# How to train a YOLO11n classifier for Spider Web Cracks

Below are the complete step-by-step instructions for dataset creation and model training process. You may wish to skip dataset creation steps, download ready-to-use dataset in YOLO format from [here](https://epam.sharepoint.com/:f:/s/EPAMFutureDial/EjDAq6qlIT5OtbJud5hkSmIB09Z0CwGiatxaHi_A93N74A?e=A9PLYS) and proceed directly to the model training step. However, to update that dataset with the new data collected, you will need to follow the instructions below.

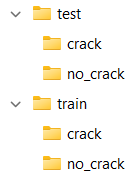
## Prerequisites:

1. Install the ultralytics package from PyPI:

**pip install ultralytics**

## Dataset preparation:

1. You need to have [Batch6](https://www.dropbox.com/scl/fo/eg69ux6zawcoiwewercwf/AD9OeokSZkYL0IuMZGbZ6-4?rlkey=gac238j9m5atcmrheerj8z69c&e=1&st=cqyj8bmt&dl=0) and other [SpiderCrack](https://www.dropbox.com/scl/fo/2duqb179kd7zixhz6elxd/APO-y88OJDyEfBhU0-zUFQ0/Spider%20Crack?rlkey=oth1aj40e98lcodivklw5nn7t&subfolder_nav_tracking=1&st=v6gok64o&dl=0) samples downloaded on the local machine.
2. Copy all sample images (\_0\_0\_ and/or \_3\_1\_) containing significant cracks/spider-web cracks to some dedicated folder. Images can be from both frontal and rear surfaces. However, do not take those which do not contain any significant cracks – minor cracks on cameras, device edges will not be detected by this model.
3. Prepare the script **YOLO\_spider\_web\_data\_preparer.py**: set BASE\_PATH to the folder, containing sample images with cracks; set output\_path where cropped sample patches will be stored. Create output\_path folder in advance!
4. Run Python script without any arguments given. The script will generate patches of selected size (320pix or 512pix) in the indicated folder (the folder must exist). Optionally, if you create in advance /random\_crops subfolder, there will also appear random scaled and shifted crops of the same image.
5. Check the contents of each patch generated, select those that contain significant cracks. This is to make sure that after patch cropping you eliminate all negative samples. Put all positive selected samples with cracks into YOLO dataset subfolder crack.
6. Repeat steps 3-5 for negative samples, i.e. samples that DO NOT contain any cracks. Put all negative patch samples without cracks into YOLO dataset subfolder no\_crack.
7. IMPORTANT: It is strongly advised to use samples of the same device type that are present for both classes. That is, if there is iPhone12White in positive class, it can be used in negative class (and vice versa). However, if there are no Samsung Galaxy samples in positive class, they cannot be used as negative samples (and vice versa). This is necessary to guarantee that the model cannot learn to ‘cheat’ on classification problem using device type or color as a simple clue.
8. Split samples from folders crack and no\_crack into train and test subsets. Create folder structure as indicated in [Image Classification Datasets Overview - Ultralytics YOLO Docs](https://docs.ultralytics.com/datasets/classify/#dataset-structure-for-yolo-classification-tasks):



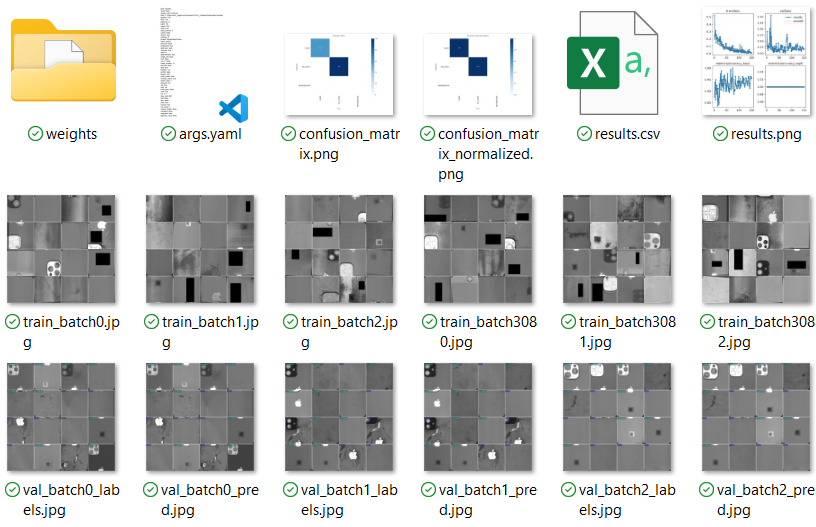
1. IMPORTANT: split into train/test should be done on original sample basis. That is, you cannot split patches/crops taken from the same original sample into both train and test subfolders.
2. The dataset is ready for training YOLO classifier.

## Model training

1. Open the script **YOLO\_spider\_web\_classifier.py**. Modify parameters section so that TRAINING = True, DATASET\_PATH points to the root folder of the prepared dataset. You may wish to also change PROJECT\_NAME variable to fit your needs.

A screen shot of a computer code

AI-generated content may be incorrect.

1. Run Python script without any arguments given.
2. Training for 50 epochs takes some time, at the end in the PROJECT\_NAME/train2 subfolder you will find complete set of artifacts including the models in weights subfolder: 

## Model testing and usage

1. Open the script **YOLO\_spider\_web\_classifier.py**. Modify parameters section so that TRAINING = False, and optionally the remaining parameters to test model either on folder with positive samples, or on any existing Batch (negative samples):

A screen shot of a computer code

AI-generated content may be incorrect.

There are parameters to select between frontal/rear surfaces, where to store collected statistics, etc.

1. The model runs on 3 cropped patches for each sample image: top, middle and bottom of the device. Patches have ~50% overlapping to make sure model doesn’t miss anything. Model outputs its confidence levels for each of three patches to stdout and stores them into statistics txt file. This has been made for future confidence level selection, based on how many correct predictions (or errors) model will get for both positive and negative samples. Currently, we propose to use threshold in range 0.5…0.9 depending on Precision/Recall preference for the model. Classification result is positive for the whole device surface if any patch has been classified as cracked.

Futher steps for model optimization and conversion to the OpenVINO format will be provided later.