Overcoming the Electronic Traffic Sign Problem

MOBILEYE
An Intel Company

Introduction

- I: Avi Hendler-Bloom.
- Our Team: Driving Semantics, Algorithms Department.
- Team Goal: Recognize and classify static objects (traffic signs, traffic lights, arrows on the road, tunnels, etc.).

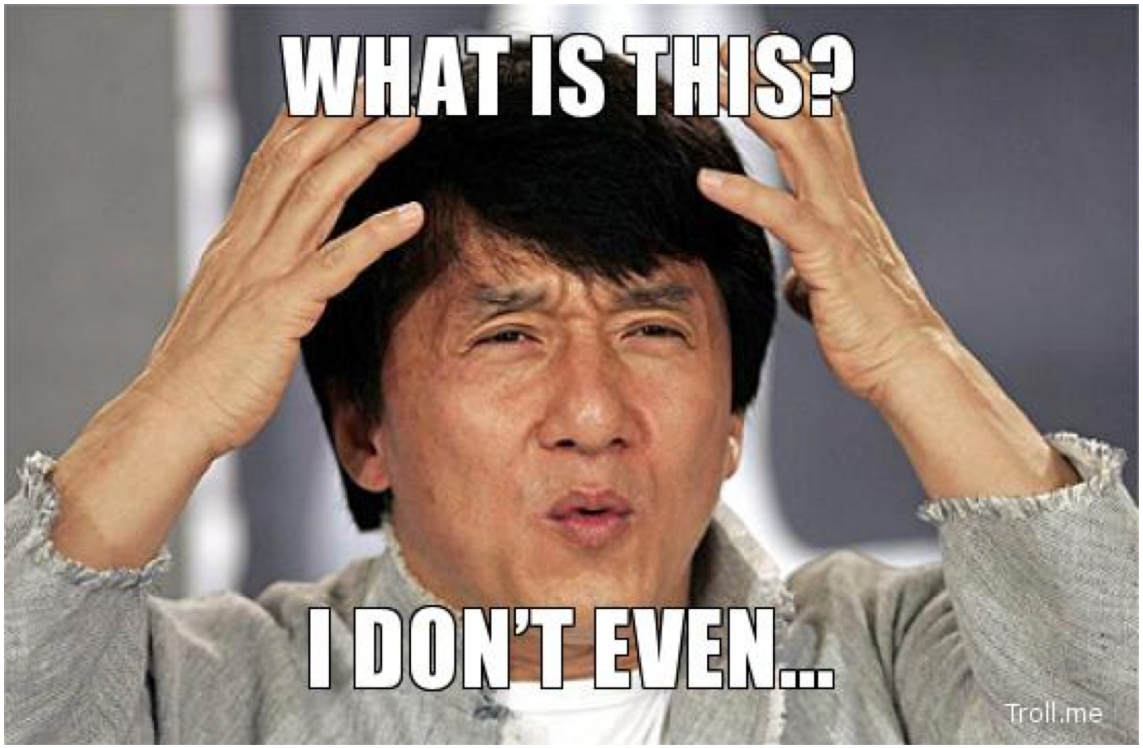


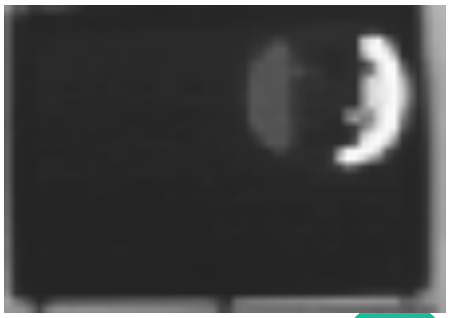
Electronic Traffic Signs

- LED is evil!
- Differences in frequency and phase may cause this:

Electronic Traffic Signs

Electronic Traffic Signs





But when looking at the video…

More examples

3 Types Of Electronic Signs

- Easy (looks like regular signs)

- “Scan” (e.g. the first video)

- “Random” (very rare and very difficult)

Previous Mobileye Solution

- Define “single image clear sign”: A single image clear sign is that which an average person can understand by seeing it.
- Ground truth rectangles should be marked only if they are “single image clear signs”.
- Train a neural network.
- High level will decide the correct type.

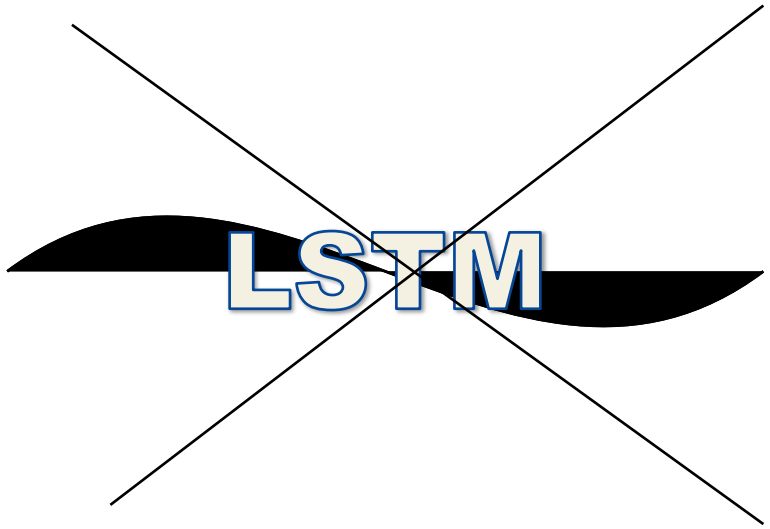
Problems

- Sometimes you get only 2-3 frames of “clear signs” - if you miss them – you miss the sign.
- False classification (80 and 60 for example).
- Waste of data.

Ideas?

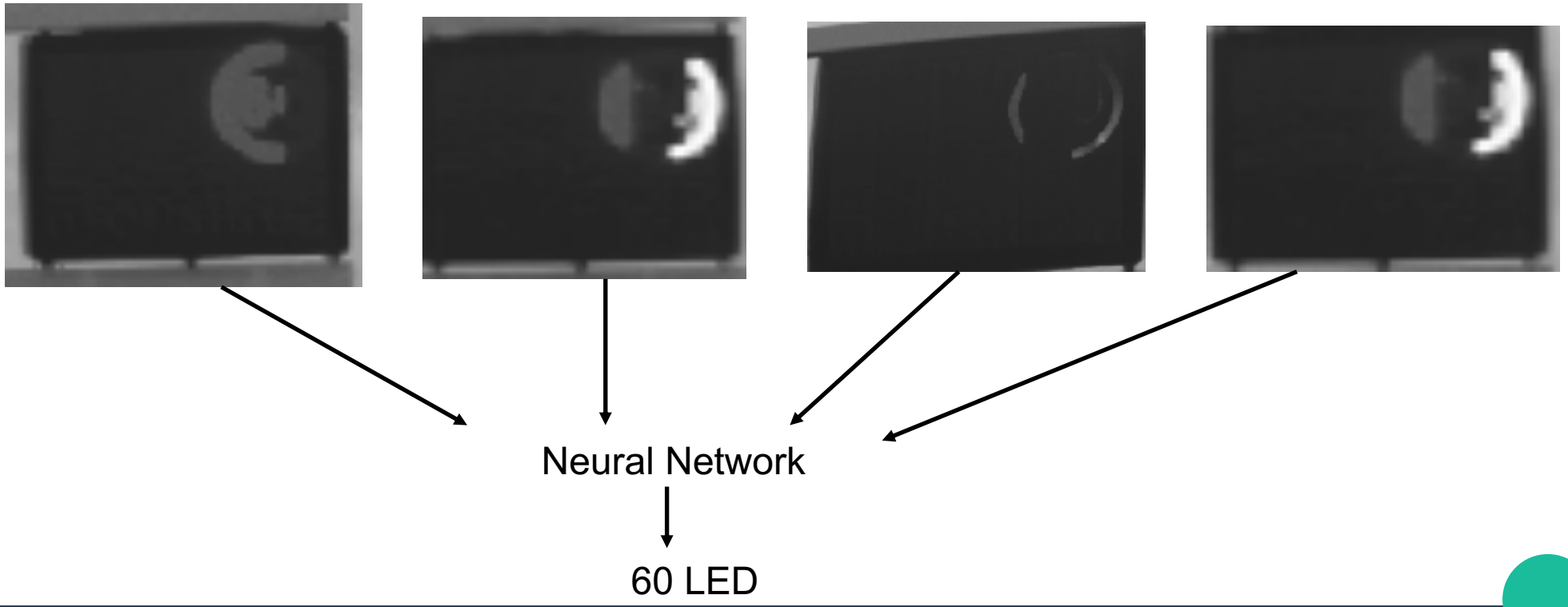
Solution Offers(1)

- Solution1: lstm
- Problems: Expensive, hard to train.



Solutions Offers(2)

- Solution 2: Take 4 images (the limit in existing memory) as input (input size 48 * 48 * 4)
- Problems: Only 4 images, expensive.

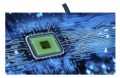


Solutions Offers(3)

- **Solution 3: Take N > 4 patches of images, save them in memory and give them as input (input size 48 * 48 * N).**
- **This actually works well!**

- **Problems:**
  - Expensive in space:
    - (48 * 48 * N * numCandidates).
  - Expensive in time.

Can We Do Better?

- Goal: great performance.
- Limitations:
  - Memory ()
  - Time (0) 0.11s
  - We're Here

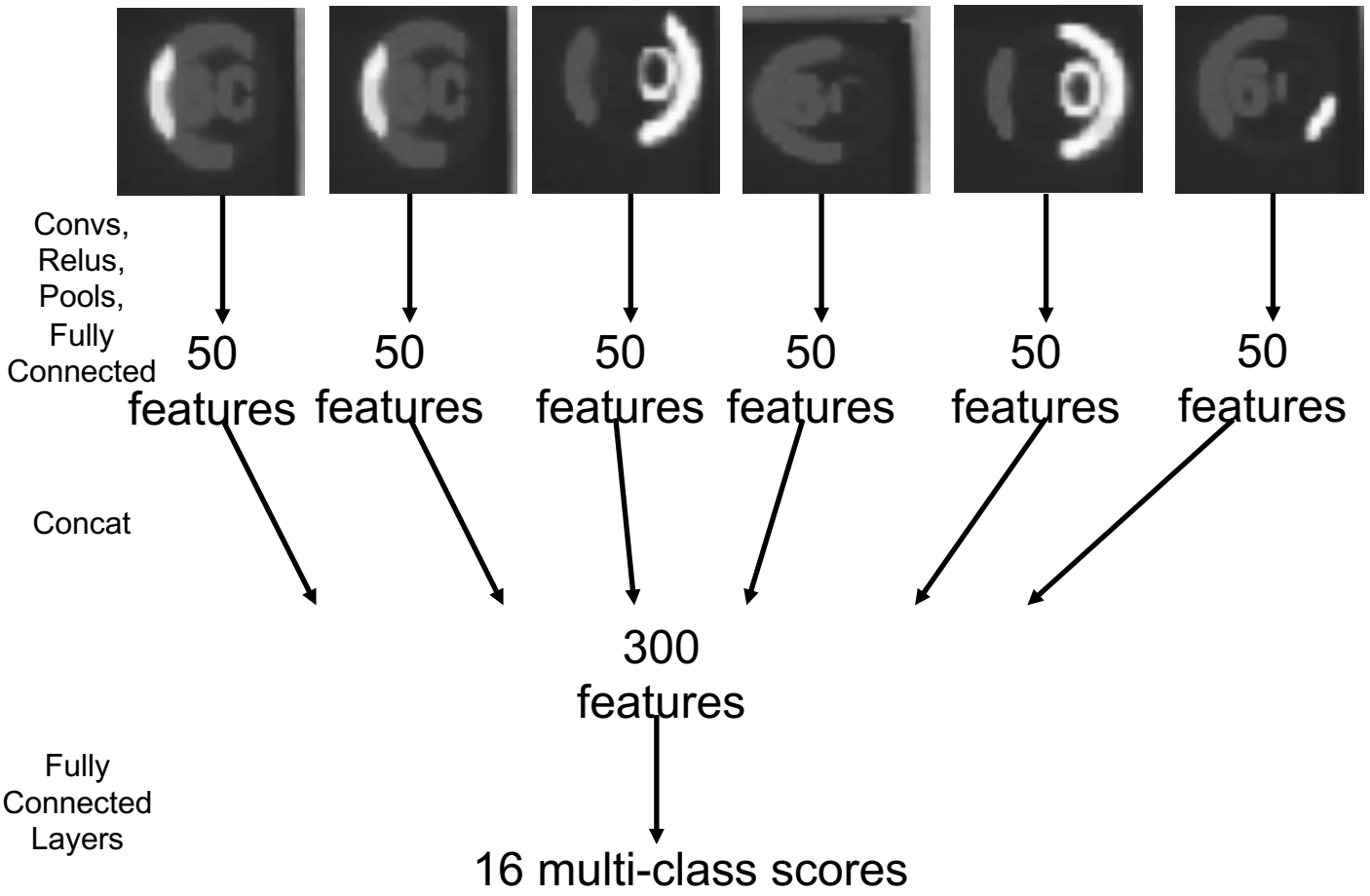
- Ideas?

Our Final Solution

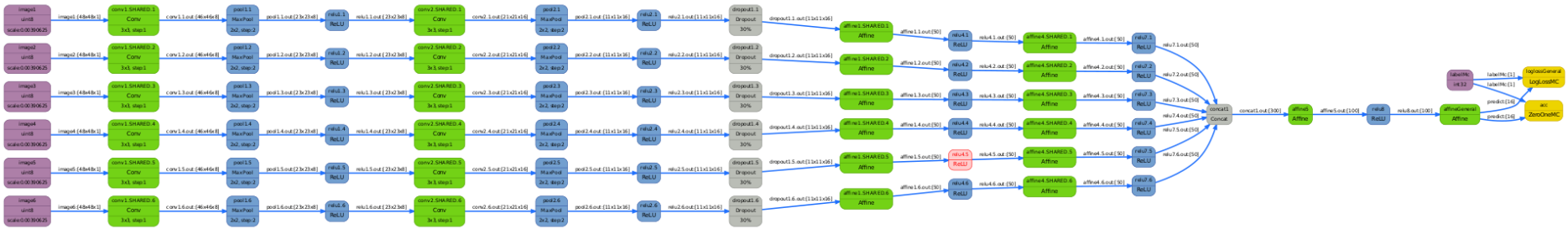
- **Idea:**
  - What if we could reduce the dimension of every image? e.g. (48 * 48 → 50).
  - We would solve the time + space problem!
- **We want to reduce the dimension using a neural network!**

Our Final Solution

- Implementation:



NN Graph



Extra Problems

- **Overfit:**
  - Some image classes are very rare (e.g. 30 electronic)
  - “Easy” electronic signs are common, “Hard” electronic signs are ~10% of the data.
  - What helped?
    - Dropout before 1-2 layers(helped a bit)
    - Dropout before every layer(helped a lot)
    - Noise after image input(helped a lot)
    - Dropout + Noise(helped the most)

Extra Problems

- **False Positive:**
  - What helped?
    - Cascade – only if the sum of the electronic scores > 0.1 of the previous classifier, then we continue to our LED classifier.

Results Videos

Questions?