A. Appendix

A.1. Implementation of Dissimilarity Network

The dissimilarity module is derived from six unique components: two encoding architectures, one fusion module, and three decoder blocks. These components are reused across the network to build the final architecture. Figure 3 shows a high-level view of all the components interconnected.

We follow the naming convention used in [16] and Pix2PixHD [37] to explain each architecture. Let $ck - sn$ denote a 3x3 Convolution-RELU layer with k filters and stride n. dk denote a 7x7 Convolution-RELU layer with k filters and stride 1. $m2$ denotes a 2x2 max pooling layer. $sp - 19$ denotes a SPADE normalization-SELU layer [29, 20], which uses the 19 channels from the predicted semantic map as one of its inputs. tk denotes a 2x2 transposed convolution with k filters. Lastly, $r2$ denote a 1x1 Convolution layer with 2 filters

Input and Generated Image Encoder. The first encoder uses the same base architecture as VGG16 [33]: $c64-s1$, $c64-s1$, $m2$, $c128-s1$, $c128-s1$, $m2$, $c256-s1$, $c256-s1$, $c256-s1$, $m2$, $c512-s1$, $c512-s1$, $c512-s1$. This architecture outputs four feature maps, one after each resolution. In other words, we output the features after the max pooling layer, as well as the final feature map from the encoder. This encoder shares weights for encoding both the input and synthesized image.

Semantic and Dispersion Maps Encoder. This other encoder architecture is divided as: $d32$, $c64 - s2$, $c128 - s2$, $c256 - s2$. The encoder outputs the feature maps after each convolutional layer block. Note that we use different weights to encode semantic and uncertainty information.

Fusion Module. The fusion module concatenates the input, synthesized, and semantic feature maps at each resolution level. We then run a 1x1 convolution to extract important areas in the map, as well as reducing complexity. Finally, we perform a point-wise correlation between the dispersion feature map and the resulting map from the 1x1 convolution. This module will output a total of four feature maps - one for each resolution.

Decoder Blocks. There are four decoder blocks used in the dissimilarity network. The first and second one follow the same structure: $c256 - s1$, $sp - 19$, $c256 - s1$, $sp - 19$, $t256$. The third one is divided as: $c384-s1$, $sp-19$, $c128-s1$, $sp - 19$, $t1258$, while the last one follows: $c192 - s1$, $sp-19$, $c64-s1$, $sp-19$, $r2$. The first decoder block takes the feature map from the lowest resolution. All subsequent decoder blocks take as input the concatenation of the feature map from the fusion module and the output of the previous decoder block.

The dissimilarity network was trained for fifty (50) epochs, using the Adam [19] solver and a learning rate of 0.0001. We reduce the learning rate on plateau with a patience of 10 epochs. Additionally, we augment the training images by flipping around the vertical axis and normalizing them using mean and standard deviation values from ImageNet [32].

A.2. Re-Implementation of Image Re-Synthesis

Image Re-synthesis [24] is a synthesis-based framework for anomaly segmentation. It consist of a segmentation model *S*, a synthesis model *G*, and a discrepancy network *D*. Given a natural image, the framework will first predict a semantic label map with *S*. Then, the synthesis model *G* will re-synthesis the semantic map and finally the discrepancy network *D* detects meaningful distances caused by mislabeled objects by comparing the natural and synthesized images. The method adopts Bayesian SegNet [17] and PSP Net [40] as its segmentation models and Pix2PixHD [37] as its synthesis model.

Performance of re-synthesis methods in anomaly segmentation, such as Image Re-synthesis, are highly related to the quality of the segmentation and synthesis predictions. If we improve these predictions, the discrepancy network will have an easier task detecting anomalies.

To ensure that performance differences between our method and [24] do not come from the differences in the segmentation or synthesis modules, we re-implemented Image Re-synthesis with state-of-the-art segmentation and synthesis networks. Specifically, we replace their segmentation and synthesis networks with the same networks used in our framework ([42] for segmentation and [25] for synthesis). By doing so, the only differences between both methods are our contributions explained in Section 3.

Table 4 shows the differences between our implementation (Image Resynthesis++) against the results presented in [24]. In our experiments, we use the same datasets and metrics used in the original publication. Specifically, we use *Lost & Found* (L&F) [30] (i.e. images in a driving environment with small road hazards) and *Road Anomaly* [24] (i.e online images with anomalous objects located on or near the road) as our datasets, as well as the area under the curve for the receiver operating curve (AUC ROC) as our performance metric. We also added a variation of *Lost & Found*, where we restrict evaluation to the road, as defined by the ground-truth annotations.

Our implementation achieves better performance across the different datasets, thus concluding our Image Resynthesis++ to be a stronger baseline when compared to our framework. The final implementation was submitted to the Fishyscapes Benchmark (private test set) to ensure equal comparison in more challenging anomaly dataset, such as FS Lost & Found and FS Static.

We use the AUC ROC as our performance metric in these experiments since it was the only metric reported in the

Method	L&F	L&F (Road Only)	Road Anomaly
		↑ AUC ROC	
Image Resynthesis	0.82	0.93	0.83
Image Resynthesis++	**0.93**	**0.99**	**0.86**

Table 4. **Image Resynthesis implementation comparison**. Image Resynthesis++ outperforms the results presented in [24] using the same datasets (Lost & Found, Road Anomaly) with the area under the curve for the receiver operating curve (AUC ROC) as the performance metric.

original work. However, as mentioned in Section 4.1, ROC metrics are not well-suited for highly imbalance problems, such as anomaly detection, as explained in [5]. Thus, for our main experiments, we use more reliable metrics for imbalance problems, such as average precision (AP) and false positive rate at 95% true positive rate (FPR95).

Note that the Road Anomaly Dataset was not used in our main experiments, as it only contains sixty (60) images, which are not enough to ensure proper generalization abilities within anomaly segmentation. Additionally, the annotations are not consistent for the anomaly objects. For example, a rock in the middle of the road is labeled as an anomaly. However, the same style of rock next to the road is classfied as an inlier.

A.3. Computational Complexity

Table 5 shows the inference time for each module in the proposed approach. Note that the perceptual difference dispersion map also requires running a CNN. As such, we also added its inference time to the running complexity. The estimated times are the average of running an image one hundred (100) times in our framework. We use an NVIDIA 1080Ti GPU with 11GB GPU memory with an input resolution for each module as described in Sec. 4.1. Additionally, we also evaluate the inference speed of our lighter framework (explained in Sec. 5.2) as a comparison.

Module	Ours	Ours Light	Resolution
Segmentation	1256	47	2048x1024
Synthesis	192	62	1024x512
Perceptual Difference	13	13	512x256
Dissimilarity	53	53	512x256
Total (ms)	*1514*	*175*	–

Table 5. **Computational Complexity Study**. Inference time and input resolution for each module in the proposed framework. Average results for one hundred (100) experiments using NVIDIA 1080Ti GPU.

Method	FS L&F		FS Static	
	↑AP	↓FPR95	↑AP	↓FPR95
Ours w grid search	55.1	**39.6**	**61.5**	**25.6**
Ours w end-to-end	**59.6**	58.6	61.1	37.3

Table 6. **Ensemble Prediction Comparisons**. Grouping the predictions through end-to-end training shows comparable results in AP against empirically selected weights (grid search). However, end-to-end training increases significantly the FPR95 as the network gets overconfident with its predictions.

A.4. Ensemble with End-to-End Training

As stated in Sec. 3.3, there are benefits to be gained by combining the calculated uncertainty maps (i.e softmax entropy, softmax distance and perceptual difference) with the output of the dissimilarity network. In our main work, we combine these predictions using a weighted average, where the weights are selected empirically through a grid search. An alternative approach would be to learn these weights during the training of the dissimilarity network. This type of training would entail adding a learnable parameter (scalar) for each prediction map at the end of the dissimilarity network. Then, the model can optimize the weights to produce an end-to-end ensemble prediction.

Table 6 compares the results between empirical and lernable weights using the validation set for FS Lost & Found and FS Static. We first find a discrepancy in AP performance. The end-to-end ensemble performs better in FS L&F while empirical weights perform slightly better in FS Static. As stated in Sec. 5.1, we explain that the ensemble prediction with empirical weights outperforms in FS Static since it is easier for uncertainty methods to detect artificially blended objects. This behavior is less evident in the end-to-end training since the network optimizes the weights before generating a final prediction. In general, we expected end-to-end ensemble to outperform emperical weights in AP as the network optimizes its weights more efficiently.

The biggest insight from this comparison is shown when comparing the FPR95. In this metric, The end-to-end training significantly decreases performance when compared to empirical weights. These results are consistent with our ablation study in Sec. 5.1, where we show that deep CNNs (e.g. dissimilarity module) tend to be overconfident with its prediction. Thus, by training in an end-to-end fashion, we are still prone to generating overconfidence outputs. As we intent to deploy our framework in safety critical environments (e.g. autonomous driving), we selected the empirical weights as our best model.

