CLASS-WISE VISUAL EXPLANATIONS FOR DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many explainable AI (XAI) methods have been proposed to interpret neural network's decisions on why they predict what they predict locally through gradient information. Yet, existing works mainly for local explanation lack a global knowledge to show class-wise explanations in the whole training procedure. To fill this gap, we proposed to visualize global explanation in the input space for every class learned in the training procedure. Specifically, our solution finds a representation set that could demonstrate the learned knowledge for each class. To achieve this goal, we optimize the representation set by imitating the model training procedure over the full dataset. Experimental results show that our method could generate class-wise explanations with high quality in a series of image classification datasets. Using our global explanation, we further analyze the model knowledge in different training procedures, including adversarial training, and noisy label learning. Moreover, we illustrate that the generated explanations could lend insights into diagnosing model failures, such as revealing triggers in a backdoored model.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved unprecedented success over various tasks across different domains. However, there is still limited understanding of how to interpret their decisions due to their black-box nature. In other words, current DNNs lack the desirable transparency ability to explain why they predict what they predict. "Correct" prediction without reasonable explanation from DNNs may bring a huge concern on their prediction's reliability, especially in security-sensitive tasks such as medical image analysis (Nwadike et al., 2020), auto-pilot in the self-driving system (Wang et al., 2021) etc.

To acquire a better understanding of model prediction, many attempts have been made to interpret DNNs from different perspectives. A major focus have been put on the local explanations, either through the saliency map (Simonyan et al., 2013; Smilkov et al., 2017; Adebayo et al., 2018) and unique identification factors (Zhou et al., 2016; Selvaraju et al., 2017), or extracting explainable attributions on models' decision for each input example (Ribeiro et al., 2016; Hinton et al., 2015; Frosst & Hinton, 2017). These techniques have a good ability to visualize feature representation utilized for debugging trained DNNs in a local manner. Yet, local explanations can only interpret resulting model decisions for a particular input sample, being unable to visualize any intermediate attributions or model knowledge learned in a whole training procedure. Thus, a global explanation for viewing overall training logic is desirable, i.e., visualizing intermediate explanations for verifying commonly-used hypotheses in different training steps. Apart from that, with respect to diagnosing model prediction, local explanations are further challenged by the recent progress in backdoor attacks (Zhao et al., 2021). DNNs have been shown vulnerable to backdoor attacks. However, local explanations fail to provide helpful (intermediate) explanations for analyzing backdoor patterns given a target label. To our best knowledge, no prior work can reveal complex backdoor patterns when visualizing a whole training process for DNNs.

Existing global explanations methods focus on extracting rules from DNN models (Hara & Hayashi, 2018; Lakkaraju et al., 2017) or distilling black-box models into small and easy-to-explained substitute models (Tan et al., 2019; Bastani et al., 2017). However, the quality of generated explanations highly depends on the complexity of the designed rules and the substitute model. Besides, the substitute model is a approximation of the explained model, which could also introduce extra biases to explanations.

Our solution. This paper gives a generic method for the global visual explanation, given an inputting class-wise dataset and training procedure only, without requiring any additional hypotheses. Our global explanation can be applied to visualize the whole training process of any DNNs, finally providing much richer visual explanations and knowledge than local explanations. Unlike prior global explanations, our method requires neither extracting rules from a black-box model nor distill knowledge to a simpler model. Instead, it visualizes learned knowledge directly over representative data to attain more comprehensive explanations, rather than processing a simplified model summarized from original data. Representative data, in the pixel space rather than feature space, enables debugging a model and studying hidden mechanisms more easily in a training procedure. Intrinsically, our superior results benefit from shortening the route of generating visual explanation, saying, distilling knowledge directly from data instead of from a model derived from data.

To be specific, given a training dataset and corresponding training procedure, we optimize representative data points so that the learned model over representative data could approach the same one trained with the full dataset. By adding more reference snapshots in the training trajectory of model parameters, our method supports extracting a high-quality visualization to reveal global class-wise patterns learned by specifying various training manners. Empirical results on datasets show desirable class-wise explanations with high fidelity in Sec. 4. Besides, Sec. 5 analyzes debugging model failure (i.e.,inserted backdoor) that exists in the training procedure. To demonstrate generality, we utilize our class-wise explanation to analyze feature representation learned by various training procedures in Sec. 6, including training dynamics, adversarial training, and noisy label learning.

Our contributions are summarized below:

- We propose a global visual explanation method in the input space to reveal the key representation points learned with respect to different classes. Our method could generate a class-discriminative, natural-looking, and high-quality visual explanation.
- We show the proposed method could help in diagnosing several model failures, including backdoor attacks where the local explanation is completely failed.
- We show a proof-of-concept of how to understand the model knowledge in different training phases and different training methods. This is critical for understanding the generalization of deep learning and sheds light on how to design a better training algorithm.

2 Related work

2.1 Local explanation methods

Local explanation methods aim to help understand the decision procedure for a specific sample. At the pixel-level, saliency map (Simonyan et al., 2013) is the first to identify the sensitivity of each pixel towards the final prediction by highlighting the largest score in the calculated gradients. The vanilla gradient method is later improved via smoothed gradient (Smilkov et al., 2017), and guided back-propagation (Adebayo et al., 2018). Class Activation Mapping (CAM) (Zhou et al., 2016) visualizes discriminative regions by simplifying the model into one without any fully-connected layer. Later, Grad-CAM (Selvaraju et al., 2017) then makes CAM without altering model architecture by incorporating the gradient information. Other than directly pixel-level explanation, LIME (Ribeiro et al., 2016) approximates the model with a linear model locally, which relies on instances randomly generated in the neighborhood of the sample to be explained. After deriving the model, interpretable features are then projected back into the original feature space to get a final explanation. SHAP (Lundberg & Lee, 2017) re-formalizes additive feature attribution problem into a cooperative game and use Shapley value to assign each feature an important score for a particular prediction. Recently, a class-wise local explanation method has been proposed to visualize a sparse representation of model knowledge (Zhao et al., 2021). However, the method heavily depends on the given canvas (example). Although our method also focuses on the class-wise pattern, our global explanation method neither depends on a specific sample nor requires sparse constraints for generating representations in the input space.

2.2 Global explanation methods

Different from local explanation methods, global explanation aims to describe the overall logic of the black box, including how the model parameter affects the resulting prediction on average, and what and how the model has been learned in the training procedure. The global explanation is first to be formalized as extracting some rules from the black-box model to interpret the model (Bastani

et al., 2017). As neural networks become more complex and deep, it is then very hard to extract rules directly from the model. Model distillation (Hinton et al., 2015) is then applied to simplify complex black-box models to a smaller yet interpretable one with a similar performance to the original model. Neural networks have been distilled into trees structure model (Craven & Shavlik, 1995; Frosst & Hinton, 2017) and an additive model for the global explanation (Tan et al., 2018).

By selecting the prototypes or representative samples, example-based explanation methods (Kim et al., 2016; Gurumoorthy et al., 2019) could provide a condensed view of the whole training dataset and also select a subset of the whole training set as global explanations. However, the selected prototype set is always very large, containing thousands of samples. It's hard for users to directly obtain clear and concise explanations.Besides, example-based methods can only select existing samples from the training dataset. Therefore, those method are unable to represent the knowledge learned in various training paradigms like adversarial training. However, our method that only need a dozen examples to consist the global explanations could reflect knowledge from various training paradigms by directly generating explanations from trained models.

Activation maximization (AM) method (Olah et al., 2017; Yosinski et al., 2015; Nguyen et al., 2017; 2019) visualize the learned feature of various neurons of DNN models in the input space. It could also be used as a global explanation to visualize learned class-wise patterns of DNN models by maximizing output neuron of each class. However, directly taking maximization always generates less coherent and high-frequency local patterns. To generate the global coherent and natural visualizations, they need to be combined with hand-designed regularization like Gaussian blur, dropout, mean initialization, and deep generative models (Nguyen et al., 2019). Compared with AM method, our method can generate high-quality, global coherent, and diverse class-wise explanations without any prior regularization.

3 METHODOLOGY

Suppose $\boldsymbol{\theta}^R$ is the model trained using the dataset R with n samples, our goal is to find a global explanation set $\mathcal{S}$ that contains class-discriminative explanation for every class $\mathcal{S}_i$, for $i = 0, \ldots, C-1$. In the meantime, the explanation set should be much smaller than the original set i.e $|\mathcal{S}| \ll n$. In this section, unlike the previous works extracting the set from the model directly (Bastani et al., 2017), we propose a new method by synthesizing $\mathcal{S}$ such that the model trained on $\mathcal{S}$ should be equal to $\boldsymbol{\theta}^R$, since the model could be expressed as a function of representation samples for every class. Intuitively, this representation could be thought of as extracting critical points around the model's decision boundary. Let's take the support vector machine (SVM) as an example. The support vectors, whose size is much smaller than the size of the whole training set, that determine the SVM model could be thought of as the representation set we aim to extract [1].

3.1 SEARCHING CLASS-WISE EXPLANATION SET

For simplicity, we consider a classification model $f_{\boldsymbol{\theta}} : \mathcal{X}^{(m)} \rightarrow \mathcal{Y}^C$ which maps input $\boldsymbol{x}$ in the input space $\mathcal{X}$with m dimension input to a label $\boldsymbol{y}$ at the label space $\mathcal{Y}$ with C class and the model parameter is $\boldsymbol{\theta}$. Given a training set R consists of n instances $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$. Consider a non-negative real-valued loss function $\mathcal{L}$ that penalize the difference between the prediction $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ and true label $\boldsymbol{y}$ from an unknown data distribution $\mathcal{P}$, $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{P}$, we aim to find the model $\boldsymbol{\theta}^R$ as:

$$\boldsymbol{\theta}^R = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{R}(\boldsymbol{\theta}) = \int \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) dP(\boldsymbol{x}, \boldsymbol{y}) \approx \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) \tag{1}$$

We formulate searching the class-wise explanation set $\mathcal{S}$ as the below bi-level optimization problem:

$$\min_{\mathcal{S}} \mathcal{D}(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\theta}^R) \quad \text{s.t} \quad \boldsymbol{\theta}^{\mathcal{S}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \ \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}) := \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) \tag{2}$$

where $D(\cdot)$ is a distance metric to measure the distance between the model trained by the dataset R and representation set $\mathcal{S}$. We use the sum of the MSE loss and the cosine similarity in the implementation. In other words, we aim to find a set of critical representation for each class that let

[1] In Soudry et al. (2018), the authors theoretically proved that SGD implicitly converges to solutions that maximally separate the dataset. This also implies that some samples are more relevant than others to decision boundary learned by the model.

the model $\boldsymbol{\theta}^{\mathcal{S}}$ be trained on $\mathcal{S}$ is close to the original. By imitating model parameters θ^R, we could then extract class-discriminative features from the dataset R into $\mathcal{S}$.

Since the above problem (Eq.2) is a bi-level optimization problem, we can solve it using alternative minimization. Specifically, for each iteration of outer loop, we utilize gradient descent algorithm with a fixed number of iterations M to solve inner problem:

$$\boldsymbol{\theta}^{\mathcal{S}} = \boldsymbol{\theta}^{\mathcal{S}}_M = \text{OPT}(\boldsymbol{\theta}^{\mathcal{S}}_0, \mathcal{S}, M), \quad \text{the update: } \boldsymbol{\theta}^{\mathcal{S}}_{t+1} \leftarrow \boldsymbol{\theta}^{\mathcal{S}}_t - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}^{\mathcal{S}}_t) \tag{3}$$

where **OPT** means optimization process (gradient descent) with M iteration updates. $\boldsymbol{\theta}^{\mathcal{S}}_0$ is initialized with same initial point of $\boldsymbol{\theta}^R$. After obtaining $\boldsymbol{\theta}^{\mathcal{S}}_M$, we solve the outer problem by taking one step gradient descent:

$$\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t - \eta_{\mathcal{S}} \nabla_{\mathcal{S}} \mathcal{D}(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\theta}^R) \tag{4}$$

After obtaining $\mathcal{S}$, we will project it into the same image pixel value range as samples in the dataset R to make the explanation directly interpretable in the input space. At the same time, we limit the the size of explanation set $\mathcal{S}$ to be 10 images per class at most. To make our global explanations independent of the sampling dataset distribution $\mathcal{P}$ and reflect the true model decision, $\mathcal{S}$ is always initialized from the standard Gaussian Noise.

However, the above bi-level optimization is rather hard to solve directly. If the inner loops are unrolled too many times, the computation and memory occupation will increase exponentially and the problem becomes infeasible. At the same time, since the model parameter space is highly non-convex, it becomes extremely difficult to accurately approach $\boldsymbol{\theta}^R$ with limited inner iterations when starting from a random initialized $\theta^{\mathcal{S}}_0$.

3.2 Imitating training trajectories

To address the aforementioned challenges, recent works about data condensation (Zhao et al., 2021; Cazenavette et al., 2022) have proposed to imitate short ranges of original model training process rather than imitating the final model parameter $\boldsymbol{\theta}^R$ directly. Inspired by those works, we propose to imitate short intervals of the whole training process for each iteration of the outer loop. Specifically, after each epoch of training model on R, we save the model checkpoint $\boldsymbol{\theta}^R_k$ as well as the random initialization point $\boldsymbol{\theta}^R_0$. We then get a whole training trajectory as $\{\boldsymbol{\theta}^R_0, \boldsymbol{\theta}^R_1, \ldots, \boldsymbol{\theta}^R_k, \ldots, \boldsymbol{\theta}^R_T\}$ and each checkpoint in the training trajectory could act as a reference point. For each iteration in outer loop, we uniformly and randomly choose a reference point as starting point $\boldsymbol{\theta}^R_k$, and choose the reference point after N epochs $\boldsymbol{\theta}^R_{k+N}$ as the end point. With the model $\boldsymbol{\theta}^{\mathcal{S}}$ initialized as $\boldsymbol{\theta}^R_k$, in the inner loop, we use M step gradient descent to obtain the $\boldsymbol{\theta}^{\mathcal{S}}_M$ and then make it approach $\boldsymbol{\theta}^{\mathcal{R}}_{k+N}$. We then update the $\mathcal{S}$ according to the distance loss between $\boldsymbol{\theta}^{\mathcal{S}}_M$ and $\boldsymbol{\theta}^R_{k+N}$. To sum, instead of directly imitating the final parameter $\boldsymbol{\theta}^{\mathcal{R}}$, we choose to imitate the short range training dynamics to obtain the $\mathcal{S}$. Therefore, we turn the original problem into the following optimization problem:

$$\min_{\mathcal{S}} \mathbb{E}_{k\sim[0,\ldots,T]} D(\boldsymbol{\theta}^{\mathcal{S}}_M, \boldsymbol{\theta}^R_{k+N}) \quad \text{s.t} \quad \boldsymbol{\theta}^{\mathcal{S}}_M = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}) := \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) \tag{5}$$

where T is the difference of the whole training trajectory. To avoid affect from randomness about initialization of $\boldsymbol{\theta}^{\mathcal{R}}$, we also optimize the above problem on multiple training trajectories obtained by training $\boldsymbol{\theta}^{\mathcal{R}}$ with different initializations.

Furthermore, to make the global set independent of the sampling dataset distribution $\mathcal{P}$ and reflect the true model decision, during the optimization, $\mathcal{S}$ is initialized from the standard Gaussian Noise.

Fidelity: It is necessary for the explanation method to have a high fidelity. In other words, the prediction produced by an explanation should agree with the original model as much as possible. By approaching the $\boldsymbol{\theta}^{\mathcal{S}}$ with $\boldsymbol{\theta}^R$, the model trained $\boldsymbol{\theta}^{\mathcal{S}}$ on $\mathcal{S}$ would have achieve a similar generalization ability on both distribution $\mathcal{P}$ and other distributions as well. Here, we show the final MSE loss, $\|\boldsymbol{\theta}^{\mathcal{R}}_{k+N} - \boldsymbol{\theta}^{\mathcal{S}}_M\|^2$ obtained by our method. We also take the original trajectory difference $\|\boldsymbol{\theta}^{\mathcal{R}}_k - \boldsymbol{\theta}^{\mathcal{R}}_{k+N}\|^2$ as the reference. We compute their average values over multiple trajectories starting from different starting point $\boldsymbol{\theta}^{\mathcal{R}}_k$. Following experimental settings in Sec. 4, the average values of $\|\boldsymbol{\theta}^{\mathcal{R}}_{k+N} - \boldsymbol{\theta}^{\mathcal{S}}_M\|^2$ and $\|\boldsymbol{\theta}^{\mathcal{R}}_k - \boldsymbol{\theta}^{\mathcal{R}}_{k+N}\|^2$ are 0.35 and 0.76 respectively. We can observe that the MSE between the obtained $\boldsymbol{\theta}^{\mathcal{S}}_M$ and end point $\boldsymbol{\theta}^{\mathcal{R}}_{k+N}$ is significantly reduced compared with the original trajectory so that the $\boldsymbol{\theta}^{\mathcal{S}}_M$ is close enough to the reference point $\boldsymbol{\theta}^{\mathcal{R}}_{k+N}$ after the update.

Side benefits: Since we have multiple reference points of the training trajectory rather than only the final parameter $\boldsymbol{\theta}^R$, we could also investigate the feature representation learned during the training dynamics, which has been a open question in the deep learning field and several hypothesis have already been proposed to study this problem. By selecting the reference model parameter in different stage, we now provide a way to visualize the feature characteristic of different training stage. We defer the discussion later in Sec. 6.1.

Although the proposed trajectory matching is used in data condensation (Cazenavette et al., 2022), the goal of our work is totally different from data condensation which aims to improve the data efficiency of the training procedure by limiting the number of training samples. With the different objectives, data condensation doesn't require generating a highly interpretable explanation, so directly applying the data condensation method is not applicable. For example, tricks like ZCA transformation are widely used in data condensation (Cazenavette et al., 2022) however cause the generated samples hard to interpret. Also, there is no constraint on pixel range for generated samples in the data condensation as well. Besides, we also utilize our method for different applications such as backdoor detection in Sec. 5, and visualizing knowledge of various training paradigms in Sec. 6, where data condensation methods haven't explored.

4 CLASSWISE EXPLANATION

In this section, we conduct experiments to show that the proposed method could generate high-quality, class-discriminative global explanation. And we also take comparisons between our method and Activation maximization (AM) method.

Datasets and model architecture: We use four popular datasets: SVHN (Netzer et al., 2011), GTSRB (Stallkamp et al., 2012), CIFAR-10 (Krizhevsky et al., 2009) and Tiny ImageNet (Chrabaszcz et al., 2017). For the model architecture, we choose the simple ConvNet architecture (Gidaris & Komodakis, 2018), AlexNet (Krizhevsky et al., 2012), and VGG-11 (Simonyan & Zisserman, 2015) for illustration. The ConvNet has 3 duplicate convolution blocks followed by a linear classifier. Each block consists of 128 filters, average pooling, ReLU activation and instance normalization. For the large scale dataset, Tiny ImageNet, we use the ConvNet with 4 duplicate convolution blocks. Due to the page limit, we show the experimental results of AlexNet and VGG in Sec. B.1 of *Appendix*.

Implementation details: We set inner loop iterations M and N to be 60 and 2 for all datasets to generate the visual explanations. For selecting reference point as the starting point, we uniformly and randomly select from 0-20 epoch checkpoints for SVHN, GTSRB, and CIFAR-10 and 0-40 epoch checkpoints for Tiny ImageNet. The learning rate η_θ is set to be 0.02 for updating the $\boldsymbol{\theta}_t^{\mathcal{S}}$ in the inner loop, while $\eta_{\mathcal{S}}$ is 1000 for updating $\mathcal{S}$ in the outer loop. And the number of outer loop iteration is 5000. We use standard Gaussian noise to initialize our class-wise explanations $\mathcal{S}$. For Activation maximization (AM) method, we directly maximize the final output of each class before softmax layer. We also add the Gaussian blur regularization in AM method to have better visual quality on the explanations. To have a fair comparison, we limit the the size of explanation set S to be 10 images per class and initialize S with Gaussian noise. The implementation details are shown in *Appendix*.

Table 1: The discriminative power of class-wise explanations obtained on ConvNet for different models.

dataset \ model	ConvNet	ResNet50	WideResNet28-10	DenseNet121
CIFAR-10	100.0	100.0	100.0	100.0
GTSRB	100	97.7	98.1	97.9
SVHN	100.0	100.0	100.0	100.0
Tiny ImageNet	100.0	100.0	100.0	100.0

As shown in Figure 1, for each class of each dataset, our method could generate high-quality class-discriminative explanations, while AM method generate a visually unidentifiable explanation with a lot of high-frequency noise and irregular color background. At the same time, our generated explanation shows a great diversity, while AM method keeps generating very similar patterns. For example, the model trained on GTSRB would learn the shape and color of traffic signs (circle, triangle, blue, red) and the content inside the sign (number, light, arrow). Moreover, it could be observed that the generated explanation has an apparent number pattern although numbers in SVHN dataset are collected from different sources with different backgrounds. Our class-wise explanations reveal that the model has learned to extract and combine important features from the background information.

To further evaluate whether the generated explanation is highly class-discriminative, we put the explanations back into the original ConvNet model to see the classification results. For example, for the generated explanation of **Cat** class, we test if the original ConvNet and other models would classify them as **Cat**. The higher the classification accuracy, the highly class-discriminative the generated

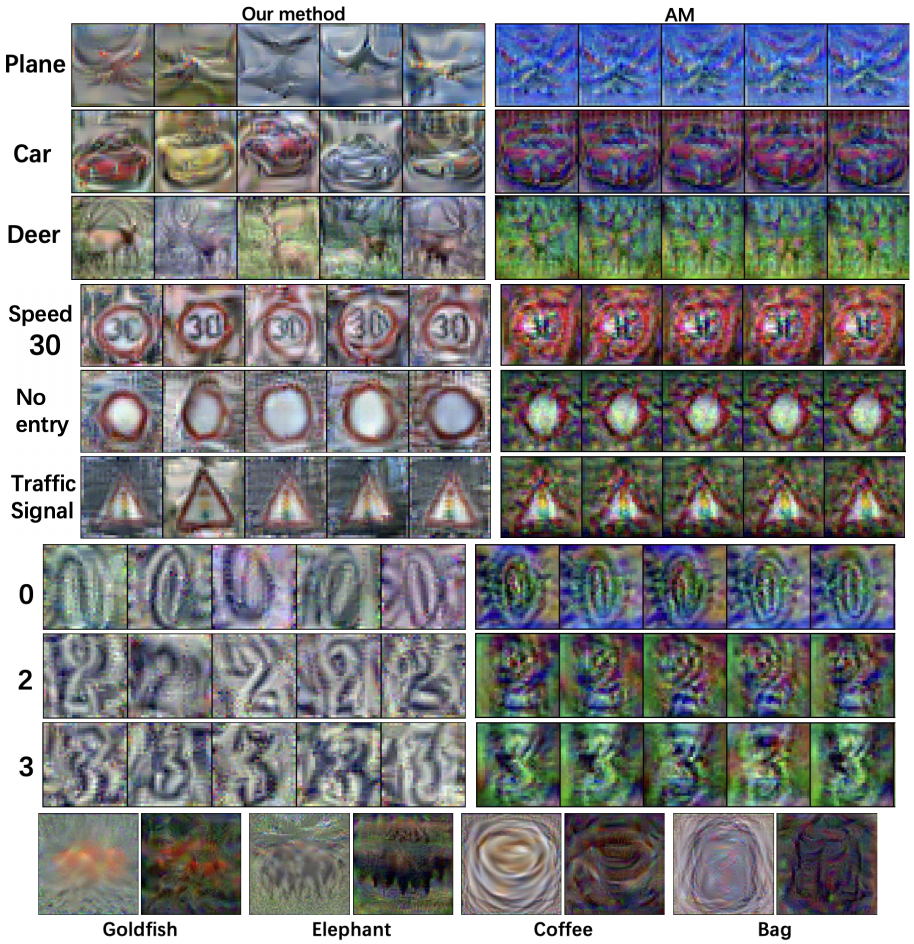

Figure 1: The comparison of our class-wise explanations and those from AM method on four datasets: CIFAR-10, GTSRB, SVHN, and Tiny ImageNet. For the first three datasets, the left column is ours and the right corresponds to AM. Each row denotes the explanation set for a class. For the final dataset, Tiny ImageNet, we generate one explanation for each class and the left is ours. Full results could be found in *Appendix*.

explanation of each class. Apart from the original ConvNet, we also utilize other pretrained models with different architectures containing ResNet50 (He et al., 2016), WideResNet28-10 (Zagoruyko & Komodakis, 2016), and DenseNet121 (Huang et al., 2017). The classification results are shown in Table 1.

5 Diagnosing Model Failures: Backdoor Attack Detection

It has been shown recently that the current state-of-art deep neural networks are vulnerable to backdoor attacks (Gu et al., 2019; Chen et al., 2017). Backdoor attacks aim to embed a hidden pattern in the training so that the trained model would still perform normally when the backdoor is not activated; otherwise, the prediction would be manipulated to the attack designated label. The backdoor trigger could be a sparse and simple pattern (Gu et al., 2019) or be a more sophisticated designed pattern like Blended attack ('Hello-Kitty' trigger) (Chen et al., 2017), SIG attack (vertical stripe trigger) (Barni et al., 2019). A lot of detection and defense methods have thus been proposed to detect whether the model is backdoored (Wang et al., 2019). One of the major requirement of the defenses is to use the find&patch strategy, where the defense method first find the exact trigger and then filter that trigger in the datasets. However, the existing defenses all depend on the assumption that the backdoor trigger has to be sparse and simple, which is unable to defend against some complex triggers such as Hello-Kitty or vertical stripe. We show the proposed method could recover both simple and complex backdoor triggers accurately. Here we apply our method and AM method to reveal the trigger learned by the ConvNet model in the CIFAR-10 dataset. Specifically, we choose three different kinds of backdoor attacks: 1) the Blended attack ('Hello-Kitty' trigger) (Chen et al.,

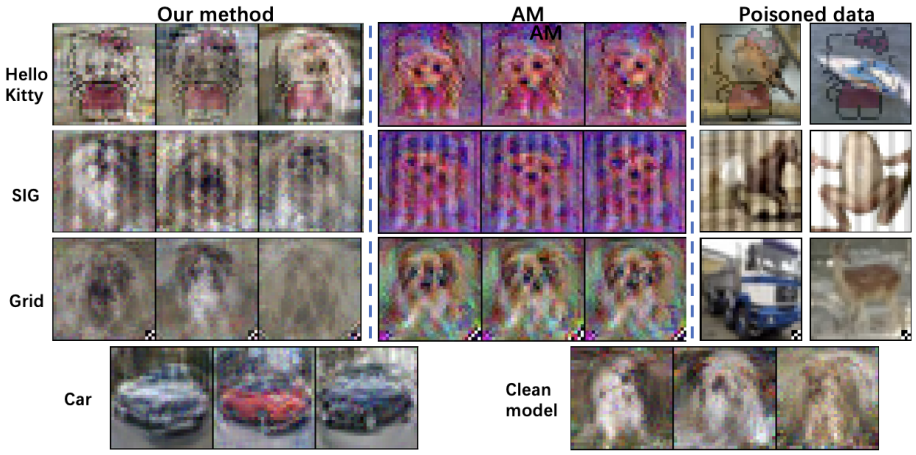

Figure 2: The comparison between our class-wise explanations of the model trained on CIFAR-10 with different backdoor attacks and explanations from AM method. The targeted class is the **Dog**. We choose three different kinds of backdoor attacks: 1) the Blended attack ('Hello-Kitty' trigger); 2) SIG attack; 3) Badnet attack (grid trigger). We visualize poisoned examples on CIFAR-10 with the above attacks in the right part of the first three rows. In the left part, the first three rows show the visualizations generated from our method of **Dog** class for the model trained with three kinds of attacks. The middle parts are the visualization generated by AM. The fourth row shows visualizations of non-targeted classes **Car**. The right part is the visualization of the class-wise explanation of **Dog** on the clean model, regarded as reference.

2017); 2) the SIG attack (vertical stripe trigger) (Barni et al., 2019); 3) the Badnet attack (grid trigger) (Gu et al., 2019). For all backdoor attacks, we select **Dog** as our targeted class. For attack setups, we follow the previous works (Li et al., 2021b;a) and set the poison rate to be 0.05. Please refer to *Appendix* for more details of backdoor attacks. For visual explanation of the model trained on CIFAR-10 with backdoor attacks, we fix the starting point with the first checkpoint in the expert trajectory and set the N as 1. The other parameters for visual explanation are the same in Sec.4.

As shown in left parts of Figure 2, our method could successfully reveal all kinds of inserted triggers with high quality, even when the poisoning rate is very low. Simultaneously, our explanation keeps the other classes natural and clean. On the other hand, AM fails to extract clear triggers from poisoned models and the reserved triggers have a clear difference with the ground truth. Moreover, our method could quickly recover the backdoor trigger as we only need one reference point i.e $N = 1$. By only poisoning a small number of examples, all explanations in the **Dog** class are consistently carrying the exact trigger with the same shape and location. We could easily notice whether a model has been backdoored through the proposed explanation, thus finding the corresponding trigger. Our method could then be used to filter out those examples with the revealed trigger and purify the model.

6 Visualizing Model Knowledge

In this section, we further demonstrate that the proposed method could be used to analyze the feature representation learned by different training methods and phases.

6.1 Training Dynamics

DNNs have shown great success on a variety of tasks. However, it is still a great challenge to understand why a model could generalize well on the test set. It is thus important to know what model has learned intrinsically in the whole training procedure. In this section, we show the proposed method could be used as a tool to visualize model knowledge in different stages of the training procedure. We use the ConvNet model trained on the CIFAR-10 dataset as an example. To reveal the difference among knowledge in different stages in the whole training process, we choose the starting point of the trajectory sequentially and set the $N = 1$. That means we make $\mathcal{S}$ imitate the training dynamics for only one epoch. The other parameters are the same in Sec.4. The knowledge learned in different stages are shown in Figure 3. We set the start point sequentially: (a) random initialization point; (b) the checkpoint saved after 2 epochs; (c) the checkpoint saved after 5 epochs; (d) the checkpoint saved after 10 epochs; (e) the checkpoint saved after 20 epochs; (f) the checkpoint saved after 30 epochs.

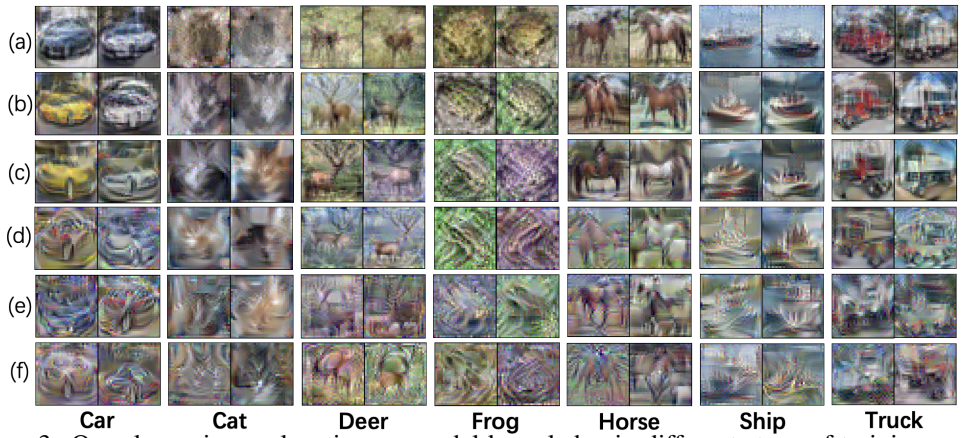

Figure 3: Our class-wise explanation on model knowledge in different stage of training process corresponding to Sec. 6.1. Here, we only show the visualizations of some classes. And, the whole demonstration is shown in the *Appendix*. Each row represents the visualization of a different training stage: (a) epoch 0-1; (b) epoch 2-3; (c) epoch 5-6; (d) epoch 10-11; (e) epoch 20-21; (f) epoch 30-31.

