RS2 output Automation – Image analysis

Prepared by Nick Mirsepassi

Date: 04/07/2025

Rev:0.0

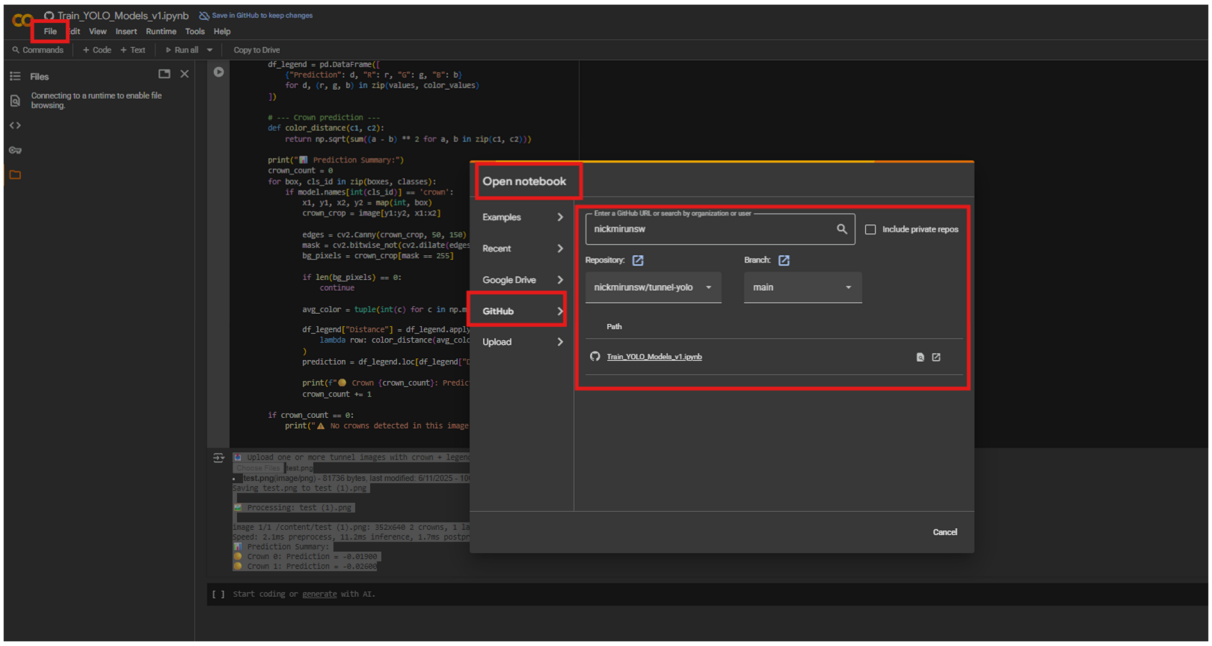
This guide walks you through training a YOLOv11s model and using it to detect crown regions in tunnel images, read the legend, and predict corresponding values. All done in Google Colab and python setup required.

What You Need

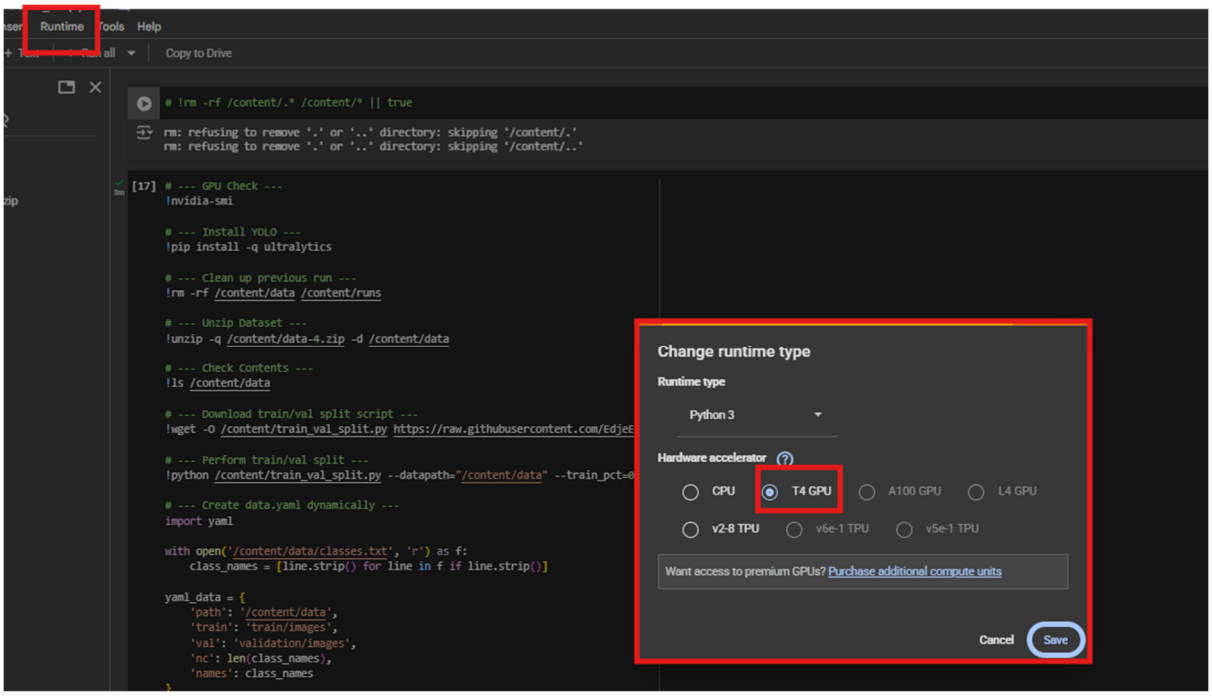
- A Google Account
- This Colab notebook :
  - https://github.com/nickmirunsw/tunnel-yolo/blob/main/Train_YOLO_Models_v1.ipynb
- These two files:
  - data-4.zip → contains training images + labels
  - matching-images.zip → contains images you want to predict

