Assignment 4

Due on December 27, 2022 (23:59:59)

Instructions. The goal of this problem set is to make you understand and familiarize with Neural Network and Convolutional Neural Network (CNN) concepts.

Vegetable Classification of Images using Neural Network and CNN

For this assignment, you will implement a Neural Network and CNN to classify the examples on the Vegetable Image Dataset mentioned below.

1 Dataset

- You will use Vegetable Image Dataset [1] contains about 21.000 images (train set: 15.000 images, validation set: 3000 images, test set: 3000 images) of different vegetable species and corresponding 15 labels. (See Figure 1) The species type for each class is stated as below:
  - Class 1: Bean
  - Class 2: BitterGourd
  - Class 3: BottleGourd
  - Class 4: Brinjal
  - Class 5: Broccoli
  - Class 6: Cabbage
  - Class 7: Capsicum
  - Class 8: Carrot
  - Class 9: Cauliflower
  - Class 10: Cucumber
  - Class 11: Papaya
  - Class 12: Potato
  - Class 13: Pumpkin
  - Class 14: Radish
  - Class 15: Tomato

2 Part 1: Multi Layer Neural Network

In this part of the assignment, you have to implement multi layer neural network for classification. In other words your network consists of one input layer, n hidden layer(s) and one output layer. You will implement forward and backward propagations with the loss function and learning setting. Actually, you will implement a back-propagation algorithm to train a neural network.

In this step, you will implement the network given in Figure 1 and train the network feeding by given training set as gray-level image values. It is important to normalize image values $(0-255)$ to between 0 and 1. We can express this network mathematically as:

$$o_i = w_{ij}x_j + b_i \tag{1}$$

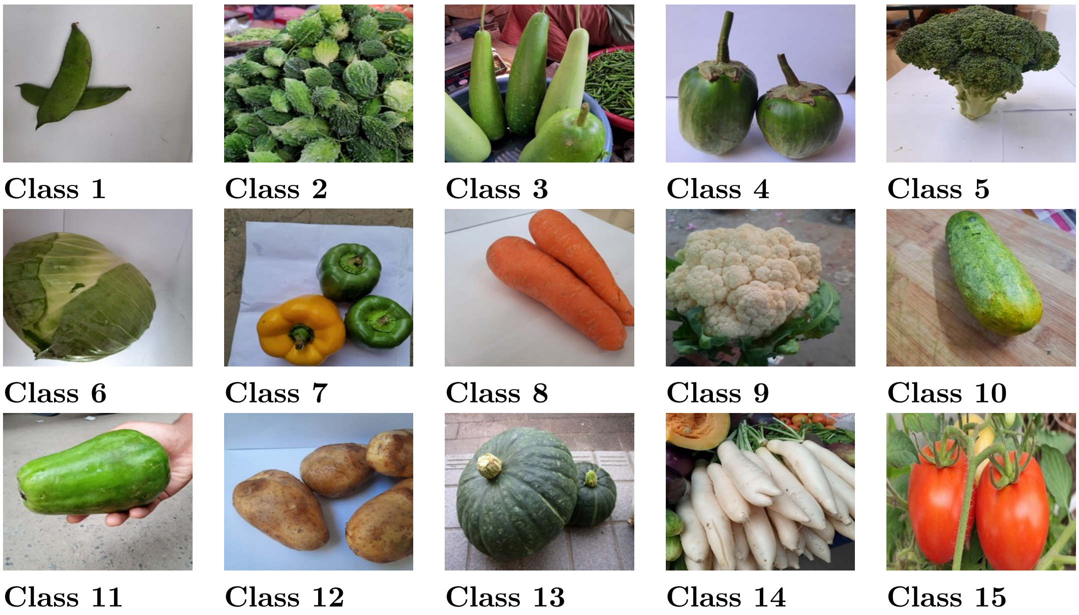
Class 1 Class 2 Class 3 Class 4 Class 5
Class 6 Class 7 Class 8 Class 9 Class 10
Class 11 Class 12 Class 13 Class 14 Class 15

Figure 1: Different vegetable species sample images from different classes.

As loss function, you will use sum of negative log-likelihood of the correct labels. Write a python function to compute loss function. Then you will update network parameters w and b to minimize the loss function using gradient descent algorithm. You will implement a function which computes the derivative of the loss function with respect to the parameters. To make sure your function is correct, you must also implement numerical approximation of gradients.

