ObjectNet: Leveraging Object Embeddings to Improve Scene Prediction

Eduardo De Leon* edeleon4@mit.edu
Harini Suresh* hsuresh@mit.edu
Nicholas Locascio* njl@mit.edu
Tiffany Wong* tcwong@mit.edu

Abstract

The Places2 challenge is organized by MIT with the goal of identifying the scene category depicted in a photograph. The Places2 dataset provided for the challenge include 10+ million photographs of over 400+ scenes. In this work, we use a subset of the Places2 dataset to explore how predicting object embeddings affects scene categorization. We run five experiments using various object embeddings as auxiliary targets. We explore 3 different types of object embeddings including (1) multi-hot object vector, (2) embedded object co-occurrence vector and (3) semantic object representation created from Word2Vec word embeddings. From these experiments, we conclude that a semantic object representation (3) is most impactful on scene categorization (+0.38%), and the combination of all 3 auxiliary targets yields best results (+0.46%) on Top 1 classification accuracy.

1. Introduction

Humans are remarkably good at quickly understanding scenes we are presented with, even when the contents or details vary. Indeed, this ability is extremely important, since our interpretation of our environment is crucial to any further action or reasoning. However, there are many challenges that must be overcome in order for a computer to achieve this kind of accuracy in classifying scenes. Unlike object classification, scenes are made up of a number of entities that may or may not be present and can appear in an arbitrary order.

Deep convolution neural networks (CNNs) have shown much promise to tackle this problem – especially in recent years. New algorithmic advances, large and publicly available datasets, and an increase in computing power have propelled CNNs to achieve state-of-the-art results in image classification tasks.

Despite their many successes, CNNs have not significantly outperformed other methods for scene classification, as they have with other tasks [16]. This deficiency could be due in part to the fact that traditional scene classification models assume an orthogonality of labels. For instance, given an image of a shower, the model would be penalized equally for predicting "Bathroom" as it would for predicting "Volcano". This introduces much noise in our gradients and is a misleading signal to provide our model. This example is illustrated in Figure 1.

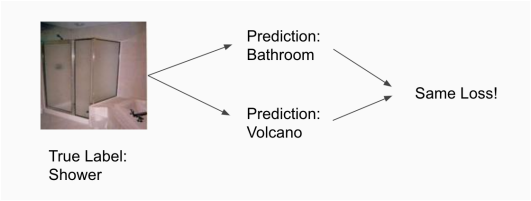

Figure 1. Illustrative example of issues with training on one-hot scene label categories.

Additionally, when building complex models such as deep neural networks, there is a risk of the model overfitting to a simple classification task. The intermediate layers of a neural network can be thought of as latent representations of the input that should be relevant to the task at hand. Studies have shown that by introducing auxiliary targets for the network to predict, the subsequent intermediate representations are more robust and generalizable [1, 2].

We aim to improve the performance of the CNN model for scene classification by having it take advantage of richer semantic information about the contents of the scene – namely, the objects contained in the scene – recreating the multifaceted input humans have when interpreting their environment. In order to do this, we use information about the objects contained in scenes to add additional attributes for the model to predict during training. This addition makes our loss function (and corresponding gradient descent) during training more robust to the one-hot problem shown in 1. It also helps the model learn more relevant and generalizable latent structures.

2. Related Work

2.1. Training with auxiliary targets

Several studies have explored multitask learning: learning with auxiliary targets that are different but related to the

* Authors contributed equally and listed alphabetically.
Code for this work can be found at https://github.mit.edu/njl/topeka

final prediction task. These studies have been done in the areas of NLP [2], healthcare informatics [3], face recognition [4], and object detection [5].

These studies have found that additional targets can help disentangle confounding factors, because the model is optimized based on multiple explanatory criteria [6]. Additionally, auxiliary targets have a regularizing effect. These models appear to achieve greater performance on the test task because the latent structures learned within the model are more general than the specific task at hand.

2.2. Embedding label text

When considering the task of incorporating auxiliary targets, it is possible to also make use of the information encoded in the actual label text. A recent study utilized label embeddings as attributes in an image classification task, and found that the model was able to learn useful additional information from the semantic text encodings [7].

As was done in this study, one way to encode the textual labels in a way that captures useful semantic information is by representing the text as embeddings from a recurrent model such as Word2Vec [8]. Using a neural skip-gram model, Word2Vec is able to represent words as lower-dimensional vectors where semantically similar words are nearer to each other.

3. Data Set

We use a subset of the Places2 [11] dataset. The original Places 2 dataset consists of 10+ million images of 400+ unique scene categories. Whereas the original Places2 contains 400+ unique scene categories, our subset only contains 100 different scene categories. We split our dataset into train, validation, and testing consisting of 100,000 examples for training, 10,000 for validation, and 10,000 for testing. All images are provided of size 128 x 128. The dataset also contains object annotation data for each image, giving the presence and location of objects in the image from 175 different classes. Object annotations include object categories as well as object location. A sampling of object annotations is provided in Figure 2.

4. Methods

We utilize AlexNet [9], previously used for object classification in the ImageNet competition [14]. Our methods emphasize improvement in accuracy on the primary task of scene classification through the addition of related auxiliary targets during training. Our additional auxiliary targets involve a combination of different object embeddings. Inspiration for our methods come from the correlation between object presence and scenes. We take advantage of our object annotations by running models that:

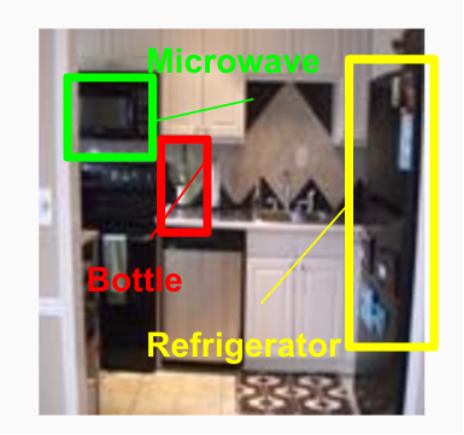

Figure 2. Object annotations for a given image in Mini-Places2 dataset.

1. Use their combinations e.g. a cup and plate may signify a dining room whereas a cup and a bottle is closer to a bar.

2. Capture semantic similarities e.g. a cow and a pig can be from similar scenes.

4.1. FastAlexNet Model

For the mini-challenge, we adopt the AlexNet [9] CNN architecture, one of the most popular early deep architectures. We take a slight departure from the original AlexNet model, to create FastAlexNet. We modify AlexNet's input layer to accept 128x128 images instead of the original 256x256. The supplied images in The Places2 dataset come in 128x128, so upscaling them to feed in 256x256 images provides no additional information and only slows down and bloats our model.

Decreasing the size of our input layer speeds up our training time, but also effectively doubles our filter size. For example, AlexNet's first filter is 11 pixels wide, or around 5% of the total image width. In our modified AlexNet, the first filter is still 11 pixels wide, but only operates over 10% of the image width. This seems to be an advantageous modification because we are classifying scenes and not objects, so a wider filter size is more appropriate because scenes are more global features than objects. However, in practice, we see a decrease in performance vs the vanilla AlexNet TA baseline.

We choose to keep our modified AlexNet, despite poorer performance, for its fast training time and stability of results. We actively choose not to use more powerful models like VGG [10], or GoogLeNet [13] in order to properly and fairly benchmark the advantages of our various auxiliary task techniques. It is possible that these powerful models would be able learn better image representations from just the scene labels than our object encodings could have provided.

4.2. Object Encodings

We create three different encodings for our given object annotations. Each encoding represents different aspects of information present in the image. ObjectNet is made to predict these encodings as a sub task.

4.2.1 Multihot Encoding

Our first encoding is a multihot vector of the image objects, named Multihot-Objects. We build Multihot-Objects by parsing the objects Extensible Markup Language file into a vector of counts related to the object ids. The vector is of length 175, and each value reflects the number of times objects of a given index's type were found in the image.

Predicting Multihot-Objects as a sub task increases the difficulty of the model's objective. We explore the sub task of learning what objects are in an image because of its relationship with the scene in question. For example, given the object classification of a toothbrush, a model should increase the likelihood that the scene is a bathroom, as opposed to a forest.

Although the model would find similar representations on its own, this added sub task allows us to guide the model more closely by providing a loss for poor object representations.

4.2.2 Autoencoded Co-Occurence Embedding

For our second encoding, we further encode Multihot-Objects into a relational representation with a reduction in dimensions from 175 to 40. We accomplish this encoding, named Relational-Objects, by training an auto-encoder over all Multihot-Objects.

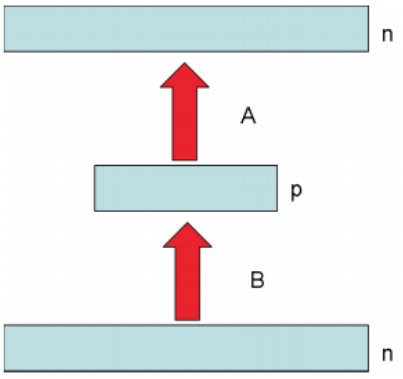

Figure 3. Auto encoder from size n to p and back to n. [**?**] Our model has n = 175 and p = 40.[15]

Auto encoders work by training an encoder-decoder model to encode a vector and then decode back the original vector. Our encoder takes as input a 175 dimensional vector and outputs a vector of 40 dimensions. This vector is then fed in to the decoder that outputs a vector of length 175. The autoencoder is trained over training and validation data. The result is an encoder that can encode Multihot-Objects into a 40 dimension representation.

Expected benefits from the encoded vectors are two-fold. First, inter-object relationships should be encoded in the resulting representation. Second, the embedding will have reduced dimensionality and be less sparse.

The representations found by autoencoders rely on the clustering of features in the training data[18]. In the process, autoencoders create an encoding of the input vector that finds a local minimum of information loss. We expect this clustering to contain semantic and inter-object relationships. Predicting groups of objects as a subtask is closer in its objective to predicting the scene, since a scene is more closely related to the holistic image than the separate and individual objects found within it.

Reduced sparsity and dimensionality also simplifies the task by having the model predict fewer outputs. This reduction in predicted outputs does not reduce the expected benefit of the subtask since we expect that the same information is encoded in the resulting space. Rather, it avoids the unnecessary difficulties that come from predicting high dimensional, sparse vectors.

4.2.3 Word2Vec Object Semantic Embedding

To create our third encoding, called Semantic-Objects, we utilize a Word2Vec [8] model in order to extract word embeddings to construct a semantic object representation for our image. We use a Word2Vec model pretrained on Google News [19] to extract embeddings for the 175 different object types. We use the gensim [20] software package [20] for loading the model and extracting the word vectors.

For each object in a given image, we extract the object label description, and run this description through our pretrained Word2Vec model, which produces a semantic embedding of the object. To compute our full Semantic-Object encoding of the image, we take the average of all word/object vectors for all objects in the image.

Our Word2Vec model does not include multi-word phrases in its dictionary of inputs, but rather only includes individual words. However, many object categories include phrases such as "arcade machine" and "billiard table". In order to accommodate for such categories, we take the average of their respective word vectors to create a single embedding for the category.

Model	Predicts
Scene Baseline	scene label
Multihot Model	scene label + multihot object vector
Co-Ref Model	scene label + compressed object encoding vector
Word-Embedding Model	scene label + average word embedding vector
Full Model	scene label + all auxilary tasks

Table 1. Our five models and their respective predictions

4.2.4 Combining Loss Functions

Our network has (1) a primary task, classifying scenes, and (2) a number of potential auxiliary tasks. For our classification task, we use softmax-cross entropy loss. For our auxillary tasks, we use L2 loss as we are predicting real-value targets, not probability distributions. We combine all of the model's loss functions using a weighted average scheme. We supply, as a hyperparameter, a vector of weights to apply to the various loss functions. Our full model is shown in Figure 4. See Equation 1 for our combined loss function expression.

$$L_{total} = \frac{\sum_i w_i * loss_i}{\sum_{j=0}^{W} w_j} \quad (1)$$

Since loss weights are configured as a hyperparameter, this setup has the additional benefit of allowing us to run our various experiments using a single model and providing a different loss weight setting for each experiment. For example, if we wanted to just penalize our model based on only the scene labels and Co-Occurence Embeddings, we would supply a vector [1.0, 0.0, 1.0, 0.0], which would zero-out the other auxiliary task losses.

