Portfolio – Computer Vision Focus

1. Trash Bag Segmentation, Detection, Classification, Tracking, and Counting in 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 YOLOv8.

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**.

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 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

Video of the Project: Watch Here
Festival Clip: Watch Here
President Award Certificate: View Award

---

2. Football Analysis System – YOLO11 Fine-Tuning & Video Tracking

Developed a complete football match analysis pipeline from scratch, focusing on detecting and tracking **players, referees, and the ball** across full-length videos. Fine-tuned a **YOLO11 model** for custom detection tasks, achieving robust performance on video-based

data. Implemented a modular training pipeline in Jupyter notebooks, where the model was trained and validated on football-specific datasets. Integrated **tracking, bounding box overlays, and video output generation** to support detailed post-match analysis. Final system enables automated detection, visual analysis, and statistics extraction for sports AI applications.
More details in: View Project

---

3. 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: GitHub Repository
Project Demo Video: Watch on YouTube

---

4. Multi-class Weather Classification using PyTorch and CNN

Employing PyTorch and Convolutional Neural Networks, this project categorises images into various weather conditions. By utilising a custom dataset of weather from Kaggle 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: GitHub Repository

---

5. 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

---

6. Stable Diffusion Image Generation System – Text-Guided Image Transformation

This project experiments with **Stable Diffusion transformers** for text-guided image generation and transformation. It implements fine-grained control mechanisms and leverages **GPU acceleration** for efficient processing, enabling high-quality and customizable image outputs from textual prompts.

Project Link: GitHub Repository

---

7. 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, OpenAI Whisper, MoviePy, LLaVA Model, LanceDB
API Keys Used: OpenAI API, Prediction Guard API
ML Algorithms: BridgeTower Embeddings, RAG System

Certificate: View Certificate
Project Link: GitHub Repository

NLP Focused Projects

1. Uzbek-English Pretrained Language Model (140M Parameters)

Collected a diverse Uzbek-English dataset and trained a Byte Pair Encoding (BPE) tokenizer from scratch, resulting in a vocabulary size of ~62,016 tokens. Built a decoder-only modified Transformer architecture with 140.7M parameters and a 1024-token context window. Developed the full AI training pipeline and pre-trained the model using open-source multilingual datasets on 2× NVIDIA A100 GPUs for 16.5 hours. Reached ~3.5% cross-entropy loss during pretraining; the model currently supports next-token generation. Planning to fine-tune the model for instruction-following and downstream task alignment. Demo includes both short and long prompt generation samples.

Project Link: GitHub Repository

---

2. LLMs from Scratch – GPT-2 Implementation & Fine-Tuning

Completed the full implementation of a GPT-2 model from scratch based on *"Build LLMs from Scratch"* by Sebastian Raschka. Developed essential components including text preprocessing, positional encodings, multi-head self-attention, and the GPT-2 architecture. Pretrained the model using causal language modeling (CLM) on unlabeled data, fine-tuned it for spam classification, and further applied instruction tuning to enable natural language prompt-following. Gained hands-on experience in building, training, and adapting large language models for both generative and classification tasks.

Project Link: GitHub Repository

---

3. LLaMA Customer Support Assistant – Fine-tuned Conversational AI

A specialized customer support AI system built by fine-tuning the 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 2× 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 2× faster training through Unsloth optimization
- Efficient memory usage under 8GB RAM
- Natural response generation for customer queries

Project Link: GitHub Repository

---

4. Neural Conversational Chatbot with Attention Mechanisms

Developed an advanced chatbot using sequence-to-sequence neural networks and attention mechanisms in PyTorch. 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 chatbot with a bidirectional GRU encoder for input understanding and a GRU decoder with attention for response generation
- Context Understanding: Implemented Luong's global attention mechanism to focus on relevant parts of the input during generation
- Efficient Processing: Optimized training through batch processing, sequence handling, and carefully tuned parameters (500 hidden units, 2 layers, 0.1 dropout)

Project Features:

- Natural conversations with context-aware dialogue generation
- Performance analysis through attention pattern visualization and response evaluation
- Modular design with separate encoder, decoder, attention, and utility modules for maintainability

Tools Used: PyTorch, Python, NLTK, NumPy, Pandas
ML Algorithms: Sequence-to-Sequence Learning, GRU Networks, Attention Mechanisms

Project Link: GitHub Repository

---

5. 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 adapting to temperature, air quality, and conditions
- Dynamic schedule adjustment for optimized travel experiences
- Comprehensive error handling and input validation
- Modular architecture for extensibility

Tools & Technologies: Python 3.7+, PyTorch, LangChain, Ollama, LLaMA 3.2, JSON

Results:

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

Project Link: GitHub Repository

---

6. Sentiment Analysis using Deep Learning and Traditional ML

Implemented 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, leveraging a sophisticated LSTM-based RNN architecture alongside traditional classifiers.

Technical Strategy:

- **LSTM-RNN Architecture:** Built a neural network with bidirectional LSTM, customizable layers, word embeddings, dropout regularization, and an 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 combining deep learning and traditional ML approaches
- Performance optimization with packed padded sequences and dropout regularization
- Comparative analysis with evaluation of different model performances
- Interactive Jupyter notebook interface for 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: GitHub Repository

---

7. Local RAG Chatbot

Built a Retrieval-Augmented Generation (RAG) based chatbot that answers questions from user-provided documents such as PDFs, CSVs, and web content. The system integrates document retrieval with language model generation, offering accurate and context-aware responses through an interactive Streamlit interface.

Key Features:

- **Document Upload:** Supports PDF and CSV ingestion for knowledge grounding
- **Web Scraping:** Extracts information from specified URLs
- **ChromaDB Integration:** Stores and retrieves embeddings for efficient semantic search
- **Ollama LLM:** Powers conversational AI with advanced language models **(`exaone3.5:2.4b`)**
- **Streamlit Interface:** Provides a user-friendly web-based chatbot experience

Technical Strategy:

- Utilized **`mxbai-embed-large`** for embedding generation
- Implemented ChromaDB for persistent vector storage and retrieval
- Designed a conversation buffer to handle multi-turn interactions

- Configurable `.env` file for model selection and chat history length

Results:

- Seamless conversational interaction grounded on uploaded or scraped content
- Multi-document support with efficient embedding search
- Memory-efficient deployment with configurable history buffer

Tools & Technologies: Python, Streamlit, LangChain, Ollama, ChromaDB, Conda

Project Link: GitHub Repository