IIT Bombay

Department of Electrical Engineering

Dual Degree Project Report

Title: ***Application of Machine Learning in Agriculture***

Submitted by:

Mrudul Jambhulkar

Roll No: 21D070044

Dual Degree (B.Tech + M.Tech) in Electrical Engineering

Supervisor:

Prof. Rajbabu

Department of Electrical Engineering

Abstract

This project explores the application of advanced machine learning and image processing techniques including image segmentation to enhance precision agriculture using aerial and drone imagery leveraging models such as Vision Transformers, and the Segment Anything Model (SAM) . The study discusses the robustness of these models, particularly under image degradation, and explores solutions like RobustSAM and HQ-SAM for improved segmentation performance.

Contents

Chapter 1

Introduction

The rapid advancement of machine learning and image processing technologies has opened new avenues for their application in agriculture, an industry critical to global food security. This project focuses on leveraging these technologies to address key challenges in agricultural practices, specifically through the use of aerial and drone imagery. By applying machine learning techniques such as image segmentation, classification, and stitching, this work aims to enhance precision agriculture, enabling better crop monitoring, yield prediction, and resource management. The integration of advanced models like Transformers, Vision Transformers, and the Segment Anything Model (SAM) provides robust solutions for processing complex agricultural data. This report outlines the methodologies, models, and findings of applying these cutting-edge techniques to improve agricultural outcomes, with a focus on scalability and real-world applicability .

Chapter 2

Literature Review

2.1 Transformers for image recognition

The Transformer model proposes a novel architecture specifically for sequence-to-sequence transduction tasks. It has largely abandoned recurrence and convolution in favor of attention mechanisms. Its architecture is basically split into two parts: an encoder and a decoder. Both components rely heavily on self-attention mechanisms, combined with feed-forward networks in stacked layers.This model has an encoder-decoder architecture. Again, both architectures consist of N = 6 identical layers. Such a structure is good at processing sequential data, again leveraging the power of attention mechanisms and feed-forward networks [2] .

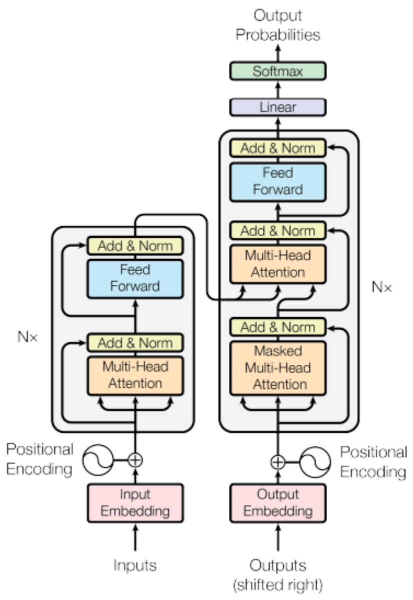

Figure 2.1: The Transformer - model architecture [2]

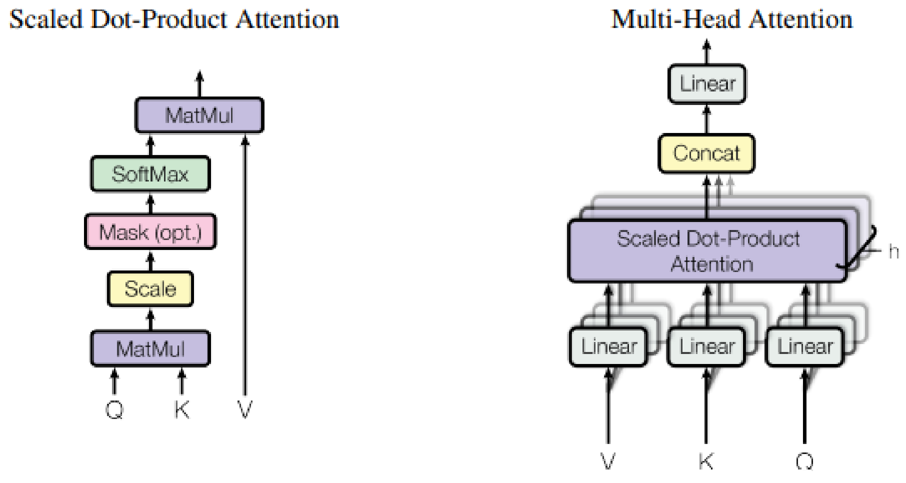

Figure 2.2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel [2]