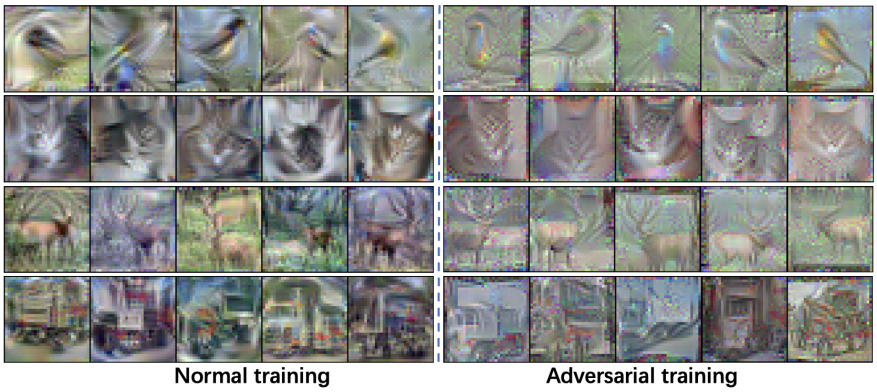

Figure 4: Comparison between our explanation of normal training and that of adversarial training (AT). Here, we only show the visualizations of some classes: **Bird, Cat, Deer, Truck**. And the whole demonstration is shown in *Appendix*. The left part of each row is the visualization of normal training, and the right part belongs to AT.

As shown in Figure 3, in the early stage of the training procedure, the knowledge learned by the model have rich information about the color and the rough contour of the class object. For example, the background of **Ship** class is always blue, and that of **Horse** class is brown. Also, some rough shapes, such as horse's body and car's body, could be easily identified. With the training process continuing, model knowledge tend to include clean and sharp local traits of the object, such as the head of the horse and the buckhorn. In the meantime, texture becomes more clear and becomes the dominant feature in the later phase of training. Although the model gets better performance with training, the learned knowledge is actually less aligned with human perception. This observation is align with Kumar et al. (2022). They also observed that representation from underfitting ImageNet models with modest validation accuracy achieves the best perception score.

6.2 Adversarial Training

Adversarial training (AT) has been one of the most effective methods to enhance adversarial robustness (Madry et al., 2018). At the same time, an adversarially trained "robust model" tends to generate a better feature representation that has a better semantic meaning and aligns better with human perception (Ilyas et al., 2019). Recently, adversarial perturbations have been used to improve the model generalization in both computer vision (Xie et al., 2020) and natural language processing domains (Gan et al., 2020). In this section, we study the model knowledge obtained by adversarially trained model.

In the experiment, we use the ℓ_∞ PGD-AT (Madry et al., 2018) to train a ConvNet AT model. We use the cross-entropy loss and set the perturbation constraint $\epsilon = 4/255$. We set the number of iterations

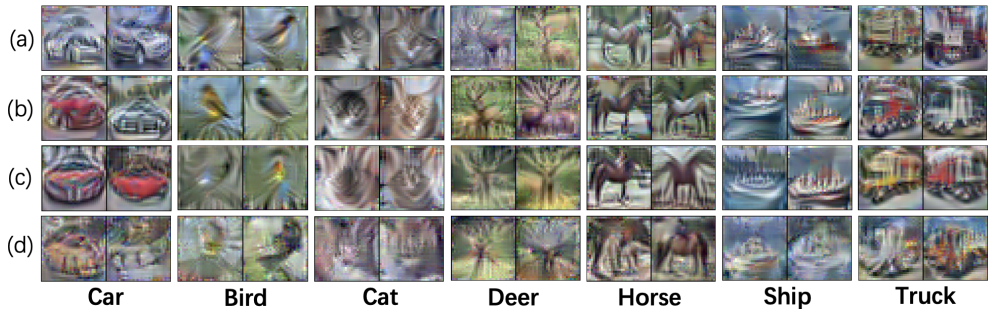

Figure 5: Our explanation of training model with different label noise corresponding to Sec. 6.3. Here, we just show the visualizations of some classes. And, the whole demonstration is shown in the *Appendix*. Each row represents the visualization of a different training process: (a) normal training (0% label noise) ; (b) 25% label noise; (c) 50% label noise; (d) 75% label noise.

of inner maximization as 10 and step size as $2/255$. More details of adversarial training are shown in the *Appendix*. The other parameters for conducting visualizations are the same as those used in Sec.4.

From Figure 4, the most obvious difference is the contrast ratio of adversarially trained feature is much smaller than the normal training. We find that the representation set from adversarial training has a much wider pixel range from $[-5, 5]$ compared to $[-1, 1]$ in the normal training. Then, we have to project the representation set into the $[-1, 1]$ space in order to do the visualization. The change of pixel range might be brought by the adversarial training mechanism. That is, the model relies on edge cases much heavier since model update depends on calculating loss on adversarial examples generated in the inner loop. As we are approaching the representation set using the adversarial training method, the representation set has a wider pixel range. Moreover, we can clearly observe that adversarially-trained feature representation aligns better with human perception and looks more "clean", which is also supported by other works (Ilyas et al., 2019; Xie et al., 2020).

6.3 Noisy label training

Since the seminal work discusses the learning algorithm should cope with incorrect training examples (Angluin & Laird, 1988), machine learning with noisy label has become a heated topic as, in real-world applications, the labels are often noisy and imperfect. Therefore, it is important to understand how the neural network knowledge will be changed when the label is noisy. On the other hand, deep learning is well-known for having ability to learn very complex features and becomes over-fitting. While label noise could be seen as a challenge in current machine learning, a proper noise strength could act as a good regularizer to help the model generalize better. In this section, we study the difference of model knowledge under the different levels of label noise.
Here, we also use ConvNet model trained on the CIFAR-10 dataset as an example. We modify the dataset by adding various levels of noise $(25\%, 50\%, 75\%)$ to the labels of the training set. This noise is added by taking, say in the case of 25% label noise, 25% of the examples at random and randomly permuting their labels. For generating class-wise explanations, we also follow the hyperparameter setting used in Sec.4.

As shown in Figure 5, a small amount of label noise like 25% does help to achieve a more human-like model knowledge. For example, birds and cats in Figure 5 have a much sharper and better semantic feature than pure training without any label noise. Our observation is that label noise could act as a good regularizer to help to get the model knowledge closer to human perception. However, as the noise level increases, the quality drops significantly. When the noise level reaches 75%, the learned knowledge becomes barely recognizable.

7 Conclusion

In this paper, we propose a global visual explanation method, which generates high-quality and class-discriminative explanation in the input space. We further show the proposed method could be utilized for debugging model failures, such as revealing backdoor triggers in the attack. Finally, we devise a way to study the model knowledge in different training mechanisms, which sheds light on building a more generalizable and trustworthy machine learning method.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018. 1, 2.1

Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988. 6.3

Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 101–105. IEEE, 2019. 5

Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504*, 2017. 1, 2.2, 3

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. *arXiv preprint arXiv:2203.11932*, 2022. 3.2, 3.2

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017. 5

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. 4

Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8, 1995. 2.2

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017. 1, 2.2

Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for vision-and-language representation learning. *arXiv preprint arXiv:2006.06195*, 2020. 6.2

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4367–4375, 2018. 4, A

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. 5

Karthik S Gurumoorthy, Amit Dhurandhar, Guillermo Cecchi, and Charu Aggarwal. Efficient data representation by selecting prototypes with importance weights. In *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 260–269. IEEE, 2019. 2.2

Satoshi Hara and Kohei Hayashi. Making tree ensembles interpretable: A bayesian model selection approach. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 77–85. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/hara18a.html. 1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015. A

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 4

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 1, 2.2

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017. 4

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pp. 125–136, 2019. 6.2

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf. 2.2

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. 4

Manoj Kumar, Neil Houlsby, Nal Kalchbrenner, and Ekin D Cubuk. On the surprising tradeoff between imagenet accuracy and perceptual similarity. *arXiv preprint arXiv:2203.04946*, 2022. 6.1

Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Interpretable & explorable approximations of black box models. *arXiv preprint arXiv:1707.01154*, 2017. 1

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34, 2021a. 5, A

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=9l0K4OM-oXE. 5, A

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017. 2.1

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018. 6.2

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 4

Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4467–4477, 2017. 2.2

Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding neural networks via feature visualization: A survey. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 55–76. Springer, 2019. 2.2

Munachiso Nwadike, Takumi Miyawaki, Esha Sarkar, Michail Maniatakos, and Farah Shamout. Explainability matters: Backdoor attacks on medical imaging. *arXiv preprint arXiv:2101.00008*, 2020. 1

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization. 2.2

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016. 1, 2.1

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017. 1, 2.1

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015. 4

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 1, 2.1

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 1, 2.1

Daniel Soudry, Elad Hoffer, and Nathan Srebro. The implicit bias of gradient descent on separable data. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1q7n9gAb. 1

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012. 4

Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. Learning global additive explanations for neural nets using model distillation. 2018. 2.2

Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. Learning global additive explanations for neural nets using model distillation, 2019. URL https://openreview.net/forum?id=SJl8J30qFX. 1

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. A

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723. IEEE, 2019. 5

Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems. *IEEE Transactions on Information Forensics and Security*, 16:4772–4787, 2021. 1

Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 819–828, 2020. 6.2

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015. 2.2

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 4

Shihao Zhao, Xingjun Ma, Yisen Wang, James Bailey, Bo Li, and Yu-Gang Jiang. What do deep nets learn? class-wise patterns revealed in the input space. *arXiv preprint arXiv:2101.06898*, 2021. 1, 2.1, 3.2

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016. 1, 2.1

A IMPLEMENTATION DETAILS

All experiments are conducted on 2 Nvidia-V100 GPUs.

Datasets SVHN dataset is a realworld digit dataset which has $73,257$ training and $26,032$ testing images of 10 numbers. CIFAR-10 has $50,000$ training and $10,000$ testing images from 10 object categories. GTSRB contains 43 classes of traffic signs, split into $39,209$ training images and $12,630$ test images. Tiny ImageNet contains 100000 images of 200 classes. Each class has 500 training images and 50 test images. The image scale of the first three datasets is 32×32. The image scale of Tiny ImageNet is 64×64. all those datasets are licensed under MIT.

Implementation details of training base models. We choose ConvNet (Gidaris & Komodakis, 2018) as our base model. The default ConvNet has 3 duplicate convolution blocks followed by a linear classifier, and each block consists of 128 filters, average pooling, ReLu activation, and instance normalization (Ulyanov et al., 2016). The ConvNet is randomly initialized with Kaiming initialization (He et al., 2015). We set the learning rate as 0.01 and adopt SGD to train the ConvNet for 100 epochs. We also adopt the data augmentations: crop, rotation, scale, and flip transformations with the implementation from https://github.com/mit-han-lab/data-efficient-gans. Our ConvNet could get 85.34% accuracy on CIFAR-10. For other models used for evaluation in Table 1, we use the implementation in https://github.com/kuangliu/pytorch-cifar.

Implementation details of Activation maximization method. We adopt the source code from this github repository[2]. We utilize the Gaussian blur regularization to further improve visualization quality. The iteration number of maximiztion procedure is set as 3000. The step size is set as 10 and the radius for Gaussian blur is 1.

Implementation details of adversarially training models. We use the implementation in https://github.com/DengpanFu/RobustAdversarialNetwork to adversarially train model on CIFAR-10. We use the ℓ_∞ norm as our norm constraint. We train the AT model against perturbations of size $\epsilon = 4/255$. We set the number of iteration of inner maximization as 10 and step size as $2/255$. We use SGD with momentum 0.9 in outer minimization and train model for 200 epochs.

Implementation details of backdoor attacks For backdoor attacks, we choose the Blended attack, Badnet attack, and SIG attack. We follow the attack setting used in Li et al. (2021b;a). We use a 3×3 grid square located at the bottom right corner as the trigger pattern on CIFAR-10 dataset for the Badnet attack. For the Blended attack, we adopt the 'Hello-Kitty' pattern on CIFAR-10 and set the blended ratio $\lambda = 0.2$. For SIG attack, we set the $\delta = 20$. For all backdoor attacks, we select the **Dog** as our targeted class and set the poison rate as 0.05. We set the learning rate as 0.01 and adopt SGD to train the ConvNet for 100 epochs.

B FULL VISUALIZATION RESULTS

In Sec. B.1, we show full visualization results of the class-wise explanations. In Sec. B.2, we show full class-wise explanations of model poisoned by backdoor attacks on CIFAR-10 dataset. In Sec. B.3, we show full class-wise explanations of different phases of the model training process on CIFAR-10 dataset. We demonstrate full class-wise explanations of the adversarially trained model on CIFAR-10 dataset in Sec. B.4. We demonstrate full class-wise explanations of the model trained under the different levels of noise on CIFAR-10 dataset in Sec. B.5.

[2] https://github.com/utkuozbulak/pytorch-cnn-visualizations

B.1 THE VISUALIZATION RESULTS OF CLASSWISE EXPLANATIONS

We demonstrate the whole class-wise explanations from our method about ConvNet on CIFAR-10, SVHN, GTSRB, and Tiny ImageNet in Figure 6, 7, 8, 9, and 10, respectively. For CIFAR-10, SVHN and GTSRB, we set the size of class-wise explanation set as 10 images per class ane 1 for Tiny Imagnet. For all datasets, we use the standard Gaussian noise as the initialization of our explanation set $\mathcal{S}$.

For VGG-11 and AlexNet models, we show the visualizations generated from our method on CIFAR-10 in Figure 11 and Figure 12.

The visualization generated from Activation maximization method of these four datasets on the ConvNet are shown Figure 13, 14, 15, 16, and 17.

The class-wise explanations about ConvNet on CIFAR-10 are shown in Figure 6. Each row corresponds to each class.

Figure 6: Our class-wise visualizations of ConvNet on CIFAR-10 dataset.

The class-wise explanations about ConvNet on SVHN are shown in Figure 7.

Figure 7: Our class-wise visualizations of ConvNet on SVHN dataset.

The class-wise explanations about ConvNet on GTSRB are shown in Figure 8 and 9. The Figure 8 is about class-wise visualizations of class label $0 - 18$. The Figure 9 is about class-wise visualizations of class label $19 - 42$.

Figure 8: Our class-wise visualizations (class label $0 - 18$) of ConvNet on GTSRB dataset.

The below Figure 9 is about class-wise visualizations of class label $19 - 42$ on GTSRB dataset.

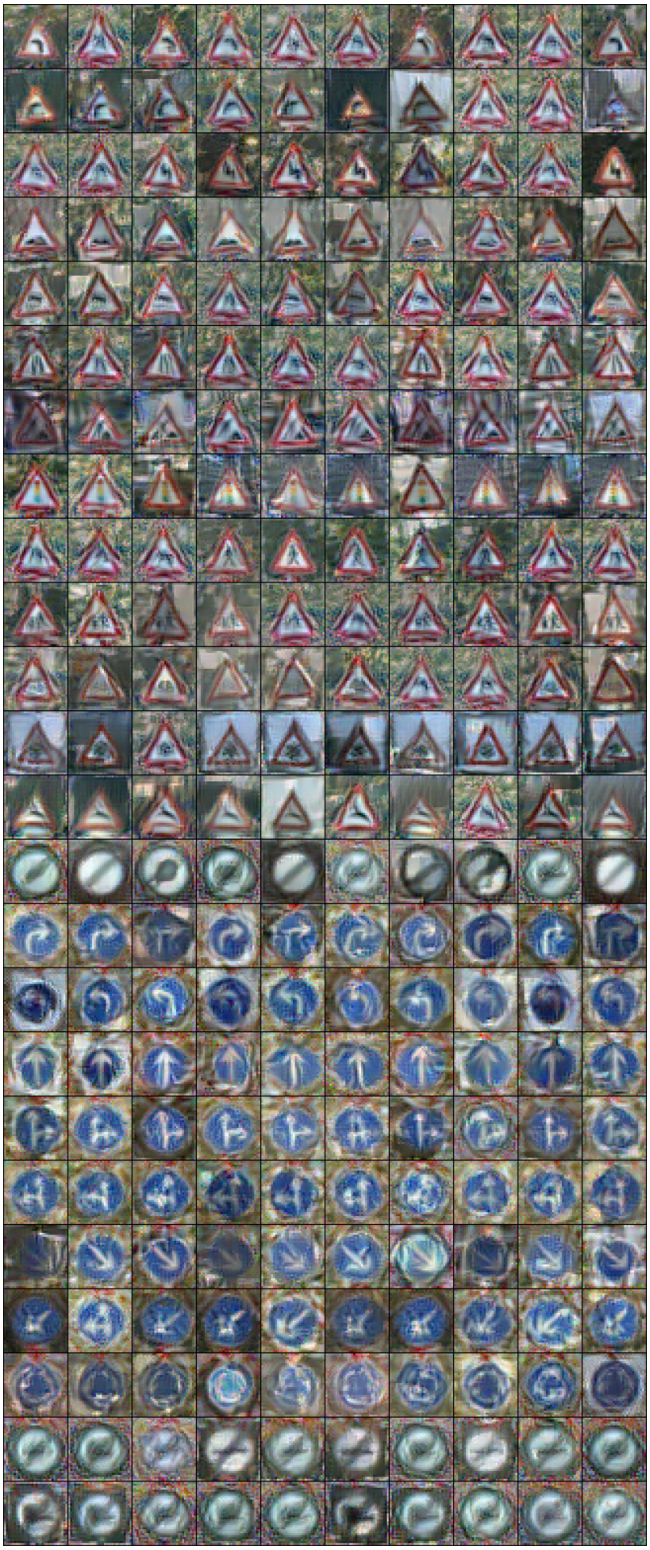

Figure 9: Our class-wise visualizations (class label $19 - 42$) of ConvNet on GTSRB dataset.

For the large scale dataset, Tiny ImageNet, we use the ConvNet with 4 convolution blocks. Due to the computation and memory cost, we set the size of class-wise explanation set as 1 image per class on Tiny ImageNet. The class-wise explanations about ConvNet on Tiny ImageNet are shown in Figure 10. Each subfigure corresponds to each class.

Figure 10: Our class-wise visualizations on Tiny ImageNet dataset.

The class-wise explanations about VGG-11 and AlexNet on CIFAR-10 generated by our method are shown in below figures, Figure 11 an 12. Each row corresponds to each class.

Being similar to the results on ConvNet model, our method can still generate high-quality visualizations on larger CNN models. And our class-wise explanations have the apparent class-wise features for each class. These also verify that our method could generalize well on various network architectures.

