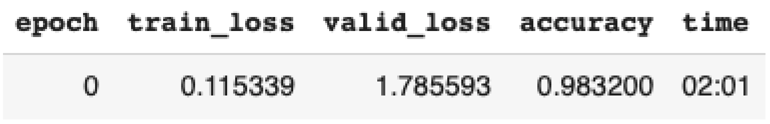
Project objective:

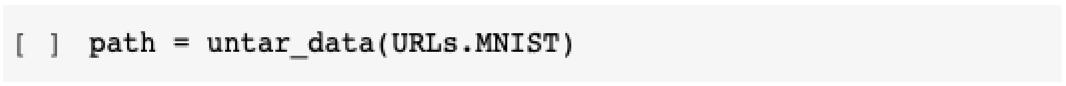
- I trained the resnet 18 network to classify the full MNIST dataset.

- 1 epoch of training produced the following accuracy using testing as the validation set

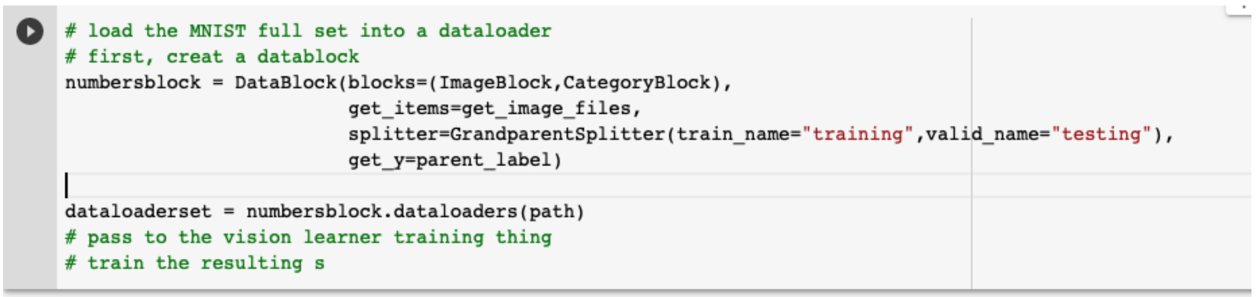


epoch	train_loss	valid_loss	accuracy	time
0	0.115339	1.785593	0.983200	02:01

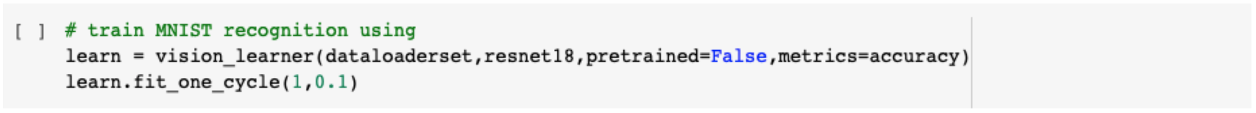
- I used the code from Chapter 2 of the fast.ai textbook to help me set up the training. It was fairly simple to set up. The following is my code to open MNIST and load the data and train the resnet18 model.



```
[ ] path = untar_data(URLs.MNIST)
```



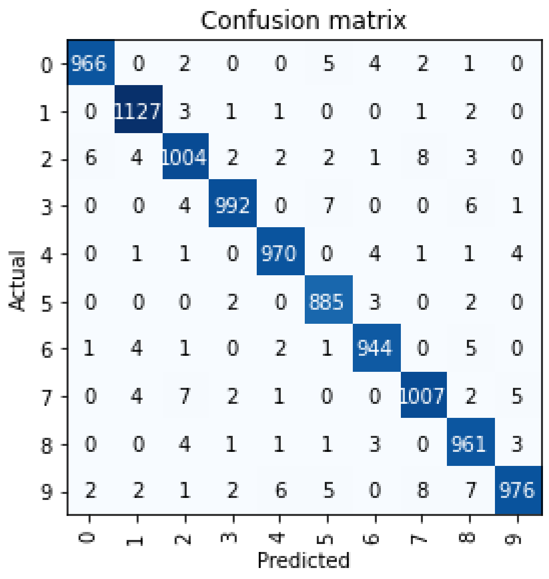
```
# load the MNIST full set into a dataloader
# first, creat a datablock
numbersblock = DataBlock(blocks=(ImageBlock,CategoryBlock),
                         get_items=get_image_files,
                         splitter=GrandparentSplitter(train_name="training",valid_name="testing"),
                         get_y=parent_label)

dataloaderset = numbersblock.dataloaders(path)
# pass to the vision learner training thing
# train the resulting s
```



```
[ ] # train MNIST recognition using
    learn = vision_learner(dataloaderset,resnet18,pretrained=False,metrics=accuracy)
    learn.fit_one_cycle(1,0.1)
```

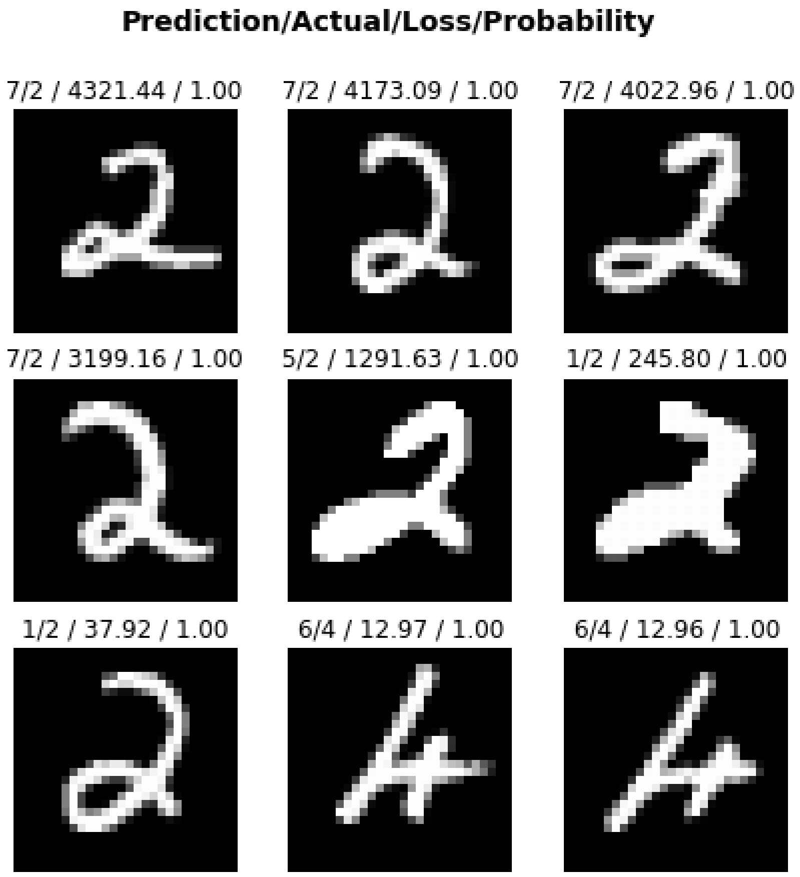
As indicated above, the results were surprisingly accurate with the resnet 18 model. The accuracy is on par with the 784, 300, 10 model used in the MNIST lab, but runs substantially faster.

Here is the confusion matrix:



	0	1	2	3	4	5	6	7	8	9
0	966	0	2	0	0	5	4	2	1	0
1	0	1127	3	1	1	0	0	1	2	0
2	6	4	1004	2	2	2	1	8	3	0
3	0	0	4	992	0	7	0	0	6	1
4	0	1	1	0	970	0	4	1	1	4
5	0	0	0	2	0	885	3	0	2	0
6	1	4	1	0	2	1	944	0	5	0
7	0	4	7	2	1	0	0	1007	2	5
8	0	0	4	1	1	1	3	0	961	3
9	2	2	1	2	6	5	0	8	7	976

Here are the top losses:



Here are the most commonly confused labels, they seem to be plausible – according to the top losses, a two sometimes does look like a 7:

```
[('2', '7', 8),
 ('9', '7', 8),
 ('3', '5', 7),
 ('7', '2', 7),
 ('9', '8', 7),
 ('2', '0', 6),
 ('3', '8', 6),
 ('9', '4', 6),
 ('0', '5', 5),
 ('6', '8', 5),
 ('7', '9', 5),
 ('9', '5', 5),
 ('0', '6', 4),
 ('2', '1', 4),
 ('3', '2', 4),
 ('4', '6', 4),
 ('4', '9', 4),
 ('6', '1', 4),
 ('7', '1', 4),
 ('8', '2', 4),
 ('1', '2', 3),
 ('2', '8', 3),
 ('5', '6', 3),
 ('8', '6', 3),
 ('8', '9', 3),
 ('0', '2', 2),
 ('0', '7', 2),
 ('1', '8', 2),
 ('2', '3', 2),
 ('2', '4', 2),
 ('2', '5', 2),
 ('5', '3', 2),
 ('5', '8', 2),
 ('6', '4', 2),
 ('7', '3', 2),
 ('7', '8', 2),
 ('9', '0', 2),
 ('9', '1', 2),
 ('9', '3', 2),
 ('0', '8', 1),
 ('1', '3', 1),
 ('1', '4', 1),
 ('1', '7', 1),
 ('2', '6', 1),
 ('3', '9', 1),
 ('4', '1', 1),
 ('4', '2', 1),
 ('4', '7', 1),
 ('4', '8', 1),
 ('6', '0', 1),
 ('6', '2', 1),
 ('6', '5', 1),
 ('7', '4', 1),
 ('8', '3', 1),
 ('8', '4', 1),
 ('8', '5', 1),
 ('9', '2', 1)]
```