Five Things You Might Overlook on Your Next Vision-Enabled Product Design

Phil Lapsley

Co-Founder and Vice President

BDTI

About BDTI

For more than 30 years, BDTI has been a trusted source for engineering, analysis, and advice on embedded processing.

For the last 10 years we've focused on computer vision, deep learning, and embedded AI.

Our specialities:

- Algorithm design and implementation
- Processor selection
- Development tool and processor evaluation
- Training and coaching on embedded AI technology

You're Designing Your New Vision Product! What Questions Come to Mind?

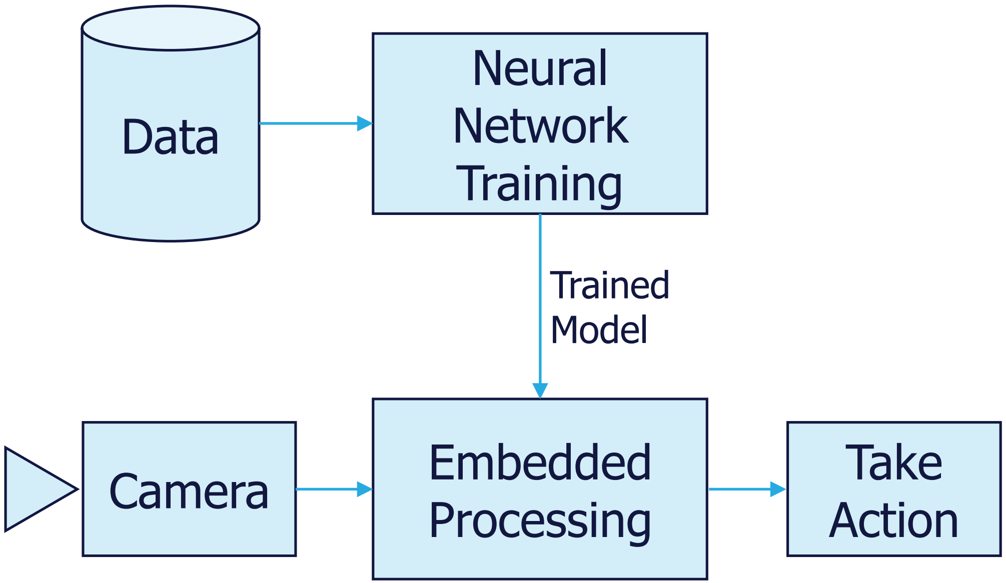
- Which processor am I going to use?
- What neural network will it run?
- Where will I get the training data?

These are important questions! But they're not the ones I'm going to talk about. ☺

