# 인공지능 기반 설계 이론 및 사례 연구

# 5차) Convolutional Neural Network (CNN)

2020년 9월

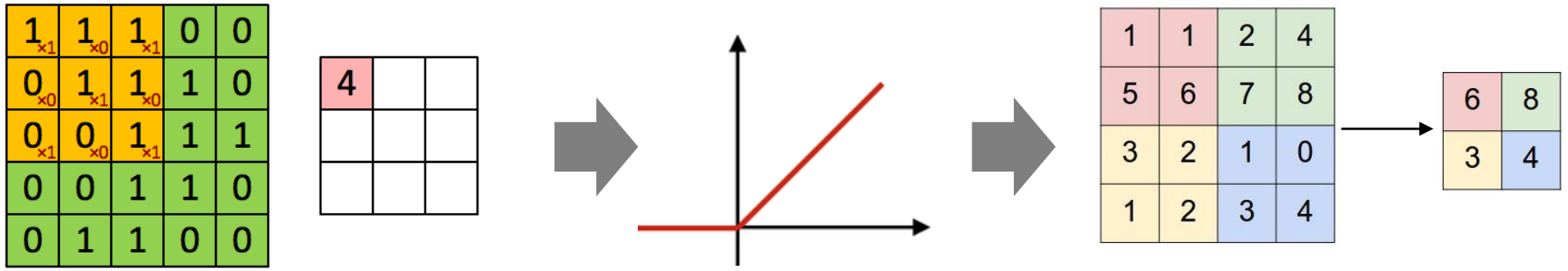
강남우

기계시스템학부

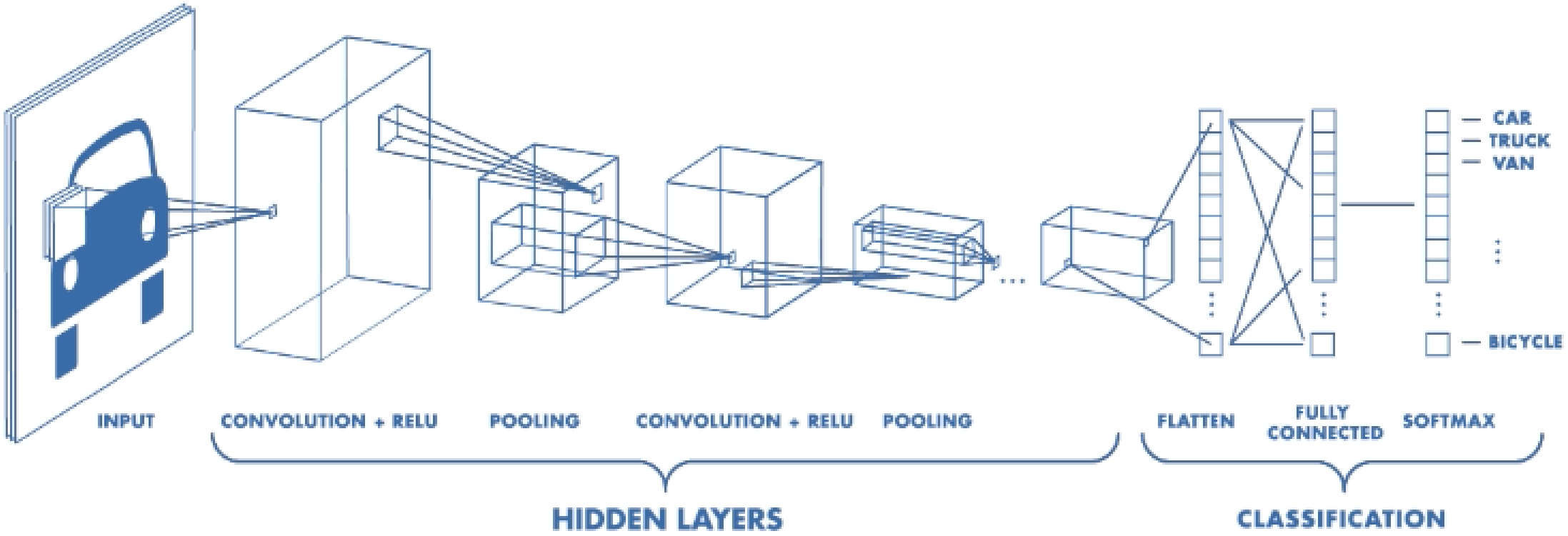
숙명여자대학교

SMART DESIGN LAB

# Convolutional Neural Network (CNN)




**Convolution** **Relu** **Pooling (Max)**

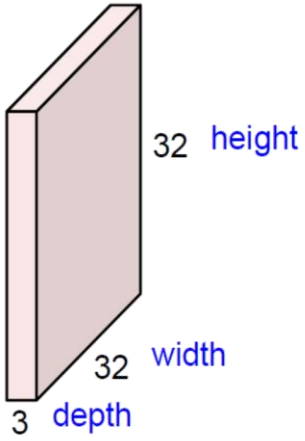


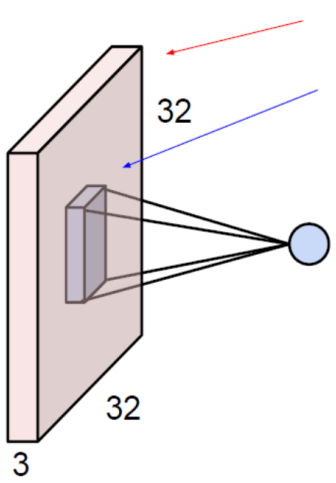



<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>

# Convolution Layer

32x32x3 image -> preserve spatial structure

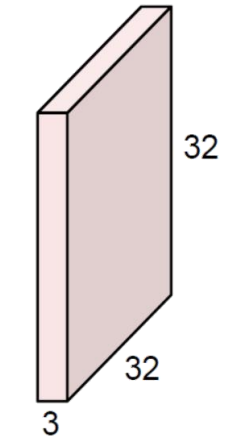


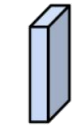




**1 number:**
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