STEP 1 – Open the Notebook

Visit the GitHub link and open the notebook in **Google Colab**.



Make sure to use the Tesla T4 GPU. This model is super heavy and without GPU it will take over 10 hours to run.

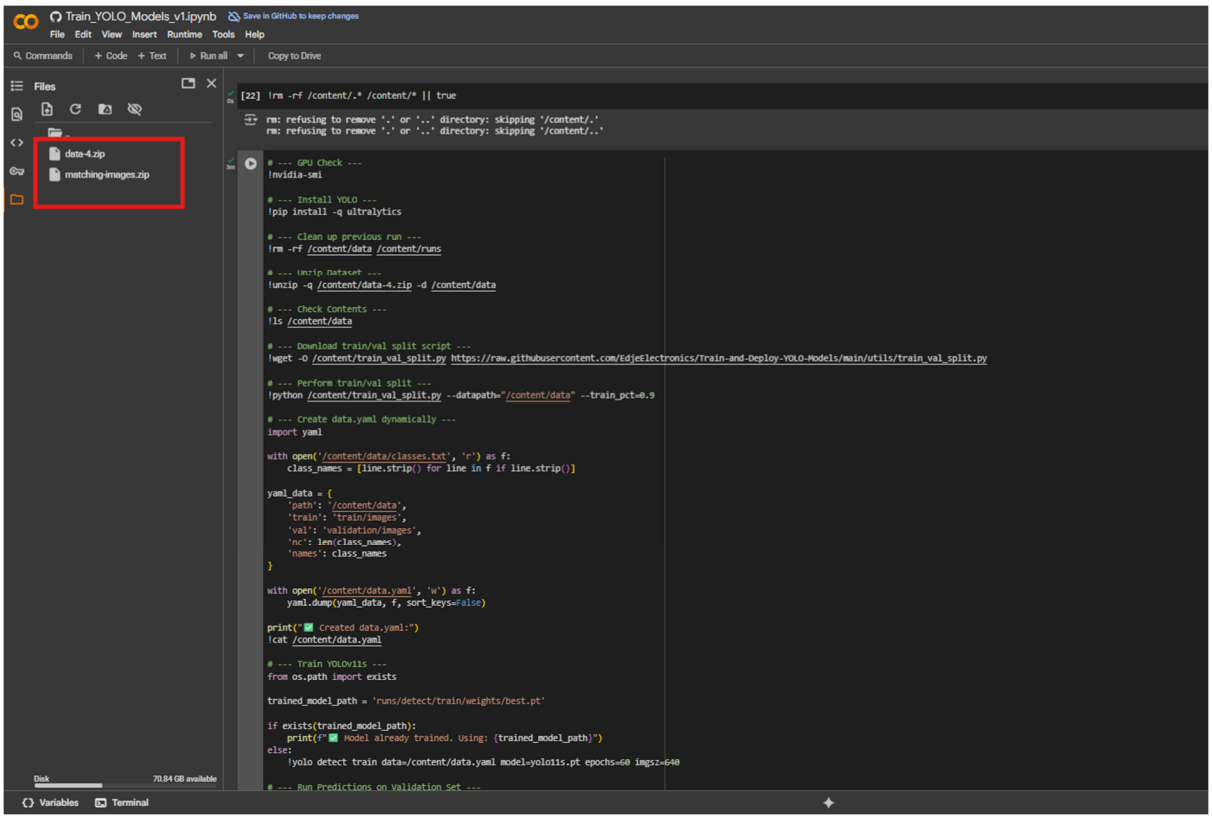


STEP 2 – Upload ZIP Files
Look to the **left-hand file browser** (📁 icon).

