Data augmentation, semi-supervised learning , adversarial learning

Data Augmentation

Creation of synthetic data by slightly modifying the original training data

Useful when we have little training data

Also useful when we have unbalanced classes, that is, when there are very few examples belonging to one or more of the classes in the dataset.

```
def synthetic_classification(x1,x2):
     # x1 and x2 are assumed to belong to the same class
     w =  np.random() #  w is a random number between 0 and 1
     return x1*w + x2*(1-w)

def synthetic_regression(x1,y1,x2,y2):
     # (x1,y1) and (x2,y2) are assumed to be 'similar'
     w =  np.random() #  w is a random number between 0 and 1
     return x1*w + x2*(1-w), y1*w + y2*(1-w)
```

Data Augmentation for image data

Depending on the application, images may be:

- Shifted
- Rotated
- Mirrored
- Changed in color
- Warped

Recently proposed augmentation techniques have provided good results in some tasks by swapping regions of images and averaging their target function values

Keras provides routines to perform image augmentation 'on-the-fly'

Semi-supervised Learning

Suppose we have a small labeled dataset but a large set of data without labels is available.
For example, images from the internet for an object recognition task

We can do the following:
Let (X,y) be the original training set
Let Xu be the unlabeled dataset

Repeat
    model.fit(X,y)
    yu,c = model.predict(Xu)
    # Where c[i] is the confidence the model has in the accuracy of prediction yu[i]
    let (Xh,yh) be the examples in (Xu,yu) highest highest c
    move Xh from Xu to X
    append yh to y

Adversarial Learning

Recall:

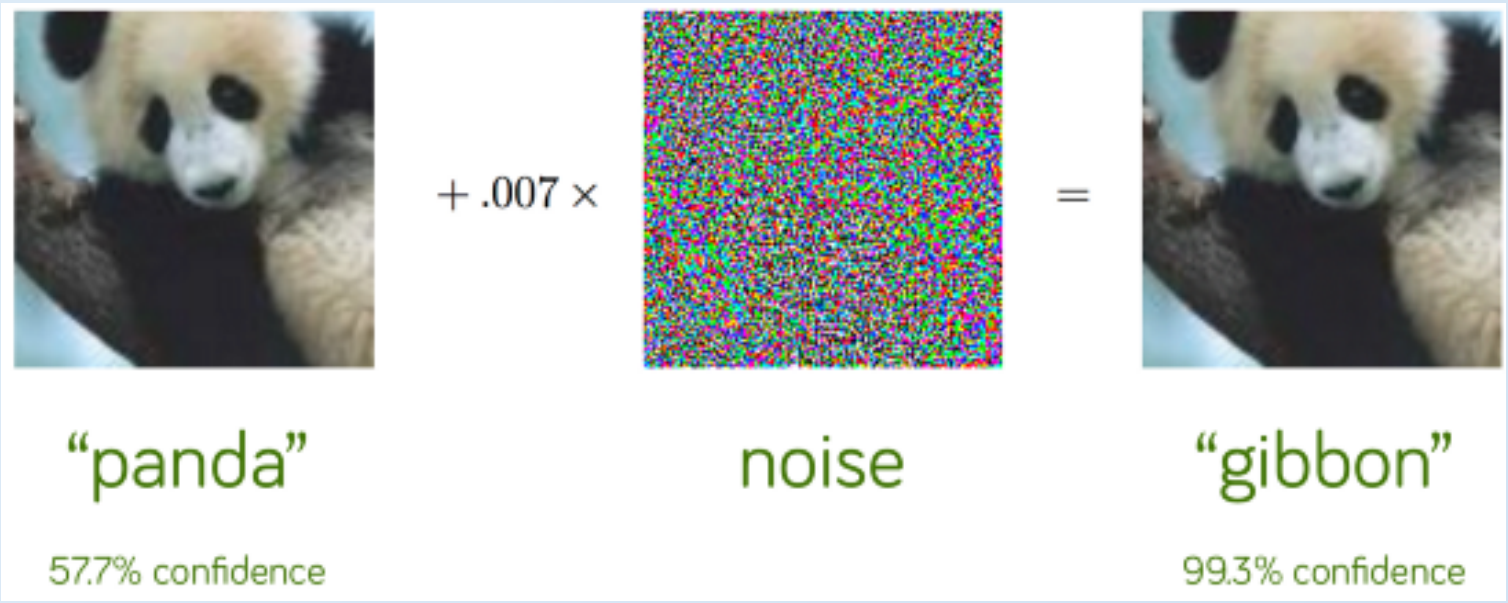
- Deep learning works by performing gradient descent
- The gradient of the network is a vector containing the partial derivative of the training error with respect to each of the network's parameters
- For every batch, each parameter is updated in the direction of decreasing error
- From the output error of the network, gradients of parameters in earlier layers are calculated using the chain rule from calculus
- Gradients are computed layer by layer, starting with the output layer and going backward – hende the term 'backpropagation'

