howest
hogeschool

Research Project

Installation manual

Content

Necessities

- A laptop/pc with a GPU and python & pytorch installed (GPU not necessary but recommended)
- A drawing tablet (makes life easier) or pen and paper
- Drawing program (illustrator, photoshop, paint, …)

Source code

You can find the source code in the repository linked below. The latest version will always be on the main branch, but each version has its own branch.

https://github.com/SoensJakob/Research-Project

Add extra data (optional)

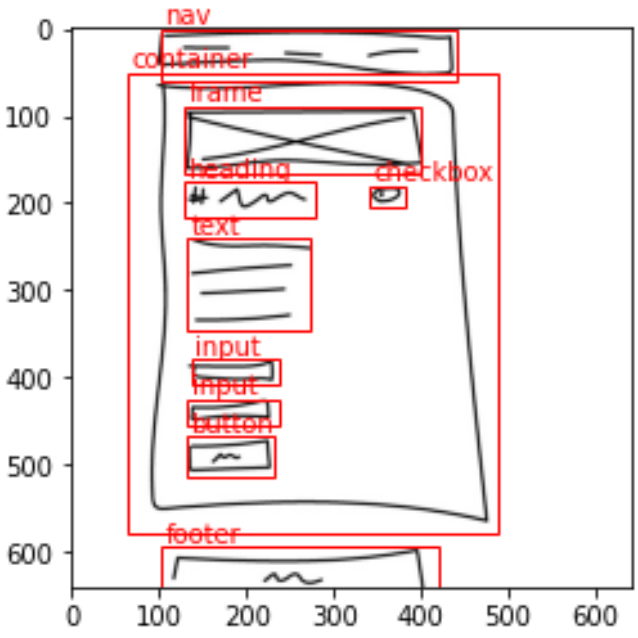
In the latest version there are 110 samples in the data folder. 85 training samples, 15 validation samples and 10 test samples in de testset map.

I made these samples with a drawing tablet in illustrator, added bounding boxes using CVAT and created the dataset using roboflow.
When creating new data. Make sure the wireframes are in a photo format such as .jpg or .png. I used a 640 x 640 canvas to draw, but you can use a different format. Thought results and model speed might be different.

When drawing wireframes make sure to use the right convention. The model will have an easier time recognizing the different elements and the accuracy will go up.

Conventies

- Navigation (nav)
- Container
- Frame
- Heading
- Text
- Input
- Button
- Footer
- Checkbox



I've added nine elements, but it is very easy to add more. Just make sure the elements are distinguishable from each other.
For more information on this TeleportHQ had a similar idea with there vision API. In the blogpost linked below they go into detail on how important the different conventions are.