A.5. Example Predictions

Figure 4 and Figure 5 display example predictions of our approach in validation images from FS Lost & Found and FS Static. Additionally, we show a qualitative comparison between our technique, an uncertainty estimation method (Softmax Entropy [14]), and an image re-synthesis method (Image Re-synthesis [24]). These images emphasizes the robustness of our framework for all anomalies scenarios (Figure 1), in comparison with previous methods. Note that the anomaly detection framework did not train with any of the anomalous instances, and they are seen for the first time during testing.

Additionally, some failure cases are presented in Figure 6. Common errors are derived scenes that differ urban landscape, anomaly instances that blend well with the background or small anomalous objects that only cover few pixels in the image.

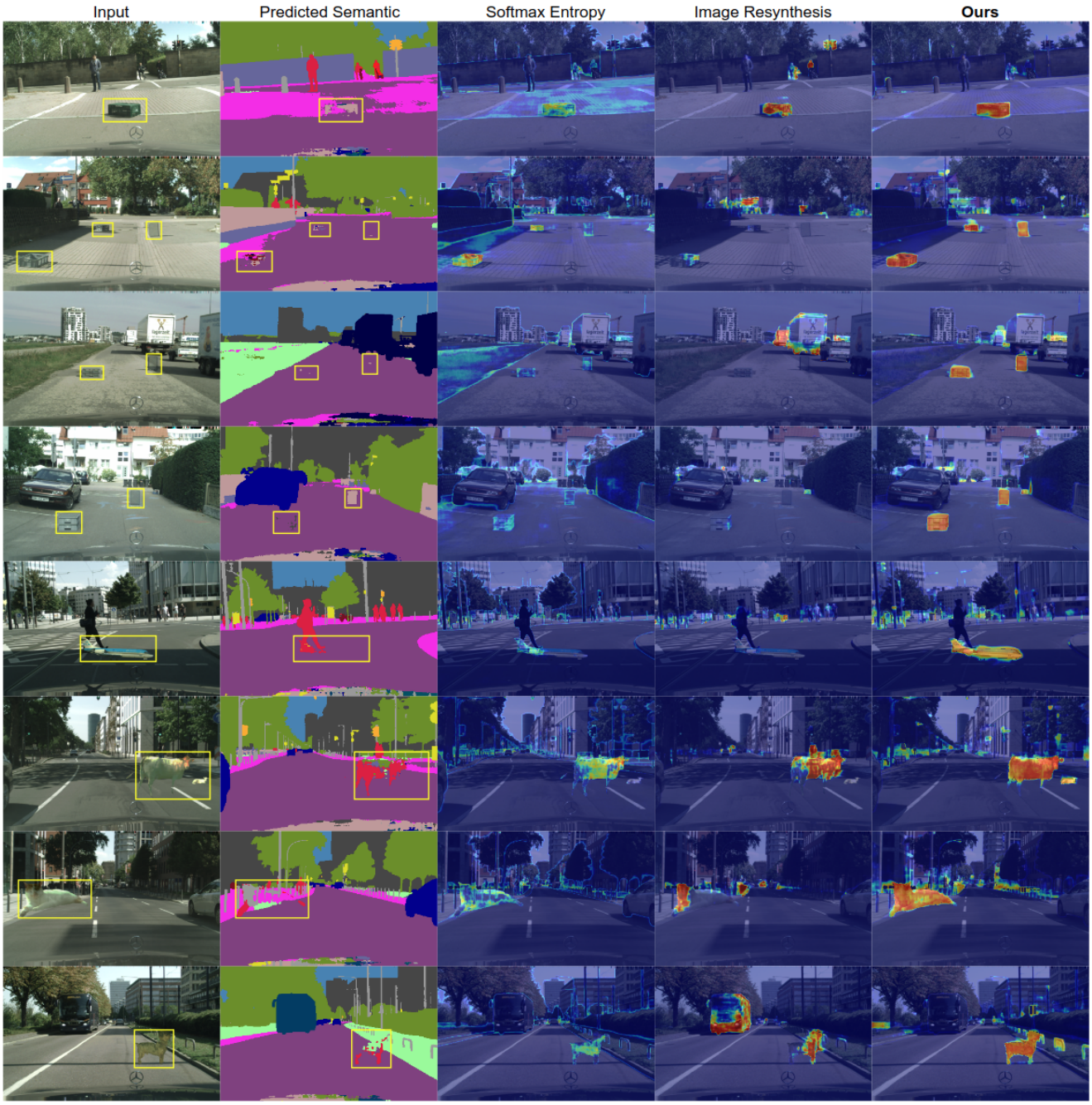

Figure 4. **Framework example predictions**. Qualitative comparison between proposed framework and baseline for uncertainty methods [14] and image-resynthesis methods [24]. The proposed framework outperforms both previous methods detecting anomalies instances. The first five images are from the FS Lost & Found, while the next five are from FS Static. Pixels labeled as *void* are excluded from the prediction visualizations for the three methods shown, as they are also excluded in the anomaly benchmarks.

