**Comprehensive ERA V4-Inspired AI Self-Study Plan**

**Course Overview**

This 26-week intensive self-study program mirrors the progression and depth of The School of AI's ERA V4 course, using entirely free resources. The plan progresses from neural network fundamentals to training 70B+ parameter models, emphasizing hands-on implementation and real-world deployment skills.

**Prerequisites:** Basic Python programming, high school mathematics  
**Duration:** 26 weeks (6 months intensive)  
**Study Time:** 15-25 hours per week  
**Hardware Requirements:** Access to Google Colab (free GPU/TPU) or cloud computing credits

**PHASE 1: FOUNDATIONS & COMPUTER VISION**

*Weeks 1-12: Building Deep Learning Fundamentals*

**Week 1: AI Foundations & Neural Networks**

**Difficulty:** Beginner  
**Study Time:** 15 hours

**Learning Objectives:**

* Understand neural network fundamentals and backpropagation
* Set up development environment (Python, PyTorch, Git)
* Implement first neural network from scratch

**Theory Resources:**

* [3Blue1Brown Neural Networks Series](https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi) - Visual neural network explanations (4 hours)
* [Fast.ai Lesson 1](https://course.fast.ai/Lessons/lesson1.html) - Practical deep learning introduction (2 hours)
* [MIT 6.034 Neural Networks Lecture](https://ocw.mit.edu/courses/6-034-artificial-intelligence-fall-2010/video_galleries/lecture-videos/) - Theoretical foundations (2 hours)

**Practical Resources:**

* [Google Colab Neural Network from Scratch](https://colab.research.google.com/github/lexfridman/mit-deep-learning/blob/master/tutorial_deep_learning_basics/deep_learning_basics.ipynb) - Implementation tutorial
* [PyTorch 60 Minute Blitz](https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html) - Framework introduction

**Hands-on Assignment:**

* Implement multi-layer perceptron from scratch using NumPy for XOR problem
* Build same network using PyTorch and compare performance
* Deploy simple neural network using Gradio interface
* **Code Repository:** Create personal GitHub repo for all implementations

**Optional Advanced Materials:**

* Ian Goodfellow's "Deep Learning" Chapter 1-2 (mathematical foundations)

**Week 2: Python Mastery & Development Tools**

**Difficulty:** Beginner-Intermediate  
**Study Time:** 12 hours

**Learning Objectives:**

* Master Python for ML (NumPy, Matplotlib, advanced concepts)
* Learn Git version control and collaborative development
* Set up professional ML development environment

**Theory Resources:**

* [Real Python Advanced Python Features](https://realpython.com/learning-paths/python-fundamentals/) - Object-oriented programming, decorators, generators (4 hours)
* [Git and GitHub Tutorial](https://www.youtube.com/watch?v=RGOj5yH7evk) - Version control fundamentals (2 hours)

**Practical Resources:**

* [NumPy Tutorial for Machine Learning](https://numpy.org/learn/) - Array operations and vectorization
* [Matplotlib Gallery](https://matplotlib.org/stable/gallery/index.html) - Data visualization techniques
* [VS Code for Data Science Setup](https://code.visualstudio.com/docs/datascience/jupyter-notebooks) - IDE configuration

**Hands-on Assignment:**

* Create comprehensive data analysis pipeline using NumPy and Matplotlib
* Implement vector operations for neural network computations
* Set up GitHub repository with proper documentation and code organization
* Practice Git workflow: branching, merging, pull requests

**Week 3: PyTorch Deep Dive & Cloud Computing**

**Difficulty:** Intermediate  
**Study Time:** 18 hours

**Learning Objectives:**

* Master PyTorch tensor operations and automatic differentiation
* Set up and use cloud computing for ML (Google Colab, Kaggle)
* Understand GPU acceleration and memory management

**Theory Resources:**

* [PyTorch Tutorials Official](https://pytorch.org/tutorials/) - Complete PyTorch fundamentals (6 hours)
* [Google Colab Guide](https://colab.research.google.com/notebooks/intro.ipynb) - Cloud computing setup (2 hours)
* [Understanding GPU Computing for ML](https://course.fast.ai/Lessons/lesson8.html) - Hardware acceleration concepts

**Practical Resources:**

* [PyTorch Examples Repository](https://github.com/pytorch/examples) - Official implementations
* [Google Colab Pro Features](https://research.google.com/colaboratory/faq.html) - Advanced cloud features

**Hands-on Assignment:**

* Implement gradient descent optimization from scratch using PyTorch
* Create custom Dataset and DataLoader for image classification
* Compare training times on CPU vs GPU using Google Colab
* Build and train basic CNN on CIFAR-10 dataset
* **Project:** Deploy trained model as web service using Streamlit

**Week 4: First Neural Network & Cloud Training**

**Difficulty:** Intermediate  
**Study Time:** 20 hours

**Learning Objectives:**

* Train production-scale neural networks on cloud platforms
* Implement proper training loops with validation and testing
* Master data preprocessing and augmentation techniques

**Theory Resources:**

* [Deep Learning Specialization Course 2](https://www.coursera.org/learn/deep-neural-network) - Optimization and hyperparameter tuning (audit for free)
* [Fast.ai Lesson 2-3](https://course.fast.ai/) - Training methodology and data augmentation

**Practical Resources:**

* [PyTorch Training Loop Tutorial](https://pytorch.org/tutorials/beginner/introyt/trainingyt.html) - Professional training practices
* [Albumentations Documentation](https://albumentations.ai/) - Advanced data augmentation

**Hands-on Assignment:**

* Implement complete training pipeline with train/validation/test splits
* Train ResNet-18 on CIFAR-10 with data augmentation
* Implement learning rate scheduling and early stopping
* Create comprehensive training visualizations and metrics logging
* **Project:** Achieve >92% accuracy on CIFAR-10 and deploy to Hugging Face Spaces

**Week 5: CNNs & Backpropagation Deep Dive**

**Difficulty:** Intermediate  
**Study Time:** 20 hours

**Learning Objectives:**

* Understand CNN architecture design principles
* Master backpropagation mathematics and implementation
* Implement convolutional layers from scratch

**Theory Resources:**

* [Stanford CS231n CNN Lectures](http://cs231n.stanford.edu/schedule.html) - Lectures 5-7 on CNNs and backpropagation (6 hours)
* [Convolution Arithmetic Guide](https://github.com/vdumoulin/conv_arithmetic) - Visual convolution explanations
* [Backpropagation Calculus](https://www.youtube.com/watch?v=Ilg3gGewQ5U) - Mathematical foundations by 3Blue1Brown

**Practical Resources:**

* [CNN Implementation from Scratch](https://github.com/pytorch/tutorials/blob/main/beginner_source/nn_tutorial.py) - PyTorch tutorial
* [CS231n Assignment Solutions](https://github.com/amanchadha/coursera-deep-learning-specialization) - Complete implementations

**Hands-on Assignment:**

* Implement convolutional layer with forward and backward pass from scratch
* Build CNN architecture with different layer types (conv, pooling, batch norm)
* Visualize feature maps and learned filters
* Compare custom implementation with PyTorch's built-in layers
* **Project:** Create interactive CNN visualizer using Streamlit

**Week 6: Advanced CNN Architectures**

**Difficulty:** Intermediate-Advanced  
**Study Time:** 22 hours

**Learning Objectives:**

* Implement state-of-the-art CNN architectures (ResNet, DenseNet, EfficientNet)
* Understand architectural innovations and design patterns
* Master transfer learning and fine-tuning techniques

**Theory Resources:**

* [ResNet Paper](https://arxiv.org/abs/1512.03385) - Original residual networks paper
* [EfficientNet Paper](https://arxiv.org/abs/1905.11946) - Compound scaling methodology
* [Papers with Code CNN Architectures](https://paperswithcode.com/methods/category/convolutional-neural-networks) - Architecture comparisons

**Practical Resources:**

* [Timm Models Library](https://github.com/rwightman/pytorch-image-models) - Pre-trained CNN implementations
* [Transfer Learning Tutorial](https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html) - Fine-tuning practices

**Hands-on Assignment:**

* Implement ResNet from scratch with skip connections
* Compare ResNet, DenseNet, and EfficientNet on ImageNet subset
* Fine-tune pre-trained models on custom dataset
* Create model comparison framework with performance metrics
* **Project:** Build image classification API with multiple model options

**Week 7: Training Optimization & Regularization**

**Difficulty:** Advanced  
**Study Time:** 20 hours

**Learning Objectives:**

* Master advanced optimization techniques (Adam, RMSprop, learning rate scheduling)
* Implement regularization methods (dropout, batch normalization, weight decay)
* Understand training dynamics and convergence analysis

**Theory Resources:**

* [Deep Learning Book Chapter 8](https://www.deeplearningbook.org/contents/optimization.html) - Optimization for machine learning
* [Batch Normalization Paper](https://arxiv.org/abs/1502.03167) - Understanding batch normalization
* [Adam Optimizer Paper](https://arxiv.org/abs/1412.6980) - Adaptive moment estimation

**Practical Resources:**

* [PyTorch Optimizers Documentation](https://pytorch.org/docs/stable/optim.html) - Optimizer implementations
* [Learning Rate Finder](https://github.com/davidtvs/pytorch-lr-finder) - Optimal learning rate discovery

**Hands-on Assignment:**

* Implement different optimizers from scratch and compare convergence
* Create learning rate scheduling experiments
* Study effect of batch normalization on training dynamics
* Build comprehensive training monitoring dashboard
* **Project:** Create training optimization toolkit with automated hyperparameter search

**Week 8: One Cycle Policy & Advanced Training**

**Difficulty:** Advanced  
**Study Time:** 18 hours

**Learning Objectives:**

* Implement One Cycle Learning Rate Policy for faster convergence
* Master gradient clipping and numerical stability techniques
* Understand training at scale considerations

**Theory Resources:**

* [Cyclical Learning Rates Paper](https://arxiv.org/abs/1506.01186) - Cyclic learning rate methodology
* [One Cycle Policy](https://arxiv.org/abs/1708.07120) - Super-convergence techniques
* [Fast.ai One Cycle](https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html) - Practical implementation guide

**Practical Resources:**

* [PyTorch One Cycle Scheduler](https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html) - Official implementation
* [Weights & Biases Tutorial](https://wandb.ai/site/articles/introduction-to-pytorch-lr-scheduler) - Learning rate visualization

**Hands-on Assignment:**

* Implement One Cycle Policy from scratch
* Compare training times: standard vs. one cycle learning
* Create learning rate range test implementation
* Build automated hyperparameter optimization pipeline
* **Project:** Train CIFAR-10 to 94% accuracy in under 100 epochs

**Week 9: Multi-GPU ImageNet Training**

**Difficulty:** Advanced  
**Study Time:** 25 hours

**Learning Objectives:**

* Set up distributed training across multiple GPUs
* Train ResNet on full ImageNet dataset from scratch
* Master data loading optimization and memory management

**Theory Resources:**

* [PyTorch Distributed Training](https://pytorch.org/tutorials/intermediate/ddp_tutorial.html) - Multi-GPU training setup
* [ImageNet Training Best Practices](https://arxiv.org/abs/1812.01187) - Scaling to large datasets
* [Mixed Precision Training](https://arxiv.org/abs/1710.03740) - Memory and speed optimization

**Practical Resources:**

* [ImageNet Training Example](https://github.com/pytorch/examples/tree/main/imagenet) - Complete training script
* [Fast.ai ImageNet Training](https://github.com/fastai/imagenette) - Optimized training pipeline

**Hands-on Assignment:**

* Set up multi-GPU training environment (use Google Colab Pro or Kaggle)
* Implement data loading optimizations for ImageNet
* Train ResNet-50 on ImageNet subset with proper validation
* Compare single vs. multi-GPU training performance
* **Project:** Achieve ImageNet baseline accuracy and document training process

**Week 10: Computer Vision Applications**

**Difficulty:** Intermediate-Advanced  
**Study Time:** 20 hours

**Learning Objectives:**

* Build object detection systems (YOLO, R-CNN family)
* Implement image segmentation models
* Create end-to-end computer vision applications

**Theory Resources:**

* [YOLO Paper](https://arxiv.org/abs/1506.02640) - Real-time object detection
* [Detectron2 Documentation](https://detectron2.readthedocs.io/) - Object detection framework
* [U-Net Paper](https://arxiv.org/abs/1505.04597) - Semantic segmentation

**Practical Resources:**

* [YOLOv5 Repository](https://github.com/ultralytics/yolov5) - Production-ready object detection
* [Detectron2 Tutorial](https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5) - Custom object detection training

**Hands-on Assignment:**

* Train YOLO on custom object detection dataset
* Implement semantic segmentation for medical images
* Create real-time object detection demo using webcam
* Build complete computer vision pipeline with preprocessing and postprocessing
* **Project:** Deploy object detection system as mobile-friendly web app

**Week 11: Generative Models Introduction**

**Difficulty:** Advanced  
**Study Time:** 22 hours

**Learning Objectives:**

* Understand generative modeling principles
* Implement Variational Autoencoders (VAEs)
* Introduction to Generative Adversarial Networks (GANs)

**Theory Resources:**

* [VAE Paper](https://arxiv.org/abs/1312.6114) - Auto-encoding variational Bayes
* [GAN Paper](https://arxiv.org/abs/1406.2661) - Generative adversarial networks
* [Lil'Log Generative Models](https://lilianweng.github.io/posts/2018-10-13-flow-models/) - Comprehensive overview

**Practical Resources:**

* [VAE PyTorch Tutorial](https://github.com/pytorch/examples/tree/main/vae) - Official implementation
* [GAN PyTorch Tutorial](https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html) - DCGAN implementation

**Hands-on Assignment:**

* Implement VAE for image generation on MNIST
* Train DCGAN on CelebA face dataset
* Create latent space interpolation visualizations
* Compare different generative approaches
* **Project:** Build interactive generative art application

**Week 12: Computer Vision Capstone**

**Difficulty:** Advanced  
**Study Time:** 25 hours

**Learning Objectives:**

* Integrate all computer vision techniques learned
* Build production-ready computer vision system
* Master deployment and scalability considerations

**Hands-on Assignment:**

* **Capstone Project Options:**
  1. **Smart City Traffic Analysis**: Real-time vehicle detection, counting, and traffic flow analysis
  2. **Medical Image Analysis System**: Disease detection from medical scans with explainable AI
  3. **Agricultural Monitoring**: Crop health assessment using satellite/drone imagery
  4. **Retail Analytics Platform**: Customer behavior analysis and product recognition

**Project Requirements:**

* Custom dataset creation and labeling
* Multiple model comparison and ensemble methods
* Real-time inference optimization
* Web application with API endpoints
* Comprehensive documentation and testing
* Deployment to cloud platform (Heroku, AWS, or Google Cloud)

**Evaluation Metrics:**

* Model performance on test dataset
* Inference speed and memory efficiency
* Code quality and documentation
* User interface and experience
* Scalability demonstration

**PHASE 2: TRANSFORMERS & LARGE LANGUAGE MODELS**

*Weeks 13-18: Modern NLP and Transformer Architectures*

**Week 13: Transformer Architecture Deep Dive**

**Difficulty:** Advanced  
**Study Time:** 25 hours

**Learning Objectives:**

* Understand attention mechanisms and self-attention
* Implement Transformer architecture from scratch
* Master positional encoding and multi-head attention

**Theory Resources:**

* [Attention Is All You Need Paper](https://arxiv.org/abs/1706.03762) - Original Transformer paper (3 hours)
* [The Illustrated Transformer](https://jalammar.github.io/illustrated-transformer/) - Visual explanation (2 hours)
* [3Blue1Brown Attention](https://www.youtube.com/watch?v=eMlx5fFNoYc) - Mathematical intuition (1 hour)
* [Harvard NLP Annotated Transformer](http://nlp.seas.harvard.edu/annotated-transformer/) - Line-by-line implementation

**Practical Resources:**

* [Hugging Face Transformers Course](https://huggingface.co/learn/nlp-course/chapter1/1) - Comprehensive transformer tutorial
* [PyTorch Transformer Tutorial](https://pytorch.org/tutorials/beginner/transformer_tutorial.html) - Official implementation

**Hands-on Assignment:**

* Implement multi-head attention mechanism from scratch
* Build complete Transformer encoder-decoder architecture
* Train Transformer on machine translation task (English-French)
* Create attention visualization tools
* **Project:** Deploy translation model with interactive web interface

**Week 14: Embeddings & Tokenization**

**Difficulty:** Intermediate-Advanced  
**Study Time:** 20 hours

**Learning Objectives:**

* Master different tokenization strategies (BPE, WordPiece, SentencePiece)
* Understand contextual vs. static embeddings
* Implement embedding techniques and similarity measures

**Theory Resources:**

* [Word2Vec Paper](https://arxiv.org/abs/1301.3781) - Efficient estimation of word representations
* [BPE Paper](https://arxiv.org/abs/1508.07909) - Subword tokenization
* [SentencePiece](https://arxiv.org/abs/1808.06226) - Unsupervised text tokenizer

**Practical Resources:**

* [Hugging Face Tokenizers Library](https://huggingface.co/docs/tokenizers/python/latest/) - Fast tokenization implementations
* [Word Embeddings Tutorial](https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html) - PyTorch implementation

**Hands-on Assignment:**

* Train Word2Vec and FastText embeddings on large corpus
* Implement different tokenization strategies and compare vocabulary sizes
* Create word analogy and similarity tasks evaluation
* Build semantic search system using embeddings
* **Project:** Multilingual document similarity search engine

**Week 15: Large Language Model Training**

**Difficulty:** Advanced  
**Study Time:** 28 hours

**Learning Objectives:**

* Understand LLM training pipeline and data preparation
* Implement GPT architecture from scratch
* Master training techniques for large-scale language models

**Theory Resources:**

* [GPT Paper](https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf) - Improving language understanding by generative pre-training
* [GPT-2 Paper](https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf) - Language models are unsupervised multitask learners
* [Training Compute-Optimal Large Language Models](https://arxiv.org/abs/2203.15556) - Chinchilla scaling laws

**Practical Resources:**

* [nanoGPT](https://github.com/karpathy/nanoGPT) - Minimal GPT implementation by Andrej Karpathy
* [GPT-2 from Scratch](https://www.youtube.com/watch?v=kCc8FmEb1nY) - Complete implementation tutorial
* [Hugging Face Transformers Training](https://huggingface.co/docs/transformers/training) - Large model training guide

**Hands-on Assignment:**

* Implement GPT architecture from scratch following nanoGPT
* Train medium-sized language model on curated text corpus
* Implement text generation with different sampling strategies
* Create model evaluation metrics (perplexity, BLEU)
* **Project:** Train domain-specific language model (code, literature, or science)

**Week 16: LLM Optimization & Evaluation**

**Difficulty:** Advanced  
**Study Time:** 25 hours

**Learning Objectives:**

* Master gradient accumulation and mixed precision training
* Implement model parallelism for large models
* Understand LLM evaluation benchmarks and metrics

**Theory Resources:**

* [Megatron-LM Paper](https://arxiv.org/abs/1909.08053) - Training multi-billion parameter language models
* [Efficient Large-Scale Language Model Training](https://arxiv.org/abs/2104.04473) - ZeRO optimizer states partitioning
* [HELM Paper](https://arxiv.org/abs/2211.09110) - Holistic evaluation of language models

**Practical Resources:**

* [DeepSpeed Tutorial](https://www.deepspeed.ai/tutorials/) - Memory and compute optimization
* [Accelerate Library](https://huggingface.co/docs/accelerate/index) - Multi-GPU training simplified
* [OpenAI Evals](https://github.com/openai/evals) - LLM evaluation framework

**Hands-on Assignment:**

* Implement gradient accumulation and mixed precision training
* Set up model parallelism for training larger models
* Create comprehensive LLM evaluation suite
* Benchmark model performance on standard tasks (GLUE, SuperGLUE)
* **Project:** Optimize training pipeline for maximum efficiency and create evaluation dashboard

**Week 17: Fine-tuning & Instruction Following**

**Difficulty:** Advanced  
**Study Time:** 24 hours

**Learning Objectives:**

* Master supervised fine-tuning techniques
* Implement instruction tuning and prompt engineering
* Understand parameter-efficient fine-tuning (LoRA, AdaLoRA)

**Theory Resources:**

* [InstructGPT Paper](https://arxiv.org/abs/2203.02155) - Training language models to follow instructions with human feedback
* [LoRA Paper](https://arxiv.org/abs/2106.09685) - Low-rank adaptation of large language models
* [Parameter-Efficient Transfer Learning](https://arxiv.org/abs/1902.00751) - Adapter modules

**Practical Resources:**

* [PEFT Library](https://github.com/huggingface/peft) - Parameter-efficient fine-tuning methods
* [Alpaca Training](https://github.com/tatsu-lab/stanford_alpaca) - Instruction-following fine-tuning
* [QLoRA Paper](https://arxiv.org/abs/2305.14314) - Efficient finetuning of quantized LLMs

**Hands-on Assignment:**

* Fine-tune pre-trained language model on instruction-following dataset
* Implement LoRA and compare with full fine-tuning
* Create custom instruction dataset for specific domain
* Build chat interface for instruction-tuned model
* **Project:** Create specialized assistant (coding, writing, or domain expert) with custom fine-tuning

**Week 18: Quantization & Model Compression**

**Difficulty:** Advanced  
**Study Time:** 22 hours

**Learning Objectives:**

* Understand quantization techniques (post-training and quantization-aware training)
* Implement model pruning and knowledge distillation
* Master deployment of compressed models

**Theory Resources:**

* [Quantization and Training of Neural Networks](https://arxiv.org/abs/1712.05877) - Comprehensive quantization survey
* [Knowledge Distillation Paper](https://arxiv.org/abs/1503.02531) - Distilling knowledge in neural networks
* [GPTQ Paper](https://arxiv.org/abs/2210.17323) - Accurate post-training quantization for GPT models

**Practical Resources:**

* [PyTorch Quantization](https://pytorch.org/docs/stable/quantization.html) - Official quantization toolkit
* [BitsAndBytes](https://github.com/TimDettmers/bitsandbytes) - 8-bit optimizers and quantization
* [AutoGPTQ](https://github.com/PanQiWei/AutoGPTQ) - Quantization toolkit for LLMs

**Hands-on Assignment:**

* Implement post-training quantization (PTQ) and quantization-aware training (QAT)
* Compare model performance vs. compression trade-offs
* Create knowledge distillation pipeline to compress large models
* Deploy quantized models for edge inference
* **Project:** Create model compression benchmark suite comparing different techniques

**PHASE 3: ADVANCED AI APPLICATIONS**

*Weeks 19-24: Multimodal AI, RL, and Production Systems*

**Week 19: Vision-Language Models**

**Difficulty:** Advanced  
**Study Time:** 26 hours

**Learning Objectives:**

* Understand multimodal learning and cross-modal attention
* Implement CLIP-style contrastive learning
* Build vision-language applications

**Theory Resources:**

* [CLIP Paper](https://arxiv.org/abs/2103.00020) - Learning transferable visual models from natural language supervision
* [DALL-E Paper](https://arxiv.org/abs/2102.12092) - Zero-shot text-to-image generation
* [Flamingo Paper](https://arxiv.org/abs/2204.14198) - Few-shot learning with frozen language models

