Semester Project – Automn 2023

Between decideable logics: ω-automata and infinite games

With 31 Illustrations

Author
Diego Dorn

Supervisor
Clément Hongler

Remaining to be done

Introduction

Artificial neural networks are famously vulnerable to adversarial attacks [Szegedy et al., 2013, Goodfellow et al., 2014, Chen et al., 2021]. Check the papers
- defense, autoencoders
- universal, transferable attacks

Contents

Conventions

Throughout this document we adopt a set of conventions and notations.

- We use $A \subset B$ to say A is included, not strictly in B and $A \subsetneq B$ if this inclusion is strict.
- $\mathcal{X} \subset \mathbb{R}^n$ is the set in which datapoints live.
- $\mathcal{Y}$ is a space of labels, which will most of the time be categorical, i.e. $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{\text{cat}, \text{dog}, \text{boat}\}$.
- We use $\mathcal{D}$ for datasets. For unlabelled datasets, $\mathcal{D} \subset \mathcal{X}$. For labelled datasets, $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$.
- Loss functions are denoted by $\mathcal{L}$.
- We write $\|\cdot\|$ for the euclidean norm on $\mathbb{R}^n$, and $\|\cdot\|_p$ for the p-norm on $\mathbb{R}^n$. As a reminder, for $x \in \mathbb{R}^n$, $\|x\|_\infty = \max_{i=1}^n |x_i|$.

1 Attacking classifiers

The common knowledge is that neural networks are vulnerable to adversarial attacks and adversarial attacks are easy to find [Szegedy et al., 2013, Goodfellow et al., 2014, Chen et al., 2021]. The first thing I wanted to do was to verify this in practice. Can I easily attack any classifier outside of the well defined confines of a classroom or a paper for which a lot of work was put into?

Add motivation for why I choose this (philo, impact)

1.1 Setup

I used three different models and datasets throughout this project. The code of every experiment is available at `https://github.com/ddorn/autoencoder-attacks`.

MNIST The smallest dataset I used is the MNIST dataset [LeCun et al., 1998], with a small convolutional classifier acheiving 98.8 % accuracy implemented in pytorch [Oikarinen, 2021].0

CIFAR-10 The second dataset is the CIFAR-10 dataset [Krizhevsky, 2009], with a convolutional classifier acheiving 92.8% accuracy [Germer, 2022].

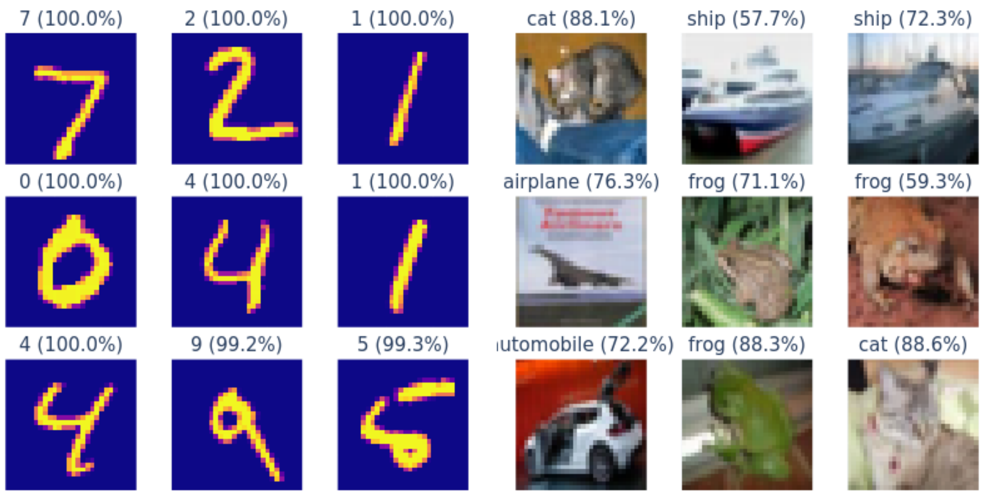

Figure 1: The first 9 test images of the MNIST dataset (left) the CIFAR-10 dataset (right). The confidence of the classifier is shown in parenthesis.

