Chapter 5 - Image Classification

Author: Ita Ćirović Donev

Questionnaire

1. Why do we first resize to a large size on the CPU, and then to a smaller size on the GPU?

When performing data augmentation on images we first resize to a larger image so that when we apply data augmentation technique we would not lose pixels in the image, i.e. certain areas of the image. For example, if we would rotate the image the edges would be black and hence providing no information for our learner. However, if we perform the same rotation on a larger image and then resize to a smaller image to be feed into a neural network we are not losing pixel information.

2. If you are not familiar with regular expressions, find a regular expression tutorial and some problem sets, and complete them. Have a look on the book's website for suggestions.

Some interesting references for regular expressions:

- RegexOne
- RegexBuddy
- Python re
- Regular Expressions in Python: A Complete Guide - Real Python

3. What are the two ways in which data is most commonly provided for most deep learning datasets?

Data is usually presented in the following two ways:

- **individual files** in form of text documents, images, etc. The filenames or the folder names in which the files are saved usually depict the target label
- **data tables** (e.g. in form of csv files) in form of rows and columns, where rows generally depict one sample and the columns provide information about that sample, such as sample ID, metadata, etc.

4. Look up the documentation for `L` and try using a few of the new methods that it adds.

Class `L` represents an enhanced version of the Python list type. For most of the functions and methods in fastai we can use the `L` class. General description for class `L` can be found here. Some properties are:

- it truncates the list if there are more than 10 elements
- in addition to the list object it also returns number of elements in the underlying list object
- uses indexing as an array in Python

For the source code use `??L` in Python with full documentation in fastcore/foundations.

Few methods:

- create a list using `range`:

```
# create a list using range
t = L(range(12)); t
>>> (#12) [0,1,2,3,4,5,6,7,8,9...]
```

- apply operations

```
t = L()
```

```
# Add 1 to t
t.append(1)
>>> (#1) [1]

# Add [2,3] to existing t
t += [3,2]
>>> (#3) [1,3,2]

# Add another list
t = t+[4]; t
>>> (#4) [1,3,2,4]

# Add 5 to the beginning of the list
t = 5 + t; t
>>> (#5) [5,1,3,2,4]

# Use map to apply a function to each element of L
t = t.map(operator.neg); t
(#5) [-5,-1,-3,-2,-4]
```

- apply methods

```
# Filter values
t = L(range(12))
t.filter(lambda i:i<2)
>>> (#2) [0,1]

# Filter but return indices of matching values
t = L([1,5,4,6,7,8,3,2,1])
t.argwhere(lambda i:i<3)
>>> (#3) [0,7,8]

# Concatenate elements
t = L([0,1,2,3],L(5,6),4,L(3,2)).concat(); t
(#9) [0,1,2,3,5,6,4,3,2]

# Sort elements
t.sorted()
>>> (#9) [0,1,2,2,3,3,4,5,6]

# Get unique elements
t.unique()
>>> (#7) [0,1,2,3,4,5,6]
```

5. Look up the documentation for the Python `pathlib` module and try using a few methods of the `Path` class.

Documentation for the `pathlib` module - link.

```
from pathlib import Path

# Define the path - Colab connected to Google drive
p = Path('/content/sample_data/')

# get current working directory
Path.cwd()
>>> Path('/content/sample_data')

# Test is it a directory
p.is_dir()
>>> True
```

```
# get the file name extensions in the directory
s = [i.suffix for i in p.iterdir() if i.is_file()]; s
>>> ['.json', '.md', '.csv', '.csv', '.csv', '.csv']

# Using L class get the unique file extensions
L(s).unique()
>>> (#3) ['.json','.md','.csv']
```

6. Give two examples of ways that image transformations can degrade the quality of the data.

- image rotation changes the corners of an image into black non-informative pixels
  - interpolation of these spaces create some pixels but not of the same quality as the original image
- zooming operations also downgrade the image requiring interpolation

7. What method does fastai provide to view the data in a DataLoaders?

With `one_batch` method we can view the real data from the `DataLoaders`. It returns the dependent and independent variables. We can use it as follows:

```
x,y = dls.one_batch()
```

where `dls` is a dataloader object.