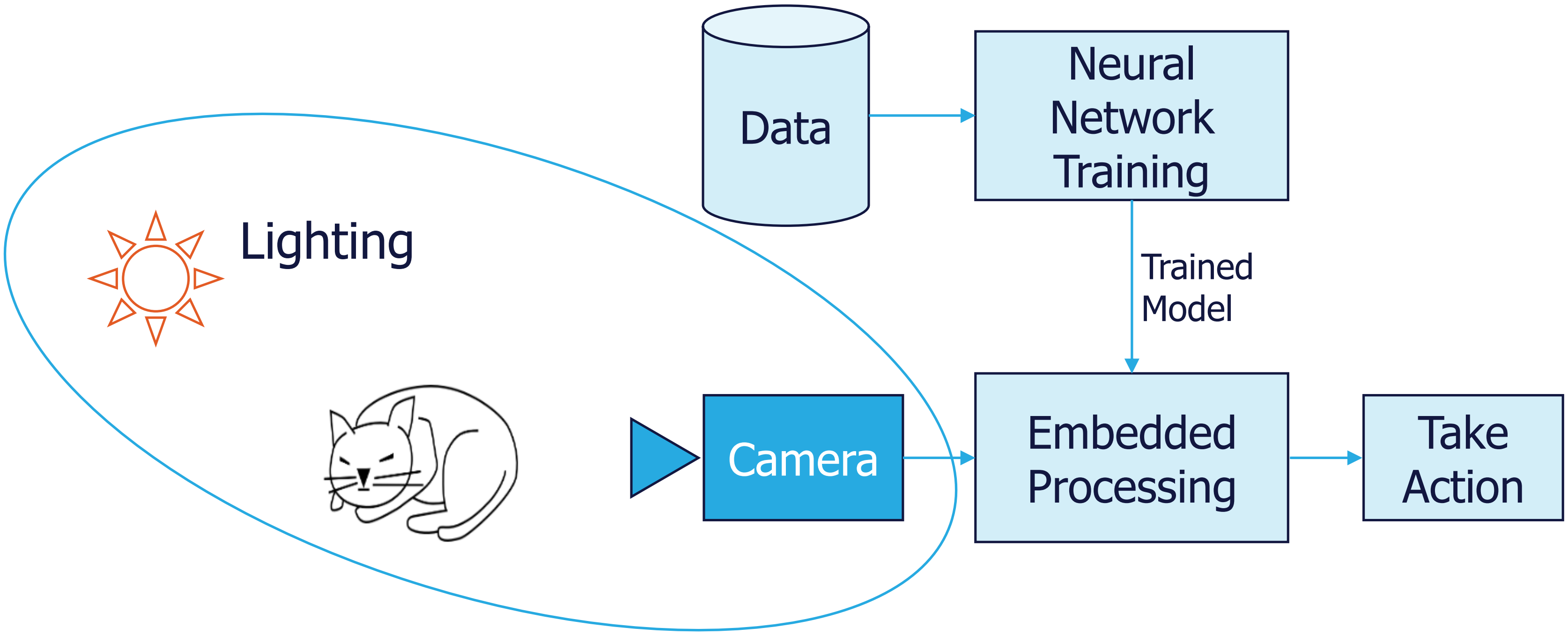
Things We Find Are Often Overlooked

- Lighting and camera placement
- ML model and training experiments
- When to leverage existing frameworks, vs. rolling your own?
- Vendor risks
- Dealing with neural network memory constraints

Typical Embedded Vision System



It All Starts With Camera and Lighting



Glare ("Hey, Where'd My QR Code Go?")

Soft, diffuse lighting is critical to avoid glare

IR Washout ("Hey, Where's My Detergent?")

Infrared cameras and lighting allows seeing defects that visible light cameras can't.

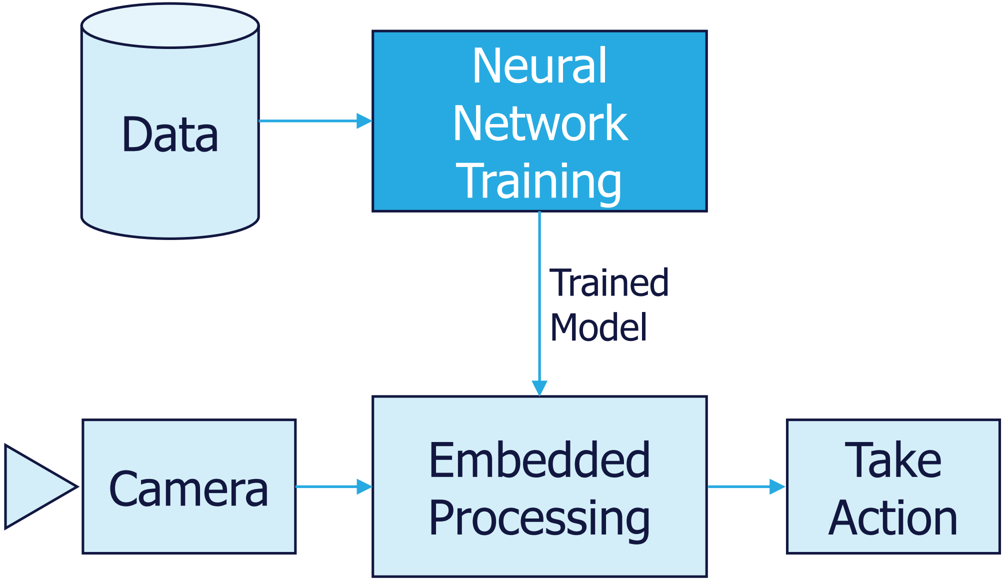
But IR is tricky!

