Problem1

April 6, 2022

1 Problem 1

We will consider five methods, AdaGrad, RMSProp, RMSProp+Nesterov, AdaDelta, Adam, and study their convergence using CIFAR-10 dataset. We will use multi-layer neural network model with two fully connected hidden layers with 1000 hidden units each and ReLU activation with minibatch size of 128.

1.1 1

Write the weight update equations for the five adaptive learning rate methods. Explain each term clearly. What are the hyperparameters in each policy? Explain how AdaDelta and Adam are different from RMSProp. (5+1)

We provide the SGD equation first for clearly explaining the following methods.

1.1.1 SGD

We use g_t to denote the gradient at time step $t.g_{t,i}$ is then the partial derivative of the objective function w.r.t. to the parameter θ_i at time step t:

$$g_{t,i} = \nabla_\theta J\left(\theta_{t,i}\right) \tag{1}$$

The SGD update for every parameter θ_i at each time step t then becomes:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}. \tag{2}$$

1.1.2 AdaGrad

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{3}$$

$G_t \in \mathbb{R}^{d\times d}$ here is a diagonal matrix where each diagonal element i,i is the sum of the squares of the gradients w.r.t. θ_i up to time step t, while ϵ is a smoothing term that avoids division by zero (usually on the order of $1e-8$).

We can also write the vectorized version of the update equation as:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{4}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{5}$$

η is the learning rate, ϵ is the smoothing term, and G_t is the the sum of the squares of the gradients.

Hyperparameters: η, ϵ

1.1.3 RMSProp

Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w. The sum of gradients is recursively updated using a decaying average of the past gradients.

$$E\left[g^2\right]_t = \gamma E\left[g^2\right]_{t-1} + (1-\gamma) g_t^2, \tag{6}$$

$E\left[g^2\right]$ is the decayed average of the past squared gradients, ϵ is the smoothing term, and γ is the decay rate.

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t \tag{7}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t \tag{8}$$

1.1.4 AdaDelta

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} g_t \tag{9}$$

$$E\left[\Delta\theta^2\right]_t = \gamma E\left[\Delta\theta^2\right]_{t-1} + (1-\gamma)\Delta\theta_t^2 \tag{10}$$

$$RMS[\Delta\theta]_t = \sqrt{E\left[\Delta\theta^2\right]_t + \epsilon} \tag{11}$$

$$\begin{aligned} \Delta\theta_t &= -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \\ \theta_{t+1} &= \theta_t + \Delta\theta_t \end{aligned} \tag{12}$$

Difference with RMSProp: The accumulated gradient $E\left[g^2\right]_t$ is decayed.

1.1.5 Adam

We compute the decaying averages of past and past squared gradients m_t and v_t respectively as follows:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\, g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2 \end{aligned} \tag{13}$$

m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively.

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{14}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{15}$$

Hyperparameters: β_1, β_2, ϵ

Difference with RMSProp: The gradient is not the decayed by sum of squares of the gradients, but the estimated mean of the past gradients.

1.2 2

Train the neural network using all the five methods with L2-regularization for 200 epochs each and plot the training loss vs number of epochs. Which method performs best (lowest training loss)?

AdaGrad, RMSProp, RMSProp+Nesterov, AdaDelta, Adam

[2]:
```python
import pytorch_lightning as pl
import torch.nn as nn
import torch
import torchvision
import torchvision.datasets as datasets
import torchvision.transforms as transforms
import wandb
from pytorch_lightning.loggers import WandbLogger

class MLP(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, use_dropout=False):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu1 = nn.ReLU()
        self.dropout1 = nn.Dropout(0.2)

        self.fc2 = nn.Linear(hidden_size, output_size)
        self.droup2 = nn.Dropout(0.5)
```

```
        self.use_dropout = use_dropout
    def forward(self, x):
        batch_size = x.size(0)
        x = x.view(batch_size, -1, self.fc1.in_features)
        if self.use_dropout:
            x = self.fc1(x)
            x = self.relu1(x)
            x = self.dropout1(x)
            x = self.fc2(x).squeeze()
            x = self.droup2(x)
        else:
            x = self.fc1(x)
            x = self.relu1(x)
            x = self.fc2(x).squeeze()
        return x

class MLPLightningModule(pl.LightningModule):
    def __init__(self, hidden_size, output_size, optimizer_name='adam',␣
↪use_dropout=False):
        super(MLPLightningModule, self).__init__()
        self.model = MLP(input_size=3*32*32, hidden_size=hidden_size,␣
↪output_size=output_size, use_dropout=use_dropout)
        self.loss_fn = nn.CrossEntropyLoss()
        self.optimizer_name = optimizer_name

    def forward(self, x):
        return self.model(x)

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.model(x)
        # print(y_hat.shape)
        loss = self.loss_fn(y_hat, y)
        self.log('train_loss', loss, prog_bar=True, on_epoch=True)
        logs = {'train_loss': loss}
        return {'loss': loss, 'log': logs}

    def train_dataloader(self):
        train_dataset = datasets.CIFAR10(root='./cached_datasets/CIFAR10',␣
↪train=True, download=True, transform=transforms.ToTensor())
        return torch.utils.data.DataLoader(train_dataset, batch_size=128,␣
↪shuffle=True, num_workers=20)

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.model(x)
        loss = self.loss_fn(y_hat, y)
```

```
        return {'val_loss': loss}
    def validation_end(self, outputs):
        avg_loss = torch.stack([x['val_loss'] for x in outputs]).mean()
        logs = {'val_loss': avg_loss}
        return {'val_loss': avg_loss, 'log': logs}

    def configure_optimizers(self):
        print(self.optimizer_name, self.optimizer_name == "RMSprop")
        if self.optimizer_name == 'Adagrad':
            optimizer = torch.optim.Adagrad(self.parameters(), lr=0.001,␣
↪weight_decay=1e-5)
        if self.optimizer_name == "RMSprop":
            optimizer = torch.optim.RMSprop(self.parameters(), lr=0.001,␣
↪weight_decay=1e-5)
        if self.optimizer_name == 'RMSprop+Nesterov':
            optimizer = torch.optim.RMSprop(self.parameters(), lr=0.001,␣
↪weight_decay=1e-5, momentum=0.9)
        if self.optimizer_name == 'Adadelta':
            optimizer = torch.optim.Adadelta(self.parameters(), lr=0.001,␣
↪weight_decay=1e-5)
        if self.optimizer_name == 'Adam':
            optimizer = torch.optim.Adam(self.parameters(), lr=0.001,␣
↪weight_decay=1e-5)
        return optimizer
```

