2025.05

Fruits Classification

Contributor: Yichen Yang, Cecilia Wu, Jiawei Xiong, Michael Peng, Wen Yao

Agenda

1. Business Problem
2. Dataset Intro and EDA
3. Model
    a. CNN(baseline)
    b. Advanced CNN Models
    c. EfficientNet
    d. Transformer-Based Models
4. Web Application Display

Business problem

Problem

- Grocery stores face frequent revenue loss and customer delays due to incorrect labeling or pricing of fresh produce at self-checkout kiosks

Motivation

- We aim to improve accuracy, speed, and user experience by leveraging visual recognition to classify the right fruit and thus streamlining the self-checkout process

Goal

- Develop several high-accuracy machine learning models to automatically classify fruits and vegetables using images, and build a functional prototype web application

Analysis Scope

1

Expected Outcome:

- Train image classifiers to detect produce categories accurately
- Compare model performance to find the most suitable one for real-time use
- Build a deployable prototype to simulate self-checkout classification with real camera input

2

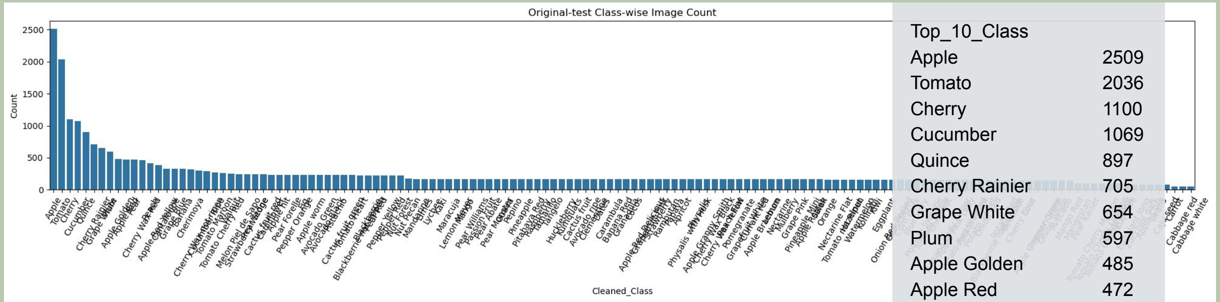
Models Explored:

- Baseline: CNN with 2 convolutional layers
- Improved Models:
    - ResNet-50
    - EfficientNet-B0
    - MobileNet-V3
    - ViT
    - Swin

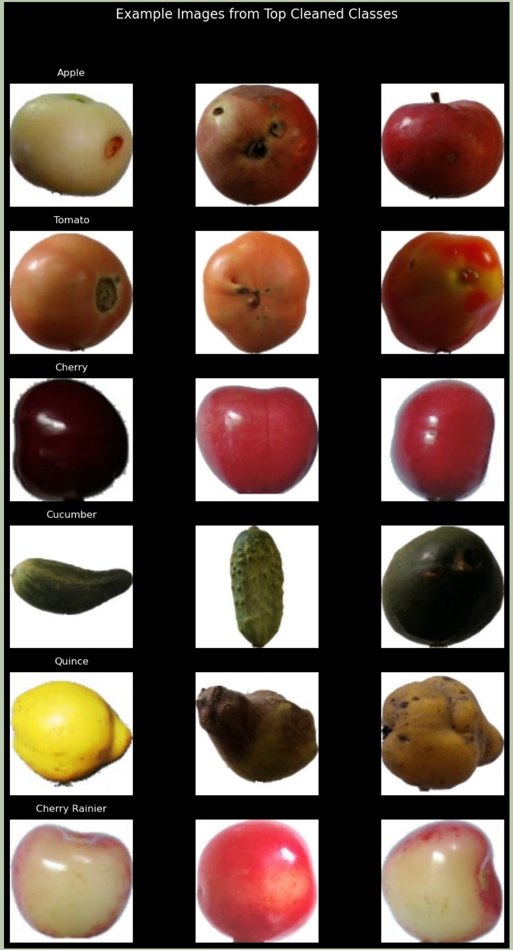
Dataset Preprocessing

- In original dataset, we get 142 categories with 102790 pictures in training dataset and 34314 in test dataset. All of them have shape of 100x100.

Top_10_Class	
Apple	7571
Tomato	6129
Cherry	3321
Cucumber	3226
Quince	2699
Cherry Rainier	2120
Grape White	1943
Plum	1767
Apple Golden	1453
Apple Red	1413

