U-FPNDet: A one-shot traffic object detector based on U-shaped feature pyramid module

Xiao Ke1,2 | Jianping Li1,2

1 Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing, College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China
2 Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou, China

Correspondence
Xiao Ke, Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing, College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China.
Email: kex@fzu.edu.cn

Abstract
In the field of automatic driving, identifying vehicles and pedestrians is the starting point of other automatic driving techniques. Using the information collected by the camera to detect traffic targets is particularly important. The main bottleneck of traffic object detection is due to the same category of targets, which may have different scales. For example, the pixel-level of cars may range from 30 to 300 px, which will cause instability of positioning and classification. In this paper, a multi-dimension feature pyramid is constructed in order to solve the multi-scale problem. The feature pyramid is built by developing a U-shaped module and using a cascade-method. In order to verify the effectiveness of the U-shaped module, we also designed a new one-shot detector U-FPNDet. The model first extracts the basic feature map by using the basic network and constructs the multi-dimension feature pyramid. Next, a pyramid pooling module is used to get more context information from the scene. Finally, the detection network is run on each level of the pyramid to obtain the final result by NMS. By using this method, a state-of-the-art performance is achieved on both detection and classification on commonly used benchmarks.

1 | INTRODUCTION
Deep learning has shown great vitality in computer vision and made a big splash in the fields of automatic driving, scene recognition, object detection and so on. Since AlexNet was proposed, the error rate in the ImageNet [1] dataset has been greatly reduced from the original 15% to 2% by using the deep learning method. Convolutional neural networks can effectively learn the features of images through a large amount of data. The method of extracting features is completely self-learning. Deep learning algorithm surpasses the previous ways by manual extraction feature, which has achieved a leading position. Due to its advantages in feature extraction, convolutional neural networks are widely used in computer vision fields such as image retrieval [2–4] and video understanding. However, convolutional neural networks have the scale invariance properties. This variation in scale which a detector needs to handle is enormous and presents an extreme challenge. There are two main strategies for solving such problems. One of both is to construct the image pyramid using the original image (scaling the images to different scales separately), and then use the detector to detect these images in every level. This method can only be applied in the testing stage, which consumes huge memory and time, because the detector has to process a series of images of different scales. Another method is to construct the feature pyramid from the original image and then predict it on every feature pyramid [5]. This method can be used in the training and testing phases. Compared with the first image pyramid strategy, the second image pyramid strategy requires less memory and computing resources while producing end-to-end results, and it can be easily integrated into the existing deep learning framework.

Although detectors with feature pyramids have achieved good results, there are some limitations in these detectors, since they simply construct the feature pyramid according to the inherent multi-scale, pyramidal architecture of the backbones which are actually used for the object detection task. For example, the SSD [6] detector simply uses two layers of the VGG16 [7] network to construct the feature pyramid; FPN [5] constructs the feature pyramid through a deep to shallow branch.
In general, the above method has the following two drawbacks. First, the feature map is not sufficient to completely handle the traffic target detection tasks in computer vision. Second, although they are built on the underlying backbone network, each feature pyramid only applies the information of one layer of the backbone network, which makes the feature maps that constitute the feature pyramids do not contain enough scale information. With these feature maps without rich scale information, it is difficult to solve the traffic detection problem. In short, the deep feature maps contain richer semantic information which is more suitable for classification tasks in traffic object detection. On the contrary, shallow feature maps have more detailed information, which is a better way to handle positioning tasks. So, the deep and shallow feature maps should be integrated together. Only in this way, we can improve the classification and positioning accuracy of the detectors and deal with the problems of multi-scale. In practice, the same category of traffic target with the same appearance may have different sizes, as shown in Figure 1.

In the KITTI dataset [8], we selected three images with high resolution to illustrate the problem. As can be seen from Figure 1, due to the visual changes between near-large and far-small, there are obvious scale differences among the same categories of objects. It varies greatly in shape and size, which cause difficulties in the detection of the detector. Therefore, how to solve the problem of scale change is the key to traffic detection.

In this paper, our goal is to build a more efficient feature pyramid to detect targets at different scales and to address the shortcomings of previous feature pyramid methods. First, we designed a U-shaped module to get more abundance scale information. Second, based on it, we reconstructed the feature pyramid and then predicted on each level feature map. In the training phase, we used L1 loss as a way to calculate the loss in the regression network and focal loss [9] was used to calculate the loss in the classification network. The structure of this paper is as follows: The related work is carried out in the Section 2, mainly describing how to solve the scale method in the near future. Section 3 mainly describes the model architecture, and Section 4 describes the model structure and some hyperparameter settings. Section 5 elaborates the experimental architecture and the experimental results. Section 6 summarizes the work of this paper, and we forecast future object detection problems in the field of automatic driving. The contributions of this paper are listed as the following:

1. We designed a new type of U-shaped module which can effectively solve the complex scale invariance problem in traffic scenarios.
2. We also designed a new one-shot detector U-FPNDet. This detector is based on RetinaNet and uses cascading U-shaped model to get richer multi-level feature pyramids, which are more effective than the features extracted by the previous feature pyramids.
3. Since the multi-level feature pyramids will lose some global contextual semantic information, we add the PPM pooling layer in semantic segmentation to U-FPNDet model to obtain a multi-level feature pyramid with global information.

