Scene Category Classification for TV Series

Abhinandan Dubey
Stony Brook University
New York, USA
adubey@cs.stonybrook.edu

Russell Walker
Stony Brook University
New York, USA
rtwalker@cs.stonybrook.edu

Abstract

Scene Category Classification is one of the key areas of research in Computer Vision. In this project, we use convolutional neural networks to classify a sequence of images, obtained from video clips of a famous TV Series, according to their scene category. We have programmed our model using a computationally efficient averaging technique on frames which gives us a higher accuracy than techniques which do not employ the use of neural networks.

1. Introduction

The field of image classification, especially multi-class classification, has taken major strides in the last half decade or so, thanks to the return of neural networks as a model for feature extraction and classification. Neural networks had fallen out of favor during the early 2000's, but by 2012 progress on accuracy in classification had stalled out between 70-75% (ImageNet Challenge 2012). It was then that a seminal paper on deep convolutional neural networks (Krizhevsky, et al. 2012 [1]) took the field by storm. Their implementation, SuperVision also known as AlexNet, immediately improved the state of the art by nearly 10 points [2] and has been the cornerstone of many results since (their paper has received over 8000 citations in the 4 years since it was published).

Previous methods made use of engineered high level feature detectors, such as SIFT and HoG. A convolutional neural network is a "feature-less" implementation which uses a neural net to learn features and then classify images based on those learned features. CNNs are feed-forward networks which implement a convolutional layer that treats each image as a series of small, locally connected patches.

This reduces the total number of parameters required to a manageable number. These parameters are then fed through many layers of convolution and pooling until the output is created and a classification result is produced. Our implementation makes use of a state-of-the-art very deep CNN implementation (Simonyan & Zisserman 2015 [3]) which has achieved very impressive results.

2. Objective

In this project, our goal was to evaluate image sequence data sourced from video of a prominent TV show, The Big Bang Theory. The data, "Big Bang Theory Video Thread Dataset," consisted of many image frames separated out by shots and threads. From this data we wished to classify the set of images in a thread based on the background scene that they were filmed in front of. There were 13 different classes in total, from Comic Book Store to Living Room to Penny's Bedroom. Our training and test data consisted of images from seasons 2 and 3 of the TV show.

3. Structuring the Huge Dataset

Our dataset needed to be structured before we could use it. A shot is a series of variable number of image frames that runs for an uninterrupted period of time. A thread is a group of some number of shots that is thought to contain the same person or object. For structuring this dataset, we wrote a small MATLAB script to fetch and store all Thread-IDs and Thread-Labels from a text file in a *.mat file so that it is easier for us to perform classification. Since different threads had different number of shots, and each shot had different number of frames, we had to adjust our script accordingly.

4. Approach

After deciding on the techniques to be used in building the learning model and processing the threads, we envisioned a streamlined approach. The first step in our approach was separating the data into a training set and a test set. Season 2 consisted of 3564 threads and season 3 consisted of 3468 threads. Our classifications were 13 multi-class labels based on the background location of each scene in the show.

We designate the first 3000 threads in each season as training data, and the remaining 564 & 468 threads as test data. Next we translated the images into image vectors using the VGG16 network. On the feature vectors, we trained supervised learning model that classifies the resulting feature vectors and evaluated our results. The

