Real-Time Anomaly Segmentation for Road Scenes

1st Shayan Taghinezhad Roudbaraki
INGEGNERIA INFORMATICA
Politecnico di Torino
Turin, Italy
s301425@studenti.polito.it

2nd Francesco Baracco
INGEGNERIA INFORMATICA
Politecnico di Torino
Turin, Italy
s317743@studenti.polito.it

3rd Abolfazl Javidian
INGEGNERIA INFORMATICA
Politecnico di Torino
Turin, Italy
s314441@studenti.polito.it

***Abstract*—In this paper, we investigate the use of an existing real-time semantic segmentation model (ERFNet: Efficient Residual Factorized ConvNet) for detecting anomalies in road scenes. Ultimately, we explore whether Self-supervised pre-trained models can improve results or not. The code used in this paper is accessible at: https://github.com/shayanit/anomaly-segmentation-for-road-scenes.**

I. Introduction

Anomaly detection within images is challenging. Modern deep neural networks (DNNs) typically only predict a predefined set of semantic classes, making them unable to classify objects outside these categories. This overconfidence in predictions, even for previously unseen objects, poses challenges, especially in real-world applications like automated driving and robot interaction.

Since anomaly detection is crucial for ensuring safety and reliability, the trade-off between speed, accuracy, and resource management becomes critical in these scenarios. Systems need to process data rapidly and make timely decisions while maintaining high accuracy to detect anomalies and avoid false positives, which could lead to dangerous situations.

To address this challenge, our paper focuses on per-pixel anomaly segmentation methods. We compare several techniques that process the outputs of segmentation models trained on Cityscapes and aim to identify pixels likely to belong to anomalies within road scenes. We generate pixel-wise anomaly scores from the output of the segmentation models by analyzing the model's predictions to identify pixels that deviate significantly from normal patterns, indicating potential anomalies. These anomaly scores provide a measure of the likelihood that each pixel belongs to an anomalous region within the image.

II. Related Works

To effectively perform anomaly segmentation, several factors must be considered: 1. Dataset and benchmarks, 2. Evaluation metrics, and 3. Techniques and architectures. Let's delve deeper into each of these aspects.

A. Dataset and benchmarks

Cityscapes[8] dataset serves as a comprehensive benchmark for semantic labeling approaches, offering stereo video sequences from streets in 50 German cities. With 5,000 detailed pixel-level annotated images and 20,000 additional images with coarse annotations, Cityscapes is widely utilized for various computer vision tasks such as semantic and instance segmentation, object detection, and scene understanding.

Fishyscapes[9] is the first public benchmark specifically tailored for anomaly detection in urban driving scenarios. It comprises two datasets:

1) **FS Static:** Derived from the Cityscapes validation set, this dataset includes out-of-distribution (OoD) and in-distribution (ID) pixels, crucial for evaluating anomaly detection methods.
2) **FS Lost and Found:** Based on the original Lost and Found dataset, it provides pixel-wise annotations distinguishing objects, background, and void content, with 100 validation images and 275 test images.

Road Anomaly[11] dataset consists of 60 high-resolution images depicting various anomalous objects located on or near roads, serving as a valuable resource for training and evaluating machine learning models focusing on anomaly detection, object recognition, and scene understanding.

SegmentMeIfYouCan[12] dataset addresses anomalous object segmentation and road obstacle segmentation through two datasets:

1) **RoadAnomaly21:** Similar to FS Lost and Found, anomalies can appear anywhere in the image, providing diverse challenges for segmentation algorithms.
2) **RoadObstacle21:** All anomalies, or obstacles, are located on the road, presenting challenges in different scenarios such as nighttime, dirty roads, and snowy conditions.

B. Evaluation metrics

Precision measures the proportion of correctly identified anomalies (true positives) among all instances classified as anomalies.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{1}$$

A high precision indicates that when the algorithm predicts an anomaly, it is likely to be correct. However, it doesn't consider missed anomalies (false negatives).

Recall measures the proportion of correctly identified anomalies among all actual anomalies.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2}$$

A high recall indicates that the algorithm is able to identify most of the anomalies present in the dataset, but it doesn't consider false alarms (false positives).

Area Under the Precision-Recall Curve (AuPRC) is calculated by considering the precision and recall as functions of some threshold δ applied to the anomaly scores. This metric is particularly useful for assessing performance in scenarios with class imbalances, where the number of anomalies may be relatively small compared to normal cases.

False Positive Rate at 95% True Positive Rate (FPR95) indicates how many false positive predictions are made to achieve a 95% true positive rate. This metric is important for understanding the trade-off between false positives and true positives.

Mean Intersection-over-Union (mIoU) is a standard metric used to assess the performance of semantic segmentation networks. IoU is calculated as:

$$\text{IoU} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives} + \text{False Negatives}} \tag{3}$$

The global mIoU value is obtained by averaging the IoU class values computed over a batch of images, providing a comprehensive measure of segmentation accuracy by considering both false positives and false negatives.

C. *Techniques and architectures*

Several compact yet high-performance architectures, such as *BiSeNet* [7], *ENet* [5], and *ERFNet* [15] have been introduced and elaborated upon in the literature, offering specific computational optimizations and valuable insights.

1) **Bilateral Segmentation Network (BiSeNet)**
Bilateral Segmentation Network includes a Spatial Path and a Context Path. The Spatial Path, consisting of three layers with convolution operations, batch normalization [24], and ReLU activation [25], preserves spatial information while maintaining computational efficiency. This path produces output feature maps that are 1/8 the size of the original image. The Context Path complements this by providing a large receptive field. While some methods use complex operations [16,17,19,20], BiSeNet employs a lightweight model like Xception [26] to downsample feature maps quickly and uses global average pooling [16,17,27] to capture global context information. The up-sampled global pooling output is then combined with features from the lightweight model using an incomplete U-shaped structure [21,22,28].
Attention refinement module: Within the Context Path, an Attention Refinement Module (ARM) refines features at each stage by employing global average pooling to capture global context. It computes an attention vector to guide feature learning, integrating global context information efficiently without up-sampling, thus incurring minimal computational cost.
Feature fusion module: BiSeNet utilizes a specialized Feature Fusion Module to effectively merge features from different levels of the Spatial and Context Paths. This module concatenates output features from both paths, normalizes scales using batch normalization, and pools concatenated features to compute a weight vector for feature selection and combination, akin to SENet.
Loss function: BiSeNet employs a principal loss function to supervise the entire network output, supplemented by two auxiliary loss functions to oversee the Context Path output, akin to deep supervision. All loss functions are Softmax-based, with a parameter α balancing the weights of principal and auxiliary losses, facilitating effective model training. The principal loss function is defined as:

$$\text{Loss} = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\log\left(\frac{e^{p_i}}{\sum_j e^{p_j}}\right) \tag{4}$$

where p is the output prediction of the network.

$$\text{L}(X;W) = l_p(X;W) + \alpha\sum_{i=2}^{K} l_i(X_i;W) \tag{5}$$

where l_p is the principal loss of the concatenated output. X_i is the output feature from stage i of Xception model. l_i is the auxiliary loss for stage i. K is equal to 3 in this proposed method. L is the joint loss function. Notice that, only the auxiliary loss has been used in the training phase.

2) **Efficient Neural Network (ENet)**
ENet is a novel architecture designed for low-latency operation. The architecture is outlined in Table I, divided into stages with horizontal lines and denoted by the first digit after each block name. Output sizes are given for an input image resolution of 512 × 512. Inspired by ResNets, the architecture features main branches and extensions with convolutional filters that merge back with element-wise addition. Each block comprises three convolutional layers, as shown in Fig. 2(b): a 1 × 1

Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
4 × bottleneck1.x		$64 \times 128 \times 128$
bottleneck2.0	downsampling	$128 \times 64 \times 64$
bottleneck2.1		$128 \times 64 \times 64$
bottleneck2.2	dilated 2	$128 \times 64 \times 64$
bottleneck2.3	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.4	dilated 4	$128 \times 64 \times 64$
bottleneck2.5		$128 \times 64 \times 64$
bottleneck2.6	dilated 8	$128 \times 64 \times 64$
bottleneck2.7	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.8	dilated 16	$128 \times 64 \times 64$
Repeat section 2, without bottleneck2.0		
bottleneck4.0	upsampling	$64 \times 128 \times 128$
bottleneck4.1		$64 \times 128 \times 128$
bottleneck4.2		$64 \times 128 \times 128$
bottleneck5.0	upsampling	$16 \times 256 \times 256$
bottleneck5.1		$16 \times 256 \times 256$
fullconv		$C \times 512 \times 512$

Table 1. ENet architecture. Output sizes are given for an example input of 512 × 512.

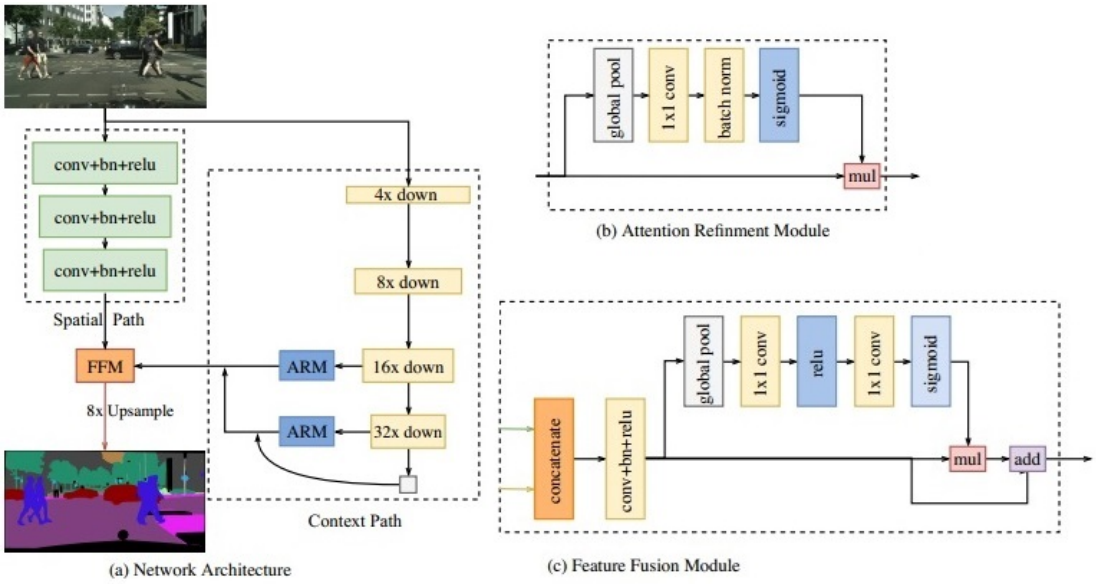

Fig. 1: Architecture of the Bilateral Segmentation Network (BiSeNet)

projection, a main convolutional layer, and a 1 × 1 expansion, with Batch Normalization [24] and PReLU [30] applied between each. We can refer to these as bottleneck modules. If downsampling, a max pooling layer is added. The first 1 × 1 projection is replaced with a 2 × 2 convolution with stride 2. Regular, dilated, or full convolutions are used, with occasional replacement by asymmetric convolutions. Spatial Dropout [31] is employed as a regularizer, with $p = 0.01$ before bottleneck2.0 and $p = 0.1$ afterwards.

The initial stage includes a single block, as shown in Fig. 2(a). Stage 1 has 5 bottleneck blocks, while stages 2 and 3 have the same structure, except stage 3 doesn't downsample the input initially. These three stages form the encoder. Stages 4 and 5 constitute the decoder.

Bias terms are omitted in projections to reduce kernel calls and memory operations, without impacting accuracy. In the decoder, max pooling is replaced with max unpooling, and padding with spatial convolution without bias. ENet did not use pooling indices in the last upsampling module, because the initial block operated on the 3 channels of the input frame, while the final output has C feature maps (the number of object classes). The last upsampling module doesn't use pooling indices, and the final module is a full convolution for performance reasons.

3) **Efficient Residual Factorized ConvNet (ERFNet)**
ERFNet is an efficient architecture for real-time semantic segmentation, aiming to overcome limitations in commonly used residual layers. These layers, found in many high-accuracy ConvNets, have inherent efficiency issues. By addressing these limitations, ERFNet efficiently utilizes parameters to achieve high segmentation accuracy while maintaining top efficiency for various applications.

Residual layers [33] have the property of allowing convolutional layers to approximate residual functions, as the output vector y of a layer vector input x becomes:

$$\mathrm{y} = F(x, W_i) + W_s(x) \tag{6}$$

where W_s is usually an identity mapping and $F(x, W_i)$ represents the residual mapping to be learned. This residual formulation facilitates learning and significantly reduces the degradation problem present in architectures that stack a large number of layers [33]. The original work proposes two instances of this residual layer: the non-bottleneck design with two 3×3 convolutions as depicted in Fig. 3 (a), or the bottleneck version as depicted in Fig. 3 (b). Both versions have a similar number of parameters and almost equivalent accuracy. However, the bottleneck requires fewer computational resources, and these scale in a more economical way as depth increases. Hence, the bottleneck design has been commonly adopted in state-of-the-art networks [33]. However, it has been reported that non-bottleneck ResNets gain more accuracy from increased depth than the bottleneck versions, which indicates that they are not entirely equivalent and that the bottleneck design still suffers from the degradation problem [33]. Therefore, a new residual layer implementation called "non-bottleneck-1D" (non-bt 1D) proposed, as shown in Fig. 3 (c). It uses 1D factorization to speed up computation and reduce parameters compared to the original non-bottleneck layer. This module maintains learning capacity and accuracy similar to the non-bottleneck design. Experimental results show a 33% reduction in parameters and faster execution time compared to the original non-bottleneck and bottleneck designs. This

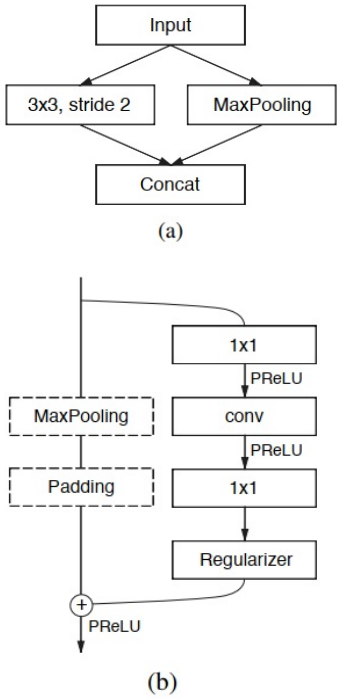

Fig. 2: (a) ENet initial block. MaxPooling is performed with non-overlapping 2 × 2 windows, and the convolution has 13 filters, which sums up to 16 feature maps after concatenation. This is heavily inspired by [32]. (b) ENet bottleneck module. conv is either a regular, dilated, or full convolution (also known as deconvolution) with 3 × 3 filters, or a 5 × 5 convolution decomposed into two asymmetric ones.

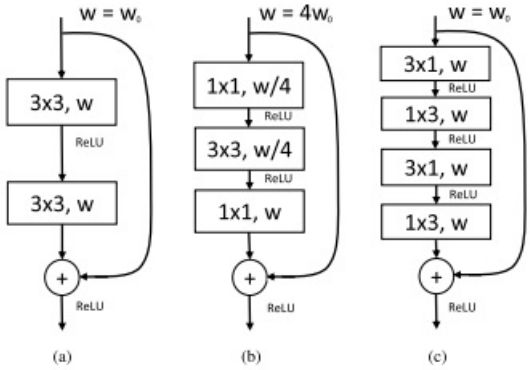

Fig. 3: Residual layers (a) Non-bottleneck (b) Bottleneck (c) Non-bottleneck-1D

approach is directly transferable to other networks using residual layers for both classification and segmentation tasks. Increasing the "width" of the network while minimizing computational resources has been shown to improve performance in classification networks, and proposed design extends this benefit to segmentation tasks, enhancing efficiency while maintaining accuracy.

III. Baseline Methodologies

A pre-trained ERFNet model on 19 Cityscapes classes was adopted and tested on three different standard evaluation methods [34]. The tests were conducted on datasets including RoadAnomaly21, RoadObstacle21, FS Lost and Found, FS Static, and Road Anomaly.

In the context of neural networks and classification tasks, a "logit", denoted as $f(x;\theta)$, refers to the raw, unnormalized output of a neural network for a given input x, parameterized by θ. The softmax function, denoted as σ, converts these logits into probabilities by exponentiating each logit and normalizing them. For example, given a logit vector $[a, b, c]$, the softmax probabilities would be $[\frac{e^a}{e^a+e^b+e^c}, \frac{e^b}{e^a+e^b+e^c}, \frac{e^c}{e^a+e^b+e^c}]$. In the context of pixel-level anomaly segmentation, the task is framed as a binary classification problem, where each pixel is classified as either "anomalous" (out-of-distribution, OoD) or "non-anomalous". The anomaly score $s_z(x)$ is calculated for each pixel $z \in Z$ in the image x. A threshold $\delta \in \mathbb{R}$ is then defined. If $s_z(x) \geq \delta$, the pixel is classified as anomalous; otherwise, it is classified as non-anomalous.

We employed the following baseline methods:

A. Maximum Softmax Probability (MSP)

The MSP (Maximum Softmax Probability) [35] method is an inference technique rooted in the observation that correctly classified in-distribution (ID) samples generally have higher maximum softmax probabilities compared to misclassified out-of-distribution (OoD) samples.

In simpler terms, this method assesses the likelihood of a pixel being anomalous based on how much its maximum softmax probability differs from 1. Higher differences indicate higher anomaly scores, suggesting a greater likelihood of the pixel being anomalous.

To compute the anomaly score of a specific pixel at location z the following formula is used:

$$\mathrm{s}_z(x) = 1 - \max_{c \in C} \sigma(f_z^c(x;\theta)) \tag{7}$$

where:

- $\sigma(f_z^c(x;\theta))$ calculates the softmax probability for the pixel belonging to class c.
- $f_z^c(x;\theta)$ represents the logit (the raw output of the network) for class c at pixel z in image x, parameterized by θ.

