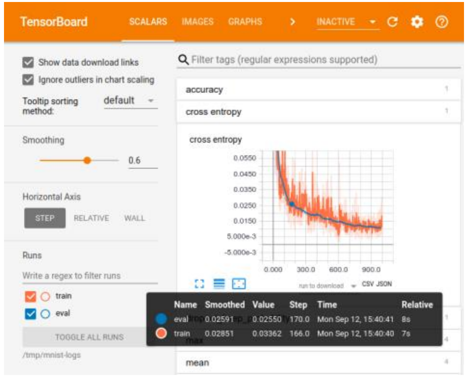
Introduction to Deep Learning (I2DL)

Exercise 7: Pytorch

Overview

- Exercise 6
  - Leaderboard
  - Case Study
- Deep Learning Frameworks
  - Static vs Dynamic
- Exercise 7
  - Pytorch, Tensorboard, Pytorch Lightning
- Outlook and Exam



Exercise 6 – Leaderboard

Exercise 1 | Exercise 3 | Exercise 4 | Exercise 5 | Exercise 6 | Exercise 7 | Exercise 8 | Exercise 9 | Exercise 10 | Exercise 11

#	User	Score
1	u1051	62.88
2	u1048	59.08
3	u1785	57.45
4	u1120	56.75
5	u0111	56.59
6	u1005	56.32

CIFAR10 - Leaderboard

RANK	MODEL	PERCENTAGE CORRECT	PERCENTAGE ERROR	FLOPS	PARAMS	ECE	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	EffNet-L2 (SAM)	99.70	0.30				✓	Sharpness-Aware Minimization for Efficiently Improving Generalization			2020
2	ViT-H/14	99.5	0.5		632M		✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020
3	ViT-L/16	99.42	0.58		307M		✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020
4	BiT-L (ResNet)	99.37	0.63				✓	Big Transfer (BiT): General Visual Representation Learning			2019
5	LaNet	99.03	0.97		45		×	Sample-Efficient Neural Architecture Search by Learning Action Space for Monte Carlo Tree Search			2019

Case Study: Power of Transformations

In [5]:
```python
import random

class SaltPepper:
    def __init__(self, grains=64, prob=0.5):
        self.grains = grains
        self.prob = prob

    def __call__(self, image):
        if random.uniform(0,1) < self.prob:
            rows = np.random.randint(0, image.shape[0], size=self.grains)
            cols = np.random.randint(0, image.shape[1], size=self.grains)

            g = self.grains // 2
            image[rows[:g], cols[:g]] = 255
            image[rows[g:], cols[g:]] = 0

        return image
```

In [6]:
```python
#Retrieve an image from the dataset and flip it
image = dataset[1890]['image']
transform = SaltPepper(64,1.0)
image_flipped = transform(image)

#Show the two images
plt.figure(figsize = (2,2))
plt.subplot(1, 2, 1)
plt.imshow(image.astype('uint8'))
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(image_flipped.astype('uint8'))
plt.axis('off')
plt.title("Left: Original Image, Right: Flipped image")
plt.show()
```

Left: Original Image, Right: Flipped image

In [7]:
```python
class RandomRoll:
    def __init__(self, dx=(-8,8), dy=(-8,8), prob=0.5):
        self.dx = dx
        self.dy = dy
        self.prob = prob

    def __call__(self, image):
        if random.uniform(0,1) < self.prob:
            x = random.randint(self.dx[0], self.dx[1])
            y = random.randint(self.dy[0], self.dy[1])
            return np.roll(image, (x, y), axis=(0,1))
        return image
```

In [8]:
```python
#Retrieve an image from the dataset and flip it
image = dataset[1]['image']
transform = RandomRoll((-8,8), (-8,8), 1.0)
image_flipped = transform(image)

#Show the two images
plt.figure(figsize = (2,2))
plt.subplot(1, 2, 1)
plt.imshow(image.astype('uint8'))
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(image_flipped.astype('uint8'))
plt.axis('off')
plt.title("Left: Original Image, Right: Flipped image")
plt.show()
```

