Two Dimensional Intelligent Character Recognition

Ryan Pattison, Douglas Anderson, and Oliver Cook[a,b,c]

[a] *ryan.m.pattison@gmail.com*
[b] *dander01@uoguelph.ca*
[c] *cooko@uoguelph.ca*

Abstract

For quite some time computer vision has been used to extract characters from images in order to gain knowledge from text. In recent years many of the techniques for recognizing characters has been based on machine learning.

Keywords: Machine Learning, Data Mining, Computer Vision, Intelligent Character Recognition

1. Introduction

The problem of Intelligent Character Recognition (ICR) in Artificial Intelligence and Machine Learning has received much attention in past years. This field of study has focused on the classification of *handwritten* or *typewritten* letters and numbers. The novelty of this project will be in the development of an algorithm which recognizes characters, or more generally symbols, arranged to form words in *two* dimensions. The practical application proposed here will be the development of an algorithm for recognizing hand drawn cartography. To this end, we have defined a small set of symbols to represent various terrains: grass, mountains, bodies of water, and more. Each symbol will be drawn into a single cell on a standard sheet of grid paper and scanned into a computer. Using the scanned images, the proposed algorithm will learn to recognize the symbols cell by cell as well as in the context of the information in the surrounding cells. In order to demonstrate the utility of such an algorithm, we propose an application which will read a hand drawn map and produce a computer generated image showing the same map. We will use the software tool WEKA[1] to create classifiers.

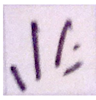
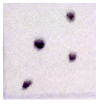
2. Process

2.1. Overview

We have created 6 symbols to be used to recreate maps for a popular video game Pokémon to ensure that our data set is sampled from a well defined distribution. An example map from the game is shown in Figure 1. There is a sample of the the symbols names and the hand drawn equivalents in Table 2.1.

Figure 1: Pallet Town map from Pokémon[2]

Table 1: Symbols Names and the Hand drawn equivalents

Water		Grass	
Rock		Tree	
Dirt		Sand	

2.2. Preprocessing

For all of our analysis we preprocess colour images into black and white images. To do this we first remove the blue grid lines from the image and replace them with white. We then remove all colour information from the image and resize it such that every cell of the grid is 75 pixels by 75 pixels.

During processing this image is divided into single cells, each with a single symbol in it to be classified.

2.3. Feature-Based Approach

The Feature-Based approach used here applies convolution filters to the cell images and then gathers statistics based on the distribution of black pixels in each of the x and y dimensions. A convolution filter is matrix of coefficients of odd size. Below is a convolution filter for a mean blur.

$$F_{blur} = \begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$

If we consider a single cell of the grid, a portion of the image with one symbol in it, as a matrix of luminance values like so:

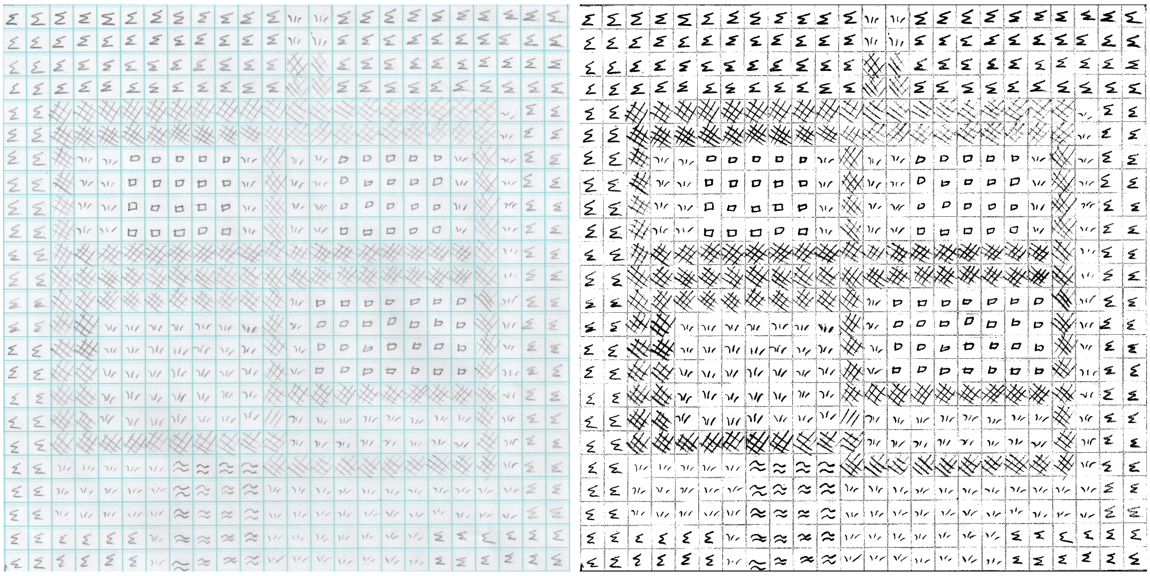


Figure 2: Image before preprocessing and image after preprocessing. Notice that the contrast has increased and the blue lines have been partially removed

$$\begin{array}{ccccc} \ddots & \vdots & \vdots & \vdots & \iddots \\ \dots & 64 & 64 & 32 & \dots \\ \dots & 32 & \boxed{32} & 128 & \dots \\ \dots & 32 & 16 & 32 & \dots \\ \iddots & \vdots & \vdots & \vdots & \ddots \end{array} \qquad \begin{array}{c} 64F_{11} + 64F_{12} + 32F_{13} \\ +32F_{21} + 32F_{22} + 128F_{23} \\ +32F_{31} + 16F_{32} + 32F_{32} \end{array} \qquad \begin{array}{ccc} \ddots & \vdots & \iddots \\ \dots & \boxed{48} & \dots \\ \iddots & \vdots & \ddots \end{array}$$

The filtering process repeats for each pixel until all pixels have been replaced. The resulting image has the luminance matrix A.

$$A = \begin{bmatrix} a_{00} & a_{10} & \cdots & a_{n0} \\ a_{01} & a_{11} & \cdots & a_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{0m} & a_{1m} & \cdots & a_{nm} \end{bmatrix}$$

We define x and y to be the row and column sum functions.

$$x(i) = \sum_j A_{i,j} \quad y(j) = \sum_i A_{i,j} \tag{1}$$

Next we record $\mu_x, \mu_y, \sigma_x, \sigma_y$ as the feature set for this filter.

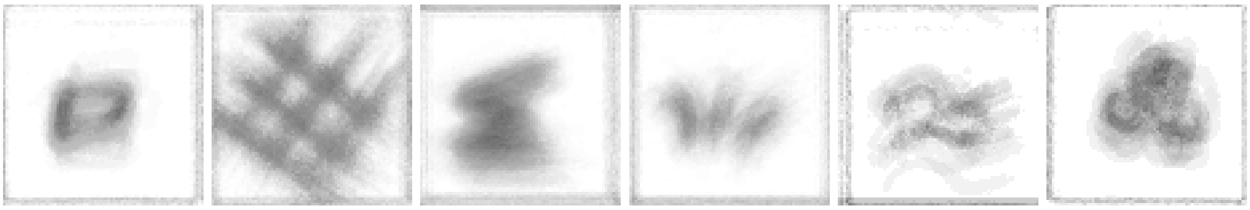


Figure 3: The gold images created as the mean image of all examples of each symbol. In order we have symbols representing buildings, dirt, forest, grass, water, and rocks.

2.4. Gold Comparison Approach

Similar to the feature-based approach, we define x and y to be the row and column sums and calculate the mean and standard deviation. To get the symbols to overlap we apply a linear transformation to each pixel using the statistics gathered in each dimension.

$$x' = (x - \mu_x^B)\,\frac{\sigma_x^G}{\sigma_x^C} + \mu_x^G \quad y' = (y - \mu_y^B)\,\frac{\sigma_y^G}{\sigma_y^C} + \mu_y^G \tag{2}$$

These new transformed pixels are placed into a new image A which now overlaps with the gold image G. We then compare the overlapping image A to the gold standard image G by taking the sum of the squared difference at each pixel position. This gives us an error measure which we record for each of the k classes.

$$\xi_k(A) = \sum_i \sum_j (A_{ij} - G_{kij})^2$$

The result is a vector of size k, the number of classes, with an error measure for the comparison against the corresponding class. We then input this vector into WEKA along with the correct answer for training purposes. We use a J48 decision tree with 10-fold validation to train WEKA to classify using these error measures.

2.5. Proximity Approach

We want to classify the symbol x, and we do this by first calculating the probability that x belongs to each class, $P(X) \quad X_i = P(x \in C_i)$. The proximity classifier is given the class of each symbol in the cardinal directions: north, south, east, and west. Using this information, we restrict the probability calculation to only consider the subspace where a symbol has the given classes neighbouring it. Such a restriction is the conditional probability:

$$P(X) = P(X \,|\, N = c_1,\, E = c_2,\, S = c_3,\, W = c_4)$$

In calculating probabilities, we can only approximate them using relative frequencies collected from our sample data. For a set of k classes, there are k^4 possible combinations of neighbouring cells and too many of these combinations occur rarely in our dataset to make confident approximations of the true probabilities. To resolve this, we assume that the conditional probabilities are independent in each direction. Under this assumption, the calculation becomes much more reasonable for our small data set since there are only $4k$ possible combinations. Our resulting calculation becomes:

$$P(X) \approx P(X|\, N\!=\!c_1)P(X|\, S\!=\!c_2)P(X|\, E\!=\!c_3)P(X|\, W\!=\!c_4) \tag{3}$$

From the definition of conditional probability we have that for some class c in the direction D:

$$P(X|\, D\!=\!c) = \frac{P(X \cap D\!=\!c)}{P(D = c)}$$

To calculate each of the conditional probabilities, we compute a table from the samples which records $P(X \cap D\!=\!c)$ and $P(D\!=\!c)$ for each class and direction. We use the table to calculate the relative conditional frequencies and, under the assumption of independence, we approximate $P(X)$ using equation 3.

Having the probability vector $P(X)$ under the assumption of independence, we create an input file for WEKA. This file includes the probability vector, the neighbouring classes, and the class of the symbol being classified for training. WEKA creates a decision tree which is used to reduce the error in the probability calculation introduced under the assumption of independence.

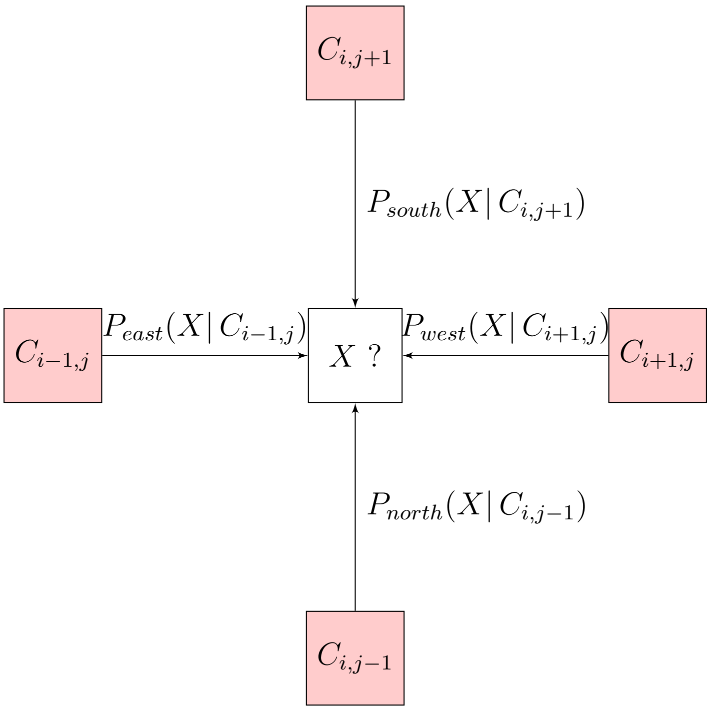


Figure 4: This figure describes the probability of class X neighbouring cells $C_{i,j}$

Let V_d be a vector of probabilities, where n is the number of classes, d is a direction.

$$V_d = (P_d(X_1|\, C_d), P_d(X_2|\, C_d),\ \dots\ , P_d(X_n|\, C_d)) \tag{4}$$

To determine the probability of a cell belonging in a particular class, we assume independence so that we can multiply the vectors of all directions together using the element-wise product.

$$C_{centre} = V_{north} \circ V_{east} \circ V_{south} \circ V_{west} \tag{5}$$

This new vector C_{centre} contains n probabilities that this cell should be classified as a particular class. One approach to classifying the unknown symbol would be to choose the class corresponding to the maximum probability in C_{centre}.

An example of this process can be seen with the following situation. Suppose that the training set has the following characteristics:

	c_B	c_D	c_{EDGE}	c_F	c_G	c_W
north	0	0.1558	0.0519	0.0519	0.7403	0.0325
east	0.0779	0.0974	0	0.1493	0.6428	0
south	0	0.2403	0.0130	0.0065	0.7403	0
west	0.0519	0.2273	0	0.0519	0.6428	0.0260

Table 2: The probability of the class of neighbours in each direction for a grass cell from the map that we created entitled PalletTown

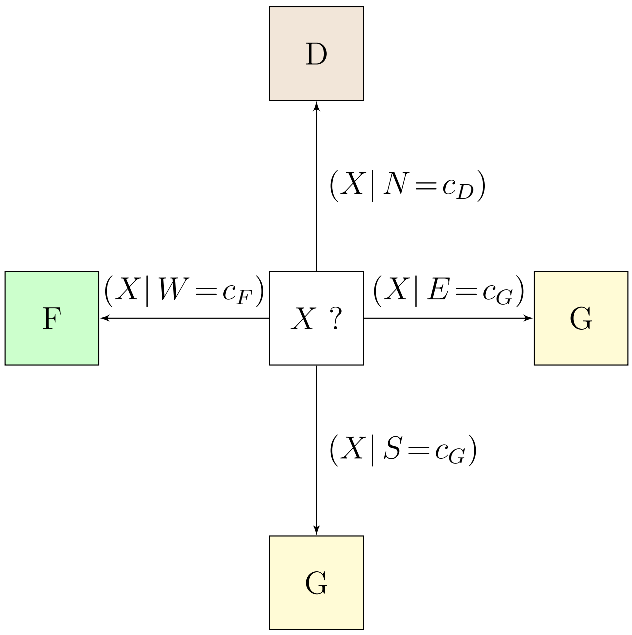


Figure 5: This situation includes a grass to the south and east, a dirt to the north and a forest to the west. It is similar to cell 2,19 in pallet town (when indexed from zero at the top left)

2.6. Classifier Training

Training was done using WEKA and 10-fold cross validation. the proximity classifier was tested on a map it had never seen in training.

3. Results

Algorithm	Samples	Feature Set	Success	Features	Tree nodes
Feature-Based	9888	Single Filter	76.33	31	1525
Feature-Based	9888	Single Filter+Blur	77.01	31	1465
Feature-Based	9888	No Spatial	80.64	205	1131
Feature-Based	9888	Spatial	82.47	207	1005
Proximity	13200	Surrounding	93.78	10	93
Gold Comparison	225	Max Probabilities	70.40	7	N/A
Gold Comparison	225	Probabilities	83.33	7	?

4. Conclusions

Each of our classification methods each have their own strengths and weaknesses. The **Feature-Based method** is able to recognize arbitrary features but requires intended representations of each of the maps during training. **Proximity method** works extremely well with the set of maps we chose because they adhere to patterns. However to detect these patterns in a data set the proximity classifier needs a relatively accurate representation of the map. This method has the added benift of haveing it's output be the same as it's input. The **Gold Comparison** method is able to classify symbols fairly well without intended representations of the map. However this method requires gold images as input and the number of features in the output generated is dependent on the number of gold images.

We believe that for the best results the methods should be used in conjunction. For example using the gold comparison method we could generate a decent representation of the map which could then be cleaned up by the Proximity method.

5. Further Work

One thing that we would like to work on the in future is to use the methods in conjunction to play off of each others strengths. For example if we were to run the Gold Comparison method to get a rough approximation of the layout of the shapes, we could then use the Proximity method to clean up the results. Combining method would allow a more hands off approach to solving the problem

Feature based classification may see improvement from considering other transformations of the pixels when calculating statistics, such as distance from the centre.

An experiment that we would like to run is to use "ideal" gold images rather then mean gold images. We suspect that performance of the gold method could be improved by using images selected from the images that illustrate the intended shape of the character.

Regions with the feature extraction method

6. References

[1] Geoffrey Holmes Bernhard Pfahringer Peter Reutemann Ian H. Witten Mark Hall, Eibe Frank. The weka data mining software; an update. *SIGKDD Explorations*, 11, 2009.

[2] Nintendo. Pokemon firered.