8. What method does fastai provide to help you debug a DataBlock?

To ensure that the dataset has been constructed correctly and does not throw any errors we can use the `summary` method, which creates a batch from the source we specified and provides all the details of the processes that have been applied on the data. If any of the processes fail it will show where the error happens as well as some insight into how to potentially fix it.

We can use it as follows:

```
pets = DataBlock(...)
pets.summary(path/'images')
```

9. Should you hold off on training a model until you have thoroughly cleaned your data?

Once we have the initial dataloader correctly defined we should train a baseline model. This provides us with the benchmark results in its simple form and we an ensure that the complete pipeline of model training is working correctly. It is much easier to correct for mistakes if the complete process is on a smaller scale. Also it is much easier to understand it. Furthermore, having a simple baseline we can easily iterate our training process and see its impacts on the model loss and the chosen metric.

10. What are the two pieces that are combined into cross-entropy loss in PyTorch?

Cross-entropy loss is comprised of the softmax and the negative log likelihood.

11. What are the two properties of activations that softmax ensures? Why is this important?

Properties of the softmax activation function:

- all activations are in the range `(0,1)`
  - we get probabilities for each target class
  - `exp()` ensures that we always have a positive result
- the sum of all activations is equal to 1
  - dividing by the sum of all `exp()`'s gives us the values which will sum to one since it gives the proportional value to the total.

Note: `exp()` amplifies the values so we will get a bigger range between results which are initially closer together - the model is more/less sure about the final class

12. When might you want your activations to not have these two properties?
Being more or less sure about the target class is good if we know the true class label, like in training. However, in inference, when we are testing our model, we want the information where the model is unsure about the target class.

13. Calculate the `exp` and `softmax` columns of Figure 5-3 yourself (i.e., in a spreadsheet, with a calculator, or in a notebook).

The figure 5-3 is done for the pet breeds example in chapter 5. The code is the following given that we have defined the `dls` dataloader:

```
# Get one batch
Xs,ys = dls.one_batch()

# Compute predictions for the batch
preds_Xs,_ = learn.get_preds(dl=[(Xs,ys)])

# Choose one image and print the target label
preds = preds_Xs[0]
y = y[0]; y
>>> tensor(15)
```

Create initial table with class names and predictions for each class for a given image:

```
# Get the names of the classes from the dataloader
tclass = dls.vocab

# Create a table
df = pd.DataFrame({'class_name': tclass,
                   'output': preds})
```

Compute the exponential and the softmax for each prediction:

```
# Exponential
df['exp'] = np.exp(df['output'])

# Softmax
sum_exp = df['exp'].sum()
df['softmax'] = df['exp']/sum_exp
```

Sort the values for the softmax (descending order) and print the first 5 rows of the table

```
df.sort_values('softmax', ascending=False)
>>>
                   class_name        output       exp   softmax
15                      beagle  9.994148e-01  2.716692  0.070167
14               basset_hound  3.874889e-04  1.000388  0.025838
12           american_bulldog  5.565286e-05  1.000056  0.025830
13   american_pit_bull_terrier  5.316383e-05  1.000053  0.025830
18      english_cocker_spaniel  1.861128e-05  1.000019  0.025829
34  staffordshire_bull_terrier  1.803941e-05  1.000018  0.025829
30               saint_bernard  1.097619e-05  1.000011  0.025829
1                       Bengal  7.443317e-06  1.000007  0.025828
28                  pomeranian  6.410583e-06  1.000006  0.025828
33                   shiba_inu  3.331156e-06  1.000003  0.025828
20          german_shorthaired  3.163265e-06  1.000003  0.025828
36           yorkshire_terrier  3.128756e-06  1.000003  0.025828
```

16	boxer	2.815527e-06	1.000003	0.025828
35	wheaten_terrier	2.218476e-06	1.000002	0.025828
17	chihuahua	1.961671e-06	1.000002	0.025828
19	english_setter	1.709348e-06	1.000002	0.025828
26	miniature_pinscher	1.442625e-06	1.000002	0.025828
2	Birman	1.307681e-06	1.000001	0.025828
11	Sphynx	1.215433e-06	1.000001	0.025828
4	British_Shorthair	6.793929e-07	1.000001	0.025828
0	Abyssinian	6.745271e-07	1.000001	0.025828
9	Russian_Blue	6.645611e-07	1.000001	0.025828
21	great_pyrenees	3.223752e-07	1.000000	0.025828
22	havanese	3.915477e-07	1.000000	0.025828
23	japanese_chin	2.533390e-07	1.000000	0.025828
31	samoyed	3.332337e-07	1.000000	0.025828
8	Ragdoll	3.235540e-07	1.000000	0.025828
10	Siamese	2.132258e-07	1.000000	0.025828
25	leonberger	1.729973e-07	1.000000	0.025828
6	Maine_Coon	1.404158e-07	1.000000	0.025828
27	newfoundland	1.091678e-07	1.000000	0.025828
5	Egyptian_Mau	1.335720e-07	1.000000	0.025828
29	pug	1.687634e-07	1.000000	0.025828
32	scottish_terrier	1.585672e-07	1.000000	0.025828
3	Bombay	1.656887e-07	1.000000	0.025828
24	keeshond	1.617980e-07	1.000000	0.025828
7	Persian	6.237953e-08	1.000000	0.025828

as we can see the model made the correct prediction that the pet breed is a beagle (15). To see the image of the pet use:

```
npimg = Xs[0].cpu()
plt.imshow(np.transpose(npimg, (1, 2, 0)))
plt.show()
```

confirming that it does look like a beagle.

14. Why can't we use `torch.where` to create a loss function for datasets where our label can have more than two categories?
In case of having more than 2 target categories we need a loss function to tell us which out of all categories is most likely to be true given our input data. In order to do this we need the predictions for each class to be proportional to the output of all classes so that they are evenly compared. Since `torch.where` does not provide this for more than 2 classes we can not use it.

15. What is the value of log(–2)? Why?
Not defined since the $log(x)$ is defined on $(0, +inf)$.

16. What are two good rules of thumb for picking a learning rate from the learning rate finder?
Pick a learning rate either:

1. one magnitude lower than the learning rate at the minimum loss,
   - because at minimum the loss is not decreasing anymore
2. the last point where the loss was clearly decreasing

17. What two steps does the `fine_tune` method do?
The `fine_tune` method does the following two things:

1. train the randomly added layers for one epoch, with all other layers frozen
2. unfreezes all the layers and trains them for the number of epochs requested

18. In Jupyter Notebook, how do you get the source code for a method or function?
By typing and running `??method_name` in a Jupyter Notebook cell.

19. What are discriminative learning rates?
Different layers in the neural network learn different features which vary in complexity, so we might not need the same learning rate for all layers in the network. Discriminative learning rates represent this varying learning rates across the network in the following way:

- use lower learning rate for early layers
- use higher learning rates for the later layers and the randomly added layers

20. How is a Python `slice` object interpreted when passed as a learning rate to fastai?
We can add `slice` as follows to the training process: `fit_one_cycle(lr_max=slice(1e-6, 1e-4))` which would apply the learning rate of `1e-6` to the first layer of the neural network and `1e-4` to the final layer of the neural network. The in-between layers will have the equidistant learning rate from the given range.

21. Why is early stopping a poor choice when using 1cycle training?
1-cycle learning starts the training with a low learning rate, which is then gradually increases as the training progresses and finally decreases in the last section of training. If we would use early stopping in such training we would not take the full advantage of the learning rate since it might never get to the point of using the small values.

22. What is the difference between `resnet50` and `resnet101`?
The difference is in the number of layers, `resnet50` has 50 layers while `resnet101` has 101 layers. They are both trained on ImageNet.

23. What does `to_fp16` do?
Large architectures take a long time to train. To reduce the training time we can use lower precision such as *half-precision floating point* or *fp16*. We can use it in fastai as follows:

```
learn = vision_learner(dls, resnet34, metrics=error_rate).to_fp16()
```