Again, soft, diffuse lighting is key

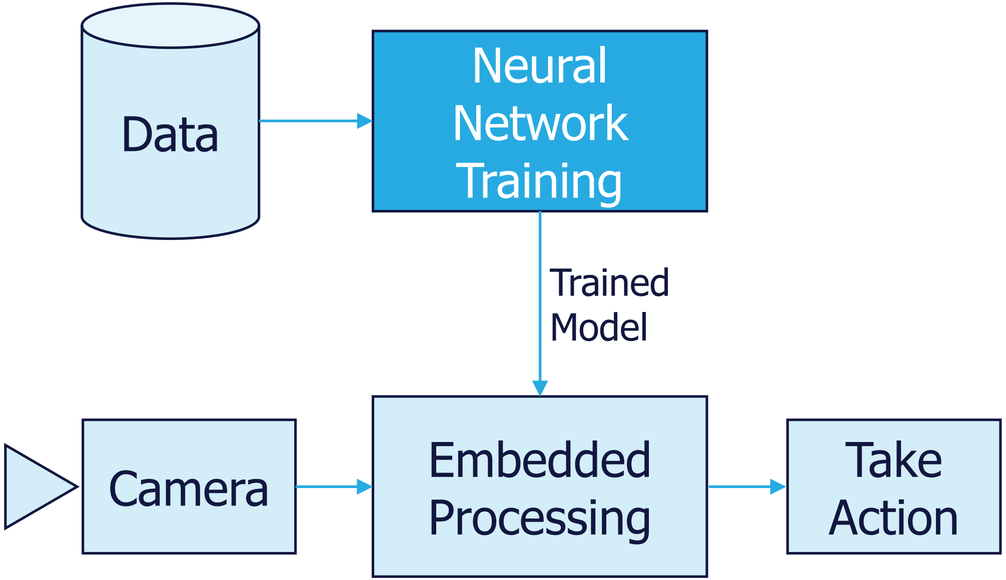
Other Camera and Lighting Issues

- SPIDERS!
  - Er: “Outdoor environ-mental challenges”
- Outdoor lighting and variability
- LED flicker
- Lens distortion
- Camera positioning
- …

Model Training



Model Training



Source: https://xkcd.com/1838

What Does "Looking Right" Mean?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.

embedded VISION SUMMIT

Possible Metrics

- Accuracy?
- Precision?
- Recall?
- False positives?
- False negatives?
- mAP?
- F1?
- … ?

Possible Measurement Conditions

- Input image resolution?
- Quantization?
- Confidence level?
  - (Per class? Global?)
- Intersection over union (IoU)?
- … ?

BDTi

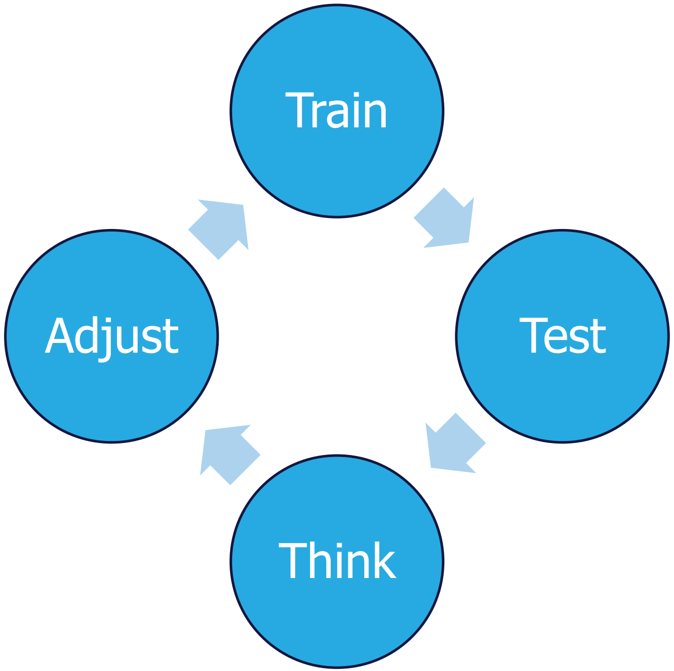
"Stirring the Pile" – Iterative Model Training

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.