What if instead of stoping at the input layer, we backpropagate the error to the input?

- The gradient would tell us how to modify the image to decrease the error
- If we modify the image in the opposite direction (gradient ascent), we would increase the error

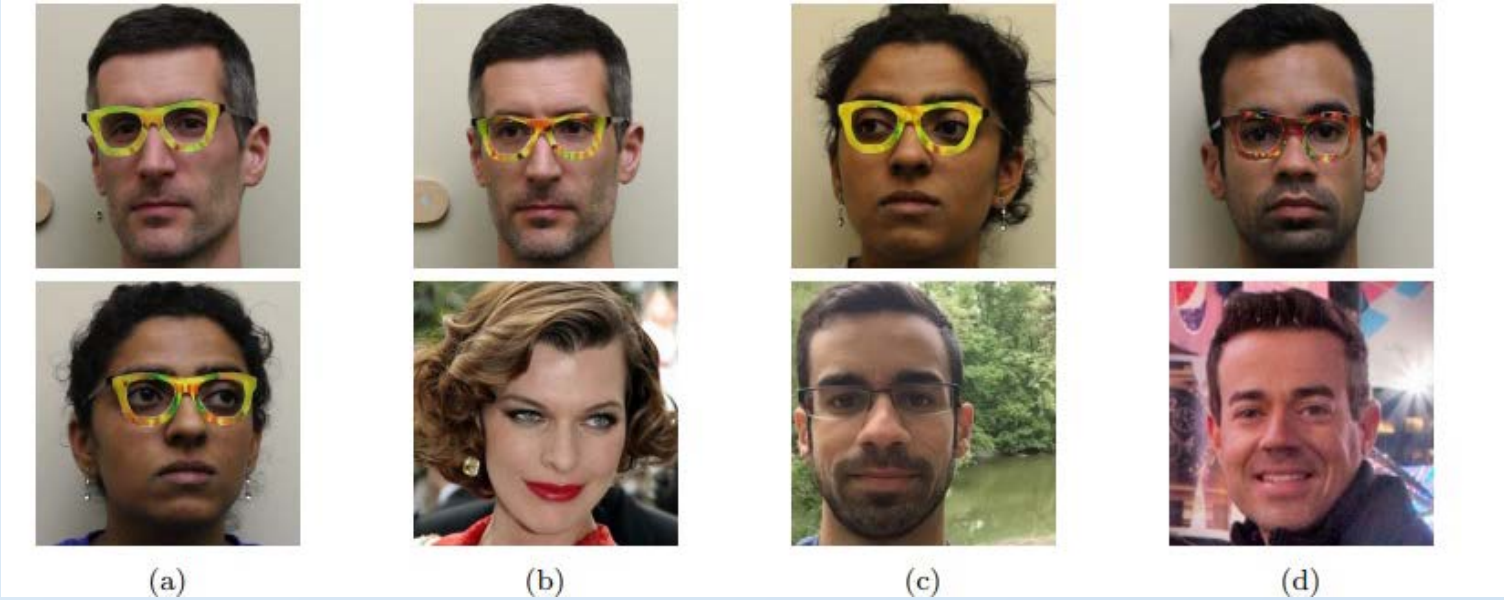
It turns out that surprisingly small changes to images are enough to fool a deep neural network

Adversarial Learning



Adversarial Learning

Adversarial Learning



(a) (b) (c) (d)

Adversarial Learning