ImageNet The third dataset is ImageNet [Deng et al., 2009], with a ResNet-50 classifier achieving 77 % accuracy [He et al., 2015].

1.2 Fast gradient sign method

The simplest attack is the fast gradient sign method (FGSM) [Goodfellow et al., 2014]. This attack requires only one forward and one backward pass through the network to

Figure 2: The first 9 test images of the ImageNet dataset. The confidence of the classifier is shown in parenthesis.

find a small perturbation of an image that can (potentially) fool the classifier.

What is small? We usually constrain the norm of the perturbation to a small value ε. The norm used can be the l_∞ norm, for $\varepsilon = 10/255$ or $\varepsilon = 4/255$ are common values, or the l_0, l_1 or l_2 norm. Clearly the four norms produce different constrains, and should be chosen depending on the context:

- l_∞ is a natural choice, and corresponds to changing each pixel value by at most ε.
- Using the l_0 norm means to change at most ε pixels. This can be one-pixel attacks [Su et al., 2017], attacks that change a small number of pixels, or patch attacks [Brown et al., 2017].
- l_1 and l_2 constraints can be used when it is fine if some pixels are completely changed, but not too many are changed a lot.

Definition 1.1. Let f be a classifier and $x \in \mathcal{X}$ an input. The **fast gradient sign method** is the attack that computes

$$x_{\text{FGSM}} = x + \varepsilon \cdot \text{Sign}(\nabla_x \mathcal{L}(f(x), y))$$

where $\mathcal{L}$ is the loss function used to train f, and ε is the desired l_∞ norm of the perturbation.

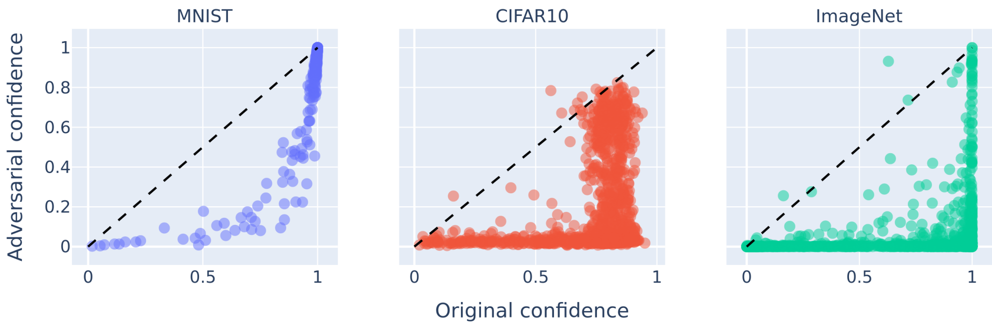

Figure 3: Confidence of the three classifiers in the correct label of a 1000 test images before and after an FGSM attack with $\varepsilon = 10/255$. The black line corresponds to no change in confidence.

1.3 Iterated projected gradient descent

2 Attacking autoencoders

2.1 Autoencoders

Autoencoders where introduced by [Hinton and Salakhutdinov, 2006] as a way to learn a low dimensional representation of data. They are a class of neural networks that are trained to reconstruct their input, and are composed of two deep neural network, an encoder and a decoder with a bottleneck in between as shown in Figure 4.

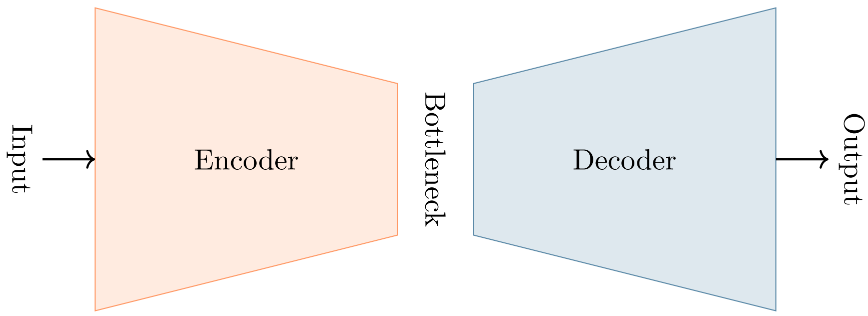

Figure 4: Overview of the autoencoder architecture

The **encoder** takes an high dimensional data point as input, processes it through a series of layers, usually fully connected layers or a residual network [He et al., 2015] in the case of visual data, and outputs a low dimensional representation of the input.