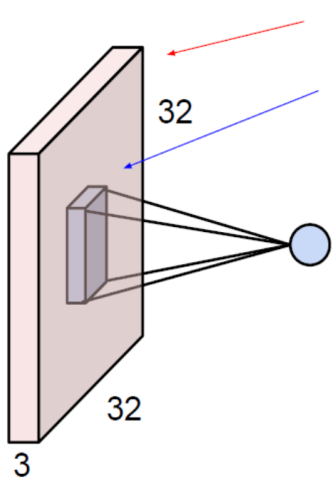
$$w^T x + b$$

Filters always extend the full depth of the input volume

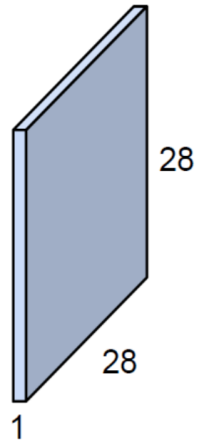



**Convolve** the filter with the image
i.e. “slide over the image spatially, computing dot products”

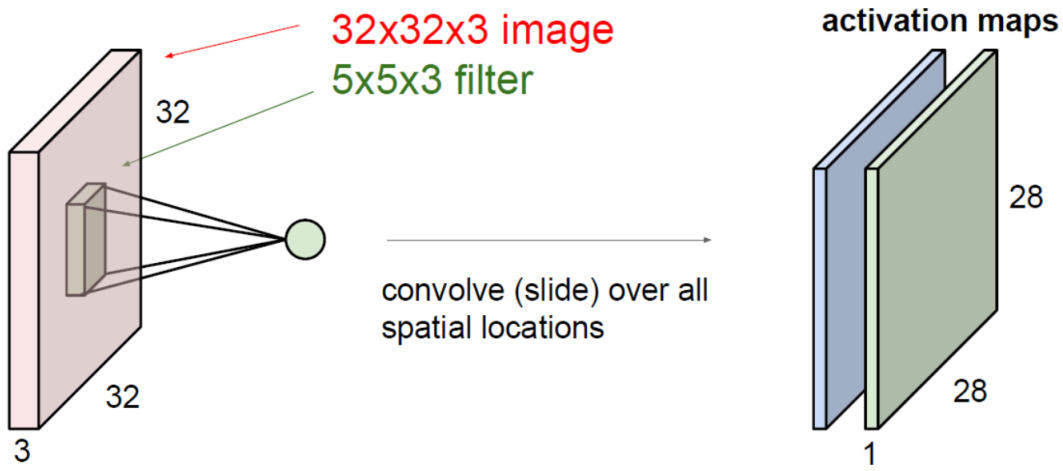



convolve (slide) over all spatial locations

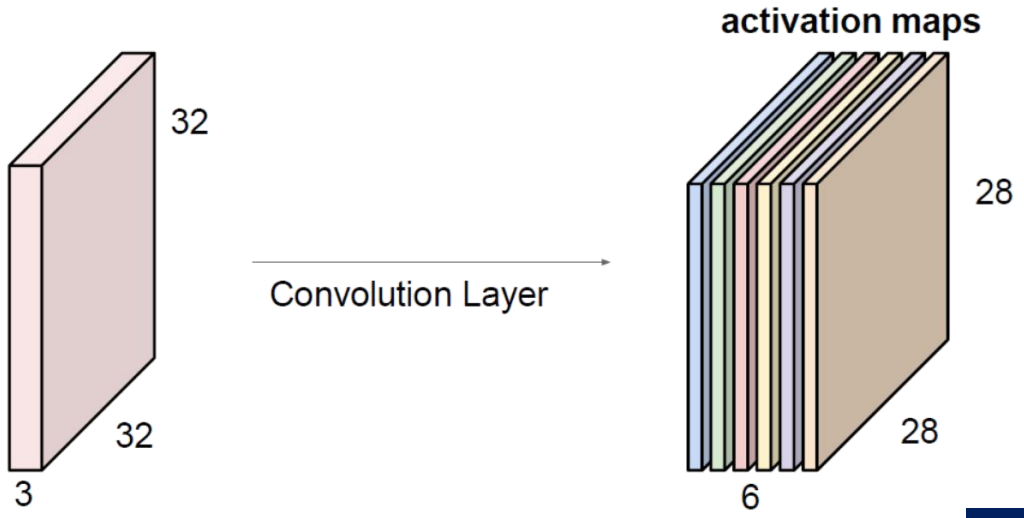


# Convolution Layer

consider a second, green filter



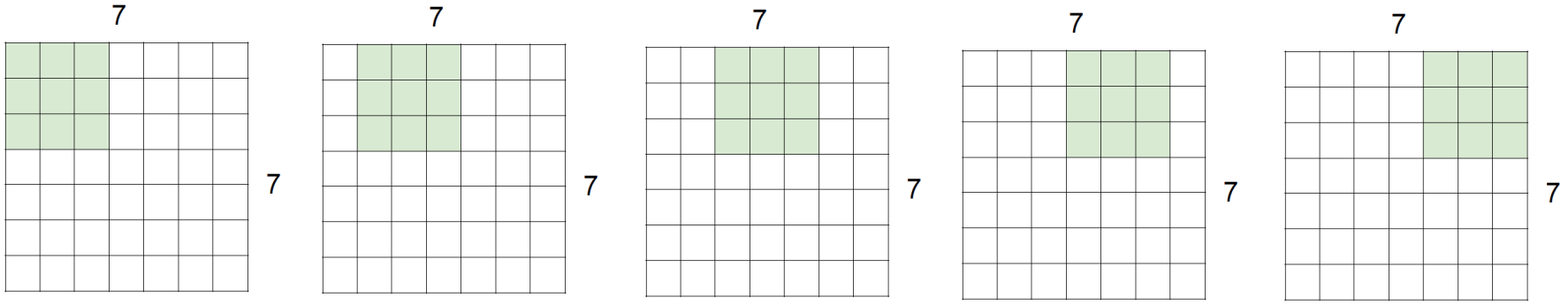

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:




We stack these up to get a "new image" of size 28x28x6!

