Lunch & Learn:

features in convolutional neural nets

presentation by: Denis Kazakov

Rough Notes (read before looking at slides)

Slides
Key points
I am going to refer to brain's analogies and how humans learn purely for metaphorical purposes. I am not making any claims whatsoever that the two are anyhow identical

0. What did I actually work on
    - Road Lane classifier - too small of a dataset - noticed that it learned form context, not from roads
    - Traffic Light detectors - pipeline -
    - got to learn DNN's architectures, spark
    - concept of fusion
1. Lack of data => retrain
    - we have to learn how to communicate what we are looking for and in images it's hard
        - imaging being born again and seeing a bunch of images of the world without knowing what the world is. It's hard to learn anything
        - introducing new information as a way to compensate for lack of data
            - * eventually we are trying to teach DNN about object's invariance (we learn it by seeing world in 3d)
            - multiple loss functions
            - augmentations
                - basic
                - noise addition (read that paper)
    - features are real
        - analogy to regression at each layer
        - about how 2 nets have partially distributed clusters of neurons in each layer
    - pre-training as a way to transfer those features, should be generic - can't always rely on truth generation
        - introduce deep dreams and how steepest ascent works
            - show what the filters catch
        - show filters of not pertained -> hmmm (suspish) -> compare to large data basis -> hmmm
        - show examples of pertained and not

Conclusion: training from scratch always isn't easy. Too much data dependency (especially if there is a way to avoid it, worth exploring)all those features - are a complex nonlinear basis. intuitively whatever basis was found for 5mil images is more uniform for "seeing" that the basis it would find fro 50000 images

2. Doubtfulness as a job necessity
    - overfitting isn't easy to catch
        - potential addition: pipeline to visualize network before using it in production.
            - if your test/val set isn't independent enough, you won't see it in the plot.
            - Ex: pertained model tested worse than not pertained. that's uncommon but just looking at the number doesn't tell us anything.
            - segway into adversarial networks?

3. Sidetones & thoughts on the field:
- **- adversarial networks - have smth <=> relation to manifold theory?**
- **- NN have many equivalent local minimas**

Agenda

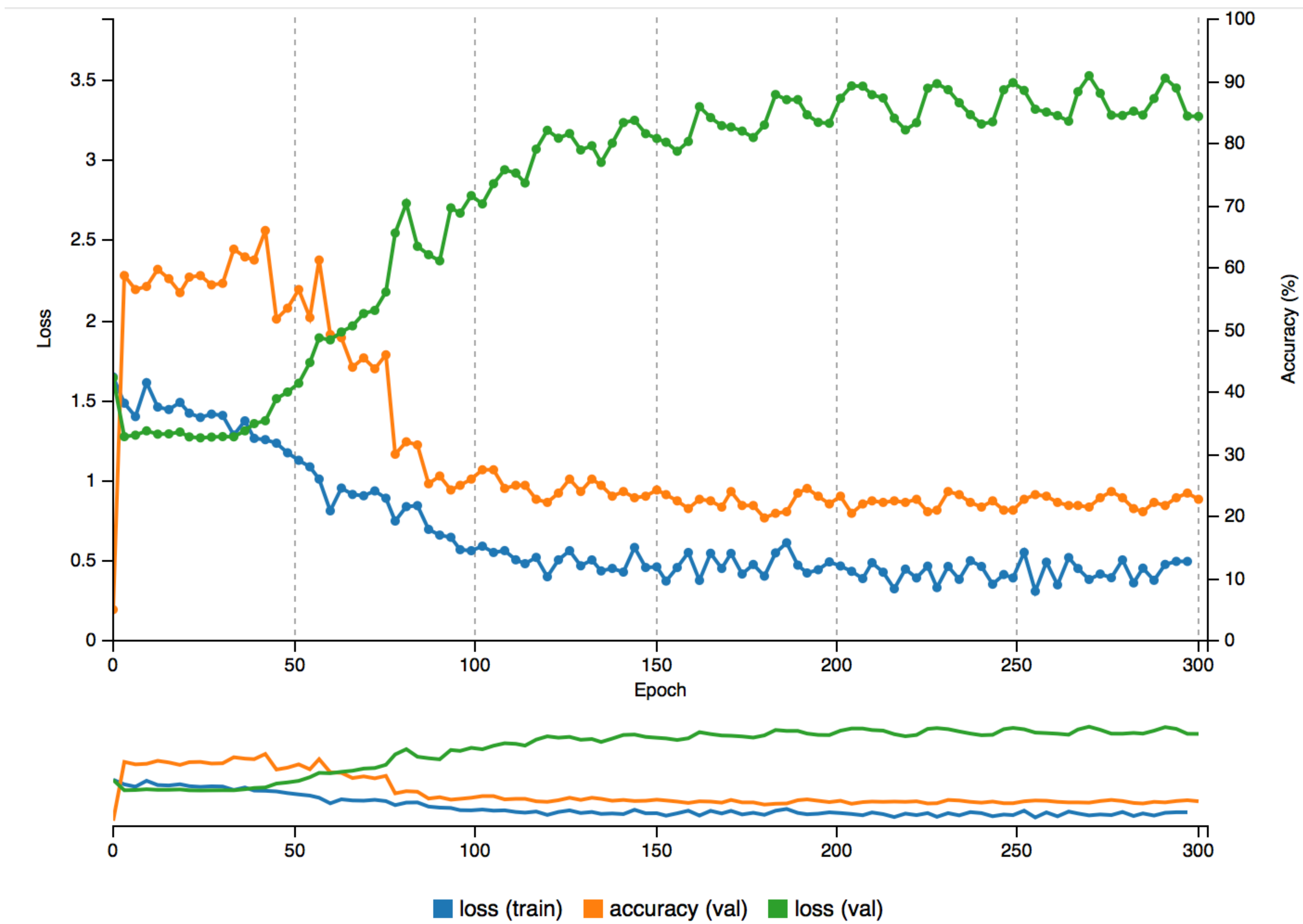
1) My summer experience, things I did
2) Addressing the problem of insufficient data
3) Transferring features in DNN
4) Visualizing layer maximization

Summer Internship

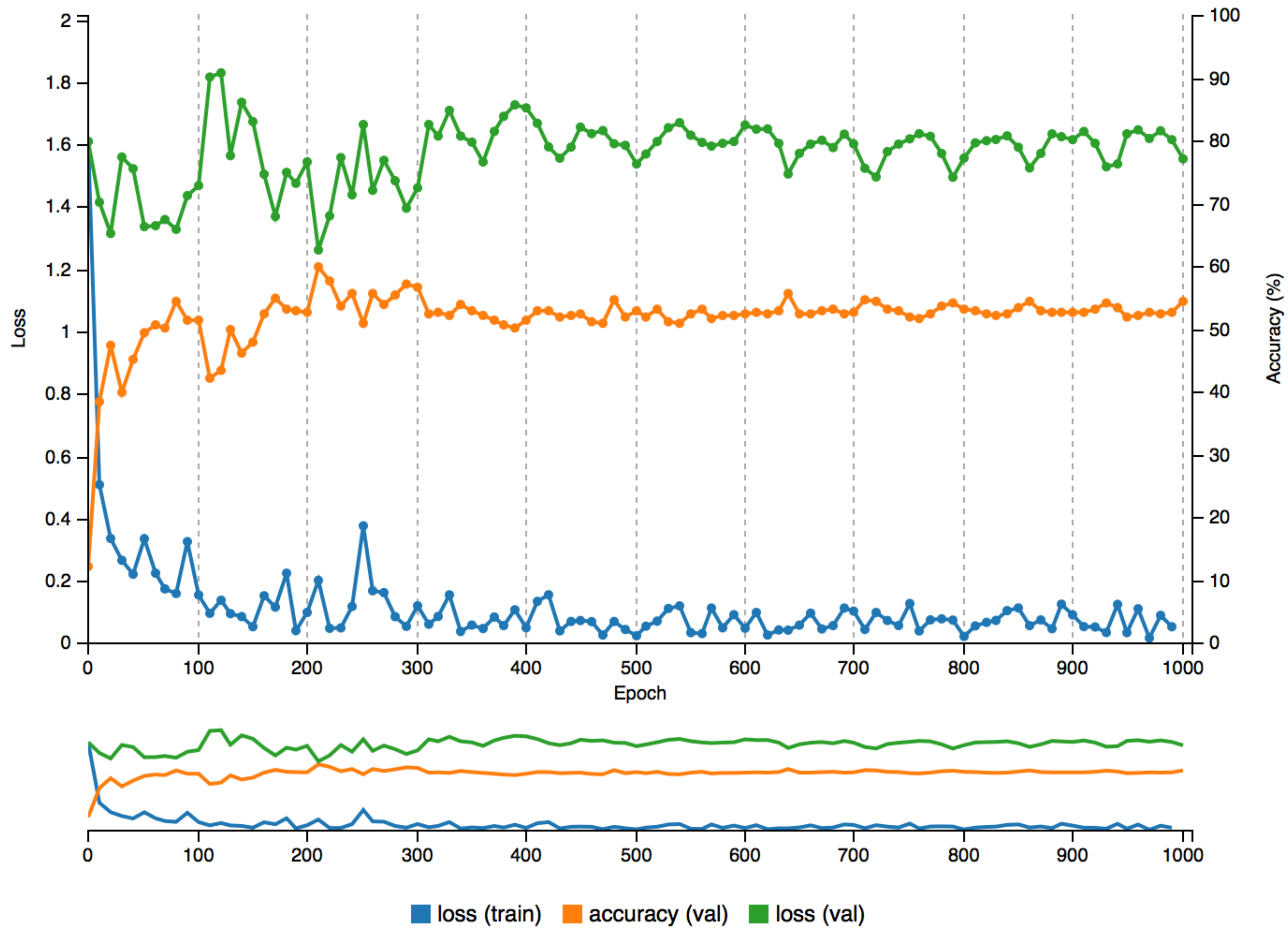
1) Road lane classifier
2) Traffic light object detector

Road Lane Classifier

Using standard AlexNet



Using pre-trained DNN



Conclusion on Road Lanes

Learned context, but didn't care about
the road lanes themselves

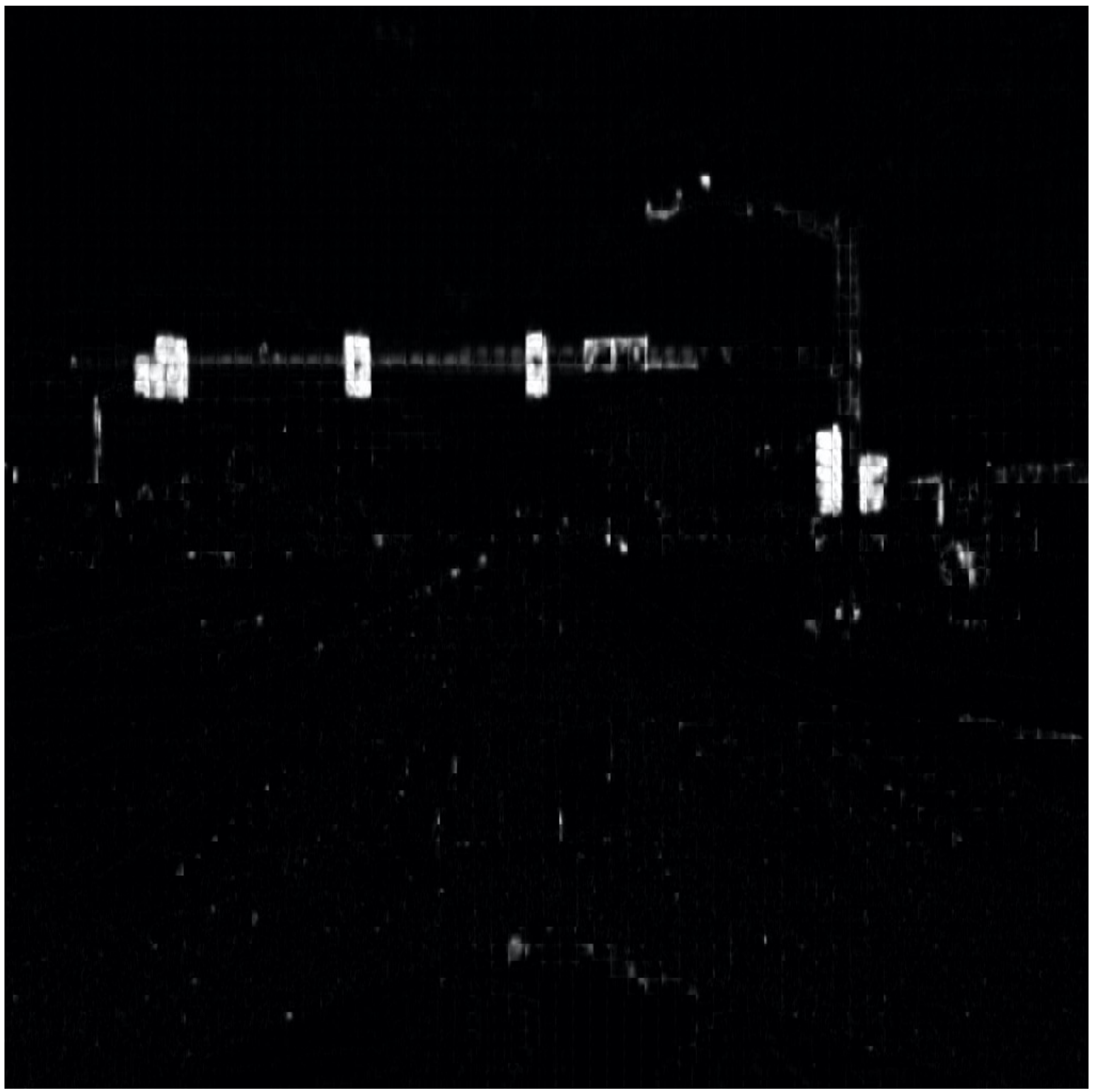
Traffic Light Detector

Simple detector: architecture

input image ⟶ heat map

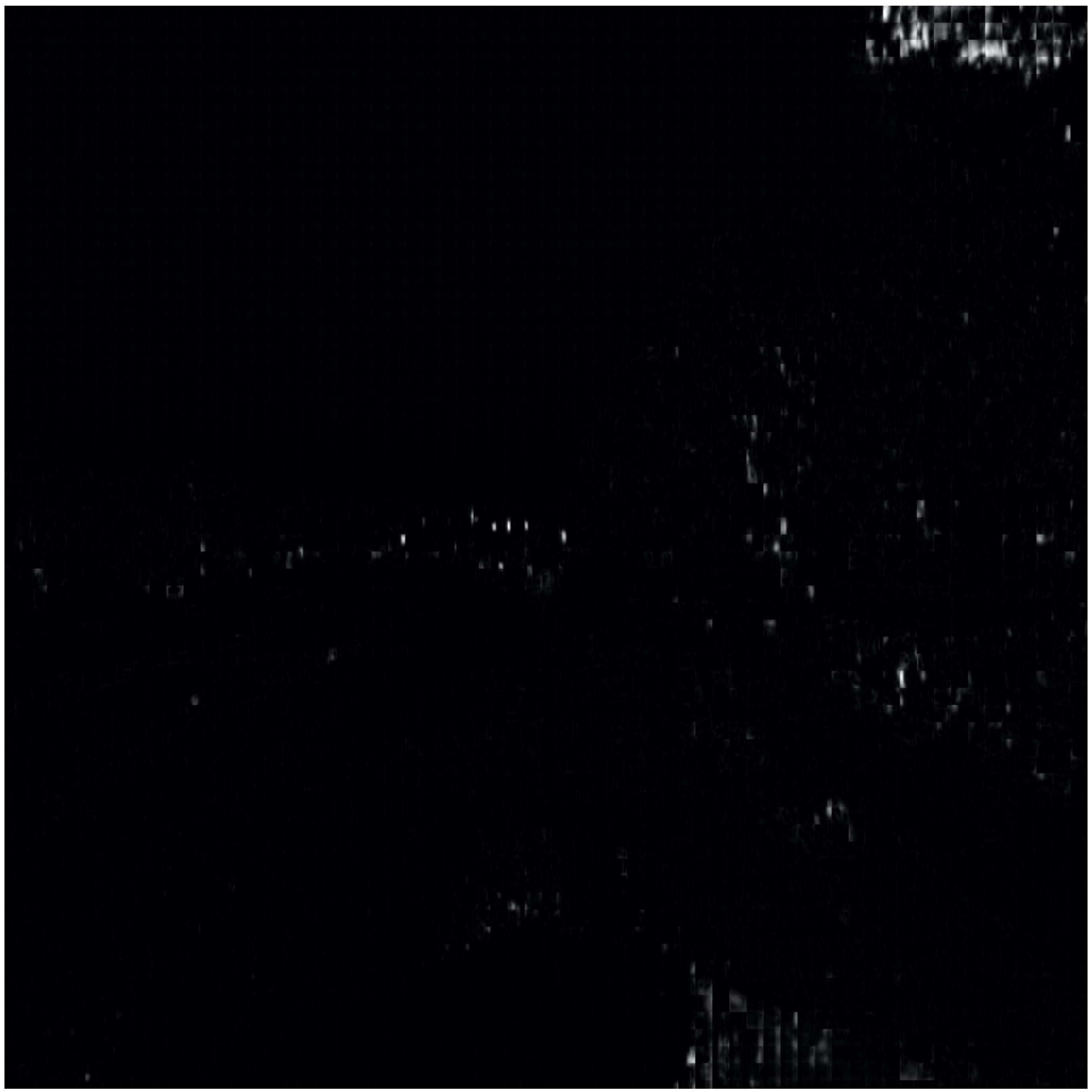
A. 2 CONV (7x7, stride 1)
B. 1POOL(5x5, stride 2)
C. 3 CONV (3x3, stride 1)
D. 1 POOL (5x5, stride 2)
E. 3 CONV (3x3, stride 1)
F. 1 heat map