B. MaxLogit

The MaxLogit [36] method addresses a limitation of the MSP method when applied to in-distribution (ID) datasets with a large number of classes. In such cases, the probability mass can be distributed among visually similar classes, making it challenging to discern anomalies accurately.

In simpler terms, MaxLogit calculates the anomaly score by selecting the most underconfident prediction (i.e., the lowest logit value) across all classes. This approach helps in identifying anomalies by focusing on instances where the model is least certain about the classification. This is particularly useful in datasets with many visually similar classes, where probability mass can be spread thinly across various categories, making

anomaly detection challenging. To tackle this issue, MaxLogit calculates the anomaly score of a specific pixel at location z using the unnormalized logits. Here's the formula:

$$\mathrm{s}_z(x) = -\max_{c \in C} f_z^c(x;\theta) \tag{8}$$

-Note that the negative sign in front of the maximum operation means that we are interested in the minimum logit value among all classes. This effectively focuses on the most underconfident prediction, as a low logit value suggests the model is uncertain about the classification.

C. Maximized Entropy or Max Entropy

The Max Entropy [37] method is designed to account for the uncertainty present in the probability distribution generated by the softmax function, which is calculated from the logits. This method computes the anomaly score of a specific pixel at location z using the calculation of the distribution entropy. In simpler terms, the Max Entropy method calculates the anomaly score by summing the negative entropy of the softmax probabilities across all classes. This approach quantifies the uncertainty in the classification, with lower entropy values indicating higher confidence in the prediction.
The formula for Max Entropy method is as follows:

$$\mathrm{s}_z(x) = -\sum_{c \in C} \sigma(f_z^c(x;\theta)) \log(\sigma(f_z^c(x;\theta))) \tag{9}$$

where:
- $\sum_{c \in C}$ denotes the summation operation over all classes c in the set C, where C represents the set of all classes.
- $\log(\sigma(f_z^c(x;\theta)))$ computes the natural logarithm of the softmax probability.
- Note that the negative sign in front of the summation indicates that we are summing the negative entropy values. Entropy measures the uncertainty or disorder in a probability distribution. By summing the negative entropy values, we obtain a measure of certainty or confidence.

In evaluating the performance of anomaly segmentation methods, two holistic metrics were used: the area under the precision-recall curve (AuPRC) and the false positive rate at 95% true positive rate (FPR95). Additionally, mean Intersection-over-Union (mIoU) values were included along with anomaly inference results. The findings presented in Table 2 illustrate the performance of different methods. It is evident from the outcomes that the MaxLogit technique surpasses others across most datasets. Comparatively, the MSP method exhibits marginally inferior performance on average compared to the Max Entropy approach. This discrepancy can be attributed to the utilization of uncertainty within the probability distribution of the softmax output by the Max Entropy method, which aids in calculating anomaly scores and enhancing results. In summary, MaxLogit demonstrates superior capacity in distinguishing between visually similar classes, thereby enabling more precise predictions regarding pixel anomaly categorization.

IV. Experiment results

A. *Temperature Scaling*

Temperature Scaling [38] serves as a confidence calibration technique. Its aim is to address the issue of providing probability estimates associated with predicted class labels that accurately reflect their true likelihood of correctness. Our study delves into the impact of temperature scaling on the MSP inference method. Specifically, we explore how it influences the anomaly score assigned to a pixel located at position $z \in Z$, which is determined by the following criterion:

$$\mathrm{s}_z(x) = 1 - \max_{c \in C} \sigma(f_z^c(x;\theta)/T) \tag{10}$$

The temperature T is a floating-point parameter. In Table 3, we present results obtained with various temperature values ranging from 0.5 to 2. Through extensive experimentation with this method and adjusting the temperature parameter, we observed that the most favorable performance across most datasets occurs within the range of 1.1 to 2. Consequently, we established a search grid to explore this interval. After thorough evaluation, we determined $T = 1.5$ as the optimal trade-off temperature. This choice is supported by the observation that, on average, for temperature values greater than 1.1, the results for most datasets show improvement.

B. *Void Classifier*

Upon scrutinizing the anomaly segmentation performance of several baseline methods, we delved deeper into comprehending the impact of including the void class in the analysis. The metric values presented in Tables 2 and 3 are derived by excluding the void class output, also referred to as the 20th class in the Cityscapes dataset, commonly known as the background class. By regarding the void class output as an indicator of anomaly, we can calculate the anomaly score of a specific pixel located at position $z \in Z$ as follows:

$$\mathrm{s}_z(x) = \sigma(f_z^{void}(x;\theta)) \tag{11}$$

The obtained classifier is called Void Classifier [39]. We performed the tests on three different architectures: ENet, ERFNet and BiSeNet. The results are shown in Table 4. From the values obtained, we can see that ERFNet outperforms better than the other architectures on most datasets.

C. *Self-supervised pre-trained models of ResNet-50*

Self-supervised pre-trained models, particularly ResNet-50, have emerged as potent tools in anomaly detection tasks. ResNet-50, known for its depth and efficiency, serves as a robust backbone for feature extraction in various computer vision tasks. These models offer advantages in anomaly detection by learning from unlabeled data, capturing rich semantic information crucial for discriminating between normal and anomalous patterns. This is especially beneficial when labeled anomaly data is scarce or costly. Fine-tuning such models on specific anomaly detection datasets involves retraining their parameters on a smaller labeled dataset of normal and anomalous samples, tailoring their representations to the task at

Methods	mIoU	SMIYC RA-21		SMIYC RO-21		FS L&F		FS Static		Road Anomaly	
		AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓
MSP	72.20	35.87	**58.27**	7.216	25.474	5.341	50.414	7.632	40.172	15.286	76.208
MaxLogit	72.20	37.953	60.401	6.733	36.569	**19.035**	**43.516**	**12.334**	**36.132**	**17.498**	**71.127**
Max Entropy	72.20	**38.601**	59.416	**10.699**	**25.339**	9.243	50.106	8.887	40.033	16.123	76.238

Table 2. ERFNet anomaly inferences on RoadAnomaly21, RoadObstacle21, FS Lost and Found, FS Static, and Road Anomaly. all values are indicated as percentages. ↑ means larger values are better and ↓ smaller values are better. **Bold numbers** indicate the best results.

