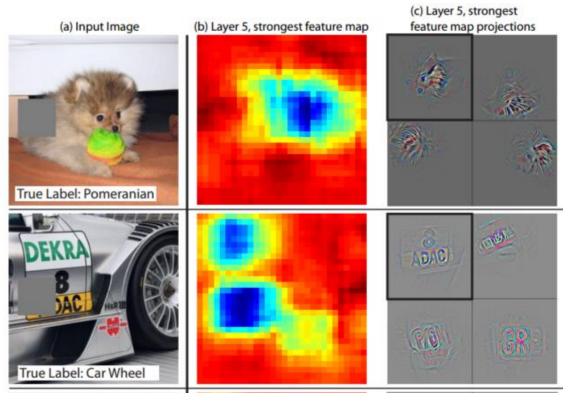
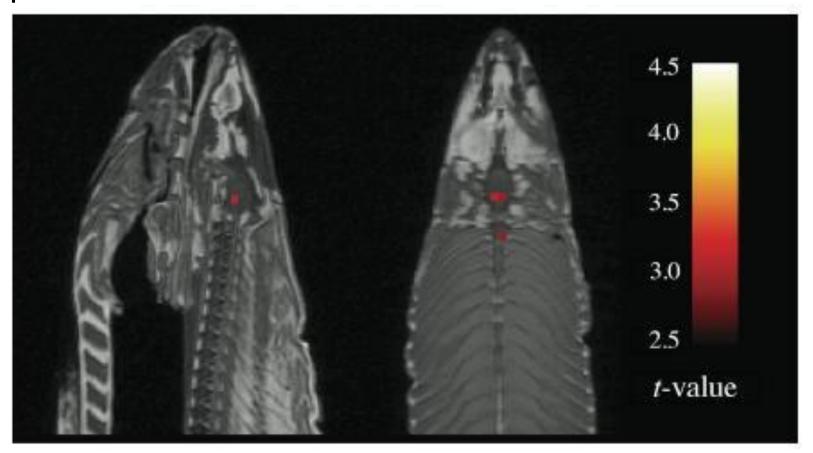
#### How Neural Networks See (Part 1)



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

CSC411: Machine Learning and Data Mining, Winter 2017

A Diversion: Neural correlates of interspecies perspective taking in the post-mortem Atlantic Salmon



#### METHODS

<u>Subject.</u> One mature Atlantic Salmon (Salmo salar) participated in the fMRI study. The salmon was approximately 18 inches long, weighed 3.8 lbs, and was not alive at the time of scanning.

<u>Task</u>. The task administered to the salmon involved completing an open-ended mentalizing task. The salmon was shown a series of photographs depicting human individuals in social situations with a specified emotional valence. The salmon was asked to determine what emotion the individual in the photo must have been experiencing.

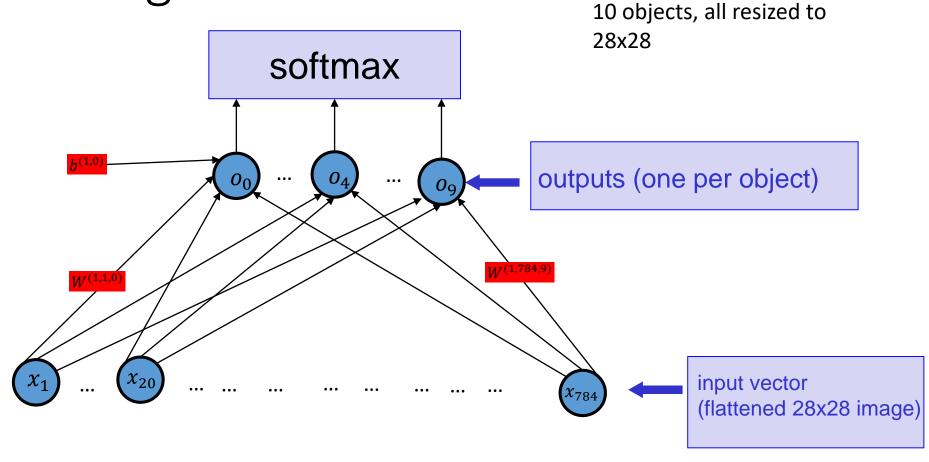
<u>Design.</u> Stimuli were presented in a block design with each photo presented for 10 seconds followed by 12 seconds of rest. A total of 15 photos were displayed. Total scan time was 5.5 minutes.

<u>Preprocessing.</u> Image processing was completed using SPM2. Preprocessing steps for the functional imaging data included a 6-parameter rigid-body affine realignment of the fMRI timeseries, coregistration of the data to a T<sub>1</sub>-weighted anatomical image, and 8 mm full-width at half-maximum (FWHM) Gaussian smoothing.

<u>Analysis</u>. Voxelwise statistics on the salmon data were calculated through an ordinary least-squares estimation of the general linear model (GLM). Predictors of the hemodynamic response were modeled by a boxcar function convolved with a canonical hemodynamic response. A temporal high pass filter of 128 seconds was include to account for low frequency drift. No autocorrelation correction was applied.

<u>Voxel Selection.</u> Two methods were used for the correction of multiple comparisons in the fMRI results. The first method controlled the overall false discovery rate (FDR) and was based on a method defined by Benjamini and Hochberg (1995). The second method controlled the overall familywise error rate (FWER) through the use of Gaussian random field theory. This was done using algorithms originally devised by Friston et al. (1994).

# Two-Layer Neural Networks for Image Classification



(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

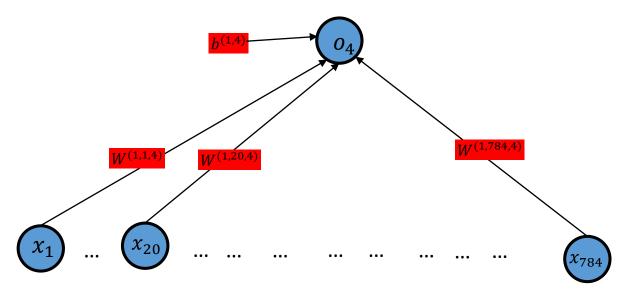
x=the input image and the W's and b's we found minimizing the cost function

 Find which output is the largest, or interpret the outputs of the softmax as the probability estimates for the different objects

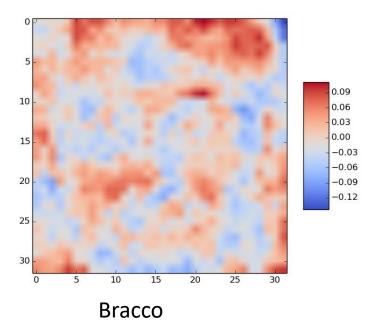
# What kind of W's would minimize the cost function?

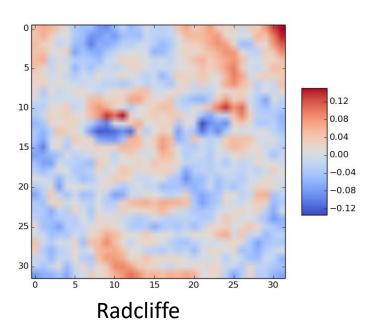
• E.g., the task is the same as in Project 1: classify an image as one of the 6 actors

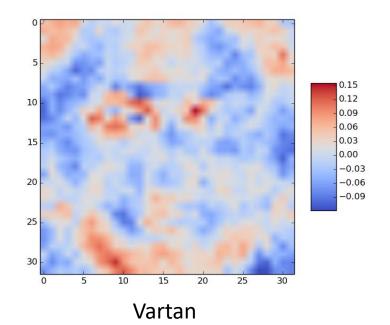
# Visualizing the W's

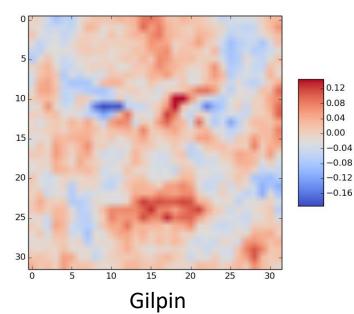


- For a given output unit, we have the strength of the connections from each of the inputs
- To understand what the network is doing, we can think of the  $W^{(1,i,4)}$  as an image









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

• Note that the input to the unit  $o_i$  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
  - (Explanation on the board: 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)

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

- If  $x^{(1)} = \alpha x^{(2)}$ , 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 *normalize* any input x before putting it in the dataset

$$\chi \to \frac{x}{|x|}$$

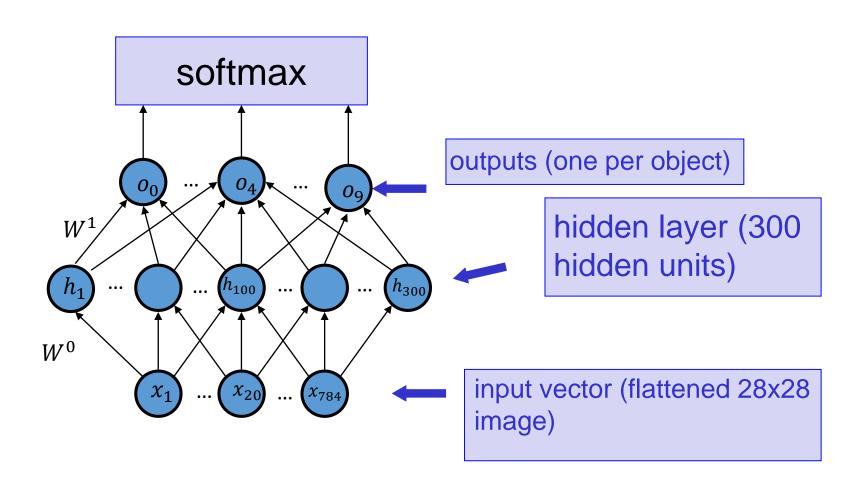
## Aside: Normalizing Data

- Usually, we also want for the mean of all the entries in x to be 0
  - Helps prevent dead neurons (reminder of why on the board)
  - $x \rightarrow (x mean(x))$
- Transformation:

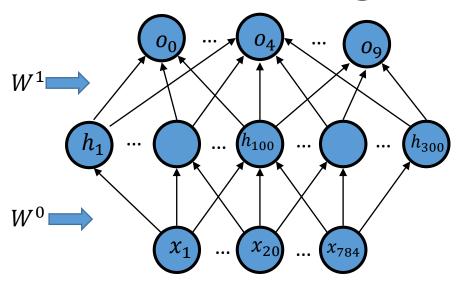
$$\chi \to \frac{x - mean(x)}{sd(x)}$$

• (Note: for mean(x) = 0, sd(x) = |x|/sqrt(dim(<math>x)))

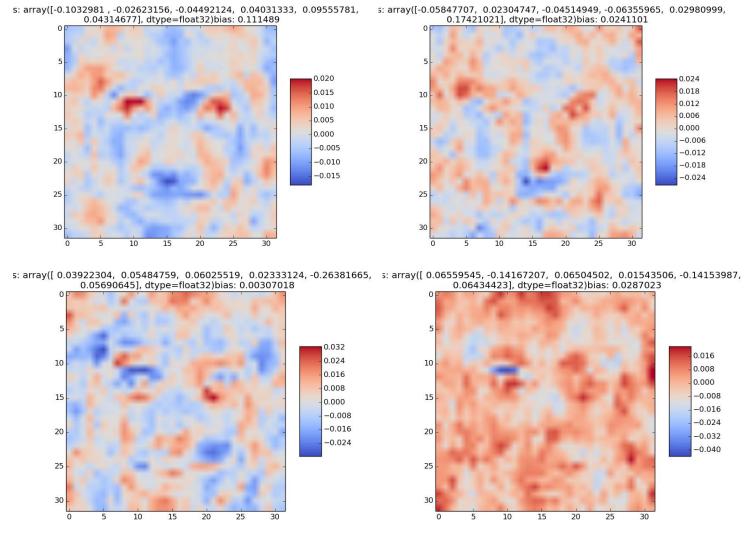
### Neural Networks with Hidden Layers



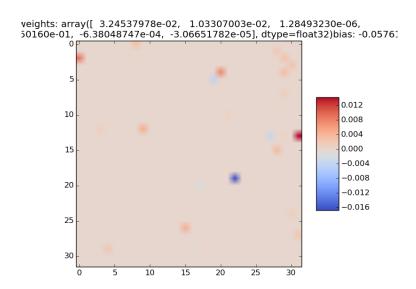
### Understanding Hidden Layers

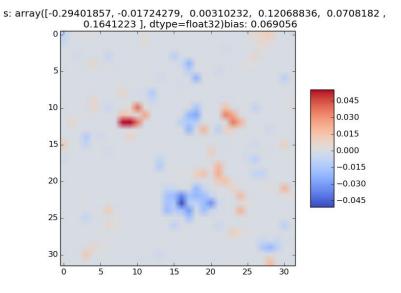


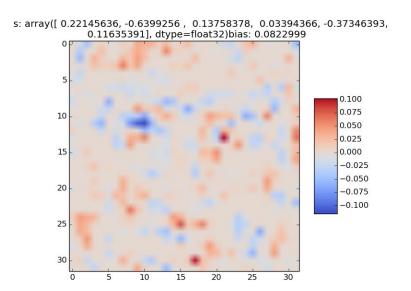
- Can visualize  $W^0$  like before
- But what does it mean for the input to e.g.  $h_5$  to be high?
  - Depends on how  $h_5$  is connected to the output layer!

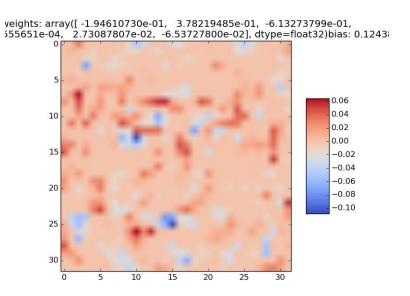


act = ['Angie Harmon', 'Peri Gilpin', 'Lorraine Bracco', 'Michael Vartan', 'Daniel Radcliffe', 'Gerard Butler']

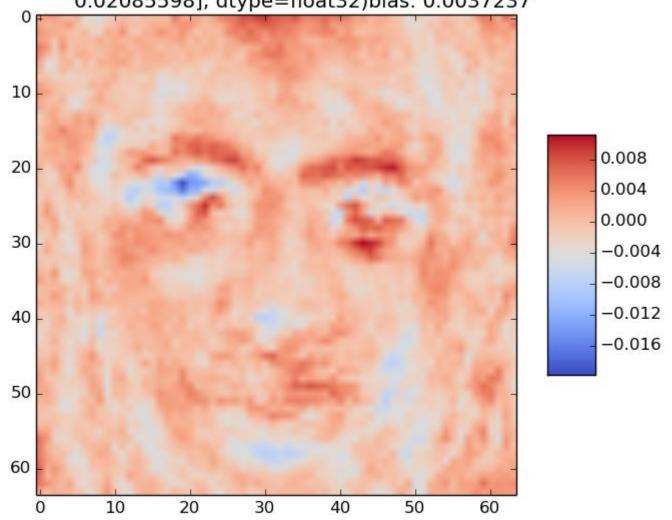






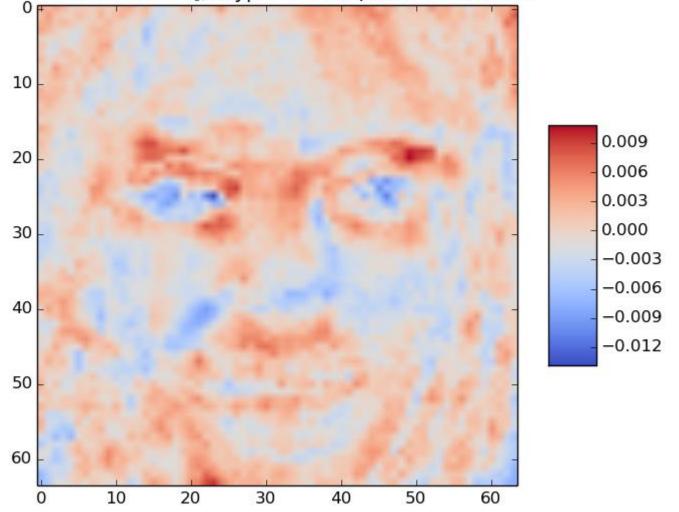


s: array([ 0.09187977, -0.01672127, -0.0360681 , 0.02101913, -0.12962481, 0.02085598], dtype=float32)bias: 0.0037237



300 hidden units, 6 actors, 40 examples each, L2-penalized,  $128 \times 128$  images

:s: array([ 0.031698 , 0.14668576, 0.03825208, 0.01261172, -0.01688866, -0.02065944], dtype=float32)bias: -0.0516133



s: array([-0.0660211, -0.02434859, -0.10672989, 0.00908299, 0.08226717, 0.02301903], dtype=float32)bias: 0.0126341 0 10 0.0075 20 0.0050 0.0025 30 0.0000 -0.0025-0.005040 -0.0075-0.010050 60

10

0

20

30

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

60

50

40

### 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)