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

```
     |████████████████████████████████| 719 kB 31.8 MB/s
     |████████████████████████████████| 452 kB 76.2 MB/s
     |████████████████████████████████| 5.8 MB 63.0 MB/s
     |████████████████████████████████| 1.3 MB 69.8 MB/s
     |████████████████████████████████| 1.6 MB 64.6 MB/s
     |████████████████████████████████| 132 kB 60.2 MB/s
     |████████████████████████████████| 182 kB 26.9 MB/s
     |████████████████████████████████| 213 kB 55.0 MB/s
     |████████████████████████████████| 127 kB 61.6 MB/s
     |████████████████████████████████| 7.6 MB 59.0 MB/s
Mounted at /content/gdrive
```

```
#hide
from fastbook import *
from fastai.vision.widgets import *
```

From Model to Production

The Practice of Deep Learning

Starting Your Project

The State of Deep Learning

Computer vision

Text (natural language processing)

Combining text and images

Tabular data

Recommendation systems

Other data types

The Drivetrain Approach

Gathering Data

clean

To download images with Bing Image Search, sign up at Microsoft Azure for a free account. You will be given a key, which you can copy and enter in a cell as follows (replacing 'XXX' with your key and executing it):

```
key = os.environ.get('AZURE_SEARCH_KEY', '9a4f7f68438548abb18d20dbf45984ac')
```

```
search_images_bing
```

```
<function fastbook.search_images_bing(key, term, min_sz=128, max_images=150)>
```

```
results = search_images_bing(key, 'grizzly bear')
ims = results.attrgot('contentUrl')
```

```
len(ims)
```

```
150
```

```
#hide
ims = ['http://3.bp.blogspot.com/-S1scRCkI3vY/UHzV2kucsPI/AAAAAAAAA-k/YQ5UzHEm9Ss/s1600/Grizzly%2BBear%2BWildlife.jpg']
```

```
dest = 'images/grizzly.jpg'
download_url(ims[0], dest)
```

```
100.99% [704512/697626 00:00<00:00]
```

```
Path('images/grizzly.jpg')
```

```
im = Image.open(dest)
im.to_thumb(128,128)
```

```
bear_types = 'grizzly','black','teddy'
path = Path('bears')
```

```
if not path.exists():
    path.mkdir()
    for o in bear_types:
        dest = (path/o)
        dest.mkdir(exist_ok=True)
        results = search_images_bing(key, f'{o} bear')
        download_images(dest, urls=results.attrgot('contentUrl'))
```

```
fns = get_image_files(path)
fns
```

```
(#426) [Path('bears/black/ea591d15-9926-4d29-bd87-87629f5e9761.jpg'),Path('bears/black/b37949f5-f60d-4039-8d50-
073eb6d89558.jpg'),Path('bears/black/842a909d-4242-4025-a5db-b8c60b47887d.jpg'),Path('bears/black/c59c8b89-7833-4fb7-a70a-
d875eac70306.jpg'),Path('bears/black/b72fb72c-9ecc-4c05-8155-a549977e4603.jpg'),Path('bears/black/9a6369d2-6e03-45b2-b3e5-
f967a5852f8c.jpg'),Path('bears/black/60dd8d32-8d45-463d-8f60-2c242dddef48.jpg'),Path('bears/black/c24d6351-b6f5-432b-bfcb-
687c70478794.jpg'),Path('bears/black/af68d7e8-bde6-49f8-aab4-40af1e346260.jpg'),Path('bears/black/b2bf7dc0-32da-48de-9099-
9d648a08fe7f.jpg')...]
```

```
failed = verify_images(fns)
failed
```

```
(#12) [Path('bears/black/250ef600-58ee-4f34-aa29-6ce466469039.jpg'),Path('bears/teddy/a8e5f85a-ff93-40bf-ab72-
e571b7d79da3.JPG'),Path('bears/teddy/34075e3f-4e89-41f6-b833-0ef5c6ec6a24.JPG'),Path('bears/teddy/c521ff08-2caa-4adb-8be0-
9548dccee80d.jpg'),Path('bears/teddy/84117b64-3c2b-4a78-866b-e25f5b5b5798.jpg'),Path('bears/teddy/0e147ad8-68d2-4f18-aee0-
2a17151c2e2a.jpg'),Path('bears/teddy/33c18420-1761-42a6-9d95-2924e6978ca8.jpg'),Path('bears/teddy/7834a552-7abf-434c-9923-
4cc8aaf4037e.JPG'),Path('bears/grizzly/72c2a039-8444-40dc-abda-eb8308c11eae.jpg'),Path('bears/grizzly/519d7060-6cab-49fb-ac9e-
4c55f46d99a1.PNG')...]
```

```
failed.map(Path.unlink);
```

Sidebar: Getting Help in Jupyter Notebooks

End sidebar

From Data to DataLoaders

```
bears = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2, seed=42),
    get_y=parent_label,
    item_tfms=Resize(128))
```

```
dls = bears.dataloaders(path)
```

```
dls.valid.show_batch(max_n=4, nrows=1)
```

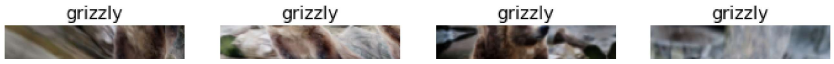

```
bears = bears.new(item_tfms=Resize(128, ResizeMethod.Squish))
dls = bears.dataloaders(path)
dls.valid.show_batch(max_n=4, nrows=1)
```


```
bears = bears.new(item_tfms=Resize(128, ResizeMethod.Pad, pad_mode='zeros'))
dls = bears.dataloaders(path)
dls.valid.show_batch(max_n=4, nrows=1)
```


```
bears = bears.new(item_tfms=RandomResizedCrop(128, min_scale=0.3))
dls = bears.dataloaders(path)
dls.train.show_batch(max_n=4, nrows=1, unique=True)
```

Data Augmentation

```
bears = bears.new(item_tfms=Resize(128), batch_tfms=aug_transforms(mult=2))
dls = bears.dataloaders(path)
dls.train.show_batch(max_n=8, nrows=2, unique=True)
```



Training Your Model, and Using It to Clean Your Data

```
bears = bears.new(
    item_tfms=RandomResizedCrop(224, min_scale=0.5),
    batch_tfms=aug_transforms())
dls = bears.dataloaders(path)
```

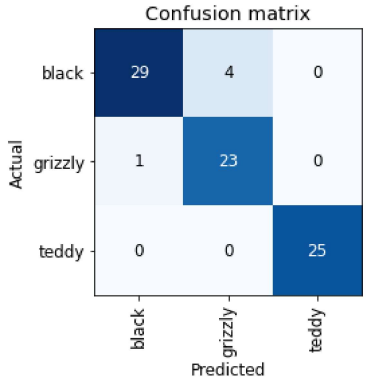
```
learn = vision_learner(dls, resnet18, metrics=error_rate)
learn.fine_tune(4)
```

```
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:208: UserWarni
  warnings.warn(
/usr/local/lib/python3.8/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, 95.5MB/s]
```

epoch	train_loss	valid_loss	error_rate	time
0	1.204868	0.215125	0.073171	00:22

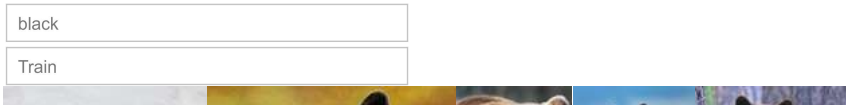
epoch	train_loss	valid_loss	error_rate	time
0	0.214717	0.184590	0.060976	00:30
1	0.174331	0.183639	0.048780	00:21
2	0.151806	0.182633	0.060976	00:21
3	0.127806	0.148523	0.060976	00:21

```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix()
```




```
interp.plot_top_losses(5, nrows=1)
```


```
cleaner = ImageClassifierCleaner(learn)
cleaner
```




```
#hide
# for idx in cleaner.delete(): cleaner.fns[idx].unlink()
# for idx,cat in cleaner.change(): shutil.move(str(cleaner.fns[idx]), path/cat)
```

