Adversarial examples, theory and evidence in computer vision

University of Turku
Department of Computing
Master of Science Thesis
Computer Science
June 2025
Laurynas Gurevicius

UNIVERSITY OF TURKU
Department of Computing

Laurynas Gurevicius: Adversarial examples, theory and evidence in computer vision

Master of Science Thesis, 58 p.
Computer Science
June 2025

Adversarial examples are input samples modified with minimal perturbations. These samples cause misclassification in machine leaning models. This thesis is constructed like a survey: first we present a broad history of computer vision and its history with neural networks, then we proceed to discuss various adversarial attacks and defenses, and thirdly we detour to anomaly detection. Purpose of these sections is to give context to the analysis of theoretical frameworks of adversarial examples. Theoretical frameworks are analyzed and evidence for their claims is explored through other more practical sources. Practical sources didn't discuss frameworks directly, rather they had their own presentation and evidence in them touched on other claims. Finally we conclude the analysis, categorization and proceed to suggest future research directions.

Keywords: adversarial, examples, robustness, vulnerability, perturbation, computer vision, deep learning, theory, theoretical, framework, anomaly, outlier, detection

Contents

List of Figures

List of Tables

1 Introduction

Deep learning has proven to be useful in various fields that require model complex functions by leveraging data and presence of patterns. For example deep learning has been applied computer vision tasks such as image classification and object detection with the help of convolutional neural networks and transformers. [1] However, these deep learning models have been found to be vulnerable to small and benign perturbations imperceptible to humans but specifically designed to cause a misclassification [2]. These perturbations, called adversarial examples, are a risk if the model gets deployed in the real world. [3] For example causing misclassification of road signs by an autonomous driving system can result in serious traffic accidents [4]. This serious issue has gotten a lot of attention from research communities in various deep learning fields, but most mature literature exists in computer vision [1]. This thesis will focus on the application of adversarial examples in deep learning image classification.

We will study adversarial examples, adversarial attacks and defenses and their theoretical explanations. There will be a detour to anomaly detection, where we will present some common themes and tools. Theory of adversarial examples will be evaluated with empirical evidence from the deep learning and anomaly detection sides. We hope that this analysis will provide some insights into limitations of neural networks and deep learning.

Guiding research questions are:

1. What are adversarial examples?

2. What are the theoretical explanations for adversarial examples?

3. Does theory provide any insights or possible paths to adversarial robustness?

Sources were searched and retrieved from citations, Google Scholar or Utu-volter. Keywords used were "adversarial examples", "adversarial robustness", "adversarial defense", "computer vision", "outlier detection", "theory" and "theoretical". These keywords where used in various combinations.

Math notation is unified and described in the table 1.1.

Table 1.1: Math notation

Symbols	Description
x	A sample of input data.
y	A sample of output data.
h	A sample of output from a hidden layer.
z	A sample of output from a final layer.
x^*	Adversarially modified sample of an input data.
θ	Parameters of the model.
ϵ	Distance between an adversarial and a clean sample.
$\|\cdot\|_2$	Euclidean norm (distance)
$\mathcal{L}(\theta, x, y)$	Loss function used to train the model. Can be simplified to $\mathcal{L}(x, y)$.
∇_x	A gradient with respect to x.
η	Learning rate.

2 Computer vision

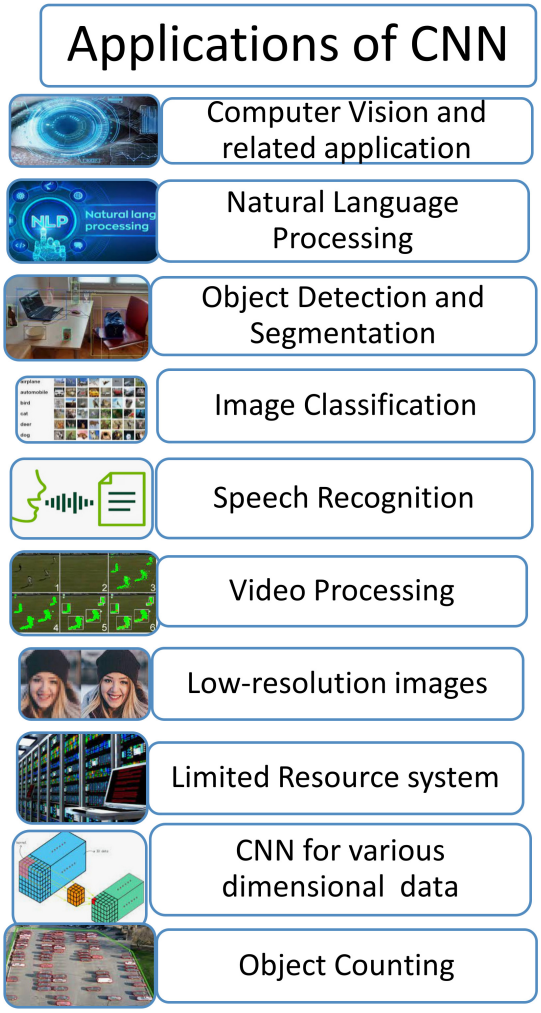


Figure 2.1: Figure 13 by Akhtar, Mian, Kardan, *et al.* [1] is licensed under CC BY 4.0.

Computer vision is a process of refining meaningful data from high dimensional real world data with some machine learning method. Purpose is to automate ex-

traction, analysis and understanding and replace human visual system with artificial system. Data can be an image, a video sequence, a point cloud, etc, as long as it models geometry, physics, motion and textures of the real world. Various subfields of computer vision are image classification, object detection, video tracking, etc (Figure 2.1). More specialized tasks based on previous methods can be object classification, pose estimation, facial recognition, motion tracking, scene and image reconstruction, etc. Applications of computer vision can be found in automatic inspection for example in manufacturing, visual surveillance and detection, visual monitoring, modeling objects in medical image analysis and navigation in autonomous vehicles.

Computer vision is unexpectedly very complicated process so any ambitious artificial system must have following processes in place: data acquisition, preprocessing, feature extraction, feature detection and segmentation, high level processing and decision making. Data acquisition involves the connection from data source to other modules, for example connecting cameras or other image sensors to the hardware. Preprocessing involves clean up, normalization and other noise reduction methods. Feature extraction in vision involves the separation and recognition of low level geometry, such as lines, points and edges. These low level primitives are combined in next stage into the detection of salient objects. Final stages involve task specific decision making and processing of intermediate data. The advent of deep learning has shown deep neural networks can learn and integrate a lot of these stages into one whole model with an effective accuracy. Learning happens with a remarkable ease without any task or domain specific engineering.

2.1 History

Early developments in 1940s focused more on modeling the neurons and their individual and combined behaviour than actual computer vision tasks. For example MCP model, the first artificial neural network, and Hebbian learning, local self mod-

ification/learning rule. These concepts were later refined into more modern concepts such as perceptron and backpropagation. The 1980s consisted of initial explorations into deep learning with unrestricted and restricted Boltzmann machines, and with recurrent neural networks. So called premodern deep learning begun in 2000s with first convolutional neural network called LeNet, long short term memory (LSTM), and Deep Belief Network that used unsupervised local greedy update rules for multiple layers of Restricted Boltzmann Machines. The advent of publicly available labelled datasets and parallel GPU computing accelerated the development of deep learning and computer vision. A true breakthrough was done in 2012 by AlexNet, a convolutional neural networks trained on GPUs for ImageNet classification. [5]

2.2 Boltzmann machines

Boltzmann machine is a primitive neural network that learns stochastic features of the data distribution. Its main features are global energy based learning and unrestricted connectome of neurons. The network is trained by updating the weights based on energy given by the neuron. Training lasts until energy equilibrium is reached. The global energy is measured with

$$E = -(\sum_{i<j} w_{ij} s_i s_j + \sum_i b_i s_i) \ ,$$

where w_{ij} is the connection strength, s_i is the state and b_i is the bias. [6]

A restricted Boltzmann machine is an undirected graph with stochastic visible neurons and stochastic hidden neurons. The main difference to ordinary Boltzmann machine is the fact that visible and hidden must fulfill separate groups of bipartite graph. [5] This structure allows for more independent hidden neurons [7] and for

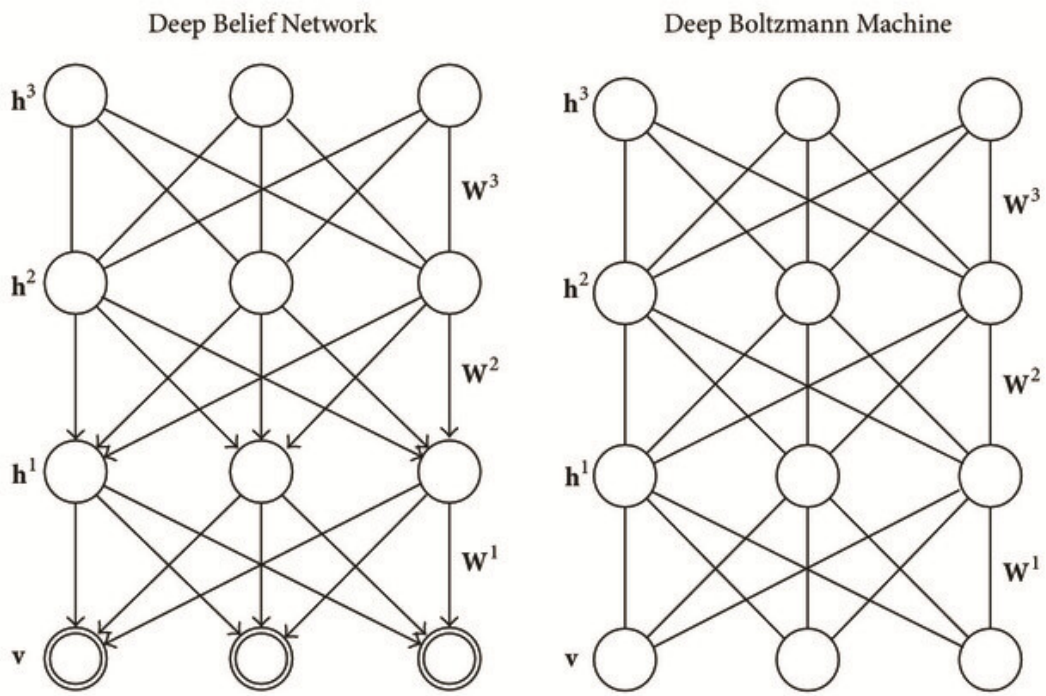


Figure 2.2: "Deep Belief Network (DBN) and Deep Boltzmann Machine (DBM). The top two layers of a DBN form an undirected graph and the remaining layers form a belief network with directed, top-down connections. In a DBM, all connections are undirected" [5]. Figure 2 by Voulodimos, Doulamis, Doulamis, *et al.* [5] is licensed under CC BY.

more efficient gradient-based learning [5]. The energy function has the form of

$$E = -\sum_{i=1}^{V}\sum_{j=1}^{H} W_{ij} v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j \ ,$$

where $\mathbf{W}$, $\mathbf{v}$ and $\mathbf{h}$ are model parameters, and a_j, b_i are the bias terms. [5] Restricted Boltzmann machines suffer in practise due to [7]:

1. intractable loss function. This leads to diverse set of solutions with various qualities. Such fluctuations are hard to compare.

2. sensitivity to hyperparameters.

3. sensitivity to data representations.

Deep Belief Networks were reached by stacking multiple restricted Boltzman machines together and training in a greedy manner. These probabilistic generative models operated under joint distribution of data and labels to fine-tune all weights for the desired outputs. This structure allowed for a deep hierarchical representation, but it suffered with heavy computational costs and inability to adapt to visual domain. Another form for deep neural networks was a Deep Boltzmann Machines, where difference to Deep Belief Networks was in last two layers. Deep Belief Network had last two layers as directed, on the other hand all layers of Deep Boltzmann Machines were undirected. Advantages of Deep Boltzmann Machines were possibility of unsupervised learning, complex and hierarchical representations. A drawback was computational costs of inference. [5]

2.3 Convolutional neural networks

Inspirations for convolutional neural networks can be traced back to studies cat visual cortices [8]. Those studies found out that neurons responded to small regions in visual fields. Neighbouring cells shared similar regions in the visual field, and similar tasks such as orientation, color, shape, object or motion detection. Primitive processing was done in primary visual cortex and more advanced processing was passed further along to other regions of the brain. [9] These insights were put together by Yann Lecunn in the design of convolution neural networks (CNN) called LeNet. Neurons were described in such a way that a collection of them would be applied to all regions on the image, achieving translation invariance. Network was trained with backpropagation. [5] Although feature hierarchies were leaned automatically, LeNet suffered from poor scaling to various picture classes, too large filter dimensions, and weak feature extraction [10].

