Lecture 9

What's Going On Inside My Model?

Kian Katanforoosh

Today's outline

I. **Case Study**
II. CNN Interpretation
    A. with saliency maps
    B. with occlusion sensitivity
    C. with class activation maps (Global Average Pooling)
    D. with gradient ascent (class model visualization)
    E. with dataset search
    F. the deconvolution and its applications
III. Modern representation analysis
IV. Training & scaling diagnostics
V. Capabilities & safety dashboards
VI. Data diagnostics
VII. Closing Remarks

You are the model trainer for a new 200B parameter language model at a frontier lab.

Overnight, a new checkpoint passes training sanity checks, but:

- It gets **worse** on some reasoning benchmarks,
- Some safety evals show **higher jailbreak rate**,
- And there is a weird spike in latency for tool-using agents built on top of it.

The VP asks: “What is going on, and what are you going to look at first?”

- List all the types of evidence you would want to inspect before touching the code or restarting training.
- Which three things would you check first, and why?

[Xie, Ras, et al. (2020): Explainable Deep Learning: A Field Guide for the Uninitiated]

Answers fall into 4 buckets

Training & scaling (loss curves, grad norms, LR, MoE routing, scaling laws)

Representations & internals (attention heads, embeddings, neuron behavior)

Data & distribution (drift, contamination, domain mix changes)

Capabilities & safety (benchmarks, adversarial evals, red-team reports)

[Xie, Ras, et al. (2020): Explainable Deep Learning: A Field Guide for the Uninitiated]

Four lenses to understand a large model”

1. **Behavioral lens**: what it does on tasks (evals, safety, agents)
2. **Training & scaling lens**: how it learned (telemetry, curves)
3. **Representation lens**: how it encodes information internally
4. **Data lens**: what it actually saw

[Xie, Ras, et al. (2020): Explainable Deep Learning: A Field Guide for the Uninitiated]

Neural networks are deployed:

- In our phones: to recommend content,
- In banks: to manage investments,
- In hospitals: to help doctors diagnose disease symptoms,
- In insurance agencies: to evaluate risk and underwrite documents,
- In cars: to help avoid accidents.

How can one who will be held accountable for a decision trust a neural network's recommendation, and justify its use?

[Xie, Ras, et al. (2020): Explainable Deep Learning: A Field Guide for the Uninitiated]

Today's outline

I. Case Study

II. CNN Interpretation

- A. with saliency maps
- B. with occlusion sensitivity
- C. with class activation maps (Global Average Pooling)
- D. with gradient ascent (class model visualization)
- E. with dataset search
- F. the deconvolution and its applications

III. Modern representation analysis

IV. Training & scaling diagnostics

V. Capabilities & safety dashboards

VI. Data diagnostics

VII. Closing Remarks

I. A. Interpreting and visualizing Neural Networks with saliency maps

Context: You have built an animal classifier for a zoo. They are reluctant to use your model without human supervision, because they don't understand the decision process of the model.

Question: How can you alleviate their concerns?

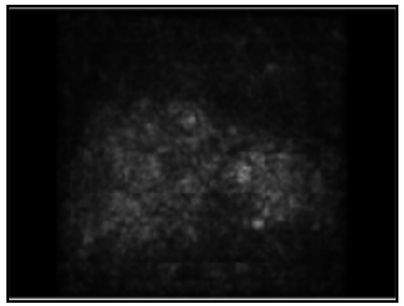
CNN

softmax

$$\begin{pmatrix} 0.04 \\ 0.85 \\ \vdots \\ 0.07 \\ 0.01 \\ \vdots \\ 0.02 \end{pmatrix} \begin{matrix} \text{“iguana} \\ \text{“dog”} \\ \\ \text{“cat”} \\ \text{“ant”} \\ \\ \text{“crab”} \end{matrix}$$

$$softmax\begin{pmatrix} s_{iguana} \\ s_{dog} \\ \vdots \\ s_{cat} \\ s_{ant} \\ \vdots \\ s_{crab} \end{pmatrix} = \begin{pmatrix} \frac{s_{iguana}}{\sum_{animals} s_{animal}} \\ \frac{s_{dog}}{\sum_{animals} s_{animal}} \\ \vdots \\ \frac{s_{crab}}{\sum_{animals} s_{animal}} \end{pmatrix}$$

$$\frac{\partial s_{dog}(x)}{\partial x} =$$



indicates which pixels need to be changed the least to affect the class score the most.

Can be used for segmentation?

Yes

Saliency maps

Kian Katanforoosh

[Karen Simonyan et al. (2014): Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps]

I. A’. Interpreting and visualizing Neural Networks with integrated gradients

$$IG_i(x) = (x_i - x_i^{\text{baseline}}) \times \int_{\alpha=0}^{1} \frac{\partial f(x^{\text{baseline}} + \alpha \times (x - x^{\text{baseline}}))}{\partial x_i} d\alpha$$

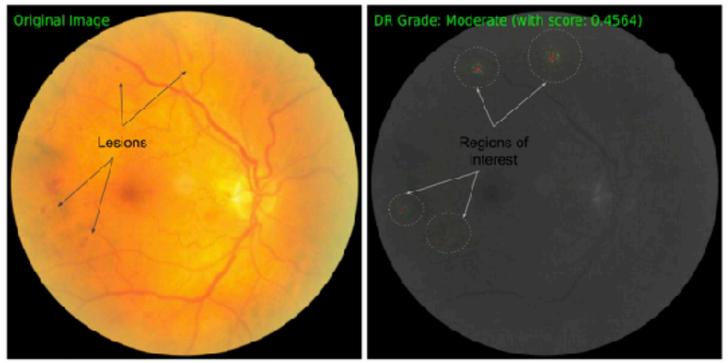
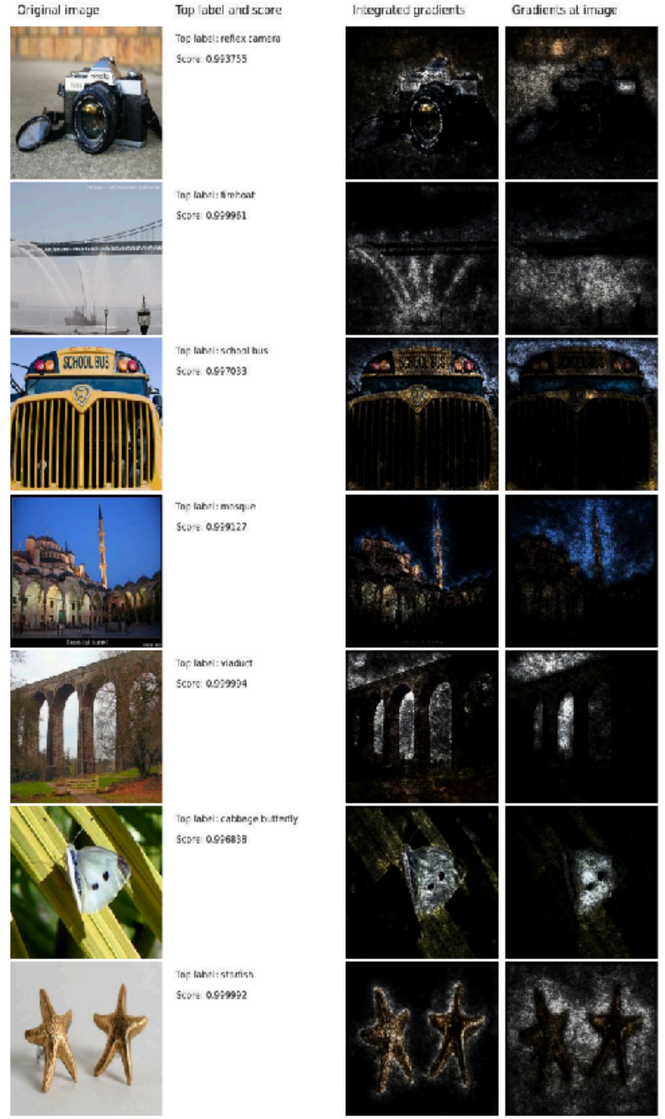


Figure 3. **Attribution for Diabetic Retinopathy grade prediction from a retinal fundus image.** The original image is show on the left, and the attributions (overlayed on the original image in gray scaee) is shown on the right. On the original image we annotate lesions visible to a human, and confirm that the attributions indeed point to them.



[Sundararajan, Ankur, & Yan (2017): Axiomatic Attribution for Deep Networks]

Kian Katanforoosh

I. B. Interpreting and visualizing Neural Networks with occlusion sensitivity

Context: You have built an animal classifier for a zoo. They are a little reluctant to use your model without human supervision, because they don't understand the decision process of the model.

Question: What can you do to alleviate their concerns?

→ CNN →

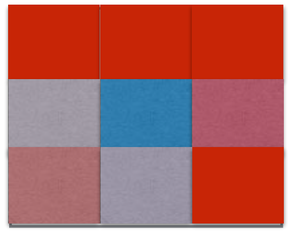
$$\begin{pmatrix} 0.04 \\ 0.85 \\ \vdots \\ 0.07 \\ 0.01 \\ \vdots \\ 0.02 \end{pmatrix}$$

"iguana"
"dog"
"cat"
"ant"
"crab"

→ CNN →

$$\begin{pmatrix} 0.05 \\ 0.83 \\ \vdots \\ 0.07 \\ 0.01 \\ \vdots \\ 0.02 \end{pmatrix}$$

"iguana"
"dog"
"cat"
"ant"
"crab"



Probability map of the true class for different positions of the grey square

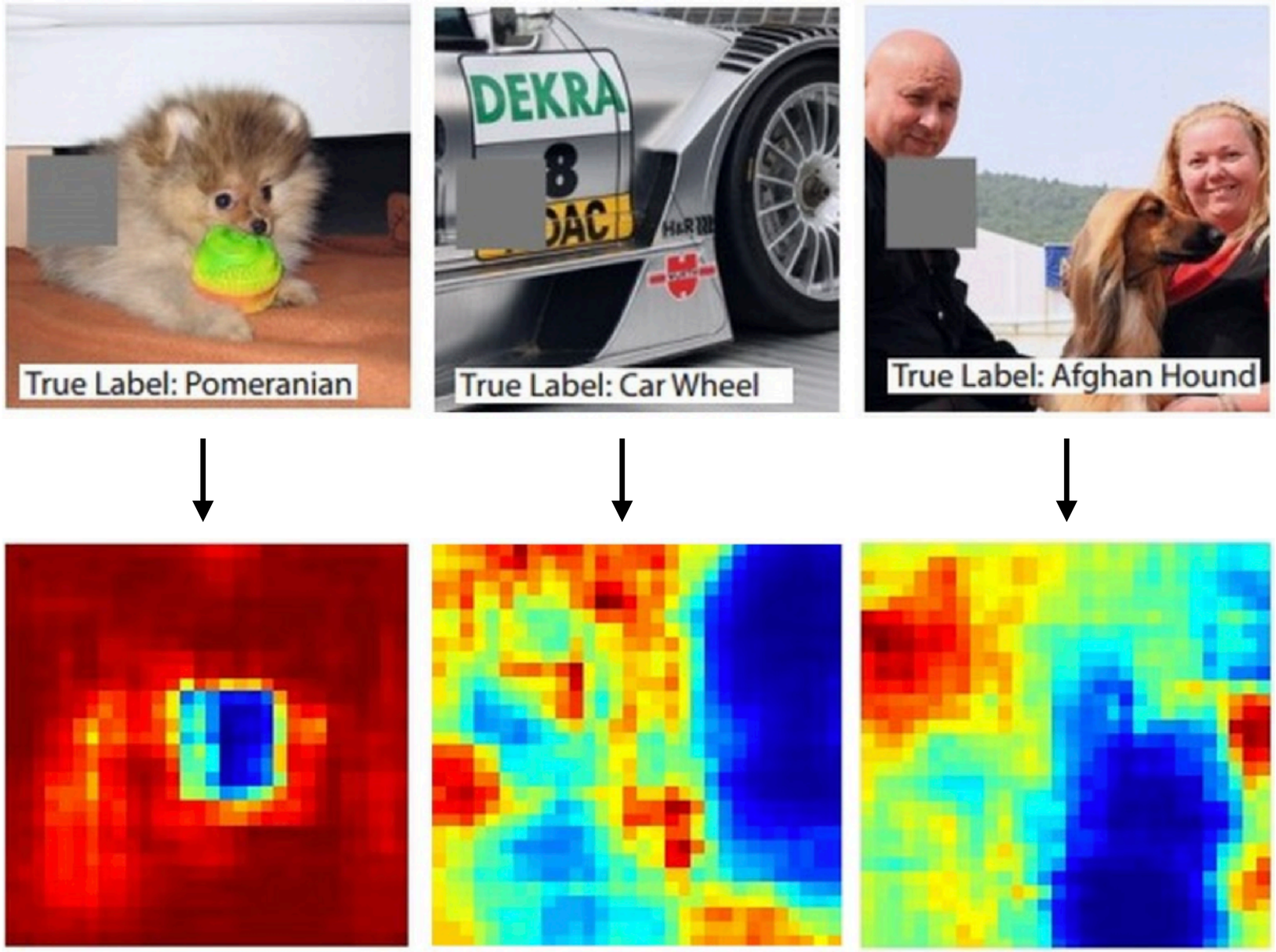
Indicates low confidence on the true class for the corresponding position of the grey square

Indicates high confidence on the true class for the corresponding position of the grey square

[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

Kian Katanforoosh

I. B. Interpreting and visualizing Neural Networks with occlusion sensitivity



Probability map of the true class for different positions of the grey square

Indicates low confidence on the true class for the corresponding position of the grey square

Indicates high confidence on the true class for the corresponding position of the grey square

Occlusion sensitivity

[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

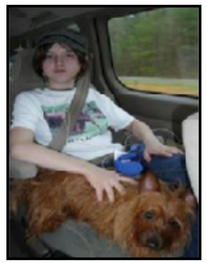
Kian Katanforoosh

I. C. Interpreting NNs using class activation maps

Context: Along with the classification output, the zoo now wants real-time visualization of the model's decision process. You have one day. What do you do?

Using a classification network for localization:

Converted to:

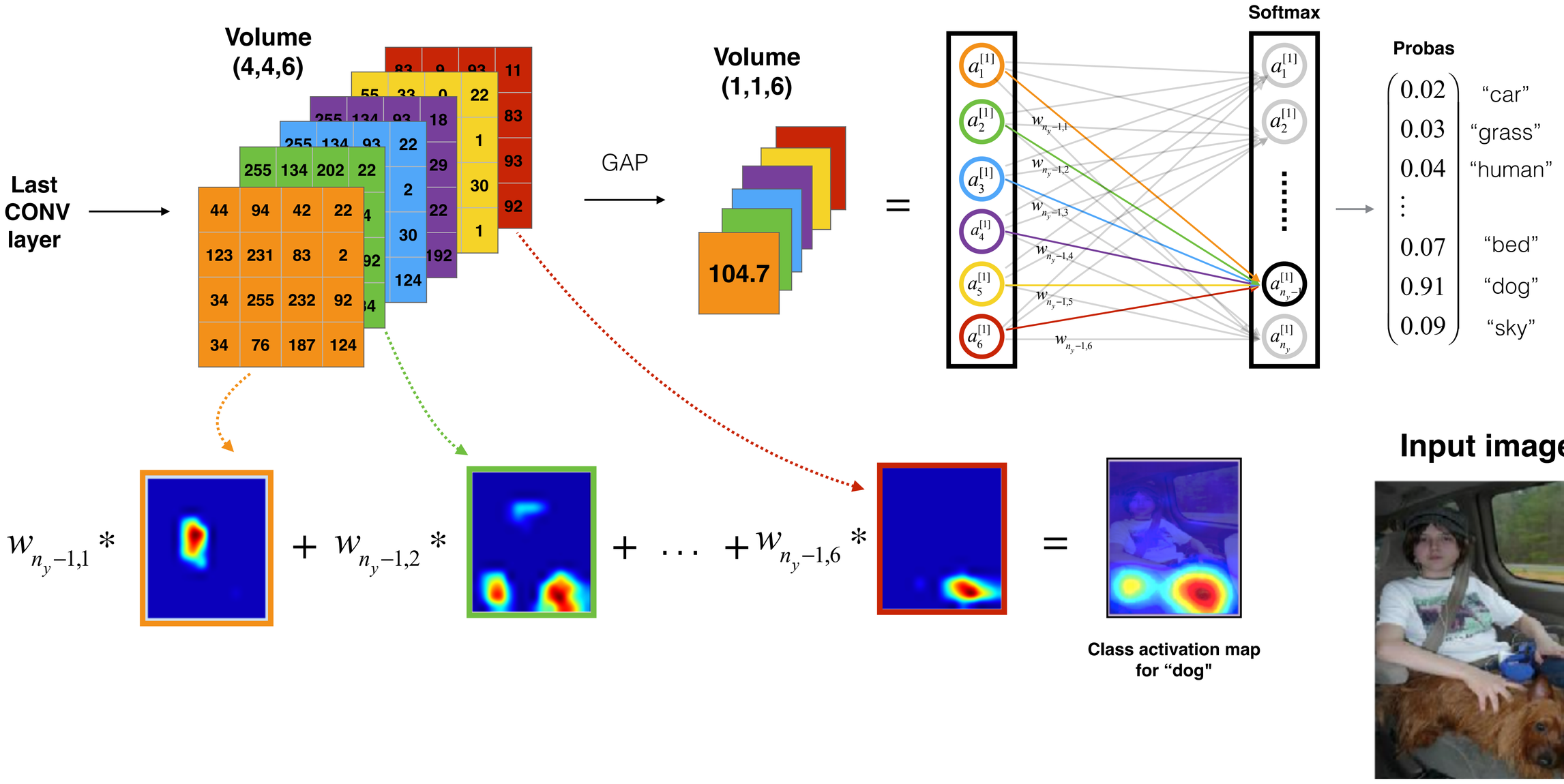


Why this?

[Bolei Zhou et al. (2016): Learning Deep Features for Discriminative Localization]

Kian Katanforoosh

I. C. Interpreting NNs using class activation maps



[Bolei Zhou et al. (2016): Learning Deep Features for Discriminative Localization]

Kian Katanforoosh

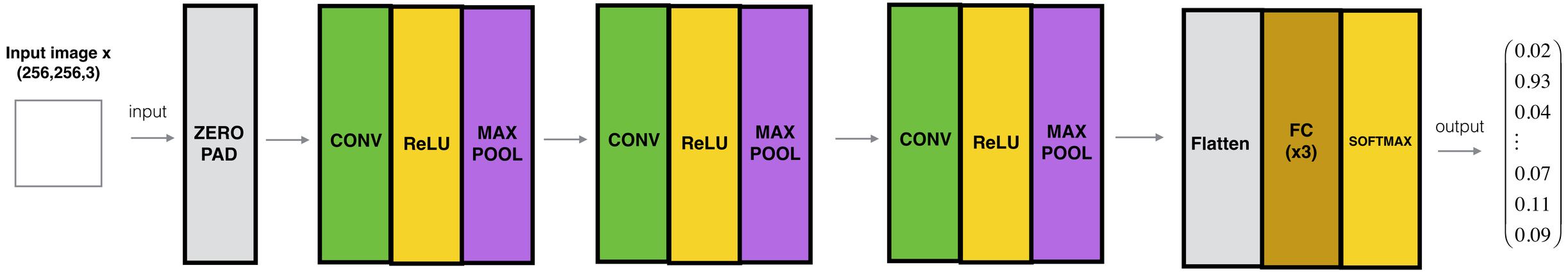
I. C. Interpreting NNs using class activation maps

[Bolei Zhou et al. (2016): Learning Deep Features for Discriminative Localization]
Source video: Kyle McDonald

II. A. Visualizing NNs from the inside using gradient ascent (class model visualization)

Context: The zoo trusts that your model correctly locates animals. They get scared and they ask you whether the model understands what a dog is.

Given this trained ConvNet, generate an image which is representative of the class “dog” according to the ConvNet



Keep the weights fixed and use gradient ascent on the input image to maximize this loss :

$$L = s_{dog}(x) - \lambda \left|\left| x \right|\right|_2^2$$

“x should look natural”

Gradient ascent:

$$x = x + \alpha \frac{\partial L}{\partial x}$$

Repeat this process:

1. Forward propagate image x
2. Compute the objective L
3. Backpropagate to get dL/dx
4. Update x’s pixels with gradient ascent

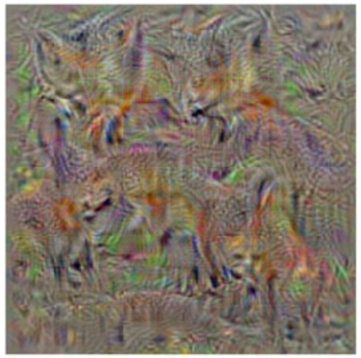
[Karen Simonyan et al. (2014): Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps]

Kian Katanforoosh

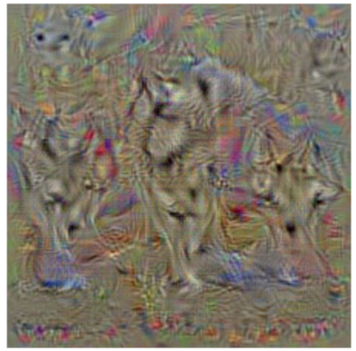
II. A. Visualizing NNs from the inside using gradient ascent (class model visualization)

goose

ostrich

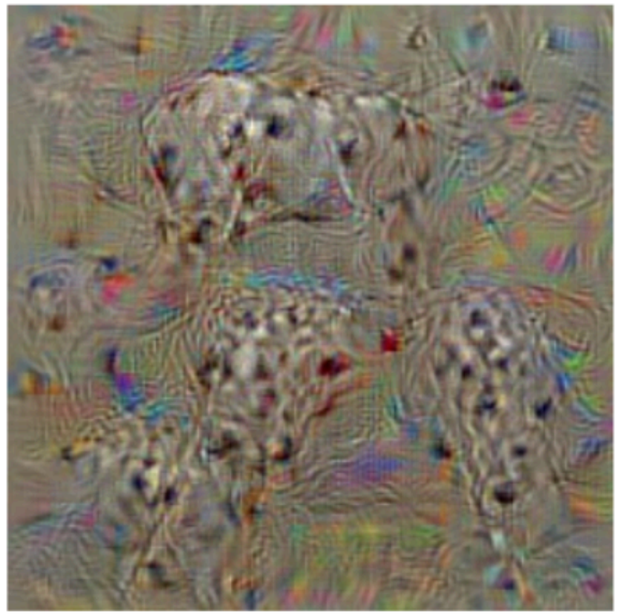


kit fox



husky

We can do this for all classes:

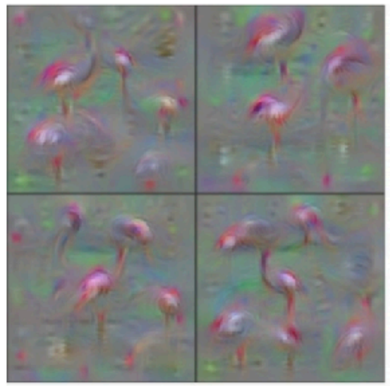


dalmatian

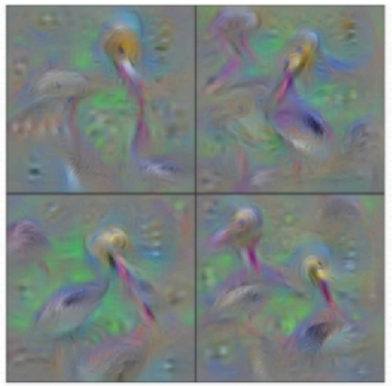
L2
Regularization

Looks better with additional regularization methods.

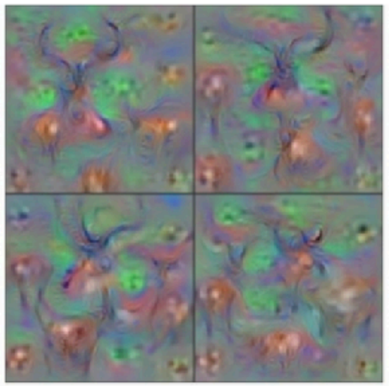
Class model visualization



Flamingo



Pelican



Hartebeest

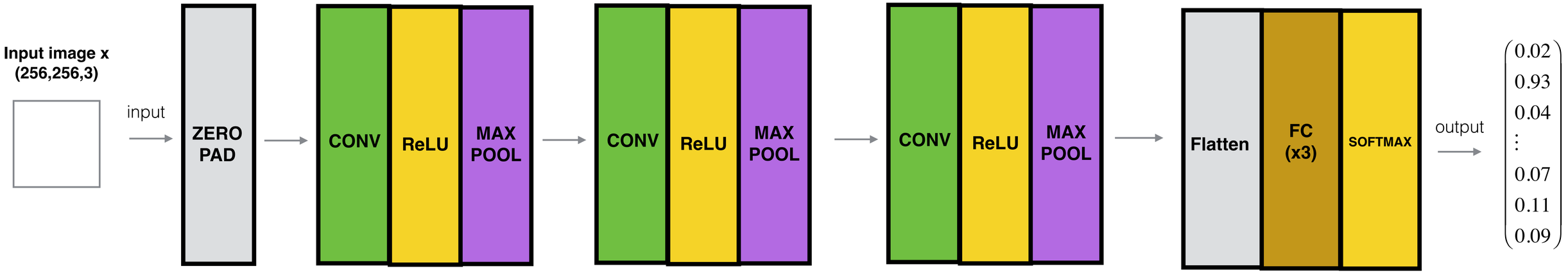
[Jason Yosinski et al. (2015): Understanding Neural Networks Through Deep Visualization]

[Karen Simonyan et al. (2014): Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps]

Kian Katanforoosh

II. A. Visualizing NNs from the inside using gradient ascent (class model visualization)

This method can be applied to any activation in the network
in order to interpret what a neuron is detecting



On the class score:

$$L = S_{dog}(x) - R(x)$$

change to →

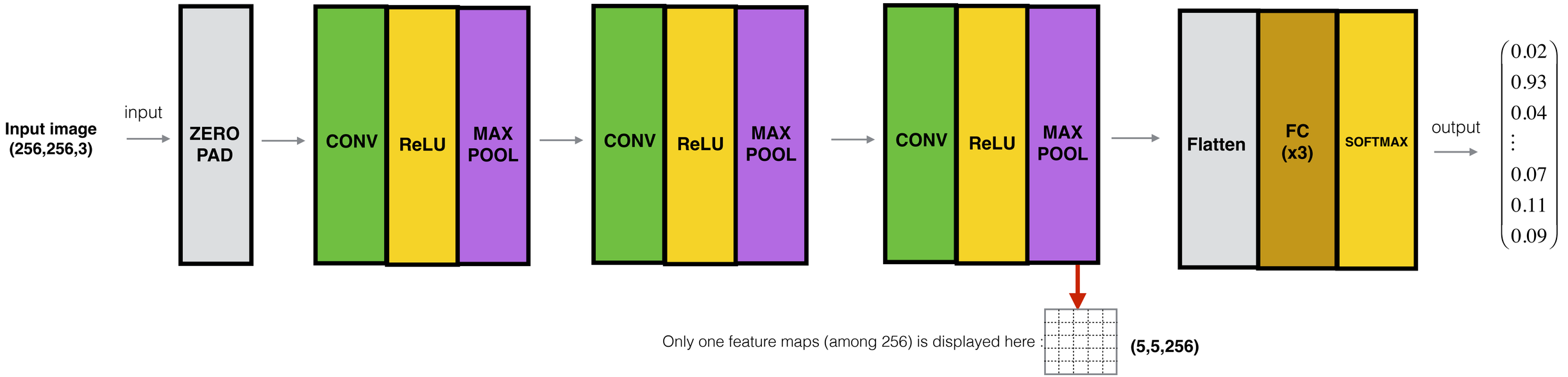
On any activation:

$$L = a_j^{[l]}(x) - R(x)$$

[Jason Yosinski et al. (2015): Understanding Neural Networks Through Deep Visualization]

II. B. Visualizing NNs from the inside using dataset search

Context: The zoo loved the technique, and asks if there are other alternatives.



Given a filter, what examples in the dataset lead to a strongly activated feature map?

Top 5 images

It seems that the filter has learned to detect shirts

Top 5 images

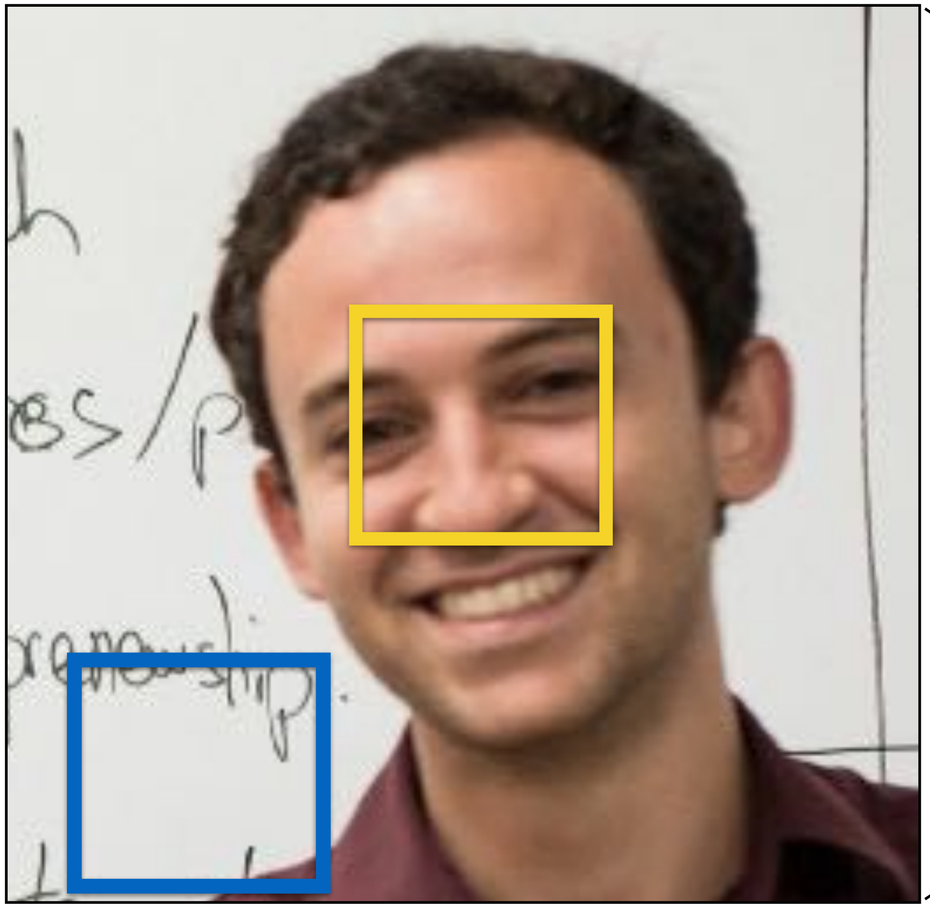
It seems that the filter has learned to detect edges

Kian Katanforoosh

II. B. Visualizing NNs from the inside using dataset search

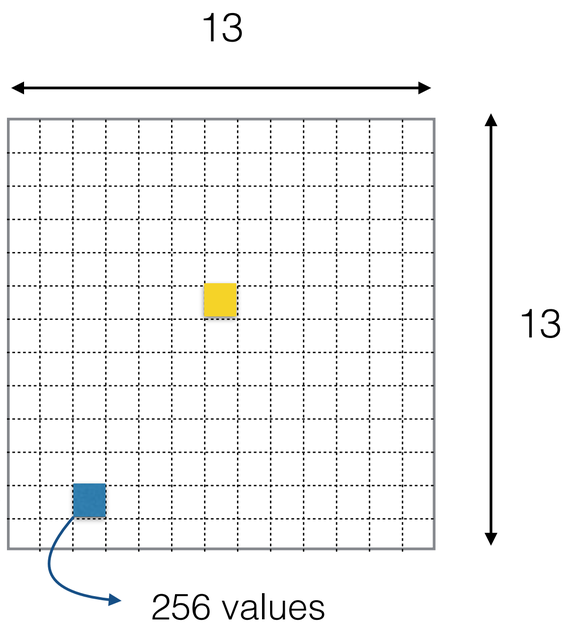
How did we crop the dataset images on the previous slide?

Input image
(64,64,3)



CONV
(5 layers)

Encoding volume
(13,13,256)



Conclusion: the deeper the activation, the more it “sees” from the image

Kian Katanforoosh

II. C. The deconvolution and its applications

Motivation behind deconvolution/upsampling layers

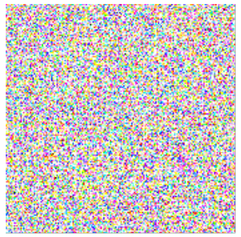
100-d
random code

$$\begin{pmatrix} 0.47 \\ \vdots \\ 0.19 \end{pmatrix}$$

z

Generator "G"
(Neural Network)
DECONV

(64,64,3)
generated image



x

Discriminator "D"
(Neural Network)

Binary classification

$$\begin{cases} y = 0 \quad if \quad x = G(z) \\ y = 1 \quad otherwise \end{cases}$$

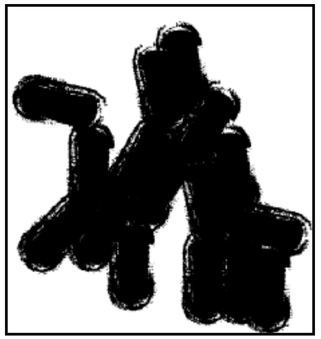
Real images
(database)

Kian Katanforoosh

II. C. The deconvolution and its applications

Motivation behind deconvolution/upsampling layers

Input image
(400, 400, 3)



Convolutions
(reduces volume height and width)

Information Encoded

Encoding

De-convolutions
(increases volume height and width)

Per-Pixel Classification
(400, 400, 1)

Get your pencils ready, we’re about to study the deconvolution.

Consider the following CONV1D:

$$n_y = \lfloor \frac{n_x - f + 2p}{s} \rfloor + 1 = \lfloor \frac{8 - 4 + 2 \times 2}{2} \rfloor + 1 = 5$$

$$x = \begin{bmatrix} 0 \\ 0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_8 \\ 0 \\ 0 \end{bmatrix} \longrightarrow$$

CONV 1D

1 filter of size 4
stride = 2

$$\longrightarrow y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

Let's define our filter as:

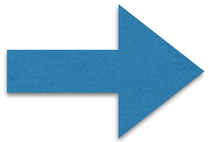
$$f = (w_1, w_2, w_3, w_4)$$

The system of equations is:

$$y_1 = w_1 \cdot 0 + w_2 \cdot 0 + w_3 \cdot x_1 + w_4 \cdot x_2$$
$$y_2 = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + w_4 \cdot x_4$$
$$\vdots$$
$$y_5 = w_1 \cdot x_7 + w_2 \cdot x_8 + w_3 \cdot 0 + w_4 \cdot 0$$

Let's rewrite the system of equations:

$$
\begin{aligned}
y_1 &= w_1 \cdot 0 + w_2 \cdot 0 + w_3 \cdot x_1 + w_4 \cdot x_2 \\
y_2 &= w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + w_4 \cdot x_4 \\
&\vdots \\
y_5 &= w_1 \cdot x_7 + w_2 \cdot x_8 + w_3 \cdot 0 + w_4 \cdot 0
\end{aligned}
$$



$$y = Wx$$

(5,1) (5,12) (12,1)

$$
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & w_4 & 0 & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & w_1 & w_2 & w_3 & w_4 & 0 & \dots \\ & & & & & & & & & \vdots \\ 0 & \dots & \dots & 0 & 0 & 0 & 0 & w_1 & w_2 & w_3 & w_4 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_8 \\ 0 \\ 0 \end{bmatrix}
$$

1D CONV can be rewritten as a matrix vector multiplication!

Kian Katanforoosh

1D CONV $y = Wx$ (5,1) (5,12) (12,1)

1D DECONV If W is invertible, then $\exists H = W^{-1}$ such that $x = Hy$ (12,1) (12,5) (5,1)

In practice, we would even assume that W is orthogonal, i.e. $W^{-1} = W^T$

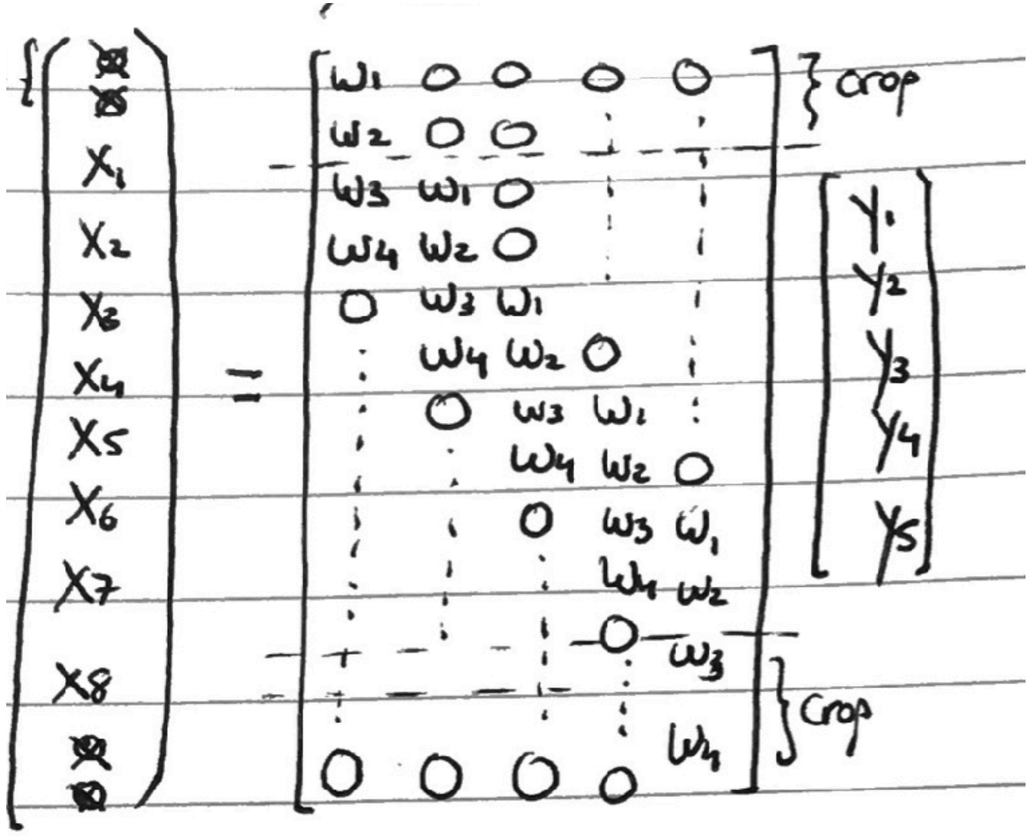
For example $(w_1, w_2, w_3, w_4) = (-1,0,0,0,1)$

