A. Generated Images on CIFAR-100

We show the images generated by JEAT on CIFAR-100 in Fig. 1. These images are rich in details and vivid.

![Generated Images on CIFAR-100](image)

Figure 1. Images generated by JEAT on CIFAR-100.

B. The Change of Energy

B.1. The Change of Energy $E_0(x, y)$ in Training

We illustrate the change of energy $E_0(x, y)$ in adversarial training on CIFAR-100 in Fig. 2. Adversarial attacks generate high-energy adversarial examples, and then the energy of adversarial examples is decreased by updating model parameters. As we show in our paper, adversarial training flattens the energy region around real data in this way.

![Energy Change in Training](image)

Figure 2. We illustrate the changes of energy in original adversarial training [4] on CIFAR-100 in 50 epochs (model has converged). The center points of the tags represent the mean value and the lengths represent the variance. Adversarial examples increase the energy $E_0(x, y)$ as $\Delta E_0(x, y) > 0$ during the training. The energy $E_0(x, y)$ of adversarial examples decrease after updating parameters as $\Delta E_0(x, y) < 0$ during the training.

B.2. The Change of Energy $E_0(x)$ in Training

We illustrate the change of energy $E_0(x)$ in adversarial training on CIFAR-10 in Fig. 3, and CIFAR-100 in Fig. 4. The value of $\Delta E_0(x)$ fluctuates around zero, sometimes positive and sometimes negative. Thus $E_0(x)$ has not been well optimized in the classification task. Using $E_0(x)$ in the generation task by a robust classifier may be harmful to the images' quality.

![Energy Change in Training](image)

Figure 3. We illustrate the changes of energy $E_0(x)$ in original adversarial training [4] on CIFAR-10 in 50 epochs (model has converged). The center points of the tags represent the mean value and the lengths represent the variance.

B.3. The Change of Energy in Generation

Image generation by an original robust classifier is introduced in the main paper, and the generating procedure of images can be formulated as:

$$x' = x - \frac{\eta}{2} \cdot \nabla_x (E_0(x, y) - E_0(y)) + \sqrt{\eta} e.$$  \hspace{1cm} (1)

As analyzed in Sec. 2.2, $E_0(x, y)$ is the key term for conditional generation by using robust classifier. We illustrate $\nabla_x E_0(x, y)$ and $\nabla_y E_0(x)$ in generation process in Fig. 5 (CIFAR-10) and Fig. 6 (CIFAR-100). In the beginning, $\nabla_x E_0(x, y)$ dominates for that the absolute value of $\nabla_x E_0(x)$ is less than that of $\nabla_y E_0(x, y)$. As the num-
Table 1. Hyperparameters of different methods.

<table>
<thead>
<tr>
<th>HYPERPARAMETER</th>
<th>LEARNING RATE</th>
<th>BATCH SIZE</th>
<th>SGD SCHEDULER</th>
<th>EPOCHS</th>
<th>WEIGHT DECAY</th>
<th>MOMENTUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEAT(Ours)</td>
<td>1E-3</td>
<td>64</td>
<td>ADAM[160,180]</td>
<td>200</td>
<td>0</td>
<td>BETA=[0.9,0.999]</td>
</tr>
<tr>
<td>JEM</td>
<td>1E-4</td>
<td>64</td>
<td>ADAM[160,180]</td>
<td>200</td>
<td>0</td>
<td>BETA=[0.9,0.999]</td>
</tr>
<tr>
<td>FREE</td>
<td>0.1</td>
<td>128</td>
<td>MULTISTEP[100,150]</td>
<td>200</td>
<td>0</td>
<td>0.9</td>
</tr>
<tr>
<td>FAST</td>
<td>0-0.2</td>
<td>128</td>
<td>CYCLIC</td>
<td>15</td>
<td>5E-4</td>
<td>0.9</td>
</tr>
<tr>
<td>TRADHS</td>
<td>0.1</td>
<td>128</td>
<td>MULTISTEP[75,90,100]</td>
<td>120</td>
<td>2E-4</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The gradient on $\theta$ of $\log p_\theta(x)$ is defined as
\[
\nabla_\theta \log(p_\theta(x)) = -\nabla_\theta E_\theta(x) - \frac{1}{Z_\theta} \nabla_\theta Z_\theta = -\nabla_\theta E_\theta(x) - \frac{1}{Z_\theta} \nabla_\theta \int_x \exp(-E_\theta(x))dx = -\nabla_\theta E_\theta(x) + \int_x \frac{\exp(-E_\theta(x))}{Z_\theta} \nabla_\theta E_\theta(x)dx = -\nabla_\theta E_\theta(x) + \mathbb{E}_{p_\theta(x)}(\nabla_\theta E_\theta(x))
\]

We use Stochastic Gradient Langevin Dynamics (SGLD) to approximate $p_\theta(x)$.

D. Proof for $Z_\theta = \tilde{Z}_\theta$

As defined in the main paper, the energy functions for classifier are:
\[
\begin{align*}
E_\theta(x, y) &= -\log(\exp(f(x; \theta)[y])), \\
E_\theta(x) &= -\log(\sum_y \exp(f(x; \theta)[y])).
\end{align*}
\]

Also $p_\theta(x, y) = \frac{\exp(-E_\theta(x, y))}{Z_\theta}$. $\tilde{Z}_\theta$ is the normalizing constant which is defined as:
\[
\begin{align*}
\tilde{Z}_\theta &= \int_x \sum_y \exp(-E_\theta(x, y))dx = \int_x \sum_y \exp(f(x; \theta)[y])dx
\end{align*}
\]

As $p_\theta(x) = \frac{\exp(-E_\theta(x))}{Z_\theta}$, $Z_\theta$ is the normalizing constant which is defined as:
\[
\begin{align*}
Z_\theta &= \int_x \exp(-E_\theta(x))dx = \int_x \exp(\sum_y \exp(f(x; \theta)[y]))dx = \int_x (\sum_y \exp(f(x; \theta)[y]))dx
\end{align*}
\]

By Eq. (5) and Eq. (6), it can be seen that $Z_\theta = \tilde{Z}_\theta$.

E. Algorithm for PreJEAT

In the main paper, we show the training algorithm called Joint Energy Adversarial Training (JEAT), in which we replace cross-entropy loss $-\log p_\theta(y|x)$ with $-\log p_\theta(x, y) = -\log p_\theta(y|x) - \log p_\theta(x)$ and use adversarial examples found by
\[
x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x E_\theta(x, y)).
\]

We also propose Preliminary Joint Energy Adversarial Training (PreJEAT) which just trains model with adversarial examples found by Eq. (7) and use cross-entropy loss. We present the algorithm here in Algorithm 1.

C. The Gradient on $\theta$ of $\log p_\theta(x)$

$p_\theta(x)$ in JEAT algorithm can be expressed as:
\[
p_\theta(x) = \frac{\exp(-E_\theta(x))}{Z_\theta},
\]
Algorithm 1 Training and Generating of PreJEAT: Given network \( f, E(x, y) = -\log(\text{exp}(f(x)[y])) \) represent energy of \((x, y), E(x) = -\log(\sum_y \text{exp}(f(x)[y])) \) represent energy of \( x \), \( \epsilon \) is adversarial perturbation radius \( e \), SGLD step-size \( \alpha \), SGLD steps \( K \), epochs \( T \), dataset of size \( M \), learning rate \( \eta \).

Training:
for \( i = 1, 2, ..., T \) do
    for \( j = 1, 2, ..., M \) do
        ► Generating energy-based adversarial samples:
        \( \delta = \mathcal{U}(-\epsilon, \epsilon) \)
        \( \delta = C \cdot \text{sign}(\nabla_x \mathcal{L}_\theta(x, y)) \)
        \( \delta = \max(\min(\delta, \epsilon), -\epsilon) \)
        \( x_{adv} = x_j + \delta \)
        ► Updating model parameters:
        \( \nabla_\theta \mathcal{L}_{\text{gen}}(y|x_{adv}) = \nabla_\theta \mathcal{L}(\theta(x_{adv}, y) - \theta(x_{adv})) \)
        \( \theta = \theta - \eta \cdot \nabla_\theta \mathcal{L}_{\text{gen}}(y|x_{adv}) \)
    end for
end for

Generating:
\( x_0 \sim \text{random sample} \)
for \( t = 0, 1, 2, ..., K - 1 \) do
    \( x_{t+1} = x_t - \frac{\alpha}{2} \cdot \nabla_x \mathcal{L}(x_t, y) + \sqrt{\alpha} \cdot \mathcal{N}(0, I) \)
end for

Output: \( x_{gen} = x_K \)

F. Training Details

The hyper-parameters for our experiments are shown in Tab. 1. We run JEM [2], Free adversarial training (Free \( m=8 \)) [3], Fast adversarial training (Fast) [6] and TRADES (1/\( \lambda=6 \)) [7] with their open-source code. MultiStep scheduler decay by 10 every time.

