Plant species classification

Svanidze Mariam

My data

47 plant species:

Calathea

Kalanchoe

Ficus elastica

Money Tree

Aloe Vera

Dracaena

Schefflera

Orchid

Yucca

Tulip

etc

Data processing

Split data to train and test sets taking 80% and 20% corresponding percentage from each class

```
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
train_val_idx, test_idx = next(split.split(np.zeros(len(labels)), labels))
```

UserWarning: Palette images with Transparency expressed in bytes should be converted to RGBA images

Solution:

```
convert_palette_images_to_rgb('/content/house_plant_species/house_plant_species', save_new=False)
delete_corrupted_images('/content/house_plant_species/house_plant_species')
```

Add Kfold Cross-validation:

```
for fold, (train_idx, val_idx) in enumerate(skf.split(np.zeros(len(labels_train_val)), labels_train_val)):
=== Fold 1 === === Fold 2 === === Fold 3 === === Fold 4 === === Fold 5 ===
```

Correct test and train sets:

```
for fold, (train_idx, val_idx) in enumerate(skf.split(np.zeros(len(labels_train_val)), labels_train_val)):
    train_subset = Subset(train_val_subset, train_idx)
    val_subset = Subset(train_val_subset, val_idx)
```

Final Result

```
Train/Val classes: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]
Test classes:      [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]

=== Fold 1 ===
Train classes: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]
Val classes:   [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]

=== Fold 2 ===
Train classes: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]
Val classes:   [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]

=== Fold 3 ===
Train classes: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]
Val classes:   [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]

=== Fold 4 ===
Train classes: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]
Val classes:   [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]

=== Fold 5 ===
Train classes: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]
Val classes:   [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46]
```

My first model

Epoch [1/10], Loss: 3.3205, Train Acc: 12.29%, Test Acc: 20.07%
Epoch [2/10], Loss: 2.7619, Train Acc: 24.66%, Test Acc: 26.23%
Epoch [3/10], Loss: 2.3523, Train Acc: 34.03%, Test Acc: 29.54%
Epoch [4/10], Loss: 1.9357, Train Acc: 45.34%, Test Acc: 31.13%
Epoch [5/10], Loss: 1.5318, Train Acc: 55.89%, Test Acc: 30.93%
Epoch [6/10], Loss: 1.1150, Train Acc: 67.15%, Test Acc: 29.48%
Epoch [7/10], Loss: 0.7394, Train Acc: 78.32%, Test Acc: 29.54%
Epoch [8/10], Loss: 0.4189, Train Acc: 87.69%, Test Acc: 28.60%
Epoch [9/10], Loss: 0.2441, Train Acc: 92.76%, Test Acc: 28.70%
Epoch [10/10], Loss: 0.1371, Train Acc: 96.15%, Test Acc: 27.72%

Pred: Tradescantia
Actual: Ctenanthe

Pred: Snake plant (Sanseviera)
Actual: Jade plant (Crassula ovata)

```python
def MScnn(num_classes):
    # Feature extraction
    feature_extractor = nn.Sequential(
        nn.Conv2d(3, 6, kernel_size=5),     # Output: (128 - 5 + 1) = 124 → 6x124x124
        nn.ReLU(),
        nn.MaxPool2d(2, 2),                 # → 6x62x62
        nn.Conv2d(6, 16, kernel_size=5),    # → (62 - 5 + 1) = 58 → 16x58x58
        nn.ReLU(),
        nn.MaxPool2d(2, 2)                  # → 16x29x29
    )

    # Fully connected classifier
    classifier = nn.Sequential(
        nn.Linear(16 * 29 * 29, 120),       # Flattened features
        nn.ReLU(),
        nn.Linear(120, 84),
        nn.ReLU(),
        nn.Linear(84, num_classes)          # Output layer
    )

    # Forward pass function
    def forward_function(x):
        x = feature_extractor(x)
        x = torch.flatten(x, 1)  # flatten all except batch
        x = classifier(x)
        return x

    # Build full model
    model = nn.Sequential(feature_extractor, nn.Flatten(), classifier)
    model.forward = forward_function

    return model
```