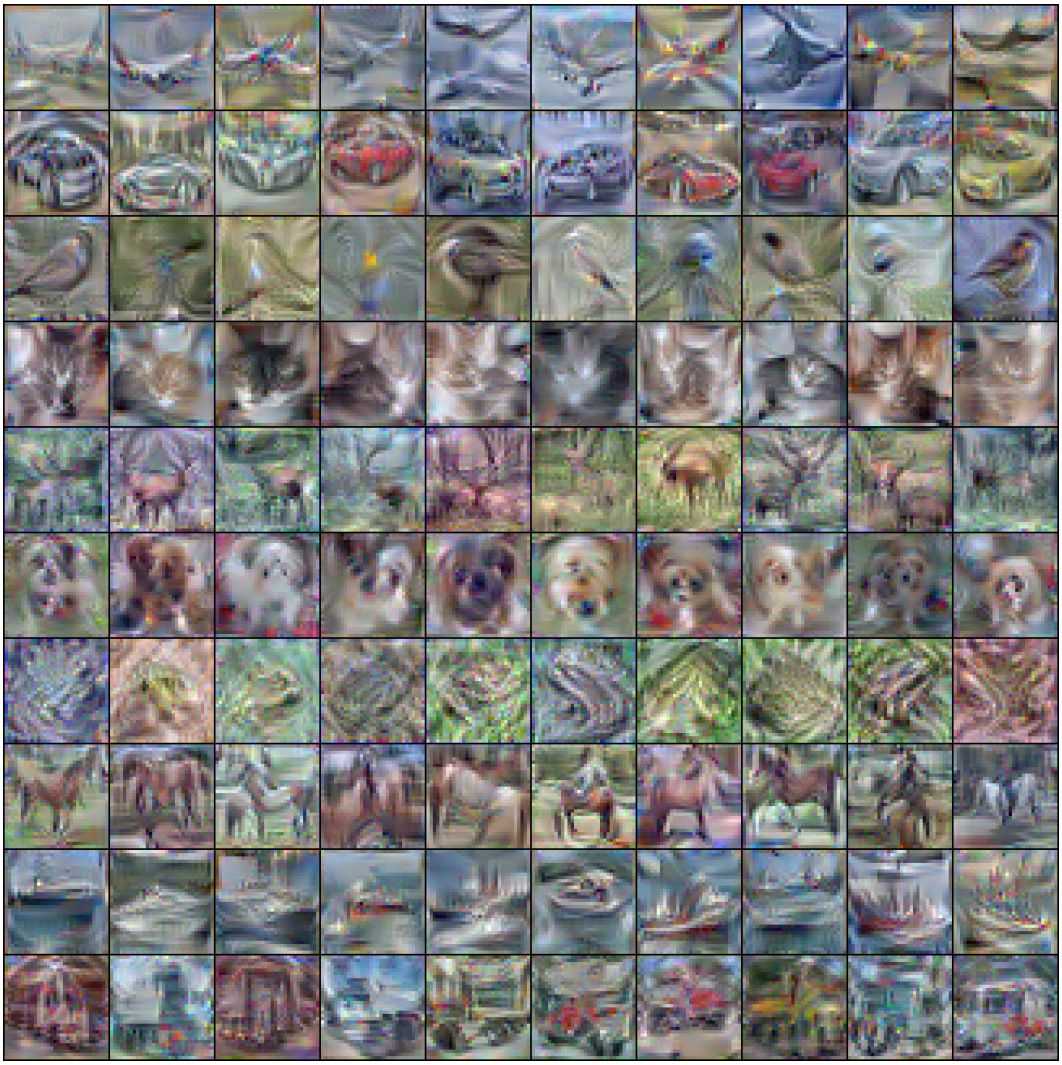

Figure 11: Our class-wise visualizations of VGG-11 on CIFAR-10 dataset.

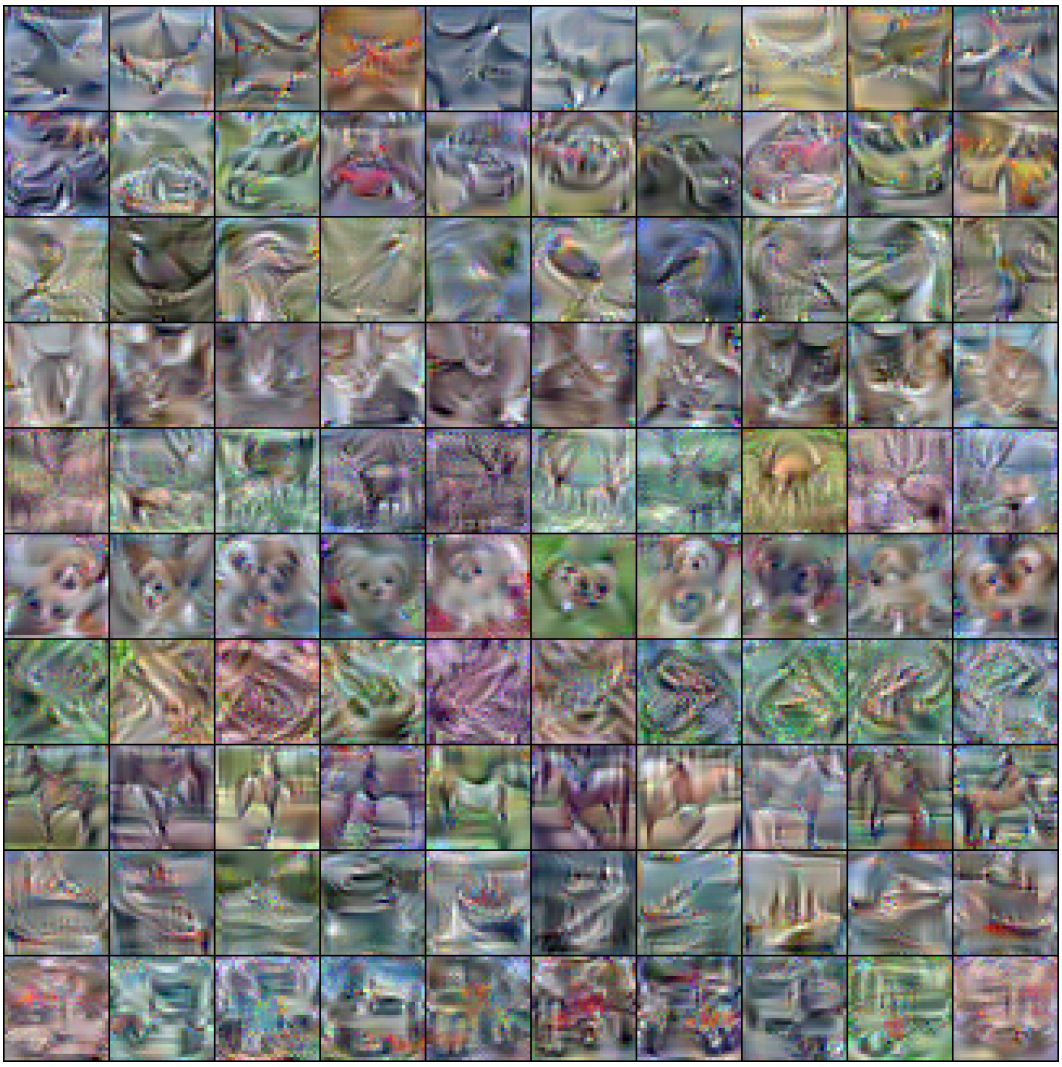

Figure 12: Our class-wise visualizations of AlexNet on CIFAR-10 dataset.

The class-wise explanations generated from AM about ConvNet on CIFAR-10 are shown in Figure 13. Each row corresponds to each class.

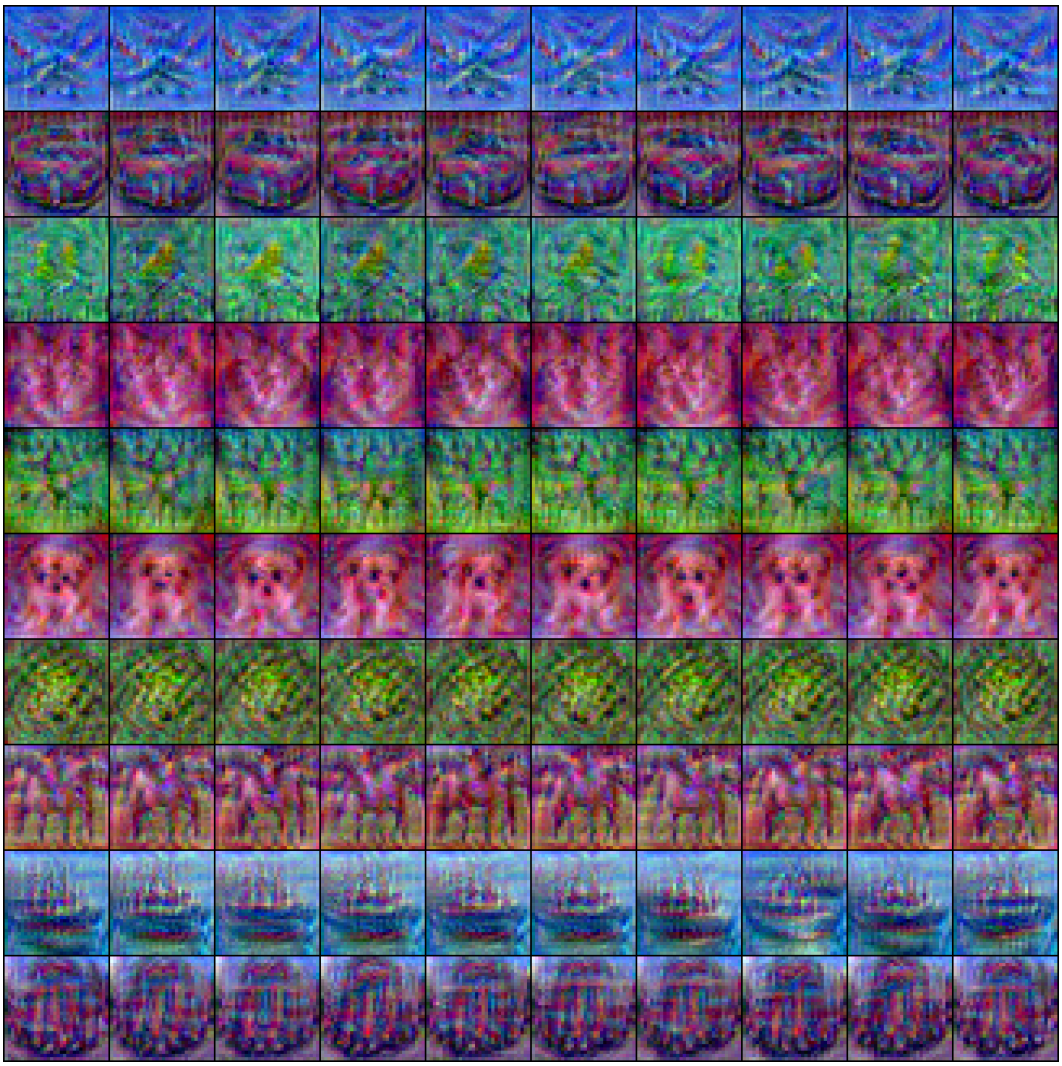

Figure 13: The class-wise visualizations generated from AM of ConvNet on CIFAR-10 dataset.

The class-wise explanations about ConvNet on SVHN are shown in Figure 14.

