



TraViQuA: Natural Language Driven Traffic Video Querying Using Deep Learning

Asim Sinan Yüksel¹ , Muhammed Abdulhamid Karabiyik^{2*} 

¹ Department of Computer Engineering, Süleyman Demirel University, Isparta 32100, Turkey

² Bor Vocational School, Niğde Ömer Halisdemir University, Niğde 51100, Turkey

Corresponding Author Email: abdulhamidkarabiyik@ohu.edu.tr



<https://doi.org/10.18280/ts.400213>

Received: 19 August 2022

Accepted: 26 January 2023

Keywords:

natural language processing (NLP), you only look once (YOLO), long short-term memory (LSTM), video query, deep learning

ABSTRACT

Video cameras are widely utilized and have ingrained themselves into many aspects of our daily life. Analysis of video contents is more challenging as the size of the data collected from the cameras increases. The fundamental cause of this challenge is because certain data, like the videos, cannot be queried. Our research focuses on converting traffic videos into a structure that can be queried. Specifically, an application called TraViQuA was suggested for natural language-based car search and localization in traffic videos. To query and identify cars, data including color, brand, and appearance time are used as features. The query is initiated in real time on live traffic feed, as the user enters the search term on the application interface. Our text to SQL conversion algorithm enables the mapping of a search term into a SQL query. Based on the response to the natural language query, TraViQuA can start the video from the relevant time. Deep neural networks were employed in our application for text to SQL conversion and feature extraction. Our research reveals that color and brand models had mean average precision of 98.714% and 91.742%, respectively. The text to SQL conversion had an 80% accuracy rate. To the best of our knowledge, TraViQuA is the first application that enables police officers to input a natural language description of a car and discover the car of interest that matches the description, bridging the gap in traffic video surveillance. Moreover, TraViQuA can be incorporated into other intelligent transportation systems to support law enforcement officials in urgent situations like hit-and-run incidents and amber alerts.

1. INTRODUCTION

Roads were required for the transportation of goods as international trade relations began to grow. Large-scale roads have been constructed in response to this requirement. Road construction increases the risks like theft, terrorism, and accidents. To cope with the risks, people have created a variety of road control techniques [1, 2]. Today, this is served by emerging technology like video surveillance cameras. The practice of recording video camera images extended throughout society in the 1990s [3]. As a result, new uses for cameras in traffic and road control emerged. However, the extent of the road networks has grown significantly over time. For instance, Turkey has 68,526 kilometers of motorways that are monitored by video cameras positioned at regular intervals [4]. More personnel are needed to interpret the video camera images [5]. The traditional manual interpretation approaches are ineffective, time-consuming, and error-prone, facing the enormous amounts of video data. Traffic officers must constantly analyze an insanely large number of live video footages as traffic camera networks continue to expand, adding significantly to their workloads. Smart systems are needed to reduce the error rate and the workload.

Studies on artificial intelligence have advanced quickly in recent years and offer solutions to many issues in several sectors [6]. Artificial intelligence will inevitably be used, particularly in fields requiring video analysis. Natural language processing is another field of artificial intelligence applications. This method is typically employed in

conjunction with image processing techniques to examine traffic camera images via natural language queries [7].

This study devises a hybrid approach coupling three different methods to analyze traffic videos, including object detection, video summarization, and text to SQL conversion. As a subfield of image processing, object detection seeks to locate objects in the content of image data by accounting for their unique properties [8]. In this instance, cars serve as the target objects for our system, and features to investigate include color and brand.

Video summarization is an important topic for the big data analysis of video images. Chen et al. [9] employed different methods for video summarization. Hussain et al. [10] implemented video summarization as a feature to provide general summary or query-based summary of data. In our car-oriented system, the images are summarized according to the frames that include cars.

In the object recognition module, deep neural networks are trained using the pretrained you only look once (YOLO) models [11]. The module only has two models: a brand model and a color model. The former was trained on the Car Connection Picture Dataset (CCPDS) [12], while the latter was trained by our own dataset.

The results from the above two models were used to develop the video summarization system. Each video summary was created by determining the changes in the video timeline, according to the results from the brand model and the color model.

Text to SQL conversion is an increasingly popular topic of

natural language processing [13]. It allows users to make queries using natural language. Our study applies the SmBoP model, which is developed based on semantic parsing. The spider dataset was used for model training [14].

Natural language applications for video querying systems have not yet been explored in the literature. There are numerous limitations with the relevant studies. For example, scholars have focused on specific queries made in natural language, without considering flexible structures of natural language expressions. The target videos cover a certain area and have a static structure. The accuracy is often reported on static images, which does not reflect the real-world conditions.

Similar studies have suggested AI-based models for identifying the make and model of cars, but they ignore crucial information like the color of the car. Color is a key factor in ensuring that an AI model is reliable and appropriate for use in video surveillance applications. The query "A white Mercedes" on a highway is more useful than the query "A Mercedes" because it includes the color feature.

TraViQuA corrects the flaws in earlier studies and suggests a fresh, reliable model for application in practical situations like traffic video monitoring. Our color model provides an additional filter to narrow down the results, and minimizes the time to recognize a car in a scene. To the best of our knowledge, TraViQuA is the first application that fills the gap in traffic video surveillance by offering a human-computer interface, which enables police officers to type a natural language description of a car and find the car of interest that matches the description.

2. LITERATURE REVIEW

For a very long time, scholars have been exploring the problem of text to SQL conversion. The studies in this field were not very successful in the early phases, due to the lack of adequately sophisticated language models. Bidirectional Encoder Representations from Transformers, or BERT, was launched by Google in 2018 and its use to convert text to SQL improved conversion success rates [15].

Studies on text to SQL conversion commonly employ the Spider dataset, which was created by Yale University's Language, Information, and Learning Laboratory (LILY) [16]. TypeSQL, SQLNet, and SyntaxSQLNet are a few examples of studies created using the Spider dataset [17-19]. However, the accuracy rates of these studies continue to hover around 30%. The BERT model helps to improve the accuracy rates. This model has been hybridized with RatSQL (Relation-Aware Schema Encoding and Linking), which is supported by Microsoft [20]. Complex databases are compatible with RatSQL. It uses a semantic parsing technique based on the connections between the tables. RatSQL+BERT has a 61.9% accuracy rate. By creating the RatSQL model's encoder, Rubin and Berant developed the text to SQL transformation model known as SmBoP (Semi-autoregressive Bottom-up Semantic Parsing) [14]. The accuracy rate for SmBoP is 71.1%.

Based on the T5-3B pre-trained model, another model called PICARD was developed [21] by programming languages Haskell and Python [22, 23]. PICARD operates with a 75.1% accuracy rate. An application called Bridge was designed by Lin et al. using BERT as the language model. Researchers have utilized database schema matching and input from natural language to accomplish SQL query prediction. Database schemas were pruned in their investigations. This

technique avoided predicting erroneous SQL queries. The accuracy rate of the developed system is 68.3% [24]. Huang et al. developed a text to SQL model named Rasap, using Electra pre-trained text encoder and RatSQL database encoder. The accuracy rate of the developed model was 70% [25, 26].

Studies on object recognition have accelerated thanks to advances in deep learning. The YOLOv2 model was used by Li et al. to study multiple object detection. Their research centered on classifying the various traffic vehicle classes. Four classes of vehicles were used in the object detection phase of the experiments: trucks, cars, buses, and vans. Multi-object detection achieved an accuracy rate of 89.64% whereas simple object detection achieved a success rate of 92.09% [27]. For the Parrot AR Drone 2, Rohan et al. created a real-time object detection system. The system processes the images captured by the drone's front camera. The authors used single shot detector (SSD) and convolutional neural network (CNN) techniques for object recognition applications. The accuracy rate for the SSD technique, which was trained on 5100 images, was 98.2% [28]. Ćorović et al. [11] examined real-time traffic camera images. The research was based on the fact that traditional networks yield sluggish results. To improve performance, they used a quicker model called YOLOv3 with 5 object classes: cars, trucks, pedestrians, traffic signs, and lights. The accuracy rate reached 46.6%. The low accuracy rate was caused by very small objects in the images [11]. Eggert et al. tested the detection of small objects called "Look Closer." The Faster R-CNN was employed to find logos of small businesses, and achieved an 80% accuracy [29]. Stuparu et al. tried to detect cars in satellite and drone images, using the RetinaNet. Their vehicle detection accuracy was 72% [30].

Video summarization systems are important to applications with a large amount of video content. Ma et al. proposed a framework for video summarization, which focuses on concepts that attracted people's attention in a video. A summary video was obtained by compressing the area occupied by the object in the time series, i.e., the area that attracts user attention [31]. Mahasseni et al. [32] presented a video summarization system using unsupervised learning. Specifically, subsets of frames in the video stream were analyzed semantically, the repetitive subsets were removed, and the video was summarized. The video frame subsets to be extracted were determined by the LSTM, a deep neural network [32]. Gong et al. [33] argued that unsupervised learning is not directed towards human thoughts, and suggested video analysts to focus on a particular topic. Following this train of thought, they put forward a summarization system with selected video frame subsets, in the context of supervised learning. The subset selection and feature extraction were achieved using greedy algorithm and Bayesian networks [33].

Wu et al. [34] proposed a high-density peak search clustering algorithm for summarizing static videos. The algorithm was developed by combining similar parts of multiple videos. The high success rate of their algorithm was observed through experiments [34]. Otani et al. [35] developed a semantic approach for video summarization based on the SumMe dataset [35], and presented the benefits of deep semantic features in video summarization [36].

Tellex and Katz are leading researchers in the field of natural language queryable videos. Tellex and Roy targeted the indoor images taken from a fish-eye camera, and tried to determine the equivalents of the words "along" and "across" in the video. This is realized by classifying these two words with

decision trees [37]. Katz et al. [38] utilized four natural language queries to identify the moving objects in videos taken by a single camera, and applied the START method for textual question and answer.

3. METHODOLOGY

TraViQuA is composed of three main modules. The first module is responsible for object detection. In this module, the video stream from the traffic surveillance camera is imported. The input is divided into frames and sent to the color model, which aims to identify cars and their colors. The results of the color model, together with the time stamps and object locations, are taken as parameters of the submodule for clipping. This submodule converts each detected object into a separate image, and imports it to the brand model for brand detection.

The detection results are further sent to the video

summarization module, which evaluates the results according to three features: color, brand and time stamp. The video summarization module compares the results of the current frame with those of the previous frame. If the results are different, the current results are stored in the database. These processes continue as long as the video stream is provided.

In the text to SQL module, the user's natural language query is imported, and converted to an SQL query via semantic parsing. The output is the textual summary of the video, which is displayed to the user. These results are filtered according to the user's query. The user can start the video from the relevant time or see the summary of results as a list on the interface. Figure 1 shows the general flow of the application.

TraViQuA requires a minimum of 2.5 GHz x64 processor, 8 GB RAM and 5 GB free disk space. More free space is needed if the video input to the system grows. The interface of the desktop application is designed to be simple and user friendly (Figure 2).

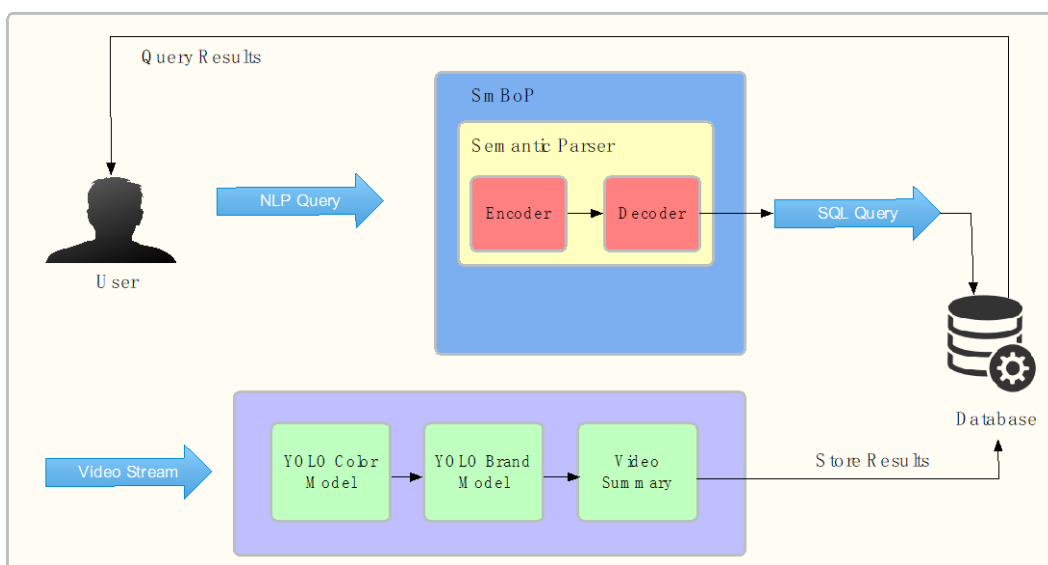


Figure 1. General flow of the application

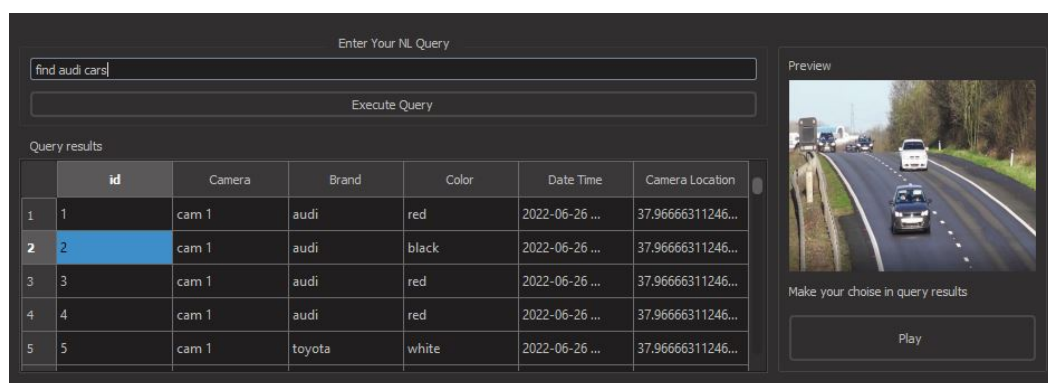


Figure 2. User interface of the desktop application

3.1 Object detection module

The object detection module performs operations on the input video. The objects are detected by real-time video processing techniques. The goal is to identify color and brand features of the cars. Therefore, feature extraction is applied to provide queryable structure. The results of this module are imported to the video summarization module.

3.1.1 Color module

The input video is firstly processed in the color module, before being divided into frames. The divided frames are tagged with the current date and time. The date and time tags are the key parameters for video analysis. The color module trained with YOLOv5 is used for detecting cars and their colors in the tagged frames. The module outputs the coordinates, color and reliability value of each car it detects on

the frame. On this basis, each frame is divided into a number of temporary frames, which are transferred to the brand module and passed to the other module of the application. Figure 3 explains the operation of the color module.

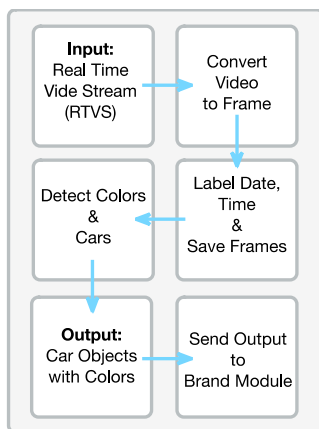


Figure 3. Operation of the color module

The deep neural network was trained by 1136 images containing 6 color classes: white, gray, red, blue, yellow and black. The dataset of the color model is prepared with car images grouped by their colors. Full supervised learning was adopted to label the images. For example, all the black cars are labeled as black. In this way, each color class only contains car objects with various brands and models (Figure 4).

There are four main parts of our YOLOv5 model for car and color detection:

- 1) **Backbone:** This part contains a CNN acting as a feature extraction network, which extracts and aggregates feature maps from input images.
- 2) **Neck:** This part, connecting the Backbone to Head, generates feature pyramids. These pyramids are designed to ensure accuracy and speed. In YOLOv5, PANet is used as Neck to produce feature pyramids.
- 3) **Head:** This part uses the features created by Neck to predict boxes and classes, producing bounding box and confidence scores.
- 4) **Detection:** This last part outputs predicted bounding box with class label and its confidence score.

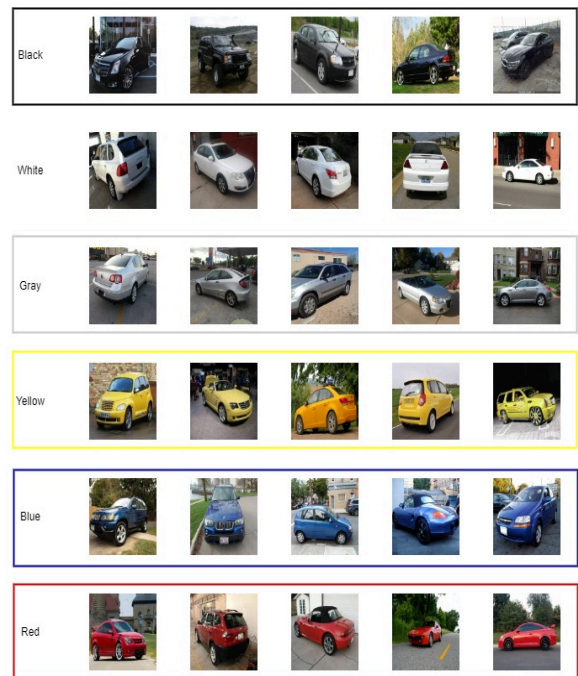


Figure 4. Car labeling

Figure 5 overviews the YOLOv5 model used to detect car brands and car colors.

3.1.2 Brand module

This module detects the brands of the cars on the temporary frames from the color module. The deep neural network for brand detection was trained with YOLOv5. The temporary frames from the color module are imported to the brand module sequentially. The brands identified by the brand module are transferred to the video summarization module. The temporary frames are removed from memory at this stage. Figure 6 explains the operation of the brand module.

The brand model was trained on the CCPDS dataset [12], which includes 297,000 car images. These images contain such car parts as tires, headlights, and door handles. The car parts are unnecessary for training, for our aim is to detect the cars as a whole in traffic surveillance videos. Therefore, as the first step, irrelevant training images are removed from the dataset.

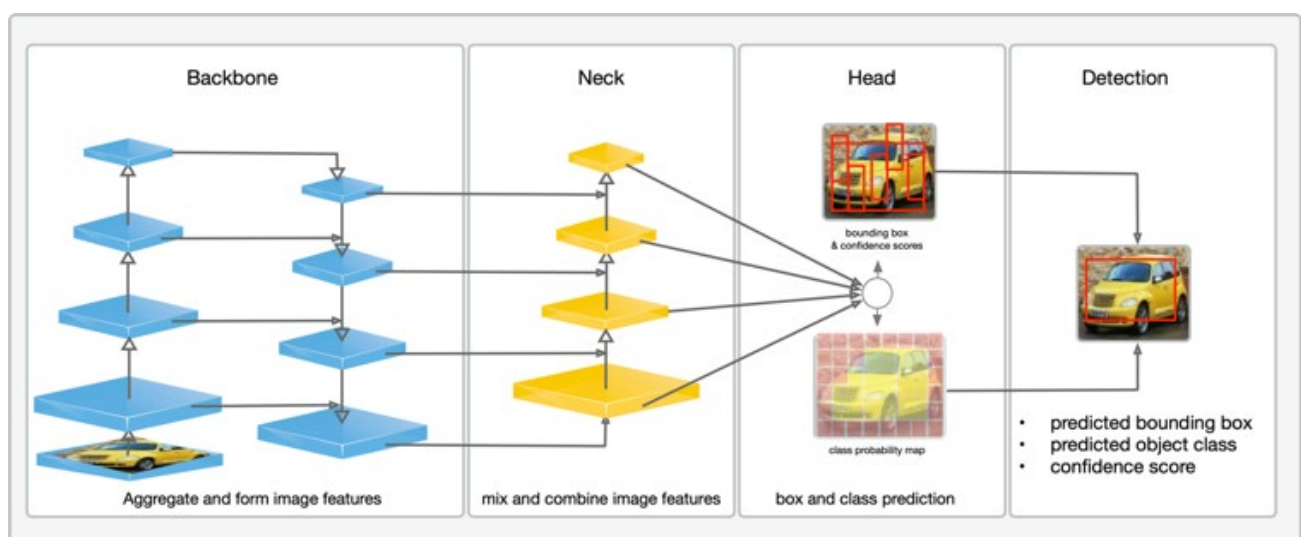


Figure 5. Overview of YOLOv5 model

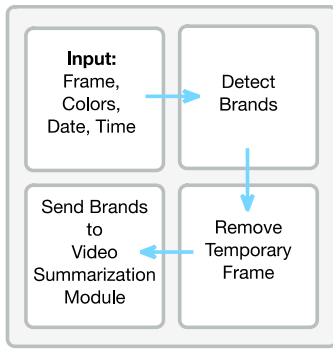


Figure 6. Operation of the brand module

Next, the remaining images are preprocessed through rescaling and cropping. YOLO detects objects with the help of anchor boxes. These boxes must be of the right size and number to reduce the time to detect these anchors and improve the positioning accuracy of cars. For this purpose, rescaling is carried out to optimize the aspect ratio and provide robust learning. As a result, the training and detection images share the same aspect ratio. Cropping ensures that training image only contains the car objects. It is a useful pre-processing method where the position of the car has large variance [39]. After pre-processing, 31583 images with 40 brand classes were obtained. Among them, 27582 images were used for training. Finally, car images with a reliability over 80% were selected for training YOLOv5 [39].

3.2 Video summarization module

The video summarization module runs our breakpoint detection algorithm on the objects detected by the previous module. The steps of the breakpoint detection algorithm are as follows:

- 1) Extract the frames from the video as a sequence of images.
- 2) Detect the cars in the current frame and store them in a list (newCarList). For example: 1 White Mercedes,

1 Blue Audi.

- 3) Pass the newCarList to the control function named “detectBreakPoint”.
- 4) Run detectBreakPoint(currentCarList, newCarList): Check if the detected objects in the newCarList are same with the cars in the currentCarList. If not, update the currentCarList and store it in the database.

Two sample scenarios are presented to better illustrate the flow of the breakpoint detection algorithm.

Scenario 1. If there is one car in the current frame and in the next frame(s): A White Mercedes appears in the current frame. Since no white Mercedes appears in our empty currentCarList, the algorithm marks it as a new car. Therefore, the first appearance time of the car in the frame is stored in the database as a breakpoint. Then, the currentCarList is updated with the new car (White Mercedes). If the White Mercedes still appears in the following frames, nothing is done for only the first appearance time is needed.

Scenario 2. If new car(s) appear(s) in the following frame(s): A White Mercedes appears in the current frame and a Blue Audi appears near it in the next frame. In this scenario, the algorithm works as the same way in Scenario 1, and stores the first appearance of White Mercedes as a breakpoint. When the Blue Audi appears in the next frame, this appearance creates two new breakpoints: one for the White Mercedes and one for the Blue Audi. These are also stored in the database. Therefore, 2 breakpoints are determined for White Mercedes and 1 breakpoint for Blue Audi. The creation of another breakpoint for White Mercedes is to ensure that our system captures the natural language queries entered by user such as “show me the White Mercedes and Blue Audi that appeared together on cam1 between 13:00 and 15:00” or “show me all the White Mercedes cars that appeared on cam1 and cam2” etc.

Figure 7 illustrates the flow of the breakpoint detection algorithm.

The video summarization module offers a semantic approach. The results of the module include date, time, car color, car brand, camera information and camera location. These results serve as an index for making queries on the video.

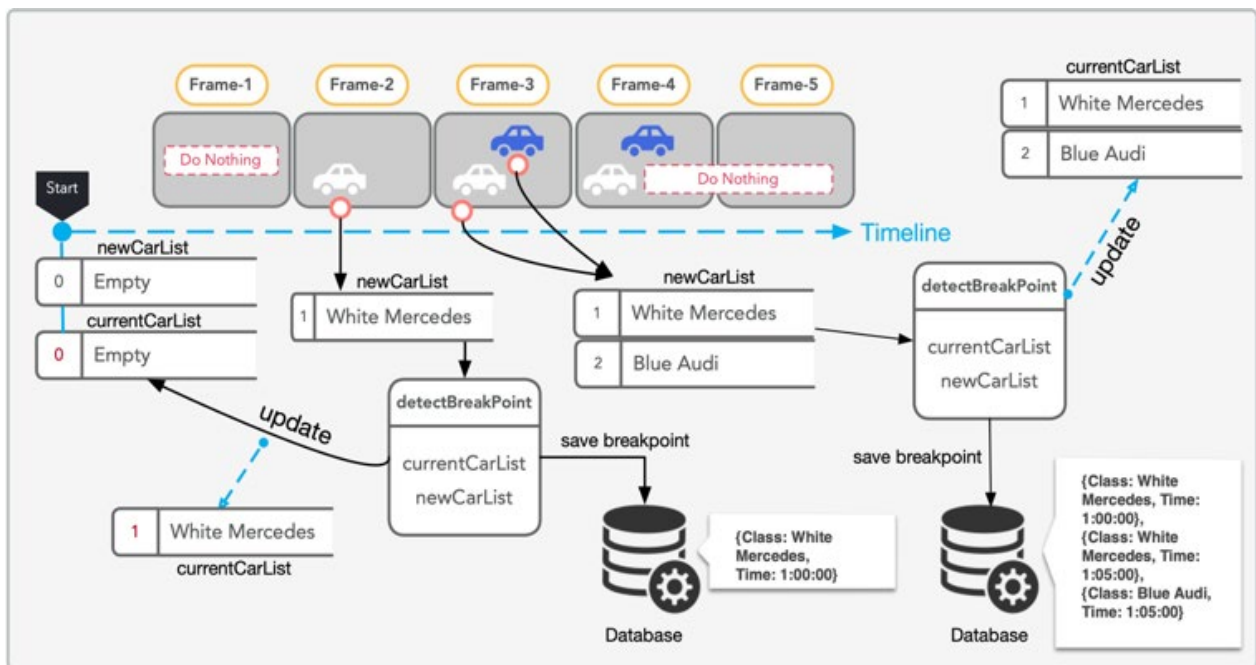


Figure 7. Flow of the breakpoint detection algorithm

3.3 Text to SQL conversation module

The LSTM is a deep learning model widely applied in real-world problems, namely, robot control, machine translation, human action recognition, speech recognition, and grammar learning. Our problem resides in the domain of machine translation. During the translation, the source language is transferred to the target language through AI models, without any human intervention. In our case, the aim is to translate sentences from a natural language, i.e., English (e.g., show me white cars appeared in cam1) to SQL language (e.g. Select * from cars where color = "white" and location="cam1"), so that users can query our database with SQL. The input from our natural language query interface is converted into an SQL query. It is more efficient and more accurate for an average user to locate a car using a natural language instead of SQL, which requires the mastery of technical knowledge. The natural language text is converted into SQL query in 4 steps: dataset collection, pre-processing, word embedding, LSTM treatment (Figure 8).

Dataset: We used the Spider dataset annotated by 11 Yale students. This large-scale cross-domain semantic parsing and text to SQL dataset contains 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains.

Pre-processing: Mathematical representation of natural

language without errors increases the success of translation. Therefore, preprocessing is needed to make the mathematical representation of natural language expression error-free. In this case, we adopted such preprocessing methods as spelling correction, tokenization, entity recognition, and stop word removal.

Word embedding: Since machine learning models cannot process text directly, we converted textual data into numerical data through word embedding, which helps to capture the semantic and syntactic context of a word, and improve the understanding of how similar/dissimilar it is to other term.

LSTM treatment: The problem of language translation can be solved by converting text to SQL. For this purpose, we employed semantic parsing algorithm. The algorithm converts sequences from one domain (sentences in English) to sequences in another domain (SQL). The LSTM-based SmBoP was adopted to realize the text-to-SQL conversion. Figure 9 shows the operation of our text to SQL module.

During the text to SQL conversion, the database design directly affects the conversion accuracy. The conversion accuracy is high, if databases do not have a large number of tables. Therefore, our database is designed based on a single table. Date, time, brand, color, camera information and camera coordinates are all stored in the database.

The number of queries that can be made on the traffic videos are limited. Figure 10 shows the possible parameters for a car.

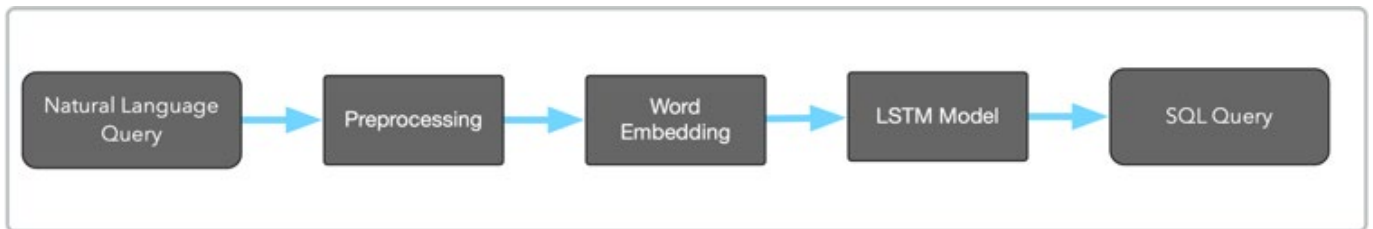


Figure 8. Operation of text to SQL module

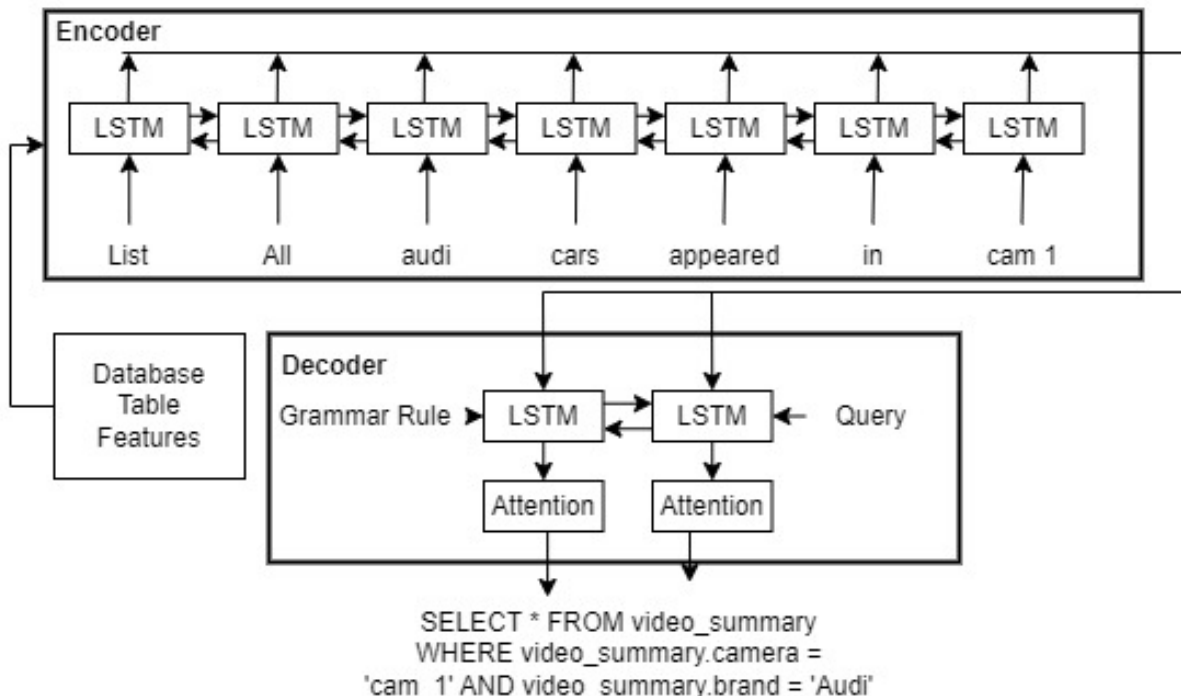


Figure 9. Operation of text to SQL module

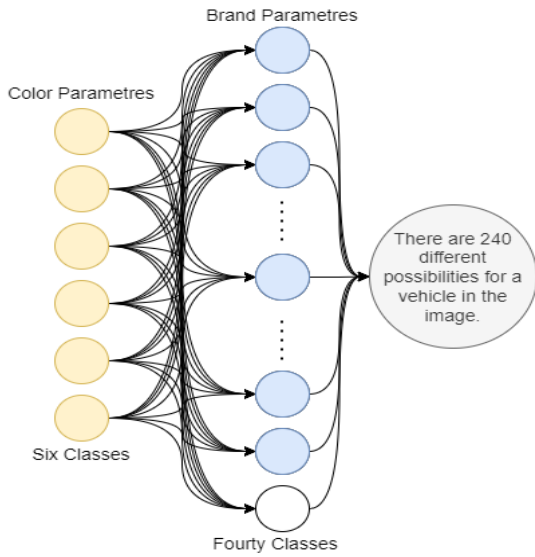


Figure 10. Possible parameters for a car

4. RESULTS AND DISCUSSIONS

Three deep neural networks are utilized to develop TraViQuA (Table 1).

Table 1. Deep learning models

Module	Model	Data Set	Model Size	Output
Color	YOLOv5	Custom	659.4 Mb	Colors
Brand	YOLOv5	CCPDS	661.1 Mb	Brands
Text to SQL	SmBoP	Spider	1.27 Gb	SQL

Specifically, YOLOv5, SSD, and Faster R-CNN were compared experimentally for model selection. The comparison intends to determine the colors of the cars. During the experiment, a dataset of 1136 images was divided into a training set and a test set at the ratio of 4:1. The success rates of the models is shown in Table 2.

Table 2. Comparison of the models

Model	mAP
YOLOv5	0.97321
Faster R-CNN	0.93100
SSD	0.79890

Compared with the literature, the YOLOv5 was more successful than other models [40, 41]. For this reason, YOLOv5 was chosen for our application. Next, a dataset consisting of 1136 images with 6 color classes was used to train the YOLOv5-based color detection model. Figure 11 shows the image distributions for each class.

The color model was trained for 200 epochs. Figure 12 shows the trends of the loss functions. It can be seen that the loss functions form a descending curve. As the training continues, the losses approach zero, indicating that the model produces successful results.

YOLOv5 calculates the final loss score by combining its three loss functions, namely, box_loss, cls_loss, and obj_loss. Mean squared error (MSE) is used for box_loss. This is the simplest and most commonly used loss function. The MSE can

be calculated by taking the difference between model predictions and the ground truth. The difference is then squared, and averaged across the whole dataset. The equation for MSE can be expressed as:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \tilde{y}_i)^2 \quad (1)$$

where, n is the number of samples; \tilde{y}_i is the predicted value; y_i is the ground truth.

Binary cross entropy (BCE) is used for Obj_loss loss function. It represents the confidence of object presence. The equation for BCE can be expressed as:

$$BCE = -\frac{1}{n} \sum_{i=1}^n y_i \log(\tilde{y}_i) - (1 - y_i) \log(1 - \tilde{y}_i) \quad (2)$$

where, \tilde{y}_i is the i -th predicted value in the model output; y_i is the corresponding target value; n is the number of scalar values in the model output.

Categorical cross entropy (CCE) is used for Cls_loss function. Formally, it is designed to quantify the difference between two probability distributions. The equation for CCE can be expressed as:

$$CCE = -\sum_{i=1}^n y_i \log(\tilde{y}_i) \quad (3)$$

The mAP metric was selected to compare the actual bounding box with the detected box, before returning a score. The higher the score, the more accurate the object detection. YOLOv5 gives mAP values in two ranges. mAP_0.5 is the mAP with 0.5 at the IoU (Intersection over Union) threshold. mAP_0.5:0.95 is the average mAP over different IoU thresholds ranging from 0.5 to 0.95. In both ranges, it is expected that the curve rises towards 1. As can be seen in Figure 12, the curve moves above 0.95 for the color model, indicating that our model is highly successful. Figure 13 shows the sample test results using the color model.

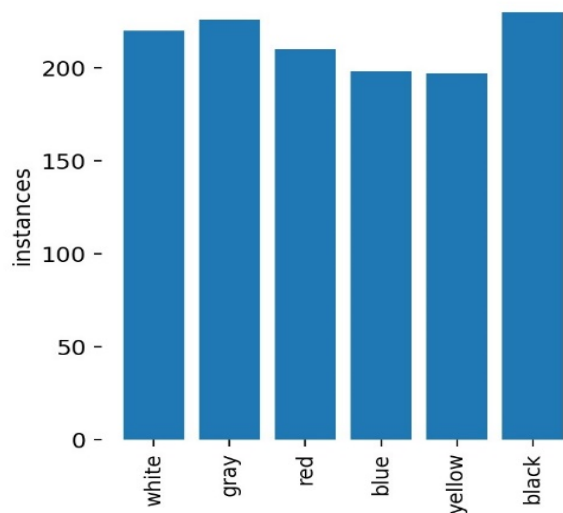


Figure 11. Image class distributions for the color model

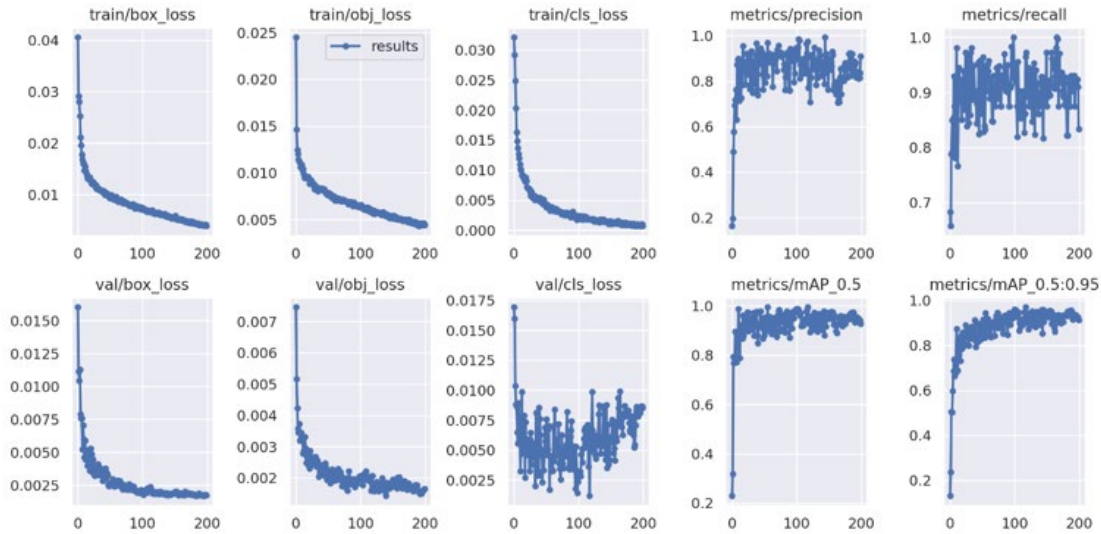


Figure 12. Training results for the color model



Figure 13. Sample test result for a yellow car

Next, a dataset of 27582 images on which 40 brand classes was used to train the YOLOv5-based brand detection model. Figure 14 displays the image distributions for each class in the brand dataset.

The imbalance in the class distributions can be attributed to two factors: One is the scarcity of special production vehicles, which is frequently encountered in real life. That is why it is impossible to create an evenly distributed dataset between classes. The second is the pre-processing on CCPDS dataset. During the preprocessing, car images with a reliability above 80% were selected. Figure 15 shows the trends of the loss functions.

It can be seen that the mAP curve approaches 0.91, suggesting that our model is highly successful. Figure 16 shows the sample test results using the brand model.

The text to SQL accuracy was evaluated by the exact match, i.e., a natural language query produced an SQL query identical to a predefined true query. The model used for text to SQL conversion was tested with different queries in natural language, producing an 80% accuracy.

Table 3 compares TraViQuA with other video summarization systems (VSS) in the literature. All video summarization systems share the video playback feature. But these systems take the places of interests from the videos and create a summary video shorter in size than the normal video. In our system, there is no interference with the videos. Playback is started by going directly to the places of interests in the video.

The VSSs were compared against three reference values: the video summarization algorithm; the queryable output

structure; the use of semantic or clustering approach in video summarization.

Our end-to-end test reports a 70% success rate. In this test, a natural language query was inputted into the system. Then, the results corresponding to the query were examined on the video and the accuracies were evaluated.

The studies on natural language expressions have a common difficulty: they cover very large domains. TraViQuA only considers the transportation domain, specifically the features of the cars. Working in a specific domain makes the results more decisive. Of course, the TraViQuA design allows for extensions. In future studies, the application can be improved by adding features such as traffic signs, vehicle types and traffic accidents to the query criteria.

In the video summarization module, the properties of the traffic video used for querying are stored in the database. A single database table is sufficient for the features. The use of a single table improves the accuracy of text to SQL conversion. However, a single table may not be sufficient to add features for future studies. To prevent the potential accuracy decline, the domain definitions of the newly created tables should be expressed clearly and each table should be designed to hold a feature.

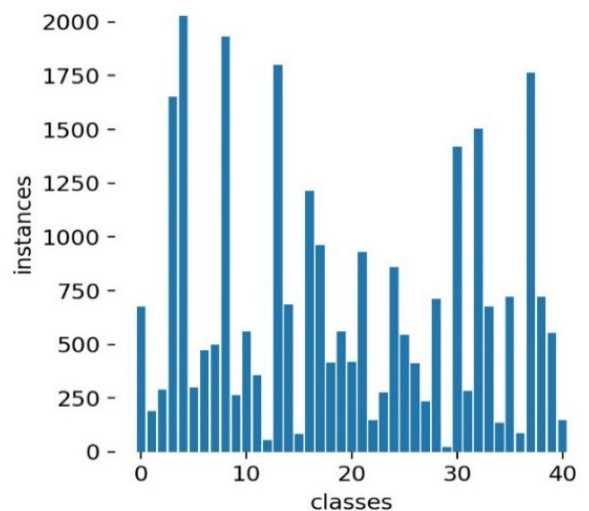


Figure 14. Image class distributions for the brand model

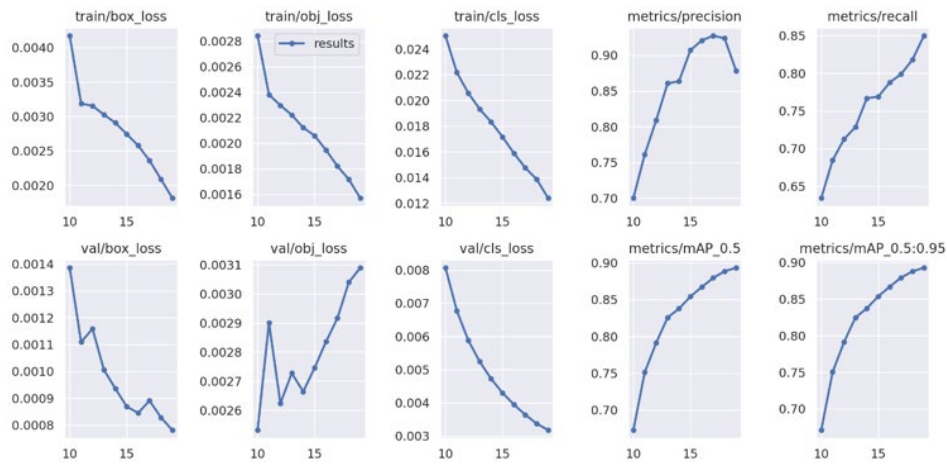


Figure 15. Training results for brand model



Figure 16. Sample results for Chevrolet brand

Table 3. Comparison with other systems

VSS	Approach	Queryable Output	Algorithms
TraViQuA	Semantic	X	YOLOv5
Mahasseni et al. [32]	Clustering	-	LSTM & GAN
Gong et al. [33]	Clustering	-	VRHDPS
Wu et al. [34]	Clustering	-	VRHDPS
Fajtl et al. [35]	Clustering	-	LSTM
Otani et al. [36]	Semantic	-	DNN

5. CONCLUSIONS

This paper puts forward TraViQuA, an application that processes video streams of traffic surveillance cameras and analyzes cars based on the following features: their color and brand features, their appearance on the camera as date-time information and in which camera they were appeared as location information. Specifically, color and brand models were developed using the YOLOv5 pre-trained model and video images were analyzed semantically. Finally, the changes in the video images were detected and a queryable summary of the video was obtained.

There are two main challenges in querying videos with natural language: it is difficult to make sense of the natural language, and the videos are not in a queryable structure. TraViQuA successfully overcomes these challenges by utilizing deep learning techniques. Our highly accurate query module enables traffic officers to find the car of their interest in a traffic surveillance video. They only need to type in natural language through a graphical user interface, without knowing technical knowledge.

In future studies, the scope of analysis can be expanded by adding more features, such as traffic accidents, car body types and traffic rule violations. TraViQuA supports the analysis of

multiple cameras, and stores the coordinates of the cameras in the database, paving the way for vehicle route determination. In addition, the structure of our application can be extended to be used in many different fields, where there is a need for automatic analysis and querying of video files.

REFERENCES

- [1] Rümeyşa, K.A.R.S. (2019). Osmanlı'da Ticarî Yol Sistemi ve Taşımacılık: Konya Örneği (1700-1750). *Journal of Universal History Studies*, 2(2): 296-307. <https://doi.org/10.38000/juhis.582506>
- [2] Huang, C.J., Hu, K.W., Ho, H.Y., Chuang, H.W. (2021). Congestion-preventing routing and charging scheduling mechanism for electric vehicles in dense urban areas. *Information Technology and Control*, 50(2): 284-307. <https://doi.org/10.5755/j01.itc.50.2.27780>
- [3] O'Regan, T. (1991). From piracy to sovereignty: international video cassette recorder trends. *Continuum: Journal of Media & Cultural Studies*, 4(2): 112-135. <https://doi.org/10.1080/10304319109388202>
- [4] KGM. Road Network Information. <https://www.kgm.gov.tr/Sayfalar/KGM/SiteEng/Root/Gdh/GdhRoadNetwork.aspx>, accessed on May 29, 2022.
- [5] Luff, P., Heath, C. (2012). Some 'technical challenges' of video analysis: social actions, objects, material realities and the problems of perspective. *Qualitative Research*, 12(3): 255-279. <https://doi.org/10.1177/1468794112436655>
- [6] Chakraborty, T., Sikdar, S.S., Ganguly, N., Mukherjee, A. (2014). Citation interactions among computer science fields: a quantitative route to the rise and fall of scientific research. *Social Network Analysis and Mining*, 4: 1-18. <https://doi.org/10.1007/s13278-014-0187-3>
- [7] Zhang, C.M., Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration*, 23: 100224. <https://doi.org/10.1016/j.jii.2021.100224>
- [8] Amit, Y., Felzenszwalb, P., Girshick, R. (2020). Object detection. *Computer Vision: A Reference Guide*, 1-9. https://doi.org/10.1007/978-3-030-03243-2_660-1
- [9] Chen, B.W., Wang, J.C., Wang, J.F. (2009). A novel video summarization based on mining the story-structure and semantic relations among concept entities. In *IEEE Transactions on Multimedia*, 11(2): 295-312. <https://doi.org/10.1109/TMM.2008.2009703>

- [10] Hussain, T., Muhammad, K., Ding, W.P., Lloret, J., Baik, S.W., de Albuquerque, V.H.C. (2021). A comprehensive survey of multi-view video summarization. *Pattern Recognition*, 109: 107567. <https://doi.org/10.1016/j.patcog.2020.107567>
- [11] Ćorović, A., Ilić, V., Đurić, S., Marijan, M., Pavković, B. (2018). The real-time detection of traffic participants using YOLO algorithm. In 2018 26th Telecommunications Forum (TELFOR), IEEE, 1-4. <https://doi.org/10.1109/TELFOR.2018.8611986>
- [12] Gervais, N. (2020). The Car Connection Picture Dataset. <https://github.com/nicolas-gervais/predicting-car-price-from-scraped-data/tree/master/picture-scraper/>, accessed on May 29, 2022.
- [13] Wang, P., Shi, T., Reddy, C.K. (2020). Text-to-SQL generation for question answering on electronic medical records. In *Proceedings of The Web Conference 2020*, 350-361. <https://doi.org/10.1145/3366423.3380120>
- [14] Rubin, O., Berant, J. (2021). SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 311-324. <https://doi.org/10.18653/v1/2021.naacl-main.29>
- [15] Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 4171-4186. <https://doi.org/10.18653/v1/N19-1423>
- [16] Brunner, U., Stockinger, K. (2021). Valuenet: A natural language-to-SQL system that learns from database information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, IEEE, pp. 2177-2182. <https://doi.org/10.1109/ICDE51399.2021.00220>
- [17] Yu, T., Li, Z., Zhang, Z., Zhang, R. and Radev, D. (2018). TypeSQL: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 588-594. <https://doi.org/10.18653/v1/N18-2093>
- [18] Gur, I., Yavuz, S., Su, Y., Yan, X.F. (2018). DialSQL: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 1339-1349. <https://doi.org/10.18653/v1/P18-1124>
- [19] Huo, S.Y., Ma, T.F., Chen, J., Chang, M., Wu, L.F., Witbrock, M. (2019). Graph enhanced cross-domain text-to-SQL generation. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, Association for Computational Linguistics, 159-163. <https://doi.org/10.18653/v1/D19-5319>
- [20] Wang, B.L., Shin, R., Liu, X.D., Polozov, O., Richardson, M. (2020). RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 7567-7578. <https://doi.org/10.18653/v1/2020.acl-main.677>
- [21] Nogueira, R., Jiang, Z.Y., Pradeep, R., Lin, J. (2020). Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Association for Computational Linguistics, 708-718. <https://doi.org/10.18653/v1/2020.findings-emnlp.63>
- [22] Dobesova, Z. (2011). Programming language python for data processing. In *2011 International Conference on Electrical and Control Engineering*, IEEE, pp. 4866-4869. <https://doi.org/10.1109/ICCEENG.2011.6057428>
- [23] Hudak, P., Peyton Jones, S., Wadler, P., Boutel, B., Fairbairn, J., Fasel, J., et al. (1992). Report on the programming language Haskell: A non-strict, purely functional language version 1.2. *ACM SigPlan Notices*, 27(5): 1-164. <https://doi.org/10.1145/130697.130699>
- [24] Lin, X.V., Socher, R., Xiong, C. (2020). Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Association for Computational Linguistics, 4870-4888. <https://doi.org/10.18653/v1/2020.findings-emnlp.438>
- [25] Ni, P., Okhrati, R., Guan, S., Chang, V. (2022). Knowledge graph and deep learning-based text-to-GraphQL model for intelligent medical consultation chatbot. *Information Systems Frontiers*, 1-20. <https://doi.org/10.1007/s10796-022-10295-0>
- [26] Clark, K., Luong, M.T., Le, Q.V., Manning, C.D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*. <https://doi.org/10.48550/arXiv.2003.10555>
- [27] Shi, B.B., Li, X., Nie, T.T., Zhang, K.B., Wang, W.J. (2021). Multi-object recognition method based on improved yolov2 Model. *Information Technology and Control*, 50(1): 13-27. <https://doi.org/10.5755/j01.itc.50.1.25094>
- [28] Rohan, A., Rabah, M., Kim, S.H. (2019). Convolutional neural network-based real-time object detection and tracking for parrot AR drone 2. *IEEE Access*, 7: 69575-69584. <https://doi.org/10.1109/ACCESS.2019.2919332>
- [29] Eggert, C., Brehm, S., Winschel, A., Zeche, D., Lienhart, R. (2017). A closer look: Small object detection in faster R-CNN. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, pp. 421-426. <https://doi.org/10.1109/ICME.2017.8019550>
- [30] Stuparu, D.G., Ciobanu, R.I., Dobre, C. (2020). Vehicle detection in overhead satellite images using a one-stage object detection model. *Sensors*, 20(22): 6485. <https://doi.org/10.3390/s20226485>
- [31] Ma, Y.F., Hua, X.S., Lu, L., Zhang, H.J. (2005). A generic framework of user attention model and its application in video summarization. *IEEE Transactions on Multimedia*, 7(5): 907-919. <https://doi.org/10.1109/TMM.2005.854410>
- [32] Mahasseni, B., Lam, M., Todorovic, S. (2017). Unsupervised video summarization with adversarial LSTM networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 202-211. <https://doi.org/10.1109/CVPR.2017.318>
- [33] Gong, B., Chao, W.L., Grauman, K., Sha, F. (2014). Diverse sequential subset selection for supervised video summarization. *Advances in Neural Information Processing Systems*, 27.

- [34] Wu, J.X., Zhong, S.H., Jiang, J.M., Yang, Y.Y. (2016). A novel clustering method for static video summarization. *Multimedia Tools and Applications*, 76: 9625-9641. <https://doi.org/10.1007/s11042-016-3569-x>
- [35] Fajtl, J., Sokeh, H.S., Argyriou, V., Monekosso, D., Remagnino, P. (2019). Summarizing videos with attention. In *Computer Vision-ACCV 2018 Workshops: 14th Asian Conference on Computer Vision*, Perth, Australia, December 2-6, 2018, Revised Selected Papers 14, Springer International Publishing, pp. 39-54. https://doi.org/10.1007/978-3-030-21074-8_4
- [36] Otani, M., Nakashima, Y., Rahtu, E., Heikkilä, J., Yokoya, N. (2017). Video summarization using deep semantic features. In *Computer Vision-ACCV 2016: 13th Asian Conference on Computer Vision*, Springer International Publishing, pp. 361-377. https://doi.org/10.1007/978-3-319-54193-8_23
- [37] Tellex, S., Roy, D. (2009). Towards surveillance video search by natural language query. In *Proceeding of the ACM International Conference on Image and Video Retrieval*, pp. 1-8. <https://doi.org/10.1145/1646396.1646442>
- [38] Katz, B., Lin, J., Stauffer, C., Grimson, E. (2003). Answering questions about moving objects in surveillance videos. In *Proceedings of 2003 AAAI Spring Symposium on New Directions in Question Answering*, California.
- [39] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). Microsoft COCO: Common objects in context. In *Computer Vision-ECCV 2014: 13th European Conference*, Springer International Publishing, pp. 740-755. https://doi.org/10.1007/978-3-319-10602-1_48
- [40] Kim, J.A., Sung, J.Y., Park, S.H. (2020). Comparison of faster-RCNN, YOLO, and SSD for real-time vehicle type recognition. In *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, IEEE, pp. 1-4. <https://doi.org/10.1109/ICCE-Asia49877.2020.9277040>
- [41] Zhu, Y.Z., Yan, W.Q. (2022). Traffic sign recognition based on deep learning. *Multimedia Tools and Applications*, 81(13): 17779-17791. <https://doi.org/10.1007/s11042-022-12163-0>