Assignment 2 –
Case Study for Implementation

Release date is 22 Feb 2023

Deadline is 26th March 2023 (**Hard deadline – no extensions**)

You are working as a research engineer or a data scientist in a company. Your team lead asked you to implement a new neural network. The network is like a Convolutional Neural Network (CNN). The difference is that instead of a Convolution layer in a CNN, you have to introduce a new layer to bring novelty in their model.

Below you can find an image from a PowerPoint (PowerPoint is attached) in which the difference between the traditional 2DCNN kernel operation is shown vs the proposed approach that you have to implement.

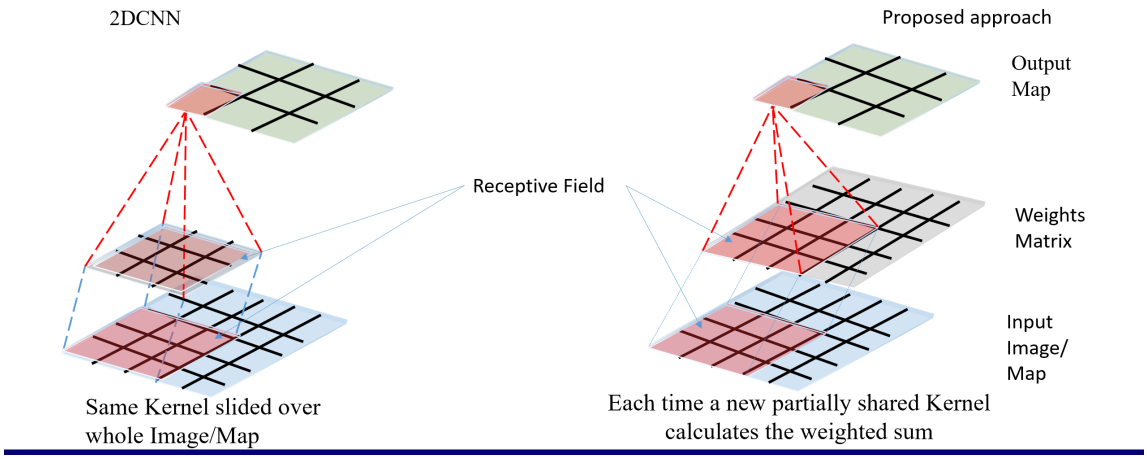

Figure of a 2D CNN convolution operation in forward propagation vs Proposed approach

Forward propagation: The main difference is that in a CNN convolve layer, you have a small (e.g. 3x3, 5x5, or 7x7) weight kernel that slides over the whole input image to generate output feature map. This weight kernel is randomly initialized. You can use more than one weight kernel to generate multiple output feature maps.

Whereas, in the proposed approach you have a bigger weight matrix (equal size to the input image/feature map) that is randomly initialized. From this matrix, each time you have to select a receptive field as a weight kernel that will do dot product with the same position receptive field in the input image/feature map (as shown in Figure/ppt) to calculate a

weighted output value for an output neuron. The same process is repeated until all the neurons/pixels/values in the output map/s are calculated.

Backward Pass (to calculate/update weights): Once you reach the output layer, you have to use the backpropagation algorithm to backpropagate the error that will be using the same receptive field i.e., each output neuron will use the same weights in the respected receptive field to calculate gradients (or local error) that it used in forward propagation (this time in backward direction). Backpropagation is the step that apply the chain rule to compute the gradient of the loss function with respect to the inputs.

Once you figure how to do that, then create the following network as a baseline:

Input layer – ProposedLayer1– ActivationLayer1 – PoolingLayer1 – ProposedLayer2 – ActivationLayer2 – PoolingLayer2 – ProposedLayer3 – ActivationLayer3 – FullyConnectedLayer1/DenseLayer1 – OutputLayer/Softmax

- At ProposedLayer1 use 16 weight matrices to get 16 feature maps while using a 3x3 receptive field and stride of 1.
- At ProposedLayer2 use 12 weight matrices to get 12 feature maps while using a 3x3 receptive field and stride of 1.
- At ProposedLayer3 use 8 weight matrices to get 8 feature maps while using a 3x3 receptive field and stride of 1.
- At ActivationLayer1, ActivationLayer2, and ActivationLayer3 use ReLU activation or other recent functions.
- At PoolingLayer1 and PoolingLayer2, use 2x2 receptive field.

Once you implement the above model, distribute the data in training, validation, and testing set, and train and test it to bring optimal accuracy, precision, and recall.

Report the following:

1. At first page, write your names, roll numbers, and group number, and then in 50-150 words, report who did what in the assignment e.g., who came up with the idea of how to implement it, who implemented it (as a whole or part of it), who debugged it, who gave suggestions (and what suggestion/s), who wrote the report, who did which simulations
2. Explain how your group implemented the layer within existing library (keras, TensorFlow, PyTorch, or other) or from scratch **[3 Marks]**
3. Report the successful implementation of the layer and complete model with a figure in tensorboard. **[10 Marks]**
4. Load and distribute the data for training, validation, and testing **[2 Marks]**

5. Train the model with the dataset provided in this assignment folder and report the training and testing performance against various hyper parameters (e.g. learning rate = 0.1, batch size = 32, epochs = 500,) using stochastic gradient decent [**9 Marks**].
6. Report the analysis of the model against changes in network depth and width as well as changing hyper parameters i.e., show how performance increase or decrease by changing learning rate {0.01, 0.001, or other}, batch sizes {8, 16, 50, or others depending on your system/laptop capacity}, epochs {100, 200, 700, 1000 or other depending on the model performance}, optimizers e.g. Adam or other factors. Report all results in a table, even if they are not good. Also show their training and testing performance in the form of a graph [**7 Marks**]
7. Report what result you achieved after training a Convolutional Neural Network of same depth [**5 Marks**].
8. How to change the code to work over 1-Dimensional data? [**2 Marks**]
9. How to change the code to work over 3-Dimensional data? [**2 Marks**]

NOTE:

Make a group of maximum 3 members and perform the following actions:

- You can use existing libraries and update them or implement the network from scratch like you implemented a neural network or a multilayer perceptron. All depends on you.
- Once you implement it, use the dataset provided and show the results.
- Upload two files. One for the code and one for the report with code, results, and your analysis. Share the code (Jupiter notebook) and the report in the Turnitin assignment.