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.

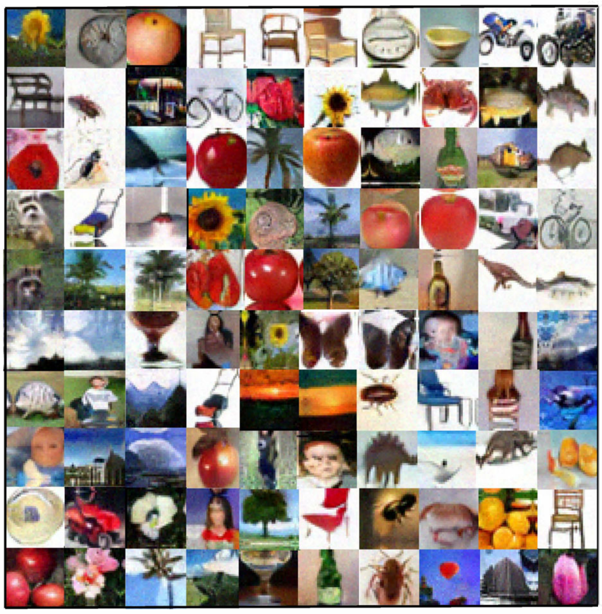

Figure 1. Images generated by JEAT on CIFAR-100.

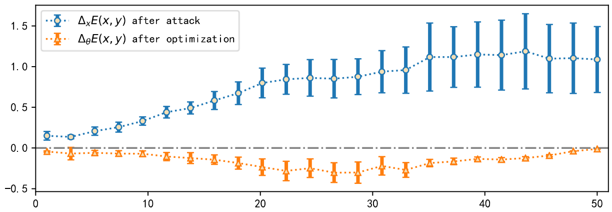

Figure 2. We illustrate the changes of energy in original adversarial training [4] on CIFAR-100 in 50 epochs (model has converged). The center points of the tags represent the mean value and the lengths represent the variance. Adversarial examples increase the energy $E_\theta(x, y)$ as $\Delta_x E_\theta(x,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

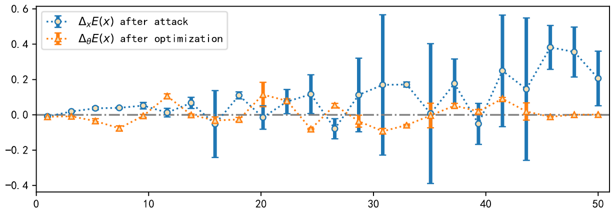

Figure 3. We illustrate the changes of energy $E_\theta(x)$ in original adversarial training [4] on CIFAR-10 in 50 epochs (model has converged). The center points of the tags represent the mean value and the lengths represent the variance.

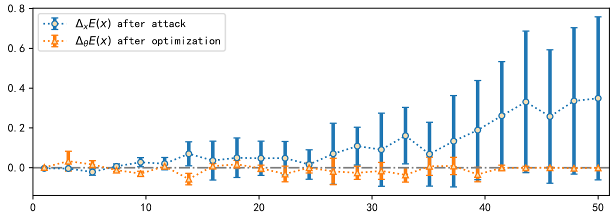

Figure 4. We illustrate the changes of energy $E_\theta(x)$ in original adversarial training [4] on CIFAR-100 in 50 epochs (model has converged). The center points of the tags represent the mean value and the lengths represent the variance.

B. The Change of Energy

B.1. The Change of Energy $E_\theta(x, y)$ in Training

We illustrate the change of energy $E_\theta(x, y)$ 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.

B.2. The Change of Energy $E_\theta(x)$ in Training

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

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:

$$x' = x - \frac{\eta}{2} \cdot \nabla_x (E_\theta(x, y) - E_\theta(x)) + \sqrt{\eta}\epsilon. \quad (1)$$

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

Table 1. Hyperparameters of different methods.

