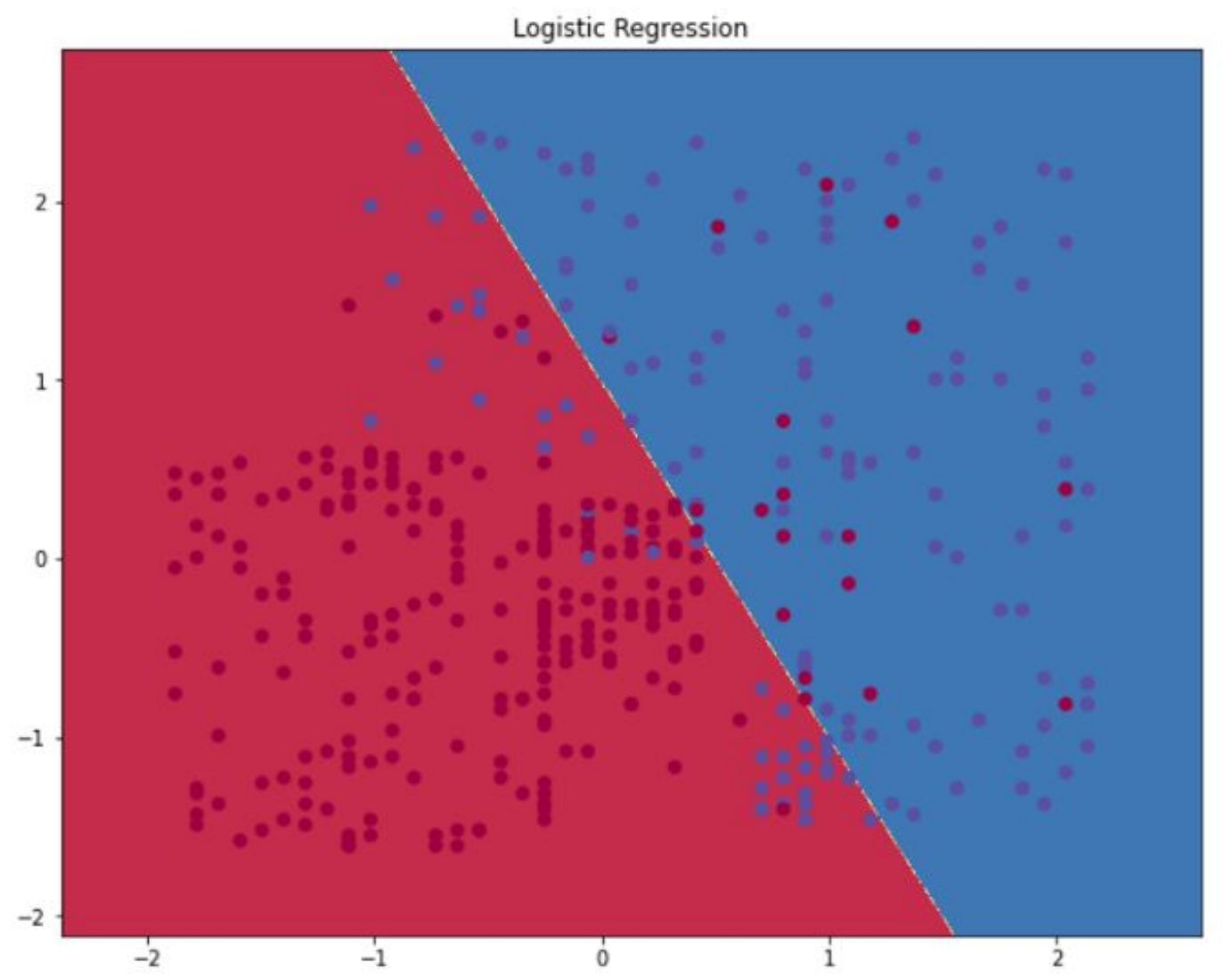
Manipulating Neural Networks

Wayne Leeke, leekew1@nku.edu

How Neural Networks Decide

Neural networks can discover similarities between, and group samples of data together. These decisions are called classifications and are based on commonalities of features in the data. How much a feature contributes to its decided class is called saliency. Observations that share features may belong to the same class, but the classifications themselves may share commonalities with each other. E.g., the classifications “dog” and “wolf” may share several features such as shape and coloration. Between each class there is a decision boundary, where similar classes intersect with each other.



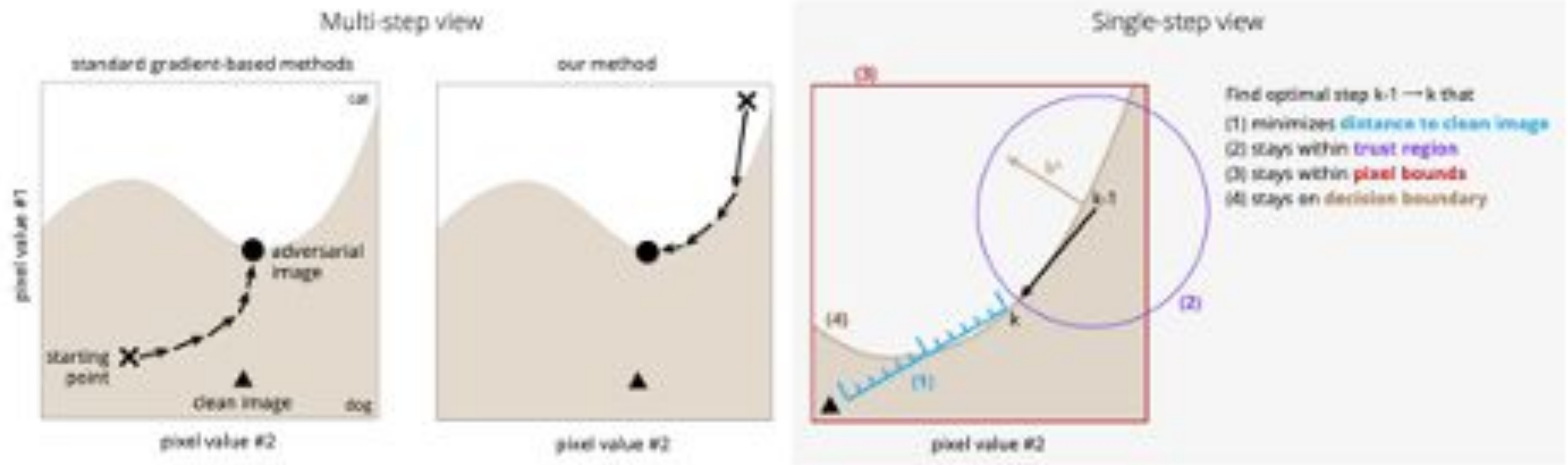
Logistic regression decision boundary between two classes. Decision boundaries in ML models with low dimensionality are easiest to understand. [1]

The decision boundaries for complex models, neural networks with 10s of features, become difficult to visualize. However, by restricting analysis to two features a 2D plane can be constructed. Three classes create a 3D space, and so on. In high dimensional data it’s possible for some classes to not share a boundary. Some more complex models such as neural networks draw can draw non-linear decision boundaries.

Adversarial Samples

Many techniques exist to examine the saliency of a model. Most are incredibly reliable in not only understanding decisions, but also in manipulating features to achieve a different classification. Ideally, an adversarial example should be indistinguishable from normal samples by both humans and machines. This makes detection more difficult. For images, this could be manipulating a limited number of pixels that maximizes saliency for the adjacent classifications. Targeted misclassification is more difficult, because some classes simply do not share a decision boundary.

Most production machine learning products are a Blackbox. Rarely is access to the internals of a network, or probabilistic outputs, available to the end user. This limitation is easily overcome with the use of a poxy model, which stealthily learns the decision boundaries of the production model. This provides a sandbox for white box techniques and evades detection by limiting abnormal traffic. Although, some accuracy is lost in the proxy model not being a 1:1 replication of the production model.


Multi-step view
standard gradient-based methods
cat
adversarial image
starting point
clean image
dog
pixel value #1
pixel value #2
our method
pixel value #2
Single-step view
Find optimal step k-1 → k that
(1) minimizes distance to clean image
(2) stays within trust region
(3) stays within pixel bounds
(4) stays on decision boundary
(1)
(2)
(3)
(4)
k-1
k
pixel value #2

Manipulation of saliency to create a “border attack” using some gradient techniques. [3]

A very successful method of manipulating saliency is the modified “border attack”. The algorithm starts with a normal image. The features are manipulated in small increments until the closest decision boundary is found. A gradient is then constructed around these features and then are minimized. The boundary is walked to its lowest point. This technique allows for an image to still visually appear as its original classification but have just enough change to misclassify.

Motivations of Adversarial Samples:

- Poisoning continuously learning models with misclassifications, lowering confidence in the model
- Misclassification to achieve some otherwise blocked action. E.g., malware detection.
- Avoidance of behavior tracking like facial or gait recognition.