My second model

```python
class ThirdMScnn(nn.Module):
    def __init__(self, num_classes):
        super(ThirdMScnn, self).__init__()

        self.feature_extractor = nn.Sequential(
            nn.Conv2d(3, 6, kernel_size=5),      # → 6x124x124
            nn.BatchNorm2d(6),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),                  # → 6x62x62
            nn.Dropout(0.25),

            nn.Conv2d(6, 16, kernel_size=3),     # → 16x60x60
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),                  # → 16x30x30
            nn.Dropout(0.25),
        )

        self.classifier = nn.Sequential(
            nn.Linear(16 * 30 * 30, 120),
            nn.ReLU(),

            nn.Linear(120, 84),
            nn.ReLU(),

            nn.Linear(84, num_classes)
        )
```

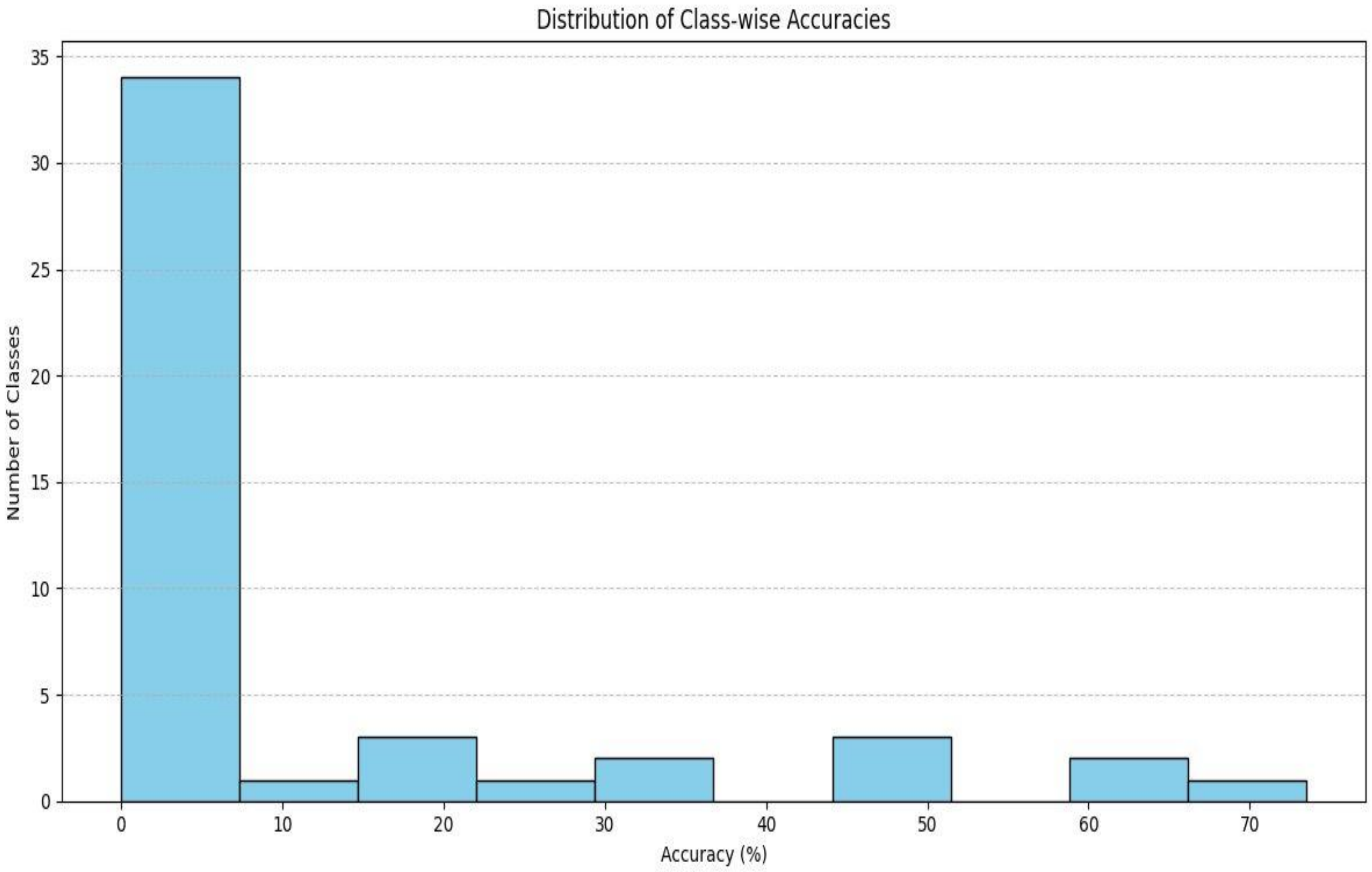
Adding data augmentation

```python
train_transform = transforms.Compose([
    transforms.Resize((128, 128), interpolation=InterpolationMode.BILINEAR),
    transforms.RandomAffine(degrees=45, translate=(0.1, 0.1), shear=15),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(brightness=0.9, contrast=0.9),
    transforms.ToTensor(),
    transforms.Normalize(mean, std)
])
```

