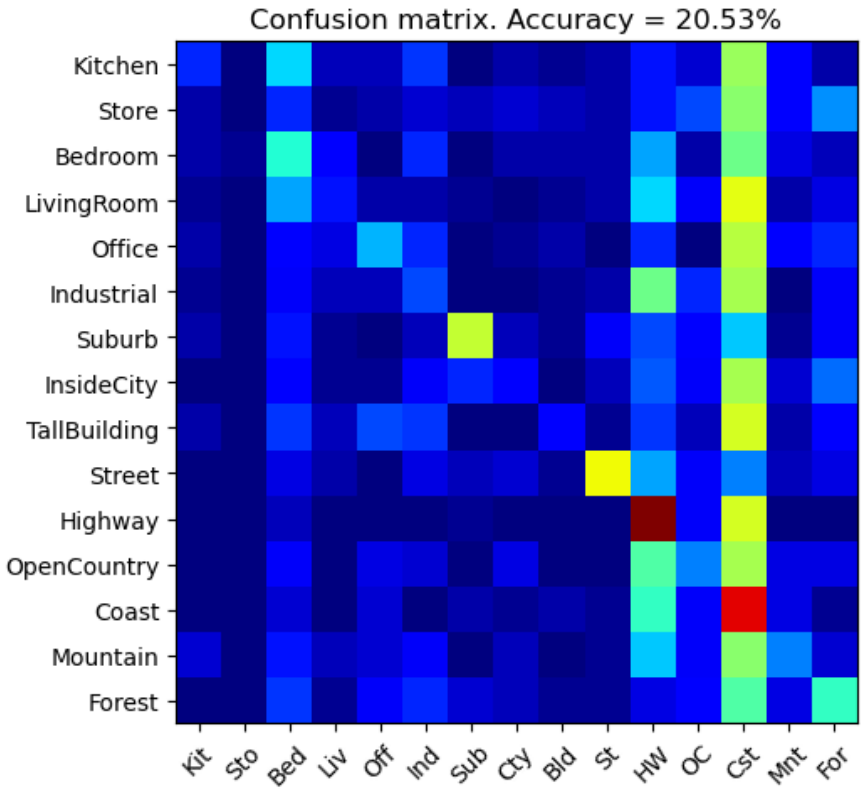
CS 4476
PS 4

Peter Gray
wileygray@gatech.edu
903393428

Part 1: Tiny Image Representation and Nearest-Neighbor Classification

Part 1.3.a: Your confusion matrix, together with the accuracy for Part 1 with the standard parameter set (image_size = 16, k = 3)



Part 1.3.b: Experiments: change image size and k individually using the following values, and report the accuracy (when tuning one parameter, keep the other as the standard (16 x 16, 3)):

ie. when you're tuning image size, keep k at 3, when changing k, keep image size as 16x16

image size:

8 x 8: 18.53%

16 x 16: 20.53%

32 x 32: 19.93%

k:

1: 20.93%

3: 20.53%

5: 20.53%

10: 20.93%

15: 21.33%

Part 1.3.c: When tuning the parameters (image size and k), what did you observe about the *processing time and accuracy*? What do you think led to this observation?

-Processing Time: Increasing the image-size and the k led to longer processing times. This is because larger images and larger k values resulting in more local features, which require more computation.

-Accuracy: Increasing the image size and k improved the accuracy up to a certain point, after which the model became negligibly more effective, or decreased in effectiveness. In general, increasing the image size or k can result in a more precise fit, but it risks overfitting and therefore becomes less generalizable. I’m assuming this is what happened in this example

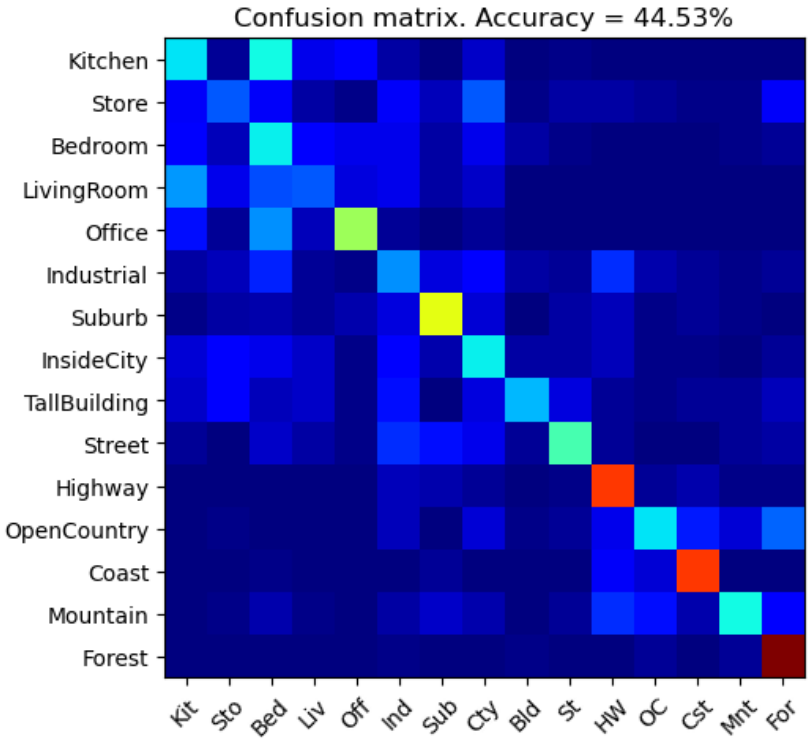
Part 2: Bag-of-words with SIFT Features

Part 2.3: Reflection on Tiny Image Representation vs. Bag of Words with SIFT features:

Why do you think that the tiny image representation gives a much worse accuracy than bag of words? Additionally why do you think Bag of Words is better in this case?

Bag of words works better in this case because tiny image only looks at the raw pixel values, which are highly variant even between members of the same set. Bag of Words uses feature matching which works better in building a model for more abstract classification tasks like this one.

Part 2.4.a: Your confusion matrix, together with the accuracy for Part 2 with the standard parameter set (vocab_size = 50, k = 3, max_iter = 10, stride(build_vocab) = 20, stride(get_bags_of_sift) = 5



Part 2.4.a: Experiments: change vocab_size and k individually using the following values, and report the accuracy (when tuning one parameter, keep the other as the standard (50, 3)):

ie. when you're tuning vocab_size, keep k at 3, when changing k, keep vocab_size as 50. (Other params max_iter = 10, stride(build_vocab) = 20, stride(get_bags_of_sift) = 5)

vocab size:

50: 44.53%

100: 48.67%

200: 49.27%

k:

1: 46.27%

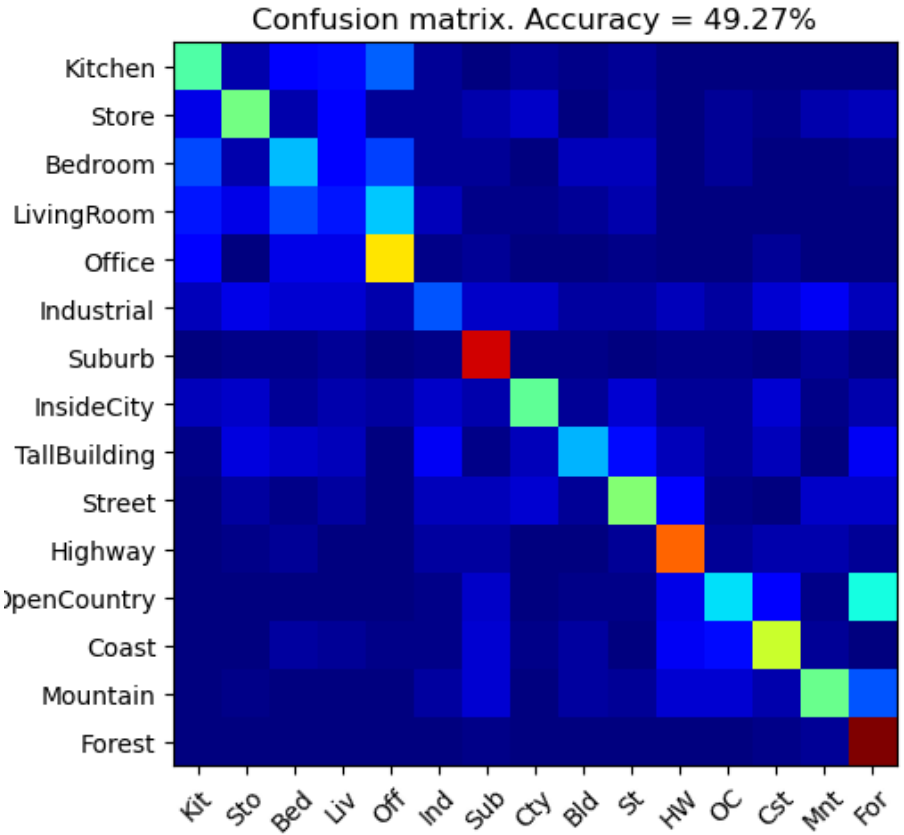
3: 44.53%

5: 47.40%

10: 47.40%

15: 47.27%

Part 2.4.a: Paste the confusion matrix for your best result with the previous experimentation in this slide.



vocab_size: 200

k: 10

max_iter: 10

stride(build_vocab): 20

stride(get_bags_of_sift): 5

Part 2.4.b: Reflection: when experimenting with the value k in kNN, what did you observe? Compare the performance difference with the k value experiment in Part 1.3, what can you tell from this?

I observed that k is performant relative to the size of the data set it is trying to fit. A larger data set will generally require a larger k to get a better fit.

Part 3: Extra Credit

EXTRA CREDIT

Part 3.1: Post best confusion matrix, together with the accuracy out of all the parameters you tested. Report the parameter settings used to obtain this result.

<Plot here>

Parameter settings:

max_iter:

stride(build_vocab):

stride(get_bags_of_sift):

vocab_size:

k (kNN):

EXTRA CREDIT

Part 3.2: Post confusion matrix along with the distance metric that you used for achieving a better accuracy on standard parameters. Why do you think it performs better?

<Plot here>

Distance metric and why it works better:

EXTRA CREDIT

Part 3.3: Post confusion matrix along with your explanation of your SVM model and detail any other changes your made to reach an accuracy of 65% or greater.

<Plot here>

Description of your model: