

NUS-ISS

Pattern Recognition using Machine Learning System



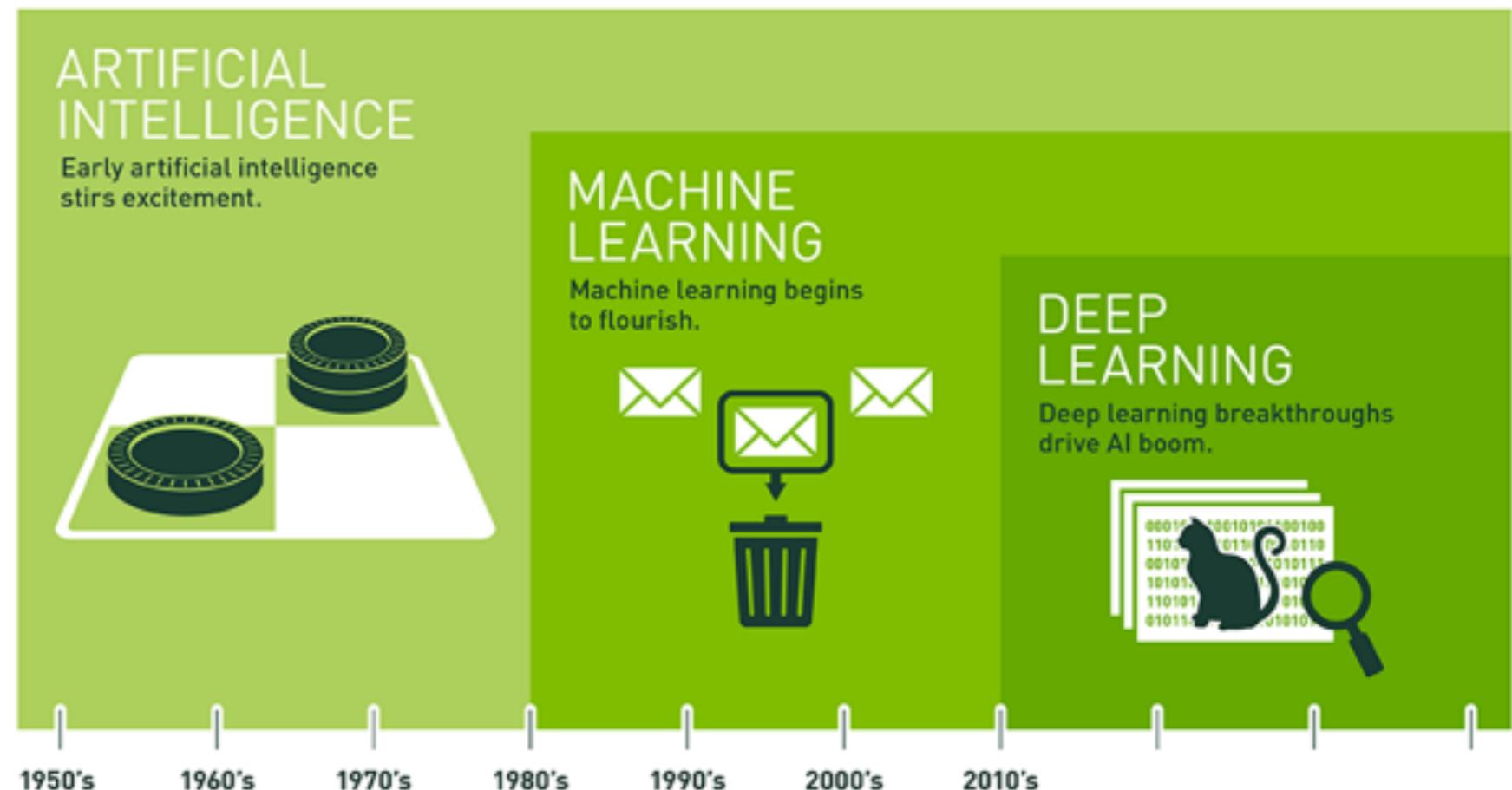
The rise of machine-learned features

by Dr. Tan Jen Hong

© 2019 National University of Singapore.
All Rights Reserved.

The AI time line

A broad overview

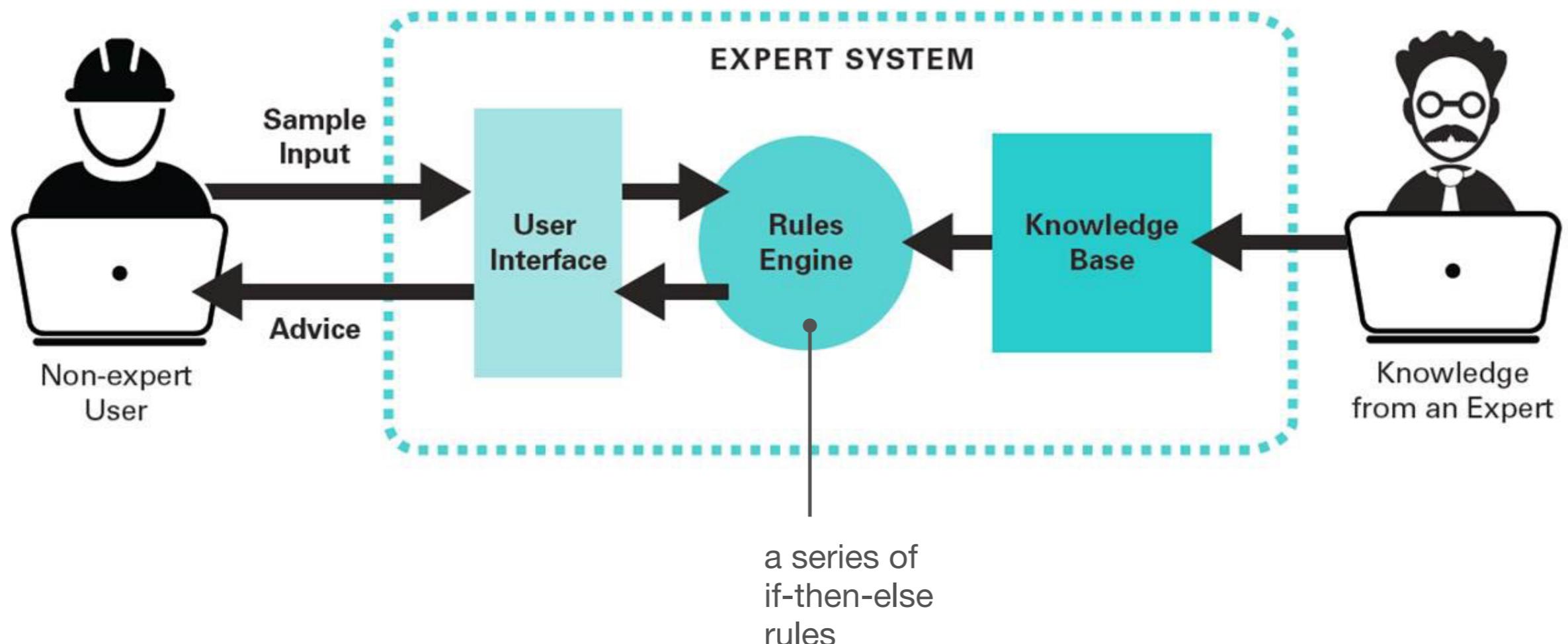


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

The AI time line

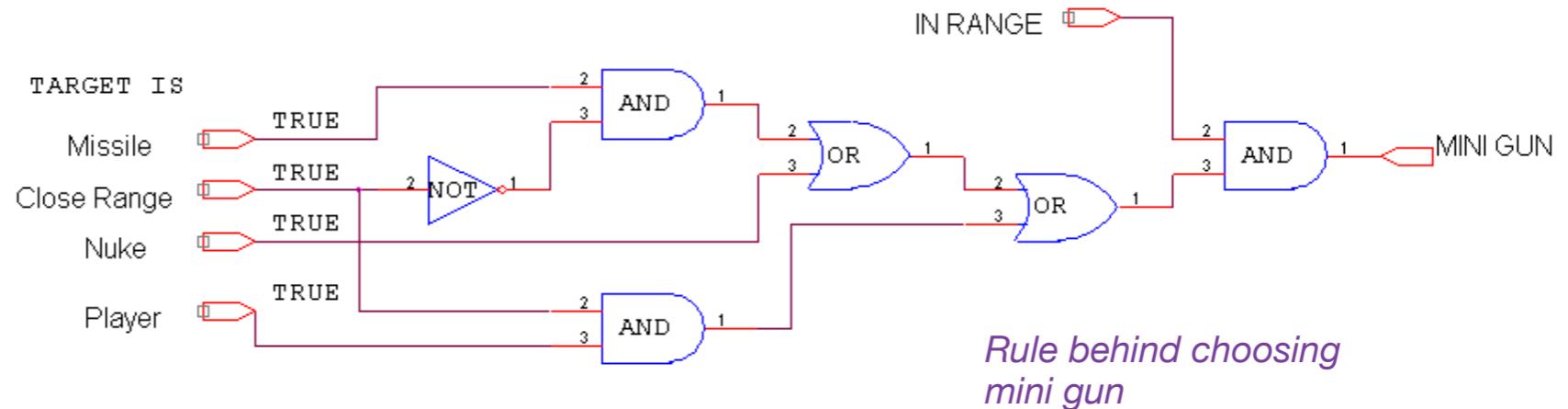
The starting point ...



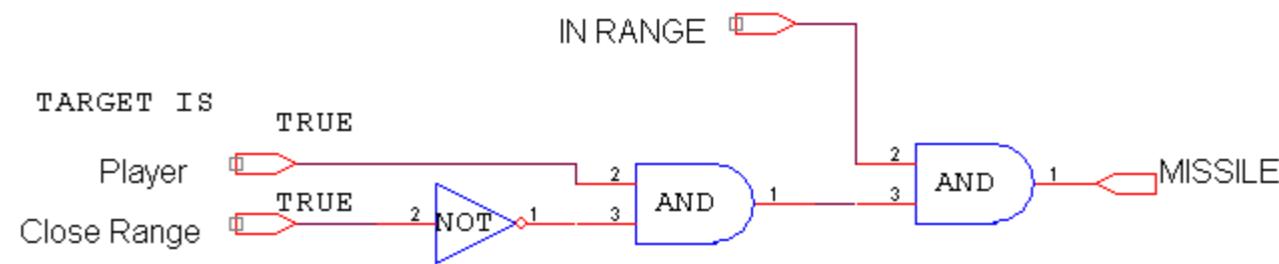
Source: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

Expert system

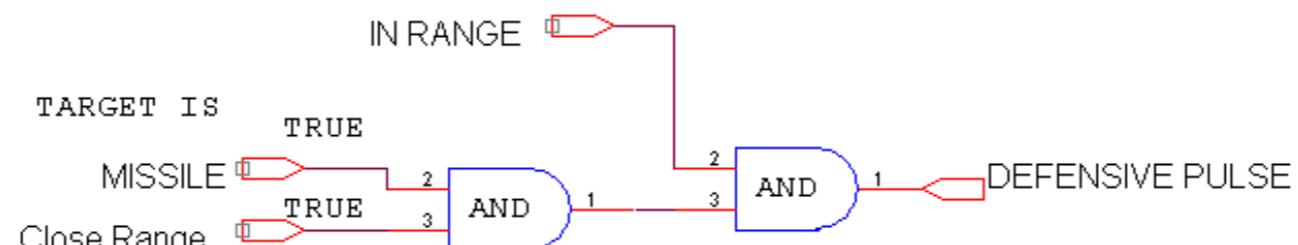
Example: Arriving decision to choose a weapon at any time by the expert system



Rule behind choosing mini gun



Rule behind choosing missile



Rule behind choosing defensive pulse

Source: <https://wiki.bath.ac.uk/display/BISAI/Expert+System+in+a+Gaming+Enviroment>

Expert system

Rules to determine ...
a yacht?

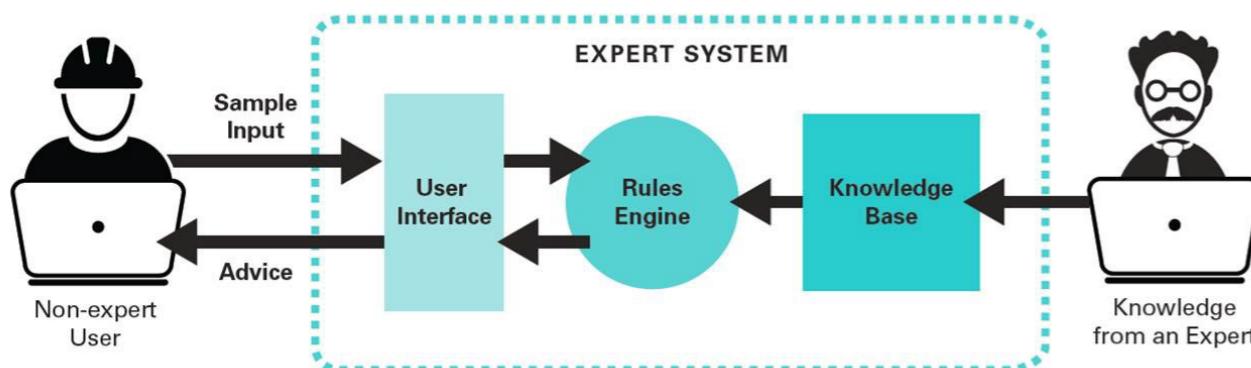


Source: https://yachtharbour.com/news/venus-spotted-in-mallorca-1887?src=news_view_page_bar

Expert system

The limitations

- Expensive and time consuming!
Experts are never cheap
- Bad in handling sophisticated sensory inputs (like signals, images)
- Possible to make dumb decision since it just goes through rules; no common sense in the system
- System not easy to be updated



Source: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

Features + classifier

A new solution to the rescue

- Feature: a number or a vector that describes something about the input

- Classifier figures out (by itself) the underlying pattern between features and output

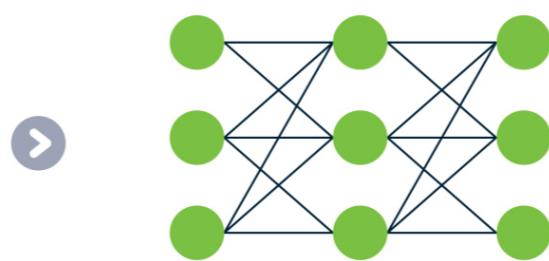
- This is where the 'learning' happens



Input



Feature extraction



Classification



Output

Source: <https://verhaert.com/difference-machine-learning-deep-learning/>

Filtering or feature extraction

Filtering

It's all about convolution

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Source: <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1>

