

STAT 453: Introduction to Deep Learning and Generative Models

Ben Lengerich

Lecture 06: Automatic Differentiation with PyTorch; Project Discussion

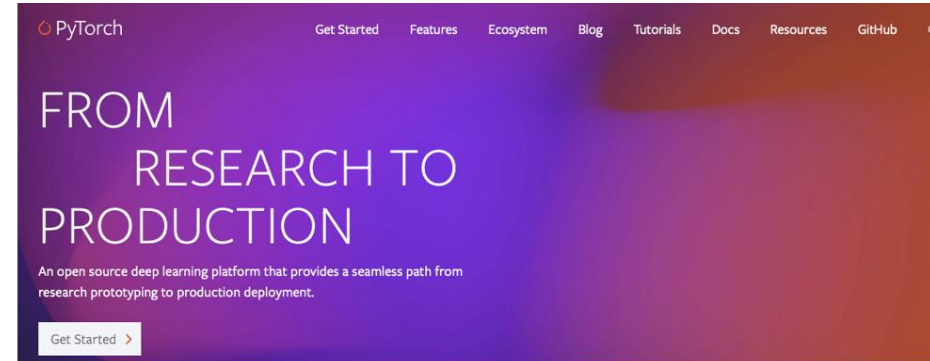
September 22, 2026





Today: Automatic differentiation; projects

- **Automatic Differentiation in PyTorch**
- A Closer Look at the PyTorch API
- Project Discussion



<https://pytorch.org/>

At a Glance:

- Based on Torch 7, which was based on Lua and inspired by Lush
- PyTorch started in 2016
- Focuses on flexibility and minimizing cognitive overhead
- Dynamic nature of autograd API inspired by Chainer
- Core features
 - **Automatic differentiation**
 - **Dynamic computation graphs**
 - NumPy integration
- written in C++ and CUDA (CUDA is like C++ for the GPU)
- Python is the usability glue

Installing PyTorch

Recommendation for Laptop (e.g., MacBook)

PyTorch Build	Stable (1.7.1)				Preview (Nightly)
Your OS	Linux		Mac		Windows
Package	Conda		Pip	LibTorch	Source
Language	Python			C++ / Java	
CUDA	9.2	10.1	10.2	11.0	None
Run this Command:	NOTE: Python 3.9 users will need to add '-c=conda-forge' for installation <code>conda install pytorch torchvision torchaudio -c pytorch</code>				

Recommendation for Desktop (Linux) with GPU

PyTorch Build	Stable (1.7.1)				Preview (Nightly)
Your OS	Linux		Mac		Windows
Package	Conda		Pip	LibTorch	Source
Language	Python			C++ / Java	
CUDA	9.2	10.1	10.2	11.0	None
Run this Command:	NOTE: Python 3.9 users will need to add '-c=conda-forge' for installation <code>conda install pytorch torchvision torchaudio cudatoolkit=11.0 -c pytorch</code>				