Sample response (close distance)



Sample response (medium distance)


Sample response (far distance)



(deleted for IP protection)

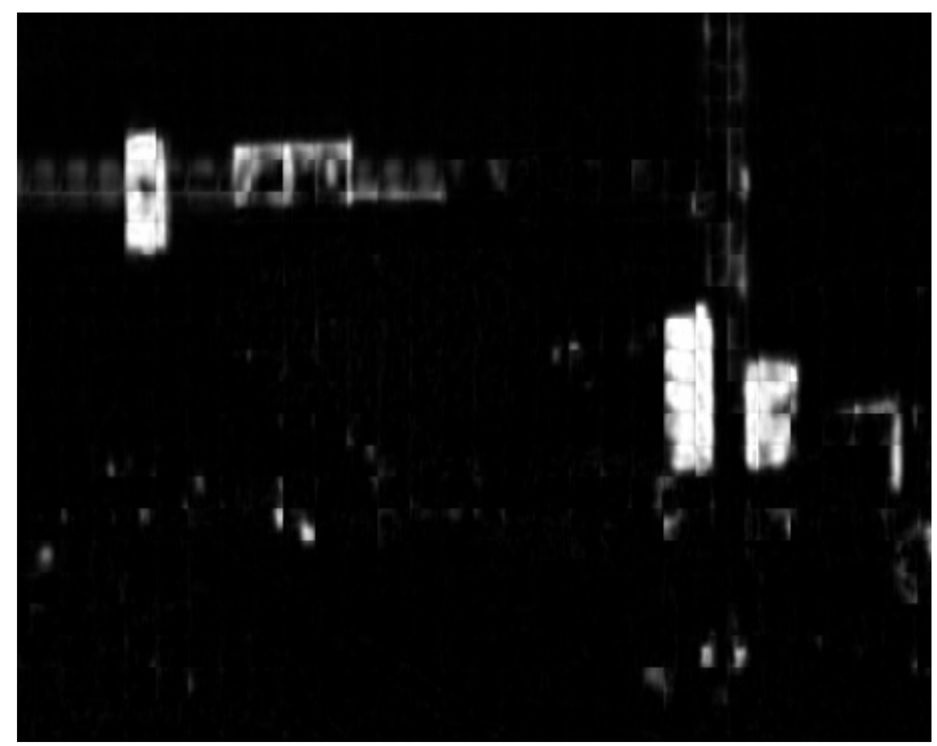
Summer Conclusion

1) Learned Spark
2) Learned industry deep learning practices
3) Read a ton of papers and learned a ton about neural nets

Lunch & Learn

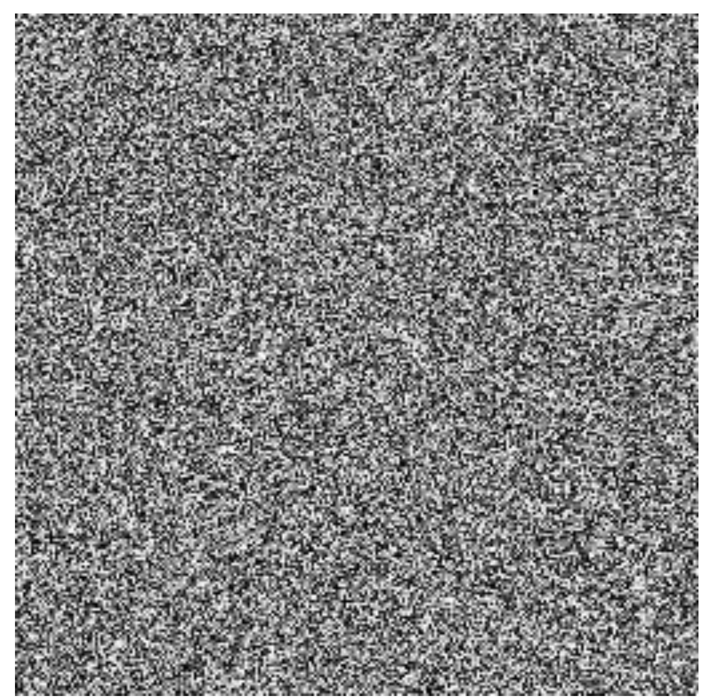
Training DNN is hard

randomly initialized filters → sensible interpretation of images

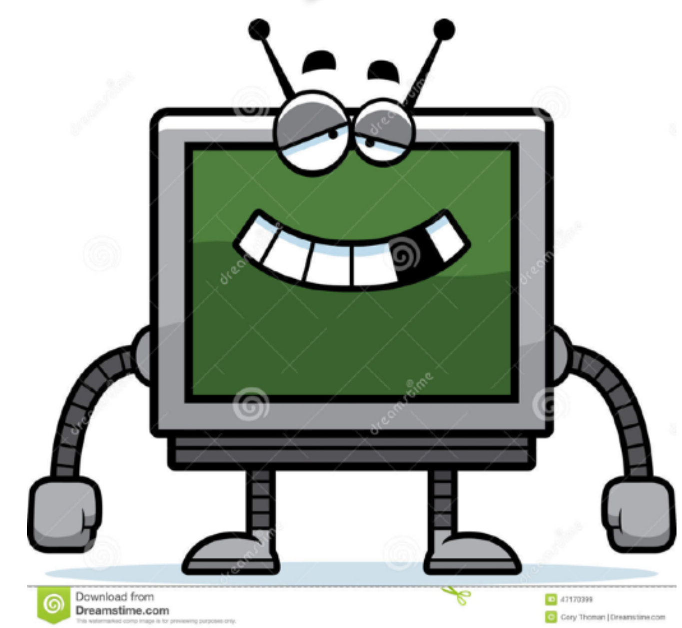


Training DNN is hard

Here, you see a traffic light



Okay…



Invariance in objects

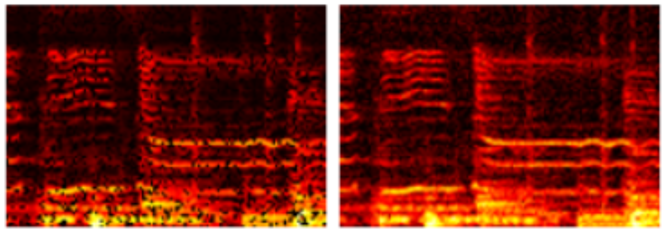
Creating more information

Two ways to teach invariance:

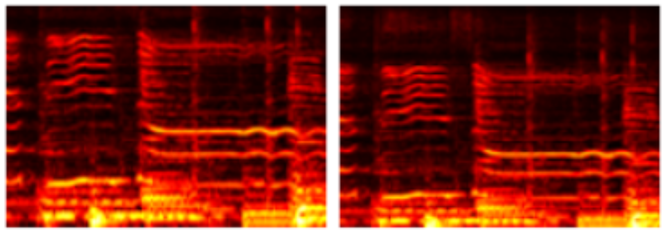
1) show examples from different contexts where something we are interested in stays constant
2) artificially ‘augment’ data to simulate different context

Data augmentation in images

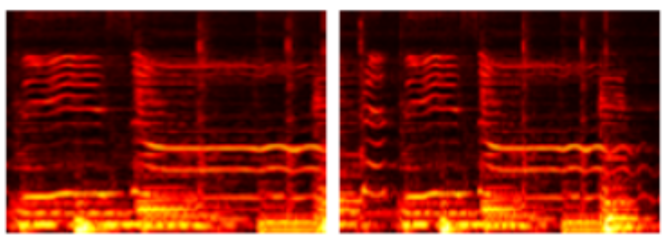
Random(
Crop,
Scale,
Sheer,
Stretch,
Flip,
Noise)



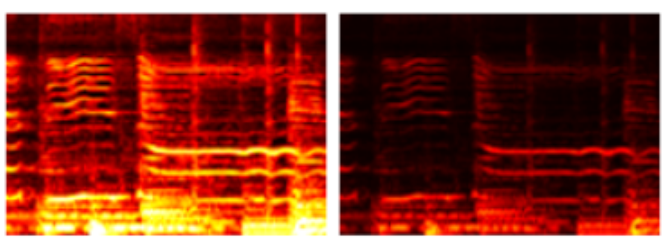
(c) Dropout and Gaussian noise.



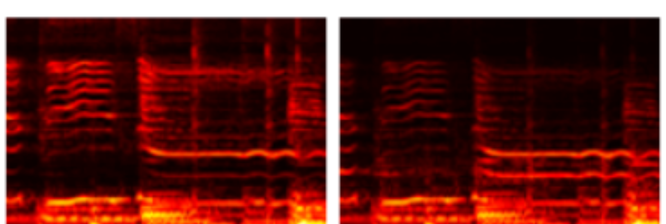
(d) Pitch shift of +/-20%.



(e) Time stretch of +/-20%.



(f) Loudness of +/-10 dB.



(g) Random frequency filters.

Data augmentation in sound

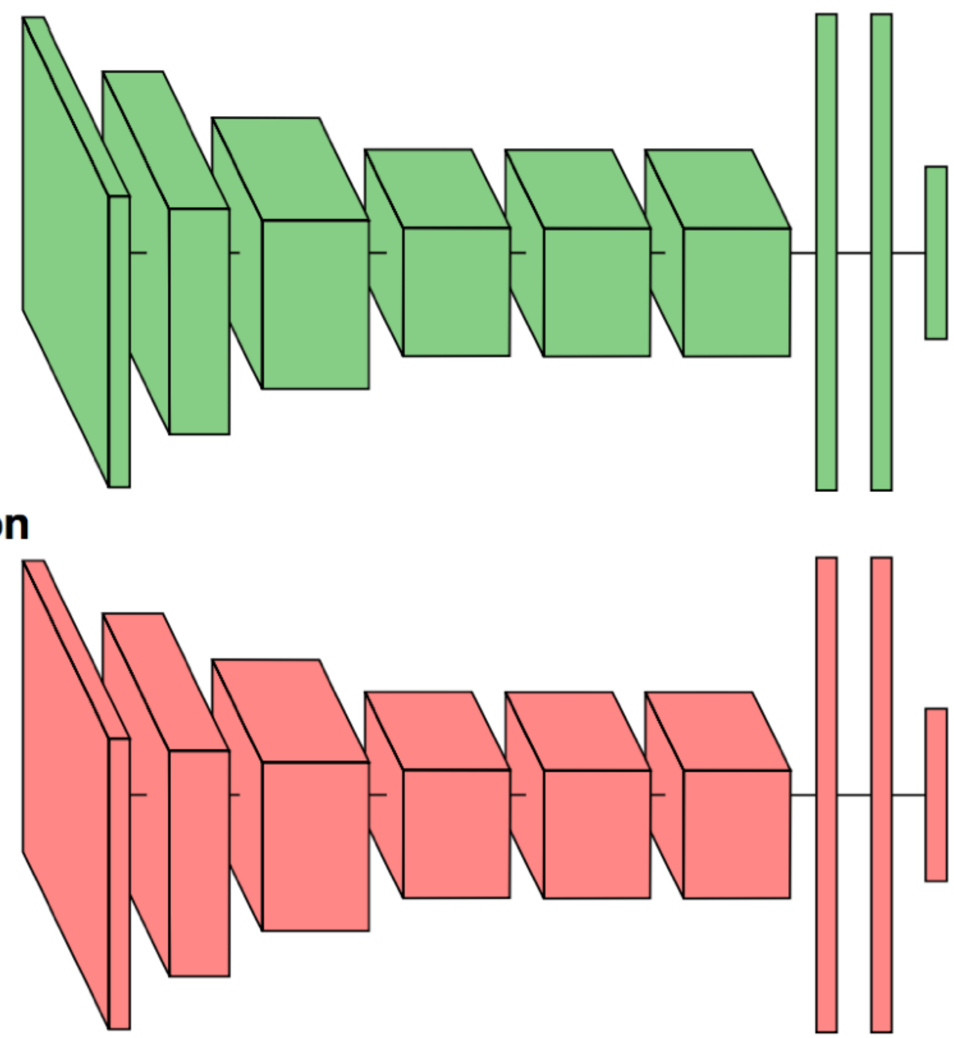
Convergent Learning: Do different neural networks learn the same representations?

