U19-Net: a deep learning approach for obstacle detection in self-driving cars

Albert Aarón Cervera-Uribe · Paul Erick Méndez-Monroy

Accepted: 3 March 2022 / Published online: 7 April 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
Development of self-driving cars aims to drive safely from one point to another in a coordinated system where an on-board system should react and possibly alert drivers about the driving environments and possible collisions that may arise between drivers and obstacles. There are many deep learning approaches available for obstacle detection especially convolutional neural networks (CNNs) with improvement accuracy, and encoder–decoder networks are CNNs with a current attraction for researchers mainly because these models provide better results than classical statistical models for image segmentation and object classification tasks. This work proposes U19-Net an encoder–decoder deep model that explores the deep layers of a VGG19 model as an encoder following a symmetrical approach with an U-Net decoder designed for pixel-wise classifications. The U19-Net has end-to-end learning successfully effectiveness for the vehicle and pedestrian detection within the open-source Udacity dataset showing an IoU score of 87.08 and 78.18%, respectively. The proposed U19-Net is compared with five recent CNN networks using the AP metric, obtaining near results (less than 5%) for the faster R-CNN, one of the most commonly used networks for object detection.

Keywords Deep learning · Self-driving cars · Object detection · Convolutional neural networks

1 Introduction
The development of Autonomous Vehicles (AVs) aims directly to transform the way we commute from one place to another and to transform our overall wellness, potentially since they could drastically reduce road accidents and thus save humans lives and road damage while increasing productivity by reducing traffic congestion and helping commute those who are unable to drive.

Every year approximately 1.5 million people die as a result of a road accident, with more than half of those deaths distributed among pedestrians, cyclists, and motorcyclists (WHO 2018); and between 20 to 50 million people suffer non-fatal injuries with many incurring a disability as result from a traffic crash accident.

According to the National Highway Traffic Safety Administration, just in the USA, an estimation of 94% of fatal crashes is caused by human error either by speeding, drowsiness, alcoholic consumption, or just distracted driving (NHTSA et al 2017). All of this reflects the need for strategies to reduce these events, this is where self-driving cars become apparent since they are meant to drive us from point A to point B safely and thus reduce traffic accidents.

When we drive, walk, and do many other everyday tasks, we use vision as the main feature for the perception of our environment. We have the visual ability to recognize with near-perfect precision the scene and objects of interest present in our field of view. For example, it is easy for us to see and differ between types of vehicles, persons or maybe to interpret the meaning of traffic signs just from plain sight.

In the case of self-driving vehicles, vision is also one of the main features within the perception system. So, the ability to perceive the vehicle’s environment at any moment is one of the main challenges. Environmental and meteorological conditions such as lighting, fog, and rain are continually
changing as a consequence also what a camera, lidar, or ultrasonic sensor can detect, added to the presence of static and dynamic objects in the scene. It showcases the need for more robust and reliable sensing systems.

In short, the perception system is responsible for answering (1) Where am I? and (2) What’s around me? in self-driving cars. The perception system is divided into other sub-systems responsible for object detection, localization, mapping of static and moving obstacles, and tracking of road scene elements such as traffic signs and lanes. Being the task of object detection one of the main utilities for the derivation of some others as mapping or control.

Numerous studies have addressed the detection of objects on the roads, presenting detection methods based on computer vision and machine learning techniques such as those discussed in Wang and Cai (2015), Seenouvong et al (2016), Brunetti et al (2018), Jordan and Mitchell (2015) with a wide area of research focused on support vector machine (SVM) (Velazquez-Pupo et al 2018), Boosting algorithms (Wang and Cai 2015) and artificial neural networks (ANN) Amine and Djoudi (2019). Follow a general pipeline consisting of (1) image acquisition, (2) feature extraction, and (3) classification.

While this pipeline based on handcrafted features has been extensively used in the literature, the deep learning models automatize feature extraction as a process completed by deep classifiers comprehending several processing layers and multiple nonlinear processes taking input images and computing its features with different abstraction levels (Brunetti et al 2018).

One of the most widely used network in deep learning for vision is Convolutional neural network (CNN) with more than 27000 research articles for vehicle and/or pedestrian detection in the last 4 years (Scholar 2021). It has outstanding results in most visual-based classification tasks with an image as input. It can learn hierarchical feature maps that characterize variations in visual data. Their common output is a label set of the detected classes (objects). In the case of objects detection for autonomous vehicles, the desired output should include the class label prediction and its precise localization defined at each pixel or bounding box for each object of interest within the input image.

This paper contributes to the literature in the development of a novel convolutional neural network, U19-Net, for object detection where road scenes and unpredictable traffic pose such a challenging environment. The proposed U19-Net model is designed for pixel-wise classifications in urban driving scenarios, with effort focused on the task of cars, trucks, and pedestrians detection within video frames from a monocular camera mounted on the dashboard of a vehicle.

Experiments were conducted evaluating the performance of U19-Net and a well-known encoder–decoder network measuring its performances with IoU and AP score using an open-source dataset. The proposed network improves the location of the objects useful for control and trajectory planning purposes. It also improves detection on overlapping and occluded objects of interest.

The rest of the paper is organized as follows: Related work on object perception can be found in sect. 2, whereas sect. 3 deepens in the architectural design of the proposed model. Sect. 4 details the classification framework followed by data preparation and the training/testing processes. Sect. 5 presents the training and testing results from experiments with vehicle and pedestrians sets followed by a discussion of findings and future work within this research at sects. 6 and 7.

2 Related work

Convolutional neural network (CNN) has a lot of application in the field of computer vision, several recent surveys focusing on object detection (Brunetti et al 2018)(Galvao et al 2021)(Chen et al 2021) where 65% of the articles reviewed use either hybrid or end-to-end CNN. Galvao et al (2021) and Brunetti et al (2018) performed a review on both machine learning and deep learning techniques for vehicle and pedestrian detection while Chen et al. Chen et al (2021) provided a guide of some deep learning techniques making a comparative analysis using the accuracy precision (AP) metric of the networks with the best results.

By analyzing the previous reviews, it is possible to detect four main architectures (Yamashige and Aono 2019) for object detection using CNN described as region CNN (R-CNN)(Chen et al 2016), region proposal network (RPN)(Xu et al 2020)(Sang et al 2017)(Girshick 2015), single-shot detector (SSD) (Liu et al 2016)(Fu et al 2017), and You Only Look Once (YOLO)(Liu et al 2018)(Amitha and Narayanan 2021).

2.1 Convolutional neural network techniques

The work of Tome et al (2016) proposed a pipeline with locally decorrelated channel features (Nam et al 2014) as region proposal algorithm and a fine-tuned deep convolutional neural network-based either in Krizhevsky et al (2012) or Szegedy et al (2015) which in its training procedure exploits both positive and negative annotated image regions proposing a greedy algorithm based on color histograms. In terms of accuracy, the CNN yield a miss rate of 0.199 and 0.197, respectively.

With a different approach, Li et al (2017) presented a model that uses neural features within the application of a fully convolutional neural network training an AdaBoost detector per layer by fine-tuning its model with bounding boxes labels. The results produced a miss rate of 0.1879 with
a single detector and 0.1650 with a combination of two detectors.

In matter of object detection with regards to vehicle detection, Cai et al (2016) proposes the multi-scale CNN (MS-CNN) consisting of a proposal sub-network and a detection network which both of them use an end-to-end learning process. Its approach uses feature upsampling as an alternative to input upsampling by introducing a deconvolution layer that increases the resolution of its feature maps, allowing small objects to have strong responses in terms of regions.

In the case of R-CNN, selective search methods are presented (Uijlings et al 2013) to extract regions within the image that are likely to have an object and passed in a pre-trained CNN to extract feature vector, examples of these are the R-CNN (Girshick et al 2014)(Chan et al 2016) achieved an mAP of 53.7% in the PASCAL VOC dataset (Everingham et al 2010) with a reduction of regions (approximately 5000) per image (224 x 224) respect to R-CNN, but the speed to detect objects is 0.021 FPS (frames per second).

The RPN meta architecture was introduced by Girshick (2015) a second version of the R-CNN, called Fast R-CNN eliminated the disk storage creating a high-resolution convolution feature map instead of each region proposal and designing a single-stage pipeline by using multitask learning. Faster R-CNN was implemented by Ren et al (2015) introducing a RPN formed by VGG-16 model (Simonyan and Zisserman 2014a) to extract a feature map, and predicting several bounding boxes proposal (typically 300), Second, these box proposals are processed by the remaining network to refine the bounding box predictions, this presented a 73.2% mAP (mean average precision) in PASCAL VOC dataset 2007 but with 7 FPS. The faster R-CNN was improved in Pop (2019) with a Inception v2(Huang et al 2017) as feature extractor to detect pedestrian in JAAD dataset (Rasouli et al 2017) with a 70.91% mAP. The work of Xiang et al (2017) utilizes also a RPN modified the Fast R-CNN network using image pyramids to handle the problem of scale variation in object instances and also by adding an extrapolating layer in the last convolutional layer. The evaluation was done on different benchmarks for detection and category classification, including the KITTI detection benchmark (Geiger et al 2012) for car, pedestrian, and cyclist detection showing promising results.

Papers using YOLO (You Only Look Once) (Redmon et al 2016) increase their frames for detection up to 30 FPS, but their maximum accuracy is 63.4% mAP. Additionally, grid-based detection methods have problems detecting more than one object per grid section and small objects. For SSD methods (single shot detector) (Fu et al 2017)(Yamashige and Aono 2019) you have good accuracy and speed ratio with a maximum 74.3% mAP and 46 FPS in the same dataset. Some improvements are presented in RefineNet (Zhang et al 2018) 80% mAP and 40.3 FPS.

In more recent papers, Sudha and Priyadarshini (2020) propose an algorithm for tracking multiple vehicles, tested under different weather conditions, using a convolutional YOLO v3 network and a background extraction algorithm, while the tracking task uses a Kalman filter and particle technique. The framework is tested in two datasets: KITTI and DETRAC with an average accuracy is 88.6% Xu et al (2020). Introducing a mask R-CNN network for pedestrian and vehicle detection employs a side-fusion network residual feature pyramid network (SF-FPN)+Restnet-86 as a backbone applied to the COCO dataset (Lin et al 2014) obtaining 62.41 mAP@0.5 improving the prediction against SSD networks (Le et al 2019), YOLOv3(Zhang et al 2019), Faster R-CNN (Sang et al 2017). Zhang et al (2021) established an image preprocessing using an algorithm called Retinex (Rahman et al 2004) to improve the brightness and contrast of objects, the detection is performed using an improved YOLO v3 network (Li et al 2020) gets an increase in performance by obtaining 90% mAP@0.5 and 50 FPS improving the performance of networks such as SSD and Faster R-CNN applied to COCO dataset.

2.2 Encoder–decoder architectures

Among the literature, it is possible to find various models based on encoder–decoder networks which are meant for general traffic scene understanding, attracting researchers for applications such as image segmentation and object classification.

In particular, encoder networks can map raw inputs encoding multi-scale contextual information into feature representations with pooling operations or filters, while decoder networks take feature representations as inputs to capture sharper boundaries of the object recovering spatial information (Chen et al 2018) training in the PASCAL VOC dataset.

In the work of Yang et al (2016), a deep learning algorithm aimed at contour detection was developed with an end-to-end convolutional encoder–decoder architecture. Initialization of the encoder is done with the VGG network (Simonyan and Zisserman 2014b) up to the sixth convolutional layer, while the decoder is elaborated by alternating unpooling and convolutional layers to upscale feature maps. With this approach, the model was able to generate high-quality segmented object proposals yielding a higher precision in object contour detection than previous methods.