**Practical Resources:**

* [OpenCLIP Implementation](https://github.com/mlfoundations/open_clip) - Open source CLIP training
* [LAION Dataset](https://laion.ai/) - Large-scale image-text pairs
* [Hugging Face Vision-Language Models](https://huggingface.co/models?pipeline_tag=visual-question-answering) - Pre-trained VLMs

**Hands-on Assignment:**

* Implement CLIP-style contrastive learning from scratch
* Train vision-language model on curated image-text dataset
* Build zero-shot image classification using text prompts
* Create visual question answering system
* **Project:** Multimodal search engine with text and image queries

**Week 20: Reinforcement Learning Fundamentals**

**Difficulty:** Intermediate-Advanced  
**Study Time:** 24 hours

**Learning Objectives:**

* Master RL problem formulation (MDPs, value functions, policies)
* Implement Q-learning and policy gradient methods
* Understand exploration vs. exploitation trade-offs

**Theory Resources:**

* [Sutton & Barto RL Book](http://incompleteideas.net/book/the-book-2nd.html) - Chapters 1-6 (free online textbook)
* [David Silver RL Course](https://www.youtube.com/playlist?list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ) - DeepMind RL lectures
* [OpenAI Spinning Up](https://spinningup.openai.com/en/latest/) - Deep RL introduction

**Practical Resources:**

* [Stable Baselines3](https://stable-baselines3.readthedocs.io/) - High-quality RL implementations
* [OpenAI Gymnasium](https://gymnasium.farama.org/) - RL environment toolkit
* [CleanRL](https://github.com/vwxyzjn/cleanrl) - Single-file RL implementations

**Hands-on Assignment:**

* Implement Q-learning and SARSA for gridworld environment
* Train DQN on Atari games using Stable Baselines3
* Create custom RL environment using Gymnasium interface
* Compare different exploration strategies
* **Project:** Multi-agent RL system (trading, game-playing, or robotics simulation)

**Week 21: Advanced Reinforcement Learning**

**Difficulty:** Advanced  
**Study Time:** 25 hours

**Learning Objectives:**

* Implement policy gradient methods (REINFORCE, Actor-Critic, PPO)
* Understand continuous action spaces and control problems
* Master advanced RL algorithms (SAC, TD3, DDPG)

**Theory Resources:**

* [Policy Gradient Methods Paper](https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf) - Theoretical foundations
* [PPO Paper](https://arxiv.org/abs/1707.06347) - Proximal policy optimization
* [SAC Paper](https://arxiv.org/abs/1801.01290) - Soft actor-critic

**Practical Resources:**

* [PPO Implementation](https://github.com/openai/baselines/tree/master/baselines/ppo2) - OpenAI baseline implementation
* [PyBullet Environments](https://pybullet.org/) - Physics simulation for continuous control
* [Ray RLLib](https://docs.ray.io/en/latest/rllib/index.html) - Distributed RL training

**Hands-on Assignment:**

* Implement PPO from scratch for continuous control tasks
* Train agents on MuJoCo/PyBullet robotic control tasks
* Create multi-agent reinforcement learning scenario
* Compare sample efficiency of different algorithms
* **Project:** Autonomous driving or robot navigation using deep RL

**Week 22: RLHF & AI Alignment**

**Difficulty:** Advanced  
**Study Time:** 23 hours

**Learning Objectives:**

* Understand reinforcement learning from human feedback (RLHF)
* Implement reward modeling and preference learning
* Master constitutional AI and AI safety techniques

**Theory Resources:**

* [InstructGPT Paper](https://arxiv.org/abs/2203.02155) - Training language models to follow instructions with human feedback
* [Constitutional AI Paper](https://arxiv.org/abs/2212.08073) - Training a helpful and harmless assistant
* [Reward Modeling Paper](https://arxiv.org/abs/1909.12917) - Fine-tuning language models from human preferences

**Practical Resources:**

* [TRL Library](https://github.com/huggingface/trl) - Transformer reinforcement learning
* [Open Assistant Dataset](https://huggingface.co/datasets/OpenAssistant/oasst1) - Human feedback conversations
* [RLHF Implementation](https://github.com/CarperAI/trlx) - Training with human feedback

**Hands-on Assignment:**

* Implement reward model training using human preference data
* Create RLHF pipeline for instruction-following fine-tuning
* Build human feedback collection interface
* Compare RLHF with supervised fine-tuning
* **Project:** Safe AI assistant with constitutional training and red teaming evaluation

**Week 23: Large-Scale Model Training**

**Difficulty:** Advanced  
**Study Time:** 30 hours

**Learning Objectives:**

* Implement data parallelism and model parallelism for 70B+ models
* Master gradient checkpointing and memory optimization
* Understand training infrastructure and monitoring

**Theory Resources:**

* [Megatron-LM Paper](https://arxiv.org/abs/1909.08053) - Training multi-billion parameter models
* [GPT-3 Paper](https://arxiv.org/abs/2005.14165) - Language models are few-shot learners
* [PaLM Paper](https://arxiv.org/abs/2204.02311) - Pathways language model scaling

**Practical Resources:**

* [Megatron-DeepSpeed](https://github.com/microsoft/Megatron-DeepSpeed) - Large-scale training framework
* [FairScale](https://github.com/facebookresearch/fairscale) - PyTorch extensions for large-scale training
* [Colossal-AI](https://github.com/hpcaitech/ColossalAI) - Making large AI models cheaper and more accessible

**Hands-on Assignment:**

* Set up distributed training environment for multi-node training
* Implement gradient accumulation and pipeline parallelism
* Train 7B parameter model using model parallelism techniques
* Create comprehensive training monitoring and logging
* **Project:** End-to-end large model training pipeline with infrastructure automation

**Week 24: Production AI Systems**

**Difficulty:** Advanced  
**Study Time:** 25 hours

**Learning Objectives:**

* Master MLOps best practices for production AI systems
* Implement model serving, monitoring, and automated retraining
* Build scalable AI infrastructure with containerization and orchestration

**Theory Resources:**

* [MLOps Principles](https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning) - Google Cloud MLOps guide
* [Hidden Technical Debt in ML Systems](https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf) - Production ML challenges
* [ML Model Serving Patterns](https://martinfowler.com/articles/ml-inference-patterns.html) - Deployment architecture patterns

**Practical Resources:**

* [MLOps Zoomcamp](https://github.com/DataTalksClub/mlops-zoomcamp) - Free comprehensive MLOps course
* [BentoML](https://github.com/bentoml/BentoML) - Model serving framework
* [Seldon Core](https://github.com/SeldonIO/seldon-core) - ML deployment on Kubernetes

**Hands-on Assignment:**

* Create complete MLOps pipeline with CI/CD for model deployment
* Implement model monitoring and drift detection systems
* Build API gateway for multiple model serving with load balancing
* Create automated model retraining pipeline
* **Project:** Production-ready AI service with monitoring, logging, and automated scaling

**PHASE 4: CAPSTONE PROJECT**

*Weeks 25-26: Integration and Portfolio Development*

**Week 25-26: Comprehensive Capstone**

**Difficulty:** Expert Level  
**Study Time:** 40+ hours

**Learning Objectives:**

* Integrate all learned techniques into comprehensive AI system
* Demonstrate production-ready implementation skills
* Create portfolio-worthy project with real-world impact

**Capstone Project Options:**

**Option 1: Multimodal AI Assistant**

Build a comprehensive AI assistant that can:

* Process text, images, and voice inputs
* Generate text, images, and speech outputs
* Maintain conversation context and user preferences
* Learn and adapt from user feedback

**Technical Requirements:**

* Custom transformer architecture with multimodal fusion
* RLHF training pipeline for alignment
* Production deployment with API endpoints
* Real-time inference optimization
* Comprehensive evaluation and safety testing

**Option 2: Large Language Model from Scratch**

Train a domain-specific large language model:

* Custom dataset curation and preprocessing
* Distributed training on cloud infrastructure
* Complete pretraining and instruction-tuning pipeline
* Quantization and optimization for deployment
* Evaluation on domain-specific benchmarks

**Technical Requirements:**

* 7B+ parameter model architecture
* Multi-GPU/multi-node training setup
* Custom tokenizer and preprocessing pipeline
* RLHF fine-tuning implementation
* Production serving infrastructure

**Option 3: AI-Powered Content Creation Platform**

Create comprehensive content generation system:

* Text generation with style and tone control
* Image generation with text prompts
* Video summarization and highlights
* Multi-language support with translation
* Content moderation and safety filters

**Technical Requirements:**

* Multiple generative models integration
* Real-time content generation API
* User interface with advanced controls
* Content quality evaluation metrics
* Scalable cloud deployment

**Final Deliverables:**

1. **Complete codebase** with documentation on GitHub
2. **Technical report** (15-20 pages) documenting architecture, experiments, and results
3. **Live demo deployment** accessible via web interface
4. **Video presentation** (10-15 minutes) explaining project and results
5. **Performance benchmarks** comparing to existing solutions
6. **Open source release** with tutorial and setup instructions

**Evaluation Criteria:**

* **Technical Innovation:** Novel approaches and implementations (25%)
* **Code Quality:** Clean, documented, tested code (20%)
* **Performance:** Quantitative results and benchmarks (20%)
* **Production Readiness:** Deployment, monitoring, scalability (20%)
* **Impact Potential:** Real-world applicability and usefulness (15%)

**RESOURCES SUMMARY**

**Primary Learning Platforms**

* **Fast.ai Course**: Practical deep learning methodology
* **3Blue1Brown**: Mathematical intuition and visualization
* **Hugging Face**: Transformer models and modern NLP
* **Papers with Code**: Latest research with implementations
* **Google Colab**: Free GPU/TPU access for training
* **Kaggle**: Datasets, competitions, and community notebooks

**Essential Tools & Frameworks**

* **PyTorch**: Primary deep learning framework
* **Hugging Face Transformers**: Pre-trained models and tokenizers
* **Weights & Biases**: Experiment tracking and visualization
* **Docker**: Containerization for reproducible environments
* **Git/GitHub**: Version control and code sharing
* **Streamlit/Gradio**: Quick deployment and demonstration

**Recommended Study Schedule**

* **Weekdays (Mon-Fri):** 2-3 hours daily (theory, reading, small coding exercises)
* **Weekends:** 8-10 hours (major implementations, projects, assignments)
* **Total:** 20-25 hours per week average
* **Peak weeks:** 30+ hours during advanced topics and capstone

**Success Metrics & Milestones**

* **Weeks 1-6:** Complete computer vision fundamentals with deployable projects
* **Weeks 7-12:** Advanced CV techniques with production-quality implementations
* **Weeks 13-18:** Transformer mastery with custom LLM training
* **Weeks 19-24:** Multimodal AI and production systems expertise
* **Weeks 25-26:** Capstone project demonstrating integration of all skills

This comprehensive 26-week program provides the depth and breadth of the ERA V4 course using entirely free resources, with emphasis on hands-on implementation and production-ready skills. Each week builds upon previous knowledge while introducing increasingly sophisticated concepts, culminating in a portfolio-worthy capstone project that demonstrates mastery of modern AI techniques.

**ADVANCED SUPPLEMENTARY TRACKS**

*Specialized Extensions for Deep AI Mastery*

The following supplementary tracks can be pursued in parallel with the main curriculum or as post-completion specializations. Each track is designed to be modular and can be integrated based on your interests and career goals.

**TRACK A: STANFORD CS236 DEEP GENERATIVE MODELS**

*Duration: 12 weeks parallel track or 8 weeks intensive*

**Module A1: Fundamentals of Generative Modeling (Weeks 1-2)**

**Study Time:** 15 hours per week  
**Prerequisites:** Complete through Week 11 of main curriculum

**Learning Objectives:**

* Master probability theory foundations for generative models
* Understand maximum likelihood estimation and variational inference
* Implement basic generative models from first principles

**Theory Resources:**

* [Stanford CS236 Lectures 1-4](https://deepgenerativemodels.github.io/syllabus.html) - Foundations and autoregressive models (8 hours)
* [Pattern Recognition and Machine Learning](http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf) - Chapters 2-3 on probability (6 hours)
* [Variational Inference Review](https://arxiv.org/abs/1601.00670) - Modern variational methods

**Practical Resources:**

* [CS236 Official Assignments](https://github.com/deepgenerativemodels/notes) - Jupyter notebooks with implementations
* [PyTorch Distributions](https://pytorch.org/docs/stable/distributions.html) - Probability distributions in PyTorch

**Hands-on Assignment:**

* Implement Gaussian Mixture Models with EM algorithm from scratch
* Build autoregressive models (MADE, PixelCNN) for image generation
* Create variational inference framework for Bayesian neural networks
* **Project:** Generative model for time series forecasting with uncertainty quantification

**Module A2: Variational Autoencoders & Flow Models (Weeks 3-4)**

**Study Time:** 18 hours per week

**Learning Objectives:**

* Master VAE mathematics and implement β-VAE variants
* Understand normalizing flows and invertible transformations
* Implement Real NVP and coupling layers

**Theory Resources:**

* [Stanford CS236 Lectures 5-8](https://deepgenerativemodels.github.io/syllabus.html) - VAEs and flow models (8 hours)
* [Flow-based Deep Generative Models](https://arxiv.org/abs/1912.02762) - Comprehensive survey
* [Real NVP Paper](https://arxiv.org/abs/1605.08803) - Density estimation using real-valued non-volume preserving transformations

**Practical Resources:**

* [Pyro Tutorials](http://pyro.ai/examples/) - Probabilistic programming for VAEs
* [Normalizing Flows Library](https://github.com/bayesiains/nsf) - Neural spline flows implementation

**Hands-on Assignment:**

* Implement β-VAE with controllable disentanglement
* Build Real NVP for density modeling on complex datasets
* Create conditional flow models for controlled generation
* Compare VAE vs. Flow model trade-offs on CelebA dataset
* **Project:** Molecular generation using graph VAEs and flows

**Module A3: Advanced GANs & Score-Based Models (Weeks 5-7)**

**Study Time:** 20 hours per week

**Learning Objectives:**

* Master GAN training dynamics and advanced architectures
* Understand score matching and diffusion probabilistic models
* Implement state-of-the-art generative models (StyleGAN, DDPM)

**Theory Resources:**

* [Stanford CS236 Lectures 9-12](https://deepgenerativemodels.github.io/syllabus.html) - GANs and score-based models (10 hours)
* [Denoising Diffusion Probabilistic Models](https://arxiv.org/abs/2006.11239) - DDPM paper
* [Score-Based Generative Models](https://arxiv.org/abs/2011.13456) - Yang Song's comprehensive survey

**Practical Resources:**

* [StyleGAN2 Implementation](https://github.com/NVlabs/stylegan2-ada-pytorch) - Official NVIDIA implementation
* [Diffusers Library](https://github.com/huggingface/diffusers) - Hugging Face diffusion models
* [Score SDE](https://github.com/yang-song/score_sde_pytorch) - Score-based generative modeling

**Hands-on Assignment:**

* Train StyleGAN2 on high-resolution custom dataset
* Implement DDPM from scratch with noise scheduling
* Create classifier-free guidance for controlled generation
* Build score-based model with SDE formulation
* **Project:** Text-to-image generation using diffusion models with custom conditioning

**Module A4: Generative Models Applications (Weeks 8-9)**

**Study Time:** 22 hours per week

**Learning Objectives:**

* Apply generative models to scientific and industrial problems
* Master evaluation metrics for generative models
* Build production-ready generative AI systems

**Theory Resources:**

* [Generative Models for Drug Discovery](https://arxiv.org/abs/1903.04388) - Applications in chemistry
* [FID and Inception Score](https://arxiv.org/abs/1801.01973) - Evaluation metrics
* [Energy-Based Models](https://arxiv.org/abs/1903.08689) - EBM training and applications