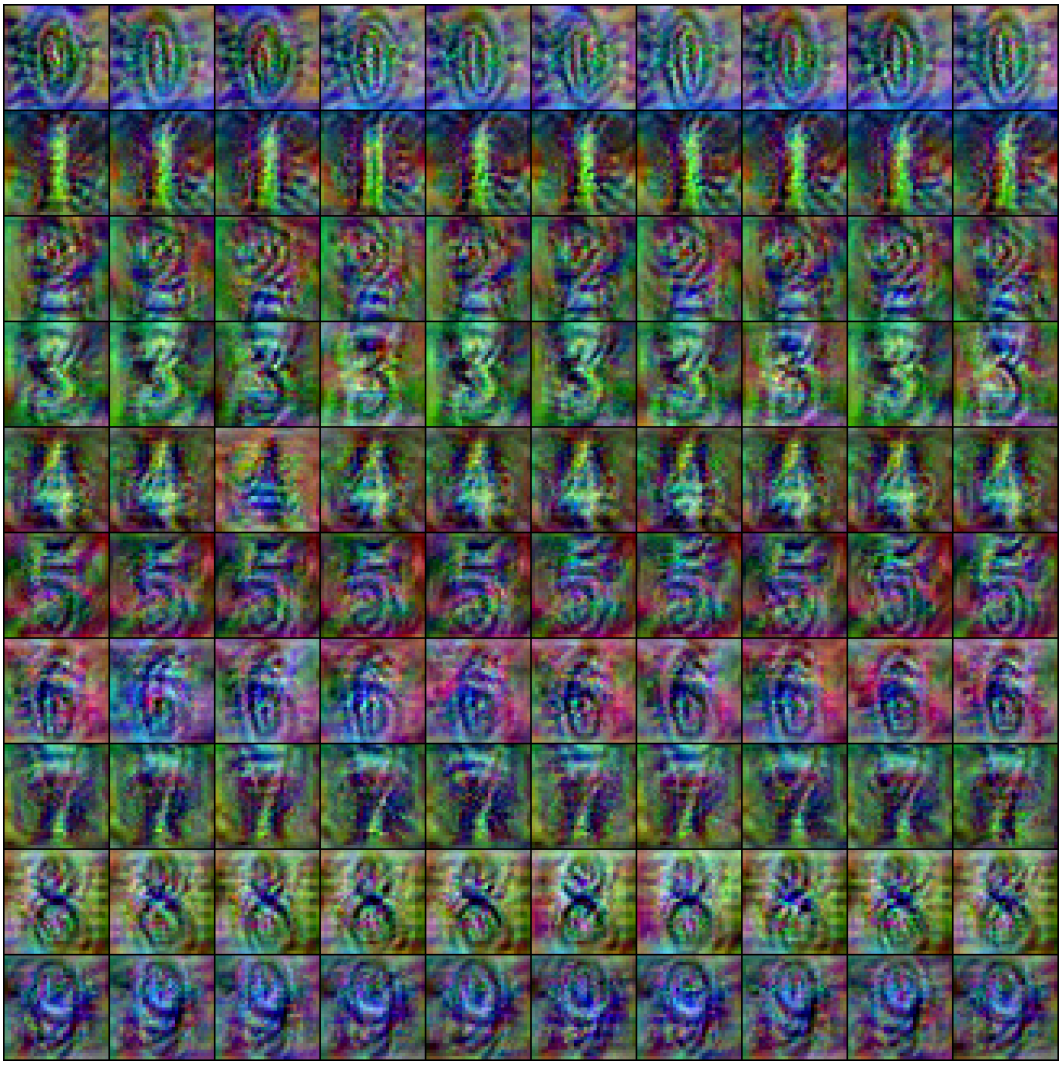

Figure 14: The class-wise visualizations generated from AM of ConvNet on SVHN dataset.

The class-wise explanations from AM method about ConvNet on GTSRB are shown in Figure 15 and 16. The Figure 15 is about class-wise visualizations of class label $0-18$. The Figure 16 is about class-wise visualizations of class label $19-42$.

Figure 15: The class-wise visualizations from AM (class label $0-18$) of ConvNet on GTSRB dataset.

The below Figure 16 is about class-wise visualizations from AM of class label $19 - 42$ on GTSRB dataset.

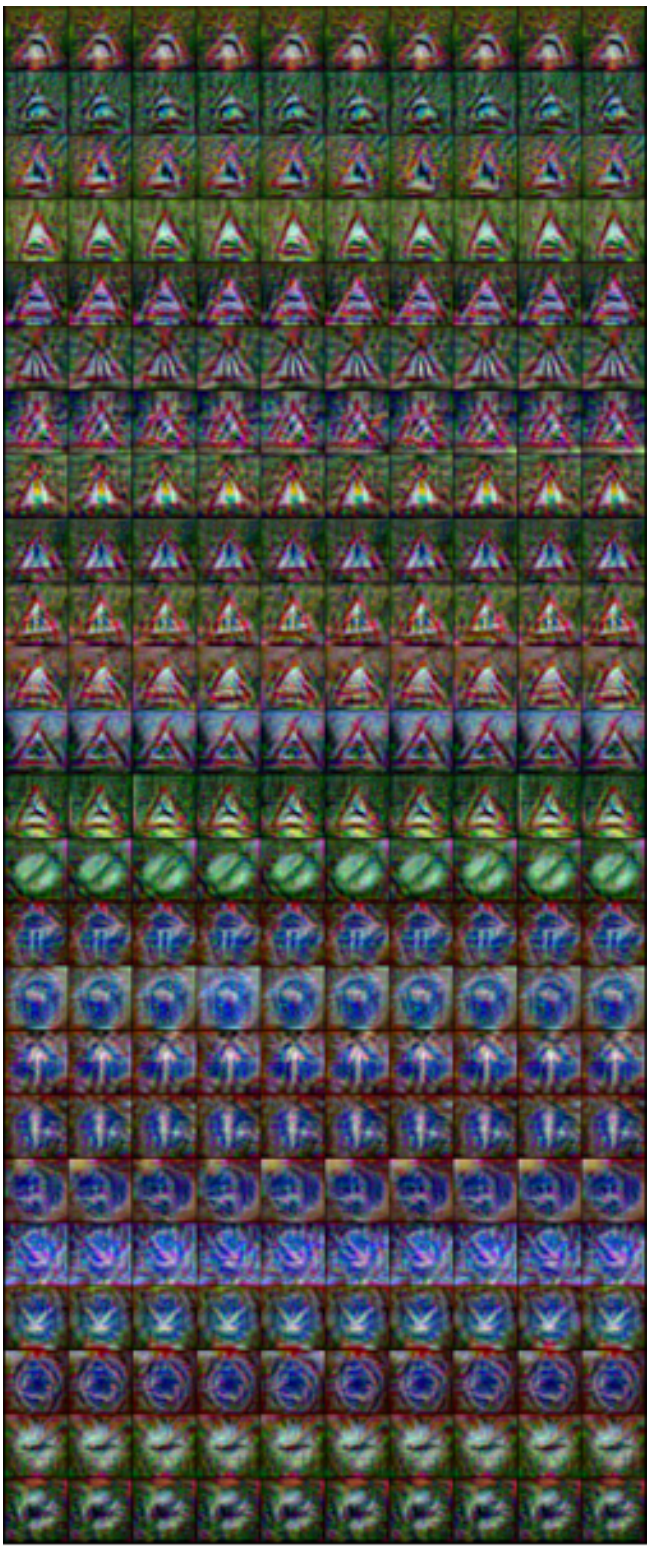

Figure 16: The class-wise visualizations (class label $19 - 42$) from AM of ConvNet on GTSRB dataset.

For the large scale dataset, Tiny ImageNet, we use the ConvNet with 4 convolution blocks. Due to the computation and memory cost, we set the size of class-wise explanation set as 1 image per class on Tiny ImageNet. The class-wise explanations generated from AM about ConvNet on Tiny ImageNet are shown in Figure 17. Each subfigure corresponds to each class.

Figure 17: The class-wise visualizations from AM on Tiny ImageNet dataset.

B.2 THE VISUALIZATION RESULTS OF BACKDOOR ATTACKS

We first demonstrate the visualizations of three different triggers from Activation maximization are shown in Figure 18. The visualizations of backdoor learning with three different attacks from our method are in the Figure 19, 21, and 20.

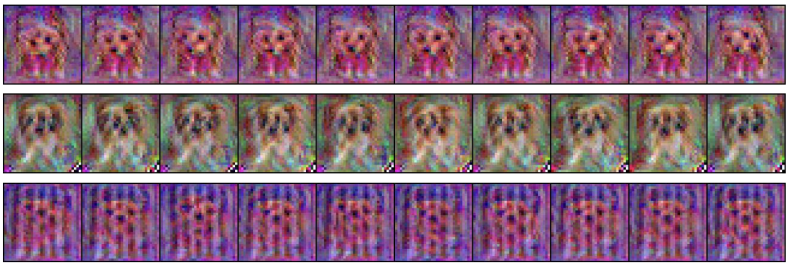

Figure 18: The visualizations of three triggers (**Hello-Kitty**, **Sig**, and **Grid** triggers) on the ConvNet model and CIFAR-10 dataset.

The visualizations from our method of three different triggers are shown in the the following 3 figures.

Figure 19: The class-wise visualizations of the ConvNet model attacked by **blended attacks (Hello-Kitty trigger)** on CIFAR-10 dataset.

Figure 20: The class-wise visualizations of ConvNet model attacked by **SIG attack** on CIFAR-10 dataset.

Figure 21: The class-wise visualizations of ConvNet model attacked by **badnet attack (grid trigger)** on CIFAR-10 dataset.

B.3 THE VISUALIZATION RESULTS OF TRAINING DYNAMICS

In this section, we show the class-wise visualizations of different phases of training process with our method.

Figure 22: The class-wise visualizations for epoch 0-1 training process of ConvNet on CIFAR-10 dataset.

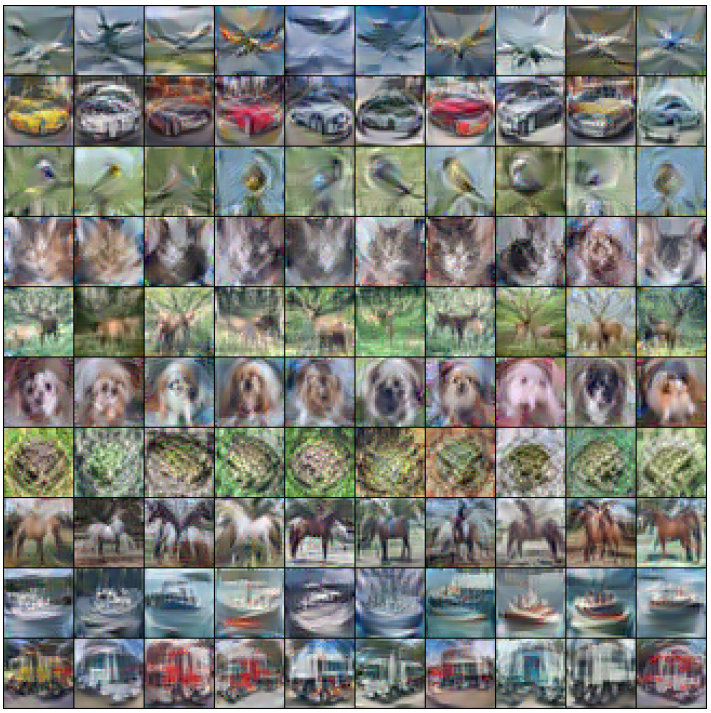

Figure 23: The class-wise visualizations for epoch 2-3 training process of ConvNet on CIFAR-10 dataset.

