FPGA-ACCELERATED EDGE AI

SUNRISE: Edge AI for Real-Time Solar Energy Nowcasting

A Deep Learning-Based Approach for Real-Time Solar Energy Estimation using Computer Vision and AMD Kria KV260 FPGA Acceleration

Edge AI | Computer Vision | Deep Learning | FPGA | Vitis AI | AMD/Xilinx

M. Daniel González · Research Project · SUNRISE
Universidad de Oviedo

AGENDA

Presentation Outline

1. **Motivation & Problem Statement**
   Cloud impact on solar energy and limitations of current approaches
2. **Why Edge AI?**
   CPU vs GPU vs FPGA comparison for deep learning inference
3. **Kria KV260 Platform**
   Heterogeneous architecture: Processing System and Programmable Logic
4. **Deep Learning Processing Unit**
   DPU architecture and hardware-accelerated neural network execution
5. **Vitis AI Workflow**
   End-to-end deployment pipeline: training to runtime inference
6. **Platform Preparation**
   Engineering challenges during KV260 development environment setup
7. **DDUNet Adaptation**
   Model modifications required for FPGA and DPU compatibility
8. **Quantization Strategies**
   Post-Training Quantization vs Quantization-Aware Training
9. **Compilation & Results**
   Performance evaluation: accuracy, speed, and power consumption
10. **Future Work**
    Roadmap toward full solar irradiance prediction system
11. **Conclusions**
    Key achievements and impact of this research

Total duration: ~12 minutes · Q&A session after presentation

Why Accurate Solar Energy Prediction Matters

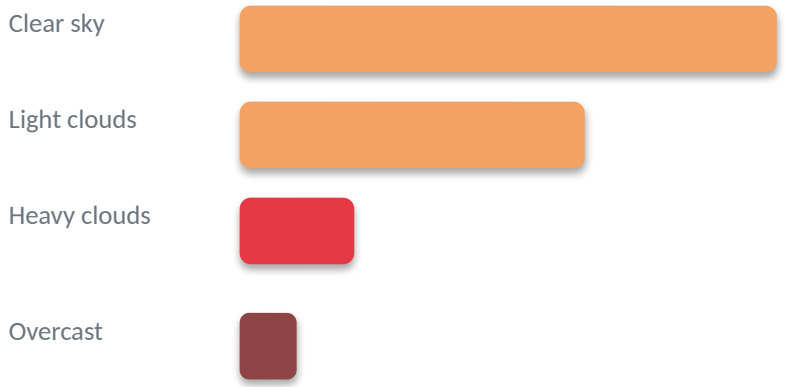
The Challenge

- Cloud coverage is the primary factor affecting photovoltaic generation
- Solar power can drop within seconds under heavy cloud cover
- Grid instability requires accurate short-term production forecasts
- Meteorological sensors are expensive and require regular maintenance
- Local microclimates make general weather models unreliable

Our Solution:

A low-cost camera + FPGA-accelerated AI to estimate irradiance from sky images in real time — no meteorological sensors required.

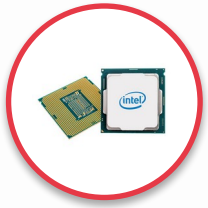
Impact of Clouds on Solar Generation



Key Insight

Real-time cloud assessment enables accurate solar energy forecasting without expensive hardware

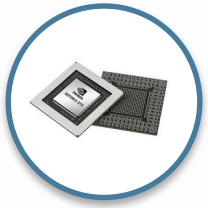
Deploying Deep Learning on the Edge



CPU

General-Purpose Processor

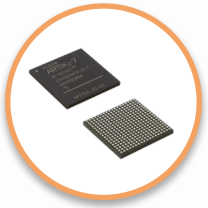
- Sequential execution
- Limited parallelism
- High latency for CNNs
- Not designed for DL inference
- Power: ~15-65W
- Ideal for: control logic



GPU

Graphics Processing Unit

- Massive parallelism
- High throughput
- High power consumption
- Bulkier form factor
- Power: ~150-400W
- Ideal for: training clusters