Thus: $x = W^T y$ Deconvolution ~ Transposed convolution

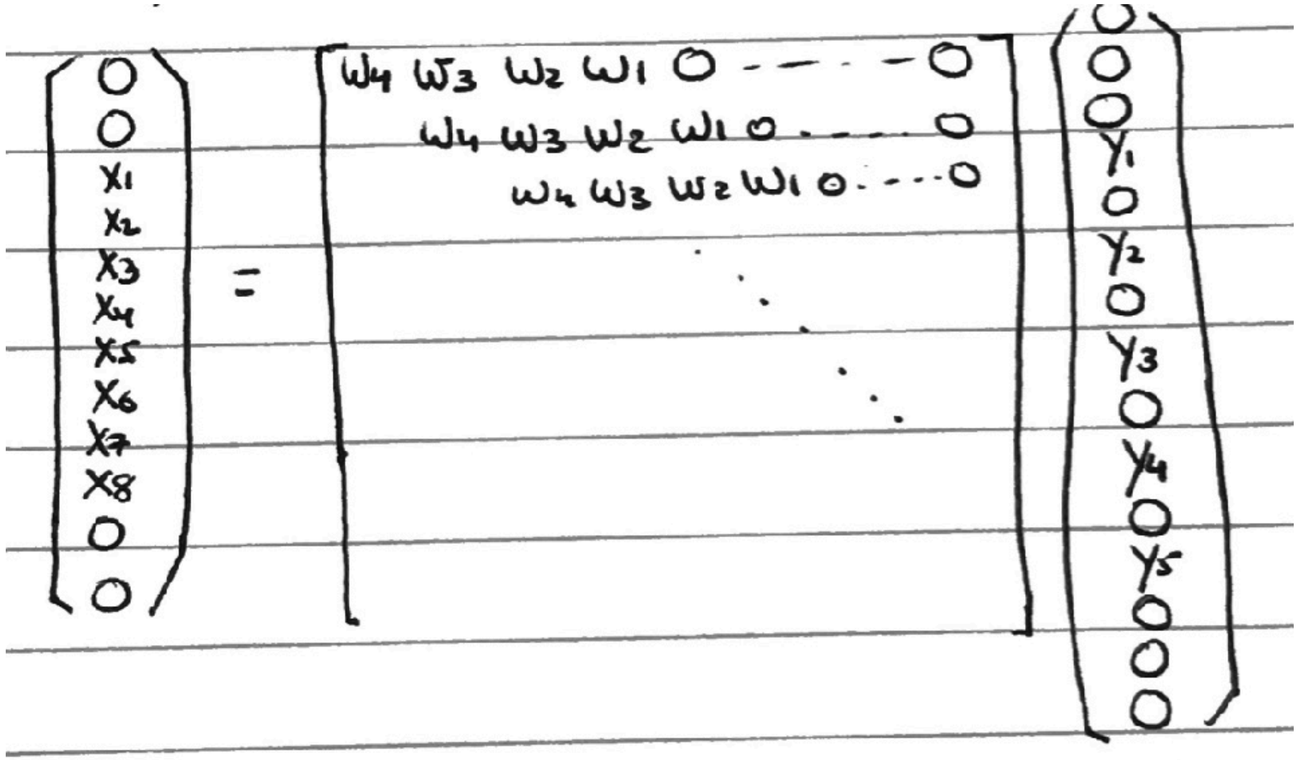
Kian Katanforoosh

Let's rewrite: $x = W^T y$

Transposed convolution with stride 2



Sub-pixel convolution with stride 1/2



Kian Katanforoosh

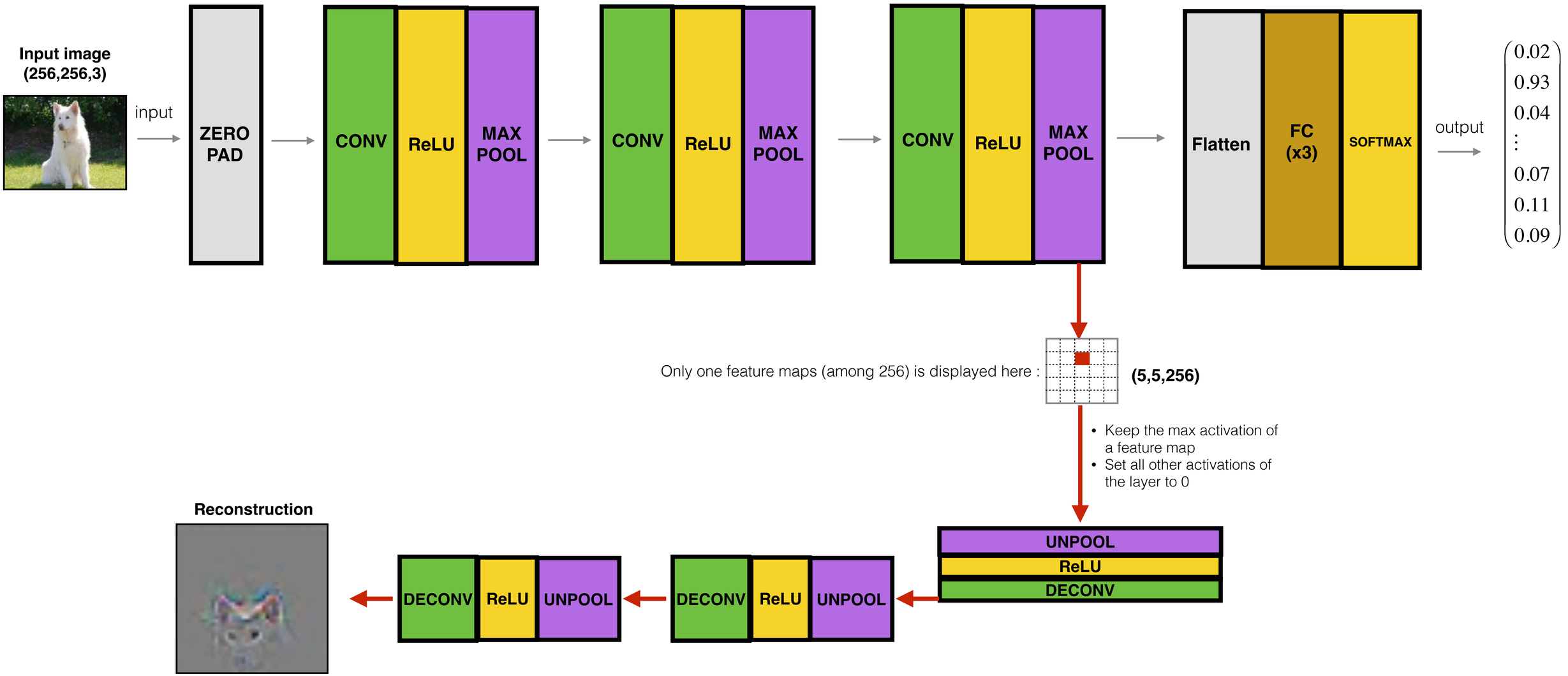
What to remember?

Implementing a deconvolution (sub-pixel version) is similar to the convolution, with:

1. Create a sub-pixel version of the input (i.e., insert zeros and pad)
2. Flip the filters.
3. Divide the stride by 2.

III. A. Interpreting NNs using deconvolutions

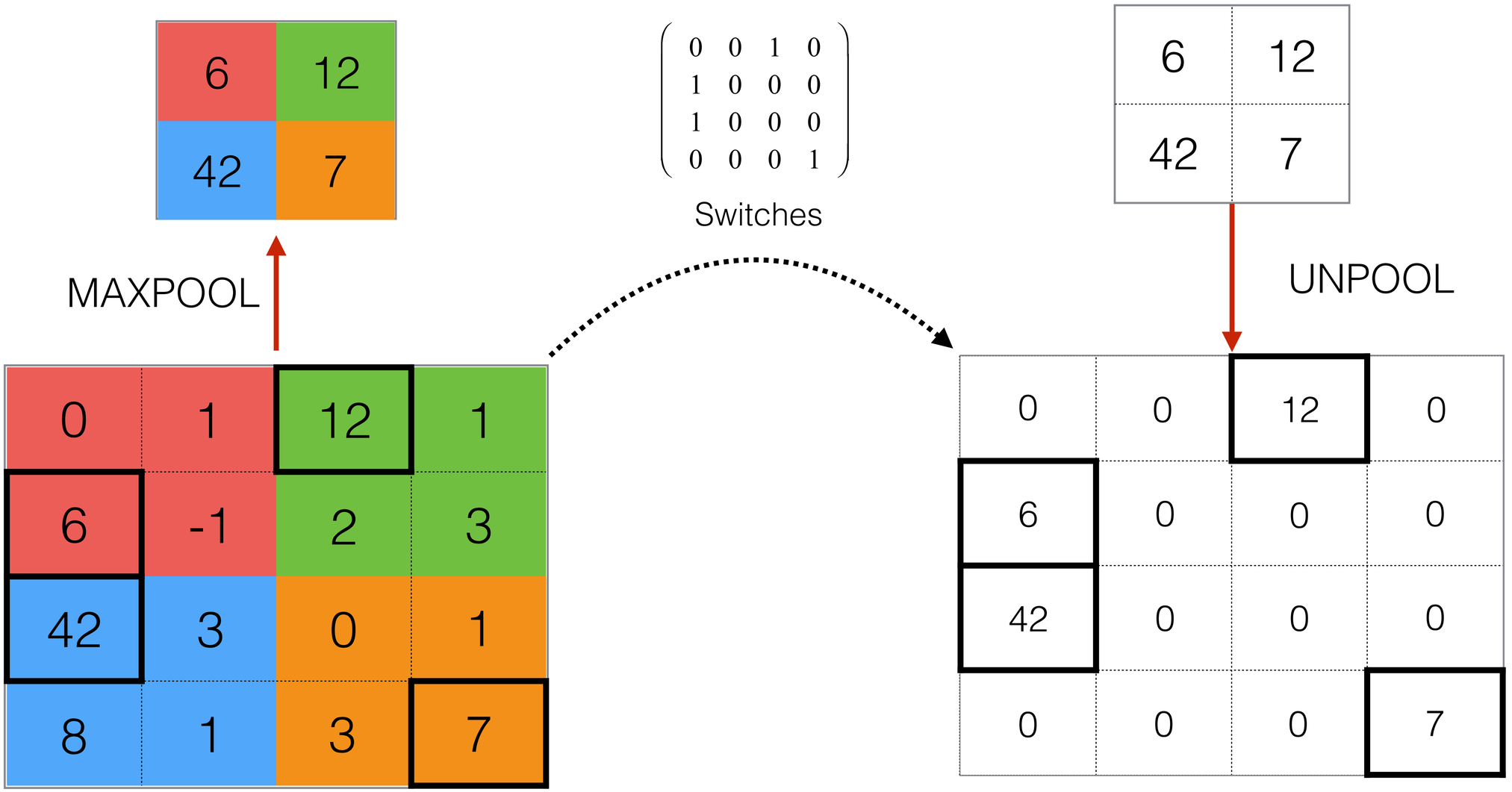
Motivation of DeconvNets for visualization: Here is a CNN, trained on ImageNet (1.3m images, 1000 classes), we're trying to interpret by reconstructing the activation's zone of influence in the input space.



[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

Kian Katanforoosh

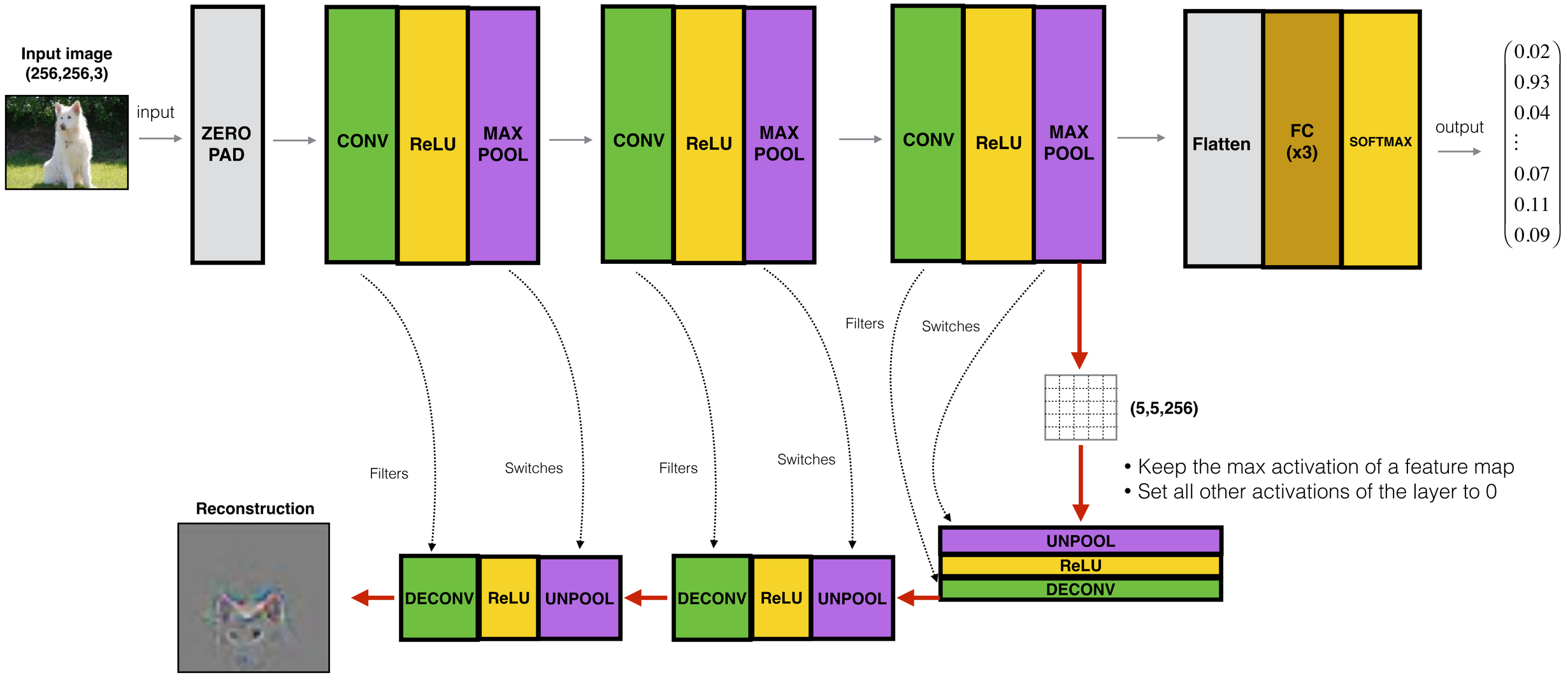
II. C. The deconvolution and its applications



Kian Katanforoosh

II. C. The deconvolution and its applications

We need to pass the filters and switches from the ConvNet to the DeconvNet.



[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

Kian Katanforoosh

II. C. The deconvolution and its applications

$$a^{[l]} = \mathrm{I}\{a^{[l+1]} \geq 0\}.a^{[l+1]}$$

1	0	3	0
4	0	0	0
30	2	0	1
1	0	0	7

“ReLU forward”

1	-2	3	-4
4	-1	-1	-2
30	2	0	1
1	-2	-9	7

Switches

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

-2	1	-31	-3
3	4	2	14
-2	12	4	1
10	10	2	1

“ReLU backward”

-2	0	-31	0
3	0	0	0
-2	12	0	1
10	0	0	1

-2	1	-31	-3
3	4	2	14
-2	12	4	1
10	10	2	1

“ReLU DeconvNet”

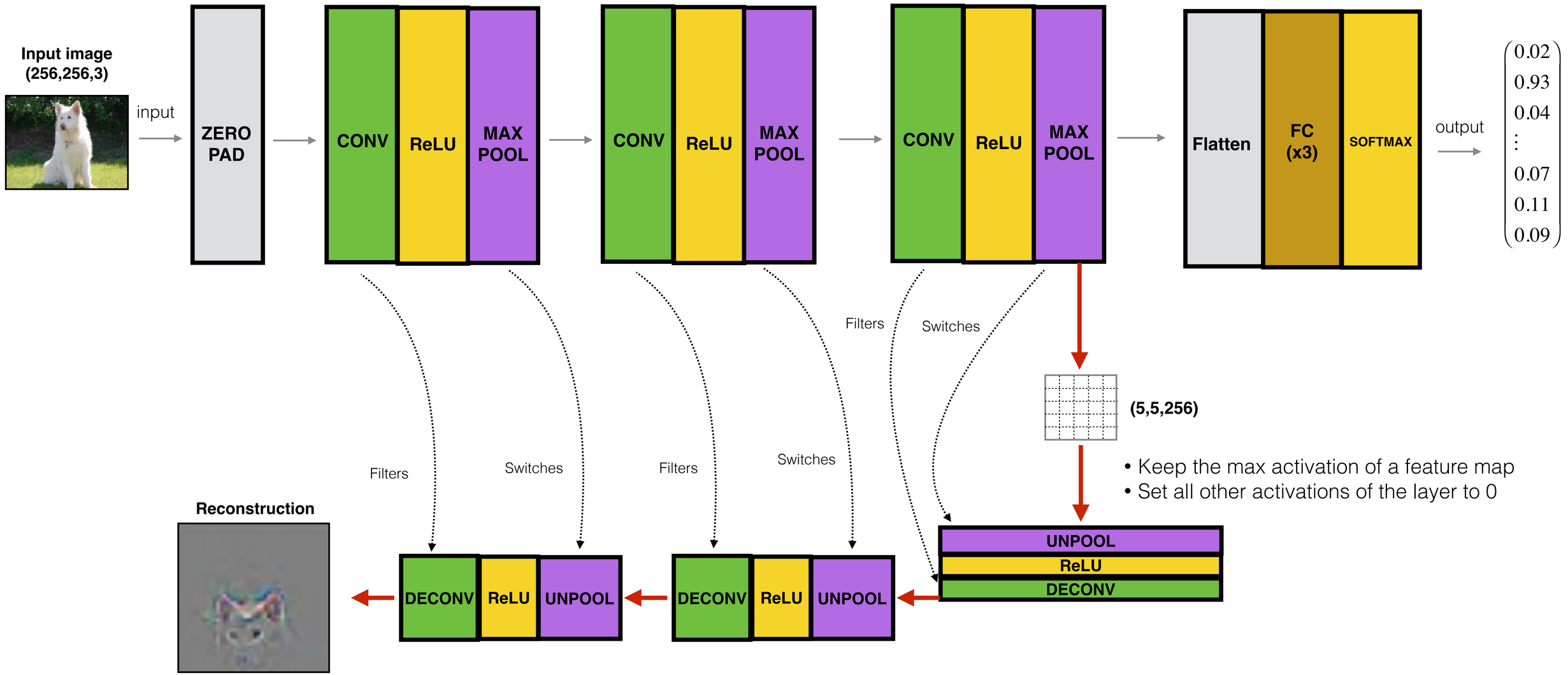
0	1	0	0
3	4	2	14
0	12	4	1
10	10	2	1

[Sringenberg & Dosovitskiy et al. (2015): Visualizing and Understanding Convolutional Networks]

Kian Katanforoosh

II. C. The deconvolution and its applications

We need to pass the filters and switches from the ConvNet to the DeconvNet.

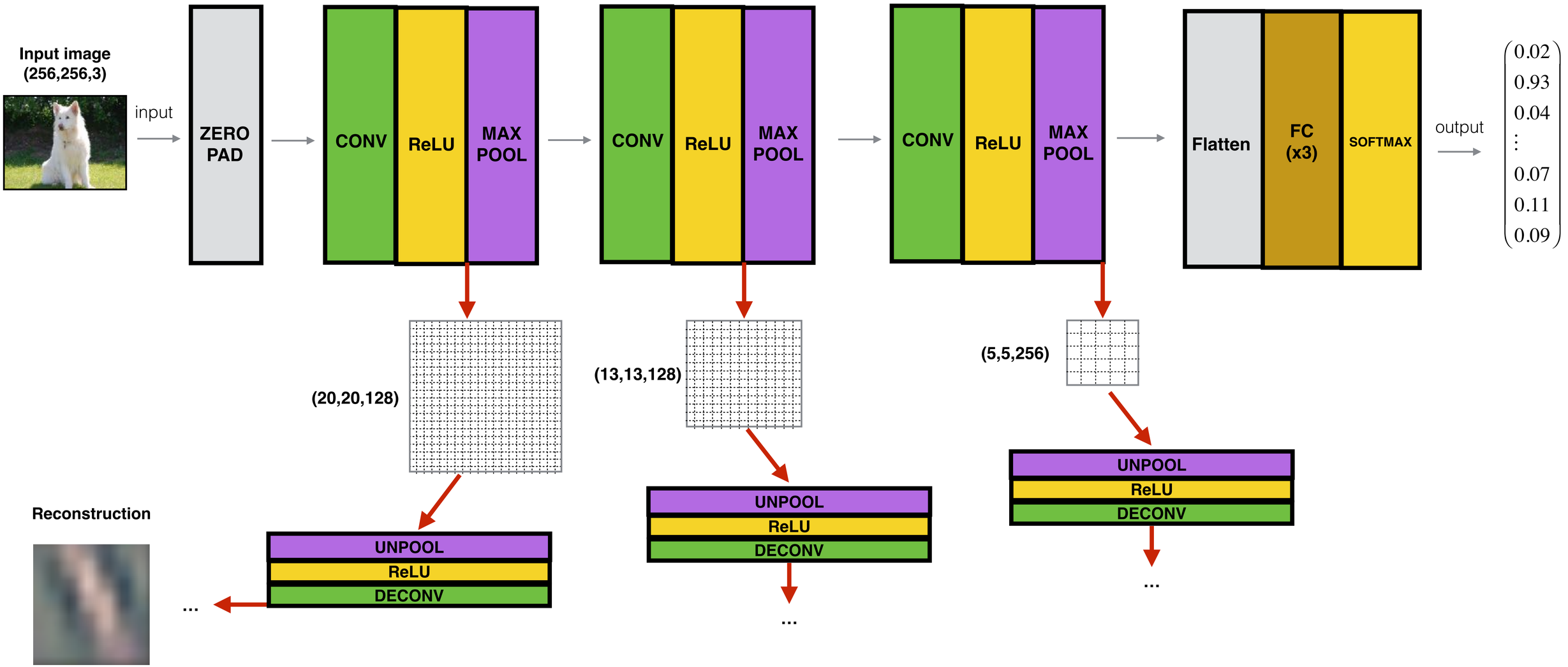


[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

Kian Katanforoosh

II. C. The deconvolution and its applications

Other CONV layers can be visualized the same way

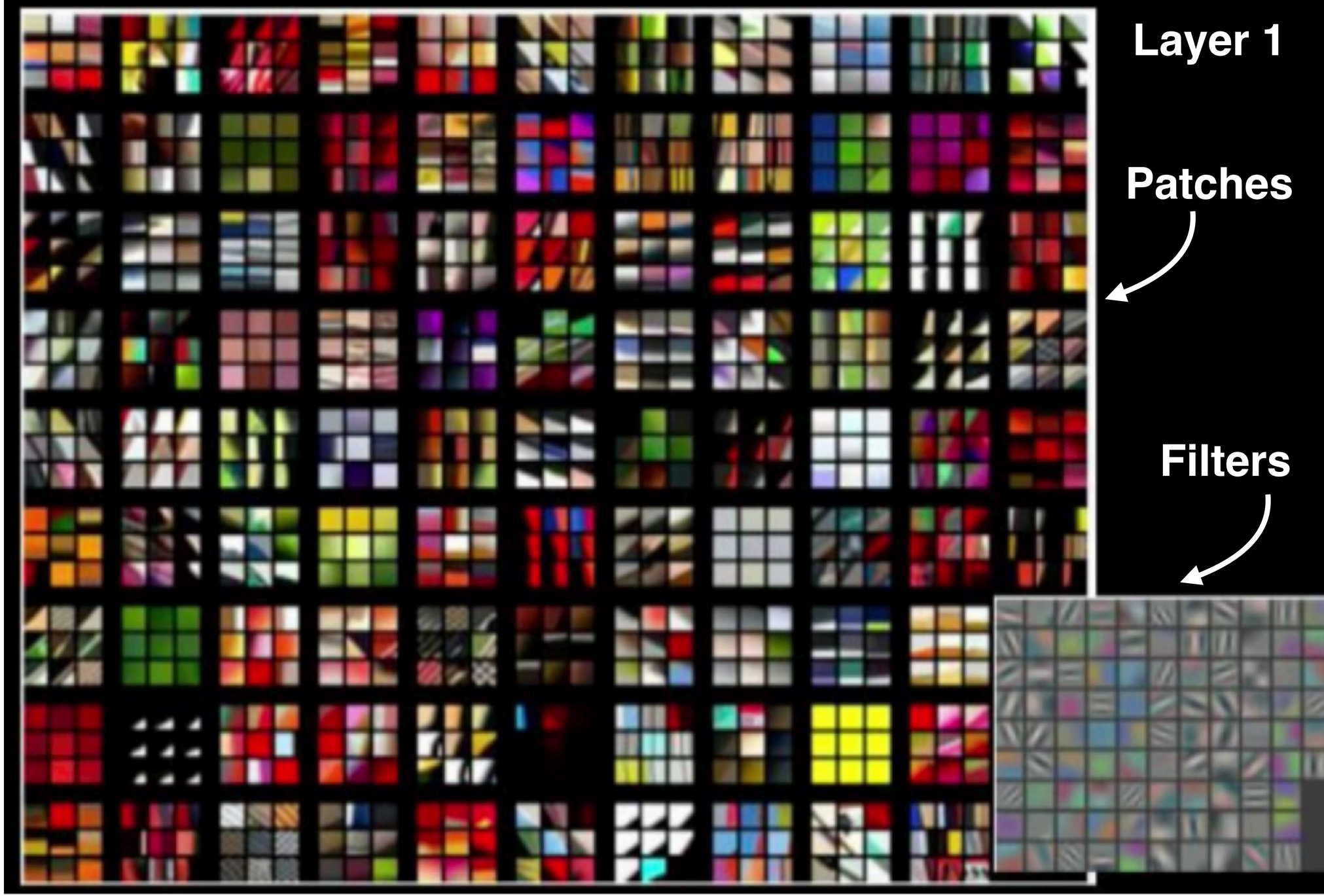


[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

Kian Katanforoosh

Results on a validation set of 50,000 images

- Top-9 strongest activations per filter in the 1st layer
- Because we know the position of the activation and all the pooling switches we can crop the part of the image that fired the activation



