Reinforcement-based Training Approach for Improving Adversarial Robustness without Sacrificing Accuracy

Dongsuk Kim

kds900910@gmail.com

Abstract

Supervised learning using cross-entropy loss has achieved excellent results in classification tasks. However, it has the disadvantage of overly depending on the labeled data such that adversarial samples can make the trained model to have mistakes. Our study shows that using reinforcement learning for classification tasks can achieves better robustness against adversarial attacks without sacrificing accuracy. A policy loss in reinforcement learning deforms the landscape of the overall loss, which leads to various advantageous searches in the policy search space. In addition, in order to use reinforcement learning for multiclass classification tasks, we propose a new training strategy in which a derived policy loss function is used to improve the accuracy and robustness of the model against adversarial attacks. Evaluations regarding flatness and adversarial defense on the MNIST and CIFAR-10 datasets give promising results, in which the model trained by a policy loss becomes robust to an adversarial attack.

Introduction

$\widehat{x_k}$ is normalized kth column vectors of im2col matrix of inputs. Supervised learning using neural networks has solved numerous classification problems in computer vision, such as image classification (Krizhevsky, Ilya, and Hinton 2012), object detection (Girshick et al. 2014) and image generation (Goodfellow et al. 2014). Those methods uses a cross-entropy loss function as the objective function to optimize network models. In general, these artificial neural networks trained via supervised learning are not well trained due to limitations of the training data, such that the generalization performance being deteriorated due to over-fitting issues (Zhang et al. 2021). Also, despite the fundamental breakthroughs in various tasks of computer vision, deep neural networks have been shown to be utterly vulnerable to adversarial attacks. Adding miniscule perturbations that are indetectable with the human eye to the adversarial sample weakens the predictive power of the model (Goodfellow et al. 2015). To overcome this serious phenomenon, many methods have been studied and proposed, while at the same time, new methods of attack against new defenses were also developed. Well-known defense methods include data augmenting (Rebuffi et al. 2021), adversarial training using adversarial examples, and developing a new regularizer (Madry et al. 2018; Zhang et al. 2019; Wang et al. 2019). We will show that it is inherently possible to overcome adversarial attacks with a simple policy loss, which is adopted from reinforcement learning methods.

In general, reinforcement learning has been applied to dynamic environments, such as computer games (Mnih et al. 2013) and robot control (Gu, Lillicrap, and Levine 2017) and has achieved good results. In recent years, deep reinforcement learning has been actively applied to classification problems. The general strategy is to maximize the expected reward by giving a penalty to the classifier whenever it misclassifies and can be as intuitive as using maximum likelihood estimation (MLE). However, only few studies have been conducted on classification problems using policy gradients.

In this paper, a method of training a classifier to maximize expected rewards using policy gradients is analyzed. In reinforcement learning, the classification task is regarded as a simple sequential decision process. The reward function is defined intuitively, as in, when the action according to the probability policy value correctly classifies a sample, a positive reward is given; otherwise, a negative reward, i.e. punishment (penalty). We introduce a new training strategy and an alternative loss to train a model using vanilla policy gradients without sacrificing the philosophy of policy gradients.

The goals of this paper are summarized in several major steps. First, we formulate the supervised problem as a sequential decision-making process. Next, we analyze the relation between the policy loss and the conventional cross-entropy loss. Then, we show that the loss landscape can be deformed by the reward function. We use this observation to introduce an alternative loss to train a deep CNN model using vanilla policy gradients. Finally, we study the performance of deep reinforcement learning through experiments and verify the consensus of the analyzed results. We also see that the trained model is robust to adversarial attacks, while achieving a high clean accuracy that exceeds models trained by the Cross-Entropy (CE) loss.

Related Works

Reinforcement Learning for the Classification Problem

Deep reinforcement learning has recently achieved excellent results in classification tasks as it helps classifiers to search a trajectory to learn advantageous features. In (Wiering et al. 2011), the classification task was constructed using multiple agents to interact with the environment to learn the optimal classification policy. In (Feng et al. 2018), Feng et al. proposed deep reinforcement learning to learn the relationship classification in noisy text data. The works in (Zhang, Huang, and Zhao 2018; Liu and Jiang 2018; Janisch, Pevny, and Lis'y 2017) utilized deep reinforcement learning to learn advantageous structured features in training data. (Martinez et al. 2018) proposed a deep reinforcement learning framework for time series data classification in which the definition of the Markov decision process is well defined. (Lin, Chen, and Qi 2020) studied the use of a deep Q-network for imbalanced binary classification. (Gupta 2020) preemptively applied the vanilla policy gradient to obtain results for adversarial defense on multi-classification. But there are no results on well-known deep convolutional neural networks like VGG (Simonyan and Zisserman 2014), and Resnet (He et al. 2016) and only focuses on a very simple model with lower accuracy. This paper proposes a training strategy to train a neural network using policy loss for multi-classification, achieving high accuracy.

Adversarial Attacks

There are two main categories among the various computational methods for generating adversarial examples: white-box adversarial, and black-box adversarial attacks. In the white-box setting, attackers have all the information about the targeted neural network, including the network structure and network parameters. On the other hand, in black-box cases, attackers don't have access to the parameters of the classifer. (Goodfellow et al. 2015) introduced a one-step attacking method, the Fast Gradient Sign Method (FGSM), which crafts an adversarial example with the perturbation and sign of the gradient of the training loss. The Projected Gradient (PGD) methods enhance FGSM by clipping an adversarial example within l_∞ and iteratively applying the gradient -based perturbation step by step, each with a small step size (Kurakin, Goodfellow, and Bengio 2016). There are many other methods, such as the Auto-Attack method, that are parameter-free, computationally affordable, and user-independent ensemble of attacks (Croce and Hein 2020) with strong attack power.

Defense for Adversarial Attacks

To defend against an adversarial attack, the commonly used defending model is advesarial training model such as standard adversarial training (AT) (Goodfellow et al. 2015), Theoretically Principled Trade-off between Robustness and Accuracy (TRADES) (Zhang et al. 2019), Robust Self-Training(RST) (Carmon et al. 2019). They are all based on CE loss by adding adittional regularizers. Although the AT methods are robust, they have trade-off between clean accuracy on inputs and adversarial robustness. These current robust state-of-the-art methods are based on cross entropy loss. Also, there other approches using a new loss function; (Pang et al. 2019) proposed a Max-Mahalanobis center loss to learn discriminative features. (Chen et al. 2019) use Guided Complement Entropy to balance between the cross-entropy and a guided term neutralizing the attacks on a trained network by looking for incorrect probabilities, and (Li et al. 2020) use Probabilistically Compact Loss with Logit Constraints that enlarge the ability gaps between true classes and false classes. Meanwhile, the logit constraints prevent the gaps from being close by a small perturbation. Although our method is not a theoretically developed method, it is simple and can train a neural network while intrinsically avoiding adversarial attacks and achieving high accuracy.

Proposed Method

Formulation

Our study is fashioned from reinforcement learning where the agent makes a decision from learning labeled instances that are sampled from a dataset. The framework is formalized as follows.

State The state $\mathbf{s}$ is randomly drawn from an state space $\mathcal{S}$ that is equivalent to a dataset of n samples $(x_i, y_i)_i^n$, where $x_i \in R^d$ is d dimensional feature vector and $y_i = k$ ($k = 1, ..., C$) is the true class label in a state. In the case of unlabeled dataset, y_i is missing.

Policy An actor (a.k.a. policy, agent, classifier) predicts an action from a given datum and a neural network. The action is drawn from the label space according to the predicted probability for sample x of the given datum. With assumption that the output layer is a fully connected layer of C neurons, the probability of predicting k is calculated using the k-th logit $q_k = \mathbf{W_k} f_x + b_k$ and the softmax activation:

$$\pi_k(x) = \frac{\exp(q_k)}{\sum_{i=1}^{C} \exp(q_i)} \quad (i = 1, .., C) \tag{1}$$

where f_x is the feature representation of x through layers of a policy, $\mathbf{W_k}$ and b_k are parameters of the k-th neuron in the output layer. We define $\pi(x)$ as $(\pi_1(x), ...\pi_C(x))$

Action The action a is drawn from an action space $\mathcal{A}$, which is also identical to the label space of the dataset. We use y to denote the true label of state $\mathbf{s}$.

Transition Function Denoted by $P(\mathbf{s_{t+1}}|\mathbf{s_t}, a_t)$ is stochastic, which means that the actor randomly moves from the current state $\mathbf{s_t}$ to the next state $\mathbf{s_{t+1}}$ according to a random sampling.

Reward The reward r_t of t-th state is the feedback to the actor of whether its classification is correct or not. In the case case of a labeled dataset, the reward is given from a reward function, $r(s, a)$, which can be customized for specified purposed. For example, in the case of unlabeled data, the reward can be given from a generative adversarial reward model. In classification tasks, this function depends only on the current sample and action.

Episode An episode is a batch of tuples $< \mathbf{s_t}, a_t, r_t >$ from the initial decision step $t = 0$ to the terminal decision step $t = N - 1$ for a fixed number N. In fact, we consider the whole batch size as an episode. This is not generated by sampling actions but a random sampling of data that already exists.

Discounted factor $\gamma \in [0, 1]$ is used as a factor that quantitatively estimates the effect of previous decisions on the current decision. In multi-class image classification task, as the current action does not affect to the next state, we do not consider this factor. We set the value of the factor to 1.

With the definitions and notations above, our problem is formally defined as finding an agent that can work under a training data and can optimize to not only classify well but also to maximize the total expected reward.

$$R = \mathbf{E}_{a\sim\pi(\mathbf{s})}[\sum_{t=0}^{N-1} r_t]. \tag{2}$$

An Analysis of Policy Loss

Let the reward function be

$$r(\mathbf{s}, a) = \begin{cases} r_{k_1} & a = y = k \\ -r_{k_2} & a \neq y = k, \end{cases} \tag{3}$$

where C is the number of classes, r_{k_i} $(k = 1, .., C, i = 1, 2)$ are all non-negative real numbers, and y is the label of $\mathbf{s}$.

Let a policy loss function be defined as follows (Sutton et al. 2000):

$$\mathcal{L}_\pi = -\mathbf{E}_{a\sim\pi(\mathbf{s})}\Big[\sum_{\mathbf{s}} r(\mathbf{s}, a)\mathbf{a} \cdot \log \pi(x)\Big], \tag{4}$$

where $\mathbf{a}$ is a one-hot vector of action a. Note that consequently, the policy gradient can be formalized as follows:

$$\begin{aligned} \nabla\mathcal{L}_\pi &= -\mathbf{E}_{a\sim\pi(\mathbf{s})}\Big[\sum_{\mathbf{s}} r(\mathbf{s}, a)\mathbf{a} \cdot \nabla \log \pi(x)\Big] \\ &= -\mathbf{E}_{a\sim\pi(\mathbf{s})}\Big[\sum_{\mathbf{s}}(r_{k_1}\delta(a, y) - r_{k_2}(1 - \delta(a, y)) \\ &\quad \mathbf{a} \cdot \nabla \log \pi(x)\Big] \end{aligned} \tag{5}$$

where

$$\delta(a, y) = \begin{cases} 1 & a = y \\ 0 & a \neq y. \end{cases} \tag{6}$$

In the case $r_{k_2} = 0$ for all k, the policy gradient $\nabla\mathcal{L}_\pi$ is simplified to:

$$\nabla\mathcal{L}^r_\pi := -\mathbf{E}_{a\sim\pi(\mathbf{s})}\Big[\sum_{\mathbf{s}} r_{k_1}\delta(a, y)\mathbf{a} \cdot \nabla \log \pi(x)\Big]. \tag{7}$$

Meanwhile, when $r_{k_1} = 0$, the policy gradient $\nabla\mathcal{L}_\pi$ is as follows,

$$\nabla\mathcal{L}^p_\pi = -\mathbf{E}_{a\sim\pi(\mathbf{s})}\Big[\sum_{\mathbf{s}} -r_{k_2}(1 - \delta(a, y))\mathbf{a} \cdot \nabla \log \pi(x)\Big] \tag{8}$$

By the linearity of the expectation value, a general policy gradient can be decomposed as:

$$\nabla\mathcal{L}_\pi = \nabla\mathcal{L}^r_\pi + \nabla\mathcal{L}^p_\pi. \tag{9}$$

As a result, the policy gradient is analyzed based on $\nabla\mathcal{L}^r_\pi$ and $\nabla\mathcal{L}^p_\pi$.

- $\nabla\mathcal{L}^r_\pi$: With $r_{k_1} = 1$, $\nabla\mathcal{L}^r_\pi$ is reduced to:

  $$\nabla\mathcal{L}^r_\pi : = -\mathbf{E}_{a\sim\pi(\mathbf{s})}\Big[\sum_{\mathbf{s}} \delta(a, y)\mathbf{a} \cdot \nabla \log \pi(x)\Big]. \tag{10}$$

  When an action is equal to the label value, the policy gradient $\nabla\mathcal{L}^r_\pi$ is identical to the gradient of the canonical cross-entropy loss. This means that a policy gradient $\nabla\mathcal{L}^r_\pi$ updates parameters of the policy network along the direction of the gradient of a cross-entropy loss.

- $\nabla\mathcal{L}^p_\pi$: With $r_{k_2} = 1$, the policy gradient $\nabla\mathcal{L}^p_\pi$ converges to a negative cross-entropy that is implemented as follows,

  $$-\sum_{\mathbf{s}}[-(1 - \delta(a, y))\mathbf{a} \cdot \nabla \log \pi(x)] \tag{11}$$

  Intuitively, if an action is not equal to the label value, a policy gradient $\nabla\mathcal{L}^p_\pi$ updates parameters of the policy network along the inverse direction of the gradient of cross-entropy loss.

In summary, depending on the values of r_{k_i} and (a, y), the policy gradient will update the policy's parameters either following a CE loss or following the inverse direction of the CE loss. In other words, the policy gradient updates parameters in the direction of either increasing the accuracy (cross-entropy) or avoiding an incorrect action; this means that the policy loss focuses on the action rather than the target. Hence, contrary to supervised learning, which follows a gradient to fit some training data, a policy loss can learn along more diverse gradient paths, avoiding simply fitting to targets.

However, in practice, like there are two sides of a coin, network training is not effective with this loss function, because in contrast to a CE loss, which reflects the information of the target to current probability, a policy loss only reflects the information of the current action probability. This implies that if the action's distribution is not good at the initial training step, learning can fail.

To succeed in training using a policy gradient, we introduce a two-step training. As log sum of prabibility of all classes is lower bounded, increasing the sum of the log of the probability of remaining classes reduces the log of a probability of the action that is not equal to the target. Therefore, we use a two-step training strategy. First, we warm up the training by the loss

$$\begin{aligned} \mathcal{L}_\pi = -\mathbf{E}_{a\sim\pi(\mathbf{s})}\Big[&\sum_{\mathbf{s}} r_{k_1}\delta(a, y)\mathbf{a} \cdot \log \pi(x) \\ &+ \frac{r_{k_2}}{C-1}(1 - \delta(a, y))\big((\sum_i^C \log \pi_i(x)) \\ &- \mathbf{a} \cdot \log \pi_a(x)\big)\Big] \end{aligned} \tag{12}$$

Note that minimizing $\mathcal{L}^p_\pi$ is intuitively equal to minimizing

$$-\big((\sum_i^C \log \pi_i(x)) - \mathbf{a} \cdot \log \pi_a(x)\big). \tag{13}$$

The loss gives a uniform distribution of the remaining classes except for the class corresponding to the incorrect action. By this substitution of terms, we can fully use the information of the action and target at the initial steps.

Next, if the action is well-distributed, we first train using the original policy loss Eq (4). To verify the effects of this strategy, we performed an ablation study. Since the main difference in network behavior with CE loss from $\mathcal{L}^{p}_{\pi}$, we set $\lambda = \left|\frac{r_{k_2}}{r_{k_1}}\right|$ and then study the policy loss by manipulating this value.

Comparing the Smoothness of the Policy Loss and the Cross Entropy Loss

Training neural networks requires minimizing a high-dimensional non-convex loss function. The trainability of neural networks highly depends on the network architecture design choices, the choice of optimizer, variable initialization, and a variety of other considerations. Several works have addressed the relationship between sharpness and flatness of local minima and their generalization ability. (Hochreiter and Schmidhuber 1997) defines "flatness" as the size of the connected region around the minimum where the training loss remains low. (Keskar et al. 2017) characterizes flatness using eigenvalues of the Hessian, and proposes sharpness as an approximation, which looks at the maximum loss in a neighborhood of a minimum. We shall show that the smoothness of the policy loss is better than that of the CE loss.

First, let us recall two definitions.

L**-Lipschitz** a function f is L-Lipschitz if

$$|f(x_1) - f(x_2)| \leq L\|x_1 - x_2\| \tag{14}$$

for all x_1, x_2.

β**-smoothness** a function f is β-smooth if its gradient is β-Lipschitz.

Recall that

$$\begin{aligned}\pi(x) &= (\pi_1(x), ..., \pi_C(x))\\ &= S(\mathbf{q})\end{aligned} \tag{15}$$

where S and $\mathbf{q}$ are the softmax activation and the logits vector.

