**YOLOv8+11-OBB with Alternative Loss Functions and Editable Source Code**

Derived from the latest (July 2025) Ultralytics’ YOLO, this custom Yolov8+11-OBB repo allows you to clone and edit the source code to accommodate custom loss functions.

To run this model along with the real-time editable source code, there are two Jupyter notebooks: 1. Training\_Validation.ipynb, & 2. Batchdetection.ipynb.

Links

* Google Docs: <https://docs.google.com/document/d/e/2PACX-1vQUAT_ui45MgeQDcBCJXXylG5c28rUqNHTVYih-ScA8FMnk_GvEKW02Vi3MLv_cS1UElM8zUpbBNXpF/pub>

**Training & Validation**

To train YOLO models against a custom dataset & custom loss function, you need to run the “Training\_Validation.ipynb” file.

!pip install supervision

!sudo apt update

!sudo apt install zip

import os

homedir = os.path.expanduser("~")

print(f"Home directory: {homedir}")

Run this block twice: once before saving the pickle file “datacar.pkl”, and once afterwards.

j = 0 #switch j between 0 & 1 to run this block twice

for i in range (j,2): #switch j between 0 & 1

if i == 0:

!pip uninstall ultralytics -y

!git clone https://github.com/Suppersine/yolo\_alternative\_loss\_functions.git

%cd yolo\_alternative\_loss\_functions

!pip install -e .

if i == 0:

import os

os.kill(os.getpid(), 9) # This will restart the runtime

else:

import sys

sys.path.insert(0, f'{homedir}/yolo\_alternative\_loss\_functions')

# Now you can import ultralytics modules

import ultralytics

from ultralytics import YOLO

print(ultralytics.\_\_file\_\_) # Should point to your cloned directory

Once done, run this block to select a loss function, then run the looped block once more.

#Extracted from picklesave.py, always run this first in the 2nd iteration (i=1) in the next block below, because the loss.py file needs to load the saved .pkl data.

%cd yolo\_alternative\_loss\_functions

import pickle

# List of available modes

modes = ["GBB", "CSL", "KLD\_none", "KLD\_sqrt", "KLD\_ln", "KLD\_exp", "KLD\_neg\_exp", "KFIOU\_dflt", "KFIOU\_ln", "KFIOU\_exp"]

# Default mode

mode = 'GBB'

print("Select a loss/IoU mode:")

for i, m in enumerate(modes):

print(f"{i+1}. {m}")

try:

user\_input = input(f"Enter the number corresponding to your choice (default: {mode}): ")

if user\_input.strip(): # Check if the input is not empty

selected\_index = int(user\_input) - 1 # Convert to 0-based index

if 0 <= selected\_index < len(modes):

mode = modes[selected\_index]

else:

print(f"Invalid number. Sticking with default mode: {mode}")

else:

print(f"No input provided. Sticking with default mode: {mode}")

except ValueError:

print(f"Invalid input. Please enter a number. Sticking with default mode: {mode}")

print(f"{mode} loss/IoU mode selected")

# Assign the selected mode (as a string) to myvar

myvar = mode

with open("datacar.pkl", "wb") as f:

pickle.dump(myvar, f)

print("Variable saved to datacar.pkl")

Once done looping, initialise the YOLO model, addresses, essential settings, etc.

#check ultralytics address

import ultralytics

from ultralytics import YOLO

print(ultralytics.\_\_file\_\_) # Should point to your cloned directory

#once the address checking is done, start training the model with (y)our custom dataset

!jupyter --paths

filedir = f'{homedir}/obb\_dataset/'

yamldir = f'{homedir}/obb\_dataset/datav11.yaml'

yolodir = f'{homedir}/yolov11-OBB/'

homedir = str(f'{homedir}/')

Then, we can commence training the model.

# Load a model (N-submodel)

model = YOLO("yolo11n-obb.pt") # load a pretrained model (recommended for training)

!yolo task=obb mode=train model=yolov8n-obb.pt data={yamldir} epochs=300 imgsz=640 plots=True

Finally, to end the “Training\_Validation.ipynb” file & save weights, we will run the validation block. Adjust the nmod integer to count how many submodels you have trained so far.

#validation & test

nmod = 4

for i in range (nmod): # where nmod is the number of models trained

if i == 0:

weightaddress = str(f'{yolodir}runs/obb/train/weights/best.pt')

else:

weightaddress = str(f'{yolodir}runs/obb/train{i+1}/weights/best.pt')

model = YOLO(weightaddress)

val = model.val(data=yamldir)

print("Overall mAPs")

print(val.box.map) #50/95

print(val.box.map50) #50

print(val.box.map75) #75

print("classwise mAPs")

print(val.box.maps)

!zip -r weights.zip runs

**Batch Detection Test**

Check environmental readiness:

#in case zip is not found in the shell environment

!sudo apt update

!sudo apt install zip

!pip install supervision

import os

homedir = os.path.expanduser("~")

print(f"Home directory: {homedir}")

Check essential addresses. If the double-nested loop fails, check again.

# Check addresses of Jupyter paths

!jupyter --paths

pretrainedwt = True

filedir = f'{homedir}/cardiacyv8/'

yamldir = f'{homedir}/cardiacyv8/datav11.yaml'

datadir = f'{homedir}/obb\_dataset/test/images/'

yolodir = f'{homedir}/yolov11-OBB/'

homedir = f'{homedir}/'

#!pip install zip

#!sudo apt-get update && apt-get install -y zip

Unless there are already pretrained weights in the HDD, run this block to unzip uploaded weights, which must be uploaded to {HOMEDIR}/weights/.

if not pretrainedwt:

if not os.path.exists(f'{homedir}/'):

os.makedirs("weights") #then upload the weights.zip file to the weights directory

%cd weights/

!unzip "\*.zip"

weightdir = f'{homedir}/weights/'

else:

print("Weights directory already exists. Skipping creation.")

%cd weights/

!unzip "\*.zip"

weightdir = f'{homedir}/weights/'

else: #in case you want to detect with newly trained weights.

weightdir = yolodir

%cd {homedir} #cd back to home dir

Then, check the Ultralytics repo directory.

#check ultralytics address

import ultralytics

from ultralytics import YOLO

print(ultralytics.\_\_file\_\_) # Should point to your cloned directory

If absent, rerun the looped blocks\*, which are structurally identical to those of the training file.

\*You can find them on the 1st and 2nd pages of this document.  
Next up, using a GUI file explorer (in Jupyter or COLAB’s left panel), make sure you count the total number of subfolders within {weightdir} to plan loop nesting.

# Define the weight directory (kept for context from the original snippet)

# weightdir = "/sample\_data/"

wtdir = f'{weightdir}runs/obb/' # Assuming 'weightdir' is defined elsewhere in the notebook

# print(wtdir)

# Execute the find command and capture its output

# -maxdepth 1: Only look in the specified directory

# -mindepth 1: Don't count the specified directory itself

# -type d: Only consider directories

# -name "train\*": Match directory names that start with "train"

# The 'capture\_output=True' flag ensures the output is returned

# The .stdout.decode().strip() part gets the string output, decodes it, and removes whitespace

command = f"find {wtdir} -maxdepth 1 -mindepth 1 -type d -name 'train\*' | wc -l"

print(command)

import subprocess

result = subprocess.run(command, shell=True, capture\_output=True, text=True)

# Convert the captured output to an integer and store it in 'subf'

# The output from wc -l is a string, so we strip any whitespace and convert to int

try:

subf = int(result.stdout.strip())

print(f"Number of 'train' subfolders found: {subf}")

except ValueError:

subf = 0

print("No 'train' subfolders found or error parsing output.")

except Exception as e:

subf = 0

print(f"An error occurred: {e}")

After planning to loop, we will proceed with batch detection.

#batch detection test

import os

from pathlib import Path # Improved file path handling

import random

#from ultralytics import YOLO

import supervision as sv

import cv2

k = 0

for i in range(subf):

k += 1

if i == 0:

weightaddress = str(f'{weightdir}runs/obb/train/weights/best.pt')

else:

weightaddress = str(f'{weightdir}runs/obb/train{i+1}/weights/best.pt')

model = YOLO(weightaddress)

# Define the test image directory

test\_dir = Path(f"{datadir}") # Use Path for better handling

# Loop through all image files

for image\_path in test\_dir.glob("\*.png"): # Modify extension if needed (e.g., \*.png)

# Perform detection

results = model(str(image\_path)) # Convert Path to string for YOLO

# Extract filename

filename = image\_path.name

# Process and annotate detections (same as your code)

detections = sv.Detections.from\_ultralytics(results[0])

oriented\_box\_annotator = sv.OrientedBoxAnnotator()

annotated\_frame = oriented\_box\_annotator.annotate(

scene=cv2.imread(str(image\_path)),

detections=detections

)

# Define output path (modify as needed)

output\_path = Path(f"{yolodir}/results{k}/{filename}")

# Save the annotated image (replace with desired format if needed)

cv2.imwrite(str(output\_path), annotated\_frame)

# Optional: Print progress

print(f"Detection results saved for: {filename}")

Finally, we will zip the detection subfolders to save on your HDD. So that you can view the results on your machine/device.

# Create a list of the directory names

%cd /home/u3618315/yolo\_alternative\_loss\_functions/

results\_dirs = [f"{yolodir}results{i+1}" for i in range(0, subf)]

# Join the directory names into a single string, separated by spaces

dirs\_to\_zip = " ".join(results\_dirs)

# Construct the full zip command

zip\_command = f"zip -r yolov11\_detections.zip {dirs\_to\_zip}"

# Execute the command using Jupyter's shell magic

!{zip\_command}

print(f"Created yolov11\_detections.zip containing: {dirs\_to\_zip}")

If you want, clear your HDD.

for i in range(0,subf):

!rm -r "results{i+1}"