In implementation, multiplying by zero does not always work, as an unmanaged loss can become 'NaN' and squash the true loss gradient. To remedy this, we have a conditional to check if the weight is non-zero before adding it to our overall loss function.

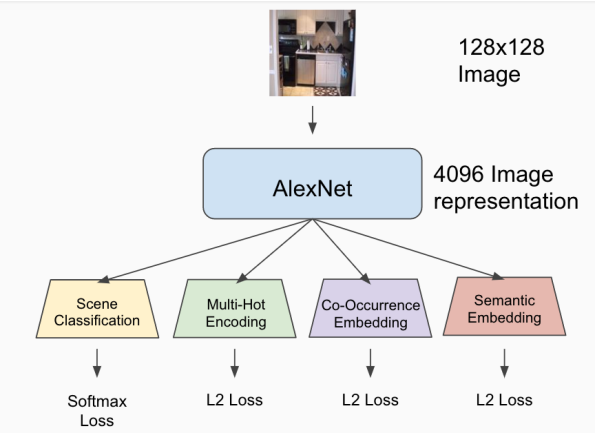

Figure 4. Our Model.

5. Experiments

We ran five different models, written in TensorFlow [21] on a K520 GPU g2.xlarge instance on AWS. Each model uses a hyperparameter settings of batch_size=32, learning_rate=0.001. We use Adam Optimizer [17] to minimize our network's loss function. We run each model for 10 epochs. We do not utilize early stopping mechanisms in order to uniformly evaluate each of our models.

We implement our modified AlexNet, called FastAlexNet, by modifying the available open-source implementation provided within the official TensorFlow examples repository. We did not use the staff-provided starter code, as we started the project before this was released. We normalize all images to contain a range of pixel values from -1.0 to 1.0.

We configure our five models to predict (1) scene label (2) scene label + multihot object vector (3) scene label + compressed object encoding vector and (4) scene label + average word embedding vector (5) scene label + all auxilary tasks, respectively. We call these models (1) Scene Baseline, (2) Multihot Model, (3) Co-Ref Model, (4) Word-Embedding Model, (5) Full Model.

6. Results

Object embeddings as auxiliary targets have a slight positive impact on scene classification. We see an improvement of +0.46% accuracy on Top 1 Classification and +0.36% accuracy on Top 5 classification over the scene baseline. Our best model uses all 3 auxiliary targets, and achieves increases of +0.08% and +0.16% respectively.

7. Discussion

Our experimental results suggest that object embedding are useful to scene classification models. However, our improvements are marginal, so we cannot conclusively say so.

The reason we don't see a much larger increase in test performance when adding auxiliary targets is likely due to many factors. Most prominently, there is a very limited amount of object data available in the dataset. For the 100,000 training images, only 3,524 images actually contained any object annotations. This means that for 97.5% of our gradient updates, all 5 models had the same loss func-

Accuracy	Scene Baseline	Multihot Model	Co-Ref Model	Word Embedding Model	Full Model
Train Top 1	71.89%	**72.91%**	71.15%	72.30%	72.24%
Train Top 5	94.92%	**94.51%**	94.16%	93.57%	93.68%
Val Top 1	27.27%	27.35%	27.56%	27.65%	**27.73%**
Val Top 5	55.90%	56.06%	56.13%	56.22%	**56.25%**

Table 2. Results table showing accuracy on train and val

tion. This is largely why all 5 models have near-equal performance.

It seems our auxiliary tasks have a regularizing effect on our model, as we see decreases in training accuracy and increases in validation accuracy. It could be that even though the different gradients for just 2% of the training examples was not enough to achieve a much better image representation, it was able to impart some regularization.

A limitation of predicting word2vec label embeddings could also be present for certain scenes. Firstly, our category semantic vectors for multi-word categories is problematic. The combination of single words from a noun phrase can convey a very different idea than the noun phrase itself. For example, the category "plant pot" refers to a pot that would exist in a garden or lawn. However, our averaging technique produces a combination of plant and pot, which could very well refer to objects in a kitchen. Secondly, averaging all objects in a single scene lead to a loss of more fine-grained information.

8. Further Work

We would like to extend this study using a dataset of more complete object annotations to get more conclusive results on the effect of object annotations on scene classification. We are also interested in exploring the provided object bounding boxes to enhance our embedddings. For example, we can scale an object's contribution by the size of its bounding box, to give more weight to larger objects that take up much of the visual scene, or create embeddings that infer relationships between objects based on how close they are.

Given our findings that limited object data provides marginal benefits, exploring scene encodings in addition to object encodings may also be of interest.

Scene encodings could be used for reformulating the task as outputting a scene encoding. The model would then receive more finely tuned feedback as to the extent to which the model was incorrect. An example, illustrating the intuition behind this idea, would be a lower MSE for labeling a jungle as a forest than as a beach. For the current one-hot prediction setup, both scene classifications would have equal loss.

This evaluation of the semantic encoding of scenes also addresses the discrepancy between training loss and evaluation. Currently, use of softmax for training penalizes a model equally for ranking the correct scene second or last. Exploring multi-class multi-label loss functions as well as MSE could lead to a model that learns more in tune with how it will be evaluated.

9. Personal Contributions

I implemented object extraction from their xml files into the appropriate data structure for use by different object embedding (peer programmed with Eduardo) and general data pre-processing. I also implemented Word2Vec Object Semantic Embedding as described in Section 4.3.2 (peer programmed with Nicholas).

Generally, the whole group brainstormed collaboratively on all aspects major aspects of the project, such as exploring the effect of word embedding in scene classification and which types of word embedding to use.

References

[1] R. Caruana, Multitask learning. *Machine learning* 28.1, 1997. 1

[2] R. Collobert, and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proc. 25th ICML*, ACM, 2008. 1, 2

[3] Lipton, Zachary C., et al. "Learning to Diagnose with LSTM Recurrent Neural Networks." arXiv preprint arXiv:1511.03677 (2015). 2

[4] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In Computer Vision ECCV 2014, pp. 94-108. *Springer International Publishing*, 2014. 2

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587. 2014. 2

[6] Zhong, Yu, and Gil Ettinger. "Enlightening Deep Neural Networks with Knowledge of Confounding Factors." *arXiv preprint arXiv:1607.02397* (2016). 2

[7] Akata, Zeynep, et al. "Label-embedding for image classification." *IEEE transactions on pattern analysis and machine intelligence* 38.7 (2016): 1425-1438. 2

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 2, 3

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012. 2

[10] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015. 2

[11] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba and A. Oliva, "Places2: A Large-Scale Database for Scene Understanding," *Arxiv*, 2015. 2

[12] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. "Object Detectors Emerge in Deep Scene CNNs." *International Conference on Learning Representations (ICLR) oral*, 2015.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. "Going Deeper with Convolutions." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, Imagenet: A large-scale hierarchical image database. *In Proc. CVPR,* 2009. 2

[15] P. Baldi, Autoencoders, Unsupervised Learning, and Deep Architectures. *JMLR: Workshop and Conference Proceedings*,2012 3

[16] Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. "Mid-level visual element discovery as discriminative mode seeking." *Advances in neural information processing systems*. 2013. 1

[17] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014). 4

[18] Baldi, Pierre, and Zhiqin Lu. "Complex-valued autoencoders." Neural Networks 33 (2012): 136-147. 3

[19] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013). 3

[20] Rehurek, Radim, and Petr Sojka. "Software framework for topic modelling with large corpora." In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. 2010. 3

[21] Abadi, Martn, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016). 4