Stony Brook University
CSE512 – Machine Learning – Spring 21
Homework 6, Due: Saturday 8 May 2021 at 11:59PM

This homework contains two questions. The first question asks you to follow PyTorch tutorials to learn about PyTorch. In the second question, you use a CNN for few-shot visual counting. The maximum score for this homework is 100 + 15 bonus points.

1 Learning PyTorch (40 points)

In this question, you will follow some tutorials, in order to learn PyTorch. Specifically, you will need to follow `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html` and `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html` and answer the following questions.

1.1 Tensors and Operations (8 points)

Follow the tutorial and make sure you understand each step: `https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html`

(a) After you follow this tutorial, construct a randomly initialized tensor x of size (3, 2) and dtype float. Report it on your answer.

(b) Construct another tensor y with the same size as x, but filled with ones. Report it on your answer. Print also its size, using the pytorch function size().

(c) Add the 2 tensors x and y, (1) saving the result on another tensor out and (2) saving the result on y (in-place addition). Report your result for both cases.

(d) Construct a randomly initialized numpy array x of size (3, 2) and dtype float. Convert it to a tensor using the pytorch function from_numpy(). Convert it back to a numpy array using the function .numpy().

1.2 Autograd (8 points)

Follow the tutorial and make sure you understand each step: `https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html`

(a) After you follow this tutorial, construct a randomly initialized tensor x of size (3, 2) and set requires_grad=True to track computation with it. Report it on your answer.

(b) Construct another tensor y by multiplying x by 10 and then adding 0.1. Then, take the maximum and save it to the final tensor out. Report all the intermediate results. Explain the grad_fn attribute of each intermediate result.

(c) Do backpropagation and compute the gradient $\frac{d(out)}{dx}$. Report it on your answer.

(d) Using the flag "with torch.no_grad()", run again the step (b). Explain what is the difference.

1.3 Neural Networks (10 points)

Follow the tutorial and make sure you understand each step: `https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html`

(a) After you follow this tutorial, define your own network. You are free to use whatever architecture you like, but it should be different from the one on the tutorial. In addition, your network should have at least one convolutional and one linear layer. Print and report your network on your answers. How many parameters does your network have?

(b) Construct a randomly initialized input of size 32x32. Run the forward function of your network and print the output. Report the output and its size on your answers.

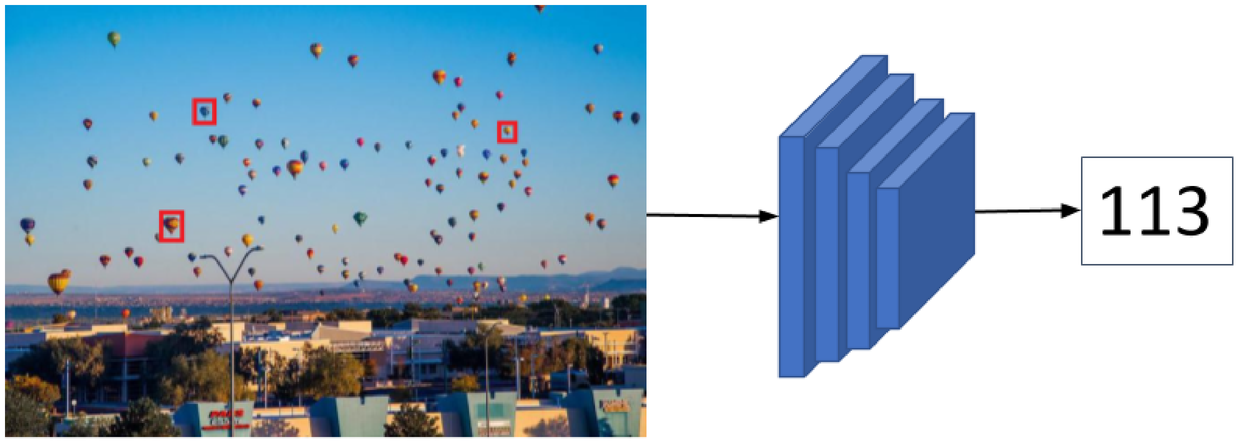

Figure 1: **Few-shot counting task.** Given an image from a novel class and a few exemplar objects from the same image delineated by bounding boxes, the objective is to count the total number of objects of the novel class in the image.

