



Alberto Colella

Efficient Computer Vision
Models for Silkworm Feeding Prediction
and Habitat Analysis

Università degli Studi di Roma **La Sapienza**
Computer Vision 2024-2025



Outline

Introduction
Related Works
Proposed Method
Dataset and Metrics
Implementation details
Experimental results
Conclusions and Future Works

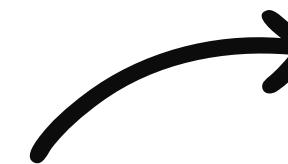


Problem statement



Context:

Silkworm rearing is traditionally labor-intensive and prone to human error



Objective:

Automate the monitoring process of silkworms' rearing



Tasks:

- Supervised Binary Classification
- Unsupervised Semantic Segmentation

Feeding or not feeding ?

where is the worms, the leaves and the background?



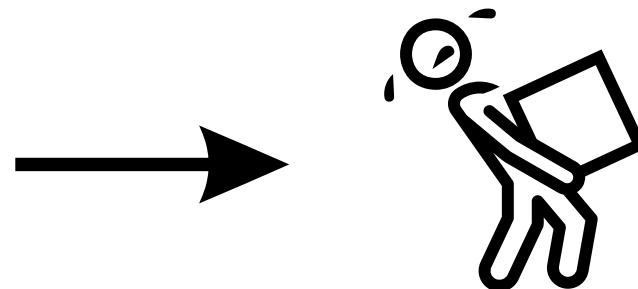
Constraints: Design lightweight neural networks for limited hardwares

Deep Learning for Vision



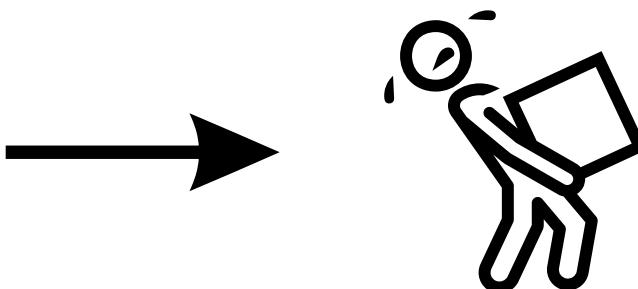
CNN:

- ✓ Efficient extraction of local features, they use data's prior biases
- ✗ Necessity of depth to have a wide receptive field



ViT:

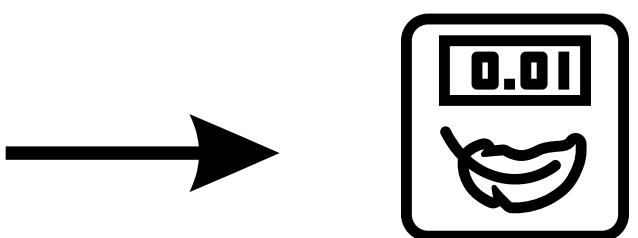
- ✓ Efficient extraction of global features by using self attention
- ✗ No prior biases, they need many parameters



Hybrid Architectures:



Local context + Global context



Hybrid architecture



Hybrid Architectures merge the advantages of CNNs and Transformers



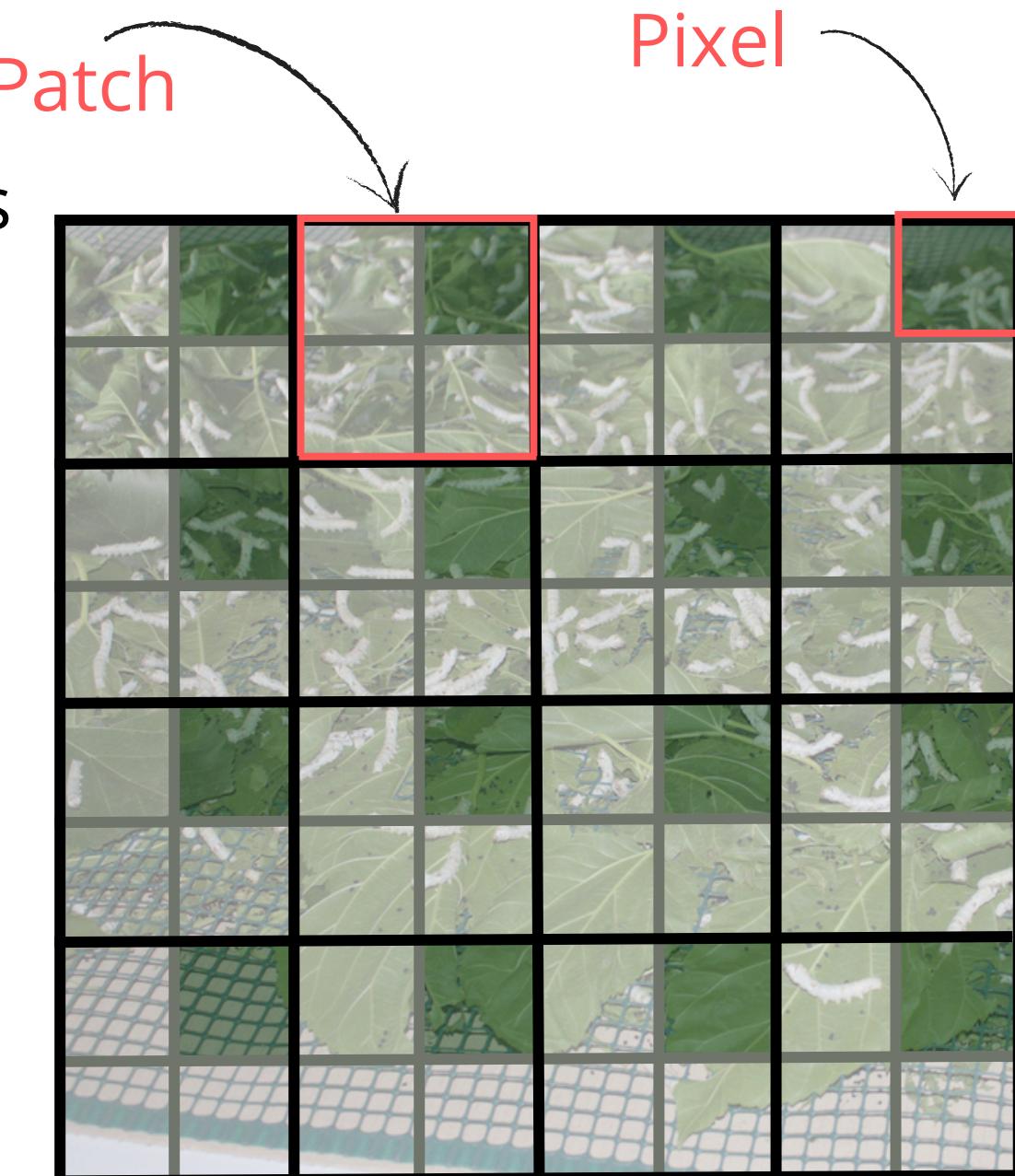
2DConv 3x3: every pixel sees his 8 nearest neighbours



