Image classification with Convolutional Neural Networks

Welcome to the first week of the second deep learning certificate! We're going to use convolutional neural networks (CNNs) to allow our computer to see - something that is only possible thanks to deep learning.

Introduction to our first task: 'Dogs vs Cats'

We're going to try to create a model to enter the Dogs vs Cats competition at Kaggle. There are 25,000 labelled dog and cat photos available for training, and 12,500 in the test set that we have to try to label for this competition. According to the Kaggle web-site, when this competition was launched (end of 2013): "State of the art: The current literature suggests machine classifiers can score above 80% accuracy on this task". So if we can beat 80%, then we will be at the cutting edge as of 2013!

In [1]:

```
# Put these at the top of every notebook, to get automatic reloading and inline
 plotting
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

Here we import the libraries we need. We'll learn about what each does during the course.

In [2]:

```
# This file contains all the main external libs we'll use
from fastai.imports import *
```

```
/home/paperspace/anaconda3/envs/fastai/lib/python3.6/site-packages/s
klearn/ensemble/weight_boosting.py:29: DeprecationWarning: numpy.cor
e.umath_tests is an internal NumPy module and should not be importe
d. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [4]:

```
from fastai.transforms import *
from fastai.conv_learner import *
from fastai.model import *
from fastai.dataset import *
from fastai.sgdr import *
from fastai.plots import *
```

`PATH` is the path to your data - if you use the recommended setup approaches from the lesson, you won't need to change this. `sz` is the size that the images will be resized to in order to ensure that the training runs quickly. We'll be talking about this parameter a lot during the course. Leave it at `224` for now.

In [5]:

```
PATH = "data/bearsnake/"
subfolder1='bear'
subfolder2='snake'
sz=224
```

PATH = "data/dogscats/" sz=224

It's important that you have a working NVidia GPU set up. The programming framework used to behind the scenes to work with NVidia GPUs is called CUDA. Therefore, you need to ensure the following line returns `True` before you proceed. If you have problems with this, please check the FAQ and ask for help on the forums (http://forums.fast.ai).

In [28]:

```
torch.cuda.is_available()
```

Out[28]:

```
True
```

In addition, NVidia provides special accelerated functions for deep learning in a package called CuDNN. Although not strictly necessary, it will improve training performance significantly, and is included by default in all supported fastai configurations. Therefore, if the following does not return `True`, you may want to look into why.

In [29]:

```
torch.backends.cudnn.enabled
```

Out[29]:

```
True
```

Extra steps if NOT using Crestle or Paperspace or our scripts

The dataset is available at http://files.fast.ai/data/dogscats.zip (http://files.fast.ai/data/dogscats.zip). You can download it directly on your server by running the following line in your terminal. `wget http://files.fast.ai/data/dogscats.zip`. You should put the data in a subdirectory of this notebook's directory, called `data/`. Note that this data is already available in Crestle and the Paperspace fast.ai template.

Extra steps if using Crestle

Crestle has the datasets required for fast.ai in /datasets, so we'll create symlinks to the data we want for this competition. (NB: we can't write to /datasets, but we need a place to store temporary files, so we create our own writable directory to put the symlinks in, and we also take advantage of Crestle's `/cache/` faster temporary storage space.)

To run these commands (**which you should only do if using Crestle**) remove the # characters from the start of each line.

In [30]:

```
os.makedirs('data/bearsnake/models', exist_ok=True)
os.makedirs('data/bearsnake/train', exist_ok=True)
os.makedirs('data/bearsnake/valid', exist_ok=True)


#!ln -s /datasets/fast.ai/bearsnake/train {PATH}
#!ln -s /datasets/fast.ai/bearsnake/test {PATH}
#!ln -s /datasets/fast.ai/bearsnake/valid {PATH}

#os.makedirs('/cache/tmp', exist_ok=True)
#!ln -fs /cache/tmp {PATH}
```

In [36]:

```
#os.makedirs('data/bearsnake/train/' + subfolder1, exist_ok=True)
#os.makedirs('data/bearsnake/train/' + subfolder2, exist_ok=True)

os.makedirs('data/bearsnake/valid/' + subfolder1, exist_ok=True)
os.makedirs('data/bearsnake/valid/' + subfolder2, exist_ok=True)
```

In [39]:

```
os.listdir('/home/paperspace/fastai/courses/dl1/data')
```

Out[39]:

```
['bearsnake.zip',
 '.ipynb_checkpoints',
 'dogscats',
 'valid',
 'bearsnake',
 'dogscats.zip',
 'train']
```

In [41]:

```
os.getcwd()
```

Out[41]:

```
'/home/paperspace/fastai/courses/dl1'
```

In [42]:

```
import zipfile

path_to_zip_file= 'data' + '/bearsnake.zip'
directory_to_extract_to='data/bearsnake/train'

zip_ref = zipfile.ZipFile(path_to_zip_file, 'r')
zip_ref.extractall(directory_to_extract_to)
zip_ref.close()
```

In [6]:

```
#os.makedirs('/cache/tmp', exist_ok=True)
#!ln -fs /cache/tmp {PATH}
```

First look at cat pictures

Our library will assume that you have *train* and *valid* directories. It also assumes that each dir will have subdirs for each class you wish to recognize (in this case, 'cats' and 'dogs').

In [6]:

```
os.listdir(PATH)
```

Out[6]:

```
['.ipynb_checkpoints', 'test', 'valid', 'models', 'train', 'tmp']
```

In [7]:

```
os.listdir(f'{PATH}valid')
```

Out[7]:

```
['snake', 'bear']
```

In [8]:

```
files = os.listdir(f'{PATH}valid/bear')[:5]
files
```

Out[8]:

```
['12. open-uri20150608-27674-gb329t_bb8ca7ae.jpeg',
 '16. bears-crossing-road1900x600i-1.jpg',
 '13. 5b62d01506bd8a53b6c4928e25fa9b8a_1.jpg',
 '1. bear-pepper-spray.jpg',
 '10. 2bear_2.jpg']
```

In [9]:

```
full=os.listdir(f'{PATH}valid/bear')
```

In [10]:

```
len(full)
```

Out[10]:

```
10
```

In [11]:

```
img = plt.imread(f'{PATH}valid/bear/{files[0]}')
plt.imshow(img);
```

Here is how the raw data looks like

In [12]:

```
img.shape
```

Out[12]:

```
(880, 1320, 3)
```

In [13]:

```
img[:4,:4]
```

Out[13]:

```
array([[[ 52, 107, 138],
        [ 55, 110, 141],
        [ 51, 106, 137],
        [ 55, 112, 142]],

       [[ 55, 110, 141],
        [ 52, 107, 138],
        [ 51, 106, 137],
        [ 55, 112, 142]],

       [[ 53, 108, 139],
        [ 55, 110, 141],
        [ 54, 109, 140],
        [ 52, 107, 138]],

       [[ 57, 112, 143],
        [ 57, 112, 143],
        [ 56, 111, 142],
        [ 53, 107, 141]]], dtype=uint8)
```

Our first model: quick start

We're going to use a **pre-trained** model, that is, a model created by some one else to solve a different problem. Instead of building a model from scratch to solve a similar problem, we'll use a model trained on ImageNet (1.2 million images and 1000 classes) as a starting point. The model is a Convolutional Neural Network (CNN), a type of Neural Network that builds state-of-the-art models for computer vision. We'll be learning all about CNNs during this course.

We will be using the **resnet34** model. resnet34 is a version of the model that won the 2015 ImageNet competition. Here is more info on resnet models (https://github.com/KaimingHe/deep-residual-networks). We'll be studying them in depth later, but for now we'll focus on using them effectively.

Here's how to train and evalulate a *dogs vs cats* model in 3 lines of code, and under 20 seconds:

In [6]:

```
# Uncomment the below if you need to reset your precomputed activations
# shutil.rmtree(f'{PATH}tmp', ignore_errors=True)
```

In [14]:

```
arch=resnet34
data = ImageClassifierData.from_paths(PATH, tfms=tfms_from_model(arch, sz))
learn = ConvLearner.pretrained(arch, data, precompute=True)
learn.fit(0.01, 3)
```

```
  0%|          | 0/2 [00:00<?, ?it/s]
```

```
---------------------------------------------------------------------------
-------
OSError                                   Traceback (most recent cal
l last)
<ipython-input-14-42d5f498bf97> in <module>()
      1 arch=resnet34
      2 data = ImageClassifierData.from_paths(PATH, tfms=tfms_from_m
odel(arch, sz))
----> 3 learn = ConvLearner.pretrained(arch, data, precompute=True)
      4 learn.fit(0.01, 3)

~/fastai/courses/dl1/fastai/conv_learner.py in pretrained(cls, f, da
ta, ps, xtra_fc, xtra_cut, custom_head, precompute, pretrained, **kw
args)
    112         models = ConvnetBuilder(f, data.c, data.is_multi, da
ta.is_reg,
    113             ps=ps, xtra_fc=xtra_fc, xtra_cut=xtra_cut, custo
m_head=custom_head, pretrained=pretrained)
--> 114         return cls(data, models, precompute, **kwargs)
    115
    116     @classmethod

~/fastai/courses/dl1/fastai/conv_learner.py in __init__(self, data,
 models, precompute, **kwargs)
     98         if hasattr(data, 'is_multi') and not data.is_reg and
 self.metrics is None:
     99             self.metrics = [accuracy_thresh(0.5)] if self.da
ta.is_multi else [accuracy]
--> 100         if precompute: self.save_fc1()
    101         self.freeze()
    102         self.precompute = precompute

~/fastai/courses/dl1/fastai/conv_learner.py in save_fc1(self)
    177         m=self.models.top_model
    178         if len(self.activations[0])!=len(self.data.trn_ds):
--> 179             predict_to_bcolz(m, self.data.fix_dl, act)
    180         if len(self.activations[1])!=len(self.data.val_ds):
    181             predict_to_bcolz(m, self.data.val_dl, val_act)

~/fastai/courses/dl1/fastai/model.py in predict_to_bcolz(m, gen, ar
r, workers)
     15     lock=threading.Lock()
     16     m.eval()
---> 17     for x,*_ in tqdm(gen):
     18         y = to_np(m(VV(x)).data)
     19         with lock:

~/anaconda3/envs/fastai/lib/python3.6/site-packages/tqdm/_tqdm.py in
 __iter__(self)
    929 """, fp_write=getattr(self.fp, 'write', sys.stderr.write))
    930
--> 931             for obj in iterable:
    932                 yield obj
    933                 # Update and possibly print the progressbar.

~/fastai/courses/dl1/fastai/dataloader.py in __iter__(self)
     86                 # avoid py3.6 issue where queue is infinite
 and can result in memory exhaustion
     87                 for c in chunk_iter(iter(self.batch_sampler)
, self.num_workers*10):
---> 88                     for batch in e.map(self.get_batch, c):
```

```
     89                         yield get_tensor(batch, self.pin_mem
ory, self.half)
     90

~/anaconda3/envs/fastai/lib/python3.6/concurrent/futures/_base.py in
 result_iterator()
    584                     # Careful not to keep a reference to the
 popped future
    585                     if timeout is None:
--> 586                         yield fs.pop().result()
    587                     else:
    588                         yield fs.pop().result(end_time - tim
e.time())

~/anaconda3/envs/fastai/lib/python3.6/concurrent/futures/_base.py in
 result(self, timeout)
    430                     raise CancelledError()
    431             elif self._state == FINISHED:
--> 432                 return self.__get_result()
    433             else:
    434                 raise TimeoutError()

~/anaconda3/envs/fastai/lib/python3.6/concurrent/futures/_base.py in
 __get_result(self)
    382     def __get_result(self):
    383         if self._exception:
--> 384             raise self._exception
    385         else:
    386             return self._result

~/anaconda3/envs/fastai/lib/python3.6/concurrent/futures/thread.py i
n run(self)
     54
     55         try:
---> 56             result = self.fn(*self.args, **self.kwargs)
     57         except BaseException as exc:
     58             self.future.set_exception(exc)

~/fastai/courses/dl1/fastai/dataloader.py in get_batch(self, indice
s)
     73
     74     def get_batch(self, indices):
---> 75         res = self.np_collate([self.dataset[i] for i in indi
ces])
     76         if self.transpose:   res[0] = res[0].T
     77         if self.transpose_y: res[1] = res[1].T

~/fastai/courses/dl1/fastai/dataloader.py in <listcomp>(.0)
     73
     74     def get_batch(self, indices):
---> 75         res = self.np_collate([self.dataset[i] for i in indi
ces])
     76         if self.transpose:   res[0] = res[0].T
     77         if self.transpose_y: res[1] = res[1].T

~/fastai/courses/dl1/fastai/dataset.py in __getitem__(self, idx)
    194             xs,ys = zip(*[self.get1item(i) for i in range(*i
dx.indices(self.n))])
    195             return np.stack(xs),ys
--> 196         return self.get1item(idx)
    197
```

```
    198     def __len__(self): return self.n

~/fastai/courses/dl1/fastai/dataset.py in get1item(self, idx)
    187
    188     def get1item(self, idx):
--> 189         x,y = self.get_x(idx),self.get_y(idx)
    190         return self.get(self.transform, x, y)
    191

~/fastai/courses/dl1/fastai/dataset.py in get_x(self, i)
    271         super().__init__(transform)
    272     def get_sz(self): return self.transform.sz
--> 273     def get_x(self, i): return open_image(os.path.join(self.
path, self.fnames[i]))
    274     def get_n(self): return len(self.fnames)
    275

~/fastai/courses/dl1/fastai/dataset.py in open_image(fn)
    249         raise OSError('No such file or directory: {}'.format
(fn))
    250     elif os.path.isdir(fn) and not str(fn).startswith("http"
):
--> 251         raise OSError('Is a directory: {}'.format(fn))
    252     else:
    253         #res = np.array(Image.open(fn), dtype=np.float32)/25
5

OSError: Is a directory: data/bearsnake/train/ snake/.ipynb_checkpoi
nts
```