Yixuan Li[1,*] **Jason Yosinski**[1,*] **Jeff Clune**[2], **Hod Lipson**[3], **& John Hopcroft**[1]
[1]Cornell University
[2]University of Wyoming
[3]Columbia University
{yli,yosinski,jeh}@cs.cornell.edu
jeffclune@uwyo.edu, hod.lipson@columbia.edu

Features

Same architecture

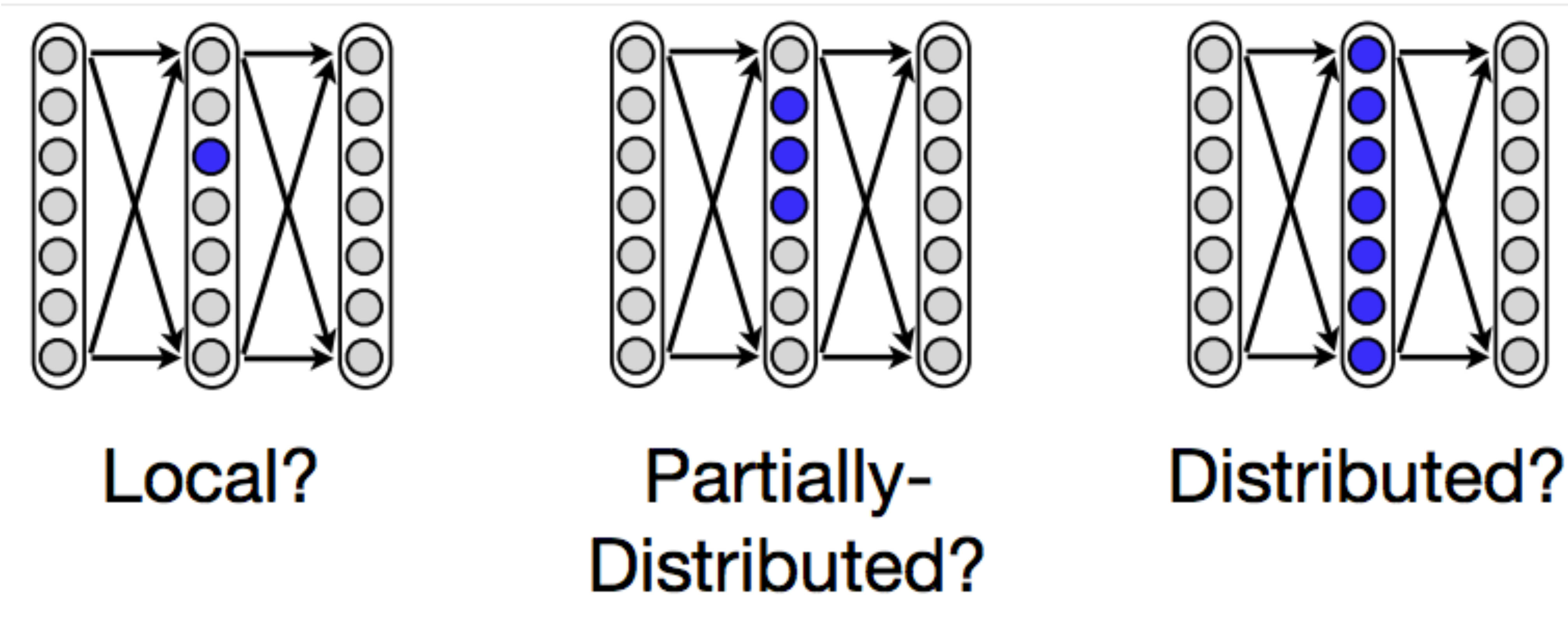
Different random initialization



Net1 58.65%

Net2 58.73%

Which one is it?



Measuring correlation & mutual information

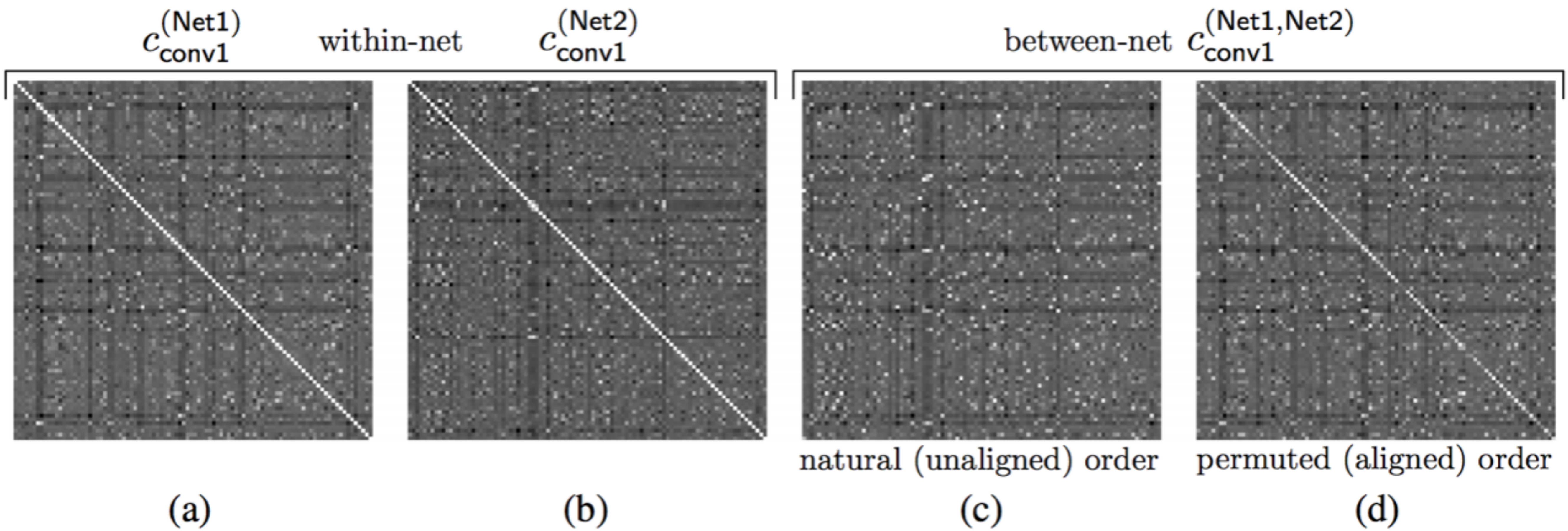
i,j - units, l - layer, X - series of activation values

$$
\begin{aligned}
\textbf{Mean:} \quad \mu_{l,i}^{(n)} &= \mathbb{E}[X_{l,i}^{(n)}] \\
\textbf{Standard deviation:} \quad \sigma_{l,i}^{(n)} &= \sqrt{(}\mathbb{E}[(X_{l,i}^{(n)} - \mu_{l,i}^{(n)})^2]) \\
\textbf{Within-net correlation:} \quad c_{l,i,j}^{(n)} &= \mathbb{E}[(X_{l,i}^{(n)} - \mu_{l,i}^{(n)})(X_{l,j}^{(n)} - \mu_{l,j}^{(n)})]/\sigma_{l,i}^{(n)}\sigma_{l,j}^{(n)} \\
\textbf{Between-net correlation:} \quad c_{l,i,j}^{(n,m)} &= \mathbb{E}[(X_{l,i}^{(n)} - \mu_{l,i}^{(n)})(X_{l,j}^{(m)} - \mu_{l,j}^{(m)})]/\sigma_{l,i}^{(n)}\sigma_{l,j}^{(m)}
\end{aligned}
$$

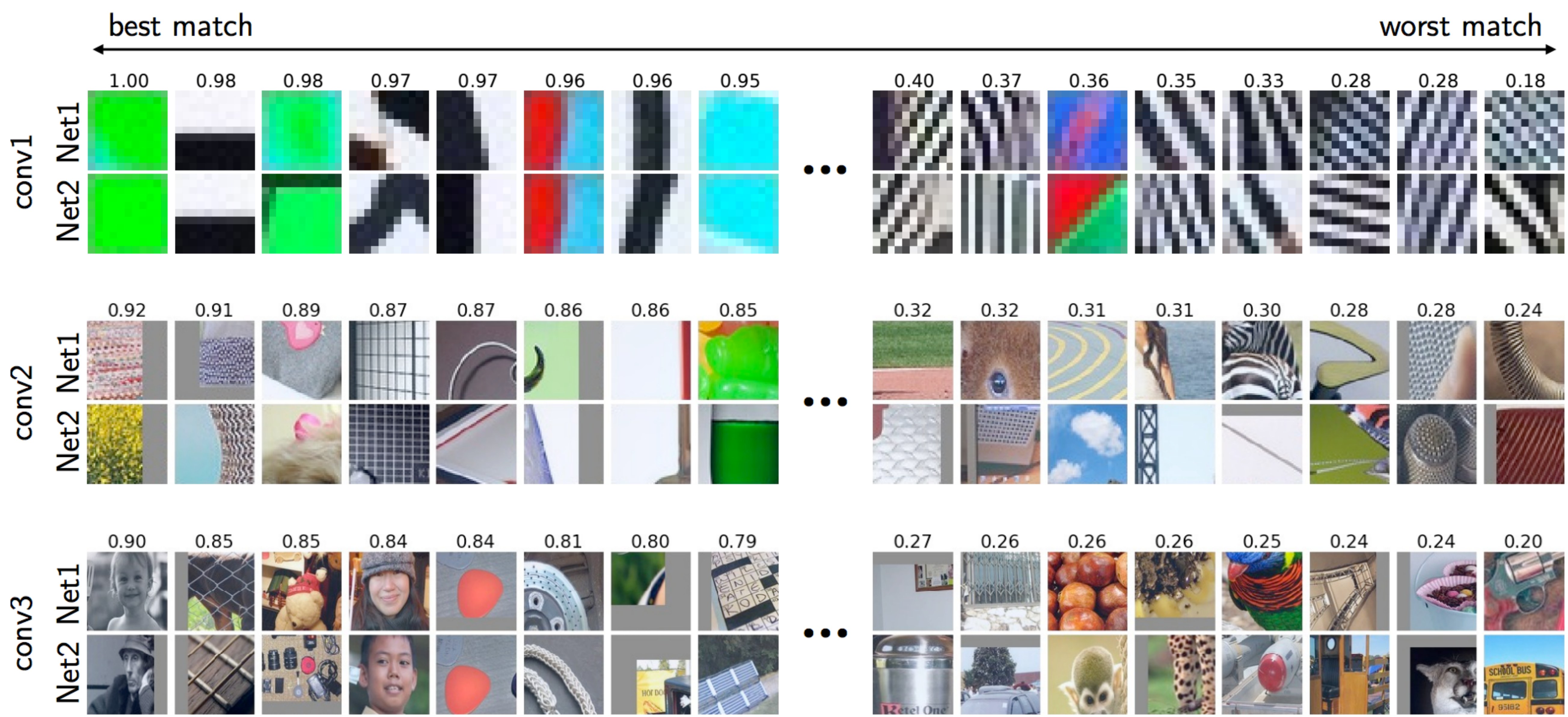
Mutual information:

$$
I\left(X_{l,i}^{(n)}; X_{l,j}^{(m)}\right) = \sum_{a \in X_{l,i}^{(n)}} \sum_{b \in X_{l,j}^{(m)}} p(a,b) \log\left(\frac{p(a,b)}{p(a)p(b)}\right),
$$

