MYNTRA T-SHIRT CHALLENGE

To do:

1. Split Validation set equally and train the model for higher accuracy - **DONE - No use as the val set has only 20 samples in each class**

1. **Can increase the val set samples for each class using Image Transformation Augmentation**

2. Over-sample the low-sample classes and Under-sample the high-sample classes and train the model for higher accuracy

1. Max samples: 3000
2. Min samples: 1000

3. Human Annotate the training set properly and train the model for higher accuracy

4. Human Annotate the test set (15000 images) and add it as validation set and train the model for higher accuracy (BEST OPTION)

5. Try different architectures like resnet34, resnext50, resnet101-64 and so on and check the accuracies

6. Ensemble these different models and check the accuracy

7. Try a Region-based CNN on the images first (preprocessing) and then train a model on the region of interest only rather on the entire image with the background and everything

70,000 Training Image Links given
Need to scrape these links and build a Train dataset
15,000 Test image links given
Need to scrape these links and build a Test dataset

Dataset Study:

myntra_train_dataset.csv —> Brand, Category, Gender, Color, Link_to_the_image, Sub_category
myntra_test.csv —> Brand, Category, Gender, Color, Link_to_the_image
submission_online.csv —> Useless as it contains the exact content as test.csv

Preliminary Data Exploration:

```
import pandas as pd
import matplotlib.pyplot as plt
myntra = pd.read_csv("myntra_train_dataset.csv")
print(myntra.head())
print(myntra['Sub_category'].value_counts())
```

```
Solid                         22350
Typography                    13114
Striped                       9650
Graphic                               6558
Colourblocked                 2542
Abstract                      2303
Geometric                     2062
People and Places             1703
Floral                        1634
Humour and Comic             1515
Conversational                1370
Superhero                     1213
Biker                         671
Sports                        624
Varsity                       608
Sports and Team Jersey       403
Music                                 358
Self Design                   332
Tie and Dye                   307
Camouflage                    207
Checked                       196
Tribal                        142
Polka Dots                    137
Horizontal Stripes            1
```

In the cloud…

```
>>> data['class'].value_counts()
Solid                  21156
Typography                 12710
Striped                  9349
Graphic                    6176
Colourblocked                2449
Abstract                  2179
Geometric                   1989
People and Places          1588
Floral                 1559
Humour and Comic            1503
Conversational               1346
Superhero                   1200
Biker                     665
Sports                     618
Varsity                    589
Sports and Team Jersey      400
```

```
Music                    345
Self Design                321
Tie and Dye                303
Camouflage                   195
Checked                    193
Tribal                   127
Polka Dots                 126
Horizontal Stripes           1
Name: class, dtype: int64
```

```
print(myntra.describe())
```

```
                Brand Category Gender  Color  \
count  70000         70000         70000  69677
unique          378             1               5            44
top             Puma            Tshirts         Men     Blue
freq            4889            70000           48910  10373

                                                     Link_to_the_image
        Sub_category
count                                       68870                        70000
unique                                      68797                        24
top     http://lacoste.in/media/catalog/product/p/f/pf...        Solid
freq                                          3                       22350
```

INSIGHTS MADE:

24-class Unbalanced Problem
No image links for 1130 images

Is there a correlation between Link_to_the_image column's text keyword to Sub_category????

At first look, there is some correlation. To find out exactly, we need to do a scatter plot on Keywords with Sub_category to match it.

Also, found cross-overs between Checked and Solid, Solid and Typography, and so on…. It is a difficult problem to crack.

First major issue, downloading all the 70K images from the given links and storing it manually in the local machine or a cloud environ like paperspace, aws, azure and so on.

Steps:

1. Download 70K images from the image URLs given in a csv file and make a Train set
2. Download 15K images from the image URLs given in another csv file and make a Test set
3. Create folders with respect to Multi-label classification as per Fast.ai’s course lecture
4. Train the train set images with a pre-trained Convnet
1. Benchmark the model
2. Make the baseline submission at HackerEarth
5. Tweak the models with different hyper parameters, and steps highlighted in Fast.ai’s course
6. Keep submitting improved models

Making a dictionary with Links as Keys and Class Name as Value

```
with open('sample.csv') as csvfile:
   readCSV = csv.reader(csvfile, delimiter=",")
        a = { row[4] : row[5] for row in readCSV }
print(a)
```

Runs

4000	-	10.00 PM
5000	-	10.08 PM
6000	-	10.14 PM
7000	-	10.19 PM
8000	-	10.26 PM
9000	-	10.32 PM
10000	-	10.38 PM
11000	-	10.44 PM
…		
20000	-	11.54 PM
…		
30000	-	12.52 AM
…		
40000	-	1.43 AM
…		
50000	-	2.

Original Train Images —> 70000 images
Downloaded Train Images —> 67163 images
Resized Train Images —> 67087 images
Unresized **76 image** filenames :
cannot reduce image for Colourblocked_36432.jpg

```
cannot reduce image for Colourblocked_5373.jpg
cannot reduce image for Conversational_68095.jpg
cannot reduce image for Graphic_3953.jpg
cannot reduce image for Graphic_5372.jpg
cannot reduce image for Graphic_68122.jpg
cannot reduce image for Graphic_68123.jpg
cannot reduce image for Solid_4060.jpg
cannot reduce image for Solid_41082.jpg
cannot reduce image for Solid_41140.jpg
cannot reduce image for Solid_51473.jpg
cannot reduce image for Solid_51474.jpg
cannot reduce image for Solid_51479.jpg
cannot reduce image for Solid_51480.jpg
cannot reduce image for Solid_51482.jpg
cannot reduce image for Solid_51483.jpg
cannot reduce image for Solid_51484.jpg
cannot reduce image for Solid_51485.jpg
cannot reduce image for Solid_51504.jpg
cannot reduce image for Solid_51509.jpg
cannot reduce image for Solid_51510.jpg
cannot reduce image for Solid_51530.jpg
cannot reduce image for Solid_51531.jpg
cannot reduce image for Solid_51533.jpg
cannot reduce image for Solid_51534.jpg
cannot reduce image for Solid_51535.jpg
cannot reduce image for Solid_51551.jpg
cannot reduce image for Solid_51556.jpg
cannot reduce image for Solid_51569.jpg
cannot reduce image for Solid_51570.jpg
cannot reduce image for Solid_51571.jpg
cannot reduce image for Solid_51572.jpg
cannot reduce image for Solid_51573.jpg
cannot reduce image for Solid_51574.jpg
cannot reduce image for Solid_51575.jpg
cannot reduce image for Solid_51576.jpg
cannot reduce image for Solid_51578.jpg
cannot reduce image for Solid_51579.jpg
cannot reduce image for Solid_51580.jpg
cannot reduce image for Solid_51581.jpg
cannot reduce image for Solid_51582.jpg
cannot reduce image for Solid_51583.jpg
cannot reduce image for Solid_51584.jpg
cannot reduce image for Solid_51585.jpg
cannot reduce image for Solid_51586.jpg
cannot reduce image for Solid_51587.jpg
cannot reduce image for Solid_51620.jpg
cannot reduce image for Solid_63770.jpg
cannot reduce image for Solid_63836.jpg
cannot reduce image for Solid_68103.jpg
cannot reduce image for Solid_68104.jpg
cannot reduce image for Solid_68113.jpg
cannot reduce image for Solid_68124.jpg
cannot reduce image for Solid_68126.jpg
cannot reduce image for Solid_68127.jpg
cannot reduce image for Solid_7420.jpg
cannot reduce image for Solid_8024.jpg
cannot reduce image for Solid_8025.jpg
cannot reduce image for Solid_8026.jpg
cannot reduce image for Sports and Team Jersey_68125.jpg
cannot reduce image for Sports and Team Jersey_68184.jpg
cannot reduce image for Sports_68072.jpg
cannot reduce image for Sports_68128.jpg
cannot reduce image for Sports_68183.jpg
cannot reduce image for Striped_49209.jpg
cannot reduce image for Striped_51481.jpg
cannot reduce image for Striped_51532.jpg
cannot reduce image for Striped_51565.jpg
cannot reduce image for Striped_51577.jpg
cannot reduce image for Striped_51619.jpg
cannot reduce image for Typography_37878.jpg
cannot reduce image for Typography_4018.jpg
cannot reduce image for Typography_68063.jpg
cannot reduce image for Typography_68114.jpg
cannot reduce image for Typography_8023.jpg
```

Original Test Images —> 15000
Downloaded Test Images —> 14732
Resized Test Images —> 14722
Only **10 images** are not resized :
10513.jpg
11104.jpg
1138.jpg
144.jpg
1512.jpg
4318.jpg
5071.jpg
6217.jpg
7573.jpg

8776.jpg

STEPS

See the given dataset —> data_explorer.py

Download images from Imagelinks given —> download.py (Include also testing side in that as well)

Also talk about Multiprocessing Threads Images downloader —> multi_downloader.py

Resize images —> resizeImage.py

Also talk about Multithreads Resizer —> ImageResizerMultiThreads.py

Check for broken image links and undownloaded images

Write a image checker and sort out the 15K test images in order with the same order with the Submission_online.csv file

Compress resized images into an archive

Train images

Test images

Upload local image Archive to paperspace cloud

```
scp -r Archive.zip paperspace@64.62.141.167:/home/paperspace/projects/myntra/train/
scp -r Archive.zip paperspace@64.62.141.167:/home/paperspace/projects/myntra/test/
```

Create labels.csv with id as filename.jpg and class as one of the 24 graphic types —> labels_csv_creator.py

Upload labels.csv to the paperspace cloud

```
scp -r labels.csv paperspace@64.62.141.167:/home/paperspace/projects/myntra/
```

Now the directory structure is like this:

```
— myntra
        labels.csv
        train folder
                train images
        test folder
                test images
```

One more Resizer

```
import Image
import os
import sys

directory = sys.argv[1]

for file_name in os.listdir(directory):
  print("Processing %s" % file_name)
  image = Image.open(os.path.join(directory, file_name))

  x,y = image.size
  new_dimensions = (x/2, y/2)
  output = image.resize(new_dimensions, Image.ANTIALIAS)

  output_file_name = os.path.join(directory, "small_" + file_name)
  output.save(output_file_name, "JPEG", quality = 95)

print("All done")
```

Train Test Validation Creation —> https://github.com/nik-hil/fastai-1/blob/master/courses/homework/TrainValidationDatasetCreation.ipynb

OVERALL STEPS

Import these class

1. resnet34 -> archi
2. ImageClassifierData to classify image
3. ConvLearner
4. tfms_from_model

Test data

1. set data path
2. import matplot lib
3. set path to data
4. get image count
5. show sample image
6. check image size, shape, content

Create CNN and train model

1. Run CNN using resnet34, ImageClassifierData.frompaths, ConvLearner.pretrained
2. now fit the learner
3. use lr = 0.01, epoch=3 => Findout if 3 is better or not #overfit Vs underfit
4. show all classes

5. this gives prediction for validation set. Predictions are in log scale
6. convert log scale to 10 scale using np.argmax
7. Analyze results by plotting images,

- A few correct labels at random
- A few incorrect labels at random
- The most correct labels of each class (ie those with highest probability that are correct)
- The most incorrect labels of each class (ie those with highest probability that are incorrect)
- The most uncertain labels (ie those with probability closest to 0.5).

8. Choosing a learning rate. Use ConvLearner & lr_find()
9. Plot learining rate

Use Data augmentation to improve results

1. Improve results with data augmentation
2. Use, tfms_from_model. Plot few sample augementated images
3. Retrain the model with precompute=False. All layers should be trained
4. Play with cycle_len.Understand " stochastic gradient descent with restarts (SGDR),". Do some plot
5. save the layers in bcolz using learn.save & learn.load

Use simulated annealing to improve lr

1. # Fine-tuning and differential learning rate annealing
2. unfreeze all model layer
3. train a model with variying lr
4. use cycle_mult to varying lr through learn cycle
5. Find accuracy with test time augmentation (TTA),

Analyze results

1. Plot confusion matrix
2. Analyze results by plotting images,

- A few correct labels at random
- A few incorrect labels at random
- The most correct labels of each class (ie those with highest probability that are correct)
- The most incorrect labels of each class (ie those with highest probability that are incorrect)
- The most uncertain labels (ie those with probability closest to 0.5).

1st RUN

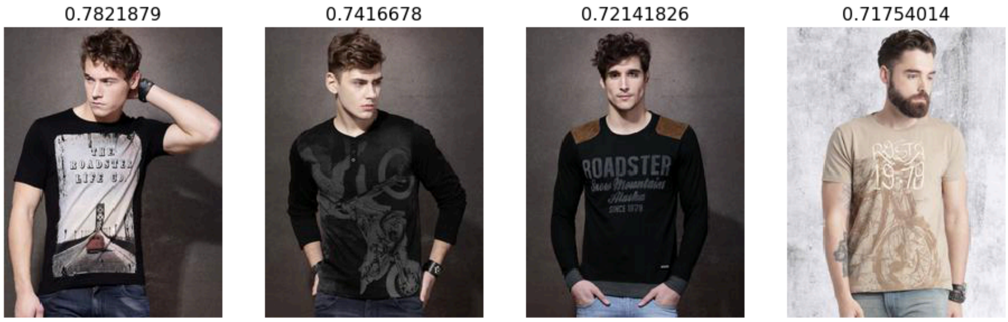
```
arch=resnet34
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms_from_model(arch, sz), bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True)
learn.fit(0.01, 2)
```

```
100%|██████████| 2684/2684 [01:34<00:00, 28.32it/s]

     Epoch ██████████████ 100% 2/2 [00:36<00:00, 18.27s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.391474   1.206224   0.647988
    1      1.259873   1.161199   0.661699
```

```
[1.1611989, 0.6616989546379105]
```

```
i = 1
plot_val_with_title(most_by_correct(i, True), "Most correct " + data.classes[i])
```

