LIFT Studio

Datasets

You need to create datasets to be able to train classification models. Each dataset should contain labels in both Train and Test lists. It is important to maintain a correct ratio between train and test lists to get better results. The default ratio is set to 80%-20%, which means that approximately 80% labels of the same type will be put into train list, and 20% into test list

To create dataset:

1. Navigate to https://studio.liftdata.ai/datasets page and click on New Dataset button
2. In the pop-up window input dataset's name and, optionally, specify the tags. Tags can be used to list all the object types in the dataset, e.g. Weapons, Grenades, Utility, Icons, etc
3. Hit Save button to finish creation process

Once the dataset is created you'll be navigated to the dataset's page, where you can manage the content of the dataset, upload images, label images, change Train-To-Test ratio, or move images between Train and Test lists to better suit your needs.

First of all, you need to upload images - click on the Upload images button in the top right corner of the page and select as many images as you need.

Once uploaded you need to label images. When the first image will be uploaded, the Label images button will appear near the Upload images button. Click on it and you'll be navigated to the labeling tool.

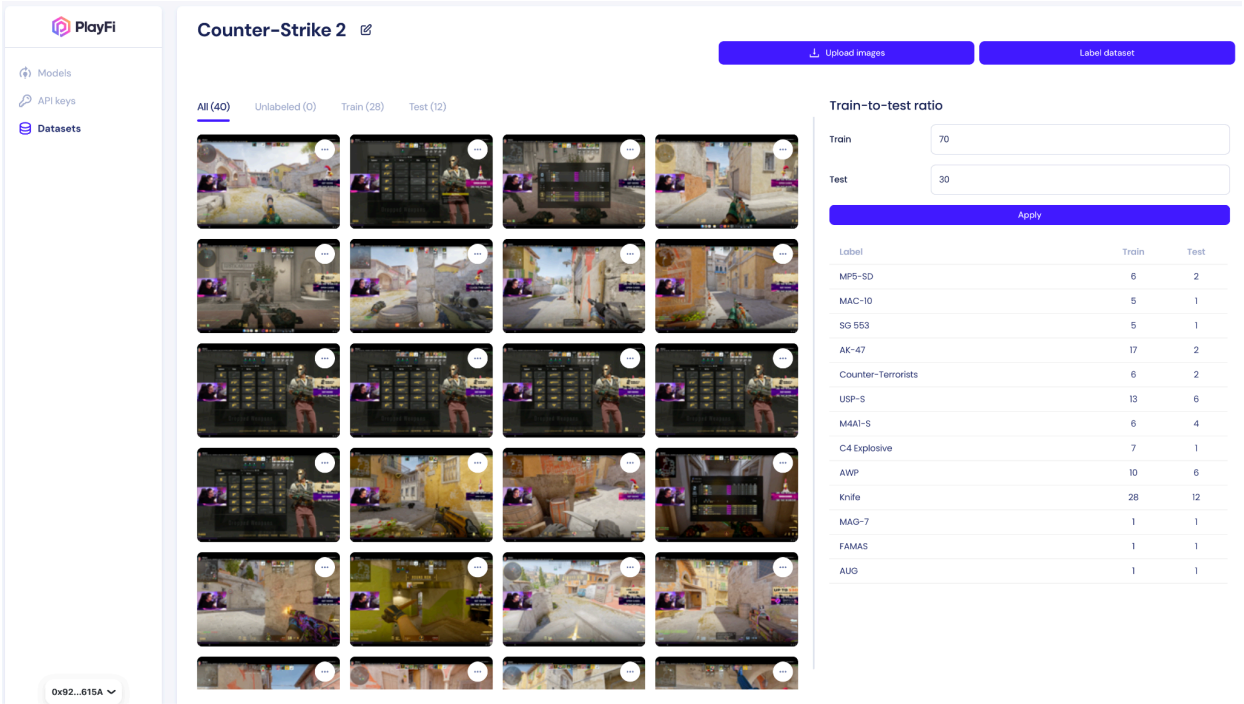
To label image you will need to:

1. Click on the image preview in the Dataset images list on the right side of the screen. It will load the original image and existing labels, if you previously added any
2. In the toolbar on the left side on the screen you'll see:
    a. Image status toggle - it controls in which list all labels from this image will be placed
    b. Add object selection button - it allows you to place new selection box on the image, which you can drag and resize as per your needs
    c. Copy selected labels button - will copy all selected labels
    d. Paste selected labels button - will paste all in-buffer labels to the image that you are currently editing
    e. Section with the labels - each block represents a label on the image. You need to click on the Gear icon to set the name of it. It is important to keep label names consistent, thus both Class and Type inputs has auto-suggestion functionality which allows you to pick from existing labels to prevent typos

f. Save button - saves all the labels, their Classes, Types and coordinates. It is important to Save your changes before you switch to another image because you'll lose your changes

3. Click on Add object selection - a new bounding box will appear in the editor area, in the middle of the screen. Drag and resize the bounding box over the object that you want to label. Ideally object should fit in the bounding box and there should be no space between object edges and bounding box border
4. Once you are happy with the placement of the bounding box look for the block named Object selection in the Section with the labels. Click on the Gear icon.
5. A small pop-up window will appear. First of all fill Class input. This input can be used to logically group objects, for example you can group AK-47, M4A1-S labels into Weapon class
6. When Class input will be filled Type input will activate and you can name your label. It is crucial to maintain correct label names, for example AK-47, Ak-47, ak-47 are three different labels. This inconsistency should be avoided, thus pick one name and stick to it. Type augo-suggestion will help you with it as it will show you a list of previously added Types within the specified Class
7. Repeat this process steps 3 to 6 until all the objects are selected
8. Click on Save button!
9. After your changes will be saved you can safely pick another image from the Dataset images list

When you have labeled images you dataset's page will look like this

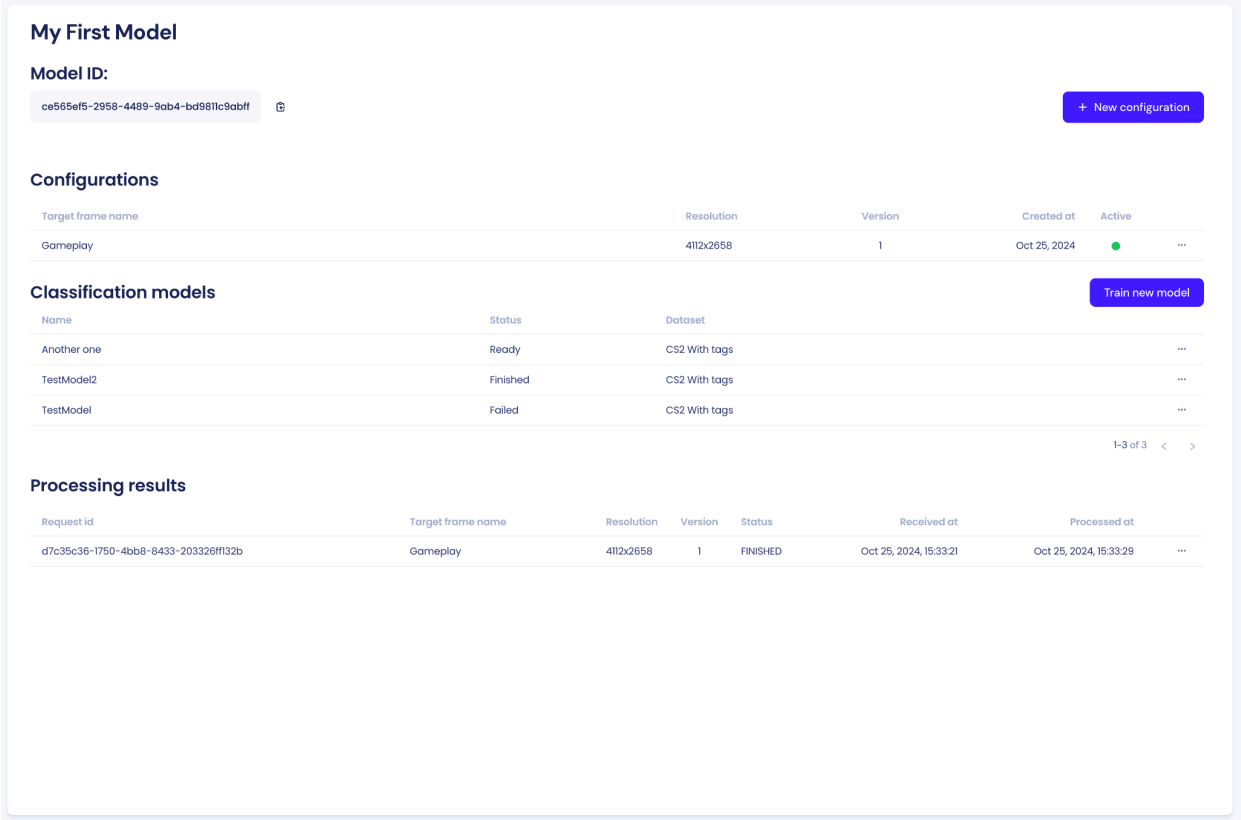


You can click between Tabs to check what images are placed in which set. It is important to have no Unlabeled images.
At the right side of the page you will see the Train-to-test ratio form. It allows you to automatically distribute images and labels between sets, while respecting the ratio. Almost. It is not always possible to split labels into two lists because one image can contain more than one label, which makes it difficult to ideally maintain the ratio. But it tries to.

List under the form contains all the labels that you have in your dataset and how they are represented in each set. Ideally, you want to have something around 100 images for each label, 80 of them are placed into the Train set, and 20 in the Test set.

Models

The process of Model creation is pretty-straightforward - navigate to https://studio.liftdata.ai/models page and click on Create new model button. Give it a name and move forward.



Here is a breakdown of what you can find on the model's page:

- Model ID - Model identifier in UUIDv4 format that you need to make Image Processing requests via API.
- Configurations - is a list of configurations that you have for different screens. For example you have a model for Counter Strike 2, and you want to extract data from different frames - match end screen, gameplay screen, scoreboard. For each of those screens you need to have a separate configuration. And you should have as many configurations for that screen as many resolutions you are going to process.
- Classification models - list of Classification models that you've created and trained. You need them in case you want to extract objects from the images.
- Processing results - list of all Image Processing requests results.

Configurations

Configuration is a group of settings that you need to specify in order to correctly detect and process the image. Configuration consists of two parts - Target Frame Detection config and Data Extraction config.

Target Frame Detection configuration contains a list of anchors that will be used to filter images. If the image passes all the checks, Data Extraction configuration will be used to extract text data and/or objects.

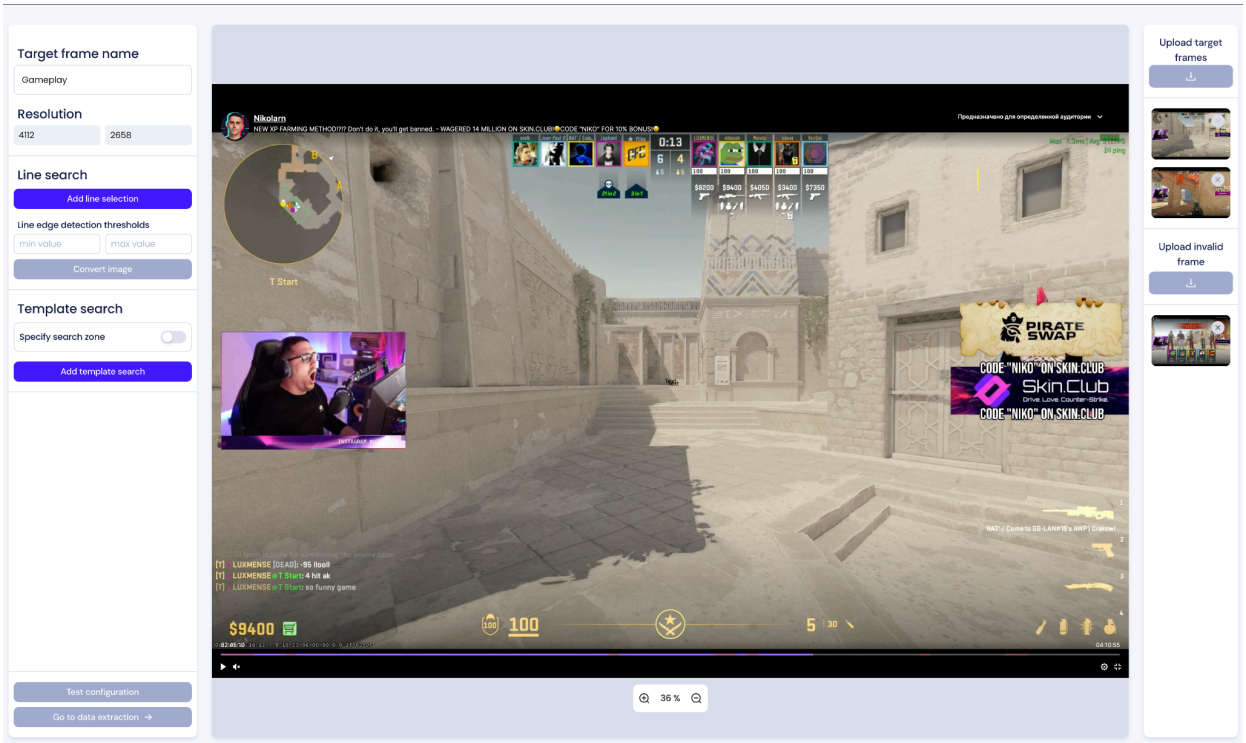
To create a new configuration click on New configuration button, and you'll be navigated to the editor.

Target Frame Detection

Editor consist of three zones:

- Toolbar - contains all the inputs that you'll have to use to create a configuration
- Working area - uploaded image will be rendered here. You can zoom in and out, place, rotate and resize bounding boxes, etc
- Images upload area - a few buttons that you'll use to upload images.

First of all you need to upload images. Pick two target frames that you want to work with. Ideally, it should be two different images, so you can test your configuration in different ways. You'll also need a non-target frame, which will be used to test the configuration for false-positives.
Once you upload all three images inputs will activate and you can start to work on the detection



It is important to note that the resolution of the uploaded images is automatically detected. If you want to process 1080p screenshots don’t upload screenshots with resolutions other than 1920x1680.

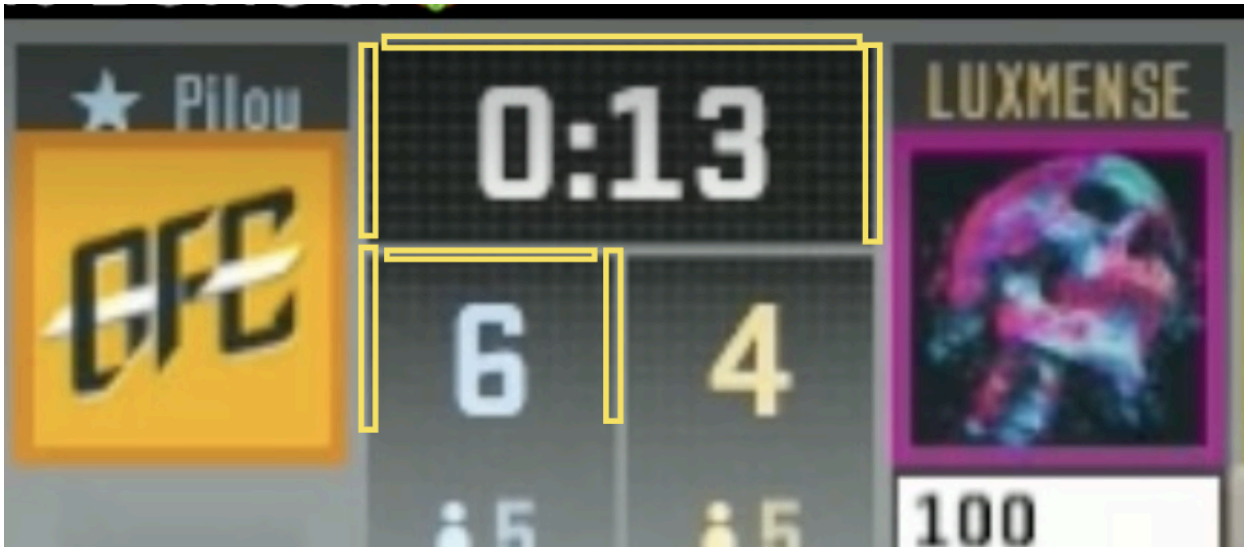
You can click on the images to change the active image in the working area.

Now to the hardest part: detection. To reliably detect target frames you need to find anchors that are present only on this type of frames to prevent any false-positive detections.

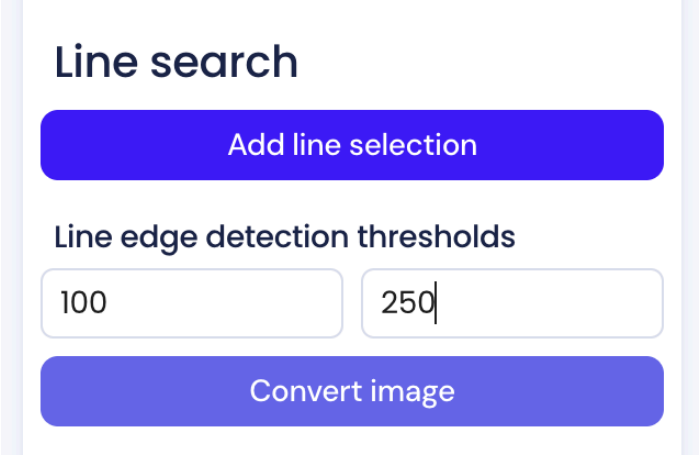
To mark the anchors you can use two options: Line search or template search.

Line search

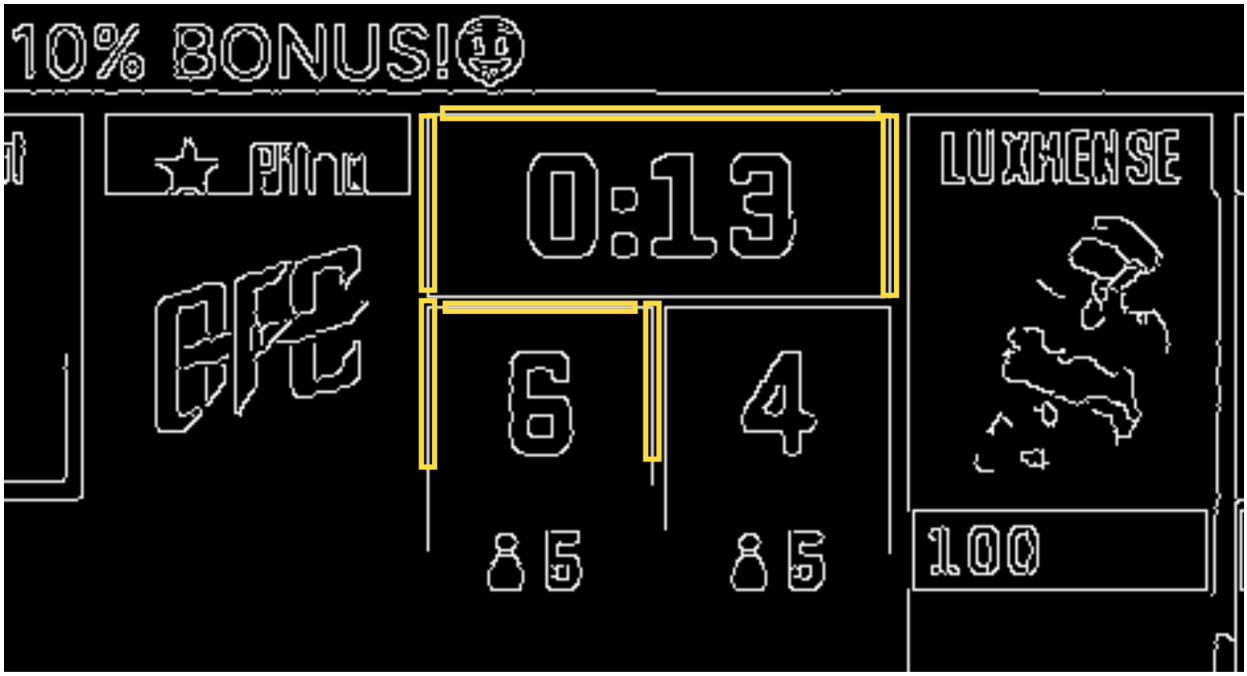
Line search is the option which allows you to select horizontal or vertical edges of some element. For example you can use the edges of the timer and the teams scores boxes.



You need to make sure that the selected lines are not fuzzy and as straight as possible without any warps. To see the lines the way the model sees it specify Line edge detection thresholds and click on Convert image button



The image in working area will be converted to the black and white image, with almost all noise removed.

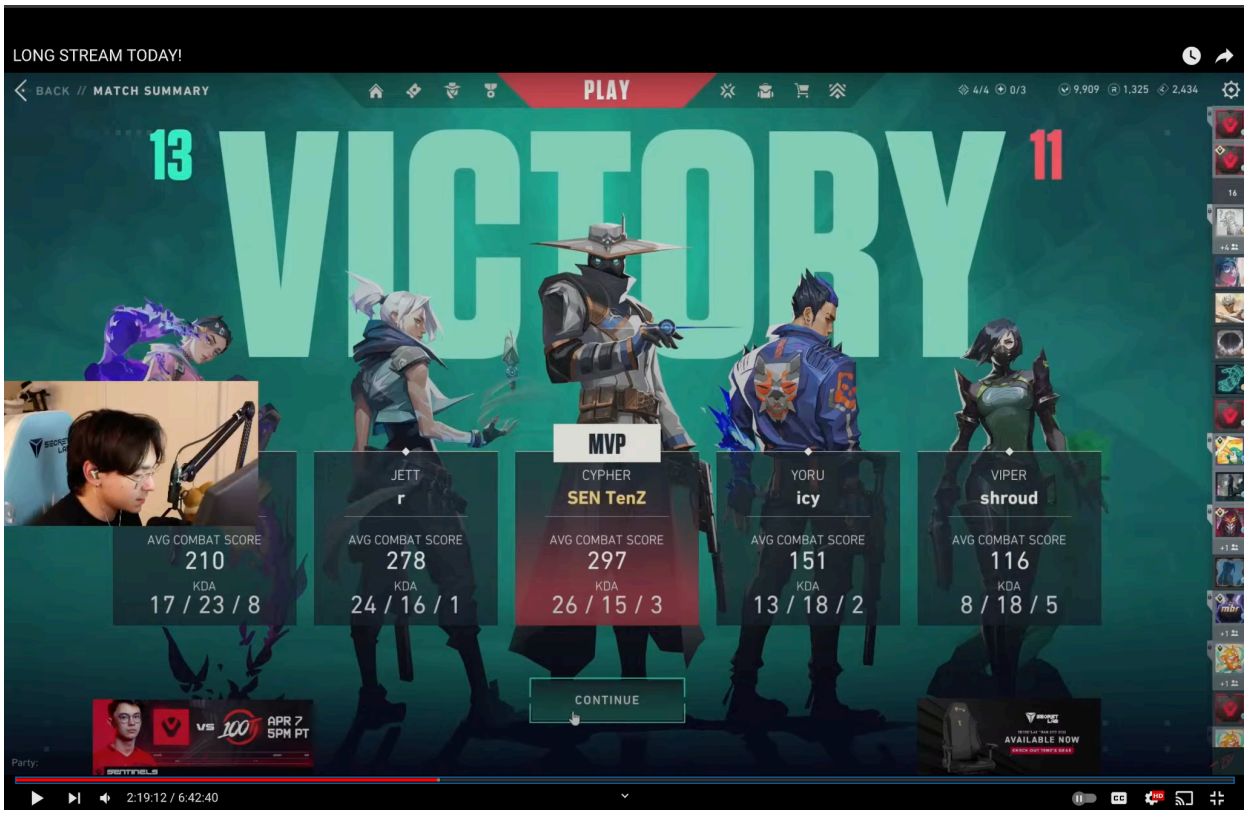


If the line is fuzzy or warped try to play with the thresholds to get the best results. This values will be used later on by the model to achieve the best results.

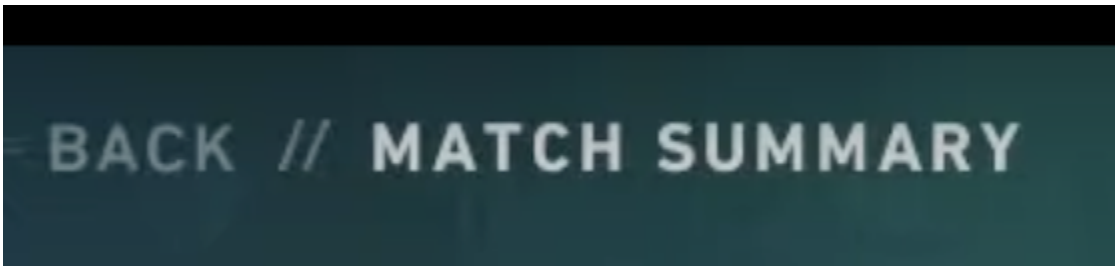
Template search

Template search is another option to mark anchors. It works great with static elements or objects, color of which isn't changing, there is no animation on the element, etc. For example you can't use template search on the timer from the previous screenshot because the time on it changes (but it still can be used if, for example, you want to capture a very-very specific frame).

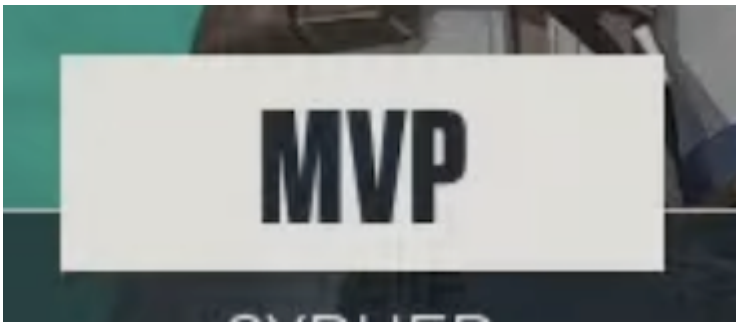
On the example of Valorant match end frame let's define appropriate anchors



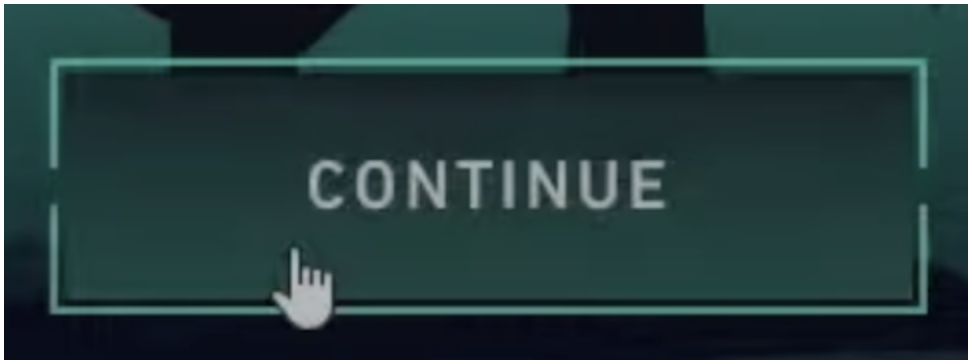
- Match summary section always appear on the end match screen, consistent in color, font size, no (or almost no) background noise



- MVP label on the Match summary screen in Valorant - always present on this target frame, consistent, doesn't change color



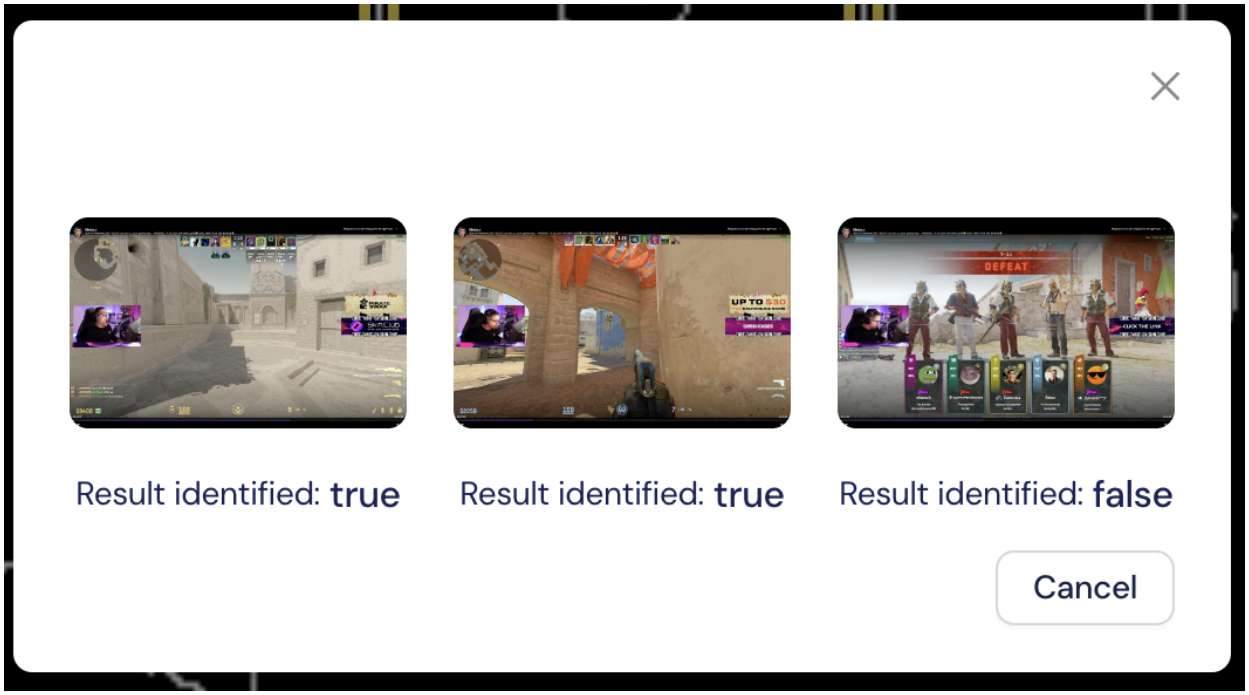
- Continue button can also be used as an anchor but it isn't perfect because it changes color on hover

- 

You need at least three template search anchors for this method to work properly.

Test Configuration

After you select all the anchors you need to test your configuration. To do it click on the Test configuration button and wait a few moments. Once the test is over you'll see the pop-up with the tests results



You need to make sure that you get 2 positives on the Target frames and a negative result on the non-target frame. All other outcomes mean that you need to update your configuration because the model is unable to detect the target frame and/or filter out invalid frames.

Once you are happy with the results click on the Go to Data Extraction button. Don't forget to name your configuration!

Data Extraction

This is the next and the last step of the configuration creation process. The overall look of the screen is the same, but the toolbar is a bit different.

Toolbar consists of:

- Specify classification model input - allows you to specify previously trained models that must be used to detect objects
- Add label selection button - places a new bounding box over the working area, which you can drag and resize to over the elements that you want to extract
- List of selections
- Test configuration button - similarly to the previous step you want to test your configuration and see how it extracts the data
- Save button

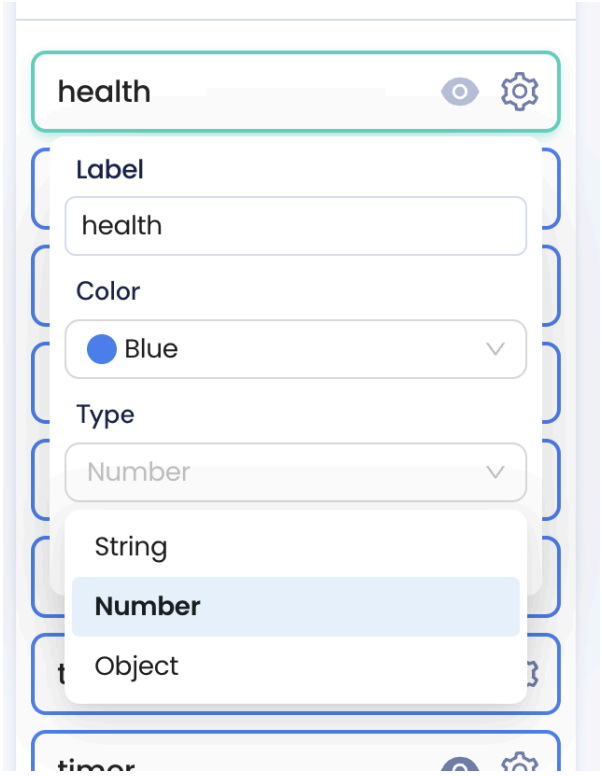
First of all, you need to understand whether you want to detect objects on the frame or not. Depending on your choice you need to train your own classification model, and pick it from a Specify classification model dropdown list, or choose Don’t use classification model in case if you don’t need it.

Selections

After it the process is very straightforward and repetitive:

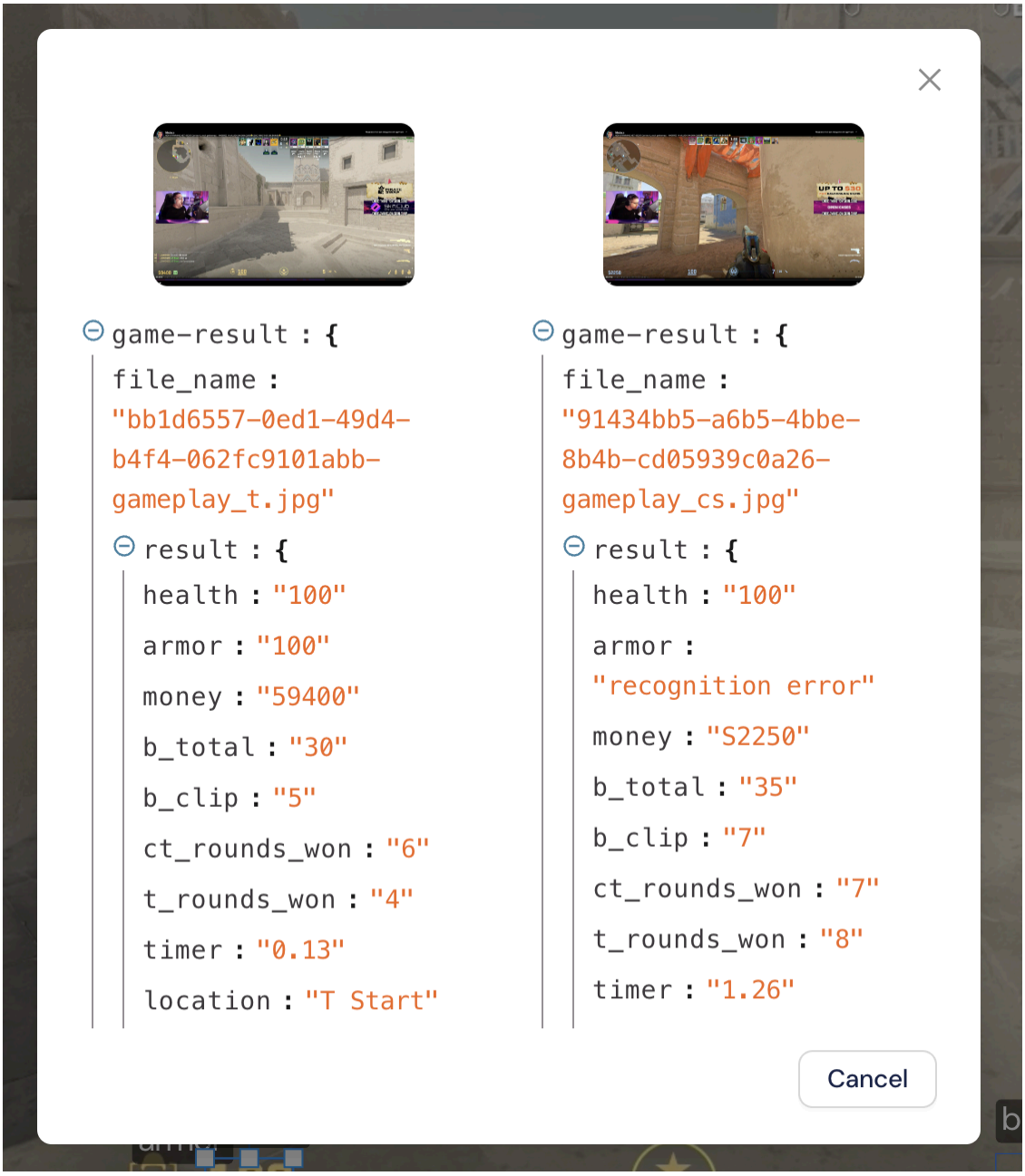
- Click on Add label selection

- Drag the bounding box to the element that you want to extract. Resize it, so the element fits into the box. Also take into account that the element can have a different width based on the content. You need to make sure that all the possible variations will fit into the box.
- Once done with the placement click on the Gear icon of the selection
- In the pop-up window name your bounding box, optionally specify bounding box color and specify type of the extracted data. For example if you want to detect which weapon user wields you need to specify the Object type, as classification model will be used to extract it.

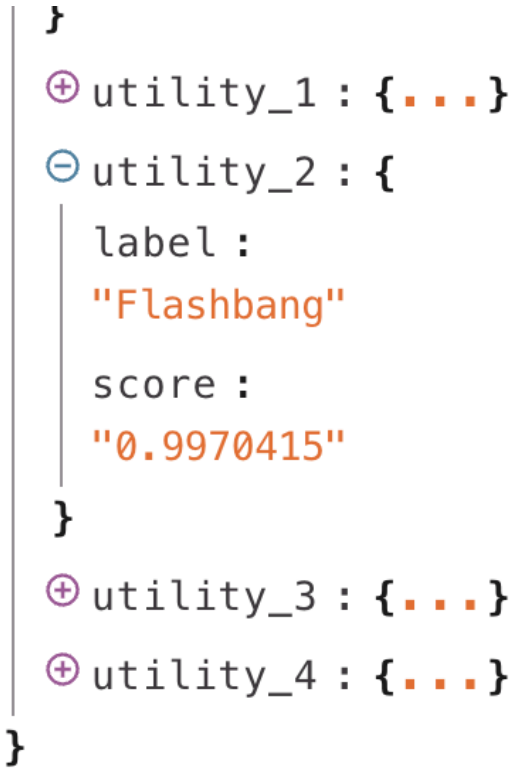


Test configuration

Once you are happy with your selections click on the Test configuration button and wait. It can take a minute or so to process everything, but once done you should see something similar to this



In response you'll see the data, that model was able to extract from the image. Label names are used as property names, so be careful with it.
Object detection result is a bit different tho. It is a nested object with the object type, and probability score

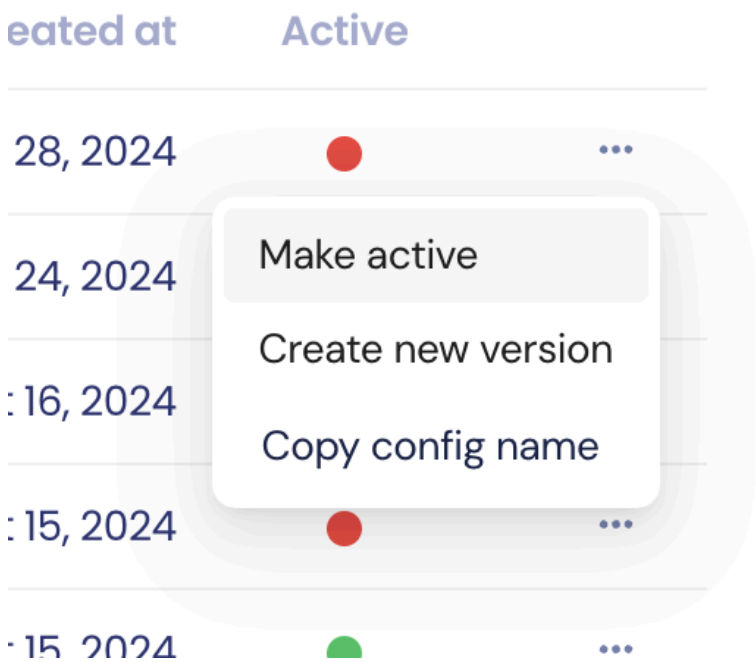


Once you are happy with what you've done, hit the Save button in the toolbar.

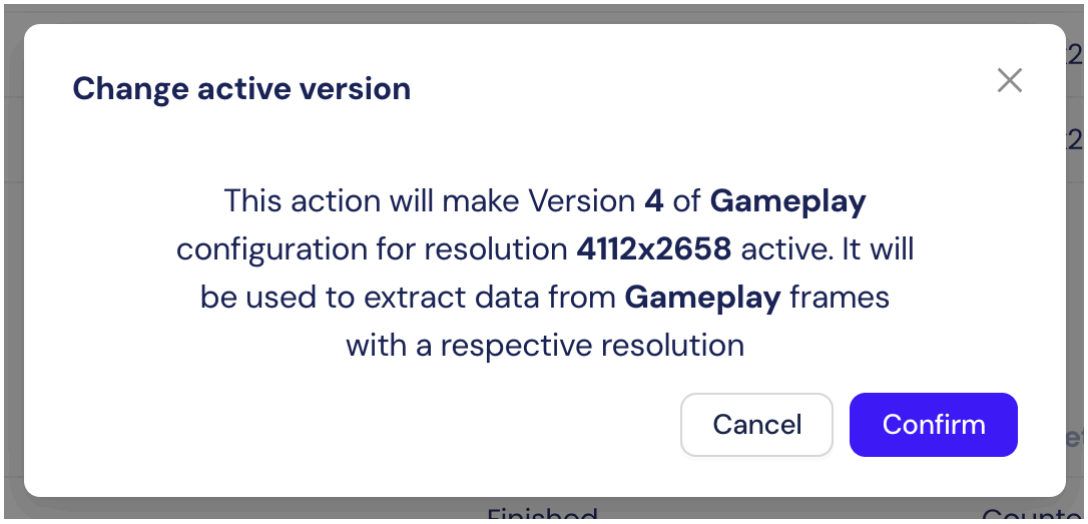
Publish configuration

In order for your configuration to become public (i.e. to be used by the AI Nodes), you need to mark it as Active. To do it:

- Go to the model's page
- Find the configuration, that you want to publish
- Click on the three dots icon and in the options list click on Make active



- You'll see the confirmation screen. Read it and click on Confirm button



That's it. Please note that only one configuration per frame per resolution can be active at a time. E.g. For my Gameplay frame, for the weird MacOS resolution Model will use version 4 of the config to detect and extract data.

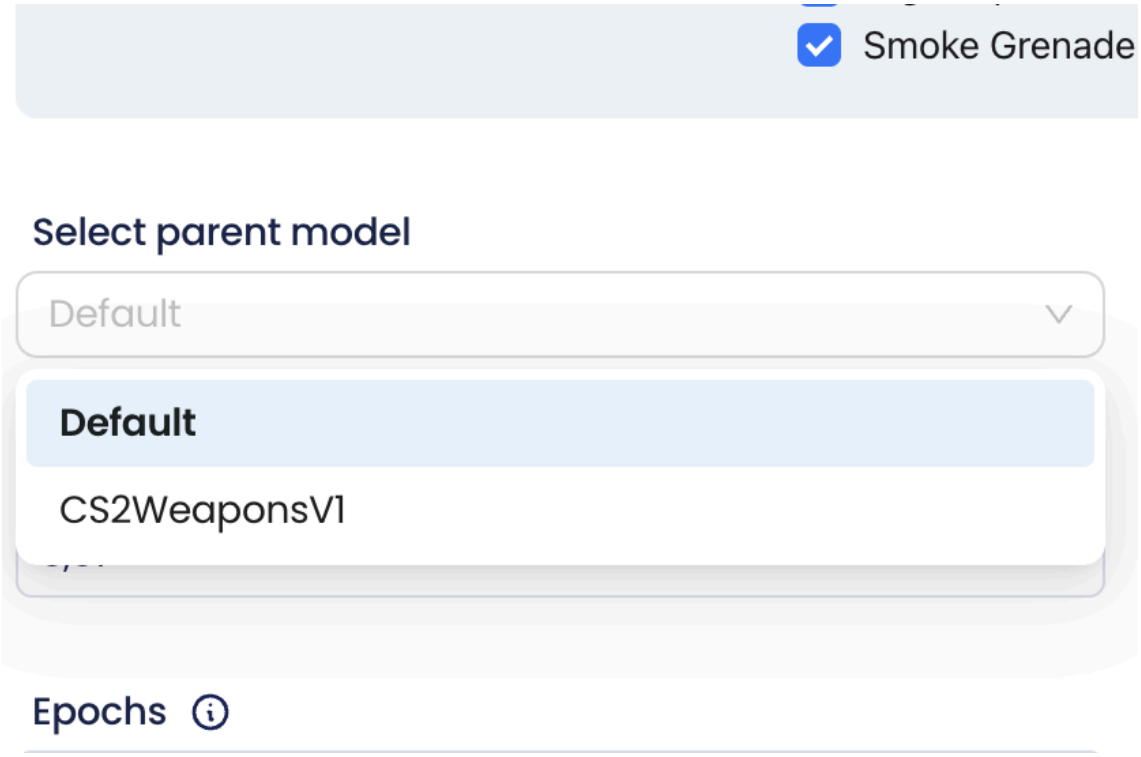
Classification Models

To detect objects you need to train a classification model based on a previously created and labeled dataset.

To create Classification model:

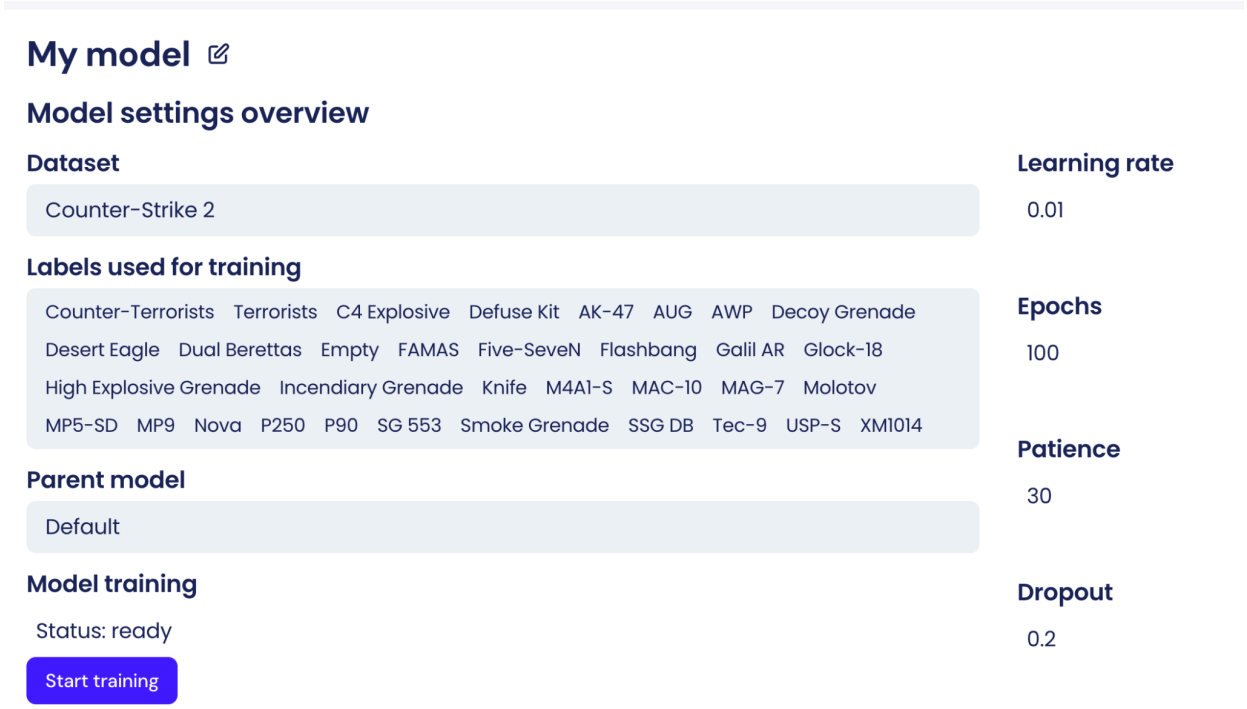
1. Go to the model's page and click on Train new model button
2. Start filling the form by naming your model
3. In Select dataset input select the dataset that you want to use
4. Once selected, all labels from the dataset will be loaded and grouped by class. You need to check only those labels that you want to train your model on.

5. In Select parent model select that model that you want your model to be based on. If it is the first version you'll have to use Default option. Later on, when you'll have more trained classification models you'll see them listed there too



6. Fill the rest of the inputs as per your needs. If you don't know what does any of those fields mean you can hover the info icon and read the tooltip
7. If you are happy with your choices click on the Save button

Once saved, you'll be navigated to the classification model's page, it contains all the information that you've specified during creation

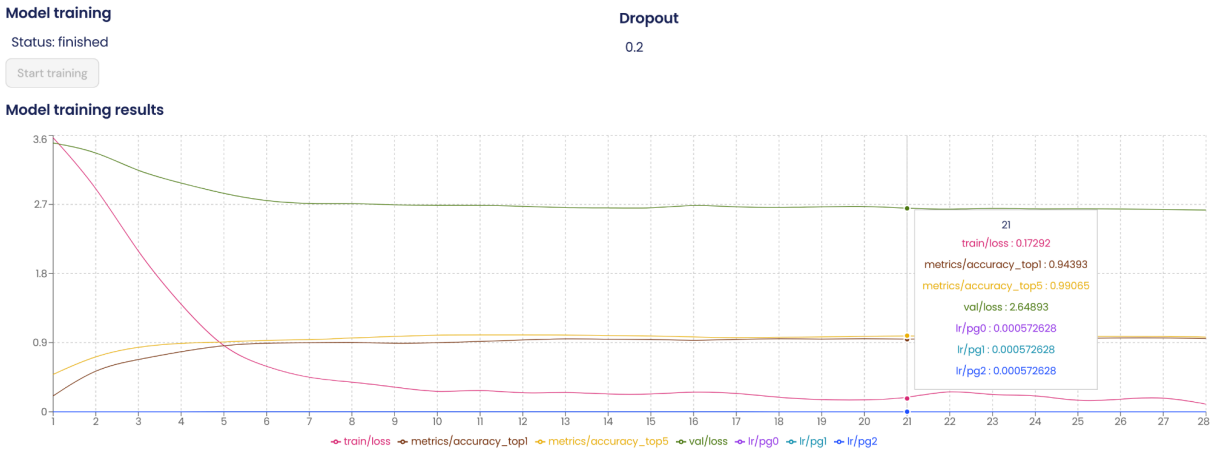


Important note about the statuses:

- **Ready -** means that you can start training.
- **Enqueued** - your model is put in the training queue. Since we the models are being trained by FIFO order you'll have to wait till the training starts
- **Training** - self explanatory
- **Finished** - your model is ready to be used
- **Failed** - sometimes model will fail during the training, which means that something needs to be adjusted before re-trying

You can start training only when the model is in Ready or Failed status.

Once the model finishes with the training you'll see a chart with the metric for each epoch.



Usually you want to have low train/loss and val/loss values. If train/loss is low but val/loss is high it means that your model is overfitted, i.e. it just remembered what you've shown it on the images, and if it'll see learned object it will detect it, but if there will be some distortion (background noise, color change, different object size) detection result will not be great