Semantic-Syntactic Discrepancy in Images (SSDI): Learning Meaning and Order of Features from Natural Images

Anonymous authors
Paper under double-blind review

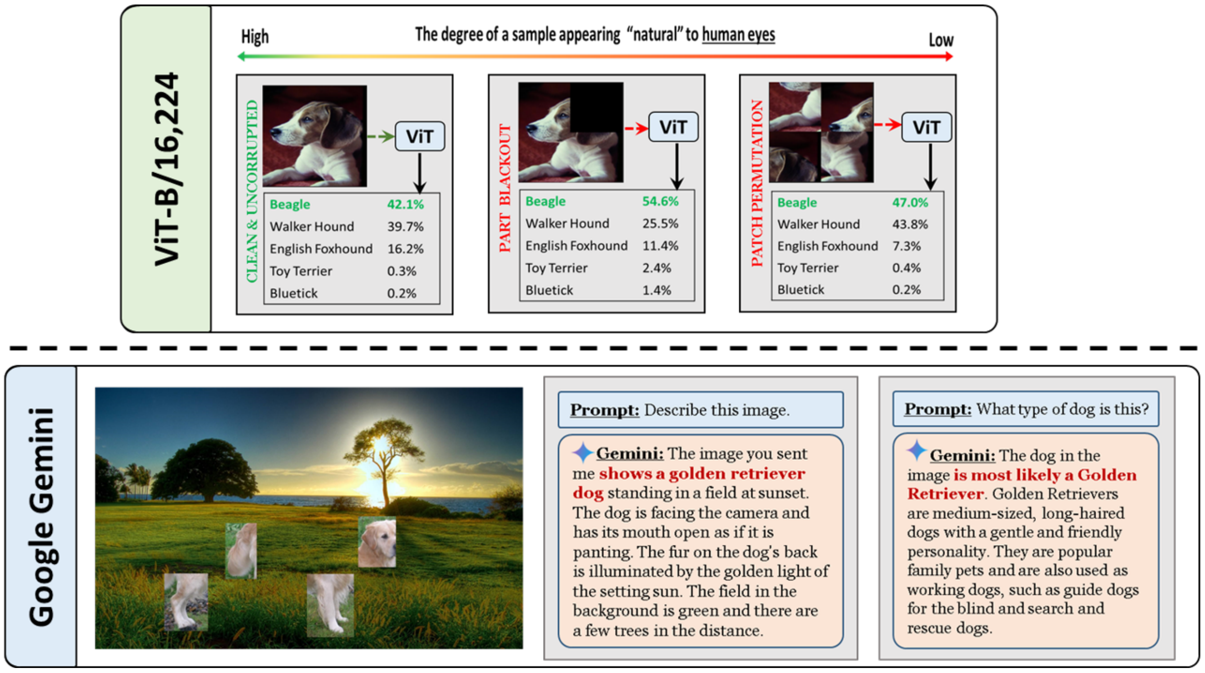
Abstract

Despite considerable progress in image classification tasks, classification models seem unaffected by the images that significantly deviate from those that appear natural to human eyes. Specifically, while human perception can easily identify abnormal appearances or compositions in images, classification models overlook any alterations in the arrangement of object parts as long as they are present in any order, even if unnatural. Hence, this work exposes the vulnerability of having semantic and syntactic discrepancy in images (SSDI) in the form of corruptions that remove or shuffle image patches or present images in the form of puzzles. To address this vulnerability, we propose the concept of "image grammar", comprising "image semantics" and "image syntax". Image semantics pertains to the interpretation of parts or patches within an image, whereas image syntax refers to the arrangement of these parts to form a coherent object. We present a semi-supervised two-stage method for learning the image grammar of visual elements and environments solely from natural images. While the first stage learns the semantic meaning of individual object parts, the second stage learns how their relative arrangement constitutes an entire object. The efficacy of the proposed approach is then demonstrated by achieving SSDI detection rates ranging from 70% to 90% on corruptions generated from CelebA and SUN-RGBD datasets.

1 Introduction

The task of image classification has significantly evolved with the advancements in deep neural networks (DNNs), to the point of achieving performance comparable with humans. For such tasks, the state-of-the-art (SoTA) models are based on convolution neural networks (CNNs) (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; He et al., 2016) and vision transformers (ViTs) (Dosovitskiy et al., 2021; Liu et al., 2021). The fundamental idea behind these models was to develop the ability to identify an object in an image by understanding (and recognizing) its features/attributes (semantics). For example, when an image is classified as a "dog", it is because it contains attributes of dogs, such as "fur", "dog tail", "dog paws". Indeed, such models learn the distribution of images belonging to different object classes by extracting low- and high-level semantic information from images and encoding them in hidden features of stacked layers (Hua et al., 2018; Ortego et al., 2021). The learning processes are designed to mainly rely on image-level labels: the global information about the object type depicted in the entire image. Hence, to learn object classes, the classification models **have to implicitly learn** class features/attributes present in the images labeled as the corresponding class. In other words, since there is no explicit information provided for each feature like "fur", "paws", or "dog tail", the models must implicitly learn these attributes as characteristic traits of a dog by observing a large variety of "dog" images. However, this learning process has an underlying assumption that causes a vulnerability in the pipeline of classification models.

The vulnerability arises from assuming that ***input images always accurately depict the object in its natural appearance and composition***. As a result, classification models don't explicitly learn how the constituent features of an object class are arranged in natural images. For instance, they do not ensure that the "dog tail" is at the posterior of the dog, or that the "paws" are in the lower part of the dog's body.



CLEAN & UNCORRUPTED

Beagle	42.1%
Walker Hound	39.7%
English Foxhound	16.2%
Toy Terrier	0.3%
Bluetick	0.2%

PART BLACKOUT

Beagle	54.6%
Walker Hound	25.5%
English Foxhound	11.4%
Toy Terrier	2.4%
Bluetick	1.4%

PATCH PERMUTATION

Beagle	47.0%
Walker Hound	43.8%
English Foxhound	7.3%
Toy Terrier	0.4%
Bluetick	0.2%

Prompt: Describe this image.

Gemini: The image you sent me **shows a golden retriever dog** standing in a field at sunset. The dog is facing the camera and has its mouth open as if it is panting. The fur on the dog's back is illuminated by the golden light of the setting sun. The field in the background is green and there are a few trees in the distance.

Prompt: What type of dog is this?

Gemini: The dog in the image **is most likely a Golden Retriever**. Golden Retrievers are medium-sized, long-haired dogs with a gentle and friendly personality. They are popular family pets and are also used as working dogs, such as guide dogs for the blind and search and rescue dogs.

Figure 1: Examples illustrating susceptibility of various models (ViT-B/16,224 and Google Gemini Advanced) to images with unnatural appearance. The output of both models seem **to be completely unaffected by any of abnormalities** present in the input images when performing the given tasks.

In other words, the question being answered is "*What constituent class features are present in the image?*", and not "*Does this image contain a meaningful object of any observed class?*". This vulnerability appears as **the discrepancy** between a human's ability to **instinctively** recognize natural images **and** a classification model's tendency to be easily **fooled** into making high-confidence predictions for unnatural images. Figure 1 illustrates how a vision transformer model (ViT-B/16,224) (Wu et al., 2020) and a large multimodal model (Google Gemini Advanced) (Team et al., 2023) are susceptible to this type of vulnerability. It can be seen that the ViT model consistently predicts various dog breeds as the top-5 predictions, disregarding the appearance of images that increasingly diverge from the clean natural image. Similarly, the Gemini model ignores the unnaturalness of a dog displayed in the image when asked to describe the image or identify the type of the dog. We use the term **Semantic-Syntactic Discrepancy in Images (SSDI)** to refer to this vulnerability.

To address this vulnerability to SSDI corruptions in the visual appearance of input samples, we propose the concept of "***image grammar***". The concept is akin to grammar in language (Gunter et al., 1997; von Stechow, 2019), and comprises both semantic meaning ("***image semantics***") and a syntactical structure ("***image syntax***"). "*Image semantics*" pertains to the existence and semantic significance of individual features defining an object, while "*image syntax*" concerns the spatial arrangement and correct placement of features to depict an object as it would naturally appear in the real world.

Consequently, we propose a semi-supervised two-stage ***deep learning framework*** that successively learns both "*image semantics*" and "*image syntax*" mentioned above. This framework combines a deep clustering method with a bi-directional LSTM. The deep clustering method treats the image as a set of semantic features, while the bi-directional LSTM ensures that the features are learned in relation to their spatial neighbors, enforcing the model to understand individual features and their natural arrangement and composition. This approach aims to train a classification model to not only predict object classes but also assess the degree of naturalness in the appearance of an image. Furthermore, the framework is specifically designed to learn the concept of "*image grammar*" in a strict setting that assumes SSDI corruptions are neither generated nor used during the training phase. Such setting is motivated by the idea that, similar to human capabilities, the meaning and the order of features should be inherently learnt together solely from natural images.

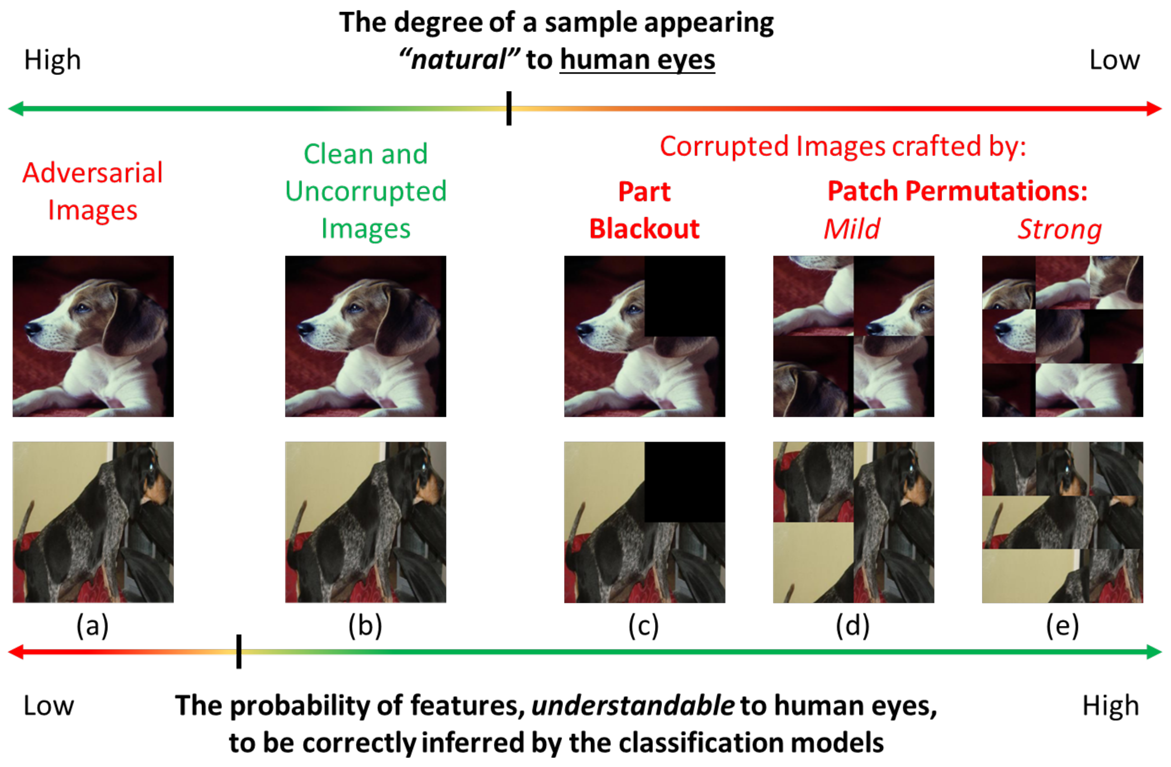

The degree of a sample appearing
"natural" to human eyes
High
Low
Adversarial Images
Clean and Uncorrupted Images
Corrupted Images crafted by:
Part Blackout
Patch Permutations:
Mild
Strong
(a)
(b)
(c)
(d)
(e)
Low
The probability of features, *understandable* to human eyes, to be correctly inferred by the classification models
High

Figure 2: Examples explaining the vulnerability of classification models to semantic-syntactic discrepancies in images, SSDI. While all images appear to display different dogs, the visual appearance and composition of images in columns (c)-(e) deviate from clean and uncorrupted images shown in column (b). This discrepancy, however, is not caught by the current classification models, which solely focus on the presence of features. This can be contrasted with adversarial examples (a), which target the decision-making of models while preserving the natural visual appearance. Note: adversarial examples are not considered in this study but are used to paint a complete picture.

2 Semantic-Syntactic Discrepancy in Images (SSDI)

Problem Definition Semantic-syntactic discrepancy (SSDI) is evident in images that have many identifiable semantic features but look unnatural to humans in their appearance and composition. Figure 2 illustrates examples that expose the described vulnerability of classification models. As humans, we can readily recognize that the shown images aim to represent dogs due to the presence of recognizable dog features. We can also perceive that the images in the three rightmost columns ((c)-(e)) are more likely to be unnatural or altered in appearance (compared to images in columns (a) and (b)). We can make such a distinction even, for example, a "puzzle" image with rearranged patches that we haven't encountered before. In contrast, a classification model trained on large-scale dataset like ImageNet (Deng et al., 2009) consistently assigns output labels corresponding to various dog breeds for the images in columns (b)-(e) solely based on the presence of identifying semantic features, disregarding any abnormalities in the overall appearance (the syntax). The contrast between human perception and classification models in these examples underscores ***three crucial properties*** needed to address this vulnerability: (1) classification models need to learn to estimate the degree to which an image depicts natural occurrence existing in the real world, (2) based on this degree, they need to be able to distinguish between natural and unnatural images, and (3) preferably, they need to be capable of learning this based solely on natural images.

Learning Setting This work proposes a way that enables classification models with properties (1) and (2) in a learning setting motivated by the property (3) mentioned above. In particular, the method describes how the architecture of a classification model and its training can be modified to allow it to distinguish between natural and unnatural images by only observing natural unaltered images during training. This setting is based on the combination of two reasons. First, humans can instinctively recognize unnaturalness of different kinds even when they never came across of such examples. Both the meaning and the order of features are learnt together without any need to see images with some parts removed or shuffled. Second, it is challenging to consider and to train on all the possible ways samples with semantic-syntactic discrepancy can be crafted (including, but not limited to corruptions described in this work). For example, if an image is divided into $n \times n$ patches, assuming that only the original image depicts meaningful object, the number of unnatural samples becomes $(n^2! - 1)$. Not only this becomes intractable for values of $n > 3$, but also has no direct way of ensuring that all of these samples do not depict meaningful objects. Hence, it is more intuitive to teach "*What the natural arrangement of features should look like?*" rather than "*How natural and unnatural images differ?*".

Importance and Application The importance of addressing SSDI becomes prominent when considering the classification models used for the downstream tasks. On one hand, there's the assumption that input images are curated or verified, and the model's primary task is to determine the object's class within the image. In this case, it's been demonstrated that DNN models are vulnerable to notorious adversarial images, which undermines their ability to accurately infer learned features while looking visually indistinguishable from natural images (as seen in column (a) of Figure 2). (Note: adversarial attacks are not considered within the scope of this paper, but are mentioned to paint a complete picture.) On the other hand, some applications use object class prediction as a means to determine if an image meets certain criteria. For instance, in image retrieval applications, the goal might be to compile a set of images belonging to specific classes. Here, correctly inferring features is crucial, *but the image itself is the primary interest.* In such cases, the SSDI vulnerability discussed in this paper could have a significantly detrimental effect, leading to the final collection of images to include unnatural, corrupted images. This is especially prominent today when the availability of generative models enables simple ways of handcrafting unnatural examples.

3 Related Works

To the best of the authors' knowledge, the problem of SSDI vulnerability is not well-established yet. This makes it hard to contextualize it as a standalone research direction. Nevertheless, the issue of SSDI can be considered in the context of two categories of works: from the perspective of learning compositionality of visual representations and from the perspective of vulnerabilities in computer vision.

3.1 Compositionality of Image Representations

Learning image representations creates a basis for visual tasks, such as image classification (He et al., 2016) and object detection (He et al., 2017; Lin et al., 2017). Despite various supervised (Krizhevsky et al., 2012) and unsupervised/self-supervised (He et al., 2022) methods were proposed to learn the underlying distribution of image features, the issue of relying purely on the presence of features is recently starting to become more prominent (Thrush et al., 2022). In the work of Yuksekgonul et al. (2023), authors discuss and analyze the effects of naive reliance on contrastive pre-training and accuracy metrics. They conclude that large vision and language models, which currently serve as the basis for a wide range of applications, disregard composition, i.e. underlying structure, of both visual and language inputs. Their findings support the ideas and highlight the relevance of SSDI vulnerability discussed in this paper. The work of Qin et al. (2022) proposed a solution for ViT-based models (Wu et al., 2020) through patch-based negative augmentation. From this perspective, our work serves as an alternative approach, which proposes a way of learning compositionality along with the semantic of features *without the need for generation and reliance on negative samples* (as described in Learning Setting of Section 2). Even though our experiments focus on CNN-based models, it is possible to extend the method to vision transformers by considering them as the feature extractor model.

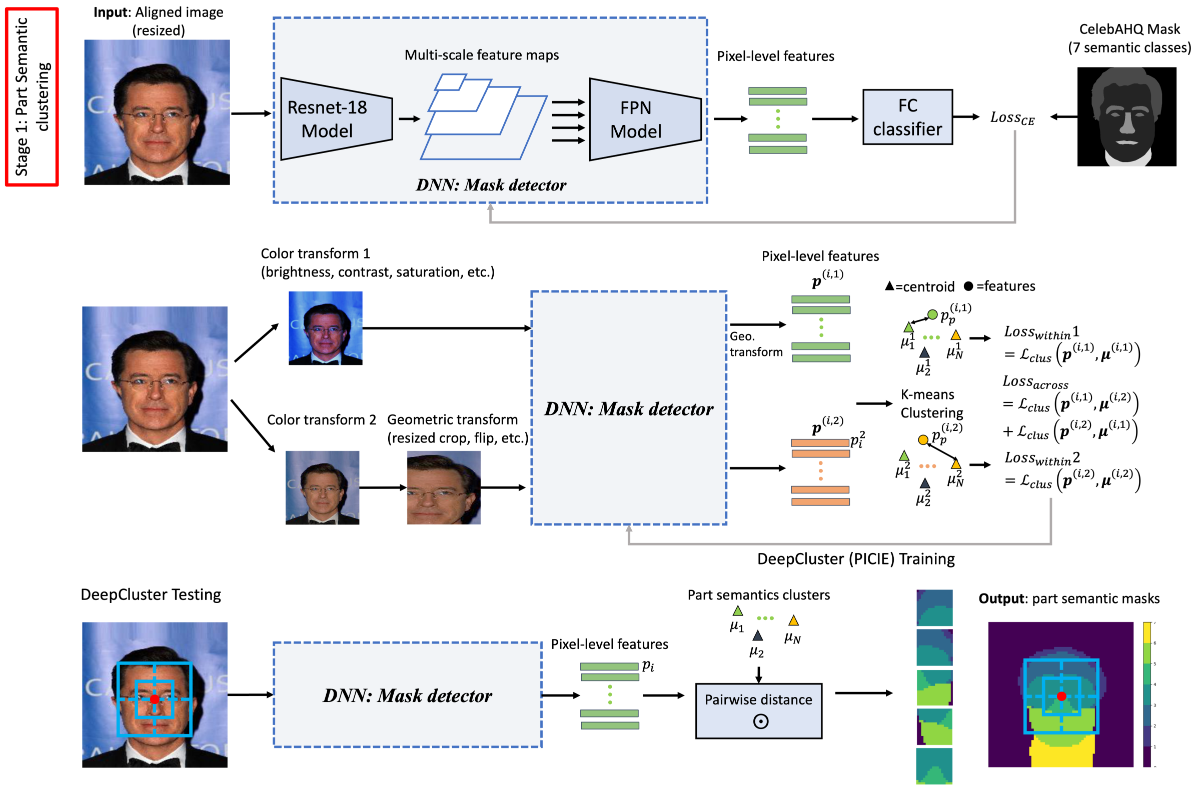

Stage 1: Part Semantic clustering
Input: Aligned image (resized)
Multi-scale feature maps
Resnet-18 Model
FPN Model
DNN: Mask detector
Pixel-level features
FC classifier
$Loss_{CE}$
CelebAHQ Mask (7 semantic classes)
Color transform 1 (brightness, contrast, saturation, etc.)
Color transform 2
Geometric transform (resized crop, flip, etc.)
Geo. transform
▲=centroid ●=features
K-means Clustering
$Loss_{within}1 = \mathcal{L}_{clus}(p^{(i,1)}, \mu^{(i,1)})$
$Loss_{across} = \mathcal{L}_{clus}(p^{(i,1)}, \mu^{(i,2)}) + \mathcal{L}_{clus}(p^{(i,2)}, \mu^{(i,1)})$
$Loss_{within}2 = \mathcal{L}_{clus}(p^{(i,2)}, \mu^{(i,2)})$
DeepCluster (PICIE) Training
DeepCluster Testing
Part semantics clusters
Pairwise distance ⊙
Output: part semantic masks

Figure 3: The first stage of training SSDI detection pipeline: learning part semantics of objects.

3.2 Vulnerability of Computer Vision

The SSDI vulnerability, where differences in the visual appearance of an image are noticeable to humans, stands in contrast to adversarial examples (Szegedy et al., 2013; Carlini & Wagner, 2017; Madry et al., 2017; Wang & He, 2021; Gubri et al., 2022; Zhang et al., 2023) where the adversarial image and the clean image appear visually similar to humans. Adversarial examples have been extensively studied in works of Madry et al. (2017); Shah et al. (2023); Mo et al. (2022); Andriushchenko & Flammarion (2020) which explore methods to prevent detrimental changes in the output predictions of classification models. However, these approaches would not apply to SSDI as they do not detect abnormalities in appearance. Moving away from the broader concept of adversarial examples, Hendrycks & Dietterich (2018) introduced a benchmark, ImageNet-C, for common corruptions and perturbations. While this research delves into changes in the natural appearance of images through noise, or color manipulations, it doesn't address cases where corruption stems from the rearrangement of natural feature order without affecting classifier predictions.

4 Methodology

In this section, we outline our proposed classification framework, which detects SSDI using a semi-supervised, two-stage approach. **The first stage** which we refer to as *part semantics*, focuses on learning "***image semantics***". The idea is to create a pixel-wise segmentation map of the input image that creates a (limited) set of clusters, each representing meaningful object parts (object semantics). **The second stage** then utilizes the part semantics to learn how their relative arrangement constitutes an entire object, i.e., the "***image syntax***". The idea is for the model to learn the expected arrangement and composition *based on* part semantics obtainable from natural images. **During inference**, the deviation from the expected arrangements and the compositions learned from natural images serve as the metric to detect SSDI.

Stage 1: Learning Part Semantics As mentioned, the first stage focuses on learning the part meaningful attributes of an object - its semantics. To learn this, the input pixels are segmented (grouped) into the set of classes that represent individual object parts instead of the objects themselves. Although fully unsupervised (no labels) or weakly supervised (only object class labels) approaches are more appealing, our observations showed that these methods did not produce semantic maps with the desired level of detail and accuracy. These approaches only allow the separation of the entire object from the background (e.g. the entire face), but not its parts (e.g. nose, eyes, mouth). This is because there is a greater variation in color between the object and the background than among the parts within the object itself. Hence, this stage was implemented in a semi-supervised approach by using only a fraction of the ground truth semantic segmentation maps with the deep clustering (Caron et al., 2018; Cho et al., 2021).

The training procedure is shown in Figure 3. Consider a set of input images $x_i, i = 1, ..., N$. Let a subset of them have pixel-level annotations $m_{ip}, i = 1, ..., M, M \ll N$, where p stands for the p-th pixel. We assume that we have C semantic classes in the pre-processed ground-truth (GT) annotations and $\forall(i,p), m_{ip} \in [0, 1, ...C-1]$. The feature extractor has an embedding function f_θ, which produces pixel-level feature vectors $z_{ip} = f_\theta(x_i)[p], z_{ip} \in \mathbb{R}^d$.

First, we fine-tune the DNN feature extractor under semi-supervision. Pixel-level embedding features are fed to a linear classifier $g_\omega, \omega \in \mathbb{R}^{d\times C}$ so that the feature dimensions are projected onto the number of semantic classes C. (Note, once the feature extractor has been fine-tuned, the classifier g can be discarded.) For all images x_i that have GT masks, the resulting pixel-level prediction becomes $g_\omega(z_{ip})$. The fine-tuning is then realized by minimizing the cross entropy loss between pixel-level prediction $g_\omega(z_{ip})$ and GT segmentation mask m_{ip} as shown in Equation 1:

$$\min_{\theta,\omega} \mathcal{L}_{CE}(g_\omega(f_\theta(x_i)[p]), m_{ip}) \quad \text{where} \quad \mathcal{L}_{CE} = -\log \frac{\exp(g_\omega(z_{ip})[m_{ip}])}{\sum_{k=0}^{C-1} \exp(g_\omega(z_{ip})[k])} \tag{1}$$

As a second step, deep clustering technique is used to expand upon semi-supervised knowledge. We used PiCIE, a deep clustering technique (Cho et al., 2021) in our approach. The core idea of PiCIE is to process every training image using different combinations of color and geometric transformations. The resulting features are then used to (a) determine the cluster centroids in K-means clustering and (b) keep the consistency of these clusters, such that features of the same image should map to the same set of clusters ("within") and features of different images should map to the different clusters ("cross"). These are achieved by iteratively optimizing the loss defined by Equation 2 for K-means clustering and the sum of losses defined by Equations 3-5 for PiCIE method:

$$\mathcal{L}_{Kmeans} = \sum_{i=1}^{N} \sum_p \sum_{k=1}^{K} \|z_{ipk} - \mu_k\|^2 \tag{2}$$

$$\mathcal{L}_{within} = \sum_{i=1}^{N} \sum_p \mathcal{L}_{DC}(z_{ip}^{(1)}, y_{ip}^{(1)}, \boldsymbol{\mu}) + \mathcal{L}_{DC}(z_{ip}^{(2)}, y_{ip}^{(2)}, \boldsymbol{\mu}) \tag{3}$$

$$\mathcal{L}_{cross} = \sum_{i=1}^{N} \sum_p \mathcal{L}_{DC}(z_{ip}^{(1)}, y_{ip}^{(2)}, \boldsymbol{\mu}) + \mathcal{L}_{DC}(z_{ip}^{(2)}, y_{ip}^{(1)}, \boldsymbol{\mu}) \tag{4}$$

$$\text{where} \quad \mathcal{L}_{DC}(z_{ip}, y, \boldsymbol{\mu}) = -\log \frac{\exp(-d(z_{ip}, \mu_y))}{\sum_{k=1}^{K} \exp(-d(z_{ip}, \mu_k))} \tag{5}$$

where K is the total (pre-selected) number of clusters, $\boldsymbol{\mu}$ is the matrix of cluster centroids, y is the centroid index in $\boldsymbol{\mu}$ that is closest to z_{ip}, and $d(\cdot,\cdot)$ is the cosine distance. As shown in Figure 3, the result of the first training stage is a DNN that can produce semantic masks for object parts, i.e. part semantics. The key is the desired granularity, such that the resulting part semantics are consistent for any given portion/patch of an image.

Stage 2: Learning Image Syntax The second stage is the core of training the pipeline to detect SSDI. The idea is to utilize the relation between neighboring part semantics. For example, if we know that some portion of an image contains a human nose, the expectation is to have two eyes and a mouth to be depicted above and below it, respectively.

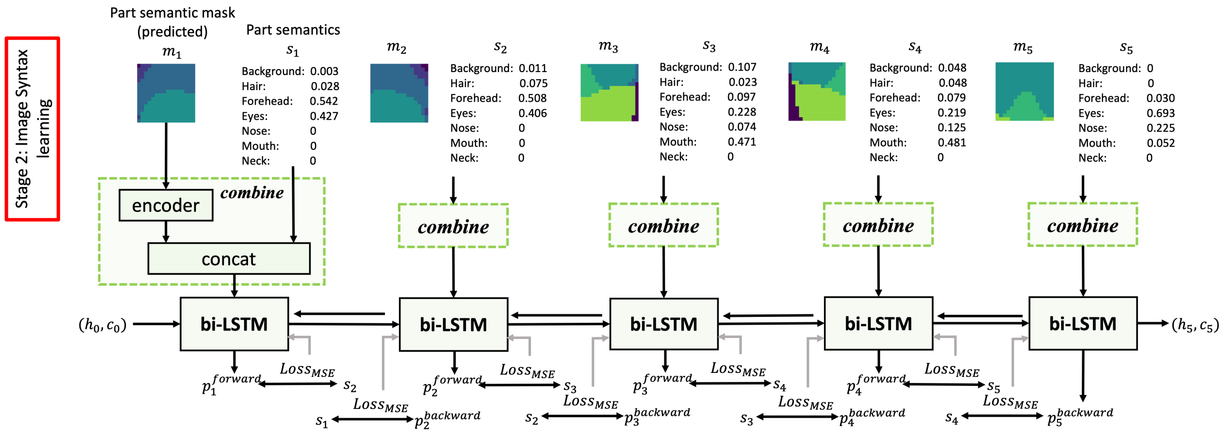


Figure 4: The second stage of training SSDI detection pipeline: learning image syntax.

The first step in achieving this is to obtain different patches from within the input image. The works of FALcon (Ibrayev et al., 2023) and GFNet (Wang et al., 2020) explore processing input in a patch-wise sequential manner with only the class label given as the supervision. In this work, we consider a simple and generic approach of dividing images, i.e. the model on its own does not make any special decisions on how the patches are obtained. For two datasets that are considered in this work, CelebA and SUN-RGBD, the patches are obtained as five crops around the center of the image (Figure 6) and as a zig-zag pattern from top to bottom, from left to right (Figure 8), respectively.

The second step is to traverse through the set of image patches, extract their corresponding part semantics, and learn the relation between the part semantics of neighboring patches. Suppose for image $x^{(i)}, i = 1, ..., N$, we extract a total of G patches and obtain their corresponding part semantics $m_t^{(i)}$ for $t = 1, ..., G$ using the DNN trained in the first stage. The image syntax depends on the presence of the object parts with respect to each other rather than the exact part semantics $m_t^{(i)}$. Hence, each of them is converted into part semantics vector $s_t^{(i)}$, which is the ratio of pixels belonging to each class in the part semantics.

Figure 4 shows how the image syntax is learned based on part semantics vectors using bidirectional long short-term memory (bi-LSTM). The bi-LSTM is used because both directions in the traversal of part semantics are valid relations to be learned. If it is parameterized by weights $\mathbf{W}$ and biases $\mathbf{b}$, then based on the inputs $m_t^{(i)}$ and $s_t^{(i)}$ as well as hidden states $h_t^{(i)}$ at the processing step t the bi-LSTM makes two predictions: $p_t^{(i,for)} = L_{\mathbf{W},\mathbf{b}}(m_t^{(i)}, s_t^{(i)}, h_t^{(i,for)})$ for the next semantics $s_{t+1}^{(i)}$ and $p_t^{(i,back)} = L_{\mathbf{W},\mathbf{b}}(m_t^{(i)}, s_t^{(i)}, h_t^{(i,back)})$ for the previous semantics $s_{t-1}^{(i)}$. Across the set of training images $\mathbf{x}$, the learning goal of bi-LSTM is to capture the transition patterns between image part semantics. This is achieved by optimizing the mean squared error loss formulated by Equation 6:

$$\mathcal{L}_{MSE}(p_t^{(i)}, s_t^{(i)}) = \sum_{i=1}^{N} \big(\sum_{t=2}^{G} \|p_t^{(i,for)} - s_t^{(i)}\|^2 + \sum_{t=1}^{G-1} \|p_t^{(i,back)} - s_t^{(i)}\|^2 \big) \tag{6}$$

Inference: Detecting SSDI The detection of images with semantic-syntactic discrepancy (SSDI) is achieved by processing images in a similar patch-wise traversing manner and verifying whether the arrangement of object parts matches the expected arrangements learned from natural images. Figure 5 illustrates three methods of formally quantifying this process. The first method detects SSDI based on the relation of part semantics present in the image itself. Similar to the training process, for every image patch t, based on its part semantics bi-LSTM predicts the part semantics of its neighboring patches. The error is then computed as the average difference between predicted part semantics and part semantics estimated by the DNN.

The other two methods detect SSDI based on the memorization of the average part semantics of every pixel. Specifically, each pixel is assigned the most frequently predicted semantic class over the training set. For a

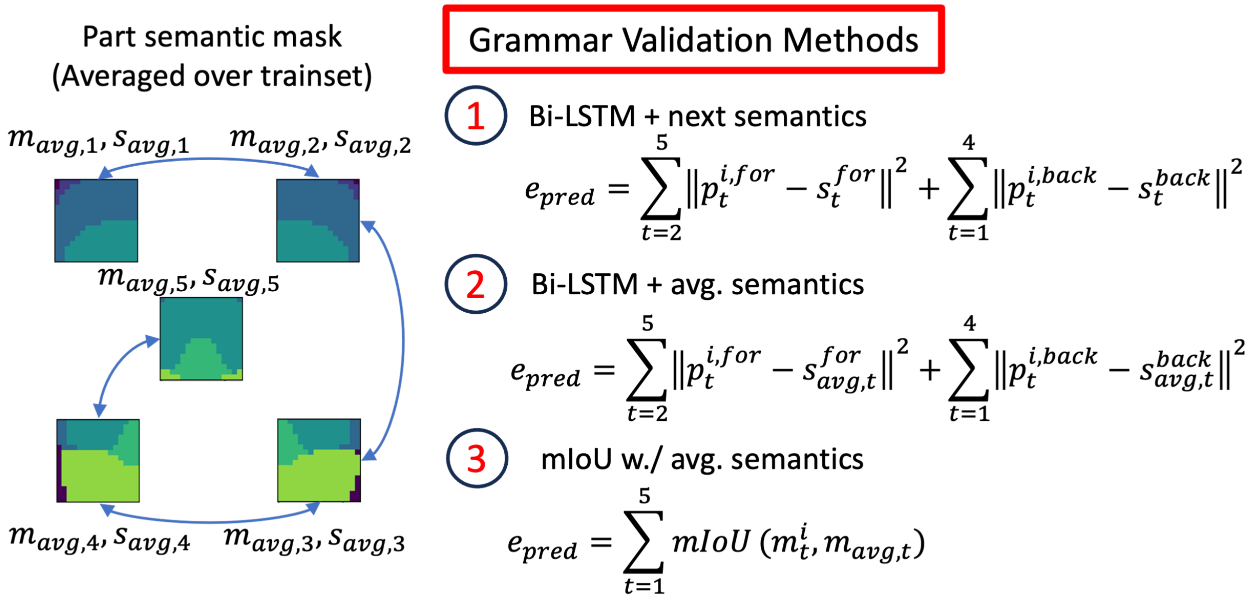

Part semantic mask
(Averaged over trainset)
$m_{avg,1}, s_{avg,1}$
$m_{avg,2}, s_{avg,2}$
$m_{avg,5}, s_{avg,5}$
$m_{avg,4}, s_{avg,4}$
$m_{avg,3}, s_{avg,3}$
Grammar Validation Methods
1 Bi-LSTM + next semantics
$e_{pred} = \sum_{t=2}^{5} \|p_t^{i,for} - s_t^{for}\|^2 + \sum_{t=1}^{4} \|p_t^{i,back} - s_t^{back}\|^2$
2 Bi-LSTM + avg. semantics
$e_{pred} = \sum_{t=2}^{5} \|p_t^{i,for} - s_{avg,t}^{for}\|^2 + \sum_{t=1}^{4} \|p_t^{i,back} - s_{avg,t}^{back}\|^2$
3 mIoU w./ avg. semantics
$e_{pred} = \sum_{t=1}^{5} mIoU\,(m_t^i, m_{avg,t})$

Figure 5: Three methods of quantifying the semantic-syntactic discrepancy in images, SSDI.

test image, the error is then computed as the average difference between predicted part semantics and part semantics memorized based on the average semantics of training images. In the third validation method, the mean intersection over union (mIoU) is used instead of the difference, which is a common segmentation metric (Garcia-Garcia et al., 2017).

In all three cases, we obtain a quantitative measure of estimating the discrepancy in semantic-syntactic relation in the form of computed errors. To use it as the detection during inference, we need to separate the error values for images with a natural and unnatural appearance. This is achieved by generating images with semantic-syntactic discrepancy and estimating thresholds for each method based on the validation set images.

5 Results

Generation of SSDI and Performance Evaluation In this work, we consider three types of corruption causing semantic-syntactic discrepancy in images. The first corruption is a *patch shuffling*, where a subset of image patches is swapped. The second corruption is a *patch blackening*, where a subset of image patches is blackened out. In both of these cases, the resulting image is processed by the framework to decide whether it is a natural image or not. The third corruption is a *puzzle solving*, where a subset of image patches is swapped, similarly to the patch shuffling. However, unlike patch shuffling, in puzzle solving, we generate all possible permutations for the specified number of image patches allowed for corruption. Then, all possible patch permutations and the original natural image are fed to the framework, which has to predict the image with the natural arrangement of image patches/object parts. For each considered dataset, SSDI corruptions are generated using half of the test set images chosen at random. All corruptions are considered separately, i.e. the paper does not consider a simultaneous mixture of different SSDI corruptions.

As described in Section 4, both natural and corrupted images with SSDI are processed by the framework in a patch-wise traversal manner and the (residual) errors in predicting forward and backward part semantics by bi-LSTM are computed. The threshold determined based on the validation set is used to distinguish between natural and corrupted images. We evaluate the performance based on two metrics: (1) the detection accuracy

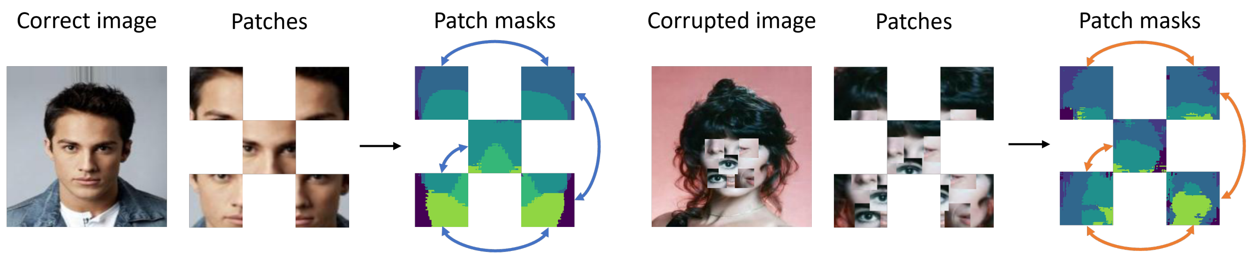


Figure 6: Image patches and the traversal sequences of processing image syntax of CelebA.

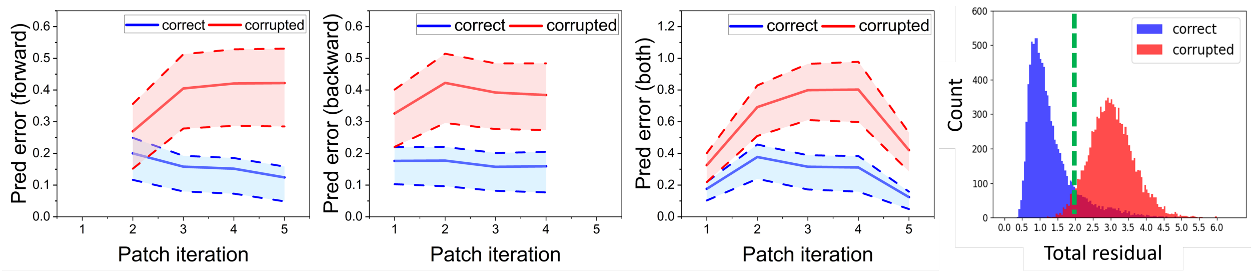


Figure 7: From left to right: the forward, the backward, and the total bi-LSTM prediction errors as well as the histogram of total errors over test samples of CelebA dataset.

(i.e. binary classification accuracy) and (2) the detection rate (i.e. recall) described by Equation 7.

$$\text{Det. Acc.} = \frac{TruePositive(TP) + TrueNegative(TN)}{FalseNegative(FN) + TP + FalsePositive(FP) + TN} \quad \text{and} \quad \text{Det. Rate} = \frac{TP}{FN + TP} \tag{7}$$

CelebA The first dataset is the aligned-and-cropped CelebA (Liu et al., 2015) dataset with 202599 RGB images resized to 256×256. The DNN feature extractor consists of Feature Pyramid Network (FPN) (Lin et al., 2017) with ImageNet pre-trained ResNet-18 (He et al., 2016) as the backbone to extract $128 \times 64 \times 64$ downsampled pixel-level features. The FPN is finetuned with 30,000 CelebAHQ (Lee et al., 2020) face part segmentation masks (about 15% of CelebA). We perform mini-batch K-means clustering every 20 batches before a single centroid update. The number of clusters is set to 20. Part semantics masks were restricted to have 7 categories, i.e. every object is limited to be described by 7 object parts (including background) from selecting the top 7 clusters. For image syntax learning, a 1-layer bi-LSTM model is trained. Input and hidden vectors are 135-dimensional (128-d encoded mask concatenated with 7-d semantics), and the projected outputs are 7-d. Figure 6 shows the natural and unnatural images during the test along with the 5 patches used in the traversal sequence and their corresponding predicted part semantics.

SSDI Detection Performance on CelebA Figure 7 shows the forward, backward, and the sum of both residual errors of bi-LSTM across iterations of processing image patches of both natural and corrupted images of the CelebA dataset. The solid lines are mean values, whereas the shaded region covers between 25% and 75% percentile. It can be seen that there is a separation between natural (blue) and corrupted SSDI (red) samples. The histogram shows the distribution of test samples based on the total residual error along with the threshold (green dashed line) used to distinguish samples from natural and corrupted.

Table 1 illustrates the performance of the proposed framework on the CelebA dataset for various grammar validation methods. Different columns represent corruption types, with the number and the size of the affected image patches. For example, **Shuffle** 2 30×30 means that 2 image patches of size 30×30 pixels are swapped to generate the patch shuffling corruption, whereas **Puzzles** 4 20×20 means that the frame-

Table 1: The performance of the proposed approach on detecting SSDI corruptions on CelebA.

Dataset	**Part Semantics Model**	**Grammar Validation Method**	**Shuffle 2 20x20**	**Shuffle 2 30x30**	**Black 1 20x20**	**Puzzles 4 20x20**	**Puzzles 4 30x30**
CelebA	ResNet18+FPN (ours)	Bi-LSTM + next semantics	65.66 (68.67)	76.22 (79.72)	70.91 (67.04)	86.74	92.51
CelebA	ResNet18+FPN (ours)	Bi-LSTM + avg. semantics	61.53 (**71.72**)	68.89 (**80.48**)	65.84 (**72.38**)	85.62	91.15
CelebA	ResNet18+FPN (ours)	mIoU w./ avg. semantics	60.04 (56.16)	69.59 (59.60)	60.90 (61.87)	**99.25**	**99.58**
CelebA	SemanticGAN (Li et al., 2021) (2021)	Bi-LSTM + next semantics	61.35 (**69.87**)	70.20 (**73.68**)	68.98 (**71.04**)	82.37	88.09
CelebA	SemanticGAN (Li et al., 2021) (2021)	Bi-LSTM + avg. semantics	58.26 (67.91)	63.57 (71.49)	61.28 (65.78)	81.23	86.79
CelebA	SemanticGAN (Li et al., 2021) (2021)	mIoU w./ avg. semantics	56.76 (58.34)	62.75 (64.25)	62.13 (61.87)	**98.60**	**99.12**

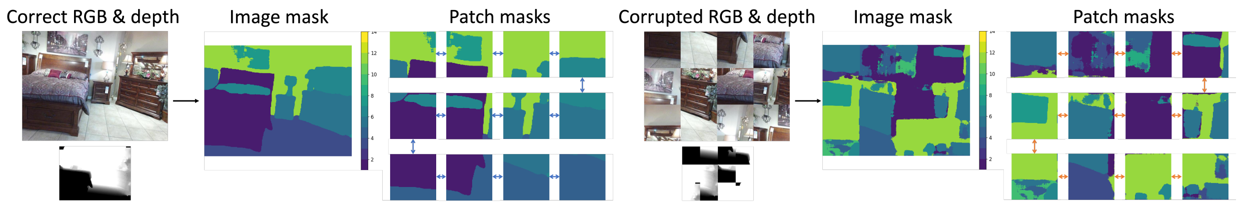


Figure 8: Image patches of size 160 and the sequences of processing image syntax of SUN-RGBD.

work processes a batch generated from all possible permutations (including that of the original image) of 4 randomly selected patches of size 20 × 20 pixels. Both the detection accuracy and the detection rate results are reported, with the latter presented in parentheses.

When ResNet18 and FPN are used as the models for extraction of part semantics, it can be seen that using the memorized average part semantics performs the best among the three grammar validation methods. This can be explained by the fact that in the CelebA dataset, the object is the human face with a very well-defined structure. Hence, it is more effective to learn not only the expected relation between the neighboring object parts (i.e. the image syntax) but also what those object parts are expected to be (i.e. the memorized average part semantics). The puzzle variation of SSDI corruption poses an interesting problem since a large number of possible rearrangements of object parts observed at the same time significantly reduces the margin at which they can be distinguished based on the residual error. Interestingly, the framework achieves above 99% performance on puzzles, highlighting that the proposed approach indeed learned the correct composition of object parts.

SUN-RGBD The second dataset is the SUN-RGBD (Song et al., 2015) dataset with 10,335 room layout RGB and depth images. The DNN feature extractor consists of a pretrained Residual Encoder-Decoder (RedNet) (Jiang et al., 2018), which is used to retrieve part semantics masks from 640 × 480 sized images. The object parts were restricted to 13 based on the obtained semantic masks. For syntax learning, we use the same bi-LSTM model as in CelebA. As shown in Figure 8, we define zig-zag traversal rules on sequences of patches of sizes 160 × 160 and 80 × 80 pixels. The residual prediction errors and the histogram for test set images are shown in Figure 9.

Table 2 shows detection accuracy and detection rates on SUN-RGBD dataset. Unlike the performance on the object-centric CelebA dataset, it is interesting to notice that the first grammar validation method is the most optimal for other corruptions on the scene-centric SUN-RGBD dataset. This can be explained by a more diverse nature of images in SUN-RGBD: different objects can be arranged in multiple different ways

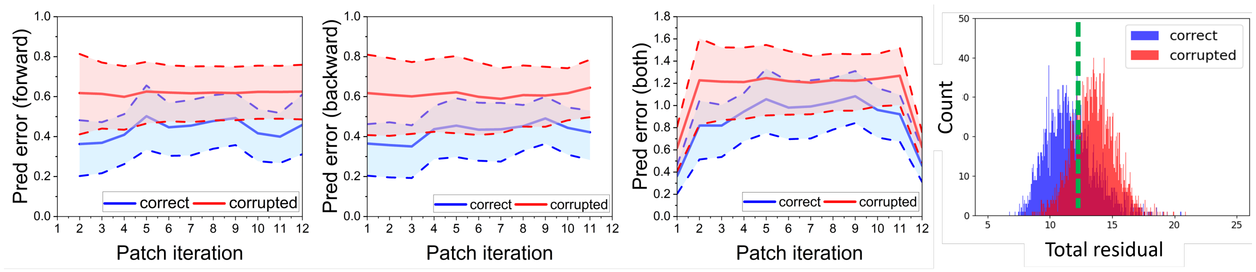


Figure 9: From left to right: the forward, the backward, and the total bi-LSTM prediction errors as well as the histogram of total errors over test samples of SUN-RGBD dataset.

Table 2: The performance of the proposed approach on detecting SSDI corruptions on SUN-RGBD.

Dataset	**Part Semantics Model**	**Grammar Validation Method**	**Shuffle 4 160x160**	**Shuffle 16 80x80**	**Black 4 160x160**	**Puzzles 4 160x160**	**Puzzles 16 80x80**
SUN-RGBD (13-cls.)	ResNet50+Encoder-Decoder (ours)	Bi-LSTM + next semantics	60.57 (**72.04**)	76.57 (**73.37**)	66.55 (**67.21**)	72.89	91.17
SUN-RGBD (13-cls.)	ResNet50+Encoder-Decoder (ours)	Bi-LSTM + avg. semantics	54.97 (65.36)	61.52 (64.09)	58.46 (59.65)	62.47	75.48
SUN-RGBD (13-cls.)	ResNet50+Encoder-Decoder (ours)	mIoU w./ avg. semantics	57.98 (68.04)	58.98 (62.24)	56.32 (57.19)	**97.09**	**98.20**
SUN-RGBD (13-cls.)	Dformer-S (Yin et al., 2023) (2023)	Bi-LSTM + next semantics	62.16 (**73.47**)	74.23 (**72.80**)	68.86 (**71.23**)	76.61	92.84
SUN-RGBD (13-cls.)	Dformer-S (Yin et al., 2023) (2023)	Bi-LSTM + avg. semantics	53.77 (67.23)	62.45 (69.09)	61.98 (62.45)	61.97	74.78
SUN-RGBD (13-cls.)	Dformer-S (Yin et al., 2023) (2023)	mIoU w./ avg. semantics	53.24 (63.38)	60.46 (66.21)	59.79 (61.80)	**98.13**	**98.62**

with respect to the entire scene and other objects. Hence, as scenes might change substantially, the method of using the stored average part semantics might be less useful than relying solely on the relations between object parts local to individual images.

Effect of Segmentation Granularity On both datasets, we also consider more specialized alternatives for the segmentation masks, which are expected to produce more accurate and finer part semantics. The methods of SemanticGAN (Li et al., 2021) and Dformer (Yin et al., 2023) were considered as part semantics models for CelebA and SUN-RGBD datasets, respectively. From test results presented in Table 1 and Table 2, it can be seen that finer semantic masks do not always lead to higher SSDI detection rates, especially on object-centric dataset like CelebA. This can probably be explained by the fact that it is easier to capture the natural arrangement of object parts if their part semantics is coarser. The relationship between semantic granularity and corruption detection deserves further investigation.

6 Limitations

While this work proposes a pioneering approach to tackle SSDI corruptions, there are a few limitations. First, the semi-supervised approach requires some fraction of semantic segmentation masks to obtain semantic separation of parts within the object, rather than objects from the background. A possible solution is to consider fully self-supervised segmentation methods, like DINO (Caron et al., 2021). Second, the current image syntax learning relies on predetermined fixed traversal across patches. It might be beneficial to consider either active vision methods (Elsayed et al., 2019; Wang et al., 2020; Ibrayev et al., 2023) or attention-driven models (Dosovitskiy et al., 2021; Jiang et al., 2024).

7 Conclusion

Motivated to bridge the gap between human and machines perception related to unnatural images, we introduce a novel deep learning framework for addressing semantic-syntactic discrepancy in images (SSDI). Aligned with the concept of language grammar, the framework learns both the meaning and the natural arrangement of object parts using deep clustering and a bidirectional LSTM. The effectiveness of the approach is shown through its capability of solving puzzles and achieving detection rates of $70 - 90\%$ on CelebA and $60 - 80\%$ on SUN-RGBD. The pioneering aspect of the problem suggests a broader impact in areas requiring reliable analysis of image content.

Broader Impact Statement

Machine learning systems and, specifically, computer vision techniques are having an increased influence on everyday tasks that benefit from automation. These tasks range from those with minimal impact on human life, such as facial recognition systems on mobile phones, to highly significant ones, such as obstacle detection in autonomous vehicles. A huge amount of trust is being placed in the learning frameworks that are used for performing these tasks. These frameworks are often trained on large datasets curated by retrieving images using machine learning recognition techniques. If the images gathered are unnatural (do not follow the syntax of real-world images), the learning of the downstream learning frameworks is also tainted. Hence, it is important for these learning frameworks to (a) be trained on data that accurately represents real-world scenarios and (b) learn the syntax and semantics of the data accurately. This work addresses both these concerns by highlighting (a) the SSDI vulnerability in datasets and the importance of learning "image grammar" consisting of "image syntax" and "image semantics" as well as (b) proposing a two-stage weakly supervised framework that attempts to minimize the possibility of the malicious use of these corruptions by learning the mentioned "image grammar". This paves the pathway for future researchers to implement computer vision techniques that operate holistically - learn both the syntax and semantics.

However, as in the many examples in the history of digital systems, the exposure of vulnerabilities may lead to a path for malicious entities to exploit those vulnerabilities on existing systems. For example, when the vulnerability of computer vision techniques to added additive noise was exposed, many adversarial techniques were developed to trick existing machine learning frameworks. Consequently, there is a potential for developing attacks or malicious techniques that exploit the SSDI vulnerability in images for nefarious purposes. Moreover, the situation is worsened by the fact that such vulnerabilities are currently not detectable purely by the machine learning, but would require the human-expert in the loop for security. This further underscores the need for systems to wholly learn the "image syntax" along with the "image grammar".

References

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In *NeurIPS*, 2020.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. Ieee, 2017.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.

Jang Hyun Cho, Utkarsh Mall, Kavita Bala, and Bharath Hariharan. Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16794–16804, June 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Gamaleldin Elsayed, Simon Kornblith, and Quoc V Le. Saccader: Improving accuracy of hard attention models for vision. *Advances in Neural Information Processing Systems*, 32, 2019.

Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

Martin Gubri, Maxime Cordy, Mike Papadakis, Yves Le Traon, and Koushik Sen. Lgv: Boosting adversarial example transferability from large geometric vicinity. In *ECCV*, 2022.

Thomas C. Gunter, Laurie A. Stowe, and Gusbertus Mulder. When syntax meets semantics. *Psychophysiology*, 34(6):660–676, 1997. doi: https://doi.org/10.1111/j.1469-8986.1997.tb02142.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8986.1997.tb02142.x.

Ankur Handa, Viorica Pătrăucean, Simon Stent, and Roberto Cipolla. Scenenet: an annotated model generator for indoor scene understanding. In *ICRA*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90. URL https://ieeexplore.ieee.org/document/7780459/.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

Yuansheng Hua, Lichao Mou, and Xiao Xiang Zhu. Lahnet: A convolutional neural network fusing low- and high-level features for aerial scene classification. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 4728–4731, 2018. doi: 10.1109/IGARSS.2018.8519576.

Timur Ibrayev, Manish Nagaraj, Amitangshu Mukherjee, and Kaushik Roy. Exploring foveation and saccade for improved weakly-supervised localization. In *NeuRIPS 2023 Workshop on Gaze Meets ML*, 2023. URL https://openreview.net/forum?id=qUfLsi3Vlm.

Hanwen Jiang, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Single-stage visual query localization in egocentric videos. *Advances in Neural Information Processing Systems*, 36, 2024.

Jindong Jiang, Lunan Zheng, Fei Luo, and Zhijun Zhang. Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation. *arXiv preprint arXiv:1806.01054*, 2018.

Alexander Kirillov, Ross B. Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, pp. 6399–6408. Computer Vision Foundation / IEEE, 2019. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2019.html#KirillovGHD19.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Daiqing Li, Junlin Yang, Karsten Kreis, Antonio Torralba, and Sanja Fidler. Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, pp. 936–944. IEEE Computer Society, 2017. ISBN 978-1-5386-0457-1. URL `http://dblp.uni-trier.de/db/conf/cvpr/cvpr2017.html#LinDGHHB17`.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Yichuan Mo, Dongxian Wu, Yifei Wang, Yiwen Guo, and Yisen Wang. When adversarial training meets vision transformers: Recipes from training to architecture. In *Advances in Neural Information Processing Systems*, 2022.

Diego Ortego, Eric Arazo, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Towards robust learning with different label noise distributions. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 7020–7027, 2021. doi: 10.1109/ICPR48806.2021.9412747.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Yao Qin, Chiyuan Zhang, Ting Chen, Balaji Lakshminarayanan, Alex Beutel, and Xuezhi Wang. Understanding and improving robustness of vision transformers through patch-based negative augmentation. *Advances in Neural Information Processing Systems*, 35:16276–16289, 2022.

Muhammad A Shah, Aqsa Kashaf, and Bhiksha Raj. Training on foveated images improves robustness to adversarial attacks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2015. URL `http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#SimonyanZ14a`.

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pp. 567–576. IEEE Computer Society, 2015. ISBN 978-1-4673-6964-0. URL `http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#SongLX15`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5238–5248, 2022.

Arnim von Stechow. Syntax and semantics: an overview. *Semantics-Interfaces*, pp. 169, 2019.

Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *CVPR*, 2021.

Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. In *Advances in Neural Information Processing Systems*, 2020.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision, 2020.

Bowen Yin, Xuying Zhang, Zhongyu Li, Li Liu, Ming-Ming Cheng, and Qibin Hou. Dformer: Rethinking rgbd representation learning for semantic segmentation. *arXiv preprint arXiv:2309.09668*, 2023.

Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=KRLUvxh8uaX`.

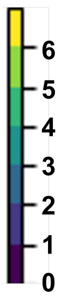
Jianping Zhang, Yizhan Huang, Weibin Wu, and Michael R Lyu. Transferable adversarial attacks on vision transformers with token gradient regularization. In *CVPR*, 2023.

A Experiment details

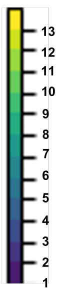
The entire framework was implemented using the PyTorch framework (Paszke et al., 2019). The models were trained and evaluated using 4 NVIDIA GeForce GTX 2080 Ti GPU cards.

A.1 Datasets

CelebA (Liu et al., 2015) contains 202,599 face images. We used the aligned-and-cropped version where faces are localized. Images are divided into a train set of size 162,770, a validation set of size 19,867, and a test set of size 19,962. First, the FPN network is finetuned on 30,000 CelebAHQ (Lee et al., 2020) images selected from CelebA. CelebAHQ images and pixel-wise labels are center-cropped (size=160) and resized to 256×256. After fine-tuning, PiCIE deep clustering technique (Cho et al., 2021) is trained on train set images resized to 256×256 and validated on the validation set. 64×64 patches are cropped from segmentation and are upsampled using bilinear interpolation to 256×256. The 7 semantic classes and their names are illustrated on the left part of Figure 10. During the test, the corruptions are generated around 5 facial part landmark locations (left eye, right eye, nose, left mouth, right mouth) using a half of the randomly selected test images.



Label	Class name
0	Background
1	Hair
2	Forehead
3	Eyes
4	Nose
5	Mouth
6	Neck



Label	Class name	Label	Class name
1	Bed	8	Picture
2	Books	9	Sofa
3	Ceiling	10	Table
4	Chair	11	TV
5	Floor	12	Walls
6	Furniture	13	Window
7	Objects		

Figure 10: Segmentation Classes. Left: CelebA; Right: SUN-RGBD

SUN-RGBD (Song et al., 2015) contains 10335 RGB images and corresponding depth images, split into a train set of size 4785, a validation set of size 1000, and a test set of size 5050. RGB and depth images are resized to 640×480 and normalized. The pre-trained RedNet (Jiang et al., 2018) also produces 640×480 part semantic masks. From each segmentation, 160×160 and 80×80 patch semantic masks are cropped. The segmentation contains 37 semantic classes, but we merge them into 13 classes with the same mapping used in the work of Handa et al. (2016). The 13 semantic classes are shown on the right part of Figure 10. During the test, we add corruptions to image patches using a half of the randomly selected test images.

A.2 Architectures

On CelebA Figure 11 illustrates the feature pyramid network (FPN) (Lin et al., 2017) that was used for part semantics clustering and segmentation of CelebA (Liu et al., 2015) images. Specifically, Panoptic FPN proposed in the work of Kirillov et al. (2019) was used in the combination with ResNet-18 (He et al., 2016) as the backbone feature extractor. While the backbone encodes multi-scale feature representation in a top-down fashion, FPN model utilizes 1×1 convolutional (Conv) layers in the bottom-up manner to project pixel-wise features to 128-dimensional space. The projected features of different spatial sizes are upsampled using bilinear interpolation to 1/4 height and width of the original image and then element-wise summed. The pixel-wise representations thus have shape $128 \times h \times w$ ($128 \times 64 \times 64$ for input images of shape $3 \times 256 \times 256$). The extracted pixel-wise features are used for the deep clustering (Cho et al., 2021). We record the resulting clustering centroids as a final 1×1 Conv layer to project these features to the dimension of the number of semantic classes to represent part semantics.

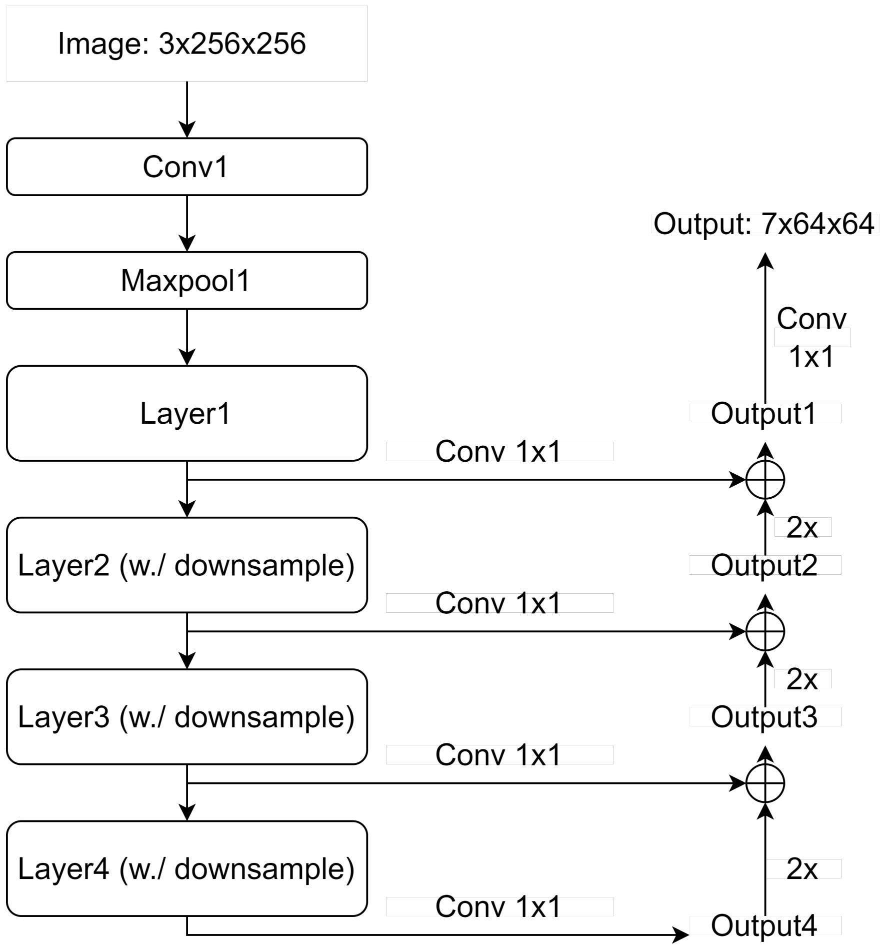


Figure 11: DNN architectures: Feature Pyramid Network (FPN) with ResNet-18 as the backbone

Figure 12 shows the structure of the bi-directional Long Short-Term Memory (bi-LSTM). We use it for image syntax learning. For each patch in the traversal sequence of length 5, the predicted mask is first flattened before being fed to a fully-connected (FC) encoder layer. The encoder projects the semantic mask to a 128-d vector, which is concatenated with the 7-d semantics vector. The resulting 135-d vector contains both spatial and semantic information about each patch's part semantics, and it is used as the input to 2 LSTM layers in forward and backward directions. After projection layers, the outputs include a 7-d forward prediction of the next part semantics semantics in the sequence and a 7-d backward prediction of the previous part semantics. For the first and last patches, only forward and backward predictions are made, respectively.

On SUN-RGBD For segmentation of SUN-RGBD (Song et al., 2015) images, we use a Residual Encoder-Decoder Network (RedNet) (Jiang et al., 2018) with ResNet-50 as the backbone. It is a state-of-the-art room scene parsing architecture that achieved 47.8% mIoU accuracy on 37-class SUN-RGBD. Similar to FPN, the RedNet extracts multi-scale feature maps from the input image with its encoder residual layers. However, each projected feature map is skip-connected with the output of upsampling residual units in a decoder (Jiang et al., 2018) before being summed with the next level in the feature pyramid. An additional depth branch is fused with the RGB branch. We performed 13-class (merged from 37-class) semantic segmentation with a trained RedNet. The RedNet produces the same-resolution (640×480) segmentation from the original image. Patch semantic masks of sizes 160×160 and 80×80 are cropped and used for preparing patch semantic vectors. At each stage of patch sequences, part semantics masks are flattened and projected to

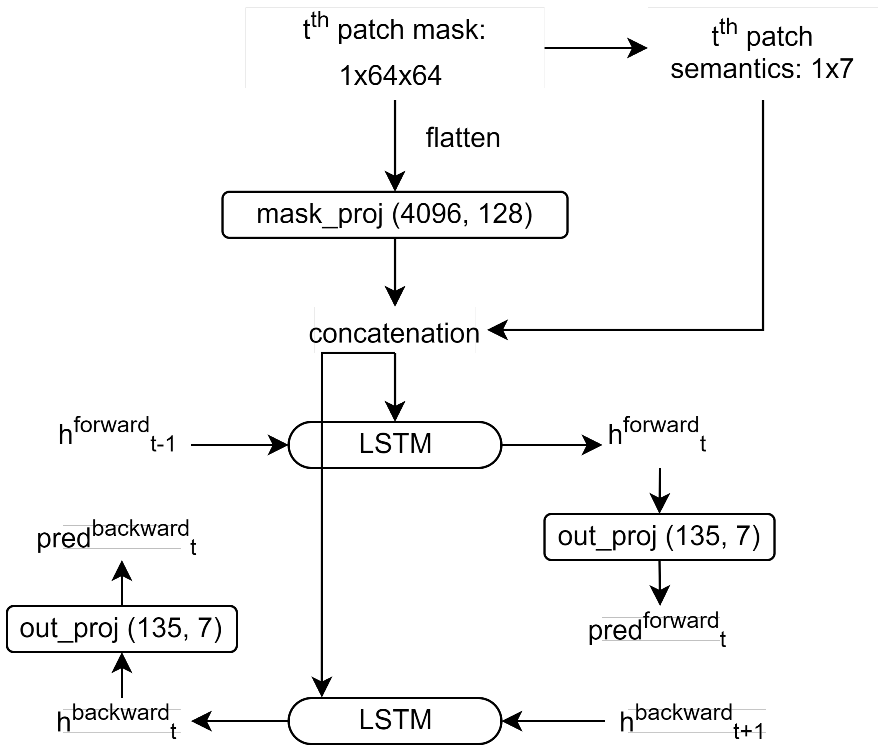


Figure 12: DNN architectures: bi-directional Long Short-Term Memory (bi-LSTM)

128-d vectors before concatenating with a 13-d part semantics vector. The resulting 141-d vectors are passed to LSTM blocks for forward and backward predictions (both 13-d).

A.3 Hyperparameters

On CelebA During fine-tuning of the FPN network pre-trained on ImageNet (Deng et al., 2009), CelebAHQ (Lee et al., 2020) masks are pre-processed to contain only 7 coarse semantic classes. *Adam* optimizer is used with $start\_lr = 1e^{-4}$. We trained for $num\_epochs = 20$ with a constant learning rate and $batch\_size\_train = 128$. The fine-tuned model is used for deep clustering PiCIE (Cho et al., 2021) training with $batch\_size\_train = 256$. We adopted over-clustering and set the number of clusters, $K\_train = 20$, as that results in better clusters compared to $KM\_num = 10$ and $KM\_num = 30$. Mini-batch K-means clustering is performed. $KM\_init = 20$ is the number of batches we collect before the first K-means clustering. $KM\_num = 20$ is the interval of batches between consecutive clustering. $KM\_iter = 100$ is the number of K-means clustering iterations before convergence. After each clustering, the total PiCIE loss is back-propagated to optimize FPN model parameters. We use *Adam* optimizer with $start\_lr = 1e^{-4}$. Despite the complexity of the total PiCIE training loss, the training converges fast in $num\_epochs = 10$ with a constant learning rate. To get semantic segmentation masks, we initiate a 1×1 Conv layer, the weights of which are loaded as the trained centroid matrix, and append it to the trained FPN network. We post-process pixel-wise class assignments to merge semantically-similar clusters and obtain 7-class face part semantics. Further, we tune the FPN network on 64×64 patches, using crops on the saved face part semantics as the supervision. The part semantics mask detector is trained with $batch\_size\_train = 128$, $start\_lr = 1e^{-4}$ for a total of $num\_epochs = 20$. We chose the $num\_epochs = 20$ model over the $num\_epochs = 40$ model because the smooth boundaries in the semantic masks help with image syntax learning.

After we obtain 7-class part semantics masks for each patch in the traversal sequence, we feed the concatenation of encoded mask (128-d) and semantics vector (7-d) into a bi-LSTM model. The parameters for bi-LSTM are: $input\_size = 135$, $hidden\_size = 135$, $num\_layers = 1$, $bidirectional = True$, $proj\_size = 7$. We use *Adam* optimizer with $start\_lr = 1e^{-4}$ and $num\_epochs = 40$, with the learning rate changed to $1e^{-5}$ after 20 epochs.

On SUN-RGBD On SUN-RGBD, the pre-trained RedNet network (Jiang et al., 2018) produces 37 segmentation classes. We merge these classes into 13-class as shown in Figure 10. When the patches are shuffled, the pre-trained network is able to produce precise part semantics, so that a part mask detector is not needed as in CelebA. We start with image syntax training with two configurations, $patch\_size = 160$ and $patch\_size = 80$. In both configurations, the part semantics masks are encoded as 128-d vectors, while the part semantics vector is 13-d. Thus, the bi-LSTM model has the configuration with $input\_size = 141$, $hidden\_size = 141$, $num\_layers = 1$, $bidirectional = True$, $proj\_size = 13$. We use *Adam* optimizer with $start\_lr = 1e^{-4}$ for a total of $num\_epochs = 40$. A multi-step learning rate scheduler is used with $gamma = 0.8$ and $milestones = [5, 10, 15, 20, 25, 30, 35]$.

B Generation of SSDI corruptions

One of the main contributions of this work is exposing the vulnerability of classification models to semantic-syntactic discrepancy in images (SSDI). We define three types of corruptions causing semantic-syntactic discrepancy in images and provide the algorithmic methodology of generating these SSDI corruptions. Furthermore, the supplementary material contains the code to generate the corresponding SSDI corruptions.

Figure 1 illustrates the algorithm used to generate the first type of SSDI corruption called a *patch shuffling*. This type of corruption is generated by swapping the positions of the subset of image patches. The performance of the SSDI detection approach is evaluated based on its capability to detect whether the given input is a natural image or not. Figure 2 illustrates the algorithm used to generate the second type of SSDI corruption called a *patch blackening*. This type of corruption is generated by blackening out a subset of image patches. Similarly to the *patch shuffling*, the performance is evaluated based on the detection capability of natural and corrupt images. Figure 3 illustrates the algorithm used to generate the third type of SSDI corruption called a *puzzle solving*. Similarly to the *patch shuffling*, this type of corruption permutes a subset of image patches. However, unlike patch shuffling, in puzzle solving, we generate all possible permutations for the specified number of image patches allowed for corruption. Then, all possible patch permutations and the original natural image are fed to the framework, which has to predict the image with the natural arrangement of image patches/object parts. For each considered dataset, SSDI corruptions are generated using half of the test set images chosen at random. All corruptions are considered separately, i.e. the paper does not consider a simultaneous mixture of different SSDI corruptions. Figure 13 illustrates all SSDI corruptions considered in this work on CelebA and SUN-RGBD test images.

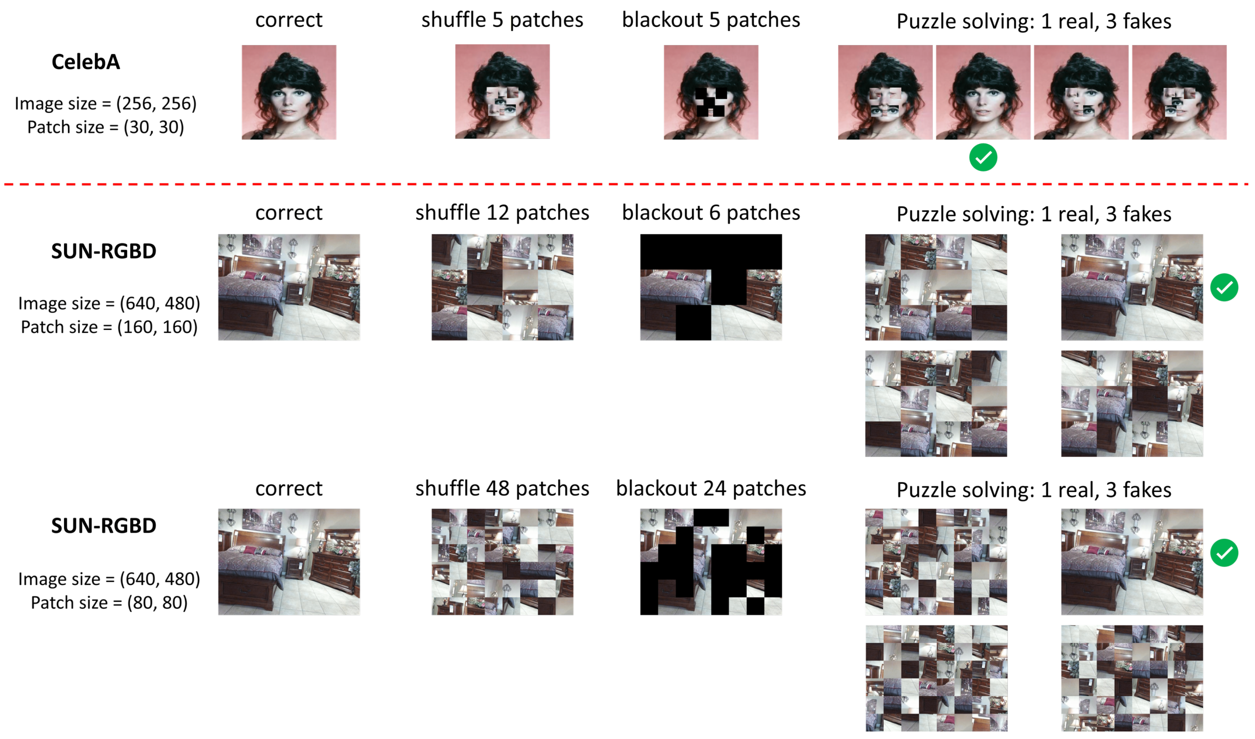


Figure 13: Demo of all types of SSDI corruptions.

Algorithm 1 Patch shuffling

```
1: def shuffleSSDI($tensors$, $num\_patch$, $ps$):
2: $result \leftarrow []$
3: for $it, X$ in enumerate($tensors$) do
4:     $patches \leftarrow$ nn.F.unfold($X$, $ps$, $ps$, 0)
5:     $p \leftarrow patches$.shape[-1]
6:     if $it == 0$ then
7:         $indices \leftarrow$ sample(range($p$), $num\_patch$)
8:         $orig \leftarrow$ tensor(range(p))
9:         $perm \leftarrow$ tensor(range(p))
10:        for $j$ in range(num_patch) do
11:            $perm[indices[j]] = orig[indices[(j+1)\%num\_patch]]$
12:        end for
13:    end if
14:    $patches \leftarrow$ concat($patch$[:, perm]for $patch$ in $patches$)
15:    $X \leftarrow$ nn.F.fold(patches, $X$.shape[-2:],$ps$, $ps$, 0)
16:    $result$.append($X$)
17: end for
18: return $result$
```

Algorithm 2 Patch blackout

```
def blackoutSSDI($tensors$, $num\_patch$, $ps$):
result ← []
for $it, X$ in enumerate($tensors$) do
    $patches$ ← nn.F.unfold($X$, $ps$, $ps$, 0)
    $p$ ← $patches$.shape[-1]
    if $it == 0$ then
        $indices$ ← sample(range($p$), $num\_patch$)
    end if
    for $idx, X$ in enumerate($patches$) do
        $patches$[$idx$][:, $indices$] ← 0
    end for
    $X$ ← nn.F.fold(patches, $X$.shape[-2:],$ps$, $ps$, 0)
    $result$.append($X$)
end for
return $result$
```

Algorithm 3 Create puzzles

```
def puzzleSSDI($tensors$, $num\_perm$, $ps$):
result ← []
for $it, X$ in enumerate($tensors$) do
    $patches$ ← nn.F.unfold($X$, $ps$, $ps$, 0)
    $p$ ← $patches$.shape[-1]
    if $it == 0$ then
        $perms$ ← []
        for $perm\_id$ in range($num\_perm$) do
            $perms$.append(randperm($p$))
        end for
    end if
    $res$ ← $X$.clone()
    for $perm$ in $perms$ do
        $new\_patches$ ← concat($patch$[:, $perm$]for $patch$ in $patches$)
        $new\_X$ ← nn.F.fold(patches, $X$.shape[-2:],$ps$, $ps$, 0)
        $res$ ← concat($res$, $new\_X$)
    end for
    $result$.append($res$)
end for
return $result$
```

C Extra quantitative results

In Table 3 and Table 6 we add more results for the detection task of SSDI corruptions, on CelebA (7 classes) and SUN-RGBD (13 classes), respectively.

CelebA The segmentation model is ResNet18+FPN. The input images have sizes 256 × 256. All results are reported using the first grammar validation method, which relies on the combination of using bi-LSTM and comparing the predicted part semantics to the estimated part semantics within the image itself (i.e. without the memorized average part semantics). From Table 3, we observe that the proposed approach exhibits reliable performance when the size of the corrupted patch exceeds 30 × 30: >80% on patch shuffling SSDI, 92.51% on puzzle solving SSDI with 4 permutations, >80% on patch blackout SSDI.

Table 3: SSDI detection performance on CelebA with 256x256 sized images.

SSDI Corruption: patch shuffling					
Test Accuracy (%)	Patch size: 10	Patch size: 20	Patch size: 30	Patch size: 40	Patch size:50
shuffle 2 patches	53.86 (75.58)	65.66 (68.67)	76.22 (79.72)	83.89 (86.33)	88.22 (93.04)
shuffle 3 patches	55.67 (75.16)	73.16 (78.97)	85.68 (91.08)	90.96 (95.22)	93.05 (97.42)
shuffle 4 patches	57.33 (74.73)	78.40 (82.29)	89.79 (94.57)	93.00 (97.34)	94.34 (98.07)
shuffle 5 patches	59.13 (70.93)	82.88 (88.15)	91.72 (96.06)	93.93 (97.25)	94.69 (97.39)
SSDI Corruption: patch puzzle					
Test Accuracy (%)	Patch size: 10	Patch size: 20	Patch size: 30	Patch size: 40	Patch size:50
1 real, 3 fake	57.84	86.74	92.51	93.97	94.81
1 real, 119 fake (all perm)	8.39	37.43	66.15	80.37	87.23
SSDI Corruption: patch blackout					
Test Accuracy (%)	Patch size: 10	Patch size: 20	Patch size: 30	Patch size: 40	Patch size:50
blacken 1 patch	54.73 (55.36)	70.91 (67.04)	81.25 (81.08)	87.37 (88.04)	91.25 (95.12)
blacken 2 patches	60.43 (63.78)	83.83 (85.22)	91.77 (93.43)	95.41 (97.41)	95.78 (97.58)
blacken 3 patches	65.79 (68.94)	90.60 (93.83)	96.48 (98.49)	97.83 (98.57)	98.50 (99.14)
blacken 4 patches	71.85 (74.34)	94.56 (97.14)	98.08 (98.88)	98.86 (99.36)	99.05 (99.49)
blacken 5 patches	77.32 (85.28)	97.22 (98.20)	98.87 (99.48)	98.95 (99.29)	99.13 (99.64)

SUN-RGBD The segmentation model is ResNet50 encoder-decoder (RedNet). The input images have sizes 640 $\times$ 480. All results are reported using the first grammar validation method, which relies on the combination of using bi-LSTM and comparing the predicted part semantics to the estimated part semantics within the image itself (i.e. without the memorized average part semantics). Two segmentation granularities are used, 13-class coarse segmentation merged from 37-classes, and the original 37-class fine segmentation. From Table 6, we observe SSDI detection performance for the two levels of granularity. For patch shuffling SSDI and puzzle solving SSDI, using coarser 13-class semantic masks boosts syntax sequence reasoning and improves grammar validation performance. This is due to the better sequential reasoning capability that the bi-LSTM learns from coarse patch masks. Meanwhile, in the task of patch blackout, using finer 37-class semantic masks has an edge. The blackened patch semantics is easier to discern when the segmentation is finer, leading to a longer syntax sequence and higher variations in the predicted semantic values.

D Extra qualitative results

D.1 Semantic clustering and segmentation

Figure 14 and Figure 15 show selected part semantics obtained on CelebA face images and SUN-RGBD room scene images, respectively. All samples are from the train set. The displayed part semantics masks are generated by trained ResNet18 and trained ResNet50 encoder-decoder, respectively.

D.2 Syntactic sequences of part semantics

Figure 16 shows the pixel-level part semantics masks and the distribution of part semantics in each mask, averaged over correct samples in the entire CelebA test set. During training, we optimized MSE loss between bi-LSTM predictions and the actual semantics inside each patch, in order to model the mean of the part semantics distribution in each patch iteration. Hence, here we show the average traversal patterns learned. It can be seen that face-part semantics transitions and the face syntax in the CelebA dataset are captured by the trained bi-LSTM model. For example, in the first 2 patch iterations, the “forehead” semantics is dominant. In the third and fourth patches, “mouth” semantics is dominant. While for the last patch, “eyes” semantics is dominant. These part semantics along with transitions of those semantics is key to identifying the presence of SSDI.

Table 4: 13 segmentation classes

SSDI Corruption: patch shuffling			
Test Accuracy (%)	Patch size: 160	Test Accuracy (%)	Patch size: 80
shuffle 4 patches	60.57 (72.04)	shuffle 16 patches	76.57 (73.37)
shuffle 8 patches	69.31 (72.59)	shuffle 32 patches	86.63 (82.38)
shuffle 12 patches	73.47 (77.50)	shuffle 48 patches	87.09 (82.97)
SSDI Corruption: patch puzzle			
Test Accuracy (%)	Patch size: 160		Patch Size: 80
1 real, 3 fake	72.89		91.17
1 real, 99 fake	28.95		69.67
SSDI Corruption: patch blackout			
Test Accuracy (%)	Patch size: 160	Test Accuracy (%)	Patch size: 80
blacken 4 patches	66.55 (67.21)	blacken 16 patches	67.43 (65.53)
blacken 8 patches	66.59 (67.17)	blacken 32 patches	75.68 (70.22)
blacken 12 patches	66.75 (67.49)	blacken 48 patches	73.66 (74.81)

Table 5: 37 segmentation classes

SSDI Corruption: patch shuffling			
Test Accuracy (%)	Patch size: 160	Test Accuracy (%)	Patch size: 80
shuffle 4 patches	62.32 (74.57)	shuffle 16 patches	67.74 (74.77)
shuffle 8 patches	71.53 (76.99)	shuffle 32 patches	78.04 (76.20)
shuffle 12 patches	74.93 (77.43)	shuffle 48 patches	79.84 (78.69)
SSDI Corruption: patch puzzle			
Test Accuracy (%)	Patch size: 160		Patch Size: 80
1 real, 3 fake	76.10		81.23
1 real, 99 fake	33.85		55.99
SSDI Corruption: patch blackout			
Test Accuracy (%)	Patch size: 160	Test Accuracy (%)	Patch size: 80
blacken 4 patches	61.58 (71.37)	blacken 16 patches	61.41 (71.25)
blacken 8 patches	67.53 (72.20)	blacken 32 patches	69.07 (71.80)
blacken 12 patches	66.18 (76.00)	blacken 48 patches	64.59 (72.24)

Table 6: SSDI detection performance on SUN-RGBD with 640x480 sized images.

Figure 17 shows the pixel-level part semantics masks and the distribution of part semantics in each mask, averaged over corect samples in the entire SUN-RGBD test set. Here, each of the 13 semantic classes is either a stuff (e.g., ceiling, floor, wall, etc.) or a thing (e.g., books, char, table, etc.). Within the traversal sequence, the first few patches at the top of the image have "wall" as the dominant part semantics, while the last few patches at the bottom have "floor" as the dominant part semantics. Semantic classes belonging to the thing category reside in the middle patches, such as "chair" and "table". This reveals that our proposed

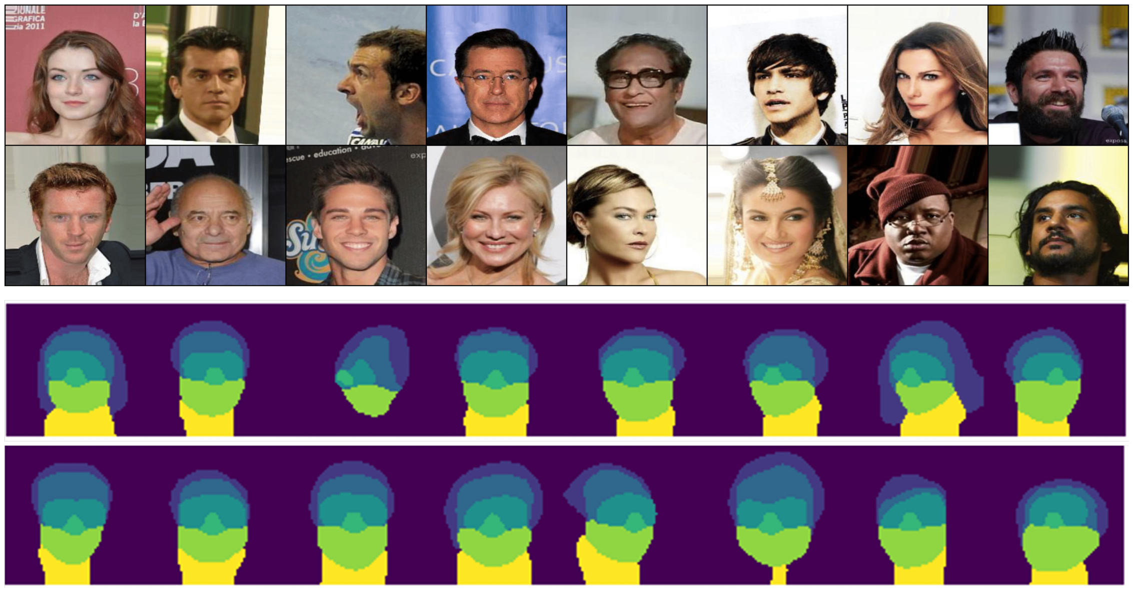


Figure 14: 7-class semantic segmentation on CelebA

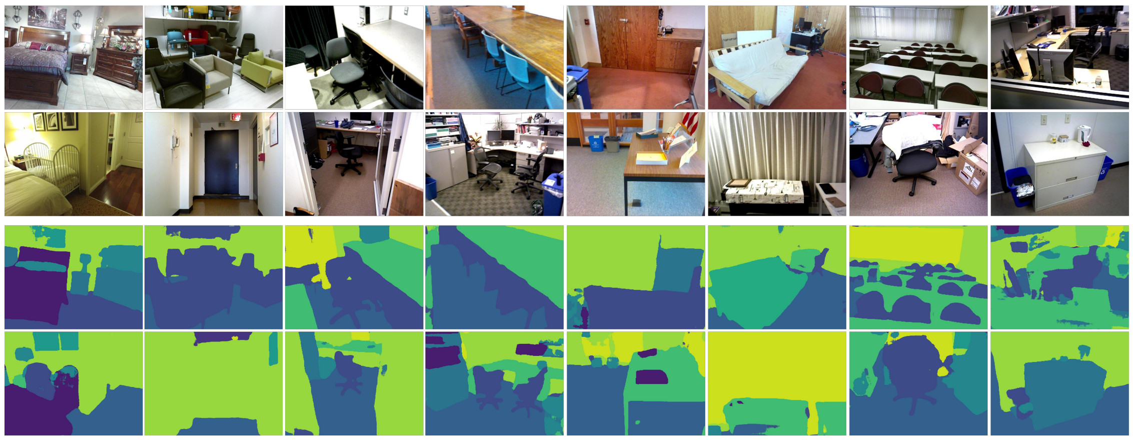


Figure 15: 13-class semantic segmentation on SUN-RGBD

framework is capable of learning the syntax of the scene-centric images containing stuff and thing layouts even from a diverse SUN-RGBD dataset with a wide variety of different room scenes.

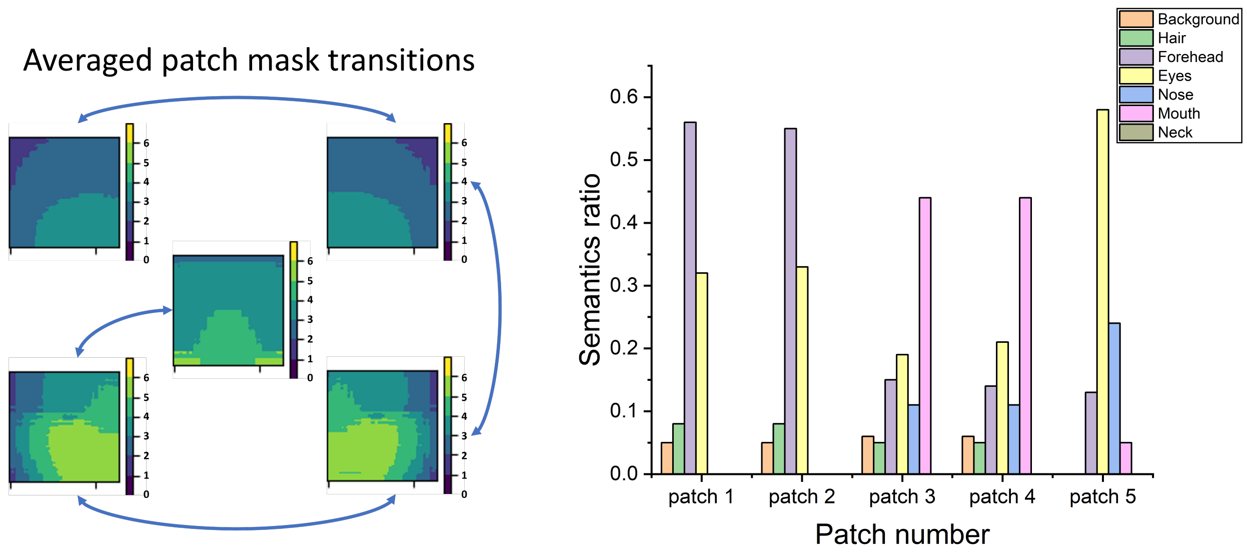


Figure 16: Left: Averaged patch mask transitions over CelebA test set; Right: Averaged patch semantics over CelebA test set

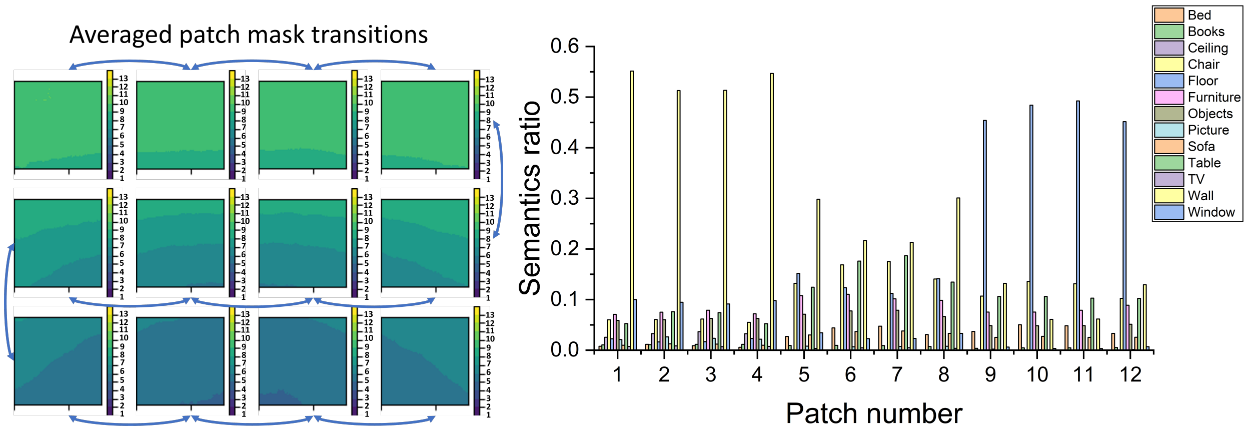


Figure 17: Left: Averaged patch mask transitions over SUN-RGBD test set; Right: Averaged patch semantics over SUN-RGBD test set