University of Michigan
EECS 442/504: Computer Vision
Fall 2022. Instructor: Andrew Owens.

Problem Set 5: Scene Recognition

Posted: Wednesday, October 5, 2022 **Due:** Wednesday, October 12, 2022
Submission instructions:

- Canvas:
  - Colab notebook (`.ipynb`) containing solutions and visualizations for 5.1(a-e). Before submitting, please make sure to rename the file to `<uniqname>_<umid>.ipynb`.
- Gradescope:
  - `.pdf` version of Colab notebook (conversion instructions here). If these instructions do not work, look into this additional method. For your convenience, we have included the PDF conversion script at the end of the notebook.

The starter code can be found at:
`https://drive.google.com/file/d/1vbQEKK2cxCaETKR2J2F_8SaO0CTdUZO7/view?usp=sharing`

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 [2] 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) **(2 points)** 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, which includes common preprocessing steps like 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 fill in the indicated part for building data loaders with the specified data preprocessing

[1]`https://github.com/CSAILVision/miniplaces`

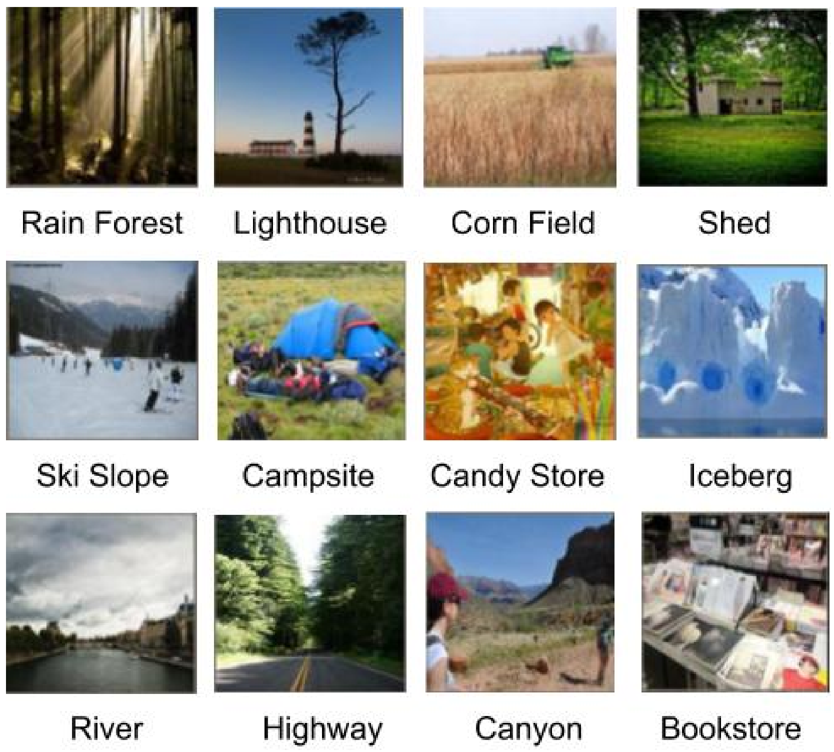

Figure 1: Sample images from MiniPlaces dataset.

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.

(b) **(3 points)** 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.

(c) **(3 points)** Implement the training and validation loops. For the training loop, you will need to implement the following steps: i) compute the outputs for each minibatch using the neural networks you build, ii) calculate the loss, iii) update the parameters using the SGD optimizer (with momentum). 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. Please see the notebook as well for further instructions.

(d) **(1 point)** Train the networks and visualize training and validation accuracy history for `MiniVGG` and `MiniVGG-BN` using the code we provide in the notebook. 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.**

(e) **(1 point)** If the correct label is within the `top_k` predicted classes according to the network output scores, we count the prediction by the neural network as a correct prediction. 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

[2] https://pytorch.org/tutorials/beginner/data_loading_tutorial.html

neural networks should have Top-5 accuracy above 55%.

Problem 5.2 *Residual Networks (EECS 504 only)*

We will now create a simplified version of the ResNet [1], which we will use to solve the scene recognition problem. To make the implementation easier, we will perform downsampling outside of the residual blocks, using strided convolutions. We have provided a description of the network in Fig. 2, and we have given you a partial implementation. Your job will be to complete it.