G. Interesting Benefits of JEAT

G.1. Denoise

Due to the influence of factors such as the environment and transmission channels, the image is inevitably contaminated by noise in the process of acquisition, compression and transmission. In the presence of noise, subsequent image processing tasks (such as video processing, image analysis, and tracking) may be negatively affected. Therefore, image denoising plays an important role in modern image processing systems. In this section, we found that the classifier trained by JEAT has the effect of denoising to some extent.

We further show the energy contours of the normal classifier, robust classifier and JEAT classifier around a given image from test dataset (CIFAR-10) in Fig. 7. The points with the lowest energy of both the normal classifier and the robust classifier deviate from the point where the real image is located. The generation process will result in entering the low energy positions along the direction of energy descent from the real image. Therefore, even if it starts from the real image, the image generated by the normal classifier will be inferior. Because the low-energy region of the original robust classifier deviates slightly from the real data, the quality of generated images will be slightly better than that of the normal classifier. Nevertheless, in the energy of JEAT classifier, the real image has the minimum energy, and the energy function is flat and smooth around the real data. Starting from the real image, the image generated by the JEAT classifier is almost identical to the real image.

This inspires us that if the energy of the noise image is high, it is possible to generate a clean image along the direction of energy descent, thereby denoising. We show the different energy contours in the process of denoising by JEAT classifier in Fig. 8. We get noisy images by injecting Gaussian noise with a mean value of zero and a variance of 0.3 into the original image. Then we generate low-energy images from these noisy images using Langevin Dynamics. As shown in Fig. 9, JEAT classifier has good denoising...
Figure 9. We show that compared to the normal classifier and original robust classifier, JEAT classifier can indeed achieve the effect of denoising. (a) Original images. (b) Noise images obtained by superimposing Gaussian noise with a mean value of 0 and a variance of 0.3 on the original image. (c) The image obtained after denoising by the normal classifier. (d) The image obtained after denoising by the robust classifier. (e) The image obtained after denoising by the JEAT classifier.

effect, while normal classifier and original robust classifier have poor denoising effect.

G.2. Calibration

The outputs of a classifier are often interpreted as the predictive confidence that this class was identified. However Guo et al. [3] claim that deep neural networks are often not calibrated which means that the confidence always does not align with misclassification rate. Expected Calibration Error (ECE) is a metric to measure the calibration of a classifier. For a perfectly calibrated classifier, ECE value will be zero.

We using the reliability diagram to find out how well the
classifier is calibrated in Fig. 10. The model’s predictions are divided into bins based on the confidence value of the target class, here, we choose 20 bins. The confidence histogram at the bottom shows how many test examples are in each bin. Two vertical lines represent accuracy and average confidence, and the closer these two lines are, the better the model calibration is. The reliability histogram at the top shows the average confidence of each bar and the accuracy of the examples in the bar. For each bin we plot the difference between the accuracy and the confidence using the red bars in the diagram.

It can be clearly seen from the histogram that the confidence of most predictions of these classifiers is greater than 0.8. The normal classifier is always over-confidence and gives more false positives. This phenomenon becomes even worse when facing adversarial data and the ECE value is 49.63%. Robust classifier will not give much over-confident prediction when encountering adversarial perturbation. However, when faced with clean data, the robust classifier is under-confidence and gives more false negatives. The good news is that the JEAT classifier has a good calibration when classifying clean data or adversarial data. When a model is deployed in a real-world scenarios, good calibration is an important feature. And the confidence of a model with good calibration can be used to judge whether to output the result or recognize it again. The JEAT classifier which is better calibrated than both normal classifier and original robust classifier can be more useful in real-world scenarios.

References


