Home Work 1

Abbas Nosrat
810199294

April 10, 2022

Contents

1 Preface

Local contrast normalization is a type of normalization layer which employs local subtractive and divisive normalizations. First of all a Gaussian kernel is formed. The sum of all elements of this kernel is equal to one. This property causes convolution with an image to be just a weighted mean. Figure one demonstrates the contours of the Gaussian kernel.

The image is convolved with the kernel and result is subtracted from the original image. This operation brings the image mean close to zero. To construct the divisive part of the normalization, the subtracted image is convolved withe the gaussian kernel again. Each element of the result is compared to the mean of the result and the substituted by the mean if the element is less than the mean. To avoid division by zero, each element is substituted with a small number if it is equal to zero. Finally, the new image is equal to the filtered-subtracted image divided by the divisive part. Figure 2 demonstrates the input and output of the LCN layer.

The LCN layer enforces a local competition between pixels of the same feature map and pixels with the same spatial location across other feature maps. This competition results in an increase in the model robustness. According to the paper, the mathematical formulation of the LCN layer is as follows:

$$v_{i,j,k} = x_{ijk} - \sum_{ipq} w_{pq}.x_{j+p,k+q}$$

$v_{i,j,k}$ is the aforementioned filtered-subtrated image and w_{pq} is the kernel weights.

$$y_{ijk} = v_{ijk}/max(c, \sigma_{jk})$$

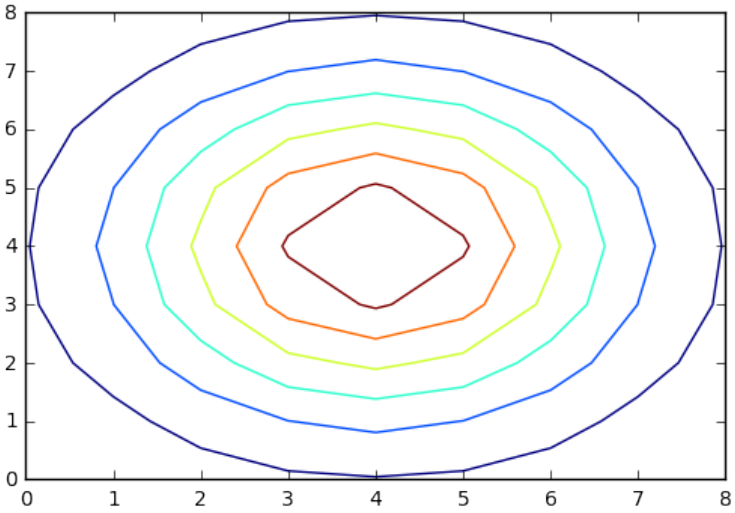
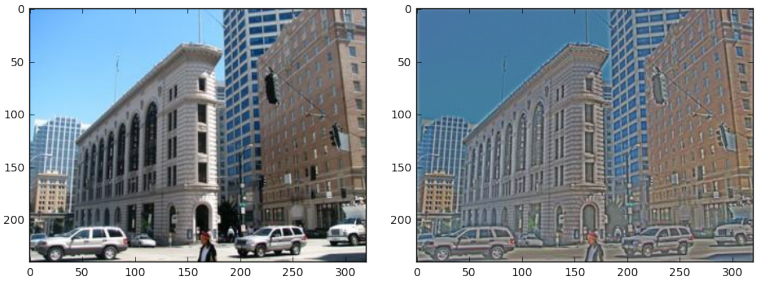


Figure 1: contours of the Gaussian kernel



(a) Input image (b) Output image

Figure 2: input and output of LCN

$y_{ijk}$is the output of LCN, $c = mean(\sigma)$ and σ is the convolution of v and the kernel. The formula above is the divisive part of LCN.

The model used in this homework is YOLOX and the dataset is VOC.

2 Problem 1

The model code was cloned from the YOLOX github repository. The dataset was downloaded via pytorch datasets and moved in the datasets directory inside YOLOX directory. After lots of tinkering with the code, max epochs was reduced to 50 and LCN implementation was added as a prepossessing to the data loader. This addition is possible since LCN has no trainable parameters and can act as a simple normalization on the data. It would have been more efficient if LCN was performed on the dataset before training since it had two convolutions which were ran on CPU which caused training time to greatly increase. The model was trained for 50 epochs. LCN is implemented at YOLOX/yolox/data/datasets/mosaicdetection.py. To run the experiments, simpy run:

```
python tools/train.py -f exps/example/yolox_voc_s/yolox_voc_s.py -b 4 --fp16
```

for training without LCN and

```
python tools/train.py -f exps/example/yolox_voc_lcn/yolox_voc_lcn.py -b 4 --fp16
```

for training with LCN. The aforementioned commands must be ran in YOLOX directory. After training, run printer.py to extract loss plot from training logs which is a text file located at "YOLOX/YOLOX_outputs". Figure 3 demonstrates how the loss decreases both with and without LCN.

As witnessed in Figure 3,training loss decreases almost normally with LCN slightly being in the lead. However, as the model trains more, the results get worse with the introduction of LCN. The model logged MAP for validation data with tensor board. Figure 4 contains tensor board results.

There is an odd behavior in MAP with LCN which cannot be explained but overall, the model performance is worse with LCN. However LCN increases model robustness and the model would perform better in real life scenarios.

Training time was 13 hours with LCN and 6 hours without LCN. Due to the high training time (on two GTX1050tis), the experiments were ran only once hence these results could be due to the stochastic nature of training.

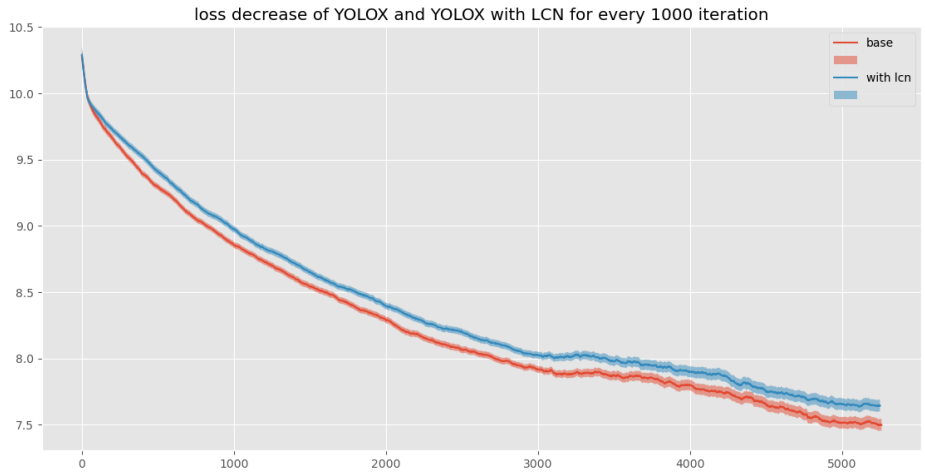
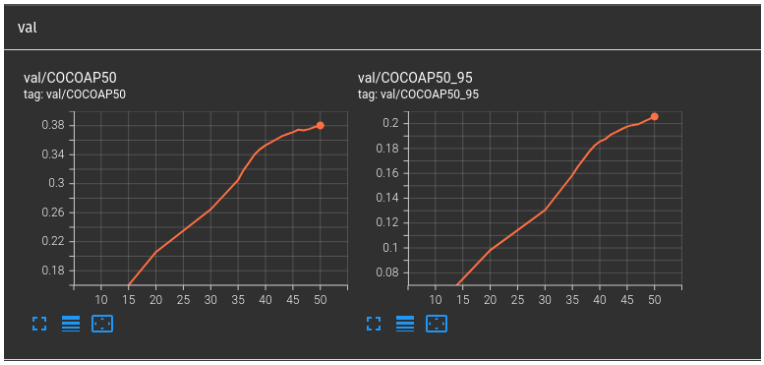
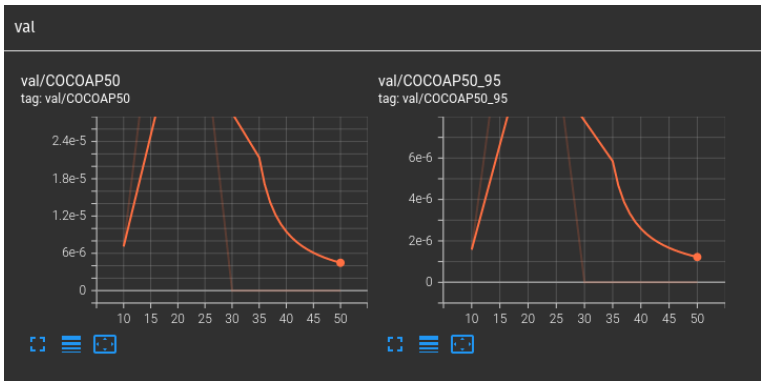


Figure 3: training loss for normal training and training with lcn



(a) normal



(b) with LCN

Figure 4: Tensor board logs