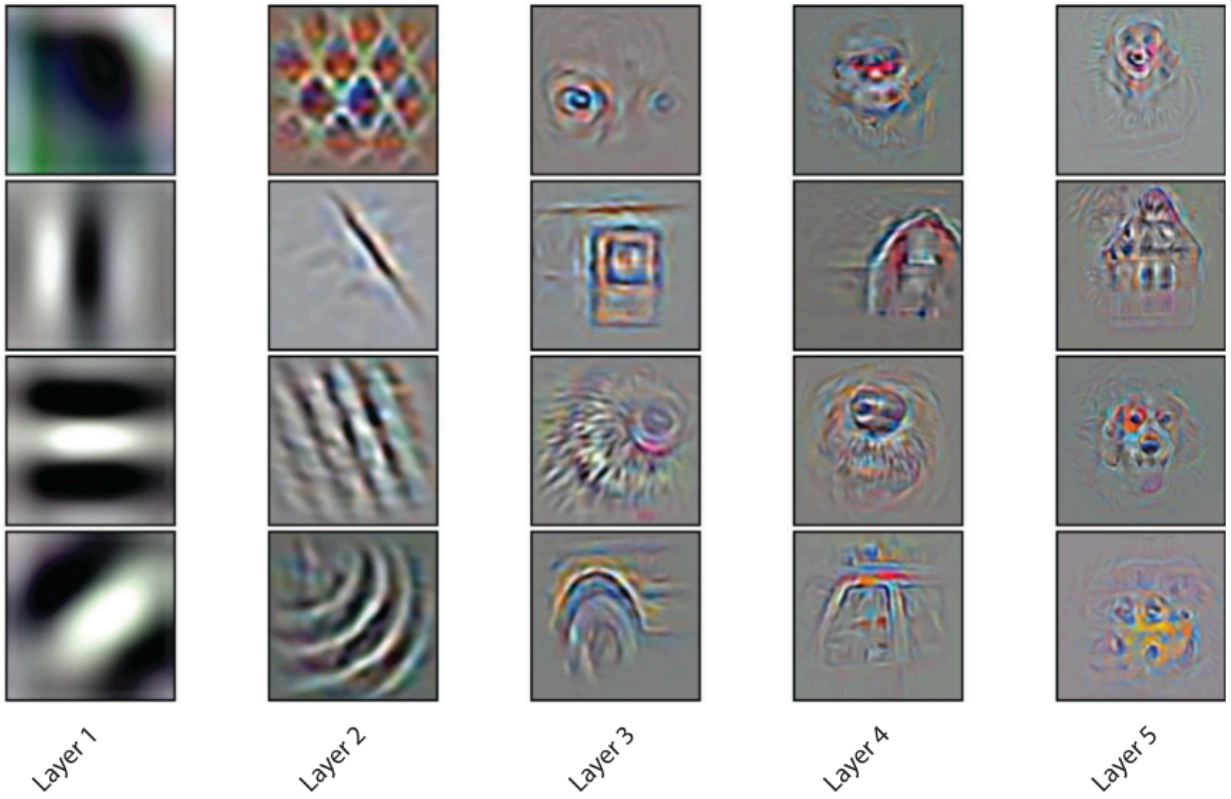
# Shedding light on a black box

## Techniques for interpreting hidden layers of a Convolutional Neural Network

Kirill Korotaev

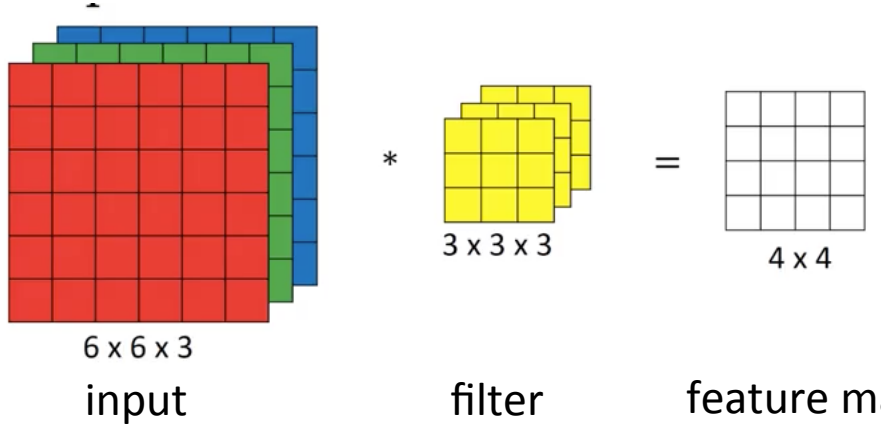


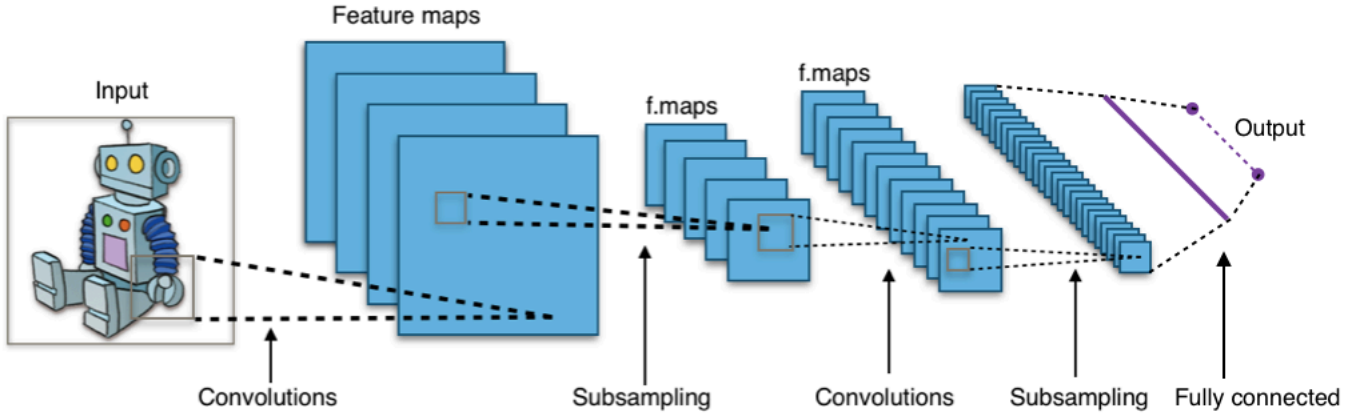

# Why CNN?

- CNN were inspired by the research on visual cortex and by design share some common features with it, such as **hierarchical organization** and **local connectivity**.
- Several studies have found surprising representational similarity between processing stages of visual cortex and layers of CNN (Cadieu et al., 2014; Khaligh-Razavi & Kriegeskorte, 2014 )
- CNN can explain both ventral and dorsal stream processing at the state-of-the-art level (Güçlü & van Gerven, 2014, 2017)
- CNN can match and sometimes even surpass human-level performance in well-defined visual tasks (He, Zhang, Ren, & Sun, 2015; Krizhevsky, Sutskever, & Hinton, 2012).

