Classification of Vanity License Plate Applications

Mounika Tadanki, Liz Friedman

The University of Texas at Austin

Abstract. The review process for vanity plate applications is a time-consuming process for state DMVs. Our research uses ideas from classification of malicious URLs and offensive language detection in social media to create a basic model for license plate application classification. We tested simpler models because there was some indication in related research that model complexity did not improve performance. But we found that the only neural network we tested, a multi-layer perceptron, performed best at this task.

1 Introduction

The vanity plate application process carried out by state Department of Motor Vehicles (DMV) offices currently requires a great deal of effort. In 2018 in California alone, over 250,000 vanity plate applications were reviewed to make sure that the proposed license plate did not convey an offensive message. The review process takes time, as it involves decoding the plate and attempting to see if it conveys any euphemisms, references, symbolism, or acronyms deemed inappropriate in any of several languages [1].

We have created a machine learning algorithm that can assess vanity license plate applications and predict if the license plate will be approved or not, removing some of the manual effort, and allowing DMV resources to be reallocated toward better serving the public. Our model is based on data from plates that were flagged for additional review by someone on the review committee, rather than data from all plate applications. This allowed us to focus on the issues that cause a plate application to be rejected, but also made our model less applicable to all plate applications. Because of this, we propose that our model will be helpful to the review committee but will not be able to replace it entirely.

To our knowledge, textual analysis has not been applied to vanity license plate evaluation. However, similar character-level analysis and classification has been done in the area of cybersecurity, in assessing if URLs, domain names, paths, are benign or malicious [2] [3]. This is a similar problem of analyzing a string for binary classification with the categories of "acceptable" and "unacceptable," but it has not yet been applied to strings that are license plates.

We propose that this classification of vanity license plate application strings will function in a similar way to previous attempts to classify URLs and other strings for cybersecurity reasons. Our application of this type of machine learning to a new domain will not only bring this useful method to the specific area of license plate evaluation, but may expand knowledge of the algorithms that perform well for the general task of text analysis and classification, when the input is a short string. For example, Yu et al. (2018) found that there is “little difference between various convolutional neural network (CNN) and recurrent neural network (RNN) based architectures in terms of accuracy, prompting a preference for the simpler architectures, since they are faster to train and to score, and less prone to overfitting” [2]. Would an even simpler architecture possibly perform as well, in the case of license plate classification?

2 Related Work

2.1 Topic 1: Character Level Text Analysis

Many recent works relating to character-level text analysis are applied to the cybersecurity domain, to identify malicious URLs, file paths, and similar artifacts. Convolutional neural networks seem to be popular for this task [3] [4]. Saxe and Berlin note that this automates the process of feature design, and that this is particularly helpful with systems that detect malware since new features would need to be developed as cyberattacks evolve. Models that can operate on raw input signals (such as characters) are particularly useful for text classification like this, when key features may evolve over time [3]. Yu et al. take this one step further and compare different deep learning techniques in this context, finding that there was little difference in accuracy between Convolutional Neural Networks and Recurrent Neural Networks [2]. Our work will apply these methods to a similar task of classification based on license plate characters, and will also explore the possibilities for even simpler neural networks to perform well for this task.

2.2 Topic 2: Offensive Language Detection

In addition to character-level text analysis, our model will include a feature that detects probability of profanity in the license plate. Language detection has been studied extensively in the context of hate speech or other offensive language on social media. MacAvaney et al. identify the challenges present in this task, such as differing definitions of what constitutes hate speech, but find that a SVM method performs very well and is simpler than neural network methods [5]. Susanty et al. on the other hand, do use an artificial neural network model to classify words as either offensive or nonoffensive in a way that includes consideration of the sentence structure for context. Here, they note challenges are related to the writer’s informal grammar or word abbreviations [6]. Davidson et al. attempted to classify hate speech separately from otherwise offensive language, and found that the SVM and logistic regression models performed best for this task. They also focused on areas where this distinction becomes difficult, noting that interestingly ”racist and homophobic tweets are more likely to be classified as hate speech but that sexist tweets are generally classified as offensive" [7]. We will apply this sort of attention to detecting offensive language to the arguably equally public (albeit more limited in number of characters) context of vanity license plates.