Self Attention: every patch sees the others



Result: each pixel sees both the local and the global context



Unsupervised Semantic Segmentation



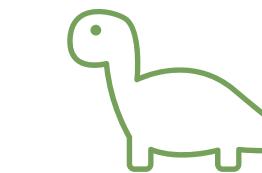
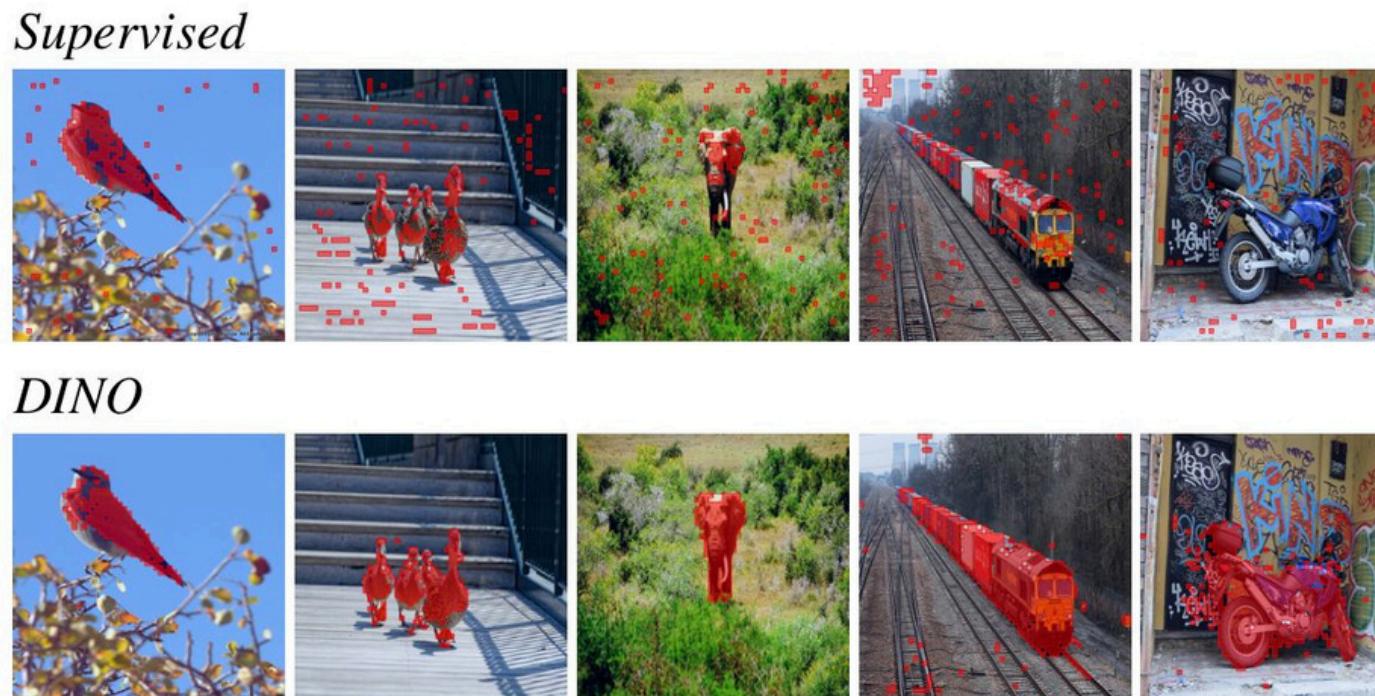
CNNs/Transformers used as feature **extractors**, followed by **clustering** on similar pixels.



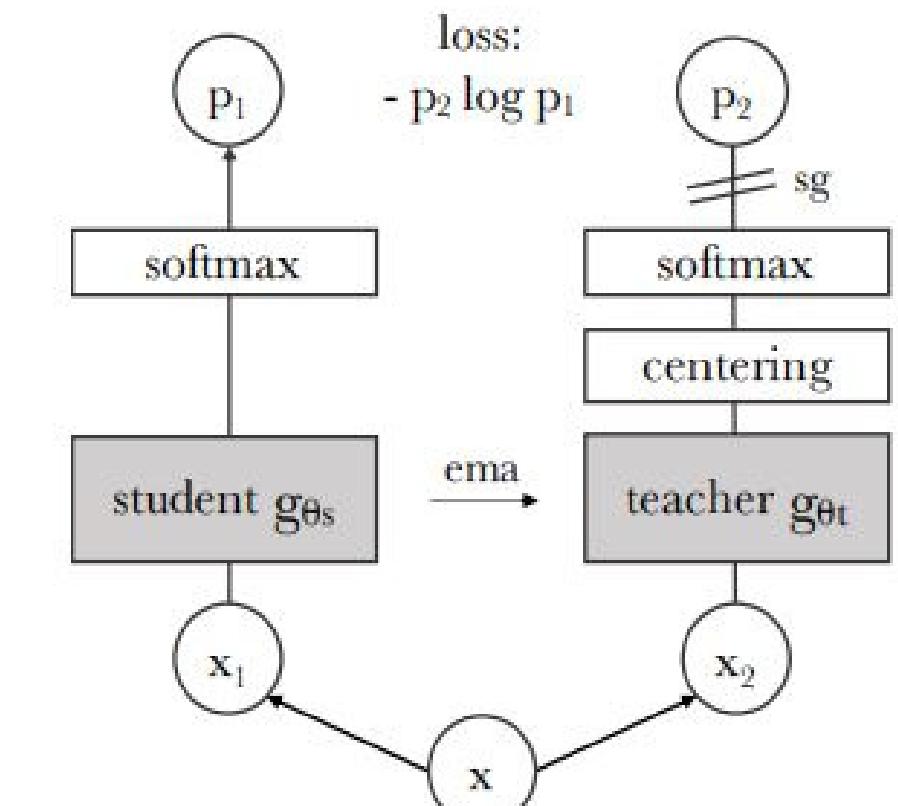
It's not clear how **supervised** models learn features



Self Supervised models highlight **semantic structures** such as edges and objects.

