Robustar Documentation

1. Summary

1.1. Revision History

- Initial draft complied on 10/21/2021 by Haohan Wang

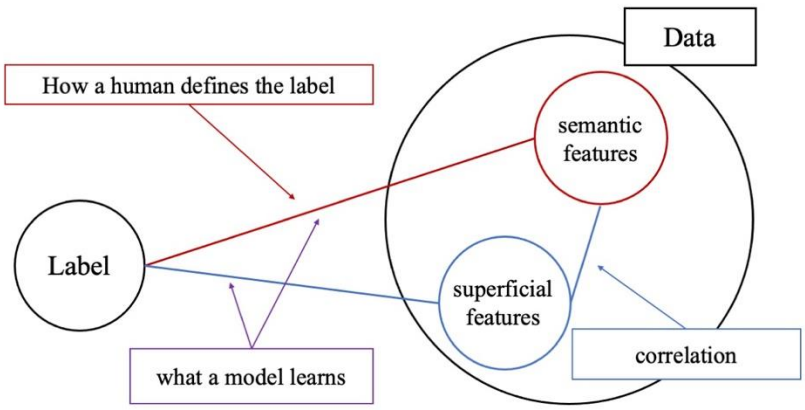
1.2. Introduction of the product

Background

Despite the recent advances of machine learning research, the understanding of the models is still mostly curbed within the study of i.i.d data, with only a few exceptions exploring the case when target domains are well understood (Ben-David et al., 2010). We notice that this scope is largely inconvenient for the nowadays industrial demand of robust models applicable to the data with unforeseen variations. Machine learning, especially deep neural networks, has achieved impressively human-paralleled performance nowadays. In the computer vision community alone, "super-human performance" has been announced repeatedly over various benchmark datasets from multiple labs in the recent years. How- ever, when these renowned models are applied to the industry, the community is soon alarmed with the vulnerability of them when facing unseen distributions.

As a result, the fragility of the neural networks is being studied extensively, and falls into multiple seeming independent, but internally coupled topics: the adversarial vulnerability studies the problem where models can be easily deceived by delicately designed artificial noises, even when these noises are not perceptible to human (Szegedy et al., 2013); the out-of-domain generalization focuses on the alignment of models' test performances over the samples from the training distribution and over the samples from novel distributions (Ben-David et al., 2010).

A closer investigation of the data reveals that a root of the vulnerability of these models is the existence of confounding signals (also known as the bias signals) in the data. For example, as the below Figure shows, within the data are the semantic signals and the superficial signals: the association between semantic signals and the label is what we desire and hope to train the model to learn, however, because of the correlation between semantic and superficial signals, there may be some correlation between the superficial signals and the label also. When a model is trained to minimize the empirical loss, it cannot differentiate these two sources of signals, and will pick up whichever one that can reduce the loss, resulting in a trained model that does not understand how the human define the label.



The above argument has been observed in many real-world applications. For example, in video sentiment analysis, where a model is trained to predict whether the speaker in the video is commenting positively or negatively, the background of the video correlates with the label. Therefore, the model does not need to understand how a human express opinions but can still predict with high accuracy based on the background (Wang et al., 2017). Similar stories can be told repeatedly with various observations. Even the intriguing problems of adversarial vulnerability can be viewed in this perspective (Wang et al., 2020b). Also, even over standard benchmark such as ImageNet, the community noticed that the models can predict the class tench without understanding the semantics: the model learns that there is always a hand holding the fish in the images depicting tench and considers this hand as the most predictive signal in the class (Brendel and Bethge, 2019; Hohman et al., 2019).

A great number of methods have been developed to mitigate this problem, but these methods have to introduce new assumptions, thus will be helpful only to a narrow scope of problems. In this project, we aim to bring in the most straightforward approach to resolve this problem: we will build a convenient platform that resorts human supervision to align the perception of a model to the one of a human.

Introduction

The project is built upon the assumption that one issue of the models' lack of the robustness in OOD generalization may be because the model learns the features that are not useful for prediction in other domains. For example, in the popular example of wolf vs. husky prediction, the model learns to predict through the background snow instead of the features a human will typically use to distinguish wolf vs. husky (Ribeiro et al., 2016).

There are tons of previous works aiming to solving this problem by introducing new inductive biases (regularization) (Wang et al., 2019b; Bahng et al., 2019; Wang et al., 2019a; He et al., 2019; Mahabadi et al., 2020; Nam et al., 2020) or new procedures for data augmentation (Geirhos et al., 2019; Hermann et al., 2020; Wang et al., 2020a; Hendrycks et al., 2019; Mahabadi et al., 2020; Shankar et al., 2018; Lee et al., 2021). Most of these methods will rely on the prior knowledge on some characteristics of the undesired features (e.g., the snow background in the previous example). While these new methods have demonstrated the practical value in various applications, one may argue that it may not be the best strategy that the community needs to develop a new method for domain experts whenever the domain experts request a new kind of features to be removed.

To better serve the domain experts, we introduce a system that allow the domain experts to directly interact with the training images, selecting the pixels that the domain experts consider useless. Then, the training process can help augment these features out and help train a model that does not use these features. Therefore, our system is designed to have the following core and other supporting functions:

- **core function**: to allow users interact with the training samples, annotating features that are useless used potentially used by the models to help train a robust model to the request of the domain expert
- **supporting functions:**
    - We use influence function to help identify the samples that need attention
    - We use saliency-map style interpretation to help the user identify the features need attention
    - We use segmentation models to help the user select certain pixels more accurately
    - We use data augmentation and regularizations to help train models invariant to the identified pixels.

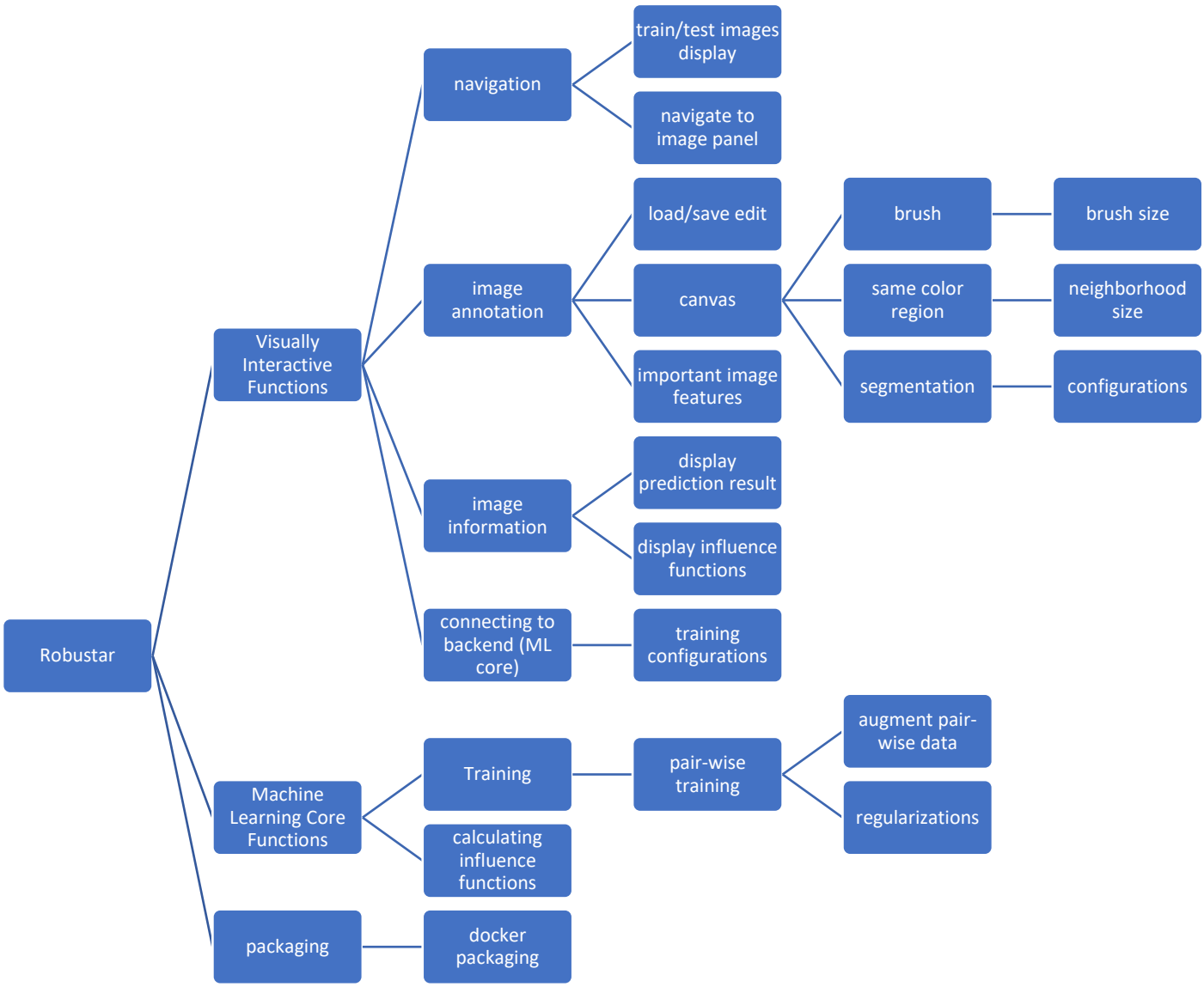
2. Potential Users

A main group of potential users is from the biomedical disciplinary, where the fact the model is learning useful information from the data is highly valued to conduct continued research with the model's learned knowledge. Since our current product only supports 2D images, the main user group will be biomedical scientists who are interested in building machine learning tool that can understand their 2D imaging data, such as molecular images.

As machine learning expeditiously takes important roles in multiple aspects from industries, the robustness challenge of AI has been widely referred to as "the elephant in the room" of machine learning. It is foreseeable that the demand of robust machine learning models will soon go beyond the biomedical setting and start to impact on multiple machine learning applications.

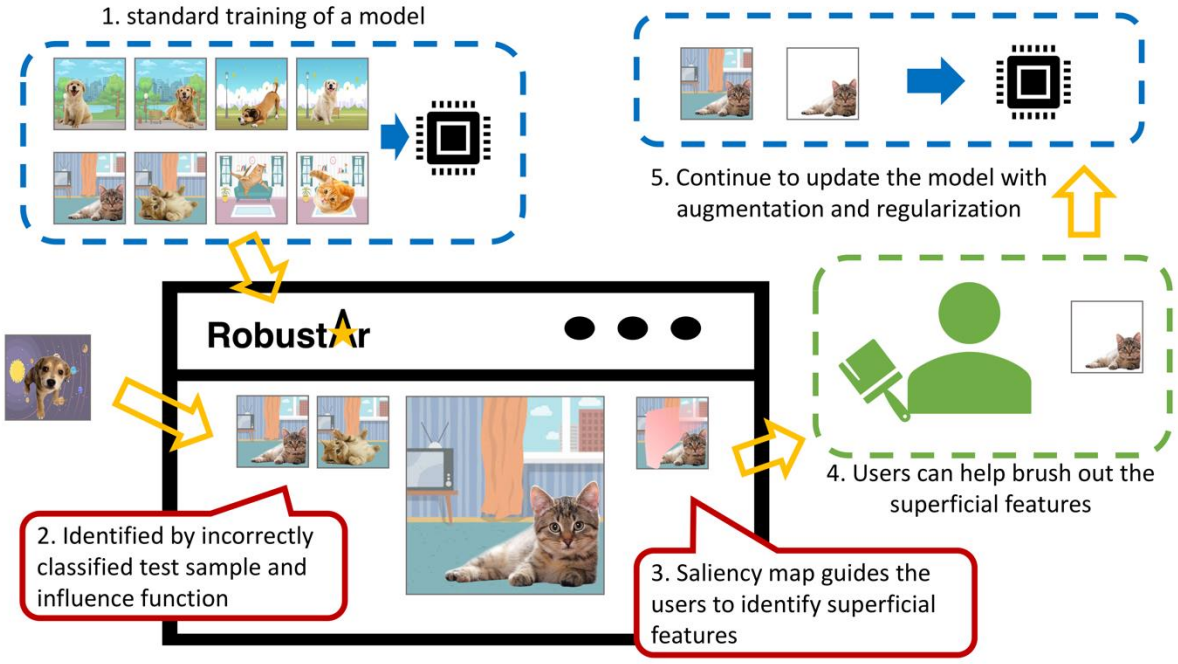
This project/product aims to first get prepared for this demand by interacting with biomedical researchers and gradually get extended to serve different real-world applications.

3. Product Structure



4. Workflow

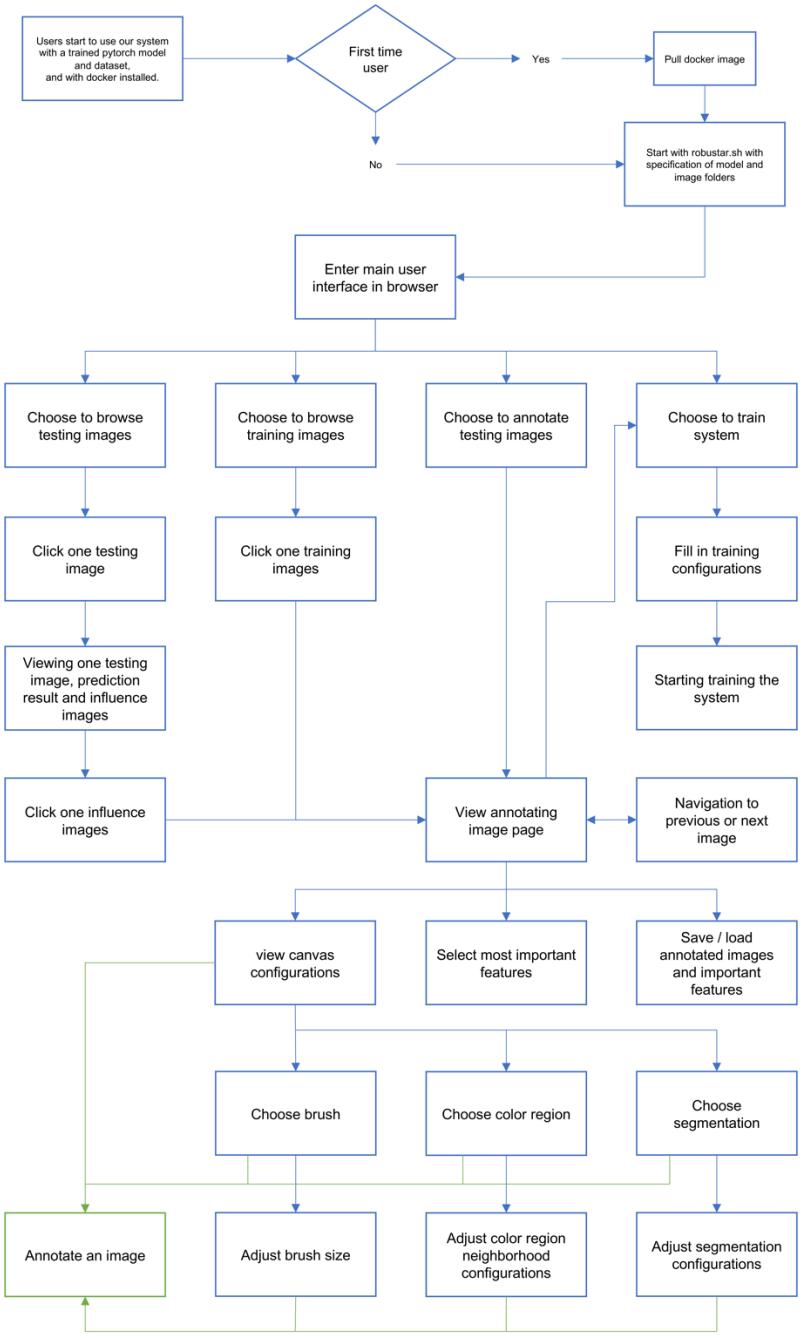
4.1. High-level usage workflow:



The working flow of the Robustar software:

1. the model can be trained elsewhere and then fed into the software;
2. With new test samples, the model can help identify the samples that are responsible for the prediction through influence function;
3. the software offers saliency map to help the user know which part of the features the model are paying necessary attention;
4. the users can use the drawing tools to brush out the superficial pixels;
5. new annotation of these images will serve as the role as augmented images for continued training.

4.2. Detailed (Regular) Workflow



5. Detailed Logic

I. Starting the system

- Install Docker
- Pull docker image
  - Is this related to the configuration of the system, like the CUDA version? (Do we need a separate docker image for every CUDA?)
- Run docker:
  - docker run …. --config <configuration_file>
  - configuration_file:
    - Model: <name of the model>, e.g., resnet18
    - Model checkpoint path: *.pt
    - Training data folder path:
      - Data folder is pytorch image folder:
        - images/
          - dog/
            - 1.png
            - 2.jpg
            - …
          - cat/
            - Abs.png
            - Dge.jpg
            - ...
          - bird/
          - …
    - Test data folder path:
      - Data folder follows the same structure
    - Influence data folder path:
    - Output data folder path:
      - Data folder follows the same structure
    - New model checkpoints folder path:
    - # of classes:
    - Each classes name:
    - Image size
    - …
- How do we save the user-annotated masks outside the docker?

II. Main Navigation Page

Four major functions (frontpage):

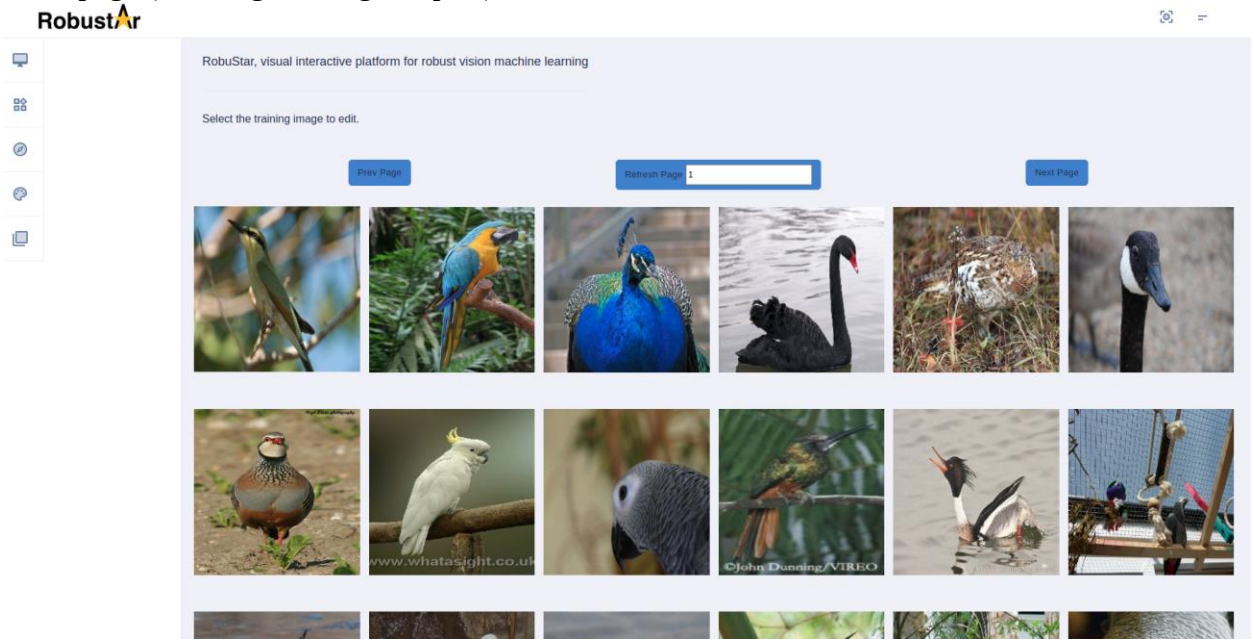
- Training samples -> training samples page (page 1)
- Testing samples -> Testing samples (page 1)
- Annotation -> annotation of the training sample (sample #1)
- Training the model -> training configuration page

Sidebar:

- System
    - Training model -> training configuration page
- Data
    - Training data -> training samples page (page 1)
    - Testing data -> testing samples page (page 1)
- Operation (only relevant in annotation page)
- Edit (only relevant in annotation page)
- Documentation

III. Training Samples Page

Main page (showing training samples):

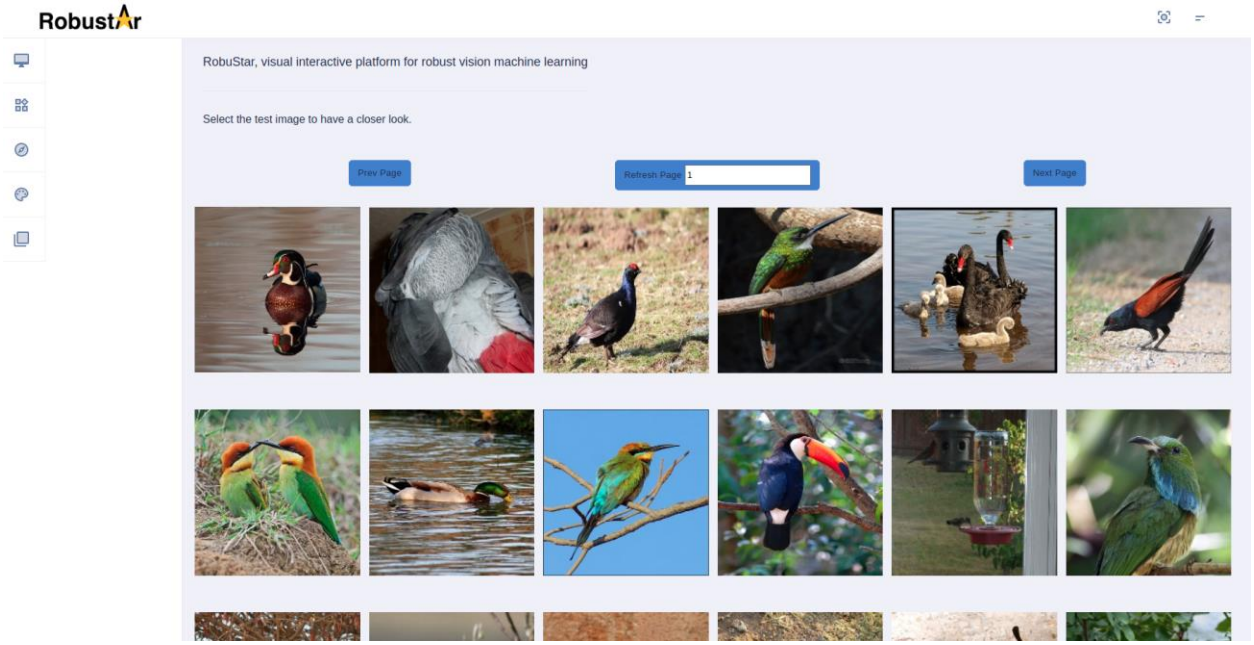


- Navigation
    - Prev Page: -> previous page
    - Goto -> go to a certain page
    - Next Page: -> next page
- Image Panel
    - Left-click an Image -> annotation page of an image

Sidebar (same as main navigation page)

IV. Testing Samples Page

Main Page (showing testing samples):

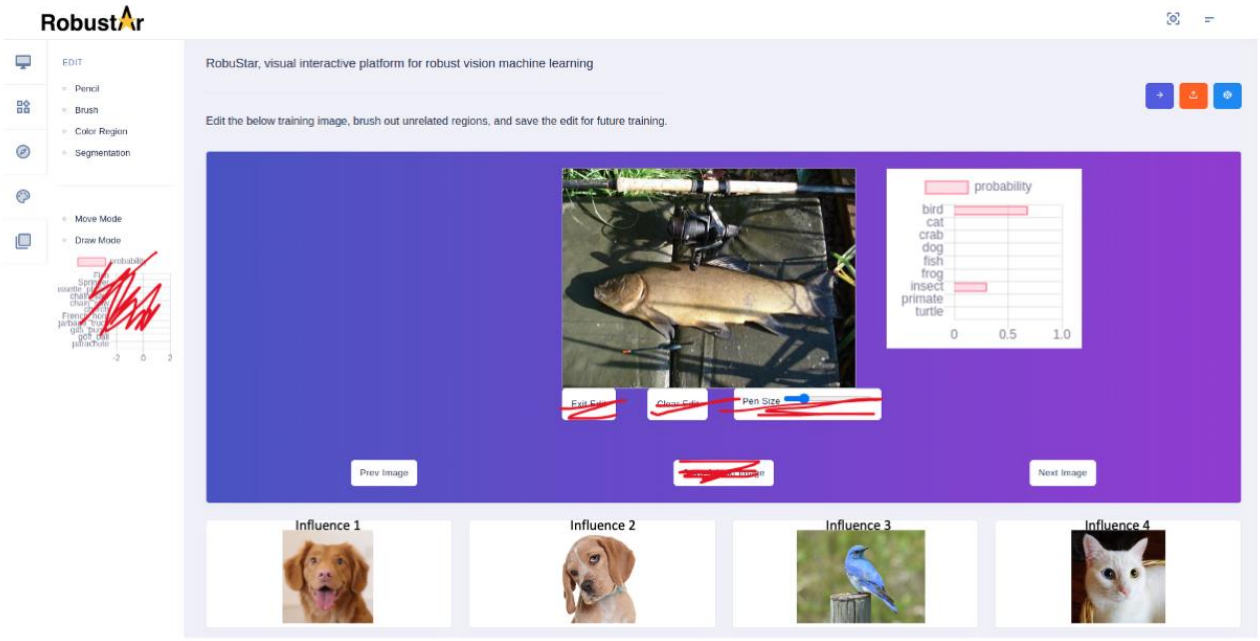


- Navigation
    - Prev Page: -> previous page
    - Goto -> go to a certain page
    - Next Page: -> next page
- Image Panel
    - Left-click an Image -> **Prediction page of an image**

Sidebar (same as main navigation page)

V. Prediction Page

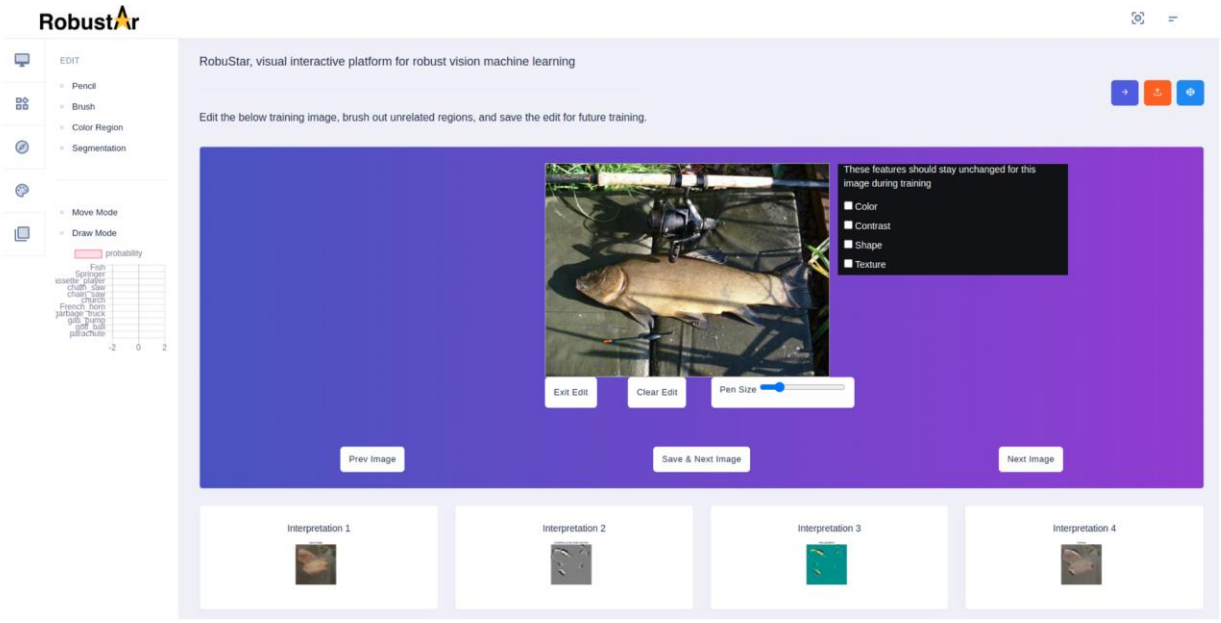
Main page:



- Displaying the image
- Displaying the prediction result
- Navigation:
    - Previous image
    - Next image
- Influence:
    - Influence images
        - Left-click -> annotation page for the image

VI. Annotation Page

Main Page:

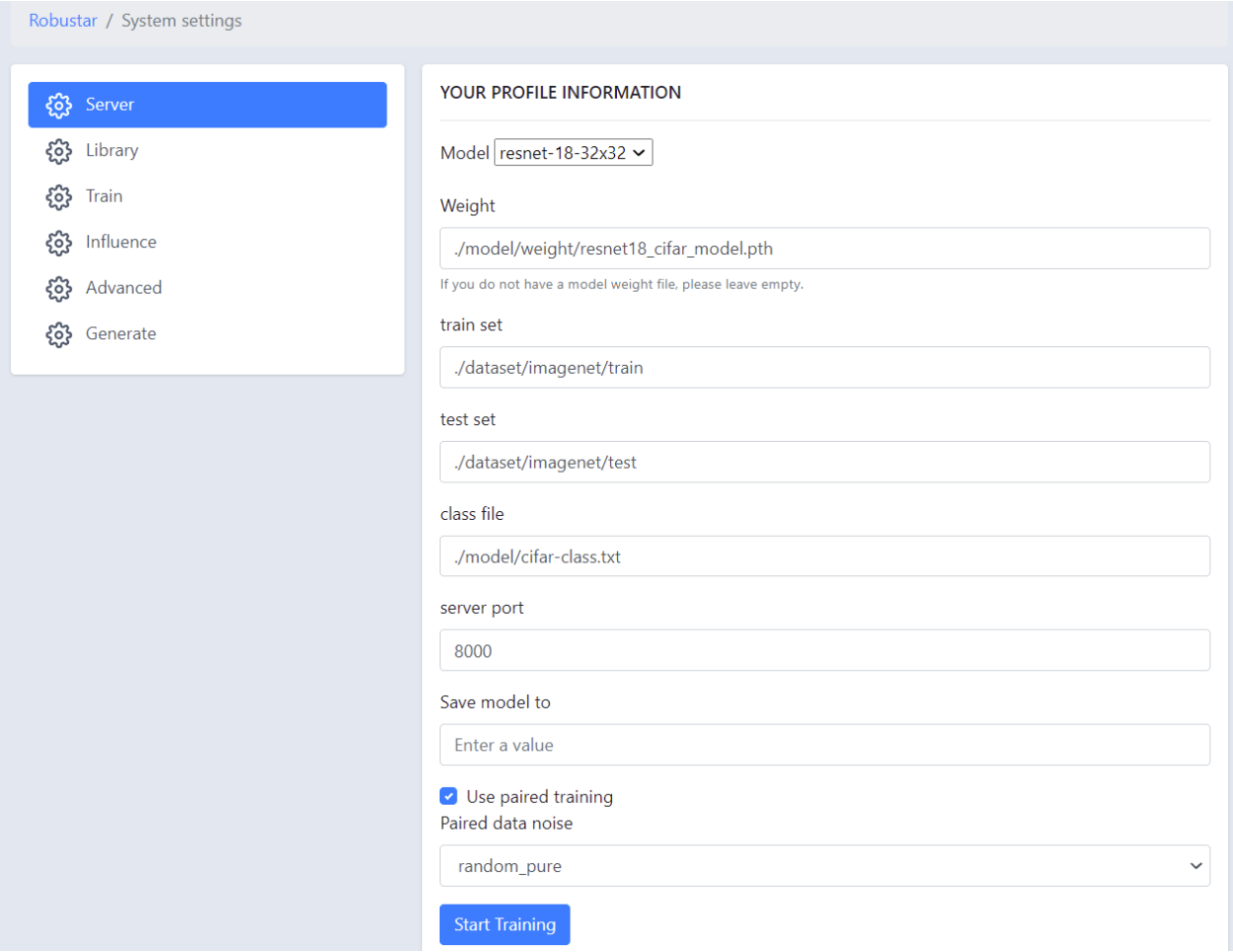


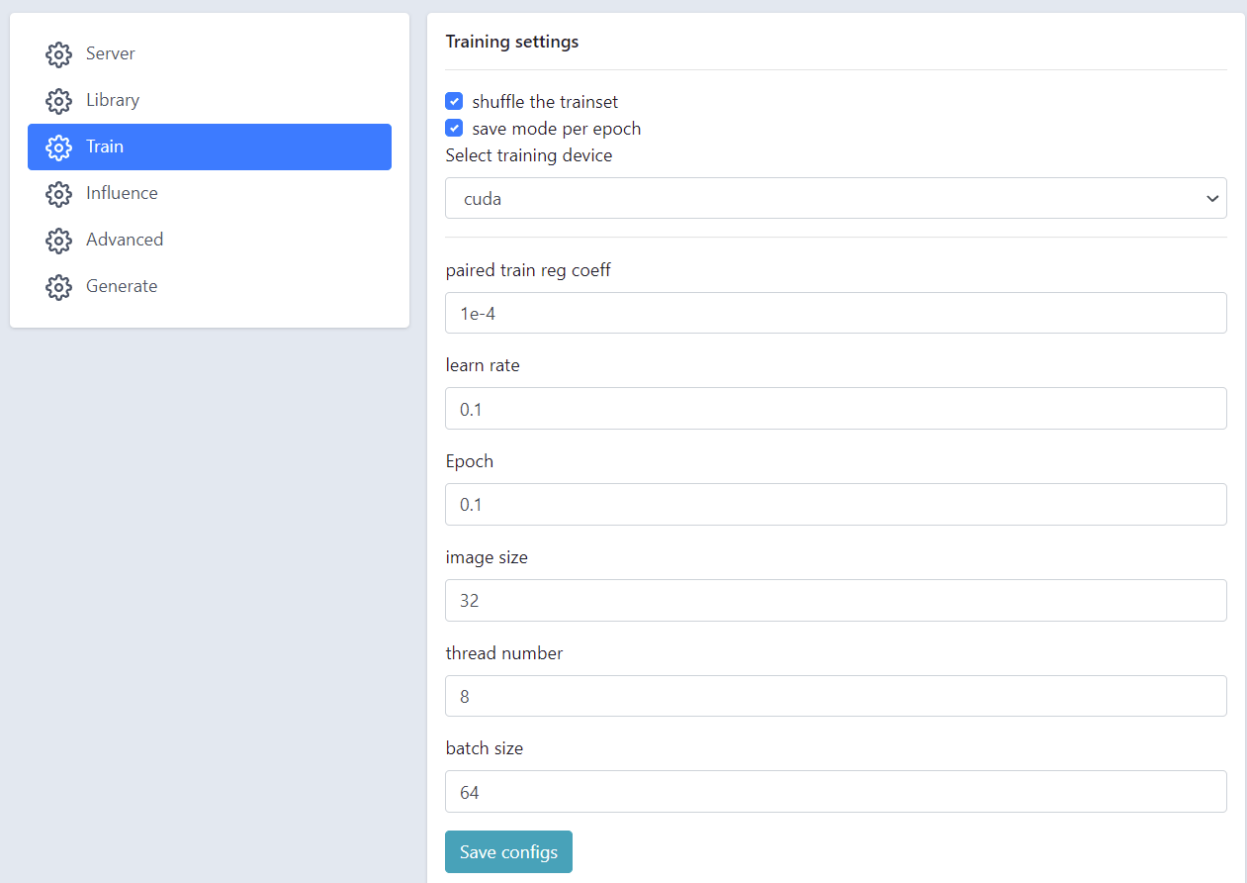
- Displaying the main image (Canvas)
- The checkbox of the important features used for training
    - Allow choosing multiple options
- Major edit control:
    - Exit Edit
    - Clear Edit
    - **Undo (undo the last edit)**
    - Pen Size
- Navigation
    - Previous image
    - Save & Next image (does not function properly)
    - Next image
- Interpretation
    - Displaying the interpretation images
        - Gradient across RGB channels
        - Max gradient
        - Overlap between original image and saliency map
- Short-cut buttons (right-top corner)
    - Next image
    - Save Edit
    - Train Model

Sidebar

- System
  - Training model -> training configuration page
- Data
  - Training data -> training samples page (page 1)
  - Testing data -> testing samples page (page 1)
- Operation
  - Save Edit -> save the current edit
  - Load Edit -> load edit and update the canvas
- Edit
  - Functions:
    - Pen/Brush
    - All pixels with the same color
    - Segmentation result
    - …
  - Modes:
    - Draw Mode
    - Move Mode
- Documentation

VII. Training Configurations





- Configurations
  - Training Mode (single choice):
    - Train from scratch (randomly initialized)
    - Train from standard pretrained weights
    - Fine-tune
  - Training Configuration:
    - Optimizer: (single choice)
    - Learning Rate:
    - Epochs:
    - BatchSize:
  - Training With Annotated Data:
    - With Regularization: checkbox
    - Regularization weight:

VIII. Training Curves

embedded tensorboard