Image Classifier and Intelligent Image Retrieval System using Deep Learning

CONTRIBUTORS: SREE GOWRI ADDEPALLI
SREE LAKSHMI ADDEPALLI

Dataset : CIFAR-10
Categories : {'cat','bird','automobile','dog'}

Training Dataset Sample:

Features - Both deep and normal Image Features

id	image	label	deep_features	image_array
24	Height: 32 Width: 32	bird	[0.242871761322, 1.09545373917, 0.0, ...	[73.0, 77.0, 58.0, 71.0, 68.0, 50.0, 77.0, 69.0, ...
33	Height: 32 Width: 32	cat	[0.525087952614, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[7.0, 5.0, 8.0, 7.0, 5.0, 8.0, 5.0, 4.0, 6.0, 7.0, ...
36	Height: 32 Width: 32	cat	[0.566015958786, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[169.0, 122.0, 65.0, 131.0, 108.0, 75.0, ...
70	Height: 32 Width: 32	dog	[1.12979578972, 0.0, 0.0, 0.778194487095, 0.0, ...	[154.0, 179.0, 152.0, 159.0, 183.0, 157.0, ...
90	Height: 32 Width: 32	bird	[1.71786928177, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[216.0, 195.0, 180.0, 201.0, 178.0, 160.0, ...
97	Height: 32 Width: 32	automobile	[1.57818555832, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[33.0, 44.0, 27.0, 29.0, 44.0, 31.0, 32.0, 45.0, ...
107	Height: 32 Width: 32	dog	[0.0, 0.0, 0.220677852631, 0.0, ...	[97.0, 51.0, 31.0, 104.0, 58.0, 38.0, 107.0, 61.0, ...
121	Height: 32 Width: 32	bird	[0.0, 0.23753464222, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[93.0, 96.0, 88.0, 102.0, 106.0, 97.0, 117.0, ...

Count of row values according to categories in training dataset:

Most frequent items from *<SArray>*

Value	Count	Percent
automobile	509	25.387%
dog	509	25.387%
cat	509	25.387%
bird	478	23.84%

Sketch Summary of training dataset:

item	value	is exact
Length	2005	Yes
# Missing Values	0	Yes
# unique values	4	No

Most frequent items:

value	automobile	cat	dog	bird
count	509	509	509	478

Logistic regression classifier (Multi Class Classification) using ‘image’ features

Iteration	Passes	Step size	Elapsed Time	Training-accuracy	Validation-accuracy
1	6	0.000012	4.512087	0.338974	0.394737
2	8	1.000000	6.041114	0.387626	0.447368
3	9	1.000000	6.927861	0.422528	0.429825
4	10	1.000000	7.816536	0.428345	0.447368
5	11	1.000000	8.717247	0.453199	0.473684
6	12	1.000000	9.573985	0.342147	0.412281
7	14	1.000000	11.264866	0.448969	0.500000
8	15	1.000000	12.249029	0.462718	0.517544
9	16	1.000000	13.154255	0.486515	0.526316
10	17	1.000000	14.044666	0.517715	0.535088

Prediction and Accuracy using Logistic Regression:

```
{'accuracy': 0.47325, 'auc': 0.7006268750000003, 'confusion_matrix': Columns:
        target_label    str
        predicted_label str
        count   int

 Rows: 16

 Data:
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     dog      |       cat       |  189  |
|     bird     |       dog       |  169  |
|     cat      |       cat       |  280  |
|  automobile  |       cat       |  107  |
|  automobile  |    automobile   |  588  |
|     bird     |    automobile   |   81  |
|     bird     |       cat       |  125  |
|     cat      |       dog       |  283  |
|     dog      |       dog       |  400  |
|     dog      |    automobile   |   66  |
+--------------+-----------------+-------+
```

Getting the best Classifier using 'image' features:

```
This model may not be optimal. To improve it, consider increasing `max_iterations`.

PROGRESS: Model selection based on validation accuracy:
PROGRESS: ---------------------------------------------
PROGRESS: BoostedTreesClassifier          : 0.483050823212
PROGRESS: RandomForestClassifier          : 0.483050823212
PROGRESS: DecisionTreeClassifier          : 0.423728823662
PROGRESS: LogisticClassifier              : 0.483051
PROGRESS: ---------------------------------------------
PROGRESS: Selecting LogisticClassifier based on validation set performance.
```

Using Boosted Trees Classifier on image features:

Iteration	Elapsed Time	Training-accuracy	Validation-accuracy	Training-log_loss	Validation-log_loss
1	1.545149	0.798742	0.463918	1.127566	1.297634
2	3.072652	0.872642	0.453608	0.941345	1.241777
3	4.574871	0.899895	0.515464	0.811309	1.195418
4	6.101800	0.925577	0.567010	0.704773	1.156680
5	7.669673	0.943396	0.597938	0.601533	1.102639
6	9.151151	0.956499	0.587629	0.525572	1.086875
7	10.726374	0.973795	0.618557	0.449845	1.072370
8	12.269240	0.984277	0.628866	0.388974	1.065300
9	13.699417	0.987421	0.628866	0.350500	1.051302
10	15.176631	0.991614	0.608247	0.308801	1.041245