Weight initialization

```python
def init_weights(m):
    if isinstance(m, nn.Conv2d):
        nn.init.kaiming_normal_(m.weight, mode='fan_out', nonlinearity='relu')
        if m.bias is not None:
            nn.init.constant_(m.bias, 0)
    elif isinstance(m, nn.Linear):
        nn.init.kaiming_normal_(m.weight)
        nn.init.constant_(m.bias, 0)
    elif isinstance(m, nn.BatchNorm2d):
        nn.init.constant_(m.weight, 1)
        nn.init.constant_(m.bias, 0)
```

Class accuracy distribution



Epoch [1/20], Loss: 3.7200, Train Acc: 3.81%, Val Acc: 6.56%
Epoch [2/20], Loss: 3.4464, Train Acc: 6.82%, Val Acc: 8.51%
Epoch [3/20], Loss: 3.4120, Train Acc: 8.54%, Val Acc: 9.44%
Epoch [4/20], Loss: 3.1666, Train Acc: 9.25%, Val Acc: 12.06%
Epoch [5/20], Loss: 3.1588, Train Acc: 9.62%, Val Acc: 11.30%
Epoch [6/20], Loss: 3.0785, Train Acc: 10.69%, Val Acc: 12.23%
Epoch [7/20], Loss: 3.1652, Train Acc: 11.03%, Val Acc: 12.95%
Epoch [8/20], Loss: 3.1107, Train Acc: 11.55%, Val Acc: 12.40%
Epoch [9/20], Loss: 3.1090, Train Acc: 11.34%, Val Acc: 13.12%
Epoch [10/20], Loss: 3.0969, Train Acc: 12.01%, Val Acc: 13.80%
Final Test Accuracy: 14.92% for 20 epochs