**Practical Resources:**

* [Molecule Generation Toolkit](https://github.com/deepchem/deepchem) - Chemical generative models
* [FID Score Implementation](https://github.com/mseitzer/pytorch-fid) - Standard evaluation metric
* [Clean FID](https://github.com/GaParmar/clean-fid) - Improved FID computation

**Hands-on Assignment:**

* Build molecular generation system with property optimization
* Create comprehensive evaluation framework for generative models
* Implement energy-based models for anomaly detection
* Deploy generative model API with quality filtering
* **Project:** Multi-modal generative system (text-to-image-to-3D pipeline)

**TRACK B: CAUSAL ARTIFICIAL INTELLIGENCE**

*Duration: 10 weeks parallel track or 7 weeks intensive*

**Module B1: Causal Inference Fundamentals (Weeks 1-2)**

**Study Time:** 16 hours per week  
**Prerequisites:** Basic statistics and probability theory

**Learning Objectives:**

* Master causal inference theory and Pearl's causal hierarchy
* Understand directed acyclic graphs (DAGs) and causal assumptions
* Implement do-calculus and identify causal effects

**Theory Resources:**

* [The Book of Why](https://www.goodreads.com/book/show/36204378-the-book-of-why) - Judea Pearl's introduction to causality (8 hours reading)
* [Causal Inference: The Mixtape](https://mixtape.scunning.com/) - Scott Cunningham's free online textbook (6 hours)
* [Brady Neal's Causal Inference Course](https://www.bradyneal.com/causal-inference-course) - Video lectures 1-8 (8 hours)

**Practical Resources:**

* [DoWhy Library](https://github.com/py-why/dowhy) - Microsoft's causal inference library
* [CausalML](https://github.com/uber/causalml) - Uber's causal ML toolkit
* [Ananke](https://github.com/ananke-causal/ananke) - Causal inference with hidden variables

**Hands-on Assignment:**

* Implement backdoor criterion and front-door criterion from scratch
* Build causal graph discovery algorithms (PC algorithm, GES)
* Create synthetic datasets with known causal structure for validation
* Use DoWhy for causal effect estimation on real observational data
* **Project:** Causal analysis of A/B testing data with confounding factors

**Module B2: Causal Discovery & Structure Learning (Weeks 3-4)**

**Study Time:** 18 hours per week

**Learning Objectives:**

* Implement constraint-based and score-based causal discovery algorithms
* Master causal discovery from time series and interventional data
* Understand limitations and assumptions of discovery methods

**Theory Resources:**

* [Causal Discovery Papers](https://github.com/rguo12/awesome-causality-algorithms) - Comprehensive algorithm collection
* [PC Algorithm Paper](https://arxiv.org/abs/math/0510436) - Constraint-based discovery
* [NOTEARS Paper](https://arxiv.org/abs/1803.01422) - Continuous optimization for structure learning

**Practical Resources:**

* [causal-learn](https://github.com/py-why/causal-learn) - Comprehensive causal discovery library
* [CASTLE](https://github.com/huawei-noah/trustworthyAI/tree/master/research/CausalStructureLearning/castle) - Causal structure learning toolkit
* [Tigramite](https://github.com/jakobrunge/tigramite) - Time series causal discovery

**Hands-on Assignment:**

* Implement PC algorithm and compare with NOTEARS on synthetic data
* Build causal discovery pipeline for time series data
* Create evaluation framework for structure learning algorithms
* Apply causal discovery to gene regulatory network inference
* **Project:** Automated causal discovery system with uncertainty quantification

**Module B3: Causal Representation Learning (Weeks 5-6)**

**Study Time:** 20 hours per week

**Learning Objectives:**

* Understand identifiable representation learning with causal constraints
* Implement causal VAEs and intervention-based representation learning
* Master causal disentanglement and invariant representations

**Theory Resources:**

* [Causal Representation Learning](https://arxiv.org/abs/2102.11107) - Comprehensive survey by Schölkopf et al.
* [β-VAE Paper](https://openreview.net/forum?id=Sy2fzU9gl) - Disentangled representation learning
* [Invariant Risk Minimization](https://arxiv.org/abs/1907.02893) - Domain generalization with causality

**Practical Resources:**

* [disentanglement\_lib](https://github.com/google-research/disentanglement_lib) - Google's disentanglement library
* [CausalVAE Implementation](https://github.com/huawei-noah/trustworthyAI/tree/master/research/CausalVAE) - Causal representation learning
* [InvariantRiskMinimization](https://github.com/facebookresearch/InvariantRiskMinimization) - IRM implementation

**Hands-on Assignment:**

* Implement causal VAE with structured latent space
* Build intervention-based representation learning system
* Create invariant risk minimization for domain adaptation
* Compare causal vs. standard representation learning on downstream tasks
* **Project:** Causal representation learning for robust computer vision

**Module B4: Causal Reinforcement Learning (Weeks 7-8)**

**Study Time:** 22 hours per week

**Learning Objectives:**

* Integrate causal reasoning with reinforcement learning
* Implement causal RL for transfer learning and generalization
* Master counterfactual reasoning in sequential decision making

**Theory Resources:**

* [Causal Reinforcement Learning](https://arxiv.org/abs/2106.02866) - Survey of causal RL methods
* [Causal Confusion in Imitation Learning](https://arxiv.org/abs/1905.11979) - Identifying spurious correlations
* [Counterfactual Data Augmentation](https://arxiv.org/abs/2106.07203) - CF-based RL improvements

**Practical Resources:**

* [CausalCity](https://github.com/huawei-noah/trustworthyAI/tree/master/research/CausalCity) - Causal RL benchmark
* [WhyNot](https://github.com/zykls/whynot) - Causal inference benchmark suite
* [Causal RL Environments](https://github.com/songrotek/CausalRL) - Custom environments for causal RL

**Hands-on Assignment:**

* Implement causal-aware policy learning with confounding
* Build transfer learning system using causal invariances
* Create counterfactual reasoning for RL safety
* Compare causal vs. standard RL on distribution shift tasks
* **Project:** Causal RL agent for robust autonomous navigation

**Module B5: Applications & Production Causal AI (Weeks 9-10)**

**Study Time:** 20 hours per week

**Learning Objectives:**

* Deploy causal AI systems in production environments
* Build causal ML pipelines for business decision making
* Master causal evaluation and A/B testing at scale

**Theory Resources:**

* [Causal Inference for Business](https://github.com/matheusfacure/python-causality-handbook) - Applied causal analysis
* [Experimentation Platform Design](https://arxiv.org/abs/1409.3174) - Large-scale A/B testing
* [Causal ML in Industry](https://blog.twitter.com/engineering/en_us/topics/insights/2021/causal-inference-at-twitter) - Real-world case studies

**Practical Resources:**

* [GrowthBook](https://github.com/growthbook/growthbook) - Open source experimentation platform
* [Statsig](https://docs.statsig.com/) - Feature flagging and experimentation
* [Netflix Experimentation](https://github.com/Netflix/experimentation) - Netflix's causal inference toolkit

**Hands-on Assignment:**

* Build end-to-end causal ML pipeline for recommendation systems
* Create automated causal analysis platform for marketing campaigns
* Implement continuous experimentation system with causal evaluation
* Deploy causal AI service with monitoring and validation
* **Project:** Industry-scale causal inference platform with real-time decision making

**TRACK C: SYMBOLIC REGRESSION & PHYSICS-INFORMED AI**

*Duration: 12 weeks parallel track or 8 weeks intensive*

**Module C1: Symbolic Regression Fundamentals (Weeks 1-2)**

**Study Time:** 16 hours per week  
**Prerequisites:** Calculus, basic optimization knowledge

**Learning Objectives:**

* Master symbolic regression principles and genetic programming
* Understand expression trees and symbolic mathematics
* Implement evolutionary algorithms for equation discovery

**Theory Resources:**

* [A Comprehensive Survey of Symbolic Regression](https://arxiv.org/abs/2211.10873) - State-of-the-art overview (4 hours)
* [Genetic Programming Theory](https://mitpress.mit.edu/9780262111706/genetic-programming/) - Koza's foundational work (selected chapters, 6 hours)
* [AI Feynman Paper](https://arxiv.org/abs/1905.11481) - Neural-symbolic regression (2 hours)

**Practical Resources:**

* [gplearn](https://gplearn.readthedocs.io/en/stable/) - Genetic programming for symbolic regression
* [PySR](https://github.com/MilesCranmer/PySR) - High-performance symbolic regression
* [SymbolicRegression.jl](https://github.com/MilesCranmer/SymbolicRegression.jl) - Julia implementation

**Hands-on Assignment:**

* Implement genetic programming algorithm from scratch for simple functions
* Use gplearn to rediscover physical laws from synthetic data
* Create symbolic regression benchmark suite with known equations
* Build expression simplification and mathematical reasoning system
* **Project:** Automated scientific discovery system for experimental data

**Module C2: Kolmogorov-Arnold Networks & Neural-Symbolic Integration (Weeks 3-4)**

**Study Time:** 20 hours per week

**Learning Objectives:**

* Master Kolmogorov-Arnold Networks (KANs) and their theoretical foundations
* Implement KANs for function approximation and symbolic discovery
* Combine neural networks with symbolic reasoning and differentiable programming
* Compare KANs vs MLPs for scientific applications

**Theory Resources:**

* [KAN: Kolmogorov-Arnold Networks](https://arxiv.org/abs/2404.19756) - Original KAN paper by Liu et al. (4 hours)
* [Kolmogorov-Arnold Representation Theorem](https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Arnold_representation_theorem) - Mathematical foundations (2 hours)
* [Neural Module Networks](https://arxiv.org/abs/1511.02799) - Composable neural architectures (2 hours)
* [End-to-End Differentiable Physics](https://arxiv.org/abs/1611.01652) - Differentiable simulation (2 hours)
* [Graph Networks for Symbolic Mathematics](https://arxiv.org/abs/1912.01412) - GNNs for symbolic reasoning (2 hours)

**Practical Resources:**

* [pykan](https://github.com/KindXiaoming/pykan) - Official KAN implementation
* [efficient-kan](https://github.com/Blealtan/efficient-kan) - Optimized KAN variants
* [JAX](https://jax.readthedocs.io/en/latest/) - Differentiable programming framework
* [SymPy](https://www.sympy.org/en/index.html) - Symbolic mathematics in Python
* [DGL](https://www.dgl.ai/) - Deep graph library for GNNs

**Hands-on Assignment:**

* Implement KAN from scratch and compare with MLPs on symbolic regression tasks
* Build efficient KAN variants (Chebyshev-KAN, Fourier-KAN) for different function classes
* Create KAN-based physics-informed networks for PDE solving
* Implement neural-symbolic regression using differentiable programming
* Build GNN for symbolic expression manipulation
* Compare KAN interpretability vs traditional neural networks
* **Project:** KAN-based automated scientific discovery system with symbolic interpretation

**Advanced KAN Topics:**

* **KAN Variants:** Chebyshev-KAN, Fourier-KAN, Wavelet-KAN for different basis functions
* **Sparse KANs:** Pruning techniques for discovering minimal symbolic representations
* **Multi-scale KANs:** Hierarchical KAN architectures for complex function approximation
* **KAN Ensembles:** Combining multiple KANs for uncertainty quantification
* **Physics-KANs:** Incorporating conservation laws and symmetries into KAN architectures

**Module C3: Physics-Informed Neural Networks (Weeks 5-7)**

**Study Time:** 20 hours per week

**Learning Objectives:**

* Master PINN theory and implementation for PDEs
* Understand conservation laws and physical constraints in neural networks
* Implement domain decomposition and multi-scale PINNs

**Theory Resources:**

* [Physics-Informed Neural Networks](https://arxiv.org/abs/1711.10561) - Original PINN paper by Raissi et al. (3 hours)
* [DeepXDE Paper](https://arxiv.org/abs/1907.04502) - Library for scientific ML (2 hours)
* [Conservative PINNs](https://arxiv.org/abs/2004.10444) - Energy-conserving neural networks (2 hours)
* [Multi-scale PINNs](https://arxiv.org/abs/2009.04544) - Handling multiple temporal/spatial scales

**Practical Resources:**

* [DeepXDE](https://github.com/lululxvi/deepxde) - Comprehensive PINNs library
* [NVIDIA SimNet](https://developer.nvidia.com/simnet) - Industrial-scale physics simulation
* [SciML](https://sciml.ai/) - Scientific machine learning ecosystem

**Hands-on Assignment:**

* Implement PINN from scratch for heat equation and wave equation
* Build conservation-enforcing neural networks for fluid dynamics
* Create multi-scale PINN for complex physical systems
* Compare PINNs with traditional numerical methods (FEM, FDM)
* **Project:** PINN-based digital twin for engineering system

**Module C4: Advanced Scientific ML (Weeks 8-9)**

**Study Time:** 22 hours per week

**Learning Objectives:**

* Implement neural ordinary differential equations (NODEs)
* Master universal differential equations (UDEs)
* Build hybrid physics-ML models for real-world systems

**Theory Resources:**

* [Neural ODEs Paper](https://arxiv.org/abs/1806.07366) - Continuous normalizing flows and NODEs
* [Universal Differential Equations](https://arxiv.org/abs/2001.04385) - SciML paradigm
* [Augmented NODEs](https://arxiv.org/abs/1904.01681) - Improving NODE expressivity

**Practical Resources:**

* [torchdiffeq](https://github.com/rtqichen/torchdiffeq) - Differentiable ODE solvers in PyTorch
* [DifferentialEquations.jl](https://docs.sciml.ai/DiffEqDocs/stable/) - Julia ODE solver suite
* [Flux.jl](https://fluxml.ai/) - Julia ML framework

**Hands-on Assignment:**

* Implement Neural ODEs for time series modeling and generation
* Build Universal Differential Equations for hybrid modeling
* Create Augmented NODEs for complex dynamical systems
* Compare NODEs with RNNs/LSTMs on sequential data
* **Project:** Hybrid physics-ML model for climate prediction or drug dynamics

**Module C5: Computational Discovery & Automation (Weeks 10-12)**

**Study Time:** 24 hours per week

**Learning Objectives:**

* Build automated scientific discovery pipelines
* Master active learning for experimental design
* Create self-improving scientific ML systems

**Theory Resources:**

* [AI for Scientific Discovery](https://www.nature.com/articles/s41586-021-03819-2) - Nature review article
* [Automated Machine Learning](https://arxiv.org/abs/1908.00709) - AutoML for scientific applications
* [Closed-Loop Optimization](https://arxiv.org/abs/2010.07435) - Self-driving laboratories

**Practical Resources:**

* [AutoML-Zero](https://github.com/google-research/google-research/tree/master/automl_zero) - Automated ML discovery
* [Ax Platform](https://ax.dev/) - Adaptive experimentation framework
* [BoTorch](https://botorch.org/) - Bayesian optimization in PyTorch

**Hands-on Assignment:**

* Build automated symbolic regression with active learning
* Create self-improving PINN system with uncertainty quantification
* Implement closed-loop optimization for materials discovery
* Deploy automated scientific discovery platform
* **Project:** AI-driven materials or drug discovery system with experimental feedback

**TRACK D: JULIA SCIML ECOSYSTEM**

*Duration: 8 weeks parallel track or 6 weeks intensive*

**Module D1: Julia Fundamentals & SciML Setup (Weeks 1-2)**

**Study Time:** 14 hours per week  
**Prerequisites:** Basic programming knowledge

**Learning Objectives:**

* Master Julia programming language fundamentals
* Set up high-performance scientific computing environment
* Understand Julia's multiple dispatch and type system

**Theory Resources:**

* [Julia Documentation](https://docs.julialang.org/en/v1/) - Complete language reference (8 hours)
* [Think Julia Book](https://benlauwens.github.io/ThinkJulia.jl/latest/book.html) - Free online textbook (6 hours)
* [Julia High Performance](https://www.packtpub.com/product/julia-high-performance-second-edition/9781788298117) - Performance optimization

**Practical Resources:**

* [Julia Academy](https://juliaacademy.com/) - Free online courses
* [JuliaBox](https://www.juliabox.com/) - Cloud Julia environment
* [Pluto.jl](https://github.com/fonsp/Pluto.jl) - Interactive notebooks

**Hands-on Assignment:**

* Implement numerical algorithms (linear algebra, optimization) in Julia
* Create performance benchmarks comparing Julia with Python/MATLAB
* Build package with proper testing and documentation
* Set up Julia environment for scientific ML workflows
* **Project:** High-performance numerical computing library in Julia

**Module D2: DifferentialEquations.jl Mastery (Weeks 3-4)**

**Study Time:** 18 hours per week

**Learning Objectives:**

* Master world's most advanced ODE/PDE solver ecosystem
* Implement stiff and non-stiff solvers with automatic method selection
* Build complex multi-scale and multi-physics simulations

**Theory Resources:**

* [DifferentialEquations.jl Documentation](https://docs.sciml.ai/DiffEqDocs/stable/) - Comprehensive solver guide (10 hours)
* [Stiff ODE Solvers](https://arxiv.org/abs/1804.04344) - Modern numerical methods (4 hours)
* [Delay Differential Equations](https://docs.sciml.ai/DiffEqDocs/stable/types/dde_types/) - Advanced equation types

**Practical Resources:**

* [DiffEqTutorials](https://github.com/SciML/DiffEqTutorials.jl) - Comprehensive tutorials
* [ModelingToolkit.jl](https://github.com/SciML/ModelingToolkit.jl) - Symbolic modeling framework
* [Catalyst.jl](https://github.com/SciML/Catalyst.jl) - Chemical reaction networks

**Hands-on Assignment:**

* Solve complex ODEs with automatic method selection and adaptivity
* Implement stochastic differential equations (SDEs) for uncertainty quantification
* Build multi-scale models combining fast and slow dynamics
* Create parallel and GPU-accelerated ODE solving pipelines
* **Project:** Comprehensive biological system simulation (epidemiology, pharmacokinetics, or ecology)

**Module D3: Neural & Universal Differential Equations (Weeks 5-6)**

**Study Time:** 20 hours per week

**Learning Objectives:**

* Implement cutting-edge Neural ODEs and UDEs in Julia
* Master DiffEqFlux.jl for scientific machine learning
* Build hybrid physics-ML models with uncertainty quantification

**Theory Resources:**

* [Universal Differential Equations](https://arxiv.org/abs/2001.04385) - Foundational UDE paper (3 hours)
* [DiffEqFlux.jl Paper](https://arxiv.org/abs/1902.02376) - Scientific ML in Julia (2 hours)
* [Augmented Neural ODEs](https://arxiv.org/abs/1904.01681) - Advanced NODE architectures (2 hours)

**Practical Resources:**

* [DiffEqFlux.jl](https://github.com/SciML/DiffEqFlux.jl) - Neural differential equations
* [Flux.jl](https://fluxml.ai/) - Native Julia deep learning
* [SciMLSensitivity.jl](https://github.com/SciML/SciMLSensitivity.jl) - Efficient gradient computation

**Hands-on Assignment:**

* Implement Neural ODEs for continuous-time sequence modeling
* Build Universal Differential Equations for hybrid system identification
* Create physics-constrained neural networks with conservation laws
* Develop uncertainty quantification for scientific ML models
* **Project:** Hybrid model for complex dynamical system (climate, biology, or engineering)

**Module D4: High-Performance Scientific Computing (Weeks 7-8)**

**Study Time:** 22 hours per week

**Learning Objectives:**

* Master GPU computing and parallel algorithms in Julia
* Implement distributed computing for large-scale simulations
* Build production-ready scientific computing workflows

**Theory Resources:**

* [Julia GPU Computing](https://juliagpu.org/) - Complete GPU programming guide (6 hours)
* [Distributed Computing in Julia](https://docs.julialang.org/en/v1/manual/distributed-computing/) - Parallel programming (4 hours)
* [High-Performance Julia](https://github.com/johnfgibson/whirlwind-tour-of-julia) - Optimization techniques

**Practical Resources:**

* [CUDA.jl](https://github.com/JuliaGPU/CUDA.jl) - NVIDIA GPU computing
* [MPI.jl](https://github.com/JuliaParallel/MPI.jl) - Message passing interface
* [Distributed.jl](https://github.com/JuliaLang/Distributed.jl) - Built-in parallelism

**Hands-on Assignment:**

* Implement GPU-accelerated PDE solvers using CUDA.jl
* Build distributed parameter sweeps for sensitivity analysis
* Create high-performance linear algebra routines
* Develop scalable scientific ML training pipelines
* **Project:** Massively parallel scientific simulation with real-time visualization

**INTEGRATION & ADVANCED PROJECTS**

**Cross-Track Integration Projects**

**Project 1: AI-Driven Scientific Discovery Platform** *Combines all tracks for comprehensive scientific AI system*

**Requirements:**

* Generative models (Track A) for hypothesis generation and synthetic data
* Causal inference (Track B) for experimental design and confounding control
* Symbolic regression (Track C) for automated equation discovery
* Julia SciML (Track D) for high-performance simulation and modeling
* Production deployment with real-time experimentation capabilities

**Project 2: Autonomous Research Laboratory** *Integration of causal AI, symbolic regression, and automated experimentation*

**Requirements:**

* Causal discovery for understanding experimental systems
* Symbolic regression for model discovery from experimental data
* Active learning for optimal experimental design
* Real-time hypothesis testing and model refinement
* Physics-informed constraints for scientifically plausible models

**Project 3: Multi-Scale Climate Modeling System** *Comprehensive climate prediction using all advanced techniques*

**Requirements:**

* Physics-informed neural networks for atmospheric dynamics
* Causal models for climate driver analysis
* Generative models for scenario generation and uncertainty quantification
* High-performance Julia implementation for real-time prediction
* Symbolic regression for discovering new climate relationships

**Recommended Study Progressions**

**Option 1: Parallel Integration (Weeks 1-26 + Supplementary)**

* Main curriculum (26 weeks) + one supplementary track (8-12 weeks parallel)
* Best for: Strong programming background, 25-30 hours/week study time
* Advantage: Immediate integration of concepts

**Option 2: Sequential Specialization (26 + 20-30 weeks)**

* Complete main curriculum first, then pursue 2-3 supplementary tracks
* Best for: Systematic learners, building solid foundations first
* Advantage: Deep mastery before specialization

**Option 3: Selective Integration (26 + 12-16 weeks)**

* Main curriculum + selected modules from multiple tracks
* Best for: Career-focused learning, specific application domains
* Advantage: Customized skill development for specific goals

**Advanced Assessment Framework**

**Technical Competency Levels:**

* **Level 1:** Reproduce paper implementations with provided code
* **Level 2:** Implement methods from scratch following papers
* **Level 3:** Extend methods with novel improvements
* **Level 4:** Combine multiple techniques for new applications
* **Level 5:** Create novel methods and contribute to research

**Portfolio Development:**

* Minimum 2 projects per supplementary track
* 1 major integration project combining 3+ techniques
* Open source contributions to relevant libraries
* Technical blog posts or papers documenting discoveries
* Production deployments with performance benchmarks

This extended curriculum transforms you into a well-rounded AI researcher and practitioner with cutting-edge expertise in the most advanced areas of artificial intelligence and scientific computing. The modular design allows for flexible learning paths while ensuring comprehensive coverage of both theoretical foundations and practical implementation skills.