Hardware Requirements: Windows. You will also need a relatively powerful computer with a NVIDIA GPU: The computer that this model was run on has a NVIDIA GeForce RTX 4090 GPU with 16GB of vRAM. The computer itself has 64 GB of RAM and the processor is a 13th Gen Intel(R) Core(TM) i9-13900H, 2.60 GHz.

Software Requirements: SonarWiz, ArcGIS Pro

Knowledge requirements: Basic proficiency in Python, Command Prompt, ArcGISPro, SonarWiz.

Notes: File names and upper/lower case must match exactly for these commands to work. In general, when working with Python, it’s best to use filenames that have no spaces.

Retraining the model requires experimentation and fine-tuning, so it may not be easy for a non-specialist to do this. Testing the model on new data, however, is a much simpler process and should be possible for someone with basic Python and geospatial skills.

**To Retrain the Model**

NOTE: This document outlines a process that will only work after completing the process outlined in the 1a\_YOLOv7InstallandTestonNewData document has been completed.

1. Annotate new sidescan sonar data that you wish to add to the training and validation datasets. These data should contain ACW. Note: Refer to file structure document in Deliverables folder for suggested file structure for data generated in this section (2bFileStructureDetailsforYOLOv7ModelTraining.docx).
   * 1. Create two folders on your hard drive: train and val. Within each of these folders create a “SonarWizMosaics” and an “ArcGISProTiles” folder. Training data is used to train the model. Validation data is the new data upon which the model is tested on to ensure that it can generalize to new data.
   1. Export test data from SonarWiz into SonarWizMosaics folder. Note: For optimal model performance, model should be trained on individual lines of sidescan sonar generated at 10 cm resolution, as opposed to mosaics composed of odd- and even-numbered rows. Below is an example of the export settings dialogue box. However, the validation data should be composed of odd- and even-numbered row mosaics, also generated at 10 cm resolution. This ensures that the model’s accuracy score is as high as possible while still being representative of the model’s ability to generalize to new data input as mosaics. The report document discusses this in greater detail.

A screenshot of a computer

Description automatically generated

* 1. Add sonar lines to ArcGIS Pro project.
  2. Adjust the symbology. Leave the first three bands (RGB) as-is. Set the alpha band to “None.” Set “Stretch Type” to “None” and all three gamma values to 1.0.
  3. Create a feature class within your geodatabase for each sonar line. The feature class should be named based on your sonar line. For example, ACW\_Mosaic\_Line for training data or ACW\_MosaicAorB for validation data.
  4. For each sonar line/mosaic, in the corresponding feature class, draw a box around each piece of ACW. Each piece must be annotated separately.
  5. Open the Geoprocessing Tool “Export Training Data for Deep Learning”
     1. For each of your lines: Select your input raster line. Create an output folder named based on the input raster and save it inside of the appropriate ArcGISProTiles folder based on whether you are generating the data to be used in model training or validation. Input Feature Class should be the corresponding feature class you just created. Image format should be TIFF.
        1. For training data, Tile Size X and Y should both be 640. Stride X and Y should both be 320. Metadata format should be KITTI. Minimum Polygon Overlap Ratio should be 0.75. Run tool. Below is an example of the export settings dialogue box

A screenshot of a computer

Description automatically generated

* + - 1. For validation data, Tile Size X and Y should both be 640. Stride X and Y should both be 640. Metadata format should be KITTI. Minimum Polygon Overlap Ratio should be 0. All other parameters are identical to those of the training data generation. Run tool. NOTE: Validation data must not appear in any form in the training dataset.

1. Format data for input into YOLOv7 model. You will need to do these steps separately for train and val folders. Note: Refer to file structure document in Deliverables folder for suggested file structure for data generated in this section (2bFileStructureDetailsforYOLOv7ModelTraining.docx).
   1. In each sonar line folder generated by the tool in the last step, you will have an images folder and a labels folder. Rename all the files in each folder by the mosaic name (CTRL A – right click – rename). You cannot have any spaces in the filenames. You will end up with a list of identically named files in the images folder and the labels folder.
   2. In your train and val folders, create three folders: a folder named “TIFF,” a folder named “TXT,” and a folder named “JPEG.”
   3. Open a Jupyter Notebook using the Preprocess virtual environment that you created in document 1a\_YOLOv7InstallandTestonNewData.docx. In the Deliverables folder, open MoveFiles.ipynb.
      1. Your root directory should be your path to ArcGISProTiles. Your target folder should be the TIFF folder you just created.
         1. Click “Run” in the menu at the top of the screen and then “Run All Cells.” All the tiff files should now be together in TIFF.
      2. Leave the root directory the same but make the target folder the TXT folder you just created. Change if name.endswith (‘tif’) to instead say (‘txt’).
         1. Click “Run” in the menu at the top of the screen and then “Run All Cells.” All the txt files should now be together in TXT.
         2. Go into the TXT file and sort them by name. Either at the top or bottom of the list you will have two files called stats.txt and map.txt. Delete these.
   4. Convert TIFF files to JPEG. In Jupyter Notebook in the Preprocess environment, open ConvertTIFtoJPEG.ipynb in the Deliverables folder.
      1. In the second text box, change the file path to the location of your TIFF folder. You will need to make sure that the path contains forward slashes (/). If it defaults to backslashes (\), then click “Edit” in the menu at the top of the screen and then “Find” and “Replace” to find \ and replace with /.
      2. Click “Run” in the menu at the top of the screen and then “Run All Cells.”
      3. Navigate to your TIFF folder and click “View.” Select “Details” to then filter by type of file. Cut all the .jpeg files and move them into the JPEG folder that you created in step 2b.
   5. Fix your txt label format.
      1. Download the most recent, stable version of Notepad ++ here: <https://notepad-plus-plus.org/downloads/>
      2. Open Notepad. In the file menu, select “Open Folder as Workspace.” Navigate to your TXT folder.
      3. In the Folder as Workspace left sidebar, right click on the name of your folder and select “Find in Files.”
         1. Find what: 1 0.00. Replace with: Debris. Check “Match whole word only.” For example:

A screenshot of a computer

Description automatically generated

* + - 1. Click “Replace in Files.”
      2. Find what: 0 0 0 0 0 0 0. Replace with: (nothing – leave box blank). Check “Match whole word only.”
      3. Find what: 0 0. Replace with: (nothing – leave box blank). Check “Match whole word only.”
      4. Find what: (4 spaces). Replace with: (1 space). Uncheck “Match whole word only.” Do the same thing for 3 and 2 spaces, replacing both with 1 space.
      5. Find what: \_Debris (Debris with 1 space in front of it). Replace with: Debris (with no space in front of it).
    1. Open Anaconda Prompt and create a new virtual environment by typing: conda create -n OIDtoolkit4. If it asks you anything, select “y” to proceed.
    2. When it finishes, type: conda activate OIDtoolkit4.
    3. Type: conda install pip.
    4. Navigate to the OIDv4\_ToolKit folder in the Deliverables folder and copy the path to it (path should end with “OIDv4\_Toolkit). Copy the path.
    5. In Anaconda Prompt, type “cd” and then a space. Then paste the path you just copied.
    6. Type: pip install -r requirements.txt. If it asks you anything, select “y” to proceed.
    7. Copy all the images in your JPEG folder. Navigate to …OIDv4Toolkit\OID\Dataset\train\Debris and paste all the JPEG images.
    8. Copy all the txt files in your TXT folder. Navigate to …OIDv4Toolkit\OID\Dataset\train\Debris\Label and paste all the TXT images.
    9. In Anaconda Prompt, type: python convert\_annotations.py.
    10. The txt files that are generated in the …train\Debris folder are now correctly formatted for YOLOv7.
    11. Close Anaconda Prompt and any other Anaconda or Jupyter windows you may have open.

1. Retraining model
   1. If you are retraining model, put your new jpeg and txt data from the last step with OID Toolkit into the “train” and “val” folders located at YOLOv7\_Custom\yolov7-custom\data.
   2. Any time this model is retrained, make sure to first delete “labels.cache” from the train and val folders.
   3. To train, type: python train.py --workers 1 --device 0 --batch-size 16 --epochs 25 --img 640 640 --data data/custom\_data.yaml --hyp data/hyp.scratch.custom.yaml --cfg cfg/training/yolov7-custom.yaml --name yolov7-custom --weights yolov7.pt
      1. Press enter
   4. Model results for train and val datasets will be output into YOLOv7\_Custom\yolov7-custom\runs\train
2. Testing new, retrained model
   1. Navigate to …YOLOv7\_Custom\yolov7-custom\runs\train\[your new, retrained model]\weights and copy “best.pt.”
   2. Paste “best.pt” in the main yolov7-custom folder and rename to “yolov7\_custom.pt.”
   3. You are now ready to run the model to test it on new data per the commands in section 5 of the Deliverables document, “1a\_YOLOv7InstallandTestonNewData.”