```
#hide
! [ -e /content ] && pip install -Uqq fastbook
import fastbook
fastbook.setup_book()
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 719.8/719.8 KB 10.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.7/6.7 MB 46.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 16.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 469.0/469.0 KB 7.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.9/132.9 KB 8.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.2/212.2 KB 22.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 28.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.2/199.2 KB 11.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 KB 6.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 53.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 264.6/264.6 KB 25.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 158.8/158.8 KB 17.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.2/114.2 KB 10.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 74.1 MB/s eta 0:00:00
Mounted at /content/gdrive
```

```
#hide
from fastbook import *
```

Other Computer Vision Problems

Multi-Label Classification

The Data

```
from fastai.vision.all import *
path = untar_data(URLs.PASCAL_2007)
```

```
100.00% [1637801984/1637796771 02:04<00:00]
```

```
df = pd.read_csv(path/'train.csv')
df.head()
```

	fname	labels	is_valid
0	000005.jpg	chair	True
1	000007.jpg	car	True
2	000009.jpg	horse person	True
3	000012.jpg	car	False
4	000016.jpg	bicycle	True

Sidebar: Pandas and DataFrames

```
df.iloc[:,0]
```

```
0       000005.jpg
1       000007.jpg
2       000009.jpg
3       000012.jpg
4       000016.jpg
           ...
5006    009954.jpg
5007    009955.jpg
5008    009958.jpg
5009    009959.jpg
5010    009961.jpg
Name: fname, Length: 5011, dtype: object
```

```
df.iloc[0,:]
# Trailing :s are always optional (in numpy, pytorch, pandas, etc.),
#   so this is equivalent:
df.iloc[0]
```

```
fname       000005.jpg
labels           chair
is_valid          True
Name: 0, dtype: object
```

```
df['fname']
```

```
0       000005.jpg
1       000007.jpg
2       000009.jpg
3       000012.jpg
4       000016.jpg
           ...
5006    009954.jpg
5007    009955.jpg
5008    009958.jpg
5009    009959.jpg
5010    009961.jpg
Name: fname, Length: 5011, dtype: object
```

```
tmp_df = pd.DataFrame({'a':[1,2], 'b':[3,4]})
tmp_df
```

	a	b
0	1	3
1	2	4

```
tmp_df['c'] = tmp_df['a']+tmp_df['b']
tmp_df
```

	a	b	c
0	1	3	4
1	2	4	6

End sidebar

Constructing a DataBlock

```
dblock = DataBlock()
```

```
dsets = dblock.datasets(df)
```

```
len(dsets.train),len(dsets.valid)
```

```
(4009, 1002)
```

```
x,y = dsets.train[0]
x,y
```

```
(fname       008663.jpg
 labels      car person
 is_valid         False
 Name: 4346, dtype: object, fname       008663.jpg
 labels      car person
 is_valid         False
 Name: 4346, dtype: object)
```

```
x['fname']
```

```
'008663.jpg'
```

```
dblock = DataBlock(get_x = lambda r: r['fname'], get_y = lambda r: r['labels'])
dsets = dblock.datasets(df)
dsets.train[0]
```

```
('005620.jpg', 'aeroplane')
```

```
def get_x(r): return r['fname']
def get_y(r): return r['labels']
dblock = DataBlock(get_x = get_x, get_y = get_y)
```

```
dsets = dblock.datasets(df)
dsets.train[0]
```

```
('002549.jpg', 'tvmonitor')
```

```
def get_x(r): return path/'train'/r['fname']
def get_y(r): return r['labels'].split(' ')
dblock = DataBlock(get_x = get_x, get_y = get_y)
dsets = dblock.datasets(df)
dsets.train[0]
```

```
(Path('/root/.fastai/data/pascal_2007/train/002844.jpg'), ['train'])
```

```
dblock = DataBlock(blocks=(ImageBlock, MultiCategoryBlock),
                   get_x = get_x, get_y = get_y)
dsets = dblock.datasets(df)
dsets.train[0]
```

```
(PILImage mode=RGB size=500x375,
 TensorMultiCategory([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.]))
```

```
idxs = torch.where(dsets.train[0][1]==1.)[0]
dsets.train.vocab[idxs]
```

```
(#1) ['dog']
```

```
def splitter(df):
    train = df.index[~df['is_valid']].tolist()
    valid = df.index[df['is_valid']].tolist()
    return train,valid

dblock = DataBlock(blocks=(ImageBlock, MultiCategoryBlock),
                   splitter=splitter,
                   get_x=get_x,
                   get_y=get_y)

dsets = dblock.datasets(df)
dsets.train[0]
```

```
(PILImage mode=RGB size=500x333,
 TensorMultiCategory([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]))
```

```
dblock = DataBlock(blocks=(ImageBlock, MultiCategoryBlock),
                   splitter=splitter,
                   get_x=get_x,
                   get_y=get_y,
                   item_tfms = RandomResizedCrop(128, min_scale=0.35))
dls = dblock.dataloaders(df)
```

```
dls.show_batch(nrows=1, ncols=3)
```

