Explaining Black Box Models for High Stakes Decisions. In *NIPS Workshop*, 2018.

[5] Naveed Akhtar and Ajmal Mian. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access*, 6(August):14410–14430, 2018.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, pages 1–11, 2015.

[7] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial Attacks on Neural Network Policies. *arXiv*, page 1702.02284, 2017.

[8] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[9] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 2019. https://distill.pub/2019/activation-atlas.

[10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, 2009.

[11] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv*, (1312.6034), 2013.

[12] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv*, 2016.

[13] Gr??goire Montavon, Mikio L. Braun, and Klaus Robert M??ller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12:2563–2581, 2011.

[14] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Interpretable Deep Convolutional Neural Networks via Meta-learning. *Proceedings of the International Joint Conference on Neural Networks*, 2018-July, 2018.

[15] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):6077–6086, 2017.

[16] Jung H. Lee. DynMat, a network that can learn after learning. *Neural Networks*, 116:88–100, 2018.

[17] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patric Haffner. Gradient-Based Learning Applied to Document Recognition. *PROC. OF IEEE*, 1998.

[18] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *Technical report, University of Toronto*, pages 1–60, 2009.

[19] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *NeurIPS*, page 1706.06083, 2019.

[21] Wikipedia contributors. Receiver operating characteristic — Wikipedia, the free encyclopedia, 2019. [Online; accessed 28-September-2019].

[22] Adam Paszke, Sam Gross, Soumith Chintala, Edward Chanan, Gregory Yang, Zachary DeVito, Alban Lin, Zeming Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[23] Pytorch-team. Pytorh reinforcement examples, 2018.

[24] Yerlan Idelbayev. pytorch resnet cifar10, 2018.

[25] Jung Hoon Lee. https://github.com/giscard88/hypothesis-building, 2019.