Fast ai lesson 3 : notes

26 June 2021 16:13

Lesson notes : (Chapter 2 remaining here)

Training data images should be in the same square format for since the neural network is gonna take the inputs which are of same in the shape
For doing this we use :

- Squish (squishing the image)
- Crop (chop off the image)
- Resize (resize in the center)
- RandomResize: (each time grabs the different part of the image) takes parameter (cutting size , min_scale= 0.3)(30 % of the image is gonnna get cut in each iteration / batch)

Resize the default is to grab the center of the image in the shape of

Data Augmentation :

- Using the aug transform function
  - Returns the list of the different transformation (more to train more to learn)

Getting the new batch and doing item transform and the batch transform on it

Bears.new(item_transform = Resize(128),batch_transform= aug_transform(mult = 2))
Mult is multiplied to each of the transform of the following
Item transfrom happens on the cpu more time it takes
batch transform happens on GPU less time + more results

Various parameters of aug_transform are are ***do_flip***=*True*, ***flip_vert***=*False*, ***max_rotate***=*10.0*, ***min_zoom***=*1.0*, ***max_zoom***=*1.1*, ***max_lighting***=*0.2*, ***max_warp***=*0.2*, ***p_affine***=*0.75*, ***p_lighting***=*0.75*, ***xtra_tfms***=*None*, ***size***=*None*, ***mode***='bilinear', ***pad_mode***='reflection', ***align_corners***=*True*, ***batch***=*False*, ***min_scale***=*1.0*

Cleaning the data with fastai cleaner :

Cleaner = ImageClassifierCleaner(learn)
Cleaner
We can select the training or the val data and then we can change the labels , decide to keep or delete etc

For cleaning the data :
For idx in cleaner.delete(): cleaner.fns[idx].unlink()
For idx, cat in
cleaner.change():shutil.move(str(cleaner.fns[idx]),path/cat)

Getting the class names from dataloaders

Dls.vocab
Otherwise if we have model then we can do it like
Learner.dls.vocab

Creating the complete application in Jupyter notebook :

For uploading the file : widgets.FileUpload()
# here we are getting the actual clickable button
Img = PILImage.create(btn_upload.data[-1])
Img
It reads and shows the iamge which we've passed into the uplaoded file image
Output widget
Out_pl = widgets.Output()
Out_pl.clear_output()
Out_pl
This is the output widget
Label_prds = widgets.Label()
It shows the value

Creating the button with ipython widgets
Btn_run = widgets.Button(description = 'what is going to show on button')
We can define what happens on button click by fn
Such as
Def on_click_classificy(change):
    Img = PILImge.create(btn_upload.data[-1])
    Out_pl.clear_output()
    With out_pl: display(img.to_thumb(128,128))
    Pred, pred_idx, probs = learn_inf.predict(img)
    Label.value = f'predicionts etc all thigns'

For attaching this funtion to the iamge handler we will do like
Btn_run.on_click(on_click_classificy)

Widgets.VBox([widgets.label('select image to uplaod') , btn_upload, btn_run, out_pl, lablel_predicion)

Using the deployments on phones (mobile phones)

Many mobile phones have special components for machine learning then we can use them for making some predictions or doing some operations but instead of this we can have like our website will be running on server (streamlit) we can just pass our image or any textual data and then we can view the output

ONNX runtime and AWS sagemaker if you want to use the server with GPU (costly method)

Biases in searches :

Images such as the healthy skin is like young white women touching the face that's what the healthy skin classifier will learn to detect but that's not the healthy skin actually
This can build the modle which can be biased
So the images or the training data must be non biased cause in case of the healthy skin problem we got women with touching face as the healthy skin for the bing image search

In the actual world the data can be tatally different which we are getting in the training time

When the data is not ready we can have the data which is totally biased we need to see how the data is gathered what are the limitations of the data

We can't know the entire behaviour of the neural network since it's learning on it's own from the data which we have asked it to train on
We are trying to get the sophisticated behaviour

We can do human check all the predictions if the training data is good

Once we have the confirmation the data is good then we can start gradual expansion

We can gradually fit or fine tune the model based on the data comes day by day so that we will not explore the domain shift cause model is adapting itself cause it's developingn daily

You can't trust the daily data also

You should have some randomness
Eg in crime detection sys

If you have this incremental approach for devloping the model then your model may consider some area which is high in crime and send many police officers over there so as they are there they find more crime and then our model is totally biased on that
It's like predicting the future policing not predicting future crime

Another example for this is arrest of black people compared to white people is more cause the bias shown in the data

We can't be sure that the system is perfect but we can check that the system is getting biased by viewing it's parameter

Chapter 3 starting :
Recognizing handwritten digits (lesson 3) notebook will be in notebooks folder :

For plotting the function in fast ai which is kind of giving some inputs and function is also giving some kind of outputs we can have the plot of fuctions as:

Plot_function(f,'x','x**2')

2nd para is label for x axis and 3rd para is the label for y axis

```
# WE for modifying the funcion later and to tell pytorch to remember that you
have to do this some derivative and calculationns later
xt = tensor(3.).requires_grad_() # underscroll at the end of function in
pytorch means in place modification it modifies the value which you are gonna
pass
```

Book notes :