Vehicle pose estimation using CenterNet on monocular street images and recommend suitable driving skill

Manmeet Kumar Chaudhuri
SRH Hochschule Heidelberg
manmeetkumar.chaudhuri@stud.hochschule-heidelberg.de

Rishabh Garg
SRH Hochschule Heidelberg
rishabh.garg@stud.hochschule-heidelberg.de

Rohit Keshav Bewoor
SRH Hochschule Heidelberg
rohitkeshav.bewoor@stud.hochschule-heidelberg.de

Sanika Medankar
SRH Hochschule Heidelberg
Sanika.Medankar@stud.hochschule-heidelberg.de

Shekhar Singh
SRH Hochschule Heidelberg
Shekhar.Singh@stud.hocschule-heidelberg.de

ABSTRACT

3D object detection, i.e. estimating the absolute pose of vehicles (6 degrees of freedom i.e. 6 DoF) from a 2D image/monocular image, is a challenging task as the critical information like depth, yaw, pitch and roll are collapsed in the 2D image. The most popular one-stage object detection algorithms, e.g. YOLO, SSD, etc. and two-stage object detection algorithms e.g. RCNN, Mask RCNN, etc. rely on anchors to refine to the final detection location. Anchor based approaches are inherently slower and lack accuracy. Anchor free approach using CenterNet (1) is accurate and fast in object detection and pose estimation as compared to anchor based approaches. CenterNet (1) is evaluated here in estimating the absolute pose of vehicles from a monocular image in a real-world traffic environment. Kaggle dataset (2) (3) provided by Peking University and Baidu is used. CenterNet (1) models with two backbones ResNet-18 (4) and ResNeXt-50_32 (5). The performance of these models is compared. We then use the information about the cars detected in the environment, to recommend a skill level (Beginner/ Medium/ Advanced) for a driver to take manual control from a hypothetical autonomous car. This determination is done based on the proximity and density of detected traffic.

1. INTRODUCTION

3D object detection is the most important task in autonomous driving. Several functions in autonomous driving like object detection, identification, motion control, etc. are dependent on precise estimation of the 3D environment around the vehicle.

There has been a lot of research in using LiDAR-based technologies to generate a point cloud for precise 3D object detection. However, this approach has certain drawbacks like high cost and low accuracy (during adverse weather conditions like rain or interference from other Lidar systems in the vicinity) (6). Therefore, active research is ongoing to find alternative approaches that can directly process a regular camera RGB image to perform the task of 3D object detection and pose estimation. Cameras are low cost-effective but extracting depth information is a challenge. Also, they suffer during dark or low light conditions or weather conditions like fog. As of today, the lidar-based systems are much more accurate still. It is noteworthy that both approaches continue to be refined and are used in the industry e.g. Waymo (LiDAR) vs Tesla (Monocular) (6).

Self-driving cars have come a long way in recent years, but they're still not flawless. Consumers and lawmakers remain wary of adoption, in part because of doubts about vehicles' ability to accurately perceive objects in traffic.

In this paper, we have implemented a monocular based approach using CenterNet (1) which is an anchor free object detection algorithm in hopes to close this gap once and for all. Data analysed is from a Kaggle competition (2) (3) consisting of images taken from a car-mounted camera of city streets with traffic. The task is to predict the 6 DoF (yaw, pitch, roll, and the three-position coordinates) and report certain accuracy metrics. We use two different backbones for feature extraction. Finally, we perform a comparison of both the models and discuss the results.

2. LITERATURE REVIEW

Object detection is a fundamental visual recognition problem in computer vision and has been widely studied in the past decades. Deep learning techniques for image classification have been tremendously successful and in recent years are being actively used for object detection. Below are major milestones in object detection research based on deep convolutional neural networks since 2012. The trend in the last year has been designing object detectors based on anchor-free (in red) and AutoML (in green) techniques, which are potentially two important research directions in the future. (8)

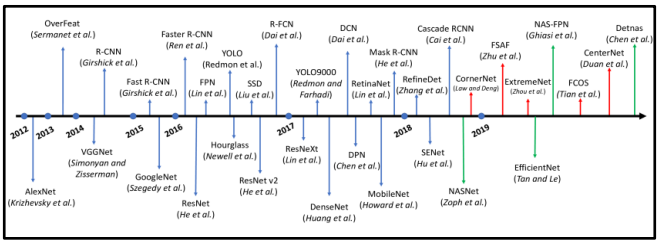

Figure 1: Various Object Detection Models Timeline

Existing methods for 3D object detection could generally be categorized into three streams: (i) geometry-based methods that

estimate the 3D bounding boxes with geometry and 3D world priors; (ii) learning-based methods that incorporate category-specific 3D shape prior or extra 2.5D information (depth, surface normal, and segmentation) to detect 3D bounding boxes or reconstruct the 3D object shape; and (iii) deep learning methods that directly estimates the 3D object bounding boxes from 2D bounding boxes. (PerspectiveNet: 3D Object Detection from a Single RGB Image via Perspective Points (7))

Mono3D: (9) The goal of this paper is to perform 3D object detection from a single monocular image in the domain of autonomous driving. Authors have proposed an approach to monocular 3D object detection, which generates a set of candidate class-specific object proposals that are then run through a standard CNN pipeline to obtain high-quality object detections. The aim of the author is to propose an energy minimization approach that places object candidates in 3D using the fact that objects should be on the ground plane, and then score each candidates box via several intuitive potentials encoding semantic segmentation, contextual information, size and location priors and typical object shape. They have adopted a CNN architecture proposed in another paper on object detection to score their proposals for object detection and orientation estimation. The accuracy of the model on moderate images of KITTI (10) dataset is 90.0% AP.

Deep3DBox: (11) In this paper, the author introduces a new model for 3D object detection and pose estimation for a single image. In contrast to other techniques that only regress the 3D orientation of an object, the method proposed in this paper first regresses relatively stable 3D object properties using a deep convolutional neural network and then combines those estimates with geometric constraints provided by a 2D object bounding box to produce a complete 3D bounding box. It is a two-stage detection model, the first network output estimates the 3D object orientation using a novel hybrid discrete-continuous loss, which significantly outperforms the L2 loss. The second output regresses the 3D object dimensions, which have relatively little variance compared to alternatives and can often be predicted for many object types. These estimates, combined with the geometric constraints on translation imposed by the 2D bounding box, can be used to recover a stable and accurate 3D object pose. They too have evaluated their method on the challenging KITTI (10) object detection benchmark and the model achieved an average precision (AP) of 89.04%.

CornerNet: (12) The author of this paper proposes a new approach for object detection where they detect the object bounding box as a pair of keypoints, the top-left corner and the bottom right corner, using a single convolutional neural network. In this method, the implication of detecting keypoints has eliminated the need for designing anchor boxes as commonly used in prior single-stage detectors. The architecture used in this paper is given below in **Figure 2**.

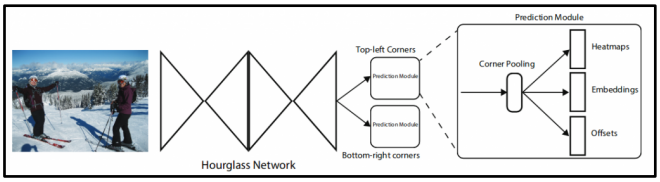

Figure 2: CornerNet Architecture

CornerNet (12) uses the hourglass network (13) (Newell et al., 2016) as its backbone network. An hourglass module first downsamples the input features by a series of convolution and max-pooling layers. It then upsamples the features back to the original resolution by a series of upsampling and convolution layer. Their hourglass network (13) consists of two hourglasses, and they have made some modifications to the architecture of the hourglass module. Instead of using max pooling, they have simply used stride 2 to reduce feature resolution. Authors of this paper also introduce the novel idea of corner pooling which helps in better localisation of corners. The convolutional network predicts two sets of heatmaps to represent the locations of corners of different object categories, one set for the top-left corners and the other for the bottom-right corners. The network also predicts an embedding vector for each detected corner (Newell et al., 2017) (14) such that the distance between the embeddings of two corners from the same object is small. To produce tighter bounding boxes, the network also predicts offsets to slightly adjust the locations of the corners. With the predicted heatmaps, embeddings and offsets, CornerNet (12) applies a simple post-processing algorithm to obtain the final bounding boxes. The proposed model achieved a 42.4% AP on MS COCO (15) dataset.

ExtremeNet: (16) In this paper the authors propose a bottom-up approach to detect objects by finding their extreme points. These points would then not only form a bounding box but also give a much tighter octagonal approximation of the object. The algorithm detects four extreme points(top-most, left-most, bottom-most, right-most) and one center point of objects using a standard keypoint estimation network. ExtremeNet (16) uses an hourglass network (13) to detect these five keypoints per class. To find these extreme points, the method predicts four multiple-peak heatmaps for each object category. In addition, they use one heatmap per category predicting the object center, as the average of two bounding box edges in both the x and y dimension. They then group the five key points into a bounding box if they are geometrically aligned. As stated in the paper, Object detection then becomes a purely appearance-based keypoint estimation problem, without region classification or implicit feature learning The proposed method has an accuracy of 43.7% AP on COCO (15) test-dev dataset.

CenterNet: (1) In the paper (Objects as Points), the author presents a new object representation technique where objects are represented as points. CenterNet (1) is a one stage detection algorithm and can be built on top of three different types of backbone networks namely, Hourglass (13) , Resnet (4) and DLA (17). Provided below in **Figure 3** is the basic architecture of this model.

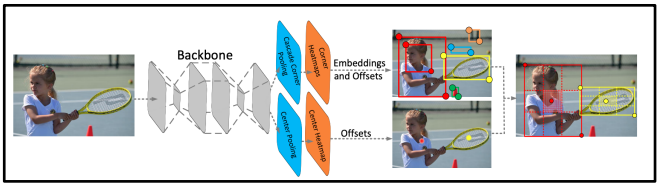

Figure 3: CenterNet Architecture

The model is built upon successful keypoint estimation networks. Using this, the author finds the object centers and regresses to their sizes. Since the proposed algorithm assigns the anchors based solely on location, we have one positive anchor per object and does not require any NMS (non max suppression) post-processing. CenterNet (1) is basically the combination of corner pooling and center pooling. The core logic of CenterNet (1) lies in extracting the local peaks in the keypoint heatmap. These heatmaps are then used with features from the corner heatmaps algorithms to estimate the pose of the object. The best accuracy of the CenterNet algorithm is with an hourglass network (13) as backbone with 42.2% AP in 7.8 FPS at a relatively good speed. The accuracy achieved using Hourglass backbone (13) outperforms the accuracy of

CornerNet (12) (40.6% AP in 4.1 FPS) and ExtremeNet (16) (40.3% AP in 3.1 FPS). The algorithm computes the fastest result with a ResNet-18 model with a performance of 28% AP on COCO (15) dataset at 142 FPS.

The CenterNet (1) algorithm can also be applied across broad application domains such as Pose estimation, depth and 3D orientation in one single stage.

3. IMPLEMENTATION

CenterNet as a preferred choice of algorithm for image detection and pose estimation

Image object detectors usually can be divided into two categories, one is a two-stage detector, the most representative one, Faster R-CNN. The other is a one-stage detector, such as YOLO, SSD. Two-stage detectors have high localization and object recognition accuracy, whereas the one-stage detectors achieve high inference speed. CenterNet (1), is end-to-end differentiable, simpler, faster, and more accurate than corresponding bounding box based detectors as evident below:

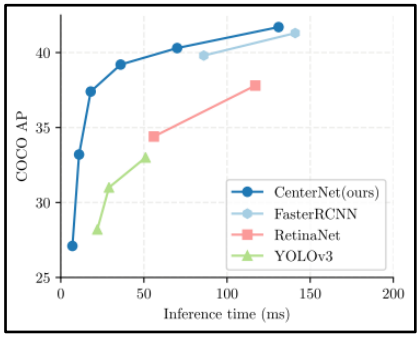

Figure 4: Speed-accuracy trade-off on COCO validation for detectors.

For 3D bounding box estimation, CenterNet is compared with slow-RCNN based Deep3DBox and Faster-RCNN based method Mono3D, on KITTI dataset. CenterNet performs on-par with its counterparts. However, CenterNet is two orders of magnitude faster than both methods.

	AP			
	Easy	Mode	Hard	Easy
Deep3DBox [38]	98.8	97.2	81.2	98.6
Ours	90.2±1.2	80.4±1.4	71.1±1.6	85.3±1.7
Mono3D [9]	95.8	90.0	80.6	93.7
Ours	97.1±0.3	87.9±0.1	79.3±0.1	93.4±0.7

Figure 5: KITTI Evaluation: Deep3DBox, Mono3D and CenterNet

In **Figure 5**, it is clearly established that CenterNet can accurately and quickly detect the image and estimate a range of additional object properties, such as pose, 3D orientation, depth and extent, in one single forward pass.

CenterNet (1) is implemented with three types of backbones i.e. ResNet (4), Hourglass (13) and DLA (17). It was found that the architecture with ResNet was the fastest among all the architectures and is less GPU intensive. The accuracy is also comparable to other architectures.

Backbones for CenterNet: We experimented with 2 architectures: ResNet-18 (4) and ResNeXt-50_32 (5) as backbones for feature extraction.. We modified both ResNet and ResNeXt using deformable convolution layers to add upsampling to the network as CenterNet requires fully convolutional neural networks.

ResNet: Xiao et al. (4) proposed ResNet. The basic building block of Resnet are conv and identity blocks. The core idea is to add an 'identity shortcut connection' which skips one or more layers. ResNet uses four modules made up of residual blocks, each of which uses several residual blocks with the same number of output channels. The number of channels in the first module is the same as the number of input channels. In the first residual block for each of the subsequent modules, the number of channels is doubled compared with that of the previous module, and the height and width are halved. Here together with the first convolutional layer and the final fully connected layer, there are 18 layers in total.

ResNeXt: Xie et al. (5) proposed a variant of ResNet that is codenamed ResNeXt. The authors introduced a hyper-parameter called cardinality - the number of independent paths, to provide a new way of adjusting the model capacity. ResNeXt is very similar to the Inception module of ResNet. They both follow the split-transform-merge paradigm, except in ResNeXt, the outputs of different paths are merged by adding them together, while in ResNet they are depth-concatenated. Another difference is that in ResNet, each path is different (1x1, 3x3 and 5x5 convolution) from each other, while in ResNeXt all paths share the same topology. Experiments show that accuracy can be gained more efficiently by increasing the cardinality than by going deeper or wider. Here we have taken cardinality of 32 and depth 4 hence it is ResNeXt-50.

CenterNet with backbone: The whole model is broken into three parts. The first part consists of 3 layers applied 5 times where each layer consists of a 3x3 convolution, batch normalization and ReLU. The 2nd part is the backbone architecture i.e. either ResNet-18 or ResNeXt-50. The 3rd part consists of an up-sampling layer and double convolution which consists of two 3x3 convolutional layers, batch normalization, and ReLU. The up-convolutional kernels are initialized as bilinear interpolation. At the output, a separate output layer which consists of a 1x1 convolution, batch normalization, and ReLU is used. The model with ResNet-18 as backbone is shown below in **Figure 6,** see **Appendix A1** for detailed architecture.

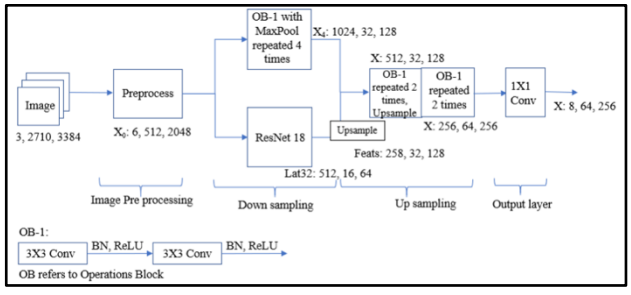

Figure 6: Block diagram of CenterNet with ResNet18 as backbone

4. METHODOLOGY

We evaluate our object detection performance on the Kaggle dataset [34], details for which are provided below. We report average precision overall IOU thresholds (AP), AP at IOU thresholds 0.5(AP_{50}) and 0.75 (AP_{75}). The supplement contains additional experiments on PascalVOC [14].

Dataset: We use Kaggle dataset (2) (3) provided by Baidu's Robotics and Autonomous Driving Lab (RAL), in association with Peking University. They have provided labelled 3D car instances from 6,277 real-world images, based on industry-

grade CAD car models. This dataset contains photos of streets, taken from the roof of a car, divided into a train (approx. 4200 images) and test (approx. 2000 images) sets. Label for train dataset is in the form of pose information provided in a .csv file consisting of 'ImageID' and 'PredictionString'. The pose string consists of 'model type', 'yaw', 'pitch', 'roll', 'x', 'y' and 'z' coordinates for a given car in the image. The 'x', 'y' and 'z' values indicate the center point of a car. The dataset also includes mask images for train and test sets. These binary masks are provided to allow competitors to remove some cars in the images (which are not of interest, as they are too far, etc.) during training. Around 50 json files provide coordinates to build a 3D model as a point cloud. However, the json file names do not match the "model type" of the prediction string and matching them is difficult.

Expected Output: The expected output is 6 DoF (Degree of Freedom) for all cars in the image. Metric used to compare the accuracy is mean average precision (mAP). Two measures which affect the mAP score are described below:-

Translational difference: This is the Euclidean difference between center points of predicted and ground truth labels (i.e. x, y, and z).

Rotational difference: This is the quaternion difference of values for the angles in radians (i.e. yaw, pitch and roll)

Using the above two differences, Precision and Recall for all the predicted data points are calculated. If both the differences are below the mentioned threshold then Precision and Recall calculated. The thresholds were provided in the competition details. Then the average precision and mean average precisions are calculated based on the precision and recall values.

Average Precision (AP): It is taking average values of precision across all the recall values.

$$AP = \frac{1}{11} \sum_{\text{Recall}_i} \text{Precision}(\text{Recall}_i)$$

To do this unambiguously, the AP score is defined as the mean precision at the set of 11 equally spaced recall values. (18)

Mean Average Precision (mAP): This score is calculated by taking the mean of AP over all classes and/or overall IoU (Intersection over Union) thresholds. The IoU is considered only in case of the bounding box prediction. For the 6DOF, IoU thresholds are not considered; only the translational and rotational differences are used for mAP calculation. The actual evaluation criteria mentioned by Peking University can be referred to at the attached link. (19)

Coordinate System Changes: The first priority was to understand how to map from the so-called "world" to "camera" to "image pixel" coordinates. This is done via matrix operations using the camera intrinsic properties matrix along with the rotation, and translation matrix. By using the 3D plotting vertices information provided, we computed an average bounding box dimension and plotted it on images.

Preprocess: All training and test images provided are of RGB 3384 x 2710 dimensions as shown in **Figure 9.** After our pre-processing steps, the input images are transformed into 512 x 2048 size as shown in **Figure 11**.

a. Some of the training images are corrupted and need to be ignored: overexposed images, not fully loaded images, etc.
b. Since this dataset contains only cars on ground level, we have chopped off the top half and used only half the image for training.

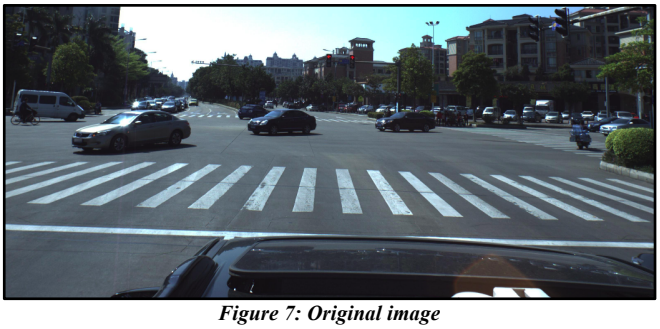

Figure 7: Original image

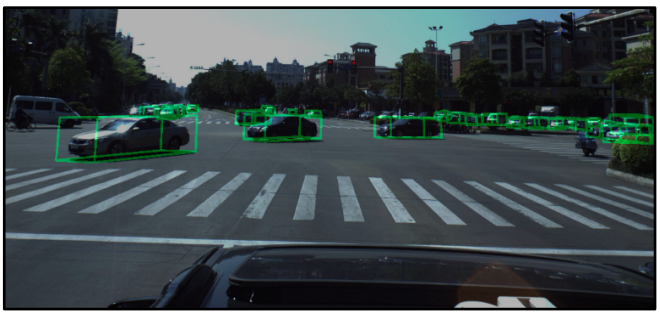

Figure 8: Image with 3D Bounding Box

c. The data for yaw and pitch are switched - which matters for visualising.
d. For around 2% of the ground truth prediction string data, the mapping to image pixel coordinates was to a point outside the image. Thus, we had to prevent training with this data. We resized the original image to 512 x 2048 and introduced a margin on either side to detect the out of bound data points. This allows the ground truth heatmap to be populated with correct centers and keep training data sanitized.
e. The heatmap dimensions used is 64 x 256 as shown in **Figure 11**.
f. While creating the heatmap, the ground truth center and its neighbouring positions are populated using a gaussian distribution. This allows us to test two different loss functions viz. binary cross entropy i.e. BCE vs Focal loss.
g. To get the values for the 6 DoF (x, y, z for position; and pose yaw, pitch, roll for pose), we are regressing directly at each of the ground truth image coordinate centers for the objects. Remember that the position specifies the center of a car in the camera coordinate system.
   - As all the cars are on the surface level, the yaw values (*pitch in our data provided*) can span the entire 360 degree range. However, the other two angles have only a small variation around a value of 0 degrees.
   - Since the cars are of somewhat similar dimensions, the variability of the y value is small. However, the possible variability for x will rise as the car is further away from the camera. This effect will be much more pronounced for the z values as that represents the distance from the camera.
   - With these factors in mind, the original position and pose value are transformed in a way to ensure more granularity and thus improve model accuracy. The model inference values are then transformed back to the camera coordinate perspective.
   - The **Table 1** below shows the transformations:

SN	Feature	
	Original	Transformed
1	x	x / 100
2	y	y / 100
3	z	z / 100
4	yaw	unchanged
5	pitch	sin(pitch)
		cos(pitch)
6	roll	roll + Pi

Table 1: Transformation of Model Output to Required Values

Figure 9: Original Image

Driver Skill determination: Out of all the models that we trained, we selected the model with the highest accuracy. We used it to predict the 6 DoF of the cars it detected. We then develop an approach to predict if the driving control can be passed to a human in actual traffic conditions.

Imagine a time in the future when autonomous driving is socially acceptable and popular; due to this it is very rare for a human to request control from the system. However, there is an option for the person to "request control". The AI system immediately analyses the number of cars present and based on certain risk factors hands over control. This action depends on a present user profile which is set as "Beginner", "Medium" and "Advanced". In certain cases, the AI may not hand over control even to an advanced user as it determines it is unsafe and this would be the "Not Allowed" condition.

The flowchart below in **Figure 10** shows the logic to determine risk factors and decide what level of driving skills is required to take manual control from the AI:

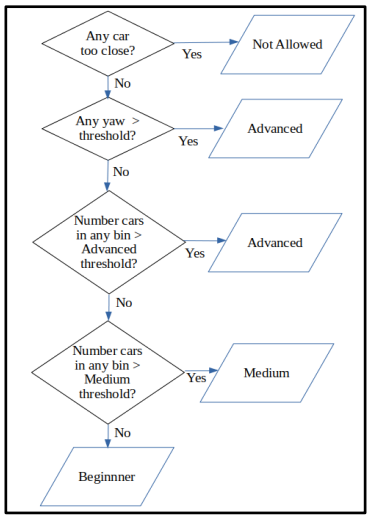

Figure 10: Manual Control Logic

The threshold values selected in our use case are:

- Car too close threshold = 20 units
- Yaw threshold = 20 degrees
- Bin sizes = 20 units
- Maximum limit for cars / bin for MEDIUM = 5
- Maximum limit for cars / bin for ADVANCED = 8

With the predicted values from the best model, we implemented this use case scenario.

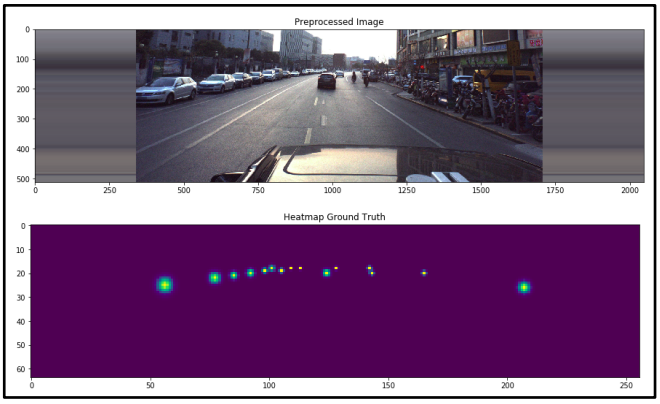

Figure 11: Preprocessed Image along with Heatmap

5. EVALUATION

Ablation Study for different backbones: We evaluated different models with hyperparameter tuning for each model. The table below summarises the mAP and processing run times for various models that were tested. In each case, we ran inference on 10% of the training data which became our dev set of 426 images. The runs were on Google Colab with 16GB of RAM and the default allocation of GPU memory. The ignore mask images (where available) were used.

SN	Backbone	Epochs Trained	Loss Function	Mask Images Used	Time (seconds)		mAP
					Total	Average Per Image	
1	Resnet-18	10	Focal	Yes	1224	2.87	0.10616
2	Resnet-18	6	Focal	Yes	786	2.45	0.07877
3	Resnet-18	4	Focal	Yes	866	2.03	0.07456
4	Resnet-18	4	BCE	Yes	1264	2.96	0.04754
5	Resnext-50_32	4	Focal	Yes	941	2.21	0.08551
6	Resnext-50_32	4	BCE	Yes	965	2.26	0.06229

Table 2: CenterNet models – Time and Precision metrics

The following observations can be made from the **Table 2**:

a. For the same number of epochs, using Focal loss instead of BCE improves mAP by about 0.02.
b. As expected, the mAP rises with more training.
c. Keeping in mind that each run was done independently on the cloud, we could not ensure the same resources would be provided by Google Colab for each run instance:
   - The Total time shows some variability which might be random. However, the average inference time per image is quite consistent within a span of 2-3 seconds.
   - As ResNeXt-50_32 has a larger depth and width, one would expect the inference time to be slightly higher compared to that of Resnet-18. But, for only one out of the 4 Resnet-18 models tested, the average time is lower. A controlled environment which allocates exactly the same resources during each test run, may demonstrate ResNeXt-50_32 is slower than ResNet-18.
d. The best model we could produce was the first entry in the table: ResNet-18 backbone, for 10 epochs and with Focal loss function.

Driving skill analysis using the best model for inference

The best model is provided with images it has not been trained on. Based on the driver skill determination logic, shown below is an example for each possible output.

Case 1: NOT ALLOWED due to proximity check fail. At least one car is within the threshold defined as shown in **Figure 12**.

Case 2: MEDIUM as the number of cars is above the threshold in some distance bin as shown in **Figure 13**.

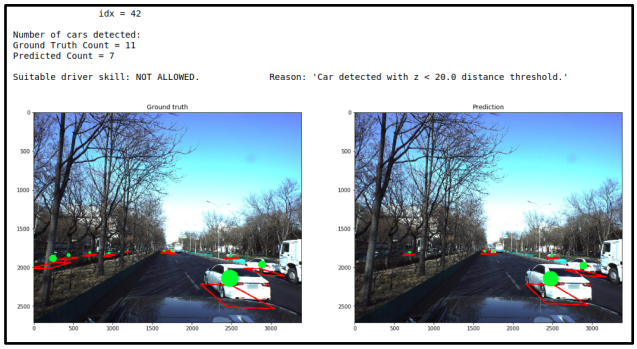

Figure 12: Case 1: NOT ALLOWED

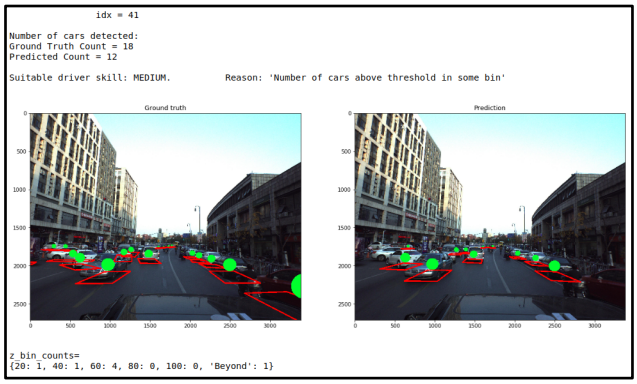

Figure 13: Case 2: MEDIUM

Case 3: ADVANCED because Yaw of some car is above the threshold as shown in **Figure 15**.

Case 4: BEGINNER as there are no cars in close proximity, no car with large Yaw and no high density in each distance bin as shown in **Figure 16**.

Thus examples for each of the four cases are covered.

Best model Training Statistics: We will now evaluate the variation of training losses, development losses and mAP with epoch for the best model i.e. CenterNet with Resnet-18 backbone, for 10 epochs and with Focal loss function.

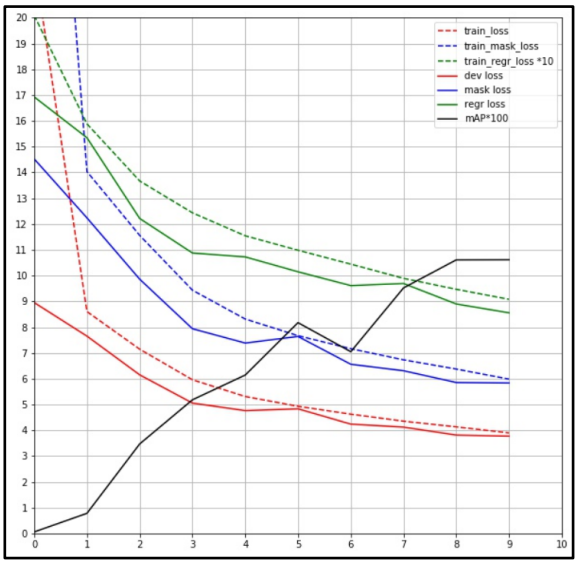

Figure 14: Training statistics against Epoch

The following observations can be made from the graph:

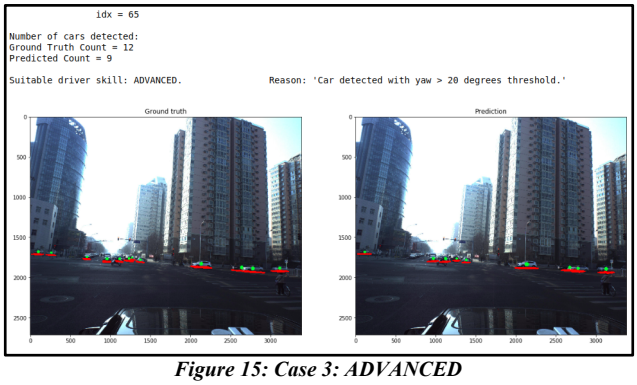

Figure 15: Case 3: ADVANCED

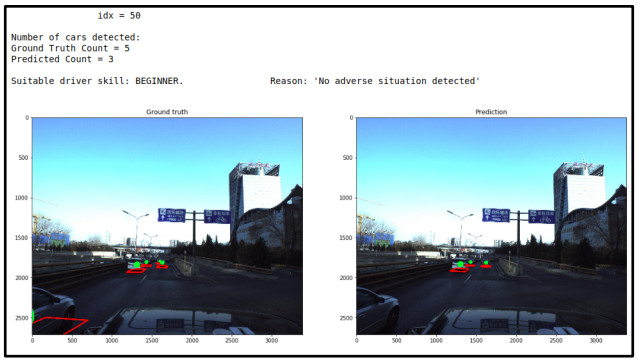

Figure 16: Case 4: BEGINNER

- The training loss and development losses fall steeply during the initial 3 epochs
- After 3 epochs, the training and development losses fall gradually
- mAP rises with each epoch. There is a fall in mAP from 5th epoch to 6th epoch.
- Overall, the trend of losses and the mAP with each epoch is as expected.

Please find Epochwise Error Rate for Model with ResNet-18 backbone in **Appendix A2**.

6. CONCLUSION & FUTURE SCOPE

Conclusion:

Based on our experimenting with different models and their training hyperparameters, we conclude the following:

a. The ResNeXt-50_32 model outperforms a Resnet-18 in terms of accuracy.

b. If we were able to train for say 50-100 epochs, the mAP most probably would be much higher than that achieved with 10 epochs. As per the CenterNet - Objects as Points paper, their best model hit an accuracy of around 45% AP. It is however hard to say if we would have reached the mid-forties level for our dataset.

c. We started off with very simple models. These used a simple BCE loss function, no use of training mask images, no pre-checks to ensure only sanitized training data and no transformation on the variables to be predicted (pose information). The mAP did not even cross very single digits.

d. Only after incorporating training data sanity checks, using a better loss function (Focal loss) and figuring out how to use the mask images, were we able to improve the accuracy to around 10%.

Future Scope:
Possible enhancements that could be done in our approach are as follows:

a. Data augmentation like h-flip, 3 axis rotate, colour, noise, blur, etc. could be incorporated in the dataset
b. A stacked Hourglass backbone is also mentioned in the CenterNet (1) paper as having high accuracy but with higher inference time. Even though low inference time is important for an autonomous driving use case, it would be interesting to see what accuracy levels are achieved on our dataset.
c. This experiment is done on the ApolloCar3D (3) dataset. Since the KITTI (10) dataset is also popular for autonomous driving use cases, if one can get the ground truth values for the pose information, the model accuracy could be improved via transfer learning.
d. A further study could look at how each of the 6 DoF variables contributes towards the overall error.

7. SOURCE CODE

Please find all the relevant code at the below mentioned GitHub Link:

https://github.com/rbewoor/Pose_Estimation_Autonomous_Driving

8. REFERENCES

1. Zhou, Xingyi, Wang, Dequan and Philipp Krahenbuhl. *Objects as Points.* Texas, USA : https://arxiv.org/pdf/1904.07850.pdf, 2019.

2. Kaggle. [Online] 2019. https://www.kaggle.com/c/pku-autonomous-driving/data.

3. Song, Xibin, et al. *Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving.* China : https://arxiv.org/abs/1811.12222, 2018.

4. He, Kaiming, et al. *Deep Residual Learning for Image Recognition.* USA : https://arxiv.org/pdf/1512.03385.pdf, 2015.

5. Xie, Saining, et al. *Aggregated Residual Transformations for Deep Neural Networks.* San Diego, USA : https://arxiv.org/pdf/1611.05431.pdf, 2017.

6. Grigorescu, Sorin, et al. *A Survey of Deep Learning Techniques for Autonomous Driving.* Brasov, Romania : https://arxiv.org/pdf/1910.07738, 2019.

7. Huang, Siyuan, et al. *PerspectiveNet: 3D Object Detection from a Single RGB Image via Perspective Points.* Vancouver, Canada : https://arxiv.org/pdf/1912.07744.pdf, 2019.

8. Wu, Xiongwei, Sahoo, Doyen and Hoi, Steven C.H. *Recent Advances in Deep Learning for Object Detection.* Singapore : https://www.groundai.com/project/recent-advances-in-deep-learning-for-object-detection/1, 2019.

9. Chen, Xiaozhi, et al. *Monocular 3D Object Detection for Autonomous Driving.* Toronto, Canada : https://www.cs.toronto.edu/~urtasun/publications/chen_etal_cvpr16.pdf, 2017.

10. Technology, Karlsruhe Institute Of. [Online] http://www.cvlibs.net/datasets/kitti/.

11. Mousavian, Arsalan, et al. *3D Bounding Box Estimation Using Deep Learning and Geometry.* Virginia, USA : https://arxiv.org/pdf/1612.00496.pdf, 2017.

12. Law, Hei and Deng, Jia. *CornerNet: Detecting Objects as Paired Keypoints.* New Jersey, USA : https://arxiv.org/pdf/1808.01244.pdf, 2019.

13. Newell, Alejandro, Yang, Kaiyu and Deng, Jia. *Stacked Hourglass Networks for Human Pose Estimation.* Michigan, USA : https://arxiv.org/pdf/1603.06937.pdf, 2016.

14. Newell, Alejandro, Huang, Zhiao and Deng, Jia. *Associative Embedding: End-to-End Learning for Joint Detection and Grouping.* Michigan, USA : https://arxiv.org/pdf/1611.05424.pdf, 2017.

15. Microsoft. [Online] http://cocodataset.org/#home.

16. Zhou, Xingyi, Zhuo, Jiacheng and Krahenbuhl, Philipp. *Bottom-up Object Detection by Grouping Extreme and Center Points.* Texas, USA : http://openaccess.thecvf.com/content_CVPR_2019/papersZhou_Bottom-Up_Object_Detection_by_Grouping_Extreme_and_Center_Points_CVPR_2019_paper.pdf, 2018.

17. Yu, Fisher, et al. *Deep Layer Aggregation.* UC Berkeley, USA : https://arxiv.org/pdf/1707.06484.pdf, 2019.

18. Arlen, Timothy C. medium.com. [Online] Mar 1, 2018. https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3.

19. University, Peking. Kaggle.com. [Online] https://www.kaggle.com/c/pku-autonomous-driving/overview/evaluation.

APPENDIX A1

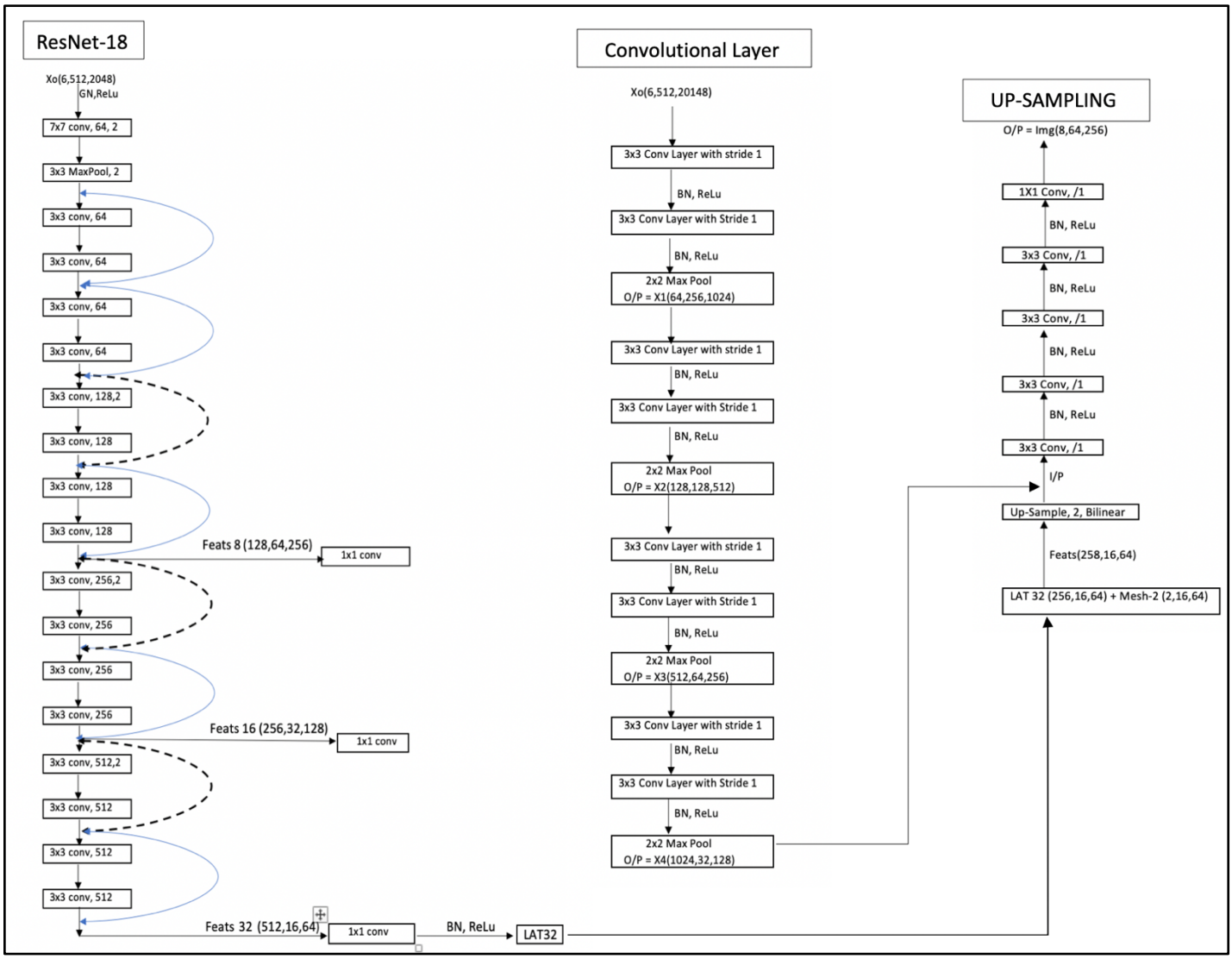

Figure 17: Detailed Architecture of CenterNet with ResNet-18 Backbone

APPENDIX A2

Epoch	TRAIN Loss (Total = 50% Mask + 100% Regr)			DEV Loss (Total = 50% Mask + 100% Regr)		
	Mask loss	Regr Loss	Total Loss	Mask loss	Regr Loss	Total Loss
0	27.2	1.98	15.58	15.94	1.65	9.62
1	12.87	1.48	7.92	10.61	1.4	6.7
2	9.47	1.27	6.01	7.93	1.13	5.1
3	7.86	1.14	5.07	7.28	1.02	4.66
4	6.84	1.05	4.47	6.74	0.99	4.36
5	6.81	0.99	4.4	6.57	0.97	4.26
6	6.01	0.94	3.95	6.19	0.88	3.98
7	5.74	0.89	3.76	6.17	0.9	3.98
8	5.09	0.83	3.38	5.74	0.83	3.7
9	4.82	0.81	3.22	5.83	0.8	3.72

Figure 18: Epochwise Error Rate for Model with ResNet-18 backbone