Group X – Car Damage & Repair-Cost Estimation

You have one Colab-week and want insight, not SOTA.

---

0 What the baseline tells us

The 15-epoch "plain CNN" reaches **75 % training accuracy but only 5 % validation accuracy**—classic over-fit or data-leakage .
Before touching architectures, rule out trivial pitfalls:

Check	How	Time
Leakage	Verify that the **same image (or near-duplicate)** is never in both train & val. Roboflow occasionally preserves directory order when re-splitting. Quick SHA-1 hash check or `imagehash` library.	15 min
Label balance	Print per-class counts in each split; target ≥ 10 samples/class in val.	10 min
Image corruption	`PIL.Image.open` inside try/except over all files.	5 min

If any of these fail, redo the 80/10/10 split with **stratify-by-class + shuffle** locally (don't rely on Roboflow's web UI) and re-run one epoch to confirm val-acc > chance.

---

1 Free-tier-friendly experiment grid

ID	Hypothesis	Implementation	Expected Δ	GPU h
A	Pre-trained features generalise better	**ResNet-50** backbone, freeze first 2 stages, lr = 1e-3 on head, 10 epochs	↑↑ val-acc	0.6
B	Better localisation aids classification	**YOLOv8-n** object detector on COCO-format boxes; evaluate mAP@0.5	↑ mAP	2.0
C	Cost is correlated with mask area	Use YOLO damage boxes → compute relative area, fit **LightGBM regressor** for €	↑ ρ, ↓ MAE	0.1
D	Over-fit stems from small set	**Albumentations** pipeline: ColorJitter, RandomShadow, MotionBlur, Cutout	↑ val-acc	0.2
E	Hard classes need focal loss	Swap CE → **Focal γ = 2** in ResNet run	↑ recall for rare damage	0.1

Pick 2-3 that feel most doable; even negative results count if discussed.

---

2 Implementation notes (do once, reuse)

- **Mixed precision** (`torch.cuda.amp`) + batch 16 saves 40 % VRAM on a Colab T4.
- **Gradient accumulation** 4 → effective batch 64 without OOM.
- Cache resized (640×640) training images to `/content/drive/…` to survive 12-h resets.
- For YOLOv8: `yolo detect train data=data.yaml model=yolov8n.pt imgsz=640 epochs=50 lr0=0.01 frozen=10`.

---

3 Cost-estimation branch in one week

1. **Rule-based baseline**:
   mean labour rate €50 / h, part price lookup → gives you an MAE you can beat quickly.
2. **Two-headed network** (stretch goal):
3. `shared CNN → GAP → [Softmax damage-type] + [ReLU-Dense-1 cost €]`

   Loss = $\alpha \cdot CE + \beta \cdot MSE$ (start $\alpha=1$, $\beta=0.1$).

4. **Ordinal buckets**: "<€500 / €500-1500 / >€1500" makes learning easier; treat as 3-class CE.

---

4 Regularisation & diagnostics

- **Early stopping** on val-loss patience = 5.
- **Label smoothing** $\epsilon = 0.05$ (works well with class imbalance).
- **Grad-CAM** on ResNet's final conv to verify it focuses on dents/scratches, not backgrounds.
- After each run, log: confusion matrix, per-class F1, PR-AUC; export as PNG for the final report.

---

5 Suggested 7-day schedule

Day	Goal
1	Sanity checks, redo split if needed, rerun baseline (1 epoch)
2	Implement & train **ResNet-50 head** (Exp A)
3	Add heavy **Albumentations** + Focal loss (Exp D, E)
4-5	Fine-tune **YOLOv8-n** on damage boxes (Exp B)
6	Derive cost features + LightGBM regressor (Exp C)
7	Error analysis, Grad-CAM visuals, write Milestone 2

---

6 Report-writing tips

- Present a **single table** comparing baseline vs. each experiment: Acc, mAP, MAE.
- Include one Grad-CAM heat-map and one YOLO prediction collage.
- Discuss *why* over-fitting shrank after transfer learning—tie back to limited data.
- Keep raw training curves in the appendix.

Quick wins checklist

- Verify no split leakage
- Freeze early layers of ResNet-50
- Use mixed-precision + accum 4
- Add ColorJitter & Cutout
- Log Grad-CAM for at least 3 samples

Good luck!