Turning Your Model into an Online Application

Using the Model for Inference

```
learn.export()
```

```
path = Path()
path.ls(file_exts='.pkl')
```

```
(#1) [Path('export.pkl')]
```

```
learn_inf = load_learner(path/'export.pkl')
```

```
learn_inf.predict('images/grizzly.jpg')
```

```
('grizzly', TensorBase(1), TensorBase([1.2416e-05, 9.9999e-01, 2.1383e-07]))
```

```
learn_inf.dls.vocab
```

```
['black', 'grizzly', 'teddy']
```

Creating a Notebook App from the Model

```
btn_upload = widgets.FileUpload()
btn_upload
```


```
#hide
# For the book, we can't actually click an upload button, so we fake it
btn_upload = SimpleNamespace(data = ['images/grizzly.jpg'])
```

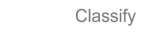
```
img = PILImage.create(btn_upload.data[-1])
```

```
out_pl = widgets.Output()
out_pl.clear_output()
with out_pl: display(img.to_thumb(128,128))
out_pl
```

```
pred,pred_idx,probs = learn_inf.predict(img)
```

```
lbl_pred = widgets.Label()
lbl_pred.value = f'Prediction: {pred}; Probability: {probs[pred_idx]:.04f}'
lbl_pred
```


```
btn_run = widgets.Button(description='Classify')
btn_run
```



```
def on_click_classify(change):
    img = PILImage.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)
    lbl_pred.value = f'Prediction: {pred}; Probability: {probs[pred_idx]:.04f}'

btn_run.on_click(on_click_classify)
```

```
#hide
#Putting back btn_upload to a widget for next cell
btn_upload = widgets.FileUpload()
```

```
VBox([widgets.Label('Select your bear!'),
      btn_upload, btn_run, out_pl, lbl_pred])
```

Select your bear!

Upload (1)

Classify

Prediction: grizzly; Probability: 0.9998

Turning Your Notebook into a Real App

```
#hide
# !pip install voila
# !jupyter serverextension enable --sys-prefix voila
```

Deploying your app

How to Avoid Disaster

Unforeseen Consequences and Feedback Loops

Get Writing!

Questionnaire

1. Provide an example of where the bear classification model might work poorly in production, due to structural or style differences in the training data.
2. Where do text models currently have a major deficiency?
3. What are possible negative societal implications of text generation models?
4. In situations where a model might make mistakes, and those mistakes could be harmful, what is a good alternative to automating a process?
5. What kind of tabular data is deep learning particularly good at?
6. What's a key downside of directly using a deep learning model for recommendation systems?
7. What are the steps of the Drivetrain Approach?
8. How do the steps of the Drivetrain Approach map to a recommendation system?
9. Create an image recognition model using data you curate, and deploy it on the web.
10. What is `DataLoaders`?
11. What four things do we need to tell fastai to create `DataLoaders`?
12. What does the `splitter` parameter to `DataBlock` do?
13. How do we ensure a random split always gives the same validation set?
14. What letters are often used to signify the independent and dependent variables?
15. What's the difference between the crop, pad, and squish resize approaches? When might you choose one over the others?

16. What is data augmentation? Why is it needed?
17. What is the difference between `item_tfms` and `batch_tfms`?
18. What is a confusion matrix?
19. What does `export` save?
20. What is it called when we use a model for getting predictions, instead of training?
21. What are IPython widgets?
22. When might you want to use CPU for deployment? When might GPU be better?
23. What are the downsides of deploying your app to a server, instead of to a client (or edge) device such as a phone or PC?
24. What are three examples of problems that could occur when rolling out a bear warning system in practice?
25. What is "out-of-domain data"?
26. What is "domain shift"?
27. What are the three steps in the deployment process?

Further Research

1. Consider how the Drivetrain Approach maps to a project or problem you're interested in.
2. When might it be best to avoid certain types of data augmentation?
3. For a project you're interested in applying deep learning to, consider the thought experiment "What would happen if it went really, really well?"
4. Start a blog, and write your first blog post. For instance, write about what you think deep learning might be useful for in a domain you're interested in.