US license plates Image Classification using Machine Learning

Shane Yokota

1. Abstract

I explore the problem of classifying U.S. license plates into 56 state and territory classes using deep learning. My final system uses a ResNet-50 convolutional neural network, fine-tuned on a dataset of ∼8,700 cropped plate images. I experiment with several preprocessing and training strategies, including full vs. partial fine-tuning, dropout regularization, and varying levels of data augmentation. My final model, trained with moderate augmentation and no freezing or dropout, achieves 98.73% training and 96.07% test accuracy — a significant improvement over the baseline ResNet-50 model's 78%. I conclude that architectural simplicity, balanced by carefully chosen data transformations, yields the most robust performance for this fine-grained classification task.

2. Introduction

2.1. 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.

2.2. 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.

2.3. Final Results

My final model, a fully fine-tuned ResNet-50 with moderate data augmentation, achieved 98.73% training accuracy and 96.07% test accuracy. This reflects a significant improvement over the baseline ResNet-50 model's performance of 78% and demonstrates strong generalization across all 56 license plate classes.

3. 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.

Dropout: A Simple Way to Prevent Neural Network from Overfitting[4]: Srivastava and others (2014) introduced Dropout, a regularization technique for neural networks aimed at reducing overfitting by randomly "dropping out" units during training. The paper highlights how dropout prevents complex co-adaptations between neurons, essentially training an ensemble of subnetworks that collectively generalize better. By turning off a random subset of neurons at each training iteration, the network becomes more robust and less reliant on specific paths of computation.
The authors demonstrated improvements across a wide range of tasks, including image classification, speech recognition, and document classification. Notably, dropout proved especially effective in deep neural networks such as CNNs, where overfitting is a common challenge due to large model capacity.
In my project, I observed that the CNN model for baseline ResNet-50 were showing early signs of overfitting in the training set (Figure 3). To address this, I implemented dropout layers to help with overfitting.

Data-augmentation on fine-tuned CNN model performance[5]: This study investigates how different data augmentation techniques affect the performance and robustness of CNN. The authors explore methods such as image flipping, rotation, cropping, and color jittering, showing that data augmentation can synthetically expand the training dataset and expose the model to more diverse input variations. This ultimately leads to better generalization, particularly in tasks where the original training set is limited or lacks variability.

The paper emphasizes that data augmentation acts as a form of implicit regularization, forcing the model to learn invariant and more generalizable features. It also highlights that different augmentation strategies impact model performance differently, with spatial transformations like rotation and scaling being especially beneficial in image classification tasks.

In my project, I introduced data augmentation to the CNN's training dataset to increase the diversity of input images and prevent the model from memorizing the training data. In addition, my implementation made it so that the data augmentation happens as we get the input data - making more epoch equivalate to synthetic data expanding. This helped expose the network to a wider range of image variations and improved its ability to generalize to unseen test samples. Additionally, it helped with overfitting as each training data now had some variance.

4. Dataset and Evaluation

4.1. 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 as 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.

4.2. 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.

4.3. Sample Visualization

Figure 1. Visual Example of Input Images

4.4. Evaluation

I evaluate model performance primarily using classification accuracy on the validation and test sets. Accuracy is com-

puted 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).

5. Methods

5.1. 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 (Figure 2). 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.

Figure 2. Transformed Input Images

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.

Modification since Midterm Report:
Compared to the midterm model, the final CNN method introduced key refinements: image resizing instead of cropping, use of ImageNet normalization, and data augmentation to increase robustness. While several variations were tested (e.g., layer freezing, dropout, alternate FC layers), the final model retains full fine-tuning of ResNet-50 with moderate augmentation and a direct linear classifier.

5.2. 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.

Modification since Midterm Report:
Given the performance of ResNet-50 CNN Model, I concluded that the potential improvement from modified Feature Extraction + SVM Model was not going to outperform the CNN model. Therefore, no further experiment was conducted with this Model.

ID	Data Augmentation	Layer Freeze	Drop-out	LR	Epoch	FC Final Layer	Train Acc	Test Acc	Valid Acc
A1	Approach1	-	-	1e-3	25	Linear(2048,56)	98.31%	84.29%	82.14%
A2	Approach2	-	-	1e-3	25 (Early Stopping)	Linear(2048,56)	87.24%	81.42%	87.14%
A3	Approach2	-	-	1e-3	35 (Early Stopping)	Linear(2048,56)	96.70%	88.93%	85%
A4	Approach2	-	-	1e-3	40	Linear(2048,56)	98.02%	93.57%	87.5%
A5	Approach2	Freeze layer 1 & 2	0.5	layer3: 1e-4 layer4: 1e-4 fc-layer: 1e-3	30	Linear(2048,512), ReLU(), Dropout(0.5), Linear(512,56)	26.85%	32.50%	32.14%
A6	Approach2	Freeze layer 1	0.5	1e-4 for all layers	30	Linear(2048,512), ReLU(), Dropout(0.5), Linear(512,56)	81.57%	74.64%	78.21%
A7	Approach1	-	0.5	1e-3	25	Linear(2048,512), ReLU(), Dropout(0.5), Linear(512,56)	79.99%	65%	68.21%
A8	Approach3	-	-	1e-3	40	Linear(2048,56)	98.73%	96.07%	92.14%

Table 1. Model Comparison Results

6. Experiment

6.1. CNN Experiments

The base CNN model described in 5.1 used a ResNet-50 CNN with a center crop of the license plate image to 224×224 and a final fully connected (FC) layer mapping from 2048 to 56 output classes. has the following result. The model achieved 99% training accuracy and 78% validation accuracy (Figure 3). While generalization was reasonable, I noticed a increasing gap between train and test accuracy starting at epoch 10 and occasional instability in loss values around epochs 12-18, suggesting potential overfitting. In addition, the fact that training accuracy is near 100%, but testing accuracy is 78% shows some concerns with my dataset variation.

This prompted a series of experiments (A1 through A8 displayed in *Table 1* below) aimed at refining data augmentation, training stability, and architectural design to close the performance gap and improve generalization.

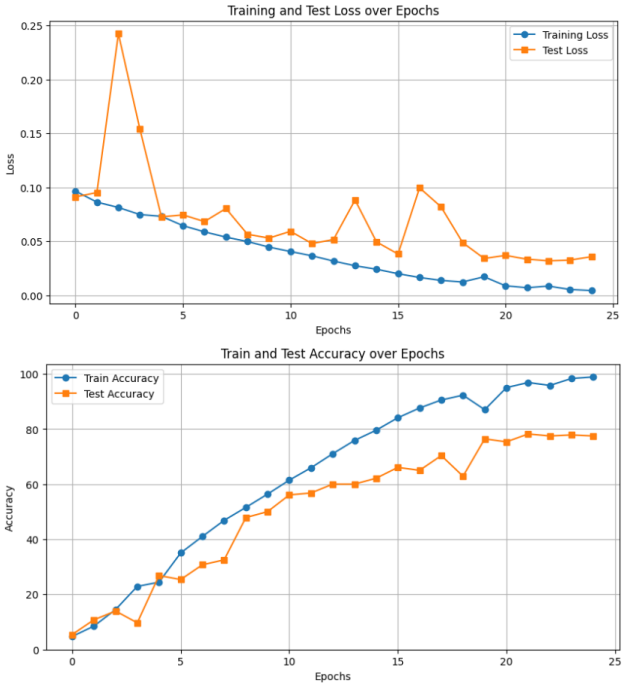


Figure 3. Base model CNN result

6.1: A1 - Stretch Input Instead of Cropping

In the first modification (A1), I hypothesized that cropping the left and right sides of the license plate image might re-

sult in losing important features, such as state logos, figures such as bird or fruit that symbolizes the state, or boundary text. To preserve all visual content, I replaced the crop with a stretch resize from 128×224 to 224×224, maintaining all image information but distorting aspect ratio - Data Augmentation Approach 1.

The learning rate, fully connected final layer, and epoch

Figure 4. Approach 1 transformed images

number was kept the same. Resulting in CNN with 25 epoch, 1e-3 learning rate, and 2048 to 56 linear transformation as the final layer.

As seen on the table, the result was very good: **98.31%** training accuracy, **84.29%** test accuracy, and **82.14%** validation accuracy.

6.1: A2, A3, A4 - Moderately Aggressive Data Augmentation and extended training (35 and 40 Epochs)

In A2, I returned to the A1 model but added data augmentation. This included moderate random rotation, brightness, contrast, saturation, hue, rotation, translate, scale, and sheer (Approach 2). In addition, the image was normalized so that each color channel (R, G, and B) was standardized. The goal was to introduce visual variance and force the model to learn more robust features invariant to lighting, minor distortions, and image quality differences.

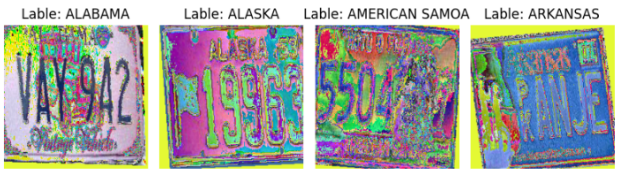


Figure 5. Approach 2 transformed images

Compared to A1, only the data augmentation step has changed. Resulting in CNN with data augmentation, 25 epoch, 1e-3 learning rate, and 2048 to 56 linear transformation as the final layer.

The results were looking good; however, the upward trend in accuracy had not plateaued, prompting a longer training test. Thus, I continued training A2 model to 35 epochs (A3), then to 40 epochs (A4), tracking whether accuracy would continue rising or begin to overfit.

A2 Result: As seen on the table, the result was good but worse than A1: **87.24%** training accuracy, **81.42%** test accuracy, and **87.14%** validation accuracy.

A3 Result: As seen on the table, the result was much better than A2: **96.70%** training accuracy, **88.93%** test accuracy, and **85.00%** validation accuracy.

A4 Result: As seen on the table, the result was very good but slight overfitting began after epoch 34: **98.02%** training accuracy, **93.57%** test accuracy, and **87.50%** validation accuracy on epoch 34.

6.1: A5 - Freezing ResNet Layers + Dropout

Following common practice in transfer learning, I explored freezing early layers of ResNet-50 to preserve pre-trained ImageNet features while training only the later layers and classifier. In A5, I froze layers 1 and 2, used a lowered learning rate (1e-4) for layers 3 and 4, and added the regularization via dropout and deeper FC layer (Linear(2048,512) -¿ReLU() -¿ Dropout(0.5) -¿ Linear(512,56)).

This resulted in a sharp performance drop with **26.85%** training accuracy, **32.50%** test accuracy, and **32.14%** validation accuracy. I attributed this to the model being overconstrained, unable to adapt the pre-trained features to license plate structures. This confirmed that in the domain, full fine-tuning was more ideal.

6.1: A6 - Partial Freezing

To test whether A5 failed due to overly restricted training, A6 repeated the experiment but only froze layer 1, with all other layers trainable and learning rate set to 1e-4 across the board.

This resulted in improved accuracy of **81.57%** training accuracy, **74.64%** test accuracy, and **78.21%** validation accuracy. However, the model showed signs of early overfitting and plateaued prematurely. Therefore, I concluded that freezing even one early layer still hindered learning meaningful task-specific features — suggesting that my dataset is clean and well-distributed enough to benefit from full fine-tuning without freezing.

6.2 A7 - Dropout and Fully Connected Layer

Due to the result in A5 and A6, I concluded that full fine-tuning of Res-Net50 was more beneficial for this usecase. However, I wanted to see if regularization via dropout and more complicated FC layer would still provide model improvement via sufficient regularization to combat the increasing gap in test and training accuracy that was seen in A2, A3, and A4. Therefore, I tested the model with A1 but FC layer and dropout.

This resulted in decrease in performance with **79.99%** train-

ing accuracy, **65.00%** test accuracy, and **68.21%** validation accuracy. I concluded that this FC architecture, perhaps in combination with dropout, was too restrictive and discarded useful features learned by the backbone. It also pointed to the strength fo a simpler and direct linear transformation for FC final layer.

6.1: A8 - Final Model: Simpler Augmentation, No Freezing

From all the previous experiment (A1-A7), I returned to fully fine-tuned ResNet50 with direct linear transformation FC Layer, no dropout, and moderate augmentation (Approach 3) to address the gap in train and testing accuracy seen in Approach 2. This approach balanced diversity with realism: instead of strong transformations (A2–A4), I applied smaller-scale random rotation, brightness, contrast, saturation, hue, rotation, translate, scale, and sheer to avoid distorting key spatial features like text layout or state related features.
This resulted in A8 achieving **98.73%** training accuracy, **96.07%** test accurct, and **92.14%** validation accuracy by epoch 40 (Figure 5). The training curve plateaued mostly fine, and generalization was the best of all variants.

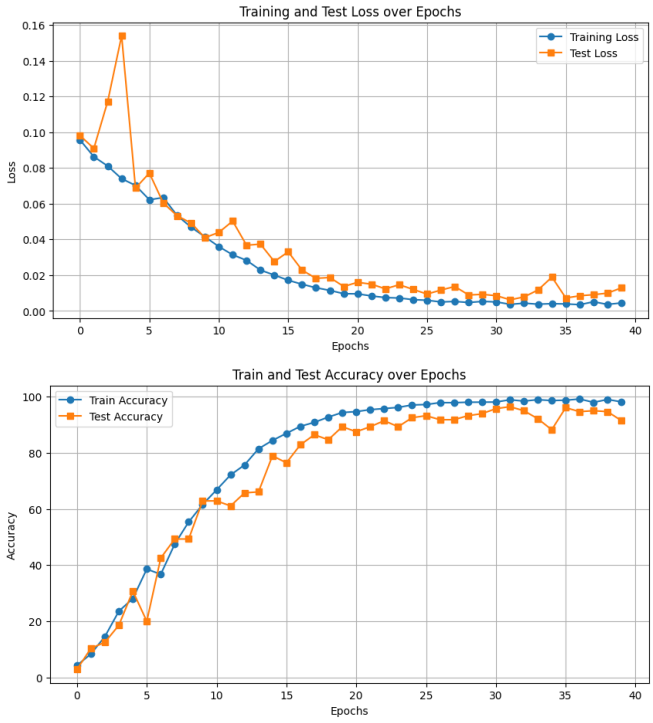


Figure 6. A8 results

6.1: Why the Final Method Works Best:

The final model (A8) overcomes the midterm's limitations by retaining full visual information in the input (no cropping), improving generalization through controlled augmentation, and avoiding over-regularization that restricted earlier variants. Unlike the baseline model, which showed instability and a large train-test gap, the final model exhibits smoother convergence and reduced generalization error (98.73% train, 96.07% test accuracy, and 92.14% validation accuracy). By balancing architectural simplicity and moderate input variability, it achieves the best real-world performance.

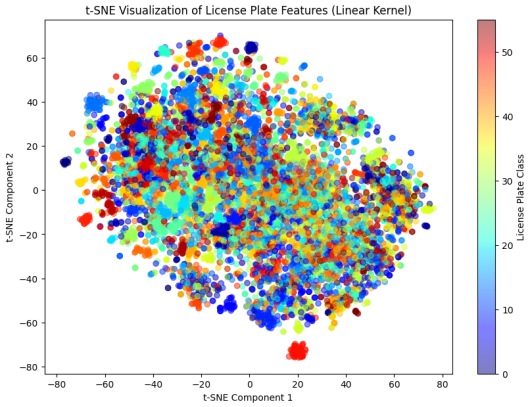
6.2. Feature Extraction + SVM Experiment

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.

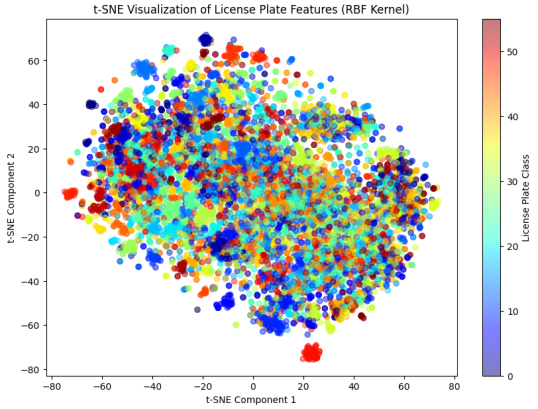
Kernel Type	Train	Test	Valid
Linear	100%	53.93%	55.36%
RBF	91.03%	52.86%	53.21%

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.

7. Discussion

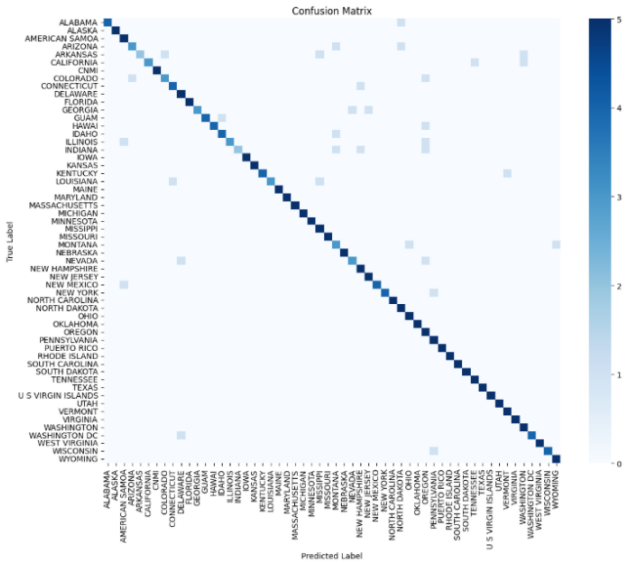


Figure 7. Final Model's Confusion Matrix

The final model achieved 96.07% test accuracy, a ∼15% absolute improvement from the base model's 78% test accuracy. This was made possible by eliminating aggressive image cropping, adding moderate augmentation, and avoiding unnecessary regularization like dropout or layer freezing.

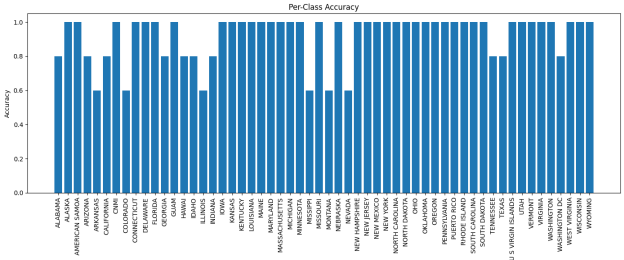


Figure 8. Final Model's per class accuracy

Analysis of the confusion matrix (Figure 7) and per-class accuracy (Figure 8) revealed that most states were classified correctly with high confidence. Errors tended to occur between visually similar plates, such as the below.

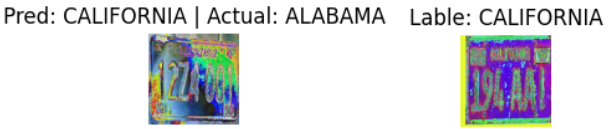


Figure 9. Wrong Prediction of California vs Real California Plate

From my experiments, I found that ResNet-50's full capacity was necessary for this fine-grained task. Freezing layers or applying dropout harmed performance, likely due to underfitting or suppression of useful features. This suggests that with clean and well-distributed data, full fine-tuning is more effective than aggressive regularization. Interestingly, even moderate augmentation helped the model generalize better, likely by exposing it to realistic variations such as lighting and slight rotation.

These insights align with the intuition that CNNs benefit most from small architectural modifications when the core model is already well-optimized for image classification tasks.

8. Conclusion

In this project, I designed and evaluated models for classifying U.S. license plates into 56 categories using a ResNet-50 CNN and a feature-extraction-based SVM. Through a series of refinements — such as stretching instead of cropping inputs, introducing moderate data augmentation, and fully fine-tuning ResNet-50 — I significantly improved performance from 78% to 96.07% test accuracy. This model also successfully performed 92.14% on validation test, which is the highest among all model from A1-A8.

I also found that dropout, additional FC layers, and freezing layers decreased performance, suggesting that my dataset

was clean and representative enough to benefit from a simpler, fully fine-tuned model. The SVM approach, while fast to train, plateaued at 53% and lacked the flexibility of end-to-end learning.
Overall, this project deepened my understanding of transfer learning, data augmentation, and architectural tuning in CNNs, and demonstrated that thoughtful refinements to data and training strategy can lead to substantial gains in fine-grained image classification.

References

Code and Results: Code and Results (Github Link)

References

[1] C. Luo *et al.*, "How Does the Data Set and the Number of Categories Affect CNN-Based Image Classification Performance?," Mar. 8, 2019. [Online]. Available: https://scispace.com/pdf/how-does-the-data-set-and-the-number-of-categories-affect-1d9c48n0nq.pdf

[2] N. N. Abd Rahman, "HOG and Haralick Feature Extractions with Machine Learning for Vehicle Type Recognition," *International Journal of Engineering Trends and Technology*, vol. 70, no. 10, pp. 202–209, 2022. [Online]. Available: https://ijettjournal.org/Volume-70/Issue-10/IJETT-V70I10P202.pdf

[3] Y. Yu, Y. Yan, and Z. Liu, "SVM License Plate Recognition Method Based on PSO Algorithm," *ResearchGate*, Mar. 2023. [Online]. Available: https://www.researchgate.net/publication/369342738_SVM_License_Plate_Recognition_Method_Based_on_PSO_Algorithm

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[5] B. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019. [Online]. Available: https://pdfs.semanticscholar.org/6086/30604cf7b62579930425ab57cc4191c034c9.pdf