

# Specialist Programme on Artificial Intelligence for IT & ITES Industry

*Convolutional neural networks, Part 1*

By Dr. Tan Jen Hong

[tanjehong@nus.edu.sg](mailto:tanjehong@nus.edu.sg)

Singapore e-Government Leadership Centre  
National University of Singapore

© 2020 NUS. The contents contained in this document may not be reproduced in any form or by any means,  
without the written permission of ISS, NUS other than for the purpose for which it has been supplied.

*Inspire*      *Lead*      *Transform*

# TAN Jen Hong



## Profile

Jen Hong develops algorithms. He specializes in deep learning, image processing and medical image diagnosis. He designs illustrations, web page and posters. He plays piano.

He invented a mathematical model to analyze dry eye. He used deep learning to correct medical images. He trained deep learning models to identify pathologies in retinal images. And he made deep learning to draw anatomical features.

### Lecturer & Consultant, Analytics & Intelligent Systems Practice

 [isstjh@nus.edu.sg](mailto:isstjh@nus.edu.sg)

He was the co-Principal Investigator of 6 research grants and 3 clinical trials. He and his team member co-developed algorithms to diagnose breast cancer, ovarian cancer, heart attack, fatty liver, diabetic retinopathy, epilepsy and glaucoma. He has published more than 90 journal articles, 12 of which are deep learning related.

Worldwide his publications are cited more than 2000 times.

### Educational Qualifications

- Ph.D. (Biomedical Engineering), Nanyang Technological University
- Bachelor of Engineering (Mechanical & Production Engineering), Minor in Chinese, Nanyang Technological University

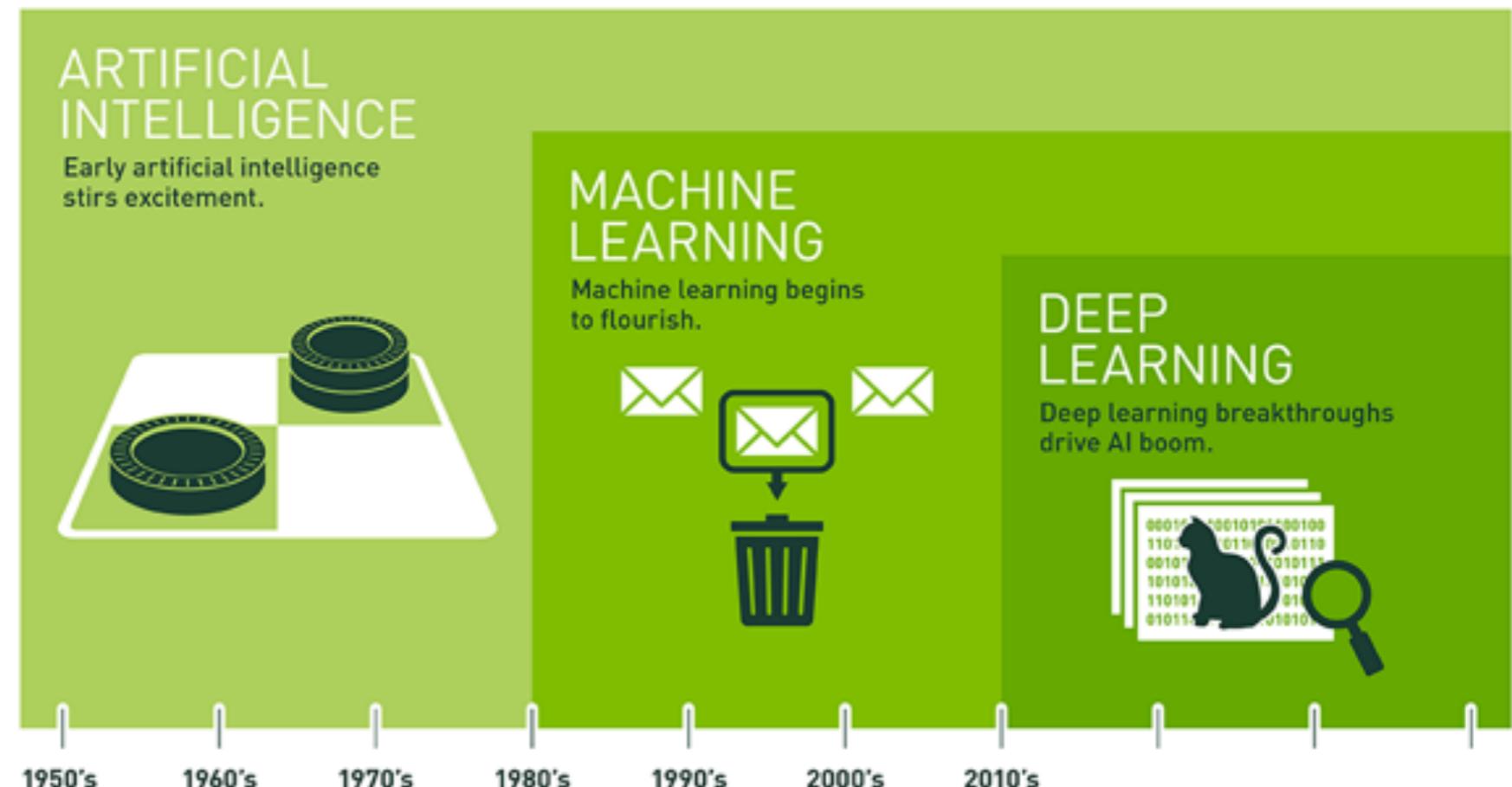
### Selected Publications



# **The rise of machine-learned features**

# The AI time line

A broad overview

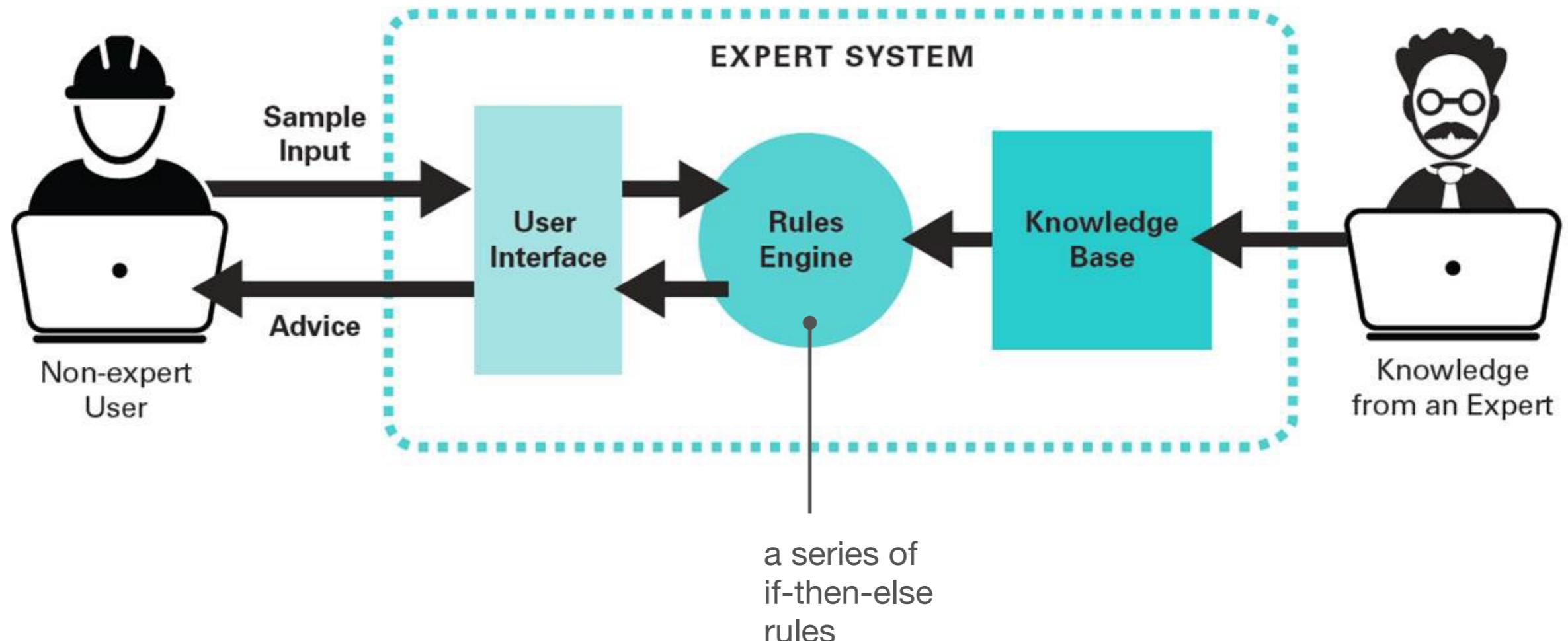


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

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

# The AI time line

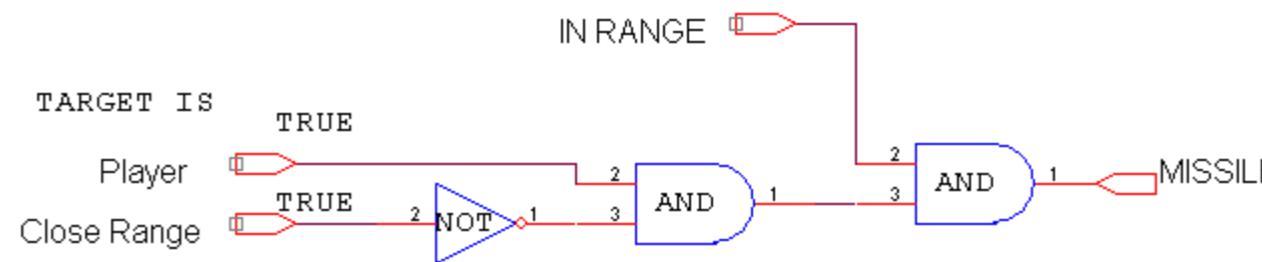
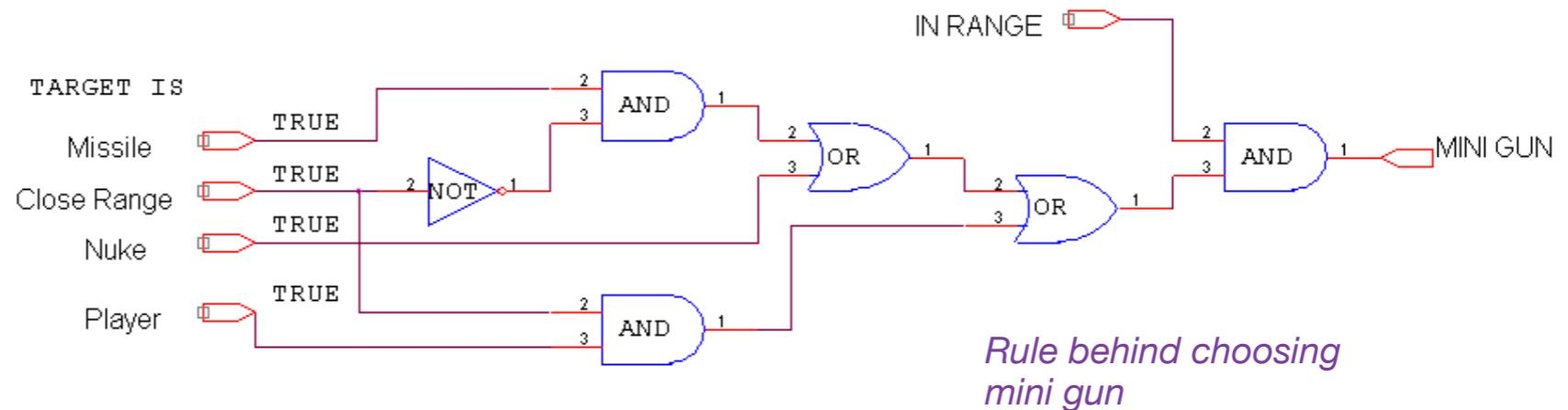
The starting point ...



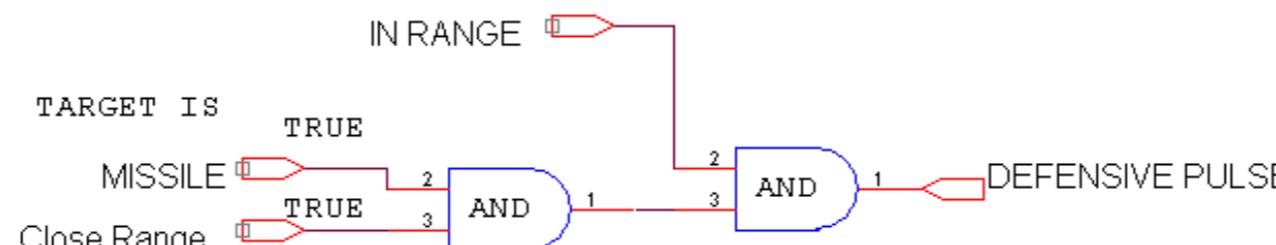
Source: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

# Expert system

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



*Rule behind choosing 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?

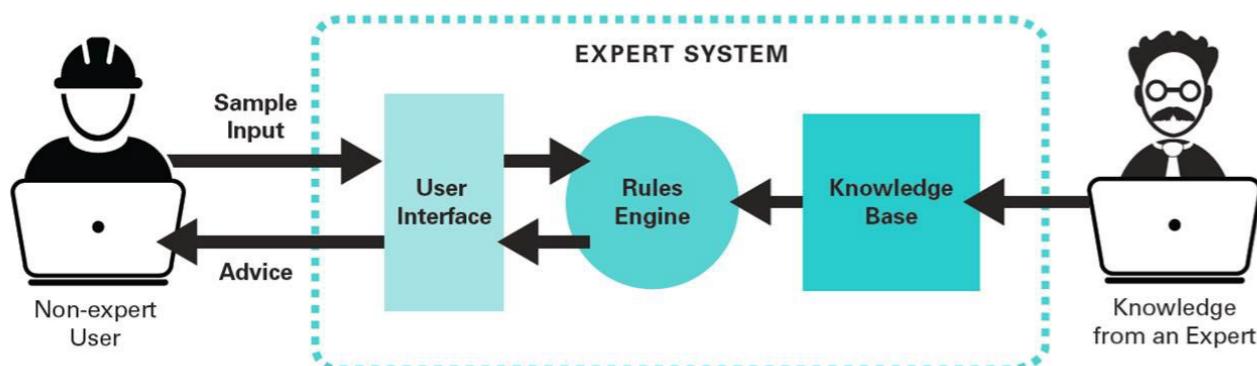


Source: [https://yachtharbour.com/news/venus-spotted-in-mallorca-1887?src=news\\_view\\_page\\_bar](https://yachtharbour.com/news/venus-spotted-in-mallorca-1887?src=news_view_page_bar)

# 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

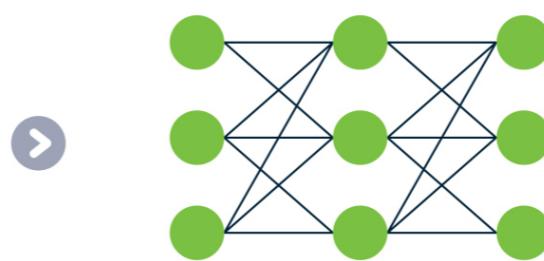
- Classifier figures out (by itself) the underlying pattern between features and output
- This is where the 'learning' happens



Input



Feature extraction



Classification



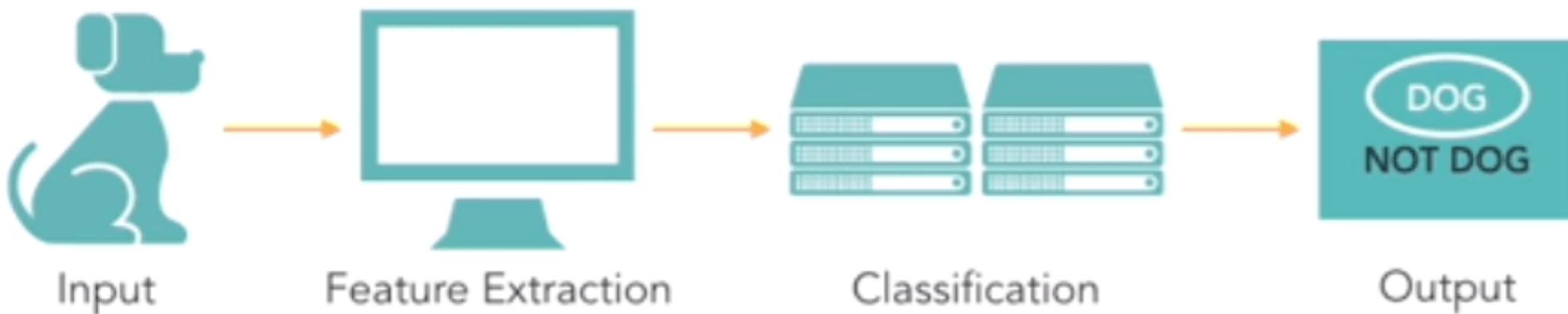
Output

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

# 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?

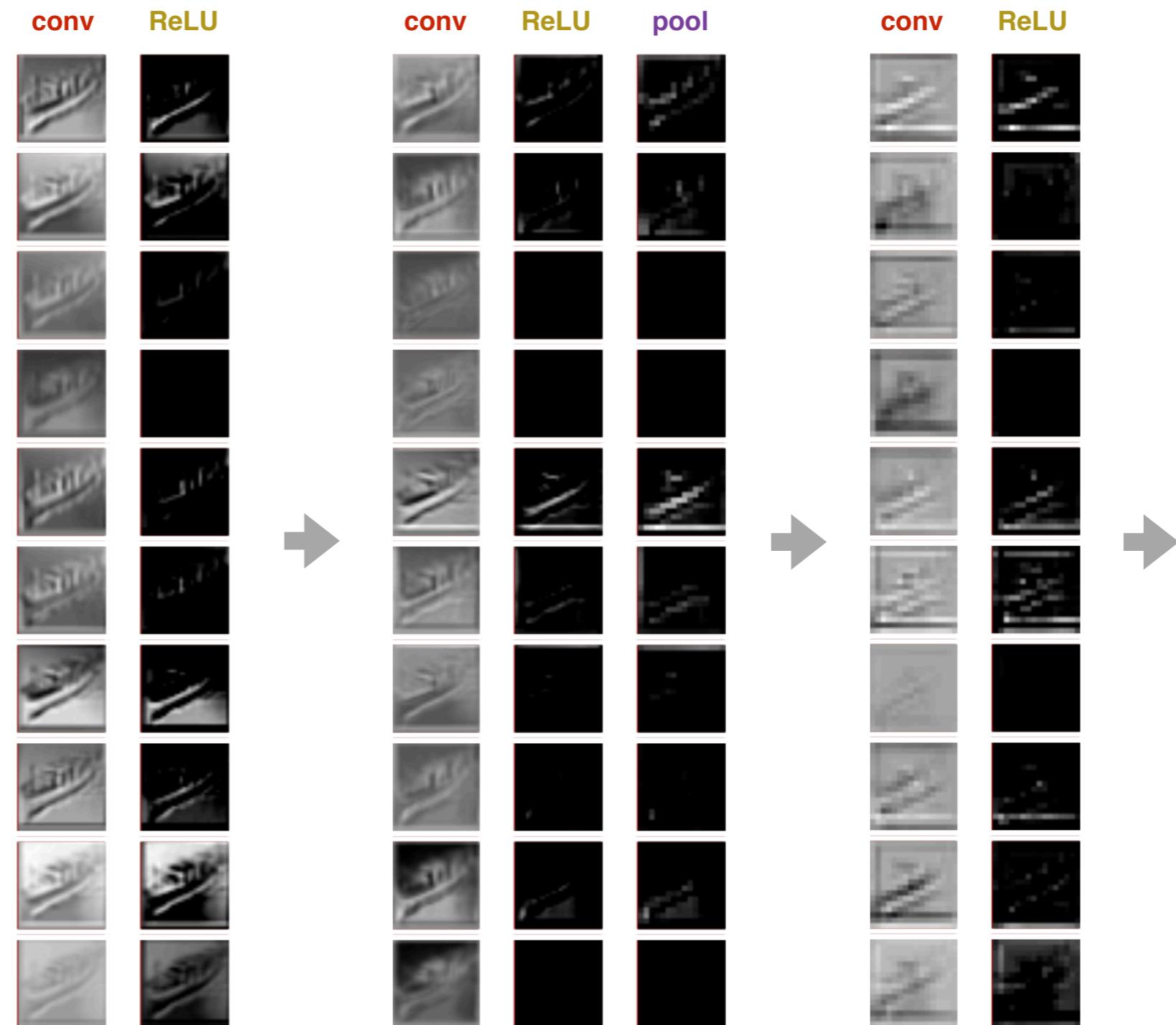
- Instead of we deciding the features, get algorithm to learn the most appropriate features by itself?
- Series of feature extractors?
- All the way from pixels to classifier, layer by layer?
- Train all the layers together?



