Starting from ImageNet Model: Fine-Tuning and Adversarial Attack

Liangliang Cao

https://columbia6894.github.io/

Administrative

By this week you should:

1. form final project team (3 ppl per team)
2. decide the topic of your final projects

Then you can

- finish 1st problem in your homework 3
- receive google cloud credits (by contacting TA)

Administrative (Continued)

Have problems with your final project? Try the following

1. Choose a **recent** paper that you are interested in
2. Implement it or test it by yourself
3. Then you can
   a) If it fails in some scenario, try to improve it
   b) If it works great, apply it to a new application

For more details, refer to

http://llcao.net/cu-deeplearning17/lecture/lecture5_llc.pdf

Outline

Last vision lecture discussed learning from ImageNet/Celeb 1M

You can impress others with a model pretrained from ImageNet:

- Fine-tuning
- Adversarial attack

Fine Tuning Models from ImageNet

Models trained from ImageNet have learned effective feature presentation.

We can treat deep CNNs as feature extractors, and fine tune a new model over it.

Several ways to do the fine tuning:

1. Train Linear SVM over deep features (least training examples)
2. Fine tuning only the cross-entropy (more training examples)
3. Fine tuning both classifier and deep CNNs (about thousands of examples)

1. Fine-tuning using linear SVM

Recommend Liblinear for linear SVM:
https://www.csie.ntu.edu.tw/~cjlin/liblinear/

Steps:

1. Take the last layer of ImageNet model as feature extractor
2. Extract the feature for all the training examples
3. Optimize liblinear model and fine the global minimization

2. Fine-tuning the top layers

Steps:

1. Freeze the convolutional layers of pre-trained model

```
model = applications.VGG16(weights='imagenet', include_top=False)
for layer in model.layers[:25]:
    layer.trainable = False
```

2. Add a top model for the new classification task

3. Train the top model using SGD optimization

One easy-to-follow reference: https://towardsdatascience.com/a-comprehensive-guide-on-how-to-fine-tune-deep-neural-networks-using-keras-on-google-colab-free-daaaa0aced8f

Differences

	SVM	New Top layer
Extra toolkit	Liblinear	Keras/Tensorflow
Optimization	Global minimum	SGD may fall to local minimum
Number of training	A few images to dozens	Dozens to hundreds

Ticks of improve fine-tuning:

1. For SGD, use slow training rate first, do not use RMSProp
2. Data augmentation may be very helpful
3. In deep learning, people usually do not add regularization. But for fine-tuning, regularization may help.

3. One Step Further

When there are more training samples, we shall consider fine-tune the convolutional filters as well:

- Use smaller learning rates for CNN filters but bigger rates for the top layer.
- Monitor the training loss whether training accuracy = 100%
- Reference: http://caffe.berkeleyvision.org/gathered/examples/finetune_flickr_style.html

Future research questions you may consider for final projects:

- Actively choose samples to label? (with a budget)
- Efficiently learn new categories, e.g., for face recognition

Adversarial Attack

How many of you think deep CNNs are very reliable?

