SDC-Net++: End-to-End Crash Detection and Action Control for Self-Driving Car Deep-IoT-Based System

Mohammed Abdou Tolba * and Hanan Ahmed Kamal *

Posted Date: 8 May 2024

doi: 10.20944/preprints202405.0468.v1

Keywords: Autonomous driving, deep learning; computer vision; multitask learning; crash detection; path planning; automatic emergency braking; camera-cocoon; IoT; system

Article

SDC-Net++: End-to-End Crash Detection and Action Control for Self-Driving Car Deep-IoT-Based System

Mohammed Abdou Tolba [1,*] and Hanan Ahmed Kamal [2]

[1] PhD Candidate, Senior Team leader; mohamed.tolba@si-vision.com, d.mabdou258@gmail.com
[2] Department of Electronics and Communications Engineering, Faculty of Engineering, Cairo University; hanan.yousef@eng.cu.edu.eg
* Correspondence: d.mabdou258@gmail.com; Tel.: +201112238861

Abstract: Few prior works study self-driving cars by deep learning with IoT collaboration. SDC-Net, which is an end-to-end multitask self-driving car camera cocoon IoT-based system, is one of the research that tackles this direction. However, by design SDC-Net is not able to identify the accident locations, it only classifies if it is a crash scene or not. In this work, we introduce an enhanced design for the SDC-Net system by 1) replacing the classification network with a detection one, 2) adapting our benchmark dataset labels built on the CARLA simulator to include the vehicles bounding boxes while keeping the same training, validation, and testing samples, 3) modifying the shared information via IoT to include the accident location. We keep the same path planning and automatic emergency braking network, the digital automation platform, and the input representations to formulate the comparative study. SDC-Net++ system is proposed to 1) output the relevant control actions, especially in case of accidents: accelerate, decelerate, maneuver, and brake, and 2) share the most critical information to the connected vehicles via IoT, especially the accident locations. A comparative study is also conducted between SDC-Net and SDC-Net++ with the same input representations: front camera only, panorama and bird-eye views, and with single-task networks: crash avoidance only, and multitask networks. Multitask network with a BEV input representation outperforms the nearest representation in precision, recall, f1-score, and accuracy by more than 15.134%, 12.046%, 13.593%, and 5%, respectively. SDC-Net++ multitask network with BEV outperforms SDC-Net multitask with BEV in precision, recall, f1-score, accuracy, and average MSE by more than 2.201%, 2.8%, 2.505%, 2%, and 18.677% respectively.

Keywords: Autonomous driving; deep learning; computer vision; multitask learning; crash detection; path planning; automatic emergency braking; camera-cocoon; IoT; system

1. Introduction

Autonomous Driving is a dream come true. Despite the restricted laws preventing fully automated vehicles, researchers and scientists do not spare any efforts in the field to prove its stability as a counter-push against these laws. The US department adopted 6 levels of driving automation [1] as shown in Synopsys Figure 1 ranging from 0 (fully manual) to 5 (fully autonomous). These levels can be categorized into 2 categories; a human driver **must** monitors the surrounding environment, this is at levels (0-2), and an automated system monitors the environment, this is at levels (3-5).

Level 0-No Driving Automation is the level at which the driver fully controls the vehicle. **Level 1-Driver Assistance** is the level in which the adaptive cruise control (ACC) [2] system is already integrated vehicle keeping the ego-vehicle at a safe distance from the up-front vehicle, however, the human driver monitors other control actions like steering and braking. **Level 2-Partial Automation** is the level at which the advanced driver assistance system (ADAS) [3] is integrated, so the vehicle controls both steering and acceleration/deceleration. Still, the driver can take control of the vehicle at any time. Tesla [4] and General Motors (GM) [5] are the largest companies in the field that achieve supercruise systems as Level-2. **Level 3-Conditional Driving Automation** is the level in which environmental detection capabilities are integrated, and the vehicle can take control actions but with human driver override who is necessary to take control actions in case of the system executes wrongly. **Level 4- High Automation** is the level at which a failure system is integrated so that the vehicle

 doi:10.20944/preprints202405.0468.v1

can perform control actions and, in case of failure, a system can perform the right actions without human-driver interaction; however, the driver override is still an option. The vehicle can be operated in self-driving mode within low-speed and limited areas like urban areas in which the vehicle speed does not exceed 30 kmph; this is also geofencing. Volvo [6], Waymo [7], and Baidu [8] are the most significant companies at this level. **Level 5-Full Driving Automation** is the level at which human attention and interaction are not needed at all, the vehicle will take control actions in both urban and Highway areas. Our proposed system targets this level.

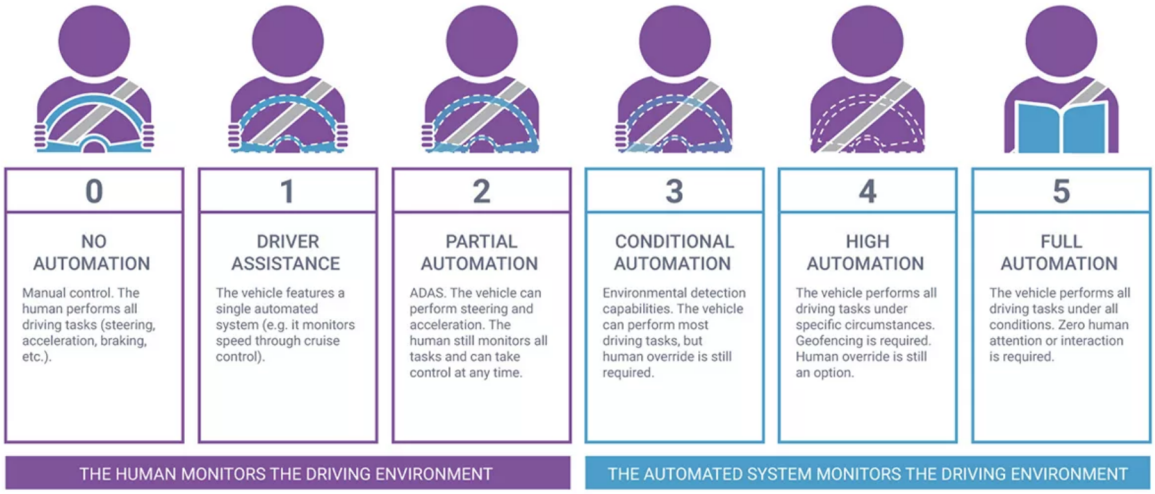

Figure 1. [Synopsys] Autonomous Driving Levels

The self-driving car pipeline consists of 3 main phases as shown in Figure 2: Perception, Planning, and Control. **Perception** of the environment is the main building block of autonomous driving. Perception is implemented by collecting sensor readings from different sensors like lidars, radars, camera sensors, and more, then fusing these sensor readings to achieve robust information about the surroundings [9]. Perception is decomposed into two sub-blocks: detection and localization. The detection part aims to classify and detect all the concerned objects around the vehicle by applying some sophisticated algorithms: lane detection, traffic light detection and classification, and object detection and tracking. The localization part aims to localize the ego vehicle in the world coordinates against the other objects with the help of high definition HD-map.**Path planning** [10] is one of the main blocks of the autonomous driving pipeline that is integrated into the system to identify all relevant control actions. Path planning is considered a multistage step that can be decomposed into the following sub-blocks: **Route Planning**: it is also called (Global Planning), which aims to identify the possible routes to reach the desired position by the HD-map assistance, **Trajectory Prediction** [11,12]: it benefits from the perception for the surrounded objects to identify all the possible trajectories. Trajectory prediction is made based on applying different motion models for the classified and detected objects in which there is a motion model for the cars, a different one for the pedestrians, etc. **Trajectory Planning** [13]: it computes and plans the trajectory based on the perception of the environment. After that, the ego vehicle plans its trajectory safely. **Control** [14,15] is responsible for taking the relevant and safe actions based on the trajectory planning step. It controls the ego-vehicles' manipulated variables: throttle, steer, and brake. The control step takes a list of waypoints (trajectories) and target velocities generated by the planning step passing through controller algorithms like PID or MPC, which calculate how much to steer, accelerate, or brake to meet the target trajectory.

