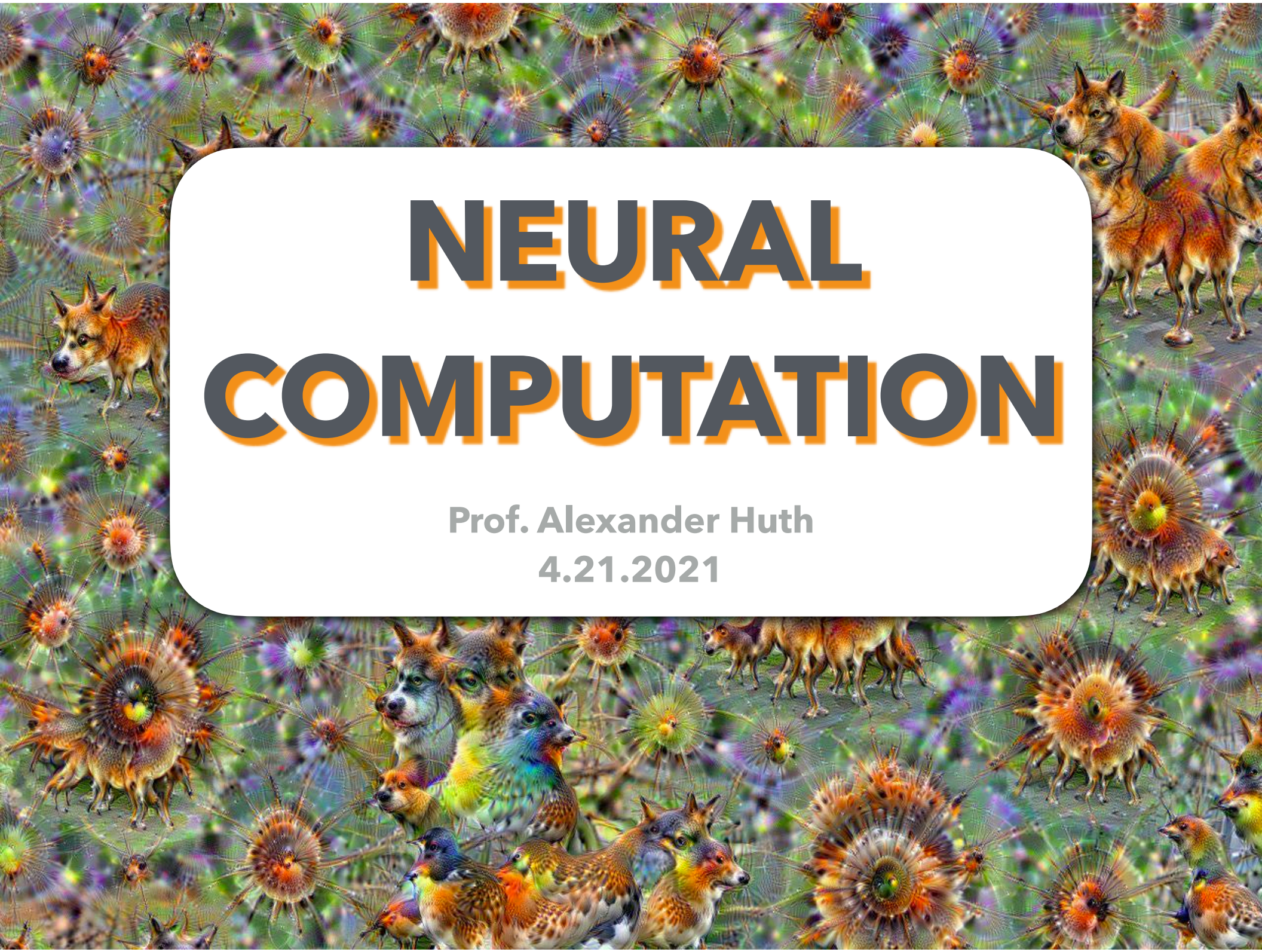
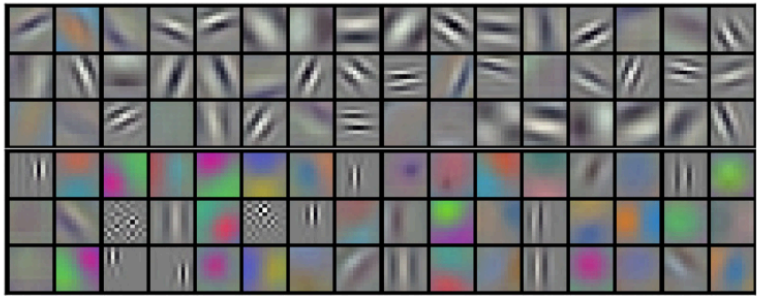

NEURAL
COMPUTATION
Prof. Alexander Huth
4.21.2021

PAPER!

RECAP

* Can we understand the representations learned in ANNs?
  * For input layers it’s ~easy: look at the weights
  * For subsequent layers it’s hard



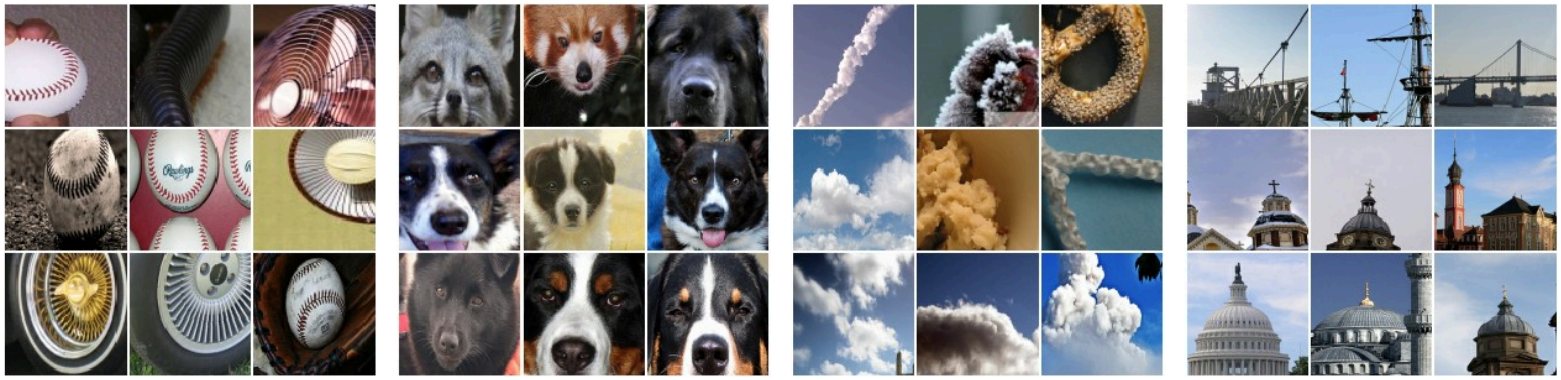
AlexNet input layer features

RECAP

* **Feature Visualization**
  * What types of inputs drive activation in some specific unit?
    * i.e. what is that unit's *receptive field*
  * …by inspecting inputs w/ high activation
  * …by optimizing inputs for high activation

RECAP: FEATURE VISUALIZATION

Best Inputs



Optimized Inputs

From https://distill.pub/2017/feature-visualization/

ATTRIBUTION

* What is it about input *X* that causes unit *i* in the network to have a high activation?
  * i.e., can we ***attribute*** the activation of unit *i* to specific elements of input *X*?
  * aka *relevance*, *contribution*

ATTRIBUTION

X =  , $a_i(X) = 1.7$

what is it about this image that makes this value so high?

PERTURBATION

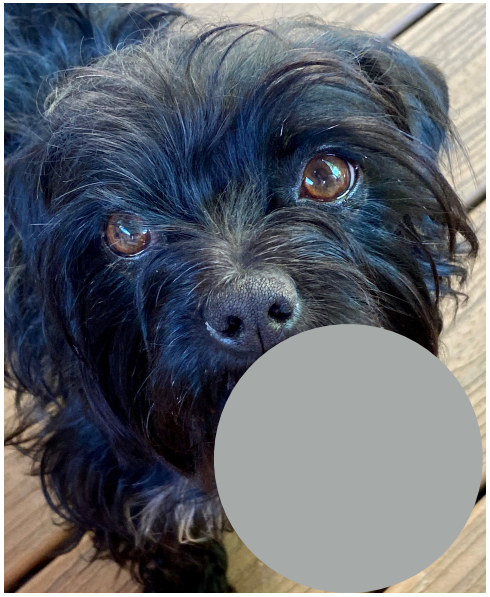
* Suppose we change the input, altering or removing some part of it
  * *Formally, change input:* $X \rightarrow X'$
* Then we compare the activation before and after the change
  * *Compute difference:* $\Delta a_i(X') = |a_i(X) - a_i(X')|$
* If the difference is **large**, the part that we changed was **important**
  * If the difference is **small**, then the part we changed was **not important**

PERTURBATION

Original image

X, $a_i(X)$ = 1.7

Perturbed image 1



X_1', $a_i(X_1')$ = 1.5

this perturbation did not cause a big change in activation

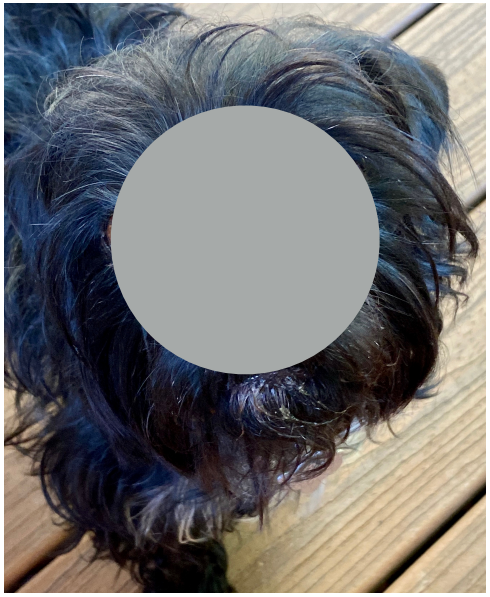
$\Delta a_i(X_1')$ = 0.2

PERTURBATION

Original image

X, $a_i(X)$ = 1.7

Perturbed image 2



X_2', $a_i(X_2')$ = 0.1

*this perturbation caused a **huge** change in activation*

$\Delta a_i(X_2')$ = 1.6

PERTURBATION

this region of the image is much more important to the activation of this unit…

X_2', $\Delta a_i(X_2') = \mathbf{1.6}$

…than this region of the image!

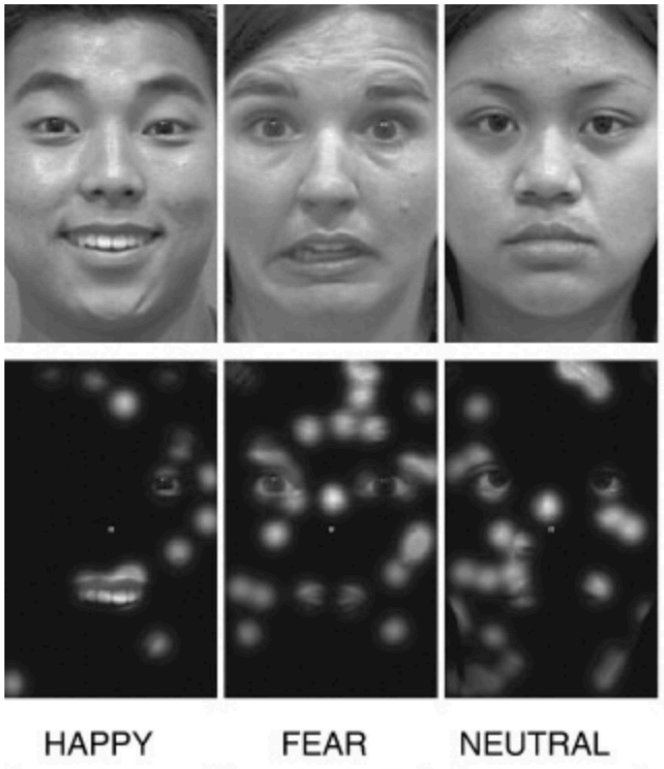
X_1', $\Delta a_i(X_1') = \mathbf{0.2}$

PERTURBATION: BUBBLES

* This technique was first used in psychology & neuroscience, where it's typically done using **bubbles** *[Gosselin & Schyns, 2001]*

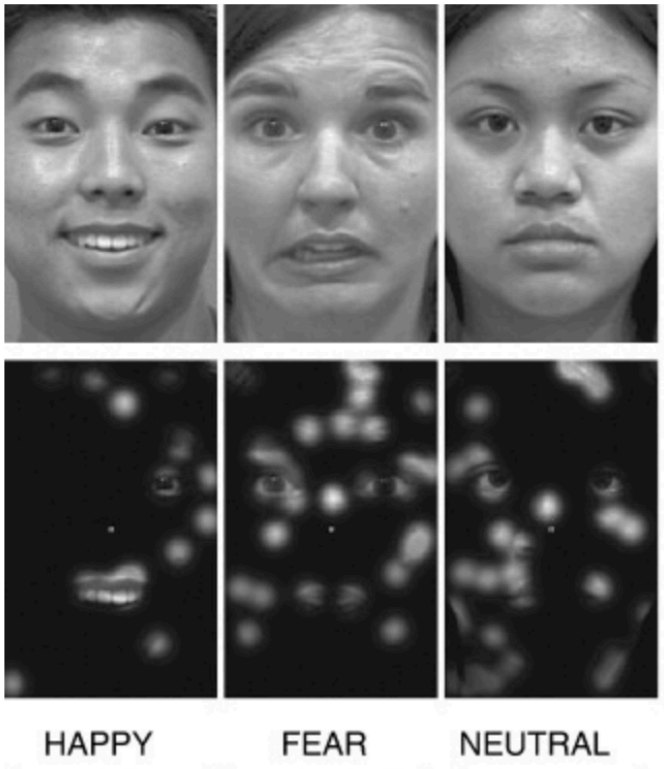
PERTURBATION: BUBBLES

* Experimental stimuli (e.g. faces with different expressions) are masked by **bubble masks**
  * Bubble masks are generated by placing random gaussian blobs on on image-shaped background



PERTURBATION: BUBBLES

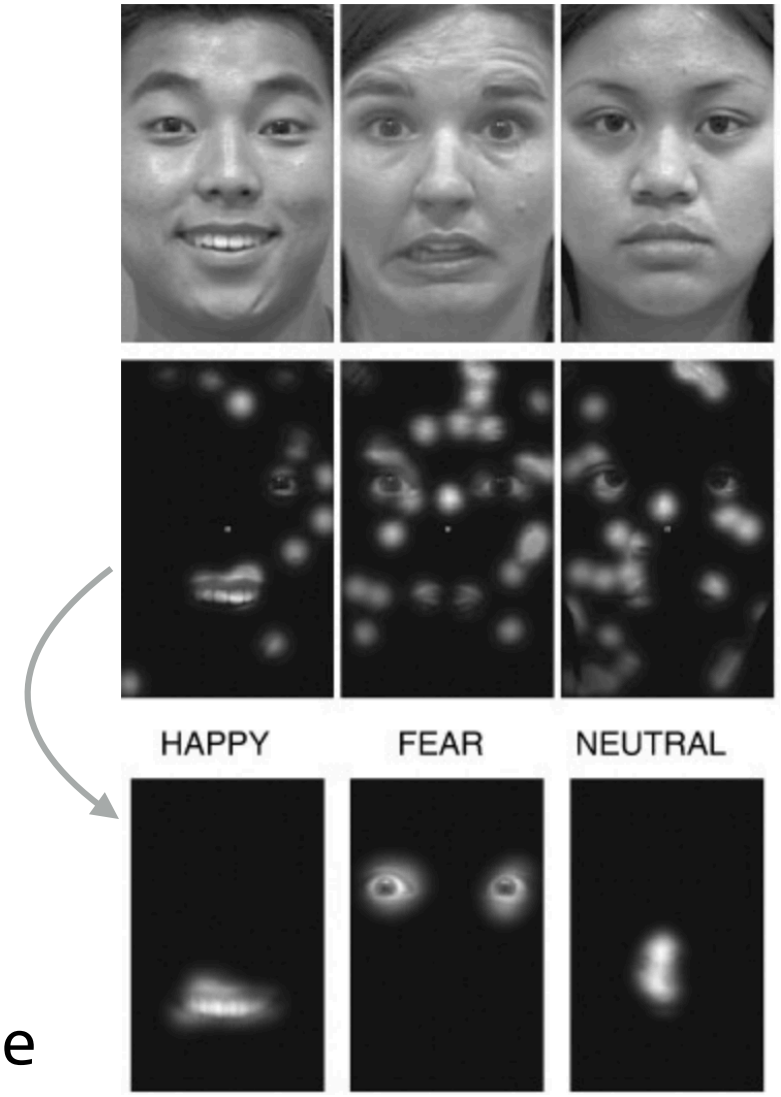
* In a behavioral experiment:
  * Subjects are asked to judge the *expression* (i.e. *happy*, *fearful*, *neutral*) of each (bubbled) stimulus



PERTURBATION: BUBBLES

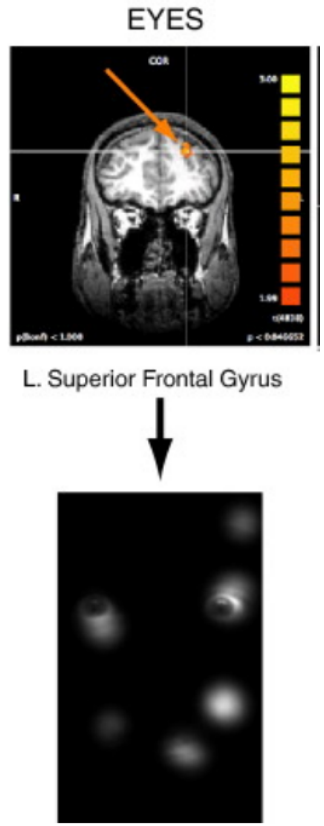
* Given:
  * **1**, lots of randomly bubbled versions of an image, &
  * **2**, a few behavioral responses for each

* We can compute the **diagnostic image** by taking the average of the bubble masks that resulted in a correct behavioral response



PERTURBATION: BUBBLES

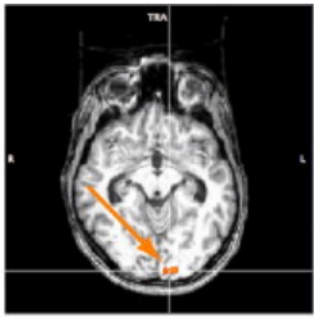
* A similar procedure can be carried out with neuroscience data: *[Smith et al., 2008]*
  * Show bubbled versions of images to subject while they undergo fMRI
  * Compute **diagnostic image** for a voxel by comparing masks that give high response to masks that give low response



*a brain area in frontal cortex that seems to care about the **eyes***

PERTURBATION: BUBBLES

* A similar procedure can be carried out with neuroscience data: *[Smith et al., 2008]*
  * Show bubbled versions of images to subject while they undergo fMRI
  * Compute **diagnostic image** for a voxel by comparing masks that give high response to masks that give low response



*a brain area in occipital cortex that seems to care about the **mouth***

THE TROUBLE WITH BUBBLES

* Lots of hyperparameters…
  * *(how many bubbles? how big? what shape? positive or negative mask?)*
* …and analysis choices
  * *(how to combine information across different bubblings of the same input?)*

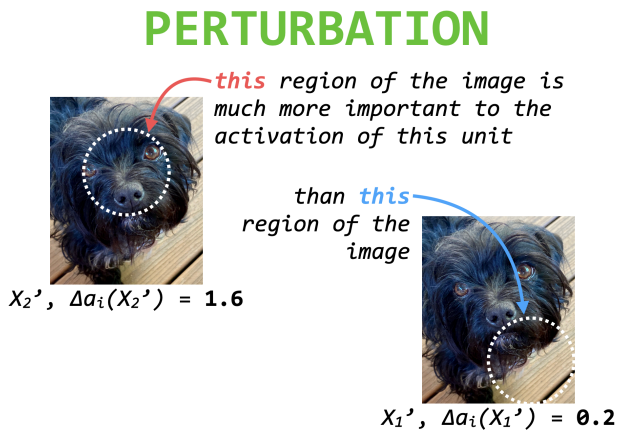
BUBBLES FOR ANNS

* This technique can also be applied to artificial neural networks
  * *(although typically a simpler version with a single square “bubble” is used)*

BUBBLES FOR ANNS

* This approach follows closely the earlier example. But how do we know it works / gives sensible results?

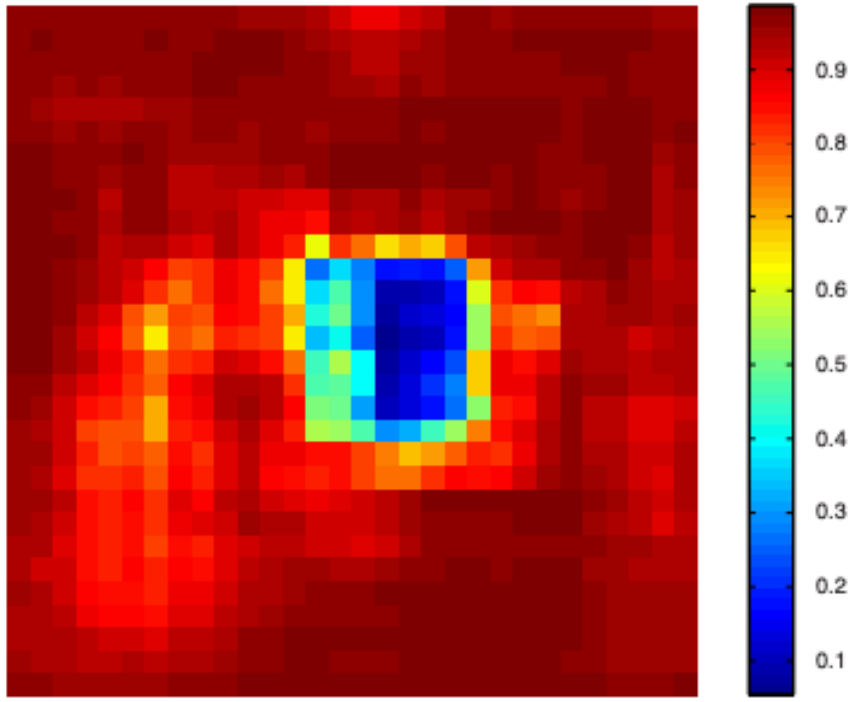
* One test: make our unit of interest one of the **outputs** (i.e. **classification nodes**) in the network *[Zeiler & Fergus, 2014]*



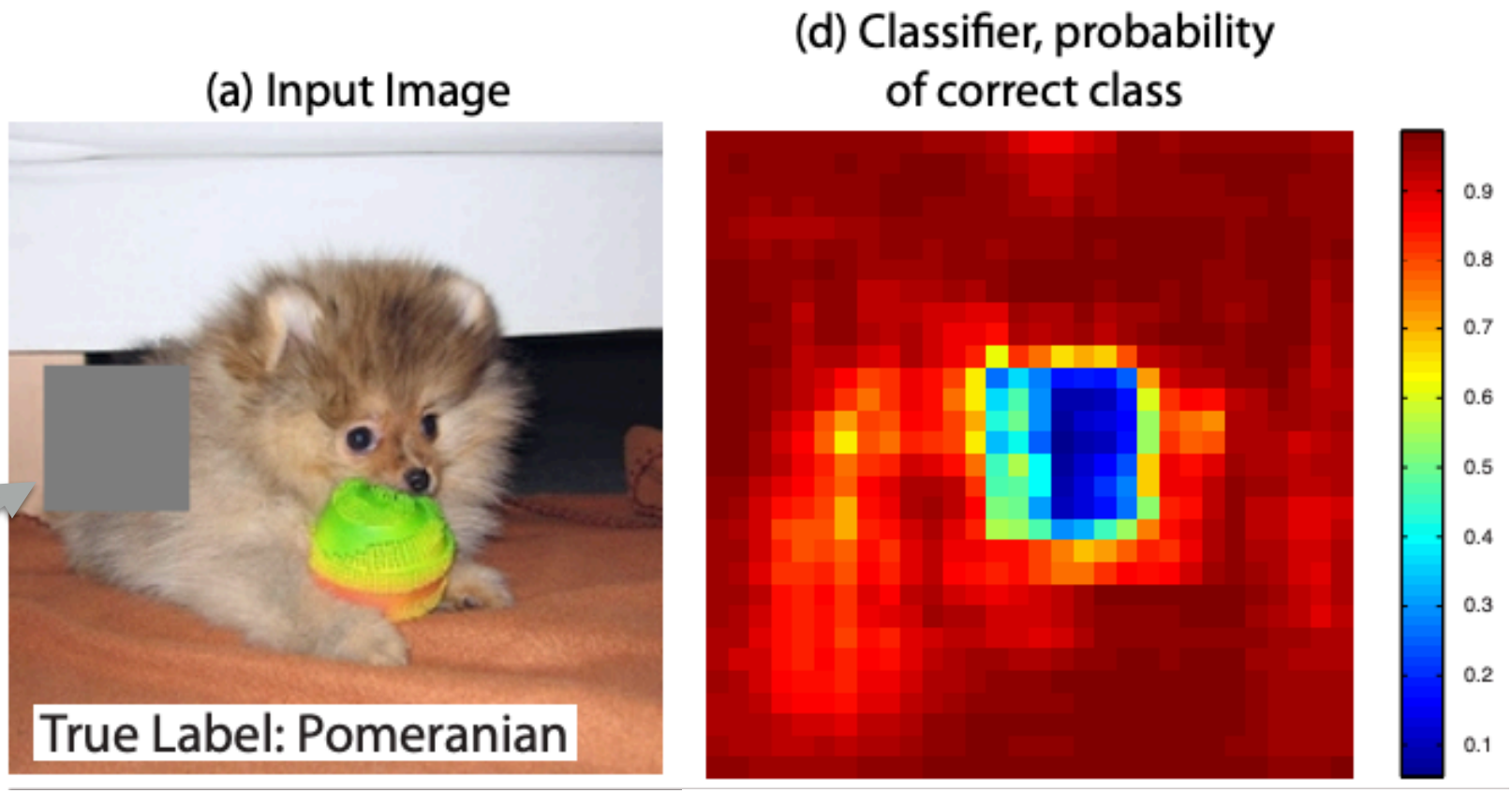
BUBBLES FOR ANNS

(a) Input Image

(d) Classifier, probability of correct class

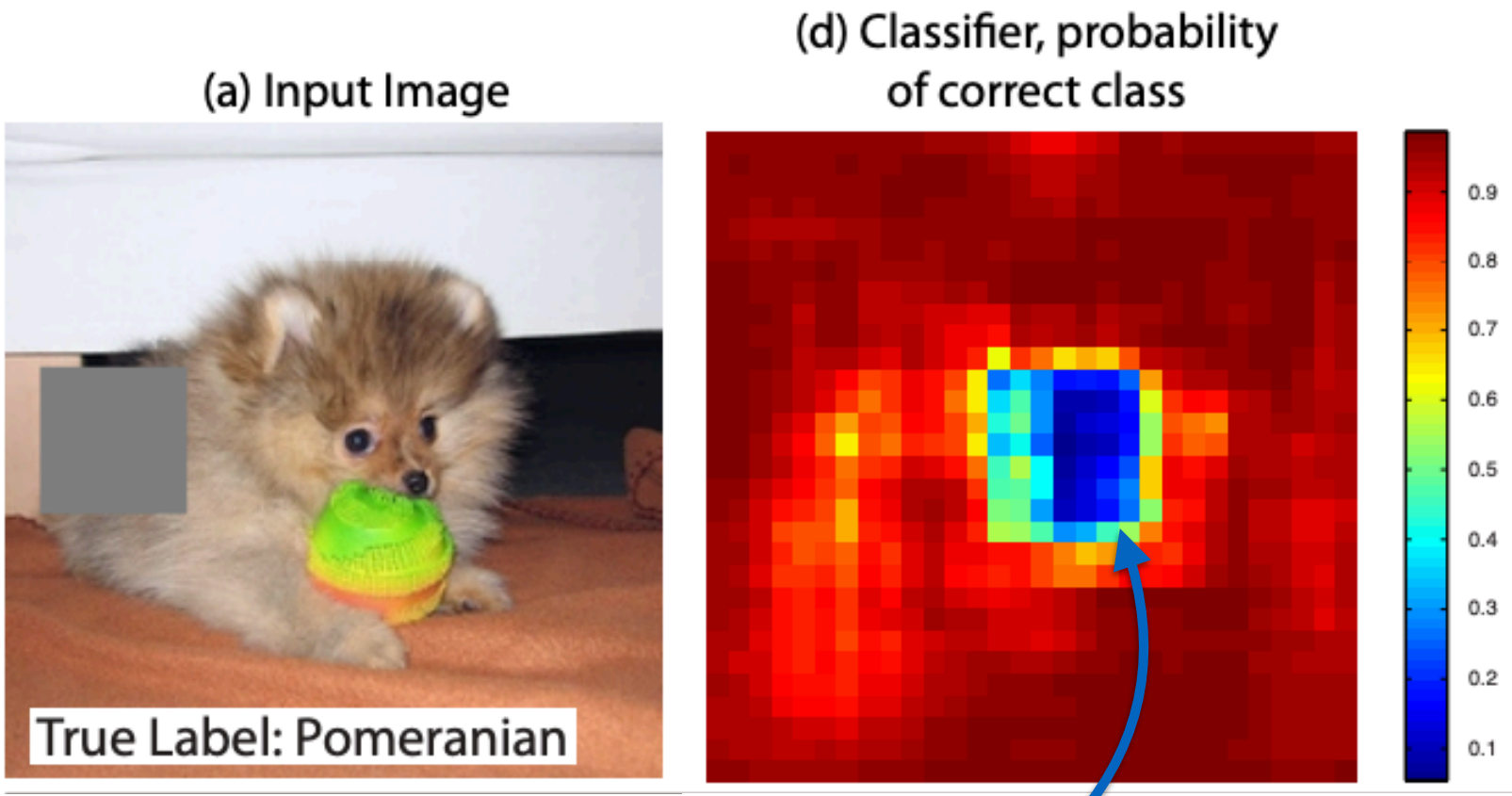


BUBBLES FOR ANNS



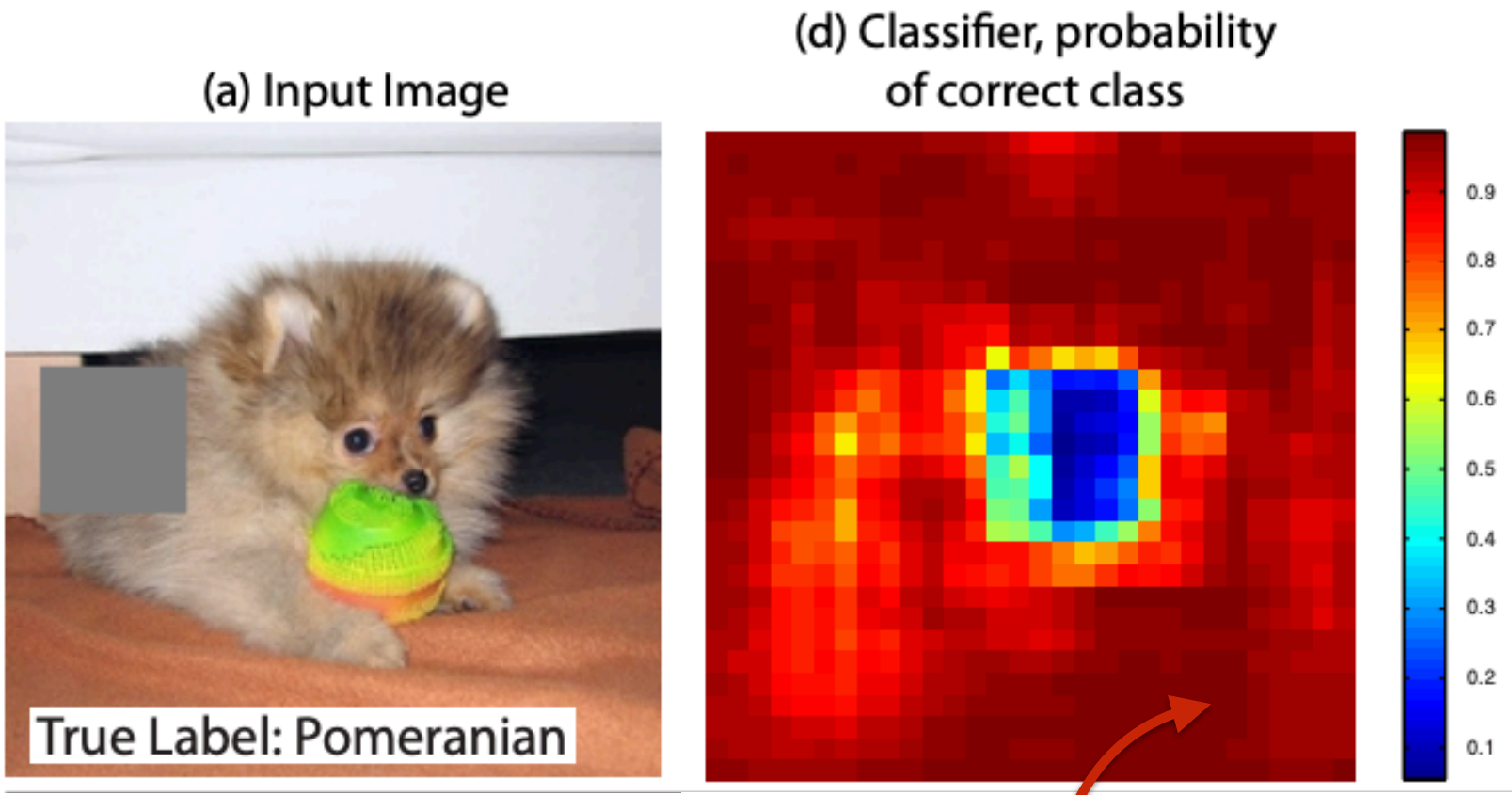
this is an example of the occluder that was used to cover up different parts of the image

BUBBLES FOR ANNS



this central region of the image is important for classification, so removing it reduces ***P(correct class | image)***

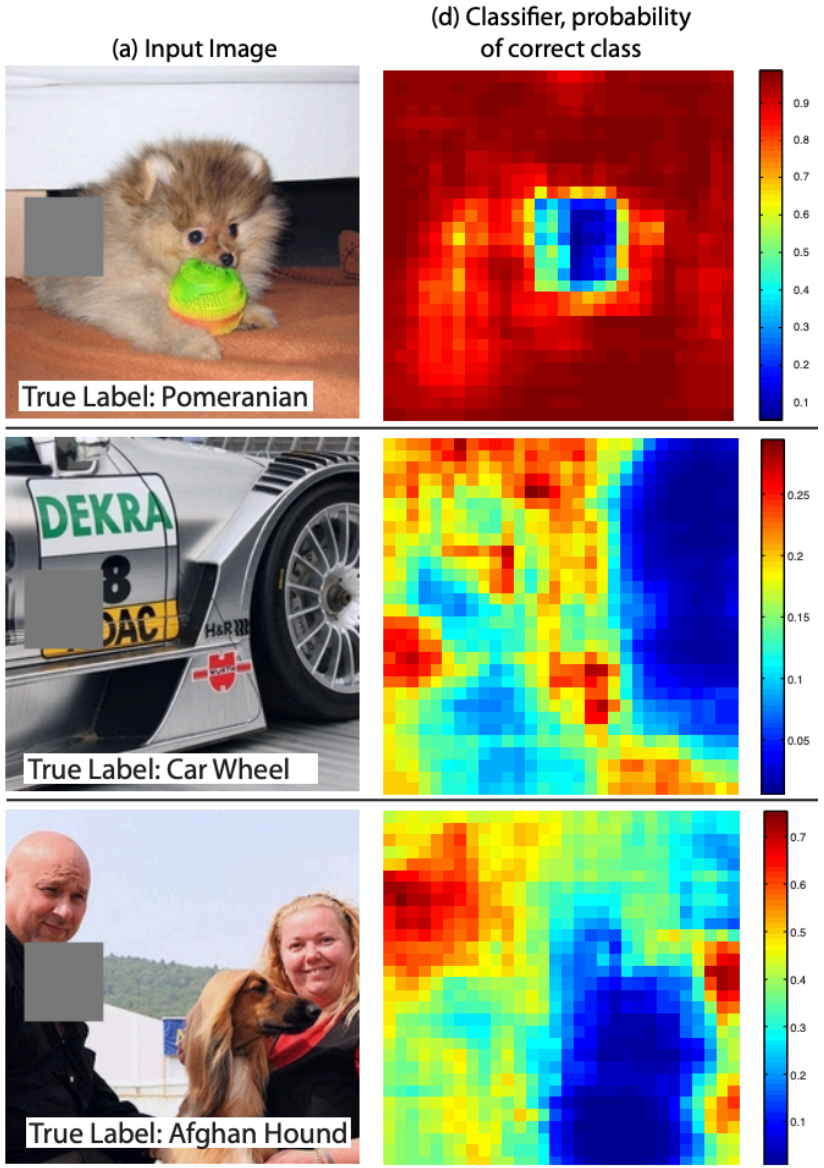
BUBBLES FOR ANNS



this peripheral region of the image is ***not*** *important for classification, so removing it* ***doesn't*** *reduce* ***P(correct class | image)***

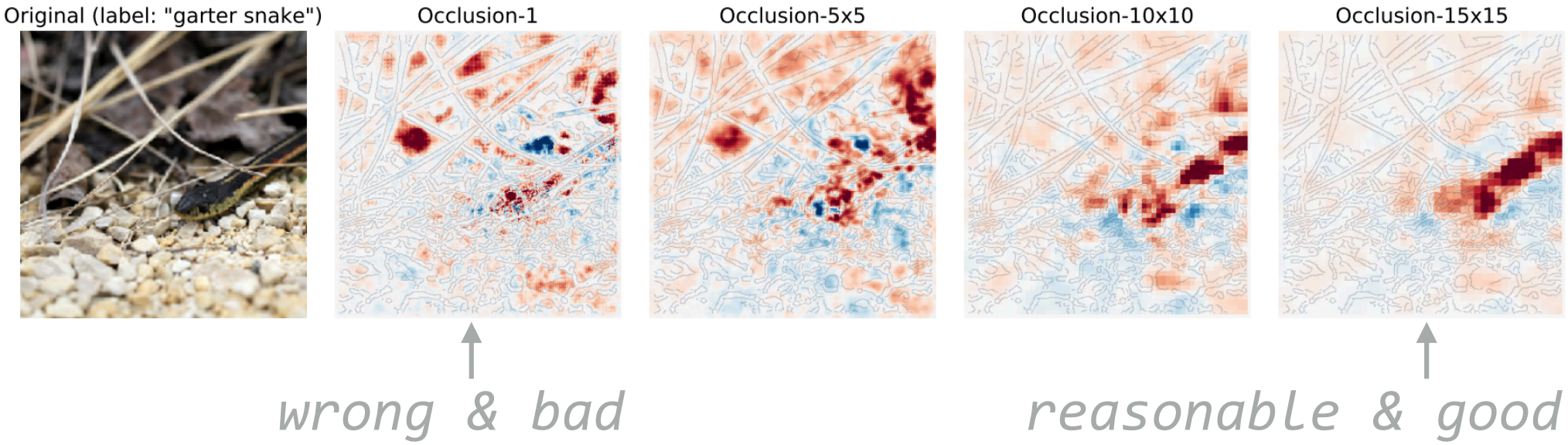
BUBBLES FOR ANNS

more examples:



BUBBLES FOR ANNS

* Here the **size** of the occluder/bubble is also important *[Ancona et al., 2018]*



GRADIENT ATTRIBUTION

* But wait! We also have **complete knowledge** of the network
  * Surely we can use this information to do attribution…?
  * –> Instead of testing different occlusions/ablations/bubbles, let’s use the fact that we can *compute derivatives* of some things in the network w/r/t other things in the network

GRADIENT ATTRIBUTION

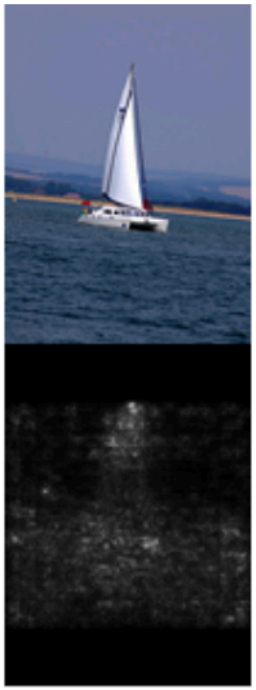
* For example, we can compute a **class saliency map** as the derivative of the **class score** with respect to each pixel in the **input image** *[Simonyan et al., 2014]*

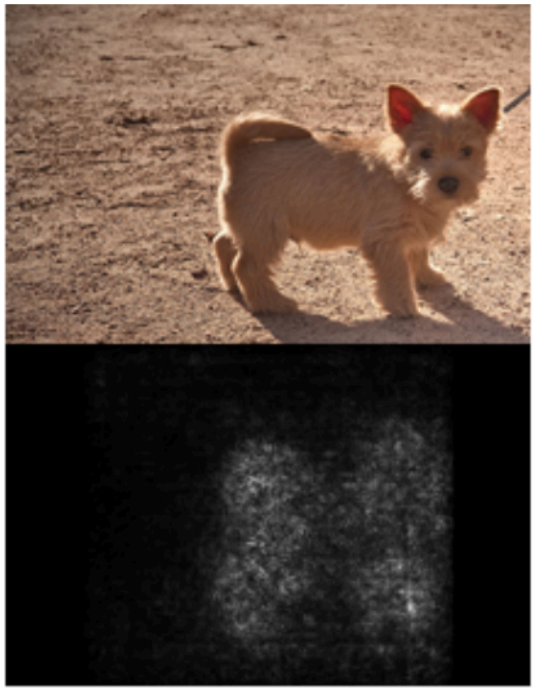
$$R_c(X) = \left| \frac{\partial S_c(X)}{\partial X} \right|$$

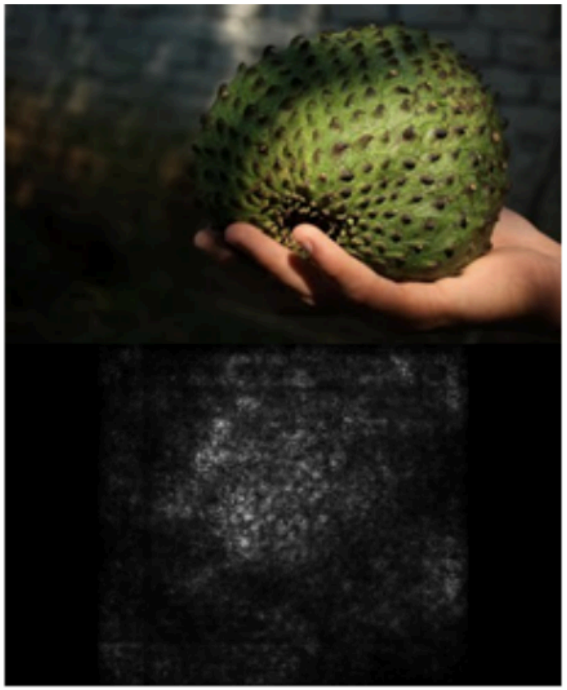
GRADIENT ATTRIBUTION

input:

saliency map:







GRADIENT ATTRIBUTION

* Problem:
  * This simple gradient method has problems with ***sensitivity***
  * i.e. nonlinear activation functions (such as the ReLu) can *saturate*, making gradients extremely small even for important features

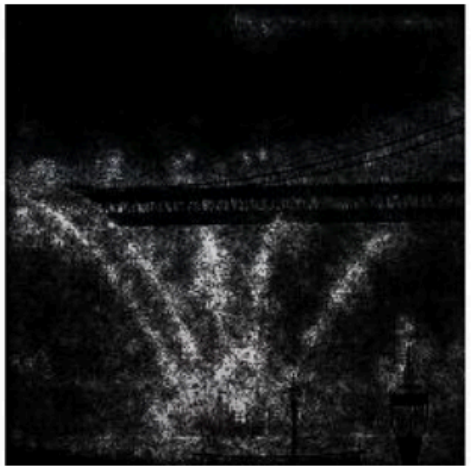
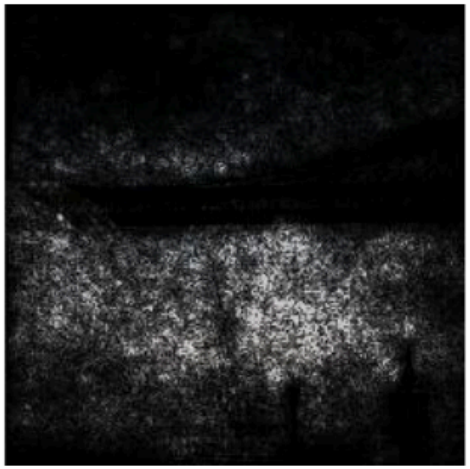
GRADIENT ATTRIBUTION

* Sensitivity example *[Sundararajan et al., 2017]*:
  * Let $f(x) = 1 - \text{ReLu}(1 - x)$
    * Recall:
      $\text{ReLu}(x) = \{0 \text{ for } x < 0;\ x \text{ for } x >= 0\}$
  * Now let $x = 2$, so $f(x) = 1$
  * → $\boldsymbol{df(x)/dx = 0}$ ⛔ 🙅 ⚠️

GRADIENT ATTRIBUTION

* Solution *[Sundararajan et al., 2017]*:
  * Instead of just computing gradient locally at the given input, compute the **integrated gradient** across a continuum of inputs ranging from a baseline (e.g. 0) to the original input

GRADIENT ATTRIBUTION

Original image	Top label and score	Integrated gradients	Gradients at image
	Top label: reflex camera Score: 0.993755		
	Top label: fireboat Score: 0.999961		

GRADIENT ATTRIBUTION

* Integrated gradients seems more mathematically correct (i.e. achieves sensitivity & other desirable qualities)
  * and the attributions it gives are somewhat different *(background vs. foreground?)*
  * but is it any more interpretable?

NEXT TIME

* Recent applications of these techniques:
  * Transformer language models, with
  * System construction / identification