In [1]:
```
import os
import random
import numpy as np
import pandas as pd
from tqdm import tqdm
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import random_split
from torch.utils.data import DataLoader, Dataset, Subset
from torch.utils.data import random_split, SubsetRandomSampler
from torchvision import datasets, transforms, models
from torchvision.datasets import ImageFolder
from torchvision.transforms import ToTensor
from torchvision.utils import make_grid
from pytorch_lightning import LightningModule
from pytorch_lightning import Trainer
import pytorch_lightning as pl
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from PIL import Image
```

```
C:\Users\anjal\New folder\Lib\site-packages\transformers\utils\generic.py:260: Use
rWarning: torch.utils._pytree._register_pytree_node is deprecated. Please use torc
h.utils._pytree.register_pytree_node instead.
  torch.utils._pytree._register_pytree_node(
```

In [2]:
```
transform=transforms.Compose([
        transforms.RandomRotation(10),      # rotate +/- 10 degrees
        transforms.RandomHorizontalFlip(),  # reverse 50% of images
        transforms.Resize(224),             # resize shortest side to 224 pixels
        transforms.CenterCrop(224),         # crop longest side to 224 pixels at ce
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
])
```

In [3]:
```
dataset0=datasets.ImageFolder(root="C:\Medicinal Leaf dataset",transform=None)

class_names=dataset0.classes
print(class_names)
print(len(class_names))
```

```
['Aloevera', 'Amla', 'Amruthaballi', 'Arali', 'Astma_weed', 'Badipala', 'Balloon_V
ine', 'Bamboo', 'Beans', 'Betel', 'Bhrami', 'Bringaraja', 'Caricature', 'Castor',
'Catharanthus', 'Chakte', 'Chilly', 'Citron lime (herelikai)', 'Coffee', 'Common r
ue(naagdalli)', 'Coriender', 'Curry', 'Doddpathre', 'Drumstick', 'Ekka', 'Eucalypt
us', 'Ganigale', 'Ganike', 'Gasagase', 'Ginger', 'Globe Amarnath', 'Guava', 'Henn
a', 'Hibiscus', 'Honge', 'Insulin', 'Jackfruit', 'Jasmine', 'Kambajala', 'Kasambru
ga', 'Kohlrabi', 'Lantana', 'Lemon', 'Lemongrass', 'Malabar_Nut', 'Malabar_Spinac
h', 'Mango', 'Marigold', 'Mint', 'Neem', 'Nelavembu', 'Nerale', 'Nooni', 'Onion',
'Padri', 'Palak(Spinach)', 'Papaya', 'Parijatha', 'Pea', 'Pepper', 'Pomoegranate',
'Pumpkin', 'Raddish', 'Rose', 'Sampige', 'Sapota', 'Seethaashoka', 'Seethapala',
'Spinach1', 'Tamarind', 'Taro', 'Tecoma', 'Thumbe', 'Tomato', 'Tulsi', 'Turmeric',
'ashoka', 'camphor', 'kamakasturi', 'kepala']
80
```

In [4]:
```
class DataModule(pl.LightningDataModule):

    def __init__(self, transform=transform, batch_size=32):
```

```
        super().__init__()
        self.root_dir = "C:\Medicinal Leaf dataset"
        self.transform = transform
        self.batch_size = batch_size

    def setup(self, stage=None):
        dataset = datasets.ImageFolder(root=self.root_dir, transform=self.transform
        n_data = len(dataset)
        n_train = int(0.8 * n_data)
        n_test = n_data - n_train

        train_dataset, test_dataset = torch.utils.data.random_split(dataset, [n_tra

        self.train_dataset = DataLoader(train_dataset, batch_size=self.batch_size,
        self.test_dataset = DataLoader(test_dataset, batch_size=self.batch_size)

    def train_dataloader(self):
        return self.train_dataset

    def test_dataloader(self):
        return self.test_dataset
```

