Image Classification using ResNet34

Image Dataset Setup

- Convert the TIFF file formatted images to JPG and zip them into a folder
- Extract the images into 38 different folders each representing a class label

Initialization

Importing required libraries

```
import os
%matplotlib inline

from fastai.vision import *
from fastai.metrics import error_rate
```

Setting the batch size and path for images

```
bs = 64  #batch size
sz = 224 #image size
PATH = 'C:/Users/sravan/OneDrive/Documents/CNN_Images/CNN_Images/
```

- PATH is the path containing all the class folders
- Convert the folder names into class label names from 1-38

```
i = 1
for file in os.listdir(PATH):
    os.rename(os.path.join(PATH,file),
              os.path.join(PATH,str(i)))
    i = int(int(i)+1)
```

- Retrieve the image classes from the foldes

```
classes = []
for d in os.listdir(PATH):
    if os.path.isdir(os.path.join(PATH, d)) and not d.startswith('.'):
        classes.append(d)
print ("There are ", len(classes), "classes:\n", classes)
```

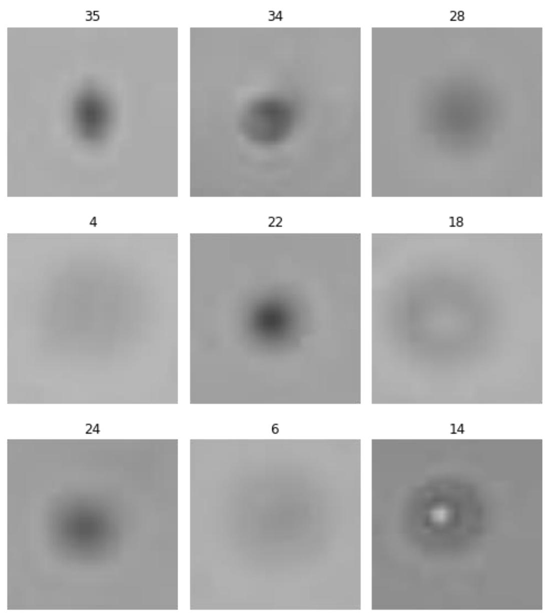
```
There are  38 classes:
 ['1', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '2', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '3', '30', '31', '32', '33', '34', '35', '36', '37', '38', '4', '5', '6', '7', '8', '9']
```

Creating and training the classifier

- Let's create our training and validation sets, 80% of the dataset will be used or training and 20% for validation
- Normalize the images to ensure that each pixel has a similar data distribution and visualizing some images from different classes

```
data  = ImageDataBunch.from_folder(PATH, ds_tfms=get_transforms(), size=sz, bs=bs, valid_pct=0.2).normalize(imagenet_stats)

data.show_batch(rows=3, figsize=(7,8))
```



- Build the Deep Convolutional Neural Network (CNN) with using ResNet34 as the model architecture

```
learn = cnn_learner(data, models.resnet34, metrics=accuracy)
```

- Training the model on the images

```
learn.fit_one_cycle(1,max_lr=slice(1e-3,1e-2))
```

epoch	train_loss	valid_loss	accuracy	time
0	1.299533	0.829972	0.662125	1:45:59

- As the run time is very large, only one epoch was run on the dataset and achieved an accuracy of **66.2%**

Results Interpretation and Visualization

```
interp = ClassificationInterpretation.from_learner(learn)

interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```

- The diagonal elements in a confusion matrix represent the number of images for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier
- The confusion matrix below shows that most of the class labels were predicted correctly except few classes