Improving MagNet with Different Autoencoder Architectures

Wenlong Xiong
204407085
wenlongx@gmail.com

1. Introduction

In the last few years, deep learning models have been shown to have great performance on a variety of datasets from almost every field, and have become a core part of many machine learning pipelines. However, neural networks have also been shown to be vulnerable to adversarial examples[1][2]; these are examples that are visually indistinguishable from legitimate examples, but have minor perturbations that cause the model to have a high rate of error on them. As neural networks are used in more security or safety-focused applications, it is increasingly important that they are robust against these adversarial attacks.

There are many existing methods both to generate adversarial examples to attack models, as well as to defend against such attacks. One particular defense method, MagNet[3] has been shown to be successful in defending against both black-box as well as "grey-box" adversarial attacks. MagNet defends against adversarial attacks by utilizing a *detector* classifier that preemptively detects and discards adversarial examples, and also a *reformer* autoencoder that projects examples onto a latent manifold to destroy the adversarial perturbations that cause the model to misperform while retaining features.

I aim to build upon MagNet's success and improve the *reformer* for my project. In the original MagNet paper, the autoencoder architectures used in the *reformer* were vanilla convolutional autoencoders for the MNIST dataset; for the CIFAR-10 dataset, it was mentioned that the autoencoder was trained as a denoising autoencoder, but no further comparison was given between the performance of the vanilla and denoising autoencoders. My project aims to improve MagNet by utilizing various types of autoencoders that result in more robust latent encodings than vanilla autoencoders. I also give a quantitative comparison between the performance of the encoders tested, which include generic convolutional autoencoders, denoising autoencoders, stacked denoising autoencoders, and contractive autoencoders. This includes both performance on black-box and "grey-box" attacks. We demonstrate that stacked denoising autoencoders have better performance than denoising autoencoders or contractive autoencoders, in both black box and grey box settings.

2. Background and Related Work

2.1. Black Box vs White Box attacks

The basic idea behind an adversarial attack is to perturb an input example to a network by some small delta and generate an adversarial example, such that there is a large change in the output of the loss or objective function of the network. This causes the input to be visually similar to other legitimate examples, but have a vastly different classification or prediction than those legitimate examples.

The methods used to find these deltas or perturbations can generally be broken down into two major categories: white box attacks and black box attacks. In white box attacks, the attack method has detailed knowledge about the network, including the architecture, parameters, weights, and all the outputs, and can use all of these to calculate how to perturb an image. However, in black box attacks, the adversary's knowledge is restricted to the input and output of the model, and the adversary cannot observe the architecture or parameters of the network.

Common white box attack methods include gradient-based methods, which use the gradient of the objective function to calculate the perturbation (such

as the fast gradient sign method[1]); Jacobian-based methods (such as the Jacobian based Saliency Map attack[4]); and iterative methods that take multiple perturbation steps (like I-FGSM[5] or MI-FGSM[6]). Common black box attack methods include Carlini & Wagner's attack[7], Zeroth Order Optimization[8], and learning substitute models so that white-box attack methods can be applied[2].

2.2. MagNet Defense

There are many defense methods against white-box attacks, which include adversarial training (training the model with adversarial examples mixed in with the legitimate ones), blocking gradients to prevent against gradient-based attacks (defensive distillation[9]), etc. However, we examine one in particular, called MagNet.

MagNet is a model architecture-agnostic defense method, which is composed of 2 parts. It passes inputs through 2 additional networks, a *detector* that detects if an input example is adversarial or not, and a *reformer* to reform the example and destroy the adversarial perturbations present. The structure of MagNet is shown below in Figure 1, where the classifier shown can be any arbitrary model.

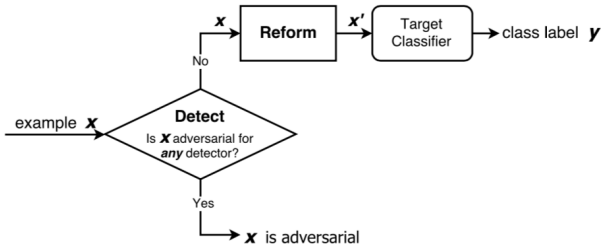

Figure 1: MagNet workflow, from the original MagNet paper[3].

The *detector* acts as a sieve to preemptively detect and discard adversarial examples if it finds that they are sufficiently different from real ones, while the *reformer* attempts to eliminate the perturbations that cause an example to be adversarial by passing examples through an autoencoder.

2.3. Autoencoders

The reason MagNet uses autoencoders for its *reformer* network is due to the manifold hypothesis. According to the manifold hypothesis, high dimensional natural data actually lies on a lower-dimensional manifold that is embedded in the higher dimensional space[10]. Theoretically, this lower-dimensional manifold that all the natural data lies on can be represented by a latent representation of dimension L, where L is less then the data dimension D. The latent representation can be intepreted as a coordinate system for points on the manifold. By finding an encoding transformation $f : R^D \rightarrow R^L$ that maps from the data dimension to the latent dimension, and the inverse decoding transformation $g : R^L \rightarrow R^D$ that reconstructs the data point from the encoded latent representation, we can project an example onto this manifold and back into the high dimensional space again.

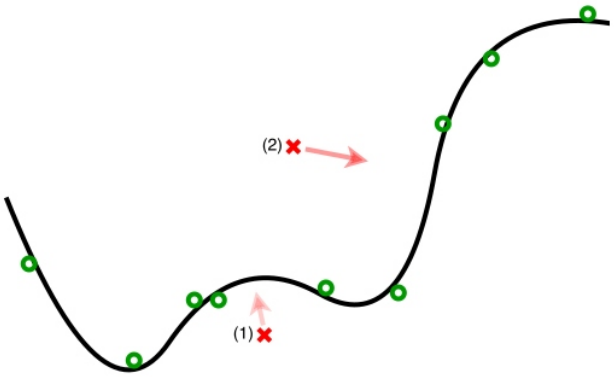

Figure 2: Visualization of lower dimensional manifold. Green points are legitimate examples near the manifold, and red X's are adversarial examples that lie off the manifold. Autoencoders push adversarial examples towards the manifold. Image from original MagNet paper[3].

Santhanam and Grnarova [11] postulate and empirically show that adversarial examples tend to lie off the manifold of real examples. Because adversarial examples are created from legitimate ones by adding a small perturbation, the adversarial perturbation can be seen as moving legitimate examples slightly off the low dimensional manifold. Therefore, autoencoders are a natural choice for the *reformer* network, since by projecting the input adversarial example onto the man-

ifold and reconstructing the example from this projection, the autoencoder is finding an example on the manifold that approximates the uncorrupted original example.

To improve the MagNet *reformer* network, we explore several different types of autoencoders that we postulate will more robustly remove these adversarial perturbations.

2.3.1. Convolutional Autoencoder

In the original MagNet paper, the autoencoder architectures used in the *reformer* network were vanilla convolutional autoencoders for MNIST. Convolutional autoencoders are autoencoders that utilize convolutional layers in their encoder network ($f : R^D \rightarrow R^L$), and transposed convolutional layers to upsample in the decoder network ($g : R^L \rightarrow R^D$). Using convolutional layers instead of fully connected layers allow us to reduce the number of parameters while maintaining expressiveness. To train the convolutional autoencoder, we minimize the squared error between the reconstructed output of the autoencoder ($\hat{X} = g(f(X))$) and the input (X).

$$Loss = \|X - \hat{X}\|_2^2$$

We test convolutional autoencoder performance to form a baseline to compare other types of autoencoders against.