In [5]:

```
class ConvolutionalNetwork(LightningModule):

    def __init__(self):
        super(ConvolutionalNetwork, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 3, 1)
        self.conv2 = nn.Conv2d(6, 16, 3, 1)
        self.fc1 = nn.Linear(16 * 54 * 54, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 20)
        self.fc4 = nn.Linear(20, len(class_names))

    def forward(self, X):
        X = F.relu(self.conv1(X))
        X = F.max_pool2d(X, 2, 2)
        X = F.relu(self.conv2(X))
        X = F.max_pool2d(X, 2, 2)
        X = X.view(-1, 16 * 54 * 54)
        X = F.relu(self.fc1(X))
        X = F.relu(self.fc2(X))
        X = F.relu(self.fc3(X))
        X = self.fc4(X)
        return F.log_softmax(X, dim=1)

    def configure_optimizers(self):
        optimizer = torch.optim.Adam(self.parameters(), lr=0.001)
        return optimizer

    def training_step(self, train_batch, batch_idx):
        X, y = train_batch
        y_hat = self(X)
        loss = F.cross_entropy(y_hat, y)
        pred = y_hat.argmax(dim=1, keepdim=True)
        acc = pred.eq(y.view_as(pred)).sum().item() / y.shape[0]
        self.log("train_loss", loss)
        self.log("train_acc", acc)
        return loss

    def validation_step(self, val_batch, batch_idx):
        X, y = val_batch
        y_hat = self(X)
        loss = F.cross_entropy(y_hat, y)
        pred = y_hat.argmax(dim=1, keepdim=True)
```

```
        acc = pred.eq(y.view_as(pred)).sum().item() / y.shape[0]
        self.log("val_loss", loss)
        self.log("val_acc", acc)

    def test_step(self, test_batch, batch_idx):
        X, y = test_batch
        y_hat = self(X)
        loss = F.cross_entropy(y_hat, y)
        pred = y_hat.argmax(dim=1, keepdim=True)
        acc = pred.eq(y.view_as(pred)).sum().item() / y.shape[0]
        self.log("test_loss", loss)
        self.log("test_acc", acc)
```

In [6]:

```
if __name__ == '__main__':
    datamodule = DataModule()
    datamodule.setup()
    model = ConvolutionalNetwork()
    trainer = pl.Trainer(max_epochs=1)
    trainer.fit(model, datamodule)
    datamodule.setup(stage='test')
    test_loader = datamodule.test_dataloader()
    trainer.test(dataloaders=test_loader)
```

