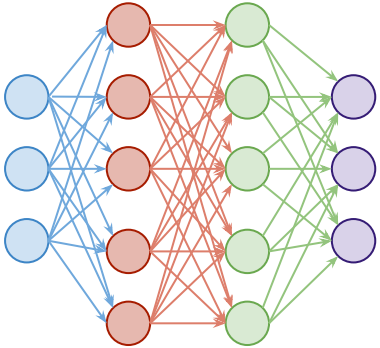
ME 536

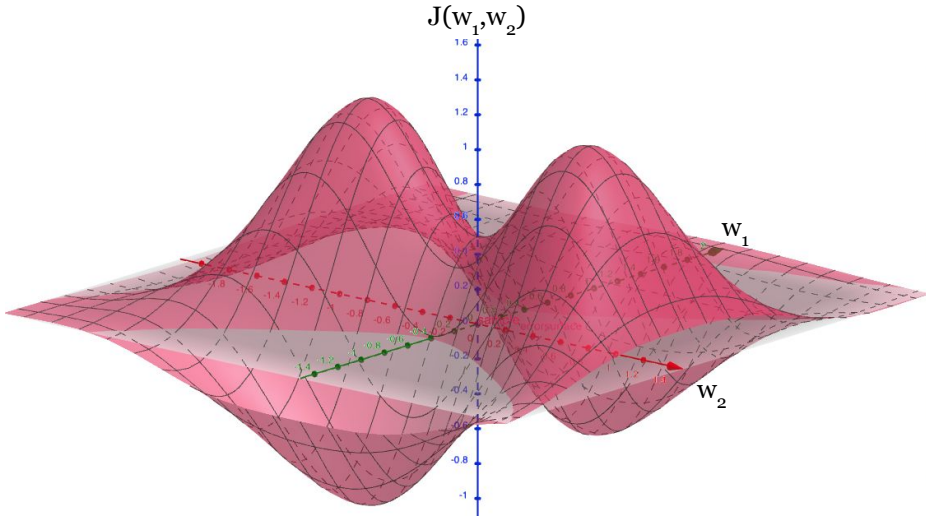
Week 13: How deep is your *learning network* ?

Recall: Fully Connected Networks

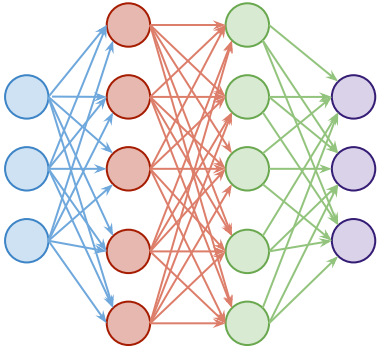


Too many parameters to tune?

The more parameters, the more complex the search space / surface



Recall: Fully Connected Networks

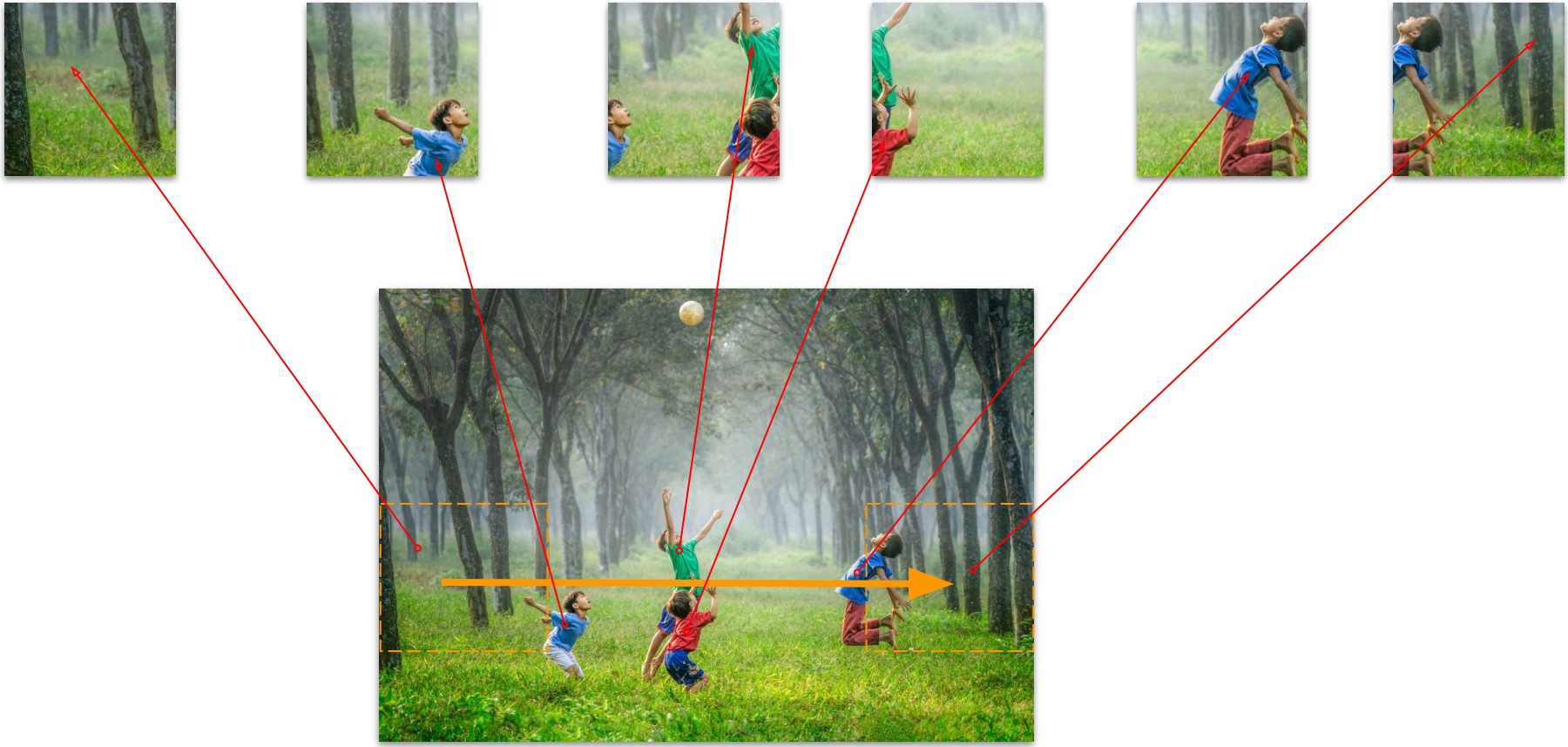


or DENSE networks

All outputs of a layer are connected to all inputs of the next.

Are all pixels in an image related?

Who cares: about that pixel

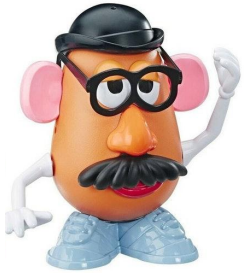


Seeing the whole: part by part

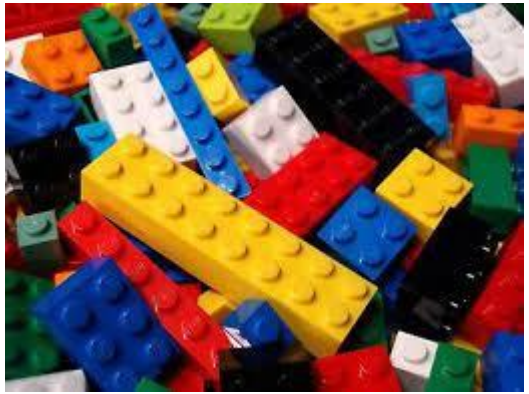
Old but still useful tactic: Divide and conquer

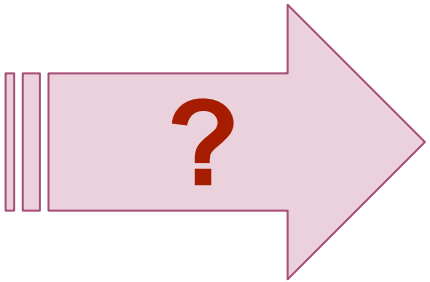
Old but still useful tactic: match and conquer

?

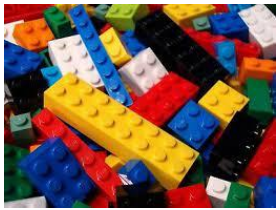


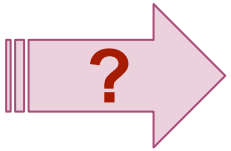
Reality Check: Divide and but how to conquer?





Hmmm: Divide and but how to conquer?

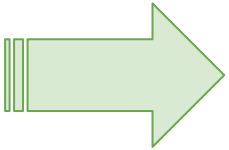


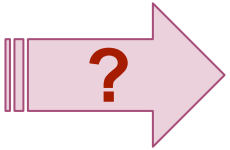


Note that the problem is:

- NOT about a path from piece to whole
- About
  - finding pieces given the whole
  - And identifying the whole form a subset of all possible pieces

May be: Divide and join to conquer





Tear down the model and by just looking at the pieces:

guess which model we had to start with?

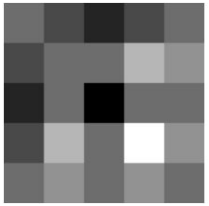
Note that you can form new small pieces by using the ones you have

Recall Convolution: Moving a window over an image

Original image that is 5x5

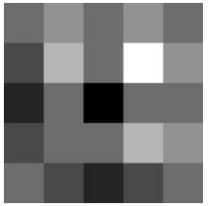
Kernel 1

-1.00	-1.00	-1.00
-1.00	1.00	1.00
-1.00	1.00	-1.00



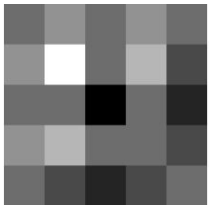
Kernel 2

-1.00	1.00	-1.00
-1.00	1.00	1.00
-1.00	-1.00	-1.00



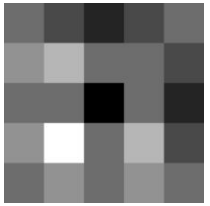
Kernel 3

-1.00	1.00	-1.00
1.00	1.00	-1.00
-1.00	-1.00	-1.00



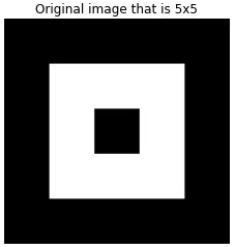
Kernel 4

-1.00	-1.00	-1.00
1.00	1.00	-1.00
-1.00	1.00	-1.00



More on Convolution: 1 kernel + ReLu + Pooling

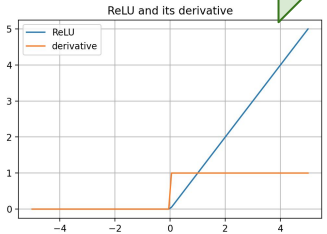
Image I → 5x5

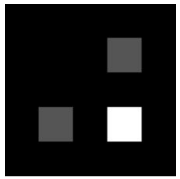



```
Kernel 1
|     -1.00     -1.00     -1.00 |
|     -1.00      1.00      1.00 |
|     -1.00      1.00     -1.00 |
```

feature map → 5x5

Non-linearity:**ReLU**





```
Feature Map after ReLU
|  0.00   0.00   0.00   0.00   0.00 |
|  0.00   0.00   0.00   1.00   0.00 |
|  0.00   0.00   0.00   0.00   0.00 |
|  0.00   1.00   0.00   3.00   0.00 |
|  0.00   0.00   0.00   0.00   0.00 |
```

Maxpooling:
3x3 into 1
with stride of 3

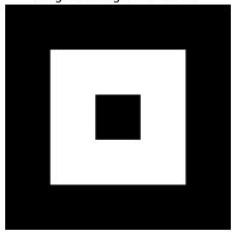
result → 2x2

```
Maxpooled
|  0.00   1.00 |
|  1.00   3.00 |
```

- **Max**
- Average
- ...

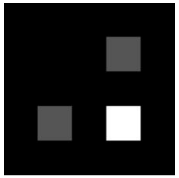
Convolution: Multiple Kernels + ReLu + Maxpooling

Image I convolved with `Kernel i` → feature map → ReLU → Maxpooling



Kernel 1

-1.00	-1.00	-1.00
-1.00	1.00	1.00
-1.00	1.00	-1.00

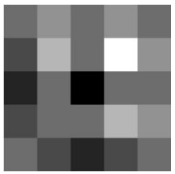


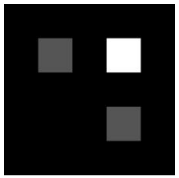
Maxpooled

0.00	1.00
1.00	3.00

Kernel 2

-1.00	1.00	-1.00
-1.00	1.00	1.00
-1.00	-1.00	-1.00



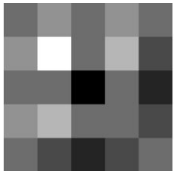


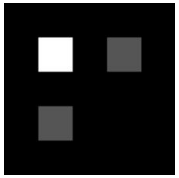
Maxpooled

1.00	3.00
0.00	1.00

Kernel 3

-1.00	1.00	-1.00
1.00	1.00	-1.00
-1.00	-1.00	-1.00



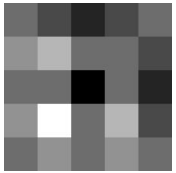


Maxpooled

3.00	1.00
1.00	0.00

Kernel 4

-1.00	-1.00	-1.00
1.00	1.00	-1.00
-1.00	1.00	-1.00

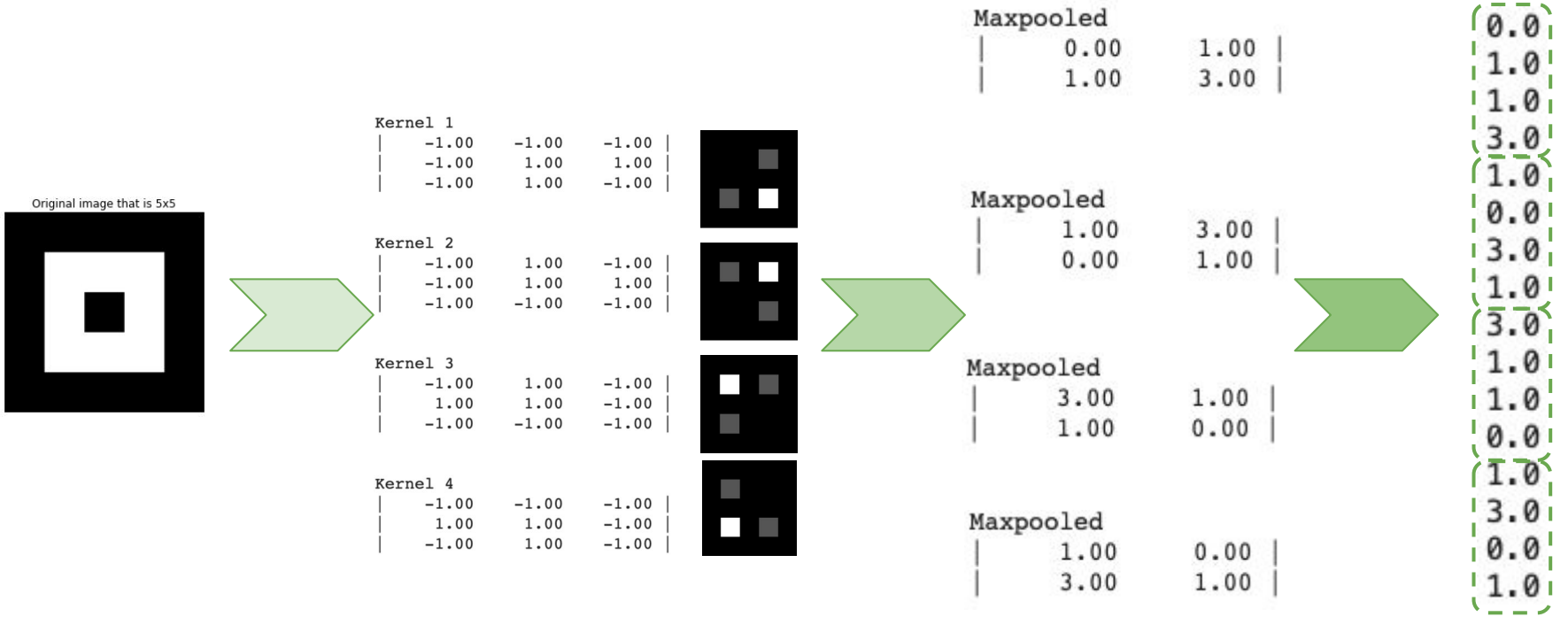


Maxpooled

1.00	0.00
3.00	1.00

More after Convolution: … feature vector …

Vectorize result:
i.e. ***flatten*** *the feature matrix*



More after Convolution: … feature vectors …

Kernel 1

-1.00	-1.00	-1.00
-1.00	1.00	1.00
-1.00	1.00	-1.00

Kernel 2

-1.00	1.00	-1.00
-1.00	1.00	1.00
-1.00	-1.00	-1.00

Kernel 3

-1.00	1.00	-1.00
1.00	1.00	-1.00
-1.00	-1.00	-1.00

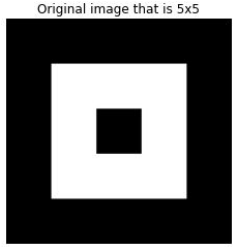
Kernel 4

-1.00	-1.00	-1.00
1.00	1.00	-1.00
-1.00	1.00	-1.00

0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
1.0	2.0	1.0	1.0	1.0	2.0	1.0	2.0
1.0	0.0	1.0	0.0	0.0	1.0	2.0	0.0
3.0	1.0	1.0	2.0	3.0	1.0	1.0	0.0
1.0	2.0	1.0	2.0	2.0	2.0	1.0	2.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3.0	0.0	1.0	0.0	2.0	1.0	2.0	0.0
1.0	1.0	0.0	2.0	1.0	1.0	0.0	0.0
3.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0
1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0
1.0	0.0	0.0	0.0	2.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	1.0	0.0	0.0	1.0	2.0	1.0
3.0	2.0	1.0	3.0	3.0	2.0	1.0	2.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0

Recall the workflow:

Image I → 5x5



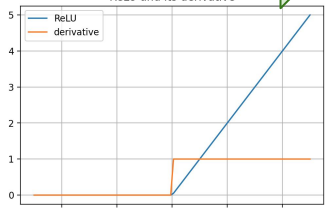

```
Kernel 1
|     -1.00     -1.00     -1.00 |
|     -1.00      1.00      1.00 |
|     -1.00      1.00     -1.00 |
```

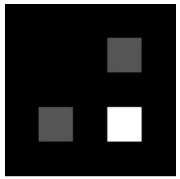
Convolve with kernels

feature map → 5x5

ReLu

Activation Func:**ReLU**





```
Feature Map after ReLU
|  0.00   0.00   0.00   0.00   0.00 |
|  0.00   0.00   0.00   1.00   0.00 |
|  0.00   0.00   0.00   0.00   0.00 |
|  0.00   1.00   0.00   3.00   0.00 |
|  0.00   0.00   0.00   0.00   0.00 |
```

