

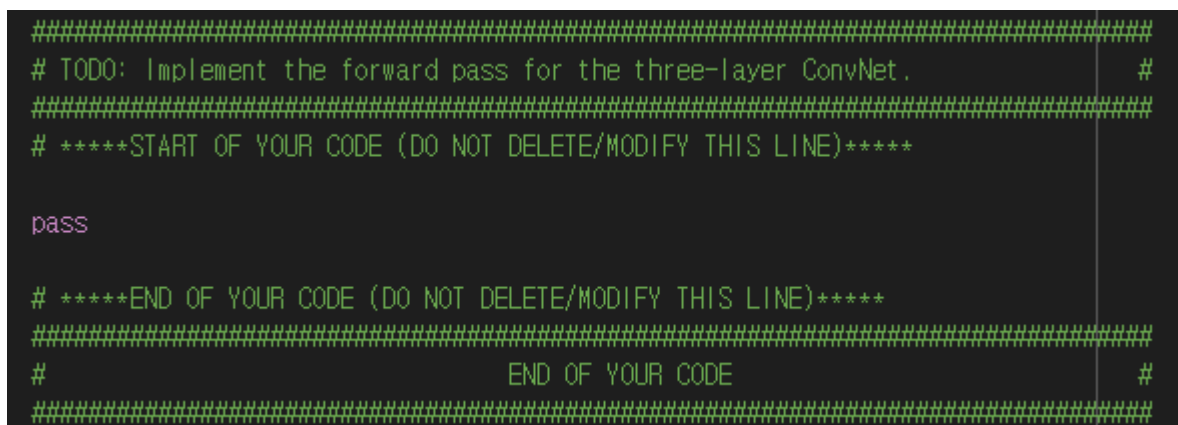
## Neural Networks

This homework assignment provides a foundational introduction to deep learning implementation. It is recommended to use [Google Colaboratory \(Colab\)](#) for completing the assignment.

Please submit all parts of this assignment on PLATO, including both the code and the report in PDF format. To receive full marks, your functions (i.e., \*.ipynb files) must not only work correctly but also be well-documented with clear comments to facilitate understanding and usage by others. Insufficient or unclear comments will result in a deduction of marks. In this assignment, you are also required to submit scripts that demonstrate the testing of your functions on all specified cases, along with any requested images and additional answers. The scripts and results (screenshots) should be compiled into a single PDF file and clearly labeled. Failure to submit either the code or the PDF will result in a loss of points.

This assignment consists of five parts: Part 1 to Part 5, and within this range, you need to complete seven problems.

You need to write the appropriate code inside the "YOUR CODE" sections as shown in the images below, execute them, save the execution results, and submit.



```
#####  
# TODO: Implement the forward pass for the three-layer ConvNet. #  
#####  
# *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****  
  
pass  
  
# *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****  
#####  
#                               END OF YOUR CODE                               #  
#####
```

Once you have completed writing the answers for all 7 problems, **please execute the entire code, save the execution results, and submit them along with a PDF report.**

### Question 1 (5 points)

TODO : Implement the network like the architecture below.

you should use `nn.Sequential` to define and train a three-layer ConvNet

1. Convolutional layer (with bias) with 32 5x5 filters, with zero-padding of 2
2. ReLU
3. Convolutional layer (with bias) with 16 3x3 filters, with zero-padding of 1
4. ReLU
5. Fully-connected layer (with bias) to compute scores for 10 classes

\*Hint

Conv2d : <https://pytorch.org/docs/stable/nn.functional.html>

relu : <https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html#torch.nn.ReLU>

\* If you want to implement ReLU using PyTorch's `nn.Module`-based class, you can use **`nn.ReLU`**. However, if you need to use the ReLU function provided by the Functional API, you should use **`F.relu`**.

### Question 2 (15 points)

TODO : Initialize your weight matrices using the ``random_weight`` function defined above, and you should initialize your bias vectors using the ``zero_weight`` function above of a **three-layer ConvNet**.

\*Hint

```
def random_weight(shape), def zero_weight(shape)
```

### Question 3 (25 points)

TODO: Set up the layers you need for a **three-layer ConvNet** with the architecture defined above **Question1** by using '**`nn.Module API`**'.

\*Hint

`nn.Linear` : <https://pytorch.org/docs/stable/nn.html>

#### Question 4 (15 points)

TODO: Implement the forward function for a **3-layer ConvNet**. You should use the layers you defined in `__init__` and specify the connectivity of those layers in `forward()`

\*Hint

Relu : [ReLU — PyTorch 2.0 documentation](https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html#torch.nn.ReLU)

nn.flatten : <https://pytorch.org/docs/stable/generated/torch.nn.Flatten.html#torch.nn.Flatten>

#### Question 5 (15 points)

TODO: Instantiate your **three layer ConvNet** model and a corresponding optimizer. You should train the model using stochastic gradient descent without momentum.

\*Hint

optim.SGD : <https://pytorch.org/docs/stable/generated/torch.optim.SGD.html#torch.optim.SGD>

#### Question 6 (25 points)

TODO: To define and train a **three-layer ConvNet** with the same architecture we used in Part III with the **Sequential API**. You can use the default PyTorch weight initialization. You should optimize your model using stochastic gradient descent with momentum 0.9.

\*Hint

Sequential API : <https://pytorch.org/docs/stable/generated/torch.nn.Sequential.html>

## Question 7 (extra points)

This problem is not included in the overall score of the homework. However, additional points are given when you perform the problem.

TODO: Experiment with any architectures, optimizers, and hyperparameters. Achieve AT LEAST 70% accuracy on the \*validation set\* within 10 epochs. Note that you can use the `check_accuracy` function to evaluate on either the test set or the validation set, by passing either `loader_test` or `loader_val` as the second argument to `check_accuracy`. You should not touch the test set until you have finished your architecture and hyperparameter tuning, and only run the test set once at the end to report a final value.

Now it's your job to experiment with architectures, hyperparameters, loss functions, and optimizers to train a model that achieves at least 70% accuracy on the CIFAR-10 validation set within 10 epochs. You can use the `check_accuracy` and `train` functions from above. You can use either `nn.Module` or `nn.Sequential` API.

Describe what you did at the end of this notebook.

>> Things you might try

- Filter size: Above we used 5x5; would smaller filters be more efficient?
- Number of filters: Above we used 32 filters. Do more or fewer do better?
- Pooling vs Strided Convolution: Do you use max pooling or just stride convolutions?
- Batch normalization: Try adding spatial batch normalization after convolution layers and vanilla batch normalization after affine layers. Do your networks train faster?
- Global Average Pooling: Instead of flattening and then having multiple affine layers, perform convolutions until your image gets small (7x7 or so) and then perform an average pooling operation to get to a 1x1 image picture (1, 1, Filter#), which is then reshaped into a (Filter#) vector. This is used in Google's Inception Network (See Table 1 for their architecture).
- Regularization: Add L2 weight regularization, or perhaps use Dropout

**Deliverables**

You will hand in your assignment in PLATO. You should hand in one zip file including two files, a file containing your code (i.e., \*.ipynb file). This file must have sufficient comments for others to easily use and understand the code. In addition, hand in a PDF document showing scripts (i.e., records of your interactions with the Python shell) showing the specified tests of your functions as well as the images and other answers requested. The PDF file has to be organized and easily readable / accessible. Assignments are to be handed in before 11:59pm on their due date