YOLOv5-Face – WIDER-Face Training & Deployment Guide

# 1. Overview

This document explains the complete workflow for training and deploying a compact YOLOv5‑Small model for face detection, tailored to the WIDER‑Face dataset and an Apple‑Silicon MacBook (MPS backend). It covers project structure, installation, training, evaluation, live demo, confidence‑threshold tuning, fine‑tuning, and troubleshooting.

# 2. Project Folder Structure

Diana YOLO/  
├── datasets/  
│ └── wider\_face.py # PyTorch dataset loader & augmentations  
├── models/  
│ └── yolo.py # Compact YOLOv5‑Small architecture  
├── utils/  
│ └── boxes.py # Box utilities (IoU, NMS, conversions)  
├── train.py # Main training script (strides 4/8/16)  
├── eval\_metrics.py # Correct mAP/Precision/Recall computation  
├── detect.py # Image / folder inference  
├── webcam\_detect.py # Real‑time webcam demo  
├── conf\_sweep.py # Confidence sweep (precision/recall/f1)  
├── weights/ # Saved weights & logs  
│ ├── best\_val.pth # Best validation‑loss checkpoint  
│ ├── best\_map.pth # Best mAP50 checkpoint  
│ └── train\_log.csv # epoch, train\_loss, val\_loss, mAP50  
└── runs/  
 └── detect/… # Saved detection results

# 3. Installation (macOS, Apple‑Silicon)

1. \*\*Create & activate a virtual environment\*\*  
 ```bash  
 python -m venv .venv  
 source .venv/bin/activate  
 ```  
2. \*\*Install core libraries\*\* (PyTorch ≥ 2.2 compiled with MPS):  
 ```bash  
 pip install torch torchvision torchaudio --upgrade  
 pip install opencv-python tqdm matplotlib (oder mit requirements.txt)  
 ```  
3. \*\*Download WIDER‑Face\*\* and place folders `WIDER\_train`, `WIDER\_val` plus the MAT‑split files (`wider\_face\_train.mat`, `wider\_face\_val.mat`) in the project root.

# 4. Training

Launch a full training run (≈ 100 epochs ~ 30 h on M2, batch 8):  
```bash  
python train.py --epochs 200 --batch\_size 8 --img\_size 640 --lr 0.01  
```  
Key points:  
\* Anchors & strides are fixed to \*\*(4, 8, 16)\*\* – perfectly aligned with the tiny model.  
\* Training logs every epoch: train‑loss, val‑loss; every 5 epochs: mAP50, Precision, Recall.  
\* Two checkpoints saved automatically:  
 \* `weights/best\_val.pth` – lowest validation loss.  
 \* `weights/best\_map.pth` – highest mAP50.  
\* Early stopping: after 35 epochs without val‑loss improvement.  
\* `weights/loss\_curve.png` shows the loss trajectory.

# 5. Inference & Demo

### Single image / folder  
```bash  
python detect.py --weights weights/best\_map.pth --source path/to/img\_or\_dir --conf 0.01  
```  
Results are saved in `runs/detect/`.  
  
### Live webcam:  
```bash  
python webcam\_detect.py --weights weights/best\_map.pth --conf 0.02 (selbst besten Wert aus png suchen)  
```  
Press \*\*Q\*\* to exit.

# 6. Choosing the Best Confidence Threshold

Run a confidence sweep on the validation split to maximise \*\*F1‑score\*\*:  
```bash  
python conf\_sweep.py --weights weights/best\_map.pth --min\_conf 0.001 --max\_conf 0.5 --steps 100 (Auch mal mit anderen .pth probieren ergibt gute Bilder in BA)  
```  
Outputs:  
\* `sweep\_results.csv` (conf, P, R, F1)  
\* `f1\_vs\_conf.png` – peak indicates optimal threshold.  
  
Typical outcome (trained run): best F1 ≈ 0.55 @ conf ≈ 0.013–0.02.

# 7. Metric Cheat‑Sheet

\* \*\*Precision (P)\*\* ‑ fraction of detections that are correct: TP / (TP + FP).  
\* \*\*Recall (R)\*\* ‑ fraction of GT faces that were recovered: TP / (TP + FN).  
\* \*\*F1\*\* ‑ harmonic mean of P & R (balances the two).  
\* \*\*AP50\*\* ‑ area under P‑R curve, IoU ≥ 0.5; \*\*mAP50\*\* is mean across classes (one class ⇒ same).  
  
Lower conf ⇒ high recall, lower precision. Higher conf ⇒ opposite.

# 8. Fine‑Tuning on Your Own Faces

1. Collect & label your own images (LabelImg or Roboflow).  
2. Create a dataset loader similar to `datasets/wider\_face.py`.  
3. Start from the pre‑trained weight:  
```bash  
python train.py --weights weights/best\_map.pth --epochs 50 --lr 0.001  
```

# 9. Troubleshooting

\* \*\*mAP = 0\*\* – use the fixed `eval\_metrics.py` (normalisation bug before).  
\* \*\*Negative strides numpy error\*\* – call `.copy()` before `torch.from\_numpy` (done in scripts).  
\* \*\*Slow MPS\*\* – batch size > 8 may swap; monitor Activity Monitor ➜ Memory.

# 10. Quick Start Commands

```bash  
# full training  
python train.py --epochs 200 --batch\_size 8  
  
# best model inference  
python detect.py --weights weights/best\_map.pth --source demo.jpg --conf 0.01  
  
# webcam demo  
python webcam\_detect.py --weights weights/best\_map.pth --conf 0.005  
  
# confidence sweep  
python conf\_sweep.py --weights weights/best\_map.pth --steps 40  
```