embedded VISION SUMMIT 2018

Harnessing the Edge and the Cloud Together for Visual AI

Au-Zone Technologies Inc.

Sébastien Taylor

May 23, 2018

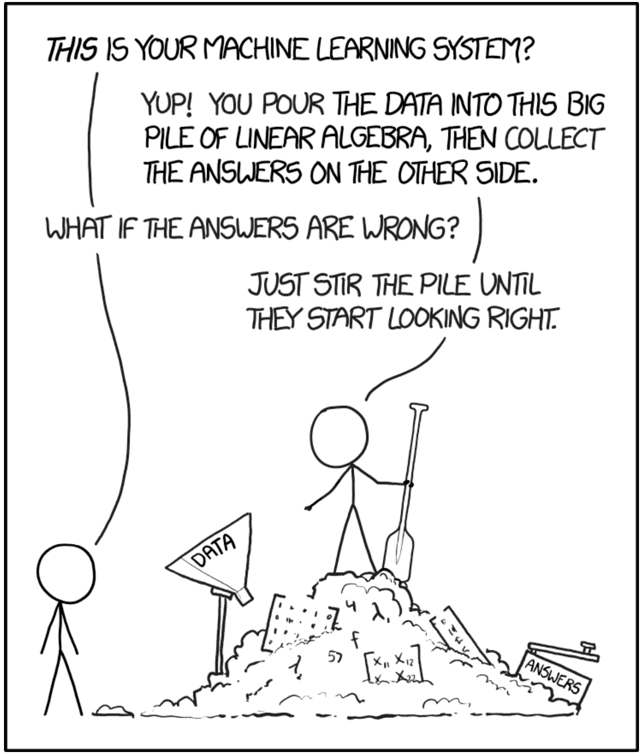
About Au-Zone

- Au-Zone Technologies is a leading provider of development tools, engineering design services, and enabling IP used for the design of intelligent embedded vision products and solutions.
- By utilizing our **Machine Learning** and **embedded Computer Vision** tools we enable our customers to quickly develop and securely deploy machine learning solutions and novel Convolutional Neural Networks on embedded hardware.

Introduction

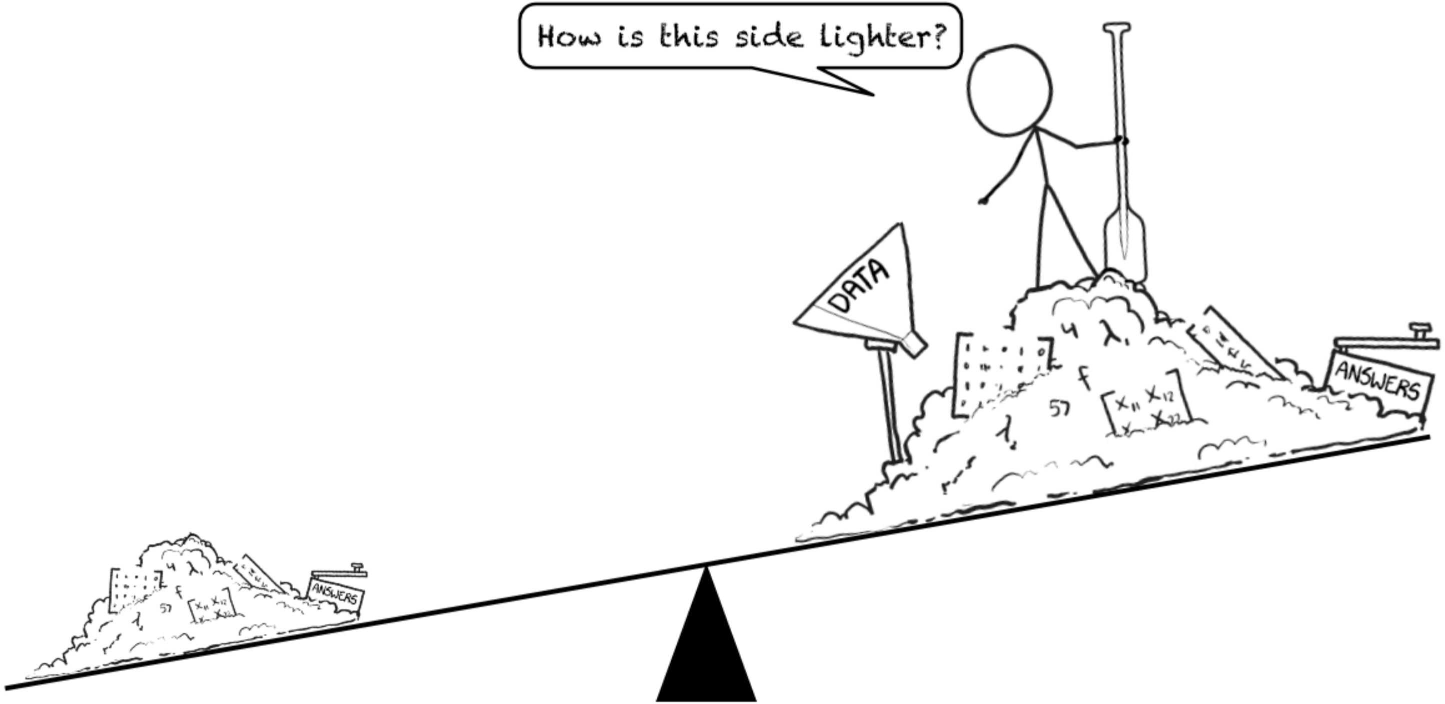
- Focus on Image Classification using Deep Neural Networks
- Building a Hybrid Solution
- Problems to solve to make this work
    - Modeling the unknown
    - Distributing models to the edge efficiently
- Example
    - Face Recognition