Left: Original Image, Right: Flipped image

Case Study: Power of Transformations

In [9]:
```
class RandomSpeckle:
    def __init__(self, std=0.1, prob=0.5):
        self.std = std
        self.prob = prob

    def __call__(self, image):
        if random.uniform(0,1) < self.prob:
            row,col,ch = image.shape
            gauss = np.random.randn(row,col,ch)
            gauss = gauss.reshape(row,col,ch)
            noisy = image + image * random.uniform(0.0, self.std) * gauss
            return noisy

        return image
```

In [10]:
```
#Retrieve an image from the dataset and flip it
image = dataset[1000]['image']
transform = RandomSpeckle(prob=1.0)
image_flipped = transform(image)

#Show the two images
plt.figure(figsize = (2,2))
plt.subplot(1, 2, 1)
plt.imshow(image.astype('uint8'))
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(image_flipped.astype('uint8'))
plt.axis('off')
plt.title("Left: Original Image, Right: Flipped image")
plt.show()
```

Left: Original Image, Right: Flipped image

Final Training with parameters:
- num_layers
- hidden_size
- Regularization
- Learning rate

In [18]:
```
trafo = ComposeTransform([
    rescale_transform,
    normalize_transform,
    RandomHorizontalFlip(),
    RandomSpeckle(prob=0.4),
    SaltPepper(prob=0.4),
    RandomRoll(prob=0.4),
    flatten_transform])

traf_dataset = ImageFolderDataset(
    mode='train',
    root=cifar_root,
    download_url=download_url,
    transform=trafo
)
data = DataLoader(
    dataset=traf_dataset,
    batch_size=512,
    shuffle=True,
    drop_last=True
)
```

In [19]:
```
best_model = ClassificationNet(activation=Relu(), num_layer=3, hidden_size=258, reg=6.357358814407898e-06)
solver = Solver(best_model, data, dataloaders['val'], learning_rate=5.535943214549563e-05)
solver.train(epochs=500, patience=21)
```

Limiting Factors

- Computational Power
- Specialized Architectures
  - CNN
- Proper Initialization

RANK	MODEL	PERCENTAGE CORRECT	PERCENTAGE ERROR	FLOPS	PARAMS	ECE	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	EffNet-L2 (SAM)	99.70	0.30				✓	Sharpness-Aware Minimization for Efficiently Improving Generalization			2020
2	ViT-H/14	99.5	0.5		632M		✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020
3	ViT-L/16	99.42	0.58		307M		✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020
4	BiT-L (ResNet)	99.37	0.63				✓	Big Transfer (BiT): General Visual Representation Learning			2019
5	LaNet	99.03	0.97		45		×	Sample-Efficient Neural Architecture Search by Learning Action Space for Monte Carlo Tree Search			2019

Deep Learning Frameworks

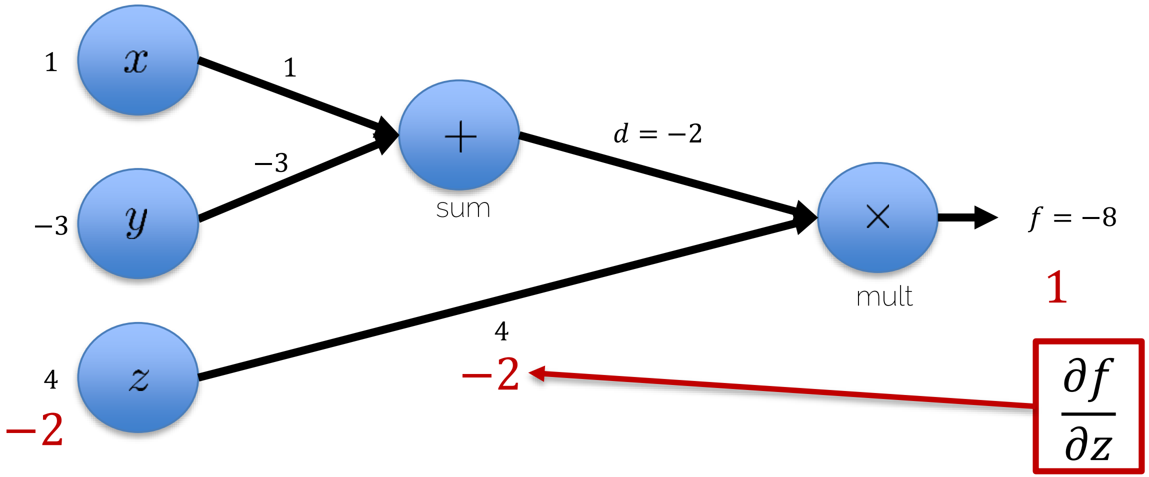
The two big ones

- Tensorflow - Google
  - As well as Keras
- Pytorch – Facebook

Other examples

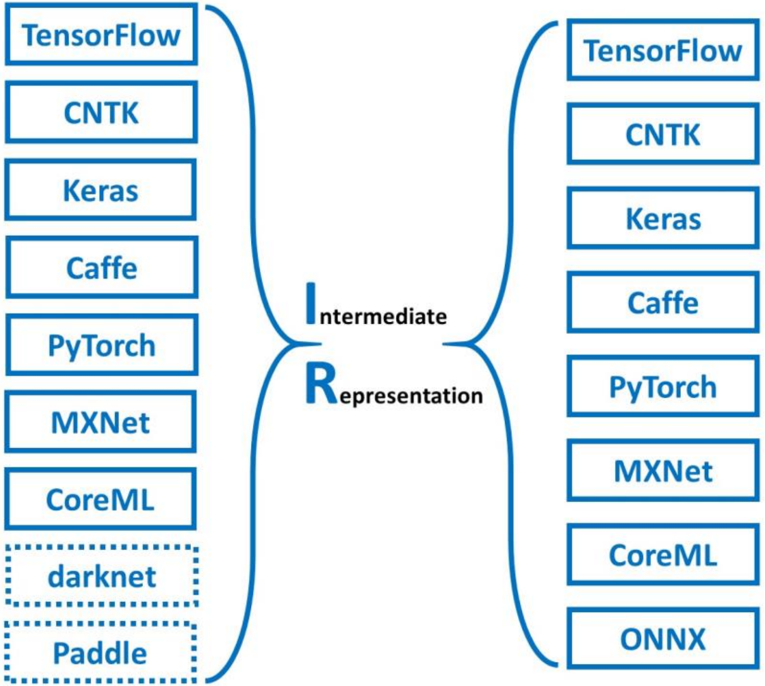
- CNTK – Microsoft
- Mxnet – Apache
- Jax - Google

Handling of Computation Graph



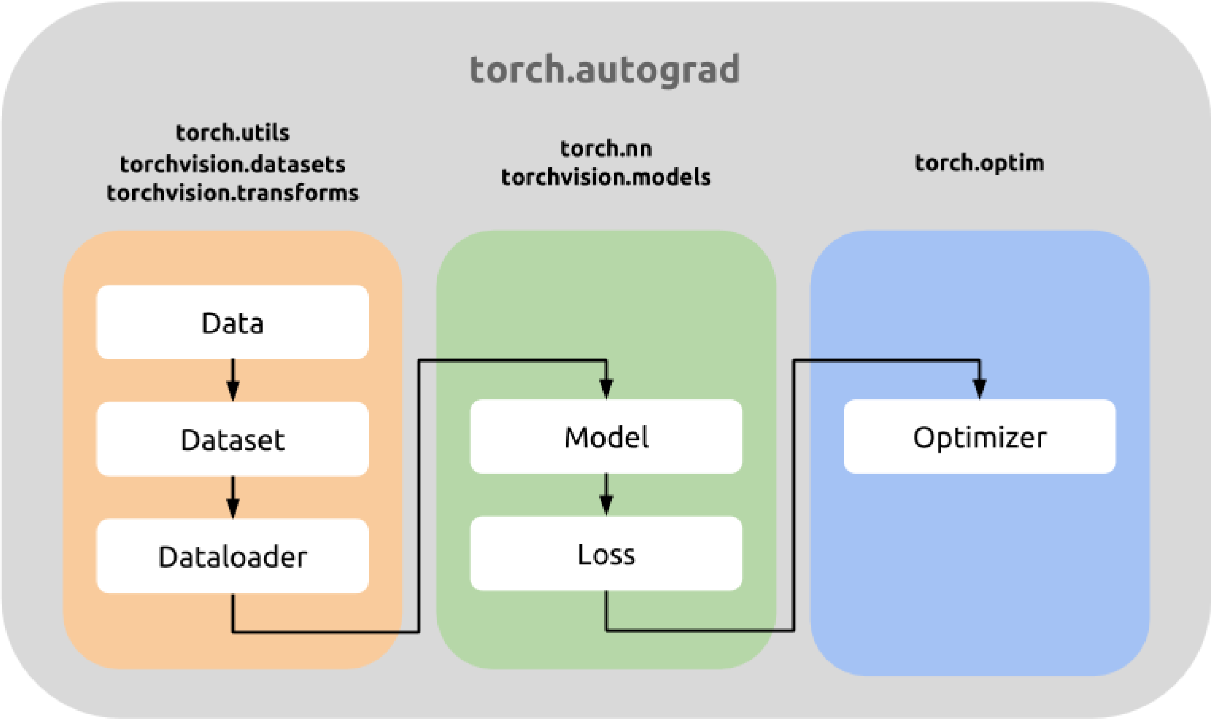
	Tensorflow	Pytorch
Graph Creation	Static/Eager	Dynamic/On Runtime
Similar to	C	Python

