Lecture 15 · Part III · Extended notes

ML system design interview: crop disease from a phone photo

Dataset: A mock interview with a deep dive, no dataset

Why these notes exist

This lecture sits outside Géron, the course's primary text. For it *these notes are the primary source*, and it is examined on exactly the same terms as the textbook lectures. Everything in the slide deck is here, with the arguments written out at length rather than compressed onto a slide; the numbers are the ones the accompanying notebook prints, and every one of them is a measurement on the dataset named above rather than a figure quoted from a paper.

Contents

1 The format, and the deep dive **2**

2 The prompt and the overview **2**

2.1 Step 1 — Clarify . . . 2

2.2 Step 2 — Frame the task . . . 3

2.3 Steps 3 and 4 — Metrics and labels . . . 3

3 The deep dive: what the model looks at **3**

3.1 Solution A — Features engineered by hand . . . 3

3.2 Solution B — Learned end to end . . . 4

3.3 The choice . . . 5

4 The rest of the system **6**

5 Follow-ups and red flags **6**

6 Exercises **7**

1 The format, and the deep dive

The format is the one of the harmful-content interview (Lecture 9): one open prompt, the candidate drives, the interviewer plays the product manager, and every choice is said as a trade-off. A real interview has two halves. First the candidate sketches the whole system, a sentence or two per step. Then the interviewer picks one box and digs — "tell me more", "what if", "why not the other way" — until the candidate reaches the edge of what they know. The deep dive is where senior and junior candidates separate: anyone can name a technique, few can say when it breaks.

This lecture runs the overview at interview pace and goes deep on one box, **the representation**: who decides what the model looks at — the engineer, by designing features, or the network, by learning them end to end. Two complete solutions are built and set against each other, and the interviewer then asks for a choice.

How to survive a deep dive

Answer the question asked, then say what it costs. When you do not know, say how you would find out: which experiment, on which data.

2 The prompt and the overview

"We are a farmers' cooperative. A farmer photographs a sick leaf with a phone. Design the system that tells them what disease it is."

2.1 Step 1 — Clarify

The questions that change the design: which crops and diseases (the classes, and whether two look alike); where the photo is taken and by whom (a lab table or a field — the hardest fact for any representation); whether there is a connection in the field (on the phone or on a server, so the size of the model); what the farmer does with the answer (a wrong treatment costs a harvest); who checks a doubtful case (whether the app may abstain); which labelled photos exist.

The interviewer's answers used in the lecture: tomato, potato and grapevine, about twenty diseases between them plus "healthy", some of them look-alikes (early and late blight, downy and powdery mildew); photos taken in the field on whatever phone the farmer owns, in sun or shade, often with soil, hands and other leaves in the frame; often no connection, so the diagnosis runs on the phone; the app shows the likely diseases with the cooperative's treatment advice, and an agronomist answers the cases it sends on; a few thousand field photos from past consultations, labelled by the agronomist who answered and mostly of the common diseases, and a large public set of single leaves shot on a plain table in a lab. Two facts decide the deep dive: the photos are

taken in the field, and the rare diseases have few field labels.

2.2 Step 2 — Frame the task

Input: the photo and the crop the farmer selected, with region and date as context. Output: a probability for each disease of that crop and for "healthy", one softmax per crop. The agronomists label the main disease of a leaf, so one answer per leaf is an assumption worth saying aloud. Decision: show the three most likely diseases with their treatments; if the top probability is low, or the photo is not a usable leaf, **abstain** and send it to an agronomist. An app that always answers is a red flag: a softmax names a disease even for a blurred photo, and a confident wrong diagnosis sends a farmer to spray the wrong product.

2.3 Steps 3 and 4 — Metrics and labels

Offline, recall per disease, the dangerous ones read first; top-three accuracy, because the app shows three; mistakes weighted by what they cost, since two diseases with the same treatment are a cheap confusion and a fungus called a virus is not; and accuracy against coverage — how right the app is on the cases it answers, against how many it sends on. Online, the agronomists' verdicts on a sample of answered cases.

The consultation photos carry a selection bias: farmers ask when they are worried, so severe cases are over-represented. Agronomists disagree on the look-alikes; a slice labelled twice measures their agreement, the ceiling for any model. The lab set is plentiful and taken in conditions the app will never see. Split by farm and by season, never by photo: one farm sends many photos of the same field, and a random split puts that field on both sides.

3 The deep dive: what the model looks at

	The pipeline	**Who decides what is measured**
A · features by hand	photo → find the leaf → measure colour, spots, texture, shape → a small classifier on a few dozen numbers	the agronomist and the engineer, in advance
B · end to end	photo → a convolutional network → one probability per disease	gradient descent, from the labelled photos

Solution A is how plant pathology was automated before convolutional networks; solution B is what most such apps ship today. Neither is obsolete: each wins under conditions that can be stated, and stating them is what the interviewer is waiting for.

