PORTFOLIO

1. Multimodal Effective Video Search and Interaction with RAG

As part of the Multimodal RAG: Chat with Videos course by DeepLearning.AI in collaboration with Intel, I developed a Multimodal Retrieval-Augmented Generation (RAG) system designed to enable intelligent querying and interaction with video content. This system leverages multimodal AI techniques including video frame extraction, transcription, multimodal embedding, and retrieval.

Technical Strategy:

- *Video Preprocessing:* Developed video preprocessing scripts to handle three different video input types (video with transcript, video with audio but no transcript, and video with no audio or transcript). This included extracting video frames and generating transcriptions using OpenAI's Whisper and LLaVA models.
- *Vector Store Ingestion:* Implemented ingestion into LanceDB vector database using BridgeTower embeddings. Experimented with augmenting fragmented transcripts to improve the quality of the data used for retrieval.
- *RAG System Implementation:* Used LangChain to create a multimodal retrieval-augmented generation (RAG) system, enabling video-based querying and interaction. Implemented both text-based and web interface querying systems using Python and Gradio.

Project Deliverables:

- *Video Frame Extraction:* Developed a system that extracts frames from videos and associates them with metadata such as transcript and video segment information.
- *Multimodal Data Ingestion:* Processed and ingested video and transcript data into a vector store to make it searchable.
- *RAG System:* Enabled users to ask multimodal queries about video content using a combination of video frames, transcripts, and embeddings.

Tools Used:

- **LangChain**: To run multimodal RAG system
- **OpenAI Whisper**: For transcription
- **MoviePy**: For audio extraction from videos
- **LLaVA Model**: For generating captions from video frames
- **LanceDB**: For vector storage

API Keys Used:

- **OpenAI API**: For natural language processing
- **Prediction Guard API**: For model predictions

ML Algorithms:

- **BridgeTower Embeddings**: For vectorizing transcripts
- **RAG System**: For querying video data

View Certificate of Completion | Project Link

2. LLaMA Customer Support Assistant: Fine-tuned Conversational AI

A specialized customer support AI system built by fine-tuning LLaMA 3.2 model with Unsloth optimization for M1 architectures. The system handles customer service queries with contextual understanding and structured responses.

Technical Strategy:

- Implemented LoRA fine-tuning on LLaMA 3.2 base model using Unsloth for 2x faster training
- Optimized memory usage for M1 Mac through gradient checkpointing and efficient batching
- Engineered dataset preprocessing pipeline for structured customer support dialogues

Project Features:

- Automated response generation for customer inquiries
- Memory-efficient training on M1 architecture
- Contextual understanding of order-related queries
- Performance visualization and metrics tracking
- Streamlined inference pipeline for quick responses

Core Components:

- Fine-tuned LLaMA 3.2 model with LoRA adaptors
- Custom data processing for customer service context
- Optimized training pipeline for Apple Silicon
- Comprehensive testing suite for model evaluation

Tools & Technologies: PyTorch, Transformers, Unsloth, Python 3.10, PEFT, Datasets, Accelerate, Apple M1 MPS backend

Dataset: 27K customer support conversations from Bitext, covering order management, cancellations, and status inquiries.

Results:

- Achieved 2x faster training through Unsloth optimization
- Efficient memory usage under 8GB RAM
- Natural response generation for customer queries

Project Link

3. Neural Conversational Chatbot with Attention Mechanisms

I developed an advanced chatbot using sequence-to-sequence neural networks and attention mechanisms in PyTorch. The chatbot is trained on the Cornell Movie-Dialogs Corpus, which contains over 220,000 conversation pairs from movie dialogues. The system learns to understand context and generate appropriate responses through sophisticated deep learning techniques.

Technical Strategy:

- *Advanced Architecture:* Built a powerful chatbot using a bidirectional GRU encoder for understanding input messages and a GRU decoder with attention for generating responses.
- *Context Understanding:* Implemented Luong's global attention mechanism to help the model focus on relevant parts of input messages when generating responses.
- *Efficient Processing:* Optimized training through proper batch processing and sequence handling, with carefully tuned parameters (500 hidden units, 2 layers, 0.1 dropout).

Project Features:

- *Natural Conversations:* The chatbot engages in human-like dialogue by understanding context and generating relevant responses
- *Performance Analysis:* Integrated tools to visualize attention patterns and evaluate response quality
- *Modular Design:* Organized code structure with separate modules for encoder, decoder, attention, and utilities for easy maintenance and updates

Tools Used: PyTorch, Python, NLTK, NumPy, Pandas.

ML Algorithms: Sequence-to-Sequence Learning, GRU Networks, Attention Mechanisms.

Project Link

4. LLM Tour Recommendation System with Multi-Agent Architecture

An advanced travel planning system leveraging LangChain and multiple AI agents to generate optimized daily schedules and contextual recommendations for travelers in Busan. The system processes weather data and attraction information to create personalized travel experiences.

Technical Strategy:

- Dual-Agent Architecture utilizing LangChain:
  - *Schedule Generator:* Optimizes daily itineraries based on attraction visit times and weather conditions.
  - *Recommendation Generator:* Delivers contextual travel advice and weather-specific precautions.
- Robust JSON processing pipeline for handling structured attraction and weather data.

Core Features:

- Weather-aware planning that adapts to temperature, air quality, and conditions
- Dynamic schedule adjustment for optimal travel experiences
- Comprehensive error handling and input validation
- Modular architecture for extensible functionality

Tools & Technologies:

- Python 3.7+, PyTorch, LangChain, Ollama
- LLaMA 3.2 model for intelligent processing
- JSON for structured data handling

Results:

- Efficient schedule generation with environmental awareness
- Robust handling of complex scheduling scenarios

Project Link

5. Mastering PyTorch 2

I explored deep learning concepts from fundamentals to advanced architectures through the **Mastering PyTorch 2** book. My learning included implementing **CNN architectures** like DenseNet, GoogLeNet, and ResNet, building **recommendation systems** and **text generation models**, and integrating **Hugging Face** for transformer-based applications. I also worked on **model interpretability** using Captum. Key implementations included MNIST digit classification, transfer learning with AlexNet, LSTM/RNN applications, and music/text generation models.

Project Link

6. Sentiment Analysis using Deep Learning and Traditional ML

This project implements comprehensive sentiment analysis using both modern deep learning approaches and traditional machine learning methods. The system analyzes movie reviews from the IMDB dataset to classify sentiments as positive or negative, utilizing a sophisticated LSTM-based RNN architecture alongside traditional classifiers.

Technical Strategy:

- *LSTM-RNN Architecture:* Built a sophisticated neural network using bidirectional LSTM with customizable layers, word embeddings, dropout regularization, and attention mechanism for enhanced text understanding
- *Traditional ML Integration:* Implemented and optimized Multinomial Naive Bayes and SGD Classifier with configurable parameters for comparative analysis

Project Features:

- *Multi-Model Sentiment Analysis:* Deep Learning LSTM-RNN with bidirectional capability and traditional ML approaches
- *Performance Optimization:* Efficient sequence handling with packed padded sequences and dropout regularization
- *Comparative Analysis:* Comprehensive evaluation and comparison of different model performances
- *Interactive Interface:* Jupyter notebook interface for easy experimentation and result visualization

Tools Used: PyTorch, Scikit-learn, Python, NLTK, NumPy, Pandas, Matplotlib, Seaborn

ML Algorithms: Bidirectional LSTM Networks, Multinomial Naive Bayes, Stochastic Gradient Descent, Word Embeddings.

Project Link

7. Stable Diffusion Image Generation System

Text-Guided Image Transformation with Fine Control

I experimented with Stable Diffusion transformers for text-guided image generation and transformation, implementing fine-grained control mechanisms and GPU acceleration for efficient processing.

Tools used:

- Python 3.7+
- PyTorch
- Hugging Face Diffusers
- CUDA GPU acceleration
- PIL (Python Imaging Library)
- Stable Diffusion v1-4 model

Project Link

8. Battery Remaining Useful Life Prediction

The main aim of this project was to forecast the Remaining Useful Life (RUL) of batteries through the application of diverse regression algorithms. The primary focus was achieving high accuracy, with RandomForestRegressor attaining an impressive 99%. For this project, I used the dataset that was obtained from Kaggle. It comprised 14 NMC-LCO 18650 batteries cycled over 1000 times at 25°C, featuring discharge time, voltage behavior, and charging time. This project includes Exploratory data analysis (EDA), Data Preprocessing, Model Selection and Evaluation, Feature Importance Analysis, Ensemble Modeling steps. For effective data visualization, Matplotlib and Seaborn were utilized to create insightful visualizations. Overall, this project successfully developed a predictive model for battery RUL, offering valuable insights into feature importance and achieving remarkable accuracy with the RandomForestRegressor.

Features Used for Prediction
The predictive features for battery RUL included:
- F1: Discharge Time (s)
- F2: Time at 4.15V (s)
- F3: Time Constant Current (s)
- F4: Decrement 3.6-3.4V (s)
- F5: Max. Voltage Discharge (V)
- F6: Min. Voltage Charge (V)
- F7: Charging Time (s)
- Total time (s)

Most Important Features
As per the model, the most crucial features for predicting RUL were:

1. F1: Discharge Time (s)
2. F4: Decrement 3.6-3.4V (s)

Machine Learning Algorithms
Diverse regression algorithms were implemented using Scikit-learn:
- RandomForestRegressor
- AdaBoostRegressor
- GradientBoostingRegressor
- BaggingRegressor
- SVR
- DecisionTreeRegressor
- ExtraTreeRegressor
- LinearRegression
- SGDRegressor
- KNeighborsRegressor

Tools Used: Python, Numpy, Matplotlib, Seaborn, Pandas, Scikit-learn, Time module, and others.

Project Link

9. Trash Bag Segmentation, Detection, Classification, Tracking, and Counting On Real Time

We designed an advanced AI project aimed at optimizing trash bag collection processes for NetVision Company . Leveraging the power of YOLOv8, our team created a sophisticated system **capable of detecting, classifying, segmenting, tracking, and estimating the size of trash bags within real-time video streams**. My part included reading research papers and training models on YOLOv 8. With this project, we participated in the capstone competition 2023 at Woosong University. We defeated **more than 200 teams** from 29 departments and got the Woosong University President Award. Here is the video of the awarding our team : Project Link | Festival Link | President Award | Team

Technical Strategy:
YOLOv8 Integration: Employed YOLOv8, a state-of-the-art object detection model, to effectively identify and categorise trash bags within live video streams.
*Real-time Processing:*Engineered algorithms for immediate analysis of video data to swiftly and accurately identify and track trash bags.
Precise Segmentation: Utilised advanced segmentation techniques to precisely isolate trash bags in video frames for detailed analysis.

Project Deliverables:
Detection & Classification: Accurate identification and classification of various types of trash bags (e.g, recyclable, non-recyclable) within video feeds.
Segmentation & Tracking: Precise isolation and continuous tracking of trash bags throughout the video streams for efficient monitoring.

Size Estimation: Providing estimations of trash bag sizes for optimal collection planning and resource allocation.

Tools Used: PyTorch, OpenCV, YOLOv8, Ultralytics
ML Algorithms: YOLOv8 object detection

10. Real Time Face Mask Detection with PyTorch and OpenCV

This project utilises PyTorch, OpenCV, and the MTCNN for face detection and mask presence identification. The model is trained on labelled data to accurately detect the presence of face masks in images or real-time video feeds.
Tools Used: PyTorch, OpenCV, MTCNN model
ML Algorithm: Resnet 18, Resnet 50 via Transfer Learning

Project link | Presentation link

11. Multi-class Weather Classification using PyTorch and CNNs

Employing PyTorch and Convolutional Neural Networks, this project categorises images into various weather conditions. By utilising a custom dataset of weather from kaggle datasets and employing techniques like data augmentation and transfer learning, the model's accuracy and performance are enhanced.
Tools Used: PyTorch, Python
ML Algorithm: CNN, Resnet 9

Project link | Presentation Link

12. StatFlow EDA Web App using Streamlit and Data Science Libraries

StatFlow EDA App is a web-based application facilitating Exploratory Data Analysis (EDA) on user-uploaded datasets. Created using Streamlit and data science libraries, the app is hosted on Streamlit cloud. It offers features like descriptive statistics, data cleaning, visualisation, and correlation analysis.
Tools Used: Streamlit, Pandas, Numpy, Matplotlib, Seaborn, Pandas Profiling

Project link | Demo link

13. Indoor Air Quality Monitoring System using IoT and ThingSpeak

This project is dedicated to an innovative IoT-based indoor air quality monitoring system. Leveraging the ESP Wi-Fi module and ThingSpeak cloud platform, the system integrates various sensors to track and display real-time air quality parameters. Additionally, it incorporates data from the OpenWeather API to compare indoor and outdoor air quality.

The system's key features include:

- IoT Integration: Utilising ESP Wi-Fi modules and various sensors to collect indoor air quality data.
- ThingSpeak Cloud: Storing and processing real-time air quality parameters received from different sensors.
- Real-Time Dashboard: Displaying air quality metrics in an interactive and user-friendly dashboard.
- OpenWeather API Integration: Fetching outdoor air quality data to facilitate a comparison with indoor air quality metrics.

Tools Used: Arduino, C language, ThingSpeak Cloud, ESP Wi-Fi, Sensors, Streamlit

Project Link | Presentation link | Demo Link

14. Kidney Disease Auto-Detection with Deep Learning Methods

Using architectures like ResNet50, VGG16, VGG19, and U-Net, this project automates the detection of kidney diseases including cysts, stone, tumor within medical x-ray images of kidney. Although U-net is used for segmentation, I used it for classification tasks by changing the last layers of the neural network. I achieved higher accuracy and precision using U-net autoencoder architecture. The models leverage deep learning and transfer learning techniques for accurate kidney identification.
Tools Used:PyTorch, Python
ML Algorithm: CNN, ResNet 50, VGG16, VGG19, U-Net via Transfer Learning

Project Link | Presentation Link | Report Link

15. Insurance Price Prediction using PyTorch

This project focuses on predicting insurance prices by employing PyTorch and machine learning algorithms. Trained on relevant datasets, the model predicts insurance premiums based on various factors.
Tools Used: Numpy, PyTorch
ML Algorithm: Neural Network, Linear Regression

Project Link

16. House Price Prediction using PyTorch

Utilising PyTorch and machine learning, this project predicts house prices based on relevant datasets from kaggle. The model is trained to predict house prices using various features.
Tools Used: Numpy, PyTorch
ML Algorithm: Neural Network, Linear Regression

Project Link