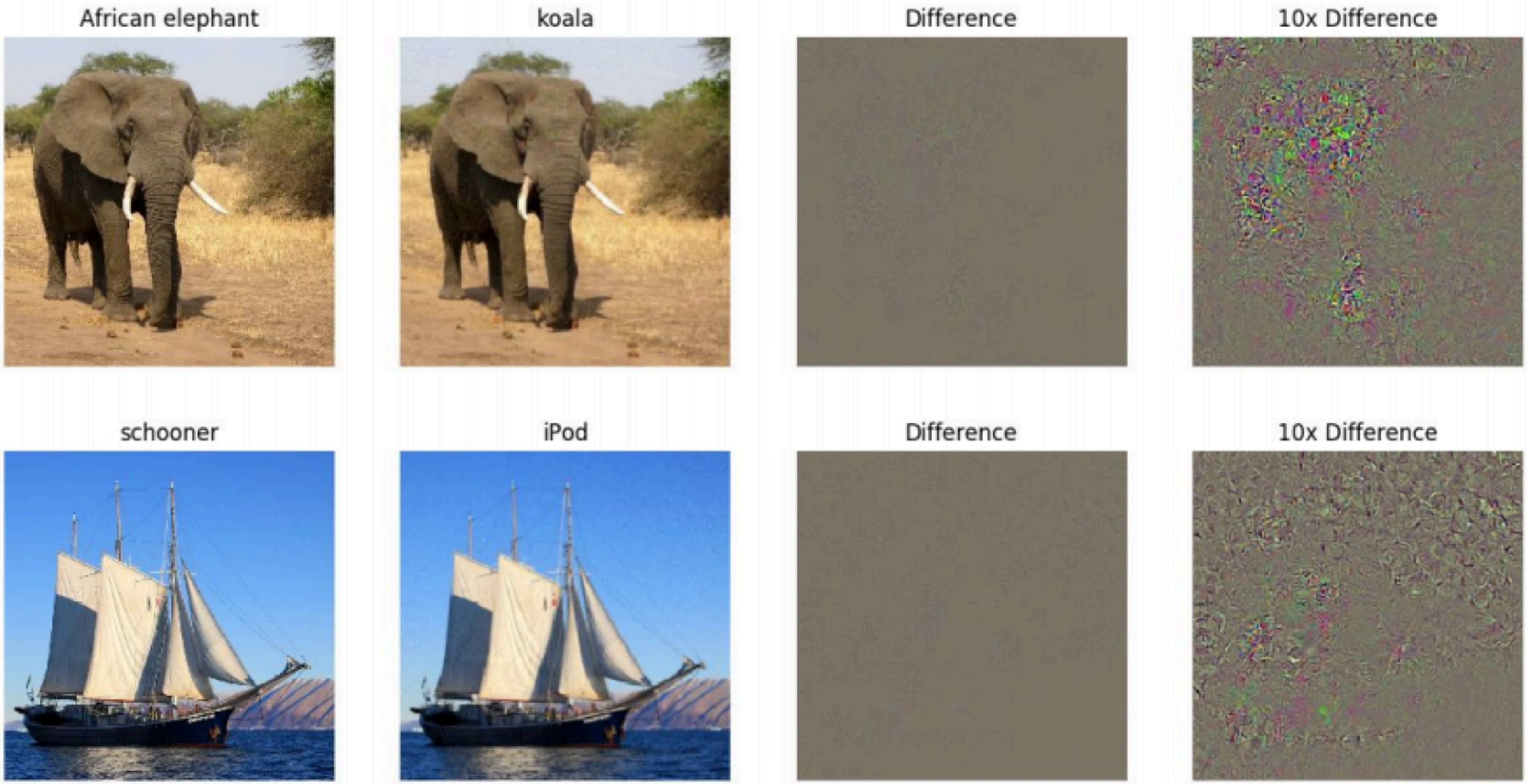
Guess what an ImageNet model will predict:

Adversarial Attack

How many of you think deep CNNs are very reliable?

Guess what an ImageNet model will predict:

Adversarial Attack



Examples are courtesy to Stanford cs231n lecture slides.

How to Compute the Adversarial Example

$$J(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}) \approx J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$$

Maximize $(\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$

Subject to $||\tilde{\boldsymbol{x}} - \boldsymbol{x}||_\infty \leq \epsilon$

The Fast Gradient Sign Method

So the adversarial example can be generated by

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \mathrm{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x})\right)$$

How to Implement?

There are a number of toolboxes such as cleverhans, Foolbox, etc

```
from cleverhans.attacks import FastGradientMethod
fgsm = FastGradientMethod(model, sess=sess)
fgsm_params = {'eps': 0.3, 'clip_min': 0., 'clip_max': 1.}
adv_x = fgsm.generate_np(orgin_x, **fgsm_params)
```

But fundamentally it is just to compute the gradient subject to input x.
You should read the code of cleverhans or Foolbox by yourself.

From White-box Attack to Black-box Attack

Fast Gradient Sign Method (fgsm) requires to know the model parameters to compute the adversarial attack. It is called a white-box attack coz we know the details of the model.

In practice attacker does not know the model. They can

- evaluate the model multiple times to approximate the gradient
- attack some venerable tasks such as object detection or QA

For final projects you may refer to:

- NIPS 2017 adversarial attack competition
- Percy Liang and Dawn Song's work on adversarial attack systems
- The Elephant in the Room attack for object detection