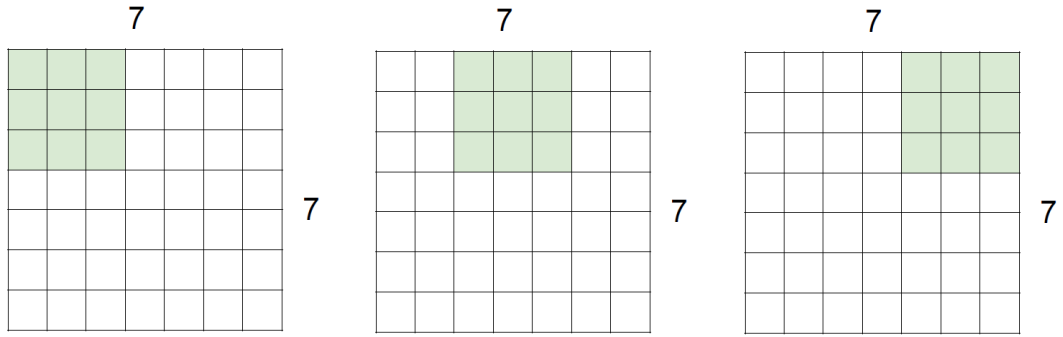
# Convolution Layer – Stride and Pad

**A closer look at spatial dimensions:**




7x7 input (spatially)
**3x3 filter** applied with **1 stride**

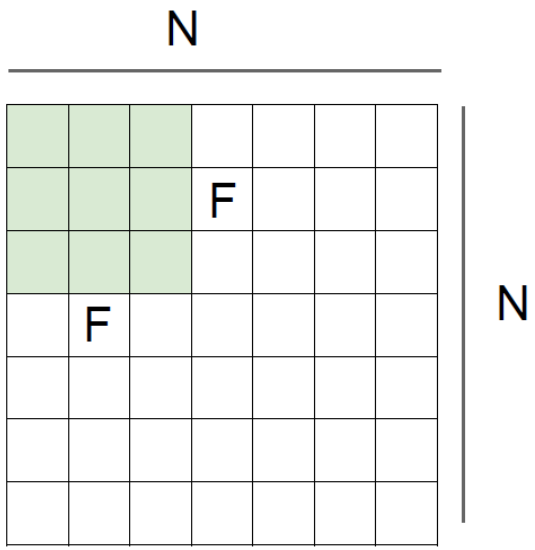
➔ **5x5 output**




7x7 input (spatially)
**3x3 filter** applied with **2 stride**

➔ **3x3 output**

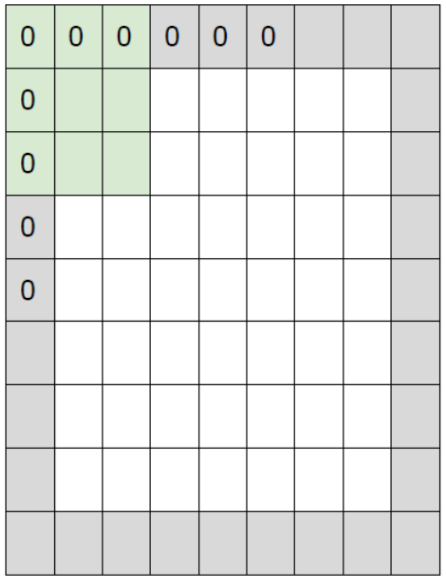
