Interpretability: Interpreting Deep Neural Networks

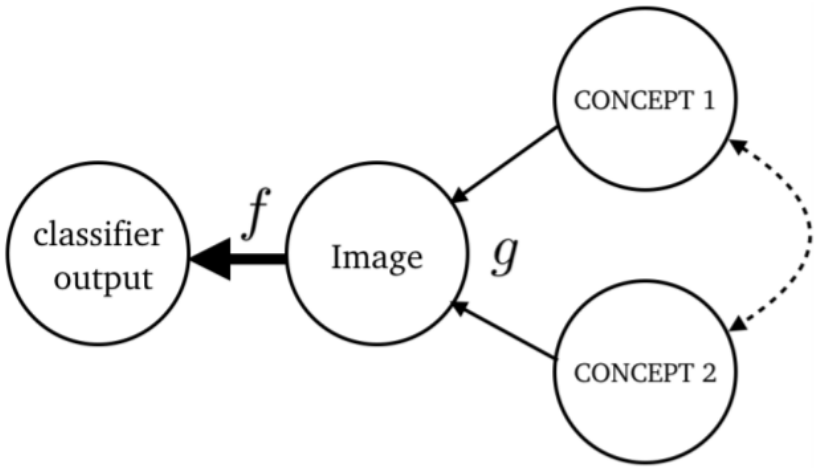
Yicheng Zhong 1901213388

May 19, 2020

Overview

1. Interpreting concepts of DNN's prediction
2. Interpreting DNN's middle-layer features
3. Interpreting DNNs with frequency parts
4. Interpreting DNNs with information bottleneck principle
5. Interpreting DNNs in Deployment

Explaining Classifiers with Causal Concept Effect (CaCE)



Causal graph relating high-level concepts. The authors propose to use a conditional-VAE conditioned on concepts to approximate this relation.

1

[1] Yash Goyal, Uri Shalit, and Been Kim. "Explaining Classifiers with Causal Concept Effect (CaCE)". In: *CoRR* abs/1907.07165 (2019). arXiv: 1907.07165. URL: http://arxiv.org/abs/1907.07165.

Explaining Classifiers with Causal Concept Effect (CaCE)

Define structural causal model for the image I:

$$(C_0, C_1, ..., C_k) = h(\epsilon_C),$$
$$I = g(C_0, C_1, ..., C_k, \epsilon_I),$$

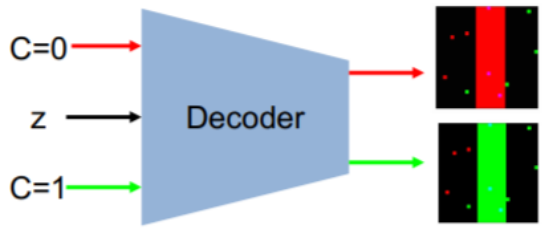
and the interventional SCM setting C_0 to $a \in 0, 1$:

$$(C_0, C_1, ..., C_k) = h(\epsilon_C),$$
$$C_0 = a,$$
$$I = g(C_0, C_1, ..., C_k, \epsilon_I).$$

Definition 1(Causal Concept Effect, CaCE)
$CaCE(C_0, f) = \mathbb{E}_g[f(I)|do(C_0 = 1)] - \mathbb{E}_g[f(I)|do(C_0 = 0)]$.
Definition 2(CaCE for N-way categorical concepts)
$CaCE(C_0, f, a, b) = \mathbb{E}_g[f(I)|do(C_0 = a)] - \mathbb{E}_g[f(I)|do(C_0 = b)]$.

Explaining Classifiers with Causal Concept Effect (CaCE)

Why do we need CaCE?



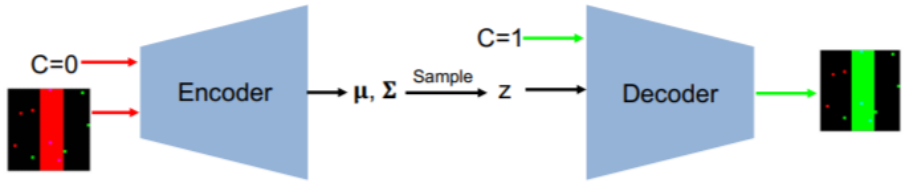
- unconfounded scenario: each concept (red or green) is equally balanced with the labels (horizontal and vertical). TCAV $= 0$.
- biased dataset: 90% horizontal red, 10% vertical bars red. TCAV $= 1.0$.

Explaining Classifiers with Causal Concept Effect (CaCE)

Measuring CaCE

Estimating CaCE in real-world is challenging.

- Ground Truth CaCE (GT-CaCE)
- VAE-CaCE



Explaining Classifiers with Causal Concept Effect (CaCE)

Experiments

$$\text{TCAV}_{C,k,l} = \frac{|x \in X_k : S_{C,k,l}(x) > 0|}{|X_k|},$$
$$\text{CONEXP}(C, f) = \mathbb{E}[f(I)|C_0 = 1] - \mathbb{E}[f(I)|C_0 = 0]$$

- BARS: CaCE for a synthetic dataset

Table 1. CaCE results for BARS dataset.

% of red in class 0 (horz)	% of red in class 1 (vert)	GT-CaCE	Dec-CaCE	EncDec-CaCE	ConExp	TCAV
60	40	0.00	0.02	0.00	0.21	0.76
99	01	0.58	0.59	0.30	1.00	1.00
98	02	0.47	0.49	0.26	0.97	0.96
99	50	0.39	0.44	0.04	0.69	0.96

- Colored-MNIST: CaCE for N-way categorical concept

Table 2. CaCE results for Colored-MNIST dataset.

σ	Avg-GT-CaCE	Dec-CaCE	EncDec-CaCE	ConExp	TCAV
0.02	0.094	0.097	0.099	0.154	0.16
0.025	0.089	0.092	0.093	0.147	0.155
0.03	0.076	0.079	0.08	0.135	0.152
0.035	0.068	0.07	0.071	0.133	0.152
0.04	0.061	0.063	0.065	0.121	0.139
0.045	0.062	0.065	0.066	0.131	0.13
0.05	0.058	0.061	0.062	0.118	0.117

Explaining Classifiers with Causal Concept Effect (CaCE)

Experiments

- COCO-Miniplaces: CaCE for high-dimensional images

Table 3. CaCE results for COCO-Miniplaces dataset.

% of obj in 'bathroom'	% of obj in 'shower'	GT-CaCE	Dec-CaCE	EncDec-CaCE	ConExp	TCAV
60	40	0.13	0.154	0.078	0.23	0.723
99	01	0.694	0.651	0.345	0.841	1.000
95	05	0.604	0.543	0.262	0.791	0.988
99	50	0.328	0.31	0.291	0.49	0.944

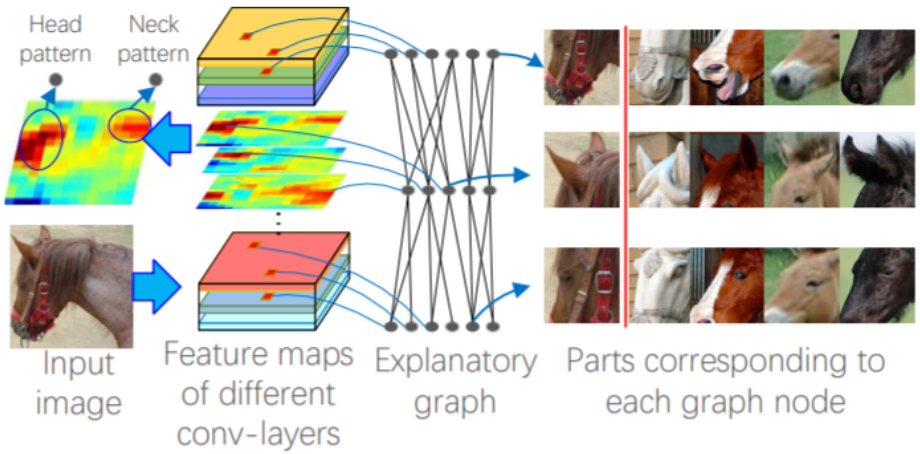
- CelebA: CaCE for naturally occurring confounding concepts

Table 4. CaCE results for CelebA dataset for 'blonde hair' concept.

% of blonde women	% of blonde men	StarGAN	Dec-CaCE	EncDec-CaCE	ConExp	TCAV
60	40	0.049	0.057	0.05	0.152	0.755
99	1	0.523	0.585	0.513	0.752	1
98	2	0.448	0.537	0.44	0.706	1
95	5	0.468	0.479	0.474	0.642	0.978
99	50	0.176	0.209	0.174	0.376	0.953

Interpreting CNN knowledge via an Explanatory Graph

- How many types of patterns are memorized by each convolutional filter of the CNN?
- Which patterns are co-activated to describe an object part?
- What is the spatial relationship between two patterns?



2

[2] Quanshi Zhang et al. "Interpreting CNN knowledge via an Explanatory Graph". In: *CoRR* abs/1708.01785 (2017). arXiv: 1708.01785. URL: http://arxiv.org/abs/1708.01785.

Interpreting CNN knowledge via an Explanatory Graph

Method

Disentangle the d-th filter of the L-th conv-layer into $N_{L,d}$ different part patterns as $N_{L,d}$ nodes in the L-th layer of explanatory graph G.
Let X_L^I be the feature map of the L-th conv-layer, R_L^I be the position inference results for all nodes in the L-th layer, θ_L be the Parameters of these nodes in the L-th layer, which mainly encode spatial relationships between these nodes and the nodes in the $(L+1)$-th layer.
Objective function for the L-th layer :

$$\arg\max_{\theta_L} \prod_{I \in \mathbf{I}} P(X_L^I | R_{L+1}^I, \theta_L)$$

Interpreting CNN knowledge via an Explanatory Graph

Objective function

Define $F(x) = \beta \max(f_x, 0)$ be the number of activation entities at the position p_x, f_x is the normalized response value of x.
Therefore, like GMM, use all patterns in the L-th layer to jointly explain the distribution of activation entities on the d-th channel of X_L:

$$P(X_L^I | R_{L+1}^I, \theta_L) = \prod_{x \in X_L} P(x | R_{L+1}^I, \theta_L)^{F(x)}$$
$$= \prod_{x \in X_L} \{ \sum_{V \in \Omega_{L,d} \cup V_{none}} P(V) P(p_x | V, R_{L+1}, \theta_L) \}_{d=d_x}^{F(x)}$$

$$P(p_x | V, R_{L+1}, \theta_L) = \begin{cases} \gamma \prod_{V' \in E_V} P(p_x | p_{V'}, \theta_L)^{\lambda}, V \in \Omega_{L,d_x} \\ \gamma\tau \qquad\qquad , V = V_{none} \end{cases}$$

$$P(p_x | p_{V'}, \theta_L) = \mathcal{N}(p_x | \mu_{V' \longrightarrow V}, \delta_{V'}^2)$$

Interpreting CNN knowledge via an Explanatory Graph

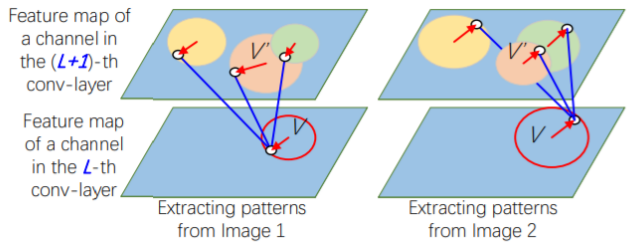


Figure 3: Related patterns V and V' keep similar spatial relationships among different images. Circle centers represent the prior pattern positions, *e.g.* μ_V and $\mu_{V'}$. Red arrows denote relative displacements between the inferred positions and prior positions, *e.g.* $\mathbf{p}_V - \mu_V$.

Inputs: feature map $\mathbf{X}_L$ of the L-th conv-layer, inference results $\mathbf{R}_{L+1}$ in the upper conv-layer.
Outputs: μ_V, E_V for $\forall V \in \Omega_L$.
Initialization: $\forall V$, $E_V = \{V_{\text{dummy}}\}$, a random value for $\mu_V^{(0)}$
for $iter = 1$ *to* T **do**
  $\forall V \in \Omega_L$, compute $P(\mathbf{p}_x, V|\mathbf{R}_{L+1}, \boldsymbol{\theta}_L)$.
  for $V \in \Omega_L$ **do**
    1) Update μ_V via an EM algorithm,
    $\mu_V^{(iter)} = \mu_V^{(iter-1)} + \eta \sum_{I\in\mathbf{I}, x\in\mathbf{X}_L} \mathbf{E}_{P(V|\mathbf{p}_x, \mathbf{R}_{L+1}, \boldsymbol{\theta}_L)}[$
    $F(x) \cdot \frac{\partial \log P(\mathbf{p}_x, V|\mathbf{R}_{L+1}, \boldsymbol{\theta}_L)}{\partial \mu_V}]$.
    2) Select M patterns from $V' \in \Omega_{L+1}$ to construct E_V based on a greedy strategy, which maximize $\prod_{I\in\mathbf{I}} P(\mathbf{X}_L|\mathbf{R}_{L+1}, \boldsymbol{\theta}_L)$.
  end
end
Algorithm 1: Learning sub-graph in the L-th layer

Interpreting CNN knowledge via an Explanatory Graph

Experiments

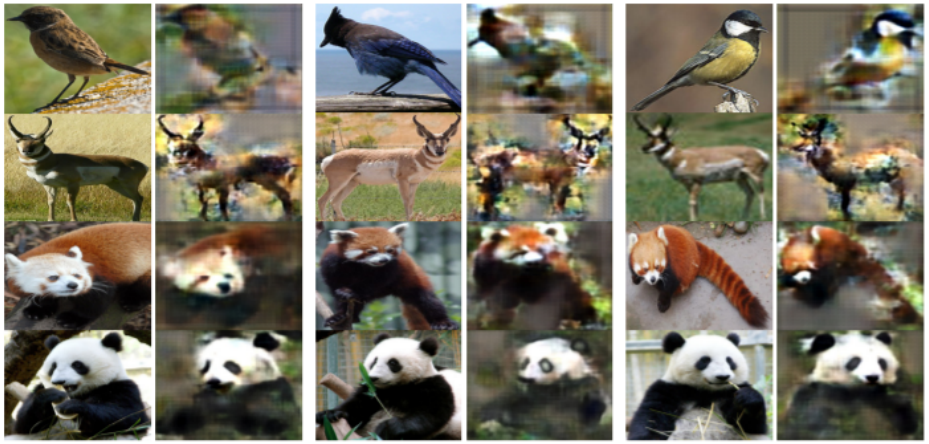


Figure 7: Image synthesis result (right) based on patterns activated on an image (left). The explanatory graph only encodes major part patterns hidden in conv-layers, rather than compress a CNN without information loss. Synthesis results demonstrate that the patterns are automatically learned to represent foreground appearance, and ignore background noises and trivial details of objects.

Interpreting CNN knowledge via an Explanatory Graph

Experiments

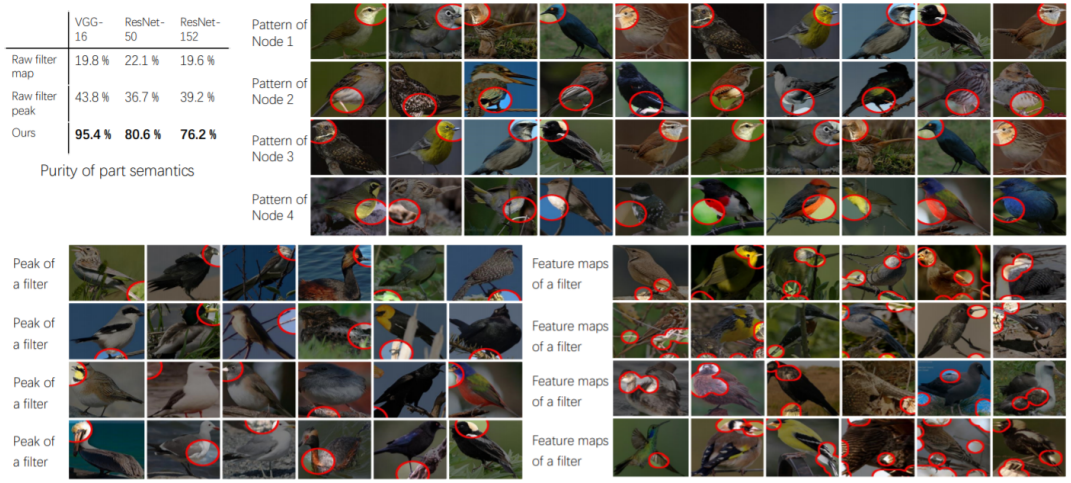


Figure 8: Purity of part semantics. We draw image regions corresponding to each node in an explanatory graph and image regions corresponding to each pattern learned by other methods (we show some examples on the right). We use human users to annotate the semantic purity of each node/pattern. Cyan boxes show inference results that do not describe the common part.

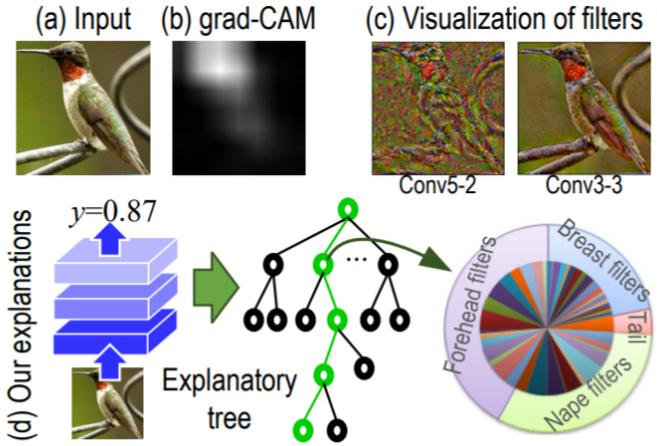
Figure 9: Notation for the computation of location instability.

	ResNet-50	ResNet-152	VGG-16	VAE-GAN
Raw filter (Zhou et al. 2015)	0.1328	0.1346	0.1398	0.1944
Ours	**0.0848**	**0.0858**	**0.0638**	**0.1066**
(Singh, Gupta, and Efros 2012)	0.1341			
(Simon, Rodner, and Denzler 2014)	0.2291			

Table 1: Location instability of patterns.

Interpreting CNNs via Decision Trees

- Bridging middle-layer features with semantic concepts.
- Bridging middle-layer features with final CNN predictions.



Define **Rationale** of a CNN prediction as the set of object parts (or filters) that are activated and contribute to the prediction.

3

[3] Quanshi Zhang et al. "Interpreting CNNs via Decision Trees". In: *CoRR* abs/1802.00121 (2018). arXiv: 1802.00121. URL: http://arxiv.org/abs/1802.00121.

Interpreting CNNs via Decision Trees

Image-specific rationale of a CNN prediction

Filter loss

$$\begin{aligned} \text{Loss}_f &= \sum_{x_f \in X_f} Loss_f(x_f) = -MI(X_f; P) \\ &= -\sum_{\mu \in P} p(\mu) \sum_{x_f \in X_f} p(x_f|\mu) \log \frac{p(x_f|\mu)}{p(x_f)} \end{aligned}$$

$P = \{\mu|\mu = [h, w], 1 \leq h, w \leq L\} \cup \{\emptyset\}$ refer to as a set of all part-location candidates.

Interpreting CNNs via Decision Trees

Quantitative rationales of CNN predictions

We can use a piece-wise linear representation to represent the function of cascaded FC layers and ReLU layers, as follows:

$$y = \sum_{h,w,d} g^{h,w,d} \cdot x^{h,w,d} + b.$$

Thus, use g, x to represent prediction rationales.

Interpreting CNNs via Decision Trees

Learning a decision tree

Algorithm 1 Learning a decision tree for a category

Input: 1. A CNN with disentangled filters, 2. training images $\mathbf{\Omega} = \mathbf{\Omega}^+ \cup \mathbf{\Omega}^-$.
Output: A decision tree.
Initialize a tree $Q = P_0$ and set $t = 0$
for each image $I_i, i \in \mathbf{\Omega}^+$ **do**
  Initialize a child of the root of the initial tree Q by setting $\overline{\mathbf{g}} = \mathbf{g}_i$ based on Equation (3) and $\boldsymbol{\alpha} = \mathbf{1}$.
end for
for $t = t + 1$ **until** $\Delta \log E \leq 0$ **do**
  1. Choose (v, v') in the second tree layer of P_{t-1} that maximize $\Delta \log E$ based on Equation (8)
  2. Merge (v, v') to generate a new node u based on Equations (5) and (6), and obtain the tree P_t.
end for
Assign filters with semantic object parts to obtain $\mathbf{A}$.

