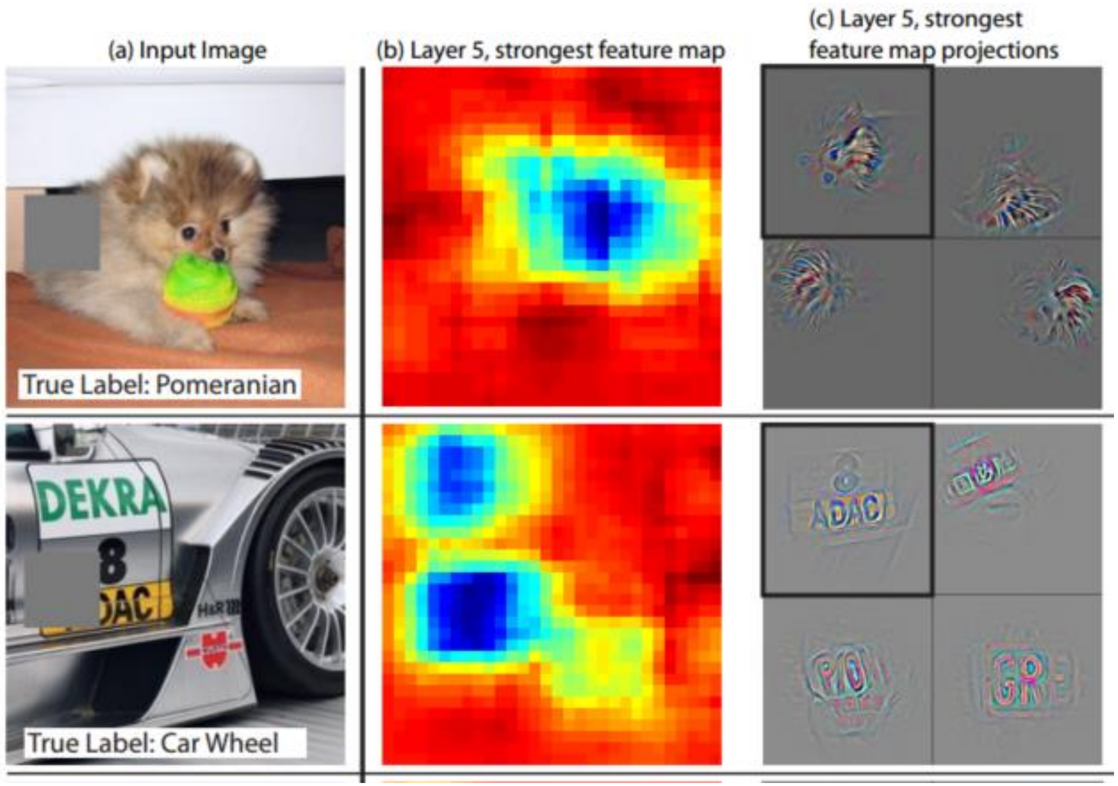
How Neural Networks See (Part 1)



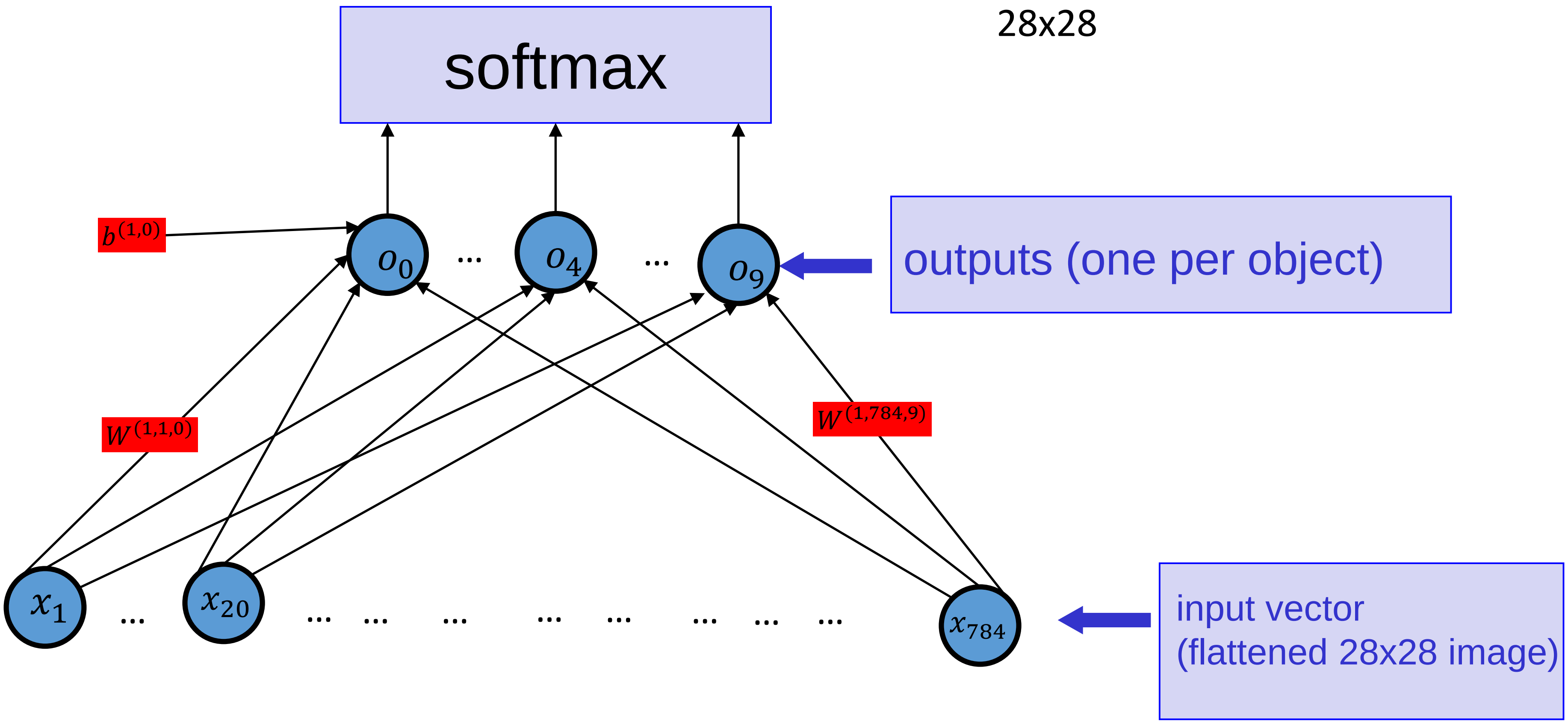
Matthew Zeiler and Rob Fergus, "Visualizing and Understanding Convolutional Networks" (ECCV 2014)

CSC411/2515: Machine Learning and Data Mining, Winter 2018

Michael Guerzhoy and Lisa Zhang

Two-Layer Neural Networks for Image Classification

10 objects, all resized to 28x28



(a.k.a. Multinomial Logistic Regression)

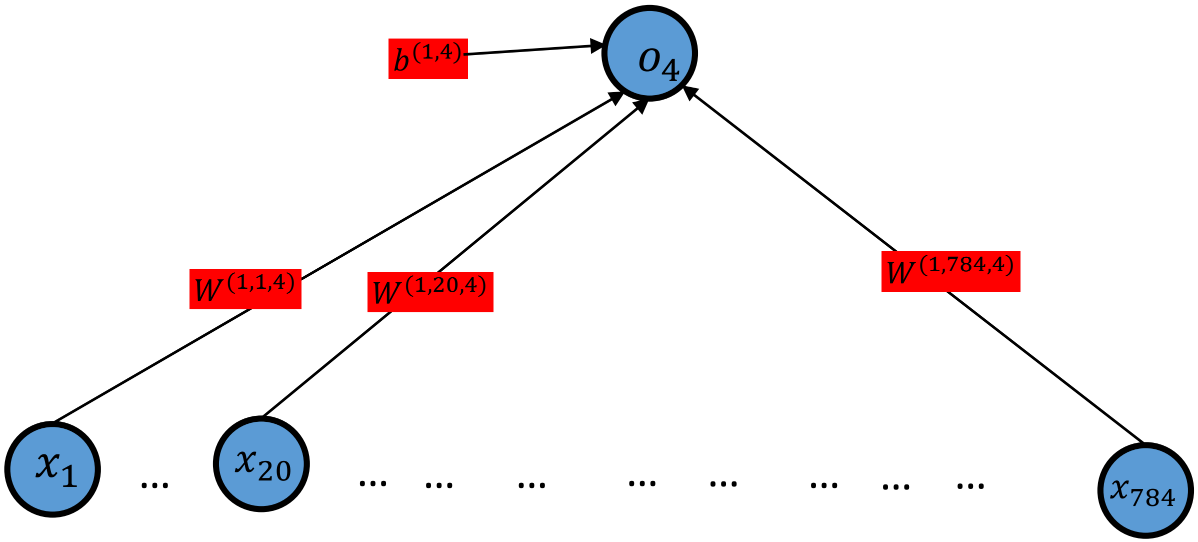
Reminder: Optimizing Neural Networks

- Use Backpropagation to compute the gradient of the cost function (e.g., the –log prob. of the answer) w.r.t. the W's and b's for the whole training set, or for a mini-batch of training examples
- Use gradient descent to find the W's and b's that minimize the cost function
- When classifying images, compute the output of the network for
  x=the input image
  and the W's and b's we found minimizing the cost function
- Find which output is the largest, or interpret the outputs of the Softmax as the probability estimates for the different objects

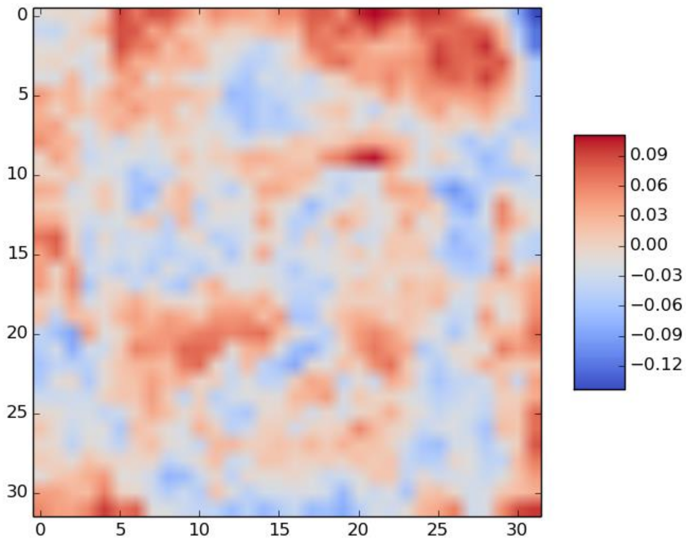
What kind of W's would minimize the cost function?

- E.g., the task is the same as in Project 1: classify an image as one of the 6 actors

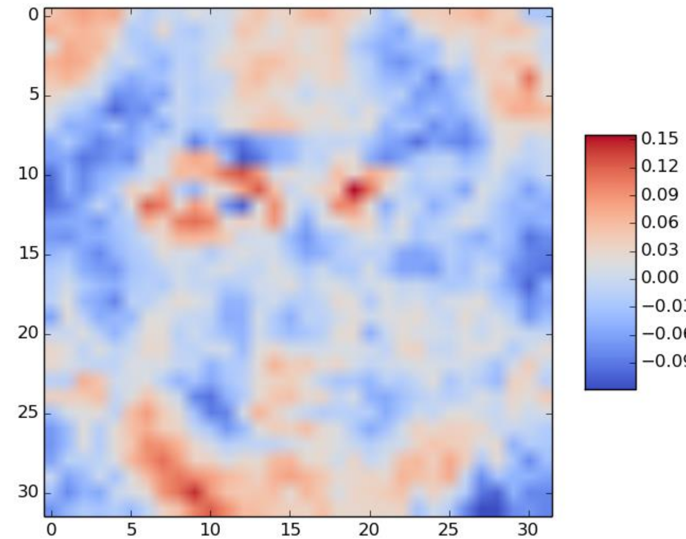
Visualizing the W's



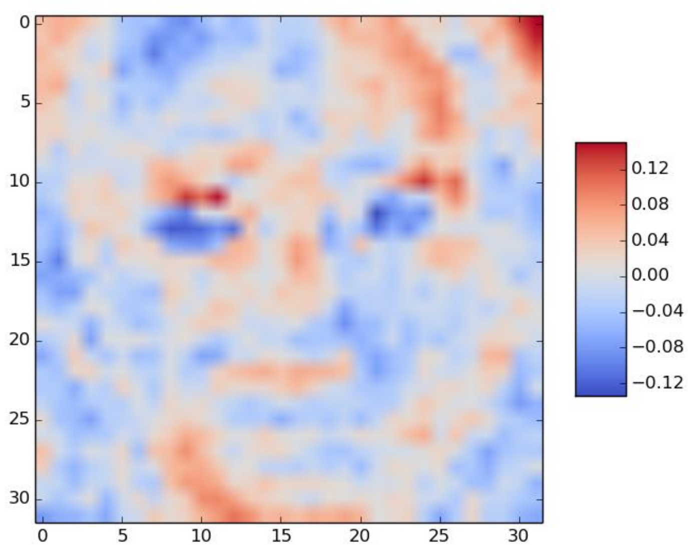
- For a given output unit, we have the strength of the connections from each of the inputs
- To understand what the network is doing, we can think of the $W^{(1,i,4)}$ as an image



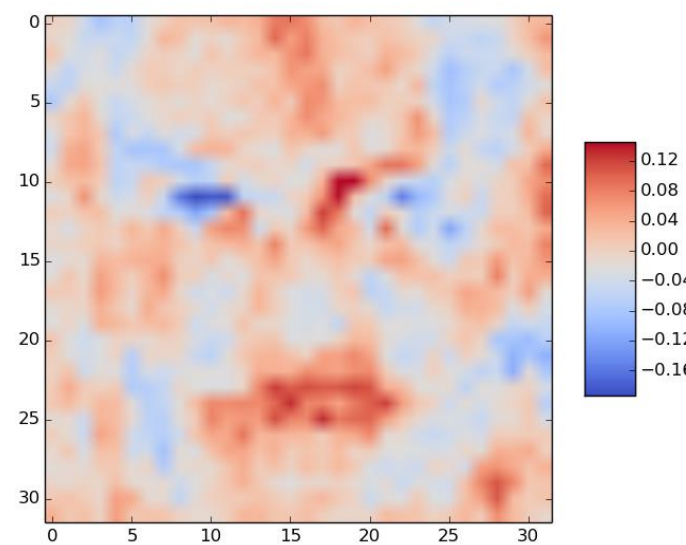
Bracco



Vartan



Radcliffe



Gilpin

Multinomial Logistic Regression with Early Stopping, 40 examples each

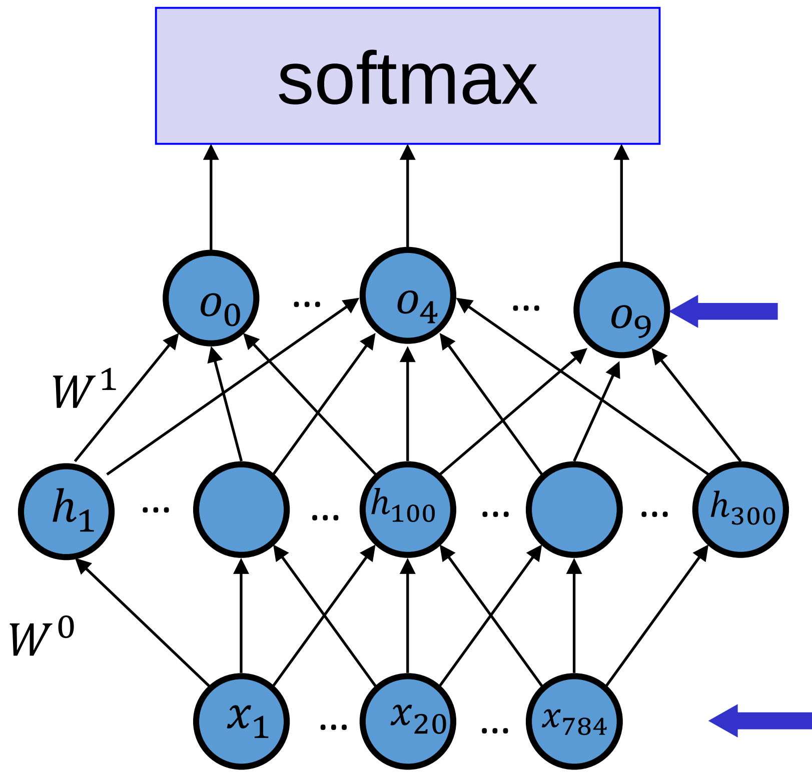
The Dot Product $W^{(1,*,j)} \cdot x$

- Note that the input to the unit o_j is
  $$W^{(1,*,j)} \cdot x + b^{(1,j)}$$
- For a vector x of a given magnitude, $W^{(1,*,j)} \cdot x$ is as large as possible when $x = \alpha W^{(1,*,j)}$
  - I.e., when x and $W^{(1,*,j)}$ point in the same direction
  - The dot product $u \cdot v$ is the length of the projection of u onto v
  - That means that o_j is larger when x looks like $W^{(1,*,j)}$, viewed as images
    - (Note: it also means we should make sure all our input x's are of similar magnitudes)
    - Why?

Aside: all the input x's should have the same magnitude

- If $\mathrm{x}^{(1)} = \alpha x^{(2)}$, they are basically the same image, just with different contrast and maximum brightness
- The output of the neural network for x_1 and x_2 should be the same
- Solution: always *standartize* any input x before putting it in the dataset
  - See optimization slides

Neural Networks with Hidden Layers

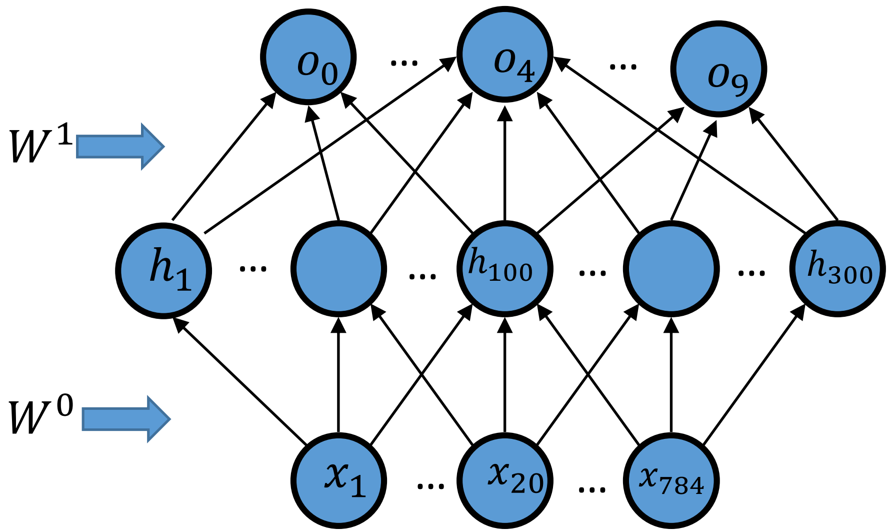


outputs (one per object)

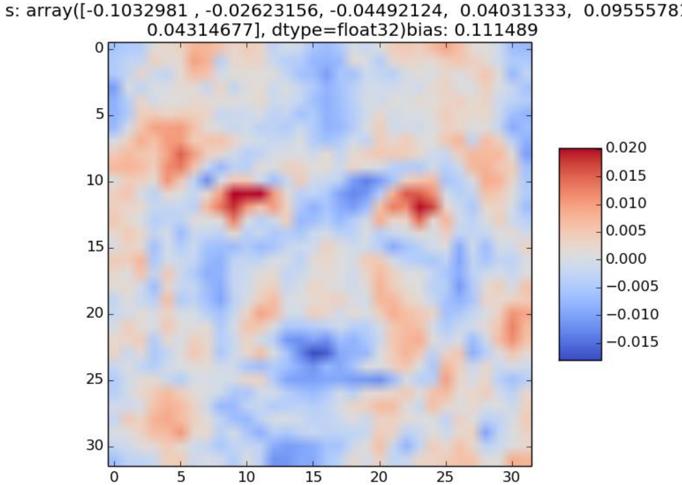
hidden layer (300 hidden units)

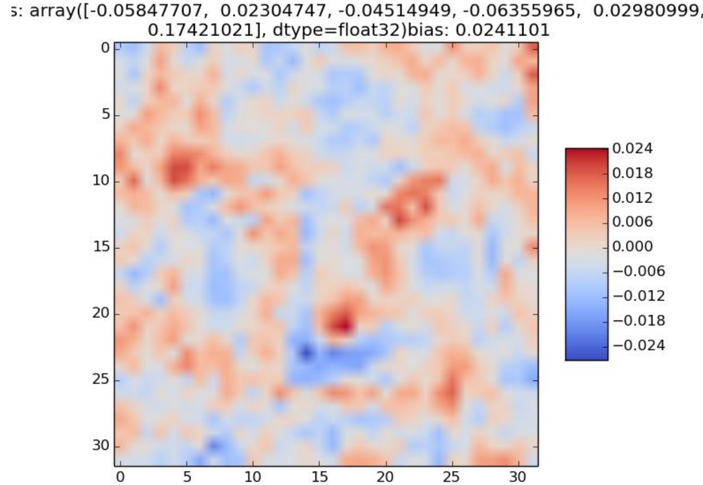
input vector (flattened 28x28 image)

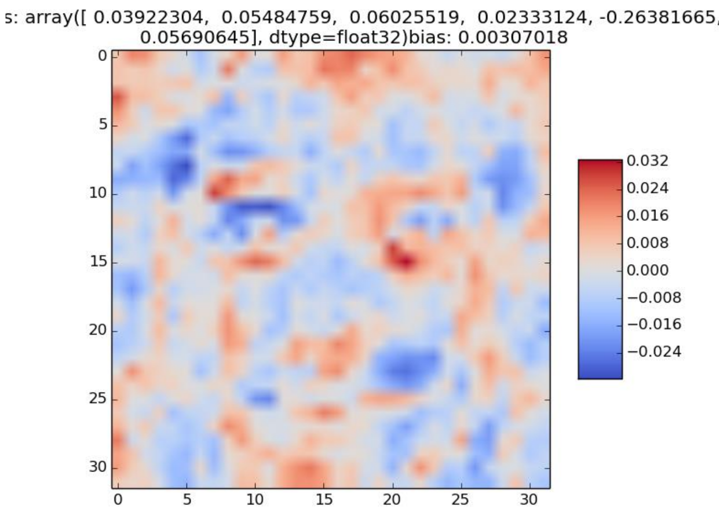
Understanding Hidden Layers

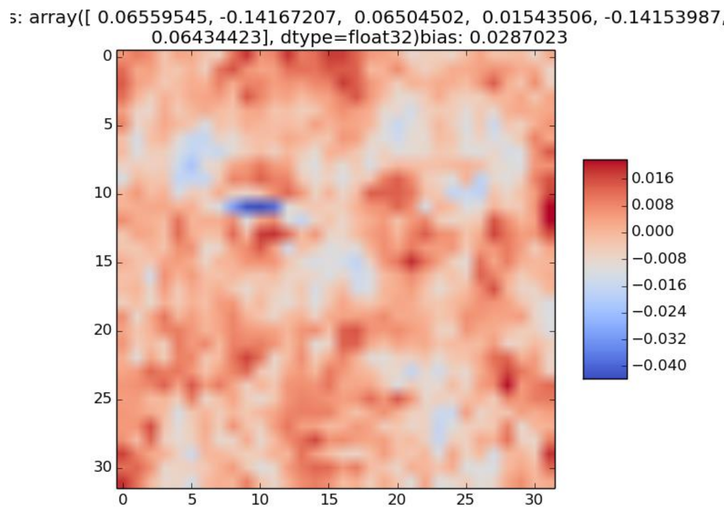


- Can visualize W^0 like before
- But what does it mean for the input to e.g. h_5 to be high?
  - Depends on how h_5 is connected to the output layer!



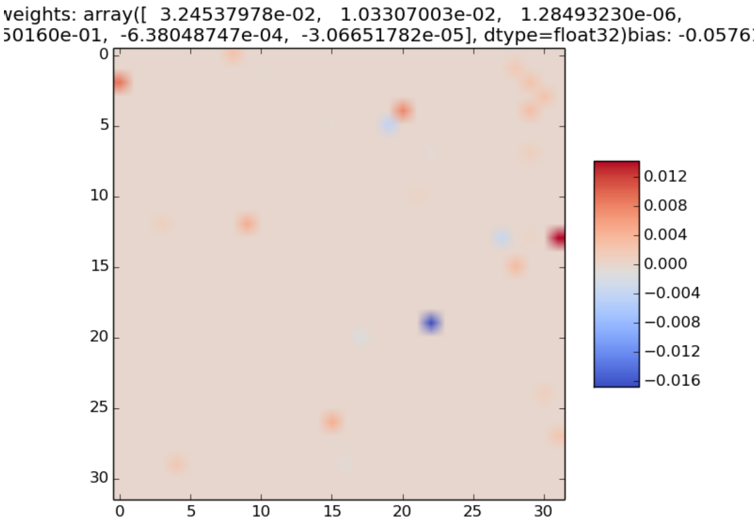


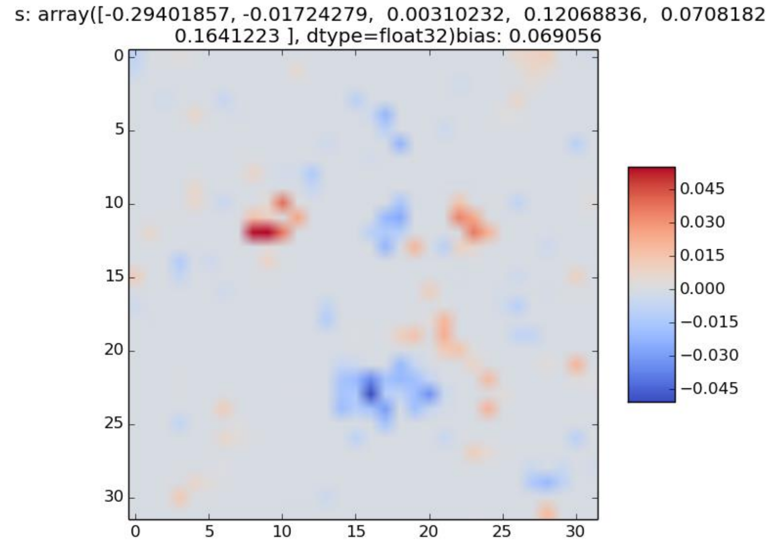


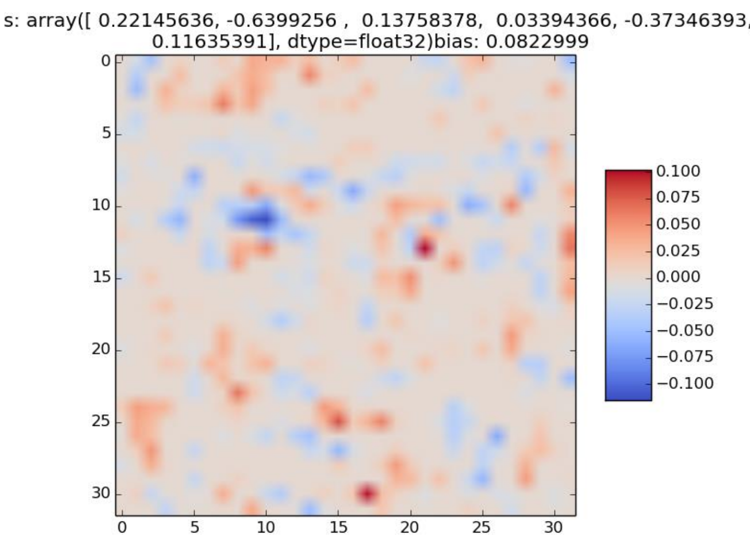


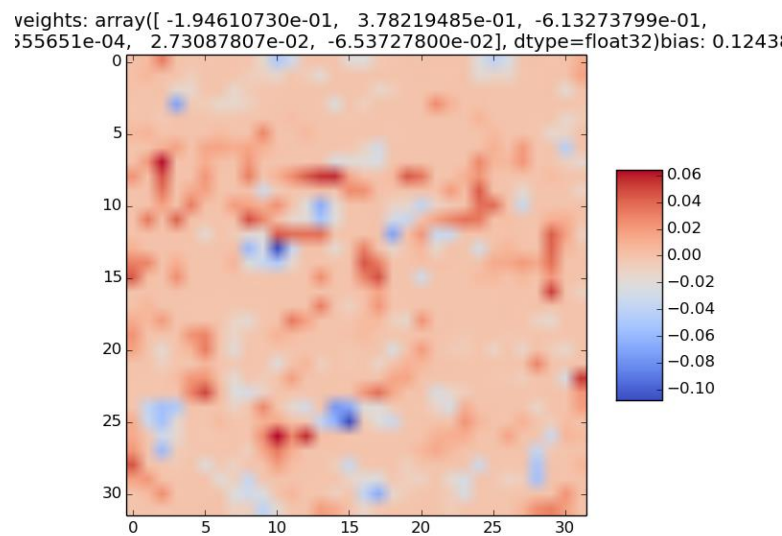
act = ['Angie Harmon‘, 'Peri Gilpin‘, 'Lorraine Bracco', 'Michael Vartan', 'Daniel Radcliffe', 'Gerard Butler']

300 hidden units, 6 actors, 40 examples each, L2-penalized

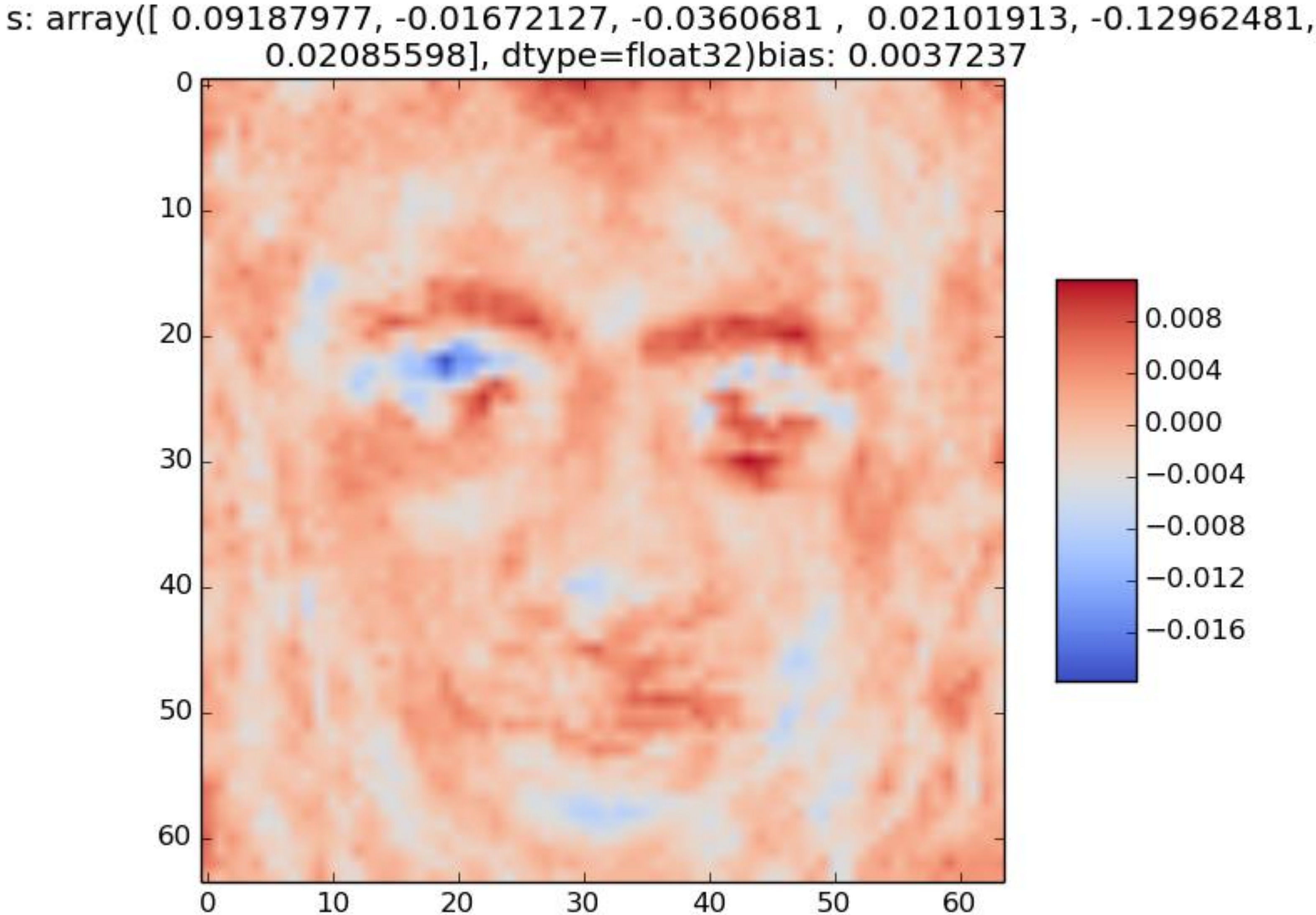




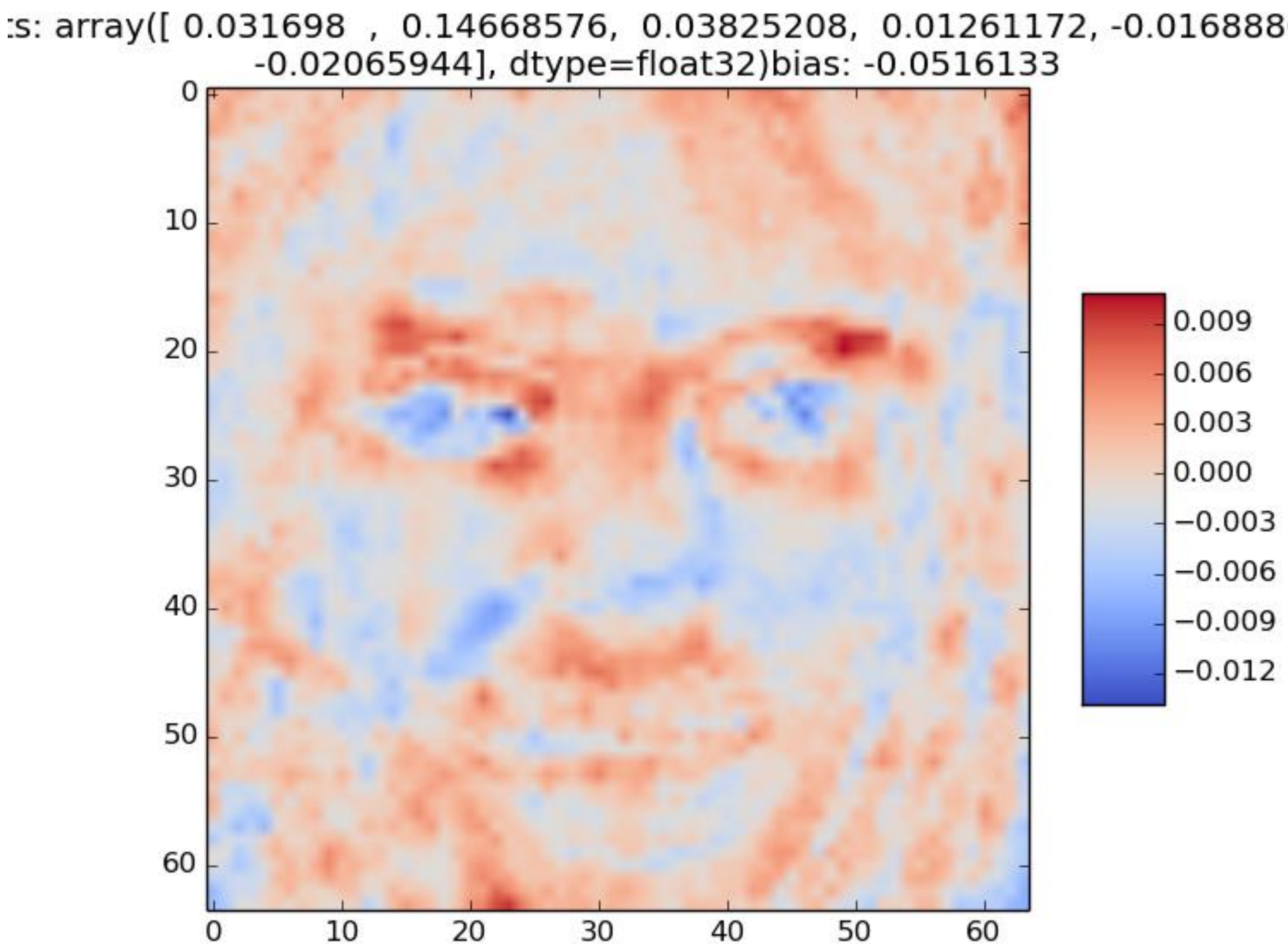




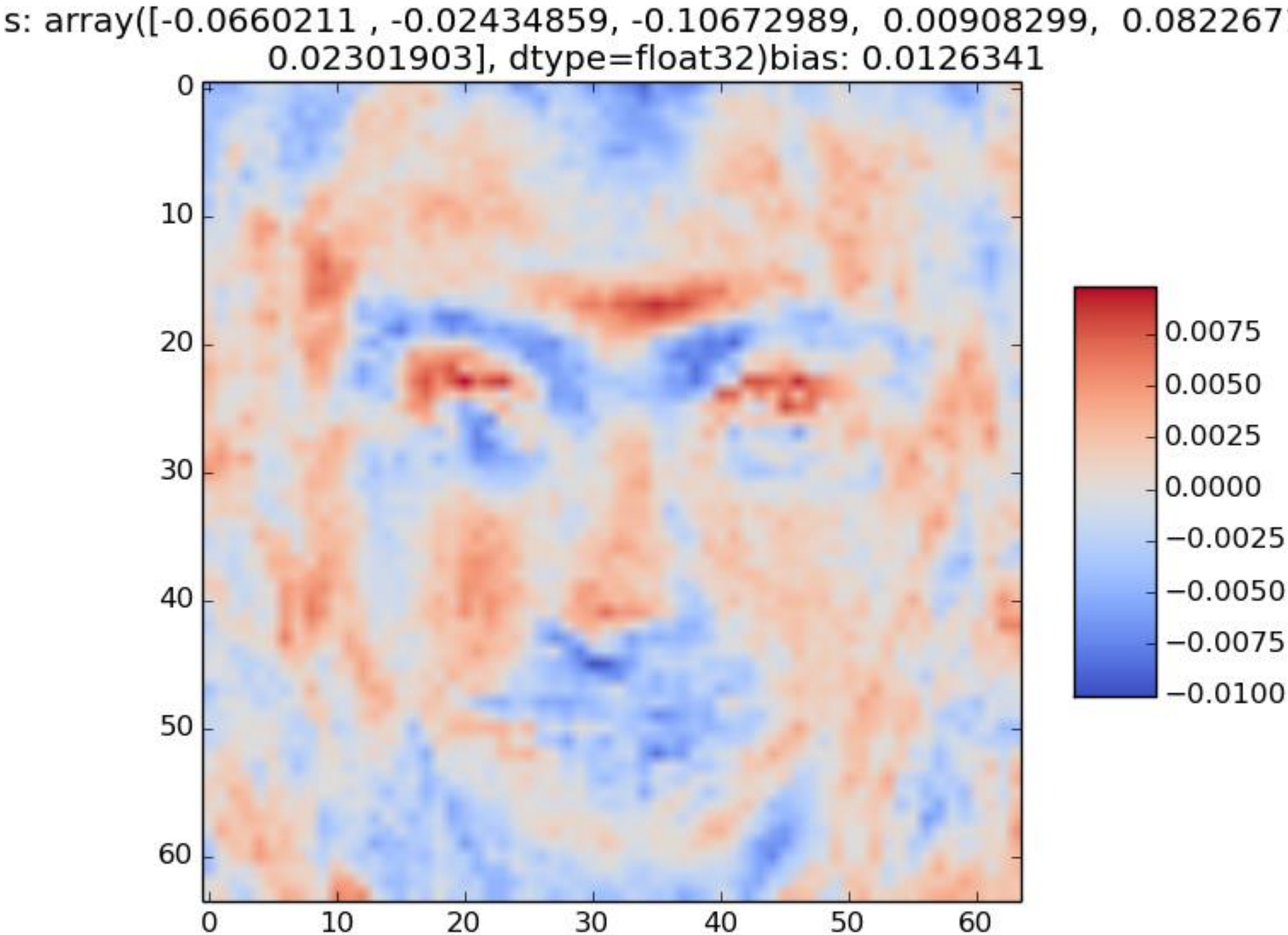
300 hidden units, 6 actors, 40 examples each, L1-penalized



300 hidden units, 6 actors, 40 examples each, L2-penalized, 128x128 images



300 hidden units, 6 actors, 40 examples each, L2-penalized, 128x128 images



300 hidden units, 6 actors, 40 examples each, L2-penalized, 128x128 images

Hidden Layer Units as Features

- Once we train the neural network, the values units in the hidden layer should be useful for computing the output units.
- The weights W^0 between the input layer and the hidden layer are such that the hidden units are useful
- Think of the hidden units as “features” of the data – summaries of the data that are useful for computing the outputs
- In networks with no hidden layer, we simply compute as many features as there are outputs
    - So the “features” should look like the inputs that we are looking for
- (Recall the XOR example: we computed the feature “x1>.5” and the feature “x2>.5” using hidden units)

Overfitting with a hidden layer

300 units + heavy-duty optimization