Novel Classifier for Smart Traffic Network

Submitted June 2020, in partial fulfillment of
the conditions for the award of the degree **Bachelor of Computer Engineering/Computer Science.**

Thomas Courtney

3175353

Supervised by Assoc. Prof. Kaushik Mahata

School of Engineering

University of Newcastle

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____ / _____ / _____

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Economies of the world have been industrializing for over two centuries [1] and the consequent advent of machines and factories has seen a great migration of humanity from rural fields to urban habitats. Such a concetration of people has posed plethora of infrastructural challenges. This has lead to many iconic solutions to high density living, like the skyscraper, subway train and traffic light. One unfortaunte metropolitan trait, however, continues unabated and is in fact growing in size, it is traffic congestion [2].

Road transport contributes 16.5% of global CO2 emissions [3] and costs the US $305 billion of productivity per year alone [4] [5]. Any small improvement in this situation will yield great benefit to society and the future of the planet.

There is no single solution to the problem of traffic congestion, whether it be the building of tunnels, the conversion of traffic lights to roundabouts, the widening of roads or the narrowing of roads. The integration of several lightweight and inexpensive techniques is a far more attractive and effective approach [6]. A 'Smart City' is such an approach, it is implemented by deploying many IoT devices in an urban area to collect data [7]. This technology could be applied to collect traffic data and use it to inform investment in a traffic network.

A versatile and inexpensive method of data collection in a traffic network is computer vision. This technique requires as input only raw images of a traffic network and could therefore piggyback onto existing surveillance cameras. The intention of this technology is to mimic the ability of human vision to rapidly identify distinct objects from the real world. A human's ability to separate a vehicle from its surroundings and track its movement is one example of this.

The implementation of a robust computer vision algorithm for traffic monitoring is not trivial. The dependence on only one feature in an image will not yield accurate results, however the combination of several different features will, features like object edges, object area, hue, contrast and reflectance. There are a great many and varying number of ways in which to extract information from these features. Whatever method of implementation is selected must provide consistent accuracy, and in the case of traffic monitoring, real time results.

Therefore, it is the objective of this report to explain the design and operation of a novel computer vision algorithm that can effectively count and estimate the speed of vehicles in traffic, in real-time, given raw images of a traffic network.

Chapter 2

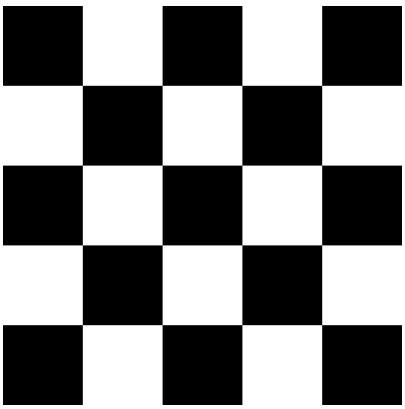
Literature Review

This section of the report expands upon the theory behind the design of the system. Digital image processing and computer vision are broad fields and so only concepts relevant to the system are explored.

2.1 Digital Images

Digital images consists of a discrete number of discrete valued data points called pixels. It is the value of these pixels that determine their color. A pixel's value is also called its *intensity* and will lie within a range of values determined by the image's encoding. For example an 8-bit image has pixel values in the range $[0, 255]$.

Digital images are often represented as a 2D array of values where each cell corresponds to an intesnity. In software in particular, digital images are stored and manipulated as arrays. An image's array is $M \times N$ pixels in size, which is also its resolution. Notice that in Figure 2.1 the lowest value is 0 (black) and the highest value is 255 (white). All other colors that can be represented by 8-bits are between these two values.



(a)

0	255	0	255	0
255	0	255	0	255
0	255	0	255	0
255	0	255	0	255
0	255	0	255	0

N

M

(b)

Figure 2.1: (a) A Simple 8-bit Image (b) Array Representation of Image.

2D array representation allows pixels to be referenced by their relative position in the M and N directions. Furthermore, M and N may be substituted for the x and y axes of the Cartesian plane. In fact, a digital image is just a two dimensional function where each pixel is described by a coordinate (x,y) and an intensity (z) at that point.

$$z = f(x, y) \tag{2.1}$$

This means that two dimnesional operations may be applied to a digital image and images may be modelled as 3D surfaces. Modelling images as a surface is useful when trying to visualize rates of change in an image. In Figure 2.1 you can see that there is a rapid change in value between different coloured squares, shown as steep cliff edges.

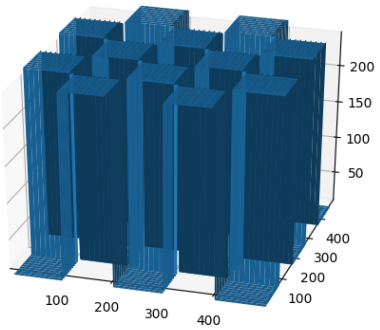


Figure 2.2: Surface Plot of Figure 2.1a

2.2 Linear Filtering

Filtering an image is a way to augment or extract information from it. Linear filters are a subset of operations that operate on a neighbourhood of pixels in an image. All linear filters operate on the same prinicpal, that is, they output a weighted average of their input. Filters take pixels in the neighbourhood covered by the filter as input. It is the weightings of a filter, known as filter coefficients, that determine the effect of the filter. These weights are stored in a matrix called a mask or kernel (see Figure 2.3) that's then convolved or correlated with an image (see sections 2.2.1 & 2.2.2).

$$Kernel = \frac{1}{\sum_{i=0}^{M}\sum_{j=0}^{N} c_{i,j}} \begin{bmatrix} c_{0,0} & c_{0,1} & \dots & c_{0,n} \\ c_{1,0} & c_{1,1} & \dots & c_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,0} & c_{m,1} & \dots & c_{m,n} \end{bmatrix}$$

Figure 2.3: General Form of a Linear Filter

A filter kernel is nearly always square so as to have a center cell which sits atop a reference pixel. The result of the filter's application at that reference pixel will be stored in the output image at the location of the reference pixel. Notice in Figure 2.4 how the mask sits over the reference pixel.

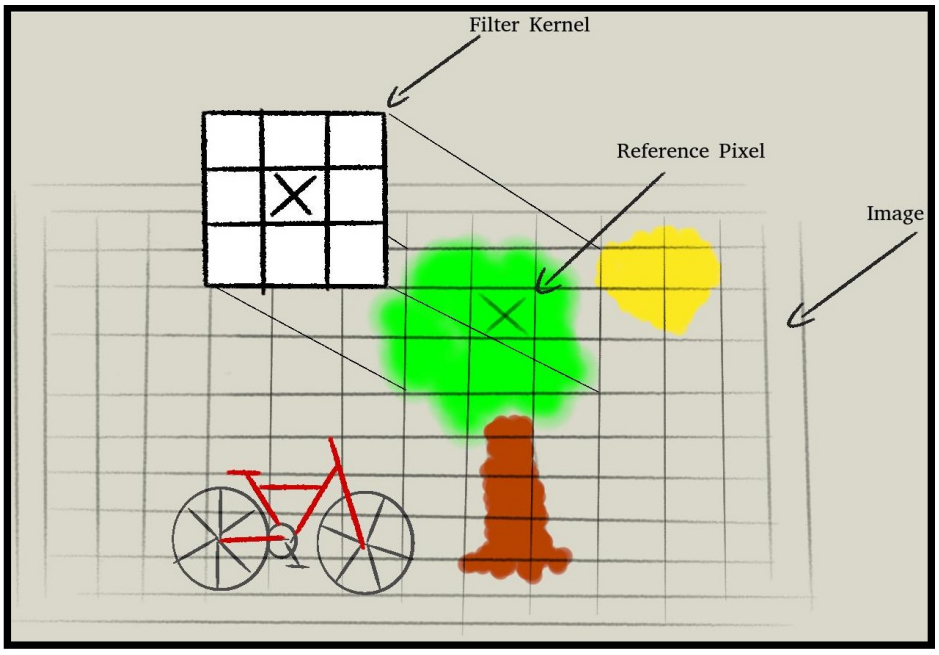
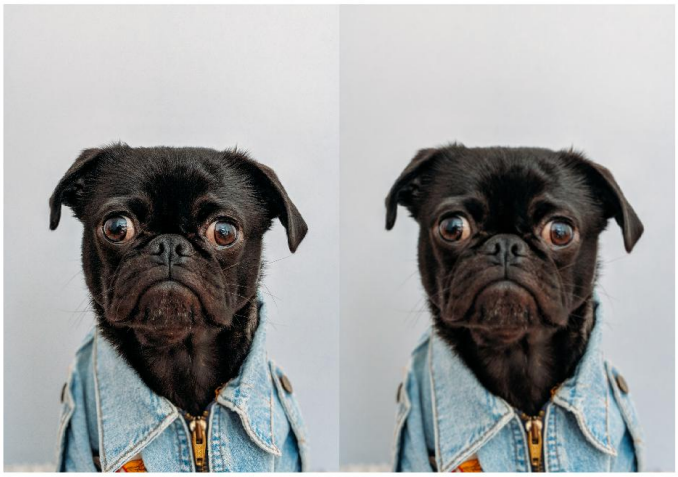


Figure 2.4: Visualization of a Filter Kernel Application

A 'box filter', for example, outputs the average of its inputs because the filter weights are evenly distributed. By passing this filter over an image its sharpness is reduced giving a smoothing or blurring effect. This can be observed in Figure 2.5a.



(a) Left: Photo by Charles Deluvio. Right: Application of 16×16 box filter.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(b) Box Filter Kernel.

Figure 2.5: Box Filter application and kernel.

2.2.1 Correlation

The application of a linear filter $h(u, v)$ to an image $f(i, j)$ may be described as follows

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f(i+u, j+v)h(u,v) \tag{2.2}$$

$g(i, j)$ is the output image. Performing correlation with a filter may be notated more

concisely by the correlation operator.

$$g = f \otimes h$$

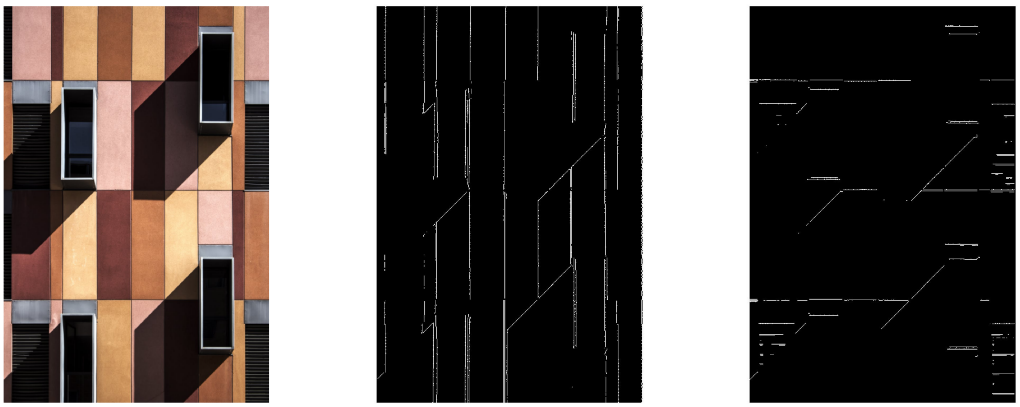
Correlation measures the similarity between two signals. Both digital images and linear filters are two dimensional signals. Performing correlation between them will yield an output image where the highest values correspond to where the image and filter were most similar [8]. This useful because if you wish to emphasise a feature in an image it can be done by correlating it with a filter that describes that feature. For example, if you wished to exagerate vertical and horizontal lines you could use Sobel filters as in 2.6.

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

(a) Horizontal Sobel Filter Mask (b) Vertical Sobel Filter Mask

Figure 2.6: Sobel Filters

The use of negative weightings means that values next to an edge are diminished and the positively weighted line sections (the 2s) strengthen line features. Notice in figure 2.7 how the lines have a high values (white) and regions that aren't lines are low valued (black).



(a) Image by Simone Hutsch (b) Vertical Sobel Filter (c) Horizontal Sobel Filter

Figure 2.7: Application of Sobel filters to exagerate lines.

Correlation is *shift invariant*, which means that it does the same thing no matter where in an image it is applied. To satisfy this property correlation may be superpositioned

$$a(f_1 + f_2) = af_1 + af_2$$

and abides by the shift invariance principle

$$g(i,j) = f(i+k, j+l) \Leftrightarrow \ (h \circ g)(i,j) = (h \circ f)(i+k, j+l)$$

Correlation has the side effect of flipping both horizontally and vertically the location of output points relative to location the center point (*reference point*) in the original image which may be undesirable.

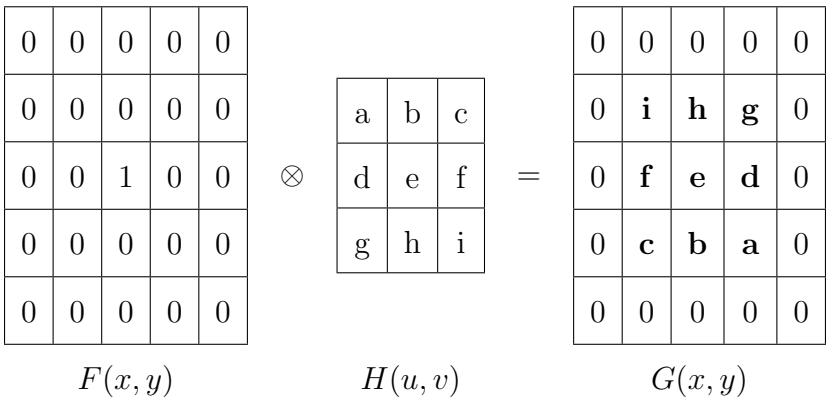


Figure 2.8: Correlation of a filter and an image.

2.2.2 Convolution

Convolution is also a linear operation that is shift invariant. It is very similar to correlation except that where correlation measure similarity between signals convolution measures the effect of one signal on another. It is described mathematically by the expression,

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f(u,v)h(i-u, j-v)$$

Notice that the filter $h(i,j)$ is rotated 180 degrees. This causes the output's orientation to match the original image. Convolution may be notated as follows,

$$g = f * h$$

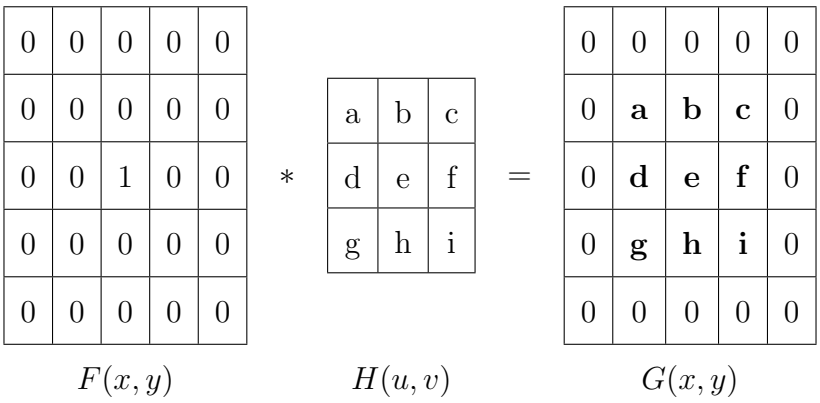


Figure 2.9: Convolution of a filter and an image.

2.3 The Gaussian

The Gaussian provides a consistent model for normal distributions, aka the bell curve, as in Figure 2.10 & 2.11. Normal distributions follow the *central limit theorem* where in if a histogram is taken of a sufficiently large number of independent random variables they will distribute with a central most probable value and symmetrically fall away either side of this value. Many datasets follow this trend closely enough that the Gaussian can be used to approximate a probability distribution for them.

The Gaussian function is defined by three parameters, its mean μ, standard deviation σ and amplitude A.

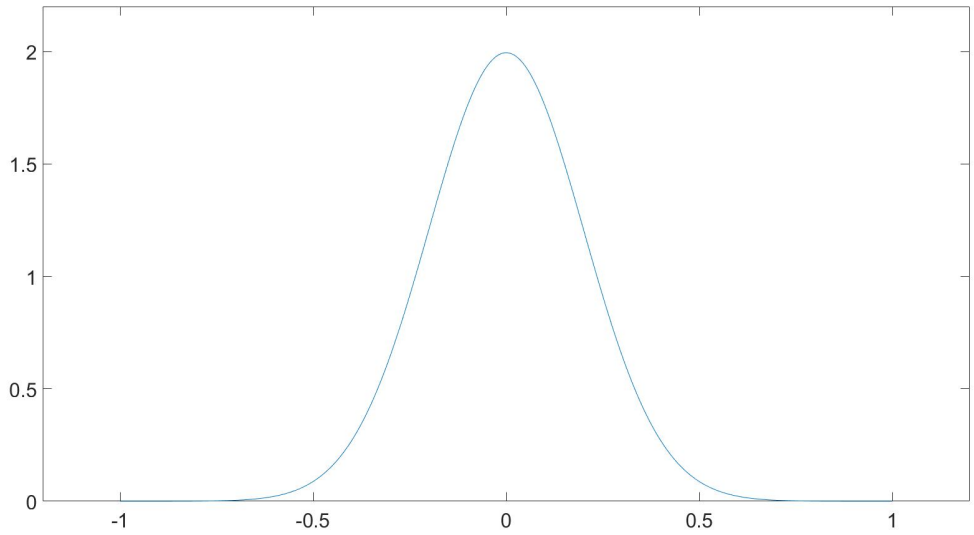


Figure 2.10: The Gaussian Function in 2D with μ=0, σ=0.2 and $A = 2$.

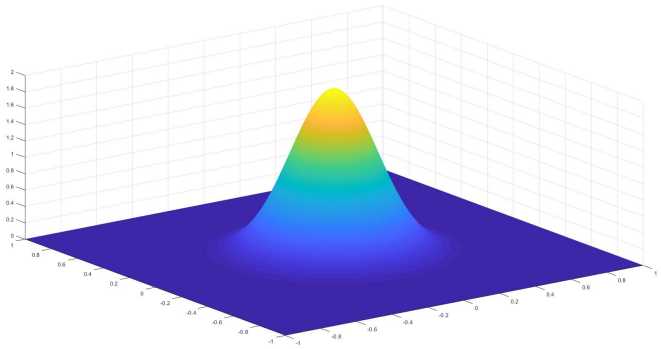


Figure 2.11: The Gaussian Function in 1D with μ=0, σ=0.2 and $A = 2$.

The general one dimensional Gaussian is described:

$$f(x) = \frac{A}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{2.3}$$

The mean value, the central limit, of the sample distribution is the value that has the greatest likelihood[1] in the distribution. The standard deviation is how much the distribution is spread out, the greater the standard deviation the fatter the distribution. The

[1]Likelihood is the value of the distribution given a fixed sample. Probability is the value of a sample given a fixed distribution.

amplitude is the likelihood at the mean of the distribution. In Figure 2.10, for example, the likelihood of a value being 0 is 2. Generally, the curve is used to determine a sample's probability density within a value range. This is the area under the distribution given an interval and is calculated using integration, as shown in 2.4.

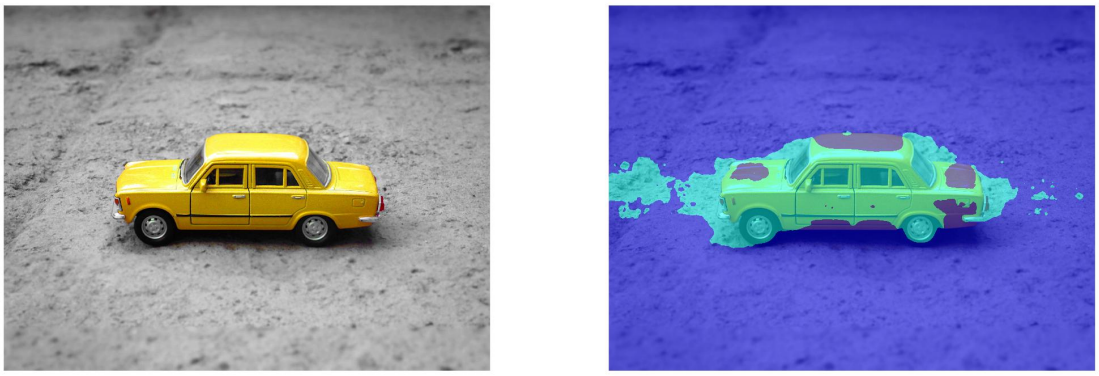
$$P(x) = \int_a^b \frac{A}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx \tag{2.4}$$

Data sets can have many dimensions and distributions need to be able to express all of them. The notation for a multi-variate Gaussian distribution, for a k-dimensional random vector $\boldsymbol{X} = (X_1, \ldots, X_k)^T$, is

$$\boldsymbol{X} \sim \mathcal{N}_k(\mu, \sigma^2) \tag{2.5}$$

2.4 Clustering

Clustering is a method of segmenting an image into disjoint sets known as classes. This is useful for separating types of features or objects in an image. For example in Figure 2.12 the toy car has been segmented from the background.



(a) Image by Gustavo, Upsplash. (b) Segmentation of car from background.

Figure 2.12: Segmentation of toy cars using K-Means Clustering.

In an image every pixel can be individually classified or groups of pixels known as super-pixels may be treated as single entities. Entities are sorted based on their similarity. For example, a pixel's intensity is a feature that could be used to cluster it with other pixels of similar intensity. As seen in (2.6) an entity's comparable traits may be represented by a feature vector. An entity may have any number of features which as also regarded as dimensions, d.

$$\vec{v} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{2.6}$$

2.4.1 K-Means

There are many methods by which to cluster, K-Means is a popular method due to its speed and simplicity. Data points are sorted into one of K clusters where the average value of the cluster is stored in a central data point known as the cluster centroid. Centroids may initially be randomly generated or selected from the given data set.

Data points or samples are placed in the class of the centroid that is closest to them. The distance between them is the Euclidean Distance between their feature vectors as in 2.7.

$$D(\vec{p}, \vec{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} \tag{2.7}$$

K-Mean's algorithm proceeds as follows:

1. Place the initial K centroids.
2. Assign all data points to their nearest centroid.
3. Update the centroids' values to be the mean of all data points in their cluster.

4. Repeat steps 2 and 3 until a criteria is met, for example until cluster centroids no longer move (converge).

K-means benefits greatly from its low computational cost such that it is oten performed a number of times with different initial centroids and the instance best result is used. The best result is defined as having the smallest intracluster variance. K-Means is disadvantaged by the implicit trait that it formulates clusters of similar sizes. This happens because the algorithm seeks to minimize variance (spread) in each cluster hence the 'ideal' centroid placement will form distributions spherically about centroids. This method cannot disentangle overlapping samples that belong in different classes. In Figure 2.13 clear edges between clusters and sphereical distributions can be observed. The algorithm can only implement *hard assignments*, as opposed to a *soft assignment* that consider the probability of a sample's class membership.

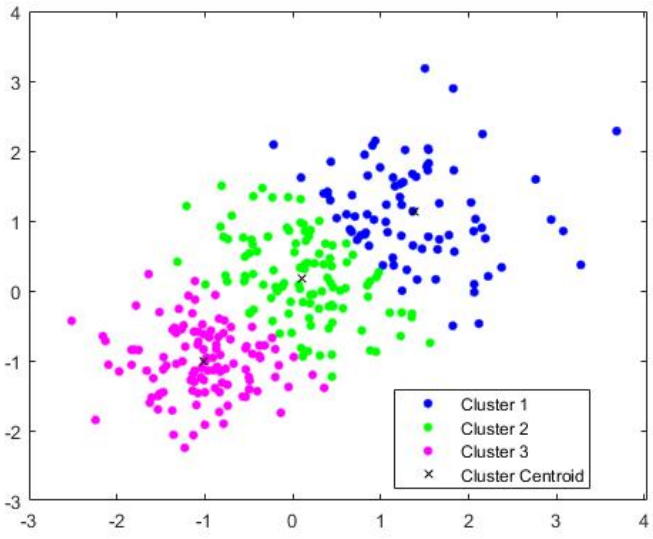


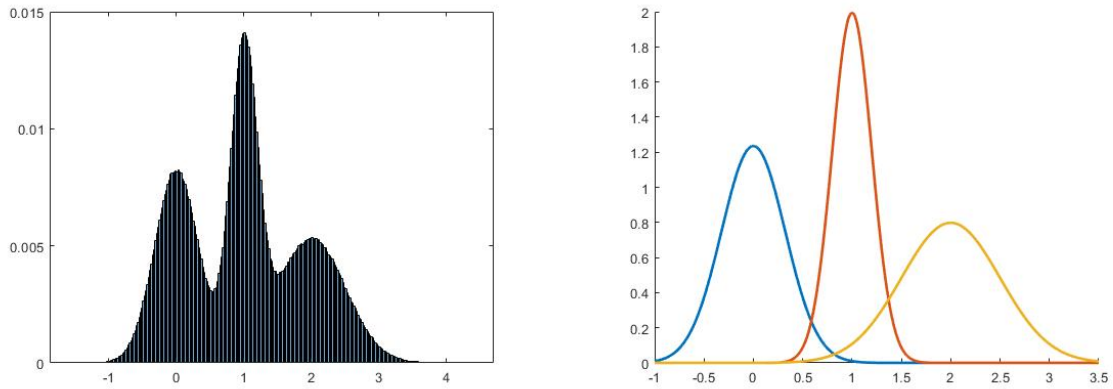
Figure 2.13: K-Means clustering performed on random data.

2.4.2 Mixtures of Gaussians

Mixtures of Gaussian or the Gaussian Mixture Model (GMM) is a method of clustering data that's able to disentangle ambiguous samples by considering a sample's probability

of belonging to a class, known as a soft assignment.

Initially, K Gaussian functions are randomly generated corresponding to K clusters. If a dataset has low dimensionality, by taking a histogram of its values the Gasussians' initial conditions can be approximated, as in Figure 2.14a. By superpositioning all of the Gaussians a sample's complete probabilistic model is created, i.e. a model for *all* clusters, see Figure 2.14b.



(a) Normalized Histogram of 1D samples. (b) Gaussians derived from 1D histogram.

Figure 2.14: Formulation of Mixture of Gaussians.

With each iteration of the GMM algorithm the parameters of the model's component Gaussians are tuned according to the covariance between samples in each cluster. In 2.8 X and Y are the variables being compared, $\overline{X}$ and $\overline{Y}$ are variable means and n is the number of samples. For a multidimensional dataset a covariance matrix $\boldsymbol{\Sigma_k}$ will be generated and each Gaussian will require a mean vector $\boldsymbol{\mu_k}$ as opposed to a scalar.

$$Cov(X,Y) = \frac{\Sigma(X_i - \overline{X})(Y_j - \overline{Y})}{n} \tag{2.8}$$

The algorithm seeks the highest covariance possible in each of its clusters. It is by considering the covariance of samples that the GMM is able to best classify ambiguous samples. The higher the covariance (aka correlation) between sample dimensions the more likely it is they belong in the same cluster. This can observed in Figure 2.15, the data being

clustered is the same as in Figure 2.13 but notice the elliptical shape of the Gaussian cluster distributions as opposed to the spherical shape of K-Means clustering.

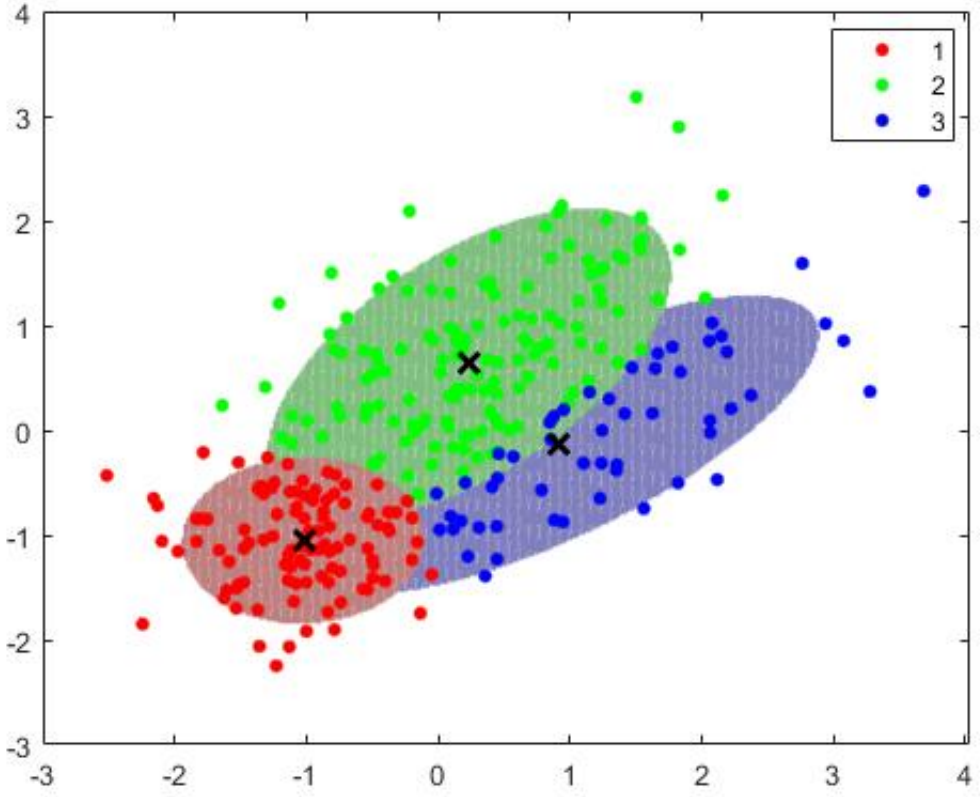


Figure 2.15: Clustering using GMM. X marks cluster mean.

In Figure 2.16 the Gaussians from Figure 2.14b have been scaled to have the amplitudes $\pi_1 = 0.3$, $\pi_1 = 0.5$ and $\pi_1 = 0.2$. These are weightings that give the mixing ratio of the GMM and represent each cluster's proportion of the total samples. The more samples a cluster contains the more likely it is a sample belongs to it. The sum of ratios must add to one because the probability a sample belongs to at least one cluster is one, as defined in 2.9 and 2.10.

$$0 \leq \pi_k \leq 1 \tag{2.9}$$

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2.10}$$

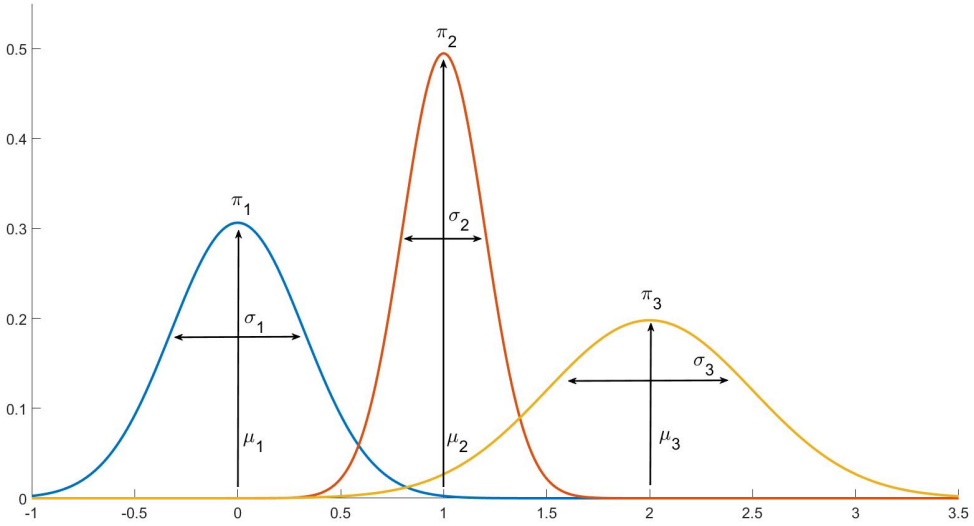


Figure 2.16: Individual Gaussians with scaled weightings.

Each Gaussian's three parameters, mean $\boldsymbol{\mu}_k$, ampltidue π_k and covariance $\boldsymbol{\Sigma}_k$ are updated following each iteration of the algorithm according to the 'Expectation Maximization' algorithm.

Expectation Maximization

Expectation Maximization (EM) is a method of updating the parameters of cluster Gaussians that seeks to maximize a sample's likelihood of belonging to said Gaussians (2.11).

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.11}$$

A binary indicator z_i exists for every sample x_i, it is 1 if the sample belongs in the cluster k and 0 otherwise such that $z_{ik} \in \{0, 1\}$ and $\sum_k z_{ik} = 1$. If the data is multidimensional then the sample and inidicator are both vectors, $\boldsymbol{x}$ and $\boldsymbol{z}$. The marginal probability $p(\boldsymbol{z}_k = 1)$ is the probability that a sample $\boldsymbol{x}$ is in cluster k. This quantity is completely specified by the mixture weight π_k for each Gaussian because the area under a Gaussian

component is equal to its mixing ratio, i.e.

$$p(\boldsymbol{z}_k = 1) = \pi_k \tag{2.12}$$

If we know that a sample $\boldsymbol{x}$ is from cluster k the *likelihood* of seeing it in the associated Gaussian is the value of the Gaussian at that point,

$$p(\boldsymbol{x} \mid \boldsymbol{z}_k = 1) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}) \tag{2.13}$$

The conditional probability of $\boldsymbol{z}_{ik}$ given the value of sample, $\boldsymbol{x}_i$ is denoted as $\gamma(\boldsymbol{z}_{ik})$. This is the probability that a sample belongs to cluster k given its value $\boldsymbol{x}_i$ and is the quantity of interest when trying to classify a sample. Using Bayes Theorem [REF APPENDIX] quantity can be found

$$\gamma(\boldsymbol{z}_{ik}) \equiv p(\boldsymbol{z}_{ik} = 1|\boldsymbol{x}_i) = \frac{p(\boldsymbol{z}_{ik} = 1)p(\boldsymbol{x}_i \mid \boldsymbol{z}_{ik} = 1)}{\sum_{j=1}^{K} p(\boldsymbol{z}_{jk} = 1)p(\boldsymbol{x}_j \mid \boldsymbol{z}_{jk} = 1)} \tag{2.14}$$

$$= \frac{\pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})}{\sum_{j=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})} \tag{2.15}$$

To maximize the likelihood of each sample being in each Gaussian the distribution parameters are modified. This is achieved by taking derivatives of the log of the likelihood function (2.11) with respect to each Gaussian parameter and setting the result to 0 to find local maxima. To take the log of the likelihood function assume all samples in are in an $N \times D$ matrix $\boldsymbol{X}$ and the corresponding indicators are in an $N \times K$ matrix $\boldsymbol{Z}$.

$$ln\, p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} ln\left\{ \sum_{k=1}^{K} \pi_k(N)(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{2.16}$$

The EM algorithm is comprised of [X] steps

1. Generate the initial K Gaussian parameters mean $\boldsymbol{\mu}_k$, covariance $\boldsymbol{\Sigma}_k$ and mixing ratios π_k either randomly or informed by a histogram.

2. Estimate the likelihood sample n was generated by cluster k for all samples.

$$\gamma(\boldsymbol{z}_{ik}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})}{\sum_{j=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})} \tag{2.17}$$

3. Maximize the Gaussian parameter using derivatives of log likelihood for each parameter.

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n \tag{2.18}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})^T \tag{2.19}$$

$$\pi_k = \frac{N_k}{N} \tag{2.20}$$

where

$$N_k = \sum_i z_{ik}$$

4. Repeat steps 2 and 3 until convergence of log likelihood (2.11) or parameters.

The Gaussian Mixture Model method of clustering is computationally complex compared to K-Means however it's ability to differentiate ambiguous samples by considering covariance is superior.

Chapter 3

Design

The design of the system is compromised of a number of submodules.

[Insert Block Diagram of Submodules Comprising Whole System]

[Then Breakdown Each Submodule]

[Why was a CNN not used?]

Bibliography

[1] The Editors of the Encyclopaedia Britannica. Industrial revolution.

[2] M. Bradley. Road congestion in australia. Technical report, 2018.

[3] WHO. Climate impacts.

[4] CITYLAB. Traffic's mind-boggling economic toll.

[5] INRIX. Congestion costs each american 1348 dollars per year.

[6] Smarter Cambridge Transport. Reducing traffic congestion and pollution in urban areas. Technical report, 2016.

[7] McKinsey & Company. Smart cities digital solutions for a more livable future. Technical report, 2016.

[8] Sabyasachi Sahoo. Deciding optimal kernel size for cnn.