# How they work?

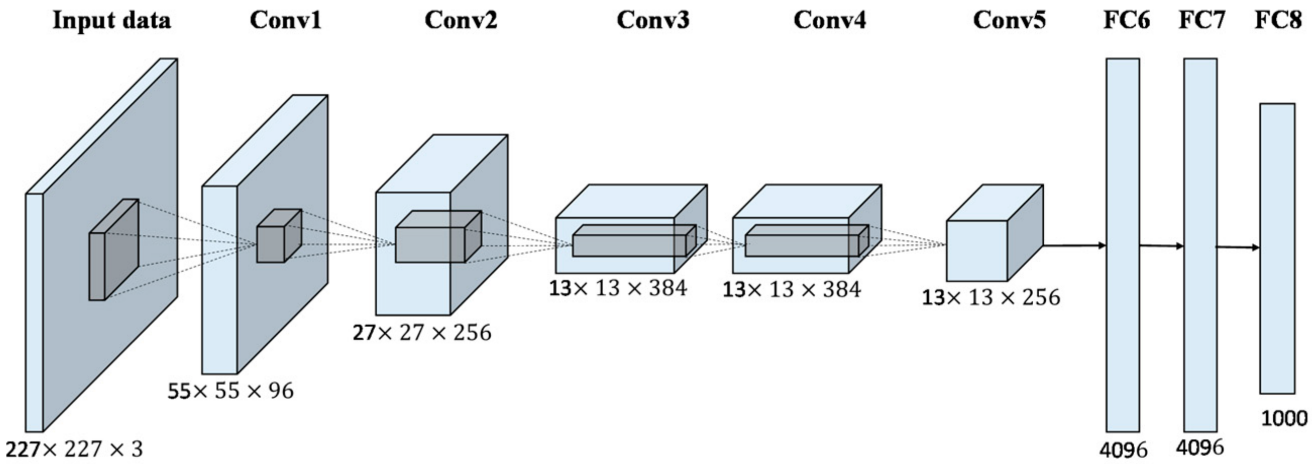
- CNN are **feedforward** and **supervised**
- **Convolutional filters** are applied to a portion of an input image to produce **feature maps**
- **Pooling** layers are used to downsample the feature maps, **ReLU** layers are used to introduce non-linearity to the system



# Loading pre-trained AlexNet

- AlexNet is a CNN trained on more than 1M images from ImageNet database
- The network is 8* layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and animals.