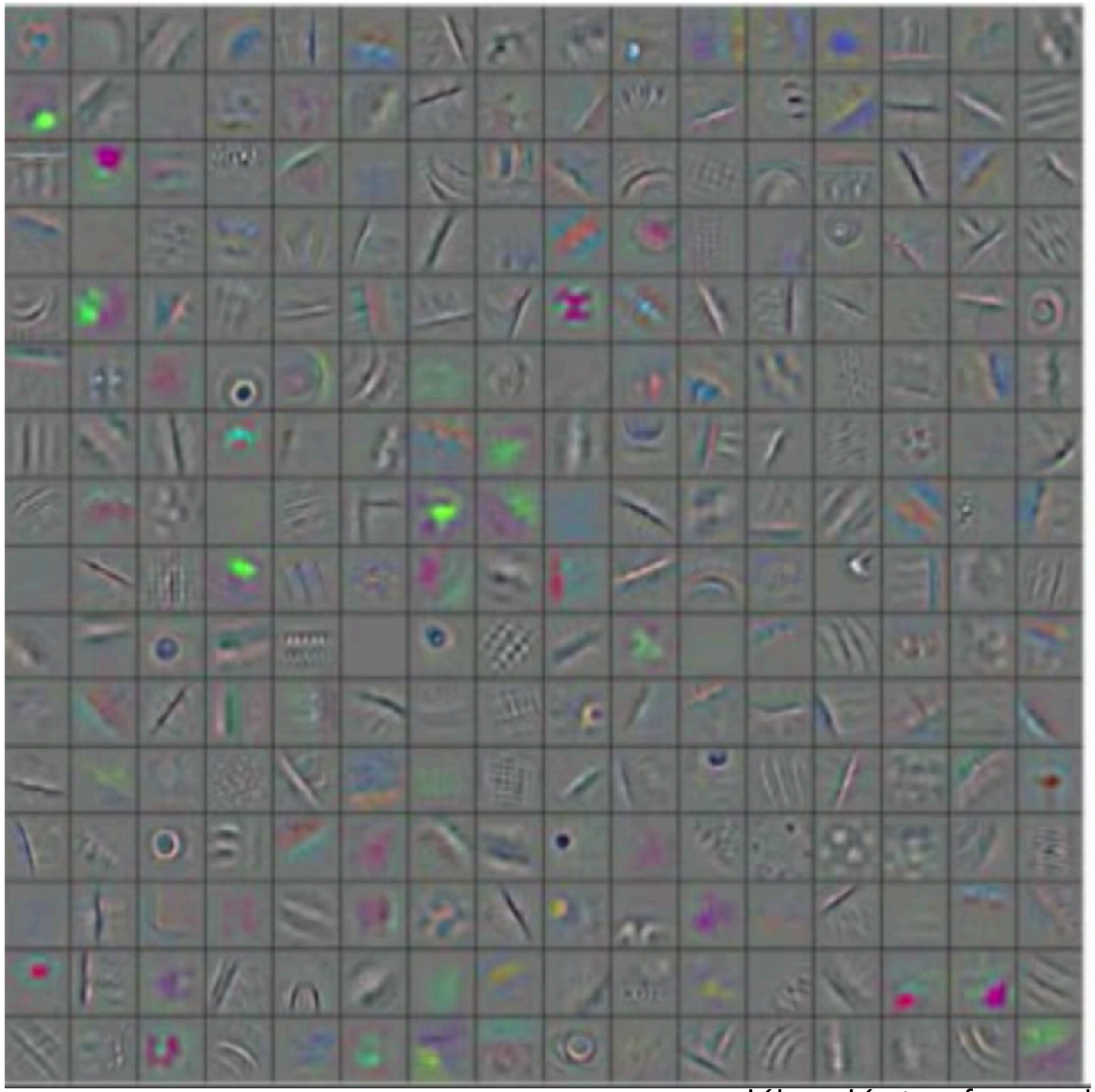
Kian Katanforoosh

[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

Results on a validation set of 50,000 images

- Learning a more complex set of patterns than 1st layer edges.
- Covers a much larger space of the image because of the pooling layer before.
- Top-1 strongest activation per feature map in the 2nd layer (256 feature maps.)

Layer 2 reconstructions (deconv)

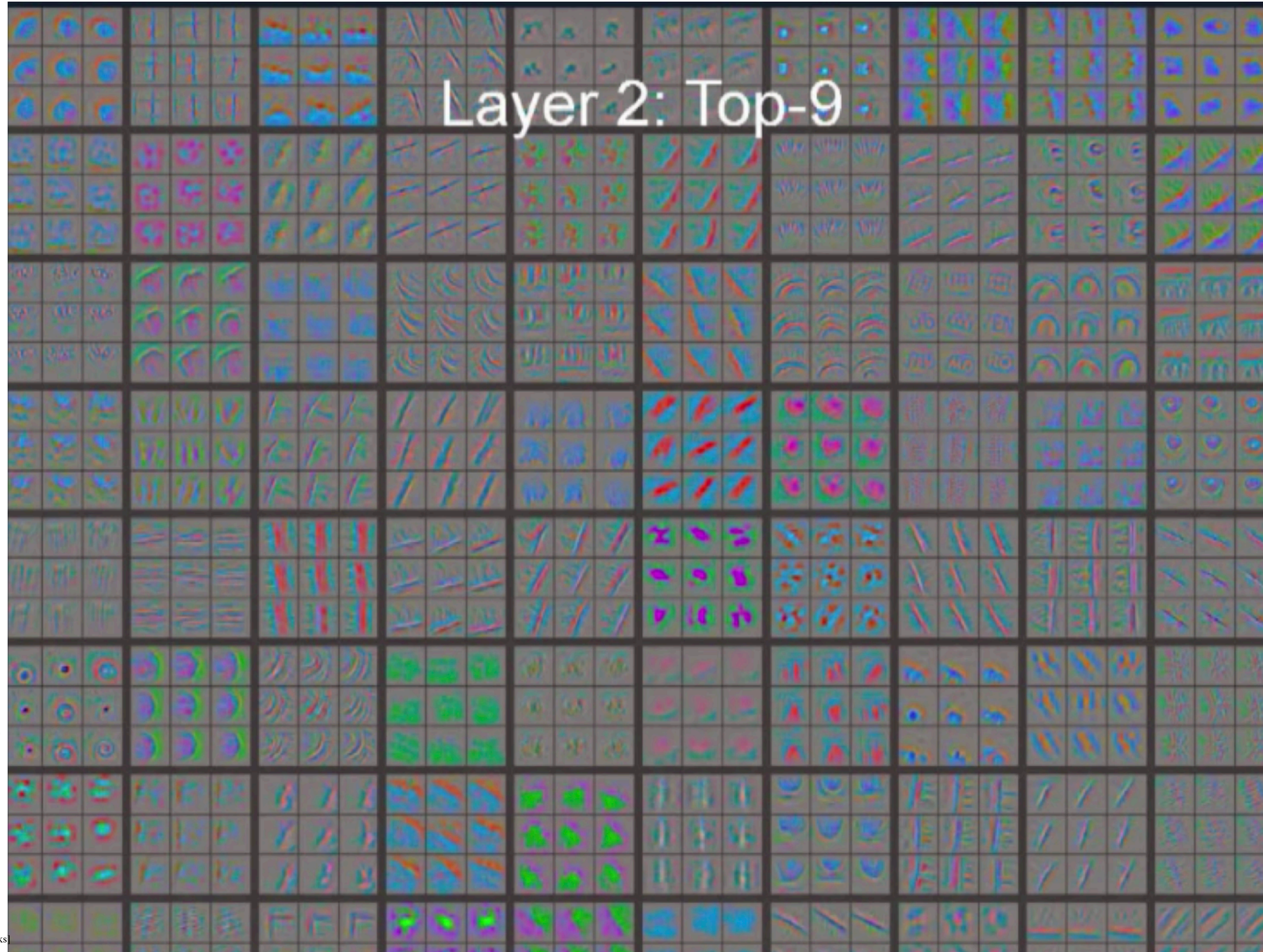


Kian Katanforoosh

[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

Results on a validation set of 50,000 images

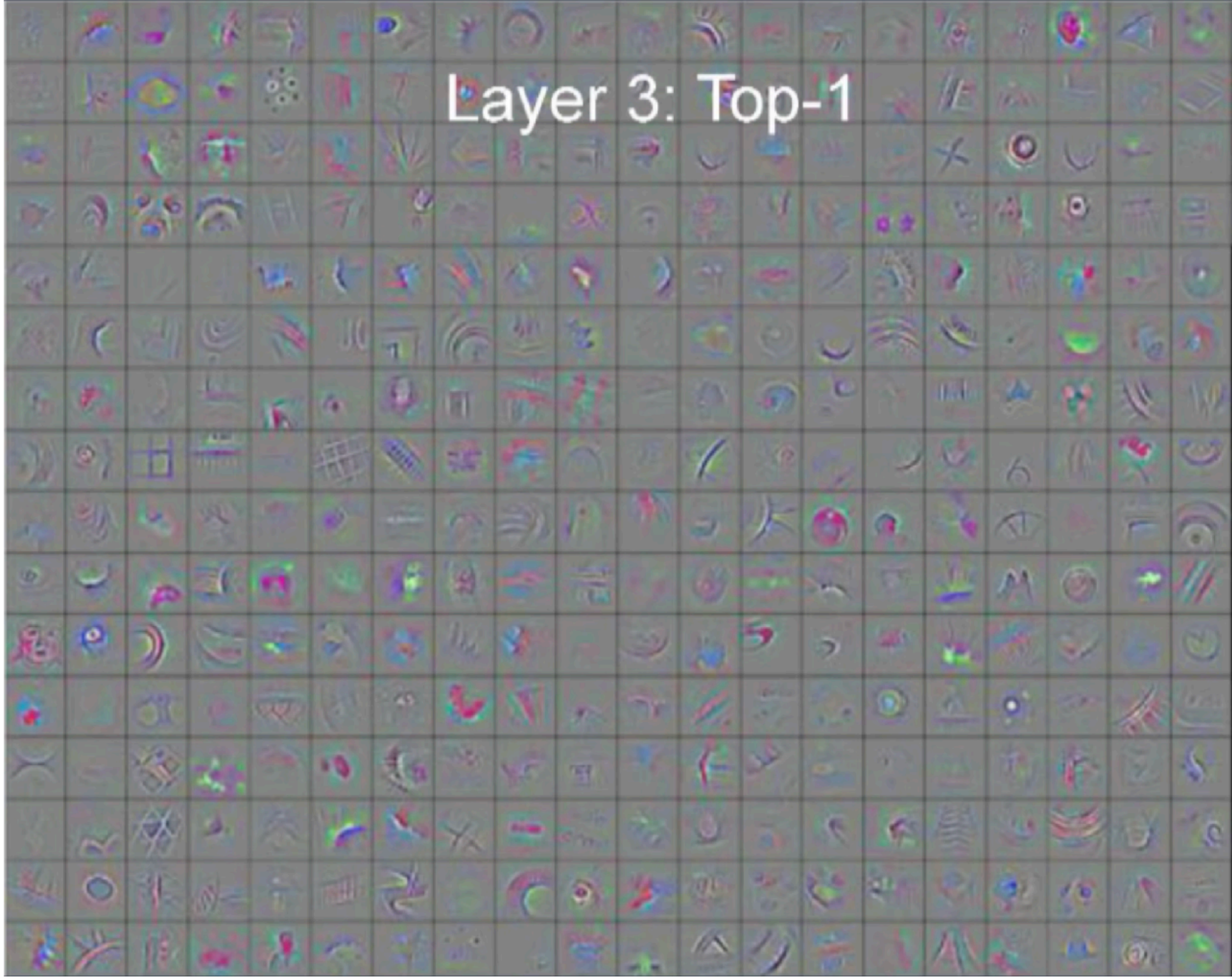
- Learning a more complex set of patterns than 1st layer edges
- Covers a much larger space of the image probably because of the pooling layer before.
- Features are more invariant to small changes. Ex: A dot, spiral, circle all fire the same 2nd layer feature very strongly



[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

Results on a validation set of 50,000 images

- 3rd layer: increased complexity
- An activated neuron is seeing ≈80x80 part of a 256x256 image
- Learning objects, faces etc..



