Sparse Image Reconstruction with Matching Pursuit Algorithms

David Klee, Luke Roberto

Abstract

Forming an abstract representation of the visual world allows robots to plan in a simpler state-action space. We propose using sparse coding to describe images as a linear combination of convolutional features. In this paper, three algorithms were implemented to produce dictionaries of convolutional features in an unsupervised manner. Variations of the matching pursuit algorithm were tested on real-world images and it is shown that orthogonal matching pursuit reconstructs the images most effectively. This work demonstrates that representing images as sparse combinations of convolutional feature patches can be used to greatly reduce information density while retaining visual appearance.

1 Introduction

A prominent problem in robotics is learning latent representations that are robust and hold enough information to make decisions. Computer vision relies on learning latent representations because the data is dense and computationally infeasible to make decisions on directly. Many of the current approaches for encoding visual information rely on large amounts of labeled datasets in order to train machine learning models in a supervised manner [1]. A better scenario would be to have the system automatically learn feature spaces that hold important information about the world without labeled data [2]. In this paper, we learn dictionaries of convolutional features using several unsupervised approaches. Then, we use three variants of matching pursuit to reconstruct real-world images as a sparse combination of these features.

2 Technical Approach

The problem of reconstructing a sparse version of a signal given a dictionary can be formulated as the following optimization problem:

$$\min_{z} \|Wz - x\|_2^2 \quad \text{s.t.} \quad \|z\|_0 \leq k \tag{1}$$

Here, the objective function to be minimized is the distance between the signal x and the linear combination of features from the dictionary W, according to the coding z. The constraint forces the coding to use at most k features from the dictionary for the reconstruction. One could attempt to solve this problem using a generalized Lagrange multipliers-based approach, but the constraint is neither differentiable nor convex. The simplest algorithm that has been developed to solve this problem is called Greedy Matching Pursuit (GMP). The algorithm is as follows:

Algorithm 1: Greedy Matching Pursuit

Input: Image (y), Dictionary (W), Sparsity (k)
Output: Sparse Code (S)

S = {};
r = y ;
for *(i = 0 ... k)* **do**
  $j = \text{argmax}_i \|W_i r\|_2^2$;
  $a = W_j x$;
  $r = r - aW_j$;
  $z_j = z_j + a$;
end

This is a greedy algorithm because it selects the feature in the dictionary that has the most overlap with the current residual, and then subtracts the selected feature's contribution to the signal from the residual for the next round.

We explore two other algorithms in this project: Orthogonal Matching Pursuit (OMP) [3] and Convolutional Matching Pursuit (CMP) [4]. OMP differs from GMP in the residual update, the algorithm updates the residuals by projecting the observation on to the linear subspace spanned by the columns that have already been selected. The update in line (6) above would be $r = (I - P)y$, where $P = W^{+}_{j_0..j_i} W^{T}_{j_0..j_i}$. CMP differs by changing the cost function in the original optimization to $\| \sum_{i=1}^{k} w_i * z_i - y \|_2^2$.

In order to make the problem more efficient, we chose to represent the images as an array of patches, rather than a single array of pixel values. This allowed us to create feature dictionaries that were the size of image patches, rather than the size of the entire image. We felt that this change would allow the features to generalize more easily, and it would make reconstruction faster, since features were only checked against patches. This resulted in blocky visual artifacts as seen in the experimental results, but quite good reconstructions otherwise.

3 Experimental Results

The dataset we will be using is the Kitti Dataset [5] that contains video benchmarks for computer vision. There are 27 videos the urban environment dataset that range from 2 min (28 frames) to 45 min (423 frames). The image resolution of the videos are 1392 x 512 pixels, so we cropped the view to the interesting sections of the scenes (200 x 200 pixels) in order to make the problem more computationally tractable.

We believe this is an effective data set to use for this project for two reasons. First, the data set is related to a robotics problem. The KITTI data set was built in order to facilitate research on automated cars so it felt natural to address the problem of forming a condensed representation of the world using this data set. Second, the domain is constrained to be of scenes on a road. Having a limited domain may allow the features in the dictionary to be more specialized and, thus, more powerful.

The experiments were broken into three phases. The first set of experiments were on the dictionaries generated. We wanted to see how various parameters of the patch sizes, number of features, and number of patch samples from the videos affected the types of features learning. The next set of experiments were to establish a baseline on the GMP algorithm. We compared the performance of each dictionary on an entire sequence of images, and also as a function of code length. Finally, we compared the performance of each algorithm to see the quality of reconstructions for a fixed code length and then for a sweep of code lengths.

3.1 Dictionary Generation

The first step is to generate the dictionaries that will be used in the matching pursuit algorithms. We explored several methods of generating dictionaries: principal component analysis (PCA), k-means clustering and dictionary learning. The input for each method was a list of sample patches drawn randomly from the data set and a desired number of features in the dictionary. The output was a list of features the same shape as the sample patches.

PCA was chosen because it decomposes a set of signals into the underlying components that best explain the variance in the data. In this domain, it was anticipated that PCA would generate patches representing basic features like edges or domain-specific features like car parts or lane dashes. PCA was implemented using the scikit-learn package [6]. K-means algorithm minimizes the sum of the distance between every point in the data set and the nearest cluster center. Unlike PCA, where the generated features have an order and are all orthogonal, k-means produces features (i.e. cluster centers) that are unordered and not related to each other. K-means was implemented using scikit-learn [6]. The L_2-norm in the setting of images corresponds to the pixel-wise distance, which is often used in computer vision loss functions. The final algorithm used to generate feature dictionaries is called dictionary learning [7]. The algorithm aims to minimize the reconstruction loss of the signal x using a dictionary D, with a constraint on the

L_1-norm of the sparse coding, as shown in Eqn. 2.

$$l(x, D, \alpha) = \frac{1}{n} \sum_{i=1}^{N} \left(\frac{1}{2} \|x_i - D\alpha_i\|_2^2 + \lambda \|\alpha_i\|_1 \right) \tag{2}$$

This is similar to the matching pursuit algorithm except here the constraint is an L_1-norm on the coding, not an L_0-norm. The algorithm was implemented using an scikit-learn package [6]. The implementation alternates between solving for the best values of α then the best value of D. Because the algorithm generates a dictionary that effectively minimizes reconstruction with a sparse coding, the output will likely perform well using the variants of matching pursuit discussed subsequently.

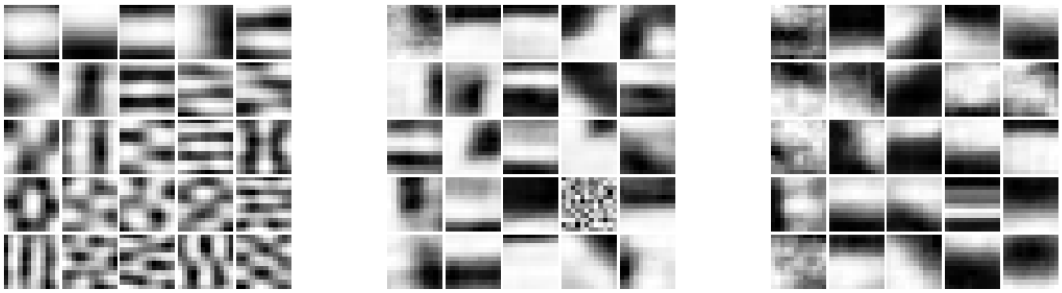

Figure 1: Dictionaries generated with: PCA, DictionaryLearning, and KMeans. The 2d arrays are arranged in decreasing order from left to right (ex. top left of PCA is the largest component, bottom right is the smallest component).

All algorithms were run offline as a pre-processing step before matching pursuit (run times were on the order of minutes so online dictionary learning was deemed untenable for this project). All features were normalized in order to facilitate fair comparisons between the three methods. An example of the feature dictionaries generated by the methods is shown in Fig. 1 for the case of using 12 by 12 pixel patches as signals. The feature dictionary generated by PCA clearly has an order and the patches range from a dot and simple edges to basic textures. However, dictionary learning and k-means produces dictionaries seem to produce variations of edges and corners.

The differences in the feature dictionaries can be noticed visually and affect reconstruction. In Fig. 2, the three dictionaries (from Fig. 1) are compared in terms of their ability to reconstruct sample patches. To create the plot, 1,000 image patches of the same size (12 by 12 pixels) are sampled (from the same video as used for dictionary generation) and the k best features are used to reconstruct the image for values of k from 1 to 25. The fraction of reconstruction is calculated using Eqn. 3 and averaged over all 1,000 samples.

$$L_{recon} = \frac{\|X_{true}\|_2^2 - \|X_{true} - X_{recon}\|_2^2}{\|X_{true}\|_2^2} \tag{3}$$

From Fig. 2, we can see that the PCA dictionary greatly outperforms the other two dictionaries when more than a few features are used. However, when only 1 or 2 features are used for reconstruction, dictionary learning and k-means dictionaries perform better, with k-means having a slight edge. In the case of reconstructing an entire signal with a sparse coding, PCA features seem best because 80% reconstruction can be achieved on a 12 by 12 pixel signal (144 dimensional) using only 20 features. Yet, in this paper, we are considering the case where the features are patches of a larger image that is being reconstructed. Thus, it is very uncommon that more than a few features will be used to reconstruct the same patch of the image due to how large the image is and how few features can be used in the sparse coding. For this reason, it is hypothesized that the dictionaries produced by k-means or dictionary learning methods will perform better for matching pursuit.

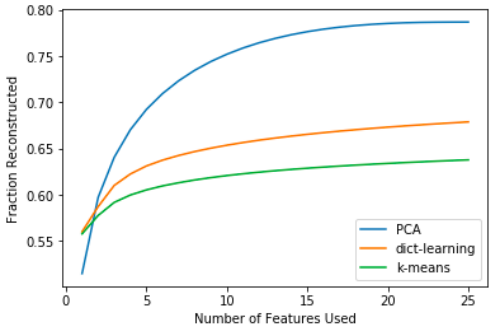

Figure 2: Reconstruction Performance of Feature Dictionaries. For each feature dictionary shown in Fig. 1, this is a plot of the fraction of individual sample patches that can be reconstructed using different numbers of features from the dictionary.

3.2 Dictionary Comparison

The effectiveness of each dictionary at reconstructing individual patches is different from what we are after: effectiveness at reconstructing an entire image. Thus, we came up with a quantitative comparison by running GMP using the three dictionaries. The metric used for comparison is the fraction of reconstruction from Eqn. 3. The samples to be reconstructed were taken from a 114 frame-long video in the KITTI dataset after cropping as detailed in the Experimental Results section.

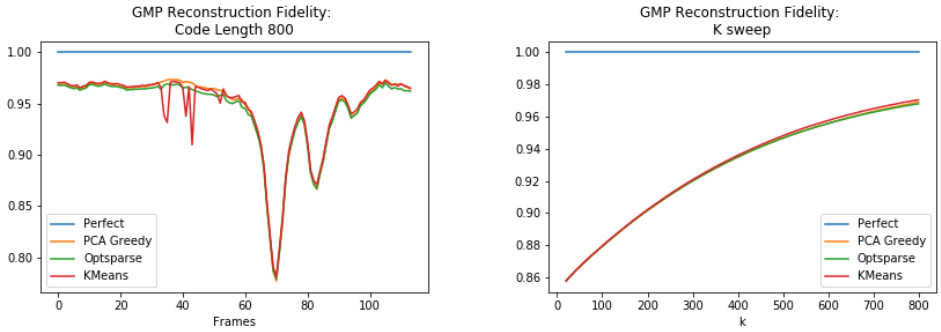

Figure 3: **Left**: Comparison of each dictionary's reconstruction fidelity over an entire video (114 frames). **Right**: Dictionary reconstruction for a sweep of the length of the sparse code from 0 to 800.

Fig. 3 shows a comparison of each dictionary's performance under the GMP algorithm. The left plot shows reconstruction ability over a single video. Throughout the video, we see that reconstruction varies quite a bit depending on the scene. We found that the 3 consistent dips at frames 70, 85, and 95 corresponded to shadows on the streets that had complex textures for reconstruction. Otherwise, the three dictionaries performed quite similarly, noting that the k-means clustered dictionary seems to be a bit unstable at some points in its reconstruction quality.

The figure on the right shows a sweep from a code length of 20-800 for each of the dictionaries under GMP for a single frame. It is interesting to note here that the PCA features perform slightly better than the rest (contrary to the hypothesis at the end of the previous section), but overall performance varies on slightly. It also makes sense for the curve to look logarithmic, as it is quite easy to make out the textureless and geometrically simple parts of a scene, but then much more effort needs to be put in to recreate the more complex textures and shapes.

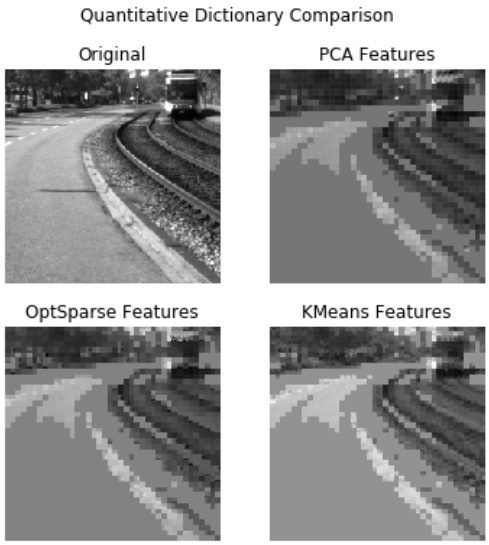

Figure 4: Example frame reconstruction for each dictionary.

Fig. 4 shows an example of a reconstructed image for each dictionary feature set. This gave a good qualitative comparison of each algorithm for debugging and other purposes.

3.3 Algorithm Comparison

Once we compared the performance of the dictionaries, we wanted to do a comparison of the algorithms as well. The following figures will show the algorithm comparison for the PCA features, due to constraints on document length. The provided Git repository contains the files that were used to generate the plots, and can easily be changed to see the results for other feature types.

We performed similar experiments as before, first comparing performance over a video with fixed code length and then on a single frame as a function of the code length.

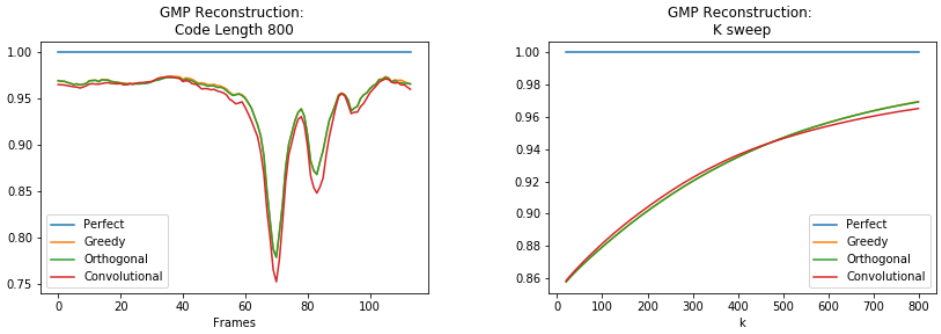

Figure 5: **Left**: Comparison of each algorithm's reconstruction ability over an entire video (114 frames). **Right**: Dictionary reconstruction for a sweep of the length of the sparse code from 0 to 1000.

Fig. 5 shows the quantitative plots for the algorithm comparison. Both figures show similar curves to before, with the same explanations for the shapes of the curves. We do note that the performance variation is slightly more significant than for the dictionaries, with OMP performing the best. This makes

sense because OMP was designed as a slightly more computationally intensive version of GMP that had better performance guarantees.

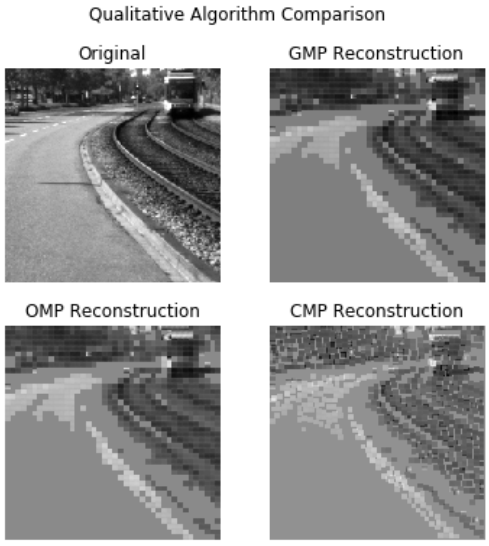

Figure 6: Comparison of algorithm reconstruction for code length of $k = 800$.

4 Conclusion

This paper explores three variants of the matching pursuit algorithm for selecting feature patches to reconstruct an image. The feature dictionaries were generated using three unsupervised learning techniques: PCA, k-means clustering, and a task-relevant method called dictionary learning. The three dictionaries generated similar performance when used for matching pursuit with a limited length of coding (¡800 features for an image of 40,000 pixels). Orthogonal matching pursuit outperformed the other methods by a small margin. This work showed that significant information reduction can be made on real-world, limited-domain images by reconstructing them as a sparse linear combination of convolutional features.

5 Participants Contribution

There were two participants in this project: David Klee and Luke Roberto.

Luke was responsible for some of the dataset utilities involved in loading the videos files and converting them into patches. He wrote up the implementations for GMP and OMP, and generated the plots for the dictionary and algorithm comparisons. David wrote the functions for generating and visualizing feature dictionaries as well as the convolutional matching pursuit algorithm. They collaborated on the writing of the report.

References

[1] Y. Romano and M. Elad, "Convolutional Neural Networks Analyzed via Convolutional Sparse Coding."

[2] K. Grauman, "See, Hear, Move: Towards Embodied Visual Perception," 2018.

[3] T. T. Cai and L. Wang, "Orthogonal matching pursuit for sparse signal recovery with noise," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4680–4688, 2011.

[4] A. Szlam, K. Kavukcuoglu, and Y. Lecun, "Convolutional Matching Pursuit and Dictionary Training," pp. 1–7.

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 689–696.

[8] "MATCHING PURSUIT BASED CONVOLUTIONAL SPARSE CODING Elad Plaut and Raja Giryes School of Electrical Engineering , Tel Aviv University , Tel Aviv , Israel," no. 1, pp. 6847–6851, 2018.

[9] Yann, ""CBLL, Research Projects, Computational and Biological Learning Lab, Courant Institute, NYU"." [Online]. Available: https://cs.nyu.edu/{~}yann/research/sparse/index.html

[10] F. Bergeaud, S. Mallat, M. Street, N. York, E. C. Paris, A. M. L, and F. Chiitenay-malabry, "Stkphane Mallat," pp. 53–56, 1995.

[11] "Matching Pursuit Algorithms - MATLAB & Simulink." [Online]. Available: https://www.mathworks.com/help/wavelet/ug/matching-pursuit-algorithms.html