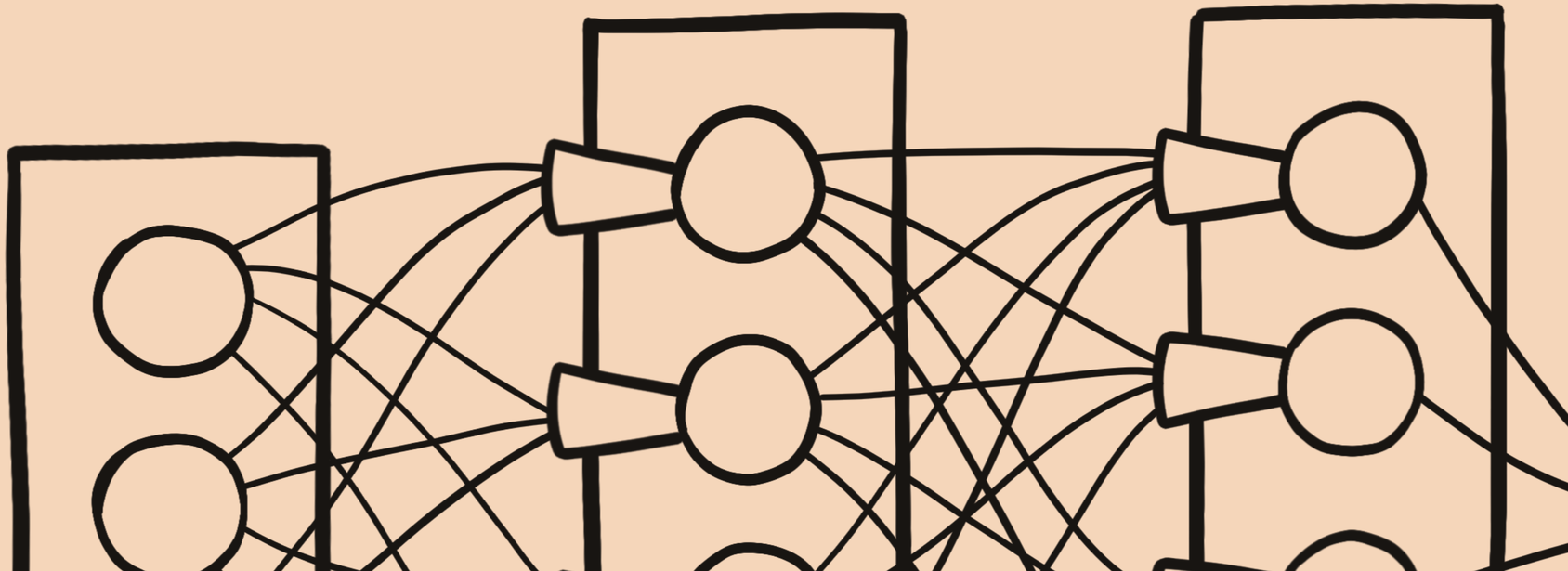
Can Neural Networks Make Me a Better Parent?