+ Pooling

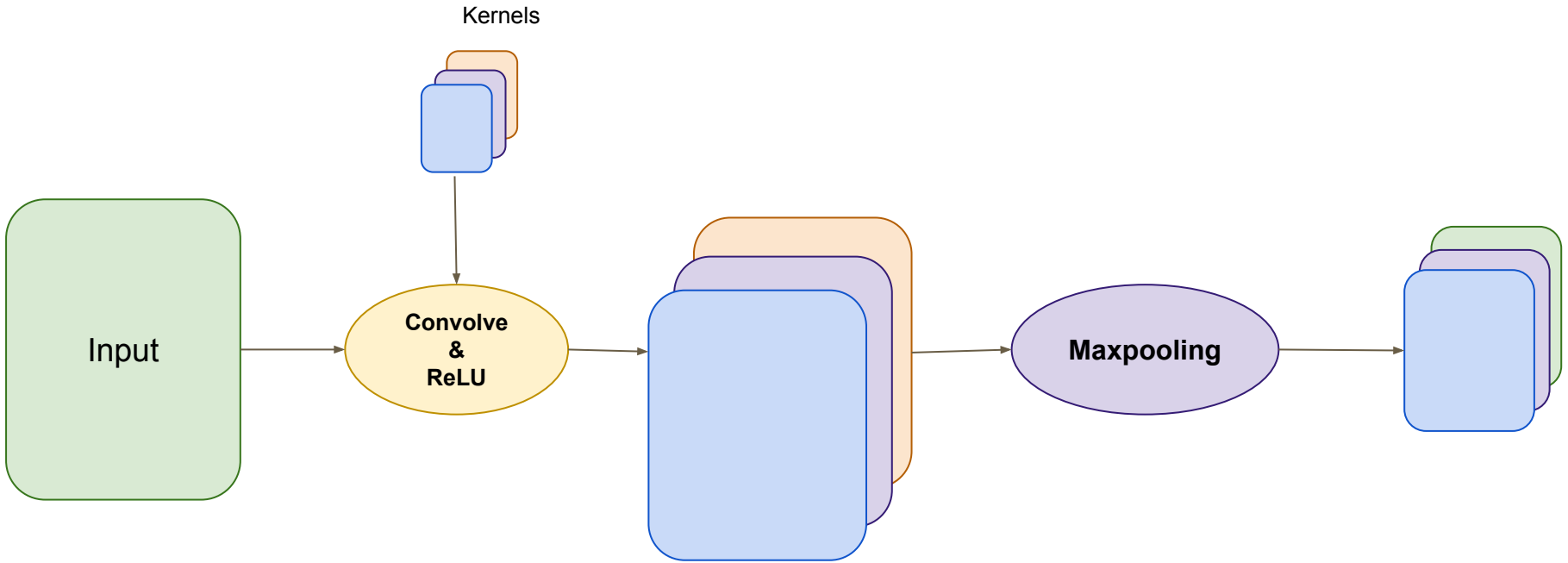
- **Max**
- Average
- ...

Maxpooling:
3x3 into 1
with stride of 3

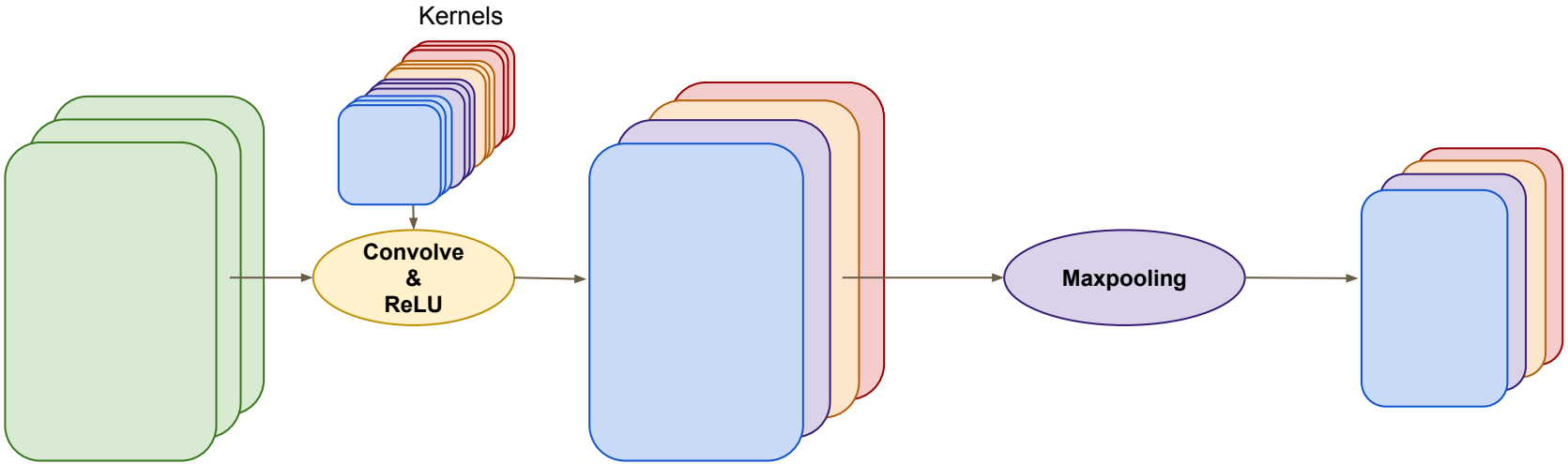
result → 2x2

```
Maxpooled
|  0.00   1.00 |
|  1.00   3.00 |
```

CNN layers: a typical case at the input layer



CNN layers: a typical case after input



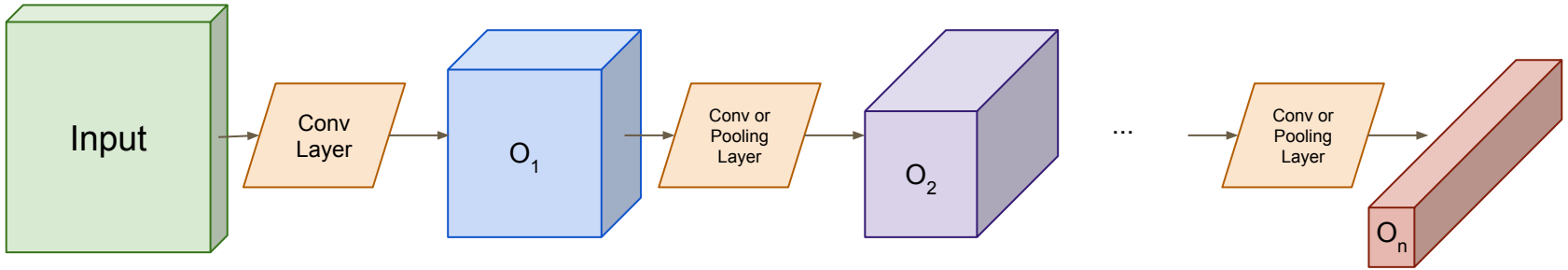
Observe that input of a layer gets a **smaller** footprint while getting **taller**

Also note that this might be the input layer for an RGB image

A deep CNN: feature extractor

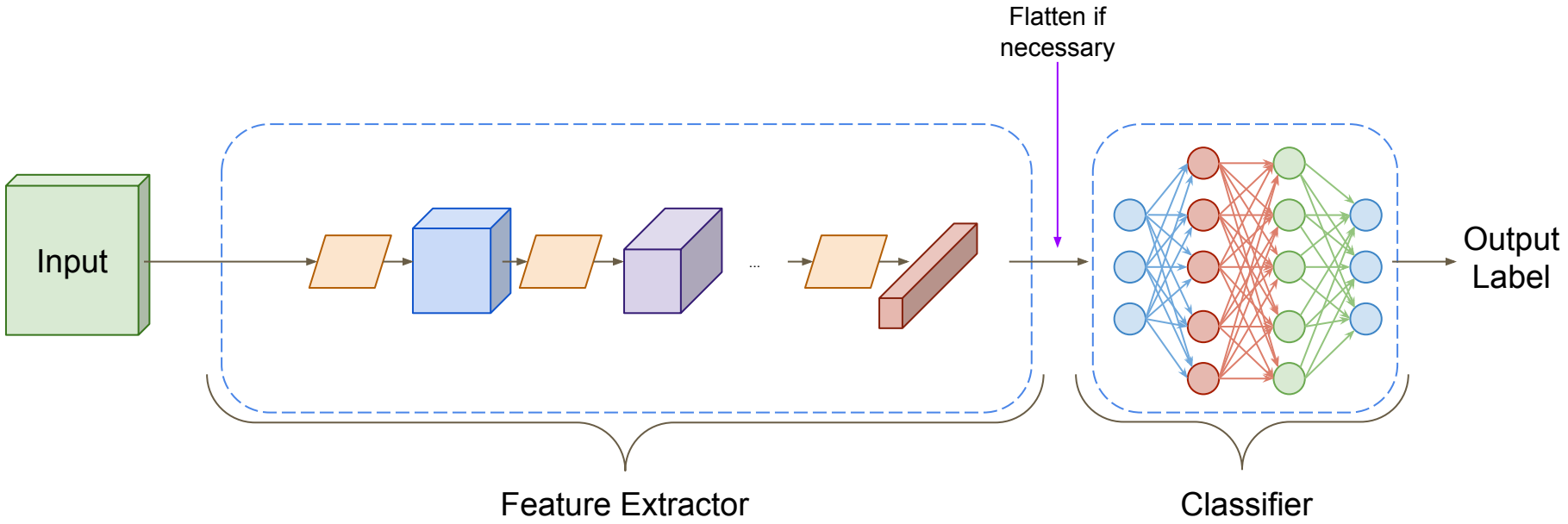
After *n* layers (convolutional or pooling) we expect *a vectorized output* form the CNN, i.e. **a feature vector**

If not, *flatten* the *feature tensor* into the *feature vector*.

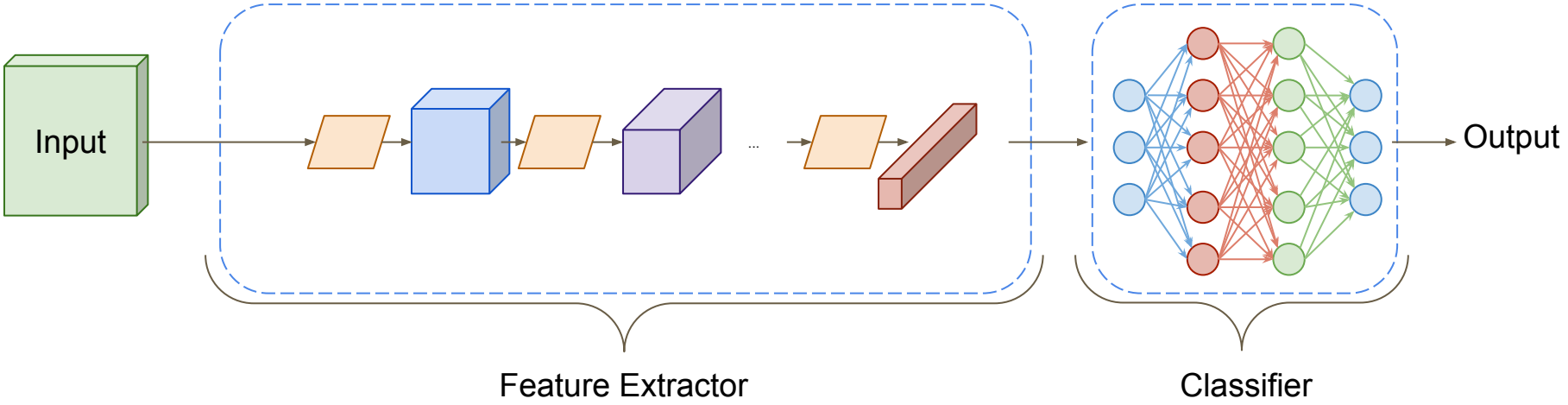


Note that both a convolutional and pooling layers can use stride other than 1 and hence can scale the input down

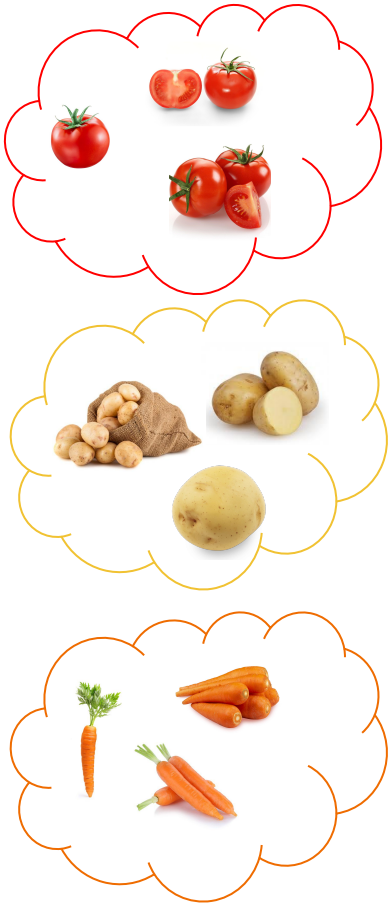
A typical deep CNN



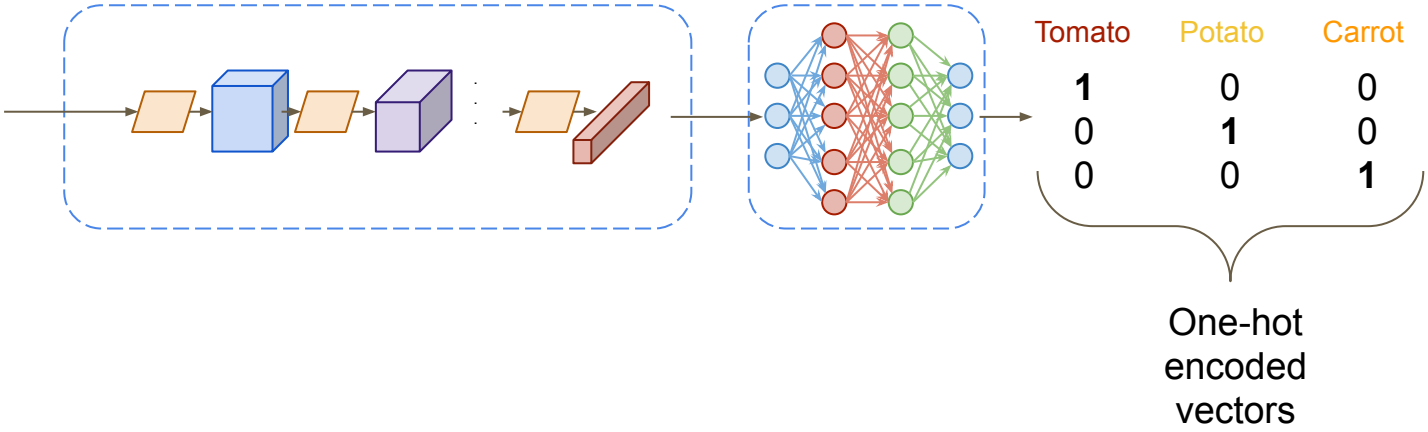
A typical deep CNN



Training a deep CNN: Inputs and expected output

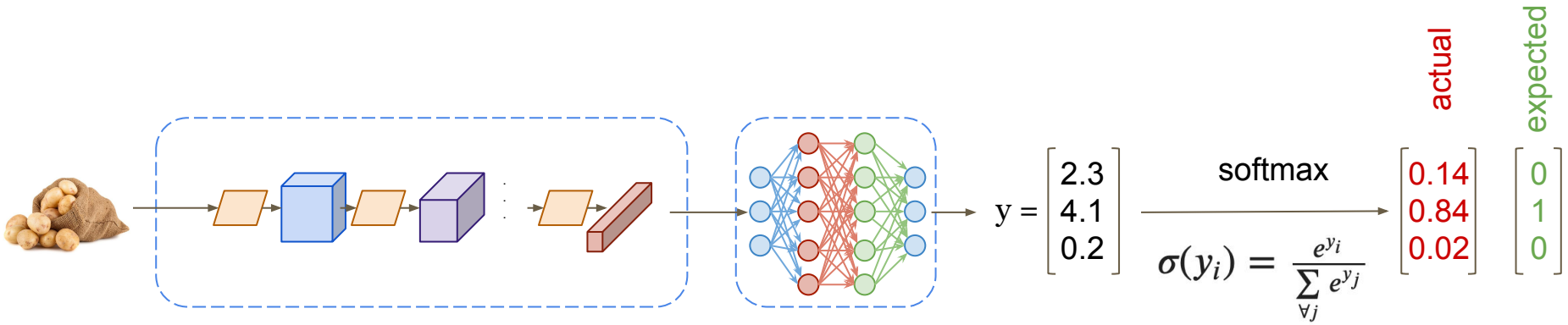


During training a ground truth / correct labels are provided with every input



Using a trained deep CNN: hope for the best

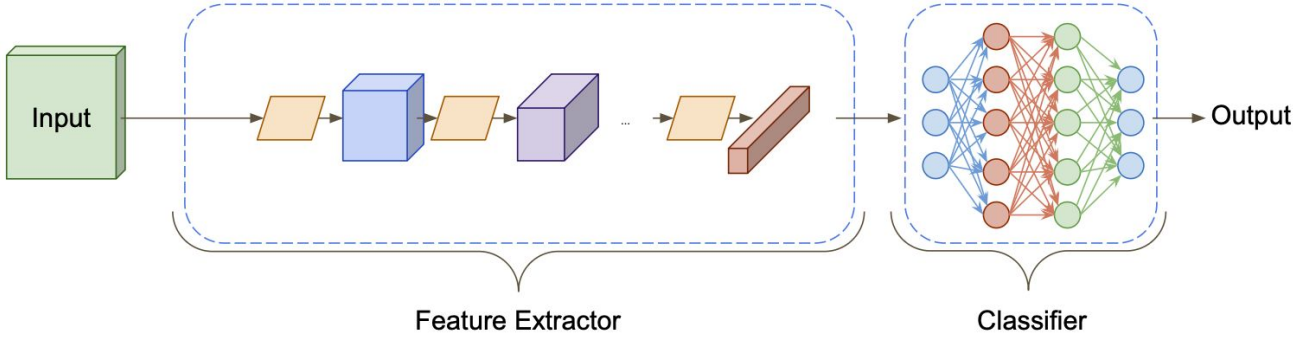
A FCN (fully connected network) outputs a vector of values when an image is presented at the input which is *not necessarily* one-hot encoded



Note that:
After softmax sum of the output vector is 1. Result is more like a PDF

In practice:
Top label or top n labels are used to assess the success of the trained network

More on CNNs...



Following links are provided to assist you in your project...

and also to show you the breadth and depth of this topic

yet we are touch the very basics in a nutshell today...

- Deep Lizard tutorials
- Deep learning course videos @ MIT

Recall: Linear Algebra

Let

$$\mathbf{AB} = \mathbf{A}$$

where $\mathbf{B}$ is not identity and neither matrices are *null*

What are possible $\mathbf{B}$s?

Among all possible ones, which one(s) are more preferable?

Recall: Linear Algebra

Let

$$\mathbf{AB} = \mathbf{A}$$

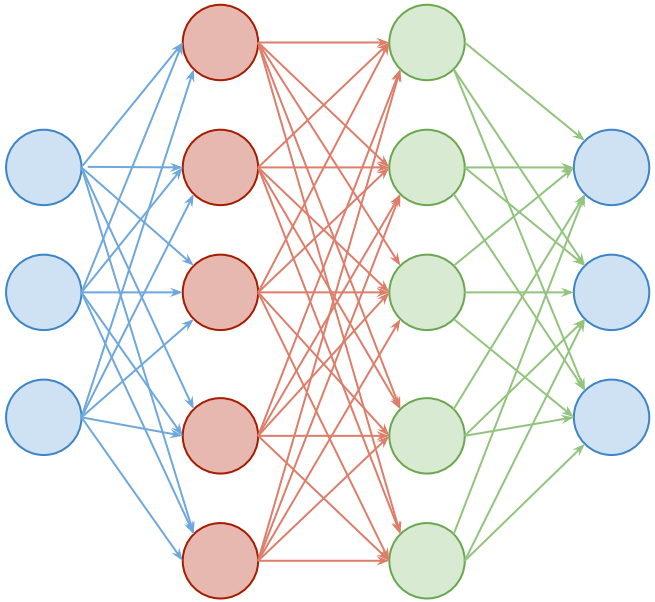
Assume columns of $\mathbf{A}$ are coming from union of 2 subspaces?

A weird network: output learns to match input

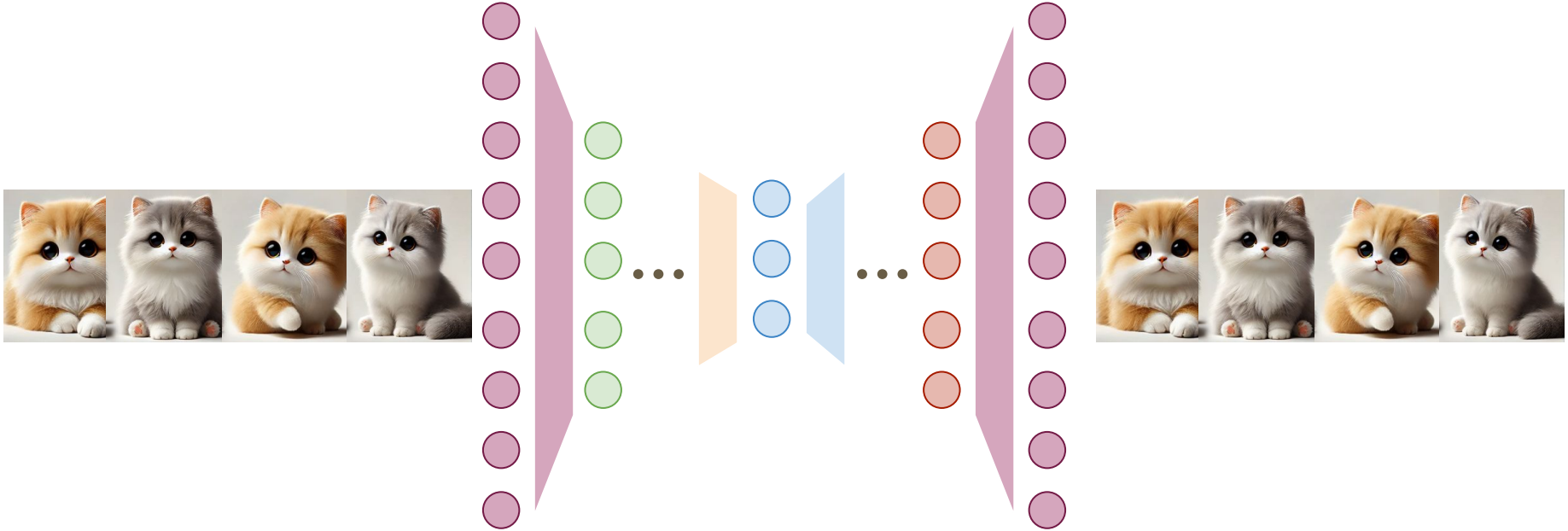
Say what?