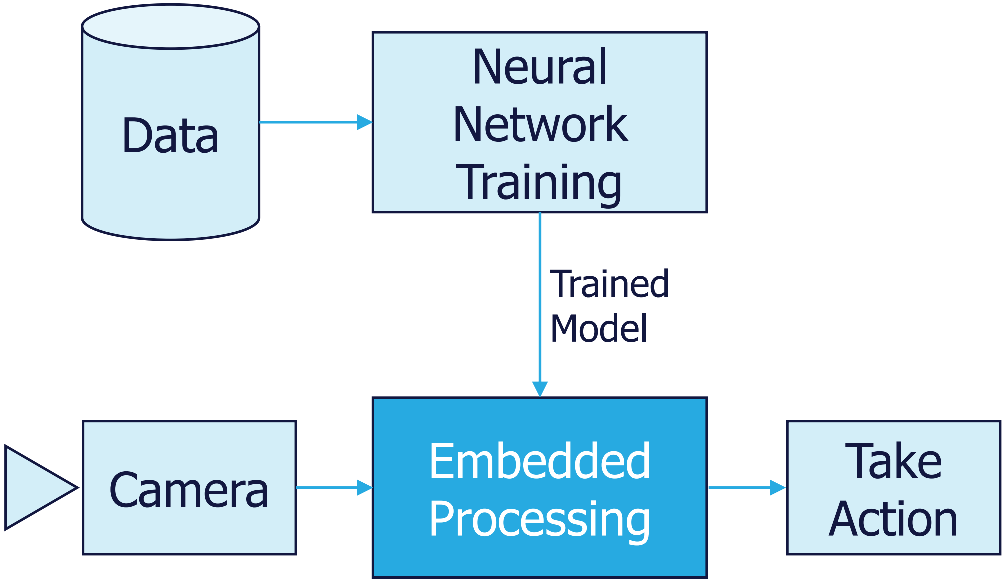
Start small with a known-good data set

- Train
- Test
- Analyze where your model is failing
- Adjust in response and train again
- Repeat until metric flattens out

Then consider your next step



Embedded Processing



What *Shouldn't* You Do?

Would you …

Write your own operating system?

Develop your own networking stack?

Design your own power supply?

Build your own neural network training framework?

Create your own computer vision library?

Architect your own neural network from scratch?

There's So Much Out There!

Some examples:

- TensorFlow Lite Micro (TFLM) for vision on microcontrollers
- PyTorch Mobile or CoreML frameworks for vision on mobile devices
- Nvidia DeepStream, TensorRT, many other Nvidia packages for those platforms (other vendors have similar things)
- OpenCV

Leverage what your vendor gives you, and all the smart people out there giving you open source.

Still, Buyer Beware…

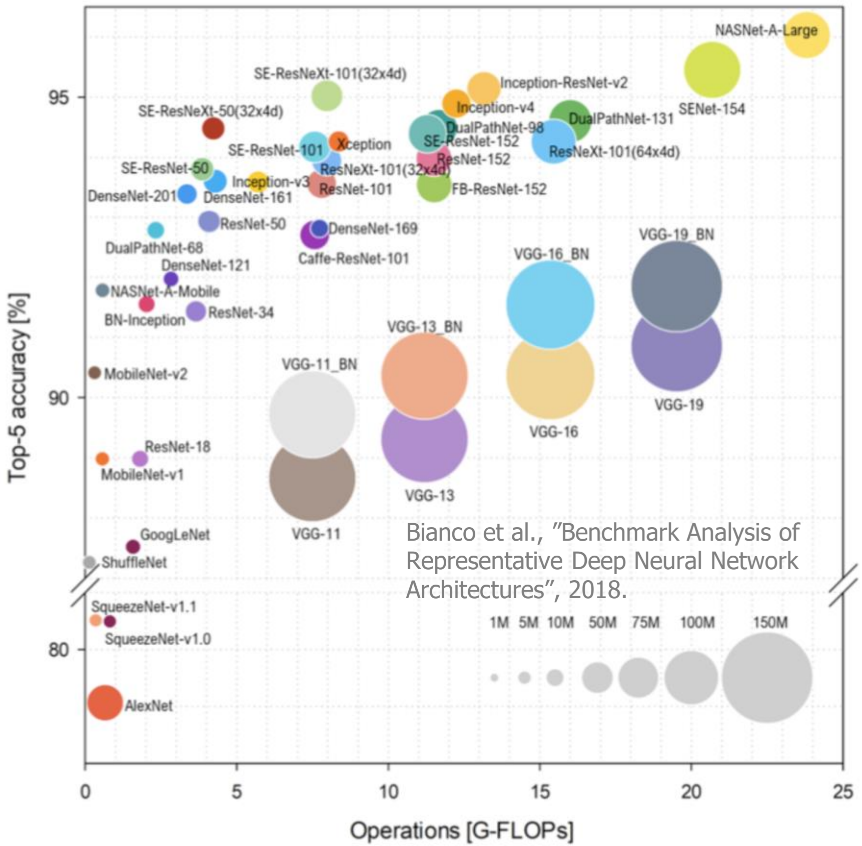
Some common places where you may run into issues with your embedded processor:

- Operating system support
- Drivers (wifi, GPU, …)
- Processor has an ISP but you can't use it, or it only works with particular image sensors
- Camera tuning
- NPU doesn't support certain operations
- NPU doesn't have enough memory to run your network
- No one has ported your network to their NPU

Memory Constraints

Modern neural networks have millions to billions of parameters.

What does this mean for memory usage on your embedded vision system?



Bianco et al., "Benchmark Analysis of Representative Deep Neural Network Architectures", 2018.

Parameters Drive Permanent Storage (Flash)

The number of parameters in a neural network determines how much *permanent storage* you need for the trained network.

Typically this is flash memory.

- E.g., 5 million parameters at 8 or 16 bits (half-float) = 5 or 10 Mbytes of storage

Largest Feature Map Drives RAM Usage

Neural networks are made up of layers

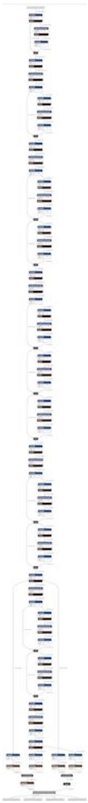
Most layers run sequentially

In general, the maximum amount of RAM needed is given by the largest feature map in the layers

(All other layers need less than this maximum)

You can get this information from the output of your neural network compiler

Example: MobileNet v2 Alpha 1.0



```
 --------------------  Base model summary  ----------------------
Model: "mobilenetv2_1.00_224"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
Conv1_pad (ZeroPadding2D)       (None, 225, 225, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
Conv1 (Conv2D)                  (None, 112, 112, 32) 864         Conv1_pad[0][0]
__________________________________________________________________________________________________
bn_Conv1 (BatchNormalization)   (None, 112, 112, 32) 128         Conv1[0][0]
__________________________________________________________________________________________________
Conv1_relu (ReLU)               (None, 112, 112, 32) 0           bn_Conv1[0][0]
__________________________________________________________________________________________________
expanded_conv_depthwise (Depthw (None, 112, 112, 32) 288         Conv1_relu[0][0]
__________________________________________________________________________________________________
expanded_conv_depthwise_BN (Bat (None, 112, 112, 32) 128         expanded_conv_depthwise[0][0]
__________________________________________________________________________________________________
expanded_conv_depthwise_relu (R (None, 112, 112, 32) 0           expanded_conv_depthwise_BN[0][0]
__________________________________________________________________________________________________
expanded_conv_project (Conv2D)  (None, 112, 112, 16) 512         expanded_conv_depthwise_relu[0][0
__________________________________________________________________________________________________
expanded_conv_project_BN (Batch (None, 112, 112, 16) 64          expanded_conv_project[0][0]
__________________________________________________________________________________________________
block_1_expand (Conv2D)         (None, 112, 112, 96) 1536        expanded_conv_project_BN[0][0]
__________________________________________________________________________________________________
block_1_expand_BN (BatchNormali (None, 112, 112, 96) 384         block_1_expand[0][0]
__________________________________________________________________________________________________
block_1_expand_relu (ReLU)      (None, 112, 112, 96) 0           block_1_expand_BN[0][0]
__________________________________________________________________________________________________
block_1_pad (ZeroPadding2D)     (None, 113, 113, 96) 0           block_1_expand_relu[0][0]
```

112 x 122 x 96 channels = 1.2 Mbytes at largest feature map.

(You may need to double-buffer this.)

Resources

- Edge AI and Vision Alliance website: https://edge-ai-vision.com
  (Lots of great articles and tutorials!)

- Stop by BDTI at Booth #417 to talk to our engineers
  (You've got a little over an hour before the show closes!)

- Email us at info@bdti.com