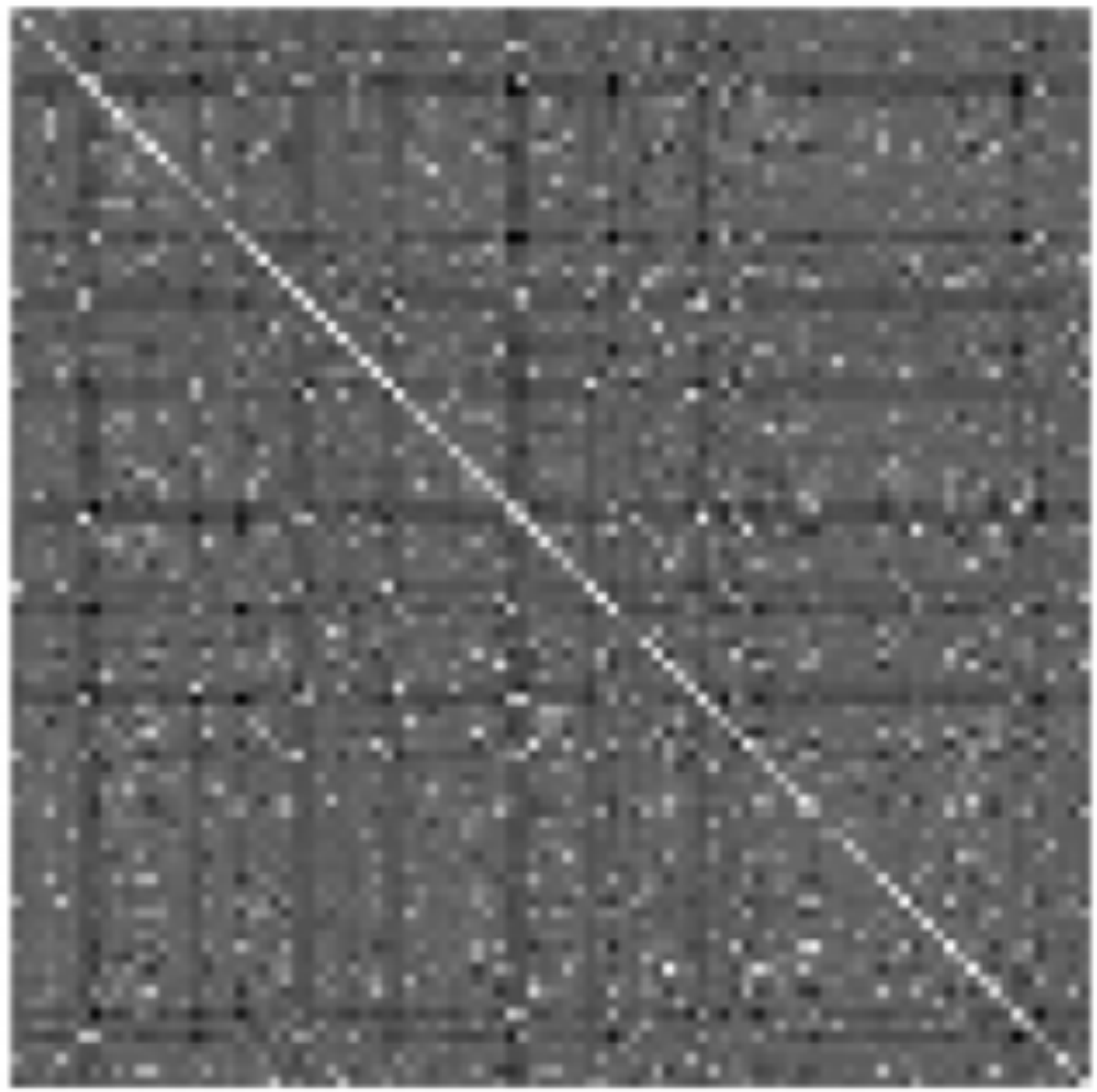
Neuron - Neuron Correlation

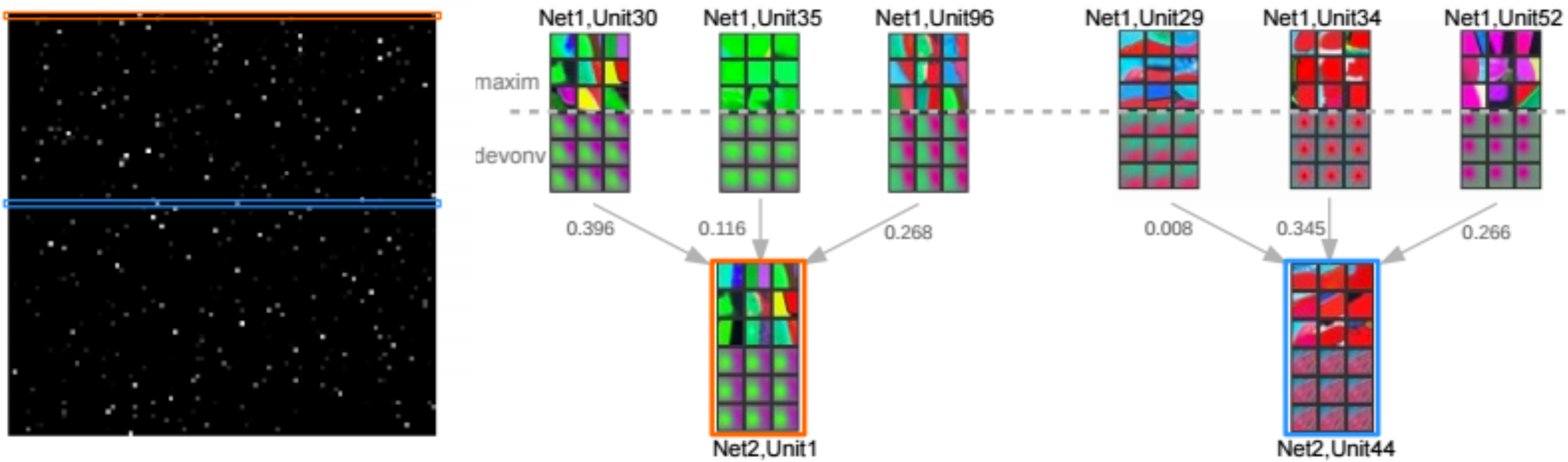


Common filters - chosen greedily



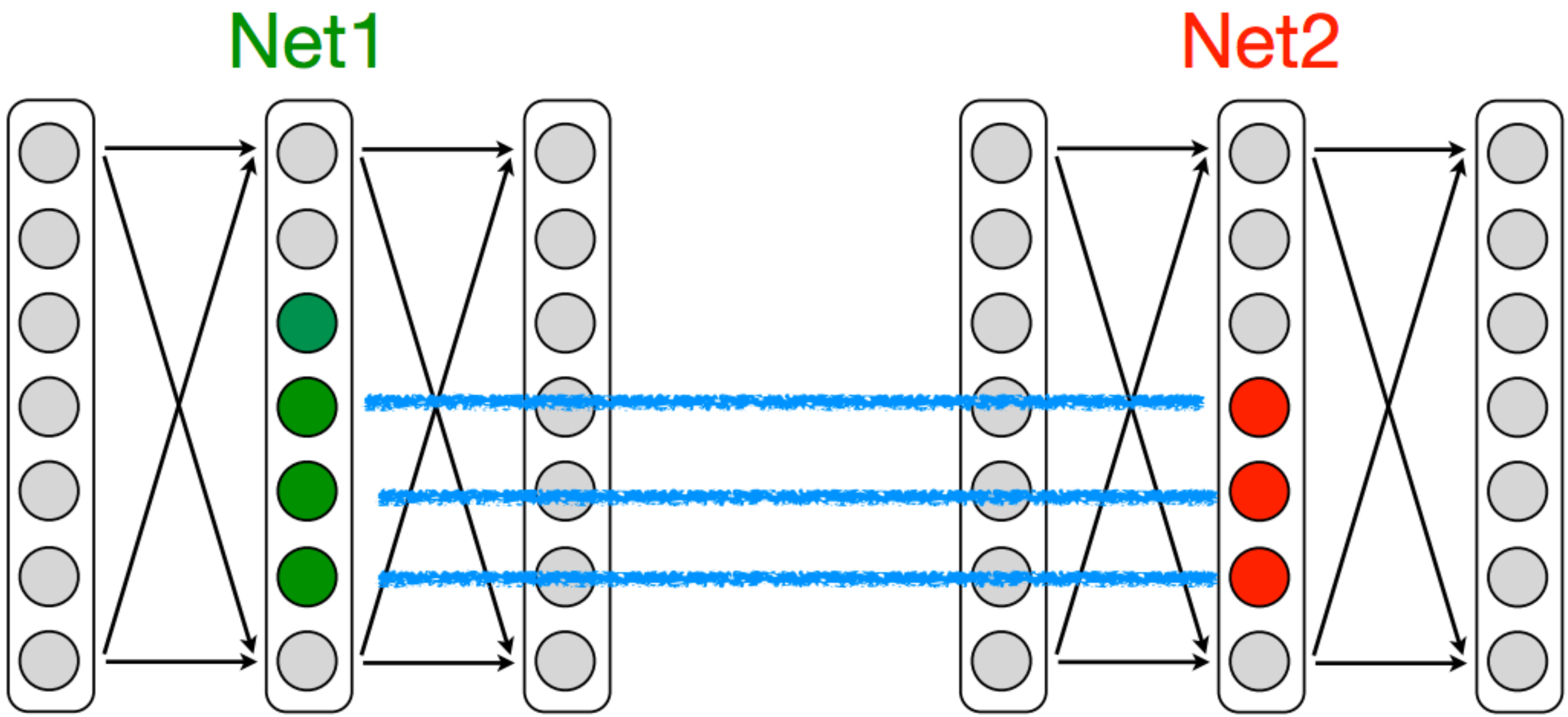
What about non diagonal entries?

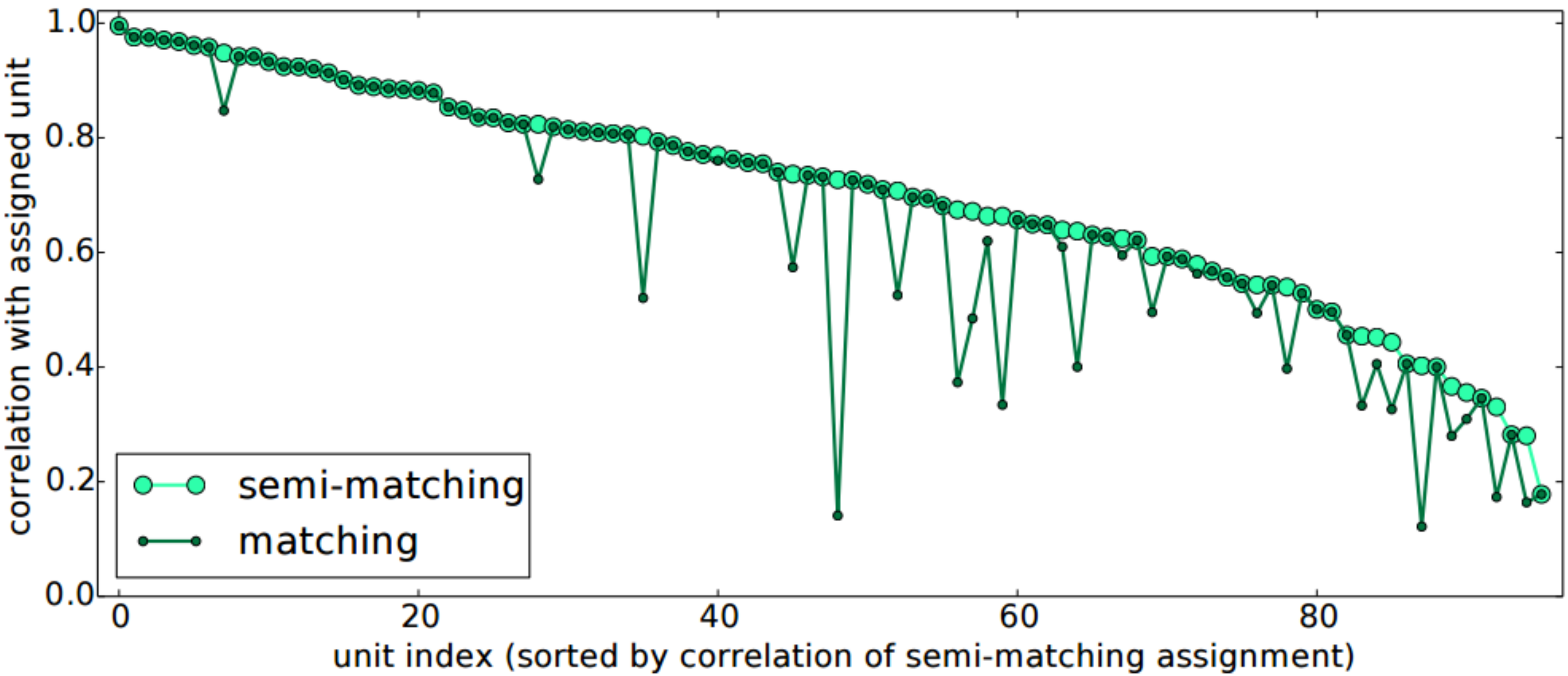




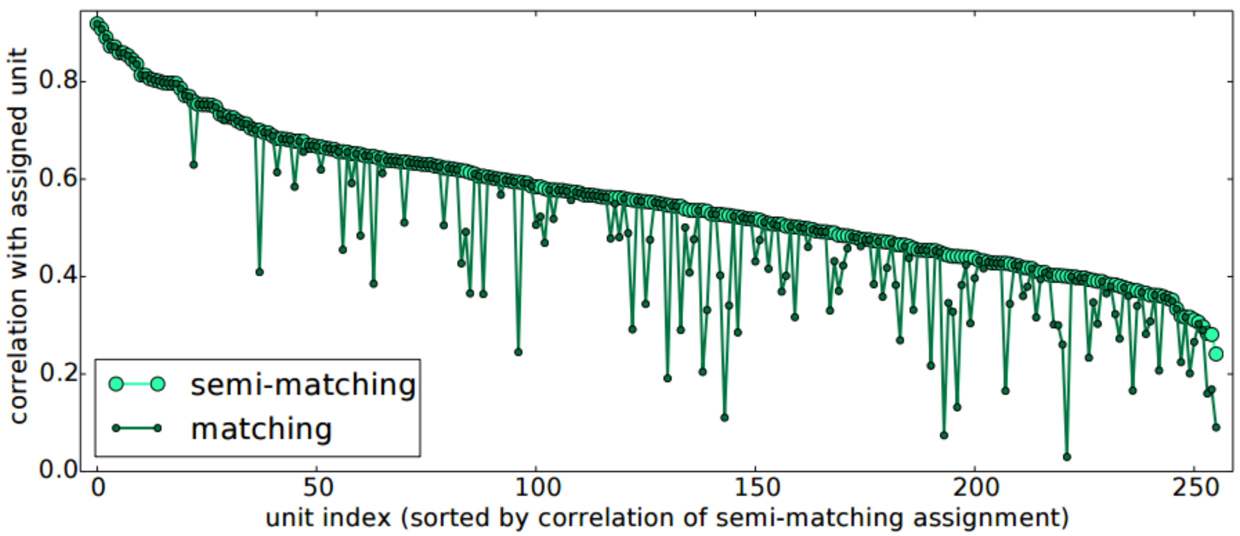
Net1,Unit30
Net1,Unit35
Net1,Unit96
Net1,Unit29
Net1,Unit34
Net1,Unit52
maxim
devonv
0.396
0.116
0.268
0.008
0.345
0.266
Net2,Unit1
Net2,Unit44