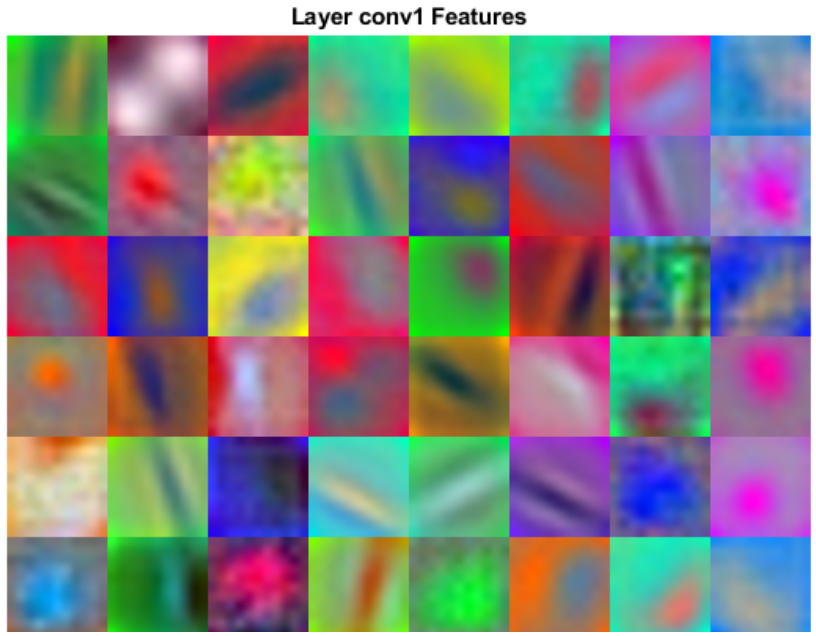


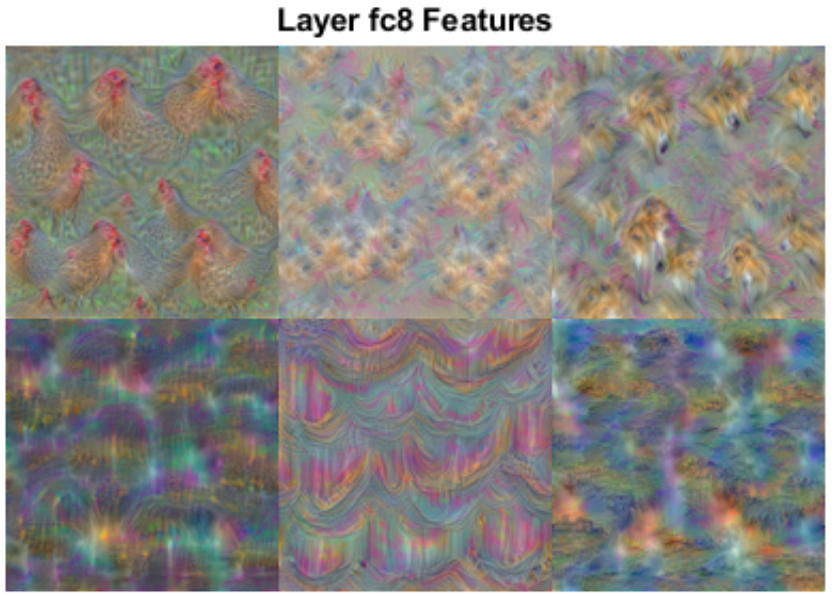

https://ch.mathworks.com/help/deeplearning/ref/alexnet.html

# Feature visualization

```
I = deepDreamImage(net,layer,channels,'PyramidLevels',1);
```

- In Matlab u can use DeepDreamImage to visualize features of convolutional and fully-connected layers
- The filters in first conv layers are edge detectors and color filters. This is consistent across different datasets or tasks
- The filters in the last FC layer are high-level combinations of features learned by previous layers and correspond to full representations of an output class, they are highly specific




https://ch.mathworks.com/help/deeplearning/examples/visualize-features-of-a-convolutional-neural-network.html

