Supervised Learning via Conditional Sampling with Energy-Based Models

Katherine Abreu, Dr. Mitchell Hill, Dr. Mubarak Shah
University of Central Florida
kkatherineaabreu@gmail.com, mitchell.hill@ucf.edu, shah@crcv.ucf.edu

Abstract

In this paper, we introduce a new method that doesn’t rely on cross-entropy loss and focuses on Energy-Based Models (EBMs) and conditional Markov Chain Monte Carlo (MCMC) sampling to improve the robustness of supervised learning. Normally, with the help of supervised learning, many models are able to obtain a high natural accuracy, but due to their prediction probabilities, they are left vulnerable to adversarial attacks. We approach this problem by training EBMs with conditional MCMC sampling using the CIFAR-10 dataset. Rather than use a normal classifier, we train EBMs to learn the true energy surface level of the labels conditional on images, making the defense stronger against adversarial attacks.

1 Introduction

When it comes to supervised learning, given an image, the model uses a probabilistic objective function where prediction probabilities are formed after performing a forward-pass in the network. These output probabilities match the ground-truth image labels where the labels assign a probability of one to the true class and a zero to the rest of the classes. Many models that use supervised learning are unreliable because their probabilistic outputs end up being poorly calibrated, which makes the models susceptible to adversarial attacks, reaching an accuracy as low as zero percent.

One of the open challenges of image classification is having a model that not only provides a high natural accuracy, but also defends well against adversarial attacks, such as Projected Gradient Descent (PGD). This white-box attack allows the attacker to create an image that looks similar to the original while also applying maximum noise to it. The end result is an adversarial image that tricks the model into classifying the image incorrectly. Thus, having an image with a small perturbation can lead to incorrect classification when it comes to supervised learning.

In our case, to help avoid this problem, we introduce a novel idea for supervised learning where we train an EBM to synthesize labels conditional on an image utilizing conditional MCMC sampling. Instead of generating label probabilities through a forward-pass, the label probabilities are coming from an energy surface. At first, labels are initialized from uniform noise in the logit space, but are later sent through the MCMC process conditional on its corresponding image where it gradually changes. After the MCMC process, the final synthesized label vector equates to the label probabilities for its image. The index from the label vector with the highest logit value corresponds to the class it predicts, similar to the prediction probabilities a normal classifier outputs. Because we focus on the energy surface itself, we are able to provide a stronger defense for predicting the class of an image, creating a much more robust decision boundary.

2 Related Work

There have been many works that used EBMs to synthesize realistic images [1, 2, 3, 4]. The two most important concepts that they focused on was the maximum-likelihood function and the MCMC process, which we utilized in our work.

Along with these works, there was one paper in particular that was able to train EBMs to purify adversarial images using MCMC sampling, removing as much noise as it could while still preserving its original image [5]. This paper proved that using MCMC sampling did provide some robustness to normal classifiers, which led us to take MCMC sampling a step further and make it conditional.

Another important concept [1] explains is the importance of calculating the energy difference between the positive and negative energy samples when learning maximum-likelihood. Because we are also incorporating EBMs and MCMC sampling, it’s important that the
Figure 1: How the label state travels downhill during conditional MCMC sampling in order to classify an image. Given an image, the model produces an energy landscape in the logit space. At the start of the MCMC process, the synthesized label state gets uniformly initialized. At each step of MCMC sampling, when we send to the network the synthesized label state and the image it’s conditional to, the synthesized label state goes down the gradient of the image’s energy landscape and the final label state gets located around the region of the correct prediction. This region has lower energy compared to the regions of the other classes and the final label state ends up predicting the correct class.

A previous work explained standard vs. adversarial decision boundaries [6]. A regular classifier is able to classify images relatively easy using a decision boundary where the images are classified according to the side it’s located on. The reason why these classifiers aren’t able to provide a good robustness is because there could be adversarial examples that pass the decision boundary and get misclassified. As mentioned before, our EBM model is able to create a more robust decision boundary, making it harder for adversarial examples to get misclassified.

### 3 Process

#### 3.1 Learning Energy Based Models

An EBM is essentially a probabilistic model and can be used to represent an unnormalized density distribution of states in a system. Our goal during training is to get the probabilistic model $p_\theta(x, y)$ to have a close approximation to the data distribution $q_\theta(x, y)$. Since we synthesize the labels conditional on the image, we would have to focus on learning the distribution $p_\theta(y|x)$.

$$p_\theta(y|x) = \frac{1}{Z_x(\theta)} \exp\{-U(y|x; \theta)\}$$  \hfill (1)

Although the intractable constant $Z_x(\theta) = \int_y \exp\{-U(y|x; \theta)\} dy$ can’t be solved, the network $U(x, y; \theta)$ is enough for MCMC sampling. In this case, $x$ represents the images and $y \in \mathbb{R}^D$ represents the labels with $D$ being the number of classes. In terms of our network, $U(x, y; \theta)$ follows a similar pattern to the discriminator for conditional Generative Adversarial Networks [7] introduced in their paper. For our network, the label gets sent through a set of fully connected layers while the images get sent through convolutional layers. The outputs of each get added to then go through a final set of fully connected layers. It finalizes with a single output channel, returning a scalar value.

