11785
Fall 23
Recitation-4

# Hyperparameter Tuning

Harini Subramanyan, Jeel Shah

# What did we cover in Recitation 2?

# What did we cover in Recitation 2?

- Data pre-processing techniques
  - Data Quality
  - Data Normalization
- Experimentation methods
  - Model Architecture
  - Weight initialization
  - Optimizer
  - Batch Norm
- Hyperparameters
  - Learning Rate
  - Dropout

# What are covering today?

- Other Normalization methods
- Schedulers
- How to work in groups
- Grid and Random Search
- How to use wandb?

# Normalization methods

# Normalizations

- Batch Norm (paper)
- Layer Norm (paper)
- Weight Norm (paper)
- Instance Norm (paper)
- Group Norm (paper)
- Batch-Instance Norm (paper)
- Switchable Norm (paper)

# BatchNorm

- Normalizing batch of inputs to eliminate internal covariate shift
- torch.nn.BatchNorm1d(num_features) - Applies Batch Normalization over a 2D or 3D input
- For 4D inputs (N,C,H,W) - torch.nn.BatchNorm2d(num_features)

```
class Network(torch.nn.Module):
    def __init__(self, context):
        super(Network, self).__init__()
        in_size = (1+2*context)*15
        layers = [

            nn.Linear(in_size, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(),
            nn.Dropout(0.1),

            nn.Linear(1024, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(),
            nn.Dropout(0.1),

            nn.Linear(1024, 40)
        ]
        self.laysers = nn.Sequential(*layers)

    def forward(self, A0):
        x = self.laysers(A0)
        return x
```

# Layer Norm

**Problems with Batch Norm**

- Poor performance if the batch size is small, possible for high dimensional inputs
- Running mean and variance might not be the best thing to calculate for sequential algorithms like RNNs

Layer normalization calculates mean and variance for each item within the batch

Pytorch syntax - torch.nn.LayerNorm(normalized_shape)

# Some Other Methods

- Weight Norm
  - Normalizes weights of each layer
  - torch.nn.utils.weight_norm(nn.Linear(20, 40), name='weight')
- Group Norm
  - Applies normalization over a mini-batch of inputs but splitted in groups of size num_channels/num_groups
  - torch.nn.GroupNorm(num_groups, num_channels)
- Instance Norm
  - Calculates normalization parameters across individual channels/features for each input
  - torch.nn.InstanceNorm1d(num_features)
  - torch.nn.InstanceNorm2d(num_features)

# Schedulers

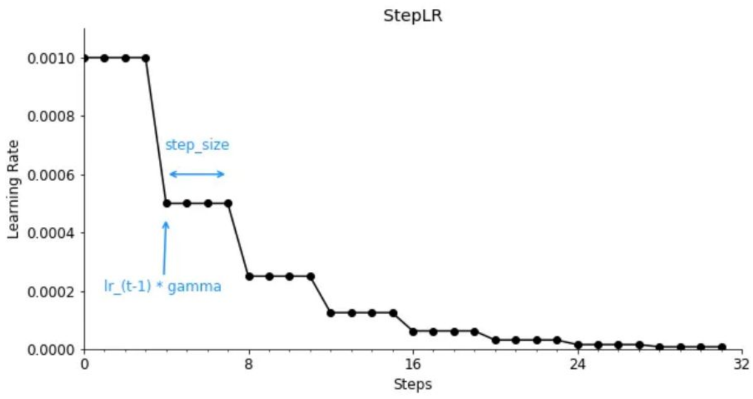
# What are Schedulers used for?

