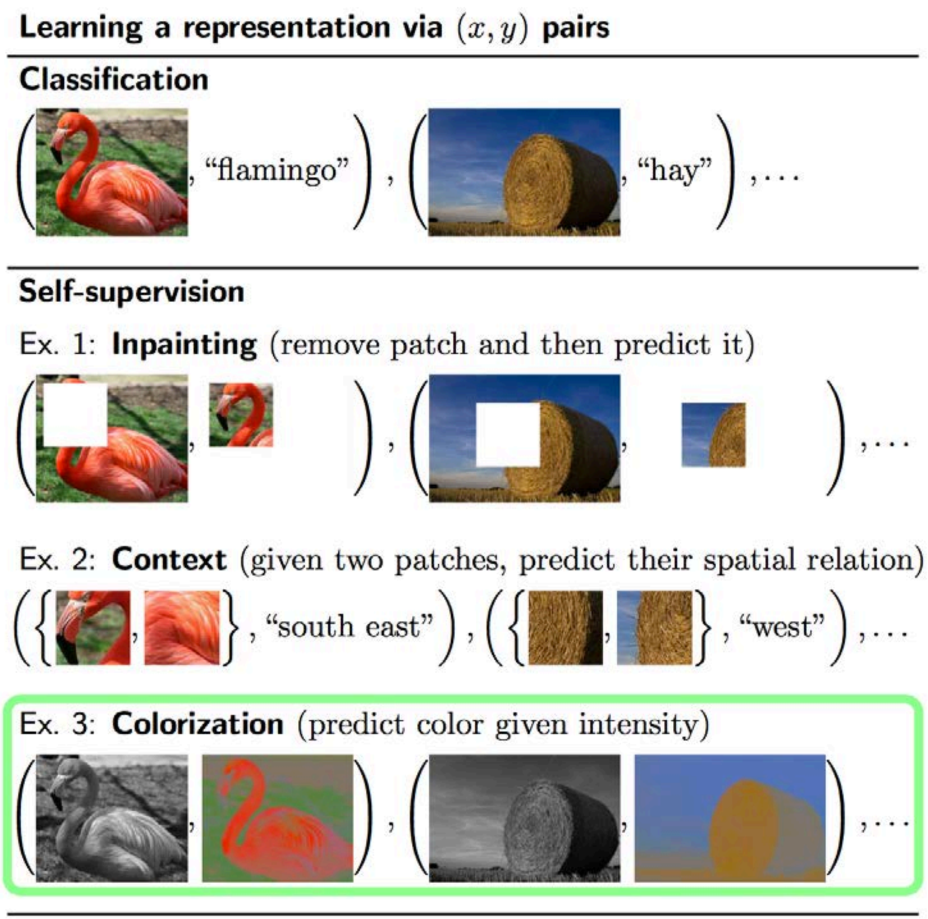
OTHER LEARNING METHODS & RELATED TOPICS

Other learning methods & related topics

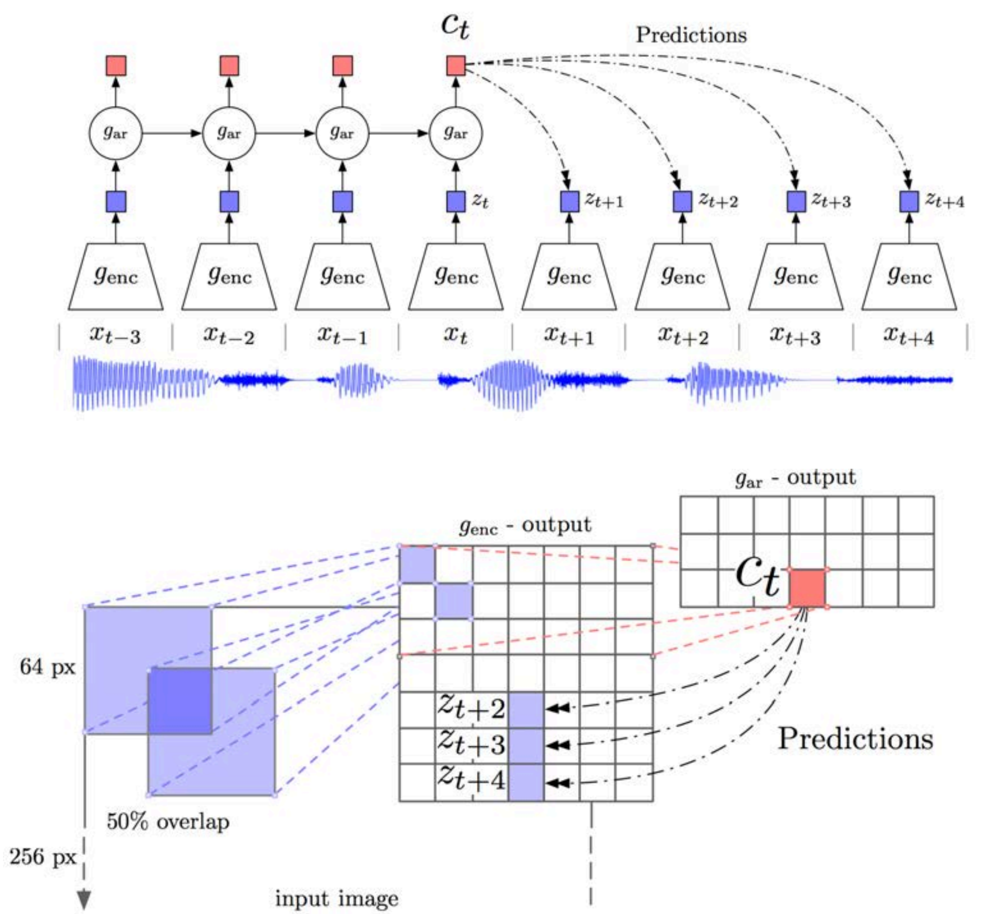
- Self-supervised learning
- Adversarial examples
- Model compression & Distillation
- Continual learning

Self-supervised learning

- Basically transfer learning; we pre-train the net on a task (called *proxy task*) for which labels are available for free, then fine-tuning it on the target task



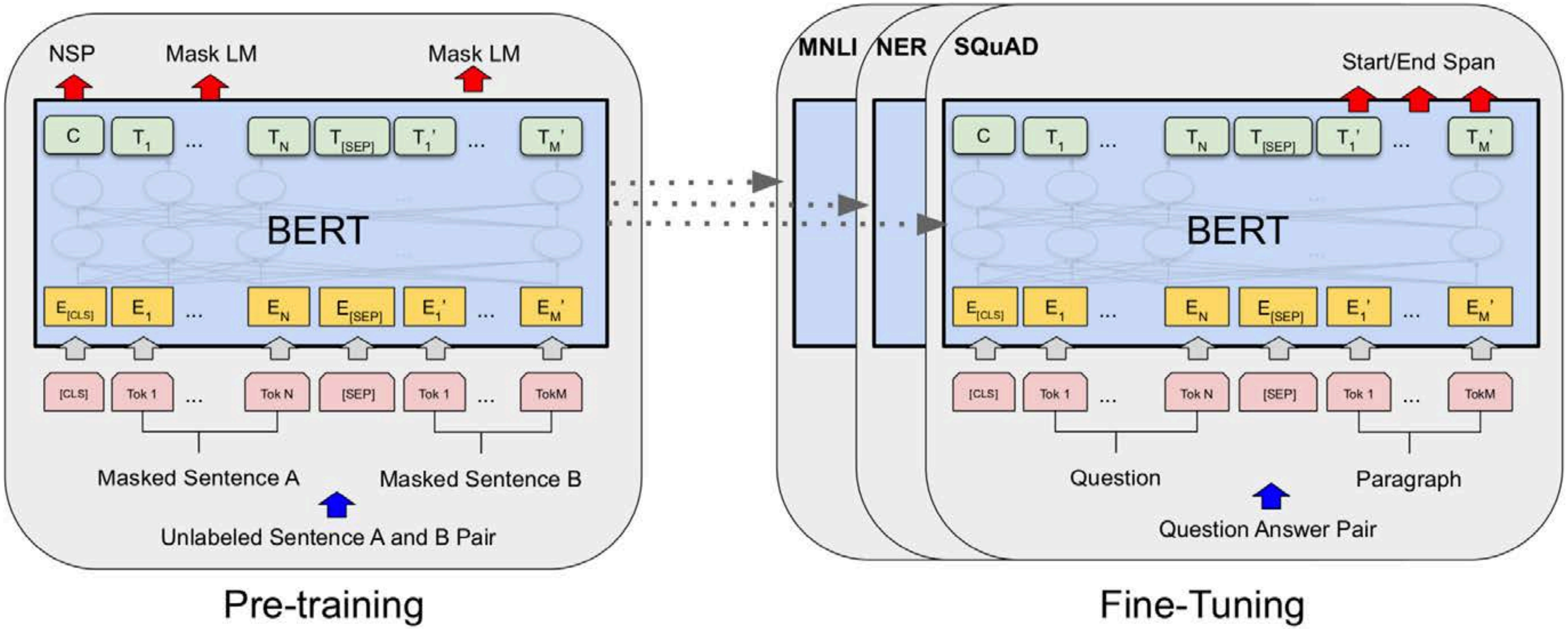
[Larsson-Maire-Shakhnarovich CVPR17]



Contrastive Predictive Coding [van den Oord-Li-Vinyals arXiv18]

