Reinforcement Explanation Learning

Siddhant Agarwal * **Owais Iqbal** **Sree Aditya Buridi** **Madda Manjusha** **Abir Das**
Indian Institute of Technology, Kharagpur
India

Abstract

Deep Learning has become overly complicated and has enjoyed stellar success in solving several classical problems like image classification, object detection, etc. Several methods for explaining these decisions have been proposed. Black-box methods to generate saliency maps are particularly interesting due to the fact that they do not utilize the internals of the model to explain the decision. Most black-box methods perturb the input and observe the changes in the output. We formulate saliency map generation as a sequential search problem and leverage upon Reinforcement Learning (RL) to accumulate evidence from input images that most strongly support decisions made by a classifier. Such a strategy encourages to search *intelligently* for the perturbations that will lead to high-quality explanations. While successful black box explanation approaches need to rely on heavy computations and suffer from small sample approximation, the deterministic policy learned by our method makes it a lot more efficient during the inference. Experiments on three benchmark datasets demonstrate the superiority of the proposed approach in inference time over state-of-the-arts without hurting the performance. Our code is made public at `https://cvir.github.io/projects/rexl.html`

1 Introduction

Deep Learning methods are enjoying enormous success in computer vision, natural language and robotics research. Over the years, the models have become overly complex and large with millions of parameters. For example, GPT-3 [4] has about 175 billion parameters. Yet, it remains largely unclear how the system comes to a decision, how certain the model is about its decision, if and when it can be trusted or when it has to be corrected. Explainable AI (XAI) is being extensively researched upon [10, 11, 24, 31, 35, 42, 43, 44] to get a better insight into AI made decisions to not only improve the models but also to instill accountability in them. The explanations could benefit users in many ways such as improving safety and fairness when relying on AI decisions, especially in safety critical applications *e.g.*, healthcare [7, 23], autonomous driving [32] or criminal justice [16].

XAI Strategies for deep visual models offer model specific [1, 28] or saliency map based explanations [11, 24, 27, 31, 44]. This work subscribes to the later where saliency maps highlight relevant regions in an input *e.g.*, image, video or natural language texts. Saliency maps are obtained either following a black box or a white/grey box technique. White/grey box approaches [31, 43, 44] either assume access to the internals of the base model (the model whose predictions are explained) or modify the base model which can cost the base model its performance [44]. This makes them unsuitable to be generalized over all architectures.

Black box methods [10, 11, 24, 42] on the contrary, do not touch the internals of the base model. Instead, the input image pixels are perturbed and passed through the base model. The output prediction resulting in from the perturbed image differentiates between the important and non-important regions as perturbing important or relevant regions affects the output score more than the case when non-important image regions are perturbed. A popular technique to achieve this is to view saliency maps

*agarwalsiddhant10@gmail.com

1st Workshop on eXplainable AI approaches for debugging and diagnosis (XAI4Debugging@NeurIPS2021).

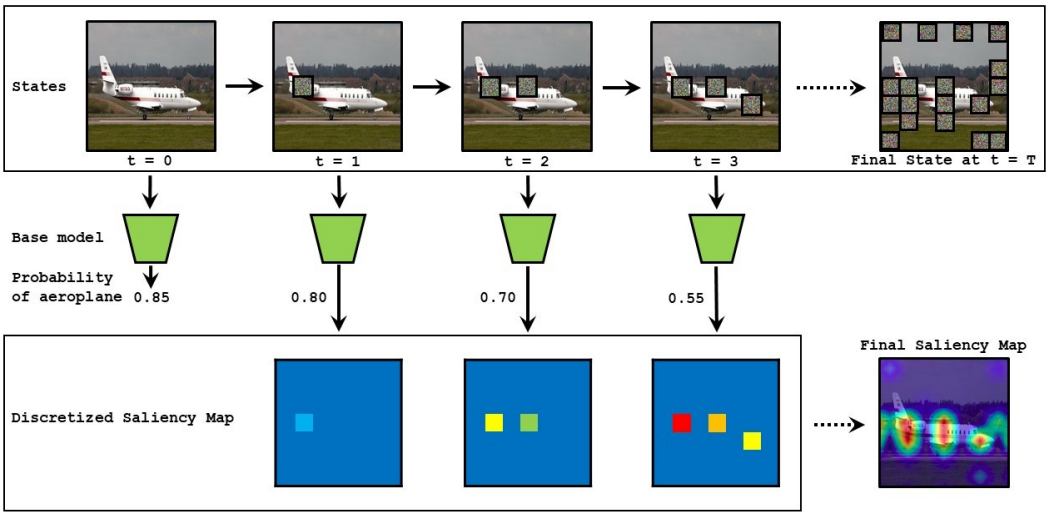

Figure 1: **Approach Overview:** At the start, the base model (the classifier whose decision is explained) provides the probability of a class (here *aeroplane*) in the unperturbed image. Subsequently, at every step, the agent learns to choose important regions of the image to be masked such that on passing the masked image through the base model, the probability drops. The drop in the probability gives the relevance of the masked region behind the base model's decision and is provided by the color-coded weight corresponding to the masked region in the saliency map (last row). The weight value increases from blue to red. Though initial importance of the chosen region at $t = 1$ is not high, our approach provides due credit to it via a cumulating factor for driving the agent to choose regions leading to greater drop subsequently. The final explanation is obtained by smoothening the discretized saliency map for better visualization. (Best viewed in color.)

as weighted masks. The perturbations to the input are performed by masking portions of the image using these weighted masks. Different methods differ in the way these masks are constructed. Fong *et al.* [10, 11] solve an optimization objective to construct optimal masks while RISE [24] uses large number of random masks for an image. A good explanation is associated with a low deletion score [24] which measures the drop in probability of a class as pixels are gradually removed according to the saliency map given importance.

In this paper, we address explainability as a sequential search problem where a sequence of image patches or regions are investigated in order to get the mask giving the saliency map. Specifically, an agent starting with an unperturbed image, learns to choose the best image region to perturb given the choices made earlier. The agent chooses the sequence of regions from the currently unperturbed portions of the image so that perturbing the region maximally drops the class probability which in turn, causes a low deletion score. An approximation of the exhaustive search is performed by RISE [24] and is shown to achieve good deletion score albeit being prohibitively expensive both in time and memory requirements. Our approach is fundamentally different as RISE does not make any conscious effort to get low deletion score whereas we *learn to intelligently search* for the mask with a goal to achieve as low deletion score as quickly possible. The goal oriented learning offered by the proposed method not only makes the inference fast but also makes it deterministic.

The core of the problem is to find a policy that assesses the image perturbed by the present mask and decides to perturb a new relevant image region. Figure (1) illustrates the decision making process of such an agent. At first, the unperturbed image is passed through the base classifier to get the probability of the class present in the image (an 'aeroplane' in Figure (1)) The agent then starts by choosing a patch from the image and replacing it with uniform random noise. The perturbed image is assessed by the base model to get the change in probability of the same class. Subsequently, the agent decides to pick the most important region so that perturbing it decreases the probability of the class further until the agent decides to stop exploring or the maximum number of patches is perturbed. As the perturbed regions contribute to a drop of class probability, these regions are pivotal evidences for the classifier to classify the image. The significance of the perturbed regions behind the base classifier's decision is manifested by the drop in probability. Thus at every step, the chosen region gets an weight proportional to the drop of the class probability. This weighted regions provide a

discrete weighted mask that after smoothening provides the saliency map explaining the decision of the classifier. As region perturbation is sequential and its effect is cumulative on the base model, we introduce a cumulating factor $\lambda \in (0, 1)$ that partially credits the regions perturbed earlier but instead of immediate drop, may have resulted in a significant drop in class probability later.

Learning such a dynamic agent capable of performing sequential actions towards a long-term goal by actively interacting with the environment in absence of any supervision is performed by Reinforcement Learning (RL). Motivated by its recent success [37, 41] coupled with its ease of use, we use ACER [40] to learn an RL agent giving the final saliency map. Our approach is equally capable of learning the optimal perturbing sequence on a single image, a class of images or a whole dataset of images. This makes our approach flexible in terms of scalability and generalization. To the best of our knowledge, we are the first to propose a RL based strategy for explaining AI made decisions and we term our approach as **R**einforcement **Ex**planation **L**earning (RExL).

We perform experiments on three popular benchmarks in Computer Vision, ImageNet [8], PASCAL VOC [9] and MSCOCO [17]. For each of these datasets, we use VGG16 [34] and ResNet50 [14] as the base models i.e. the classifiers who's decisions are to be explained. We compare our results with recent black box and white box methods. We show that RExL does not hurt the performance and provides competitive results. In fact it performs better than the baselines in a few cases. Finally we also show the speedup given by RExL compared to other the black box techniques.

2 Related Work

There have been a lot of work in Explainable AI [33, 36, 3, 24, 10, 31] recently. These have used different techniques to explain the decisions taken by Deep Learning models. Some of them [3, 6] leverage the use of Information Theory to maximize the mutual information between the compressed representation and the decision. [3], in addition, also reduces the mutual information between the explanation and the input to ensure that the explanation chooses the minimal subset of input features. These involve complex objective functions which are difficult to use.

Backpropagation Based Methods generate the important measure of a pixel by backpropagating the output of a deep neural network back to the input space using gradients or their variants. Some of them [2, 33] use the derivatives of a class score with respect to the image as an importance score. Guided Backprop [36] and DeConvNet [42] modify the backpropogation scheme for ReLU while Excitation Backprop [43] ensures that the sum of the attribution signal is unity. Integrated gradients [25] additionally accumulate gradients along a path from base image to input image. Activation based methods like CAM [44] (and its variants like Grad-CAM [31] and NormGrad [27]) use a linear combination of activation values across the convolutional layers. Grad-CAM and NormGrad uses gradients information to weigh the maps. These methods can generate saliency maps for any activation layer in the network but it is found that later layers, i.e. layers closer to the output generate more informative saliency maps.

These techniques achieve interpretability by making changes to a white-box model or utilising the internals of it. Thus, they cannot always be generalised over all model architectures. This demands for a need of black box techniques that can be used for any base model. LIME [28] is a black box approach which draws random samples around each instance to be explained and fits an approximate local linear decision model in the vicinity of the input. For complex non-linear classifiers, LIME fails to achieve good performance. Its dependence on superpixels leads to inferior saliency maps. Perturbation based methods perturb the inputs (blur some pixels, or add some noise etc) and measure the response of the model to these. [39] is such a method that also fine grains gradients to avoid adversarial examples. RISE [24] is a perturbation based method that generates a large number of random masks and applies them on the image. It then generates the saliency map as the weighted sum of these masks. The weights are the class scores predicted by the model after the mask is applied. It has been successful in generating decent saliency maps but it uses large number of masks (around 5000) which makes it computationally expensive and time taking. Meaningful perturbations [11] and Extremal perturbations [10] solve optimization objective on single images to generate explanations but these involve several iterations of every images and take time.

Reinforcement Learning is increasingly been adopted in standard computer vision tasks like Object Detection [38, 5, 19], Visual Question Answering [18], Action Recognition [12] etc. TRPO [29], PPO [30] and DQN [21] are the most commonly used RL algorithms. TRPO and PPO are on-policy

while DQN is off-policy. There have been lots of developments on Actor Critic Methods [20, 40, 13]. These include both off-policy algoritms [13] and on-policy methods [20]. There has been some work done [26] where RL agents are used to perturb the inputs to study model behaviour. In most cases, the agent chooses a particular perturbation on the input and then gets rewards from the outputs of a downstream model for the perturbed input. We follow a perturbation based explanation technique that uses RL to learn an optimal mask and generate accurate saliency maps quickly.

