Michael Doescher
Eugene Yoong
Due Monday April 20, 2015
Computer Vision: Project 3

Locality-Constrained Linear Coding for Scene Classification

Scene classification is a classical problem in computer vision which addresses the challenge of assigning a semantic label to images. In our project we build upon the spatial pyramid matching work of Lazebnik et al.[1] and ultimately implement Wang et al.'s Locality-constrained Linear Coding for Image Classification algorithm.[2] Performance of these two algorithms is compared experimentally on Lazebnik's fifteen-class natural scene dataset.[3]

Spatial Pyramid Matching (SPM): Lazebnik's spatial pyramid algorithm[1] operates by transforming each image in a semantically labeled training set into a feature vector suitable for use with a standard support vector machine (SVM) for classification. This transformation occurs by first representing the image as a collection of local feature points or descriptors; in our project we use SIFT features. These descriptors are transformed into an index of a codebook which represents the closest vector to the feature in the codebook. The codebook is generated by the k-means clustering algorithm on the collection of descriptors from the training data to generate the desired number of codebook entries. The algorithm doesn't just use these features as a simple bag of words but instead encodes these features into spatial pyramids. The pyramids are constructed by placing a grid over a feature space in an increasing sequence of coarseness. At the bottom level of a spatial pyramid the image is subdivided into 2^{p-1} by 2^{p-1} small regions where p represents the number of levels in the pyramid and a histogram of the encoding vectors is built over the codebook entries for all features that fall in each region. The weighted sum of the number of matches, which are two points that occur in the same cell of the grid, at each level of resolution is taken, with finer resolution matches being given a higher weight. Each level of the "pyramid" is then pooled together using the sum pooling method, whereby the codes for each sub-region are concatenated by summing the codes together, to create a final image 'feature vector'.[1] Specifically, SPM tries to solve the following optimization problem:

$$y = \arg min_c \sum_{i=0}^{N} \|x_i - Bc_i\|^2 \;\; s.t. \|c_i\|_{l^0} = 1, \|c_i\|_{l^1} = 1, c_i \geq 0, \forall i$$

These training feature vectors are input into a SVM to generate a classifier. For our project we use the open source LIBLINEAR library which supports linear support vector machines as well as logistic regression and is designed for use with large datasets.[4] Test images are subjected to the same spatial pyramid histogram based transformation as the training images using the same codebook generated from the training images and are then presented to the SVM for classification. When multiple image categories are available for classification, one SVM is prepared for each category as a one vs. everything else classifier. The image is tested against each SVM and the SVM reporting the highest probability of success is chosen as the image category.[1]

Approximated Locality-constrained Linear Coding (LLC): The approximated LLC algorithm instead tries to solve the following constraint problem:

$$\min_C \sum_{i=0}^{N} \|x_i - Bc_i\|^2 \; s.t. \; 1^T c_i = 1, \forall i$$

The locality-constrained linear coding algorithm upgrades the quantized vectors used in SPM to use a vector in terms of a local basis of codebook entries. For each descriptor, the k-nearest neighbors in the codebook are determined and a linear system of equations is solved to generate the encoding. These vectors are also compiled into spatially localized histograms which are concatenated together by using the max pooling method, whereby the max of the codes is taken as the representative of the sub-region, as opposed to the sum pooling used in SPM.

Implementation Overview:

The natural scene dataset, SPM code, LIBLINEAR, and demo code to solve equation 7 in the LLC paper were provided in the assignment description. We are submitting three versions of a 'main' function which implement either the SPM, the approximated LLC algorithm, or the LLC algorithm with codebook optimization based on algorithm 4.1 of Wang et al.[2]: main_SPM.m, main_LLC.m, and main_Optimized_LLC.m. These methods work fundamentally the same way. They take a directory of directories of semantically sorted images. They select at random 100 images from each directory to form a training set and the rest of the images represent the test set. The script then calls buildcodebook.m which generates a collection SIFT descriptors for each image in the training set and then runs k-mean on these descriptors to select the best k vectors to represent the composite collection. We built the codebook using 200 and 1024 centers. The optimized codebook improves on this k-means generated codebook by employing algorithm 4.1. Main continues by building pyramids for each image through two more steps first SIFT descriptors are generated for each image and then encoded using the codebook either by finding the nearest neighbor in the codebook as in vector quantization in the SPM model or by using the nearest k neighbors (we use k=5) and solving a system of equations to generate the encoding as in LLC. Finally the encoded descriptors are pooled and normalized either by sum pooling in SPM or max pooling in LLC and finally compiled into a pyramid. Main then includes functionality to train a SVM for each category using the training data and classify the each image in the testing data to ultimately calculate accuracy of the model and generate a confusion matrix.

We implement image classification using approximated LLC[2] where our project modifies the provided spatial pyramid code by Lazebnik et al. using Matlab[1]. We implement LLC by modifying the provided spatial pyramid code which includes the following two modifications.

1. The hard codeword assignment used by the spatial pyramid algorithm is adapted to locality constrained linear coding as used by the LLC algorithm as in the paper. This required modifying the BuildHistograms class. k-nearest neighbors identifies the nearest 5 codebook entries to each feature vector and these entries are used to generate local basis. Then the

encoding vectors are solved for using code inspired by equations 5 and 6 of Wang et al.[2]. These vectors are stored as a vector of nonzero values and the indices of these values to save space.

2. The sum pooling method used by the spatial pyramid algorithm in the CompilePyramid script is replaced by max pooling with l^2 normalization for the LLC algorithm. This method adapts to the new data structure used by the LLC version of BuildHistogams by reconstructing the complete non-sparse encoding vectors and then taking the maximum value of each feature in the vector over each spatial region in the lowest level of the pyramid. The pyramid formation at higher levels also uses the max function instead of a summation.

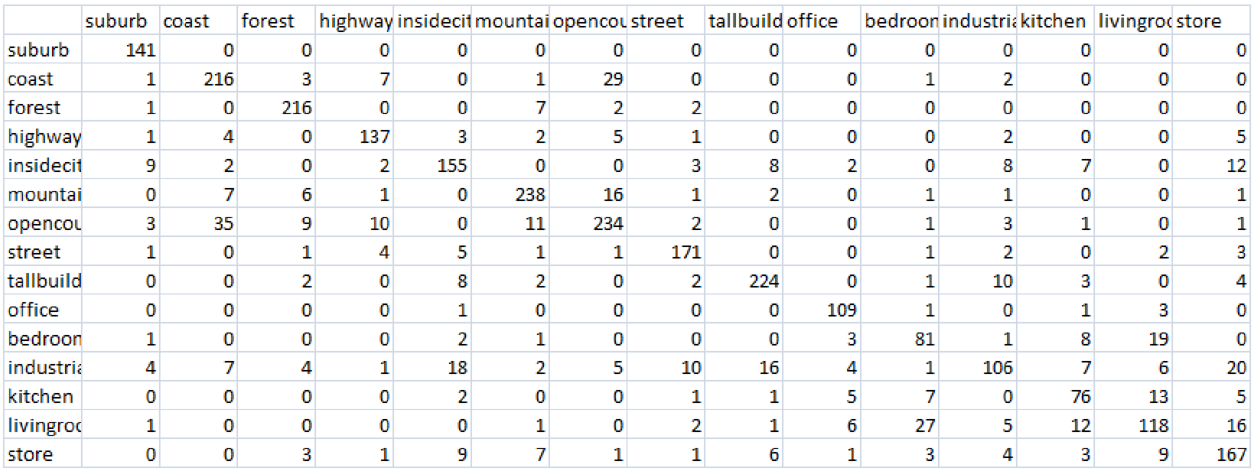
Classification: We classify the features that were extracted using the spatial pyramid and LLC algorithms using a SVM. The SVM implementation that we use is LIBLINEAR in Matlab which supports multi-class classification. For each semantic category we train a model on our training data and use the model to classify our test data. Using the predicted classifications, we compute the accuracy of the predictions for each category and construct the confusion matrix.

Experimental Methods: The directory of images contained approximately 300 images each in 15 different categories. We separated these images into a training set and a testing set. The training set contained 100 randomly selected images from each of the categories and the testing set contained the remaining images.

We tested our SPM and approximated LLC algorithms using dictionary sizes of 200 and 1024, and 1, 2, and 3 pyramid levels. The dictionary was constructed using all 1500 training images. For approximated LLC we use 5 nearest neighbors to solve the LLC problem and generate the codebook.

Results: Using the spatial pyramid for classification we obtain the following confusion matrix when classifying over the scene categories in our dataset. This example matrix is shown for the LLC method using 1024 codebook entries with 2 levels in the pyramid. Other confusion matrices for different combinations of parameters can be found in the appendix.

Figure 1: Confusion Matrix (LLC 1024 dictionary entries, 2 pyramid levels)



	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industri	kitchen	livingroc	store
suburb	141	0	0	0	0	0	0	0	0	0	0	0	0	0	0
coast	1	216	3	7	0	1	29	0	0	0	1	2	0	0	0
forest	1	0	216	0	0	7	2	2	0	0	0	0	0	0	0
highway	1	4	0	137	3	2	5	1	0	0	0	2	0	0	5
insidecit	9	2	0	2	155	0	0	3	8	2	0	8	7	0	12
mountai	0	7	6	1	0	238	16	1	2	0	1	1	0	0	1
opencou	3	35	9	10	0	11	234	2	0	0	1	3	1	0	1
street	1	0	1	4	5	1	1	171	0	0	1	2	0	2	3
tallbuild	0	0	2	0	8	2	0	2	224	0	1	10	3	0	4
office	0	0	0	0	1	0	0	0	0	109	1	0	1	3	0
bedroon	1	0	0	0	2	1	0	0	0	3	81	1	8	19	0
industri	4	7	4	1	18	2	5	10	16	4	1	106	7	6	20
kitchen	0	0	0	0	2	0	0	1	1	5	7	0	76	13	5
livingroc	1	0	0	0	0	1	0	2	1	6	27	5	12	118	16
store	0	0	3	1	9	7	1	1	6	1	3	4	3	9	167

We observed that while SPM did perform adequately on many categories. It consistently failed to classify industrial scenes no matter what parameters we chose. We speculate the features available in industrial images vary more than typical features in the other scenes. The LLC classifier did perform quite well but struggled on some categories including the challenging industrial scenes. We further observed that indoor scenes such as the kitchen, living room, and bedroom were more likely to be misclassified as one another while outdoor scenes were more likely to be misclassified as a different outdoor scene.

Figure 2: Overall Accuracy

Method	Dictionary Size	Pyramid Levels	Accuracy
SPM	200	1	0.5360
SPM	200	2	0.5002
SPM	200	3	0.4586
SPM	1024	1	0.4767
SPM	1024	2	0.4523
SPM	1024	3	0.4281
LLC	200	1	0.5980
LLC	200	2	0.7189
LLC	200	3	0.7595
LLC	1024	1	0.7059
LLC	1024	2	0.7762
LLC	1024	3	**0.8003**

We observed that overall accuracy actually decreased when more pyramid levels were included using the SPM method. Accuracy also decreased when the dictionary size was increased with the original SPM method. Both of these observations seem counterintuitive. The linear SVM classifier clearly does not perform well with these data sets. However, with increasing dictionary size and with increasing number of levels the LLC method does perform more accurate classification. Further in all cases the LLC method outperforms the SPM method.

Figure 3: Accuracy by class

	SPM	SPM	SPM	SPM	SPM	SPM
	200	200	200	1024	1024	1024
	1	2	3	1	2	3
suburb	0.7518	0.6383	0.5461	0.6241	0.4823	0.4468
coast	0.7538	0.7269	0.6577	0.6923	0.6346	0.5885
forest	0.9737	0.9781	0.9781	0.9605	0.9518	0.9298
highway	0.7312	0.8125	0.8313	0.7438	0.7688	0.7875
insidecity	0.5240	0.4519	0.4135	0.4712	0.4087	0.3798
mountain	0.5657	0.3978	0.2518	0.3212	0.2117	0.1387
opencountry	0.3839	0.3903	0.3613	0.2387	0.2774	0.2903
street	0.4323	0.5052	0.4688	0.4688	0.4844	0.5052
tallbuilding	0.7656	0.6523	0.5742	0.7461	0.7148	0.6406
office	0.7565	0.7391	0.7565	0.8435	0.8696	0.8522
bedroom	0.3103	0.3362	0.3362	0.2069	0.2931	0.3276
industrial	0.0047	0.0095	0.0284	0.0095	0.0047	0.0000
kitchen	0.5545	0.5636	0.5364	0.7091	0.7000	0.6636
livingroom	0.0582	0.0529	0.0582	0.0212	0.0106	0.0212
store	0.4698	0.3488	0.2744	0.3302	0.2698	0.2000
	LLC	LLC	LLC	LLC	LLC	LLC
	200	200	200	1024	1024	1024
	1	2	3	1	2	3
suburb	0.8723	0.9362	0.9716	0.9929	1.0000	1.0000
coast	0.8192	0.8346	0.8462	0.8654	0.8269	0.8308
forest	0.8816	0.9342	0.9693	0.8991	0.9298	0.9474
highway	0.7500	0.7562	0.8063	0.8375	0.8750	0.8562
insidecity	0.5769	0.7500	0.7548	0.6971	0.7837	0.7452
mountain	0.7774	0.8029	0.8285	0.8358	0.8394	0.8686
opencountry	0.4323	0.6387	0.6387	0.6000	0.7355	0.7548
street	0.6719	0.8229	0.8125	0.8698	0.8802	0.8906
tallbuilding	0.8516	0.8906	0.8359	0.8633	0.8477	0.8750
office	0.7391	0.8957	0.8783	0.8696	0.9217	0.9478
bedroom	0.4138	0.6034	0.6552	0.4914	0.6379	0.6983
industrial	0.1659	0.3318	0.4787	0.3412	0.5024	0.5024
kitchen	0.3364	0.4364	0.5727	0.5273	0.5818	0.6909
livingroom	0.1270	0.4339	0.6296	0.2275	0.5344	0.6243
store	0.3953	0.6047	0.6884	0.5814	0.7023	0.7767

Extensions

1. Parallelization

We leverage Matlab's Parallel Computing Toolbox to parallelize the several portions of the required computation. The BuildPyramid script is called in a loop to operate on each directory of images one at a time. This batch computation of pyramids for each image in a directory is run in a parallel for loop and we observe a dramatic increase in speed performing this processing in parallel. Further, we perform the model building for each of the one vs. everything else classifiers in parallel. In all we observed approximately an eight fold increase in speed over the single threaded code.

Figure 4: Build pyramid Run times for all images in all fifteen directories

Method	Dictionary Size	Time
SPM	200	9 minutes 36 seconds
SPM	1024	28 minutes 1 second
LLC	200	66 minutes 48 seconds
LLC	1024	143 minutes 1 second

2. Parameter Variation

We compare and contrast the results given by LLC by varying the size of the codebook and the number of pyramid level. For the codebook size, we use a size of 200 which is suggested to be optimal for the SPM method and 1024 which is suggested to be optimal for the LLC method. We run SPM and LLC on both codebook sizes and compare their performances. Finally, we vary the number of pyramid levels for both methods and compare their performances. Please see the results section for a complete discussion of these experiments.

3. Optimized Codebook

We implemented a batch processing method to produce and optimized codebook obtained by solving LLC. This is done by first computing a codebook using k-means then solving LLC for each descriptor representation. Then using the equations provided in Algorithm 4.1 of the LLC paper[2], we update the basis entries of the codebook generated by k-means to produce the LLC codebook. BuildHistograms and CompilePyramid are then run using the LLC codebook to generated the image feature representation. We did not observe significant changes in the accuracy during classification.

References

1. Lazebnik, S., Schmid, C., Ponce, J. (2006) Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. CVPR.

2. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y. (2010) Locality-constrained Linear Coding for image classification. CVPR.

3. Natural Scene Dataset. Retrieved from www.cs.unc.edu/~lazebnik/research/scene-categories.zip.

4. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J. LIBLINEAR: A Library for Large Linear Classification. www.csie.ntu.edu.tx/~cjlin/papers/liblinear.pdf and www.csie.ntu.edu.tw/~cjlin/liblinear/.

Contributions:

Eugene

1. LLCSolver.m
2. BuildLLCCodebook.m
3. buildcodebook.m
4. Subfolder recursion for main_LLC and main_SPM
5. CalculateDictionaryLLC.m
6. Wrote first draft of the paper

Mike

1. Implemented the basic pipeline including both SPM and LLC versions
2. Modified BuildHistogram.m and CompilePyramid.m to implement LLC
3. Developed experimental procedure and developed code to run our experiments
4. Revised the code to parallelize using Matlab's Parallel Computing Toolbox.
5. Data Analysis.
6. Final draft of the paper.

Appendix A: Confusion Matrices

Algorithm = SPM; Dictionary Size = 200; Number of Pyramid Levels = 1

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industri	kitchen	livingroc	store
suburb	106	2	9	9	1	2	6	0	2	0	0	0	0	0	4
coast	2	196	4	42	0	0	10	0	0	1	5	0	0	0	0
forest	0	0	222	1	0	4	1	0	0	0	0	0	0	0	0
highway	0	20	3	117	0	4	3	1	0	2	0	0	1	0	9
insidecit	5	14	7	13	109	1	2	4	18	3	0	0	20	0	12
mountai	2	13	42	35	0	155	19	1	5	1	1	0	0	0	0
opencou	10	60	88	24	0	8	119	0	1	0	0	0	0	0	0
street	0	2	14	34	7	7	0	83	30	1	4	0	0	1	9
tallbuild	2	5	8	16	4	6	0	3	196	2	7	0	3	2	2
office	0	2	0	3	4	0	0	0	1	87	8	0	9	0	1
bedroon	1	3	3	6	0	2	1	0	5	19	36	0	31	4	5
industri	15	30	11	27	14	7	4	7	32	10	12	1	10	1	30
kitchen	0	1	0	0	8	1	0	0	3	28	6	0	61	1	1
livingroc	1	7	2	2	2	3	0	0	10	38	26	0	64	11	23
store	4	2	43	0	21	8	2	1	9	5	1	0	16	2	101

Algorithm = SPM; Dictionary Size = 200; Number of Pyramid Levels = 2

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industri	kitchen	livingroc	store
suburb	90	4	19	13	0	1	9	0	3	0	0	0	0	0	2
coast	1	189	4	50	0	0	11	0	0	1	4	0	0	0	0
forest	1	0	223	4	0	0	0	0	0	0	0	0	0	0	0
highway	0	10	3	130	0	2	4	1	0	1	0	0	0	0	9
insidecit	3	15	12	19	94	0	2	5	18	3	0	0	26	0	11
mountai	2	18	55	59	0	109	23	0	5	1	2	0	0	0	0
opencou	3	53	94	34	0	4	121	0	1	0	0	0	0	0	0
street	3	2	15	37	5	1	2	97	17	1	2	1	1	0	8
tallbuild	0	9	17	33	4	5	1	1	167	3	8	2	4	1	1
office	0	3	0	8	5	0	0	0	0	85	2	0	11	0	1
bedroon	0	5	6	5	0	1	1	0	3	21	39	0	27	5	3
industri	11	35	19	32	11	5	3	5	29	10	12	2	11	1	25
kitchen	0	1	0	1	7	1	0	0	4	26	7	0	62	1	0
livingroc	0	8	6	2	0	1	0	0	5	29	35	0	72	10	21
store	0	3	81	0	17	3	2	1	8	5	2	0	17	1	75

Algorithm = SPM; Dictionary Size = 200; Number of Pyramid Levels = 3

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industri	kitchen	livingroc	store
suburb	77	3	20	24	0	0	14	1	1	0	0	0	0	0	1
coast	0	171	4	70	0	0	11	0	0	2	2	0	0	0	0
forest	0	0	223	4	0	0	0	0	0	0	0	1	0	0	0
highway	0	8	6	133	0	0	5	0	0	1	0	0	0	0	7
insidecit	3	13	16	28	86	1	2	4	19	6	1	0	21	0	8
mountai	2	17	56	96	0	69	25	0	7	1	1	0	0	0	0
opencou	6	44	92	54	0	1	112	0	1	0	0	0	0	0	0
street	2	0	16	55	4	1	1	90	17	1	1	0	1	0	3
tallbuild	0	11	21	57	2	1	2	2	147	0	7	1	4	0	1
office	0	6	0	11	1	0	0	0	0	87	0	0	10	0	0
bedroon	0	5	6	10	0	0	2	1	2	19	39	0	25	4	3
industri	4	38	27	50	6	5	2	2	25	8	8	6	11	2	17
kitchen	0	1	0	2	5	1	0	1	6	24	10	0	59	1	0
livingroc	0	7	9	8	0	1	0	0	2	33	34	0	67	11	17
store	0	2	99	0	15	2	2	2	9	5	3	0	17	0	59

Algorithm = SPM; Dictionary Size = 1024; Number of Pyramid Levels = 1

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industri	kitchen	livingroc	store
suburb	88	3	21	6	3	0	2	13	1	0	0	0	1	0	3
coast	3	180	4	47	0	0	21	3	0	1	1	0	0	0	0
forest	0	0	219	4	0	0	1	3	1	0	0	0	0	0	0
highway	1	18	2	119	0	3	3	3	4	0	1	0	0	0	6
insidecit	4	11	7	12	98	1	1	12	17	6	0	2	28	1	8
mountai	5	12	66	52	0	88	26	14	7	1	2	1	0	0	0
opencou	17	66	101	30	0	8	74	8	4	1	1	0	0	0	0
street	3	7	10	42	6	1	1	90	19	3	0	0	2	1	7
tallbuild	3	3	5	20	2	2	1	11	191	4	1	0	8	2	3
office	0	3	0	0	1	0	0	3	0	97	1	0	10	0	0
bedroon	0	6	5	4	0	0	1	2	5	22	24	0	42	1	4
industri	7	36	10	22	13	4	8	20	35	13	6	2	16	3	16
kitchen	0	0	0	0	5	0	0	1	2	19	2	0	78	1	2
livingroc	2	7	5	2	3	1	0	1	10	36	16	0	85	4	17
store	1	4	60	0	29	6	0	2	16	7	3	0	16	0	71

Algorithm = SPM; Dictionary Size = 1024; Number of Pyramid Levels = 2

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industri	kitchen	livingroc	store
suburb	68	2	20	19	4	1	1	22	1	0	0	0	1	0	2
coast	3	165	3	56	0	0	26	4	1	1	1	0	0	0	0
forest	0	0	217	5	0	1	1	3	1	0	0	0	0	0	0
highway	1	18	2	123	2	0	3	4	3	0	0	0	0	0	4
insidecit	3	13	7	17	85	1	1	13	19	6	0	1	39	0	3
mountai	5	17	60	80	0	58	28	18	6	0	2	0	0	0	0
opencou	12	65	79	48	0	3	86	11	6	0	0	0	0	0	0
street	2	7	8	46	6	1	0	93	17	3	0	0	3	0	6
tallbuild	3	4	5	32	0	1	1	12	183	3	0	0	11	1	0
office	1	3	0	1	1	0	0	2	0	100	0	0	7	0	0
bedroon	0	6	4	4	0	0	1	2	5	21	34	0	36	1	2
industri	8	37	10	32	13	2	8	19	37	12	6	1	16	2	8
kitchen	0	0	0	0	2	0	0	1	5	21	2	0	77	1	1
livingroc	2	7	4	3	2	1	0	3	10	36	16	0	94	2	9
store	1	6	66	0	32	3	2	2	14	8	3	0	20	0	58

Algorithm = SPM; Dictionary Size = 1024; Number of Pyramid Levels = 3

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industri	kitchen	livingroc	store
suburb	63	1	12	32	4	0	1	26	0	0	0	0	1	0	1
coast	3	153	2	66	0	0	28	5	1	1	1	0	0	0	0
forest	2	0	212	8	0	0	3	3	0	0	0	0	0	0	0
highway	1	16	2	126	2	1	3	4	3	0	0	0	0	0	2
insidecit	4	14	5	25	79	1	1	16	21	8	0	0	34	0	0
mountai	7	15	46	105	0	38	35	21	6	0	1	0	0	0	0
opencou	15	60	64	61	0	1	90	15	4	0	0	0	0	0	0
street	3	2	6	63	4	0	0	97	12	0	0	0	1	0	4
tallbuild	2	4	3	51	0	1	1	17	164	5	0	0	8	0	0
office	1	3	0	4	1	0	0	3	0	98	0	0	5	0	0
bedroon	1	7	4	6	0	0	1	3	6	18	38	0	30	1	1
industri	8	33	9	46	9	1	8	22	39	8	4	0	18	1	5
kitchen	0	0	0	0	1	0	0	4	6	21	3	0	73	1	1
livingroc	2	7	4	5	2	1	0	3	9	39	19	0	89	4	5
store	0	8	65	4	31	2	3	3	16	8	5	0	27	0	43

Algorithm = LLC; Dictionary Size = 200; Number of Pyramid Levels = 1

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industria	kitchen	livingroc	store
suburb	123	0	0	1	2	0	2	1	2	3	0	2	0	1	4
coast	0	213	5	14	0	8	15	0	0	3	0	1	0	0	1
forest	2	1	201	1	0	14	3	2	0	0	0	0	0	0	4
highway	3	19	1	120	3	3	0	2	0	1	0	1	1	0	6
insidecit	14	0	0	6	120	0	1	8	21	4	1	5	9	1	18
mountai	2	14	13	14	0	213	12	1	3	0	0	0	0	1	1
opencou	9	90	23	19	1	23	134	3	3	0	1	1	0	0	3
street	11	0	0	17	7	4	2	129	12	1	0	3	1	2	3
tallbuild	1	0	2	2	8	1	0	8	218	0	5	2	2	2	5
office	1	0	0	3	3	0	0	3	0	85	6	0	12	1	1
bedroon	3	0	0	3	3	2	0	2	4	17	48	1	20	5	8
industria	12	7	3	12	17	4	1	17	35	13	13	35	10	5	27
kitchen	3	0	0	2	5	0	0	3	3	32	12	1	37	7	5
livingroc	7	0	0	1	9	1	0	6	9	28	40	3	34	24	27
store	4	0	8	6	24	11	1	18	7	9	14	4	14	10	85

Algorithm = LLC; Dictionary Size = 200; Number of Pyramid Levels = 2

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industria	kitchen	livingroc	store
suburb	132	0	0	0	0	0	0	3	0	0	3	0	0	1	2
coast	0	217	4	7	0	3	24	1	0	2	1	1	0	0	0
forest	0	0	213	0	1	8	4	0	0	0	0	0	0	0	2
highway	5	18	0	121	3	1	0	1	0	0	1	3	0	0	7
insidecit	10	1	0	0	156	0	0	10	8	0	0	10	6	0	7
mountai	4	14	6	3	0	220	16	2	4	0	3	1	0	0	1
opencou	8	49	23	10	0	12	198	1	1	0	0	1	0	1	6
street	5	0	1	7	6	2	1	158	5	0	1	4	0	1	1
tallbuild	0	0	2	0	4	2	0	1	228	1	0	11	3	1	3
office	0	0	0	0	1	0	0	1	0	103	3	1	2	3	1
bedroon	2	0	0	0	3	1	1	1	0	14	70	2	5	13	4
industria	12	3	2	5	21	5	6	5	24	13	3	70	7	5	30
kitchen	1	0	0	0	5	0	0	1	1	14	10	2	48	19	9
livingroc	2	0	0	0	1	1	0	1	6	13	39	1	21	82	22
store	2	0	4	2	16	6	1	3	5	1	5	8	14	18	130

Algorithm = LLC; Dictionary Size = 200; Number of Pyramid Levels = 3

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroon	industria	kitchen	livingroc	store
suburb	137	0	0	0	2	0	0	0	0	0	1	0	0	1	0
coast	0	220	3	8	0	3	20	1	0	2	0	3	0	0	0
forest	0	0	221	0	0	6	0	0	0	0	0	0	0	0	1
highway	1	8	0	129	3	0	2	3	0	0	1	3	1	1	8
insidecit	3	1	0	1	157	0	0	8	7	0	0	12	10	3	6
mountai	0	9	6	4	0	227	17	0	3	0	1	5	0	0	2
opencou	6	48	14	12	0	17	198	3	0	0	0	4	1	2	5
street	2	0	0	10	5	2	2	156	4	0	1	6	0	1	3
tallbuild	0	0	0	1	5	1	0	1	214	1	0	24	4	2	3
office	0	0	0	0	0	0	0	1	0	101	2	2	6	0	3
bedroon	0	0	0	0	1	0	0	3	0	8	76	3	4	18	3
industria	2	2	2	2	14	6	5	6	16	8	7	101	6	7	27
kitchen	0	0	0	0	0	1	0	0	0	9	12	2	63	18	5
livingroc	1	0	0	0	4	1	0	2	3	9	24	4	11	119	11
store	0	1	2	1	13	4	0	2	4	5	3	10	10	12	148

Algorithm = LLC; Dictionary Size = 1024; Number of Pyramid Levels = 1

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroom	industria	kitchen	livingroo	store
suburb	140	0	0	1	0	0	0	0	0	0	0	0	0	0	0
coast	1	225	5	4	1	6	17	1	0	0	0	0	0	0	0
forest	0	0	205	0	0	15	3	4	0	0	0	0	0	0	1
highway	0	5	0	134	2	4	5	4	0	0	0	1	0	0	5
insidecit	11	1	0	1	145	0	0	14	15	0	2	6	2	0	11
mountai	1	12	9	4	0	229	13	3	2	0	1	0	0	0	0
opencou	7	52	20	7	0	34	186	2	0	0	0	1	0	0	1
street	7	0	1	4	5	2	0	167	2	0	0	2	0	0	2
tallbuild	2	0	2	0	5	3	0	13	221	0	2	4	0	0	4
office	0	0	0	0	1	0	0	0	0	100	2	0	10	1	1
bedroom	1	0	0	0	1	0	0	1	1	14	57	3	23	11	4
industria	10	1	6	6	16	3	3	12	28	7	14	72	3	5	25
kitchen	1	0	0	0	2	0	0	2	3	10	15	1	58	10	8
livingroo	3	0	0	1	4	1	0	7	3	25	45	3	28	43	26
store	7	0	6	0	19	10	2	6	10	5	4	6	10	5	125

Algorithm = LLC; Dictionary Size = 1024; Number of Pyramid Levels = 2

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroom	industria	kitchen	livingroo	store
suburb	141	0	0	0	0	0	0	0	0	0	0	0	0	0	0
coast	0	215	4	3	0	3	32	1	0	0	0	2	0	0	0
forest	1	0	212	0	0	9	3	2	0	0	0	0	0	0	1
highway	1	3	0	140	3	1	4	2	0	0	0	3	0	1	2
insidecit	6	1	0	1	163	0	0	8	8	2	0	10	0	0	9
mountai	1	9	8	3	0	230	18	0	3	0	1	0	0	0	1
opencou	6	32	17	11	0	11	228	2	0	0	1	2	0	0	0
street	3	0	1	5	7	0	1	169	1	0	0	2	0	1	2
tallbuild	1	0	2	1	8	5	0	1	217	0	0	15	1	0	5
office	1	0	0	0	1	0	0	1	0	106	1	0	4	1	0
bedroom	1	0	0	0	2	1	0	1	1	8	74	2	10	15	1
industria	7	3	1	1	20	0	3	6	19	4	4	106	6	2	29
kitchen	1	0	0	0	1	0	0	1	1	8	3	2	64	21	8
livingroo	2	0	0	1	1	1	0	6	1	10	33	6	10	101	17
store	3	0	4	1	13	9	1	1	6	2	3	6	7	8	151

Algorithm = LLC; Dictionary Size = 1024; Number of Pyramid Levels = 3

	suburb	coast	forest	highway	insidecit	mountai	opencou	street	tallbuild	office	bedroom	industria	kitchen	livingroo	store
suburb	141	0	0	0	0	0	0	0	0	0	0	0	0	0	0
coast	1	216	3	7	0	1	29	0	0	0	1	2	0	0	0
forest	1	0	216	0	0	7	2	2	0	0	0	0	0	0	0
highway	1	4	0	137	3	2	5	1	0	0	0	2	0	0	5
insidecit	9	2	0	2	155	0	0	3	8	2	0	8	7	0	12
mountai	0	7	6	1	0	238	16	1	2	0	1	1	0	0	1
opencou	3	35	9	10	0	11	234	2	0	0	1	3	1	0	1
street	1	0	1	4	5	1	1	171	0	0	1	2	0	2	3
tallbuild	0	0	2	0	8	2	0	2	224	0	1	10	3	0	4
office	0	0	0	0	1	0	0	0	0	109	1	0	1	3	0
bedroom	1	0	0	0	2	1	0	0	0	3	81	1	8	19	0
industria	4	7	4	1	18	2	5	10	16	4	1	106	7	6	20
kitchen	0	0	0	0	2	0	0	1	1	5	7	0	76	13	5
livingroo	1	0	0	0	0	1	0	2	1	6	27	5	12	118	16
store	0	0	3	1	9	7	1	1	6	1	3	4	3	9	167