https://teleporthq.io/blog/new-vision-api

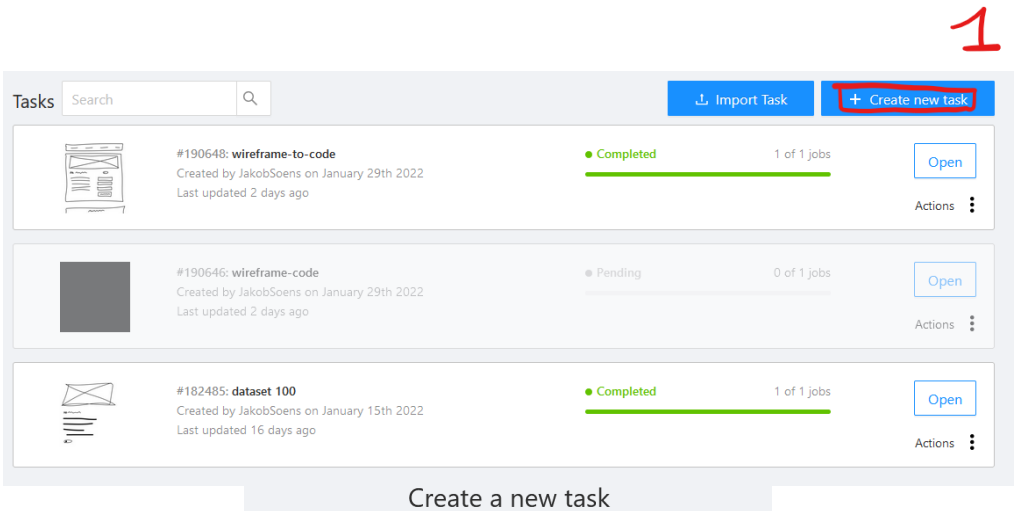
Bounding boxes

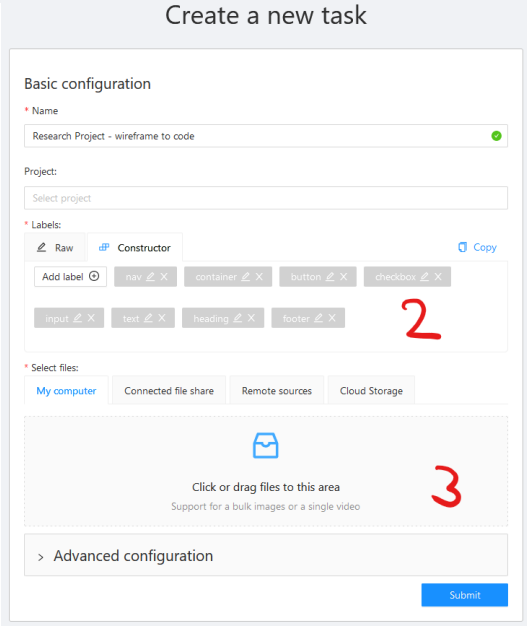
When you've made new data, you also have to add bounding boxes to this. So the model knows where all the elements are. There are many tools for this, but