

Regularization and Robustness

Advanced Computer Vision

Niels Landwehr

Overview

- Introduction: Computer Vision
- Data, Models, Optimization
- Neural Networks and Automatic Differentiation
- Convolutional Architectures For Image Classification
- Metric Learning for Computer Vision
- Image Segmentation
- Object Detection
- **Regularization and Robustness**

Agenda Today

- Regularization and Data Augmentation
- Robustness and Adversarial Attacks

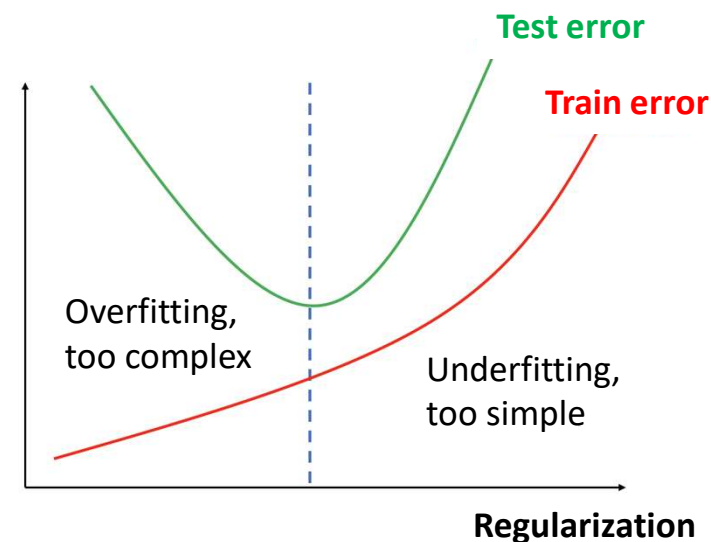
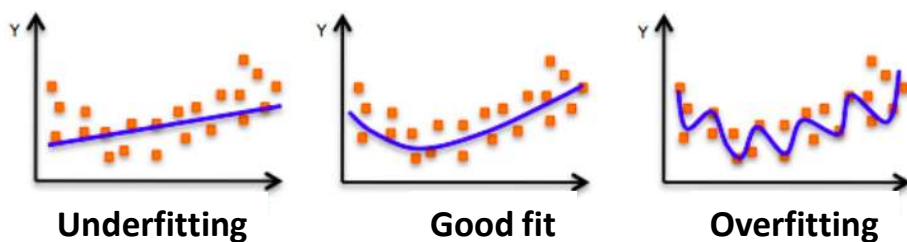
Agenda Today

- Regularization and Data Augmentation
- Robustness and Adversarial Attacks

Recap: Regularization and Overfitting

- Recap regularization: express preference for models beyond matching training data
 - Regularization makes it harder for model to perfectly fit training data
 - Avoid **overfitting**: regularized models (sometimes) generalize better
 - Mostly important for small data sets

Simple versus complex models



Regularization for Computer Vision

- Regularization a common theme in machine learning in general
 - e.g. norm regularizers for linear models, kernel machines, etc
 - e.g. priors in Bayesian learning
 - general principle that is relevant for all of machine learning/ statistics
- How do we regularize computer vision models?
 - General approaches: weight regularization, early stopping
 - Approaches specific to neural networks: dropout, batch normalization
 - Approaches specific to computer vision networks: visual data augmentation, cutout, mixup

Regularization for Computer Vision

- Regularization a common theme in machine learning in general
 - e.g. norm regularizers for linear models, kernel machines, etc
 - e.g. priors in Bayesian learning
 - general principle that is relevant for all of machine learning/ statistics
- How do we regularize computer vision models?
 - General approaches: weight regularization, early stopping
 - Approaches specific to neural networks: dropout, batch normalization
 - Approaches specific to computer vision networks: visual data augmentation, cutout, mixup

Recap: Weight Regularization

- Weight regularization: add a regularization term to loss function

$$L(\boldsymbol{\theta}) = \underbrace{\frac{1}{n} \sum_{i=1}^n \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)}_{\text{Loss: model predictions should match training data}} + \underbrace{\lambda R(\boldsymbol{\theta})}_{\text{Regularization: prevent model from doing too well on the training data, prefer simpler models}}$$

Loss: model predictions
should match training data

Regularization: prevent model from doing
too well on the training data, prefer simpler models

L2-regularizer $R(\boldsymbol{\theta}) = \sum_j \theta_j^2$

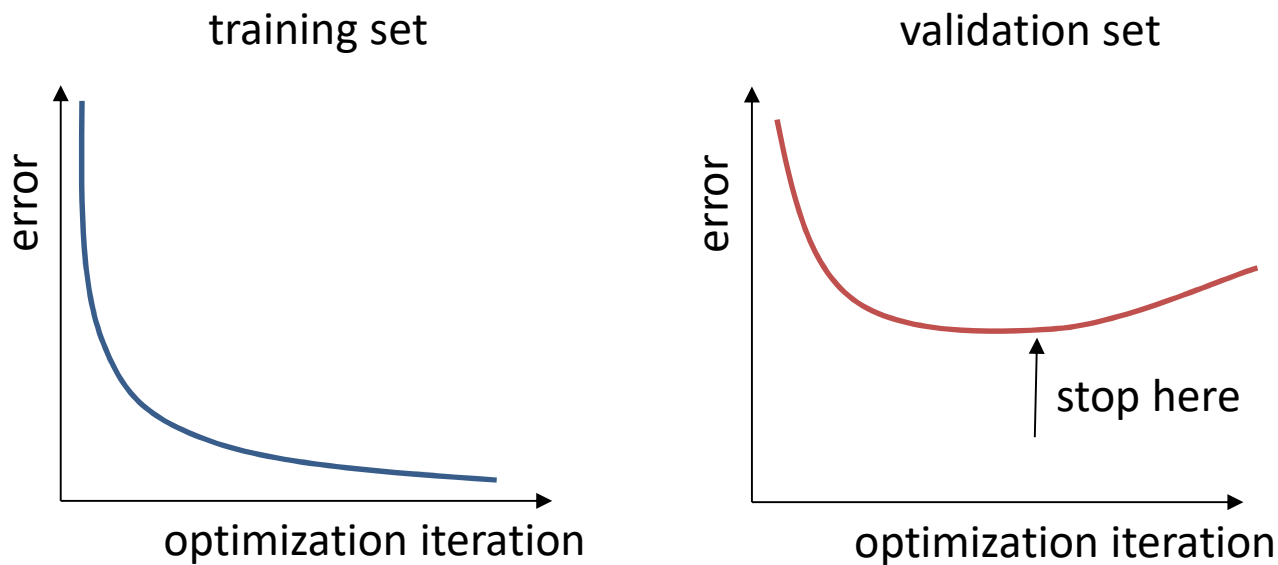
L1-regularizer $R(\boldsymbol{\theta}) = \sum_j |\theta_j|$

θ_j : any model parameter,
for CNNs all convolution
filters and possibly biases

- Strength of regularization is a hyperparameter that can be tuned

Early Stopping

- Simple practical regularization strategy: early stopping



- Split off validation set and stop training when validation error increases
- Note that for estimating future error of model, separate test set is needed

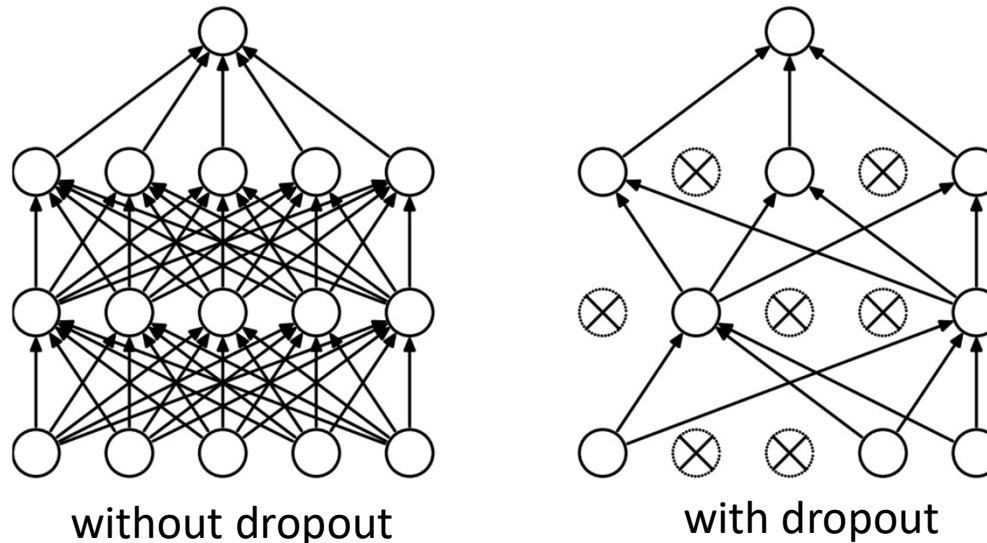


Regularization for Computer Vision

- Regularization a common theme in machine learning in general
 - e.g. norm regularizers for linear models, kernel machines, etc
 - e.g. priors in Bayesian learning
 - general principle that is relevant for all of machine learning/ statistics
- How do we regularize computer vision models?
 - General approaches: weight regularization, early stopping
 - Approaches specific to neural networks: dropout, batch normalization
 - Approaches specific to computer vision networks: visual data augmentation, cutout, mixup

Dropout

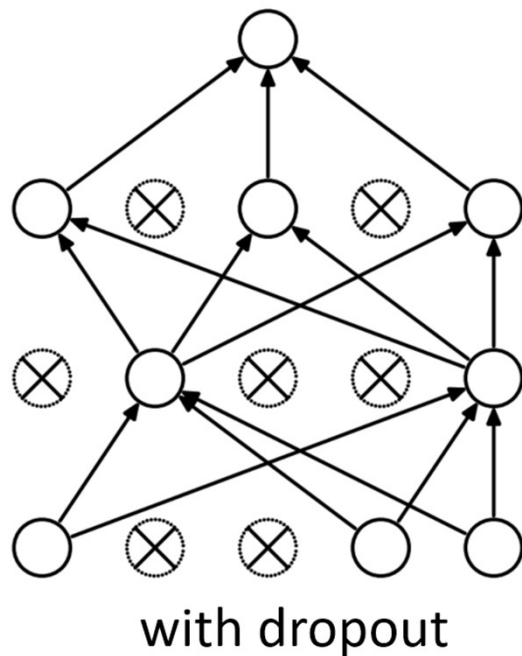
- **Dropout:** a widely used regularization method for general neural networks
- In each forward pass (and corresponding backpropagation) during training, randomly set some neurons to zero in compute graph



- Implemented as **dropout layer** with hyperparameter dropout probability (e.g. probability 0.5)

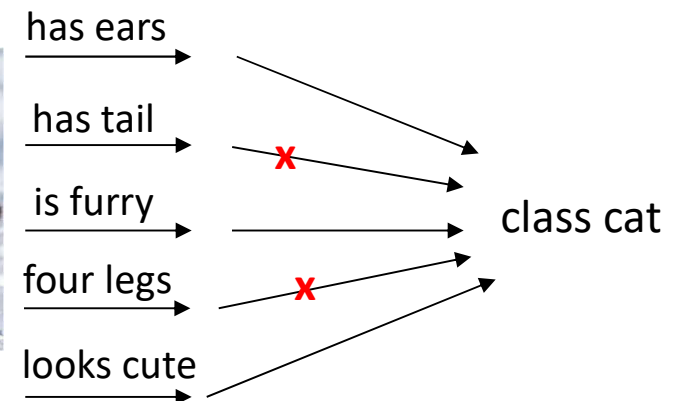
Dropout: Motivation

- What is the motivation behind dropout?



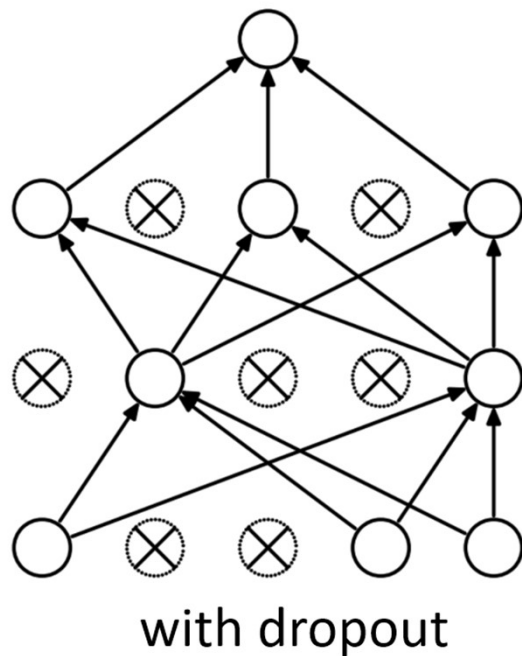
Network should learn a representation that is robust with respect to missing individual features

Also prevents co-adaptation of features (features that are always activated together and depend on each other)



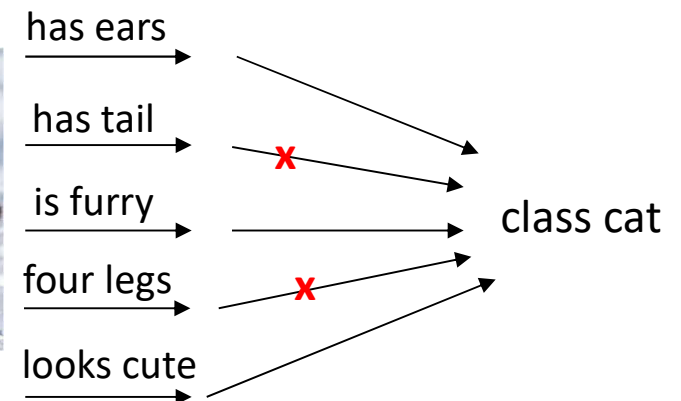
Dropout: Motivation

- What is the motivation behind dropout?



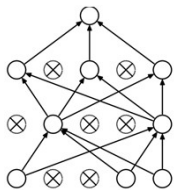
Another view on dropout:

Learning a model with dropout resembles learning an ensemble of models, as for each random dropout mask the model looks slightly different



Dropout at Test Time

- What do we do with a model trained with dropout at test time?
- Keeping dropout at test time would make the output random:



$$\mathbf{z} \sim p(\mathbf{z})$$

draw a random dropout mask \mathbf{z}

$$y = f_{\theta}(\mathbf{x}, \mathbf{z})$$

compute output based on input \mathbf{x} and mask \mathbf{z}

- Instead, use the following approach at test time:
 - For a node that has been dropped out with probability p at training time, compute its activation as usual and then multiply with $(1 - p)$
 - This computes something like an "expected value" of the activation of the node under the dropout scheme
 - With probability $(1 - p)$ the node was not dropped out and retains its usual activation
 - With probability p it was dropped out and the activation was set to zero

Batch Normalization as Regularizer

- **Recap: batch normalization** (i =index in batch, x, y = spatial positions in feature map, c = channel index, $\mathbf{z}[i, x, y, c]$ feature map before batch normalization)

$$\bar{\mathbf{z}}[i, x, y, c] = \alpha \frac{\mathbf{z}[i, x, y, c] - \mu_c}{\sigma_c} + \beta$$

↑
Learnable parameters

$$\sigma_c = \sqrt{\frac{1}{nlm} \sum_{i, x, y} (\mathbf{z}[i, x, y, c] - \mu_c)^2}$$
$$\mu_c = \frac{1}{nlm} \sum_{i, x, y} \mathbf{z}[i, x, y, c]$$

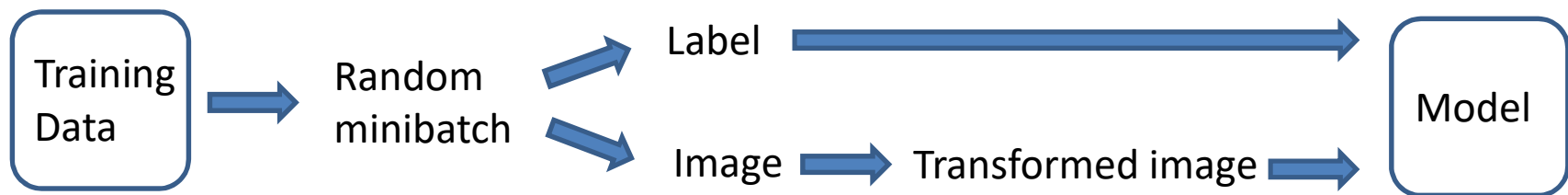
- It has been empirically observed that adding batch normalization to a network can have a regularizing effect and reduce overfitting
- One interpretation is that batch normalization adds variation/noise to the training in a similar way as dropout
 - Divides hidden units by standard deviation and subtracts mean from minibatch. Both are randomly fluctuating due to random minibatches
 - These sources of noise mean that every layer has to learn to be robust to a lot of variation in its input, similar as with dropout

Regularization for Computer Vision

- Regularization a common theme in machine learning in general
 - e.g. norm regularizers for linear models, kernel machines, etc
 - e.g. priors in Bayesian learning
 - general principle that is relevant for all of machine learning/ statistics
- How do we regularize computer vision models?
 - General approaches: weight regularization, early stopping
 - Approaches specific to neural networks: dropout, batch normalization
 - Approaches specific to computer vision networks: visual data augmentation, cutout, mixup

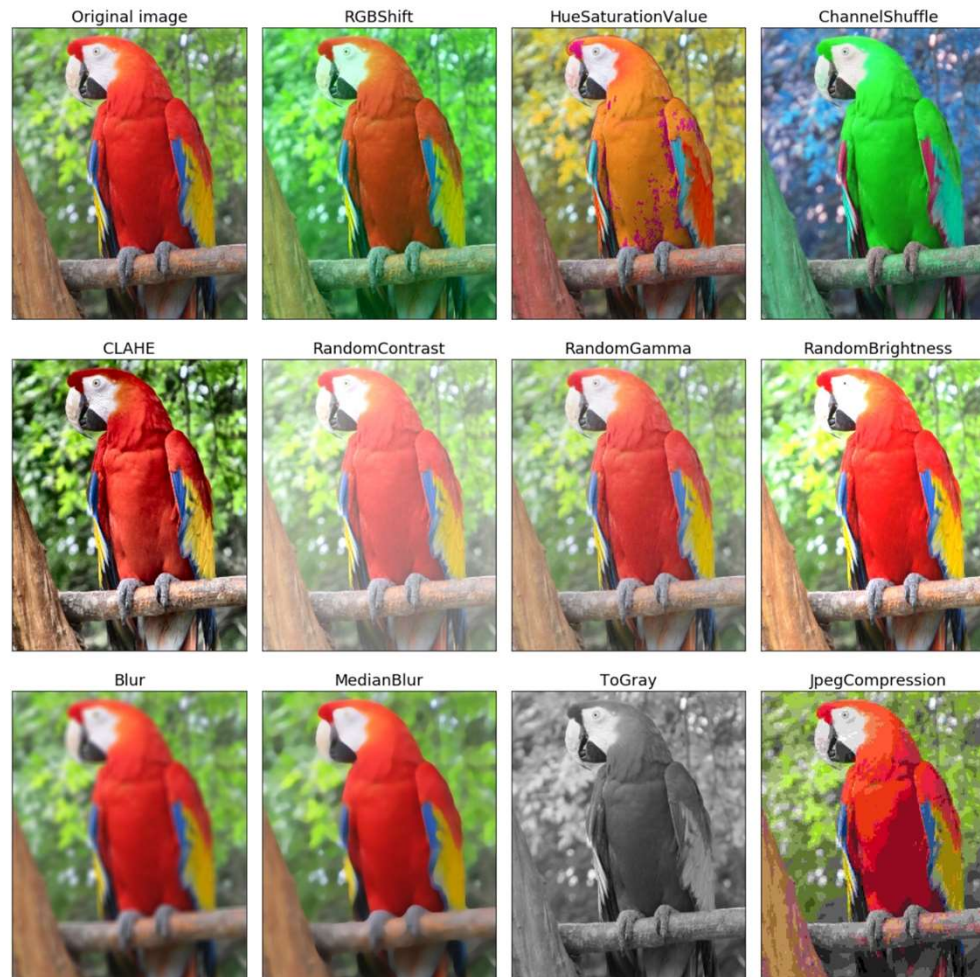
Motivation: Data Augmentation

- Amount of available data is a crucial determinant of success for machine learning approaches: the more data, the better the final model
- Collecting/labeling data can be expensive: data is a valuable „resource“
- **Data augmentation:** increasing the amount of training data by transformations of the original training instances
- For image data, this is particularly appealing, because there are many known transformations that do not change the label, or for which the label can also be transformed
- Typically transforming „on-the-fly“:



Data Augmentation

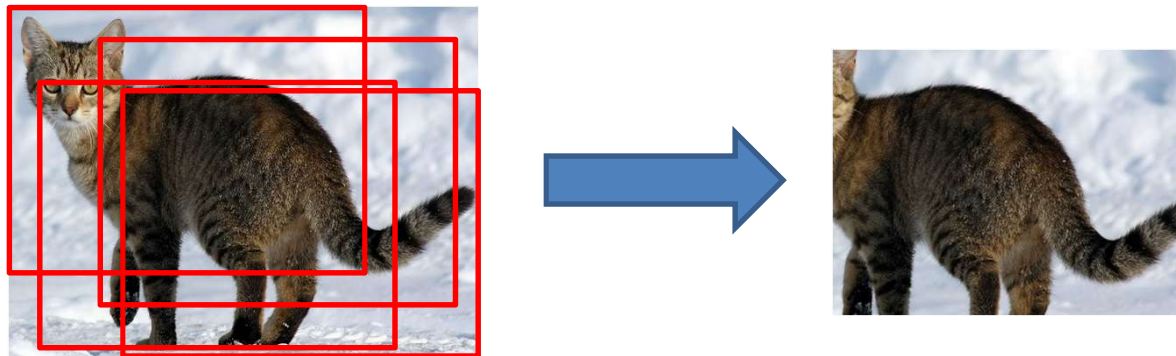
- Transformations: Color space, contrast, blur, ...



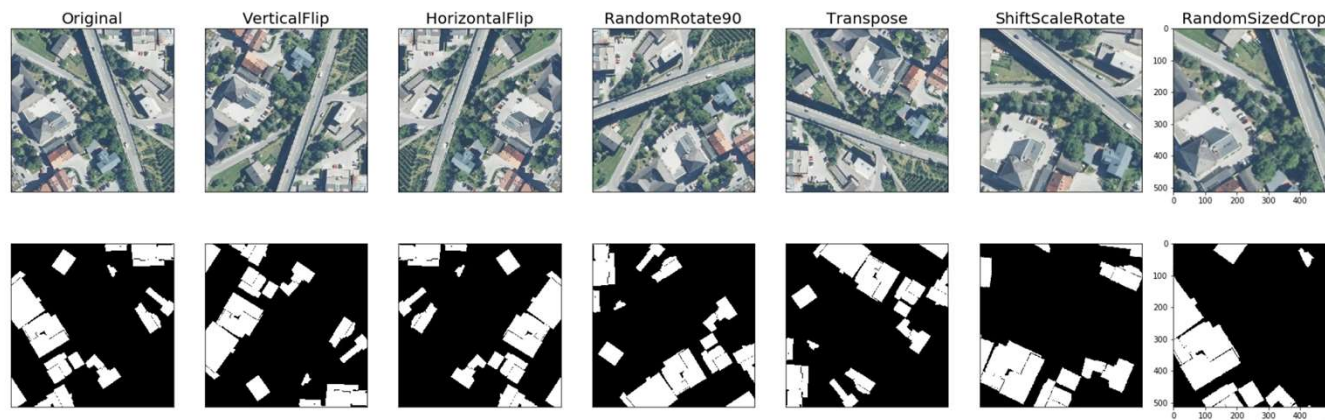
„Albumentations“
library for PyTorch

Data Augmentation

- Transformations: random crops



- Transformations: mirror, rotate, scale, ...

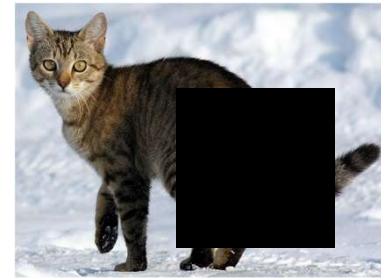
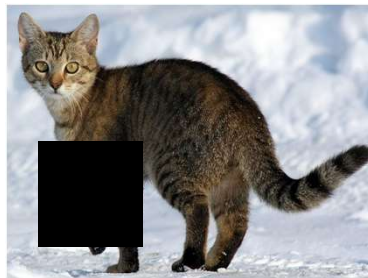


transformed
image

transformed
segmentation
label

Cutout

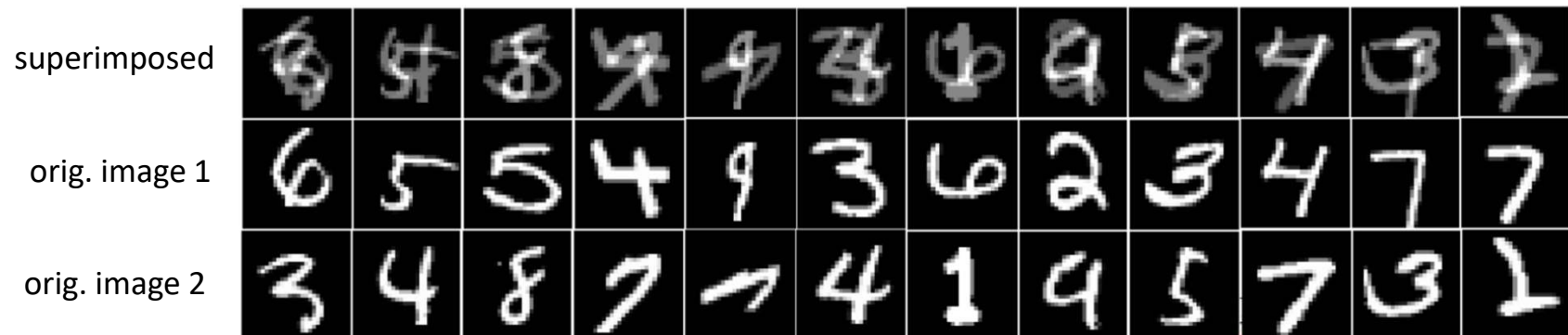
- **Cutout:** regularization technique applied on inputs of neural network
- During training, set random image regions to zero (on-the-fly as the images are used in SGD batches)



- At test time, use normal images
- Similar idea as for dropout: add variation, reduce reliance on specific image features
- Simple technique that often works well for small data sets
- Related to idea of **data augmentation**: add variation to available inputs

Mixup

- **Mixup**: another regularization technique applied on inputs of neural network
- During training, superimpose (linearly interpolate) two images weighting one image with $\alpha \in [0,1]$ and the other image with $1 - \alpha$:



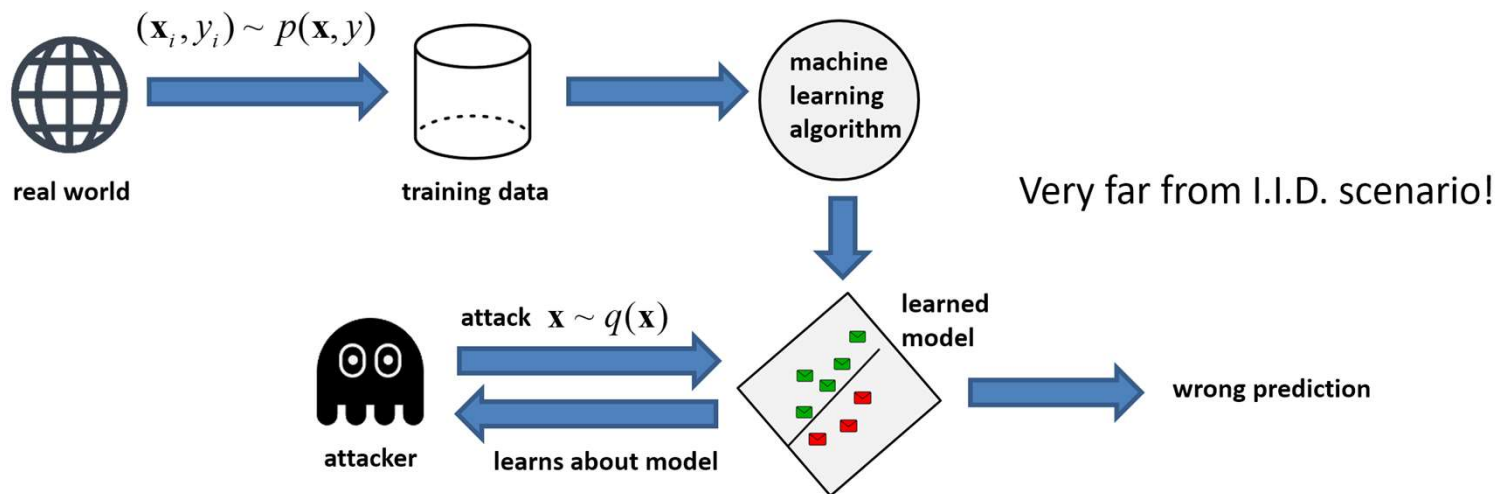
- Motivation for mixup: regularizes the neural network to favor simple linear behavior in-between training examples
- Empirical evidence that it can help avoid overfitting and stabilize training
- Also related to data augmentation, as it adds variation to the training data

Agenda Today

- Regularization and Data Augmentation
- Robustness and Adversarial Attacks

Adversarial Scenarios

- Some application domains show characteristics of an **adversarial** scenario:
 - After model deployment, an attacker deliberately tries to circumvent the model, that is, provoke a misclassification
 - Examples: spam filtering (make spam email pass filter), face recognition (impersonate subject), information retrieval (improve ranking of website)
- In such scenarios, distribution at test time deviates from training distribution



Adversarial Examples

- Most widely studied threat to machine learning models are evasion attacks
- **Evasion attack:** presenting an input that has been deliberately designed in such a way that it will be misclassified by the classifier
- The input in the evasion attack is called an **adversarial example**
- Example for evasion attack on the GoogLeNet CNN:

Original image
(correctly classified)



x

“panda”

57.7% confidence

Small additive perturbation



$\text{sign}(\nabla_x J(\theta, x, y))$

“nematode”

8.2% confidence

Adversarial example
(misclassified)



$x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$

“gibbon”

99.3 % confidence

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572* (2014).

Example: Attack By Small Image Patch

- Example: Changing only small patch in image



Padlock (92.7%)



Tiger Cat (94.4%)



Car Mirror (94.5%)



Stingray (90.5%)

Karmon, Danny, Daniel Zoran, and Yoav Goldberg. "Lavan: Localized and visible adversarial noise." *International Conference on Machine Learning*. PMLR, 2018.

- Instead of slightly changing all or most pixels in image, can also make stronger changes to a small number of pixels (small rectangular patch in images above)
- Patches contain patterns that strongly activate the target class

Evasion Attacks: Formalization

- Different formalizations of evasion attacks are possible
- Here, we will discuss an approach called **targeted regularization-based attack**:

Given

- a K -class classification model $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^K$ that outputs class scores
- an initial instance $\mathbf{x}_0 \in \mathcal{X}$
- a target class $t \in \{1, \dots, K\}$
- some norm or distance $\|\cdot\| : \mathcal{X} \rightarrow \mathbb{R}$

„Simultaneously minimize size of perturbation and maximize score for the target class“

Find

- an input $\mathbf{x}^* \in \mathcal{X}$ with $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{X}} \left(\|\mathbf{x} - \mathbf{x}_0\| + \lambda \underbrace{\left(\max_{j \neq t} \{f_\theta(\mathbf{x})_j\} - f_\theta(\mathbf{x})_t \right)}_{\text{minimizing this enforces that score of target class is higher than score of second-highest scoring class}} \right)$

minimizing this enforces that score of target class is higher than score of second-highest scoring class

Approach of Carlini and Wagner

- Example for advanced regularization-based attack: **Carlini and Wagner, 2017**
 - minimize squared norm between \mathbf{x} and \mathbf{x}_0
 - additional hyperparameter to control confidence of misclassification

λ : hyperparameter that trades off between confidence in target classification and strength of perturbation

Difference in score between target class and next highest-scoring class. We want to minimize this to maximize the relative score of the target class.

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{X}} \left\| \mathbf{x} - \mathbf{x}_0 \right\|_2^2 + \lambda \max \left(\underbrace{\max_{j \neq t} \{ f_{\theta}(\mathbf{x})_j \} - f_{\theta}(\mathbf{x})_t}_{\text{Difference in score}}, -\kappa \right) \quad (1)$$

Outer maximum with the hyperparameter κ : this part of the loss does not further decrease once score for target class is higher than score of next highest-scoring class by at least κ .

If we set $\kappa = 0$, we only want an example that is just classified as the target class, for $\kappa > 0$ we want a classification with higher confidence.

Carlini, Nicholas, and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods." *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017.

Constructing Adversarial Examples by Optimization

- How do we construct adversarial examples?
- **Optimization problem** of Carlini and Wagner:

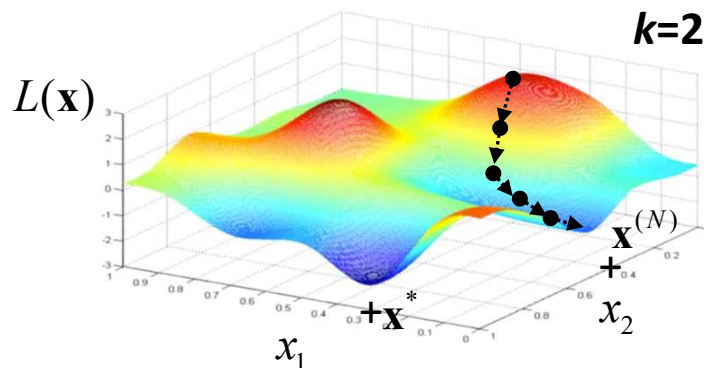
$$\begin{aligned}\mathbf{x}^* &= \arg \min_{\mathbf{x} \in \mathcal{X}} \underbrace{\left\| \mathbf{x} - \mathbf{x}_0 \right\|_2^2 + \lambda \max \left(\max_{j \neq t} \left\{ f_{\theta}(\mathbf{x})_j \right\} - f_{\theta}(\mathbf{x})_t, -\kappa \right)}_{L(\mathbf{x})} \\ &= \arg \min_{\mathbf{x} \in \mathcal{X}} L(\mathbf{x})\end{aligned}$$

