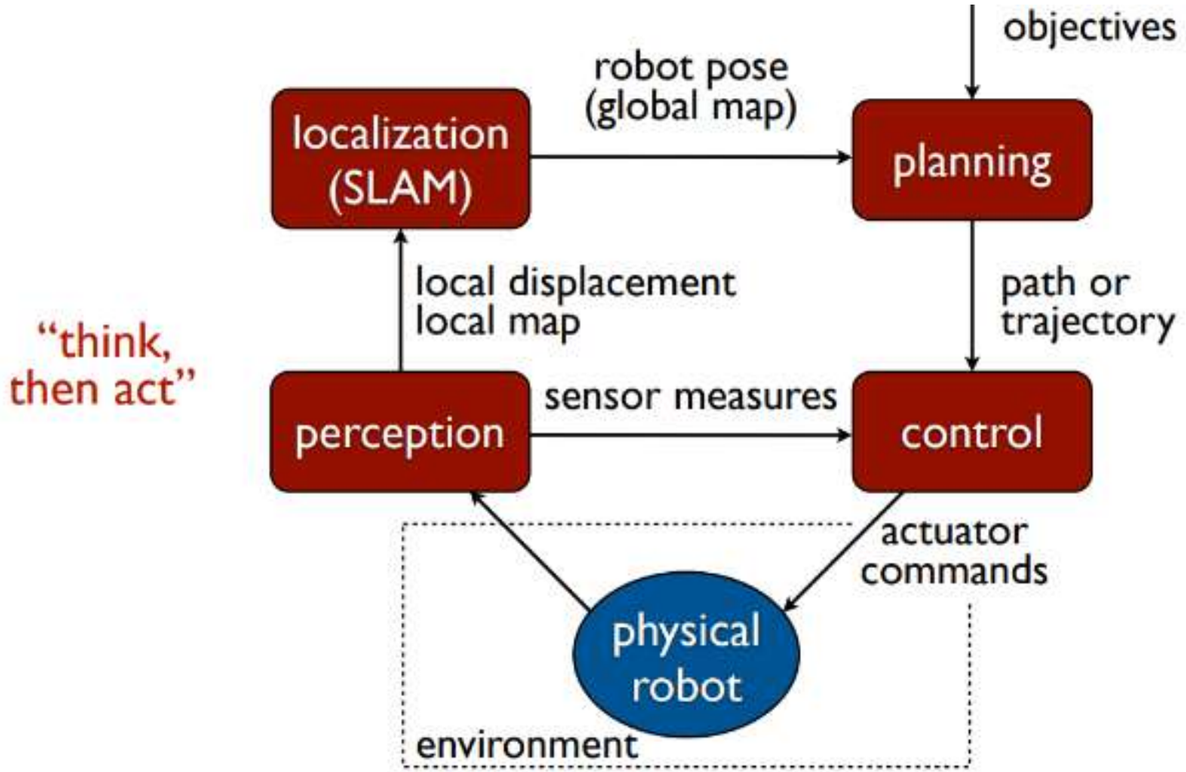
CNNs and MediaPipe

Outcomes

1. Understanding the big picture of the course
2. CNNs
3. MediaPipe Hands
4. ROS 2 Image Publisher (ImagePublisher node)
5. ROS 2 Image Subscriber (Hands node)

