AI Project: Build a Small End-to-End Project

Put it all together: pick a real problem, gather data, train a small model, test

Introduction

You have now met the whole toolkit: neural networks (DL.1), the training loop (DL.2), activations and building blocks (DL.3), CNNs for images (DL.4), and sequences and language (DL.5). A certificate does not come from reading about these — it comes from finishing one small project end to end, with all its messy steps: a real problem, real data, a trained model, an honest test, and something you can show someone.

This lesson gives you one project plan you can actually follow this week, plus a second option if you prefer text over images. Both plans are deliberately small: a few hundred examples, a small model, an evening or two of work. The goal is not a record-breaking score. The goal is a finished pipeline you understand, and a short write-up that says what worked, what did not, and what you would do next.

The plan follows six steps in order: **choose the problem, gather and check the data, split it honestly, pick a small model, train and watch the curves, test and show**. Skip none of them: every skipped step in real projects returns later as a mysterious failure.

Where beginners get stuck: scope creep in week one. The project that fails is usually not the one with too little ambition — it is the one that tries to classify 50 animals, in two languages, with a live website, before a single photo has been collected. The plans in this lesson are small on purpose. Finish the small version first, with its honest test score and five-line write-up. Only then decide what to add. A finished small project teaches more than an unfinished big one, and it is the one you can show someone. *(Roman Urdu: Asli seekhna aik chhota project poora karne se hota hai — masla chuno, data jama karo, model train karo, imandari se test karo, aur natija dikhao.)*

Key points

- **Pick a problem you can check with your eyes.** Image classification ("which of these 5 flowers is this?") is ideal: you can look at a photo and know the right answer. Problems you cannot verify yourself are hard to debug.
- **Data first, model second.** A few hundred clean, correctly labelled examples beat thousands of sloppy ones. Look at your data before training: open 20 random examples and check their labels by hand.
- **Split before you touch the model.** Put 20% of the data aside as a test set first, and never train on it (ML.3's rule). Your final score comes only from that untouched set.
- **Start with transfer learning, not from scratch.** For images: a pre-trained CNN, early layers frozen, a new classifier head (DL.4). For text: a pre-trained language model, fine-tuned (DL.5). You will finish in hours instead of days.
- **Watch two curves, not one.** Training loss should fall; validation loss should fall too. If training loss falls while validation loss rises, you are overfitting — add dropout, augmentation, or stop early (DL.2, DL.3).

- **Test honestly and write five lines.** Report: what the model does, its test score, one example it gets right, one it gets wrong (and why you think so), and what you would try next. That short write-up is what turns homework into a portfolio piece.
- **Show it simply.** A one-page demo (a photo upload → the model's answer) or a short video of you using it beats a 50-page report nobody reads.
- **Keep a simple baseline and beat it.** Before the fancy model, build the simplest thing that could work (word counts for text, a tiny classifier for images) and record its score. If the deep model only beats it by 2%, that is useful knowledge — and it protects you from celebrating a score any simple method could reach.
- **Save your work as you go: data, code and checkpoints.** Keep the exact train/test split, the model weights at the best epoch, and a short log of what you changed. When something breaks later, you can go back to the last version that worked instead of starting again.
- **Get consent for other people's photos and words.** If your dataset includes photos of people or their reviews, use your own material or material you have permission to use. A portfolio project built on someone else's private photos is a problem, not a showcase.

Examples

Example 1 — Project A: flower classifier (images, recommended)

- **Problem:** given a photo, name one of 5 local flowers.
- **Data:** 60 photos per flower (300 total) taken with your phone, in different light and angles. Check 20 labels by hand. Split: 240 train, 60 test, before any training.
- **Model:** a pre-trained CNN (the exact one matters less than the method), early layers frozen, a new 5-class head. Add simple augmentation: flips and small rotations.
- **Training:** watch both curves for ~20 epochs. Expect training loss to fall fast; if validation loss rises after epoch 10, stop there or add dropout.
- **Test:** run the untouched 60 photos. Suppose you score 52/60 (87%). Look at the 8 misses: maybe two flowers look alike in shade — write that down as a real finding, not a failure.
- **Show:** a one-page demo where a visitor uploads a flower photo and sees the top prediction with confidence, plus your five-line write-up.

Example 2 — Project B: review mood checker (text)

- **Problem:** given a short shop review, say happy or unhappy.
- **Data:** 400 short reviews you write or collect (200 happy, 200 unhappy), labelled by hand. Split 320 train / 80 test first.
- **Model:** a small pre-trained language model, fine-tuned on your reviews (DL.5's transfer trick). A simple baseline first — word counts + a classic classifier — gives you a score to beat (ML series habit: always build the simple baseline).
- **Test:** run the untouched 80 reviews. Compare against the baseline honestly: if the fancy model only beats it by 2%, say so — that is a finding, and it teaches you when big models are not worth it.

- **Show:** a text box demo: type a review, see happy/unhappy plus confidence, plus your five-line write-up.

Example 3 — The five-line write-up (use this template)

1. What it does: "A flower classifier for 5 local flowers, from phone photos."
2. Test score: "87% on 60 untouched test photos (52 correct)."
3. One it gets right: "A bright sunflower in sunlight — confident and correct."
4. One it gets wrong: "A shaded rose confused with a shaded tulip — both dark red in shade."
5. Next step: "Add 50 more shaded photos per flower and retrain; try unfreezing one more CNN layer."

Example 4 — Project C: spoken command checker (audio, a third option)

- **Problem:** given a one-second voice clip, say whether the speaker said "yes" or "no".
- **Data:** record yourself and two friends saying "yes" and "no" 100 times each, in quiet and noisy rooms — 400 clips total. Split 320 train / 80 test before any training.
- **Model:** a small sequence model (DL.5) over the sound's features, or a tiny CNN over the sound picture (a spectrogram, which is just an image of sound — DL.4 again). A classic classifier on sound features is your baseline.
- **Test:** run the untouched 80 clips. Suppose you score 74/80 (93%). Check the 6 misses: maybe noisy-room "no" clips get confused with "yes" — that tells you to record more noisy examples, not to build a bigger model.
- **Show:** a demo page with a record button: speak, and it shows "yes" or "no" with confidence, plus your five-line write-up.

Example 5 — Debugging diary: when the curves turn against you

You train the flower classifier from Example 1 and log the loss each epoch:

| Epoch | Training loss | Validation loss | |------:|--------------:|----------------:| | 5 | 0.80 | 0.85 | | 10 | 0.45 | 0.50 | | 12 | 0.35 | 0.52 | | 15 | 0.20 | 0.65 |

At epoch 10 both curves are falling nicely; after that, training loss keeps falling but validation loss rises. Diagnosis: overfitting, starting around epoch 10. Fix, in order: (1) keep the epoch-10 weights (early stopping, DL.2); (2) add dropout and more augmentation (flips, rotations — DL.3, DL.4); (3) only if that fails, collect more photos, especially of the flowers the model confuses. Write this diary entry in your project log — it is exactly the kind of honest debugging a portfolio reviewer wants to see.

Example 6 — Real world: small shops reading their own reviews

A small online shop gets hundreds of short reviews a week and cannot read them all. A review mood checker like Project B sorts them into happy and unhappy, so the owner reads the unhappy ones first and fixes real problems — a late delivery, a wrong size — before they become lost customers. The model is small, trained on the shop's own past reviews, and its weekly unhappy list is a business tool, not a research demo. That is the pattern this whole series builds toward: a small model, trained on real local data, solving one concrete job.

Quick recap

- A finished small project beats a half-finished big one.
- Order: problem → data → honest split → small model (transfer learning) → watch both curves → test → show.
- Check labels by hand, split before training, and never test on training data.
- Report honestly: score, one success, one failure with a reason, and a next step.
- A one-page demo plus a five-line write-up is a portfolio piece.

Common mistakes

1. **Starting with the model instead of the data** — weeks on architecture, minutes on data, then confusion when it fails. Fix: spend the first session only on data: collect, look, label.
2. **Testing on data the model trained on** — a 99% score that means nothing. Fix: split first, lock the test set away, touch it once at the end.
3. **Hiding the failures** — a write-up that only shows successes teaches nobody, including you. Fix: always include one miss and your best guess at its cause.

1. **Peeking at the test set, then changing the model** — if you adjust the model because of a test score, the test set is no longer untouched and the next score means less. Fix: tune on the validation set only; touch the test set once, at the very end.
2. **Adding more data without checking its labels** — 500 new photos with 100 wrong labels make the model worse, not better. Fix: hand-check a random sample of every new batch before it joins the training set.

Practice

1. Pick Project A or B and write your six-step plan in six lines: problem, data, split, model, training watch, test + show.
2. You have 100 photos and a friend says "train a CNN from scratch". Write two sentences explaining why transfer learning is the better plan.
3. Fill in the five-line write-up template for an imaginary project (any score, one success, one failure, one next step). Keep each line to one sentence.

1. Pick Project C and write its six-step plan in six lines, including the baseline you would build first and the split you would use.
2. Your validation loss starts rising at epoch 8 while training loss keeps falling. Write the three fixes you would try, in order, and say which one costs no new data.
3. A friend shows you a project with "99% accuracy" but no test set and no write-up. Write three sentences explaining why you cannot trust that number, and what you would ask them to show you.

Practice quiz

1. What is the first step of the project plan?

- Pick the model
- Pick a problem you can check with your own eyes
- Train for 100 epochs
- Write the report

Answer: Pick a problem you can check with your own eyes

2. Why must the test set be split off before training?

- To save time
- So the final score comes from data the model never trained on
- It is a tradition
- To make the dataset smaller

Answer: So the final score comes from data the model never trained on

3. Training loss falls but validation loss rises. What is happening?

- The model is perfect
- Overfitting — it memorises training data instead of learning the pattern
- The data is too easy
- The learning rate is too small

Answer: Overfitting — it memorises training data instead of learning the pattern

---

Watch the video lesson: https://youtu.be/9laZxE8zyDU

Read online: https://scopetreasure.com/notes/dl-ai-project · More free tools and notes at https://scopetreasure.com