

# Introduction to Deep Learning (I2DL)

Exercise 7: Pytorch

## Today's Outline

- Exercise 6 Recap

   Or: why are you bad?
- Pytorch
  And other libraries
- "Optional Submission"
  It's good for you <3</li>
- Organization









## Exercise 6 Recap

#### Our Leaderboard

#	User	Score
1	u0533	59.16
2	a0012	58.19
3	u1249	57.72
4	u0736	56.43
5	u1289	56.40
6	u0438	56.29
7	u1566	56.16
8	u0871	56.14
9	u0036	56.00
10	u1180	55.93

#### Image Classification on CIFAR-10



https://paperswithcode.com/sota/image-classification-on-cifar-10

#### Some Limiting Factors

• Computational power and/or time

Pytorch -> GPU support

• Specialized architectures



• More knowledge e.g., proper initialization



#### Lecture Recap: Initialization

#### Lecture

- Network weights shouldn't only be randomly initialized
- They should be tailored to our activation function



#### We in exercise 6 ^^



# Pytorch

#### **Exercise Overview**

Exercise 01: Organization Exercise 02: Math Recap	Intro
Exercise 03: Dataset and Dataloader Exercise 04: Solver and Linear Regression Exercise 05: Neural Networks Exercise 06: Hyperparameter Tuning	Numpy (Reinvent the wheel)
Exercise 07: Introduction to Pytorch Exercise 08: Autoencoder	Pytorch/Tensorboard

## Deep Learning Frameworks

#### The two big ones

- Tensorflow Google
  As well as Keras
- Pytorch Facebook

#### Other examples

- CNTK Microsoft
- Mxnet Apache
- Jax Google



**O** PyTorch



#### Different Paradigms



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

#### Framework Conversion

Usual workflow:

- Develop and train network in your favourite framework
- Convert and optimize in target framework for production



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

#### Pytorch: Overview



## Some key features

#### • Simple device management

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

- Implementations of:
  - Optimizers, etc.
  - Datasets
  - Automatic gradients



airplane	
automobile	
bird	
cat	
deer	
dog	

									-
mil		-	×	*	-	1	-		-
		-		-		-			-
N	J.	2	2	1	4	1	1	2	W.
1		1	2		٩Ŀ	2	Z,	No.	4
15	40	<u> </u>	-		¥	Y	1	<b>.</b>	
ş٩.	1.	P	٩.	1			R <sup>a</sup>	4	N.

#### Easy network creation



Where is the backward pass?

 $\frac{dL}{dz}$ 

## References on Pytorch

- Repository: <a href="https://github.com/pytorch/pytorch">https://github.com/pytorch/pytorch</a>
- Examples (recommendation):
   <u>https://github.com/pytorch/examples</u>
- PyTorch for NumPy users: <u>https://github.com/wkentaro/pytorch-for-numpy-users</u>
- Look up your own and share! 😳

#### Tensorboard (also in Pytorch)

• Directly access tensorboard in your training loop

• Tensorboard generates the graph/timestamps etc. for you

TensorBoard SCALARS IMAGES	
Show data download links Ignore outliers in chart scaling Tooltip sorting method: default ~	Q Filter tags (regular expressions supported) Tsaining_loss Tsaining_loss
Smoothing 0.6	12 0.8
Horizontal Axis STEP RELATIVE WALL	
Runs	
Write a regex to filter runs	
C introduction	
TOGGLE ALL RUNS	
nins	

#### Visualize Networks

• Using a single forward pass, tensorboard can map and display your network graph

Graph creation needs network & one batch!

Tenso	orBoard MAGES	GRAPHS
Search	nodes. Repexes supported.	
-	Te to Screen	
± :	Download PND	
Run (1)	Introduction	10
Tag (7)	Default	
Upload	Chanted File	
	en onghuat brash The	
	frace inputs	
-	Show health pills	
+ Oin	ingent	
Graph	(* - expandate)	
308	Normespace*1 Openate 2 Openate 2 Openated server*2 Openated server*2 Openated 1	
	Commery 2	
	Datafine edge 2 Cartosi digendering edge 2	

#### In short: document everything!



## Example: Weight Initialization

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





## More Abstraction: Pytorch Lightning

Classify our code into three categories

- Research code (the exciting part!, changes with new tasks, models etc.) → LightningModule
- 1. Engineering code (the same for all projects and models)

→ Trainer

1. Non-essential code (logging, organizing runs)

→ Callbacks

## Lightning Module

#### PyTorch

* 1727978.A.	
class Net(nn.Module);	
definit(self):	
self.layer_1 = torch.nn.Linear(28 + 28, 128)	
self.layer_2 = torch.nn.Linear(128, 10)	
def forwardiself, x):	
x = x.view(x.size(0), -1)	
<pre>x = self.layer_l(x)</pre>	
$x = F_{relu}(x)$	
<pre>x = self.layer_2(x)</pre>	
return x	
# train loader	
mnist_train = PWIST(os.getcwd(), train=True, download=True,	
transform=transforms.ToTensor())	
mnist_train = OutsLooder(mnist_train, batch_size=64)	
# optimizer + scheduler optimizer = torch.optim.Adam(net.psrameters(), lr=1e-3) scheduler = SteptB(optimizer, step_sizer1)	
# optimizer + scheduler optimizer = torch.optim.Adam(net.parameters(), lr=1e-3) scheduler = StepLB(optimizer, step_size=1) # train	]
<pre># optimizer + scheduler optimizer = torch.optim.Adam(net.psrameters(), 1r=1e-3) scheduler = Step(#[optimizer, step_size=1] # train for each in range(1, 100);</pre>	]
<pre># optimizer + scheduler optimizer + torch.optim.Adam(net.parameters(), lr=l=-3) scheduler = Step(B(optimizer, step_size=1) # train for epoch in range[1, 100); model.train()</pre>	][
<pre># optimizer + scheduler optimizer = torch.optim.Adam(net.parameters(), lr=l=-3) scheduler = SteptR(optimizer, step_sizer1) # train for epoch in range(1, 100); model.train() for batch_dax, (data, target) in enumeratmitrain_imoder);</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adam(net.parameters(), [r=1=-3) scheduler = SteptB(optimizer, step_sizer1) # train for epoch in range[1, 100); model.train() fur batch_ide, (dota, target1 in enumeratm(train_isader); data, target = data.ts(device), target.t6(device)</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adam(met.parameters(), [r=1e-3) scheduler = SteptB(optimizer, step_sizer1) # train for epoch in range[1, 100); model.train() for batch_idx, (data, target) in enumeratm(train_lsader); data, target = data.to(device), target.to(device) optimizer.zerz_grad()</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adsm(net.parameters(), [r=1e-3) scheduler = StepLB(optimizer, step_sizer1) # train for epoch in mange[1, 100); model.train() fur batch_bDs, (Sota, target) in enumeratu(train_imader); data, target = data.ts(device), target.to(device) uptimizer.zero_grad() output = model(data)</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adam(met.parameters(), [r=1=-3]) scheduler = SteptB(cotimizer, step_sizer]) # train fur epoch in range[1, 100]) model.train() fur batch_idx, (data, target) in enumeratm(train_loader)) data, target = data.to(device), target.to(device) mptimizer.gerd_grad() outpute = model(data) loss = f.mlt_loss(output, target)</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adsm(net,parameters(), [r=1e-3) scheduler = SteptB(optimizer, step_sizer1) # train for epoch in range[1, 109); model.train() for batch_bds, (dota, target) in enumeratm(train_imader); data, target = data.ts(device), target.to(device) optimizer.zerm_grad() output = model(dota) loss = f.mlt_loss(output, target) loss.backward()</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adam(net.parameters(), [r=1e-3] scheduler = SteptR(optimizer, step_sizer]) # train for epoch in range[1, 100): model.train[] for batch_ldw, (data, target] in enumeratmitrain_lmoder): data, target = data.to(device), target.to(device) optimizer.rerg.grad() output = model(data) loss = f.mli_loss(output, target) fors.backward() optimizer.step()</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adam(net.parameters(), [r=1e-3] scheduler = SteptR(optimizer, step_sizer1) # train for easch in range[1, 100); model.train[] for batch_ide, (data, target1 in enumerate(train_imsder); data, target = data.to(device), target.to(device) mutimizer.zera_prad() output = model(data) loss = f.=lt_loss(output, target) foss.backmard() aptimizer.step[] if batch ide &amp; arms.log interval == 0;</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adjme(net.parameters(), lr=l=-3) scheduler = SteptB(optimizer, step_sizer1) # train for epoch in rampe(1, 100): model.train() fur batch_idex, (data, target1 in enumeratm(train_imader): data, target = data.to(device), target.to(device) motimizer.zer_most() output = model(data) loss = f.=lt_loss(output, target) toss.backward() mptimizer.step() if tatch_ide &amp; args.log_interval == 0:</pre>	]
<pre># optimizer + scheduler optimizer = torch.optim.Adam(net.parameters(), ir=1=-3) scheduler = SteptB(cotimizer, step_sizer1) # train for epoch in range(1, 100)) model.train() for batch_ids, (data, target) in enumeratm(train_lsader)) data, target = data.to(device), target.to(device) aptimizer.zera.grad() output = model(data) loss = f.mlt_loss(output, target) loss.backward() aptimizer.step() if batch_ids % args.log_interval == 0: print(Train Epoch( () (()/() ((.dr)%))(tissa) (1.0) eooch, batch_ids, ken(data), los(train loader.d)</pre>	]