Bigger Picture from real-world example

skydio: https://www.youtube.com/watch?v=Gh5pAT1o2V8

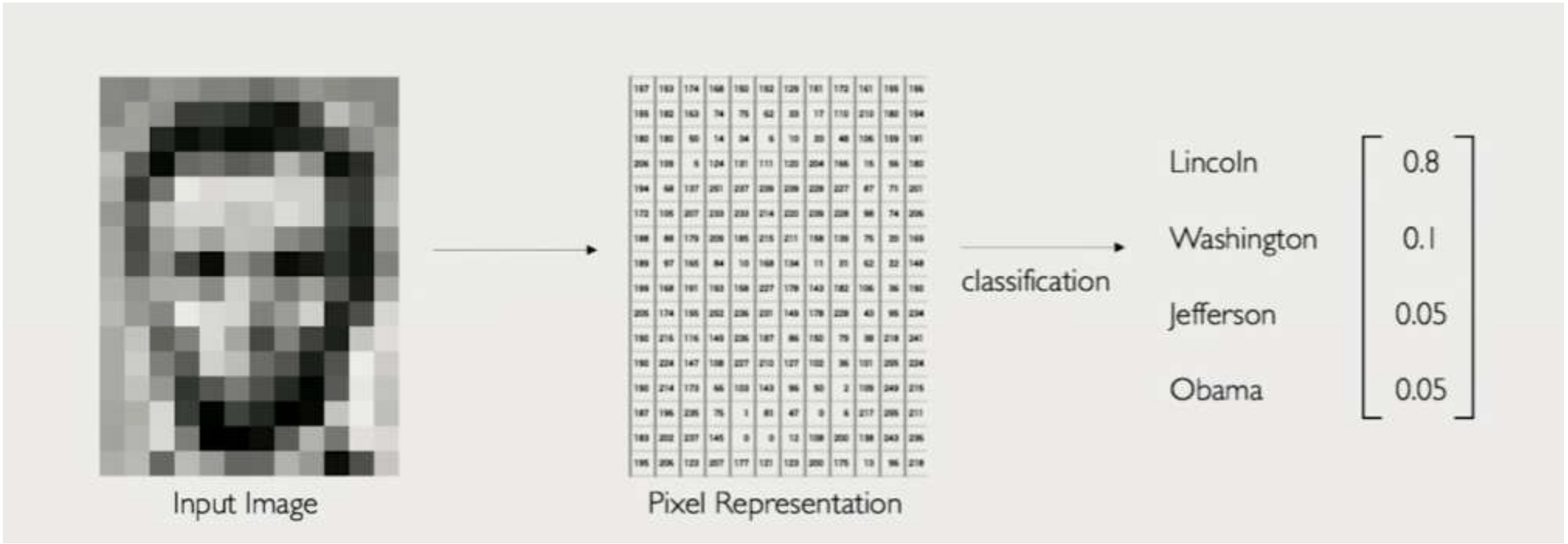


CNNs

- How to do feature engineering of visual inputs ?

CNNs

- What the computer sees as 3D numpy Array

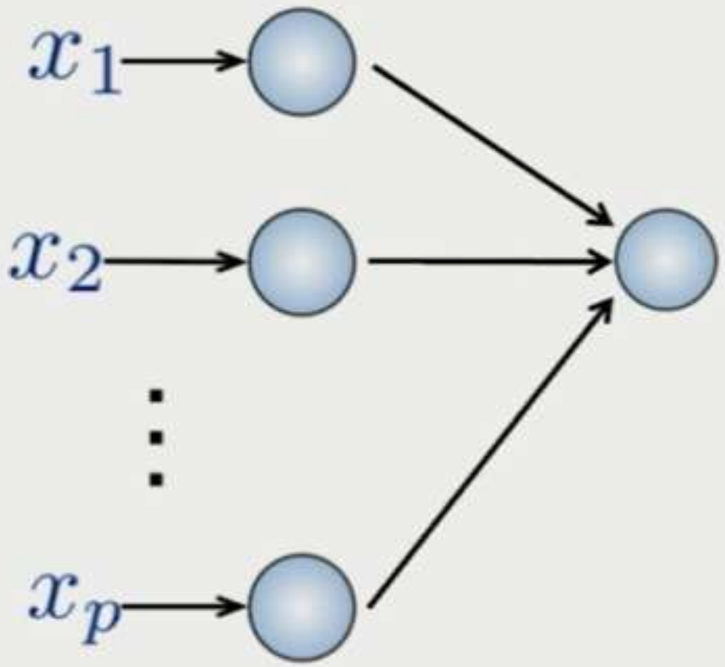


CNNs

- Using the concatenate vector will make us loss spatial information (understanding is this part of the image an edge/circle)

Input:

- 2D image
- Vector of pixel values

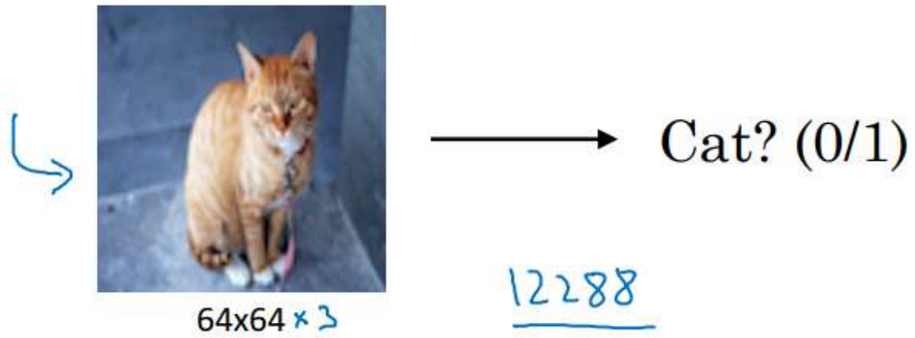


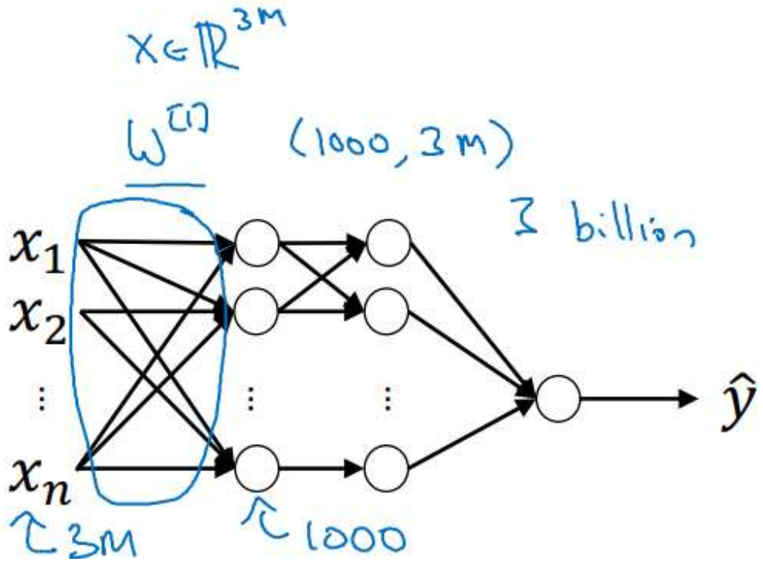
Fully Connected:

- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

CNNs

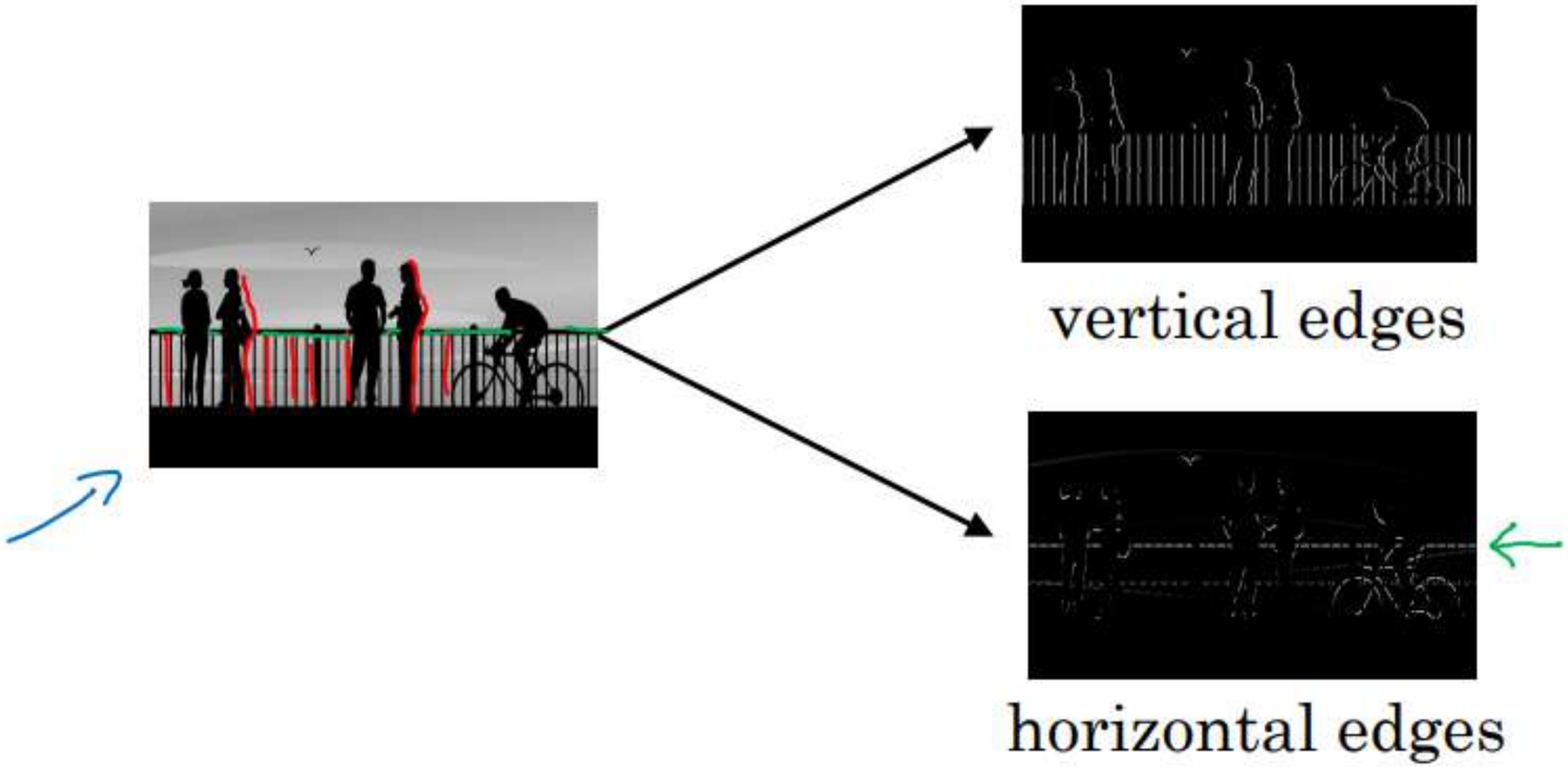
- Number of Parameters can explode while using the concatenate vector and dense layer.





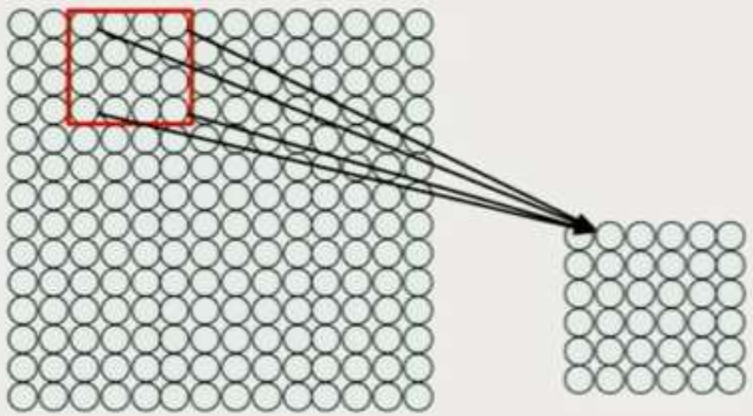
CNNs

-We want to detect patterns as horizontal, vertical, or circular images.