2.4. Denoising Autoencoder

A denoising autoencoder takes the same architecture as a generic convolutional autoencoder, but makes the manifold projection during the encoding process an explicit part of the autoencoder training. During the training process, we corrupt the input with noise, and used the corrupted examples as input to our denoising autoencoder. In our case, we use additive isotropic Gaussian noise as described by Vincent, et al [10].

$$\begin{aligned} \tilde{X} &\sim \mathcal{N}(X, \sigma^2 I) \\ \hat{X} &= g(f(\tilde{X})) \\ Loss &= \|X - \hat{X}\|_2^2 \end{aligned}$$

By explicitly adding noise to examples during training, the denoising autoencoder learns features that are robust to noise. This means that the encoding/decoding function attempt to project the corrupted input to the nearest features, instead of simply learning the noise in an identity mapping. In comparison to the generic autoencoder, the denoising autoencoder has a lower chance of learning a mapping that preserves small noise perturbations; in our case, has a lower chance of learning a mapping that retains the adversarial perturbations after the reconstruction.

Vincent (2010) [10] describes an extension to denoising autoencoders; by stacking several denoising autoencoders and training each autoencoder with noisy inputs, it is possible to obtain higher level features. After a single autoencoder in a stack has been pretrained, we use the output from the original non-noisy inputs as the non-noisy input to the following autoencoder.

2.5. Contractive Autoencoder

Rifai et al [12] described another type of autoencoder that results in robust feature extraction, called a contractive autoencoder. The contractive autoencoder adds a penalty term to the loss function, which consists of the frobenius norm of the Jacobian of the latent representation with respect to the input. This is shown in the equation below:

$$\begin{aligned} \hat{X} &= g(f(X)) \\ Loss &= \|X - \hat{X}\|_2^2 + \lambda \|J_{f(X)}(X)\|_F^2 \end{aligned}$$

By penalizing the Jacobian of the latent representation, these autoencoders add a contractive penalty to the encoding function ($f(X)$). This means that inputs with small perturbations will result in very similar latent mappings (encouraging the encoding function to be robust to noise). In comparison, Rifai showed that denoising autoencoders are equivalent to a contractive penalty to the reconstruction / decoder function ($g(f(X))$), which makes the decoder function resist small changes to the input.

Empiricially, Rifai showed that contractive autoencoders have better performance in robust feature extraction compared to denoising autoencoders on certain datasets (including MNIST and CIFAR-10). In our case, this means that contractive autoencoders may result in an encoding scheme that is resistant to the perturbations in adversarial examples.

3. Implementation

To quantitatively compare the performance of these different types of autoencoders in MagNet, I trained autoencoders of each type to use as the *reformer* network, and attacked the reformer and classifier under grey-box and white-box settings for the MNIST dataset.

3.1. Network Architectures

3.1.1. Classifier

The classifier used in all our experiments was a simple convolutional network similar to LeNet5. It has two (5, 5) convolutional layers with 32 filters each, each followed by a ReLU activation and a (2, 2) max pooling layer. Padding is added in the convolutional layers such that the input and output of each convolution is the same shape. Afterwards, the classifier has a fully connected layer with 1024 units, followed by the logits layer. The classifier was trained with categorical cross entropy loss.

3.1.2. Vanilla and Denoising Autoencoder

The vanilla convolutional autoencoder architecture consists of an implicitly defined encoder and decoder network. The encoder consists of two (3, 3) convolutional layers with 32 filters each, each followed by a ReLU activation and a (2, 2) max pooling layer. Padding is added in the convolutional layers such that the input and output of each convolution is the same shape. Given an input shape of (28, 28, 1), this encoder network results in a latent representation of shape (7, 7, 1). The decoder consists of two (3, 3) transposed convolutional layers with stride 2. The first has 32 filters and uses the ReLU activation, and the second has 1 filter and uses the sigmoid activation.

The denoising autoencoder shared the same architecture as the vanilla autoencoder. The only difference between them was during training time; the vanilla autoencoder was trained on uncorrupted inputs, while the denoising autoencoders' training data was corrupted with gaussian noise with a variance of σ^2, for various values ("noise volume").

3.1.3. Stacked Denoising Autoencoder

The stacked denoising autoencoder consists of multiple denoising autoencoders chained together. The output of one denoising autoencoder is used as the input of the next one, and each denoising autoencoder is pretrained individually.