Methods that need to be implemented

- \_\_init\_\_
- forward
- training\_step
- configure\_optimizers

## Lightning Trainer

#### PyTorch

# model		# sodel
class Net(nn.Module):		class NetfLightning
4.4.3. (1994) a. (1991) 707	and the second second second	
active and charges, 2001		
# train loader		out train metaland
	Section Sector	
net = Not()		def configure_noti
# antipiter - scheduler		
* optimizer * scheduter		
		det tratation steel
# train		data, target - 1
for epoch in range(1, 100):	The second second	
model.train()	1 S	
for batch_ids, (data, target) in enumerate(train_loader):		
data, target = data.toldevice), target.toldevice)		
optimizer.zera_grad()		
	10001	
less, backward()		
optimizer.step()		
if batch idx & args.log interval == 0:		
print('Train Epoch: () (()/() ((:,0f)%))/tLoss: ().(	if)', formati	
epoch, batch_idx + len(data), len(train loader.d	(ataset),	6
100. + batch idx / len(train loader), loss, item	211	

# model	ning the late
out train pat	aloader(self):
det continues	antimizers (salifie
def training	steelself, batch, batch idals
	n - Sanon D, terramitatina S, terramitatina, stargarti
	ifname == *main1 net = Net()
	trainer = Trainer()

trainer, fit(net)

- 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 🕑

#### What to use? Your call!

- 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



# "Optional" Submission

## CIFAR10... Again...

• Task: CIFAR10 classification (but now in Pytorch)



- New:
  - More knowledge from lecture 7
  - Can use everything but no: convolutional layers/transformers/ pre-trained networks
  - Filesize and parameter limit

## So... Tuning again?

- Make sure
  - To get into pytorch (read docs and source code!)
  - To improve upon your previous submission
  - How can you select good hyperparameters?
  - Discuss with fellow students
     -> Code sharing on campuswire allowed!





# Organization

#### Post Deadline Submissions

#### Post Deadline Exercises

The following entries are identical to the ones above, but allow you to submit solutions after the deadline of the respective exercise has passed. Submitting solutions to these "Post Deadline" exercises does not count towards the bonus nor can they substitute a missed exercise!

Exercise 3 (Post Deadline) – Dataset and Dataloader [Optional]	$\sim$
Exercise 4 (Post Deadline) – Solver and Linear Regression [Optional]	$\sim$
Exercise 5 (Post Deadline) – Neural Networks [Optional]	$\sim$
Exercise 6 (Post Deadline) – Hyperparameter Tuning [Optional]	$\sim$
Exercise 7 (Post Deadline) – Intro to Pytorch [Optional]	$\sim$
Exercise 8 (Post Deadline) – Autoencoder [Optional]	$\sim$
Exercise 9 (Post Deadline) – Convolutional Neural Networks [Optional]	$\sim$
Exercise 10 (Post Deadline) – Semantic Segmentation [Optional]	$\sim$
Exercise 11 (Post Deadline) – Recurrent Neural Networks [Optional]	$\sim$

## Solutions on Github

- Please don't upload solutions
  - You only hurt future students progression
  - We will issue take-downs!
  - No future employer cares



- What can you do else?
  - Choose an exercise of 7, 9, 10, or any other task/paper
  - Document your whole journey
  - Create: visualizations, ablations
  - Outline: key changes, maybe a story
  - Share: your documents with students



# See you next week