 doi:10.20944/preprints202405.0468.v1

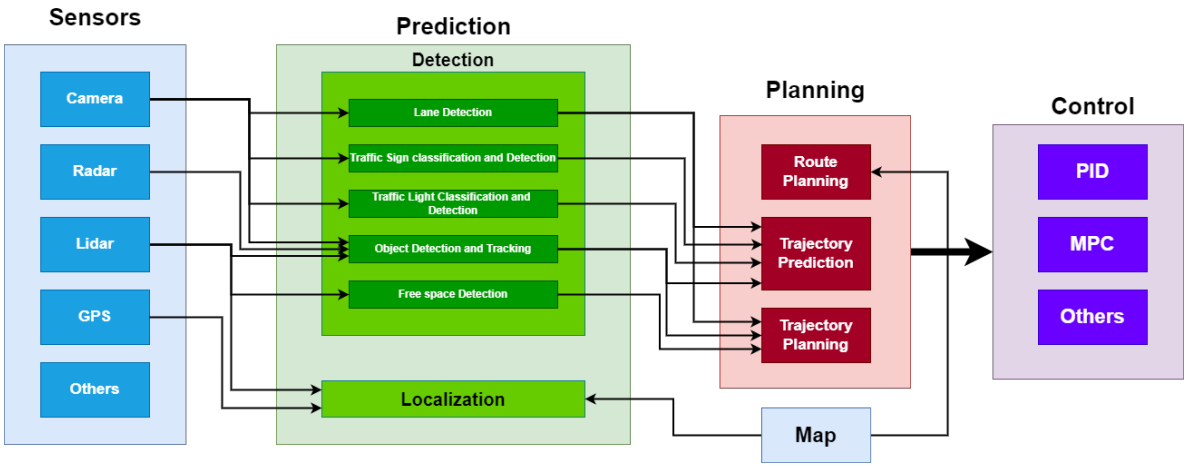

Figure 2. Self-Driving Pipeline

In this paper, we introduce the (SDC-Net++) system as an extension to our previous research (SDC-Net) system. SDC-Net++ is an end-to-end system that aims to make vehicles navigate autonomously through the predicted action control: throttle, maneuver, and brake, in addition to a crash detection feature not only to detect crashes but also to help in navigation. SDC-Net++ takes the input of 4 cameras installed around the ego vehicle (camera cocoon) covering 360^{o} around the vehicle. Then, SDC-Net++, following the same way of SDC-Net, applies different input representations: front camera, panorama, and bird's eye views which input to a multitask deep neural network that can predict the vehicle's action control and detect crashes. The multitask network differs from the SDC-Net system in which the crash classification network in SDC-Net is replaced by a crash detection network in SDC-Net++; bounding boxes prediction instead of binary classification. SDC-Net++ is attached to IoT platforms that can communicate with other vehicles and share the crash information gathered. Likewise SDC-Net, we created our dataset using the CARLA simulator serving the main goal of having path planning, automatic emergency braking, and crash detection functionalities which are the main building blocks to set up a self-driving car.

The remainder of the paper is decomposed as follows: section 2 includes a literature review on self-driving cars including our previous paper (SDC-Net). Section 3 describes the main contributions showing the SDC-Net++ system and the network architecture. Dataset adaptations and analysis are described in section 4. Results and the conducted experiments are shown in section 5 comparing crash detection only and multitask results. Section 6 elaborates meaningful comments on the conducted experiments, and then discusses comparing SDC-Net++ with our previous network SDC-Net. A Summarized conclusion is shown in section 7.

2. Literature Review

No vehicle can drive itself, but these come the closest. There is no fully autonomous vehicle for sale today, but some automakers are advancing the field. Over the past few years, a few cars have gone on sale with driver assistance features that take much of the burden off the driver. These systems alleviate driver fatigue by assisting with steering and acceleration. This technology is beneficial for drivers with long commutes. And it can come in handy during road trips requiring lots of driving. Much research has been conducted to achieve self-driving vehicles as soon as possible. Sallab and Abdou et al. proposed a framework in [16] to achieve autonomous driving, then proved the success of this framework in [17] when applied their proposed algorithm on TORCS simulator [18]. Autonomous driving using imitation learning is also tackled in [14] which is inspired by Intel [19].

Hoang Duy Loc et al. proposed a fast and accurate 3D multiobject detection framework in [20] to balance speed and accuracy using bird's eye view (BEV) representation. Experiments on the challenging KITTI dataset show that this work outperforms other methods by significant margins

 doi:10.20944/preprints202405.0468.v1

while running at 62 FPS on a single GTX-1080 GPU. Henan Hu et al. depended on pseudo-LiDAR to achieve monocular 3D object detection in [21]. The pseudo-LiDAR point cloud is generated using two steps: 1) joint image segmentation and geometric constraints, used to predict the target depth and provide the depth prediction confidence measure. The predicted target depth is fused with the overall depth of the scene and results in the optimal target position, 2) they utilize the target scale, normalized with the gaussian function, as a priori information, so the uncertainty of depth distribution is reduced. With the refined depth information, they convert the optimized depth map into the point cloud representation (pseudo-LiDAR).

Jiayu Zou et al. analyzed the vital differences between camera model-based feature transformation (CBFT) and camera model-free feature transformation (CFFT) in [22] to perform front BEV semantic segmentation tasks. The CBFT transforms features based on the flat-world assumption, which may distort regions lying above the ground plane. The CFFT is limited in the segmentation performance due to the absence of geometric priors and time-consuming computation. To combine the benefits and avoid the drawbacks of both, they proposed a hybrid feature transformation module (HFT) in which they decouple the feature maps produced by HFT for estimating the layout of outdoor scenes in BEV. A two-stage geometry prior-based transformation framework named GitNet [23] is proposed to perform the BEV semantic segmentation task. It consists of the geometry-guided pre-alignment and a ray-based transformer. In the first stage, Shi Gong et al. decoupled the BEV segmentation into the perspective image segmentation and geometric prior-based mapping, with explicit supervision by projecting the BEV semantic labels onto the image plane to learn visibility-aware features and learnable geometry to translate into the BEV space. Second, the pre-aligned coarse BEV features are further deformed by ray-based transformers to take visibility knowledge into account.

BEVFormer [24] learned unified BEV representations with spatio-temporal transformers to support multiple autonomous driving perception tasks: 3D detection and map segmentation based on multi-camera images. BEVFormer exploited both spatial and temporal information by interacting with spatial and temporal space through predefined grid-shaped BEV queries. Regarding Spatial information, Zhiqi Li et al. designed a spatial cross-attention in which each BEV query extracts the spatial features from the regions of interest across camera views. However, regarding the temporal information, they proposed a temporal self-attention to recurrently fuse the history of BEV information. BEVSegFormer [25] is proposed to have an effective transformer-based method for BEV semantic segmentation from arbitrary camera rigs. This method encoded image features from arbitrary cameras with a shared backbone. These image features were then enhanced by a deformable transformer-based encoder. Moreover, Lang Peng et al. introduced a BEV transformer decoder module to parse BEV semantic segmentation results. An efficient multi-camera deformable attention unit was designed to carry out the BEV-to-image view transformation. The queries were reshaped according to the layout of grids in the BEV and upsampled to produce the semantic segmentation result in a supervised manner.

Oskar Natan and Jun Miura proposed a BEV end-to-end and multi-task learning manners in [26] to perform both perception and control tasks simultaneously. The model aimed to drive the ego vehicle safely by following a sequence of routes defined by the global planner. The perception part of the model was used to encode high-dimensional observation data provided by an RGBD camera while performing semantic segmentation, semantic depth cloud mapping, and traffic light state and stop sign prediction. The control part then decoded the encoded features along with additional information provided using GPS and a speedometer to predict waypoints that come with a latent feature space. Enze Xie et al. proposed M^2BEV [27] which efficiently transformed multi-view 2D image features into the 3D BEV feature in ego-car coordinates. Likewise, this BEV representation is important as it enables different tasks to share a single encoder. This framework contains four important designs that benefit both accuracy and efficiency: (1) an efficient BEV encoder design that reduces the spatial dimension of a voxel feature map, (2) a dynamic box assignment strategy that uses learning-to-match to assign ground-truth 3D boxes with anchors, (3) a BEV centerness re-weighting that reinforces with

 doi:10.20944/preprints202405.0468.v1

larger weights for more distant predictions, and (4) large-scale 2D detection pre-training and auxiliary supervision.

CoBEVT [28], which is a multi-agent multi-camera perception framework, generates cooperatively BEV map predictions. This framework is a transformer-based architecture with a fused axial attention module (FAX). Spatial interactions are captured locally and globally across views and agents achieving state-of-the-art performance when tested on OPV2V [29] for BEV semantic segmentation. OPV2V is a large-scale V2V perception dataset that depends on the cooperation between OpenCDA [30] and CARLA [31], where OpenCDA is a simulator that aims to develop and test Cooperative Driving Automation (CDA) systems. V2VNet [32] is a V2V communication system that covers a 70m radius of vehicle connectivity, so vehicles can receive and publish information within this area. A spatial graph neural network (GNN) is used to collect information from all of the connected self-driving cars. It is also tested on V2X-Sim. COOPERNAUT [33] is an end-to-end model that cooperatively drives self-driving cars based on a cross-vehicle perception. It converts lidar information into a point-based representation that can be transmitted wirelessly between the connected vehicles. The authors also developed a CARLA-based framework AUTOCASTSIM with challenging accident-prone scenarios.