3.1.4. Contractive Autoencoder

The vanilla convolutional autoencoder architecture consists of a similar architecture to the vanilla convolutional autoencoder, but adds a dense layer between the encoder and decoder networks for the latent representation. After the second max pooling layer, the contractive autoencoder has a fully connected layer with 1568 units and a sigmoid activation.

3.2. Adversarial Attacks

The adversarial attacks on the networks were performed under both grey box and white box settings. Under the grey box setting, it is assumed that the adversary can only view the weights of the classifier, and not the reformer network. In this case, the adversarial examples are generated by only attacking the classifier. In the white box setting, the adversary can view the weights of both the reformer network and classifier, so we treat the two of them as a single network which we generate adversarial examples from. The adversarial example generation was performed using the Fast Gradient Sign Method, from the cleverhans library[13]. We used an epsilon of 0.3 for the FSGM attack.

3.3. Training

Each of the classifiers and autoencoders were trained using the Adam optimizer, using a learning rate of 0.001, β_1 of 0.9, β_2 of 0.999, and ϵ of $1e-07$. The classifier was trained for a total of 30 epochs, until convergence. Each of the autoencoders were each trained for 20 epochs. In the case of the stacked denoising autoencoder, each individual autoencoder was pretrained for 10 epochs, then the entire stack was trained for an additional 20 epochs.

For the denoising autoencoder, we varied the volume of gaussian noise added. We tested the values $\sigma^2 = [0.1, 0.2, 0.3, 0.4, 0.5]$. For the contractive autoencoder, we tested different levels of regularization strength: $\lambda = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1]$.

Classifier		(Denoising) Autoencoder		Contractive Autoencoder	
Conv.ReLU	$5 \times 5 \times 32$	Conv.ReLU	$3 \times 3 \times 32$	Conv.ReLU	$3 \times 3 \times 32$
MaxPool	2×2	MaxPool	2×2	MaxPool	2×2
Conv.ReLU	$5 \times 5 \times 32$	Conv.ReLU	$3 \times 3 \times 32$	Conv.ReLU	$3 \times 3 \times 32$
MaxPool	2×2	MaxPool	2×2	MaxPool	2×2
Dense.Relu	1024	ConvTranspose.ReLU	$3 \times 3 \times 32$	Dense.Relu	1568
Dense.Softmax	10	ConvTranspose.Sigmoid	$3 \times 3 \times 1$	ConvTranspose.ReLU	$3 \times 3 \times 32$
				ConvTranspose.Sigmoid	$3 \times 3 \times 1$

Table 1: Classifier and autoencoder architectures. The vanilla convolutional autoencoder and denoising autoencoder share the same architecture; the denoising autoencoder is trained with noisy inputs instead. The stacked denoising autoencoder architecture consists of multiple stacks of denoising autoencoders, where the output of one denoising autoencoder is fed directly as the input of the next autoencoder in the stack. The contractive autoencoder has an explicit latent / encoded layer.

σ^2	0.1	0.2	0.3	0.4	0.5
	Grey Box Adversarial Accuracy				
Denoising Autoencoder	11.35%	11.35%	75.15%	**77.17**%	77.03%
Stacked Denoising Autoencoder (2)	12.70%	11.11%	10.01%	80.85%	**81.25**%
Stacked Denoising Autoencoder (3)	11.04%	09.90%	9.72%	**83.59**%	82.93%
	White Box Adversarial Accuracy				
Denoising Autoencoder	11.35%	11.35%	**77.66**%	76.09%	73.25%
Stacked Denoising Autoencoder (2)	11.35%	8.62%	7.46%	**78.46**%	72.73%
Stacked Denoising Autoencoder (3)	11.88%	8.51%	7.93%	**77.06**%	68.97%
	Test Accuracy				
Denoising Autoencoder	11.35%	11.35%	**98.78**%	98.46%	98.15%
Stacked Denoising Autoencoder (2)	13.44%	13.75%	9.56%	**98.14**%	97.54%
Stacked Denoising Autoencoder (3)	8.96%	6.55%	8.27%	**97.65**%	96.65%

Table 2: Test accuracies of classifier using denoising autoencoders as the *reformer* network, for both the grey box attack, white box attack, and non-adversarial case. Denoising autoencoder were trained with varying levels of gaussian noise.

4. Results

The basic classifier trained on the MNIST dataset achieved a 99.16% classification accuracy on legitimate examples, after just 30 epochs of training. However, when an adversarial attack was applied, the classification accuracy on the adversarial examples generated from the FGSM dropped to 11.42%. By using autoencoders as the *reformer* network however, we succeed in increasing the accuracy on adversarial examples while only slightly reducing the test accuracy on legitimate examples.

In Table 2, we see that both denoising autoencoders and stacked denoising autoencoders greatly improve accuracy on adversarial examples, increasing the test accuracy from 11.42% to approximately 80%, both in the white box and grey box settings. Stacked denoising autoencoders tend to have better grey box adversarial accuracy than a single denoising autoencoder (up to 6.42% improvement from the best denoising autoencoder trained). However, when we generate adversarial examples in the white box setting, this improvement disappears, and a stack of several denoising autoencoders do not perform drastically differently from a single one. We also noticed that the test accuracy on legitimate examples decreased slightly when we had a *reformer* network, similar to what is described in MagNet. However, this decrease is generally a neg-

λ	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
	Grey Box Adversarial Accuracy				
Contractive Autoencoder	64.38%	63.39%	**67.40%**	60.39%	62.07%
	White Box Adversarial Accuracy				
Contractive Autoencoder	60.15%	60.46%	**63.62%**	60.42%	43.73%
	Test Accuracy				
Contractive Autoencoder	**98.91%**	98.84%	98.51%	98.30%	97.84%

Table 3: Test accuracies of classifier using contractive autoencoders as the *reformer* network, for both the grey box attack, white box attack, and non-adversarial case. Contractive autoencoder were trained with varying weights for the contractive penalty.

	Test Acc.	Grey Box Adv. Acc.	White Box Adv. Acc.
No Preprocessing	99.16%	–	11.42%
Vanilla Autoencoder	98.97%	32.93%	48.29%
Denoising Autoencoder ($\sigma^2 = 0.3$)	98.78%	75.15%	77.66%
Denoising Autoencoder ($\sigma^2 = 0.4$)	98.46%	77.17%	76.09%
(2) Stacked DAE ($\sigma^2 = 0.4$)	98.14%	80.85%	**78.46%**
(3) Stacked DAE ($\sigma^2 = 0.4$)	97.65%	**83.59%**	77.06%
Contractive Autoencoder ($\lambda = 10^{-3}$)	98.51%	67.40%	63.62%

Table 4: Comparison of *reformer* network autoencoder choices, for best hyperparameters found. Accuracy of legitimate and adversarial examples on the classifier with no *reformer*, and the classifier with the vanilla autoencoder as the *reformer* are shown as baselines.

ligible $< 1\%$ decrease. In addition, for lower volumes of noise ($\sigma^2 = 0.1, 0.2$), we observe that the test accuracy for both legitimate and adversarial examples are about 10%, meaning that the predictions made are no better than a random guess. This means that for lower volumes of noise, the denoising autoencoders are not training and converging, and the resulting autoencoders destroy the any signal in the input.

In Table 3, we see that using a contractive autoencoder in the *reformer* network also results in an increase in adversarial test accuracy, by more than 50%. We see similar results as with the denoising autoencoders; the accuracy on adversarial examples in the white box setting is slightly lower than that in the grey box setting. Overall, the contractive autoencoder does not improve adversarial results as much as the denoising autoencoders with similar architectures do; contractive autoencoders only raise adversarial accuracies to about 65%, while denoising autoencoders obtain adversarial accuracies averaging $\sim 10\%$ higher.

In comparison to vanilla autoencoders, both contractive and denoising autoencoders result in higher classification accuracy on adversarial examples. Vanilla autoencoders only prevent white box and black box attacks fewer than 50% of the time.

5. Discussion and Future work

After selecting the best performing autoencoder models, we showed a comparison of their performance to the baseline vanilla autoencoder + classifier and no preprocessing + classifier networks, in Table 4. Overall, the stacked denoising autoencoders obtained the best results; for the grey box setting, a stack of three denoising autoencoders obtained a 83.59% test accuracy on the adversarial examples, and for the white box setting a stack of two denoising autoencoders obtained a 78.46% accuracy. However, these settings resulted in a decrease in test accuracy on legitimate examples by 1.51% and 1.02%, respectively. Deeper stacks of denoising autoencoders result in incrementally better adversarial accuracy than a single autoencoder for the grey box setting, at the expense of accuracy on legitimate examples. In the white box setting, the adver-

sarial performance does not improve by as large an amount. These results show a small but noticeable improvement over the results of MagNet; MagNet assumes that denoising autoencoders or vanilla autoencoders are used; stacked denoising autoencoders show a 6.42% and 0.8% increase over the best denoising autoencoder models in the grey box and white box settings, respectively.

On the other hand, contractive autoencoders have worse performance than denoising autoencoders. The initial motivation behind testing them was that contractive autoencoders have been shown to learn feature extractors that are robust to input noise. We thought that this contractive property would help the autoencoder learn a feature extractor that reduces the perturbations that cause examples to be adversarial, by pushing them towards the latent manifold. However, empirically we see that even if the contractive autoencoders were able to reduce perturbations, the resulting reconstruction still contains enough adversarial information to cause the classifier to misclassify at a high rate. It is possible that with adversarial training, this contractive autoencoder could learn an encoding function that specifically contracts the space in the dimensions of adversarial perturbations. However, more experiments are needed to verify this.

In addition, denoising autoencoders have been shown to be equivalent to putting a contractive penalty on the decoder / reconstruction function in a vanilla autoencoder, while contractive autoencoders put the contractive penalty on the encoder function. In our results, we have seen that the denoising autoencoder results in higher test accuracies than the contractive autoencoder. It is worth exploring why penalizing the reconstruction function over the encoding function is more effective, and if putting a contractive penalty on a denoising autoencoder will help performance as well.

We also observe that the denoising autoencoders show higher accuracy on adversarial examples when the volume of noise used in training is lower, but also do not converge when the volume of noise is too low. This could be an indication of a learning rate that is too large or small, so future work could involve rerunning this code with a different learning rate so that the denoising autoencoders with smaller σ^2 converge.

6. Conclusion

I built on MagNet's success in defending against adversarial attacks by modifying its *reformer* network, and trying various types of autoencoders to quantitatively compare their performance in reducing adversarial classification error. I compared vanilla convolutional autoencoders, denoising autoencoders, stacked denoising autoencoders, as well as contractive autoencoders. I demonstrated that stacked denoising autoencoders empirically improve classification accuracy on adversarial examples generated with FGSM, as compared the denoising autoencoder architectures described in MagNet. In addition, I showed that while contractive autoencoders have been shown to result in a robust feature extraction, they are not very effective at reducing the adversarial perturbations that cause classifiers to misclassify. As far as I know, this is the first time contractive autoencoders have been evaluated as a possible defense against adversarial attacks.

References

[1] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[2] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey, 2018.

[3] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples, 2017.

[4] Rey Wiyatno and Anqi Xu. Maximal jacobian-based saliency map attack, 2018.

[5] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2016.

[6] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum, 2017.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2016.

[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. 2017.

[9] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks, 2015.

[10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[11] Gokula Krishnan Santhanam and Paulina Grnarova. Defending against adversarial attacks by leveraging an entire gan, 2018.

[12] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 833–840, USA, 2011. Omnipress.

[13] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, Rujun Long, and Patrick McDaniel. Technical report on the cleverhans v2.1.0 adversarial examples library, 2016.