3 Proposed Approach

We now describe the proposed method which leverages Reinforcement Learning for getting saliency based explanation of a convnet classifier where at any time, most convincing evidences are collected depending on what evidences have already been collected.

3.1 Problem Definition

Formally, we deal with a dataset $\mathcal{D}$ of images where each image $\mathcal{I}$ can contain one or more objects. The base model or the model we want to explain provides the probabilities $p_c, \forall c \in \{1, 2, \cdots, C\}$ indicating the presence of at least one object of class c among C classes. For each object in the image, the explanation generation task is to output importance values for each pixel (or sub-images/patches) of the image. An explanation is good if gradually removing pixels from the image in order of decreasing importance score confuses the base model. This will be manifested by a sharp drop of the probability of the object compared to the case when all pixels were present.

We cast explanation generation as a Reinforcement Learning (RL) problem in a Markov Decision Process (MDP) setting. An MDP has a set of actions $\mathcal{A}$, a set of states $\mathcal{S}$ and a reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. At any time instant t, the agent selects an action $a_t \in \mathcal{A}$, executes it and as a result, the state changes from s_t to s_{t+1}. From the environment, the agent receives 1) a scalar reward $R(s_t, a_t)$ and 2) the next state in the form of an observation. The action to be taken by the agent is governed by a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that takes in environment states at any point of time and outputs a probability distribution over a set of actions. The goal of an RL agent is to learn an optimum policy π^* such that the sum of rewards through an episode is maximized. Next we outline the actions, states and rewards of the proposed RL setup.

States: For an image of size $H \times W$, the number of possible masks naturally, is $2^{H \times W}$ as each pixel can be either masked or not. The state at any time instant t is the masked image where masked pixels are replaced with random noise sampled uniformly in the range of the pixel intensity values. For a reasonably sized image, the state-space and thus the search-space of size $2^{H \times W}$ can be huge. Thus we relax the state-space size by discretizing the image into a $K \times K$ grid, (K being 7 or 14).

Actions: Denoting the probability given by the base model for the image to belong to class $c \in \{1, 2, \cdots, C\}$ as p_c, the actions taken by the agent aims to reduce p_c to a minimum. The agent chooses regions to construct the mask sequentially so that the base model provides a low probability or a low score for the masked image to belong to that class. It should be noted that theoretically, the agent can choose an already masked region also. However, a well trained agent would find no incentive in performing such an action as the probability value will not reduce much (if at all, it reduces) as a result of masking an already masked region. For an image divided as $K \times K$ grid, the number of possible actions is naturally K^2. There seems to be an incentive in terms of training computation to stop the search early by limiting the agent to take only a preset number of actions. However, this may restrict the exploration ability of the agent. Thus, to strike a balance, we allow the agent to take as many actions as there are number of grids *i.e.*, K^2.

Rewards: The reward function quantifies the worth of the agent's effort for successfully masking important image regions. A successful masking operation in turn, provides a good saliency map explaining the decision behind predicting the presence of a class in the image. This is manifested by the drop in probability of a class present in it. As a natural choice, thus, the reward is taken to be the negative of the probability given by the base model for the class when the perturbed image is passed through the base model. It is given by,

$$R(s_t, a_t) = -p_c^{(t)} \tag{1}$$

where s_t is the state and a_t is the action executed at timestep t. $p_c^{(t)}$ is the probability of the image to belong to class c, when the masked image at timestep t is passed through the base model. To

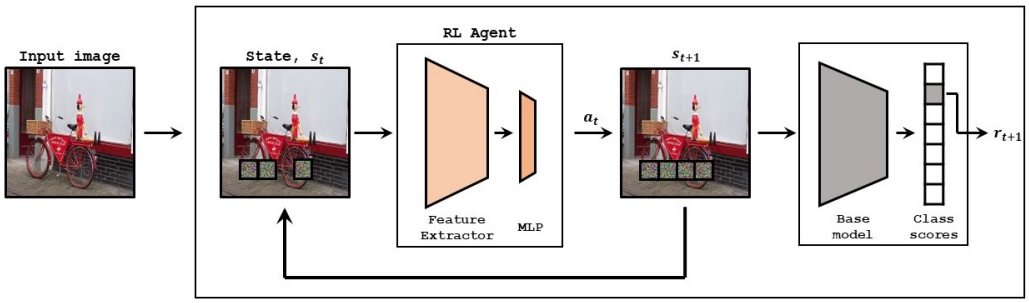

Figure 2: **Training the Reinforcement Learning Agent for Explaining Image Classification Model**. The state, s_t, at time t is a partially masked image. The RL Agent, which consists of a fixed feature extractor and a learnable MLP, chooses an action a_{t+1}. The action is the region of the image that needs to be masked. The result of the action is the next state i.e. s_{t+1}. The base model evaluates s_{t+1} to provide the probability of the class being explained which is used to construct the reward r_{t+1}.

maximize the discounted return $G = \mathbb{E}\big[\sum_t \gamma^t R(s_t, a_t)\big]$, the agent will have to choose the regions such that the classification score of the class drops as quickly as possible thereby minimizing the deletion score. The discount factor $\gamma \in (0, 1]$ determines the present value of future rewards. As $\gamma \to 0$, the agent tries to maximize only the immediate rewards, while a value close to 1 means the agent becomes farsighted.

3.2 Solution Strategy

The goal of the agent is to choose relevant regions of the input image to mask such that the probability or the score of the masked image drops as quickly as possible during the interactions with the environment until termination. The core of the problem is to find the policy function π mapping states to actions that guides the agent's decision making process. We use a recently proposed Actor Critic with Experience Replay (ACER) [40] which is an improvement to the traditional actor-critic algorithm that is more sample efficient.

The policy function of the agent is represented by a multi-layer neural network with learnable parameters θ. An episode in an RL setting is composed of all the state action pairs from start to the terminal action and is denoted as $\tau = (s_1, a_1, s_2, a_2, \cdots s_T, a_T)$ where T is the number of steps taken before the termination of the episode. The agent learns the optimal policy by maximizing the expected return $J(\theta)$ over the possible trajectories.

$$J(\theta) = \mathbb{E}_{p_\theta(\tau)}\big[\sum_t \gamma^t R(s_t, a_t)\big] = \mathbb{E}_{p_\theta(\tau)}\big[\sum_t \gamma^t r_t\big] \tag{2}$$

$R(s_t, a_t)$ is abbreviated as r_{t+1} above to reduce clutter. The objective function in Equation (2) can be directly optimized using standard optimization techniques like SGD but suffers from high variance especially under small samples resulting in very fragile convergence. With the help of the state and action value functions, the Actor Critic algorithms [15] overcome this limitation. While the state value function $V^\pi(s)$ gives the expected return given the agent is at state s and follows the policy π, the action value function $Q^\pi(s, a)$ is the expected return given that the agent performs an action a at state s and follows the policy π thereafter. The modified objective in terms of the advantage A^π is given by,

$$J(\theta) = \mathbb{E}_{p_\theta(\tau)}\big[\sum_t A_t^{\pi_\theta}\big] \qquad \text{where } A_t^{\pi_\theta} = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t) \tag{3}$$

The basic objective function (Equation (3)) optimized by the Actor-Critic methods is inherently on policy. ACER's objective contains the both off-policy and on-policy components. For the off-policy component, it uses Retrace algorithm [22] to estimate the action value function and further uses importance sampling. The on-policy part is very similar to the basic objective shown in Equation (3).

An illustration of our proposed approach RExL is provided in Figure (2). We provide a saliency map based explanation to the classification decision made by the base model on an image. The image is given as the input. The RL agent of RExL consists of a pretrained feature extarctor and a multi-layer

perceptron (MLP). The feature extractor remains fixed during the training and the learnable MLP is the policy network to predict the action. At every step, the agent observes the perturbed image which constitutes it's state and predicts the next action. The action is a specific region in the image to mask. Once the image is perturbed or the action is executed, the base model gives the classification score for the class to be explained. This confidence score is used as reward during training (ref. Equation (1)) and also to generate the explanations as saliency maps. The perturbed image becomes the next state and the loop continues until termination.

RExL gives the flexibility to train a single agent to explain all images of a dataset (*dataset specific*), multiple agents each one expertly trained to explain a single class (*class specific*) or extremely fine-grained agents that are experts in explaining a single image (*image specific*). We train the dataset specific agents by using the whole training partition of the dataset in which the base model was trained. A class specific agent is trained on a single class of training images. Finally, an image specific agent is trained only on the image to be explained. The explainability of image specific agents are very good but this is not scalable. Although dataset specific agent is scalable and performs good for images containing mostly single objects, the performance may degrade for a complex dataset with images having multiple instances from possibly multiple classes. As our experiments show, the best trade-off between performance and scalability can be seen for class specific agents.

3.3 Inference

After a model is trained, it is used to generate saliency maps. Given an image, the agent sequentially deletes a block of the grid from it and incrementally builds the saliency map. Let block b_t of an image is perturbed as a result of the action a_t taken by the agent at time t and the change in classification confidence from the previous state is δ_t. The saliency weight assigned to block b_t is $\sum_{i=t}^{T} \lambda^{i-t}\delta_t$, where $\lambda \in [0, 1]$ is the cumulating factor that gives the due credit to previous deletions that may have resulted in a greater drop in probability at a later stage. A λ value equal to 0 is the trivial case where only the grid that is deleted most recent gets all the credit. Finally, the map is normalized so that the saliency weights sum to 1.

4 Experiments

Datasets and Evaluation Metrics: We evaluate the performance of our proposed method on three benchmark datasets namely ImageNet [8], PASCAL VOC 2007 [9] and MSCOCO 2014 [17]. Following the standard practice, we have reported the results from the *val*, *test* and *val* splits respectively. For all the three datasets, we use pretrained VGG16 [34] and ResNet50 [14] as base models. For ImageNet, we used the trained models provided by Pytorch Models Zoo [1]. For PASCAL VOC and MSCOCO, we used the same multilabel models as RISE and made public by [43].

We follow [24] and use the causal deletion and insertion metrics to compare the performance with different approaches [24, 10, 27, 31].While GradCAM [31] and NormGrad [27] are white box techniques that make use of the intermediate activations and gradients of the network, RISE [24] and Extremal Perturbations [10] are black box techniques that only study the input-output relationship of the model. RISE is a sampling based method and samples thousands of random masks and weighs them based on their performance. Extremal Perturbations on the other hand is a learning based method which solves an optimization objective for each image to obtain the corresponding mask.

