Building Embedded Vision Products: Management Lessons from the School of Hard Knocks

Phil Lapsley
Vice President
Edge AI and Vision Alliance

Management Lessons From The School of Hard Knocks

- In my role with the Edge AI and Vision Alliance, I see and talk to lots of folks about their embedded vision products, and have gotten to work on many of them myself.
  - **Expert: n. One who has made all the mistakes there are to make in a sufficiently narrow field.**
- I'm not an expert, but I've made my share of mistakes
- I wanted to share some lessons about embedded vision projects
  - When to use vision?
  - What are the management pitfalls?
  - What can you do about them?
- I'll mostly focus on management lessons, not technical ones

Say you have a problem.

Say you have a problem.

You say: “I know! I’ll use embedded vision!”

Say you have a problem.

You think: “I know! I’ll use computer vision!”

Now you have two problems.

(With apologies to Jamie Zawinski, who originally made this joke regarding regular expressions.)

Why on Earth Would You Use Computer Vision?

- Seriously, computer vision is hard.
- It's harder still when it's embedded computer vision.
- Even with all the new tools and chips and algorithms and examples we have today, it's still hard.
- Why on earth would you voluntarily choose embedded vision to solve your problem?

Probably Because Whatever You Want to Do, You Can't Do It Without Computer Vision

Create an Entirely New Product

Something that has never existed before, *and* isn't possible without computer vision.

This is really quite rare.

Improve an Existing Product

Adds features to an existing product that wouldn't be possible without computer vision.

This is much more common.

Two Examples of Product Improvements with Vision

June Oven
Recognizes food and cooks it perfectly
$850

Roomba j7+
Avoids dog and cat poop
~~$800~~ *Priceless*

Vision Needs to Benefit Both You and Your Customer

Benefits to Customer

Reduces operating cost

Improves quality

Improves safety

Improves efficiency and scale

Improves convenience

Improves user experience

Benefits to You

Allows you to charge a premium

Differentiates you from competition

(Or allows you to keep up with the competition!)

Creates new markets and applications

Lesson #1: Be Really Clear Why You're Using Vision

Generally, vision needs to be:

- Either the only way, or the best way, to do something
- It has to add significant value for your customers
- It has to add significant value for you

If you can't crisply articulate why you're using vision, and what value it brings both you and your customers, that's a warning sign.

Lesson #2: Close Collaboration Between Marketing and Engineering is Key

To get this clarity, engineering and marketing need to work together.

- What features are needed?
- What features are practical?
- What are the requirements?

Difficulty: sometimes we may not know what's possible or practical!

Dilbert, February 20, 1994

Lesson #3: It's Going to Take Longer Than You Think

Why is this?

- Inductive vs. deductive process
- The real world is a pain in the you know what
- Uncertainties and degrees of freedom multiply

Inductive vs. Deductive Logic

Deductive Logic

- Deductive logic works on facts and rules, with certainty
- "All birds have wings. Swans are birds. Therefore all swans have wings."

Inductive Logic

- Inductive logic generalizes rules from examples, with probability
- "Every swan I've seen is white, therefore probably all swans are white."

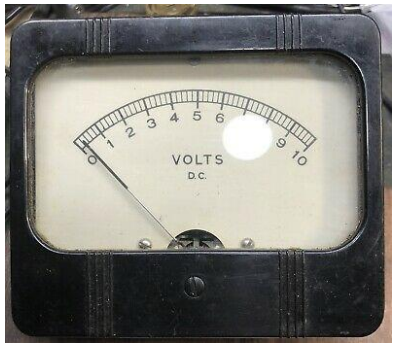
We have little (10-20 years?) experience with inductive logic in computer systems, vs. ~60 years experience with deductive logic in computer systems.

Inductive systems require examples (training data) and it's hard to know how much training data you need or how long training will take; we don't have formal rules for this, and are just starting to develop rules of thumb for it.

Thanks to Chris Rowen for this insight!

The Real World is a Pain in the You Know What

- Lighting
  - Daytime, nighttime
  - Glare from light sources (expected and unexpected)
- Indoors: steam, smoke
- Outdoors: rain, fog, snow, smoke
- Dirt, condensation on lens
- Animals, bugs
- Camera position fixed or variable?
- Will there be a later model of this product where the camera position changes?



