DS 6050: Deep Learning

School of Data Science, University of Virginia

Spring 2026

Instructor: Heman Shakeri, PhD	**Email:** hs9hd@virginia.edu
TAs: Fokhrul Islam, and Tom Lever	sdh4py & tsl2b@virginia.edu

- **Asynchronous Lectures & Readings:** Available weekly on Canvas.
- **Zoom Sessions:** Wednesdays, 7:15 PM – 8:15 PM
- **Course Website:** Link

Other Resources:
- Assignments submission (UVA Canvas)
- Discussion Forums & Announcements (Ed Discussion)

Why Should You Care About Learning Deep Learning?

How can you design an AI that truly understands complex medical images, not just mimics patterns? How do you build a language model that can generate coherent text, or an algorithm that can innovate in scientific discovery? These are the kinds of "grand challenges" that today's data scientists and ML engineers tackle, requiring more than just off-the-shelf code. They demand a deep, quantitative understanding of how these models work, from first principles.

You've likely seen countless AI tutorials online, much like the millions of fitness videos on YouTube. Yet, professional athletes and Olympic medalists dedicate themselves to rigorous training camps with expert coaches. Why? Because true mastery—the kind that forges resilient skill and deep intuition—isn't about mimicking surface-level actions. It's built on structured guidance, disciplined hands-on practice with expert feedback, and a profound grasp of fundamentals.

This course is your intensive "training camp" for deep learning. It's where you'll move beyond superficial understanding to forge genuine expertise, from the core out, embracing the deliberate practice needed to push your boundaries and achieve lasting skill.

How Will This Course Help You Succeed (As a Data Scientist/ML Engineer)?

Grand challenges in AI require data scientists who can think critically, design robust systems, and make informed trade-offs. This course will equip you with a conceptual and practical framework to tackle such complex problems in your future research, engineering practice, or advanced studies.

By the end of this training camp, you will be better able to answer critical questions like:

1. How do I use math and core principles to truly understand why, how, and where deep learning models succeed or fail? (No more magical black boxes!)
2. When and how can I build fundamental components (like backpropagation or an attention mechanism) "from scratch" to solidify my understanding, before leveraging powerful frameworks like PyTorch?
3. I've taken calculus, linear algebra, and ML I – how do I integrate all that knowledge with programming to solve real-world AI problems and design innovative systems?
4. Beyond the buzzwords and LinkedIn cheatsheets, how do I develop the critical judgment to make sound engineering decisions, evaluate new research, and maintain my intellectual integrity?
5. How do I use the foundational skills and "learning to learn" strategies from this course to confidently tackle future AI advancements and projects long after graduation?

Course Structure & How You'll Master Deep Learning

This course follows a carefully designed 12-module progression that builds deep learning expertise through three integrated phases:

Phase 1: Foundations & From-Scratch Understanding (Modules 1-3) Master the mathematical underpinnings and implement core algorithms from scratch. You'll build linear models, neural networks, and backpropagation using only NumPy, developing deep intuition about how and why deep learning works. Critical emphasis on optimization foundations and ablation methodology that will recur throughout the course.

Phase 2: Architectural Innovations & Domain Specialization (Modules 4-9) Explore how different data modalities drive architectural choices. From CNNs for spatial data to RNNs for sequences to Transformers for universal modeling, you'll understand why specific architectures excel at specific tasks. Each module revisits optimization challenges unique to that architecture, building your ability to diagnose and fix real-world training problems.

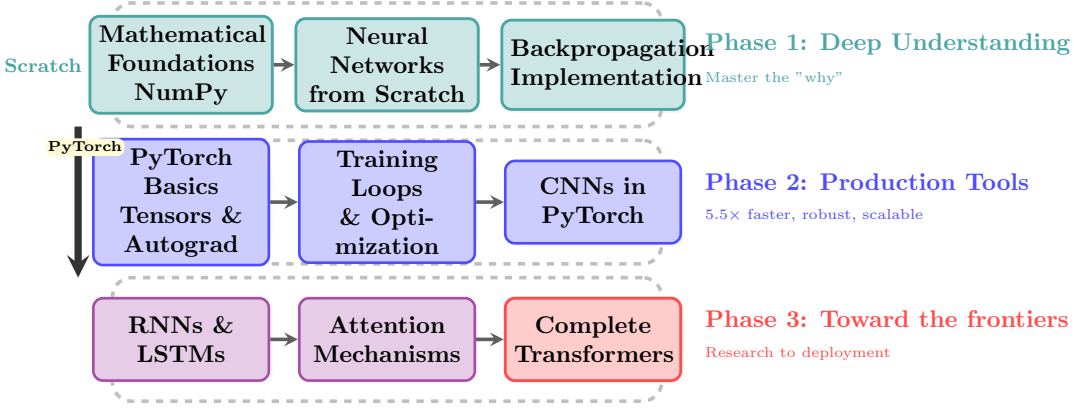

Figure 1: Learning Progression: From Mathematical Foundations to Modern AI Systems

Phase 3: Modern Practice & Research Skills (Modules 10-12) Master contemporary techniques including vision transformers, large-scale pretraining, and generative models. Learn systematic ablation studies, paper reproduction, and research methodology. The course culminates with your team tackling a "grand challenge" using all the skills you've developed.

Course Calendar

Table 1: Course Schedule

Week	Dates	Live Session Meetings
Week 1	Kick off/Logistics.	Wed 1/14, 7:15–8:15PM
Week 2	Module 1	Wed 1/21, 7:15–8:15PM
Week 3	Module 2	Wed 1/28, 7:15–8:15PM
Week 4	Module 3	Wed 2/04, 7:15–8:15PM
Week 5	Module 4	Wed 2/11, 7:15–8:15PM
Week 6	Module 5	Wed 2/18, 7:15–8:15PM
Week 7	Module 6	Wed 2/25, 7:15–8:15PM
Week 8	No class	Wed 3/04 No live session
Week 9	Module 7	Wed 3/11, 7:15–8:15PM
Week 10	Module 8	Wed 3/18, 7:15–8:15PM
Week 11	Module 9	Wed 3/25, 7:15–8:15PM
Week 12	Module 10	Wed 4/01, 7:15–8:15PM
Week 13	Module 11	Wed 4/08, 7:15–8:15PM
Week 14	Module 12	Wed 4/15, 7:15–8:15PM
Week 15	No class	Wed 4/22 No live session

Module Design Tables

Table 2: Module Design — Deep Learning (Domain-Agnostic)

	Module	Learning Goals	Learning Objectives	Instructional Materials
Chapter 1 Foundations & MLPs	**Module 1: Linear → Nonlinear**	• Linear ops & geometry • Why nonlinearity • Build an MLP • SGD overview	• Compute matrix–vector products and visualize linear transforms. • Explain XOR nonlinearity and demonstrate failure → fix with ReLU. • Implement a minimal MLP (NumPy/PyTorch) and describe SGD updates.	• **D2L:** 2.3; 3.1.4; 4.1; 5.1; 12.4 • PyTorch basics; NumPy refresher • Short geometric notes/figures
	Module 2: Backprop & Autograd	• Chain rule & comp. graphs • Backward pass • Autograd usage	• Derive gradients for a 1–2 layer network and draw the computation graph. • Implement gradient checking and validate layer derivatives. • Use `torch.autograd` to build a tiny autograd toy and inspect backward.	• **D2L:** 2.4; 2.5; 5.3 • PyTorch autograd guide
	Module 3: Training & Experiments	• Ablations & HPO • Reproducibility & logging	• Run an optimizer sweep (SGD/Momentum/Adam) with LR schedules; compare results. • Design a 2–3 factor ablation; log all runs with consistent seeds and configs.	• **D2L:** 3.6–3.7; 12; 19 • W&B (or equivalent) quickstart
Chapter 2 Convolutions & Depth	**Module 4: CNN Basics**	• Why convolution & sharing • Spatial invariance • Regularization	• Show MLP spatial limits (shifted inputs) vs a simple CNN. • Implement convolutional layers and train a small CNN. • Apply augmentation and dropout; run a brief augmentation ablation.	• **D2L:** 5.6; Ch. 7; 8.1; 14.1 • *AlexNet* (Krizhevsky et al. 2012) • Conv visualizers/notes
	Module 5: Modern CNNs & Transfer	• Deeper blocks & normalization • Residual connections • Transfer learning	• Inspect gradient flow; compare networks with/without BatchNorm. • Implement a ResNet block; reproduce a mini-ResNet training. • Fine-tune a pretrained model; summarize efficiency trade-offs (e.g., MobileNet).	• **D2L:** 5.4.1; 8.2–8.6; 14.2 • *ResNet* (He et al. 2016); BN (Ioffe–Szegedy) • Model zoo pointers

