50.021 – AI

Alex

Week 04: 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.]

Due: as announced

1 In class Problem 1 - Task: Test the performance of a pretrained net – simple crop.

Take the 2500 Imagenet val images, unpack them. the labels are in `ILSVRC2012_bbox_val_v3.tgz`. `getimagenetclasses.py` has example routines to get the label for one image.

- write a dataset class for this dataset (you need to implement in the class three methods: __init__(...) for storing all the knowledge that you need, then __len__() for the number of samples per epoch to be returned, then __getitem__(index) for returning a crop of the index-th image).
  - Suggestion: do not load all images in the __init__ method of the dataset class. That does not scale well if you have 500000 images :) . load the filenames and the labels instead into a list or the like. load an image in `getitem` of your Dataset-derived class!
- rescale the images so that the smaller side is $s = 224$ and perform a center crop of 224×224
  - do this one time using pytorch transforms inside the dataset class
    - load an image into an instance of PIL.Image
    - apply your composed set of pytorch transforms and return the pytorch tensor
  - do this one time by hand by the following steps:

        * loading an image into PIL.Image,
        * then run PIL.Image.resize(), to resize the smaller side to 224

          use PIL.Image.BILINEAR in PIL.resize, otherwise you get a big difference `https://github.com/pytorch/vision/blob/master/torchvision/transforms/functional.py`
        * then reads it into a numpy array,
        * normalize subpixel values to be in $[0, 1]$
        * then you need to swap axes from (h, w, c) to $(1, c, h, w)$,
        * then you apply proper neural network input normalization (see pytorch code for the standard imagenet values),
        * then you crop the center in numpy – check if it uses floor, ceil or round to determine coordinates!!
        * then you convert it by `torch.tensor(...)` into a pytorch tensor,
    - call your two dataset classes `dl1[k]` and `dl2[k]` with the same image index k and compare that their output is the same up to floating point problems.
    - compare the difference when removing `PIL.Image.BILINEAR` in the resize(). Its quite trashing the numbers, isnt it ??

- initialize the weights so that you load weights from the so-called model zoo. pytorch model zoo or gluon model zoo.

- compare performance to the case when you do not subtract the mean and normalize the subpixels. If too slow, use only the first 250 or first 500 images.

- hint: use a net with little parameters, avoid VGG, Alexnet.

Learning goals: dataloader for a custom dataset, prediction with a pretrained neural network.
Another take away: prediction with a neural net runs fairly fast. it is not a must to use GPU for prediction.

2 In class Problem 2 - Task: Test the performance of a pretrained net – five crop

- Take the code from Problem 2, now rescale smaller side to 280 (or 256) and implement a five crop as shown above.

- Compute for every image the average probability vector of the five predictions over the five crop. If too slow, use only the first 250 or first 500 images.

- You **can** use pytorch's five crop, or write a dataloader which does the job by
  - loading an image into PIL.Image,
  - then run PIL.Image.resize(), to resize the smaller side to 224
  - then reads it into a numpy array,
  - normalize subpixel values to be in $[0, 1]$
  - then you needs to swap axes from (h, w, c) to $(1, c, h, w)$,
  - then you apply to each crop proper neural network input normalization (see pytorch code for the standard imagenet values),
  - then you crop the corners and the center in numpy,
  - then you convert it by `torch.tensor(...)` into a pytorch tensor,
  - then you convert it into a valid pytorch tensor of shape $(1, 5{*}3, cropsize, cropsize)$
  - and in your code you sample it, and compute an average of 3-dim slices of that 15-channel image
- Compare performance of the average over five crops to using just the centercrop.
- Consider the tencrop: this is five crop with vertical mirroring for five of the ten. For what datasets mirroring is a baaad augmentation idea?

Leaning goal: You should observe improved performance, without training anything!

3 In class Problem 3 - Task: Different input size of the neural network (very easy).

Try to classify a few images with two different pretrained neural network architectures with an input size of 330×330 .
If it does not work out of the box (as of summer 2019 most neural nets in pytorch use AdaptiveAvgPool2D / AdaptiveMaxPool2D before the fully connected layer(s), so they should not cause problems with network resizing), then take the network, and create a derived class from it. Modify in the derived class the average/max pooling so that it works with an input size if 330×330 . Classify with 5-crop of size 330 as in the class problem 2.

4 Homework

Finish the above In class Problems 1,2,3 and submit them as homework. Submit the code and report prediction accuracies (simple classification error here does the job).