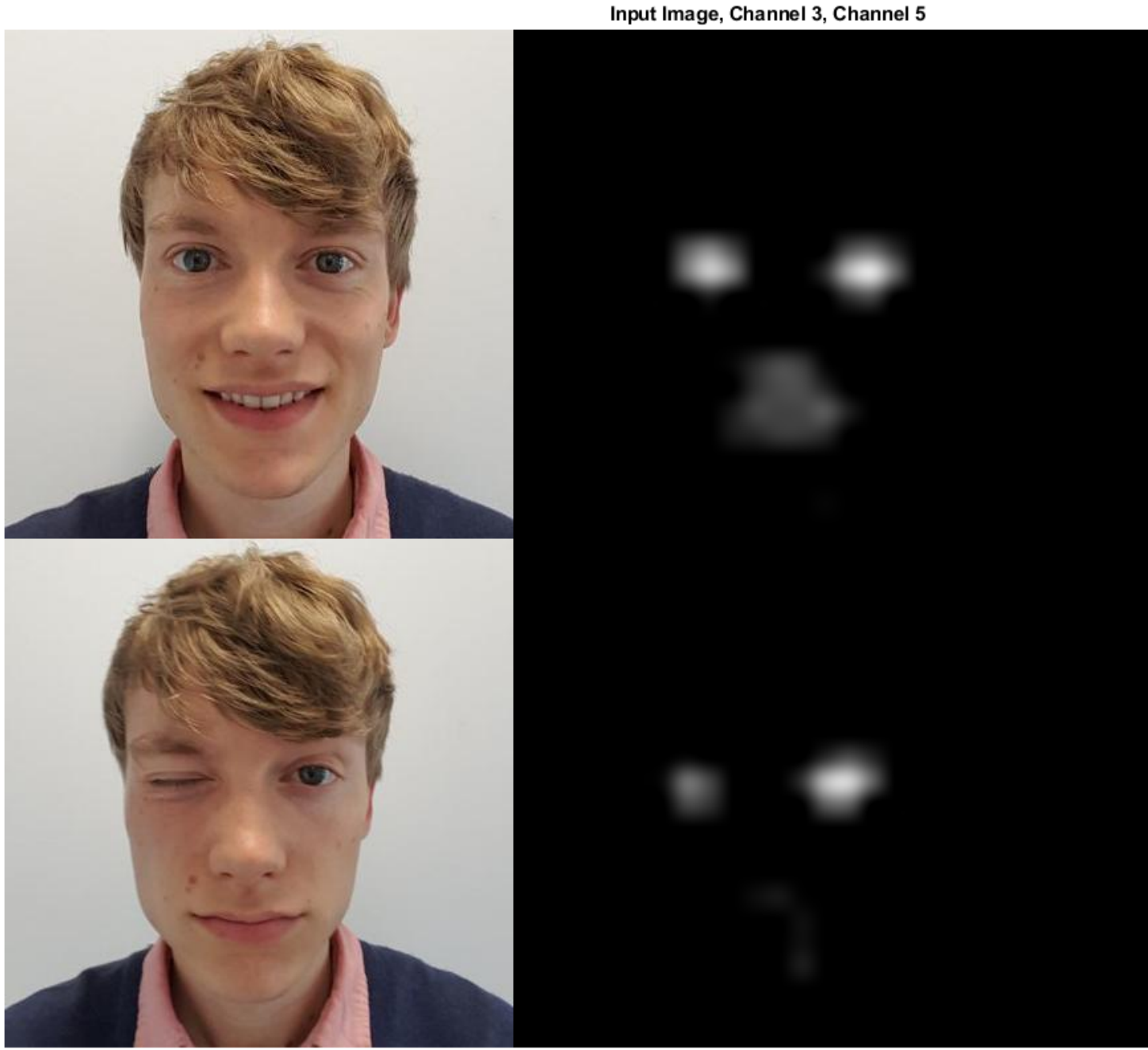
# Activation visualization

- Another method to explore which features do hidden layers of CNN learn is to feed a new image to it and look which channels in the given layer end up activated.
- White pixels correspond to the strong positive activations, and black pixels correspond to strong negative activations.