Zero accuracy model

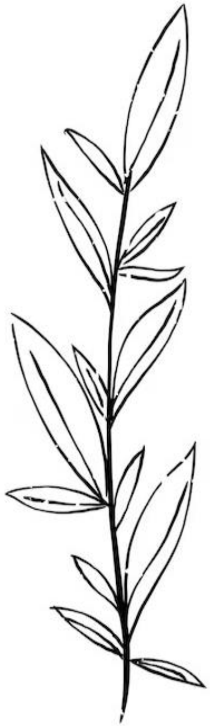
Epoch 1/10 - Train Loss: 3.2668 - Test Accuracy: 21.52%
Epoch 2/10 - Train Loss: 2.6699 - Test Accuracy: 25.18%
Epoch 3/10 - Train Loss: 2.2177 - Test Accuracy: 30.63%
Epoch 4/10 - Train Loss: 1.7280 - Test Accuracy: 32.39%
Epoch 5/10 - Train Loss: 1.2307 - Test Accuracy: 33.10%
Epoch 6/10 - Train Loss: 0.7755 - Test Accuracy: 31.24%
Epoch 7/10 - Train Loss: 0.4228 - Test Accuracy: 30.59%
Epoch 8/10 - Train Loss: 0.2082 - Test Accuracy: 30.49%
Epoch 9/10 - Train Loss: 0.1167 - Test Accuracy: 29.00%
Epoch 10/10 - Train Loss: 0.0786 - Test Accuracy: 29.54%

```
Accuracy for class: African Violet (Saintpaulia ionantha) is 28.4 %
Accuracy for class: Aloe Vera            is 0.0 %
Accuracy for class: Anthurium (Anthurium andraeanum) is 1.1 %
Accuracy for class: Areca Palm (Dypsis lutescens) is 0.0 %
Accuracy for class: Asparagus Fern (Asparagus setaceus) is 0.0 %
Accuracy for class: Begonia (Begonia spp.) is 0.0 %
Accuracy for class: Bird of Paradise (Strelitzia reginae) is 0.0 %
Accuracy for class: Birds Nest Fern (Asplenium nidus) is 0.0 %
Accuracy for class: Boston Fern (Nephrolepis exaltata) is 0.0 %
Accuracy for class: Calathea             is 1.5 %
Accuracy for class: Cast Iron Plant (Aspidistra elatior) is 0.0 %
Accuracy for class: Chinese Money Plant (Pilea peperomioides) is 0.0 %
Accuracy for class: Chinese evergreen (Aglaonema) is 7.8 %
Accuracy for class: Christmas Cactus (Schlumbergera bridgesii) is 1.6 %
Accuracy for class: Chrysanthemum        is 0.0 %
Accuracy for class: Ctenanthe            is 2.9 %
Accuracy for class: Daffodils (Narcissus spp.) is 45.2 %
Accuracy for class: Dracaena             is 0.0 %
Accuracy for class: Dumb Cane (Dieffenbachia spp.) is 62.0 %
Accuracy for class: Elephant Ear (Alocasia spp.) is 0.0 %
Accuracy for class: English Ivy (Hedera helix) is 0.0 %
Accuracy for class: Hyacinth (Hyacinthus orientalis) is 32.8 %
Accuracy for class: Iron Cross begonia (Begonia masoniana) is 1.9 %
Accuracy for class: Jade plant (Crassula ovata) is 18.3 %
Accuracy for class: Kalanchoe            is 0.0 %
Accuracy for class: Lilium (Hemerocallis) is 65.6 %
Accuracy for class: Lily of the valley (Convallaria majalis) is 73.5 %
Accuracy for class: Money Tree (Pachira aquatica) is 0.0 %
Accuracy for class: Monstera Deliciosa (Monstera deliciosa) is 20.0 %
Accuracy for class: Orchid               is 6.4 %
Accuracy for class: Parlor Palm (Chamaedorea elegans) is 0.0 %
Accuracy for class: Peace lily           is 6.5 %
Accuracy for class: Poinsettia (Euphorbia pulcherrima) is 50.8 %
Accuracy for class: Polka Dot Plant (Hypoestes phyllostachya) is 48.5 %
Accuracy for class: Ponytail Palm (Beaucarnea recurvata) is 0.0 %
Accuracy for class: Pothos (Ivy arum)    is 0.0 %
Accuracy for class: Prayer Plant (Maranta leuconeura) is 0.0 %
Accuracy for class: Rattlesnake Plant (Calathea lancifolia) is 3.2 %
Accuracy for class: Rubber Plant (Ficus elastica) is 0.0 %
Accuracy for class: Sago Palm (Cycas revoluta) is 0.0 %
Accuracy for class: Schefflera           is 0.0 %
Accuracy for class: Snake plant (Sanseviera) is 6.3 %
Accuracy for class: Tradescantia         is 33.8 %
Accuracy for class: Tulip                is 0.0 %
Accuracy for class: Venus Flytrap        is 5.0 %
Accuracy for class: Yucca                is 0.0 %
Accuracy for class: ZZ Plant (Zamioculcas zamiifolia) is 21.6 %
```