Filtering

What filtering can do



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



$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



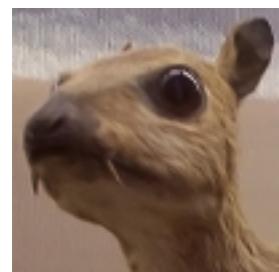
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



$$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$



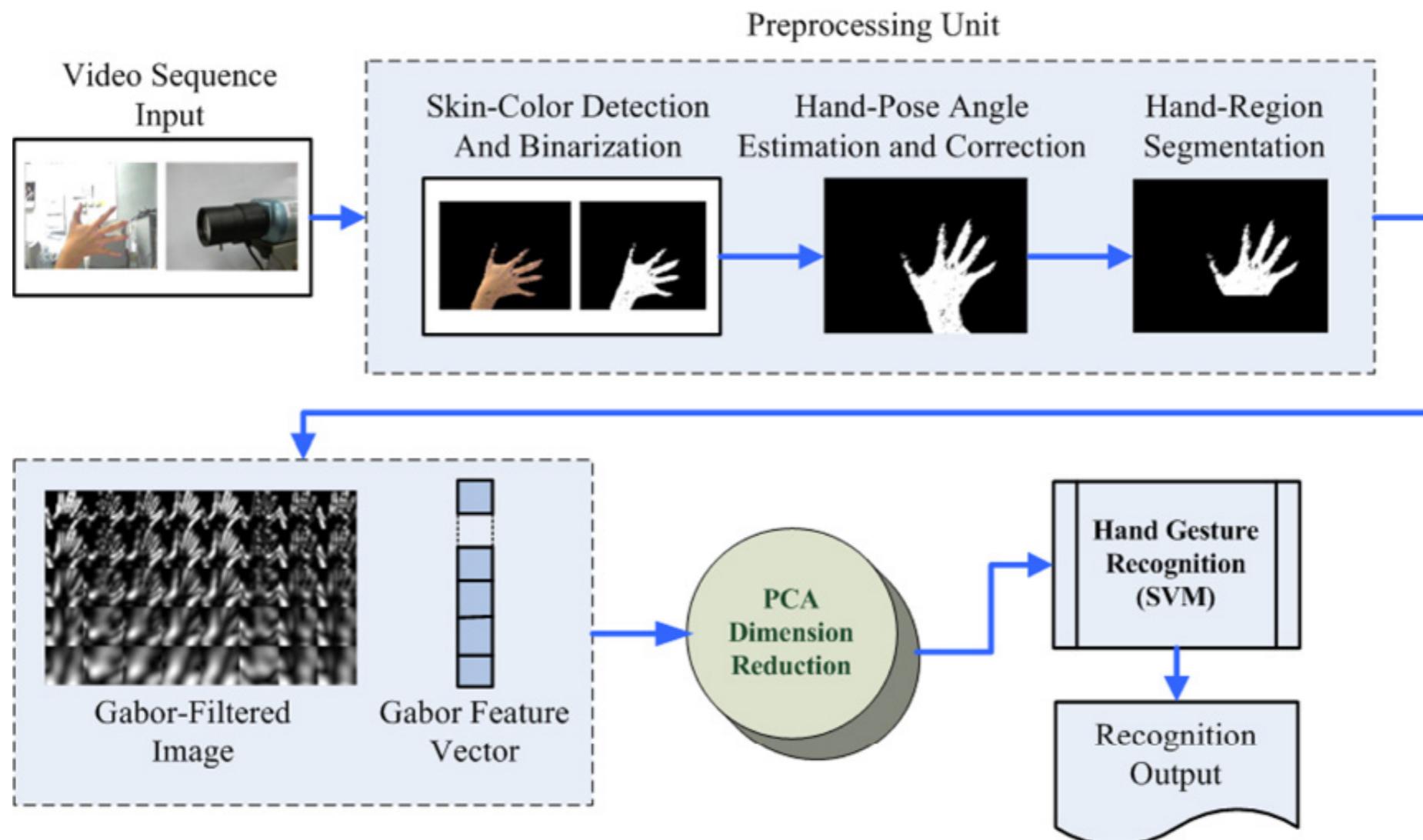
$$\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$



Source: [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

Features + classifier

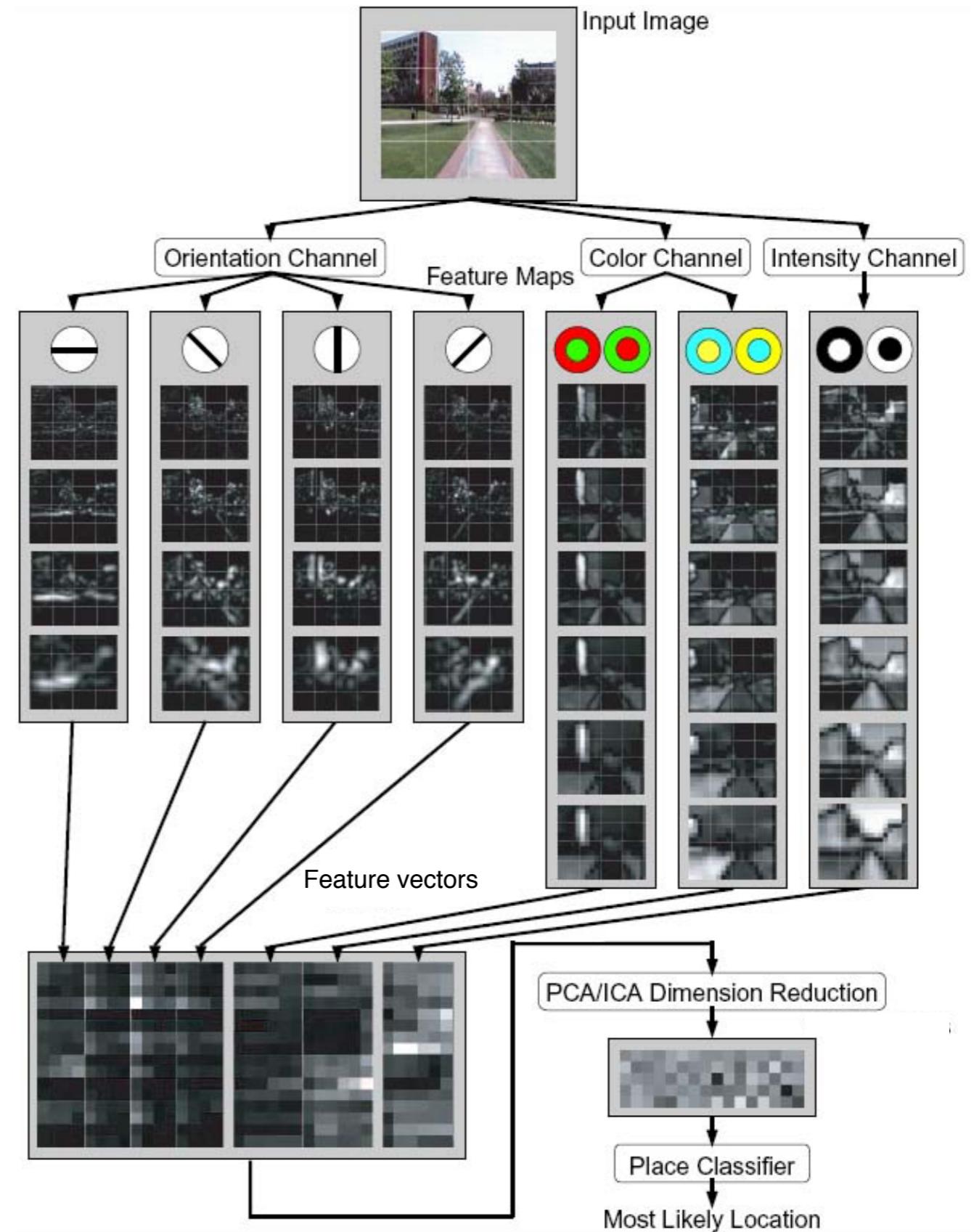
Gabor filtering



Source: <https://doi.org/10.1016/j.eswa.2010.11.016>

Features + classifier

Plethora of features for scene recognition



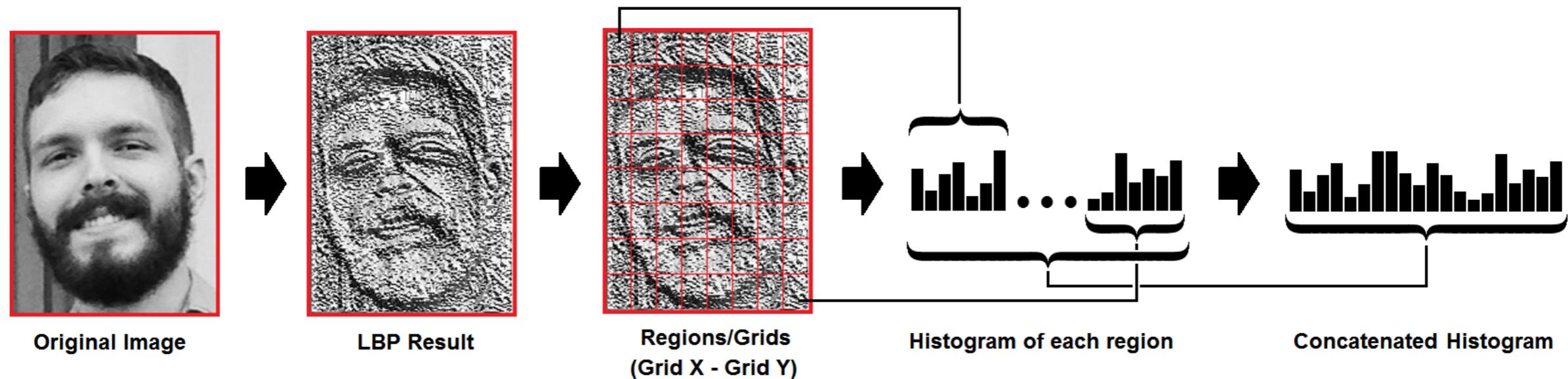
Source: <http://ilab.usc.edu/siagian/Research/Gist/Gist.html>

Features + classifier

Features?

- Example: Local Binary Pattern

- Perform face recognition



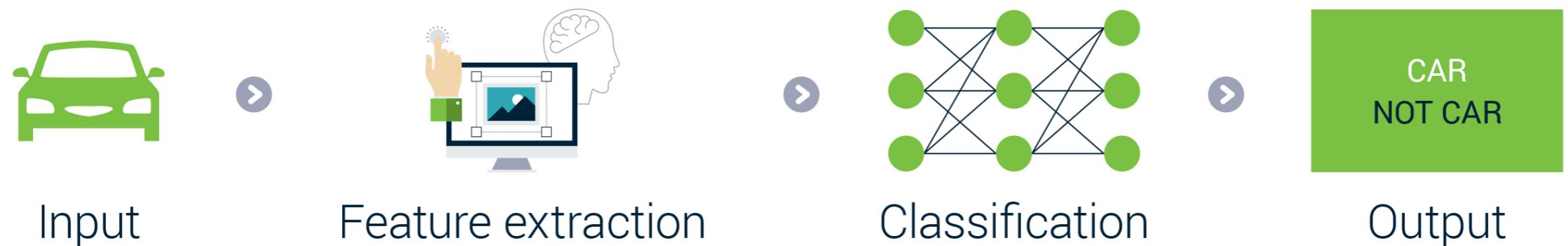
Source: <https://towardsdatascience.com/face-recognition-how-lbp-works-90ec258c3d6b>

Features + classifier

Conclusion?

- We need to design features manually, through much trial and error, with luck

- Classifiers used are generic (like SVM)

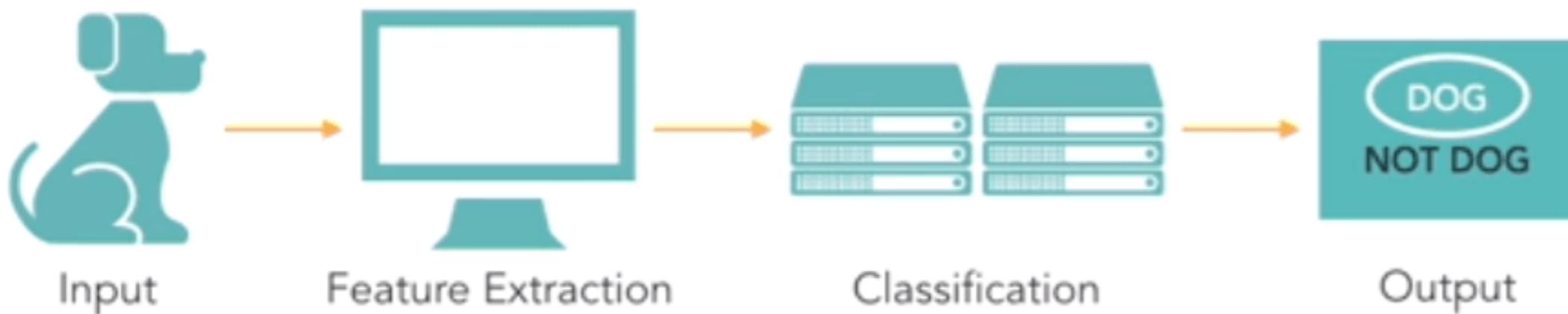


Source: <https://verhaert.com/difference-machine-learning-deep-learning/>

Features + classifier

Conclusion?

- Progress in recognition accuracy powered by better features
- Plethora hand-crafted features proposed and used, such as HOG, SIFT, LBP and etc...
- But what next? Come out more new features? Better classifiers?



Source: <https://www.guru99.com/machine-learning-vs-deep-learning.html>

Learning the features

Better performance?

- Instead of we deciding the features, get algorithm to learn the most appropriate features by itself?

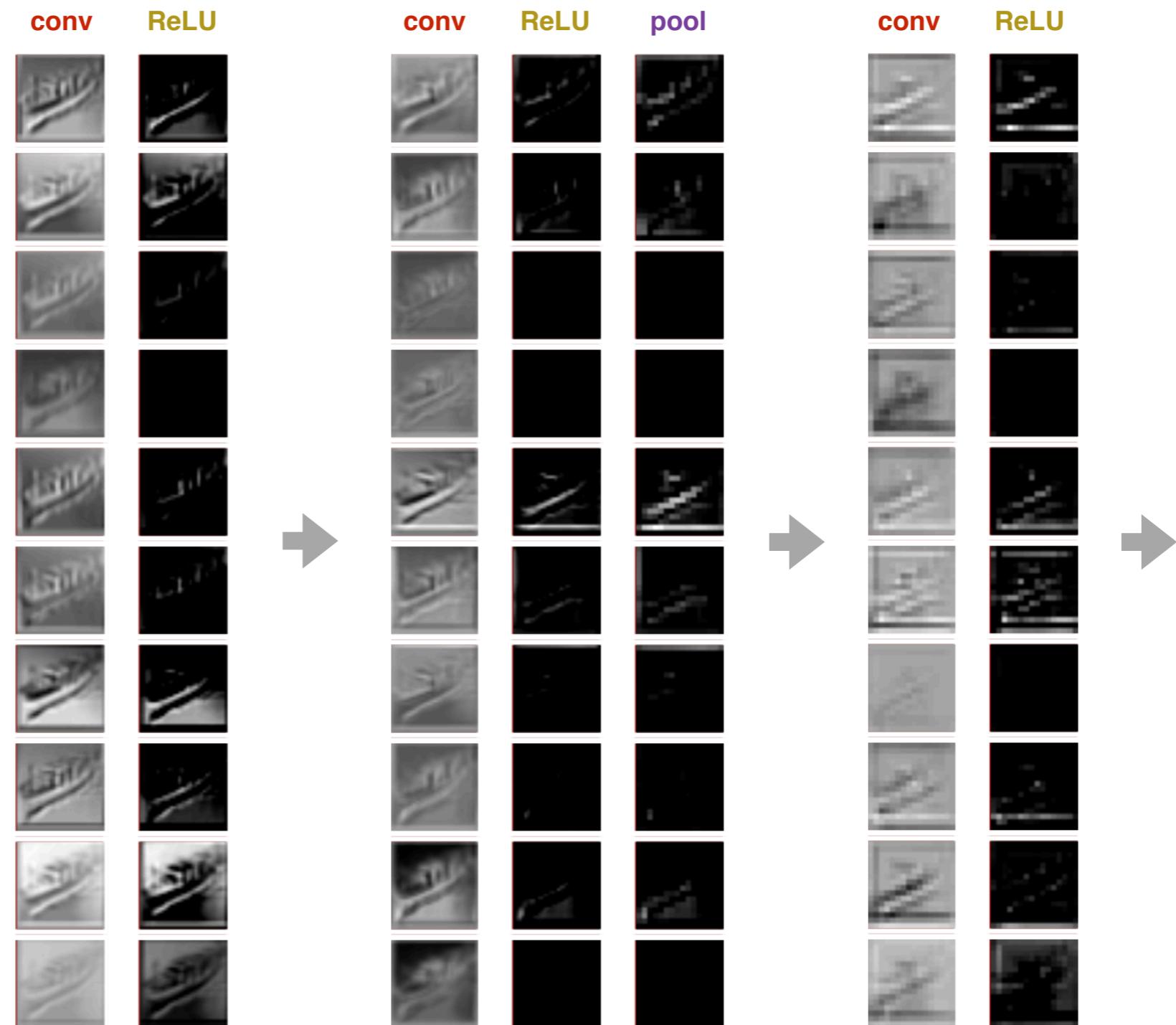
- Series of feature extractors?
- All the way from pixels to classifier, layer by layer?
- Train all the layers together?



Source: <https://www.guru99.com/machine-learning-vs-deep-learning.html>

Deep in action

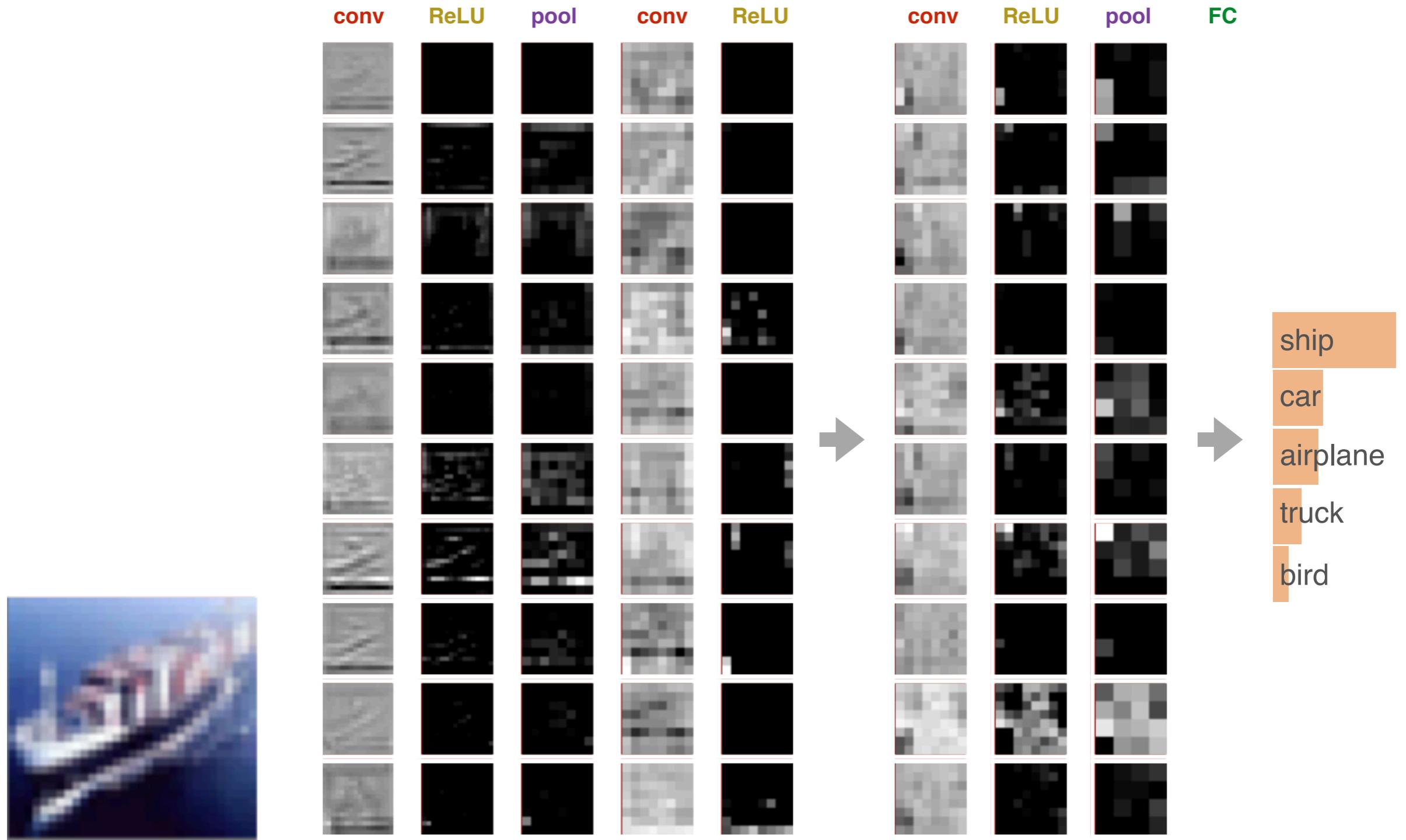
Part 1



Source: <http://cs231n.stanford.edu>

Deep in action

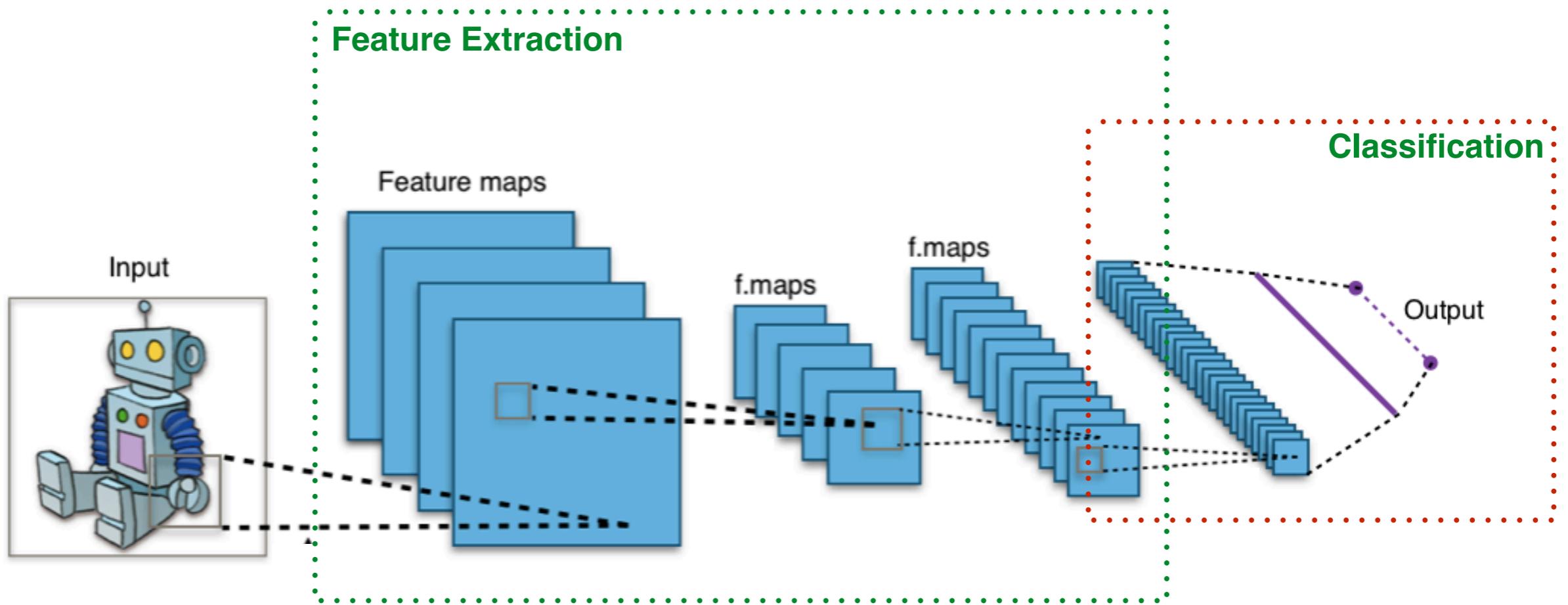
Part 2



Source: <http://cs231n.stanford.edu>

Learning the features

The idea behind convnet



Source: https://commons.wikimedia.org/wiki/File:Typical_cnn.png

From then on ...

ILSVRC 2012

Rank	Error - 5	Algorithm	Team
1	0.153	Deep convolutional neural network	University of Toronto
2	0.262	Features + Fisher vectors + linear classifiers	ISI
3	0.270	Features + Fisher vectors + SVM	Oxford VGG
4	0.271	Not specified	XRCE/INRIA
5	0.300	Dense SIFT + colour SIFT + Fisher vectors + SVM	University of Amsterdam

Source: <http://www.image-net.org/challenges/LSVRC/2012/results.html>

From then on ...

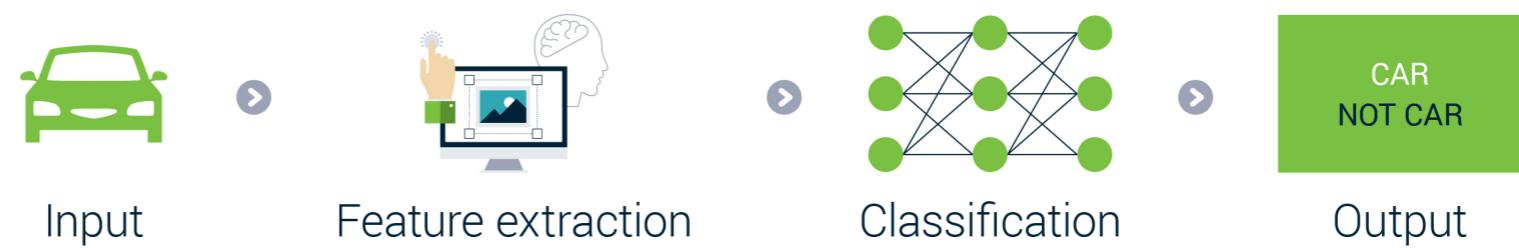
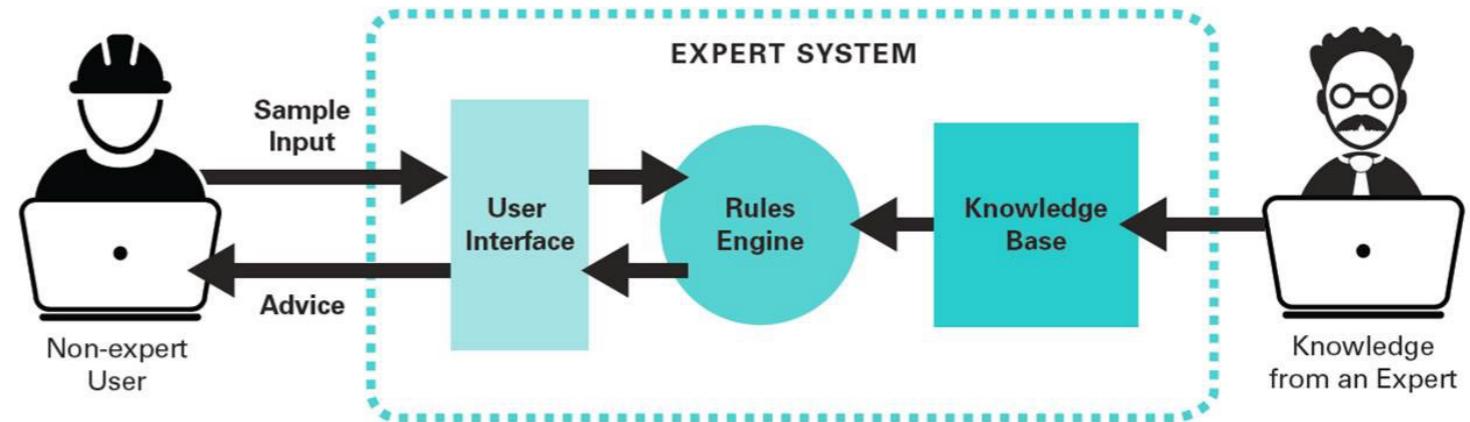
ILSVRC 2013

Rank	Error - 5	Algorithm	Team
1	0.117	Deep convolutional neural network	Clarifai
2	0.129	Deep convolutional neural network	NUS
3	0.135	Deep convolutional neural network	ZF
4	0.136	Deep convolutional neural network	Andrew Howard
5	0.142	Deep convolutional neural network	NYU

Source: <http://www.image-net.org/challenges/LSVRC/2012/results.html>

The progress

So far



Comparison

Machine learning vs deep learning

	Machine learning	Deep learning
Data dependencies	Excellent performances on a small/medium dataset	Excellent performance on a big dataset
Hardware dependencies	Work on a low-end machine	Requires powerful machine
Feature engineering	Need to understand the features that represent the data	No need to understand the learned features
Execution time	From few minutes to hours	Up to weeks
Interpretability	Possible for some (logistic, decision tree); some not possible (SVM, XGBoost)	Difficult to impossible

Source: <https://www.guru99.com/machine-learning-vs-deep-learning.html>

Reminder ...

Not every problem need a deep learning solution ...



Source: <https://twitter.com/MachinePix/status/1102302252142813185>

Components in deep learning

2D convolution

The original

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1																	
2	Input				Kernel				Output								
3																	
4	1	3	2	1					1	2	3						
5	1	3	3	1					0	1	0			23	22		
6	2	1	1	3					2	1	2			31	26		
7	3	2	3	3													
8																	

Source: <https://medium.com/apache-mxnet/convolutions-explained-with-ms-excel-465d6649831c>

2D convolution

The padded

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1																			
2	Input						Kernel						Output						
3																			
4	0	0	0	0	0	0													
5	0	1	3	2	1	0	1	2	3				8	14	13	8			
6	0	1	3	3	1	0	0	1	0				16	23	22	10			
7	0	2	1	1	3	0	2	1	2				20	31	26	17			
8	0	3	2	3	3	0							10	9	15	10			
9	0	0	0	0	0	0													
10																			

Source: <https://medium.com/apache-mxnet/convolutions-explained-with-ms-excel-465d6649831c>

2D convolution

Determine the output size

- M_r, M_c : Output size in rows and columns respectively

- W_r, W_c : Input size in rows and columns respectively

- F_r, F_c : Filter size in rows and columns respectively

- P_r, P_c : Amount of zero-padding in rows and columns respectively

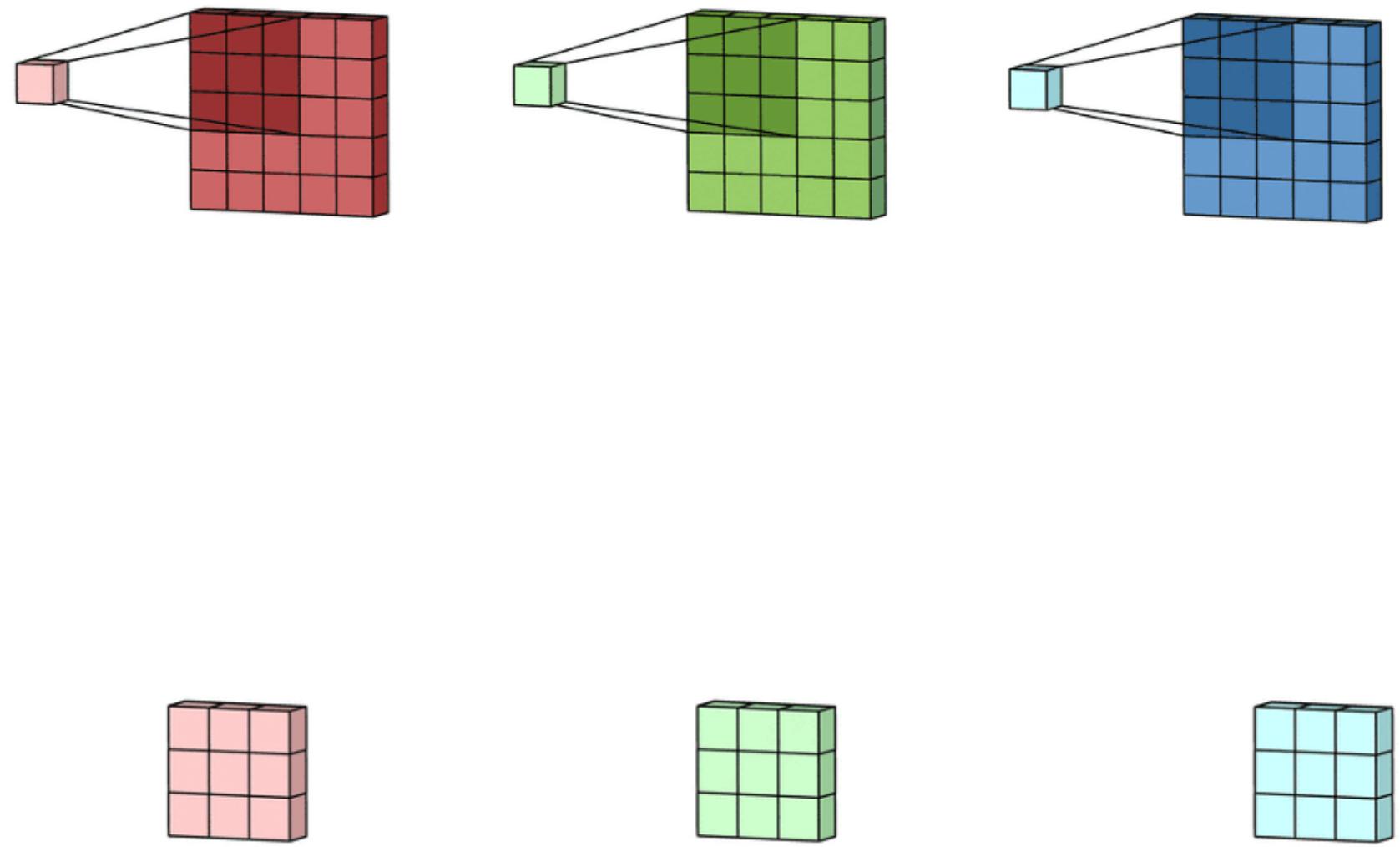
- S_r, S_c : Stride in rows and columns respectively

$$M_r = \left\lfloor \frac{W_r - F_r + 2P_r}{S_r} \right\rfloor + 1$$

$$M_c = \left\lfloor \frac{W_c - F_c + 2P_c}{S_c} \right\rfloor + 1$$

2D convolution

Multi-channel



Source: <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1>

2D convolution

Multi-channel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1																									
2	Input					Kernel					Intermediate Output														
3																									
4	1	0	1	0	2																				
5	1	1	3	2	1																				
6	1	1	0	1	1																				
7	2	3	2	1	3																				
8	0	2	0	1	0																				
9																									
10	1	0	0	1	0																				
11	2	0	1	2	0																				
12	3	1	1	3	0																				
13	0	3	0	3	2																				
14	1	0	3	2	1																				
15																									
16	2	0	1	2	1																				
17	3	3	1	3	2																				
18	2	1	1	1	0																				
19	3	1	3	2	0																				
20	1	1	2	1	1																				
21																									

Source: <https://medium.com/apache-mxnet/convolutions-explained-with-ms-excel-465d6649831c>

Max pooling

The original

Single depth slice

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters
and stride 2



6	8
3	4

Source: <http://cs231n.github.io/convolutional-networks/>

Max pooling

With situation

1	4	4	5	6
3	9	2	3	2
8	1	6	0	7
0	3	2	1	1



9	5
8	6

Source: <https://software.intel.com/en-us/daal-programming-guide-2d-max-pooling-forward-layer>

Maxpooling

Determine the output size

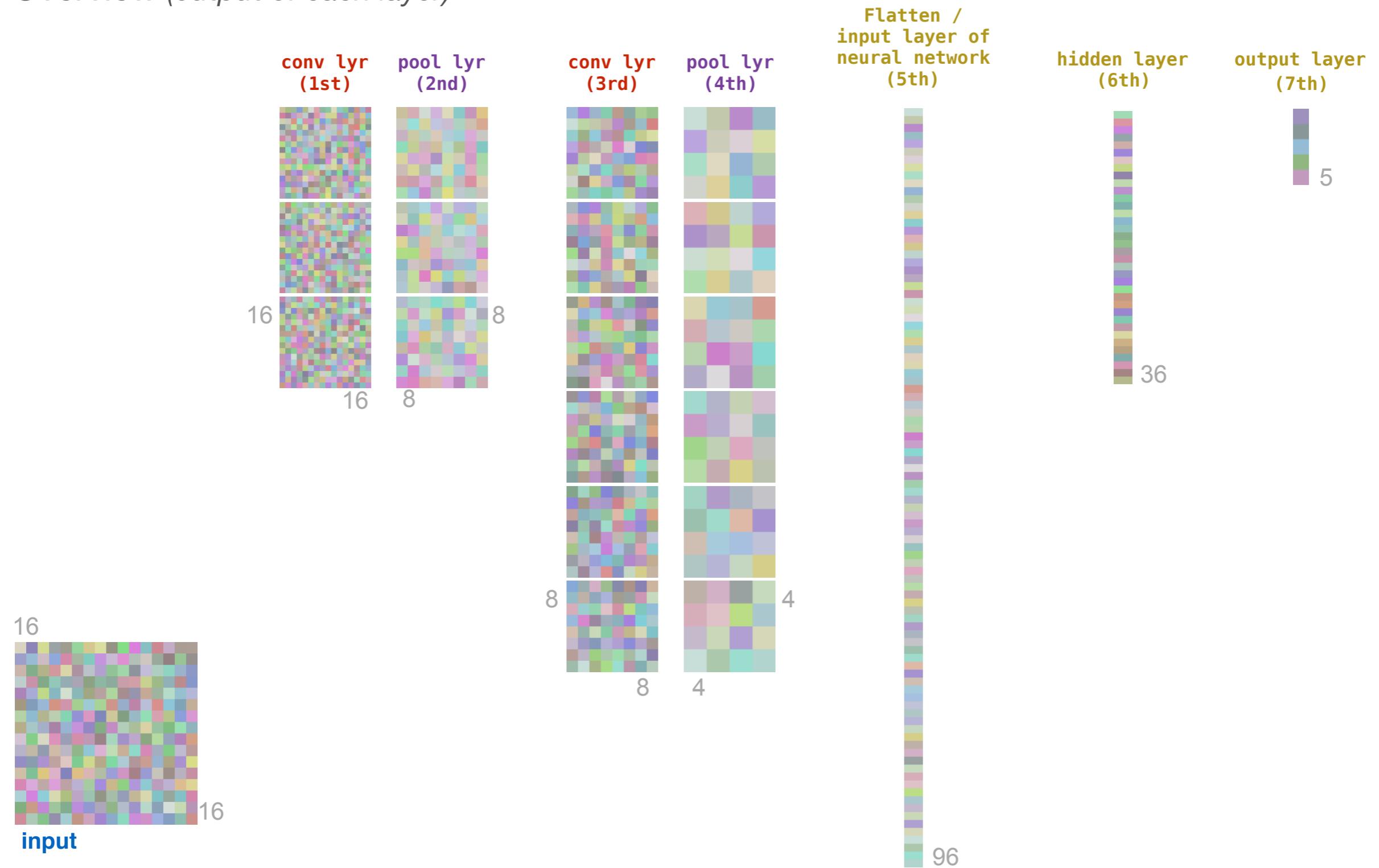
- M_r, M_c : Output size in rows and columns respectively
- W_r, W_c : Input size in rows and columns respectively
- F_r, F_c : Filter size in rows and columns respectively
- S_r, S_c : Stride in rows and columns respectively

$$M_r = \left\lfloor \frac{W_r - F_r}{S_r} \right\rfloor + 1$$

$$M_c = \left\lfloor \frac{W_c - F_c}{S_c} \right\rfloor + 1$$

Convolutional neural network

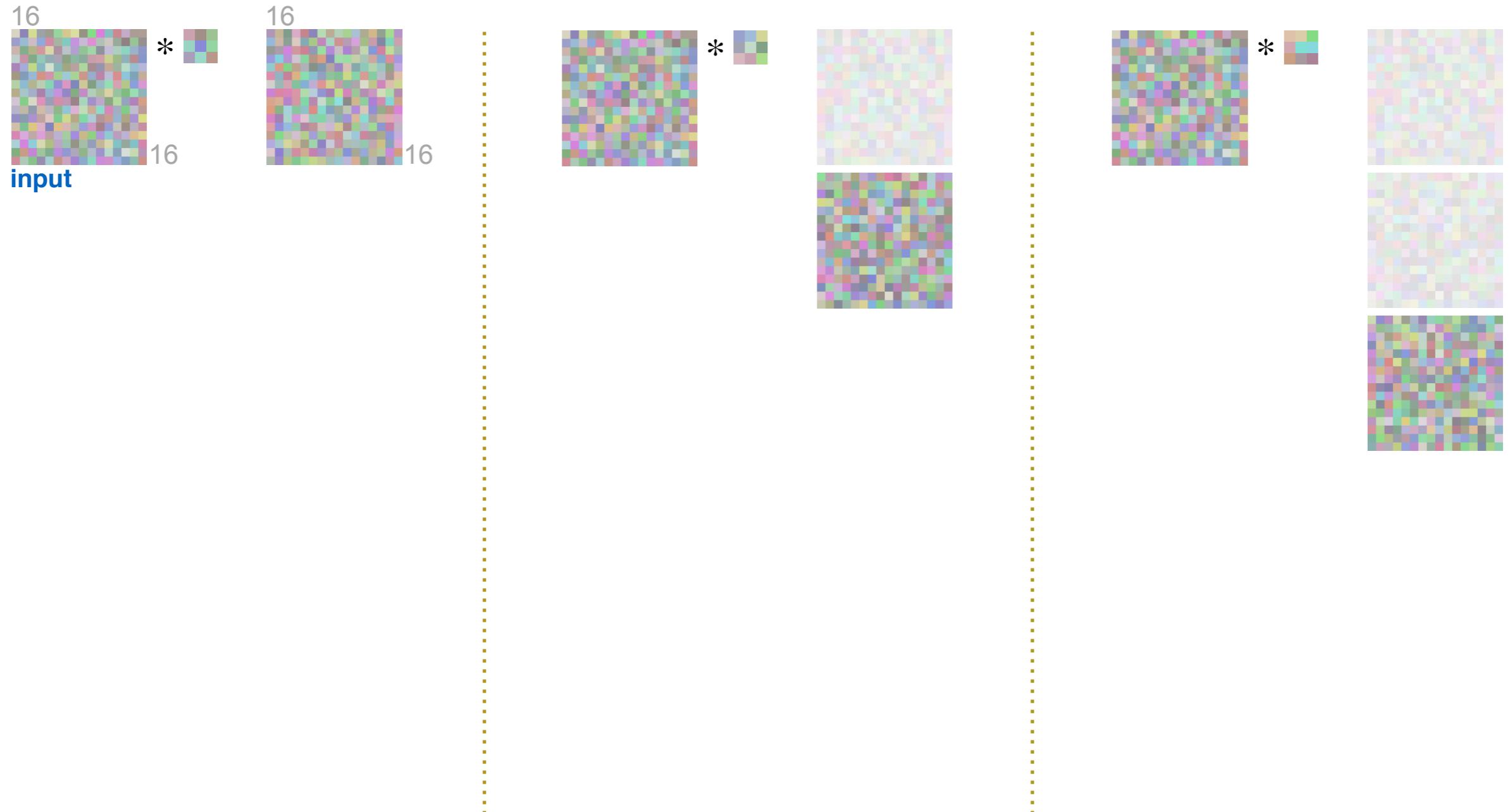
Overview (*output of each layer*)



The making of ...

The first convolutional layer
(part 1)

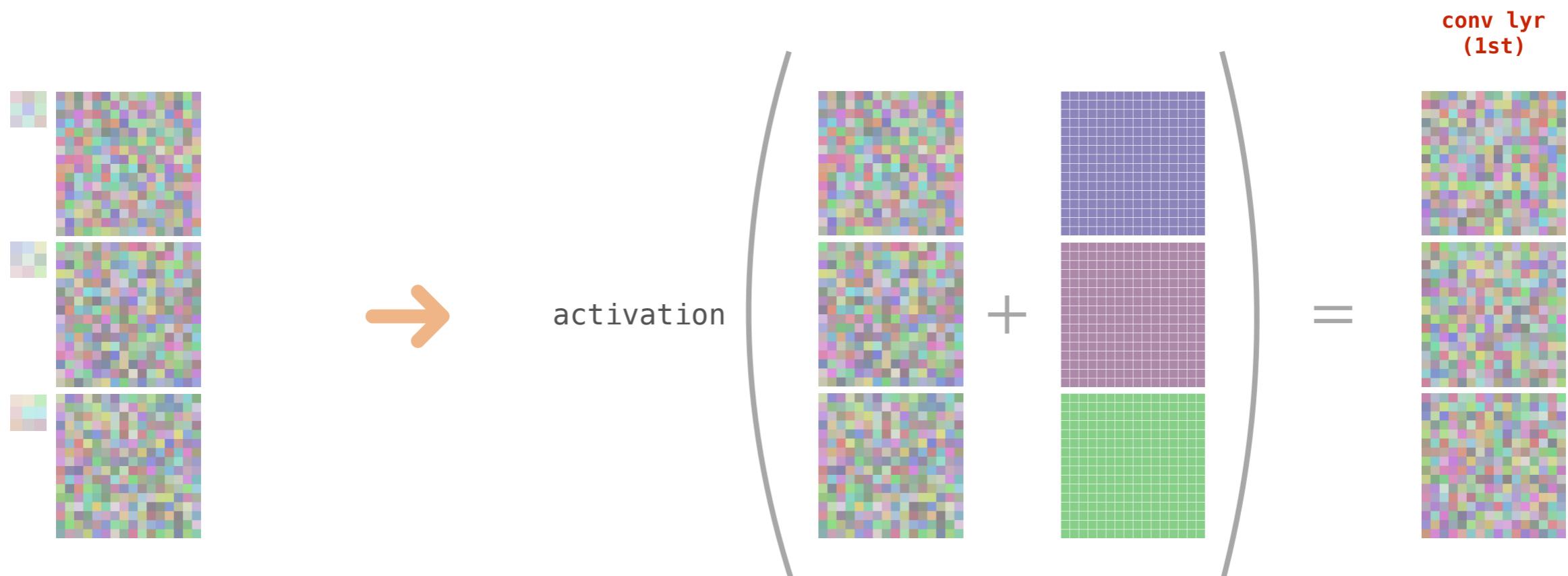
- Performs 3 separate 2D convolutions (with padding) to generate 3 intermediate outputs



The making of ...

The first convolutional layer
(part 2)

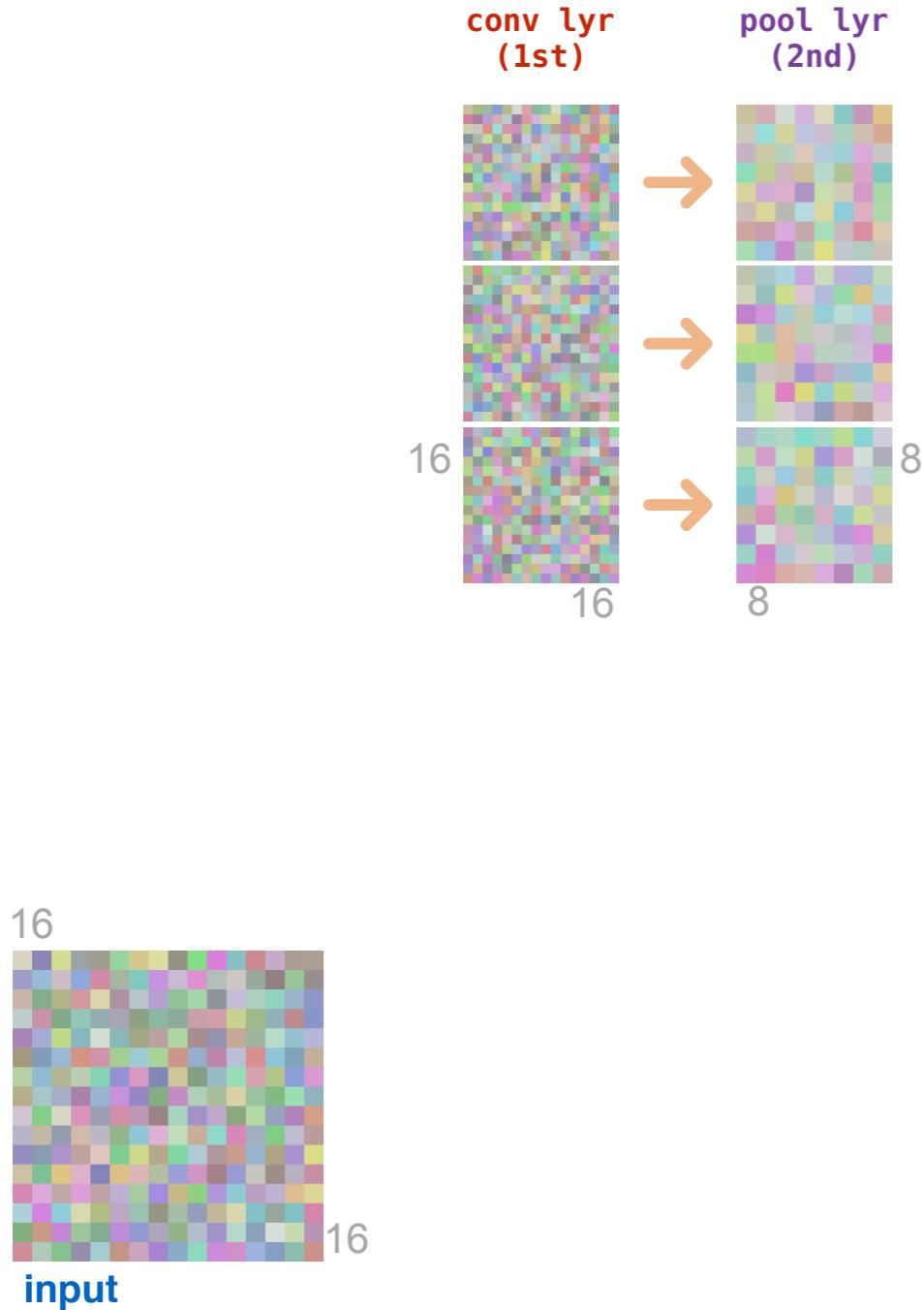
- Add bias to each convolution output, and apply activation function to get the final output for the convolutional layer



The making of

The pooling layer

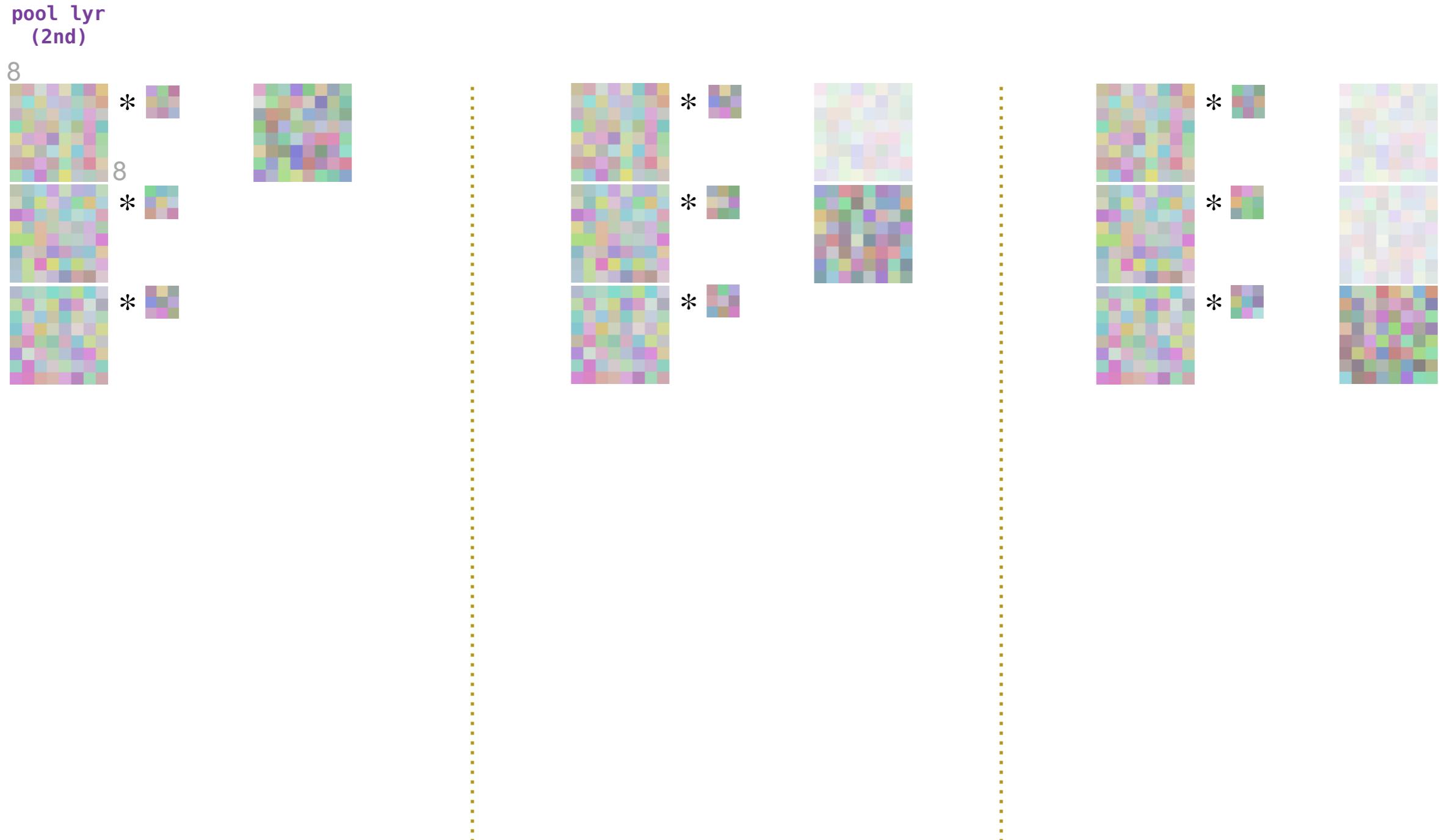
- Apply 2×2 max-pooling (stride 2) on the outputs from the first convolutional layer



The making of ...

The second convolutional layer (part 1)

- Performs 6 separate multi-channel 2D convolutions (with padding) to generate 6 convolution outputs

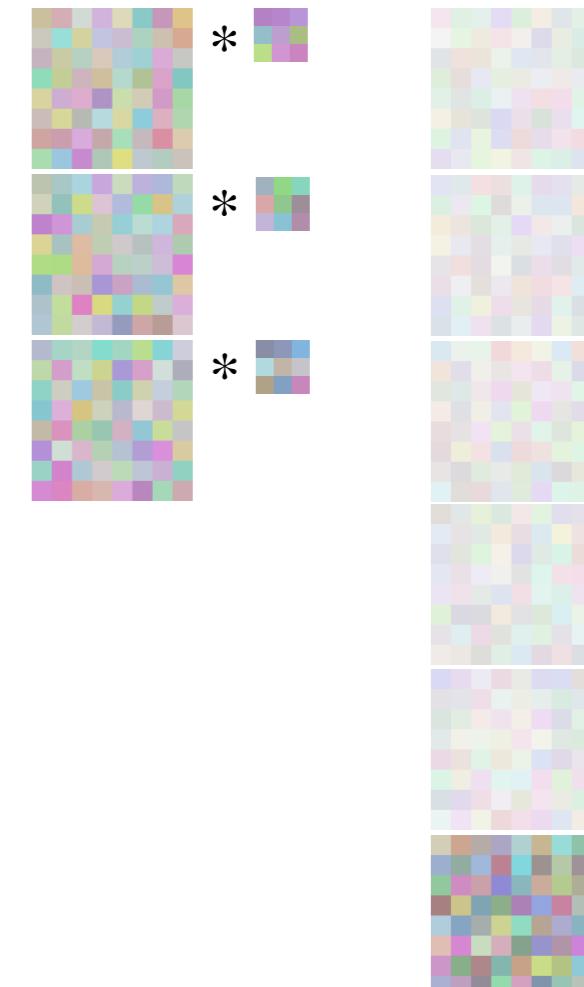
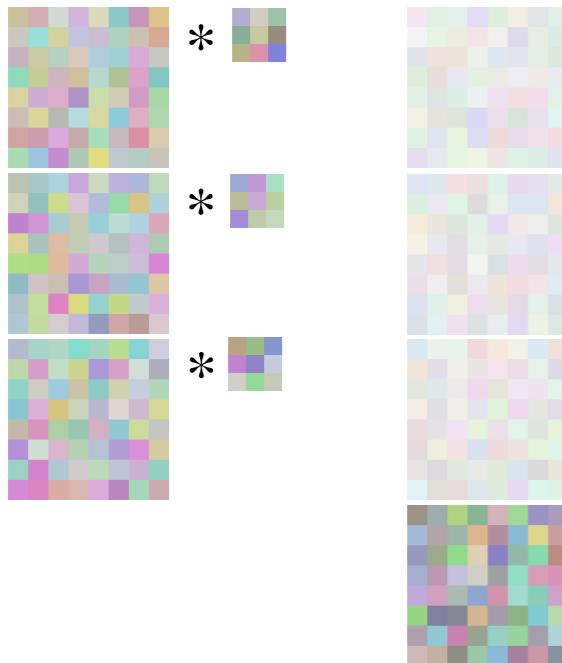


The making of ...

The second convolutional layer (part 2)

- Performs 6 separate multi-channel 2D convolutions (with padding) to generate 6 convolution outputs

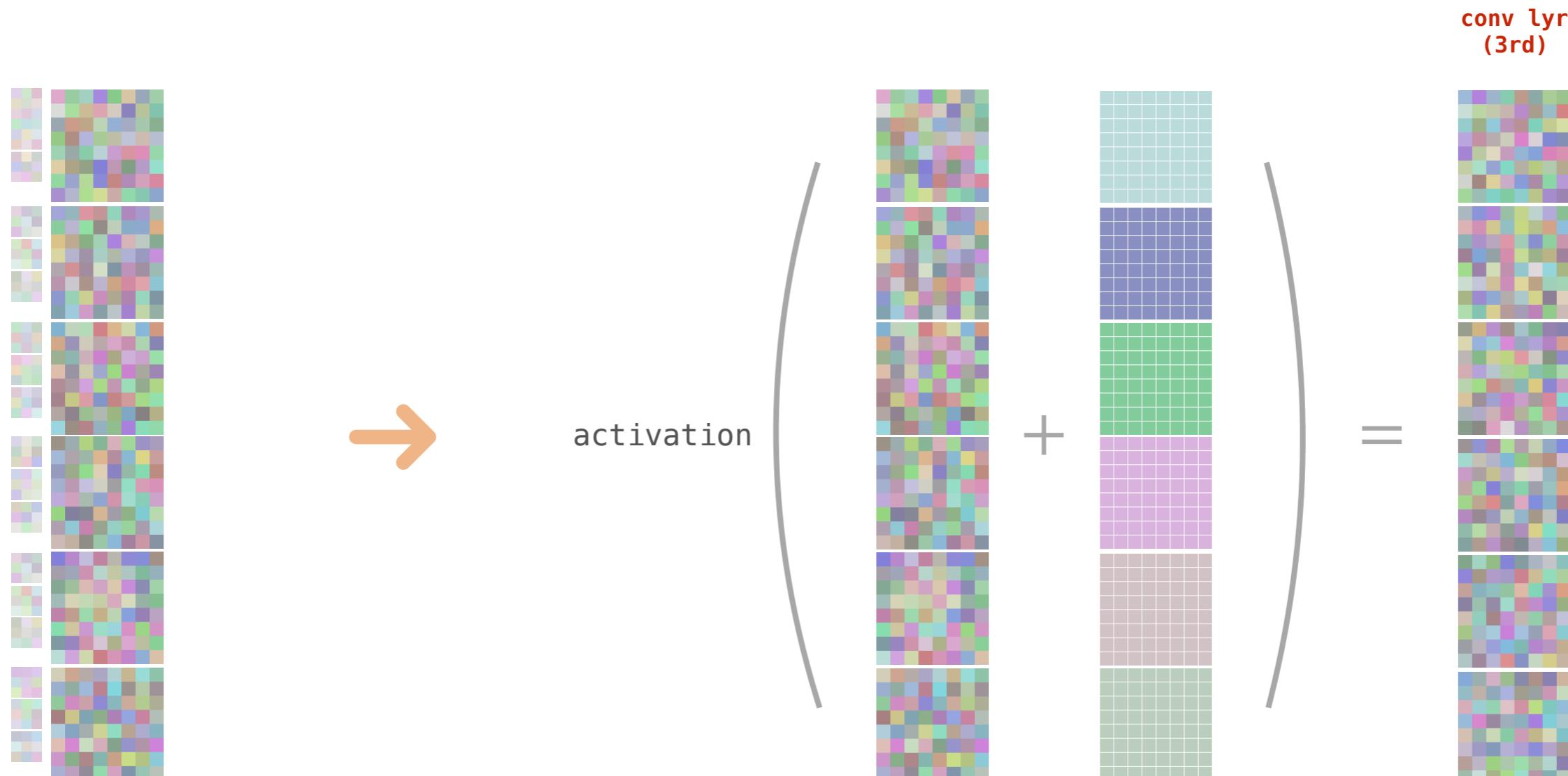
pool lyr
(2nd)



The making of ...

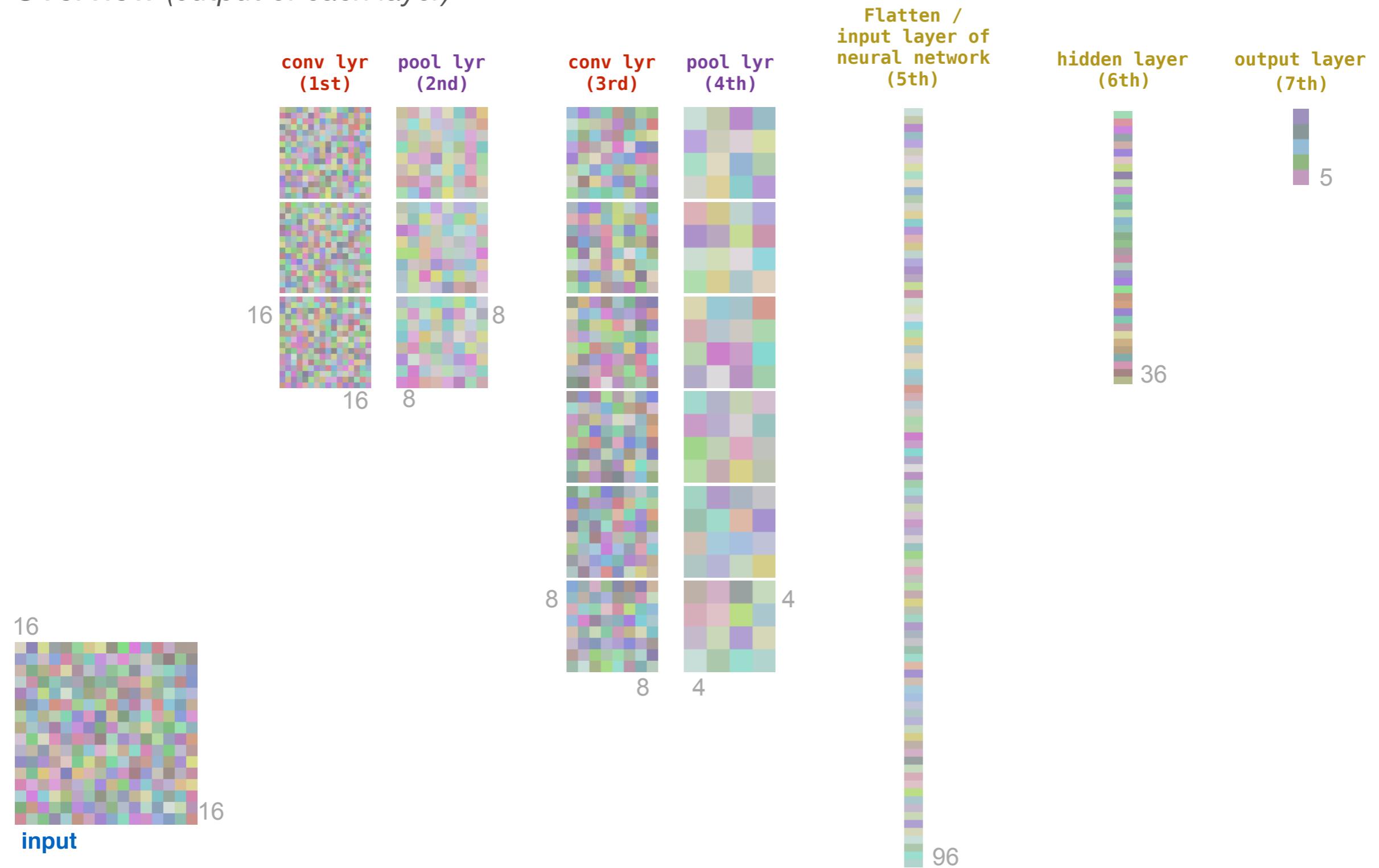
The first convolutional layer
(part 3)

- Add bias to each intermediate output, and apply activation function to get the final output for the convolutional layer



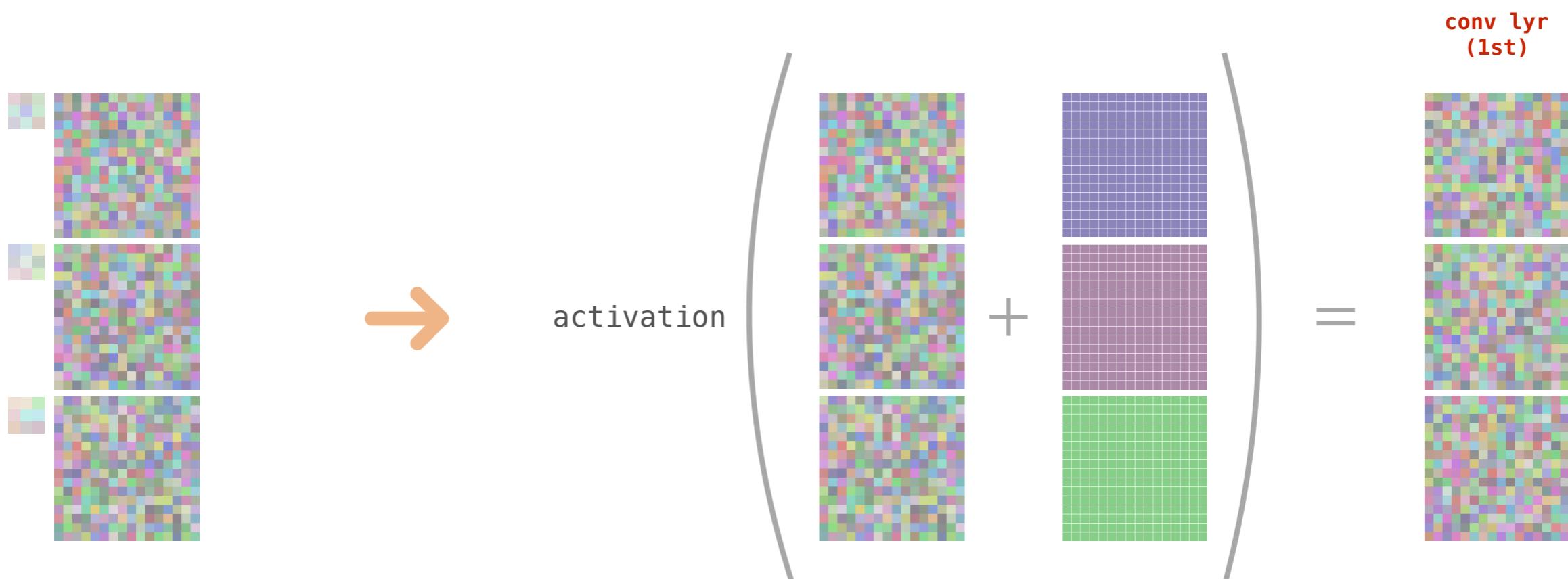
Convolutional neural network

Overview (*output of each layer*)



Calculating parameters

For the first convolutional layer

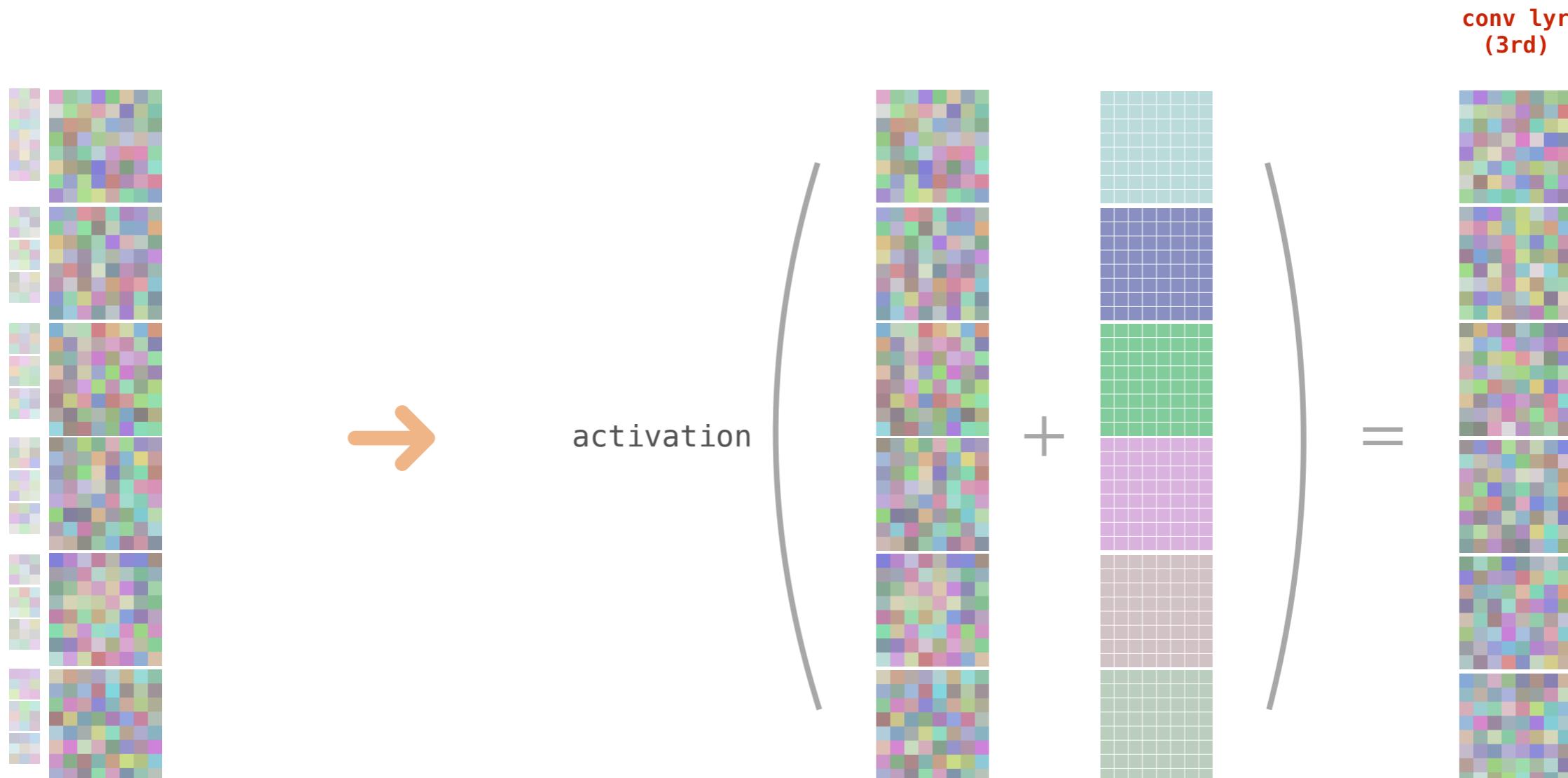


Number of parameters:

$$(3 \times 3) \times 3 + 3 = 30$$

Calculating parameters

For the second convolutional layer



Number of parameters:

$$(3 \times 3) \times 3 \times 6 + 6 = 6 \times [(3 \times 3) \times 3 + 1] = 168$$

Convolutional neural network

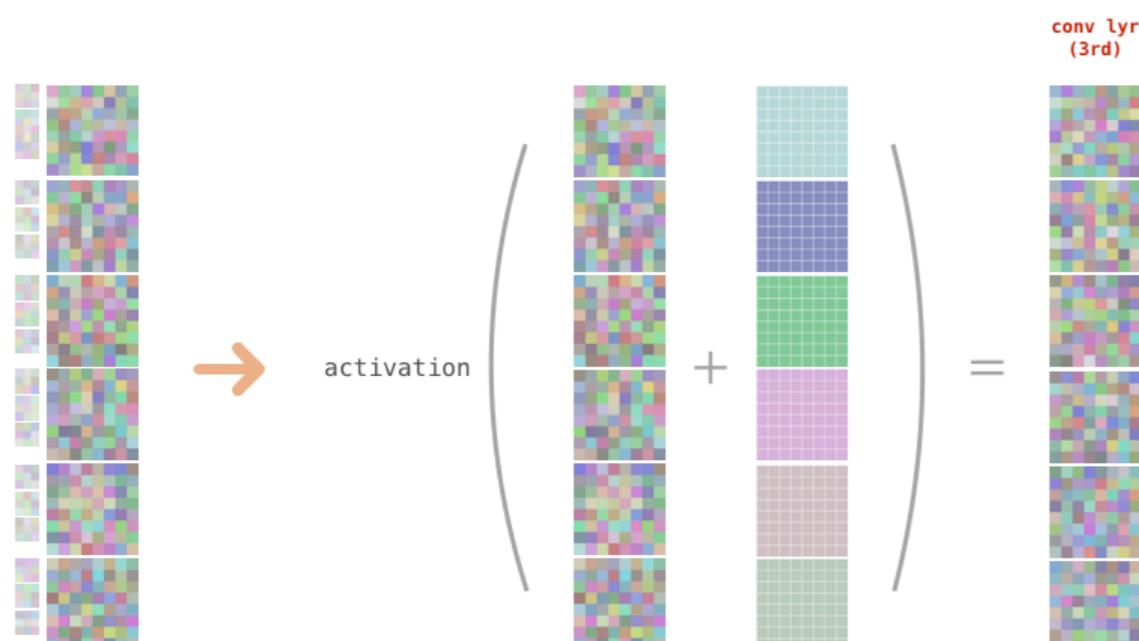
Calculate the parameters

- Assume the size of an input to a layer is (I_r, I_c, C_i) and a filter kernel size of (F_r, F_c)

- Thus to produce **1** feature map involves C_i number of (F_r, F_c) filters. The number of trainable parameters in this case is: $C_i \times (F_r \times F_c) + 1$

- To produce D number of feature maps, it involves $C_i \times D$ number of (F_r, F_c) filters, the number of trainable parameters is

$$[C_i \times (F_r \times F_c) + 1] \times D$$

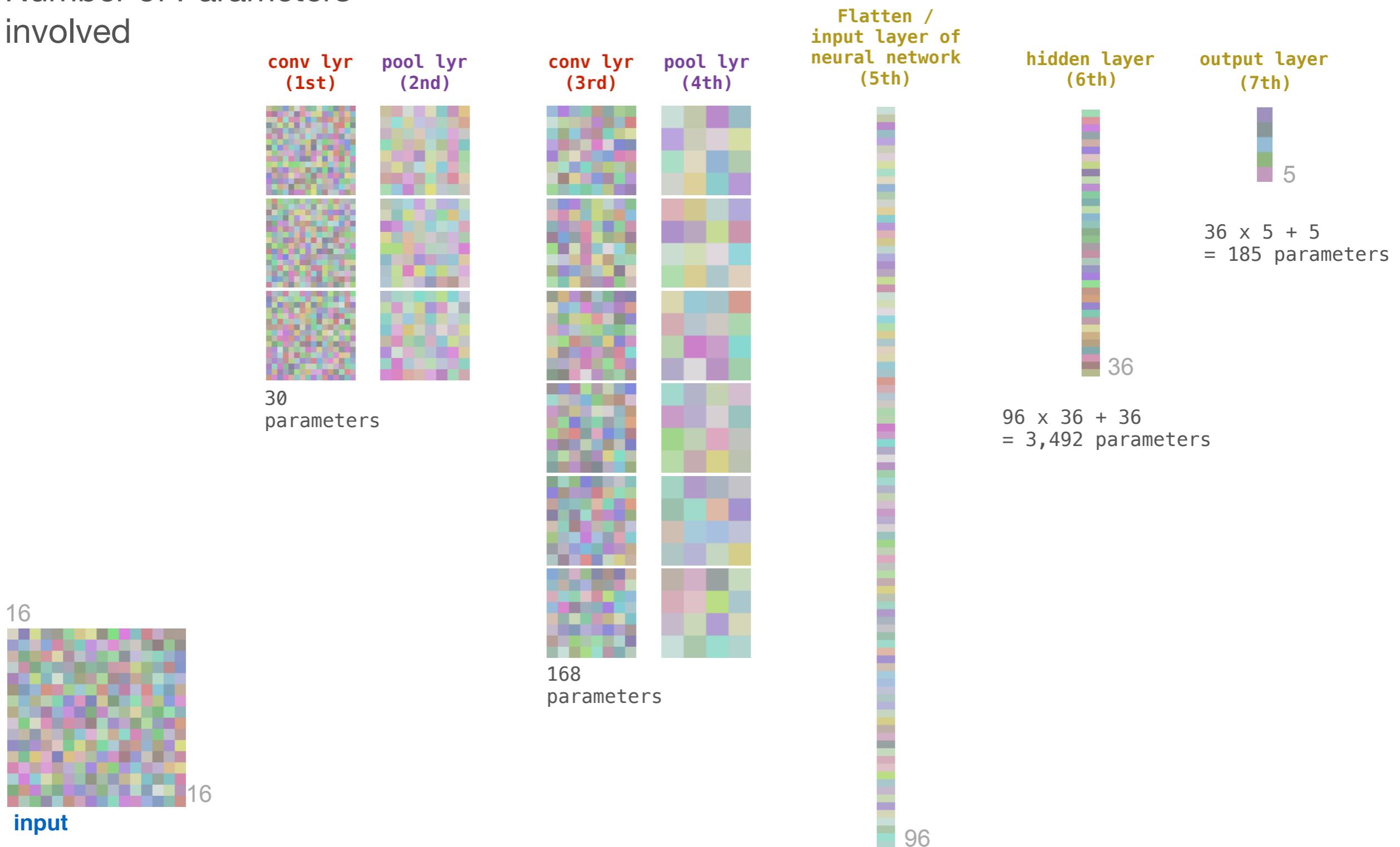


Number of parameters:

$$(3 \times 3) \times 3 \times 6 + 6 = 6 \times [(3 \times 3) \times 3 + 1] = 168$$