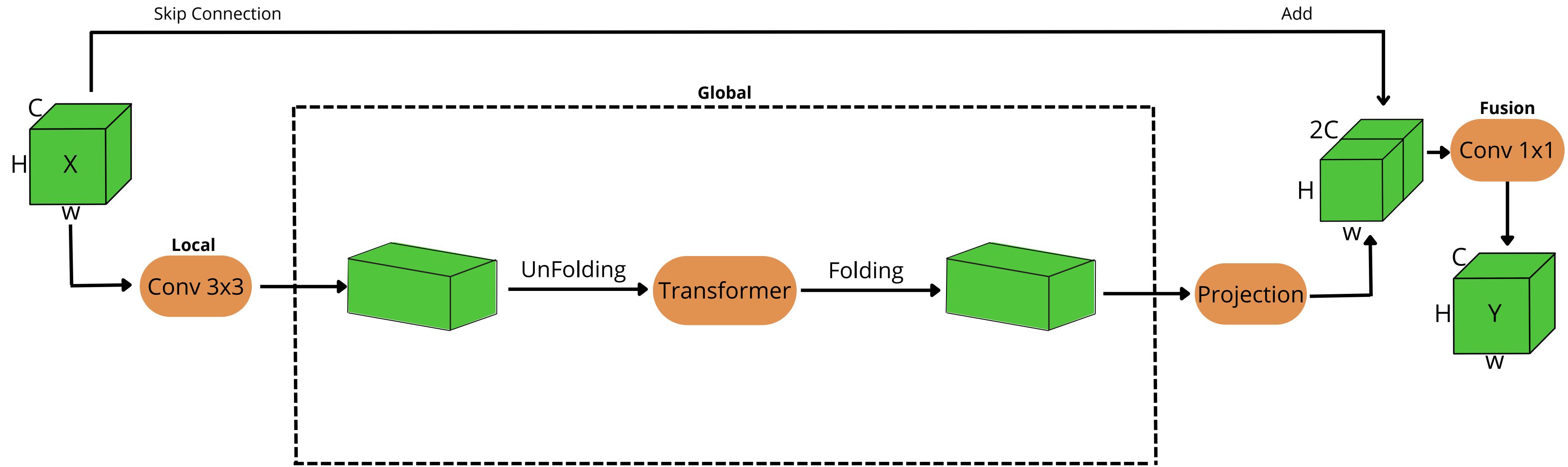


Dino self distillation
unsupervised :
single pair of views

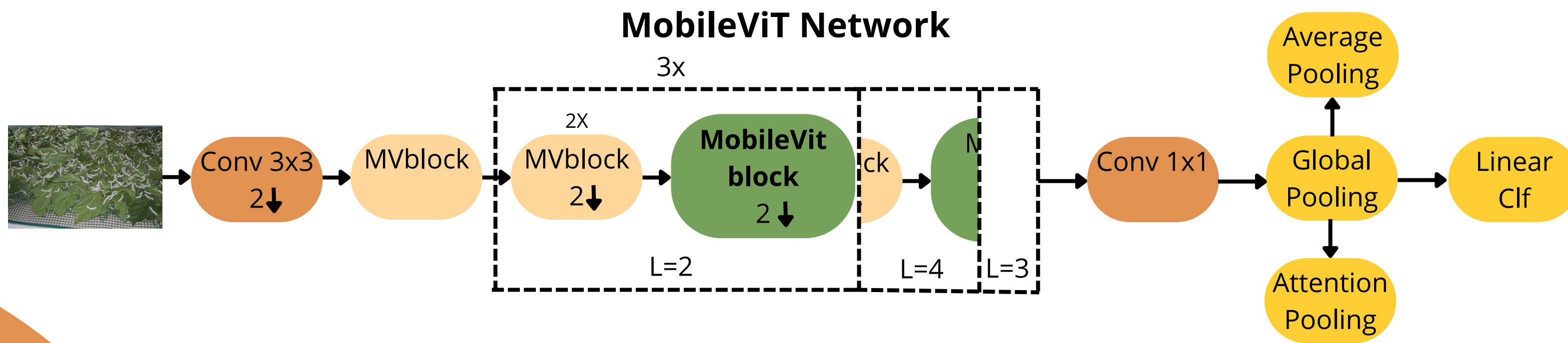


Proposed Method

MobileViT Block

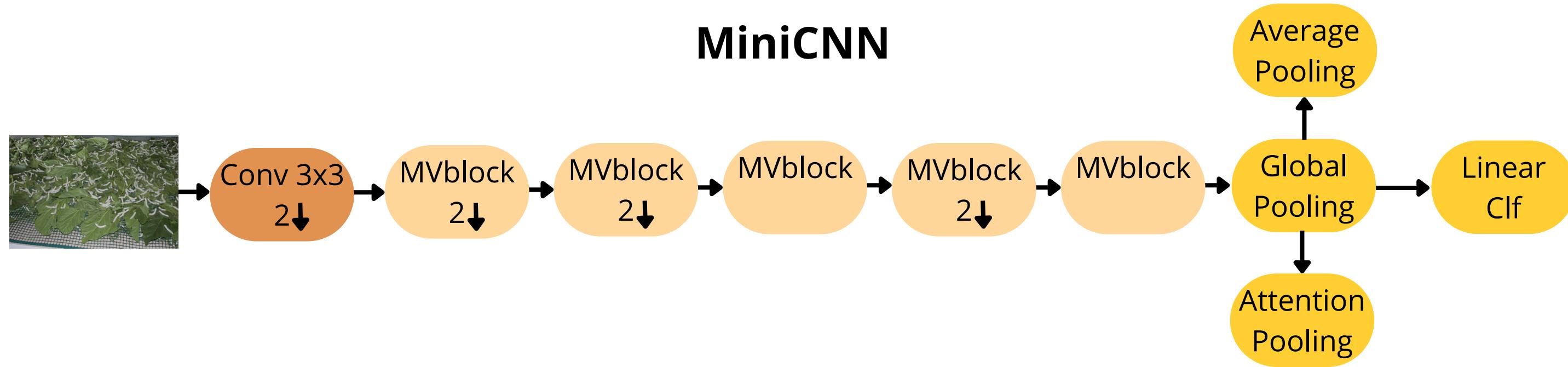


MobileViT Network



Another Network

Let's also consider a network that doesn't use the Transformer Architecture

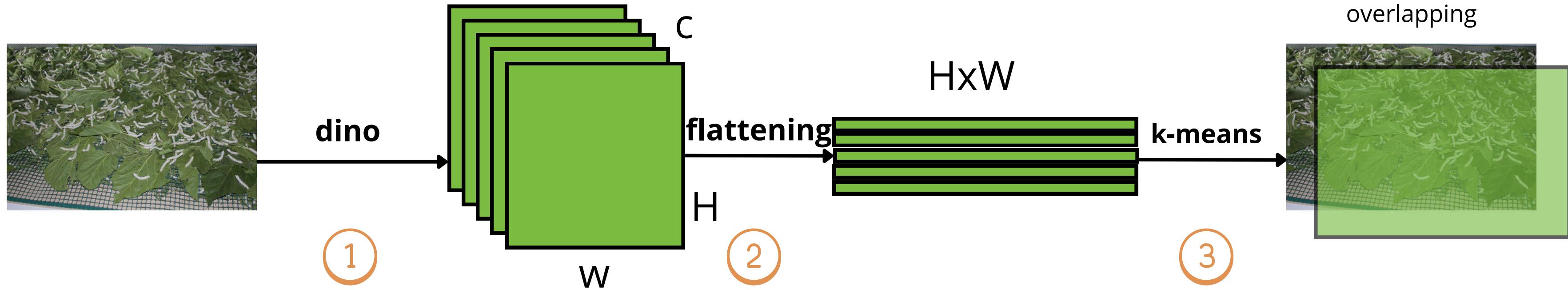


Number of parameters for various configuration of models used in the project

| Model | N_parameters |
|---------------|--------------|
| MiniCNN | 948.130 |
| MobileViT-xxs | 1.040.898 |
| MobileViT-xs | 2.068.858 |
| MobileViT-s | 5.327.074 |



DINO as feature extractor



- ① **Extracting** the feature from an intermediate layer of pretrained DINO
- ② **Flattening** the spatial feature, prepare of k-means
- ③
 - **Clustering** the similar feature with k-means $n_classes=3$
 - **Upsampling** the clusters on the image
 - **Overlapping** with original image