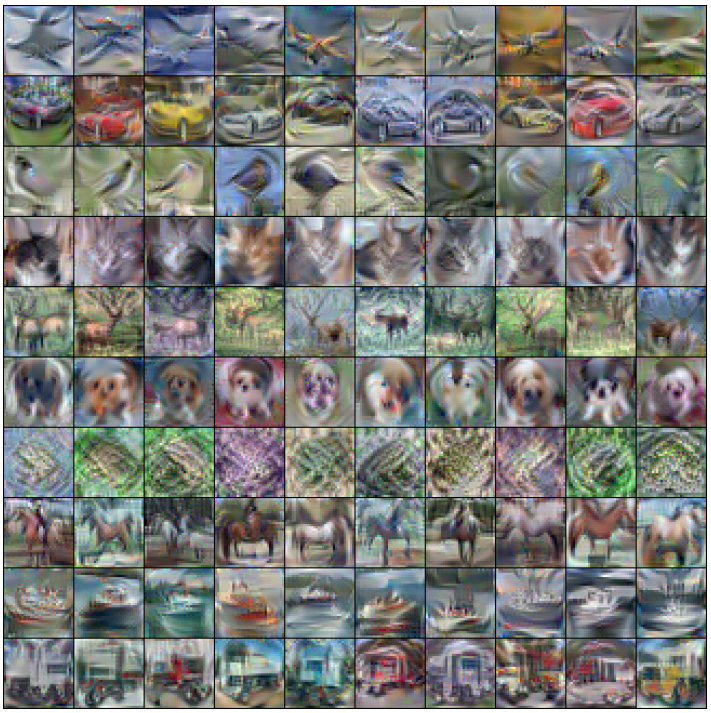

Figure 24: The class-wise visualizations for epoch 5-6 training process of ConvNet on CIFAR-10 dataset.

Figure 25: The class-wise visualizations for epoch 10-11 training process of ConvNet on CIFAR-10 dataset.

Figure 26: The class-wise visualizations for epoch 20-21 training process of ConvNet on CIFAR-10 dataset.

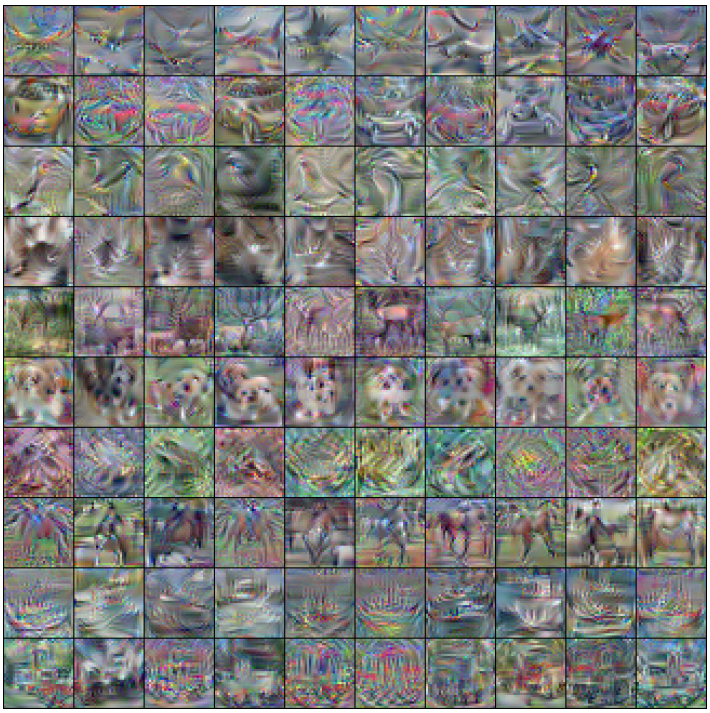

Figure 27: The class-wise visualizations for epoch 30-31 training process of ConvNet on CIFAR-10 dataset.

B.4 THE VISUALIZATION RESULTS OF AT MODEL

In this section, we show the class-wise explanations of adversarially trained model.

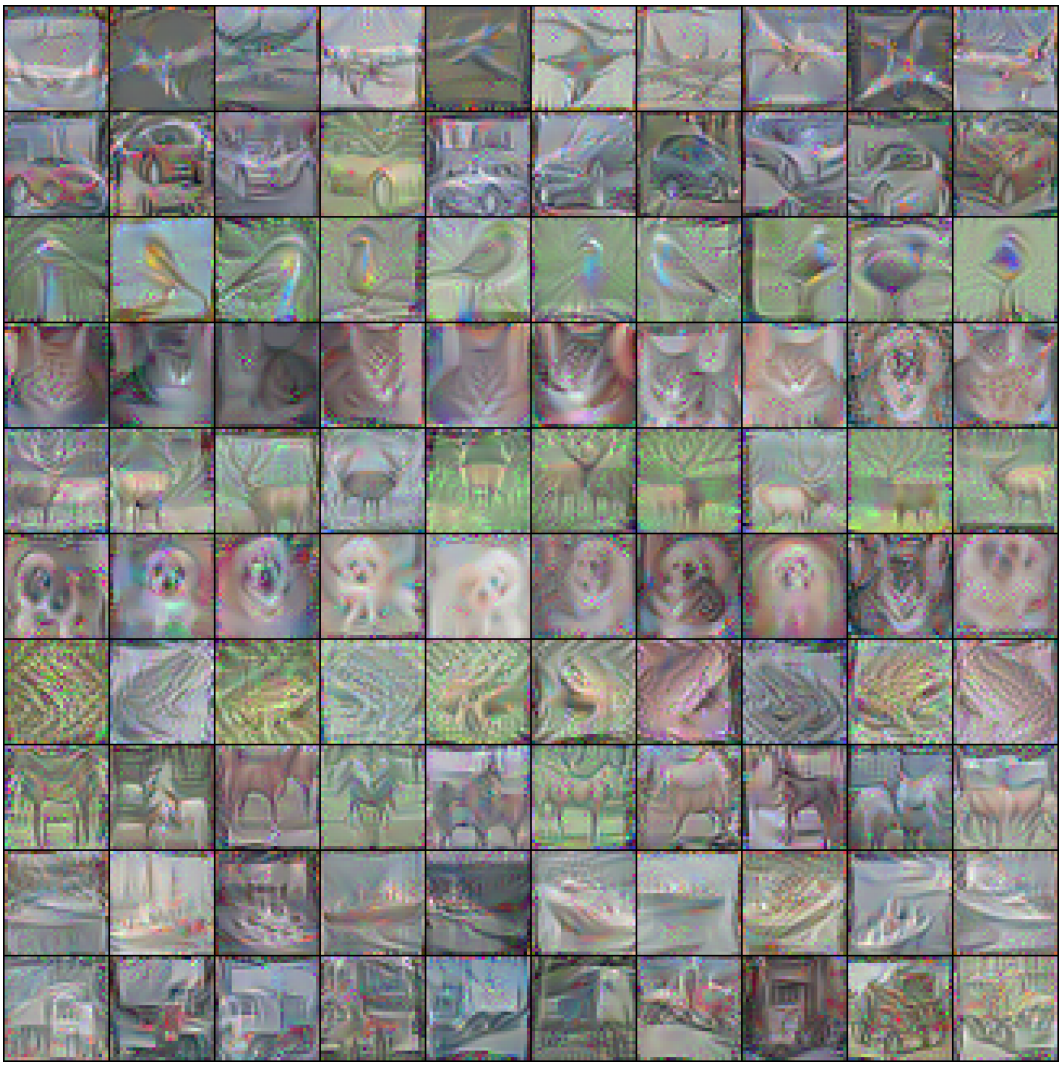

Figure 28: The class-wise visualizations of adversarial trained ConvNet model on CIFAR-10 dataset.

B.5 THE VISUALIZATION RESULTS OF NOISY LABEL TRAINING

In this section, we show the class-wise explanations of models trained with the different levels of label noise in the following figures.

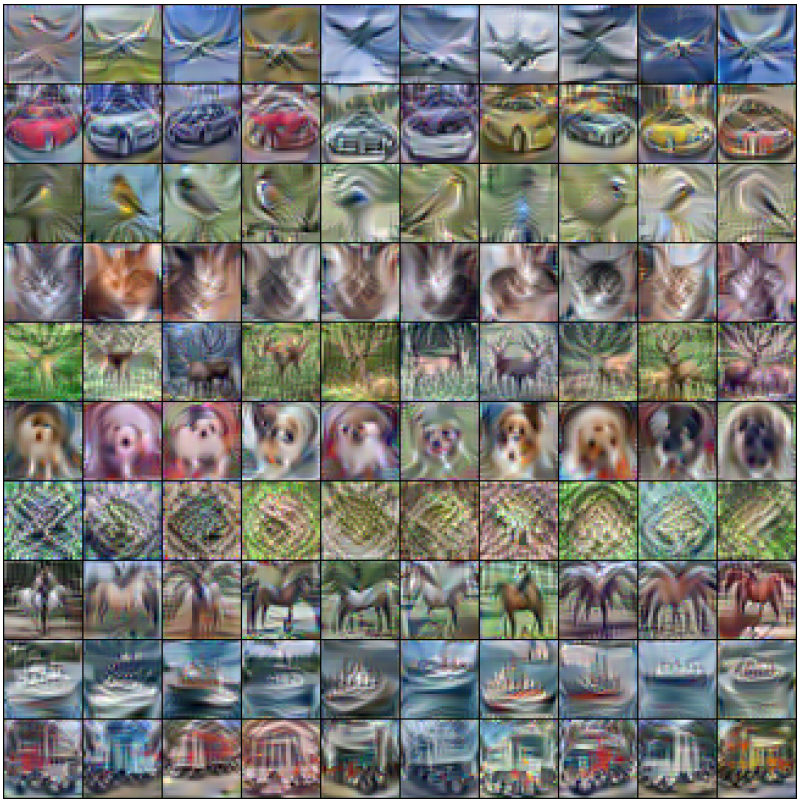

Figure 29: The class-wise visualizations of label noise training ConvNet model with 25% noise on CIFAR-10 dataset.

Figure 30: The class-wise visualizations of label noise training ConvNet model with 50% noise on CIFAR-10 dataset.

Figure 31: The class-wise visualizations of label noise training ConvNet model with 75% noise on CIFAR-10 dataset.