Convolutional neural network

Number of Parameters involved



Convolutional neural network

Determine the filter size

- You manage to see the details of a net in the below format

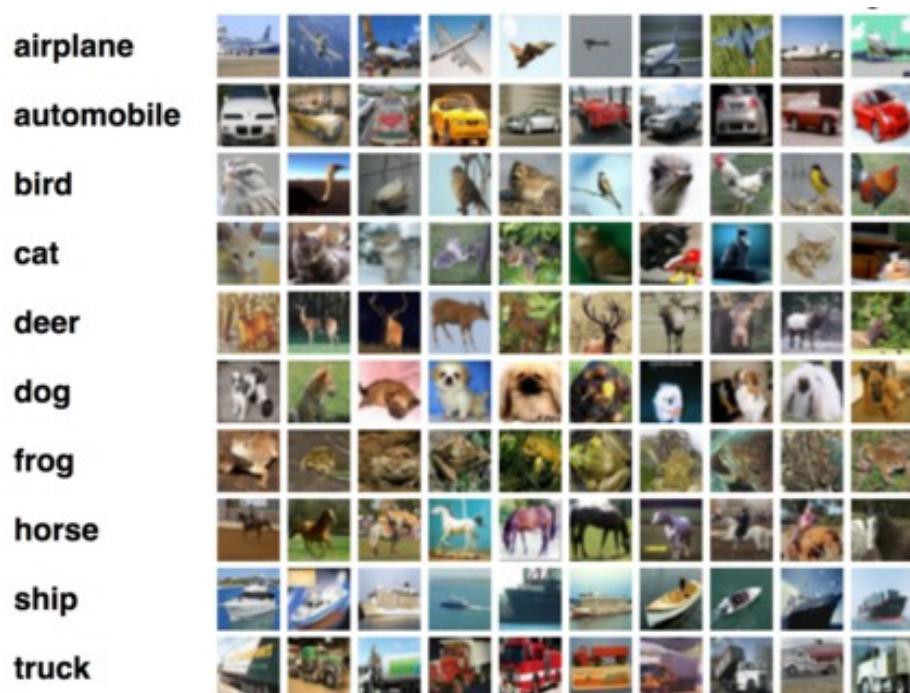
- The number of parameters and the layers' size are given, but not the filter size
- Assume the filter is a square, what is the filter size that produces the feature maps in second layer?

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 32, 32, 3)	0
conv2d (Conv2D)	(None, 32, 32, 32)	896

Time for coding

Before we start ...

Dataset



- Cifar 10: Cifar stands for Canadian Institute For Advanced Research

- 60,000 32 x 32 colour images in 10 distinct classes

Airplane
Automobile
Bird
Cat
Deer
Dog
Frog
Horse
Ship
Trucks

- Each class has 6,000 images

Source: <https://appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks/>

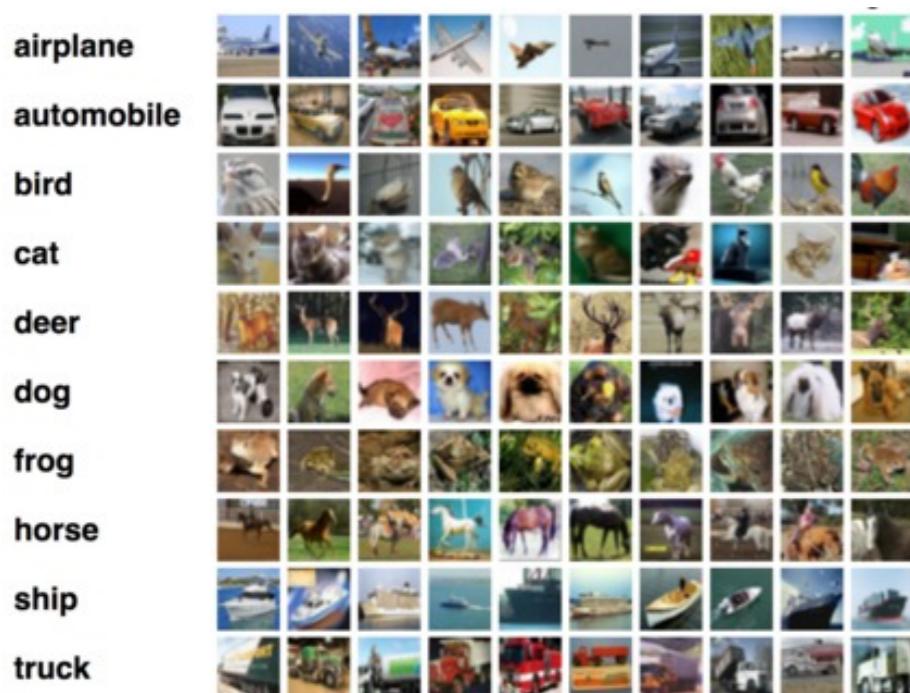
Before we start ...

Dataset

- Dataset collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton

- Advantages: small image size yet large samples, good for quick idea testing

- One of the most widely used datasets for machine learning research



- Not easy to get good and comparable alternative (dataset)

Source: <https://appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks/>

Keras

Or Tensorflow?



- Tensorflow is powerful, but a pain to learn

- Keras is simpler, but not as powerful/flexible as Tensorflow

- However, since Tensorflow r1.13, Keras has become officially the preferred higher level API to build deep learning model

- In Tensorflow 2.0, Keras-way is the default way to build deep learning model in Tensorflow

- Thus in this course, we use Tensorflow but build model in Keras way!

Cifar 10

The main layout for the code

1. Import libraries

2. Matplotlib setup

3. Data preparation

4. Define model

5. Train model

6. Test model

Cifar 10

1. Import libraries, part 1

- numpy for matrix manipulation
- sklearn for measuring performance
- matplotlib to show image and plot result;

```
> import numpy as np  
> import sklearn.metrics as metrics  
> import matplotlib.pyplot as plt
```

Cifar 10

1. Import libraries, part 2

- Import all the Keras functions that we are going to use in this problem
- Most of the key components to build a deep learning model fall under **tensorflow.keras.layers**

```
> from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger  
> from tensorflow.keras.models import Sequential  
> from tensorflow.keras.layers import Dense  
> from tensorflow.keras.layers import Dropout  
> from tensorflow.keras.layers import Flatten  
> from tensorflow.keras.layers import Conv2D  
> from tensorflow.keras.layers import MaxPooling2D  
> from tensorflow.keras.utils import to_categorical  
> from tensorflow.keras.datasets import cifar10  
> from tensorflow.keras import optimizers
```

Cifar 10

2. Matplotlib setup

- Use 'ggplot' style to plot our training and testing result
- The setup uses 'ggplot' style for plot
- Also, for y axis, the labels and ticks put on right rather than left

```
> plt.style.use('ggplot')
> plt.rcParams['ytick.right']      = True
> plt.rcParams['ytick.labelright']= True
> plt.rcParams['ytick.left']       = False
> plt.rcParams['ytick.labelleft']  = False
> plt.rcParams['font.family']     = 'Arial'
```

Cifar 10

3. Data preparation, part 1

- Use Keras in-built cifar10 module to load data

- If `cifar10.load_data()` is never run before, it will download data from the internet

```
> data          = cifar10.load_data()  
> (trDat, trLbl) = data[0]  
> (tsDat, tsLbl) = data[1]
```

```
> trDat        = trDat.astype('float32')/255  
> tsDat        = tsDat.astype('float32')/255
```

```
> imgrows      = trDat.shape[1]  
> imgclms      = trDat.shape[2]  
> channel      = trDat.shape[3]
```

Cifar 10

3. Data preparation, part 1

- The shape of trDat is (50000, 32, 32, 3)
- The shape of tsDat is (10000, 32, 32, 3)
- For deep learning training and testing, the data must be in the form of (sample, row, clm, channel)

Name	Type	Size	Value
channel	int	1	3
data	tuple	2	((Numpy array, Numpy array), (Numpy array, Numpy array))
imgclms	int	1	32
imgrows	int	1	32
trDat	float32	(50000, 32, 32, 3)	[[[[0.23137255 0.24313726 0.24705882] [0.16862746 0.18039216 0.1764 ...
trLbl	uint8	(50000, 1)	[[6] [9]
tsDat	float32	(10000, 32, 32, 3)	[[[[0.61960787 0.4392157 0.19215687] [0.62352943 0.43529412 0.1843 ...
tsLbl	int64	(10000, 1)	[[3] [8]

Cifar 10

3. Data preparation, part 1

- (sample, row, clm, channel) is a 'channel last' channel ordering format
- Some frameworks prefer 'channel first' format, which is (sample, channel, row, clm)
- Why? Also, why put sample in the first dimension?

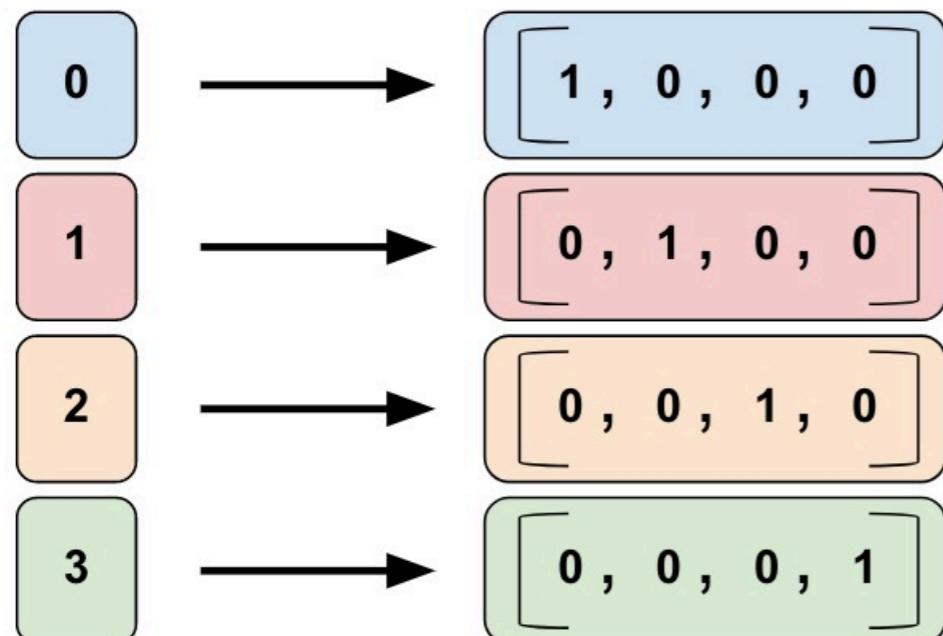
Cifar 10

3. Data preparation, part 2

- One-hot encode the train and test label information;

- `to_categorical` is imported from `tensorflow.keras.utils` in the beginning

```
> trLbl      = to_categorical(trLbl)
> tsLbl      = to_categorical(tsLbl)
> num_classes = tsLbl.shape[1]
```



Source: <https://arxiv.org/pdf/1812.01718.pdf>

Cifar 10

3. Data preparation, part 2

- One-hot encoding

Before

	0
0	6
1	9
2	9
3	4
4	1
5	1
6	2
7	7

After

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	1	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0

Cifar 10

4. Define model, part 1

- Set up random seed
- Set up optimizer. Use RMSprop, lr stands for learning rate
- Give this model a name for later model saving and files saving usage

```
> seed      = 29
> np.random.seed(seed)

> optmz     = optimizers.RMSprop(lr=0.0001)
> modelname = 'wks2_3a'
```

$$E_{dw} = 0 \quad \text{Initialization}$$

$$E_{dw} = \beta \cdot E_{dw} + (1 - \beta) \cdot dw^2$$

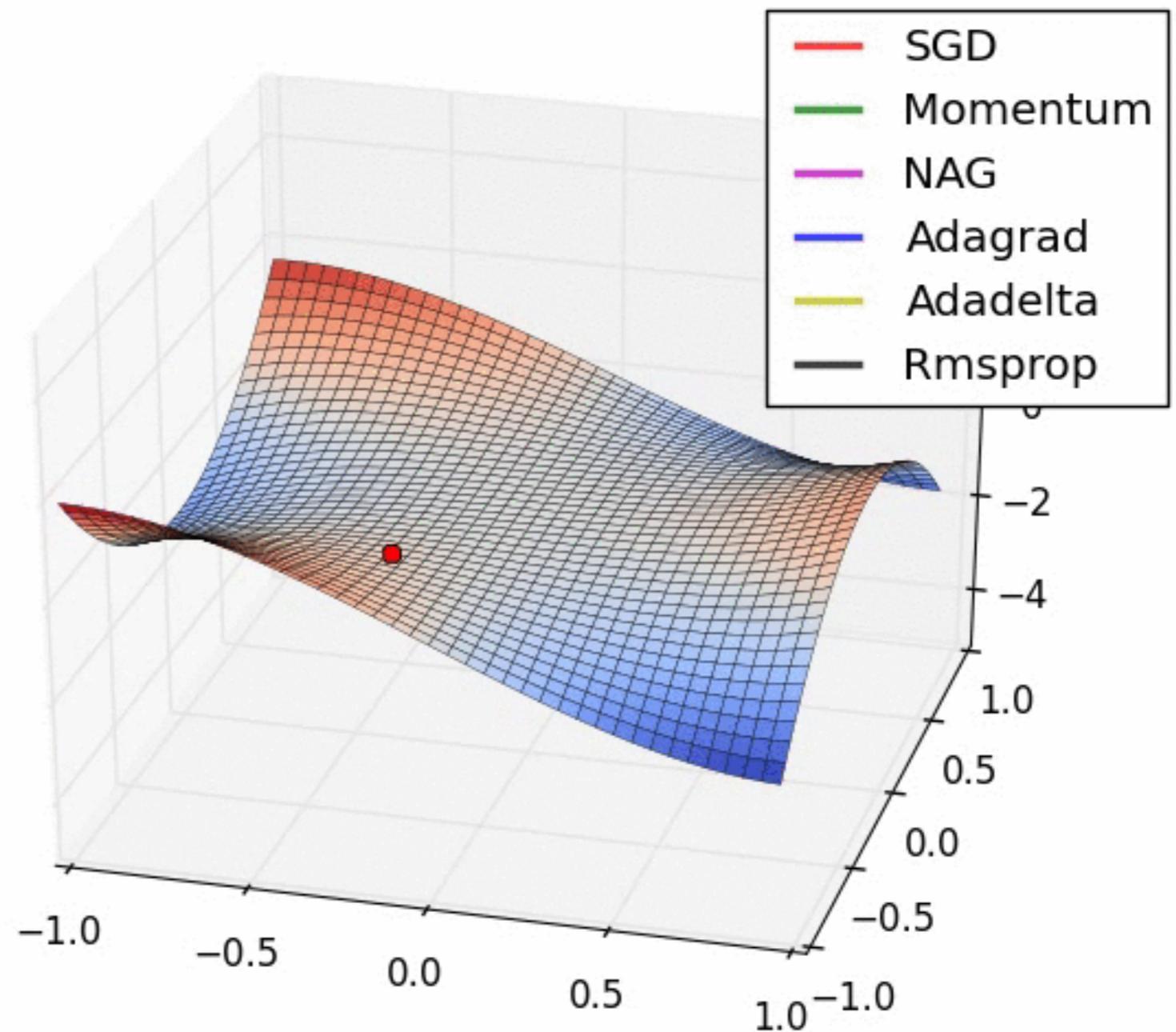
$$w = w - \alpha \cdot \frac{dw}{\sqrt{E_{dw}} + \epsilon}$$

$$b = b - \alpha \cdot \frac{db}{\sqrt{E_{db}} + \epsilon}$$

Source: <https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a>

Cifar 10

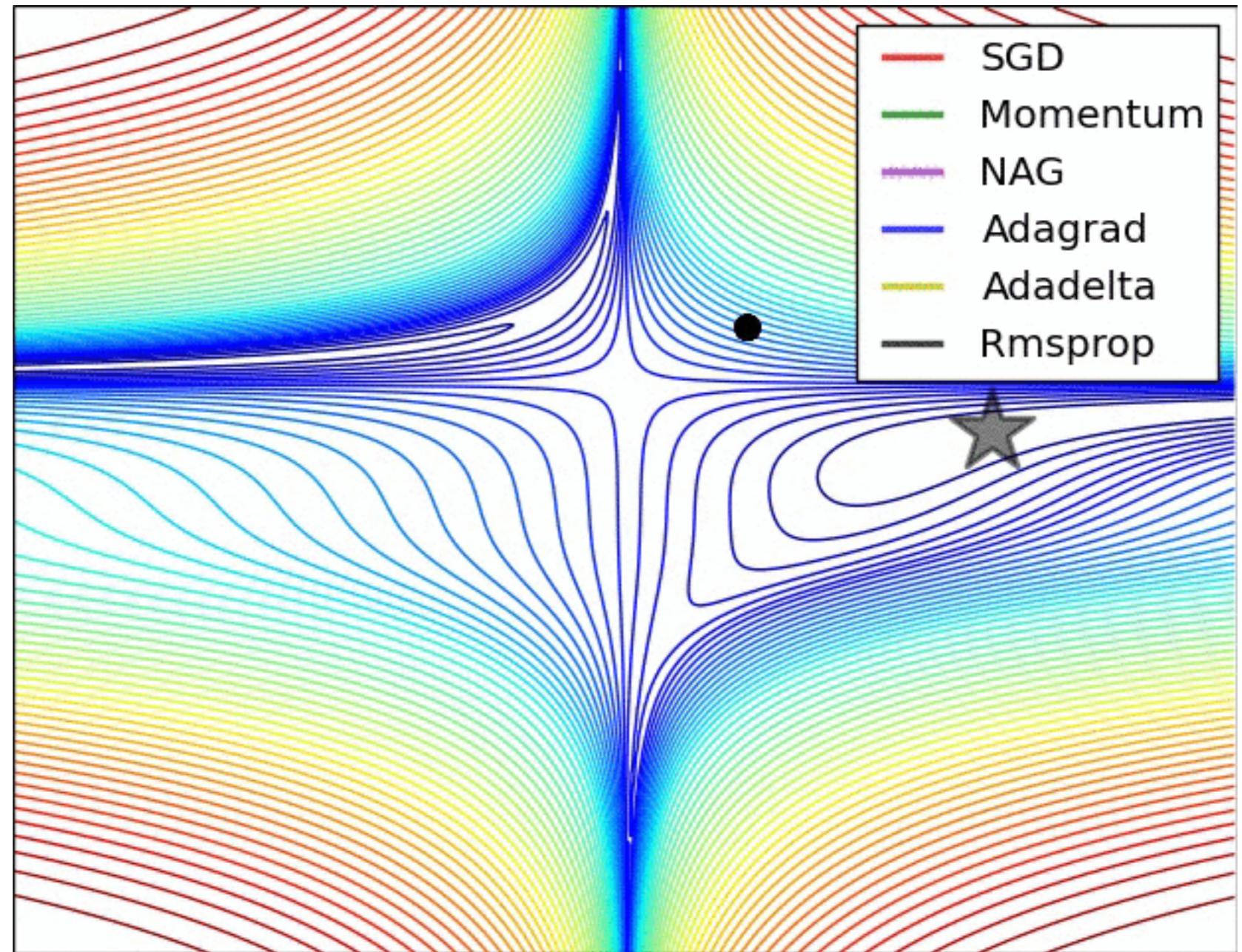
RMSprop



Source: <https://imgur.com/a/Hqolp#NKsFHJb>

Cifar 10

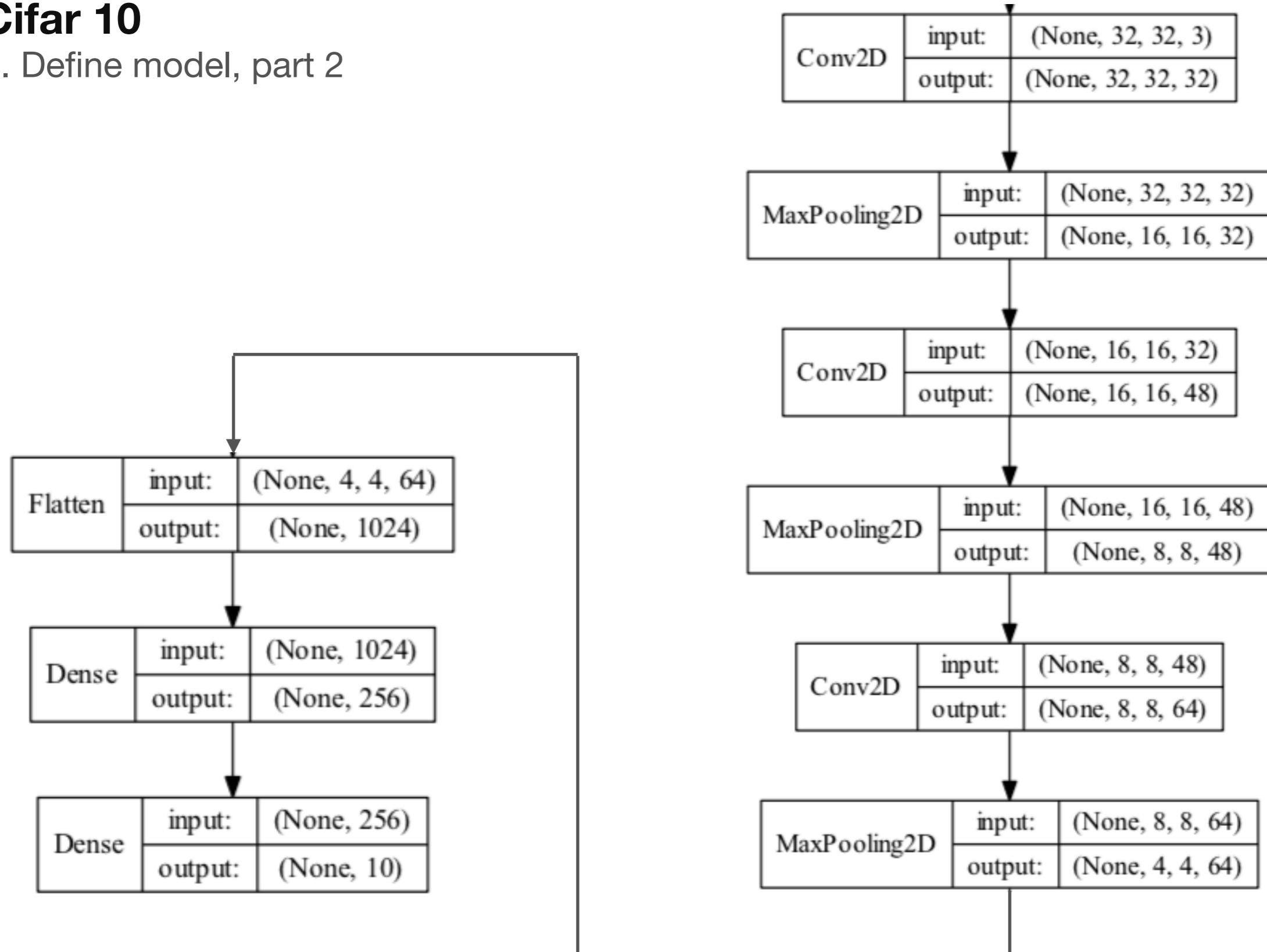
RMSprop



Source: <https://imgur.com/a/Hqolp#NKsFHJb>

Cifar 10

4. Define model, part 2



Cifar 10

4. Define model, part 2

```
> def createModel():
    model = Sequential()
    model.add(Conv2D(32,(3,3),
                    input_shape=(imgrows,imgclms,channel),
                    padding='same',
                    activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(48,(3,3),padding='same',activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(64,(3,3),padding='same',activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dense(256,activation='relu'))
    model.add(Dense(num_classes,activation='softmax'))

    model.compile(loss='categorical_crossentropy',
                  optimizer=optmz,
                  metrics=['accuracy'])
    return model
```

Cifar 10

4. Define model, part 2

- 'model' for training; 'modelGo' for final evaluation

```
> model      = createModel()  
> modelGo    = createModel()  
  
> model.summary()
```

Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 32, 32, 32)	896
<hr/>		
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
<hr/>		
conv2d_1 (Conv2D)	(None, 16, 16, 48)	13872
<hr/>		
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 48)	0
<hr/>		
conv2d_2 (Conv2D)	(None, 8, 8, 64)	27712
<hr/>		
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 64)	0
<hr/>		
flatten (Flatten)	(None, 1024)	0
<hr/>		
dense (Dense)	(None, 256)	262400
<hr/>		
dense_1 (Dense)	(None, 10)	2570
<hr/>		
Total params: 307,450		
Trainable params: 307,450		
Non-trainable params: 0		
<hr/>		

Cifar 10

4. Define model, part 3

- Create checkpoints to save model during training and save training data into csv
- ‘monitor’ can be ‘val_acc’ or ‘val_loss’
- When set to ‘val_acc’, ‘mode’ must be ‘max’; when set to ‘val_loss’, ‘mode’ must be ‘min’

```
> filepath      = modelname + ".hdf5"
> checkpoint    = ModelCheckpoint(filepath,
                                    monitor='val_acc',
                                    verbose=0,
                                    save_best_only=True,
                                    mode='max')

> csv_logger    = CSVLogger(modelname + '.csv')
> callbacks_list = [checkpoint,csv_logger]
```

Cifar 10

5. Train model

- Training is only a single line

```
> model.fit(trDat,  
           trLbl,  
           validation_data=(tsDat, tsLbl),  
           epochs=100,  
           batch_size=128,  
           shuffle=True,  
           callbacks=callbacks_list)
```

```
Train on 50000 samples, validate on 10000 samples  
Epoch 1/100  
50000/50000 [=====] - 4s 87us/sample - loss: 1.9316 - acc: 0.3231 - val_loss: 1.7215 - val_acc: 0.3937  
Epoch 2/100  
50000/50000 [=====] - 3s 61us/sample - loss: 1.6368 - acc: 0.4176 - val_loss: 1.5545 - val_acc: 0.4454  
Epoch 3/100  
50000/50000 [=====] - 3s 61us/sample - loss: 1.5251 - acc: 0.4542 - val_loss: 1.4865 - val_acc: 0.4624  
Epoch 4/100  
50000/50000 [=====] - 3s 61us/sample - loss: 1.4538 - acc: 0.4819 - val_loss: 1.4130 - val_acc: 0.4941  
Epoch 5/100  
50000/50000 [=====] - 3s 61us/sample - loss: 1.3988 - acc: 0.5034 - val_loss: 1.3744 - val_acc: 0.5135  
.....
```

Cifar 10

6. Test model, part 1

- Use a new object to load the weights and re-compile again

```
> modelGo.load_weights(filepath)
> modelGo.compile(loss='categorical_crossentropy',
                    optimizer=optmz,
                    metrics=['accuracy'])
```

Cifar 10

6. Test model, part 2

- Test the model, calculate the accuracy and confusion matrix

```
> predicts      = modelGo.predict(tsDat)  
  
> predout      = np.argmax(predicts, axis=1)  
> testout       = np.argmax(tsLbl, axis=1)  
> labelname     = ['airplane',  
                  'automobile',  
                  'bird',  
                  'cat',  
                  'deer',  
                  'dog',  
                  'frog',  
                  'horse',  
                  'ship',  
                  'truck']  
  
> testScores    = metrics.accuracy_score(testout, predout)  
> confusion     = metrics.confusion_matrix(testout, predout)
```

cifar 10

6. Test model, part 3

- Test the model, calculate the accuracy and confusion matrix

```
> print("Best accuracy (on testing dataset): %.2f%%" % (testScores*100))
> print(metrics.classification_report(testout,predout,target_names=labelname,digits=4))
> print(confusion)
```

	precision	recall	f1-score	support	[[738 18 36 18 40 4 10 11 52 73]	[7 771 7 10 1 2 11 2 18 171]	[60 6 595 75 108 44 51 35 9 17]	[17 11 49 610 81 103 53 33 11 32]	[11 2 49 43 764 23 37 56 7 8]	[13 2 46 224 63 562 26 48 4 12]	[7 4 32 79 46 24 788 5 3 12]	[7 7 22 43 61 31 10 793 2 24]	[44 34 13 24 11 6 8 3 799 58]	[17 36 6 11 2 3 5 12 12 896]]
airplane	0.8013	0.7380	0.7683	1000										
automobile	0.8653	0.7710	0.8154	1000										
bird	0.6959	0.5950	0.6415	1000										
cat	0.5365	0.6100	0.5709	1000										
deer	0.6491	0.7640	0.7019	1000										
dog	0.7007	0.5620	0.6238	1000										
frog	0.7888	0.7880	0.7884	1000										
horse	0.7946	0.7930	0.7938	1000										
ship	0.8713	0.7990	0.8336	1000										
truck	0.6876	0.8960	0.7781	1000										
avg / total	0.7391	0.7316	0.7316	10000										

Cifar 10

6. Test model, part 4

- Plot the result

```
> import pandas as pd  
  
> records      = pd.read_csv(modelname + '.csv')  
> plt.figure()  
> plt.subplot(211)  
> plt.plot(records['val_loss'])  
> plt.yticks([0.00,0.60,0.70,0.80])  
> plt.title('Loss value', fontsize=12)  
  
> ax          = plt.gca()  
> ax.set_xticklabels([])  
  
> plt.subplot(212)  
> plt.plot(records['val_acc'])  
> plt.yticks([0.5,0.6,0.7,0.8])  
> plt.title('Accuracy', fontsize=12)  
> plt.show()
```