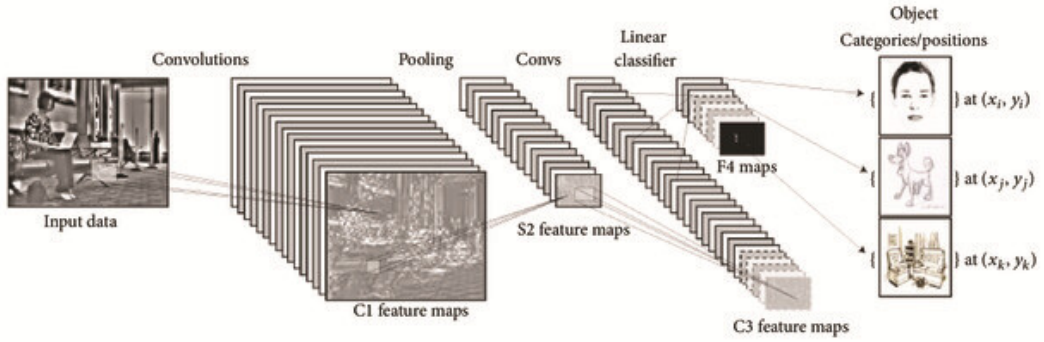
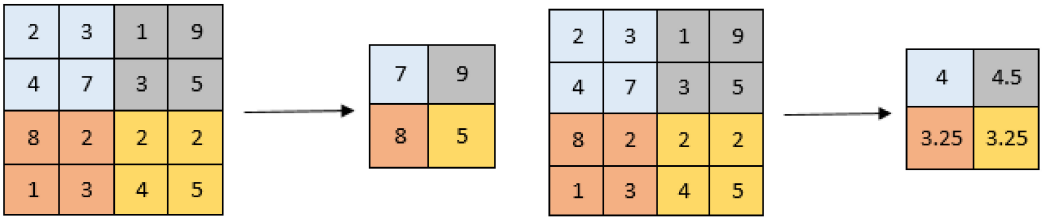


Figure 2.3: An example of CNN architecture applied to a computer vision task. Figure 1 by Voulodimos, Doulamis, Doulamis, *et al.* [5] is licensed under CC BY.

Default CNNs consist of three different types of layers: convolutional, pooling and fully connected layers (Figure 2.3). Each layer has a different task. Convolutional layers are described as kernels/filters and their job is to separate and generate various features for example edges and lines. These layers target some local receptive field, process them and transfer them to next layer. The gradual combination of features allows for the separation and extraction of elementary visual features. [5] Latter layers deal with more abstract features and objects [11]. Pooling layers reduce spatial dimensions (width and height) and volume (channels). The loss of information reduces computational costs and overfitting. Most used pooling strategies are average and max pooling (Figure 2.4), but other variations, such as stochastic and spatial pyramid pooling, are used for more specific purposes. For example spatial pyramid pooling is used in object segmentation. [5] By increasing the the width each filter, pyramidal structure could retain all the information and extract features from every location. In PyramideNet this came at the cost of high conceptual, spatial and time complexity. [10]

Ultimately, a great advantage of both convolutional and pooling layers is the convolution operation because it can substitute fully connected layers for fractional



(a) Max pooling with a stride of 2. (b) Average pooling with a stride of 2.

Figure 2.4: Figures 9 and 10 by Voulodimos, Doulamis, Doulamis, *et al.* [5] are licensed under CC BY 4.0.

neuron count. Fully connected layers are simplest layers used for high level processing in CNNs. Each neuron has a connection to each output of previous layer. [5]

The convolutional layer d of dimension $m \times m$ receives the input

$$y_{ij}^{(d)} = \sigma(x_{ij}^{(d)} + b)$$

and transforms it by convolving over the receptive field:

$$x_{ij}^{(d)} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} y_{(i+a)(j+b)}^{(d-1)},$$

where b is a bias term and w is an element of a kernel. [5]

Tombe and Viriri [12] gives five different axes for evaluating variations of CNNs: convolutional layer configurations, pooling method, nonlinear activation functions, network parameter regularization, and network optimization.

Because both convolutional and pooling filters are the underlying unit of convolutional neural networks, modifying them leads to intrinsic change in bias. For example, models using deconvolution filters may remove noise by upsampling the input with appropriate dilation factors. Other convolution reconfigurations are tiled convolution filters and inception modules. More unique one, inception module, attempts to increase separation power by pooling convolution operations of various

sizes in parallel. This module reduced training costs and network parameter count. [12]

Various models that used inception modules were Inception-V3, Inception-V4 and Inception-ResNet. These models either added bottlenecks, asymmetrical filters, skip connections. Due to complexity in basic inception module and its various variations, the models were able to represent variation in high-detail photographs and boost the richness in intermediate layers, but the final architecture always suffered from conceptual, memory and temporal complexity leading to slow learning and inference. [10]

The change in pooling layers doesn't change the underlying bias of pooling, instead it changes how it is done. For example, mixed pooling attempts to combine the strengths of max and average pooling by interpolating between them. This pooling method doesn't change the purpose of pooling, it only tries to reduce overfitting problems of max and average pooling. Other methods called spectral pooling that operates and crops in frequency domain, and stochastic pooling that picks activations depending on distribution, follow the same implicit bias. [12]

Changes in activation functions can lead to performance boosts in some specific tasks. It seems to inherit simplicity of ReLU function was so powerful, that significant portion of literature was dedicated to hiding its weaknesses or exploiting its strengths to a greater degree. For example, LeakyReLU function added a non zero tail so that gradient wouldn't be a zero. On the other hand regularization had a very simple purpose, to reduce overfitting. [12] However, no CNN specific techniques were invented, just old machine learning methods such as dropout and l_p-norm were used.

On the other hand, convolutional neural networks did force the research in to the methods that would improve training speeds. These variations on optimization included various weight initialization, data augmentation schemes, batch normal-

ization and skip connections. Different weight initialization procedures attempted to gain quick convergence in training and simultaneously avoid the vanishing gradient problem. These schemes used various distribution, preferably Gaussian, with various standard deviations based on the dimensionality of inputs and outputs. To address the lack of data geometric transformations were used to fill the void. [12] Batch normalization was used to solve issues with gradient descent by forcing internal representation to zero mean and unit variance [13]. It allowed for greater learning rates without the risk of vanishing or exploding gradients [12].

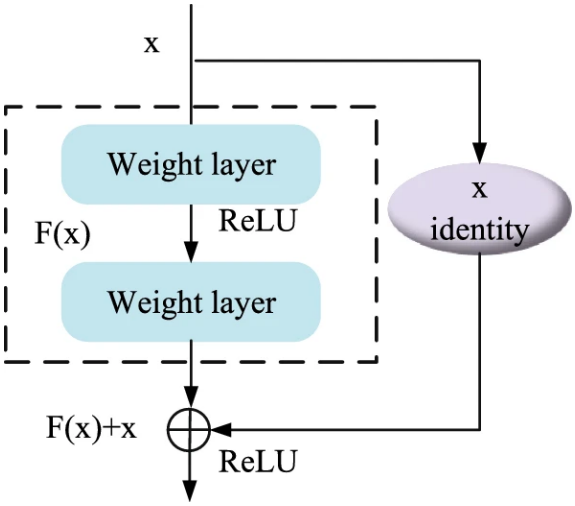


Figure 2.5: A skip connection. Figure 21 by Zhao, Wang, Zhang, *et al.* [13] is licensed under CC BY 4.0.

Another solution just for vanishing and exploding gradients were skip connections. These connections worked as information highways (Figure 2.5) allowing for infinitely deep networks in the process. [12] First practical model built on skip connections was called ResNet. It clearly reduced the error rate for deeper networks by reducing the vanishing gradient problem. It suffered from complicated architecture, degraded feature-map and overspecialization of hyperparameters. The wide version, a so called Wide ResNet, exchanged depth for wideness. This allowed for more effective feature reuse and dropout, at the cost of overfitting. [10]

Although strong performance has been demonstrated, there are some broad and

practical problems with this architecture. First, explainability of individual filters is questionable leading to unverifiable models. Second, CNN architecture's performance is dependent on hyperparameters such as filter stride, depth or size. The choise of these parameters is highly dependent on heuristics and knowledge, and down the road lead to significant differences in performance. Third problem is the data and computational consumption. Although not on the level of transformers, convolutional neural networks still require a lots of data and quality labels. The learning process requires to hold a lot of parameters in memory, which in turn limits the architectural size. Final problem is the lack of mathematical and theoretical framework to guarantee some bounds and properties. Without such a framework all the model design will be based on the intuition and knowledge of an engineer. [13]

2.4 Transformers

A latter breakthrough in computer vision was the attention mechanism of transformers. This mechanism can be described as dynamic weight adjustment based on the data and patterns at hand. However, attention mechanism didn't start with computer vision but with natural language processing (NLP) [14]. Transformers were originally intended for parallel processing of sequences and learning long-range relationships. This inductive bias opposed rivaling architecture such as recurrent neural networks, which could only hold a short-term context. The simple design and performance strengths allowed the application of transformers in various other modalities, for example audio and video, by dividing the data in discrete processing blocks. [15]

Self-attention layer of transformers processes the data in a following manner: first input vectors were transformed into query and key space; then scores between queries and keys were calculated and normalized; and finally softmax probabilities would be weighted with value vectors. The idea is to determine shared weight

between two input vectors by using query and key spaces, and then weight the attention appropriately based on the original input vectors. [16]

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{Q \cdot K^T}{\sqrt{d}}) \cdot V,$$

where $Q = W_Q X, K = W_K X, V = W_V X$.

In computer vision, attention can be applied to spatial domains, channels or both. Spatial attention attempts to select most important regions or predict the placement of most important regions and channel attention attempts to select most important channels. By actively calibrating the channel weights, the attention mask guides computation to the specific objects. The hybrid of both spatial and channel attention will produce a joint 3D mask at the cost of performance and limited receptive fields. However, it can highlight more informative features in the intersection of spatial and channel domains. [17]

The backbone of computer vision transformers that exploit spatial domain are so called vision transformers. Simplest vision transformers use pure transformers by converting images into local patches and run them through the stacks of transformers, attention layers and skip connections. Finally, if necessary, patches are going to be joined, flattened and fed to MLP classifier network. More advanced version inject spatial location, introduce bottlenecks for efficiency, model pixel level features with a smaller and embedded transformer, introduce convolutional inductive biases, modify patch design and hierarchy, and force communication between attention heads. [18]

The backbone of channel attention based transformers is a so called SENet that uses *squeeze-and-excitation* block to collect global information and capture channel-wise relationships. The global average pooling squeezes the information and the attention excites input features. The successor attempt to rectify weaknesses, for example by replacing global average pooling due to its ability to only capture first-

order statistics, utilizing mean and standard deviation of the input features, or replacing channel reduction with 1D convolution. [17]

Hybrid transformers don't have uniting core architectures, so we will present three different ones. First, residual attention network that combines an attention mechanism with the skip connections. Spatial and cross-channel dependencies are retrieved by applying max pooling, interpolation to original size, and finally 1D convolution. Second, convolutional block attention module stacks channel attention and spatial attention in series. This allows to use both types without the overwhelming cost of joint attention. Third, spatial and channel SE blocks use *squeeze-and-excitation* block for both spatial and channel attention separately in parallel. The final outputs are joined allowing the capture of pixel-wise spatial information.. [17]

Transformers however also have some performance considerations. They have an appetite for a large amounts of data and a demand for large computational capacity. Demand for a large data amount is attributed to high parameter count and lack of inductive biases for visual data. For example transformers need to figure out translation invariance, weight sharing and scaling. Tranformers require hundreds of millions of images, if you want to get a reasonable performance on the ImageNet benchmark. And allthough transformers have high initial barrier in terms of computational cost, with enough data and time transformers surpass due to more efficient exploitation of data. [15]

3 Adversarial examples

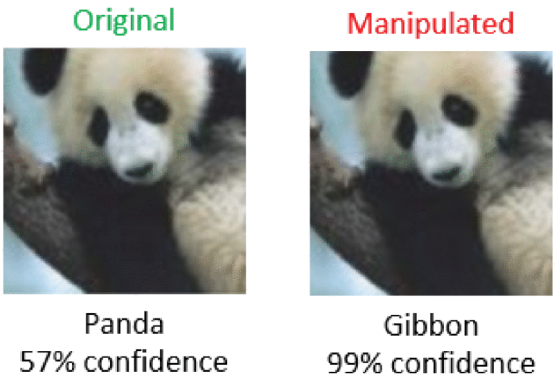


Figure 3.1: Predictions from clean input image on the left, and predictions from adversarially modified input image on the right from deep visual model. Figure 1 by Akhtar, Mian, Kardan, *et al.* [1] is licensed under CC BY 4.0.

What is an adversarial example? We will use the definition from Qian, Huang, Wang, *et al.* [19] that defines it as "a perturbed input sample with imperceptible noise that would deceive the model to output incorrect results" (Figure 3.1). The noise that causes this misclassification is referred to as adversarial perturbation. This manipulative noise should be "inconspicuous" (Figure 3.2), so barely perceivable to the human evaluator. [1], [19] A clean sample would be a sample without the perturbation. An adversary would be the agent creating and applying the perturbations, an attack would be a method that generates adversarial examples, and an adversarial defense would be the mechanism that ideally prevents the misclassification. [1], [20]

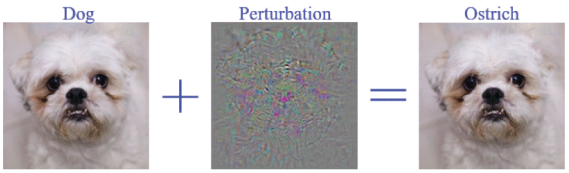


Figure 3.2: Picture of a dog combined with magnified perturbation. Final result doesn't use exaggerated perturbation. Figure 3 by Akhtar, Mian, Kardan, *et al.* [1] is licensed under CC BY 4.0.

