Fast ai lesson 1 : notes

23 June 2021 10:51

For viewing various documentation of the functions

!pip install nbdev
From nbdev.showdoc import *

# **for viewing the documentation**
Help(function/ method)

For loading the fast ai into colab notebook

```
!pip install fastai --upgrade
from fastai.vision.all import *
```

Getting file names from file_path

Path_img = get_file_names(path)

Getting the labels from filenames

ImageDataLoaders.from_path_re : get's the labels from the file names
Takes Parametersa as path, filenames , pattern (re)
Valid_pct (valid percentage)
Seed = for controlling randomlness
Bs = batch_size
Shuffle = for shuffling the data
Device += "gpu or cpu "
The compulsory parameters are path, filenames and patterns

Creating the simple cnn model with fast ai

Learn = cnn_learner(dls, resnet34, metrics = error_rate).to_fp_16
To_fp_sixteen : for mixed precision we are specifying the model to use floatpoint 16 predictions
So that the model will load faster with GPU access
Dls = data loaders
Resnet34 : model architecure
Metrics = error_rate, accuracy
Normalization and pretrained = normalization does the normalization of the data
Pretrained is the model which is already built and the data is loaded and normlized with the help of the data which pretrained model really wants

Viewing model layers ;

Model_name.model
Returns the total architecture of the model

Fitting the model :

Model_name.fit_one_cycle(no of epochs)
Fit_one_cyclles deciedes the learning rate from cosine annealing form

Saving the model :

Model_name.save('path')
Saves the model in .pth format
Another way

Save_model(file,model,opt,with_opt,pickle_protocol)
File : name of the file which you want to save your model in
Opt :
Pickle_protocol :for saving in pkl format

Laoading the model :

Load_model(file_path , model , opt , with_opt, device = None, strict = True)

Device : if passed nothing then loads of CPU otherwise gpu can be passesd

Strict : true : all file must exactly contain certain weights for every parameter key in model

If it's false : only key that are in the saved model are loaded in the model

To evaluate and see when the model is performing most of wrong predictions :

Interp = ClassificationInterpretation.from_learner(learn)

From_learner is to pass the model

Learn : the model which you want to pass

Getting indexes and the losses of the model

Loss, indexes = interp.top_losses()

the top losses model did

For plotting where the model is getting wrong mostly

Interp.plot_top_losses(9, figsize=(15,11))

9 : no of figures which we want to display

Figsize : the size of the figure

For plotting the confusion matrix :

Interp.plot_confusion_matrix(figsize= (12,12), dpi= 60)

It just plots the confusion matrix just like scikitlearn's confusion matrix function

For viewing the most wrong predictions :

Interp.most_confused(min_val= 2)

Min_val paramter : it to show the confusion between the number fo values it's like the model is confused between those 2 classes

Fine tuning model procedures :

For unfreezing the model :

Learn.unfreeze()

Learn – model_name which is already pretrained

Unfreezing the whole model layers (setting it for fine tuning)

For fitting the unfreezed model

Learn.fit_one_cycle(no_of_epochs)

Train whole model again

For loading the model with weights we can do like :

Learn.load(path)

It loads the model in learn

It takes parameter as device so we can specify device which we want to load it on

For finding the learning rate of the model :

Learn.lr_find()

This will give us the valley or suggesting learning rates on which we can see the loss is kind of decreasing

For fitting in between certain learning rate ranges ;

Learn.fit_one_cycle(no_of_epochs lr_max = slice(1e-6,1e-4))

Slice object : provides the iterable object as the output of the we can slice string with this iterable object

So here the slice is giving out the object which will cut the learning rate from 1e-6 to 1e-4

Training the resnet model :

The steps are same

1. **To define the data loaders**

   i. **ImageDataLoaders.from_path_re(image_paths,file_names,pattern(re) , transformationsas batch_tfms = [*aug_transforms(size = 299, max_warp = 0)**
   ii. **Normalize.from_stats(*imagenet_stats)])**
2. **Defining the model learn = cnn_learner(dls, reset, metcrics = error_rate) .to_fp16()**
3. **Learn.lr_find()**
4. **Learn.fit_one_cycle(10)**
5. **Learn.save(filepath) for saving the modl**
6. **Then unfreezing the layers**
7. **Fittign for some more epochs**
8. **If doesn't helps then we can again roll back to the previous model learn.load(filePath);**
9. **Interp = ClassificationInterpretation.from_learner(learn)**
10. **Interp.most_confused(min_val = 2)**
11. **Interp.plot_confusion_matrx(figsize =(,) , dpi = 60)**

Experimenting with the other data fromats same way we can import data from folders	**ImageDataLoaders.from_folder(path,batch_tfms , size , bs)**
Experimeting and we want to load the data from csv files	**ImageDataLoaders.from_csv(path), batch_tfms, size)**
From dataframe	**ImageDataLoaders.from_df(df, path, batch_tfms = tfms, size = 24)**
From path functions	**ImageDataLoaders.from_path_func(path,fn_paths, tfms = tfms , size ,label_func = lambda x : '3' if '/3/' in str (x) else '7'**
From lists	**ImageDataLoaders.from_lists(path, fn_paths, labels, batch_tfsm)**