Lab 2: Convolutional Neural Networks, Custom Data loader and Transfer Learning

ENGN8536, 2021

August 20, 2021

In this lab we will be working with PyTorch to address a binary classification task of cats and dogs. The main goal for this lab is for you to develop a deep network in PyTorch, and tune it to work effectively, write a data loader for custom datasets, to build and train a convolutional neural network and to fine-tune a pre-trained model.

Make good use of the PyTorch tutorials and documentation as listed in the Resources section, or any other resources to assist you to complete the lab task.

For using the Student GPU server at CECS, see *Server Guideline 2021* on Wattle.

1 Custom Data Loader (3 Marks)

1. Download the cats and dogs dataset from here: https://drive.google.com/file/d/15tR7fhOwxsU7HJm292E4Ruq3hfCINKIX/view?usp=sharing. If you are using the student GPU server (MLCV1), you can find the data in /courses/engn8536/Datasets/Cat-Dog-data.

2. You are required to use the PyTorch DataLoader to load, read and split the dataset. Split the data into training set (18000 samples), validation set (2000 samples) and test set (4000 samples). (1 mark)

3. Generate a list of labels according to filenames. Use number 0 and 1 to denote labels cat and dog respectively. (0.5 mark)

4. Use PyTorch torchvision to complete Q4 & Q5. When loading the data (0.5 mark):

   - resize the image to 224x224
   - normalize the data to range between (-1, 1)

5. Perform the following data augmentation during **training** (1 mark):

   - randomly flip the image left and right with 50% probability
   - zero-pad 4 pixels on each side of the input image and randomly crop 224x224 as input to the network

2 Implement a CNN (8 Marks)

1. Build a CNN with the following architecture (4 mark: Q1, Q2 & Q3):

   - 3×3 Convolutional Layer with 32 filters, stride 1 and padding 1.
   - 3×3 Convolutional Layer with 32 filters, stride 1 and padding 1.
   - Batch Normalisation Layer.
   - ReLU Activation Layer.
   - 2×2 Max Pooling Layer with a stride of 2.
   - 3×3 Convolutional Layer with 64 filters, stride 1 and padding 1.
   - Batch Normalisation Layer.

- ReLU Activation Layer.
- 2×2 Max Pooling Layer with a stride of 2.
- Fully-connected layer with 1024 output units.
- Batch Normalisation Layer.
- ReLU Activation Layer.
- Fully-connected layer with 1 output units (or 2 output units, if you prefer).

2. Set up a binary cross-entropy loss (BCEWithLogitsLoss). Feel free to use other loss functions, and modify the last layer of the network if necessary.
3. Set up an Adam optimizer. Feel free to use other optimizers.
4. Train your model with customize hyperparameters (e.g. batch size, learning rate, number of training epochs). Draw the following plots, describe and explain your findings (3 marks):

   - Training loss vs. epochs.
   - Training accuracy vs. epochs.
   - Validation loss vs epochs.
   - Validation accuracy vs. epochs.

   Test your model on the test split once your training is finished. You can either use *Tensorboard* to draw the plots, or you can save the data first then use *Matplotlib* to plot the curve.
5. Train a better model that further improves the performance. You may need to investigate the following options: hypeparameter, different options for settings; Network Architecture, e.g. Batch Normalisation and Dropout. You can use the literature and experimental results to justify your decisions. More marks will be given to better models.

3 Transfer Learning (4 Marks)

The idea of transfer learning is to take a network that is pre-trained on a large-scaled dataset (e.g. ImageNet) and then fine-tune the network on our own dataset, in this case, cat and dog images. It allows us to use a powerful feature extractor as a "backbone" network and saves us the effort of training our own deep network from scratch, which often takes a long time to converge and requires large computational resources.

1. Load a pretrained ResNet-18 using torchvision (feel free to use other networks such as DenseNet, AlexNet or VGGNet). (0.5 mark)
2. Modify the final fully-connected layer so that it fits our classification task. (0.5 mark)
3. Train the model. (1 mark)

   If you don't have sufficient computational power or if you want to train faster, you can freeze all previous layers except the final fully-connected layer, i.e., only fine-tune the classification layer.
4. Draw plots (as in Task 2.4) and compare them with the previous results (Task 2.4), describe and explain your findings. (2 marks)

4 Resources

- PyTorch data loading tutorial:
  https://pytorch.org/tutorials/beginner/data_loading_tutorial.html
- PyTorch training a classifier tutorial:
  https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- PyTorch 1.6.0 documentation:
  https://pytorch.org/docs/stable/index.html
- PyTorch torchvision documentation:
  https://pytorch.org/docs/stable/torchvision/index.html

5 Submission

1. The lab will be due on Sunday, 5 Sept 2021, 6:00pm. Please submit a lab report that clearly documents all the implementation details and results, and attach all your code in Appendix. Name your submission as Lab_2_u0000000.pdf, replacing u0000000 with your uni-ID.

2. The full grade of this lab is 15 marks. This Lab worth 7.5% of the total course assessment.

3. The page limit is 4 excluding the appendix. Only content within the page limit will be assessed.