Advanced Model Design


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

www.xkcd.org

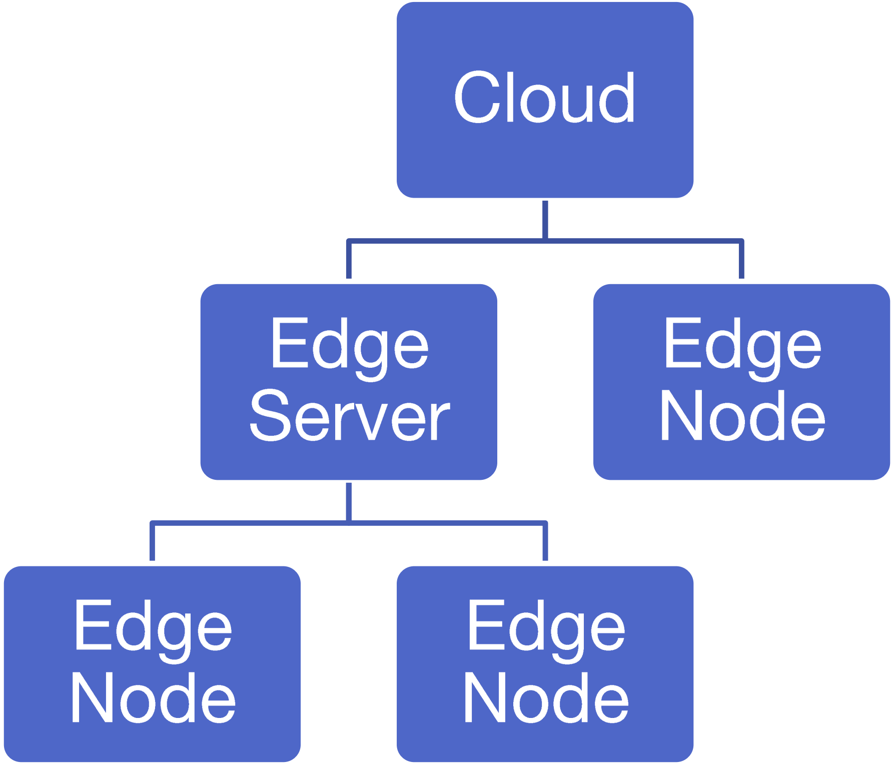
Advanced Model Balancing



Architecture

Hybrid Edge-Cloud Architecture

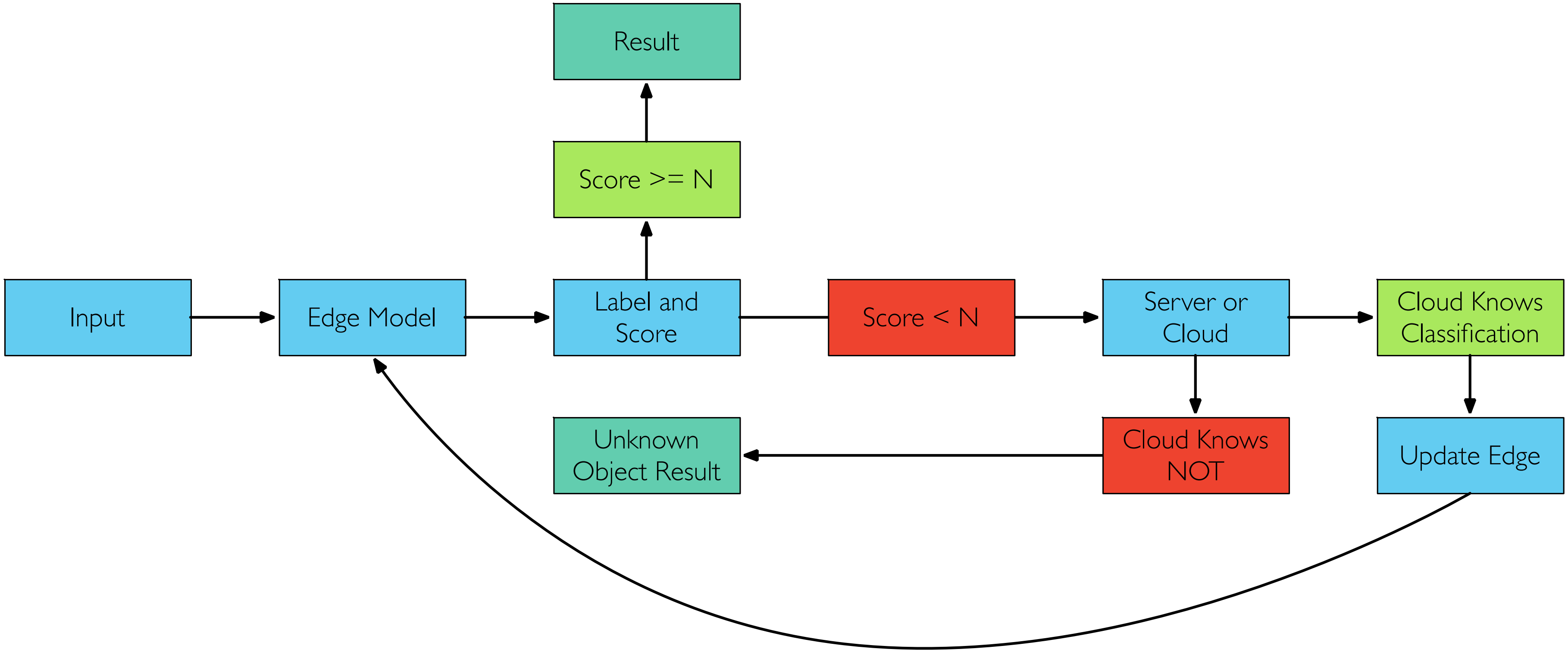
- Typical Cloud Server
  - Multiple, large models
  - Central point
- Optional Edge Server
  - Intermediate between cloud and edge
  - Caching, computational offloading
  - Can handle training, dataset evolution
