UTILIZING SUPER-RESOLUTION FOR ENHANCED AUTOMOTIVE RADAR OBJECT DETECTION: SUPPLIMENTARY MATERIAL

Asish kumar Mishra[⋆] *Kanishka Tyagi*[†] *Deepak Mishra*[⋆]

[⋆] Department of Avionics, Indian Institute of Space Science and Technology
Thiruvananthapuram, Kerala, India
[†]Aptiv Advance Research Center
Agoura Hills, California, USA

1. INTRODUCTION

This supplementary material presents the various experiments and the results obtained while working on the idea presented in the paper. Various unsuccessful and minor experiments that we conducted have been presented along with the obtained results.

2. RESOURCES

The codes used in this work have been submitted along with this supplementary pdf on GitHub `https://github.com/kanishkaisreal/DLSR_CRUW`. As per the three datasets, we could not find a way to host them anonymously without violating the rules cited by WACV. However, the examples shown in the paper are direct samples from the dataset, thus can help the readers generalize the dataset.

3. UNSUCCESSFUL IDEAS AND RESULTS FOR GENERATING SR DATA

Data Generation is a crucial aspect of the Radar Super Resolution scenario. To the best of our awareness, no single dataset contains pairs of Radar images with low and high resolution. The lack of low-high Radar picture pairs prevents the use of Deep Learning Methods for Super-resolution. Consequently, it was necessary to construct a Radar data pair from a low-resolution or high-resolution Radar image. The creation of low-resolution Radar images cannot be accomplished by simple downsampling since the definition of resolution in this context differs from that of photographs. In this section, we have examined every attempted approach to create a pair of low-high quality images. Due to several factors, none of the approaches were able to create an impactful pair of low-high resolution Radar images.

3.1. Image SR

Since Image Super-resolution is a well-researched field in the deep learning domain and data are available, the first experiment was undertaken for domain adaptation on Image Super-resolution architectures. Pre-trained models which performed the best to date, such RDNs, GANs, et cetera, were applied to synthesize high-resolution Radar images from low-resolution Radar images. All the models are trained first using low-high resolution image pair, and then the pre-trained model is employed for the Radar Super-resolution data creation. The implemented networks are Residual Dense Networks for Image Resolution [**?**], Enhanced Super Resolution GAN [**?**] and SRGANS [**?**] (code was readily available from internet).

While the idea of Domain Adaptation from Image Super-resolution may sound extremely persuasive in tackling the problem of Radar Super-resolution, the models could not enhance the resolution at all. All the networks trained for Image Super-resolution enhanced the pixel resolution and sharpened the border a little more but could not raise the smearing or the spread caused by the poor resolution of Radar. Figure. 1 contrasts the input and output Radar image from the RDN model used for Image Super-Resolution based Domain Adaptation. Such a failure to enhance Radar resolution explains the disparity in the definition of resolution for Image and Radar. As per the definition of Radar Super-resolution, it relates to lowering the spread and smearing of the signal, which is not the case with Image Super-resolution.

Though the RDN model results are provided, the findings remained the same for other ESRGAN and SRGANS. Therefore, the domain adaptation approach is not a way to obtain Radar Super-resolution by domain adaptation.

3.2. Traditional filters

To obtain Radar Super-resolution, various static filters, including averaging, sharpening, gaussian-denoising, and chambolle total variation denoising filters. The averaging, sharpening and gaussian-denoising were applied through the kernel, while the chambolle total variation denoising filter was carried out using various partial differential equations. Equation 3.2 represents several kernels used for averaging, sharpening, and gaussian denoising. The sharpening here is defined as subtracting the average from the original image. It is to be

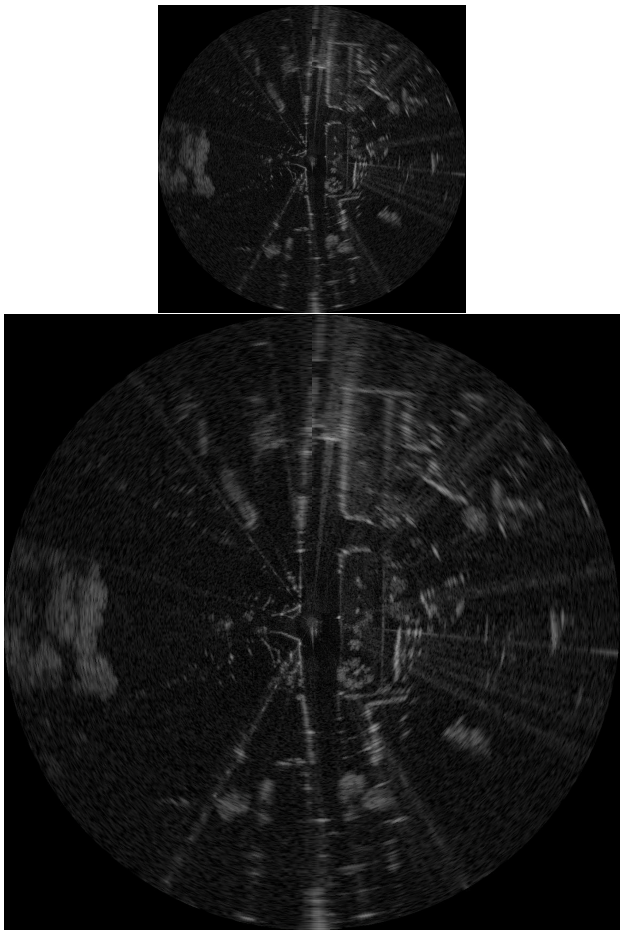

Fig. 1. Left image indicates the input (1152×1152 pixel) image into a pre-trained RDN model used in Image Super Resolution. The right image indicates the output from that same RDN network(2306×2306 pixel). Both the Radar maps are in cartesian coordinates

observed that the kernels are chosen such that the sum of the matrix elements always totals 1.

$$K_{\mathrm{G}} = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}, K_{\mathrm{Avg}} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, K_{\mathrm{Sharp}} = \frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 17 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

where K_{G} denotes the Gaussian-denoising kernel, K_{Avg} denotes the averaging kernel and K_{Sharp} denotes the sharpening kernel.

The Gaussian-denoising, averaging, and sharpening were able to generate some modifications in the input image but could not de-noise the Radar image(in our instance, making the Radar images clutter-free). Figure. 2 displays the different filters on a radar low-resolution image. The fundamental goal behind developing these filters was to minimize the noise assumed to be Gaussian noise. Though it could minimize the noise to some level, it did not increase the resolution of the Radar images. This is because the Radar noise cannot be eliminated by simple filtering as the clutter is not constrained to a frequency range different from the object in the Radar map.

The total-variation filter was unable to make much difference to the Radar resolution. Figure. 3 illustrates the results when the original low-resolution Radar image is put through a Chambolle Total-variation denoising filter. The total-variation filter typically boosts the edges and removes all other noise, as it did on the low-resolution Radar image. However, it could not remove the clutter in the Radar images. Some background noise also was eliminated, but the filter was not strong enough to decrease the clutter.

Therefore, the static filters could not increase the Radar resolution, but they could eliminate some Gaussian noise and enhance the edges in the Radar image.

3.3. Deconvolution

The deconvolution-based approach may also be utilized to enhance the Radar image resolution. As mentioned in theory, the deconvolution method seeks to conduct a de-convolve a noisy image and generate a less smeared image based on equation **??**. Here, random ground truth may be generated, and a simulation environment can be utilized to carry out the antenna and noisy signal simulations. Once the H matrix from equation **??** is known for an antenna, a Gaussian noise may be added to the matrix multiplication to form a noisy signal, and the SFSBA [**?**] method; can be denoised. Using the SFSBA technique, the Radar resolution may be enhanced, and low-high pair can be recovered, which may be used for further investigation.

However, one needs to know the convolution matrix by the antenna field pattern created by the object's interaction with the environment to use the deconvolution algorithm. Both the object and the environment can vary from scenario to scenario. No standard deconvolution algorithm can be used to reduce the smearing in the azimuth axis. Nevertheless, a standard deconvolution matrix with smoothing (to reduce the effects of noise) helps to reduce smearing and achieve near-super-resolution. The deconvolution matrix can be calculated from the number of antennas present in the transmitter and receiver array if the antennas' angular beam-width, directivity, and gain are known. These variables define the antenna and are kept exclusively by the authors to avoid duplication. Because these details are unique, we could not know the exact deconvolution matrix needed to decode the environment and improve super-resolution. Therefore, we cannot use this method to generate pairs of high and low-resolution radar images.

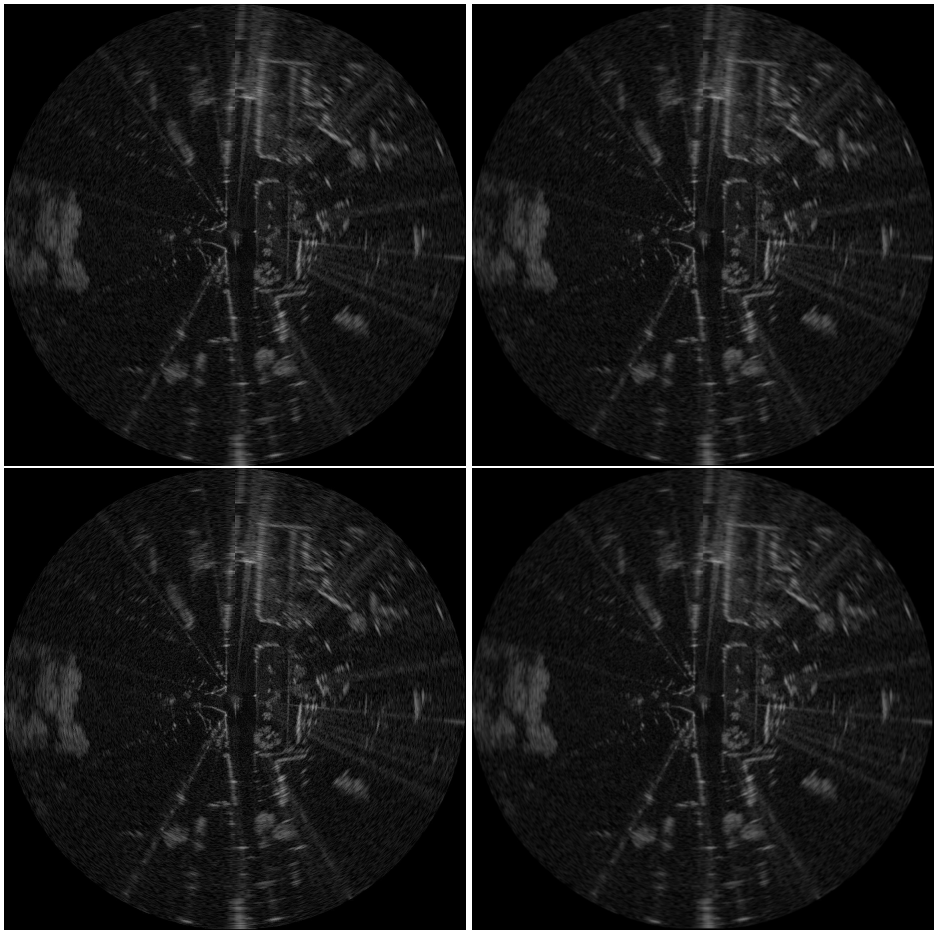

Fig. 2. Various filters that we applied to the low-resolution Radar image. (Clockwise from top left) Original Image, Image passed through Average filter, Image passed through Gaussian denoising filter, and Image sharpened through sharpening filter. The filters were not able to improve the resolution of the Radar images. All the Radar maps are in cartesian coordinates

4. SIMULATION

All of the above methods failed to generate the Radar Low - High image pair dataset needed to train a deep learning model that can be used for super-resolution. Therefore, the most viable option for generating radar data is to use Simulation. The authors trained a super-resolution model using Simulation in the SFSBA [**?**] paper. This model was later tested with real data for validation. The simulation parameters are unknown, but some antenna patterns are known, as described in the paper. The Table 1 shows the antenna parameters listed in the publication. The parameters given in the table have been considered while carrying out the Simulation.

Table 1. Antenna Parameters as mentioned in [**?**]. The authors have considered these antenna characteristics while carrying out the Simulation for generating Radar images.

Parameter	**Value**	**Units**
Bandwidth	45	MHz
Antenna scanning azimuth	$[-5, 5]$	$^\circ$
Antenna scanning velocity	50	$^\circ/\mathrm{s}$
Antenna scanning range	100	m
Beam Width	3.5	$^\circ$
Pulse Repetition Frequency	1000	Hz

4.1. Different options explored in Simulation

Various noises like Gaussian, Speckle, Salt and Pepper, and Poisson have been experimented with. However, only Gaussian and Speckle were taken as they disturbed the image significantly more than the other noises. Other experiments include simulating various object shapes to accommodate real-life scenarios.

4.1.1. Gaussian Noise

Gaussian noise with high mean and variance was added to the images to incorporate practicality. Even though the simulated

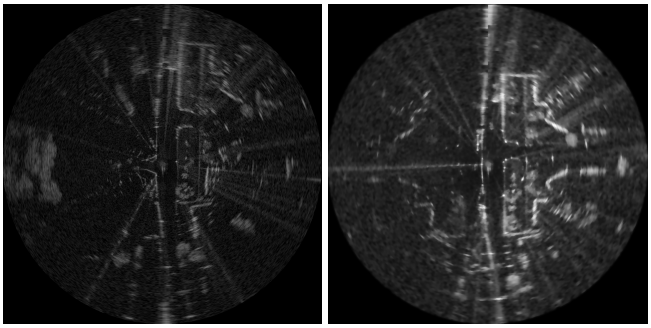

Fig. 3. Chambolle Total-Variation Denoising filter applied to a low-resolution image. Left - Original Image, Right - Total variation denoised Image. Only the colors and edges were enhanced, and some noise was eliminated. Both the Radar maps are in cartesian coordinates

noise is not as noisy as the noise seen in the actual life image, efforts were made to make it as close as possible. The mean was taken as 130, and the gaussian was taken as 70 while adding the noise. Such a high mean and variance value is taken to disturb the images significantly. Figure. 4 shows the simulated result with the Gaussian Noise.

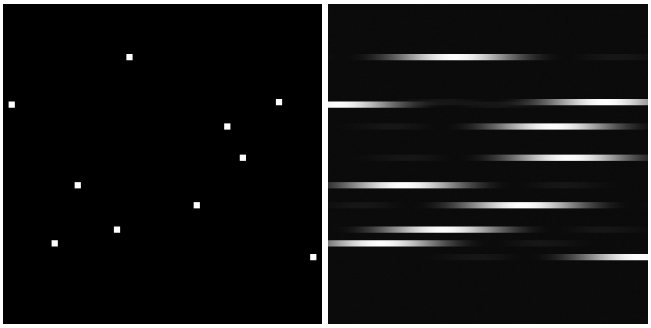

Fig. 4. Polar Radar images with Gaussian noise generated from Simulation. Left - Ground Truth/High Resolution Radar Image, Right - Low resolution Radar image with Gaussian noise ; Top pair - Sparse Environment, Bottom pair - Dense Environment (Azimuth limit : $[-5°, 5°]$, Range limit - [0,100] metres)

4.1.2. Speckle Noise

The other type of noise taken was Speckle Noise. In real-life scenarios, speckle noise is assumed to be noise in the background received due to the back-scattering of emitted rays. This noise stays dominant in Radar; therefore, it is a viable option to add speckle noise into the dataset. Figure. 5 shows the simulated result with the Speckle Noise.

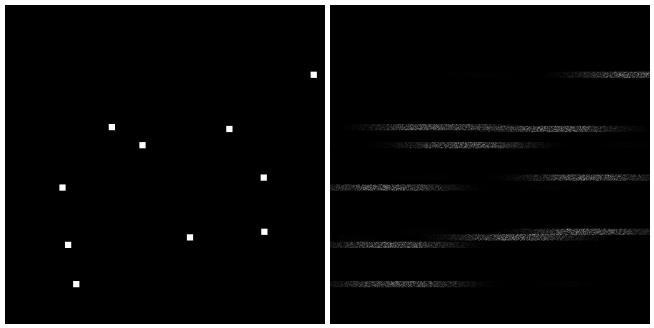

Fig. 5. Polar Radar images with Speckle noise generated from Simulation. Left - Ground Truth/High Resolution Radar Image, Right - Low resolution Radar image with Speckle noise ; Top pair - Sparse Environment, Bottom pair - Dense Environment (Azimuth limit : $[-5°, 5°]$, Range limit - [0,100] metres)

4.1.3. Gaussian and Speckle Noise

Gaussian and Speckle noise was combined to make the dataset harder and closer to the actual data. The figure seems to be affected by these noises, but the model used for Super-resolution did not respond well for this kind of image as it did for other variations. Figure. 6 shows the simulated result with both Gaussian and Speckle Noise.

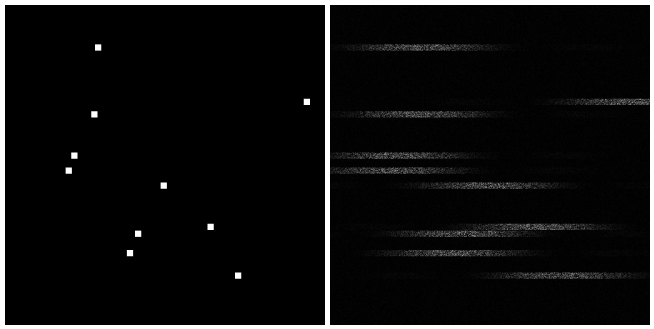

Fig. 6. Polar Radar images with both Gaussian and Speckle noise generated from Simulation. Left - Ground Truth/High-Resolution Radar Image, Right - Low-resolution Radar image with both Gaussian and Speckle Noise; Top pair - Sparse Environment, Bottom pair - Dense Environment (Azimuth limit: $[-5°, 5°]$, Range limit - [0,100] meters)

4.1.4. Various object shapes

In all the earlier Simulations, the object shape was a fixed square with 20×20 pixels. Though the Radar does not distinguish much between object shapes due to smearing and distorted edges, it is better to generate different objects to tackle extreme cases like comparing buses and pedestrians. Hence, five object shapes with different aspect ratios were chosen as

models for object creation. Figure. 7 shows the simulated result with various object shapes.

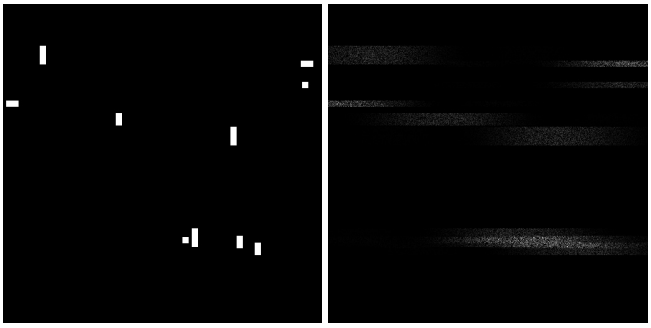

Fig. 7. Polar Radar images with various object shapes generated from Simulation. Left - Ground Truth/High Resolution Radar Image, Right - Low resolution Radar image ; Top pair - Sparse Environment, Bottom pair - Dense Environment (Azimuth limit : $[-5°, 5°]$, Range limit - [0,100] metres)

5. VISION-BASED OBJECT DETECTORS FOR RADAR

Different architectures of vision-based object detectors like EfficientDet and SSD have been experimented with for detecting objects in Radar maps. As explained in the paper, the vision-based object detectors fail to recognize the objects on a Radar map because of other noises and clutters present on the Radar map. Therefore, we cannot expect the detectors to classify objects in the Radar map if we pass a low-resolution CRUW dataset. The results from our experiment show abysmal accuracy. However, the masked low-resolution image generated with the help of Super-resolution stands as a better candidate for these object detectors as they filter out the unnecessary noises and clutters from the image. The vision-based object detectors trained on these masked low-resolution images show better accuracy than the original low-resolution images.

5.1. EfficientDet Implementation and Results

The "EficientDetLiteV0" model was used for training in two cases to show the usefulness of Super-Resolution. In the first case, the object detection model was trained and tested on simple low-resolution Radar images without any changes or data augmentation. In the second case, the low-resolution model was multiplied with the mask generated from the high-resolution image, as explained above. Tensorflow Lite framework was used for the whole process, and a 32 GB, NVIDIA V100 card was used for training and testing. An acceptable train-test-validation ratio of $70 : 20 : 10$ was used for both cases. Furthermore, the model trained only on low-resolution was tested on the masked low-resolution images for observation. The implementation details for the object detection can be found in the list below:

- Framework Used: Tensorflow Lite
- GPU: NVIDIA V100
- Data : CRUW - 36,765 Images (Train:Val:Test = 70:20:10)
- Batch size: 64
- Number of Epochs: 20
- Average training time: 10 GPU-minutes per epoch
- Evaluation metrics: Average Precision, Average Recall
- Loss functions: Focal loss
- Regularization Used: L2

This experiment has been carried out with EficientDetLiteV0, a very light model. For training the model, a high number of epochs are required. Taking the time and computation constraints into account, the number of epochs has been restricted to 20, and the results have been compared.

Although the training is not fully completed, the number of epochs is kept at 20 for both cases to obtain a fair comparison. Table. 2 compares the various evaluation metrics value for Object detection for Low Resolution and Object detection for Low Resolution aided by Super-Resolution through Masking. The evaluation metrics have taken mean Average Precision (AP) and mean Average Recall(AR). The various mean APs like AP50 and AP75 are based on various threshold values for the Intersection over Union (IoU) parameter. By default, the threshold for IoU is assumed as 0.9. The AP50 and AP75 consider the threshold for IoU as 0.5 and 0.75, respectively. The other APs like AP_/car, AP_/cyclist, and AP_/pedestrian are Average Precision values only for a specific object class like a car, cyclist, or pedestrian. The Average Recall evaluation metric (ARs) is the mean Average Recall. In contrast, the other types of ARs like ARmax1, ARmax10, et cetera consider the number of detection per class per image. ARmax1 metric considers only one object detection per class per image, whereas ARmax10 takes a maximum of 10 detections per class per image. For example, suppose fifteen pedestrians are in a Radar image. In that case, ARmax1 gives a full reward if one pedestrian is predicted, whereas ARmax10 gives total rewards only if ten or more pedestrians are predicted correctly.

After observing the findings shown in Table. 2, we can conclude that all evaluation metrics rise when Super-resolution is used for Object Detection. Each experiment was conducted on the same model with the same number

Table 2. Result Comparison between Object detection for Low Resolution and Object detection for Low Resolution aided by Super Resolution through Masking. AP - Average Precision, AR - Average Recall

Radar LR	Masked Radar LR
AP: 0.06974416	AP: 0.08460694
AP50: 0.17842645	AP50: 0.22804701
AP75: 0.018305851	AP75: 0.028328348
AP_/car: 0.076472606,	AP_/car: 0.096251525
AP_/cyclist: 0.003204726	AP_/cyclist: 0.01806273
AP_/pedestrian: 0.08067515	AP_/pedestrian: 0.10950657
ARmax1: 0.090764165	ARmax1: 0.095020615
ARmax10: 0.20705192	ARmax10: 0.24551837
ARmax100: 0.28472316	ARmax100: 0.3320174
ARs: 0.28472316	ARs: 0.3320174

of epochs and quantity of data. However, the Average accuracy and Average recall with SR-assisted Object Detection are more accurate. This indicates that utilizing Super-Resolution for object identification is superior to using a low-resolution model alone. Although this model is not adequately trained, we anticipate that more training will improve Super-resolution assisted Object Detection outcomes. Thus, we may now improve Radar object identification without needing a camera, Lidar, or any other domain, such as Doppler or velocity, as was previously the case.

5.2. SSD Implementation and Results

Similar to EfficientDet, SSD was trained and tested on two types of datasets: One involving low-resolution radar maps and the other involving masked low-resolution radar maps. The network composes of a VGG network as the feature extractor followed by multibox convolution layers as the object classifier. Each multibox layer predicts class confidence scores, localization predictions, and associated prior-box layers for producing default boundary boxes. The model has been implemented with the help of the code presented in `https://github.com/midasklr/SSD.Pytorch`.

The implementation details for the object detection can be found in the list below:

- Framework Used: Pytorch
- GPU: NVIDIA V100
- Data : CRUW - 36,765 Images (Train:Val:Test = 70:20:10)
- Batch size: 64
- Number of Epochs: 250
- Evaluation metrics: Accuracy
- Average training time: 5 GPU minutes per epoch
- Loss functions: Smooth L1 loss, Cross-entropy loss
- Hard-negative mining used: Yes

Using this model, we could barely achieve some results on object detection. While the accuracy on the low-resolution CRUW Radar maps was around 8%, the accuracy on masked low-resolution maps was close to 22% while implementing object detection. On analyzing the model's prediction, the probability scores were very low and rarely exceeded 0.2 for the low-resolution dataset, whereas it sometimes exceeded 0.25 for the masked low-resolution dataset. Figure. 8 shows the prediction from the SSD model.

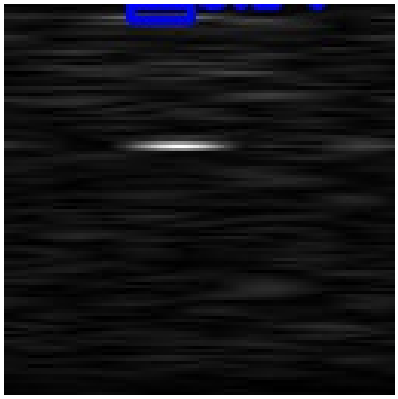

Fig. 8. Result predicted by SSD on the low-resolution CRUW dataset. We can observe from the image that the predictions from SSD are poor to comment on its performance.

From the results presented, we can observe that the accuracy has improved, but the accuracy score is too poor to claim the superiority of super-resolution in object detection. Though each experiment was repeated three times, the accuracy scores stayed almost the same for all the cases.

6. OBJECT CLASSIFICATION

Because of the high class imbalance between the background (none) class and the rest three classes: car, cyclist, and pedestrian, some other experiments were conducted in Object Classification. The car, cyclist, and pedestrian classes comprised around 45000 images per class, whereas the none class comprised roughly 1,80,000 images. To reduce this class imbalance, the three classes: car, cyclist, and pedestrian, were merged into one class - the object class.

For this experiment, a classifier of roughly the same structure as the network mentioned in the paper was trained on two classes. The classifier model was trained on a train:test:val split of $70 : 20 : 10$ with the exact implementation details as presented in the paper, and it achieved around 75% accuracy in classifying into the two classes. In order to classify various object classes, another follow-up classifier classifying into three classes: car, cyclist, and pedestrian, was used.

This follow-up classifier achieved around 80% accuracy when classifying. Since the total accuracy from this system of two classifiers accounted for 60%, it was not selected as a reliable model for object classification.