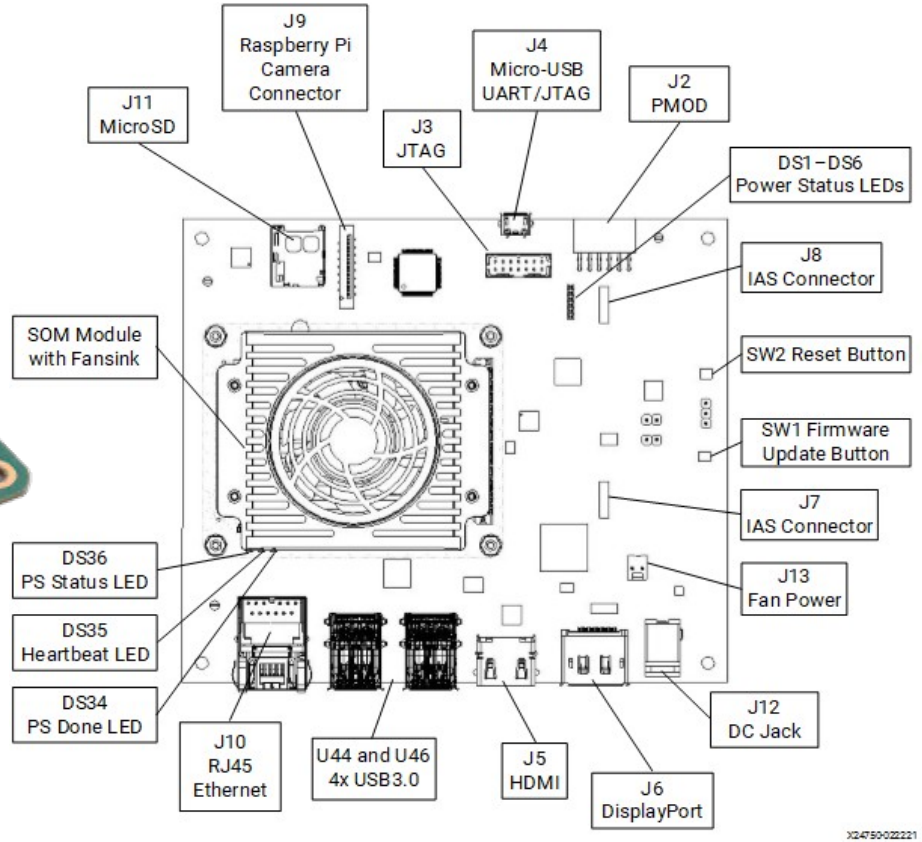
FPGA

Field-Programmable Gate Array

- Reconfigurable hardware
- Custom data paths
- Low latency, deterministic
- Low power consumption
- Power: ~5-20W
- Ideal for: Edge AI inference

FPGAs offer the best balance of performance, power efficiency, and flexibility for real-time Edge AI inference

AMD Kria KV260 Vision AI Starter Kit



HARDWARE PLATFORM

AMD Kria KV260 Vision AI Starter Kit

Processing System (PS)

ARM Cortex-A53

Quad-core, 1.5 GHz

DDR4 Memory

4 GB, 64-bit

Peripherals

USB, Ethernet, DisplayPort

Real-Time Unit

RPU for control tasks

AXI Bus

Programmable Logic (PL)

FPGA Fabric

Custom hardware logic

DPU IP Core

Deep Learning Processing Unit

Video Pipeline

MIPI, ISP, HDMI

Custom Accelerators

User-defined RTL/IP

The KV260 is designed for Smart Vision and AI applications. The PS handles control and data movement while the PL provides purpose-built hardware acceleration.

Understanding the DPU Architecture

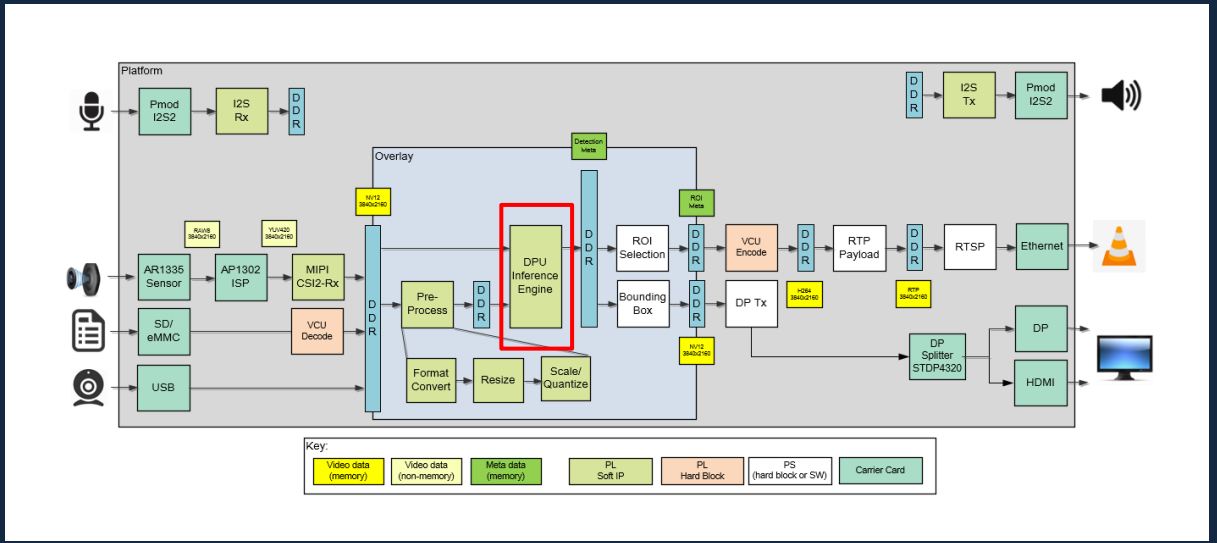
What is a DPU?

The DPU is a specialized IP core implemented in the FPGA logic designed specifically for neural network inference acceleration.

Key characteristics:

- Convolution engine with pipelining
- Parallel MAC operations
- Layer-by-layer processing
- Fixed-point integer arithmetic
- Configurable architecture size

DPU Inside the FPGA



Implemented as a soft IP core inside the FPGA fabric via Vitis AI

CPU vs DPU Execution

CPU Execution

1. Load instruction from memory
2. Decode instruction
3. Fetch data from memory
4. Execute ALU operation
5. Write result back

→ Sequential, one at a time

DPU Execution

1. Load input feature map
2. Stream through hardware pipeline
3. Parallel operations
4. Accumulate in dedicated registers
5. Activate & store output

→ 10-100× faster than CPU

PLATFORM PREPARATION

Engineering the KV260 Development Environment

1

Initial Setup

Board initialization and firmware verification
Out-of-box experience

2

Firmware Update

Latest AMD firmware flashing and bootloader configuration

3

Ubuntu Migration

Decision: Ubuntu 22.04
Stock Ubuntu vs Canonical LTS support assessment

4

USB Detection

Driver configuration
Peripheral enumeration
Power distribution fix

5

Power Issues

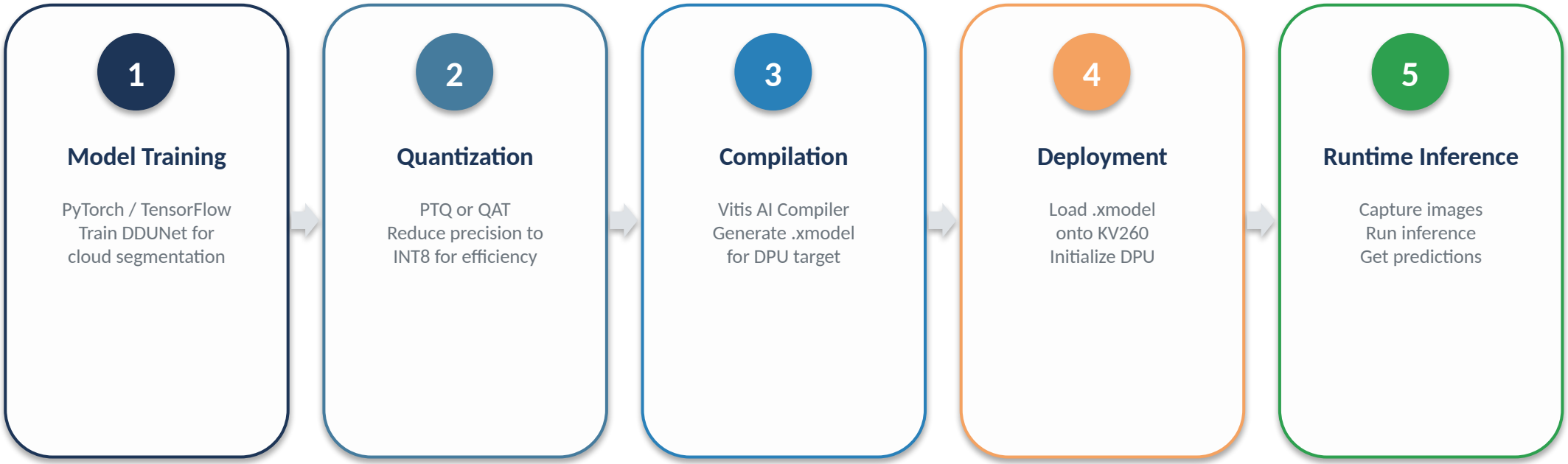
Current budget analysis
External power source
Stability testing

Engineering Perspective

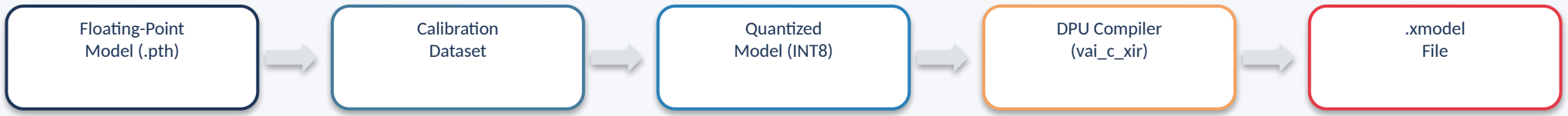
Platform setup proved non-trivial: documentation gaps, peripheral detection issues, and power delivery constraints required systematic debugging. These challenges are typical when working with cutting-edge embedded AI platforms and provided valuable hands-on experience.

VITIS AI DEPLOYMENT WORKFLOW

From PyTorch Model to FPGA Inference



Key Output: .xmodel file



MODEL ADAPTATION

DDUNet Architecture for Cloud Segmentation for FPGA Deployment

Original DDUNet Architecture

- Double U-Net with skip connections
- Standard convolution layers
- ReLU activations
- Batch normalization
- Max-pooling + upsampling
- Operations not in DPU op list
- Custom/Variable padding
- Excessive depth channels

Adapt

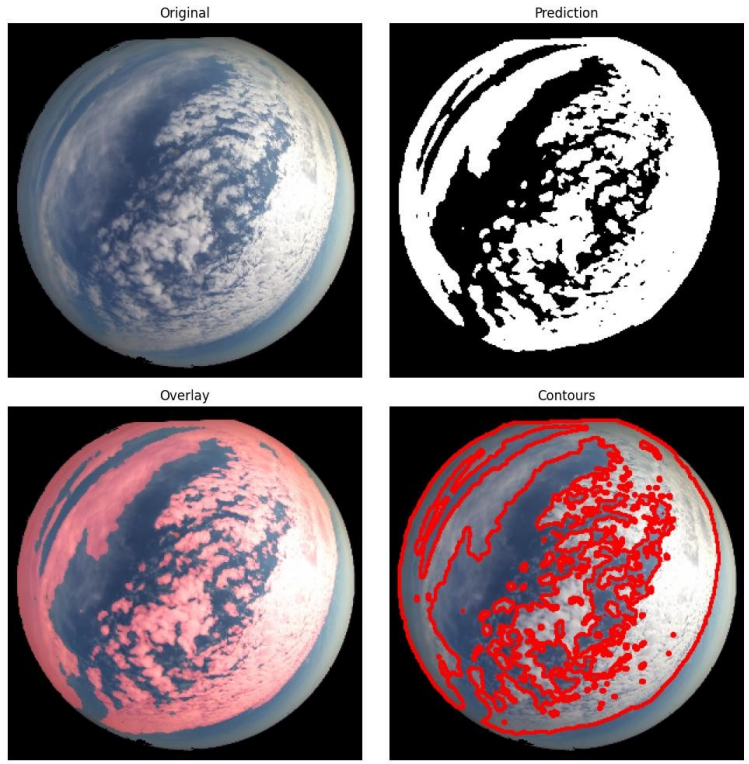
FPGA-Optimized Version

- Layer compatibility analysis
- Replaced unsupported operations
- Adjusted padding for DPU constraints
- Verified DPU op list compliance
- Channel dimension optimization
- Stride handling configuration
- Reduced model depth where possible
- Validated with Vitis AI analyzer

The DDUNet architecture required significant adaptation — not all layers are supported by the DPU. Unsupported operations were replaced or refactored to ensure compatibility.