3.1 Adversarial attacks

Attacks can be further divided into two broad categories: white-box and black-box attacks. White-box attackers have full access to the target model, so they have knowledge of model weights and architecture, training and test data, and they can query models as many times as they want [20], [21]. Black-box attackers lack access to the parameters and architecture of the model and the defense, so they can only observe the results. These types of attackers either build queries or synthetic models, which can be exploited with white-box attacks. [20], [21] This is the most important separation of attacks, as it can be used to evaluate defensive capabilities to all attacks or attacks in the real world [20]. Other orthogonal categorization axes of attacks are the universality and targeting. Universality touches on how wide do the adversarial examples reach, in other words do they cause misclassifications in all the models, or only in specific families. Targeting means if misclassification is intended to work in a specific way or is any misprediction fine. [20]

Formally this perturbation is defined as a vector applied to a sample such that distance is less than some threshold, and it causes a mislabeling of the clean sample.

$$f(x) \neq f(x + \epsilon)$$

Then any adversarial attack can be defined simply as an optimization problem, such that they attempt to increase misprediction rate constrained by the distance from

the original sample or magnitude of the noise vector. Distance is not limited to an Euclidean distance, it can be any norm. Misprediction metrics and magnitudes are defined more precisely by each attacking method. [19], [20]

3.1.1 L-BFGS

L-BFGS in an early attack introduced to target computer vision models. Under misclassification requirements, authors used the L-BFGS algorithm (Limited Memory Broyden – Fletcher – Goldfarb – Shanno algorithm) to transform a problem into a constraint problem. [3], [20] Formally the goal is to minimize

$$c\|\epsilon\|_2 + \mathcal{L}(x^*, y) \quad \text{such that } x^* \in [0, 1]$$

where x^* is normalized to $[0, 1]$, and c is a variable optimization process iterates with line search to find the minimum $c > 0$. Informally this algorithm calculates inverse Hessians with quasi-Newton computations [1]. Formulation via the Langrangian multiplier had enough flexibility to include additional limitations into the problem. Constrained with euclidean distance, the algorithm provided perturbations that were imperceptible to humans but adversarial for computer vision models. This attack was applied to the AlexNet that was state-of-the-art at that time. [3], [20]

3.1.2 Fast gradient sign method

Another technique armed with greater access was developed by Goodfellow, Shlens, and Szegedy [22] called fast gradient sign method (FGSM). This method was constructed under the assumption that adversarial examples were artefacts of noisy dot products in high dimensional space. [22] This one-step gradient based method focuses on efficiency rather than fooling rate [1]. This method uses gradients to

update perturbation in the direction of the gradient at each pixel:

$$x^* = x + \eta \operatorname{sign}(\nabla_x \mathcal{L}(\theta, x, y))$$

Algorithmic and conceptual simplicity allows to add various extensions for example involving momentum or normalization [3]. For example Fast Gradient Value Method (FGVM) removes sign function, or I-FGSM adds iteration which was further extended to use momentum in MI-FGSM [1]. These attacks transferred well but they were also easy to defend [3]. This method is heavily used in adversarial training because it is easy to conceptualize and perturbations are analytically computed. Formalization of FGSM also allows for application in different fields, for example fooling reinforcement learning models in terms of the quality of their actions. [20]

3.1.3 Universal adversarial perturbation

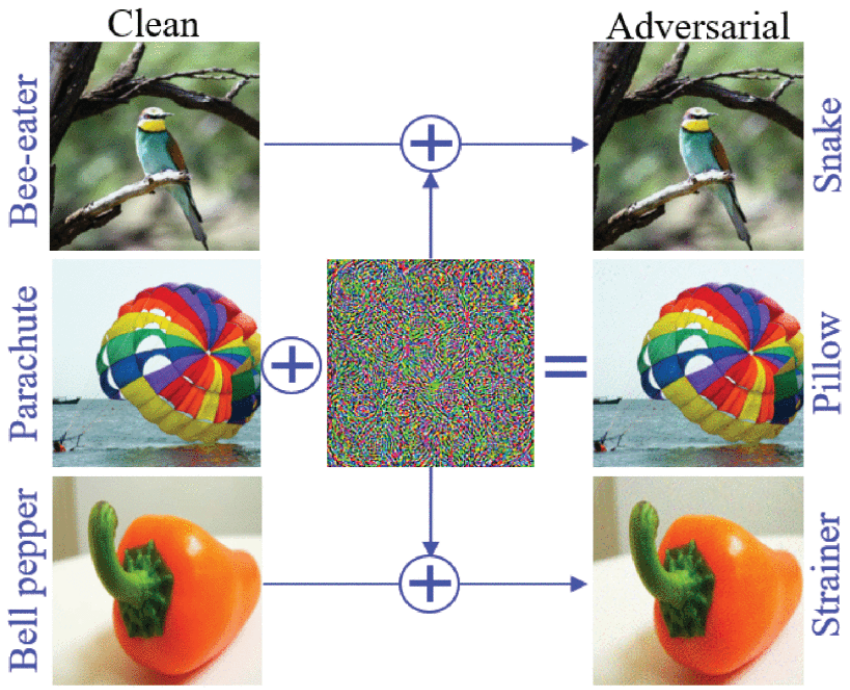


Figure 3.3: An universal adversarial perturbation pattern applied to multiple images. Figure 6 by Akhtar, Mian, Kardan, *et al.* [1] is licensed under CC BY 4.0.

Universal adversarial perturbation (UAP) is an attack based on an iterative algorithm that attempts to reach a predefined misclassification rate (fooling rate in literature) under perturbation constraints [3]. Algorithm itself has a simple idea, it simply adds an offset to the next decision boundary until the desired fooling rate is reached. The algorithm 1 is defines some additional functions such as

$$\text{Error}(X) = \frac{1}{n}\sum_{i=1}^{n} 1_{f(x_i+v)\neq f(x_i)}$$

and

$$\mathcal{P}(v) = \arg\min_{v^*} \|v - v^*\|_2 \text{ subject to } \|v^*\|_2 \leq \epsilon \, .$$

This method assumes that high dimensional decision boundaries of classifiers share geometric correlations. [23] At each iteration, each sample's accumulated perturbation gets summed together and normalized against some maximum and minimum bounds. In essence the algorithm pushes out of the decision boundaries of all classes simultaneously. [20] The greatest boon of this method is the fact that adversarial examples can be generated on very few samples of training data [3], and that they are universal, in other words a single perturbation vector can be applied to all samples of the same dataset (Figure 3.3) [20]. Universal adversarial perturbations transfer well across different models and image datasets [23], however they are not fully imperceptible to humans [1]. Fine-tuning on adversarial samples did reduce the fooling rate slightly, but it didn't make networks immune to universal perturbations [23].

Listing 1 Universal Adversarial Perturbation

```
INPUT data X, classifier f(...), threshold δ
OUTPUT universal perturbation vector v
SET v ← 0
WHILE Error(X) ≤ 1 − δ DO
    FOR EACH x ∈ X DO
        IF f(x + v) = f(x) THEN
            Δv ← arg min_r ‖r‖_2 subject to f(x + v + r) ≠ f(x)
            v ← P(v + Δv)
        END IF
    END FOR
END WHILE
```

3.2 Adversarial defenses

Defenses can be broadly categorized into methods that somehow modify training, model architecture or parameters, modify inputs (3.2.12) or add external modules that detect perturbations (3.2.13) [1]. Each of these methods have their own benefits and drawbacks, and each of them can be further subdivided into more granular subclasses. Some defenses can defend against a broad number of norms or perturbation budgets, some specialize in white-box or black-box attacks, and some deal with various computational constraints [2].

Modification to learning are adversarial training (3.2.1), active learning (3.2.2), feature representation learning (3.2.3), stochastic activation (3.2.4) and game theoretic methods (3.2.5) [2]. Architecture-based defensive modifications are discussed in 3.2.6. Modifications to parameters can be of various complexities and are usually done in tandem with adversarial training [1]. Dhamija and Bansal [2] divides them into defensive distillation (3.2.7), gradient masking (3.2.8), feature map construction (3.2.9), logit investigation (3.2.10) and regularization (3.2.11).

3.2.1 Adversarial training

Conceptually simplest one is called adversarial training. This training uses adversarial examples in hopes of achieving immunity and robustness, so in practice the model is trained using the clean samples and their adversarial equivalents [2]. Standard formulation of adversarial training, for example used with FGSM, is defined by Wiyatno, Xu, Dia, *et al.* [20] into:

$$\alpha\mathcal{L}(x, y) + (1 - \alpha)\mathcal{L}(x, y) \ .$$

Single-step adversarial training is too simple, so models memorize and overfit them easily, and fail to generalize the attacks [20]. More advanced version is formulated as a min-max problem where inner maximization searches for the worst perturbations and the outer minimization optimizes for the best classifier [19]. It is called Projected Gradient Descent (PGD) -adversarial attack and formally defined by Wiyatno, Xu, Dia, *et al.* [20] as:

$$\arg\min_{\theta} \mathbb{E}\left[\max_{\|x-x^*\|_\infty} \mathcal{L}(x^*, y) \right] .$$

Due to the non-convex nature of perturbations and parameters, Lagrangian multiplier method is usually applied to the formulation [19]. PGD adversarial training benefits from increased robustness in bigger models and converging training results, but suffer from reduced clean accuracy [20].

Due to the conceptual simplicity and "principled" nature of adversarial training, it is the most popular defensive method in use and it has a lot of attempted theoretical frameworks [1], [2]. However, some literature notes that increased generalization requires bigger models since adversarial training increases decision boundary complexity. Early stopping is offered as a solution for overfitting. [19] In an attempt to reduce the cost of other drawbacks, other variants for example focus on mislabeled samples, schedule dropouts in tandem with adversarial samples, include randomness

and shuffling of adversarial/clean samples, or increase training speed. [1], [2]

Other drawbacks are computational cost, sensitivity to other new attacks, and a failure to construct a unifying theory of adversarial training. [1], [2] Also Dhamija and Bansal [2] discusses such issues as decline in clean accuracy after adversarial training and failure to unify image classification models with object detection and object segmentation.

3.2.2 Active learning

Active learning uses optimized queries that attempt to find most informative labels in the large dataset so that need for human annotations would be minimal [2]. A practical and simple algorithm would use an unlabeled training set and gradually offer some for annotation. Picked samples could be randomly sampled, generated with GANs [24] or based on biggest adversarial vulnerability (the shortest effective adversarial perturbation) [25]. Performance of this method depends on query strategy, and it requires iteration, so it can suffer from computational costs. Broadly these methods cannot ingest highly correlated datasets and they should be integrated with model selection. [2] These methods could be thought as unsupervised or self-supervised adversarial training.

3.2.3 Feature representation learning

Feature representation learning attempts to improve feature separation ability [2]. Motivated by an assumption that adversarial examples add noise to intermediate feature representations, Xie, Wu, Maaten, *et al.* [26] adds various trainable denoising blocks, consisting of non-local means, bilateral, mean and median filters. Using similar assumptions Joe, Hwang, and Shin [27] embeds variational autoencoders (VAEs) into the architecture. This VAE should force internal mini-max game where defender uses VAE as a threshold and adversary tries to perturb the features [27].

Feature representation learning is vulnerable to transferable attacks and robustness is very dependent on dataset quality, but it does increase robustness to all kind of noise and variation including adversarial attacks [2].

3.2.4 Stochastic activation

Stochastic activation adds random noise to intermediate representations. For example Stochastic Activation pruning (SAP) prunes a random subset of smallest activations and encourages them during training towards larger magnitudes. [2] The difference to dropout is the uniform pruning instead of threshold based random pruning. SAP has been shown to be vulnerable to iterative gradient-based attacks. [20] Block switching (BS) extends this random selection to whole filters and channels, meaning the selected sets don't process the input data. Stochastic activation defenses increase robustness against gradient-based attacks at the cost of reduced clean accuracy, computational overhead and reduced interpretability. [2]

3.2.5 Game theoretical methods

Inspired by the concept of a game equilibrium involving two strategic players, game theoretic approach attempts to formulate and apply various optimization methods that treat the attacker and the defender as players [2]. For example Lindqvist and Izmailov [28] constructs a mini-max game where GAN generates samples and specific discriminator separates them. Samples are projected to an alien manifold by integrating special discriminating *Real/Fake* labels (addition of such labels changes the original manifold). [28] Game theoretical methods can increase robustness to various attacks, but success depends on the quality of the problem/game formulation. This is a challenge because adversarial attacks might evolve or change, and by extension attackers targets and approach might evolve or change. This issue can be combated by forming classification ensembles. [2]

3.2.6 Architectural modification based methods

Changes to architecture are changes to the structure of a model, an activation function or a loss function. Architecture can be optimized with neural architecture search, where the target is a specific adversarial metric [1]. Other, more intentional changes to architecture for example in CNNs is the change of filter size and reach. Dhamija and Bansal [2] refers to these changes broadly as tightening receptive fields. Explicit changes to the architecture will always have to balance inductive biases and in case of pattern recognition architectures those biases would be the tradeoffs between local and global features [2]. Other examples given by Akhtar, Mian, Kardan, *et al.* [1] are the change of loss function away from cross-entropy, which would explicitly transfer adversarial learning signals, and combining ReLU with k-winner-take-all scheme that would increase sparsity and by extension robustness.

3.2.7 Defensive distillation

distillation was originally introduced as a way to compress ensemble of neural networks into one single neural network [20]. Defensive distillation is a process of training a larger network and transferring information to the smaller network [2].

Defensive distillation works in following manner [20]:

1. first, train the "teacher" network with modified softmax at the output layer. This softmax function is adjusted with temperature T that controls amount of noise in the class predictions;

$$\text{softmax}(z) = \frac{e^{z_i/T}}{\sum e^{z_i/T}}$$

2. then train the "student" network where training labels are class predictions of "teacher" network.

The main idea behind distillation is to reduce overfitting in "student" network [20], increase generalization and robustness to various perturbations [2]. Temperature controls how many features get transferred through [20]. Defensive distillation is a specific distillation regimen aimed to protect against gradient-based attacks. However, it is unsuitable against targeted attacks, since the added layer of complexity leaves holes in the defenses. [2]

3.2.8 Gradient masking

Gradient masking reduces information content of the gradients making models less vulnerable to iterative gradient-based attacks. Gradient masking damages the quality of gradients, which might lead to underfitting or overfitting and inability to generalize. [2] Many defensive methods rely on gradient masking [20] since any modification to inputs (for example data augmentation), intermediate representation (dropout) or loss function (regularization) can change the gradients. Although it is generally difficult for an attacker to create perturbations in noisy, flat or undefined loss landscape, it has been found that gradients are possible recover from high dimensional space. More protection against gradient recovery is done by concealing gradient information further. This is done by smoothing the gradients and labels or undergoing the distributional shift in the gradients. [2]

3.2.9 Feature map construction

Feature map construction tries to reconstruct internal representations robust to adversarial noise. This is done by passing internal representations through a separate module that filters and hopefully fixes vulnerabilities, for example convolutions, tree embeddings or sparse transformations. These model agnostic defenses might lead to increased model interpretability at the cost of information loss. [2]

3.2.10 Logit investigation

Logit investigation is a broad category that contains more specific methods such as logit correction and logit pairing. Logits are defined as class scores produced by final softmax layer of the neural network. logit investigation specifically tries to integrate clean logits with adversarial ones in a loss function [2]. Both logit correction and logit pairing can be categorized as a variation of adversarial training.

Adversarial Logit Pairing (ALP) tries to enforce aligned clean and corresponding adversarial logits [19]. Formally uses regularization term to penalize logits difference between clean inputs and adversarial ones:

$$\alpha \cdot \frac{2}{n} \sum_{i=1}^{n/2} \mathcal{L}(x_i, y_i) + (1-\alpha) \cdot \frac{2}{n} \sum_{i=1}^{n/2} \mathcal{L}(x_i^*, y_i) + \lambda \cdot \frac{2}{n} \sum_{i=1}^{n/2} \mathcal{L}_{LP}(z_i, z_i^*) \ ,$$

where $\mathcal{L}_{LP}$ is the logit pairing loss. ALP and PGD-adversarial training produced very robust networks but suffered from computational complexity at scale. [20]

Clean Logit Pairing (CLP) optimized for alignment between **all** clean and adversarial logits. A formal training objective for CLP was [20]:

$$\mathcal{L}(x_i, y_i) + \lambda \cdot \frac{2}{n} \sum_{i=1}^{n/2} \mathcal{L}_{LP}(z_i, z_{i+\frac{n}{2}}) \ .$$

Even though CLP compares logits from two different classes, CLP alone was enough to increase adversarial robustness [19]. It was suggested that reason for it was implicit encouragement for smaller norms that in turn supported the lack of confidence and by extension lack of overfitting in the model [20].

Ultimately, logit investigation does increase stability in the model by forcing the inputs closer to each other. However, performance and robustness are dependent on the choice of hyperparameters. [2]

3.2.11 Regularization

Regularization is a method to remove statistically insignificant losses from the gradient. This forces the loss to act on local training data. [2] Practically all various proposed regularization schemes improve generalization because they flatten the loss landscape and make gradient descent easier and faster [29], [30]. For example one of the earliest adversarial defenses, Deep Contractive Networks (DCN) added a regularization term to the loss function that penalized irrelevant directions:

$$\sum_{i=1}^{N} \left(\mathcal{L}(x_i, y_i) + \sum_{j=1}^{H+1} \lambda_j \left\| \frac{\partial h_j^i}{\partial h_{j-1}^i} \right\|_2 \right),$$

where H is the number of trainable layers and h_j^i is an activation at layer j and input i [20].

However, adversarial robustness is heavily dependent on the choice of hyperparameters [2]. High regularization term will prune low budget adversarial attacks, but it will also penalize clean performance and hurt generalization [20]. Ultimately, Akhtar, Mian, Kardan, *et al.* [1] notes that these defensive regularization schemes are generally used in tandem with other defensive methods, for example adversarial training. This will hide the weaknesses of regularization and magnify the strengths of other defensive methods.

3.2.12 Data modification methods

Simple input modifications contain various operations, such as data augmentation and compression [1]. Used augmentations are rotation, translation, scale and brightness. Other less usual important augments are flips and removal of information for example by using small black boxes to block some part of an image. These augmentations are then used to expand the dataset and fill out missing information. Excessive augmentation can lead to overfitting or it can reduce performance of a

model on clean samples. [2] Compressions for examples via SVDs or JPEGs are used to clean up the data. These modifications are usually paired up with some other defense. [1] Compressing contributes to reducing computational costs, but excessive compression may remove some vital information and lead to the lack of generalization [2].

More involved types of input modifications are input reconstruction and input denoising/filtering [1]. Input reconstruction can be applied to the whole image, or just a small patch. These reconstructions can be done with GANs or some other specialized algorithms. Nonetheless, these reconstructions increase robustness by removing the noise and by extension reduce the quality. Thus the most important factor is to balance feature preservation with noise removal. [2] Denoising can be done with various filters or some image restoration algorithm [1], [2]. Filters can quantize, smooth out, blur or do some other kind of transformation. Point is to remove unnecessary information and force the model to focus on the most relevant features. Potential drawbacks are the loss of relevant information, too specific and strong filtering, limited robustness and sensitivity to the hyperparameters. [2]

3.2.13 Detection based methods

Detection based methods are just additional modules that perform no modifying transformations or operations on inputs or parameters. Given the input, these modules detect adversarial examples at the inference time by classifying them into clean and non-clean samples. These modules have "plug-and-play" ability and can be deployed in various situations without any aversion. Methods themselves could be based on statistical techniques, separate classifiers, and they can be enhanced further with ensembles. [2] However, it is important to note that these methods all mix theory of statistics and math, so categories are not so clearly cut.

Benefit of separate classifiers is that they don't have to learn anything about

the data. Since these classifiers can be anything from machine learning, theory is well grounded, and it can be difficult to engineer appropriate attacks. However, model complexity increases and training becomes more complicated with two separate pipelines. [2]

Statistical detection could be based on detecting differences in distributions, dimensionality reductions, or other statistical techniques. Differences in distributions could be detected by modeling input or hidden distributions for example with GANs. These methods are known to be robust, but dynamic attacks will require retraining. Dimensionality reduction for example with PCA simply projects inputs to its principal axes. PCA is vulnerable to non-linear adversarial examples and exploitation of principal axes. Other dimensionality reduction techniques will have their own drawbacks. Other miscellaneous statistical methods presented by Dhamija and Bansal [2] were Kernel density estimation, Bayesian Uncertainty and Maximum Mean Discrepancy. These methods cover a lot of ground and offer robustness by focusing on adversarial patterns instead of data. Tradeoffs are inductive biases, assumptions of distributions and sensitivity to hyperparameters. [2]

4 Anomaly detection

Purpose of anomaly detection is to find abnormalities in data [31]. This field is an essential research branch for data mining, and is applied in various places such as fraud detection, cybersecurity and healthcare [32]. We are able to connect adversarial examples to anomalies in data because adversarial examples hold abnormal patterns that cause models to misbehave. Although anomaly detection literature has vague definitions of anomalies, domain dependent assumptions, and claims of weakly verified slight performance improvements [33], [34], we hope that the general frameworks and insights in anomaly detection prove to be useful in the analysis of adversarial examples.

4.1 Anomaly detection and adversarial examples

Foorthuis [33] provides a framework to categorize all anomalies (Figure 4.1 and figure 4.1). These anomalies lie on five orthogonal axes named in the following way: data type, cardinality of relationships, anomaly levels, data structure and data distribution. Data types can be of three types: quantitative, qualitative and mixed. Cardinality of relationships means how different features relate to each other: univariate or multivariate manner. In univariate cases it can be assumed that attributes have no relationships with each other and anomalies can be detected separately. Multivariate assumes some relationship exists. Anomaly levels simply mean how pervasive are anomalous features, are they atomic or aggregate. In other

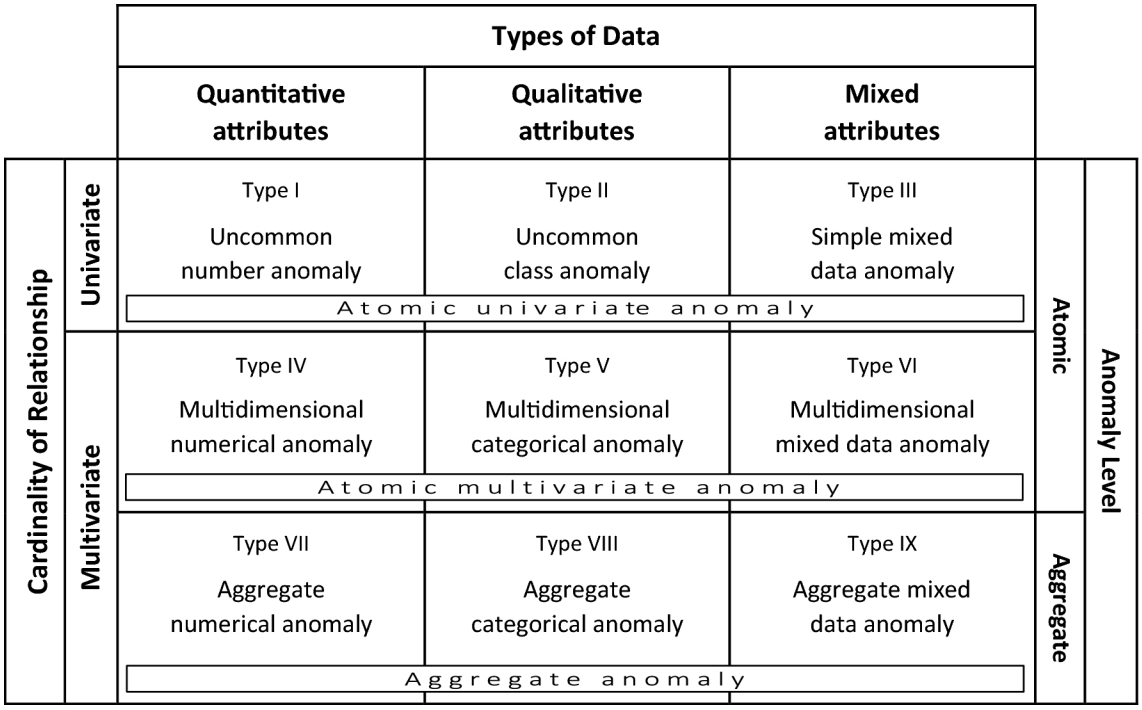


Figure 4.1: Typology of anomalies. Figure 2 by Foorthuis [33] is licensed under CC BY 4.0.

words this level differentiates anomalous samples and anomalous categories. Data structure refers to what kind of domain is involved: temporal or spatial, time series or graphs. Data distribution on the other hand refers to dispersion of patterns and samples in space. [33]

One way adversarial examples can be categorized is quantitative, multivariate and atomic. Foorthuis [33] calls this category "multidimensional numerical anomaly" (type 4). Anomalies of this type are further characterized as single cases where deviance appears as a combination of values not as single deviations. Furthermore these types are generally hidden in high dimensionality. Possible subtypes are divided based on where samples are positioned in the context of normal data. They can be on the periphery, enclosed by normal data, isolated in local space, or separated in global space. Authors note that algorithms that detect peripheral anomalies are not good with enclosed anomalies and detection of enclosed anomalies requires

bias, since enclosure could be a maze or a spiral. Local anomalies can only be separated in its neighborhood and such algorithms that detect these anomalies need to account for the context of the neighborhood. It is important to note that locality can mean the degree of multivariate subset, subsets of data distributions or some other domain specific neighborhood, for example in spatial neighborhood in images.

The other way adversarial examples can be categorized is quantitative, multivariate and aggregate. Remember how adversarial perturbations could be shared between different classes and transferred between different models. This type was called “aggregate numerical anomaly” (type 7). Recognizing type 7 anomalies requires domain knowledge and recognizing biases in the algorithms. For example in images there could be moving objects, altered light sources, transformed colors and changed camera settings. [33] Ultimately, Foorthuis [33] encourage to explicitly discuss for what kind of anomaly is the algorithm intended. Such clear behaviour would lead to “objective insight into the functional capabilities of anomaly detection algorithms”. Practically this kind of division and categorization could be used for performance metrics and test set evaluations [33].

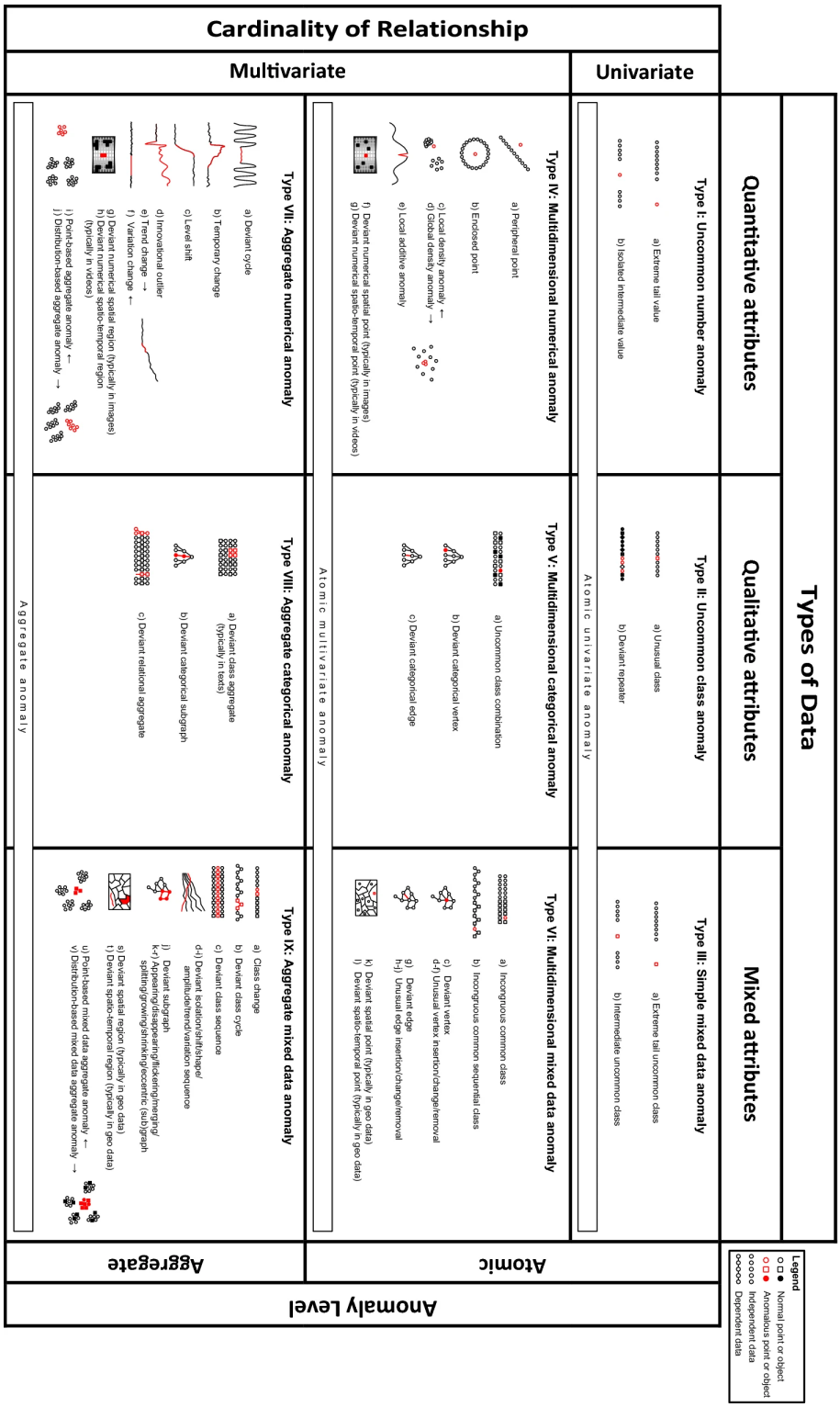


Figure 4.2: A closer look at the typology of anomalies. Figure 3 by Foorthuis [33] is licensed under CC BY 4.0.

4.2 Anomaly detection methods

Wang, Bah, and Hammad [32] divides anomaly detection methods into these seven broad groups: statistically based, distance based, density based, graph based, ensemble based and learning based methods:

- Statistically based methods assume that underlying data follows some distribution. Fast and easy to use, but cannot go beyond their mathematical definitions. [32] Statistical models, especially parametric ones, are vulnerable to noise and overfitting. Assuming the distribution is a double-edged sword which can lead to great generalization or low fitness. Also all statistical models might have some opaque measurements and tests that are hard to interpret. [35]

- Distance based methods focus on distance computations between data points. They are easy to implement and comprehend, and have a robust theoretical background. [32] Distance based methods can distinguish anomalies very well and find isolated groups, but they fail to provide local insights [35].

- Density based methods assume that outliers exist in low density regions. They are non-parametric but complicated and computationally costly [32]. Both distance and density based methods suffer from failure to generalize beyond training distribution [32], however density based methods are able to provide local insights [35].

- Clustering based methods simply search for clusters of data points and assume that outliers are not in the vicinity of those clusters. They are simple, scalable, robust to data types and unsupervised. Disadvantages are binary representation of outliers (in cluster or out of cluster), dependence on user parameters (number of clusters), and sensitivity to noise. [32]

- Graph based methods use graphs to capture dependencies and find outliers. They capture interdependencies of features well because they can be represented as nodes, edges, subgraphs, evolving edges, evolving subgraphs, or evolving distances [35]. However, these methods have not been studied in great detail [32] and their application requires some domain specific bias [35].

- Ensemble based methods combine results from dissimilar models with hopes of producing more robust models. Stable and robust to noise but increase the complexity of the task at hand. [32] They have great theoretical foundations which can be used in variance and bias reduction functions [35]. However, good performance requires diversity in underlying models, for example multiple vanilla neural networks will not give better adversarial defenses (Figure 4.3) [21].

- Learning based methods learn some subset of data and attempt to generalize learned patterns. Since these methods learn under supervision, they can save time. However, computational costs [32], class imbalances and novelties [35] might be a challenge.

These broad descriptions already highlight the strengths and weaknesses of various adversarial defenses. For example adversarial training shares weaknesses with learning based methods. Since most adversarial attacks try to minimize the adversarial perturbation, both Statistically and distance based methods would be weak and narrow defenses. Grosse, Manoharan, Papernot, *et al.* [36] shows this is true for a small FGSM-attacks with small budgets when trying to detect them. Seemingly best defenses would be provided by locally aware defenses such as graph, cluster and density based methods.

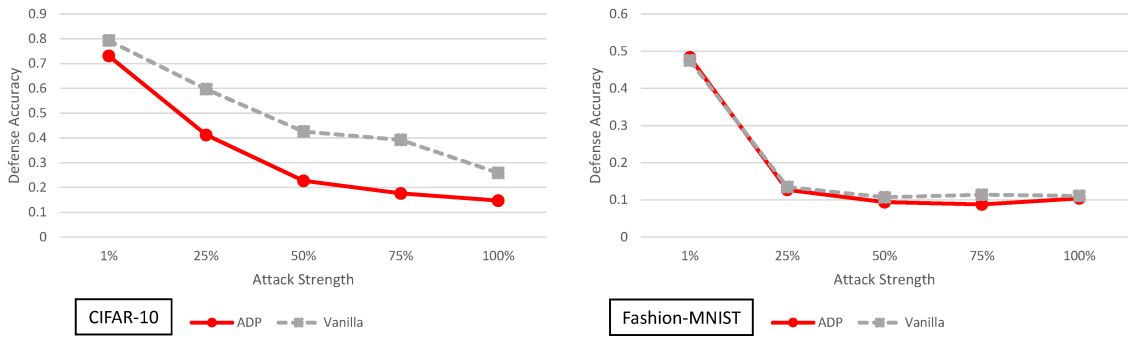


Figure 4.3: Adversarial defense accuracy of ensemble defenses against adversarial black-box attacks for CIFAR-10 and Fashion-MNIST [21]. Figure 11 by Mahmood, Gurevin, Dijk, *et al.* [21] is licensed under CC BY 4.0.

4.3 High dimensional spaces

Another axis of categorization tackles the methods that attempt to deal with high dimensional spaces. Challenges with high dimensional spaces are broadly characterized as "curse of dimensionality" or more concretely "distance concentration effects" [31], [37]. This means that Euclidean or any other distance metric fails to separate points as dimensionality grows [38]. Specialized algorithms choose to address these issues in two possible ways: by implicitly searching for relevant features and operating in smaller subspaces, or sacrificing implicit search for improved performance and efficiency since a simple subspace search suffers from combinatorial explosion with increasing dimensions. [37].

Zimek, Schubert, and Kriegel [37] provides further details on the "curse of dimensionality" in Euclidean spaces. Firstly, although distance between minimum and maximum vanishes as dimensionality approaches infinity, mean and standard deviation approaches some specific value too (Figure 4.4). So if you have different mechanisms generating clean data and outliers, with assumption of meaningful dimensions, as dimensionality increases it should be easier to separate outliers. Secondly, if it is assumed that outliers live in irrelevant dimensions, the more relevant dimensions there are, the easier it is to separate outliers. On the flip side,

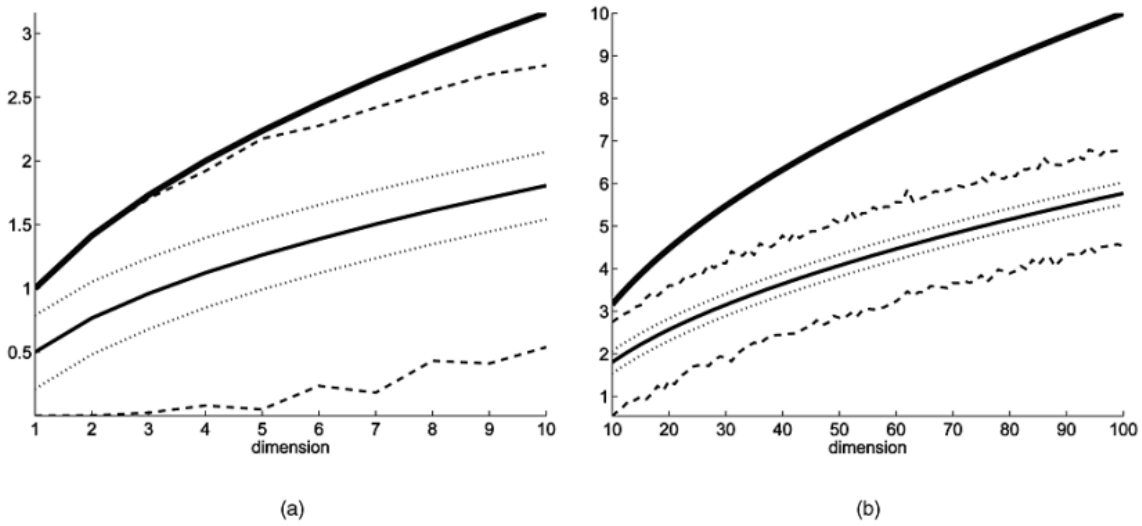


Figure 4.4: "From bottom to top: minimum observed value, average minus standard deviation, average value, average plus standard deviation, maximum observed value, and maximum possible value of the Euclidean norm of a random vector. The expectation grows, but the variance remains constant. A small subinterval of the domain of the norm is reached in practice." [39]. Figure 1 from François, Wertz, and Verleysen [39], © 2007 IEEE.

authors note that "A high portion of irrelevant i.i.d. attributes can mask the relevant distances". Finally discerning the distribution of data points becomes harder with increasing dimensionality because of how volume behaves in high dimensional spaces. [37] François, Wertz, and Verleysen [39] show that this kind of behaviour is intrinsic to high dimensionality, and that any norms, including fractional ones, will concentrate. For any norms, it was also proved that upper and lower bounds of standard deviations depend on the shape of the data distribution François, Wertz, and Verleysen [39].

4.4 Subspace methods

Subspace methods thus offer one big advantage. They try to select only relevant features thus reducing the complexity of a task [31], thus avoiding dilution of meaningful information in high dimensions [35]. However, as noted before, searching for

appropriate features can be a very expensive task [35]. Since it is always possible to find a subspace for any point where it is an outlier [37] subspace methods lend themselves by necessity to ensemble methods [35]. Aggarwal [35] gives broad categories for subspace methods: rarity-based, where the computationally costly algorithm tries to find few subspaces that have differing underlying distributions; unbiased methods, where the algorithm samples subspaces in unbiased way (these methods don't seem to work generally); and aggregation-based, where the algorithm attempts to combine various statistics to find appropriate subspaces.

4.4.1 Subspace outlier detection

One possible subspace based anomaly detection algorithm is called Subspace Outlier Detection (SOD). Broadly, the algorithm selects reference points and creates a minimal hyperplane. Outliers are detected in dimensions where the reference points have too small variance (meaning those features don't describe reference points enough), and then the distance to the hyperplane is calculated for a score. For reference points authors suggest using nearest neighbors to find shared nodes. Variance is calculated using the mean vector of reference points. Hyperplane is defined by mean vector μ and so called subspace defining vector v, where features irrelevant to the test get zeroed out. Outlierness is then calculated with distance to the mean vector masked with the subspace defining vector. Finally this algorithm is tested against other anomaly detection algorithms on synthetic and real data. Although authors find that for synthetic data, their method is the best based on ROC curves, on real data scores and outputs are similar to other methods. This method detects and considers local outliers and local feature correlations. [38] However, it suffers from computational demand when selecting appropriate reference points [32].

4.4.2 Global–Local Outliers in SubSpaces

Van Stein, Van Leeuwen, and Bäck [40] argue that many density-based algorithms perform well because:

1. they consider local neighbourhoods of data samples;

2. and consider feature subspaces.

Inspired by this they attempt to create an algorithm for complex and high dimensional data by searching local outliers within subspaces. Targeted data is specifically "a mixture of high-dimensional data points drawn from different data distributions". [40]

Gloss algorithm 2 integrates two parts into it. The subspace selection is done with HiCS that uses a statistical test to evaluate causality in randomly selected slices. The distance evaluation is done with LoOP that calculates the local distance within a subset modulated by various error and bias constants. [40]

Ultimately, Gloss algorithm outperforms state-of-the-art one experiments with both synthetic and real-world data. It successfully discovers local outliers that would otherwise be left undetected by other algorithms and was applied as a case study in the real world with manufacturing. Computationally it is also more efficient than its predecessors. [40]

Listing 2 GLOSS algorithm [40]

INPUT data D, subset size k
OUTPUT outlier probability scores p
$\mathcal{F} = \text{SubspaceSearch}(D)$
FOR EACH $d \in D$ DO
    $G_d = NN_k(d)$ # calculate the global neighbourhood
END FOR
FOR EACH $d \in D$ DO
    FOR EACH $F \in \mathcal{F}$ DO
        $\sigma(d_F, G_d) = \sqrt{\frac{\sum_{s \in G_d} \text{dist}(d_F, s_F)^2}{\|G_d\|}}$
        $\text{pdist}(\lambda, d_F, G_d) = \lambda \cdot \sigma(d_F, G_d)$
        $\text{PGLOF}_{\lambda, G_d}(d_F) = \frac{\text{pdist}(\lambda, d_F, G_d)}{\mathbb{E}_{s \in G_d}[\text{pdist}(\lambda, s, G_d)]} - 1$
        $p_{F,k}(d) = \max\left\{0, \text{erf}\left(\frac{\text{PGLOF}_{\lambda, G_d}(d_F)}{n \cdot \text{PGLOF} \cdot \sqrt{2}}\right)\right\}$
    END FOR
END FOR
return p

5 Adversarial theory and evidence

With such a large presence in the literature, it is important to understand adversarial examples. However, this task is difficult with rapid appearance and disappearance of state-of-the-art defensive and offensive techniques with various motivations and differing evidence. Each new defense and attack is usually "proven" with some benchmark or a theoretical formalization, but they sometimes contradict and usually don't overlap.

The research around adversarial examples is very fragmented due to a couple of reasons: firstly for some reason every new offensive or defensive method requires to publish some amount of theoretical reasoning and (handpicked) empirical evidence proven with small toy models and small flawed datasets, such as MNIST or CIFAR10; secondly there is no accepted tests or metrics to evaluate offensive and defensive metrics; thirdly new offensive and defensive methods don't attempt to evaluate contradicting evidence, instead only small portion of previous and most relevant work is cited.

Although no unified theoretical framework has been created, there are some broad buckets all theories fall into. These are model based and data based theories. Model based ones focus on architecture, loss functions, training, etc. and data based focus on inherent properties of datasets. Another axis of categorization divides theories into following groups: decision boundary based theories, manifold based theories, dimensionality based theories and other theories. Decision boundaries are

defined as hyperplanes that separate different classes of a dataset, and data manifolds are defined as a lower dimensional object embedded into high dimensional space. Some theories will overlap, but this kind of broad categorization would allow for more efficient and targeted research.

5.1 Model based theories

5.1.1 Architecture

Various architectural decisions might also generate the adversarial vulnerability. The clearest example is the robustness comparisons of vision transformers and convolutional neural networks [41]. Broadly adversarial robustness can be increased with max pooling, average pooling, separable convolution, dilated convolution, and skip connection. However, there is some evidence that skip connections in addition to allowing the creation of deeper models by preserving low-level features and preventing vanishing gradients, also increase black-box adversarial transferability due to back-propagation of gradients via the skip connections. Also increasing width and depth doesn't necessarily increase the robustness, but reducing width and depth almost always increases the robustness. This however might be a side effect of increased connection density, not decreased linearity. Density of connections have been found to be useful in increasing adversarial robustness. Concluding statements about activation functions cannot be made, beyond the broad statements such as "smooth activation functions with low curvature can behave like regularizers". Other more or less vague architectural insights were: LeakyReLU is better than ReLU, and activation functions behave differently in adversarial training. Clearest thing is that ReLU is not robust theoretically or empirically, because in theory it allows adversarial perturbations in proportion to dimensionality, and in practice with small toy models it is possible to create adversarial examples in a single step of gradient descent. [42]

Various architectural decisions might also generate the adversarial vulnerability. The clearest example is the robustness comparisons of vision transformers and convolutional neural networks [41]. Broadly adversarial robustness can be increased with max pooling, average pooling, separable convolution, dilated convolution, and skip connection. However, there is some evidence that skip connections in addition to allowing the creation of deeper models by preserving low-level features and preventing vanishing gradients, also increase black-box adversarial transferability due to back-propagation of gradients via the skip connections. Also increasing width and depth doesn't necessarily increase the robustness, but reducing width and depth almost always increases the robustness. This however might be a side effect of increased connection density, not decreased linearity. Density of connections have been found to be useful in increasing adversarial robustness. Concluding statements about activation functions cannot be made, beyond the broad statements such as "smooth activation functions with low curvature can behave like regularizers". Other more or less vague architectural insights were: LeakyReLU is better than ReLU, and activation functions behave differently in adversarial training. Clearest thing is that ReLU is not robust theoretically or empirically, because in theory it allows adversarial perturbations in proportion to dimensionality, and in practice with small toy models it is possible to create adversarial examples in a single step of gradient descent. [42]

If adversarial vulnerability was a result of a bias in model architecture then the inductive biases in architecture should lead to different results with identical data and training setup. Initial tests by Naseer, Ranasinghe, Khan, *et al.* [41] measured the performance of transformers against severe occlusions, domain shifts, spatial permutations, adversarial and natural perturbations. They reported that transformer models were significantly more robust compared CNN models due to the bias for shapes instead of textures [41]. However, Bai, Mei, Yuille, *et al.* [43] claim that both transformers and CNNs have similar robustness levels to adversarial

perturbations when tests are adjusted for parameter count, training data, number of epochs, and data augmentation strategies. Interestingly, transformers outperform CNNs in out-of-distribution samples due to the self-attention [43]. This highlights the fact that even if biases in both attention and convolution didn't have different adversarial robustness, there might be some other architectural decisions that could increase the adversarial accuracy. We can't dismiss the data perspective outright, however, failure to elevate either architecture with identical data goes against it. Another point for [43] is the fact that the authors used various ImageNet datasets and large established architectures, for example ResNet and vision transformers (ViT).

Previous discussion could also be extended to other places, for example with the optimizers. Since optimizers are the core algorithm that explores data landscapes in intimate detail, the effects of data on adversarial robustness should be clearly visible. Wang, Liu, Mišić, *et al.* [44] shows that optimizers such as SGD (stochastic gradient descent), RMS, AdaDelta and Adam have different adversarial robustness levels. All optimizers converge to some specific value, interestinly for both SGD and RMS the clean accuracy grows slower than the adversarial accuracy. The gap between clean and adversarial accuracy in the final results are similar for all optimizers (around 15 %). [44] Allthough impact of optimizers is small and bias in them didn't elevate adversarial elements of data, this conclusion should be taken with a grain of salt, since Wang, Liu, Mišić, *et al.* [44] used the MNIST dataset. Interestingly, Min and Vidal [45] mentions that inductive bias in gradient-based learning algorithms generalize well to small inter-cluster correlation, but fall to adversarial attacks with small radius. This would highlight a couple of things. Firstly, no non-gradient-based learning algorithms were experimented on, and secondly, some bias in learning algorithms transfers to adversarial vulnerability.

5.1.2 Linearity hypothesis

Simplest theory is the so-called "linearity hypothesis". It states that adversarial examples exploit high dimensional linearity of neural networks. Hypothesis claims that dot product between weights and perturbations can lead to large output changes because infinitesimal changes in the input add up. [22] The vulnerability gets slightly magnified with the scale of linear parameters and reduced with normalized linear parameters [46]. The scale of add-up is directly proportional to the dimensions used in the computations. This theory is attractive because it is simple and easy to verify, and furthermore an attack called FGSM was created based on it. Simplicity also allows for some strong statements, such as: adversarial examples occur in broad and contiguous subspaces, not if small discrete pockets. This statement appears due to the fact that adversarial perturbations need only broad directions, not precise steps. This would explain the abundance of adversarial examples. [22]

Literature references two different and contradicting pieces of evidence. First is by Li, Fan, Ganz, *et al.* [46], who claim that linear neural network compared to probabilistic ones absolutely suffer more from adversarial examples, yet confidence remains almost constant. On the other hand Tanay and Griffin [47] claim linearity and dimensionality is not a problem when activations get normalized, but the complexity of data is the problem.

5.1.3 Evolutionary stalling hypothesis

Cross-entropy loss calculates the distance between ground truth and predicted labels [42]. It is used in machine learning due to theoretical backing of Kullback-Leibler divergence. It has been especially successful with neural networks classifiers. However, in theory this loss function is suboptimal under couple conditions. First training points must lie in lower dimensional affine subspace and second classes must be linearly separable. Practically gradient descent will do the minimum amount of work

and the decision boundary will not maximize distance or the margin in full dimensions. With small margins, any training point assigned with small perturbation could cross the boundary with ease. [48] This issue likely follows other loss functions, because loss functions move into the directions that are not penalized [42], [48].

The sentiment of the last statement is packed in a generalized way into the so-called "evolutionary stalling hypothesis". It claims that training samples stop contributing to the weight updates once they are correctly classified. Lack of updates leads to decision boundaries that are very closely packed around sample clusters and let small perturbations cross the boundary with ease. [1], [42]

5.2 Data based theories

Finally, adversarial examples could be the result of some flawed property in the data. Xiong, Tegegn, Sarin, *et al.* [49] proposed eight categories to separate data: the number of samples, data dimensionality, distribution, density, concentration, separation, label quality, and domain-specific properties. Difference of training samples required to achieve certain performance between robust generalization and standard generalization is called sample complexity gap. Ideally the gap is zero. However, this gap is dependent on data distributions and data dimensionality. For example data generated with Bernoulli distribution requires less training samples compared to data generated with Gaussian distribution. Generally distributions can be separated based on their variance, mean and symmetry. Most robust features are with smallest meaningful variances. However, vulnerability can be caused if means are close to each other, or variance in the true population is larger than the variance in the training data. Finally, it is preferred that distribution is symmetric. It is important to note that uniform distributions don't follow the same rules. To train a robust classifier on uniform data, it is a requirement for data to lie in a unit cube

rather than in a unit sphere. [49]

As for dimensionality, standard generalization requires a constant amount of samples, but robust generalization will require the number of samples directly proportional to the data dimensionality. This was further proven by the fact that MNIST requires less samples compared to the dataset with greater dimensionality, such as CIFAR-100. Dimensionality problems themselves might be a result of how volume behaves in high dimensions or the difference between natural dimensionality and intrinsic dimensionality. [49] This connection to dimensionality transfers to both adversarial and natural training [42]. More details about high and low dimensional data is discussed in Section 4.3.

Density of data measures the number of samples in some local neighbourhood. In simple cases, this can be described with probability density function or probability mass function. Broadly it can be said that the higher feature density in training data leads to higher robustness and imbalance across feature densities will lead to adversarial vulnerability. [49] Furthermore, most transferable adversarial examples lie in low density regions, and no amount of adversarial training will increase robustness [42], [49]. Separation refers to distances between classes. Intuitively, the performance of a classifier will increase as the separation increases. However, separation might occur in totality or just in some hidden representations. As such, lower bounds for adversarial risk can be calculated for certain separation. [49]

Concentration refers to the "concentration of measure" principle that informally measures the probability the distance of some points under some distance budgets to the chosen region. In adversarial examples the probability function would be an error rate, an adversarial perturbation would be the budget, and a chosen region would be the space around the decision boundary. High concentration would mean highly vulnerable and this effect is further magnified with high dimensional data. And so, no matter the distribution of data, if it is concentrated, adversarial vulnerability

will follow. It is also noted that concentration especially does not explain all the adversarial examples since there is a difference between theoretical and practical robustness. [49]

Label quality refers to the correctness of the labels. Other terms in the same context are label noise and informativeness. Because incorrect labels won't match the data manifold model will be forced to overfit and generate fragile decision boundaries. Even suboptimal labels will contribute to overfitting. More importantly vulnerability will be proportional to the number of classes so having more detailed classes will increase robustness. These decision boundaries are ripe for adversarial exploitation. This idea can be further extended to adversarial training, where we assume that adding perturbation to the data reduces informativeness of labels. This practically shows the limit of such training. [49]

Domain specific properties in our case mean various properties of image data, for example frequency is the rate of change of pixel intensities across the image. Generally it is thought that humans evaluate images on low frequency properties and machine learning models evaluate them on both high and low frequency properties. Various architectural biases and modification to training can change the focus. [49] However, since the model targets high performance, it can inadvertently focus on local statistics such as textures. This allows models to generalize across augmentations, but fails to generalize to the underlying world or learn domain invariant features. [50] This focus leads to vulnerabilities in some frequency range [49].

Even with all these categories we can broadly state a couple points that all data categories share. Firstly the greater the difference between real world and training data distribution, the greater the vulnerability. Paradoxically there might be a difficulty discerning such gaps but this point is helpful when designing defenses. Secondly the complexity of data and the number of samples is a difficult problem to deal with. The addition of adversarial training or data augmentations have to be

evaluated against the risk of overfitting or increased computational costs on a case by case basis. [42]

In addition to section 4.3 there are other evidence pointing to a connection between adversarial examples and high dimensional data. For example Godfrey, Kvinge, Bishoff, *et al.* [51] constructed a theoretical upper bound, which was later verified with adversarial ImageNet data on ResNet50 models. However, the framework was reverse engineered, used one type of distribution shift, and derived from linear binary classifiers. [51] More theoretical reasoning can be seen with Chattopadhyay, Chattopadhyay, Gupta, *et al.* [52] where authors show that as the volume of reduced object divided by volume of default object, the result approaches zero as dimensions approach infinity. This was verified by testing models on SVD compressed MNIST and CIFAR-10 images. Note that compression reduces **intrinsic** dimensions. [52] In opposition of simple "high dimension are bad"-statement, Shafahi, Huang, Studer, *et al.* [53] claim adversarial examples are not the side effect of high dimensionality, rather they come from concentration of upper bounds of object classes. For example MNIST pixels are highly correlated with each other and so images concentrate in low dimensional manifolds resulting in appropriate upper bound. [53]

5.3 Decision boundary hypotheses

Various theories also attempt to define and formalize behaviour around decision boundaries more precisely. A boundary tilting hypothesis attempts to extend the linearity hypothesis and states that the sampled data manifold is vulnerable when the decision boundary extends in parallel or stays close to it. Formally this framework allowed authors to define adversarial examples and their strength as an angle between weight vector of studied classifier and weight vector of nearest centroid

classifier [1]. [42] Framework suggested to use regularization to mitigate vulnerability [42]. However regularization has not been found to be an effective defensive method.

Other lens to evaluate adversarial examples and decision boundary hypotheses is to evaluate loss landscape and its peaks and valleys. Yu, Qin, Liu, *et al.* [54] found out that adversarial examples happen due to uneven local neighbourhoods and decision boundaries. This vulnerability is intrinsic either to neural networks, gradient descent or data. [54]

A theory called a decision boundary similarity on the other hand proposes distances to decision boundaries were bigger than distances to random or adversarial directions. Interboundary distances (difference in decision boundaries between models) were also small and directly comparable to transferability of adversarial examples. This was attributed to the presence of proportionally small data manifolds in large vector space. [55] This framework is supported by the fact that attacks engineered to exploit decision boundaries in this way are generally successful and transferable. [42]

Opposing sentiment was reached by Jetley, Lord, and Torr [56], who are claim that adversarial vectors are in the same direction as directions important for classification performance. Attacks MUST exploit genuine decision boundaries to keep norms low. Any defensive method discarding directions (for example compression) will lead to decreased accuracy and increased robustness. [56]

Also there have been attempts to mathematically standardize the relationship between the surface of the decision boundary and robustness of a model. Under the random noise regime it was shown that the small curvature made robustness proportional to the dimensionality. [42] This implies that robustness to noise can be achieved even in high dimensions [57]. Adversarial examples are assumed to point to highly curved regions leading to greater transferability (if models share sensitive

[1] A classifier that assigns label based on nearest mean of class samples

regions) and easy crossings of decision boundaries [42]. However, there are a few problems: theory bases its conclusions on random or semi-random noise regimes, not adversarial regimes; if small curvatures were the solution, then regularization and skip connections would be THE solutions to adversarial robustness. Local visualization of decision boundaries showed that naturally trained models had sharp peaks and large slopes, and adversarial trained models had high plateaus [42] showing that curvature could be measured and encouraged. Practical proposal was to use Kullback-Leibler divergence between the clean prediction probability distribution and the adversarial prediction probability distribution to measure model robustness [42].

There are also difficulties when evaluating the geometry of vector space due to high dimensionality of internal representations and output labels, and interpolative nature of real number and linear algebra. Constant shifting from linear computations to non-linear transformations don't make any theory particularly stable as most of them are "proven" on a simple linear model at the worst, or on a toy MNIST classifier at the best. Switches from global lens to local tangent space also obfuscate shared elements of these theories. For example, decision boundary similarity shares common effects with the curvature of decision boundaries. Ultimately, it is hard to evaluate any decision boundary based theory because of fragmented standards, no shared measurement metrics of spatial properties, sometimes contradictory properties, and similar results.

5.4 Manifold hypotheses

Framework constructed by Khoury and Hadfield-Menell [58] claims the active subsection plays a very important role in adversarial vulnerability. By construction of the framework, authors showed that "no single decision boundary can be optimally robust in all norms". And because decision boundaries and manifolds were defined

in a certain way, authors also were able to prove Voronoi cells are elongated in the directions normal to manifolds at dense regions, the nearest neighbour classifiers was provably robust. Ineffectiveness of adversarial training came from the fact that as intrinsic dimensions rose, samples needed for training also rose exponentially. [58]

Lyu, Huang, and Liang [59] presented a framework to support gradient regularization methods to effectively penalize the gradient of loss function with respect to inputs. By mathematically reversing the gradient, it was shown that adversarial examples erases correlation of correct components, because Jacobian's model local space and by extension manifolds. [59]

5.4.1 Feature based hypotheses

Features are defined as patterns derived from the training data [60]. A popular theoretical branch claims quality and robustness of these features is the main reason for the adversarial vulnerabilities. For example Yin, Gontijo Lopes, Shlens, *et al.* [50] has shown that computer vision CNN models learn repeating and locally correlating textures at the cost of shape recognition. On the other hand Ilyas, Santurkar, Tsipras, *et al.* [60] shows that it is possible to separate features based on how much they contribute to the accuracy. Models trained on separated robust features show better adversarial robustness [60]. Similarly adversarial perturbations applied to features that contribute most to the accuracy lead to greatest decay in performance [61].

Actual formal definition for features and their location in the manifold requires more work. However, Han, Lin, Shen, *et al.* [42] concludes that any discussion about lacking features ties back to difference in distributions between real world and training data. And this calls for more samples, better loss functions and improved biases. [42]

5.5 Other evidence

Broad and practical evidence on how adversarial examples change the behaviour of models are summarized by Han, Lin, Shen, *et al.* [42]. By focusing on individual neurons and synapses it has been found that clean samples activate neurons with strong correlations with the perceptible human attributes and adversarial samples exploit neurons correlated imperceptible attributes. Expanded focus to whole intermediate layers it was shown that adversarial examples were consistently more active compared to clean samples. These neuron activations were reasoned to be redundant. On the other hand, by focusing on activation paths of training and adversarial samples it has been noticed that different classes activate different paths. Also, it was shown that adversarial samples activated unconventional and inactive paths. [42]

On interesting theory is from Tian, Zhou, Li, *et al.* [62], who note that in computer vision adversarial examples are sensitive to some spatial fluctuations, and based on that they construct a separate detector to weed out adversarial examples. This vulnerability of adversarial examples happen because adversarial directions usually point to highly curved regions of the decision boundary. [62]

Singular value decomposition generalizes the concept of eigen decomposition into non square matrix form, where a singular value describes the weight or importance given to a specific basis vector under some linear transformation [63]. If adversarial examples were connected to the embedded low dimensional manifolds, or to the bias in the architecture and training, then it could be measured from the distribution of singular values. By assuming that low dimensional manifolds would exploit few neurons at the time, or a biased architecture forces reasoning through a small set of important neurons or paths, we would expect to see greater decay of singular values in non robust neural networks. Mathematically this means that few basis vectors are more meaningful than the rest of the basis. This can be seen with a

defensive method called Singular Value Suppression (SVS) that increases adversarial robustness by reducing the decay in singular values [64]. However, we note that practical improvements were slight compared to defensive methods and authors of Zhao, Zhan, Duan, *et al.* [64] cautions against extreme conclusions, because there are signs of overfitting of adversarial noise caused by suppression of certain singular values. Another piece of evidence is shown in Moosavi-Dezfooli, Fawzi, Fawzi, *et al.* [23], the singular values were recovered from normals of low dimensional manifolds. The presence of adversarially vulnerable correlations and redundancies in the decision boundaries were connected to quickly decaying singular values [23].

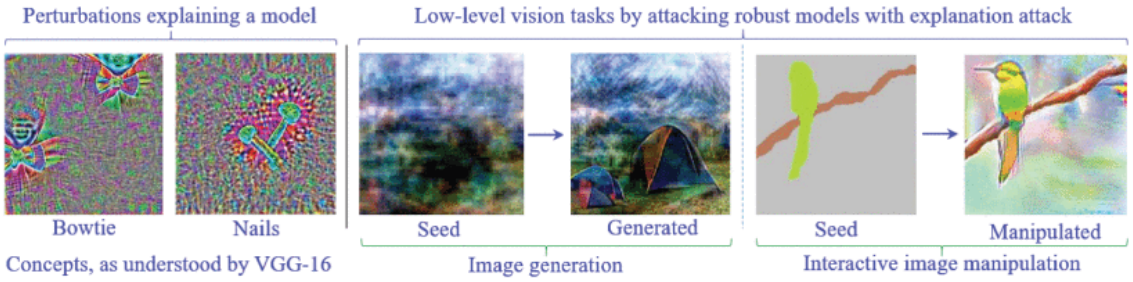


Figure 5.1: Human-defined semantic concepts refined from adversarial perturbations [1]. Figure 16 by Akhtar, Mian, Kardan, *et al.* [1] is licensed under CC BY 4.0.

Other interesting evidence lies in the geometric patterns of adversarial perturbations (Figure 5.1 and figure 5.2). Akhtar, Jalwana, Bennamoun, *et al.* [65] show and prove the emergence of human defined concepts in those perturbations. To refine these patterns, it is necessary to construct strict perturbation from a one class to another. This fact reveals the meaningfulness of decision boundaries and that adversarial perturbation exploit those geometric correlations. [65] A remaining questions are, if pattern quality can be further manipulated with changes in data or architecture, and does this refinement apply to all possible adversarial perturbations, for example perturbations generated by UAP might hold some repeating patterns but most of it is chaotic (Figure 5.3).

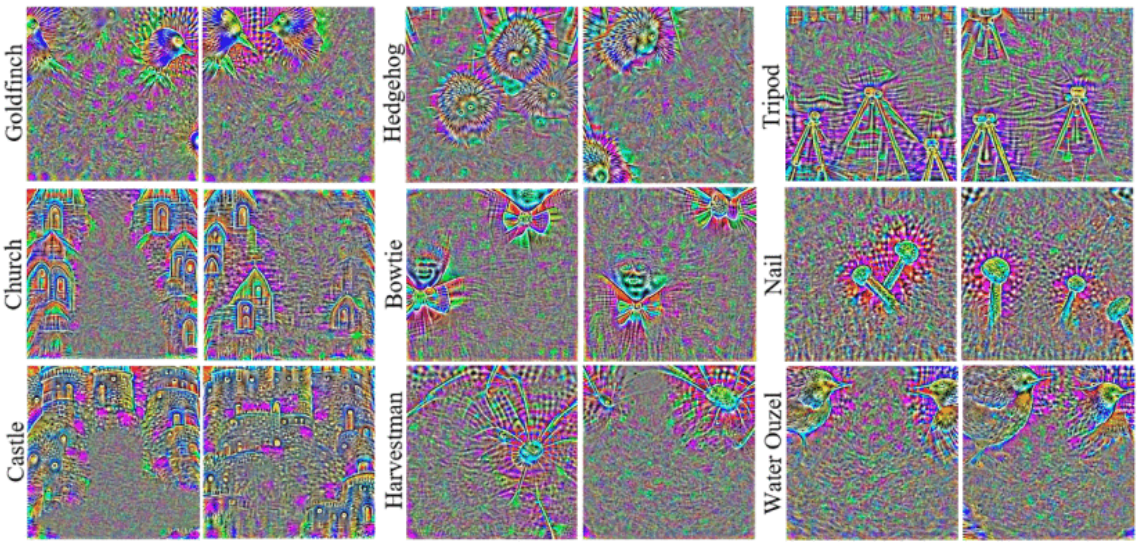


Figure 5.2: By accumulating the gradient based perturbations with some objective class it is possible to refine salient patterns. Figure 8 from Akhtar, Jalwana, Bennamoun, *et al.* [65], © 2021 IEEE.

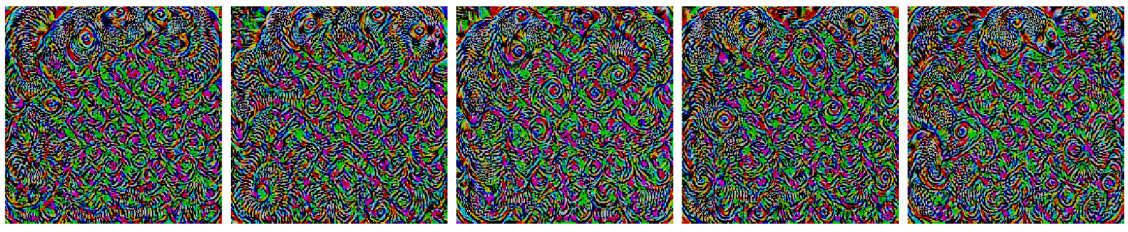


Figure 5.3: Five perturbations generated with Universal Adversarial Perturbation algorithm. Figure 5 from Moosavi-Dezfooli, Fawzi, Fawzi, *et al.* [23], © 2017 IEEE.

6 Conclusions

We will conclude the thesis by answering the research question in direct manner and providing some future research directions.

1. **What are adversarial examples?** Adversarial examples are optimized or generated perturbations that change the final output of any machine learning model.

2. **What are the theoretical explanations for adversarial examples?** Theoretical explanations chalk up adversarial examples to weakly sampled data (final manifolds are not contiguous nor smooth), architectural bias, high dimensions and shallow learning algorithms (decision boundaries are left to maximize accuracy not robustness).

3. **Does theory provide any insights or possible paths to adversarial robustness?** Outlier detection contains a lot of insights and discussion on high dimensionality, architectural bias has been explored through the comparisons of both transformers and convolutional neural networks. Theories discussing decision boundaries highlight the importance of maximizing angles/distances to manifolds, and theories discussing manifolds highlight the importance of appropriate data.

Based on the theory and empirical evidence we have explored, we list out some future research directions for achieving adversarial robustness:

1. Simplest way would be an architectural change that forces a neural network to operate in smaller subspaces. Similar to the SOD algorithm (Section 4.4.1), there should be a module that attempts to pick appropriate subspace/hyperplane, and a module/s that operate data in parallel, small, chosen subspaces. Subspaces could be chosen by the correlations gradient descent finds or by exploiting the attention mechanism to dynamically search for the best subspaces. Important part also would be some mechanism/module that periodically integrates these subspaces into one cohesive domain. We imagine this will work as a bottleneck for the neural networks and would provide greater robustness, although accuracy probably will suffer.

2. Some have hypothesized that adversarial vulnerability is directly dependent on features neural networks learn (Section 5.4.1). For example dividing features into non-robust and robust [60], or using Fourier perspective and dividing features into high frequency and low frequency features [50]. This might be the side effect of the local nature of neurons, for example in regression tasks, individual neurons reserve themselves a slice of time, or in image classification tasks neurons reserve themselves spatial regions [29], [30]. From these lenses we see that neural networks naturally gravitate to exploitation of every bit of information available. Inspired by the nature of Fourier transform which is to transform problem domain from temporal space to frequency space, we propose similarly to transform problem domain from probabilistic correlation space to alternative abstract domain space. Although this already happens in higher layers, shown by Bau, Zhu, Strobelt, *et al.* [11], it is clear that statistical correlation alone is not enough to guard against perturbations. Practically we propose the use of procedural domain and integration of explicitly causal relationships in to the network, for example with mechanisms similar to interaction in statistics.

3. Final proposal would be the changing or replacement of the inner product. Most important feature of a new inner product would be its invariance to increasing dimensionality, so standard deviation of randomly sampled vectors would increase preferably at the same rate as dimensionality, or upper and lower bounds of inner products will not grow with increased dimensionality. However, the most important property it should have is the ability to propagate gradients. Remaining questions are the unclear change in the nature of neural networks and deep learning, unexplored drawbacks of the other inner product or some other non-linear operation, and is the benefit of theoretical grounding of linear algebra and by extension dot products too large.

References

[1] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in adversarial attacks and defenses in computer vision: A survey", *IEEE Access*, vol. 9, pp. 155 161–155 196, 2021.

[2] L. Dhamija and U. Bansal, "How to defend and secure deep learning models against adversarial attacks in computer vision: A systematic review", *New Generation Computing*, vol. 42, no. 5, pp. 1165–1235, 2024.

[3] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning", *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[4] C. Li, H. Wang, W. Yao, and T. Jiang, "Adversarial attacks in computer vision: A survey", *Journal of Membrane Computing*, vol. 6, no. 2, pp. 130–147, 2024.

[5] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review", *Computational intelligence and neuroscience*, vol. 2018, no. 1, p. 7 068 349, 2018.

[6] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines", *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[7] K. Cho, T. Raiko, and A. Ilin, "Enhanced gradient for training restricted boltzmann machines", *Neural computation*, vol. 25, no. 3, pp. 805–831, 2013.

[8] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[9] M. F. Bear, B. W. Connors, and M. A. Paradiso, *Neuroscience : exploring the brain*, eng, Enhanced fourth edition. Burlington, MA: Jones & Bartlett Learning, 2016, ISBN: 9781284618747.

[10] D. Bhatt, C. Patel, H. Talsania, *et al.*, "Cnn variants for computer vision: History, architecture, application, challenges and future scope", *Electronics*, vol. 10, no. 20, p. 2470, 2021.

[11] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba, "Understanding the role of individual units in a deep neural network", *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 071–30 078, 2020.

[12] R. Tombe and S. Viriri, "Effective processing of convolutional neural networks for computer vision: A tutorial and survey", *IETE technical review*, vol. 39, no. 1, pp. 49–62, 2022.

[13] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision", *Artificial Intelligence Review*, vol. 57, no. 4, p. 99, 2024.

[14] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[15] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey", *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[16] K. Han, Y. Wang, H. Chen, *et al.*, "A survey on visual transformer", *arXiv preprint arXiv:2012.12556*, 2020.

[17] M.-H. Guo, T.-X. Xu, J.-J. Liu, *et al.*, "Attention mechanisms in computer vision: A survey", *Computational visual media*, vol. 8, no. 3, pp. 331–368, 2022.

[18] Y. Xu, H. Wei, M. Lin, *et al.*, "Transformers in computational visual media: A survey", *Computational Visual Media*, vol. 8, pp. 33–62, 2022.

[19] Z. Qian, K. Huang, Q.-F. Wang, and X.-Y. Zhang, "A survey of robust adversarial training in pattern recognition: Fundamental, theory, and methodologies", *Pattern Recognition*, vol. 131, p. 108 889, 2022.

[20] R. R. Wiyatno, A. Xu, O. Dia, and A. De Berker, "Adversarial examples in modern machine learning: A review", *arXiv preprint arXiv:1911.05268*, 2019.

[21] K. Mahmood, D. Gurevin, M. van Dijk, and P. H. Nguyen, "Beware the black-box: On the robustness of recent defenses to adversarial examples", *Entropy*, vol. 23, no. 10, p. 1359, 2021.

[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", *arXiv preprint arXiv:1412.6572*, 2014.

[23] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[24] J.-J. Zhu and J. Bento, "Generative adversarial active learning", *arXiv preprint arXiv:1702.07956*, 2017.

[25] M. Ducoffe and F. Precioso, "Adversarial active learning for deep networks: A margin based approach", *arXiv preprint arXiv:1802.09841*, 2018.

[26] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 501–509.

[27] B. Joe, S. J. Hwang, and I. Shin, "Learning to disentangle robust and vulnerable features for adversarial detection", *arXiv preprint arXiv:1909.04311*, 2019.

[28] B. Lindqvist and R. Izmailov, "Minimax defense against gradient-based adversarial attacks", *arXiv preprint arXiv:2002.01256*, 2020.

[29] A. Sivaram, L. Das, and V. Venkatasubramanian, "Hidden representations in deep neural networks: Part 1. classification problems", *Computers & Chemical Engineering*, vol. 134, p. 106 669, 2020.

[30] L. Das, A. Sivaram, and V. Venkatasubramanian, "Hidden representations in deep neural networks: Part 2. regression problems", *Computers & Chemical Engineering*, vol. 139, p. 106 895, 2020.

[31] S. Thudumu, P. Branch, J. Jin, and J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data", *Journal of big data*, vol. 7, pp. 1–30, 2020.

[32] H. Wang, M. J. Bah, and M. Hammad, "Progress in outlier detection techniques: A survey", *Ieee Access*, vol. 7, pp. 107 964–108 000, 2019.

[33] R. Foorthuis, "On the nature and types of anomalies: A review of deviations in data", *International journal of data science and analytics*, vol. 12, no. 4, pp. 297–331, 2021.

[34] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection", *Data mining and knowledge discovery*, vol. 28, pp. 190–237, 2014.

[35] C. C. Aggarwal, *Outlier analysis*, eng, Second edition. Cham, Switzerland: Springer, 2017, ISBN: 9783319475783.

[36] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples", *arXiv preprint arXiv:1702.06280*, 2017.

[37] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data", *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012.

[38] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier detection in axis-parallel subspaces of high dimensional data", in *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*, Springer, 2009, pp. 831–838.

[39] D. François, V. Wertz, and M. Verleysen, "The concentration of fractional distances", *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 873–886, 2007.

[40] B. Van Stein, M. Van Leeuwen, and T. Bäck, "Local subspace-based outlier detection using global neighbourhoods", in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016, pp. 1136–1142.

[41] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang, "Intriguing properties of vision transformers", *Advances in Neural Information Processing Systems*, vol. 34, pp. 23 296–23 308, 2021.

[42] S. Han, C. Lin, C. Shen, Q. Wang, and X. Guan, "Interpreting adversarial examples in deep learning: A review", *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–38, 2023.

[43] Y. Bai, J. Mei, A. L. Yuille, and C. Xie, "Are transformers more robust than cnns?", *Advances in neural information processing systems*, vol. 34, pp. 26 831–26 843, 2021.

[44] Y. Wang, J. Liu, J. Mišić, V. B. Mišić, S. Lv, and X. Chang, "Assessing optimizer impact on dnn model sensitivity to adversarial examples", *IEEE Access*, vol. 7, pp. 152 766–152 776, 2019.

[45] H. Min and R. Vidal, "Can implicit bias imply adversarial robustness?", in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24, Vienna, Austria: JMLR.org, 2024.

[46] H. Li, Y. Fan, F. Ganz, A. Yezzi, and P. Barnaghi, "Verifying the causes of adversarial examples", in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 6750–6757.

[47] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples", *arXiv preprint arXiv:1608.07690*, 2016.

[48] K. Nar, O. Ocal, S. S. Sastry, and K. Ramchandran, "Cross-entropy loss and low-rank features have responsibility for adversarial examples", *arXiv preprint arXiv:1901.08360*, 2019.

[49] P. Xiong, M. Tegegn, J. S. Sarin, S. Pal, and J. Rubin, "It is all about data: A survey on the effects of data on adversarial robustness", *ACM Computing Surveys*, vol. 56, no. 7, pp. 1–41, 2024.

[50] D. Yin, R. Gontijo Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision", *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[51] C. Godfrey, H. Kvinge, E. Bishoff, *et al.*, "How many dimensions are required to find an adversarial example?", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2353–2360.

[52] N. Chattopadhyay, A. Chattopadhyay, S. S. Gupta, and M. Kasper, "Curse of dimensionality in adversarial examples", in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–8.

[53] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein, "Are adversarial examples inevitable?", *arXiv preprint arXiv:1809.02104*, 2018.

[54] F. Yu, Z. Qin, C. Liu, L. Zhao, Y. Wang, and X. Chen, "Interpreting and evaluating neural network robustness", *arXiv preprint arXiv:1905.04270*, 2019.

[55] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples", *arXiv preprint arXiv:1704.03453*, 2017.

[56] S. Jetley, N. Lord, and P. Torr, "With friends like these, who needs adversaries?", *Advances in neural information processing systems*, vol. 31, 2018.

[57] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "Robustness of classifiers: From adversarial to random noise", *Advances in neural information processing systems*, vol. 29, 2016.

[58] M. Khoury and D. Hadfield-Menell, "On the geometry of adversarial examples", *arXiv preprint arXiv:1811.00525*, 2018.

[59] C. Lyu, K. Huang, and H.-N. Liang, "A unified gradient regularization family for adversarial examples", in *2015 IEEE international conference on data mining*, IEEE, 2015, pp. 301–309.

[60] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features", *Advances in neural information processing systems*, vol. 32, 2019.

[61] X. Liu, L. Li, X. Wang, and L. Hu, "Adversarial examples generated from sample subspace", *Computer Standards & Interfaces*, vol. 82, p. 103 634, 2022.

[62] J. Tian, J. Zhou, Y. Li, and J. Duan, "Detecting adversarial examples from sensitivity inconsistency of spatial-transform domain", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 9877–9885.

[63] D. Kalman, "A singularly valuable decomposition: The svd of a matrix", *The college mathematics journal*, vol. 27, no. 1, pp. 2–23, 1996.

[64] Z. Zhao, W. Zhan, H. Duan, and Y. Wu, "Study on adversarial robustness of deep learning models based on svd", chi, *Ji suan ji ke xue*, vol. 50, no. 10, pp. 362–, 2023, ISSN: 1002-137X.

[65] N. Akhtar, M. A. Jalwana, M. Bennamoun, and A. Mian, "Attack to fool and explain deep networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 5980–5995, 2021.