50.039 – Theory and Practice of Deep Learning

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.]

Learning objectives:

- custom data loaders
- prediction with a pretrained neural network
- one can use a deep neural network with a different input size than the specified size
- a simple way to achieve better performance is to perform multiple crops of one image, compute for each crop a prediction, then to average the crops.

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

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 dataloader for this dataset.
  - 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

- initialize the weights so that you load weights from the so-called pytorch 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, report the performance difference
- hint: use a net with little parameters, avoid VGG, Alexnet.

2 In class Problem 2 - Task: Test the performance of a pretrained net – five crop (see below if you cannot use a five crop transform)

- Take the code from Problem 2, now rescale smaller side to 280 (or 256) and implement a five crop as shown above.
- For every of the five crops, compute the prediction.
- 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.
- Compare performance of the average over five crops to using just the centercrop.
- **Bonus (+20% of the whole points of this task set)**: reimplement your own five crop, and reimplement your own `ToTensor()` and `Normalize(mean,std)` from PIL Image. (you can use the step through numpy)

  one way: You output for one image a 4-dim tensor in your dataset class (5, c, h, w) instead of a 3-dim tensor (c, h, w). The dataloader will then turn it into 5-tensor (batchsize, 5, c, h, w) which you can then decompose when you run a for loop over outputs of your dataloader class.

  Regarding using normalize and transforms:

  Either: You can use the way to read the image by e.g. PIL, run all the pytorch transforms including ToTensor(), then create crops on that object same as you would play on a numpy array.

  Or if you want to use transforms in the dataloader, then : for ToTensor() you need to use a trick like

```
normalizer=transforms.Normalize(mean=[0.485, 0.456, 0.406],  std=[0.229, 0.224, 0.225])
data_transform = transforms.Compose([
    transforms.Lambda(lambda crops:
        torch.stack([normalizer(transforms.ToTensor()(crop)) for crop in crops]) )
    ])
```

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

3 In class Problem 3 - Task: Different input size of the neural network.

Try to classify with two different pretrained neural network architectures with an input size of 330×330 .
If it does not work out of the box, then take the network, and create a derived class from it. change in the derived class the last average/max pooling into an Adaptive pooling. Then it will work with an input size if 330×330 .

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).

The dataset `imagenet2500.tar` is in: `https://www.dropbox.com/sh/1lbees2cn34ysse/AAB9VKqNFMJSmzdtZgaU1s9ua?dl=0`