House Number Recognition For Autonomous Delivery Robots

Ran Xin (1005875019)

***Abstract*— Autonomous last-mile delivery in residential communities remains challenging because GPS and detailed maps can be unreliable or costly to obtain, motivating the development of a perception-based localization model akin to human house-number reading. This project presents a lightweight, modular vision pipeline that detects house-number signs and transcribes multi-digit sequences in real time. The system integrates a YOLOv8n detector for localization and a CRNN sequence recognizer followed by temporal fusion for video streams. Experiments on the SVHN dataset show that the detector achieves 0.917 precision, 0.886 recall, and 0.931 mAP@50 on the test set, while the recognizer attains 4.46% character error rate and 92.58% sequence accuracy on test set. When integrated end-to-end on uncropped SVHN images, overall sequence accuracy is 82.1% with an estimated ~16M parameters, providing a practical accuracy–efficiency trade-off for embedded deployment.**

I. INTRODUCTION

In the modern society, delivery service has become an irreplaceable activity for the proper functioning of people's daily life. More merchants require delivery services to create end-to-end experiments to their customers. Businesses like food delivery and online shopping have grown largely after the pandemic, which has potentially changed people's shopping habits. Delivery networks that solely depend on human couriers are facing challenges in labor shortages, labor costs, and other workforce-related problems. In some cases, couriers have to drop the items in collective storage to save time, and customers will not be able to collect their delivery from the doorstep, leading to the 'last mile of delivery' problem. To solve the shortage of labor force, tech companies have been trying seeking help from autonomous robots. Over the past few years, modern autonomous robots and vehicles have seen great improvements in terms of accuracy, efficiency and security. Autonomous vehicles now can navigate through complex urban areas and deal with most traffic conditions with minimal human interaction. However, autonomous vehicles still perform poorly on atomizing the delivery—provide end-to-end delivery service. Specifically, most commercial autonomous delivery vehicles focus on dispensing goods from regional logistics station to sub-level receiving point, e.g. from courier station in campus to receiving point located at each dormitory building or from district station to office building receiving point. Moreover, these vehicles are often deployed in areas with high population density like downtown leaving other house-dominated communities unaddressed. This is primarily because before deployment to a certain area, high-accuracy GPS as well as detailed road maps need to be equipped and constructed, which are both resource and time consuming. This limitation prevents the development and expansion of autonomous delivery-to-doorstep solutions to normal communities consisting of houses. Therefore, it is crucial to devise a system that is capable of provide accurate localization function which can be seen as a complementary source of information besides GPS and SLAM.

When examining this demand, it is natural to think about how human delivery personnel complete their job— drive to a residential area and find the target delivery location by recognizing the house number with visual inspection. Different from GPS or SLAM information, this information provides a context of perception which is absolute with little uncertainty and is able to serve as a strong reassurance to GPS signal or a valuable replacement when GPS is interfered or blocked. In addition, with the advancement of object detection and sequence recognition neural network, there are enough tools that can be leveraged to solve the specific problem. The solution can be concisely formulated into 2 different stages: detection and recognition, where detection stage will be responsible for extracting region of interest (ROI) in the image feed and the recognition is specialized on making prediction according to the input region. This design is well-suited to a real-time environment by separating the detection phase from recognition phase, which allows the recognizer to focus on the pre-defined image region instead of searching exhaustedly for target in the complex and cluttered scene from the camera feed. Also, the constantly changing outdoor environment also requires a reliable image processing pipeline that can instruct the backend to put attention on the correct area, which is more like the attention mechanism concept in Transformers.

II. RELATED WORKS

A. CNN Based Method

Despite the fact that the proposed solution is an end-to-end pipeline, which consists of detection and recognition, the detection itself in the context only serves as a tracking feature that is used to accommodate for the complex scene fed in from real camera footage. The task can still be safely distilled or generalized into a full-sequence SVHN house number recognition task while not explicitly considering detection. There is previous work that deals with the sequence recognition problem only using variants of CNN. In 2013, Goodfellow et al. proposed a method of recognizing multi-digit numbers from street-view images.[1] The recognition task is done without explicitly conducting pre-segmentation for each digit. Instead, they implemented a deep CNN that jointly handles localization, segmentation and recognition of multi-digit numbers. This can be seen as a milestone of multi-number sequence recognition since instead of separating the entire recognition problem into different phases, it provides an end-to-end solution that combines all the tasks. They evaluated SVHN and a much larger internal Street View dataset and reported an accuracy of over 96% on complete street numbers on SVHN.

However, in order to get satisfactory accuracy, a deep CNN network with multiple hidden layers and a large dataset is required to prevent overfitting. There is also another work from Sermanet et al. using a carefully tuned CNN network with features like shared weights, careful preprocessing, and multi-stage training to conduct robust digit-level recognition. It should be noted that this work is targeted at single-digit recognition, but the insight when conducting the experiment

will be helpful in multi-digit and scene-text recognition tasks.[2] To increase the variety of house number images, Zhong et al., 2019 targets SVHN and builds a DCGAN-based system for recognizing complete house numbers without splitting digits.[3] In this context, DCGAN is used to generate additional realistic digit images and improve robustness. They reported to achieve around 95.56% accuracy on complete numbers on SVHN.

B. *Combining RNN with CNN*

For the previous solutions methods that merely rely on CNN, special considerations must be made when designing the loss function that is to be optimized. Limited by the structure of CNN, only relying on CNN to conduct house number recognition can be difficult for tuning and the accuracy could also be sub-ideal. Aside from CNN, in terms of a varied length sequence prediction task, the recurrent neural network (RNN) is suitable with its ability to handle sequence objects and producing meaningful embeddings for future predictions.[4] When considering making predictions on sequence objects within images, it is natural to make use of the model that combines CNN and RNN. In this setting, CNN will be responsible for extracting feature maps from designated areas and feed the map to subsequent RNN network for sequence recognition. In 2015, Baoguang Shi et al. proposed a trainable neural network (so-called CRNN) for image-based sequence recognition which can be applied in scene text recognition[5]. This is an end-to-end solution in recognizing sequence-like objects from images. This work is specially targeted at scene text recognition, but it can also be easily extended to house number recognition, which shares a lot of similarities with text recognition. This work provides a solid foundation for tackling sequence object recognition from images and it can be treated as a meaningful template which house number recognition tasks can use.

III. METHODOLOGY

As discussed above, the model will essentially have 2 different modules: detector and recognizer. The detector will be responsible for generating regional proposals. It should be noted that the detector has to complete object detection of complicated scenes with changing view angles, varying light, and occasional occlusions and still maintain low complexity for the sake of inference latency and computational resources limitations on low power devices. This naturally brings us to the well-known YOLO network, which has prescribed structure that is reliable, robust and easy to train. The YOLO network is also renowned for its abundant varieties of model sizes that could satisfy the need for different purposes. Specifically, the nano version of the network is able to achieve satisfactory performance with relatively low number of parameters, which makes it a perfect choice for our context.[6] In addition, the localization provided by YOLO also makes it convenient for the extraction of area of interest for future processing. Instead of tight-fitted bounding box, the YOLO network will be configured so that the output bounding box will create a pad around the target sign to accommodate and make space for future recognition. The recognizer will leverage the benefit of CRNN model and focus on the selected area from the YOLO output. The image is first passed through a CNN model used for feature extraction; the height dimension is concatenated along the channel dimension to form a single column in a candidate sequence. The number of vectors in a sequence are determined by the width of feature map after CNN, and before feeding in the CRNN, all cropping generated from YOLO region proposals are padded to max width so that the width of CRNN raw input is the same across different samples. One variant in the RNN family—long short term memory (LSTM) network is chosen as the recognizer with a bi-directional configuration. A CTC loss function designed for sequence recognition task is used during training. CTC loss defines a probability over all valid alignments (path) that could produce the target sequence, where each path probability is a product of frame SoftMax probabilities.[7] The formulation of CTC loss is given in (1):

$$P(y|x) = \sum_{\pi:B(\pi)=y} \prod_{t=1}^{T} p_t(\pi_t), \qquad (1)$$

where $P(y|x)$ stands for the probability that the model outputs the target sequence y given the ground truth x, and path π is a predicted sequence of labels generated from model output with length T of labels. We also define a deterministic function $B(\pi)$ that removes consecutive duplicates and

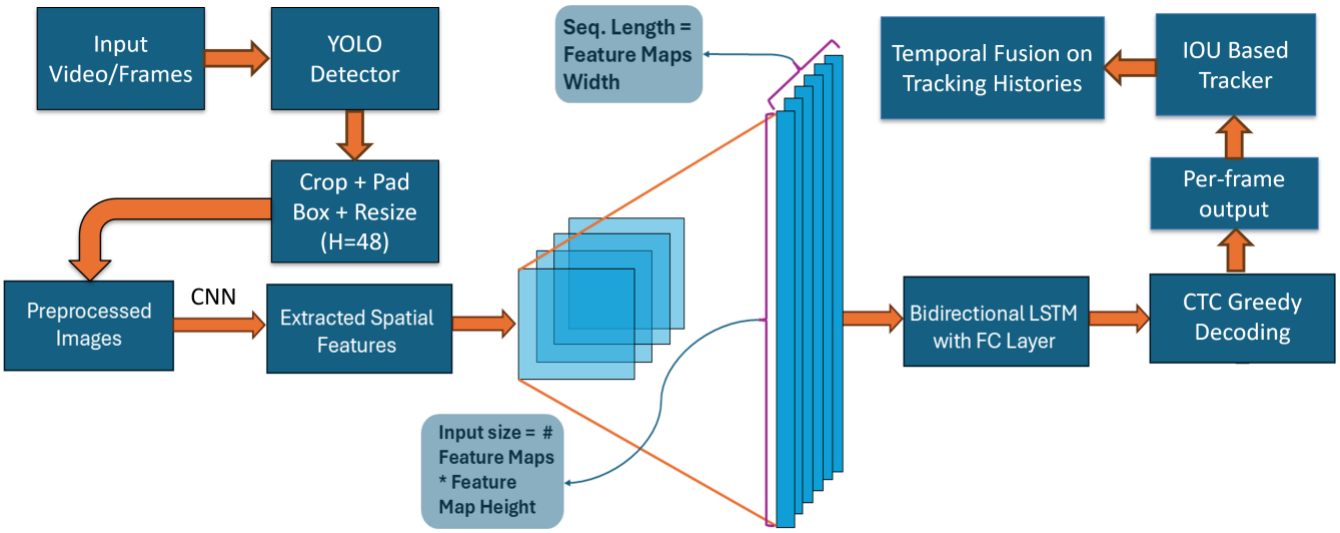

Figure 1. Overview of completed pipeline during inference/deployment, which merges detector and recognizer

blanks, and $p_t(\pi_t)$ signifies the per-frame network output softmax probability. Note that a key assumption is necessary to justify this formulation: CTC assumes each time-step prediction is conditionally independent given the input sequence. Thus, this brings us to the fact that:

$$P(\pi|x) = \prod_{t=1}^{T} p_t(\pi_t), \quad (2)$$

where $P(\pi|x)$ represents the probability of a specific that can be mapped to the ground truth sequence using the

deterministic function $B(\pi)$, and $\prod_{t=1}^{T} p_t(\pi_t)$ results in this probability of the entire sequence by multiplying the per-frame (i.e. per time-step) softmax probability from model output.

This configuration enables normal back propagation of the neural network and allows for proper training. During evaluation or inference, the decoding is done using greedy decode schemes, which selects the target with highest log probability at each timestep, and maps the raw path to actual sequence using the same logic in CTC loss. For video stream as input, a temporal fusion technique is implemented to collect the predictions on multiple frames with weighted majority vote to yield a final prediction with regard to the input in a past interval. The complete pipeline is shown in Fig. 1.

IV. Experiments Configuration

A. Datasets

In this project, the primary source of training data comes from the Street View House Number (SVHN) Dataset, which is a comprehensive house number database created using real footage from Google Street View images. Compared to traditional MNIST dataset, which is composed of synthetic digits, SVHN dataset includes real-life background, which excels in complexity and variety in terms of lighting conditions, view angle, and sharpness.[8] More importantly, the real scene information and pattern contained in background pixels is crucial for the purpose of real-life deployment. Although the model is designed as an end-to-end pipeline functionally, it has to be noted that there is no hard coupling in terms of structure between the detector and recognizer, which means it is feasible to train detector and recognizer separately before merging them for deployment. In addition, the concept of separate training also eliminates the need of creating multi-head mechanism with shared backbone, which greatly simplifies both model architecture and data preparation. As a result, the datasets are prepared for the YOLOv8n detector and CRNN recognizer separately which aligns to the training plan.

The raw dataset is retrieved from Stanford website in Format 1-- full sequence of images. There are over 30k raw images in the train set, and over 13k images in the test set. The images in the train folder are accessed during training, and the test images are reserved for final test. The images contain the full house number with labels information stored in .mat file. For each image, the struct is formatted with both bounding box and digit label information, and several struct can exist in the image. Noted that the bounding box information comes with the coordinates of left top corner point as well the width and height of the bounding box, which sums up to 4 entries. On the other hand, the label information contains the ground truth digit inside the corresponding box, and there are 10 classes for the label with digit from 1 to 9 having label as themselves and digit 0 having label 10. Therefore, it can be easily noticed that for each struct in the image, there are 5 entries that describe the bounding box position and sizes as well as the ground truth label for classification task. Since we are aiming at the detection and recognition of the entire sequence, the character level bounding box for every digit in every image needs to be first merged to a sign level bounding box, which encapsulates the entire house number sign in one image. At the same time, the bounding box is converted to the YOLO format which uses 4 coordinates to represent the top left and bottom right corner of bounding box. The top left coordinates of new sign-level bounding box are determined by taking the minimum coordinates in x and y direction, whereas the bottom right point is decided by either maximum sum of top left point x coordinate and width or the sum of y-coordinate and height. The yielded bounding box coordinates are then padded by a fixed fraction in order to create enough buffer space for the CRNN recognizer. As a result, the detector will be trained to learn to output the house number region with a pre-defined cropping. The label information is determined by the purpose of training. For detector training, only the sign level bounding box information comprised of 4 integers is provided, whereas the sequence ground truth is provided by combining the character-level label following a left-to-right order. The detector will directly take in the raw image from SVHN dataset and generate a bounding box around the desired target, while the recognizer will take in the cropping formulated by ground truth bounding box as an input and tries to decode the right sequence within the region. In order to prepare for the training data for CRNN detector, we apply the same cropping and padding logic to the raw dataset. Different from the detector, we actually crop the image using the merged sign-level house number bounding box and treat this as the input to CRNN while the label is the ground truth sequence. It is also worth mentioning that a pre-trained stage is also implemented when working with the CRNN recognizer to help the model to achieve a faster convergence rate and better performance on the main training step. During the pre-train phase, instead of working on actual house number in real scene, we train the model on synthetic digits with varying background noise so that the model can have a preliminary impression on the features of digits. The synthetic images with digit sequence are generated using the open source TextRecognitionDataGenerator (TRDG) project available on GitHub.[9] 50k synthetic images with random number sequence of different length are generated for training and 5k images are used for validation. By varying intensity of background noises, occlusion level, rotation angle, font style, and font size, it allows the model to have an initial grasp on a comprehensive pattern of digits, which fully prepares the model for the main training phase. Some sample images of the generated synthetic digit sequences using TRDG is shown in Fig. 2.

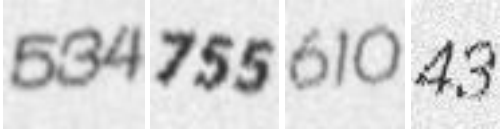

Figure 2. Synthetic digit sequence with variations

B. CRNN Structure

The convolutional backbone of the CRNN network is designed as a customization VGG style feature extractor with 6 convolutional layers with 2 of them focusing on the "isometric" processing.[10] This specific set-up is used to refine features and integrate context by capturing the pattern across pixels and thus increases the effective receptive field of the network. After the convolutional network, 2 layers of bi-directional LSTM with a hidden size of 256 is created and followed by a single fully connected layer. To reduce overfitting, a dropout ratio of 0.3 is applied on both the bi-LSTM layer and the fully connected multi-perceptron.

C. Training

To improve the performance of the model, the detection and recognition components are trained in different stages, rather than in a single monolithic run. The YOLO detector is optimized using a three-phase fine-tuning schedule, and each subsequent schedule is started from the best checkpoint saved from the last training phase. For the YOLO detector, we started from ImageNet/COCO-pretrained weights and fine-tune the model so that it adapts itself on SVHN house-number signs. In Phase 1, the model is trained with a learning rate of 0.002 which is relatively higher and the backbone is frozen by setting "freeze" parameter to 10. This setting only leaves the neck and detection head as the trainable section, which can be seen as warm-up phase of the detection head. This phase keeps the feature extraction core untouched since the network has already been well-tuned, so it performs ideally on extracting meaningful features from images. Training on detection head specifically lets the model quickly turn its focus on the desired category and learns the object probability distribution in high dimensional space. The first phase is trained for 15 epochs with a batch size of 32. In Phase 2, all the layers are unfrozen so that both the backbone and detection head are able to learn the specific features of the training data. The learning rate is reduced to 0.001, and lighter augmentation is applied through setting mosaic=0.8, perspective=0.001, translate=0.1, and scale=0.5. Here is the main learning stage of the model and the focus shifts to teaching crucial and unique features of the target object. At the same time, this stage also takes the responsibility of stabilizing the detector, improving box regression accuracy, and refining class confidence scores on SVHN's representative images. In order to give the model enough time to adapt for the current target, it is trained for 25 epochs with a batch size of 32. Finally, in Phase 3, the model is trained for 15 epochs, and the learning rate is further decreased to 0.0005, which corresponds to a short, low-learning-rate fine tuning. Lighter augmentation is also implemented with hsv_h=0.0, hsv_s=0.1, hsv_v=0.1, perspective=0.0005, translate=0.05, and scale=0.4. Noted that mosaic is set to 0 because at the final stage of training, it is desirable to focus on the performance of each individual target by avoid generating predictions for multiple targets. Such setting is beneficial for a clean and refined learning behavior of the model. While trying to force the model learn to adapt more variations of the images, this last phase also behaves like a "polishing" step that pushes the model toward its optimum on the validation set and prevent overfitting from happening. By using multiple phases instead of a single long run, it is easier to monitor metrics like mAP / F1 at each stage and if later training becomes unstable, we can always fall back to the last checkpoint with optimal behavior.

On the other hand, the CRNN recognizer is trained in two stages: a pre-training phase on synthetic digit strings followed by a main training phase on real SVHN crops. This staged strategy is chosen to balance convergence speed, robustness, and overfitting control under limited compute and data. In the pre-training stage, the CRNN is trained on a dataset containing 50k images of synthetic digit strings generated using TRDG. In this stage, the model is trained with a batch size of 64 and learning rate of 1e-3 for 8 epochs. This gives the network a broad and general prior knowledge over the basic information of digit including shapes, spacing, and common nuisances. As a result, when entering the main training stage, the model will already be equipped with a preliminary concept about the digit, which greatly lowers the pressure of learning in the main learning phase. In the main training stage, we then fine-tune the same CRNN on crops extracted from SVHN using ground-truth union boxes over the digits. These crops are rectified and resized to a fixed height, and the model is trained using CTC loss against the true house-number strings. During this stage, the model is trained for 40 epochs with the same batch size as the pre-train phase but at a much smaller learning rate of 2e-4. Also, noted that in order to improve the training stability and final performance of the network, a learning rate scheduler following the "One Cycle" policy is applied, which changes learning rate and momentum dynamically to fit the different need in different phase of a training.[11] The SVHN fine-tuning stage adapts the learnt features from pre-train phase according to the real camera style, color distribution, and noise characteristics of the dataset. This two-step procedure effectively results in a better generalization ability and a faster convergence rate compared to a single training on SVHN alone. Such training scheme is also helpful in preventing overfitting of a relatively powerful sequence recognition network on a modest-size dataset.

V. Results

A. YOLO

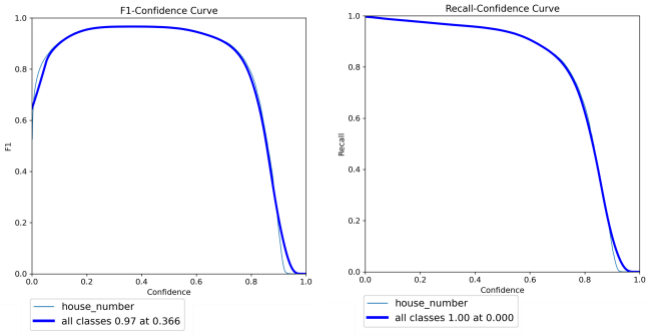

Figure 3. F1 score-Confidence curve and recall-confidence curve for YOLO

After completing the training of YOLO, the training metrics automatically generated from the model shows a stable convergence on both detection and localization. The bounding box precision, recall, mAP@50, and mAP@50-95 results for train, validation, and test set is shown in Table I. After fine-tuning across three phases, YOLO achieved strong performance: about 91.7 percent precision and 88.6 percent recall on test set. The mAP@50 reached 0.931, demonstrating

reliable localization capabilities. The recall on test set is slightly lower, meaning that it struggles on catching house number objects occasionally. Also, the high mAP50 metrics signifies that the model performs ideally in normal IoU threshold; however, the score drops significantly when raising the bar to mAP50-95. This phenomenon could be resulted from the padding around the actual house number sign when constructing labels to accommodate for recognizer, and thus may hurt performance on precise bounding box localization. In order to select the proper confidence threshold for the inference stage, we further examine the training result with specific focus on the F1 score and the recall-confidence curve, which are shown in Fig. 3. From the F1-confidence curve, it can be seen that the harmonic mean between recall and precision reaches its maximum value at a confidence value of 0.366, which signifies that the model has achieved a robust balanced performance. The recall of the model measures the scenarios where the model fails to capture the house number sign, which will significantly harm the performance of model. Thus, it is preferable to favor more on the recall when selecting the confidence threshold while keeping the deviation of F1 score from its maximum at its lowest. It can be seen that the recall-confidence curve reaches 1 when confidence threshold is set to 0, and the recall slowly decreases before seeing a quick jump when the confidence reaches 0.4. With careful inspection on the F1 curve, it remains at a high value plateau between 0.2 and 0.5 with a maximum at 0.366. Following the previous logic on the selection of confidence threshold, a confidence of 0.25 is selected to ensure that the recall is maximized as much as possible and the F1 score can be assured to only decrease by few percentages from its maximum.

TABLE I: YOLO metrics after training

Dataset	Box Precision	Box Recall	mAP@50	mAP@50-90
Val	0.971	0.963	0.988	0.734
Test	0.917	0.886	0.931	0.576
	Box Loss	**cls Loss**	**dfl Loss**	-
Train	0.962	0.385	1.221	-

B. CRNN Results

The training/validation loss curve for the training of CRNN model is shown in Fig. 4. Both training and validation loss steadily decrease throughout the training and reach a stable plateau at the final stage around 0.18, which demonstrates smooth and successful training. This also proves the necessity of the implementation of pre-train, which gives the model is impactful warm-up before entering the main fine-tuning stage. For the evaluation on performance of CRNN, we introduce two metrics: character error rate (CER) and sequence accuracy. The CER is calculated using the concept of Levenstein distance[xx], which measures the minimum number of edits required to transform one string into another. In this context, allowed operations in the edit include insertion, deletion, and substitution, which are self-explanatory. If a ground truth is "385" and the predicted string "35", the required operation for chaining the prediction to ground truth is to insert an "8" between 3 and 5, which is counted as 1 in edit and Levenstein distance. The CER is then formulated by taking the ratio of the Levenshtein distance and the length of ground truth sequence [12]. The CER is reported

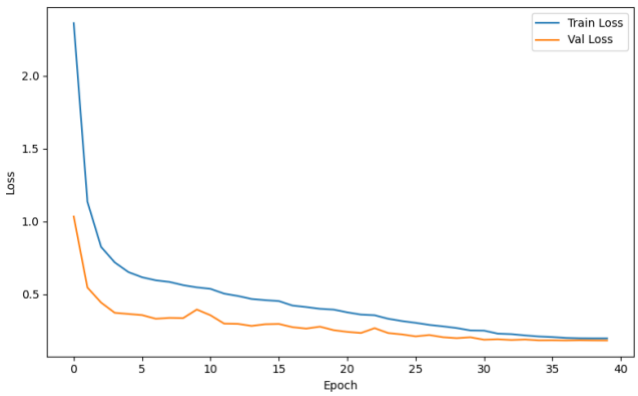

Figure 4. CRNN train set and validation set loss curve with respect to training epochs

against the entire dataset rather than the single sequence, which means the total Levenstein distance of the dataset is divided by the total length of ground truth sequence to yield CER. The sequency accuracy can be considered as a more stricter metric compared to CER: it is more similar to a traditional accuracy metric on classification task, which only classify the result as right when it perfectly matches the ground truth. The performance of CER and sequence accuracy are shown in Table II. It can be seen that both the CER and sequency accuracy perform ideally on both test and validation test with a minimum deterioration on generalization capabilities.

TABLE II. Performance Metrics of CRNN on train, validation, and test set

	Train	Validation	Test
CER (%)	2.74	4.70	4.46
Sequence Accuracy(%)	94.91	91.62	92.58

The performance of the trained CRNN network is compared with similar work that focuses on number sequence recognition within images and is shown in Table III. In 2016, Guo et al. reported an accuracy of 91% on the full-number SVHN dataset (unsegmented house-number images) in the work "Convolutional Recurrent Neural Network for Scene Text Recognition", which is comparable to the customized CRNN network we proposed[13]. In 2013, Goodfellow et al. proposed a pure deep CNN without RNN reports a 96.03% sequence transcription accuracy on the public SVHN full-number dataset.[1] This performance is more competitive compared to our result, especially when no RNN structure is utilized to help with sequence recognition. In a recent research by Li et al. on channel-attention CRNN (CACRNN) using SVHN dataset, they report an accuracy of 82.04% using plain CRNN, and an accuracy of 89.63% using ResNet+LSTM+SE block[14]. Overall, our results on the recognition task demonstrates strong performance when compared to model with similar purposes. It has to be noted that, nevertheless, the superior accuracy could be brought by the fact that the training and test images are pre-cropped using the union ground truth bounding box, which greatly reduces the pressure on object detection and localization.

TABLE III. Performance comparison of CRNN only to other models with similar purposes

Model	Result
CNN (framewise)	23%
GMM–HMM	56%
Hybrid CNN–HMM	81%
Plain CRNN	82.04%
CRNN from Guo et al., 2016	91%
Our Model (Recognizer Only)	**92.58%**
Goodfellow et al.'s deep CNN	96.03%

C. *Full Pipeline Performance & Model Complexity*

After evaluating the performance of the YOLO detector and CRNN recognizer separately, it is meaningful to test the overall performance of the end-to-end pipeline by combing the detection and recognition module with simple rectification. Instead of testing on the recognizer that treats pre-cropped images as input, the test is conducted using uncropped SVHN images with full house number sequence. The detector will take in the raw images and output the bounding box around the target region, and the recognizer will then make predictions to the number sequence inside the ROI. The full pipeline is evaluated on the test set with individual images as input. The results are shown in Table IV with comparisons included for 2 other state of the art (SOTA) models that also combine both detection and recognition tasks in a single shot. In addition to the model using deep CNN from Goodfellow et al., the model that takes the leverage of recurrent attention structure (DRAM) is also included for comparison.[15] The model complexity is also measured by approximating the number of trainable parameters in the model. It can be noticed that after integrating the detector and recognizer to form a complete pipeline for implementation, the sequence accuracy drops from 92.58% (only evaluated using CRNN) to 82.1%, which is reasonable and expected since both the error and uncertainty are coupled, hurting the entire performance of the model. However, aside from the prediction accuracy, the efficiency and complexity of the model is also of significance in the context of this project since the model is targeted at real-time deployment on resource-limited platforms like autonomous robots. It is easily noticed that compared to our model with around 16 million parameters, the other 2 competitors require 75%-212% more parameters to achieve a better result. Subsequently, there are always trade-offs between the accuracy and complexity of the deep learning model, and it is believed that our model reaches a balance between these 2 factors in the given context.

TABLE IV. Performance and complexity of comparable SOTA models

Model	Result	Number of Parameters
Deep CNN (Goodfellow et al., 2013)	96%	~50M
Our Model (Full)	**82.1%**	**~16M**
Ba et al. (DRAM visual attention)	96.1%	~28M

VI. CONCLUSION

This work presents an end-to-end perception pipeline for house-number recognition aimed at supporting autonomous last-mile delivery in community environments where GPS and detailed maps may be unreliable or unavailable. In the proposed solution scheme, the model consists of a YOLOv8n-based detector that balances between performance and efficiency followed by a customized CRNN recognizer trained using CTC loss. The detector and recognizer are connected through a simple rectification module, and the per-frame output is processed through temporal fusion module for video feed. This modular design allows the detector to robustly localize house-number signs in street scenes while the recognizer focuses on sequence interpretation within the proposed region. Experimental results on the SVHN dataset demonstrate that both components achieve strong performance for individual assessment: the YOLO detector reaches a mAP@50 of 0.931 with 0.917 precision and 0.886 recall on test set, while the CRNN recognizer achieves a CER of 4.46% and sequence accuracy of 92.58% during testing. When the detector and recognizer are integrated into a full pipeline and run on raw SVHN images, the overall sequence accuracy comes at 82.1%. Despite lower than some deep CNN and attention-based SOTA models, the proposed system reaches such performance with only roughly 16M parameters—which is substantially fewer than comparable models for similar work. Such setting resides in a practical balance between accuracy and model complexity for embedded deployment. At the same time, the evaluation process also presents several limitations and yields opportunities for improvement. The performance drop from recognizer-only to full-pipeline evaluation reveals sensitivity to imperfect detections and sign-level padding, while the sharp decline in mAP at higher IoU thresholds suggests chances of further improve on bounding box prediction quality. Future work could start from formulating a tighter integration between detection and recognition by constructing shared backbones between YOLO and CRNN. It is also feasible to improve performance by implementing a more advanced rectification technique between detector and recognizer like spatial transformation networks (STN) that learns the optimal transformation or rectification between 2 modules.[16] By modifying the CRNN structure to a more complex attention-enhanced or transformer-based sequence model, it is expected to see improvements on model robustness and accuracy. At last, by extending the dataset beyond SVHN via domain adaptation to street-view data collected in real life, it would better bridge the gap between the theoretical performance and practical autonomous operation in everyday community environments.

REFERENCES

[1] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," in *Proc. Int. Conf. Learn. Representations (ICLR)*, Scottsdale, AZ, USA, 2013.

[2] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," *arXiv preprint arXiv:1204.3968*, Apr. 2012.

[3] J. Zhong, J. Gao, R. Chen, and J. Li, "Digital recognition of Street View House numbers based on DCGAN," *Proceedings of the 2nd International Conference on Image and Graphics Processing*, pp. 19–22, Feb. 2019. doi:10.1145/3313950.3313963

[4] I. D. Mienye, T. G. Swart, and G. Obaido, "Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications," *Information*, vol. 15, no. 9, Art. no. 517, Aug. 2024, doi: 10.3390/info15090517.

[5] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *arXiv preprint arXiv:1507.05717*, Jul. 2015.

[6] M. Yaseen, "What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector," *arXiv preprint arXiv:2408.15857*, Aug. 2024, doi:10.48550/arXiv.2408.15857.

[7] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, Pittsburgh, PA, USA, 2006, pp. 369–376, doi: 10.1145/1143844.1143891.

[8] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[9] E. Belval, "TextRecognitionDataGenerator," ver. 1.6.0, GitHub repository, May 9, 2020. [Online]. Available: https://github.com/Belval/TextRecognitionDataGenerator. [Accessed: Dec. 12, 2025].

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

[11] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1—learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[12] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, Feb. 1966.

[13] Q. Guo, D. Tu, G. Li, and J. Lei, "Memory Matters: Convolutional Recurrent Neural Network for Scene Text Recognition," *arXiv preprint arXiv:1601.01100*, Jan. 2016, doi: 10.48550/arXiv.1601.01100.

[14] J. Li, C. Wang, and R. Cai, "Channel attention convolutional recurrent neural network on street view symbol recognition," *Highlights in Science, Engineering and Technology*, vol. 9, pp. 390–397, Sep. 2022, doi: 10.54097/hset.v9i.1869.

[15] J. L. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," in *Proc. Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, May 2015.

[16] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial Transformer Networks," in *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.