License Plate Recognition system via Sparse Coding Representation

Long Chung Chan, Xiaoyi Zheng, Xinhui Ding
University of Waterloo
Ontario, Canada
lc6chan ,x94zheng ,x55ding@uwaterloo.ca

***Abstract*— License plate recognition system is one of the key pillars towards supporting intelligent traffic management. Since plate recognition system allow employing various image processing techniques and a complicated process, the research interest of this topic is huge.**

The recognition process often include three stages. Firstly, the image of a vehicle is being de-noise and turn into grey-scale. Secondly, the polygon region of the plate is detected and character separation is being done. Thirdly, the characters are being identified using intelligent classification methods. There are high demands of the license plate recognition systems, both public and private.

In this paper, an system is proposed that can recognize plates in China. License plate images in the dataset are taken at different angles, distances, vehicle conditions and periods of the day. This allows the system to incorporate various illumination conditions into account. Since license plate recognition usually takes image or video source as input, using sparse representation can be a promising direction.

The plate is localized using various contour detection methods and the plate features. Plate structure and government regulation are used for character segmentation. Finally, character recognition is done by convolution dictionary learning and convolution sparse coding. A comparison between normal convolution neural network and sparse representation is also included in this paper.

I. Introduction

License plate recognition is a common component in traffic management system and is crucial to a successful intelligent city infrastructure. The key idea is that automatic license plate recognition can significantly reduce the need of human labour and thus reducing the cost of urban city maintenance. Since this fall into the huge category of image recognition, there is a huge interest in leveraging the ability of neural network to improve the accuracy and to replace the heuristic method using deep learning.

Sparse representation has been used in a large variety of research, including de-noising [1], pattern recognition [2], image up-scaling [3] and facial recognition [4]. A key idea in sparse coding is how to select or "learn" a dictionary. In this paper, the CCPD2019 [5] dataset is being used as the main training data. Some of the early works has purposed creating a representation using fixed-base [6] while more recent research has providing proofs that a learned dictionary can achieve better performance [1]. The original goal of the sparse coding problem is not targeted towards classification problem but to signal representation and compression. However, we can perform classification by exploiting the discriminative nature of sparse representation. In this work, instead of selecting a dictionary, we learned the dictionary using the training data themselves. With sufficient samples from each class, it is possible to form a linear representation to present each samples. Yet, the representation has to be sparse enough. The generated sparse representation is then directly used to classify the samples.

Sparse representation is well suited to reduce the complexity of the input dataset of images of license plate due to the inherit regular format of license plate following the local government law. The challenge presented here is that the license plate in China include Chinese characters presenting different provinces. Even the characters come from a limited set, the chinese characters have more strokes than common alphabet or numerical digits used in the western countries. Since most of Chinese character recognition project are target towards hand-written images, they are well suit for the purpose of this research.

A key limitation of using deep learning method for a license plate recognition system is that the size of the incoming stream of data is huge compared to normal image recognition or pattern detection. It is common to have thousand of traffic flows in a urban city which license plate recognition are needed in a lot of the corners of the city's highway and road. Sending all the data to a data center for plate recognition is not an ideal solution since a lot of the old highways don't have ideal telecommunication system built in the infrastructure. Therefore a certain amount of light pre-processing is required on the embedded device in the monitoring camera. Using sparse representation can allow less amount of processing power of the distributed system and have part of the recognition implanted in camera system which has limited battery life and processing power.

The key contributions of this paper include:

- Provide detail description of the end-to-end process from vehicle image to license plate digit output
- Provide a usage example of incorporating sparse encoding into license plate recognition
- Developed a license plate recognition targeting China license plate format
- Comparison of convolution dictionary learning (CDL)

methods between Convolution Basis Pursuit De-Noising (CBPDN) [7] and Online CDL with spatial masking [8]
- Comparison of convolution sparse coding (CDC) methods between Single channel CSC [7], FISTA CBPDN solver [9] and Single channel CSC with lateral inhibition [7]
- Quantification of the accuracy-complexity trade-offs, runtime for neural network using data from the CCPD2019 [5] dataset

II. Background

A. Dataset

For our project, we use CCPD2019 [5], [10] as our dataset. CCPD is a projects refers to Chinese City Parking Dataset which consists nearly 300 thousand Chinese licence plate pictures which is recently updated in 2019. The dataset are collected from several parking lots in China, and the collecting time was from 7:30 a.m to 10:00 p.m. All pictures are taken with a handheld Android device. An example is shown in Figure 1. The dataset involves a variety of complex light and weather conditions including blurry, rainy and snowy and the dimension of each image is 720 x 1160 with RGB color (3 channels).

In this dataset, the annotations of each images is in their respective file name. The meta info is being leveraged to help us built a more robust system. The 2 most crucial data pieces are the four vertices forming around the bounding box and the license plate value encoding according the their custom encoding format.

B. Licence Plate detection algorithm

There are mainly 2 ways to employ licence plate detection which is the traditional contour detection methods and Neural Network based methods such as YOLO [11].

Traditional methods [12]–[14], leverage some common techniques of image processing, including but not limit to edge detection, de-noising, feature extraction to calculate the location the licence plate in the image.

By introducing the morphology method [12] to reduce the number of candidates significantly allows speeding up the process of plate detection. A wavelet transform based method [13] and a empirical mode decomposition analysis can also be used to detect the licence plate On top of that, detection can also be done by extracting a background color of the license plate as a crucial feature [14], [15].

For this project, the major focus is the follow-up stage on the plate character recognition. Therefore, we have developed a relatively simple plate detection algorithm having 6 stages:

1) Converting RGB-colored images to gray-scale images
2) Smoothing and de-noising are applied to the gray-scale images

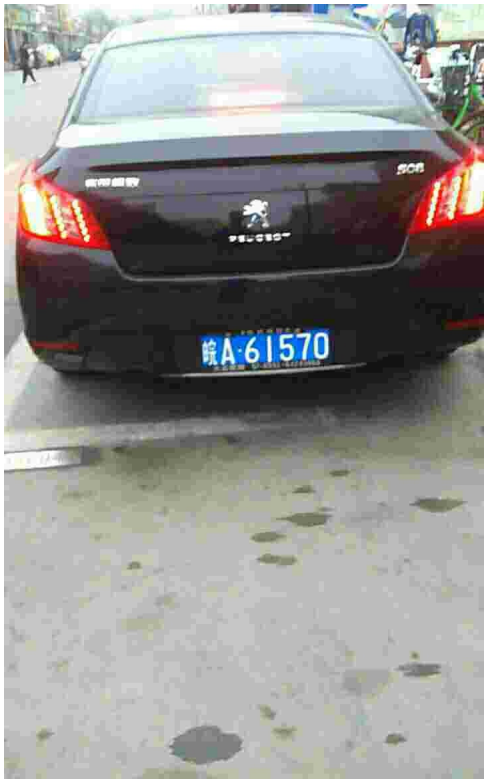


Figure 1: Example of an original image from the data set

3) Employing a SobelFeldman operator [16] to perform edge detection
4) Extracting the edges and Converting the gray-scale images into binary images
5) Filtering out the unrelated background area to create bounding area around the plate
6) Segmenting the plate area into individual characters

After the images have gone through the pre-processing steps, we can then further apply CNN-based techniques for character recognition. Also, we can then "learn" the dictionary for sparse encoding and solve for the representation for the images which more detail will be discussed in the following sections.

C. Sparse Representation

Sparse representation [17] has now become a new norm in image, signal and computational imaging. The core idea behind this successive approach is the reconstruction of a signal s from a sparse representation x according to a linear dictionary matrix D so Dx is approximately equals to s [7]. Instead of using the original signal, which is usually larger in memory size, we can then pass the sparse representation to neural-network based approach for further processing.

There are two major challenges using this kind of approach

1) How to construct a the linear dictionary D?
2) How to solve for the sparse representation x from D?

In the earlier stage of the sparse representation, fixed-base dictionary combining multiple linear filters are used [6]. The advantage of this approach is that we don't need any inputs or training set to develop the dictionary. However, the disadvantage are obvious that a one-size-fit-all approach is not going to give us the optimal result. On top of that, the quality of final sparse representation heavily depends on the dictionary it is based on. Luckily more recent researches have proposed methods to "learn" a dictionary from a set of training approaches.

To solve for a suitable sparse coding, many researches have pointed toward algorithms that is based on Alternating Direction Method of Multipliers (ADMM) [7]. ADMM's 2 main features are the decomposition of the problem into 2 stages and exploitation of the FFT for computational advantage.

Since we are mainly dealing with 2D images of license plates, the dictionary Learning and sparse coding problem being discussed will be in the convolution variant namely, Convolution Dictionary Learning (CDL) and Convolution Sparse Coding (CSC).

D. Convolution Dictionary Learning (CDL)

There are 3 main CDL variants we want to explore in this paper:

1) CDL with ADMM update [9]
2) CDL with ADMM consensus update [7], [9]

CDL with ADMM update: A very simple summary of the ADMM framework proposed is provided below:

1) Use shrinkage or soft thresh-holding to solve for the penalty parameter
2) Use Fast Fourier Transform (FFT) to deal with computationally expensive convolution operation

CDL with ADMM consensus update: Consensus update allows the images to be feed to the ADMM framework in a batch mode due to the introduction of different block matrix for mapping coefficient. This methods can reduce the time needed to solve for the dictionary.