Interpreting CNNs via Decision Trees

Objective function

The **decision mode** is given as:

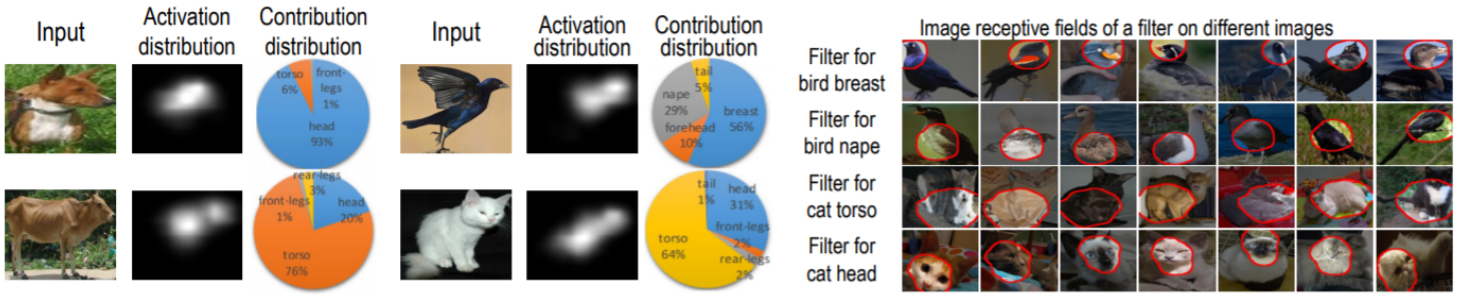
$$h_v(x_i) = w^T x_i + b, w = \alpha \circ \bar{g}$$
$$\max_{\bar{g}} \sum_{i \in \Omega_v} cosine(g_i, \bar{g}), s.t. \bar{g}^T \bar{g} = 1$$
$$\min_{\alpha, b} \frac{1}{||\Omega_v||} \sum_{i \in \Omega_v} (w^T x_i + b - y_i)^2 + \lambda ||\alpha||_1.$$

w is the rationale of the decision mode. They gradually revise the initial tree $Q = P_0$ towards the final tree after T merging operations. Thus formulate the objective for learning as follows:

$$\max_P E, \qquad E = \underbrace{\frac{\prod_{i \in \Omega^+} P(\mathbf{x}_i)}{\prod_{i \in \Omega^+} Q(\mathbf{x}_i)}}_{\text{Discrimination power}} \cdot \underbrace{e^{-\beta \|V\|}}_{\substack{\text{Sparsity of} \\ \text{decision modes}}}$$

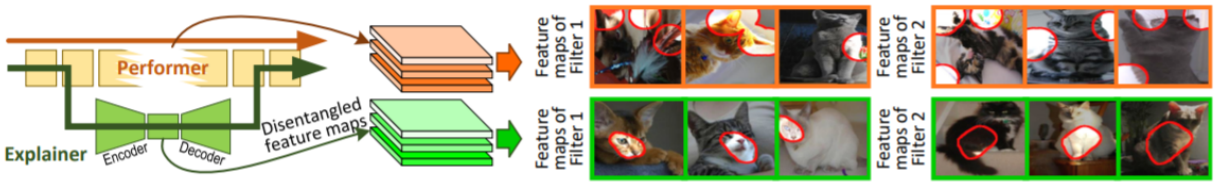
Interpreting CNNs via Decision Trees

Experiments



Unsupervised Learning of Neural Networks to Explain Neural Networks

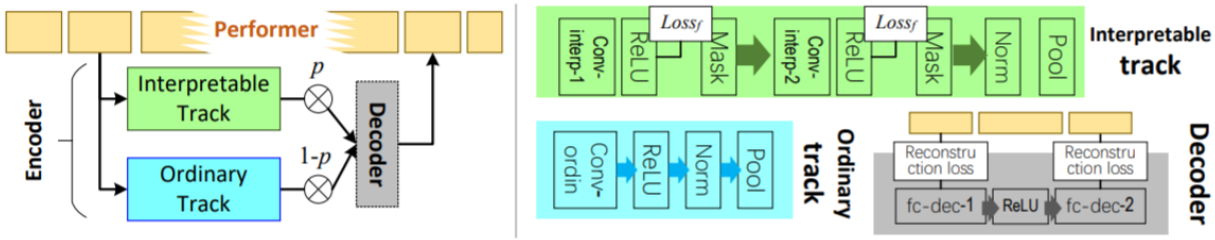
High interpretability is not necessarily equivalent to, and sometimes conflicts with a high discrimination power. Therefore, they propose to learn an additional neural network, namely an explainer network, to explain features inside the CNN.



4

[4] Quanshi Zhang et al. "Unsupervised Learning of Neural Networks to Explain Neural Networks". In: *CoRR* abs/1805.07468 (2018). arXiv: 1805.07468. URL: http://arxiv.org/abs/1805.07468.

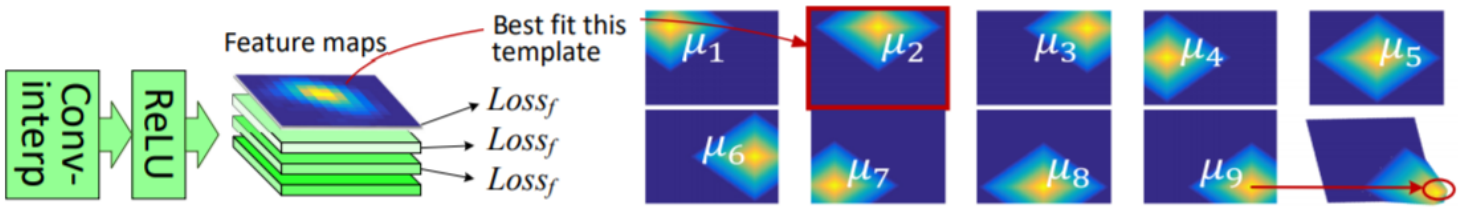
Explainer structure



They sum up output features of the interpretable track x_{interp} and those of the ordinary track x_{oridn} as the final output of the encoder,
$x_{enc} = p \cdot x_{interp} + (1 - p) \cdot x_{oridn}$.

Objective function

$$\sum_{l\in L}\lambda_l||x_l - x_l^*||^2 + \sum_f \lambda_f Loss_f(x_f) - \eta \log p$$



Filter loss: the minus mutual information between feature maps and a set of pre-defined templates.

$$\text{Loss}_f = \sum_{x_f\in X} Loss_f(x_f) = -MI(X;T)$$
$$= -\sum_{T\in \mathbf{T}} p(T) \sum_{x_f\in X} p(x_f|T) \log \frac{p(x_f|T)}{p(x_f)}$$

Experiments

Table 1: Location instability of feature maps in performers and explainers for the evaluation of filter interpretability. Please see the appendix for comparisons with more baselines.

Results based on the Pascal-Part dataset [4]

	Single-category							Multi-category
	bird	cat	cow	dog	horse	sheep	Avg.	Avg.
AlexNet	0.153	0.131	0.141	0.128	0.145	0.140	0.140	–
Explainer	**0.104**	**0.089**	**0.101**	**0.083**	**0.098**	**0.103**	**0.096**	–
VGG-M	0.152	0.132	0.143	0.130	0.145	0.141	0.141	0.135
Explainer	**0.106**	**0.088**	**0.101**	**0.088**	**0.097**	**0.101**	**0.097**	**0.097**
VGG-S	0.152	0.131	0.141	0.128	0.144	0.141	0.139	0.138
Explainer	**0.110**	**0.085**	**0.098**	**0.085**	**0.091**	**0.096**	**0.094**	**0.107**
VGG-16	0.145	0.133	0.146	0.127	0.143	0.143	0.139	0.128
Explainer	**0.095**	**0.089**	**0.097**	**0.085**	**0.087**	**0.089**	**0.090**	**0.109**

CUB200-2011 dataset [33]

AlexNet	0.1502
Explainer	**0.0906**
VGG-M	0.1476
Explainer	**0.0815**
VGG-S	0.1481
Explainer	**0.0704**
VGG-16	0.1373
Explainer	**0.0490**

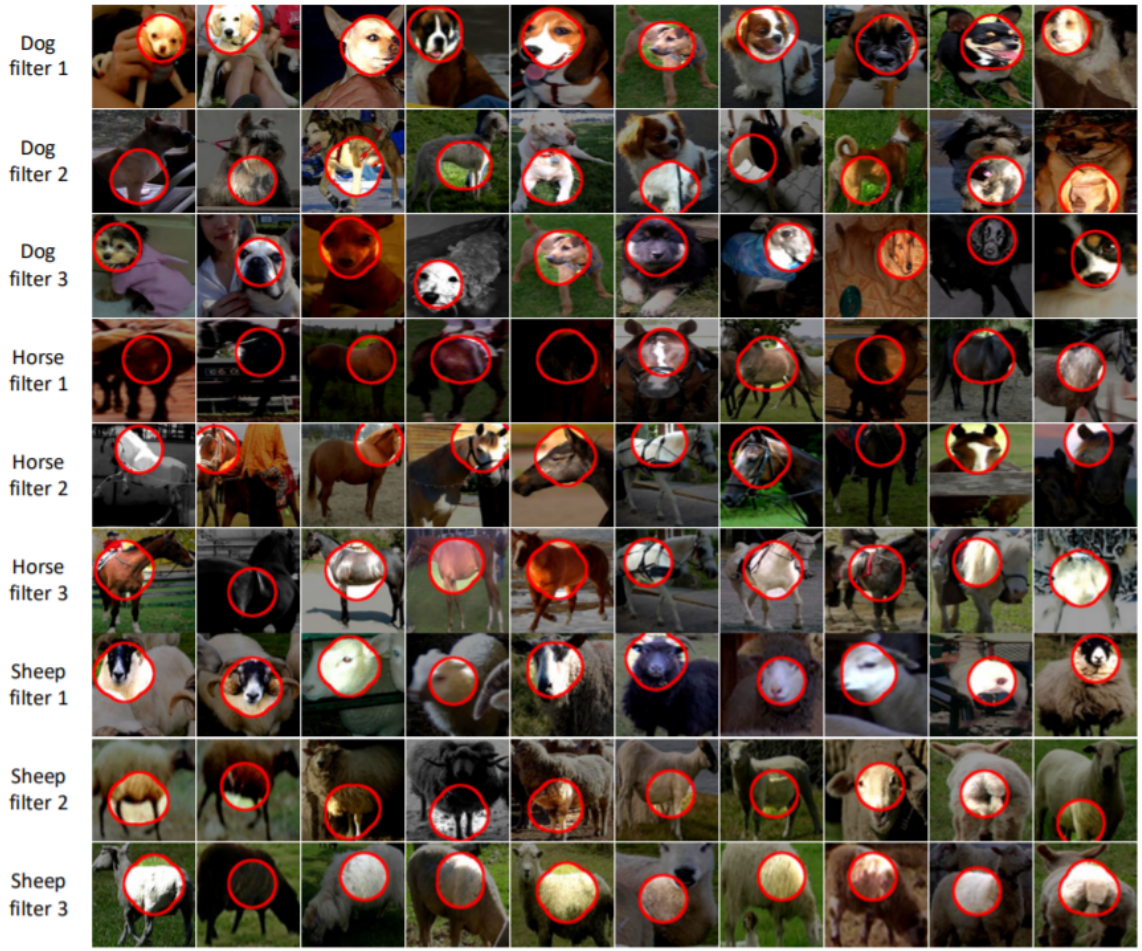
Table 2: Average p values of explainers in different experiments.