PREDICTION AND ACCURACY USING BOOSTED TREES CLASSIFIER:

```
{'accuracy': 0.5285, 'auc': 0.773375624999997, 'confusion_matrix': Columns:
        target_label    str
        predicted_label str
        count   int

 Rows: 16

 Data:
```

target_label	predicted_label	count
dog	automobile	95
cat	dog	303
dog	dog	433
automobile	cat	106
automobile	automobile	720
bird	automobile	104
automobile	bird	86
bird	bird	556
bird	dog	151
cat	cat	405

Got 46 % accuracy using Logistic regression and 53% using Boosted trees classifier on test Data.

IMPROVING MULTI CLASS CLASSIFICATION ACCURACY USING DEEP FEATURES:

Using transfer learning: using deep features trained on the full ImageNet dataset

LOGISTIC REGRESSION CLASSIFIER USING DEEP FEATURES

Iteration	Passes	Step size	Elapsed Time	Training-accuracy	Validation-accuracy
1	5	0.000131	4.308244	0.760252	0.834951
2	9	0.250000	8.104835	0.767087	0.864078
3	10	0.250000	9.417259	0.772345	0.873786
4	11	0.250000	10.668947	0.781283	0.883495
5	12	0.250000	12.023254	0.795478	0.883495
6	13	0.250000	13.312408	0.821241	0.873786
7	14	0.250000	14.621989	0.829653	0.864078
8	15	0.250000	15.942435	0.831230	0.873786
9	16	0.250000	17.253108	0.858044	0.864078
10	17	0.250000	18.535019	0.871188	0.873786

TERMINATED: Iteration limit reached.

PREDICTION AND ACCURACY USING DEEP FEATURES WITH LOGISTIC REGRESSION:

```
{'accuracy': 0.7855, 'auc': 0.9403419166666667, 'confusion_matrix': Columns:
        target_label    str
        predicted_label str
        count   int

 Rows: 16

 Data:
```

target_label	predicted_label	count
dog	cat	193
cat	bird	70
bird	dog	72
automobile	bird	19
bird	bird	784
automobile	dog	9
cat	automobile	40
dog	bird	39
bird	cat	115
dog	automobile	18

Accuracy improved from 46% to 78% using Logistic regression.

Getting the best classifier using Deep features:

```
PROGRESS: Model selection based on validation accuracy:
PROGRESS: ---------------------------------------------
PROGRESS: BoostedTreesClassifier          : 0.743119239807
PROGRESS: RandomForestClassifier          : 0.743119239807
PROGRESS: DecisionTreeClassifier          : 0.733944952488
PROGRESS: LogisticClassifier              : 0.788991
PROGRESS: ---------------------------------------------
PROGRESS: Selecting LogisticClassifier based on validation set performance.
```

Using Boosted Trees Classifier using Deep Features:

Iteration	Elapsed Time	Training-accuracy	Validation-accuracy	Training-log_loss	Validation-log_loss
1	1.405870	0.916711	0.657407	0.992044	1.123013
2	2.755728	0.954138	0.740741	0.761517	0.985010
3	4.129766	0.971007	0.731481	0.596606	0.881255
4	5.488433	0.977860	0.722222	0.480316	0.815839
5	6.849249	0.987348	0.750000	0.387638	0.754803
6	8.236676	0.991039	0.731481	0.318542	0.717652
7	9.630469	0.993147	0.722222	0.261695	0.691556
8	10.991722	0.996310	0.750000	0.220069	0.670150
9	12.365204	0.997364	0.750000	0.184992	0.649670
10	13.731441	0.999473	0.740741	0.154338	0.636612

PREDICTION AND ACCURACY USING BOOSTED TREES CLASSIFIER:

```
{'accuracy': 0.7625, 'auc': 0.9280465833333373, 'confusion_matrix': Columns:
        target_label    str
        predicted_label str
        count   int

 Rows: 16

 Data:
 +--------------+-----------------+-------+
 | target_label | predicted_label | count |
 +--------------+-----------------+-------+
 |  automobile  |       cat       |   30  |
 |     bird     |       cat       |  132  |
 |     cat      |       dog       |  231  |
 |     dog      |       dog       |  682  |
 |     dog      |    automobile   |   15  |
 |     cat      |       cat       |  648  |
 |     cat      |    automobile   |   34  |
 |     dog      |       bird      |   68  |
 |  automobile  |       dog       |   3   |
 |     cat      |       bird      |   87  |
 +--------------+-----------------+-------+
```

Using deep features we were able to improve accuracy from 53% to 76%

IMAGE RETRIEVAL AND RECOMMENDATION USING DEEP LEARNING FEATURES USING NEAREST NEIGHBOUR:

TO FIND SIMILAR IMAGES FOR THE FOLLOWING CAT:

All 1 images in <*SArray*>

query_label	reference_label	distance	rank
0	384	0.0	1
0	6910	36.9403137951	2
0	39777	38.4634888975	3
0	36870	39.7559623119	4
0	41734	39.7866014148	5

[5 rows x 4 columns]

RECOMMENDATION FOR CAT:

All 5 images in <*SArray*>

RECOMMENDATION FOR SIMILAR CARS:

All 1 images in <*SArray*>

All 5 images in <*SArray*>

CREATING CATEGORY SPECIFIC MODELS (CAT/ DOG/ CAR /AUTOMOBILE)

```
knn_model_cat = graphlab.nearest_neighbors.create(image_train_cat,features=['deep_features'],
                                                  label='id')
```

```
Starting brute force nearest neighbors model training.
```

```
knn_model_dog = graphlab.nearest_neighbors.create(image_train_dog,features=['deep_features'],
                                                  label='id')
```

```
Starting brute force nearest neighbors model training.
```

```
knn_model_bird = graphlab.nearest_neighbors.create(image_train_bird,features=['deep_features'],
                                                  label='id')
```

```
Starting brute force nearest neighbors model training.
```

```
knn_model_automobile = graphlab.nearest_neighbors.create(image_train_automobile,features=['deep_features'],
                                                  label='id')
```

INPUT IMAGE:

All 1 images in <*SArray*>

NEAREST CAT IMAGES FOR THE INPUT (CAT MODEL):

All 5 images in *<SArray>*

NEAREST DOG IMAGES FOR THE INPUT (DOG MODEL):

All 5 images in *<SArray>*

PERFORMING K NEAREST CLASSIFICATION USING CATEGORIES SPECIFIC MODELS:

CREATING SUBSET OF TEST DATA AND FINDING DISTANCE BETWEEN DIFFERENT MODELS:

```
dog_cat_neighbors = knn_model_cat.query(image_test_dog, k=1)
dog_dog_neighbors = knn_model_dog.query(image_test_dog, k=1)
dog_bird_neighbors = knn_model_bird.query(image_test_dog, k=1)
dog_automobile_neighbors = knn_model_automobile.query(image_test_dog, k=1)
```

```
Starting blockwise querying.
```

CALCULATING USING DOG MODEL:

dog-automobile	dog-bird	dog-cat	dog-dog
41.9579761457	41.7538647304	36.4196077068	33.4773590373
46.0021331807	41.3382958925	38.8353268874	32.8458495684
42.9462290692	38.6157590853	36.9763410854	35.0397073189
41.6866060048	37.0892269954	34.5750072914	33.9010327697
39.2269664935	38.272288694	34.778824791	37.4849250909
40.5845117698	39.1462089236	35.1171578292	34.945165344
45.1067352961	40.523040106	40.6095830913	39.0957278345
41.3221140974	38.1947918393	39.9036867306	37.7696131032
41.8244654995	40.1567131661	38.0674700168	35.1089144603
45.4976929401	45.5597962603	42.7258732951	43.2422832585

[1000 rows x 4 columns]

ACCURACY FOR K NEAREST NEIGHBOUR CLASSIFICATION:

```
total_correct = dog_distance_sframe.apply(is_dog_correct).sum()
print total_correct
```

```
678
```

```
accuracy = total_correct/10
```

```
accuracy
```

```
67
```

FOR CAT MODEL:

cat-automobile	cat-bird	cat-cat	cat-dog
39.6710582792	38.074265869	34.623719208	37.4642628784
43.0089056688	36.3674024138	33.8680579302	29.3472319585
38.6010006604	35.3039394947	32.4615168902	32.2599640475
39.3566307091	38.8944029601	35.7708210254	35.3852085188
38.3572372618	34.2820409875	31.1577686417	30.0442985088
42.0904793181	44.5352170178	41.3986035847	35.4741000424
39.0520251253	34.0290595084	30.9894594959	32.5845275226
39.3058645069	39.0236924983	37.0814607387	37.6502852614
43.0248129799	40.8334054297	39.9883863688	36.9801353512
45.6749176426	40.1258835601	39.7076633097	41.1259410707

[1000 rows x 4 columns]

ACCURACY FOR CAT MODEL:

```
cat_accu_total = cat_distance_sframe.apply(is_cat_correct).sum()
```

```
accu = cat_accu_total/10
```

```
accu
```

```
54
```

REFERENCES:

https://www.analyticsvidhya.com/blog/2016/04/deep-learning-computer-vision-introduction-convolution-neural-networks/

https://www.coursera.org/learn/ml-foundations/home/welcome