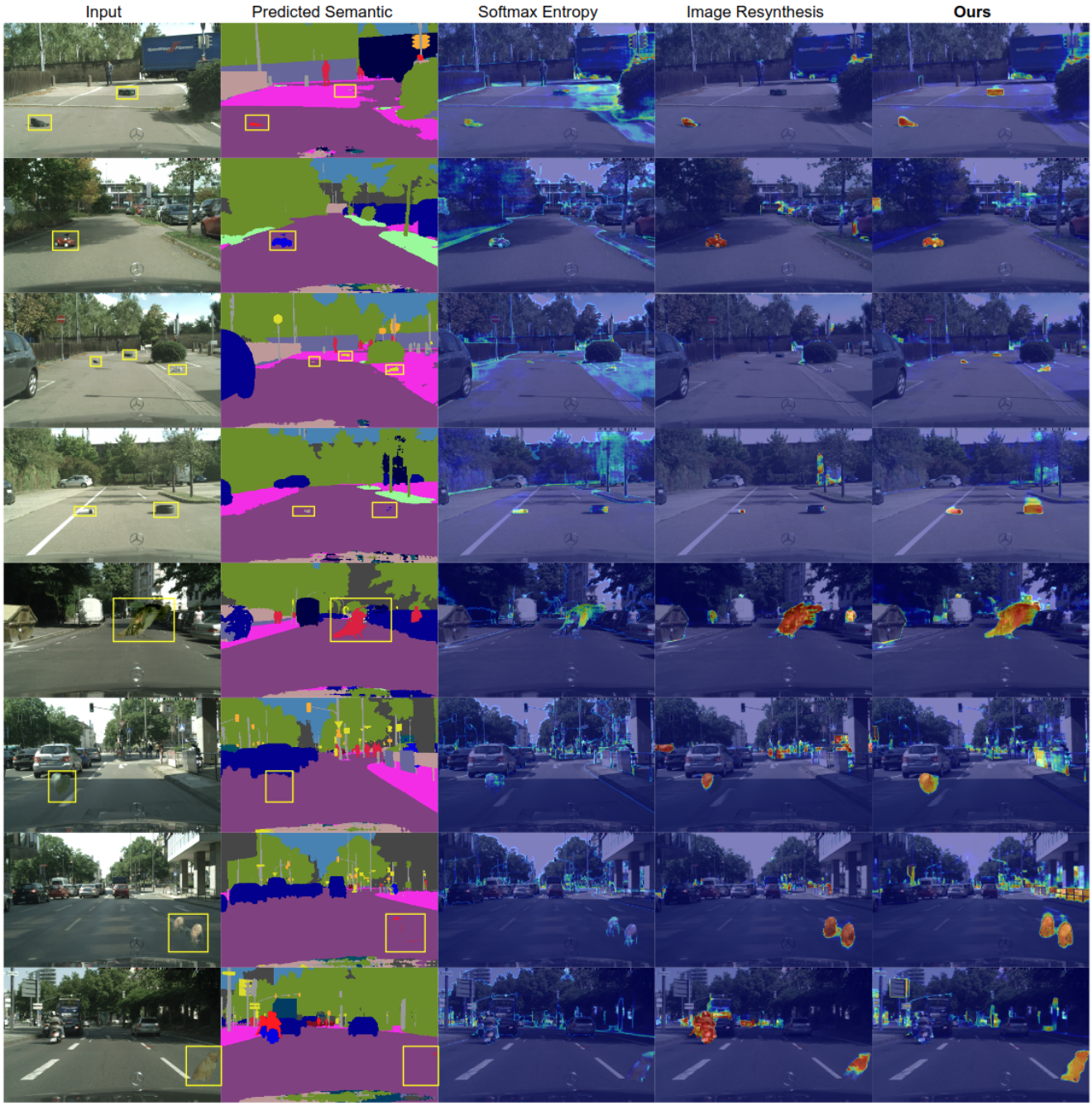

Figure 5. **Additional predictions examples.** Our framework reliable detects all three outcomes when a segmentation network encounters an anomalous instances, as explained in Figure 1. First four images are from FS Lost & Found, while the next four are from FS Static. Softmax Entropy [14] and Image Resynthesis [24] are also shown as reference. Pixels labeled as *void* are excluded from the prediction visualizations for the three methods shown, as they are also excluded in the anomaly benchmarks.

