Low-Data Deep Learning for License Plate Blurring

Parth Sarin Patricia Wei
Stanford University
Palo Alto, California
{psarin,patwei}@stanford.edu

Abstract

In this paper we develop an image segmentation system for license plates that performs nearly as well as the state-of-the-art YOLO model, but is trained from scratch on less than 1% *of the data used to train YOLO. We begin by analyzing the role of surveillance in supporting the project of computer vision, to sit with the problematic nature of the task at the center of this work. For example, none of the license plate datasets we found reported the provenance of their images or whether people consented to or were compensated for having their car's picture taken. Somewhat ironically, we next describe how we trained our model, drawing on prior work about license plate recognition and segmentation. Although prior studies addressing the same task have transferred weights from large pretrained models, this study focuses on low-data situations. Consequently, we describe a carefully chosen model architecture, loss function, and training routine, including data augmentation and hyperparameter tuning, that led to good results when training from scratch. We report the performance of that model, along with a number of ablations, and discuss implications for deep learning on small datasets, license plate blurring, and systems of surveillance.* [1]

1. Introduction

The surveillance camera is a widely proliferated piece of technology: In California, just a few years after the September 11th attacks, 37 cities had already adopted systematic, city-wide surveillance programs and the number has grown rapidly since then [16]. At the same time, private companies like Amazon have enabled and promoted widespread surveillance conducted by individuals, in the name of their own safety, through devices like the Ring doorbell [17]. Under United States law, police can generally obtain access to these recordings during criminal proceedings, so police in California—and, indeed, governments everywhere—have access to more data about their citizens than ever before, leading to a regime that academic Roger Clarke has called "dataveillance" [17, 3].

As many critical technology scholars have pointed out, these regimes are highly opinionated about which bodies ought to be monitored and controlled—namely, systems of surveillance tend to reinforce normative social hierarchies including various systems of caste [1, 7, 12, 13]. As a field, computer vision has largely profited off these systems of control in the form of datasets that can be used for training. One review of 114 datasets commonly used in computer vision concluded they value "efficiency at the expense of care; universality at the expense of contextuality; [and] impartiality at the expense of positionality" [15]. Given that so much of decolonial theory is about provincializing and problematizing efficiency, universality, and impartiality, an inescapable conclusion is that computer vision is complicit in Western hegemonic and imperial projects [4, 10, 5, 9].

Even in our attempts to "fix" systems, the computer vision community has carelessly disregarded the social contexts of algorithms in favor of quantifiable conceptions of equity. When Joy Buolamwini and Timnit Gebru published *Gender Shades*, showing that facial recognition systems don't work as well for black people or women, they called for more dataset diversity to address the problem [2]. People listened, and updated facial recognition algorithms. But the result, as Ruha Benjamin has pointed out, was that surveillance cameras became much more useful for police—rather than "fix" algorithms with more representative data, she has called on the community to consider whether it is even possible to "create fair and just algorithms in an unfair and unjust society" [14].

We don't pretend that our project is free of those concerns: we are working to train a model to identify sensitive information in pictures so that information can be blurred out. We built a model that can segment images into two categories: the background and a license plate. Of course, these kinds of systems can be transferred to *identify* license plates rather than blur them out, yet another computer vi-

[1] The code for this project can be found at this URL: https://github.com/parthsarin/cs231n-final-project

sion tool that would aid police [23, 11]. Our dataset comes from photos and videos taken of cars, but we don't know if the owners gave consent for this or where, when, or how the data were collected. More broadly, by engaging in this work with our GPUs and convolutional neural networks, our paper sits within a long line of work that lends credence to the very systems we seek to problematize. But, since this type of work is a requirement of CS 231N, we don't feel like we can avoid that. Instead, our goal, in engaging with critical theory, is to surface, unhide, and *unblur* the systems of power that surround computer vision.

2. Technical Literature Review

2.1. The Field of ALPR

Our project falls in the realm of Automated License Plate Recognition (ALPR), which has many applications in society including automatic toll collection, traffic law enforcement, parking lot access control, road traffic monitoring [6], as well as fraud prevention and security [18]. ALPR systems can be generalized as having four stages: 1) Image acquisition 2) License Plate Extraction 3) License Plate Segmentation and 4) Character Recognition [6, 18]. Our project focuses on stage (3), license plate extraction — but instead of extracting identifying license plate information from an image, we seek to cover it. Different techniques to extract a license plate from an image include: boundary/edge detection (since a license plate is normally rectangular), connected component analysis (detecting connectedness of pixels that have the same geometrical features of a license plate), texture features (changes in background color), color features (identifying noticeable colors in a region's license plate), and character features (looking for characters in an image).

2.2. CNN and YOLO Approaches for ALPR

CNN and YOLO are common approaches for ALPR systems [21, 19, 22], as they are used for object detection. Vig et al. write that the CNN-RNN model handles the various patterns (characters, sizes, fonts, colors, distortions, etc) in a two-stage method involving unconstrained localization of the license plate and utilizing deep learning's strengths in high-level semantic feature representations. LSTM (Long Short Term Memory) can also be added for optimal results. YOLO is a customized version of GoogLeNet, a CNN with 22 layers where $\text{loss} = l_{box} + l_{cls} + l_{obj}$. YOLO uses a single bounding box regression and IOU (Intersection Over Union) to predict the best bounding box, and has advantages of high accuracy and speed and is able to detect small-sized objects, which is important because license plates are often small in comparison to its background.

Concerned about privacy issues arising from ALPR systems, Shringarepure created a system to blur plates [19]. He uses a pre-trained YOLOv5 model, one of the fastest object detection algorithms, to detect the license plate. The images pass through 3 architectural blocks: CSPDarknet as a backbone layer for feature extraction, Panet as the neck to take input from the head and perform aggregation on the features to pass to the next stage for making predictions, and a Head layer to make predictions by generating a box. Finally, Gaussian blurring is used to blur the license plate. He cites the use of ALPR systems in Ireland's highways to scan cars going across the highway to collect tolls. The license plate information is then stored for two years or more. He writes that this is a privacy concern because storing such sensitive data could be used for malicious activity, which is why he developed a system that blurs license plates, which aligns with our goals as well. Inspired by his work, we chose to use Gaussian blurring for data augmentation and also compare our model to a state-of-the-art YOLOv5 model.

Figure 1: Examples of images in our data. Note how the license plate is a small but important part of the image

Additionally, low-data approaches have also been of interest because to make ALPR systems accurate in generalizable settings around the world: day/night, foggy weather, low quality cameras, different colors/shapes/languages on license plates. Lee et al propose using a shared encoder, detecting license plates and the surround

3. Dataset & Features

We are using Hugging Faces' Vehicle Registration Plates Dataset published by Roboflow [20]. It includes 8823 images: 6176 for training, 1765 for validation, and 882 for testing. Each image has a bounding box for the license plate, in COCO format. From the box, we created a mask where every pixel in the license plate bounding box is marked with a 1 and pixels not part of the license plate are marked with a 0. We experimented with the following data augmentation techniques: random photometric distortion, random horizontal flip, and Gaussian blur, which we dis-

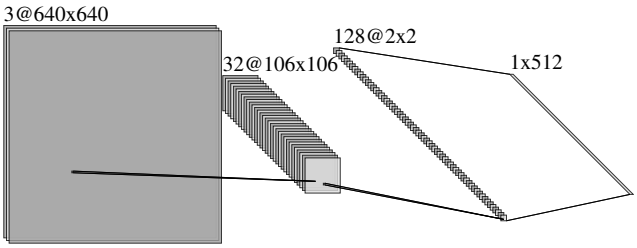

Figure 2: The baseline model uses a sequence of `conv-relu-pool` layers to generate a 512-dimensional embedding of an image, then adds a linear layer followed by softmax to predict probabilities for each pixel

cuss more in Section 4.2. Examples from our dataset can be found in 1.

4. Methods

We trained a model that was a sequence of three `conv-relu-pool` layers. Each convolution layer had a 5×5 filter and a stride length of 3. The pooling layers were all max-pool with a 2×2 kernel and a stride length of 2. A simplified version of the model is displayed in Figure 2. Then, the tensor was flattened into a 512-dimensional vector, passed through a linear layer, and then reshaped into a $2 \times 320 \times 320$ dimensional tensor. Finally, we applied softmax to the first dimension. We initialized the model using random weights sampled from an Xavier uniform distribution.

A version of this architecture is what we call the "Baseline" model. Later, we deepened the model, keeping the same general architecture, and added more channels to the convolution layers. We experimented with shallower and deeper fully-connected prediction layers after the `conv-relu-pool` sequence, but did not find much improvement on a validation set, and we report only on the models which were followed by a single linear and softmax layer.

We also trained a "FullConv" model which adds an additional `conv-relu-pool` sequence until the image is 1×1 with 512 channels. Then, it's flattened, passed through a linear layer, and reshaped, just like before.

We compare our results to a state-of-the-art pre-trained YOLOv5 model.

4.1. Loss function

First we used cross-entropy loss, averaged over all the pixels. But the resulting model predicted that every pixel was part of the background with probability 1. We suspect this is because the license plate generally makes up a small part of the image. A few samples from our data are shown in Figure 1 to illustrate this point. This means that the model trained using cross-entropy loss still achieved high accuracy because the license plate is so small compared to the rest of the image.

To fix this issue, we came up with a custom loss function that applied cross-entropy loss to the background and an exaggerated loss to the license plate. Fix an image x and let $y_{i,j}$ be the two-dimensional, one-hot vector that represents the correct class for pixel (i, j). Represent the model as f, so the model's predictions for the same pixel are the two-dimensional vector $f(x)_{i,j}$, whose entries sum to one. We also write $y_{i,j,0}$ to refer to the zeroth entry and $y_{i,j,1}$ to refer to the first (and similarly for the predictions). The loss function we used was:

$$\mathcal{L}(x, y) = \lambda \sum_{y_{i,j,1}=1} \log^2 f(x)_{i,j,1} - \sum_{y_{i,j,0}=1} \log f(x)_{i,j,0}.$$

We tested a few different values of the hyperparameter λ on a validation set and ultimately took $\lambda = 5$. With this configuration, we found that early checkpoints of the model generally predicted that most pixels are part of the license plate (the opposite problem from earlier) but, as training progressed, the model learned to differentiate better.

There was some instability with this training paradigm: occasionally, a gradient would propagate that greatly increased the loss. To address this, we implemented gradient clipping and, to encourage convergence, we used a cosine annealed learning rate schedule. We validated all of the hyperparameters for these approaches, especially the maximum absolute value for the gradient in each parameter, and settled on 0.1 for that value. We trained all models for 100 epochs, finding that the training loss seemed to converge around 80 to 90 epochs, depending on the model size.

We also tried a loss function which increased the gradient corresponding to mistakes on the license plate even more by replacing $\log^2 f(x)_{i,j,1}$ with $-\log^3 f(x)_{i,j,1}$ in the above equation, but we found this led to much slower training. Early checkpoints predicted that the entire image was part of the license plate, no matter how low we made λ and, once it started distinguishing (around 30 epochs into training) it was learning much more slowly. This loss function especially incurred a lot of instability perhaps because the amount of the gradient that was clipped was larger than with the log-squared loss, so the gradient that was propagated differed more substantially from the actual gradient than with log-squared loss.

4.2. Data augmentation

We tried a variety of data augmentation techniques during training and validated them on a validation set. The most basic was to crop and zoom from the original image and adjust the box accordingly. We trained using this paradigm up to the milestone report. After we had a model that was performing reasonably well, we tried a variety of additional augmentation techniques and conducted ablation

Table 1: Accuracy of Different Models

Model	CRP Depth	Channels	Loss	λ	Augmentation	$\mathcal{A}$	$\mathcal{A}_1$
YOLO (SOTA)	–	–	–	–	–	**0.98**	**0.99**
CNN (baseline)	3	512	XELoss	–	–	0.94	0
CNN	3	512	$\log^2$	3	–	0.91	0.86
CNN	3	512	$\log^2$	5	–	0.89	0.93
CNN	3	512	$\log^2$	8	–	0.83	0.97
CNN	3	512	$\log^3$	5	–	0.71	0.95
CNN	3	1024	$\log^2$	5	–	0.91	0.93
CNN	3	1024	$\log^2$	5	Gaussian blur	**0.91**	**0.96**
FullConv	4	512	$\log^2$	10	–	0.05	1

tests for each of them. We tried the augmentations one by one, training for 100 epochs, even if they hadn't converged at the end.

Random Photometric Distort. We applied this transformation, which jitters contrast, saturation, hue, brightness, and also randomly permutes channels, with probability 0.3. This augmentation reduced the accuracy of the model on the validation dataset, so we did not include it in the final model. We expect this happened because color is a valuable and relevant piece of information for a model predicting the location of a license plate—most of the license plates in the dataset are white-ish. Based on this test, we expect that the best performing model is using that information to identify the license plate.

Random Horizontal Flip. We also tried randomly flipping each image with probability 0.3 but this also reduced the model's accuracy on the test dataset. Applying similar reasoning, we hypothesize that the model might be relying on information about the text on the license plate that becomes harder to represent when we flip the license plate. For example, all of the license plates use Arabic numerals, so a model which has to identify the flipped numbers as well must be able to legibly interpret twice as many characters, and training it might take longer.

Gaussian Blur. We added a Gaussian blur with a 5×9 kernel and a standard deviation randomly chosen between 0.1 and 5. This augmentation improved the model's performance on the validation set, and we imagine it helped the model learn more stable representations related to shape and color rather than overfitting on the training set.

Because of these results, the final model we trained only used Gaussian blurring on the training data, in addition to the random crop, which resized the image to 320×320.

4.3. Hyperparameter tuning

Effective hyperparameter tuning was important for optimizing the performance of our training routine. Above, we've referenced a few places where we used hyperparameter tuning and, in this section, we explain in more detail how we arrived at the best hyperparameters for our final model. Specifically, we employed a combination of grid search and random search strategies to identify the best set of hyperparameters. The key hyperparameters tuned included the learning rate, weight decay, batch size, and the parameter λ used in our custom loss function.

The learning rate was tested over a logarithmic scale from 10^{-5} to 10^{-2}, weight decay values ranged from 10^{-6} to 10^{-3}. We also tried a variety of batch sizes, but generally just used the highest value that our GPU configuraiton could support. For the hyperparameter λ in our loss function, we tested values from 1 to 10. Each set of hyperparameters was validated on a hold-out validation set, and the performance was measured using accuracy and cross-entropy loss. We chose, by hand, the hyperparameters corresponding to the highest accuracy and cross-entropy loss, breaking ties to favor simplicity (e.g. lower values of λ).

The final model was trained with a learning rate of 10^{-3}, weight decay of 10^{-4}, and a batch size of 512. We also chose $\lambda = 5$ to balance the background and the license plate predictions.

4.4. Evaluation metrics

We used three metrics to evaluate the performance of each model: the custom loss function described earlier and the accuracy of pixel predictions, after rounding, restricted to the license plate and the background. That is, we define two accuracy metrics: the overall accuracy

$$\mathcal{A}(x, y) = \frac{1}{320^2} \sum_{i,j} \mathbb{1}\{\text{round}(f(x)_{i,j,0}) = y_{i,j,0}\},$$

and the on-plate accuracy,

$$\mathcal{A}_1(x, y) = \frac{1}{\sum_{i,j} y_{i,j,1}} \sum_{y_{i,j,1}=1} \mathbb{1}\{\text{round}(f(x)_{i,j,1}) = 1\}.$$

Here, $\mathbb{1}$ is the indicator function and, of course, "round" rounds the probability to the nearest integer, breaking at 0.5. We evaluated each of these metrics on the training and validation set at each epoch and on the test set every ten epochs.

5. Results & Discussion

We report the accuracy of each of the models we trained in Table 1. The acronym "CRP" in CRP Depth stands for "Conv, ReLU, Pool" and refers to the number of those blocks that we had in the model. The "Channels" column records the number of channels in the image after the final CRP block. The three loss functions referenced are XELoss which is the vanilla cross-entropy loss, $\log^2$ which corresponds to $\mathcal{L}(x, y)$, and $\log^3$ which is $\mathcal{L}$ with the $\log^2$ term replaced with $-\log^3$. Our best model had an on-plate accuracy $\mathcal{A}_1 = 0.96$, which is just slightly lower than YOLOv5's on-plate accuracy $\mathcal{A}_1 = 0.99$, but ours is trained from scratch on less than 1% of YOLOv5's data. We do not believe our model over-fitted to the training data, as our training loss and test loss curves are quite similar, as shown in Figure 5, which can be found in our paper's appendix.

5.1. Saliency Maps

For qualitative analysis of our results, we analyze saliency maps of success and failure examples to see which pixels were most influential in our model's output. In Figure 3, we depict 4 examples where our model successfully covered the license plates. The license plate's character features were highly influential for our model when detecting license plates, as the pixels outlining the structure of letters and numbers on the license plate are the brightest in our saliency map. For lower quality images where the license plate characters are not as clear such as the third row of Figure 3, the model placed more emphasis on pixels surrounding the license plate, such as the contract between a bright license late and a dark car. For the fourth row image, the pixels on the edge of the license plate (particularly the left edge) were more salient than the characters.

Judging from most of our saliency maps, color was not a salient feature for detecting a license plate. One reason for this is many of our license plates are white, and many images from our dataset are also taken from a parking garage exit, where there is a white access barrier arm that is very close in color to the license plate in many of our images (as shown in the first row of Figure 3). So, our model needed to distinguish from features other than color.

5.2. Failure Case Examples

Figure 4 shows examples where our model failed to fully detect and cover license plate information in an image. In most error cases, the model either 1) only covered part of the license plate or 2) did not cover the license plate at all.

From these failed examples, we experimented with the threshold for generating our red mask to cover up license plate information. Our model outputs a probability between 0 to 1 for each pixel, representing whether it believes the pixel is part of the background or not. If the probability is less than 0.1, the pixel is masked with red. Otherwise, it is not masked. We used a threshold of 0.1 because in most cases it provided a good balance of covering the license plate while not covering other parts of the vehicle/background that were not part of the license plate.

However, as the examples in Figure 4 show, 0.1 was not always a suitable threshold. Our first row shows a lower-quality image, where only part of the license plate was covered. With a slightly higher threshold of 0.2, the entire license plate was covered (as well as some of the background). For the examples on the second and third row, the license plates were yellow and red respectively, which contrast to the mostly white license plates in the dataset. In these cases, we needed to adjust our threshold to 0.4 or more to fully cover the license plate, and this introduced noise where parts of the car that were not part of the license plate were also covered.

6. Conclusion & Future Works

Even for a paper like this which is seemingly "less" problematic because it does not engage with questions of identity, the coloniality of computer vision is inescapable. As evidenced by our inability to trace the provenance of our dataset, we are reliant on the same systems of surveillance and monitoring that the field has benefitted and profited from.

One reason that working with small datasets and models is interesting is because it is the opposite paradigm of much modern AI development that tends to prioritize large datasets and big models with stacks of compute resources. In the end, by carefully choosing hyperparameters, model architectures, data augmentation, and a loss function, we were able to train a model on a small dataset that performs nearly as well as state of the art models trained on COCO. We hope to make a case for the legitimacy of similar approaches.

At the same time, small is not necessarily better. Others have used the phrases "well-defined" and "narrow" to characterize tasks that the community should focus on [8], but it's quite challenging to define what counts as a "narrow" task in AI development. In our example, the weights of the model we trained could be transferred to a system that, coupled with OCR, can be used to recognize license plates and aid police.

We conclude, then, with more questions than answers: Future work on this problem should seek to account for the social contexts of algorithmic systems, moving beyond accuracy and loss to ask questions about power and what kind of just and fair societies we want to build. Rather than being outside the realm of computer science, these questions are central and existential.

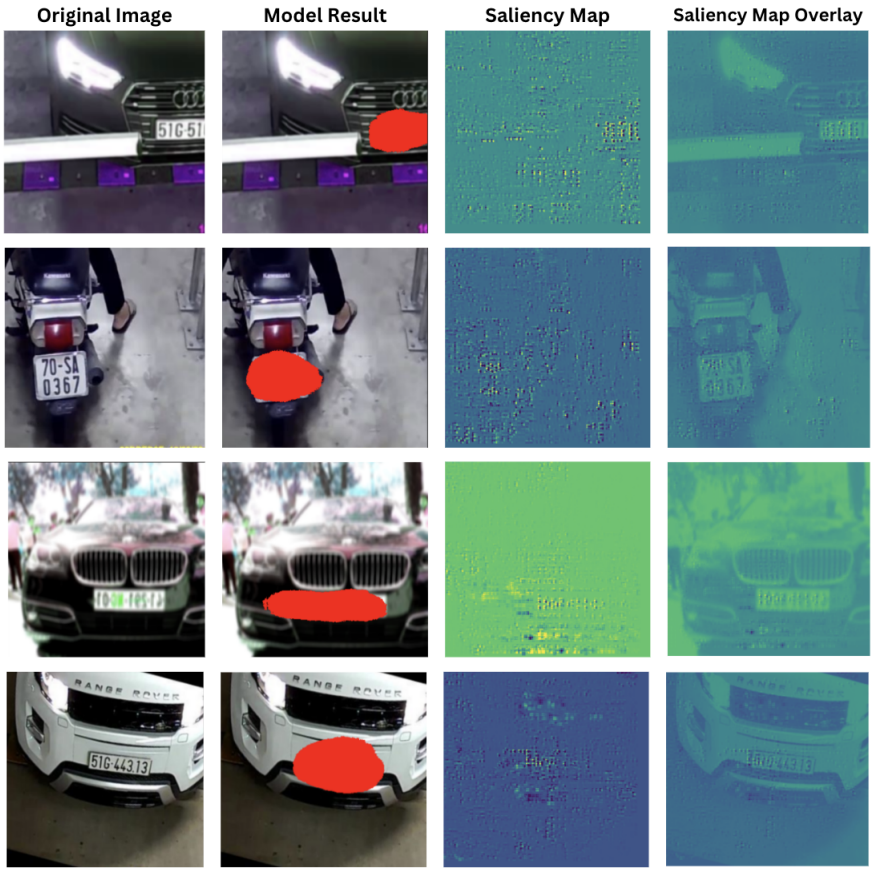

Figure 3: Successfully Covered License Plate Examples: model result and saliency maps

7. Contributions & Acknowledgements

Parth and Patricia collaborated together on all parts of the project, so there is a lot of overlap and it's difficult to separate out the components of the project into things they did individually. Here is our best approximation: Parth designed the model architecture, loss function, tuned the hyperparameters, and wrote Sections 1, 4, and 6, along with small pieces of the others. Patricia conducted the literature review, generated saliency maps for the results, so wrote Sections 2, 3, and 5.

We are tremendously grateful to all the CS 231N course staff for a lovely course with interesting lectures, guest speakers, and assignments. We learned so much and are proud to display some of those learnings here!

References

[1] S. Browne. *Dark matters: On the surveillance of blackness*. Duke University Press, 2015. 1

[2] J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018. 1

[3] R. Clarke. Information technology and dataveillance. *Communications of the ACM*, 31(5):498–512, 1988. 1

[4] N. De Lissovoy. Decoloniality as inversion: decentring the west in emancipatory theory and pedagogy. *Globalisation, Societies and Education*, 17(4):419–431, 2019. 1

[5] L. M. T. M. de Souza and A. P. M. Duboc. De-universalizing the decolonial: between parentheses and falling skies. *Gragoatá*, 26(56):876–911, 2021. 1

[6] S. Du, M. Ibrahim, M. Shehata, and W. Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on circuits and systems for video technology*, 23(2):311–325, 2012. 2

[7] V. Eubanks. *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press, 2018. 1

[8] T. Gebru and P. Torres. The TESCREAL bundle: Eugenics and the promise of utopia through artificial general intelligence. *First Monday*, 29(4), Apr. 2024. 5

[9] R. Grosfoguel. A decolonial approach to political-economy: Transmodernity, border thinking and global coloniality. *Kult*, 6(1):10–38, 2009. 1

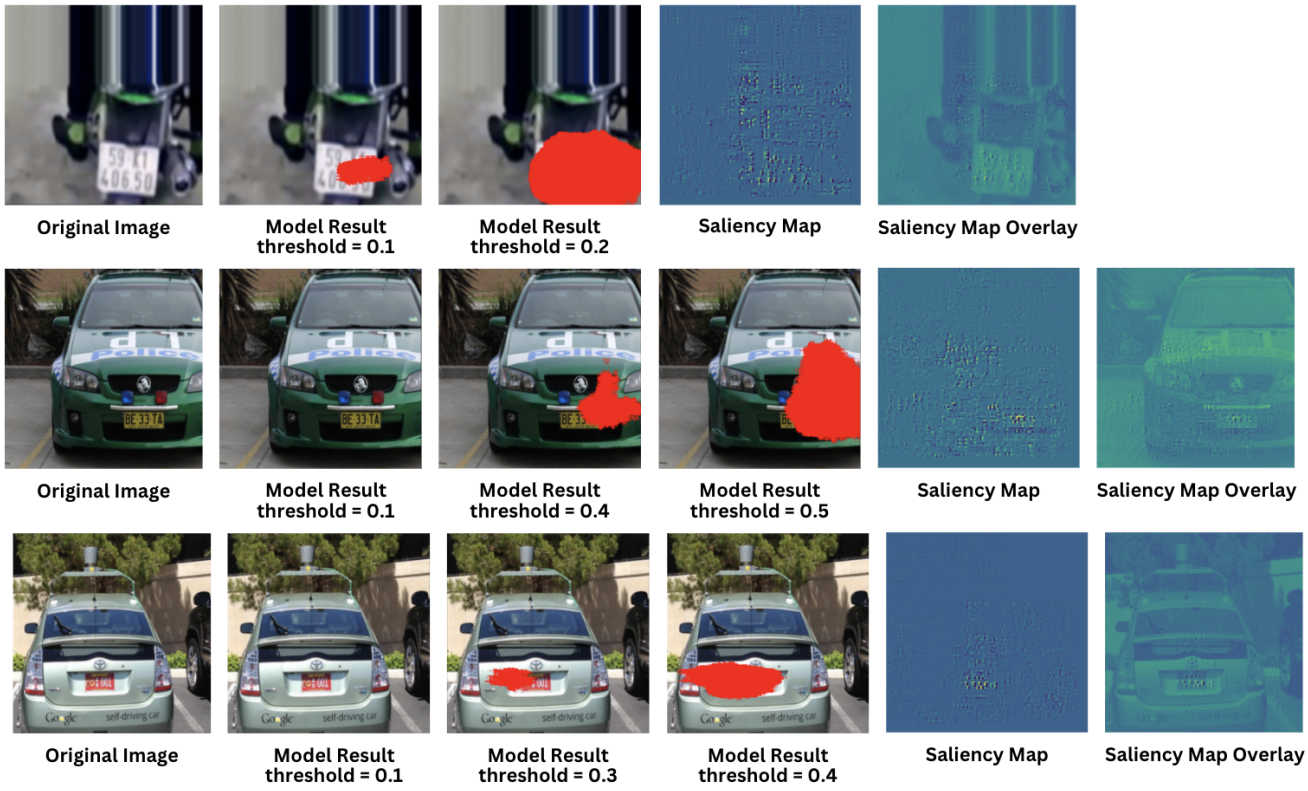

Figure 4: Examples of Errors from our best model when threshold was set to the default value of 0.1, along with our results when we changed the threshold

[10] R. Grosfoguel. Decolonizing Western Universalisms: Decolonial Pluri-versalism from Aime Cesaire to the Zapatistas 1. In *Towards a Just Curriculum Theory*, pages 147–164. Routledge, 2017. 1

[11] T. E. Hubbard. Automatic license plate recognition: an exciting new law enforcement tool with potentially scary consequences. *Syracuse Sci. & Tech. L. Rep.*, page 1, 2008. 2

[12] B. Jefferson. *Digitize and punish: Racial criminalization in the digital age*. U of Minnesota Press, 2020. 1

[13] D. Lyon. *Identifying citizens: ID cards as surveillance*. Polity, 2009. 1

[14] H. O'Brien. "The New Jim Code" – Ruha Benjamin on racial discrimination by algorithm, Sept. 2019. 1

[15] M. K. Scheuerman, A. Hanna, and E. Denton. Do Datasets Have Politics? Disciplinary Values in Computer Vision Dataset Development. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW2), oct 2021. 1

[16] M. Schlosberg and N. A. Ozer. Under the watchful eye: The proliferation of video surveillance systems in California. 2007. 1

[17] E. Selinger and D. Durant. Amazon's Ring: Surveillance as a Slippery Slope Service. *Science as culture*, 31(1):92–106, 2022. 1

[18] J. Shashirangana, H. Padmasiri, D. Meedeniya, and C. Perera. Automated license plate recognition: a survey on methods and techniques. *IEEE Access*, 9:11203–11225, 2020. 2

[19] D. V. Shringarpure. *Vehicle Number Plate Detection and Blurring using Deep Learning*. PhD thesis, Dublin, National College of Ireland, 2023. 2

[20] A. Startups. Vehicle registration plates dataset. https://universe.roboflow.com/augmented-startups/vehicle-registration-plates-trudk, Jun 2022. visited on 2024-05-14. 2

[21] S. Vig, A. Arora, and G. Arya. *Automated License Plate Detection and Recognition Using Deep Learning*, pages 419–431. 01 2023. 2

[22] L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang. A new cnn-based method for multi-directional car license plate detection. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):507–517, 2018. 2

[23] Z. Zeng, P. Gao, and S. Sun. License plate recognition system based on transfer learning. *Lecture Notes in Electrical Engineering*, 2018. 2

Appendix

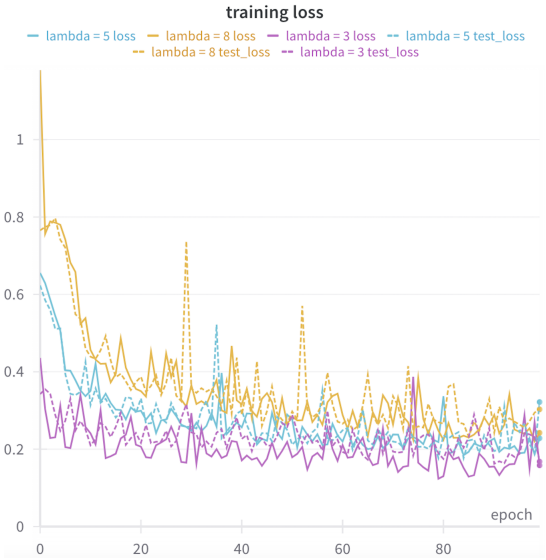

Figure 5: Loss curves for training (solid lines) and test (dashed lines) are very similar, sugesting no overfitting on the training data