```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
import numpy as np
import matplotlib.pyplot as plt
import torch.optim as optim
torch.set_grad_enabled(True)
```

```
<torch.autograd.grad_mode.set_grad_enabled at 0x7fd94c1e5470>
```

```
def get_num_correct(preds, labels):
  return preds.argmax(dim=1).eq(labels).sum().item()


class Network(nn.Module):
  def __init__(self):
    super(Network, self).__init__()
    self.conv1 = nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5)
    self.conv2 = nn.Conv2d(in_channels=6, out_channels=12, kernel_size=5)

    self.fc1 = nn.Linear(in_features=12*4*4, out_features=120)
    self.fc2 = nn.Linear(in_features=120, out_features=60)
    self.out = nn.Linear(in_features=60, out_features=10)

  def forward(self, t):
    # (1) Input layer
    t=t

    # (2) Hidden conv layer
    t=self.conv1(t)
    t=F.relu(t)
    t=F.max_pool2d(t, kernel_size=2, stride=2)

    # (3) Hidden conv layer
    t=self.conv2(t)
    t=F.relu(t)
    t=F.max_pool2d(t, kernel_size=2, stride=2)

    # (4) Hidden Linear layer
    t=t.reshape(-1, 12*4*4)
    t=self.fc1(t)
    t=F.relu(t)

    # (5) Hidden Linear layer
    t=self.fc2(t)
    t=F.relu(t)

    # (6) Output layer
    t=self.out(t)
```

```
    return t

train_set = torchvision.datasets.FashionMNIST(root='./data/FashionMNIST',
                                              train=True,
                                              download=True,
                                              transform=transforms.Compose([transforms.ToT

# train_loader = torch.utils.data.DataLoader(train_set, batch_size=100, shuffle=True)
```

Starting with TensorBoard

```
!pip uninstall -q tensorboard tb-nightly
```

```
Proceed (y/n)? y
WARNING: Skipping tb-nightly as it is not installed.
```

```
import tensorflow as tf
print(tf.__version__)
```

```
2.0.0
```

```
# !pip install -q tb-nightly
!pip install tensorflow==2.0.0

!pip install tensorboard==2.0.0

%load_ext tensorboard

from torch.utils.tensorboard import SummaryWriter

from itertools import product

parameters = dict(
    lr = [.01]
    ,batch_size = [10, 1000]
    ,shuffle = [True]
)

param_values = [v for v in parameters.values()]
param_values

for lr, batch_size, shuffle in product(*param_values):
  comment = f' batch_size={batch_size} lr={lr} shuffle={shuffle}'

  network = Network()
  train_loader = torch.utils.data.DataLoader(train_set, batch_size=batch_size)
  optimizer = optim.Adam(network.parameters(), lr=lr)

  images, labels = next(iter(train_loader))
  grid = torchvision.utils.make_grid(images)
```

```
comment = f'batch_size={batch_size} lr={lr}'
tb = SummaryWriter(comment=comment)

tb = SummaryWriter()
tb.add_image('images', grid)
tb.add_graph(network, images)

for epoch in range(10):

  total_loss = 0
  total_correct = 0

  for batch in train_loader:
    images, labels = batch

    preds = network(images)
    loss = F.cross_entropy(preds, labels) # Calculate loss

    optimizer.zero_grad()
    loss.backward() #Calculate gradients
    optimizer.step() #Update weights

    total_loss += loss.item() * batch_size
    total_correct += get_num_correct(preds, labels)

  tb.add_scalar('Loss', total_loss, epoch)
  tb.add_scalar('No of correct', total_correct, epoch)
  tb.add_scalar('Accuracy', total_correct/len(train_set), epoch)

  # tb.add_histogram('conv1.bias', network.conv1.bias, epoch)
  # tb.add_histogram('conv1.weight', network.conv1.weight, epoch)
  # tb.add_histogram('conv1.weight.grad', network.conv1.weight.grad, epoch)

  for name, weight in network.named_parameters():
    tb.add_histogram(name, weight, epoch)
    tb.add_histogram(f'{name}.grad', weight.grad, epoch)


  print('epoch:', epoch, 'total_correct:', total_correct, 'loss:', total_loss)

tb.close()
```

```
epoch: 0 total_correct: 45384 loss: 39227.8400269011
epoch: 1 total_correct: 47717 loss: 33542.272071147454
epoch: 2 total_correct: 48370 loss: 32023.40586784412
epoch: 3 total_correct: 48665 loss: 31716.696253316477
epoch: 4 total_correct: 48749 loss: 31722.163917105645
epoch: 5 total_correct: 48935 loss: 31339.337306956295
epoch: 6 total_correct: 48990 loss: 31286.740586287342
epoch: 7 total_correct: 49130 loss: 30803.673764311243
epoch: 8 total_correct: 49019 loss: 30815.12336960528
epoch: 9 total_correct: 49072 loss: 31457.84654219402
epoch: 0 total_correct: 37106 loss: 60704.16992902756
epoch: 1 total_correct: 47450 loss: 32053.430527448654
epoch: 2 total_correct: 50004 loss: 26692.74941086769
epoch: 3 total_correct: 51300 loss: 23701.770544052124
epoch: 4 total_correct: 51838 loss: 22166.13867878914
epoch: 5 total_correct: 52276 loss: 20823.42180609703
epoch: 6 total_correct: 52737 loss: 19517.51658320427
epoch: 7 total_correct: 52982 loss: 18828.92218232155
epoch: 8 total_correct: 53183 loss: 18164.218455553055
epoch: 9 total_correct: 53409 loss: 17676.67293548584
```

```
tensorboard --logdir=runs
```

```
Reusing TensorBoard on port 6006 (pid 409), started 0:00:20 ago. (Use '!kill 409' to k
```

TensorBoard SCALARS IMAGES GRAPHS INACTIVE

Show data download links

Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE

WALL

Runs

Write a regex to filter runs

Jan03_08-05-17_595b89f4ac18batch_size=10 lr=0.01

Jan03_08-05-18_595b89f4ac18

Jan03_08-09-33_595b89f4ac18batch_size=10 lr=0.01

Jan03_08-09-33_595b89f4ac18

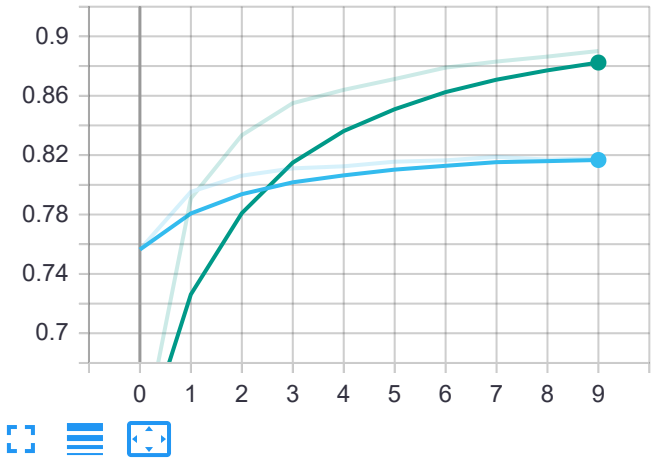
Jan03_08-14-27_595b89f4ac18batch_size=1000 lr=0.

TOGGLE ALL RUNS

runs

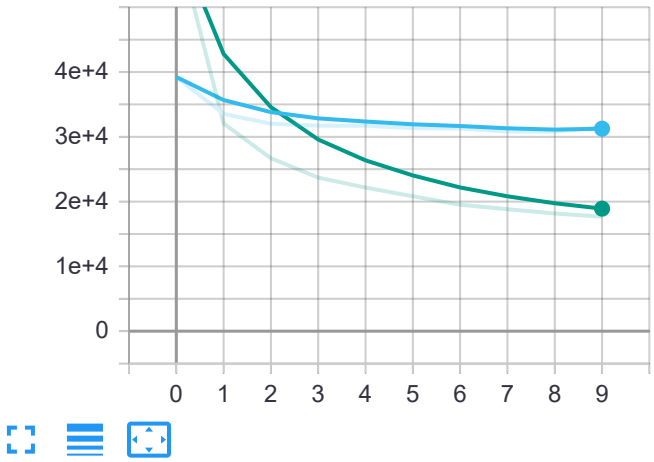
Accuracy

Accuracy



Loss

Loss



No_of_correct