My third model

Selected classes:

Begonia
Calathea
Dracaena
Sanseviera
Ctenanthe
Aglaonema
Anthurium
Tradescantia
Monstera Deliciosa
Maranta leuconeura
Dumb Cane (Dieffenbachia)
Money Tree (Pachira aquatica)



```
def MScnn(num_classes):
    feature_extractor = nn.Sequential(
        nn.Conv2d(3, 6, kernel_size=5),     # → 220x220
        nn.ReLU(),
        nn.MaxPool2d(2, 2),                 # → 110x110
        nn.Conv2d(6, 16, kernel_size=5),    # → 106x106
        nn.ReLU(),
        nn.MaxPool2d(2, 2)                  # → 53x53
    )

    classifier = nn.Sequential(
        nn.Linear(16 * 53 * 53, 120),
        nn.ReLU(),
        nn.Linear(120, 84),
        nn.ReLU(),
        nn.Linear(84, num_classes)
    )

    def forward_function(x):
        x = feature_extractor(x)
        x = torch.flatten(x, 1)
        x = classifier(x)
        return x

    model = nn.Sequential(feature_extractor, nn.Flatten(), classifier)
    model.forward = forward_function
    return model
```

My third model

Selected classes:
Begonia
Calathea
Dracaena
Sanseviera
Ctenanthe
Aglaonema
Anthurium
Tradescantia
Monstera Deliciosa
Maranta leuconeura
Dumb Cane (Dieffenbachia)
Money Tree (Pachira aquatica)

Outcome:

Epoch [1/10], Loss: 2.4640, Train Acc: 10.75%, Test Acc: 18.86%
Epoch [2/10], Loss: 2.2097, Train Acc: 23.09%, Test Acc: 25.85%
Epoch [3/10], Loss: 2.0342, Train Acc: 28.73%, Test Acc: 27.97%
Epoch [4/10], Loss: 1.9165, Train Acc: 33.02%, Test Acc: 29.87%
Epoch [5/10], Loss: 1.8056, Train Acc: 36.68%, Test Acc: 29.66%
Epoch [6/10], Loss: 1.6470, Train Acc: 42.03%, Test Acc: 32.42%
Epoch [7/10], Loss: 1.4319, Train Acc: 50.08%, Test Acc: 32.94%
Epoch [8/10], Loss: 1.0327, Train Acc: 64.22%, Test Acc: 32.63%
Epoch [9/10], Loss: 0.6522, Train Acc: 77.09%, Test Acc: 33.47%
Epoch [10/10], Loss: 0.3434, Train Acc: 88.08%, Test Acc: 33.47%

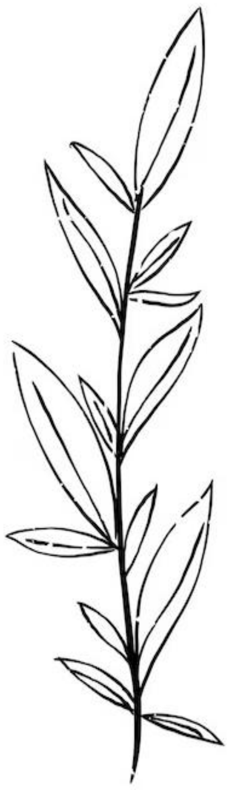
Results

Train Accuracy: 94.57%

Test Accuracy: 33.47%

Accuracy per class

Begonia : 97.85%
Calathea : 97.35%
Dracaena : 100.00%
Sanseviera : 97.16%
Ctenanthe : 98.18%
Aglaonema: 85.64%
Anthurium : 96.69%
Tradescantia : 97.80%
Monstera Deliciosa : 94.75%
Maranta leuconeura : 95.62%
Dumb Cane (Dieffenbachia) : 85.45%
Money Tree (Pachira aquatica) : 98.95%



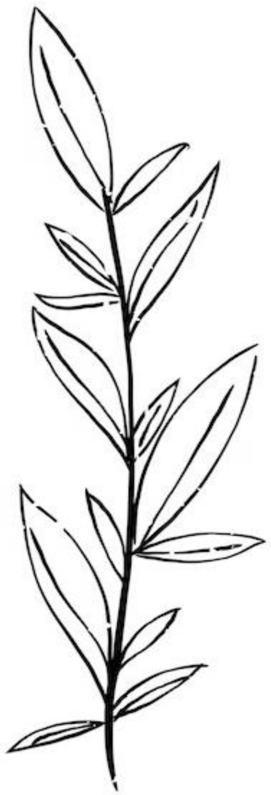
Begonia: 19.15%
Calathea : 18.18%
Dracaena : 26.92%
Sanseviera: 46.84%
Ctenanthe : 28.99%
Aglaonema : 14.56%
Anthurium: 60.44%
Tradescantia : 50.00%
Monstera Deliciosa: 34.86%
Maranta leuconeura: 21.25%
DumbCane (Dieffenbachia) : 28.70%
Money Tree (Pachira aquatica) : 47.22%

Pre-trained model example

```python
# Load pretrained VGG16 and modify for 47 classes
vgg16 = models.vgg16(pretrained=True)
for param in vgg16.parameters():
    param.requires_grad = False  # Freeze feature extractor

# Replace final layer
vgg16.classifier[6] = nn.Linear(4096, 47)
vgg16 = vgg16.to(device)

# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(vgg16.classifier[6].parameters(), lr=0.001)
```



Results:

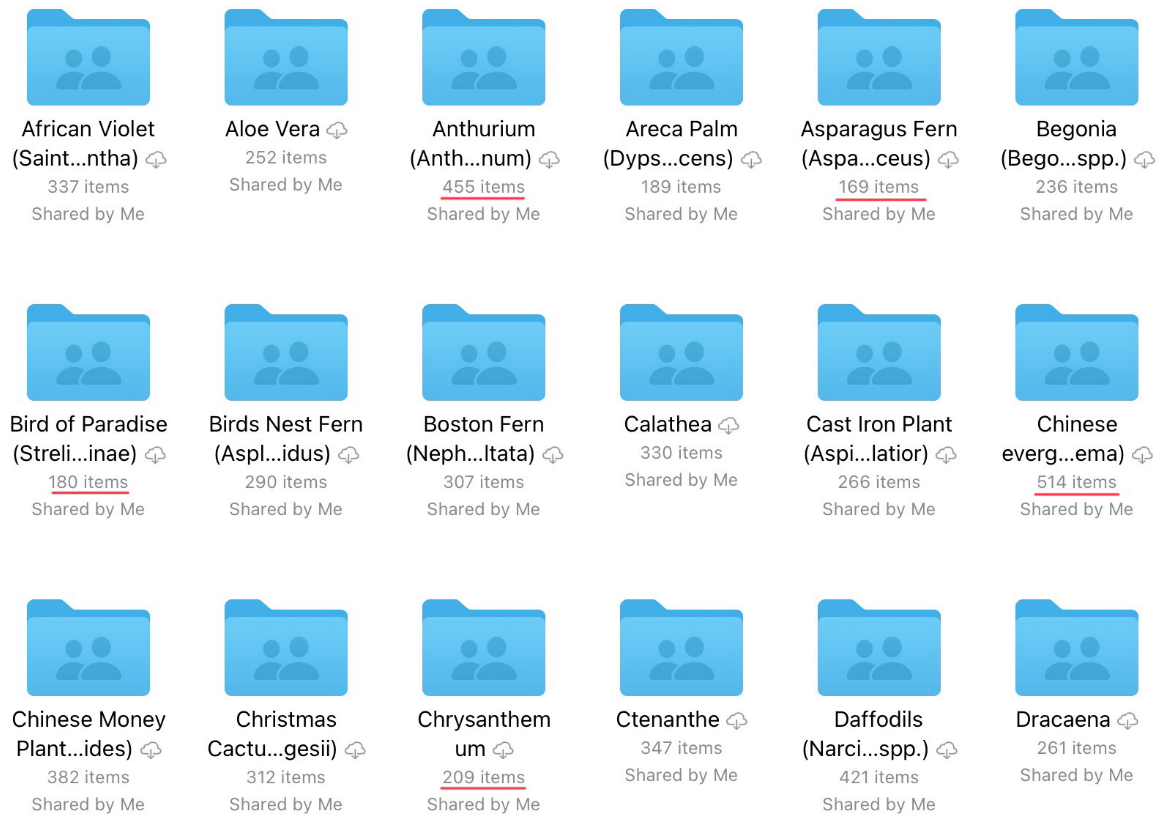
Epoch [1/10], Loss: 408.5554, Train Acc: 27.68%
Epoch [2/10], Loss: 334.3769, Train Acc: 37.95%
Epoch [3/10], Loss: 326.3154, Train Acc: 40.38%
Epoch [4/10], Loss: 318.5549, Train Acc: 41.23%

Data issues

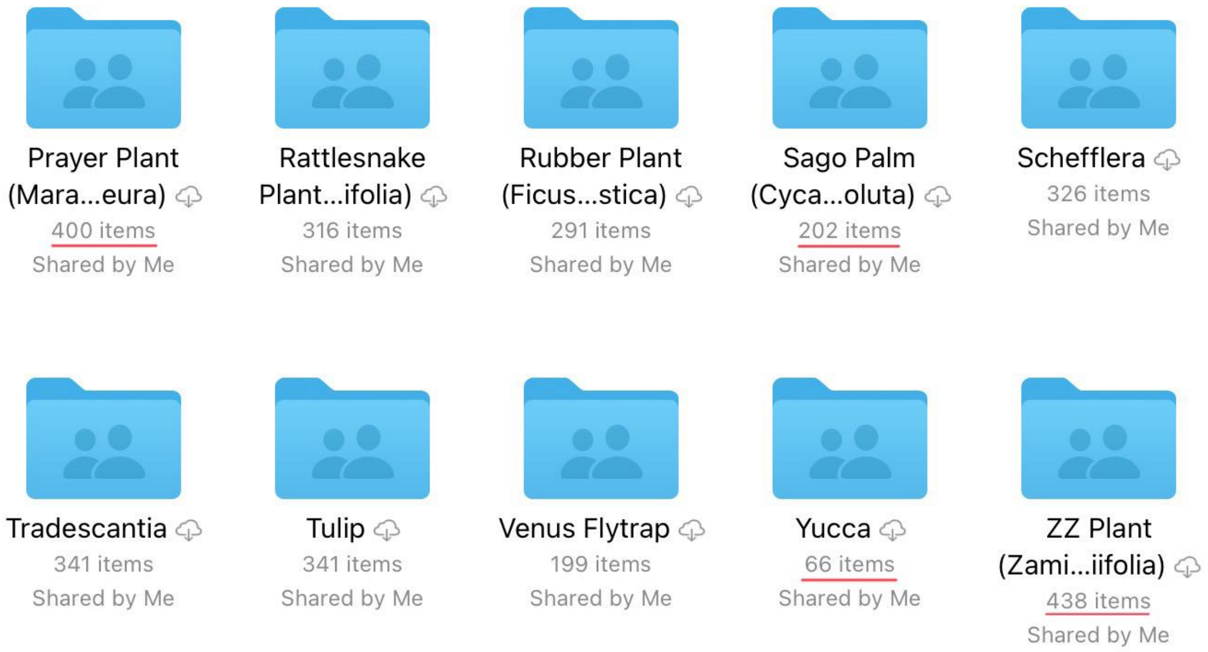
Unrealistic images

Money tree (Pachira aquatica)

Unbalanced classes



252. 455. 189. 169. 180. 307. 514. 209. 421. 261.



400. 316. 291. 202. 326. 341. 341. 199. 66. 438.

Variegation within a plant species

Begonia

Antherium

Similarities across different species

Marantaceae family

Maranta

Calathea

Ctenanthe

Every seed planted today is tomorrow's harvest.