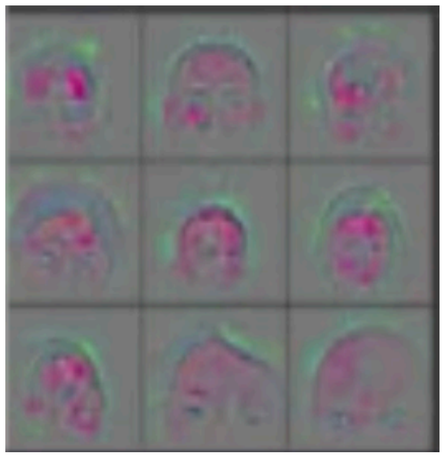
Kian Katanforoosh

[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]

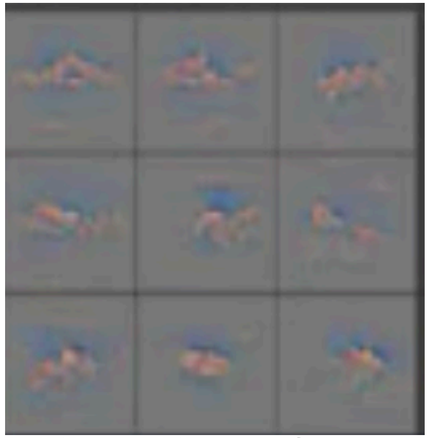
Results on a validation set of 50,000 images

- 3rd layer: increased complexity
- An activated neuron is seeing ≈80x80 part of a 256x256 image
- Learning objects, faces etc..
- Patches: Semantic grouping, not structural

Faces

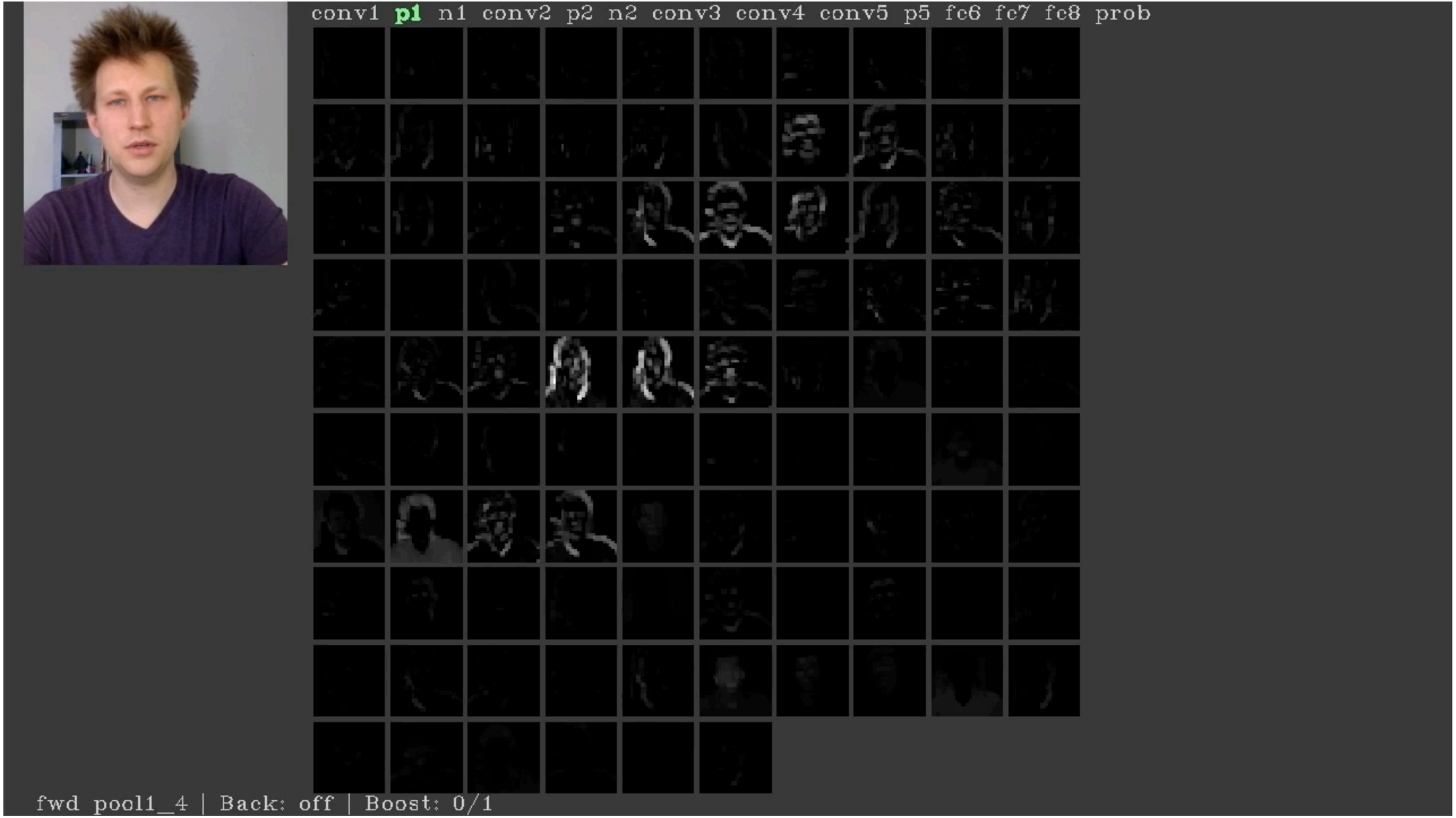


Clouds



Kian Katanforoosh

[Matthew D. Zeiler and Rob Fergus (2013): Visualizing and Understanding Convolutional Networks]



[Link to video: https://www.youtube.com/watch?v=AgkfIQ4IGaM]
[Jason Yosinski et al. (2015): Understanding Neural Networks Through Deep Visualization]

Questions we are now able to answer:

- *What part of the input is responsible for the output?*
    - *Occlusion sensitivity*
    - *Class Activation Maps*

- *What is the role of a given neuron/filter/layer?*
    - *Deconvolutions can help visualize the role of a neuron*
    - *Search dataset images maximizing the activation*
    - *Gradient ascent (class model visualization)*

- *Can we check what the network focuses on given an input image?*
    - *Occlusion sensitivity*
    - *Saliency maps (one-time gradient ascent)*
    - *Class Activation Maps*

- *How does a neural network see our world?*
    - *Gradient ascent (class model visualization)*
    - *Deep Dream*

- *Do these visualization have use cases?*
    - *Segmentation (saliency maps)*
    - *Art (Deep Dream)*

Today's outline

I. Case Study

II. CNN Interpretation

   A. with saliency maps

   B. with occlusion sensitivity

   C. with class activation maps (Global Average Pooling)

   D. with gradient ascent (class model visualization)

   E. with dataset search

   F. the deconvolution and its applications

III. Modern representation analysis

IV. Training & scaling diagnostics

V. Capabilities & safety dashboards

VI. Data diagnostics

VII. Closing Remarks

How Transformers Represent Information

In CNNs, we visualize edges, textures, and shapes. In modern language models, we visualize **relationships** and **meaning**.

Transformers represent language using two simple ideas:

1. Attention (what the model focuses on)

- Attention connects each word to the earlier words that matter.
- It behaves like a relevance heatmap: darker = more influence.
- Helps the model handle structure, references, and long-range dependencies.
- Attention maps are the Transformer equivalent of saliency maps for CNNs.

2. Embeddings (the model's map of meaning)

- Every word is converted into a vector inside the model.
- Similar words live close together (verbs with verbs, numbers with numbers).
- This is how the model organizes concepts internally.

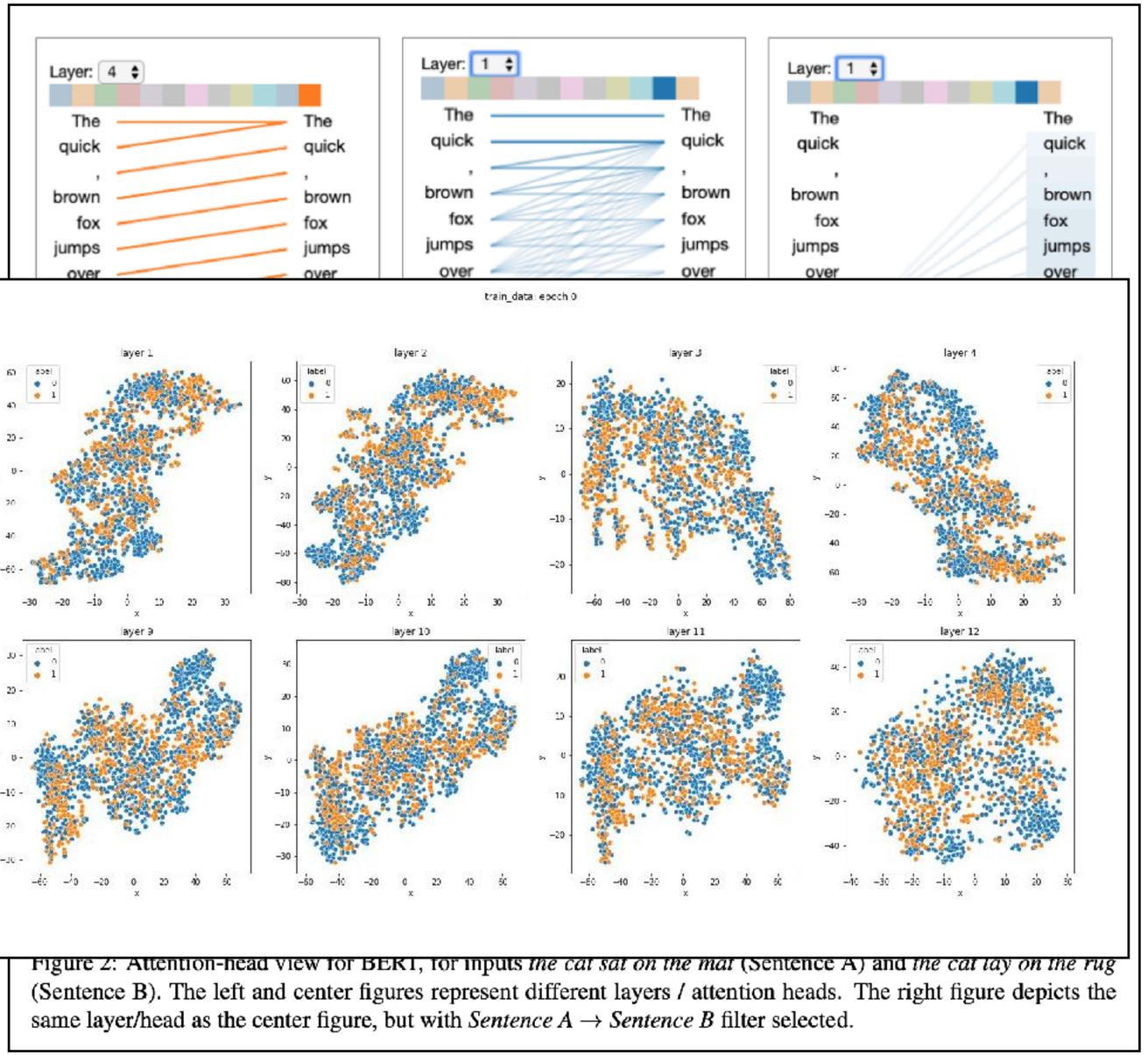


Figure 2: Attention-head view for BERT, for inputs *the cat sat on the mat* (Sentence A) and *the cat lay on the rug* (Sentence B). The left and center figures represent different layers / attention heads. The right figure depicts the same layer/head as the center figure, but with *Sentence A* → *Sentence B* filter selected.

Together, attention + embeddings let large models track relationships and meaning.

[Vaswani et al., 2017: Attention Is All You Need]
[Jesse Vig, 2019: A Multiscale Visualization of Attention in the Transformer Model] [Tanmay Garg, 2021: Visualize BERT sequence embeddings: An unseen way]

Kian Katanforoosh

If you're curious about how researchers analyze large models in more detail, Anthropic has written two of the clearest explainers in the field:

1. A Mathematical Framework for Transformer Circuits (2023): A higher-level, conceptual overview of how different components inside a transformer interact.

2. In-Context Learning and Induction Heads (2023): A simple introduction to how some attention heads learn patterns like copying previous words or tracking repeated sequences.

Today's outline

I. Case Study
II. CNN Interpretation
    A. with saliency maps
    B. with occlusion sensitivity
    C. with class activation maps (Global Average Pooling)
    D. with gradient ascent (class model visualization)
    E. with dataset search
    F. the deconvolution and its applications
III. Modern representation analysis
IV. Training & scaling diagnostics
V. Capabilities & safety dashboards
VI. Data diagnostics
VII. Closing Remarks

How Labs Check If a Model Is “Training Well”

- **Loss curves**: Do training and validation loss decrease as expected? Sudden jumps or plateaus often mean something is wrong (data issues, optimization issues).
- **Scaling laws**: Is it true that: more data + more compute → better performance? If a new checkpoint falls below the predicted curve, something might be off.
- **Training telemetry**: Signals like: gradient norm, learning rate schedule, model efficiency (how well GPUs are used) are monitored to catch instabilities early.

Together, these form a “health dashboard” for the model.

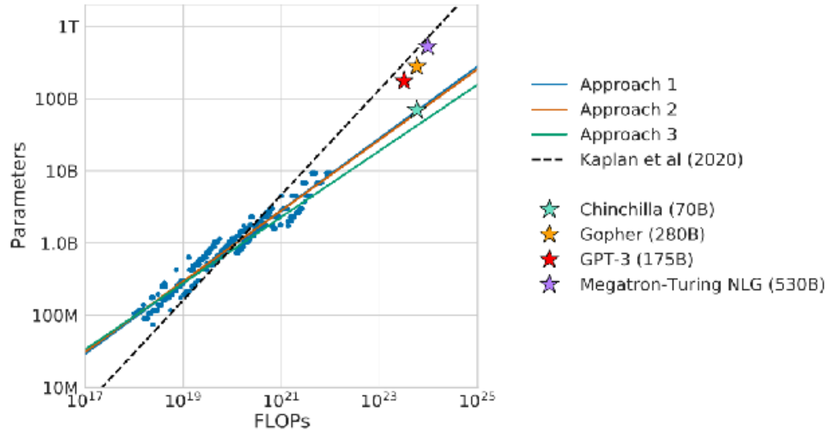


Figure 1 | **Overlaid predictions.** We overlay the predictions from our three different approaches, along with projections from Kaplan et al. (2020). We find that all three methods predict that current large models should be substantially smaller and therefore trained much longer than is currently done. In Figure A3, we show the results with the predicted optimal tokens plotted against the optimal number of parameters for fixed FLOP budgets. ***Chinchilla* outperforms *Gopher* and the other large models (see Section 4.2).**

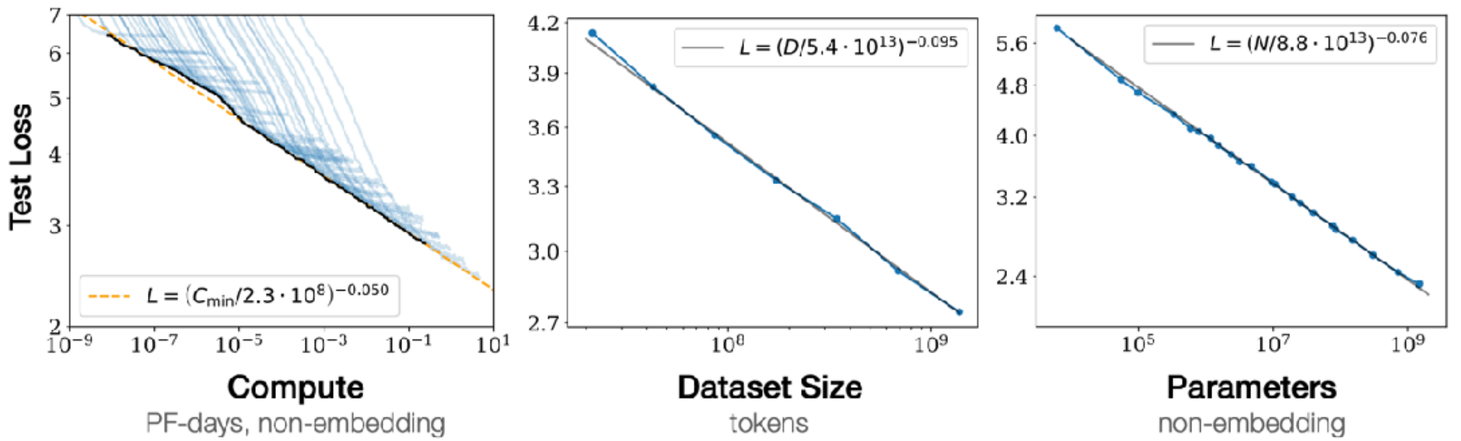


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

[Training Compute-Optimal Large Language Models, DeepMind (2022)]
[Scaling Laws for Neural Language Models, OpenAI (2020)]

Today's outline

I. Case Study
II. CNN Interpretation
    A. with saliency maps
    B. with occlusion sensitivity
    C. with class activation maps (Global Average Pooling)
    D. with gradient ascent (class model visualization)
    E. with dataset search
    F. the deconvolution and its applications
III. Modern representation analysis
IV. Training & scaling diagnostics
V. Capabilities & safety dashboards
VI. Data diagnostics
VII. Closing Remarks

How Labs Evaluate Model Capabilities & Safety

- **Capability Benchmarks**: These show what the model *can do* across different skills.

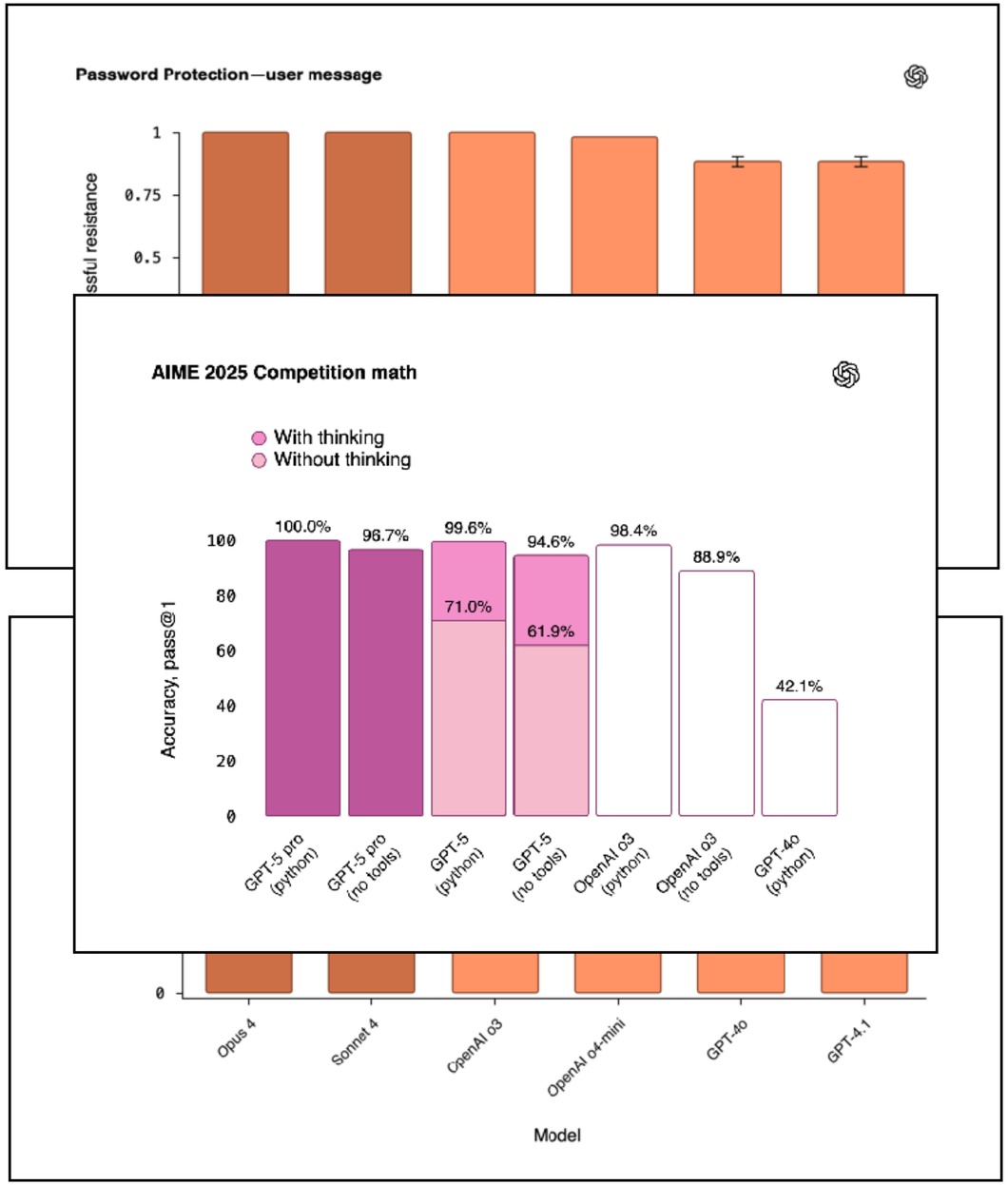
  Examples: reasoning, coding, math, translation, knowledge, instruction following, etc.

  You can compare checkpoints to see where performance improves or regresses.

- **Safety Evaluations:** These show how the model behaves under stress tests.

  Examples: jailbreak attempts, harmful content generation, misinformation, bias / fairness, privacy risks, etc.

Together, these dashboards tell us what's going well and what needs attention before release.



[GPT-4 Technical Report, OpenAI (2023)] [GPT-5 Technical Report, OpenAI (2025)]
[OpenAI + Anthropic Joint Safety Evaluation (2024)]

Kian Katanforoosh

Today's outline

I. Case Study
II. CNN Interpretation
    A. with saliency maps
    B. with occlusion sensitivity
    C. with class activation maps (Global Average Pooling)
    D. with gradient ascent (class model visualization)
    E. with dataset search
    F. the deconvolution and its applications
III. Modern representation analysis
IV. Training & scaling diagnostics
V. Capabilities & safety dashboards
VI. Data diagnostics
VII. Closing Remarks

How Labs Detect Data Issues

- **Distribution checks**

  Did the mix of domains change?

  Example: less code, more social media → coding benchmark drops.

- **Token statistics**

  Are some tokens appearing far more or far less than before?

  Example: certain math tokens disappear → math scores regress.

- **Contamination checks**

  Did parts of the evaluation set accidentally leak into the training set?

  Contamination = artificially high scores + sudden regressions later.

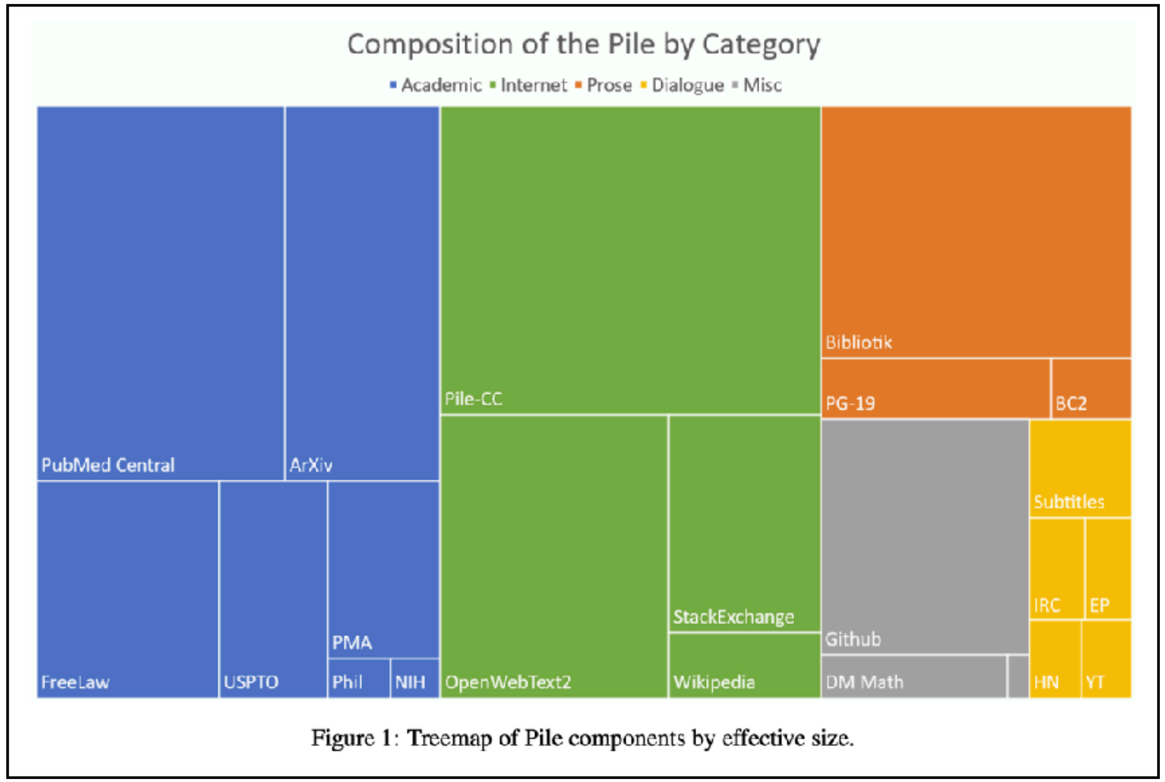


Figure 1: Treemap of Pile components by effective size.

[Gao et al. (2020): The Pile: An 800GB Dataset of Diverse Text for Language Modeling]

Duties for next week

Project Final Report Due
12/5/2025 (due 11:59 pm PST)
Instructions

Please read over the final project guidelines here for information on the rubric and late submissions.

Project Poster Session
12/10/2025 (Poster session, 12:15-3:15 pm)