- We have to solve the optimization problem
- **Black-box setting:** we can evaluate the model f_{θ} , but we do not know the full model. Use heuristic optimization approaches (hill-climbing, ...)
- **White-box setting:** we have access to the full model. Can use the model to efficiently find minimum

Adversarial Examples by Gradient Descent

- Carlini and Wagner study a white-box setting, in which an adversarial example minimizing the objective (1) is found using gradient descent
- For neural networks, can compute the gradient with respect to the input variables using backpropagation

$$\nabla L(\mathbf{x}) = \begin{pmatrix} \frac{\partial L(\mathbf{x})}{\partial x_1} \\ \dots \\ \frac{\partial L(\mathbf{x})}{\partial x_M} \end{pmatrix} \quad \text{input } \mathbf{x} = \begin{pmatrix} x_1 \\ \dots \\ x_M \end{pmatrix}$$



Gradient descent algorithm

1. $\mathbf{x}^{(0)} = \mathbf{x}_0$
2. for $i = 0, \dots, N$:
$$\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} - \eta \nabla L(\mathbf{x}^{(i-1)})$$
3. return $\mathbf{x}^{(N)}$

White-Box Versus Black-Box Attacks

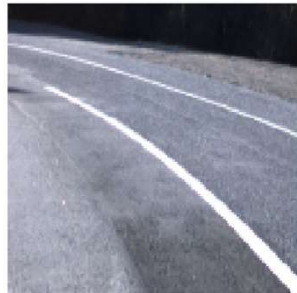
- Most methods for constructing adversarial examples (such as the Carlini and Wagner method) assume that the attacker has access to the full model
- In practice, this is a relatively rare scenario, at least in IT security
- However, often adversarial examples are transferable:
 - if an adversarial example for one model is found, it may also work for a different model that solves the same task
 - attacker might try to train a similar model and use it to construct adversarial examples
- There are also black-box approaches for attacking a model which an attacker can probe (that is, obtain output for particular inputs) but not fully access
- Black-box approaches tend to be a lot less effective than white-box approaches

Example: Adversarial Attack on Autonomous Driving

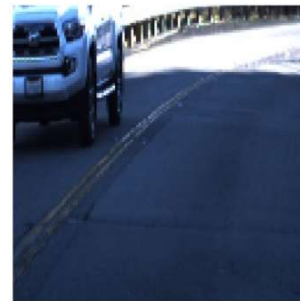
- Predicting steering angle from forward-facing camera image (public challenge data set, „Udacity Self-Driving Car Challenge“)
- Modified version of data set, cast as a direction classification problem with classes straight, left, and right



straight

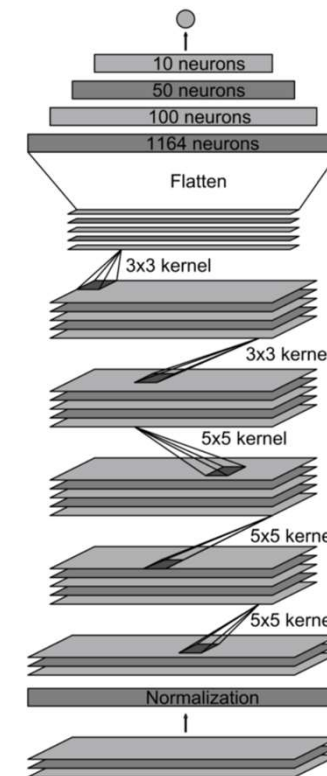


left



right

Chernikova, Alesia, et al. "Are self-driving cars secure? Evasion attacks against deep neural networks for steering angle prediction." *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019.



„NVIDIA“ architecture

Example: Adversarial Attack on Autonomous Driving

- Gradient-based adversarial attack similar to the Carlini and Wagner attack discussed above

Examples for adversarial images



original image (right)

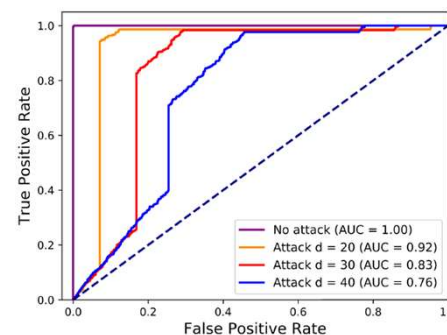
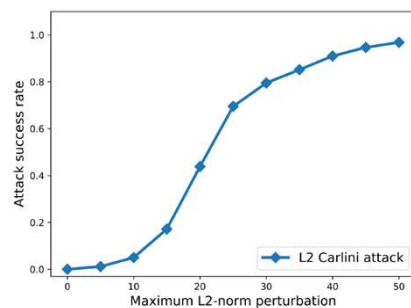


adversarial image (straight)



adversarial image (left)

Attack success rate depends on maximum perturbation

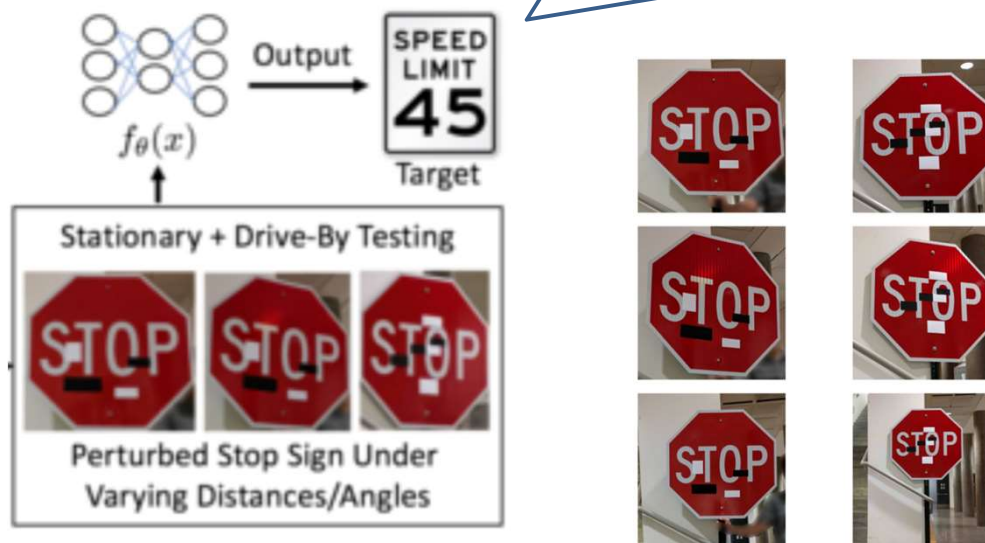


Adversarial Attacks in the Physical World: Stop Signs

- Adversarial attack in the physical world: creating real-world stop signs engineered to be misclassified by a traffic sign classifier

Challenge: create a real stop sign that will be misclassified as „Speed limit 45mph“ by a CNN-based traffic sign detector

Main difficulty: varying distances and viewing angles distort any adversarial patterns













Eykholt, Kevin, et al. "Robust physical-world attacks on deep learning visual classification." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

Adversarial Attacks in the Physical World: Stop Signs

- Adversarial attack in the physical world: creating real-world stop signs engineered to be misclassified by a traffic sign classifier

Attacks are often successful, as evaluated by real drive-by testing (white-box scenario)

Perturbation	Attack Success	A Subset of Sampled Frames $k = 10$				
Subtle poster	100%					
Camouflage abstract art	84.8%					

Eykholt, Kevin, et al. "Robust physical-world attacks on deep learning visual classification." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

Adversarial Attacks in the Physical World: Faces

- Adversarial attacks in the physical world: attacking face recognition system with carefully fabricated eye glasses
- White-box scenario, real eye glasses printed with adversarial patterns enable impersonating other subject or dodging face detection



classified as



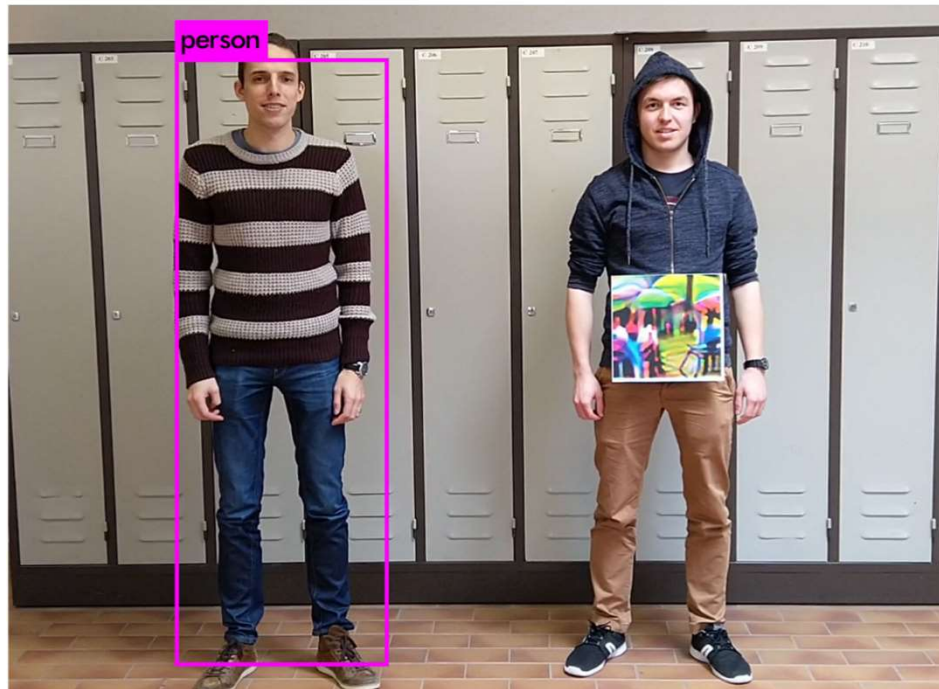
Successful
impersonation
example

<i>DNN</i>	Subject (attacker) info		Dodging results		<i>Target</i>	Impersonation results		
	<i>Subject</i>	<i>Identity</i>	<i>SR</i>	$E(p(\text{correct-class}))$		<i>SR</i>	<i>SRT</i>	$E(p(\text{target}))$
<i>DNN_B</i>	<i>S_A</i>	3rd author	100.00%	0.01	Milla Jovovich	87.87%	48.48%	0.78
	<i>S_B</i>	2nd author	97.22%	0.03	<i>S_C</i>	88.00%	75.00%	0.75
	<i>S_C</i>	1st author	80.00%	0.35	Clive Owen	16.13%	0.00%	0.33
<i>DNN_C</i>	<i>S_A</i>	3rd author	100.00%	0.03	John Malkovich	100.00%	100.00%	0.99
	<i>S_B</i>	2nd author	100.00%	<0.01	Colin Powell	16.22%	0.00%	0.08
	<i>S_C</i>	1st author	100.00%	<0.01	Carson Daly	100.00%	100.00%	0.90

Sharif, Mahmood, et al. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition." *Proceedings of the 2016 ACM sigsac conference on computer and communications security*. 2016.

Adversarial Attacks in the Physical World: Person Detection

- Adversarial image patches can make a person detector fail to detect a person
- This works in the real world using „adversarial sign“ held by a person that wants to stay undetected
- Relevant for surveillance scenarios



Thys, Simen, Wiebe Van Ranst, and Toon Goedemé.
"Fooling automated surveillance cameras: adversarial
patches to attack person detection." *Proceedings of the
IEEE/CVF Conference on Computer Vision and Pattern
Recognition Workshops*. 2019.

Defending Against Adversarial Attacks

- **How can we defend against adversarial attacks on learned classifiers?**
- Approach 1: Detecting and blocking adversarial examples
 - adversarial perturbations are usually imperceptible to the human eye
 - but maybe we can train models to detect adversarially manipulated inputs?
- Approach 2: Increase robustness by training with adversarial examples
 - if we generate adversarial examples at training time, and add them to the training set with the correct label, maybe classifier learns to become robust

Auxiliary Model for Detecting Adversarial Examples

- **Approach 1: Detecting and blocking adversarial examples**
- Idea: train a binary classifier to detect adversarial examples
- Two-step approach:

1. Train main classification model $f_{\theta} : \mathcal{X} \rightarrow \mathbb{R}^K$

Inputs $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$, $\mathbf{x}_n \in \mathcal{X}$

Labels $\mathbf{y} = (y_1, \dots, y_N)$, $y_n \in \mathcal{Y}$

2. Generate adversarial examples $(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N)$ for model $f_{\theta} : \mathcal{X} \rightarrow \mathbb{R}^K$

Train auxiliary classification model $g_{\bar{\theta}} : \mathcal{X} \rightarrow \mathbb{R}^2$

Inputs $\bar{\mathbf{X}} = (\mathbf{x}_1, \dots, \mathbf{x}_N, \bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N)$

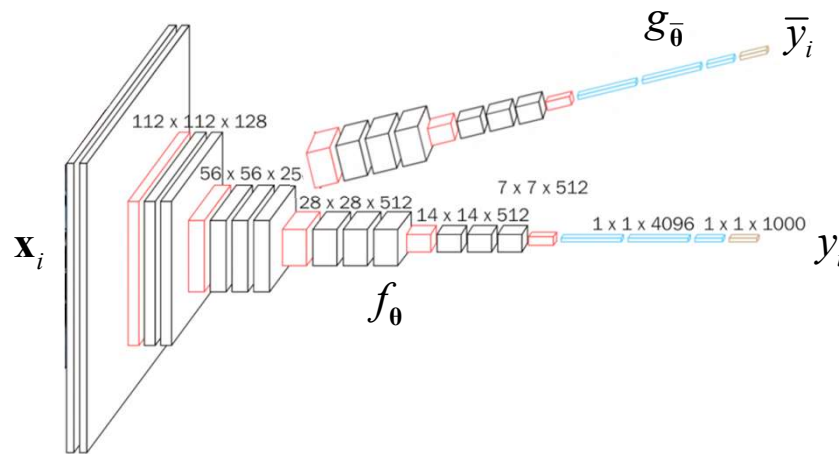
Labels $\mathbf{y} = (\bar{y}_1, \dots, \bar{y}_{2N})$ with $\bar{y}_n = \begin{cases} -1 & : n \leq N \\ 1 & : n > N \end{cases}$

This is a balanced binary classification problem.

Metzen, Jan Hendrik, et al. "On detecting adversarial perturbations." *arXiv preprint arXiv:1702.04267* (2017).

Auxiliary Model for Detecting Adversarial Examples

- **Approach 1: Detecting and blocking adversarial examples**
- At application time, run the auxiliary model on every example. If it predicts that the example is adversarial, block it
- For neural networks, computational efficiency can be improved by branching the auxiliary classifier off the main classifier



Shared structure:
Both models work on identical input.
Features in lower layers are useful
for solving both classification problems

- However: In a full white-box setting, an adversary could attack both models simultaneously, and still be successful...

Metzen, Jan Hendrik, et al. "On detecting adversarial perturbations." *arXiv preprint arXiv:1702.04267* (2017).

Adversarial Training

- **Approach 2:** Increase robustness by training with adversarial examples
- General idea: during training, would like to find a classifier that performs as well as possible under adversarial attacks
- Optimize the following objective [Madry et al, 2017]:

$$\theta^* = \arg \min_{\theta} L(\theta)$$

e.g. cross-entropy loss

$$L(\theta) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)} \left[\max_{\bar{\mathbf{x}} \in \mathcal{X}_{\mathbf{x}, \eta}} \ell(f_{\theta}(\bar{\mathbf{x}}), y) \right]$$

Expectation over data distribution
(represented by training data)

Loss of model under worst-case adversarial manipulation of instance \mathbf{x} (within an η -ball around \mathbf{x}).

Adversarial attack here takes the form of maximizing the loss (that is, confidently misclassifying example)

$$\mathcal{X}_{\mathbf{x}, \eta} = \{\bar{\mathbf{x}} \in \mathcal{X} \mid \|\bar{\mathbf{x}} - \mathbf{x}\| \leq \eta\} \text{ is an } \eta\text{-ball around initial } \mathbf{x} \in \mathcal{X}$$

Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083* (2017).

Adversarial Training

- **Approach 2:** Increase robustness by training with adversarial examples
- Practical translation of minimax objective into gradient-based optimization:
 - use mini batch gradient descent
 - at each iteration, replace the standard loss for instances by the loss of an adversarial example generated from the current model

$$\nabla L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \nabla \ell(f_{\boldsymbol{\theta}}(\bar{\mathbf{x}}_i), y_i) \quad \text{where} \quad \bar{\mathbf{x}}_i = \text{adversarial}(\mathbf{x}_i, \boldsymbol{\theta})$$

Given the current model, compute an adversarial example $\bar{\mathbf{x}}_i$ for the training instance \mathbf{x}_i by performing gradient ascent of loss in input

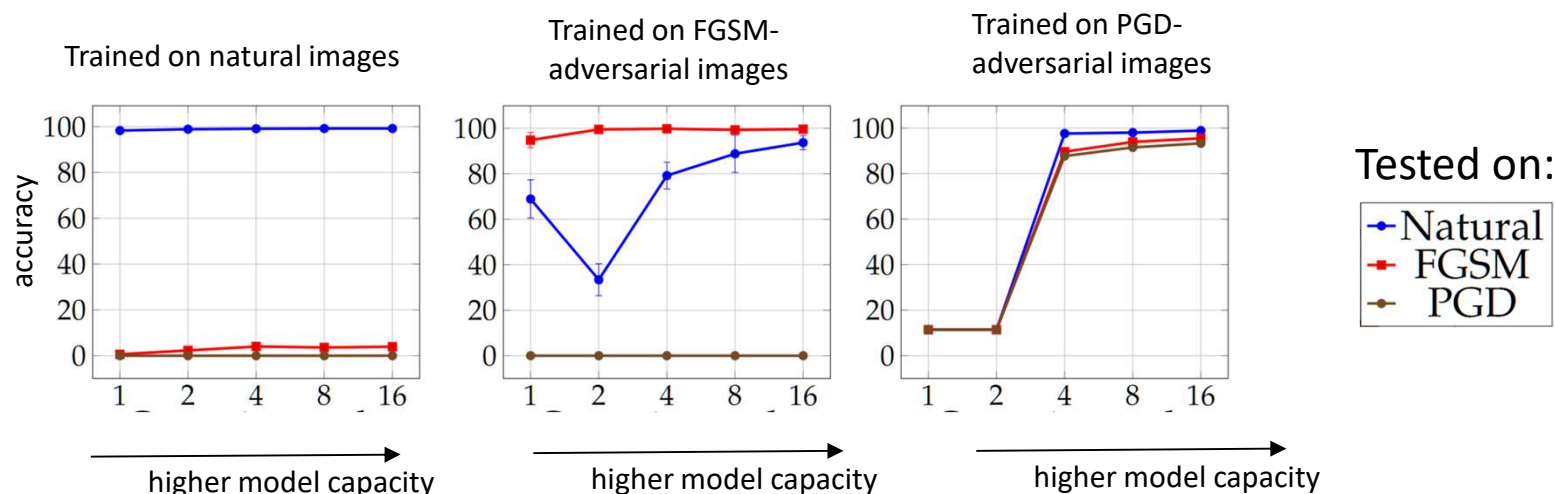
- General principle for minimax optimization: to get gradient of minimax expression, compute gradient at maximizer of inner expression
 - Not guaranteed to give the same result, but works well in practice
 - inner maximization is approximated by an adversarial attack

Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083* (2017).

Adversarial Training

- This kind of adversarial training is quite expensive: need to compute adversarial example for the current model at each pass over a training example (and this requires an iterative optimization)
- Yields quite good results in practice, if the model is large enough:

MNIST digit classification:



FGSM: fast gradient sign attack (attack type)

PGD: projected gradient descent (attack type)

Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083* (2017).

Adversarial Training as Data Augmentation

- Adversarial training can be seen as an extreme form of data augmentation:
 - we are augmenting or replacing the original training instances with transformed instances
 - transformation is not random, but adversarial to specifically increase robustness to adversarial attacks
- Empirically, there is often a trade-off between predictive accuracy on natural examples and robustness to adversarial attacks
 - often have to sacrifice some accuracy on natural images in order to become more robust to adversarial images
 - can be mitigated somewhat by increasing the model capacity

Summary: Regularization and Robustness

- As most machine learning approaches, computer vision models often benefit from regularization to avoid overfitting
 - General techniques such as weight regularization, early stopping, dropout; specific techniques such as cutout, mixup
 - For many vision problems, data augmentation is a simple and powerful method to improve generalization performance
- In adversarial scenarios, deliberately engineered inputs to model can lead to high error rates at application time
- Defenses against adversarial attacks: detecting adversarial inputs, and adversarial training