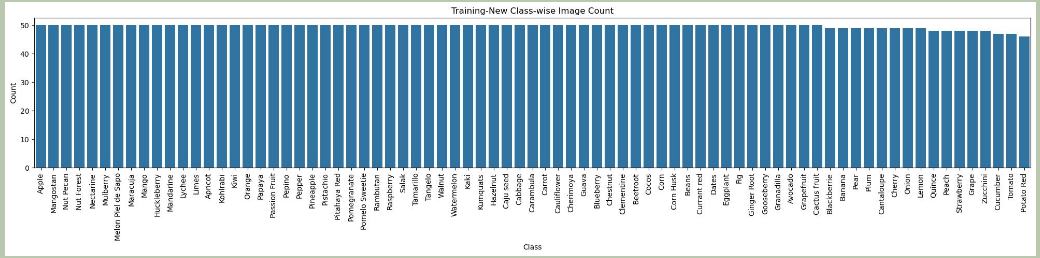


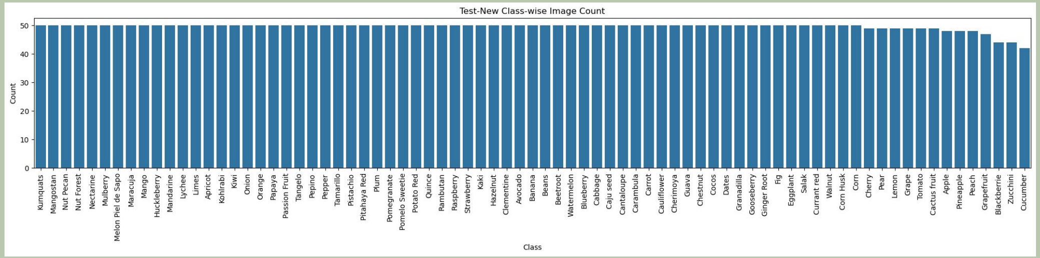
Top_10_Class	
Apple	2509
Tomato	2036
Cherry	1100
Cucumber	1069
Quince	897
Cherry Rainier	705
Grape White	654
Plum	597
Apple Golden	485
Apple Red	472

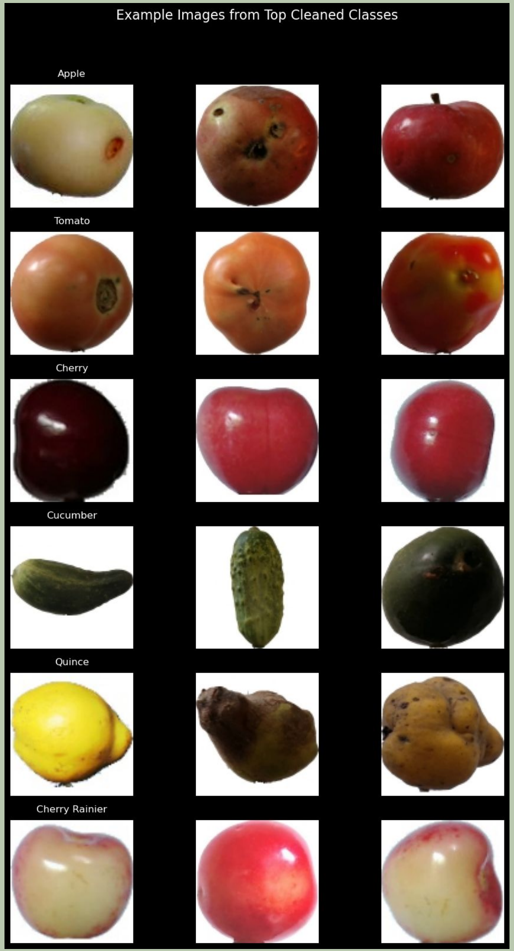


Dataset Preprocessing

- We noticed there are some similar categories, like ‘apple’ and ‘apple red’ in original data, so we extracted out the fruits name and treated them both as ‘apple’. Also, to keep the balance of each category, we randomly picked 50 pictures from each category (for the category with less than 50 pictures, we just use them all). We got 77 classes with 3822 pictures in training dataset and 3815 pictures in testing dataset.







Baseline Model : 2-Layer CNN

Architecture

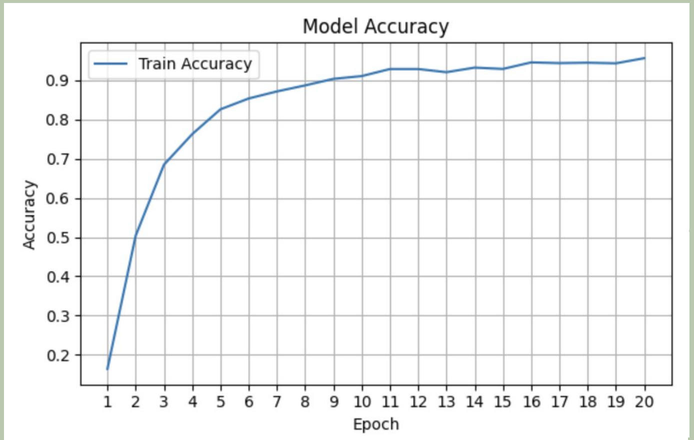
- A lightweight Convolutional Neural Network (CNN)
- 2 convolutional layers with ReLU activation:
  Conv2D(32, 3x3) → Conv2D(64, 3x3)
- MaxPooling2D layers after each convolution to reduce spatial dimensions
- A Flatten layer followed by:
  a. Dense(128) with ReLU
  b. Dropout(0.5) to reduce overfitting
  c. Output layer with softmax activation

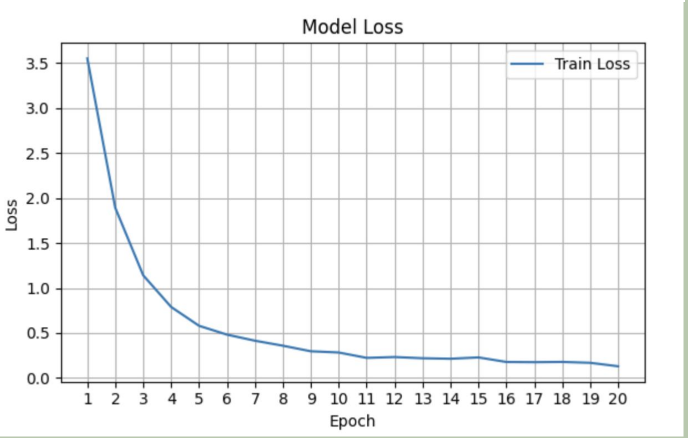
Training Setup - Use best hyperparameters

- Loss Function: categorical_crossentropy
- Optimizer: Adam
- Epochs: 20

Results

- **Final Training Accuracy**: 95.74%
- **Test Accuracy:** 87.87%





ResNet-50 Model

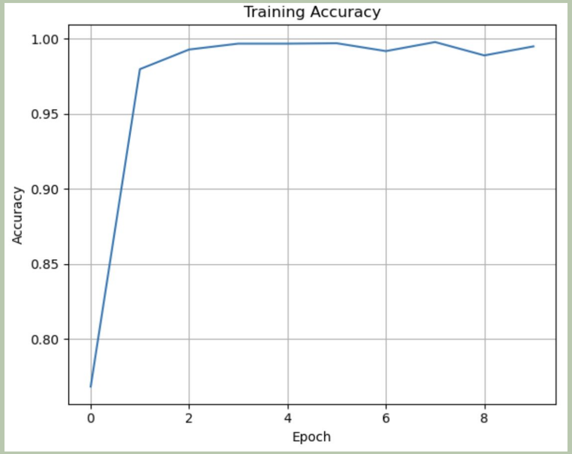
Architecture

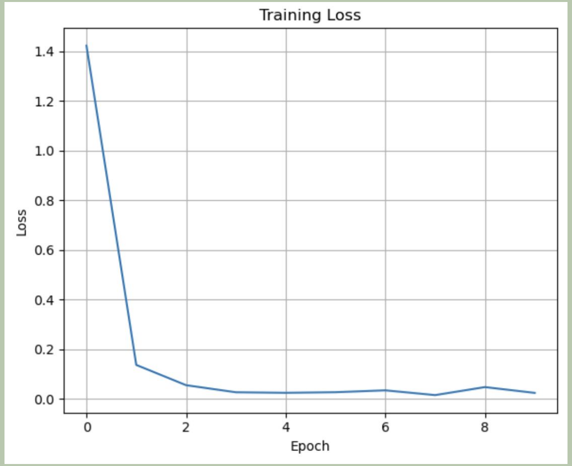
- A deep Residual Network with 50 layers, designed to combat vanishing gradients and allow efficient training of very deep models

- Initial layers:
  Conv2D(64, 7x7) → MaxPooling(3x3)

- Core:
  Uses bottleneck blocks:
  Conv1x1 → Conv3x3 → Conv1x1 + skip connection
  repeated across 4 stages:
  Conv2_x: 3 blocks
  Conv3_x: 4 blocks
  Conv4_x: 6 blocks
  Conv5_x: 3 blocks

- Final layers:
  GlobalAveragePooling → Dense(1000) → Softmax





ResNet-50 Model

Training Setup - Use best hyperparameters

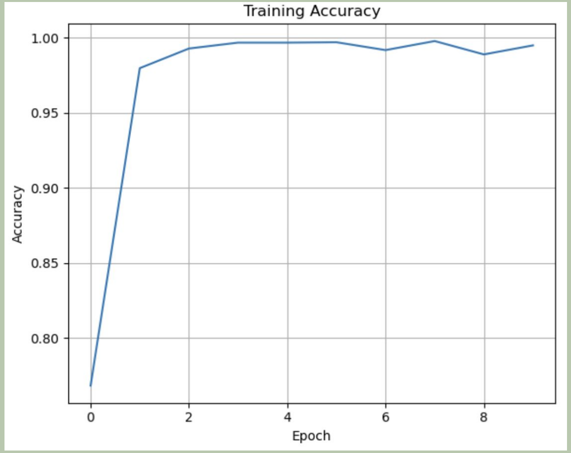
- Loss Function: CrossEntropyLoss
- Optimizer: Adam
- Epochs: 10 (fine-tuning on custom dataset)

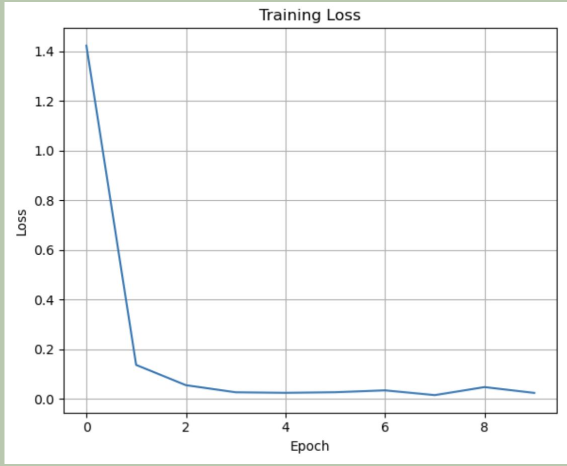
Results

- **Final Training Accuracy**: 99.76%
- **Test Accuracy**: 97.80%
- Notable improvement over baseline CNN due to deeper architecture and pretrained features

Misclassified Classes:

- Avocado: 0.680 (34/50)
- Apple: 0.833 (40/48)
- Corn Husk: 0.860 (43/50)
- Corn: 0.860 (43/50)
- Cherry: 0.898 (44/49)
- Eggplant: 0.960 (48/50)
- Blackberrie: 0.977 (43/44)





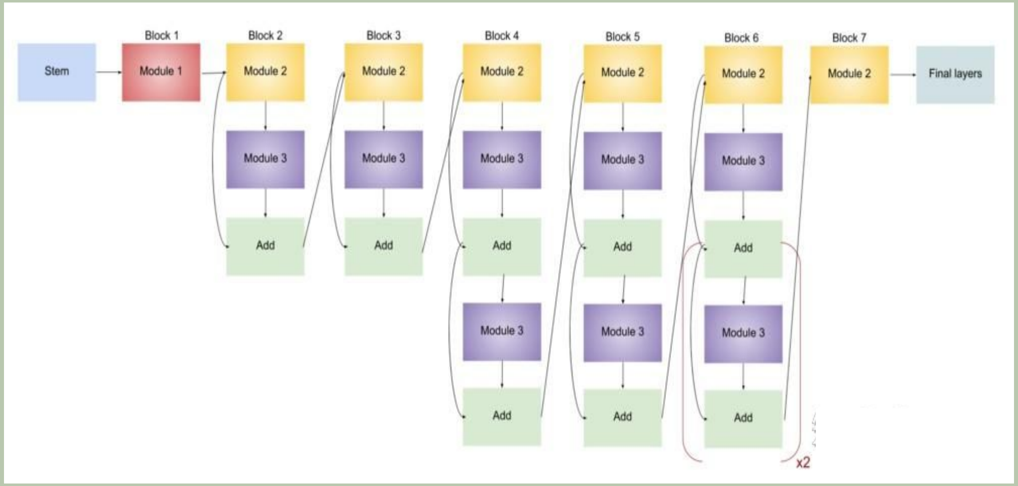
Efficient Model in Computer Vision

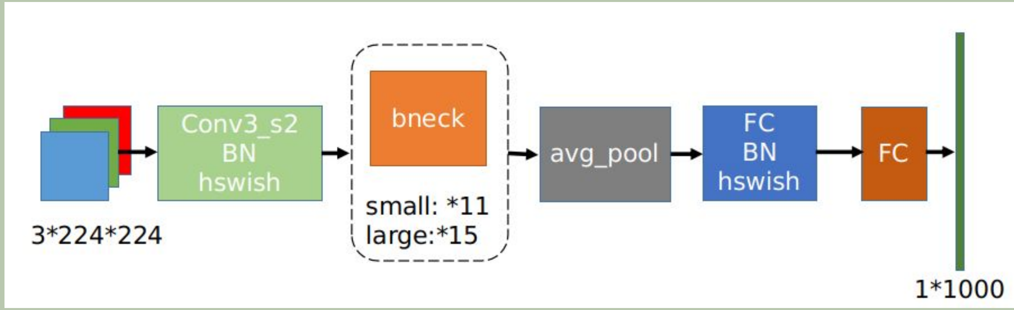
EfficientNet-B0 (5.3M parameters):

- Uses compound scaling to balance depth, width, and resolution.
- Better accuracy with fewer parameters.
- Good for tasks with limited compute budget.

MobileNet-V3 (5.4M parameters):

- Lightweight CNN optimized for mobile and embedded devices.
- Combines depthwise separable convolutions with squeeze-and-excitation modules.
- Fast inference speed, suitable for real-time applications.





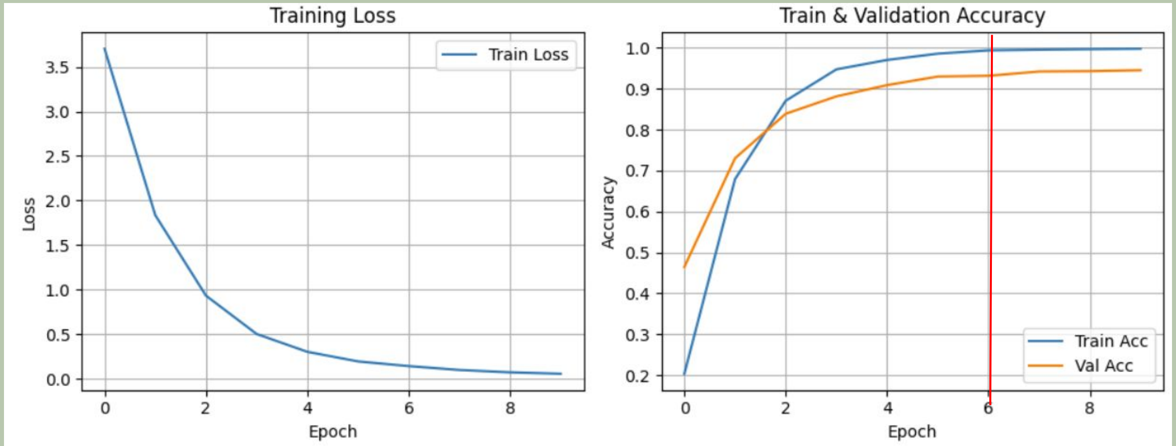
Training Setting

Setting	Details
Models	EfficientNet-B0 and MobileNet-V3
Loss Function	CrossEntropyLoss
Optimizer	AdamW (lr=3e-5)
Epochs	10
Device	GPU (if available)
Training Process	Train on training set, evaluate on validation set
Metrics	Training loss, training accuracy, validation accuracy

