US license plates Image Classification using Machine Learning

Shane Yokota

1. Introduction

1.1. Task Definition

The goal of this project is to design and evaluate machine learning models to classify U.S. license plates by states. The dataset used for this project is from Kaggle (Dataset). The dataset contains images of size 3×128×224 with 56 labels including 50 states in the United States, District of Columbia(Washington DC), and 5 US territories. All images are same size in a jpg format and has been cropped so that at least 90% of the images are occupied by the license plates.

1.2. Motivation behind this project

The motivation for this project comes from an interest in computer vision and the challenge of fine-grained image classification. U.S. license plates often share similar layouts (rectangular plates with a sequence of alphanumeric characters) and may have only subtle differences in color schemes, fonts, or small state logos. Distinguishing 56 different plate designs is significantly more complex than a binary or few-class problem, making it a good testbed for advanced image recognition techniques. The task is also relevant to automated license plate recognition systems and state identification for traffic analysis. Personally, applying deep learning to a real-world image classification task is appealing, and comparing it with more traditional machine learning approaches will provide insight into the strengths and weaknesses of each approach on a fine-grained, multi-class problem.

2. Related Work

Convolutional Neural Networks (CNNs) for Image Classification[1]: CNNs have become the dominant approach for image classification in recent years. They automatically learn hierarchical feature representations from raw images and have achieved state-of-the-art results on benchmarks like ImageNet with hundreds or even thousands of classes. In general, "Convolution neural network (CNN) has been widely applied in many fields and achieved excellent results, especially in image classification tasks.". Architectures such as AlexNet, VGG, and ResNet have demonstrated that deep CNNs can far outperform traditional methods on image recognition. For example, a CNN-based model (AlexNet) winning the ImageNet 2012 competition was a breakthrough that spurred broad adoption of deep learning for vision. Given this success, using a CNN is a natural choice for license plate classification, as it can learn to recognize the distinguishing patterns (colors, text, logos) of each state's plate with enough data.

Feature Extraction + SVM Classification[2][3]: Before deep learning became big, a common strategy for image classification was to extract hand-crafted features and then apply a classifier like a Support Vector Machine (SVM). Popular feature extraction techniques seem to include converting images to grayscale and computing descriptors such as Histogram of Oriented Gradients (HOG) or applying Principal Component Analysis (PCA) for dimensionality reduction. These methods aim to distill the important visual information (edges, textures, shapes) into feature vectors that an SVM can use to separate classes. Studies have shown that using HOG features can significantly improve the accuracy of classifiers like SVM on image tasks. For instance, Yusof et al. report that an SVM combined with HOG features outperformed other combinations and achieved the highest accuracy in their image classification problem. In general, "HOG has demonstrated a good performance in improving ... image classification" with SVMs. Even in license plate recognition, hybrid approaches have been explored – e.g., using a CNN to extract features and an SVM to classify – sometimes yielding competitive results in scenarios with limited training data. This related work suggests that while deep CNNs are powerful, feature-engineering plus classical classifiers remain a viable baseline or complementary approach, especially for moderate-sized datasets.

3. Dataset and Evaluation

3.1. Dataset information

The dataset is from Kaggle (Dataset). The dataset contains coloar images on vehicle liccense plates. There are total of 8,721 license plate images (all in 3x128x224 and jpg format) classified one of the 56 classes (50 states + D.C. + 5 territories).
Additionally, all images are cropped so that at least 90% of the images are filled with the license plate.

3.2. Train/Test/Valid split

The data set already has train, test, and valid split with 8161 images(93.6%) for train, 280 images (3.2%) for test and valid.
The images are relatively uniformly distributed for each class in the training data: all classes have around 140-160 images.
The test set and validation set each contain 5 images per class for a total of 280 each as well. This test set and validation set simplify the evaluation as accuracy will not be biased by class frequency.

3.3. Sample Visualization

3.4. Evaluation

We evaluate model performance primarily using classification accuracy on the validation and test sets. Accuracy is computed as the percentage of license plate images correctly classified into the right state/territory. Given the balanced classes in val/test sets, accuracy is an appropriate summary metric (and is equivalent to macro-average recall in this case).
In the future, I plan to use confusion matrix. With confusion matrix we examine the resulted predictions to see which specific classes are misidentified as others. This will help me identify if certain states are commonly confused.

4. Methods

4.1. Method 1: Convolutional Neural Network Model

For deep learning approach, I implemented a Convolutional Neural Network (CNN) based on the ResNet-50 architecture. ResNet-50 is a 50-layer deep residual network known for its effective accuracy on image classification tasks due to its use of skip connections that mitigate vanishing gradients. For the final layer, I have added a linear transformation so that ResNet-50's default output of 2048 dimensional vector will be transformed to a 56 dimensional vector to match my class count.

Transforming Images:
Since ResNet-50's input layer is 224x224 by default, I transformed the provided images by resizing them and cropping them to 224x224 size. The reason behind this was from my observation of the dataset. The key differentiator of the license plate was in the background, top section, and bottom section in my opinion. Therefore, this resizing not only fit the image to the required input size, but also got rid of some unncessary information on the left and right side of the license plates.

Dataset design:
The given data from the dataset were in jpg form and label was in strings (i.e. ALABAMA, ALASKA, etc). Therefore, using custom dataset class, I transformed the image into a tensor ranging from 0 to 1 and label into a one hot encoded tensor where a specific index is 1 and rest are 0 for each classes.

Training with ResNet-50:
Now that all the data were formatted to my requirements, I used binary cross entropy as the loss function. CNN was trained using Adam optimizer with a learning rate of 0.001. I trained for 25 epochs on the training set (8161 images), using batches of 32. During training, I monitored the loss and accuracy on the test set after each epoch. During each epoch, I saved the current best model which were compared with the average loss on the test dataset.

4.2. Method 2: Feature Extraction + SVM Model

For the second approach, I used a traditional machine learning pipeline that extracts features from the images and feeds them into a Support Vector Machine (SVM) for classification. Instead of hand-crafted features like HOG or SIFT, I leveraged the power of modern deep networks by using a pre-trained ResNet-50 model to extract features. This method is often referred to as transfer learning through feature extraction.

Transforming Images and Feature Extraction:
Each image in the dataset was resized and normalized before being passed through the pre-trained ResNet-50 model. I removed the final classification layer and extracted the output from the second-to-last layer, resulting in a 2048-dimensional feature vector for each image. These features represent high-level patterns that the ResNet model learned on ImageNet, including shapes, colors, and textures that are useful for differentiating license plates.

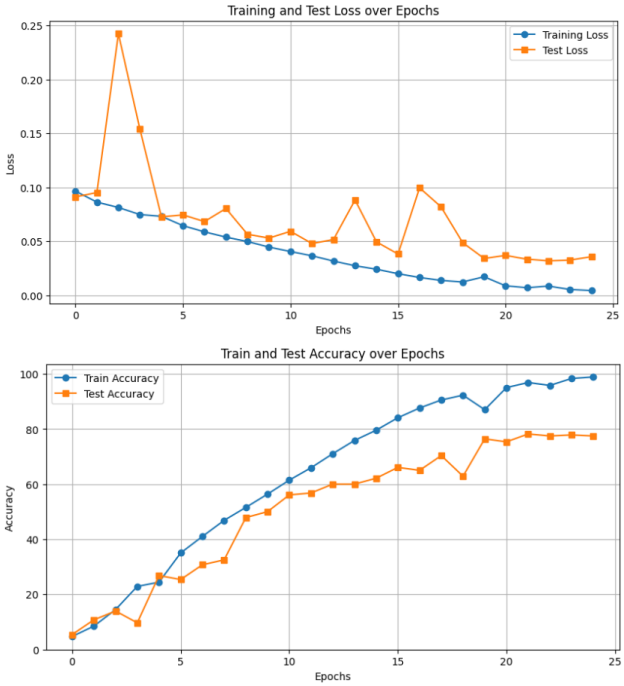
Preprocessing Features:
To prepare the data for the SVM, I applied feature normalization using scikit-learn's StandardScaler. This ensured that each feature dimension had zero mean and unit variance, which is important for SVMs to perform well. The labels were again one-hot encoded to allow multi-class classification.

Training the SVM::
Using the processed features, I trained an SVM classifier with an RBF kernel. The SVM was trained on the 8161 examples from the training set and then evaluated on the validation and test sets. The RBF kernel was chosen to capture potential non-linear decision boundaries between states. While the SVM training was fast due to the fixed feature size, its performance lagged behind the CNN model, likely due to the complexity of the classification task and the lack of end-to-end learning.

5. Experiment

5.1. CNN Experiments

After running the CNN model, I got the following result:



The CNN model achieved the result of 99% accuracy on the training dataset and 81% on the validation dataset. This data shows that CNN retained good generalization to the new images.
As shown in the graph, the accuracy over epochs were improving consistently (except for the 19th epoch for training and 18th epoch for testing dataset) throughout the 25 epochs. Importantly, test loss in general was not increasing after a certain point (despite few big bumps). This indicates that the model did not suffer from overfitting during training.
The performance seems good in general, but there are still room for improvement to reduce certain loss and accuracy bumps through out the epochs and to reduce the gap between training accuracy and test accuracy.

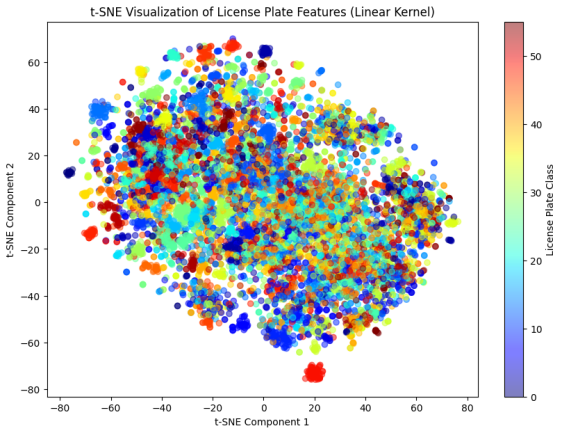
5.2. Feature Extraction + SVM Experiment

I tested two different SVM models using the ResNet-50 extracted features: one with a linear kernel and one with an RBF (Radial Basis Function) kernel.
For the linear kernel, the model achieved an accuracy of 100% on the training set, 53.93% on the test set, and 55.36% on the validation set. For the RBF kernel, the accuracy dropped slightly to 91.03% on the training set, 52.86% on the test set, and 53.21% on the validation set.

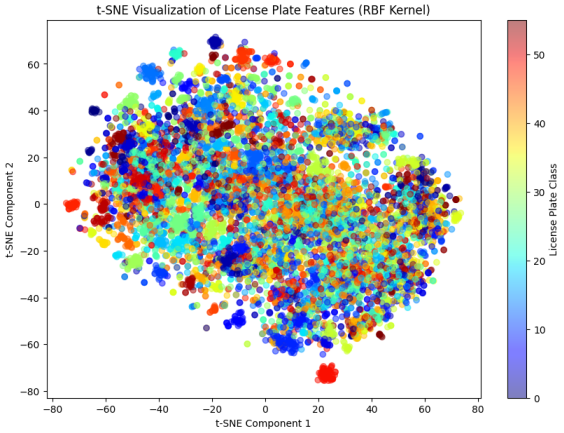
Kernel Type	Train	Test	Valid
Linear	100%	53.93%	55.36%
RBF	91.03%	52.86%	53.21%

These results suggest that while the linear kernel perfectly fit the training data (potentially overfitting), it did not generalize well to new images. On the other hand, the RBF kernel generalized slightly better but is still far worse than the CNN model by a lot.

To visualize the feature space, I used t-SNE to reduce the 2048-dimensional ResNet features to 2 dimensions. The resulting scatter plots show the separability of different license plate classes:



Scatterplot: SVM Linear kernel



Scatterplot: SVM RBF kernel

In both visualizations, we can see that while some clusters of classes are distinguishable (i.e. red cluster at the bottom), there are lots of overlapping dots. This could be contributing to the lower classification performance. These plots indicate that although ResNet features capture some meaningful structure, they may not be linearly separable for this complex 56-way classification task.

6. Discussion

In the final CNN model, I noticed couple bumps in training and validation loss and accuracy, especially around epochs 18–20. While the overall trend was positive, these bumps may be an indications of instability in images that I plan to investigate further—possibly by tuning the learning rate, adding regularization, or trying other image transformation methods.

Although the CNN achieved 99% training and 81% validation accuracy, there's still a large gap (20%). To push the test accuracy closer to 90%, I may explore other image preprocessing strategy. My current approach crops the left and right sides of images, which may have unintentionally removed useful features. Using padding, different cropping strategies, or even color space adjustments could help keep more relevant information. I also wonder whether converting to grayscale could simplify the task or hurt performance due to the loss of color-based cues—this is something worth testing.

For the SVM model, even with ResNet50-based features and an RBF kernel, test accuracy dropped around 53–55%. Since these features are fixed, the model lacks the ability to adapt to the specific classification task. While further tuning or dimensionality reduction may help slightly, I believe there's limited room for significant improvement without fine-tuning the feature extractor or moving to a hybrid CNN-SVM approach.

7. Conclusion

Through this project, I learned that fine-grained image classification, such as identifying U.S. license plates by state, is a challenging task even with powerful models. The CNN approach, especially using ResNet-50, performed well and showed the strength of deep learning in learning discriminative features directly from data. However, I also found that small decisions in preprocessing—like how images are cropped—can significantly impact accuracy. On the other hand, the SVM model, while much faster to train, struggled to match CNN performance due to its reliance on fixed features and lack of end-to-end learning. Overall, this project helped me understand the trade-offs between traditional and deep learning methods, the importance of thoughtful data handling, and the value of analyzing model behavior beyond just accuracy.