In [28]:

```
## utils
import torch

## data
import torchvision
from torch.utils.data import DataLoader, random_split
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt

## model
import torch.nn as nn
import torchvision.models as models

## training
import pytorch_lightning as pl
```

In [4]:

```
# reproducibility
torch.manual_seed(0)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
```

In [5]:

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

1. Data Preparation

In [6]:

```
root = "./dataset/"
```

In [17]:

```
# transforms
data_transforms = torchvision.transforms.Compose([
        torchvision.transforms.RandomResizedCrop(224),
        torchvision.transforms.RandomHorizontalFlip(),
        torchvision.transforms.RandomRotation(20),
        torchvision.transforms.ColorJitter(hue=.05, saturation=.05),
        torchvision.transforms.ToTensor(),
#         transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
```

In [18]:

```
# create the dataset
dataset = datasets.ImageFolder(
    root=root,
    transform=data_transforms
)
```

In [32]:

```
# split the data into train and valid set
val_size = int(0.05*len(dataset))
train_size = len(dataset) - val_size

train, valid = random_split(dataset=dataset, lengths=[train_size, val_size])
```

In [26]:

```
```

Out[26]:

```
27558
```

In [19]:

```
classes = dataset.classes
```

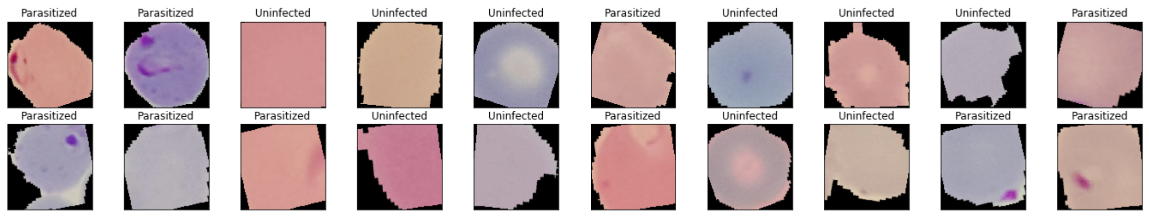
In [20]:

```
loader = DataLoader(dataset=dataset, batch_size=32, shuffle=True)
```

In [23]:

```
dataiter = iter(loader)
images, labels = dataiter.next()
images = images.numpy() # convert images to numpy for display

# plot the images in the batch, along with the corresponding labels
fig = plt.figure(figsize=(24, 4))
for idx in np.arange(20):
    ax = fig.add_subplot(2, 20/2, idx+1, xticks=[], yticks=[])
    plt.imshow(np.transpose(images[idx], (1, 2, 0)))
#     plt.title("abc")
    ax.set_title(classes[labels[idx]])
# plt.legend()
plt.show()
```



In [ ]:

```
loader = DataLoader
```

In [24]:

```
datasets.ImageFolder?
```

```
Init signature:
datasets.ImageFolder(
    root,
    transform=None,
    target_transform=None,
    loader=<function default_loader at 0x7f08971158c0>,
    is_valid_file=None,
)
Docstring:
A generic data loader where the images are arranged in this way: ::

    root/dog/xxx.png
    root/dog/xxy.png
    root/dog/xxz.png

    root/cat/123.png
    root/cat/nsdf3.png
    root/cat/asd932_.png

Args:
    root (string): Root directory path.
    transform (callable, optional): A function/transform that  takes
in an PIL image
        and returns a transformed version. E.g, ``transforms.RandomC
rop``
    target_transform (callable, optional): A function/transform that
takes in the
        target and transforms it.
    loader (callable, optional): A function to load an image given i
ts path.
    is_valid_file (callable, optional): A function that takes path o
f an Image file
        and check if the file is a valid file (used to check of corr
upt files)

 Attributes:
    classes (list): List of the class names.
    class_to_idx (dict): Dict with items (class_name, class_index).
    imgs (list): List of (image path, class_index) tuples
File:           ~/miniconda3/lib/python3.7/site-packages/torchvisio
n/datasets/folder.py
Type:           type
Subclasses:     ImageNet
```

In [ ]: