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 - Visual neural network explanations (4 hours)
- Fast.ai Lesson 1 - Practical deep learning introduction (2 hours)
- MIT 6.034 Neural Networks Lecture - Theoretical foundations (2 hours)

Practical Resources:

- Google Colab Neural Network from Scratch - Implementation tutorial
- PyTorch 60 Minute Blitz - 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 - Object-oriented programming, decorators, generators (4 hours)
- Git and GitHub Tutorial - Version control fundamentals (2 hours)

Practical Resources:

- NumPy Tutorial for Machine Learning - Array operations and vectorization
- Matplotlib Gallery - Data visualization techniques
- VS Code for Data Science Setup - 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 - Complete PyTorch fundamentals (6 hours)
- Google Colab Guide - Cloud computing setup (2 hours)
- Understanding GPU Computing for ML - Hardware acceleration concepts

Practical Resources:

- PyTorch Examples Repository - Official implementations
- Google Colab Pro Features - 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 - Optimization and hyperparameter tuning (audit for free)
- Fast.ai Lesson 2-3 - Training methodology and data augmentation

Practical Resources:

- PyTorch Training Loop Tutorial - Professional training practices
- Albumentations Documentation - 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 - Lectures 5-7 on CNNs and backpropagation (6 hours)
- Convolution Arithmetic Guide - Visual convolution explanations
- Backpropagation Calculus - Mathematical foundations by 3Blue1Brown

Practical Resources:

- CNN Implementation from Scratch - PyTorch tutorial
- CS231n Assignment Solutions - 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 - Original residual networks paper
- EfficientNet Paper - Compound scaling methodology
- Papers with Code CNN Architectures - Architecture comparisons

Practical Resources:

- Timm Models Library - Pre-trained CNN implementations
- Transfer Learning Tutorial - 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 - Optimization for machine learning
- Batch Normalization Paper - Understanding batch normalization
- Adam Optimizer Paper - Adaptive moment estimation

Practical Resources:

- PyTorch Optimizers Documentation - Optimizer implementations
- Learning Rate 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 - Cyclic learning rate methodology
- One Cycle Policy - Super-convergence techniques
- Fast.ai One Cycle - Practical implementation guide

Practical Resources:

- PyTorch One Cycle Scheduler - Official implementation
- Weights & Biases Tutorial - 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 - Multi-GPU training setup
- ImageNet Training Best Practices - Scaling to large datasets
- Mixed Precision Training - Memory and speed optimization

Practical Resources:

- ImageNet Training Example - Complete training script
- Fast.ai ImageNet Training - 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 - Real-time object detection
- Detectron2 Documentation - Object detection framework
- U-Net Paper - Semantic segmentation

Practical Resources:

- YOLOv5 Repository - Production-ready object detection

- Detectron2 Tutorial - 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 - Auto-encoding variational Bayes
- GAN Paper - Generative adversarial networks
- Lil'Log Generative Models - Comprehensive overview

Practical Resources:

- VAE PyTorch Tutorial - Official implementation
- GAN PyTorch Tutorial - 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 - Original Transformer paper (3 hours)
- The Illustrated Transformer - Visual explanation (2 hours)
- 3Blue1Brown Attention - Mathematical intuition (1 hour)
- Harvard NLP Annotated Transformer - Line-by-line implementation

Practical Resources:

- Hugging Face Transformers Course - Comprehensive transformer tutorial
- PyTorch Transformer Tutorial - 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 - Efficient estimation of word representations
- BPE Paper - Subword tokenization
- SentencePiece - Unsupervised text tokenizer

Practical Resources:

- Hugging Face Tokenizers Library - Fast tokenization implementations

- Word Embeddings Tutorial - 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 - Improving language understanding by generative pre-training
- GPT-2 Paper - Language models are unsupervised multitask learners
- Training Compute-Optimal Large Language Models - Chinchilla scaling laws

Practical Resources:

- nanoGPT - Minimal GPT implementation by Andrej Karpathy
- GPT-2 from Scratch - Complete implementation tutorial
- Hugging Face 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 - Training multi-billion parameter language models
- Efficient Large-Scale Language Model Training - ZeRO optimizer states partitioning
- HELM Paper - Holistic evaluation of language models

Practical Resources:

- DeepSpeed Tutorial - Memory and compute optimization
- Accelerate Library - Multi-GPU training simplified
- 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 - Training language models to follow instructions with human feedback
- LoRA Paper - Low-rank adaptation of large language models

- Parameter-Efficient Transfer Learning - Adapter modules

Practical Resources:

- PEFT Library - Parameter-efficient fine-tuning methods
- Alpaca Training - Instruction-following fine-tuning
- QLoRA Paper - 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 - Comprehensive quantization survey
- Knowledge Distillation Paper - Distilling knowledge in neural networks
- GPTQ Paper - Accurate post-training quantization for GPT models

Practical Resources:

- PyTorch Quantization - Official quantization toolkit
- BitsAndBytes - 8-bit optimizers and quantization
- 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 - Learning transferable visual models from natural language supervision
- DALL-E Paper - Zero-shot text-to-image generation
- Flamingo Paper - Few-shot learning with frozen language models

Practical Resources:

- OpenCLIP Implementation - Open source CLIP training
- LAION Dataset - Large-scale image-text pairs
- Hugging Face Vision-Language Models - 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 - Chapters 1-6 (free online textbook)
- David Silver RL Course - DeepMind RL lectures
- OpenAI Spinning Up - Deep RL introduction

Practical Resources:

- Stable Baselines3 - High-quality RL implementations
- OpenAI Gymnasium - RL environment toolkit
- 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 - Theoretical foundations
- PPO Paper - Proximal policy optimization
- SAC Paper - Soft actor-critic

Practical Resources:

- PPO Implementation - OpenAI baseline implementation
- PyBullet Environments - Physics simulation for continuous control
- Ray RLLib - 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 - Training language models to follow instructions with human feedback
- Constitutional AI Paper - Training a helpful and harmless assistant
- Reward Modeling Paper - Fine-tuning language models from human preferences

Practical Resources:

- TRL Library - Transformer reinforcement learning
- Open Assistant Dataset - Human feedback conversations
- RLHF Implementation - 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 - Training multi-billion parameter models
- GPT-3 Paper - Language models are few-shot learners
- PaLM Paper - Pathways language model scaling

Practical Resources:

- Megatron-DeepSpeed - Large-scale training framework
- FairScale - PyTorch extensions for large-scale training
- Colossal-AI - 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 - Google Cloud MLOps guide
- Hidden Technical Debt in ML Systems - Production ML challenges
- ML Model Serving Patterns - Deployment architecture patterns

Practical Resources:

- MLOps Zoomcamp - Free comprehensive MLOps course
- BentoML - Model serving framework
- 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 - Foundations and autoregressive models (8 hours)
- Pattern Recognition and Machine Learning - Chapters 2-3 on probability (6 hours)
- Variational Inference Review - Modern variational methods

Practical Resources:

- CS236 Official Assignments - Jupyter notebooks with implementations
- PyTorch Distributions - 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 - VAEs and flow models (8 hours)
- Flow-based Deep Generative Models - Comprehensive survey
- Real NVP Paper - Density estimation using real-valued non-volume preserving transformations

Practical Resources:

- Pyro Tutorials - Probabilistic programming for VAEs
- Normalizing Flows Library - 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 - GANs and score-based models (10 hours)
- Denoising Diffusion Probabilistic Models - DDPM paper
- Score-Based Generative Models - Yang Song's comprehensive survey

Practical Resources:

- StyleGAN2 Implementation - Official NVIDIA implementation
- Diffusers Library - Hugging Face diffusion models
- Score SDE - 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 - Applications in chemistry
- FID and Inception Score - Evaluation metrics
- Energy-Based Models - EBM training and applications

Practical Resources:

- Molecule Generation Toolkit - Chemical generative models
- FID Score Implementation - Standard evaluation metric
- 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 - Judea Pearl's introduction to causality (8 hours reading)
- Causal Inference: The Mixtape - Scott Cunningham's free online textbook (6 hours)
- Brady Neal's Causal Inference Course - Video lectures 1-8 (8 hours)

Practical Resources:

- DoWhy Library - Microsoft's causal inference library
- CausalML - Uber's causal ML toolkit

- 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 - Comprehensive algorithm collection
- PC Algorithm Paper - Constraint-based discovery
- NOTEARS Paper - Continuous optimization for structure learning

Practical Resources:

- causal-learn - Comprehensive causal discovery library
- CASTLE - Causal structure learning toolkit
- 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 - Comprehensive survey by Schölkopf et al.
- β-VAE Paper - Disentangled representation learning
- Invariant Risk Minimization - Domain generalization with causality

Practical Resources:

- disentanglement_lib - Google's disentanglement library
- CausalVAE Implementation - Causal representation learning
- 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 - Survey of causal RL methods
- Causal Confusion in Imitation Learning - Identifying spurious correlations
- Counterfactual Data Augmentation - CF-based RL improvements

Practical Resources:

- CausalCity - Causal RL benchmark
- WhyNot - Causal inference benchmark suite
- Causal RL Environments - 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 - Applied causal analysis
- Experimentation Platform Design - Large-scale A/B testing
- Causal ML in Industry - Real-world case studies

Practical Resources:

- GrowthBook - Open source experimentation platform
- Statsig - Feature flagging and experimentation
- 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 - State-of-the-art overview (4 hours)
- Genetic Programming Theory - Koza's foundational work (selected chapters, 6 hours)
- AI Feynman Paper - Neural-symbolic regression (2 hours)

Practical Resources:

- gplearn - Genetic programming for symbolic regression
- PySR - High-performance symbolic regression
- 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 - Original KAN paper by Liu et al. (4 hours)
- Kolmogorov-Arnold Representation Theorem - Mathematical foundations (2 hours)
- Neural Module Networks - Composable neural architectures (2 hours)
- End-to-End Differentiable Physics - Differentiable simulation (2 hours)
- Graph Networks for Symbolic Mathematics - GNNs for symbolic reasoning (2 hours)

Practical Resources:

- pykan - Official KAN implementation
- efficient-kan - Optimized KAN variants
- JAX - Differentiable programming framework
- SymPy - Symbolic mathematics in Python
- DGL - 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 - Original PINN paper by Raissi et al. (3 hours)
- DeepXDE Paper - Library for scientific ML (2 hours)
- Conservative PINNs - Energy-conserving neural networks (2 hours)
- Multi-scale PINNs - Handling multiple temporal/spatial scales

Practical Resources:

- DeepXDE - Comprehensive PINNs library
- NVIDIA SimNet - Industrial-scale physics simulation
- SciML - 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 - Continuous normalizing flows and NODEs
- Universal Differential Equations - SciML paradigm
- Augmented NODEs - Improving NODE expressivity

Practical Resources:

- torchdiffeq - Differentiable ODE solvers in PyTorch
- DifferentialEquations.jl - Julia ODE solver suite
- Flux.jl - 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 - Nature review article
- Automated Machine Learning - AutoML for scientific applications
- Closed-Loop Optimization - Self-driving laboratories

Practical Resources:

- AutoML-Zero - Automated ML discovery
- Ax Platform - Adaptive experimentation framework
- BoTorch - 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 - Complete language reference (8 hours)
- Think Julia Book - Free online textbook (6 hours)
- Julia High Performance - Performance optimization

Practical Resources:

- Julia Academy - Free online courses
- JuliaBox - Cloud Julia environment
- 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 - Comprehensive solver guide (10 hours)
- Stiff ODE Solvers - Modern numerical methods (4 hours)
- Delay Differential Equations - Advanced equation types

Practical Resources:

- DiffEqTutorials - Comprehensive tutorials
- ModelingToolkit.jl - Symbolic modeling framework
- 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 - Foundational UDE paper (3 hours)
- DiffEqFlux.jl Paper - Scientific ML in Julia (2 hours)
- Augmented Neural ODEs - Advanced NODE architectures (2 hours)