One of the most popular models following this architecture is SegNet (Badrinarayanan et al 2017) for pixel-wise classification. This model is based on the VGG network utilizing 13 layers giving its symmetric characteristic with an equal number of layers for pooling in the encoder and unpooling for the decoder. The encoder network saves the max pooling indices while downsampling. The decoder network maps low-resolution encoder feature maps to a greater resolution.
with upsampling layers. Segnet eliminates the need for learning to upsample which in turn allows for upsampled sparse maps and produces dense feature maps and proves to be competitive and computationally efficient.

Another encoder–decoder network for pixel-wise classification is LinkNet (Chaurasia and Culurciello 2017). For the encoder network, the model is based on ResNet18 (He et al 2016) while a fully convolutional layer technique is used for the decoder. The input of each encoder layer is bypassed to the output of its corresponding decoder, aiding in the recovery of lost spatial information useful for the decoder and its upsampling operations.

In the work of Naresh et al (2018), a residual encoder–decoder network is proposed for semantic segmentation, this model is based on the LinkNet architecture and it uses the first 13 layers of VGG to capture shape information from the objects. In the decoder network, the convolutional layers are replaced by deconvolution layers while max pooling layers are replaced with upsampling layers. Since some spatial information is lost during downsampling operations, the model incorporates resolution-preserving paths to transfer missing res-information from encoding stages to the decoder network.

U-Net (Ronneberger et al 2015) is a convolutional neural network that follows the encoder–decoder architecture. In this model, pooling operators are replaced by upsampling operators with a large number of feature channels which in turn allow these layers to propagate context information and to increase the resolution of the output. To localize objects of interest, high-resolution features from the contracting path are combined with the upsampled output. This model was originally proposed for image segmentation in the biomedical context proving itself to be very useful.

More recently, the TernausNet model (Iglovikov and Shvets 2018) was developed by implementing the U-Net model as the main architecture with the key difference that the encoder network was constructed with a pre-trained implementation of the VGG11 network to detect building area images needed to train the model are size-invariant. U-Net proved itself to be a competitive network by the achievement of good performances at different biomedical segmentation and testing in the segmentation of 7 categories of the IDD dataset (Varma et al 2019).

Following the idea behind encoder–decoder architectures, this work proposes the U19-Net model, a U-Net architecture with a VGG19 as the encoder and transposed convolutions layers as the decoder.

While pre-training speeds up convergence on target tasks, it does not necessarily reduce overfitting unless very small data amounts are available, and moreover, pre-training a model helps less if the target task is more sensitive to localization rather than classification (He et al 2018). Thus, key differences with TernausNet are that our model is trained from scratch following a random weight initialization approach so the obtained feature representations are not domain-dependent making it more suitable for broader localization tasks, our method uses a loss function more simple and applied to the vehicle and pedestrian detection with greater detection complexity with overlapping or occluded objects.

The VGG19 encoder is used to map raw inputs and contextual information into more representative features, while the decoder upsamples and learns details of the feature map reducing the channels by half.

### 3 Proposed architecture

U19-Net is a very deep convolutional neural network that follows the encoder–decoder architecture, it is composed by the VGG19 network (Simonyan and Zisserman 2014b) as encoder while constructing the decoder as followed by U-Net (Ronneberger et al 2015) but in a more extensive way since it is desirable to maintain the same number of channels between encoder and decoder.

U-Net is an encoder–decoder network that follows a symmetrical approach giving its characteristic u-shape as Fig. 1 shows. In this model, the contracting path consists of a typical convolutional network architecture which in combination with successive layers it alternates convolutions and pooling operations followed by downsampling steps that doubles the number of feature maps, allowing these layers to propagate context information. Regarding the expanding path, it consists of upsampling steps of the feature maps followed by 2x2 convolutions (upconvolutions) increasing in this way the resolution of the output. To localize objects of interest, the expanding path concatenates upsampled features with high-resolution features from the contracting path. The output consists of a pixel-level mask that predicts the class of each pixel. One of the main advantages of this network is that since no dense (fully connected) layers are used, the input images needed to train the model are size-invariant. U-Net proved itself to be a competitive network by the achievement of good performances at different biomedical segmentation
U19-Net: a deep learning approach to medical image analysis

Fig. 1 U-Net configuration. The contracting path follows the architecture of typical convolutional network, while the expansive path consists of upsampling steps of the feature maps.

Fig. 2 VGG19 configuration. Notation shows the convolutional layer parameters while the number of feature channels is grouped by the horizontal bars.

applications; however, it relied on data augmentation techniques to maintain a low cardinality on the training set.

The VGG19 model is a deep neural network where its input is passed through a stack of 16 convolutional layers, each one equipped with a rectified linear activation function (ReLU), and five max pooling operations with a pixel window size of 2x2 and stride 2 for downsampling. This stack of 16 layers is then followed by three fully connected layers (giving its characteristic name of 19 layers) and a softmax operation. While all convolutional layers have filters with receptive fields of size 3x3, the number of channels increases as the network deepens starting from 64 in the first layer until it reaches 512 after each max pooling operation; thus, it remains still after some layers as shown in Fig. 2. VGG19 relies on dense layers making the inputs of the model scale-variant, therefore Simonyan and Zisserman suggested the usage of 224x224 fixed-sized inputs since it will capture whole-image statistics meaningful for training. The networks from the VGG family have demonstrated that as a network architecture deepens, the classification is enhanced in terms of accuracy while allowing the generalization of a wide range of datasets.

A key step in the design of U19-Net is to implement an improved VGG19 encoder version as the encoder path of U-Net, but it should be modified reflecting the non-usage of fully connected layers since they interfere with the scale invariance property needed by U-Net and which allows the network to take advantage of high-resolution inputs that could be meaningfully used for training. Owing to this fact, the fully connected layers and the softmax operation were replaced by a single convolution layer of 512 channels as followed in Iglovikov and Shvets (2018), allowing a clear transition between encoder and decoder. Pursuing a symmetrical approach, the decoder was built with upsampling steps of feature maps that doubles its size while halving the number of channels with transposed convolutions. Then, the output of each transposed convolution is concatenated with its corresponding feature map from the encoder, finally, a convolution operation is applied to maintain the same number of channels symmetrically. This upsampling process is completed five times, one for each max pooling operation, giving its particular u-shaped appearance.

Since the goal of the model is to predict whether a pixel belongs to the desired class or not, the output of the network consists of a pixel-level prediction mask which is preceded by a 1x1 convolution with a sigmoid activation function after the last upsampling step.

In Fig. 3, a representation of the proposed model is found. The multi-channel feature maps are represented by the orange boxes depicting how each feature map passes a series of transformations. The number of channels is written below each box, gray boxes represent copied and concatenated feature maps. The arrows depict different NN operations as found in the label box of the figure.

4 Classification framework

This section details the data processing and target preparation as model data, the IoU score used as the objective function, and a description of the training stages of the model.
Table 1  Summary of bounding boxes sizes after image rescaling within dataset

| Attribute   | Maximum | Minimum | Average |
|-------------|---------|---------|---------|
| Veh. height | 562.00  | 0.00    | 107.43  |
| Veh. width  | 947.00  | 0.00    | 127.80  |
| Ped. height | 428.00  | 6.00    | 81.99   |
| Ped. width  | 244.00  | 6.00    | 31.67   |

4.1 Data and target preparation

To train and evaluate the network, the Udacity open-source dataset (Udacity Inc. 2016) was used, since it consists of a data collection of 9,420 RGB video frames from a vehicle while driving in urban daylight scenarios and provides annotated labels for cars, trucks, and pedestrians. For each frame, it is possible to find one or more instances per class, in total, there are 5,675 pedestrians and 66,389 vehicles (cars + trucks) instances within the dataset. The annotated data consists of labels of the instances occurrences within frames along with pixel information in \((x - \text{min}, y - \text{min})\) and \((x - \text{max}, y - \text{max})\) terms as coordinates that define boxes where the instances are bounded.

To measure the performance of U19-Net, two models were trained and evaluated individually (as sect. 4.3 discusses in detail) in vehicles and pedestrians detection tasks for self-driving cars, where a Monte Carlo cross-validation as split method (Dubitzky et al. 2007) creates multiple random splits of the dataset into training and validation data. For each such split, the model is fit to the training data, and predictive accuracy is assessed using the validation data was used to train and performance evaluation, repeat 5 times per model ensuring that all frames in the dataset are used for training at least once. From the 5,675 pedestrians instances available, 60% is used for training while the other 40% is used for validation and testing sets, respectively. In the case of vehicles, from the 66,389 available instances, a subset of 2,500 was used to train and evaluate the deep model. From this subset, the split rate for training, validation, and testing sets were similar to those of pedestrians in percentage terms.

Data preprocessing is done by rescaling the input image resolution from 1920x1200 to 960x640 and by doing a computation of the bounding boxes that represents the instances of classes found within frames. These bounding boxes are needed to obtain the region of interest (ROI) of instances needed to feed the network and perform gradient updates. This process is illustrated in figure 4 in which the left image is a randomly sampled frame from dataset and the right image is the ROI mask of computed ground truth. The ultimate goal of the convolutional neural network is to predict an ROI mask given the original frames.

4.2 Intersection over union (IoU)

Intersection over Union (IoU), also referred to as the Jaccard index, is a statistic metric used to evaluate and aid in the understanding of the similarity between sample sets, this metric is invariant to the scale of the problem (Rezatofighi et al. 2019). In particular, it is used as an evaluation metric to measure the accuracy of an object detector on a particular dataset. It is formally defined as the amount of intersection divided by the amount of the union of the sample sets. In other words, IoU is a measure of the overlap percentage between sets defined as:

\[
\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}
\]

where \(A\) and \(B\) are two finite sets, in this case corresponding to the ground truth, and the prediction masks output of the proposed model. The resulting IoU has a score in the closed set of \([0, 1]\), where 0 indicates no overlap, resulting in poor detection, and 1 indicates a complete overlap between the sets.
meaning excellence in object detection tasks. In general, an IoU threshold greater than 0.7 is to be considered as a “good” prediction in order to determine true positives and false positives since it provides a reasonable compromise between loose and very strict scores (Zitnick and Dollar (2014)).

For loss function proposes, the Jaccard distance was used in its straightforward implementation in continuous domain replaces intersection and union by product and sum as follows (Rahman and Wang (2016) and ( Martire et al (2017)):

$$\text{loss}(y, \hat{y}) = 1 - \frac{(y \star \hat{y}) + \epsilon}{(y + \hat{y} - y \star \hat{y}) + \epsilon} \quad (2)$$

where $\epsilon$ prevents zero division. Both IoU score as a cost function and loss function should evaluate each pixel $i$ between its ground truth $y_i$ and the current result of the deep model $\hat{y}_i$.

We use the following performance metrics to evaluate our model using TP, TN, FP, and FN denote the true positives, true negatives, false positives, and false negatives, respectively, obtained by (1).

First, precision is a measure of what proportion of predicted positives were truly positive (3), Recall is a measure of what proportion of actual positives were classified (4), F1 score is the harmonic mean of recall and precision (5).

Average precision (AP) and mean average precision (mAP) of a single category were used to calculate the performance according to formula (19)

$$\text{Pre} = \frac{TP}{TP + FP} \quad (3)$$

$$\text{Rec} = \frac{TP}{TP + FN} \quad (4)$$

$$F1 = \frac{2 \times \text{Pre} \times \text{Rec}}{\text{Pre} + \text{Rec}} \quad (5)$$

$$AP = \frac{TP + TN}{TP + FP + TN + FN} \quad (6)$$

### 4.3 Training

With the main goal of identifying and assessing the performance of U19-Net, two models were trained: one for vehicles detection and another for pedestrians.

As mentioned in sect. 4.1, the data for experiments were built by Monte Carlo cross-validation as split method, shuffling the datasets and splitting them into 60-20-20 percentages for training, validation, and testing, respectively.

Each training task was repeated 5 times ensuring that all frames in the dataset are used for training and test at least once, minimizing the variance with an intermediate bias of the test error rates. One of the major advantages of this method is that it is computationally inexpensive compared to other cross-validation techniques.

The structure and design parameters of the U19-Net model were the same in both cases, and the differences in the experiments are the amount of the input images and the classes number of the target.

Implementation was done with Tensorflow (Abadi et al 2016) within a cluster running Ubuntu Server 12.4 with 8 Xeon-3 processors, and 16 GB of RAM. To maximize usage of the GPU and memory components, a batch-oriented approach was followed where the number of samples per gradient update was 3,405 for the pedestrians’ case and 1,500 for vehicles within training sessions.

The training procedure for both networks was the same, in which the RGB input images and their corresponding segmented masks from ROI were used to learn the parameters of the model. The learning rate was set to $1e-04$, while the number of epochs was 55 in both cases. Learning of the weights is carried out by Adam optimizer, an extension to the stochastic gradient descent algorithm, which updates the weights of the network on an iterative basis on the training data. Implementation of the algorithm was done through Keras, where the parameters set were $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-08}$ and decay = 0.0. In a deep neural network with many convolutional layers and different paths through it, a correct initialization of the weights is important, therefore, a random initialization approach was used, preventing activation layers to vanish or skyrocketing during passes within the deep neural network, allowing feature maps to have unit variance (Goodfellow et al 2016).

### 5 Classification experiments and results

In this section, training and testing results for U19-Net are presented, as well as a comparison with U-Net performing the same classification tasks within the chosen dataset. The experiments were conducted on a private cluster with 8 Xeon-3 processors as subsection 4.3 refers. While the training sets used for both vehicles and pedestrians were described in the previous sections, for validation and testing purposes of the networks, 500 instances were used for vehicles in each stage, while for pedestrians 1,135 instances were used in both stages. Training the model requires about 9 seconds per step and about 2 seconds per step in evaluation and testing phases within-cluster configuration.

Once a model is trained and validated, the final performance is evaluated through the testing set, in which at the output of a given neural network, a pixel-level prediction mask is generated where each pixel corresponds to a probability value reflecting whether an obstacle is to be found or not. Those pixels are then mapped to a color value and superposed with the images being evaluated to present objects of interest within the tests. Although the training parameters and configurations for U19-Net with vehicles and pedestri-
Table 2. Summary of network accuracy with IoU score and configuration with vehicles experiments

| Attribute          | Training | Validation | Testing |
|--------------------|----------|------------|---------|
| Accuracy           | 94.50    | 86.31      | 87.08   |
| Time in seconds    | 791435.49| 421.03     | 405.95  |
| Epochs             | 55       | –          | –       |
| Batch size         | 1500     | 500        | 500     |

Vehicle experiment summary with U19-Net. Precision 91.7, recall 91.9 and F1 score 91.8.

Fig. 5 Training plot for vehicle classification. The number of epochs was set to 55 and the final training accuracy was about 94.5%.

5.1 Experimentation with vehicles

Table 2 summarizes the performance results in the training, validation, and testing stages as well as network configuration and metrics while assessing U19-Net’s performance in the vehicle classification task. The total training time within-cluster configuration was 9.16 days, with a mean time of 9 seconds per step. An illustration of the accuracy vs epochs plot in the training session is shown in Fig. 5.

The performance of the convolutional network was evaluated with 500 instances, some detection examples are illustrated in Fig. 6 in which the left image is a randomly sampled frame from the testing set, the center image is the predicted ROI mask which denotes vehicle detection and the right image is the ROI mask of computed ground truth. The accuracy obtained with the testing data was about 87.08% IoU score.

Fig. 6 U19-Net predictions with vehicles. Images from center panel denotes the NN predictions in yellow with the ground truth bounding boxes superimposed.

5.2 Experimentation with pedestrians

Similar to the case of the vehicles, Table 3 summarizes the performance results in train-val-test stages as well as network configuration and metrics for U19-Net pedestrian evaluation. Training time was 19.26 days, with a mean time of 9 seconds per step. Figure 7 illustrates the accuracy vs epochs plot for the training process.
Table 3  Summary of network accuracy with IoU score and configuration with pedestrians experiments

| Attribute         | Training | Validation | Testing |
|-------------------|----------|------------|---------|
| Accuracy          | 88.88    | 77.81      | 78.18   |
| Time in seconds   | 1663902.01 | 1867.05  | 880.05  |
| Epochs            | 55       | –          | –       |
| Batch size        | 3405     | 1135       | 1135    |

Fig. 7  Training plot for pedestrian classification. The number of epochs was set to 55 and the final training accuracy was about 88.88%

In this case, the performance of the convolutional network was evaluated with 1,135 instances. Figure 8 shows results obtained in the testing process. The accuracy obtained with the testing data was about 78.18% IoU score.

5.3 Edge cases

Given the fact that the predictions of the neural network are a matter of pixel regions of interest rather than predicting bounding square boxes around the objects, there are some particular edge cases in which the prediction holds a more accurate representation of the delimited predicted object than the ground truth bounding boxes; as an example, Fig. 9 shows two scenarios: In the upper row part, U19-Net predicted something that was not considered in the ground truth annotations; however, a close inspection reveals that there is indeed a vehicle there; on the other hand, in the lower row we can appreciate a parking lot where a car is hidden by some bushes, while the ground truth selects all that portion, and the prediction avoids those pixels related to non-car instances. While the visible portion of the vehicles in these examples is relatively small, it is interesting to see how the prediction adapts to just those areas where the object of interest is, since

Fig. 8  U19-Net predictions with pedestrians. Images from center panel denotes the NN predictions in yellow with the ground truth bounding boxes superimposed

this could lead to better and more accurate predictions while fine-tuning the model.

5.4 Result comparison

As a means of comparison, the proposed U19-Net model is compared with an U-Net network (Ronneberger et al 2015) in the same detection tasks for vehicles and pedestrians. Pursuing performance maximization in terms of both training
times and accuracy, two network configurations were used within U-Net. While the used architecture was the same as defined in Fig. 1, the differences were at the resolution of the feature maps; in particular, at the first U-Net configuration, they were reduced starting with a feature map size of 8 and then doubling its size at each convolution step reaching a maximum size of 128; after that, the feature map size was halved at each upconvolution reaching a size of 8 before the last convolution is applied resulting in faster convergence and relatively small training times.

For convenience, this first low-resolution U-Net configuration is referred to as “small U-Net.” The second model configuration uses the same exact size of feature maps as defined by U-Net, resulting in better accuracy results rather than small training times. This configuration is referred to as full U-Net.

The training, validation, and testing configurations of both small and full U-Net’s were the same as U19-Net. Table 4 summarizes results and configuration of both vehicles and pedestrians’ experiments with small U-Net while Table 5 is analog but for the full U-Net implementation.

Table 4 Summary of small U-Net results and configuration

|                      | Vehicles | Training | Validation | Testing |
|----------------------|----------|----------|------------|---------|
| Accuracy             | 74.38    | 70.34    | 70.24      |         |
| Time in seconds      | 13291.15 | 84.30    | 85.70      |         |
| Batch size           | 1500     | 500      | 500        |         |
| Pedestrian           |          |          |            |         |
| Accuracy             | 66.86    | 54.89    | 53.98      |         |
| Time in seconds      | 695729.24| 172.85   | 171.72     |         |
| Batch size           | 3405     | 1135     | 1135       |         |

Table 5 Summary of full U-Net results and configuration

|                      | Vehicles | Training | Validation | Testing |
|----------------------|----------|----------|------------|---------|
| Accuracy             | 95.99    | 85.61    | 86.44      |         |
| Time in seconds      | 1061150.96| 1044.50  | 1052.12    |         |
| Batch size           | 1500     | 500      | 500        |         |
| Pedestrian           |          |          |            |         |
| Accuracy             | 89.97    | 75.39    | 75.25      |         |
| Time in seconds      | 1886628.98| 1829.95  | 1825.40    |         |
| Batch size           | 3405     | 1135     | 1135       |         |

Table 6 Summary of detection scores within networks

|                      | Network | Training | Validation | Testing |
|----------------------|---------|----------|------------|---------|
| Vehicle detection    |         |          |            |         |
| Small U-Net          | 74.38   | 70.34    | 70.24      |         |
| Full U-Net           | 95.99   | 85.61    | 86.44      |         |
| U19-Net              | 94.50   | 86.31    | 87.08      |         |
| Pedestrian detection |         |          |            |         |
| Small U-Net          | 66.86   | 54.89    | 53.98      |         |
| Full U-Net           | 89.97   | 75.39    | 75.25      |         |
| U19-Net              | 88.88   | 77.81    | 78.18      |         |

Table 7 presents a comparison of U19-Net with some models presented by Chen et al. (2021) where it was necessary to obtain a bounding box of the predicted objects of our network because the prediction is at pixel level and the comparisons are at bounding box level. The table shows a comparison through the AP@0.75 with FPS of each of the networks.

6 Discussion

As more layers are added, U19-Net could take bigger training times when running it on larger datasets due to all the processing in middle layers; however, it can also be appreciated that the structure and added depth benefits the final scores as well as testing times as the experiments in sect. 5 demonstrated, where the proposed U19-Net model performs well for both vehicles and pedestrians in the Udacity dataset.
Table 7 Comparative between U19-Net and several CNN network models

| Network                  | AP@0.75 Vehicles | AP@0.75 Pedestrians | max. FPS | Parameters (m) | Resolution |
|--------------------------|------------------|---------------------|----------|----------------|------------|
| Faster R-CNN Inception v2| 85.4             | 48.1                | 12.47    | 12.87          | 181 x 600  |
| Faster R-CNN ResNet50    | 85               | 50.2                | 8.61     | 43.25          | 181 x 600  |
| R-FCN ResNet101          | 83.8             | 48.1                | 8.93     | 64.42          | 181 x 600  |
| SSD Inception v2         | 49.8             | 12.1                | 29.6     | 13.32          | 91 x 300   |
| SSD MobileNet v2         | 47.4             | 10                  | 70       | 4.63           | 91 x 300   |
| U19-Net                  | 80.3             | 46                  | 1.29     | 340.38         | 960 x 640  |

when compared to U-Net (Ronneberger et al 2015) in terms of IoU score.

While looking for a 'light' version of U-Net with the small U-Net implementation, it could be observed that reducing the resolution of the feature map sizes considerably reduces the time spent in training and testing than using the full implementation; it is sufficient to see that the training time for full U-Net with vehicles was 12.28 days versus the 3.69 hours of small U-Net. As a matter of reference, the small U-Net implementation had 491,137 parameters, whereas the full U-Net consisted of 313,789,405 parameters. On the other hand, while small U-Net is faster, the final results were lower than using the full implementation, to the point that it misclassified at the pedestrians’ detection task rendering itself not good for this task.

Now, seeking for the best result scores in training within the neural networks, it turned out that the full U-Net implementation achieved higher scores than U19-Net by a slight margin, however, when assessing the final evaluation of the models with the testing sets, U19-Net performed better at both pedestrians and vehicles detection scores than U-Net. While in detecting vehicles the gaining at the final score of U19-Net was discrete, there is a considerable margin when comparing the final scores at pedestrians detection between U19-Net and U-Net.