Left half is the encoder part and the right half is the decoder part in the above figure.

Attention: The attention function could be defined as a mapping, the input to which would be query and set of key-value pairs to get the output. All query, keys, values, and output are vectors in nature. The output is computed to be the weighted sum of the values, where the weight for each value is provided by a compatibility function, which will establish the level of relationship of the query with the respective key.

Scaled Dot-Product Attention: A scaled dot-product attention mechanism is one of the key constituents of Transformer architecture. It processes three kinds of inputs: queries Q, keys K, and values V. The dot product of a query with each key gives an attention score that is scaled by the square root of the key dimension. Subsequently, a softmax operation is applied to result in attention weights:

$$\operatorname{Attention}(Q, K, V) = \operatorname{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2.1}$$

Multi-Head Attention:

$$\begin{aligned} \operatorname{MultiHead}(Q, K, V) &= \operatorname{Concat}(\text{ head }_1, \ldots, \text{ head }_{\mathrm{h}}) W^O \\ \text{where head } &= \operatorname{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2.2}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at each position, which helps the model learn diverse dependencies across the sequence. The Transformer strikes a balance between model complexity and computational cost by using h=8 heads.

Position-wise Feed-Forward Networks: Each block in both the encoder and decoder has a position-wise feed-forward network. That is essentially two linear transformations with a ReLU activation in between, and it does this entirely independently on each position:

$$\mathrm{FFN}(x) = \max\left(0, xW_1 + b_1\right) W_2 + b_2 \tag{2.3}$$

Another way of describing this is as two convolutions with kernel size 1.These layers enable the model to transform its representations produced by attention in such a way as to improve its expressibility power.

Embeddings and SoftMax: It does use learned embeddings as input, since it translates input tokens into possibly very dense vector representations. These are shared between the encoder and decoder's input layers and the pre-softmax linear transformation in the output layer, scaled by the square root of model dimension.

At the output, then, a linear transformation followed by a softmax maps the decoder's outputs to probabilities over the target vocabulary.

Positional Encoding: Because the Transformer does not apply recurrence or convolution, it adds positional encodings to the input embeddings, which signify the order of an element in a sequence. The authors utilize fixed sinusoidal functions to encode such positions.

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\text{model}}}\right) \end{aligned} \tag{2.4}$$

$$\text{where pos is the position and } i \text{ is the dimension.} \tag{2.5}$$

These encodings enable the model to identify both absolute and relative positional dependencies.

WHY SELF ATTENTION: The central mechanism in the Transformer architecture is self-attention, with very strong advantages over recurrent and convolutional networks.

