50.021 -AI

Alex

Week 05: Data Augmentation

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

Data augmentation

The goal of this lesson is to understand about data augmentation - for training and for testing.

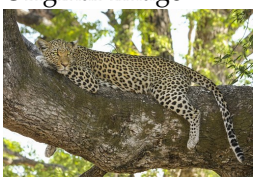
One does not simply predict on a raw sample. One does not simply train with a raw sample.

Data augmentation is about changing your data a little. The goals of data augmentation are:

- Testing time: allow the neural net to see *multiple views* of the same sample. One creates multiple views of one sample. The final prediction will be an average of the prediction scores, averaged over all views. "Look at all aspects to understand a thing."

  Example in this class: Do not input an image as a whole, but input 5 crops: a cropped image from the center, and 4 corner crops.

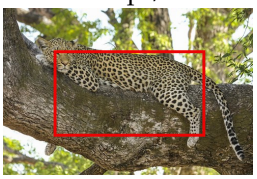
  Original Image

  compute the prediction as an average over crops:

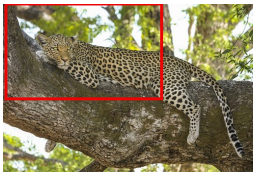
  $prediction =$

  Center crop $/5$

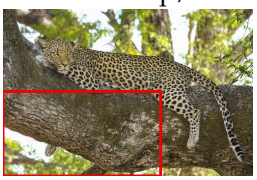
  $+$

  upper left crop$/5$



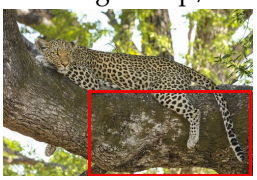
+

lower left crop/5



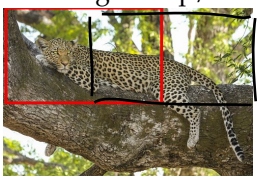
+

lower right crop/5



+

upper lower right crop/5



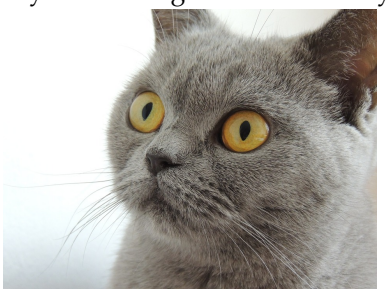
- Training time: allow the neural net to be trained with more data samples. Data augmentation increases the data size by creating many similar samples from one sample. Example: one can take an MNIST digit and distort its shape slightly before putting it into the training loop.
- Training time: training with slight deformations of a sample (e.g. darker or brighter images) will allow the network to be able to recognize similarly brighter or darker images (relative to your original training images) when deployed at testing time.

  This is very important, in particular for finetuning over small datasets.
- **Getting the scale of relevant objects right – fixed scale augmentations:** another example of a fixed, non-randomized augmentation for images (will see that in a later lecture):
  make the image size such that relevant structures (e.g. cats, cars) can be identified well.

A neural network for images has convolutional kernels with a fixed size. During training their weights are learned and adapt to some structures.

The sizes of input images at test time must be such that ... the structures to be identified at test time (e.g. cats, cars) – have similar size as – structures used at training time.

If your training set contains only screen filling cat faces,



then dont complain that you will not be able to predict cat correctly on this:

Your localized convolutional kernels were simply not trained to find structures on such small scales!
but data augmentation (or training a bounding box detector) can help with that – you can classify over many small windows.

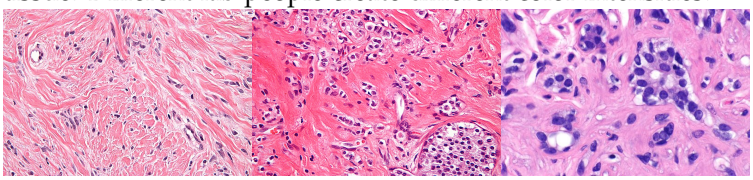
Using data augmentation is a fundamental standard in deep learning. Unfortunately, many tutorials, which focus on technical aspects ("how to program using toolbox X") skip this.

examples of data augmentation at training time for images:

- geometric distortions: distort an MNIST digit. Not used often outside of MNIST or similar shape recognition tasks.
- the most common type are: photometric distortions: modify brightness, and contrast/gamma of an image. for some scenario (e.g. medical stainings) one can think also to play with hues (colors).

An example where playing with hues a little bit may make sense: Below is the same stain Haematoxylin and Eosin, of the same

tissue. Different lab people create different color intensities!



For one quick intro on photometric distortions : Andrew Howard, Some Improvements on Deep Convolutional Neural Network Based Image Classification `https://arxiv.org/pdf/1312.5402.pdf`. See also the googlenet paper.

- distortions depends typically on some parameter. One needs to train with multiple choices of the parameter, and find the best parameter by looking at validation dataset errors. Once all parameters have been selected, then one does one final evaluation on the test data set – in order to check that one did not overfit by optimizing parameters by looking at the validation error.

Data augmentation is used at training time in a randomized fashion – because mild randomization often helps against overfitting.

- at training time: *randomized transforms* – The randomization depends on the transformation. Example: For image brightness, one choose a random multiplier within some range, like $[0.8, 1.1]$, every time an image batch is loaded.
- at training time: often one keeps an image as it is with p probability, and applies randomized transforms with $1 - p$ probability. One does usually not apply one choice of such parameter, but every time an image batch is loaded, one uses a randomly chosen parameter.
- **one does usually NOT apply these distortions on the validation or test set !! (except one wants averaging over multiple views)**

Task:

Take `alexnetcode.zip`. It contains a file `alexnet_1x_val_centercrop.py` . This file takes some images from the imagenet validation dataset and computes the prediction performance using only one centercrop.

In order to get it running, you need to set

- synsetfile - the full path to `synset_words.txt`
- impath - the path where the untarred images lie in
- xmlpath - the path where the unpacked xml-labels lie in

- At first compute the prediction performance (top-1 and top-5 error) for: centercrops of images with the smaller side being 250, and then centercrops of images with the smaller side being 227 which is the network square input dimension. This size is governed by:

  ```
  im=preproc_py2(imname,250)
  ```

- Implement: create a copy of `alexnet_1x_val_centercrop.py` implement four edge crops, compute the prediction preformance when averaging the network outputs over the center and the four edge crops.

  Hint: you can set the batchsize at test time to 5, and do a sess.run for all the crops of one image.

- implement mirroring the image left-right (you can do that in the numpy array). Now compute the average of five crops, and five mirrored crops.

- compare the prediction performance of: center crop, 5 crops, $5 + 5$ mirrored crops

- submit this as homework!

- For what datasets mirroring is a bad augmentation? traffic signs

Another take away: prediction with a neural net runs fairly fast. it is not an absolute must to use GPU for prediction.