Hyperparameter	Learning Rate	Batch Size	SGD Scheduler	Epochs	Weight Decay	Momentum
JEAT(ours)	1e-4	64	Adam[160,180]	200	0	beta=[0.9,0.999]
JEM	1e-4	64	Adam[160,180]	200	0	beta=[0.9,0.999]
Free	0.1	128	MultiStep[100,150]	200	2e-4	0.9
Fast	0-0.2	128	Cyclic	15	5e-4	0.9
TRADES	0.1	128	MultiStep[75,90,100]	120	2e-4	0.9

ber of iterations increases, the influence of $\nabla_x E_\theta(x)$ and $\nabla_x E_\theta(x, y)$ gradually become much more closer. Thus, $E_\theta(x, y)$ plays a key role for conditional generation by using robust classifier.

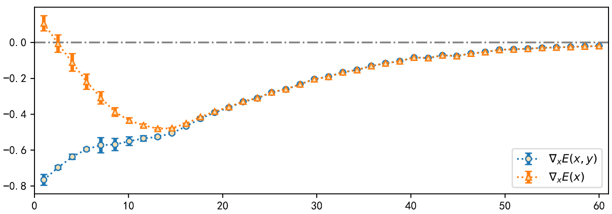

Figure 5. $E_\theta(x, y)$ plays a major role when generating images by the original robust classifier (CIFAR-10). We illustrate the $\nabla_x E_\theta(x, y)$ and $\nabla_x E_\theta(x)$ in the generation process of a robust classifier (WRN-28-10) on CIFAR-10. The center points of the tags represent the mean value and the lengths represent the variance. The horizontal axis represents the number of iterations.

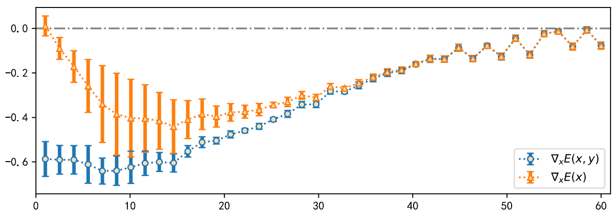

Figure 6. $E_\theta(x, y)$ plays a major role when generating images by the original robust classifier (CIFAR-100). We illustrate the $\nabla_x E_\theta(x, y)$ and $\nabla_x E_\theta(x)$ in the generation process of a robust classifier (WRN-28-10) on CIFAR-100. The center points of the tags represent the mean value and the lengths represent the variance. The horizontal axis represents the number of iterations.

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

$p_\theta(x)$ in JEAT algorithm can be expressed as:

$$p_\theta(x) = \frac{\exp(-E_\theta(x))}{Z_\theta}, \tag{2}$$

The gradient on θ of $\log p_\theta(x)$ is defined as

$$\begin{aligned}\nabla_\theta \log(p_\theta(x)) &= -\nabla_\theta E_\theta(x) - \tfrac{1}{Z_\theta}\nabla_\theta Z_\theta \\ &= -\nabla_\theta E_\theta(x) - \tfrac{1}{Z_\theta}\nabla_\theta \int_x \exp(-E_\theta(x))dx \\ &= -\nabla_\theta E_\theta(x) + \int_x \tfrac{\exp(-E_\theta(x))}{Z_\theta}\nabla_\theta E_\theta(x)dx \\ &= -\nabla_\theta E_\theta(x) + \mathbb{E}_{p_\theta(x)}(\nabla_\theta E_\theta(x))\end{aligned} \tag{3}$$

We use Stochastic Gradient Langevin Dynamics (SGLD) to approximate $\mathbb{E}_{p_\theta(x)}$.

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

As defined in the main paper, the energy functions for classifier are:

$$\begin{cases} E_\theta(x, y) = -\log(\exp(f(x;\theta)[y])), \\ E_\theta(x) = -\log(\sum_{k=1}^{n}\exp(f(x;\theta)[k])). \end{cases} \tag{4}$$

Also $p_\theta(x, y) = \frac{\exp(-E_\theta(x,y))}{\tilde{Z}_\theta}$, $\tilde{Z}_\theta$ is the normalizing constant which is defined as:

$$\begin{aligned}\tilde{Z}_\theta &= \int_x \sum_y \exp(-E_\theta(x, y))dx \\ &= \int_x \sum_y \exp(f(x;\theta)[y])dx\end{aligned} \tag{5}$$

As $p_\theta(x) = \frac{\exp(-E_\theta(x))}{Z_\theta}$, Z_θ is the normalizing constant which is defined as:

$$\begin{aligned}Z_\theta &= \int_x \exp(-E_\theta(x))dx \\ &= \int_x \exp(\log(\sum_y \exp(f(x;\theta)[y])))dx \\ &= \int_x (\sum_y \exp(f(x;\theta)[y]))dx\end{aligned} \tag{6}$$

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

$$x_{adv} = x + \epsilon \cdot sign(\nabla_x(E_\theta(x, y))). \tag{7}$$

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.