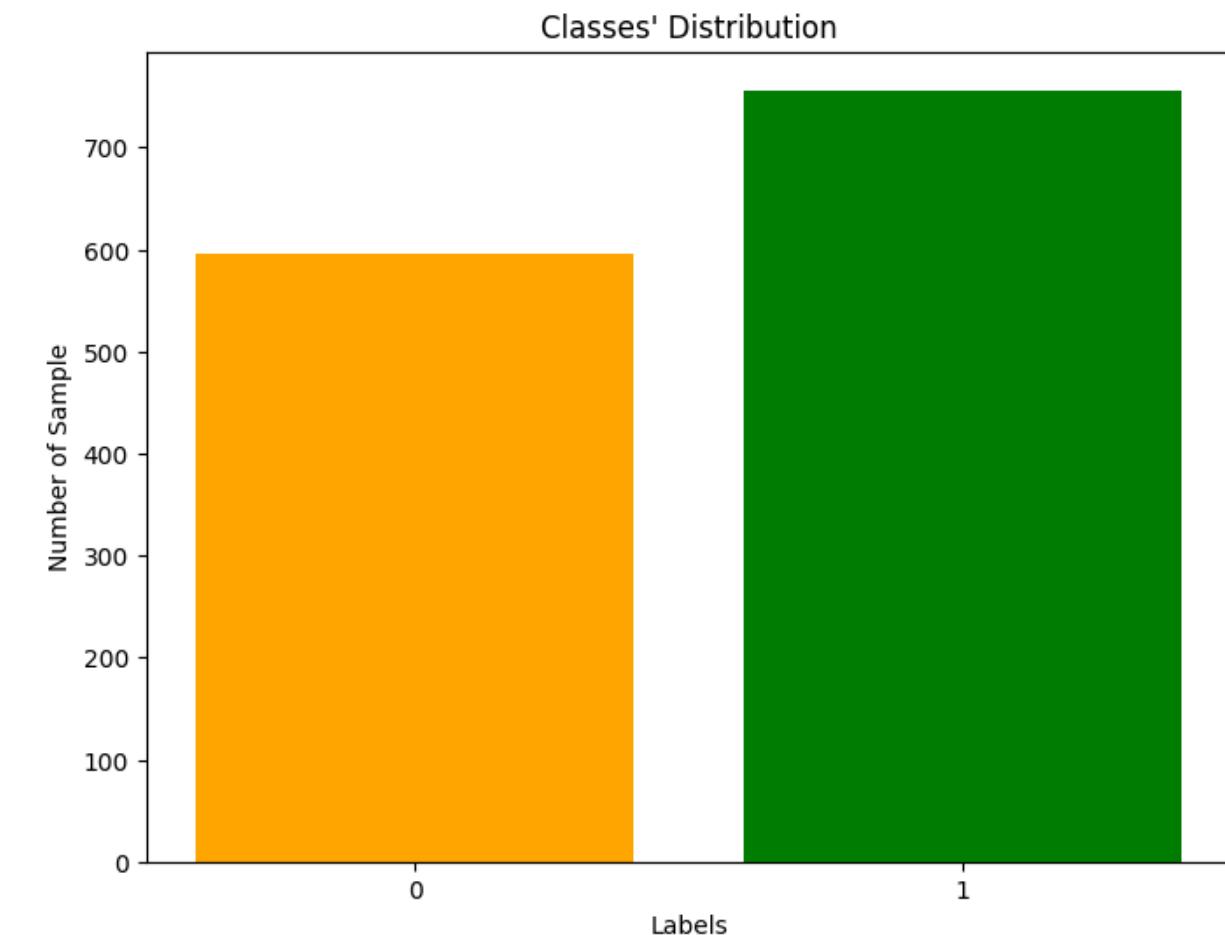
Dataset

💡 Let's have a look at the dataset to understand which strategies to execute



1351 Labeled Image

- Size: (1559,1038)
- 2 classes: feed/no-feed
- no-feeding:600 / feeding:751



no-feeding



feeding



Metrics

Classification

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Average infer time per image [ms]

Semantic Segmentation

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$\text{silhouette-score} = \frac{1}{N} \sum_{i=1}^N s(i)$$

Sperimental setup

Classification problem

Augmentation strategies:

- Standard:**
- RandomHorizontalFlip (50%)
 - RandomRotation (10°)
 - ColorJitter
 - Normalize

-
- Crop:**
- RandomResizedCrop (60%-100%)
 - + standard

Global Pooling strategies:

Attn pooling: Weighted sum of the spatial features

Avg pooling: AdaptiveAvaragePooling



| Hyperparameters | N_epochs | Batch_size | Learning rate |
|-----------------|----------|------------|---------------|
| Value | 10 | 32 | 0.0001 |

Semantic segmentation Problem:

Dino+
k_means

| | | |
|-----------|-------|-------------|
| BackBone | Layer | Input Size |
| Res-Net50 | Third | (1559,1038) |

Average Performance of Lightweight Models

Across Augmentation and Pooling Strategiess

| Augmentation | Pooling | F1_mean | F1_std | InferTime [ms] |
|--------------|---------------|---------|--------|----------------|
| standard_512 | AdaAvgPooling | 0,9534 | 0,0021 | 117,34 |
| crop_512 | Attn Pooling | 0,9506 | 0,0104 | 119,97 |
| crop_512 | AdaAvgPooling | 0,9489 | 0,0069 | 118,62 |
| standard_512 | Attn Pooling | 0,9433 | 0,0109 | 118,97 |
| standard_256 | AdaAvgPooling | 0,9431 | 0,012 | 26,05 |
| crop_256 | Attn Pooling | 0,942 | 0,0134 | 26,61 |
| standard_256 | Attn Pooling | 0,9383 | 0,0172 | 26,29 |
| crop_128 | Attn Pooling | 0,9376 | 0,0129 | 4,84 |
| standard_128 | AdaAvgPooling | 0,9368 | 0,007 | 5,365 |
| crop_256 | AdaAvgPooling | 0,9335 | 0,0071 | 26,26 |
| standard_128 | Attn Pooling | 0,9322 | 0,0112 | 4,745 |
| crop_128 | AdaAvgPooling | 0,9258 | 0,0135 | 4,75 |

💡 Observations:

- Larger input sizes consistently lead to higher performances
- Attn pooling shows higher variance compared to average pooling → less stable
- Crop augmentation combined with attention pooling yields a synergistic effect

The best performance for input size: 128

| Model | Augmentation | Pooling | F1_score | infer_time[ms] |
|-----------------|-----------------|---------------------|---------------|----------------|
| MobileViT-S | standard_128 | AdaAvgPooling | 0,9397 | 8,58 |
| MobileViT-S | crop_128 | AdaAvgPooling | 0,9238 | 7,3 |
| MobileViT-S | standard_128 | Attn Pooling | 0,9386 | 7,43 |
| MobileViT-S | crop_128 | Attn Pooling | 0,9447 | 7,54 |
| MobileViT-XS | standard_128 | AdaAvgPooling | 0,9292 | 6,65 |
| MobileViT-XS | crop_128 | AdaAvgPooling | 0,914 | 5,17 |
| MobileViT-XS | standard_128 | Attn Pooling | 0,9156 | 5,11 |
| MobileViT-XS | crop_128 | Attn Pooling | 0,9204 | 5,43 |
| MobileViT-XXS | standard_128 | AdaAvgPooling | 0,9333 | 1,92 |
| MobileViT-XXS | crop_128 | AdaAvgPooling | 0,9205 | 1,8 |
| MobileViT-XXS | standard_128 | Attn Pooling | 0,9362 | 1,7 |
| MobileViT-XXS | crop_128 | Attn Pooling | 0,9362 | 1,69 |
| MINI_CNN | standard_128 | AdaAvgPooling | 0,9451 | 4,31 |
| MINI_CNN | crop_128 | AdaAvgPooling | 0,9451 | 4,76 |
| MINI_CNN | standard_128 | Attn Pooling | 0,9386 | 4,74 |
| MINI_CNN | crop_128 | Attn Pooling | 0,9492 | 4,7 |