Deletion metric removes pixels from the image gradually according to the importance weight given by the saliency map and measures the classification score. The deletion score is the area under the curve (AUC) of these classification scores with percentage of pixels removed. Similarly the insertion score evaluates the performance inversely *i.e.*, uncovering highly blurred image regions gradually according to the importance weights and getting the AUC. While a low deletion score is indicative of better explanation, in case of insertion score the opposite is true. Though, RExL follows the deletion philosophy while training, the insertion score for it is also competitive. We posit that an agent trained with an insertion strategy (*i.e.*, starting with a blank canvas and learning to choose relevant regions to uncover) would be equally useful and we keep it as a possible future work. While our primary aim was to improve on the inference time of black box model, we first show that RExL does not hurt the

[1] https://pytorch.org/vision/stable/models.html

Method	ImageNet		PASCAL VOC 2007		COCO 2014	
	VGG16	ResNet50	VGG16	ResNet50	VGG16	ResNet50
GradCAM [31]	0.109	0.123	**0.275**	0.310	**0.225**	0.261
NormGrad [27]	**0.086**	0.126	0.285	0.349	0.232	0.303
RISE [24]	0.098	**0.108**	**0.275**	0.290	0.240	0.227
Extremal [10]	0.125	0.126	0.355	0.357	0.292	0.299
RExL-DS	0.130	0.122	0.342	**0.243**	0.284	**0.200**

Table 1: **Comparison of Deletion scores on Imagenet, PASCAL VOC 2007 and COCO 2014 Datasets**. RExL-DS outperforms white box methods on VOC 2007 and COCO 2014 for ResNet50 base model. For a less strong VGG16 base model, RExL-DS is competitive.

performance, infact in some cases, it performs better than even white box methods. Thereafter we provide a runtime analysis showing the significant speedup given by us compared to similar black box XAI strategies.

Implementation Details: The policy feature extractor can be any good pretrained feature extractor and is independent from the base model. We use ResNet50 pretrained on the dataset the RL agent being trained on as the feature extractor for the policy. The learnable MLP consists of two hidden layers with 256 and 128 units respectively. We fix the value of K as 7 and the length of the episode, thus, will be $7^2 = 49$. We trained ACER [2] for 2×10^6 steps when training class specific and 10^7 steps when training dataset specific and used discount factor, $\gamma = 1$ throughout our experiments. Also for all the inferences, we have used $\lambda = 1$ unless explicitly specified. As dataset specific setting is a single agent to explain all the classes in the dataset, information about the class to be explained was passed as an one-hot encoding along with the image during training. We used NVIDIA GeForce RTX 2080Ti GPUs for training all our models. Next we discuss the experimental results for all the three variants of RExL *i.e.*, dataset specific (*RExL-DS*), class specific (*RExL-CS*) and image specific (*RExL-IS*). Due to limitations in space, we provide the comparison over deletion metric only. Additional results including insertions scores, more implementation details and source codes are provided in the supplementary material.

4.1 Results and Discussions

Table (1) shows a comparative evaluation of the deletion scores on the three benchmark datasets with two different base models explaining the classes present in the images. RExL-DS performs better when explaining a stronger model like ResNet50. For PASCAL VOC 2007 and COCO 2014 datasets, RExL-DS outperforms all the baselines (including the white box approaches like GradCAM and NormGrad) while for ImageNet, its performance is at par (slightly better than most of the other baselines). With VGG16, the performance is at par with the competing methods. The comparatively higher deletion scores of RExL-DS for VGG16 is intuitive as this being an inferior base model the reward for a right choice of image region to delete may not always be right.

Method	VGG16	ResNet50
RExL-DS	0.342	0.243
RExL-CS	0.292	0.216

Table 2: **Comparison of the Deletion scores of RExL-DS and RExL-CS on VOC2007 test split**. The deletion score is improved for the more specific RExL-CS agent compared to RExL-DS for both the base models.

As we go to class specific RExL agents (ref Table (2)), our approach becomes more competitive for VGG16 base model while further improvement is seen in case of ResNet50 making RExL-CS the new SOTA for PASCAL VOC. In case of Imagenet, due to the presence of a large number of classes, we focused on explaining the top 10 classes for which VGG16 and ResNet50 classifiers give best performance in terms of top-1 accuracy. The average deletion score of the RExL-CS agents trained on these 10 classes for ResNet50 base model is as low as 0.111 while RISE being the closest with a value of 0.149. Likewise, for VGG16, the proposed approach gives a value of 0.139 with RISE being close with a deletion score of 0.149. The improvement over RExL-DS shows the flexibility of our approach as well as the expertise imbibed by the class specific models compared to one single agent trained to explain all the classes in the dataset. Additional results on both Imagenet and PASCAL VOC are provided in the supplementary.

[2] Implementation: https://stable-baselines.readthedocs.io/en/master/modules/acer.html

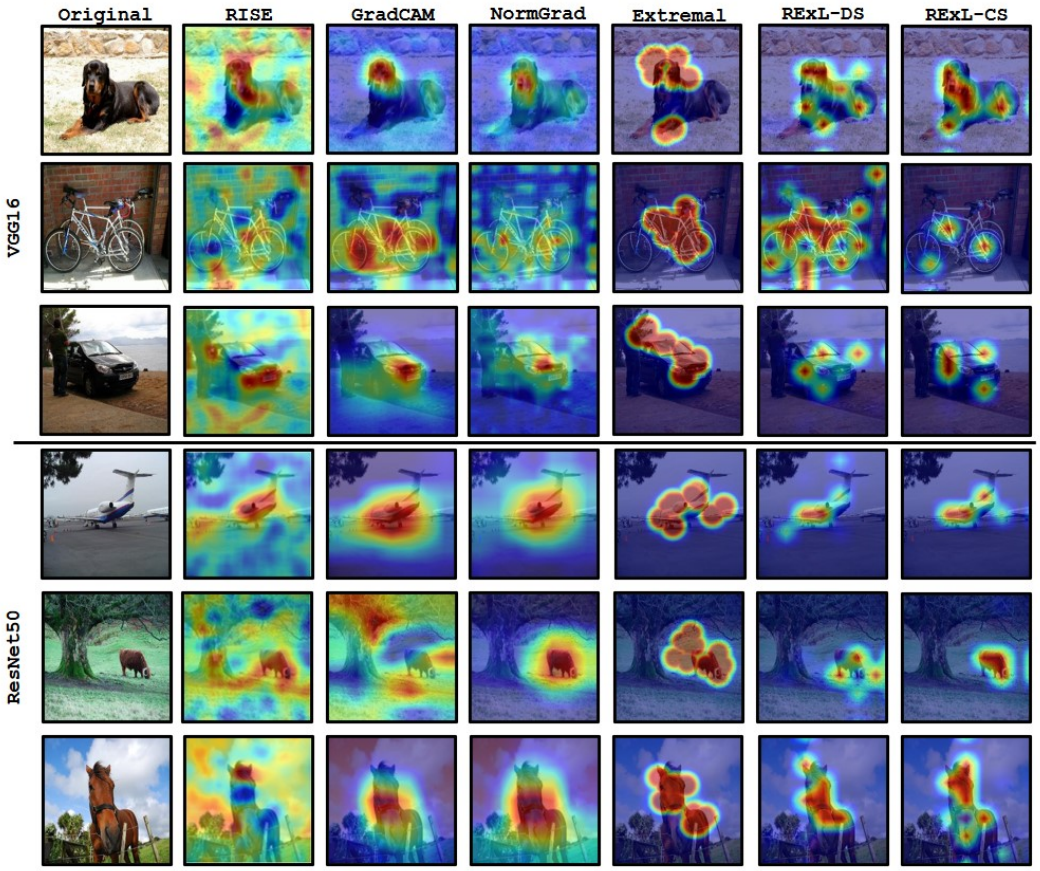

Figure 3: **Comparison of the saliency maps**: The top three rows show saliency maps for 3 random samples from VOC2007 when RExL agent is applied on VGG16 base model. Likewise the bottom 3 rows correspond to ResNet50 base model. The leftmost column shows the original image. Rest of the column headings show the approach used to generate the respective saliency maps. The results show the superiority of the proposed approach over the black box and white box approaches as well as the specificity of the maps by the RExL-CS approach compared to the RExL-DS approach.

Figure (3) shows a comparison between the saliency maps generated by different baseline methods and the two variants of RExL. Targeted saliency and much less noise of the RExL maps make them qualitatively better compared to the others. While extremal perturbation [10] generates less noisy maps, its dependence on manually set area of relevant regions can harm if the object of interest does not conform to this preset value (pronounced in the top row for VGG16 and penultimate row for ResNet50). This is especially detrimental for RISE where the probability of the masked regions in the image is fixed likewise. RExL, on the other hand, learns to mask regions that is important for making the base model decide the class without constraining the area to be prespecified. Improvement by our method is also observed when the specificity of the agent increases and it becomes more expert *i.e.*, when we go from dataset-specific to class-specific agents. This is especially prominent for the image of the cycle in case of VGG16 and for the image of the cow in case of ResNet50.

Although RExL-IS is the extremely specific, time to train image specific models for all images is also high. As a result, we don't specifically evaluate RExL-IS on all images and provide quantitative results. Instead, we do qualitative comparison with existing approaches on a few images along with a comparison of the deletion scores. Due to space limitation we provide additional qualitative examples including RExL-IS comparisons in the supplementary.

4.2 Comparison of Inference Time

The major advantage of and motivation for RExL is the targeted fast search of relevant image regions without degrading the quality of the explanations. In this section we compare the inference times of

Method	ImageNet		PASCAL VOC		MSCOCO	
	VGG16	ResNet50	VGG16	ResNet50	VGG16	ResNet50
RExL-CS	1.461	1.494	1.748	1.488	1.738	1.528
RExL-DS	1.869	1.872	1.754	1.532	1.800	1.519
RISE [24]	14.618	8.853	20.408	8.370	20.397	8.392
Extremal [10]	13.722	15.975	19.114	16.298	19.118	16.298

Table 3: **Comparison of Running Times (in seconds) while generating saliency maps**. RExL-DS and RExL-CS are compared with the related black box methods (RISE and Extremal Perturbations). Both variants of RExL outperforms the competing methods by a large margin.

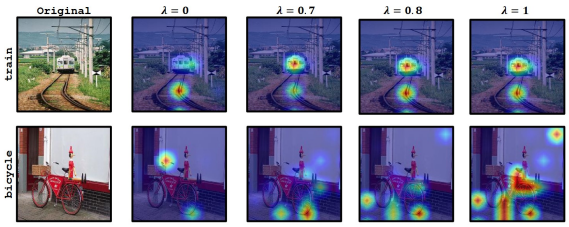

Figure 4: **Comparison between the saliency maps produced by RExL-DS for different values of the cumulating factor λ**. The λ values are put on the top. The leftmost column shows the original image from the PASCL VOC 2007.

Inference	VGG16	ResNet50
$\lambda = 0$	0.362	0.290
$\lambda = 0.7$	0.349	0.266
$\lambda = 0.8$	0.347	0.260
$\lambda = 1$	0.342	0.243

Table 4: **Comparison of the deletion scores for the different values of the cumulating factor λ** during inference on VOC2007 test split using RExL-DS.

RExL with that of the state-of-the-art black box explanation techniques *e.g.*, RISE and the extremal perturbation approaches.We do not compare the running time of RExL with white box techniques like GradCAM and NormGrad because they are much quicker but come at the cost of peeking into the model. Our aim is to bring about considerable speedup in the inference time of a black box explanation technique without hurting the performance.

For comparing the running times, we use a computer with one NVIDIA GeForce GTX 1080Ti, ryzen 2700x CPU and 16GB DDR4 3000MHz RAM. Table (3) shows the average running times over 100 images. RExL provides almost $10\times$ speedup over both the approaches for VGG16 base model. The improvement in speed is almost $5\times$ and $10\times$ respectively over RISE and extremal perturbation in case of ResNet50 base model. RExL-CS is comparatively fast than RExL-DS as RExL-DS gets an additional class encoding as input. The faster running time with comparable or better performance shows the merit of our RL based explainable AI approach.

4.3 Ablation on Cumulating Factor

To see the effect of the cumulating factor λ while assigning credit to the chosen image regions we experimented with different values of it for RExL-DS on the PASCAL VOC dataset. Table (4) shows the comparison. Though the values are close, one important observation is that giving the entire credit at a certain instant to the region that is deleted most recently (*i.e.*, $\lambda = 0$) hurts the performance. As $\lambda = 1$ provides the best performance, we went ahead in experimenting with this value for all the datasets. Figure (4) shows sample saliency maps with different λ values applied on the same image. For a large object, naturally, a large portion of the image is responsible for the base model to get its decision and this is rightly captured when credit assignment is delayed instead of immediate.

5 Conclusions

We have presented a reinforcement learning based model for explaining deep classifiers. In contrast to methods that approximate an exhaustive space, we propose to formulate XAI as an intelligent search and direct an agent to efficiently choose and accumulate evidences responsible behind AI made decisions. Our model is dataset, category and image specific striking a perfect balance of model specificity and training burden. Ours is a plug and play black box approach that is applicable to any classifier without requiring to modify the internals of them. We report encouraging results in three benchmark datasets achieving 5 to 10 times speed-up over related black box XAI approaches.

References

[1] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Deep compositional question answering with neural module networks. *CoRR*, abs/1511.02799, 2015.

[2] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, August 2010.

[3] Seo-Jin Bang, Pengtao Xie, Wei Wu, and Eric P. Xing. Explaining a black-box using deep variational information bottleneck approach. *CoRR*, abs/1902.06918, 2019.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165, 2020.

[5] Juan C. Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. *CoRR*, abs/1511.06015, 2015.

[6] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. *CoRR*, abs/1802.07814, 2018.

[7] Devam Dave, Het Naik, Smiti Singhal, and Pankesh Patel. Explainable AI meets healthcare: A study on heart disease dataset. *CoRR*, abs/2011.03195, 2020.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[10] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[11] Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[12] Shreyank N. Gowda, Laura Sevilla-Lara, Frank Keller, and Marcus Rohrbach. CLASTER: clustering with reinforcement learning for zero-shot action recognition. *CoRR*, abs/2101.07042, 2021.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[15] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Neural Information Processing Systems*, pages 1008–1014. Citeseer, 2000.

[16] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica*, 2016.

[17] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[18] Feng Liu, Tao Xiang, Timothy M. Hospedales, Wankou Yang, and Changyin Sun. Inverse visual question answering: A new benchmark and VQA diagnosis tool. *CoRR*, abs/1803.06936, 2018.

[19] Stefan Mathe, Aleksis Pirinen, and Cristian Sminchisescu. Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[22] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. *CoRR*, abs/1606.02647, 2016.

[23] Urja Pawar, Donna O'Shea, Susan Rea, and Ruairi O'Reilly. Explainable ai in healthcare. In *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–2, 2020.

[24] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-box Models. In *British Machine Vision Conference*, Sep 2018.

[25] Zhongang Qi, Saeed Khorram, and Fuxin Li. Visualizing Deep Networks by Optimizing with Integrated Gradients. *CoRR*, abs/1905.00954, 2019.

[26] Mrigank Raman, Siddhant Agarwal, Peifeng Wang, Aaron Chan, Hansen Wang, Sungchul Kim, Ryan A. Rossi, Handong Zhao, Nedim Lipka, and Xiang Ren. Learning to deceive knowledge graph augmented models via targeted perturbation. *CoRR*, abs/2010.12872, 2020.

[27] Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, Hakan Bilen, and Andrea Vedaldi. NormGrad: Finding the Pixels that Matter for Training. *CoRR*, abs/1910.08823, 2019.

[28] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016.

[29] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[31] 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 *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[32] Yuan Shen, Shanduojiao Jiang, Yanlin Chen, Eileen Yang, Xilun Jin, Yuliang Fan, and Katie Driggs Campbell. To explain or not to explain: A study on the necessity of explanations for autonomous vehicles. *CoRR*, abs/2006.11684, 2020.

[33] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[35] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

[36] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

[37] Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve J. Young. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. *CoRR*, abs/1707.00130, 2017.

[38] Burak Uzkent, Christopher Yeh, and Stefano Ermon. Efficient object detection in large images using deep reinforcement learning. *CoRR*, abs/1912.03966, 2019.

[39] Jorg Wagner, Jan Mathias Kohler, Tobias Gindele, Leon Hetzel, Jakob Thaddaus Wiedemer, and Sven Behnke. Interpretable and fine-grained visual explanations for convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[40] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *CoRR*, abs/1611.01224, 2016.

[41] Gellért Weisz, Pawel Budzianowski, Pei-Hao Su, and Milica Gasic. Sample efficient deep reinforcement learning for dialogue systems with large action spaces. *CoRR*, abs/1802.03753, 2018.

[42] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *CoRR*, abs/1311.2901, 2013.

[43] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *Int. J. Comput. Vision*, 126(10):1084–1102, October 2018.

[44] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

Supplementary Material

1 Implementation Details

In this section, we discuss all the details that are needed to reproduce our results as well as the scores obtained for the baseline methods. These parameters can be categorized as follows:

1.1 Environment and Agent Parameters

The mask is discretized into a $K \times K$ grid as discussed in section 3 of the main paper. For our experiments we choose $K = 7$. Naturally, the number of steps in an episode as discussed will be 49. For VOC 2007 and COCO 2014, we used VGG16 and ResNet50 trained by [6]. These models are trained on BGR images loaded with pixel values in $[0, 255]$ and then mean corrected to $[104.01, 116.67, 122.68]$ respectively. For ImageNet, we use the pretrained models provided by Pytorch model zoo [1] and used the standard normalization practices described there. The random noise that is used to mask the image is sampled from the same distribution as that of the image. For example in case of VOC or COCO, the noise has a mean $[104.01, 116.67, 122.68]$.

The RL agent consists of two parts, a feature extractor which is fixed during training and a learnable MLP. The feature extractor used is a pretrained ResNet50. So, for VOC2007, we will be using the ResNet50 module that was pretrained on VOC2007 by [6]. The learnable MLP consists of two hidden layers of 256 and 128 units each. The activation functions at these hidden units is ReLU. The output layer predicts the catergorical distribution over actions and hence contains $49(K^2)$ output neurons.

Parameter	Value
discount factor, γ	1.0
Number of environment steps per update	490
weight for loss of Q value	1.0
RMS prop alpha	0.9
learning rate	0.0001

Table 1: RL training hyperameters

During inference, as reported in section 4.4, $\lambda = 1$ gives the best performance so we report the results using $\lambda = 1$. RExL agents are trained on the respective partitions of the train split and following standard practices [3, 2, 1, 6] tested on the val, test and val split of ImageNet, VOC and COCO.

1.2 RL training parameters

We used ACER[2] to train the policy. The parameters that we used during training are in Table (1). We used the default hyperparameters as mentioned in the official documentation[3] for the rest of the parameters.

1.3 Baselines

For RISE we used the official implementation[4] with the hyperparameters discussed in [3]. For GradCAM and NormGrad, we used the publically available implementation[5]. Finally for Extremal

[1] https://pytorch.org/vision/stable/models.html

[2] https://stable-baselines.readthedocs.io/en/master/_modules/stable_baselines/acer/acer_simple.html#ACER

[3] https://stable-baselines.readthedocs.io/en/master/modules/acer.html

[4] https://github.com/eclique/RISE

[5] https://github.com/ruthcfong/TorchRay/tree/normgrad

1st Workshop on eXplainable AI approaches for debugging and diagnosis (XAI4Debugging@NeurIPS2021).

Method	ImageNet		PASCAL VOC 2007		COCO 2014	
	VGG16	ResNet50	VGG16	ResNet50	VGG16	ResNet50
GradCAM [5]	0.615	0.677	0.850	0.850	0.713	0.740
NormGrad [4]	0.517	0.455	0.795	0.807	0.640	0.690
RISE [3]	0.666	0.727	0.764	0.782	0.631	0.701
Extremal [1]	0.641	0.690	0.871	0.870	0.718	0.750
RExL-DS	0.464	0.529	0.755	0.755	0.592	0.678

Table 2: **Comparison of Insertion scores on Imagenet, PASCAL VOC 2007 and COCO 2014 Datasets**.

Perturbations, we used the implementation in the official repository[6] using contrastive rewards and area parameter of 0.12 while generating explanations.

2 Insertion Scores

RExL gets competitive scores for all the datasets even though the preocess of insertion is complementary to our training process. A method that optimizes deletion scores will not necessary also give good insertion scores. For deletion score to be low, the agent will learn the mask that will remove some features that increases the confusion in the model for that class. But inserting the same features may not necessarily lead to a high probability for that class. For example, in ImageNet, there are a lot of classes pertaining to different kinds of dogs. There will be some features (like the some portion of the face) on removal of which, the base model will get confused among all the other dogs. But if we insert this feature on a blank image, the base model will not necessarily classify it correctly.

Method	VGG16	ResNet50
RExL-DS	0.755	0.755
RExL-CS	0.778	0.777

Table 3: **Comparison of the Insertions scores of RExL-DS and RExL-CS on VOC2007 test split**. The deletion score is improved for the more specific RExL-CS agent compared to RExL-DS for both the base models.

Naturally this will be more pronounced where the classes are a lot similar to each other. As a result, (ref. Table (2)) RExL does not do as well in terms of insertion scores (compared to) on ImageNet but does very well on VOC where the classes are a lot different. Similar to the deletion score, insertion score for RExL-CS is better than RExL-DS, as seen from Table (3). RExL-CS results will be further analyzed in the next sections.

Inference	VGG16	ResNet50
$\lambda = 0$	0.783	0.785
$\lambda = 0.7$	0.771	0.769
$\lambda = 0.8$	0.768	0.765
$\lambda = 1$	0.755	0.755

Table 4: **Comparison of the insertion scores for the different values of the cumulating factor $\boldsymbol{\lambda}$** during inference on VOC2007 test split using RExL-DS.

One interesting observation in Table (4) is about the general trend that shows insertion scores are better for lower values of λ. This is counterintuitive to the trend seen in case of deletion scores (Table 4 in the main paper). For an object, the lower values of λ credit the later actions more while the larger λ value the earlier actions. The results show that the later actions are more important for insertion scores. These are the actions that are probably removing the significant portions of the object while the earlier actions aim to remove the features that separate similar classes. We posit that using an *insertion-style* training can lead to agents which will do better in terms of insertion scores too. This is a possible future work.

[6] `https://github.com/facebookresearch/TorchRay`

3 RExL-CS on VOC2007

For VOC2007, we train RExL-CS on all the 20 classes. Saliency maps generated by RExL-CS is compared with the baseline methods for every class. For quantitative comparisons, we use both the deletion scores and insertion scores. The maps generated by RExL-CS and baselines for every class are also compared qualitatively.

3.1 VGG16

Figure (1) and (2) show the comparison of class wise deletion and insertion scores among the different methods. Moreover, Figure (3) and Figure (4) qualitatively compare the saliency maps generated by RExL-CS with the baselines for every class. VGG16 gives competitive results for both deletion and insertion scores. The saliency maps produced are also more accurate and concise.