- Peer Nodes
  - Idle or more powerful
- Edge Nodes
  - Small models
  - Solution focused



Target Edge Devices

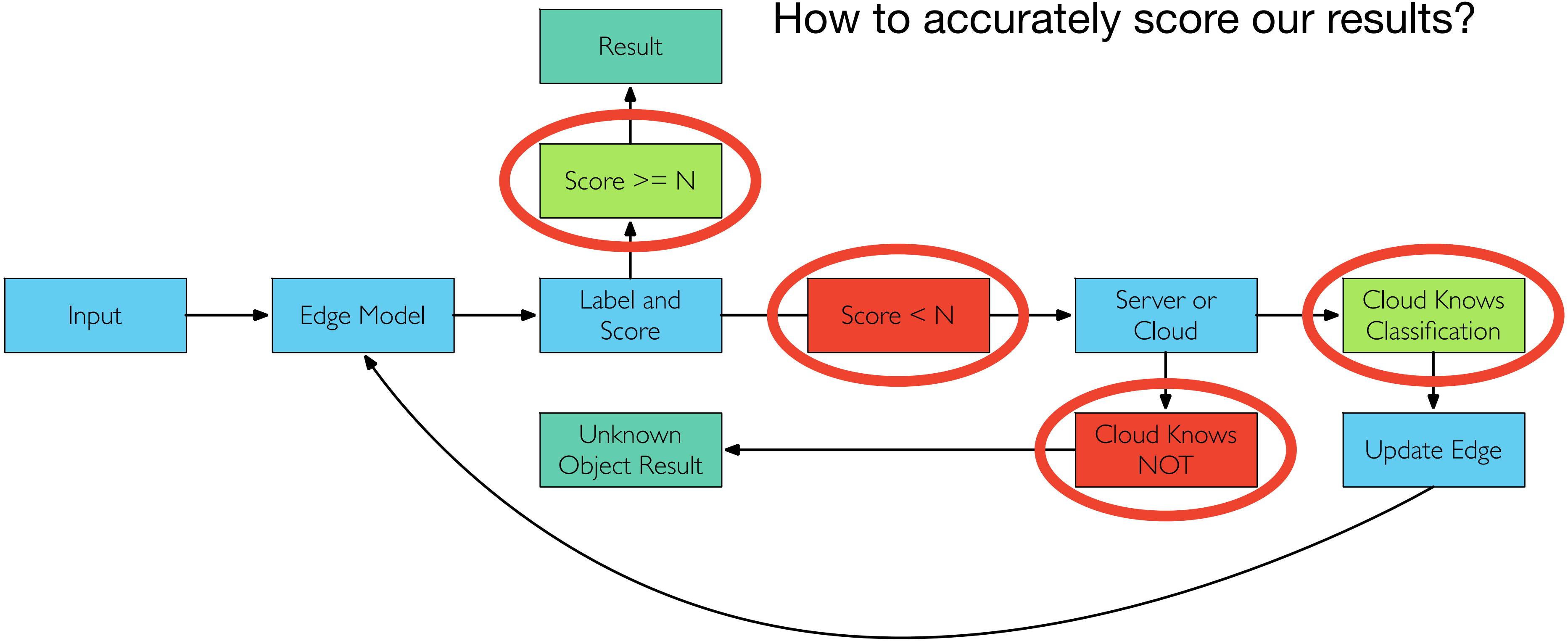
- Our examples cover devices as small as Cortex-M4 and Cortex-M7
    - Sub-150 mW devices (CPU under $3)
    - Bare metal/RTOS
    - Hundreds of KB of RAM
- Scaling up to Cortex-A and beyond
    - Examples on Cortex-A9 and Cortex-A53 (CPU under $30)
    - Linux
    - Sub-2500 mW devices
    - Hundreds of MB of RAM

Architecture Flow

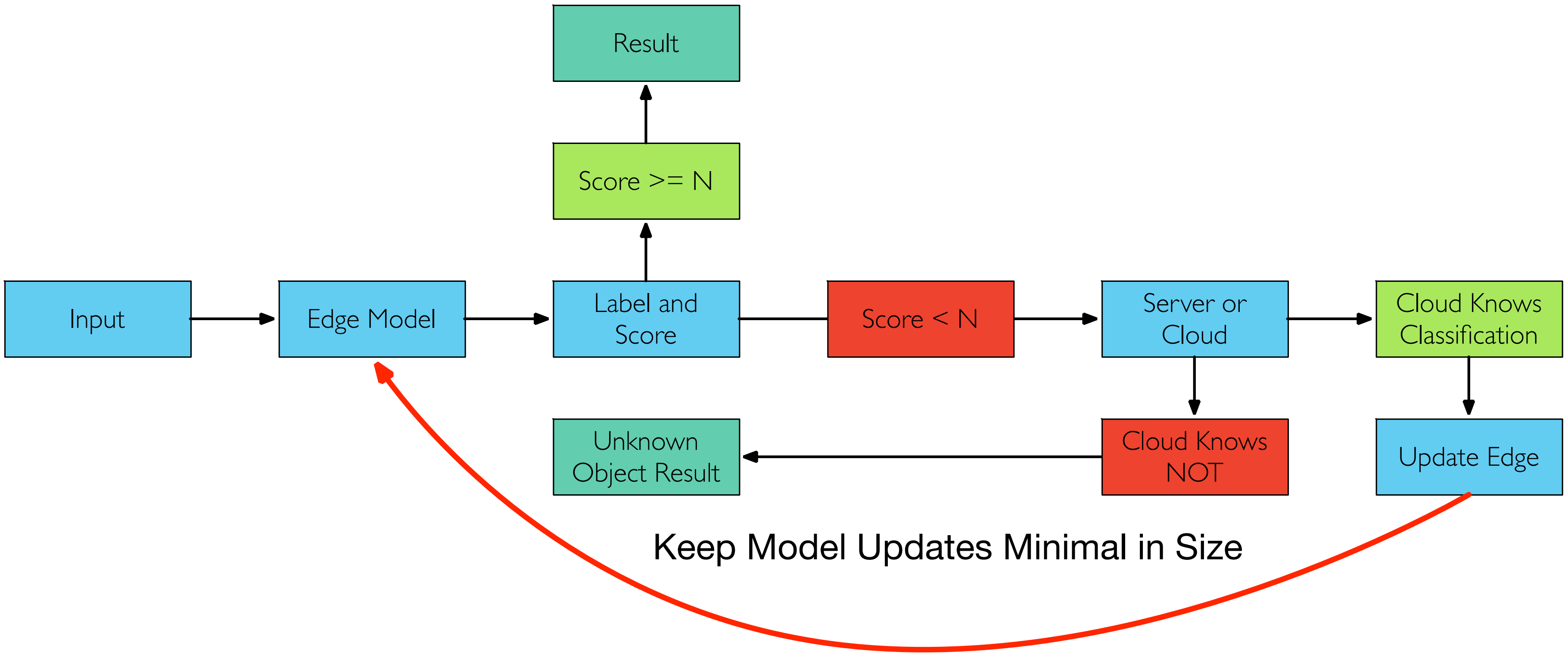


Architecture Flow – Result Scoring

How to accurately score our results?



Architecture Flow – Model Update Size

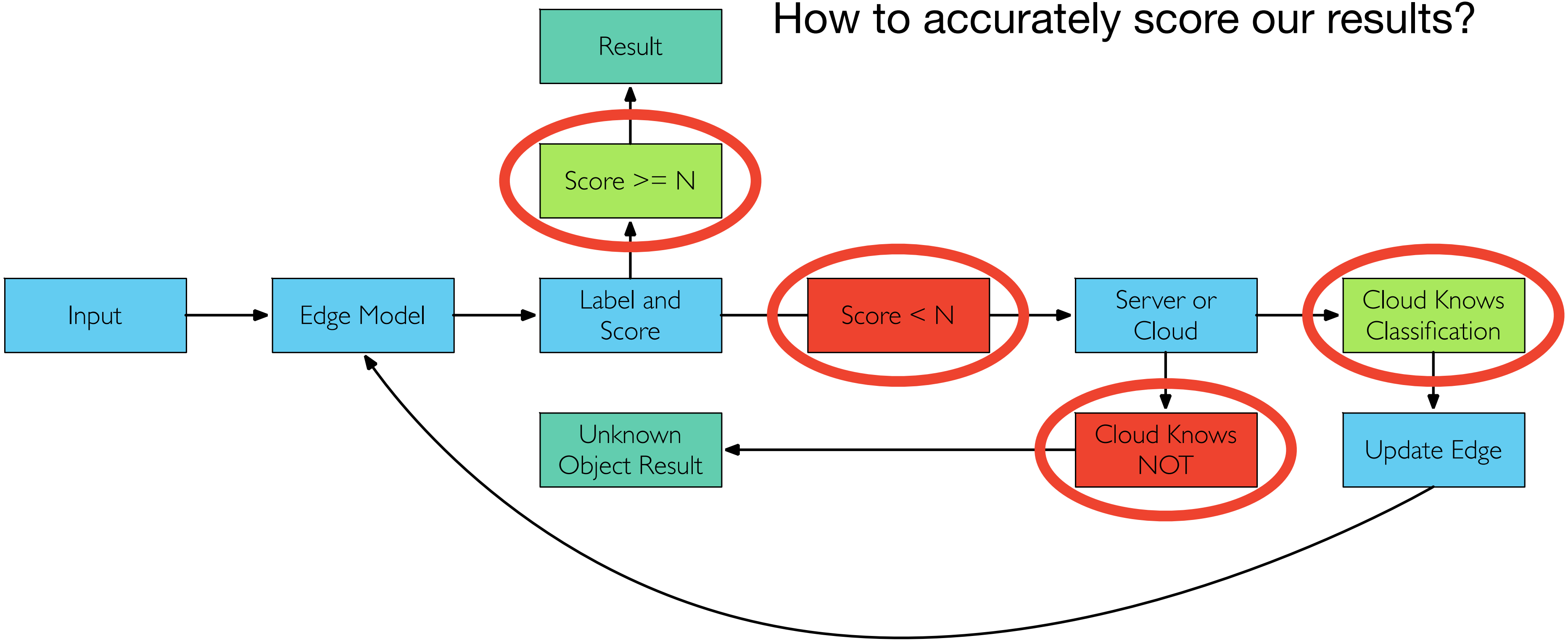


Keep Model Updates Minimal in Size

How to know what you do not know?

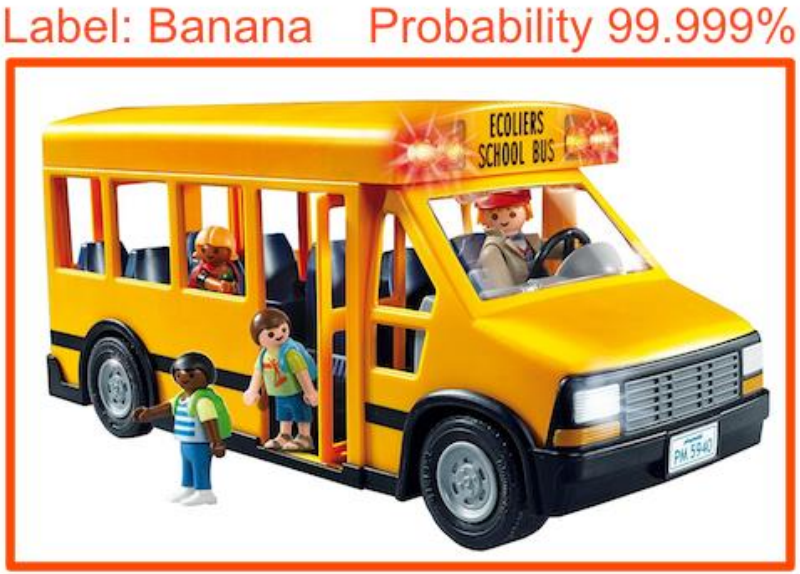
Architecture Flow – Result Scoring

How to accurately score our results?



How to know you do not know?

- We need to know WHEN to go to the cloud for an update.
- Models tend to be overconfident in their results.
- Softmax is relative to KNOWN labels.
- Most objects probably UNKNOWN.
- model of all unknown labels…



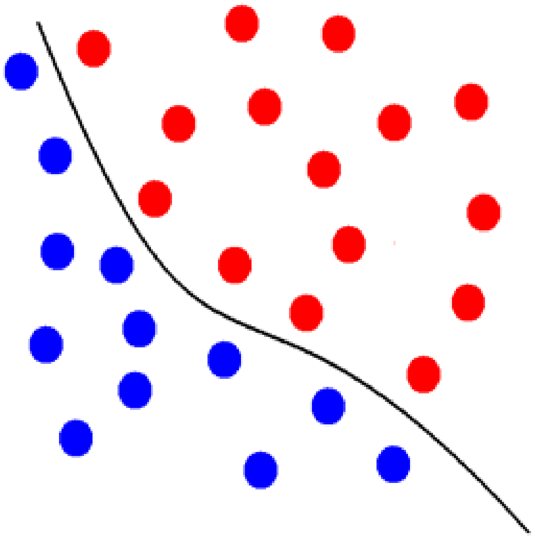
Other Solutions?

- Model the Universe…

- …probably not practical on an embedded device yet.

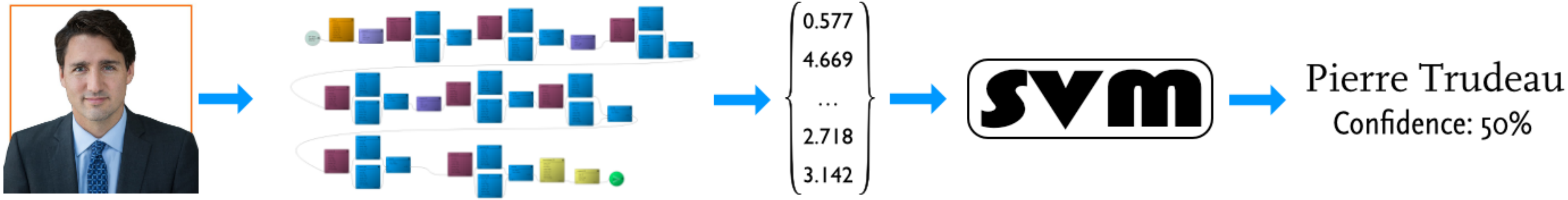
Solutions – SVM Classifier

- Use the neural network as a feature extractor
- Reduces an image into a small vector
- The CNN output becomes the SVM input
- Measures how well the features fit “probability”



SVM Face Detection

- CNN can be used to extract features from an image
  - Trained to generate discriminating features
- SVM uses these features as inputs
  - Trained to fit a label and a **probability** from the input features
  - The probability is reliable and accurately reports unknown samples



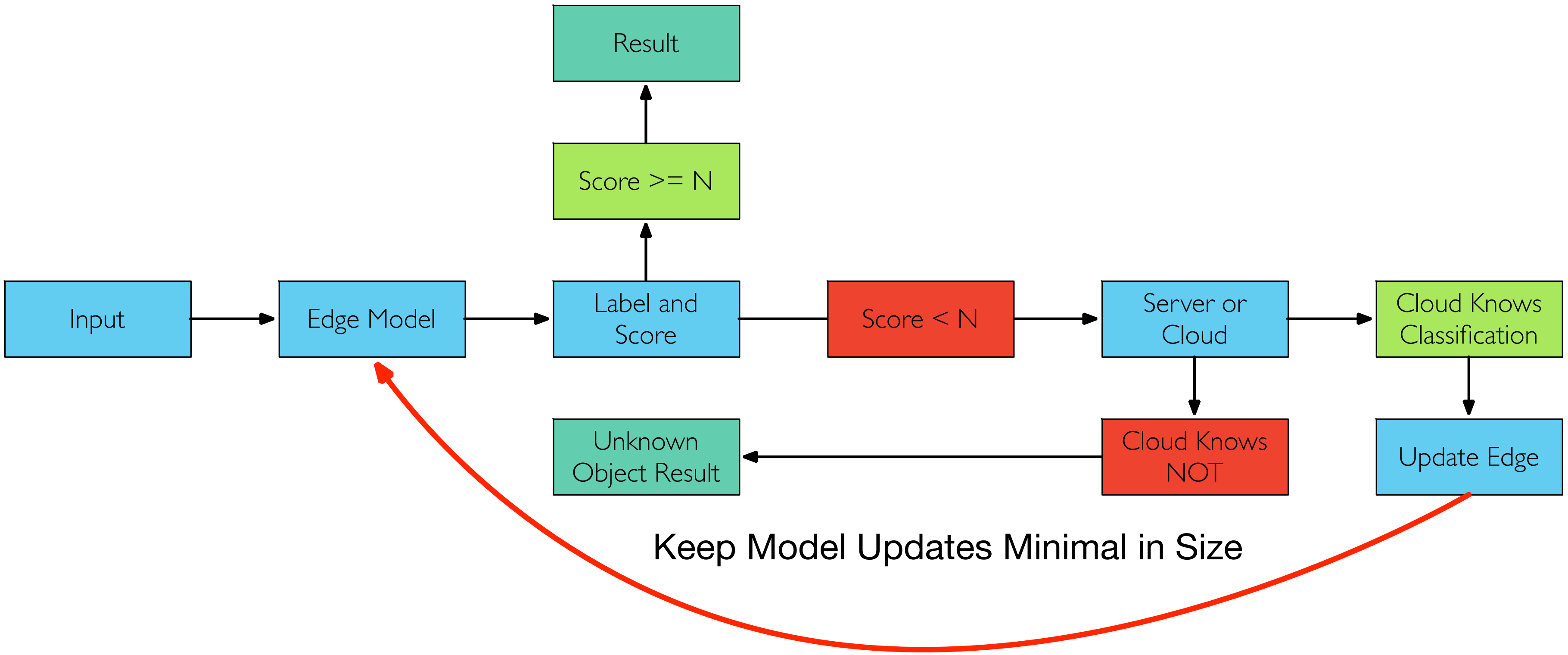
$$\begin{pmatrix} 0.577 \\ 4.669 \\ \cdots \\ 2.718 \\ 3.142 \end{pmatrix}$$