Uncertainties in Degrees of Freedom Multiply

	Uncertainty/Variability	
Degree of Freedom	**Low**	**High**
Nature of object's appearance	Always looks identical (e.g., a manufactured 12 mm bolt)	Organic, highly variable (e.g., banana, dogs)
Multiplicity in class	Single member (e.g., just one bolt)	Many (e.g., breeds of dogs)
Lighting	Controlled	Outdoor (dark, light, glare, ...)
Positioning of object	Controlled	Random (can be anywhere, at any orientation)
Positioning of camera	Controlled	Random (human snapshot)
Camera lens	Clean	Clean or dirty
Optical environment	Controlled	Uncontrolled, may be obscured (e.g., by smoke or steam)
Background	Controlled	Random

Lesson #4: Understand the Cost of Being Wrong

- Your vision system is going to be wrong
  - More than you'd like
  - Maybe a lot
- What does that cost you?
  - For an ADAS system, it might cost you a lot
  - For a smart oven application, it might not.
- Is a human in the loop?
  - Humans in the loop can go a long way
  - System still has to deliver value and not be annoying
- Marketing and user experience input is key here

Lesson #5: Choose Your Metrics Wisely

- What metrics matter for your vision system?
  - Speed?
  - Accuracy?
    - Accuracy of what?
    - Will Glaser of Grabango: "Revenue Accuracy"
- Are these metrics simple enough to explain to company executives?
- Still another example where engineering needs to work closely with marketing

Lesson #6: Begin with the End in Mind

- "Let's get a bunch of training data and start training a model!"
- No. No no no.
- Let's first get a bunch of test data
  - Requires agreement on what we're trying to identify
  - How will you measure success? (*Can* you measure success?)
  - Can humans do it?
- Ideally, have different teams get different types of test data
- Then, and only then, should you worry about training data

Lesson #7: Use Modern Data Management Tools

- Before you know it, you'll have thousands of images
- It is easy to lose track of which images were used with which training run
  - Or even what the images are of, or where they're from
- "Hey, we had an accuracy crash on the new model! What happened?"
- There are lots of great tools out there to keep track of your data and experiments.
  - ClearML, Weights and Biases, …
- For your product's success and your own sanity, use them!

Lesson #8: Iterate, Iterate, Iterate

- The sooner you can start your train and test loops, with real data, the better
- Plan on multiple small iterations
- Use these iterations to
  - Understand where your model is strong and weak
  - Understand where you need more training data

Lesson #9: Expose Your Solution to the Real World as Early as Possible

- Ok, so …
  - You went out and collected test data
  - And then training data
  - And then trained your model
- Take your model out into the real world as early as you can
- Get it into the hands of alpha testers
- They will use it in ways you didn't think of
- Early failure will make your product stronger later

Lesson #10: Avoid Bias as Best You Can

- You can create a biased model without being a biased person
- We're not just talking racial bias, we're talking many different kinds of bias
  - E.g., you want to recognize different breeds of dogs
  - But, you've never trained your network on French poodles (or only on a handful of such images)
  - E.g., you've only trained your model with images taken during the day
- Think about your data set -- what's missing? What's overrepresented?
- Tip: more voices and opinions reduce bias
- Tip: avoid groupthink; have people come up with things independently
- Tip: assign a "red team" to poke holes in your data strategy

Lesson #11: Don’t Wait Too Long to Get Embedded

- Typical design flow:
  - Prototype on a desktop or server machine
  - Get your algorithm working
  - Choose an embedded target
  - Port your algorithm to target
- It’s tempting to wait until the last minute to choose your embedded platform and move to it
- Don’t. Nothing gets easier when you move to embedded, and you’ll find new problems
- Don’t wait to discover those problems until the last minute.

Parting Thoughts

A quote from one of my favorite authors (about random number generators), repurposed to be about managing embedded vision projects:

> It's harder than it looks
> It's not for the unwary
> It can be done if you keep your wits about you
>
> -- Neal Stephenson, *Cryptonomicon*

Hopefully these (identified) lessons have been somewhat helpful!