The best performance for input size: 256

| Model | Augmentation | Pooling | F1_score | infer_time[ms] |
|-----------------|-----------------|---------------------|---------------|----------------|
| MobileViT-S | standard_256 | AdaAvgPooling | 0,9573 | 36,51 |
| MobileViT-S | crop_256 | AdaAvgPooling | 0,9304 | 37,08 |
| MobileViT-S | standard_256 | Attn Pooling | 0,9286 | 37,79 |
| MobileViT-S | crop_256 | Attn Pooling | 0,9381 | 38,25 |
| MobileViT-XS | standard_256 | AdaAvgPooling | 0,9386 | 28,61 |
| MobileViT-XS | crop_256 | AdaAvgPooling | 0,9298 | 29,47 |
| MobileViT-XS | standard_256 | Attn Pooling | 0,9565 | 28,63 |
| MobileViT-XS | crop_256 | Attn Pooling | 0,9369 | 30,13 |
| MobileViT-XXS | standard_256 | AdaAvgPooling | 0,9292 | 11,23 |
| MobileViT-XXS | crop_256 | AdaAvgPooling | 0,9442 | 11,02 |
| MobileViT-XXS | standard_256 | Attn Pooling | 0,9196 | 11,49 |
| MobileViT-XXS | crop_256 | Attn Pooling | 0,9316 | 11,15 |
| MINI_CNN | standard_256 | AdaAvgPooling | 0,9474 | 27,88 |
| MINI_CNN | crop_256 | AdaAvgPooling | 0,9298 | 27,5 |
| MINI_CNN | standard_256 | Attn Pooling | 0,9487 | 27,26 |
| MINI_CNN | crop_256 | Attn Pooling | 0,9617 | 26,94 |



