CV-Based Smart Parking Monitoring System

Zixuan Zhang (zz2777)
Haoran Zhu (hz2712)

Motivations

Parking management has been an headache for cities across the world as urbanization picks up pace and car ownership goes up. In cities like New York, parking enforcement is mostly carried out by patrol officers who drive around and hand out tickets to violators. This process is, first and foremost, labour intensive as officers need to check every single car in every parking space in their duty areas. Another issue with the current enforcement diagram is its low effective coverage. There's no effective enforcement at night since no one is on duty or between two sweeps as it takes hours two check all parking spaces in one round.

We acknowledge that there are systems with automated functionalities. A common application can be seen in shopping malls where a display board shows the remaining capacity of each garage level. However, such systems require expensive investment as each parking space must be accompanied with one sensor. The functionality is also limited as they only offer empty space detection, but omit important features such as no-parking zone monitoring. In summary, the current parking management system is labour intensive, inflexible and offers low coverage. We believe that a more cost-efficient and versatile solution is needed to address the problem of parking management.

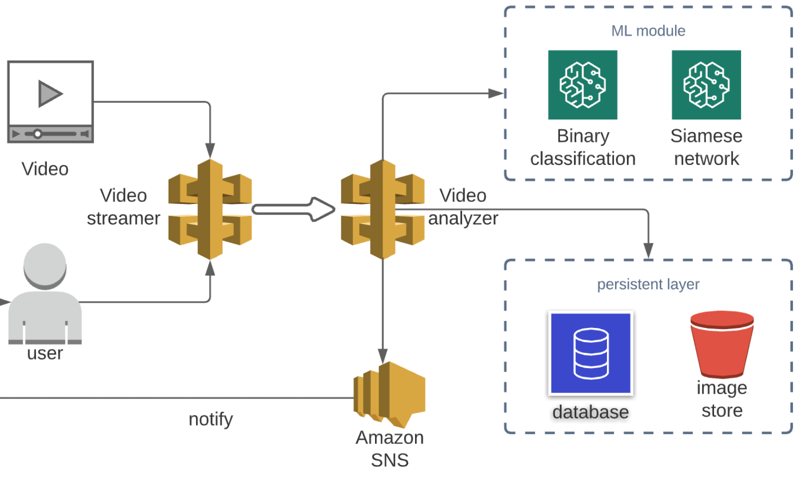
Objectives

We expect our solution to meet all three primary objectives - versatility, cost effectiveness, and automation. Versatility means the system must go beyond simple empty space detection and implements more complex features such as no-parking zone violation detection and overtime violation detection (time measurement). Unlike current systems that require a large number of sensors, our solution should use much fewer sensors and reduce cost per parking space. Last but not least, humans must be kept in the loop but with minimum intervention needed.

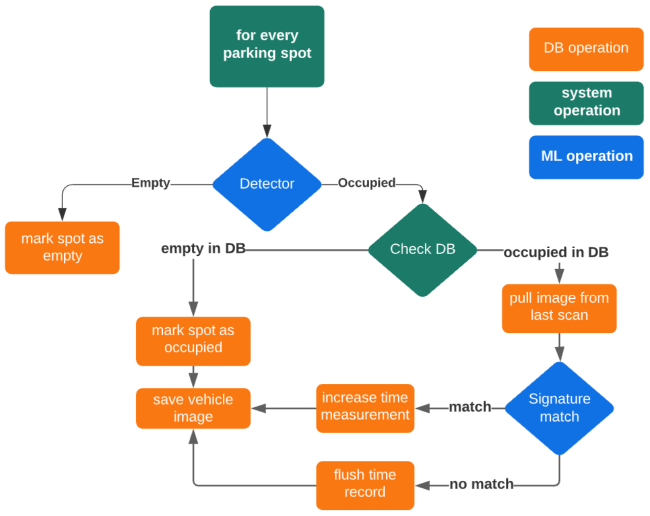
Proposal

After careful consideration, camera sensors are picked for their rich information. A properly mounted camera can cover 10+ spaces on the street or ~5 spots indoors, which yields a much lower per-space cost in comparison to ultrasonic sensors. More importantly, raw images captured by camera empowers the backend system to do CV-based analysis such as no-parking violation. This is the unique advantage of cameras since it is impossible to install other sensors to cover places where people cannot park.

Below is the high level system architecture of our project. We utilize cloud services like AWS to make the system reliable and cheap. Because of the nature of time measurement, stateless ML models must be supported by a persistence layer such as databases.

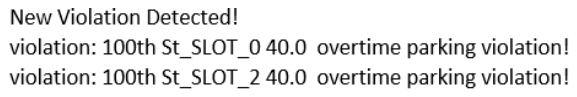


To monitor the desired parking lot, the user just needs to provide annotations containing the allowed parking spaces with the maximum parking time, the prohibited parking area (such as emergency exit) and ID for each spot. Note that the annotation is provided once at setup time. The system will regularly run CV algorithms to determine which spots are empty, a feature essential for remaining capacity estimation. For designated parking spaces, the backend system will do time measurement and make sure vehicles do not stay over allowed duration. No-parking zone is constantly checked for occupation. Any violation detected is sent to the manager via Email, hence eliminating the need to patrol the parking lot.

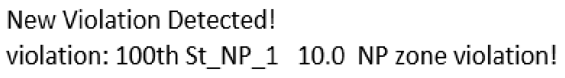


The system has two machine learning models - a car detector and a signature matcher. The car detector is a simple binary classifier used to check if a spot is occupied by vehicles. If a spot is occupied, its corresponding entry in the database is modified. If in the last scan the spot is also occupied, it means the car could be there for a while. At this stage the signature matcher is invoked to test if the two cars are the same (it could be a different car if the sampling rate is low). If the signature is identical, then the time measurement is updated.

Finally, a violation analyzer is regularly called to scan the database. If a no-parking zone is occupied for a certain amount of time (to eliminate false positives of cars passing through) or if cars have parked overtime (common in street parking), a violation report is sent via email. Below are sample reports sent to the administrator.



```
New Violation Detected!
violation: 100th St_SLOT_0 40.0  overtime parking violation!
violation: 100th St_SLOT_2 40.0  overtime parking violation!
```



```
New Violation Detected!
violation: 100th St_NP_1   10.0  NP zone violation!
```

Binary Classifier

In our project, we used a relatively simple LeNet-5 style CNN to detect the existence of a car in a parking spot. We chose a binary classifier for our project because we want to detect both well painted and unpainted parking spots. Also, we want the network to be resilient to noise such as non-parking cars passing through the parking lot. The input images are essentially bounding boxes that were annotated and cropped out of the entire image of a parking lot beforehand. The input images for both the binary classifier and the Siamese network, which we will discuss in the next section, are represented as 3x150x150 torch tensors. The binary classifier is tasked to determine if the bounding box contains a car or not. Thus, the output of the classifier is a scalar probability. The loss function used for backpropagation and weight update is binary cross entropy loss with weight decay of 0.0005. The model was implemented in PyTorch. For training, we used an Adam optimizer with 0.0005 weight decay and trained the model for 5 epochs with batch size of 32. Learning rate was scheduled to be in [1e-4, 1e-5, 1e-6] throughout training.

Siamese Network

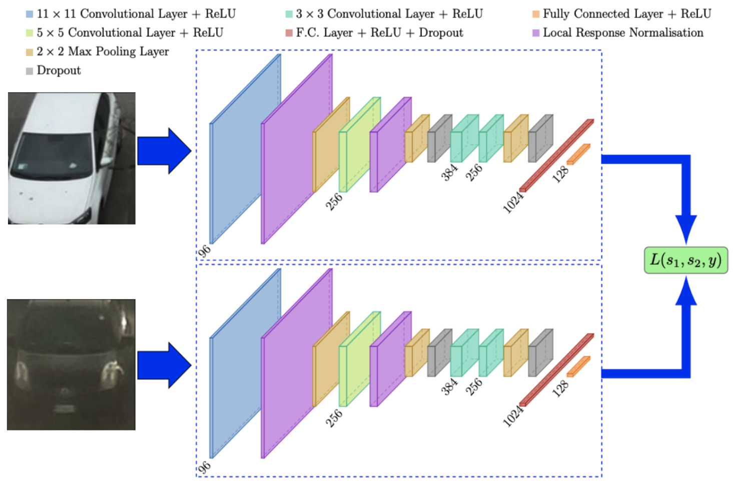
Siamese network is a class of neural network architecture which contains two or more identical subnetworks (Bromley et al. 1993). All the subnetworks in a siamese network have the same configuration with the same parameters and weights. Parameter updating is mirrored across all subnetworks. During training, two inputs are passed through the two subnetworks, one for each. Then subnetworks output embeddings of inputs, based on which we can compute the similarity between inputs. Eventually, we are able to classify if two inputs contain similar or different objects/information based on the similarity score. Therefore, the objective of a Siamese network is to learn embeddings of inputs that could accurately capture the similarity between inputs. Siamese network is a *one-shot* classifier that uses a supervised training approach to learn generic input features and, based on the training data, makes predictions about unknown class distributions.

In our project, we employ Siamese network to determine if the car in the current frame of parking spot is the same car as the one in the last frame of the same spot. The loss function we use is *contrastive loss* (Chopra et al. 2005). It is a distance-based loss as opposed to the more conventional error-prediction losses such as cross entropy loss. Contrastive loss is used to learn embeddings in which two similar points have a low Euclidean distance and two dissimilar points have a large Euclidean distance. The formula for computing contrastive loss is given below. Y corresponds to the label indicating if two inputs are similar or different. And we define D_W as the Euclidean distance between the embeddings of inputs. m is the margin that defines the radius to indicate that dissimilar pairs beyond this margin will not contribute to the loss. Notice that in the expression, $Y=0$ means two inputs are similar.

$$L_{contrast} = (1-Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\max(0, m - D_W)^2$$

Below figure shows the architecture and workflow of the Siamese network designed for our project. We used a deep CNN as the basic architecture of one subnetwork. The output embedding for one image is a 128-dimension vector. In our implementation, since weights are constrained to be identical for both subnetworks, we used one model and fed it two batches of images in succession. After that, we calculated the contrastive loss using both batches and then backpropagated to update weights of the model. By using one model instead of two with the same weights, we were able to save a significant amount of memory and increase computational efficiency. The model was implemented in PyTorch. For training, we used an Adam optimizer with 0.0005 weight decay and trained the model for 5 epochs with batch size of 64. Learning rate was scheduled to be in [1e-4, 1e-5, 1e-6] throughout training.

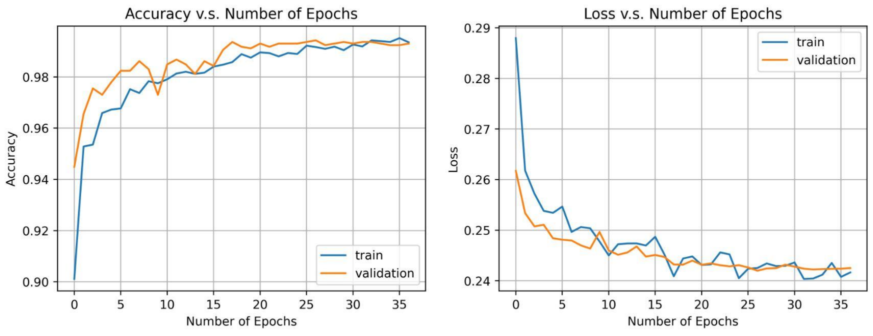
To adapt the Siamese network to our classification task, we applied a hard threshold on the Euclidean distance of generated embeddings of two input images to make predictions. If the distance is larger than the threshold, then we conclude that the two cars in the two images are different. Otherwise, they are deemed to be the same car. We experimented with different candidate thresholds, and the optimal threshold on the test set was 0.7338.



11 × 11 Convolutional Layer + ReLU
5 × 5 Convolutional Layer + ReLU
2 × 2 Max Pooling Layer
Dropout
3 × 3 Convolutional Layer + ReLU
F.C. Layer + ReLU + Dropout
Fully Connected Layer + ReLU
Local Response Normalisation
96
256
384
256
1024
128
$L(s_1, s_2, y)$

Evaluation

For the binary classifier, the final validation accuracy we managed to achieve is 0.992, and the test accuracy is 0.987. This good performance is largely due to the simplicity of the task. We only need a binary result, and there is a significant difference in images containing a car and images without a car. Thus, even though our classifier is relatively simple, it achieves very high accuracy. We include the training statistics plot below. One observation we make is that the validation accuracy is higher than training accuracy for most epochs (we re-distributed the iterations and created 37 “epochs” for better visualization). Two possible explanations are a small validation set and usage of dropout. With a small validation set, the estimation is likely to be noise. The usage of dropout during training introduces noise, which might drag the training accuracy down. To confirm our hypothesis, we evaluated on the training data without dropout. And that led to higher training accuracy than validation accuracy.



Since the Siamese network only learns embeddings, accuracy was not measured during training. Instead, different thresholds on embedding distance were experimented to produce the best accuracy. Based on experiment results, we set the threshold to 0.7338, which led to 0.931 test accuracy. Note that this is an empirical observation and we cannot provide any explanation on why this number gave us the best result. This is definitely something we would like to improve on in the future. To the best of our knowledge, we obtained higher accuracy than most existing Siamese networks supervised tasks with thresholding approach. This is largely due to the simplicity of the task. Since a new frame is taken in a short interval of time (a few minutes), the environmental conditions in the current frame is unlikely to greatly differ from those in the last frame. Also, with a fixed camera, the images are taken at the same angle. Even though we perturbed the input images to increase the complexity of the task, it’s relatively straightforward to tell if two cars are the same compared to tasks such as signature verification on which Siamese networks are usually applied.

Conclusion

In our project, we designed a smart parking system that automatically checks the status of cars in parking lots. If violations such as overtime parking are found, the system will send out notifications about the situation. Our project is a combination of software engineering and deep learning. We built our pipeline based on AWS and designed a binary classifier and a Siamese network to perform existence detection and vehicle comparison. We smartly divided the problem of detecting parking violations into two subproblems which are much easier to solve. Then we presented solutions to both subproblems and integrated them to construct our final system.

While both networks had good performance in their respective tasks, there are a number of improvements that can be made on our current system. First of all, the dataset we used contains the already annotated image of parking lots. However, in reality, we will not acquire these preprocessed images. Manually annotating images and cropping bounding boxes will be labor-intensive and inefficient. Thus, we would like to design another network that identifies at least some of the parking spots. There will be spots such as unpainted ones that still require manual effort. But human workload is overall reduced with such a network and the entire system will be less error-prone. Second, instead of setting a hard threshold on the Euclidean distance of learned embeddings, we want to rely on the power of deep learning to adapt contrastive loss to supervised learning. The basic idea is that after training the Siamese network to generate embeddings, we freeze the encoder and learn a classifier on top of the learned embeddings to predict labels using cross entropy loss. This idea is discussed and implemented in Khosla et al. (2020). Last but not least, a more comprehensive pipeline is desirable for the system. For example, we can design a web or mobile application that has a proper frontend and user system and ask officers to register themselves and document parking lots they are responsible for. If a violation is detected in a parking space, the app will only send out notifications to the officer who is responsible for that parking space.

References

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., Lecun, Y., Moore, C., Sackinger, E., & Shah, R. (1993). Signature Verification using a “Siamese” Time Delay Neural Network. International Journal of Pattern Recognition and Artificial Intelligence, 07(04), 669–688. https://doi.org/10.1142/s0218001493000339

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a Similarity Metric Discriminatively, with Application to Face Verification. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). https://doi.org/10.1109/cvpr.2005.202

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., & Krishnan, D. (2020). Supervised Contrastive Learning. 18661–18673.