```
!pip install -Uqq fastbook
import fastbook
fastbook.setup_book()
```

```
     |████████████████████████████████| 720 kB 5.2 MB/s
     |████████████████████████████████| 48 kB 4.8 MB/s
     |████████████████████████████████| 189 kB 49.4 MB/s
     |████████████████████████████████| 1.2 MB 45.2 MB/s
     |████████████████████████████████| 55 kB 2.6 MB/s
     |████████████████████████████████| 51 kB 322 kB/s
     |████████████████████████████████| 561 kB 48.2 MB/s
     |████████████████████████████████| 130 kB 51.5 MB/s
Mounted at /content/gdrive
```

```
from fastbook import *
```

```
!pwd
```

```
/content/gdrive/MyDrive/ML
```

```
from fastai.vision.all import *
path = Path('/content/gdrive/MyDrive/ML/Dataset')
```

```
train_fnames = get_image_files(path/"Train")
```

```
augs = [RandomResizedCropGPU(size=224, min_scale=0.75), Rotate(), Zoom()]
dblock = DataBlock(blocks=(ImageBlock(cls=PILImage), CategoryBlock),
                   splitter=RandomSplitter(valid_pct=0.2, seed=23),
                   get_y=parent_label,
                   item_tfms=Resize(512, method="squish"),
                   batch_tfms=augs,
                   )
dls = dblock.dataloaders(train_fnames)
```

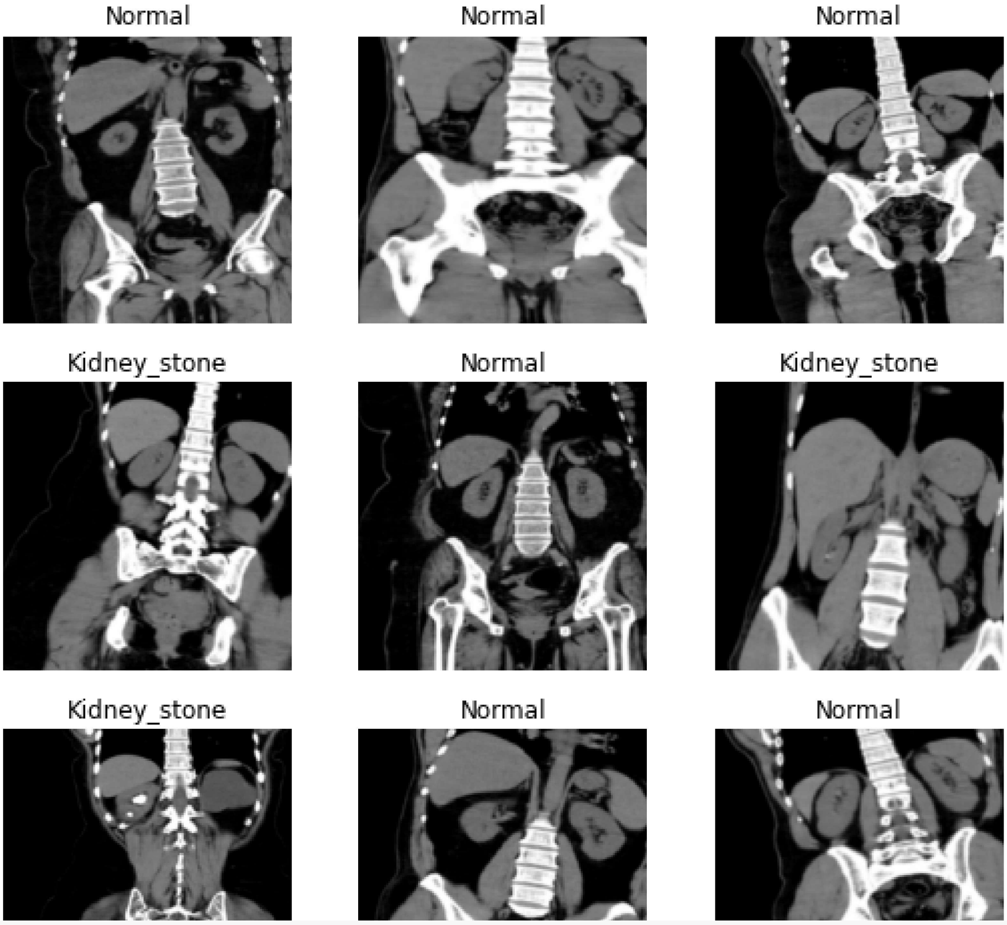
```
# print number of classes and length of train and valid datasets
dls.c, len(dls.train_ds), len(dls.valid_ds)
```

```
(2, 1163, 290)
```

```
dls.vocab
```

```
['Kidney_stone', 'Normal']
```

```
dls.show_batch()
```