A tale in three acts



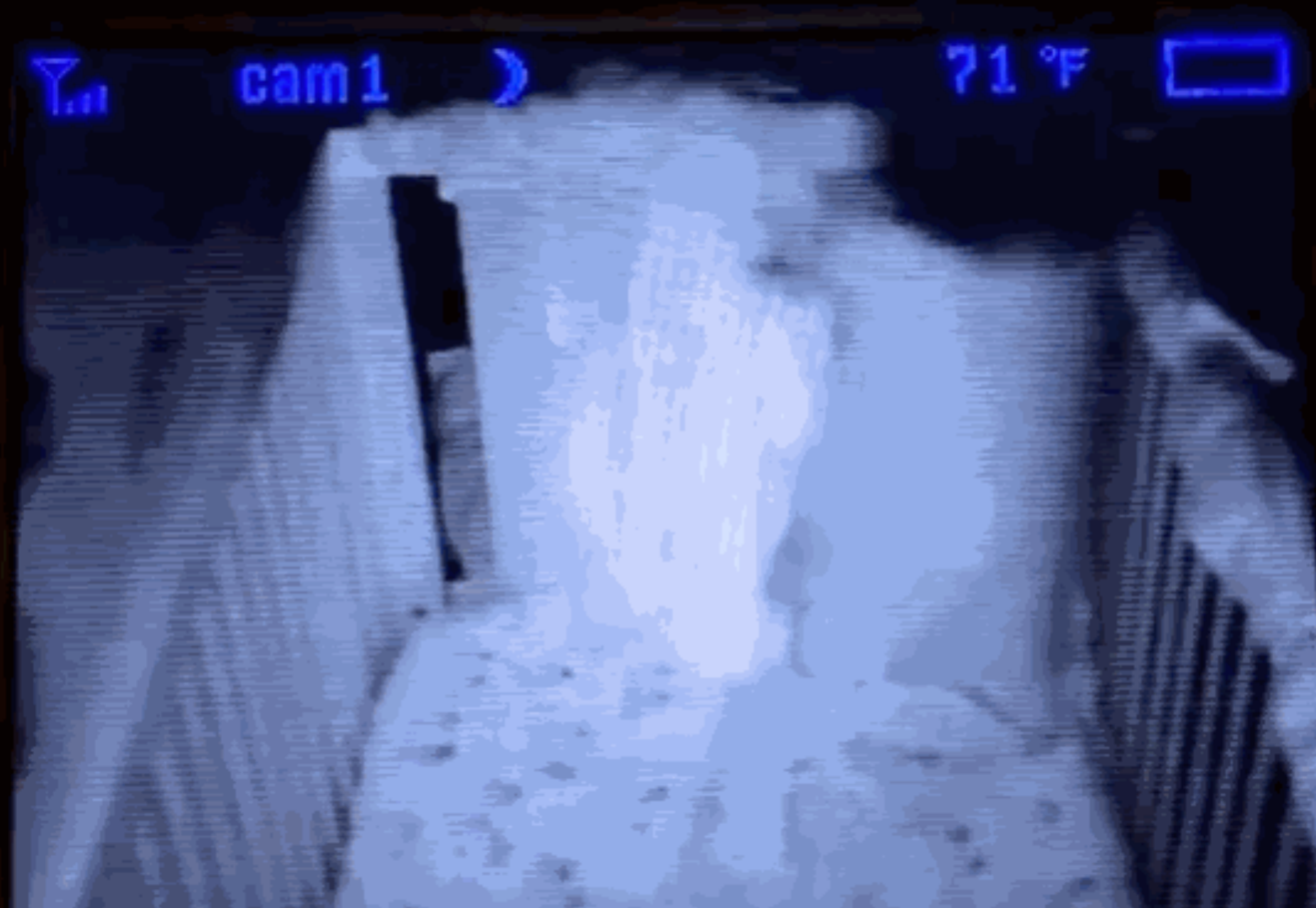
Andrew Hao

@andrewhao
ahao@lyft.com

Act I

Night & Chaos

put the little one down


cam1
71 °F

helplessness

frustration

anger

despair

I need data

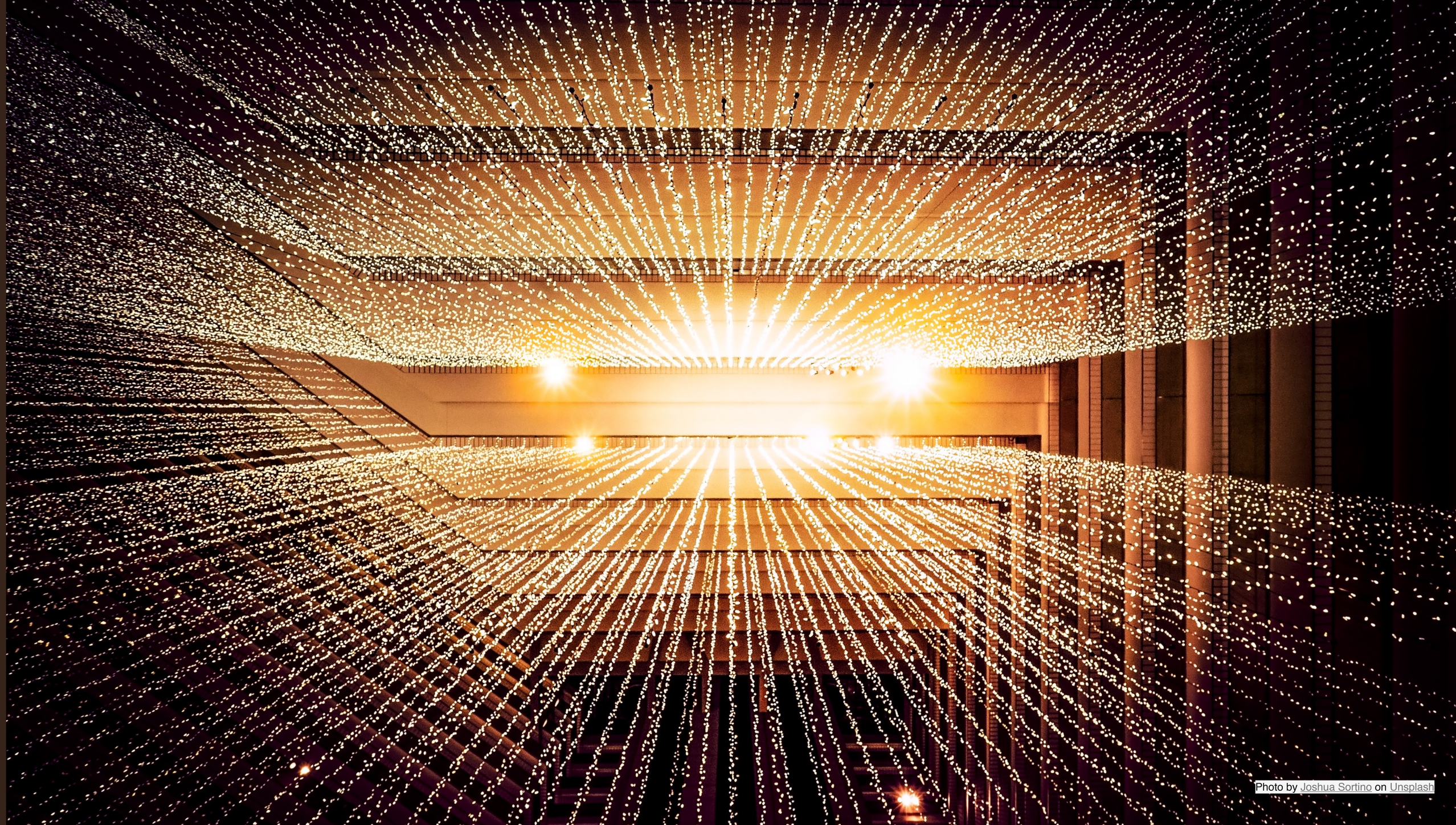


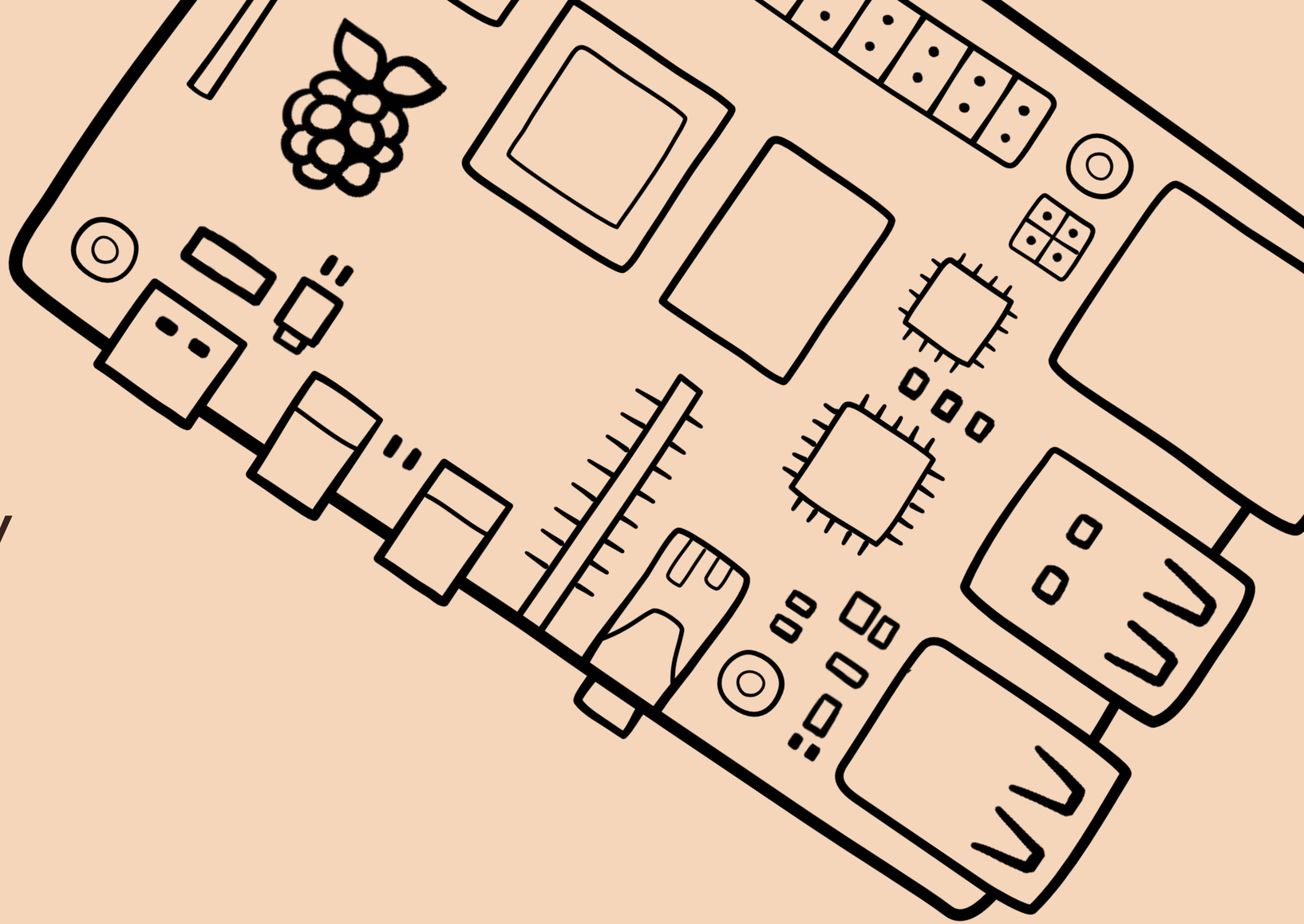
Photo by Joshua Sortino on Unsplash

Act II

Enter the Machine

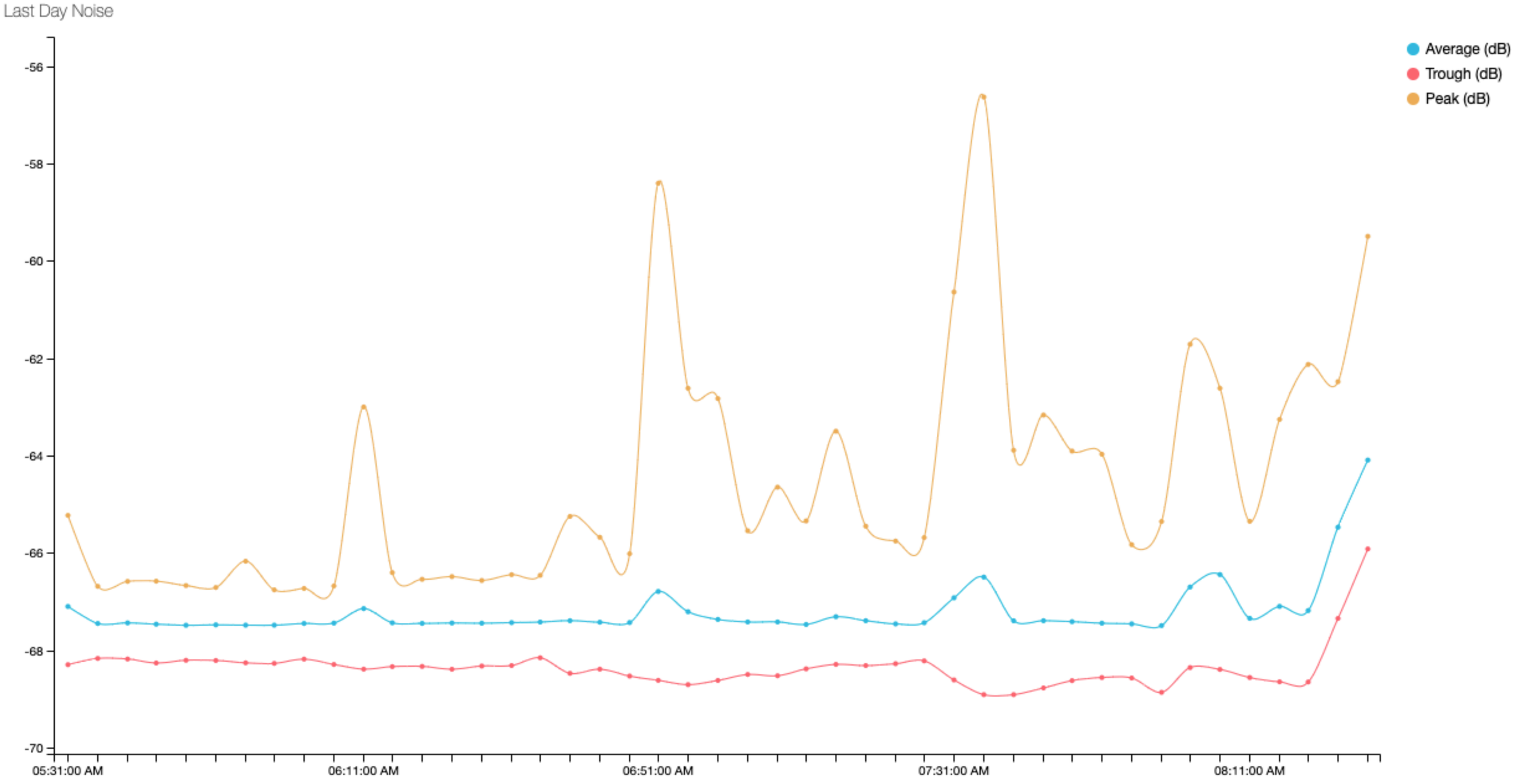
Parts List

- Raspberry Pi
- USB microphone
- DHT22 temp/humidity sensor

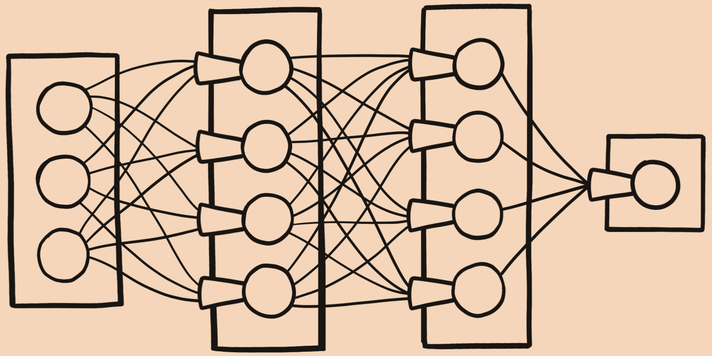


```
$ arecord --device=hw:1,0 --format S16_LE --rate 22050 -c1 -
d 10 "${RECORDING_FILE}"
$ sox -V3 ${RECORDING_FILE} -n stats 2>&1 | grep dB

Pk lev dB      -57.05
RMS lev dB     -67.10
RMS Pk dB      -64.94
RMS Tr dB      -68.26
```


Last Day Noise
Average (dB)
Trough (dB)
Peak (dB)
-56
-58
-60
-62
-64
-66
-68
-70
05:31:00 AM
06:11:00 AM
06:51:00 AM
07:31:00 AM
08:11:00 AM

Deep learning & neural nets



Get started with TensorFlow
Learn and use ML
Research and experimentation
ML at production scale
Generative models
Distributed training
Images
Sequences
Load data
Data representation
Non-ML

Simple Audio Recognition

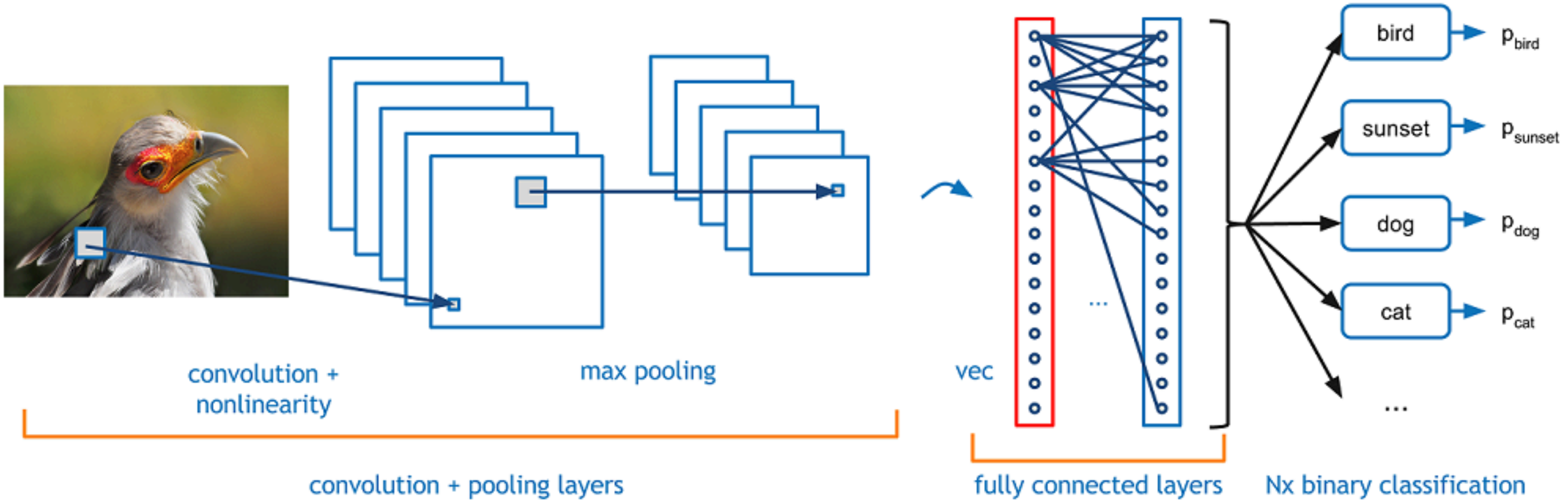
This tutorial will show you how to build a basic speech recognition network that recognizes ten different words. It's important to know that real speech and audio recognition systems are much more complex, but like MNIST for images, it should give you a basic understanding of the techniques involved. Once you've completed this tutorial, you'll have a model that tries to classify a one second audio clip as either silence, an unknown word, "yes", "no", "up", "down", "left", "right", "on", "off", "stop", or "go". You'll also be able to take this model and run it in an Android application.

Preparation

You should make sure you have TensorFlow installed, and since the script downloads over 1GB of training data, you'll need a good internet connection and enough free space on your machine. The training process itself can take several hours, so make sure you have a machine available for that long.

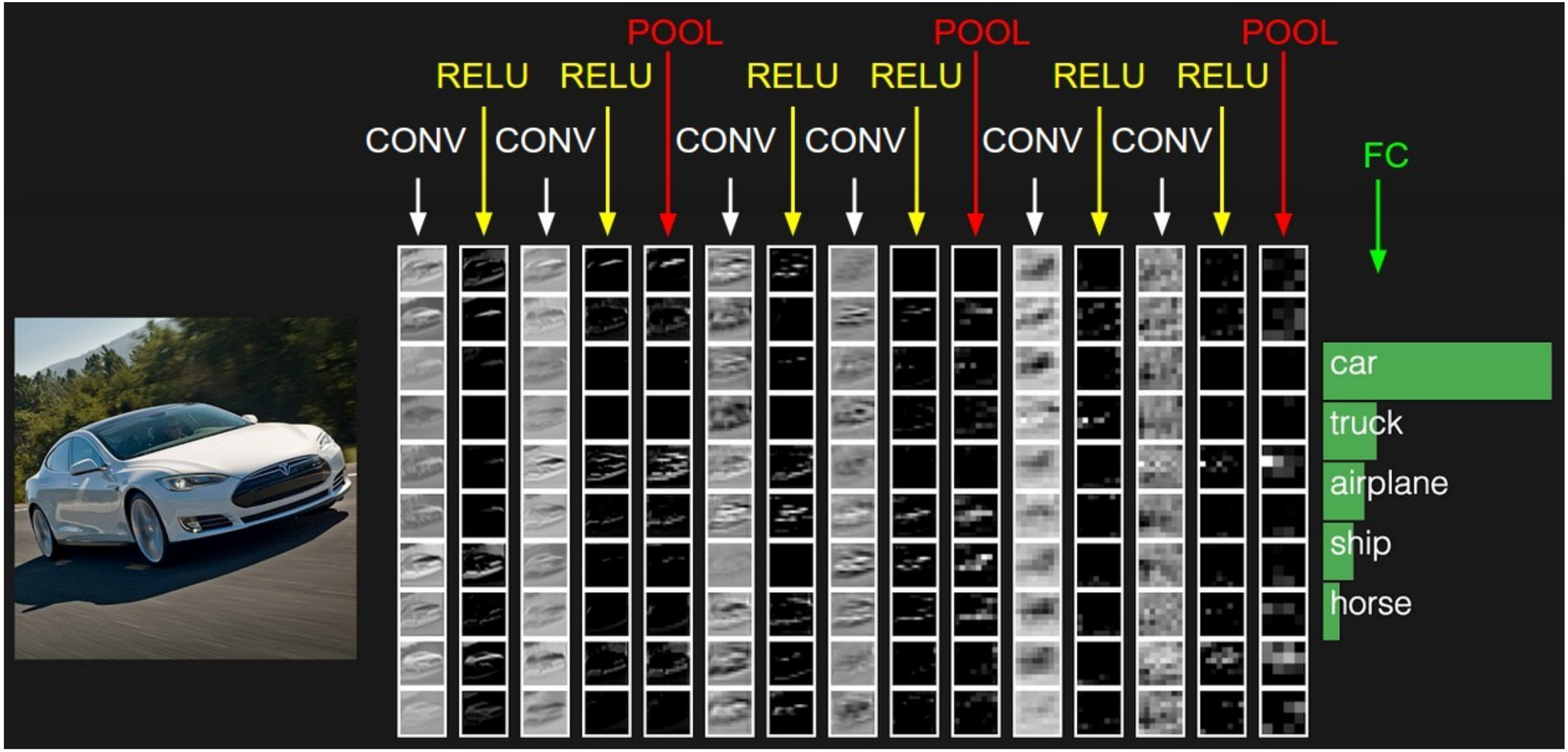
Training

convolutional neural network



https://www.fastcompany.com/3037882/how-flickrs-deep-learning-algorithms-see-whats-in-your-photos

CNNs

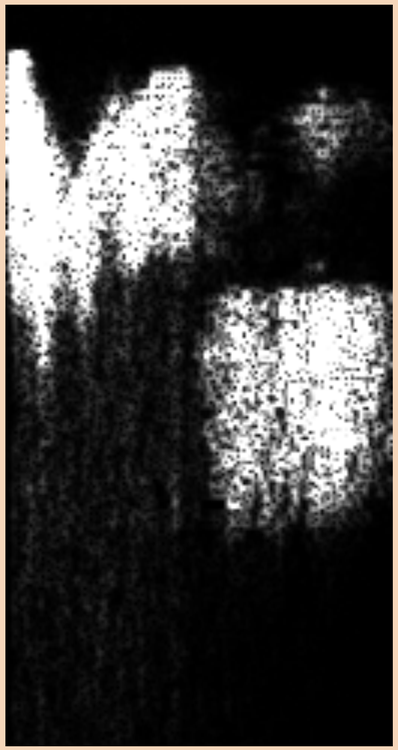


Karpathy, Andrej "Convolutional Neural Networks (CNNs / ConvNets)" (http://cs231n.github.io/convolutional-networks/)

audio spectrogram = image

frequency

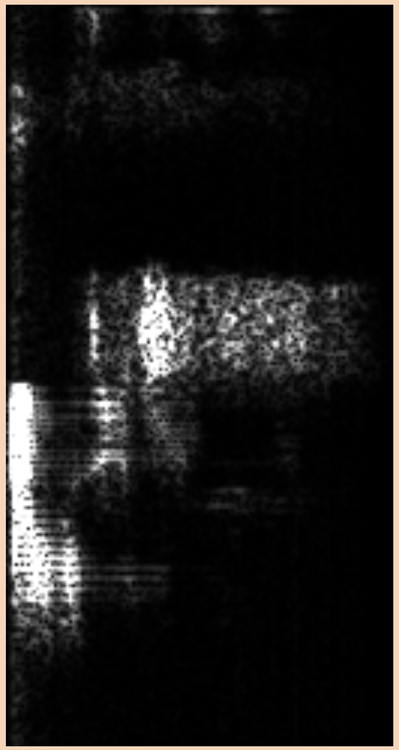
time



“yes”

frequency

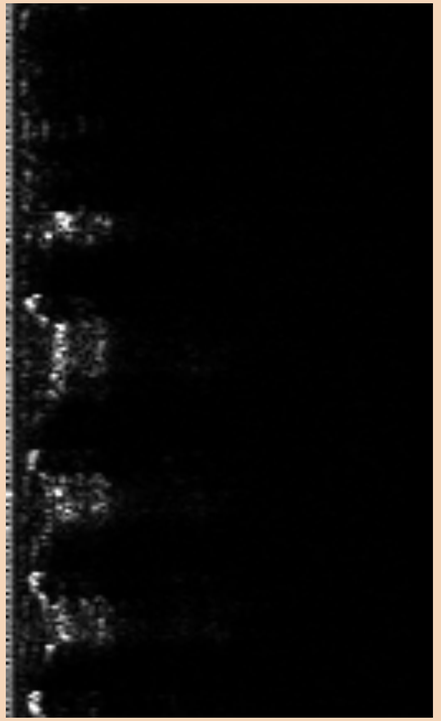
time



“Sheila”

frequency

time



“wah, wah, wah”

INTERSPEECH 2015

Convolutional Neural Networks for Small-footprint Keyword Spotting

Tara N. Sainath, Carolina Parada

Google, Inc. New York, NY, U.S.A

{tsainath, carolinap}@google.com

Abstract

We explore using Convolutional Neural Networks (CNNs) for a small-footprint keyword spotting (KWS) task. CNNs are attractive for KWS since they have been shown to outperform DNNs with far fewer parameters. We consider two different applications in our work, one where we limit the number of multiplications of the KWS system, and another where we limit the number of parameters. We present new CNN architectures to address the constraints of each applications. We find that the CNN architectures offer between a 27-44% relative improvement in false reject rate compared to a DNN, while fitting into the constraints of each application.

1. Introduction

With the rapid development of mobile devices, speech-related technologies are becoming increasingly popular. For example, Google offers the ability to search by voice [1] on Android phones, while personal assistants such as Google Now, Apple's Siri, Microsoft's Cortana and Amazon's Alexa, all utilize speech recognition to interact with these systems. Google has enabled a fully hands-free speech recognition experience, known as "Ok Google" [2], which continuously listens for specific keywords to initiate voice input. This keyword spotting (KWS) system runs on mobile devices, and therefore must have

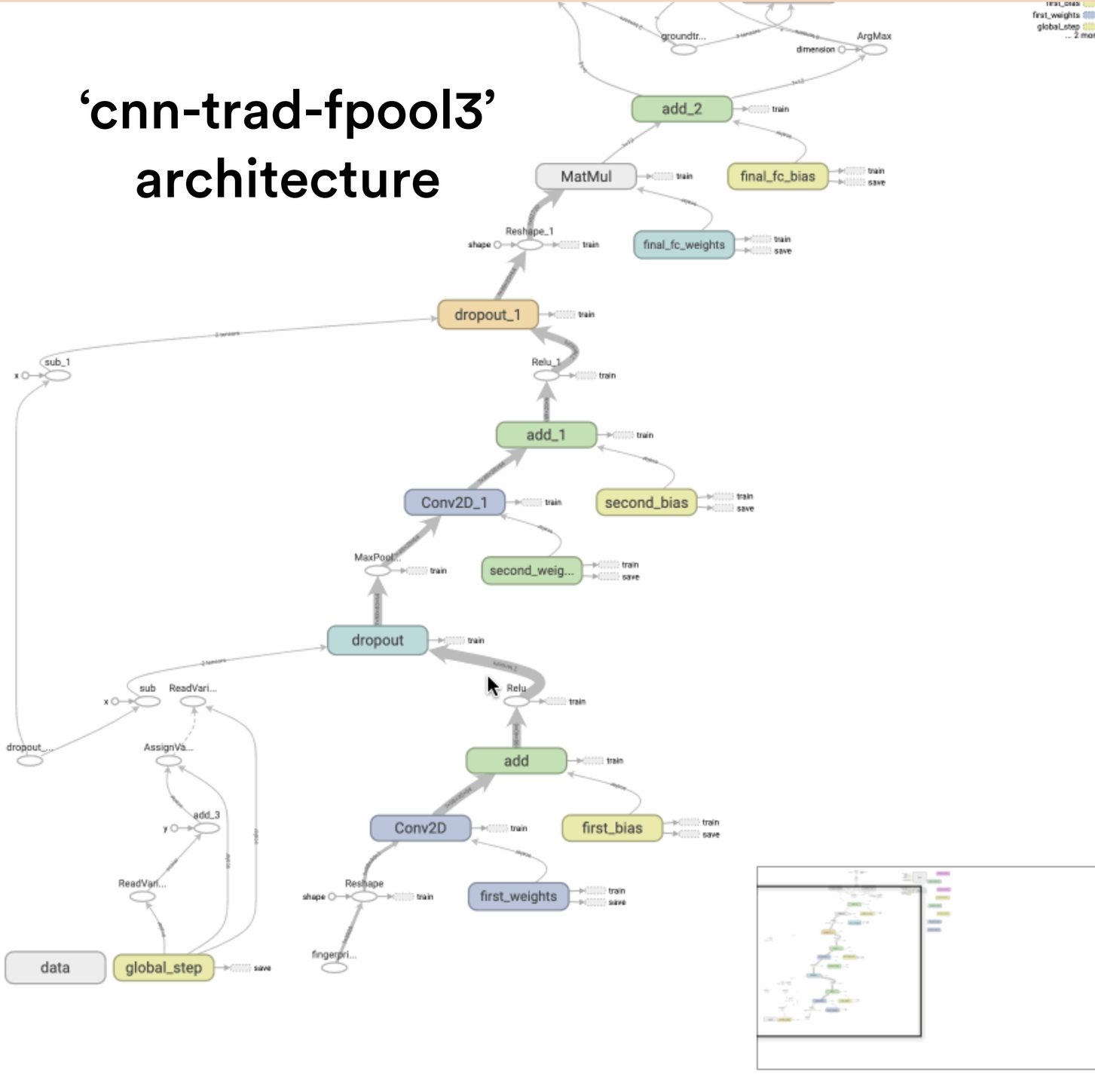
duce feature variation. While DNNs of sufficient size could indeed capture translational invariance, this requires large networks with lots of training examples. CNNs on the other hand capture translational invariance with far fewer parameters by averaging the outputs of hidden units in different local time and frequency regions.

We are motivated to look at CNNs for KWS given the benefits CNNs have shown over DNNs with respect to improved performance and reduced model size [4, 5, 6]. In this paper, we look at two applications of CNNs for KWS. First, we consider the problem where we must limit the overall computation of our KWS system, that is parameters and multiplies. With this constraint, typical architectures that work well for CNNs and pool in frequency only [8], cannot be used here. Thus, we introduce a novel CNN architecture which does not pool but rather strides the filter in frequency, to abide within the computational constraints issue. Second, we consider limiting the total number of parameters of our KWS system. For this problem, we show we can improve performance by pooling in time and frequency, the first time this has been shown to be effective for speech without using multiple convolutional blocks [5, 9].

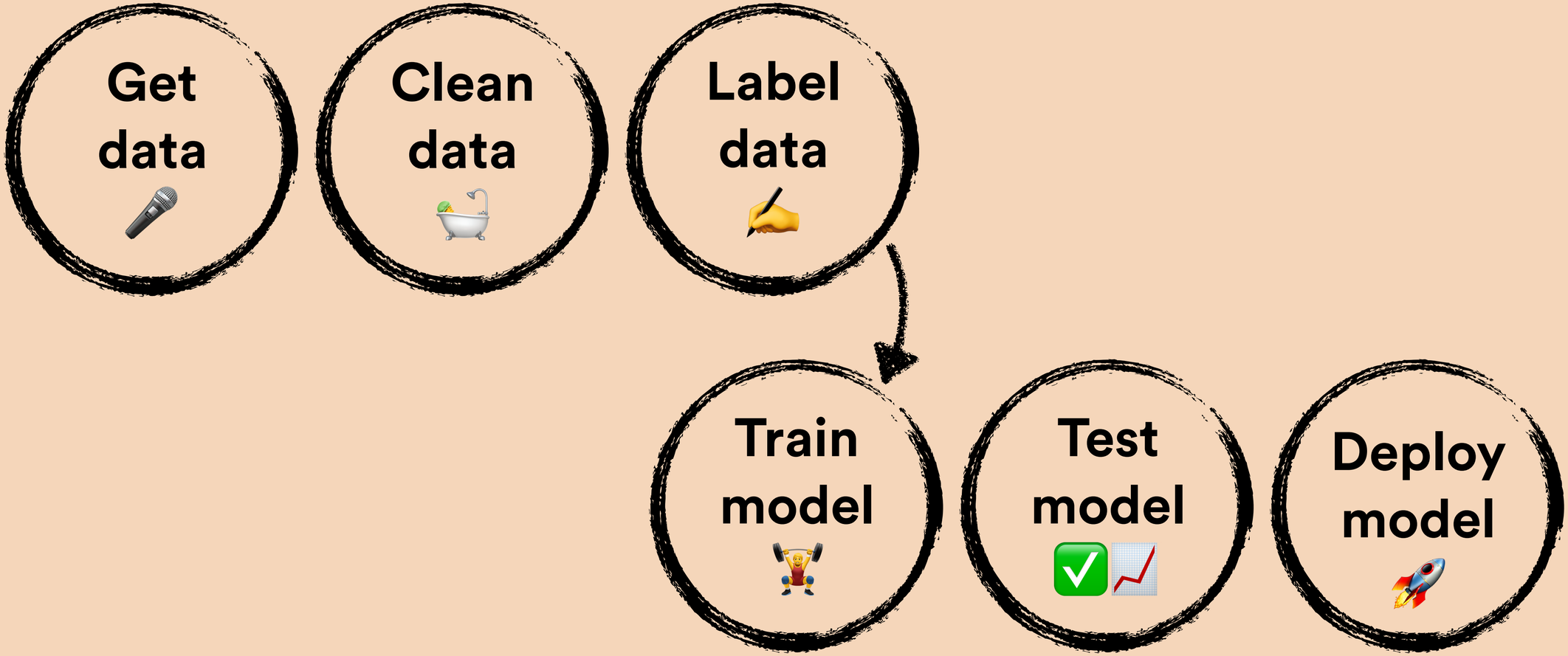
We evaluate our proposed CNN architectures on a KWS task consisting of 14 different phrases. Performance is measured by looking at the false reject (FR) rate at the operating threshold of 1 false alarm (FA) per hour. In the task where we

‘cnn-trad-fpool3’ architecture

```
(fingerprint_input)
        v
[Conv2D]<-(weights)
        v
 [BiasAdd]<-(bias)
        v
      [Relu]
        v
    [MaxPool]
        v
[Conv2D]<-(weights)
        v
 [BiasAdd]<-(bias)
        v
      [Relu]
        v
    [MaxPool]
        v
[MatMul]<-(weights)
        v
 [BiasAdd]<-(bias)
        v
```

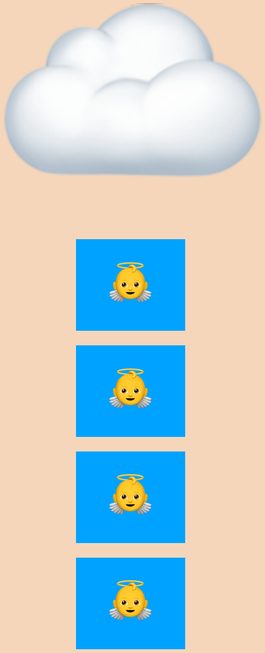


The Secret Life of ML Models



Get data 🎤

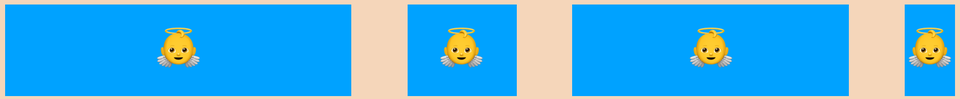
- Record data continuously in a cronjob
- Upload to the cloud



Clean data 🛁

```
# Concatenate
sox crying/**/*.wav tmp/batch-crying.wav
```

- Align to a frame
- Normalize volume

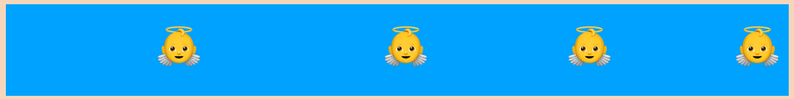


Clean data

🛁

```
# Split
ffmpeg -i tmp/batch-crying.wav -
f segment -segment_time 10.1 -c
```

- Align to a frame
- Normalize volume



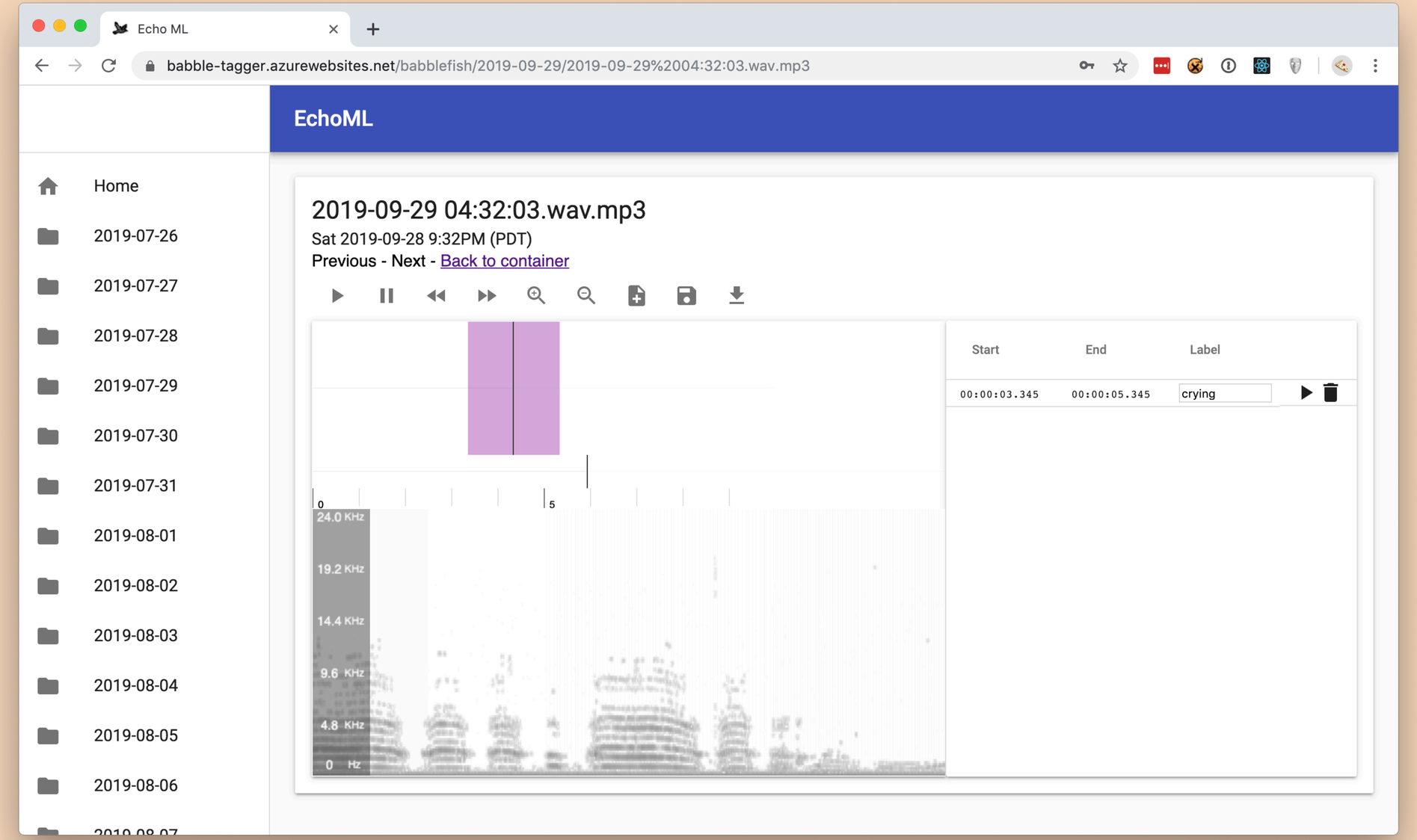
Clean data

```
# Trim to exactly 10000ms, boost 45 dB, resample @ 22050 hz
sox $in data/crying/$out vol 45 dB trim 0 10 rate 22050
```

- Align to a frame
- Normalize volume

Label data ✍️

Using EchoML to classify data



Label data ✍️

- Bucket each audio sample in a folder by label

crying

whining

dog_barking

city_noise


```
python app/train.py
--data_url=
--data_dir=./data
--wanted_words=room_empty,whining,crying
--sample_rate=22050
--clip_duration_ms=10000
--how_many_training_steps=400,50
--train_dir=./training
```

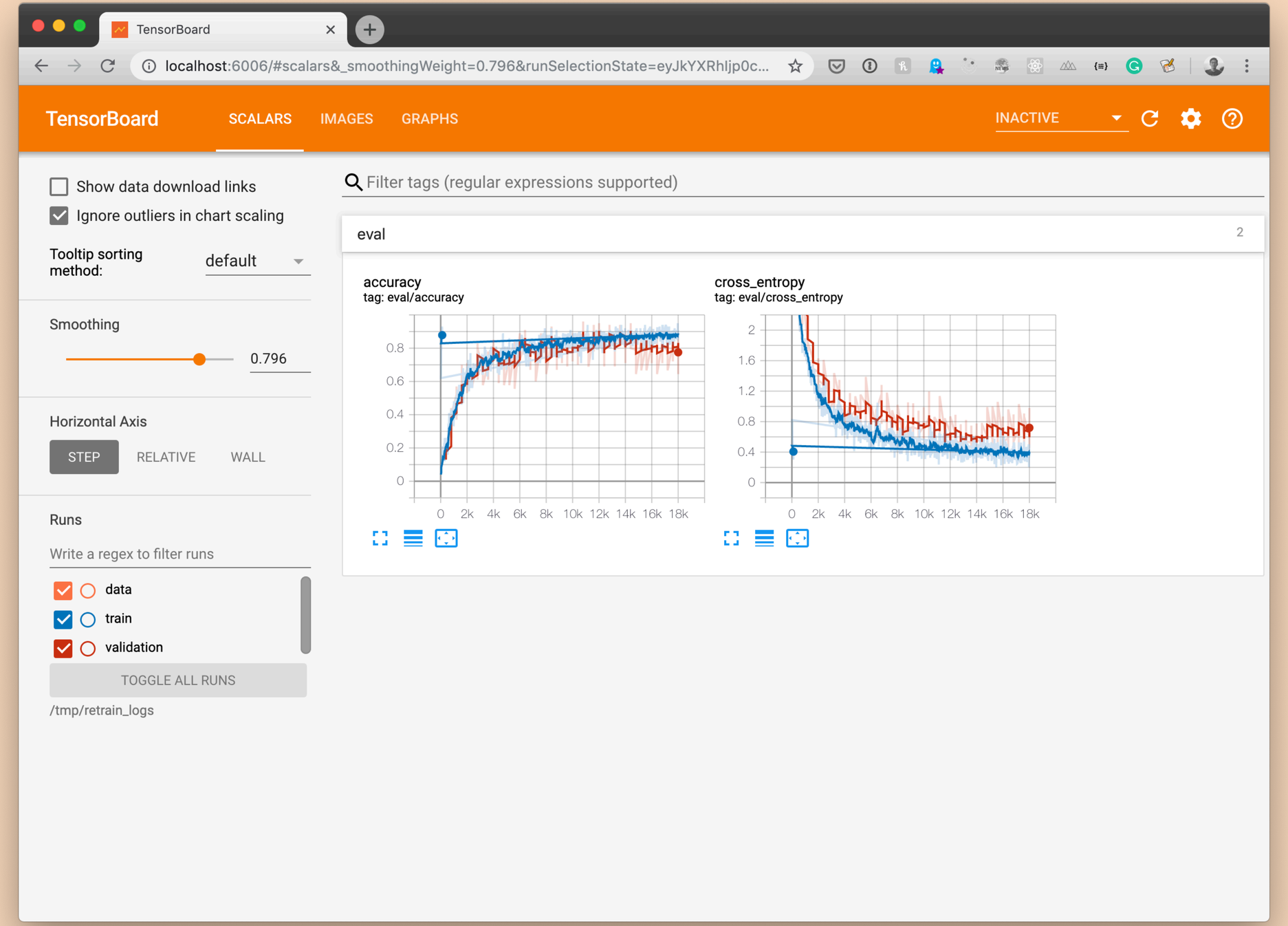

```
I1002 03:58:24.679805 4741027264 train.py:287] Step #430: rate 0.000100, accuracy 99.0%, cross entropy 0.04[95/3172]
INFO:tensorflow:Step #431: rate 0.000100, accuracy 99.0%, cross entropy 0.035736
I1002 03:59:02.294903 4741027264 train.py:287] Step #431: rate 0.000100, accuracy 99.0%, cross entropy 0.035736
INFO:tensorflow:Step #432: rate 0.000100, accuracy 98.0%, cross entropy 0.118002
I1002 03:59:40.747909 4741027264 train.py:287] Step #432: rate 0.000100, accuracy 98.0%, cross entropy 0.118002
INFO:tensorflow:Step #433: rate 0.000100, accuracy 98.0%, cross entropy 0.073030
I1002 04:00:19.569001 4741027264 train.py:287] Step #433: rate 0.000100, accuracy 98.0%, cross entropy 0.073030
INFO:tensorflow:Step #434: rate 0.000100, accuracy 97.0%, cross entropy 0.092732
I1002 04:00:55.971426 4741027264 train.py:287] Step #434: rate 0.000100, accuracy 97.0%, cross entropy 0.092732
INFO:tensorflow:Step #435: rate 0.000100, accuracy 100.0%, cross entropy 0.042072
I1002 04:01:35.398041 4741027264 train.py:287] Step #435: rate 0.000100, accuracy 100.0%, cross entropy 0.042072
INFO:tensorflow:Step #436: rate 0.000100, accuracy 100.0%, cross entropy 0.043324
I1002 04:02:13.348515 4741027264 train.py:287] Step #436: rate 0.000100, accuracy 100.0%, cross entropy 0.043324
INFO:tensorflow:Step #437: rate 0.000100, accuracy 100.0%, cross entropy 0.038082
I1002 04:02:56.560698 4741027264 train.py:287] Step #437: rate 0.000100, accuracy 100.0%, cross entropy 0.038082
INFO:tensorflow:Step #438: rate 0.000100, accuracy 98.0%, cross entropy 0.068574
I1002 04:03:35.357985 4741027264 train.py:287] Step #438: rate 0.000100, accuracy 98.0%, cross entropy 0.068574
INFO:tensorflow:Step #439: rate 0.000100, accuracy 99.0%, cross entropy 0.058624
I1002 04:04:13.681237 4741027264 train.py:287] Step #439: rate 0.000100, accuracy 99.0%, cross entropy 0.058624
INFO:tensorflow:Step #440: rate 0.000100, accuracy 99.0%, cross entropy 0.061955
I1002 04:04:51.182194 4741027264 train.py:287] Step #440: rate 0.000100, accuracy 99.0%, cross entropy 0.061955
INFO:tensorflow:Step #441: rate 0.000100, accuracy 97.0%, cross entropy 0.091335
I1002 04:05:31.424205 4741027264 train.py:287] Step #441: rate 0.000100, accuracy 97.0%, cross entropy 0.091335
INFO:tensorflow:Step #442: rate 0.000100, accuracy 100.0%, cross entropy 0.034299
I1002 04:06:07.957736 4741027264 train.py:287] Step #442: rate 0.000100, accuracy 100.0%, cross entropy 0.034299
INFO:tensorflow:Step #443: rate 0.000100, accuracy 99.0%, cross entropy 0.080818
I1002 04:06:49.358937 4741027264 train.py:287] Step #443: rate 0.000100, accuracy 99.0%, cross entropy 0.080818
INFO:tensorflow:Step #444: rate 0.000100, accuracy 99.0%, cross entropy 0.041448
I1002 04:07:28.075536 4741027264 train.py:287] Step #444: rate 0.000100, accuracy 99.0%, cross entropy 0.041448
INFO:tensorflow:Step #445: rate 0.000100, accuracy 100.0%, cross entropy 0.030128
I1002 04:08:06.771522 4741027264 train.py:287] Step #445: rate 0.000100, accuracy 100.0%, cross entropy 0.030128
INFO:tensorflow:Step #446: rate 0.000100, accuracy 99.0%, cross entropy 0.045735
I1002 04:08:47.032963 4741027264 train.py:287] Step #446: rate 0.000100, accuracy 99.0%, cross entropy 0.045735
INFO:tensorflow:Step #447: rate 0.000100, accuracy 99.0%, cross entropy 0.043301
I1002 04:09:25.154629 4741027264 train.py:287] Step #447: rate 0.000100, accuracy 99.0%, cross entropy 0.043301
INFO:tensorflow:Step #448: rate 0.000100, accuracy 97.0%, cross entropy 0.119577
I1002 04:10:01.199583 4741027264 train.py:287] Step #448: rate 0.000100, accuracy 97.0%, cross entropy 0.119577
INFO:tensorflow:Step #449: rate 0.000100, accuracy 97.0%, cross entropy 0.104553
```

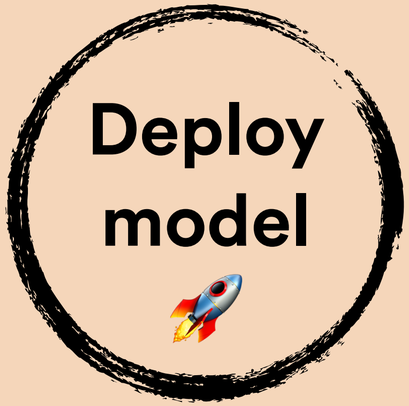
Train
mode

```
⌥⌘1                                    Default (tmux)
INFO:tensorflow:Step #449: rate 0.000100, accuracy 97.0%, cross entropy 0.104553                [58/3172]
I1002 04:10:40.378780 4741027264 train.py:287] Step #449: rate 0.000100, accuracy 97.0%, cross entropy 0.104553
INFO:tensorflow:Step #450: rate 0.000100, accuracy 98.0%, cross entropy 0.079582
I1002 04:11:18.869298 4741027264 train.py:287] Step #450: rate 0.000100, accuracy 98.0%, cross entropy 0.079582
INFO:tensorflow:Confusion Matrix:
 [[ 0  7  0  0  0]
 [ 0  7  0  0  0]
 [ 0  0 15  0  0]
 [ 0  0  0 33  0]
 [ 0  0  2  0 18]]
I1002 04:11:27.616060 4741027264 train.py:316] Confusion Matrix:
 [[ 0  7  0  0  0]
 [ 0  7  0  0  0]
 [ 0  0 15  0  0]
 [ 0  0  0 33  0]
 [ 0  0  2  0 18]]
INFO:tensorflow:Step 450: Validation accuracy = 89.0% (N=82)
I1002 04:11:27.616290 4741027264 train.py:319] Step 450: Validation accuracy = 89.0% (N=82)
INFO:tensorflow:Saving to "./training/conv.ckpt-450"
I1002 04:11:27.616428 4741027264 train.py:331] Saving to "./training/conv.ckpt-450"
INFO:tensorflow:set_size=79
I1002 04:11:27.838648 4741027264 train.py:336] set_size=79
WARNING:tensorflow:Confusion Matrix:
 [[ 0  7  0  0  0]
 [ 0  7  0  0  0]
 [ 0  0 10  0  0]
 [ 0  0  0 35  0]
 [ 0  0  0  1 19]]
W1002 04:11:36.138626 4741027264 train.py:357] Confusion Matrix:
 [[ 0  7  0  0  0]
 [ 0  7  0  0  0]
 [ 0  0 10  0  0]
 [ 0  0  0 35  0]
 [ 0  0  0  1 19]]
WARNING:tensorflow:Final test accuracy = 89.9% (N=79)
W1002 04:11:36.139487 4741027264 train.py:359] Final test accuracy = 89.9% (N=79)

babblefish on  master [!?] via venv took 4h 59m 49s
0  0 ❯ Python -  1 ❯ zsh  2 ❯ [tmux] *                    2019-10-02 〈 09:41  Andrews-MacBook-P
```

Test model ✅📈

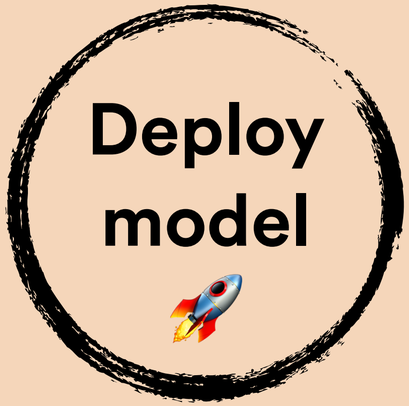

TensorBoard
localhost:6006/#scalars&_smoothingWeight=0.796&runSelectionState=eyJkYXRhIjp0c...
TensorBoard
SCALARS
IMAGES
GRAPHS
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method:
default
Smoothing
0.796
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
data
train
validation
TOGGLE ALL RUNS
/tmp/retrain_logs
Filter tags (regular expressions supported)
eval
2
accuracy
tag: eval/accuracy
0.8
0.6
0.4
0.2
0
0 2k 4k 6k 8k 10k 12k 14k 16k 18k
cross_entropy
tag: eval/cross_entropy
2
1.6
1.2
0.8
0.4
0
0 2k 4k 6k 8k 10k 12k 14k 16k 18k



Freeze the model

```
$ python app/freeze.py
  --start_checkpoint=./training/conv.ckpt-450
  --output_file=./graph.pb
  --clip_duration_ms=10000
  --sample_rate=22050
  --wanted_words=white_noise,room_empty,crying
  --data_dir=./data

$ cp training/conv_labels.txt .
```



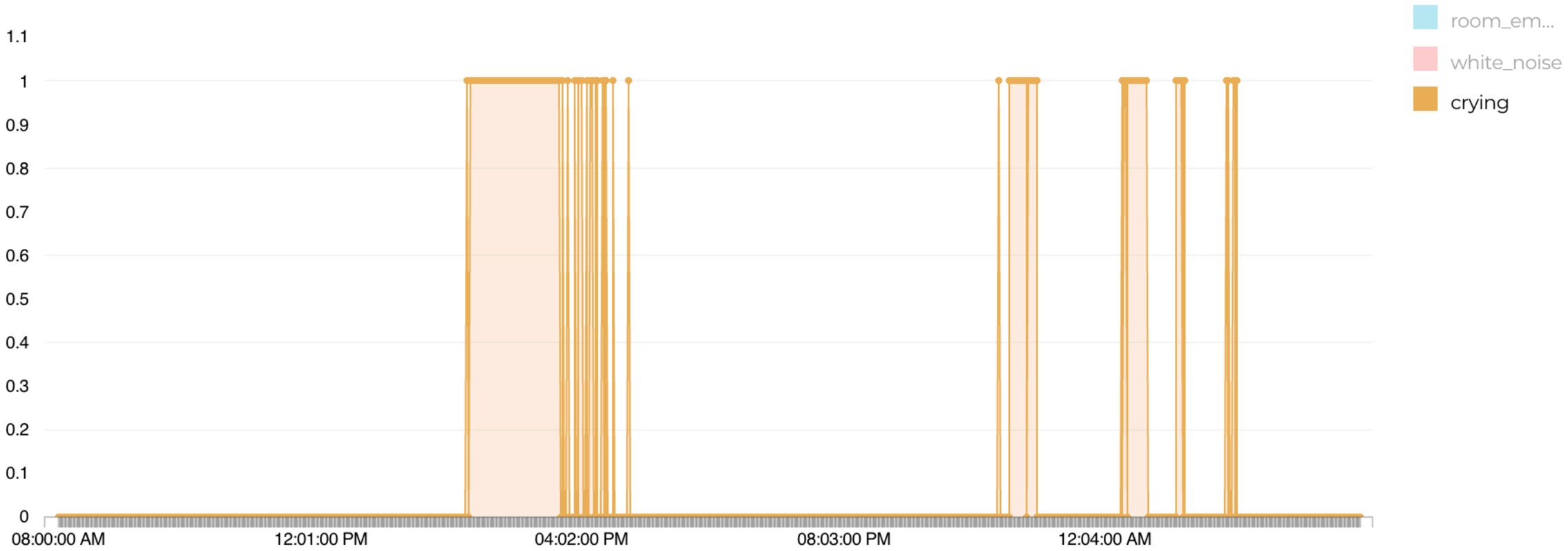
Execute the model in prod

```
$ arecord --format S16_LE --rate 22050 -c1
-d 10 $wav
$ python app/label_wav.py
   --graph=./graph.pb
   --labels=./conv_labels.txt
   --wav=$wav
```

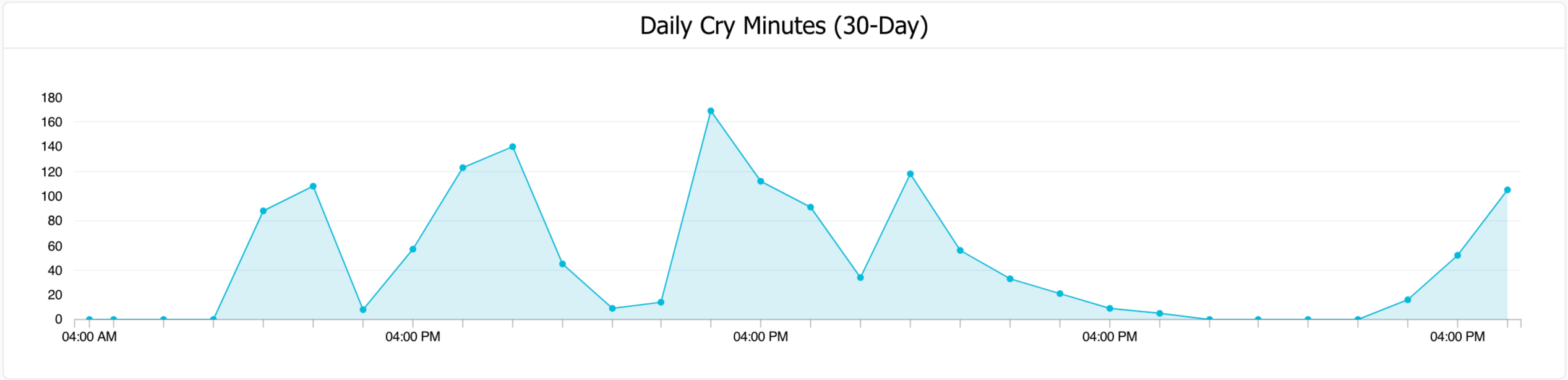
```
> crying (score = 0.95350)
   room_empty (score = 0.01888)
   _silence_ (score = 0.01746)
```

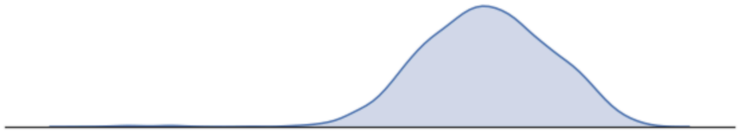
Will it work?

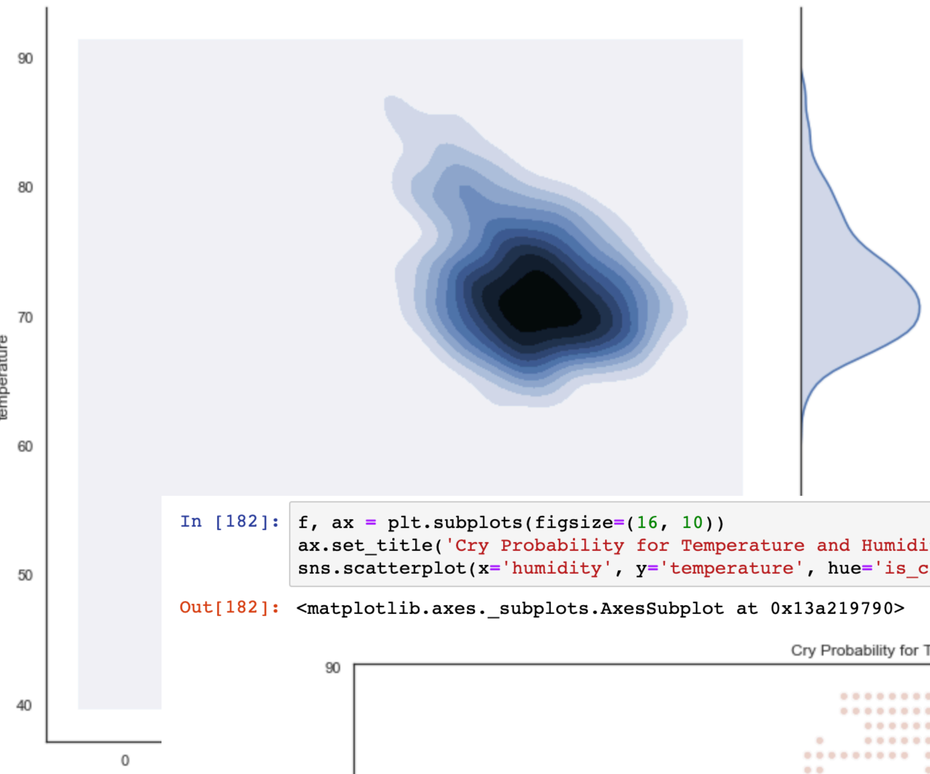
Yes!


room_em...
white_noise
crying
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
08:00:00 AM
12:01:00 PM
04:02:00 PM
08:03:00 PM
12:04:00 AM

..yes?


Daily Cry Minutes (30-Day)
180
160
140
120
100
80
60
40
20
0
04:00 AM
04:00 PM
04:00 PM
04:00 PM
04:00 PM

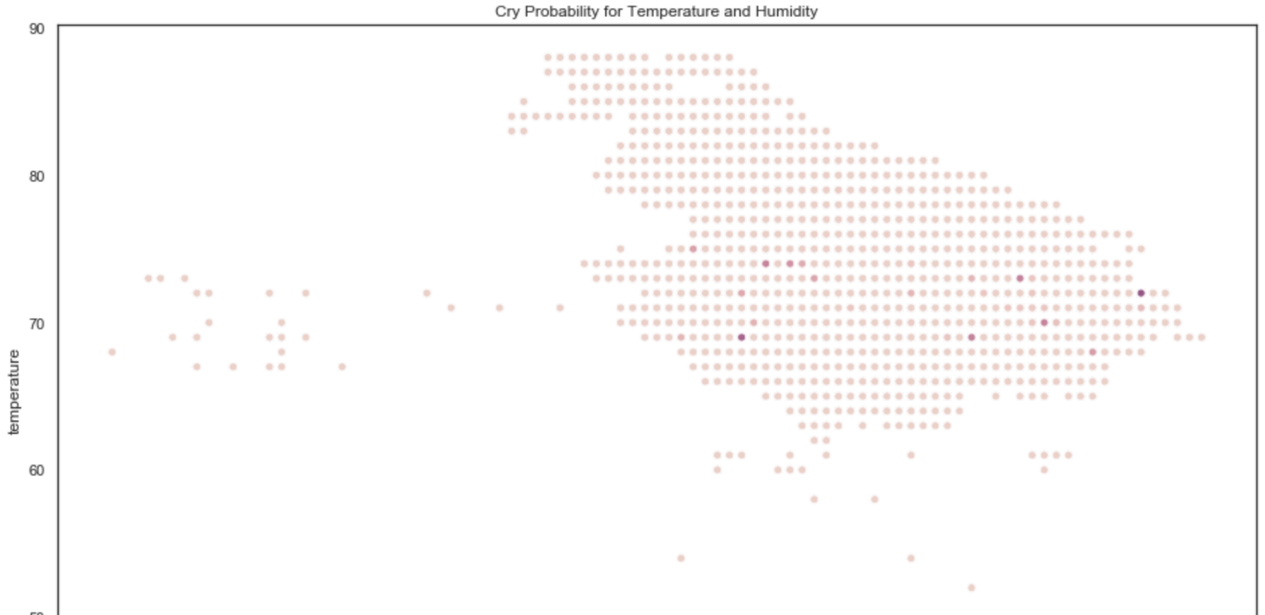




hmm.

```
In [182]: f, ax = plt.subplots(figsize=(16, 10))
          ax.set_title('Cry Probability for Temperature and Humidity')
          sns.scatterplot(x='humidity', y='temperature', hue='is_crying', data=df_humidity_temp, hue_norm=(0,1), legend=False)
```

```
Out[182]: <matplotlib.axes._subplots.AxesSubplot at 0x13a219790>
```

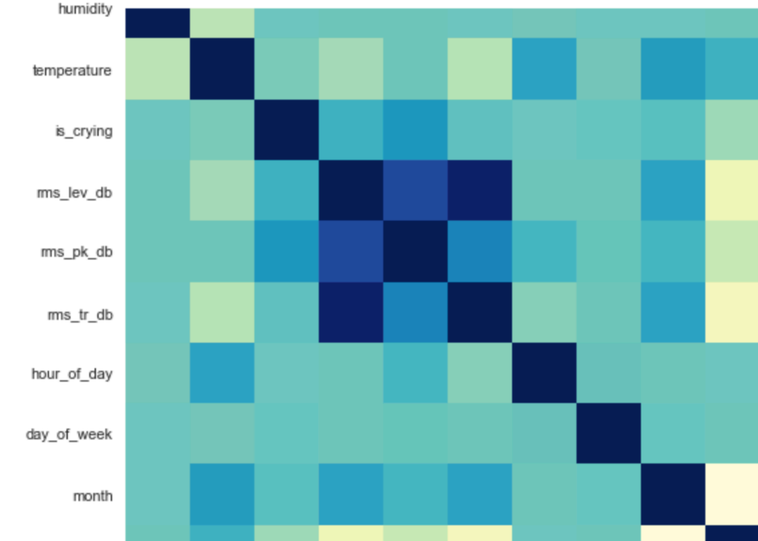


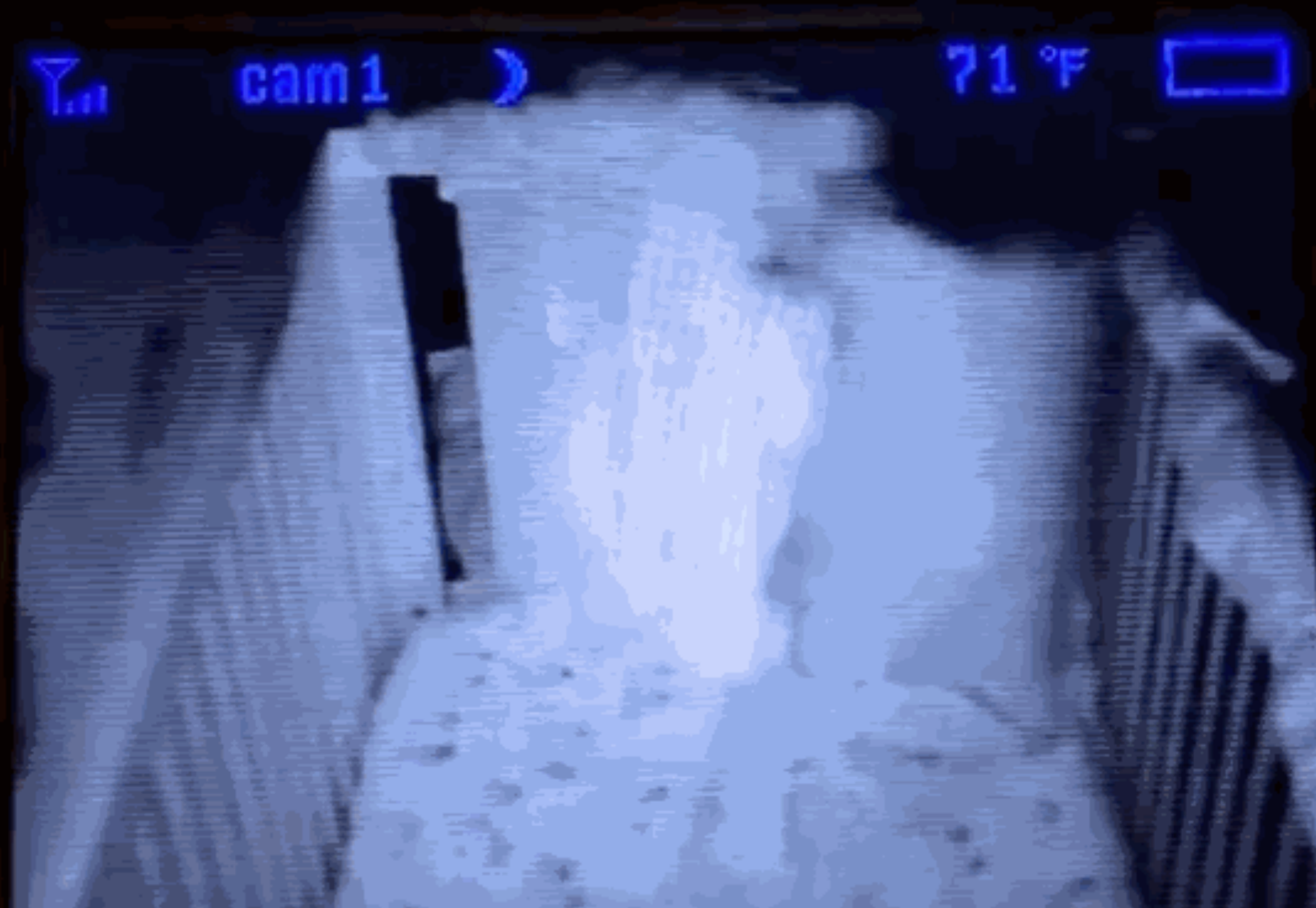

```
In [209]: df_corr = df_merged.corr()['is_cry
          df_corr.reset_index().sort_values
```

Out[209]:

	index	is_crying
2	is_crying	1.000000
4	rms_pk_db	0.318907
3	rms_lev_db	0.176254
8	month	0.050564
5	rms_tr_db	0.039469
7	day_of_week	0.010542
10	minute_of_day	0.002138
6	hour_of_day	0.002137
0	humidity	0.000150
1	temperature	0.048025

```
Out[197]: <matplotlib.axes._subplots.AxesSubplot at 0x13a571f50>
```



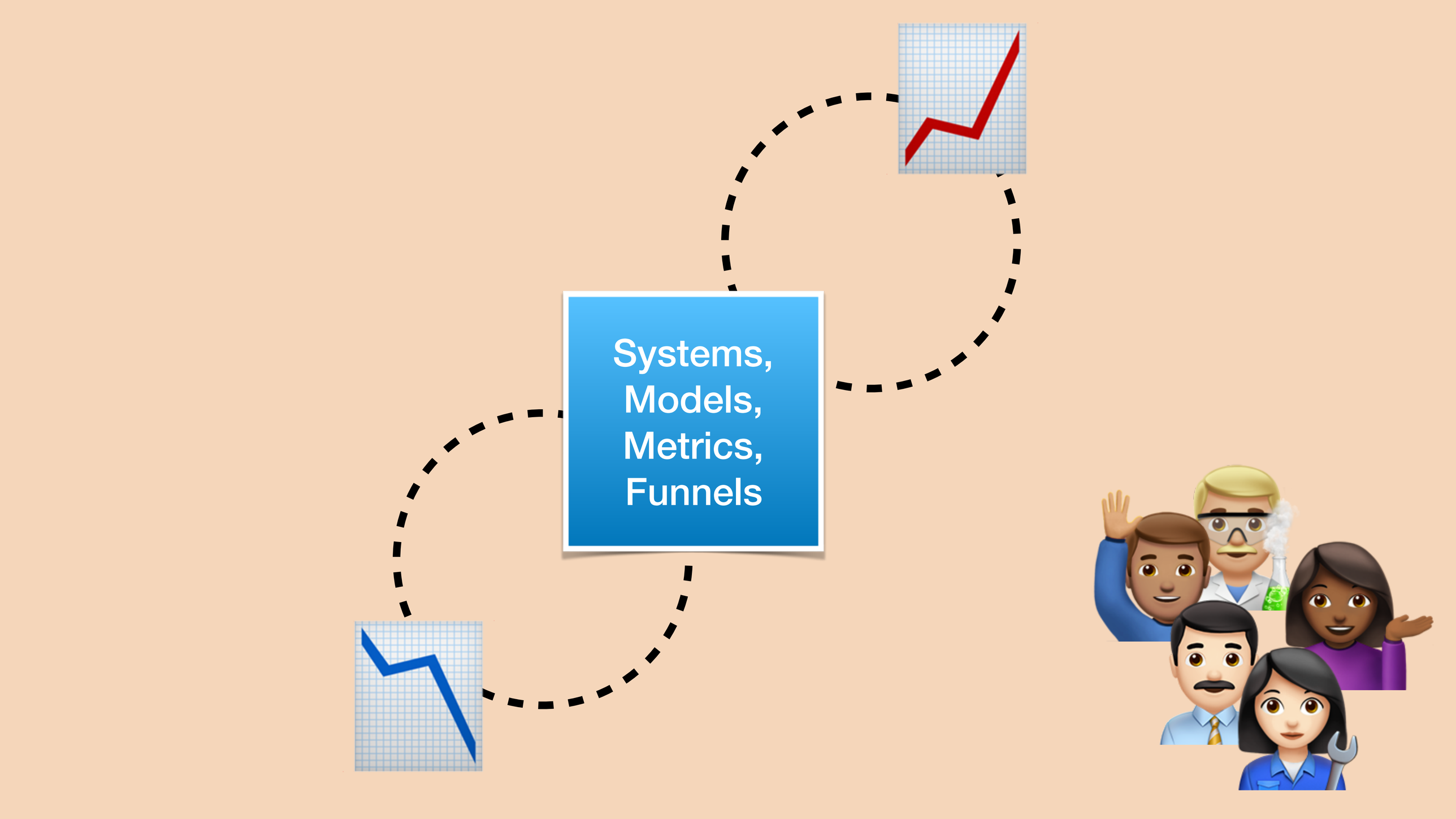

cam1
71 °F

Act III

The Illusion of Insight

We just looked at each other


cam1
78°F

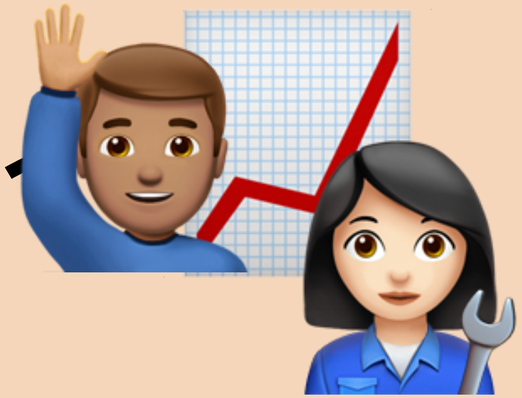

Systems,
Models,
Metrics,
Funnels

When we lose insights about the humans we serve,

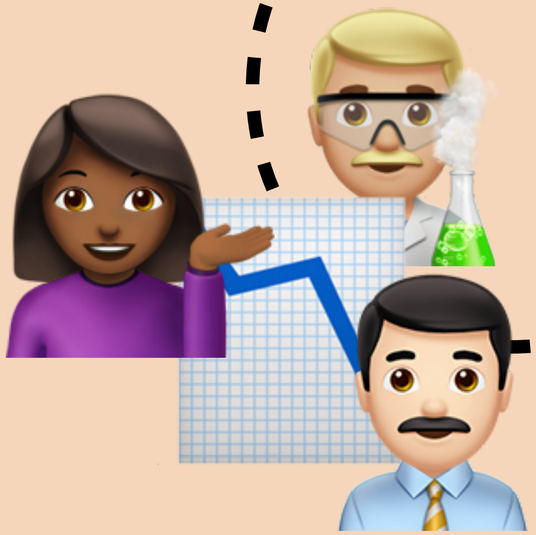
It’s ultimately bad for business

In a Big Data world,

Let’s re-emphasize the human experience



Systems,
Models,
Metrics,
Funnels



Are our assumptions right?
How do we find out?

Human-centered

UX research
Anthropology
Lean Startup: Customer interviews

Human-centered ML?

Personas

Photo by Toa Heftiba on Unsplash

Jenna

- Customer
- Recent college graduate
- Aspiring indie electronica producer
- Wants to pay off college loans
- Challenges: ...

Photo by Philip Martin on Unsplash

Reeve

- Internal stakeholder
- Director, Digital Marketing
- Concerned about ad budget spend

Photo by Jhon David on Unsplash

Lydia

- Customer
- Civil engineer
- Mother of one & caretaker of her aging parents
- Wants to save for her daughter's college fund
- Challenges: ...

We need human-centered design at all stages of the ML lifecycle

Let’s start by connecting with people first

Thank you

github.com/andrewhao/babblefish

Andrew Hao
@andrewhao
ahao@lyft.com