Write a function to minimize your cost function using mini-batch gradient descent. You should try different learning rate$(0.005 - 0.02)$ and batch sizes$(16 - 128)$. Make a table to show learning performance for each setting you tried.

Finally, you will visualize the learned parameters as if they were images. Visualize of the each set of parameters that connect to $O_0, O_1, \ldots, O_n$.

Training a network

- You should determine the number of units in your hidden layers.
- You should determine batch size.
- You should determine a learning rate for your gradient descent method.
- Remember, learning rate parameter may be a problem (too big - may not converge, too small - very slow convergence). For this reason you can define a learning rate decay parameter. You will start with a learning rate value and after each epoch you will reduce the learning rate by multiplying it by a decay rate. This operation can deal with mentioned problem.

- You can use different activations functions: Sigmoid, tanh, ReLU etc.
- You can control your implementation by plotting loss. You can see if it converges or if it needs a different parameter setting.
- **You should discuss about your each experiment in the report. Comment about their effects.**
- Save your trained models to use later in test time.

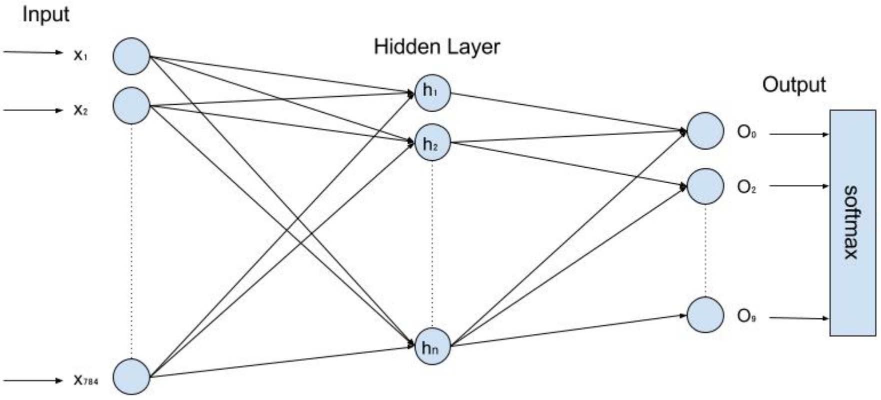

Figure 1: Multi layer neural network.

3 Part 2: Convolutional Neural Network

In this part of the assignment, you will use pretrained VGG-19 convolutional neural network (CNN) and finetune this network to classify the sample images (Figure 2).

You will use two different cases when finetuning the CNN network:

1. You will finetune the weights of all layers in the VGG-19 network.
2. You will finetune the weights of only two last fully connected (FC1 and FC2) layers in the VGG-19 network.

Implementation Details:

1. You must implement your Neural Network and visualization process from Part 1 section, by your own. You cannot use readymade libraries for Part 1. You may also find useful links for implementing negative log-likelilood, visualization process and calculating optimal neuron size for hidden layer :
   - `https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/`
   - `https://ml4a.github.io/ml4a/looking_inside_neural_nets/`

