Benchmarking the Robustness of Instance Segmentation Models

Yusuf Dalva, Hamza Pehlivan, Said Fahri Altindis, and Aysegul Dundar

Abstract—This article presents a comprehensive evaluation of instance segmentation models with respect to real-world image corruptions as well as out-of-domain image collections, e.g., images captured by a different set-up than the training dataset. The out-of-domain image evaluation shows the generalization capability of models, an essential aspect of real-world applications, and an extensively studied topic of domain adaptation. These presented robustness and generalization evaluations are important when designing instance segmentation models for real-world applications and picking an off-the-shelf pretrained model to directly use for the task at hand. Specifically, this benchmark study includes state-of-the-art network architectures, network backbones, normalization layers, models trained starting from scratch versus pretrained networks, and the effect of multitask training on robustness and generalization. Through this study, we gain several insights. For example, we find that group normalization (GN) enhances the robustness of networks across corruptions where the image contents stay the same but corruptions are added on top. On the other hand, batch normalization (BN) improves the generalization of the models across different datasets where statistics of image features change. We also find that single-stage detectors do not generalize well to larger image resolutions than their training size. On the other hand, multistage detectors can easily be used on images of different sizes. We hope that our comprehensive study will motivate the development of more robust and reliable instance segmentation models.

Index Terms—Deep networks, domain adaptation, image corruptions, instance segmentation, robustness.

I. INTRODUCTION

CONVOLUTIONAL neural networks (CNNs) have achieved impressive results on various computer vision tasks for scene understanding [11], [19], [22], [23], [45], [49], [56]. These CNN models are evaluated and continuously improved based on hyper-parameter and architectural searches conducted on clean validation subsets. Such validation subsets share similarities with their corresponding training sets, e.g., collected with the same camera, at a similar time of the day and weather [13], and sometimes even in the same room [37]. However, when these models are deployed for real-world applications, they may be used to infer on various domains, such as from images with corruptions to images collected by different cameras and with different pre-processings applied. Therefore, it is vital to evaluate models across different domains to measure their robustness, generalization, and reliability.

Understanding and analyzing the robustness of deep networks for corruption is an important topic since corrupted images naturally exist in many forms. Corruption in the image may be caused by environmental factors such as occlusions on the camera lens due to rain, mud, or frost; the image blurs due to a fast-moving camera, and various noise corruptions due to a defect in the recording hardware or sensor. Hence, it is important that we build robust models across image corruptions. Additionally, when a model is deployed, test images, even when they are clean, may exhibit significant differences with the training sets the model was trained on. For example, they may be recorded with different aspect ratios, different camera parameters, and under different illuminations. These differences in the test and train time data cause considerable degradation in terms of accuracy, and many works propose to adapt models onto new test domains, which is an important research topic known as domain adaptation [20], [52], [59], [65]. Domain adaptation methods require a pipeline for each test domain and may be impractical if there is no fixed test domain. Therefore, building robust models that generalize well across domains is another important research endeavor.

In this work, we conduct an extensive evaluation of instance segmentation models across various real-world domains. We do not target adapting a CNN model to a pre-defined domain such as a specific type of corruption. This would require specific actions for that domain, e.g., adding their corresponding augmentations. Such an assumption would be impractical given the large number of existing domains. Instead, we are interested in evaluating models trained on clean datasets with respect to a broad range of image corruptions and images collected from different environments that exhibit domain gaps with the training data.

We are interested in instance segmentation models because of their wide range of applications. They are an important part of autonomous driving applications [13] and robotics [72]. Additionally, off-the-shelf instance segmentation models are used to generate labels to train other tasks such as image synthesis [21], [75] as well as 3-D reconstruction [3], [18]. Considering the application domains, these models are...
expected to achieve good accuracies on a diverse set of images. For a task at hand, while it may be attractive to pick the best performing model on the clean validation set to deploy, it may not be the model that generalizes best to other domains. In this article, we aim to investigate the instance segmentation models’ robustness across domains and find the ingredients of the models that enhance the robustness.

In summary, our contributions can be summarized as follows,

1) Comprehensive benchmarking of instance segmentation algorithms with different network architectures, network backbones, normalization layers, and network initializations.
2) Measuring robustness across a wide range of real-world image corruptions as well as measuring robustness across out-of-domain image collections, which is an important evaluation set-up for domain adaptation.
3) Measuring the effect of joint-task training on robustness by comparing detection and instance segmentation models.
4) Based on this study, we provide several new insights. We find that normalization layers play an important role in robustness. Single-stage instance segmentation models are more robust to image corruptions. On the other hand, multistage instance segmentation models achieve better generalizations to other image collections that contain objects with a wide range of scales. Finally, we find that network backbones and copy-paste augmentations affect robustness significantly. Surprisingly, ImageNet pre-training compared to training from scratch does neither affect the robustness nor the generalization of the model. We provide more observations and insights in Section V.

II. RELATED WORK

In this section, we first review related work on robustness studies and domain adaptations. Then we extensively review instance segmentation methods that are included in this article.

A. Robustness Studies

Recently, deep networks are evaluated under real-world image corruptions for image classification [28], [33], object detection [54], and segmentation [40] tasks and found to have degraded performance in terms of accuracy compared to when tested on clean images. Heavy occlusions [42] are the other challenges to the robustness of instance segmentation models, but we leave it out of the scope for this work. Various methods are proposed to improve the robustness of deep networks [39], [79]. One popular approach is to increase their network bias toward object shapes which has become popular after the discovery that deep networks trained on ImageNet are biased toward texture [27]. Researchers propose to provide shape bias to networks with data augmentations, for example, by stylizing training images [27], [54], and by alpha-blending, a portion of RGB training images with texture-free representations, which are semantic maps with randomly chosen colors that are fixed for each semantic class [41]. Domain randomization is another technique to provide a shape bias by aggressively augmenting synthetic data with various color manipulations [61]. It is shown to be successful in generalizing a model learned from synthetic data to work on real-world cases [61], [62].

B. Domain Adaptation

Studies have demonstrated that deep learning methods suffer from significant degradation in terms of accuracy when the test time input feature distributions differ from the training datasets [52], [65]. The change in the distribution may come from different noise conditions [57], synthetic versus real image datasets [20], among images that are collected in different cities, with different lighting, and season [76] just to name a few. Since the differences in distributions may also come from real-world image corruptions, improving robustness and generalizing the model to an unseen domain go hand in hand. The adaptation is usually proposed to adapt a machine learning model to a pre-defined target dataset by aligning two domains of images via adversarial training [58], [60], aligning feature levels in networks [26], [34], [64], and network output level [63], or a combination of several [20]. In this article, we are not interested in adapting a network to a predefined target dataset but in finding out which recipes show improved robustness across unseen domains.

C. Instance Segmentation Studies

Instance segmentation is a challenging computer vision task that requires correct pixel-level and instance-level predictions simultaneously. There are dedicated surveys on topics of object detection and instance segmentation [38], [50], [78] for further interested readers. In this section, we review different instance segmentation models and recipes that are included in our benchmark.

1) Network Architectures: In this benchmarking study, we investigate different choices of instance segmentation architectures. We categorize them as multistage and single-stage architectures.

a) Multistage architecture: They include Mask-RCNN [31], PointRend [43], and QueryInst [25] frameworks. Mask-RCNN models are one of the most popular instance segmentation models with many variations that include dilated convolution, feature pyramid network, and versions with deformable convolution and cascaded architectures. These multistage architectures are built on Faster-RCNN and follow a detect and then segment paradigm. They are explained below.

1) C4: The original implementation of Faster R-CNN [56] with ResNet component extracting features from the final convolutional layer of the fourth stage, which is referred to as C4. R50-C4 denotes this backbone with ResNet-50 architecture.

2) DC5: Dilated convolution [77] is a modified version of the default convolution algorithm where calculations of some points are skipped according to the given set of dilation parameters. As a consequence, it has a larger receptive field than the standard convolution layers.
In [14], dilation is used on the fifth convolutional layer of ResNet architecture. This method is referred to as DC5.

3) **FPN**: Feature pyramid network [46] is a feature extractor that can generate feature maps at different scales from a single-scale image in a fully convolutional manner. FPN includes two main components: bottom-up and top-down pathways. The bottom-up pathway is the usual convolutional network for feature extraction as it calculates feature maps at different scales. The computed outcome of each stage’s last residual block is used as feature activations for ResNets. On the other hand, the top-down pathway creates higher-resolution features by upsampling feature maps from higher pyramid levels. The generated features are improved with features from the bottom-up pathway using lateral connections to obtain these improved features.

4) **FPN-Deform**: Deformable convolution augments the spatial sampling locations in the modules with additional offsets providing better flexibility compared to the default convolutional layer, which is limited to geometric model transformations due to the fixed geometric structures in their building modules [14], [80]. Deformable convolution can adapt to the geometric variations of objects and has shown to be beneficial in object detection, instance segmentation tasks [14], [80] and has been successfully incorporated for image synthesis [1], [53]. Deformable convolution is used in the FPN model by replacing the last set of convolution layers.

5) **FPN-Cas**: Cascade R-CNN [6], [7] is a multistage extension of the R-CNN, where the detector stages deeper into the cascade, and the model becomes sequentially more selective against close false positives. The cascaded architecture is extended to Mask-RCNN with the addition of a mask head to the last stage. The cascaded head is added to the FPN architecture and referred to as FPN-Cas.

6) **PointRend**: Point-based rendering [43] module is inspired by the sampling methods used for efficient rendering algorithms. PointRend starts with a coarse instance segmentation prediction and adaptively samples points on the image in a higher resolution to achieve fine-detailed instance predictions. It replaces the default mask head of Mask-RCNN with a module that iteratively computes predictions of a set of points with a small MLP (multilayer perceptron).

7) **QueryInst**: Recently, query-based set predictions are used to tackle object detection and instance segmentation tasks [8], [15], [25]. Instances as queries [25] is one of them, which uses dynamic mask heads in the query-based end-to-end detection framework. The dynamic mask heads are set in parallel with each other, and they transform each mask RoI feature adaptively according to the corresponding query. They are simultaneously trained in all stages. During inference, all the dynamic mask heads in the intermediate stages are thrown away, and only the final stage predictions are used for inference.

b) **Single-stage architecture**: They include TensorMask [12], SOLO [68] (SOLO, SOLOv2), YOLACT [4] (YOLACT, YOLACT++), and PolarMask [73]. Single-stage architectures are more challenging to build and first successful single-stage methods are developed after second-stage architectures were well-established such as Mask-RCNNs.

1) **TensorMask** [12] is a single-stage model that uses a dense sliding window approach to produce instance masks. The instance predictions are encoded with 4-D tensors for each window. This method achieves almost comparable results with Mask-RCNN.

2) **SOLO** [68] is another one-stage instance segmentation model which estimates masks based on their locations without outputting any bounding boxes. An image is divided into a grid, and according to the coordinates of the object center, an object instance is assigned to one of the grid cells. There is an instance mask prediction for each grid, and it is outputted as the final prediction if a semantic category is detected for that grid. Since the predictions of instance masks need to be spatially variant, normalized pixel coordinates are fed to a convolutional layer at the beginning of the network. ResNet backbones and FPN architecture are used by default.

3) **SOLOv2** [69] improves upon SOLO by replacing the mask predictions that are calculated for each grid separately by decoupled mask kernel prediction and mask feature learning. Mask kernel predictions are used as convolution weights on the mask features. This dynamic instance segmentation model provides a more efficient mask representation and learning and additionally enables higher-resolution mask predictions for finer details.

4) **PolarMask** [73] predicts the contour of instance through instance center classification and dense distance regression in a polar coordinate via fully convolutional single-stage architecture. It uses FPN to extract features of different levels.

5) **YOLACT** [4] is an another one stage instance segmentation model. The architecture is similar to YOLO [55], a single-stage detector. YOLACT extends the bounding box detection branch to estimate mask coefficients in addition to the bounding box locations and class confidence. The mask coefficients are used to linearly combine the prototypes, which are estimated in parallel via a second branch. The architecture is built on various backbones with FPN.

6) **YOLACT++** [5] improves on YOLACT by utilizing deformable convolution layers, optimized numbers of anchors, and a mask re-scoring regime [35]. Deformable convolutions are used in the last three ResNet stages with an interval of 3 (i.e., skipping two ResNet blocks in between), resulting in total 11 deformable layers.

2) **Network Backbones**: We experiment with instance segmentation models based on different types of backbone architectures that are both revolutionary in terms of network design and comprehensive in terms of instance
segmentation scores. In this regard, we experiment using ResNet [32], ResNeXt [74], SpineNet [17], Swin [51], and MViTV2-T [44].

1) **ResNet:** ResNet architecture [32] is commonly used for various computer vision tasks due to its accuracy. For our robustness analysis, we use ResNet-50 (R50) and ResNet-101 (R101) architectures, where the number indicates the depth of the backbone networks. These networks involve residual blocks formed by three consecutive convolutional blocks, as specified by the original study. In these backbones, the residual connection is defined as an element-wise summation of extracted feature maps.

2) **ResNeXt:** In addition to the existence of residual connections between building blocks, ResNeXt [74] introduces the concept of split-transform-merge to backbones with residual connections. Originating from the representation of a neuron combining input features with learned weights, ResNeXt offers a network structure aggregating set of transformations. This structure introduces cardinality to building blocks, which denotes the number of transformation blocks to be aggregated. With these stronger representations, the ResNeXt structure achieves learning a more complex representation compared to ResNet variants. In order to assess the robustness of this increase in predictive performance, we include a ResNeXt containing 101 layers in our experiments and refer to it as X101.

3) **SpineNet:** This architectural paradigm adopts scale-permuted architecture design instead of scale-decreased methods [17]. The study introducing this methodology argues that the success of architectures that apply feature downsampling in consecutive levels does not translate well to tasks requiring simultaneous recognition and localization. Different from scale-decreased methods, SpineNet backbones are constructed after performing a neural architecture search (NAS) and consider permutations of different design decisions on network design in a learnable way using reinforcement techniques. We include SpineNet-49/96/143 architectures in our robustness studies, where the numerical value defines the number of layers.

4) **Swin:** Transformer based architectures have been recently popularized and achieve state-of-the-arts on many computer vision tasks [16]. Transformer-based backbones split images into patches and treat each patch as a word embedding. Patches are processed individually, followed by an attention mechanism across patches. We experiment with a recently proposed Swin architecture [51], which computes cross-windows self-attention by shifting windows between layers so that attention computation is not limited to the same patches. We experiment with two variants of Swin architecture, SwinT, and SwinS, which have a complexity similar to those of ResNet-50 and ResNet101, respectively.

5) **MViTV2-T:** Another transformer based architecture we experiment with is MViTV2-T [44], multiscale vision transformers. MViTV2-T proposes to process high-resolution visual input by using pooling attention to overcome computational cost. Experiments of MViTV2-T show that their proposed pooling attention is more effective than local window attention, which is used by Swin architectures.

3) **ImageNet Pretraining and Normalization Layers:** Since the initial breakthrough of deep learning applied to object detection and instance segmentation, fine-tuning networks that were pretrained on the ImageNet dataset has been a popular recipe [31], [46], [49], [56]. However, it is recently shown that ImageNet pre-training may not be beneficial if the dataset is big enough [30], [81]. Furthermore, they show that better accuracies are obtained when starting from random initialization at the expense of longer iterations on the COCO dataset. Our experiments investigate if models trained from random initialization show less or more robustness to image corruptions than their ImageNet pretrained counterparts.

We couple these experiments with different choices of normalization layers. Normalization layers, since the discovery of batch normalization (BN) [36] layer, have become an important ingredient of deep neural networks. Various normalization layers are proposed for different tasks [2], [36], [66], [70] that are applied to different groupings of input features. Since they collect statistics of input features from datasets, their role in robustness is significant as image corruptions or out-of-domain images may exhibit different feature statistics. Among these normalization layers, we compare the models that use BN, synchronized BN (SyncBN), and group normalization (GN) layers since they are shown to work well on instance segmentation models.

1) **BN** [36] is a feature normalization module that is found to be helpful in training deeper and more accurate networks. BN normalizes the features by computing the mean and standard deviation for each channel in each batch. BN, as in other normalization methods, learns a trainable scale and shift parameter for each channel. Since the statistics are collected over a batch, small batch sizes severely degrade the accuracy of BN. When pretrained networks are tuned for instance segmentation tasks, the batch statistics of the pretrained model can be frozen since instance segmentation models can only fit small batches to a GPU, given that they are trained on high-resolution images. Otherwise, BN cannot be used to train instance segmentation models from scratch.

2) **SyncBN** [48] is an extension of BN that collects batch statistics computed across multiple devices (GPUs). SyncBN is an essential module for training models that can only fit a few images to each GPU, such as instance segmentation model trainings. When few images, e.g., only 1–2 images, fit a GPU, BN cannot compute robust statistics, and SyncBN resolves this issue by increasing the effective batch size when using many GPUs. This extension also enables training from scratch instance segmentation models.

3) **GN** [70] is one of the proposed alternatives to BN. Different than other feature normalization methods,
Fig. 1. Image from the COCO validation dataset, the original clean version, and corrupted ones by different noise models (severity = 3). B. = blur, N. = noise, D. = digital, and W. = weather. Rows 2 and 4 show inference results of the R50-FPN model on images from rows 1 and 3, respectively. On blur corrupted images, traffic lights and most of the persons are no longer detected. With impulse noise, the pole is detected as a tie. While changing the brightness of an image does not cause a significant accuracy drop, the parking meter is no longer detected in this example. The model does not detect persons on the bus and falsely detects a person on top of the bus on the image with JPEG compression. Snow and frost cause falsely duplicated detections on the bus, which are accurately detected on the other corrupted images.

GN divides channels into groups and computes statistics per group. Therefore, with this grouping, the computation is independent of the batch dimension as it operates on each example independently.

4) Copy-Paste Augmentation: Data augmentation is an essential part of computer vision applications to increase the effective data size and improve results. Other than standard augmentation techniques like color and scale jittering, datasets with instance annotations allow copy–paste augmentation of instances. Copy-paste augmentation is recently performed with a simple pipeline where between two randomly sampled images, instances from one image are pasted onto the other after both images going through scale augmentations, creating a large variety of unique training images [29]. With this approach, significant improvements are obtained in instance segmentation scores with longer trainings. For example, where training a network from scratch without such augmentation requires around 100 epochs to converge (the validation set performance no longer improves), the models trained with this augmentation require around 400 epochs. Even though the improvements over the baseline are already visible around epoch 100, longer training further boosts the accuracy. This benchmarking study investigates the robustness of these models, which are trained with and without copy-paste augmentation. We experiment with this augmentation because it is specifically proposed for the instance segmentation task and achieves significant improvements.

III. METHODS

We evaluate the robustness of instance segmentation models under a wide range of image corruptions [33]. The corruptions are applied to validation sets and not used during training. Models are trained on noncorrupted (clean) COCO training set [47]. Additionally, we investigate robustness across image collections that exhibit domain gaps with COCO. For this setting, we use models trained on the COCO training set and validate them on the Cityscapes [13] and BDD [76] datasets.

A. ImageNet-C Corruptions

We investigate many image corruptions from the ImageNet-C dataset [33]. These corruptions are mainly categorized into four categories, several types of: 1) blur (motion, defocus, gaussian); 2) noise (gaussian, impulse, shot and speckle); 3) digital (brightness, contrast, and JPEG compression); and 4) weather (snow, spatter, fog, and frost). Examples of these corrupted images are shown in Fig. 1. Each corruption can be tuned for different severity. We find severity set to three looks realistic and challenging enough and use this setting as the standard in our evaluations. Corruptions are as follows.

1) Blur corruptions include motion blur, which results from a fast-moving camera when recording. Defocus blur happens when an image is out of focus. Gaussian blur is the last blur corruption, which is implemented...
TABLE I  
BLUR CORRUPTION PARAMETERS

| Motion             | Defocus                       | Gaussian          |
|--------------------|-------------------------------|-------------------|
| Wand motion blur   | OpenCV Gaussian blur function | The skimage       |
| library with radius| with $3 \times 3$ kernels,   | gaussian filters  |
| 15, sigma 8, and   | and sigmaX values of 0.5      | are used with a   |
| a randomly sampled |                               | sigma value of 3. |
| angle from a       |                               |                   |
| uniform distribution|                              |                   |
| between (-45, 45). |                               |                   |

with a low-pass filter and outputs a blurred pixel by weighted averaging a pixel with its neighbors. The libraries and parameters used for these corruptions are given in Table I.

2) Among noise corruptions, Gaussian noise appears in low-lighting conditions. It is implemented as a summation of the input image and a noise map where the noise is generated from a normal distribution with a scale of 0.18. Impulse noise naturally happens due to a defect in recording hardware or sensor and corrupts some of the pixels in an image. It is implemented as a color analog of salt-and-pepper noise by using the skimage random noise function with an amount of 0.09. Shot noise is electronic noise caused by the discrete nature of light itself. It is implemented with a Poisson distribution parameterized by the input image multiplied by 12. The image is then normalized by 12 to be in the correct range. Speckle noise occurs due to the effect of environmental conditions on the imaging sensor during image acquisition. It is an additive noise and the noise added to a pixel tends to be proportional to the original pixel intensity. We use the parameter of 0.35 for the normal distribution’s scale and multiply it with the input image to output the noise, which is then added to the input image.

3) Digital corruptions include brightness, contrast, and saturation. These types actually may not be considered as corruptions, since they are natural characteristics of images with their intensity varies with daylight, lighting conditions, and the photographed object’s color. They are also commonly manipulated in image editing tools to make images more colorful and bright. For brightness, images are converted to HSV from RGB, and 0.3 is added to the last channel and converted back to RGB. For contrast, the mean of the image is removed, and the image is multiplied by 0.2 before the mean is added back. For saturation, images are again converted to HSV, and the first channel is multiplied by 2. Additionally, we look at the corruption caused by JPEG format, a commonly used lossy image compression format, and introduce compression artifacts. We save the images with JPEG quality of 15.