Normal Normal Normal

Kidney_stone Normal Kidney_stone

Kidney_stone Normal Normal

```
model = nn.Sequential(create_body(xresnet50, pretrained=False),create_head(nf=2048, n_out=
learn = Learner(dls, model, loss_func=CrossEntropyLossFlat(), metrics=accuracy)
```

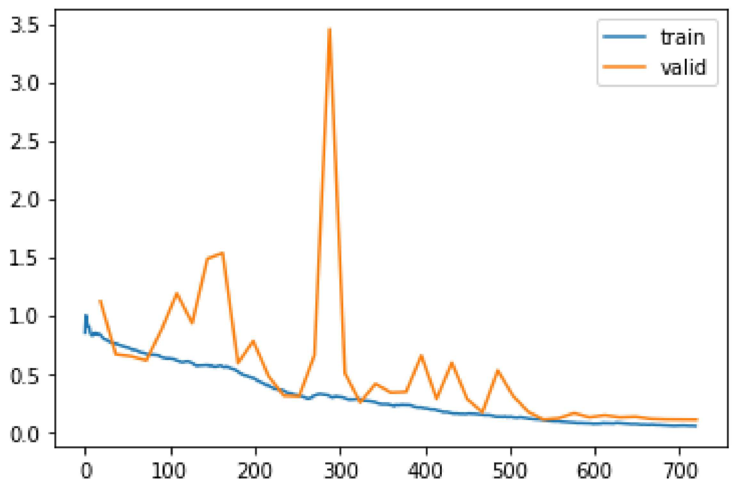
```
learn.fit_one_cycle(40,1e-2)
```

```
                                              85.00% [34/40 38:45<06:50]
epoch  train_loss  valid_loss  accuracy  time
    0    0.833935    1.117885  0.472414  02:27
    1    0.755901    0.664261  0.634483  01:06
    2    0.713859    0.646639  0.658621  01:06
    3    0.670504    0.611010  0.668966  01:06
    4    0.642908    0.876091  0.668966  01:07
    5    0.613907    1.186414  0.555172  01:06
    6    0.592079    0.930579  0.703448  01:06
    7    0.570772    1.483751  0.482759  01:05
    8    0.562933    1.533056  0.558621  01:07
    9    0.521182    0.587817  0.806897  01:06
   10    0.461264    0.777866  0.793103  01:06
   11    0.402013    0.476761  0.834483  01:06
   12    0.353640    0.308439  0.872414  01:05
   13    0.311995    0.303097  0.882759  01:06
   14    0.306595    0.654657  0.758621  01:06
   15    0.308935    3.452801  0.565517  01:06
   16    0.292403    0.501665  0.789655  01:06
   17    0.281374    0.250491  0.906897  01:06
   18    0.255006    0.410930  0.848276  01:05
   19    0.230731    0.335607  0.920690  01:06
   20    0.228009    0.341254  0.917241  01:05
   21    0.209063    0.653068  0.824138  01:05
   22    0.186937    0.281508  0.900000  01:05
   23    0.160952    0.592099  0.865517  01:05
   24    0.151536    0.279620  0.944828  01:06
   25    0.146778    0.165988  0.948276  01:04
   26    0.129728    0.526051  0.872414  01:05
   27    0.125158    0.307634  0.913793  01:05
   28    0.114485    0.169957  0.951724  01:05
   29    0.099520    0.103764  0.962069  01:05
   30    0.090930    0.115620  0.965517  01:05
   31    0.079979    0.159145  0.968966  01:05
   32    0.073621    0.122909    0.972414   01:05

   33    0.074845    0.141113    0.975862   01:05

                                         55.56% [10/18 00:31<00:25 0.0716]
epoch  train_loss  valid_loss  accuracy   time

    0    0.833935    1.117885    0.472414   02:27

    1    0.755901    0.664261    0.634483   01:06

    2    0.713859    0.646639    0.658621   01:06

    3    0.670504    0.611010    0.668966   01:06

    4    0.642908    0.876091    0.668966   01:07

    5    0.613907    1.186414    0.555172   01:06

    6    0.592079    0.930579    0.703448   01:06

    7    0.570772    1.483751    0.482759   01:05

    8    0.562933    1.533056    0.558621   01:07

    9    0.521182    0.587817    0.806897   01:06

   10    0.461264    0.777866    0.793103   01:06

   11    0.402013    0.476761    0.834483   01:06

   12    0.353640    0.308439    0.872414   01:05

   13    0.311995    0.303097    0.882759   01:06

   14    0.306595    0.654657    0.758621   01:06

   15    0.308935    3.452801    0.565517   01:06

   16    0.292403    0.501665    0.789655   01:06

   17    0.281374    0.250491    0.906897   01:06

   18    0.255006    0.410930    0.848276   01:05

   19    0.230731    0.335607    0.920690   01:06

   20    0.228009    0.341254    0.917241   01:05

   21    0.209063    0.653068    0.824138   01:05

   22    0.186937    0.281508    0.900000   01:05

   23    0.160952    0.592099    0.865517   01:05

   24    0.151536    0.279620    0.944828   01:06

   25    0.146778    0.165988    0.948276   01:04
```

```
learn.save(path/'kidney-50')
```

```
Path('/content/gdrive/MyDrive/ML/Dataset/kidney-50.pth')
```

```
learn.recorder.plot_loss(skip_start=0, with_valid=True)
```



```
#test_items = get_image_files(path/"Test")
all_files= get_image_files(path)
```

Evaluating model performance using the unseen test set

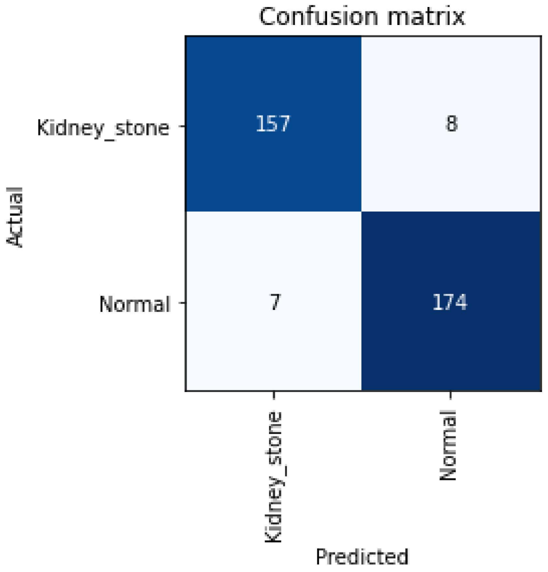
```
augs = [RandomResizedCropGPU(size=224, min_scale=0.75), Rotate(), Zoom()]
dblock = DataBlock(blocks=(ImageBlock(cls=PILImage), CategoryBlock),
                   splitter=GrandparentSplitter(train_name='Train', valid_name='Test'),
                   get_y=parent_label,
                   item_tfms=Resize(512, method="squish"),
                   batch_tfms=augs,
                   )

dls_test = dblock.dataloaders(all_files)
```

```
print("Number of examples in testing:", len(dls_test.valid_ds))
```

```
     Number of examples in testing: 346
```

```
model = nn.Sequential(create_body(xresnet50, pretrained=False),create_head(nf=2048, n_out=
```

```
learn1 = learn1 = Learner(dls_test, model, loss_func=CrossEntropyLossFlat(), metrics=accur
interp = ClassificationInterpretation.from_learner(learn1)
interp.plot_confusion_matrix()
```



```
interp.print_classification_report()
```

```
              precision    recall  f1-score   support

Kidney_stone       0.96      0.95      0.95       165
      Normal       0.96      0.96      0.96       181

    accuracy                           0.96       346
   macro avg       0.96      0.96      0.96       346
weighted avg       0.96      0.96      0.96       346
```

✓ 0s completed at 19:25 ● ×