Result

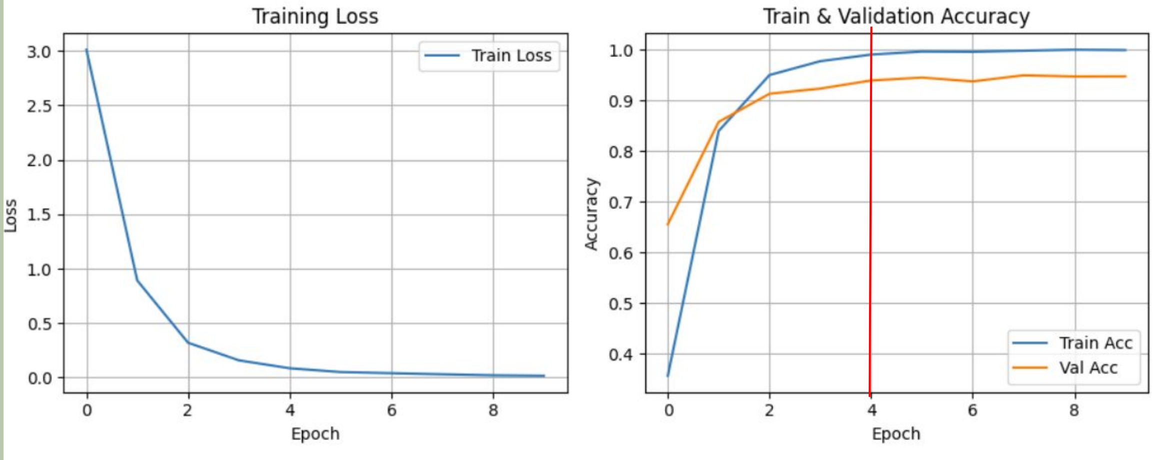
EfficientNet-B0：

Training loss: 99.77%
Test loss: 94.52%



MobileNet-V3：

Training loss: 99.97%
Test loss: 94.72%



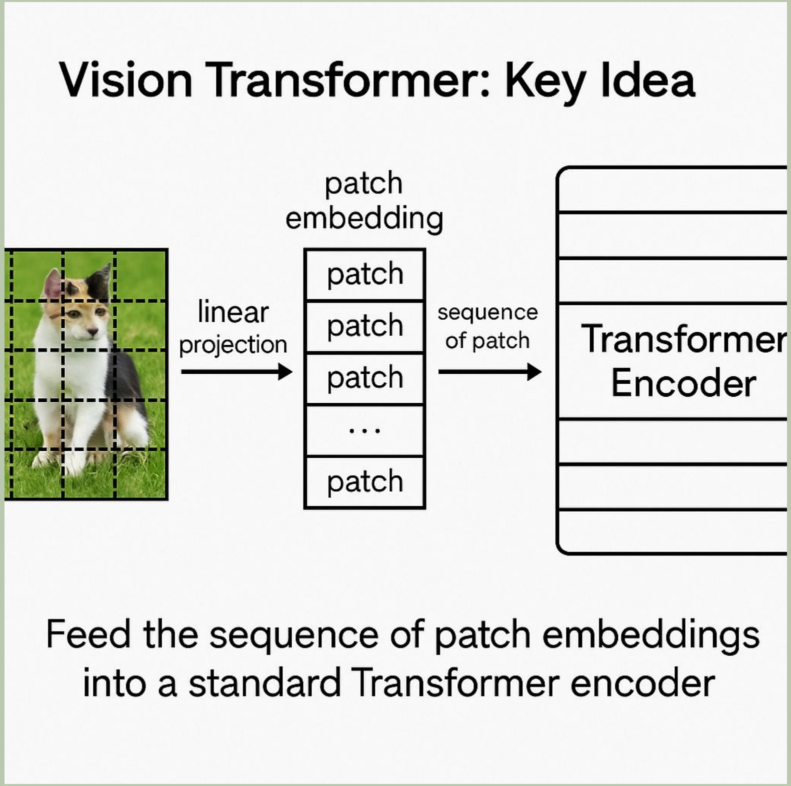
Transformer in Computer Vision

Transformers, originally developed for natural language processing, have been successfully adapted to computer vision by treating images as sequences of patches instead of words.

The key idea is to:

- **Split an image into fixed-size patches**
- **Flatten and linearly project each patch into an embedding vector**
- **Feed the sequence of patch embeddings into a standard Transformer encoder**

This allows the model to capture **global dependencies** between image regions using **self-attention**, unlike CNNs which focus on local neighborhoods