4) Weather corruptions include snow, spatter, frost, and fog. Among these, spatter and frost corruptions are modeling the effect of these weathers on the camera lenses. Spatter can occlude a lens in the form of rain or mud, and frost forms when lenses are coated with ice crystals. Implementations of these methods include many parameters, and we use the severity three for the given packages [33].
D. Cross Dataset

Cross-dataset evaluation, training a network on a source dataset and evaluating it on a different, target dataset, is a standard setup to evaluate domain adaptation methods. In such a setup, methods additionally use unlabeled images from the target dataset. However, we are not interested in adopting a method with an unlabeled target dataset but in finding the model that achieves the best score on the unseen target dataset without any retraining. For this, we use the Cityscapes [13] and BDD datasets [76] as our target datasets. Cityscapes consist of urban street scene images from various cities in Germany. This dataset is collected for autonomous driving applications with cameras mounted on a car as it drives through and therefore shows significant differences with COCO dataset [47], which is our source dataset. This dataset has annotations of eight instance classes: person, rider, car, truck, bus, train, motorcycle, and bicycle. Among these instance classes, COCO does not include riders. Given that riders are detected as persons by COCO labeling guidelines, we relabel a rider as a person in the ground truth and run the evaluation on seven classes. We evaluate the models both on the Cityscapes validation set (500 images) and training dataset (2975 images). The second dataset we use is the BDD dataset which includes images from New York, San Francisco, and Bay Area with diverse scene types such as city streets, residential areas, and highways. BDD has the same classes as the Cityscapes, and we follow the same setup for evaluation. We use the BDD validation dataset, which has 1000 images. We infer on Cityscape and BDD images on their original resolutions, 1024 \times 2048 and 720 \times 1280, respectively. The cross-dataset experiments also measure the robustness across geometric distortions such as varying scales, objects viewed from varying angles, scale transformations, and occlusions.

E. Joint-Task Training

We also investigate if joint-task training helps to improve the robustness of deep networks. We compare an object detection model and instance segmentation model in which addition to bounding boxes, instance masks are estimated. We compare these models with respect to their bounding box detection AP. Previously, it is shown that learning instance masks over bounding box detection provides better performance for detection showing benefits of joint-task training [31]. In this article, we study if such behavior exists for corrupted and out-of-domain image collections. The results of this setup are important as they may lead practitioners to add additional tasks into their training to increase robustness.

F. Evaluation Metrics

We report AP metric averaged over intersection over union (IoU) thresholds. In instance segmentation experiments, we report the AP evaluation using mask IoU. For joint-task experiments, we report the AP evaluation using bounding boxes. We calculate APs for each corrupted dataset. We additionally report the average of them (APs on each corrupted dataset) as given in the following equation:

\[ AP_{\text{avg}} = \frac{1}{C} \sum_{i=1}^{C} AP_i \]  

where \( i \) corresponds to various corruptions. We exclude the AP calculation on the clean validation dataset from this averaging. With this metric, we can observe which models are more robust to the set of corruptions mentioned before. We additionally report corruption degradation (CD) and relative CD (rCD) following other robustness works [33], [40] as given in the following equations:

\[ \text{CD}_i = \frac{1}{C} \sum_{i=1}^{C} \frac{D_i}{D_i^{\text{ref}}} \]  

\[ \text{rCD}_i = \frac{1}{C} \sum_{i=1}^{C} \frac{D_i^{\text{ref}} - D_i}{D_i^{\text{ref}}} \]  

The degradation in this context is defined as \( D = 1 - \text{AP} \), \( i \) corresponds to different corruptions, and \( f \) corresponds to different models. We consider the degradation of models relative to clean data by subtracting \( D_i^{\text{clean}} \) from \( D_i \). Additionally, we divide the CD by a reference model, ref. With CD, we measure the robustness of a model compared to the reference model. With rCD, we measure if a model gets less or more affected by the corruptions compared to their performance on clean images relative to the reference model.

IV. EXPERIMENTS

A. Augmentations

In our experiments, we use the released models from Detectron2 [71], and MMDection [10] frameworks and codebase provided by the authors of YOLACT [4]. These instance segmentation models are often trained with relatively weaker augmentation techniques compared to object recognition tasks. The models we use in our experiments are trained with only scale augmentation, in which images are resized by randomly sampled scales for accuracy improvements. On the other hand, image recognition tasks use a broader range of data augmentation techniques, such as random crop and various versions of color jitter. Therefore, the pretrained models on the ImageNet dataset are trained with these additional augmentations. In our analysis, we do not consider targeted augmentation techniques for any corruption and instead evaluate the models that are optimized to achieve the best validation score on clean datasets.

B. Training Parameters

The same parameters are used following the author’s released code. They are optimized to achieve the best validation scores by the authors. For the architecture studies, Mask-RCNN-based models (C4, DC5, FPN, FPN-Deform, and FPN-Cas), PointRend, and SOLO (SOLO and SOLOv2) models use the same setting. Specifically, models are trained on eight GPUs with 16 images per batch for 270000 steps (approximately 36 epochs). The base learning rate of 0.2 is
| Model Type | Clean | MC | CD | RCD | Motion | Defoc. | Gaus. | Impulse | Shot | Speckle | Bright. | Contr. | Satur. | JPEG | Snow | Spatter | Fog | Frost | GTL |
|------------|-------|----|----|-----|--------|-------|-------|---------|------|---------|---------|--------|--------|-------|------|--------|-----|-------|-----|
| C4         | 34.5  | 7.5 | 19.7 | 100 | 100  | 12.8  | 14.7  | 15.8  | 14.6 | 12.5    | 15.1   | 18.3   | 30.2  | 21.7  | 32.3   | 16.1 | 15.9   | 23.5| 25.9   | 17.2| 39.4 |
| DC5        | 35.8  | 8.3 | 20.8 | 99  | 102  | 13.2  | 15.1  | 16.0  | 15.0 | 13.3    | 15.8   | 18.9   | 31.6  | 22.9  | 33.9   | 18.2 | 16.6   | 24.6| 27.5   | 18.1| 30.9 |
| FPN        | 37.1  | 8.6 | 21.6 | 98  | 106  | 13.5  | 15.5  | 17.2  | 16.1 | 14.2    | 16.9   | 19.9   | 32.9  | 26.1  | 35.2   | 17.8 | 17.2   | 25.1| 28.8   | 18.8| 31.8 |
| FPN-Cat    | 38.4  | 9.1 | 22.4 | 97  | 109  | 13.7  | 16.0  | 18.0  | 17.1 | 14.9    | 18.0   | 21.0  | 33.9  | 25.2  | 36.4   | 18.7 | 18.3   | 25.9| 29.9   | 19.9| 32.7 |
| FPN-Deform.| 38.5  | 10.6| 23.5 | 94  | 102  | 16.1  | 18.0  | 19.5  | 18.0 | 16.0    | 18.7   | 22.0  | 34.8  | 25.9  | 36.6   | 19.9 | 19.1   | 26.4| 30.7   | 21.1| 33.2 |
| PointRend  | 38.3  | 8.9 | 22.4 | 97  | 108  | 13.9  | 15.7  | 17.6  | 17.0 | 15.2    | 17.7   | 20.9  | 34.0  | 25.1  | 36.3   | 18.3 | 18.0   | 26.3| 29.9   | 19.8| 32.8 |
| QueryInst  | 41.4  | 11.0| 25.8 | 92  | 105  | 15.9  | 18.6  | 21.4  | 20.7 | 18.7    | 21.0   | 24.1  | 37.6  | 29.4  | 39.6   | 23.5 | 20.0   | 29.1| 35.8   | 23.0| 36.5 |

**Table II:** Quantitative Results on COCO Validation Dataset With Images Corrupted by Various Techniques. Instance Segmentation Scores Are Presented. The First Column Provides Scores on the Original (Clean) Validation Dataset; The Second Column Provides the Results on the MC Dataset, the Third Presents the Average of All Scores on Corrupted Images (AVG.) With Single Corruptions. The Fourth and Fifth Columns Provide CD and RCD Scores, Respectively. RCD Scores Are Based on Reference Models Given in the First Row of Each Table, and the Rest of the Columns Show Scores on the Specified Corrupted Validation Sets. (a) Networks Trained With Different Network Architectures Using RSO Backbone. Multiple-Stage and Single-Stage Instance Segmentation Models Are Separated by a Horizontal Line. (b) Networks Trained With Different Backbones Using FPN Architecture. (c) Networks Trained With Different Normalization Methods, Pretrained Versus From Scratch Trainings. (d) Networks Trained From Scratch With and Without Strong Augmentations With Longer Trainings With SyncBN 50-FPN scr. Architecture Set-Up

---

used with a learning rate that is decreased by 0.1 at 21 000 and 25 000 iterations. Weight decay is set to 0.0001 and momentum to 0.9. QueryInst is also trained for 36 epochs. The initial learning rate is set to 2 × 10−5, divided by 10 at the 27th epoch and 33th epoch, respectively. TensorMask is trained for 540 000 steps with an initial learning rate of 0.0001, decreased by 10 at 480 000 and 520 000 iterations. PolarMask is trained for 12 epochs. YOLACT models (YOLACT, YOLACT++) are trained on a single GPU with batch size 8 for 800 000 iterations. The initial learning is set to 0.001 and decreased by 10 at iterations 280 000, 600 000, 700 000, and 750 000. Weight decay is set to 0.0005 and momentum to 0.9. For network backbones comparisons, ResNet and ResNeXt models are trained the same as the previously explained Mask-RCNN-based models. During inference, images are resized such that their scale (shorter edge) is 800 pixels. SwinTransformer models are trained with AdamW optimizer with a learning rate of 0.0001 betas of (0.9, 0.999), and weight decay of 0.05 for 36 epochs. For inference, images are again resized for their shorter edge to be 800 pixels. MViTV2-T is also trained for 36 epochs with a learning rate of 0.00016. SpineNet models are trained for 350 epochs from scratch with

---

**Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.**
SGD optimizer and a learning rate of 0.14, momentum of 0.9, and weight decay of 0.00004. The SpineNet-49 model infers on $640 \times 640$ image resolution whereas the SpineNet-96 model on 1024 and the SpineNet-143 model on 1280.

For the models that are not trained with copy-paste augmentation, we evaluate a converged model trained for 108 epochs. For the models that are trained with copy-paste augmentations, we compare the converged model trained for 400 epochs and the intermediate models that the checkpoints are obtained at epochs 100 and 200. Experiments that start from pretrained models are trained on eight GPUs with 16 images per batch for 270,000 steps which correspond to approximately 36 epochs on the COCO dataset, as mentioned previously. Models trained from scratch are trained for $3 \times$ more number of epochs. They are trained with 64 images per batch and 202,500 iterations with a base learning rate of 0.08. The learning rate is decreased by $10^2 \times$ iteration steps of 187,500 and 197,500.

V. Results

In this section, we share our results. First of all, we find that the mean AP of the methods under clean dataset and corrupted datasets show significant differences, as can be seen from Table II. Among the studied corruptions, the models are relatively robust on the digital type corruptions (contrast, brightness, and saturation) and geometric transformations, which preserve texture to a certain point. On others, performance is almost halved compared to the accuracy obtained on clean data. Additionally, models trained on the COCO dataset obtain lower scores on the Cityscapes and BDD datasets compared to models that are trained on Cityscapes and BDD datasets. Among the inspected models, the drop in accuracy is more for some of the models and is investigated in detail in this section.

A. Network Architectures

Results of different instance segmentation architectures with a fixed backbone of R50 are given in Table II(a) for various image corruptions and shown in Fig. 4 for Gaussian noise corruption. Improvements in clean images translate similarly but with a diminishing return to the corrupted images, as can be seen from rCD scores, which stay relatively constant and slightly degraded compared to the reference model on Mask-RCNN variants. Single-stage detectors, SOLO, and YOLACT are more robust to blur, noise, and weather corruptions. It can be seen from our results that deformable convolution both improves the clean AP and the average of the corrupted image scores when comparing FPN versus FPN-Deform and YOLACT versus YOLACT++ (the main change is the deformable convolutions over YOLACT). However, it does not affect the rCD score. Note that even though MC scores are usually in agreement with averaged corruption results, MC is significantly better for YOLACT++, even though on average QueryInst achieves a better score. That is because MC usually showcases the results of the worst case scenario. Even though QueryInst achieves better results on average, YOLACT++ achieves consistently better scores for both blur and noise corruptions. Therefore, when different corruptions are added together, the method that achieves better scores on the majority usually achieves better MC score.

On Cityscapes and BDD images, as given in Table III(a), the multistage models achieve significantly better scores than the single-stage instance segmentation models. One important difference between COCO images and these two datasets is that cross-dataset images are in much higher resolution, especially Cityscapes images have the size of $1024 \times 2048$. Network architectures are extensively tuned for different image resolutions for better accuracies. However, in our experiments, we use the same evaluation settings proposed for COCO and do not change any parameters for cross-dataset experiments except the input image resolution which we use the original input resolutions without scaling. With this setup, single-stage architectures achieve worse results than the multistage instance segmentation models except for TensorMask, which achieves similar scores with its two-stage counterparts. SOLO and SOLOv2 divide images into uniform grids, and if the center of an object falls into a grid cell, that grid cell is responsible for predicting the semantic category and masks of instances. Therefore, grids should be small enough to contain only one instance and Cityscapes and BDD images are very crowded. Even though we do not target tuning any parameters for different datasets, we tried with a few options of grid size for SOLO and SOLOv2 but did not achieve better results. It is possible to adapt the inference parameters to achieve better scores on cross-datasets. However, we are interested in using...
TABLE III
QUANTITATIVE RESULTS ON COCO, CITYSCAPES VALIDATION, AND TRAINING SETS, AND BDD VALIDATION SET ARE PRESENTED. RCD IS CALCULATED BY TAKING THE FIRST ROW MODEL AS A REFERENCE, AND THE COCO VALIDATION SCORE AS THE CLEAN SCORE.

(a) NETWORKS TRAINED WITH DIFFERENT NETWORK ARCHITECTURES WITH R50. MULTISTAGE AND SINGLE-STAGE INSTANCE SEGMENTATION MODELS ARE SEPARATED BY A HORIZONTAL LINE. (b) NETWORKS TRAINED WITH DIFFERENT NETWORK BACKBONES WITH FPN.

(c) NETWORKS TRAINED WITH DIFFERENT NORMALIZATION METHODS, PRE-TRAINED VERSUS FROM SCRATCH TRAININGS (SCR.).

(d) NETWORKS TRAINED FROM SCRATCH WITH AND WITHOUT STRONG AUGMENTATIONS WITH LONGER TRAININGS WITH SyncBN 50-FPN SCR. SET-UP

|       | COCO | City val | City train | BDD  | CD  | rCD |
|-------|------|----------|------------|------|-----|-----|
| C4    | 34.3 | 26.0     | 25.3       | 19.4 | 100 | 100 |
| DC5   | 35.8 | 27.8     | 26.5       | 21.5 | 96  | 99  |
| FPN   | 37.1 | 26.9     | 25.8       | 21.8 | 98  | 117 |
| FPN-Cas | 38.4 | 27.9     | 27.6       | 24.8 | 96  | 113 |
| FPN-Deform | 38.5 | 27.2     | 26.1       | 23.0 | 98  | 131 |
| PointRend | 38.3 | 28.3     | 27.1       | 24.0 | 96  | 114 |
| QueryInst | 41.4 | 29.5     | 27.0       | 23.8 | 96  | 140 |

|       | COCO | City val | City train | BDD  | CD  | rCD |
|-------|------|----------|------------|------|-----|-----|
| R50   | 37.1 | 26.9     | 25.8       | 21.8 | 100 | 100 |
| R101  | 39.5 | 28.5     | 27.7       | 22.6 | 97  | 100 |
| X101  | 39.5 | 29.6     | 29.1       | 23.0 | 97  | 99  |
| S49   | 37.8 | 29.8     | 28.8       | 24.9 | 96  | 81  |
| S96   | 41.2 | 32.5     | 31.4       | 25.5 | 93  | 91  |
| S143  | 41.3 | 31.7     | 30.2       | 24.8 | 95  | 100 |
| SwinT | 41.7 | 30.1     | 29.1       | 23.3 | 96  | 92  |
| SwinS | 43.2 | 33.0     | 31.8       | 25.2 | 93  | 106 |
| MViTv2-T | 43.8 | 30.1     | 29.6       | 25.2 | 95  | 127 |

|       | COCO | City val | City train | BDD  | CD  | rCD |
|-------|------|----------|------------|------|-----|-----|
| R50-FPN     | 37.1 | 26.9     | 25.8       | 21.8 | 100 | 100 |
| GN R50-FPN  | 38.6 | 27.8     | 26.7       | 22.4 | 99  | 106 |
| SyncBN R50-FPN | 37.8 | 28.0     | 27.4       | 24.5 | 96  | 92  |
| GN R50-FPN set  | 39.6 | 29.8     | 28.6       | 24.0 | 96  | 94  |
| SyncBN R50-FPN set | 39.3 | 30.2     | 29.1       | 24.0 | 96  | 92  |

|       | COCO | City val | City train | BDD  | CD  | rCD |
|-------|------|----------|------------|------|-----|-----|
| Baseline - 100 epoch | 59.3 | 29.2     | 29.1       | 26.0 | 100 | 100 |
| Aug - 100 epoch    | 40.3 | 30.7     | 29.9       | 25.3 | 99  | 103 |
| Aug - 200 epoch    | 41.7 | 31.6     | 30.8       | 27.7 | 97  | 104 |
| Aug - 400 epoch    | 42.5 | 32.7     | 31.8       | 28.5 | 96  | 102 |

Fig. 5. Rows present results of R50-C4, R50-DC5, and R50-FPN Cascade models, respectively. Models are trained on the COCO dataset and inferred on Cityscapes images. C4 architecture outputs false detections of persons, and cars are misclassified as trucks. DC5 does not output false detections but misses true detections of cars and bicycles. Among these models, the FPN Cascade model detects all bicycles and the majority of the cars.

that the network backbones affect robustness significantly. On clean data, ResNet101 performs better than ResNet50, and ResNeXt101 achieves the best score among these three. On the other hand, this ranking is not preserved for the corrupted validation datasets. On blur and noise corruptions, ResNet101 shows more robustness compared to ResNeXt101. In contrast, the ResNeXt101 backbone results in better accuracies on digital and weather-type corruptions. SpineNet, Swin, and MViTv2-T architectures achieve significantly higher robustness to image corruptions and achieve very robust rCD scores as well as impressive clean AP scores. These architectures have powerful representation capacities. Swin and MViTv2-T architectures have global receptive fields due to their attention mechanism, whereas SpineNet architecture, due to its scale permuted architecture, has a very large receptive field. We believe their global receptive field contributes to their robustness to image corruptions. Visual results of this comparison are shown in Fig. 6. On cross-dataset evaluations, as given in Table III(b), model performances on COCO, Cityscapes, and BDD shows positive correlations. Original SpineNet inference set-ups process images on different resolutions as mentioned in Section II-C2. SpineNet-49 infers on 640 × 640 images, whereas the image size gradually increases for SpineNet-96 and SpineNet-143. If we use that original set-up when inferring on Cityscapes training images, methods achieve 18.1, 23.8, and 25.3 scores, respectively. However, when we do not scale Cityscapes images, and run the inference on the original resolution, the scores increase to 21.8, 31.4, and 30.2, respectively. S96 achieves slightly better
Fig. 6. Rows present results of R101-FPN, X101-FPN, and S96-FPN models, respectively. On clean images, models output comparable results. On defocus blur image, R101 and S96 can detect the highly occluded person (behind white), whereas X101 fails to detect. Furthermore, S96-FPN impressively detects the backpack and the ski on the challenging defocus blur image.

results than S143 when it does not have the shortcoming of processing images in a lower resolution than S143. We observe that, with an exception of S143, the larger models come with better Cityscapes and BDD scores. For example, going from SwinT to SwinS, the number of parameters increases from 48 to 96 M. While SwinT achieves only 1.5 AP better on the COCO validation set compared to SwinS, it achieves a 2.7 AP better result on the Cityscapes training dataset, which has a much higher resolution. MViTV2-T, on the other hand, has 44 M parameters similar to SwinT and has improved architecture when measured on the COCO validation set. However, its performance on the Cityscapes dataset is only slightly better than SwinT and worse than SwinS, which has a larger capacity.

C. ImageNet Pretrained Versus From Scratch

Recently, it has been shown that ImageNet pretraining does not improve accuracy on the COCO dataset, and even better accuracies can be obtained with training from scratch at the expense of longer training iterations [30]. Another interesting question is if ImageNet pretraining improves the robustness across different image corruptions. During pretraining, the networks learn from an additional million images, which hypothetically can improve the robustness of learned features. Results of this comparison are shown in Table II(c). Surprisingly, the models trained from scratch achieve higher accuracy on clean data and show significantly better robustness across different image corruptions except for JPEG corruption. ImageNet pretrained models achieve significantly higher accuracies than their trained from scratch counterparts in JPEG corruption (22.65 versus 19.89 for GN and 19.33 versus 14.83 for SyncBN models). On the other hand, among all other corruption types, models trained from scratch achieve higher robustness. The same also holds on the Cityscapes and BDD evaluation, where we observe that models trained from scratch achieve higher accuracy results as given in Table III(c) on all sets.

D. Normalization Layers

Our results presented in Table II(c) show that across normalization layers that are employed in instance segmentation models, GN provides increased robustness to networks under image corruptions. This increase in robustness holds for both models trained from ImageNet pretrained networks and randomly initialized networks. Interestingly, the improvements from GN compared to SyncBN are pretty significant for corrupted images than the clean images. GN R50-FPN and SynchBN R50-FPN from scratch models perform closely on the clean datasets, 39.62 and 39.26, respectively. On the other hand, looking at the corrupted image results on average, accuracy is 26.3 and 22.8, respectively, showing a large margin of improvements as can be measured by rCD scores. Example output results are shown in Fig. 7.

Results on Cityscapes and BDD images are quite interesting. SyncBN models achieve significantly better results than GN models, as shown in Table III(c) showing an opposite behavior than the performance on corrupted images. Even though GN-based models achieve a higher score in the COCO validation set, SyncBN-based models achieve higher scores on Cityscapes training datasets and BDD dataset. Note that there is a disagreement in the ranking of models between Cityscapes validation and training datasets, and we find the training dataset results to be more reliable given the larger number of samples it contains (2975 training images versus 500 validation images).

This is a very interesting result. Adaptive normalization layers and domain-specific BN layers are extensively studied.
TABLE IV
QUANTITATIVE RESULTS ON COCO VALIDATION DATASET WITH IMAGES CORRUPTED BY VARIOUS TECHNIQUES. BOUNDING BOX DETECTION SCORES OF NETWORKS Trained WITH AND WITHOUT INSTANCE SEGMENTATION HEADS. THE FIRST COLUMN PROVIDES SCORES ON THE ORIGINAL (CLEAN) VALIDATION DATASET; THE SECOND COLUMN PROVIDES THE RESULTS ON THE MC DATASET, THIRD PRESENTS THE AVERAGE OF ALL SCORES ON CORRUPTED IMAGES (AVG.) WITH SINGLE CORRUPTIONS. THE FOURTH AND FIFTH COLUMNS PROVIDE CD AND rCD SCORES, RESPECTIVELY. rCD SCORES ARE BASED ON REFERENCE MODELS GIVEN IN THE FIRST ROW, AND THE REST OF THE COLUMNS SHOW SCORES ON THE SPECIFIED CORRUPTED VALIDATION SETS.

|               | R50-FPN | Clean | MC | Avg | CD | rCD | Motion | Defocus | Gauss | Guass | Impulse | Shot | Speckle | Brightness | Contrast | Saturation | JPEG | Snow | Spatter | Fog | Post | GTx |
|---------------|---------|-------|----|-----|----|-----|--------|---------|-------|-------|---------|------|--------|------------|----------|-------------|-------|-------|---------|-----|-------|-----|
| without Mask  | 40.2    | 9.1   | 23.2| 100 | 100| 15.0| 17.1   | 18.6   | 16.8  | 14.0  | 17.4  | 21.3 | 35.8   | 26.0       | 37.9     | 19.2       | 18.6  | 27.2  | 31.0    | 20.8| 34.7   |
| with Mask     | 41.0    | 9.8   | 24.2| 99  | 99 | 15.9| 17.9   | 19.4   | 18.1  | 15.9  | 18.9  | 22.5 | 36.5   | 26.9       | 38.8     | 20.1       | 19.4  | 27.9  | 31.9    | 21.1| 35.6   |

Fig. 8. Examples from BDD dataset together with the predictions from Aug. 400 network. BDD dataset includes diverse images as shown here from day time to night time to different weather conditions and even photographs in tunnels.

for domain adaptation problems [9], [24], [67]. In this work, we are not interested in adapting one network to a target domain but interested in finding the most reliable network without additional tuning. Both corruptions and cross-datasets change the statistics of the datasets. Corruptions change pixel-wise image statistics and cross-datasets change the content-based object statistics. While corruptions do not change the content, on cross-datasets, objects appear in different densities and sizes. GN is better at making the network more robust to corruptions added onto images by removing their effects through normalization. On the other hand, SyncBN is better at aligning the statistics of cross-datasets.

E. Copy-Paste Augmentation

It has been shown that copy-paste augmentation combined with longer training significantly improves performance on clean validation sets [29]. In Table II(d), we compare a baseline that is not trained with copy-paste augmentation for 108 epochs and networks trained with copy-paste augmentations for 100, 200, and 400 epochs. Note that the baseline trained for 108 epochs and the network trained with copy-paste augmentations for 400 epochs are the ones that converged (the validation set performance no longer improves). Our results show that copy-paste augmentation significantly improves the robustness, especially early in training (100 epoch results). Results on the cross-dataset experiments are mostly in agreement with the corruption scores, as shown in Table III(d). Except for the slight decline in BDD scores from baseline to Aug 100 model, all scores improve consistently as the COCO score increases. We provide visual predictions for Aug 400 model on BDD images in Fig. 8.

F. Effect of Joint-Task Training

The results of this experiment are given in Tables IV and V. Both rows in tables share the same setup except that the model reported in the first rows is trained only to detect bounding boxes. In contrast, the second-row model learns to predict instance segmentation in addition to detecting bounding boxes. They both use the ResNet50-FPN architecture. We compare the bounding box detection APs of these two models differing in terms of their prediction heads. We find that the joint-task training improves the scores on the clean images and corrupted ones even more so than the improvement on clean images. While the AP improvement coming from joint-task training is 0.77 on clean images, it is 1.00 on corrupted images on average and results in an rCD score of 99. Especially on noise corruptions, the improvements are pretty significant. The results are similar on cross-dataset experiments as shown in Table V. Even though without a mask slightly outputs better than with a mask on the Cityscapes training dataset, on average, we observe better generalization of the method with joint-task training.

VI. DISCUSSION

Researchers switch their attention to the robustness of deep models as they now achieve high-performance results on clean/in-domain validation sets. Previously, the object recognition [33], and semantic segmentation [40] tasks are rigorously evaluated for image corruptions. We are the first to conduct a similar study on the instance segmentation task to the best of our knowledge. In addition to evaluating models under image corruptions, we also evaluate on images collected with a different set-up and from different environments than
the training images; in other words, the images that show the domain gap with the training images. We believe this is a very important addition to our study, which was not studied before among the previous robustness papers [33], [40]. One may argue that cross-dataset evaluations measure how reliable models are in a more realistic setting. For models to be useful in real-world, they should be able to generalize well to different image resolutions and different sizes of objects and density of them in a scene. Our evaluation is also not limited to network architectures and backbones but extended to models with different normalizations, augmentations, initializations, and also investigates the impact of joint-task training.

Through this study, we gain several insights. In Fig. 9(a), we compare different multistage network architectures and see that networks that achieve better clean AP scores also achieve better CD scores but with incremental returns, as can be measured by rCD. On the other hand, on single-stage networks, as shown in Fig. 9(b), YOLACT variants achieve significantly more robust results. For these novel architectures except YOLACT, it seems that architectures have slowly and consistently improved their representations over time. However, their corruption robustness improvements (CD) are explained mainly by better-learned representations on clean images as their rCD scores are larger than 100%. On the other hand, YOLACT and YOLACT++ show a very different behavior. While they have lower clean AP scores, they achieve significantly better scores on corrupted images and an impressive rCD score, as can be seen in Fig. 9(b). This behavior may be coming from its single-stage architecture.

In our backbone studies, Fig. 9(c), we find that deeper models and models with large receptive fields generally show higher robustness and generalization. The most significant results are from SpineNet and Swin backbones, which achieve below 80% rCD scores on corrupted images. Especially S49 achieves the best rCD score both on corrupted and cross-dataset images, 75% and 81%, respectively. The underlined difference regarding SpineNet architectures is an important finding, as previous benchmarking studies state that rCD scores are relatively constant among different network backbones [33], [40]. Those studies did not include SpineNet architecture [17], which seems to have an improved architectural design for robustness. Transformer-based architectures are similar, they also show improved architectural design for robustness and generalization.

We find that normalization layers affect robustness significantly. As shown in Fig. 9(d), the better clean AP score of normalization layers correlates with a better CD. Seeing such a positive correlation between clean AP and CD is expected, and the same observations are made for semantic segmentation tasks [40]. The more interesting is the rCD score, which is reported to stay relatively constant in previous studies [33], [40]. In Fig. 9(d), we see that GN achieves a very robust rCD score. The model shows larger improvements in robustness than it does for the clean validation set. On the other hand, in cross-dataset experiments, we see that SyncBN achieves better results. This opens a new research direction for normalization layers. When new normalization methods are proposed, it is best to evaluate them on clean validation sets as well as on corrupted sets and different domain images. The trend so far is to either test a proposed normalization layer on clean validation datasets [36], [66] or in domain adaptation setting on target datasets [9], [24] but not altogether, which we hope our work will inspire to change. We also compare models trained from pretrained networks and randomly initialized networks and observe that pretraining does not improve the robustness. As a result, we observe that models trained from scratch for longer iterations achieve higher robustness than pretrained models.

VII. CONCLUSION

This article presented a comprehensive evaluation of instance segmentation models with respect to real-world image corruptions and images that exhibit domain gaps with the training datasets the models are learned from. We investigated different model ingredients in terms of their robustness, such as their architectural designs, network backbones, initializations, normalization modules, and the effect of joint-task training. With these analyses, we provide insights into the
robustness of state-of-the-art instance segmentation models. These findings are essential for researchers when designing their models for applications where image corruptions are known in advance and when picking a trained off-the-shelf model to infer on the dataset at hand.

REFERENCES

[1] S. Fahri Altindis, A. Meric, Y. Dalva, U. Gudukbay, and A. Dundar, “Refining 3D human texture estimation from a single image,” 2023, arXiv:2303.03471.
[2] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” 2016, arXiv:1607.06450.
[3] A. Bhattad, A. Dundar, G. Liu, A. Tao, and B. Catanzaro, “View generalization for single image textured 3D models,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 6077–6086.
[4] D. Bolya, C. Zhou, F. Xiao, and J. Y. Lee, “YOLACT: Real-time instance segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 9157–9166.
[5] D. Bolya, C. Zhou, F. Xiao, and J. Y. Lee, “YOLACT++ better real-time instance segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 2, pp. 1108–1121, Feb. 2022.
[6] Z. Cai and N. Vasconcelos, “Cascade R-CNN: Delving into high quality object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6154–6162.
[7] Z. Cai and N. Vasconcelos, “Cascade R-CNN: High quality object detection and instance segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 5, pp. 1483–1498, May 2021.
[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 213–229.
[9] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han, “Domain-specific batch normalization for unsupervised domain adaptation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 7354–7362.
[10] K. Chen et al., “MMDetection: Open MMLab detection toolbox and benchmark,” 2019, arXiv:1906.07155.
[11] C.-L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder–decoder with atrous separable convolution for semantic image segmentation,” 2018, arXiv:1802.02611.
[12] X. Chen, R. Girshick, K. He, and P. Dollar, “TensorMask: A foundation for dense object segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 2061–2069.
[13] M. Cordts et al., “The cityscapes dataset for semantic urban scene understanding,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 3213–3223.
[14] J. Dai et al., “Deformable convolutional networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 764–773.
[15] B. Dong, F. Zeng, T. Wang, X. Zhang, and Y. Wei, “SOLQ: Segmenting objects by learning queries,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 21898–21909.
[16] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” 2020, arXiv:2010.11929.
[17] X. Du et al., “SphineNet: Learning scale-permutated backbone for recognition and localization,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 11589–11598.
[18] A. Dundar, J. Gao, A. Tao, and B. Catanzaro, “Fine detailed texture learning for 3D meshes with generative models,” 2022, arXiv:2203.09362.
[19] A. Dundar, J. Jin, B. Martini, and E. Culurciello, “Embedded streaming deep neural networks accelerator with applications,” IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 7, pp. 1572–1583, Jul. 2017.
[20] A. Dundar, M.-Y. Liu, Z. Yu, T.-C. Wang, J. Zedlewski, and J. Kautz, “Domain stylization: A fast covariance matching framework towards domain adaptation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 7, pp. 2360–2372, Jul. 2021.
[21] A. Dundar, K. Sapra, G. Liu, A. Tao, and B. Catanzaro, “Panoptic-based image synthesis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 8067–8076.
[46] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jul. 2017, pp. 2117–2125.

[47] T. Lin et al., “Microsoft COCO: Common objects in context,” in Proc. Eur. Conf. Comput. Vis., 2014, pp. 740–755.

[48] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8759–8768.

[49] W. Liu et al., "S3D: Single shot multibox detector," in Proc. Eur. Conf. Comput. Vis., Oct. 2016, pp. 21–37.

[50] Y. Liu et al., "A survey of visual transformers," IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 30, 2023, doi: 10.1109/TNNLS.2022.3227717.

[51] Y. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 10012–10022.

[52] M. Long and J. Wang, "Learning transferable features with deep adaptation networks," in Proc. Int. Conf. Mach. Learning (ICML), 2015, pp. 97–105.

[53] M. Mardani, G. Liu, A. Dundar, S. Liu, A. Tao, and B. Catanzaro, "Neural FFTs for universal texture image synthesis," in Proc. Adv. Neural Inf. Process. Syst., 2020.

[54] C. Michaelis et al., “Benchmarking robustness in object detection: Autonomous driving when winter is coming,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[55] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 779–788.

[56] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, arXiv:1506.01497.

[57] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2010, pp. 213–226.

[58] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2242–2251.

[59] B. Sun and K. Saenko, “Deep coral: Correlation alignment for deep domain adaptation," in Proc. Eur. Conf. Comput. Vis. (ECCV) Workshop, 2016, pp. 443–450.

[60] Y. Taigman, A. Polylie, and L. Wolf, "Unsupervised cross-domain image generation," in Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[61] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Sep. 2017, pp. 23–30.

[62] J. Tremblay et al., “Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2017, pp. 1492–1500.

[63] A. Burak Yildirim, V. Baday, E. Erdem, A. Erdem, and A. Dundar, "Instainpaint: Instructing to remove objects with diffusion models," 2023, arXiv:2304.03246.

[64] F. Yu et al., "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 2636–2645.

[65] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions," in Proc. Int. Conf. Learn. Represent. (ICLR), 2016.

[66] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, “Object detection with deep learning: A review," IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[67] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, “Improving the robustness of deep neural networks via stability training," in Proc. CVPR, Jun. 2016, pp. 4480–4488.

[68] X. Zhu, H. Hu, S. Lin, and J. Dai, “Deformable ConvNets v2: More deformable, better results," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 9308–9316.

[69] B. Zoph et al., “Rethinking pre-training and self-training," in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 3833–3845.

Yusuf Dalva received the B.Sc. and M.Sc. degrees in computer science from Bilkent University, Ankara, Turkey, in 2020 and 2023, respectively. He is currently pursuing the Ph.D. degree with Virginia Tech, Blacksburg, VA, USA.

His research interests include image understanding and feature manipulation, generative models, and robustness enhancement of deep neural networks.

Hamza Pehlivan received the B.Sc. and M.Sc. degrees in computer science from Bilkent University, Ankara, Turkey, in 2021 and 2023, respectively. He is currently pursuing the Ph.D. degree with MPI, Saarbrücken, Germany.

His research interests include generative models, image inpainting, and understanding of deep networks.

Said Fahri Altundış received the B.Sc. degree in computer science from TED University, Ankara, Turkey, in 2020. He is currently pursuing the M.Sc. degree with Bilkent University, Ankara.

He is a Software Engineer with HAVELSAN, Ankara. His research interests include domain adaptation, instance segmentation, and generative models.

Aysegul Dundar received the B.Sc. degree in electrical and electronics engineering from Boğaziçi University, Istanbul, Turkey, in 2011, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 2016.

She is an Assistant Professor of computer science with Bilkent University, Ankara, Turkey.

Dr. Dundar won the first place in the Domain Adaptation for Semantic Segmentation Competition in the Workshop on Autonomous Vehicle Challenge, in CVPR 2018.