Common filters - chosen by weighted bipartite matching

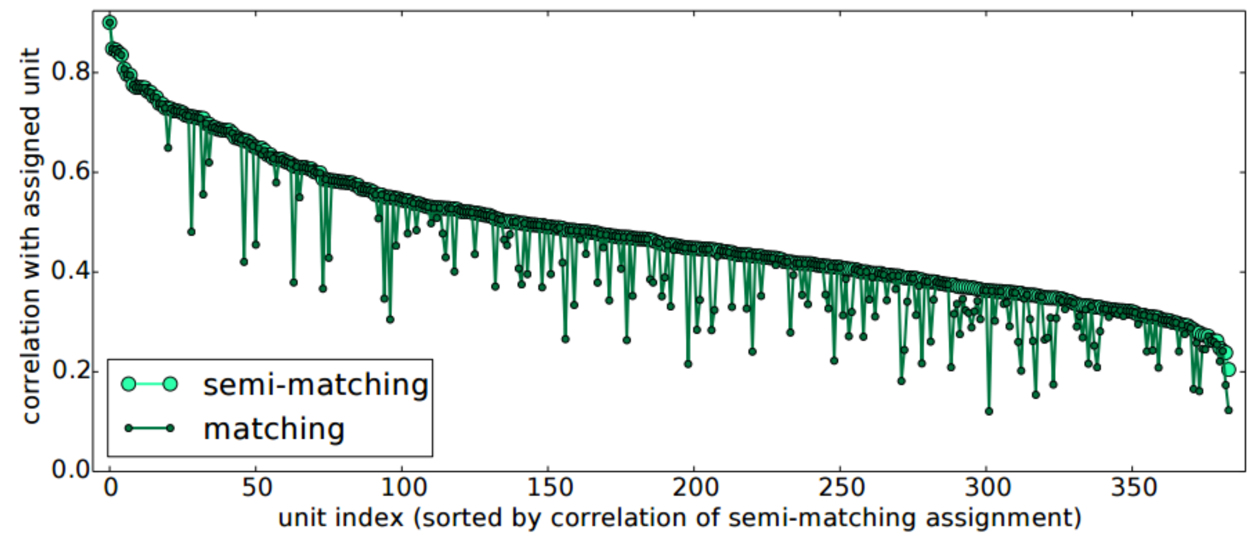




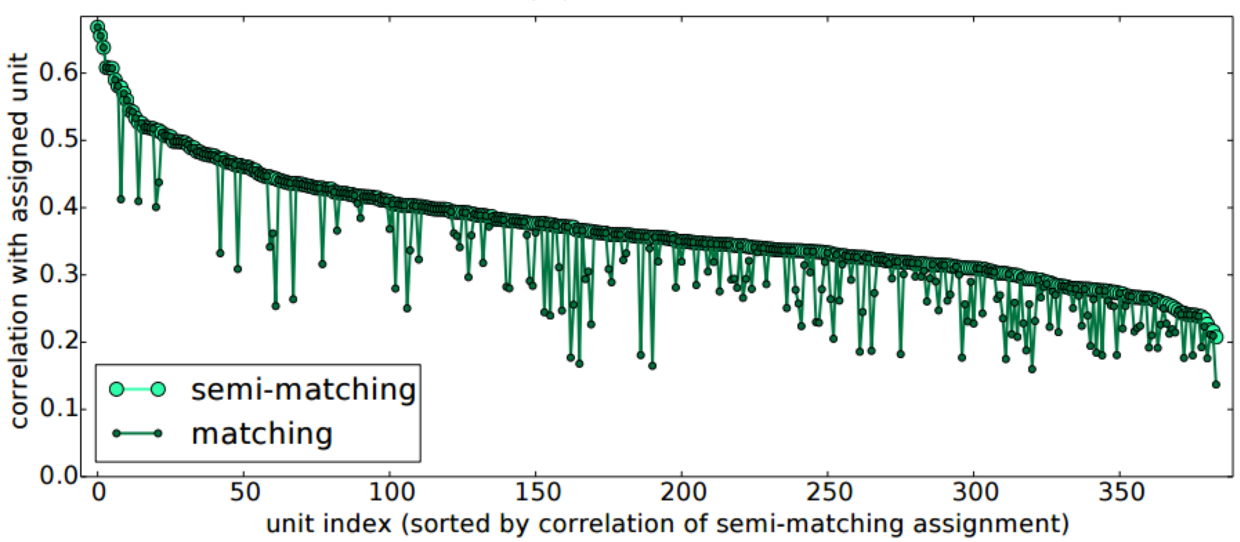
Takeaway: Some units didn't get a match



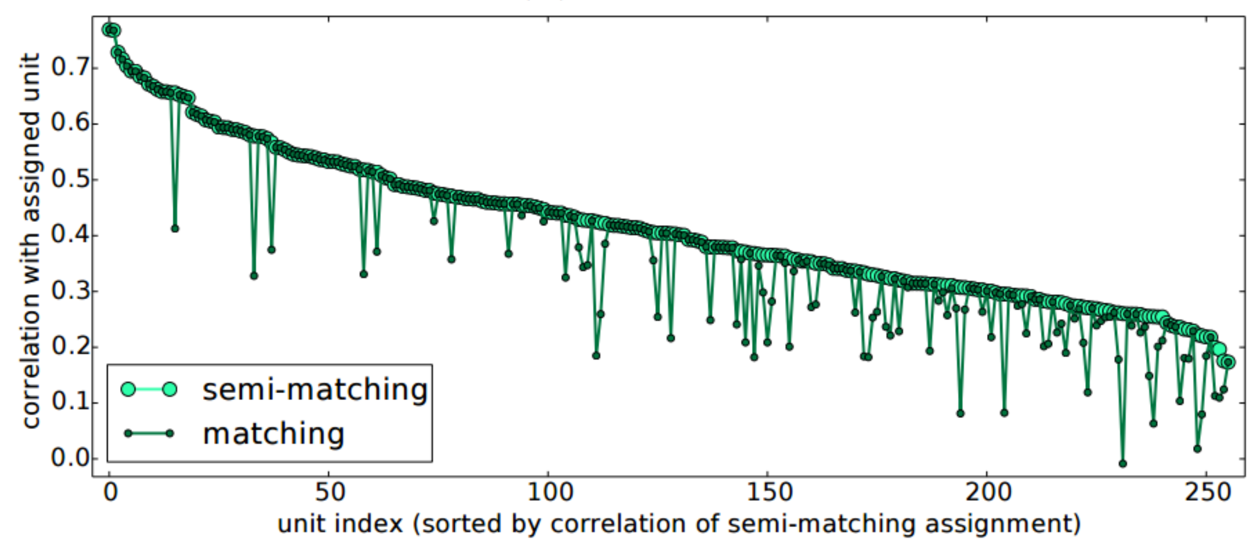
(a) conv2



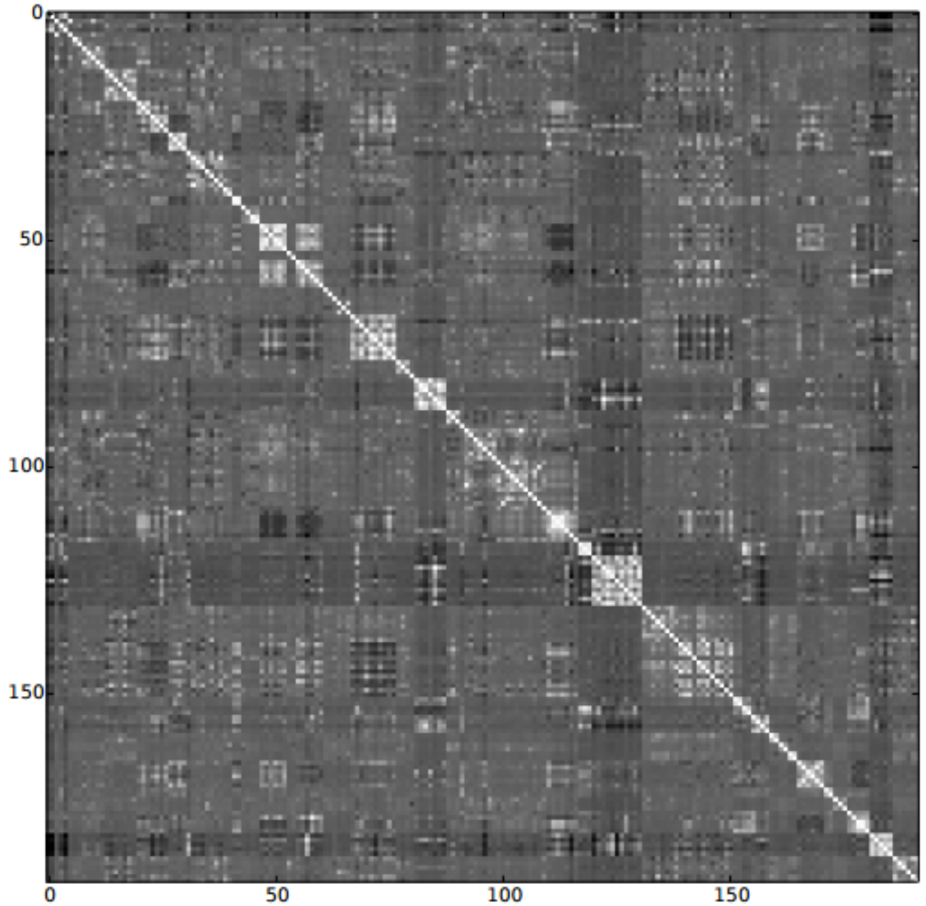
(b) conv3



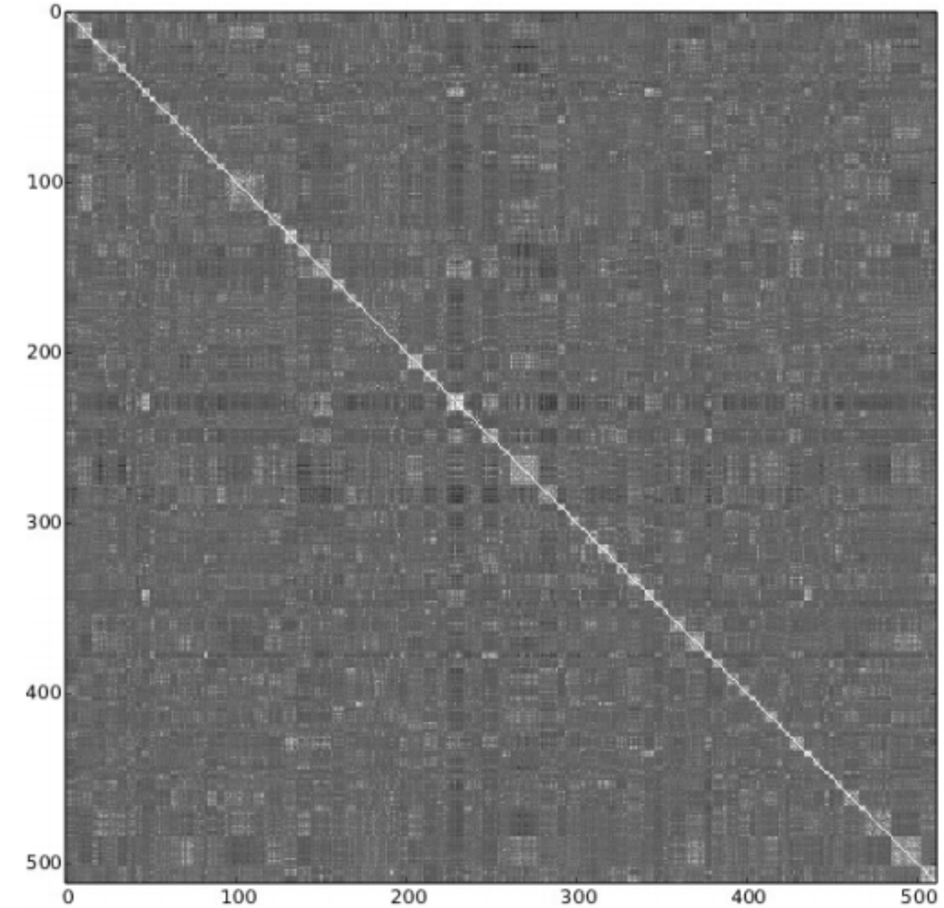
(c) conv4



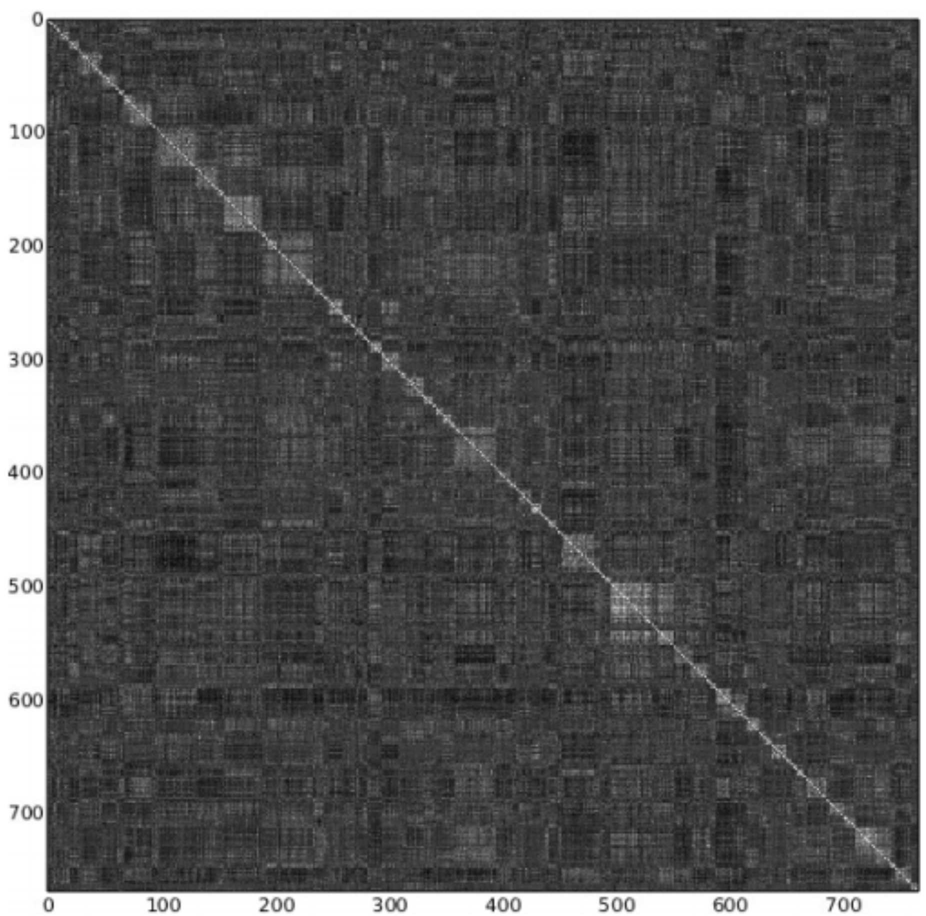
(d) conv5



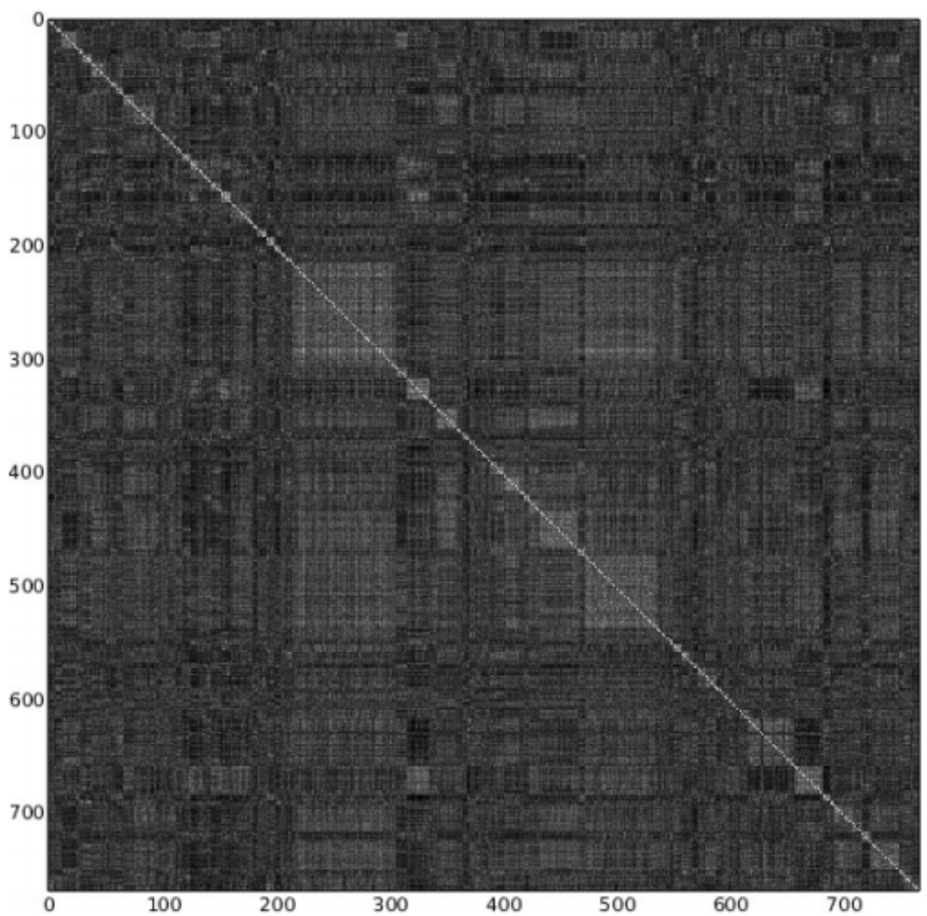
(a) conv1



(b) conv2



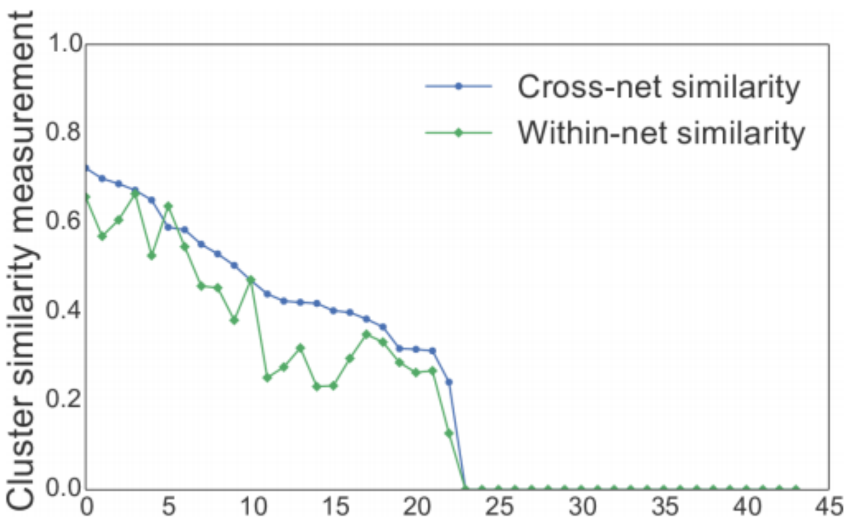
(c) conv3



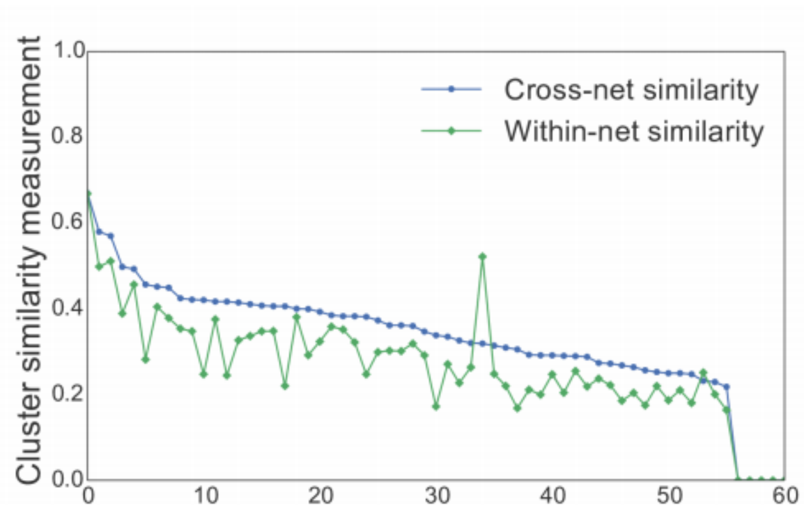
(d) conv4

$$\textbf{Between-net similarity:} \quad \mathrm{Sim}_{X_l \to Y_l} = \Big(\sum_{p=1}^{S_l}\sum_{q=1}^{S_l} \mathrm{corr}(X_l, Y_l)_{pq}\Big)/S_l^2$$

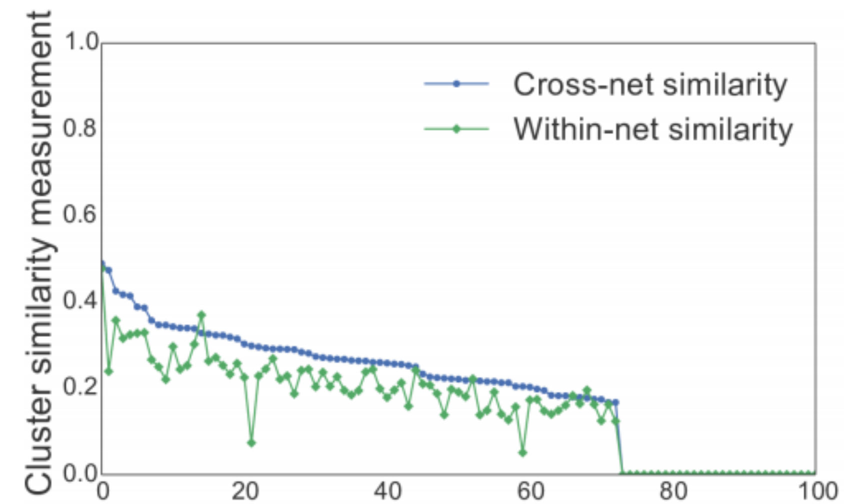
$$\textbf{Within-net similarity:} \quad \mathrm{Sim}_{X_l, Y_l} = (\mathrm{Sim}_{X_l \to X_l} + \mathrm{Sim}_{Y_l \to Y_l})/2$$



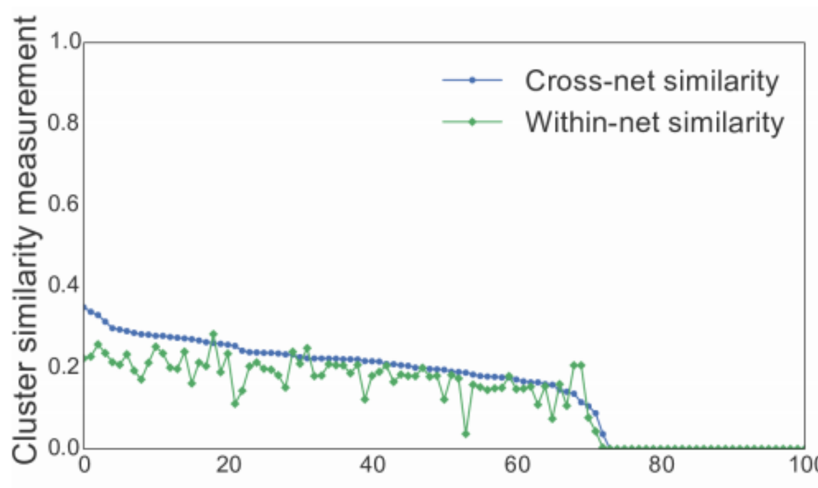
(a) conv1



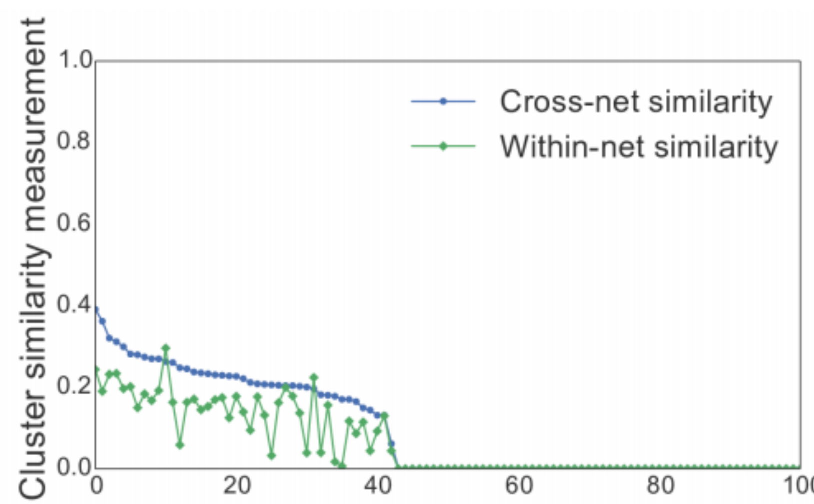
(b) conv2



(c) conv3



(d) conv4



(e) conv5

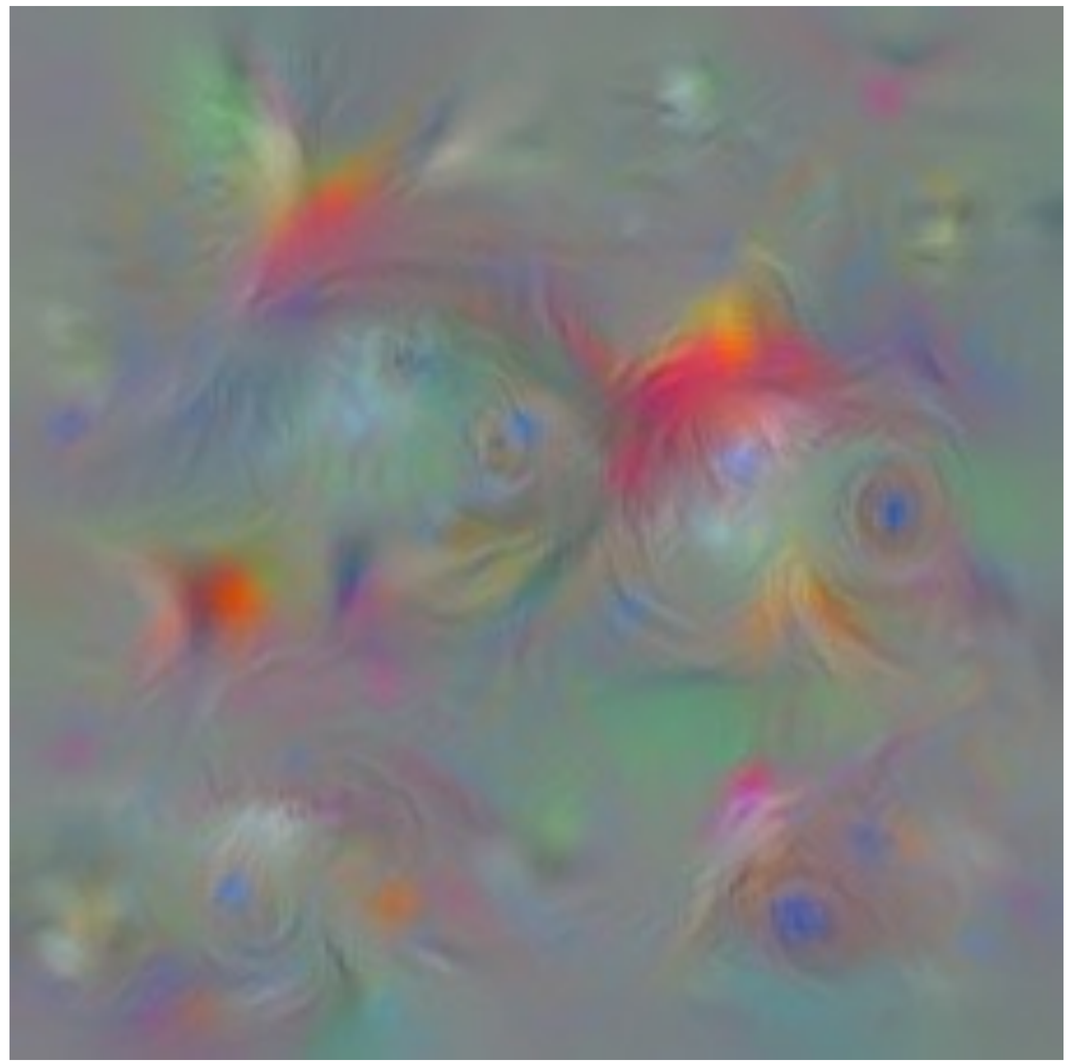
Figure S9: The distribution of between-net and within-net similarity measurement after clustering neurons (conv1 – conv5). The x-axis represents obtained clusters, which is reshuffled according to the sorted between-net similarity value.

Transferring features by ‘pre-training’

*side knowledge: layer maximization

1) Input random noise
2) Maximize (steepest ascent) a chosen layer with respect to input
Result: input image that will maximize a chose layer (aka we see what the layer is “expecting” to see)

Examples



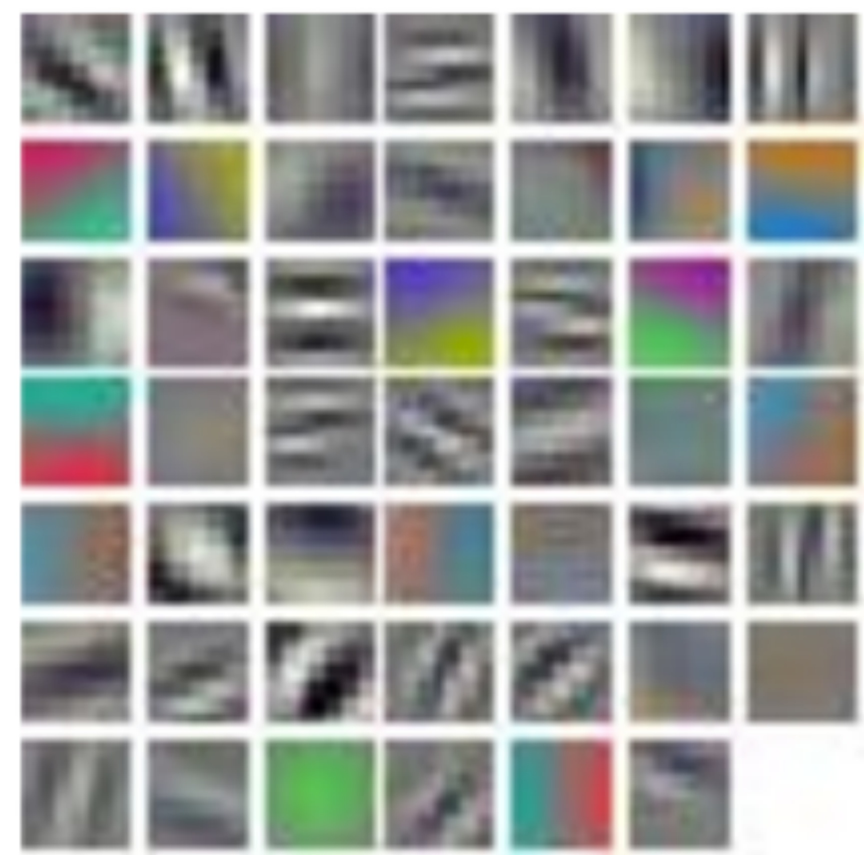
goldfish

Why useful?

