Indian Institute of Technology Hyderabad
Deep Learning (AI2100/AI5100): Assignment-2

Topic: Convolutional Neural Networks and Recurrent neural Networks
Assigned on: 13th **February, 2024**
Deadline: 27th **February, 2024**
Maximum Marks: 40

1 Instructions

- Answer all questions. We encourage best coding practices by not penalizing (i.e., you may not get full marks if you make it difficult for us to understand your submission. Hence, use intuitive names for the variables, functions, etc., and comment your code liberally. You may use the text cells in the notebook to briefly explain the objective of a code cell.)
- It is **expected** that you work on these problems individually. If you have any doubts please contact the TA or the instructor no later than 4 days before the deadline.
- You may use built-in implementations only for the basic functions such as `sqrt, log`, etc. from libraries such as `numpy` or `PyTorch`. Other high-level functionalities are expected to be implemented by the students. (Individual problem statements will make this clear.)
- For plots, you may use `matplotlib` and generate clear plots that are complete and easy to understand.
- You are expected to submit the Python Notebooks saved as <your-roll-number>.ipynb
- If you are asked to report your observations, use the mark down text cells in the notebook.
- Late submission policy: < 1 day delay $\rightarrow$ 10% penalty, $\{> 1$, but $< 2\}$ days delay $\rightarrow$ 25% penalty, $\{> 2$, but $< 3\}$ days delay $\rightarrow$ 50% penalty, $\{> 3$, but $< 4\}$ days delay $\rightarrow$ 75% penalty, and for a submission beyond 4 days, there won't be any reward.

2 Problems

1. **Convolution function**: It accepts an image input, a filter kernel, stride, padding, and the non-linear function. The function must convolve the input image (after padding if specified) with the kernel (at the specified stride size) and generate an output activation after applying the specified non-linearity. Verify with the standard options for the non-linear activation functions - sigmoid, tanh, ReLU, Leaky ReLU. Display the input image (e.g. a small image of the IITH logo), the filter kernel, and the output activation map. Ensure that your function can accept multi-channel input and a corresponding kernel volume. (5)
2. **Pooling function**: It accepts as input the activation map output from the convolution function, a pooling function, and stride. The function must output the appropriately pooled activation map. Display the input activation map and the pooled output. (2)
3. **Convolution layer function**: It accepts as input a volume (image or activation maps), number of filters, kernel dimensions, stride, padding, and the non-linear activation function. The function must convolve the input volume (after padding if specified) with each kernel (at the specified stride size) and generate an output activation volume after applying the specified non-linearity. Display the input image or activation maps, the filter kernels, and the output activation maps. Verify that the output of this function does indeed have the expected size (W $\times$ H $\times$ C) as discussed in class. (3)
4. **Pooling layer function**: It accepts as input the activation map volume, the pooling function, stride, and generates a pooled output volume. A special case for performing Global Average Pooling should also be provided. (2)

5. **Multilayer Perceptron (MLP) function**: It accepts as input a vector, the number of hidden layers, the size of each hidden layer, the non-linear function, and the size of the output layer. This function should generate an output vector of the specified size. Generate the output with and without the softmax function applied to the output layer. (3)

6. **Putting-it all together**: Finally, use the functions you have written to implement a CNN with the following architecture. The CNN must accept an image input and output a vector of appropriate dimension. In other words, the function must effectively implement the feed-forward path in a CNN. (5)

   - Input image of size $224 \times 224 \times 3$. Use validation images (or, random crops) from the ILSVRC dataset.
   - Convolution layer with 16 kernels of size 5×5 spatial dimensions and ReLU activation.
   - Max pooling layer of size 2×2 with a stride of 2 along each dimension.
   - Convolution layer with 32 kernels of spatial size 3×3 and sigmoid activation.
   - Max pooling layer of size 2×2 with a stride of 2 along each dimension.
   - A Global Average Pooling (GAP) layer.
   - An MLP with one hidden layer (size same as input) that accepts as input the previous layer's output and maps it to 1000 output nodes. Use ReLU activation for the MLP (softmax in the o/p layer).

   Verify that your composition of function accepts and image input and outputs a vector.

7. **The adding problem**: In this task, each data sample consists of a sequence of variable length, but a constant depth (size of feature vector at each time instance) of 2. All values of the first dimension (randomly) lie in $[0, 1]$, and the second dimension is all zeros except for two elements that are marked by 1. The objective of the task is to sum the random values whose second dimensions are marked by 1. Train the different RNNs (Elmon network, LSTM, and GRU) discussed in the class and compare their performance against a baseline that always predicts a sum of 1 plotting the learning curves and final performance. Note that you are expected to implement these models (as opposed to using the built-in constructs).[3 (Elmon) + 7 (LSTM) + 4 (GRU)+ 6 (Dataset & baseline & comparison) = 20]

   The following table presents two data samples (x) along with their labels (y). Note that the samples should be of different lengths (n), so the dimensions of each sample can be represented as $n \times 2$. Given examples have lengths of 5 and 8 respectively. You have to generate/create a big dataset (≥ 5000) of such samples for training and testing of the RNNs.

	x								y
1	0.1	0.9	0.25	0.17	0.76				1.07
	0	1	0	1	0				
2	0.86	0.31	0.43	0.12	0.01	0.29	0.95	0.09	0.52
	0	0	1	0	0	0	0	1	