2.3 Topic 3: License Plate Evaluation

Many machine learning models related to license plates involve image processing to accurately predict a license plate in a given image. Kim et al. note that license plate recognition is in sharply increasing demand as states shift to Intelligent Transportation Systems, and that plate detection is still sensitive to environmental factors like lighting. They use a convolutional neural network to deal with these challenges [8]. Montazzolli Silva et al. note that although there are many of these license plate recognition system models, most only function for a specfic region or country, and focus specifically on views of the front of the car. Their system is built to work without these constraints, also using a convolutional neural network [9]. There seem to be many variations of this—Zhang et al. developed a particularly complicated model that first identifies the location of a license plate using a Sobel-Color, MESR, and SVM algorithms, and then uses a LeNet-5 convolutional neural network [10]. Our work will be useful to some of the same stakeholders and organizations as these works, but it will deviate from this in that it will be evaluating the license plate as a set of characters rather than an image.

3 Methods

3.1 Data Processing and Feature Selection

Our first step involved choosing the scope of the data we would like to use. We found two complete datasets: one from New York that included all vanity plate applications, separated out by whether they were accepted or rejected; and one from California that consisted only of license plate applications that were flagged for additional review. These two datasets were opposites in terms of the label ratio. The New York dataset had an overwhelming number of accepted plates compared to rejected plates, and the California dataset had the opposite. Additionally, the New York dataset only included the requested plate and the date of the request as features, while the California dataset included a review reason code that referenced why a plate was flagged for additional review as well as comments from the reviewer and applicant. We decided to use the Calfornia dataset so that there would be more data about the rejected plates, and we could better attune an algorithm to recognize what should be rejected with more data. The drawback to this is that our algorithm will be most useful only once a plate has already been flagged for review. Applying this to all applications could be an area for further research, and in the meantime this remains an example of machine learning algorithms assisting, but not replacing, human labor.

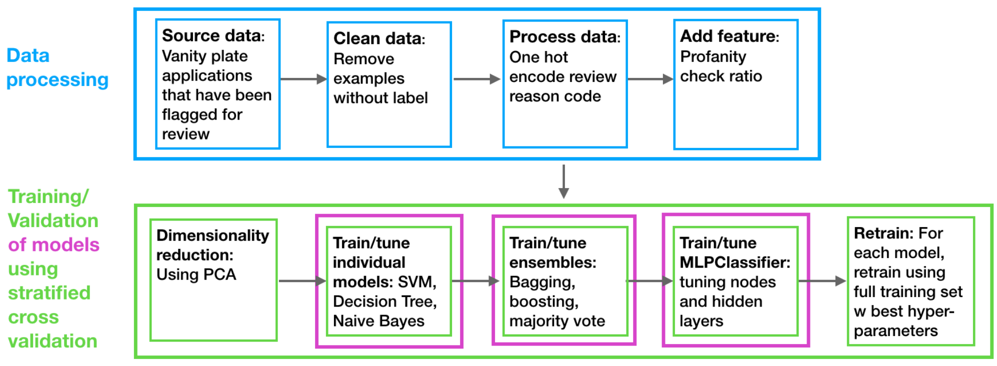
Data processing
Source data: Vanity plate applications that have been flagged for review
Clean data: Remove examples without label
Process data: One hot encode review reason code
Add feature: Profanity check ratio
Training/ Validation of models using stratified cross validation
Dimensionality reduction: Using PCA
Train/tune individual models: SVM, Decision Tree, Naive Bayes
Train/tune ensembles: Bagging, boosting, majority vote
Train/tune MLPClassifier: tuning nodes and hidden layers
Retrain: For each model, retrain using full training set w best hyper-parameters

Fig. 1. Flowchart illustrating steps in our machine learning system

To clean the data, we removed any examples that did not have the binary Y/N label and made sure all examples included a proposed plate. In order to process the license plate characters and the review reason code, we decided to one-hot encode these features. License plate characters were one-hot encoded on letters, digits, and a limited number of special characters as well as a space. With this method characters that appear more than once in a plate will just be counted once. A more complicated encoding may be an area for further research. For the review reason code, we used categorical one-hot encoding. Our final feature was added using a pre-existing profanity check that outputs a probability that a string includes profanity.

3.2 Training and Validation

Our first training step was to reduce the number of features, which was high with the multiple one-hot encoded features. In order to reduce the risk of over-fitting, we used Principal Component Analysis as our dimensionality reduction technique. To do this we used stratified cross-validation to determine the number of principal components. We used stratified cross-validation for determining all hyperparameters, and made sure to do this because our dataset was so skewed toward the rejected class. We wanted to make sure that an equal number of accepted class examples were used in validation folds.

