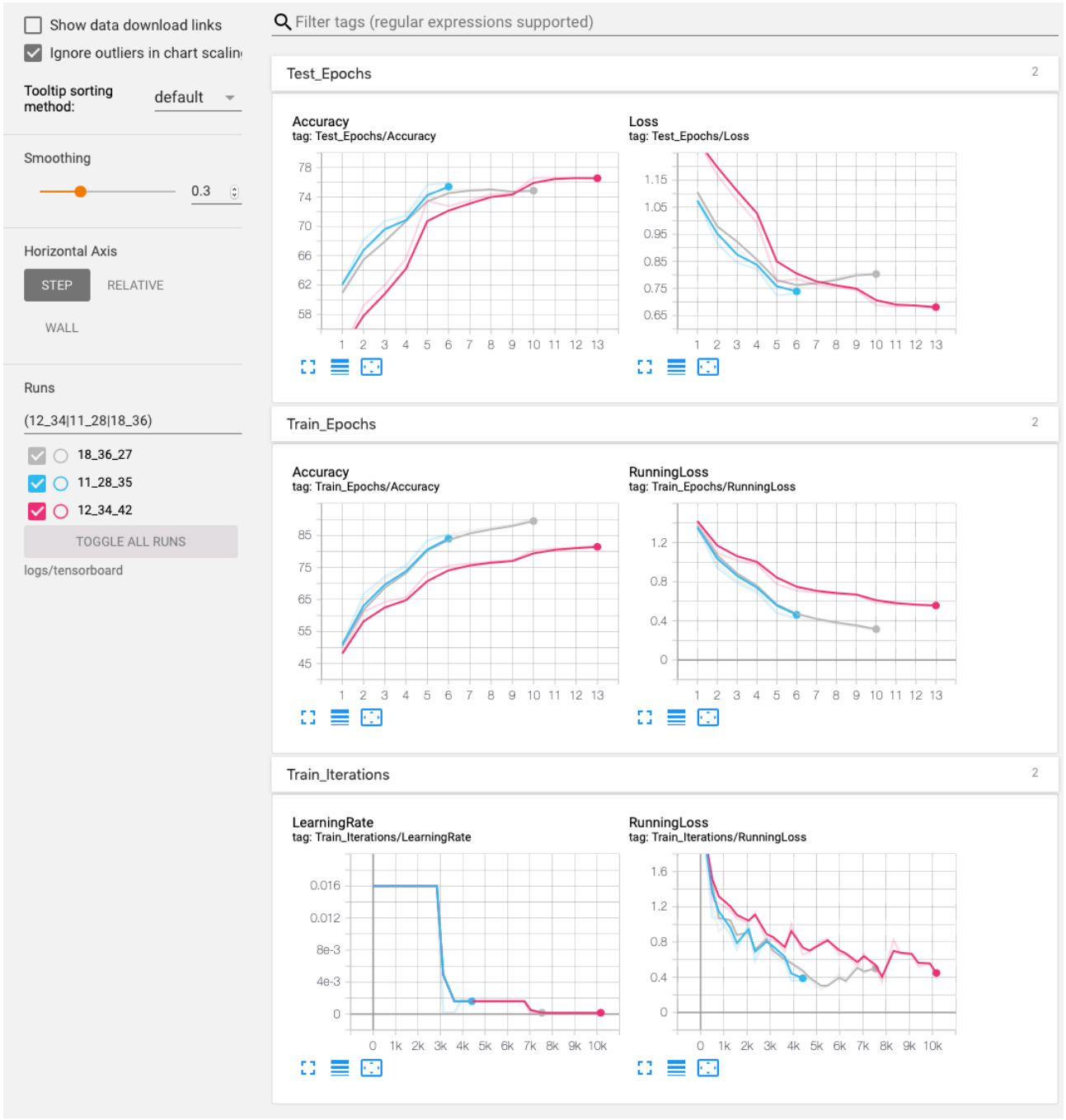
Experiments with CIFAR-10

- For CIFAR-10 classification problem I trained a CNN with GeForce GTX 1080.
- I slightly modified `cifar10.ipynb` to get insights.
- I put all essentials in `net.py`, `utils.py`, `train.py`.
- **The experimental pipeline is in** Experiments.ipynb
- The top-3-performance models are following:

timestamp	model	criterion	batch_size	optimizer	scheduler	n_epochs	device	test_accuracy
12_34_42	NetCustom((pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1)) (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1)) (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (bn3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (fc1): Linear(in_features=1152, out_features=128, bias=True) (fc2): Linear(in_features=128, out_features=64, bias=True) (fc3): Linear(in_features=64, out_features=10, bias=True))	CrossEntropyLoss()	64	SGD(lr: 0.016, momentum: 0.9, dampening: 0, weight_decay: 0.01, nesterov: True)	StepLR(gamma: 0.1, step_size: 5)	13	cuda	76.88%
11_28_35	NetCustom((pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1)) (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1)) (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (bn3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (fc1): Linear(in_features=1152, out_features=128, bias=True) (fc2): Linear(in_features=128, out_features=64, bias=True) (fc3): Linear(in_features=64, out_features=10, bias=True))	CrossEntropyLoss()	64	SGD(lr: 0.016, momentum: 0.9, dampening: 0, weight_decay: 0, nesterov: True)	StepLR(gamma: 0.1, step_size: 5)	6	cuda	75.89%
18_36_27	NetCustom((pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1)) (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1)) (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (bn3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (fc1): Linear(in_features=1152, out_features=128, bias=True) (fc2): Linear(in_features=128, out_features=64, bias=True) (fc3): Linear(in_features=64, out_features=10, bias=True))	CrossEntropyLoss()	64	SGD(lr: 0.016, momentum: 0.9, dampening: 0, weight_decay: 0, nesterov: False)	StepLR(gamma: 0.1, step_size: 5)	10	cuda	74.90%



- The best model gained 76.88% test accuracy. It was trained with Stochastic Gradient Descent (batch size 64, learning rate 0.016, with Nesterov momentum; 5-step learning rate decay by 0.1; cross-entropy objective and regularization on weights with 0.01 multiplier) for 10 epochs, then continued up to 15 epochs and stopped on 13th epoch due to Plateau. The architecture is:
  - 16 conv3x3 - relu - bn - maxpool2x2 ->
  - 32 conv3x3 - relu - bn - maxpool2x2 ->
  - 32 conv3x3 - relu - bn ->
  - 128 fc - relu ->
  - 64 fc - relu ->
  - 10 fc
- For the details see Experiments.ipynb
- I also compared the training time with CPU / GPU. See below the results:

1. **GPU (2 min)**

time python train.py

real 0m46.655s
user 1m51.989s
sys 0m9.353s

2. **CPU (22 min)**

time python train.py --no-cuda

real 3m12.462s
user 22m2.023s
sys 0m14.506s

The configurations fot this experiment were:

model	criterion	batch_size	optimizer	scheduler	n_epochs
NetCustom((pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1)) (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1)) (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (bn3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (fc1): Linear(in_features=1152, out_features=128, bias=True) (fc2): Linear(in_features=128, out_features=64, bias=True) (fc3): Linear(in_features=64, out_features=10, bias=True))	CrossEntropyLoss()	64	SGD(lr: 0.016, momentum: 0.9, dampening: 0, weight_decay: 0, nesterov: False)	StepLR(gamma: 0.1, step_size: 5)	10