Wang et al. proposed DeepAccident [34] which is the first large-scale V2X autonomous driving dataset that includes various collision accident scenarios. They also introduced an end-to-end motion and accident prediction task with some measurement metrics to assess the accuracy of accident prediction. DeepAccident contains sensor data and annotation labels from four vehicles and infrastructure for each scenario, allowing V2X research for perception and prediction. The proposed DeepAccident also serves as a direct safety benchmark for autonomous driving algorithms and as a supplementary data set for single-vehicle and V2X perception research in safety-critical scenarios.

SDC-Net [35], which is an end-to-end multitask deep neural network, was designed to navigate safely by performing crash avoidance, path planning, and automatic emergency braking. Mohammed Abdou et al. proposed a full system that combines deep learning with IoT to achieve self-driving cars, they built their dataset using the CARLA simulator due to the lack of real benchmark datasets. There was an initial trial in merging IoT with deep learning in [36], however, it tackled only crash avoidance functionality. SDC-Net provided a comparative study between different input representations: front camera only, panorama, and BEV to be applied on a single crash avoidance-only task, a single path planning and AEB-only task, or multitask. As expected, they concluded that the multitask BEV experiment outperforms other ones in precision, recall, f1-score, accuracy, throttle MSE, steer MSE, and brake MSE.

3. System Methodology

This section tackles system methodology identifying the main modifications done on the SDC-Net system, especially in the detection network. Figure 3 shows SDC-Net++ system architecture in which the main modification is in the crash detection network, which is green. However, the other blocks are kept to formulate a comparative study between SDC-Net and our proposed system SDC-Net++.

 doi:10.20944/preprints202405.0468.v1

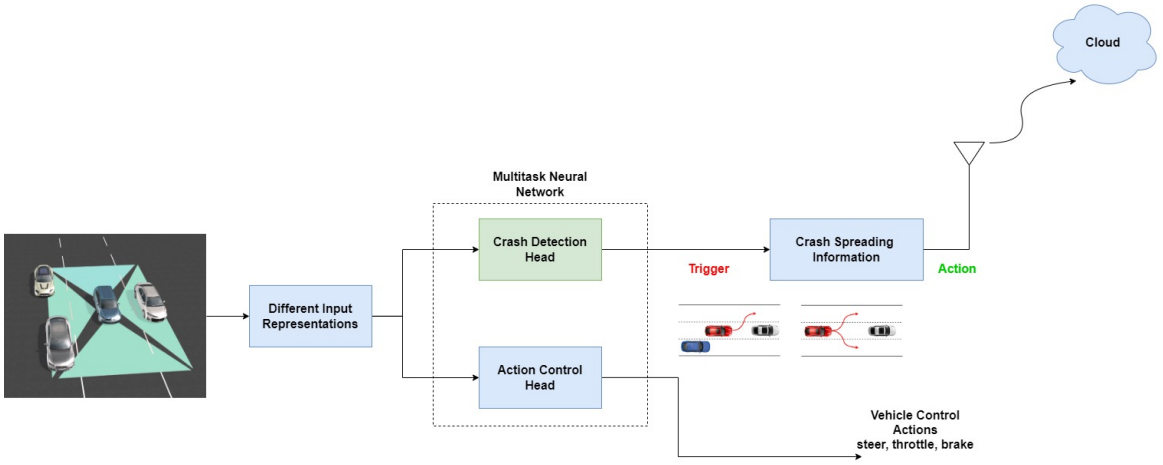

Figure 3. System Architecture

3.1. Multitask Deep Neural Network

Multitask deep neural network, inspired by [35] with some modifications, starts with an input normalization phase to ensure a zero mean and one standard deviation. The **feature extraction part** for the multitask network is made up of 10 convolution layers, the first having kernel size $(5,5)$, while the remaining 7 layers have kernel sizes $(3,3)$. The first 4 convolution layers have 32 and 64 filters for every two consecutive layers. The next 3 convolution layers have 128, 64, and 128 filters, while the last 3 convolution layers have 256, 128, and 256 filters. Every convolution layer is followed by ReLU activation function introducing nonlinearities. The first and second 2 convolution layers are followed by a max-pooling layer with a $(2,2)$ kernel size, while the third and fourth 3 convolution layers are followed by a max-pooling layer with a $(2,2)$ kernel size. The **crash detection head** is composed of 7 consecutive convolution layers in which the first 6 layers' kernel sizes are alternating between $(3,3)$, and $(1,1)$, and the applied filters are also alternating between 512 and 256 filters as shown in Figure 4. However, the last convolution layer is a 1-D convolution, kernel size $(1,1)$, with 18 filters reshaped to $(2,9)$ in which 9 regression values correspond to 1 for confidence, 3 for center point (x, y, z), 3 for dimensions (L, W, D), and 2 for two classes (C_0 or C_1). A Sigmoid activation function is finally applied to output the normalized continuous values for the crash/no crash bounding box and center with their confidence values. The **action control head** starts with receiving speed input from the CARLA simulator passed through 2 fully connected layers with 128 neurons each. Then, the output from the feature extraction part is flattened and passed through 2 fully connected layers with 256 neurons each. A concatenation layer is applied to the outcome from the extracted features from CARLA speed information and the feature extraction, after that 2 fully connected layers with 256 neurons are applied to relate the extracted features together. The last 2 fully connected layers with 10 and 3 neurons respectively are applied with a Sigmoid activation function layer to output the normalized continuous values for throttling, braking, and steering between 0 and 1. Table 1 shows each layer with the output shape, the number of parameters per layer, and the total number of parameters.

As SDC-Net++ design follows the multitask (two heads) networks, the total loss function for the network will be a combined loss between the two heads (loss from the action control part and loss for the detection part) as shown in equation 1.

$$\mathcal{L}_{combined} = \mathcal{L}_{control} + \mathcal{L}_{detection} \tag{1}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2024 doi:10.20944/preprints202405.0468.v1

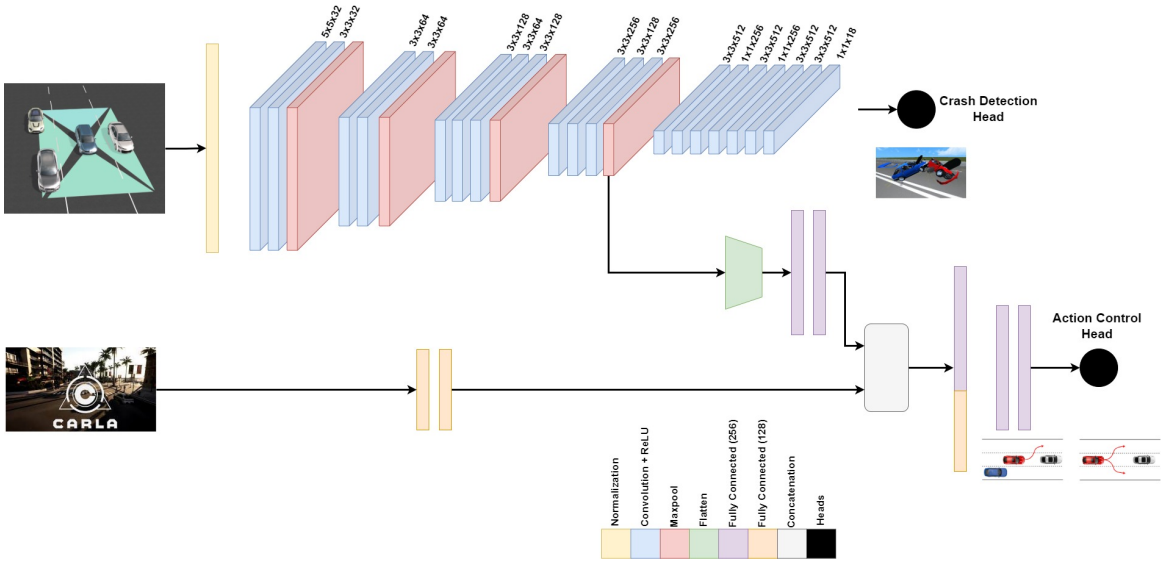

Figure 4. SDC-Net++: Multitask Deep Neural Network

Table 1. SDC-Net++ Network Parameters

Layer	Output Shape	Number of Parameters
Input Layer	(None, 120, 300, 3)	-
Conv1	(None, 120, 300, 32)	2,432
Conv2	(None, 120, 300, 32)	9,248
MaxPool1	(None, 60, 150, 32)	-
Conv3	(None, 60, 150, 64)	18,496
Conv4	(None, 60, 150, 64)	36,928
MaxPool2	(None, 30, 75, 64)	-
Conv5	(None, 30, 75, 128)	73,856
Conv6	(None, 30, 75, 64)	73,792
Conv7	(None, 30, 75, 128)	73,856
MaxPool3	(None, 15, 38, 128)	-
Conv8	(None, 15, 38, 256)	295,168
Conv9	(None, 15, 38, 128)	295,040
Conv10	(None, 15, 38, 256)	295,168
MaxPool4	(None, 8, 19, 256)	-
Crash Head(Conv11)	(None, 8, 19, 512)	1,180,160
Crash Head(Conv12)	(None, 8, 19, 256)	131,328
Crash Head(Conv13)	(None, 8, 19, 512)	1,180,160
Crash Head(Conv14)	(None, 8, 19, 256)	131,328
Crash Head(Conv15)	(None, 8, 19, 512)	1,180,160
Crash Head(Conv16)	(None, 8, 19, 512)	2,359,808
Crash Head(Conv17)	(None, 8, 19, 18)	9,234
Crash Head(Reshape)	(None, 8, 19, 2, 9)	-
Flatten	(None, 38,912)	-
FC1-256	(None, 256)	9,961,728
FC2-256	(None, 256)	65,792
Input Speed	(None, 1)	-
FC1-128	(None, 128)	256
FC2-128	(None, 128)	16,512
Concat (FC2-256,FC2-128)	(None, 384)	-
FC3-256	(None, 256)	98,560
FC4-256	(None, 256)	65,792
Control Head (FC1-10)	(None, 10)	2,570
Control Head (FC2-3)	(None, 3)	33
Sigmoid	(None, 3)	-
		17,557,405

 doi:10.20944/preprints202405.0468.v1

The action control loss function is a typical mean square error function as shown in equation 2. In which, m normalized samples whose iterator is i, $\hat{Y}_i$ is the output prediction from the regression model, Y_i is the continuous ground truth labels (throttle, steer, brake) corresponding to the inputs, MSE is the mean square error. However, the detection loss function combines the 4 components shown in equation 3: confidence, bounding box center, bounding box dimensions, and classification components.

$$\mathcal{L}_{control} = MSE = \frac{1}{m}\sum_{i=1}^{i=m}(\hat{Y}_i - Y_i)^2 \tag{2}$$

$$\mathcal{L}_{detection} = \mathcal{L}_{conf} + \mathcal{L}_{center} + \mathcal{L}_{dimensions} + \mathcal{L}_{classification} \tag{3}$$

The 4 loss components functions are shown in equations 4, 5, 6, and 7. In which, λ_{conf}, λ_{coor}, and λ_{class} are the weights for confidence prediction, coordinates, and probabilities of the classifications respectively, S^2 is the (SxS) grid cells in which the images are decomposed whose iterator is i, B bounding boxes predicted in each grid cell whose iterator is j, I_{ij}^{obj} is a binary variable that takes values $(0, 1)$ as per the ground truth box is in i^{th} and j^{th} location in which 1 in case of there is a box and 0 is otherwise, I_{ij}^{Noobj} is always the opposite of I_{ij}^{obj}, C_i and $\hat{C}_i$ are the predicted and ground truth of the confidence, (x_i, y_i, z_i) and $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$ are the predicted and ground truth of the box center in the 3-D coordinates, (w_i, l_i, d_i) and $(\hat{w}_i, \hat{l}_i, \hat{d}_i)$ are the predicted and ground truth of the box dimensions (width, length, depth), $p_i(c)$, and $\hat{p}_i(c)$ are the predicted and ground truth probabilities of the classes.

$$\mathcal{L}_{conf} = \lambda_{conf}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{I}_{ij}^{obj}(C_i - \hat{C}_i)^2 + \lambda_{conf}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{I}_{ij}^{Noobj}(C_i - \hat{C}_i)^2 \tag{4}$$

$$\mathcal{L}_{center} = \lambda_{coor}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{I}_{ij}^{obj}[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (z_i - \hat{z}_i)^2] \tag{5}$$

$$\mathcal{L}_{dimensions} = \lambda_{coor}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{I}_{ij}^{obj}[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{l_i} - \sqrt{\hat{l}_i})^2 + (\sqrt{d_i} - \sqrt{\hat{d}_i})^2] \tag{6}$$

$$\mathcal{L}_{classifications} = \lambda_{class}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{I}_{ij}^{obj}\sum_{c\in classes}(p_i(c) - \hat{p}_i(c))^2 \tag{7}$$

3.2. Input Representations and IoT Integration

SDC-Net uses different input representations to formulate a comparative study and identify which of them is the best one. These input representations are front camera view, panorama with either normal or equirectangular stitching view, and Bird's eye view. Front camera view means that the SDC-Net takes the front camera image only from the camera cocoon as an input image to the system; however, panorama views take the four camera images of the cocoon, then either stitch them together or tilt the edges of every image to isolate between them. Bird's eye view depends on the intrinsic and extrinsic camera calibration parameters to warp the four cocoon images forming the BEV shape. SDC-Net++ keeps the same input representations to compare with the SDC-Net results.

SDC-Net uses a trigger-action digital automation platform like Zapier, IFTTT, or Integromat, so in case of an accident (trigger), the system provides connected vehicles with the accident information via IoT. SDC-Net is also integrated with Valeo's standardized platform specific for automotive that uses digital gate and let's do (LDO) platforms. SDC-Net++ also maintains the same IoT solution to share the relevant information about the accident, but with the location of the accident (which lane) as additional information. The additional information about the accident location comes out of the

 doi:10.20944/preprints202405.0468.v1

modification of changing the classification network to be a detection network so that the accident lane could be identified easily.

4. Dataset Setup

Likewise SDC-Net, due to the lack of datasets that serve our multitask system of controlling vehicle action and detecting crashes, we created our dataset using the CARLA simulator. SDC-Net++ uses exactly the same dataset proposed by SDC-Net to achieve a fair comparative study, so this section tackles the modification done on the dataset labels.

4.1. CARLA Adaptations

SDC-Net is implemented to have a classification network that can identify if the scene contains an accident or not, so the constructed dataset labels were a binary classification (0, 1) means no accident, and there is an accident respectively. However, the SDC-Net++ detection network needs the bounding boxes generated from the CARLA simulator and then applies some processing over them to save the crash label. The flow chart Figure 5 shows the accident detection label and the steps are also described as follows:

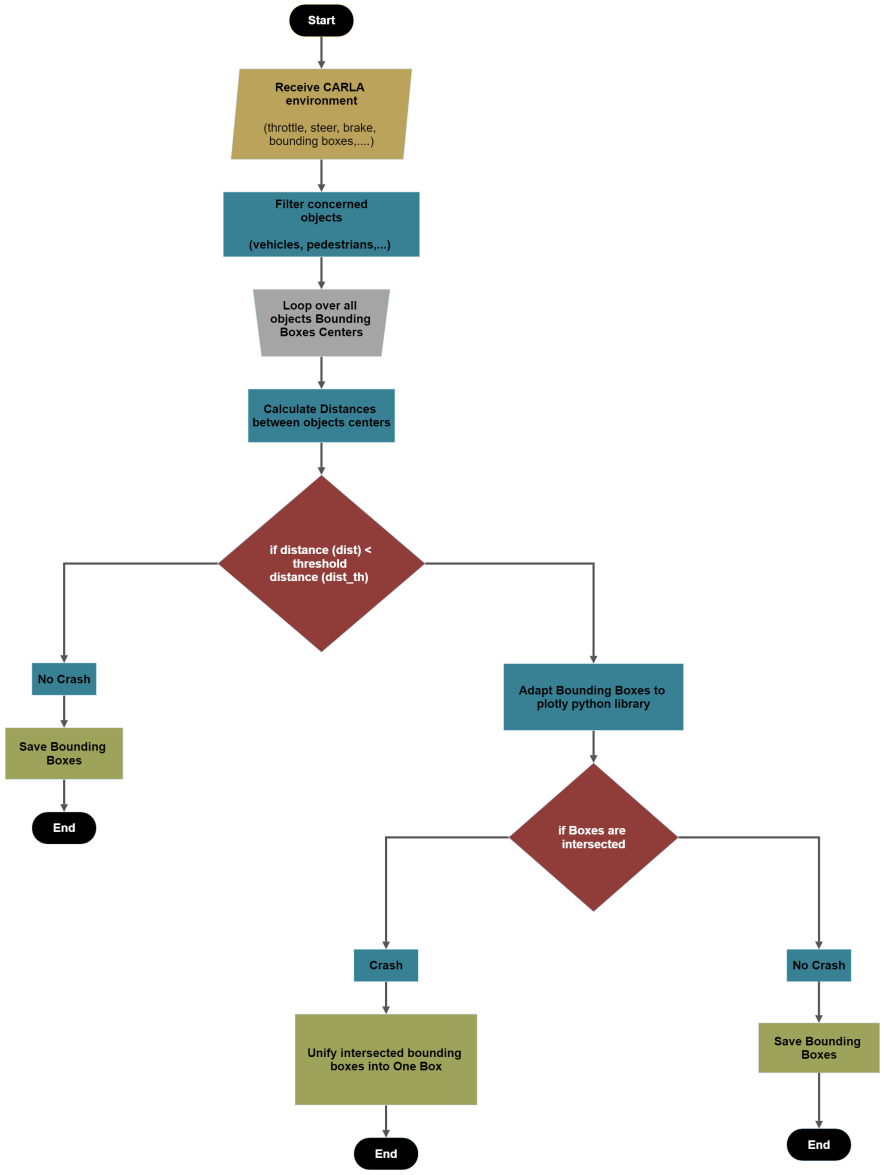

Figure 5. Crash labels processing flowchart

 doi:10.20944/preprints202405.0468.v1

- Filter the objects by keeping only the concerned objects such as vehicles, pedestrians, etc.
- Loop over all the bounding box centers received from the CARLA simulator;
- Calculate the distances between bounding box centers;
- Check if the distances are greater than the threshold tunable distance. If yes, no crash label is applied; however, if no, this means that we have two or more vehicle centers in close proximity to each other;
- Adapt the bounding boxes information to the plotly [37] python library to check if there are two intersecting boxes;
- Check if the boxes are intersecting. If yes, unify the two intersected bounding boxes in one box throughout the Min-Max boxes' boundaries as shown in Figure 6. If no, save the bounding boxes.

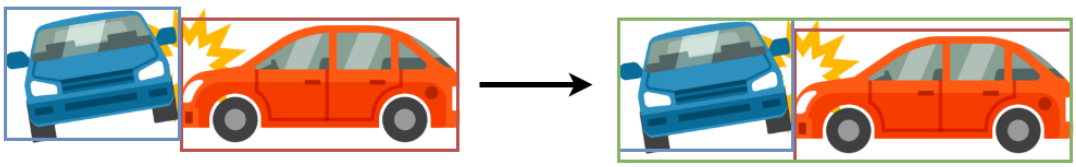

Figure 6. Unified Boxes

4.2. Data Analysis

Our Dataset is collected using the CARLA simulator. The total number of samples is **125K** samples in which each sample consists of 4 synchronized images: front, left, right, and rear. Around **62.5%** of the samples are used as training data, **10.7%** for validation, and **26.8%** for testing as shown in Figure 7. Training data is decomposed into **56.2%** with no crashes and **43.8%** with crash samples as in Figure 8. Validation data is decomposed into **60%** with no crash and **40%** with a crash as in Figure 9. Testing data is decomposed into **50%** with and without crashes as in Figure 10.

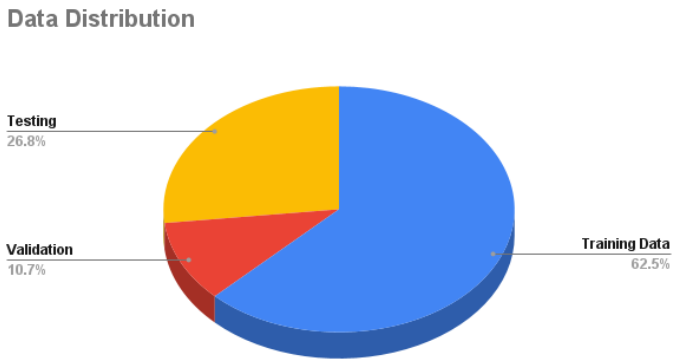

Figure 7. Dataset Distribution

 doi:10.20944/preprints202405.0468.v1

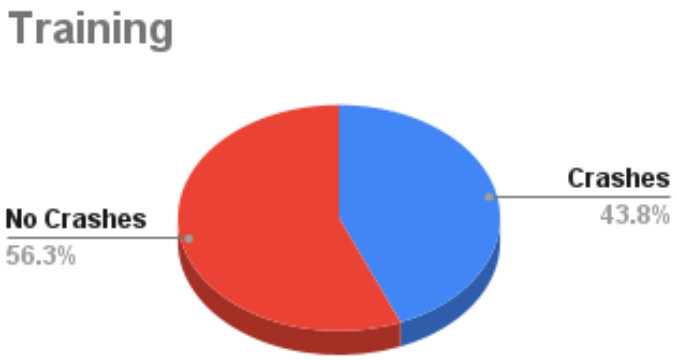

Figure 8. Training Data

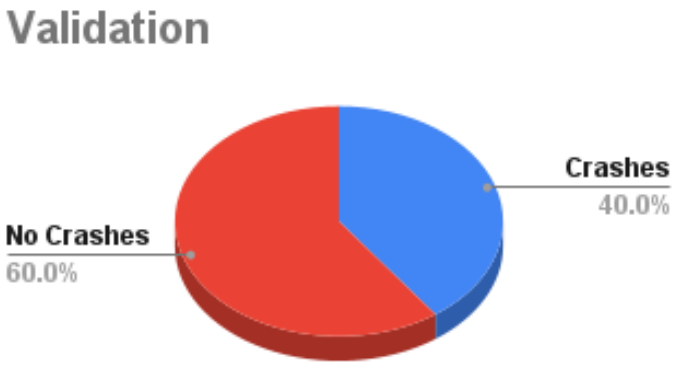

Figure 9. Validation Data

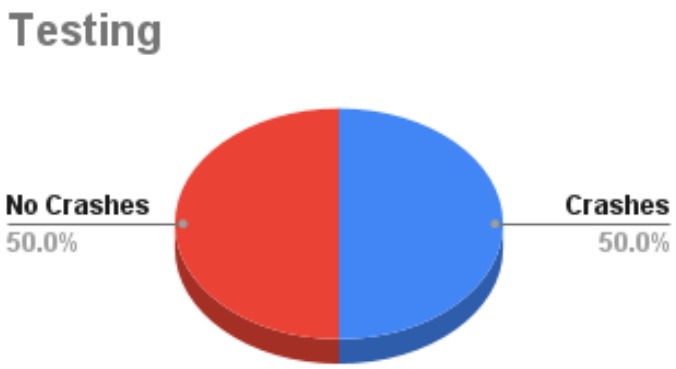

Figure 10. Testing Data

5. Results

Likewise SDC-Net, the conducted experiments are decomposed into 2 variants: different input view and single-task network or multitask output. The first variant (input view) is categorized into different views: front camera, panorama, and BEV views; however, the second variant is categorized into running using a single/multiple output head network. As long as SDC-Net++ replaces the classification network with a detection one, the comparative study will tackle only: crash avoidance only and the multitask network (crash avoidance with path planning and AEB) because there is no

 doi:10.20944/preprints202405.0468.v1

meaning to run the path planning and AEB only because it will be identical to SDC-Net conducted experiments. This section covers the SDC-Net++ conducted experiments to compare the different input representations as shown in Table 2 in which the rows show the different input representations and columns show single/multi-task networks. The main measurement metrics reported are identical to SDC-Net: precision, recall, f1-score, accuracy for the detection network, and mean square error (MSE) in the control action regression.

5.1. Crash Detection Only Results

The crash avoidance-only experiment is a single-head detection output network in which the experiment is conducted three times: the first uses the front camera only, the second uses the panorama view, and the last one uses the BEV. Due to the detection problem, the measurement metrics can be intersection over union (IoU), however to compare with SDC-Net, the measurement metrics are only precision, recall, f1-score, and accuracy. It is obvious that the BEV results are better than the panorama camera and front camera results in precision by 15.197% and 33.203%, respectively; in recall by 15.273% and 44.374%, respectively; in f1-score by 15.246% and 38.867%, respectively; and in accuracy by 5% and 14%, respectively.

5.2. Multitask Results

Crash avoidance, path planning, and AEB experiments are multi-head detection-regression output networks (multitask) in which the experiments are conducted three times: the first uses the front camera only, the second uses the panorama view, and the last one uses the BEV. Due to the multitask problem, the measurement metrics are the already previously mentioned ones of precision, recall, f1-score, and accuracy, in addition to the control actions MSE. It is obvious that the BEV results are better than panorama camera (experiments conducted in SDC-Net) and front camera results in precision by 15.134% and 27.212%, respectively; in recall by 12.046% and 28.706%, respectively; in f1-score by 13.593% and 27.962%, respectively; in accuracy by 5% and 13%, respectively; in throttle MSE by 32.97% and 48.835%, respectively; in steer MSE by 29.173% and 51.624%, respectively; and in brake MSE by 36.578% and 55.924% respectively.

 doi:10.20944/preprints202405.0468.v1

Table 2. Measurement Metrics Comparison.

		Experiments														
		Crash Avoidance Only				Path Planning and AEB Only			Crash Avoidance, Path Planning and AEB							
		Precision	Recall	F1-Score	Accuracy	MSE throttle	MSE Steer	MSE brake	Precision	Recall	F1-Score	Accuracy	MSE throttle	MSE Steer	MSE brake
Input Representations	Front Camera	0.5662	0.5102	0.5367	0.73	0.2214	0.2552	0.3285	0.7188	0.7075	0.7131	0.81	0.1804	0.1877	0.2557
	Panorama	0.6547	0.639	0.6467	0.82	0.2176	0.2333	0.2805	0.7942	0.8127	0.8033	0.89	0.1377	0.1282	0.1777
	Bird Eye View (BEV)	0.7542	0.7366	0.7453	0.87	0.1988	0.2279	0.2794	**0.9144**	**0.9106**	**0.9125**	**0.94**	**0.0923**	**0.0908**	**0.1127**

 doi:10.20944/preprints202405.0468.v1

6. Discussion

This technical discussion section is decomposed into two sub-sections: the first sub-section tackles the SDC-Net++ results only, however, the second sub-section tackles the comparison between SDC-Net and SDC-Net++ experimental results.

6.1. SDC-Net++ Results Discussion

Regarding **crash avoidance-only experiments**, the front camera-only experiments detect the accidents in front of the ego-vehicle only, however, it is not able to detect accidents beside and behind. This is due to the absence of sensors that can visually see behind and beside the ego vehicle, so it was expected that the measurement metrics were not so good. Panorama and BEV experiments have better results because depending on the camera cocoon setup that covers 360^o around the ego vehicle, in case of an accident that happened beside or behind the vehicle, the sensors will be able to visually see it. **Why did the BEV experiments outperform the panorama experiments?** Panorama depends on stitching the raw cocoon images together; however, BEV depends on warping the cocoon images as a pre-processing phase before input to the network. BEV indirectly includes the distance factor because of warping, this facilitates detecting collision.

Regarding **path planning and AEB experiments**, front camera camera-only experiments take control action according to the front image only; however, it fails because it visually cannot see other vehicles besides the ego vehicle, so it takes inaccurate control actions (unsafe) that were expected due to lack of information about the environment. Panorama and BEV experiments have better results due to the different camera cocoon setups that cover 360^o around the ego vehicle so that in the case of another vehicle in addition to the ego vehicle, the ego vehicle will not be able to maneuver. **Again, we must ask why the BEV experiments outperformed the panorama experiments?** BEV depends on warping the cocoon images in which the warped front image becomes in the upper part of the total image, the warped left image becomes in the left part of the total image, the warped right image becomes in the right part of the total image, and the rear warped image becomes in the lower part of the total image. This setup facilitates extracting the environment features compared to stitching the raw cocoon images together, as in the panorama view.

The multitask network outperforms the single task in both detection and regression measurement metrics. Figure 11 shows the bar charts for precision, recall, f1-score, and accuracy; multitask with the BEV is the best experiment metric. Figure 12 shows the bar charts for the MSE of throttle, brake, and steer for the conducted experiments; also multitask with BEV is the best lower experiment metric. Conceptually, multitask networks prove their efficiency compared with single-task ones because the presence of two or more tasks supports each other to learn in a better way.

 doi:10.20944/preprints202405.0468.v1

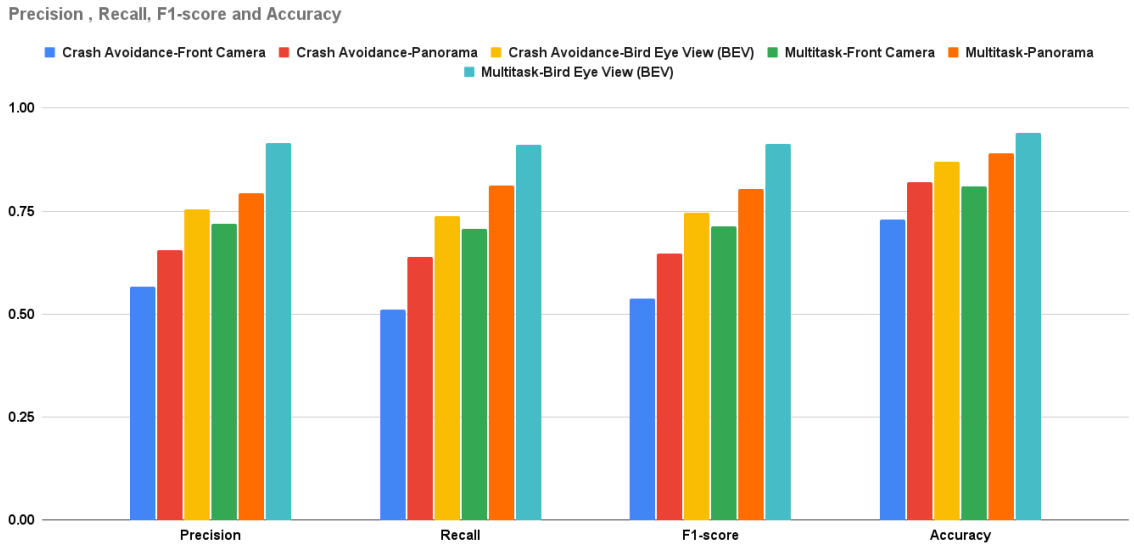

Figure 11. Precision, recall, f1-score, and accuracy for crash avoidance only vs. multitask

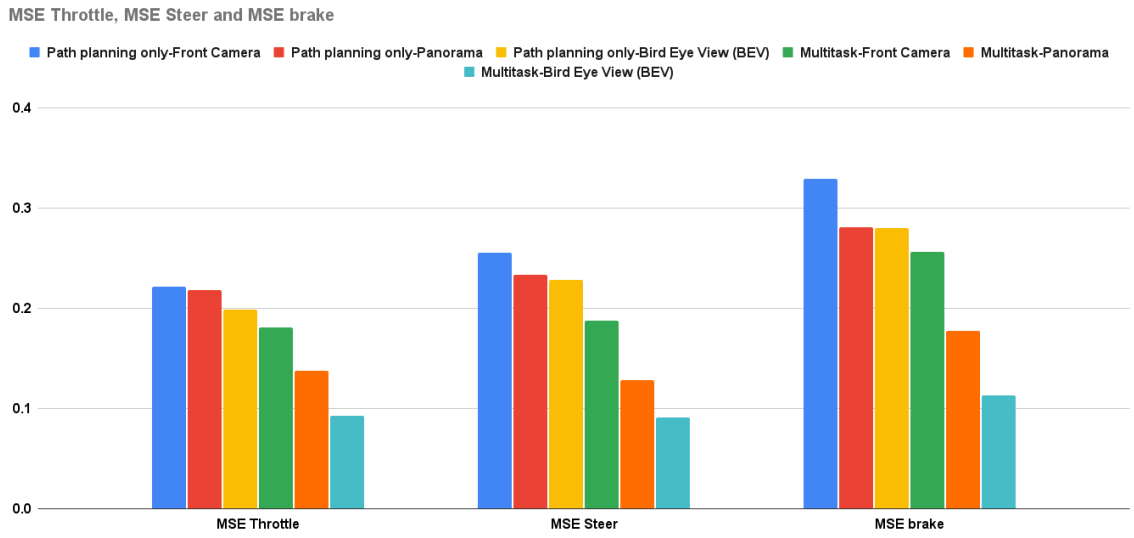

Figure 12. MSE throttle, steer, and brake for path planning, AEB only vs. multitask

6.2. SDC-Net and SDC-Net++ Results Discussion

Table 3 provides the combined experimental results for both SDC-Net and SDC-Net++. Regarding **crash avoidance only experiments** with **front camera** input, it is obvious that SDC-Net++ outperforms SDC-Net in precision by 2.702%, recall by 6.203%, f1-score by 4.538%, and accuracy by 2%, with **panorama** view input, SDC-Net++ outperforms SDC-Net in precision by 5.291%, recall by 5.794%, f1-score by 5.549%, and accuracy by 4%, and with **BEV** input, SDC-Net++ also outperforms SDC-Net in precision by 6.135%, recall by 9.694%, f1-score by 7.951%, and accuracy by 5%. Regarding **multitask** with **front camera** input, it is obvious that SDC-Net++ outperforms SDC-Net in precision by 9.090%, recall by 17.916%, f1-score by 13.55%, and accuracy by 2%, with **panorama** view input, SDC-Net++ outperforms SDC-Net in precision by 2.016%, recall by 2.678%, f1-score by 2.344%, and accuracy by 3%, and with **BEV** input, SDC-Net++ also outperforms SDC-Net in precision by 2.201%, recall by 2.8%, f1-score by 2.505%, and accuracy by 2%. Regarding **multitask** with **front camera** input, it is also obvious that SDC-Net++ outperforms SDC-Net in throttle MSE by 6.673%, steer MSE by 1.314%, and

 doi:10.20944/preprints202405.0468.v1

brake MSE by 3.691%, with **panorama** view input, SDC-Net++ outperforms SDC-Net in throttle MSE by 11.275%, steer MSE by 9.335%, and brake MSE by 9.613%, and with **BEV** input, SDC-Net++ also outperforms SDC-Net in throttle MSE by 18.678%, steer MSE by 16%, and brake MSE by 21.353%.

