MIT CSAIL

6.819/6.869 Advances in Computer Vision

Fall 2022

Miniplaces

Posted: Tuesday, Mar 15, 2022 **Due:** Tuesday, Mar 29, 2022

Note: 6.869 and 6.819 students are expected to finish all problems unless there is an additional instruction.

We provide a python notebook with the code to be completed. You can run it locally or in Colab (upload it to Google Drive and select 'open in colab') to avoid setting up your own environment. Once you finish, run the cells and download the notebook to be submitted.

Submission Instructions: Please submit a .zip file named <your kerberos>.zip containing 1) the python notebook provided, with the cells run and the relevant source code and 2) your trained model file for Problem 4 at the root of your submission (zip both files instead of a folder).

Late Submission Policy: If your pset is submitted within 7 days (rounding up) of the original deadline, you will receive partial credit. Such submissions will be penalized by a multiplicative coefficient that linearly decreases from 1 to 0.5.

In this problem set, you will be experimenting with neural networks using PyTorch. To speed up training, you will need access to a GPU. We recommend using Colab to run the experiments, since it comes with GPU support. To enable it simply do:

Runtime - Change Runtime type - Hardware accelerator - Gpu.

Colab has recently introduced some restrictions surrounding GPU usage. Don't worry if you can't get one – training should only take 5 minutes on a decent CPU anyways.

Problem 1 *Filter Visualization* (2 pts)

Convolutional kernels in different layers are used to extract information from the input image in different levels. In this problem, we will focus on visualizing the filters of the ResNet network pretrained on Places365 Dataset. You need to download the model and the pretrained weights first.

(a) (1 pt) Implement `normalize_tensor` to normalize the input kernel for better visualization in the `visualize_filters` function.

(b) (1 pt) Visualize filters for another convolutional layer in ResNet.

Problem 2 *Internal Activation Visualization* (2-5 pts)

In this problem, we will visualize the activations of the internal units in the model. We will chop the fully connected layers of ResNet and visualize the activations of the last convolutional layer into different locations of the input image.

(a) (1 pt) Create a version of the ResNet model without the last two layers to expose the last convolutional layers in `generate_featuremap_unit` function.

(b) (1 pt) As you can see, the unit 300 activates the mountain part of the image. Find other units that detect 1) sky and 2) building in the image.

(c) (1.5 pts) **(6.869 required; optional for 6.819)** Each unit contributes differently to the final prediction. The contribution weight for each unit is determined by the weights of the fully connected layer (last layer in ResNet). If we deactivate top-5 units (force the kernel to be zeros) that have the maximum weights for the top 1 category of some input image, the prediction for that category will drop dramatically. The code to find the index of those units is given, try to deactivate those units and compare the difference between the original and new top 5 predictions. **Show the feature map of that unit.**

(d) (1.5 pts) **(6.869 required; optional for 6.819)** In (c), we deactivate top-5 units and observe dramatic changes in predictions. Try to deactivate fewer units and. **Report the lowest number of dropped units required to change the top prediction class.**

Problem 3 *Class Activation Map (CAM)* (3 pts)

So far, we know the output of the last convolutional layer activate different parts of the input image. In this problem, we will explore how to visualize which parts of the image are responsible for the final decision. Use the chopped ResNet from problem 2 as the model.

(a) (1 pt) Running the chopped ResNet will produce a tensor with the size of $(1 \times 2048 \times 8 \times 8)$. 1 is the batch size (b), 2048 is the number of channels (c) and 8 & 8 is the height (h) and width (w) of the kernel. Convert the output tensor to a new tensor with the size of $(hw \times c)$.

(b) (1 pt) Feed the new tensor from (a) to the fully connected layer of ResNet to compute the weighted average. You will get a output tensor with the size of $(hw \times 365)$, where 365 is the number of classes in the Places365 Dataset.

(c) (1 pt) show the feature maps (combined with input image) of the top 5 predicted categories and explain the relationship between the feature activation and the corresponding category.

(d) *(Bonus, 0.5 pts)* Try running this visualization method on another image of a scene. Is the visualization still effective for your image? Explain why or why not.

Problem 4 *Network Training* (3-5 pts)

The goal of this problem is to train a small convolutional neural network to classify images

of clothing items from the FashionMNIST dataset. You'll first fill in critical components of a simple PyTorch training pipeline, evaluate the model on the test set, and explore the impact of specific design choices and hyperparameters on the model's performance.

(a) (1 pt) Read and execute the notebook cells until you reach the training loop section. Here you will complete the function `train` that trains a network for 1 epoch by filling in the missing code snippets.

(b) (1 pt) Implement the `get_prediction` function which returns the index of the predict class given an image and a network and is called during the evaluation routine. Also implement the accuracy computation in the (evaluate) function

(c) (1 pt) Here, we bring all of the pieces together to train the network. Instantiate an optimizer to update the weights of the network during training. Run training and validation for 10 epochs and report your final validation accuracy.

(d) (2 pts) **(6.869 required; optional for 6.819)** We want you to get a feel for the impact of specific design choices on the performance of the network. Experiment with the following hyperparameters / techniques:

- Data augmentation
- Weight initialization
- Number of layers, or number of layer features
- Type of optimizer
- Learning rate and/or schedule
- Regularization

Save your model as described in the file. You will receive credit if your model attains $> 90\%$ accuracy.