# Convolution Layer – Stride and Pad




Output size:
**(N - F) / stride + 1**

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

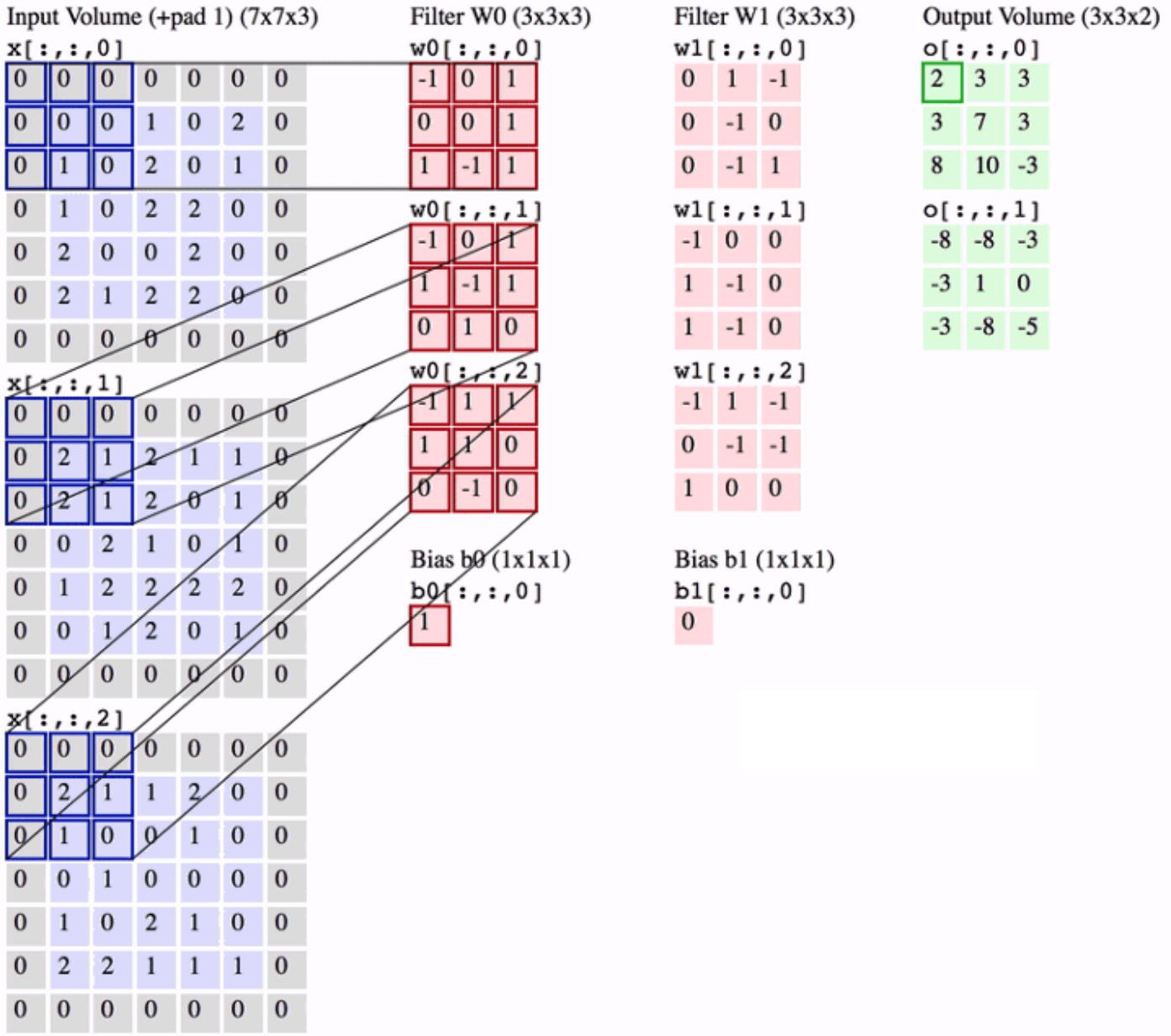
## In practice: Common to zero pad the border




e.g. input 7x7
**3x3** filter, applied with **stride 1**
**pad with 1 pixel** border => what is the output?