Drag & Drop the following files:

data-4.zip
matching-images.zip

Just like in this screenshot:
Colab will automatically upload them into /content/.



STEP 3 – Run the Setup & Training Cells

The notebook will:

- Check GPU
- Install YOLOv11 (Ultralytics)
- Unzip training images
- Auto-split into training/validation
- Generate data.yaml
- Train the model (or reuse existing one if found)

“””

You’ll see messages like:

CopyEdit

✅ Model already trained. Skipping training.

or

nginx

CopyEdit

Epoch 1/60...

“””

Let the whole block finish before moving on.

STEP 4 – Run Predictions on Matching Images

Once training is complete:

- The model will be used to predict on matching-images.zip
- It will extract:
  - Label area using YOLO (label)
  - Crown area(s) using YOLO (crown)
- The legend in the label is then read using OCR
- The average crown colour is matched to the legend

- You get a clean result printed in the terminal

You don't need to manually do anything here — just run the cells.
This step is just a validation that our training has been successful.

STEP 5 – One-Stop Prediction Tool for Engineers

Now that we have completed the training and validation, At the bottom of the notebook is a **simple user-friendly interface**:

- Allows you to **upload one or more PNG images**
- Automatically:
    - Runs the trained model
    - Extracts the label and crown
    - Matches color to legend
    - Prints prediction to terminal

✅ No Excel or CSVs generated
✅ No manual editing required

At this step it is important that you visually assess the predictions with what you see in the image and note any discrepancies as this will determine any future fine-tuning.

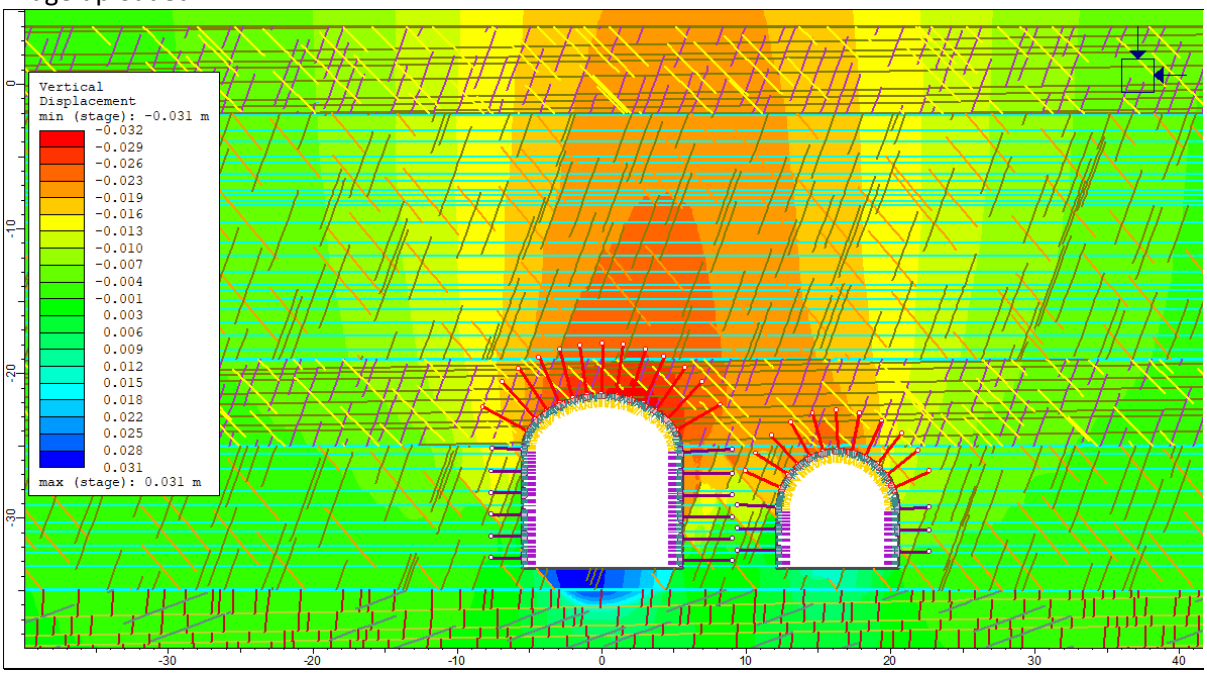
Note: I have only trained this with 140 images, for better results we will need to expand on this which will be our next step

Common errors you might encounter

Problem	**Solution**
FileNotFoundError for zip files	Check if ZIPs are properly dragged in
YOLO not detecting crowns	Confirm images have clear crown features
OCR not extracting legend properly	Ensure label box has readable text
Crashing on a file	Notebook will skip and move to next

TEST EXAMPLE

Image uploaded:



==========

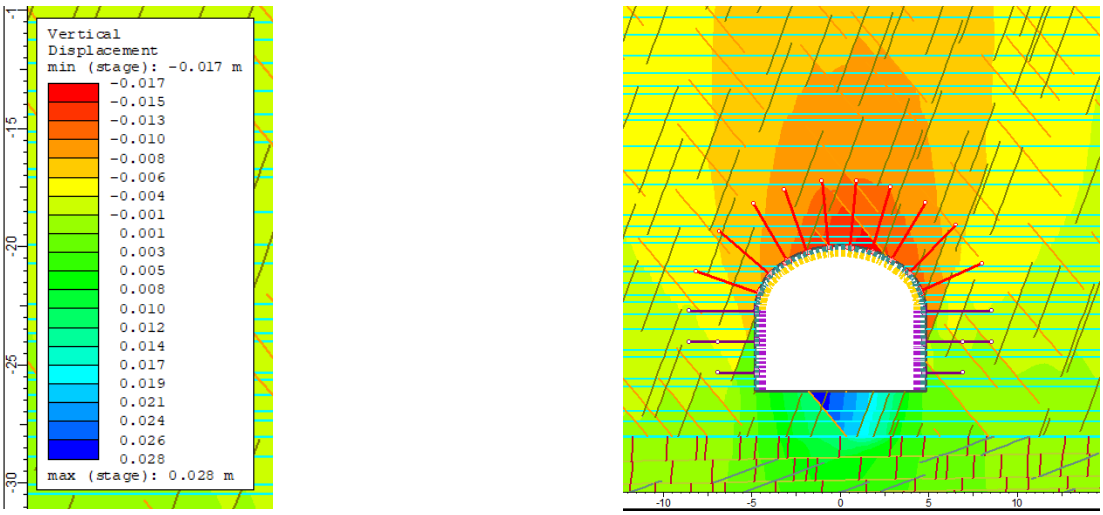
Upload one or more tunnel images with crown + legend:
test.png*(image/png) - 81736 bytes, last modified: 6/11/2025 - 100% done*
Saving test.png to test.png

Processing: test.png

image 1/1 /content/test (1).png: 352x640 2 crowns, 1 label, 2 tunnel-openings, 11.2ms
Speed: 2.1ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 352, 640)
Prediction Summary:
Crown 0: Prediction = -0.01900
Crown 1: Prediction = -0.02600
==========

ADDITIONAL NOTES

It is essential that all image outputs strictly follow the formatting conventions demonstrated in the example below. Specifically, the number format must be in standard decimal notation (e.g., 0.024) and not scientific notation (e.g., 2.4E-2), as the current version of the model is unable to process scientific formats accurately. Ensuring consistency in visual presentation is critical to enable precise extraction of values from the legend.



At present, the model automatically detects and processes two key components from each image: the colour legend label and the tunnel crown. The legend is read using OCR, while the tunnel crowns are identified using object detection. Once both components are identified, the model uses pixel-level colour matching to find the closest legend value corresponding to each detected crown. The final prediction, which represents the value matched from the legend (such as displacement or stress), is then displayed directly in the terminal or saved as part of a structured output if needed.

The underlying approach is designed to be metric-agnostic because it uses pixel colour matching rather than interpreting the values themselves, the system can work across different types of results including displacement, stress, or other analysis outputs, provided the image formatting is preserved.

Once the pipeline is verified for tunnel crown detection, the next step will be to train the model to detect additional structural elements such as sidewalls. This will allow for a more comprehensive summary of numerical modelling results. Looking ahead, we aim to scale this system further to automate processing across multiple sections extracted from full 3D FLAC models, enabling rapid analysis and report generation that integrates seamlessly with existing modelling workflows.

BACKGROUND IF OF INTEREST
What is YOLO and what we do here?

YOLO (You Only Look Once) is a real-time object detection framework that identifies and classifies objects within an image in a single forward pass through a neural network. It is widely used in computer vision tasks due to its speed and accuracy.

In this application, we have successfully trained a YOLO model to detect key elements in tunnel model outputs specifically, the tunnel opening, the crown zone, and the color-coded legend. The model is capable of identifying these components with high precision across multiple images. This detection allows us to extract useful engineering insights by linking the spatial location of tunnel elements to corresponding values in the legend, such as displacement or stress.

The potential for extending this work is substantial. As the model improves, we can introduce additional classes (e.g., sidewalls, invert, floor) to extract even more detailed structural information. With further training and refinement, this tool can become a core part of an automated pipeline for summarising outputs from numerical models—reducing manual effort and enhancing consistency in reporting.