CMU ONLINE
Carnegie Mellon University

Syllabus Overview Sample: Introduction to Deep Learning

Program: AI Engineering for Mechanical Engineering

Faculty: Dr. Amir Barati Farimani, Russell V. Trader Associate Professor, Mechanical Engineering, College of Engineering

Course Number: 24888 **Units:** 12

Zoom Class: Weekday Evening, ET

Course Description

This course provides an introduction to deep learning. We will learn about the basics of deep neural networks, and their applications to different tasks in engineering. Students will be able to apply deep learning to a variety of artificial intelligence tasks pertinent to different engineering problems. Applications of deep learning in mechanical, chemical, biological, electrical, and material engineering will be discussed.

Course Learning Objectives

By the end of this course, students will be able to:

- Explain the basics of deep neural networks and their applications to different tasks in engineering.
- Describe and compare commonly used deep learning algorithms, their theoretical underpinnings, and how they are applied to solve real world problems.
- Implement deep learning algorithms in Python using PyTorch.
- Analyze practical engineering problems to identify and apply relevant deep learning algorithms.

Faculty Introduction



Amir Barati Farimani received his Ph.D. in 2015 in mechanical science and engineering from the University of Illinois at Urbana-Champaign. His Ph.D. thesis was titled “Detecting and Sensing Biological Molecules using Nanopores.” He extensively used atomistic simulations to shed light on the DNA sensing and detection physics of biological and solid state nanopores. Right after that, he joined Professor Vijay Pande’s lab at Stanford. During his

post-doc, he combined machine learning and molecular dynamics to elucidate the conformational changes of G-Protein Coupled Receptors (GPCRs). He specifically was focused on Mu-Opioid Receptors to elucidate their free energy landscape and their activation mechanism and pathway.

The Barati Farimani's lab, the Mechanical and Artificial Intelligence laboratory (MAIL), at Carnegie Mellon University is broadly interested in the application of machine learning, data science, and molecular dynamics simulations to health and bio-engineering problems. The lab is inherently a multidisciplinary group bringing together researchers with different backgrounds and interests, including mechanical, computer science, bio-engineering, physics, material, and chemical engineering. The mission is to bring the state-of-the-art machine learning algorithm to mechanical engineering. Traditional mechanical engineering paradigms use only physics-based rules and principles to model the world, which does not include the intrinsic noise/stochastic nature of the system. To this end, the lab is developing the algorithms that can infer, learn, and predict the mechanical systems based on data. These data-driven models incorporate the physics into learning algorithms to build more accurate predictive models. They use multi-scale simulation (CFD, MD, DFT) to generate the data.

Course Topics

Topic	Learning Objectives
What is Deep Learning?	• Compare and contrast traditional machine learning with deep learning. • Explain when to use each approach (traditional ML vs Deep Learning) based on problem complexity and data availability. • Explain how neural networks learn from data through training.
Neural Networks and Back Propagation	• Explain Empirical Risk Minimization • Describe different components of Neural Network • Explain different loss functions for both classification and regression tasks • Elucidate how neural network learns via backpropagation • Explain what is gradient descent and how it works to train a neural network
Regularization and Convolutional Neural Networks	• Describe different methods of weight initializations • Enumerate different regularization techniques • Explain how drop out works • Explain why we need Convolutional Neural Network
Convolutional Neural Networks	• Describe different components of the convolution layer. • Explain how CNNs learn features. • Explain and use pooling, padding and feature maps. • Explain and use how backpropagation works in CNNs.

	● Implement a simple convolutional neural network in Pytorch.
Convolutional Neural Networks (CNN) + Graph Neural Network (GNN)	● Elaborate different architecture of CNNs (AlexNet, VGG, ResNet,…). ● Explain the differences between architectures. ● Learn to load and use different CNN architecture. ● Explain where one should use graph neural networks. ● Explain how graph neural network works (GNN).
Recurrent Neural Networks	● Describe where we should use RNNs. ● Implement both RNNs and LSTMs. ● Explain the challenges in training RNNs and the solution to these challenges. ● Describe the advantages of LSTMs compared to RNNs
Dimensionality Reduction in Deep Learning	● Describe the differences between PCA and Autoencoders. ● Explain how Auto-encoders work. ● Enumerate the applications of autoencoders and U-Net. ● Describe Variational Auto-encoders and their difference with vanilla auto-encoders.
Generative Adversarial Networks 1	● Describe how Generative Adversarial Networks (GAN) work. ● Enumerate the applications of GANs for image generation and engineering. ● Explain the challenges in training GANs and what are the solutions. ● Implement a simple GAN using Pytorch and generate new images.
Generative Adversarial Networks 2	● Describe applications of Generative Adversarial Networks (GAN) and how they work ● Explain conditional GAN and how it works ● Implement a simple cGAN using Pytorch and generate conditional images
Diffusion Models	● Define bias ● Identify low- vs. high-bias models ● Define variance ● Identify low- vs. high-variance models ● Describe the diffusion models and how they are different from GANs ● Explain how diffusion models work ● Describe the diffusion training process ● Write a simple code implementing Diffusion
Transformer	● Describe the attention mechanism and how it works ● Explain how attention solves the challenge of sequential computing ● Implement a simple attention in PyTorch

	● Use Transformers from Huggingface
Self-Supervised Learning (SSL)	● Describe how self supervised learning works ● Explain why SSL improves the accuracy of prediction ● Implement a simple contrastive learning in Pytorch ● Explain different pre-training techniques
Foundational Models 1	● Describe what is a foundational model ● Explain what are the implications of the size of a model for its prediction ● Explain the differences between different versions of GPT-X ● Explain how foundational models are trained and what types of Transformers are used
Foundational Models 2	● Explain different foundational models such as T5, GPT-2, GPT-3 ● Use Huggingface models and generate text ● Code up using foundational model in Transformer package

---

NOTE: This is a sample designed to provide an overview of topics covered in the course based on previous course syllabi. All students will be provided an updated and final syllabus at the start of the course. Faculty, content, and schedule are all subject to change.