Method	mIoU	SMIYC RA-21		SMIYC RO-21		FS L&F		FS Static		Road Anomaly	
		AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓
MSP ($T = 0.5$)	72.20	31.518	**57.037**	5.244	26.255	2.763	54.555	6.762	41.271	14.406	76.434
MSP ($T = 0.75$)	72.20	33.836	57.42	6.239	25.686	4.02	51.459	7.158	40.601	14.843	76.283
MSP ($T = 1$)	72.20	35.87	58.27	7.216	25.474	5.341	50.414	7.632	40.172	15.286	76.208
MSP ($T = 1.1$)	72.20	36.536	58.701	7.656	25.421	5.82	50.011	7.826	**40.116**	15.452	**76.207**
MSP ($T = 1.5$)	72.20	38.346	60.525	8.023	**25.417**	7.284	48.546	8.525	40.12	15.998	76.288
MSP ($T = 2$)	72.20	**39.3**	62.521	**8.965**	26.058	**8.252**	**46.988**	**9.098**	40.542	**16.422**	76.51

Table 3. ERFNet anomaly inferences using temperature scaling on RoadAnomaly21, RoadObstacle21, FS Lost and Found, FS Static and Road Anomaly. All values are indicated as percentages. ↑ means larger values are better, and ↓ means smaller values are better. **Bold numbers** indicate the best results.

Models	mIoU	SMIYC RA-21		SMIYC RO-21		FS L&F		FS Static		Road Anomaly	
		AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓
ENet	59.50	14.749	94.395	**2.196**	91.117	1.412	53.441	10.231	94.701	9.793	97.566
ERFNet	72.20	**25.088**	**69.459**	1.403	**44.423**	**7.794**	**30.083**	**12.471**	**32.933**	**11.95**	**86.182**
BiSeNet	**78.86**	17.155	84.613	0.549	99.463	2.524	59.873	2.364	76.184	8.901	91.573

Table 4. Anomaly inferences using a Void Classifier on ENet, ERFNet and BiSeNet on RoadAnomaly21, RoadObstacle21, FS Lost and Found, FS Static and Road Anomaly datasets. All values are indicated as percentages. ↑ means larger values are better, and ↓ means smaller values are better. **Bold numbers** indicate the best results.

hand. This adaptation enhances their performance in anomaly detection by aligning learned features with dataset nuances. Moreover, these models can be augmented with additional techniques like attention mechanisms, ensemble learning, or post-processing methods to further refine predictions and enhance overall performance.

1) **Barlow Twins: Self-Supervised Learning via Redundancy Reduction**

The objective function of BARLOW TWINS [40] evaluates the cross-correlation matrix between the embeddings generated by two identical networks, each processing different augmented versions of the same batch of samples. The goal is to make this matrix approximate the identity matrix. This approach ensures that the embedding vectors of the augmented versions of a sample remain similar while minimizing redundancy between the components of these vectors. BARLOW TWINS is competitive with state-of-the-art self-supervised learning methods due to its conceptual simplicity, its ability to naturally avoid trivial constant (collapsed) embeddings, and its robustness to variations in training batch size.

It operates on a joint embedding of distorted images (Fig. 4). More specifically, it produces two distorted views for all images of a batch X sampled from a dataset. The distorted views are obtained via a distribution of data augmentations T. The two batches of distorted views Y^A and Y^B are then fed to a function f_θ, typically a deep network with trainable parameters θ, producing batches of embeddings Z^A and Z^B respectively. To simplify notations, Z^A and Z^B are assumed to be mean-centered along the batch dimension, such that

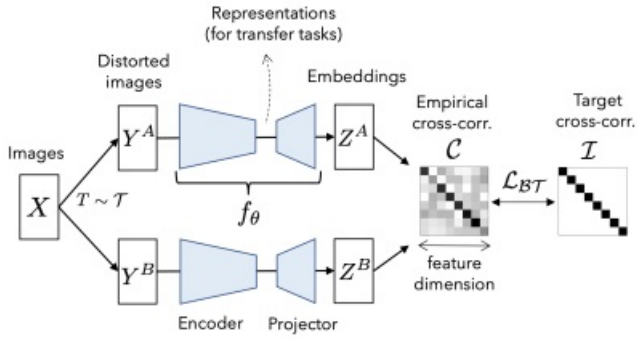

Fig. 4: Architecture of Barlow Twins

each unit has mean output 0 over the batch. BARLOW TWINS distinguishes itself from other methods by its innovative loss function L_{BT}:

$$L_{BT} = \underbrace{\sum_i (1 - C_{ii})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_i \sum_{j \neq i} C_{ij}^2}_{\text{redundancy reduction term}} \tag{12}$$

where λ is a positive constant trading off the importance of the first and second terms of the loss, and where C is the cross-correlation matrix computed between the outputs of the two identical networks along the batch dimension. C is a square matrix with size equal to the dimensionality of the network's output, with values ranging from -1 (i.e., perfect anti-correlation) to 1 (i.e., perfect correlation).

Intuitively, the invariance term of the objective, by trying to equate the diagonal elements of the cross-

correlation matrix to 1, makes the embedding invariant to the distortions applied. The redundancy reduction term, by trying to equate the off-diagonal elements of the cross-correlation matrix to 0, decorrelates the different vector components of the embedding. This decorrelation reduces the redundancy between output units, so that the output units contain non-redundant information about the sample.

2) **MoCo: Momentum Contrast for Unsupervised Visual Representation Learning**

Momentum Contrast (MoCo) [41] trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, \ldots\}$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued. The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder (Fig. 5). This method enables a large and consistent dictionary for learning visual representations.

From this perspective, contrastive learning constructs

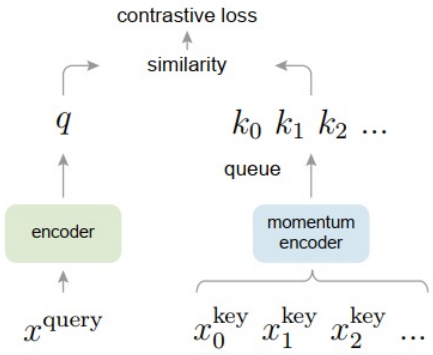

Fig. 5: Architecture of Momentum Contrast (MoCo)

a discrete dictionary from high-dimensional continuous inputs, like images. The dictionary is dynamic since the keys are randomly sampled, and the key encoder evolves throughout training. An effective features has been proposed that they can be learned through a large dictionary encompassing a diverse set of negative samples, while ensuring the key encoder remains as consistent as possible despite its evolution.

This method allows us to reuse the encoded keys from recent mini-batches. By using a queue, it decouples the dictionary size from the mini-batch size. This means the dictionary can be significantly larger than a typical mini-batch size and can be set flexibly as a hyper-parameter. The samples in the dictionary are continuously updated. This ensures the dictionary consistently represents a sampled subset of the entire dataset, with the extra computational overhead being manageable.

3) **SimSiam: Exploring Simple Siamese Representation Learning**

Siamese networks have become a common structure in various recent models for unsupervised visual representation learning. These models maximize the similarity between two augmentations of one image, subject to certain conditions for avoiding collapsing solutions. In this paper, we report surprising empirical results that simple Siamese networks can learn meaningful representations even using none of the following: (i) negative sample pairs, (ii) large batches, (iii) momentum encoders.

In this model, as shown in Fig.6, two augmented

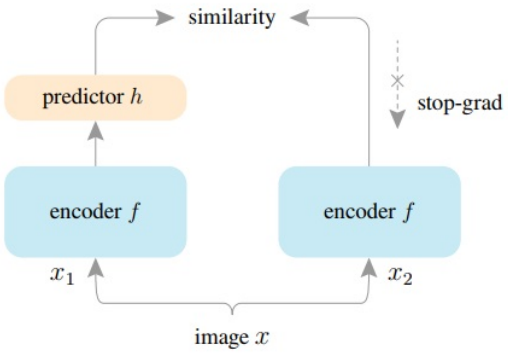

Fig. 6: SimSiam architecture

views of one image are processed by the same encoder network f (a backbone plus a projection MLP). Then a prediction MLP h is applied on one side, and a stop-gradient operation is applied on the other side. The model maximizes the similarity between both sides. It uses neither negative pairs nor a momentum encoder.

In other words, this model takes as input two randomly augmented views x_1 and x_2 from an image x. The two views are processed by an encoder network f consisting of a backbone (e.g., ResNet) and a projection MLP head. The encoder f shares weights between the two views. A prediction MLP head, denoted as h, transforms the output of one view and matches it to the other view.

In our approach to training ResNet-50 models for anomaly segmentation, we made significant modifications to the network architecture to better suit our specific task. Initially, we removed the last two layers of the standard ResNet-50, which consist of a fully connected layer and an average pooling layer. These layers are typically used for classification tasks and output a single prediction per image. To adapt the network for segmentation, we added a Conv2D layer at the end of the network. This Conv2D layer transforms the 2048 feature channels produced by the last convolutional block of ResNet-50 into 20 channels, corresponding to the 20 classes in our dataset. This layer effectively reinterprets the high-level features extracted by ResNet-50 into class-specific feature maps. Following the Conv2D layer, we incorporated an upsampling layer to adjust the output dimensions from $20 \times 7 \times 7$ to $20 \times 244 \times 244$, matching the original spatial dimensions of the input images. This upsampling step is crucial because it ensures that the final output maintains the

same resolution as the input images, allowing us to perform pixel-level classification required for segmentation tasks. By transforming the ResNet-50 model in this manner, we leverage its powerful feature extraction capabilities while tailoring the architecture to generate segmentation maps. This approach allows us to maintain the spatial context and fine-grained details necessary for accurate anomaly detection and segmentation in road scenes (see Fig. 7).

We then loaded the pretrained weights for each model variant, froze the backbone, and fine-tuned the network for up to 10 epochs, saving the weights that resulted in the lowest validation loss. Subsequently, we continued fine-tuning the models for an additional 10 epochs without freezing the backbone to evaluate the impact of unfreezing. The results are shown in Table. 5 and Table. 6 for freezing the backbone versus fine-tuning the backbone.

D. Analyze the effect of freezing the backbone versus fine-tuning the backbone

Based on our experiments using these ResNet50 pretrained models for anomaly segmentation of road scenes, we observed that models fine-tuned with the backbone frozen produced better results compared to those fine-tuned with the backbone unfrozen. Here are our conclusions explaining why this might be the case:

1) **Overfitting to the Fine-tuning Data**: Unfreezing the backbone increases the risk of overfitting to the specific characteristics of the fine-tuning dataset, like Cityscapes, reducing the model's ability to generalize to different anomalies or road scenes.
2) **Loss of Pretrained Representations**: Pretrained backbones capture general, robust features from diverse datasets. Unfreezing the backbone during fine-tuning can alter these general representations, degrading the useful, generalizable features learned during pretraining.
3) **Suboptimal Learning Rates and Training Dynamics**: Fine-tuning an unfrozen backbone requires careful tuning of hyperparameters, particularly learning rates. Incorrect learning rates can destabilize pretrained features or prevent sufficient adaptation, contributing to performance drops.
4) **Increased Complexity and Training Instability**: An unfrozen backbone introduces complexity due to more trainable parameters, leading to training instability, especially with smaller datasets. A frozen backbone limits training to fewer parameters, reducing the risk of instability and overfitting.

Now we can address the question: why do some pre-trained models perform better in anomaly segmentation tasks? Certain pre-trained models, like SimSiam and MoCo, outperform others, like Barlow Twins, due to differences in the diversity and representativeness of the features they capture during pretraining. SimSiam and MoCo might learn more robust and diverse representations, enabling better generalization to unseen anomalies. Barlow Twins may capture less discriminative features, leading to poorer performance. Moreover, the architectural design and self-supervised learning objectives affect performance. SimSiam and MoCo use techniques like momentum contrast and instance discrimination, encouraging learning of invariant and semantically meaningful features across diverse data samples. Barlow Twins, despite capturing complementary features, might not prioritize feature diversity or semantic consistency, resulting in suboptimal anomaly detection. In summary, fine-tuning with a frozen backbone maintains the integrity of robust, general features learned during pretraining, preventing overfitting and ensuring better generalization to anomaly segmentation tasks. Unfreezing the backbone increases the risk of overfitting, disrupts learned representations, and introduces training instability, leading to poorer performance. Thus, keeping the backbone frozen during fine-tuning is a more effective strategy for our specific task and dataset.

V. Conclusions

In our experiments with anomaly segmentation using techniques like MSP, MaxLogit, MaxEntropy, and the void classifier, we observed notable differences in performance among pre-trained models. Specifically, SimSiam and MoCo outperformed Barlow Twins but still lagged behind ERFNet. The primary reason for this disparity lies in the nature of the features learned during pretraining. SimSiam and MoCo, through their use of contrastive learning, are better at capturing diverse and meaningful representations, which helps in generalizing to anomaly detection tasks. Barlow Twins, however, may focus more on redundancy reduction and less on capturing diverse feature representations, leading to poorer performance. Despite this, even the robust features from SimSiam and MoCo cannot fully match the tailored design of ERFNet for semantic segmentation tasks, which is crucial for accurate anomaly detection.

ERFNet's superiority over ENet and BiSeNet in anomaly segmentation can be attributed to its efficient architecture, which strikes a balance between computational speed and segmentation accuracy. ERFNet's design, with its combination of downsampling and upsampling modules, ensures that it can process road scenes quickly while maintaining high accuracy, essential for real-time anomaly detection. ENet, while efficient, may lack the necessary depth and feature extraction capability, and BiSeNet, although powerful, may be too resource-intensive and complex for optimal performance in this specific task. ERFNet's streamlined architecture allows it to focus on capturing relevant features for anomaly detection without unnecessary complexity, making it the most effective model in our tests. Thus, ERFNet's tailored efficiency and architectural design make it particularly well-suited for anomaly segmentation tasks compared to other models.

References

[1] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," arXiv preprint arXiv:1505.07293, 2015.

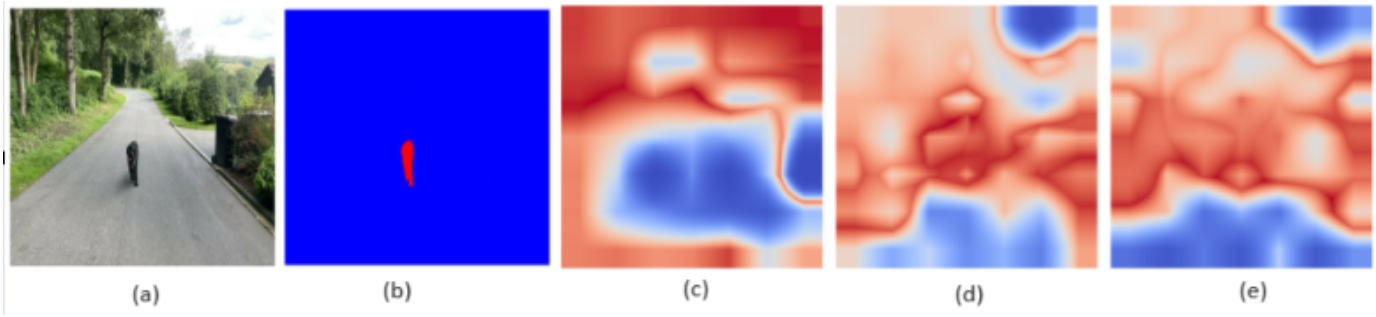

Fig. 7: This visualization compares an original image (a) with ground truth anomalies (b) and anomaly scores estimated by three Resnet50 models: BarlowTwins (c), MoCo (d) and SimSiam (e) using Maximum Softmax Probability inference. Hotter pixels represent higher anomaly scores.

Models	Methods	SMIYC RA-21		SMIYC RO-21		FS L&F		FS Static		Road Anomaly	
		AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓
BarlowTwins	MSP	14.972	**90.09**	0.719	**98.298**	0.375	94.481	**2.369**	85.318	11.854	**86.922**
	MaxLogit	11.563	94.34	**0.736**	98.967	0.359	95.766	1.949	83.32	10.484	92.316
	Max Entropy	12.205	90.454	0.637	98.697	0.499	94.398	2.147	85.182	10.757	88.089
	Void	**18.816**	93.374	0.631	98.381	**0.856**	**92.363**	1.469	**84.091**	**12.121**	88.887
MoCo	MSP	**16.863**	97.655	2.185	**59.191**	0.59	94.417	2.369	85.888	**14.658**	87.484
	MaxLogit	15.076	**94.034**	2.202	70.417	0.626	**87.721**	**2.795**	84.057	12.44	87.423
	Max Entropy	15.99	96.376	**2.213**	60.815	0.699	93.725	2.688	**82.316**	13.737	**87.021**
	Void	12.228	98.667	0.858	95.795	**0.801**	88.104	1.241	84.033	8.397	92.567
SimSiam	MSP	19.76	87.268	1.993	64.735	0.57	96.191	1.862	87.343	**13.907**	**86.378**
	MaxLogit	**20.385**	**75.177**	**2.809**	63.643	0.614	92.981	1.987	**83.337**	13.298	89.405
	Max Entropy	19.857	79.334	2.237	**62.646**	0.629	96.392	1.811	88.113	13.299	87.072
	Void	13.411	98.559	0.592	95.914	**0.708**	**91.044**	**1.149**	85.878	9.119	94.608

Table 5. Three Resnet50 models: BarlowTwins, MoCo, and SimSiam using **freezing the backbone** on RoadAnomaly21, RoadObstacle21, FS Lost and Found, FS Static and Road Anomaly. All values are indicated as percentages. ↑ means larger values are better, and ↓ means smaller values are better. **Bold numbers** indicate the best results.

Models	Methods	SMIYC RA-21		SMIYC RO-21		FS L&F		FS Static		Road Anomaly	
		AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓	AuPRC ↑	FPR95 ↓
BarlowTwins	MSP	**20.248**	88.728	**0.891**	**97.019**	0.398	94.022	2.782	82.831	14.957	**84.467**
	MaxLogit	15.375	91.437	0.799	99.034	0.375	95.161	2.634	**81.574**	14.553	88.595
	Max Entropy	16.048	**88.367**	0.731	97.566	0.396	94.122	**3.115**	82.69	**15.289**	85.688
	Void	18.719	92.439	0.608	97.724	**0.901**	**90.231**	1.538	82.904	12.304	86.395
MoCo	MSP	23.477	72.525	**2.531**	76.15	0.467	94.15	4.129	80.262	15.191	73.005
	MaxLogit	25.983	**70.612**	2.438	**64.148**	0.482	92.731	3.24	**70.116**	**17.156**	**71.747**
	Max Entropy	**26.777**	72.328	2.509	74.443	0.489	94.181	**4.215**	79.009	16.762	72.405
	Void	15.624	88.244	0.958	86.559	**0.984**	**86.151**	1.977	70.182	10.273	87.074
SimSiam	MSP	18.117	80.128	1.464	**70.045**	0.489	94.699	2.7	80.477	14.804	80.84
	MaxLogit	**23.129**	**65.527**	1.356	72.995	0.487	92.595	2.329	**73.054**	**16.442**	81.638
	Max Entropy	19.789	78.839	**1.47**	72.966	0.494	94.901	**2.88**	79.714	15.802	**80.118**
	Void	12.555	86.693	0.615	93.311	**1.219**	**85.05**	1.618	86.506	9.917	83.31

Table 6. Three Resnet50 models: BarlowTwins, MoCo, and SimSiam using **fine-tuning the backbone** on RoadAnomaly21, RoadObstacle21, FS Lost and Found, FS Static and Road Anomaly. All values are indicated as percentages. ↑ means larger values are better, and ↓ means smaller values are better. **Bold numbers** indicate the best results.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," arXiv preprint arXiv:1511.00561, 2015.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[5] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147, 2016.

[6] Eduardo Romera, Jose M. ´ Alvarez, Luis M. Bergasa, and ´ Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. IEEE Transactions on Intelligent Transportation Systems, 19(1):263–272, 2018.

[7] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In Proceedings of the European conference on computer vision (ECCV), pages 325–341, 2018.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3213–3223, 2016.

[9] Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. The fishyscapes benchmark: Measuring blind spots in semantic segmentation. International Journal of Computer Vision, 129:3119–3135, 2021.

[10] Peter Pinggera, Sebastian Ramos, Stefan Gehrig, Uwe Franke, Carsten Rother, and Rudolf Mester. Lost and found: detecting small road hazards for self-driving vehicles. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1099–1106. IEEE, 2016.

[11] Krzysztof Lis, Krishna Nakka, Pascal Fua, and Mathieu Salzmann. Detecting the unexpected via image resynthesis. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2152–2161, 2019.

[12] Robin Chan, Krzysztof Lis, Svenja Uhlemeyer, Hermann Blum, Sina Honari, Roland Siegwart, Pascal Fua, Mathieu Salzmann, and Matthias Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. arXiv preprint arXiv:2104.14812, 2021.

[13] Yu, C., Gao, C., Wang, J., Yu, G., Shen, C. and Sang, N., 2021. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. International Journal of Computer Vision, 129, pp.3051-3068.

[14] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, Jiaya Jia; Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 405-420.

[15] Eduardo Romera, Jose M. ´ Alvarez, Luis M. Bergasa, and ´ Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. IEEE Transactions on Intelligent Transportation Systems, 19(1):263–272, 2018.

[16] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv (2016).

[17] Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv (2017).

[18] Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G.: Understanding convolution for semantic segmentation. IEEE Conference on Computer Vision and Pattern Recognition (2017).

[19] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. IEEE Conference on Computer Vision and Pattern Recognition (2017).

[20] Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters-improve semantic segmentation by global convolutional network. IEEE Conference on Computer Vision and Pattern Recognition (2017).

[21] Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(12), 2481-2495 (2017).

[22] Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: A deep neural network architecture for real-time semantic segmentation. arXiv (2016).

[23] Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J.: Icnet for real-time semantic segmentation on high-resolution images. arXiv (2017).

[24] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448-456 (2015).

[25] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: International Conference on Artificial Intelligence and Statistics. pp. 315-323 (2011)

[26] Chollet, F.: Xception: Deep learning with depthwise separable convolutions. IEEE Conference on Computer Vision and Pattern Recognition (2017).

[27] Liu, W., Rabinovich, A., Berg, A.C.: Parsenet: Looking wider to see better. ICLR (2016)

[28] Xie, S., Tu, Z.: Holistically-nested edge detection. In: IEEE International Conference on Computer Vision (2015)

[29] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv (2017).

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," pp. 1026–1034, 2015.

[31] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 648–656.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," arXiv preprint arXiv:1512.00567, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun. (Dec. 2015). "Deep residual learning for image recognition." [Online]. Available: https://arxiv.org/abs/1512.03385

[34] Robin Chan, Matthias Rottmann, and Hanno Gottschalk. Entropy maximization and meta classification for out-ofdistribution detection in semantic segmentation. In Proceedings of the ieee/cvf international conference on computer vision, pages 5128–5137, 2021.

[35] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136, 2016.

[36] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joe Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for realworld settings. arXiv preprint arXiv:1911.11132, 2019.

[37] Robin Chan, Matthias Rottmann, and Hanno Gottschalk. Entropy maximization and meta classification for out-ofdistribution detection in semantic segmentation. In Proceedings of the ieee/cvf international conference on computer vision, pages 5128–5137, 2021.

[38] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In International conference on machine learning, pages 1321–1330. PMLR, 2017.

[39] Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Oct 2019.

[40] Zbontar, Jure, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. "Barlow twins: Self-supervised learning via redundancy reduction." In International conference on machine learning, pp. 12310-12320. PMLR, 2021.

[41] He, Kaiming, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 9729-9738. 2020.