SDC-Net++ outperforms SDC-Net in both detection and regression measurement metrics. Figure 13 shows the bar charts for precision, recall, f1-score, and accuracy for both SDC-Net and SDC-Net in the case of crash avoidance only. Figure 14 shows the bar charts for the MSE of throttle, brake, and steer for both SDC-Net and SDC-Net++. Thus, Figure 15 shows the bar charts for precision, recall, f1-score, and accuracy for both SDC-Net and SDC-Net in the case of multitask. Conceptually, multitask networks prove their efficiency compared with single-task ones because the presence of two or more tasks supports each other to learn in a better way. Finally, SDC-Net++ detection network enhances the measurement metrics compared with the classification one in SDC-Net, this is because

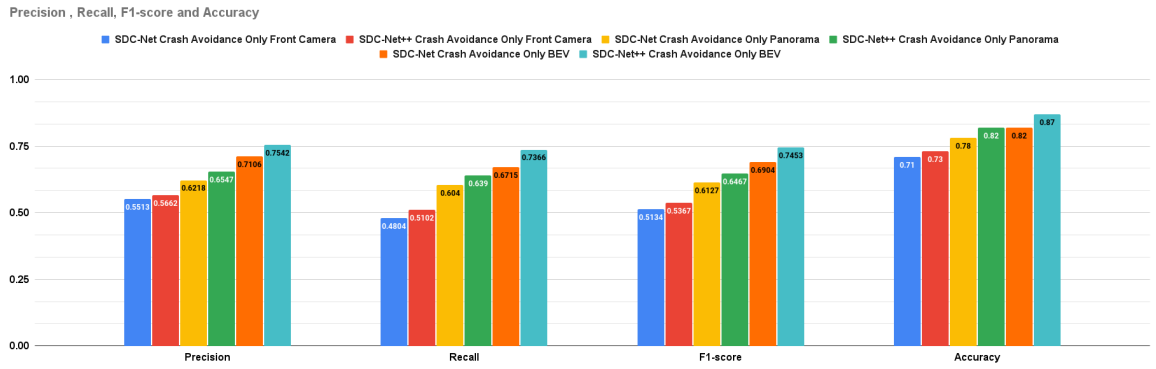

Figure 13. Detection Metrics for SDC-Net vs. SDC-Net++ with crash avoidance only

- directing the labels to detect where exactly the accident (bounding box) is in the scene rather than the classification which has only a binary label for the scene if there is an accident or not.
- enlarging the size of the network, so enlarging the trainable parameters which means that SDC-Net design was not able to extract the most important features (accident features) in the input scene.

Table 3. SDC-Net vs. SDC-Net++

		Experiments													
		Crash Avoidance Only				Path Planning and AEB Only			Multitask						
		Precision	Recall	F1-Score	Accuracy	MSE throttle	MSE Steer	MSE brake	Precision	Recall	F1-Score	Accuracy	MSE throttle	MSE Steer	MSE brake
SDC-Net	Front Camera	0.5513	0.4804	0.5134	0.71	0.2214	0.2552	0.3285	0.6589	0.6	0.628	0.79	0.1933	0.1902	0.2655
	Panorama	0.6218	0.604	0.6127	0.78	0.2176	0.2333	0.2805	0.7785	0.7915	0.7849	0.86	0.1552	0.1414	0.1966
	BEV	0.7106	0.6715	0.6904	0.82	0.1988	0.2279	0.2794	**0.8947**	**0.8858**	**0.8902**	**0.92**	**0.1135**	**0.1081**	**0.1433**
SDC-Net++	Front Camera	0.5662	0.5102	0.5367	0.73	-	-	-	0.7188	0.7075	0.7131	0.81	0.1804	0.1877	0.2557
	Panorama	0.6547	0.639	0.6467	0.82	-	-	-	0.7942	0.8127	0.8033	0.89	0.1377	0.1282	0.1777
	BEV	0.7542	0.7366	0.7453	0.87	-	-	-	**0.9144**	**0.9106**	**0.9125**	**0.94**	**0.0923**	**0.0908**	**0.1127**

 doi:10.20944/preprints202405.0468.v1

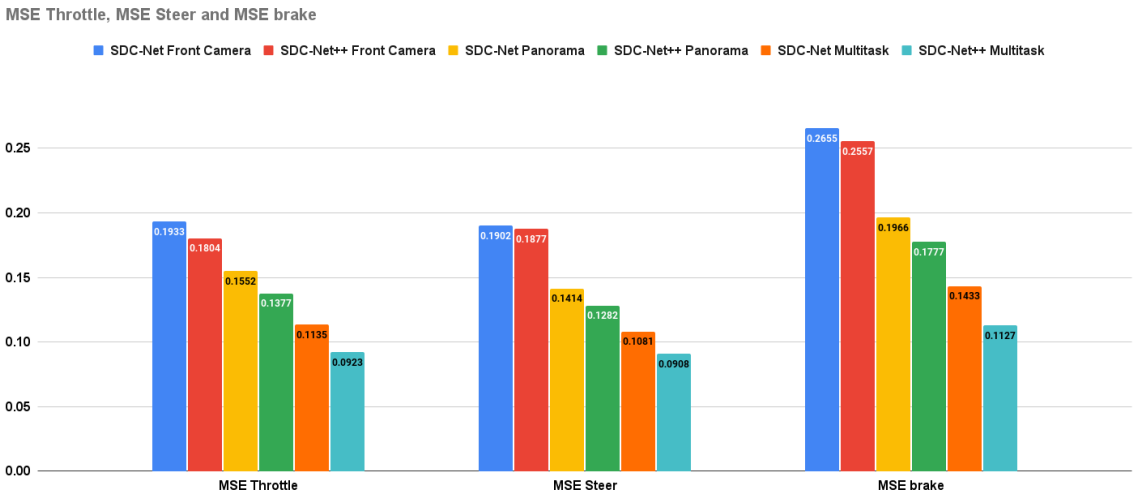

Figure 14. Regression Metrics for SDC-Net vs. SDC-Net++

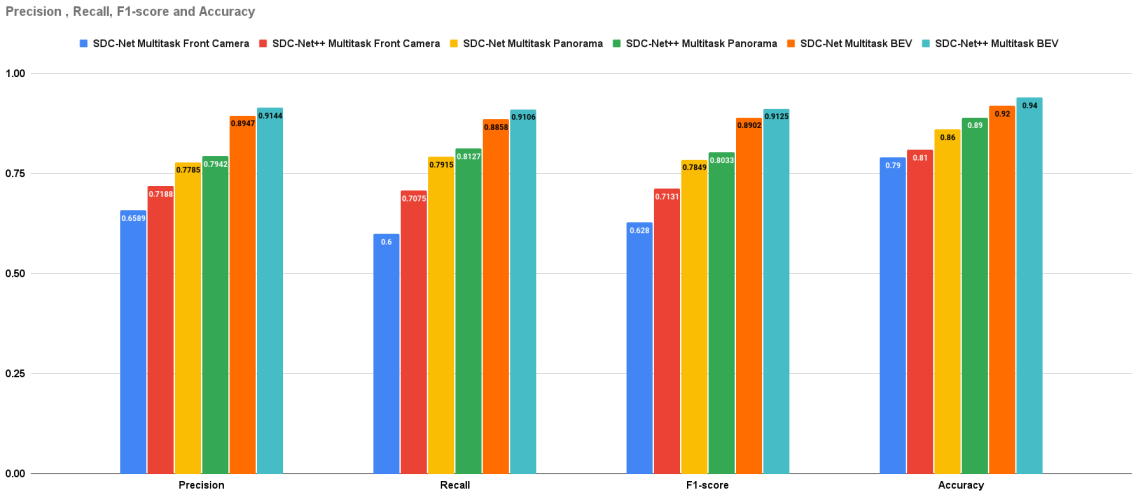

Figure 15. Detection Metrics for SDC-Net vs. SDC-Net++ with Multitask

7. Conclusions

Currently, self-driving cars harness the power of the artificial intelligence (AI). Self-driving cars benefit from sensor fusion to perceive and understand the surrounding environment. IoT plays a very vital role in functioning self-driving cars, as it connects vehicles wirelessly sharing environment on-board sensors' information. Autonomous vehicles utilize this connectivity to update the driving algorithms as a sort of incremental enhancement. Our proposed system, SDC-Net++ which is an end-to-end crash detection and action control self-driving car system, is proposed to enhance a prior study, SDC-Net system, that tackles exactly the same problem. SDC-Net++ replaced the classification network with a detection one keeping the same benchmark dataset to achieve a fair comparative study. SDC-Net++ with BEV input outperforms the nearest representation in precision, recall, f1-score, and accuracy by more than 15.134%, 12.046%, 13.593%, and 5%, respectively. It also outperforms SDC-Net in precision, recall, f1-score, accuracy, and average MSE by more than 2.201%, 2.8%, 2.505%, 2%, and 18.677% respectively.

Author Contributions: Conceptualization, Mohammed Abdou and; methodology, Mohammed Abdou; software, Mohammed Abdou; validation, Mohammed Abdou; investigation, Mohammed Abdou; data curation,

Mohammed Abdou; writing—original draft preparation, Mohammed Abdou; writing—review and editing, Mohammed Abdou and Hanan Kamal; visualization, Mohammed Abdou; supervision, Hanan Kamal; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Herrmann, A.; Brenner, W.; Stadler, R. *Autonomous driving: how the driverless revolution will change the world*; Emerald Group Publishing: 2018.
2. Kamal, M.A.S.; Hashikura, K.; Hayakawa, T.; Yamada, K.; Imura, J.i. Adaptive Cruise Control with Look-Ahead Anticipation for Driving on Freeways. *Appl. Sci.* **2022**, *12*, 929. https://doi.org/10.3390/app12020929.
3. Greenwood, P.M.; Lenneman, J.K.; Baldwin, C.L. Advanced driver assistance systems (ADAS): Demographics, preferred sources of information, and accuracy of ADAS knowledge. *Transp. Res. Part Traffic Psychol. Behav.* **2022**, *86*, 131–150. https://doi.org/10.1016/j.trf.2021.08.006.
4. Shao, X.; Wang, Q.; Yang, H. Business Analysis and Future Development of an Electric Vehicle Company–Tesla. *Int. Conf. Public Relations Soc. Sci. ICPRSS* **2021**.
5. Boudette, N.E.; Davenport, C. GM will sell only zero-emission vehicles by 2035. *The New York Times* **2021**.
6. Pelliccione, P.; Knauss, E.; Heldal, R.; Ågren, S.M.; Mallozzi, P.; Alminger, A.; Borgentun, D. Automotive architecture framework: The experience of volvo cars. *J. Syst. Archit.* **2017**, *77*, 83–100. https://doi.org/10.1016/j.sysarc.2017.02.005.
7. Schwall, M.; Daniel, T.; Victor, T.; Favaro, F.; Hohnhold, H. Waymo public road safety performance data. *arXiv* **2020**. arXiv:2011.00038
8. Smith, C.S. Baidu and Geely Will Mass-Produce an Autonomous EV: The Chinese tech giants aim for a fully self-driving car. *IEEE Spectrum* **2023**, *60*, 36–37. https://doi.org/10.1109/MSPEC.2023.10006688.
9. Xu, X.; Zhang, L.; Yang, J.; Cao, C.; Wang, W.; Ran, Y.; Tan, Z.; Luo, M. A Review of Multi-Sensor Fusion SLAM Systems Based on 3D LIDAR. *Remote. Sens.* **2022**, *14*, 2835. https://doi.org/10.3390/rs14122835.
10. Sang, H.; You, Y.; Sun, X.; Zhou, Y.; Liu, F. The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations. *Ocean. Eng.* **2021**, *223*, 108709. https://doi.org/10.1016/j.oceaneng.2021.108709.
11. Lin, L.; Li, W.; Bi, H.; Qin, L. Vehicle Trajectory Prediction Using LSTMs With Spatial–Temporal Attention Mechanisms. *IEEE Intell. Transp. Syst. Mag.* **2021**, *14*, 197–208.
12. Wang, C.; Chen, X.; Wang, J.; Wang, H. ATPFL: Automatic Trajectory Prediction Model Design Under Federated Learning Framework. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 6563–6572.
13. Quintanar, A.; Fernández-Llorca, D.; Parra, I.; Izquierdo, R.; Sotelo, M. Predicting vehicles trajectories in urban scenarios with transformer networks and augmented information. In Proceedings of the 2021 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2021; pp. 1051–1056.
14. Abdou, M.; Kamal, H.; El-Tantawy, S.; Abdelkhalek, A.; Adel, O.; Hamdy, K.; Abaas, M. End-to-end deep conditional imitation learning for autonomous driving. In Proceedings of the 2019 31st International Conference on Microelectronics (ICM). IEEE, 2019; pp. 346–350.
15. Guo, K.; Liu, W.; Pan, J. End-to-End Trajectory Distribution Prediction Based on Occupancy Grid Maps. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; pp. 2242–2251.
16. Sallab, A.E.; Abdou, M.; Perot, E.; Yogamani, S. Deep reinforcement learning framework for autonomous driving. *arXiv* **2017**. arXiv:1704.02532.
17. Sallab, A.; Abdou, M.; Perot, E.; Yogamani, S. End-to-end deep reinforcement learning for lane keeping assist. *arXiv* **2016**. arXiv:1612.04340.
18. Espié, E.; Guionneau, C.; Wymann, B.; Dimitrakakis, C.; Coulom, R.; Sumner, A. TORCS, The Open Racing Car Simulator. 2005.

19. Naumann, M.; Poggenhans, F.; Lauer, M.; Stiller, C. Coincar-sim: An open-source simulation framework for cooperatively interacting automobiles. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018, pp. 1–6.
20. Loc, H.D.; Kim, G.W. Fast and Accurate Deep Learning-Based Framework for 3D Multi-Object Detector for Autonomous Vehicles. In Proceedings of the 2022 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE, 2022, pp. 320–322.
21. Hu, H.; Zhu, M.; Li, M.; Chan, K.L. Deep Learning-Based Monocular 3D Object Detection with Refinement of Depth Information. *Sensors* **2022**, *22*, 2576.
22. Zou, J.; Xiao, J.; Zhu, Z.; Huang, J.; Huang, G.; Du, D.; Wang, X. HFT: Lifting Perspective Representations via Hybrid Feature Transformation. *arXiv* **2022**. arXiv:2204.05068
23. Gong, S.; Ye, X.; Tan, X.; Wang, J.; Ding, E.; Zhou, Y.; Bai, X. GitNet: Geometric Prior-based Transformation for Birds-Eye-View Segmentation. *arXiv* **2022**. arXiv:2204.07733
24. Li, Z.; Wang, W.; Li, H.; Xie, E.; Sima, C.; Lu, T.; Yu, Q.; Dai, J. BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers. *arXiv* **2022**. arXiv:2203.17270
25. Peng, L.; Chen, Z.; Fu, Z.; Liang, P.; Cheng, E. BEVSegFormer: Bird's Eye View Semantic Segmentation From Arbitrary Camera Rigs. *arXiv* **2022**. arXiv:2203.04050.
26. Natan, O.; Miura, J. Fully End-to-end Autonomous Driving with Semantic Depth Cloud Mapping and Multi-Agent. *arXiv* **2022**. arXiv:2204.05513.
27. Xie, E.; Yu, Z.; Zhou, D.; Philion, J.; Anandkumar, A.; Fidler, S.; Luo, P.; Alvarez, J.M. Mˆ 2BEV: Multi-Camera Joint 3D Detection and Segmentation with Unified Birds-Eye View Representation. *arXiv* **2022**. arXiv:2204.05088.
28. Xu, R.; Tu, Z.; Xiang, H.; Shao, W.; Zhou, B.; Ma, J. CoBEVT: Cooperative bird's eye view semantic segmentation with sparse transformers. *arXiv* **2022**. arXiv:2207.02202.
29. Xu, R.; Xiang, H.; Xia, X.; Han, X.; Li, J.; Ma, J. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022; pp. 2583–2589.
30. Xu, R.; Guo, Y.; Han, X.; Xia, X.; Xiang, H.; Ma, J. OpenCDA: an open cooperative driving automation framework integrated with co-simulation. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE, 2021; pp. 1155–1162.
31. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An open urban driving simulator. In Proceedings of the Conference on robot learning. PMLR, 2017; pp. 1–16.
32. Wang, T.H.; Manivasagam, S.; Liang, M.; Yang, B.; Zeng, W.; Urtasun, R. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In Proceedings of the European Conference on Computer Vision. Springer, 2020; pp. 605–621.
33. Cui, J.; Qiu, H.; Chen, D.; Stone, P.; Zhu, Y. COOPERNAUT: End-to-End Driving with Cooperative Perception for Networked Vehicles. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; pp. 17252–17262.
34. Wang, T.; Kim, S.; Ji, W.; Xie, E.; Ge, C.; Chen, J.; Li, Z.; Ping, L. DeepAccident: A Motion and Accident Prediction Benchmark for V2X Autonomous Driving. *arXiv* **2023**. arXiv:2304.01168.
35. Abdou, M.; Kamal, H.A. SDC-Net: End-to-End Multitask Self-Driving Car Camera Cocoon IoT-Based System. *Sensors* **2022**, *22*, 9108.
36. Abdou, M.; Mohammed, R.; Hosny, Z.; Essam, M.; Zaki, M.; Hassan, M.; Eid, M.; Mostafa, H. End-to-end crash avoidance deep IoT-based solution. In Proceedings of the 2019 31st International Conference on Microelectronics (ICM). IEEE, 2019; pp. 103–107.
37. Inc., P.T. Collaborative data science. https://plot.ly, 2015.