Traffic Light DNN classifier

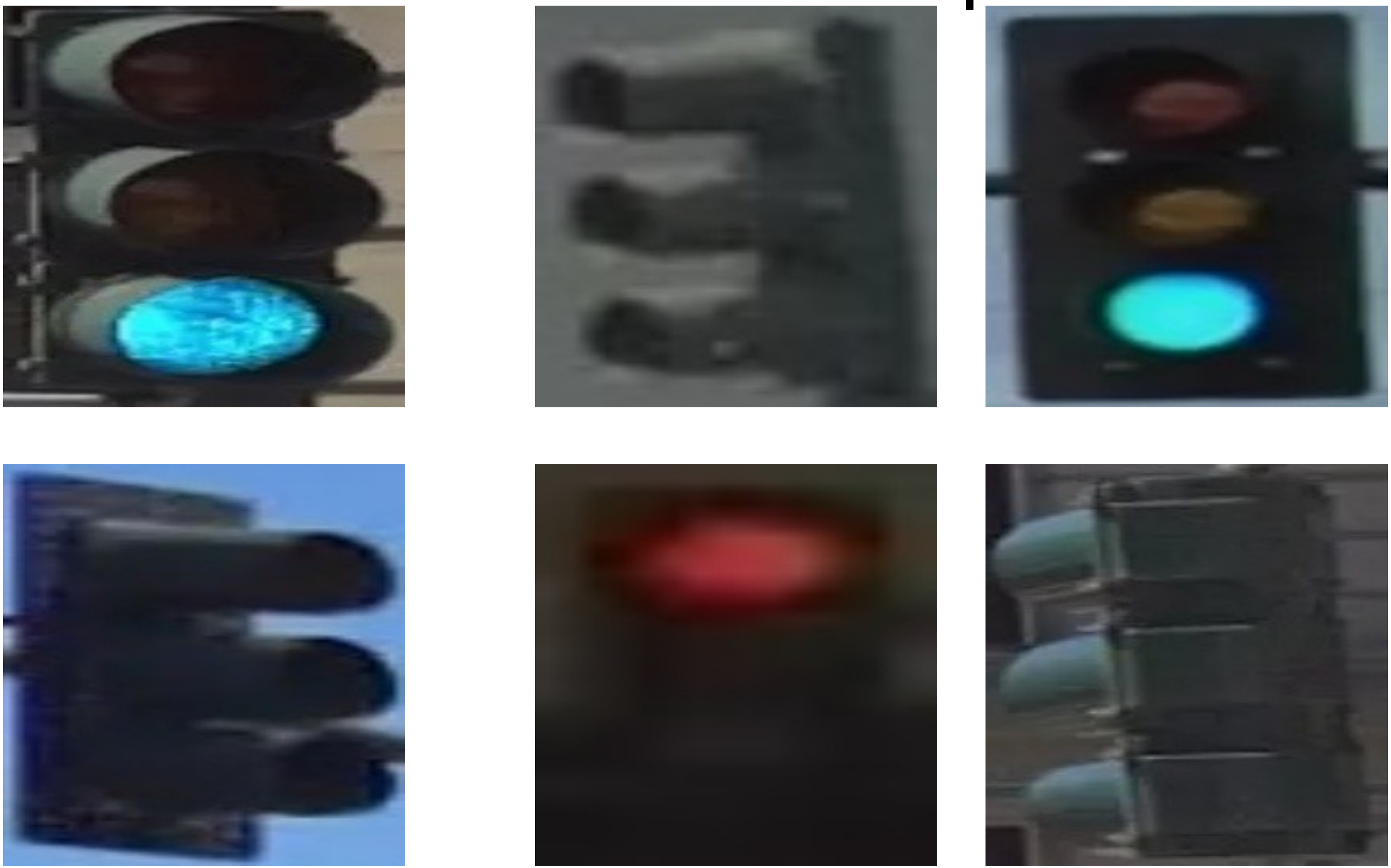
Conv 1 filters

from scratch

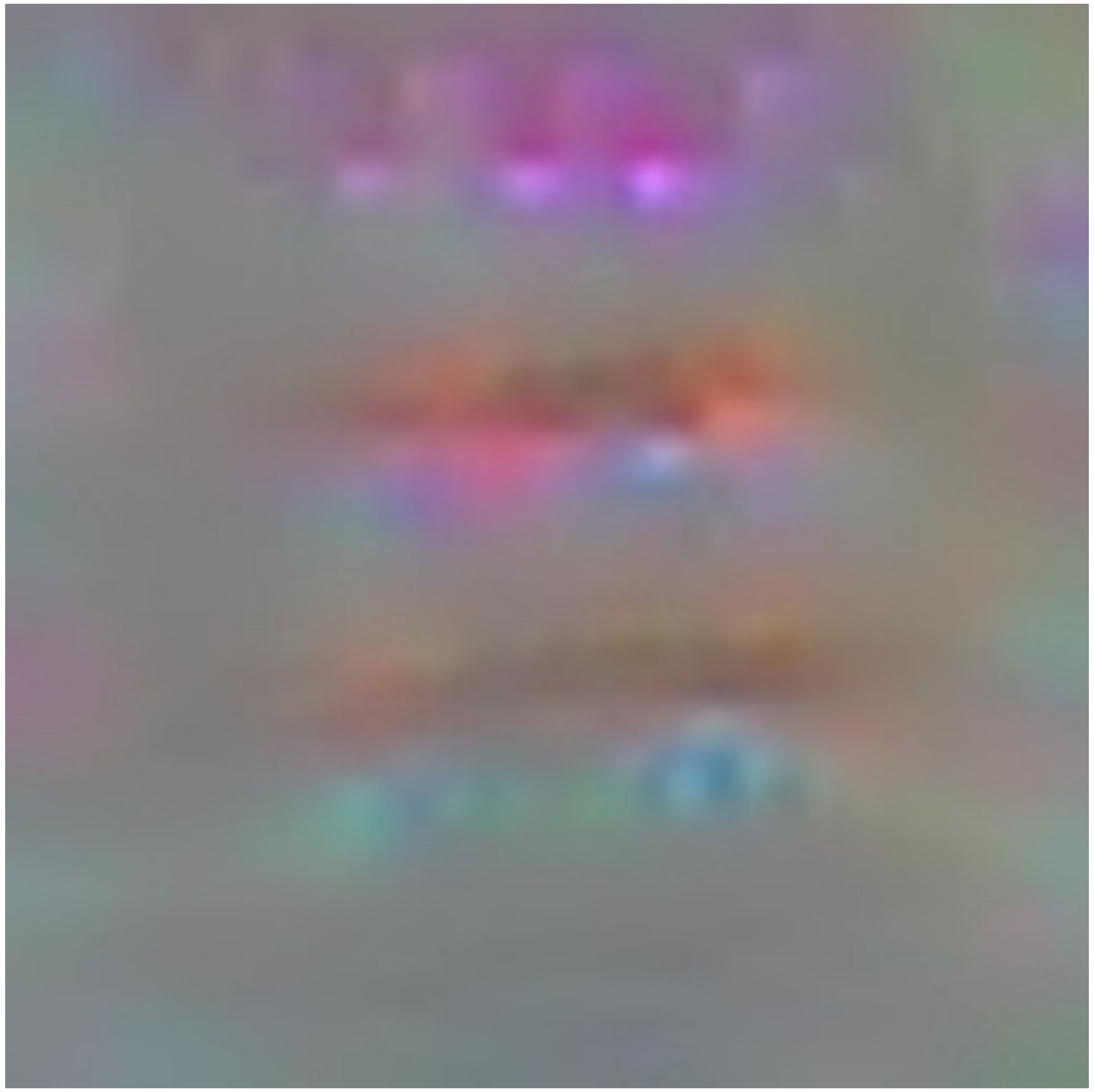


pre-trained

Data Example

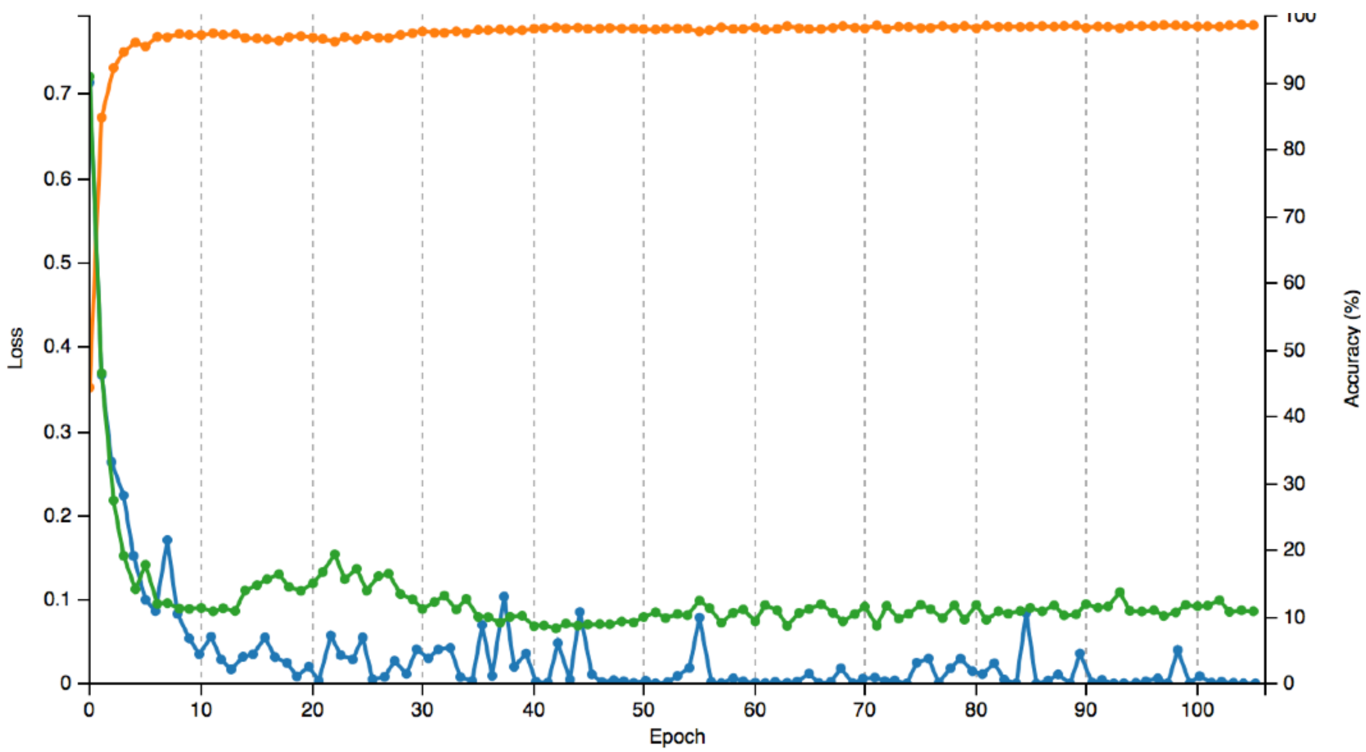


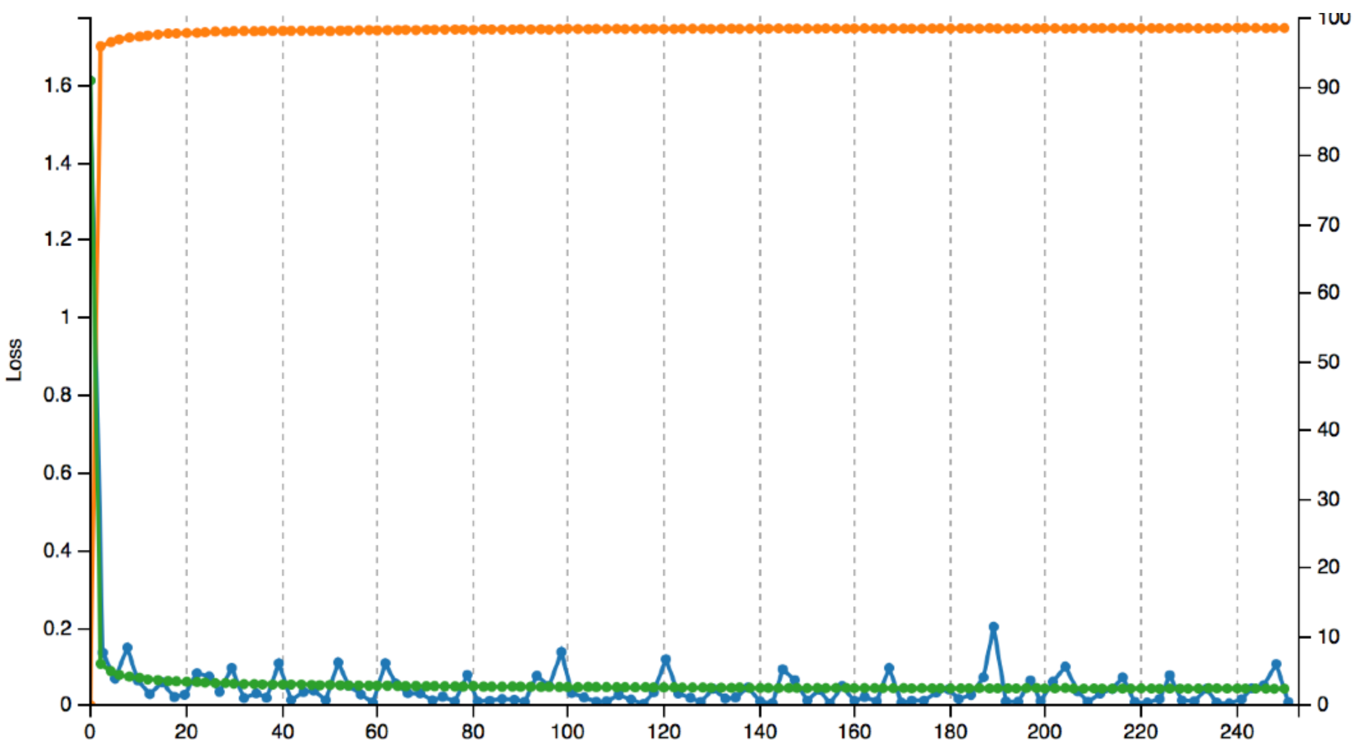
Traffic Light?



Essentially, we found a hack

Loss functions

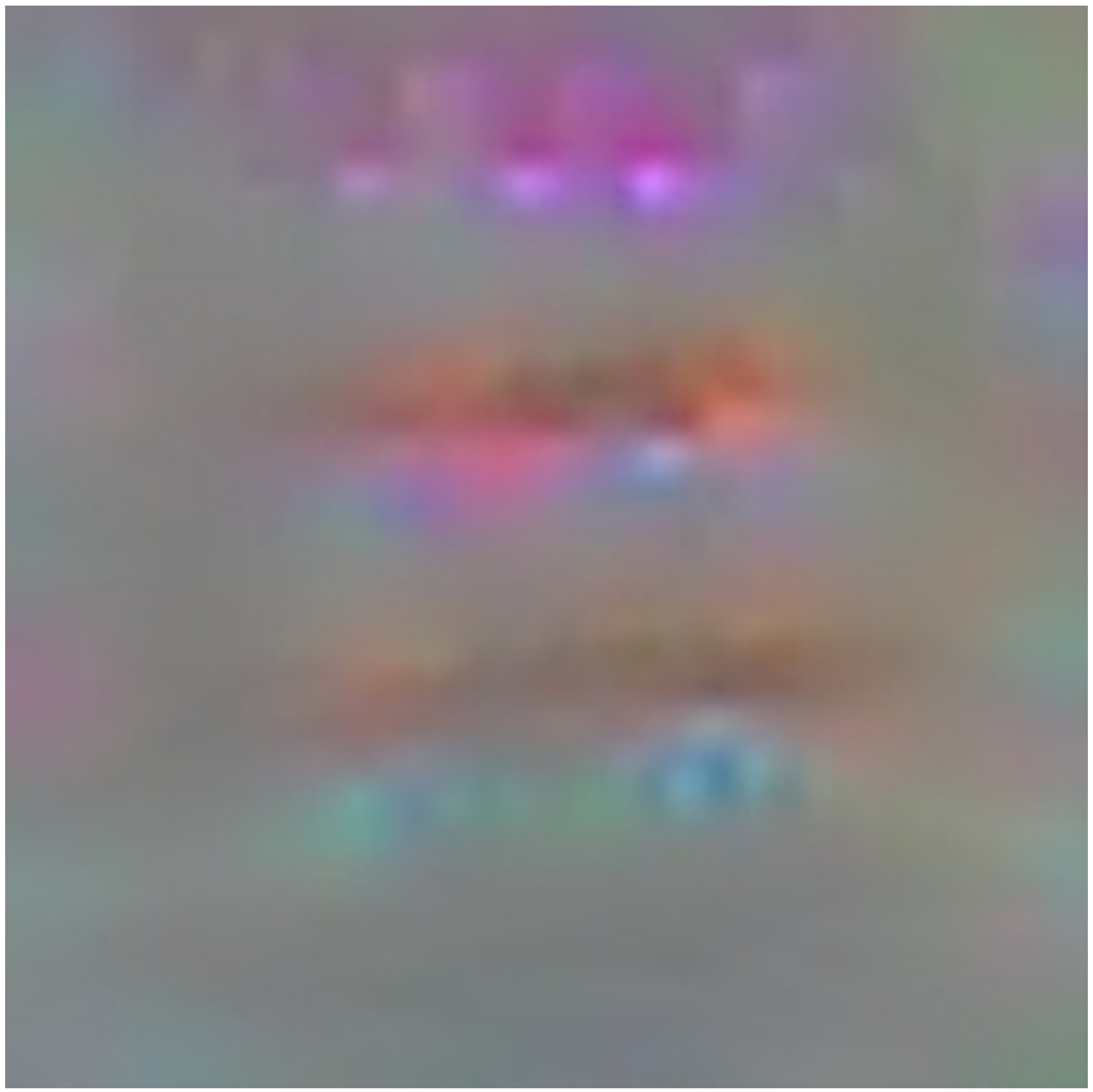




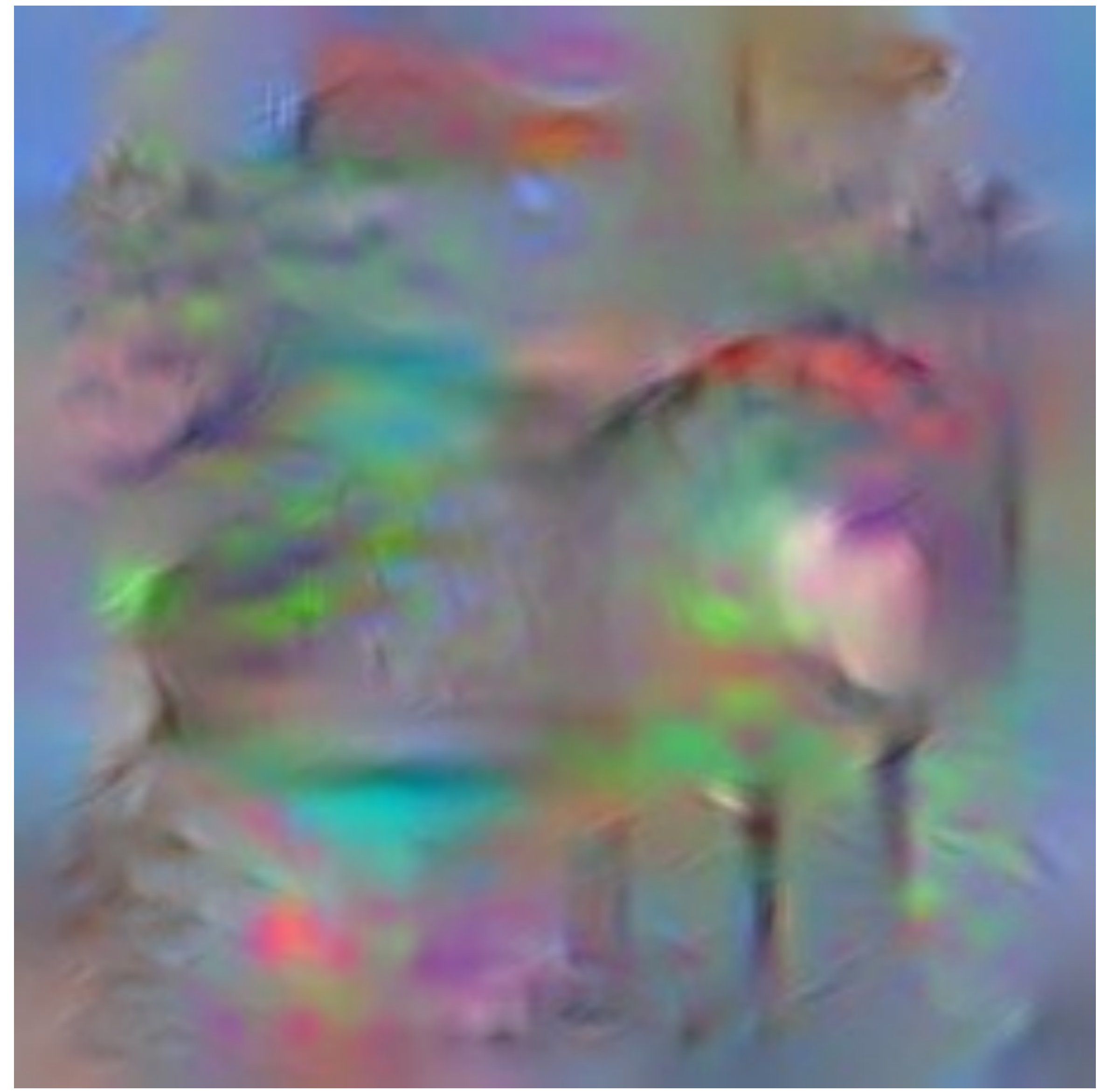
from scratch

pre-trained

‘Prob’ layer (Traffic Light)



