Transfer learning in computer vision

Fine-tuning a pretrained model on the pets dataset.

Using MobileNet v2

In [2]:
```
from local.test import *
from local.basics import *
from local.callback.all import *
from local.vision.all import *
```

In [3]:
```
from torchvision.models import mobilenet_v2
```

In [4]:
```
pretrained = True
model = mobilenet_v2(pretrained)
```

load pet databunch

In [5]:
```
pets = DataBlock(types=(PILImage, Category),
                 get_items=get_image_files,
                 splitter=RandomSplitter(),
                 get_y=RegexLabeller(pat = r'/([^/]+)_\d+.jpg$'))

dbunch = pets.databunch(untar_data(URLs.PETS)/"images", item_tfms=RandomResizedCrop(300,
min_scale=0.5), bs=64,
                        batch_tfms=[*aug_transforms(size=224), Normalize(*imagenet_stats
)])

num_classes = 22
```

cut the model and attach custom head

In [6]:
```
ll = list(enumerate(model.children()))
body = ll[0]
body = nn.Sequential(body[1])
```

In [7]:
```
ll[1] #original head
```

Out[7]:
```
(1, Sequential(
   (0): Dropout(p=0.2, inplace=False)
   (1): Linear(in_features=1280, out_features=1000, bias=True)
 ))
```

In [8]:
```
custom_head = nn.Sequential(nn.Dropout(p=0.2, inplace=False), nn.Linear(in_features=1280
, out_features=num_classes))
#TODO we could just use the create_head function in learner.py
```

In [9]:
```
model = nn.Sequential(body, custom_head)
```

In [10]:
```
opt_func = partial(Adam, lr=3e-3, wd=0.01)
```

In [11]:
```
loss_func = getattr(dbunch.train_ds, 'loss_func', None)
```

In [12]:
```
def _default_split(m:nn.Module): return L(m[0], m[1:]).map(params)
```

In [13]:
```
learn = Learner(dbunch, model, opt_func=opt_func, loss_func=loss_func, splitter=_default
_split)
learn.freeze()
```

In [14]:
```
model
```

Out[14]:
```
Sequential(
  (0): Sequential(
    (0): Sequential(
      (0): ConvBNReLU(
        (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=Fals
e)
        (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU6(inplace=True)
      )
      (1): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), group
s=32, bias=False)
            (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
            (2): ReLU6(inplace=True)
          )
          (1): Conv2d(32, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (2): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(16, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), group
s=96, bias=False)
            (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(96, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (3): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=144, bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(144, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (4): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(144, 144, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), gro
ups=144, bias=False)
            (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_st
```

```
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(144, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (5): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=192, bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(192, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (6): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=192, bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(192, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (7): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(192, 192, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), gro
ups=192, bias=False)
            (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (8): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(64, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
```

```
          (1): ConvBNReLU(
            (0): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=384, bias=False)
            (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(384, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (9): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(64, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=384, bias=False)
            (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(384, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (10): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(64, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=384, bias=False)
            (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(384, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (11): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(64, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=384, bias=False)
            (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(384, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (12): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(96, 576, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

```
            (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(576, 576, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=576, bias=False)
            (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(576, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (13): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(96, 576, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(576, 576, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=576, bias=False)
            (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(576, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        )
      )
      (14): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(96, 576, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(576, 576, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), gro
ups=576, bias=False)
            (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(576, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
        )
      )
      (15): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=960, bias=False)
            (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
        )
      )
```

```
      (16): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=960, bias=False)
            (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
        )
      )
      (17): InvertedResidual(
        (conv): Sequential(
          (0): ConvBNReLU(
            (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (1): ConvBNReLU(
            (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), gro
ups=960, bias=False)
            (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
            (2): ReLU6(inplace=True)
          )
          (2): Conv2d(960, 320, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
        )
      )
      (18): ConvBNReLU(
        (0): Conv2d(320, 1280, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(1280, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
        (2): ReLU6(inplace=True)
      )
    )
  )
  (1): Sequential(
    (0): Dropout(p=0.2, inplace=False)
    (1): Linear(in_features=1280, out_features=22, bias=True)
  )
)
```

In [15]:
```
learn.fit_one_cycle(1)
```

epoch	train_loss	valid_loss	time
0	0.000000		00:01

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-15-4dfb24161c57> in <module>
----> 1 learn.fit_one_cycle(1)

~/projects/fastai_dev/dev/local/callback/schedule.py in fit_one_cycle(self, n_epoch, lr_
max, div, div_final, pct_start, wd, moms, cbs, reset_opt)
     96     scheds = {'lr': combined_cos(pct_start, lr_max/div, lr_max, lr_max/div_fina
l),
     97               'mom': combined_cos(pct_start, *moms)}
---> 98     self.fit(n_epoch, cbs=ParamScheduler(scheds)+L(cbs), reset_opt=reset_opt, wd
=wd)
     99
    100 #Cell

~/projects/fastai_dev/dev/local/learner.py in fit(self, n_epoch, lr, wd, cbs, reset_opt)
    233                     try:
    234                         self.epoch=epoch;          self('begin_epoch')
--> 235                         self._do_epoch_train()
    236                         self._do_epoch_validate()
    237                     except CancelEpochException:   self('after_cancel_epoch')

~/projects/fastai_dev/dev/local/learner.py in _do_epoch_train(self)
    211         try:
    212             self.dl = self.dbunch.train_dl;                  self('begin_train')
--> 213             self.all_batches()
    214         except CancelTrainException:                         self('after_cancel_
train')
    215         finally:                                             self('after_train')

~/projects/fastai_dev/dev/local/learner.py in all_batches(self)
    189     def all_batches(self):
    190         self.n_iter = len(self.dl)
--> 191         for o in enumerate(self.dl): self.one_batch(*o)
    192
    193     def one_batch(self, i, b):

~/projects/fastai_dev/dev/local/learner.py in one_batch(self, i, b)
    195         try:
    196             self._split(b);                                  self('begin_batch')
--> 197             self.pred = self.model(*self.xb);                self('after_pred')
    198             if len(self.yb) == 0: return
    199             self.loss = self.loss_func(self.pred, *self.yb); self('after_loss')

~/anaconda3/envs/fastai_dev/lib/python3.7/site-packages/torch/nn/modules/module.py in __
call__(self, *input, **kwargs)
    539             result = self._slow_forward(*input, **kwargs)
    540         else:
--> 541             result = self.forward(*input, **kwargs)
    542         for hook in self._forward_hooks.values():
    543             hook_result = hook(self, input, result)

~/anaconda3/envs/fastai_dev/lib/python3.7/site-packages/torch/nn/modules/container.py in
forward(self, input)
     90     def forward(self, input):
     91         for module in self._modules.values():
---> 92             input = module(input)
     93         return input
     94

~/anaconda3/envs/fastai_dev/lib/python3.7/site-packages/torch/nn/modules/module.py in __
call__(self, *input, **kwargs)
    539             result = self._slow_forward(*input, **kwargs)
    540         else:
--> 541             result = self.forward(*input, **kwargs)
    542         for hook in self._forward_hooks.values():
    543             hook_result = hook(self, input, result)

~/anaconda3/envs/fastai_dev/lib/python3.7/site-packages/torch/nn/modules/container.py in
forward(self, input)
     90     def forward(self, input):
     91         for module in self._modules.values():
---> 92             input = module(input)
     93         return input
     94
```

```
~/anaconda3/envs/fastai_dev/lib/python3.7/site-packages/torch/nn/modules/module.py in __
call__(self, *input, **kwargs)
    539             result = self._slow_forward(*input, **kwargs)
    540         else:
--> 541             result = self.forward(*input, **kwargs)
    542         for hook in self._forward_hooks.values():
    543             hook_result = hook(self, input, result)

~/anaconda3/envs/fastai_dev/lib/python3.7/site-packages/torch/nn/modules/linear.py in fo
rward(self, input)
     85
     86     def forward(self, input):
---> 87         return F.linear(input, self.weight, self.bias)
     88
     89     def extra_repr(self):

~/anaconda3/envs/fastai_dev/lib/python3.7/site-packages/torch/nn/functional.py in linear
(input, weight, bias)
   1370         ret = torch.addmm(bias, input, weight.t())
   1371     else:
-> 1372         output = input.matmul(weight.t())
   1373         if bias is not None:
   1374             output += bias

RuntimeError: size mismatch, m1: [573440 x 7], m2: [1280 x 22] at /opt/conda/conda-bld/p
ytorch_1570910687650/work/aten/src/THC/generic/THCTensorMathBlas.cu:290
```

fin