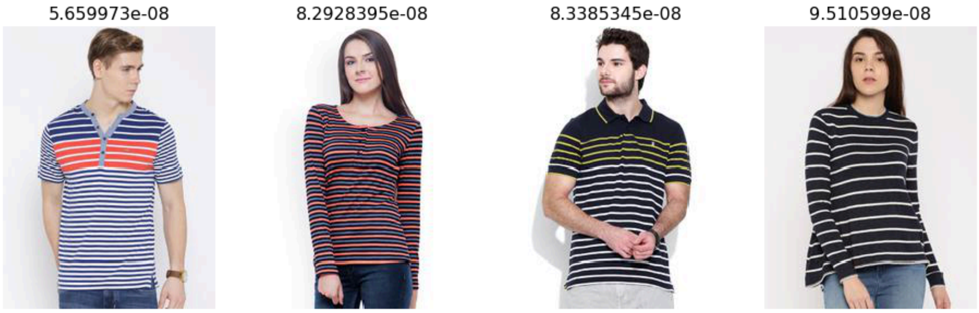
```
Most correct Biker
```



0.7821879 0.7416678 0.72141826 0.71754014

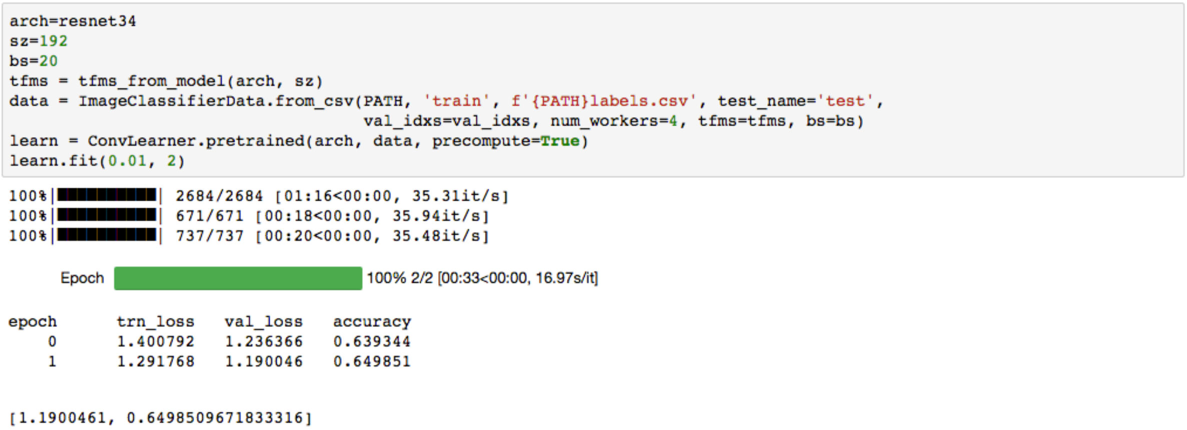
```
i = 18
plot_val_with_title(most_by_correct(i, True), "Most correct " + data.classes[i])
```

```
Most correct Striped
```



5.659973e-08 8.2928395e-08 8.3385345e-08 9.510599e-08

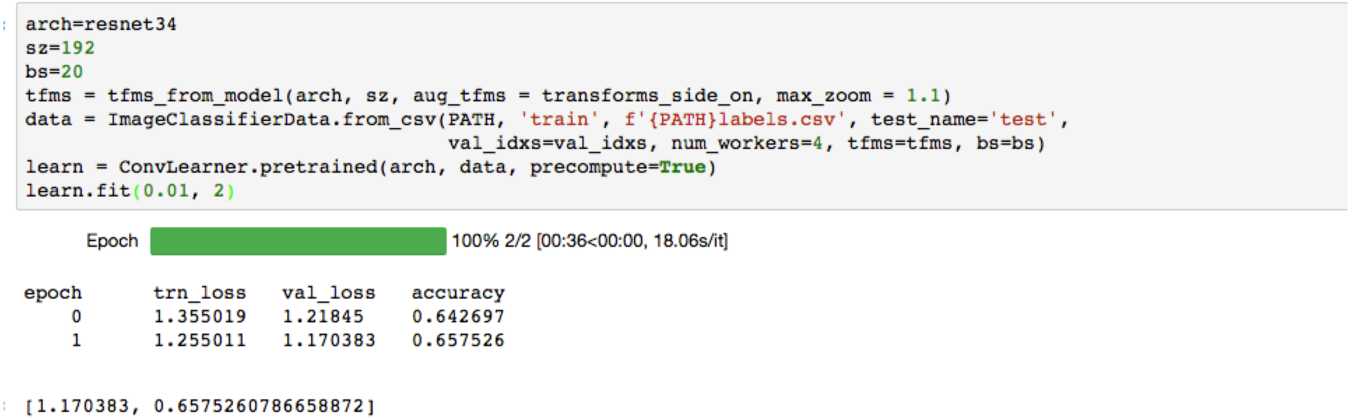
2nd RUN



```
arch=resnet34
sz=192
bs=20
tfms = tfms_from_model(arch, sz)
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms, bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True)
learn.fit(0.01, 2)
```

```
100%|██████████| 2684/2684 [01:16<00:00, 35.31it/s]
100%|██████████| 671/671 [00:18<00:00, 35.94it/s]
100%|██████████| 737/737 [00:20<00:00, 35.48it/s]

     Epoch ████████████ 100% 2/2 [00:33<00:00, 16.97s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.400792   1.236366   0.639344
    1      1.291768   1.190046   0.649851


[1.1900461, 0.6498509671833316]
```

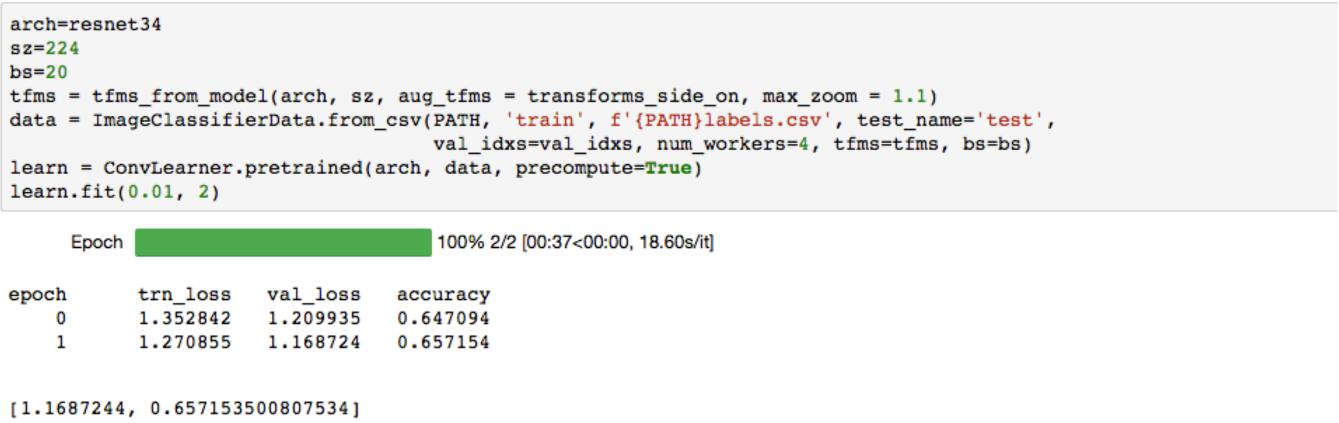
3rd RUN



```
arch=resnet34
sz=192
bs=20
tfms = tfms_from_model(arch, sz, aug_tfms = transforms_side_on, max_zoom = 1.1)
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms, bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True)
learn.fit(0.01, 2)
```

```
     Epoch ████████████ 100% 2/2 [00:36<00:00, 18.06s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.355019   1.21845    0.642697
    1      1.255011   1.170383   0.657526


[1.170383, 0.6575260786658872]
```

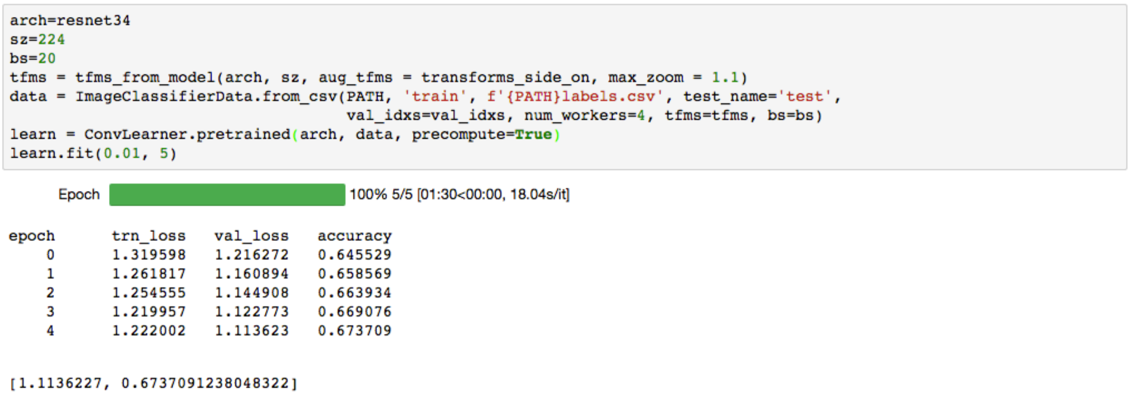
4th RUN



```
arch=resnet34
sz=224
bs=20
tfms = tfms_from_model(arch, sz, aug_tfms = transforms_side_on, max_zoom = 1.1)
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms, bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True)
learn.fit(0.01, 2)
```

```
     Epoch ████████████ 100% 2/2 [00:37<00:00, 18.60s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.352842   1.209935   0.647094
    1      1.270855   1.168724   0.657154


[1.1687244, 0.657153500807534]
```

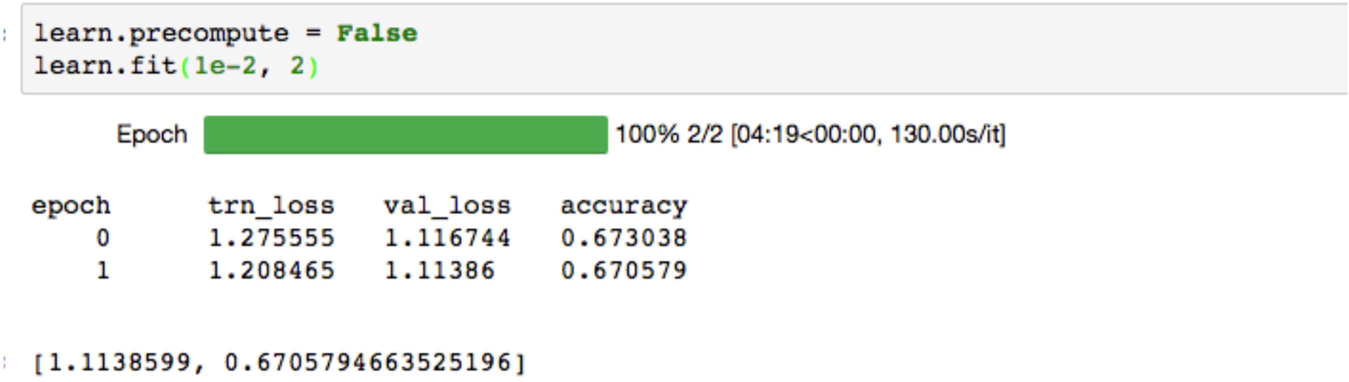
5th RUN



```
arch=resnet34
sz=224
bs=20
tfms = tfms_from_model(arch, sz, aug_tfms = transforms_side_on, max_zoom = 1.1)
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms, bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True)
learn.fit(0.01, 5)
```

```
Epoch 100% 5/5 [01:30<00:00, 18.04s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.319598   1.216272   0.645529
    1      1.261817   1.160894   0.658569
    2      1.254555   1.144908   0.663934
    3      1.219957   1.122773   0.669076
    4      1.222002   1.113623   0.673709

[1.1136227, 0.6737091238048322]
```

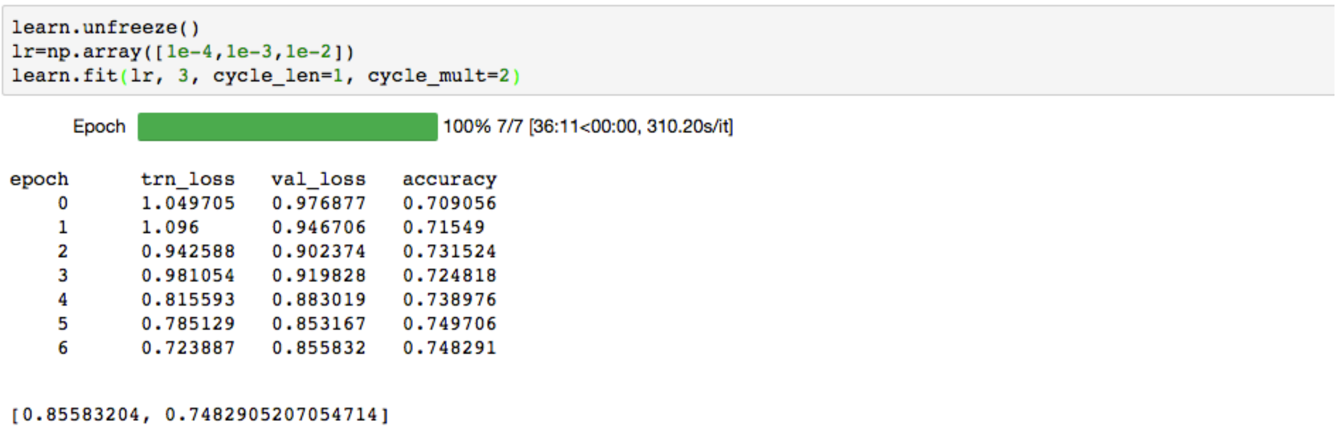
6th RUN



```
learn.precompute = False
learn.fit(1e-2, 2)
```

```
Epoch 100% 2/2 [04:19<00:00, 130.00s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.275555   1.116744   0.673038
    1      1.208465   1.11386    0.670579

[1.1138599, 0.6705794663525196]
```

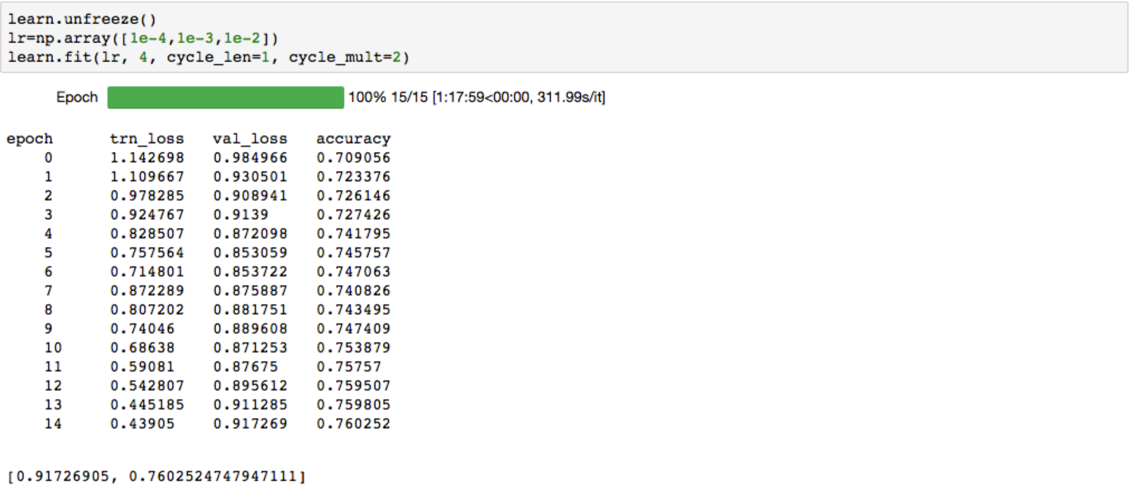
7th RUN



```
learn.unfreeze()
lr=np.array([1e-4,1e-3,1e-2])
learn.fit(lr, 3, cycle_len=1, cycle_mult=2)
```

```
Epoch 100% 7/7 [36:11<00:00, 310.20s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.049705   0.976877   0.709056
    1      1.096      0.946706   0.71549
    2      0.942588   0.902374   0.731524
    3      0.981054   0.919828   0.724818
    4      0.815593   0.883019   0.738976
    5      0.785129   0.853167   0.749706
    6      0.723887   0.855832   0.748291

[0.85583204, 0.7482905207054714]
```

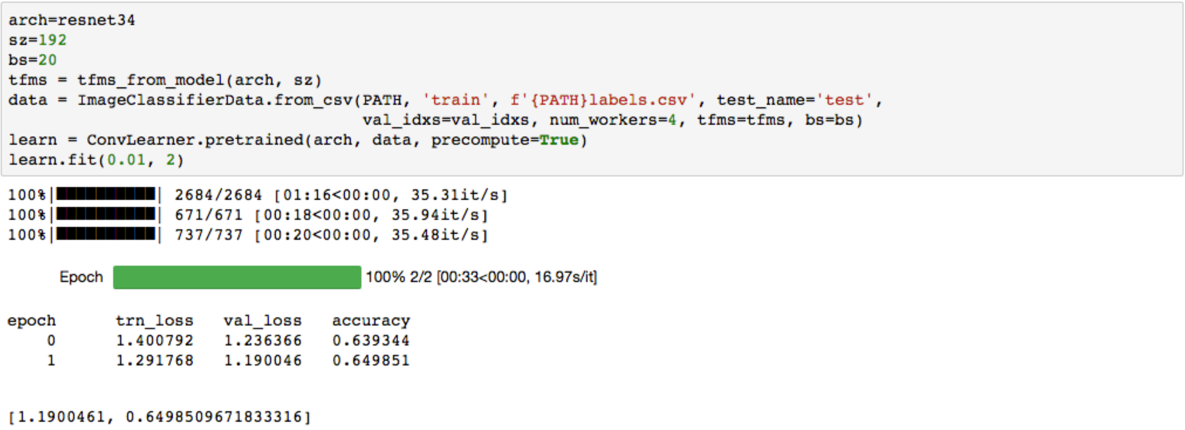
8th RUN with 16 epochs —> Overfitting?



```
learn.unfreeze()
lr=np.array([1e-4,1e-3,1e-2])
learn.fit(lr, 4, cycle_len=1, cycle_mult=2)
```

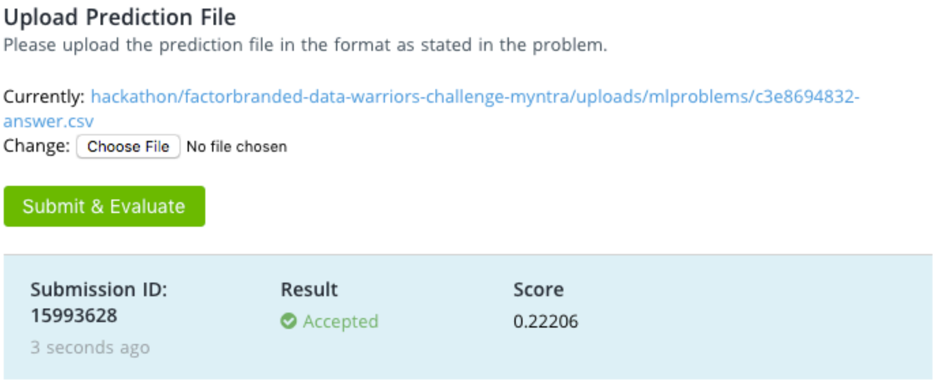
```
    Epoch  100% 15/15 [1:17:59<00:00, 311.99s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.142698   0.984966   0.709056
    1      1.109667   0.930501   0.723376
    2      0.978285   0.908941   0.726146
    3      0.924767   0.9139     0.727426
    4      0.828507   0.872098   0.741795
    5      0.757564   0.853059   0.745757
    6      0.714801   0.853722   0.747063
    7      0.872289   0.875887   0.740826
    8      0.807202   0.881751   0.743495
    9      0.74046    0.889608   0.747409
    10     0.68638    0.871253   0.753879
    11     0.59081    0.87675    0.75757
    12     0.542807   0.895612   0.759507
    13     0.445185   0.911285   0.759805
    14     0.43905    0.917269   0.760252

[0.91726905, 0.7602524747947111]
```

FOR BASELINE MODEL



```
arch=resnet34
sz=192
bs=20
tfms = tfms_from_model(arch, sz)
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms, bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True)
learn.fit(0.01, 2)
```

```
100%|██████████| 2684/2684 [01:16<00:00, 35.31it/s]
100%|██████████| 671/671 [00:18<00:00, 35.94it/s]
100%|██████████| 737/737 [00:20<00:00, 35.48it/s]

    Epoch  100% 2/2 [00:33<00:00, 16.97s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.400792   1.236366   0.639344
    1      1.291768   1.190046   0.649851

[1.1900461, 0.6498509671833316]
```



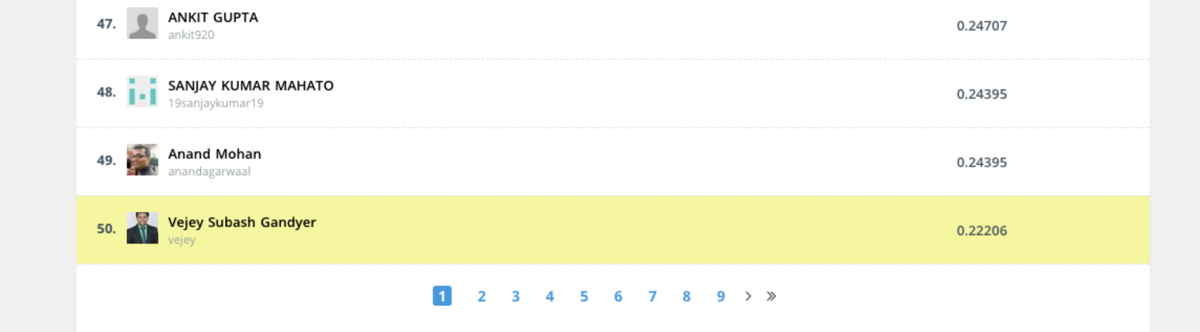
Upload Prediction File

Please upload the prediction file in the format as stated in the problem.

Currently: hackathon/factorbranded-data-warriors-challenge-myntra/uploads/mlproblems/c3e8694832-answer.csv

Change: No file chosen

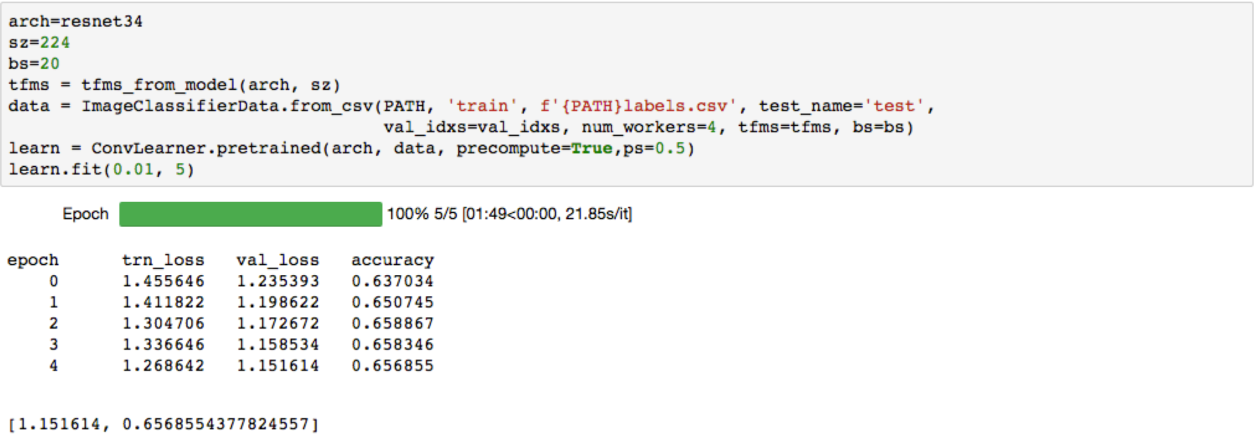
Submission ID:	Result	Score
15993628 3 seconds ago	✓ Accepted	0.22206



47.	ANKIT GUPTA ankit920	0.24707
48.	SANJAY KUMAR MAHATO 19sanjaykumar19	0.24395
49.	Anand Mohan anandagarwaal	0.24395
50.	Vejey Subash Gandyer vejey	0.22206

1 2 3 4 5 6 7 8 9 › »

IMPROVED BASELINE MODEL

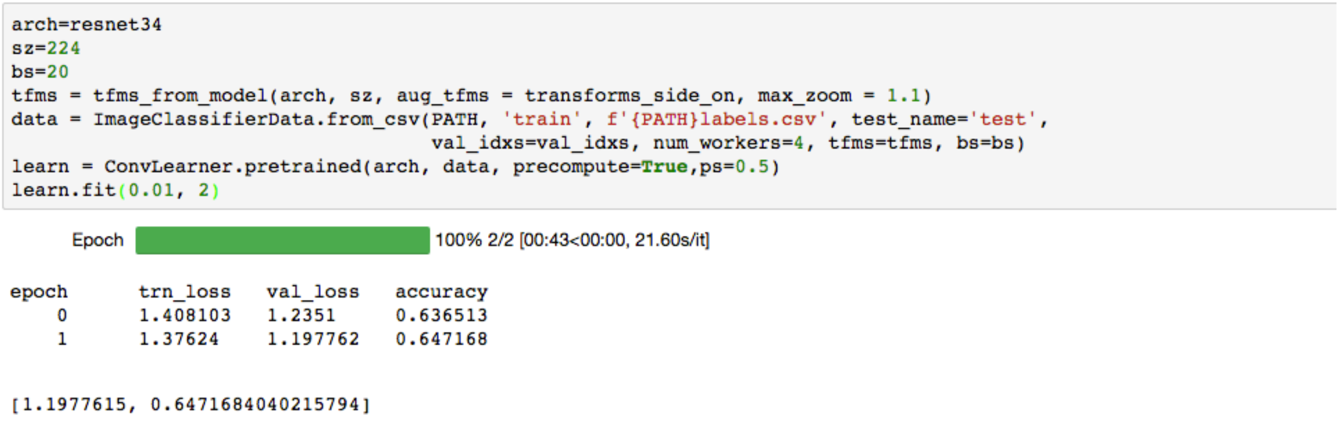


```
arch=resnet34
sz=224
bs=20
tfms = tfms_from_model(arch, sz)
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms, bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True,ps=0.5)
learn.fit(0.01, 5)
```

```
Epoch 100% 5/5 [01:49<00:00, 21.85s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.455646   1.235393   0.637034
    1      1.411822   1.198622   0.650745
    2      1.304706   1.172672   0.658867
    3      1.336646   1.158534   0.658346
    4      1.268642   1.151614   0.656855


[1.151614, 0.6568554377824557]
```

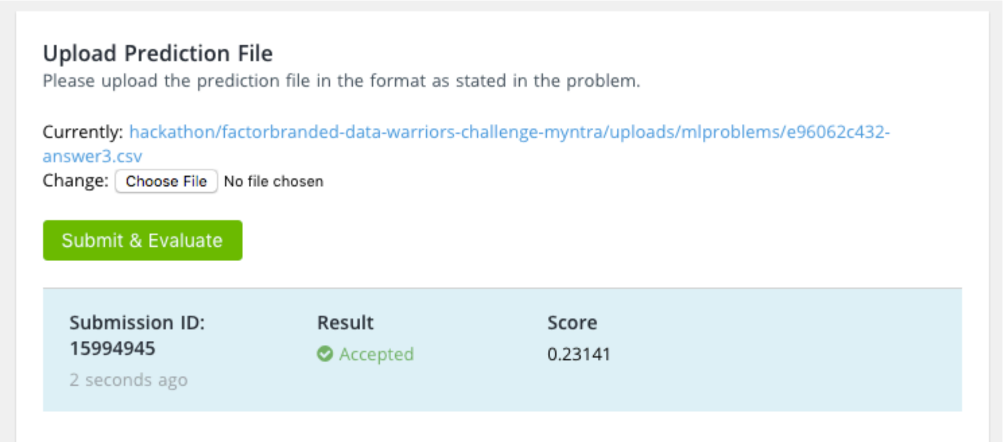
0 8 Graphic
1 15 Solid
2 18 Striped
3 22 Typography
4 15 Solid
5 15 Solid
6 15 Solid
7 8 Graphic
8 18 Striped
9 22 Typography
10 8 Graphic
11 4 Colourblocked
12 22 Typography
13 22 Typography
14 22 Typography
15 18 Striped
16 22 Typography
17 15 Solid
18 8 Graphic
19 18 Striped

IMPROVED AUGMENTED MODEL



```
arch=resnet34
sz=224
bs=20
tfms = tfms_from_model(arch, sz, aug_tfms = transforms_side_on, max_zoom = 1.1)
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms, bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True,ps=0.5)
learn.fit(0.01, 2)
```

```
Epoch 100% 2/2 [00:43<00:00, 21.60s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.408103   1.2351     0.636513
    1      1.37624    1.197762   0.647168

[1.1977615, 0.6471684040215794]
```



Upload Prediction File

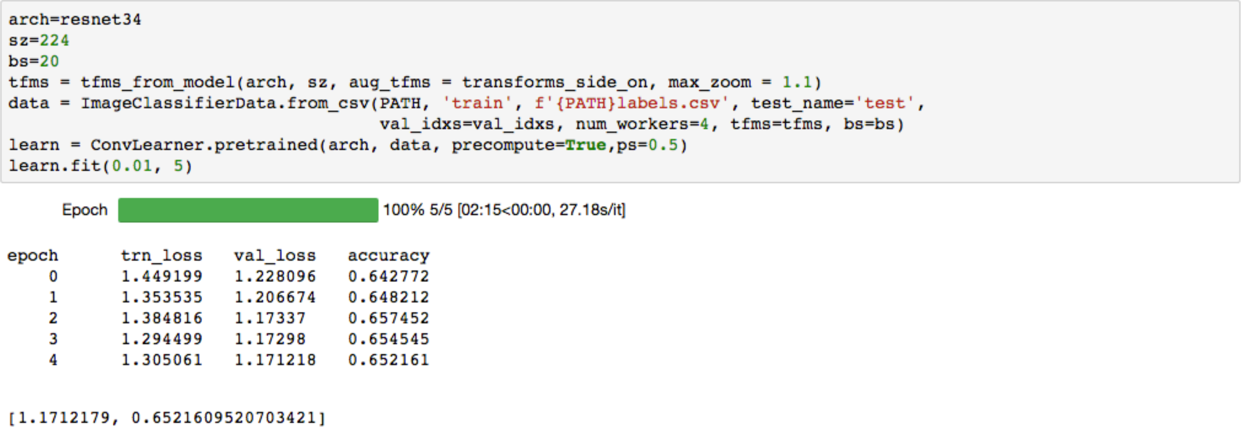
Please upload the prediction file in the format as stated in the problem.

Currently: hackathon/factorbranded-data-warriors-challenge-myntra/uploads/mlproblems/e96062c432-answer3.csv

Change: Choose File No file chosen

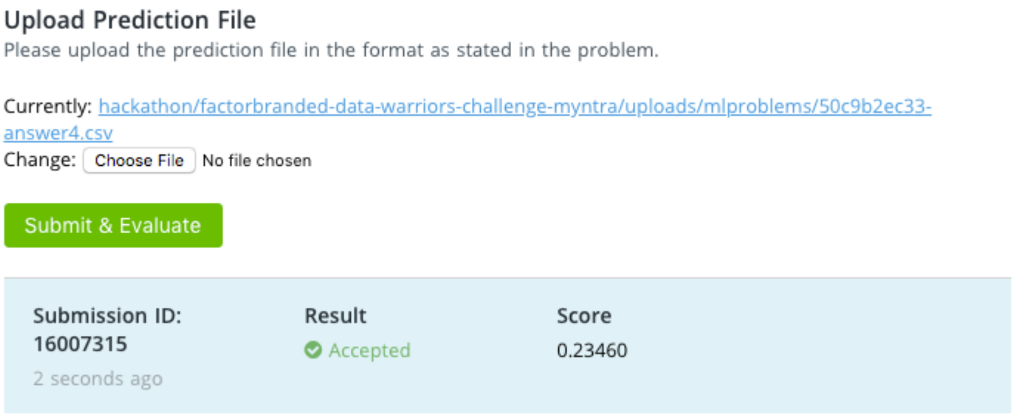
Submission ID:	Result	Score
15994945 2 seconds ago	✓ Accepted	0.23141

IMPROVED AUGMENTED MODEL with 5 Epochs



```
arch=resnet34
sz=224
bs=20
tfms = tfms_from_model(arch, sz, aug_tfms = transforms_side_on, max_zoom = 1.1)
data = ImageClassifierData.from_csv(PATH, 'train', f'{PATH}labels.csv', test_name='test',
                                    val_idxs=val_idxs, num_workers=4, tfms=tfms, bs=bs)
learn = ConvLearner.pretrained(arch, data, precompute=True,ps=0.5)
learn.fit(0.01, 5)
```

```
Epoch 100% 5/5 [02:15<00:00, 27.18s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.449199   1.228096   0.642772
    1      1.353535   1.206674   0.648212
    2      1.384816   1.17337    0.657452
    3      1.294499   1.17298    0.654545
    4      1.305061   1.171218   0.652161

[1.1712179, 0.6521609520703421]
```



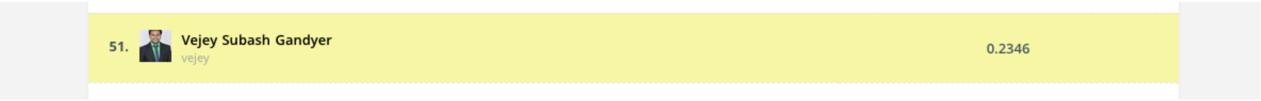
Upload Prediction File

Please upload the prediction file in the format as stated in the problem.

Currently: hackathon/factorbranded-data-warriors-challenge-myntra/uploads/mlproblems/50c9b2ec33-answer4.csv

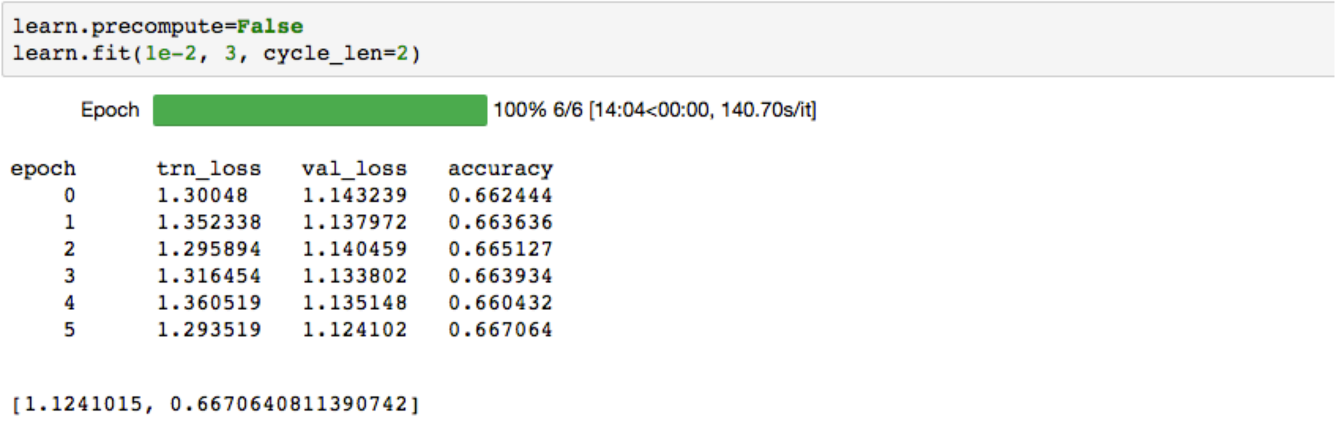
Change: No file chosen

Submission ID:	Result	Score
16007315 2 seconds ago	Accepted	0.23460



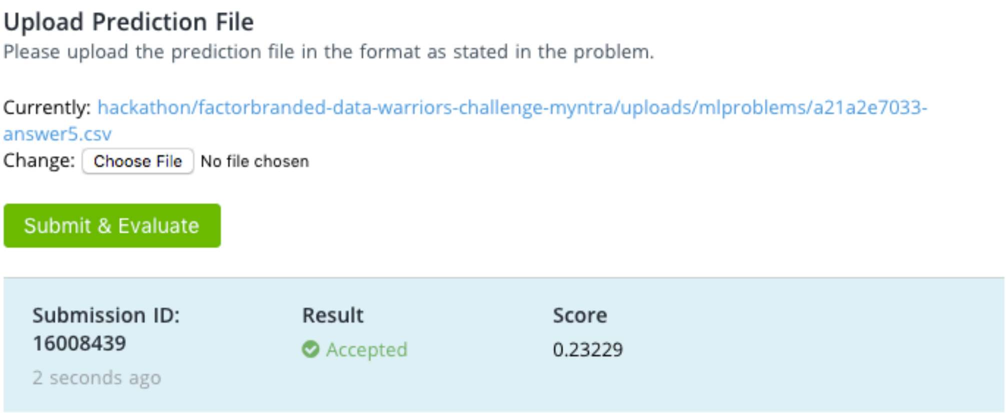
51. Vejey Subash Gandyer vejey 0.2346

IMPROVED AUGMENTED MODEL with 6 Epochs using cyclelen=2



```
learn.precompute=False
learn.fit(1e-2, 3, cycle_len=2)
```

```
Epoch 100% 6/6 [14:04<00:00, 140.70s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.30048    1.143239   0.662444
    1      1.352338   1.137972   0.663636
    2      1.295894   1.140459   0.665127
    3      1.316454   1.133802   0.663934
    4      1.360519   1.135148   0.660432
    5      1.293519   1.124102   0.667064


[1.1241015, 0.6670640811390742]
```



Upload Prediction File

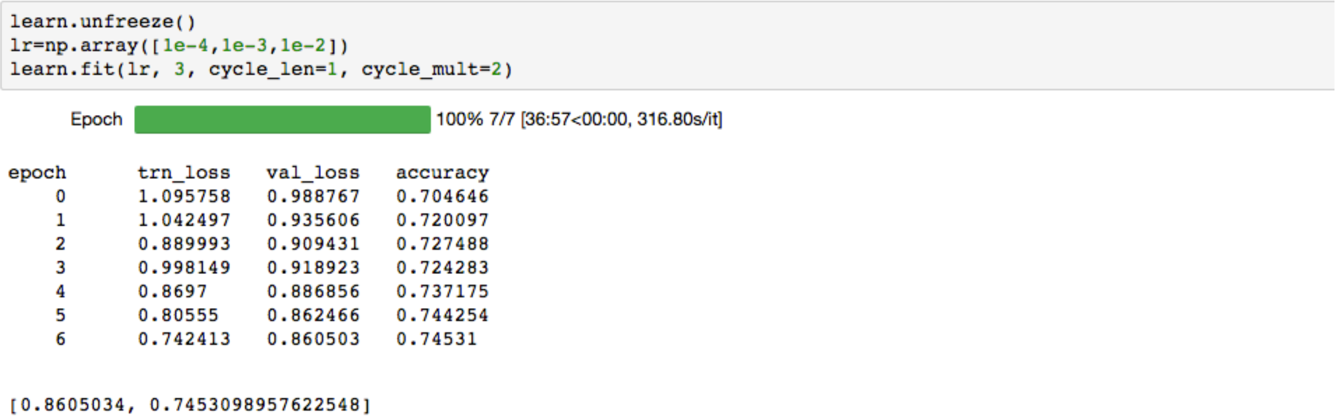
Please upload the prediction file in the format as stated in the problem.

Currently: hackathon/factorbranded-data-warriors-challenge-myntra/uploads/mlproblems/a21a2e7033-answer5.csv

Change: Choose File No file chosen

Submission ID:	Result	Score
16008439 2 seconds ago	✓ Accepted	0.23229

IMPROVED AUGMENTED MODEL with 7 Epochs using cyclelen=1, cyclemult=2 with unfreezing layers and differential lr’s



```
learn.unfreeze()
lr=np.array([1e-4,1e-3,1e-2])
learn.fit(lr, 3, cycle_len=1, cycle_mult=2)
```

```
Epoch                          100% 7/7 [36:57<00:00, 316.80s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.095758   0.988767   0.704646
    1      1.042497   0.935606   0.720097
    2      0.889993   0.909431   0.727488
    3      0.998149   0.918923   0.724283
    4      0.8697     0.886856   0.737175
    5      0.80555    0.862466   0.744254
    6      0.742413   0.860503   0.74531


[0.8605034, 0.7453098957622548]
```

That’s how I reached 30th place in this competition and it was a wonderful journey of putting my DL chops into practice. Learned a lot in the process. Amazing journey! Worth the sleepless nights!!