- The learning rate controls how big of a step for an optimizer to reach the minima of the loss function.
- A learning rate scheduler adjusts the learning rate according to a pre-defined schedule during the training process.
- PyTorch supports:
  - StepLR
  - MultiStepLR
  - ConstantLR
  - LinearLR
  - ExponentialLR
  - PolynomialLR
  - CosineAnnealingLR
  - CosineAnnealingWarmRestartsLR
  - CyclicLR
  - OneCycleLR
  - ReduceLROnPlateauLR
  - Custom Learning Rate Schedulers with Lambda Functions

# Step LR

- The StepLR reduces the learning rate by a multiplicative factor after every predefined number of training steps.

```
scheduler = StepLR(optimizer, step_size = 4, gamma = 0.5)
```

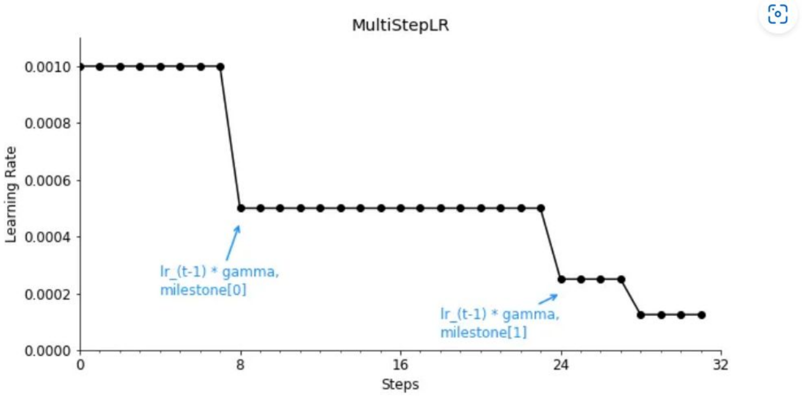



PyTorch Learning Rate Scheduler `StepLR` (Image by the author)

# Multistep LR

- The MultiStepLR — similarly to the StepLR — also reduces the learning rate by a multiplicative factor but after each pre-defined milestone.

```
scheduler = MultiStepLR(optimizer, milestones=[8, 24, 28], gamma =0.5)
```

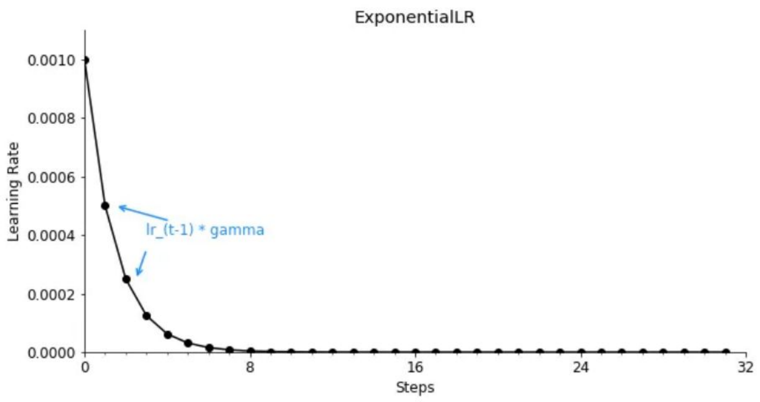



PyTorch Learning Rate Scheduler `MultiStepLR` (Image by the author)

# Exponential LR

- The ExponentialLR reduces learning rate by a multiplicative factor at every training step.

```
scheduler = ExponentialLR(optimizer, gamma = 0.5)
```

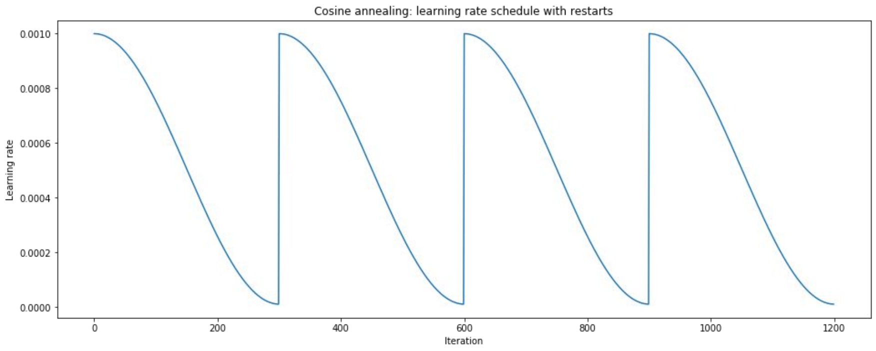


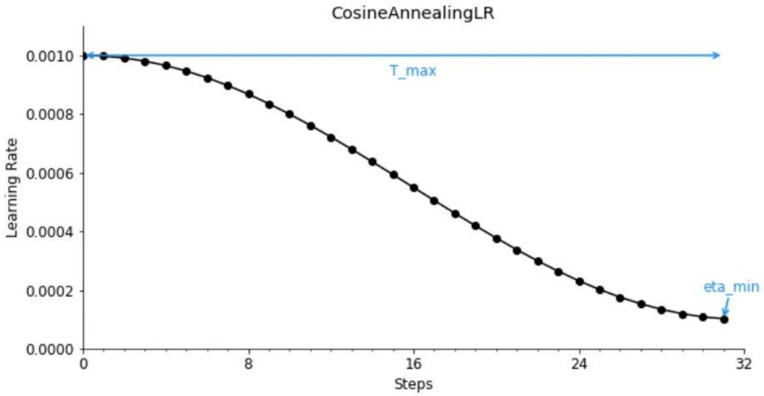

PyTorch Learning Rate Scheduler `ExponentialLR` (Image by the author)

# CosineAnnealingLR

- The CosineAnnealingLR reduces learning rate by a cosine function.
- While you could technically schedule the learning rate adjustments to follow multiple periods, the idea is to decay the learning rate over half a period for the maximum number of iterations.

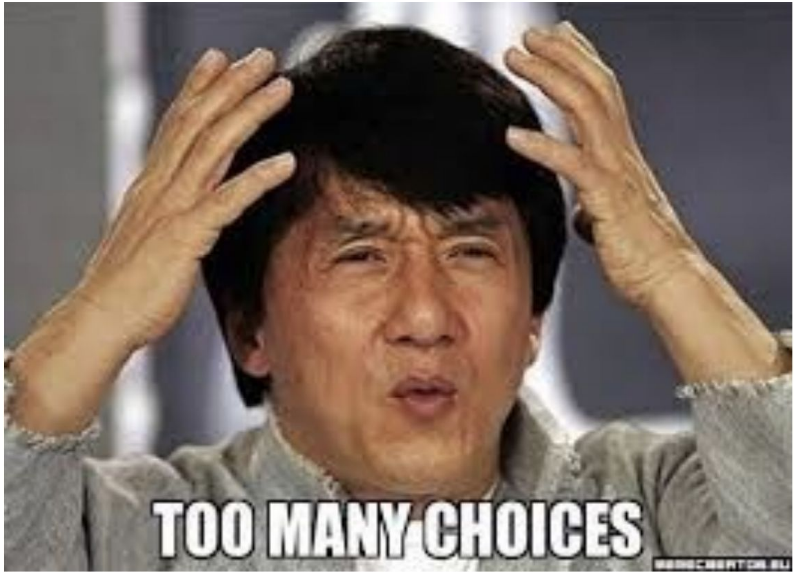
```
scheduler = CosineAnnealingLR(optimizer, T_max = 32,eta_min = 1e-4)
```



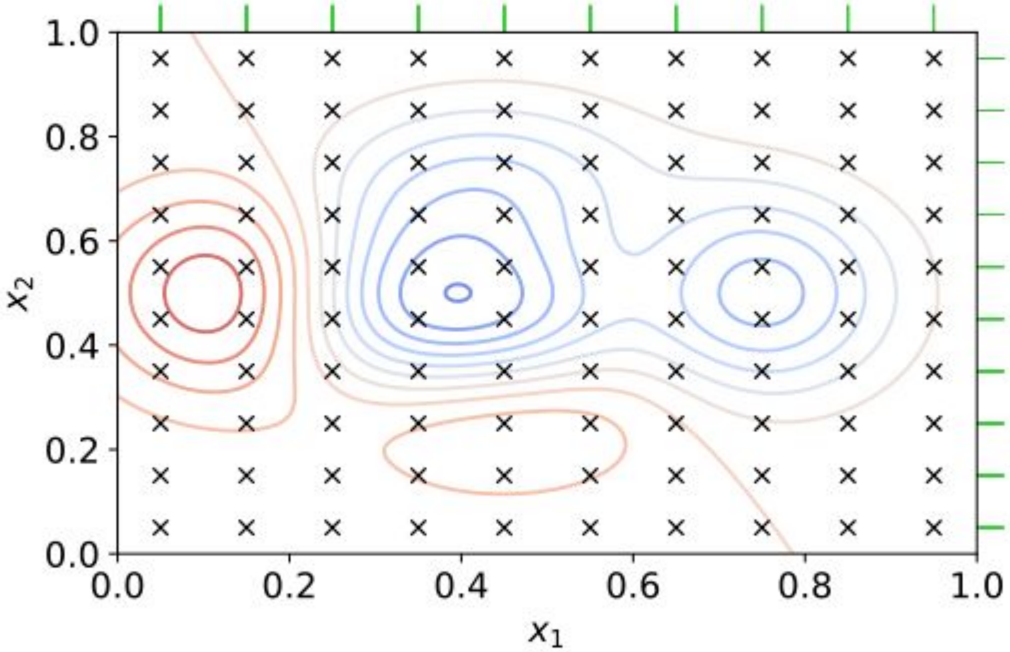

PyTorch Learning Rate Scheduler `CosineAnnealingLR` (Image by the author)

# Hyperparameter Searching


TOO MANY CHOICES

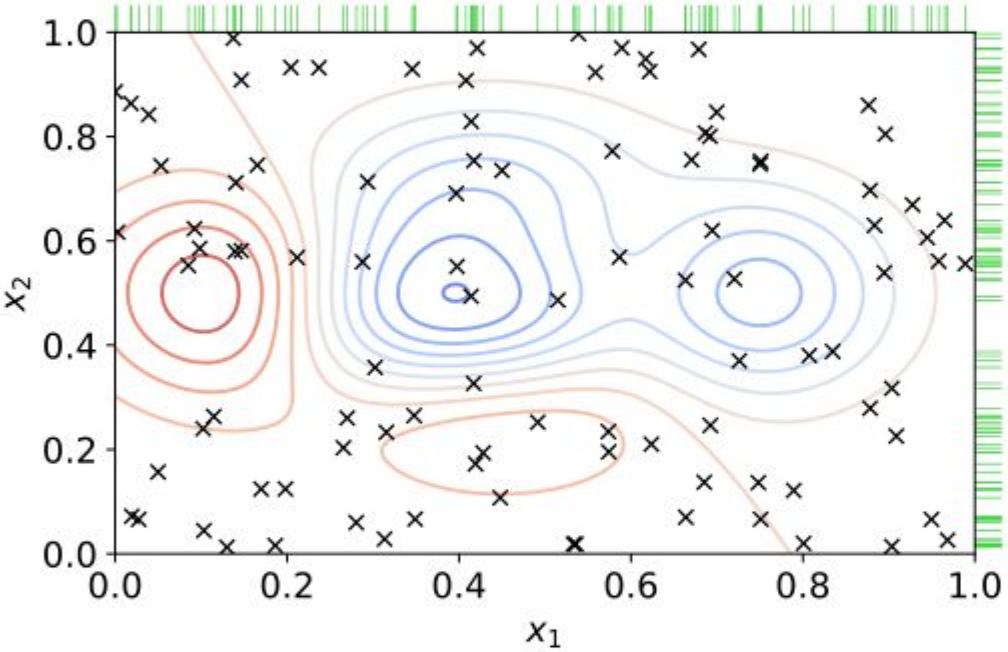
# Grid Search

```
MAX_HIDDEN_SIZE = 8000
MAX_CONTEXT = 64
hidden_size = (int)(x1 * MAX_HIDDEN_SIZE)
context = (int)(x2 * MAX_CONTEXT)
in_size = 2*context + 1
model = torch.nn.Sequential(
  torch.nn.Linear(in_size, hidden_size),
  torch.nn.ReLU(),
  torch.nn.Linear(hidden_size, out_size),
)
```



# Random Grid Search

```
MAX_HIDDEN_SIZE = 8000
MAX_CONTEXT = 64
hidden_size = (int)(x1 * MAX_HIDDEN_SIZE)
context = (int)(x2 * MAX_CONTEXT)
in_size = 2*context + 1
model = torch.nn.Sequential(
  torch.nn.Linear(in_size, hidden_size),
  torch.nn.ReLU(),
  torch.nn.Linear(hidden_size, out_size),
)
```



# Wandb for sweeps

```
[ ] # Initialize the sweep and set the method (grid, random or bayes"ian")

    sweep_config = {
        'method': 'random'
        }
```

```
[ ] # What is the objective of the sweep (minimize loss, maximize accuracy)

    metric = {
        'name':'loss',
        'goal':'minimize'
    }
    sweep_config['metric'] = metric
```

```
    # Hyperparameters to work with

    parameters_dict = {
        'optimizer':{
            'values': ['sgd', 'adam']
        },
        'learning_rate':{
            'distribution':'uniform',
            'min':2e-4,
            'max':1e-1
        },
        'batch_size': {
            'distribution': 'q_log_uniform_values',
            'q':4,
            'min': 16,
            'max': 128
        },
        'epochs':{
            'value': 5
        }
    }
    sweep_config['parameters'] = parameters_dict
```

Covered in Recitation 0P- part2

[Colab notebook]

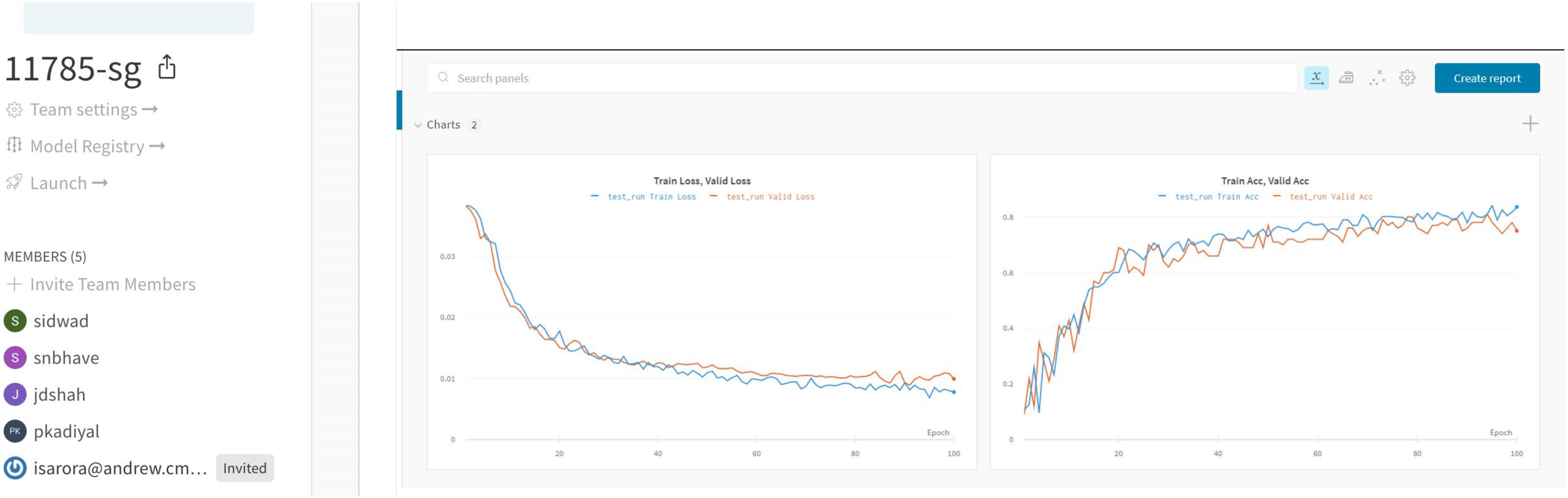
[Video]

Insights from sweep:
https://wandb.ai/11785-sg/CIFAR-Sweep/sweeps/5h4vbyqk

# Getting the most of Study Groups

- **Why study groups?**
  - Get more from the course
  - Move towards caviar hyperparameter tuning
  - Get to High Cutoffs!!
- **How to get the most out of them?**
  - Contact each other
  - Contact your mentors
  - Join the slack channels
  - Start creating ablation sheets

# Random Grid Search & WandB Team Ablations



# Interpreting wandb graphs