SeeBel: **See**ing is **Bel**ieving

Sourajit Saha, Shubhashis Roy Dipta, Budhini Amaraneni,
Shalima Binta Manir, Vishnu Gokanakonda and Varun Kunde

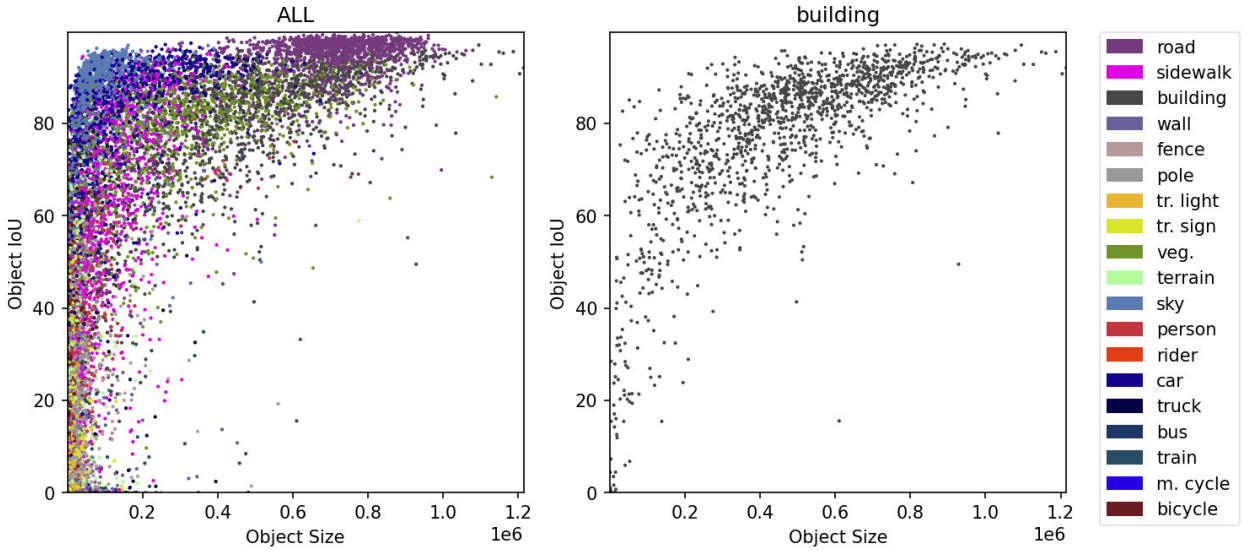


Fig. 1: Correlation between object size and object performance for different object categories in cityscapes dataset [1] with an overview-detail view. In overview view, all object categories are shown, whereas in the detail view, specific object is shown.

Abstract—Semantic Segmentation is a significant research field in Computer Vision. Despite being a widely studied subject area, there do not exist many visualization tools that capture segmentation quality and dataset statistics such as a class imbalance in the same view. While the significance of discovering and introspecting the correlation between dataset statistics and AI model performance for dense prediction computer vision tasks such as semantic segmentation is well established in the computer vision literature, to the best of our knowledge, no visualization tools have been proposed to view and analyze the aforementioned tasks. Our project aims to bridge this gap by proposing three visualizations that enable users to compare dataset statistics and AI performance for segmenting all images, a single image in the dataset, explore the AI model's attention on image regions once trained and browse the quality of masks predicted by AI for any selected (by user) number of objects under the same tool. Our project tries to further increase the interpretability of the trained AI model for segmentation by visualizing its image attention weights. For visualization, we use Scatterplot and Heatmap to encode correlation and features, respectively. Surveys on real users have shown that our visualization helps the target users better understand their AI model and dataset. The full system can be accessed at `https://github.com/dipta007/SeeBel`.

Index Terms—Data Visualization, Computer Vision, Semantic Segmentation, Machine Learning, Explainable AI

1 Introduction

In this project, we study how Data Visualization can aid machine learning systems users (researchers, engineers, practitioners) in analyzing the effects of class imbalance on semantic segmentation performance. Semantic segmentation is a high-level computer vision task where each pixel is assigned a class label delineating the object type (e.g., car, person, etc.) present in that pixel. Image datasets that are usually used to train machine learning models to perform semantic segmentation are imbalanced in terms of class distribution [1–8]. Moreover, class imbalance can negatively impact [9] the performance of semantic segmentation systems. Therefore, investigating how class imbalance and the performance of semantic segmentation models are correlated is a significant research topic that requires investigation. The dataset that we use is Cityscapes [1] dataset. Our proposed visualization aims to visually assist machine learning systems users in inspecting the aforementioned correlation. Our proposed visualization consists of three key tasks:

1.1 User Discovers Correlation

For this task, the user aims to discover correlation between class distribution and performance for semantic segmentation. We aim to design this visualization so that users can visually inspect prediction scores (by the trained model) and object size and compare these two attributes for all object classes in a given dataset. This task can potentially help users gain insights into how class distribution and AI performance relate to each other.

1.2 User Locate/Explore/Browse Features

For this task, the user aims to locate, explore, or browse the model's attention weights across the entire image for each class, depending on the user's requirement. When segmenting an object, a trained AI model gives different amounts of attention to different portions of the image. With our visualization, the user can explore, browse, and locate through the image to find the amount of attention by AI at different image locations. Through this visualization, users can potentially gain

• *All authors are with the Department of Computer Science & Electrical Engineering at the University of Maryland, Baltimore County.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

insights into the AI model's behavior while predicting different objects.

1.3 User Locate/Browse/Explore Distributions

For this task, the user's goal is to locate, browse or explore different object categories at different image regions as a trained AI model segments an image. This task will help the user visually/qualitatively inspect the segmentation quality of the model, and we propose an interaction for this task that allows users to bring all or any number of objects into view. Another objective of this task aims to help the user discover a quantitative correlation between object size and the confidence of the AI model to predict that object for a set of object categories that users can select. While the aim of the first task (described in section) is to discover a correlation between object size and performance quantitatively for all images in the entire dataset, this task, however, does the same (both qualitatively and quantitatively) but for a single image.

1.4 The problem potentially solved with our proposed visualization

Our proposed visualization can potentially become a helping tool for researchers who train semantic segmentation models and individuals who are responsible for policy-making to deploy these systems in real-world applications. Semantic segmentation datasets are often imbalanced, and different techniques such as weighted loss function [10] and resampling data [11] are usually used to mitigate these challenges. However, all of these techniques require an understanding of the dataset's statistics. These statistics include the distribution of classes across a number of images (the number of images containing some object may be different than the number of images containing some other object), distribution of classes across the amount of pixel area (amount of pixel area containing some object may be different than the amount of pixel area containing some other object). With the knowledge of class distribution, the user can then design a loss function by assigning higher weights to the rare classes (low frequency, small object size) and lower weights to the abundant classes (high frequency, larger object size) before training the machine learning models. Sometimes, the aforementioned loss function's weights require an adaptive update. To understand in which direction an update is required (if at all), it is essential to know the classes' performance. And this knowledge can guide the user to change loss weights adaptively and learning rates or even to re-initiate training with a different approach if the performance for rare classes is still not improved, which can reduce both time, cost, and carbon footprint since the training of large machine learning models is expensive, time-consuming and power hungry. Our visualization, thus, can help the user depict these statistics, both before initiating the training and after training.

1.5 Domain & Typical Users

The domain of our visualization is Machine Learning & Computer Vision. Hence, some of the pronounced users of our visualization are Machine Learning (ML) and Computer Vision (CV) practitioners/researchers who might want to discover correlation between dataset statistics and performance. Another typical user is self-driving vehicle manufacturers who might want to discover how their trained AI model is doing for some crucial objects, i.e., roads, pedestrians, and traffic lights. The medical imaging community can be another set of typical users who might find it useful to discover the effect of dataset statistics on training, as medical data is often highly imbalanced.

1.6 Our Contribution

Our contributions are the following:

- Our first visualization (detailed in section 4.1) helps the user discover correlation between class distributions and performance for each class.
- Our second visualization (detailed in section 4.2) allows the user to explore the attention weights of trained Semantic Segmentation models (AI) on images while predicting particular objects.
- Our last visualization (detailed in section 4.3) helps the user quantitatively discover correlations between object size and performance, both qualitatively and quantitatively for any single image and any number of object categories.
- Our visualizations can potentially help the users better understand their machine learning(AI) models and the effect of dataset statistics on training, contributing to enhancement in AI interpretability and building robust AI models.
- To the best of our knowledge, research efforts that are invested in building robust AI models do not typically use visualization tools to view dataset statistics and AI performance in a unified platform. While it is realized that dataset statistics and AI performance are closely related, a **one-stop** visualization tool to uncover that closeness is of sheer research significance.

2 Related Works

Some studies discovered the distribution of different classes in the cityscapes dataset in terms of the number of pixels occupied by each class [1, 12], the proportion of labeled pixels for each class [1], number of images for each class [1], etc. These studies, however, are limited to analyzing (discovering) class distribution alone and do not compare any correlation between class distribution and performance. On the other hand, some studies - to depict the efficacy of their respective machine learning models - have reported [13, 14] a detailed overview of performance for each class trained on various semantic segmentation datasets. Yet, they do not establish any connection between performance and class distribution. To the best of our knowledge, our visualization is the first to address a correlation between the aforementioned attributes.

Cost Curves [15] is an essential visualization method to uncover machine learning model performance over a range of class distributions. The idiom for the cost curve is a line plot. The authors plot the probability cost of a wide range of sampling distributions over a dataset and their corresponding normalized expected cost. Moreover, to encode data, the authors have used horizontal and vertical spatial positions, connection lines, point marks, and shape channels for the user to find trends and correlations. However, generalizing the cost curve method to a semantic segmentation model is computationally (both memory and time) expensive since (1) cityscapes have 19 trainable classes, and the cost curve method uses only two classes to sample a wide range of distribution samples and (2) the complexity of a segmentation model is much higher than that of a classification model because, in the classification model, the output is a single/multi-label whereas, in semantic segmentation models, each pixel of the image needs to be classified to a single class over a set of candidate classes. Furthermore, the cost curve is not easily interpretable from a visualization perspective. It takes domain-specific knowledge on the user's end and added cognitive load to compare a correlation between a class distribution and model performance. Therefore, we only propose to report model performance and class distributions of the dataset using Scatter plots to discover their correlation in our project.

In Natural Language Processing (NLP), attention-based models have achieved state-of-the-art results in different domains of NLP tasks. But the interpretability of how they work remains a mystery. In this work [16], the authors have used different idioms to increase the interpretability of attention-based models. The authors have provided three interactive idioms. In the attention-head view, authors have explored the self-attention weights for one or multiple heads for each input. They have used color hue to express the weight and line for word connections. Subsequently, in the model view, the authors have presented an overview-detail view with color hue as categorical layers and lines as word connections in a detailed view. And in the neuron view, authors have drawn a simple line connection idiom to show the individual neuron weights with respect to the query, key, and value. All of the views are interactive, with the option to choose layers and heads and zoom in on the individual head. Even though our proposal domain (computer vision) differs from this work's, the interpretability of their visualization has inspired us to study similar analogies for Computer Vision.

Another study [17] was recently conducted, which proposed radial bar chart-based visualization of classification performance. They used the radial positions to represent model performance and the color and angular positions to represent classes. The authors further show how they draw different numbers of samples from a dataset and report the error rate for all of them after certain iterations. However, using such a design for our purpose will be cumbersome as we have more classes in cityscapes, and discovering trends from different figures (since it requires one visualization per iteration in this design mechanism) will increase the cognitive load on the user's part. Due to the inefficiency of this proposed idiom, we have extensively used scatter plots in our visualization, which can show more points with less cognitive load.

A subsequent study [18] proposes a visualization that depicts the internal information processed by an autonomous vehicle so that the user, with a reduced cognitive load, can access that information and decide whether or not to trust the vehicle's (AI) next set of steps and take control of the vehicle to themselves. It allows the users to perceive the vehicle's detection capabilities, and the authors processed the semantic segmentation of road scene images for image classification based on pixels. They use color hue for categorical attributes to show the color distributions of the object visualization. Additionally, different shapes are used as identity channels to show dynamic and static objects. Furthermore, box plots and violin plots are used to show the distribution of data points. More specifically, violin plots are used to show the perceived ability to detect dynamic and static objects and the system's ranking. Whereas this visualization can successfully depict the information of autonomous vehicles, it has a limitation on the number of bins. Our dataset [1] has 19 classes, and a scatter plot is a more efficient choice than using a box plot.

In a later study [19], the authors have proposed to learn insights into the training dataset from the corresponding classification results. Furthermore, this study aims to find how classification performance is related to class distribution by visually inspecting how far a predicted class is from the ground truth class in the class map space. The authors further show how they use scatter plots as their choice of idiom. The horizontal spatial position denotes class distance, and the vertical axis spatial position denotes the probability that the model assigns to alternative classes. One of the major issues with this visualization is that it needs to be repeated for each class; therefore, using this mechanism to visualize the semantic segmentation performance for all the classes (19 of them) will be challenging, and discovering trends from multiple idioms will pose added cognitive load on the user's part.

InstanceFlow [20] is a sophisticated visualization tool that allows users to compare learning behavior between iterations on an instance level. They use a Sankey diagram with glyphs to depict a temporal analysis of the training process. They have used color to show the importance of the flow. With their proposed visualization, the authors offer a unified solution to discover trends between performance at different iterations at the expense of a complicated visualization and added cognitive load. We aim to allow the user to discover the exact correlation with a much simpler visualization (Scatter plot, images, and superimposed layer) with lesser strains on cognition.

3 Implementation

3.1 Data

The cityscapes [1] dataset has 5000 images with an identical resolution of 1024 × 2048 pixels. The dataset further contains masks for different objects in each image, and all of the images are taken while driving in different cities in Germany. Cityscapes contains 2975 training images, 500 validation images, and 1525 test images, contributing to a total of 5000 images. Moreover, the 1525 test images do not contain annotated masks. Those labels are not made public and are only used internally to score models submitted to the Cityscapes leaderboard [1]. And therefore, we only use the training images for training our AI model and all relevant visualizations throughout the entirety of this project. The annotated masks provided in the dataset account for 30 classes. Although only 19 classes are used for evaluation, and therefore all of our experiments, including the pre-processing of the data and the corresponding visualizations, will contain 19 classes.

[1] https://www.cityscapes-dataset.com/benchmarks/

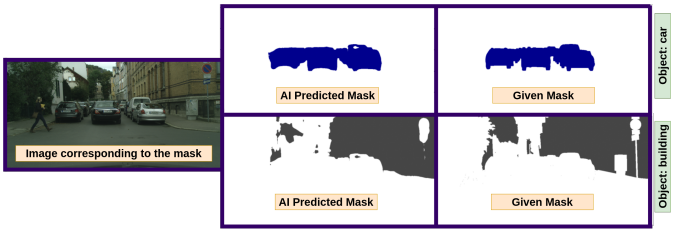


Fig. 2: Input image along with it's given, and AI predicted masks for object category: car, building.

The semantics of this dataset are street scene images captured from vehicles. For every image in the dataset, there exists a set of corresponding masks that encompasses the objects present in that image and the position (pixel location) and category (i.e., car, bus, traffic light, etc.) of the object. Another way to think of this data is that all the pixels in an image contain an object of some kind (people, car, road, etc.). The dataset uses color hue (different color hues to represent different objects) for every pixel to describe what kind of object is present in that pixel.

3.2 Data Transformation

The cityscapes dataset contains images (RGB images of scenes) and corresponding mask images. The dataset type for Cityscapes is geometry since images convey information about items with explicit spatial grid positions on a 2D surface. Therefore, the data type for cityscapes is items and spatial positions. One of the limitations of the Cityscapes dataset is the abundance of background classes meaning only a few objects of interest are labeled, leaving a lot of pixel space unlabeled (background class). Moreover, since we visualize the performance on the cityscapes dataset, one of the challenges in getting good performance on this dataset is the limited number of labeled training images [3, 4]. Moreover, to denote the performance of AI models, we use the IoU (Intersection over Union) metric that signifies the overlap between the given mask and predicted mask for an object category in an image. The ideal range of IoU is $[0, 1]$ [21] however, for semantic segmentation IoU is traditionally [5, 7, 22–24] expressed as a percentage and therefore the range becomes $[0, 100]$ (multiplied by 100).

Now, there are multiple stages of transformation that we need to perform on the data. The first stage of transformation is task agnostic, and the second stage is task specific.

3.3 Task Agnostic Data Transformation

Firstly, since we need to get our data ready to be trained, we need to Resize the images from the original resolution (1024 x 2048) to 512 x 1024 pixels to match the computational capacity of our training hardware (NVIDIA RTX 3090 GPU). We use this image size for the remainder of this project in all of our experiments and analysis. Therefore, all the visualizations that use object size depend on the resized image rather than the original image. Training the AI model on the dataset is a significant data transformation process for our project. Because once trained, we store the trained model to perform prediction later. Specifically, we have trained a Convolutional Neural Network based AI model called HRNet [22] on the cityscapes dataset. Furthermore, the hyper-parameters we have used for training are identical to this study [22]. Now, we perform these transformations because prediction masks are some of the attributes in our proposed visualizations, as described in section 1.3, 1.2, and 1.3.

3.4 Task Specific Data Transformation for Task 1

Each image contains multiple objects (e.g., car, bus, road, etc.) and multiple masks. Consequently, a *one-to-many* mapping exists between images and masks. Figure 2 depicts two of many masks (mask of object category: car and building) for an image in the cityscapes dataset. To

discover a correlation between object size and object IoU for different category classes, we transform the original dataset into a table using Algorithm 1. We iterate over all the masks ($m \in M$) in the original cityscapes dataset where M is the set of all masks. Then, for each mask ($\forall m \in M$), we run the corresponding image, i_m, through an AI model (AI) to compute prediction mask p. Sequentially, we compute the IoU score (IoU) between each prediction p and given mask m, object size of the given mask m ($Size_m$), and object type of the given mask m ($Object_m$) to construct each row of the table.

Algorithm 1 Data transformation to construct a table from the original image dataset for the user to discover correlation among object size and object IoU for different object classes.

```
Table = []
for m ∈ M do                          ▷ M is the set of all masks in the dataset
    p ← AI(i_m)                       ▷ Compute AI prediction
                                      ▷ for current mask m
    IoU ← Score_IoU(p, m)             ▷ Compute IoU between AI prediction
                                      ▷ mask and current mask
    Size_m ← Area_pixel(m)            ▷ Compute pixel area
                                      ▷ for current mask
    Object_m ← ObjectType(m)          ▷ Extract object class of
                                      ▷ the current mask
    Table.add(Object_m, IoU_m, Size_m) ▷ Add new mask to the table
end for
return Table
```

3.5 Task Specific Data Transformation for Task 2

To visualize the attention of the AI model for segmenting different objects, we utilize Grad-cam [25], an Explainable AI method. Grad-cam produces an image that shows the area of the input image to which the model is paying more attention than others which is referred to as grad-cam weights. We transform the images (given in the dataset) to a superimposed view where the original images are superimposed with a grad-cam weight. The grad-cam algorithms must be called to retrieve AI attention weights to predict a particular object type. Now, the grad-cam algorithm requires a trained AI model (AI), image (i), and object category (c). We denote the grad-cam algorithm as G and the grad-cam weight, $w = G(i, AI, c)$. Precisely, we repeat this process for all possible (i, c) pairs over the entire dataset $\forall i \in I$ and $\forall c \in C$. Here I is the set of all images in the dataset, and C is the set of all object categories in the dataset. Finally, to superimpose image i and grad-cam weight w together, we first display the original image i with opacity α_1. Then we display the grad-cam weight w with opacity α_2 to get the superimposed view S. This sequential process is described in Equation 1. Here, $\alpha_1 > \alpha_2$ so that the original image is visually accessible behind the grad-cam weight in the superimposed view.

$$S(i, w) = \{Display(i, \alpha_1), Display(w, \alpha_2)\} \quad (1)$$

3.6 Task Specific Data Transformation for Task 3

We transform the images in the dataset into a table for this task with the help of the trained AI model (AI). We first run each image $i \in I$ in the dataset (where I is the set of all images) through the trained AI model to get the predicted mask $p = AI(i)$. Then, we extract the given mask m corresponding to image i. From both m and p we compute the number of pixels occupied by each object present in the corresponding masks. From the pixel count for each object category, we normalize these count values in the $[0, 100]$ range to express these pixel counts in pixel occupancy percentage. Sequentially, we construct a table where each row is the current object's category in the image, the object's pixel occupancy in the given mask, and pixel occupancy in the predicted mask.

3.7 Programming Language & Visualization Tool

We use Python3.7 [26] as our base programming language. We use Python Image Library (PIL) [27] to manipulate and display images in our visualization. For the visualization itself, we use matplotlib [28], GRAD-Cam [29], and Jupyter Widgets [30] for the interaction in our visualizations. Also, we used PyTorch [31] to train the machine learning model to predict the labels and mask. Also, for data manipulation, we used pandas [32], chardet [33], and einops [34].

3.8 Software Architecture

We use conda [35] and pip [36] package manager to install all of our dependencies for the code environment. We use JupyterLab [37] to run, host, and interact with the plotting. In our early alpha and beta release experiments, we tried other tools, i.e., seaborn [38], tableau [39], but we switched to JupyterLab because we required more control over manipulating visualization, data transformation, and computational efficiency.

4 Results

4.1 Task 1: Discovering Correlation Between Object Size and Object IoU for Different Category Classes

4.1.1 Proposed Artifact

The goal of users while using our proposed visualization tool (Figure 3) here is to discover a correlation between object size and IoU for different category classes. Our proposed artifact uses two scatter plots with an overview/detail view, shown in Figure 3. While our tool shows a scatter plot of object size and IoU for all object classes on the left (overview), the user, with the help of our dropdown menu, can select a particular object class and the scatter plot for the selected object category is then displayed on the right (detail). Figure 3 shows a snippet of how users can interactively use our tool for the aforementioned task.

4.1.2 Data Abstraction

The dataset type for this task is a 2D table with items and attributes as data types. In the aforementioned table, each item (row) is an object mask in the dataset. And there are three attributes (columns): (1) categorical attribute: object type (e.g., road, person), (2) ordinal attribute: object IoU (e.g., 55 %, 92 %), and (2) ordinal attribute: object size (number of pixels occupied). We use a horizontal position channel to encode object size, a vertical position channel to encode object IoU, and a color-hue channel to encode object category (class). Finally, we use a point mark for this idiom.

4.1.3 Task Abstraction

The user task is to **discover** a **correlation** between object size and object IoU for different category classes.

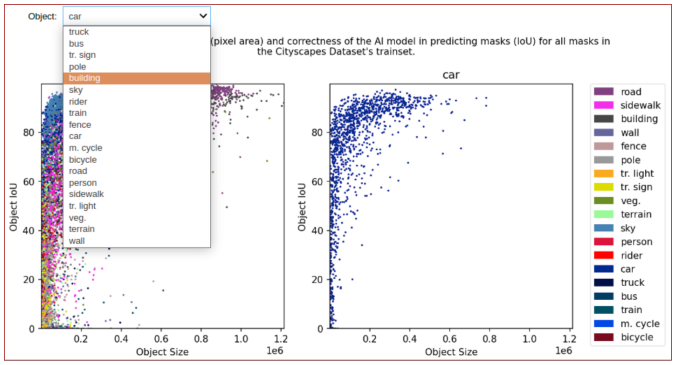
4.1.4 Encoding and Interaction Idiom

For visual encoding, we use a scatter plot. Furthermore, we use overview/detail interaction with the same encoding but with a subset of the data. Finally, the dropdown menu we use to choose object category is another form of interaction that falls under the select category. In the initial overview, our tool shows an overview scatter plot of object size and IoU for all object categories and a detail-view scatter plot of object size and IoU for the car object category (Figure 3).

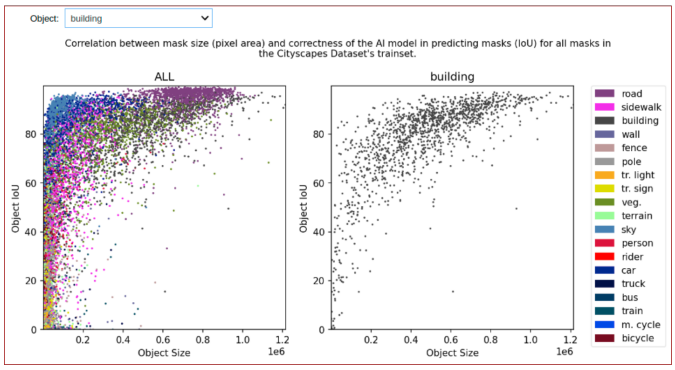
4.1.5 Critical Analysis

In *overview* mode, our proposed tool (Figure 3) is supposed to give a sense of correlation between object size and IoU across the whole dataset for all object categories to the user. However, it is difficult to track objects in *overview* mode simply because there are too many masks in the dataset and, therefore, too many data points and colors in the scatter plot, as shown in Figure 3. However, in *detail* mode, it is much easier for the user to discover correlation because there is only a single object in the view, and the user is not shown data points for other object categories. Also, showing a scatter plot for a single object eliminates the need for color to denote object category, reducing a channel.

Figure 3, 4 further suggests that correlation can be observed using our proposed tool. For example, it can be observed (Figure 4) that for certain objects (e.g., road, train), there is no obvious correlation



Step 1: From the dropdown button user can select object class



Step 2: Once a object is chosen, a overview/detail is created for the selected object class on the right, whereas on the left encoding shows data for all object classes in the dataset.

Fig. 3: Snippet of our proposed tool for Task (section 4.1) from jupyter notebook. Top: a screen capture of how users can interact with the tool. The users can select an object class from the dropdown menu. At the initial condition when the user is yet to interact with the tool, on the left is displayed scatterplot for all objects (*overview*), and on the right is displayed the scatterplot for car object (*detail*). Down: screen capture of when the user selects building; on the left is displayed scatterplot for all objects (*overview*), and on the right is displayed the scatterplot for building object (*detail*).

followed by all the masks. While for other objects (e.g., buildings), there is a positive correlation (Figure 3).

Interestingly, along with the goal of discovering correlation, it is further observed in Figure 4 that from the density of the scatter plot, the user can also identify data distribution (e.g., there are fewer instances of train objects in the cityscapes dataset in comparison to that of traffic sign). Furthermore, our proposed visualization tool (Figure 3, 4) also helps the user identify other interesting distributions, such as most of the car objects sizes are below an upper limit of 0.6 Million, whereas most of the road object is sized within 0.4 to 1 Million and certain objects such as traffic signs are very tiny in size (within 0.05 Million).

On the other hand, one limitation of our design (Figure 3) is the scalability issues that can arise with increased data because scatter plots are usually best suited for hundreds of data points. Another design limitation is the separability of objects with different color hues in *overview* mode. For example, bus and train are not easily differentiable (Figure 3) with the assigned colors, and as the number of object categories grows, the problem worsens.

Color hue denotes different object categories proposed in the cityscapes dataset [1] are traditionally followed in computer vision literature [23, 24, 40]. Since our target users are vision researchers, we have decided to honor that choice using the same color map used originally in the cityscapes dataset. We hope this decision will partly attract more target users to use our tool. Data transformation performed for this task (described in section 3.4) improves the match between data semantics and the task because discovering correlations between

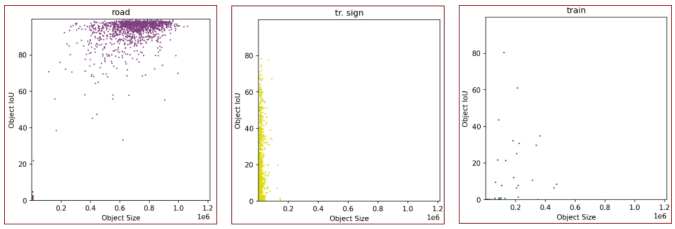


Fig. 4: Cropped screenshot of our proposed tool for Task (section 4.2) from Jupyter notebook. Left: *detail* view of the correlation between object size and IoU for road objects, Middle: *detail* view of the correlation between object size and IoU for traffic sign objects, Right: *detail* view of the correlation between object size and IoU for train objects.

attributes (Object size and IoU) would not have been possible without the transformation. Additionally, we have used a horizontal position channel to denote object size, a vertical position channel to denote object IoU, and a color hue channel to denote object category. Removing any of these channels will disable us from showing all of these three attributes (object type, size, and IoU).

4.2 Task 2: Locating, Browsing and Exploring Features

4.2.1 Proposed Artifact

The goal of users with our proposed visualization tool (Figure 5) for this task is to locate, browse, and explore grad-cam features. Our proposed artifact uses one image and one superimposed layer in a multiform view. As with our current design, the user cannot choose a different image for this tool; therefore, this tool runs with a static image chosen in the background. With the help of the dropdown menu, the user can select object category and colormap. The static RGB image is displayed on the left, and GRAD-Cam weights (superimposed with the static RGB image) are displayed on the right side of the multiform view. Moreover, the user can interact with the superimposed GRAD-CAM layer by hovering with the mouse, and the corresponding GRAD-CAM weight is annotated at the mouse location. Figure 5 shows an overview of our proposed tool for this task. GRAD-CAM [25] weights signify the amount of importance given by the AI model to the image while predicting a certain object category.

4.2.2 Data Abstraction

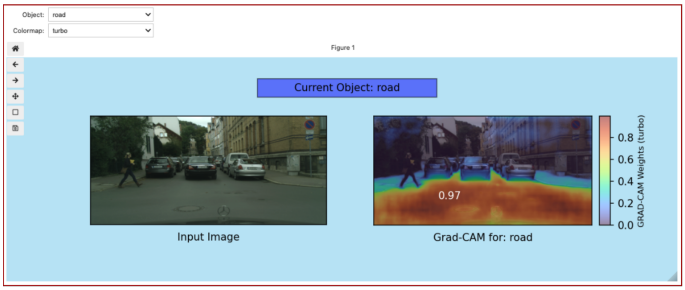
The dataset type for task 2 is Spatial Field, and the data type is grid, position, and attribute. In the grid, each location denotes pixel coordinates while the attributes are pixel values. In the case of RGB images, the data can be viewed as 3 spatial fields where each spatial field's grid positions contain quantitative attributes such as R/G/B color values (different color channels at different spatial fields). There are several channels: (1) horizontal spatial position, (2) vertical spatial position, (3) color hue (when categorical color maps are used with color bins for the grad-cam superimposed layer), and (4) color saturation (when sequential color maps are used for the grad-cam superimposed layer or for showing images). Finally, we use the area as the mark for this idiom.

4.2.3 Task Abstraction

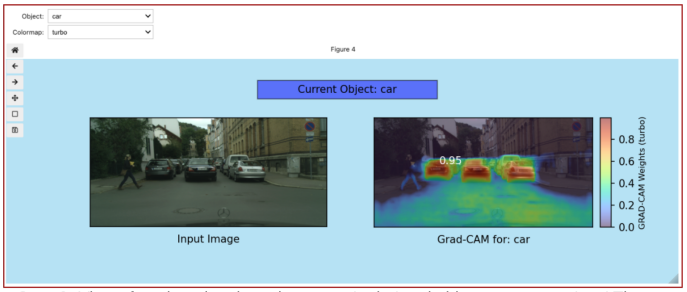
When the user hovers to a particular grad-cam superimposed layer position, their goal is to **browse features** (grad-cam weight). When the user looks at the color bar and wants to find a particular color in the grad-cam superimposed layer, the user wants to **locate feature** (either grad-cam weight or color). And finally, another use-case is when the user hovers through the grad-cam superimposed layer to **explore features** (grad-cam weight) to get a sense of the AI models' attention on different parts of the image.

4.2.4 Encoding and Interaction Idiom

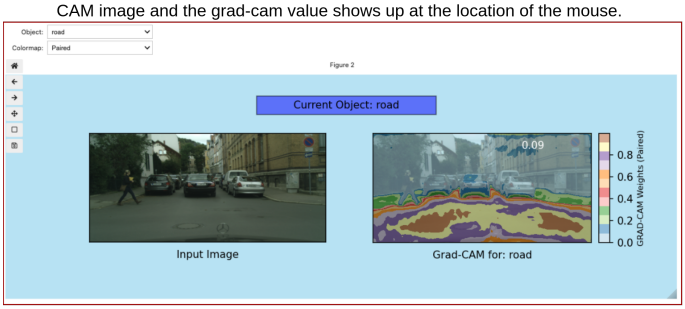
We use a multiform view encoding with multiple interaction choices. As shown in Figure 5, on the left of the multi-form view is an image, and on the right is a superimposed layer. One interaction choice is



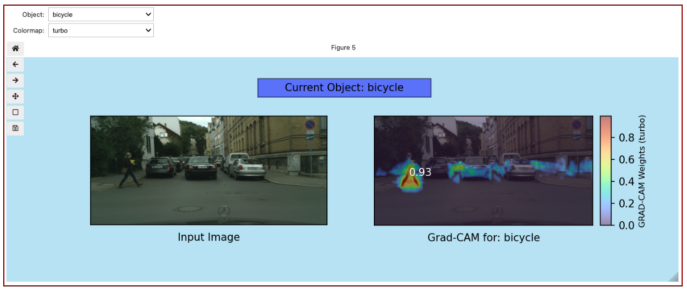
Step 1: View at the initial condition of interaction, the user has the option to select object category and colormap from the dropdown menu. The user hover to the Grad-CAM image and the grad-cam value shows up at the location of the mouse.



Step 3: View after changing the colormap to 'turbo' and object category to 'car.' The user again hovers to the Grad-CAM image and the grad-cam value shows up at the location of the mouse.



Step 2: View after changing the colormap to 'Paired.' The user again hovers to the Grad-CAM image and the grad-cam value shows up at the location of the mouse.



Step 4: View after changing the object category to 'bicycle.' The user again hovers to the Grad-CAM image and the grad-cam value shows up at the location of the mouse.

Fig. 5: Screenshot of our proposed tool for Task (section 4.2) from Jupyter notebook. Top Left: A screen capture of the view where the user hovers over the GRAD-CAM image without interacting with colormap and object choices. Bottom Left: A screen capture of the view where the user only hovers over the GRAD-CAM image and changes the colormap. Top Right: A screen capture of the view where the user hovers over the GRAD-CAM image and changes both the colormap and object category. Bottom Right: A screen capture of the view where the user hovers over the GRAD-CAM image and changes the object category only.

select (mouse hover) over the superimposed layer. Another form of interaction we use is highlighting; at the hovered mouse position of the superimposed layer, we display grad-cam weight at that position. Finally, the dropdown menu we use to choose object categories and colormaps is another form of interaction (select). In the initial overview, our tool shows an image, and it's a superimposed version with grad-cam weights for the road object category in the turbo color map.

4.2.5 Critical Analysis

Firstly, when the user hovers in the superimposed layer of our multiform-view tool, the corresponding grad-cam weight value is displayed (Figure 5). This design is effective because the user - without needing additional interaction with or prompt on the tool - can browse grad-cam attentions at different image regions, and the view (superimposing image with the grad-cam weight) updates when the user hovers to a new location are relatively seamless. Moreover, the user can easily select object categories and colormaps from the dropdown menu to load into the view (Figure 5).

Additionally, if the user attempts to locate a particular color (from the displayed color bar) in the image, they can do so by searching through the entire superimposed layer (Figure 5). Now, how easy or difficult it is for the user to do so is highly contingent on the choice of the colormap. For example, if the user uses sequential color maps such as 'paired', locating a color hue becomes much easier. In Figure 6, it can be observed that locating the second highest ranked color bin on the top right screen capture (categorical color map) is much easier than it is in the bottom left screen capture (sequential color map) where a sequential color map ('turbo') has been used. However, when the user focuses on exploring features which for this task translates to exploring which regions of the image are more important than the other when the AI model tries to find a certain object, sequential colormap ('turbo', 'viridis) are more user friendly (Figure 6). This advantage stems from the fact that the superimposed layer with sequential colormap yields a smoother distribution of color hue and, therefore, gradcam weights, as can be seen in Figure 6. Our recommended colormap for the user to explore features is turbo [41] as it is more perceptually uniform and

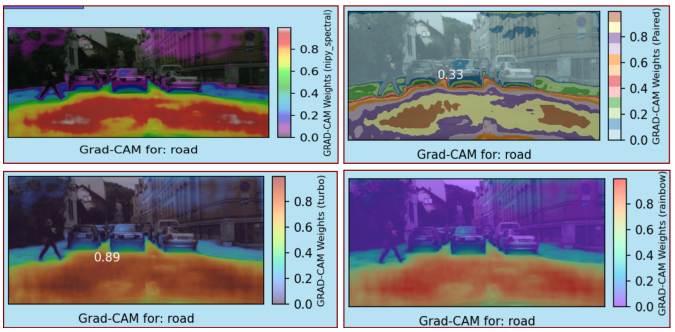


Fig. 6: Cropped screenshot of our proposed tool for Task (section 4.2) from Jupyter notebook. Top Left: GRAD-CAM image with categorical (color-hue) colormap: nippy-spectral. Top Right: GRAD-CAM image with categorical (color-hue) colormap: Paired. Bottom Left: GRAD-CAM image with sequential colormap: Turbo. Bottom Right: GRAD-CAM image with sequential colormap: rainbow.

therefore imposes a lesser cognitive load on the user.

Interestingly, while using our proposed tool for task 2, it can be further observed (Figure 5) that when the AI model tries to predict bicycles, it looks at a portion of the image (legs of a person) that has some level of pseudo-similarity with the shape of bicycles (two rods connecting the wheels). This is a feature explored by decoding our design that may be of interest to the users (computer vision, AI researchers) while using this tool.

One limitation of our current design (Figure 5, 6) for this task is that AI attention weight can be explored/browsed/located for one object category at a time. Though it can be posited that viewing weights for too many objects at a time can be a high-load task, it may as well be desirable for particular users to explore/browse/locate grad-cam features for more than one category at a time. Moreover, running the gradcam algorithm [25] is time-consuming, which is why we have

restricted our tool (Figure 5) to view grad-cam weights for one object category at a time since viewing multiple categories at once requires running the gradcam algorithm multiple times as well.

Data transformation performed for this task (described in section 3.5) improves the match between data semantics and the task because exploring/browsing/locating grad-cam features would not have been possible without the transformation (running grad-cam algorithm on the original data and training AI on the original data). The interactions tool that we use here (Figure 5, 6) is effective because the user can locate grad-cam weights only by hovering. Also, while the user can locate the gradcam feature with the help of a color map and color bar, the hover gives the user the additional ability to locate grad-cam features more easily as the user does not have to depend on the color map while hovering (Figure 6). Moreover, we have used the horizontal spatial position channel, vertical spatial position channel to denote each image pixel, and the color-hue/color-saturation (depending on the chosen color map) channel is used to show (Figure 5) the image and grad-cam weight (superimposed with image). Therefore, eliminating any of these channels won't allow us to show the aforementioned images and superimposed layers, which corroborates our hypothesis of "no redundant marks and channels".

4.3 Task 3: Discovering Correlation Between Object Size and Object IoU for Different Category Classes

4.3.1 Proposed Artifact

The goal of users while using this proposed tool (Figure 7) is to locate, browse, and explore object categories in segmented images from a given image in the dataset. In our multiform view, we first display the input image, the corresponding given mask, and AI predicted mask. Additionally, this tool also provides (with three scatter plots) the user the ability to discover correlation among three pairs of attributes: (1) given and predicted pixel occupancy, (2) IoU and given pixel occupancy, and (3) IoU and predicted pixel occupancy for a given image, it's mask and AI predicted mask. We have used check boxes for all object categories in the cityscapes dataset. The user can select or deselect any of them at any given time. The information (Image, given mask, predicted mask, 3 scatter plots) is updated into the view (multiform) according to the objects selected.

4.3.2 Data Abstraction

The dataset type for this task is a spatial field (images and masks) and table (computed IoU, pixel occupancy for different objects in an image). While the data type for images and masks is grid, position, and attribute, the data types for IoU and pixel occupancy for different objects are items and attributes. In the table, each item is an object mask in the chosen image. And there are three attributes: (1) categorical attribute: object type, (2) ordinal attribute: object IoU, and (2) ordinal attribute: object size. We use a point mark and one horizontal position channel to encode pixel occupancy, a vertical position channel to encode object IoU in one view and pixel occupancy in another, and a color-hue channel to encode object category (class). For the images and masks, there are four channels: (1) horizontal spatial position, (2) vertical spatial position, (3) color saturation (while showing images), and (4) color hue (while showing masks). The mark used in this visualization is area.

4.3.3 Task Abstraction

When the user hovers over a given or predicted mask, the goal is to **locate**, **browse**, and **explore distribution**. In the case when they hover over a particular object and the object name is displayed, the user locates objects. Then, when the user hovers over a particular location of the image and the corresponding object name displays, the user browses objects. Finally, the user simply explores objects when they hover over different locations of the image, and our tool displays the corresponding object category. Furthermore, the user task is to discover the correlation among (1) pixel occupancy (in the given mask) and object IoU, (2) pixel occupancy (in the prediction mask) and object IoU, (3) pixel occupancy (in the given mask) and pixel occupancy (in the prediction mask).

4.3.4 Encoding and Interaction Idiom

We use a multiform view encoding. As shown in Figure 7, on the top left of the multi-form view, we show an image; on the top middle, we show the given mask, and on the top right, we display the predicted mask and all of them are visually encoded in the form of an image. Then, we show three correlations on the bottom and use three scatter plots to encode them. Our design has multiple interaction choices. Firstly the user can select any number of object categories using the checkboxes that we implement. Then the user can also hover (select and highlight) the prediction and given masks. The initial overview shows the scatter plots, images, and masks for all the object categories in the cityscapes dataset.

4.3.5 Critical Analysis

One of the principal goals of this task is to locate/browse/explore object categories in the image by looking at given and predicted masks. While looking at the predicted and given a mask with our tool (Figure 7), the users can discern different objects with the user of color hue, though this can be difficult sometimes. In cases with too many objects in an image or the objects are densely meshed, our designed color hue may not be enough to tell different objects apart. We have incorporated the hover (select) option to remedy that situation. The users can easily locate/explore different objects by hovering since our tool displays the object category at each mouse position, as seen in Figure 7.

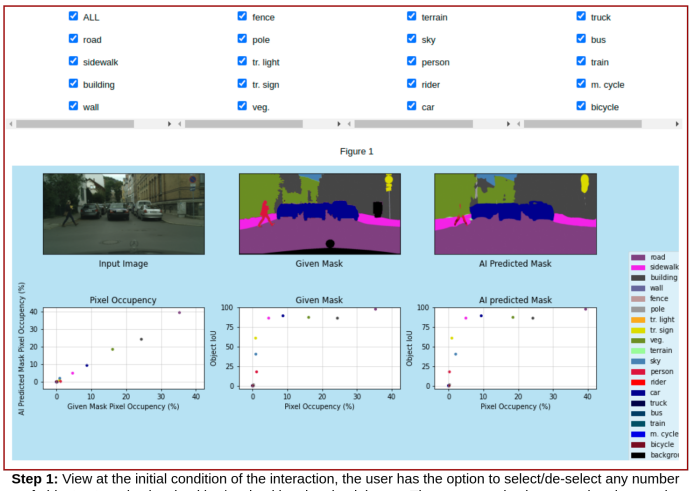
Another efficacious design choice that we use for this tool is the multiform view. Since we are showing (Figure 7) different information/data (6 in this case) with different idioms (2 in this case), this choice helps take the cognition load off of the user. And when there is information being displayed for too many object categories (how many is too many can be different for every image since some images may contain too many object categories meshed with each other, some may contain the same amount of objects but spaced apart evenly), decoding the object categories from the mask can be challenging. To remedy that, we introduce (Figure 7) the ability to interact as the users can choose for which object categories they want our tool to load information.

It is further observed in Figure 7 that for all objects, some of the scatter plots are difficult to read, especially when many points are positioned densely (the last two scatter plots in the Top Left of Figure 7). On the other hand, Figure 7 shows that the user can understand the correlation between pixel occupancy and IoU for different objects by looking at the scatter plots. Since the scatter plots exhibit a positive correlation (Figure 7), it helps the users answer questions such as: Does occupying more pixels lead to higher AI performance? The first scatter plot in Figure 7 also shows if, for different objects, the pixel occupancy in the given mask is equal or more or less in comparison to that of the predicted mask.

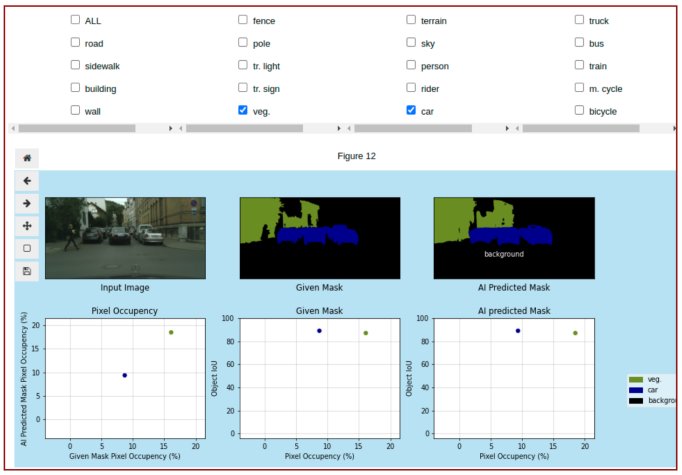
Section 3.6 describes the required data transformations to change the data semantics for task 3. These transformations are exceedingly crucial because, without the trained AI model, we cannot display AI-predicted masks and cannot compute IoU and pixel occupancy in the predicted mask, which are all visualized attributes (Figure 7) in the proposed idiom.

5 Future Work

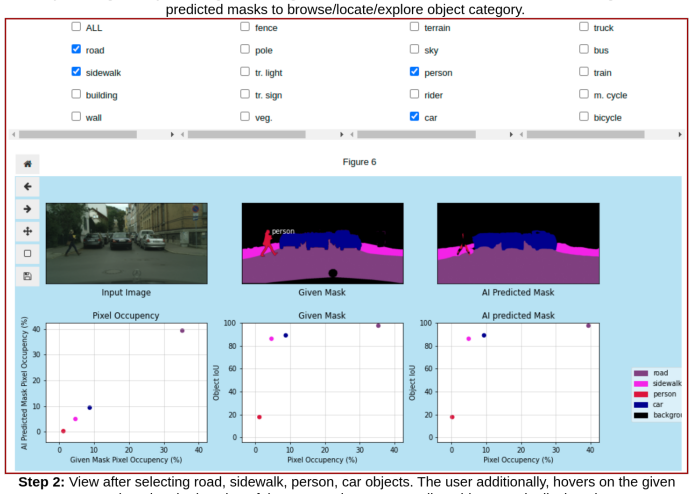
One of the more pronounced stretch goals of our visualization tool is to facilitate the usage of our tool in the training phase. What currently prevents us from incorporating this feature is the computational complexity of our tool. In order to run the grad-cam algorithm during the training phase, which is computationally expensive, becomes an added stress to the user. Because, usually, while training AI models, most of the hardware memory is occupied by models and data loaders, allowing computational resources to the visualization is challenging. More sophisticated re-factorization of our code to ensure optimal hardware usage is required to solve the aforementioned issue. If Incorporated, this feature will help the user gain insights into the model's behavior as they train models. Furthermore, this feature will help the user understand how model weights change during training. This visualization can, therefore, potentially become a helping tool towards demystifying the infamous *black-box* issues [42] in machine learning.



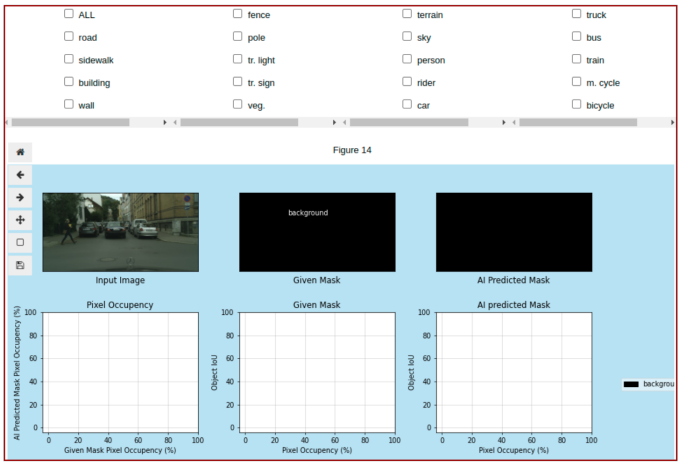
Step 1: View at the initial condition of the interaction, the user has the option to select/de-select any number of object categories by checking/unchecking the check boxes. The user can also hover on the given and predicted masks to browse/locate/explore object category.



Step 3: View after selecting veg, car objects. The user additionally, hovers on the predicted mask and at the location of the mouse, the corresponding object type is displayed.



Step 2: View after selecting road, sidewalk, person, car objects. The user additionally, hovers on the given mask and at the location of the mouse, the corresponding object type is displayed.



Step 4: View after de-selecting all objects. The user additionally, hovers on the given mask and at the location of the mouse, the corresponding object type is displayed.

Fig. 7: Cropped screenshot of our proposed tool for Task (section 4.3) from Jupyter notebook. Top Left: initial view of the proposed tool with no interaction. Bottom Left and Top Right: Original image, given mask, AI predicted mask, 3 scatter plots - (1) given and predicted pixel occupancy, (2) IoU and given pixel occupancy, and (3) IoU and predicted pixel occupancy for selected objects with hovers at the given mask (Bottom Left) and predicted mask (Top Right). Bottom Right: View after deselecting all objects with hover at the given mask.

Another stretch goal of ours is applying our visualization tool to additional datasets and different types of AI models. Currently, we have only experimented on CNN-based AI models; the next candidate AI model we plan to experiment with is vision-transformer [43] based AI models. Then we plan to expand our experimentation on different semantic segmentation datasets from various application sub-domains such as autonomous vehicle application: Scannet [44], GTA5 [45], IDD [12] and medical application: Kvasir-SEG [46], PROMISE12 [47]. Additional experiments such as those mentioned above can potentially yield more insights into our visualization's usability, efficacy, and adaptability in different domains.

In terms of usage, one of the biggest limitations of our visualization tool is its adaptability to another dataset by a different user. Developing a web portal or a generic Python library can potentially solve that problem. A user can easily use the web portal without configuring any new environment on their system and can use our visualization from the server. On the other hand, developing a Python library will help users integrate the visualization into their project easily with some basic configuration changes due to their (the user's) dataset and AI model. To mitigate these challenges, we require proper documentation of our code, which can further attract more contributors to our GitHub repository to drive project growth, quality, and usability.

Another limitation of our visualization tool is that users can't select or upload images for the tasks described in Section 4.2, 4.3. Giving that level of control to users will enable them to use our visualization tool for demonstration purposes more easily; our tool can be viewed as a readily available platform for the aforementioned tasks. Finally, there is always more room for making the visualizations more interactive, and these decisions and design policies need to be made with extensive research, prototyping, and adoption of user feedback.

6 Conclusion

We have presented three visualizations to understand AI interpretability and explore semantics between dataset statistics and AI models in a novel way. By surveying real users, we show how our proposed visualization can help target users better correlate their data and models. We demonstrate improvement in completing the aforementioned tasks driven by our visualization tool in our target user's real-life work. We believe that with the incorporation of our stretch goals, our proposed visualization tool can potentially be a significant research tool in computer vision.

References

[1] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223. 1, 2, 3, 5

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. 1

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755. 1, 3

[4] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641. 1, 3

[5] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," *arXiv preprint arXiv:1301.3572*, 2013. 1, 3

[6] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 891–898. 1

[7] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009. 1, 3

[8] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 davis challenge on video object segmentation," *arXiv preprint arXiv:1704.00675*, 2017. 1

[9] M. S. Hossain, J. M. Betts, and A. P. Paplinski, "Dual focal loss to address class imbalance in semantic segmentation," *Neurocomputing*, vol. 462, pp. 69–87, 2021. 1

[10] M. J. Jozani, É. Marchand, and A. Parsian, "On estimation with weighted balanced-type loss function," *Statistics & Probability Letters*, vol. 76, no. 8, pp. 773–780, 2006. 2

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002. 2

[12] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. Jawahar, "Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1743–1751. 2, 8

[13] X. Zhang, H. Xu, H. Mo, J. Tan, C. Yang, L. Wang, and W. Ren, "Dcnas: Densely connected neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 956–13 967. 2

[14] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-scnn: Gated shape cnns for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5229–5238. 2

[15] C. Drummond and R. C. Holte, "Cost curves: An improved method for visualizing classifier performance," *Machine learning*, vol. 65, pp. 95–130, 2006. 2

[16] J. Vig, "A multiscale visualization of attention in the transformer model," *arXiv preprint arXiv:1906.05714*, 2019. 2

[17] A. Luque, M. Mazzoleni, A. Carrasco, and A. Ferramosca, "Visualizing classification results: Confusion star and confusion gear," *IEEE Access*, vol. 10, pp. 1659–1677, 2021. 3

[18] M. Colley, B. Eder, J. O. Rixen, and E. Rukzio, "Effects of semantic segmentation visualization on trust, situation awareness, and cognitive load in highly automated vehicles," in *Proceedings of the 2021 CHI conference on human factors in computing systems*, 2021, pp. 1–11. 3

[19] J. Raymaekers, P. J. Rousseeuw, and M. Hubert, "Class maps for visualizing classification results," *Technometrics*, vol. 64, no. 2, pp. 151–165, 2022. 3

[20] M. Pühringer, A. Hinterreiter, and M. Streit, "Instanceflow: Visualizing the evolution of classifier confusion at the instance level," in *2020 IEEE visualization conference (VIS)*. IEEE, 2020, pp. 291–295. 3

[21] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union," June 2019. 3

[22] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3349–3364, 2020. 3

[23] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 603–612. 3, 5

[24] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding convolution for semantic segmentation," in *2018 IEEE winter conference on applications of computer vision (WACV)*. Ieee, 2018, pp. 1451–1460. 3, 5

[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626. 4, 5, 6

[26] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. 4

[27] P. Umesh, "Image processing in python," *CSI Communications*, vol. 23, no. 2, 2012. 4

[28] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. 4

[29] "GitHub - jacobgil/pytorch-grad-cam: Advanced AI Explainability for computer vision. Support for CNNs, Vision Transformers, Classification, Object detection, Segmentation, Image similarity and more. — github.com," https://github.com/jacobgil/pytorch-grad-cam, [Accessed 10-May-2023]. 4

[30] "Jupyter Widgets x2014; Jupyter Widgets 8.0.5 documentation — ipywidgets.readthedocs.io," https://ipywidgets.readthedocs.io/en/stable/, [Accessed 10-May-2023]. 4

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019. 4

[32] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445. Austin, TX, 2010, pp. 51–56. 4

[33] "chardet — pypi.org," https://pypi.org/project/chardet/, [Accessed 10-May-2023]. 4

[34] "Einops — einops.rocks," https://einops.rocks/, [Accessed 10-May-2023]. 4

[35] "Anaconda software distribution," 2020. [Online]. Available: https://docs.anaconda.com/ 4

[36] "pip — pypi.org," https://pypi.org/project/pip/, [Accessed 10-May-2023]. 4

[37] "Project Jupyter — jupyter.org," https://jupyter.org/, [Accessed 10-May-2023]. 4

[38] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021 4

[39] "Tableau: Business Intelligence and Analytics Software — tableau.com," https://www.tableau.com/, [Accessed 10-May-2023]. 4

[40] J. Breitenstein and T. Fingscheidt, "Amodal cityscapes: A new dataset, its generation, and an amodal semantic segmentation challenge baseline," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 1018–1025. 5

[41] "Turbo, An Improved Rainbow Colormap for Visualization — ai.googleblog.com," https://ai.googleblog.com/2019/08/turbo-improved-rainbow-colormap-for.html, [Accessed 10-May-2023]. 6

[42] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019. 7

[43] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020. 8

[44] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839. 8

[45] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 2016, pp. 102–118. 8

[46] D. Jha, P. H. Smedsrud, M. A. Riegler, P. Halvorsen, T. de Lange, D. Johansen, and H. D. Johansen, "Kvasir-seg: A segmented polyp dataset," in *MultiMedia Modeling: 26th International Conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, Proceedings, Part II 26*. Springer, 2020, pp. 451–462. 8

[47] G. Litjens, R. Toth, W. Van De Ven, C. Hoeks, S. Kerkstra, B. van Ginneken, G. Vincent, G. Guillard, N. Birbeck, J. Zhang *et al.*, "Evaluation

of prostate segmentation algorithms for mri: the promise12 challenge,"
Medical image analysis, vol. 18, no. 2, pp. 359–373, 2014. 8