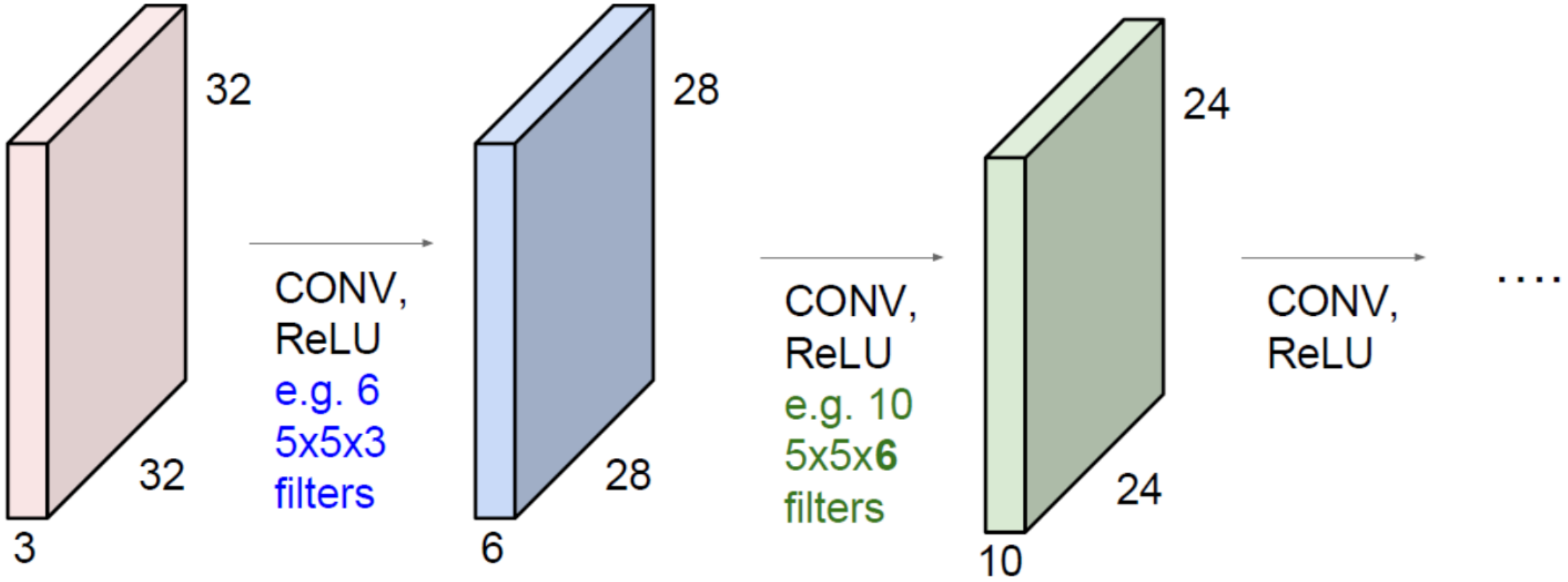
**7x7 output!**
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

# Convolution Layer – Stride and Pad



# Convolution Layer

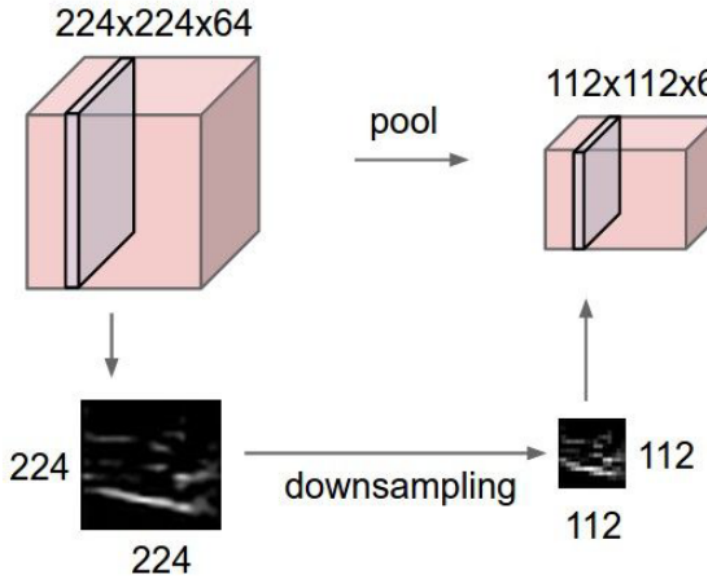
**ConvNet is a sequence of Convolution Layers, interspersed with activation functions**



# Pooling Layer

## Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:




MAX POOLING

Common settings:

F = 2, S = 2
F = 3, S = 2

Single depth slice

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

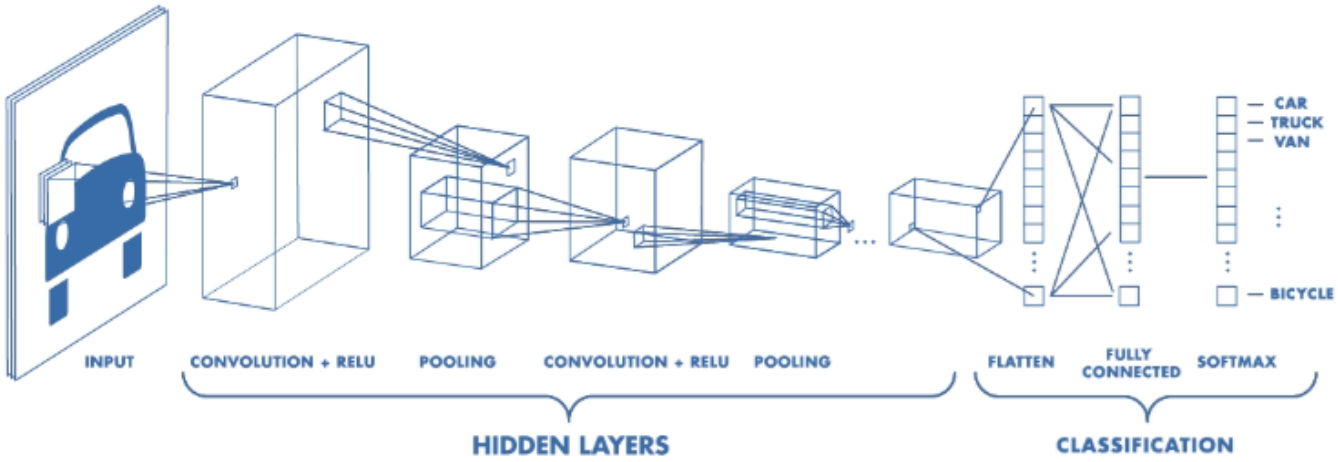
x

y

max pool with 2x2 filters and stride 2

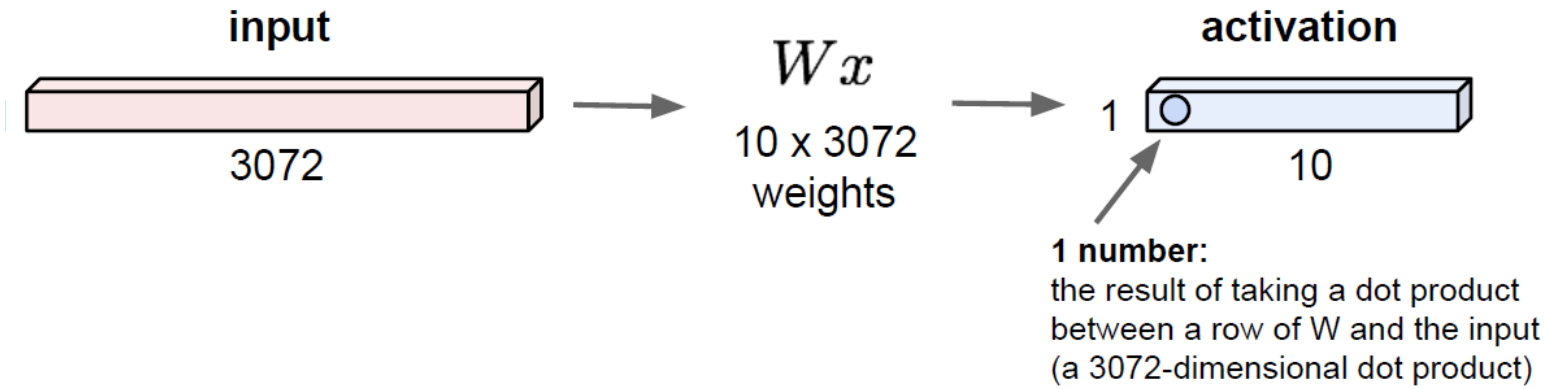
| | |
|---|---|
| 6 | 8 |
| 3 | 4 |

# Fully Connected Layer

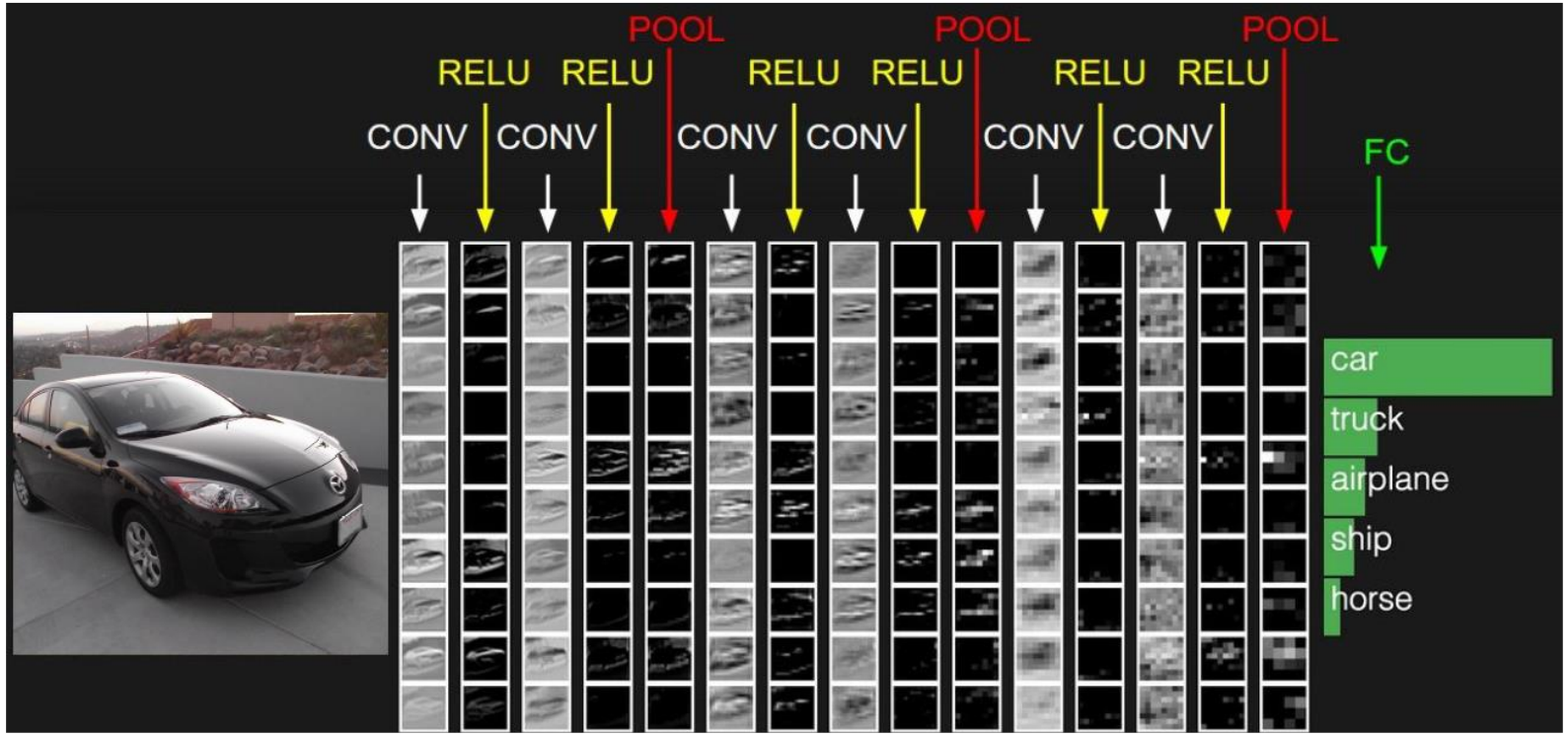



## Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



# CNN Summary




- CNNs stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like **[(CONV-RELU)*N-POOL]*M-(FC-RELU)*K,SOFTMAX** where N is usually up to ~5, M is large, 0 <= K <= 2.
- but recent advances such as ResNet/GoogLeNet challenge this paradigm

# Implementation Flow in TensorFlow

TesorFlow 1.0

**Graph mode**


TesorFlow 2.0

**Eager execution**

**Colab**

# Implementation Flow in TensorFlow

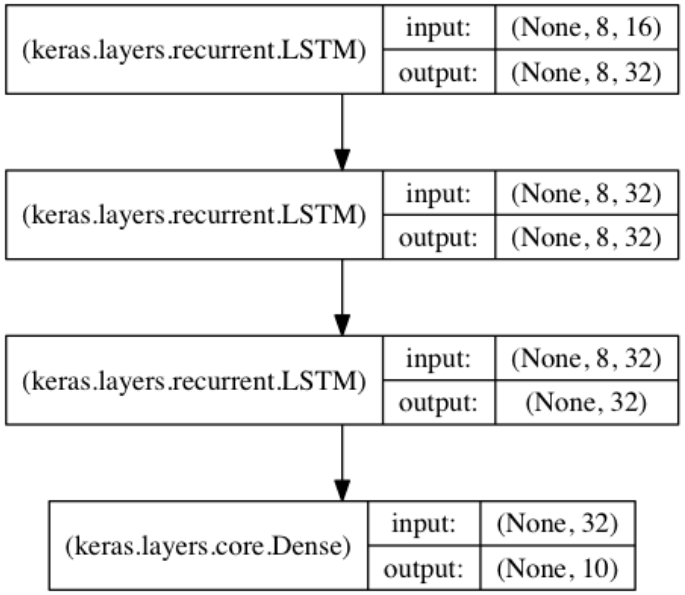
## NN Implementation Flow in TensorFlow

1. Set hyper parameters –learning rate, training epochs, batch size, etc.
2. Data Augmentation
3. Make a data pipelining
4. Build a neural network model –sequential or functional
5. Define a loss function –cross entropy or MSE
6. Calculate a gradient –use fit or tf.GradientTape
7. Select an optimizer –Adam optimizer etc.
8. Define a metric for model's performance –accuracy, RMSE, MAPE, etc.
9. (optional) Make a checkpoint for saving
10. Train and Validate a neural network model

# Implementation Flow in TensorFlow

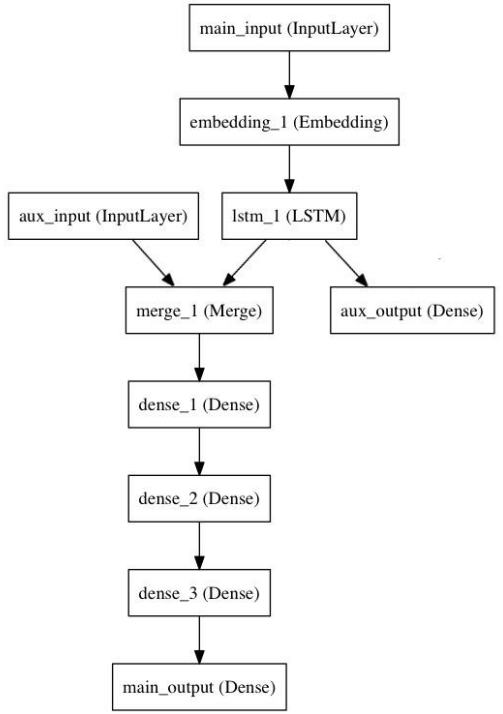
## Building a NN model

### 1. Sequential API




```
model = keras.Sequential()
model.add(keras.layers.Conv2D(filters=32, kernel_size=3,
     padding='SAME', input_shape=(28, 28, 1)))
model.add(keras.layers.MaxPool2D(padding='SAME'))
```

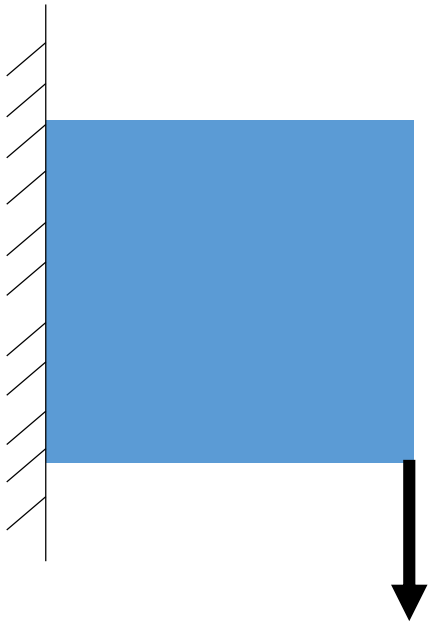
### 2. Functional API



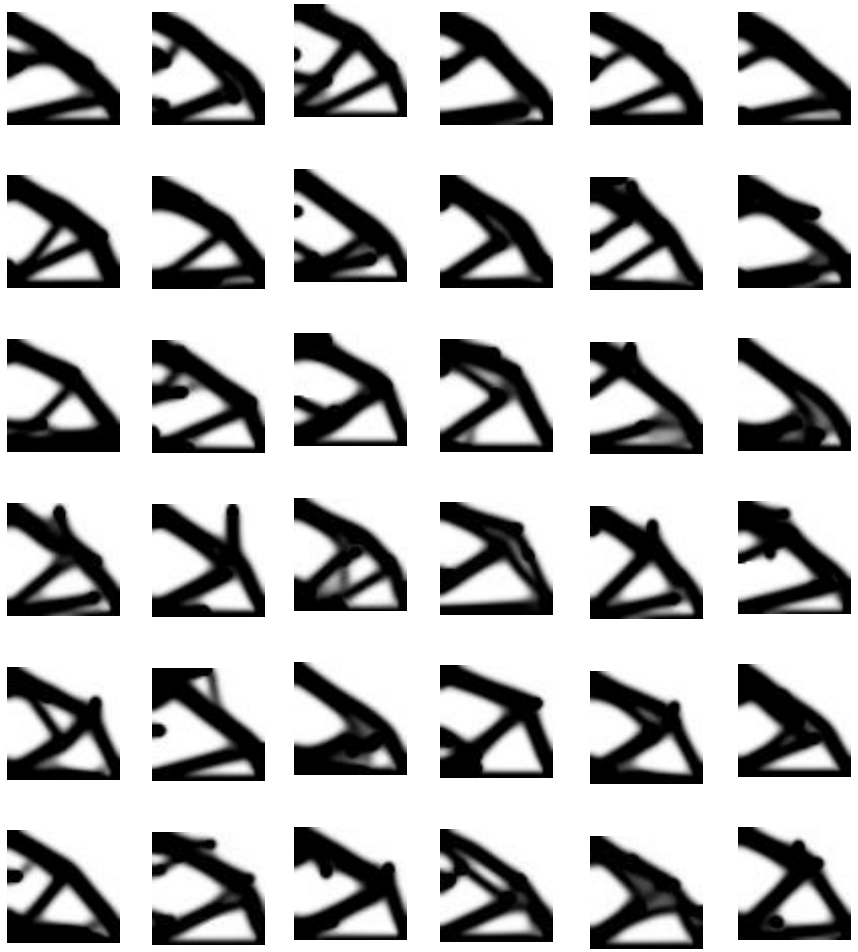

```
inputs = keras.Input(shape=(28, 28, 1))
conv1 = keras.layers.Conv2D(filters=32, kernel_size=3, padding='SAME',
       activation=tf.nn.relu)(inputs)
pool1 = keras.layers.MaxPool2D(padding='SAME')(conv1)
```

# Implementation - Structural Design Data


$y$= compliance



data_3000 / data_5000



# What Questions Do You Have?

**nwkang@sm.ac.kr**

**www.smartdesignlab.org**