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

# Deep in action

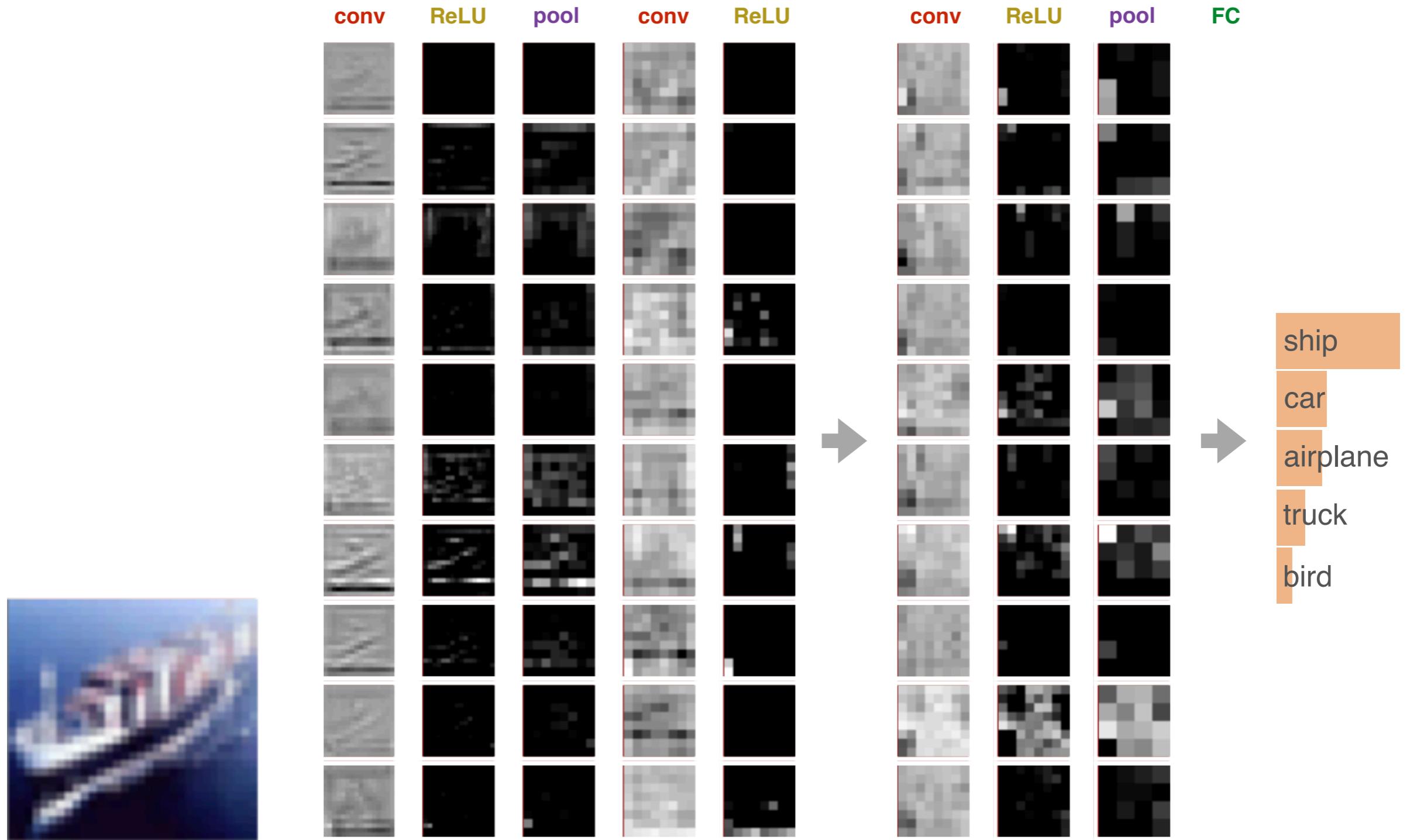
## Part 1



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

# Deep in action

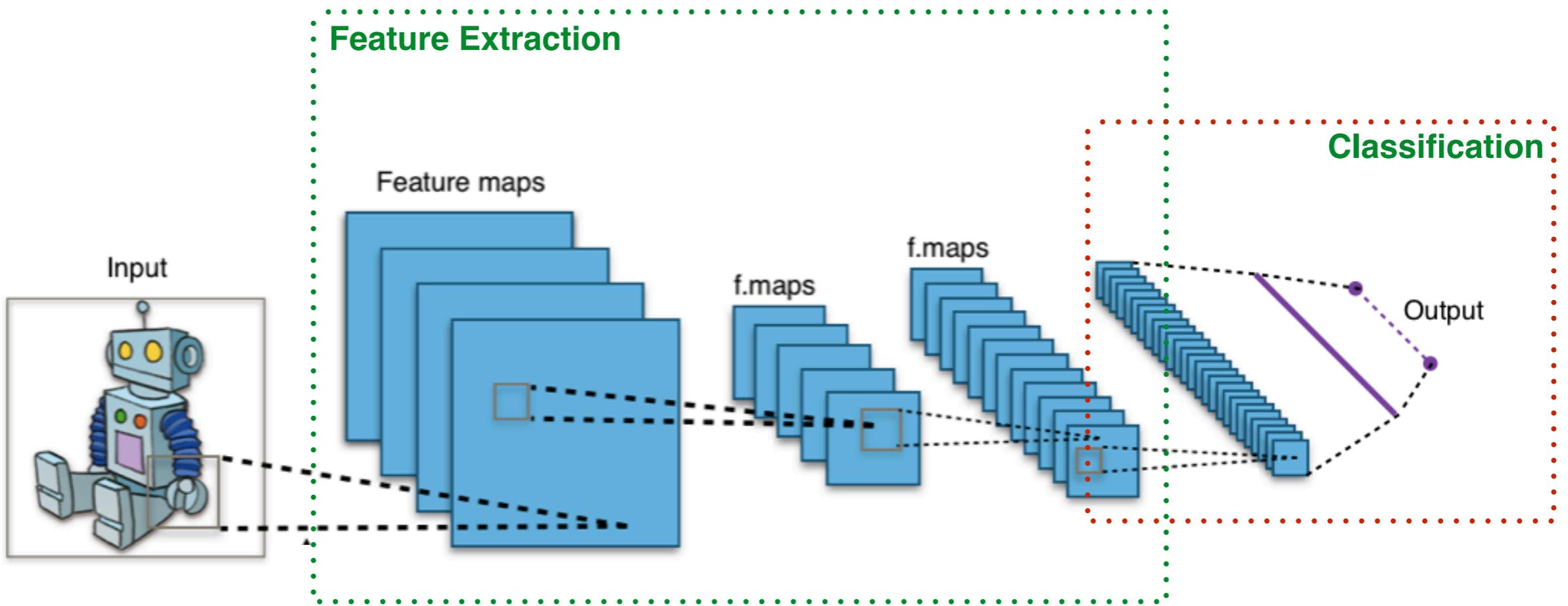
## Part 2



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

# Learning the features

The idea behind convnet



Source: [https://commons.wikimedia.org/wiki/File:Typical\\_cnn.png](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 ...

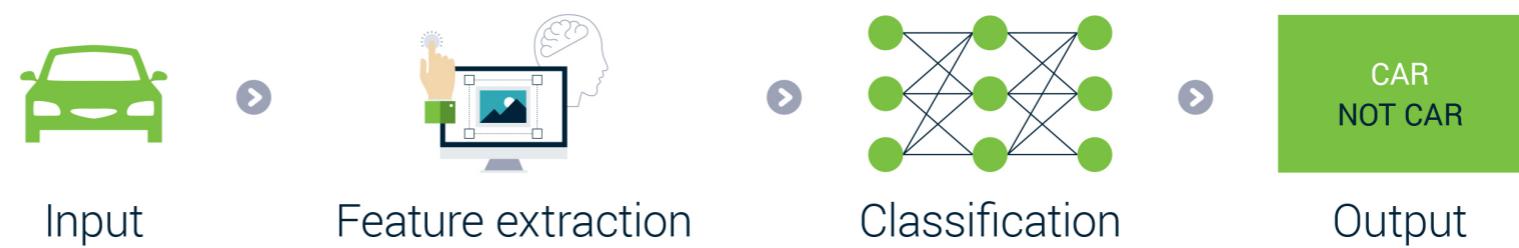
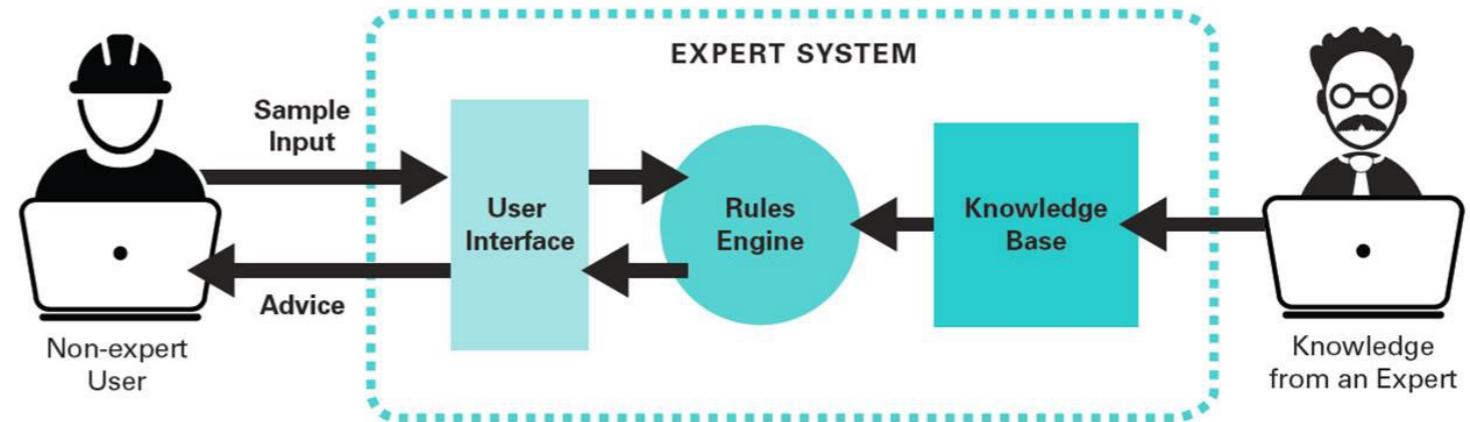
ILSVRC 2013

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
<b>Data dependencies</b>	Excellent performances on a small/medium dataset	Excellent performance on a big dataset
<b>Hardware dependencies</b>	Work on a low-end machine	Requires powerful machine
<b>Feature engineering</b>	Need to understand the features that represent the data	No need to understand the learned features
<b>Execution time</b>	From few minutes to hours	Up to weeks
<b>Interpretability</b>	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>

# **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	<u>Input</u>				<u>Kernel</u>				<u>Output</u>								
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

Output size	input size	padding size	filter size
row	$ $	$ $	$ $
	$\frac{W_r + P_r - F_r}{S_r}$		

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

stride length

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

column

## Example

Calculate the output size

- Assume we have an input of size 128 x 128 (row x column) going into a 2D convolution layer

- The filter / kernel size for the layer is 7 x 7, and the stride is 2 x 2, no padding is applied, calculate the output size of the 2D convolution

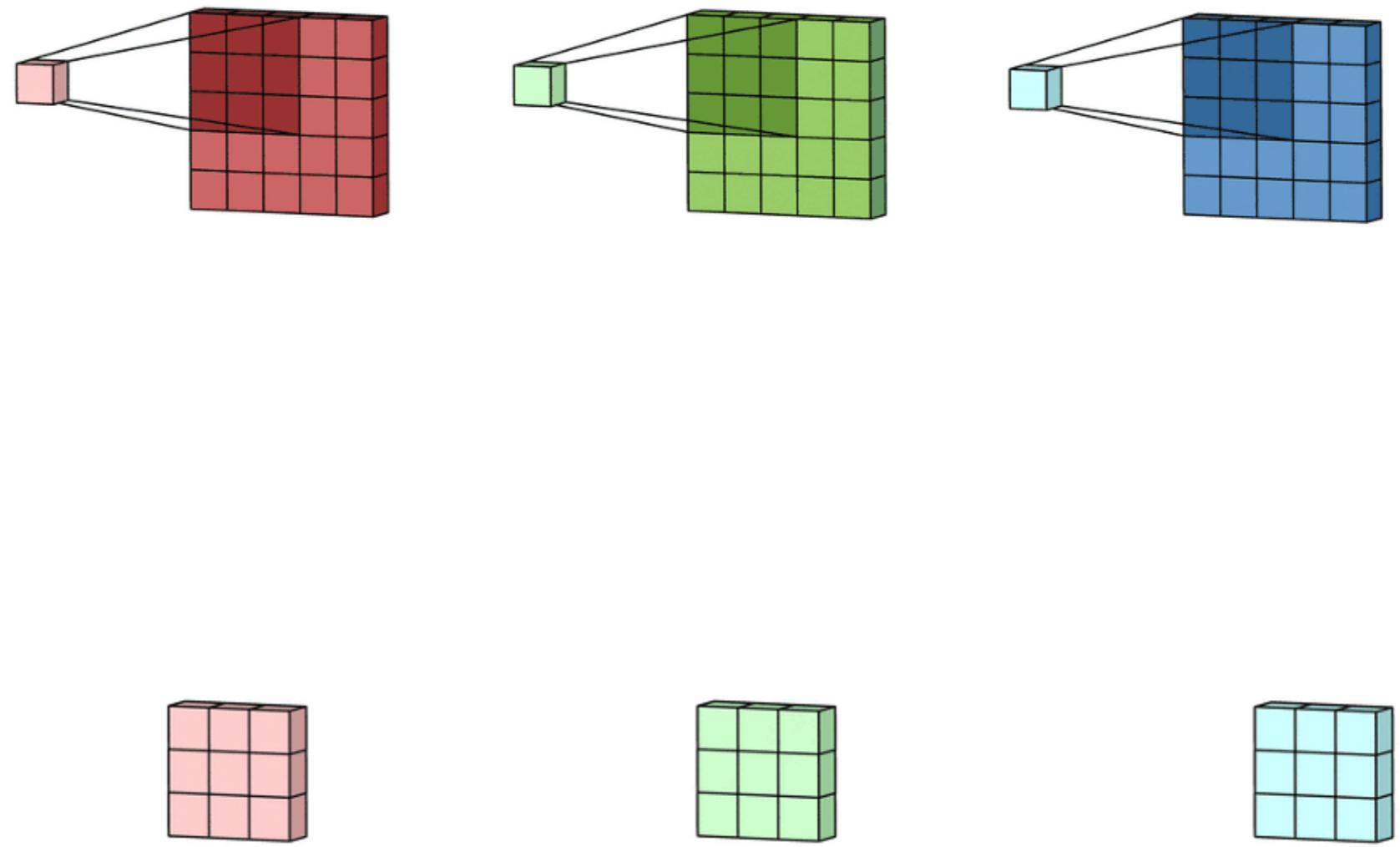
Output size	input size	padding size	filter size
row	$ $	$ $	$ $
	$M_r = \left\lfloor \frac{W_r + P_r - F_r}{S_r} \right\rfloor + 1$		

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

$$\begin{aligned} M_r &= M_c = \left\lfloor \frac{128 + 0 - 7}{2} \right\rfloor + 1 \\ &= \left\lfloor \frac{121}{2} \right\rfloor + 1 \\ &= 61 \end{aligned}$$

# 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	<b>9</b>	2	3	2
<b>8</b>	1	<b>6</b>	0	7
0	3	2	1	1



<b>9</b>	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

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

row  
stride length

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

column

# Maxpooling

# Determine the output size

- Assume we have an input of size  $61 \times 61$  (row x column) going into a 2D pooling layer
  - The kernel size for the layer is  $4 \times 4$ , and the stride is  $2 \times 2$ , calculate the output size of the 2D pooling layer

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

row | | stride length

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

column

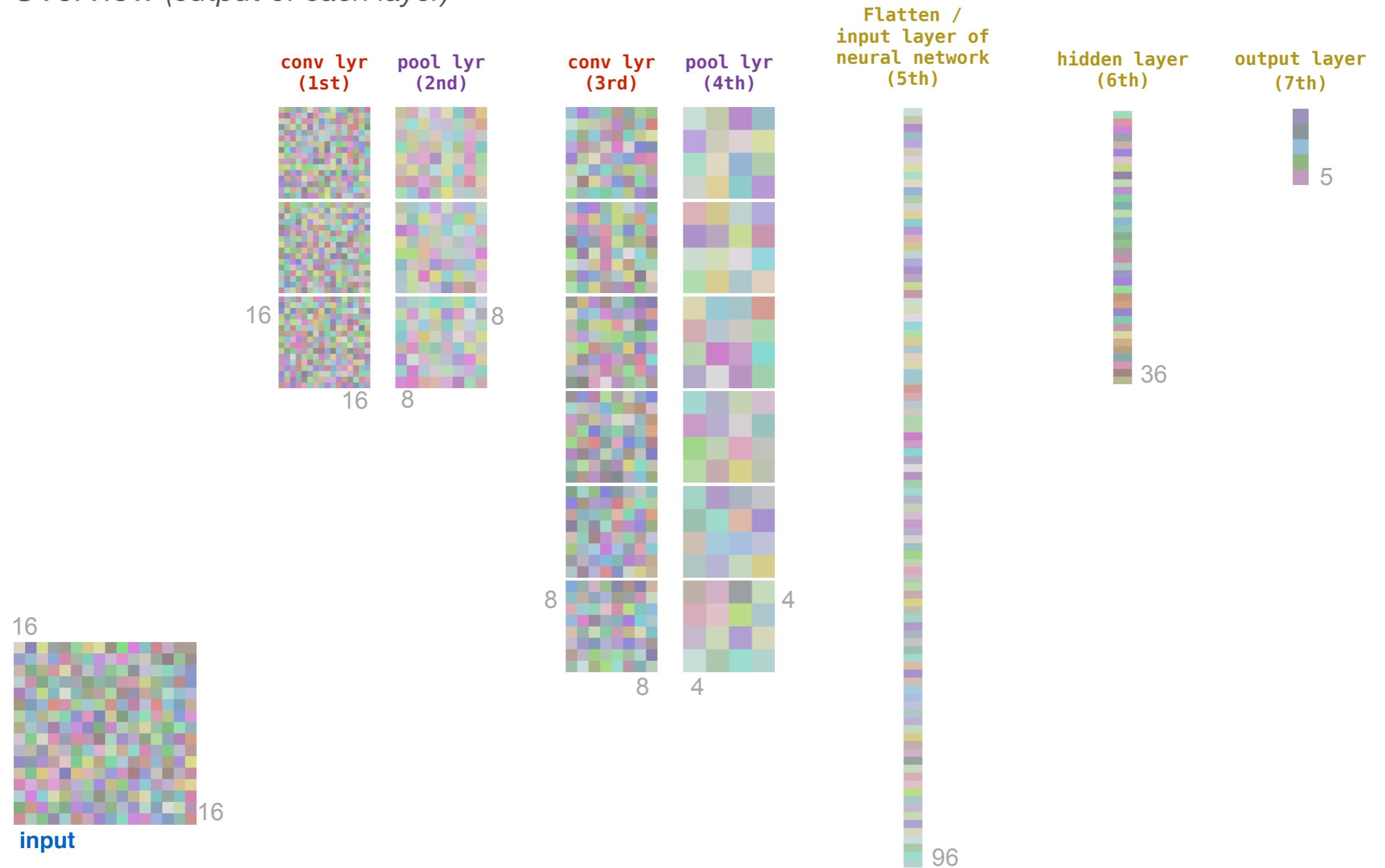
$$M_r = M_c = \left\lfloor \frac{61 - 4}{2} \right\rfloor + 1$$

$$= \left\lfloor \frac{57}{2} \right\rfloor + 1$$

$$= |28.5| + 1 = 29$$

# Convolutional neural network

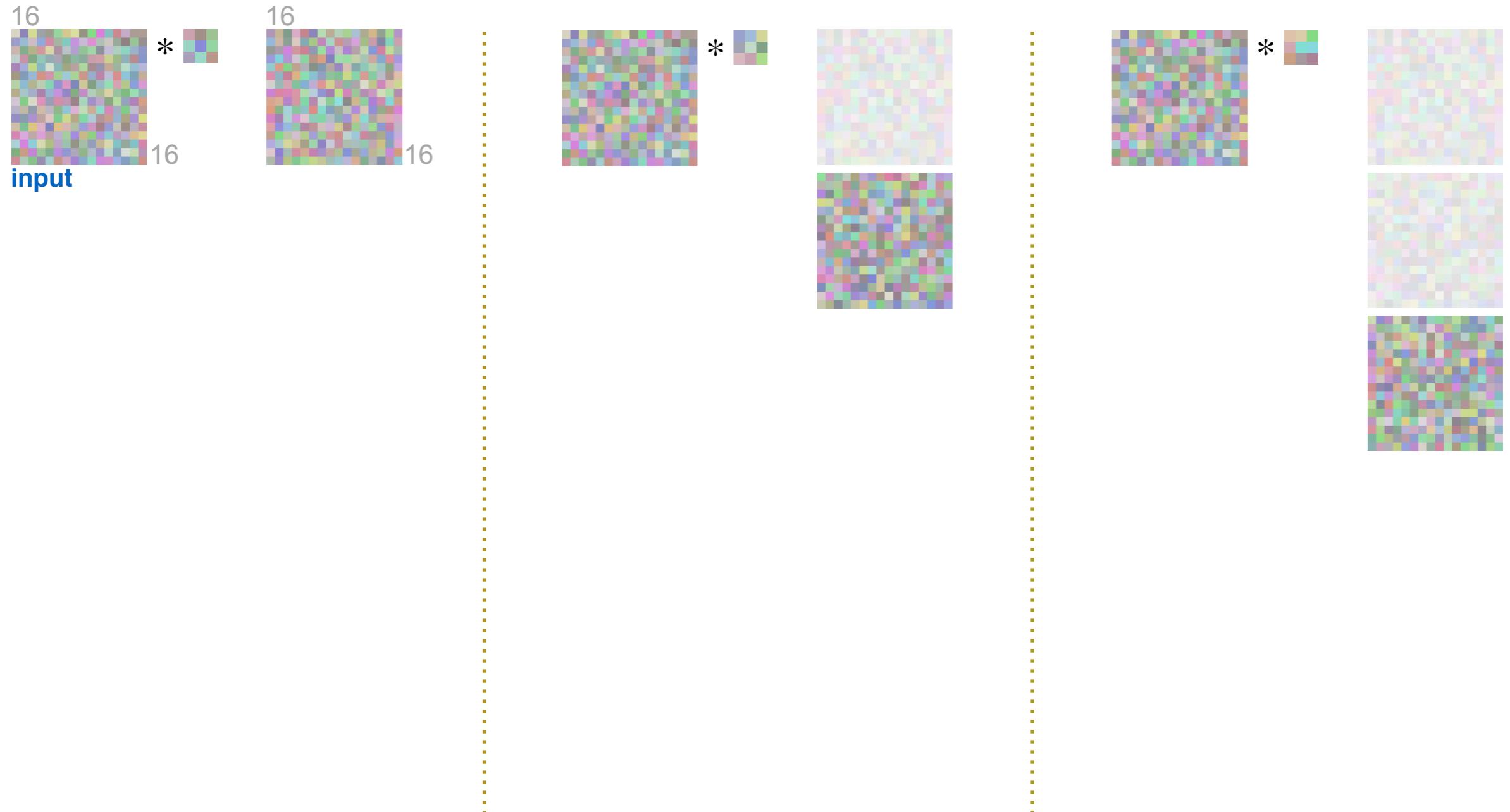
Overview (*output of each layer*)



# The making of ...

The first convolutional layer  
(part 1)

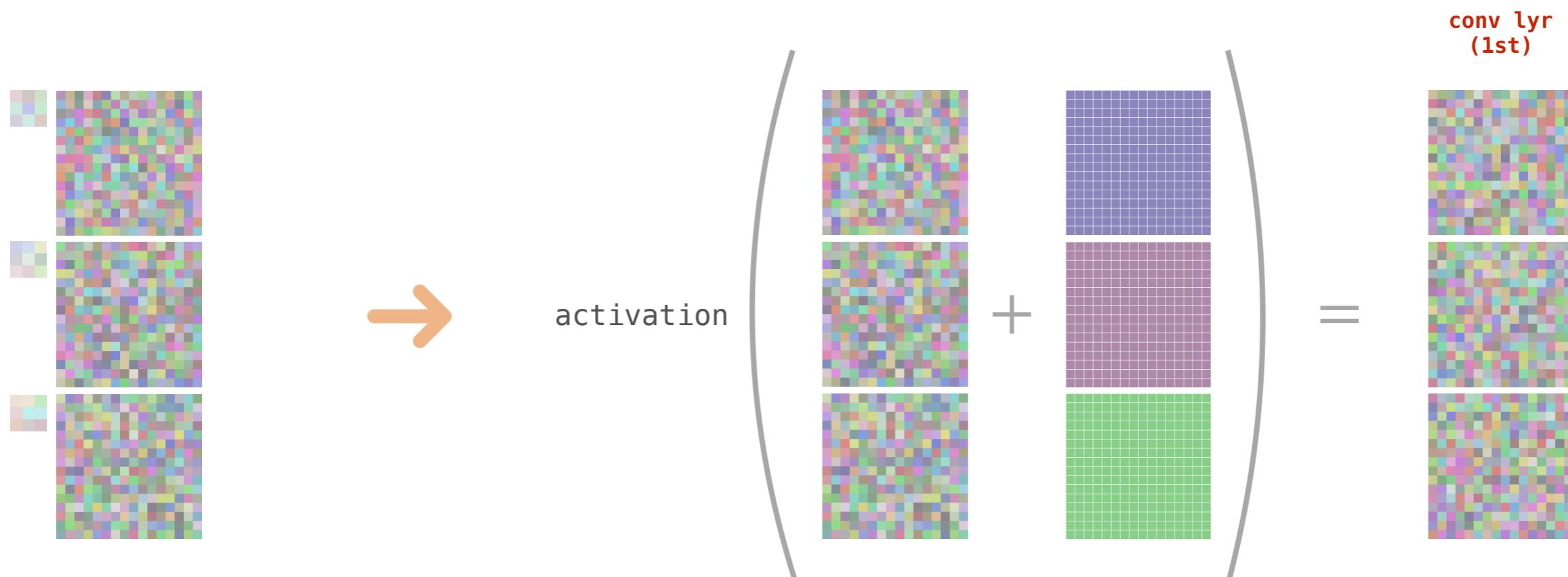
- Performs 3 separate 2D convolutions (with padding) to generate 3 intermediate outputs



# The making of ...

The first convolutional layer  
(part 2)

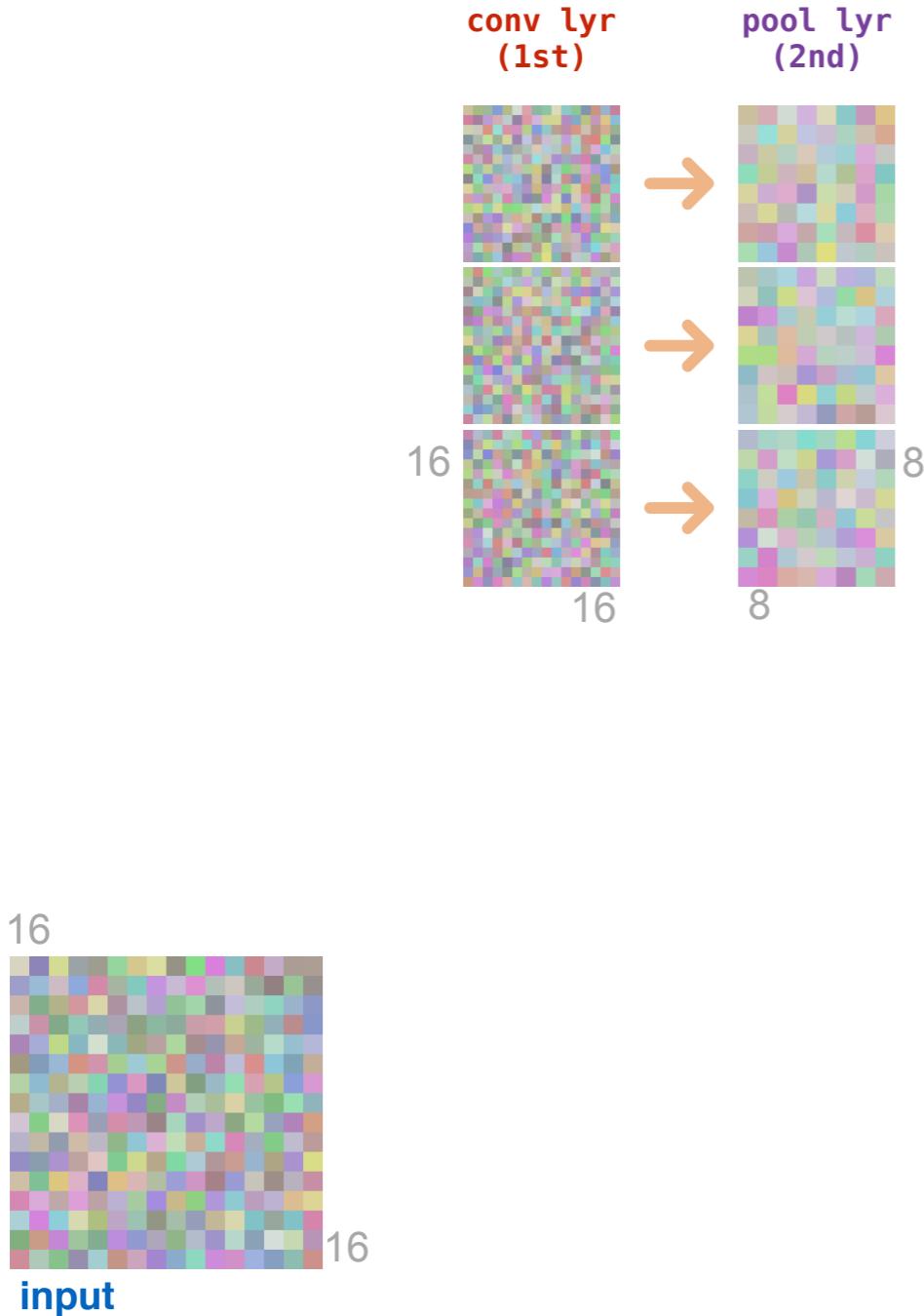
- Add bias to each convolution output, and apply activation function to get the final output for the convolutional layer



# The making of ....

## The pooling layer

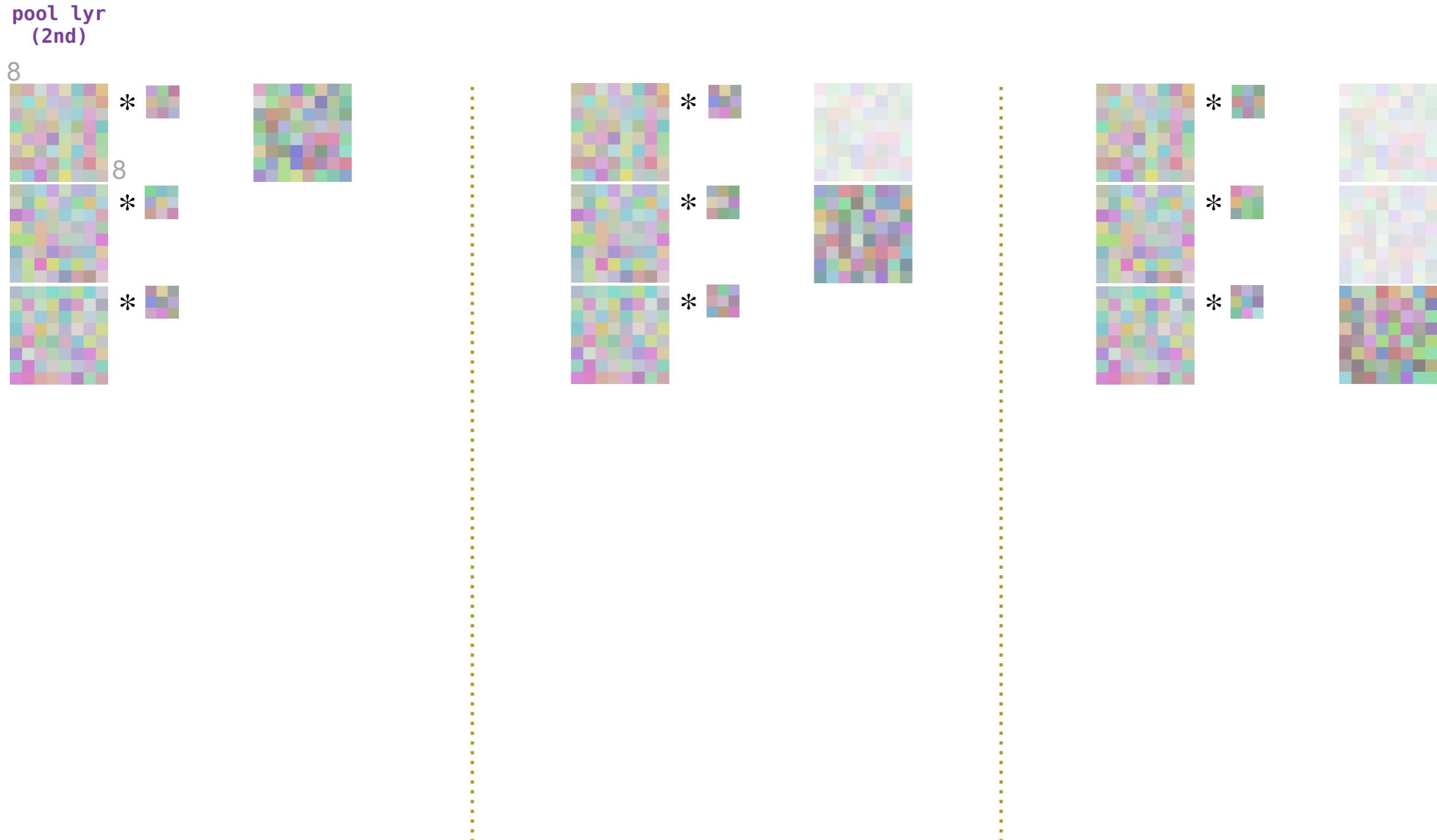
- Apply  $2 \times 2$  max-pooling (stride 2) on the outputs from the first convolutional layer



# The making of ...

## The second convolutional layer (part 1)

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

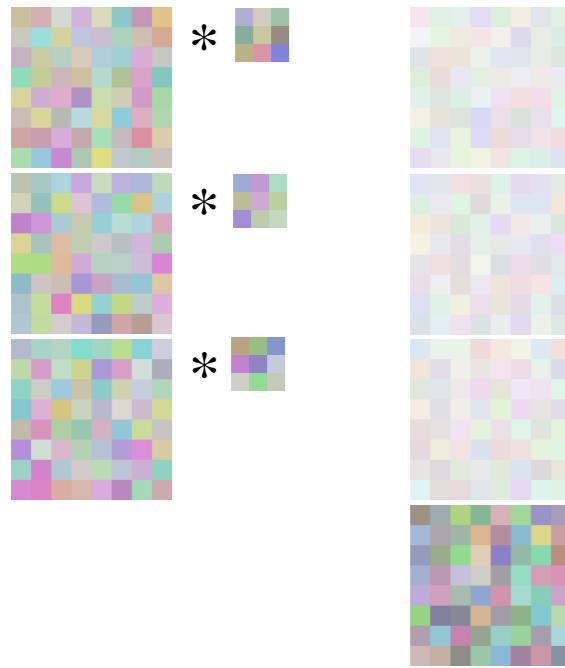


# The making of ...

The second convolutional layer (part 2)

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

pool lyr  
(2nd)



\*



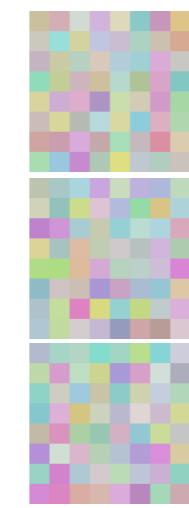
\*



\*



⋮



\*



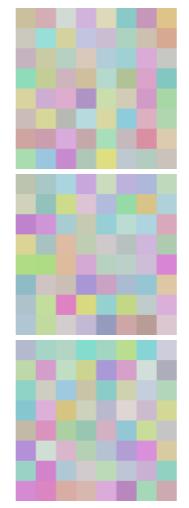
\*



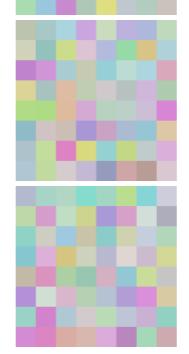
\*



⋮



\*



\*



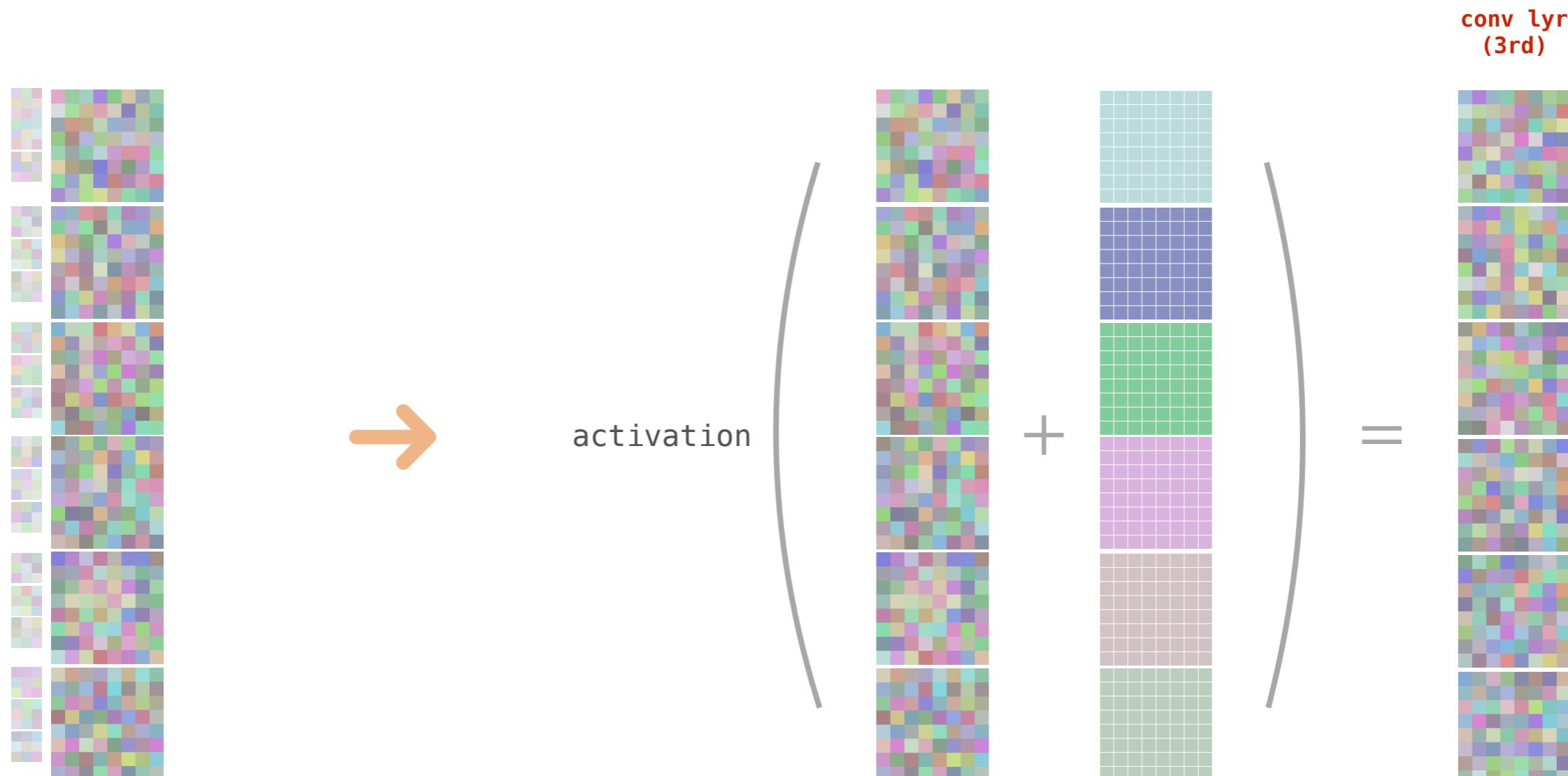
\*



# The making of ...

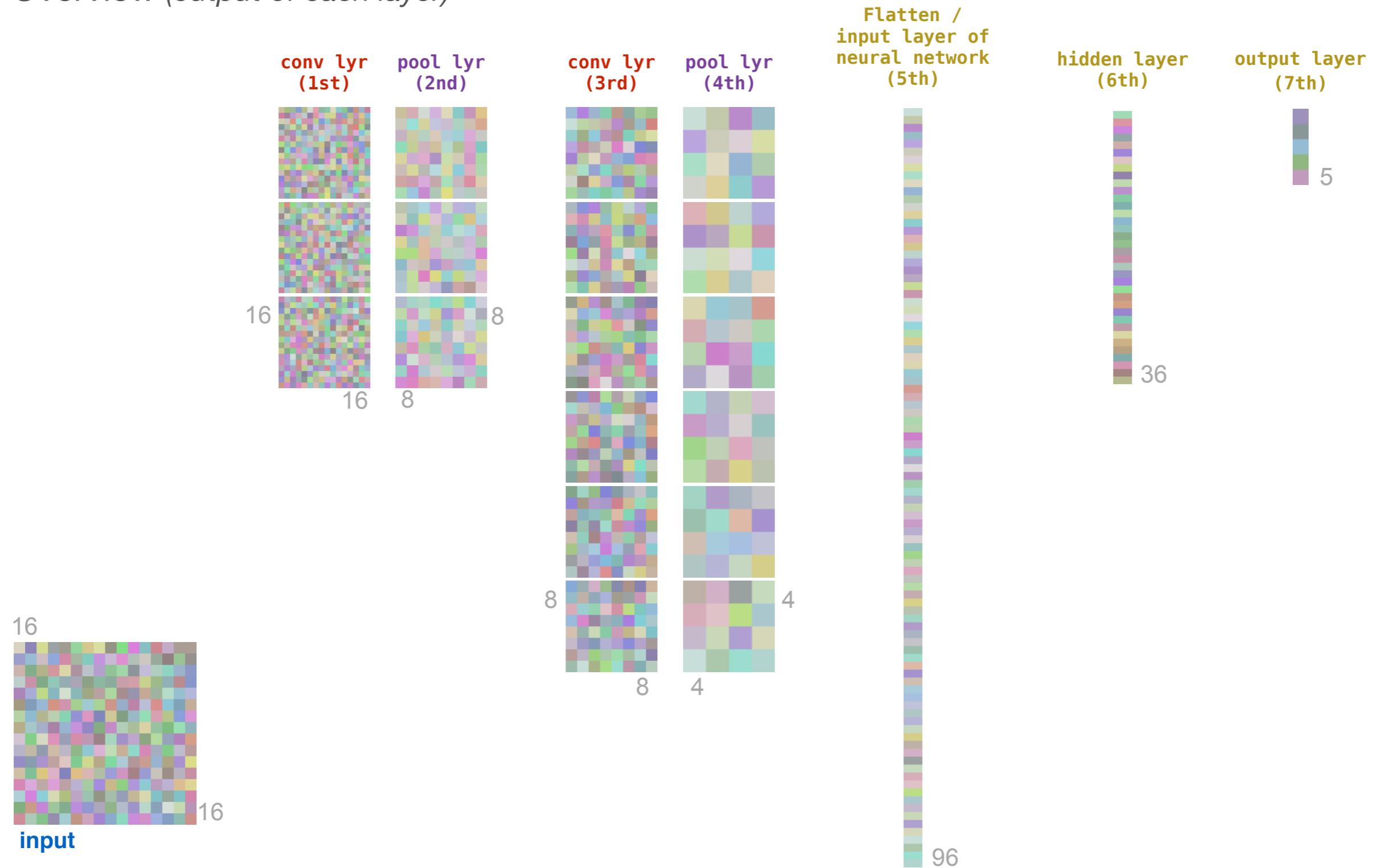
The first convolutional layer  
(part 3)

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



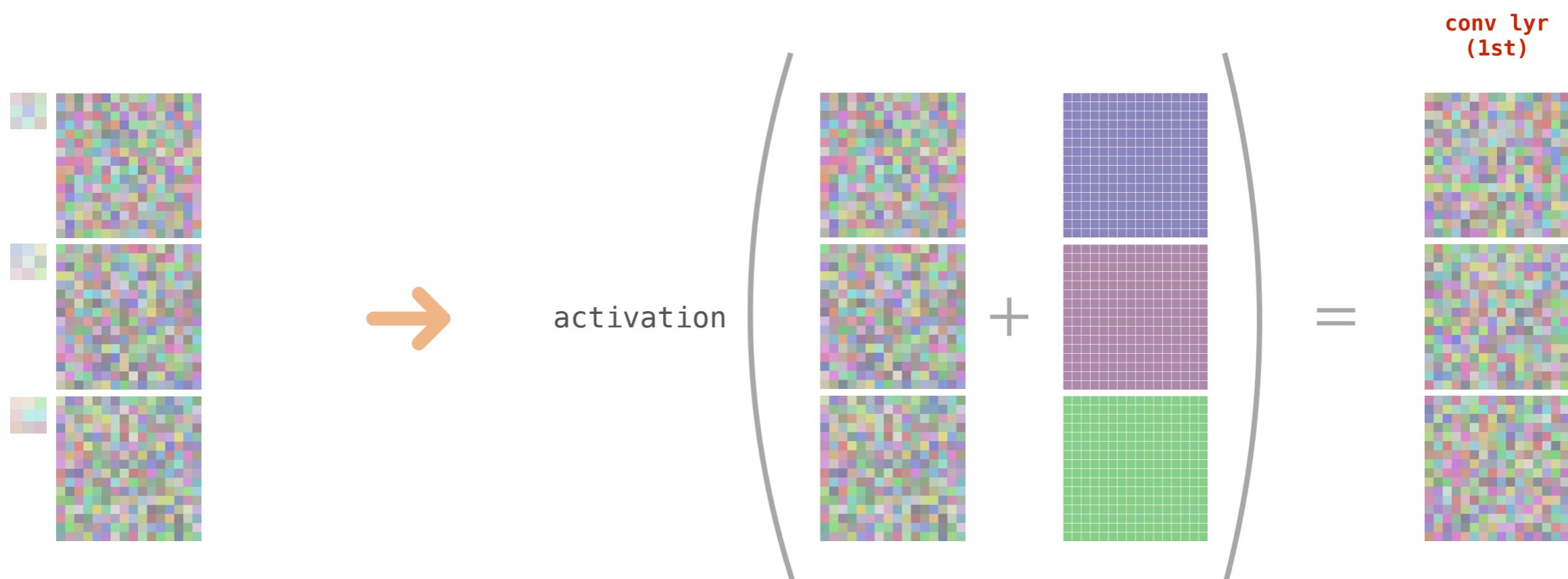
# Convolutional neural network

Overview (*output of each layer*)



# Calculating parameters

For the first convolutional layer

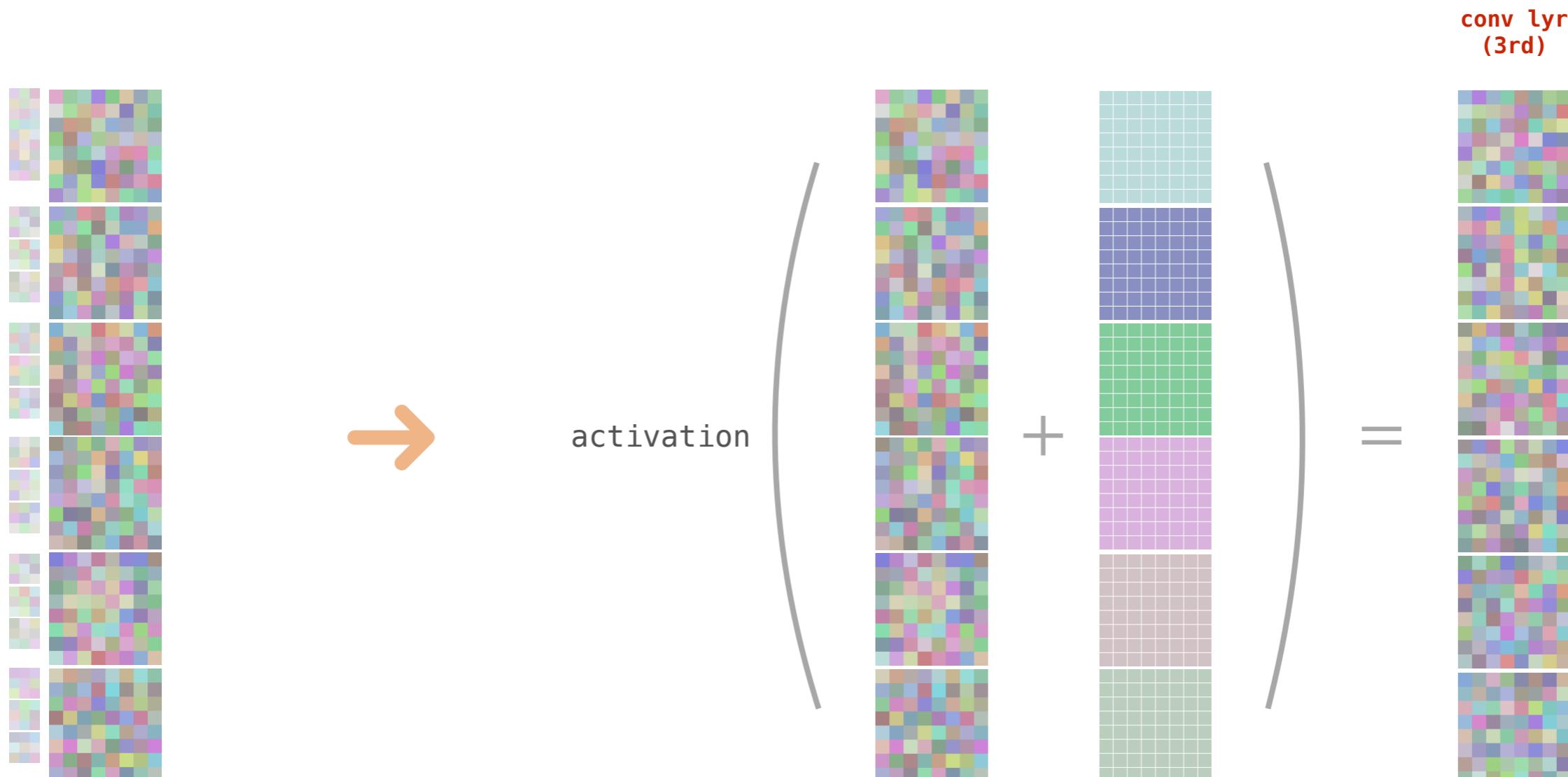


Number of parameters:

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

# 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

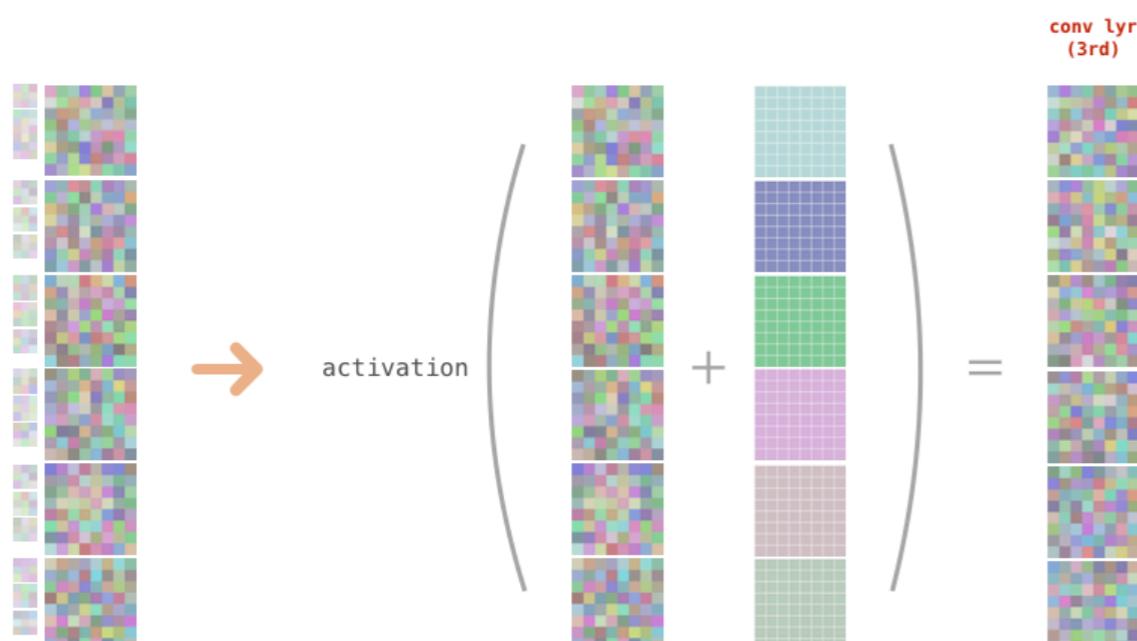
Calculate the parameters

- 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

$$p_{tr} = [C_i \times (F_r \times F_c) + 1] \times D$$

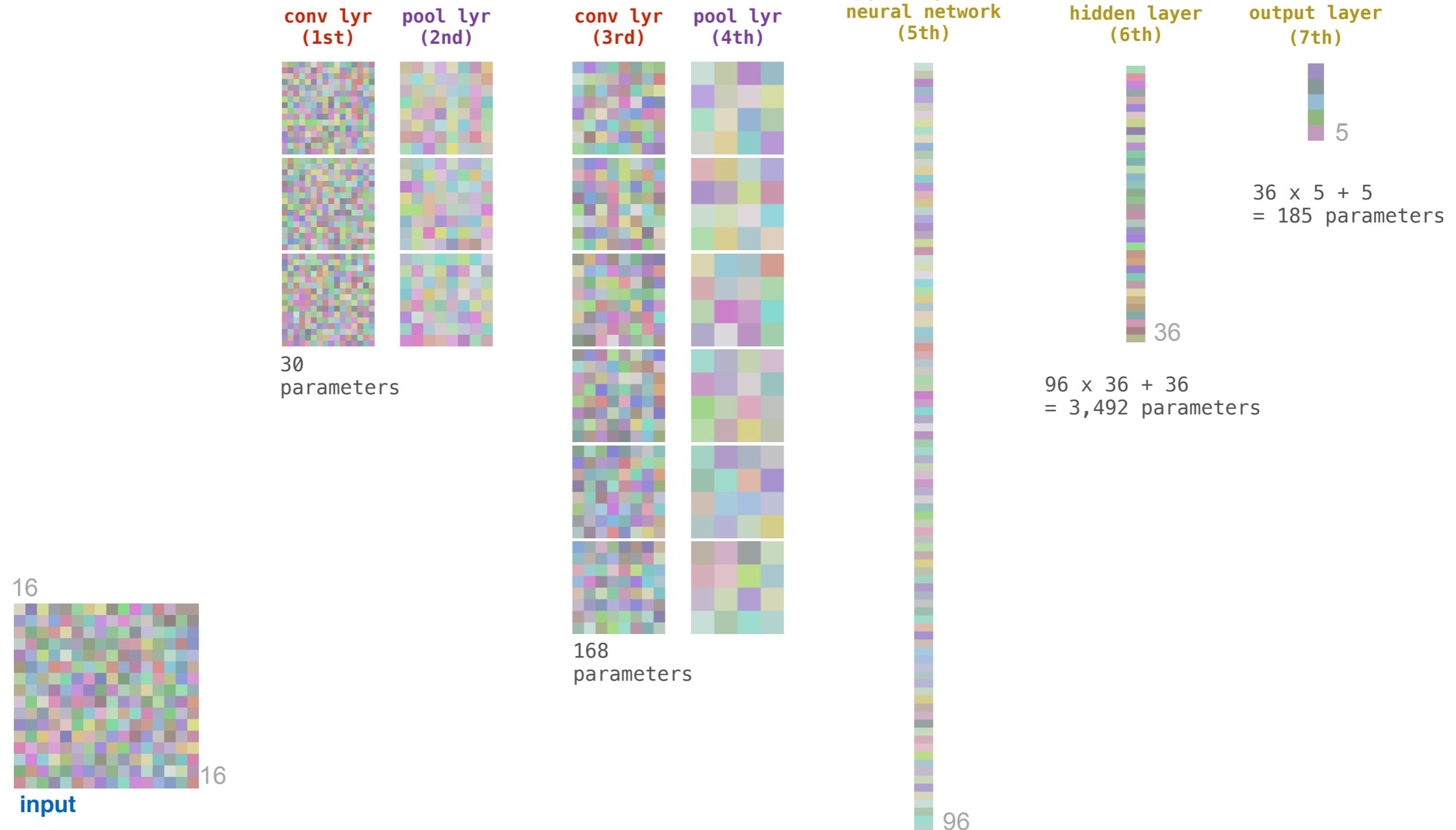


Number of parameters:

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

# Convolutional neural network

Number of Parameters involved



# Parameters

Determine the number of parameters

- Assume we have an input of size  $128 \times 128 \times 3$  (row  $\times$  column  $\times$  channel) going into a 2D pooling layer

- The kernel size for the layer is  $7 \times 7$ , the stride is  $2 \times 2$ , and the number of channels in the output of this layer is 18

$$p_{tr} = [C_i \times (F_r \times F_c) + 1] \times D$$

the  
number of  
channels  
in input

filter size  
for column

the  
number of  
channels  
in output

filter size  
for row

$$\begin{aligned} p_{tr} &= [3 \times (7 \times 7) + 1] \times 18 \\ &= [147 + 1] \times 18 \\ &= 2664 \end{aligned}$$

# **Exercise**

# Convolutional neural network

Calculate the necessary

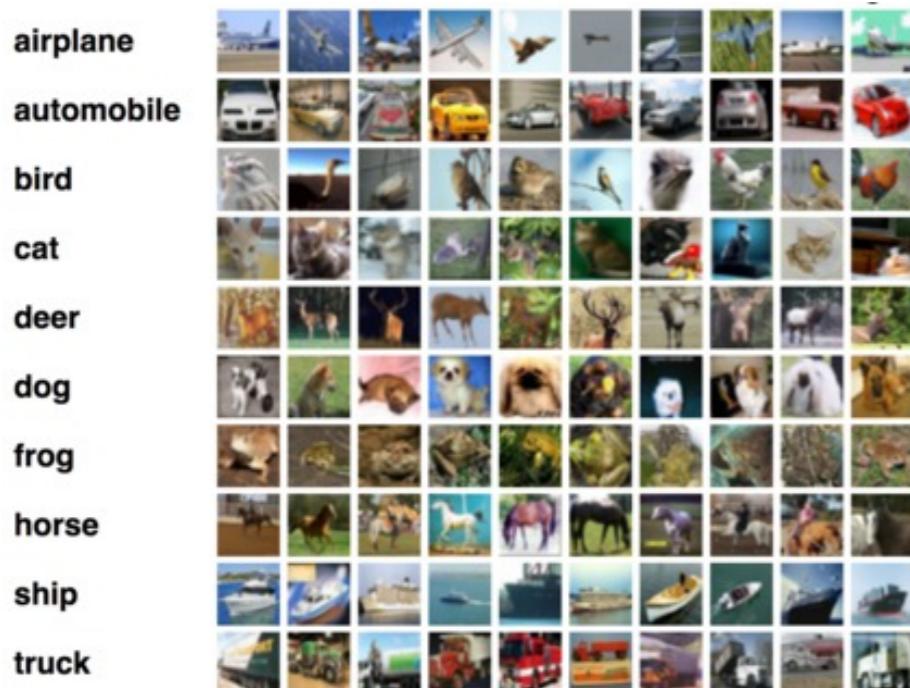
- Assume the size of an input to a layer is ( 32, 32, 3 )
- No padding for all convolutions

Layer	Type	Kernel	Stride	No. of feature maps / neurons	Input size	Output size	No. of parameters
1	Conv	(3,3)	(1,1)	16		(32,32,3)	
2	Pool	(2,2)	(2,2)	16			
3	Conv	(5,5)	(1,1)	32			
4	Pool	(2,2)	(2,2)	32			
5	Conv	(3,3)	(1,1)	64			
6	Dense	-	-	128			
7	Dense	-	-	2			2

# **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/>

# 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?

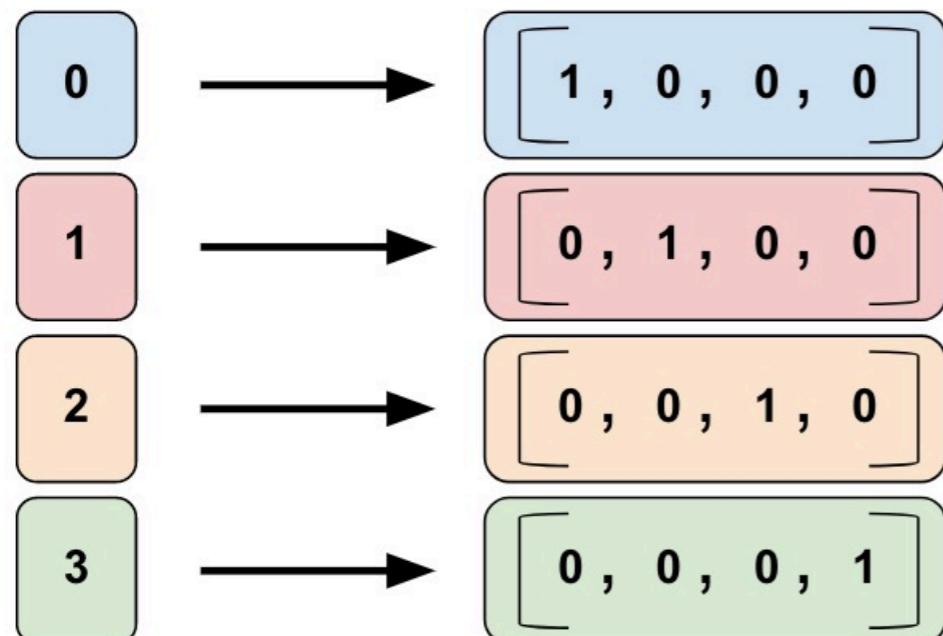
# Cifar 10

## 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'
```

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

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

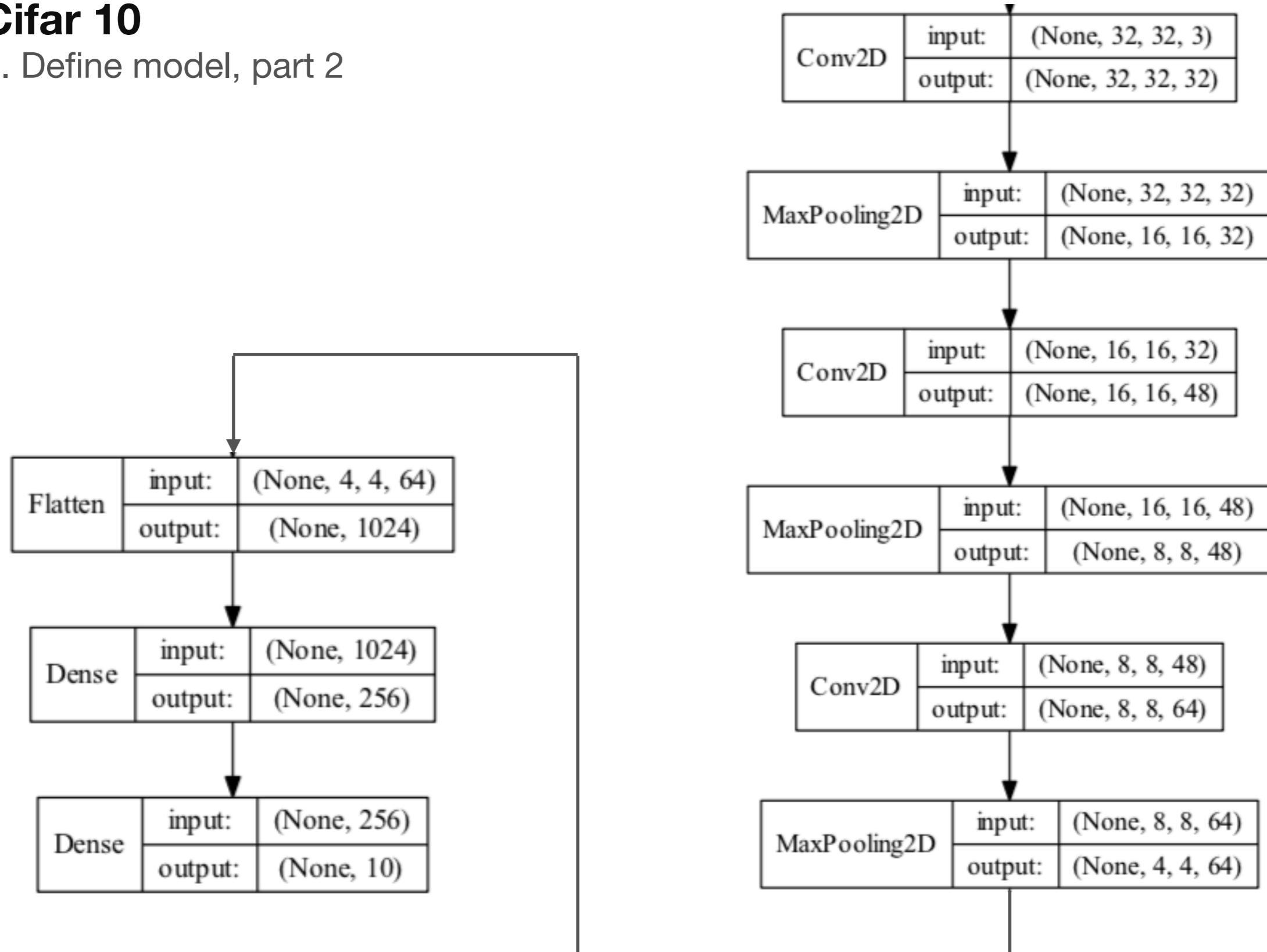
$$w = w - \alpha \cdot \frac{dw}{\sqrt{E_{dw}} + \epsilon}$$

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

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

# Cifar 10

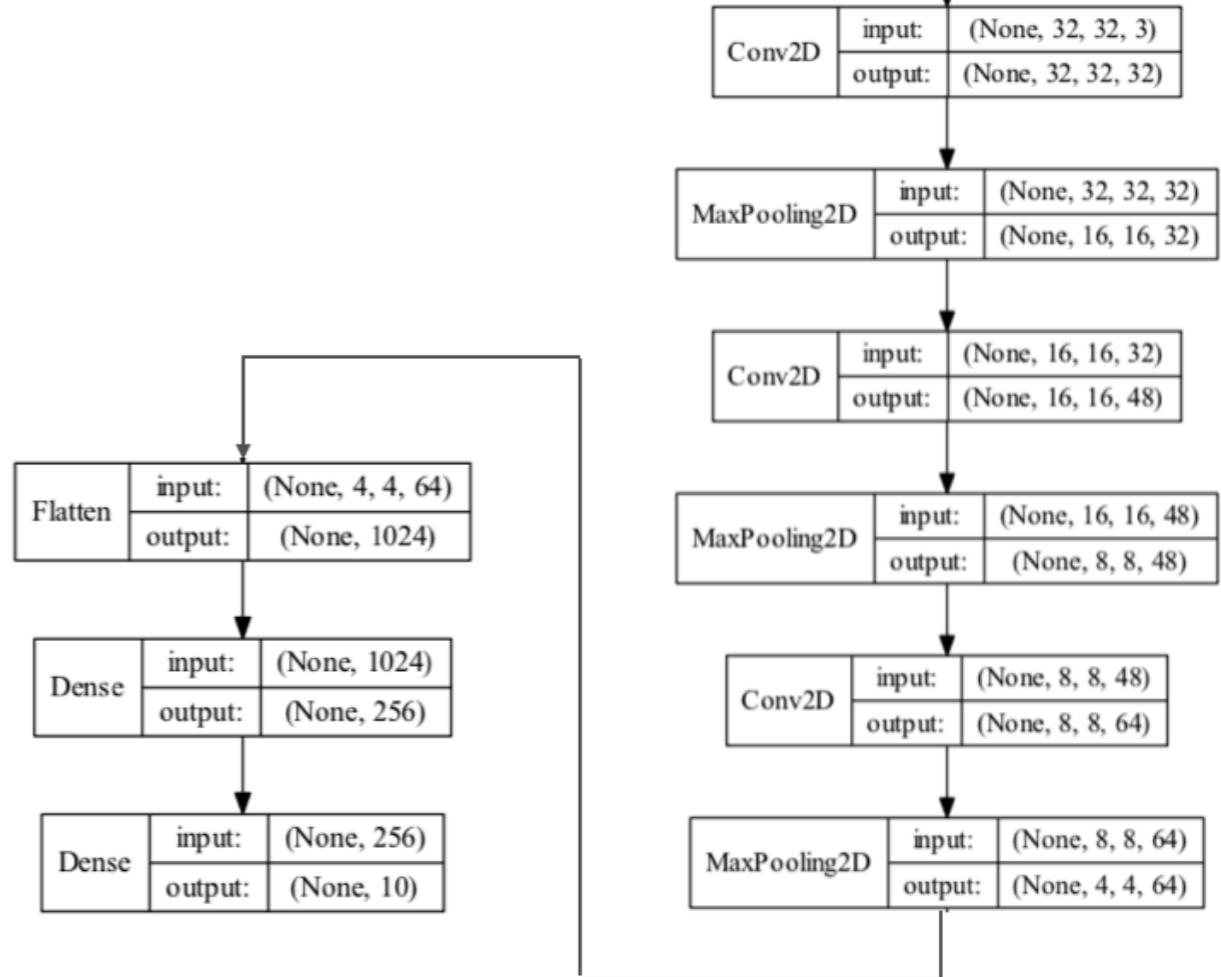
## 4. Define model, part 2



# 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 #
conv2d (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 48)	13872
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 48)	0
conv2d_2 (Conv2D)	(None, 8, 8, 64)	27712
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 64)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 256)	262400
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)
```

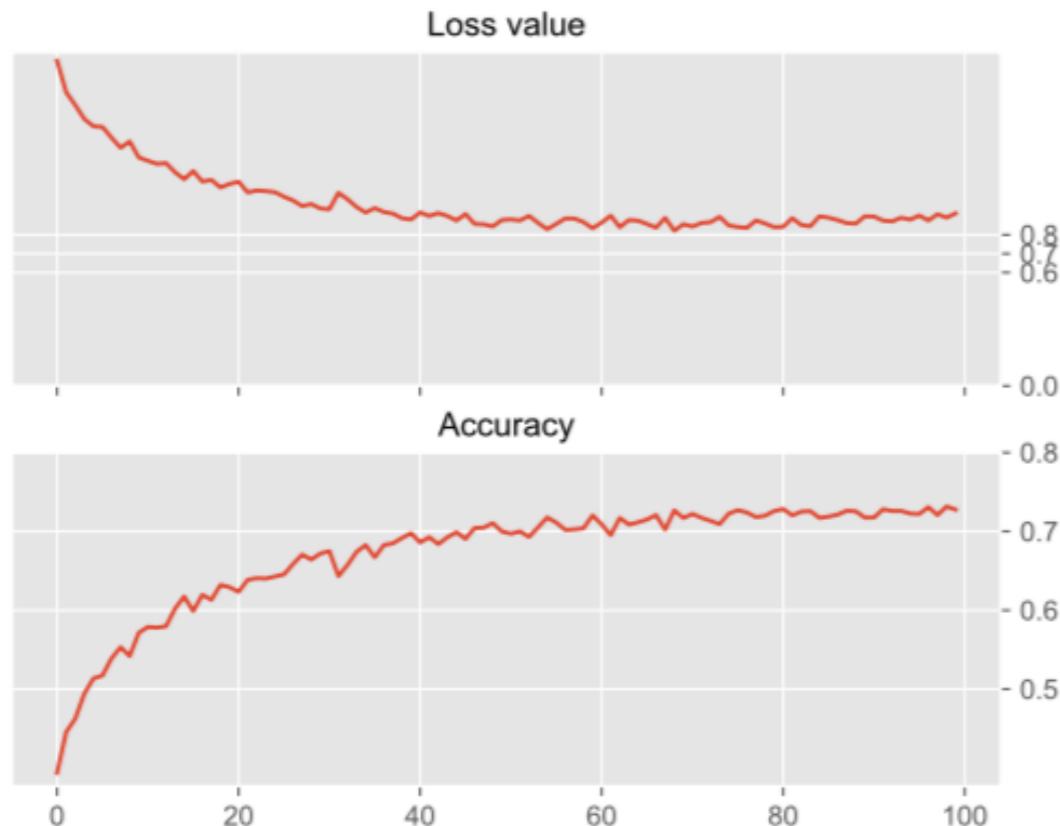
	precision	recall	f1-score	support	[[738 18 36 18 40 4 10 11 52 73]
airplane	0.8013	0.7380	0.7683	1000	[ 7 771 7 10 1 2 11 2 18 171]
automobile	0.8653	0.7710	0.8154	1000	[ 60 6 595 75 108 44 51 35 9 17]
bird	0.6959	0.5950	0.6415	1000	[ 17 11 49 610 81 103 53 33 11 32]
cat	0.5365	0.6100	0.5709	1000	[ 11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



# Contact eGL

**Singapore e-Government Leadership Centre  
National University of Singapore  
29 Heng Mui Keng Terrace  
Block D & E  
Singapore 119620**

**Tel** : (65) 6516 1156  
**Fax** : (65) 6778 2571  
**URL** : [www.egl.sg](http://www.egl.sg)  
**Email** : [egl-enquiries@nus.edu.sg](mailto:egl-enquiries@nus.edu.sg)

*Transform*

*Lead*

*Inspire*