<https://pytorch.org/>

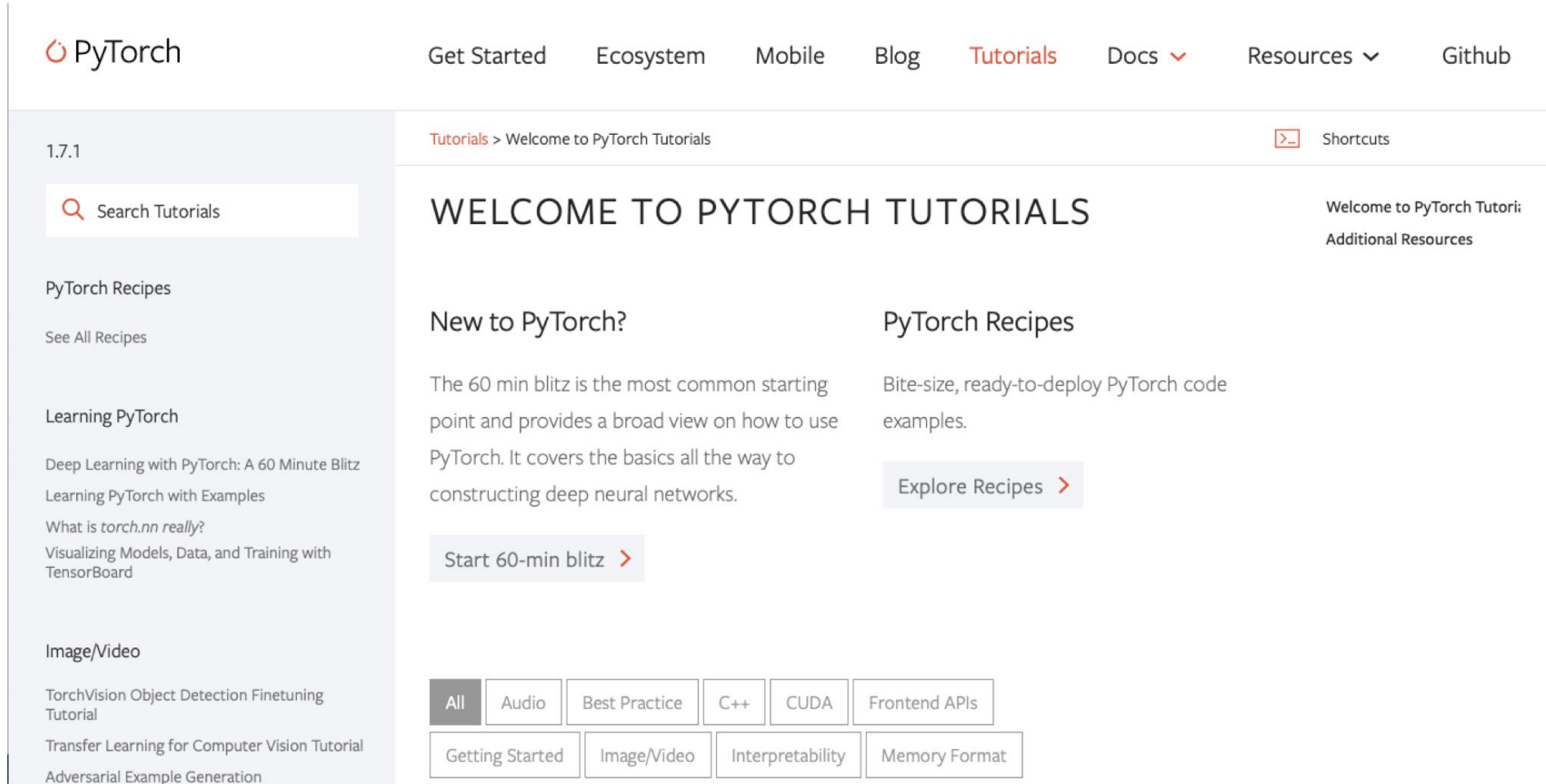
And don't forget that you import PyTorch as "import torch," not "import pytorch" :)

```
[In [1]: import torch

[In [2]: torch.__version__
Out[2]: '1.7.0'

In [3]:
```

Many useful tutorials (recommend you read some)



The screenshot shows the PyTorch Tutorials page. The header includes the PyTorch logo and navigation links: Get Started, Ecosystem, Mobile, Blog, Tutorials (highlighted), Docs, Resources, and Github. The left sidebar contains a search bar, a version selector (1.7.1), and a list of tutorial categories: PyTorch Recipes, Learning PyTorch (with links to 'Deep Learning with PyTorch: A 60 Minute Blitz', 'Learning PyTorch with Examples', 'What is torch.nn really?', and 'Visualizing Models, Data, and Training with TensorBoard'), and Image/Video (with links to 'TorchVision Object Detection Finetuning Tutorial', 'Transfer Learning for Computer Vision Tutorial', and 'Adversarial Example Generation'). The main content area is titled 'WELCOME TO PYTORCH TUTORIALS' and features a 'New to PyTorch?' section with a 'Start 60-min blitz' button. To the right, there's a 'PyTorch Recipes' section with an 'Explore Recipes' button. At the bottom, a grid of filters allows users to narrow down tutorials by category (All, Audio, Best Practice, C++, CUDA, Frontend APIs, Getting Started, Image/Video, Interpretability, Memory Format).

PyTorch

Get Started Ecosystem Mobile Blog **Tutorials** Docs Resources Github

Tutorials > Welcome to PyTorch Tutorials

Shortcuts

WELCOME TO PYTORCH TUTORIALS

Welcome to PyTorch Tutorials
Additional Resources

New to PyTorch?

The 60 min blitz is the most common starting point and provides a broad view on how to use PyTorch. It covers the basics all the way to constructing deep neural networks.

Start 60-min blitz >

PyTorch Recipes

Bite-size, ready-to-deploy PyTorch code examples.

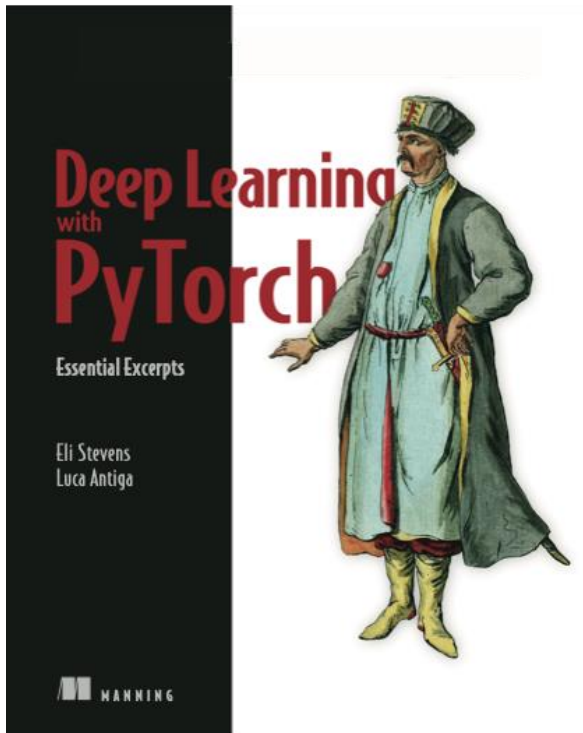
Explore Recipes >

Filters:

All	Audio	Best Practice	C++	CUDA	Frontend APIs
Getting Started	Image/Video	Interpretability	Memory Format		

<https://pytorch.org/tutorials/>

Other resources



PyTorch

Do you want live notifications when people reply to your posts? [Enable Notifications](#)

all categories ▾ all ▾ Latest New (47) Unread (104) Top Categories + New Topic

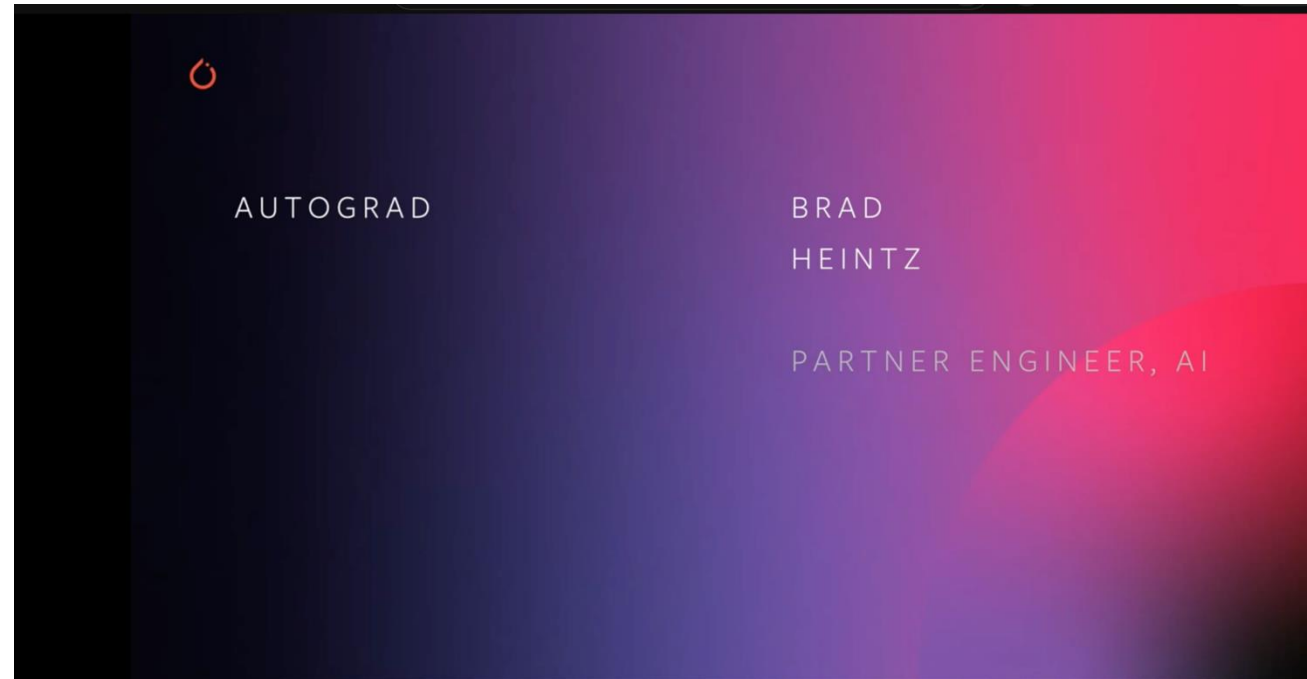
Topic	Replies	Views	Activity
Using MSELoss instead of CrossEntropy for Ordinal Regression/Classification vision	2	83	1h
Optimizer.load_state_dict() weird behaviour with Adam optimizer vision	7	2.0k	1h
Is there a way to train 3 dataloaders using multiprocessing?	0	11	2h
Getting different feature vectors from frozen layers after training vision	5	86	2h
Libtorch_cuda.so is too large (>2GB) deployment	22	346	2h
Undo pruning - How to 'unmask' pruned weights vision	0	8	2h
If input.dim() == 2 and bias is not None: AttributeError: 'tuple' object has no attribute 'dim'	3	41	2h
Export unsupported/compound ops to ONNX deployment	0	9	2h

<https://discuss.pytorch.org>

And...

Ask ChatGPT/Claude if your
PyTorch code is not working

Automatic Differentiation in PyTorch



[The Fundamentals of Autograd](#) (video)



Automatic Differentiation in PyTorch

- Live walkthrough notebook:
`Lecture_06_autograd_walkthrough.ipynb`

Distribution → Likelihood → Loss

- Specify: $z = w^T x + b$, $\hat{y} = \sigma(z)$, $y \mid x \sim \text{Bernoulli}(\hat{y})$
- Likelihood: $P(y \mid x) = \hat{y}^y (1-\hat{y})^{1-y}$
- Log-likelihood: $\log P(y \mid x) = y \log \hat{y} + (1-y) \log(1-\hat{y})$
- Loss = negative log-likelihood: $L = -[y \log \hat{y} + (1-y) \log(1-\hat{y})]$ (binary cross-entropy)
- In the notebook: this formula, `torch.distributions.Bernoulli( $\hat{y}$ ).log_prob(y)`, and `nn.BCELoss()` all give the exact same number.



Today: Automatic differentiation; projects

- Automatic Differentiation in PyTorch
- **A Closer Look at the PyTorch API**
- Project Discussion

PyTorch Usage: Step 1 (Definition)

```
class MultilayerPerceptron(torch.nn.Module):  
  
    def __init__(self, num_features, num_classes):  
        super(MultilayerPerceptron, self).__init__()  
  
        ### 1st hidden layer  
        self.linear_1 = torch.nn.Linear(num_feat, num_h1)  
  
        ### 2nd hidden layer  
        self.linear_2 = torch.nn.Linear(num_h1, num_h2)  
  
        ### Output layer  
        self.linear_out = torch.nn.Linear(num_h2, num_classes)  
  
    def forward(self, x):  
        out = self.linear_1(x)  
        out = F.relu(out)  
        out = self.linear_2(out)  
        out = F.relu(out)  
        logits = self.linear_out(out)  
        probas = F.log_softmax(logits, dim=1)  
        return logits, probas
```

Backward will be inferred automatically if we use the nn.Module class!

Define model parameters that will be instantiated when created an object of this class

Define how and it what order the model parameters should be used in the forward pass

PyTorch Usage: Step 2 (Creation)

```
torch.manual_seed(random_seed)
model = MultilayerPerceptron(num_features=num_features,
                             num_classes=num_classes)
model = model.to(device)

optimizer = torch.optim.SGD(model.parameters(),
                             lr=learning_rate)
```

Instantiate model
(creates the model parameters)

Define an optimization method

PyTorch Usage: Step 3 (Training)

```

for epoch in range(num_epochs):
    model.train()
    for batch_idx, (features, targets) in enumerate(train_loader):

        features = features.view(-1, 28*28).to(device)
        targets = targets.to(device)

        ### FORWARD AND BACK PROP
        logits, probas = model(features)
        cost = F.cross_entropy(probas, targets)
        optimizer.zero_grad()

        cost.backward()

        ### UPDATE MODEL PARAMETERS
        optimizer.step()

    model.eval()
    with torch.no_grad():
        # compute accuracy

```

Run for a specified number of epochs

Iterate over minibatches in epoch

If your model is on the GPU, data should also be on the GPU

`y = model(x)` calls `__call__` and then `.forward()`, where some extra stuff is done in `__call__`;

don't run `y = model.forward(x)` directly

Gradients at each leaf node are accumulated under the `.grad` attribute, not just stored. This is why we have to zero them before each backward pass

PyTorch Usage: Step 3 (Training)

```
for epoch in range(num_epochs):
    model.train()
    for batch_idx, (features, targets) in enumerate(train_loader):

        features = features.view(-1, 28*28).to(device)
        targets = targets.to(device)

        ### FORWARD AND BACK PROP
        logits, probas = model(features)  ← This will run the forward() method
        loss = F.cross_entropy(logits, targets) ← Define a loss function to optimize
        optimizer.zero_grad() ← Set the gradient to zero
                                   (could be non-zero from a previous forward pass)

        loss.backward()

        ### UPDATE MODEL PARAMETERS
        optimizer.step() ← Compute the gradients, the backward is
                               automatically constructed by "autograd" based on
                               the forward() method and the loss function

    model.eval()
    with torch.no_grad():
        # compute accuracy

        Use the gradients to update the weights according to
        the optimization method (defined on the previous
        slide)
        E.g., for SGD,  $w := w + \text{learning\_rate} \times \text{gradient}$ 
```

PyTorch Usage: Step 3 (Training)

```
for epoch in range(num_epochs):
    model.train()
    for batch_idx, (features, targets) in enumerate(train_loader):

        features = features.view(-1, 28*28).to(device)
        targets = targets.to(device)

        ### FORWARD AND BACK PROP
        logits, probas = model(features)
        loss = F.cross_entropy(logits, targets)
        optimizer.zero_grad()

        loss.backward()

        ### UPDATE MODEL PARAMETERS
        optimizer.step()

    model.eval()
    with torch.no_grad():
        # compute accuracy
```