To get $p_\theta(x, y)$ to be close to the distribution $q_\theta(x, y)$, we would be learning by minimizing

$$\mathcal{L}(\theta) = -E_{q(x)}E_{q(y|x)}[\log(q(x)p_\theta(y|x))].$$  \hfill (2)

Rather than rely on cross-entropy loss, we use Contrastive Divergence (CD) to estimate the energy function’s gradient [8]. In this case, following Baye’s Theorem, $q(x)q(y|x)$ represents the distribution $q(x, y)$ and
$q(x)p_0(y|x)$ represents the distribution $p_0(x,y)$. Because we are learning the parameters $\theta$, $log(q(x))$ in (2) can be taken out as a constant term. Looking at (1), we are able to expand it and get

$$\mathcal{L}(\theta) = -E_{q(x)q(y|x)}[logp_0(y|x)] + C \quad (3)$$

$$= E_{q(x)q(y|x)}[U(y|x;\theta)] + E_{q(x)}[logZ_x(\theta)] + C. \quad (4)$$

The expectation in regards to $q(y|x)$ disappears in (4) because $y$ isn’t involved in $logZ_x(\theta)$. We can then minimize $\mathcal{L}(\theta)$ by taking the derivative. When doing so, we know that $\frac{d}{d\theta} logZ_x(\theta)$ is intractable, but it can be rewritten as

$$\frac{d}{d\theta} logZ_x(\theta) = E_{p_0(y|x)}[-\frac{\partial}{\partial \theta} U(y|x;\theta)]. \quad (5)$$

Based on (5) and when taking the derivative of $E_{q(x)}[logZ_x(\theta)]$, we get

$$E_{q(x)} \left[ \frac{d}{d\theta} logZ_x(\theta) \right] = E_{q(x)} \left[ E_{p_0(y|x)} \left[ -\frac{\partial}{\partial \theta} U(y|x;\theta) \right] \right] \quad (6)$$

$$= -E_{q(x)p_0(y|x)} \left[ \frac{\partial}{\partial \theta} U(y|x;\theta) \right]. \quad (7)$$

Now when we find $\frac{d}{d\theta} \mathcal{L}(\theta)$, we are able to learn $\theta$. $\frac{d}{d\theta} \mathcal{L}(\theta) = \frac{d}{d\theta} E_{q(x)q(y|x)}[U(y|x;\theta)]$

$$- E_{q(x)p_0(y|x)} \left[ \frac{\partial}{\partial \theta} U(y|x;\theta) \right] \quad (8)$$

$$\approx \frac{\partial}{\partial \theta} \left( \frac{1}{n} \sum_{i=1}^{n} U(Y_i^+|X_i^+;\theta) - \frac{1}{m} \sum_{i=1}^{m} U(Y_i^-|X_i^+;\theta) \right) \quad (9)$$

In this case, $\{X_i^+\}_{i=1}^n$ are positive images taken from the data distribution $q$ and $\{Y_i^+\}_{i=1}^n$ are positive labels that are conditional on the positive images. The other set of images $\{X_i^+\}_{i=1}^m$ are another set of positive images from the data distribution $q$ and $\{Y_i^+\}_{i=1}^m$ are the negative labels taken from the distribution $p$. These negative labels are conditional on $\{X_i^+\}_{i=1}^m$. During training, $\{X_i^+\}_{i=1}^n$, $\{Y_i^+\}_{i=1}^n$, and $\{X_i^+\}_{i=1}^m$ are taken from the training set while $\{Y_i^+\}_{i=1}^m$ are the ones synthesized after MCMC sampling. Looking at (9), $n$ and $m$ would normally be the same value and represent the batch size.

In the end, the goal is for the energy difference in (9) to reach approximately zero. As shown in (9), when calculating the positive energy, we send into the network $U$ the positive images and the positive labels conditional to it. When calculating the negative energy, we send into the network $U$ another batch of positive images and the synthesized labels conditional to it. For the energy difference to be around zero, this would mean that given an image, the model is able to place the synthesized labels close to the real labels of the image in an energy landscape because they have similar energy levels.
3.2 MCMC Sampling in Logit Space

One important concept is how the labels are represented. Normally, an output probability prediction for an image is a label vector where each value is a probability between zero and one. The index with the highest probability is the class the vector predicts. Instead, we decided to represent the numbers as logits. That way, they are in a real number space with less restrictions. If we were to keep it in a probability space, we might end up getting probabilities that are negative, that are over one, or that all the probabilities in the vector added up together do not end up adding up to one.

As mentioned before, we train our model using conditional MCMC sampling. This means that given an image, the label state initially starts as random noise, but through each step of the MCMC process, it gets gradually altered until the final label state looks as close as possible to the true label of the image. We update each step using the langevin equation

\[ y_{t+1} = y_t - \frac{\epsilon^2}{2} \nabla_{y_t} U(y_t|x; \theta) + \epsilon Z_t \tag{10} \]

where \( y_t \) is the old label state, \( y_{t+1} \) is the new label state, and \( Z_t \) is a normal variable. It’s also important to note that based on (10), we send into the network \( U \) both the label state and the image it’s conditional to.

Looking at Figure 1, you can see that given an image, the model forms an energy landscape in the logit space. In this case, a synthesized label vector is uniformly initialized in the space. As the synthesized label vector is gradually refined in the MCMC process, the logit values change and the vector travels downhill to the correct logit space where the energy is lowest. In this case, the index with the highest logit value corresponds to the class it predicts and its location in the energy landscape. Figure 2 is a closer view of how the logit values get altered throughout the MCMC process. In this case, the index that predicts the correct class will have a higher logit value whereas the other decreases as it gets updated in each step of MCMC sampling.

3.3 Adversarial Attacks

One type of adversarial attack we use to test the defense of the EBM is the white-box attack. In this case, the attacker has access to the model’s parameters. This makes it stronger compared to a black-box attack, where it wouldn’t have access to it. One white-box attack we adapted from was the PGD attack. As mentioned before, the PGD attack allows the attacker to create an adversarial image that will fool the model into coming out with a wrong output. PGD tries to find the perturbation that maximizes the loss of a model given its adversarial image while making sure that the perturbation is small enough that the adversarial image doesn’t end up looking completely different from the original image.

In our case, we took the PGD equation mentioned in [6] and changed it to account for the labels.
Looking at (11), $x_t$ is the old adversarial image and $x_{t+1}$ is the new adversarial image. In this case, because we focus on the labels conditional on the image, the attacker would send to the network $U$ the adversarial image $x_t$ and the true label $y_{true}$ that corresponds to the image the attacker perturbs. In this case, the attacker would be trying to lift just the minimal point in the energy surface where the true label is located. This would be considered a weak attack. See the middle graph in Figure 3 for reference.

We also implemented a stronger second PGD attack

$$x_{t+1} = \Pi_{x_{nat} + S} (x_t + \alpha \nabla_x U(y_{true} | x_t))$$  

(12)

where the attacker tries to raise, not only the minimal point in the energy surface, but also some of the area around it as well. If they were able to do this, it would make it harder for the label to be located in the correct class area because the energy would be too high in relation to the rest of the area. During label synthesis, this would cause the label vectors to not travel downhill to the correct region. See the bottom graph in Figure 3 for reference.

4 Results

We compared the two attacks using the CIFAR-10 dataset and saw how well the model could defend against it compared to the model’s natural accuracy before the attacks occurred. The natural accuracy of the model was between 79 to 80 percent. When defending against the weaker attack, as the adversarial image got stronger, the model obtained a final accuracy of about 48 percent. See Figure 4 for reference.

When defending against the stronger attack, the model had a harder time of defending and was able to obtain a lower robust accuracy of about 45 percent with only about a 3 percent difference compared to the defense against the weaker attack. As we expected, the model had an easier time defending against the weaker attack, but fortunately, there wasn’t a huge difference between the two, meaning a stronger attack won’t have a huge change on its robust accuracy. See Figure 5 for reference.

Looking at Figure 6, we compared the defense of the EBM model against a standard classifier. In this case, we focused on the strongest attack for the EBM model and the standard PGD attack for the normal classifier. The only difference in network architecture between the two is that the normal classifier excludes the labels in the process and outputs probabilities for all the classes rather than outputting a scalar value. Although the normal classifier had a higher natural accuracy, the EBM still maintained a good accuracy while the classifier broke during the attack.

The most important thing to note here is that no matter what, the model was able to achieve a good robustness against adversarial attacks, meaning the attacker had a more difficult time altering the energy landscape itself.
Figure 6: Comparison between the robustness of a standard classifier defending against a PGD attack and our EBM model defending against our strongest attack.

compared to manipulating a classifier to have wrong prediction probabilities after a forward pass. By making it difficult for the attacker to alter the energy landscape, the EBM was still able to classify almost half of the images correctly with the use of conditional MCMC sampling. In doing so, it created a more robust decision boundary that made it harder for adversarial images to be misclassified.

5 Future Work

In the future, we plan on adding to it by training the model to also learn the probability distribution of $p(x)$ where it focuses on the images themselves and not the labels. By involving the process of learning the unconditional energy landscape of the images as well as the energy landscape with regards to $p(y|x)$, it will be able to defend better. In this case, we would be learning the joint distribution $p(x, y)$. This would mean that during the attack, the model will be able to purify the adversarial image and then classify it while only utilizing energy landscapes.

6 Conclusion

Although 65 percent of robust accuracy is considered state-of-the-art results, these state-of-the-art results are trained against specific attacks while ours is only trained using natural images. This makes it clear that our method does provide some natural robustness against adversarial attacks. By training it to create energy landscapes that are close to the true energy landscapes of the dataset, it was able to make it more difficult for the attacker to manipulate the energy surface directly. Therefore, the EBM model’s decision boundary, made it more difficult for the adversarial images to be on the wrong side of the boundary and get misclassified. Although there have been previous works that focused on EBMs, ours was a new method that didn’t utilize cross-entropy loss and still provided a new way that offered both high natural accuracy as well as robust accuracy.

References