Framework Conversion



See: https://github.com/microsoft/MMdnn

Pytorch: Overview



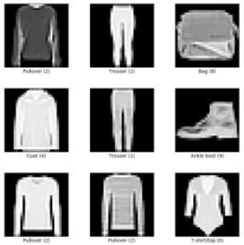
Easy Device Assignment

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)

print(f"Original device: {x.device}") # "cpu", integer

tensor = x.to(device)
print(f"Current device: {x.device}") #"cpu" or "cuda", double
```

```
cpu
Original device: cpu
Current device: cpu
```

Datasets: Torchvision

- Torchvision
  - torchvision.datasets contains many datasets, such as ImageNet, FashionMNIST, etc.



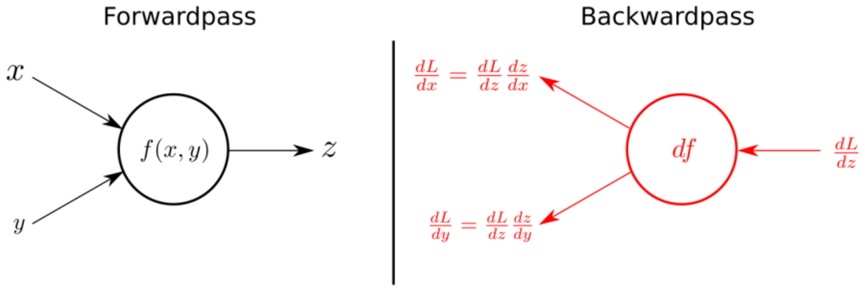
```
fashion_mnist_dataset = torchvision.datasets.FashionMNIST(root='../datasets',
                                                           train=True,
                                                           download=True,
                                                           transform=transform)

fashion_mnist_dataloader = DataLoader(fashion_mnist_dataset, batch_size=8)
```

Source: https://www.tensorflow.org/datasets/catalog/fashion_mnist

Easy Network Creation

```
import torch.nn as nn
# defining the model
class Net(nn.Module):
    def __init__(self, input_size=1*28*28, output_size=100):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(input_size, output_size)

    def forward(self, x):
        x = self.fc1(x)
        return x
net = Net()
net = net.to(device)
```

Where is the backward pass?



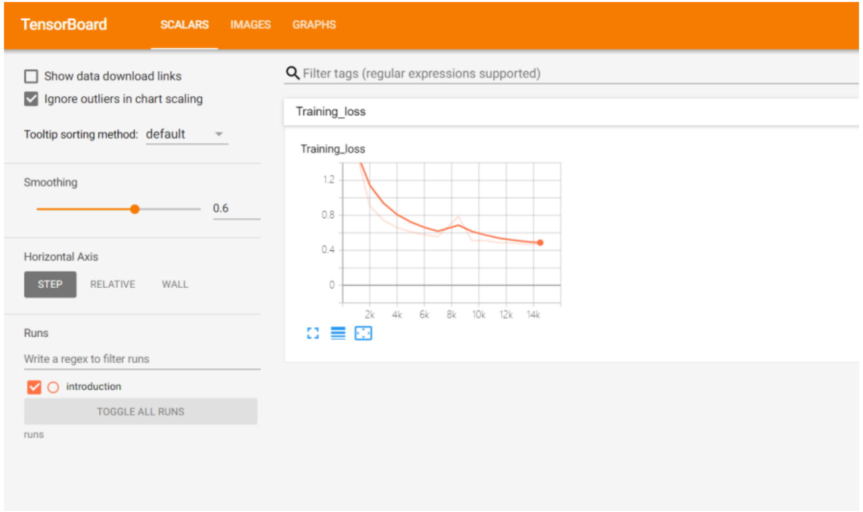
References on Pytorch

- Repository: https://github.com/pytorch/pytorch
- Examples (very nice): https://github.com/pytorch/examples
- PyTorch for NumPy users: https://github.com/wkentaro/pytorch-for-numpy-users

Tensorboard: Simple Logging

- Directly access tensorboard in your training loop

- Tensorboard generates the graph/timestamps etc. for you

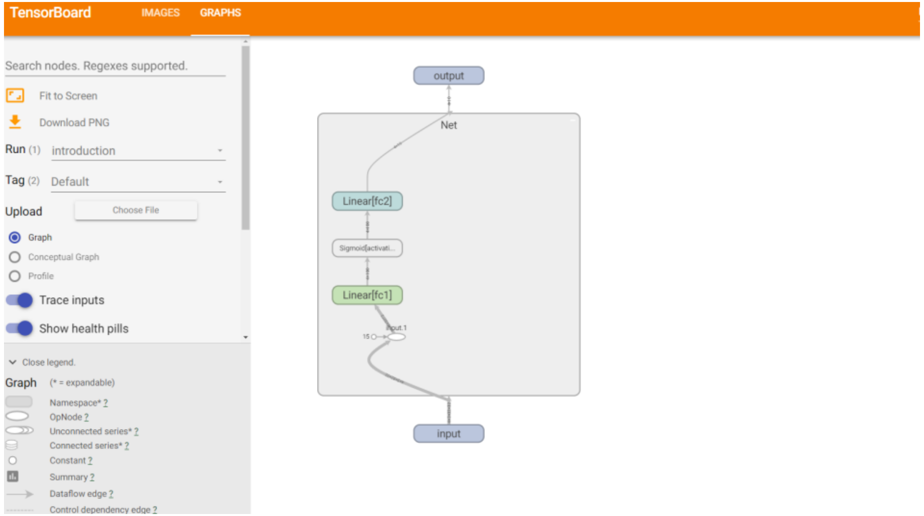


Tensorboard: Visualize Networks

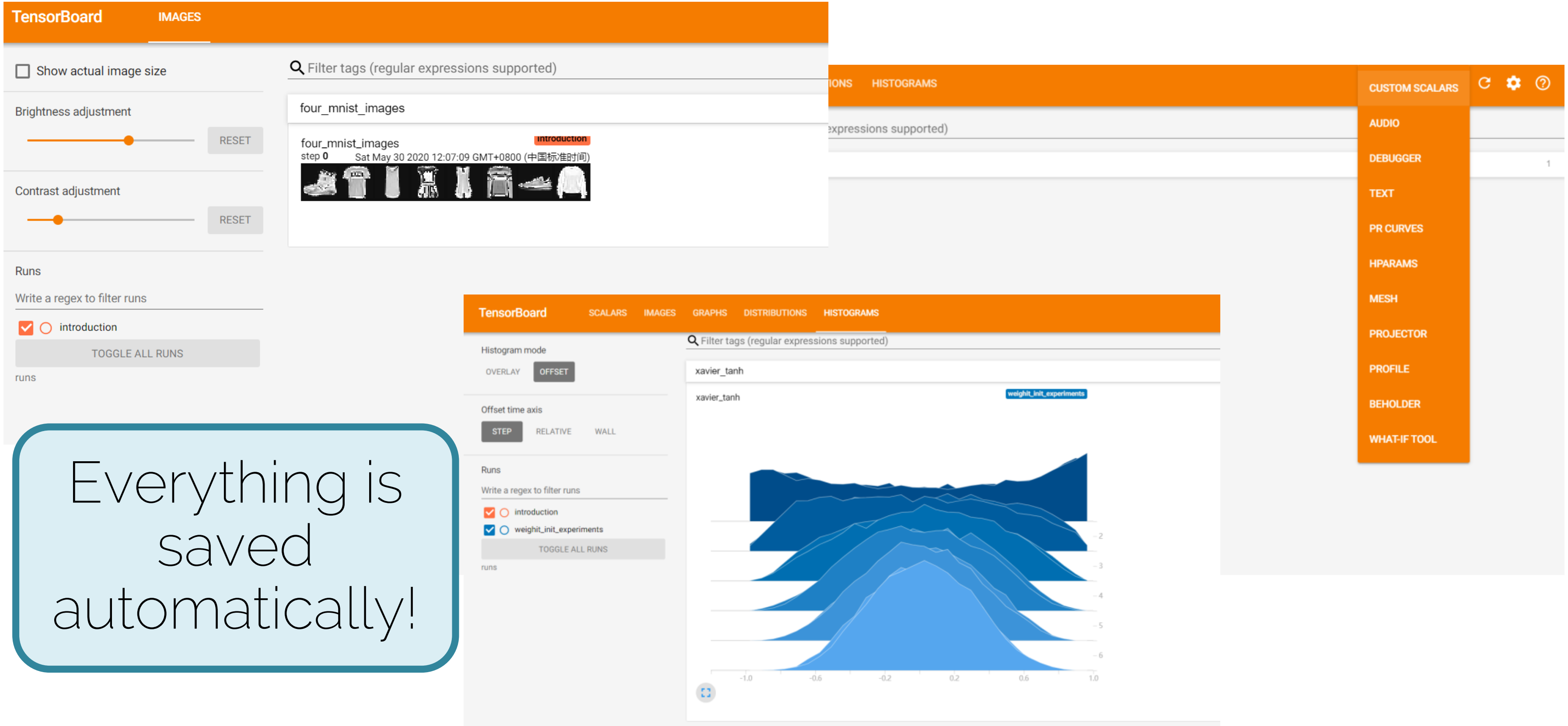
- Using a single forward pass, tensorflow can map and display your network graph

```python
# Initialize your model
net = Net()

# Visualize its architecture in TensorBoard
writer.add_graph(net, images)
```

Graph creation needs network & one batch!



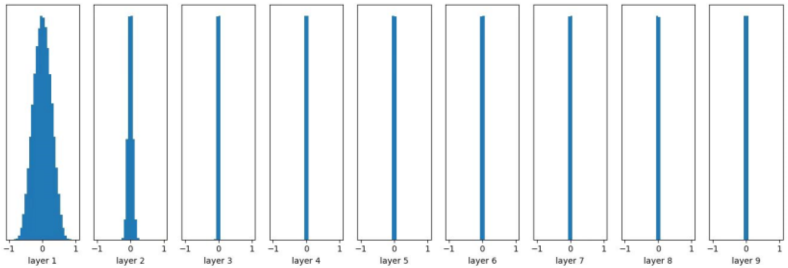
Document Everything!

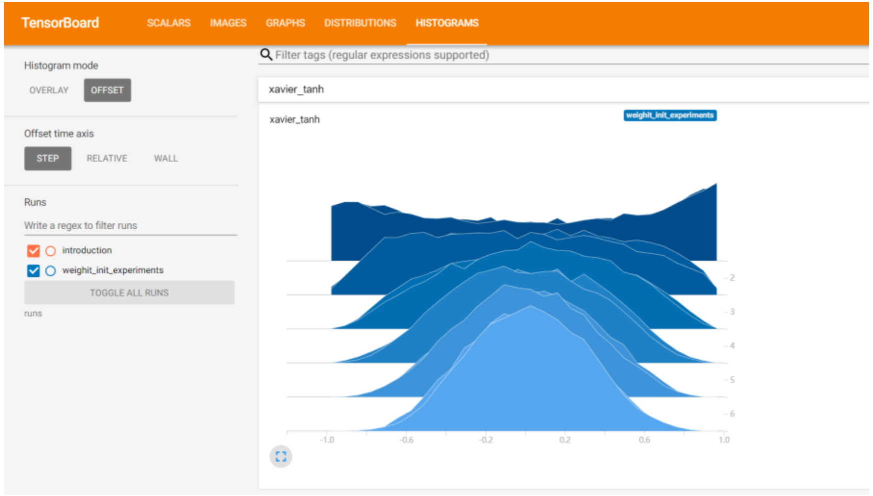


Everything is saved automatically!

Weight Initialization

- Histogram visualization for layer outputs can show off effects of weight initialization as shown in the lecture





Pytorch Lightning

Idea Behind PyTorch Lightning

Classify our code into three categories

1. **Research** code (the exciting part!, changes with new tasks, models etc.)

   → LightningModule

2. **Engineering** code (the same for all projects and models)

   → Trainer

3. **Non-essential** code (logging, organizing runs)

   → Callbacks

Lightning Module

PyTorch