2 | RELATED WORK

2.1 | Vehicle detection

In this chapter, we briefly introduce the relevant methods of vehicle detection. In the early work, the difference information of relative motion between target and background is used to detect vehicles, such as Gauss mixture model [10], linear Sigma model and so on. By establishing the background model, the distribution of background model will change when the moving targets appear, thus the location of vehicles could be located. This technology is widely used in vehicle detection. Apart from this, optical flow is also a commonly used method to solve the problem of vehicle detection [11, 12]. It is a common technique for gathering vehicles spatio-temporal information by simulating the pattern of motion of objects over time. In addition, the optical flow method combines symmetrical tracking and hand-extracted shape features to improve the detector performance. However, this approach does not distinguish between sub-category moving targets such as cars, buses, pedestrians etc. In summary, these methods require a lot of complex post-processing methods to improve the accuracy of the model, such as vehicle shadow detection and vehicle occlusion processing which is time consuming. It is hard to achieve real-time detection.
Next, some statistical learning methods based on manual feature extraction are also applied to directly detect vehicles from images. First, some feature description operators are used to describe the targets we are interested in, then, by training some classifiers, the objects in the image area are divided into different sub-categories, such as vehicles and non-vehicles. Features like HOG, SURF [13], Gabor and Haar-like [14, 15] are commonly used for vehicle detection followed by classifiers like SVM, artificial neural network and Adaboost algorithm. These features have limited the ability of feature representation, which is difficult to handle complex scenarios.

Recently, due to the development of convolutional neural networks, deep learning has shown strong vitality in understanding the abstract semantic information, and deep learning methods rapidly developed in the field of vehicle detection. How to design a detection network considering real-time detection and solve the problem of convolution kernel scale sensitivity are particularly important.

### 2.2 Scale sensitivity in CNNs

Scale space theory explains how to learn scale-invariant features and applies it to many fields of computer vision, such as object detection, pose estimation and instance segmentation. Learning scale-invariant features are important for identifying and locating objects, and many methods have been proposed for detecting multi-scale targets. These methods can be generally divided into two categories. The first one is the improvement of the original image, SNIP uses the images to construct the image feature pyramid, and predicts different scale targets on different levels of images. Finally, the results on each level pyramids are combined as the ultimate output. Compared with a single image, this method has the obvious improvement in classification and positioning. But, the disadvantages are also very obvious at the same time. Using this method, we must first construct an image pyramid, which would consume a certain amount of time. In addition, the convolutional neural network must be run on each level pyramid. If parallel processing is performed, the image of each level should be placed on different computing nodes, which would cause great space consumption. On the other hand, serial processing also consumes a lot of time and cannot be detected in real-time. The second method is to use the feature pyramid to solve the problem of scale diversity. The image feed into the convolutional neural network, a series of feature maps with different resolutions are obtained. Furthermore, the shallow feature map and the deep feature map are combined to construct the feature pyramid. The feature pyramid obtained in this way usually contains richer semantic information. For example, FPN [5], Mask-RCNN [16], RetinaNet [9], SSD [6] etc. The method can use the feature pyramids to predict the target with a large variation in the scale range. However, this feature pyramids are only constructed from a layer in the backbone network, and it is difficult to contain more abundant image information. In addition, if a target is only 30 × 30 resolution, then the resolution of the feature map is only 30 × 30, even if it is enlarged to 60 × 60. It is also difficult to be captured by a detector. Although the feature pyramids can effectively solve the problems caused by the scale sensitivity and does not consume time and space, it is difficult for the feature pyramids to mine deeper and richer information of the images, especially difficult to solve problems of small object detection.

### 3 THE ARCHITECTURE OF OBJECT DETECTOR

#### 3.1 General object detector

The generic object detector always contains the necessary two parts, one of is the feature extraction part, the other is the regression and classification part, where the feature extraction part is called the “body” of the detector, and the classifier and the regression are called the “head” of the detector. We can use f to represent the general feature extraction function, then the mathematical expression of the feature extraction is shown in Equation (1):

\[ f(x, b) = f^n o f^{n-1} ... f^1(x, b) \] (1)

In Equation (1), where x is the image area, \( b = (b_x, b_y, b_w, b_h) \) is the corresponding coordinate about x, and the process of feature extraction is obtained by u functions compound, where u is the number of convolution layers and the operation o is the convolution operation. The next task consists of the classifier and the regression which learned from a training sample \( \{g_x, b\} \). The task of the regression is to match each bounding box \( b_x \) with a real box \( g_x \). So that the regression task is to minimize location risk function \( \mathcal{R}_{lo}[f] \):

\[ \mathcal{R}_{lo}[f] = \sum_{i=1}^{N} L_{lo}(f(x_i, b_i), g_i). \] (2)

The classifier is a function \( h(x) \) that assigns an image patch x to one of the \( M+1 \) classes, where class 0 contains background and the remaining objects to detect. The goal of \( h(x) \) is to learn a distribution function \( b_k(x) = p(\gamma = k \mid x) \) from the image area to the category, where \( \gamma \) is category label of the object. Given a training sample \( (x_i, y_i) \), the target of the classifier is to minimize classification risk function \( \mathcal{R}_{cl}[h] \):

\[ \mathcal{R}_{cl}[h] = \sum_{i=1}^{N} L_{cl}(h(x_i), y_i), \] (3)

where \( L_{cl} \) is the classic cross-entropy loss in Equation (3).

#### 3.2 U-FPNDet detector

In this section, we will describe the specific details of the U-FPNDet, the structure of which is shown in Figure 2. In Figure 2, our model consists of three sub-networks. In
We divided our model into three subnetworks. First, we extracted the features by backbone network and get the feature maps. Second, we fed the features maps into U shape feature pyramid network and the multi-level features can be generated. Then the PPM layers are added on the multi-level features that can make more effective features. Finally, on each feature maps, we run the classification and location networks to obtain final results.

Phase one, the image first passes through the backbone network. Taking VGG16 as an example, we use the feature maps of block 3, 4 and 5. The feature map of block 4 is sampled two times the feature map of block 5 is sampled four times and added to block 3, respectively so that the feature map of each layer in the backbone network can be fully utilized.

After phase one, we obtain the basic feature map. Then the basic feature map passes through the U-shaped modules of phase two. These U-shaped modules are connected in series by a cascade. Each U-shaped module produces a feature pyramid, then the feature pyramids of each dimension are aggregated together, so that the multi-dimension feature pyramid will contain more rich scale information, and the second phase of the final output is the feature pyramid with rich scale information. Finally, the aggregated feature pyramid is sent to the third-stage detection networks, which mainly includes classification and regression sub-networks. The classification and regression sub-networks must be run once on each layer of the pyramid. So, for a total of $M \times N$ layer feature pyramids, we have to run $M \times N$ times. Since the detection network uses the detection branch of RetinaNet, which is composed of a very light full-convolution neural network, it takes very little time. The sub-network used for classification generates $W \times H \times KA$ feature maps for predictions, where $W$, $H$, $K$, $A$ represents the width, height, number of categories and number of anchors. Similarly, the regression subnetwork produces $W \times H \times 4A$ predicted values. That is, each pixel point on the feature map generates a coordinate bounding box. The final output is then obtained using a non-maximum suppression algorithm.

4 | Model Structure

4.1 | Backbone network

The backbone network is the first step in extracting the image features. In order to generate a richer feature map for the second stage, we use the feature maps of the last three blocks and aggregate them together. The basic feature map obtained in this way contains both the details of the shallow network and the abstract information of the deep network. Taking VGG16 as an example, the specific design structure of phase one is shown in Figure 3.

As shown in Figure 3, when a batch of images data is inputted, the images pass through the five-block of VGG16,
and finally we extract the feature map from the three modules of three-block, four-block, and five-block. Then in order for each feature map to have the same dimension, we add a \( W \times H \times K \times 1 \) convolution kernel with each feature map. Finally, the feature map is upsampled and normalized to the same scale. The aggregated feature map has abstract information of deep feature maps, which are more conducive to building feature pyramids. The process of feature extraction of the backbone network can be expressed by the following relation.

\[
\begin{align*}
Y_1 &= W'_1(x) + b_1, \\
Y_2 &= W'_2(Y_1) + b_2, \\
Y_3 &= W'_3(Y_2) + b_3, \\
F &= c_1(Y_1) + c_2(Y_2) \times 2 + c_3(Y_3) \times 4,
\end{align*}
\]

where \( x \) is image data, \( Y_1, Y_2 \) and \( Y_3 \) are feature map output by three blocks, \( c_1, c_2 \) and \( c_3 \) are three \( 1 \times 1 \) convolution kernel modules, and final feature map is represented by \( F \).

### 4.2 Construct feature pyramid

The feature map generated by the backbone network first passes through several U-shaped modules, each of which generates a single-level feature pyramid of different scales, and finally aggregates the single-level feature pyramid into a multi-dimensional feature pyramid. This feature pyramid contains rich multi-scale information. The structure of the U-shaped module is as shown in Figure 4.

The basic feature map passes through \( N \) convolutional layers and \( N \) pooling layers, and after each pooling layer, its size becomes half of them before the feature map. Then the feature map is upsampled to obtain \( N \) levels of feature maps. Finally, by cascading \( M \) U-shaped modules, an \( M \)-dimensional multi-scale pyramid is obtained. In practice, we take the value of \( N \) as 3 and the value of \( M \) as 2.

### 4.3 Pyramid pooling module layers

It can be roughly considered that the receptive field is the size of the context information. In many object detection networks, we attach great importance to obtaining global information. Without full use of the context information of the scene, there will be the following problems (as shown in the figure above): (1) mismatched relationship, (2) confusion categories, and (3) inconspicuous classes. In short, PPM [17] is a relatively good way to make full use of global information. This idea of retaining global information is actually very similar to Atrous Spatial Pyramid Pooling (ASPP). Intuitively, this multi-scale pooling can indeed retain global information at different scales, and it can retain global context information more than ordinary single pooling.

In our model, the feature pyramid generated in the second stage will lose some global information due to the U-shaped module. Although more abstract features can be obtained, it is based on the loss of certain full-text information. Therefore, we use PPM [17] to get richer global information. In each feature map from pyramid, we push PPM on it and the output is the same as the input layer. We can obtain the different pyramid feature maps from original which have more global information. The structure of the U-shaped module is as shown in Figure 5.

First, we pool the feature maps to the target size, and then perform a \( 1 \times 1 \) convolution on the pooled results to reduce the channel to \( 1/N \), where \( N \) is 4. Next, use bilinear interpolation upsampling to obtain the same size of the original feature map for each feature map in the previous step, and then perform concatenation of the original feature map and the upsampled feature map, according to the channel dimension. The resulting channel is twice that of the original feature map, and finally the channel is reduced to the original channel by \( 1 \times 1 \) convolution. The final feature map is the same as the original feature map size and channel.

### 4.4 Detection network

After the cascaded U-shaped module, a feature pyramid of \( M \times N \) dimension is generated, and then at each layer, the detection network consists of a regression subnetwork and a classification subnetwork. Each subnetwork consists of a fully convolutional neural network. For any point on the feature map, \( K \) anchor boxes are generated. The learning task of the classification network is to learn the category of the \( K \) anchors by focal loss, and the task of regression network.

---

**FIGURE 4** U-shaped module structure

**FIGURE 5** Four different pyramid scales are used in the model. The number of layers of the pyramid pooling module and the size of each layer can be modified. The pyramid pooling module in the paper is four layers, and the size of each layer is \( 1 \times 1, 2 \times 2, 3 \times 3, 6 \times 6 \).
In the classification subnetwork, we use focal loss as the loss function during training. Focal loss is designed to solve the problem of class imbalance in a one-shot detector. Unlike the one-shot object detector, the two-stage detector has a step which generates several candidate areas, which are proposed by region proposed network or select search method. Therefore, the ratio of positive and negative samples will not differ too much while the one-shot object detector will generate a lot of useless anchors during the training phase. But the real targets are very rare, which will cause many anchor boxes to be divided into the negative sample (e.g., 1:10 000). Focal loss is proposed to solve this problem which is improved version of the cross-entropy function:

\[
CE(p, y) = \begin{cases} 
-\log(p) & y = 1 \\
-\log(1 - p) & y = 0.
\end{cases}
\]

In the above relationship, the value of \( y \) is 0 or 1 and \( p \in [0, 1] \) is the model’s estimated probability of the class with label \( y = 1 \). For notational convenience, we define \( p_t \):

\[
p_t = \begin{cases} 
p & y = 1 \\
1 - p, & y = 0
\end{cases}
\]

and rewrite \( CE(p_t, y) = CE(p_t) = -\log(p_t) \).

The easiest way to solve the problem of class imbalance is to add a factor of balance weight before the cross-entropy function. In the case of a large number of sample categories, the improved cross-entropy function can simply distinguish between positive and negative samples. It does not differentiate between easy and hard examples (Hard samples, refers to the model is difficult to learn a certain category of sample, which is difficult to learn their feature.) The solution for the hard samples is that using \( 1 - p_t \) to indicate the probability of hard sample. Therefore, on the basis of the above, the factor of \((1 - p_t)^\gamma\) that distinguishes the difficult samples can be added to the cross-entropy function. Finally, focal loss can be defined as:

\[
FL(p_t) = -\alpha(1 - p_t)^\gamma \log(p_t). 
\]

In Equation (7), \( \alpha = 0.25 \), \( \gamma = 2 \).

Similarly, in the regression subnetwork, we assign a best matching real box to each anchor. The output of the regression subnetwork is the difference between the anchor and the ground truth box.

\[
\begin{align*}
\Delta x_i &= \frac{(g_{xi} - anchor_{xi})}{anchor_{xi}} \\
\Delta y_i &= \frac{(g_{yi} - anchor_{yi})}{anchor_{yi}} \\
\Delta w_i &= \log(\frac{g_{wi}}{anchor_{wi}}) \\
\Delta h_i &= \log(\frac{g_{hi}}{anchor_{hi}})
\end{align*}
\]

And on rewriting Equation (8), \( r_i = (\Delta x_i, \Delta y_i, \Delta w_i, \Delta h_i) \), finally, we get the Equation (9).

\[
RL(r_i, p_t) = \sum_{i=1}^{n} abs(r_i - p_i)
\]

where \( abs \) is \( L_1 \) loss function in our network and \( p_i = (out_{x_i}, out_{y_i}, out_{w_i}, out_{h_i}) \) is the output of regression subnetwork.

5 | EXPERIMENT

To verify the validity of our model, we performed experiments on three traffic datasets: the KITTI [8] dataset, the BDD100K [18] dataset, and the LSVH [19] dataset. The experimental software environment is Linux Ubuntu 16.04 operating system, python 3.5, and the hardware environment is GPU Tesla P100.

5.1 | Experimental Setting

BDD100k Dataset: The Berkeley AI Lab (BAIR) has released the largest and most diverse public driving dataset- BDD100K. The BDD100K dataset contains 100 000 segments of HD video, each video is about 40 s, 720p, 30 Fps. The database is informative and rich, including different time, lighting, weather and geographical location, and even contains GPS and IMU with timestamp information. In order to understand the distribution and location of targets on the road, this dataset provides the bounding boxes of objects in 100k key frames.

KITTI Dataset: The KITTI dataset, co-founded by the Karlsruhe Institute of Technology in Germany and the Toyota Institute of Technology in the United States, is the most famous dataset for computer vision algorithms in the world’s largest autopilot scenario. The dataset is used to evaluate the performance of computer vision technology such as stereo image, optical flow, visual, 3D object detection and tracking in the vehicle environment. In the KITTI dataset, there are three items in the field of object detection: vehicle detection, pedestrian detection and cyclist detection. It contains 7481 training pictures and 7518 test pictures. The data set is divided into three levels based on size, clarity and occlusion: easy, medium and difficult.

LSVH Dataset: Vehicle detection on highways is an important aspect to measure the effectiveness of autonomous driving technology. The dataset is classified into three categories (car, bus, van) in two scenes (sparse and crowded). These two scenes contain 1409 and 12 979 images. We divide these two scenes into training and test data on average with a 7:3 ratio.
### TABLE 1  
U-FPNDet is compared with other models (BDD100K dataset experimental results)

| Method       | Input size  | mAp   | Time [s] |
|--------------|-------------|-------|---------|
| SSD [6]      | 512 × 512   | 14.1  | 0.05    |
| RefineDet [19]| 512 × 512   | 17.4  | 0.041   |
| RetinaNet [9]| 608 × 1024  | 27.3  | 0.09    |
| CFEv2 [20]   | 512 × 512   | 19.1  | 0.052   |
| WL0D [21]    | 512 × 512   | 28.7  | 0.06    |
| YOLO v3 [22] | 512 × 512   | 25.8  | 0.05    |
| Ours         | 512 × 512   | 30.1  | 0.08    |

### TABLE 2  
The accuracy compared with CFENet800-MS model

| Category name | CFENet [20] | Ours  |
|---------------|-------------|-------|
| Bike          | 20.51       | 28.62 |
| Bus           | 50.43       | 35.50 |
| Car           | 51.29       | 58.66 |
| Motor         | 16.73       | 17.13 |
| Person        | 29.08       | 44.66 |
| Rider         | 23.86       | 21.98 |
| Traffic light | 15.27       | 21.93 |
| Traffic sign  | 37.54       | 31.45 |
| Truck         | 52.23       | 40.90 |
| Mean          | 29.69       | 30.10 |

### 5.2  
**Experimental result**

#### 5.2.1  
**BDD100k dataset**

We use our model to do experiments on the BDD100k dataset, compared with existing experimental results, as shown in Table 1.

As shown in Table 1, U-FPNDet has achieved good results in performance and speed compared with other models. In particular, taking the SSD model as an example, the mAp of the SSD model is only 14.1, while the mAp of our model is 30.1. The results show that our model is more effective than state-of-the-art methods, which can prove advancement of our model. We compare our model with the CFENet model, we can get the results shown in Table 2.

In Table 2, we can see that our model achieved an Ap value of 58.66% in the car category and reached the highest in ten categories, indicating that our model has the best effect in the car category. Since the number of samples is only 179 in the train category, the Ap values of both models are zero.

We show the visualization results for each category of mAp in Figure 6. As shown in Figure 6, the mAp of car category is 57%, and the value is the highest among all categories, because the number of training samples in the car category is the largest. Therefore, in the case of a large number of samples, this model has better discriminative power for the car category and can obtain better results in it.

#### 5.2.2  
**KITTI dataset**

We use our model to do experiments on the KITTI dataset, compared with existing experimental results on the KITTI website, as shown in Table 3. We compared our models with other 16 published state-of-the-art methods in this experiment. It is clear that our model has achieved the highest accuracy in the moderate case and faster speed in the car category. In the three levels of simple, medium and difficult, the Ap values of the car are 91.3%, 89.7% and 81.8%, respectively. On the other side, although SSD and YOLO are fast, the accuracy is low. Our method can achieve the same speed as YOLO and SSD and the accuracy is still much better than these two detectors. For the computational efficiency among the two-stage deep learning-based detectors, our model takes only 1/33 of faster-RCNN.

#### 5.2.3  
**LSVH dataset**

In the LSVH dataset, we use the Crowded dataset and get the experimental results of our model, as shown in Figure 7. In Figure 7, we show the visualization results of each category of mAp. Since the size of the truck is much larger than the car and bus, the mAp value is 68.1%. Although the scale of the car is small, the result of mAp = 58.9% is obtained. We also get the accuracy of other models in this dataset, as shown in Table 4.

In Table 4, we use YOLO, YOLOv2, SInet and other models to compare with our model. In the car and truck categories, we have achieved 55.1% and 68.1% mAp, which shows that our model has strong robustness, and the model is better for small objects. We also use EfficientDet [40] method and compared with our result, our method is better in car and van category.

### 5.3  
**Parameter analysis**

In this section, we analyse the parameters of the model, statistically the parameters of each part, and visualize them in the paper. The model in total includes five parts and the number of parameters in each part of the model is shown in Table 5.

As shown in Table 5, the number of parameters of the model is measured in millions. The model has a large proportion in extracting features and constructing feature pyramids, but it has a small proportion in the detection part.

As shown in Figure 8, although the structure of the model is complex, most of the parameters are concentrated in the base network of feature extraction and in the part of building U shaped pyramid features, which also shows that the running time of the model is low and can achieve real-time effect.

### 5.4  
**Analysis of results**

In this section, we conclude that constructing a multi-level feature pyramid can effectively solve the scale invariance on the KITTI data set and our model has achieved good results under
KE AND LI

FIGURE 6 mAP for each category in the BDD100K dataset

FIGURE 7 PR curves for each category on LSVH dataset

the car category on the BDD100K dataset, because the scenes contained in the data set are more complex. The detection effects of other models are not very good for which mAP are only around 20%, while the mAP of our model is about 30%. Similarly, on the LSVH data set, we are the best in the single category of cars and vans. This shows that our model is effective for traffic target detection. In order to illustrate the effectiveness of the model more intuitively, we display the results of the model on different datasets in Figure 9.

In Figure 9, we divide the images into three parts, which are based on three different subsets according to a different dataset. The first and second rows are the results of the BDD100k dataset, and the third and fourth rows are the results from KITTI dataset. The fifth and sixth lines are the result of the
TABLE 3  U-FPNDet is compared with other models (KITTI dataset experimental results)

| Method        | Time [s] | Car E | Car M | Car H | Pedestrian E | Pedestrian M | Pedestrian H | Cyclist E | Cyclist M | Cyclist H |
|---------------|----------|-------|-------|-------|--------------|--------------|--------------|-----------|-----------|-----------|
| Kinematic3D[23] | 0.12     | 89.7  | 71.7  | 55.0  | null         | null         | null         | null      | null      | null      |
| DPM[24]       | 8        | 82.2  | 66.7  | 49.0  | 59.2         | 43.3         | 38.1         | 41.6      | 27.7      | 24.6      |
| SubCat[25]    | 0.7      | 84.1  | 75.5  | 59.7  | 54.7         | 42.3         | 38.0         | null      | null      | null      |
| D4LCN[26]     | 0.03     | 90.3  | 83.7  | 65.4  | 59.6         | 43.5         | 37.1         | 65.3      | 42.9      | 36.3      |
| AS3[27]       | 0.8      | 79.0  | 81.2  | 70.6  | null         | null         | null         | null      | null      | null      |
| YOLOv2[28]    | 0.02     | 26.7  | 14.3  | 11.0  | 15.4         | 11.5         | 9.7          | 0.2       | 0.1       | 0.1       |
| SSD[6]        | 0.06     | 83.9  | 67.2  | 79.2  | 23.1         | 16.3         | 16.1         | null      | null      | null      |
| Faster RCNN[29]| 2        | 89.0  | 83.2  | 72.6  | 80.0         | 66.2         | 61.1         | 72.4      | 63.0      | 55.0      |
| CompACT[30]   | 1        | null  | null  | null  | 70.9         | 58.1         | 52.3         | null      | null      | null      |
| DeepParts[31] | 1        | null  | null  | null  | 71.5         | 58.2         | 52.3         | null      | null      | null      |
| FilteredICF[32]| 2       | null  | null  | null  | 69.8         | 56.5         | 50.3         | 70.4      | 58.7      | 51.8      |
| 3D-GCK[33]    | 1        | 80.2  | 89.6  | 68.1  | null         | null         | null         | null      | null      | null      |
| MonoGRNet[34] | 0.4      | 88.7  | 77.9  | 63.3  | 85.7         | 75.0         | 68.3         | 84.1      | 75.3      | 65.3      |
| MS-CNN[35]    | 0.4      | 90.0  | 89.0  | 76.1  | 85.7         | 75.0         | 68.3         | 84.1      | 75.3      | 65.3      |
| subCNN[36]    | 5        | 90.7  | 88.6  | 79.8  | 79.1         | 66.1         | 61.3         | 74.4      | 62.0      | 54.8      |
| SqueezeDet[37]| 0.017    | 90.2  | 84.7  | 73.9  | 82.9         | 75.4         | 72.1         | 77.1      | 68.3      | 65.8      |
| Ours          | 0.06     | 91.3  | 89.7  | 81.8  | 75.7         | 72.1         | 68.4         | 74.5      | 68.4      | 62.1      |