Suppose that $\pi(x)$ be a twice differentiable function. Let the policy loss be

$$\begin{aligned}\mathcal{L}_\pi = -\,\mathbf{E}_{a\sim\pi(\mathbf{s})}\Big[\sum_{\mathbf{s}}&(r_{k_1}\delta(a,y) - r_{k_2}(1-\delta(a,y))\\ &\mathbf{a}\cdot\nabla\log\pi(x))\Big]\end{aligned} \tag{16}$$

and the classical categorical CE loss be

$$\mathcal{L}_c = -\sum_{\mathbf{s}}\mathbf{y}\cdot\log\pi(x) \tag{17}$$

where $\mathbf{y}$ is the one hot vector of label y. Let $\mathcal{H}_\pi$ be the Hessian of the policy loss and $\mathcal{H}_c$ be the Hessian of the CE loss.

First, note that

$$\begin{aligned}\frac{\partial\log S}{\partial q_i} = \frac{\partial\log\pi(x)}{\partial q_i}\\ = \pi_i - y_i\end{aligned} \tag{18}$$

where y_i is the i-th element of $\mathbf{y}$.
Then, for a parameter variable u,

$$\nabla_u\mathcal{L}_c = \sum_{\mathbf{s}}\big[\sum_{k=1}^{C}(\frac{\partial q_k}{\partial u})(\pi_k(x) - y_k)\big]. \tag{19}$$

then the policy gradient is equal to

$$\begin{aligned}\nabla_u\mathcal{L}_\pi = \mathbf{E}_{a\sim\pi(\mathbf{s})}&\Big[\sum_{\mathbf{s}}\big[\sum_{k=1}^{C}(r_{k_1}\delta(a,y)\\ &- r_{k_2}(1-\delta(a,y))(\frac{\partial q_k}{\partial u})(\pi_k(x) - y_k)\big]\Big]\\ &= \sum_{\mathbf{s}}\mathbf{E}_{a\sim\pi(\mathbf{s})}\big[\sum_{k=1}^{C}(r_{k_1}\delta(a,y)\\ &- r_{k_2}(1-\delta(a,y))(\frac{\partial q_k}{\partial u})(\pi_k(x) - y_k)\big]\\ &= \sum_{\mathbf{s}}\mathbf{a}\cdot\pi(x)\big[\sum_{k=1}^{C}(r_{k_1}\delta(a,y)\\ &- r_{k_2}(1-\delta(a,y))(\frac{\partial q_k}{\partial u})(\pi_k(x) - y_k)\big]\end{aligned} \tag{20}$$

Next, the hessian element of the CE loss with respect to u, v is

$$\begin{aligned}\mathcal{H}_{c,u,v} = \sum_{\mathbf{s}}&\Big[(\frac{\partial\mathbf{q}}{\partial u})^T[diag(\pi(x)) - \pi(x)\pi(x)^T]\frac{\partial\mathbf{q}}{\partial v}\\ &+\sum_{k=1}^{C}\left(\frac{\partial^2 q_k}{\partial u\,\partial v}\right)(\pi_k(x) - y_k)\Big].\end{aligned} \tag{21}$$

where $diag(\pi(x))$ is a $C\times C$ matrix $\left(\pi_i(x)\delta_{i,j}\right)_{i,j}$.

Similarly, a component of the Hessian matrix of the policy loss is

$$\begin{aligned}\mathcal{H}_{\pi,u,v} = \sum_{\mathbf{s}}&\Big[\mathbf{a}\cdot\pi(x)((r_{k_1}\delta(a,y) - r_{k_2}(1-\delta(a,y))\\ &(\frac{\partial\mathbf{q}}{\partial u})^T[(diag(\pi(x)) - \pi(x)\pi(x)^T)](\frac{\partial\mathbf{q}}{\partial v})\\ &+((r_{k_1}\delta(a,y) - r_{k_2}(1-\delta(a,y))\\ &(\frac{\partial\mathbf{q}}{\partial u})^T[(\pi(x) - \mathbf{y})(\frac{\partial\mathbf{a}\cdot\pi(x)}{\partial\mathbf{q}})^T](\frac{\partial\mathbf{q}}{\partial v})\\ &+\sum_{k=1}^{C}(r_{k_1}\delta(a,y) - r_{k_2}(1-\delta(a,y))\\ &\left(\frac{\partial^2 q_k}{\partial u\,\partial v}\right)\mathbf{a}\cdot\pi(x)(\pi_k(x) - y_k)\Big].\end{aligned} \tag{22}$$

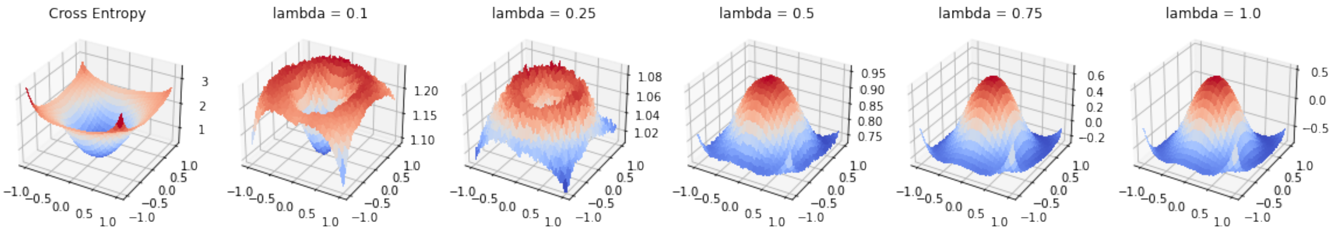


Figure 1: 3D visualization figure for each loss function around a local minima of policy loss. It can be seen that the more dominant the punishment, the more concave.

