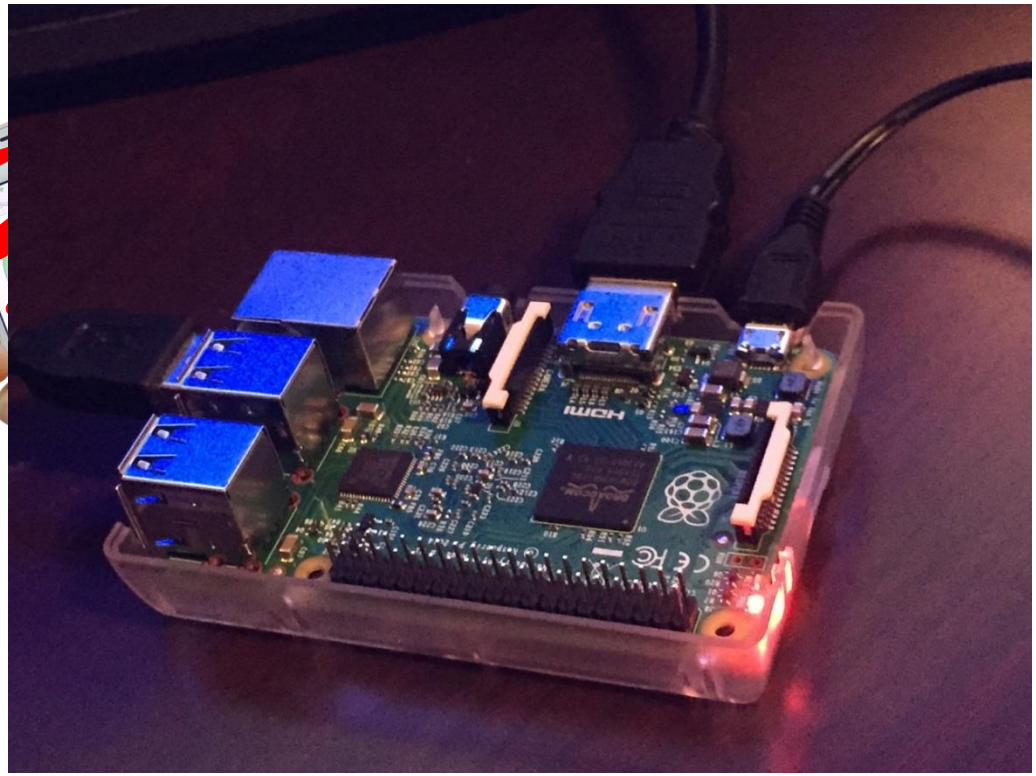
Reliable Real-time Lip Reading

with no sound and minimal ado

wasn’t quite sure what to do with this birthday present

Apps take too long to open

I am not a computer person



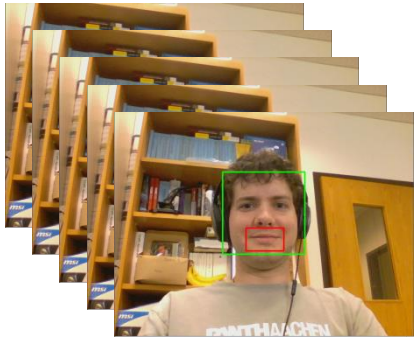
Long Term Plan

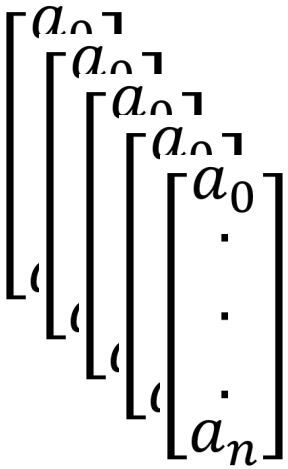
Why Machine Learning?

- Project requirements might change in future
  - Include audio
  - Add more color options
  - Add functionality for other devices

- Data may change
  - New camera
  - New desk location ➔ new lighting conditions

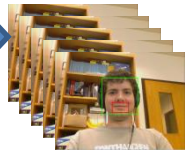
Implementation





Implementation

Viola Jones
Boosted Haar
Cascade



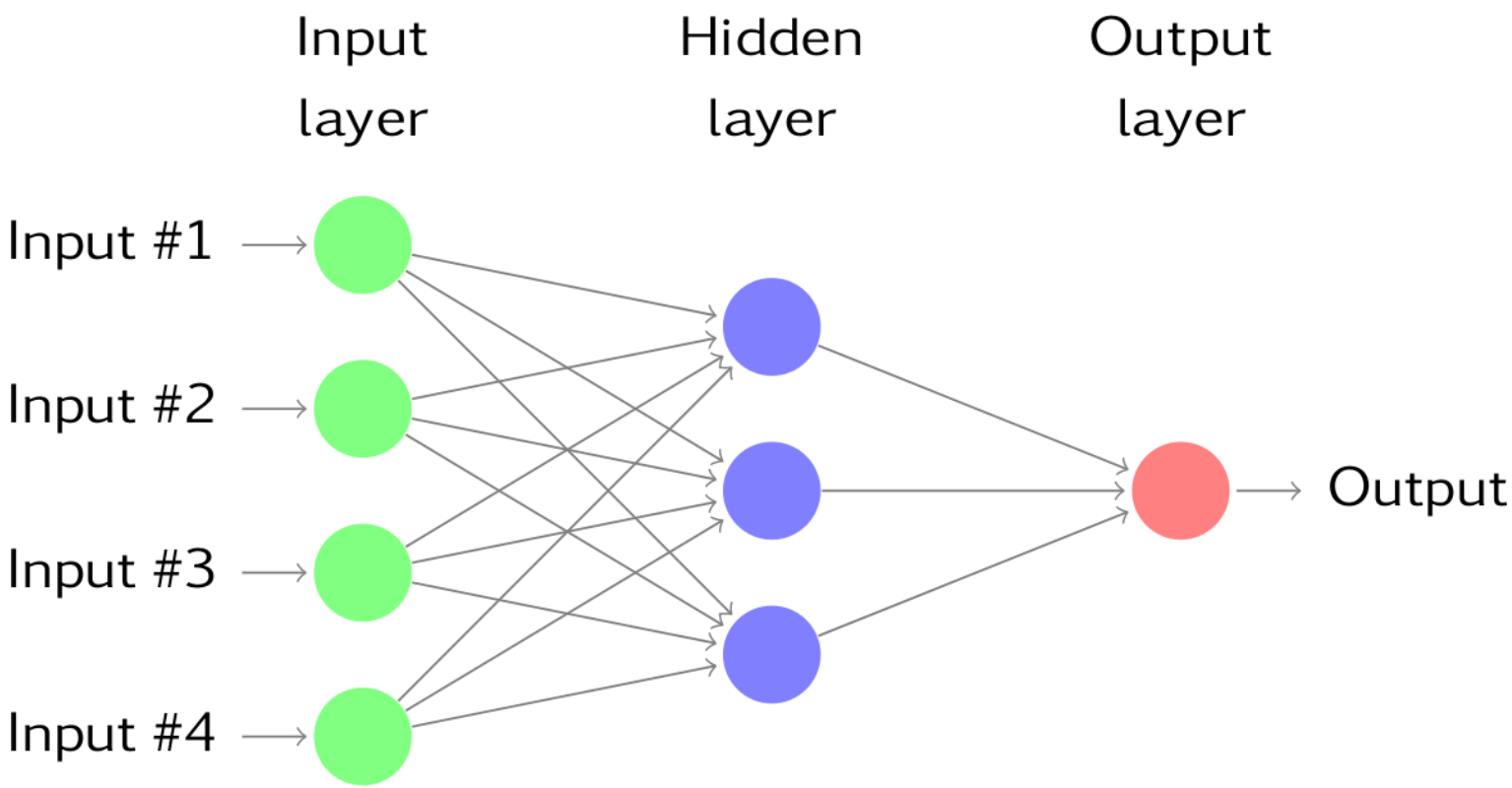
Crop

(Unsupervised)
Variational
Convolution
Neural Net
Autoencoder

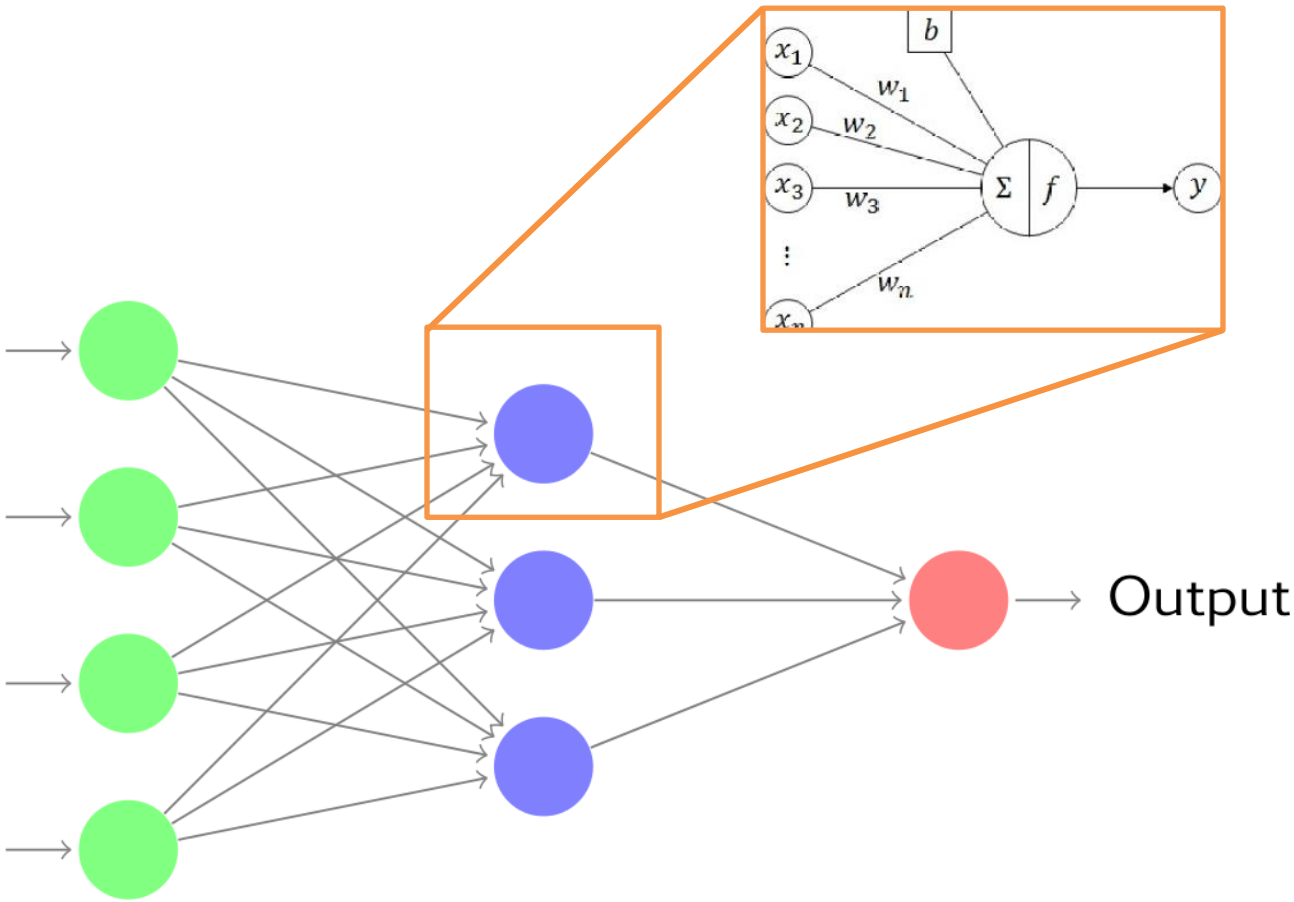
$$\begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix}$$

(Supervised) Recurrent
Neural Net

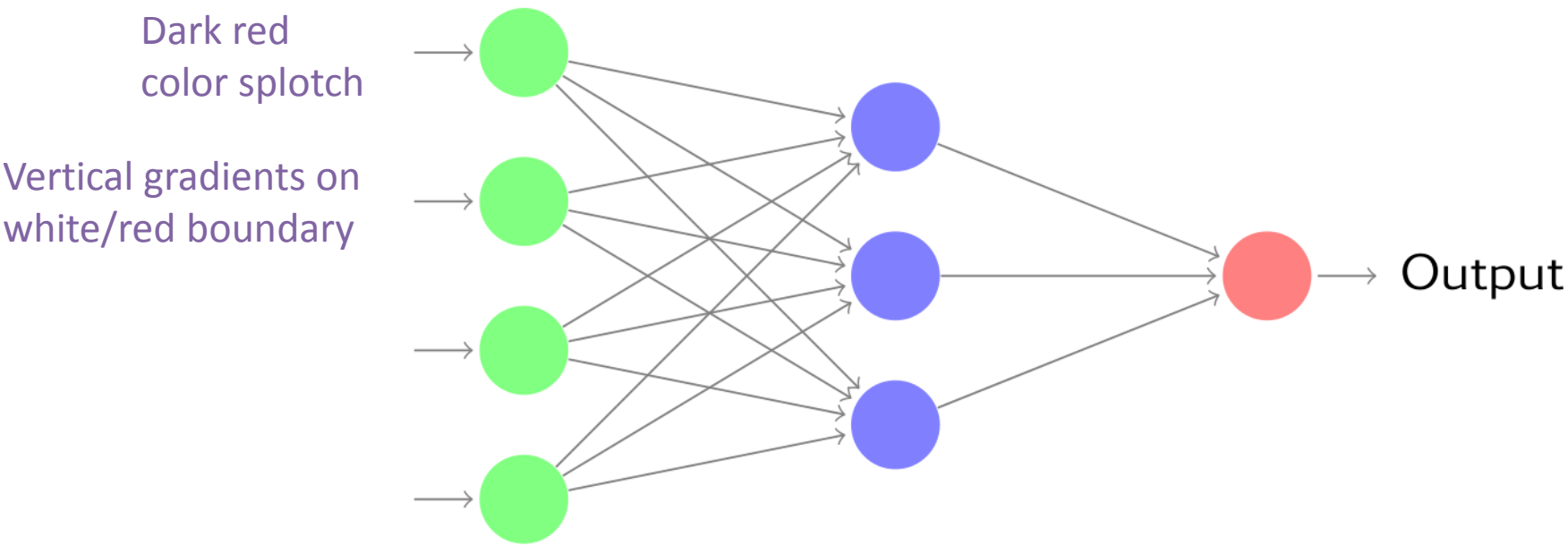
Variational Convolution
Neural Net Autoencoder



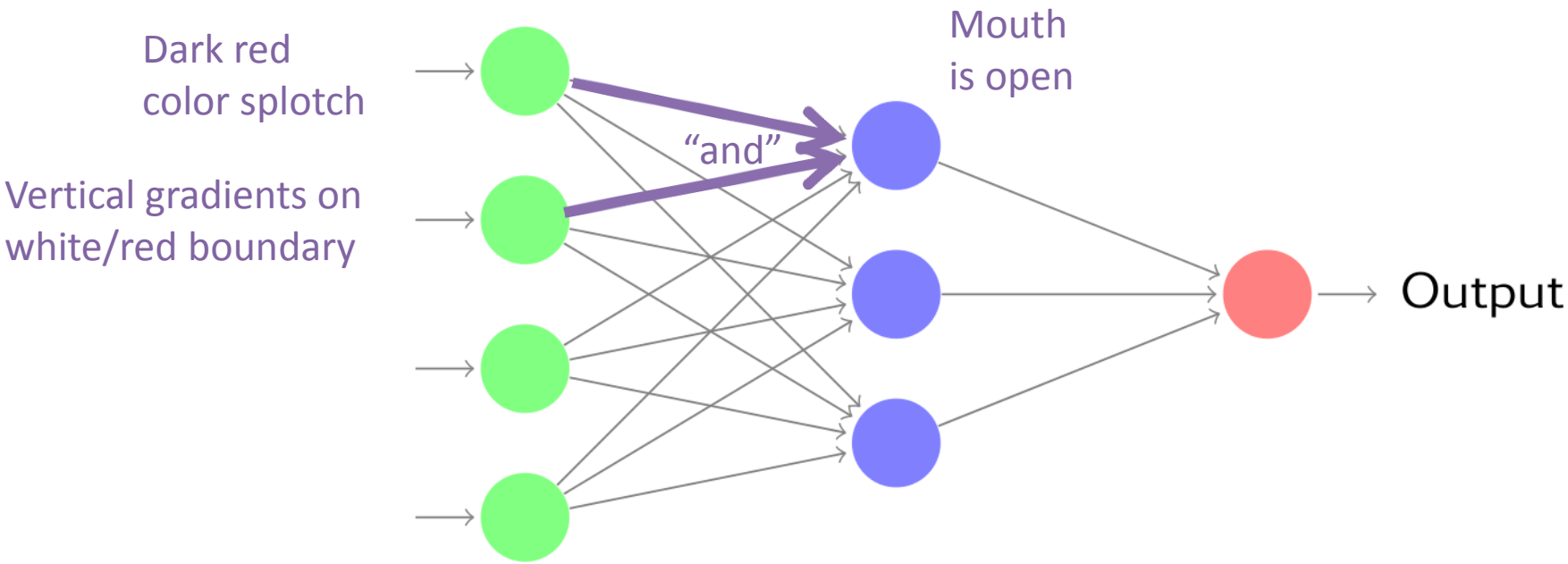
Neural Net



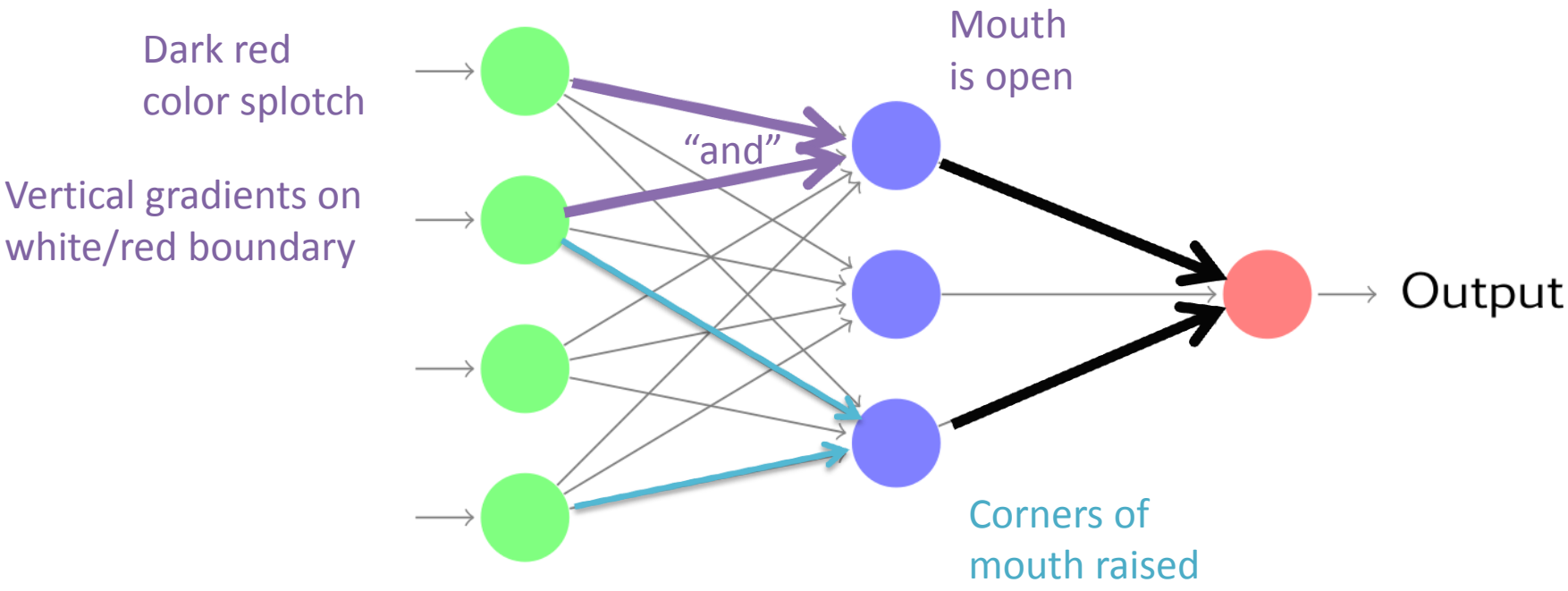
Neural Net can Implement Logic



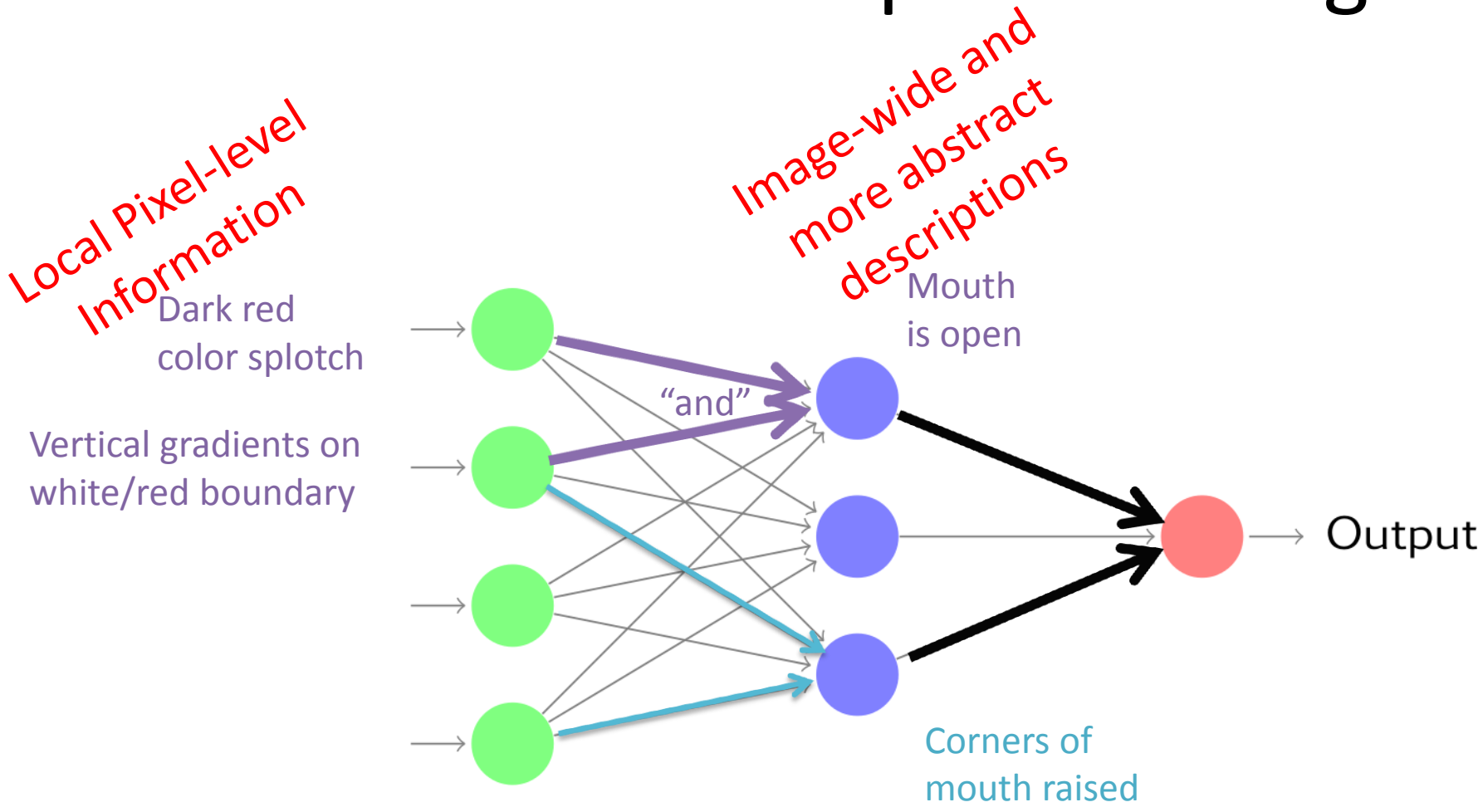
Neural Net can Implement Logic



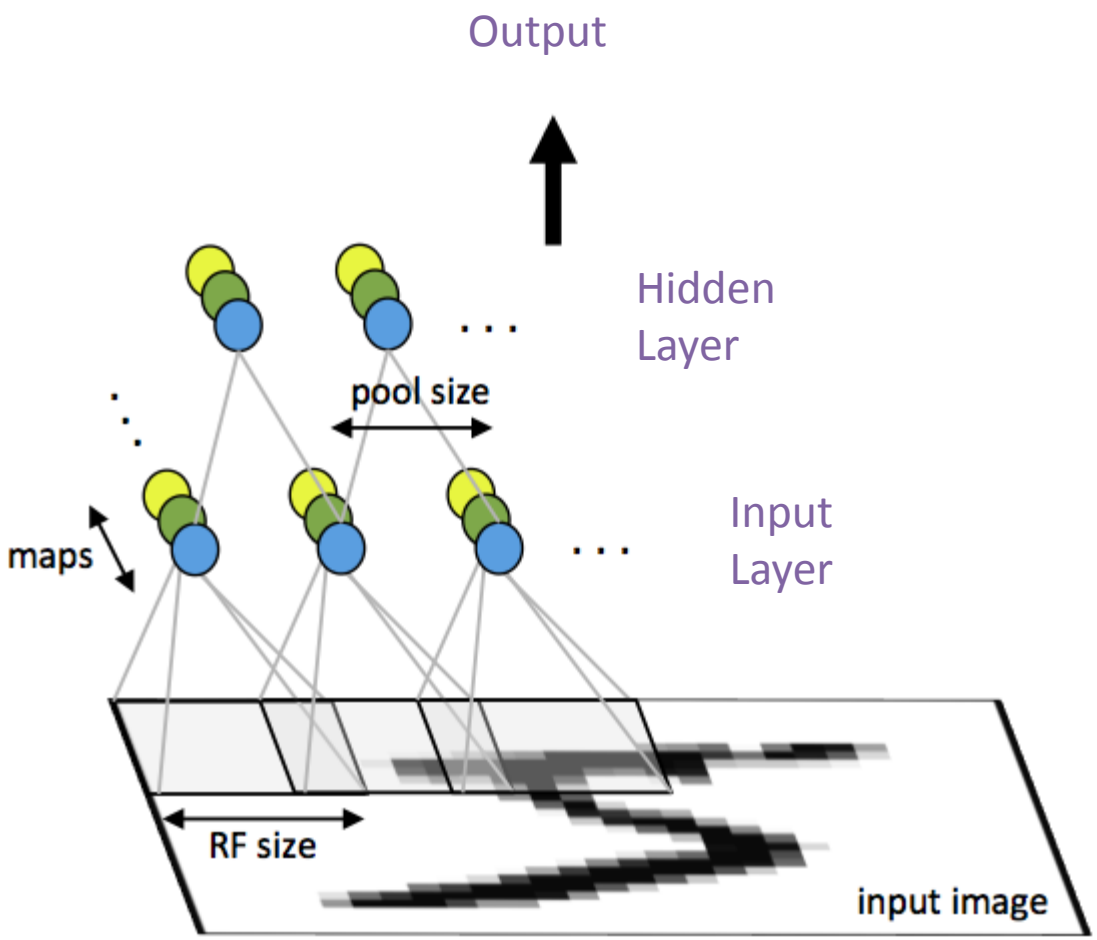
Neural Net can Implement Logic



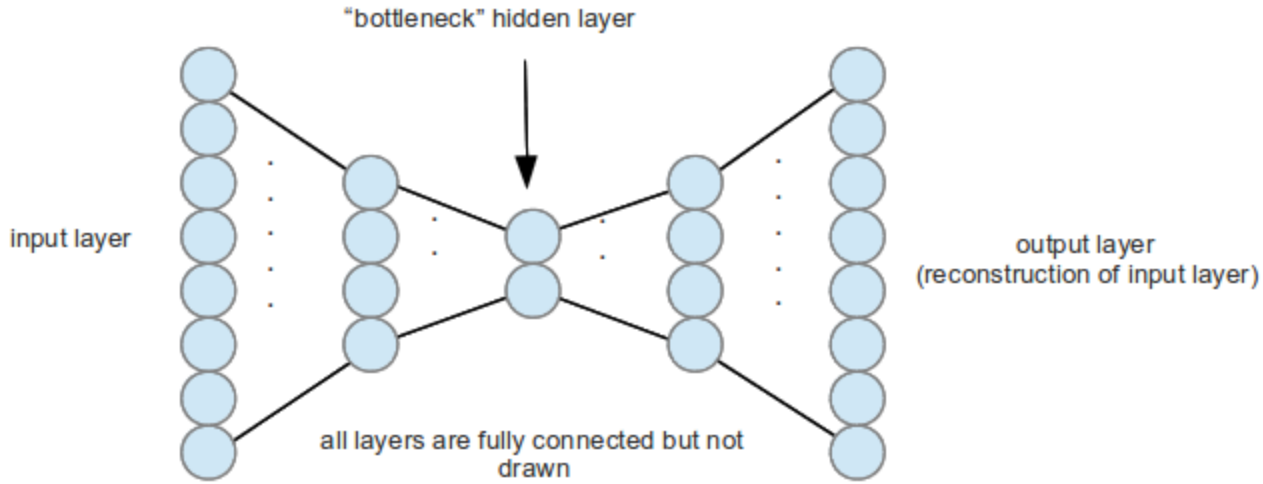
Neural Net can Implement Logic



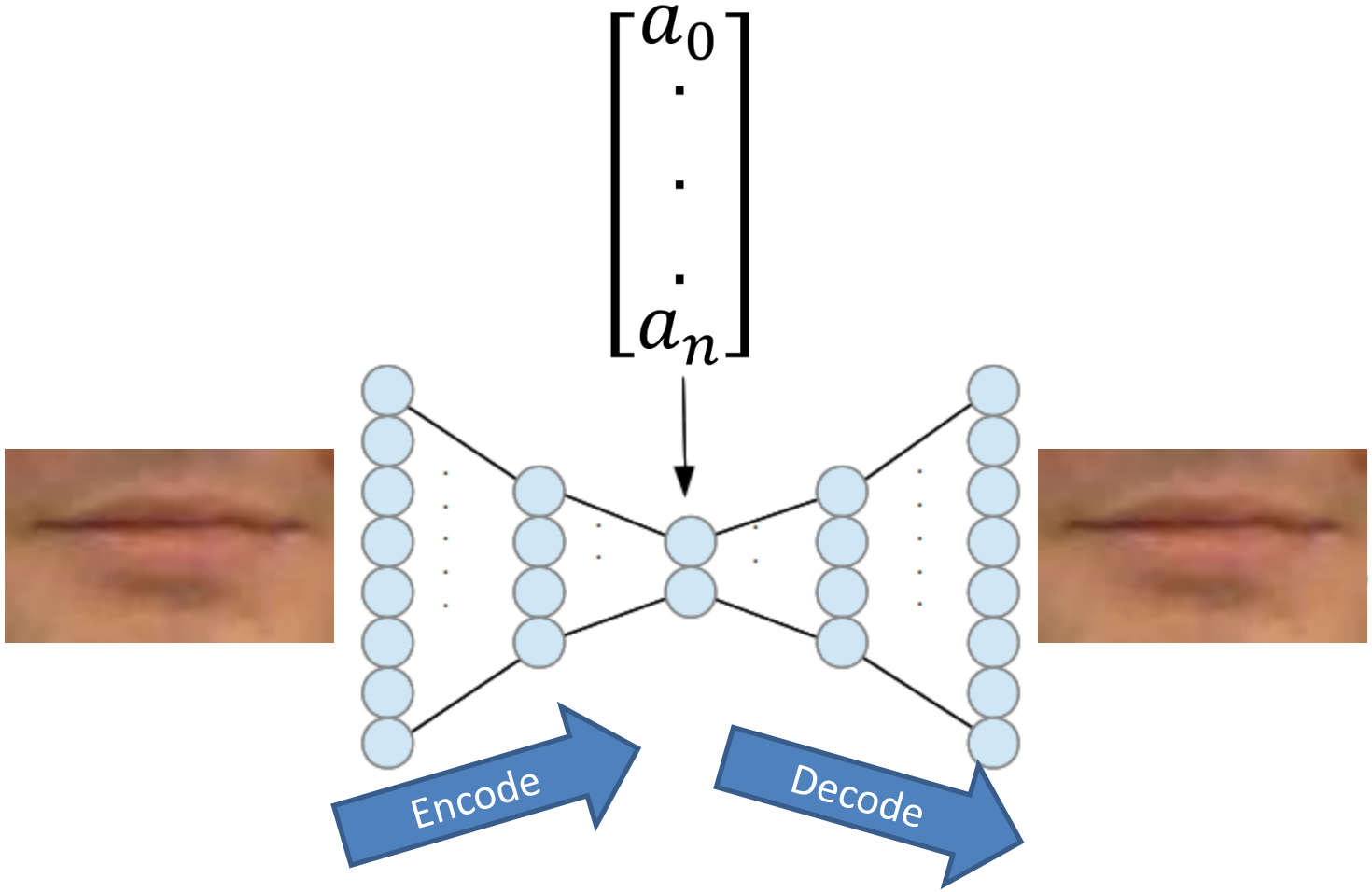
Variational Convolution Neural Net Autoencoder



Variational Convolution Neural Net Autoencoder

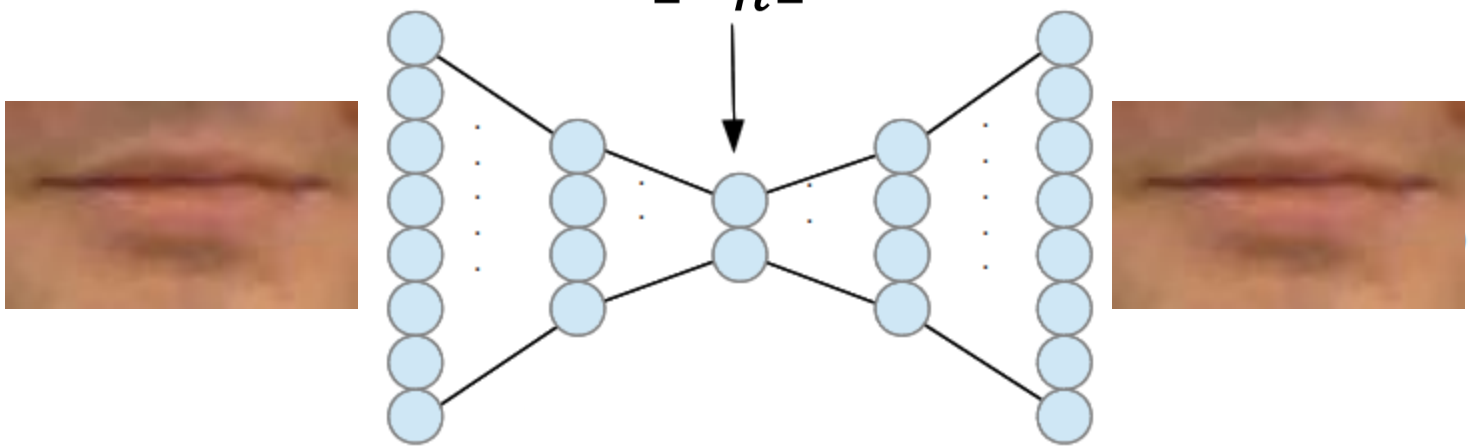


Variational Convolution Neural Net Autoencoder

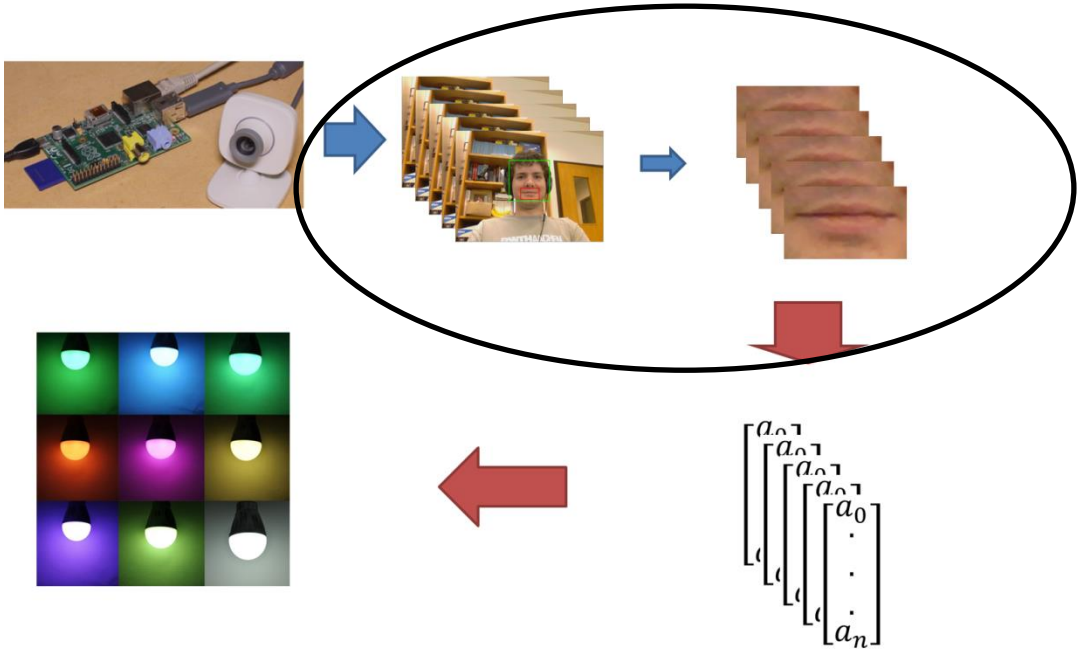


Variational Convolution Neural Net Autoencoder

$$\begin{bmatrix} a_0 \\ \cdot \\ \cdot \\ \cdot \\ a_n \end{bmatrix} \sim N(0,1)$$

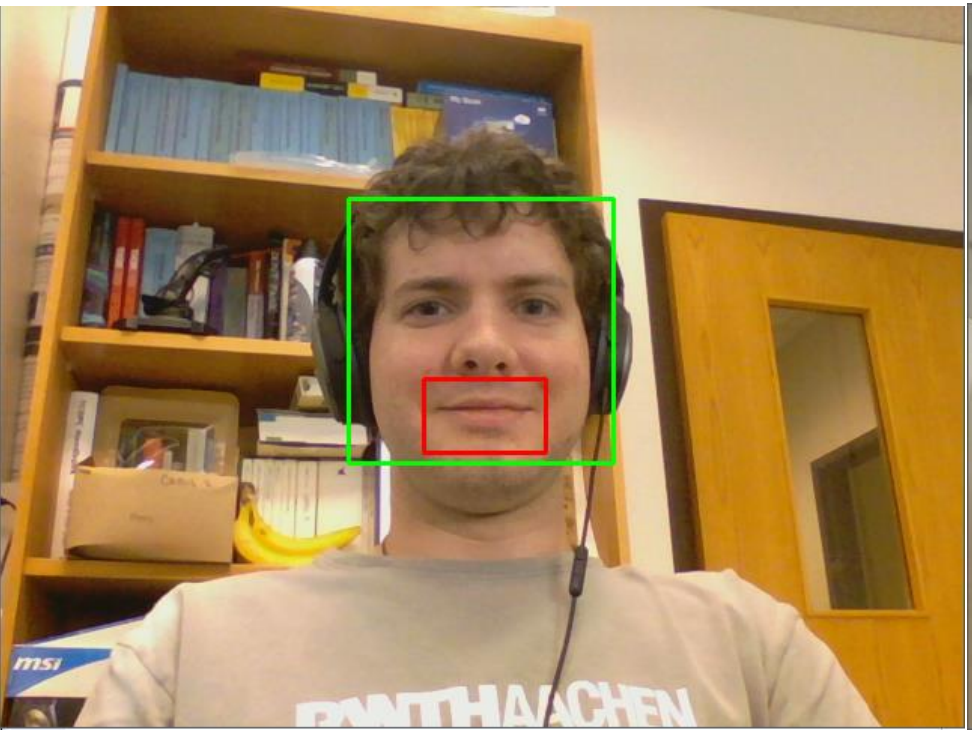


Step 1: Detect and Crop

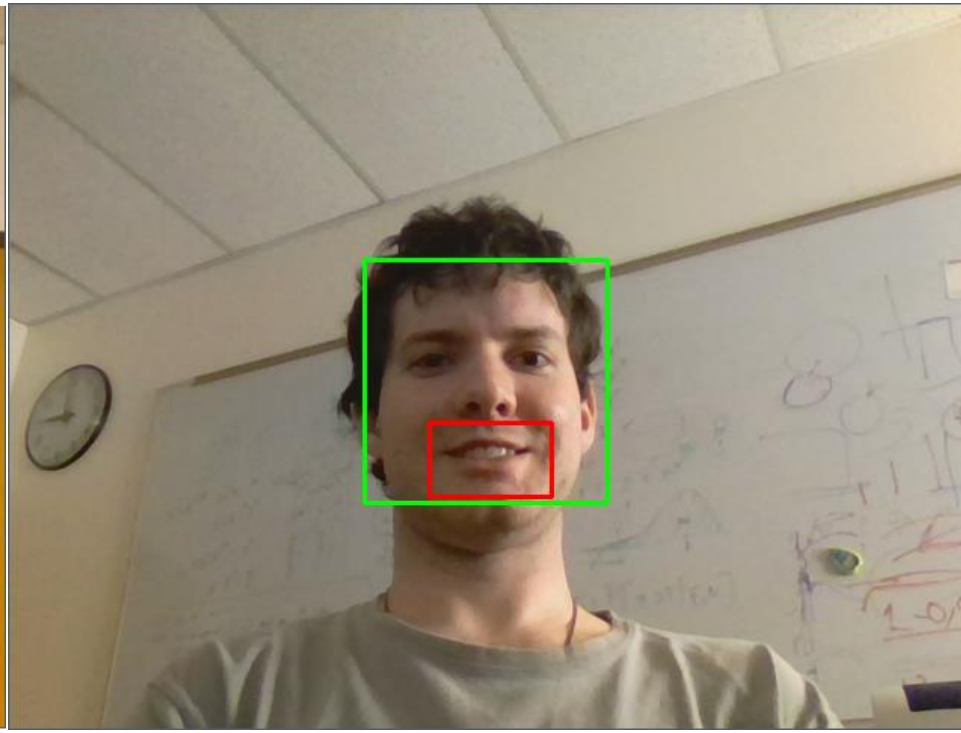


Step 1: Detect and Crop

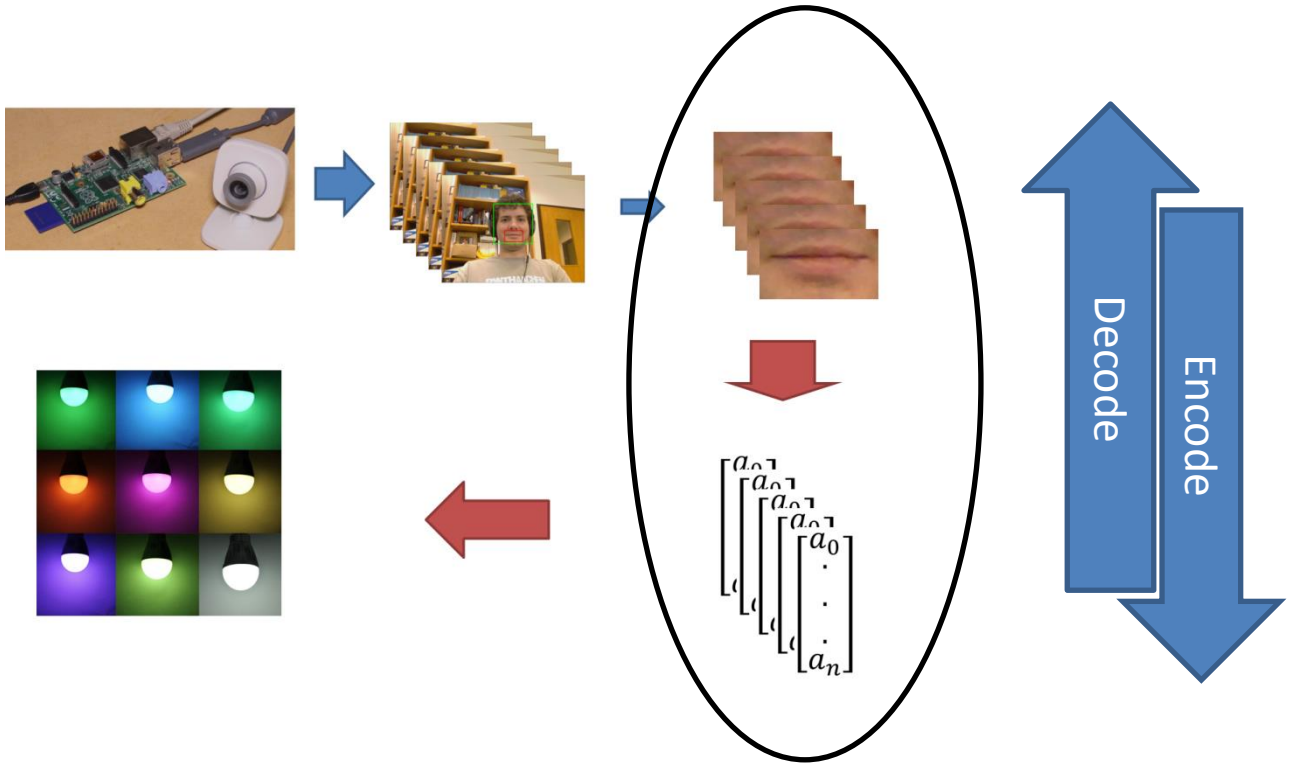
Even Lighting
(easy)



Uneven Lighting
(difficult)



Step 2: Encode

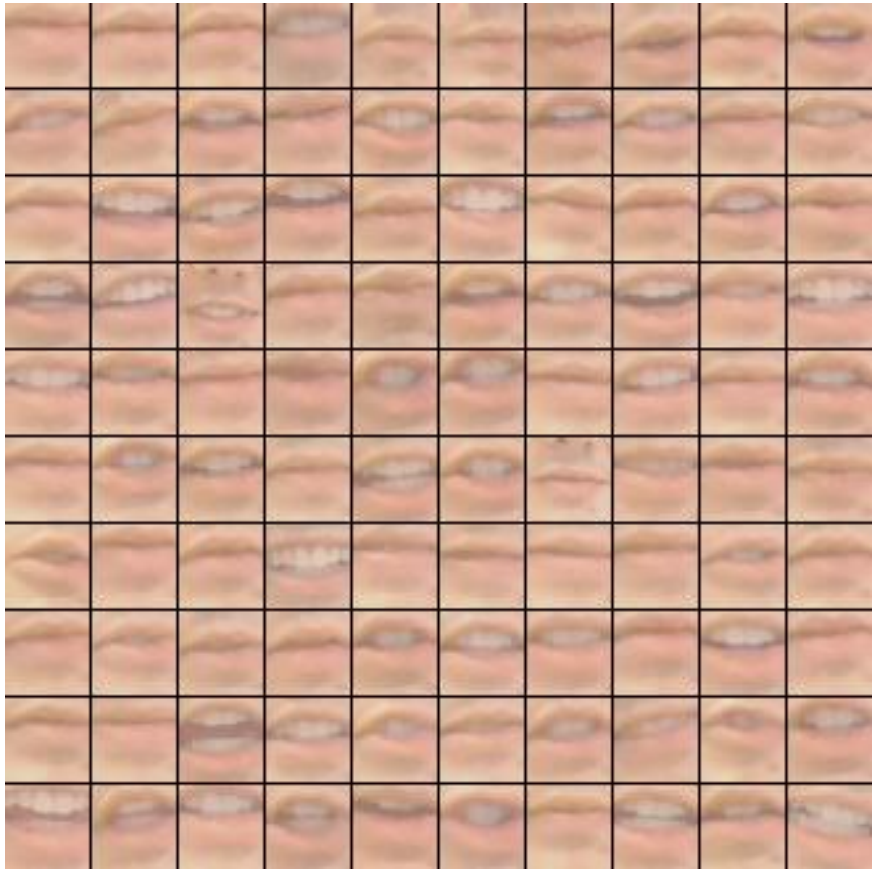


Training
(Variational Autoencoder)

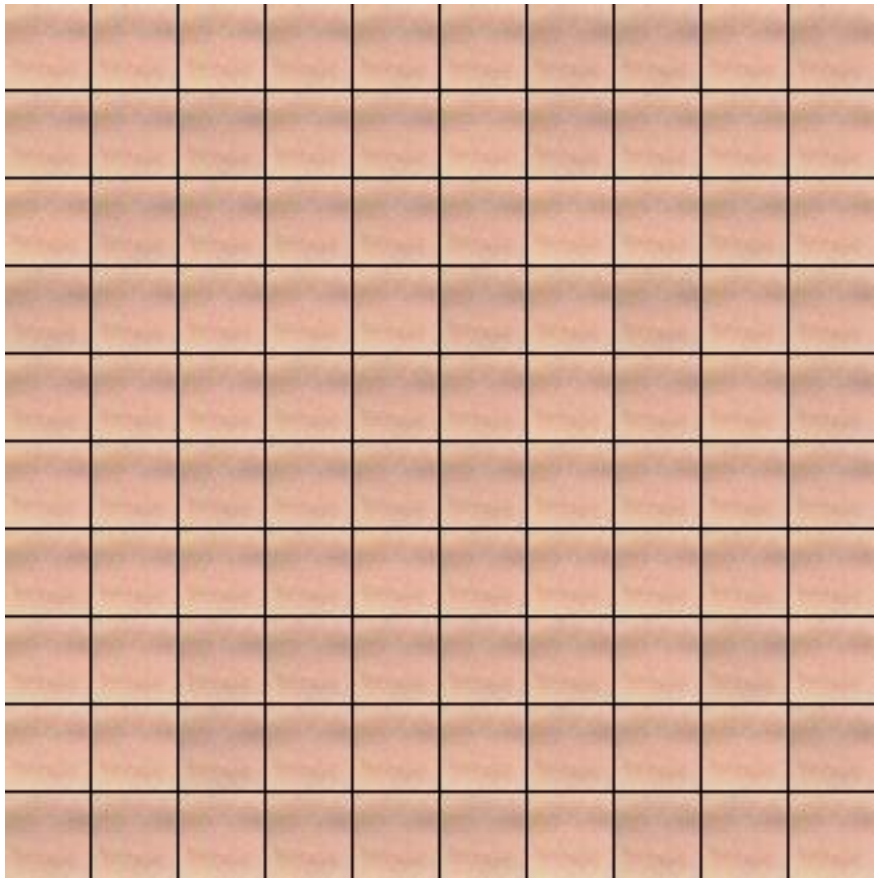
- 20,000 images (32x32x3) taken during Skype conversations
- 11,966,848 parameter model
  - 500x as many parameters as independent inputs!
  - Only 20mil pixels in training set!
- Projected 7 months for model to converge on modest laptop cpu
- ~14 hours on TACC gpu

Training

Original



"Reconstructed"



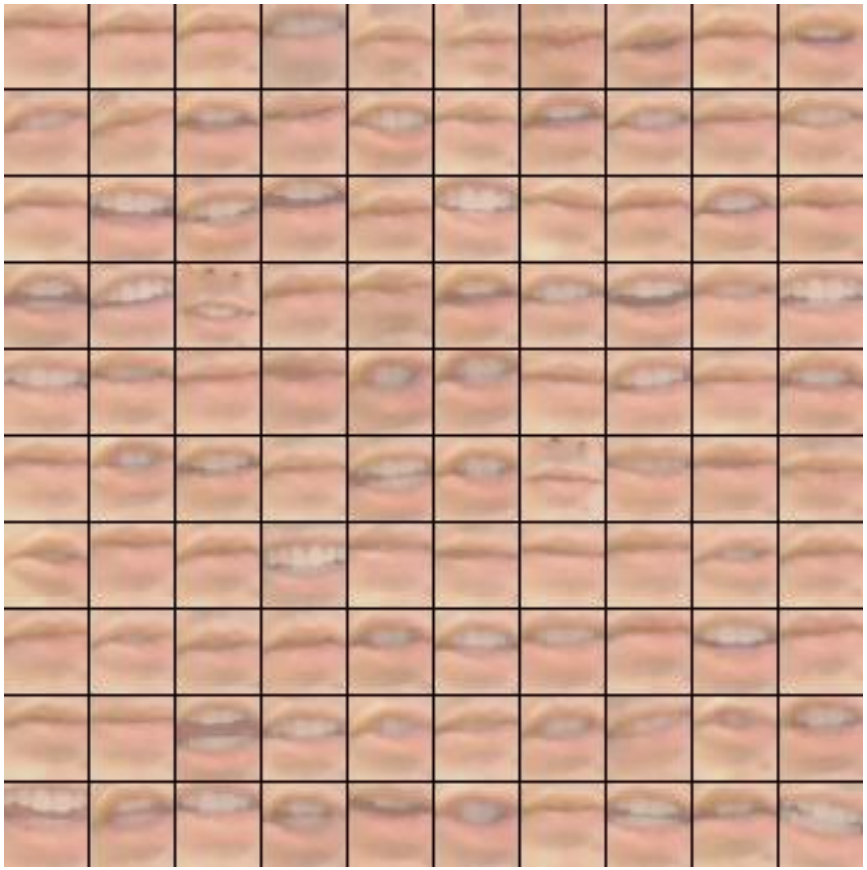
Encode

$$\begin{bmatrix} a_0 \\ \cdot \\ \cdot \\ \cdot \\ a_n \end{bmatrix}$$

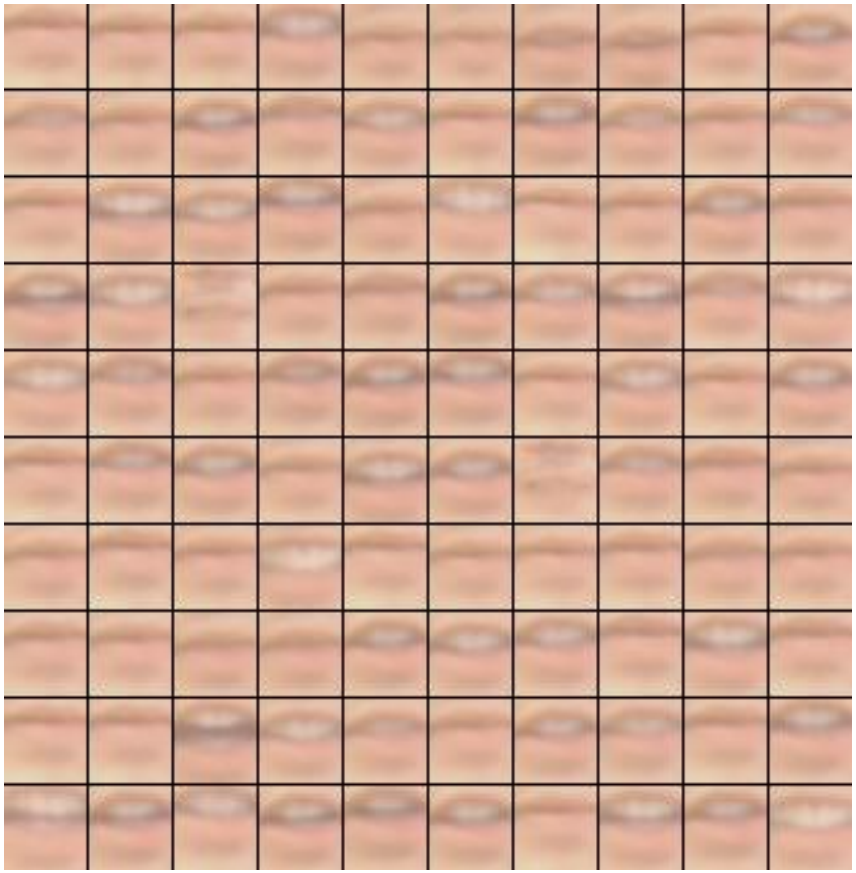
Decode

Training: 300epoch (6mil iter) later…

Original



"Reconstructed"



Encode

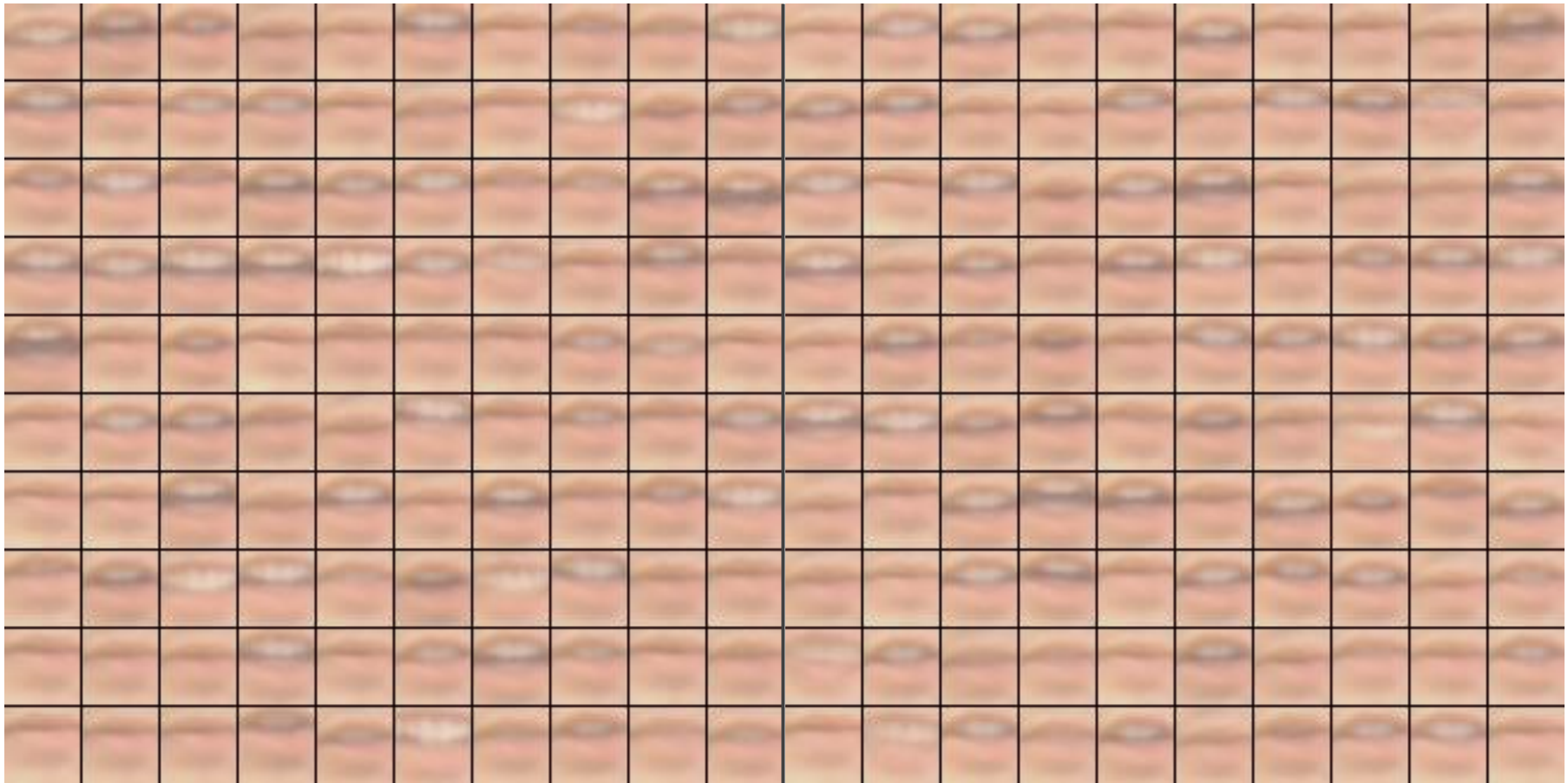
$$\begin{bmatrix} a_0 \\ \cdot \\ \cdot \\ \cdot \\ a_n \end{bmatrix}$$

Decode

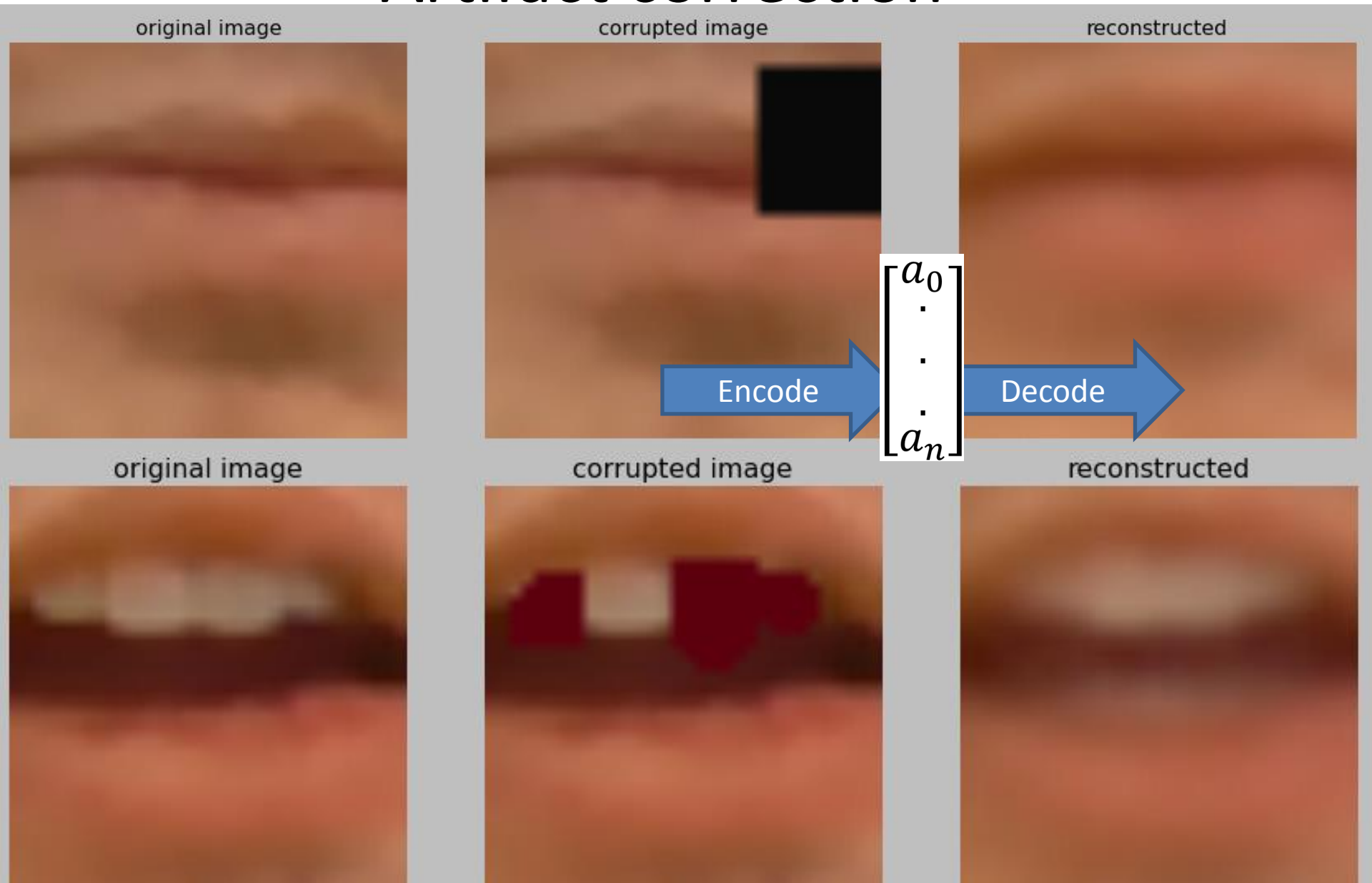
“Variational” Convolution Neural Net Autoencoder:

- Allows for generation of new images, which is nice

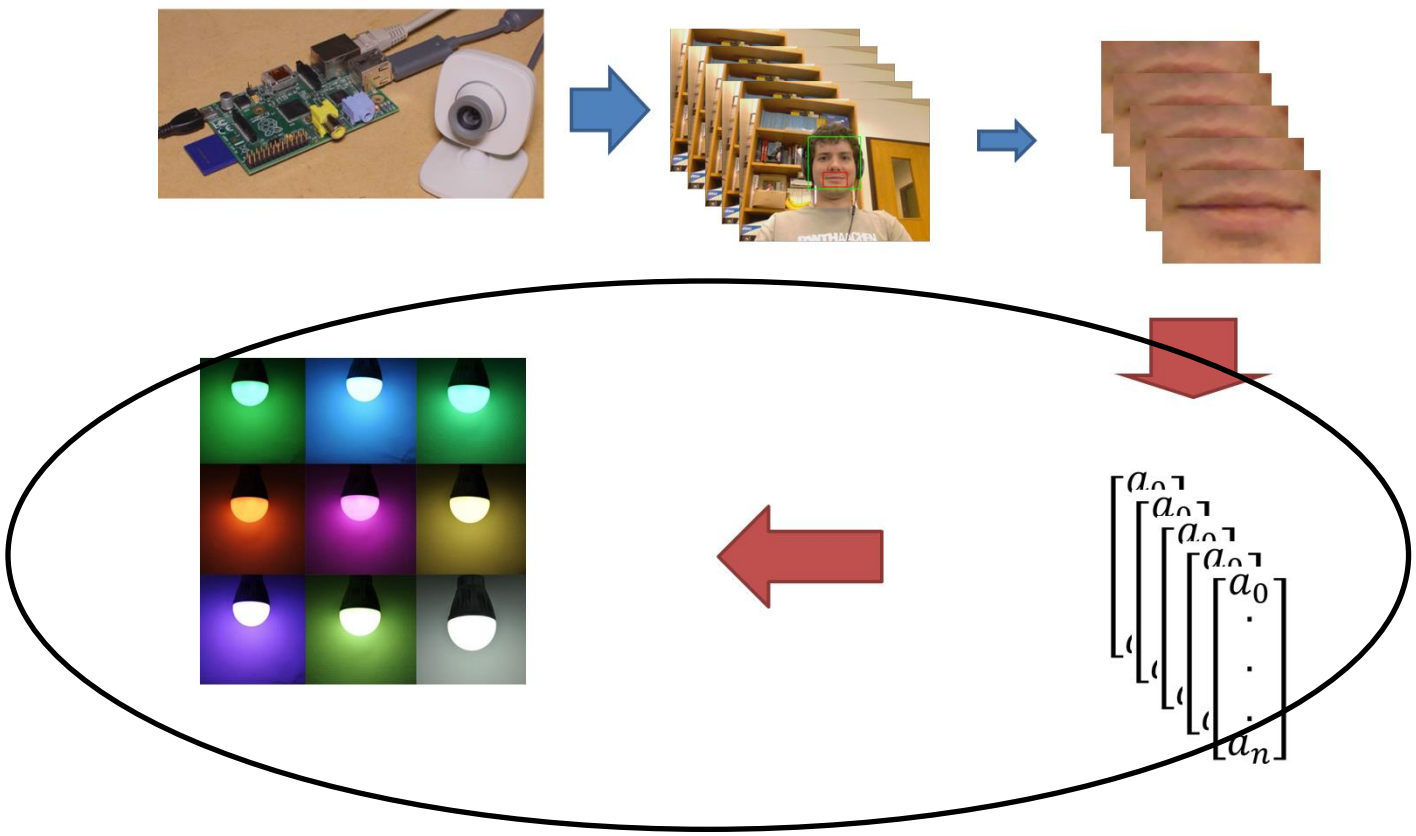
Algorithm Imagined or “Hallucinated” images



Just for fun: Image Inpainting and Artifact correction

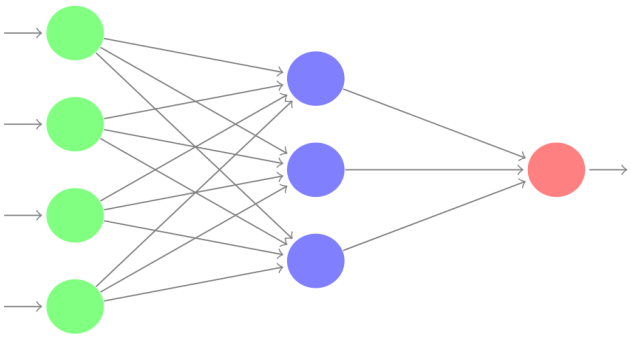


Step 3: Predict Color with vector sequence



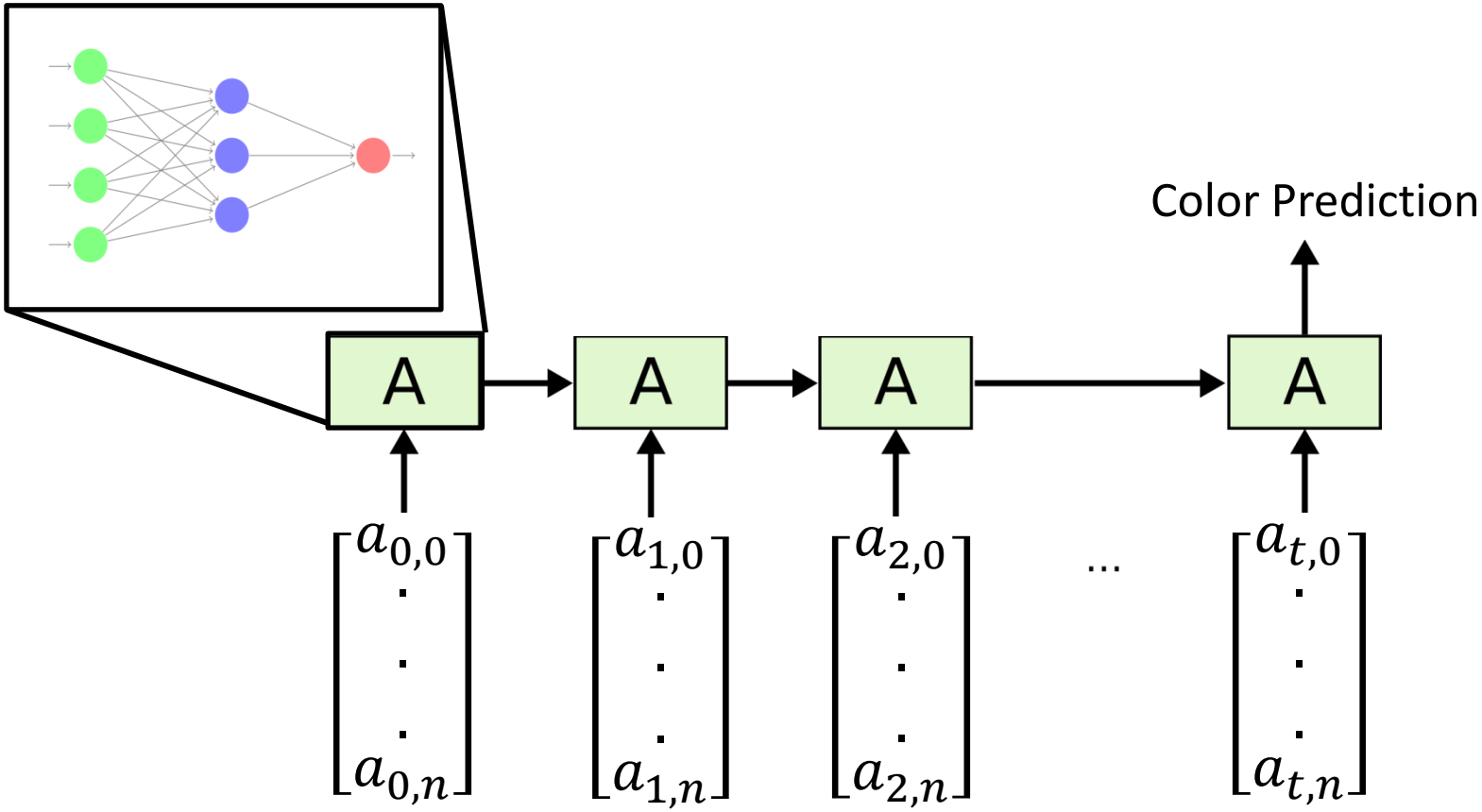
Recurrent Network

Let “ A ” Represent a single layer neural network



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Network



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Network

- Model motivation: pass on a “memory” vector of output of previous frames to influence how the next frame is interpreted

- Color outputs { red, purple, yellow }
  - Purple is similar to two syllables of red
  - Yellow could potentially be predicted by a single wide-mouth frame alone

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Network: training

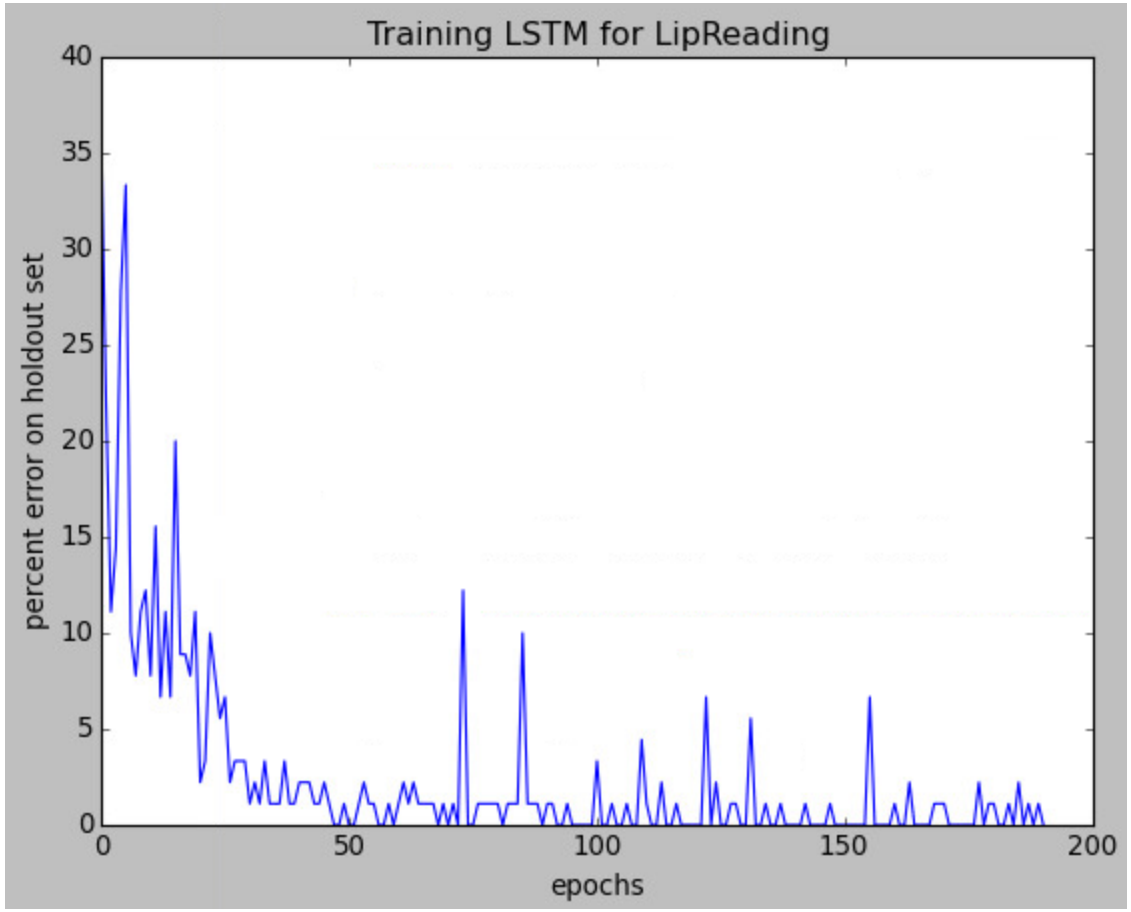
- ~300 training examples of each color
  - 903 total
  - 16-25 time points each example (*variable!*)
    - 512 dim vector “code” for each frame
- A modest 2,099,200 variables
- Training takes ~1hr (gpu)

- Randomly choose 90%/10% split :

  803 for training : 90 for testing

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

A Surprise! A Perfect Classifier



- Quickly achieve 0% error on both training and test set (all colors)
- Rare feat in computer vision
- Probably would not be flawless with 10x more examples