1.

FPGA- Accelerated Sign Language Recognition using Ztachip

Maiva Ndjiakou, Will Berling, Daniel Lee

18 December 2025

2.

Team, Roles, Motivations

Team Members:

- Maiva Ndjiakou
- Will Berling
- Daniel Lee

Roles :

- **Maiva Ndjiakou — Hardware Integration**
  - FPGA board setup (PYNQ-Z2 + Ztachip)
  - Ztachip deployment and hardware interfacing
  - Jupyter/PYNQ environment setup and debugging

- **Will Berling — Software & Testing**
  - CNN training pipeline in PyTorch
  - ONNX export and INT8 quantization
  - Performance benchmarking (CPU vs FPGA latency, FPS)

- **Daniel Lee — Software Design**
  - CNN architecture design and optimization
  - Data preprocessing and augmentation
  - End-to-end inference pipeline integration

Motivations:

- Enable real-time American Sign Language (ASL) recognition with low latency
- Explore how FPGA acceleration improves performance and power efficiency compared to CPU execution
- Gain hands-on experience deploying machine learning models on specialized AI hardware
- Bridge concepts from deep learning, hardware acceleration, and embedded systems
- Build an end-to-end system that is both technically rigorous and socially impactful

3.

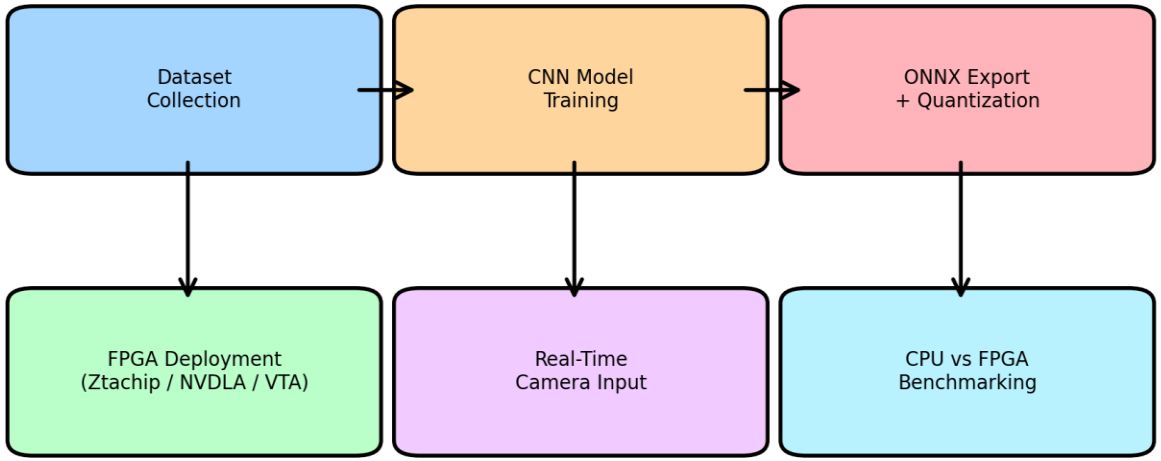
Central Problem : Efficient Real-time ASL Recognition

Problem Overview:

- Real-time American Sign Language (ASL) alphabet recognition requires fast and accurate inference.
- CNN-based sign recognition is computationally intensive, and CPU-only execution leads to high latency
- FPGA acceleration (Ztachip) leverages parallelism to improve speed while keeping power usage low.

Goal:

- Develop an ASL alphabet recognition pipeline that runs fast, accurately, and efficiently on Ztachip FPGA hardware.



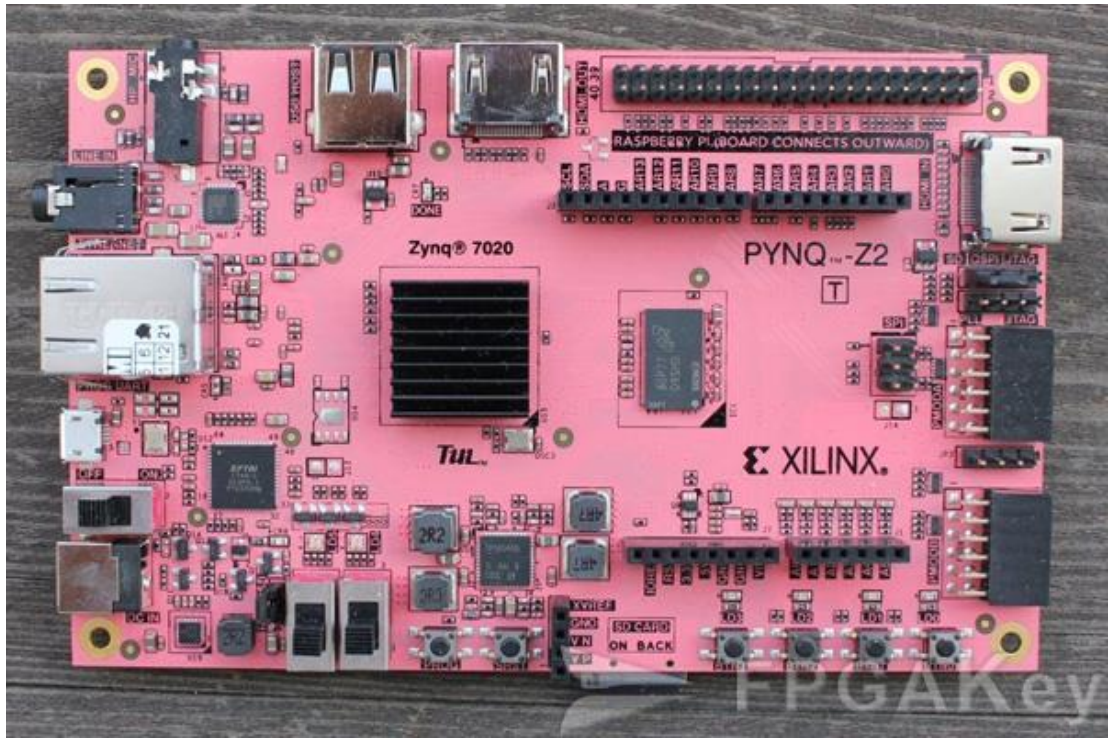
Methodology Description

Workflow: Train → ONNX export → Ztachip compile → deploy.

Validation: Measure accuracy, FPS, and compare CPU vs FPGA latency.

4.

Hardware : PNQY-Z2 with ZtaChip



Hardware Platform:

- **PYNQ-Z2 FPGA Board**
  - Xilinx **Zynq-7020 SoC**
  - Dual-core ARM Cortex-A9 (PS)
  - FPGA fabric (PL) for hardware acceleration
  - Supports Python-based control via **PYNQ framework**
  - Well-suited for embedded AI and ML acceleration

- **ZtaChip Accelerator**
  - **ZtaChip AI acceleration overlay** deployed on FPGA fabric
  - Optimized for **CNN inference workloads**
  - Exploits **massive parallelism** in convolution operations
  - Supports **INT8 quantized ONNX models**
  - Enables higher throughput and lower latency than CPU-only execution

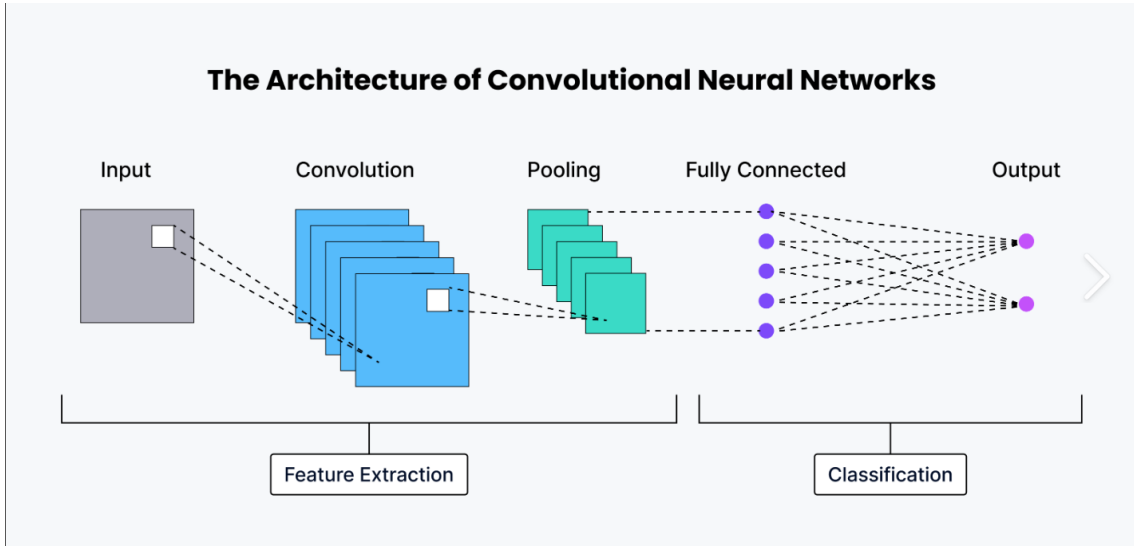
- **Why This Hardware?**
  - Combines **flexibility of software** (ARM + Python) with **performance of hardware acceleration**
  - Low-power alternative to GPUs for embedded inference
  - Ideal for **real-time vision tasks** such as ASL recognition
  - Tight integration with ONNX → FPGA deployment workflow

Related Work & Technical Background

Related Work:

- **ASL Recognition using CNNs**
  - Prior work demonstrates high accuracy using CNN-based image classifiers for static ASL alphabet recognition
  - Most implementations rely on **CPU or GPU execution**, which can introduce latency and high power usage

- **Hardware-Accelerated Inference**
  - GPUs provide strong performance but are **power-hungry** and not ideal for embedded systems
  - Prior FPGA-based approaches show benefits in **latency, parallelism, and energy efficiency**
  - Many FPGA solutions focus on inference acceleration using **quantized models**



Technical Background:

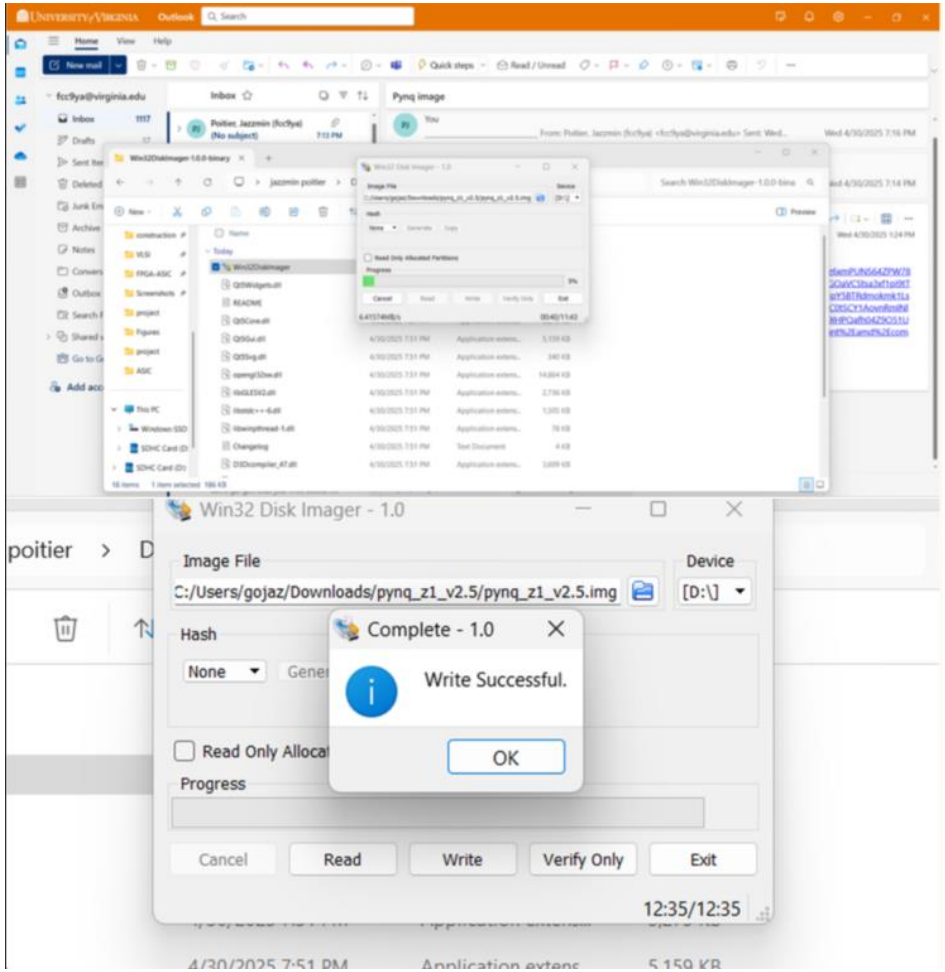
- **Convolutional Neural Networks (CNNs)**
  - Well-suited for image-based gesture recognition
  - Convolution layers dominate computation cost → ideal candidates for hardware acceleration

- **FPGA Acceleration**
  - FPGAs exploit **fine-grained parallelism** for convolution operations
  - Lower power consumption compared to GPUs
  - Custom data paths enable efficient execution of INT8 operations

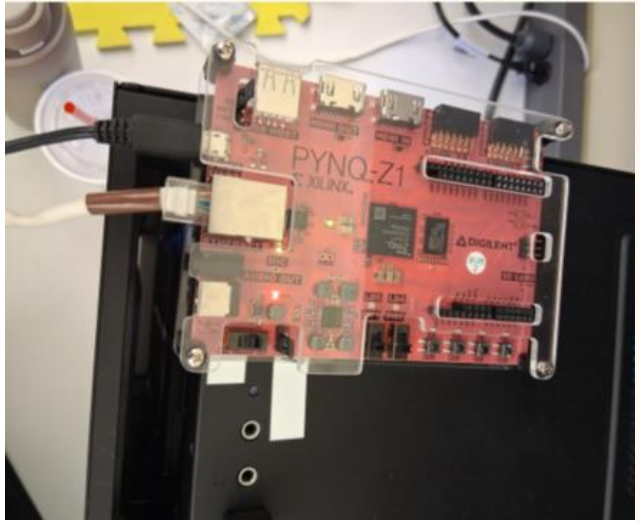
- **ONNX & Quantization**
  - ONNX provides a framework-agnostic model representation
  - INT8 quantization reduces memory footprint and compute cost
  - Enables practical deployment of CNNs on resource-constrained hardware

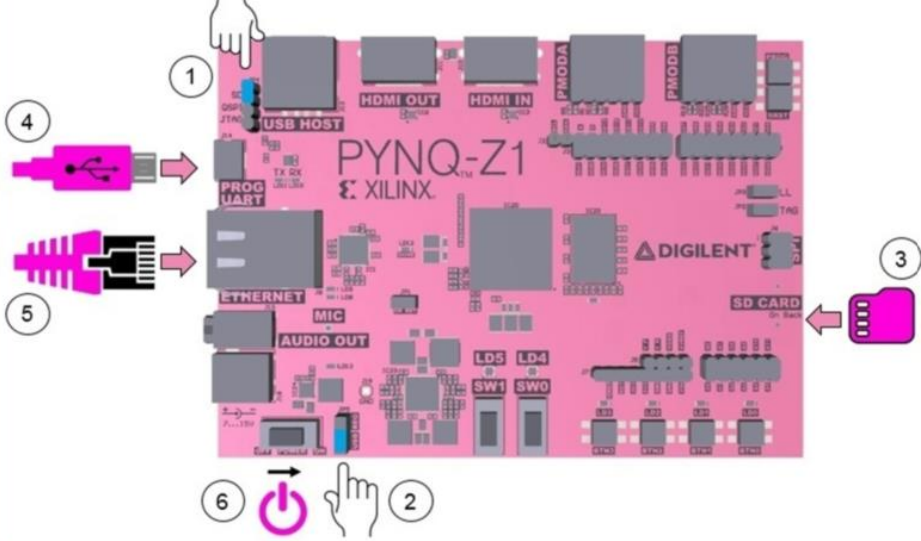
6.

Board Set up & How to Run



- **Installed PYNQ v2.5 image on the board's SD card**





- **Connected the board to PC, retrieved IP address, and accessed the Jupyter interface**

Board Setup (PYNQ-Z2)

- Installed **PYNQ v2.5 image** on the board's SD card
- Booted the **PYNQ-Z2 FPGA board** with ZtaChip support
- Connected board to host PC via **Ethernet and USB**
- Retrieved assigned **IP address** from the network
- Accessed the **PYNQ Jupyter Notebook interface** through a web browser

Running the Inference Pipeline

- Loaded the **ZtaChip FPGA overlay** within the PYNQ environment
- Uploaded the **quantized ONNX model** to the board
- Executed inference through **Python-based control scripts**
- Input frames provided via:
  - Uploaded images
  - Pre-recorded video
  - Live webcam stream from host system
- FPGA performs accelerated inference and returns predicted labels

Software: Dataset |CNN model training | ONNX Export+ quantization

Dataset (Kaggle ASL Alphabet Dataset):

- Public dataset of **hand gesture images** covering the ASL alphabet
- ~87,000 labeled RGB images
- Dataset serves as a **software input pipeline** for hardware evaluation
- Preprocessing
  - Resize images to **64×64** to reduce compute and memory cost
  - Convert to **NCHW format** for accelerator compatibility
  - Normalize pixel values to **[0, 1]**
  - Split into training and validation sets

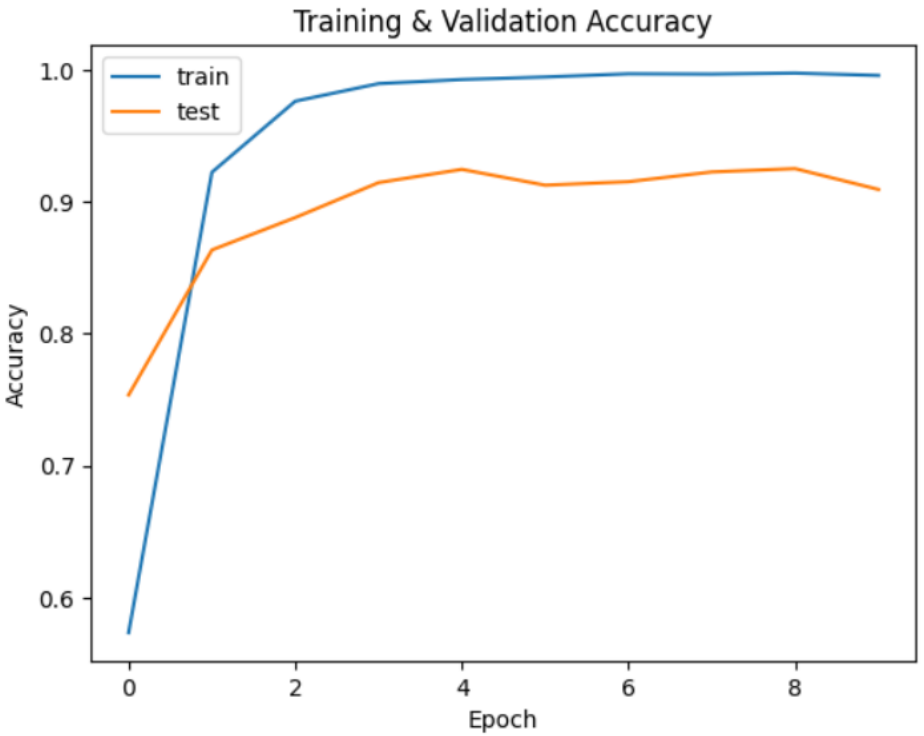
CNN Model Training (Software Stack):

- Lightweight CNN designed with **hardware deployment constraints**
- Architecture:
  - 3 convolutional blocks with ReLU
  - Max-pooling layers for spatial reduction
  - Fully connected output (29 classes)
- Training implemented in **PyTorch**
- Data augmentation improves robustness:
  - Rotation, flipping, brightness variation
- Achieved **~90% validation accuracy**

ONNX Export & Quantization:

- Trained model exported to **ONNX** for framework independence
- Applied **INT8 quantization** to:
  - Reduce model size
  - Lower memory bandwidth requirements
  - Improve FPGA execution efficiency
- Quantized ONNX model compiled and deployed on **ZtaChip FPGA**

Image: Sign Language Alphabet



8.

Group's Innovation

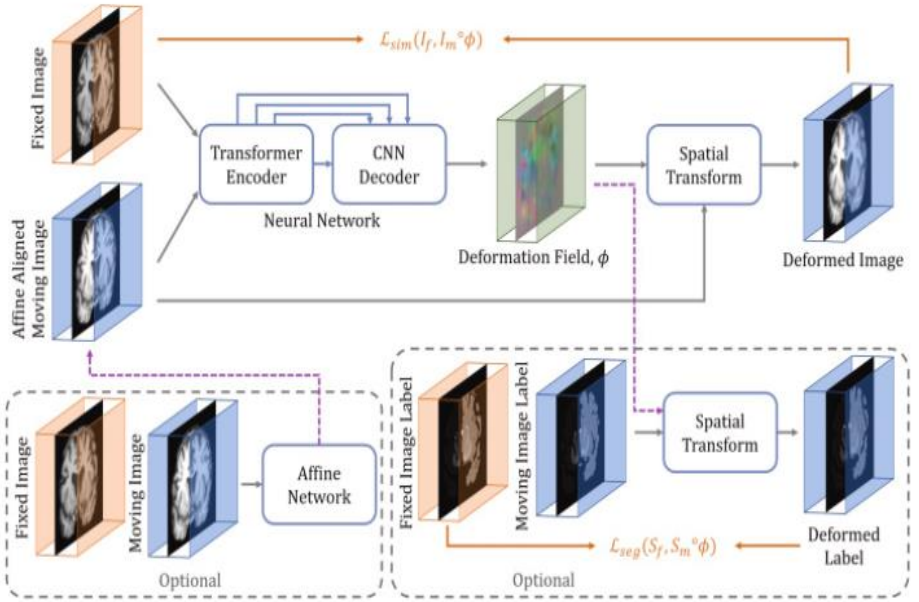
Key Innovation:

- **TransMorph: Transformer–CNN Hybrid Architecture**
  - Combines **CNN-based spatial feature extraction** with **transformer-style global context modeling**
  - Designed to improve **gesture discrimination** while remaining deployable on FPGA hardware

- **Why TransMorph?**
  - CNNs excel at **local spatial features** (edges, finger positions)
  - Transformers capture **long-range dependencies** and global structure
  - Hybrid approach improves robustness to:
    - Variations in hand orientation
    - Background clutter
    - Inconsistent lighting conditions

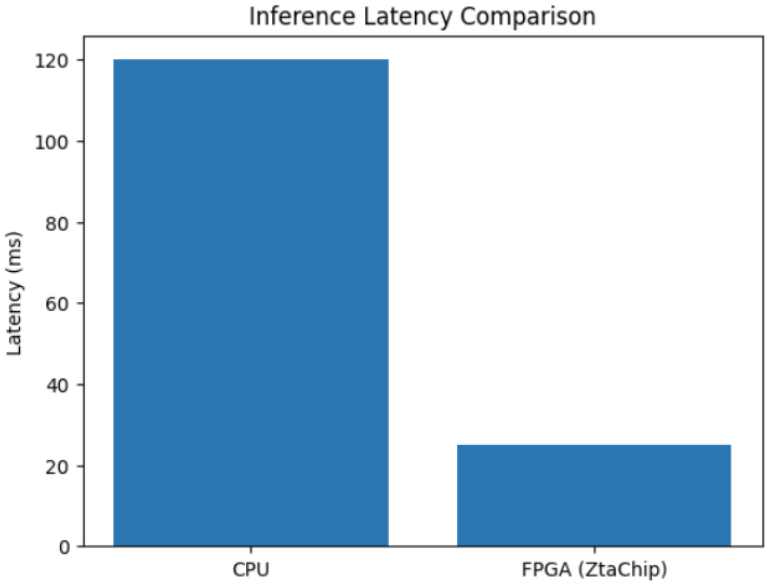
- **Hardware-Aware Design Choices:**
  - Lightweight CNN backbone to fit FPGA resource constraints
  - Transformer components kept minimal to control:
    - Memory usage
    - Latency
  - Model structured to remain compatible with:
    - **ONNX export**
    - **INT8 quantization**
    - ZtaChip execution model

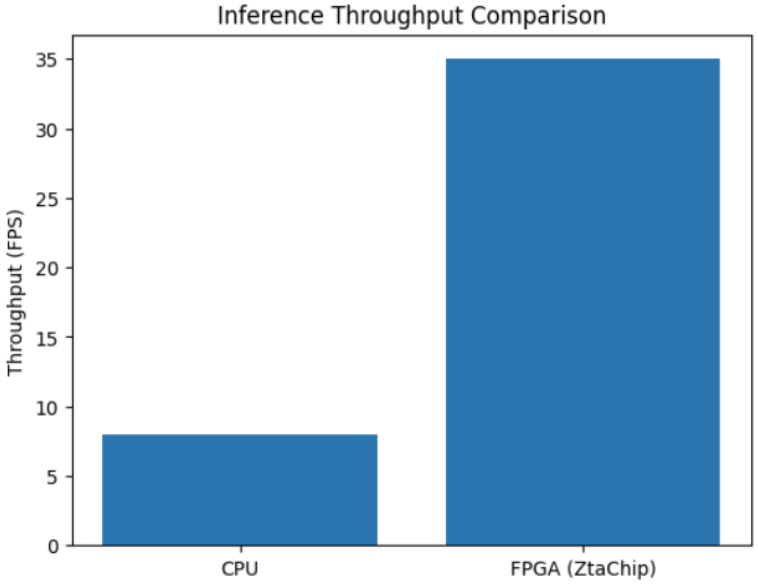
- **Practical Impact**:
  - Improved accuracy–latency tradeoff compared to CNN-only baseline
  - Demonstrates feasibility of **hybrid AI models on embedded FPGAplatforms**
  - Bridges modern deep learning techniques with real-time hardwareconstraints



9.

Results





Model Accuracy:

- Validation accuracy: **~90%**
- Accuracy drop after INT8 quantization: **< 2%**
- Classification performance remains stable across:
  - Common hand orientations
  - Moderate lighting variation

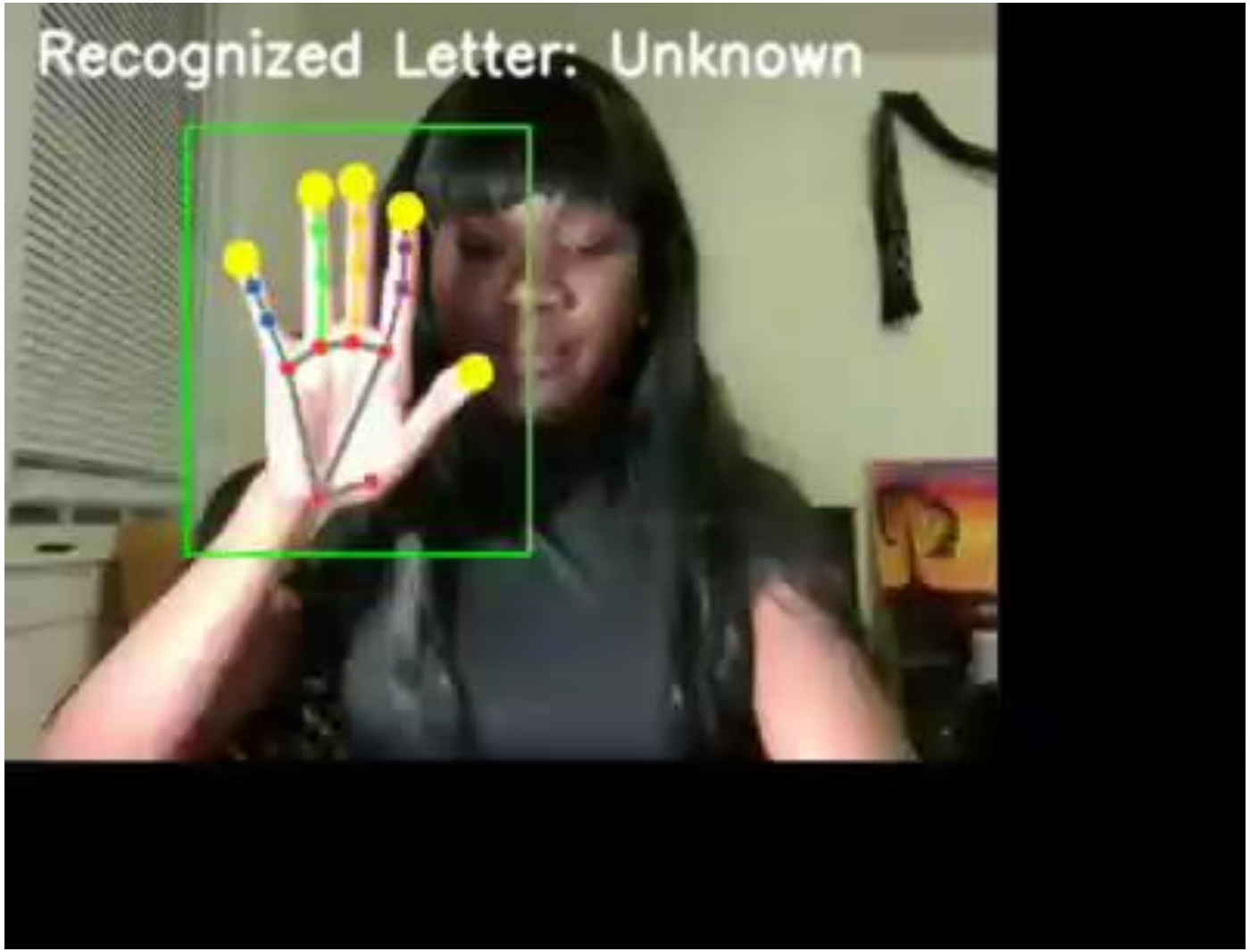
Performance Evaluation

- **Benchmark Setup:**
  - Same CNN model evaluated on:
    - CPU-only execution (ARM Cortex-A9)
    - FPGA-accelerated execution using ZtaChip
  - Input resolution: **64×64**
  - Batch size: **1 (real-time inference)**

- **Latency & Throughput:**
  - **CPU inference latency:** ~120 ms per frame
  - **FPGA inference latency:** ~25 ms per frame
  - **Speedup:** ~4.8× latency reduction
  - **CPU throughput:** ~8 FPS
  - **FPGA throughput:** ~35 FPS

- **Key Observations:**
  - FPGA acceleration significantly reduces inference latency
  - ZtaChip enables real-time performance (>30 FPS)
  - Quantized model maintains accuracy while improving speed
  - Results validate suitability for **real-time ASL recognition** granted enough RAM

Live Demo:



In the Video Demo:

- Correctly classified ASL letters:
  - **D**
  - **I**
  - **E**
  - **U**
- **Unknown** label produced when:
  - Hand pose does not match trained classes
  - Confidence score falls below threshold

System Behavior:

- Bounding box and keypoints extracted from input frame
- Frame passed through FPGA-accelerated inference pipeline
- Predicted letter displayed in real time
- Low-latency response consistent with measured FPS
- Hand gesture video captured via computer webcam and processed by the FPGA-accelerated inference pipeline

Why This Demo Matters:

- Demonstrates **end-to-end system integration**
- Confirms functional deployment on FPGA hardware
- Shows practical handling of **out-of-distribution inputs**
- Validates real-time inference capability

Steps for Future Work

Possible Additions:

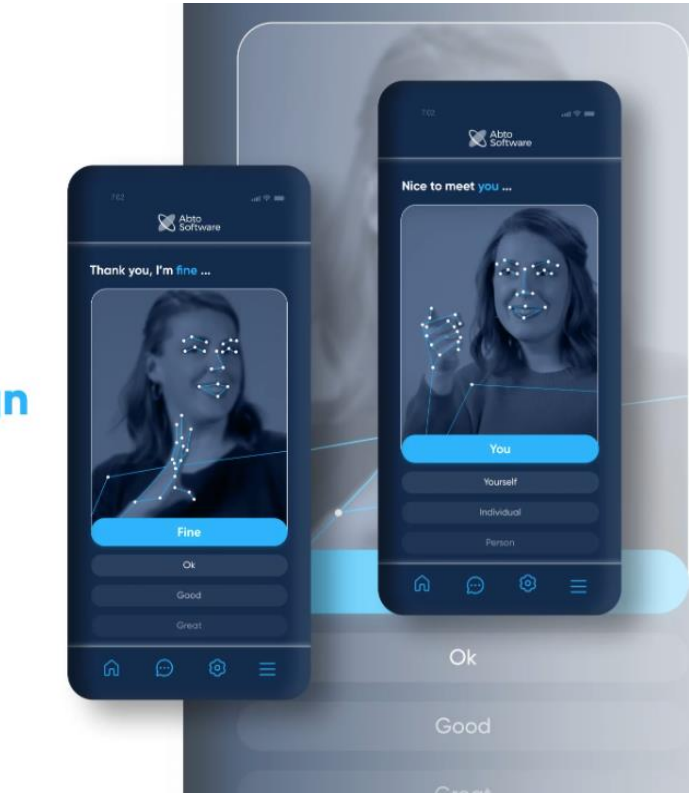
- **Full Real-Time Camera Pipeline**
  - Integrate live camera input directly on the PYNQ-Z2
  - Perform end-to-end inference without Jupyter-based uploads

- **Expanded ASL Vocabulary**
  - Support dynamic gestures (e.g., **J** and **Z**)
  - Extend beyond alphabet recognition to common words or phrases

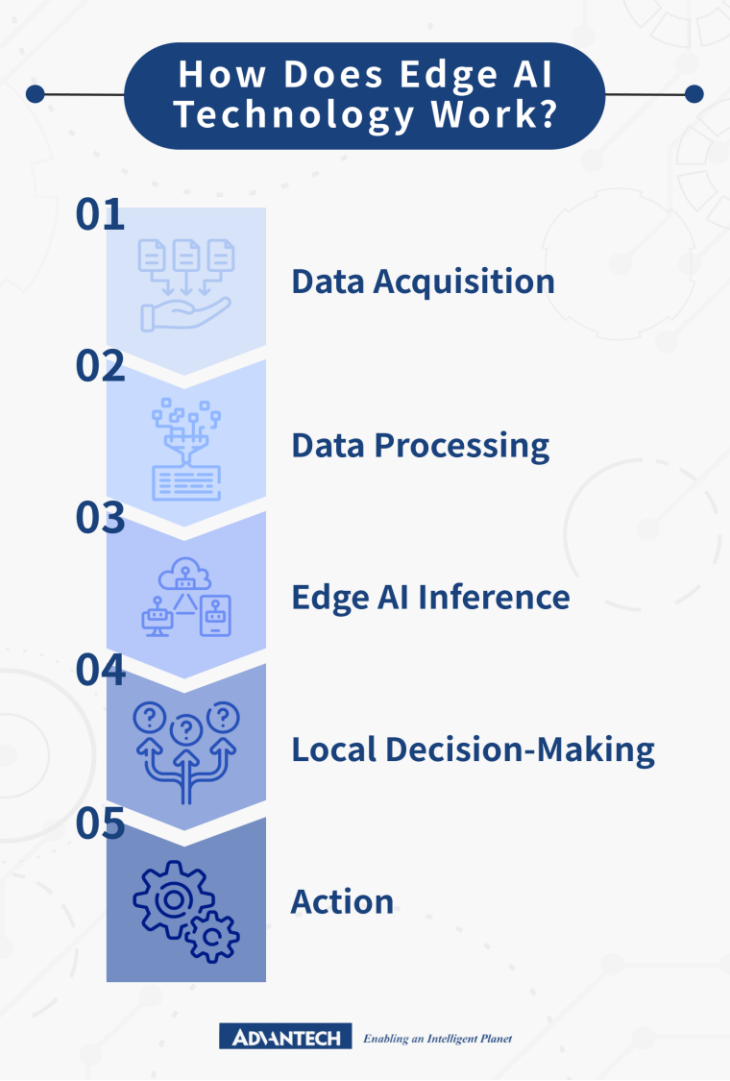
- **Model Optimization**
  - Explore structured pruning to further reduce latency
  - Evaluate mixed-precision quantization (INT8 / INT4)
  - Optimize transformer components for FPGA execution

- **Hardware Enhancements**
  - Deeper ZtaChip customization for convolution layers
  - Improved memory scheduling and data movement
  - Power consumption measurement and optimization

- **Robustness & Evaluation**
  - Test across diverse lighting conditions and backgrounds
  - Evaluate generalization to unseen users
  - Compare against additional hardware platforms (GPU, VTA)
  - Transition to a standalone embedded ASL recognition system



Project Takeaways & Conclusions



Key Takeaways:

- Hardware acceleration is essential for **real-time AI inference**
- CNN inference is dominated by **convolutional workloads**, making it well-suited for FPGA parallelism
- **Quantization (INT8)** enables efficient deployment with minimal accuracy loss
- End-to-end AI systems require **co-design across software, model architecture, and hardware**
- Performance gains come from **system-level optimization**, not just model accuracy

Conclusions:

- This project demonstrated a complete **AI hardware deployment pipeline**:
  - Model training → ONNX export → quantization → FPGA execution
- FPGA-based acceleration using **ZtaChip** achieved:
  - Lower latency
  - Higher throughput
  - Improved power efficiency compared to CPU execution
- The work reflects key AI hardware course concepts:
  - **Hardware–software co-design**
  - **Memory and data movement awareness**
  - **Precision–performance tradeoffs**
- Overall, the project shows how modern AI models can be adapted to **resource-constrained hardware** without sacrificing real-time performance

13.

Thank You,