Algorithm 1 Training and Generating of PreJEAT: Given network f, $E(x,y) = -\log(\exp(f(x)[y]))$ represent energy of (x,y), $E(x) = -\log(\sum_y \exp(f(x)[y]))$ represent energy of x, ℓ_∞ adversarial perturbation radius ϵ, SGLD step-size α, SGLD steps K, epochs T, dataset of size M, learning rate η.

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

Generating:
$x_0 \sim$ random sample
for $t = 0, 1, 2..., K-1$ **do**
  $x_{t+1} = x_t - \frac{\alpha}{2} \cdot \nabla_{x_t} E_\theta(x_t, y) + \sqrt{\alpha} \cdot \mathcal{N}(0, I)$
end for
Output: $x_{gen} = x_K$

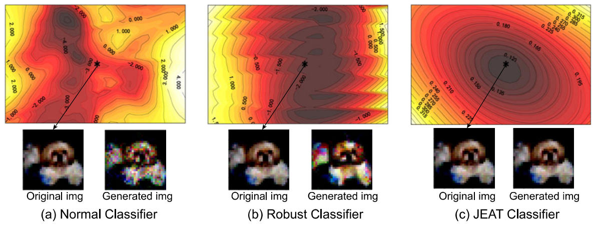

Figure 7. We plot the energy contour around a given image of normal classifier (a), robust classifier (b), and JEAT classifier (c). The center point represents the energy of the given image. Darker colors in the diagram correspond to lower energy. Starting from the original image, we generate images by different classifiers using Langevin Dynamics.

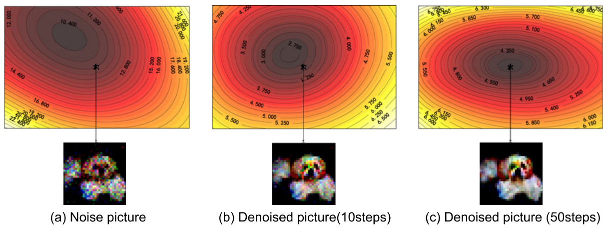

Figure 8. We show the change of energy contour in the process of denoising. The initial noisy image has higher energy, and then the denoised image is gradually obtained along the direction of energy descent.

F. Training Details

The hyper-parameters for our experiments are shown in Tab. 1. We run JEM [2], Free adversarial training (Free m=8) [5], Fast adversarial training (Fast) [6] and TRADES (1/λ=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 an 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