1.2.1 Adagrad

[ ]:
```
run = wandb.init(group='1.2')
run.display(height=720)
optimizer_name = "Adagrad"
wandb.run.name = f"{optimizer_name}_cifar10"
logger = WandbLogger(project='HW3_1', save_dir="./outputs", name=optimizer_name)
trainer = pl.Trainer(gpus=1, max_epochs=200, logger=logger)
model = MLPLightningModule(hidden_size=1000, output_size=10,␣
↪optimizer_name=optimizer_name)
trainer.fit(model)
```

1.2.2 RMSprop

[10]:
```
run = wandb.init(group='1.2')
run.display(height=720)
optimizer_name = "RMSprop"
wandb.run.name = f"{optimizer_name}_cifar10"
logger = WandbLogger(project='HW3_1', save_dir="./outputs", name=optimizer_name)
trainer = pl.Trainer(gpus=1, max_epochs=200, logger=logger)
```

```
model = MLPLightningModule(hidden_size=1000, output_size=10,␣
 ↪optimizer_name=optimizer_name)
trainer.fit(model)
```

```
<IPython.core.display.HTML object>


<IPython.core.display.HTML object>

VBox(children=(Label(value='0.001 MB of 0.001 MB uploaded (0.000 MB␣
 ↪deduped)\r'), FloatProgress(value=1.0, max…

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/IPython/core/display.py:724: UserWarning: Consider using
IPython.display.IFrame instead
  warnings.warn("Consider using IPython.display.IFrame instead")

<IPython.core.display.HTML object>

GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/pytorch_lightning/trainer/configuration_validator.py:126: UserWarning:
You defined a `validation_step` but have no `val_dataloader`. Skipping val loop.
  rank_zero_warn("You defined a `validation_step` but have no `val_dataloader`.
Skipping val loop.")
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type             | Params
---------------------------------------------
0 | model   | MLP              | 3.1 M
1 | loss_fn | CrossEntropyLoss | 0
---------------------------------------------
3.1 M     Trainable params
0         Non-trainable params
3.1 M     Total params
12.332    Total estimated model params size (MB)

RMSprop True
Files already downloaded and verified
```

```
Training: 0it [00:00, ?it/s]
```

```
/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/pytorch_lightning/loggers/wandb.py:341: UserWarning: There is a wandb
run already in progress and newly created instances of `WandbLogger` will reuse
this run. If this is not desired, call `wandb.finish()` before instantiating
`WandbLogger`.
  rank_zero_warn(
```

1.2.3 RMSprop+Nesterov

[5]:
```
run = wandb.init(group='1.2')
run.display(height=720)
optimizer_name = "RMSprop+Nesterov"
wandb.run.name = f"{optimizer_name}_cifar10"
logger = WandbLogger(project='HW3_1', save_dir="./outputs", name=optimizer_name)
trainer = pl.Trainer(gpus=1, max_epochs=200, logger=logger)
model = MLPLightningModule(hidden_size=1000, output_size=10,␣
 ↪optimizer_name=optimizer_name)
trainer.fit(model)
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
VBox(children=(Label(value='0.001 MB of 0.001 MB uploaded (0.000 MB␣
 ↪deduped)\r'), FloatProgress(value=1.0, max…
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/IPython/core/display.py:724: UserWarning: Consider using
IPython.display.IFrame instead
  warnings.warn("Consider using IPython.display.IFrame instead")
```

```
<IPython.core.display.HTML object>
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/pytorch_lightning/trainer/configuration_validator.py:126: UserWarning:
```

```
You defined a `validation_step` but have no `val_dataloader`. Skipping val loop.
  rank_zero_warn("You defined a `validation_step` but have no `val_dataloader`.
Skipping val loop.")
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type             | Params
---------------------------------------------
0 | model   | MLP              | 3.1 M
1 | loss_fn | CrossEntropyLoss | 0
---------------------------------------------
3.1 M     Trainable params
0         Non-trainable params
3.1 M     Total params
12.332    Total estimated model params size (MB)
```

```
RMSprop+Nesterov False
Files already downloaded and verified
```

```
Training: 0it [00:00, ?it/s]
```

```
/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/pytorch_lightning/loggers/wandb.py:341: UserWarning: There is a wandb
run already in progress and newly created instances of `WandbLogger` will reuse
this run. If this is not desired, call `wandb.finish()` before instantiating
`WandbLogger`.
  rank_zero_warn(
```

1.2.4 Adadelta

[6]:
```
run = wandb.init(group='1.2')
run.display(height=720)
optimizer_name = "Adadelta"
wandb.run.name = f"{optimizer_name}_cifar10"
logger = WandbLogger(project='HW3_1', save_dir="./outputs", name=optimizer_name)
trainer = pl.Trainer(gpus=1, max_epochs=200, logger=logger)
model = MLPLightningModule(hidden_size=1000, output_size=10,␣
 ↪optimizer_name=optimizer_name)
trainer.fit(model)
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
VBox(children=(Label(value='0.001 MB of 0.001 MB uploaded (0.000 MB␣
 ↪deduped)\r'), FloatProgress(value=1.0, max…
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/IPython/core/display.py:724: UserWarning: Consider using
IPython.display.IFrame instead
  warnings.warn("Consider using IPython.display.IFrame instead")
```

```
<IPython.core.display.HTML object>
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/pytorch_lightning/trainer/configuration_validator.py:126: UserWarning:
You defined a `validation_step` but have no `val_dataloader`. Skipping val loop.
  rank_zero_warn("You defined a `validation_step` but have no `val_dataloader`.
Skipping val loop.")
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type             | Params
---------------------------------------------
0 | model   | MLP              | 3.1 M
1 | loss_fn | CrossEntropyLoss | 0
---------------------------------------------
3.1 M     Trainable params
0         Non-trainable params
3.1 M     Total params
12.332    Total estimated model params size (MB)
```

```
Adadelta False
Files already downloaded and verified
```

```
Training: 0it [00:00, ?it/s]
```

```
/home/xiangpan/.conda/envs/39/lib/python3.9/site-
packages/pytorch_lightning/loggers/wandb.py:341: UserWarning: There is a wandb
run already in progress and newly created instances of `WandbLogger` will reuse
this run. If this is not desired, call `wandb.finish()` before instantiating
`WandbLogger`.
  rank_zero_warn(
```