Furthermore, even though U19-Net could be considered as a deep architecture with its 340,348,177 parameters, final testing times were much lower at receiving, processing, and classifying inputs from the testing set; in particular, it took 2.10 seconds for full U-Net to process one instance versus 0.81 seconds of U19-Net, while in the pedestrians experiment, it took 1.61 seconds to full U-Net to process one instance versus 0.77 seconds of U19-Net.

Results suggest that despite the small amount of the data subset used at vehicle classification, the proposed model performed better than the pedestrian experiment mainly because the context information retrieved from the network was more meaningful due to larger bounding boxes for vehicles in comparison with pedestrians, where this class is to be considered specifically challenging, however, by utilizing a deep network architecture U19-Net performed better than full U-Net and small U-Net in these experiments with the added benefit that training can be done end-to-end.

Finally, through the comparative Table 7 it can be observed that the best performance for both detection tasks is the Faster R-CNN with backbone Inception v2 and RestNet50; however, the proposed U19-Net would be positioned very close to these networks, and the comparison has to be considered unbalanced because all the networks except the proposal are trained for Kitti object 2D which has a different resolution to the proposal. This is due to the lack of comparisons with the Udacity dataset in the literature for recent networks. Another important aspect of imbalance is found in the training, the comparative models were trained by 800K epochs with 7481 images while our proposal was trained only 55 epochs with 5675 images. Undoubtedly our proposal is the largest number of parameters, the cost to consider a very deep network. It would be interesting to make a comparison on equal terms, for example, the same resolution 181 x 600 which would reduce the number of parameters and therefore increase the FPS. And because the encoder layers are concatenated with the decoder it could be less affected for the detection of small objects such as what happened in comparative networks when detecting pedestrians.

7 Conclusions and future work

In this work, the U19-Net model developed for the specific task of vehicles and pedestrians classification is proposed and explored by the improvement of an encoder–decoder architecture with the utilization of a very deep convolutional neural network. Despite the amount of data used for training with vehicles being constrained, results of the model show an overall improvement for classification tasks, achieving an outstanding performance of 80.3% AP@0.75 for vehicles and 46% AP@0.75 for pedestrians.

A comparison was made with recent CNN networks obtaining competitive results in most of the performance metrics, and it should be taken into account that the AP and IoU
metrics were established for a detection greater than 0.75 and 0.7, respectively, equivalent to a very good location of the object by pixel or bounding box.

We believe that encoder–decoder networks with representation depth are particularly beneficial for vision tasks and as future work, the use of other datasets to make a more egalitarian comparison is contemplated, as well as looking for optimization for the reduction of the parameters of the proposed network. Also, safety being the highest priority in the research and development stages for self-driving cars, it is of main interest the application of real-time solutions in perception and control systems, thus we suggest evaluation of the proposed model in real-time driving scenarios with a monocular camera system to get real-time feedback and better fine-tune the model.

Acknowledgements I would like to thank Dr. Paul Erick Mendez Monroy for his insightful comments and suggestions as an advisor to make this study possible; last but not least, I would like to thank the Autonomous University of Mexico (UNAM) for the programs UNAM-PAPIIT IA104218 and IA102620 for their support and funding of this research.

References

Abadi M, Barham P, Chen J, et al (2016) Tensorflow: A system for large-scale machine learning. In: 12th Symposium on Operating Systems Design and Implementation, pp 265–283
Amine M, Djoudi K (2019) Vehicles detection using the mlp and the correlation measurement. In: 2019 International Conference on Advanced Electrical Engineering (ICAAE), IEEE, pp 1–5
Amitha I, Narayanan N (2021) Object detection using yolo framework for intelligent traffic monitoring. In: Machine Vision and Augmented Intelligence-Theory and Applications. Springer, p 405–412
Badrinarayanan V, Kendall A, Cipolla R (2017) SegNet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence 39(12):2481–2495
Baheti B, Imnani S, Gajre S, et al (2020) Eff-unet: A novel architecture for semantic segmentation in unstructured environment. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp 358–359
Brunetti A, Buongiorno D, Trotta GF et al (2018) Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. Neurocomputing 300:17–33
Cai Z, Fan Q, Feris RS, et al (2016) A unified multi-scale deep convolutional neural network for fast object detection. In: European conference on computer vision, Springer, pp 354–370
Chan YM, Fu LC, Hsiao PY, et al (2016) Pedestrian and vehicle detection and tracking with object-driven vanishing line estimation. In: Asian Conference on Computer Vision, Springer, pp 436–451
Chaurasia A, Calucci E (2017) Linknet: Exploiting encoder representations for efficient semantic segmentation. In: 2017 IEEE Visual Communications and Image Processing (VCIP), IEEE, pp 1–4
Chen C, Liu MY, Tuzel O, et al (2016) R-cnn for small object detection. In: Asian conference on computer vision, Springer, pp 214–230
Chen L, Lin S, Lu X, et al (2021) Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems
Chen LC, Zhu Y, Papandreou G, et al (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp 801–818
Dubitzky W, Granzow M, Berrair DP (2007) Fundamentals of data mining in genomics and proteomics. Springer Science & Business Media
Everingham M, Van Gool L, Williams CK et al (2010) The pascal visual object classes (voc) challenge. International journal of computer vision 88(2):303–338
Fu CY, Liu W, Ranga A, et al (2017) Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659
Galvao LG, Abbod M, Kalgunova T et al (2021) Pedestrian and vehicle detection in autonomous vehicle perception systems—a review. Sensors 21(21):7267
Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp 3354–3361
Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
Girshick R, Donahue J, Darrell T, et al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press
He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
He K, Girshick R, Dollár P (2018) Rethinking imagenet pre-training. arXiv preprint arXiv:1811.08883
Huang J, Rathod V, Sun C, et al (2017) Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7310–7311
Iglovikov V, Shvets A (2018) Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. arXiv preprint arXiv:1801.05746
Jordan MI, Mitchell TM (2015) Machine learning: Trends, perspectives, and prospects. Science 349(6245):255–260
Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
Le X, Xiang L, Hua Z (2019) Vehicle detection on road frontage based on ssd. Software Guide 5(1):27–36
Li C, Wang X, Liu W (2017) Neural features for pedestrian detection. Neurocomputing 238:420–432
Li C, Wang R, Li J, et al (2020) Face detection based on yolov3. In: Recent Trends in Intelligent Computing, Communication and Devices. Springer, p 277–284
Lin TY, Maire M, Belongie S, et al (2014) Microsoft coco: Common objects in context. In: European conference on computer vision, Springer, pp 740–755
Liu W, Anguelov D, Erhan D, et al (2016) Ssd: Single shot multibox detector. In: European conference on computer vision, Springer, pp 21–37
Liu Z, Chen Z, Li Z, et al (2018) An efficient pedestrian detection method based on yolov2. Mathematical Problems in Engineering 2018
Maggiori E, Tarabalka Y, Charpiat G, et al (2018) Can semantic labelness affect the classification of images? the inria aerial image labeling benchmark. In: IEEE International Geoscience and Remote Sensing Symposium (IGARSS), IEEE
Martire I, Da Silva P, Plastino A et al (2017) A novel probabilistic jaccard distance measure for classification of sparse and uncertain data. Tech. rep. Kent Academic Repository
Nam W, Dollar P, Han JH (2014) Local decorrelation for improved pedestrian detection. In: Advances in Neural Information Processing Systems, pp 424–432
Naresh Y, Little S, O’Connor NE (2018) A residual encoder-decoder network for semantic segmentation in autonomous driving scenarios. In: 2018 26th European Signal Processing Conference (EUSIPCO), IEEE, pp 1052–1056
NHTSA NHTSA, et al (2017) Automated driving systems 2.0: A vision for safety. Washington, DC: US Department of Transportation, DOT HS 812:442
Pop DO (2019) Detection of pedestrian actions based on deep learning approach. Studia Universitatis Babes-Bolyai Informatica
Rahman MA, Wang Y (2016) Optimizing intersection-over-union in deep neural networks for image segmentation. In: International symposium on visual computing, Springer, pp 234–244
Rahman Zu, Jobson DJ, Woodell GA (2004) Retinex processing for automatic image enhancement. Journal of Electronic imaging 13(1):100–110
Rasouli A, Kotseruba I, Tsotsos JK (2017) Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp 206–213
Redmon J, Divvala S, Girshick R, et al (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
Ren S, He K, Girshick R, et al (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems 28:91–99
Rezatofighi H, Tsoi N, Gwak J, et al (2019) Generalized intersection over union: A metric and a loss for bounding box regression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 658–666
Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer, pp 234–241
Sang J, Guo P, Xiang Z, et al (2017) Vehicle detection based on faster-rcnn. Journal of Chongqing University 40(7):32–36
Scholar G (2021)
Seenouvong N, Watcharerruetai U, Nuthong C, et al (2016) A computer vision based vehicle detection and counting system. In: 2016 8th International Conference on Knowledge and Smart Technology (KST), IEEE, pp 224–227
Simonyan K, Zisserman A (2014a) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
Simonyan K, Zisserman A (2014b) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
Sudha D, Priyadarshini J (2020) An intelligent multiple vehicle detection and tracking using modified vibe algorithm and deep learning algorithm. Soft Computing 24:17,417–17,429
Szegedy C, Liu W, Jia Y, et al (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
Tan M, Le Q (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, PMLR, pp 6105–6114
Tome D, Monti F, Baroffio L et al (2016) Deep convolutional neural networks for pedestrian detection. Signal Processing: Image Communication 47:482–489
Udacity Inc. (2016) Open sourcing 223gb of driving data
Uijlings JR, Van De Sande KE, Gevers T et al (2013) Selective search for object recognition. International journal of computer vision 104(2):154–171
Varma G, Subramanian A, Namboodiri A, et al (2019) Idl: A dataset for exploring problems of autonomous navigation in unconstrained environments. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, pp 1743–1751
Velazquez-Pupo R, Sierra-Romero A, Torres-Roman D, et al (2018) Vehicle detection with occlusion handling, tracking, and oc-svm classification: A high performance vision-based system. Sensors 18(2):374
Wang H, Cai Y (2015) Monocular based road vehicle detection with feature fusion and cascaded adaboost algorithm. Optik 126(22):3329–3334
WHO (2018) Global status report on road safety 2018. World Health Organization and others https://apps.who.int/iris/bitstream/handle/10665/276462/9789241565684-eng.pdf
Xiang Y, Choi W, Lin Y, et al (2017) Subcategory-aware convolutional neural networks for object proposals and detection. In: 2017 IEEE winter conference on applications of computer vision (WACV), IEEE, pp 924–933
Xu C, Wang G, Yan S, et al (2020) Fast vehicle and pedestrian detection using improved mask r-cnn. Mathematical Problems in Engineering 2020
Yamashige Y, Aono M (2019) Epsod7: Real-time object detection using 7 layers of convolution based on ssd. 2019 International Conference of Advanced Informatics: Concepts. Theory and Applications (ICAICTA), IEEE, pp 1–6
Yang J, Price B, Cohen S, et al (2016) Object contour detection with a fully convolutional encoder-decoder network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 193–202
Zhang F, Yang F, Li C (2019) Fast vehicle detection method based on improved yolov3. Computer Engineering and Applications 55(02):12–20
Zhang H, Hu Z, Hao R (2021) Joint information fusion and multi-scale network model for pedestrian detection. The Visual Computer 37(8):2433–2442
Zhang S, Wen L, Bian X, et al (2018) Single-shot refinement neural network for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4203–4212
Zitnick CL, Dollár P (2014) Edge boxes: Locating object proposals from junctions. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp 1042–1050

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.