CNNs

- The Idea is to use a local filters at each part of the image.



- Filter of size 4x4 : 16 different weights
- Apply this same filter to 4x4 patches in input
- Shift by 2 pixels for next patch

This "patchy" operation is **convolution**

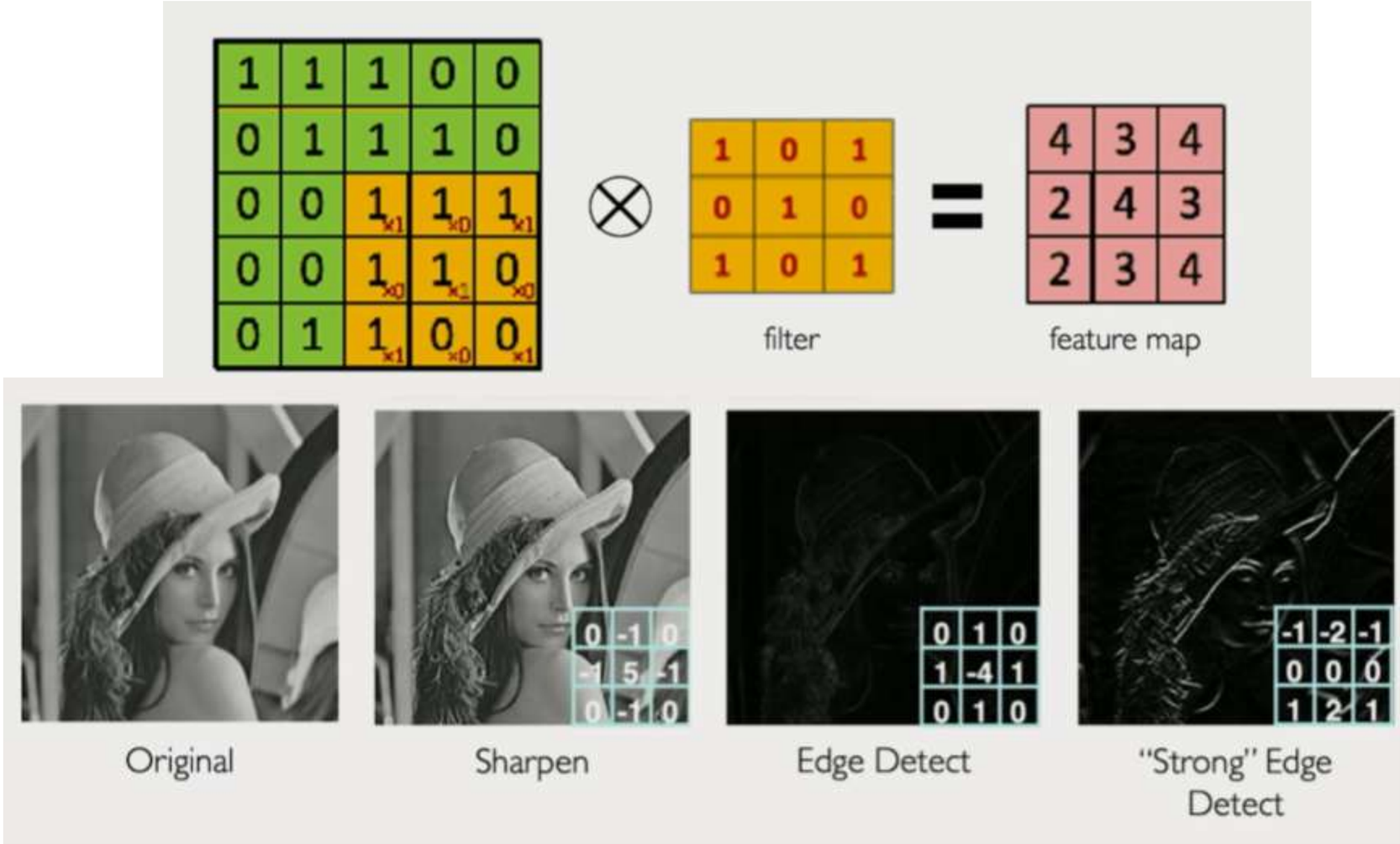
1) Apply a set of weights – a filter – to extract **local features**

2) Use **multiple filters** to extract different features

3) **Spatially share** parameters of each filter

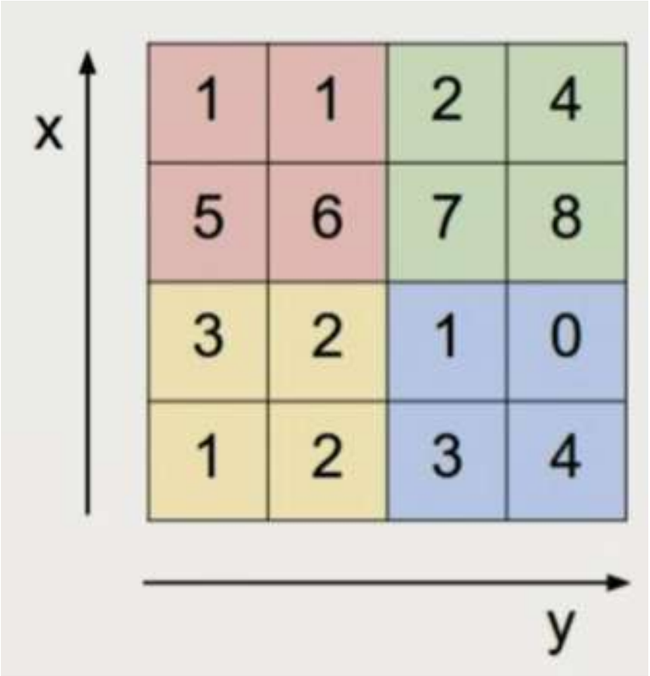
CNNs

- The Idea is to use a local filters at each part of the image.

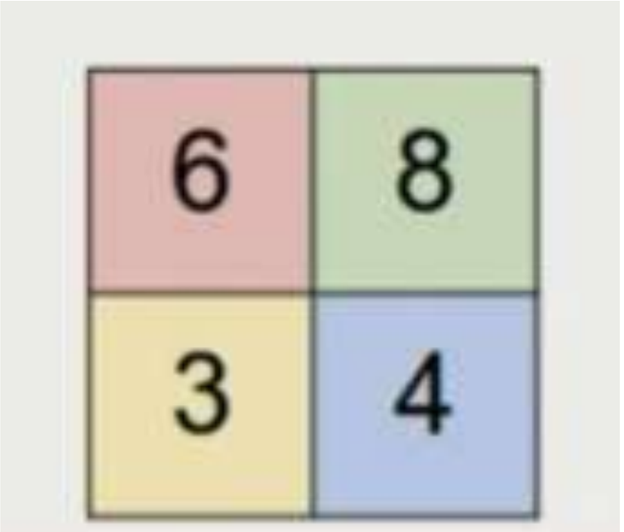


CNNs

- The Idea is to use a local filters at each part of the image.

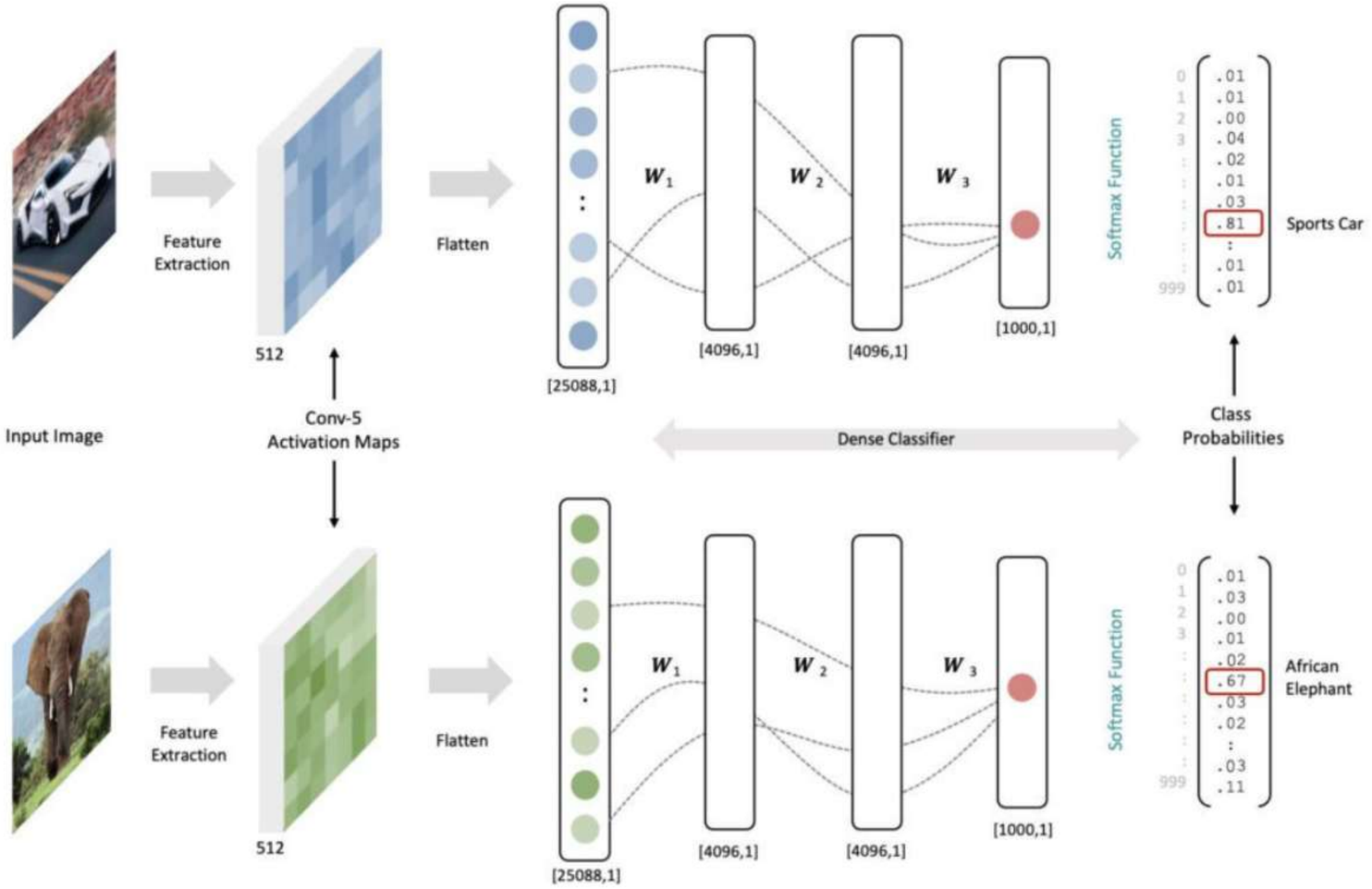


max pool with 2x2 filters
and stride 2



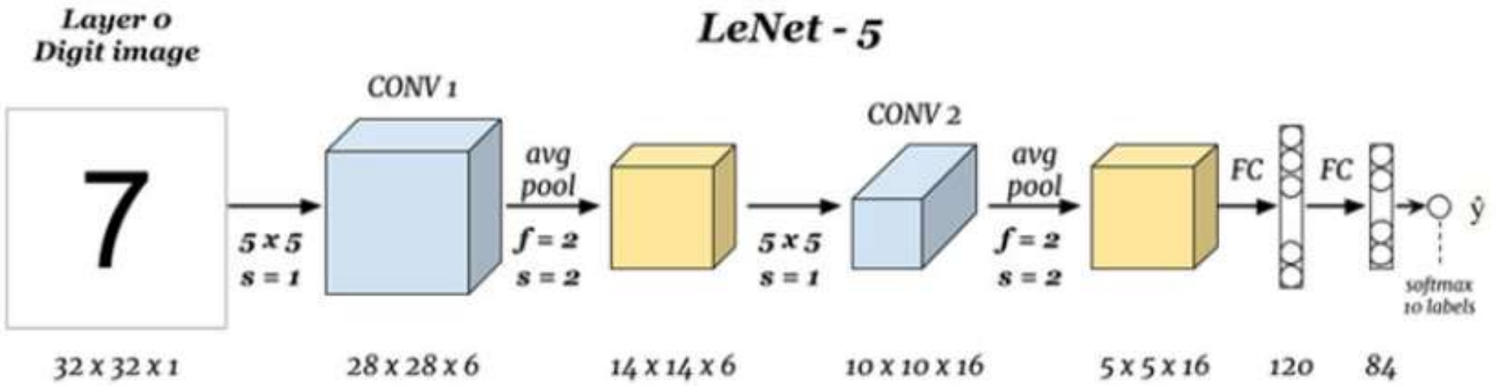
CNNs

The big Picture



CNNs

LetNet-5 Model. What is the model that we will use for assignment 2 ?.
Can we just use LetNet5 for hand tracking?



Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

CNNs