1.2.5 Adam

[3]:
```
run = wandb.init(group='1.2')
run.display(height=720)
optimizer_name = "Adam"
wandb.run.name = f"{optimizer_name}_cifar10"
logger = WandbLogger(project='HW3_1', save_dir="./outputs", name=optimizer_name)
trainer = pl.Trainer(gpus=1, max_epochs=200, logger=logger)
model = MLPLightningModule(hidden_size=1000, output_size=10,␣
 ↪optimizer_name=optimizer_name)
trainer.fit(model)
```

```
Failed to detect the name of this notebook, you can set it manually with the
WANDB_NOTEBOOK_NAME environment variable to enable code saving.
wandb: Currently logged in as: xiang-pan (use `wandb login
--relogin` to force relogin)

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

/home/xiangpan/.conda/envs/310/lib/python3.10/site-
packages/IPython/core/display.py:419: UserWarning: Consider using
IPython.display.IFrame instead
  warnings.warn("Consider using IPython.display.IFrame instead")

<IPython.core.display.HTML object>

/home/xiangpan/.conda/envs/310/lib/python3.10/site-
packages/pytorch_lightning/loggers/wandb.py:345: UserWarning: There is a wandb
run already in progress and newly created instances of `WandbLogger` will reuse
this run. If this is not desired, call `wandb.finish()` before instantiating
`WandbLogger`.
  rank_zero_warn(
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
/home/xiangpan/.conda/envs/310/lib/python3.10/site-
packages/pytorch_lightning/trainer/configuration_validator.py:133: UserWarning:
You defined a `validation_step` but have no `val_dataloader`. Skipping val loop.
  rank_zero_warn("You defined a `validation_step` but have no `val_dataloader`.
Skipping val loop.")
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type              | Params
---------------------------------------------
```

```
0 | model   | MLP              | 3.1 M
1 | loss_fn | CrossEntropyLoss | 0
---------------------------------------------
3.1 M     Trainable params
0         Non-trainable params
3.1 M     Total params
12.332    Total estimated model params size (MB)
```

```
Adam False
Files already downloaded and verified
```

```
Training: 0it [00:00, ?it/s]
```

1.2.6 Plot

[15]:
```python
import pandas as pd
df = pd.read_csv("./problem1/1_2.csv")
df
```

[15]:
```
     epoch  RMSprop_cifar10 - _step  RMSprop_cifar10 - _step__MIN  \
0        1                        7                             7
1        2                       16                            16
2        3                       25                            25
3        4                       34                            34
4        5                       43                            43
..     …                        …                             …
195    196                     1727                          1727
196    197                     1736                          1736
197    198                     1745                          1745
198    199                     1754                          1754
199    200                     1763                          1763

     RMSprop_cifar10 - _step__MAX  RMSprop_cifar10 - train_loss_epoch  \
0                               7                            2.663809
1                              16                            1.774574
2                              25                            1.688607
3                              34                            1.636065
4                              43                            1.593487
..                            …                                 …
195                          1727                            0.792143
196                          1736                            0.797538
197                          1745                            0.793693
198                          1754                            0.793397
199                          1763                            0.783713

     RMSprop_cifar10 - train_loss_epoch__MIN  \
0                                   2.663809
1                                   1.774574
```

```
2                                    1.688607
3                                    1.636065
4                                    1.593487
..                                        …
195                                  0.792143
196                                  0.797538
197                                  0.793693
198                                  0.793397
199                                  0.783713

     RMSprop_cifar10 - train_loss_epoch__MAX  \
0                                    2.663809
1                                    1.774574
2                                    1.688607
3                                    1.636065
4                                    1.593487
..                                        …
195                                  0.792143
196                                  0.797538
197                                  0.793693
198                                  0.793397
199                                  0.783713

     RMSprop+Nesterov_cifar10 - _step  RMSprop+Nesterov_cifar10 - _step__MIN  \
0                                   7                                      7
1                                  16                                     16
2                                  25                                     25
3                                  34                                     34
4                                  43                                     43
..                                ...                                    ...
195                              1727                                   1727
196                              1736                                   1736
197                              1745                                   1745
198                              1754                                   1754
199                              1763                                   1763

     RMSprop+Nesterov_cifar10 - _step__MAX  …  Adagrad_cifar10 - _step__MAX  \
0                                        7  …                             7
1                                       16  …                            16
2                                       25  …                            25
3                                       34  …                            34
4                                       43  …                            43
..                                     …  …                            …
195                                   1727  …                          1727
196                                   1736  …                          1736
197                                   1745  …                          1745
198                                   1754  …                          1754
```

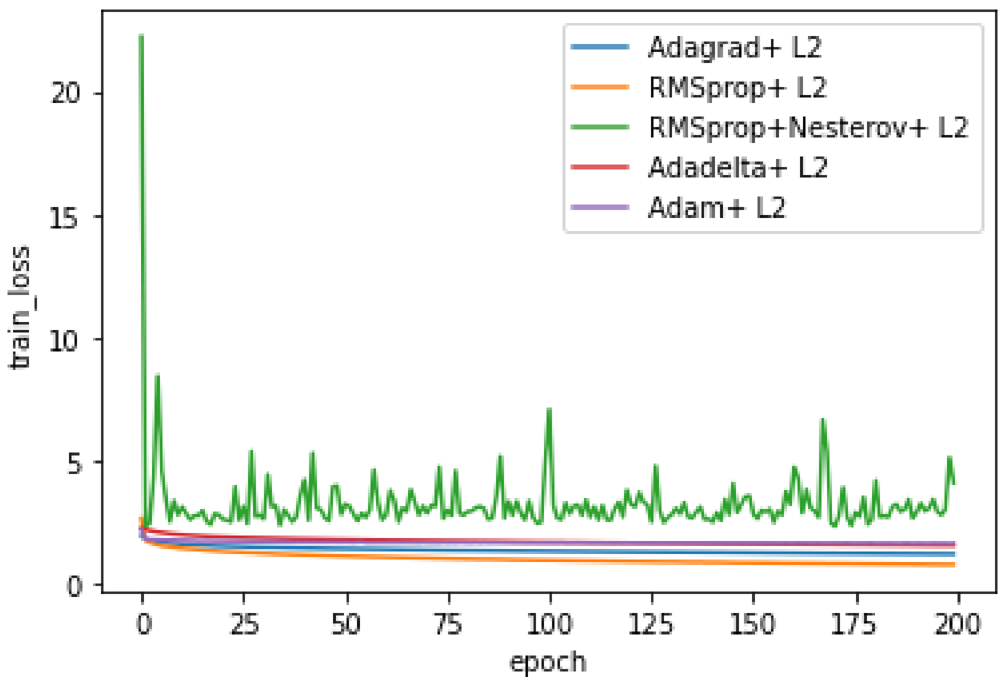
```
199                                  1763  …                           1763

     Adagrad_cifar10 - train_loss_epoch  \
0                              1.961369
1                              1.823225
2                              1.772303
3                              1.737732
4                              1.710066
..                                  …
195                            1.220296
196                            1.219495
197                            1.218581
198                            1.217557
199                            1.216772

     Adagrad_cifar10 - train_loss_epoch__MIN  \
0                                   1.961369
1                                   1.823225
2                                   1.772303
3                                   1.737732
4                                   1.710066
..                                       …
195                                 1.220296
196                                 1.219495
197                                 1.218581
198                                 1.217557
199                                 1.216772

     Adagrad_cifar10 - train_loss_epoch__MAX  Adadelta_cifar10 - _step  \
0                                   1.961369                         7
1                                   1.823225                        16
2                                   1.772303                        25
3                                   1.737732                        34
4                                   1.710066                        43
..                                       …                        …
195                                 1.220296                      1727
196                                 1.219495                      1736
197                                 1.218581                      1745
198                                 1.217557                      1754
199                                 1.216772                      1763

     Adadelta_cifar10 - _step__MIN  Adadelta_cifar10 - _step__MAX  \
0                                7                              7
1                               16                             16
2                               25                             25
3                               34                             34
4                               43                             43
```

```
..                          …                           …
195                      1727                        1727
196                      1736                        1736
197                      1745                        1745
198                      1754                        1754
199                      1763                        1763

     Adadelta_cifar10 - train_loss_epoch  \
0                              2.268250
1                              2.214725
2                              2.173219
3                              2.138775
4                              2.109075
..                                  …
195                            1.572403
196                            1.571365
197                            1.570372
198                            1.569414
199                            1.568389

     Adadelta_cifar10 - train_loss_epoch__MIN  \
0                                   2.268250
1                                   2.214725
2                                   2.173219
3                                   2.138775
4                                   2.109075
..                                       …
195                                 1.572403
196                                 1.571365
197                                 1.570372
198                                 1.569414
199                                 1.568389

     Adadelta_cifar10 - train_loss_epoch__MAX
0                                   2.268250
1                                   2.214725
2                                   2.173219
3                                   2.138775
4                                   2.109075
..                                       …
195                                 1.572403
196                                 1.571365
197                                 1.570372
198                                 1.569414
199                                 1.568389

[200 rows x 31 columns]
```

```
[16]: ori_optimize_name_list = ["Adagrad", "RMSprop", "RMSprop+Nesterov", "Adadelta",␣
       ↪"Adam"]
      post_fix = "_cifar10 - train_loss_epoch"
      optimize_name_list = [x + post_fix for x in ori_optimize_name_list]
      display_name_list = [x + "+ L2" for x in ori_optimize_name_list]
      data = df[optimize_name_list]
```

```
[18]: # plot data
      import matplotlib.pyplot as plt
      data = df[optimize_name_list]
      plt.plot(data)
      plt.legend(display_name_list)
      plt.xlabel("epoch")
      plt.ylabel("train_loss")
```

```
[18]: Text(0, 0.5, 'train_loss')
```

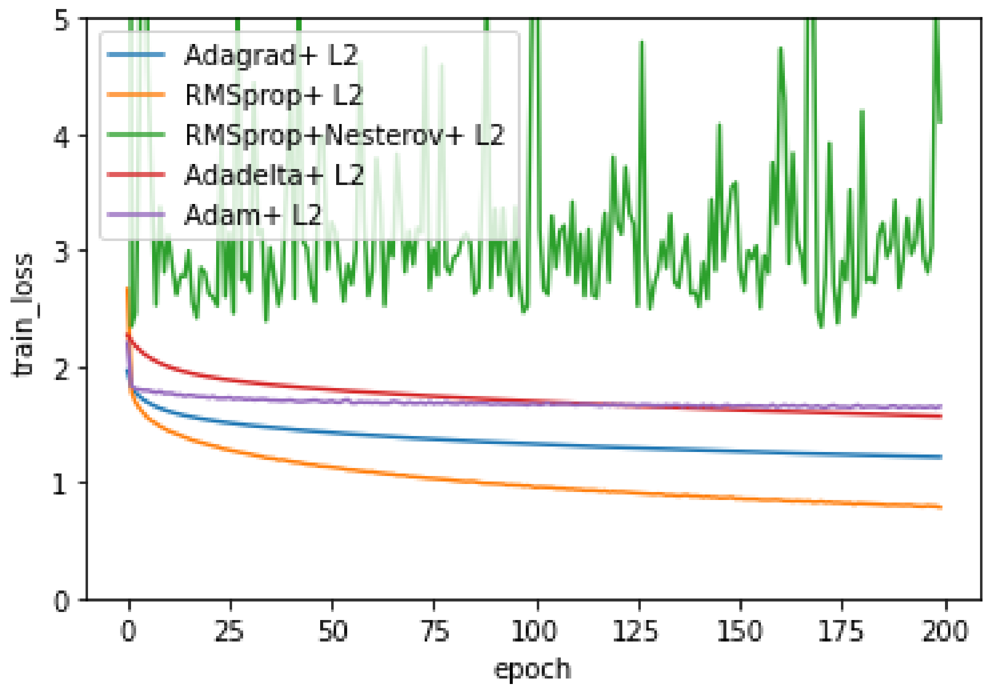


We set the range of training_loss to give more clear visualization.

```
[19]: # plot data
      import matplotlib.pyplot as plt
      data = df[optimize_name_list]
      plt.plot(data)
      plt.legend(display_name_list)
      plt.ylim(0, 5)
```

```
plt.xlabel("epoch")
plt.ylabel("train_loss")
```

[19]:
```
Text(0, 0.5, 'train_loss')
```



RMSprop+Nesterov performs best.

1.3 3

Add dropout (probability 0.2 for input layer and 0.5 for hidden layers) and train the neural network again using all the five methods for 200 epochs. Compare the training loss with that in part 2. Which method performs the best? For the five methods, compare their training time (to finish 200 epochs with dropout) to the training time in part 2 (to finish 200 epochs without dropout). (5)

[ ]:
```
'''
Author: Xiang Pan
Date: 2022-03-26 16:51:35
LastEditTime: 2022-03-26 17:40:20
LastEditors: Xiang Pan
Description:
FilePath: /HW3/problem1/1_3.py
@email: xiangpan@nyu.edu
'''
import argparse
import pytorch_lightning as pl
```

```
import torch.nn as nn
import torch
import torchvision
import torchvision.datasets as datasets
import torchvision.transforms as transforms
import wandb
from pytorch_lightning.loggers import WandbLogger

class MLP(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, use_dropout=False):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu1 = nn.ReLU()
        self.dropout1 = nn.Dropout(0.2)

        self.fc2 = nn.Linear(hidden_size, output_size)
        self.droup2 = nn.Dropout(0.5)

        self.use_dropout = use_dropout
    def forward(self, x):
        batch_size = x.size(0)
        x = x.view(batch_size, -1, self.fc1.in_features)
        if self.use_dropout:
            x = self.fc1(x)
            x = self.relu1(x)
            x = self.dropout1(x)
            x = self.fc2(x).squeeze()
            x = self.droup2(x)
        else:
            x = self.fc1(x)
            x = self.relu1(x)
            x = self.fc2(x).squeeze()
        return x

class MLPLightningModule(pl.LightningModule):
    def __init__(self, hidden_size, output_size, optimizer_name='adam',
 ↪use_dropout=False):
        super(MLPLightningModule, self).__init__()
        self.model = MLP(input_size=3*32*32, hidden_size=hidden_size,
 ↪output_size=output_size, use_dropout=use_dropout)
        self.loss_fn = nn.CrossEntropyLoss()
        self.optimizer_name = optimizer_name

    def forward(self, x):
        return self.model(x)

    def training_step(self, batch, batch_idx):
```

```
        x, y = batch
        y_hat = self.model(x)
        # print(y_hat.shape)
        loss = self.loss_fn(y_hat, y)
        self.log('train_loss', loss, prog_bar=True, on_epoch=True)
        logs = {'train_loss': loss}
        return {'loss': loss, 'log': logs}

    def train_dataloader(self):
        train_dataset = datasets.CIFAR10(root='./cached_datasets/CIFAR10',␣
 ↪train=True, download=True, transform=transforms.ToTensor())
        return torch.utils.data.DataLoader(train_dataset, batch_size=128,␣
 ↪shuffle=True, num_workers=20)

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.model(x)
        loss = self.loss_fn(y_hat, y)
        return {'val_loss': loss}
    def validation_end(self, outputs):
        avg_loss = torch.stack([x['val_loss'] for x in outputs]).mean()
        logs = {'val_loss': avg_loss}
        return {'val_loss': avg_loss, 'log': logs}

    def configure_optimizers(self):
        # print(self.optimizer_name, self.optimizer_name == "RMSprop")
        if self.optimizer_name == 'Adagrad':
            optimizer = torch.optim.Adagrad(self.parameters(), lr=0.001)
        if self.optimizer_name == "RMSprop":
            optimizer = torch.optim.RMSprop(self.parameters(), lr=0.001)
        if self.optimizer_name == 'RMSprop+Nesterov':
            optimizer = torch.optim.RMSprop(self.parameters(), lr=0.001,␣
 ↪momentum=0.9)
        if self.optimizer_name == 'Adadelta':
            optimizer = torch.optim.Adadelta(self.parameters(), lr=0.001)
        if self.optimizer_name == 'Adam':
            optimizer = torch.optim.Adam(self.parameters(), lr=0.001)
        return optimizer



if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--use_dropout', action='store_true')
    parser.add_argument('--optimizer_name', type=str, default="Adagrad")
    parser.add_argument('--gpus', nargs='+', type=int, default=[0])
    args = parser.parse_args()
```

```
    run = wandb.init(group='1.3')
    run.display(height=720)
    optimizer_name = args.optimizer_name

    log_name = optimizer_name
    if args.use_dropout:
        log_name += "+Dropout"

    wandb.run.name = log_name
    logger = WandbLogger(project='HW3_1', save_dir="./outputs", name=log_name)
    trainer = pl.Trainer(gpus=args.gpus, max_epochs=200, logger=logger)
    model = MLPLightningModule(hidden_size=1000, output_size=10,␣
↪optimizer_name=optimizer_name, use_dropout=args.use_dropout)
    trainer.fit(model)
```

[22]:

```
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("./problem1/1_3.csv")
df
```

[22]:

```
     epoch  Adagrad+Dropout - _step  Adagrad+Dropout - _step__MIN  \
0        1                        7                            7
1        2                       16                           16
2        3                       25                           25
3        4                       34                           34
4        5                       43                           43
..     …                      …                           …
195    196                     1727                         1727
196    197                     1736                         1736
197    198                     1745                         1745
198    199                     1754                         1754
199    200                     1763                         1763

     Adagrad+Dropout - _step__MAX  Adagrad+Dropout - train_loss_epoch  \
0                               7                            2.159661
1                              16                            2.073707
2                              25                            2.046373
3                              34                            2.029272
4                              43                            2.011730
..                            …                                 …
195                          1727                            1.776351
196                          1736                            1.775455
197                          1745                            1.777960
198                          1754                            1.773476
199                          1763                            1.775750
```

```
     Adagrad+Dropout - train_loss_epoch__MIN  \
0                                 2.159661
1                                 2.073707
2                                 2.046373
3                                 2.029272
4                                 2.011730
..                                     …
195                               1.776351
196                               1.775455
197                               1.777960
198                               1.773476
199                               1.775750

     Adagrad+Dropout - train_loss_epoch__MAX  \
0                                 2.159661
1                                 2.073707
2                                 2.046373
3                                 2.029272
4                                 2.011730
..                                     …
195                               1.776351
196                               1.775455
197                               1.777960
198                               1.773476
199                               1.775750

     RMSprop+Nesterov+Dropout - _step  RMSprop+Nesterov+Dropout - _step__MIN  \
0                                   7                                      7
1                                  16                                     16
2                                  25                                     25
3                                  34                                     34
4                                  43                                     43
..                                …                                      …
195                              1727                                   1727
196                              1736                                   1736
197                              1745                                   1745
198                              1754                                   1754
199                              1763                                   1763

     RMSprop+Nesterov+Dropout - _step__MAX  …  Adam+Dropout - _step__MAX  \
0                                        7  …                          7
1                                       16  …                         16
2                                       25  …                         25
3                                       34  …                         34
4                                       43  …                         43
..                                     …  …                        …
195                                   1727  …                       1727
```

```
196                              1736  …                      1736
197                              1745  …                      1745
198                              1754  …                      1754
199                              1763  …                      1763

     Adam+Dropout - train_loss_epoch  Adam+Dropout - train_loss_epoch__MIN  \
0                           2.172784                              2.172784
1                           2.064521                              2.064521
2                           2.042034                              2.042034
3                           2.027263                              2.027263
4                           2.018904                              2.018904
..                               …                                   …
195                         1.816487                              1.816487
196                         1.816880                              1.816880
197                         1.811998                              1.811998
198                         1.814392                              1.814392
199                         1.823912                              1.823912

     Adam+Dropout - train_loss_epoch__MAX  RMSprop+Dropout - _step  \
0                                2.172784                        7
1                                2.064521                       16
2                                2.042034                       25
3                                2.027263                       34
4                                2.018904                       43
..                                    …                       …
195                              1.816487                     1727
196                              1.816880                     1736
197                              1.811998                     1745
198                              1.814392                     1754
199                              1.823912                     1763

     RMSprop+Dropout - _step__MIN  RMSprop+Dropout - _step__MAX  \
0                               7                             7
1                              16                            16
2                              25                            25
3                              34                            34
4                              43                            43
..                            …                            …
195                          1727                          1727
196                          1736                          1736
197                          1745                          1745
198                          1754                          1754
199                          1763                          1763

     RMSprop+Dropout - train_loss_epoch  \
0                              3.009720
1                              2.081418
```

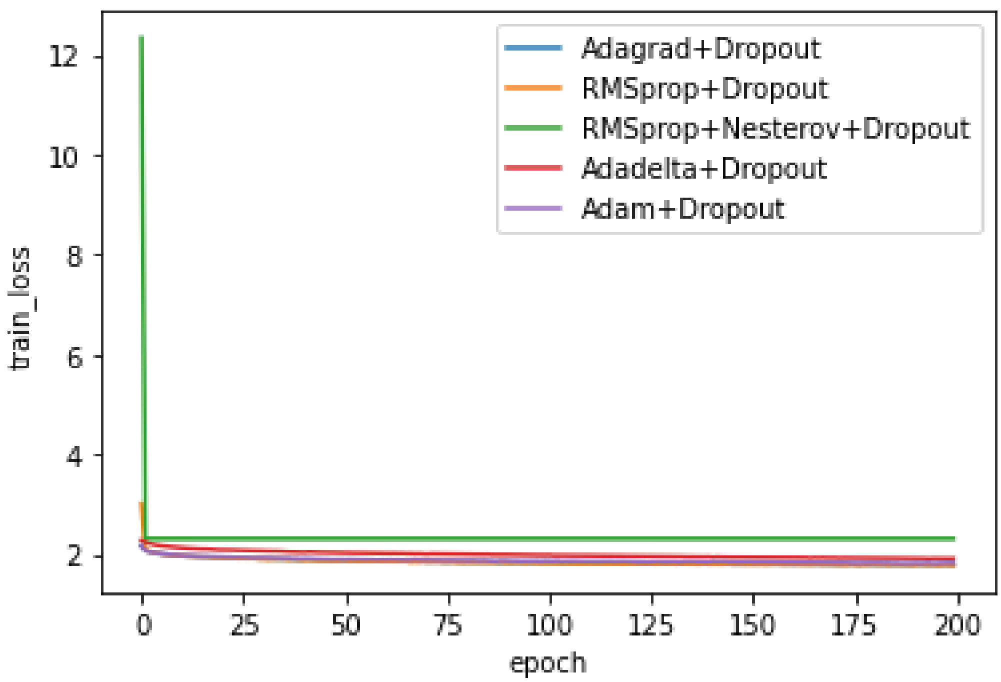
```
2                             2.052173
3                             2.031971
4                             2.014198
..                                 …
195                           1.763995
196                           1.769017
197                           1.773915
198                           1.771836
199                           1.765055

     RMSprop+Dropout - train_loss_epoch__MIN  \
0                                   3.009720
1                                   2.081418
2                                   2.052173
3                                   2.031971
4                                   2.014198
..                                       …
195                                 1.763995
196                                 1.769017
197                                 1.773915
198                                 1.771836
199                                 1.765055

     RMSprop+Dropout - train_loss_epoch__MAX
0                                   3.009720
1                                   2.081418
2                                   2.052173
3                                   2.031971
4                                   2.014198
..                                       …
195                                 1.763995
196                                 1.769017
197                                 1.773915
198                                 1.771836
199                                 1.765055

[200 rows x 31 columns]
```

[23]:
```python
ori_optimize_name_list = ["Adagrad", "RMSprop", "RMSprop+Nesterov", "Adadelta",␣
 ↪"Adam"]
post_fix = "+Dropout - train_loss_epoch"
optimize_name_list = [x + post_fix for x in ori_optimize_name_list]
display_name_list = [x + "+Dropout" for x in ori_optimize_name_list]
data = df[optimize_name_list]
```

1.3.1 Plot

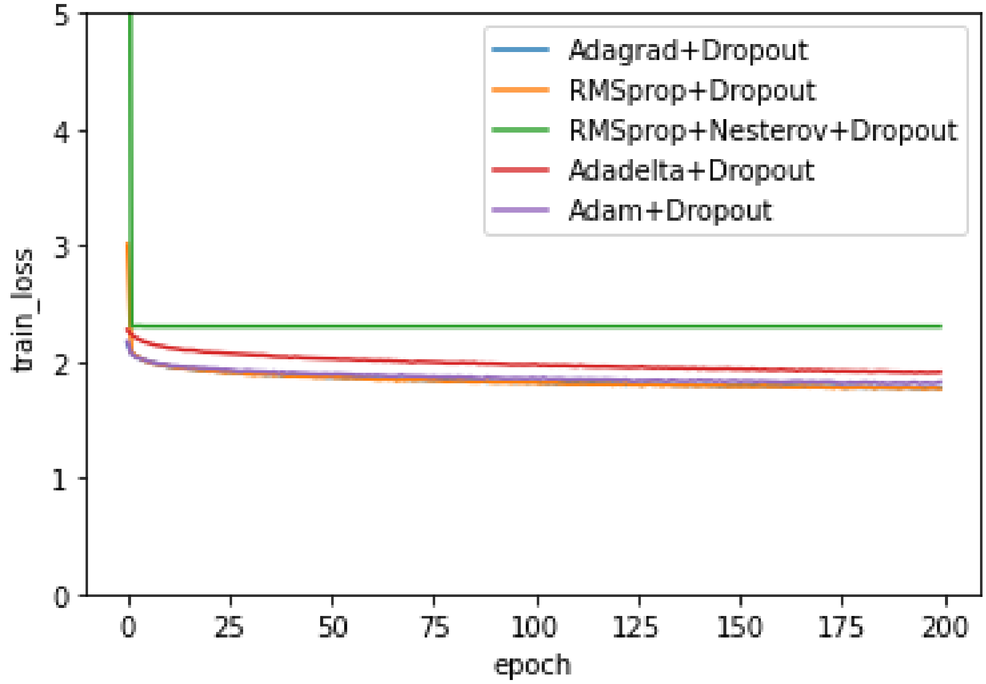
[24]:
```
import matplotlib.pyplot as plt
data = df[optimize_name_list]
plt.plot(data)
plt.legend(display_name_list)
plt.xlabel("epoch")
plt.ylabel("train_loss")
```

[24]:
```
Text(0, 0.5, 'train_loss')
```



[25]:
```
import matplotlib.pyplot as plt
data = df[optimize_name_list]
plt.plot(data)
plt.legend(display_name_list)
plt.xlabel("epoch")
plt.ylim(0, 5)
plt.ylabel("train_loss")
```

[25]:
```
Text(0, 0.5, 'train_loss')
```



1.3.2 Compare the training loss

[13]:
```python
df = pd.read_csv("./problem1/1_2.csv")
ori_optimize_name_list = ["Adagrad", "RMSprop", "RMSprop+Nesterov", "Adadelta",␣
 ↪"Adam"]
post_fix = "_cifar10 - train_loss_epoch"
optimize_name_list = [x + post_fix for x in ori_optimize_name_list]
data = df[optimize_name_list]
data.iloc[199]
```

[13]:
```
Adagrad_cifar10 - train_loss_epoch            1.216772
RMSprop_cifar10 - train_loss_epoch            0.783713
RMSprop+Nesterov_cifar10 - train_loss_epoch   4.109675
Adadelta_cifar10 - train_loss_epoch           1.568389
Adam_cifar10 - train_loss_epoch               1.653735
Name: 199, dtype: float64
```

[14]:
```python
df = pd.read_csv("./problem1/1_3.csv")
ori_optimize_name_list = ["Adagrad", "RMSprop", "RMSprop+Nesterov", "Adadelta",␣
 ↪"Adam"]
post_fix = "+Dropout - train_loss_epoch"
optimize_name_list = [x + post_fix for x in ori_optimize_name_list]
data = df[optimize_name_list]
data.iloc[199]
```

[14]:
```
Adagrad+Dropout - train_loss_epoch           1.775750
RMSprop+Dropout - train_loss_epoch           1.765055
RMSprop+Nesterov+Dropout - train_loss_epoch  2.303345
Adadelta+Dropout - train_loss_epoch          1.910109
Adam+Dropout - train_loss_epoch              1.823912
Name: 199, dtype: float64
```

RMSprop_L2: 0.783713

RMSprop+Dropout: 1.765055

The L2 method have lower training loss, the best optimization problem is RMSprop.

1.3.3 Compare the training time

Method	Training_Time
Adagrad+L2	8m 5s
RMSprop+L2	9m 9s
RMSprop+Nesterov+L2	8m 21s
Adadelta+L2	8m 40s
Adam+L2	8m 10s
Adagrad+Dropout	24m 15s
RMSprop+Dropout	26m 14s
RMSprop+Nesterov+Dropout	24m 18s
Adadelta+Dropout	24m 41s
Adam+Dropout	24m 25s

1.4 4

Compare test accuracy of trained model for all the five methods from part 2 and part 3. Note that to calculate test accuracy of model trained using dropout you need to appropriately scale the weights (by the dropout probability).

[11]:
```python
import pandas as pd
import wandb
api = wandb.Api()

# Project is specified by <entity/project-name>
runs = api.runs("xiang-pan/NYU_DL_Sys-HW3")

summary_list, config_list, name_list = [], [], []
run_dict = {}
for run in runs:
    name_list.append(run.name)
    name = run.name.split("_")[0]
    run_dict[name] = run.id
print(run_dict)
```

```
{'Adam': '3lmyveoh', 'Adadelta': '19z7xfa2', 'RMSprop+Nesterov': '1klv9acs',
'RMSprop': '2jtip8s5', 'Adagrad': '3tj7qb6o'}
```

[9]:
```
test_dataloader = torch.utils.data.DataLoader(datasets.CIFAR10(root='./
↪cached_datasets/CIFAR10', train=False, download=True, transform=transforms.
↪ToTensor()), batch_size=128, shuffle=True, num_workers=20)
```

```
Files already downloaded and verified
```

[17]:
```
def get_lastest_file_from_dir(dir):
    # get lastest file
    import os
    import glob
    file_list = glob.glob(dir + "/*")
    file_list.sort(key=os.path.getmtime)
    return file_list[-1]
```

[74]:
```
import pytorch_lightning as pl
import torch.nn as nn
import torch
import torchvision
import torchvision.datasets as datasets
import torchvision.transforms as transforms
import wandb
from pytorch_lightning.loggers import WandbLogger
import torchmetrics

class MLP(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, use_dropout=False):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu1 = nn.ReLU()
        self.dropout1 = nn.Dropout(0.2)

        self.fc2 = nn.Linear(hidden_size, output_size)
        self.droup2 = nn.Dropout(0.5)

        self.use_dropout = use_dropout
    def forward(self, x):
        batch_size = x.size(0)
        x = x.view(batch_size, -1, self.fc1.in_features)
        if self.use_dropout:
            x = self.fc1(x)
            x = self.relu1(x)
            x = self.dropout1(x)
            x = self.fc2(x).squeeze()
            x = self.droup2(x)
```

```
        else:
            x = self.fc1(x)
            x = self.relu1(x)
            x = self.fc2(x).squeeze()
        return x

class MLPLightningModule(pl.LightningModule):
    def __init__(self, hidden_size, output_size, optimizer_name='adam',␣
 ↪use_dropout=False):
        super(MLPLightningModule, self).__init__()
        self.model = MLP(input_size=3*32*32, hidden_size=hidden_size,␣
 ↪output_size=output_size, use_dropout=use_dropout)
        self.loss_fn = nn.CrossEntropyLoss()
        self.acc_metric = torchmetrics.Accuracy()
        self.optimizer_name = optimizer_name

    def forward(self, x):
        return self.model(x)

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.model(x)
        # print(y_hat.shape)
        loss = self.loss_fn(y_hat, y)
        self.log('train_loss', loss, prog_bar=True, on_epoch=True)
        logs = {'train_loss': loss}
        return {'loss': loss, 'log': logs}

    def train_dataloader(self):
        train_dataset = datasets.CIFAR10(root='./cached_datasets/CIFAR10',␣
 ↪train=True, download=True, transform=transforms.ToTensor())
        return torch.utils.data.DataLoader(train_dataset, batch_size=128,␣
 ↪shuffle=True, num_workers=20)

    def validation_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.model(x)
        loss = self.loss_fn(y_hat, y)
        return {'val_loss': loss}
    def validation_end(self, outputs):
        avg_loss = torch.stack([x['val_loss'] for x in outputs]).mean()
        logs = {'val_loss': avg_loss}
        return {'val_loss': avg_loss, 'log': logs}

    def test_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.model(x)
```

```
        loss = self.loss_fn(y_hat, y)
        acc = self.acc_metric(y_hat, y)
        self.log('test_loss', loss, on_step=False, on_epoch=True)
        self.log('test_acc', acc, on_step=False, on_epoch=True)
        return {'test_loss': loss, 'test_acc': acc}

    def configure_optimizers(self):
        print(self.optimizer_name, self.optimizer_name == "RMSprop")
        if self.optimizer_name == 'Adagrad':
            optimizer = torch.optim.Adagrad(self.parameters(), lr=0.001,␣
 ↪weight_decay=1e-5)
        if self.optimizer_name == "RMSprop":
            optimizer = torch.optim.RMSprop(self.parameters(), lr=0.001,␣
 ↪weight_decay=1e-5)
        if self.optimizer_name == 'RMSprop+Nesterov':
            optimizer = torch.optim.RMSprop(self.parameters(), lr=0.001,␣
 ↪weight_decay=1e-5, momentum=0.9)
        if self.optimizer_name == 'Adadelta':
            optimizer = torch.optim.Adadelta(self.parameters(), lr=0.001,␣
 ↪weight_decay=1e-5)
        if self.optimizer_name == 'Adam':
            optimizer = torch.optim.Adam(self.parameters(), lr=0.001,␣
 ↪weight_decay=1e-5)
        return optimizer
```

[75]:
```
def test_model(model_name):
    # model = MLPLightningModule(hidden_size=1000, output_size=10,␣
 ↪optimizer_name=optimizer_name)
    dir = f"./outputs/{model_name}/version_None/checkpoints/"
    file = get_lastest_file_from_dir(dir)
    model = MLPLightningModule.load_from_checkpoint(file, hidden_size=1000,␣
 ↪output_size=10, optimizer_name=optimizer_name)
    trainer = pl.Trainer(gpus=[0], max_epochs=200, logger=logger)
    res = trainer.test(model, test_dataloader)
    return res
res = test_model("Adagrad")
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
/home/xiangpan/.conda/envs/310/lib/python3.10/site-
packages/pytorch_lightning/trainer/connectors/data_connector.py:486:
PossibleUserWarning: Your `test_dataloader`'s sampler has shuffling enabled, it
is strongly recommended that you turn shuffling off for val/test/predict
dataloaders.
```

```
  rank_zero_warn(
```

```
Testing: 0it [00:00, ?it/s]
```

```
        Test metric             DataLoader 0

         test_acc            0.5260000228881836
         test_loss           1.3472567796707153
```

[76]:

```python
res_dict = {}
for op in ["Adagrad", "RMSprop", "RMSprop+Nesterov", "Adadelta", "Adam"]:
    print(op)
    res = test_model(op)
    res_dict[op+"+L2"] = res
    print(op, res)
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
```

```
Adagrad
```

```
Testing: 0it [00:00, ?it/s]
```

```
        Test metric             DataLoader 0

         test_acc            0.5260000228881836
         test_loss           1.3472566604614258
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
```

```
Adagrad [{'test_loss': 1.3472566604614258, 'test_acc': 0.5260000228881836}]
RMSprop
```

```
Testing: 0it [00:00, ?it/s]
```

```
        Test metric             DataLoader 0

         test_acc            0.4681999981403351
```

test_loss	2.165738582611084

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

RMSprop [{'test_loss': 2.165738582611084, 'test_acc': 0.4681999981403351}]
RMSprop+Nesterov

Testing: 0it [00:00, ?it/s]
```

Test metric	DataLoader 0
test_acc	0.19689999520778656
test_loss	2.1493773460388184

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

RMSprop+Nesterov [{'test_loss': 2.1493773460388184, 'test_acc':
0.19689999520778656}]
Adadelta

Testing: 0it [00:00, ?it/s]
```

Test metric	DataLoader 0
test_acc	0.4507000148296356
test_loss	1.5840909481048584

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

Adadelta [{'test_loss': 1.5840909481048584, 'test_acc': 0.4507000148296356}]
Adam

Testing: 0it [00:00, ?it/s]
```

```
   Test metric            DataLoader 0

    test_acc           0.34540000557899475
    test_loss          1.8569005727767944
```

```
Adam [{'test_loss': 1.8569005727767944, 'test_acc': 0.34540000557899475}]
```

[80]:
```python
res_dict_2 = res_dict
```

[81]:
```python
res_dict_2
```

[81]:
```
{'Adagrad+L2': [{'test_loss': 1.3472566604614258,
   'test_acc': 0.5260000228881836}],
 'RMSprop+L2': [{'test_loss': 2.165738582611084,
   'test_acc': 0.4681999981403351}],
 'RMSprop+Nesterov+L2': [{'test_loss': 2.1493773460388184,
   'test_acc': 0.19689999520778656}],
 'Adadelta+L2': [{'test_loss': 1.5840909481048584,
   'test_acc': 0.4507000148296356}],
 'Adam+L2': [{'test_loss': 1.8569005727767944,
   'test_acc': 0.34540000557899475}]}
```

[58]:
```python
import pandas as pd
import wandb
api = wandb.Api()

# Project is specified by <entity/project-name>
runs = api.runs("xiang-pan/NYU_DL_Sys-HW3_Problem1")

summary_list, config_list, name_list = [], [], []
run_dict = {}
for run in runs:
    name_list.append(run.name)
    if "Dropout" not in run.name:
        continue
    name = run.name.split("_")[0]
    run_dict[name] = run.id
print(run_dict)
```

```
{'Adagrad+Dropout': '3d5ryojb', 'RMSprop+Nesterov+Dropout': '19p3qzn4',
'Adadelta+Dropout': '378qsqof', 'Adam+Dropout': '2a71cof7', 'RMSprop+Dropout':
'3eke1ws1'}
```

[77]:
```python
import json
res_dict_3 = json.load(open("./problem1/1_3.json"))
```

```
res_dict_3
```

[77]:

```
{'Adagrad+Dropout': [{'test_loss': 1.5562536716461182,
   'test_acc': 0.5156000256538391}],
 'RMSprop+Dropout': [{'test_loss': 1.6157069206237793,
   'test_acc': 0.4878000020980835}],
 'RMSprop+Nesterov+Dropout': [{'test_loss': 2.303044080734253,
   'test_acc': 0.10000000149011612}],
 'Adadelta+Dropout': [{'test_loss': 1.730869174003601,
   'test_acc': 0.45730000734329224}],
 'Adam+Dropout': [{'test_loss': 1.6133182048797607,
   'test_acc': 0.4763999879360199}]}
```

[82]:

```
index_name = ["Adagrad", "RMSprop", "RMSprop+Nesterov", "Adadelta", "Adam"]
col = ["test_loss_L2", "test_acc_L2", "test_loss_Dropout", "test_acc_Dropout"]
df = pd.DataFrame(index=index_name, columns=col)
for i in range(len(index_name)):
    name = index_name[i]
    name_2 = name + "+L2"
    name_3 = name + "+Dropout"
    row = [res_dict_2[name_2][0]["test_loss"],␣
 ↪res_dict_2[name_2][0]["test_acc"], res_dict_3[name_3][0]["test_loss"],␣
 ↪res_dict_3[name_3][0]["test_acc"]]
    df.loc[name] = row
    # df.loc[name, "test_loss_L2"] = res_dict_2[name_2]["test_loss"]
```

[83]:

```
df
```

[83]:

	test_loss_L2	test_acc_L2	test_loss_Dropout	test_acc_Dropout
Adagrad	1.347257	0.526	1.556254	0.5156
RMSprop	2.165739	0.4682	1.615707	0.4878
RMSprop+Nesterov	2.149377	0.1969	2.303044	0.1
Adadelta	1.584091	0.4507	1.730869	0.4573
Adam	1.856901	0.3454	1.613318	0.4764

1.4.1 Comment

Adagrad L2 have the best test accuracy. Generally, if we consider the test accuracy, the Adagrad+L2/Dropout method is better than the other methods.