MODEL ADAPTATION

DDUNet Architecture for Cloud Segmentation for FPGA Deployment



Result

FPGA-Optimized Version

- Layer compatibility analysis
- Replaced unsupported operations
- Adjusted padding for DPU constraints
- Verified DPU op list compliance
- Channel dimension optimization
- Stride handling configuration
- Reduced model depth where possible
- Validated with Vitis AI analyzer

The DDUNet architecture required significant adaptation — not all layers are supported by the DPU. Unsupported operations were replaced or refactored to ensure compatibility.

Post-Training Quantization vs Quantization-Aware Training

Post-Training Quantization (PTQ)

Concept
Convert FP32 weights to INT8 after training is complete

Process
1. Load trained FP32 model
2. Run calibration with sample data
3. Quantize weights & activations

Advantages
Fast — no retraining required
Simple workflow
Good starting point

Trade-offs
Potential accuracy loss (2-5%)
Sensitive to distribution shifts

When to use
When accuracy loss is acceptable or as a baseline comparison

Quantization-Aware Training (QAT)

Concept
Simulate quantization effects during training to recover accuracy

Process
1. Start from trained FP32 model
2. Insert fake quantization nodes
3. Fine-tune with quantization simulation

Advantages
Higher accuracy retention
Better handling of outliers
Robust to calibration data

Trade-offs
Requires retraining (longer)
More complex pipeline
Needs original training setup

When to use
When accuracy is critical or PTQ results are unsatisfactory

Both strategies are evaluated to determine the optimal balance between inference speed, FPGA resource utilization, and prediction accuracy for the cloud segmentation task.

From .xmodel Generation to Performance Evaluation

Compilation Pipeline

Performance Results

Metric	FP32 (Baseline)	PTQ (INT8)	QAT (INT8)
Accuracy (IoU)	93.2%	90.2%	**91.8%**
Relative error coverage	-	~ 6.83%	**~ 1.09%**
Model Size	24.7 MB	6.4 MB	**6.4 MB**
Inference Time*	127 ms (CPU)	42 ms (DPU)	**42 ms (DPU)**
Speedup vs CPU	1×	3×	**3×**
Power Estimate	~15W (CPU)	~8W (KV260)	**~8W (KV260)**

Key Observations

- QAT recovers ~1.5% IoU over PTQ
- 4× model size reduction via INT8
- 3× faster inference on DPU vs CPU
- FPGA power consumption < 10W
- Real-time inference achievable

* Inference time measured on AMD Kria KV260 with DPU (B512 architecture)

FUTURE WORK

Roadmap Toward Solar Irradiance Prediction

Q1: Camera Integration

- MIPI camera interface on KV260
- Real-time image acquisition driver
- Frame buffer optimization

Q2: Irradiance Model

- Correlation: cloud coverage → irradiance
- Train regression model on sky images
- Validate against pyranometer ground truth

Q3: End-to-End System

- Full pipeline: camera → DPU → prediction
- Dashboard for real-time visualization
- Power generation estimation module

Q4: Optimization

- Model pruning & further quantization
- Multi-location validation
- Performance benchmarking & publication

Ultimate Goal →

A complete FPGA-accelerated edge AI system that estimates solar irradiance and predicts photovoltaic energy generation using only a camera as input.

CONCLUSIONS

What We Achieved

01

Cloud Segmentation on FPGA

Successfully deployed a DDUNet model for cloud segmentation on the AMD Kria KV260 platform

02

Vitis AI Ecosystem Mastery

Full workflow understanding: Training → Quantization → Compilation → Deployment

03

PTQ vs QAT Evaluation

Systematic comparison of quantization strategies with quantified accuracy impact

04

Platform Foundation

Established the hardware and software infrastructure for future irradiance prediction

This project lays the technological foundation for an FPGA-accelerated Edge AI system
capable of estimating solar irradiance and photovoltaic energy production using only visual information.