CAAVO HackerEarth Computer Vision Challenge

Problem Statement: To classify images of the Garment type

APPROACH

- Download the dataset (train, test, sample_submission.csv)
- Explore the dataset through some visualizations to make sure what we are dealing with. As the class labels are not given, they are given as 0 to 14. It makes a lot of sense to go through the images in each folder (0 to 14) to get an idea of what we are dealing with.
- Split the Downloaded train into validation set and store them in a separate valid folder with each class with its class label as a folder under the valid folder.
- Use a pre-trained network of Resnet34 that is trained using Imagenet images.
- I am using Fastai wrapper of a PyTorch Deep Learning framework created by Jamie Howard.
- Build a baseline model with Fully connected layer training and benchmark the score in the leaderboard
- Introduce Image augmentation for over-fitting problems
- Try Unfreezing the previous layers of Resnet34 architecture and train them with differential learning rates for layers
- Experiment changing the size of the images fed, batch size and learning rates

Feature Engineering

As it an image classification problem, nothing much of a feature engineering required here but image augmentation.
I tried Image augmentation with varying zoom parameters and other stuff.
I also split the train data into valid data for feeding into the resnet34 architecture

TOOLS

PyTorch Deep Learning Framework
Fastai wrapper
Pandas
Scikit-learn
Python
Jupyter for Python notebooks
Paperspace Cloud GPU for processing

SOURCE CODE

Importing the libraries

```
%reload_ext autoreload
%autoreload 2
%matplotlib inline

from fastai.imports import *
from fastai.transforms import *
from fastai.conv_learner import *
from fastai.model import *
from fastai.dataset import *
from fastai.sgdr import *
from fastai.plots import *
```

Specifying the data to be used

```
PATH = "/home/paperspace/projects/caavo/with_valid_dataset/"
sz=64
```

Specifying the architecture

```
arch=resnet34
data = ImageClassifierData.from_paths(PATH, tfms=tfms_from_model(arch,
sz),test_name='test')
```

Specifying the learner

```
learn = ConvLearner.pretrained(arch, data, precompute=True)
```

Learning the model

```
learn.fit(0.01, 2)
```

Image Augmentation

```
tfms = tfms_from_model(resnet34, sz, aug_tfms=transforms_side_on,
max_zoom=1.1)

def get_augs():
   data = ImageClassifierData.from_paths(PATH, bs=2, tfms=tfms, num_workers=1)
   x,_ = next(iter(data.aug_dl))
   return data.trn_ds.denorm(x)[1]
```

Loading the data and the learner

```
data = ImageClassifierData.from_paths(PATH, tfms=tfms, test_name='test')
learn = ConvLearner.pretrained(arch, data, precompute=True)
```

Fitting the model

```
learn.fit(1e-2, 1)
```

Make way for augmentation by making the precomputations to False

```
learn.precompute=False
```

Fitting with Cycle length

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

Unfreezing the layers

```
learn.unfreeze()
```

Differential Learning rates for layers

```
lr=np.array([1e-4,1e-3,1e-2])
```

Fitting with Cycle Multiplier

```
learn.fit(lr, 3, cycle_len=1, cycle_mult=2)
```

Storing and loading models

```
learn.save('caavo_1')
learn.load(‘caavo_1’)
```

Making predictions on the test set

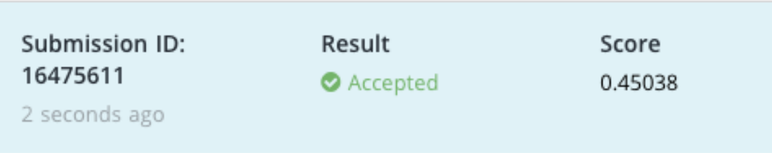
```
log_preds = learn.predict(is_test=True)
log_preds = np.argmax(log_preds,axis=1)
print(log_preds)
```

Getting the predictions for submission

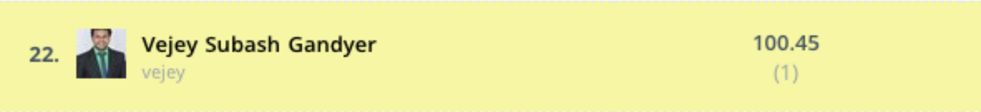
```
for i in range(log_preds.shape[0]):
   arr.append(data.classes[log_preds[i]])
print(arr)
```

Submitting the solution csv file

```
import pandas as pd
df2 = pd.DataFrame(arr)
df2.insert(0,'image_name',[ item[5:] for item in data.test_ds.fnames ] )
df2.columns = ['image_name', 'category']
print(df2.head())
df2.to_csv('ANS1.csv',index=False)
```



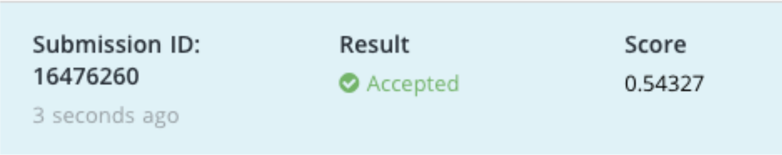
Submission ID:	Result	Score
16475611 2 seconds ago	✓ Accepted	0.45038



22. Vejey Subash Gandyer
vejey
100.45
(1)

Iteration 2 → Rank 20

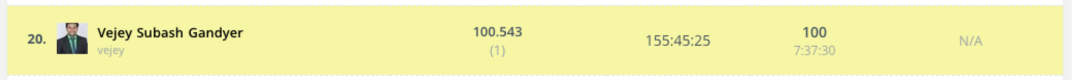
Tried more epochs



Submission ID:
16476260
3 seconds ago

Result
✓ Accepted

Score
0.54327



20. Vejey Subash Gandyer
vejey

100.543
(1)

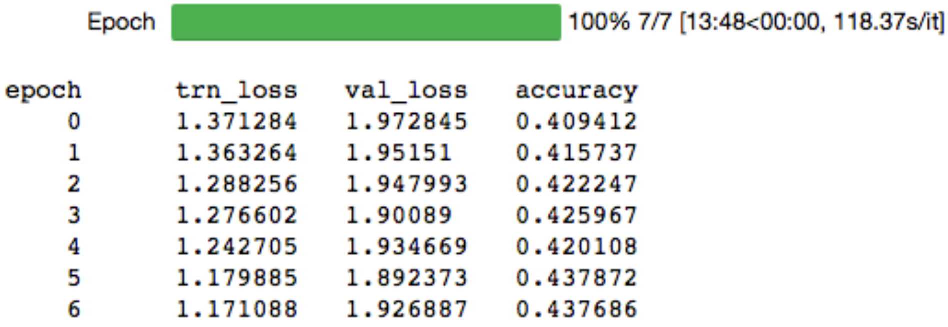
155:45:25

100
7:37:30

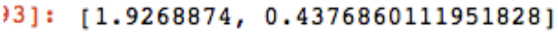
N/A

Iteration 3 → Rank 17

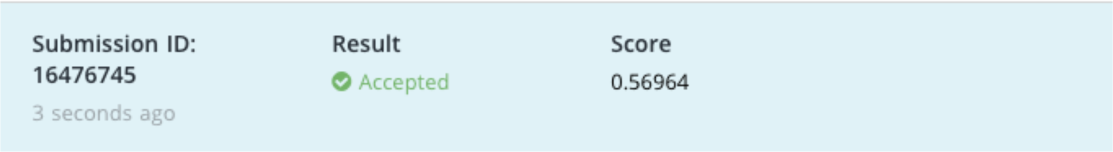
Tried more epochs, learning rates



```
Epoch  100% 7/7 [13:48<00:00, 118.37s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.371284   1.972845   0.409412
    1      1.363264   1.95151    0.415737
    2      1.288256   1.947993   0.422247
    3      1.276602   1.90089    0.425967
    4      1.242705   1.934669   0.420108
    5      1.179885   1.892373   0.437872
    6      1.171088   1.926887   0.437686
```



```
93]:  [1.9268874, 0.4376860111951828]
```



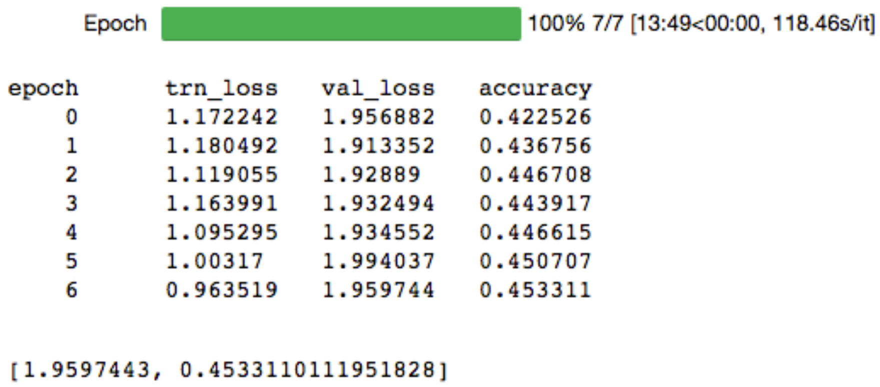
Submission ID:
16476745
3 seconds ago

Result
✓ Accepted

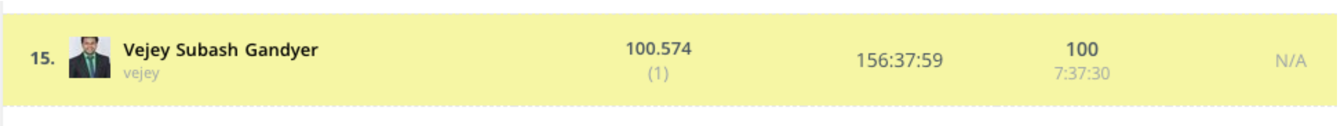
Score
0.56964

Iteration 4 → Rank 15

More epochs, learning rates

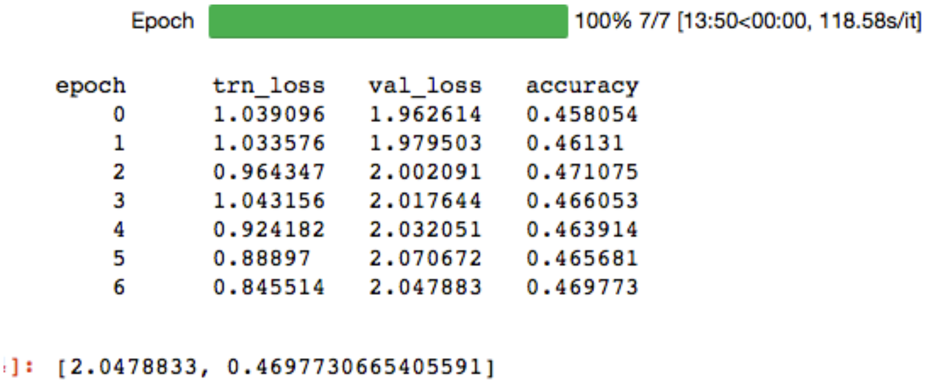


```
Epoch ████████████████ 100% 7/7 [13:49<00:00, 118.46s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.172242   1.956882   0.422526
    1      1.180492   1.913352   0.436756
    2      1.119055   1.92889    0.446708
    3      1.163991   1.932494   0.443917
    4      1.095295   1.934552   0.446615
    5      1.00317    1.994037   0.450707
    6      0.963519   1.959744   0.453311

[1.9597443, 0.4533110111951828]
```

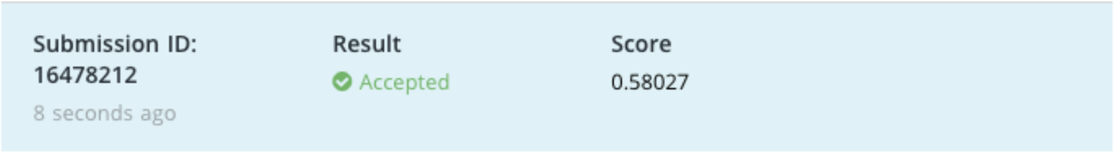


15. Vejey Subash Gandyer
vejey
100.574
(1)
156:37:59
100
7:37:30
N/A

Iteration 5 → Rank 13



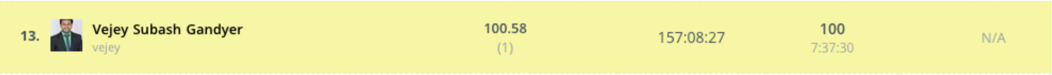
```
Epoch ████████████████ 100% 7/7 [13:50<00:00, 118.58s/it]

epoch      trn_loss   val_loss   accuracy
    0      1.039096   1.962614   0.458054
    1      1.033576   1.979503   0.46131
    2      0.964347   2.002091   0.471075
    3      1.043156   2.017644   0.466053
    4      0.924182   2.032051   0.463914
    5      0.88897    2.070672   0.465681
    6      0.845514   2.047883   0.469773

]: [2.0478833, 0.4697730665405591]
```



Submission ID:
16478212
8 seconds ago

Result
Accepted

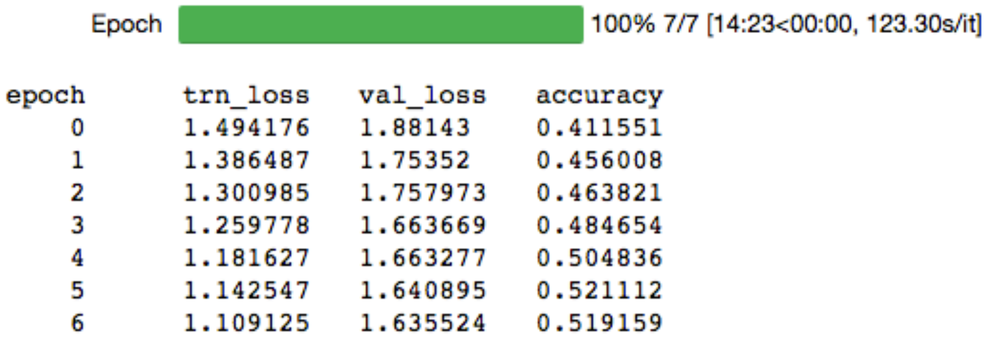
Score
0.58027



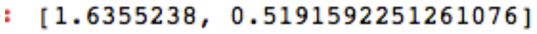
13. Vejey Subash Gandyer
vejey
100.58
(1)
157:08:27
100
7:37:30
N/A

Iteration 6 → Rank 5

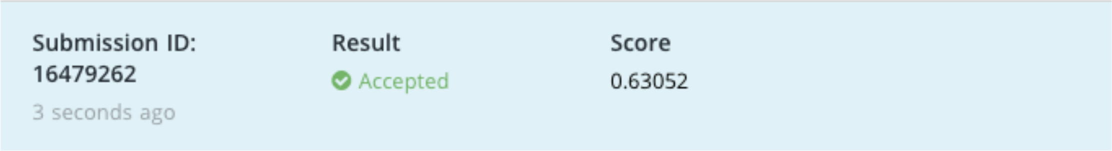
Changing the size of image to 128 x 128 from 64 x 64



```
Epoch 100% 7/7 [14:23<00:00, 123.30s/it]

epoch     trn_loss   val_loss   accuracy
    0     1.494176   1.88143    0.411551
    1     1.386487   1.75352    0.456008
    2     1.300985   1.757973   0.463821
    3     1.259778   1.663669   0.484654
    4     1.181627   1.663277   0.504836
    5     1.142547   1.640895   0.521112
    6     1.109125   1.635524   0.519159
```



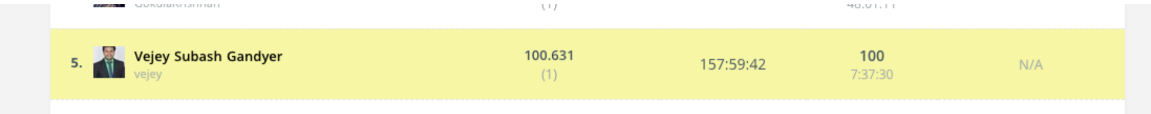
```
: [1.6355238, 0.5191592251261076]
```



