Programming Assignment 2

Convolutional Neural Network

SE395: Introduction to Deep Learning

Objective

- (Main) Convolution Neural Network for Classification without deep learning framework

- (Extra credit) Visualize the activation map using Class Activation Map using the author provided code [1]

[1] https://poddeeplearning.readthedocs.io/ko/latest/CNN/VGG19%20+%20GAP%20+%20CAM/

Overall steps

1. Prepare the training and test datasets (MNIST)
2. Design *n-layer* ***Convolutional Neural Networks*** from scratch & using pre-defined layers (*n=2,3*)
3. Design the training process and train the network
4. Test the network on the test data and visualization the results on the report using ***Tensorboard***
5. Extra credit (30% of total) – Class Activation Map (CAM)

Refer: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

1. Prepare training/test dataset

1. **Download MNIST datasets**
   1. Download link: http://yann.lecun.com/exdb/mnist/
      1. train-images-idx3-ubyte.gz: training set images (9912422 bytes)
         train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
         t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
         t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)
   2. Data load: https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/mnist/download/

2. **Prepare the datasets for training**
   1. Ex) normalization

2. Design four Convolutional Neural Networks

1. **Design (1)** ***2-layer CNN*****, (2)** ***3-layer CNN from scratch (No deep learning framework)***
    1. 2-layer sequence: Conv-ReLU-MaxPool - Conv-ReLU-MaxPool - Linear-SoftMax (The input and output size of NN: input 28x28, output 10)
    2. 3-layer sequence: Conv-ReLU-MaxPool - Conv-ReLU-MaxPool - Conv-ReLU-MaxPool - Linear-SoftMax (The input and output size of NN: input 28x28, output 10)
2. **Design (3)** ***2-layer CNN*****, (4)** ***3-layer CNN*** **using predefined layers from deep learning framework (ex, pytorch, tensorflow etc.)**
    1. Such as nn.Conv2D, nn.Linear, nn.MaxPool2D

You can use the sub-layers (Linear/ReLU/SoftMax) designed for your PA1.

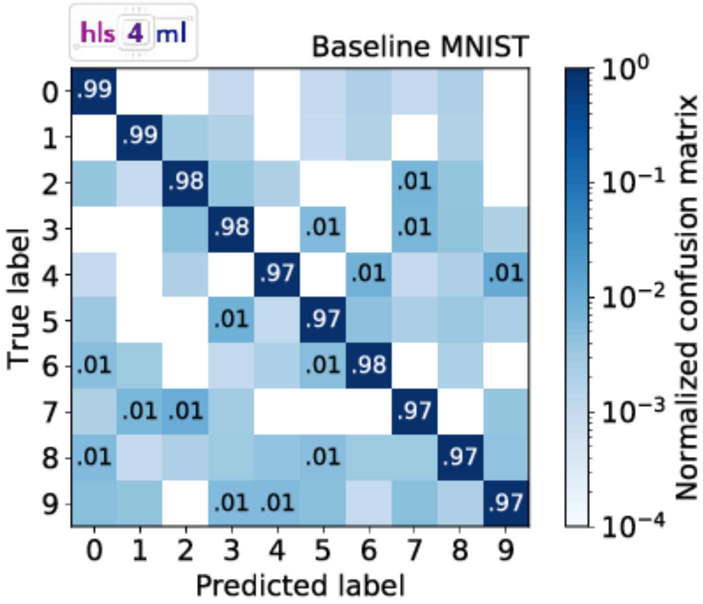
3. Design the training process and train the network

1. Initialize the model parameters
2. Implement and do forward propagation
3. Implement and compute the cross-entropy loss
4. Implement and do backward propagation
5. Implement and update model parameter using gradient descent (SGD)
6. Draw the plot of the loss

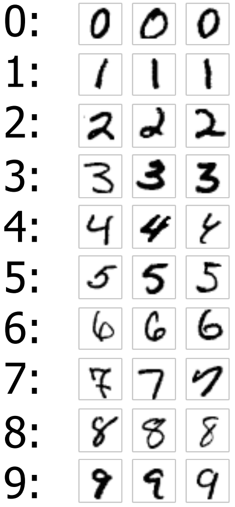
I guess you only need to design the Conv layer and its backpropagation

4. Test the network on the test data and visualization the results *with Tensorboard*

1. **Show 10x10 confusion matrix**
   1. the probability of classification results for all classes (*No Tensorboard*)
2. **Show top 3 scored images with probability (for each class)** *with Tensorboard*
3. **Show training Loss graph** *with Tensorboard* (Train & Validation)



1. 10x10 confusion matrix

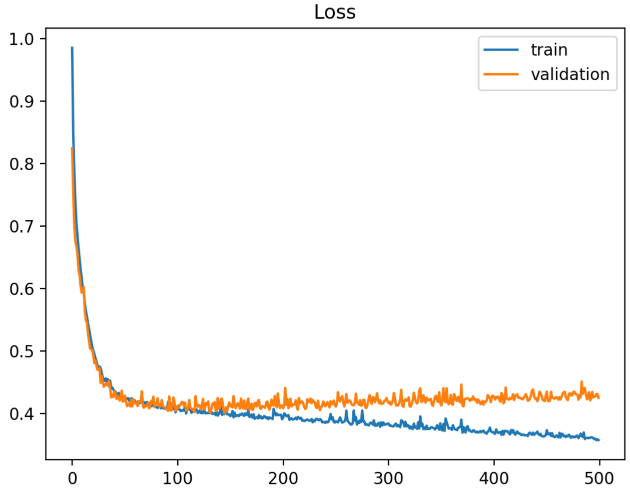


100%, 99.8%, 99.7%

⋮

100%, 99.9%, 99.9%

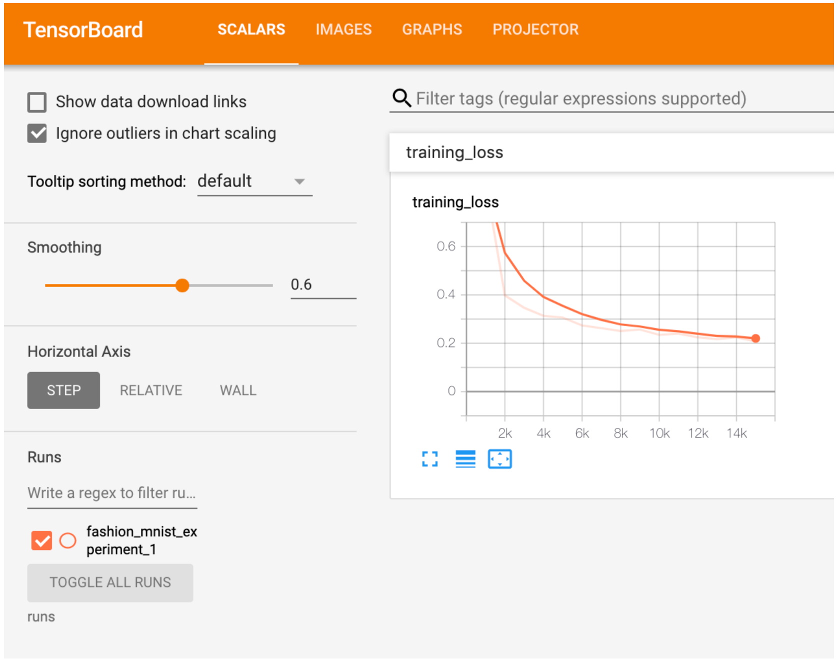
2. Top-3 images with probability



3. Loss graph

4. Test the network on the test data and visualization the results *with Tensorboard*

1. **Draw the training loss graph on Tensorboard**
2. **Display the top 3 scored images on Tensorboard**
    1. https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html



□ Show data download links

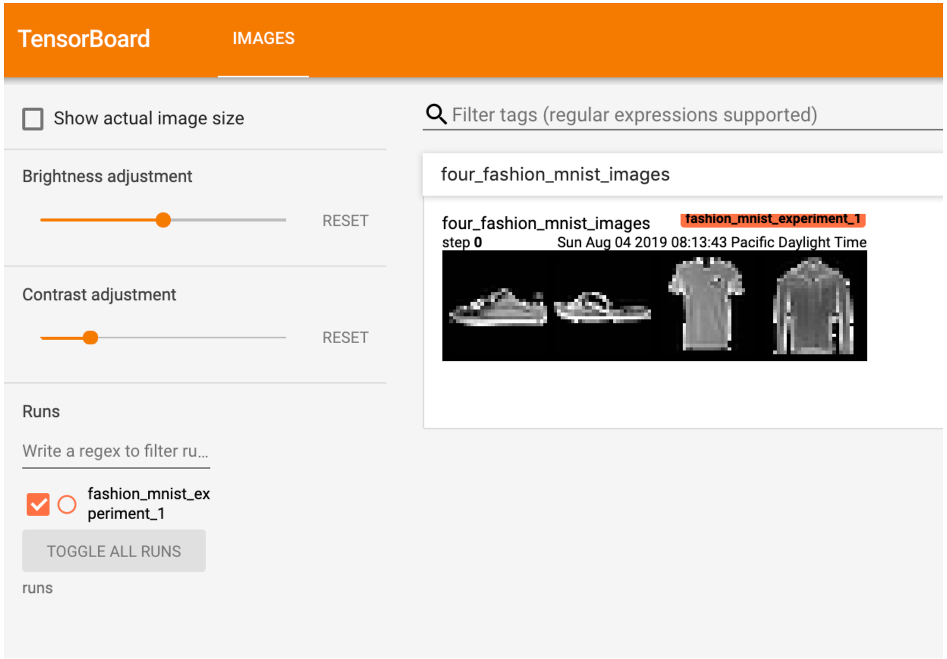
☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP

☑ ○ fashion_mnist_experiment_1



□ Show actual image size

four_fashion_mnist_images fashion_mnist_experiment_1

step 0 Sun Aug 04 2019 08:13:43 Pacific Daylight Time

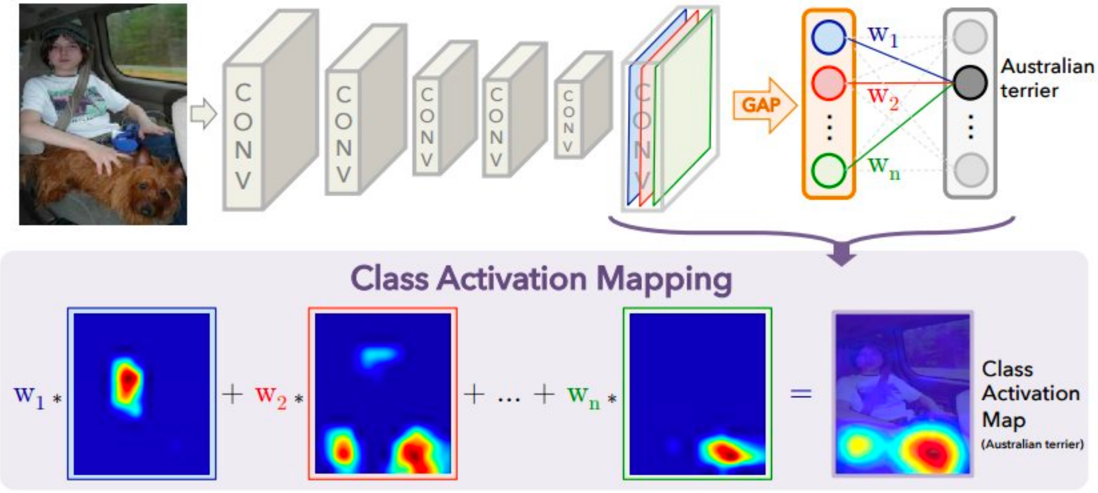
☑ ○ fashion_mnist_experiment_1

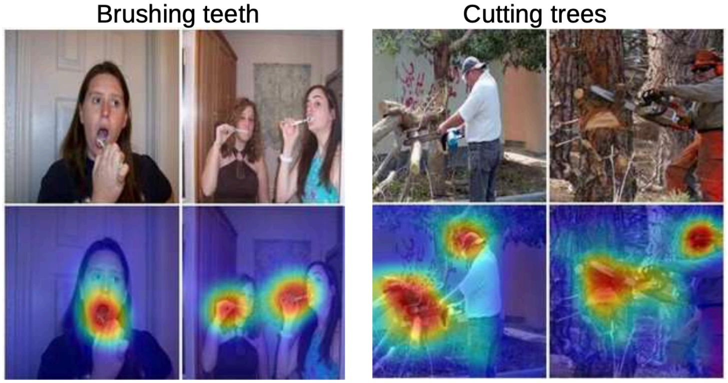
TensorBoard
IMAGES
Filter tags (regular expressions supported)
four_fashion_mnist_images
Brightness adjustment
RESET
Contrast adjustment
RESET
Runs
Write a regex to filter ru...
TOGGLE ALL RUNS
runs

If you do not use Tensorboard for visualization, your score will be degraded

5. Extra Credit– Class Activation Map (CAM)

1. **Visualize the class activation map**
    1. CAM allows the classification-trained CNN to both classify the image and localize class-specific image regions in a single forward-pass
    2. Paper:http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf
    3. Code:https://poddeeplearning.readthedocs.io/ko/latest/CNN/VGG19%20+%20GAP%20+%20CAM/





