# Fast ai chapter 5 video lesson 4 remaining

03 July 2021 22:36

While loading the dataset initially we don't know what's inside the dataset so for knowing this we do Path.BASE_PATH = path
Where the path is the untar_data returned path and we are setting the base path of colab instance to that path

To know what's inside the path we can do path.ls

Most of the functions in the fast ai that return the collection class named L where It's the advanced version of list which gives the #no_of_files & list of the files

(path/'images').ls() will return the modified L class

In this datset which we're loading the file names which are starting from capital letters are the dogs and smaller letter starting are the dogs
Further name contains the name of the breed
And furthermore the number contains the unique idenitfication for the files
For taking the advantage of this file structure we are going to use the regular expressions (re)

For this purpose we are going to use the
Re.findall(r'(.+)_\d+.jpg$',fname.name)
Then we are getting the file names

Another thing to do is getting the data using the datablock api
Pets = DataBlock(blocks = (ImageBlock, CategoryBlock),
Get_items = get_image_files,
Splitter = RandomSplitter(seed = 42),
Get_y = using_attr(RegexLabeller(r'(.+)_\d+.jpg$'),'name')
# here usingn attr takes the path object and just applies the function to the file name so that we don't have to write regex for whole path
Item_tfms = Resize(460),
Batch_tfms = aug_transfroms(size = 224,min_scale = 0.75))
Dls = pets.dataloaders(path/'images')
Here the aug_transform for doing the image augmentation random resize crop

Rotation zooming and the zooming something in the smaller since the 1st step the resizing is happening and all the images are in same size the second step can happen on the GPU
in fast ai they don't do it in the regular fashion they stores the changing values of the pixels and then they do it when the data is completely loaded
So that the image is presized effectively without any pixel detortion (less)
If we set the dls.show_batch(nrows, ncols, unique = True ) then here the dataloaders will show the images which are of same breed

When you call the dataloader.summary() it'll go through your dataloader and gives you summary of what your dataloaders contain at each time

For the most of the times the dataloaders will be necessary since the dataloaders are the blueprint of your model they will provide what are you gonna provide to the model

Most of the peoples say that the data should be cleaning before building the model but in fast ai we can do it without cleaning and then we can clean afterwards cause the model will give us the images on which it's most confused on

Here we can see that the loss function is choosen automatically by fast ai
Since we are working on the dogand cats breed classification problem we are going to use the CNN learner

We have seen the binary-cross-entropy function previously while classifying the dogs vs cats model for doing the same now we've to determine the

For checking the loss function we can check it as
Learn.loss_func

For the last time all we did was to make a function which will be differenciating how wrong the model went and give us the sigmoid of the model

For getting the model names which were into the vocab we can have it like
Dls.vocab *# it'll return us the vocab array
After that we can map the vocab to the label array

For getting the predictions for the data batch we have separated we can

Do learn.get_preds(dl = [(x,y)])
Preds[0]

To have the multiple predictions
We use softmax

It's the extension of the sigmoid functions

The softmax is defined as the e(x) / exp(x).sum(dim(acts, dim = 1 )

It works same as the sigmoid for the cases where we've binary classification problem otherwise it'll be different like

| | output | exp | softmax |
|---|---|---|---|
| teddy | 0.02 | 1.02 | 0.22 |
| grizzly | -2.49 | 0.08 | 0.02 |
| brown | 1.25 | 3.49 | 0.76 |
| | | 4.60 | 1.00 |

For doing the softmax function in practice we can just use the torch.softmax

in the mnist loss we have used torch.where(target==1 and some more things ) but we can't use that anymore since the outputs can be more than 1 and 0 since we are doing the classification with the various categories

We want to transform the result to be in between 0 and 1 not in the range of positive and the negative infinity

**Cross_entropy loss :**

**F.CrossEntropyLoss() : log softmax followed by nll_loss**

The understood concept

so the cross entropy loss gives the loss based on the probability which I've predicted if it's less then the loss will be high so like if I predicted something with probability 0.1000 then that should give me high loss but if I just take loss of 0.1000 then it'll give me -ve value * that will be quite high negative but I wanted to add them up since I want to minimize the loss and it will back propagate to the model and it'll in turn increase weights so the more my value close to 1

**Building the loss function in fast ai**

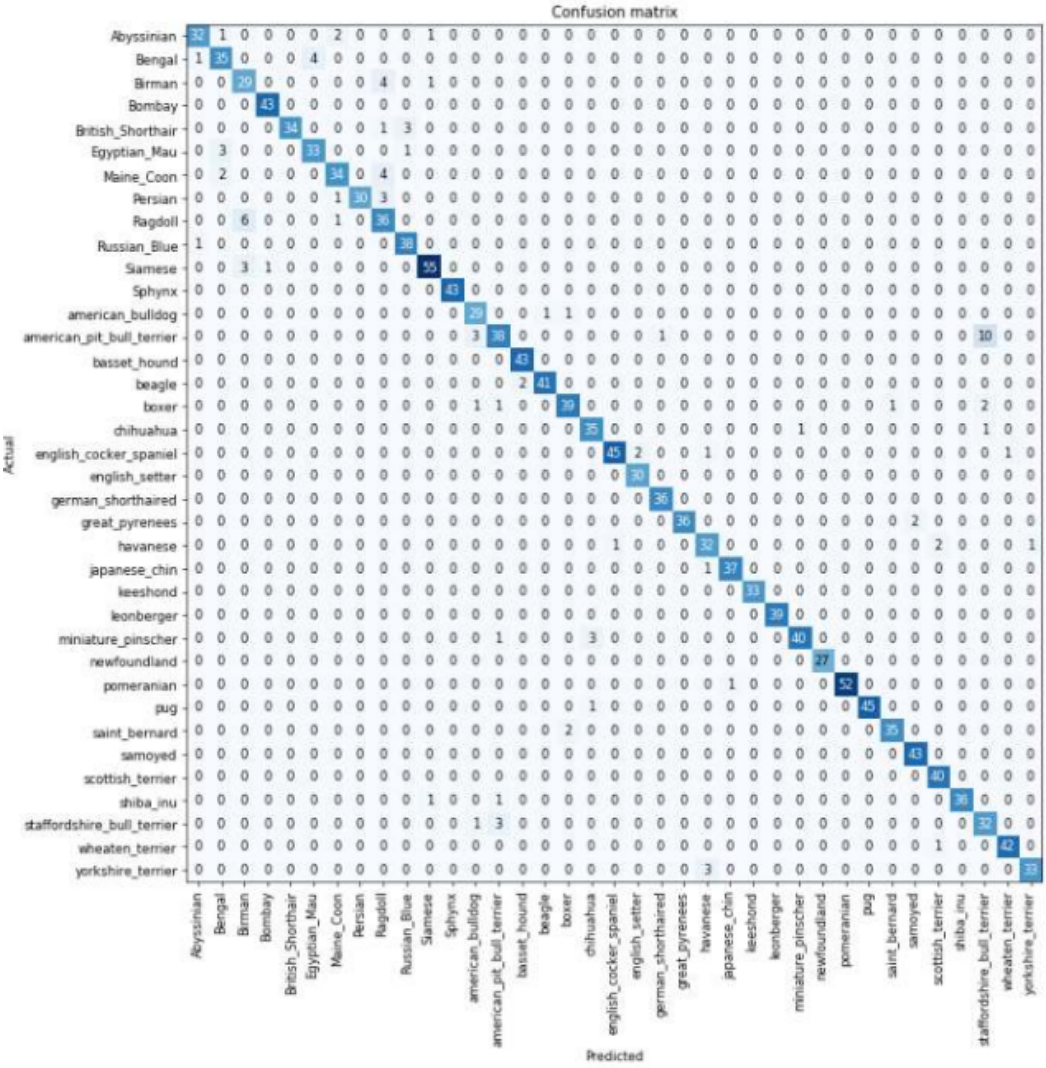
For taking NLL loss we have to do the F.nll_loss(sm_acts , targ, reduction = 'none')

But the log_softmax() and the nll_loss since the nll loss is not doing the log

Nn.crossentropy is the combination of the log_softmax() and nll_loss() function

Since the f.nll_loss requires the sm_acts and the target with the log taken we have to use the log_softmax for getting log of the predictions

**Visualizing the matrix**




**Another things which we can do with interp:**

For visualizing the confusion matrix in fast ai all we have to do is

Interp = ClassificationInterpretation.from_learner(learn)

Interp.plot_confusion_matrix(figsize=(12,12),dpi = dots per inch ( 60 )

Interp also having various type of measures in case the diagram becomes hard to find out what's in we can use the interp.most_confused(min_val=5)
Its giving us the model got confused on this val as this and in turn telling us the most confused states of the model

```
interp.most_confused(min_val=5)
```

```
[('american_pit_bull_terrier', 'staffordshire_bull_terrier', 10),
 ('Ragdoll', 'Birman', 6)]
```

**Improving the model:**

**Finding the learning rate :**

For model to perform ideally we should have ideal learning rate so that model will train well and able to find out patterns inside the data

Learner.lr_find(suggest_func= [valley,step and various parameters can be given here])