Fast ai lesson 6 book multicat chapter 6 other computer vision problems

In this we are trying to load the data from csv files with the help of pandas

Ther are many tutorials available online you can look into them for studying more about pandas

Constructing the data block :

Dataset : abstract id of the class anything which you can index or take the len of it

DataLoader : which provides the stream of the mini batches whhere each mini batch is a couple of the batch is a couple of independent variables and batch of dependent variable

From the dataloader api we can pass the batch size ,shuffle parameter , and the dataset / lists of the datasets which we wanted to combine

We can get the list of tuples using the python list(zip(b1, b2))

But for combining the dataset which are haiving like more of the columns which we wanted to combine we can do it as

List(zip(*b)) it'll combine all the columns of B

Datasets : object which have the training dataset and validation datset

DataLoaders : an objects which contains a training dataloader and validation dataloader

Often we don't have the labels sepearated for us for this purpose we have to find the labels using the file names * by slicing the filenames etc

If we want to do the number of transformations on the filenames of the images we can determine the transformations in functions and then pass them by using Dataset

Suppose we have the fun f(a) : return a + 'o'

For passing this functions to our datsets all we got to do is

Dss = Datasets(a, f,f) # notice here we are passing the list of the transformations which we want to do on the datases

If we pass the multipe functions then the datasets api does the multiple things which we are specifying inside functions

Here we are passing the lists of list so if we pass it like

Datasets(a, f],[f) then it'll give us the output with the 2 values in each tuple one with the value which is applied to the first tuple and the second will be value which get applied by the second function

So if we want to have both the transformatoins on the same data then we have to pass it in the single list

DataBlock : further we will discuss how to make a datablock

We can initilize the datablock using

Dblock = DataBlock()

We can pass the dataframe by using

Dsets = dblock.datasets(df)

it'll create the training and validation datasets for us it randomly splits with the 20 % of validation sets

X['fname']

We can pass the params as the lambda code or function which will give out the train data and the labels

Get_x = return s fname

Get_y = return the labeles

Dblock.dataset

Datablock api is also having the blocks blcock we cna get ias the ImageBlock and MultiCategoryBlock fro the multiple categories

Splitter argument take the param as function which is responsible for where the data needed to split

The function here we are using it's gonna written the indexes where the training data is valid and the indexes where trainin data is not valid

![def splitter(df): train = df . 'is valid']] .tolist()
valid = df . valid']] . tolist()
return train, valid
— MultiCategoryBlock) ,

dblock —
splitter—splitter,
get_x=get x,
dsets = dblock.datasets(df)
dsets.train[0]]

```
def splitter(df):
    train = df.index[~df['is_valid']].tolist()
    valid = df.index[df['is_valid']].tolist()
    return train,valid

dblock = DataBlock(blocks=(ImageBlock, MultiCategoryBlock),
                   splitter=splitter,
                   get_x=get_x,
                   get_y=get_y)

dsets = dblock.datasets(df)
dsets.train[0]

(PILImage mode=RGB size=500x333
```

Now we are using the correct train and validation set

learner = cnn_learner(dls,resnet18)
for getting the model as the function we have to use the learner.model()

changing parameters inside the functions :
for changing parameters suppose we have function fun already defined with say_this params and we wanted to change that parameter everytime we call it so for doing this we can use partial functions
f = partial(fun, say_this = 'this')
this will create the partial function of fun in var f
we can call this by using f('asdfkj')

for doing the same while passing function to matrix you can do it like
leran = cnn_learner(dls, resnet50 , metrics= partial(accuracy_multi,thresh = 0.2))
here the threshold is telling that to our model that even if the sigmoid function gives out the freq > 0.2 it shoudl consider it as true
it's in code lile(pred.sigmoid()> threshold) == target

learn.fine_tune(3, base_lr - 3e-3, freeze_epoch = 4)
here we are fine tuning for 3 epochs and fit_one_cycle for 4 epochs

learn.validate : is to see how does model performs if we did change some functional parameteres

for finding out varience happening with different threshold in model

```
xs = torch.linspace(0.05,0.95,29) accs = [accuracy_multi(preds, targs, thresh=i,
sigmoid=False) for i in xs] plt.plot(xs,accs);
```

while training the point model we shoud give teh range of points so that our model will not go beyond that point *while doing regression related task*

learn = cnn_learner(dls, resnett18, y_range= (-1,1))

most of the times in fast ai we don't have do explicitely tell fast ai to choose loss function it does that automatically for us
cross entropy loss for single label classification
BCEwithLogists loss for multilabel classification
* mse loss for regresssion