The Ultimate PyTorch Lock-In Guide

From Zero to Tensor Hero: Math Notation + PyTorch Fundamentals

3-Hour Masterclass Session

PART 1: MATH FOUNDATION (30 minutes)

Master these symbols before touching PyTorch code.

1.1 Essential Math Notation Crash Course

Symbol	Name	What It Means	Simple Example
x, y, z	Variables	Unknown numbers we want to find	$x = 5$
$\mathbb{R}$	Real Numbers	All numbers on number line	$3, -2.5, 0, \pi \in \mathbb{R}$
$\in$	Belongs to	Shows membership in a set	$x \in \mathbb{R}$ (x is a real number)
$\mathbb{R}^n$	n-dimensional space	List of n real numbers	$\mathbb{R}^3 = (x, y, z)$ (3D coords)
$\sum$	Summation	Add up a series of numbers	$\sum_{i=1}^{5} i = 1+2+3+4+5 = 15$
$\prod$	Product	Multiply a series of numbers	$\prod_{i=1}^{3} i = 1 \times 2 \times 3 = 6$
∂	Partial derivative	Rate of change while others stay constant	$\frac{\partial y}{\partial x}$
∇	Gradient	Vector of all partial derivatives	$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$
$\|\mathbf{x}\|$	Norm/Magnitude	Length of a vector	$\|[3,4]\| = \sqrt{3^2+4^2} = 5$
$\mathbf{x}^T$	Transpose	Flip rows and columns	$[1,2,3]^T$ becomes vertical
$\cdot$	Dot product	Multiply corresponding elements and sum	$[1,2] \cdot [3,4] = (1)(3) + (2)(4) = 11$

1.2 Breaking Down the Scary Symbols

Sigma Notation ($\sum$) - The Loop of Math

Read as: "Sum from $i = 1$ to n of $f(i)$"

$$\sum_{i=1}^{4} i^2 = 1^2 + 2^2 + 3^2 + 4^2 = 30$$

Python equivalent: `sum([i**2 for i in range(1, 5)])`

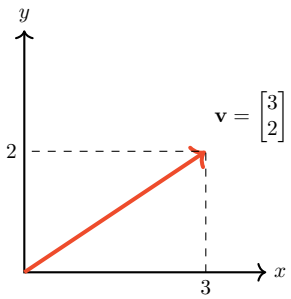
Partial Derivatives (∂) - The Slope Finder

Regular derivative (dy/dx): Slope walking straight east.
Partial ($\partial y/\partial x$): Slope walking east, keeping north constant.

If $z = x^2 + 3xy + y^2$

$$\frac{\partial z}{\partial x} = 2x + 3y \quad \text{(Treat } y \text{ as constant)}$$

1.3 Matrix & Vector Basics



1-D Tensor (Vector)
Represents magnitude and direction in space.

PART 2: PYTORCH CORE CONCEPTS (90 minutes)

2.1 TENSOR - The Fundamental Object

A tensor is a multi-dimensional array that can run on a GPU.

- **Scalar (0-D):** $a \in \mathbb{R} \rightarrow$ `torch.tensor(5)`
- **Vector (1-D):** $\mathbf{v} \in \mathbb{R}^n \rightarrow$ `torch.tensor([1, 2, 3])`
- **Matrix (2-D):** $\mathbf{M} \in \mathbb{R}^{m \times n} \rightarrow$ `torch.tensor([[1, 2], [3, 4]])`
- **Tensor (n-D):** $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_n} \rightarrow$ e.g., Batch of RGB Images: (Batch, Channels, Height, Width)

2.2 Tensor Operations & Broadcasting

Element-wise (Hadamard Product) $\odot$

```
A = torch.tensor([[1, 2], [3, 4]])
B = torch.tensor([[5, 6], [7, 8]])
C = A * B
# Result: [[5, 12], [21, 32]]
```

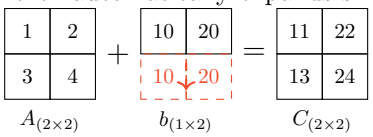
Matrix Multiplication (Dot Product) @

Inner dimensions must match! $\mathbb{R}^{m \times n} \times \mathbb{R}^{n \times p} = \mathbb{R}^{m \times p}$

```
E = A @ B  # or torch.matmul(A, B)
# E[0,0] = (1*5) + (2*7) = 19
```

Visualizing Broadcasting

PyTorch automatically expands smaller tensors.



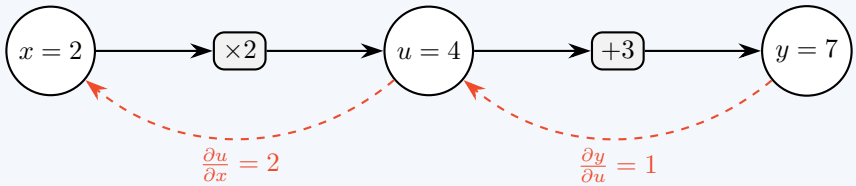
$A_{(2\times2)}$ $b_{(1\times2)}$ $C_{(2\times2)}$

2.3 AUTOGRAD & The Computation Graph

The Chain Rule: If $y = f(g(x))$, then $\frac{dy}{dx} = \frac{dy}{dg} \times \frac{dg}{dx}$. PyTorch tracks this dynamically!

Visualizing the Computational Graph (Forward & Backward Pass)

FORWARD PASS (Compute y)



$\frac{\partial u}{\partial x} = 2$ $\frac{\partial y}{\partial u} = 1$

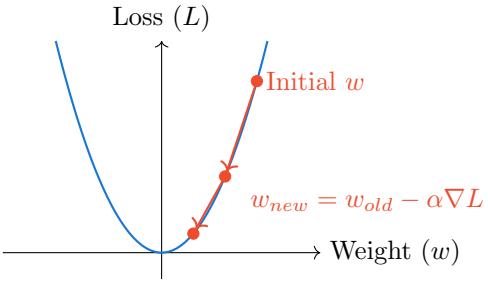
BACKWARD PASS: $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \times \frac{\partial u}{\partial x} = 1 \times 2 = \mathbf{2}$

PyTorch Autograd Code:

```
import torch
x = torch.tensor(2.0, requires_grad=True)
y = (x * 2) + 3

y.backward()
print(x.grad)  # Output: tensor(2.)
```

Gradient Descent Visualized



PART 3: MEMORY TABLES & CHEAT SHEETS

Table 1: Tensor Creation

Math / Concept	PyTorch Code
Scalar $5 \in \mathbb{R}$	`torch.tensor(5)`
Vector $\mathbf{v} \in \mathbb{R}^3$	`torch.tensor([1,2,3])`
Zeros $\mathbf{0}^{2\times3}$	`torch.zeros(2,3)`
Identity $\mathbf{I}_3$	`torch.eye(3)`
Uniform $\mathcal{U}(0,1)$	`torch.rand(2,2)`
Normal $\mathcal{N}(0,1)$	`torch.randn(2,2)`
Range $\{0..9\}$	`torch.arange(10)`
5 points 0 to 1	`torch.linspace(0,1,5)`

Table 2: Operations Symbols

Sym	PyTorch	What It Does
$+$	`a + b`	Element-wise add
$*$	`a * b`	Hadamard (element mult)
@	`a @ b`	Matrix multiply (dot)
**	`a ** 2`	Element-wise exponent

Table 3: Shape Manipulation

Operation	Change
Reshape	$(2,3) \to (3,2)$
Transpose	$(2,3) \to (3,2)$
Add dim	$(3,4) \to (1,3,4)$
Drop dim	$(1,3,1) \to (3)$
Flatten	$(2,3) \to (6,)$
Permute	$(H,W,C) \to (C,$

Table 4: Gradient Flow

Action	PyTorch Code
Compute Grad	`y.backward()`
Access Grad	`x.grad`
Clear Grads	`optimizer.zero_g`
Stop Tracking	`with torch.no_gra`

PART 4: PRACTICE QUESTION SET

Level 1: Tensor Basics

Q1. Create a tensor representing this matrix: $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Q2. What is the shape of `torch.tensor([[[1,2],[3,4]],[[5,6],[7,8]]])`? Draw it.

Q3. Create a 3×3 identity matrix I_3. Verify $I_3 \cdot A = A$.

Q4. Generate 100 random numbers from $\mathcal{N}(0,1)$, find mean and std.

Q5. Create a tensor with values 0 to 100, step 5 using `arange` and `linspace`.

Level 2: Operations & Broadcast

Q6. Given $\mathbf{u} = [1,2,3]^T, \mathbf{v} = [4,5,6]^T$. Compute: element-wise product, dot product, and outer product $\mathbf{u}\mathbf{v}^T$.

Q7. Explain why `A = torch.ones(3,4); b = torch.arange(4); C = A + b` works. What is the shape of C?

Q8. Matrix multiply puzzle: Can you multiply $A_{(3\times4)}$ and $B_{(4\times2)}$? Output shape?

Q9. Reshape challenge: Tensor shape '(2,3,4)'. What are all possible 2D shapes?

Q10. Prove $(A^T)^T = A$ via PyTorch code.

Level 3: Autograd

Q11. Find derivative of $f(x) = x^3 - 2x^2 + 5x - 7$ at $x = 2$ using autograd.

Q12. For $f(x,y) = x^2y + y^3$, compute ∇f at $(2,3)$. Verify manually.

Q13. Let $u = 2x + 1$, $y = u^2$. Find dy/dx at $x = 1$ via Chain Rule and Autograd.

Q14. What happens if you run `.backward()` twice without zeroing gradients?

Q15. Compute $\frac{\partial L}{\partial w}$ for MSE Loss $L = \frac{1}{n}\sum(y_i - (wx_i + b))^2$.

Level 4: Applied Math

Q16. Implement Softmax: $\frac{e^{x_i}}{\sum e^{x_j}}$ for $\mathbf{x} = [1,2,3]$.

Q17. Implement MSE Loss function manually.

Q18. Batch multiply: Tensors '(32, 10, 20)' and '(32, 20, 30)'. What is the shape?

Q19. Image shape '(3, 224, 224)'. How to convert to '(224, 224, 3)'?

Q20. If $\nabla L = [2,-3]$, $\alpha = 0.1$. What is the update direction? Why subtract?

PART 5: SOLUTIONS & EXPLANATIONS

Level 1 Solutions

A1. `A = torch.tensor([[1,2,3],[4,5,6]])` (Shape: 2x3)
A2. Shape is `(2, 2, 2)`. It is a 3D Cube (2 layers of 2x2 matrices).
A3. `I = torch.eye(3); result = I @ A`
A4. `x = torch.randn(100); x.mean(); x.std()`
A5. `torch.arange(0,101,5)` and `torch.linspace(0,100,21)`

Level 2 Solutions

A6. Hadamard: `u * v` $\rightarrow [4, 10, 18]$. Dot: `torch.dot(u,v)` $\rightarrow$ 32. Outer: `torch.outer(u,v)` $\rightarrow 3 \times 3$ matrix.
A7. Broadcasting expands `b` from '(4,)' to '(1,4)' to '(3,4)'. Shape is '(3,4)'.
A8. Yes. Inner dims match ($4 = 4$). Output is '(3, 2)'.
A9. Total elements = 24. 2D shapes: (2,12), (3,8), (4,6), (6,4), (8,3), etc.
A10. `double_t = A.t().t(); torch.equal(A, double_t)` $\rightarrow$ True.

Level 3 Solutions

A11. $f'(x) = 3x^2 - 4x + 5 \rightarrow f'(2) = 9$. `x.grad` returns 9.
A12. $\partial f/\partial x = 2xy = 12$. $\partial f/\partial y = x^2 + 3y^2 = 31$.
A13. $dy/dx = (dy/du)(du/dx) = (2u)(2) = 4(3) = 12$.
A14. Gradients accumulate! PyTorch adds new gradients to existing ones. Call `optimizer.zero_grad()` to fix.
A15. Uses Autograd. Chain rule computes $-2x_i(y_i - \hat{y}_i)/n$.

Level 4 Solutions

A16. `exp_x = torch.exp(x); probs = exp_x / exp_x.sum()`
A17. `loss = ((y_true - y_pred)**2).mean()`
A18. `torch.bmm(A,B)`. Output: '(32, 10, 30)'.
A19. `image.permute(1, 2, 0)`. Converts Channel-First to Channel-Last.
A20. $\theta_{new} = \theta_{old} - 0.1[2, -3] = \theta_{old} + [-0.2, 0.3]$. Subtract to go against gradient (descent).

Speed Hacks & Memory Techniques

Mental Models

- **Tensor:** Just a fancy multi-dimensional array on steroids (GPU + Grads).
- **BCHW:** Image dimensions are always Batch, Channel, Height, Width.
- **Autograd Pipes:** Imagine water flowing backward through pipes. Gradients are valves.

Coding Speed Hacks

- Use `torch.zeros_like(x)` to copy shape and device simultaneously.
- Use `with torch.no_grad():` during inference to prevent memory explosion.
- Never do `x += 1` if tracking grads. Do `x = x + 1`.

Final Exam: Check Your Intuition

Question	Quick Answer
What does $\mathbb{R}^{3\times4}$ mean?	A matrix with 3 rows and 4 columns of real numbers.
Shape of $(AB)^T$ if $A_{(2\times3)}, B_{(3\times4)}$?	(4×2). The matrix multiplication yields (2×4), transpose flips it.
Why write `loss.backward()`?	Triggers the computation graph to calculate $\partial L/\partial w$ for all weights.
How to fix device errors?	Ensure data `x` and `model` are both on `.to('cuda')` or `'cpu'`.

Lock-In Complete. $\nabla \mathbf{L} \rightarrow \mathbf{0}$

You now have the mathematical and practical foundation to build neural networks.