Binary Cross-Entropy

```
learn = vision_learner(dls, resnet18)
```

```
/usr/local/lib/python3.9/dist-packages/torchvision/models/_utils.py:208: UserWarni
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/torchvision/models/_utils.py:223: UserWarni
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/
100%                                  44.7M/44.7M [00:00<00:00, 125MB/s]
```

```
x,y = to_cpu(dls.train.one_batch())
activs = learn.model(x)
```

```
activs.shape
```

```
torch.Size([64, 20])
```

```
activs[0]
```

```
TensorBase([ 0.7476, -1.1988,  4.5421, -1.5915, -0.6749,  0.0343, -2.4930, -0.8330, -0.3817, -1.4876, -0.1683,  2.1547, -3.4151,
-1.1743,  0.1530, -1.6801, -2.3067,  0.7063, -1.3358, -0.3715],
           grad_fn=<AliasBackward0>)
```

```
def binary_cross_entropy(inputs, targets):
    inputs = inputs.sigmoid()
    return -torch.where(targets==1, inputs, 1-inputs).log().mean()
```

```
loss_func = nn.BCEWithLogitsLoss()
loss = loss_func(activs, y)
loss
```

```
TensorMultiCategory(1.0342, grad_fn=<AliasBackward0>)
```

```
def say_hello(name, say_what="Hello"): return f"{say_what} {name}."
say_hello('Jeremy'),say_hello('Jeremy', 'Ahoy!')
```

```
('Hello Jeremy.', 'Ahoy! Jeremy.')
```

```
f = partial(say_hello, say_what="Bonjour")
f("Jeremy"),f("Sylvain")
```

```
('Bonjour Jeremy.', 'Bonjour Sylvain.')
```

```
learn = vision_learner(dls, resnet50, metrics=partial(accuracy_multi, thresh=0.2))
learn.fine_tune(3, base_lr=3e-3, freeze_epochs=4)
```

```
/usr/local/lib/python3.9/dist-packages/torchvision/models/_utils.py:223: UserWarni
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/
100%                                      97.8M/97.8M [00:00<00:00, 163MB/s]
```

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.943426	0.692230	0.235896	00:45
1	0.823277	0.564228	0.285199	00:45
2	0.604020	0.199862	0.827908	00:34
3	0.359526	0.124002	0.944323	00:34

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.131472	0.116906	0.944203	00:38
1	0.116399	0.106551	0.951096	00:35
2	0.096168	0.104706	0.951116	00:35

```
learn.metrics = partial(accuracy_multi, thresh=0.1)
learn.validate()
```

```
(#2) [0.10470623522996902,0.9281672835350037]
```

```
learn.metrics = partial(accuracy_multi, thresh=0.99)
learn.validate()
```

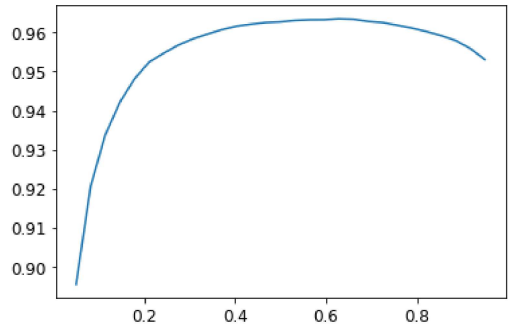
```
(#2) [0.10470623522996902,0.9431872963905334]
```

```
preds,targs = learn.get_preds()
```

```
accuracy_multi(preds, targs, thresh=0.9, sigmoid=False)
```

```
TensorBase(0.9571)
```

```
xs = torch.linspace(0.05,0.95,29)
accs = [accuracy_multi(preds, targs, thresh=i, sigmoid=False) for i in xs]
plt.plot(xs,accs);
```



Regression

Assemble the Data

```
path = untar_data(URLs.BIWI_HEAD_POSE)
```

100.00% [452321280/452316199 00:35<00:00]

```
#hide
Path.BASE_PATH = path
```

```
path.ls().sorted()
```

```
(#50)
[Path('01'),Path('01.obj'),Path('02'),Path('02.obj'),Path('03'),Path('03.obj'),Path('04'),Path('04.obj'),Path('05'),Path('05.obj').
```

```
(path/'01').ls().sorted()
```

```
(#1000)
[Path('01/depth.cal'),Path('01/frame_00003_pose.txt'),Path('01/frame_00003_rgb.jpg'),Path('01/frame_00004_pose.txt'),Path('01/frame
```

```
img_files = get_image_files(path)
def img2pose(x): return Path(f'{str(x)[:-7]}pose.txt')
img2pose(img_files[0])
```

```
Path('05/frame_00566_pose.txt')
```

```
im = PILImage.create(img_files[0])
im.shape
```

```
(480, 640)
```

```
im.to_thumb(160)
```

```
cal = np.genfromtxt(path/'01'/'rgb.cal', skip_footer=6)
def get_ctr(f):
    ctr = np.genfromtxt(img2pose(f), skip_header=3)
    c1 = ctr[0] * cal[0][0]/ctr[2] + cal[0][2]
    c2 = ctr[1] * cal[1][1]/ctr[2] + cal[1][2]
    return tensor([c1,c2])
```

```
get_ctr(img_files[0])
```

```
tensor([346.0781, 267.9450])
```

```
biwi = DataBlock(
    blocks=(ImageBlock, PointBlock),
    get_items=get_image_files,
    get_y=get_ctr,
```

```
    splitter=FuncSplitter(lambda o: o.parent.name=='13'),
    batch_tfms=aug_transforms(size=(240,320)),
)
```

```
dls = biwi.dataloaders(path)
dls.show_batch(max_n=9, figsize=(8,6))
```

```
xb,yb = dls.one_batch()
xb.shape,yb.shape
```

```
(torch.Size([64, 3, 240, 320]), torch.Size([64, 1, 2]))
```

```
yb[0]
```

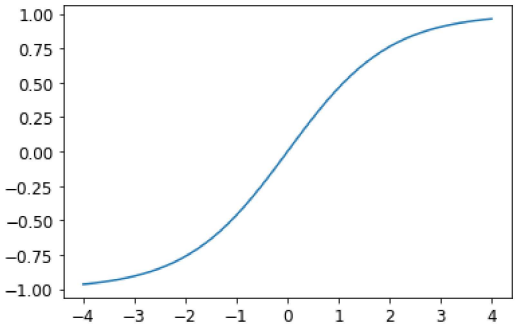
```
TensorPoint([[ 0.1568, -0.0116]], device='cuda:0')
```

Training a Model

```
learn = vision_learner(dls, resnet18, y_range=(-1,1))
```

```
def sigmoid_range(x, lo, hi): return torch.sigmoid(x) * (hi-lo) + lo
```

```
plot_function(partial(sigmoid_range,lo=-1,hi=1), min=-4, max=4)
```




```
dls.loss_func
```

```
FlattenedLoss of MSELoss()
```

```
learn.lr_find()
```

```
SuggestedLRs(valley=0.0010000000474974513)
```


```
lr = 1e-2
learn.fine_tune(3, lr)
```

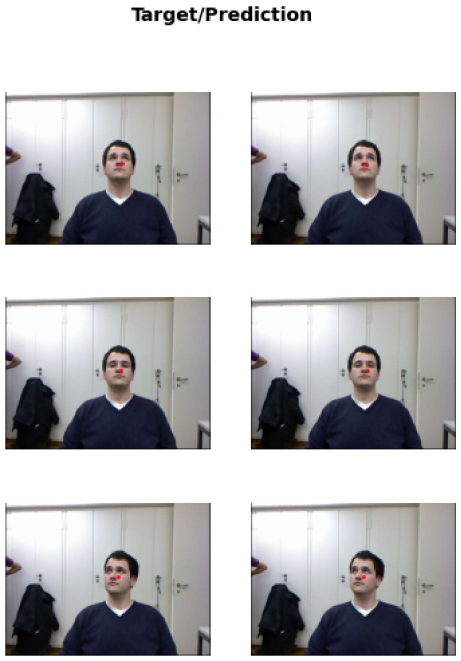
epoch	train_loss	valid_loss	time
0	0.048756	0.013748	02:16

epoch	train_loss	valid_loss	time
0	0.008029	0.002170	02:22
1	0.003059	0.000180	02:23
2	0.001466	0.000081	02:24

```
math.sqrt(0.0001)
```

```
0.01
```

```
learn.show_results(ds_idx=1, nrows=3, figsize=(6,8))
```



Conclusion

Questionnaire

1. How could multi-label classification improve the usability of the bear classifier?
2. How do we encode the dependent variable in a multi-label classification problem?
3. How do you access the rows and columns of a DataFrame as if it was a matrix?
4. How do you get a column by name from a DataFrame?
5. What is the difference between a `Dataset` and `DataLoader`?
6. What does a `Datasets` object normally contain?
7. What does a `DataLoaders` object normally contain?
8. What does `lambda` do in Python?
9. What are the methods to customize how the independent and dependent variables are created with the data block API?
10. Why is softmax not an appropriate output activation function when using a one hot encoded target?
11. Why is `nll_loss` not an appropriate loss function when using a one-hot-encoded target?
12. What is the difference between `nn.BCELoss` and `nn.BCEWithLogitsLoss`?
13. Why can't we use regular accuracy in a multi-label problem?
14. When is it okay to tune a hyperparameter on the validation set?
15. How is `y_range` implemented in fastai? (See if you can implement it yourself and test it without peeking!)
16. What is a regression problem? What loss function should you use for such a problem?

17. What do you need to do to make sure the fastai library applies the same data augmentation to your input images and your target point coordinates?

Further Research

1. Read a tutorial about Pandas DataFrames and experiment with a few methods that look interesting to you. See the book's website for recommended tutorials.
2. Retrain the bear classifier using multi-label classification. See if you can make it work effectively with images that don't contain any bears, including showing that information in the web application. Try an image with two different kinds of bears. Check whether the accuracy on the single-label dataset is impacted using multi-label classification.