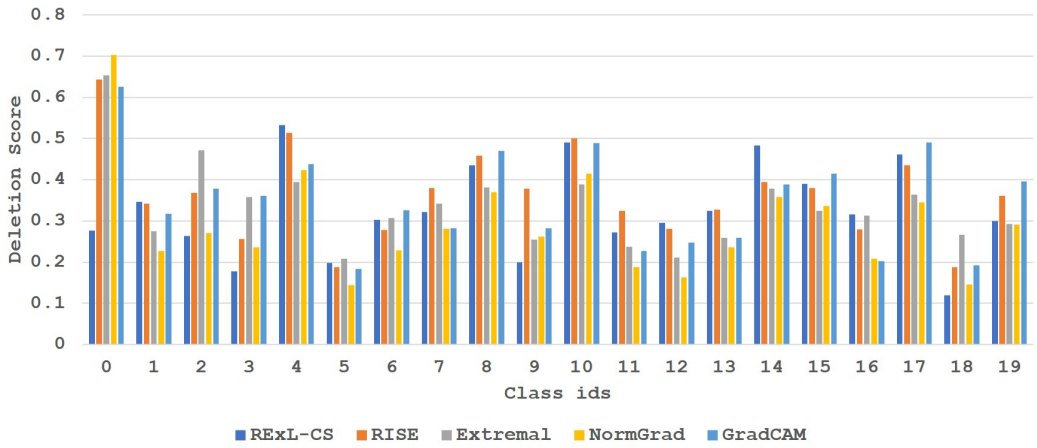

Figure 1: **Comparison of deletion scores (the lower the better) for VGG**16 **on all classes of VOC**2007**:** RExL-CS gets competitive deletion scores for all the classes. It outperforms all the baselines for 5 classes.

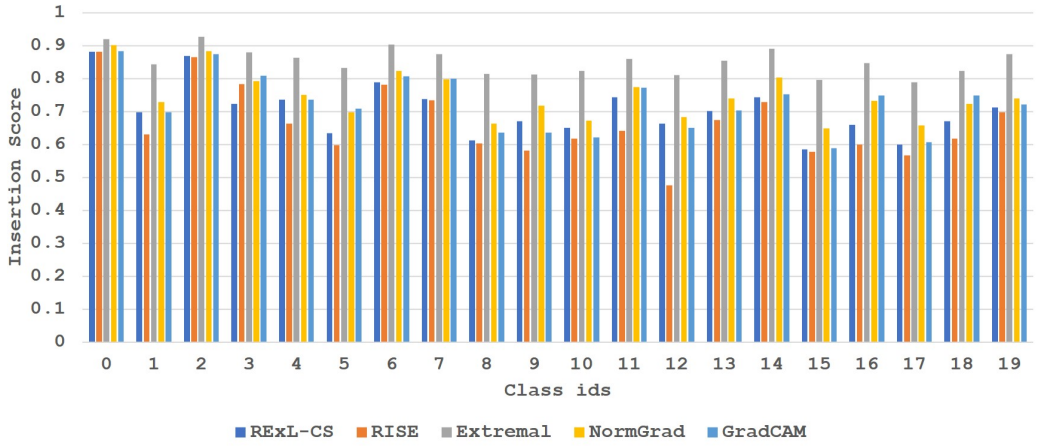

Figure 2: **Comparison of insertion scores (the higher the better) for VGG**16 **on all classes of VOC**2007**:** Insertion scores of RExL-CS are at par with the baselines. On average it has the better insertion score than RISE.

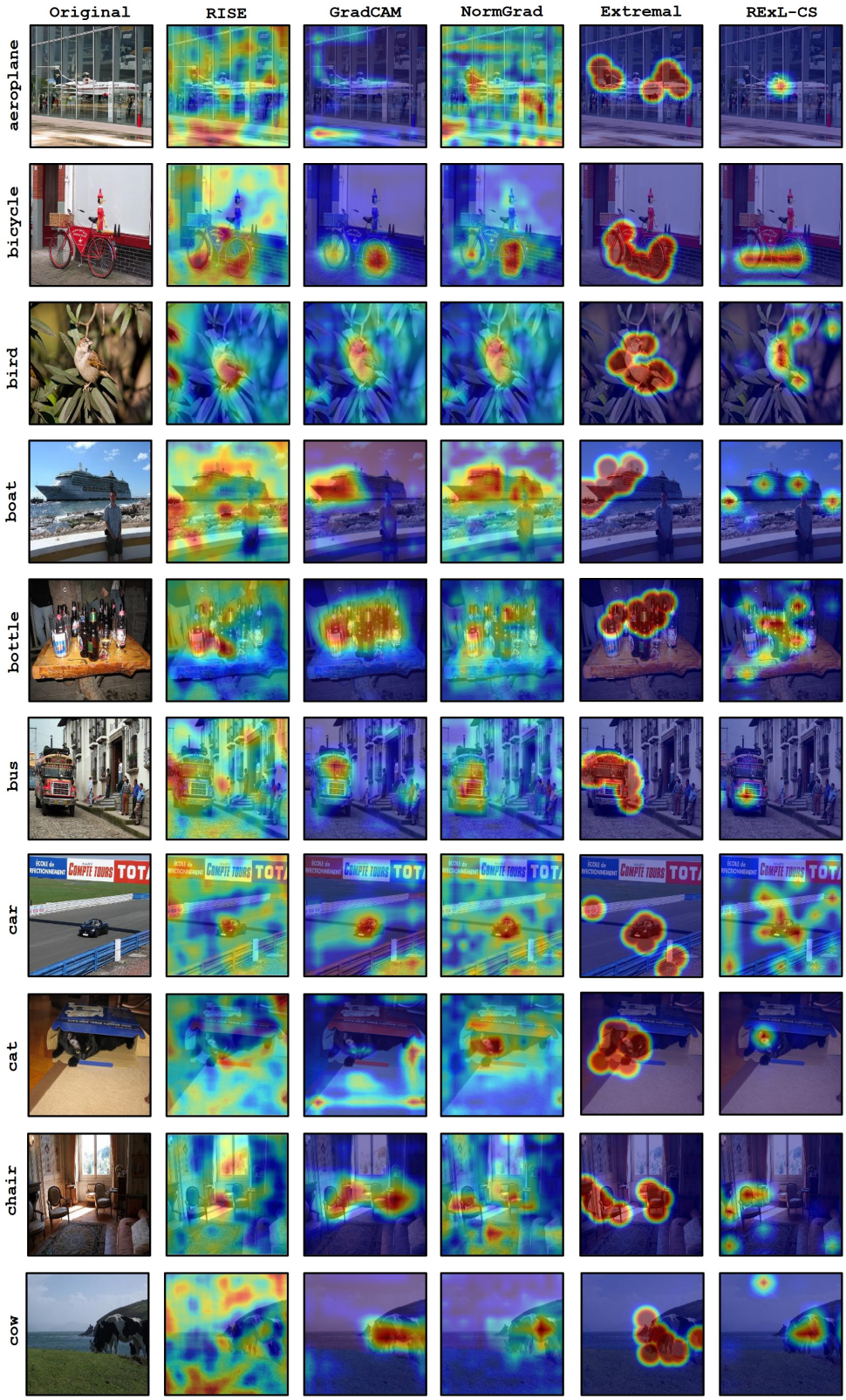

Figure 3: **Qualitative comparison of saliency maps for VGG**16 **on classes** $0-9$ **of VOC**2007**:** RExL-CS always generates better maps than RISE. Extremal perturbations gives almost equal importance to the entire object but its is difficult to identify the sailient features of the object. RExL-CS is at par with the whitebox methods in most cases and definitely better in some cases like "aeroplane" and "cat".

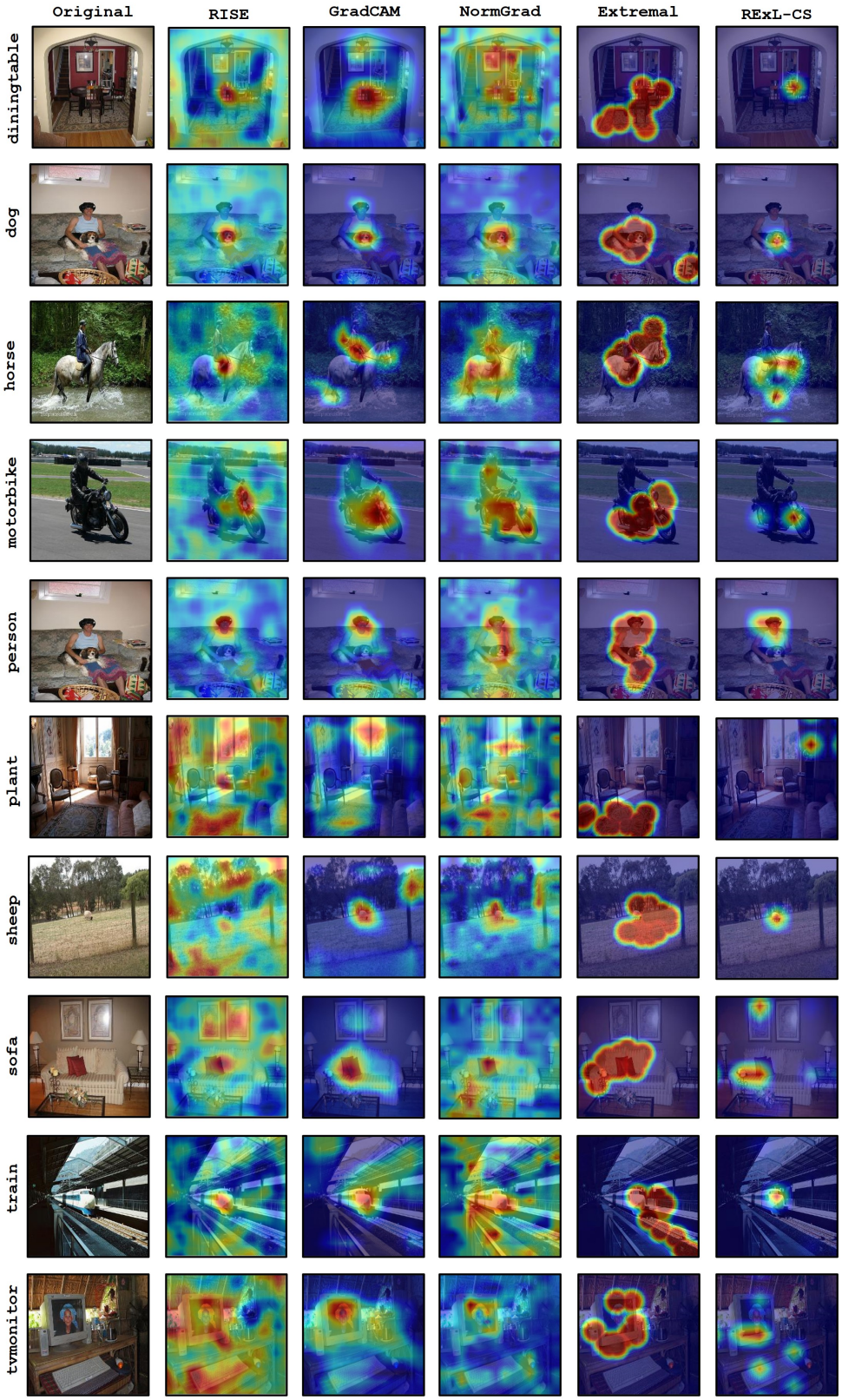

Figure 4: **Qualitative comparison of saliency maps for VGG**16 **on classes** 10 – 19 **of VOC**2007**:** Similar to Figure (3), RExL-CS generated better maps than both RISE and Extremal Perturbations. RExL-CS outperforms all the baselines for classes like "plant" and "tvmonitor".

3.2 ResNet50

Similarly, RExL-CS agents are trained for all the 20 classes using ResNet50 as the base model. Figure (5) and Figure (6) respectively show the comparison of deletion and insertion scores for every class. As seen in all the results, RExL-CS improves considerable as we use a stronger base model like ResNet50. Finally, Figure (7) and Figure (8) contains the saliency maps generated by the different methods for qualitative comparison.

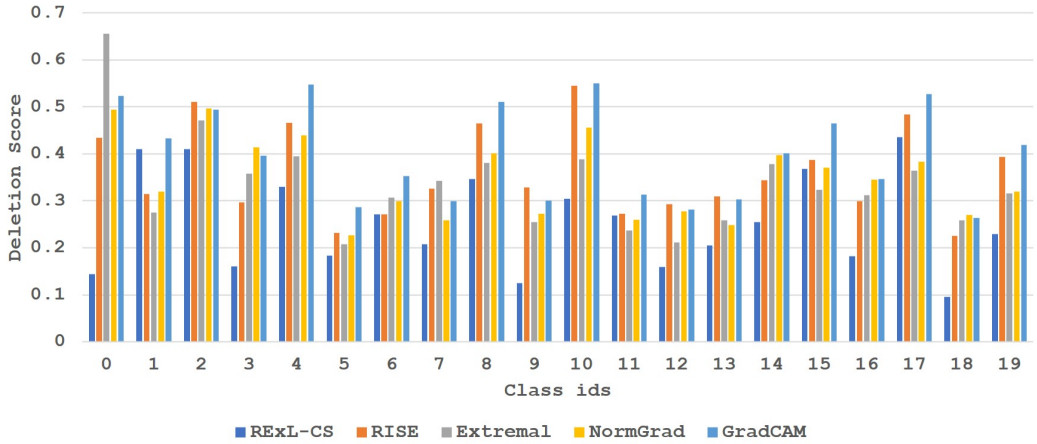

Figure 5: **Comparison of deletion scores (the lower the better) for ResNet**50 **on all classes of VOC**2007**:** RExL-CS outperforms the baselines most of the times (in 15 classes). It performs exceptionally well on some classes like "aeroplane" (class id 0) and this will be further highlighted in the saliency maps in Figure (7).

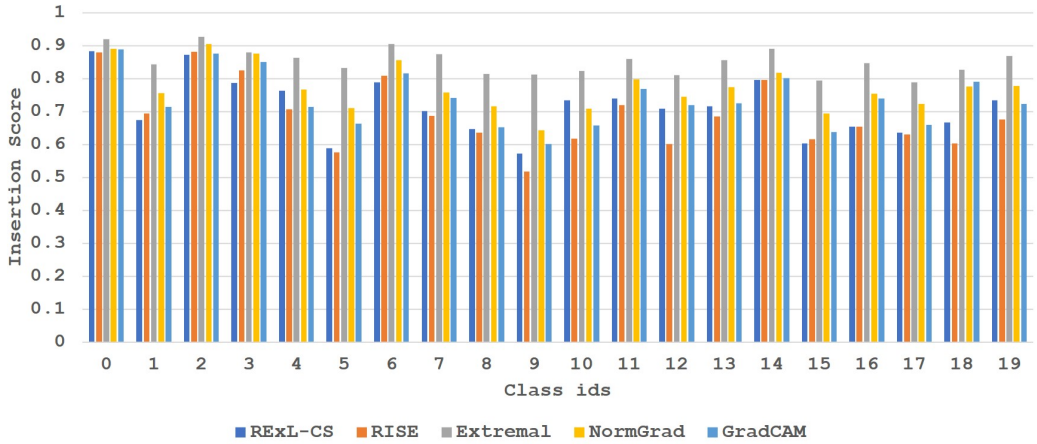

Figure 6: **Comparison of insertion scores (the higher the better) for ResNet**50 **on all classes of VOC**2007**:** RExL-CS is at par with all the baselines and on average better than RISE.

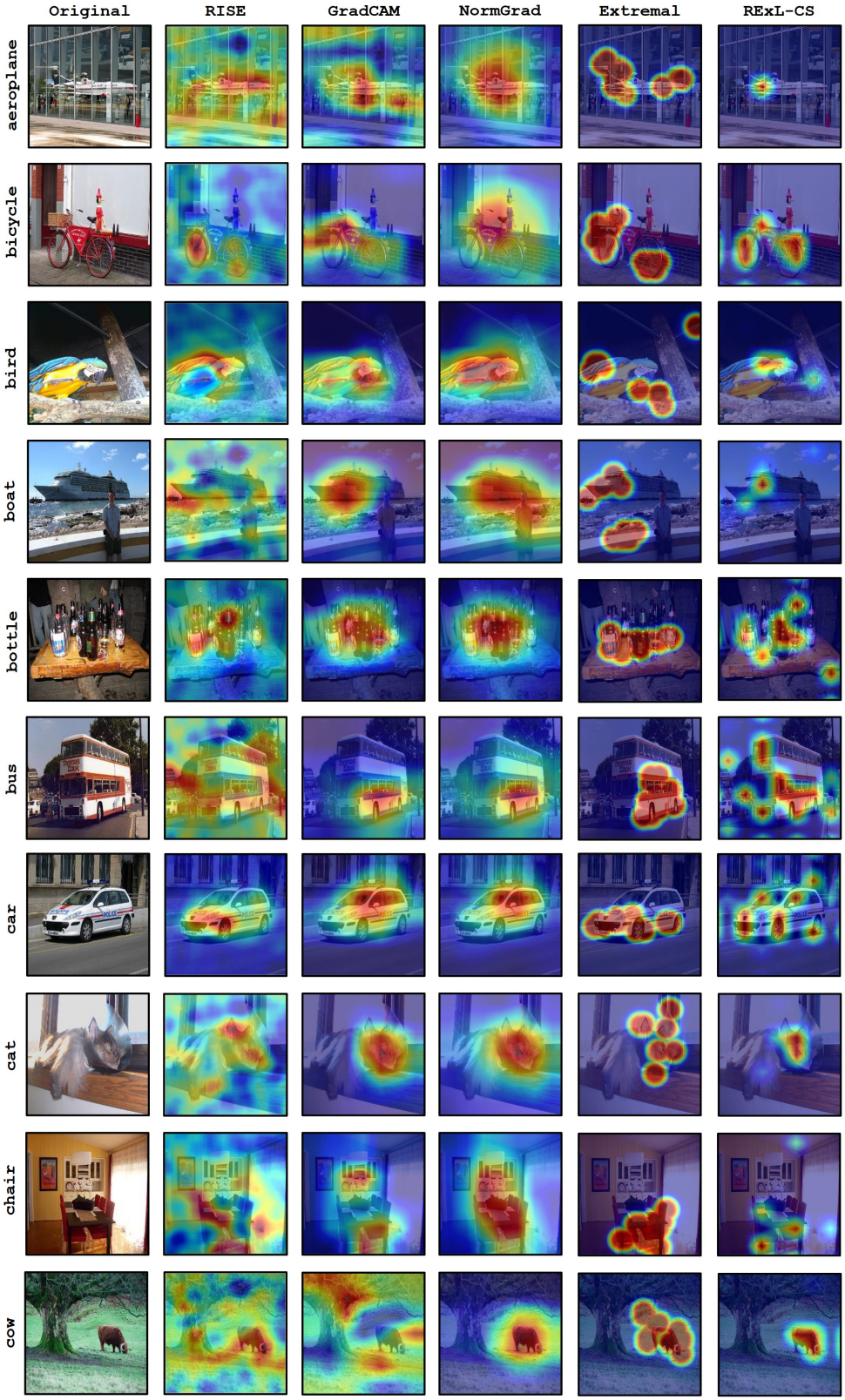

Figure 7: **Qualitative comparison of saliency maps for ResNet**50 **on classes** $0-9$ **of VOC**2007**:** As expected, the saliency map for "aeroplane" is better than the baseline methods. Moreover, for classes "bicycle", "bus", "chair" and "cow", the maps are much better than the four baselines. For the same input for class "boat", RExL performs way better while using ResNet50 than when it was using VGG16.

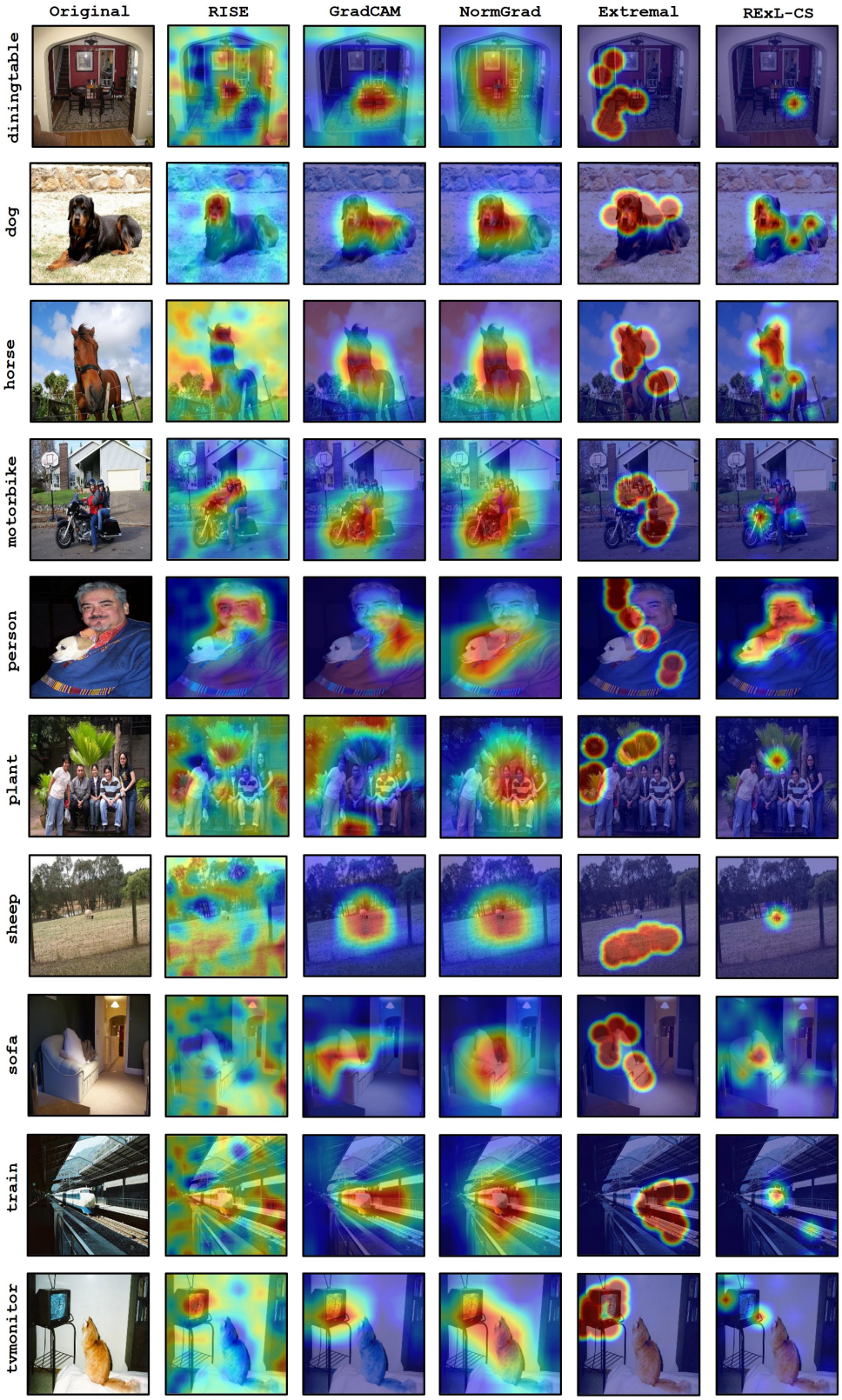

Figure 8: **Qualitative comparison of saliency maps for ResNet**50 **on classes** $10-19$ **of VOC**2007**:**Like the previous Figure, RExL-CS outperforms the baseline methods in most of the classes. It almost always generates better explanations than the two black box methods and is always at par (and sometimes better) with the white box methods.

4 RExL-CS on ImageNet

We also perform classwise comparison of the saliency maps generated by RExL-CS with the baseline methods on ImageNet. But instead of all the 1000 classes, we show results only on the top 10 classes (as per the classifcation accuracy). Similar to VOC2007, we use deletion and insertion scores to compare quantitatively while qualitatively we show the saliency maps produced for a random example from each of the 10 classes. Again we perform the experiments for both the base models.

4.1 VGG16

Figure (9) and (10) show the comparison of class wise deletion and insertion scores among the different methods respectively. And, Figure (11) qualitatively compare the saliency maps generated by RExL-CS with the baselines for every class. For VGG16, the scores are at par with the baselines, outperforming the baselines few times.

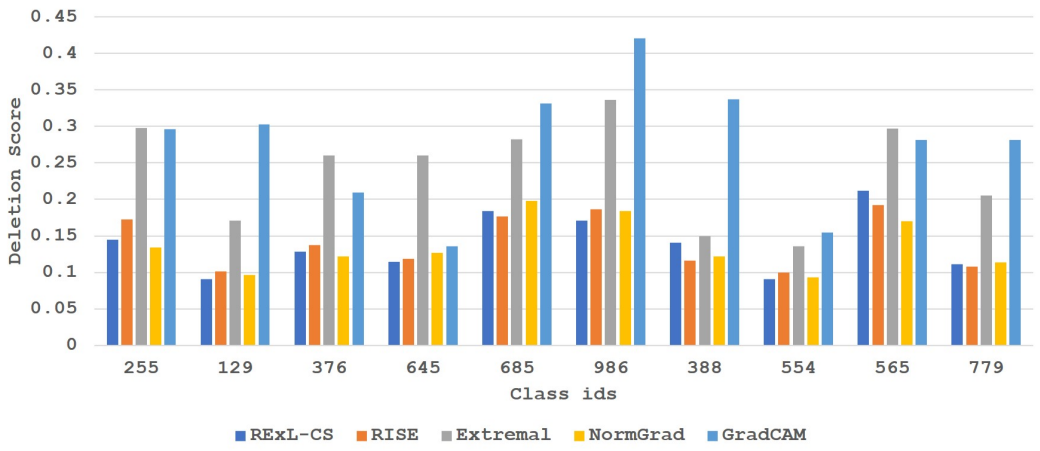

Figure 9: **Comparison of deletion scores (the lower the better) for VGG**16 **on top 10 classes of ImageNet:** RExL-CS gets competitive deletion scores for all the classes and outperforms in 4 classes.

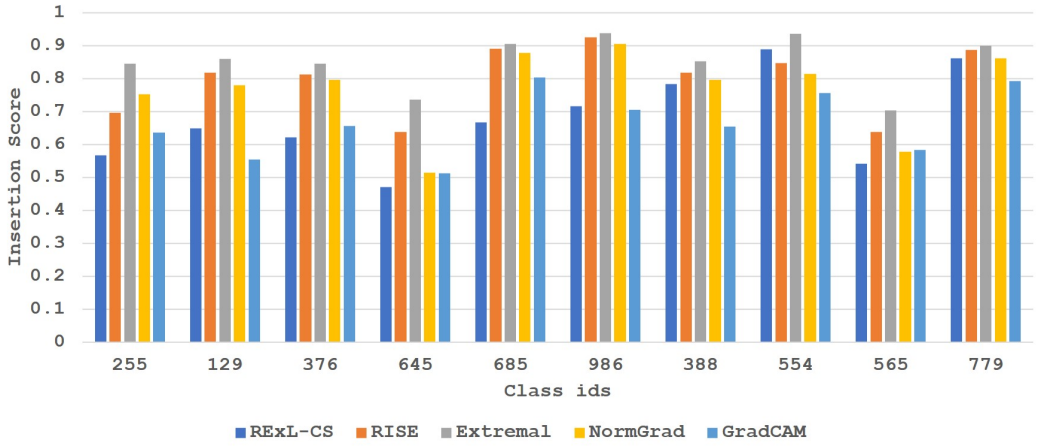

Figure 10: **Comparison of insertion scores (the higher the better) for VGG**16 **on top 10 classes of ImageNet:** RExL-CS gets competitive insertion scores for all the classes.

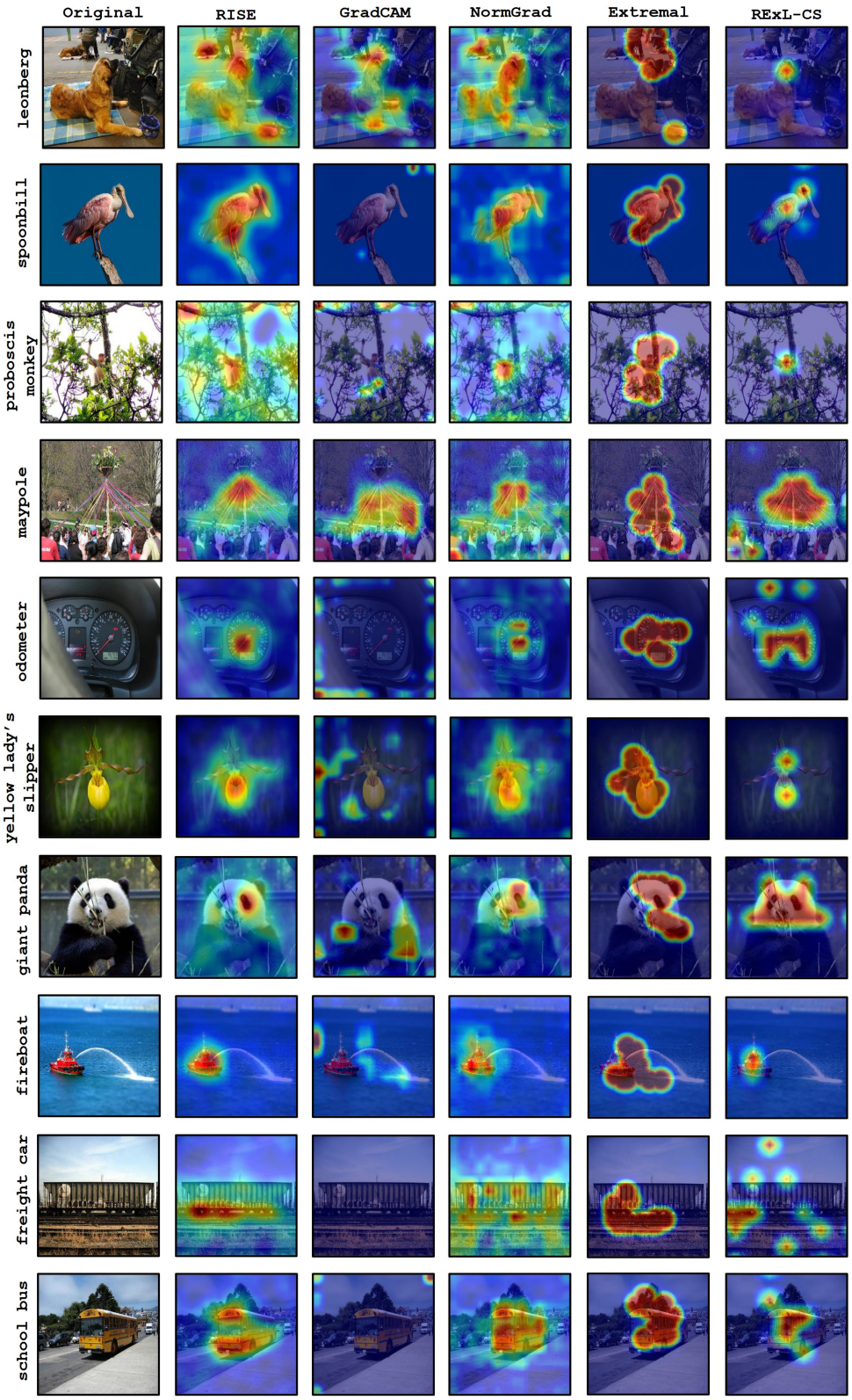

Figure 11: **Qualitative comparison of saliency maps for VGG16 on top 10 classes of ImageNet:** RExL-CS always generated better maps than Extremal. In most cases, RExL-CS outperforms RISE as well. GradCAM does not perform well on some classes which can also be reflected from its poor scores on these classes. In several classes like "giant panda", "leonberg" and "proboscis monkey", RExL-CS generates better maps than all the baselines including the white box methods.

4.2 ResNet50

We perform similar comparisons using ResNet50 as the base model. Figure (12) and (13) compare the class wise deletion and insertion scores among the different methods respectively. And, Figure (14) qualitatively compares the saliency maps generated by RExL-CS with the baselines for every class. As expected, RExL performs better using ResNet50 as the base model and this can be illustrated from both the scores and the saliency maps.

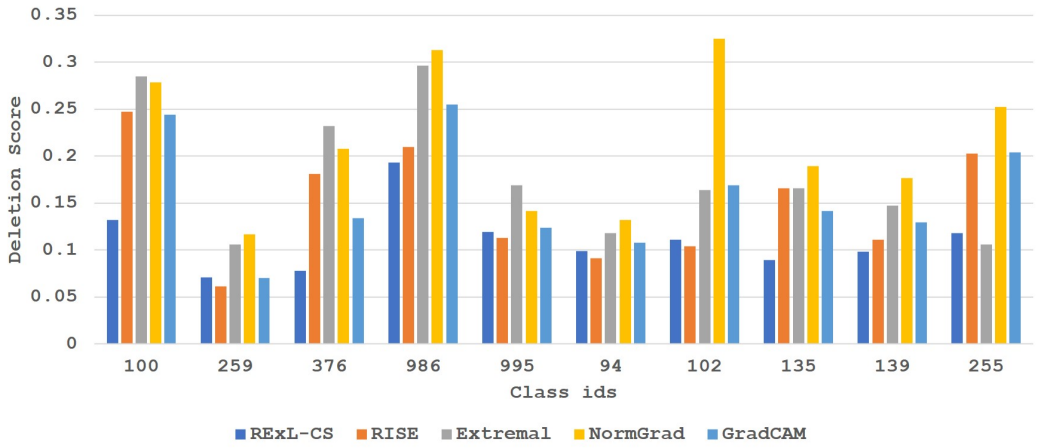

Figure 12: **Comparison of deletion scores (the lower the better) for ResNet50 on top 10 classes of ImageNet:** RExL-CS outperforms the baselines on 5 classes. Wherever it fails to be the best, it just falls behind by a very small margin and is almost always the second best.

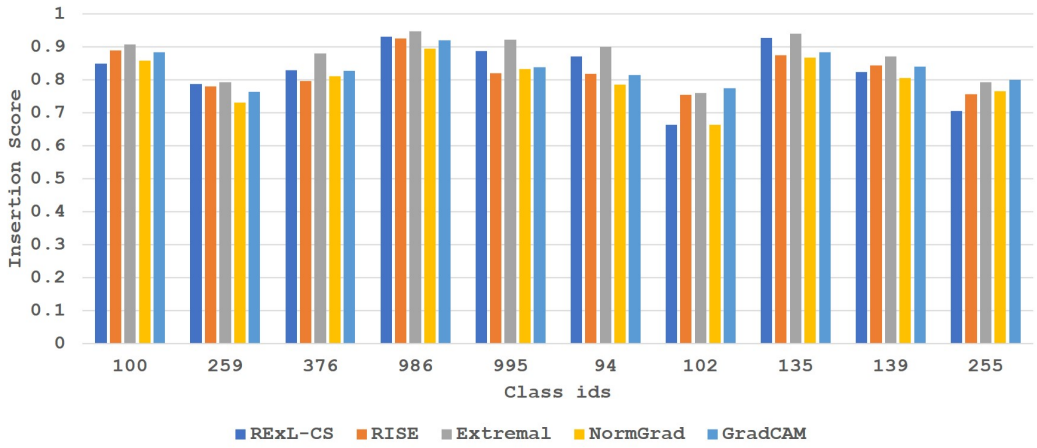

Figure 13: **Comparison of insertion scores (the higher the better) for ResNet50 on top 10 classes of ImageNet:** RExL-CS gets competitive insertion scores for all the classes. On average, it is better than three of the baselines.

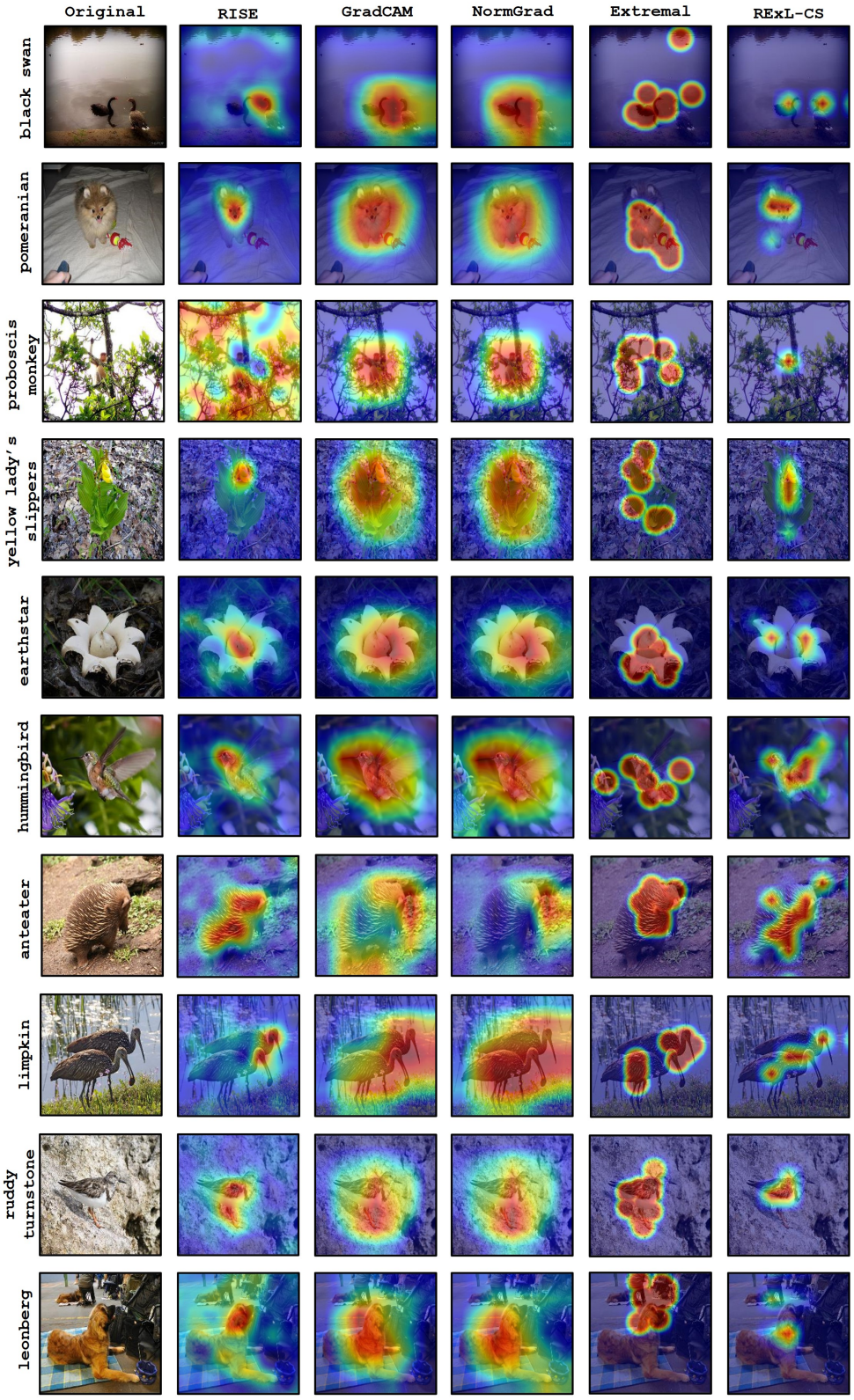

Figure 14: **Qualitative comparison of saliency maps for ResNet**50 **on top 10 classes of ImageNet:** RExL-CS again is always better than Extremal and at par (if not better) than RISE. It is also better than the two white box methods which give a large blob of importance rather than providing any intricate information. For classes "black swan", "proboscis monkey", "hummingbird", "leonberg" and "ruddy turnstone", RExL by far outperforms the baselines.

5 RExL-IS

These agents are trained on single images and explain the decisions made by the base model on that image only. These are hence not scalable for any practical datasets but if the base model needs to be explained very accurately and for a small number of images, RExL-IS will provide the best results. A comparison between the different baselines and RExL-IS can be seen in Figure (15). These images are taken from VOC2007 test split.

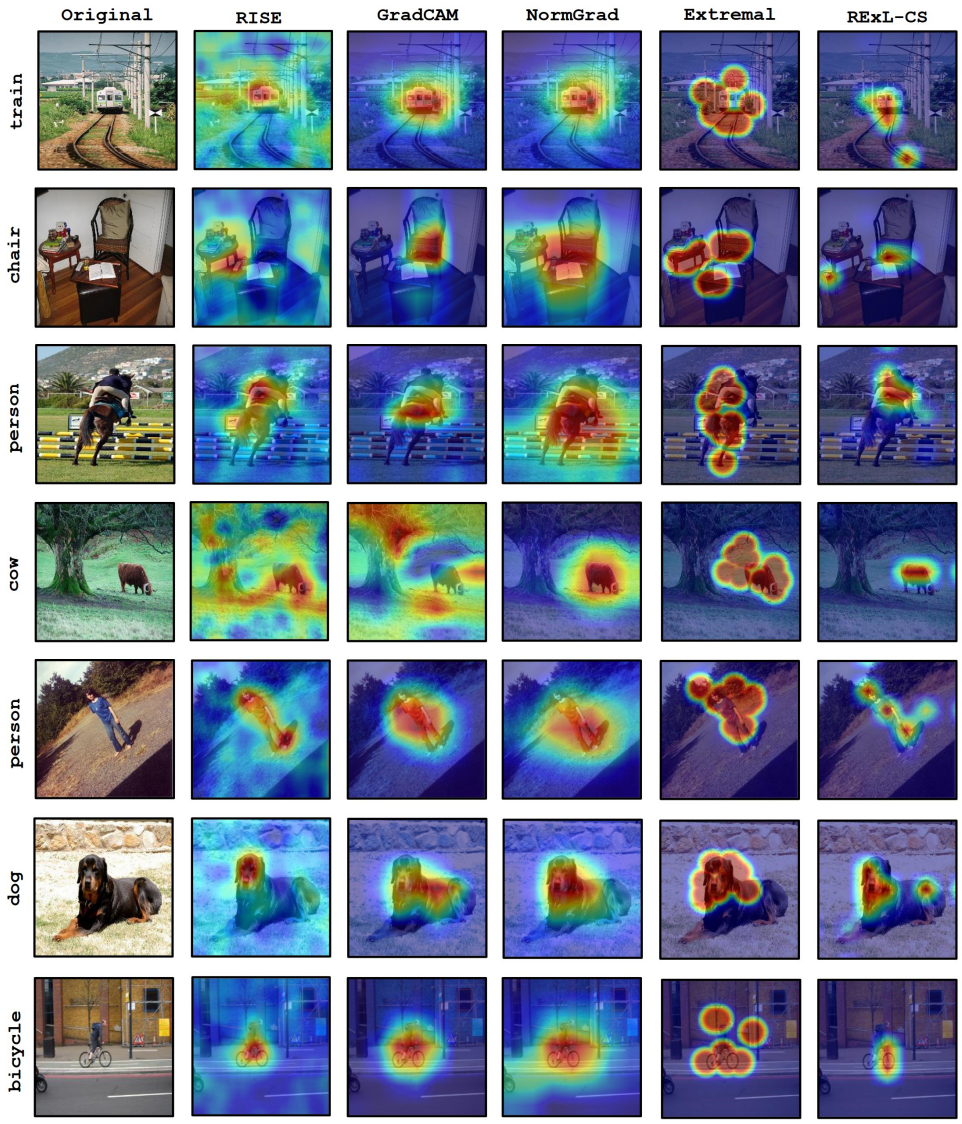

Figure 15: **Qualitative comparison of saliency maps for RExL-IS using ResNet**50 **base model on VOC**2007**:**For all the images, RExL-IS is significantly better than all the four baselines. While Extremal Perturbation provides almost equal importance to the entire object, white box methods (GradCAM and NormGrad) also simple add a large region of importance on the image instead of looking for the specific features. This is clearly highlighted in the "bicycle" image where GradCAM and NormGrad give large importance to the entire region containing the human as well.

References

[1] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[2] Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[3] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-box Models. In *British Machine Vision Conference*, Sep 2018.

[4] Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, Hakan Bilen, and Andrea Vedaldi. NormGrad: Finding the Pixels that Matter for Training. *CoRR*, abs/1910.08823, 2019.

[5] 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 *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[6] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *Int. J. Comput. Vision*, 126(10):1084–1102, October 2018.