The **decoder** takes the low dimensional output of the encoder and processes it similarly through a series of layers, and outputs a high dimensional reconstruction of the input.

The **bottleneck** is not a layer, but rather the middle of the autoencoder, where the activations are the lowest number of dimensions.

Training Autoencoders are trained to reconstruct their input, that is, they learn the identity function. The loss is a natural metric on the data space, such as the mean squared error for real valued data.

Definition 2.1. The **reconstruction loss** for an autoencoder f on an input $x \in \mathcal{X}$ is

$$\mathcal{L}_{\text{recon.}}(x) = \|x - f(x)\|^2$$

To prevent overfitting and to perform a directly useful task, an autoencoder can be train to reconstruct a noisy or corrupted version of the input.

Definition 2.2. The **denoising loss** for an autoencoder f on an input $x \in \mathcal{X}$ is

$$\mathcal{L}_{\text{denoising}}(x) = \|x - f(x + \varepsilon)\|^2$$

where ε is a random vector of the same dimension as x, of white noise whose variance is an hyperparmeter of the training setup.

Note that the denoising loss is stochastic, as it depends on the random vector ε. In practice, we use the compute the loss on one sample of ε per input.

Variational autoencoders An specific kind of autoencoders intruduced by are variational autoencoders (VAE). Technically, they are not very different from regular autoencoders, but they come from a different background than data compression. Indeed, the hope is that VAEs model the process from which the data was generated. Oftentimes, we expect a datapoint (for instance the image of a leaf) to be determined only by *a few* variables (for instance, the species of the tree, its age, the season, the angle at which the picture was taken etc.). We will call P, the vector of those few variables that generate the datapoint.

ref intro VAE

The encoder of a VAE tries to find some representation of P and outputs two vectors, μ and σ instead of one, which are interpreted as the mean and the variance of the prior on P, which is assumed to be a normal distribution.

The variable $P \sim \mathcal{N}(\mu, \sigma)$ are then sampled and fed to the decoder that tried to reconstruct what should be generated from the underlying variables. The decoder thus tries to model the process that generated the dataset and outputs a distribution Q over the data space.

Definition 2.3. Let f be a VAE and $x \in \mathcal{X}$ a datapoint. It loss on x is composed of two terms, the **likelyhood loss** and the **regularisation loss**.

$$\mathcal{L}_{\text{vae}}(x) = \underbrace{\mathbb{P}(x \mid f(x))}_{\text{likelyhood loss}} + \underbrace{D_{\text{KL}}(P \mid\mid \mathcal{N}(0,1))}_{\text{regularisation loss}}$$

Remark. The KL divergence is a measure of how different two distributions are. In this case, it is used to measure how far the prior on P is from the standard normal distribution.

$$D_{\text{KL}}(P \mid\mid Q) = \int_{\mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} \, \mathrm{d}x$$

Here we can use the fact that both P and Q are n-dimensional normal distributions to compute the KL divergence in closed form.

$$D_{\text{KL}}(\mathcal{N}(\mu,\sigma) \mid\mid \mathcal{N}(0,1)) = \frac{-1}{2n} \sum_{i=1}^{n} \left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2\right)$$

Say why KL is used

β-VAE

2.2 Attacks

Autoencoders can be used to prevent adversarial attacks against classifier by prepocessing images through the autoencoder. The setup is shown in Figure 5.

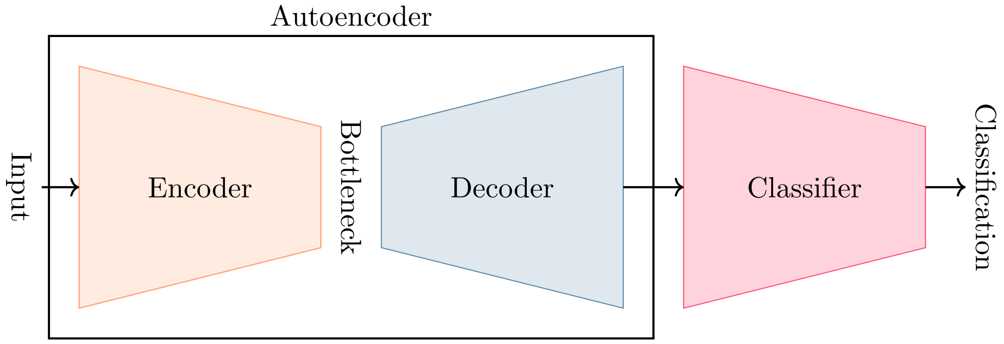

Figure 5: Autoencoder used as a defense against adversarial attacks

The hope is that an adversarial perturbation of the input will not pass through the bottleneck of the autoencoder, and thus will not be able to fool the classifier. Indeed, the bottleneck is small, and therefore information constrained, so we expect to the autoencoder to not faithfully reconstruct patterns that it has never seen during training. In particular, we expect the latent respresentation of an adversarial input to be the same as the representation of the original input, and thus the autoencoder should reconstruct the original input when fed the adversarial one.

We can verify this empirically.

3 Phase transition: ease of attack

4 Phase transition: norm detection

Conclusion

...

References

[Bailey et al., 2023] Bailey, L., Ong, E., Russell, S., and Emmons, S. (2023). Image hijacks: Adversarial images can control generative models at runtime. *ArXiv*, abs/2309.00236.

[Brown et al., 2017] Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. (2017). Adversarial patch. *ArXiv*, abs/1712.09665.

[Chen et al., 2021] Chen, Y., Zhang, M., Li, J., and Kuang, X. (2021). A survey on adversarial attacks and defenses in image classification: A practical perspective. *arXiv preprint arXiv:2201.01809*.

[Chen et al., 2022] Chen, Y., Zhang, M., Li, J., and Kuang, X. (2022). Adversarial attacks and defenses in image classification: A practical perspective. *2022 7th International Conference on Image, Vision and Computing (ICIVC)*, pages 424–430.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[Germer, 2022] Germer, T. (2022). Train cifar10 to 94% accuracy in a few minutes/seconds. Accessed: 2024-01-01.

[Goodfellow et al., 2014] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Higgins et al., 2017] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[Krizhevsky, 2009] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

[Kuzina et al., 2021] Kuzina, A., Welling, M., and Tomczak, J. M. (2021). Diagnosing vulnerability of variational auto-encoders to adversarial attacks.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324.

[Liu et al., 2023a] Liu, H., Li, C., Li, Y., and Lee, Y. J. (2023a). Improved baselines with visual instruction tuning.

[Liu et al., 2023b] Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2023b). Visual instruction tuning. In *NeurIPS*.

[Nguyen et al., 2014] Nguyen, A. M., Yosinski, J., and Clune, J. (2014). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436.

[Oikarinen, 2021] Oikarinen, T. (2021). Training a nn to 99% accuracy on mnist in 0.76 seconds. Accessed: 2024-01-01.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*.

[Shin, 2017] Shin, R. (2017). Jpeg-resistant adversarial images.

[Su et al., 2017] Su, J., Vargas, D. V., and Sakurai, K. (2017). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23:828–841.

[Szegedy et al., 2013] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2013). Intriguing properties of neural networks. *CoRR*, abs/1312.6199.

[Zhang et al., 2021] Zhang, C., Benz, P., Lin, C., Karjauv, A., Wu, J., and Kweon, I. S. (2021). A survey on universal adversarial attack. In *International Joint Conference on Artificial Intelligence*.

[Zou et al., 2023] Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. *ArXiv*, abs/2307.15043.