We chose to train and compare the performance of three non-deep learning classification algorithms: Decision Tree, Support Vector Machine, and Naive Bayes. We chose these three algorithms because we have found them to perform best for tasks like this. For each algorithm we used stratified cross validation to determine the optimal hyperparameters. Next, we created ensemble models from these three algorithms and tested different ensemble methods (majority vote, bagging, and boosting). Finally, we trained a Multi-Layer Perceptron to compare the performance of these simpler models with the neural network. With all of these models, we tested one version without the profanity check ratio feature and one with this feature. Our goal here in testing all of these algorithms is to see if a simpler model than the Convolutional Neural Networks used in related works can perform this task well, and to see if the profanity check ratio improves performance.

4 Experimental Design

4.1 Evaluate feature dimension size using PCA

The purpose of this experiment is to identify what feature dimension sizes are best, through observing the performance of a decision tree classifier for different feature dimension values using Principal Component Analysis. We used stratified cross validation to evaluate the decision tree classifier, and used recall macro as the evaluation metric because our dataset was skewed toward the rejected class. We chose recall macro instead of recall micro or recall weighted because although the other recall metrics had higher scores, they did not reflect recall performance in the way we expected. We realized this after comparing each version of the recall score to confusion matrices. We ran each experiment (including this one) twice, once with including profanity check ratio feature and once without to see if that feature improved performance.

4.2 Compare performance of individual classification algorithms

The purpose of this experiment is to identify the optimal hyperparameters for different classification algorithms (Decision Tree, SVM, and Naive Bayes) using stratified 5-fold cross validation, and to see if with these optimal hyperparameters they can beat the baseline of a model that predicts one class or guesses randomly. Because we were not able to find a previous research for this topic, the baseline we used to compare against these classifiers was just a model that used random guessing, or guessed all No (Rejected) or all Yes (Accepted). After identifying hyperparameters we re-trained the classification algorithms with these optimal hyperparameters and compare their performances to identify the best performing model.

For the Decision Tree classifier, we tuned the split criterion and tree depth hyperparameters. For the SVM classifier, we tuned the polynomial degree, kernel bandwidth, and the regularization parameter. We continued to use recall macro as the evaluation metric but also reported accuracy and precision.

4.3 Evaluate ensemble learning methods

The purpose of this experiment is to evaluate the effect of using different types of ensemble methods for the classification task. We used stratified cross validation to evaluate Majority Vote, Bagging method, and Boosting method classifiers. The individual classifiers in these ensembles were either SVM, Naive Bayes, or Decision Tree models. The baseline here was to see if the ensembles performed better than the individual models did. We continued to rely on recall as our main evaluation metric.

4.4 Evaluate performance of a multi-layer perceptron model

The purpose of this experiment is to compare the performance of different classifiers with the multi-layer perceptron model and see how the use of this neural network improves performance. So the baseline we are comparing this experiment to is the best-performing model from the previous experiments. We tuned the hidden layers and number of neurons per layer hyperparameters for this model. We set all other parameters constant when training; e.g., activation function, number of iterations for training, batch size, and gradient descent approach. We evaluated the different classifiers using stratified cross validation and continued to use recall as the evaluation metric.

5 Experimental Results

5.1 Evaluate feature dimension size using PCA

When we used PCA to test different feature dimension sizes on a Decision Tree classifier, we found that the highest recall score for the Decision Tree classifier that did not include the profanity check ratio was about 0.5532, and this was with 25 features. The highest recall score for the Decision Tree classifier that did include the profanity check ratio was about 0.5527, and this was with 30 features. The recall scores here were very similar, but using a different number of features. We used these two feature dimension sizes when performing the other experiments.

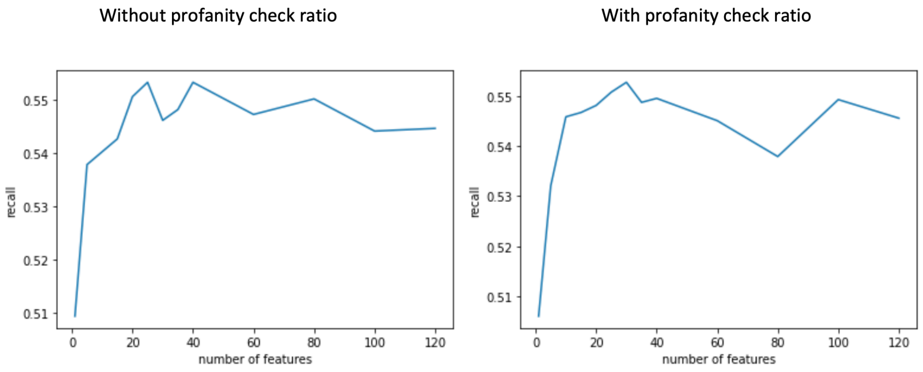

Fig. 2. Number of principal components and recall for Decision Tree classifier with the profanity check ratio and without

5.2 Compare performance of individual classification algorithms

We optimized hyperparameters for each algorithm twice, once without including the profanity check ratio feature and once with including it. For the decision tree classifier, the optimal criterion was gini, and the max depth was 24 when not including offensive ratio and 25 when including it. For both SVM models, the optimal hyperparameters were degree = 2, gamma = auto, C = le-12, kernel = poly. The hyperparameters used for both Naive Bayes models were the default.

We found that the decision tree model with the profanity check ratio had the highest recall score, and this was slightly higher than decision tree without the profanity check ratio. Both SVM models performed the worst, and actually did not improve on the baseline—it always predicted to reject the license plate application. However the Decision Tree and Naive Bayes models both improved upon the baseline.

Algorithm	Mean Recall(Without Profanity)	Mean Recall(With Profanity)
DecisionTree	0.5523	0.5535
SVM	0.5000	0.5000
Naive Bayes	0.5237	0.5230

Table 1. Evaluation metrics for individual classification algorithms with and without the profanity check ratio

After observing that DecisionTree performed better during evaluation, we retrained DecisionTree algorithm with criterion value as gini and max-depth value as 25 and checked the performance of this model against the test dataset with and without profanity check ratio. We observed that the DecisionTree performed slightly better with the profanity check ratio. The performance of the DecisionTree algorithm against the test dataset was similar to what was observed during training.

Metric	DecisionTree(Without)	DecisionTree(With)
Recall	0.55	0.56
Precision	0.55	0.56
Accuracy	0.71	0.73

Table 2. Test Prediction results for DecisionTree with and without the profanity check ratio

We found that DecisionTree model with the profanity check feature performed with a recall of 0.56 against the test dataset.

5.3 Evaluate ensemble learning methods

We found that the ensemble learning methods did not improve much on the baseline from the previous experiment, which was the Decision Tree model. The recall for the bagging and boosting Decision Tree ensemble methods remained

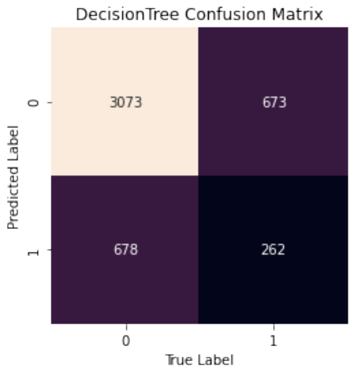

Fig. 3. Confusion Matrix for DecisionTree without profanity check ratio

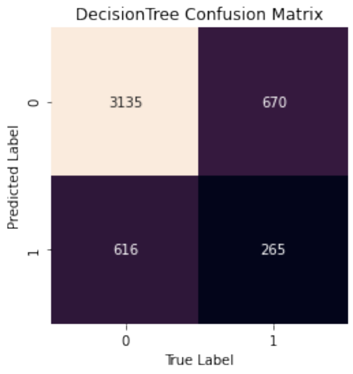

Fig. 4. Confusion Matrix for DecisionTree with profanity check ratio

the same, and was actually slightly less for the model that included the profanity check ratio.

After evaluating the ensemble methods, we tested the Bagging - DecisionTree and Boosting - DecisionTree models against the test dataset and found that the results were similar to what was observed during training. These ensemble learning methods failed to improve much on the baseline from the previous experiment.

5.4 Evaluate performance of a multi-layer perceptron model

We found that the MLP model did perform better than the baseline, which was the Decision Tree classifier (individual and ensembles, since those performed equally as well). The MLP model improved recall scores as well as precision and accuracy. While tuning the hyperparameters for the MLP, we observed that a single hidden layer performed better than multiple hidden layers. During evaluation, we also observed that the MLP model performed better without PCA

Algorithm	Mean Recall(Without)	Mean Recall(With)
Majority Vote Classifier	0.5161	0.5163
Bagging classifer - DecisionTree	0.5430	0.5428
Boosting classifier - DecisionTree	0.5480	0.5438

Table 3. Evaluation metrics of ensemble methods with and without profanity check ratio

Metric	Bagging - DecisionTree	Boosting - DecisionTree
Recall(Without)	0.55	0.55
Recall(With)	0.55	0.55

Table 4. Test Prediction results for DecisionTree Bagging and Boosting methods with and without the profanity check ratio

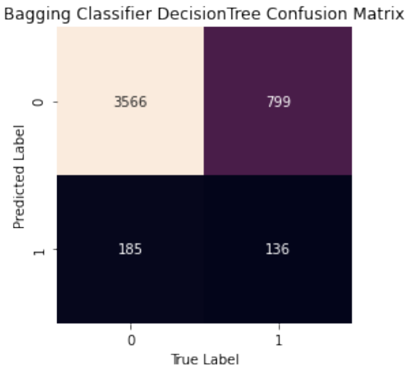

Fig. 5. Confusion Matrix for Bagging-DecisionTree without profanity check ratio

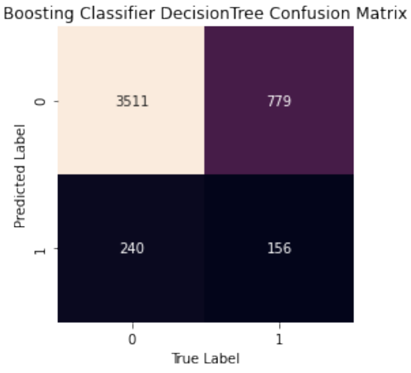

Fig. 6. Confusion Matrix for Boosting-DecisionTree without profanity check ratio

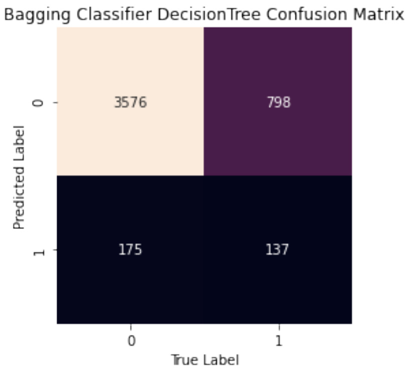

Fig. 7. Confusion Matrix for Bagging-DecisionTree with profanity check ratio

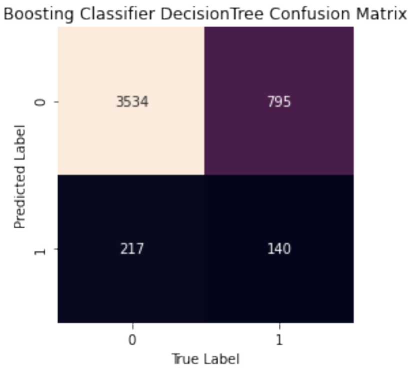

Fig. 8. Confusion Matrix for Boosting-DecisionTree with profanity check ratio

than with PCA. The models performed similarly with and without the profanity check during evaluation. We tested the MLP models without PCA against the test dataset.

Model	Recall
MLP without Profanity and without PCA	0.5926
MLP without Profanity and with PCA	0.5899
MLP with Profanity and without PCA	0.5925
MLP with profanity and with PCA	0.5889

Table 5. Evaluation metrics for MLP Classifier with and without PCA and with and without the profanity check ratio

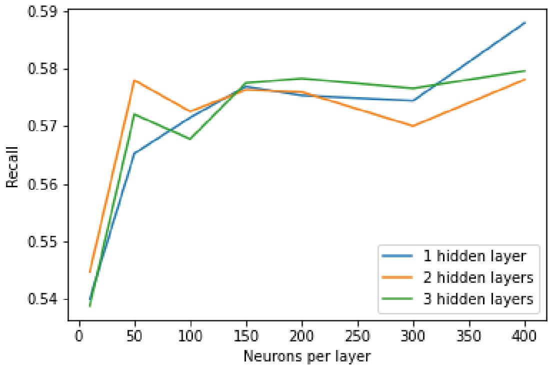

Fig. 9. Hidden layers Vs No. of Neurons Vs Recall

Metric	MLP(Without Profanity Check)	MLP(With Profanity Check)
Recall	0.59	0.58
Precision	0.60	0.60
Accuracy	0.75	0.75

Table 6. Test Prediction results for MLP without PCA and with and without the profanity check ratio

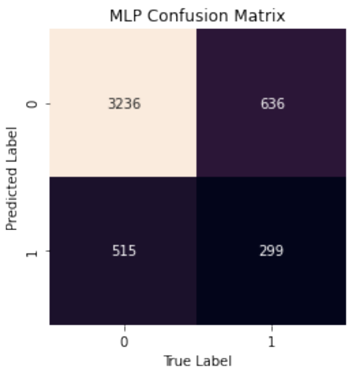

Fig. 10. Confusion Matrix for MLP without Profanity check

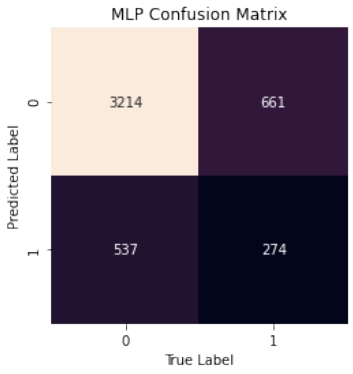

Fig. 11. Confusion Matrix for MLP with Profanity check

The MLP model without profanity check performed better than the model with profanity check against the test dataset with a recall of 0.59 which was better than the decision tree model which had a recall of 0.56.

5.5 Further research

We conducted these experiments to see how simpler models with character level analysis performed in the case of license plate classification, and found that the most complex model (the MLP classifier) performs the best. An extension to this research could be to analyze how convolutional neural network (CNN) and recurrent neural network (RNN) based architectures perform the task of classifying license plate applications. As the California dataset that we used contains only vanity plates that have already been flagged for review, applying this to all applications could be another area for further research. And because we have no knowledge about the offensive word list that the California DMV uses to accept/reject applications, we have used an open source list of offensive words to check for offensive license plates. More knowledge about the DMV's procedure in reviewing vanity license plate applications might help in improving this model.

6 Conclusions

We set out to discover if a simpler neural network than a convolutional neural network could perform character-level analysis, given that CNNs were found to perform just as well as RNNs in at least one case. [2] But we were also testing to see if it is possible that a non-deep learning model could match or exceed this performance, particularly when given the additional profanity check ratio and review reason code features included in this work. We found, however, that the MLP neural network outperformed the simpler models for this specific dataset.

With a more balanced dataset, the simpler model performance may improve to the point that they can compete with the neural networks.

Nevertheless, this model, or an improved one with further research, can be used to reduce the time spent on reviewing vanity plate applications. For example, if it takes a review team 30 minutes to review a vanity plate application after it has been flagged for review, and their ability to reference the machine learning model's prediction reduces that time by just 5 minutes, the time savings for 250,000 plate applications (the number of plates reviewed in California in 2018) is more than 20,000 hours [1].

References

1. Braslow, S.: The dmv reviewed thousands of hilarious vanity plate applications last year: These are our favorites. Los Angeles Magazine (2019)
2. Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M.: Character level based detection of dga domain names. In: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE (2018) 1–8
3. Saxe, J., Berlin, K.: expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys. arXiv preprint arXiv:1702.08568 (2017)
4. Lin, S.Z., Shi, Y., Xue, Z.: Character-level intrusion detection based on convolutional neural networks. In: 2018 International Joint Conference on Neural Networks (IJCNN). (2018) 1–8
5. MacAvaney, S., Yao, H.R., Yang, E., Russell, K., Goharian, N., Frieder, O.: Hate speech detection: Challenges and solutions. PloS one **14**(8) (2019)
6. Susanty, M., Sahrul, Rahman, A.F., Normansyah, M.D., Irawan, A.: Offensive language detection using artificial neural network. In: 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT). (2019) 350–353
7. Davidson, T., Warmsley, D., Macy, M., Weber, I.: Automated hate speech detection and the problem of offensive language (2017)
8. Kim, H.H., Park, J.K., Oh, J.H., Kang, D.J.: Multi-task convolutional neural network system for license plate recognition. International Journal of Control, Automation and Systems **15**(6) (2017) 2942–2949
9. Montazzolli Silva, S., Rosito Jung, C.: License plate detection and recognition in unconstrained scenarios. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 580–596
10. Zhang, M., Yu, W., Su, J., Li, W.: Design of license plate recognition system based on machine learning. In: 2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC), IEEE (2019) 518–522