TABLE 4  Accuracy of each model on the LSVH dataset

| Model         | Car | Bus | Van | Time [s] |
|---------------|-----|-----|-----|----------|
| YOLO [38]     | 3.87| 8.35|10.32| 0.03     |
| YOLO v2 [28]  | 17.39|21.55|40.42| 0.03     |
| Faster RCNN[29]| 26.08|24.55|40.24| 0.31     |
| MS-CNN [35]   | 51.74|32.95|54.26| 0.23     |
| SInet [39]    | 56.8 | 55.78|62.38| 0.2      |
| EfficientDet [40]| 57.2 | 58.7| 67.6 | 0.06     |
| Ours          | 58.9 | 45.0 | 68.1 | 0.06     |

TABLE 5  The parameter analysis in each part of our model

| Parts of the model | Parameters [M: million] |
|--------------------|-------------------------|
| Backbone network   | 20.48                   |
| U_FPN network      | 11.91                   |
| PPM layers         | 2.36                    |
| Classification model| 3.01                     |
| Regression model   | 3.03                    |

LSVH dataset detection. At the same time, they can be divided into day and night according to the time order. In addition, the images can be divided into two different subsets according to the weather. According to the results, our model can achieve excellent detection in different conditions of weather.

FIGURE 8  The number of parameters in each part of the model

It shows that our model has a good effect to deal with different scenarios.

6  | CONCLUSION

We proposed a new detector U-FPNDet, which can extract more abundant scale features from the original image to solve the scale-invariant problem of convolutional neural networks. It has good scalability and adaptability, and can effectively detect vehicle targets of different scales. Our proposed model can achieve advanced results in a variety of datasets, for examples, in BDD100k dataset, we get the mAp as 30.1%. In KITTI dataset, we achieve the Ap of car which is 87.6%. In LSVH dataset, the Ap of car is 58.9%. All results illustrating that our model has
good generalization performance, providing a simple and operational strategy for multi-scale research.

ACKNOWLEDGEMENTS
This work is partially supported by the National Natural Science Foundation of China under Grant Nos. 61972097, 61672159 and 61672158, the Technology Guidance Project of Fujian Province under Grant No. 2017H0015, the Natural Science Foundation of Fujian Province under Grant No. 2018J1798, the Fujian Natural Science Funds for Distinguished Young Scholar under Grant No.2015J06014, the University Production Project of Fujian Province under Grant No. 2017H6008, the Fujian Collaborative Innovation Center for Big Data Application in Governments, the Fujian Engineering Research Center of Big Data Analysis and Processing.

REFERENCES
1. Kornblith, S., Shlens, J., Le Q, V.: Do better imagenet models transfer better? In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, pp. 2661–2671 (2019)
2. Öztürk, Ş.: Stacked auto-encoder based tagging with deep features for content-based medical image retrieval. Expert Syst. Appl. 161, 113693 (2020)
3. Öztürk, Ş.: Image inpainting based compact hash code learning using modified U-Net. In: 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). Istanbul, Turkey, pp. 1–5 (2020)
4. Öztürk, Ş., Ozkaya, U.: Skin lesion segmentation with improved convolutional neural network. J. Digit. Imaging 33, 958–970 (2020)
5. Kirillov, A., et al.: Panoptic feature pyramid networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, pp. 6399–6408 (2019)
6. Chen, X., Yu, J., Wu, Z.: Temporally identity-aware SSD with attentional LSTM. IEEE Trans. Cybernetics 50(6), 2674–2686 (2020)
7. Liu, Z., et al.: Improved kiwifruit detection using pre-trained VGG16 with RGB and NIR information fusion. IEEE Access 8, 2327–2336 (2019)
8. Trivedi, M.M.: Attention monitoring and hazard assessment with biosensing and vision: Empirical analysis utilizing CNNs on the kiti dataset. In: 2019 IEEE Intelligent Vehicles Symposium (IV). Paris, France, pp. 1673–1678 (2019)
9. Afif, M., Ayachi, R., Said, Y., et al.: An evaluation of RetinaNet on indoor object detection for blind and visually impaired persons assistance navigation. Neural Process. Lett. 51(10), (2020)
10. Daemi, A., Kodamana, H., Huang, B.: Gaussian process modelling with Gaussian mixture likelihood. J. Process Control 81, 209–220 (2019)
11. Wu, X., et al.: Vehicle detection of multi-source remote sensing data using active fine-tuning network. ISPRS J. Photogramm. Remote Sens. 167, 39–53 (2020)
12. Song, H., Liang, H., Li, H., et al.: Vision-based vehicle detection and counting system using deep learning in highway scenes. European Transp. Res. Rev. 11(1), 1–16 (2019)
13. Ansari, S.: A review on SIFT and SURF for underwater image feature detection and matching. In: 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICEECT). Coimbatore, India, pp. 1–4 (2019)
14. Ai, J., et al.: Multi-scale rotation-invariant Haar-like feature integrated CNN-based ship detection algorithm of multiple-target environment in SAR imagery. IEEE Trans. Geosci. Remote Sens. 57(12), 10070–10087 (2019)
15. Lin, Y.N., et al.: Fast Iris localization using Haar-like features and AdaBoost algorithm. Multimedia Tools Appl. 79(45), 34339–34362 (2020)
16. Huang, Z., et al.: Mask scoring r-cnn. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, pp. 6409–6418 (2019)
17. Fang, H., Lafarge, F.: Pyramid scene parsing network in 3D: Improving semantic segmentation of point clouds with multi-scale contextual information. ISPRS J. Photogramm. Remote Sens. 154, 246–258 (2019)
18. Yu, F., et al.: Bdd100k: A diverse driving dataset for heterogeneous multi-task learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Seattle, WA, USA, pp. 2636–2645 (2020)
19. Zhang, S., et al.: Single-shot refinement neural network for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Seattle, WA, USA, pp. 4203–4212 (2018)
20. Zhao, Q., et al.: CFENet: An accurate and efficient single-shot object detector for autonomous driving. arXiv preprint arXiv:1806.09790 (2018)
21. Han, Y., et al.: Wasserstein loss-based deep object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, pp. 998–999 (2020)
22. Tian, Y., et al.: Apple detection during different growth stages in orchards using the improved YOLO-V3 model. Comput. Electron. Agric. 157, 417–426 (2019)
23. Brazil, G., et al.: Kinematic 3D object detection in monocular video. In: European Conference on Computer Vision. Springer, Cham, pp. 135–152 (2020)
24. Pepik, B., et al.: Multi-view and 3d deformable part models. IEEE Trans. Pattern Anal. Mach. Intell. 37(11), 2232–2245 (2015)
25. Ohm-Bar, E., Trivedi, M.M.: Fast and robust object detection using visual subcategories. In: Proceedings of the IEEE Conference on Computer Vision and
Vision and Pattern Recognition Workshops, Columbus, OH, USA, pp. 179–184 (2014)

26. Ding, M., et al.: Learning depth-guided convolutions for monocular 3D object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, pp. 1000–1001 (2020)

27. Gustafsson, F., Linder-Norén, E.: Automotive 3D object detection without target domain annotations. Master of Science Thesis, Linköping University (2018)

28. Xu, K., et al.: A dedicated hardware accelerator for real-time acceleration of YOLOv2. J. Real-Time Image Process. 2020

29. Fang, F., et al.: Combining faster R-CNN and model-driven clustering for elongated object detection. IEEE Trans. Image Process. 29, 2052–2065 (2019)

30. Cai, Z., Saberian, M., Vasconcelos, N.: Learning complexity-aware cascades for deep pedestrian detection. In: Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, pp. 3361–3369 (2015)

31. Tian, Y., et al.: Deep learning strong parts for pedestrian detection. In: Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, pp. 1904–1912 (2015)

32. Zhang, S.; Benenson, R.; Schiele, B.: Filtered channel features for pedestrian detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, pp. 1751–1760 (2015)

33. Gählert, N., et al.: Single-shot 3D detection of vehicles from monocular RGB images via geometry constrained keypoints in real-time. arXiv:2006.13084 (2020)

34. Qin, Z., Wang, J., Lu, Y.: Monogrnet: A geometric reasoning network for monocular 3D object localization. Proceedings of the AAAI Conference on Artificial Intelligence, Hawaii, USA, 33(01), 8851–8858 (2019)

35. Cai, Z., et al.: A unified multi-scale deep convolutional neural network for fast object detection. In: Leibe B., Matas J., Sebe N., Welling M. (eds), European Conference on Computer Vision, Lecture Notes in Computer Science, vol 9908, pp. 354–370. Springer, Cham (2016)

36. Xiang, Y., et al.: Subcategory-aware convolutional neural networks for object proposals and detection. In: 2017 IEEE winter conference on applications of computer vision (WACV), Honolulu, HI, USA, pp. 924–933 (2017)

37. Wu, B., Wan, A., Iandola, F.: Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: CVPR Workshops, Honolulu, HI, USA, pp. 446–454 (2017)

38. Chen, R.C.: Automatic license plate recognition via sliding-window darknet-YOLO deep learning. Image Vision Comput. 87, 47–56 (2019)

39. Hu, X., et al.: SiNet: A scale-insensitive convolutional neural network for fast vehicle detection. IEEE Trans. Intell. Transp. Syst. 20(3), 1010–1019 (2019)

40. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, pp. 10781–10790 (2020)

How to cite this article: Ke X, Li J. U-FPNDet: A one-shot traffic object detector based on U-shaped feature pyramid module. IET Image Process. 2021;15: 2146–2156. https://doi.org/10.1049/ipr2.12182