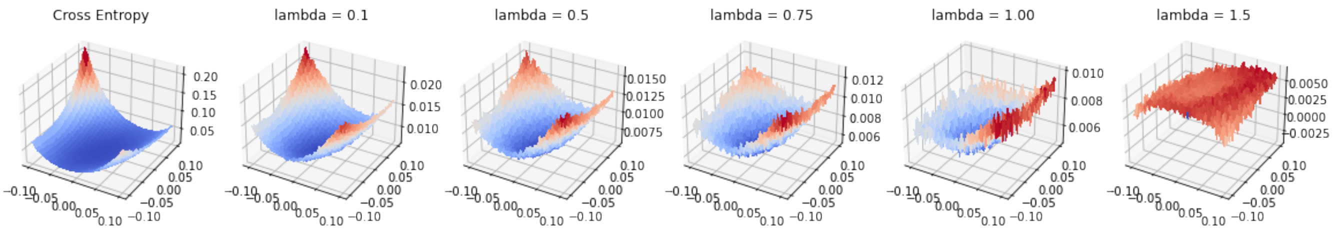


Figure 2: 3D visualization figure for each loss function around a local minima of CE loss. It can be seen that the more dominant the punishment, the minima is no longer a optima of a policy loss.

Let's denote A by

$$(\frac{\partial \mathbf{q}}{\partial u})^T[diag(\pi(x)) - \pi(x)\pi(x)^T](\frac{\partial \mathbf{q}}{\partial v}) \\ = \sum_{i,j}^{C}(\frac{\partial q_i}{\partial u})[\pi_i(x)\delta_{i,j} - \pi_i(x)\pi_j(x)](\frac{\partial q_j}{\partial v}) \tag{23}$$

and similarly $\tilde{A}$ by

$$\mathbf{a}\cdot\pi(x)(\frac{\partial \mathbf{q}}{\partial u})^T[diag(\pi(x)) - \pi(x)\pi(x)^T]\frac{\partial \mathbf{q}}{\partial v} \\ + (\frac{\partial \mathbf{q}}{\partial u})^T[(\pi(x) - \mathbf{y})(\frac{\partial \mathbf{a}\cdot\pi(x)}{\partial \mathbf{q}})^T](\frac{\partial \mathbf{q}}{\partial v}) \\ = \sum_{i,j}^{C}(\frac{\partial q_i}{\partial u})[\pi_i(x)\delta_{i,j} - \pi_i(x)\pi_j(x) \\ + (\pi_i(x) - y_i)\pi_{\mathbf{a},j}(x)](\frac{\partial q_j}{\partial v}) \tag{24}$$

where

$$\pi_{\mathbf{a},j} = \begin{cases} \pi_j(x)(1-\pi_j(x)) & a_j = 1 \\ -\pi_a(x)\pi_j(x) & a_j = 0. \end{cases} \tag{25}$$

where a_j means the j-th elements of the one-hot vector $\mathbf{a}$.

Finally, let's B be

$$\sum_{k=1}^{C}\left(\frac{\partial^2 q_k}{\partial u\,\partial v}\right)(\pi_k(x) - y_k). \tag{26}$$

Then

$$\mathcal{H}_{c,u,v} = \sum_{\mathbf{s}} A + B. \tag{27}$$

and

$$\mathcal{H}_{\pi,u,v} = \sum_{\mathbf{s}}(r_{k_1}\delta(a,y) - r_{k_2}(1-\delta(a,y))\tilde{A} \\ + (\mathbf{a}\cdot\pi(x))(r_{k_1}\delta(a,y) - r_{k_2}(1-\delta(a,y))B. \tag{28}$$

The only difference terms of A and $\tilde{A}$ are $(\pi_i(x) - y_i)\pi_{\mathbf{a},j}(x)$ $(i,j = 1,..,C)$ which consists of small positive value except negative values $(\pi_y(x)-1)\pi_{\mathbf{a},j}(x)$ contrary to $\left(diag(\pi(x)) - \pi(x)\pi(x)^T\right)$ consisted of small negative value except diagonal elements. Hence A and $\tilde{A}$ could have different signs.

Also if we suppose $r_{k_i} \leq 1$, there may be a situation where the sign of $A, \tilde{A}$ are different depending on the action. Although A and $\tilde{A}$ is almost same, depending on an action, the possibility of an appearance of a term having a different sign of $A + B$ could be increased.

In summary, because of these two possible cases, the probability that the norm will be reduced increases. Since the effect of reducing hessian norm helps optimization, the general reward function can help a gradient-based trajectory search.

Assume the CE loss tends to convex. In the region where the penalty r_{k_2} dominates the policy loss, that is, the point where the errors of actions are high, the loss landscape tends to concave, which means that r_{k_1} is related to convexity. This observation shows that a trajectory could move towards to depending on the terrain in various directions.

Attacks	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD
λ_2	0.1			0.25			0.5			0.75			1.0			1.25			1.5		
$\lambda_1 = 0.1$	90.00	53.942	37.46	90.54	51.48	31.68	91.47	51.15	23.38	91.88	55.18	28.11	92.18	60.07	36.58	**92.29**	**62.01**	**46.50**	92.16	58.38	43.84
$\lambda_1 = 0.25$	90.03	55.77	41.30	90.82	53.81	35.28	91.44	51.95	25.09	91.95	55.06	27.83	92.19	61.58	42.39	92.29	**61.83**	**46.14**	**92.41**	53.65	41.03
$\lambda_1 = 0.5$	89.96	55.18	41.51	90.89	55.59	37.15	91.56	52.46	24.44	92.05	58.68	33.39	92.08	62.37	42.16	**92.34**	**61.78**	**46.08**	92.23	58.05	44.57
$\lambda_1 = 0.75$	89.68	55.14	41.94	90.62	57.41	40.3	91.37	55.96	31.01	92.03	54.29	33.32	92.08	64.64	47.79	**92.16**	**64.87**	**49.1**	92.12	57.05	41.19
$\lambda_1 = 1.0$	89.35	49.88	34.15	90.15	50.79	31.95	91.04	50.75	24.11	91.64	54.61	27.10	91.81	59.73	41.51	**91.99**	**61.12**	**45.05**	91.93	55.74	41.02
$\lambda_1 = 1.25$	88.93	28.31	8.97	89.72	35.15	15.08	90.41	38.93	9.17	91.46	44.70	20.16	91.81	**46.16**	21.14	91.64	42.26	25.20	**91.93**	41.10	**28.43**
$\lambda_1 = 1.5$	88.27	25.53	6.40	89.21	22.57	4.78	89.82	22.72	1.99	91.03	**43.1**	11.39	91.52	35.51	14.95	**91.71**	42.58	**20.07**	91.68	30.31	17.19

Table 1: The test accuracy on CIFAR-10 against a set of adversarial attacks with $\epsilon = 0.04$ PGD with 40 iteration and step size 0.01. For each row, the best results are bold (averaged over 5 runs).

Attacks	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD
λ_2	0.1			0.25			0.5			0.75			1.0			1.25			1.5		
$\lambda_1 = 0.1$	99.07	87.42	**64.30**	99.1	85.28	61.14	99.06	85.6	60.13	99.15	86.69	59.31	**99.17**	**87.15**	60.85	99.06	85.69	62.16	**99.17**	85.35	56.82
$\lambda_1 = 0.25$	99.12	**86.35**	**64.96**	99.11	86.23	63.78	99.09	83.66	57.17	99.13	84.95	59.17	99.12	86.17	64.86	**99.18**	85.30	64.73	99.15	84.68	61.11
$\lambda_1 = 0.5$	99.10	86.25	62.93	99.11	**86.96**	**67.34**	99.12	83.59	59.54	99.14	85.35	63.06	99.17	83.94	58.00	**99.19**	85.62	64.47	99.17	82.68	56.18
$\lambda_1 = 0.75$	99.02	**87.86**	**69.06**	99.15	82.83	57.37	99.05	85.67	60.1	99.11	84.32	57.45	99.13	83.02	58.05	99.16	84.11	56.6	**99.21**	86.40	63.73
$\lambda_1 = 1.0$	99.19	85.90	59.5	99.10	**87.2**	**64.15**	99.02	85.9	60.52	99.16	83.37	58.39	99.19	85.55	61.13	99.16	84.82	59.02	**99.22**	85.81	62.05
$\lambda_1 = 1.25$	99.14	85.6	**63.83**	99.14	**86.01**	62.46	99.16	84.7	60.90	**99.20**	84.32	56.37	99.12	85.94	62.83	99.14	85.37	61.06	99.14	81.78	59.41
$\lambda_1 = 1.5$	99.12	**87.44**	**64.67**	99.10	85.36	60.47	99.17	86.32	59.94	99.19	85.96	62.92	99.19	85.64	59.47	99.3	86.07	59.67	**99.20**	86.07	60.20

Table 2: The test accuracy on MNIST against a set of adversarial attacks with $\epsilon = 0.1$ PGD with 40 iterations and a step size of 0.01. For each row, the best results are bold (averaged over 3 runs).

Relation between a Policy Loss and the Fast Gradient Sign Method

Adversarial attacks exploit the vulnerability that by adding a small perturbation to the inputs, in which the output can fool neural networks to produce incorrect logits with high probabilities. Adversarial attacks can be divided into two categories of black-box and white-box attacks based on the level of information available to the attacker. Fast Gradient Sign Methods(FGSM) are the simplest, but efficient white-box attacks. The adversarial attack generated by the FGSM attack can be described as:

$$x' = x + \epsilon sign(\nabla_x \mathcal{L}), \tag{29}$$

where ϵ is the attack strength and sign is the sign of the tensor gradient vector. During training, if the policy network misclassifies a sample, a policy does not update parameters along the direction of the cross-entropy but updates parameters along the opposite direction of the wrong action to correctly classify the sample. This allows for when the policy network classifies the sample, the CE loss of the sample can be highwhich is why the network trained by a policy loss is robust to an adversarial attack. Under the proposed scenario, a small perturbation is much less likely to create a dramatic change in the network behavior and outputs.

Experiments

In this section, we present the experiments that demonstrate the efficacy of our model and verify our theoretical analysis.

Datasets and Model

We perform experiments on MNIST and CIFAR-10 datasets. In our experiments, we trained LeNet5 (MNIST), VGG13 (CIFAR-10) on two main losses : CE loss, the alternative policy loss with origin policy loss. For all the models, we use the SGD optimizer without any additional constraints. For the CE loss, we train 100 epochs and for the policy loss, we train 200 epochs. We set the initial learning rate to 10^{-1} and divide by 10 for each 50 and 150 epochs.

When we trained a policy network, a two-step strategy was used to train a policy network. First, we warm up the network using the alternative loss Eq (12). Actions are chosen randomly according to probability. Then, if the test accuracy is higher than 80, we start to train using the originally defined policy loss Eq (4) without randomness. Experiments were performed while changing the values of $\lambda = (\lambda_1, \lambda_2)$. Each value of λ_i $(i = 1, 2)$ is in $\{0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5\}$. The whole results are in Table 1-2.

To demonstrate the need for this strategy, we include the results of the ablation study: *(1)* Warm up using CE loss. *(2)* Warm up using the alternative loss Eq (12). *(3)* Trained by only Eq (4). In the ablation study, methods using only Eq (4) do not train a network at all. Next, we can verify that the alternative loss work in synergy with the origin policy loss Eq (4). A pre-trained model with CE also shows that the robustness increased, but was relatively weak (See Table 4).

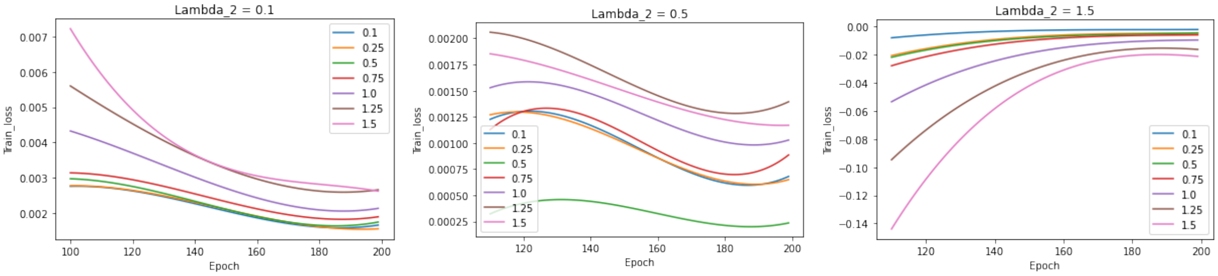
Loss Landscape of A Policy Loss

In the case of the visual confirmation of the loss landscape, we used a CIFAR-10 dataset. For visualizing, the two pre-trained VGG13 model with the two main losses were used and visualized with 2D interpolation on $[-1, 1]^2$, with each axis dividing into 50 equal parts. This follows the methods of (Li et al. 2017), as follows

$$f(\alpha, \beta) = \mathcal{L}(\theta^* + \alpha\omega_1 + \beta\omega_2), \tag{30}$$

where α,β are the reduced dimensions. θ^* are the optimal parameters, and ω_1 and ω_2 are standard Gaussian random vectors.

We use grid search to verify that the loss landscape is deforming appropriately. We visualized loss landscapes on two local optima of CE loss and policy loss. For the policy



(a) Training loss curves for $\lambda_2 = 0.1$.

(b) Training loss curves for $\lambda_2 = 0.5$

(c) Training loss curves for $\lambda_2 = 1.5$

Figure 3: Training curves on fixed $\lambda_2 = 0.1, 0.5, 1.5$. The model is VGG13 on CIFAR-10

model, the absolute ratio values of penalty/reward λ ranges from 0.1 to 1.0, and for the other cases, λ ranges from 0.1 to 1.5. As shown in Figure 1, assuming CE loss tends to convex, we can see that the concavity in loss landscape increases as λ_2 increases, which shows that $\mathcal{L}_\pi^r$ is related to the convexity and $\mathcal{L}_\pi^p$ is related to the concavity. Therefore, by appropriately adjusting the λ value, we can change the loss landscapes. This indicates that it is possible to converge to different local optimal points even for the same optimizer.

In Figure 2, when increasing the λ, the optimal point of CE loss is no longer the optima of policy loss.

Table 1 shows some trade-offs between the reward ratio and the flatness. Since we execute two-step training, we denote the values λ as (λ_1, λ_2). When λ_1 was fixed, as the value λ_2 increased, it showed a tendency to have higher adversarial robustness. Conversely, When λ_2 was fixed, robustness showed a negative correlation with λ_1. During the first stage of the two stages of learning, if we increase λ_1, the convexity can be also increased if the penalty term of altenative loss Eq (13) contains $\pi_y(x)$ term, i.e, CE loss. In conclusion, it can be seen indirectly that the robustness increases as the concavity increases. Thus, it is necessary to appropriately balance the $\lambda = (\lambda_1, \lambda_2)$ values when training the model using our strategy. In the grayscale dataset MNIST, it seems that there is no particular tendency.

Table 1 shows that some combinations of λ_i $(i = 1, 2)$ have relatively low robustness compared to other λ. In the case $\lambda_2 = 0.5$, the training loss curves showed fluctuations, which implies having the lowest robustness for each λ_1 (See Figure 3).

Dataset	MNIST			CIFAR-10		
Attacks	Clean	FGSM	PGD	Clean	FGSM	PGD
CE	99.19	71.45	46.88	92.12	4.51	0
Ours ($\lambda_1 = 0.75, \lambda_2 = 1.5, 1.25$)	**99.21**	**86.40**	**63.73**	**92.16**	**64.87**	**49.1**

Table 3: The accuracy (%) between CE loss and our method.

Adversarial Robustness Compared to CE

We tested adversarial attacks on MNIST, CIFAR10. We then evaluated our method using FGSM and PGD only for white-box attacks. FGSM and PGD are set with $\epsilon = 0.04$. PGD has a step size of $\alpha = 0.01$ and number of steps $k = 40$ for each iteration.

Attacks	Clean	FGSM	PGD
CE with Eq (4) ($\lambda_2 = 1.0$)	92.21	18.77	0
Eq (12) with Eq (4) ($\lambda_1 = 0.75, \lambda_2 = 1.25$)	92.16	64.87	49.1
Use only Eq (4) ($\lambda_2 = 1.0$)	10.00	-	-

Table 4: Ablation result on CIFAR-10 shows the need to use the alternative loss function Eq (12) to maintain the effect of the origin policy loss.

Attacks	Perturbations	PC	Ours ($\lambda_1 = 0.75, \lambda_2 = 1.25$)
Clean	$\epsilon = 0.$	91.2	**92.1**
FGSM	$\epsilon = 0.04$	53.1	**64.8**
	$\epsilon = 0.12$	30.3	**34.9**
	$\epsilon = 0.2$	18.7	**21.64**
PGD	$\epsilon = 0.04$	27.6	**43.8**
	$\epsilon = 0.12$	14.6	**16.3**
	$\epsilon = 0.2$	**7.5**	6.4

Table 5: The accuracy (%) between PC and our method ($\lambda_1 = 0.75, \lambda_2 = 1.25$) on CIFAR-10 under the white-box setting. Results are directly from (Li et al. 2020).

Table 3 shows that our methods achieve both the highest accuracy and robustness. In addition to comparing the proposed method to the standard CE loss, we compare our defense approach with Probabilistically Compact Loss with Logit Constraints (Li et al. 2020) that achieved high robustness by simply using a drop-in replacement with pre-training CE loss. Since the hyperparameters of the attacks are the same, the results were extracted from the paper. Table 5 shows that our method outperforms PC loss on both clean accuracy and robustness.

Improving Robustness with Adversarial Pre-trained Model

We investigate the performance of our method in combination with adversarial training TRADES . We use ResNet-18 as a backbone with all the hyperparameters setting used in the original paper (Zhang et al. 2019) : regularization hyperparameter $\beta = 6.0$, SGD optimizer with momentum 0.9, weight decay 10^{-4}, 100 epoch and the initial learning rate 0.1 divided by 10 at the 75-th. After 75 epoch, we replace the CE loss just by using Eq (4) without randomness of action.

Dataset	CIFAR-10		
Attacks	Clean	PGD	AutoAttack
TRADES (CE)	82.31	52.55	48.25
TRADES ($\lambda = 0.1$)	76.21	52.62	48.83
TRADES ($\lambda = 0.25$)	81.14	53.62	49.10
TRADES ($\lambda = 0.5$)	81.22	**54.15**	**49.46**
TRADES ($\lambda = 0.75$)	81.20	53.96	49.11
TRADES ($\lambda = 1.0$)	82.07	53.32	48.37
TRADES ($\lambda = 1.25$)	82.41	53.83	48.39
TRADES ($\lambda = 1.5$)	**82.83**	53.83	48.14

Table 6: The accuracy (%) on CIFAR-10 with pre-training using TRADES against $\epsilon = 8/255$ PGD with 20 iteration and step size 0.003 and Auto-Attack.

Since the advantage of adversarial defense was sufficiently acquired with AT, we do not pre-training using Eq (12). We evaluated using PGD with with the same set up as above, and additionally tested on a parameter free attack AutoAttack (Croce and Hein 2020); we tested on only CIFAR-10 dataset.

With better exploration of the policy gradient, we were able to confirm that accuracy and defense were improved simultaneously at the same time in the same situation despite the trade-off between clean accuracy and robust accuracy (See $\lambda = 1.25$ case of Table 6). It is possible to achieve better performance with simple substitutions.

Conclusion

In multi-classification using reinforcement learning, the reward function is crucial and has various advantages because of its non-convexity. The policy loss is affected by the reward function in a gradient-based trajectory search to enable trajectory searches along various routes by deforming the loss landscape. It was confirmed that the proposed method provides a great advantage against adversarial attacks in the vision domain without sacrificing clean accuracy. In the future, it is necessary to check what advantages there are for various other domains and a need for studying its application to various well- known model structures using a canonical CE function, just like a transformer (Vaswani et al. 2017). The latest reinforcement learning techniques should be deconstructed, simplified, and studied, such as Actor-Critic Algorithms (Konda and Tsitsiklis 2000). Furthermore, we can study more diverse geometries by continuously producing deformations in various reward environments.

References

Carmon, Y.; Raghunathan, A.; Schmidt, L.; Liang, P.; and Duchi, J. C. 2019. Unlabeled data improves adversarial robustness. arXiv:1905.13736.

Chen, H.-Y.; Liang, J.-H.; Chang, S.-C.; Pan, J.-Y.; Chen, Y.-T.; Wei, W.; and Juan, D.-C. 2019. Improving adversarial robustness via guided complement entropy. In *Proceedings of the IEEE International Conference on Computer Vision,*, 4881–4889.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning. PMLR.*, 2206–2216.

Feng, J.; Huang, M.; Zhao, L.; Yang, Y.; and Zhu, X. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580–587.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *In Advances in neural information processing systems*, 2672–2680.

Goodfellow, I. J.; Shlens, J.; ; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 1–11.

Gu, E., S.and Holly; Lillicrap, T.; and Levine, S. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017*, 3389–3396.

Gupta, S. K. 2020. Reinforcement Based Learning on Classification Task Could Yield Better Generalization and Adversarial Accuracy. arXiv:2012.04353.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hochreiter, S.; and Schmidhuber, J. 1997. Flat minima. *Neural computation*, 9(1): 1–42.

Janisch, J.; Pevny, T.; and Lis‘y, V. 2017. Classification with costly features using deep reinforcement learning. arXiv:1711.07364.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*.

Konda, V. R.; and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014.

Krizhevsky, A.; Ilya, S.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems 25*, 1097–1105.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2017. Improving adversarial robustness via probabilistically compact loss with logit constraints. arXiv:1712.09913..

Li, X.; Li, X.; Pan, D.; and Zhu, D. 2020. Improving adversarial robustness via probabilistically compact loss with logit constraints. arXiv:2012.07688.

Lin, E.; Chen, Q.; and Qi, X. 2020. Deep reinforcement learning for imbalanced classification. In *Applied Intelligence, 50(8)*, 2488–2502.

Liu, D.; and Jiang, T. 2018. Deep reinforcement learning for surgical gesture segmentation and classification. arXiv:1806.08089.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

Martinez, C.; Perrin, G.; Ramasso, E.; and Rombaut, M. 2018. A deep reinforcement learning approach for early classification of time serie. In *EUSIPCO 2018*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. arXiv:1312.5602.

Pang, T.; Xu, K.; Dong, Y.; Du, C.; Chen, N.; and Zhu, J. 2019. Rethinking Softmax Cross-Entropy Loss for Adversarial Robustness. arXiv:1905.10626.

Rebuffi, S. A.; Gowal, S.; Calian, D. A.; Stimberg, F.; Wiles, O.; and Mann, T. 2021. Fixing data augmentation to improve adversarial robustness. arXiv:2103.01946.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556..

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, Y.; Zou, D.; Yi, J.; Bailey, J.; Ma, X.; and Gu, Q. 2019. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*.

Wiering, M.; Hasselt, H.; Pietersma, A.-D.; and Schomaker, L. 2011. Reinforcement learning algorithms for solving classification problems. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on. IEEE, 2011,*, 91–96.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2021. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM, 64(3),*, 107–115.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; and El Ghaoui, M., L.and Jordan. 2019. Theoretically Principled Trade-off between Robustness and Accuracy. In *International Conference on Machine Learning, PMLR.*, 7472–7482.

Zhang, T.; Huang, M.; and Zhao, L. 2018. Learning structured representation for text classification via reinforcement learning. *AAAI.*