2.1.1 Vision Transformer

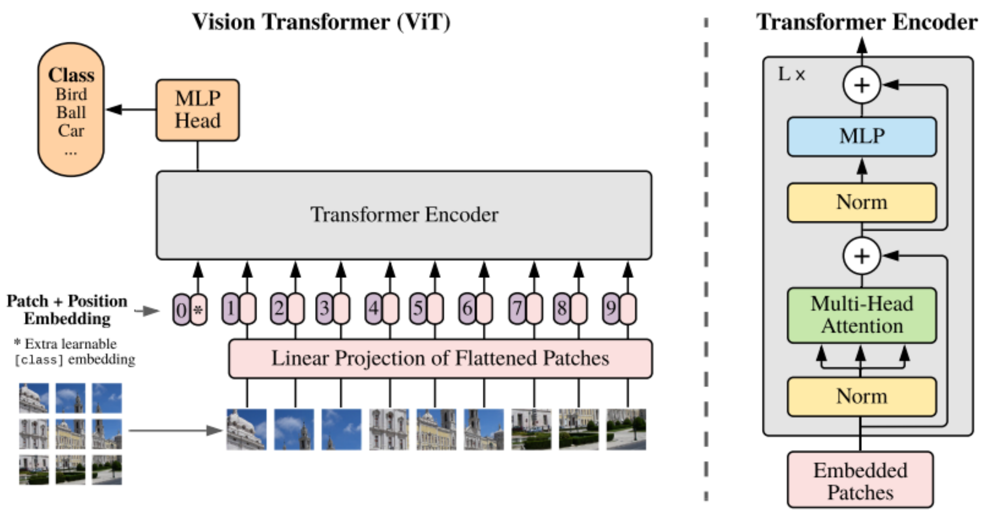

Figure 2.3: Vision Transformer [3]

If we naively apply self attention to images it would require each pixel to attend to every other pixel resulting in very high computational complexity . Several approximations have been explored to make self-attention mechanisms more efficient. Parmar et al. (2018) restricted self-attention to local neighborhoods around each query pixel, rather than applying it globally. These local multi-head dot-product self-attention blocks have been shown to effectively replace convolutions (Hu et al., 2019; Ramachandran et al., 2019; Zhao et al., 2020). Alternatively, Sparse Transformers (Child et al., 2019) introduce scalable approximations to global self-attention to enable its application to image data. Another approach scales attention by applying it in blocks of different sizes (Weissenborn et al., 2019), or even along individual axes in extreme cases (Ho et al., 2019; Wang et al., 2020a). Although these specialized attention mechanisms perform well on computer vision tasks, they often demand complex engineering for efficient deployment on hardware accelerators. Cordonnier et al. (2020) proposed a model which extracts patches of size 2 $\times$ 2 from the input image and applies full self-attention on top [3].
To handle 2D images, we reshape the image $x \in \mathbb{R}^{H\times W\times C}$ into a sequence of flattened 2D patches $x_p \in \mathbb{R}^{N\times(P^2\cdot C)}$, where (H, W) represents the original image resolution, C is the number of channels, and (P, P) is the size of each image patch. The number of patches is given by $N = \frac{HW}{P^2}$, which also becomes the input sequence length for the Transformer. Since the Transformer operates with a constant latent dimension D throughout its layers, each flattened patch is projected to D dimensions using a trainable linear layer . These

projected vectors are referred to as patch embeddings.
Just like BERT's `[class]` token, we prepend a learnable embedding x_{class} to the sequence of patch embeddings, resulting in $z_0^0 = x_{\text{class}}$. The final state of this token after passing through the Transformer encoder, denoted z_0^L, is used as the image representation y . During both pre-training and fine-tuning, a classification head is attached to z_0^L. This head is a multi-layer perceptron (MLP) with one hidden layer during pre-training, and a single linear layer during fine-tuning. Position embeddings are added to patch embeddings to retain positional information . The resulting sequence of embedding vectors act as input to the encoder.
[**The MLP contains two layers with a GELU non-linearity.**]

$$\begin{aligned}
&z_0 = [x_{\text{class}}; x_p^1 E; x_p^2 E; \cdots; x_p^N E] + E_{\text{pos}}, \\
&E \in \mathbb{R}^{(P^2 \cdot C) \times D}, \quad E_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}, \\
&z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \quad \ell = 1, \ldots, L, \\
&z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell, \quad \ell = 1, \ldots, L, \\
&y = \text{LN}(z_L^0)
\end{aligned}$$

Inductive bias : ViT has much less inductive bias (prior knowledge or constraints that guide the model in choosing one hypothesis over another when multiple hypotheses fit the training data equally well) as compared to CNNs . In CNNs local neighbourhood and translation equivariance is considered at each layer . In VIT , the 2D neighbourhood structure is utilized only in the beginning of model while dividing the image into patches and during fine tuning for adjusting position embeddings for images with different resolutions . Apart from these , position embedding don't carry information about spatial structure or 2D positions of patches .
Hybrid architecture : Instead of raw image patches , the input sequence to ViT can be given as feature maps from CNN. The patch embedding is then applied to patches extracted from CNN feature map .

