Data Preprocessing

In [2]:

```
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import cv2
import numpy as np
import glob
import random
```

In [ ]:

```
def resize_with_pad(img, target_size=(100, 100), pad_color=(0, 0, 0)):
    h, w = img.shape[:2]
    target_w, target_h = target_size

    scale = min(target_w / w, target_h / h)
    new_w = int(w * scale)
    new_h = int(h * scale)
    resized_img = cv2.resize(img, (new_w, new_h), interpolation=cv2.INTER_AREA)

    delta_w = target_w - new_w
    delta_h = target_h - new_h
    top, bottom = delta_h // 2, delta_h - (delta_h // 2)
    left, right = delta_w // 2, delta_w - (delta_w // 2)

    padded_img = cv2.copyMakeBorder(resized_img, top, bottom, left, right,
                                   borderType=cv2.BORDER_CONSTANT, value=pad_color)
    return padded_img


def split_data(data, val_ratio=0.2):
    random.shuffle(data)
    val_size = int(len(data) * val_ratio)
    return data[val_size:], data[:val_size]
```

Train, Test and Validation Split

In [3]:

```
raw_path_train="../data/raw/ninjacart_data/train"
raw_path_test="../data/raw/ninjacart_data/test"

train_onions = glob.glob(raw_path_train + "/onion/*.*")
train_potatoes = glob.glob(raw_path_train + "/potato/*.*")
train_tomatoes = glob.glob(raw_path_train + "/tomato/*.*")
train_indian_market=glob.glob(raw_path_train + "/indian_market/*.*")

train_onions, val_onions = split_data(train_onions)
train_potatoes, val_potatoes = split_data(train_potatoes)
train_tomatoes, val_tomatoes = split_data(train_tomatoes)
train_indian_market, val_indian_market = split_data(train_indian_market)

test_onions = glob.glob(raw_path_test + "/onion/*.*")
test_potatoes = glob.glob(raw_path_test + "/potato/*.*")
test_tomatoes = glob.glob(raw_path_test + "/tomato/*.*")
test_indian_market=glob.glob(raw_path_test + "/indian_market/*.*")

all_train_data = train_onions + train_potatoes + train_tomatoes + train_indian_market
all_test_data = test_onions + test_potatoes + test_tomatoes + test_indian_market
all_val_data = val_onions + val_potatoes + val_tomatoes + val_indian_market
```

In [4]:

```
print("Onions: ", len(train_onions), len(val_onions), len(test_onions))
print("Potatoes: ", len(train_potatoes), len(val_potatoes), len(test_potatoes))
print("Tomatoes: ", len(train_tomatoes), len(val_tomatoes), len(test_tomatoes))
print("Indian Markets: ", len(train_indian_market), len(val_indian_market), len(test_indian_market))
```

```
Onions:  680 169 83
Potatoes:  719 179 81
Tomatoes:  632 157 106
Indian Markets:  480 119 81
```

In [14]:

```
for pth in all_test_data + all_train_data:
    img = cv2.imread(pth)
    output_path = pth.replace("raw", "processed")
    processed_img = resize_with_pad(img)
    cv2.imwrite(output_path, processed_img)


for pth in all_val_data:
    img = cv2.imread(pth)
    output_path = pth.replace("../data/raw/ninjacart_data/train", "../data/processed/ninjacart_data/val")
    processed_img = resize_with_pad(img)
    cv2.imwrite(output_path, processed_img)
```

In [5]:

```
data_dir = "../data/processed/ninjacart_data/train"

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

dataset = datasets.ImageFolder(data_dir, transform=transform)
loader = DataLoader(dataset, batch_size=32, shuffle=False)

mean = torch.zeros(3)
std = torch.zeros(3)
total_images_count = 0

for images, _ in loader:
    batch_samples = images.size(0)
    images = images.view(batch_samples, images.size(1), -1)
    mean += images.mean(2).sum(0)
    std += images.std(2).sum(0)
    total_images_count += batch_samples

mean /= total_images_count
std /= total_images_count

print("Mean:", mean)
print("Std:", std)
```

```
Mean: tensor([0.4136, 0.3702, 0.3049])
Std: tensor([0.2880, 0.2694, 0.2525])
```

Image Transformation

In [6]:

```
data_dir = "../data/processed/ninjacart_data"

def generate_dataloader(mean, std):

    train_transforms = transforms.Compose([
        transforms.RandomHorizontalFlip(p=0.5),  # simulate camera flipping
        transforms.RandomRotation(15),           # allow small camera tilt
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2),  # natural lighting changes
        transforms.RandomAffine(translate=(0.1, 0.1), degrees=0),  # small object shifts

        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ])
    test_transforms = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ])

    val_transforms = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ])


    train_data = datasets.ImageFolder(data_dir + '/train', transform=train_transforms)
    test_data = datasets.ImageFolder(data_dir + '/test', transform=test_transforms)
    val_data = datasets.ImageFolder(data_dir + '/val', transform=val_transforms)

    train_loader = torch.utils.data.DataLoader(train_data, batch_size=32, shuffle=True)
    test_loader = torch.utils.data.DataLoader(test_data, batch_size=32)
    val_loader = torch.utils.data.DataLoader(val_data, batch_size=32)

    return {
        'train_loader': train_loader,
        'val_loader': val_loader,
        'test_loader': test_loader,
        'train_data': train_data,
        'val_data': val_data,
        'test_data': test_data
    }
```

Data Augmentation

In [ ]:

```
mean=[0.4136, 0.3702, 0.3049]
std=[0.2880, 0.2694, 0.2525]

content = generate_dataloader(mean, std)
train_loader = content['train_loader']
val_loader = content['val_loader']
test_loader = content['test_loader']
train_data = content['train_data']
```

In [11]:

```
train_data.classes, train_data.class_to_idx
```

Out[11]:

```
(['indian_market', 'onion', 'potato', 'tomato'],
 {'indian_market': 0, 'onion': 1, 'potato': 2, 'tomato': 3})
```

In [6]:

```
torch.save(train_loader, '../data/processed/dataset/train_loader.pth')
torch.save(test_loader, '../data/processed/dataset/test_loader.pth')
torch.save(val_loader, '../data/processed/dataset/val_loader.pth')
```

Resnet Data Augmentation

In [7]:

```
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]

content = generate_dataloader(mean, std)
train_loader = content['train_loader']
val_loader = content['val_loader']
test_loader = content['test_loader']
```

In [8]:

```
torch.save(train_loader, '../data/processed/dataset/train_loader_resnet.pth')
torch.save(test_loader, '../data/processed/dataset/test_loader_resnet.pth')
torch.save(val_loader, '../data/processed/dataset/val_loader_resnet.pth')
```