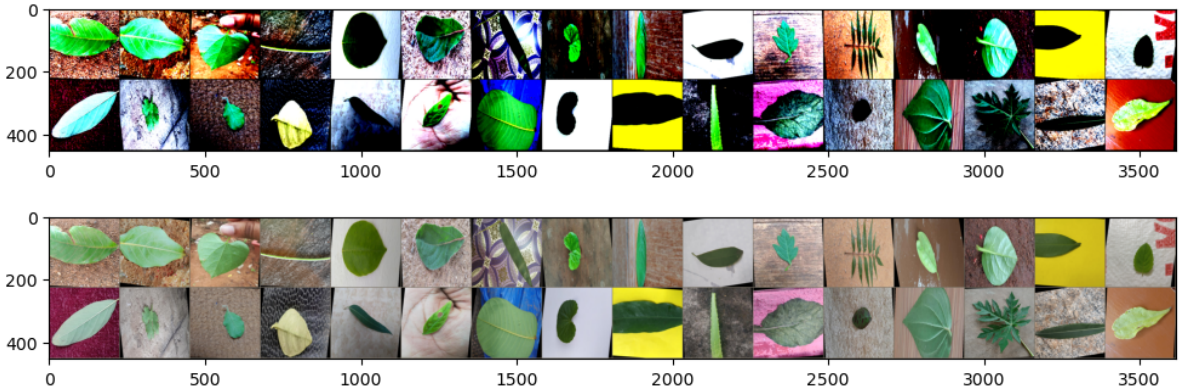
```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
C:\Users\anjal\New folder\Lib\site-packages\pytorch_lightning\trainer\connectors\l
ogger_connector\logger_connector.py:75: Starting from v1.9.0, `tensorboardX` has b
een removed as a dependency of the `pytorch_lightning` package, due to potential c
onflicts with other packages in the ML ecosystem. For this reason, `logger=True` w
ill use `CSVLogger` as the default logger, unless the `tensorboard` or `tensorboar
dX` packages are found. Please `pip install lightning[extra]` or one of them to en
able TensorBoard support by default
C:\Users\anjal\New folder\Lib\site-packages\pytorch_lightning\trainer\configuratio
n_validator.py:74: You defined a `validation_step` but have no `val_dataloader`. S
kipping val loop.

  | Name  | Type   | Params
---------------------------------
0 | conv1 | Conv2d | 168
1 | conv2 | Conv2d | 880
2 | fc1   | Linear | 5.6 M
3 | fc2   | Linear | 10.2 K
4 | fc3   | Linear | 1.7 K
5 | fc4   | Linear | 1.7 K
---------------------------------
5.6 M     Trainable params
0         Non-trainable params
5.6 M     Total params
22.454    Total estimated model params size (MB)
C:\Users\anjal\New folder\Lib\site-packages\pytorch_lightning\trainer\connectors\d
ata_connector.py:441: The 'train_dataloader' does not have many workers which may
be a bottleneck. Consider increasing the value of the `num_workers` argument` to `
num_workers=7` in the `DataLoader` to improve performance.
Training: |
| 0/? [00:00<…
```

```
`Trainer.fit` stopped: `max_epochs=1` reached.
C:\Users\anjal\New folder\Lib\site-packages\pytorch_lightning\trainer\connectors\checkpoint_connector.py:145: `.test(ckpt_path=None)` was called without a model. The best model of the previous `fit` call will be used. You can pass `.test(ckpt_path='best')` to use the best model or `.test(ckpt_path='last')` to use the last model. If you pass a value, this warning will be silenced.
Restoring states from the checkpoint path at C:\Users\anjal\lightning_logs\version_5\checkpoints\epoch=0-step=173.ckpt
Loaded model weights from the checkpoint at C:\Users\anjal\lightning_logs\version_5\checkpoints\epoch=0-step=173.ckpt
C:\Users\anjal\New folder\Lib\site-packages\pytorch_lightning\trainer\connectors\data_connector.py:441: The 'test_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of the `num_workers` argument` to `num_workers=7` in the `DataLoader` to improve performance.
Testing: |
| 0/? [00:00<…
──────────────────────────────────────────────────────────────────────────────
       Test metric             DataLoader 0
──────────────────────────────────────────────────────────────────────────────
        test_acc            0.09275362640619278
        test_loss           3.7952725887298584
──────────────────────────────────────────────────────────────────────────────
```

In [7]:
```
for images, labels in datamodule.train_dataloader():
    break
im=make_grid(images,nrow=16)

plt.figure(figsize=(12,12))
plt.imshow(np.transpose(im.numpy(),(1,2,0)))

inv_normalize=transforms.Normalize(mean=[-0.485/0.229,-0.456/0.224,-0.406/0.225],
                                   std=[1/0.229,1/0.224,1/0.225])
im=inv_normalize(im)

plt.figure(figsize=(12,12))
plt.imshow(np.transpose(im.numpy(),(1,2,0)))
```

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

Out[7]:
```
<matplotlib.image.AxesImage at 0x240eccb5b90>
```



In [8]:
```
device = torch.device("cpu")   #"cuda:0"

model.eval()
y_true=[]
y_pred=[]
with torch.no_grad():
```

```
    for test_data in datamodule.test_dataloader():
        test_images, test_labels = test_data[0].to(device), test_data[1].to(device)
        pred = model(test_images).argmax(dim=1)
        for i in range(len(pred)):
            y_true.append(test_labels[i].item())
            y_pred.append(pred[i].item())

print(classification_report(y_true,y_pred,target_names=class_names,digits=4))
```