Fine tuning and higher resolution: ViT is typically trained on large datasets and fine tuned to smaller downstream tasks .During fine-tuning, the pre-trained classification head is replaced with a zero-initialized feedforward layer of size D × K, where K is the number of target classes. Fine-tuning often benefits from using higher image resolutions than those used during pre-training. To do this, the patch size remains unchanged, resulting in longer input sequences. Although the Vision Transformer can process varying sequence lengths, the original positional embeddings may not align correctly. To address this, the pre-trained positional embeddings are resized using 2D interpolation based on their spatial positions. Importantly, this interpolation step and the patch extraction are the only parts where a structural prior about image geometry is explicitly added, introducing a minimal inductive bias [3].

2.2 Segment Anything Model (SAM)

This model is used for image segmentation problems using prompt engineering . The success of this model relies on task , model and data [1] .

- **Task:** The idea of prompt from NLP can be used for image segmentation . A prompt can be a set of foreground / background points, a rough box or mask, free-form text, or any information indicating what to segment in an image. The task is to develop a valid segmentation mask given the prompt . Here valid means that even if the prompt given is ambiguous (can refer to multiple objects) , it should return a reasonable mask for atleast on of those objects .
- **Pre-training:** The promptable segmentation task suggests a pre-training algorithm that simulates a sequence of prompts for each training sample and compares the model's mask predictions against the ground truth . Aim is to predict a valid mask even when the prompt is ambiguous ensuring that pre-trained model is effective even in cases that involve ambiguity , including automatic annotation as required by the data engine .
- **Zero-shot transfer:** The pre trained model should have the ability to respond to any promot at the time of inference . The model should be to correctly perform a task without having seen any explicit examples of that task during training.

2.2.1 Model Architecture

The SAM model has 3 components -

1. **Image encoder :** MAE pre-trained Vision Transformer (ViT) is used to process high resolution inputs . It runs once per image and can be applied before providing prompts to the model .
2. **Prompt encoder :** 2 sets of prompts are considered - sparse (points , boxes and text) and dense (masks) . Points and boxes are represented by positional embeddings summed with learned embeddings for each prompt type and free-form text with an off-the-shelf text encoder from CLIP . Dense prompts (masks) are embedded using convolutions and summed element-wise with the image embedding.

3. **Mask decoder :** The mask decoder maps the image embedding, prompt embeddings, and an output token to a mask . This block uses prompt self-attention and cross-attention in two directions (prompt-to-image embedding and vice-versa) to update all embeddings . After running the two blocks , the image embedding is upsampled and an MLP maps output token to linear classifier which computes the mask foreground probability at each image location .

If an ambiguous prompt is given , then the model will average multiple valid masks . 3 mask outputs are addressed for this . The masks are ranked based on their IoU score . To predict accuracy of mask prediction a linear combination of dice and focal loss is used [1] .

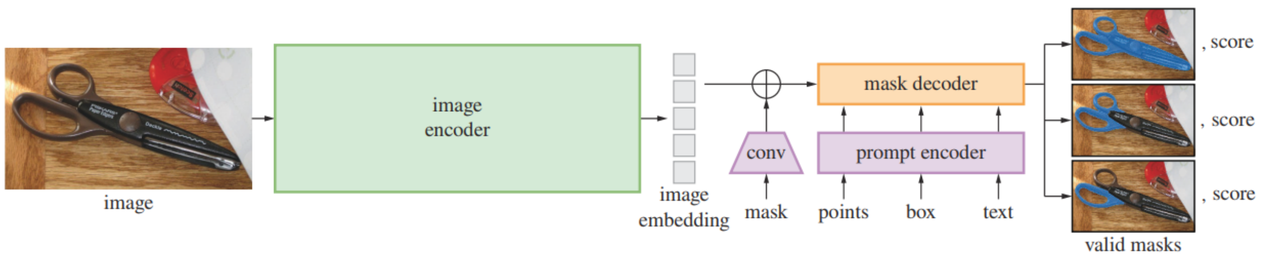

Figure 2.4: SAM overview [1]

2.3 RobustSAM

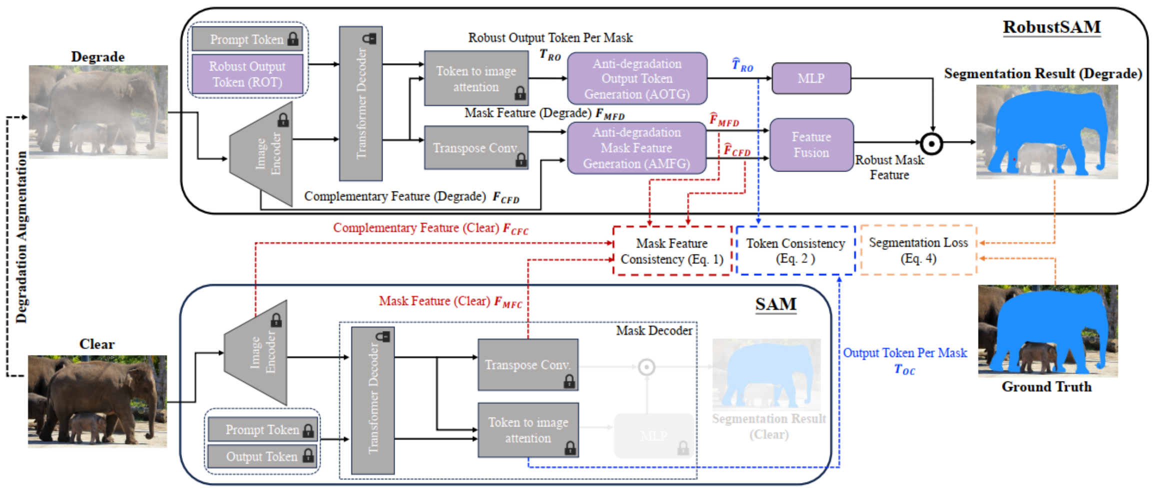
RobustSAM
Degrade
Prompt Token
Robust Output Token (ROT)
Image Encoder
Transformer Decoder
Robust Output Token Per Mask
T_{RO}
Token to image attention
Anti-degradation Output Token Generation (AOTG)
$\hat{T}_{RO}$
MLP
Segmentation Result (Degrade)
Mask Feature (Degrade) F_{MFD}
Transpose Conv.
Anti-degradation Mask Feature Generation (AMFG)
$\hat{F}_{MFD}$
$\hat{F}_{CFD}$
Feature Fusion
Robust Mask Feature
Complementary Feature (Degrade) F_{CFD}
Degradation Augmentation
Complementary Feature (Clear) F_{CFC}
Mask Feature Consistency (Eq. 1)
Token Consistency (Eq. 2)
Segmentation Loss (Eq. 4)
Mask Feature (Clear) F_{MFC}
SAM
Clear
Mask Decoder
Transpose Conv.
Token to image attention
MLP
Segmentation Result (Clear)
Prompt Token
Output Token
Output Token Per Mask
T_{OC}
Ground Truth

Figure 2.5: RobustSAM [4]

RobustSAM allows for zero shot learning even when there is image degradation . It enhances the robustness of model . RobustSAM contains AntiDegradation Output Token Generation (AOTG) and AntiDegradation Mask Feature Generation (AMFG) modules, to extract degradation-invariant information . 15 pairs of clear and degraded images are generated . Different losses are then applied to enforce consistency between clear and degraded features , as well as between the predicted segmentation and ground truth . RobustSAM works well with real-world images [4].

To train RobustSAM, we begin by applying various degradation augmentations to a clean input image. This degraded image is then fed into RobustSAM, where the Image Encoder extracts relevant features. A fine-tuned *Robust Output Token* (ROT), along with prompt tokens and extracted features, is passed through the original SAM layers to yield degraded mask features (FMFD) and output tokens per mask (TRO).

The **Anti-Degradation Output Token Generation (AOTG)** module transforms TRO into a degradation-resilient output token, denoted as $\hat{T}_{RO}$. Simultaneously, the **Anti-Degradation Mask Feature Generation (AMFG)** module refines early and final features from the encoder, removing degradation-related noise to generate $\hat{F}_{MFD}$ and $\hat{F}_{CFD}$.

A Feature Fusion block then combines these refined features into a robust mask representation to enhance segmentation quality.

In parallel, the original clean image is processed by the standard SAM to obtain clear complementary features (F^C_{CF}), mask features (F^C_{MF}), and output tokens (T^C_O). Consistency losses are applied between these clear outputs and their robust counterparts to ensure proper alignment. A segmentation loss is used to evaluate the degraded output against ground truth.

Training incorporates 15 types of degradation along with identity mapping, ensuring that clean images remain unaffected in performance.

During inference, only RobustSAM is utilized to produce segmentation masks [4].

2.3.1 Anti-Degradation Mask Feature Generation

Input features are first passed for instance normalization (IN) . It is done to standardize variations caused due to image degradation . This is important to mitigate the influence caused by image distortions , degradation etc . In parallel batch normalization (BN) is also done to account for the loss of detail occuring due to IN . The features from IN and BN are then merged . Attention mechanism is then used to weigh the importance of the merged feature maps to generate a feature set that contains information/advantages from both normalization techniques . To compensate for loss of information this feature set is merged with original input features along the channel dimension . Channel attention is then further integrated to refine the feature integration adaptively. A Fourier Degradation Suppression module further isolates degradation, focusing on maintaining structural integrity. These modules ensure the generation of robust, degradation-invariant features for precise image assessment and segmentation [4].

This module processes two degraded features: the complementary feature (F_{CFD}) and the mask feature (F_{MFD}). To ensure that the refined outputs align with their clean-image counterparts (F_{CFC} and F_{MFC}) produced by the original SAM, we apply the *Mask Feature Consistency Loss* ($\mathcal{L}_{\text{MFC}}$) :

$$\mathcal{L}_{\text{MFC}} = \left\| \hat{F}_{CFD} - F_{CFC} \right\|_2 + \left\| \hat{F}_{MFD} - F_{MFC} \right\|_2$$

2.3.2 Anti-Degradation Output Token generation

The Anti-Degradation Output Token Generation (AOTG) module is responsible for refining the Robust Output Token (TRO) for each mask to remove degradation-related information. Since TRO is designed to enhance classification boundary clarity rather than encode detailed textures, we observe that a lightweight module suffices to filter out degradation-sensitive content effectively . This module has multiple layers of IN followed by a single MLP layer [4]. . This enables the design to be computationally efficient and also allowing the model to learn robust mask features from degraded inputs. The refined token $\hat{T}_{RO}$ is compared to the output token T_{OC}, which is extracted by the original SAM model under clear input conditions. The comparison is performed using the Token Consistency Loss $\mathcal{L}_{TC}$:

$$\mathcal{L}_{TC} = \left\| \hat{T}_{RO} - T_{OC} \right\|_2$$

This loss enforces that the refined token remains consistent with the one extracted under clear image conditions [4].