Pierre Trudeau

Confidence: 50%

Distributing Model Updates

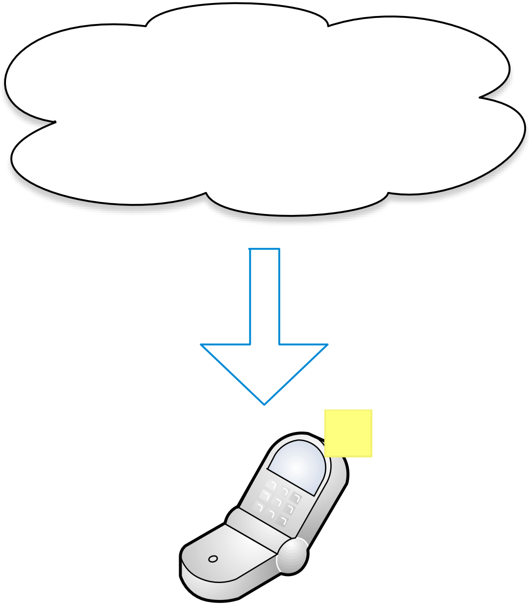
Architecture Flow – Model Update Size



Keep Model Updates Minimal in Size

Updating the Edge

- IoT data transmission is still expensive
- Must Keep usage to a minimum

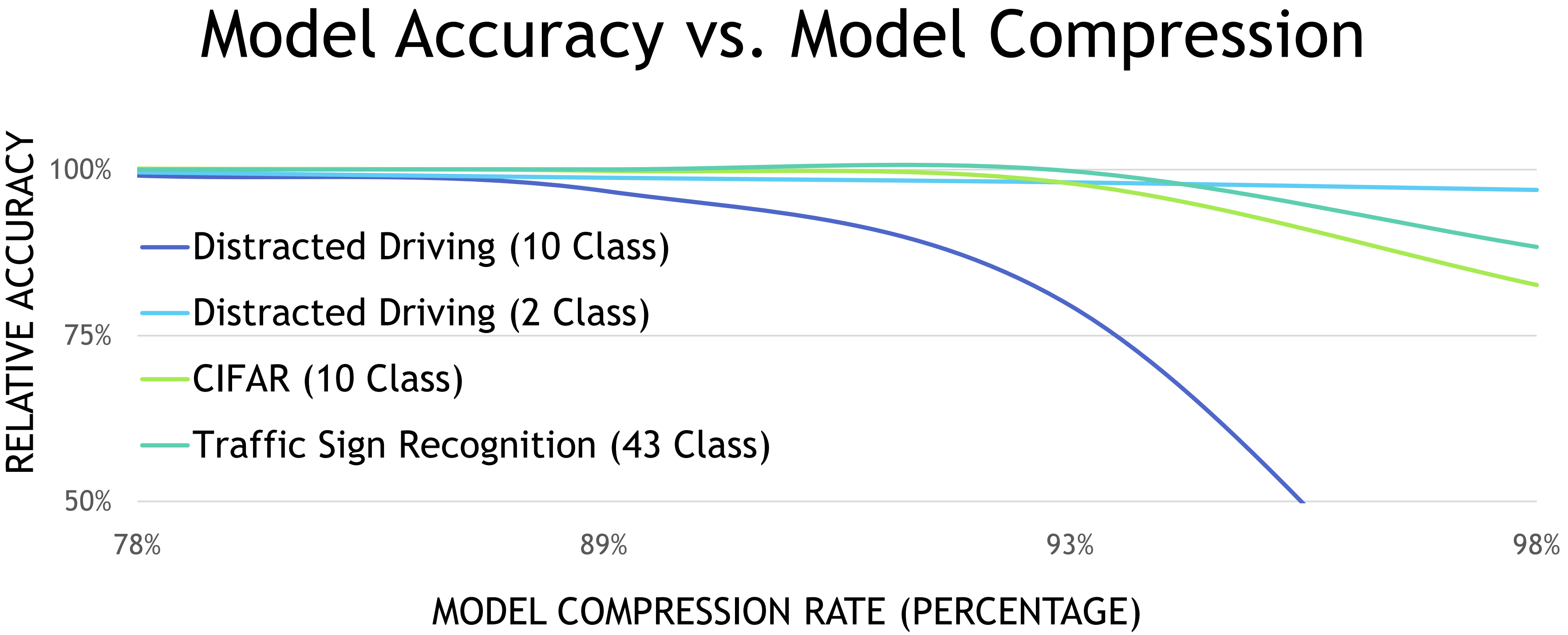


- Compress Models
  - Specifically compress the weights
- Send differences
  - Keep differences to a minimum

Limits of Compression

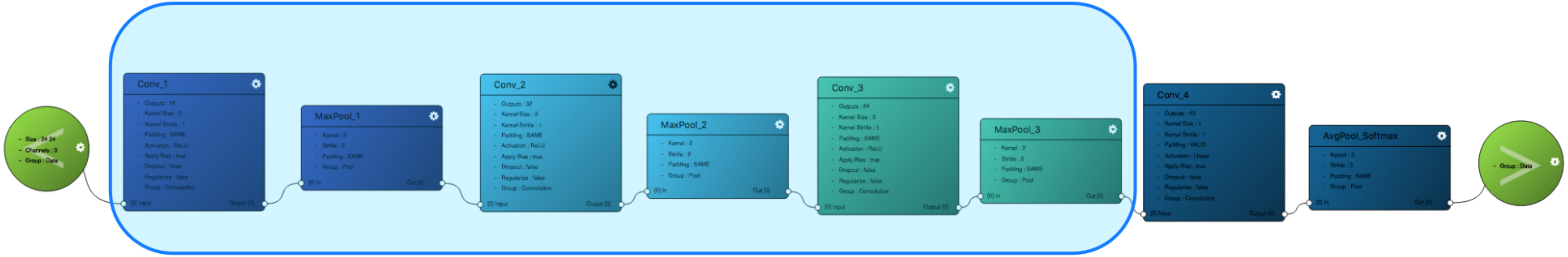
- Models generally do not compress well
  - Random data cannot be compressed
  - ”Smooth” data can be compressed
  - Are model weights more random or are they smooth?
- Lossy compression can greatly help
  - Neural networks are very resilient to error from lossy compression
  - Some models work well even down to 2 bits per weight!

Lossy Compression



Transmit Differences

- Need to partially freeze model to avoid updating ALL weights
  - Cannot efficiently send differences if everything is changing
- If the front end is well generalized we only need to train the tail end
  - Same idea as transfer learning, in this case to help reduce data exchange
- In the SVM example we only need to update SVM weights, not the CNN model



Frozen Model 23,472 Weights

Transfer 2816 Weights (10%)

Summary

Summary

- Cloud can use larger, evolving models
  - Can be used to train more focused models for the edge
  - Allows us to keep smaller models at the edge
- Need to know when to ask the cloud for help
  - Accurately detect when a sample is unknown
  - Go to the cloud for verification when unknown
  - Get updated models if available
- Need to efficiently distribute model updates
  - Lossy compression
  - Partial model updates

Resources

- www.au-zone.com
- www.embeddedml.com
- RT1050 https://www.youtube.com/watch?v=B2zwx6BYsKg
- i.MX8 Model Transfer https://www.youtube.com/watch?v=z0WtwXSIA9M
- DeepView MLTK https://www.youtube.com/watch?v=IS0QgM1VHaY
- Model Compression https://www.embedded-vision.com/platinum-members/embedded-vision-alliance/embedded-vision-training/documents/pages/deep-learning-software
- CNN Calibration https://arxiv.org/abs/1706.04599
- Modelling Uncertainty https://arxiv.org/abs/1509.05909