from scratch

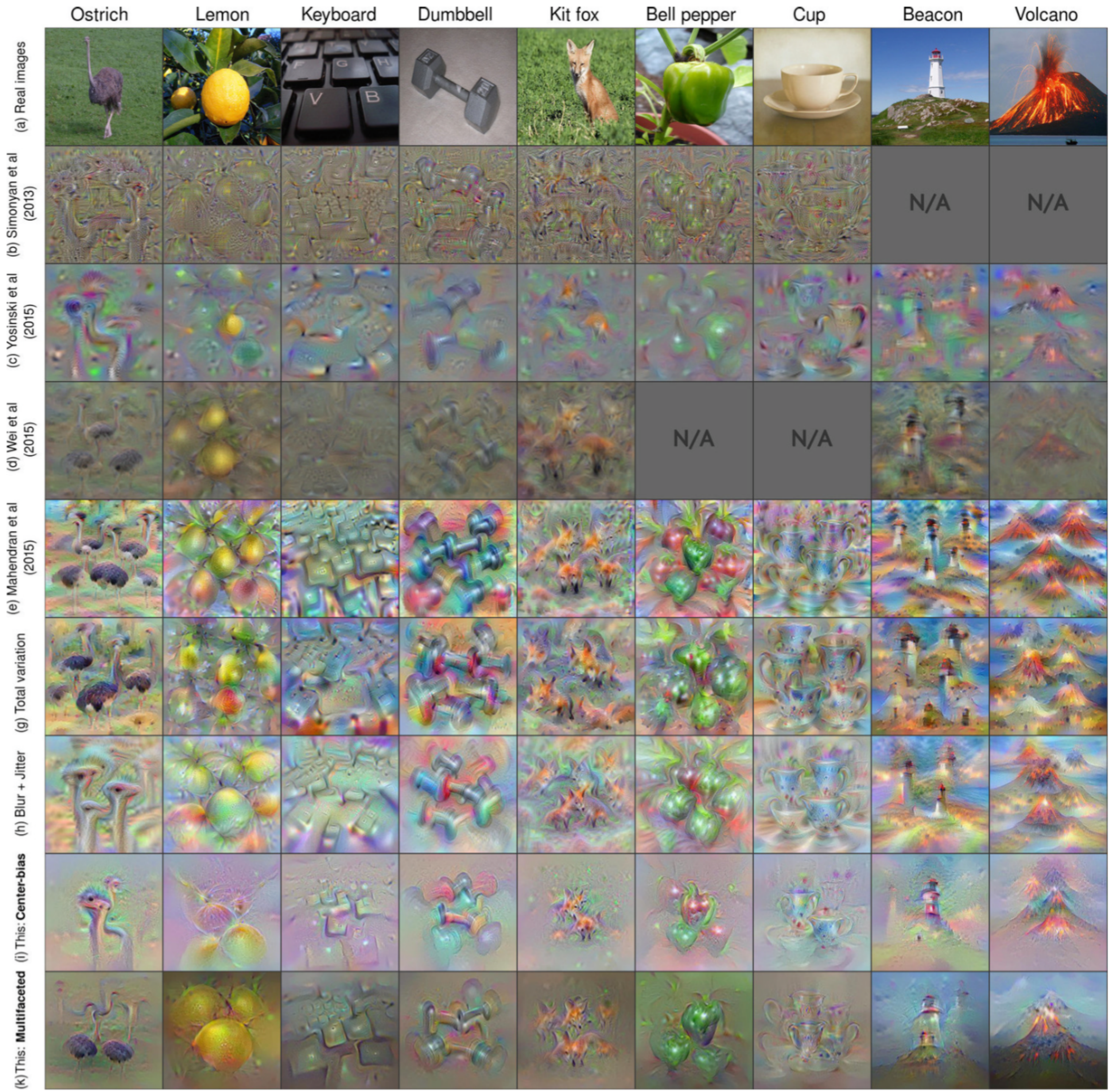


pre-trained

In Practice


Layer 8
Restaurant
Ostrich
Bell gong
Beacon
Cradle
Layer 7
Layer 6
Layer 5
Layer 4
Layer 3
Layer 2

Methods

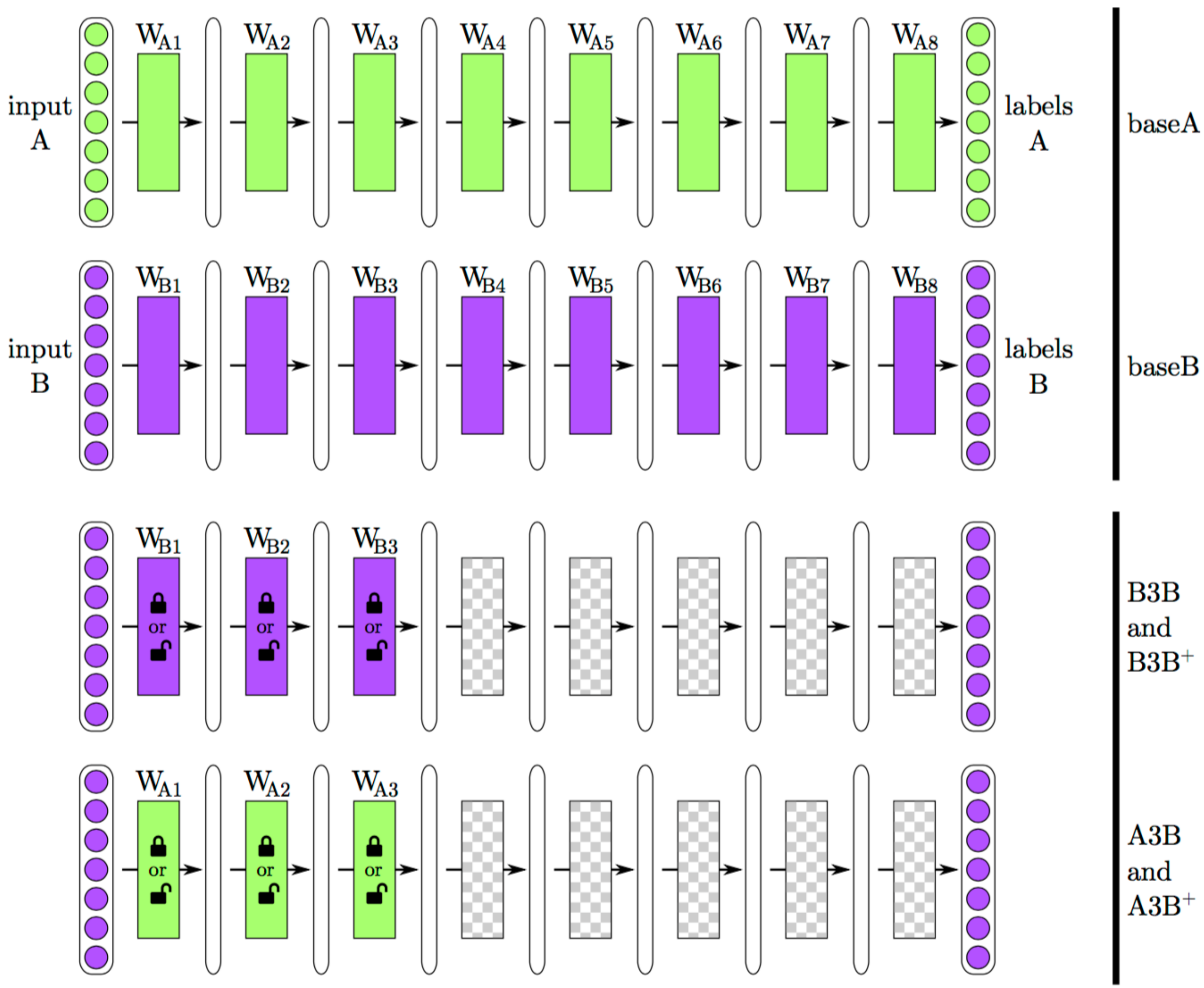


Ostrich
Lemon
Keyboard
Dumbbell
Kit fox
Bell pepper
Cup
Beacon
Volcano
(a) Real images
(b) Simonyan et al (2013)
(c) Yosinski et al (2015)
(d) Wei et al (2015)
(e) Mahendran et al (2015)
(g) Total variation
(h) Blur + Jitter
(i) This: Center-bias
(k) This: Multifaceted
N/A
N/A
N/A
N/A

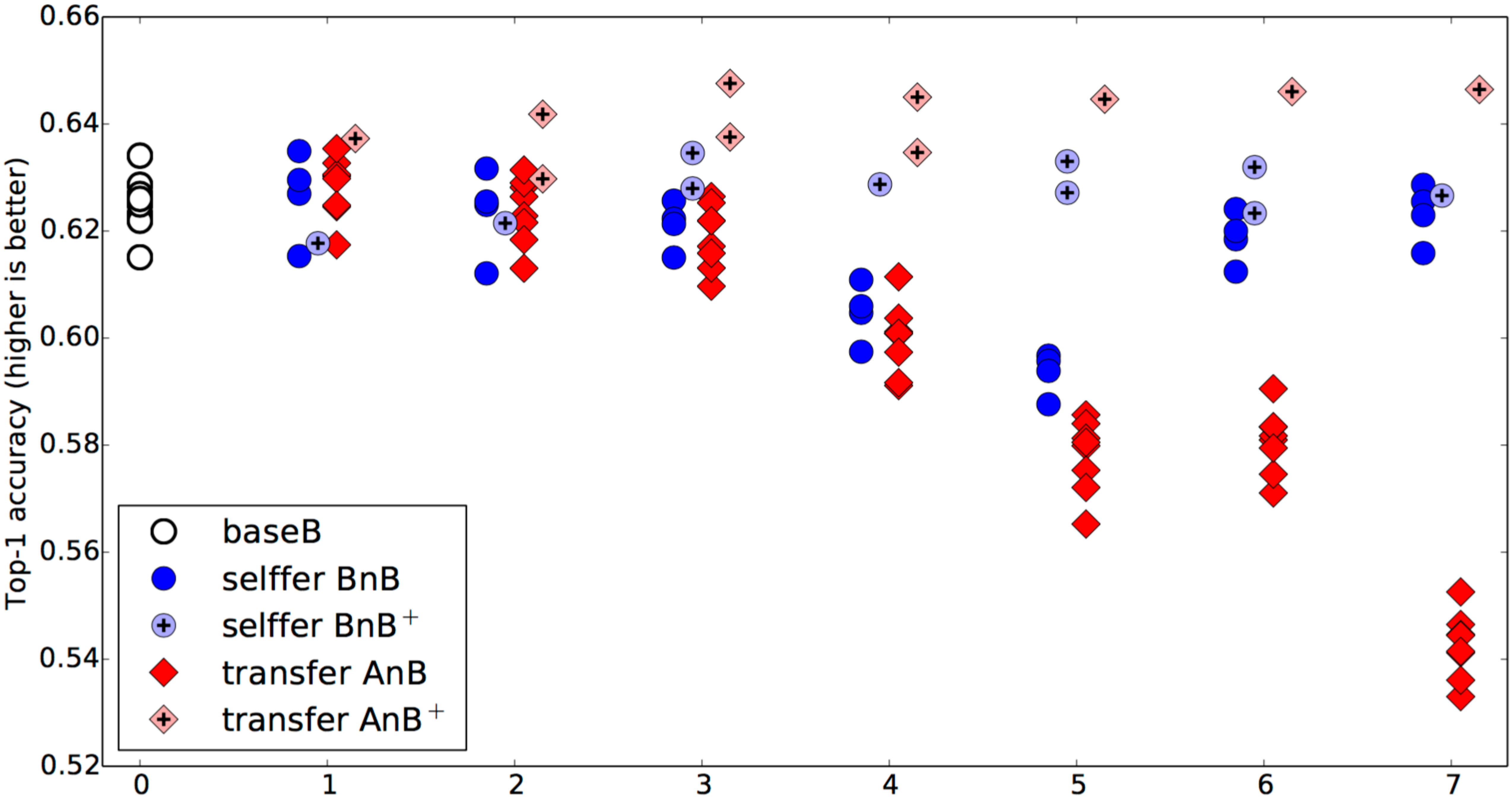
How transferable are features in deep neural networks?

Jason Yosinski,[1] **Jeff Clune,**[2] **Yoshua Bengio,**[3] **and Hod Lipson**[4]
[1] Dept. Computer Science, Cornell University
[2] Dept. Computer Science, University of Wyoming
[3] Dept. Computer Science & Operations Research, University of Montreal
[4] Dept. Mechanical & Aerospace Engineering, Cornell University

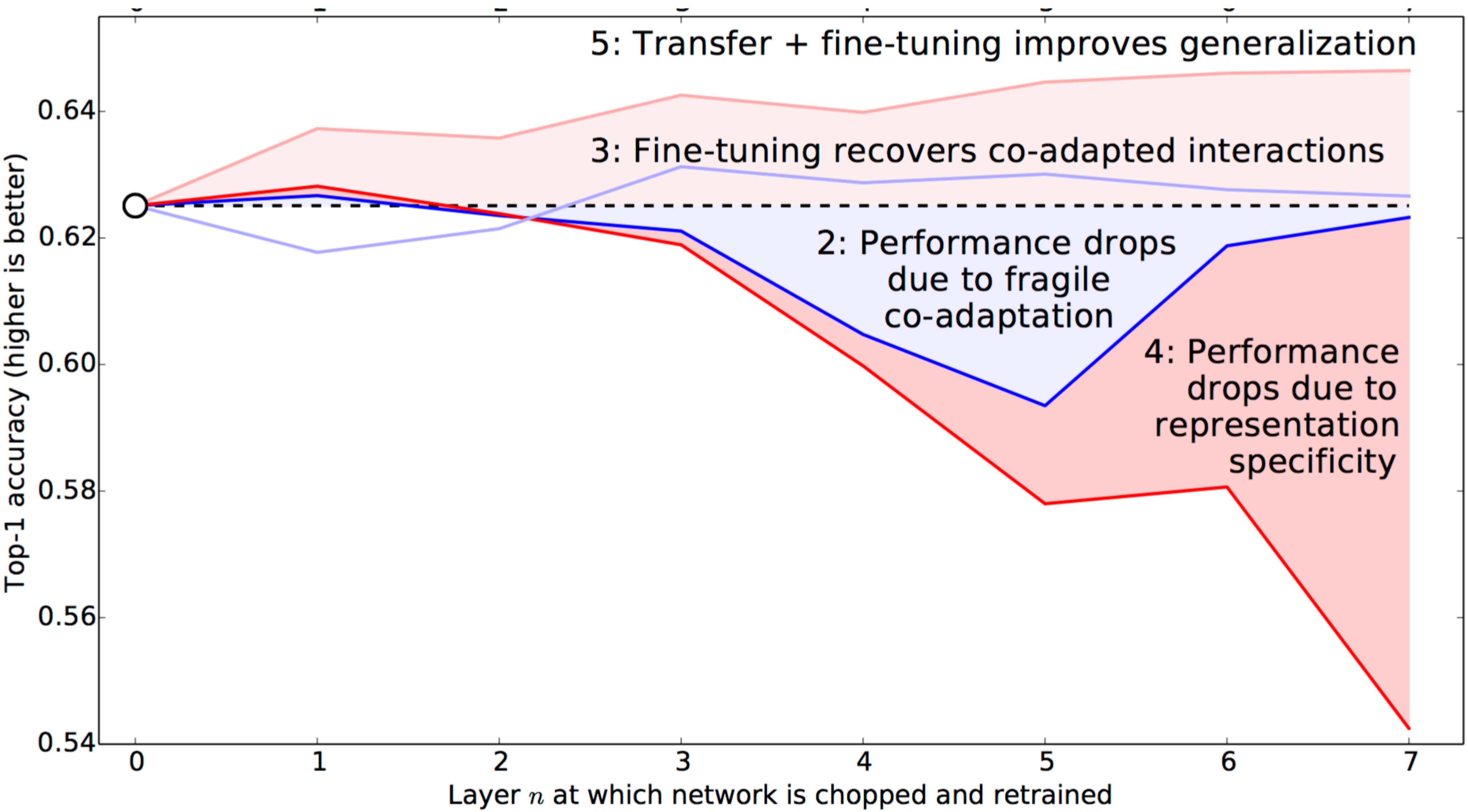
Experiment setup



Where to chop?



Interpretation



Conclusion

1) features are difficult to learn from small datasets
2) features are transferable
3) pre-training is a valid solution to lack of data if task is similar
4) layer visualization is a good way to assess DNN quality

Fin