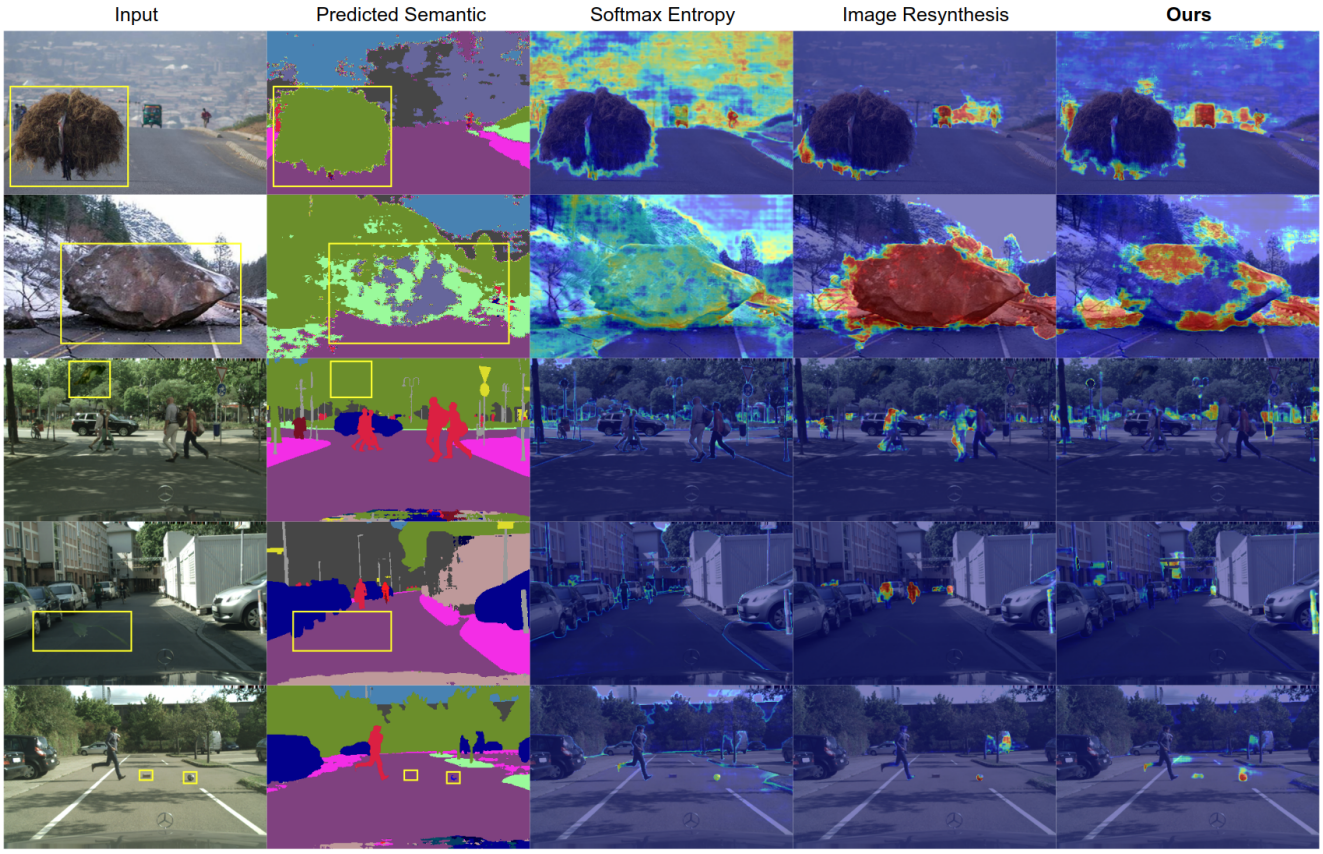

Figure 6. **Failure cases.** Our framework still fails at detecting some challenging anomaly instances. Common errors are derived scenes that differ urban landscape (top two images), anomaly instances that blend well with the background (middle two images) or small anomalous objects that only cover few pixels in the image. Softmax Entropy [14] and Image Resynthesis [24] are also shown as reference.