Continued on next page

Table 2 – continued

Chapter	Module	Learning Goals	Learning Objectives	Instructional Materials
Chapter 3 Sequence Data	**Module 6: Encoder-Decoder Architectures**	• Compression & representation • Fixed-size bottlenecks • Skip connections • Segmentation applications	• Implement an autoencoder from scratch; visualize learned representations. • Build a CNN encoder-decoder for image tasks.	• **D2L:** 10.6; 14.11 • U-Net (Ronneberger et al.) • Encoder-decoder papers
	Module 7: RNNs: Vanilla, LSTM & seq2seq	• Sequential processing • Vanishing/exploding gradients • LSTM/GRU gates • Training challenges	• Implement char-RNN and visualize gradient flow over time steps. • Build LSTM from scratch; run gate ablation studies. • Compare LSTM vs GRU performance and computational efficiency. • Apply gradient clipping and truncated BPTT; analyze training stability.	• **D2L:** 9; 10 • LSTM (Hochreiter–Schmidhuber); GRU (Cho et al.) • Gradient visualization tools
Chapter 4 Attention & Transformers	**Module 8: Attention & Seq2Seq**	• Dot-product attention • Encoder–decoder • Alignment/weights	• Implement QKV attention; visualize attention weights. • Build a toy seq2seq (copy/reverse) with attention; interpret alignments. • Run a small ablation on attention variants and report findings.	• **D2L:** 10.5–10.7; 11.1–11.4 • Bahdanau et al. (additive attention)
	Module 9: Transformer Blocks	• Multi-head self-attn • Positional encodings • Norm/warmup/dropout	• Implement a transformer block and train a tiny Transformer. • Compare positional encodings; pre- vs post-norm; analyze patterns. • Use warmup and dropout; document stability/perf. impacts.	• **D2L:** 11.5–11.7 • Vaswani et al. *Attention Is All You Need*
	Module 10: ViT Essentials (Domain-Agnostic)	• Patches as tokens • Patch embeddings • CNN–ViT trade-offs	• Implement patchify + linear embedding; train a tiny ViT. • Run a patch-size ablation; inspect attention maps qualitatively. • Compare a small ViT vs small ResNet for efficiency vs accuracy.	• **D2L:** 11.8 • Dosovitskiy et al. (ViT); Touvron et al. (DeiT)

Continued on next page

Table 2 – continued

Chapter	Module	Learning Goals	Learning Objectives	Instructional Materials
	Module 11: Prompting, PEFT & Quantization	• Prompt engineering & RAG • Parameter-efficient FT (LoRA/Prefix) • Quantization for deployment	• Build and critique prompts; experiment with retrieval-augmented responses. • Fine-tune a small model with LoRA vs full FT; compare quality/efficiency. • Quantize a model and report latency/quality impacts; reflect on trade-offs.	• **D2L:** 11.9; 15.8–15.10; 16.6–16.7 • Hu et al. (LoRA); Brown et al. (GPT-3); Wei et al. (CoT); Dettmers et al. (QLoRA)
Chapter 5 Generative & Multimodal Models	**Module 12: Multimodal Learning & GenAI**	• Vision-language fusion • Generative pipelines (VAE/-GAN/diffusion) • Safety & evaluation	• Implement a simple fusion model (e.g., CLIP-style) and inspect attention. • Train a small generative model (VAE/GAN or tiny diffusion) and visualize samples. • Discuss evaluation metrics and basic safety/ethics checks for generative systems.	• **D2L:** 4.7.5; 20.1–20.2 • Radford et al. (CLIP); Kingma–Welling (VAE); Goodfellow et al. (GAN); Ho et al. (DDPM)