(a) **(2 points)** Implement the *residual block*. Each block applies two convolutions and uses a residual connection. We will implement this as a class `ResidualBlock`, which is parameterized with the number of channels.

(b) (**1 point**) Complete the implementation of the ResNet. Your task will be to add the initial layers of the network, and insert the residual blocks.

(c) (**1 point**) Train the model. Your Top-5 accuracy should be at least 55% after 20 epochs of training (i.e., after 20 full passes through the training data).

(d) (Optional, 0 points) Add extra residual blocks to the network, and measure the improvement in accuracy.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

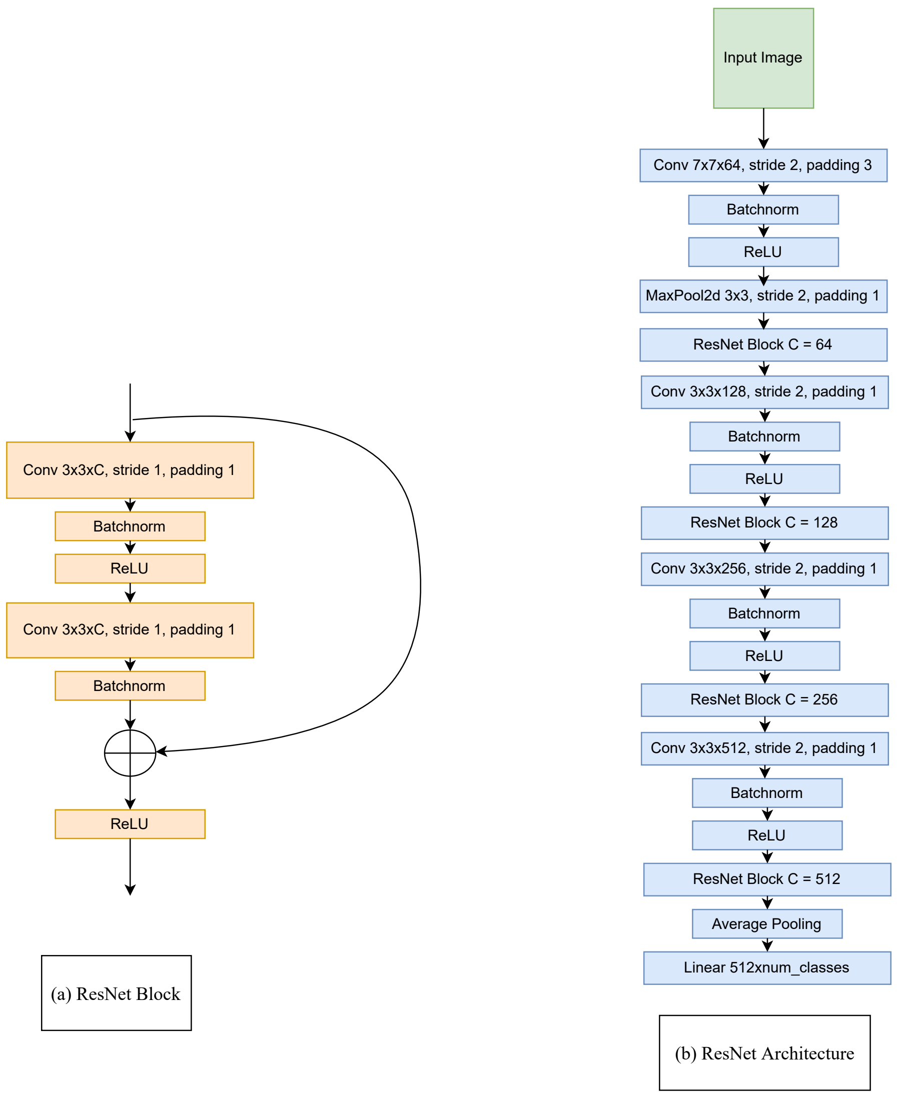

Figure 2: **Simplified ResNet.** You will implement a simplified ResNet. The network (right) reuses the residual block (left) with different numbers of channels, at each spatial scale.