(c) Construct a random target output of the same size as your network's output. Use MSE for the loss and SGD for the optimizer and perform backpropagation. Select one intermediate layer of your network and report its bias gradients after backpropagation.

1.4 Training (6 points)

Follow the tutorial and make sure you understand each step: `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`

Similarly to the tutorial, train a network on the CIFAR10 dataset. Use the network architecture you defined for Question 1.3. Train it for 2-3 epochs and follow the same steps as in the tutorial. You do not need to train it on a GPU. Report the accuracy on the whole test set, as well as the accuracy per class.

1.5 Transfer Learning (8 points)

Follow the tutorial and make sure you understand each step: `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`

(a) Similarly to this tutorial, fine-tune the pre-trained ResNet-18 on a new dataset. Use the CIFAR10 dataset, similarly to Question 1.4. After loading the pre-trained model, change only the last linear layer, in order to output the correct number of classes (Be careful to set the number of classes of CIFAR10). Train it for only 2-3 epochs. You do not need to train it on a GPU. Report the accuracy on the whole test set, as well as the accuracy per class.

(b) Run the same experiment as in (a), but now use the pre-trained model as a feature extractor. Freeze all the network except the final layer. Again, report the accuracy on the whole test set, as well as the accuracy per class.

2 Few Shot Counting (60 points + 15 bonus)

For this question, we will use a CNN for counting objects in images. Given an image from a novel class and a few exemplar objects from the same image delineated by bounding boxes, the objective of few shot counting is to obtain the total number of objects of the novel class in the image, as illustrated in Figure 1. We are providing you with FamNet [1], which is a CNN trained on the training set of FSC147 dataset for few shot visual counting task. There are two ways to use FamNet: 1) we can use the pretrained FamNet to obtain the count for any test image. 2) We can adapt the FamNet to any test image using few bounding boxes from the test image. This adaptation is called test time adaptation. The paper discusses two loss functions for doing the test time adaptation. Read the paper to get a better understanding.

We are also providing you with Validation and Test sets from FSC147 dataset. The Val and Test sets consist of 1286 and 1190 images respectively. We have divided each of the Val and Test sets into two subsets: PartA and PartB. Val-PartA consists of 100 images from the Val set of FSC147 dataset, and Val-PartB consists of rest of the 1186 images. Similarly, Test-PartA and Test-PartB consists of 100 and 1090 images respectively from the FSC147 Test set.

Additionally and also different from the paper [1], we are providing you with *negative regions* for the Val and Test images, which are manually-specified regions without any object of interest. The negative regions are provided as a binary map where a value of 1 is assigned to pixels where there isn't any object of interest. At locations with value 0, there may or may not be any object of interest. Binary maps for negative areas can be found here: `https://bit.ly/3gshsvP`

2.1 Reading and comprehension (5 points)

Read the paper [1], and summarize it in one paragraph. Reading the paper will give you a clear understanding of the few shot counting task and how to adapt FamNet to any test image. The paper can be found here: `https://www3.cs.stonybrook.edu/~minhhoai/papers/fewshot_counting_CVPR21.pdf`

2.2 Result analysis (10 points)

Setup the Github repo `https://github.com/cvlab-stonybrook/LearningToCountEverything`. Follow the instructions to install it and download the Val and Test sets. Run the quick demo to see some example results. For this question, you might need a GPU. You can use Google Colab.

Run the evaluation code *test.py* without test time adaptation on the Val set, and report the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics. Also include a scatter plot for Val set where the X-axis contains the ground truth count, and the Y-axis contains the predicted count. Also, include 5 images and corresponding predicted density maps (output of FamNet) with the highest over-count error (predicted-count - gt-count) in the pdf report. Similarly, include 5 images and corresponding predicted density maps (output of FamNet) with the highest under-count error (gt-count - predicted-count) in the pdf report.

2.3 Adaptation (35 points)

Test time adaptation of FamNet using Negative Strokes: In the paper [1], we adapt FamNet for any test image using few exemplar bounding boxes from the test image. Your task is to use the negative stroke annotation for a test image and come up with a new loss function to adapt FamNet for any test image. You'll have to tune two hyper-parameters for the test time adaptation: number of gradient descent steps for the adaptation, and the scalar weight to be multiplied with the loss. Use the Val-PartA subset to find the best hyper parameters. Report the MAE and RMSE metrics on Val-PartA. Feel free to use Val-PartB for any finetuning, but it is not required. Also, present a scatter plot on Val-PartA which shows groundtruth count in the X-axis and predicted count on the Y-axis. Include plots for two cases: 1) with test time adaptation using your designed loss function 2) Without any test time adaptation (Hint for designing negative stroke loss: since a value of 1 in the negative stroke map signifies pixels where there isn't any object of interest, the network should predict 0 density values at such locations. Design a loss to enforce this behavior.)

2.4 Kaggle submission (10 points)

Kaggle Submission for Test-PartA subset: Once you've found the best hyperparameters for doing the test time adaptation for the previous question, use the same set of hyperparameters and adapt FamNet for each image in Test-PartA subset. Submit your results in a csv file to Kaggle: `https://www.kaggle.com/c/few-shot-count-cse512-spring21`.

Your csv file should have the format of the given *sample.csv*, where the header is ['Id', 'Count'], the first column corresponds to the image names and the second column to the predicted counts. **Important:**

Your csv file should contain all the images of the Test set. For the images of the Test-PartB, set the predicted counts to 0.

Your results will be ranked with respect to MAE and the numbers of points that you get for this question might be proportionate to how well your method performs. The deadline for the Kaggle submission is the same as the due date.

2.5 Bonus question (maximum 15 points)

Kaggle Submission for Test-PartB: Submit your results to Kaggle on Test-PartB subset. Your csv file should contain your predictions for the Test-PartB, in addition to your predictions for the Test-PartA from Question 2.4. The top three people in the leader board will receive 15, 10, and 5 bonus points.

3 What to submit?

3.1 Blackboard submission

You will need to submit both your code and your answers to questions on Blackboard. Put the answer file and your code in a folder named: SBUID_FirstName_LastName (e.g., *10947XXXX_lionel_messi*). Zip this folder and submit the zip file on Blackboard. Your submission must be a zip file, i.e, SBUID_FirstName_LastName.zip. The answer file should be named: answers.pdf. The first page of the answers-hw6.pdf should be the filled cover page at the end of this homework. Overall, you should submit:

1. Answers to Questions 1 and 2 on the answers-hw6.pdf file.
2. For Question 1, include all the code you used. It can be in separate files, but they should be named hw6_q1_*.py or hw6_q1_*.ipynb.
3. For Question 2, include all the code you used. It can be in separate files, but they should be named hw6_q2_*.py or hw6_q2_*.ipynb.
4. For Question 2, also submit the final prediction csv file that you submitted to Kaggle.

3.2 Report

You can use Latex or MS Word for preparing the report; submit report in pdf format. Report should be typed. Don't submit scans/pics of handwritten answers.

4 Cheating warnings

Don't cheat. You must do the homework yourself, otherwise you won't learn. You cannot ask and discuss with students from previous years. You cannot look up the solution online.

References Cited

[1] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Cover page for answers.pdf
CSE512 Spring 2021 - Machine Learning - Homework 6

Your Name:

Solar ID:

NetID email address:

Names of people whom you discussed the homework with: