Image Processing Methods for Environment Classification

Rahul Sharma
Department of Mathematics and Statistics
York University
Toronto Ontario Canada
rahul494@my.yorku.ca

ABSTRACT

Digital image processing is a field that has although been around for several years, has slowly begun to gain more attraction. This is largely due to the increase of computing power and computer memory, as well as the innovation we have seen in the machine learning community, as new techniques are still being proposed till this day.

This report presents various classification solutions in order solve an image recognition problem, to determine whether or not an image has been captured outdoors.

Outlined are steps taken to read images into memory, summarize and partition pixel intensities, and utilize machine learning algorithms such as support-vector machines, penalized logistic regression, and convolutional neural networks to determine which method provides the best classification accuracy.

KEYWORDS

Image Processing, data mining, data analytics, machine learning, digital image processing, lasso, logistic regression, support-vector machine (SVM), convolutional neural network (CNN)

1 Introduction

The area of image processing has grown to a wide variety of applications. These applications include areas such as medical image processing, autonomous vehicles, banking, and many others. As image recognition may be obvious to a human, we have come to a point in our society where we have the computing power to automate this process and allow computers to now preform the classification for us.

Along with having the machine learning and data mining tools to help us with these image processing problems, in this report we outline models such as support-vector machines, logistic regression, and convolutional neural networks to solve an image processing problem.

This report provides multiple solutions of the problem of distinguishing whether a photograph has been captured outdoors or not.

Ahead in this paper is not only the models we use to conduct this classification, but we elaborate how the images were collected, how the pixels are used as parameters, and how they are used in our various models.

2 Data

The dataset utilized for this task was gathered by the Columbia University, consisting of various images including pictures taken indoors, outdoors, and images created artificially [1].

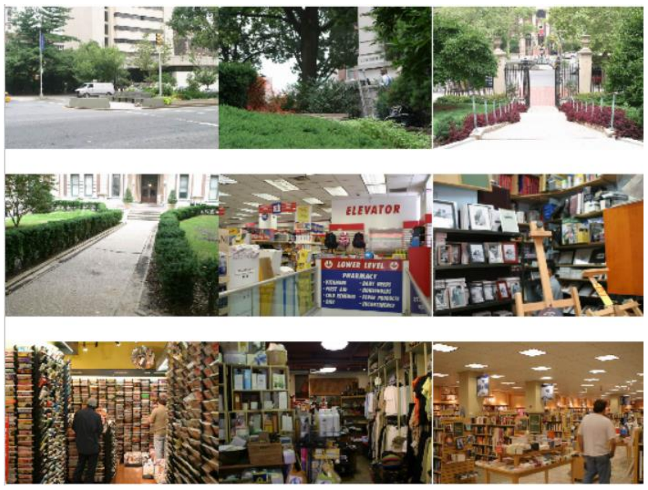

Fig. 1 Various images that are provided in our dataset

This dataset consists of 800 images, all with a variety of dimensions. As well as the images themselves, the dataset comes along with a metadata.csv, where we also have information about parameters such as the photograph location, author, camera type, and image type (e.g. outdoor-day, indoor-dark, artificial, etc.). It is this image type predictor we use as the Y vector for classification $Y = [y_1 \quad y_2 \quad \cdots \quad y_n]^T$. For this case, n = 800 as we have 800 images in our dataset.

3 Methodology

There are several ways pictures can be recognized by computers. In our case, we utilize the 3 channels contained by every image: red, green, and blue values. Using these pixel intensities, we outline various summarization and partitioning techniques that can be used in the classification problem.

3.1 Aggregate Pixel Intensity

The first technique we use to summarize the image is to look at each image channel individually, and store each mean and median pixel intensity of the entire channel. Doing so we create 2 datasets we can use for our observation list.

1. Aggregate Mean – We define this set as X to train and test our models, where X has 3 predictors containing the mean red, green, and blue of each image.
2. Aggregate Median – Defined similarly as above, instead we compute the median value of each channel and store it into X.

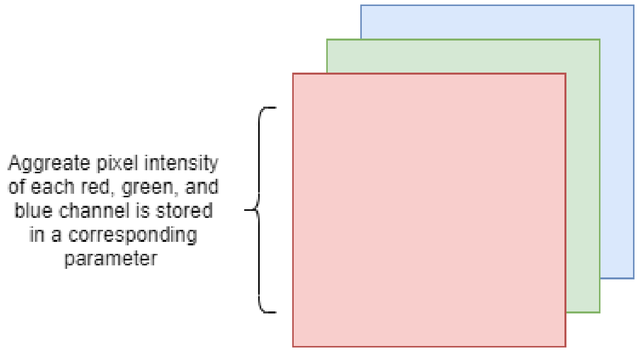
Aggreate pixel intensity of each red, green, and blue channel is stored in a corresponding parameter

Fig. 2 Illustration of how we read and process an image

3.2 Three by Three Partition

The second technique we use to summarize the image is to partition the image into a 3 by 3 grid, where we calculate the individual mean and median values for our image. Doing so, we come up we define an additional 2 possible set of parameters we can use for X.

3. 3 by 3 Mean – We define this set as X to train and test our models, where X has 27 predictors corresponding to each grid section.

$$grid\ size\ \times \#\ of\ channels\ = parameter\ size$$

4. 3 by 3 Median – Defined similarly as above, instead we compute the median value of each grid section and store it into X.

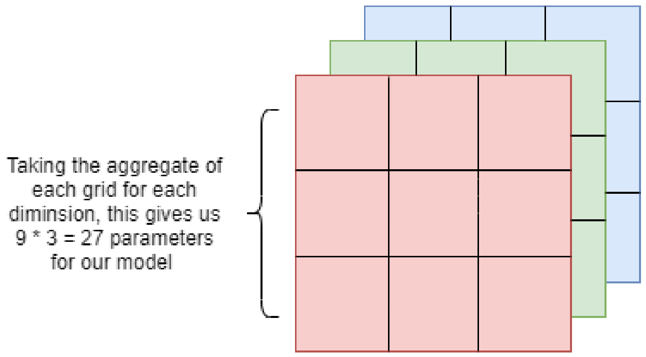
Taking the aggregate of each grid for each diminsion, this gives us 9 * 3 = 27 parameters for our model

Fig. 3 Illustration of how we partition images into a grid

3.3 Resized Pixel Intensities

The final technique we use is to store all our pixel information as parameters for our model. Doing this, we must ensure that the size of the size of the image must be uniform to get the appropriate pixel information for each image.

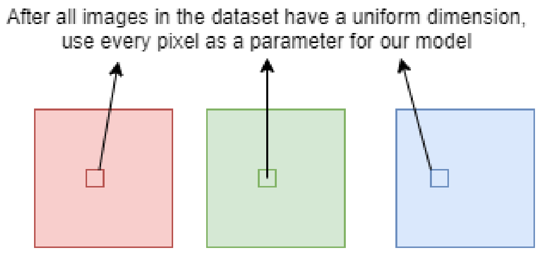
After all images in the dataset have a uniform dimension, use every pixel as a parameter for our model

Fig. 4 Illustration of how we read each corresponding channel pixel

4 Model Analysis and Results

In this section, we outline the various models used to classify our dataset, as well as evaluate our performance based on the accuracy metric of each model.

4.1 Support-Vector Machine

Support-vector machines have been widely known used for classification problems, as their effectiveness truly shows in their ability to handle high dimensional data. This model is especially useful for our problem, as SVM are traditionally used for 2-class classification problems [2].

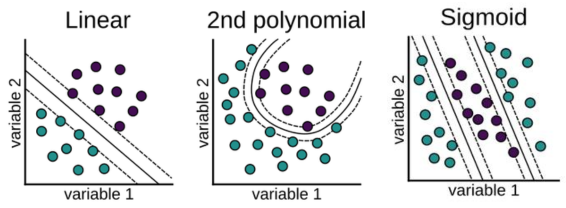

Fig. 4 Kernels used in our analysis

For analyzing the dataset using SVM, we utilize various kernels to see which method provides the greatest accuracy. After training our model and preforming a series of cross validations, our average accuracy can be displayed in the figure below.

	Linear	Polynomial	Sigmoid
Aggregate Mean	65.6%	65.6%	65.6%
Aggregate Median	65.5%	65.5%	65.3%
3 by 3 Mean	65.5%	65.3%	65.4%
3 by 3 Median	65.3%	65.2%	65.4%

Fig. 5 Average performances for SVM kernels

Although the performance of the models is relatively as we utilize various types of kernels, what we do see is the set of predictors that do give us the slight best accuracy is from Aggregate Mean, where we simply take the average pixel intensity of each channel to represent a picture.

4.2 Penalized Logistic Regression

In addition to utilizing a classification algorithm such as logistic regression, we will perform variable selection using lasso to enhance our prediction accuracy [3]. The steps taken are as follows:

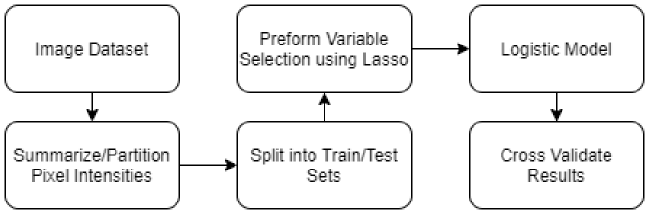
Image Dataset
Summarize/Partition Pixel Intensities
Split into Train/Test Sets
Preform Variable Selection using Lasso
Logistic Model
Cross Validate Results

Fig. 6 Approach used for penalized logistic regression

Upon creating and training our model, we achieve the following results:

	Accuracy
Aggregate Mean	79.2%
Aggregate Median	74.1%
3 by 3 Mean	71.3%
3 by 3 Median	70.6%

Fig. 7 Average performances for pixel summaries under logistic regression

Under this method, we see that penalized logistic regression gives us a much better performance than the SVM. This is especially the case for the Aggregate Mean, where we see an 13.6% increase in accuracy.

4.3 Convolutional Neural Network

Below are layers we outline that we use in our convolutional neural network. For this method, we use the entire image and store each pixel in its corresponding parameter. Doing this we resize each image to a 128 by 128 dimension, and thus we obtain a parameter size as follow:

1. Convolution Layer: $(3 \times 3 \times 3 \times 16) + 16 = 448$

   Our layer uses 16 filters, as a result of the first layer we have 448 parameters.

2. Convolution Layer: $(3 \times 3 \times 16 \times 16) + 16 = 2320$

   Our second layer uses 16 filters as well, as a result we have 2320 parameters.

3. Pooling Layer: Parameter size remains 2320

   Here we use a pool size of 2 by 2.

4. Convolution Layer: $(3 \times 3 \times 16 \times 32) + 32 = 4640$

   In this third layer, we introduce a filter size of 32, thus at this point we have 4640 parameters.

5. Convolution Layer: $(3 \times 3 \times 32 \times 32) + 32 = 9248$

Our fourth layer uses 32 filters as well the previous, as a result we have 9248 parameters.

6. Pooling Layer: Parameter size remains 9248

   Here we once again use a pool size of 2 by 2.

7. After preforming a dropout and dense layer, we are ultimately left with 285,808 parameters in our CNN.

In order to evaluate the performance of the model, we repeatedly select a random sample of images to train/test our network. Doing so on average we see an accuracy of approximately 90% for our model.

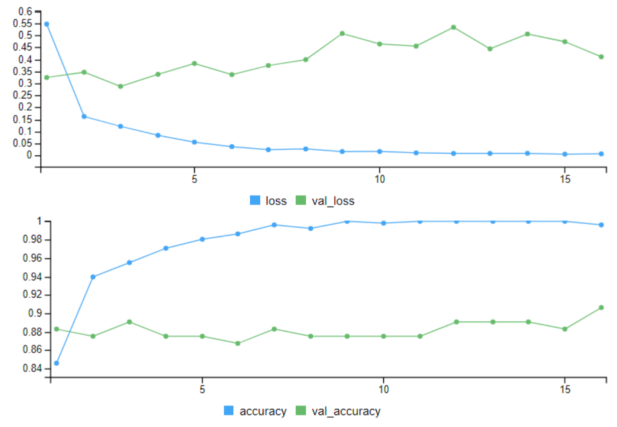

Fig. 8 Example performance run of CNN

5 Conclusion

Given the various summarization techniques and machine learning models presented, we can make the distinction that using the entire pixel information of a picture for a convolutional neural network provides us with the greatest level of accuracy.

Although 90% accuracy is in many cases very good performance, in the future we can always improve the model by increasing the dataset size and add more observations to train/test, as we only had 800 to begin with.

6 Source Code

For further inspection of the code used to partition pixels and utilize models, please visit:
https://github.com/rahul494/Environment-Image-Recognition

REFERENCES

[1] Ng, T.-T. (2005). Columbia Photographic Images and Photorealistic Computer Graphics Dataset. Columbia Photographic Images and Photorealistic Computer Graphics Dataset, 1–13. Retrieved from http://www.ee.columbia.edu/ln/dvmm/publications/05/ng_cgdataset_05.pdf

[2] Jothilakshmi, S. (2016). Cognitive Computing: Theory and Applications. Retrieved April 8, 2020, from https://www.sciencedirect.com/topics/computer-science/support-vector-machine

[3] Sperandei, S. (2014). Understanding logistic regression analysis. *Biochemia Medica*, 12–18. doi: 10.11613/bm.2014.003