I used CVAT for the bounding boxes and Roboflow to create the dataset.

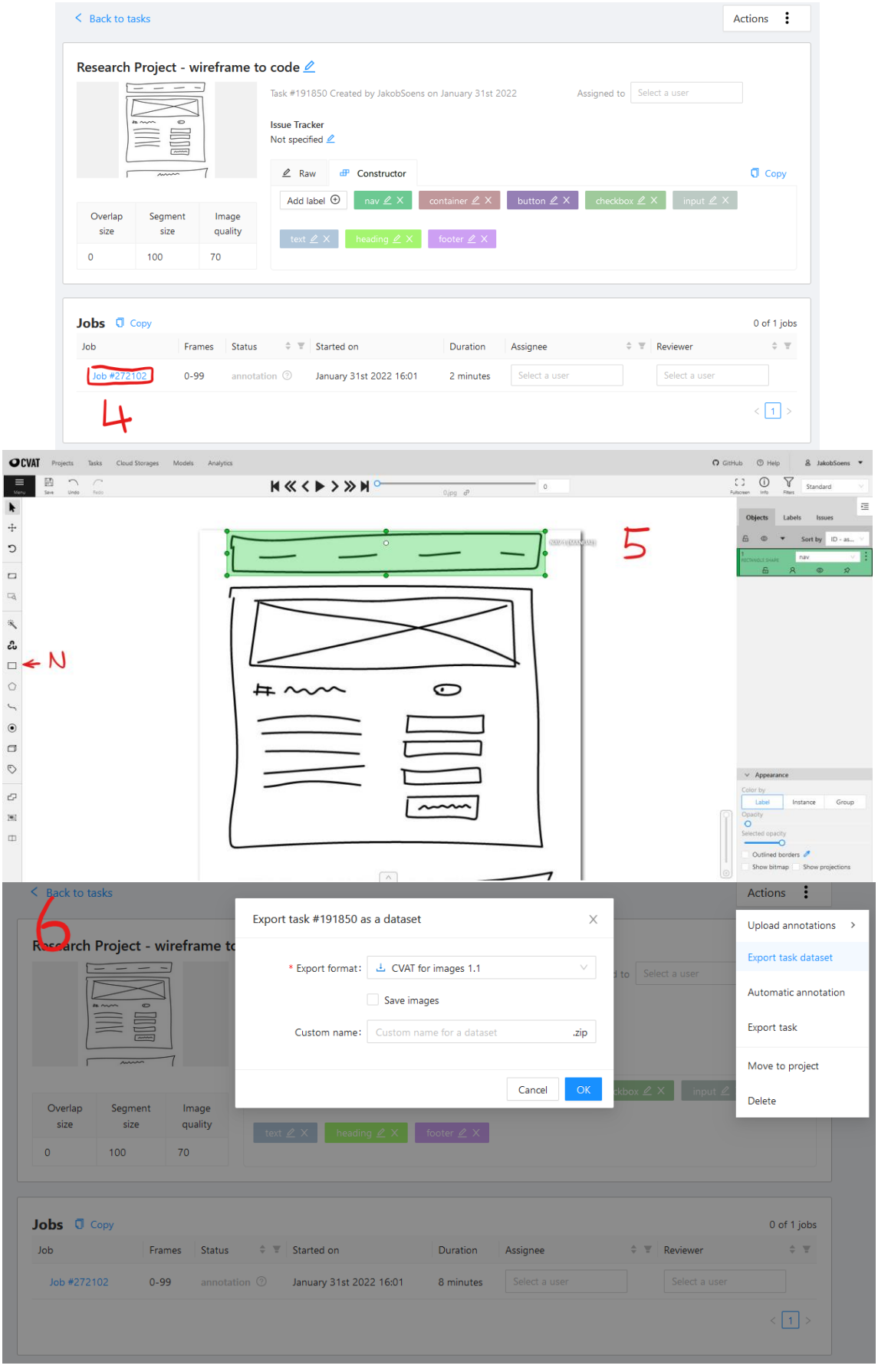
Wanneer je nieuwe data hebt aangemaakt moet je hier ook bounding boxes aan toevoegen om aan te tonen waar de verschillende elementen staan. Ik heb dit gedaan via CVAT en Roboflow.

CVAT

1. Create a new task
2. Add labels in the constructor
3. Upload the dataset and click "submit"
4. Go to tasks and click "job"
5. Now you can start creating bounding boxes
    a. Use shortcut "N" to create a new bounding box
    b. When you've drawn a bounding box around the element give it the right label
    c. Repeat for every element in the dataset
6. Wanneer alle elementen een bounding box hebben kan je de dataset downloaden op de task pagina. Kies hier "Pascal VOC"

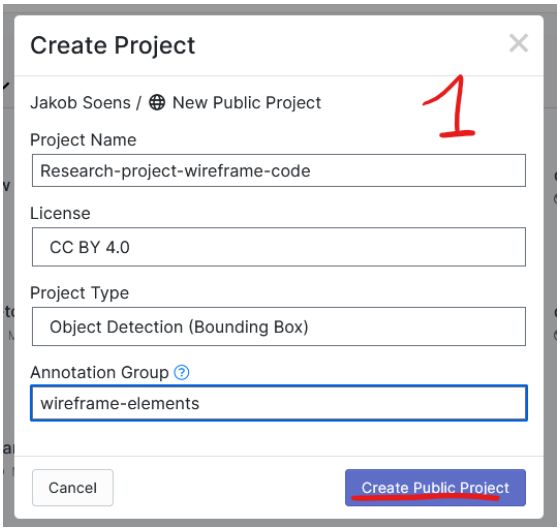


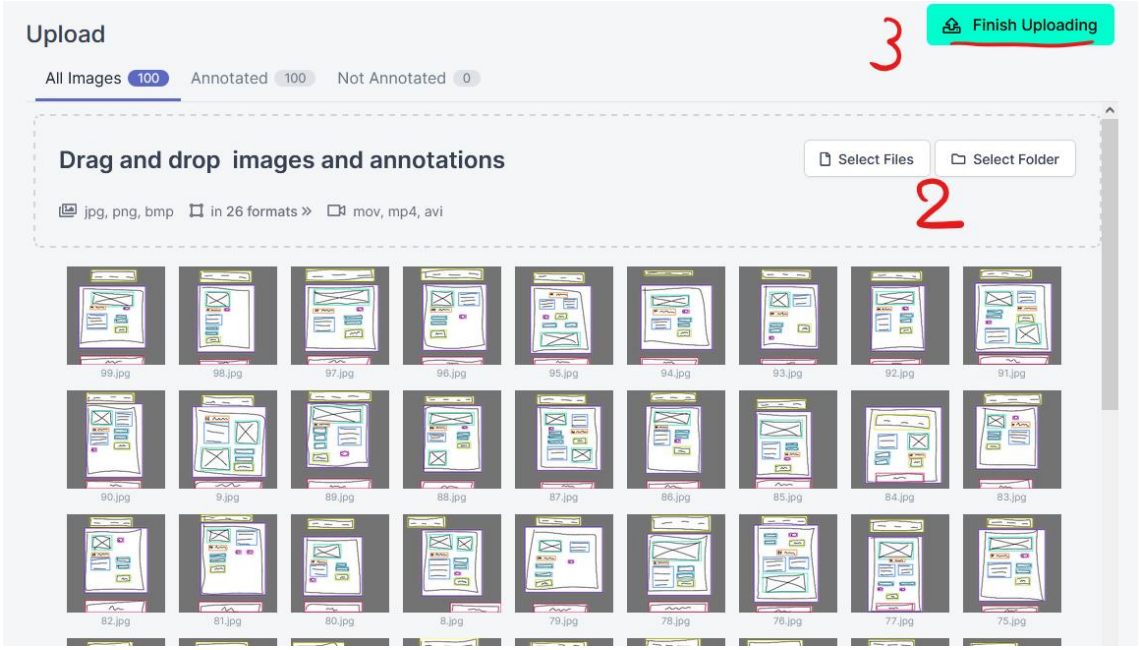


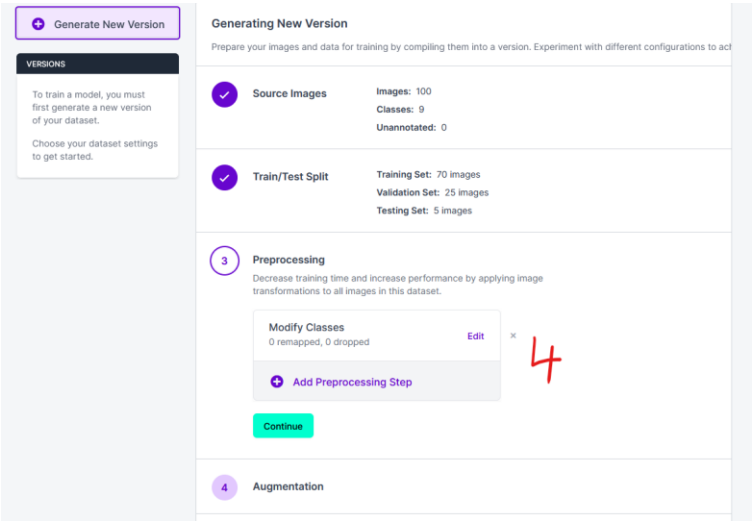

Back to tasks
Actions
Research Project - wireframe to code
Task #191850 Created by JakobSoens on January 31st 2022
Assigned to
Select a user
Issue Tracker
Not specified
Raw
Constructor
Copy
Add label
nav
container
button
checkbox
input
text
heading
footer
Overlap size
Segment size
Image quality
0
100
70
Jobs
Copy
0 of 1 jobs
Job
Frames
Status
Started on
Duration
Assignee
Reviewer
Job #272102
0-99
annotation
January 31st 2022 16:01
2 minutes
Select a user
4
CVAT
Projects
Tasks
Cloud Storages
Models
Analytics
GitHub
Help
JakobSoens
5
N
Objects
Labels
Issues
Appearance
Back to tasks
6
Export task #191850 as a dataset
* Export format:
CVAT for images 1.1
Save images
Custom name:
Custom name for a dataset
.zip
Cancel
OK
Actions
Upload annotations
Export task dataset
Automatic annotation
Export task
Move to project
Delete
8 minutes

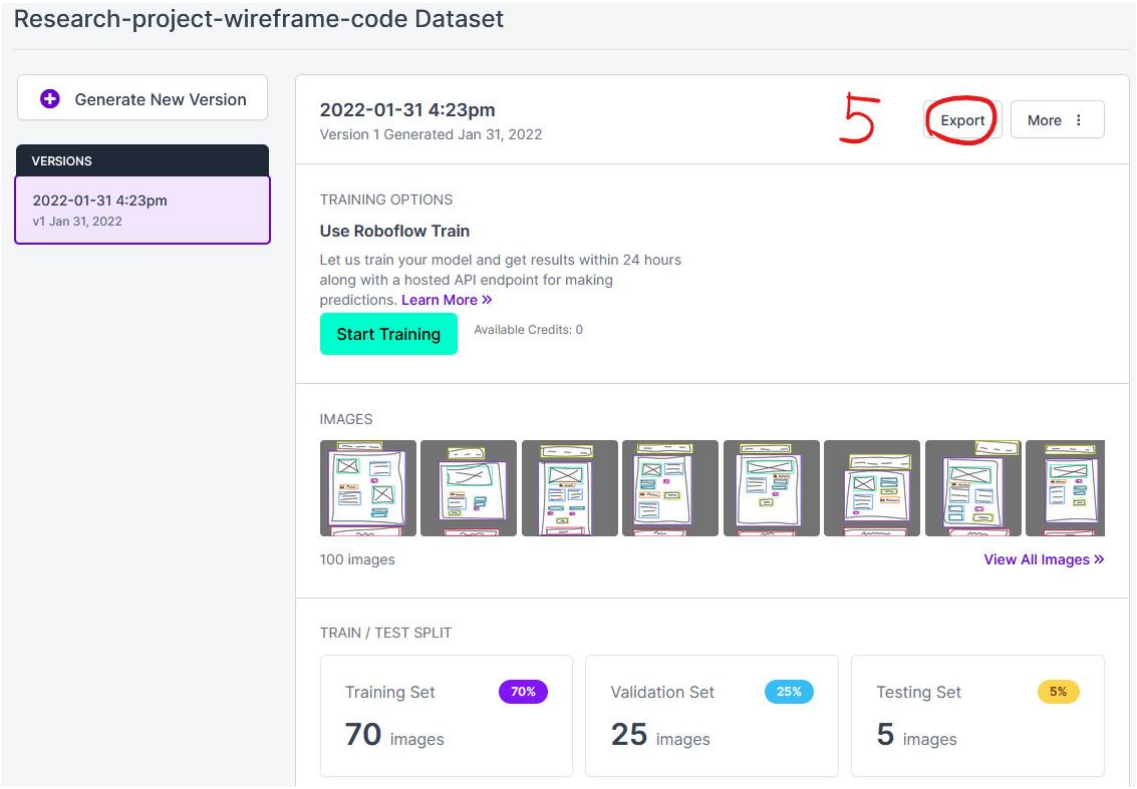
Roboflow

1. Create a new project
2. Upload the images from CVAT
3. Click “Finish Uploading” and choose train, valid and test split
4. Remove all preprocessing steps but check if the labels haven’t been renamed. You can change this by adding a modify class step then click generate
5. Export the dataset and put it in the data folder






Generate New Version
VERSIONS
To train a model, you must first generate a new version of your dataset.
Choose your dataset settings to get started.
Generating New Version
Prepare your images and data for training by compiling them into a version. Experiment with different configurations to act
Source Images
Images: 100
Classes: 9
Unannotated: 0
Train/Test Split
Training Set: 70 images
Validation Set: 25 images
Testing Set: 5 images
3
Preprocessing
Decrease training time and increase performance by applying image transformations to all images in this dataset.
Modify Classes
0 remapped, 0 dropped
Edit
Add Preprocessing Step
4
Continue
4
Augmentation


Research-project-wireframe-code Dataset
Generate New Version
VERSIONS
2022-01-31 4:23pm
v1 Jan 31, 2022
2022-01-31 4:23pm
Version 1 Generated Jan 31, 2022
5
Export
More
TRAINING OPTIONS
Use Roboflow Train
Let us train your model and get results within 24 hours along with a hosted API endpoint for making predictions. Learn More »
Start Training
Available Credits: 0
IMAGES
100 images
View All Images »
TRAIN / TEST SPLIT
Training Set
70%
70 images
Validation Set
25%
25 images
Testing Set
5%
5 images

Create weights

I used Detecto to create an object detection model. It's very simple and easy to set up.
It uses a Faster R-CNN ResNet-50 FPN pre-trained pytorch model.
For more information look at the API documentation

https://detecto.readthedocs.io/en/latest/api/index.html

Install the requirements. (Make sure you're in the correct directory)

```
pip install -r requirements.txt
```

To create weights, you just need to execute the generate_weights.py file. This will take about 15 minutes depending on how strong your computer is. Without an enabled GPU this will take a lot longer.

```
C:\Users\Jakob\Documents\Repositories\Research-Project\v2>python generate_weights.py
cuda:0
Epoch 1 of 15
Begin iterating over training dataset
  0%|                                                                                | 0/43 [00:00<?, ?it/s]C:\Users\Jakob\AppData\Local
\Programs\Python\Python39\lib\site-packages\torch\functional.py:445: UserWarning: torch.meshgrid: in an upcoming release, it
will be required to pass the indexing argument. (Triggered internally at  ..\aten\src\ATen\native\TensorShape.cpp:2157.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
 28%|████████████                                                                   | 12/43 [00:13<00:17,  1.74it/s]
```

Note

If you added new data with different elements you will need to change a line of code.
On line 27 there is a list of strings with the name of each element. Change this to whatever elements you have.

```
23  if __name__ == "__main__":
24      train_set = core.Dataset("data/train/",transform=custom_transforms)#L1
25      test_set = core.Dataset("data/valid/")#L2
26      loader=core.DataLoader(train_set, batch_size=2, shuffle=True)#L3
27      model = core.Model(["nav","frame","heading","text","checkbox","input","button","container","footer"])#L4
28
29      losses = model.fit(loader, test_set, epochs=15, lr_step_size=5, learning_rate=0.001, verbose=True)
30
31      # plt.plot(losses)
32      # plt.show()
33
34      model.save("model_weights_v2.pth")
35
```

howest
hogeschool

howest.be