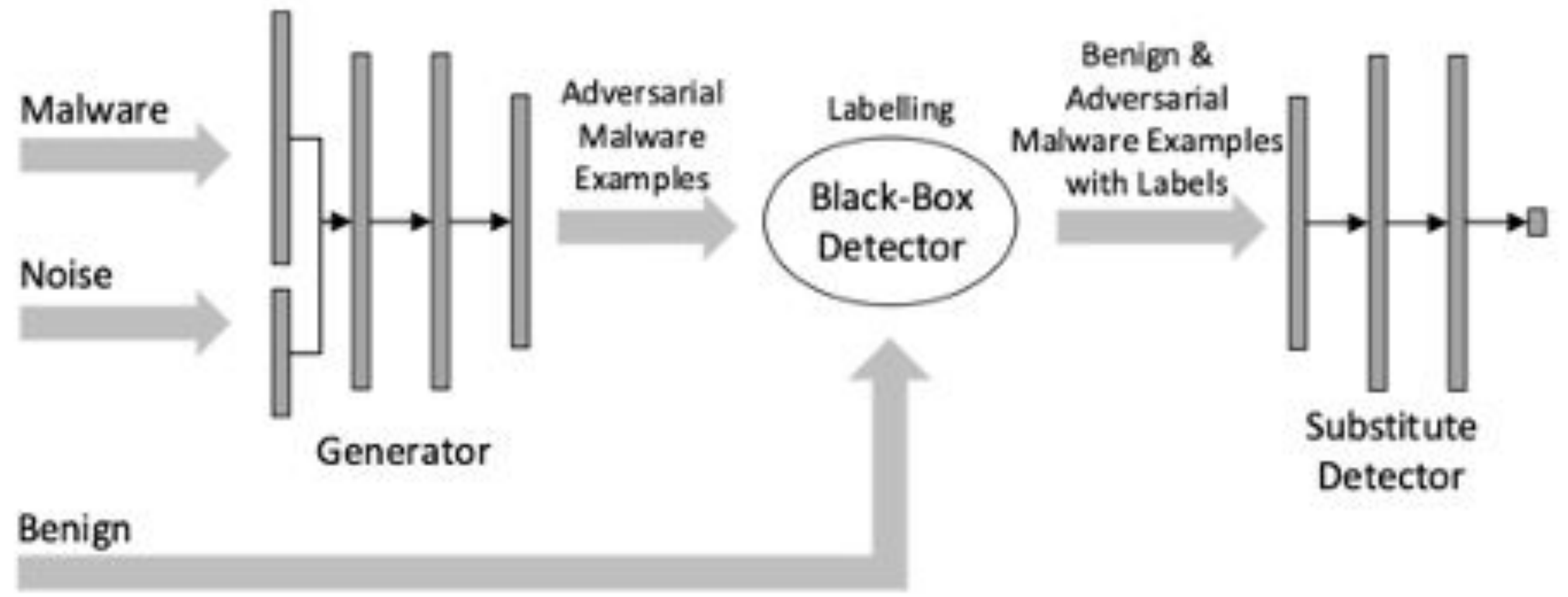
Adversarial Networks

Generative adversarial networks, or GANs, are a way of configuring two models together to play a competitive game. The game is adversarial in nature; the generator model will produce fakes and the discriminator model will have to distinguish these fakes from real examples.

When the discriminator is wrong, it’s trained accordingly based on its error. However, when the discriminator is correct the error is propagated back up to the generator model so it can self-correct. Although connected, the error is always back propagated separately. The goal of this architecture is for the generator to produce examples that the discriminator can no longer separate from real data. The generator isn’t producing anything new. It learns the features of the real data and maps those features to random noise. As the model learns that noise gains meaning and can be used to produce an example of previously learned features combined in a novel way.

Adversarial Model Generated Adversarial Inputs

Previous research has shown that GANs can be used to generate adversarial inputs for binary classification models—models with only two classes. One such algorithm is malgan, which can generate adversarial examples that perform at or better than attack methods like the border-gradient attack.


Malware
Noise
Generator
Adversarial Malware Examples
Labelling
Black-Box Detector
Benign & Adversarial Malware Examples with Labels
Substitute Detector
Benign

The MalGan architecture is comparable to a normal GAN. Instead of having a pool of real examples, the output of the targeted black box model is used to update the detector model. The generator is punished and updated when the detector finds a true positive. [7]

Misclassifications created by malgan are difficult to defend against. A typical mitigation is to retrain the black box classifier with adversarial inputs labeled as malicious. However, “It is a long process to collect a large number of malware samples and label them.” [7] Additionally, “Once the black-box detector is updated, malware authors will attack it immediately by retraining MalGAN and our experiments showed that retraining takes much less time than the first-time training.” [7] Retraining the black box model is no different than updating the discriminator of a GAN. Error will propagate upward correcting causing a correction in the generator. The GAN will then produce increasingly better adversarial examples.

The current MalGan architecture is limited in that it attacks simple, feed forward binary classifiers with binary features. A suggested improvement is to change the architecture to match the data. E.g., changing to a convolutional architecture to produce images. However, this complexity increases the hardware requirements to train the model. It is also likely that adversarial examples produced by MalGan will be human discernible.

References

[1] Sahu, Suchismita. “Decision Boundary for Classifiers: An Introduction.” Medium, Analytics Vidhya, 8 Sept. 2021, https://medium.com/analytics-vidhya/decision-boundary-for-classifiers-an-introduction-cc67c6d3da0e.
[2] Brendel, Wieland, et al. “Decision-Based Adversarial Attacks: Reliable Attacks against Black-Box Machine Learning Models.” ArXiv.org, 16 Feb. 2018, https://arxiv.org/abs/1712.04248.
[3] Brendel, Wieland, et al. “Accurate, Reliable and Fast Robustness Evaluation.” ArXiv.org, 12 Dec. 2019, https://arxiv.org/abs/1907.01003.
[4] Goodfellow, Ian J., et al. “Generative Adversarial Networks.” ArXiv.org, 10 June 2014, https://arxiv.org/abs/1406.2661.
[5] Brownlee, Jason. Generative Adversarial Networks with Python. 1.8 ed.
[6] Warr, Katy. Strengthening Deep Neural Networks: Making AI Less Susceptible to Adversarial Trickery. O'Reilly, 2019.
[7] Hu, W., &amp; Tan, Y. (2017, February 20). Generating adversarial malware examples for black-box attacks based on gan. arXiv.org. Retrieved August 14, 2022, from https://arxiv.org/abs/1702.05983