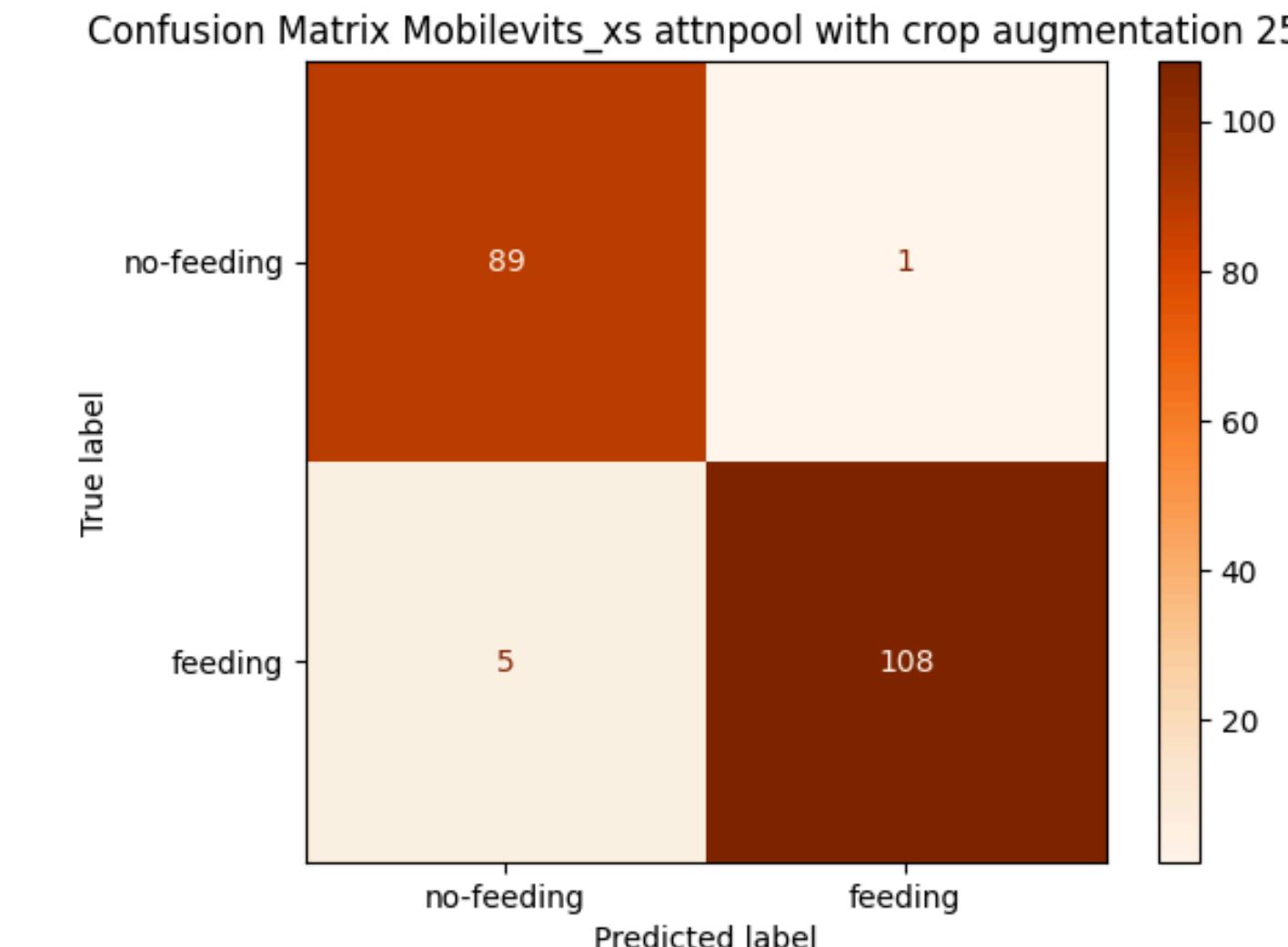
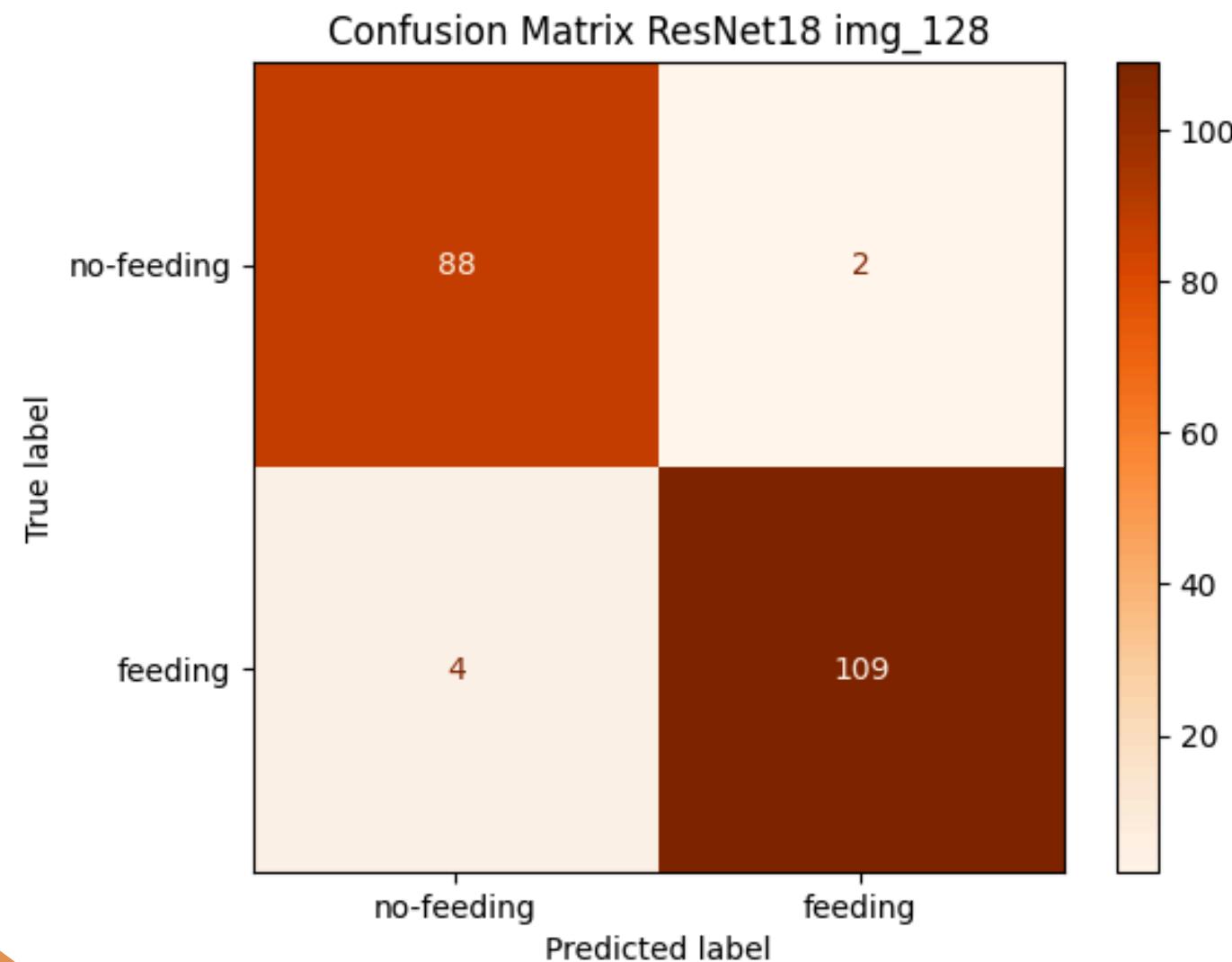
The best performance for input size: 512



| Model | Augmentation | Pooling | F1_score | infer_time[ms] |
|--------------------|-----------------|---------------------|---------------|----------------|
| MobileViT-S | standard_512 | AdaAvgPooling | 0,9565 | 164,01 |
| MobileViT-S | crop_512 | AdaAvgPooling | 0,9397 | 166,38 |
| MobileViT-S | standard_512 | Attn Pooling | 0,9524 | 165,28 |
| MobileViT-S | crop_512 | Attn Pooling | 0,9652 | 172,57 |
| MobileViT-XS | standard_512 | AdaAvgPooling | 0,9524 | 131,8 |
| MobileViT-XS | crop_512 | AdaAvgPooling | 0,952 | 134,01 |
| MobileViT-XS | standard_512 | Attn Pooling | 0,9279 | 136,07 |
| MobileViT-XS | crop_512 | Attn Pooling | 0,9407 | 137,4 |
| MobileViT-XXS | standard_512 | AdaAvgPooling | 0,9524 | 60,28 |
| MobileViT-XXS | crop_512 | AdaAvgPooling | 0,9558 | 58,66 |
| MobileViT-XXS | standard_512 | Attn Pooling | 0,9437 | 59,47 |
| MobileViT-XXS | crop_512 | Attn Pooling | 0,9487 | 56,93 |
| MINI_CNN | standard_512 | AdaAvgPooling | 0,9524 | 113,27 |
| MINI_CNN | crop_512 | AdaAvgPooling | 0,9483 | 115,44 |
| MINI_CNN | standard_512 | Attn Pooling | 0,9492 | 115,06 |
| MINI_CNN | crop_512 | Attn Pooling | 0,9478 | 112,98 |

Lightweight Models vs Heavy Models

| Model | Pooling | Augmentation | N_Parameters | Accuracy | F1-score | avg_infer_time[ms] |
|---------------|---------------|--------------|--------------|---------------|----------|--------------------|
| ResNet_18 | AdaAvgPooling | standard_128 | 11.169.858 | 0,9704 | 0,9732 | 84,72 |
| MobileViT-xxs | Attn Pooling | crop_256 | 1.040.898 | 0,9655 | 0,9692 | 11,03 |
| MobileViT-xs | Attn Pooling | crop_256 | 2.068.858 | 0,9704 | 0,973 | 29,19 |
| MobileViT-S | Attn Pooling | crop_256 | 5.327.074 | 0,9652 | 0,9683 | 32,87 |



Future works



We want to minimize the false negatives for **welfare of silkworms**

Predict label —→ no-feeding

True label —→ feeding



How to do this ?

Hyperparameter tuning
to maximize the **recall**

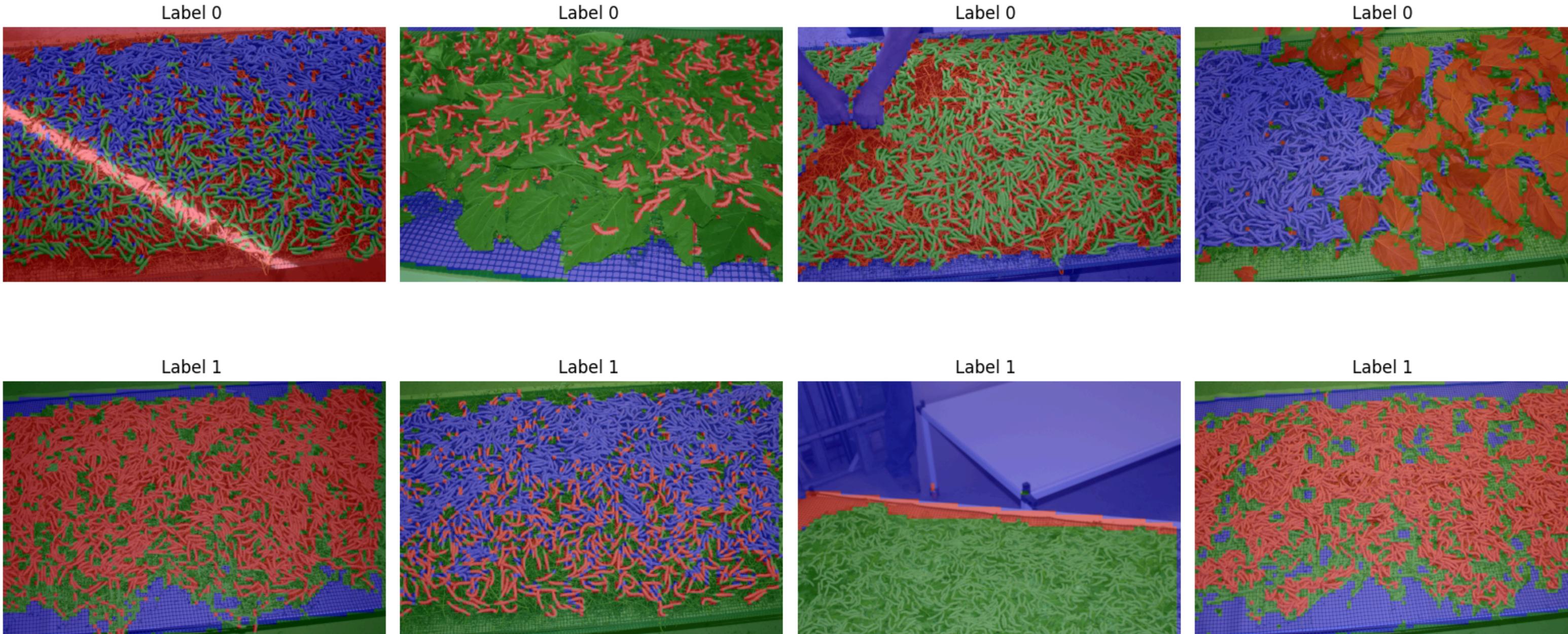


$$\text{Recall} = \frac{TP}{TP + FN}$$



Results of semantic segmentation

Model output of some **representative examples**:



- 📌 Noisy backgrounds introduce segmentation confusion
- 📌 When a semantic class is missing from the image, the model overestimate the number of clusters

Future works

- 📌 Automitize the choice of n_clusters that **maximize** the **silhouette score**

👁️ missing the leaves case

n_clusters=2 ?

DINO(ResNet50)_layer3 2clusters Sil_Score: 0.1667



Sil_score=0.17

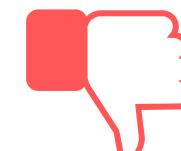


n_clusters=3 ?

DINO(ResNet50)_layer3 3clusters Sil_Score: 0.0838



Sil_score=0.08



📌 Let's test the method to ensure that it doesn't underestimate clusters when classes are well separated



All the classes are present case

n_clusters=2 ?

DINO(ResNet50)_layer3 2clusters Sil_Score: 0.0425



Sil_score=0.04



n_clusters=3 ?

DINO(ResNet50)_layer3 3clusters Sil_Score: 0.0753



Sil_score=0.07



Bibliography

- **MobileNetV2**

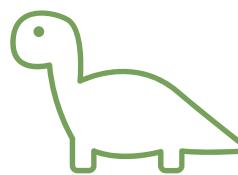
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR 2018.
<https://arxiv.org/abs/1801.04381>

- **MobileViT**

Mehta, S., & Rastegari, M. (2021). MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer
<https://arxiv.org/abs/2110.02178>

- **DINO**

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2021). Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021.
<https://arxiv.org/abs/2104.14294>



Thanks for your
attention

Alberto Colella