Why on earth?

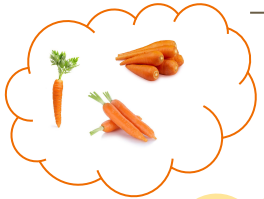
What good does it do?



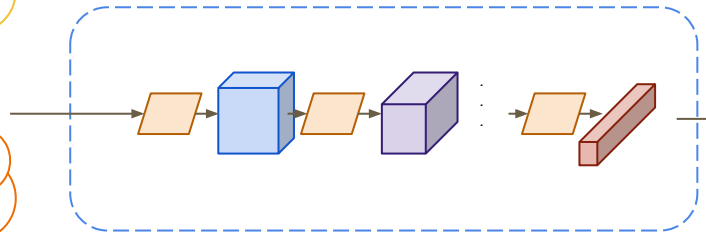
A weird network: How about this one

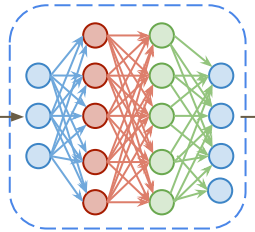


Transfer Learning: transfer what?



After training with 3 classes, a new class of object can be added
Training with already trained network yields better results

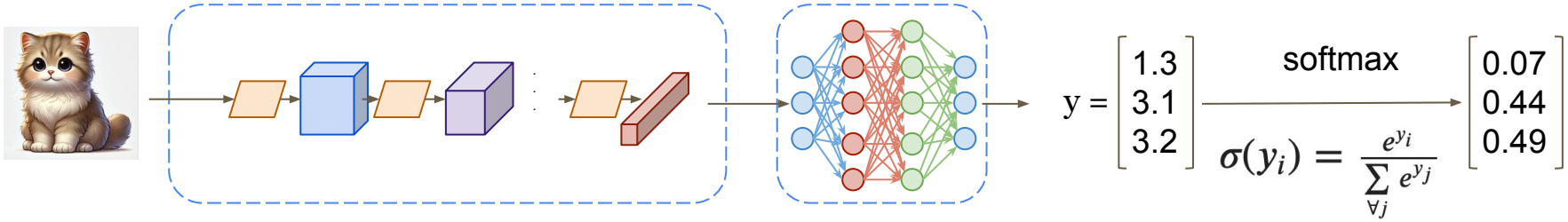




Tomato	Potato	Carrot	Durian
1	0	0	0
0	**1**	0	0
0	0	**1**	0
0	0	0	**1**

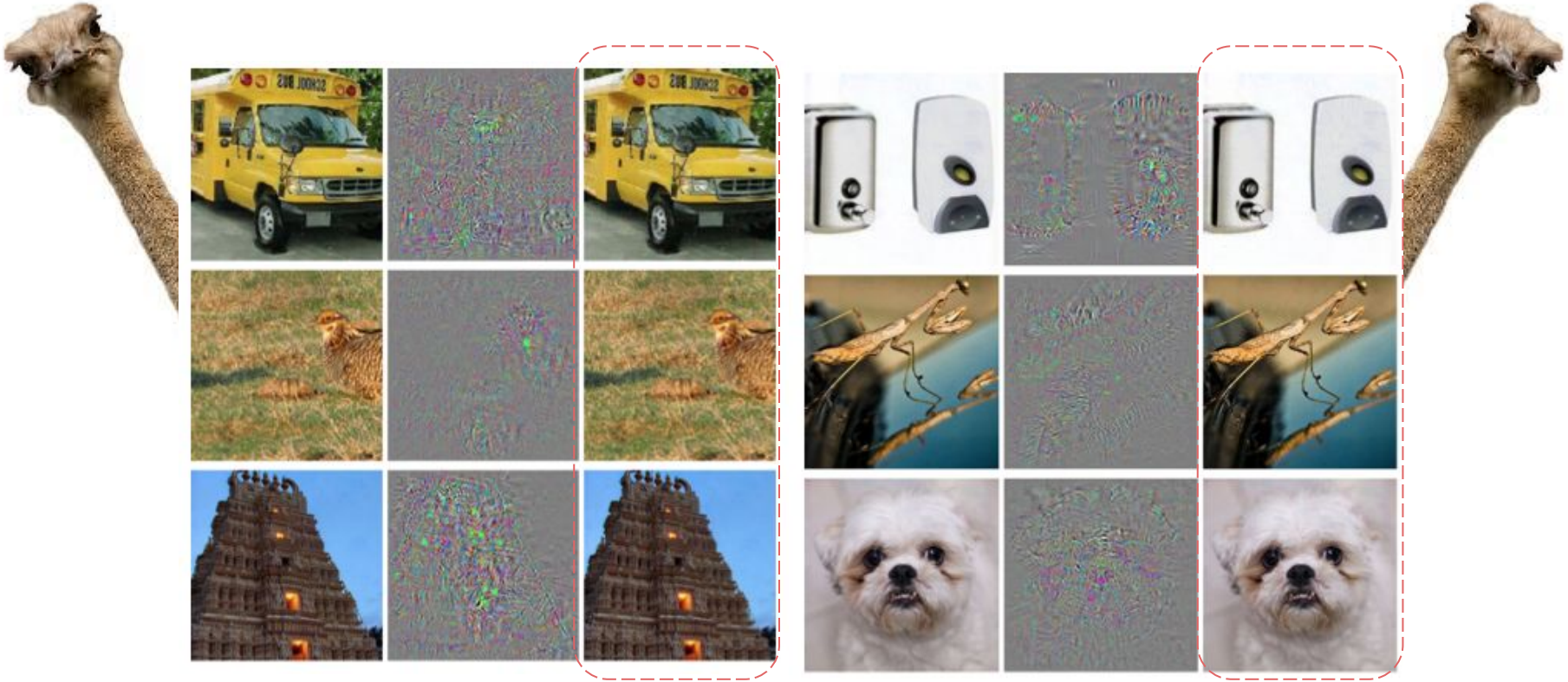
What if: network sees something *NEW*

Network will still output some value !!!
Shame on you deep network



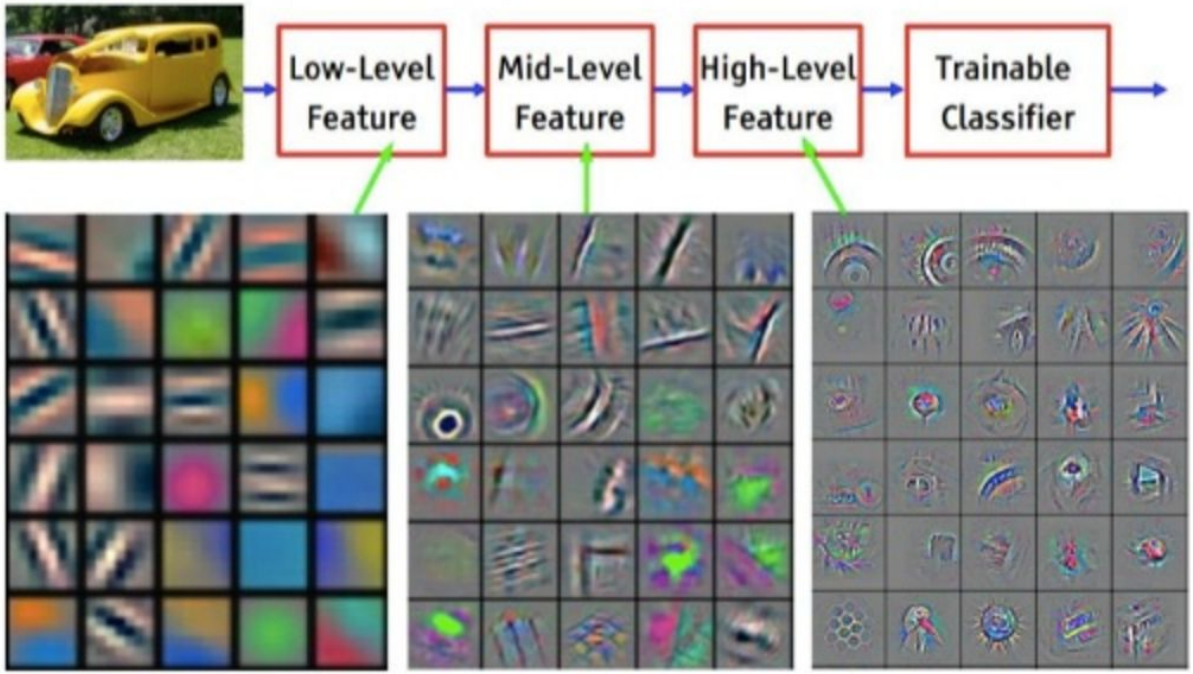
Hot question:
How can we detect new?

A shameless case: adversarial examples



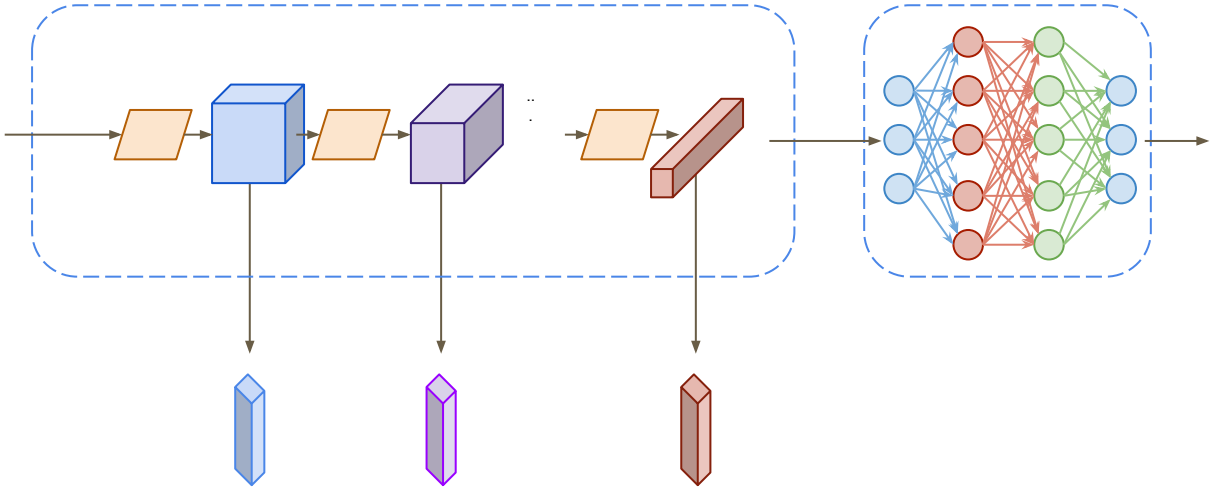
https://arxiv.org/pdf/1312.6199.pdf

What is happening: to kernels as it learns



https://towardsdatascience.com/what-makes-deep-learning-different-from-traditional-machine-learning-methods-fc154db33939

What if: output is not at the output?

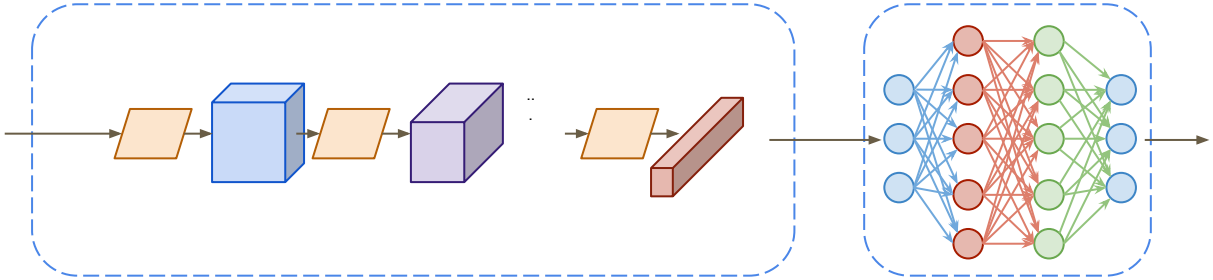


Hot question:

Can the output layer be too specific?

What if we are not interested in the output but an intermediate output?

What if: a network has seen everything?



Hot question:

Can a CNN be a UNIVERSAL feature extractor?

Self-learning follows...

The project will require you to:

- Define a problem
- Select proper approach
  - Learn tools for the proposed approach
  - Implement and demonstrate

Datasets are already available: Free practice

An easy way to start testing is using existing publicly available datasets.

tensorflow: https://www.tensorflow.org/datasets/catalog/overview

pyTorch: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html

NOT to be continued...

This was the last official lecture