Submission ID:
16479262
3 seconds ago

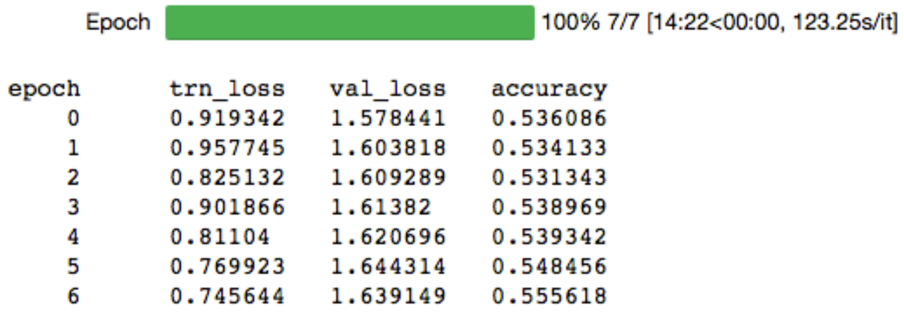
Result
✓ Accepted

Score
0.63052

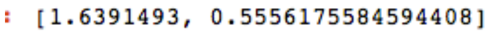


5.	Vejey Subash Gandyer vejey	100.631 (1)	157:59:42	100 7:37:30	N/A

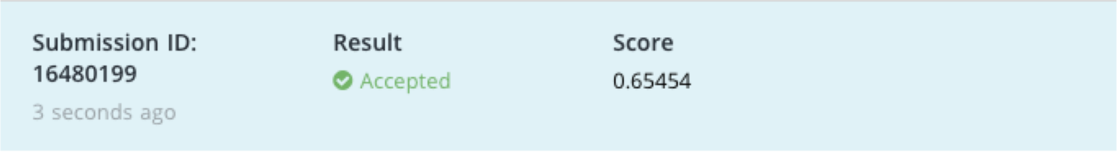
Iteration 7 → Rank 3



```
Epoch 100% 7/7 [14:22<00:00, 123.25s/it]

epoch     trn_loss   val_loss   accuracy
    0     0.919342   1.578441   0.536086
    1     0.957745   1.603818   0.534133
    2     0.825132   1.609289   0.531343
    3     0.901866   1.61382    0.538969
    4     0.81104    1.620696   0.539342
    5     0.769923   1.644314   0.548456
    6     0.745644   1.639149   0.555618
```



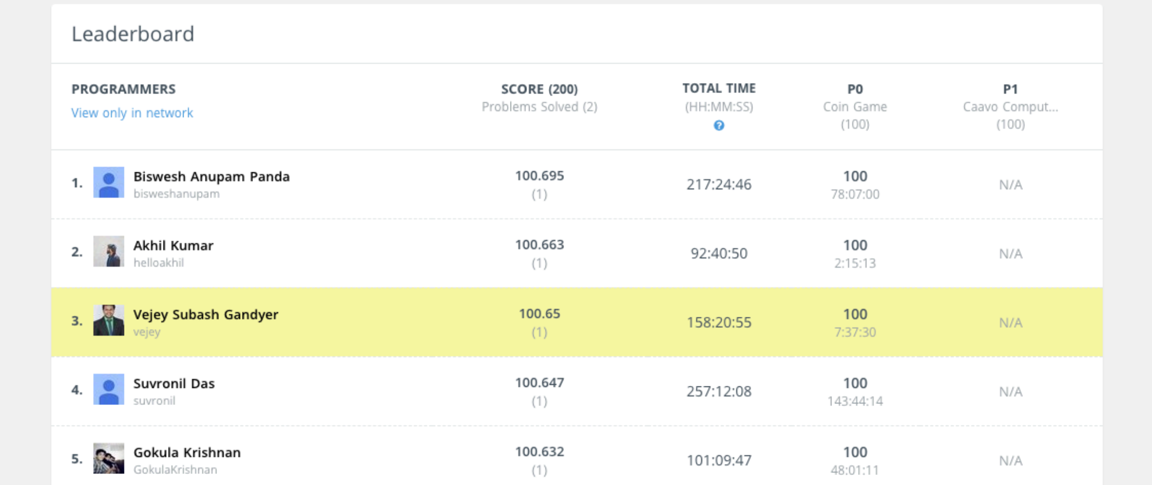
```
: [1.6391493, 0.5556175584594408]
```



Submission ID:
16480199
3 seconds ago

Result
✓ Accepted

Score
0.65454



Leaderboard

	PROGRAMMERS View only in network	SCORE (200) Problems Solved (2)	TOTAL TIME (HH:MM:SS)	P0 Coin Game (100)	P1 Caavo Comput... (100)
1.	Biswesh Anupam Panda bisweshanupam	100.695 (1)	217:24:46	100 78:07:00	N/A
2.	Akhil Kumar helloakhil	100.663 (1)	92:40:50	100 2:15:13	N/A
3.	Vejey Subash Gandyer vejey	100.65 (1)	158:20:55	100 7:37:30	N/A
4.	Suvronil Das suvronil	100.647 (1)	257:12:08	100 143:44:14	N/A
5.	Gokula Krishnan GokulaKrishnan	100.632 (1)	101:09:47	100 48:01:11	N/A

That's the journey I made and reached 3rd place in the Computer Vision Challenge.

Edit: I ended up with 8th place at the end of the competition when I checked the leaderboard 2 days after the competition was over.