16720 — PS1

Connor W. Colombo (cwcolomb)

September 27th 2020

1 Representing the World with Visual Words

1.1 Extracting Filter Responses

Q1.1.1

- The Gaussian filters pick up / accentuate bright regions, that is the bodies objects, by blurring out the high frequency content of an image.
- The Laplacians of the Gaussians pick up edges of increasing size with increasing scale *sigma* as zero crossings.
- The x-derivatives of Gaussians pick up vertically oriented edges of increasing size with increasing scale *sigma* as peaks in brightness.
- The y-derivatives of Gaussians pick up horizontally oriented edges of increasing size with increasing scale *sigma* as peaks in brightness.

We need multiple scales of filter responses because there's always a trade-off between the smoothing, which gives increased noise resilience with increased filter scale, and localization, which gives better/more fine edge detection with decreased filter scale.

Q1.1.2

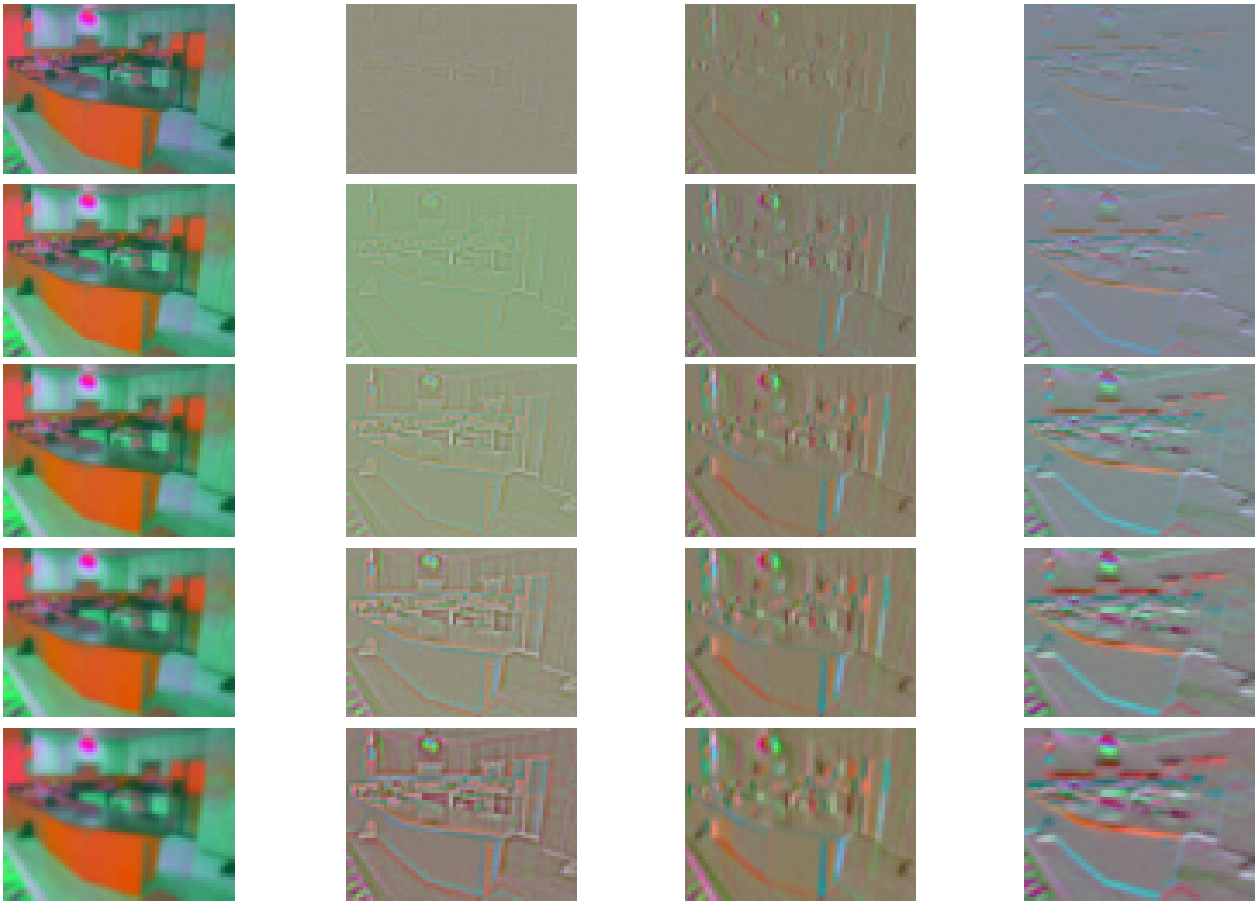


Figure 1: Q1.1.2: Responses of Kitchen Image to Filter Bank as a Collage of all 4 filters (along x axis) at 5 scales (along y axis).

1.2 Computing Visual Words

Q1.3

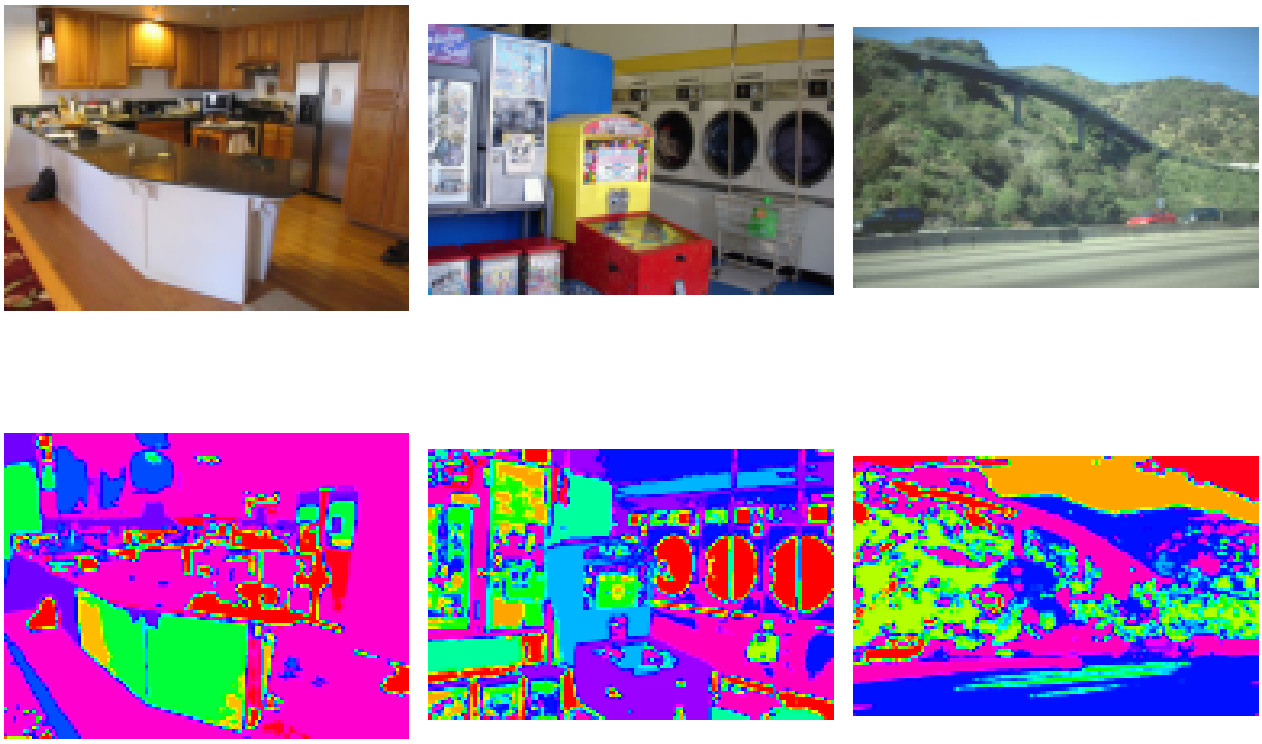


Figure 2: Q1.3: Wordmaps (bottom) corresponding to three selected images (top), displayed using the `hsv` colormap instead of the default `viridis` colormap to increase contrast. Images chosen were, from left to right, `kitchen/sun_aasmevtpkslccptd.jpg`, `laundromat/sun_aakuktqwgbgavllp.jpg`, `highway/sun_abzhjprcojyuuqci.jpg`

The word boundaries in these images appear to be somewhat semantic/textural in that areas of a similar texture content have similar color, irrespective of their location in the image: in the first image's wordmap, brown wood (or the reflections of wood) appear pink, appliance metal appears red, white painted surface under shadow appear purple and, notably, the white cabinet face appears green but the corner of white flooring in the bottom right appears red (so, it's clearly not just remapping colored areas). Likewise, in the second image's wordmap, the shadowed interiors of the washing machines appear red while their white upper exterior appears dark blue and the lower white exterior appears pink; and, in the third image's colormap, the vertical bits of shadowed concrete in the road barrier and the shadowed concrete overpass/bridge appear pink while the concrete road appears blue. With each of these colors indicating a unique textural class and being ascribed to discrete zones in the images, it's clear the scenes' compositions are being broken down by the wordmaps.
However, since there are only $K = 10$ unique textures under the default map for all the various textures across all the various scenes, it is reasonable that a number of similar textures were given the same cluster ids. If K were higher and the number of filter scales used were increased, it's less likely that collisions —such as the reflection in the fridge and the front of the counter appearing to have the same cluster label— would occur.

2 Building a Recognition System

2.1 Quantitative Evaluation

Q2.5

```
confusion_matrix=
[[29  2  5  1  2  2  5  4]
 [ 1 24  6 10  5  0  0  4]
 [ 4  6 17  1  3  4  5 10]
 [ 2  7  0 29  8  0  4  0]
 [ 0  4  5 10 22  4  4  1]
 [ 3  1  3  1  4 28  7  3]
 [10  1  3  3  4  7 17  5]
 [ 3  3  7  1  5  1 10 20]]

accuracy=
0.465
```

2.2 Find the Failures

Q2.6

Since the largest off-diagonal (misclassified) cells in the confusion matrix shown in **Q2.5** were a tie between $(i, j) = (7, 6)$, $(6, 0)$, $(4, 3)$, $(2, 7)$, and $(1, 3)$, this indicates the most common failure modes for the algorithm were:

- Misclassifying a windmill (7) as a waterfall (6)
- Misclassifying a waterfall (6) as an aquarium (0)
- Misclassifying a laundromat (4) as kitchen (3)
- Misclassifying a highway (2) as a windmill (7)
- Misclassifying a desert (1) as a kitchen (3)

The likely culprit for these confusions is that each mismatched pair contains similar textures which may cover a wide swathe of the image and contain little high amplitude and high frequency information (like grass patches), thus dominating the histogram, including the responses at a wide range of scales across all scales. Such sources of large like-texture confusion that exist in these pairs more than in any of the other pairs include:

- Many of the windmill and waterfall scenes contain large sections of green grass and blue skies, largely interrupted by a large vertical white-ish object (the windmill or waterfall itself).
- Waterfalls and aquariums both contain water and, while the water in the waterfalls tend to be more white than blue, many of the aquarium photos contain large central white glares or beams of like, structurally resembling the central white flow of water in the waterfalls.
- Many of the kitchens and many of the laundromats contain shiny silver appliances.
- Many of the highway scenes contain large patches of grass which, as with the waterfalls, serve to dominate the histogram.
- Many of the kitchens contain a light brown wood which (when blurred) seems texturally similar to desert sand.

Q3.1 Hyperparameter Tuning

Parameter tuning occurred in two phases: visual word tuning (σ, K, α) and recognition system tuning (L). The primary objective of this tuning was to increase performance, even at the expense of computation time. As a result, the immediate change was to augment `filter_scales` to include larger filters. $[1, 2, 4, 16, 128]$ were chosen, initially squaring the previous to capture features substantially different frequency content until $16^2 = 256$ was reached — this was deemed too high since some images are `256x256` pixels in size; so, this scale was reduced by half to 128. This new filter bank failed to improve the overall accuracy, leaving it at 0.465 as part of model **[a]**. Despite this, the change kept since: (1) it provides no worse accuracy and is no more arbitrary than the default `filter_scales` of $[1, 2]$ and (2) it is hypothesized that a wider range of `filter_scales` will provide tangible benefits later on when L is increased. However, to save on computation time, the largest kernel (128) was removed for subsequent trials.

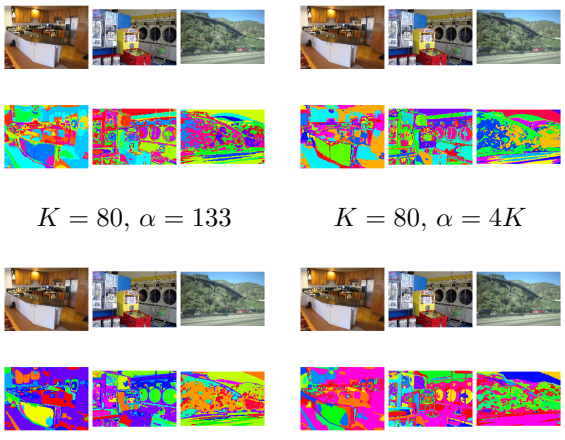
For the next step in visual word tuning, there were two hypotheses:

- K should be increased so that there can be a substantial number of clusters per scene to allow for unique clusters for unique objects in each of the scenes. This is due to the observation that, under default parameters, $K = 10$ seemed to be too few and several objects within an image of a particular scene (such as the carpet and the cabinets in the test kitchen scene) were given the same cluster even though they are texturally distinct. Likewise, this should help reduce the confusion seen in Q2.5, since it appears much of that could be attributed to large, unbroken areas of texture. A rough manual count of the dataset showed there were about 10 unique types of objects (counter, water, appliance, wood trim, concrete, etc) in each of the 8 scenes, so K was set to $10 * 8 = 80$.

- α should be increased to avoid undersampling / not at all sampling certain small scale textures (though this can also be remedied by increasing L). Two theories for a satisfactory value of α are: (1) α should be set such that if a reasonably sized image were divided into blocks 3σ in width (roughly the size of a Gaussian kernel) for one of the larger σ values, there will be at least two sample points in each direction per block — for a σ of 16 and a reasonable but small image size of 256, this gives $\alpha = \texttt{int}((2 * \frac{256}{16*3})^2) = 113$; (2) α should be set such that there can be at least a 2x2 grid of samples per each texture which the clusters correspond to — that is, based on the criterion for K above, $\alpha = 4K$.

To save on computation time, each of these was run first on just the dictionary to determine how accurately their `wordmap` boundaries met semantic expectations and only the one which seemed to conform best was chosen to undergo the full train-test procedure. After K was set to 80, it looked in the `wordmap` like it might be overfitting (splitting up clusters that shouldn't be unique), so a reduced $K = 40$, corresponding to 5 unique clusters per scene, was also tried. From the `wordmap` boundaries of these images, it seemed (d) $K = 40$, $\alpha = 133$ offered the most semantic boundaries so it was chosen for the next trial as model **[b]**, increasing the accuracy to 0.5575. Since this still didn't achieve the desired 0.65 accuracy level, its opposite $(K = 80, \alpha = 4K)$ was also evaluated as model **[c]**, increasing the accuracy slightly to 0.5625.

Lastly, a new (1-indexed) value of L was chosen such that the smallest pyramid chunk size would match the 3σ size of the largest filter (16) for a reasonable but small image (of size 256), so that smaller details of the scene could be captured without being drowned out in the histogram by large unchanging features like grass. That is: $256 * 2^{-(L-1)} = 3 * 16 \rightarrow L = \texttt{ceil}(1 - \texttt{log}_2(\frac{3*16}{256})) \rightarrow L = 4$. This new L value was explored in model **[d]**, which brought the performance up to 0.6525, above the desired 0.65.



$K = 80, \alpha = 133$ $K = 80, \alpha = 4K$

$K = 40, \alpha = 4K$ $K = 40, \alpha = 133$

(a) Parameter Selection for models b and d

	`filter_scales`(σ)	K	α	L	Accuracy
Baseline	$[1, 2]$	10	25	1	0.465
[a]	$[1, 2, 4, 16, 128]$	10	25	1	0.465
[b]	$[1, 2, 4, 16]$	40	133	1	0.5575
[c]	$[1, 2, 4, 16]$	80	320	1	0.5625
[d]	$\mathbf{[1, 2, 4, 16]}$	**80**	**320**	**4**	**0.6525**

(b) Results of Progressive Hyperparameter Tuning

3.2 Further Improvement

Extra Credit

Two hypothesized improvements to the recognition system are proposed:

- Using a cosine vector similarity test instead of a histogram intersection similarity to assess histogram similarity so that two histogram vectors with the same direction (content) but different magnitudes (amount of objects) will still score highly as similar. This test is often used for sentiment analysis / sentence classification in text; so, it seems it should fit well for this texton-based approach. Note: here, cosine similarity is defined as: $s = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$.
- The Recursive Spatial Pyramid will only add a new layer behind a given cell if there's still changes occurring in the cell. This would prevent unnecessary splitting of cell which are predominantly responding to 1 dictionary cluster, while allowing a higher L_{max} to be chosen allowing deeper dives into areas with more activity / content. One computationally inexpensive way of performing this check would be to compare ratio of the (normalized) histogram maximum to the (normalized) histogram centroid/mean for a given cell against some threshold.

For reasons of time, only the proposed first improvement was implemented, giving a new accuracy of 0.5357 — making the best performing model the second worst. It's possible that this would suggest that using cosine similarity is a poor fit for this task; however, when attempting to rerun the previous model without cosine similarity enabled, the same accuracy of 0.5357 is achieved. Rolling back this change restores the old performance, suggesting that this dip in performance is due to faulty implementation rather than a necessarily poor use of cosine similarity; though, such behavior is still strange since the cosine distance was simply computed using `scipy.spatial.distance.cosine(word_hist, histograms[t,:])`.