For evaluation, set the model to eval mode (will be relevant later when we use DropOut or BatchNorm)

This prevents the computation graph for backpropagation from automatically being build in the background to save memory

Simple “print” statements don’t work for debugging

```
[7]: model.net
[7]: Sequential(
  (0): Linear(in_features=784, out_features=128, bias=True)
  (1): ReLU(inplace)
  (2): Linear(in_features=128, out_features=256, bias=True)
  (3): ReLU(inplace)
  (4): Linear(in_features=256, out_features=10, bias=True)
)

[ ]: If we want to get the output from the 2nd layer during the forward pass, we can register a hook as follows:

[8]: outputs = []
def hook(module, input, output):
    outputs.append(output)

model.net[2].register_forward_hook(hook)

[8]: <torch.utils.hooks.RemovableHandle at 0x7f659c6685c0>

Now, if we call the model on some inputs, it will save the intermediate results in the "outputs" list:

[9]: _ = model(features)

print(outputs)

[tensor([[0.5341, 1.0513, 2.3542, ..., 0.0000, 0.0000, 0.0000],
        [0.0000, 0.6676, 0.6620, ..., 0.0000, 0.0000, 2.4056],
        [1.1520, 0.0000, 0.0000, ..., 2.5860, 0.8992, 0.9642],
        ...,
        [0.0000, 0.1076, 0.0000, ..., 1.8367, 0.0000, 2.5203],
        [0.5415, 0.0000, 0.0000, ..., 2.7968, 0.8244, 1.6335],
        [1.0710, 0.9805, 3.0103, ..., 0.0000, 0.0000, 0.0000]],
        device='cuda:3', grad_fn=<ThresholdBackward1>)]
```



Today: Automatic differentiation; projects

- Automatic Differentiation in PyTorch
- A Closer Look at the PyTorch API
- **Project Discussion**

Project Ideas

- Apply a deep generative model to a new domain or dataset (your own research data counts!)
- Reproduce and extend a result from a paper we've read, or one you find
 - ICML, NeurIPS, ICLR, AISTATS
- Build/benchmark a tool: an evaluation metric, an interpretability method, an efficient-training trick
- Compare architectures or training methods on a shared task
- Examples from prior years: breast cancer detection from ultrasound (LSTM), DCGAN surrealist art, chest X-ray diagnosis, sentiment analysis with BERT, audio-to-video animation



Datasets

- Vision: MNIST, CIFAR-10/100 (demos), ImageNet, or your own image data
- Text: Hugging Face Datasets hub, OpenWebText, or your own corpus
- Tabular: UCI Machine Learning Repository, Kaggle
- Your own research group's data, if applicable to your project
- Check licensing and any human-subjects/IRB considerations before using sensitive data



An option for exploratory data analysis

summand.com

 SUMMAND

Sign In [Get Started](#) 

Welcome to Summand

Chat with your data. Upload a CSV or connect a database to get insights in minutes.

Summa

  Connectors 

 Thinking level 



Analyze trends in my sales data

Find anomalies in a CSV

Generate a report from my dataset

Connect my Postgres database

Deliverables & Timeline

- Teams of 2-4 students (3-4 strongly encouraged)
 - Email the instructors once your team is formed
- Proposal (5%): due Fri Oct 9
 - Problem statement, literature review (4+ papers), dataset & activity plan
- Midway Report (5%): due Fri Nov 6
 - Real progress: implementation + preliminary results, not just a plan
- In-Class Presentation (5%): Tue, Dec 1 / Thu, Dec 3
 - Team talk, individually graded
- Final Report (15%): due Tue Dec 8
 - 8-page ICML-format paper with baseline, ablation, and error analysis

Let's talk about projects



Questions?