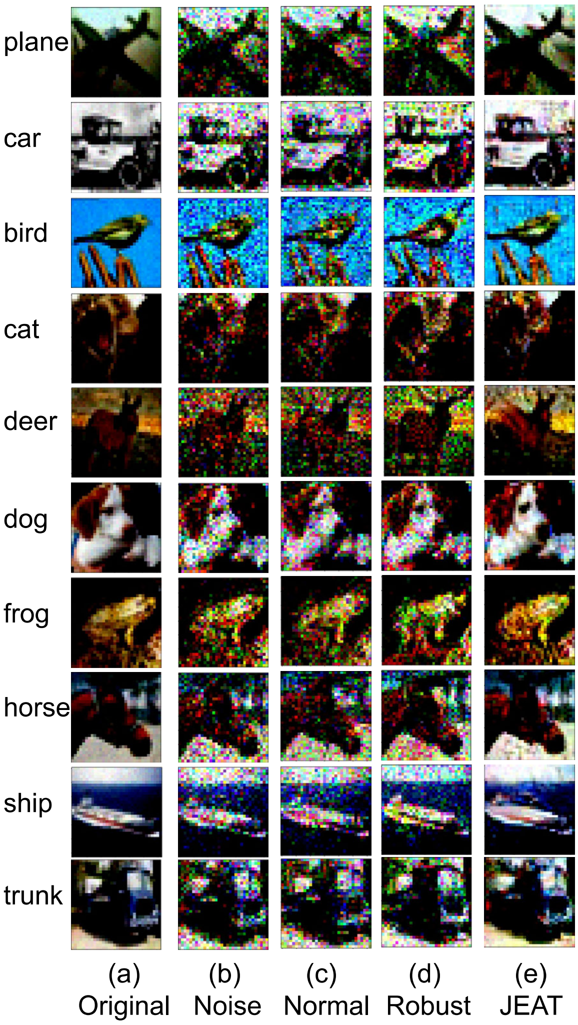

Figure 9. We show that compared to the normal classifier and original robust classifier, JEAT classifier can indeed achieve the effect of denoising. (a) Original images. (b) Noise images obtained by superimposing Gaussian noise with a mean value of 0 and a variance of 0.3 on the original image. (c) The 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.

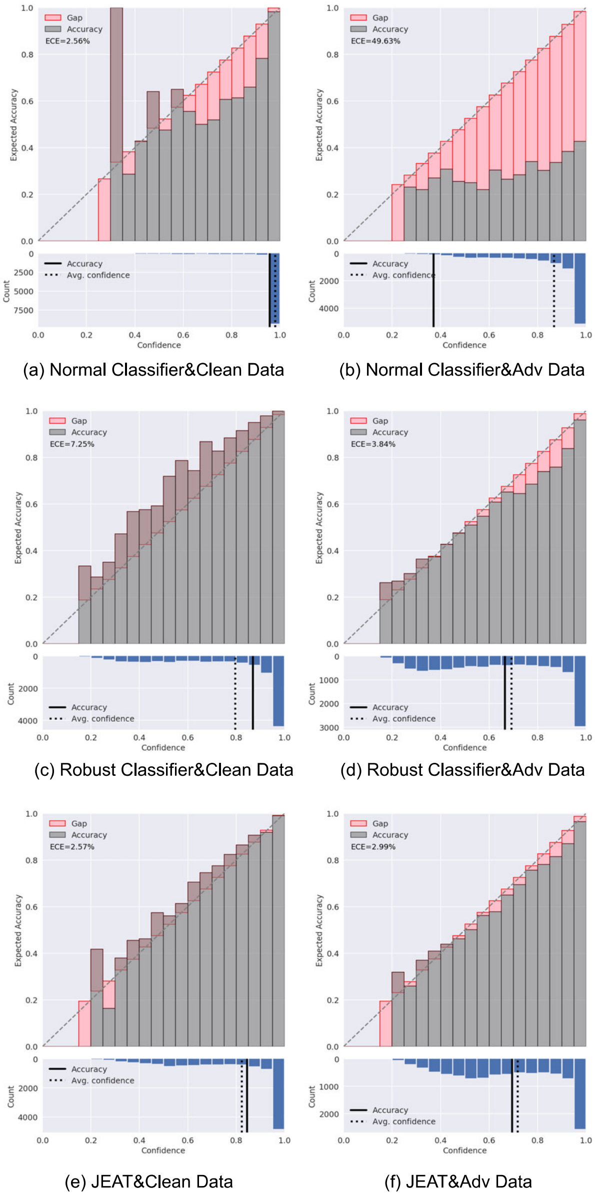

Figure 10. We draw the reliability diagram of different classifiers on original test data of CIFAR-10 and adversarial data. We use FGSM attack ($\epsilon = 8/255$) [1] to get adversarial data. (a) The reliability diagram of normal classifier on original clean data. (b) The reliability diagram of normal classifier on adversarial data. (c) The reliability diagram of robust classifier on original clean data. (d) The reliability diagram of robust classifier on adversarial data. (e) The reliability diagram of JEAT classifier on original clean data. (f) The reliability diagram of JEAT classifier on adversarial data.

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

[1] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015. 4

[2] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one, 2020. 3

[3] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. 4

[4] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 1

[5] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3353–3364, 2019. 3

[6] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020. 3

[7] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy, 2019. 3