Practical Resources:

- DiffEqFlux.jl - Neural differential equations
- Flux.jl - Native Julia deep learning
- 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 - Complete GPU programming guide (6 hours)
- Distributed Computing in Julia - Parallel programming (4 hours)
- High-Performance Julia - Optimization techniques

Practical Resources:

- CUDA.jl - NVIDIA GPU computing
- MPI.jl - Message passing interface
- 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

---

TRACK E: LLM AGENTS & AGENTIC AI SYSTEMS

Duration: 10 weeks parallel track or 7 weeks intensive
Based on Berkeley RDI's cutting-edge LLM Agents course

Module E1: Agent Foundations & Reasoning (Weeks 1-2)

Study Time: 18 hours per week
Prerequisites: Complete through Week 15 of main curriculum (LLM Training)

Learning Objectives:

- Master LLM reasoning capabilities (Chain-of-Thought, Tree-of-Thought)
- Understand ReAct paradigm (Reasoning + Acting)
- Implement planning and tool use for autonomous agents
- Build foundation for agentic behavior

Theory Resources:

- Chain-of-Thought Reasoning Without Prompting - Advanced CoT techniques (2 hours)
- ReAct: Synergizing Reasoning and Acting in Language Models - Core agent paradigm (3 hours)
- Tree of Thoughts - Deliberate problem solving (2 hours)
- Berkeley LLM Agents Lectures 1-2 - Foundational concepts (4 hours)

Practical Resources:

- LangChain Agents - Agent development framework
- AutoGPT - Autonomous agent implementation
- ReAct Implementation - Official ReAct codebase

Hands-on Assignment:

- Implement Chain-of-Thought and Tree-of-Thought reasoning from scratch
- Build ReAct agent with tool integration (calculator, web search, code execution)
- Create reasoning evaluation framework for complex problem-solving
- Develop planning algorithms for multi-step task execution
- **Project:** Autonomous research assistant that can gather information, reason, and provide comprehensive answers

Module E2: Multi-Agent Systems & Frameworks (Weeks 3-4)

Study Time: 20 hours per week

Learning Objectives:

- Master multi-agent collaboration and communication protocols
- Implement AutoGen and CrewAI frameworks for team-based AI systems
- Understand agent orchestration and workflow management

- Build scalable multi-agent architectures

Theory Resources:

- AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation - Multi-agent frameworks (3 hours)
- StateFlow: Enhancing LLM Task-Solving through State-Driven Workflows - Agent workflow management (2 hours)
- Multi-Agent Reinforcement Learning Survey - Collaborative learning (3 hours)

Practical Resources:

- AutoGen - Microsoft's multi-agent framework
- CrewAI - Role-based agent collaboration
- LangGraph - Agent workflow orchestration
- Swarm - OpenAI's lightweight multi-agent framework

Hands-on Assignment:

- Implement multi-agent debate system for complex decision making
- Build specialized agent teams (researcher, writer, critic, reviewer)
- Create agent communication protocols and shared memory systems
- Develop hierarchical agent architectures with delegation capabilities
- **Project:** Multi-agent software development team (PM, architect, developer, tester)

Module E3: Compound AI Systems & DSPy (Weeks 5-6)

Study Time: 22 hours per week

Learning Objectives:

- Master compound AI system design and optimization
- Implement DSPy for programmatic LLM application development
- Understand retrieval-augmented generation (RAG) at scale
- Build self-improving AI systems with automatic optimization

Theory Resources:

- DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines - Programming paradigm for LLMs (4 hours)
- Optimizing Instructions and Demonstrations for Multi-Stage Language Model Programs - Automatic optimization (3 hours)
- The Needle In a Haystack Test - RAG evaluation (2 hours)

Practical Resources:

- DSPy - Stanford's programming model for LMs
- LlamaIndex - Advanced RAG framework
- Haystack - End-to-end NLP framework
- Vector Databases Comparison - Scalable retrieval systems

Hands-on Assignment:

- Implement DSPy programs with automatic prompt optimization
- Build enterprise-scale RAG systems with advanced retrieval strategies
- Create self-optimizing AI pipelines using DSPy's compilation
- Develop multimodal knowledge assistants with tool integration
- **Project:** Comprehensive document analysis system with automatic optimization

Module E4: Computer Use & Web Automation (Weeks 7-8)

Study Time: 24 hours per week

Learning Objectives:

- Master computer use capabilities (screen understanding, GUI interaction)
- Implement web automation agents for complex workflows
- Build agents that can interact with software applications
- Understand visual-language models for computer interfaces

Theory Resources:

- WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents - Web interaction (3 hours)
- WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks? - Enterprise web automation (3 hours)
- Anthropic Computer Use - State-of-the-art computer interaction (2 hours)

Practical Resources:

- SWE-agent - Software engineering agents
- OpenHands - Code generation agents
- Selenium - Web automation framework
- Playwright - Modern browser automation

Hands-on Assignment:

- Implement screen understanding using vision-language models
- Build web automation agents for e-commerce and business workflows
- Create software testing agents that can interact with GUIs
- Develop agents for code generation and debugging in IDEs
- **Project:** End-to-end business process automation agent

Module E5: Agent Safety & Evaluation (Weeks 9-10)

Study Time: 20 hours per week

Learning Objectives:

- Master agent evaluation frameworks and benchmarks
- Implement safety measures and alignment for autonomous agents
- Understand responsible scaling and deployment of agent systems
- Build monitoring and governance systems for production agents

Theory Resources:

- Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models - Agent capability assessment (3 hours)
- DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models - Trust evaluation (4 hours)
- Anthropic Responsible Scaling Policy - Safety frameworks (2 hours)

Practical Resources:

- AgentBench - Comprehensive agent evaluation
- WebArena - Web agent benchmarking
- SWE-bench - Software engineering evaluation
- HELM - Holistic evaluation framework

Hands-on Assignment:

- Build comprehensive agent evaluation suite across multiple domains
- Implement safety monitoring and intervention systems
- Create red-teaming framework for identifying agent vulnerabilities
- Develop governance protocols for enterprise agent deployment
- **Project:** Complete agent safety and evaluation platform with real-time monitoring

---

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
- **NEW:** Multi-agent systems (Track E) for collaborative scientific reasoning
- 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
- **NEW:** Agent-based experimental design and hypothesis generation
- **NEW:** Multi-agent scientific collaboration (theorist, experimenter, analyzer)
- 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
- **NEW:** Agent-based climate policy analysis and recommendation systems
- High-performance Julia implementation for real-time prediction
- Symbolic regression for discovering new climate relationships

NEW Project 4: Autonomous Software Development Platform

Complete agentic software engineering system

Requirements:

- Multi-agent development teams (PM, architect, developer, tester, reviewer)
- Computer use capabilities for IDE interaction and testing
- Causal analysis for debugging and root cause analysis
- Generative models for code synthesis and documentation
- Symbolic regression for algorithm discovery and optimization
- Production deployment with continuous integration and monitoring

Recommended Study Progressions

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

- Main curriculum (26 weeks) + two supplementary tracks (16-20 weeks parallel)
- **NEW:** Highly recommend Track E (LLM Agents) + Track B (Causal AI) combination
- Best for: Strong programming background, 30-35 hours/week study time
- Advantage: Immediate integration of concepts, modern skill combination

Option 2: Sequential Specialization (26 + 25-35 weeks)

- Complete main curriculum first, then pursue 3-4 supplementary tracks
- **NEW:** Recommended sequence: Main → Track E → Track B → Track C
- Best for: Systematic learners, building solid foundations first
- Advantage: Deep mastery before specialization, complete coverage

Option 3: Selective Integration (26 + 15-20 weeks)

- Main curriculum + selected modules from multiple tracks
- **NEW:** Must-have modules: E1-E3 (Agent foundations and frameworks)
- Best for: Career-focused learning, rapid deployment needs
- Advantage: Customized skill development for immediate application

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.