	precision	recall	f1-score	support
Aloevera	0.0625	0.1667	0.0909	24
Amla	0.0000	0.0000	0.0000	14
Amruthaballi	0.0000	0.0000	0.0000	18
Arali	0.0000	0.0000	0.0000	22
Astma_weed	0.0000	0.0000	0.0000	21
Badipala	0.0000	0.0000	0.0000	13
Balloon_Vine	0.0000	0.0000	0.0000	13
Bamboo	0.1250	0.0435	0.0645	23
Beans	0.1852	0.2778	0.2222	18
Betel	0.0851	0.6400	0.1502	25
Bhrami	0.0000	0.0000	0.0000	17
Bringaraja	1.0000	0.1250	0.2222	16
Caricature	0.0000	0.0000	0.0000	17
Castor	0.0000	0.0000	0.0000	21
Catharanthus	0.1356	0.2963	0.1860	27
Chakte	0.0000	0.0000	0.0000	15
Chilly	0.0000	0.0000	0.0000	17
Citron lime (herelikai)	0.0000	0.0000	0.0000	25
Coffee	0.0000	0.0000	0.0000	17
Common rue(naagdalli)	0.0000	0.0000	0.0000	12
Coriender	0.0000	0.0000	0.0000	27
Curry	0.0690	0.0645	0.0667	31
Doddpathre	0.0800	0.3478	0.1301	23
Drumstick	0.0000	0.0000	0.0000	9
Ekka	0.0000	0.0000	0.0000	19
Eucalyptus	0.0000	0.0000	0.0000	18
Ganigale	0.0000	0.0000	0.0000	8
Ganike	0.0000	0.0000	0.0000	14
Gasagase	0.0000	0.0000	0.0000	20
Ginger	0.0000	0.0000	0.0000	19
Globe Amarnath	0.2500	0.2667	0.2581	15
Guava	0.0000	0.0000	0.0000	24
Henna	0.0000	0.0000	0.0000	13
Hibiscus	0.3333	0.0476	0.0833	21
Honge	0.0270	0.1111	0.0435	18
Insulin	0.0000	0.0000	0.0000	14
Jackfruit	0.0000	0.0000	0.0000	18
Jasmine	0.0000	0.0000	0.0000	9
Kambajala	0.2500	0.1250	0.1667	16
Kasambruga	0.0000	0.0000	0.0000	12
Kohlrabi	0.0000	0.0000	0.0000	12
Lantana	0.0000	0.0000	0.0000	15
Lemon	0.0000	0.0000	0.0000	22
Lemongrass	0.0000	0.0000	0.0000	3
Malabar_Nut	0.0000	0.0000	0.0000	9
Malabar_Spinach	0.0000	0.0000	0.0000	23
Mango	0.0000	0.0000	0.0000	30
Marigold	0.0000	0.0000	0.0000	16
Mint	0.0000	0.0000	0.0000	24
Neem	0.0823	0.4483	0.1390	29
Nelavembu	0.0000	0.0000	0.0000	15
Nerale	0.0000	0.0000	0.0000	13
Nooni	0.0000	0.0000	0.0000	18
Onion	1.0000	0.0588	0.1111	17
Padri	0.0278	0.2308	0.0496	13
Palak(Spinach)	0.1304	0.3214	0.1856	28
Papaya	0.0667	0.0690	0.0678	29
Parijatha	0.0000	0.0000	0.0000	12
Pea	0.0000	0.0000	0.0000	5
Pepper	0.0000	0.0000	0.0000	2
Pomoegranate	0.0000	0.0000	0.0000	11
Pumpkin	0.0000	0.0000	0.0000	21

```
     Raddish     0.1429    0.1667    0.1538         6
        Rose     0.0000    0.0000    0.0000        20
     Sampige     0.0000    0.0000    0.0000        11
      Sapota     0.0000    0.0000    0.0000         8
Seethaashoka     0.0000    0.0000    0.0000        15
  Seethapala     0.3333    0.0769    0.1250        26
    Spinach1     0.0000    0.0000    0.0000         8
    Tamarind     0.1642    0.9778    0.2812        45
        Taro     0.0000    0.0000    0.0000        13
      Tecoma     0.0000    0.0000    0.0000        12
      Thumbe     0.0000    0.0000    0.0000        16
      Tomato     0.0000    0.0000    0.0000        11
       Tulsi     0.0000    0.0000    0.0000        34
    Turmeric     0.0000    0.0000    0.0000         8
      ashoka     0.0000    0.0000    0.0000        12
     camphor     0.0000    0.0000    0.0000        15
 kamakasturi     0.0000    0.0000    0.0000        11
      kepala     0.0000    0.0000    0.0000        19

    accuracy                         0.0942      1380
   macro avg     0.0569    0.0608    0.0350      1380
weighted avg     0.0659    0.0942    0.0469      1380
```

```
C:\Users\anjal\New folder\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\anjal\New folder\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\anjal\New folder\Lib\site-packages\sklearn\metrics\_classification.py:146
9: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]: