**HW-6**

**Question 1**: Implement CNN on CIFAR-10 dataset **(5 Points**)

* Use the following network architecture
* Convolutional layer (with bias) with 32 5x5 filters, with zero-padding of 2
* ReLU
* Convolutional layer (with bias) with 16 3x3 filters, with zero-padding of 1
* ReLU
* Fully-connected layer (with bias) to compute scores for 10 classes

Note that we have **no softmax activation** here after our fully connected layer: this is because PyTorch's cross-entropy loss performs a softmax activation.

* You should optimize your model using stochastic gradient descent with a Nesterov momentum of 0.9.
* Use learning rate of 1e-2
* You should initialize the weight matrices of the model using the Kaiming normal initialization method.

**Question 2 (15 Points)**: In this question, you can experiment with different convent architectures on CIFAR-10. You can experiment with different architectures, hyper-parameters, loss functions, and optimizers to train a model that achieves **close to 80%** accuracy on the CIFAR-10 **validation** set within 10 epochs.

**Students getting the best accuracy on the validation set will get two extra points for this HW.**

## Submit a solution with your final model (pdf file and ipynb notebook). However, you should also provide a detailed report of experiments that you did. When explaining your experiments, also provide a link of your runs on wandb.

Things you might try:

Use a small sample (as we did for MLP) to find the combination of network architecture and learning rate that can overfit a small sample.

* **Network architecture**: You can try variations of the following : [conv-relu-batchnorn-conv-relu-batchnorm-maxpool] x N -> [Fully connected] X M. N and M here are repetitions.
* **Different filter size**
* **Number of filters**
* **Pooling vs Strided Convolution**: Do you use max pooling or just stride convolutions?

Now gradually increase the size of the sample and fine-tune other hyperparameters. For hyperparameter tuning: start by testing a large range of hyperparameters for just a few epochs (1 to 5) to find the combinations of parameters that are working. Once you have found some sets of parameters that seem to work, search more finely around these parameters. You may need to train for more epochs.

* **Alternative optimizers:** You can try Adam, Adagrad, RMSprop, etc.
* **Batch Size**
* **Alternative activation functions** such as leaky ReLU, parametric ReLU, ELU, or MaxOut
* **Regularization**: Add l2 weight regularization or use Dropout.
* **Learning Rate Scheduler - Try at least two different learning rate schedulers.**
* **Data Augmentation**