https://ch.mathworks.com/help/deeplearning/examples/visualize-activations-of-a-convolutional-neural-network.html

# Activation visualization

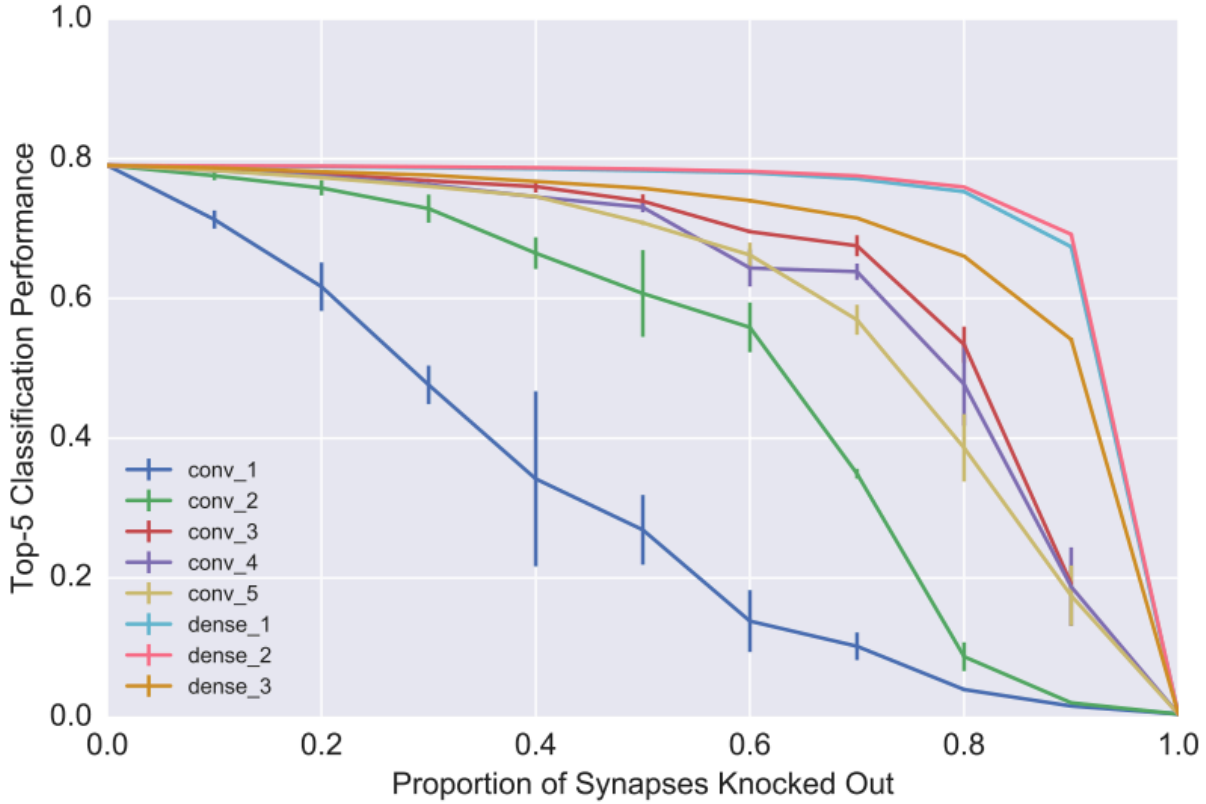
- Than you can investigate specific channels. You can manipulate the input to gather a solid evidence what does the filter do.
- Here the filter learned to be an eye-detector.