```
# model
class Net(nn.Module):
  def __init__(self):
      self.layer_1 = torch.nn.Linear(28 * 28, 128)
      self.layer_2 = torch.nn.Linear(128, 10)

  def forward(self, x):
    x = x.view(x.size(0), -1)
    x = self.layer_1(x)
    x = F.relu(x)
    x = self.layer_2(x)
    return x

# train loader
mnist_train = MNIST(os.getcwd(), train=True, download=True,
                    transform=transforms.ToTensor())
mnist_train = DataLoader(mnist_train, batch_size=64)

net = Net()

# optimizer + scheduler
optimizer = torch.optim.Adam(net.parameters(), lr=1e-3)
scheduler = StepLR(optimizer, step_size=1)

# train
for epoch in range(1, 100):
   model.train()
   for batch_idx, (data, target) in enumerate(train_loader):
      data, target = data.to(device), target.to(device)
      optimizer.zero_grad()
      output = model(data)
      loss = F.nll_loss(output, target)

      loss.backward()
      optimizer.step()
      if batch_idx % args.log_interval == 0:
          print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
              epoch, batch_idx * len(data), len(train_loader.dataset),
              100. * batch_idx / len(train_loader), loss.item()))
```

Methods that need to be implemented
- __init__
- forward
- training_step
- configure_optimizers

Optional methods
- validation_step
- validation_end
- Data handling

Trainer

PyTorch

```python
# model
class Net(nn.Module):
    def __init__(self):
        self.layer_1 = torch.nn.Linear(28 * 28, 128)
        self.layer_2 = torch.nn.Linear(128, 10)

    def forward(self, x):
      x = x.view(x.size(0), -1)
      x = self.layer_1(x)
      x = F.relu(x)
      x = self.layer_2(x)
      return x

# train loader
mnist_train = MNIST(os.getcwd(), train=True, download=True,
                    transform=transforms.ToTensor())
mnist_train = DataLoader(mnist_train, batch_size=64)

net = Net()

# optimizer + scheduler
optimizer = torch.optim.Adam(net.parameters(), lr=1e-3)
scheduler = StepLR(optimizer, step_size=1)

# train
for epoch in range(1, 100):
   model.train()
   for batch_idx, (data, target) in enumerate(train_loader):
      data, target = data.to(device), target.to(device)
      optimizer.zero_grad()
      output = model(data)
      loss = F.nll_loss(output, target)
      loss.backward()
      optimizer.step()
      if batch_idx % args.log_interval == 0:
          print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
              epoch, batch_idx * len(data), len(train_loader.dataset),
              100. * batch_idx / len(train_loader), loss.item()))
```

PyTorch Lightning

```python
# model
class Net(LightningModule):
    def __init__(self):
        self.layer_1 = torch.nn.Linear(28 * 28, 128)
        self.layer_2 = torch.nn.Linear(128, 10)

    def forward(self, x):
      x = x.view(x.size(0), -1)
      x = self.layer_1(x)
      x = F.relu(x)
      x = self.layer_2(x)
      return x

  def train_dataloader(self):
    mnist_train = MNIST(os.getcwd(), train=True, download=True,
                        transform=transforms.ToTensor())
    return DataLoader(mnist_train, batch_size=64)

  def configure_optimizers(self):
    optimizer = torch.optim.Adam(self.parameters(), lr=1e-3)
    scheduler = StepLR(optimizer, step_size=1)
    return optimizer, scheduler

  def training_step(self, batch, batch_idx):
    data, target = batch
    output = self.forward(data)
    loss = F.nll_loss(output, target)
    return loss

if __name__ == '__main__':
  net = Net()

  trainer = Trainer()
  trainer.fit(net)
```

1. Initialize the model with hyperparamers for training (e.g. as a dictionary)
2. Trainer contains all code relevant for training our neural networks
3. Call the method .fit() for training the network

That's all you need to train you model ☺

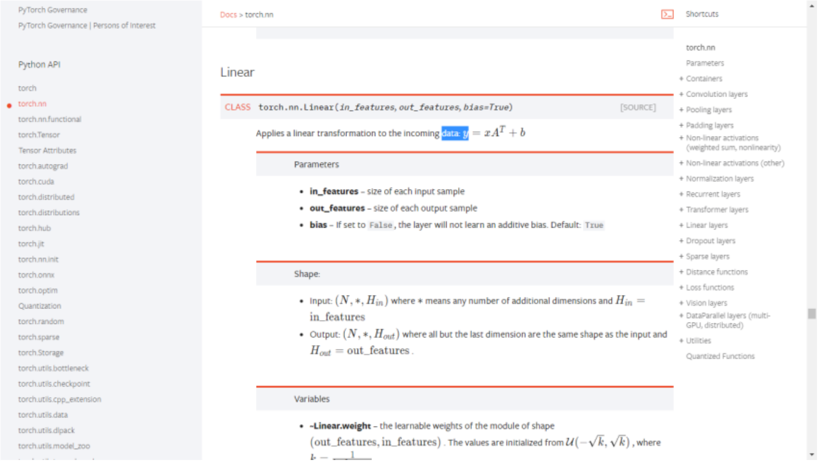
Pros/Cons

- Advantages
  - Better overview of the relevant code
  - Nice debugging features
  - Many automated options, like logging

- Potential Problems
  - Can have issues like any stock library
  - Not always straightforward to add features yourself

Your tasks

- Go over all notebooks
  - There is no submission or code to implement
- Experiment yourself!
- Check out documentations



Optional Submission: CIFAR10 (again)

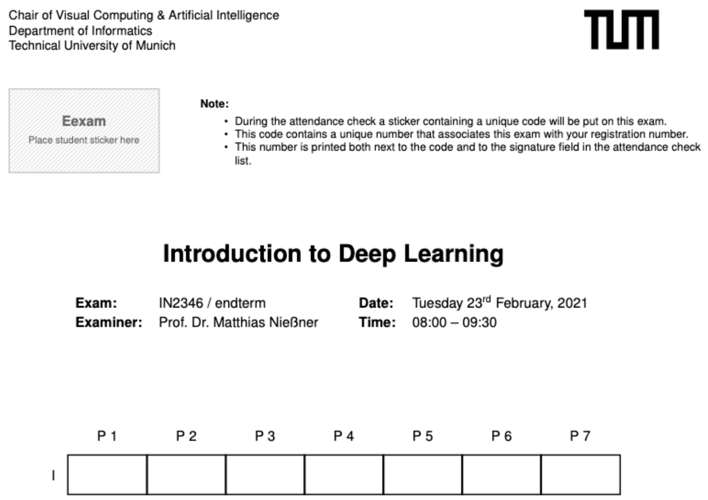
- We will open an optional leaderboard
  - TBD

- Rules (not enforced):
  - Use the previous Cifar10 dataset class in pytorch
  - No convolutional layers

- Use Piazza, especially if you have a successful approach, do a quick write up and discuss

Exam

- Exam date
  - 10.02.22, 10:00
- Exam Format
  - Online
- Mock Exam
  - Uploaded



Chair of Visual Computing & Artificial Intelligence
Department of Informatics
Technical University of Munich

Eexam
Place student sticker here

Note:
- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Introduction to Deep Learning

Exam: IN2346 / endterm **Date:** Tuesday 23rd February, 2021
Examiner: Prof. Dr. Matthias Nießner **Time:** 08:00 – 09:30

	P 1	P 2	P 3	P 4	P 5	P 6	P 7
I							

Summary

- **Monday 6.12**: Watch Lecture 8
  - Training NNs 3
- **No exercise submission**
- **Thursday 9.12**: Tutorial 8
  - Autoencoder