University of Michigan
EECS 504: Foundations of Computer Vision
Winter 2020. Instructor: Andrew Owens.

Problem Set 5: Scene Recognition

Posted: Tuesday, February 11, 2020	**Due:** Tuesday, February 18, 2020

Please submit your solution to Canvas as a notebook file (`.ipynb`), containing the visualizations that we requested. Your `.ipynb` notebook should be named as `<unique_name>_<umid>.ipynb`. Example: `adam_01100001.ipynb`. Also, please remember to put your name and unique name in the first text block of the notebook.

The starter code can be found at:
https://drive.google.com/open?id=1rrsj-FkNEsxPXg8MXS4hDLZu7e1waci5

We recommend editing and running your code in Google Colab, although you are welcome to use your local machine instead.

Problem 5.1 *Scene Recognition*

In this problem set, you will train a CNN to solve the *scene recognition* problem, i.e., the problem of determining which scene category a picture depicts. You will train two neural networks, which we call `MiniVGG` and `MiniVGG-BN`. `MiniVGG` is a smaller, simplified version of the VGG [1] architecture, while `MiniVGG-BN` is identical to `MiniVGG` except that we added batch normalization layers after each convolution layer.

You will train the neural networks on the MiniPlaces dataset[1]. We'll use 90,000 images for training, 10,000 images for validation, and the remaining 10,000 images for testing. Sample images from MiniPlaces dataset (along with their categories) are shown in Figure 1.

Below is an outline for your implementation. For more detailed instructions, please refer to the notebook.

(a) We often train deep neural networks on very large datasets. Because it is impossible to fit the whole dataset into the RAM, loading the data one batch at a time is a common practice. We also often need to preprocess the data: common preprocessing steps include normalizing the pixel values, resizing the images to have a consistent size, and converting `numpy` arrays to PyTorch tensors. As the first step of this problem set, please build data loaders with the specified data preprocessing steps (for the specific preprocessing you need to perform, please see the comments in the provided notebook). You may also find the PyTorch tutorial[2] on data loading to be helpful. (2 points)

[1] https://github.com/CSAILVision/miniplaces
[2] https://pytorch.org/tutorials/beginner/data_loading_tutorial.html

clothing_store | yard | butchers_shop | martial_arts_gym
| baseball_field | swamp | phone_booth | butchers_shop
| martial_arts_gym | kitchen | temple | boat_deck
| ski_slope | airport_terminal | driveway | lobby

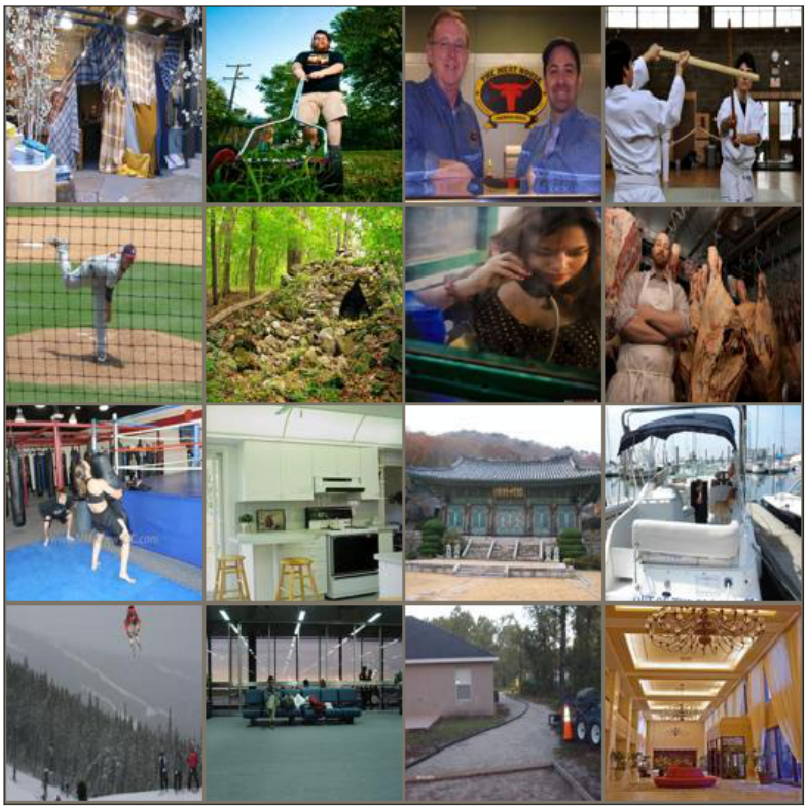

Figure 1: Sample images from MiniPlaces dataset.

(b) Construct the `MiniVGG` and `MiniVGG-BN` models. Make sure the neural networks you build have the same architectures as the ones we give in the notebook before you start training them. (3 points)

(c) Implement the training and validation loops. For the training loop, you will need to: i) compute the outputs for each minibatch using the neural networks you build, ii) calculate the loss, iii) update the parameters using the Adam optimizer [2]. Please use PyTorch's built-in automatic differentiation, rather than implementing backprogagation yourself. The validation loop is almost identical to the training loop except that we do not perform gradient update to the model parameters; we'll simply report the loss and accuracy. (3 points)

(d) Train the networks and visualize training and validation accuracy history. Comment on the effect of batch normalization. Please leave your comments in the text block we provide at the end of Step 4 section in the notebook. **Note: training each neural network will take about 25 minutes.** (1 point)

(e) Measure Top-1 and Top-5 accuracy on the test set (i.e. the probability that the true class appears as the most likely class, and within the top-5 most likely respectively) . To pass the test, both neural networks should have Top-5 accuracy above 50%. (1 point)

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.