Design Document

HALO: High Autonomous Low-SWaP Operations

Team Members

Sloan Hatter (shatter2022@my.fit.edu)
Blake Gisclair (bgisclair2022@my.fit.edu)

Faculty Advisor

Dr. Ryan T. White (rwhite@fit.edu)

September 29, 2025

Table of Contents

1. Introduction
    1.1. Purpose
    1.2. Scope
2. System Design
    2.1. Vision Transformer Model
    2.2. Processing Pipeline
        2.2.1. Training/Quantization-Aware Training (QAT)
        2.2.2. On-Device Inference

1. Introduction

1.1. Purpose

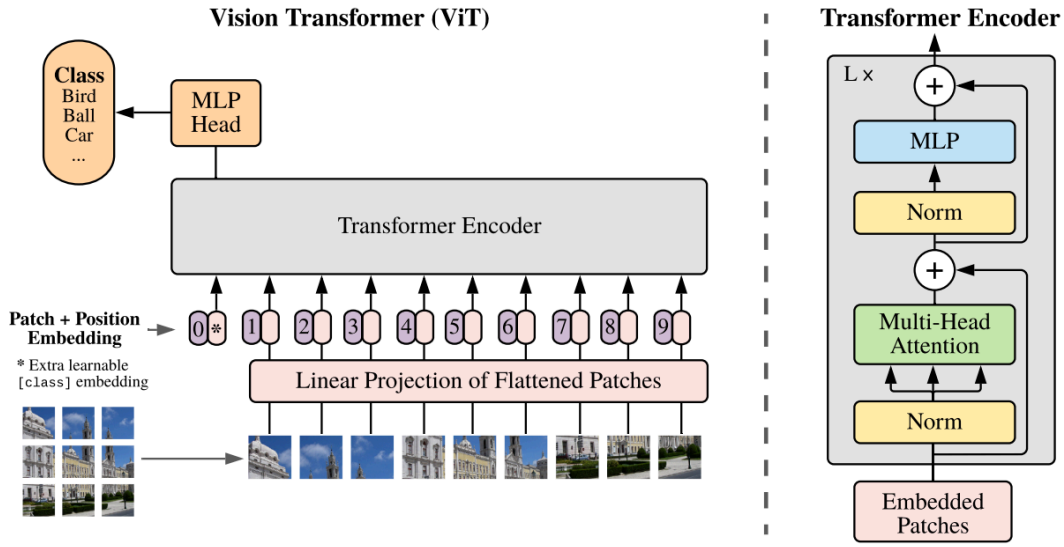
The purpose of this Design Document is to describe the architecture, components, and design decisions for the HALO system through system architecture diagrams and schematics.

1.2. Scope

This document covers the design and architecture of HALO, including its neural network model, data processing pipeline, and deployment platform.

2. System Design

2.1. Vision Transformer Model



* weights restricted to the set {0,1}

Image to Patches

- The $H \times W \times C$ image size is split into non-overlapping $P \times P$ patches. Each patch is flattened into a vector of length P^2C. This converts the 2-D image into a token sequence of length $N = \frac{HW}{P^2}$ that the transformer can read.

Patch Embedding

- Each flattened patch is mapped by a learned linear layer to a D-dimensional token. This standardizes patch features into a common space so subsequent layers can compare and combine tokens efficiently.

CLS Token

- A learned D-dimensional vector is prepended to the sequence. It is optimized to aggregate global information across the entire sequence and is used later by the classifier to produce the final prediction.

Position Embeddings

- A position code is added to each token (including [CLS]) to encode its location in the original image grid. This restores spatial order, which the transformer does not infer on its own.

Embedded Token Sequence

- After embedding and position addition, the model holds a $(N + 1) \times D$ sequence $[CLS, patch_1, .., patch_N]$. This is the standardized input to all encoder blocks.

Transformer Encoder Block

- Each block applies Normalization → Multi-Head Self-Attention → Residual Add → Normalization → MLP → Residual Add to Whole Sequence. Stacking L blocks progressively refines token representations.
- Normalization
  - LayerNorm/RMSNorm rescales features per token to stabilize optimization and maintain numerically well-conditioned activations across depth.
- Multi-Head Self-Attention
  - Query, Key, and Value projections are computed for all tokens. Attention weights are obtained from QK^T (scaled and softmaxed) and used to mix V. Multiple heads allow the model to capture diverse relations across tokens.
- Residual Connections
  - The block input is added to the outputs of attention and MLP sublayers. These shortcuts preserve information and improve gradient flow in deep stacks.
- MLP (Feed-Forward)
  - A two-layer per-token network with an activation function increases representational capacity by mixing channel information independently of sequence position.