	Pascal-Part [4]		CUB200
	Single	Multi	-2011 [33]
AlexNet	–	0.7137	0.5810
VGG-M	0.9012	0.8066	0.8611
VGG-S	0.9270	0.8996	0.9533
VGG-16	0.8593	0.8718	0.9579

Table 3: Multi-category classification errors using features of performers and explainers based on the Pascal-Part dataset [4]. We evaluated loss of feature information in explainers based on the classification-error gap between performers and explainers. Please see the appendix for more results.

	Performer	Explainer	Δ Error	Performer+cls	Δ Error
VGG-M	6.12%	6.62%	0.5%	5.22%	-0.9%
VGG-S	5.95%	6.97%	1.02%	5.43%	-0.52%
VGG-16	2.03%	2.17%	0.14%	2.49%	0.46%

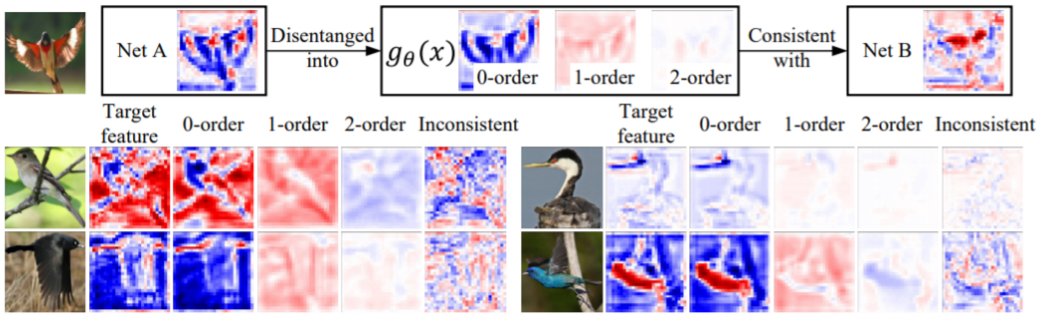
Experiments



Knowledge Consistency between Neural Networks and Beyond

Given two DNNs pre-trained for the same task, regardless of architectures, they aim to examine whether intermediate layers of the two DNNs encode similar visual concepts.

Knowledge as the set of visual concepts that are encoded by features of an intermediate layer. The well learned DNNs are supposed to converge to similar knowledge representations.



5

[5] Ruofan Liang et al. *Knowledge Consistency between Neural Networks and Beyond*. 2019. arXiv: 1908.01581 [cs.LG].

Knowledge Consistency between Neural Networks and Beyond

Let A and B denote two DNNs learned for the same task. x_A and x_B denote two intermediate-layer features of A and B, Features x_A and x_B can be decomposed as $x_A = \hat{x}_A + \epsilon_A$ and $x_B = \hat{x}_B + \epsilon_B$, $\hat{x}_A$ and $\hat{x}_B$ are triggered by same image regions. Thus, $\hat{x}_A$ and $\hat{x}_B$ are termed **consistent features** between A and B. ϵ_A and ϵ_B are independent with each other, and they are termed **inconsistent features**.
Usually, consistent components represent common and reliable knowledge. Whereas, inconsistent components mainly represent unreliable knowledge or noises.

Knowledge Consistency between Neural Networks and Beyond

Fuzzy consistency at different levels (orders):

Use non-linear transformations during feature reconstruction to approximate the fuzziness. Therefore, they propose a model g_k for feature reconstruction. Then, consider $\hat{x}_A = g_k(x_B)$ to represent consistent knowledge *w.r.t.* the DNN B at the k-th fuzziness level (or the k-th order).

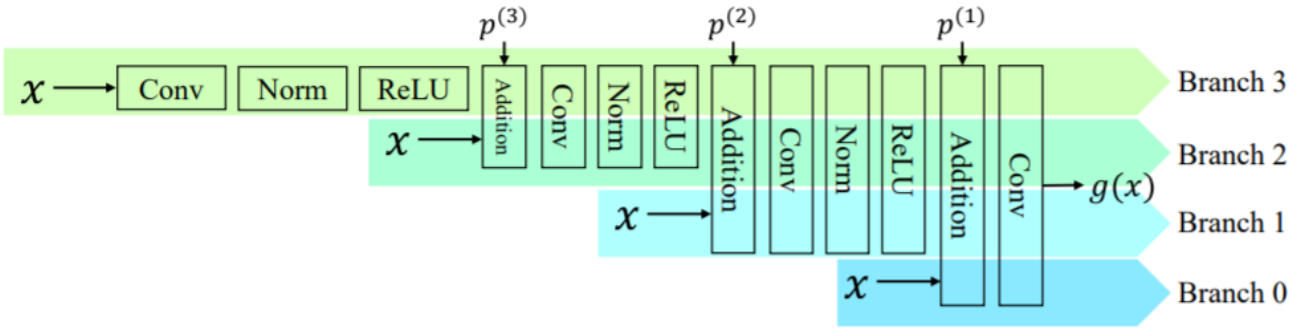
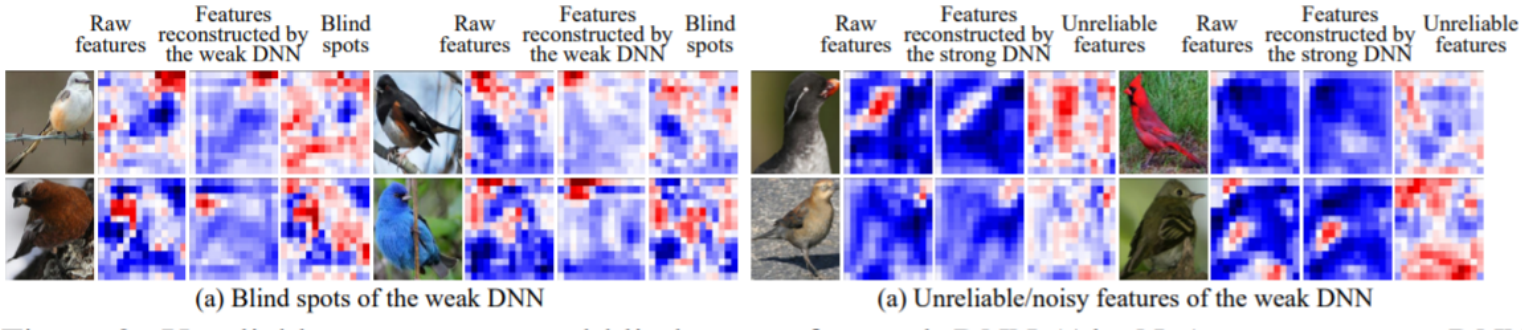


Figure 2: Neural network for disentanglement of consistent features ($K = 3$).

Knowledge Consistency between Neural Networks and Beyond

Comparative Studies

Network diagnosis based on knowledge consistency Use a strong (well learned) DNN to diagnose representation flaws hidden in a weak DNN.



(a) Blind spots of the weak DNN

(a) Unreliable/noisy features of the weak DNN

Unreliable features: feature components, which cannot be reconstructed by features of the strong DNN.

Blind spots feature components in the strong DNN, which are inconsistent with features of the weak DNN. These feature components usually reflect blind spots of the knowledge of the weak DNN.

Knowledge Consistency between Neural Networks and Beyond

Stability of learning

Examining whether or not all DNNs represent the same knowledge, when people repeatedly learn multiple DNNs for the same task.

Learning DNNs from different initializations

conv4 @ AlexNet	conv5 @ AlexNet	conv4-3 @ VGG-16	conv5-3 @ VGG-16	last conv @ ResNet-34
0.086	0.116	0.124	0.196	0.776

Learning DNNs using different training data

conv4 @ AlexNet	conv5 @ AlexNet	conv4-3 @ VGG-16	conv5-3 @ VGG-16	last conv @ ResNet-34
0.089	0.155	0.121	0.198	0.275

Table 1: Instability of learning DNNs from different initializations and instability of learning DNNs using different training data. Without a huge training set, networks with more layers (*e.g.* ResNet-34) usually suffered more from the over-fitting problem.

	conv4 @ AlexNet	conv5 @ AlexNet	conv4-3 @ VGG-16	conv5-3 @ VGG-16	last conv @ ResNet-34
$Var(x^{(0)})$	105.80	424.67	1.06	0.88	0.66
$Var(x^{(1)})$	10.51	73.73	0.07	0.03	0.10
$Var(x^{(2)})$	1.92	43.69	0.02	0.004	0.03
$Var(x^{\Delta})$	11.14	71.37	0.16	0.22	2.75

Table 2: Magnitudes of consistent features of different orders, when we train DNNs from different initializations. Features in different layers have significantly different variance magnitudes. Most neural activations of the feature belong to low-order consistent components.

Knowledge Consistency between Neural Networks and Beyond

Feature refinement

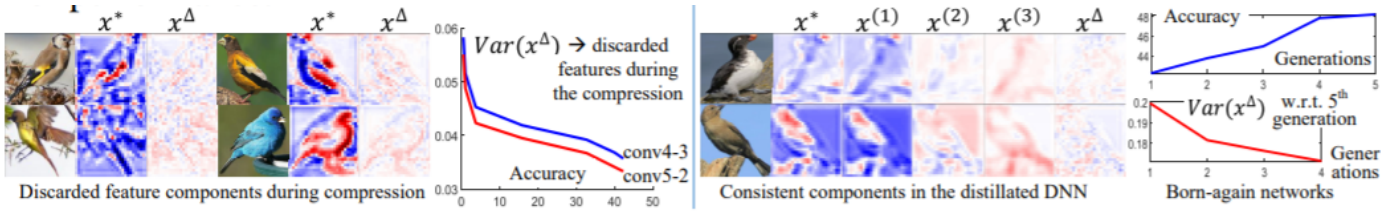
Intermediate-layer features can be refined by removing inconsistent components and exclusively using consistent components to accomplish the task.

	VGG-16 conv4-3	VGG-16 conv5-2	ResNet-18	ResNet-34	ResNet-50
Network A	43.15		34.74	31.05	29.98
Network B	42.89		35.00	30.46	31.15
$x^{(0)}$	**45.15**	**44.48**	38.16	31.49	30.40
$x^{(0)} + x^{(1)}$	44.98	44.22	**38.45**	31.76	31.77
$x^{(0)} + x^{(1)} + x^{(2)}$	45.06	44.32	38.23	**31.96**	**31.84**

	VGG-16 conv4-3			VGG-16 conv5-2		
	VOC-animal	Mix-CUB	Mix-Dogs	VOC-animal	Mix-CUB	Mix-Dogs
Features from the network A	51.55	44.44	15.15	51.55	44.44	15.15
Features from the network B	50.80	45.93	15.19	50.80	45.93	15.19
$x^{(0)} + x^{(1)} + x^{(2)}$	**59.38**	**47.50**	**16.53**	**60.18**	**46.65**	**16.70**
	ResNet-18			ResNet-34		
	VOC-animal	Mix-CUB	Mix-Dogs	VOC-animal	Mix-CUB	Mix-Dogs
Features from the network A	37.65	31.93	14.20	39.42	30.91	12.96
Features from the network B	37.22	32.02	14.28	35.95	27.74	12.46
$x^{(0)} + x^{(1)} + x^{(2)}$	**53.52**	**38.02**	**16.17**	**49.98**	**33.98**	**14.21**

Knowledge Consistency between Neural Networks and Beyond

Network compression and knowledge distillation



Hold me tight! Influence of discriminative features on deep network boundaries

- How is the margin in different directions related to the training features?
- How does the learning algorithm use the training samples to shape the decision boundaries?

6

[6] Guillermo Ortiz-Jimenez et al. *Hold me tight! Influence of discriminative features on deep network boundaries*. 2020. arXiv: 2002.06349 [cs.LG].

Hold me tight! Influence of discriminative features on deep network boundaries

Proposed framework

The decision boundary between classes k and l of a neural network is the set of points $\mathcal{B}_{k,l}(f) = \{x \in \mathbb{R}^D : f_k(x) - f_l(x) = 0\}$, and training set be $\tau = \{(x^i, y^i)\}_{i=0}^N$

Definition 1(Minimal adversarial perturbations).Given a classifier F, a sample $x \in \mathbb{R}^D$, and a sub-region of the input space $\mathcal{S} \subseteq \mathbb{R}^D$,we define the l_2 minimal adversarial perturbation of x in $\mathcal{S}$ as:

$$\delta_{\mathcal{S}}(x) = \arg\min_{\delta \in \mathcal{S}} ||\delta||_2, s.t.\ F(x+\delta) \neq F(x)$$

If $u \in \mathcal{S}^{D-1}$ denotes a unit-norm vector in $\mathcal{R}^D$, $\delta_u(x)$ means $\delta_{span\{u\}}(x)$

Definition 2 (Margin). The magnitude $||\delta_{\mathcal{S}}(x)||_2$ is the margin of x in $\mathcal{S}$.

Hold me tight! Influence of discriminative features on deep network boundaries

Invariance and discriminative features

The implicit bias of the training dynamics pushes the networks to become invariant along non-discriminative directions and construct boundaries only along discriminative ones.

Evidence on synthetic data

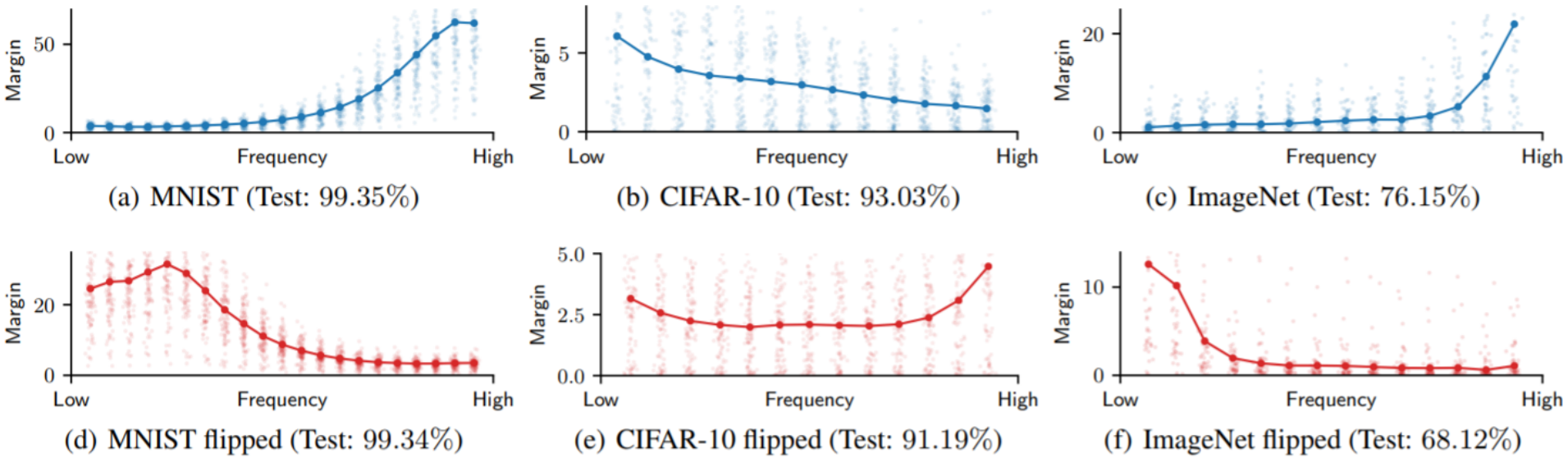
Table 1. Margin statistics of an MLP trained on $\mathcal{T}_1(\epsilon = 5, \sigma = 1, N = 10,000)$ along different directions ($M = 1,000$, $S = 3$).

	$\boldsymbol{u}_1$	$\text{span}\{\boldsymbol{u}_1\}^\perp$	$\mathcal{S}_{\text{ORTH}}$	$\mathcal{S}_{\text{RAND}}$
5-PERCENT.	1.74	4.85	30.68	17.21
MEDIAN	2.50	12.36	102.0	27.90
95-PERCENT.	3.22	31.60	229.5	80.61

Hold me tight! Influence of discriminative features on deep network boundaries

Invariance and discriminative features

Evidence on real data

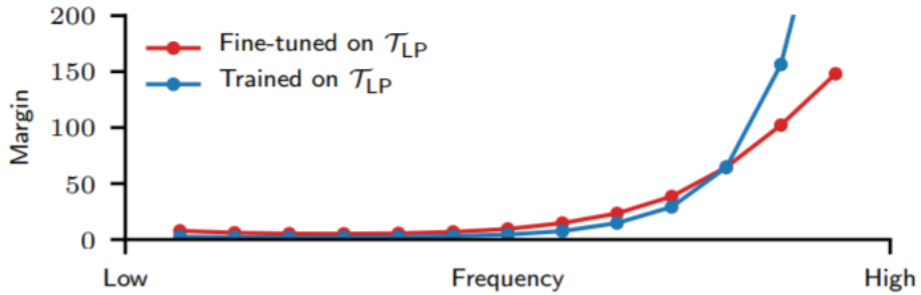


Hold me tight! Influence of discriminative features on deep network boundaries

Hypothesis 1. For any $z \in \mathcal{B}_{k,l}(f)$, there exists a unique set of discriminative directions $\mathcal{F}_{k,l}(z) \subseteq \mathbb{R}^D$. This is the smallest set spanned by a subset of all the pairwise differences between samples of class k and l in τ, such that there exists an $\epsilon > 0$ for which all x in an ϵ-ball around z satisfy:

$$||\delta_{\mathcal{F}_{k,l}(z)}(x)||_2 \ll ||\delta_{\mathbb{R}^D \setminus \mathcal{F}_{k,l}(z)}(x)||_2$$

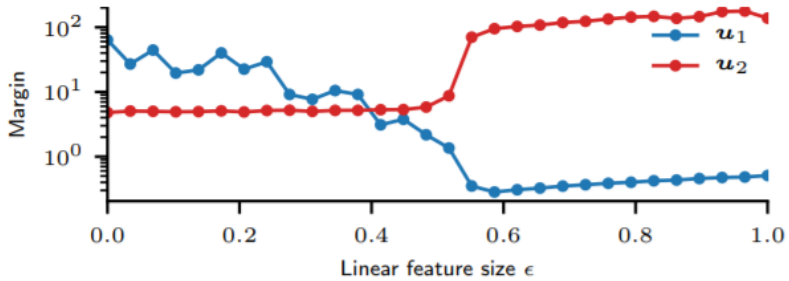
We then say that the classifier is locally invariant to the non-discriminative directions $\mathbb{R}^D \setminus \mathcal{F}_{k,l}(z)$.



Hold me tight! Influence of discriminative features on deep network boundaries

Sensitivity to position of training samples

Hypothesis 2. (Connectedness of training samples). The inductive bias of the learning algorithm has a tendency to build classifiers in which every pair of training samples with the same label belongs to the same decision region. If possible, connected by a straight path.



Hold me tight! Influence of discriminative features on deep network boundaries

Anchor points and decision boundaries

The missing piece of **Hypothesis 1** is the lack of an explicit definition of how the set of discriminative directions is constructed from the training set.

Conjecture (Anchors). For any point $z \in \mathcal{B}_{k,l}(f)$, there exists a minimal set of anchor samples $\mathcal{A}(z) \subseteq \tau$ such that

$$\exists N > 0 \ s.t. \ F(x^i; \bar{z}) = y^i, \forall (x^i, y^i) \in \mathcal{A}(z)$$

The set of discriminative directions $\mathcal{F}_{k,l}(z)$ in Hypothesis 1 is strictly generated by the pairwise differences between samples of class k and l in $\mathcal{A}(z)$. Besides, $|\mathcal{A}(z)| \ll |\tau|$.

Hold me tight! Influence of discriminative features on deep network boundaries

Anchor points and decision boundaries

Redundancy in training samples

Table 2. Test accuracy and boundary radius of an MLP trained on different combinations of concentric hyperspheres.

TRAIN. r	RADIUS	ACC. $r \in \{1,5\}$	ACC. $r \in \{3,4\}$
$\{1,3,4,5\}$	3.47	100.0%	99.99%
$\{1,5\}$	2.75	100.0%	54.8%
$\{3,4\}$	3.50	100.0%	100.0%

Sensitivity to few samples

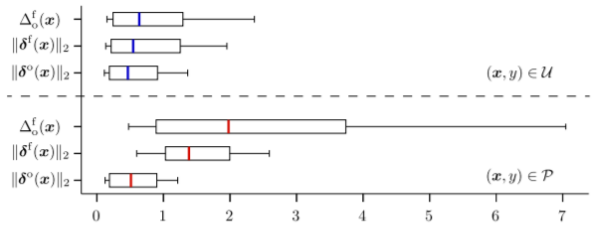
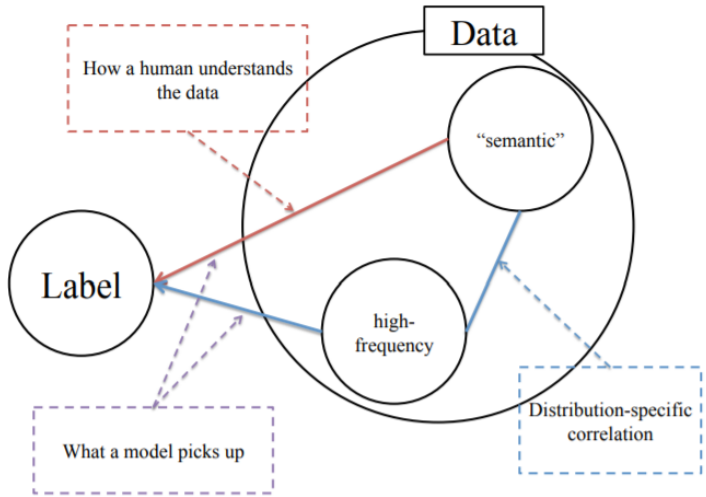


Figure 8. Margin distribution in different directions of a ResNet-18 trained on CIFAR-10 and fine-tuned on 100 DeepFool examples.

High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

Investigate the generalization behaviors of CNN from a data perspective. CNN can view the data at a much higher granularity than the human can. And CNN can exploit the high-frequency image components that are not perceivable to human.



7

[7] Haohan Wang et al. "High Frequency Component Helps Explain the Generalization of Convolutional Neural Networks". In: *CoRR* abs/1905.13545 (2019). arXiv: 1905.13545. URL: http://arxiv.org/abs/1905.13545

High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

CNN Exploit High-frequency Components

The raw data $x = \{x_l, x_h\}$, x_l and x_h the low-frequency component (shortened as LFC) and high-frequency component (shortened as HFC). $f(\cdot;\theta)$ denotes a convolutional neural network whose parameters are denoted as θ. $\mathcal{H}$ denote a human model, thus $f(\cdot;\mathcal{H})$ denotes how human will classify the data $\cdot$. Therefore, we have

$$y = f(x;\mathcal{H}) = f(x_l;\mathcal{H})$$

But a CNN is trained with

$$\arg\min_\theta l(f(x;\theta), y) = \arg\min_\theta l(f(x_l, x_h;\theta), y)$$

So CNN may learn to exploit x_h to minimize the loss. As a result, CNN's generalization behavior appears unintuitive to a human.

High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

Rethinking Data before Rethinking Generalization

Hypothesis

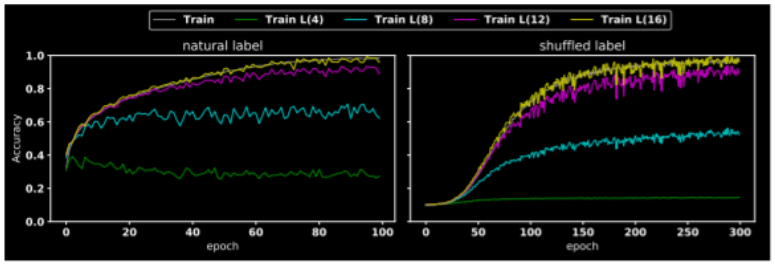
- In the original label case, the model will first pick up LFC, then gradually pick up the HFC to achieve higher training accuracy.
- In the shuffled label case, as the association between LFC and the label is erased due to shuffling, the model has to memorize the images when the LFC and HFC are treated equally.

Table 1. We test the generalization power of LFC and HFC by training the model with $\mathbf{x}_l$ or $\mathbf{x}_h$ and test on the original test set.

LFC			HFC		
r	train acc.	test acc.	r	train acc.	test acc.
4	0.9668	0.6167	4	0.9885	0.2002
8	0.9786	0.7154	8	0.9768	0.092
12	0.9786	0.7516	12	0.9797	0.0997
16	0.9839	0.7714	16	0.9384	0.1281

High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

Rethinking Data before Rethinking Generalization



High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

Comparison of Different Training Heuristics

Heuristics

- Batch Size
- Dropout
- Mix-up
- BatchNorm
- Adversarial Training

High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

Comparison of Different Training Heuristics

Heuristics

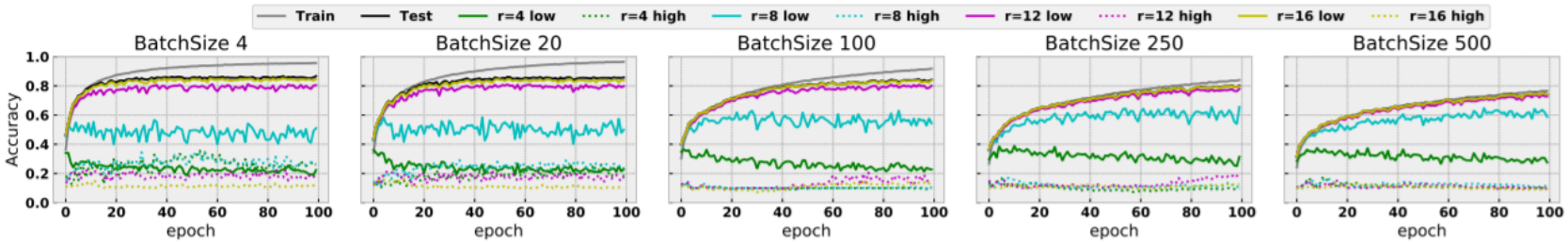


Figure 4. Plots of accuracy of different epoch sizes along the epoches for train, test data, as well as LFC and HFC with different radii.

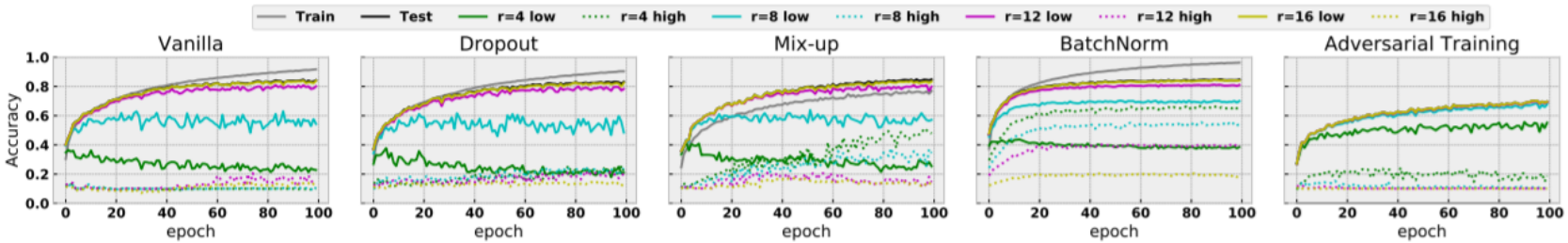


Figure 5. Plots of accuracy of different heuristics along the epoches for train, test data, as well as LFC and HFC with different radii.

High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

A hypothesis on Batch Normalization

Align the distributional disparities of different predictive signals. If one of BatchNorm's advantage is to encourage the model to capture different predictive signals, the performance gain of BatchNorm is the most limited when the model is trained with LFC when $r = 4$.

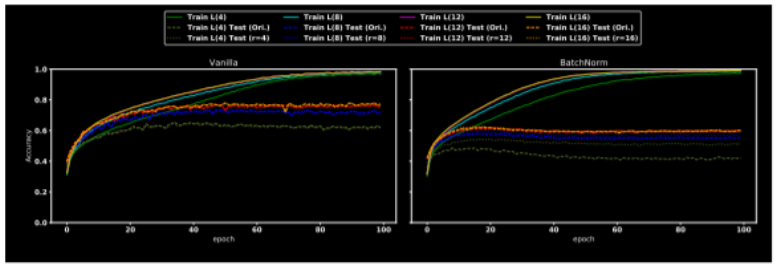


Figure 6. Comparison of models with vs. without BatchNorm trained with LFC data.

High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

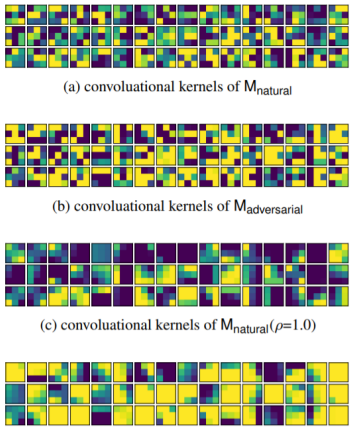
Adversarial Attack and Defense

Kernel Smoothness vs. Image Frequency

To push the model to ignore the HFC, one can consider to force the model to learn the convolutional kernels that have only negligible weights at the high-end of the frequency domain.

Robust Models Have Smooth Kernels

Smoothing Kernels Improves Adversarial Robustness



(a) convoluational kernels of $M_{natural}$

(b) convoluational kernels of $M_{adversarial}$

(c) convoluational kernels of $M_{natural}(\rho=1.0)$

On the Spectral Bias of Neural Networks

- **spectral bias**: a phenomenon that networks prioritize learn the low frequency modes.
- complex manifold shapes can facilitate the learning of higher frequencies.

8

[8] Nasim Rahaman et al. *On the Spectral Bias of Neural Networks*. 2018. arXiv: 1806.08734 [stat.ML].

On the Spectral Bias of Neural Networks

Theorem 1. The Fourier components of the ReLU network f_θ with parameters θ is given by the rational function:

$$\bar{f}_\theta(k) = \sum_{i=0}^{d} \frac{C_n(\theta,k)1_{H_n^\theta}(k)}{k^{n+1}}$$

where H_n^θ is the union of n-dimensional subspaces that are orthogonal to some n-codimensional faces of some polytope P_ϵ and $C_n(\cdot,\theta) : \mathbb{R}^d \longrightarrow \mathbb{C}$ is $\Theta(1)(k \longrightarrow \inf)$

Discussion.

- The spectral decay of ReLU networks is highly anisotropic in large dimensions. Almost all directions: k^{-d-1}, directions orthogonal to the $d-1$ dimensional faces: as slow as k^{-2}.
- The numerator is bounded by $N_f L_f$.

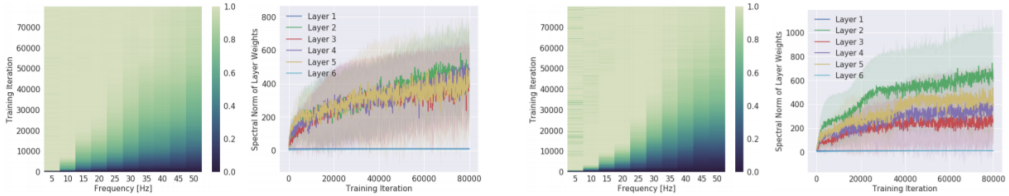
On the Spectral Bias of Neural Networks

Lower Frequencies are Learned First

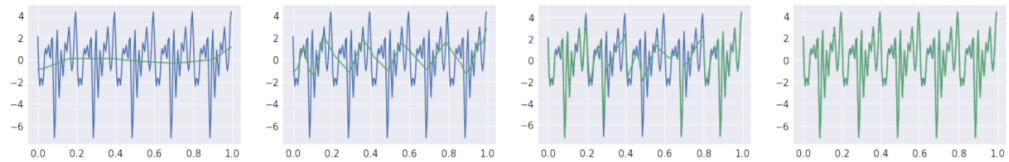
Synthetic Experiments

- **Experiment 1.**: consider the mapping $\lambda : [0, 1] \longrightarrow \mathbb{R}$ given by

$$\lambda(z) = \sum_i A_i \sin(2\pi k_i z + \phi_i)$$



(a) Equal Amplitudes (b) Increasing Amplitudes



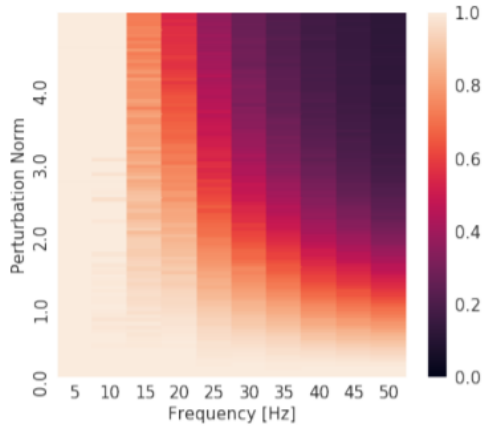
(a) Iteration 100 (b) Iteration 1000 (c) Iteration 10000 (d) Iteration 80000

On the Spectral Bias of Neural Networks

Lower Frequencies are Learned First

Synthetic Experiments

- **Experiment 2.**: lower frequencies are more robust to parameter perturbations.

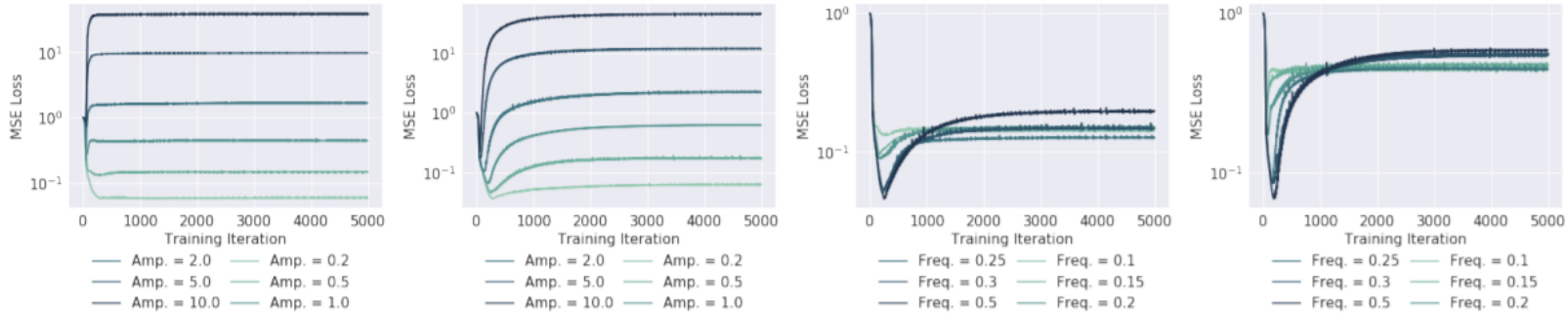


On the Spectral Bias of Neural Networks

Lower Frequencies are Learned First

Real-Data Experiments

- **Experiment 3.**: how the validation performance dependent on the frequency of noise added to the training target.



(a) $k = 0.1$ (b) $k = 1$ (c) $\beta = 0.5$ (d) $\beta = 1.$

On the Spectral Bias of Neural Networks

Not all Manifolds are Learned Equal

When the data lies on a lower dimensional manifold embedded in the higher dimensional input space of the model, the shape of the data-manifold impacts the learnability of high frequencies in a non-trivial way, because low frequency functions in the input space may have high frequency components when restricted to lower dimensional manifolds of complex shapes.

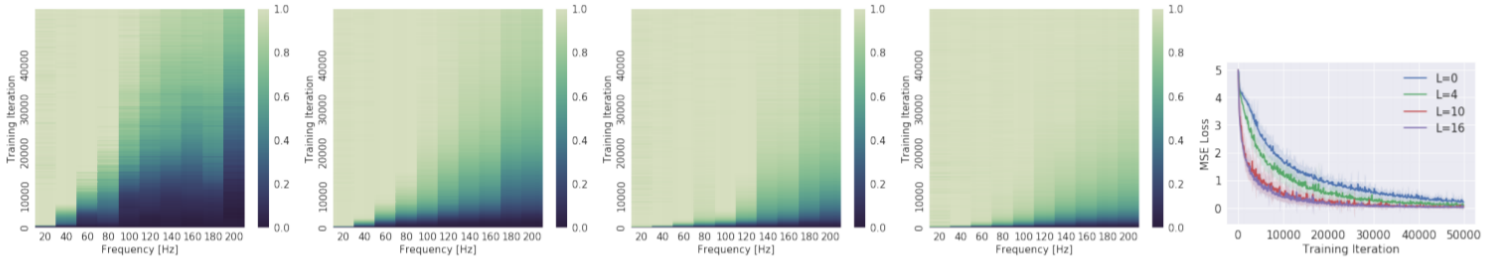
Manifold hypothesis. consider the image $\gamma([0,1]^m)$ of some injective mapping $\gamma : [0,1]^m \longrightarrow \mathbb{R}^d$ defined on a lower dimensional latent space $[0,1]^m$. A target function $\tau : \mathcal{M} \longrightarrow \mathbb{R}^d$ defined on the data manifold can be identified with a function $\lambda = \tau \circ \gamma$ defined on the latent space. Regressing τ is therefore equivalent to finding $f : \mathbb{R}^d \longrightarrow \mathbb{R}$. The mean square error can be expressed as:

$$\mathrm{MSE}^x_\mu[f,\tau] = \mathbb{E}_{x\sim\mu}|f(x)-\tau(x)|^2 = \mathbb{E}_z|f(\gamma(z))-\lambda(z)|^2 = \mathrm{MSE}^z_U[f\circ\gamma,\lambda].$$

On the Spectral Bias of Neural Networks

Not all Manifolds are Learned Equal

- **Experiment 4.**: attenuation of the spectral bias as L grows and the larger the L, the easier the learning task.

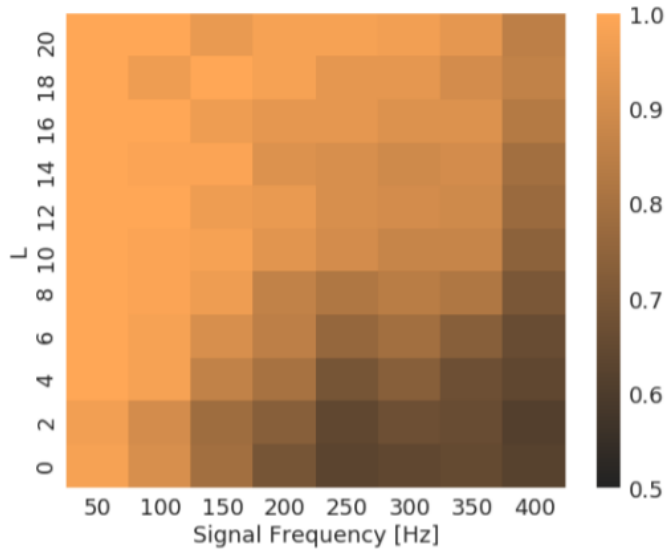


(a) $L = 0$ (b) $L = 4$ (c) $L = 10$ (d) $L = 16$ (e) Loss curves

On the Spectral Bias of Neural Networks

Not all Manifolds are Learned Equal

- **Experiment 5.**: increasing L results in better (classification) performance for the same target signal regardless of loss function (BCE instead of MSE).



What Information dose a ResNet Compress?

An empirical study of the relevance of the Information Bottleneck principle in realistic large-scale settings.

Information Bottleneck interpretation of deep learning: an optimal representation h exists between input data, x, and target data, y, that captures all relevant components in x about y, which should retain only the information relevant for the task described by y. By studying the information plane $I(x;h)$ and $I(y;h)$ changing over time –NNs have two learning phases:

- Fitting, or empirical error minimisation, where the information shared between hidden representations and input data was maximised.
- Compression, where the the information shared between hidden representation and input data was minimised but constrained to the classification task at hand.

9

[9] Luke Nicholas Darlow and Amos Storkey. *What Information Does a ResNet Compress?* 2020. arXiv: 2003.06254 [cs.LG].

What Information dose a ResNet Compress?

Mutual Information Lower Bound Computation

They derive a lower bound on the MI between two random vectors as follows:

$$
\begin{aligned}
I(x;h) &= \mathbb{E}_{P_D(x,h)}[\log \frac{p_D(x|h)q(x|h)}{q(x|h)}] - C \\
&= \mathbb{E}_{P_D(x,h)}[\log q(x|h)] + \mathbb{E}_{P_D(x,h)}[D_{KL}(p(x|h)||q(x|h))] - C \quad \text{With} \\
&\geq \mathbb{E}_{P_D(x,h)}[\log q(x|h)] - C.
\end{aligned}
$$

sufficiently large data (of size N) we can estimate:

$$\mathbb{E}_{P_D(x,h)}[\log q(x|h)] \simeq \frac{1}{N}\sum_{x^i,h^i} \log q(x^i|h^i).$$

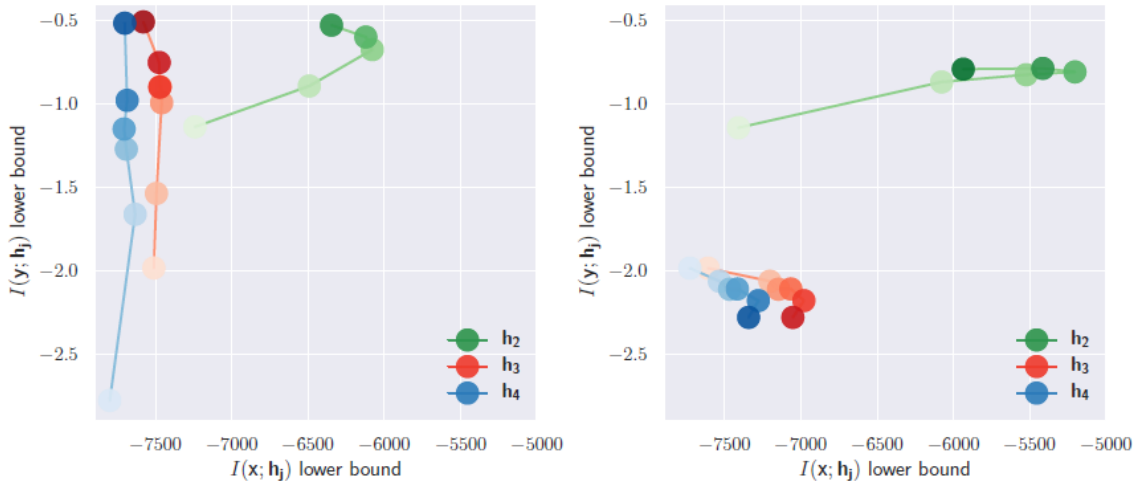
The task now becomes defining the decoder models $q(x|h)$ and $q(y|h)$ to estimate the MI.

What Information dose a ResNet Compress?

Experiments

They define four hidden layers for which they track the h_1, h_2, h_3, h_4, h_4 is the penultimate layer and is therefore expected to be most compressed. None of these layers have skip connections over them. The autoencoder's bottleneck is h_4.

MI Tracking

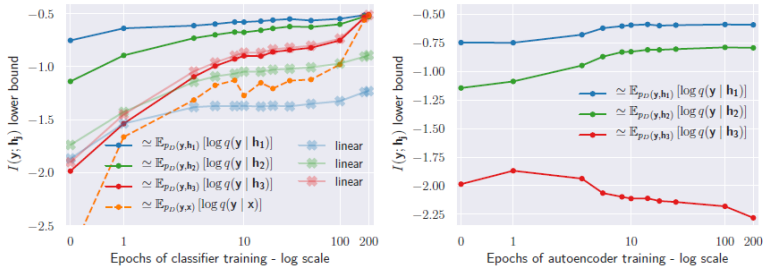


(a) Classification (b) Autoencoding

What Information dose a ResNet Compress?

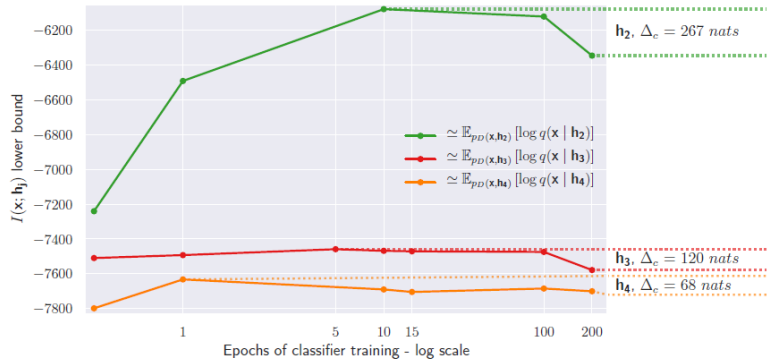
Experiments

MI Tracking



(a) Forward MI tracking, classification

(b) Forward MI tracking, autoencoding

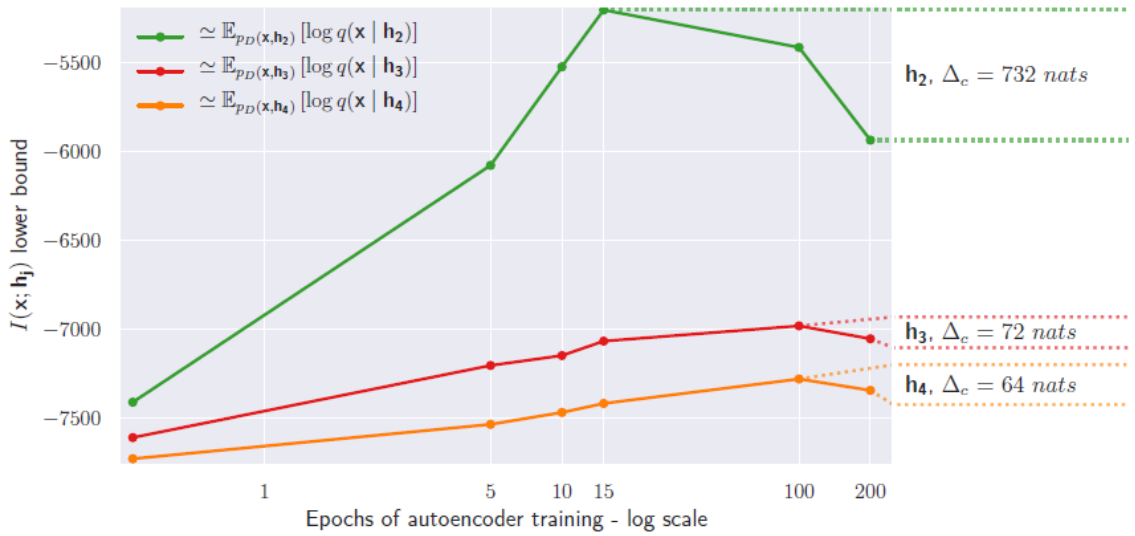


(c) Inverse MI tracking, classifier

What Information dose a ResNet Compress?

Experiments

MI Tracking

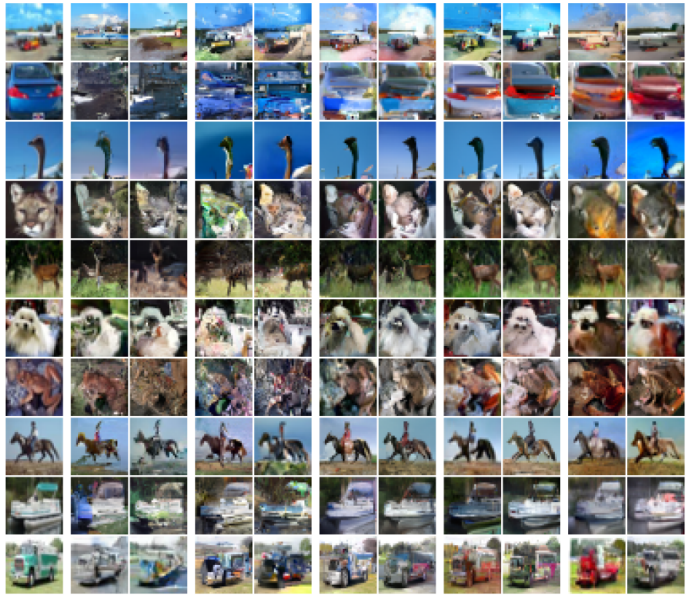


(d) Inverse MI tracking, autoencoding

What Information dose a ResNet Compress?

Experiments

Conditionally Generated PixelCNN++ Samples To illustrate the type of information kept and discarded by a ResNet classifier.



(a) (b) No learning (c) 1 Epoch (d) 10 Epochs (e) 100 Epochs (f) 200 Epochs

The network learns to keep the useful information. In contrast to this observation we note that not all irrelevant information is discarded.

Explainable Machine Learning in Deployment

This study explores how organizations view and use explainability for stakeholder consumption. They interview about **50** individuals from **30** approximately thirty organizations including 2 groups: data scientists not currently using explainable machine learning and individuals working for organizations and deploying explainable machine learning in practice.

10

[10] Umang Bhatt et al. *Explainable Machine Learning in Deployment*. 2019. arXiv: 1909.06342 [cs.LG].

Explainable Machine Learning in Deployment

Summary of Findings

- Explainability Needs
  - Model performance debugging
  - Model monitoring
  - Model transparency
  - Model audit
- Explainability Usage
- Key Takeaways

Explainable Machine Learning in Deployment

Deploying Local Explainability

- Feature Importance
  - Feature importance is not shown to end users, but is used by machine learning engineers as a sanity check.
  - Heatmaps are hard to aggregate, which makes it hard to do false positive detection at scale.
  - Spurious correlations can be detected with simple gradient-based techniques.
- Counterfactual Explanations
  - Organizations are interested in counterfactual explanation solutions since the underlying method is flexible and such explanations are easy for end users to understand.
  - It is not clear exactly what should be optimized for when generating a counterfactual or how to do it efficiently. Still, approximate solutions may suffice in practical applications.

Explainable Machine Learning in Deployment

Deploying Local Explainability

- Adversarial Training
  - There is a relation between model robustness and explainability. Model robustness improves the quality of feature importances (specifically saliency maps)
  - Feature importance helps find minimal adversarial perturbations for language models in practice
- Influential Samples
  - Influence functions can be intractable for large datasets; as such, a significant effort is needed to improve these methods to make them easy to deploy in practice.
  - Influence functions can be sensitive to outliers in the data, such that they might be more useful for outlier detection than for providing end users explanations.

Reference I

[1] Yash Goyal, Uri Shalit, and Been Kim. "Explaining Classifiers with Causal Concept Effect (CaCE)". In: *CoRR* abs/1907.07165 (2019). arXiv: 1907.07165. URL: http://arxiv.org/abs/1907.07165.

[2] Quanshi Zhang et al. "Interpreting CNN knowledge via an Explanatory Graph". In: *CoRR* abs/1708.01785 (2017). arXiv: 1708.01785. URL: http://arxiv.org/abs/1708.01785.

[3] Quanshi Zhang et al. "Interpreting CNNs via Decision Trees". In: *CoRR* abs/1802.00121 (2018). arXiv: 1802.00121. URL: http://arxiv.org/abs/1802.00121.

[4] Quanshi Zhang et al. "Unsupervised Learning of Neural Networks to Explain Neural Networks". In: *CoRR* abs/1805.07468 (2018). arXiv: 1805.07468. URL: http://arxiv.org/abs/1805.07468.

Reference II

[5] Ruofan Liang et al. *Knowledge Consistency between Neural Networks and Beyond*. 2019. arXiv: 1908.01581 [cs.LG].

[6] Guillermo Ortiz-Jimenez et al. *Hold me tight! Influence of discriminative features on deep network boundaries*. 2020. arXiv: 2002.06349 [cs.LG].

[7] Haohan Wang et al. "High Frequency Component Helps Explain the Generalization of Convolutional Neural Networks". In: *CoRR* abs/1905.13545 (2019). arXiv: 1905.13545. URL: http://arxiv.org/abs/1905.13545.

[8] Nasim Rahaman et al. *On the Spectral Bias of Neural Networks*. 2018. arXiv: 1806.08734 [stat.ML].

[9] Luke Nicholas Darlow and Amos Storkey. *What Information Does a ResNet Compress?* 2020. arXiv: 2003.06254 [cs.LG].

Reference III

[10] Umang Bhatt et al. *Explainable Machine Learning in Deployment*. 2019. arXiv: 1909.06342 [cs.LG].

Thanks for listening!