2.3.3 Overall loss

The overall loss function integrates the Mask Feature Consistency Loss $\mathcal{L}_{\text{MFC}}$, Token Consistency Loss $\mathcal{L}_{\text{TC}}$, and the Segmentation Loss $\mathcal{L}_{\text{Seg}}$, forming a complete optimization objective. It is defined as:

$$\mathcal{L}_{\text{Overall}} = \mathcal{L}_{\text{MFC}} + \lambda_1 \mathcal{L}_{\text{TC}} + \lambda_2 \mathcal{L}_{\text{Seg}}$$

Here, $\mathcal{L}_{\text{Seg}}$ is a hybrid segmentation loss composed of Dice Loss and Focal Loss:

$$\mathcal{L}_{\text{Seg}} = \mathcal{L}_{\text{Dice}}(P, G) + \lambda_3 \mathcal{L}_{\text{Focal}}(P, G)$$

where P represents the predicted mask and G is the ground truth mask. The weights

λ_1, λ_2, and λ_3 are hyperparameters used to balance the contributions of each component. This compound loss encourages better segmentation performance and increases robustness to image degradations [4].

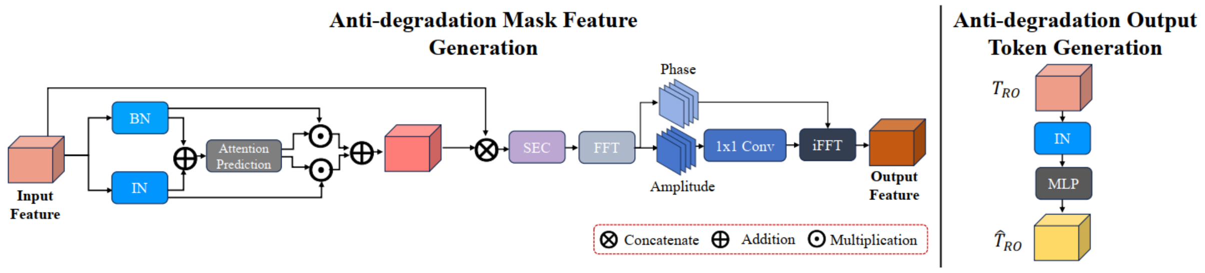

Figure 2.6: Overview of AMFG and AOTG [4]

2.4 High Quality SAM

HQ-SAM is used to upgrade SAM for high-quality zero-shot segmentation . It reuses the pre-trained model weights of SAM as much as possible but introduces two new key components - High-Quality Output Token and Global-local Feature Fusion .

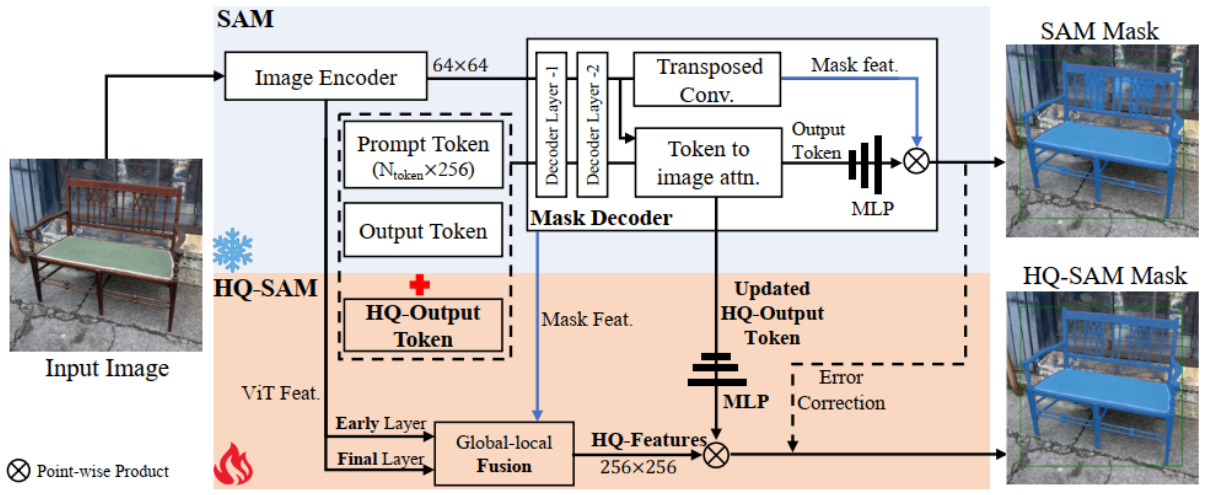

Figure 2.7: HQ-SAM [5]

2.4.1 High-Quality Output Token

Instead of directly using the mask inputs in SAM , HQ-Output token and a new mask prediction layer for high-quality mask prediction are introduced . A new learnable HQ-Output Token (size of 1×256) is concatenated with SAM's output tokens (size of 4×256) and prompt tokens (size of $N_{prompt} \times 256$) as the input to the SAM's mask decoder . In

each attention layer , HQ-Output Token first performs self-attention with other tokens and then conducts both token-to-image and the reverse image-to-token attention for updating the features . HQ-output token uses point-wise MLP shared by other tokens in each decoder layer . After passing throught the 2 decoder layers it has now global context information of image , the critical geometric/type information of prompt tokens as well as hidden mask information of the other output tokens . Finally, a three-layer MLP is used to generate dynamic convolutional kernels from the updated HQ-Output Token to perform spatial point-wise product with the fused HQ-feature for high-quality mask generation [5].
Instead of fine-tuning the entire SAM model , only the HQ-Output Token and its associated three-layer MLPs are trained for correcting the mask errors of SAM's output token . This improves the quality while introducing negligible extra parameters compared to the original SAM model . Also , HQ-SAM avoids overfitting [5] .

2.4.2 Global-local Fusion for High-quality Features

Accurate segmentation requires image features that include both rich global semantic context and local boundary details. To improve mask prediction quality, we enrich SAM's mask decoder output by fusing both high-level and low-level features, forming what we refer to as high-quality features (HQ-Features).

Rather than solely relying on SAM's decoder output, we extract and combine features from multiple stages of the SAM model: 1. The early layer **local** feature from SAM's ViT encoder, with a spatial resolution of 64×64 captures general boundary and edge information. 2. The final layer **global** feature from the encoder, also with a resolution of 64×64, representing the global semantic content. 3. The **mask decoder feature** from SAM, with a shape of 256×256, containing strong shape-specific mask information.

The local and global encoder features are first upsampled to 256×256 using transposed convolutions. These three feature maps are then combined element-wise following a simple convolutional refinement. This fusion strategy effectively preserves detail while being lightweight in terms of memory and computation.

Chapter 3

Conclusion and Future Work

This project demonstrates the potential of machine learning and image processing techniques, particularly Transformers, Vision Transformers, and the Segment Anything Model, in addressing critical challenges in agriculture. Through image segmentation, classification, and stitching of aerial and drone imagery, the proposed methods have shown promise in enhancing precision agriculture, improving crop monitoring, and optimizing resource utilization. The findings highlight the effectiveness of these models in handling complex agricultural data, even under challenging conditions such as image degradation. However, limitations such as computational complexity and the need for large-scale, diverse datasets remain. Future work will focus on optimizing these models or exploring better machine learning techniques for real-time applications specific to aerial / drone images and addressing the challenges faced while solving these image processing problems .

Bibliography

[1] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al., "Segment anything," Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4015–4026, 2023

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[4] W.-T. Chen, Y.-J. Vong, S.-Y. Kuo, S. Ma, and J. Wang, "Robustsam: Segment anything robustly on degraded images," 2024.

[5]] L. Ke, M. Ye, M. Danelljan, Y. Liu, Y.-W. Tai, C.-K. Tang, and F. Yu, "Segment anything in high quality," in NeurIPS, 2023.