https://ch.mathworks.com/help/deeplearning/examples/visualize-activations-of-a-convolutional-neural-network.html

# Synapse KO

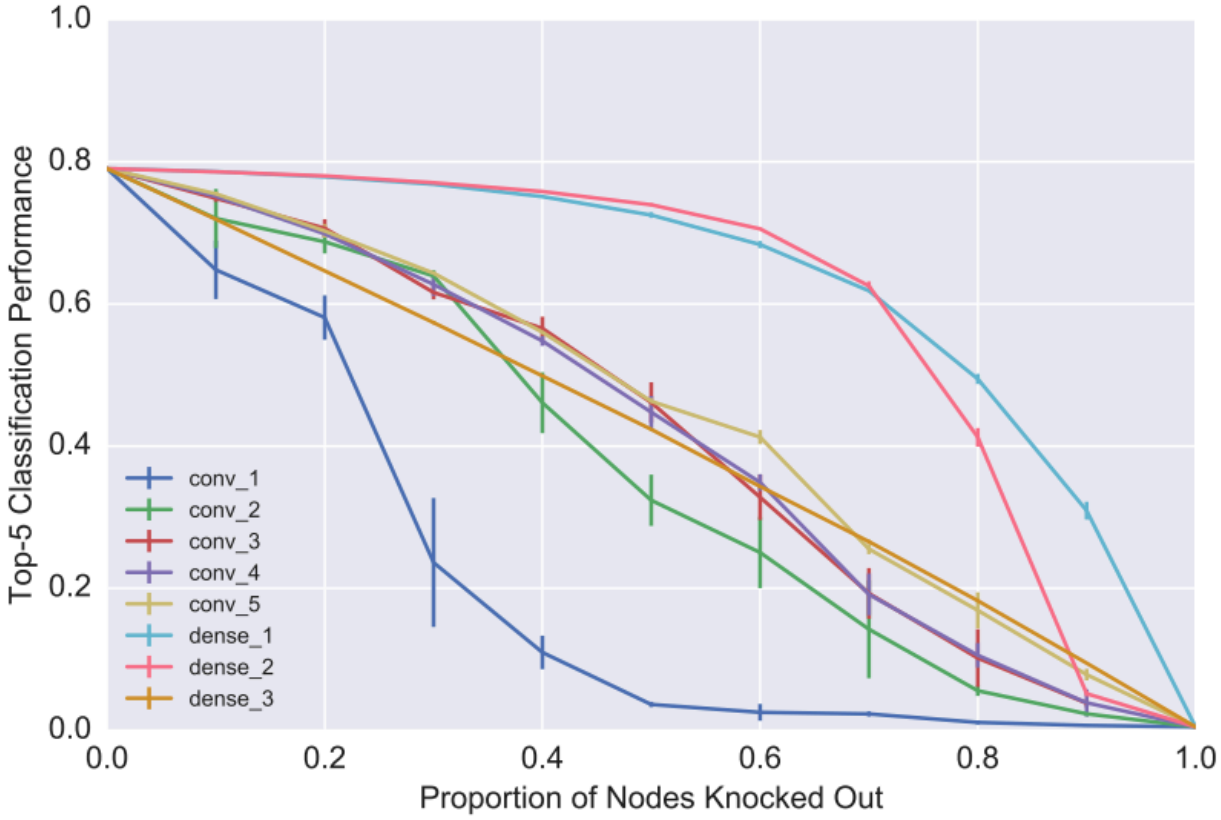
- The proportion of weights leading into a given layer is set to 0. The removal of weights from the deep conv layers demonstrate incredible robustness. In contrast, removal of weights from early convolutional layers classification accuracy dramatically, so it falls at the chance level.




Cheney, Schrimpf, & Kreiman, 2017

# Node KO

- In CNN, Node KO corresponds to the removal of convolutional filter. Compared to synapse KO, it demonstrates more clustered and focused removal of information. Node KO showed similar dynamics, as removal of nodes in earlier layers impaired networks performance more dramatically.




Cheney, Schrimpf, & Kreiman, 2017

# Synapse KO in Matlab

Synapse KO in Matlab is tricky, because the trained network is stored as a “net” object which doesn’t permit editing.

**1)Create new layers array with the same weight values**

**2)Knockout weights and biases in a layer of interest**

**3) Pseudo-train a new network from layers, for that set ‘InitialLearnRate’ to a low value, set ‘L2Regularization’ to 0 (weight decay) and abort training early**

```
for l = 1:length(layers)
  if isprop(layers(l), 'Weights') % Does layer l have weights?
    layers(l).Weights = net.Layers(l).Weights;
  end
  if isprop(layers(l), 'Bias')% Does layer l have biases?
    layers(l).Bias = net.Layers(l).Bias;
  end
end
```

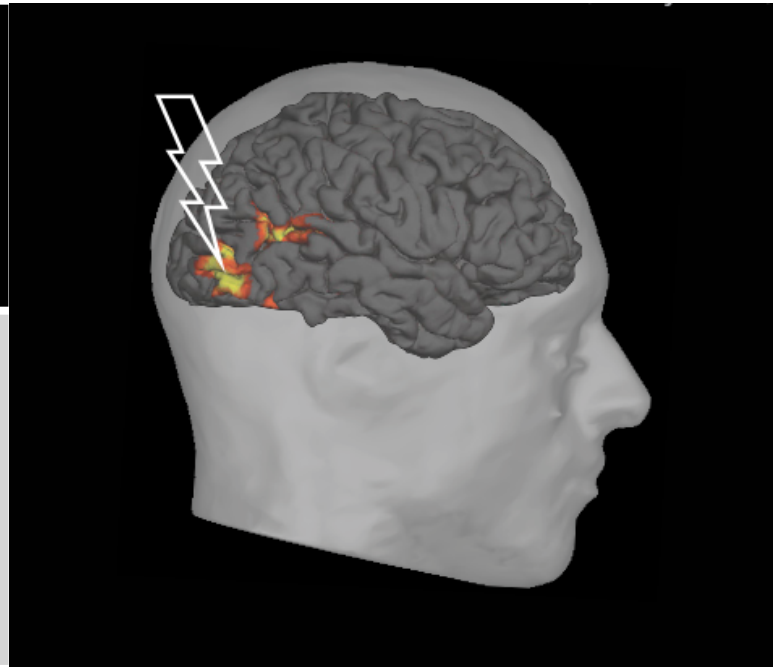
```
layers(11).Weights(:,:,:)=0;
layers(11).Bias(:,:,:)=0;
```

# Background: TMS

Stimulation of visual cortex introduces noise to the target visual areas in region specific ways:

**TMS to OFA impairs recognition of faces, but has no effect on recognition of objects** (Pitcher, Charles, Devlin, Walsh, & Duchaine, 2009).

**TMS to OPA impairs recognition of scenes, but has no effect on recognition of either faces or objects** (Dilks, Julian, Paunov, & Kanwisher, 2013).



# Modeling OFA and OPA with CNN

Dataset consisted of paired images of faces and houses. The task for CNN was to classify images pairs on those who depict same and different identity.

- Training set : 2400 images
- Testing set : 480 images

Overall accuracy: 75%

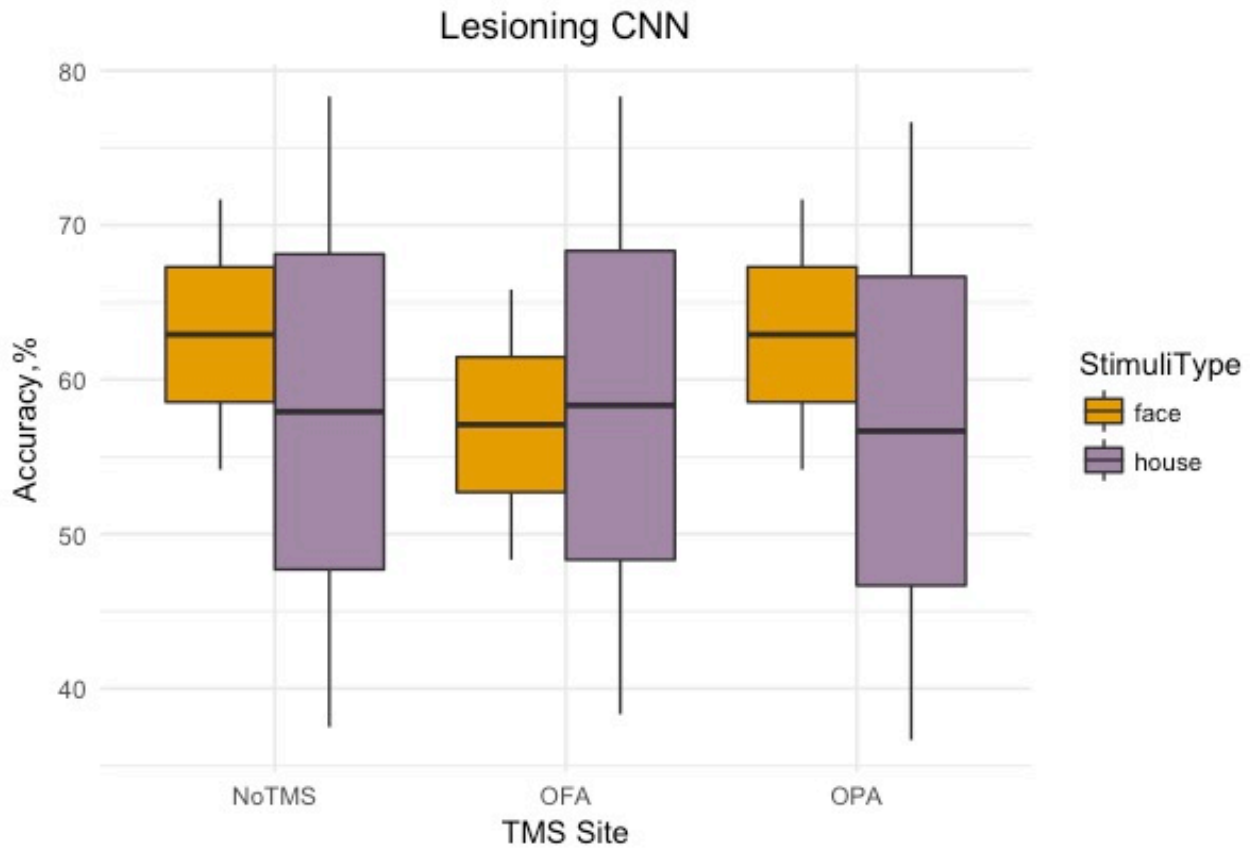
**same**

**different**

# Lesioning CNN

To mimic TMS interfering effects on CNN during the object identification task we knocked out weights in the part of conv3 layer (Cheney, Schrimpf, & Kreiman, 2017).

We have proven the concept that it is possible to selectively knockout weights that are exclusively sensitive to either faces or places.



Thank you for your attention !