A diagram of a training process

AI-generated content may be incorrect.

**Phase 1 – Prototype & Feasibility**

**Goal:** Get a quick baseline model working  
**Steps:**

1. **Pick a Pretrained Model** – Start with something like YOLOv8, YOLO-NAS, or SSD with person class from COCO.
2. **Collect a Small Dataset** – 200–500 images with people in varied poses, lighting, and backgrounds.
3. **Label Data Quickly** – Use tools like [LabelImg](https://github.com/heartexlabs/labelImg), [Roboflow Annotate](https://roboflow.com), or CVAT. Label just bounding boxes at first.
4. **Train with Transfer Learning** – Freeze backbone layers, fine-tune detection head only.
5. **Evaluate Quickly** – Measure mAP@0.5 and inference speed on validation set.

**Output:**

* A working but not perfect people detector.
* Initial benchmarks (accuracy, FPS).

**Phase 2 – Dataset Expansion & Model Refinement**

**Goal:** Improve robustness and reduce false positives/negatives.  
**Steps:**

1. **Expand Dataset** – Collect 2–5k images including edge cases:
   * Crowds
   * Different angles
   * Partial occlusions
   * Low light / night
2. **Data Augmentation** – Use:
   * Random scaling & cropping
   * Motion blur
   * Lighting changes
   * CutMix / Mosaic augmentation
3. **Model Choice Adjustment** – Switch to a larger variant (YOLOv8m/l, Faster R-CNN) if accuracy is more important than speed.
4. **Progressive Unfreezing** – Train with more layers unfrozen for better feature adaptation.

**Output:**

* mAP improved by 10–20% over Phase 1.
* Handles moderate occlusions and background noise.

**Phase 3 – Domain Specialization & Tracking**

**Goal:** Adapt to your exact deployment environment.  
**Steps:**

1. **Collect Environment-Specific Data** – Same camera angle, resolution, and lighting as deployment site.
2. **Fine-Tune** – Train starting from Phase 2 weights with this specialized dataset.
3. **Introduce Tracking** – Use DeepSORT, ByteTrack, or OC-SORT for person tracking across frames.
4. **Test in Field Conditions** – Evaluate on live streams, edge devices, or cloud pipelines.

**Output:**

* Reliable detection in your real-world setup.
* Stable tracking IDs over time.

**Phase 4 – Optimization & Production**

**Goal:** Make it fast, light, and deployable.  
**Steps:**

1. **Model Pruning & Quantization** – Use TensorRT, ONNX, or TFLite for optimization.
2. **Benchmark on Target Hardware** – Jetson, Coral, Raspberry Pi, or CPU/GPU.
3. **Edge-Case Retraining** – Feed failure cases back into dataset.
4. **Add Privacy Layers** – Blur faces or silhouettes if needed.

**Output:**

* Production-grade people detector.
* Meets latency, accuracy, and compliance requirements.

**Visual Summary – Phased Approach Pipeline**

**Data → Label → Baseline Model → Expanded Training → Domain Fine-tuning → Optimization → Deployment**

Here’s the **People Detection Model Selection Matrix (Apache 2.0 / MIT Licensed Only)** so you can deploy on edge devices **without GPL or dataset license complications**.

**Commercial-Safe People Detection – Edge Model Selection Matrix**

| **Edge Hardware** | **Recommended Models (Apache 2.0 / MIT)** | **Expected FPS(640×640)** | **Model Size** | **Notes** |
| --- | --- | --- | --- | --- |
| **Tiny & Low-Power**(ESP32-CAM, Pi Zero, Cortex-M MCU) | - MobileNet SSD (TFLite)- PP-PicoDet-S | 1–5 FPS | 0.5–4 MB | Small, low-power models; run on TFLite Micro or Paddle Lite |
| **Mid-Range Edge AI**(Pi 4, Coral TPU, Jetson Nano) | - NanoDet-Plus- EfficientDet-D0- PP-YOLOE-S | 10–25 FPS | 4–20 MB | EdgeTPU-friendly variants available |
| **High-End Edge AI**(Jetson Xavier NX, Orin Nano/NX) | - PP-YOLOE-M- EfficientDet-D1- NanoDet-Plus (High Res) | 30–60 FPS | 10–50 MB | Can run at higher input resolutions for better accuracy |
| **Specialized Accelerators**(Jetson AGX, FPGA, VPU) | - PP-YOLOE-M (TensorRT)- EfficientDet-D2- NanoDet-Plus (TensorRT) | 45–100+ FPS | 20–80 MB | Use TensorRT or FPGA toolchains for acceleration |

**Why These Models Are Safe**

* **License** → Apache 2.0 or MIT (no copyleft like GPL)
* **Pretrained Weights** → Free for commercial & research use
* **Training Code** → Fully open source