Assignment 3

Due on January 5, 2024 (23:59:59)

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

Flower 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 Flower Species Dataset mentioned below. You will use the given train, test, and validation split.

The Flower Species Dataset

The Flower Species Dataset is an image dataset that consists of 15 different flower species (Figure 1).

- You can download the dataset from given link
- Dataset includes train and test images. Each species dataset includes 700 images for training, 150 images for validation, and 150 images for testing.
- The species type for each class is stated as below:
  - Class 1: Aster
  - Class 2: Calendula
  - Class 3: California Poppy
  - Class 4: Coreopsis
  - Class 5: Dandelion
  - Class 6: Daisy
  - Class 7: Iris
  - Class 8: Lavender
  - Class 9: Lily
  - Class 10: Marigold
  - Class 11: Orchid
  - Class 12: Poppy
  - Class 13: Rose
  - Class 14: Sunflower
  - Class 15: Tuli

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.

(a) Aster (b) Calendula (c) Cal. Poppy (d) Coreopsis (e) Dandelion

(f) Daisy (g) Iris (h) Lavender (i) Lily (j) Marigold

(k) Orchid (l) Poppy (m) Rose (n) Sunflower (o) Tulip

Figure 1: Examples for flower species.

In this step, you will implement the network given in Figure 2 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:

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

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 $O_0, O_1, \ldots, O_n$.

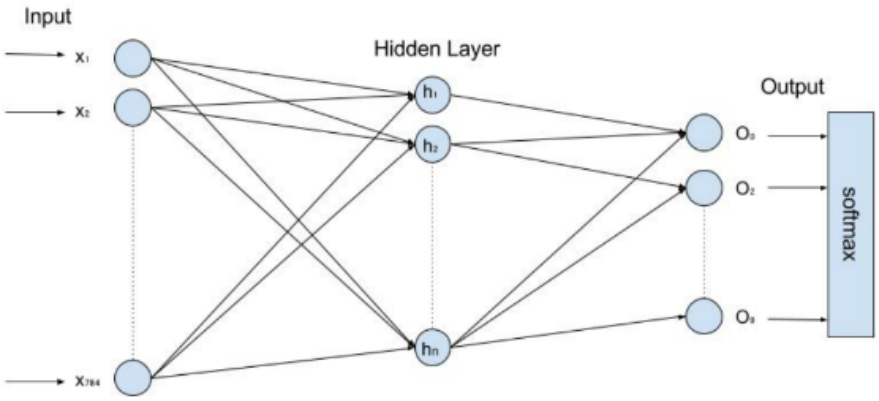

Figure 2: Multi layer neural network.

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.

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.

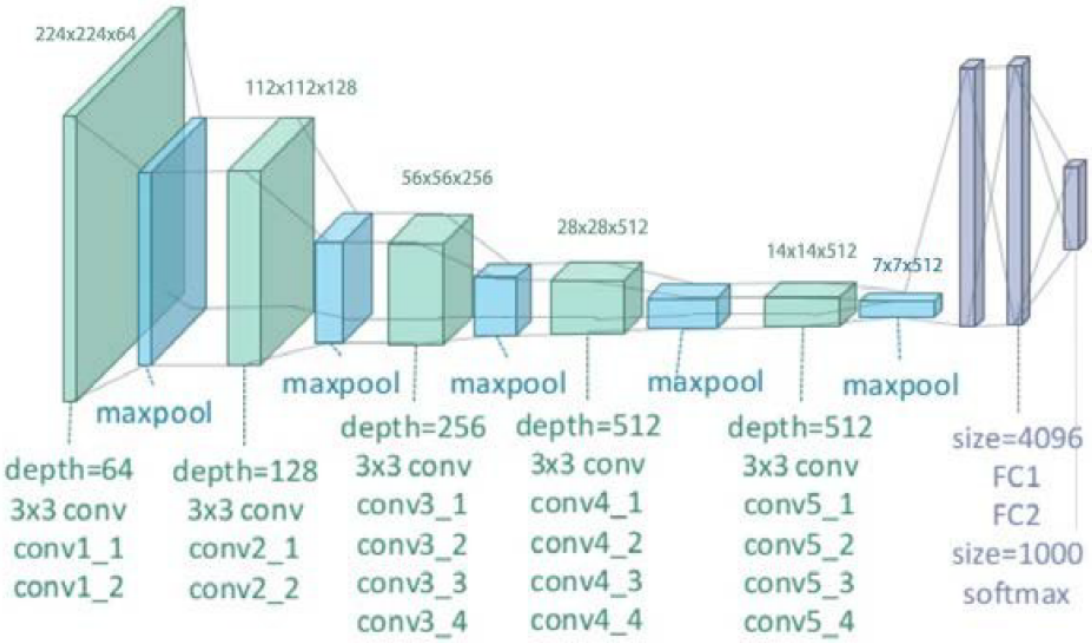

Figure 3: VGG-19 Network Architecture.

Implementation Details

1. You must implement your Neural Network and visualization process from Part 1 section, on your own. You cannot use readymade libraries for Part 1. You may also find useful links for implementing negative log-likelihood, 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/`
   - `https://stats.stackexchange.com/a/136542`
2. For using fine-tuning pre-trained 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 analysis 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 a single-layer neural network and run experiments on the Flower Species 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 that 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 the same experiments and comment on 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 the pre-trained VGG-19 network for classification with two different cases mentioned in the Part 2 section. **Obligatory**
- For both two hidden layers of the neural network in Part 1 and the VGG-19 network in Part 2, you have to extract the "Confusion Matrix" and compare/analyze them in your report.
- Comment your code with corresponding mathematical functions and explain what is going on in your code in the code scripts.

- **You should apply the 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 the test set by using the performance metrics mentioned in the 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 to get full points on the analysis report.**

Submit

You are required to submit all your code in a Jupyter Notebook, along with a report in ipynb format, which should also be prepared using Jupyter Notebook. The code you submit should be thoroughly commented on. Your report should be self-contained and include a concise overview of the problem and the details of your implemented solution. Feel free to include pseudocode or figures to highlight or clarify specific aspects of your solution. Finally, prepare a ZIP file named name-surname-a3.zip containing:

- assignment_3.ipynb (including your report and code)
- assignment_3.py (py file version of your ipynb file)
- Do not send the dataset.

Grading

- Code part(60 points): MLP: 30 points, VGG-19: 30 points
- Theory part(40 points): Analysis of the results for classification

Note: Preparing a good report is important as well as the correctness of your solutions! You should explain your choices and their effects on the results. You can create a table to report your results.

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.