3.1 Solution A — Features engineered by hand

Find the leaf. Convert the photo to a colour space in which "green" is the range of one channel (hue, or the a* axis of Lab), threshold it, and keep the largest connected region. Inside the leaf, separate the lesions the same way: brown, yellow, black and white pixels that are not healthy

green. Correct the white balance, so that the same yellow under sun and shade gives the same numbers. Every feature is measured on this mask, so a wrong mask makes every feature wrong, and the classifier is never told.

Measure what an agronomist looks at.

Group	Measured as	The symptom it reads
Colour	histograms of the lesion pixels' colour; the share of the leaf that is yellow	chlorosis; the brown of blight against the white of mildew
Spots	number of lesions, their sizes, how round they are, concentric rings	the target-shaped spots of early blight
Texture	statistics of local intensity patterns inside the lesions	the powdery film of powdery mildew; the mosaic of a virus
Shape and position	leaf outline and curl; lesions on the margin or along the veins	leaf curl; spots bounded by the veins
Severity	lesion area over leaf area	how far it has gone

On a few dozen numbers and few labels, the classifier is logistic regression, or gradient-boosted trees for the interactions between colour and spot shape.

The push: a real photo. Three leaves, soil, a thumb, the sun behind them. The colour threshold keeps the green of the other leaves and drops the brown dead tissue, which is the disease; the largest region is the wrong leaf. Backlight turns healthy green to yellow and the chlorosis feature fires on a healthy plant; soil is brown and spotted, and the spot counter counts it. The strong answer names the first step as the one that fails and fixes the capture before the model: the app asks for one leaf against the sky or a sheet of paper, and a quality check refuses dark, blurred or cluttered photos. The user interface is part of the feature engineering.

What it buys and what it costs. It needs few labels — tens of examples per disease for a few dozen numbers, not hundreds — explains itself ("large brown spots with rings", a severity in percent), runs on any phone, and fails in ways that can be debugged one feature at a time. Its ceiling is what the agronomist could name in advance; its capture is brittle, since every feature inherits the segmentation's mistakes; a new disease means new features written and tested by hand; and look-alikes that differ in ways nobody wrote down stay confused.

3.2 Solution B — Learned end to end

The model. A few thousand field photos cannot train a network from scratch (Lecture 12); a backbone pretrained on a large image collection already knows edges, textures and blobs. Probe first — a linear head on frozen features, the number to beat — then fine-tune the last blocks (Lecture 13). The backbone is one designed for mobile devices, chosen by measuring accuracy against size, because it runs on the phone. An early lesion is a few pixels once the photo is shrunk to the network's input, so a small detector finds the leaf and the classifier sees the crop (Lecture 14): B keeps A's first step, done by a learned detector instead of a colour threshold. Output: one head per crop, or the crop as an input; a softmax, and abstention below a threshold.

The push: augmentation. An augmentation is a claim that the label does not change, and it may be made only where an agronomist would agree. Flips and rotations keep the label (a leaf has no "up"), and so do random crops and scale (farmers hold the phone at every distance) and mild changes of brightness, contrast and blur (sun, shade, phones that do not focus). Hue shifts and strong colour jitter do not: yellow against green *is* the symptom, and shifting it relabels the photo. Random erasing needs care, since it can erase the only lesion and leave a "diseased" label on a healthy leaf. Augmentation belongs on the training loader only: an augmented validation set no longer measures the model.

The push: a near-perfect test score. Trained and tested on the lab set, the network scores almost perfectly. That is no reason to ship: the test photos share the lab's table, light and sessions, often one session per disease. A network takes whatever separates the classes most cheaply; if the background or the light differs between diseases it learns those, a **shortcut**, and a test set that shares the shortcut cannot see it. This has happened in published work: near-perfect on lab photos, far lower on photos from the field. To find out, test on field photos split by farm and season, paste the leaves onto other backgrounds and see whether the answer changes, and check with the explanation tools of Lecture 10 whether the network attends to the lesion or to the table.

What it buys and what it costs. A higher ceiling: it finds cues nobody wrote down and separates look-alikes better. Robustness to clutter and light — if, and only if, it was trained on photos that have them. A new disease is new labelled photos, not new code, and one pipeline serves every crop. It costs labels (hundreds of field photos per disease to fine-tune well, where the rare diseases have tens); shortcuts learned silently; no explanation a farmer can read and no severity for free; a larger model on the phone, and a retraining pipeline to run.

3.3 The choice

The interviewer asks for one, with one engineer and one season. Each fact of the prompt favours one side: cluttered field photos on any phone favour B, because A's segmentation fails first and everything inherits it; look-alike diseases favour B, because the difference lies in cues nobody has named; few field labels, with tens for the rare diseases, favour A or a pretrained B, because few labels favour few parameters to fit; running offline on the phone favours A or a small B; advice that must be explained, with a severity, favours A.

A strong answer

Ship B: a pretrained mobile backbone fine-tuned on the field photos, with the lab set used only as extra training data and never as the test. Keep A's pieces where they win: the capture guide and the quality check in front, the leaf found before it is classified, the lesion area shown as severity. A is the baseline B must beat, on field photos split by farm, disease by disease; a rare disease on which B does not beat A keeps A's answer or goes to an agronomist. The choice would switch to A if the photos were taken at a fixed station — same light, plain background — and labels were scarce.

Red flags in the deep dive

"Hand-made features are obsolete", or "networks are a black box, so hand features are safer", with no conditions; "the network learns its own features, so the data does not matter"; a lab test score reported as field accuracy; a hue augmentation where colour is the label; training from scratch on a few thousand photos; B with no baseline to beat; no condition under which the choice would change.

4 The rest of the system

Step 7 — Thresholds. The abstention threshold is set with the agronomists from accuracy against coverage: as many answers as the cooperative will stand behind. Check calibration per crop first, since a threshold on uncalibrated probabilities means nothing (Lecture 4).

Step 8 — Serving. On the phone, offline; the model ships with the app and is updated with it; photos and answers upload when there is signal.

Step 9 — Launch and monitoring. Shadow first: the app answers, an agronomist answers too, and the two are compared; then one region at a time. A new season, a new disease or a new phone's colour processing moves the inputs: watch the abstention rate and the agronomists' sample. The abstained cases come back labelled, but only the hard ones, so a random sample of answered cases is labelled as well.

The whole system. Photo with the crop selected → capture guide and quality check → a small detector finds and crops the leaf → a fine-tuned mobile network, one probability per disease of that crop → above the threshold the top three with treatments and the lesion area as severity, below it an agronomist → the agronomists' answers and a random sample become labels → retrain each season, tested on the newest season by farm, disease by disease, against the hand-feature baseline.

5 Follow-ups and red flags

Typical follow-ups: a new disease reaches the region (agronomists first; in A new features, in B labelled photos and a new output, measured before it is shown); a farmer photographs a fruit or the dog (the quality check and the detector refuse it; a softmax always names a disease, so abstain); the cooperative adds olives (a new head, with A as the first baseline); a phone brand saturates its colours (a slice by phone model, white balance at capture, photos from that phone in training); two agronomists disagree (a third one or a lab test, and the agreement rate as the ceiling).

Red flags an interviewer writes down: designing before asking where the photos are taken; an app that always answers; accuracy on a random split of photos; lab photos as the test of a field system; one solution named with no baseline and no condition under which the other would win;

augmentations that change the label; a model too big for the phone it must run on; stopping at training, with no capture guide, no abstention and no feedback loop.

6 Exercises

The five set at the end of this lecture's deck, in the style of the written paper. The solutions are on Lecture 16's deck and in the exercise book.

1. A crop-disease classifier is fine-tuned on a few field photos per disease, with random hue shifts among its augmentations. Say why this augmentation is wrong here, and state the rule it breaks. [4]

2. A network trained and tested on a large public set of leaves photographed one at a time on a lab table reaches a near-perfect test score. Explain why this does not tell you how the app will do in the field, and say how you would find out. [5]

3. Solution A finds the leaf by thresholding green pixels, then measures colour, spots and texture on that mask. Name the step that fails first on a cluttered field photo, why the classifier cannot notice, and two ways to fix it. [4]

4. Two cooperatives want the same diagnosis app. One photographs leaves at a fixed station with the same light and a plain background, and has tens of labelled photos per disease. The other has farmers photographing in the field, and hundreds per disease. Recommend hand-made features or an end-to-end network for each, and justify both. [5]

5. The diagnosis app ends in a softmax over the diseases of the selected crop. Say why it must be able to abstain, and what sets the abstention threshold. [4]

Summary

A second mock ML system design interview, with a deep dive on the representation. The task, crop disease from a farmer's phone photo, framed as one softmax per crop with abstention, measured by recall per disease, top three and accuracy against coverage on field photos split by farm and season. Two solutions: features engineered by hand — find the leaf, measure colour, spots, texture and shape, a small classifier — which needs few labels and explains itself but inherits every mistake of its segmentation and is capped by what can be named; and a pretrained mobile network fine-tuned end to end, with the leaf cropped first and only label-preserving augmentations, which has the higher ceiling but needs more field labels and learns shortcuts a lab test cannot see. The choice: the network, built on the hand pipeline's capture step, severity and baseline, with the conditions under which the hand features would win.