In [33]:

```
!pip install numpy pandas matplotlib seaborn plotly requests tqdm opencv-python
```

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Requirement already satisfied: numpy in /environment/miniconda3/lib/python3.10/si
te-packages (1.24.1)
Requirement already satisfied: pandas in /environment/miniconda3/lib/python3.10/s
ite-packages (2.1.2)
Requirement already satisfied: matplotlib in /environment/miniconda3/lib/python3.
10/site-packages (3.8.1)
Requirement already satisfied: seaborn in /environment/miniconda3/lib/python3.10/
site-packages (0.13.0)
Requirement already satisfied: plotly in /environment/miniconda3/lib/python3.10/s
ite-packages (5.19.0)
Requirement already satisfied: requests in /environment/miniconda3/lib/python3.1
0/site-packages (2.31.0)
Requirement already satisfied: tqdm in /environment/miniconda3/lib/python3.10/sit
e-packages (4.65.0)
Requirement already satisfied: opencv-python in /environment/miniconda3/lib/pytho
n3.10/site-packages (4.8.1.78)
Requirement already satisfied: pillow in /environment/miniconda3/lib/python3.10/s
ite-packages (9.3.0)
Requirement already satisfied: wandb in /environment/miniconda3/lib/python3.10/si
te-packages (0.16.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /environment/miniconda3/
lib/python3.10/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /environment/miniconda3/lib/python
3.10/site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /environment/miniconda3/lib/pyth
on3.10/site-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in /environment/miniconda3/lib/py
thon3.10/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /environment/miniconda3/lib/python
3.10/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /environment/miniconda3/lib/p
ython3.10/site-packages (from matplotlib) (4.44.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /environment/miniconda3/lib/p
ython3.10/site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from matplotlib) (23.0)
Requirement already satisfied: pyparsing>=2.3.1 in /environment/miniconda3/lib/py
thon3.10/site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from plotly) (8.2.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /environment/miniconda
3/lib/python3.10/site-packages (from requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /environment/miniconda3/lib/python
3.10/site-packages (from requests) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /environment/miniconda3/lib/
python3.10/site-packages (from requests) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /environment/miniconda3/lib/
python3.10/site-packages (from requests) (2023.7.22)
Requirement already satisfied: Click!=8.0.0,>=7.1 in /environment/miniconda3/lib/
python3.10/site-packages (from wandb) (7.1.2)
Requirement already satisfied: GitPython!=3.1.29,>=1.0.0 in /environment/minicond
a3/lib/python3.10/site-packages (from wandb) (3.1.42)
Requirement already satisfied: psutil>=5.0.0 in /environment/miniconda3/lib/pytho
n3.10/site-packages (from wandb) (5.9.5)
Requirement already satisfied: sentry-sdk>=1.0.0 in /environment/miniconda3/lib/p
ython3.10/site-packages (from wandb) (1.40.5)
Requirement already satisfied: docker-pycreds>=0.4.0 in /environment/miniconda3/l
ib/python3.10/site-packages (from wandb) (0.4.0)
Requirement already satisfied: PyYAML in /environment/miniconda3/lib/python3.10/s
```

```
ite-packages (from wandb) (6.0.1)
Requirement already satisfied: setproctitle in /environment/miniconda3/lib/python
3.10/site-packages (from wandb) (1.3.3)
Requirement already satisfied: setuptools in /environment/miniconda3/lib/python3.
10/site-packages (from wandb) (67.8.0)
Requirement already satisfied: appdirs>=1.4.3 in /environment/miniconda3/lib/pyth
on3.10/site-packages (from wandb) (1.4.4)
Requirement already satisfied: protobuf!=4.21.0,<5,>=3.19.0 in /environment/minic
onda3/lib/python3.10/site-packages (from wandb) (4.23.4)
Requirement already satisfied: six>=1.4.0 in /environment/miniconda3/lib/python3.
10/site-packages (from docker-pycreds>=0.4.0->wandb) (1.16.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from GitPython!=3.1.29,>=1.0.0->wandb) (4.0.11)
Requirement already satisfied: smmap<6,>=3.0.1 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from gitdb<5,>=4.0.1->GitPython!=3.1.29,>=1.0.0->wandb)
(5.0.1)
```

Download and install Pytorch

In [34]:

```
!pip3 install torch torchvision torchaudio --extra-index-url https://download.py
```

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple, https://download.py
torch.org/whl/cu113
Requirement already satisfied: torch in /environment/miniconda3/lib/python3.10/si
te-packages (2.0.1+cu118)
Requirement already satisfied: torchvision in /environment/miniconda3/lib/python
3.10/site-packages (0.15.2+cu118)
Requirement already satisfied: torchaudio in /environment/miniconda3/lib/python3.
10/site-packages (2.0.2+cu118)
Requirement already satisfied: filelock in /environment/miniconda3/lib/python3.1
0/site-packages (from torch) (3.9.0)
Requirement already satisfied: typing-extensions in /environment/miniconda3/lib/p
ython3.10/site-packages (from torch) (4.8.0)
Requirement already satisfied: sympy in /environment/miniconda3/lib/python3.10/si
te-packages (from torch) (1.11.1)
Requirement already satisfied: networkx in /environment/miniconda3/lib/python3.1
0/site-packages (from torch) (3.0)
Requirement already satisfied: jinja2 in /environment/miniconda3/lib/python3.10/s
ite-packages (from torch) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /environment/miniconda3/lib/pytho
n3.10/site-packages (from torch) (2.0.0)
Requirement already satisfied: cmake in /environment/miniconda3/lib/python3.10/si
te-packages (from triton==2.0.0->torch) (3.25.0)
Requirement already satisfied: lit in /environment/miniconda3/lib/python3.10/site
-packages (from triton==2.0.0->torch) (15.0.7)
Requirement already satisfied: numpy in /environment/miniconda3/lib/python3.10/si
te-packages (from torchvision) (1.24.1)
Requirement already satisfied: requests in /environment/miniconda3/lib/python3.1
0/site-packages (from torchvision) (2.31.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /environment/miniconda3/l
ib/python3.10/site-packages (from torchvision) (9.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from jinja2->torch) (2.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /environment/miniconda
3/lib/python3.10/site-packages (from requests->torchvision) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /environment/miniconda3/lib/python
3.10/site-packages (from requests->torchvision) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /environment/miniconda3/lib/
python3.10/site-packages (from requests->torchvision) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /environment/miniconda3/lib/
python3.10/site-packages (from requests->torchvision) (2023.7.22)
Requirement already satisfied: mpmath>=0.19 in /environment/miniconda3/lib/python
3.10/site-packages (from sympy->torch) (1.2.1)
```

```
In [35]: !wget https://zihao-openmmlab.obs.cn-east-3.myhuaweicloud.com/20220716-mmclassif
```

```
--2024-02-26 15:04:33--  https://zihao-openmmlab.obs.cn-east-3.myhuaweicloud.com/
20220716-mmclassification/dataset/SimHei.ttf
Connecting to 172.16.0.13:5848... connected.
Proxy request sent, awaiting response... 200 OK
Length: 10050868 (9.6M) [application/x-font-ttf]
Saving to: ‘SimHei.ttf.1’

SimHei.ttf.1        100%[===================>]   9.58M  20.9MB/s    in 0.5s

2024-02-26 15:04:34 (20.9 MB/s) - ‘SimHei.ttf.1’ saved [10050868/10050868]
```

Create a catalogue

In [36]:

```python
import os
```

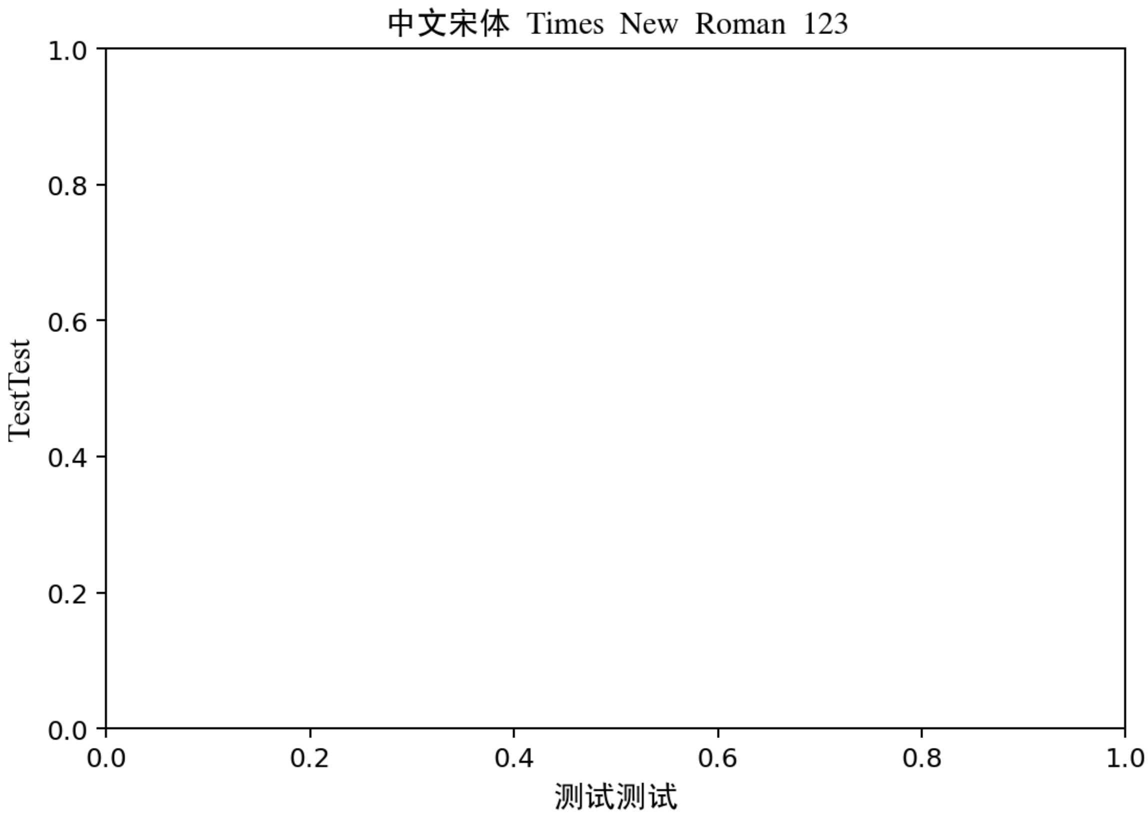
In [37]:

```python
# Store the results file
# os.mkdir('output')

# Store the trained model weights
os.mkdir('checkpoint')


# Store the generated charts
os.mkdir('diagrams')
```

Setting matplotlib Chinese and English fonts

In [38]:

```python
## Font Environment Settings
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.font_manager import FontProperties

# global font settings
SimSun = FontProperties(fname='/home/featurize/SimHei.ttf')  # Used to display C
plt.rcParams['axes.unicode_minus'] = False  # Used to display the negative sign
Times_New_Roman = FontProperties(fname='/home/featurize/times.ttf')

# mixed font settings
config = {
#     "font.family":'serif',
#     "font.size": 80,
      "mathtext.fontset":'stix',
#     "font.serif": ['SimSun'],
}
rcParams.update(config)

#Canvas Settings
fig = plt.figure(num=1, figsize=(9, 6),dpi=180)
ax = plt.axes((0.23,0.23,0.6,0.6))


# Application of font effects
ax.set_title('中文宋体 $\mathrm{Times}$ $\mathrm{New}$ $\mathrm{Roman}$ $\mathrm
                                                   ,fontproperties=SimSun,fontsize=12)

ax.set_xlabel('测试测试',fontproperties=SimSun,fontsize=12)

ax.set_ylabel('TestTest',fontproperties=Times_New_Roman,fontsize=12)


plt.show()
```



In [42]:

```
!sudo snap install tree
```

```
snap "tree" is already installed, see 'snap help refresh'
```

In [17]:

```
!tree /home/featurize/data -L 2
```

```
/home/featurize/data
├── fruit25_split.zip
├── train
│   ├── CherryTomatoes
│   ├── Mangosteen
│   ├── MomordicaCharantia
│   ├── NavelOrange
│   ├── Sandsugaroranges
│   ├── apple
│   ├── banana
│   ├── carrot
│   ├── cherries
│   ├── cucumber
│   ├── durian
│   ├── grape
│   ├── hamimelon
│   ├── kiwi
│   ├── lemon
│   ├── lichee
│   ├── longan
│   ├── mango
│   ├── pear
│   ├── pineapple
│   ├── pitaya
│   ├── pomegranate
│   ├── strawberry
│   ├── tomato
│   └── watermelon
└── val
    ├── Cherrytomatoes
    ├── Mangosteen
    ├── MomordicaCharantia
    ├── NavelOrange
    ├── Sandsugaroranges
    ├── apple
    ├── banana
    ├── carrot
    ├── cherries
    ├── cucumber
    ├── durian
    ├── grape
    ├── hamimelon
    ├── kiwi
    ├── lemon
    ├── lichee
    ├── longan
    ├── mango
    ├── pear
    ├── pineapple
    ├── pitaya
    ├── pomegranate
    ├── strawberry
    ├── tomato
    └── watermelon

52 directories, 1 file
```

In [19]:

```
import time
import os
import numpy as np
from tqdm import tqdm

import torch
import torchvision
import torch.nn as nn
import torch.nn.functional as F

import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [20]:

```
# test cpu
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
print('device', device)
```

```
device cuda:0
```

In [21]:

```
from torchvision import transforms

# Training Set Image Preprocessing - RCTN: Scaling, Cropping, Turn Tensor, Norma
train_transform = transforms.Compose([transforms.RandomResizedCrop(224),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.485, 0.456, 0.406]
                                     ])

# Test Set Image Preprocessing - RCTN: Scaling, Cropping, Turn Tensor, Normalisa
test_transform = transforms.Compose([transforms.Resize(256),
                                     transforms.CenterCrop(224),
                                     transforms.ToTensor(),
                                     transforms.Normalize(
                                         mean=[0.485, 0.456, 0.406],
                                         std=[0.229, 0.224, 0.225])
                                    ])
```

In [37]:

```
# Dataset folder path
dataset_dir = '/home/featurize/data'
```

In [ ]:

```
```

In [70]:

```
train_path = os.path.join(dataset_dir, 'train')
test_path = os.path.join(dataset_dir, 'val')
print('Training_set_path', train_path)
print('Testing_set_path', test_path)
```

```
Training_set_path /home/featurize/data/train
Testing_set_path /home/featurize/data/val
```

In [79]:

```
from torchvision import datasets

# Load training set
train_dataset = datasets.ImageFolder(train_path, train_transform)

# Load Test Set
test_dataset = datasets.ImageFolder(test_path, test_transform)
```

In [71]:

```
print('Number of images in the training set', len(train_dataset))
print('Number of categories', len(train_dataset.classes))
print('Name of each category', train_dataset.classes)
```

```
Number of images in the training set 3649
Number of categories 25
Name of each category ['CherryTomatoes', 'Mangosteen', 'MomordicaCharantia', 'Nav
elOrange', 'Sandsugaroranges', 'apple', 'banana', 'carrot', 'cherries', 'cucumbe
r', 'durian', 'grape', 'hamimelon', 'kiwi', 'lemon', 'lichee', 'longan', 'mango',
'pear', 'pineapple', 'pitaya', 'pomegranate', 'strawberry', 'tomato', 'watermelo
n']
```

In [72]:

```
print('Number of test set images', len(test_dataset))
print('Number of categories', len(test_dataset.classes))
print('Name of each category', test_dataset.classes)
```

```
Number of test set images 898
Number of categories 25
Name of each category ['Cherrytomatoes', 'Mangosteen', 'MomordicaCharantia', 'Nav
elOrange', 'Sandsugaroranges', 'apple', 'banana', 'carrot', 'cherries', 'cucumbe
r', 'durian', 'grape', 'hamimelon', 'kiwi', 'lemon', 'lichee', 'longan', 'mango',
'pear', 'pineapple', 'pitaya', 'pomegranate', 'strawberry', 'tomato', 'watermelo
n']
```

In [73]:

```
# Name of each category
class_names = train_dataset.classes
n_class = len(class_names)
```

In [74]:

```
class_names
```

Out[74]:

```
['CherryTomatoes',
 'Mangosteen',
 'MomordicaCharantia',
 'NavelOrange',
 'Sandsugaroranges',
 'apple',
 'banana',
 'carrot',
 'cherries',
 'cucumber',
 'durian',
 'grape',
 'hamimelon',
 'kiwi',
 'lemon',
 'lichee',
 'longan',
 'mango',
 'pear',
 'pineapple',
 'pitaya',
 'pomegranate',
 'strawberry',
 'tomato',
 'watermelon']
```

In [75]:

```
# Mapping relationship: category to index number
train_dataset.class_to_idx
```

Out[75]:
```
{'CherryTomatoes': 0,
 'Mangosteen': 1,
 'MomordicaCharantia': 2,
 'NavelOrange': 3,
 'Sandsugaroranges': 4,
 'apple': 5,
 'banana': 6,
 'carrot': 7,
 'cherries': 8,
 'cucumber': 9,
 'durian': 10,
 'grape': 11,
 'hamimelon': 12,
 'kiwi': 13,
 'lemon': 14,
 'lichee': 15,
 'longan': 16,
 'mango': 17,
 'pear': 18,
 'pineapple': 19,
 'pitaya': 20,
 'pomegranate': 21,
 'strawberry': 22,
 'tomato': 23,
 'watermelon': 24}
```

In [76]:
```
# Mapping relationship: index number to category
idx_to_labels = {y:x for x,y in train_dataset.class_to_idx.items()}
```

In [77]:
```
idx_to_labels
```

Out[77]:
```
{0: 'CherryTomatoes',
 1: 'Mangosteen',
 2: 'MomordicaCharantia',
 3: 'NavelOrange',
 4: 'Sandsugaroranges',
 5: 'apple',
 6: 'banana',
 7: 'carrot',
 8: 'cherries',
 9: 'cucumber',
 10: 'durian',
 11: 'grape',
 12: 'hamimelon',
 13: 'kiwi',
 14: 'lemon',
 15: 'lichee',
 16: 'longan',
 17: 'mango',
 18: 'pear',
 19: 'pineapple',
 20: 'pitaya',
 21: 'pomegranate',
 22: 'strawberry',
 23: 'tomato',
 24: 'watermelon'}
```

In [78]:
```
# Save as local npy file
np.save('idx_to_labels.npy', idx_to_labels)
np.save('labels_to_idx.npy', train_dataset.class_to_idx)
```

Define the data loader DataLoader

In [52]:
```
from torch.utils.data import DataLoader
```

In [53]:
```
BATCH_SIZE = 32

# Data loader for the training set
train_loader = DataLoader(train_dataset,
                          batch_size=BATCH_SIZE,
                          shuffle=True,
                          num_workers=4
                         )

# Data Loader for Test Sets
test_loader = DataLoader(test_dataset,
                         batch_size=BATCH_SIZE,
                         shuffle=False,
                         num_workers=4
                        )
```

View images and annotations for a batch

In [54]:
```
# DataLoader is a python generator that returns a batch of data per call.
images, labels = next(iter(train_loader))
```

In [55]:
```
images.shape
```

Out[55]:
```
torch.Size([32, 3, 224, 224])
```

In [56]:
```
labels
```

Out[56]:
```
tensor([16, 16, 10,  1, 24,  3,  4,  4, 17, 23,  3,  3, 15,  8, 19, 12,  1, 10,
        19,  4, 18,  3, 18,  4,  5, 16, 24,  0,  8, 20, 24, 11])
```

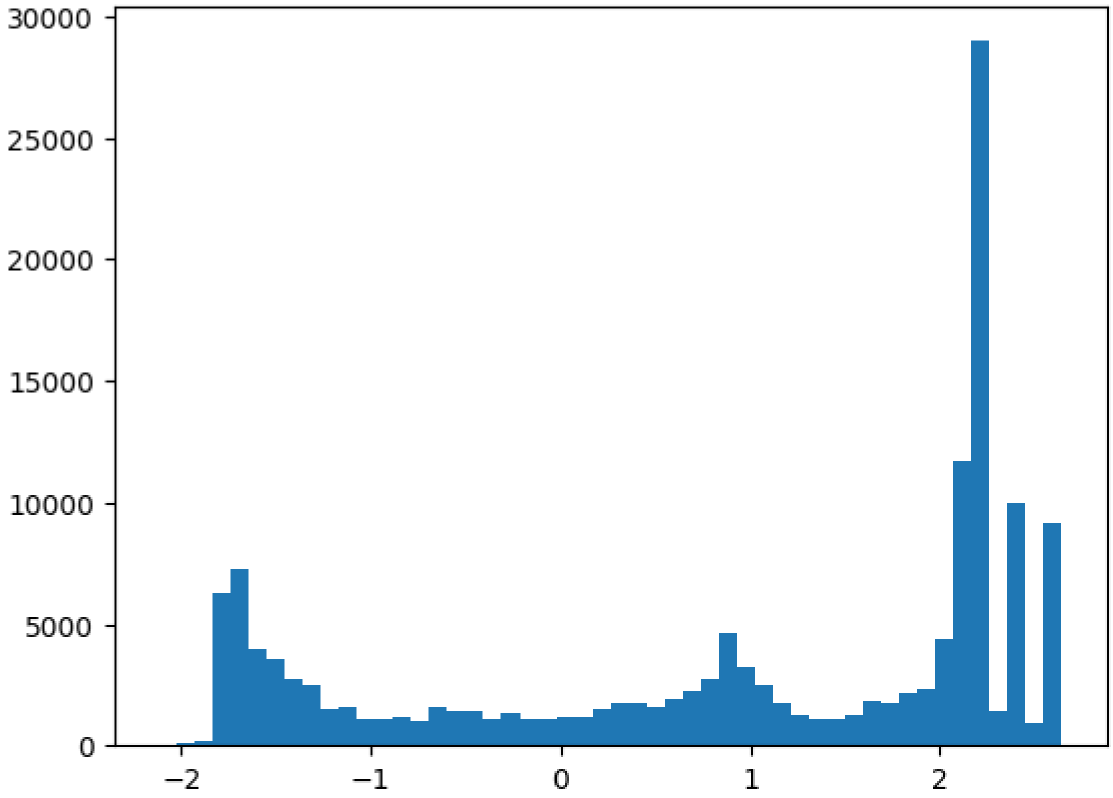
Visualising a batch of images and annotations

In [57]:
```
# Converting the Tensor tensor in a dataset to numpy's array data type
images = images.numpy()
```

In [58]:
```
images[5].shape
```

Out[58]:
```
(3, 224, 224)
```

In [59]:
```
plt.hist(images[5].flatten(), bins=50)
plt.show()
```



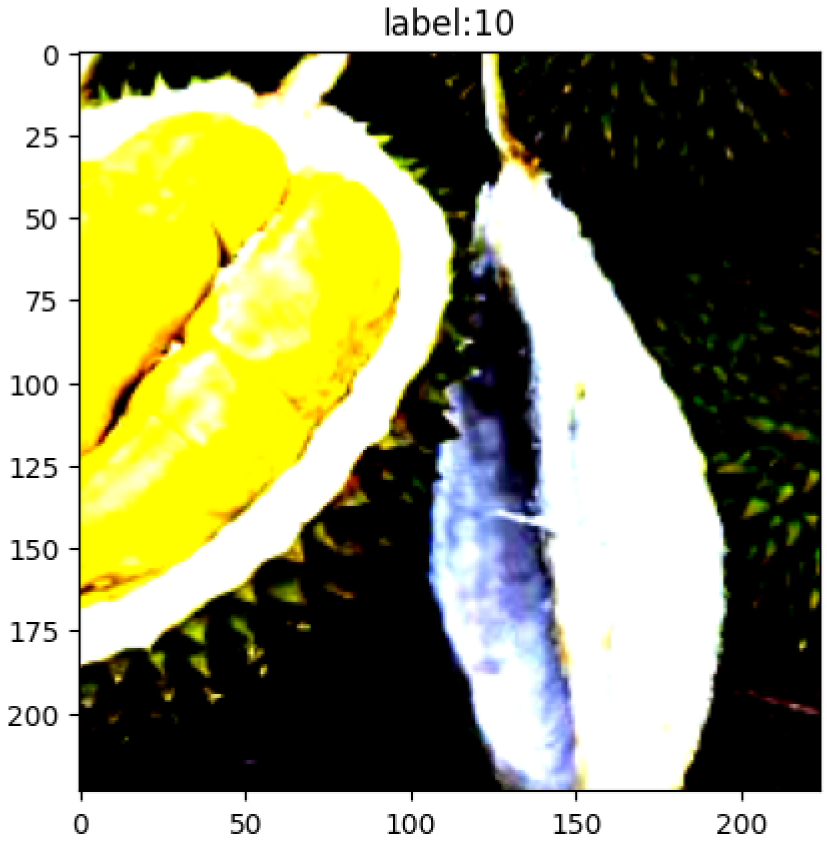
In [60]:

```
# Preprocessed images in batch
idx = 2
plt.imshow(images[idx].transpose((1,2,0))) # 转为(224, 224, 3)
plt.title('label:'+str(labels[idx].item()))
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for float
s or [0..255] for integers).
```

Out[60]:

```
Text(0.5, 1.0, 'label:10')
```



In [61]:

```
label = labels[idx].item()
```

In [62]:

```
label
```

Out[62]:

```
10
```

In [63]:

```
pred_classname = idx_to_labels[label]
```

In [64]:

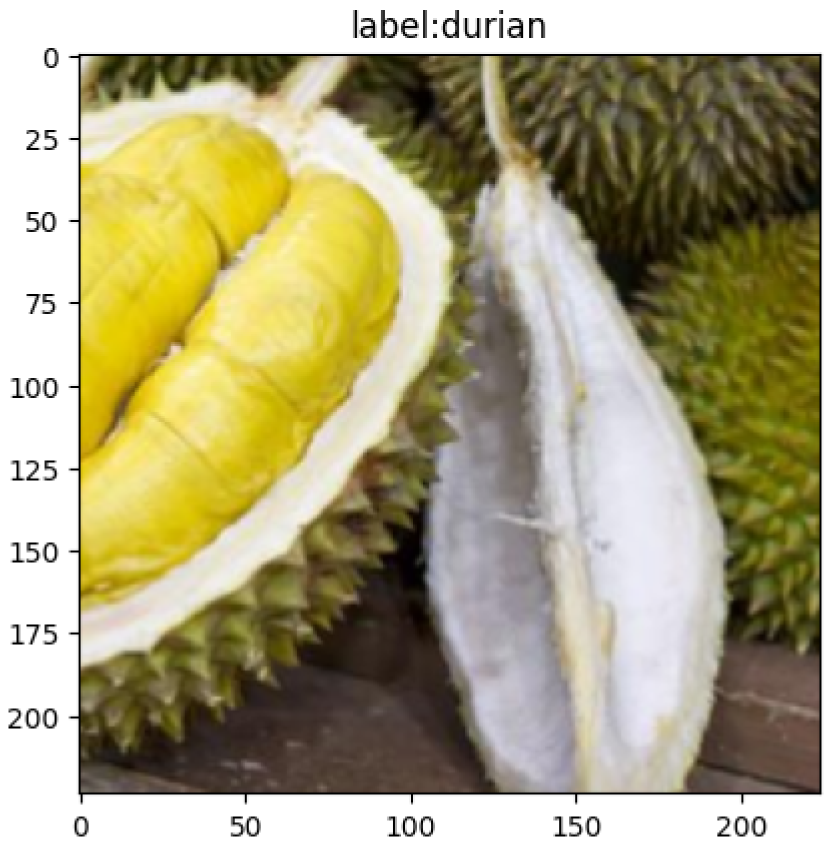
```
pred_classname
```

Out[64]:

```
'durian'
```

In [65]:

```
# original image
idx = 2
mean = np.array([0.485, 0.456, 0.406])
std = np.array([0.229, 0.224, 0.225])
plt.imshow(np.clip(images[idx].transpose((1,2,0)) * std + mean, 0, 1))
plt.title('label:'+ pred_classname)
plt.show()
```



Toolkit to be used for importing training

In [66]:

```
from torchvision import models
import torch.optim as optim
```

Randomly initialise all weights of the model and train all layers from scratch

In [68]:

```
model = models.resnet18(pretrained=False) # Load only the model structure, not t

model.fc = nn.Linear(model.fc.in_features, n_class)

optimizer = optim.Adam(model.parameters())
```

Training configuration

In [111…

```
model = model.to(device)

# Cross Entropy Loss Function
criterion = nn.CrossEntropyLoss()

# Training rounds Epoch
EPOCHS = 20
```

In [81]:

```
# Get a batch of data and annotations
images, labels = next(iter(train_loader))
images = images.to(device)
labels = labels.to(device)
```

In [82]:

```
# Input model to perform forward prediction
outputs = model(images)
```

In [83]:

```
# Get the predicted category logit scores for all images in the current batch
outputs.shape
```

Out[83]:

```
torch.Size([32, 25])
```

In [84]:

```
# From logit, calculate the average cross-entropy loss function for each sample
loss = criterion(outputs, labels)
```

In [85]:

```
optimizer.zero_grad() # Clearing the gradient
loss.backward() # backward propagation
optimizer.step() # Optimisation Updates
```

In [86]:

```
# Get the prediction categories for all images in the current batch
_, preds = torch.max(outputs, 1)
```

In [87]:

```
preds
```

Out[87]:

```
tensor([13, 14, 19,  3, 14,  9,  3, 19, 10, 14, 19, 19, 14, 13, 12, 21,  3, 19,
        15, 14, 14, 19,  3, 14,  3, 15, 19, 13,  3, 13, 14,  3],
       device='cuda:0')
```

In [88]:

```
labels
```

Out[88]:

```
tensor([23, 15, 10, 24, 15,  3,  9, 15, 17,  0,  5,  1, 24,  0, 21, 14,  6, 13,
        24, 19,  0, 10, 10,  0, 19, 10, 23,  2, 12, 22,  1,  2],
       device='cuda:0')
```

Run the full training

```
In [94]: # Iterate through each EPOCH
         for epoch in tqdm(range(EPOCHS)):

             model.train()

             for images, labels in train_loader:  # Get a batch of the training set with
                 images = images.to(device)
                 labels = labels.to(device)

                 outputs = model(images)           # Forward Prediction, get the predicti
                 loss = criterion(outputs, labels) # Compare the predictions with the ann

                 optimizer.zero_grad()
                 loss.backward()                   # Loss function back propagation of ne
                 optimizer.step()                  # Optimisation to update neural networ
```

```
100%|██████████| 20/20 [01:46<00:00,  5.33s/it]
```

Initial testing on a test set

```
In [95]: model.eval()
         with torch.no_grad():
             correct = 0
             total = 0
             for images, labels in tqdm(test_loader): # Get a batch of the test set with
                 images = images.to(device)
                 labels = labels.to(device)
                 outputs = model(images)              # Forward prediction, get the predi
                 _, preds = torch.max(outputs, 1)     # Obtain the category corresponding
                 total += labels.size(0)
                 correct += (preds == labels).sum()   # Number of correct samples predict

             print('The accuracy on the test set is {:.3f} %'.format(100 * correct / tota
```

```
100%|██████████| 29/29 [00:01<00:00, 17.24it/s]
The accuracy on the test set is 62.695 %
```

```
In [114… torch.save(model, 'checkpoint/fruit25_pytorch_xintian.pth')
```