Scene Classification using Scene Parsing, NLP and Neural Networks

Abstract

The recent state of the art scene classification models use Convolutional Neural Networks to perform the scene classification task. To improve the accuracy of these models further, several statistical correlation methods can be applied to the output of these CNN's. We present a few such correlation approaches in this text. We present use of standard TF-IDF and Word Vector and NLP approaches. We also compare them against a Neural Network implementation which gives an accuracy of 82% on the test dataset of [3]. As a part of our model, we obtain object labels of an image using state of the art scene parsing [13]. We experiment with information retrieval methods, word embeddings and Neural networks for mapping the correlation between object labels and scenes for refining the probabilities in scene classification.

1. Introduction

Our project aims to find accurate methods for generating a correlation between objects in a scene and the corresponding scene category, that can be used to refine the accuracy of a scene classification CNN model. State of the art scene classification model has an accuracy of 55.24% on the Places365 data-set[3]. The accuracy can be improved by refining the probabilities corresponding to top-5 scenes by using frequency-based approaches in Natural Language Processing over the contextual information provided by objects in the scene. Additionally, we also check the effectiveness of a neural network for mapping a scene based on the objects in the scene.

2. Background / Related Work

The work by Zhou et. al. [5] introduced the ADE20K dataset, which is a densely annotated dataset with the instances of objects, and their parts, covering a diverse set of visual concepts in the scenes. The dataset was carefully annotated by a single annotator to ensure precise object boundaries within the image and the consistency of object naming across the images. Benchmarks

Architecture	MultiScale Testing	Mean IoU	Pixel Accuracy(%)
MobileNetV2dilated + C1_deepsup	No	34.84	75.75
	Yes	33.84	76.80
MobileNetV2dilated + PPM_deepsup	No	35.76	77.77
	Yes	36.28	78.26
ResNet18dilated + C1_deepsup	No	33.82	76.05
	Yes	35.34	77.41
ResNet18dilated + PPM_deepsup	No	38.00	78.64
	Yes	38.81	79.29
ResNet50dilated + PPM_deepsup	No	41.26	79.73
	Yes	42.14	80.13
ResNet101dilated + PPM_deepsup	No	42.19	80.59
	Yes	42.53	80.91
UperNet50	No	40.44	79.80
	Yes	41.55	80.23
UperNet101	No	42.00	80.79
	Yes	42.66	81.01
HRNetV2	No	42.03	80.77
	Yes	43.20	81.47

Figure 1. Comparison of various Scene parsing modules

for scene parsing and instance segmentation were constructed on the ADE20K dataset[5]. Various state-of-the-art models were evaluated on the benchmark. Figure 1 contains the performance of various models for scene parsing.

Scene parsing uses encoder-decoder architecture that is a staple in semantic segmentation problems.[15]. The model selected uses Resnet[**?**] architecture as the encoder and UperNet[13] as the decoder. The role of the encoder is to generate low-resolution feature maps. Often it is achieved by removing the fully-connected layers from an architecture typically used for classification. The role of the decoder network is to map the low-resolution encoder feature maps to the complete input resolution feature maps for pixel-wise categorization.

Scene-centric datasets contain images that correspond to a scene category, unlike traditional object datasets. Scene15 database [2] was the first benchmark for scene classification, which initially had 8 scenes in its dataset[10]. The dataset contains 15 scene cate-

gories with only a few hundred images in each class, the near-human performance of 95% has been achieved a long time ago on this, with so less number of categories and data, SVM works better than Neural Networks in this case. The MIT Indoor67 database [11] contains 67 categories, which are only Indoor, and no outdoor categories are considered. The Scene Understanding database (SUN) [6] , with 397 categories containing 130,519 images provided a larger coverage of place categories. SUN seems to lack the amount of data needed for using deep learning, and hence deep learning approaches have not worked well on SUN dataset. Other scene-centric datasets like ImageNet-88, SUN88 were obtained by considering a subset of the larger object-centric datasets. "places365" [3] in scale is similar to the massive ImageNet dataset. It has 1,803,460 training images with3,068 to 5,000 images per class. The validation set has 50 images for every scene category furthermore the test set has 900 images for every scene category.

Figure 2 shows the various CNN architectures and their corresponding top-4 and top-1 accuraccy. We have chosen the pre-trained model with Resnet architecture[8], for obtaining scene classification results. Although the model performs better on the training set, the performance on the validation set doesn't increase as much due to overfitting. Other pre-trained models on Places2 do not seem to have similar issues with overfitting, but they do not downsample the images. The overfitting is mainly due to the downsampling of the images and only using the Standard dataset. However, after manually looking at examples of the images in 64x64 resolution and the corresponding predictions the model clearly has a very good performance and hence, despite a minor overfitting issue, it is still one of the best models for scene classification.

Xin Chen et. al. [4] introduced a new method for Scene Classification using a reduced taxonomy for different indoor environments. Although the system was designed for use with robots that can recognize different indoor places with high accuracy, the methodology can be used to improve any scene classification model, which has a lot of context-dependent objects. A hierarchical taxonomy of places allowed pruning many irrelevant classes, thereby reducing the complexity and training time. The pruning of classes is a useful way since many of the classes in the dataset are quite similar. For example there is little use of separate categories "restaurant" and "fastfood restaurant". A word embedding approach was implemented to refine the top-5 scores for the scenes. It is demonstrated that context has the potential to improve the Scene Classification results to some extent. Since the scene labels of the Places 365 test dataset are not publicly available, the methodology was tested with a real-world test dataset that was built by them to show the real world implementation of the approach. The results show that there can be an increase in the accuracy using this approach.

Term frequency-inverse document frequency[1], one of the approaches that we use to determine correlations between objects and scenes, is a statistical weight often used in information retrieval and text mining. This statistical weight measurement is used to evaluate how important a word is to a document in a set. The increase in importance is proportional to the number of times a word appears in the document, but for a lot of redundant words that do not add value can be offset by the frequency of the word in the set.

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. We use GloVe embeddings by Pennington et. al. [7] which is a global log-bilinear regression model for unsupervised learning of word representations that outperforms other models like Word2Vec[14] on word analogy, word similarity, and named entity classification tasks. The commonly used similarity metrics for nearest neighbour evaluations return a single scalar that signifies the correlation between the words. This can often be an oversimplification since two words almost always show more complex relations than those that can be captured by a single number. For instance, a woman may be regarded as similar to a man in that both words describe a human being; but, the two words in common belief are regarded distinct. In order to perceive the refinement necessary to distinguish a woman from a man, we need more than a single number for the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two-word vectors. GloVe captures the meaning specified by the proximity of two words in the form of vector difference. This makes it suitable to use for our use case.

Other approaches like Bag of Visual Words[9], Histogram of Oriented Uniform Patters[12] have been used for obtaining image representations and have obtained good results, but still don't seem to capture the underlying complex relationships as well as GloVe.

3. Approach

Figure 2 explains our project pipeline. The training and test dataset used is MIT CSAIL's Places 365 dataset[3] which has a total of 1.8 million images over 365 unique scene categories. We use the state of the art Scene Parsing model[13] to detect the objects present

	Validation Set of Places365		Test Set of Places365	
	Top-1 acc.	Top-5 acc.	Top-1 acc.	Top-5 acc.
Places365-AlexNet	53.17%	82.89%	53.31%	82.75%
Places365-GoogLeNet	53.63%	83.88%	53.59%	84.01%
Places365-VGG	**55.24%**	84.91%	**55.19%**	85.01%
Places365-ResNet	54.74%	**85.08%**	54.65%	**85.07%**

Figure 2. Comparison of performance of various CNN architectures

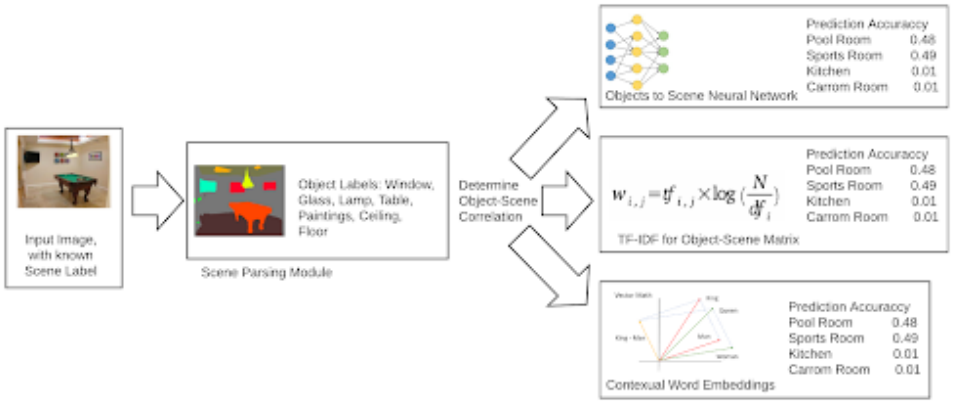

Figure 3. Overview of our Algorithm

in the training and test dataset images. Further we map the correlation between the objects present in each image (obtained from [13]) to the scene labels (obtained from [3]) and create parameter models. The calculation of correlation is done using three distinct approaches. Firstly using TF-IDF method, secondly using word embeddings and finally using a Neural Network. The result of each of these approaches generates a correlation matrix of their own. It has dimensions |O * S|, where O is the maximum number of objects the scene parsing model can detect, and S is the number of possible scenes. Each element of this matrix W[Obj.i, Scene j] signifies the confidence of an object i present to the scene j being classified. During inference, we perform scene parsing on the image to generate object labels, and using the parameter matrix, find the scores of each of the scene classes. The class with maximum score is considered to be the prediction of the model.

4. Experiments

A large part of the work was obtaining a large amount of object labels corresponding to a scene category. We wrote the inferencing code for scene parsing that would create a dataset that can be used as an input for our approaches. The optimal hardware platform of choice after some experimenting was found to be Nvidia T4, which was cost-effective as well could inference one image in 2-3 seconds. The scene classification challenge contains 5000 images for 365 classes. Since each image took 3 seconds for inferencing, obtaining object labels for 365 x 5000 images needed both more compute resources as well as a lot of time. For the sake of our problem, we consider 10 classes out of the 365 classes. We consider Auditorium, Bedroom, Coffee shop, Gas station, Kitchen, Pond, Railroad track, Valley, Industrial Area, Hospital Room. In the future, given more compute capacity, these approaches can be used for all the 365 classes, or on the reduced classes like the one used by Xin Chen et. al. [4].

The training dataset contained a total of 50,000 images which had 5,000 images of each scene category. The test set contained a 1,000 images including 100 images of each scene category. We measure the accuracy of each of our approach by the percentage of images with correct scene label predictions by the model.

$$\text{Accuracy} = \frac{Number of Images Correctly Classified}{Number of Images in Test Set}$$

We compare the results from the three different ways to solve the problem. TF-IDF[1] technique used commonly in information retrieval, the second was finding similarity from the word embeddings and lastly using a neural network.

4.1. TF - IDF Method

Term Frequency and Inverse Document Frequency is one of the oldest, most common and popular method from Information Retrieval used to find relevant documents in a corpus given a particular word. The TF-IDF method in IR maps the importance of the presence of word i in document j. Similarly, we expect that computing the TF - IDF parameters in this use case would be a good representation of the correlation of object(i) in identifying scene(j). Using the idea from [4], we considered scene as documents and objects present in them as words. With this tweak, we were able to compute the TF-IDF object to scene mapping using the following -

$$\text{TF (Obj i,Scene j)} = \frac{No.of Images of scene j and Image i}{No.of images of scene j}$$

$$\text{IDF (Object i)} = \log(\frac{No.of scenes}{No.of scenes containing object i})$$

The final matrix is calculated as the element wise product of the TF and IDF matrices. This matrix will henceforth be referenced as W(tfidf).

Algorithm 1 TF IDF Approach

Input: Train Dataset D of RGB Images, $I_k \, \epsilon D : D = [\mathrm{I}_1, I_2, I_3, \ldots, I_{50000}]$
Train Dataset Class Labels S, = $[S_{I1}, S_{I2}, \ldots, S_{I50000}]$
Train Dataset Dt of RGB Images, $I_k \, \epsilon D : D = [\mathrm{I}_1, I_2, I_3, \ldots, I_{1000}]$
Output: Scene Vector $S_{test} \; \epsilon[1, 1000] where \mathrm{S}_I$ is scene predicted for Image I of Test Dataset Dt

Objects = Scene Parsing Module(Train Dataset)
W(tfidf) = TF * IDF from (Objects, Class Labels)
Objects test = Scene Parsing Module(Test Dataset)
Scores = Objects test * W(tfidf)
Scenes Vector Si = Max[$Scores[i, :]_{index}$ i $\epsilon[1, 1000]$
Return Scenes Vector

The scene parsing module[13], returns a list of objects present in an image of a total of 335 objects it can detect. We suitably convert this into a 0 - 1 335 dimensional vector (present or absent vector). The concatenation of the vectors of all images is the final Objects matrix. Using the objects matrix and ground truth class labels we compute the parameters of the model i.e the W(tfidf). Objects test matrix is calculated in the same way as the Objects train matrix above. The scores vector is computed through a matrix multiplication and the scene with the maximum score is assigned as the predicted scene from the model.

We got an accuracy of 66.7% using this model.

4.2. TF-IDF and NLP Model

The TF IDF model can capture the correlation between objects that are usually present in images of a particular scene. The model gets better when trained on a large number of training examples. However, it misses out on an important property of how similar the scene and the words are when used in natural language. In this algorithm, we captured this similarity by creating the word vectors of the scenes and the objects and then finding their cosine similarity.

We used the 300 dimensional glove vectors as word embeddings. The psuedo-code below explains in detail, how the algorithm works.

Algorithm 2 NLP Approach

Input: Train Dataset D of RGB Images, $I_k \, \epsilon D : D = [\mathrm{I}_1, I_2, I_3, \ldots, I_{50000}]$
Train Dataset Class Labels S, = $[S_{I1}, S_{I2}, \ldots, S_{I50000}]$
Train Dataset Dt of RGB Images, $I_k \, \epsilon D : D = [\mathrm{I}_1, I_2, I_3, \ldots, I_{1000}]$
Output: Scene Vector $S_{test} \; \epsilon[1, 1000] where \mathrm{S}_I$ is scene predicted for Image I of Test Dataset Dt

Objects = Scene Parsing Module(Train Dataset)
W(tfidf) = TF * IDF from (Objects, Class Labels)
Objects test = Scene Parsing Module(Test Dataset)
Scenes vectors = Word Vector(scenes)
while *For each image* **do**
    word vector += W(tfidf [obj(i) * scene(j)]) * word vector(i)
    Similarities = Cosine Similarity (scenes vectors, word vector)
    Scene predictions [Ik] = Max[Scores (Row i)]index where i bel. [1, 10]
end
Return Scene predictions

In this algorithm, we compute the TF-IDF matrix as explained before. However, in this case we multiply this score with the word vector of the object under consideration and finally sum up the weighted word vectors of

all the objects present in the image. The cosine similarity of the weighted objects word vector and the scenes word vector determines the confidence of the objects present in the image. The scene having the maximum confidence is the predicted scene by the model.

This model achieves an accuracy of 67.1%.

4.3. Neural Network Approach

In this approach we use neural networks to train a model to map the correlation between the objects present in an image to the scene classified. We use the same objects train matrix from the TF-IDF approach but use neural networks to learn the parameters of the correlation weight matrix (matrices) and pick the scene with the maximum score. The pseudo code below shall explain the details concretely -

Algorithm 3 Neural Network Approach

Input: Train Dataset D of RGB Images, $I_k \, \epsilon D : D = [\mathrm{I}_1, I_2, I_3, \ldots, I_{50000}]$
Train Dataset Class Labels S, = $[S_{I1}, S_{I2}, \ldots, S_{I50000}]$
Train Dataset Dt of RGB Images, $I_k \, \epsilon D : D = [\mathrm{I}_1, I_2, I_3, \ldots, I_{1000}]$
Output: Scene Vector $S_{test} \; \epsilon[1, 1000] where \mathrm{S}_I$ is scene predicted for Image I of Test Dataset Dt

Objects = Scene Parsing Module(Train Dataset)
Net = NeuralNet(Objects, Class Labels)
Objects test = Scene Parsing Module(Test Dataset)
Scores = Net(Objects test)
while *For each row of scores* **do**
    Scene$_predictions[Ik]$ =Max[Scores (Row i)]$_{index}$, i $\epsilon[1, 10]$
end
Return Scene Predictions

We achieved a maximum accuracy of 82.1% using the neural nets pproach.

5. Discussion

In each of the three approaches, we performed many minor tweaks such as hyper parameter tuning in neural networks, trying various word embeddings like Word2Vec, GloVe in the NLP approach. The accuracy reported above is the maximum accuracy obtained through these approaches after experimentation with the minor tweaks. In this section we discuss the various experiments we performed, and also give intuition and analysis of the varying accuracies we got through these experiments.

5.1. TF - IDF Approach

This approach is overall a standard straight forward algorithm with few places for experimentation. During the calculation of the inverse document frequency, smoothing is applied to ensure that under represented classes are given a minor boost and the weight of over represented classes is reduced. Considering our numerator would be 10 (number of classes) and the denominator would vary from 0 to 10, we tried out various parameters for smoothing. However, not much variation in the accuracy was observed with accuracies ranging from 64.1% to the maximum 66.7%. The maximum accuracy was obtained by adding 0.5 to the numerator and 0.25 to the denominator.

5.2. NLP - Approach

In this approach we tried out several different ways and the accuracy would vary significantly for every tweak. The most important factor which drastically improved accuracy was the choice of word embeddings. We tried out the word2vec model, fasttext and glove vectors. In all the three cases, we used pretrained models from the gensim library. The models used were the most recent state-of-the-art models and used for different purposes.

Fast text vectors performed the worst of all. We achieved an accuracy of 57.9% when trained with the Word2Vec model, and an accuracy of 67.1% on using the Glove 300 dimensional word vector model.

We believe that the size of the corpus used for training the word vector model had a direct relation to the generated quality of word vectors. Since the Glove vector model was trained on a huge dataset obtained from all wikipedia articles with a combined size of 1.7 gigabytes the model was better at detecting similar and dissimilar words as compared to the Word2Vec model trained on a corpus of 600 MB.

5.3. Neural Networks Approach

The neural networks approach performed the best of all three with a much higher accuracy than the other two approaches. There was a lot of scope for experimentation in this part.

First decision was to decide on the loss function, which we finalized as the Softmax loss function since

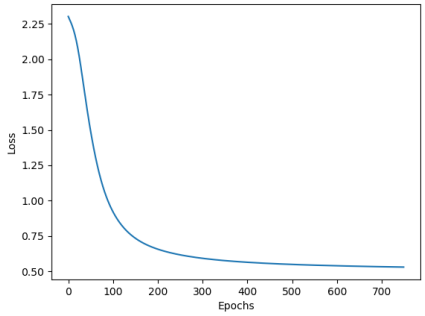

Figure 4. Loss vs Epoch

it was the most suitable one in this case. After a lot of tweaking we set the learning rate to be 0.20, and trained the neural network for 750 epochs. The network trains faster with less data, as we saw that it converged in around 100 epochs when we trained it on 1/10th of the training data.

We tried a lot of different architectures of neural networks. Since this network had 335 input classes and 10 output classes and not a lot of training data we used Fully Connected Neural Networks. Having more than 3 layers in the network, would gradually reduce the accuracy of the network on the test dataset. We used a two layer neural network with 200 hidden neurons to get the maximum accuracy.

We trained the neural network first on just 1/10th of the training data, i.e 500 images of each class and 5,000 total images. We achieved a maximum accuracy of 78.9% on the test set. On training with 5,000 images of each class and 50,000 training class the accuracy improved to 82.2%. This change of just about 4% suggested that the dataset contained plenty of images similar to each other.

A plot of the loss function over varying epochs can be seen in the figure 4.

5.4. Comparison of the NLP Approach and the TF - IDF Approach

The TF - IDF approach was a baseline model and was guaranteed to perform the worst given it's obvious shortcomings like not taking into consideration the similarity of objects and scenes, and primitive purely statistical approaches.

We expected the NLP approach to perform better than the TF-IDF approach. While it did outperform the TF IDF approach by 1% it was still lesser than we theoretically expected. One of the reasons we think is

Figure 5. Industrial Area

the addition of all object vectors allowed a considerable amount of noise in the final vector by the addition of several dissimilar objects.

5.5. Comparison of Neural Networks approach and the NLP approach

The neural networks approach as expected outperformed the NLP approach by a considerable margin. The main reason is the increase in expressibility of neural networks by the addition of hidden layers. Also the noise due to a few common dissimilar objects from the true scene label led to a corresponding decrease in accuracy in the NLP approach.

6. Results

We have tested our models on the Test Dataset of the Places 365 dataset which contains 1000 images of the 10 classes selected. The basis of comparison of the models is the number of images whose scene was correctly classified by the model. The results have been summarized in Table 1.

Qualitatively, we give a few examples of different images and their scene labels as predicted by the model. These examples can be seen in Figures 5, 6 and 7 with their captions being the respective class labels.

7. Conclusion

From the project, we learned applications of neural networks in different domains. We were able to under-

Method	Images Correctly Classified	Total Images Tested	Accuracy
TF - IDF	667	1000	66.7
NLP Approach	671		67.1
Neural Networks	820		82

Table 1. Accuracies obtained with different approaches

Figure 6. Pond

Figure 7. Gas Station

stand how different tasks in Computer Vision are interdependent. The improvements in one task could then be used to solve various other tasks. In the end, we would like to note a few things we could perform given more computational capability.

1. The neural network approach should be scaled up to include all 365 classes of the Places 365 dataset[3]. This model could be complicated and would require a more complex architecture with several hidden fully connected layers than the one presented in this text. If the new model obtained on training with 365 classes, could have as much accuracy, this new approach could set a new benchmark in scene classification models.

2. The lack of compute power forced us to use low resolution 256 * 256 images of the Places365 [3] dataset. This led to an increased error on the scene parsing task. An error in this task, gave an incorrect output to our various approaches and hence led to a decreased accuracy. One of the future tasks would be to train the scene parsing model on the High Resolution (atleast 512 * 512, some images even higher) dataset and use those outputs in our approach.

3. The NLP approach can be modified slightly wherein the similarity of the average of the weighted object vectors present in an image and the scene word vector could be measured. Currently, the sum of the word vectors is used, which might give an excessive weight to objects quite dissimilar to the appropriate scene. This case can be avoided by taking the average as the weight of dissimilar objects would still be quite low and the final objects vector would be reasonably similar to the scene vector.

References

[1] Term Frequency - Inverse Document Frequency Calculation of Scenes Considered and Objects Present. 2, 4

[2] Z. B. Ali, Nouman. 15-Scene Image Dataset. 1

[3] J. X. A. T. B. Zhou, A. Lapedriza and A. Oliva. Places 365 Dataset - Used for Image Scene Recognition Task. 1, 2, 3, 7

[4] D. W. X. Z.-M. P. Bao Xin Chen, Raghavender Sahdev and J. K. Tsotsos. York University - Scene Classification in Indoor Environments for Robots using Context Based Word Embeddings. 2018. 2, 3, 4

[5] X. P. S. F. A. B. Bolei Zhou, Hang Zhao and A. Torralba. ADE20K dataset - A part of the Broden Dataset used for Scene Parsing. 1

[6] K. E. A. O. J. Xiao, J. Hays and A. Torralba. SUN Database: Large-scale Scene Recognition from Abbey to Zoo. 2

[7] C. D. M. Jeffrey Pennington, Richard Socher. Glove Vectors - Type of Word Vector Representation. 2

[8] F. R. Josh King, Vayu Kishore. Scene classification with Convolutional Neural Networks. 2

[9] A. G. H. C.-W. N. Jun Yang, Yu-Gang Jiang. Evaluating Bag-of-Visual-Words Representations in Scene Classification. 2

[10] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope," IJCV, 2001. 1

[11] A. Quattoni and A. Torralba. Recognizing Indoor Scenes. 2

[12] J. K. T. Raghavender Sahdev. Indoor Place Recognition System for Localization of Mobile Robots. 2

[13] B. Z. Y. J. Tete Xiao, Yingcheng Liu and J. Sun. Unified Perceptual Parsing for Scene Understanding. 1, 2, 3, 4

[14] K. C. G. C. J. D. Tomas Mikolov, Ilya Sutskever. Distributed Representations of Words and Phrases and their Compositionality. 2

[15] R. C. Vijay Badrinarayanan, Alex Kendall. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. 1