Self-supervised learning

- Basically transfer learning; we pre-train the net on a task (called *proxy task*) for which labels are available for free, then fine-tuning it on the target task

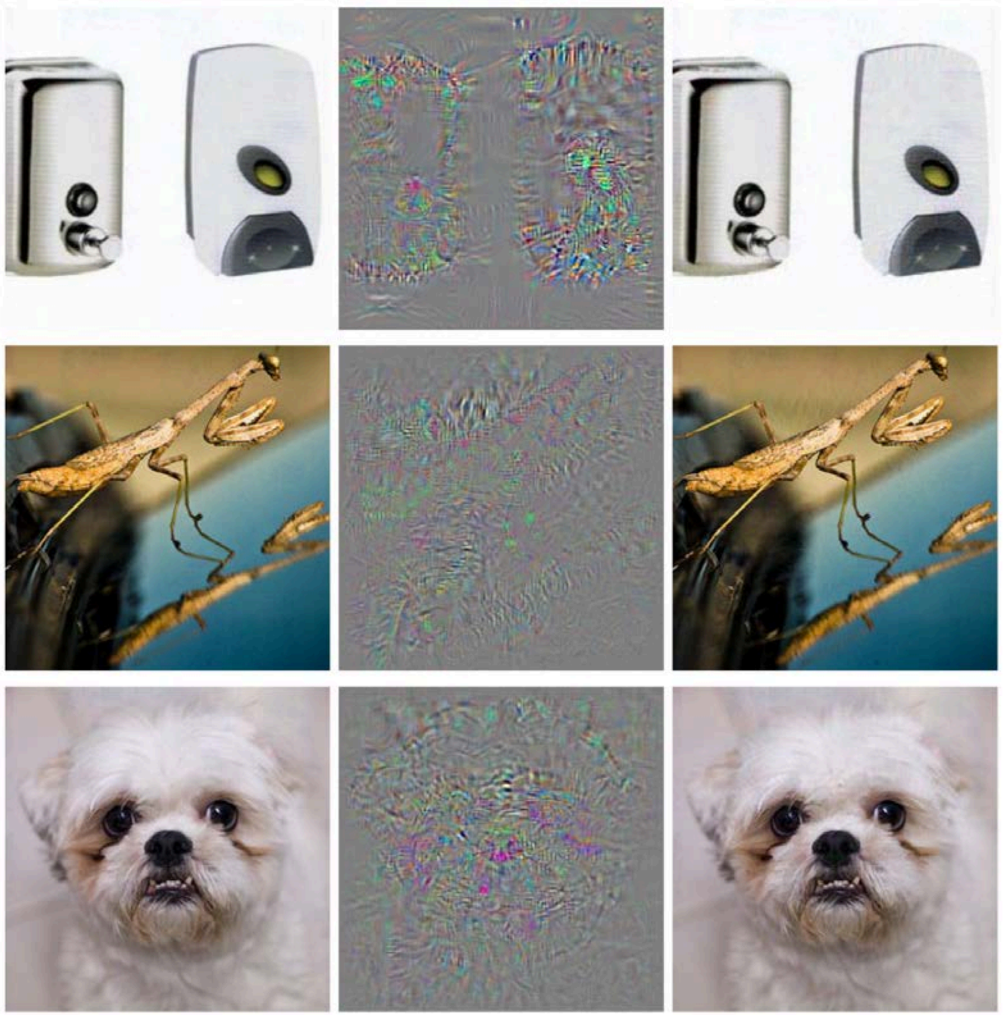


BERT: Pre-training of Deep Bidirectional Transformers
for Language Understanding [Devlin+18]

Adversarial examples

Szegedy+, Intriguing properties of neural networks, 2014

- Easy to fool CNNs with unnoticeable perturbation (for human)



Correctly recognized | Additive noise | Recognized as"Ostrich"

- Find small perturbation that makes the prediction wrong
  - You can specify to which class your net misclassify the input
  - Similar to optimization-based visualization

Minimize $\|r\|_2$ subject to:

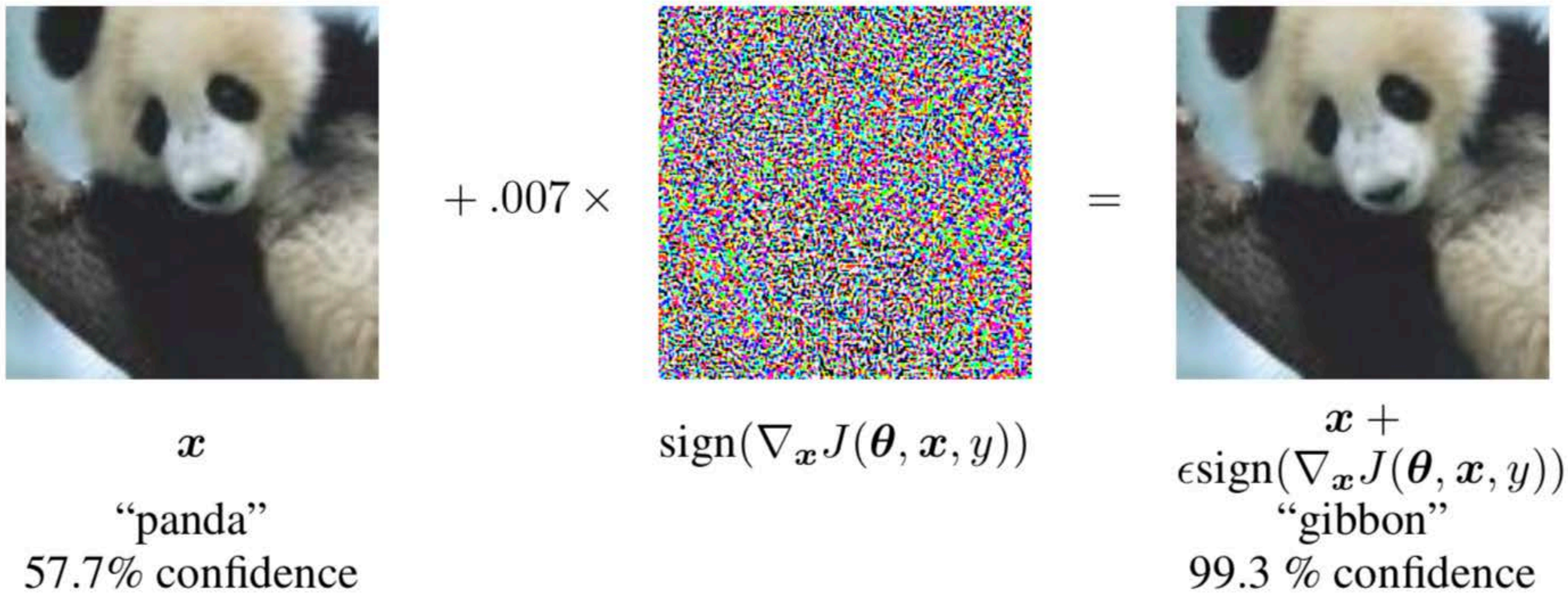
1. $f(x+r) = l$
2. $x + r \in [0, 1]^m$

A simple method: Fast Gradient Sign Method

Goodfellow+, EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES, ICLR2015

Loss for the true label for x

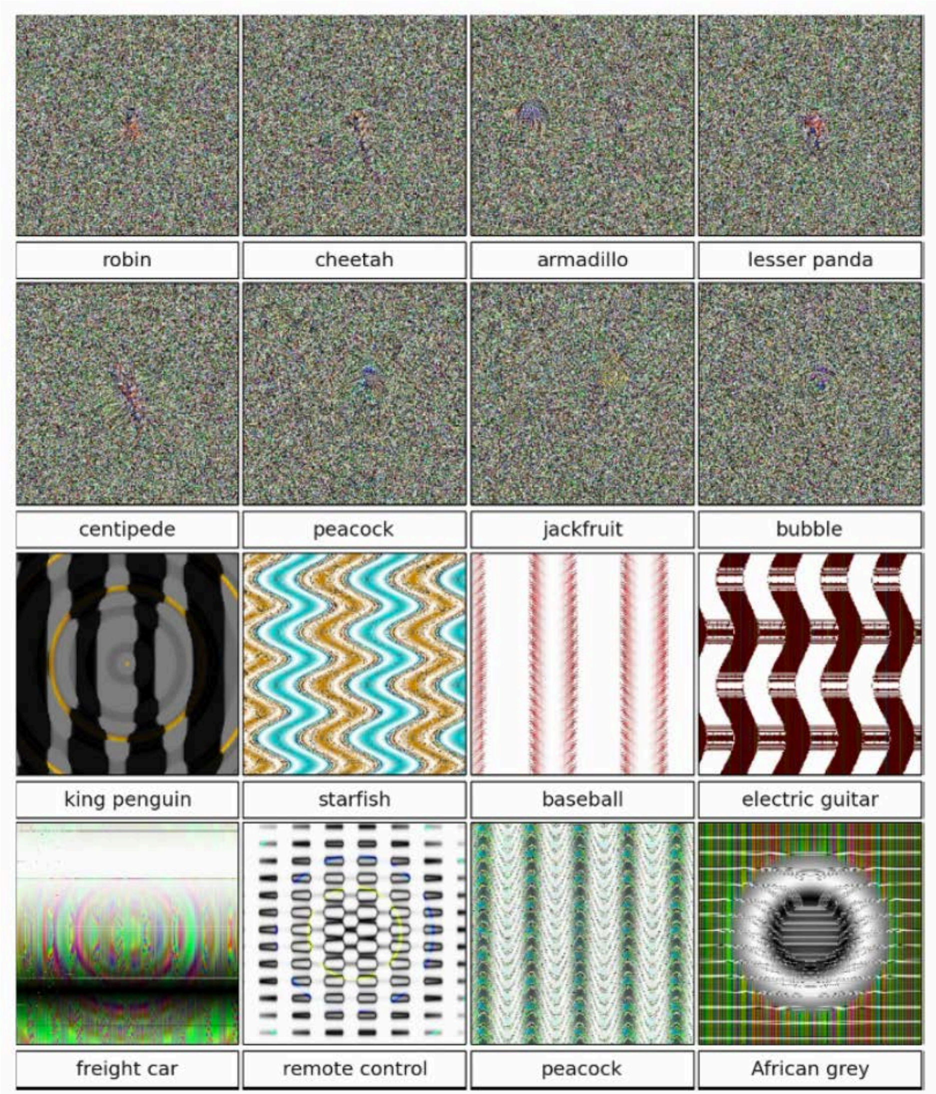
- Perturb an image in the direction of the gradient of the loss
  - It is the perturbation maximally changing the output *if the input-output relation is linear*
  - DNNs are highly nonlinear but is very linear locally
- To create small perturbation, we compute the sign of their elements and multiply them with small *ε*



Fooling samples (or ”rubbish class”)

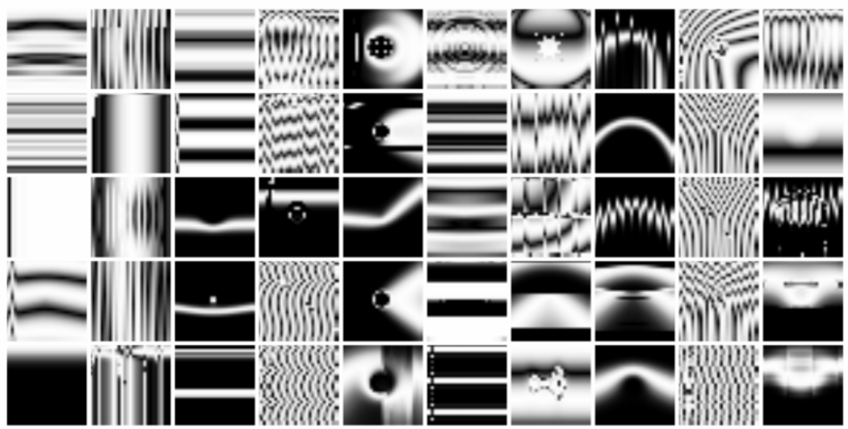
Nguyen+, Deep Neural Networks are Easily Fooled..., 2014

- Images fooling CNNs but not us; many generation methods



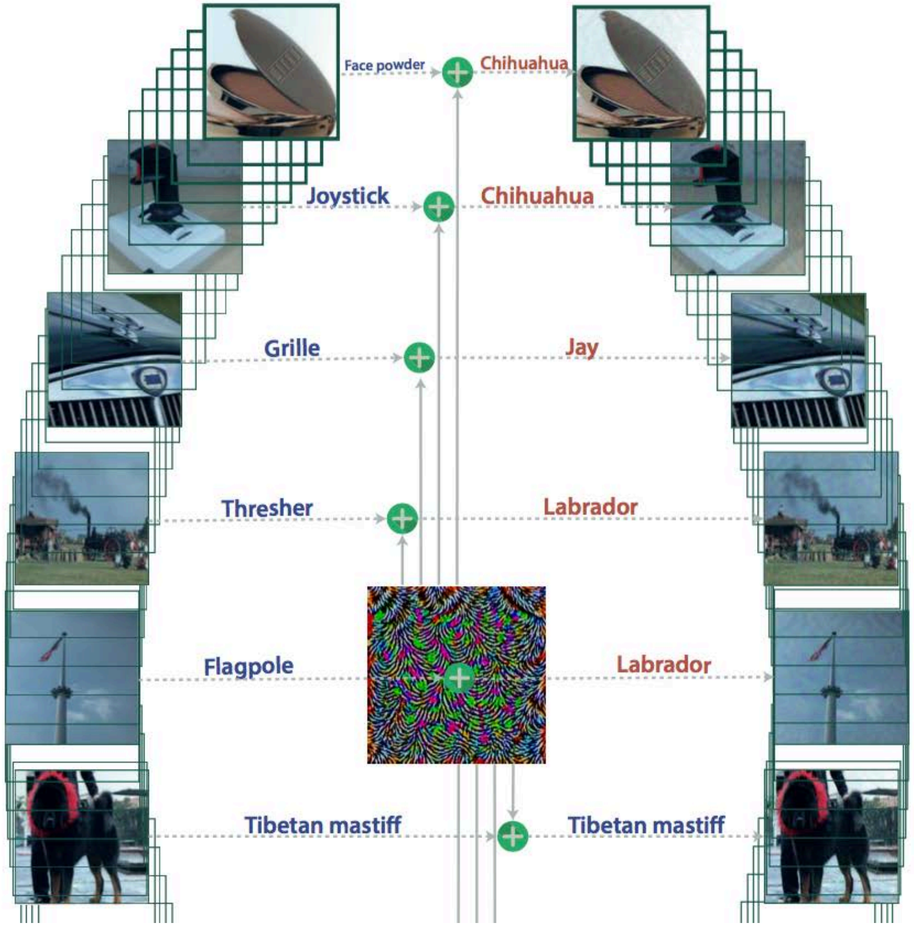
Images fooling a CNN recognizing MNIST

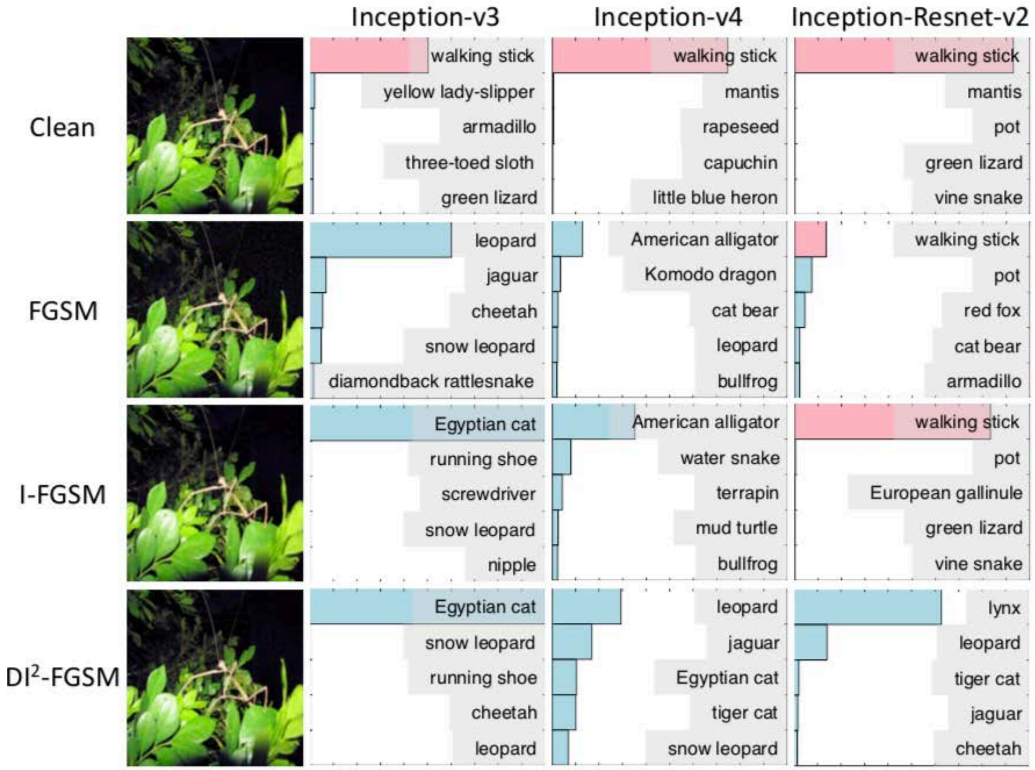
0 1 2



Adversarial examples: Universality

- Single noise pattern effective for many classes
  - Moosavi-Dezfooli & Fawzi, Universal adversarial perturbations, CVPR2017
- A pattern effective for a net can also be effective for others
  - Xie+, Improving Transferability of Adversarial Examples with Input Diversity, arXiv2018



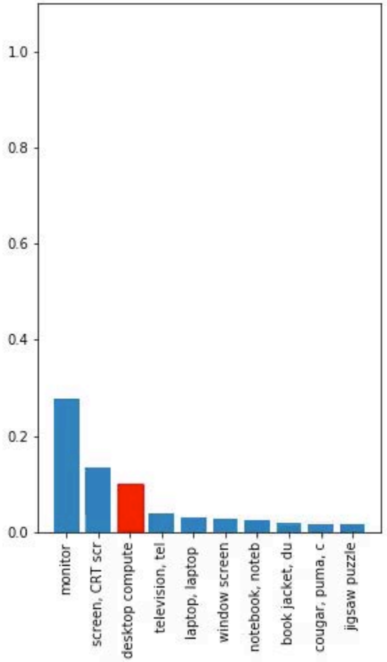


Adversarial examples: Attacks in the real world

‘Robust’ adversarial examples

[Athalye-Sutskever2017]

- Printed images
- Robust to imaging conditions

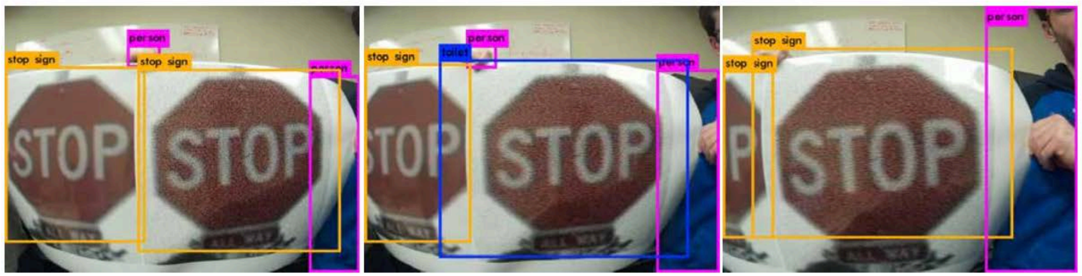


Adversarial patches

[Brown+arXiv Dec. 2017]

- A patch that forces a CNN misrecognize objects in the scene

Traffic signs: a small negligible perturbation can make a CNN misrecognize them



[Lu+2017]

Assignment 4

- Mission: Examine the dependency of adversarial examples on networks with different architectures/initial values
  - Send your submission (all other assignments you haven't submitted so far if any) to okatani@vision.is.tohoku.ac.jp by Dec. 2
- Minimum requirements: https://drive.google.com/open?id=17yEw7FRhzRqoE0bPSKfNZ85hOU_yHv2Q
  - Design at least two networks (e.g., those you created in Assignment 1)
    - Let them denoted by N_A & N_B
  - Train each net from two different initial weights
    - So you have four models, N_A, N_A', N_B, & N_B'
  - Generate many adversarial examples for each model and test them on all the models
    - Generate x*'s for N_A and test them on N_A', N_B, & N_B' and repeat this for all the combinations
  - Report the accuracy for each pair
    - Create a table of the format shown in the following page
    - Provide your observations on the results
- Optional (5% additional score will be given if you accomplish this):
  - Try a different set of epsilons (a parameter of the adversarial attacks) and report its effects

Assignment 4

Networks for evaluation of accuracy

Target networks of adversarial attacks

	N_A	N_A'	N_B	N_B'
N_A	*15%*	*18%*	*50%*	…
N_A'	…	…		
N_B				
N_B'				

Model compression

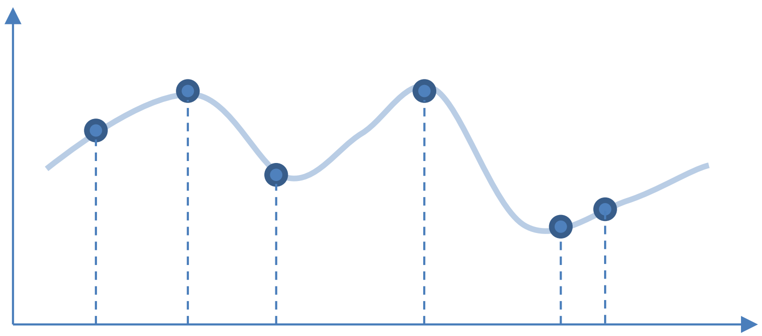
Bucil˘+, Model Compression, KDD2006 / Ba-Caruana, Do Deep Nets Really Need to be Deep?,NIPS2014

- Background
  - An ensemble of multiple models works better than a single model
  - However, it is inefficient in computational cost and/or memory usage
- Basic idea of model compression
  - Let the ensemble model (= teacher) predict the labels of inputs
  - Train a student model using the predicted labels as targets
  - The student learns the function represented by the ensemble model
- Observation
  - It is generally true that a student w/ only a fewer parameters can achieve similar accuracy to the teacher
  - DNNs are sometimes very large only for make their training successful

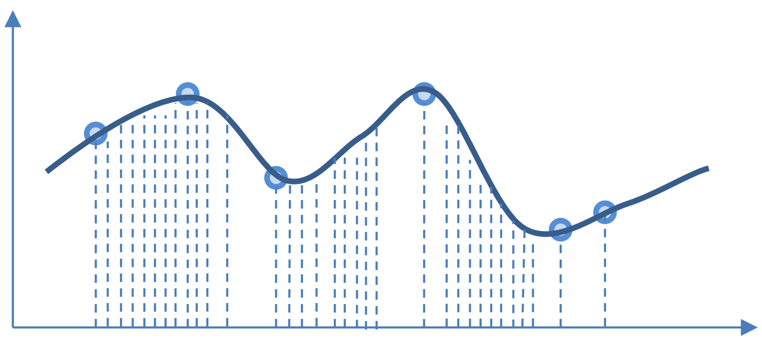
Model compression

Bucilˇ+, Model Compression, KDD2006 / Ba-Caruana, Do Deep Nets Really Need to be Deep?,NIPS2014

- Train a large model (ensemble etc.) with a certain amount of data

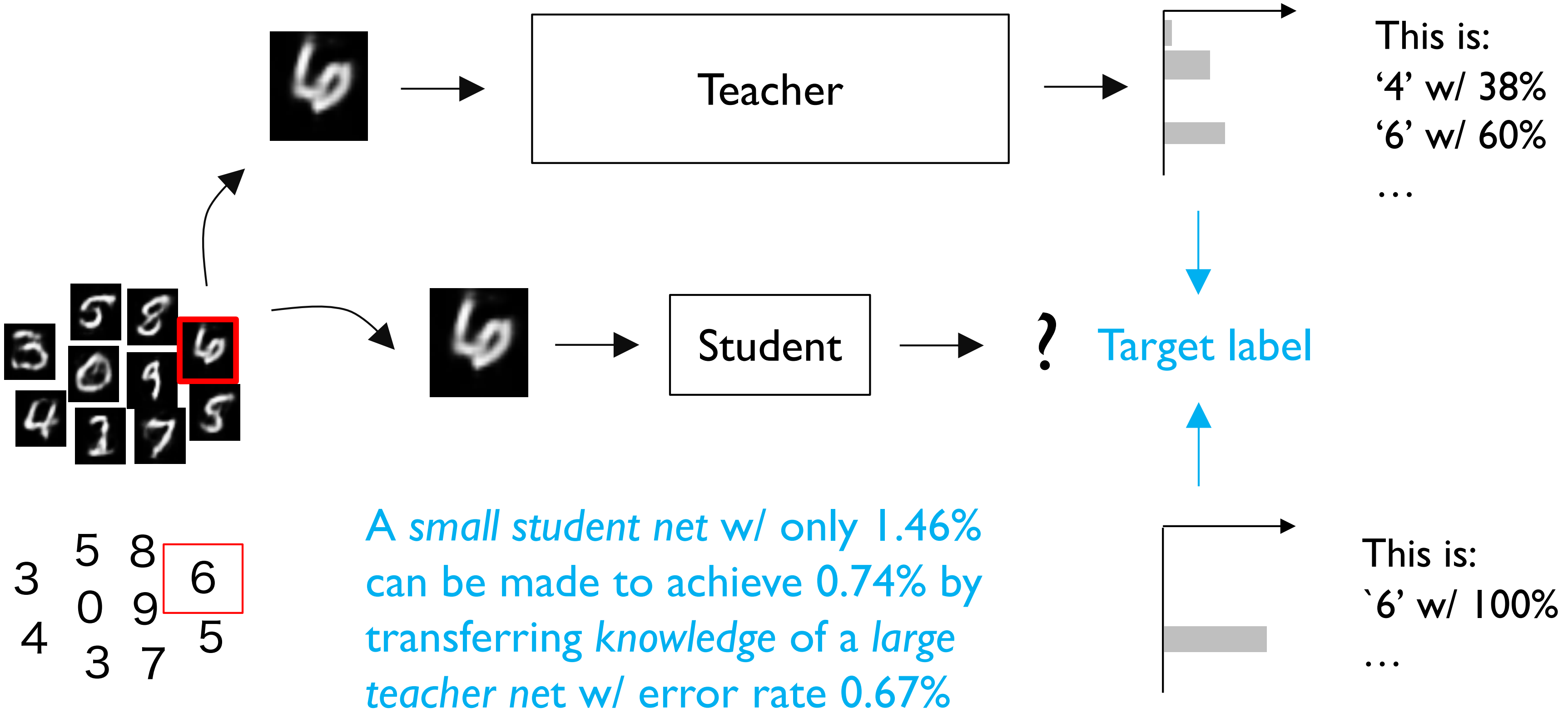


- We can train another (smaller) model using more data
  - We don't need to have true labels for them



Knowledge distillation (Student-Teacher model)

- Given a network already trained on a task (=a teacher), we want to train another network (=a student)
- Method: Use the output of the teacher as supervised signals for training the student



Knowledge distillation (Student-Teacher model)

- Minimize the sum of two losses for the training of the student net
  - A standard cross-entropy loss with provided 'true' labels (1-hot vec.)
  - A cross-entropy loss with 'soft' targets, prediction from the teacher, using *temperature scaling*

$$L = -\lambda \sum_{j=1} t_j \log(y_j) - (1-\lambda) \sum_{j=1} t'_j \log(y'_j)$$

$$y_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$