Architecture Overview

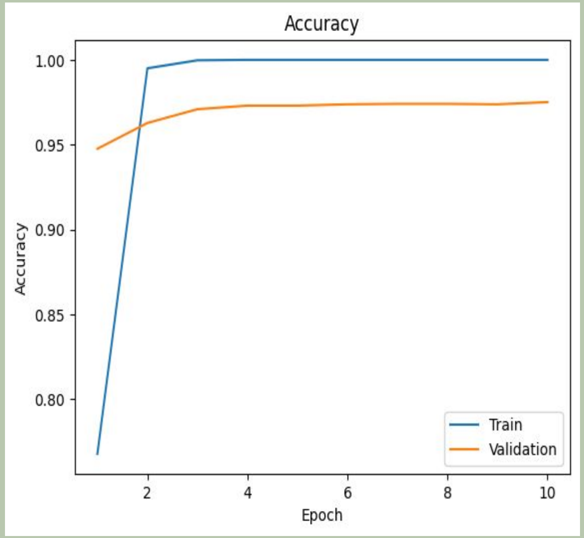
Source:
ViT (Vision Transformer): Implemented via **timm** library (~ 327 Mb)
Swin(Shifted Window Transformer) :Implemented via **timm** library (~105 Mb)

Component	**ViT**	**Swin Transformer**
Patch Embedding	Linear projection of 16×16 patches	Conv2D patch embed with 4×4 kernel
Position Encoding	Absolute positional embeddings	Implicit via window shifting
Attention	Global self-attention	Local window-based self-attention
Structure	Flat, fixed resolution	Hierarchical, multi-stage
Downsampling	Not used	Patch merging between stages
Head	[CLS] token + linear classifier	Global average pooling + linear classifier

Unlike ViT, which uses **global attention**, Swin Transformer restricts self-attention to **non-overlapping local windows**.

Training Process

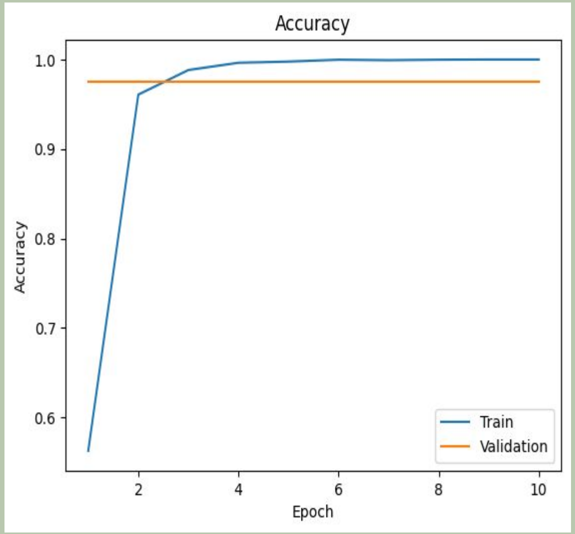
ViT



Training accuracy: 100%
Test accuracy: 97.51%

Misclassified Classes:
Avocado: 0.680 (34/50)
Apple: 0.688 (33/48)
Corn Husk: 0.720 (36/50)
Beetroot: 0.760 (38/50)
Pepino: 0.820 (41/50)
Pear: 0.878 (43/49)
Tomato: 0.898 (44/49)
Corn: 0.900 (45/50)
Blackberrie: 0.909 (40/44)
Eggplant: 0.940 (47/50)
Cucumber: 0.952 (40/42)
Cherry: 0.959 (47/49)
Strawberry: 0.980 (49/50)
Potato Red: 0.980 (49/50)

Swin



Training accuracy: 100%
Test accuracy: 98.82%

Misclassified Classes:
Avocado: 0.680 (34/50)
Corn Husk: 0.780 (39/50)
Pear: 0.796 (39/49)
Cherry: 0.796 (39/49)
Apple: 0.833 (40/48)
Tomato: 0.837 (41/49)
Pepino: 0.960 (48/50)
Eggplant: 0.980 (49/50)

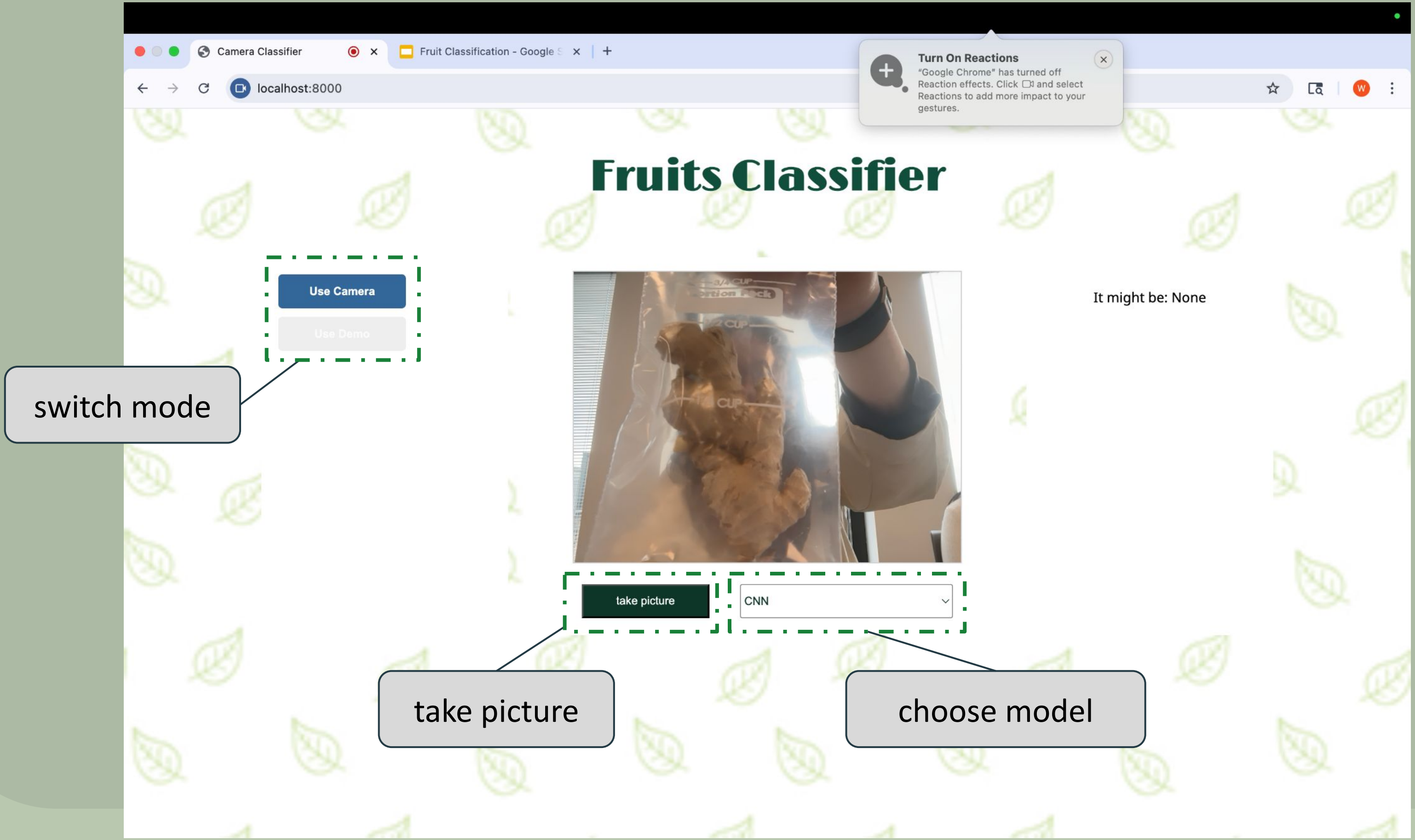
Model Comparison

Model	**CNN(base)**	**ResNet-50**	**EfficientNet-B0**	**MobileNet-V3**	**ViT**	**Swin**
Training Accuracy	95.40%	99.76%	99.77%	99.97%	100%	**100%**
Test accuracy	88.52%	97.80%	94.52%	94.71%	97.51%	**98.82%**

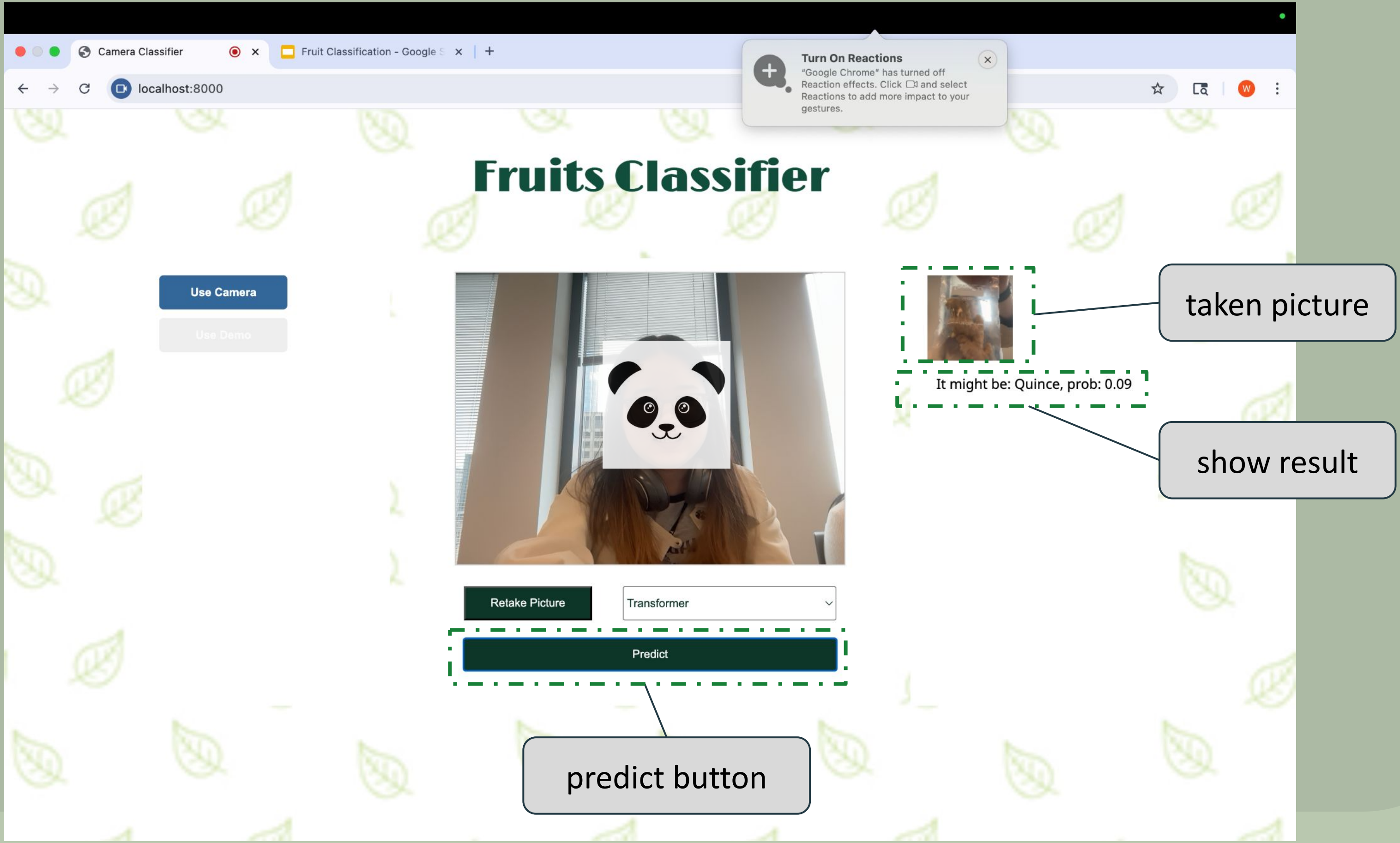
Conclusion

- **CNN (base)** achieves decent training accuracy but shows signs of underfitting compared to deeper architectures.
- **ResNet-50** is a strong and reliable choice with high generalization capability.
- Lightweight models like **EfficientNet-B0** and **MobileNet-V3** offer good trade-offs between accuracy and efficiency, though they may not match transformer models in pure performance.
- **Swin Transformer** shows the **best overall performance**, achieving perfect training accuracy and the **highest test accuracy**

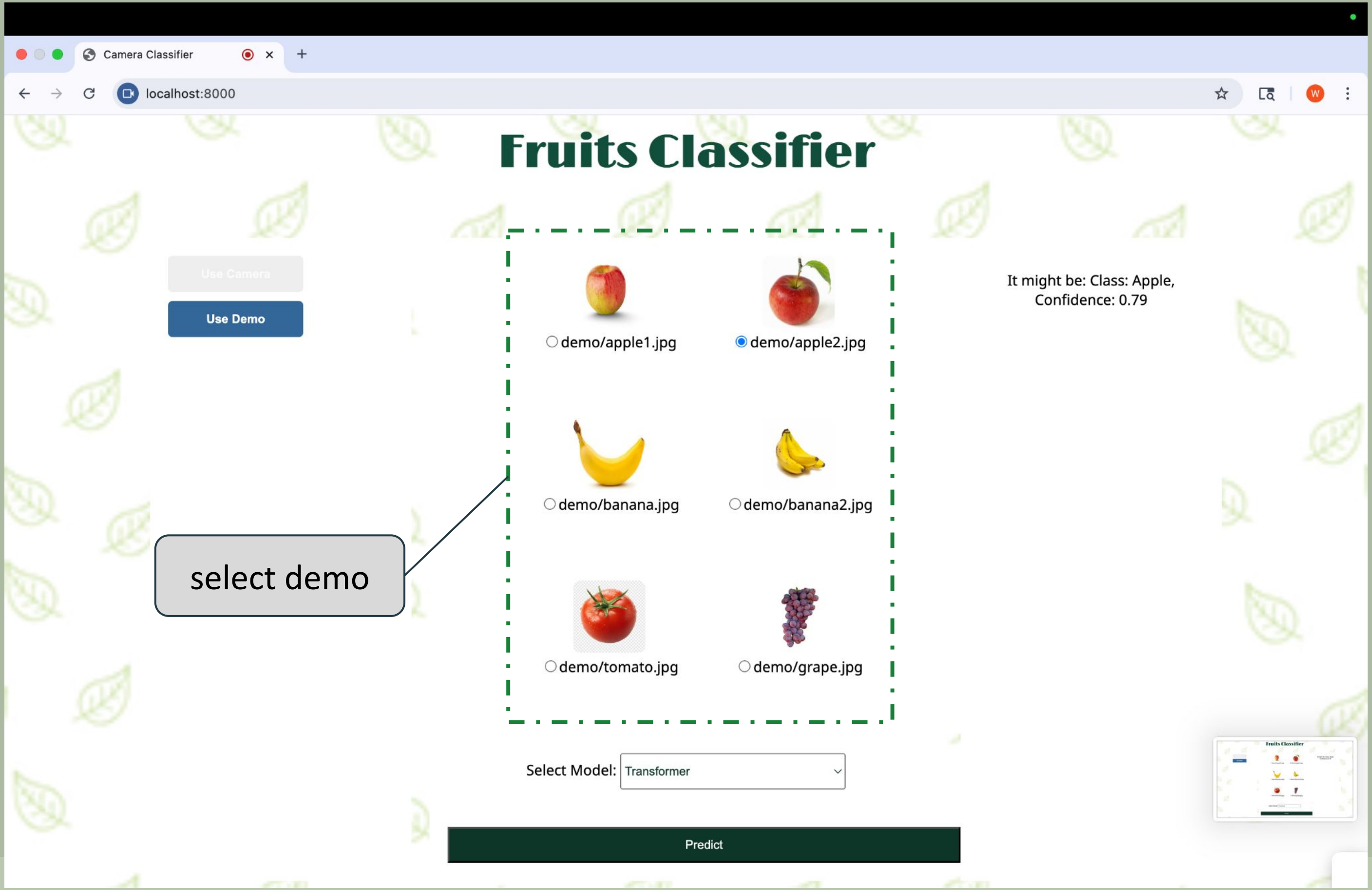
Sample Interface - Use Camera



Sample Interface - Use Camera



Sample Interface - Use Demo



Thank You!
Q&A