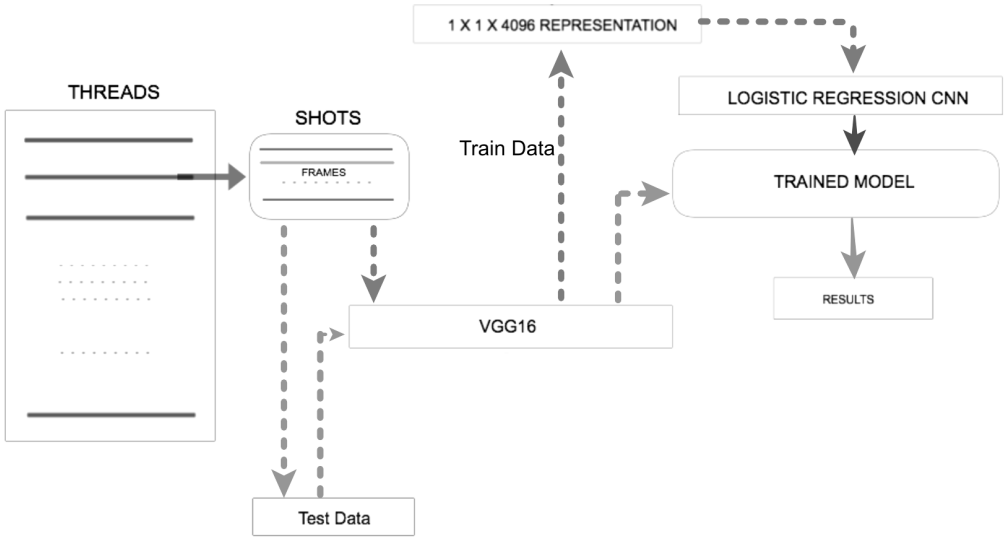

Figure 1: The Procedure

classifier was trained only on the training data and tested only on the test data.

5. Procedure

The first step in our procedure was translating the images into feature vectors recognizable by our classifier. For this purpose, we used the pre-trained CNN VGG16 Very Deep Network provided by the Visual Geometry Group at Oxford (Simonyan & Zisserman). We implemented this network using the MatConvNet in Matlab2016b by VLFeat. The CNN we use is a linear sequence of computational layers.

Our MATLAB script traverses the dataset iteratively by selecting a thread, identifying the sequence of shots and the number of frames in that shot. Then it obtains the VGG16 representations of all frames in the thread by getting cropped and flipped images from each frame. Finally, we feed these representations to our Logistic Regression CNN model which gets trained over 15 epochs. To translate the data into the CNN we iterated through the training data by thread and by frame within each thread. For each image frame we sampled 10 patches of approximately 224x224 pixels from each image. First one patch from each corner and one from the middle, then the image is flipped horizontally and 5 more patches are sampled. Each patch is fed into the CNN which produces an output image feature vector at the layer before the last layer, of dimension 1x1x4096. The ten 4096-dimensional feature vectors are averaged together to produce an overall feature vector for each image. The feature vectors for each image in the thread were then averaged together to produce a single final vector for each thread.

This procedure was repeated for every image in the training set. After these feature vectors were composed, they were fed into a stochastic gradient descent algorithm with momentum which evaluates the total loss incurred when predicting on the training set.

This is a learning process, which means that the network's parameters will be learned through iteration on a training set to improve the results on a separate test set. In our method, we used 15 epochs with a batch size of 100 and a learning rate of 0.001 in our training step. After each epoch gradient descent is used to update the parameters in the direction that reduces loss the most.

Once training is complete, the feature vector extraction procedure outlined above is repeated on the test set. Once the vector for each thread has been found, this data set is run through our algorithm and each vector gets a likelihood number assigned for each class label. We choose the label with the highest likelihood and compare that to the true

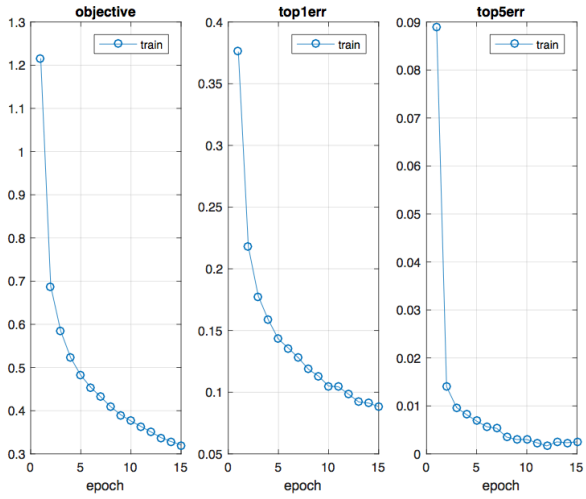

Figure 2: The Top-1 & Top-5 Errors over the Training Dataset

		PREDICTED												
		1	2	3	4	5	6	7	8	9	10	11	12	13
ACTUAL	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	35	0	0	0	0	0	0	0	0	0	0	4
	3	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	2	0	53	0	0	0	3	0	0	0	0	3
	5	0	0	0	0	92	0	0	5	0	0	1	1	11
	6	0	0	0	0	0	21	0	10	0	0	0	0	2
	7	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	1	2	0	0	4	6	0	229	0	0	0	1	33
	9	0	0	0	0	1	0	0	6	7	0	0	0	1
	10	0	0	0	0	0	0	0	0	0	121	0	0	1
	11	0	1	0	0	0	1	0	12	0	0	41	0	11
	12	0	1	0	0	0	0	0	4	0	0	3	6	3
	13	9	0	0	0	8	0	0	45	0	2	0	0	229

Figure 3: The Confusion Matrix

label known from our database. The percentage of threads that were correctly predicted through this method is reported as our accuracy. We use a streamlined procedure to train and test our datasets. After studying some key research papers in the area and exploring what technique could work the best, we identified that we can obtain a considerably good accuracy by averaging the VGG16 representations of different shots and frames in a thread.

Obtaining feature vectors for a single thread, which has multiple shots and each shot having multiple frames takes a lot of time. As shown in Figure 2, we have successfully trained our model with 6000 threads and have tested it over 1032 different threads.

6. Results

Based on the above procedure, our model was trained achieving an accuracy of around 80.8139%. Figure 2 shows the objective, top1err and top5err for our final model trained with 6000 different threads each thread containing some shots, and each shot containing some frames. The model was trained over 15 epochs and 6000 threads with 13 different categories such as “Cafeteria”, “Car”, “Comic Store”, “Hallway”, “Living Room”, “Sheldon’s Room” etc. Our preliminary study shows that simply averaging the representations of different frames in the same shot in a particular thread leads to a very high accuracy score. The Top-1 Error is reduced to less than 0.1 by the last epoch, and Top-5 Error is reduced to almost zero, which is a considerably good performance for a classification model.

Class\Metric	Precision	Recall	F1-Score	Support
1 - Cheese Cake Factory	0.00	0.00	0.00	0
2 - Cafeteria	0.85	0.90	0.88	39
3 - Car	0.00	0.00	0.00	0
4 - Comic Book Store	1.00	0.87	0.93	61
5 - Hallway	0.88	0.84	0.86	110
6 - Kitchen	0.75	0.64	0.69	33
7 - Laundry Room	0.00	0.00	0.00	0
8 - Living Room	0.73	0.83	0.78	276
9 - Leonard's Bedroom	1.00	0.47	0.64	15
10 - Open Theme	0.98	0.99	0.99	122
11 - Penny's Room	0.91	0.62	0.74	66
12 - Sheldon's Bedroom	0.75	0.35	0.48	17
13 - Other	0.77	0.78	0.77	293
Average / Total	0.82	0.81	0.81	1032

Figure 4: Model Evaluation

Figure 4 shows the metrics and scores which briefly summarize our main results. Note that some classes did not have true samples in the test data and therefore Recall and F1-Score were ill-defined and have been set to 0.0.

We have also included the confusion matrix for our test data (Figure 3), which shows where our predicted labels differed from the actual labels and the classes that were erroneously picked instead of the true label. The most common label to be wrongly picked was 8 - Living Room, particularly when the true label was one of the other rooms, such as 6 - Kitchen, 9 - Leonard’s Bedroom, and 11 - Penny’s Room. This can be seen in Table 2 below. This makes sense because different rooms in an apartment will still have some commonalities, especially in comparison to areas like 5 - Hallway or a place of business like 4 - Comic Book Store.

Overall, our average F1-score for all labels in the test data was 0.81, with a Precision of 0.82 and Recall of 0.81. We are also currently exploring other techniques apart from averaging the VGG16 representations of different frames in

the same shots in a particular thread which are not only accurate but also computationally faster.

7. Conclusions

We achieved our results by using a supervised learning method that implemented a deep neural network with training parameters to extract our un-engineered features, followed by a logistic regression classifier which was trained using a stochastic gradient descent algorithm to improve its classification accuracy. Our vector results were combined through a simple averaging for each image and again for each thread of images. Our F1-Score of 0.81 means that we were able to fairly accurately predict the correct class in each thread in our test set.

We conclude that our methods were sound and the algorithms that we implemented are effective at accomplishing the task outlined in the earlier section above. Further work with more extensive training data could improve these results, as well as different methods of combining feature vectors for images within a thread can improve the results drastically. An algorithm that does more sophisticated feature selection to better enhance the differences between similar classes also has strong potential to increase classification accuracy.

8. References

[1] Alex Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks" NIPS 2012: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[2] Xavier Giro, "Deep Learning for Computer Vision: ImageNet Challenge"- Summer Seminar – UPC Telecom Slide 6 - http://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-imagenet-challenge-upc-2016

[3] Karen Simonyan & Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition" ICLR 2015: https://arxiv.org/pdf/1409.1556.pdf

[4] Andrea Vedaldi, Karel Lenc, Ankush Gupta - VLFeat – MatConvNet - Convolutional Neural Networks for MATLAB.