Classifier Head

- The final [CLS] token is passed through a small linear/MLP head to produce class logits. This converts the aggregated representation into task-specific scores.

Output

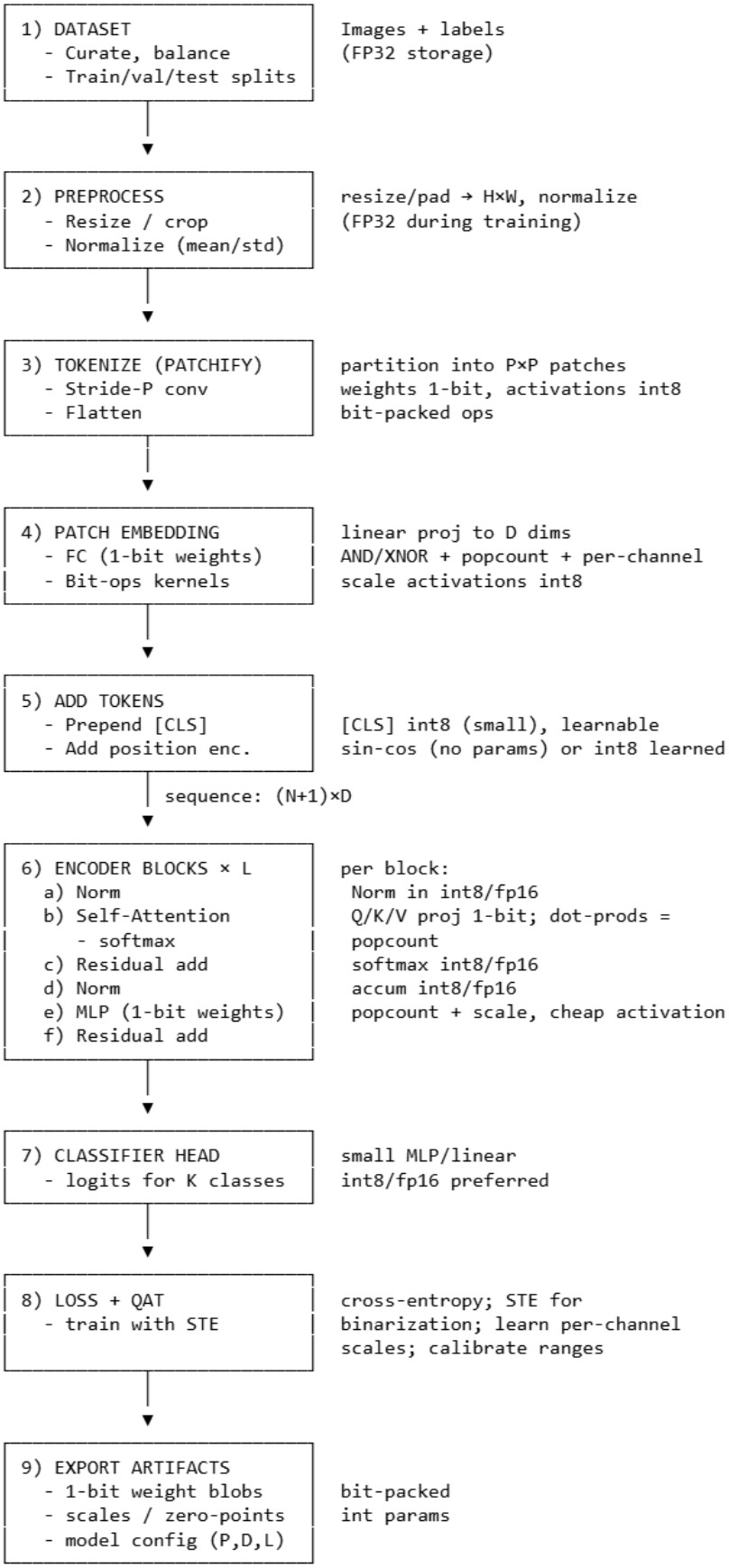
- Logits are converted to probabilities (e.g., softmax) and the top class is selected. This is the model's final decision.

1-Bit Notes

- Binary weights are used in the heavy linear layers (patch embedding, Q/K/V projections, and MLP), replacing multiplications with bit operations plus popcount and a learned scale.
- Normalization, softmax, and (optionally) embeddings/head remain in low integer or half precision for stability at negligible cost.
- Larger patch size reduces sequence length and attention cost; smaller depth/width and fewer heads reduce compute and memory.
- Bit-packing of weights/activations minimizes storage and bandwidth, improving throughput and energy efficiency on constrained hardware.

2.2. Processing Pipeline

2.2.1. Training/Quantization-Aware Training (QAT)



2.2.2. On-Device Inference