E. Convolution Sparse Coding (CSC)

There are 3 main CSC variants that are explored in this paper:

1) Single-channel CSC
2) FISTA CBPDN Solver

Since the final image used for CDL and CSC are gary-scale images with one channels, most the the CSC algorithms chosen is single channel. The major formulation of CSC is shown below:

$$\text{argmin}_{\mathbf{x}} \frac{1}{2} \left\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \right\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 \ , \quad (1)$$

Single-channel CSC: The most basic form of a sparse coding algorithm based on the standard Basis Pursuit De-Noising(BPDN). The convolution form of the BPDN is called Convolution BPDN (CBPDN)

FISTA CBPDN Solver: This variant incorporates the use of the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) which is an accelerated proximal gradient method.

III. Workflow

A. Image Preprocessing

The raw image from the dataset contains a lot of information where they are not necessary info the character recognition stage needs. Therefore, we have to isolate the pixels related to the license plate so that we don't pass too many noise to the character recognition part.

The first step is to figure out the location of licence plate in the whole picture. Related details regarding edge detection has been mentioned back in background section. After we can draw a bounding box on the location of the license plate, we apply a binarization on the bounding area. An visualization of the mentioned workflow can be seen in Figure 2.

After obtaining the binary picture of licence plate, we then segment it into multiple pictures of individual characters. We can then obtain 7 independent character images which contains the abbreviation of province, letters and numbers.

As for the structure of a Chinese civil licence plate, it mainly consists of one Chinese character which is the abbreviation of province, follows by a letter from A to Z, which refers to the code-name of the area. For example, A normally refers to the capital of the province and finally following by 5 characters which are combinations of digits and letters(except O and I).

Applying the above-mentioned workflow to the whole raw data set allow us to generate a large number of single character images. Classification and labelling are done through a script using the meta-info contained in the images' filename, including the province, letters and numbers. This step allows us to prepare the dataset and CSV files for training the CNN used for classification.

According to the regular structure of a Chinese civil licence plate, We divide these character images into three categories:

1) Province
2) Area
3) Letter

The pre-processed dataset contains the following:

1) 13175 samples and 34 classes in letter category
2) 6713 samples and 26 classes in area category
3) 2779 samples and 31 classes in province category

In each of the categories, the samples are separated into a validation set, a training set and a testing set. The training and testing sets are for training the CNN models while the

(a) Location of the licence plate

(b) Image after cropping

(c) Image after binarization

Figure 2: Example flow of the license plate image pre-processing

validation set is for evaluating the final accuracy of the models being trained.

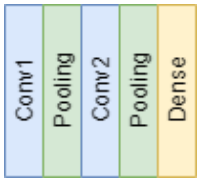
B. Sparse Preprocessing

The training images will used to learn a convolution dictionary and the sparse coding will be generated using the learned dictionary. Details on the algorithm being used was mentioned in the last section.

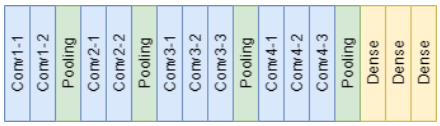
C. Classifier

We choose CNN to classify the character images. Conventional Neural Network is a class of deep neural networks, which has a good performance on analyzing visual imagery. As mentioned before, we have three categories of images, so we intend to train three independent models on these data. For letter and area, the character are letters and digits which their strokes are not complex. Therefore, the architecture of the network dealing with these two categories are similar which will be containing only two convolution layers and two dense layers.

For the province category, the Chinese characters considerably more complex and harder to classify than letters and numeric digits, so we follow a more complex architectures which is similar to VGG16 [18].



(a) Architecture of the network of letter and area



(b) architecture of network of province

Figure 3: Example flow of the license plate image pre-processing

Figure 3 shows a simplified version of the architecture of networks we are planning to use.

CNN-Sparse-based method: Considering sparse coding as pre-processing procedure to the images, we still want to compare the impact of different inputs on the performance of CNN model. We intend to use the similar network architecture to training on the sparse representing dataset.

SVM-Sparse-based method: In order to check if other machine classification methods could perform well on classifying licence plate, SVM is introduced. The main principle of SVM is that for linear data, it will find a hyper-plane to separate different classes and maximize the geometric spacing of each. It could also efficiently perform a non-linear classification by using kernel which will implicitly turn inputs into high-dimensional feature spaces.

IV. Experiments

A. Convolution Dictionary Learning (CDL) Experiments

As mentioned in the background section, we want to try out different learning algorithm mainly comparing the difference between with and without consensus update. All of the sparse representation related experiments are done on a computer running Ubuntu 19 with a Intel i7-6700K CPU with 8 threads. From the experiments we have chosen to use the province training data with 2223 20x20 gray-scale images. Both algorithms will attempt to build a dictionary with 25 channels of 4x4 filters.

Before running either of the algorithms, the images are being feed into a Tikhonov Filter to extract their respective High-Pass and Low-Pass components. An example is illustrated in Figure 8 and 10.

Figure 4 and 6 shows that we stop when the functional value of the BRPDN has been stabilized. Both algorithms runs for 200 iterations. However, the algorithm with consensus update takes only 187.18s while the other one takes

499.37s. Figures 5 and 7 shows that both resulting dictionaries are similar visually meaning that consensus method provide a 62.52% reduction in runtime without a significant reduction in the quality of the resulting dictionary.

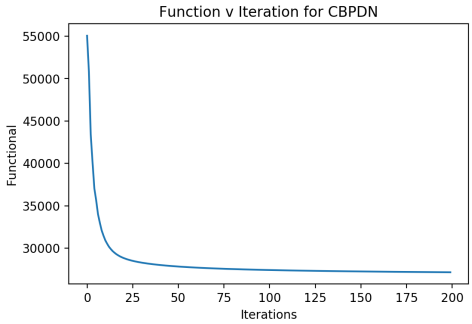


Figure 4: Functional Value against Iteration Plot without consensus

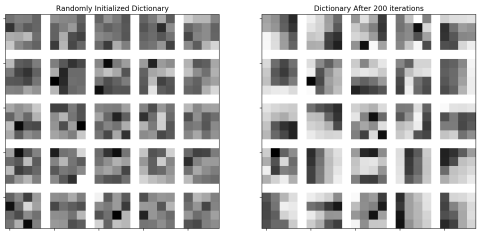


Figure 5: Resulting Dictionary without consensus

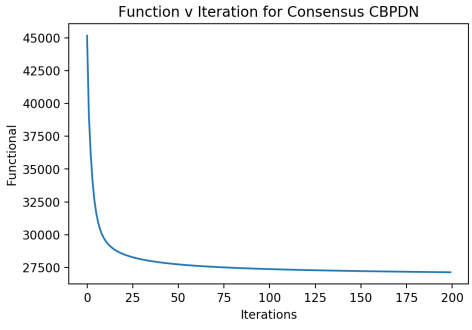


Figure 6: Functional Value against Iteration Plot with consensus

Since the CDL algorithm with consensus update save a significant amount of time, the dictionary generated using this algorithm has been chosen as the dictionary to solve for the CSC problem. However side experiment has also shown that the reconstruction Peak Signal-to-Noise Ration (PSNR) using the dictionary generated using the CDL with consensus has a lower value than the one using CDL without consensus when single-channel CSC is being used. Under the same codex of the image, a higher PSRN value usually means a better reconstruction of the image

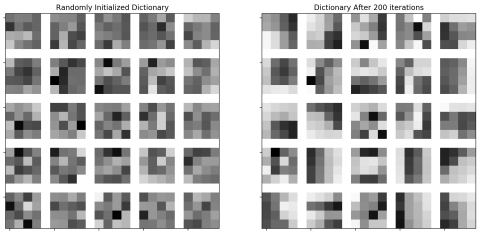


Figure 7: Resulting Dictionary with consensus

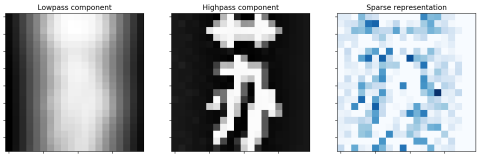


Figure 8: Lowpass, Highpass and Parse representation using single channel CSC with non-consensus dictionary

B. Convolution Sparse Coding (CSC) Experiments

In order to solve for a sparse representation for each of the images, we have tested 2 algorithms before applying it to the full dataset.

Similar to CDL, the images are being feed into a Tikhonov Filter to extract their respective High-Pass and Low-Pass components.

Comparing the runtime of single-channel CSC and the FISTA method, the FISTA methods run 10 times faster than the single-channel CSC method. The FISTA cost around 0.01s while single-channel CSC cost around 0.1s. However, as shown in Figure 15 and 11, the reconstruction of the image is not as good as the reconstruction shown in Figure 13 and 9 which use the single channel CSC method. The PSRN value using the single channel CSC method is around 23-25 dB which is always higher than 18.49 dB using the FISTa method.

On top of that, when we compare Figures 8 and 12 as a set to Figures 10 and 14, we can see that the sparse representation generated using the faster FISTA method is more noisy and less sparse.

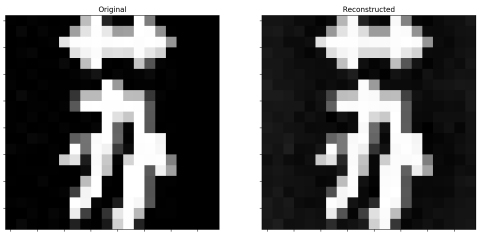


Figure 9: Image Reconstruction using single channel CSC with non-consensus dictionary

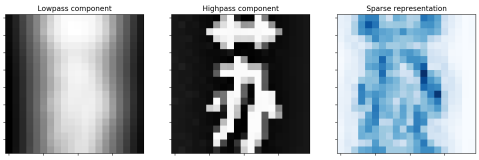


Figure 10: Lowpass, Highpass and Parse representation using FISTA CSC with non-consensus dictionary

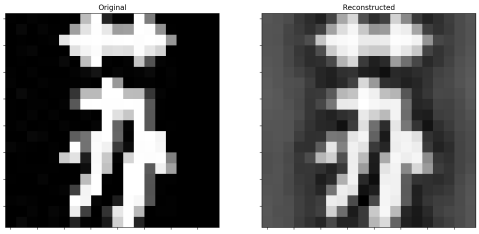


Figure 11: Image Reconstruction using FISTA CSC with non-consensus dictionary

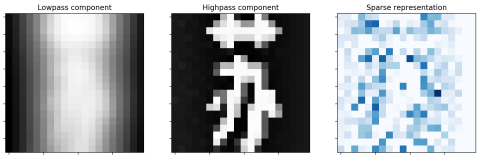


Figure 12: Lowpass, Highpass and Parse representation using single channel CSC with consensus dictionary

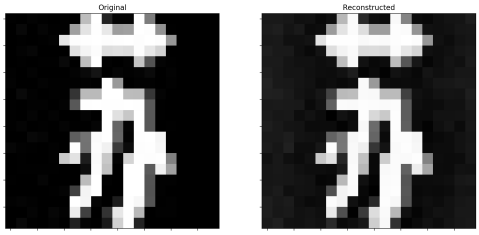


Figure 13: Image Reconstruction using single channel CSC with consensus dictionary

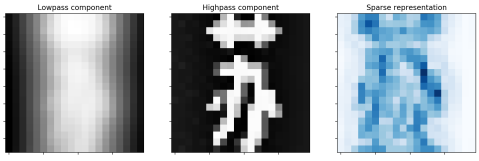


Figure 14: Lowpass, Highpass and Parse representation using FISTA CSC with consensus dictionary

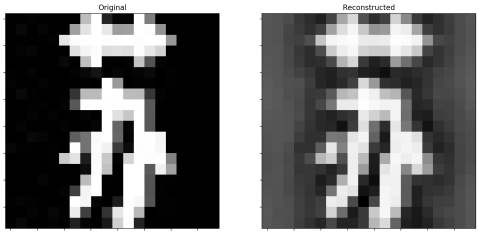


Figure 15: Image Reconstruction using FISTA CSC with consensus dictionary

C. Decision made after CDL and CSC experiments

Since there is no significant difference in the CDL algorithms in terms of quality of the dictionary. We generate to final dictionaries for each of the 3 classes using the CDL with consensus update since we are dealing with over 20000 images and the time saving from allow the consensus is significant.

However, the single channel CSC produce a more ideal sparse representation and less noisy reconstruction of the images with runtime cost penalty. Since it cost less than a second for each image, it is more reasonable to use single channel CSC to generate the sparse representation for each image instead of the FISTA method.

D. CNN Based Training

After obtaining the sparse representing data as mentioned before, we use various ways to classify them. For CNN method, we use both original data and sparse representation data to compare their performance in terms of accuracy where the original data are 20*20 binary character images. For all the data, we choose 80% of them as training data and the remain as testing data. Before feeding to the model, we normalize the original images through dividing them by 255.

1) Letter:

Original data: The data of category letter consists of 24 letters(except O and I) and 10 digits, so the architecture is very simple. After trying several times to find the best architecture. The best architecture we come up with is to use one convolution layer followed by one dense layer as the output layer. For the convolution layer, it has 16 kernels since the size of images is not big. For the output layer, it has 34 neurons which is corresponding 34 classes and uses softmax as its activation function. The softmax function allows for a probabilistic output from 0 to 1.

The figure 16 shows its architecture. And when we train the model on the Google Colab GPU, the running time of each epoch takes around 0.2s and the whole training time is about 10s for 50 epochs.

Sparse representing data: To compare the performance with the original data, we use the same architecture. When we use the same device to train the network, the running

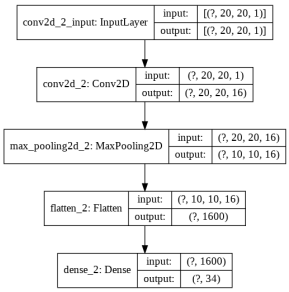


Figure 16: The architecture of CNN model of letter

time of each epoch is around 0.2s and the whole training time is about 40s for 200 epochs.

2) Area:

Original data: The data of category area consists of 26 letters which is similar to that of letter. Therefore, we use the same architecture except that for the output layer, it only has 26 neurons for its correspond 26 classes. We train the model on the same device and each epoch takes around 0.2s and the whole training time is about 10s for 50 epochs.

Sparse representing data: We use the same model as the original data. The training time for each epoch is the same except the total training time is 40s since it has 200 epochs.

3) Province:

Original data: The data of category province mainly consist of Chinese character which is not as easy as the data of letter to classify them. So after trying different models, we finalize on one with a similar architecture is similar to VGG16 [18] which is a famous and classic Deep Neural Network architecture and being used in many fields, especially in image recognition and classification. We modify the number of kernels since the size of our input image is 20*20. We also add some dropout layers to avoid over-fitting. Besides that, we also modify the number of neurons of output layers to 31 corresponding to 31 classes. The figure 17 shows the architecture of the network. Each step takes 11ms and 0.15s for each epoch. The whole train process takes around 30s for 200 epochs.

Sparse representing data: The above architecture could not work on these data and the result shows if we still use the same architecture, the accuracy will remain 10%. The reason may because the the architecture is too deep to the sparse representing data. After multiple trials, we use two convolution layers which contains 64 kernels and one fully-connected layer with 256 neurons. The output layer has 31 neurons which is of the same shape. Figure 18 presents the architecture of the model. For the run-time performance, each step takes around 6ms and 84ms for each epoch. The total training time is 42s for 500 epochs.

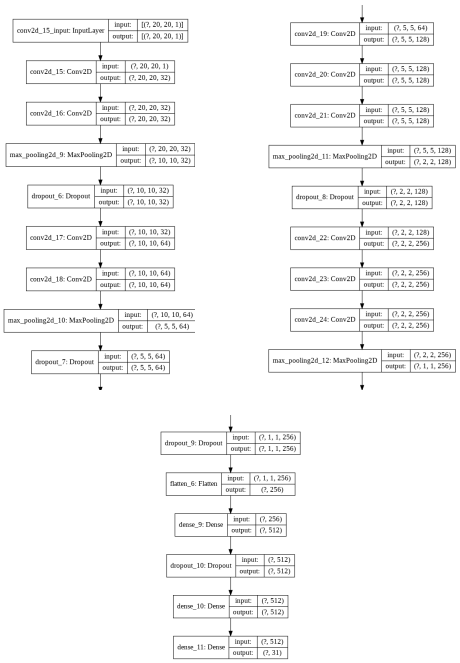


Figure 17: The architecture of original province data

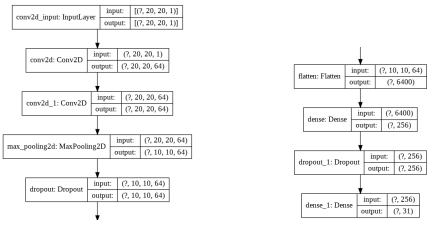
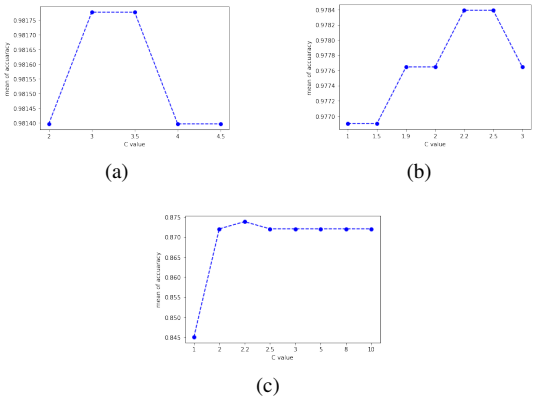


Figure 18: The architecture of sparse province data

E. SVM Based Method

Besides neuron network, we want to find the performance of using sparse data on some classical machine learning methods where SVM appears to be a reasonable option. Unlike CNN, SVM doesn't require us to spend time training the model being a classical mathematical method, therefore it is more lightweight than neuron networks. Before feeding data into the a svm classifier, we have to do PCA on the data because the sparse data has a total of 400 features which will take lots of time for SVM to classify them. We set the parameter of PCA to 0.9 which means we want to retain 90% of the variance in the original information. For SVM, it has multiple hyper-parameters. We have chosen RBF as our kernel function. For the parameter C,the regularization parameter, we try multiple C values to find the best one. From figure 19, we can see the best C values for each category is 3, 2.2 and 2.2 respectively.



(a) (b) (c)

Figure 19: (a),(b),(c) respectively shows the mean of accuracy on different C on letter,area and province

V. Evaluation

We now analysis the performance of our various methods on different inputs. For CNN, we separately use original data and sparse representing data as our inputs. For SVM, we only use sparse representing data as our inputs. We will mainly focus on the validation accuracy to evaluate different methods.

A. CNN based method

Letter and Area For letter and area, we use the same network architecture to train both original data and sparse data. Figure 20 shows the training performance on original data and sparse data. The running time and training accuracy are shown in table I. From the table, we can see the final accuracy on test, train or validation set are mostly the same. We can also see that the rates of convergence are different. For the original data, the network only need 10 epochs to converge whereas the network of sparse data converge after 175 epochs.

Method	Prediction time	Train accuracy	Validation accuracy	Test accuracy
Letter (Sparse)	249ms	0.9824	0.9782	0.9772
Letter (Original)	166ms	0.9991	0.9872	0.9905
Area (Sparse)	126ms	0.9886	0.9851	0.9851
Area (Original)	126ms	0.9988	0.9879	0.9910

Table I: Running time and accuracy on letter and area category

Province For the data of province, we use different architectures on different data set. Figure 21 shows the training performance using different input set. Table II shows the running time and accuracy. The difference between the original images and sparse representation is much more than the category of area and letter. Firstly, original images could reach 97.48% whereas the sparse data could only reach 87.74% on validation set. That could be explained that the architecture of network trained using the original images are similar to VGG16 being is a deep convolution network. As

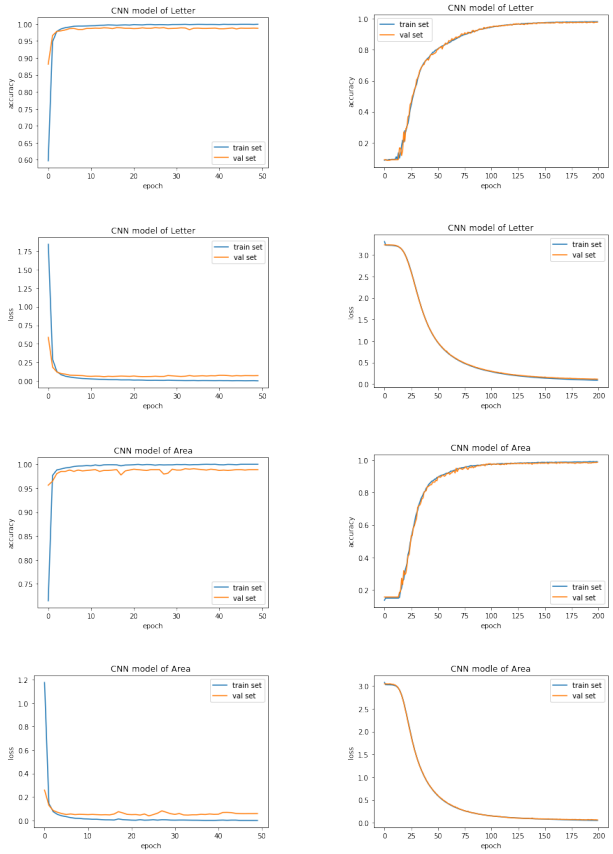


Figure 20: The right side shows the performance on original data, the left side shows the performance on sparse data

for sparse data, it has two convolution network which is a lot shallower. In addition, we can find the network of sparse data is over-fitting, although we have tried multiple ways to avoid over-fitting such as adding drop layer, reducing the depth, it still exists.

Also, in the terms of running time, the sparse data model takes less time than the original data model meaning the sparse representation can reduce the running time to some extent.

Method	Prediction time	Train accuracy	Validation accuracy	Test accuracy
Province (Sparse)	72ms	0.9927	0.8539	0.8774
Province (Original)	108ms	0.9949	0.9640	0.9748

Table II: Running time and accuracy on province category

B. SVM Based method

For SVM method, we only use sparse data to investigate the performance. The result and performance are shown in table III. From the table we can find compared with CNN method, the test accuracy is very similar, however the

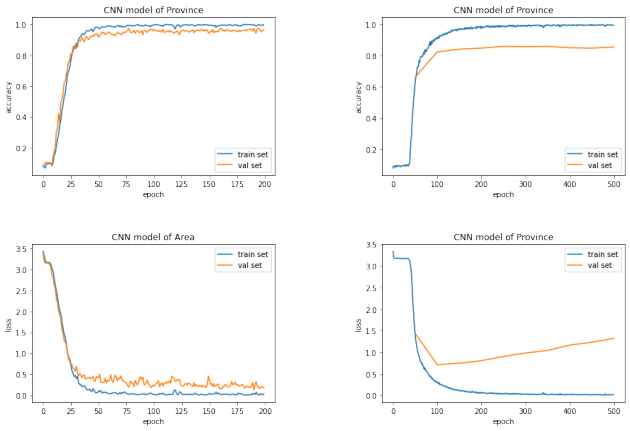


Figure 21: The right side shows the performance on original data, the left side shows the performance on sparse data

running time is different. SVM takes less time to train the model whereas more time on prediction.

Category	Training time	Prediction time	Testing accuracy
Letter	24.04s	6.17s	0.981
Area	7.67s	1.51s	0.9784
Province	4.07s	0.45s	0.875

Table III: Running time and accuracy on different inputs of province

VI. Conclusion

A. Performance

In term of CNN model, we can find that the performance of sparse data doesn't perform better than original data, especially for the province category. The reason behind it may because our original image is small enough and obtaining the sparse representation of a very small and binaries image cause an adverse effect on the info being retained in the data. In another words, causing a information loss in the sparse representation. In the end, causing an accuracy loss in the classifier.

Comparing SVM and CNN, the validation accuracies are similar. Therefore, applying sparse representation data on SVM can also be an viable option. On top of that, SVM is more lightweight than CNN which could be used easily on a resource hungry environment, such as parking-lot cameras. Also, it takes less time to train the data.

If the scenario requires a quick response time, such as the camera on highway, CNN will a better choice.

B. System Proposal

In this paper, we have successful proposed a workflow of an end-to-end license plate recognition system. Starting with image pre-processing, sparse dictionary learning and sparse coding problem to classifier model choice. We have also demonstrated the accuracy of our workflow can be up to 99.05% accuracy with the CCPD2019 [5], [10] dataset containing over thousands of images. Even if the sparse representation is not hitting as high accuracy as expected, the accuracy achievable is around 97.46% which is still a significant number. Also the sparse nature of this representation allow rooms for further compression for data storage.

References

[1] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[2] J. Mairal, F. Bach, and J. Ponce, "Sparse modeling for image and vision processing," *Foundations and Trends in Computer Graphics and Vision*, vol. 8, no. 2-3, pp. 85–283, 2014. [Online]. Available: http://dx.doi.org/10.1561/0600000058

[3] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[4] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[5] Z. Xu, W. Yang, A. Meng, N. Lu, and H. Huang, "Towards end-to-end license plate detection and recognition: A large dataset and baseline," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 255–271.

[6] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[7] C. Garcia-Cardona and B. Wohlberg, "Convolutional dictionary learning: A comparative review and new algorithms," *IEEE Transactions on Computational Imaging*, vol. 4, no. 3, pp. 366–381, 2018.

[8] J. Liu, C. Garcia-Cardona, B. Wohlberg, and W. Yin, "First- and second-order methods for online convolutional dictionary learning," *SIAM Journal on Imaging Sciences*, vol. 11, no. 2, pp. 1589–1628, 2018. [Online]. Available: https://doi.org/10.1137/17M1145689

[9] M. orel and F. roubek, "Fast convolutional sparse coding using matrix inversion lemma," *Digital Signal Processing*, vol. 55, pp. 44 – 51, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1051200416300276

[10] H. Li, P. Wang, and C. Shen, "Towards end-to-end car license plates detection and recognition with deep neural networks," 2017.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015.

[12] Jun-Wei Hsieh, Shih-Hao Yu, and Yung-Sheng Chen, "Morphology-based license plate detection from complex scenes," in *Object recognition supported by user interaction for service robots*, vol. 3, 2002, pp. 176–179 vol.3.

[13] S. Yu, B. Li, Q. Zhang, C. Liu, and M. Meng, "A novel license plate location method based on wavelet transform and emd analysis," *Pattern Recognition*, vol. 48, p. 114125, 01 2015.

[14] Z. Yao and W. Yi, "License plate detection based on multistage information fusion," *Information Fusion*, vol. 18, pp. 78 – 85, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253513000663

[15] K. Deb and K. Jo, "Hsi color based vehicle license plate detection," in *2008 International Conference on Control, Automation and Systems*, 2008, pp. 687–691.

[16] I. Sobel, "An isotropic 3x3 image gradient operator," *Presentation at Stanford A.I. Project 1968*, 02 2014.

[17] J. Mairal, F. R. Bach, and J. Ponce, "Sparse modeling for image and vision processing," *CoRR*, vol. abs/1411.3230, 2014. [Online]. Available: http://arxiv.org/abs/1411.3230

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.