CS231A Course Project Final Report
Unsupervised Learning of Text-sensitive Features For Large-scale Scene Classification

Maurizio Calo Caligaris
Stanford University
maurizio@cs.stanford.edu

Abstract

"It's not neccesarily who has the most powerful algorithm who wins, but rather who has the most data" is a widely held belief in the machine learning community. One could always try to find more labeled data, but this may sometimes be expensive or prohibitive. In this work, we seek ways to leverage vast amounts of freely available unlabeled data along with weak or noisy annotations (possibly coming from different modalities, such as images and text) to our advantage and create meaningful feature representations. Specifically, we use freely available Flickr images along with their tags to learn features that capture both visual and sematic information, and evaluate the performance of such features on the SUN large-scale image classification task.

1. Introduction

Unsupervised feature learning addresses the challenge of levering vast amounts of data that is available at little or no cost to develop meaningful feature representations for a task of interest. For example, if we're trying to perform a computer vision task on a specific dataset, we might look at other images on the web to gather natural image statistics and come up with features that are useful for the task we're interested in. The analogy here is, in loose terms, that we let out a robot "in the wild" and let it learn a understand what the world looks like. We let it remain there for as long as possible (which corresponds to using as much data as we can) and then evaluate it on a particular task(s) of interest. The advantage of such approach is that unlabeled data is often cheap and plentiful, so we can learn from lots of it and can thus obtain greatly improved results.

Very often, we can obtain vast amounts of data that comes with (noisy) annotations at no extra cost. Such annotations might even come from a different modality from the original data source. For instance, Wikipedia contains text, audio and images; YouTube contains audio, video and text; and Flickr contains images and text. To maximize performance on specific tasks, we would like to use all of the information available to us. Hand-engineering task-specific features for a single modality in itself is a notoriously difficult and time-consuming task. The challenge gets significantly pronounced when the data comes from multiple sources. We develop unsupervised feature learning algorithms that relate information from disparate data sources to take the most advantage of data (e.g. weak annotations) that comes at no extra cost to us.

In this particular project, we focus on a computer vision application. We've downloaded hundreds of thousands of images from Flickr along with their corresponding tags using the Flickr API - we would like to use such data to learn computer vision features that are sensitive to tags. The premise here is that the tags serve as weak annotations for an almost unlimited number of images available for free, which can be used to learn semantically meaningful feature representations. We can then use such features for a wide-variety of computer vision tasks that are not necessarily directly related to Flickr; in fact, the images in the dataset may (and probably will) have different statistics from the Flickr images.

One approach consists in taking an off-the-shelf descriptor such as HOG and trying to predict the tags associated with an image using neural networks. This way, we learn a representation of images that captures both visual and semantic information. Another approach consists in using unsupervised feature learning to learn our own descriptors from image patches, and combine such descriptors to predict tags. The result is that local descriptors end up being sensitive to tags and therefore captual semantic information.

We focus on evaluating the performance of our method in the SUN scene classification task [1]. Torralba *et al*. have put up together a large-scale database of images containing 397 scene categories, ranging from abbey to zoo, which serves as a benchmark to evaluate numerous state-of-the-art algorithms for scene recognition. With such a large number

of classes (and sometimes not many training examples from each class), it seems that we can use vast amounts of freely available data with weak annotations such as Flickr images along with corresponding tags to create a meaningful feature representation.

In the following sections we describe related work and describe some background related to unsupervised feature learning. We then present our models and describe an experimental setting which demonstrates the effectiveness of our approach. Ultimately, we conclude and offer suggestions for further work.

2. Related Work

2.1. Unsupervised Feature Learning

Raina *et al*. have introduced the paradigm of "self-taught learning" [5], in which a vast amount of unlabeled images are downloaded from the World Wide Web to learn good feature representations and improve performance on a given computer vision classification task.

Algorithms commonly used for unsupervised feature learning include deep auto-encoders ,convolutional neural networks, Restricted Boltzmann Machines (RBMs) and sparse coding. Such methods have had numerous success, obtaining state-of-the-art results in benchmark datasets such as NORB and CIFAR[7].

While much of the work in unsupervised feature learning and deep learning has focused on learning features for single modalities, there has been work regarding multiple-modality feature learning. In particular, Ngiam *et al*. have applied unsupervised feature learning and deep learning techniques to learn features for both audio and video[4]. They've demostrated an instance of cross-modality learning in which better video features can be learned if audio features are present during feature learning time. We adopt a similar approach to learn better computer vision features by taking advantage of text information during feature learning time.

2.2. Scene Classification

Scene classification is an area of active research in computer vision. A standard approach for scene categorization is the visual bag-of-words model, which essentially "chops" an image into patches and disregards their original position in the image. The bag-of-words model often works surprisingly well, as it is very simple and efficient and can be made robust to clutter and occlusion. The drawback of these methods is that they disregard the spatial layout of the images, which is very useful for scene classification. A common way of improving the effectiveness of bag-of-words models to take into account the spatial arrangement of features is to perform pyramid matching [6]: the image into increasingly fine sub-regions and compute histograms of local features inside each sub-region. In our approach, we learn local descriptors using codebooking and combine them using spatial pyramid.

We compare our method with other hand-engineered features commonly used for scene classification such as GIST, SIFT and HOG.

Torralba and Oliva have shown that urban and natural scenes can be distinguished on the basis of simple second order statistics (global amplitude spectra)[3] . We hypothesize that the Flickr tags can help us discover interesting image statistics that can be helpful for image categorization.

3. Background

3.1. Autoencoders

An autoencoder is a simple neural network with one hidden layer that learns to predict the input form itself (i.e. output is set to be equal to input). If there are less hidden units than input units, the network is forced to learn a *compressed* representation by discovering correlations among the input features, and yields results similar to PCA. The autoencoder is also able to discover meaningful structure in the data by using an overcomplete hidden layer with a *sparsity constraint* that the average hidden unit activation is ρ, where ρ, the sparsity parameter is a small number. The parameters of the model are $W^{(1)}, W^{(2)}, b_1, b_2$ where $W^{(1)}$ and $W^{(2)}$ are the weights connecting the input layer to the hidden layer, and the hidden layer to the output, respectively, and b_1 and b_2 are the bias terms. The activations (for an input x) are:

$$h = \sigma(W^{(1)}x + b_1) \qquad \text{(hidden activations)}$$
$$y = \sigma(W^{(2)}h + b_2) \qquad \text{(reconstruction of the input)}$$

where σ denotes the sigmoid function.

The loss function (corresponding to a single training example) we seek to optimize is

$$J(W1, W2, b1, b2) = \|y - x\|_2^2 + \beta \mathrm{KL}(\rho||\hat{\rho}) + \lambda(\left\|W^{(1)}\right\|_2^2 + \left\|W^{(2)}\right\|_2^2)$$

i.e., a squared-norm of reconstruction error, a sparsity penalty (KL denotes the Kullback-Leibler divergence) and weight decay to prevent overfitting (β and λ are hyperparameters of the model controlling the sparsity penalty and weight decay penalty, respectively).

We use backpropagation to compute the gradients and use stochastic gradient descent to find the best parameters $W^{(1)}, W^{(2)}, b_1, b_2$ that minimize the loss function.

Having obtained the optimal parameters, for a given input x we compute the corresponding hidden unit activations $h = \sigma(W^{(1)}x + b_1)$, which constitutes our learned features.

The real power of the autoencoders come from stacking several auto-encoders together to form a deep auto-encoder, using greedy layer-wise pre-training to initialize

near a good local optima [2]. With many layers, the network is able to predict several non-linearities and is thus able to discover more useful structure.

3.2. Y-shaped Network

For the Flickr data, we have image data as well text data. We use a network that uses image data as input and tries to reconstruct both the original input as well as corresponding tags (the name comes from the fact that there is one input and 2 outputs). We use cross-entropy loss function for the text reconstruction and weight the error corresponding to the text reconstructions by α and the error corresponding to the image by $1 - \alpha$, where α is the text weight which indicates how much we care about text reconstructions relative to the image reconstructions. For more details, see [8].

4. Our Models

4.1. First Approach: Predict Tags from Hand-engineered Features

Our first approach is basically a "brute-force" approach of encoding semantic information into hand-engineered computer vision features, such as SIFT or HOG. More specifically:

1. We are given a computer vision feature (e.g. HOG or SIFT wit Spatial Pyramid) that transforms image pixel data into a vector, along with a computer vision task (e.g. scene classification).

2. We train a Y-shaped network that learns to predict the concatenation of the input with the Flickr tags from the given computer vision features applied to the corresponding Flickr images (for which we have tags).

3. The hidden layer constitutes the learned joint features that captures image/text correlations.

4. We then forward propagate the network using as input the images from the dataset for the task of interest to obtain a new set of features for the specified task.

Since this approach does not take into account any task-specific knowledge, this approach would presumably work with a number of computer vision features for a wide variety computer vision tasks.

4.2. Learning Our Own Descriptors

The problem with the previous approach is that most features are hand-engineered without considering text information, so it would be fairly difficult to hard-code textual information into already existing descriptors. Further, it would be difficult for an auto-encoder to capture much of the predictive power of the features given the complex non-linearities that such features entail.

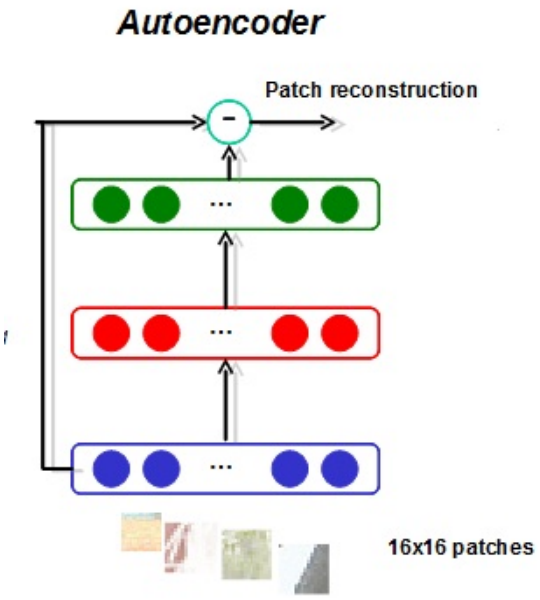

Figure 1. Text-Free Model. We train a one-layer autoencoder that learns a useful representation of a PCA-whitened 16 by 16 patch

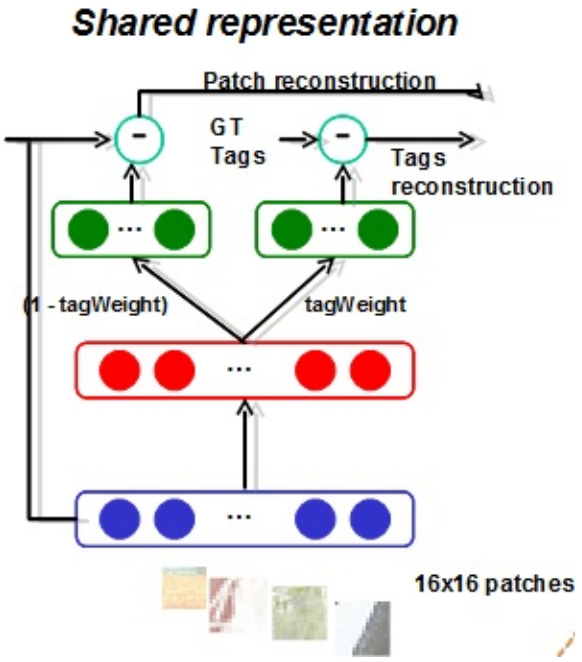

Figure 2. Y-shaped Network. We train a single-layer network that predicts tags from a visual input, while trying to reconstruct the input. The textWeight controls the relative importance of the text/visual features.

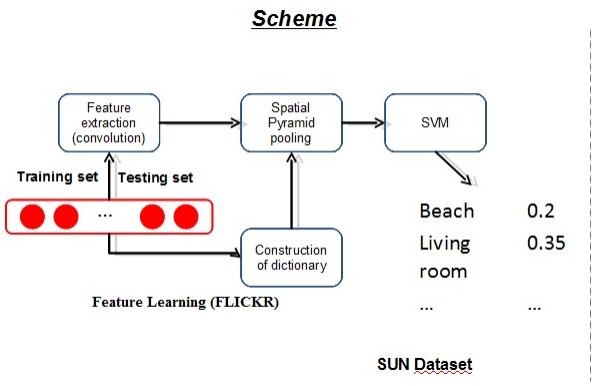

Figure 3. Feature Learning Scheme. We learn our own descriptors and codebook of visual words using Flickr data. We then densely extract features for the SUN task, and feed it into a one-vs-all SVM which predicts a class label given the learned features.

Thus, we would like to include tag-sensitivity *during feature learning phase*, i.e. use unsupervised feature learning to learn our own local descriptors while predicting tag output as well. We first describe an approach that learns good descriptors using image data alone, which serves as a good starting point for more complex models that produce tag-sensitive descriptors.

4.3. Starting Point: Unsupervised Learning of Descriptors From Image Data

Essentially, we use auto-encoders to learn a good feature representations of local 16x16 patches.

The feature learning phase is as follows (again, features are learned using Flickr dataset).

1. From an rgb image, extract a 16 by 16 patch and unroll it into a 256-D vector.
2. We normalize the data so as to have zero mean and unit variance.
3. Since pixels are highly redundant, we use PCA to reduce the dimensionality of the data.
4. We learn an auto-encoder that learns to predict the preprocessed patch from itself (The hidden layer constitutes our learned features).

We rely on the *stationarity* property of natural images, meaning that the statistics of one part of the image are the same as any other part. This suggests that the features that we learn at one part of the image can also be applied to other parts of the image, and we can use the same features at all locations.

Once we have learned features for local patches, we do feature extraction as follows (for images in the dataset corresponding to the task of interest):

1. Convolve the learned weights with the whole image, i.e. use the weights to densely extract the corresponding feature for *all* (overlapping) 16x16 patches in the image. Using overlapping patches ensures that the descriptors are *translationally invariant*.
2. Compute spatial pyramid representation ($L = 2$) for the image from the extracted features. We obtain the codebook of $n = 300$ visual words by running k-means on patches extracted randomly from Flickr data.

4.4. Towards tag-sensitive descriptors

The challenge is how to incorporate text information into our learned descriptors.

4.4.1 Y-shaped Network

Our first approach is very simple modification of the previous idea: We do exactly the same as before, but instead of using an autoencoder to predict a PCA'd patch from itself, we use a Y-shaped network to predict tags from each 16 x 16 patches. We don't expect this to work very well as it is unreasonable for most 16x16 patches of an image to give us any tag information (the statistics corresponding to most 16x16 patches are different from the statistics corresponding to a particular tag), but it's an approach worth trying and serves as a baseline for more complex models.

4.4.2 A Convolutional Neural network

.

We can take all the learned features densely extracted from an image, combine them together using mean/max pooling or spatial pyramid and predict the tags from such a representation. We then backpropagate errors all the way to finetune the weights corresponding to the low-level patches. Such afinetuning essentially encodes textual information into the local descriptors themselves.

More specifically, we train a convolutional neural network that takes an entire image as input and uses the tags corresponding to the image as the prediction output. The network works as follows:

1. From an entire image, use dense sampling to extract all 16 by 16 patches in the image. We preprocess each patch by normalizing the data so as to have zero mean and unit variance and then runnning PCA. This transforms an image into a set of PCA'd patches $x^{(1)}, \ldots, x^{(m)}$.
2. For each patch $x^{(i)}$, we predict hidden activations $h^{(i)} = \sigma(Wx^{(i)} + b_1)$

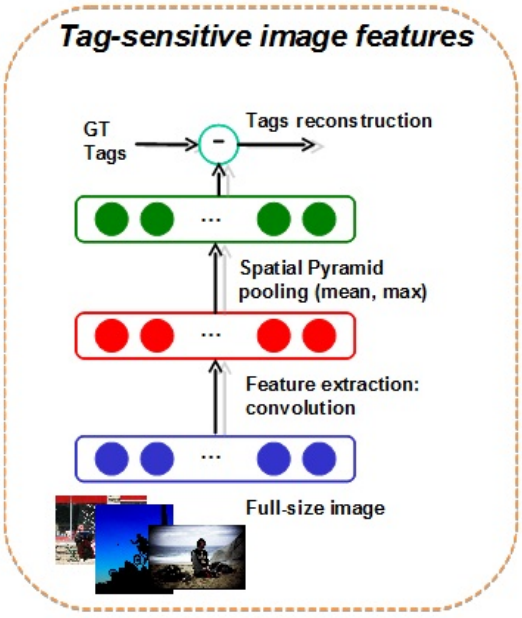

Figure 4. Convolutional Neural Network

3. We aggregate the hidden unit activations using spatial pyramid or pooling to create a layer s. e.g. $s = \frac{1}{m}\sum h^{(i)}$ if we're using mean pooling.

4. We predict $p_j = \sigma(v_j^T s + b_{2,j})$ as our prediction for tag j, t_j which is either zero or one, depending on whether the tag j is present for the image or not. We use a dictionary of size 2000 by considering only the 2000 most frequent tags in Flickr.

We seek to optimize a cross-entropy loss function $L = \sum t_j log(p_j) + (1 - t_j) log(1 - p_j)$ with respect to the parameters $W, V, b1, b2$.

We use backpropagation to compute the derivatives of the loss function with respect to the parameters, and run stochastic gradient descent to find the optimal parameters.

Given that this optimization problem is highly non-convex, we initialize the weights W, b_1 using the previously learned features (learned descriptors from patches) to get near a good local optima. THus, it is very important that the tag-free descriptors themselves achieve good predictive power.

We use the weights W, b_1 to create tag-sensitive descriptors for each patch of a given image.

As before, the feature extraction phase is as follows: We densely extract 16 by 16 patches from an image, take the learned weights to predict descriptors, which are then aggregated using spatial pyramid matching.

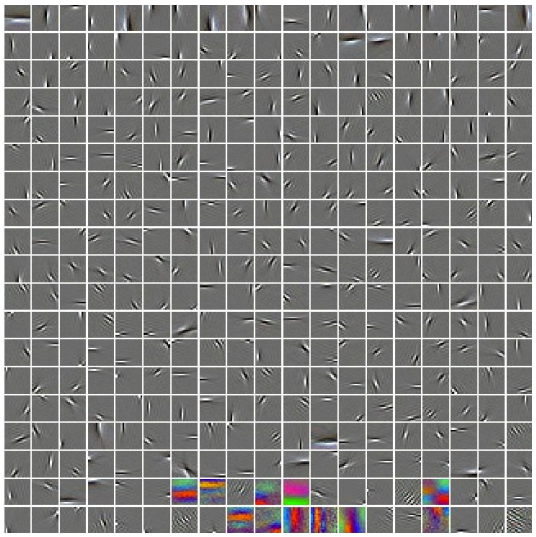

Figure 5. Text-Free

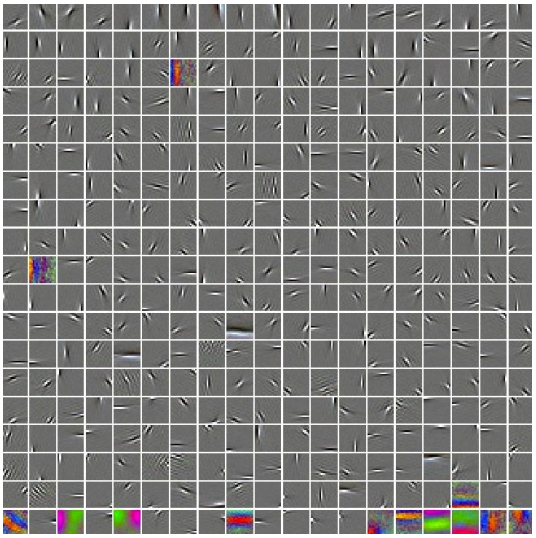

Figure 6. Text-Weight = 0.25

5. Visualizations

Before we present classification results, we first show a visualization of the bases learned by the algorithms we've implemented. Each square is a 2D depiction of a patch that would cause each hidden unit of the autoencoder to be maximally activated. In other words, each square depicts what each hideen unit is "looking for". We see that the algorithm yields localized filters that resemble Gabor filters - different hidden units have learned to detect edges at different positions,colors and orientations of the image.

Furthermore, we observe that the tag-sensitive bases learned by the convolutional network are more sensitive to

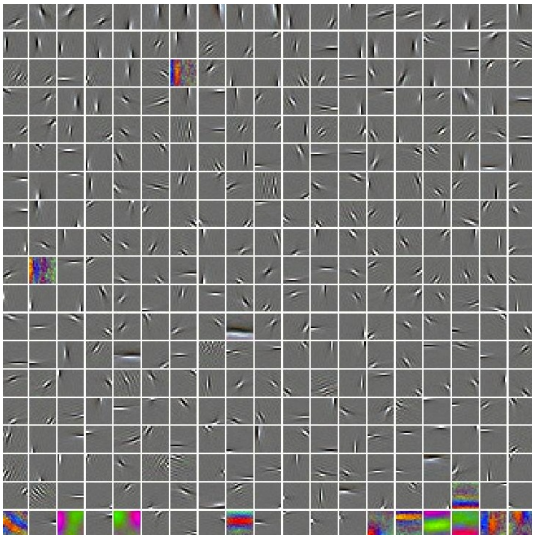

Figure 7. Text-Weight = 0.5

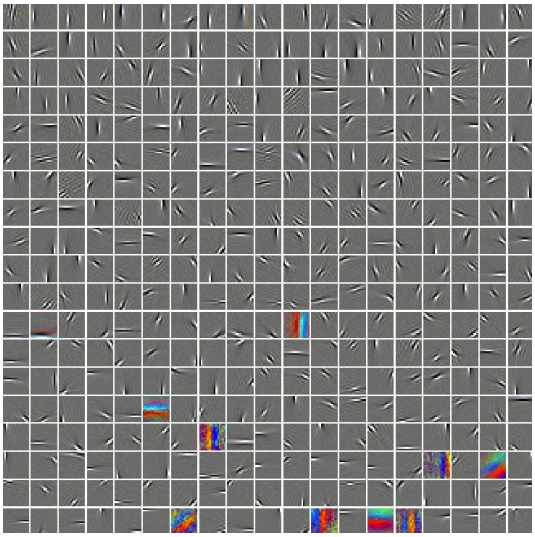

Figure 8. Text-Weight = 0.75

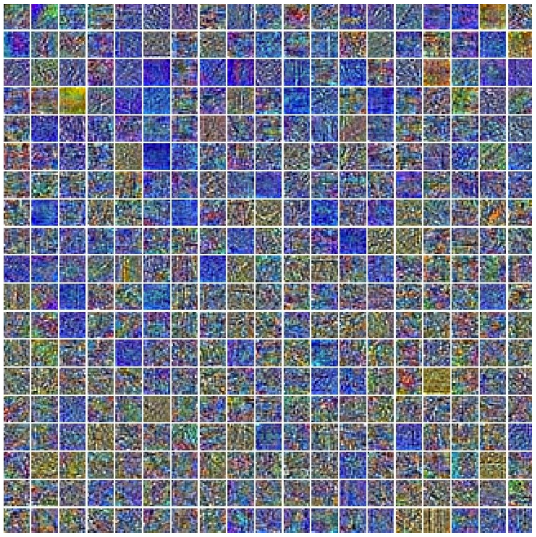

Figure 9. Text-Weight = 1. This corresponds to only trying to predict tags from images without trying to reconstruct the input. This is a degenerate case of the Y-shaped network and no learning occurs. The features learned are just gibberish

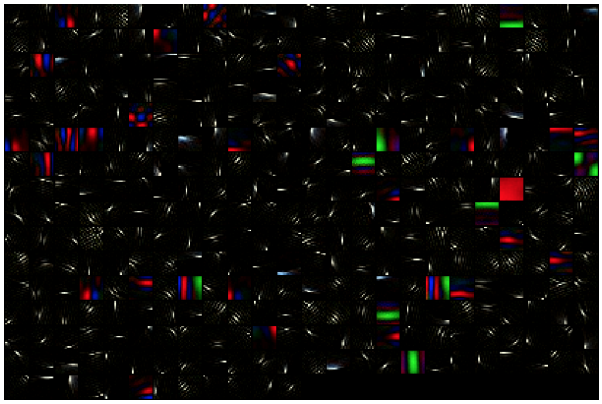

Figure 10. Text sensitive weights. We can see that the weights are more sensitive to color, which could presumably be helpful for scene recognition.

color, which seems like a good feature to learn if we're doing scene classification.

6. Results

We evaluate the performance of our method on the SUN scene classification task, using $n = 5$ training examples per class and test on 50 examples per class. The dataset contains 397 indoor and outdoor scene categories, ranging from abbey to zoo. We do feature extraction as described above, and train a one-vs-all svm to predict the category of each image. We repeat the experiment for 10 different splits of the data to obtain more statistically significant results.

The task becomes computationally expensive as we start using more training examples. Therefore, we have decided to use 5 training examples per class so we can iterate more quickly and try out several ideas. In general, as shown in [1], the relative performance of different descriptors is not dependent on the number of training examples.

We further compare performance of our models with state-of-the art features such as SIFT,GIST and HOG. We evaluate performance of such features as described in [1].

6.1. Results of First Approach

HOG is the feature that performs best in this task, so we've devoted most of our efforts into trying to improve the performance of HOG.

Feature	Classification Accuracy
Sparse-SIFT	3.2
Tag-Free Descriptors	4.71
TextWeight = 0.25	4.12
TextWeight = 0.50	4.04
TextWeight = 0.75	4.13
Our Convolutional NN	**6.12**
GIST	6.9
HOG	10.8

Table 1. Results for $n = 5$ training examples per class. We note that our (text-free) descriptors perform reasonably well, and that performance is improved by leveraging tag data using the convolutional neural network described earlier.

We've observed that most of the predictive power of the features such as HOG comes from the similarity histogram kernel. Unfortunately, since our learned features are sigmoid units in the 0-1 range, the similarity histogram kernel applied to our features does not yield particularly good results. We've experimented with many different kernels such as chi-squared and Gaussian kernels, but the best results that we've obtained are 5.32 % accuracy in the task, which are not very good compared to HOG's performance of 11.15 %, using $n = 5$ training examples per class in both cases. Concatenating the learnt features with the original features yields no improvement whatsoever (compared to the performance of the original features by themselves). This is mostly because our learn features don't perform very well by themselves.

It is worth noting that the our learned features are not bad by themselves. In fact, our learned features beat HOG if we use a linear kernel in both cases. The performance of HOG features using a linear kernel is of 4.25 %, whereas our learned features (concatenated with the original features) yield a performance of 4.98 % (again, using $n = 5$ training examples per class). Furthermore, using text information does better than using image information alone (4.98 vs. 4.45 % classification accuracy). The problem is that we don't know of a kernel that we can apply to our learned features to improve on HOG+similarity histogram.

So, the autoencoder is able to learn reasonable features from hand-engineered features, but this is more or less a "brute-force" approach that tries to encode text information into already engineered features, and in the process we destroy the properties of the hand-engineered features that make them suitable for powerful kernels.

6.2. Results of Our Own Descriptors

We've evaluated performance of our models: the tag-free model, and the tag-sensitive models (convolutional network and the y-shaped for local patches model). For the convolutional network, we use the tag-free model as a starting point

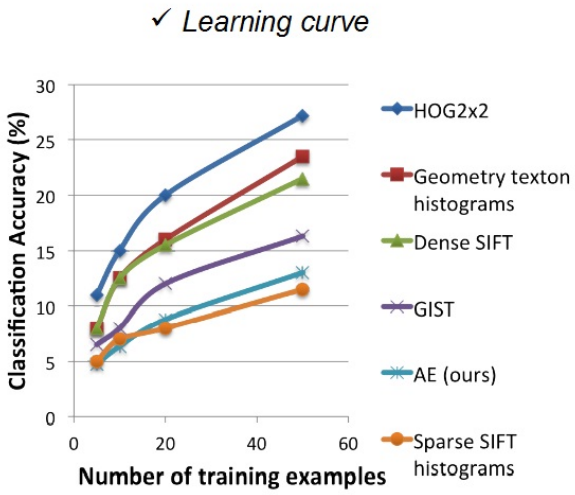

Figure 11. Comparison of our learned descriptors (tag-free) relative to other hand-engineered features for varying training set size.

for the optimization, and use mean pooling to aggregate the local descriptors. The results are summarized in Table 1.

We see that the learned descriptors (tag-free) by themselves do a reasonable job - yielding classification performance comparable to sparse-sift histograms and not much worse than GIST. As we've suspected, trying to predict tags from local patches does not help at all, in fact, performance decreases.

Furthermore, we see that the convolutional neural network is able to use the tag annotations to our advantage: performance jumps from 4.7% (using tag-free descriptors) to 6.2%, which is better than sparse-sift and approximately equal to GIST.

The results are not yet as good as HOG, but there is still much room for improvement. In particular, spatial pyramid aggregation will likely work better than mean pooling. Moreover, by training deep networks, we will be able to learn more complex non-linearities and thus we would expect to greatly improve performance. We may also take advantage of much more data available from Flickr to create even better representations.

To get a feel of where our descriptors help, we take a look at the confusion matrix and compare it to other features . Table 2 shows the types of images that our features are best at classifying. It is worth noting that these categories are not generally the easiest categories for other hand-engineered features, so our features are essentially providing new information which can be exploited to our advantage when combining all of the features together. We also show on the 3rd column a list of scenes for which tag-sensitive features help the most (compared to our own tag-free descriptors). It is worth noting that some of the features for which we

Tag-Free Descriptors	Most-Improved (by leveraging text)
arrival gate	aquarium
florist shop	beach
racecourse	athletic field
aquarium	wind farm
hayfield	aqueduct

Table 2. Results. Ours is better.

obtain the most benefit by levereging text are beach, aquarium, etc. which are generally correlated with frequently occurring tags such as water, etc.

7. Conclusion

We've introduced a general technique to take the most advantage of vast amounts of unlabeled data with weak annotations that come at no cost to improve performance on tasks of interest.

We establish a proof-of-concept by evaluating our method on a large-scale scene categorization task. We are able to use unsupervised feature learning to learn descriptors that are comparable to state-of-the-art descriptors such as SIFT and GIST. Further, we are able to leverage the weak annotations from Flickr images to our advantage and learn tag-sensitive features that yield even better performance.

We envision our approach to be more generally applicable, and can be used for a wide variety of tasks in which we're able to obtain weakly annotated data for free. For instance, the same method can be applied for video data in which the task is to recognize actions, sat. We may take advantage at weak segmentation that comes for free using motion cues, for example, to improve performance on the action categorization task.

References

[1] J. Xiao, J. Hays, K. Ehinger, A. Oliva, A. Torralba. SUN Database: Large-scale Scene Recognition from Abbey to Zoo. *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[2] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of Data with Neural Networks. *Science*, 313(5876):504–507, 2006.

[3] A. Torralba and A. Oliva Statistics of Natural Image Categories. *Network*, 14:391-412, 2003.

[4] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee and A. Ng. Multimodal Deep Learning *Prc. ICML* , 2011.

[5] R. Raina, A. Battle, B. Packer, H. Lee and A. Ng. Self-taught learning: Transfer learning from unlabeled data. *Prc. ICML* , 2007.

[6] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[7] A. Coates, H. Lee and A. Ng. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. *AISTATS*, 14, 2011.

[8] M. C. Caligaris Unsupervised Multi-modal Learning: Images and Text *CS 229 Final Project*, 2010.

8. Appendix

This course is part of a larger research project in advised by Prof. Andrew Ng. I worked in this project with Andrew Maas and Andre Filgueiras de Araujo. The problem we're trying to tackle is within a bigger context of machine learning - that of learning mearningful representations from vast amounts of unlabeled or weakly annotated data.

In this project, we apply the machine learning technique to a computer vision task: Scene Classification. We use computer vision techniques such as

1. Bag of Words Model in computer vision
2. Spatial Pyramid Matching
3. Similarity Histogram Kernel
4. Evaluated performance of common computer vision descriptors such as SIFT, GIST, etc.
5. Pooling

The ideas for the models were discussed in research meetings and small group meetings. I personally gathered the Flickr dataset, implemented code for the Y-shaped network and the convolutional network, and ran SUN experiments using code provided by Torralba *et al*.

I verify and confirm that I am the sole author of this writeup. As of right now, this is the first paper related to this work. We're planning to improve on these results and apply the technique to other tasks to eventually submit a paper to a machine learning conference such as ICML.