Assessment as Learning: How You'll Build and Demonstrate Mastery

Assessment in this course is designed primarily as a learning tool, with evaluation as a by-product. Each assessment provides opportunities to deepen understanding, receive feedback, and refine your skills.

Programming Assignments: Deliberate Practice with Feedback

Five comprehensive assignments that build your implementation skills progressively:

- **Learning Focus:** Each assignment includes extensive unit tests that guide your implementation and provide immediate feedback
- **Iterative Development:** You can discuss together and explore ideas in the forum
- **Reflection Component:** Brief write-ups explaining your design choices and debugging process
- **Peer Code Review:** Optional exchanges with classmates to learn from different approaches

Assignment Progression (current format):

1. Foundations & Backprop (Modules 1–2) → Homework 1 (Colab on Module 1 page): linear models, gradients, and early autograd practice.
2. Optimization & CNNs (Modules 4–5) → Homework 2 (Colab on Module 5 page): training stability, convolutional stacks, and transfer learning.
3. Seq2Seq (Module 7) → Homework 3: seq2seq (Colab). Link: Colab notebook.
4. Attention (Modules 8) → Homework 4: Add cross-attention to your GRU-based seq2seq model
5. Module 10 → Homework 5: Transformer 2.0 (Colab). Link: Colab notebook.

Module Quizzes: Knowledge Reinforcement

- **Purpose:** Reinforce key concepts from asynchronous content and readings
- **Format:** Short (10-15 questions), open-book, focused on understanding not memorization
- **Completion-Based:** Full credit for thoughtful completion, encouraging exploration over perfection
- **Immediate Feedback:** Explanations provided for all answers to support learning

Group Project: Solving a Grand Challenge

Multi-phase project applying course concepts to real-world problems:

Formative Milestones:

1. **Proposal & Literature Review:** Receive feedback on problem framing and approach
2. **Mid-Project Check-in:** Share challenges and get guidance from peers and instructors
3. **Final Deliverables:** Complete solution with documentation and reflection

Peer Learning Through Discussion Boards:

- Teams post project report that has key findings, and link for GitHub repo on Canvas
- Structured peer feedback using provided rubric
- Asynchronous Q&A threads for each project

Participation: Building a Learning Community

Active engagement that enhances everyone's learning:

- **Discussion Leadership:** Each student leads one module discussion thread
- **Peer Support:** Help classmates with conceptual questions and debugging
- **Reflection Posts:** Weekly "aha moments" or challenging concepts
- **Module Quiz Completion:** Demonstrating engagement with all content

Grading: How Your Learning Translates to Grades

While assessment focuses on learning, grades provide a summary of your demonstrated mastery. The grading schema and Late Policy and Extensions is the standard MSDS practice. Here is the breakdown for the total grade:

Participation Breakdown (20%)

- **Module Quiz Completion (10%):** 12 quizzes, full credit for thoughtful completion
- **Discussion Contributions (10%):** Quality posts, peer responses, Project Peer Feedback, and discussion leadership

Component	Weight	Details
Programming Assignments	40%	5 assignments × 8% each
Group Project	40%	Proposal (5%), Mid-checkpoint (10%), Final deliverables (25%)
Participation	20%	See detailed breakdown.

Professional and Academic Integrity

As future leaders in data science and AI, uphold the highest standards of ethics and integrity. All work must comply with the UVA Honor System. Individual assignments must be your own original work, though conceptual discussions are encouraged. Any external code, ideas, or resources must be appropriately cited. The pledge should be included with relevant submissions.