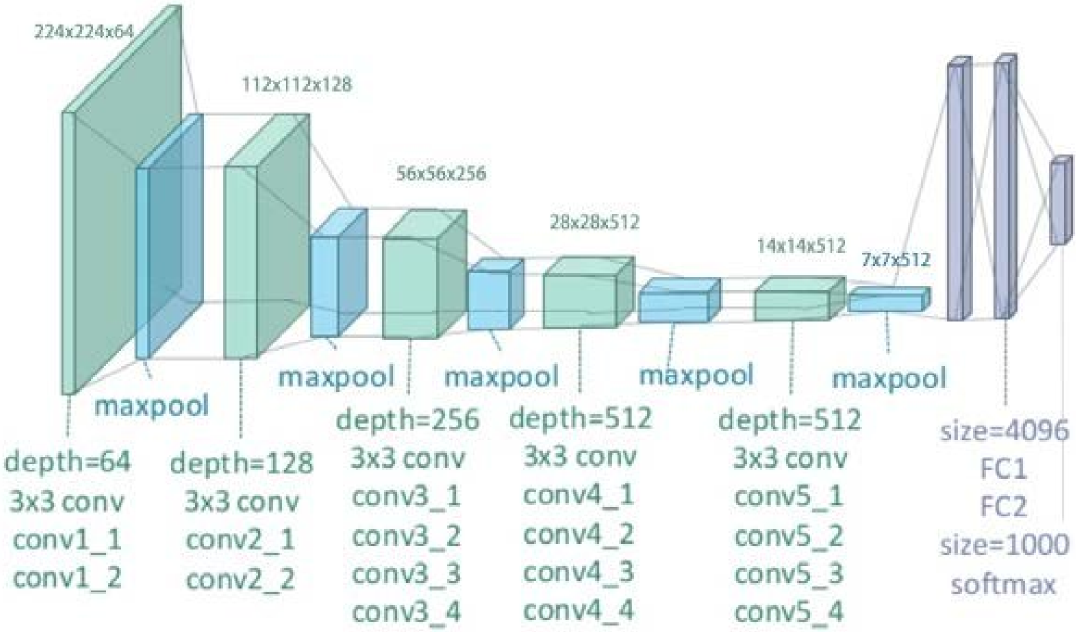

Figure 2: VGG-19 Network Architecture.

- https://stats.stackexchange.com/a/136542

2. For using/finetuning pretrained VGG-19 model and visualization process, you can utilize libraries from the PyTorch (https://pytorch.org/)

3. You can also look into the links below for further analyze for Part 2:

   - http://cs231n.github.io/convolutional-networks/
   - https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
   - https://pytorch.org/docs/stable/nn.html
   - https://pytorch.org/vision/stable/models.html
   - https://discuss.pytorch.org/t/how-to-visualize-the-actual-convolution-filters-in-cnn/13850
   - https://towardsdatascience.com/visualizing-convolution-neural-networks-using-pytorch-3dfa8443e74e
   - https://www.linkedin.com/pulse/custom-function-visualizing-kernel-weights-activations-arun-das

Notes:

- For Part 1, you will implement single layer neural network and run experiments on Vegetable Image Dataset. You will change parameters (activation func., objective func. etc.) and report results with a table format in your reports. **Obligatory**
- For Part 1, you will implement a neural network which contains one hidden layer. You will change the mentioned parameters (unit number in the hidden layer, activations function etc.) and report the results. **Obligatory**
- For Part 1, you will change your architecture and use a network that contains two hidden layers. Repeat same experiments and comment about the results. **Obligatory**
- For Part 1, you have to comment about results and parameters' effects on different values(learning rate, batch size, layer size, hidden neuron size in the layers etc...).
- For Part 1, Your implementation should be reproducible, in other words, do not write separate code for each architecture. If you use n layers, your method should create a n-layer network and learn the classifier.
- For Part 2 You will use and finetune pratrained VGG-19 network for classification with two different cases mentioned in Part 2 section. **Obligatory**
- For both two hidden layers neural network in Part 1 and VGG-19 network in Part 2, you have to extract "Confusion Matrix" and compare/analyze them in your report.
- Comment your code with corresponding mathematical functions and explain what is going on your code in the code scripts.
- **You should apply Early Stopping method to all for both your implemented neural network and CNN models by constantly measuring your error values for both train and validation sets in every 10 Epoch. Then you should measure your early-stopped model's performance on test set by using the performance metrics mentioned in Notes Section.**
- **You should also compare your neural network and VGG-19 with respect to performance metrics (Accuracy, Precision, Recall, F1-Measure), for your neural network model, parameter (weight size) size and training-test error curve plots and you must state every experiment you accomplish and related proper explanation/conclusion about why you obtain such a result in your analysis report in order to get full points on analysis report.**

Submit

You are required to submit all your code with a report in ipynb format (should be prepared using Jupyter notebook). The codes you will submit should be well commented. Your report should be self-contained and should contain a brief overview of the problem and the details of your implemented solution. You can include pseudocode or figures to highlight or clarify certain aspects of your solution. Finally, prepare a ZIP file named **name-surname-pset4.zip** containing

- report_and_code.ipynb (PDF file containing your report and also your learned parameters to use in test time (you can use numpy.save and numpy.load)
- code/ (directory containing all your codes as Python file .py)

Grading

- Code (60 points) : 30 points for multi-layer neural network implementations, 30 points for VGG-19 implementations.
- Report(40 points): Analysis of the results for classification: 40 points.

  Notes for the report: You should analyse the method you employed. How did you improve your results? Explain every step you choose. Comment about the results. Compare your single layer, multi-layer and CNN models. Comment about the activation functions, loss functions etc. that you used for your experiments. Your reports have to include your classification accuracy.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else. [1]

References

[1] `https://www.kaggle.com/datasets/misrakahmed/vegetable-image-dataset`

[1] This assignment is adapted from http://www.cs.toronto.edu/ guerzhoy/321/proj2/