Top: Input image
Bottom: Activation map

Submission

Due : Nov 13, 11:59PM
To : lms.dgist.ac.kr

1. Submission should include (1) Source code, (2) Report

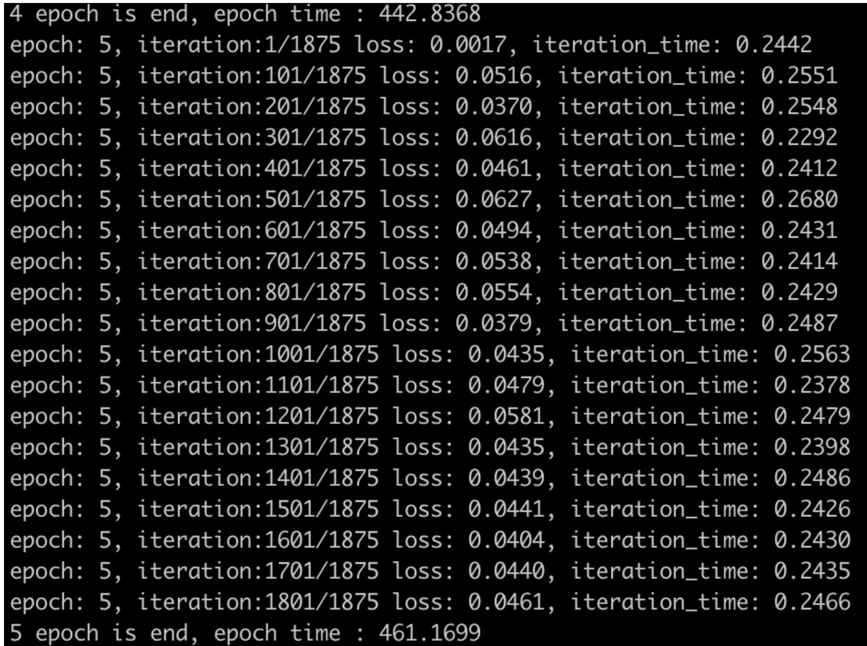
2. Report should include the results and results comparison

   1. For all networks *(1)* ***2-layer CNN***, *(2)* ***3-layer CNN***, *(3)* ***2-layer CNN using DL-framework***, *(4)* ***3-layer CNN using DL-framework*** and (5) ***3-layer NN with ReLU (PA1)***, show the results and compare
      (a) 10x10 Confusion Matrix,
      (b) Top 3 score images (all classes), using Tensorboard
      (c) Training Loss graph using Tensorboard

   2. (Optional – extra 30%) Visualize four Class Activation Map (CAM) from above four CNNs (1)-(4) with the same sample.

Training setting & outputs (for my case)

- **Setting (3-layer CNN)**
  - CPU / batch_size: 32 / epoch: 5 / 60000 training datasets
- **Elapsed time**
  - One iteration: about ***0.25s***
  - One epoch: about ***7.8min (0.25s x 1875 iteration)***
  - Total training: ***39min (7.8min x 5)***
- **Accuracy**
  - 1-epoch: 96.14%
  - 2-epoch : 97.93%
  - 5-epoch : 98.57%
  - 10-epoch: 98.72%



```
4 epoch is end, epoch time : 442.8368
epoch: 5, iteration:1/1875 loss: 0.0017, iteration_time: 0.2442
epoch: 5, iteration:101/1875 loss: 0.0516, iteration_time: 0.2551
epoch: 5, iteration:201/1875 loss: 0.0370, iteration_time: 0.2548
epoch: 5, iteration:301/1875 loss: 0.0616, iteration_time: 0.2292
epoch: 5, iteration:401/1875 loss: 0.0461, iteration_time: 0.2412
epoch: 5, iteration:501/1875 loss: 0.0627, iteration_time: 0.2680
epoch: 5, iteration:601/1875 loss: 0.0494, iteration_time: 0.2431
epoch: 5, iteration:701/1875 loss: 0.0538, iteration_time: 0.2414
epoch: 5, iteration:801/1875 loss: 0.0554, iteration_time: 0.2429
epoch: 5, iteration:901/1875 loss: 0.0379, iteration_time: 0.2487
epoch: 5, iteration:1001/1875 loss: 0.0435, iteration_time: 0.2563
epoch: 5, iteration:1101/1875 loss: 0.0479, iteration_time: 0.2378
epoch: 5, iteration:1201/1875 loss: 0.0581, iteration_time: 0.2479
epoch: 5, iteration:1301/1875 loss: 0.0435, iteration_time: 0.2398
epoch: 5, iteration:1401/1875 loss: 0.0439, iteration_time: 0.2486
epoch: 5, iteration:1501/1875 loss: 0.0441, iteration_time: 0.2426
epoch: 5, iteration:1601/1875 loss: 0.0404, iteration_time: 0.2430
epoch: 5, iteration:1701/1875 loss: 0.0440, iteration_time: 0.2435
epoch: 5, iteration:1801/1875 loss: 0.0461, iteration_time: 0.2466
5 epoch is end, epoch time : 461.1699
```

Notice

Due : Nov 13, 11:59PM
To : lms.dgist.ac.kr

1. **Delayed submission**
   1. 25% score will be degraded every 1-day delay & after 3 days delayed, you will get 10% of total score
      (e.g., 100% → 75% (1day) → 50% (2day) → 25% (3day) → 10% (> 3day)

2. **Plagiarism**
   1. No grade for copied codes (from friends and internet)
   2. You can refer source from internet, but do not copy and paste.

3. **Partial credit**
   1. Even though you are not successfully design the network and obtain reasonable result, please send your code.
   2. There will be partial credit for each module implementation.