- Convolutional Neural Networks (CNNs) have at least one convolutional layer
- Used for image processing/classifiers
- Typical CNNs have the following layers
    - Convolutional layers
    - Pooling layers
    - Dense layers

Let's look at this example together:
https://www.tensorflow.org/tutorials/images/cnn

CNNs

- Famous CNNs

- AlexNet (2012)
- GoogLeNet / Inception (2014)
- VGG (2014)
- ResNet (2015)

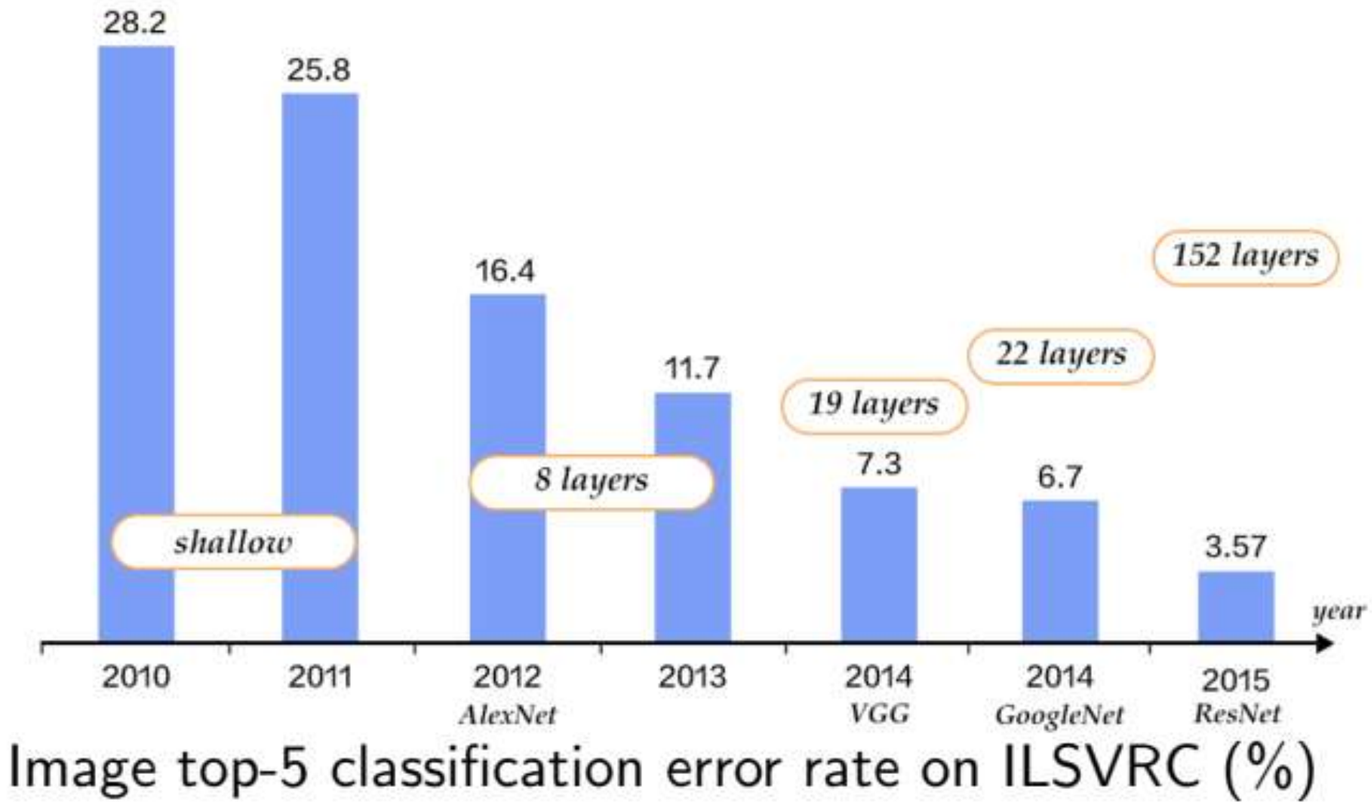
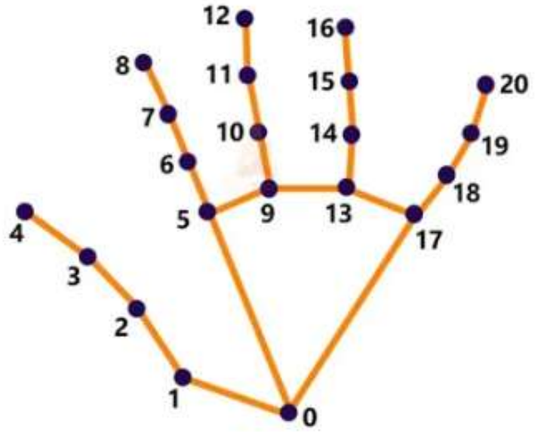


Image top-5 classification error rate on ILSVRC (%)

MediaPipe

- MediaPipe is a customizable machine learning solutions framework developed by Google. It is an open-source and cross-platform framework, and it is very lightweight. MediaPipe comes with some pre-trained ML solutions such as face detection, pose estimation, hand recognition, object detection, etc.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Hand Tracking Pretrained model

.

Searching for MobileNetV3

Andrew Howard[1] **Mark Sandler**[1] **Grace Chu**[1] **Liang-Chieh Chen**[1] **Bo Chen**[1] **Mingxing Tan**[2]
Weijun Wang[1] **Yukun Zhu**[1] **Ruoming Pang**[2] **Vijay Vasudevan**[2] **Quoc V. Le**[2] **Hartwig Adam**[1]
[1]Google AI, [2]Google Brain
{howarda, sandler, cxy, lcchen, bochen, tanmingxing, weijunw, yukun, rpang, vrv, qvl, hadam}@google.com

Abstract

We present the next generation of MobileNets based on a combination of complementary search techniques as well as a novel architecture design. MobileNetV3 is tuned to mobile phone CPUs through a combination of hardware-aware network architecture search (NAS) complemented by the NetAdapt algorithm and then subsequently improved through novel architecture advances. This paper starts the exploration of how automated search algorithms and network design can work together to harness complementary approaches improving the overall state of the art. Through this process we create two new MobileNet models for release: MobileNetV3-Large and MobileNetV3-Small which

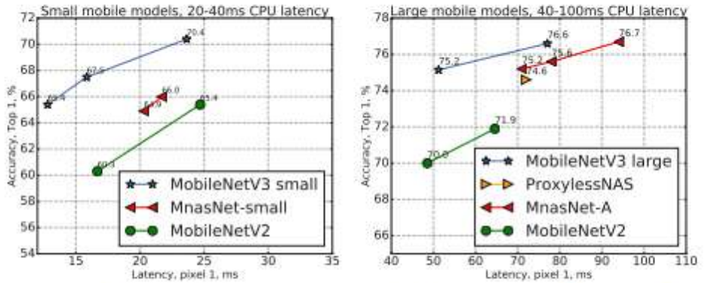


Figure 1. The trade-off between Pixel 1 latency and top-1 ImageNet accuracy. All models use the input resolution 224. V3 large and V3 small use multipliers 0.75, 1 and 1.25 to show optimal frontier. All latencies were measured on a single large core of the same device using TFLite[1]. MobileNetV3-Small and Large are our proposed next-generation mobile models.

Getting the state-of-the-art models

- General AI:
  1) NeurIPS = Conference on Neural Information processing systems.
  2) ICML= International Conference on Machine Learning.
  3) ICLR = International Conference on Learning Representations.

There are other flagship conference for each topic as computer vision, NL, etc

- Robotics and Control application:
  1) ICRA = IEEE International Conference on Robotics and Automation.
  2) IROS=IEEE/RSJ International Conference on Intelligent Robots and Systems
  3) CDC=*IEEE Conference on Decision and Control.*
  4) ACC=American Control Conference.

ROS2 Workspace (Assignment 2)

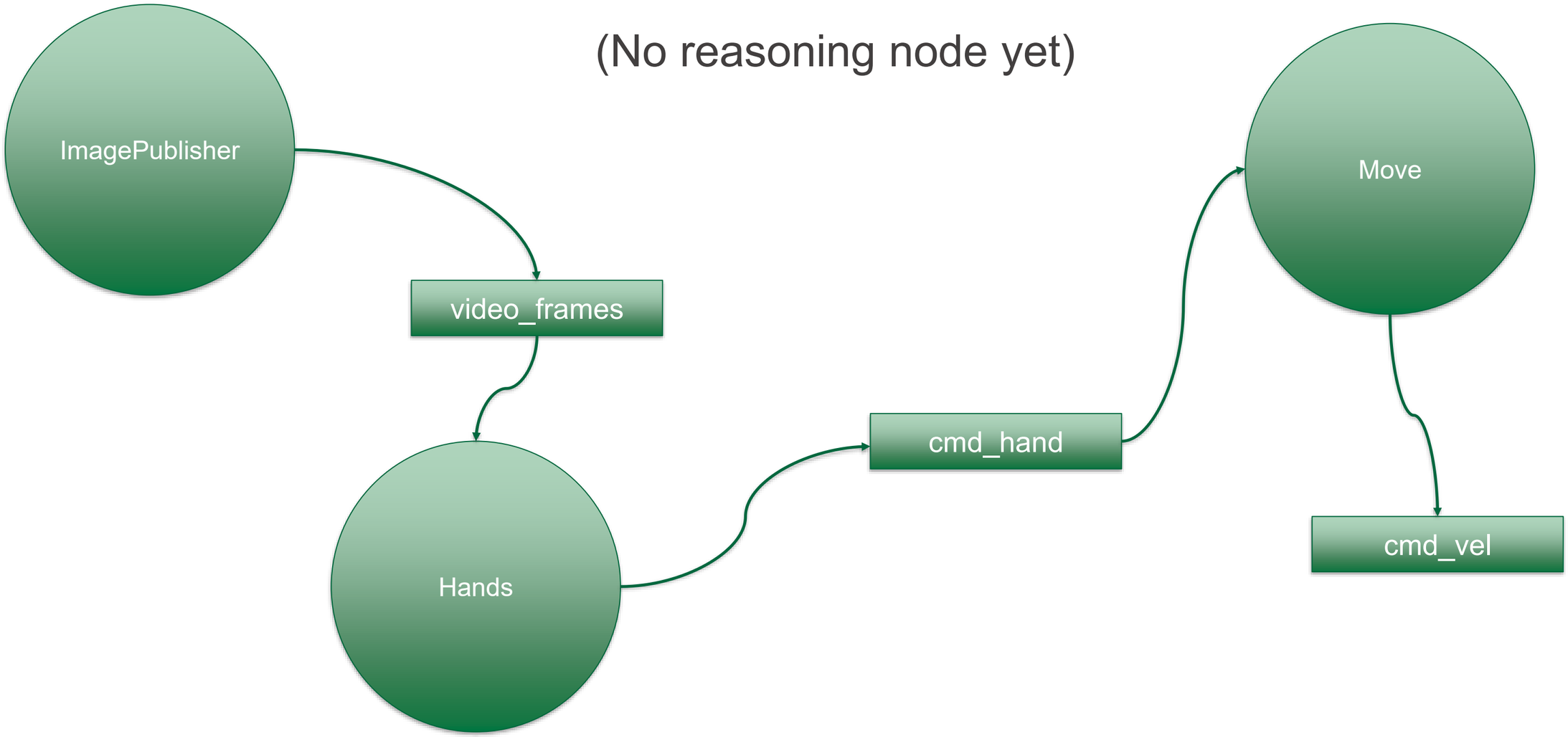
- aisd_msgs
  - Speak.srv
    - String words
    - String response
  - Hand.msg
    - float64 xpinky
    - float64 xindex
- aisd_vision
  - ImagePublisher node
    - Publisher Image messages on video_frames topic
  - Hands node
    - Subscriber to Image messages on video_frames topic
    - Publisher to Hand messages on cmd_hand topic
- aisd_motion
  - Move node
    - Subscriber to Hand messages on cmd_hand topic
    - Publisher Twist messages on cmd_vel topic

Parts needed for Assignment 2

- Create ROS 2 workspace (lab5)
- ROS 2 aisd_msgs (given in assignment 2, coming)
- Create ROS 2 aisd_vision and aisd_motion packages
- Create ROS 2 Nodes (python modules) in packages
- Run **rosdep** to install package dependencies
- Run **colcon build** to build packages ready to run
- Spin up Nodes with command line and test them out

aisd_vision and aisd_motion

(No reasoning node yet)



MediaPipe

- https://google.github.io/mediapipe/
    - https://developers.google.com/mediapipe/solutions/vision/hand_landmarker#get_started
    - Python Solution API has code we need, along with the following code:

```
INDEX_FINGER_TIP = 8

PINKY_FINGER_TIP = 20

results = myhands.process(image)

      if results.multi_hand_landmarks:

          #publish the hand position in terms of index finger and pinky

          msg = Hand()

          msg.xpinky = results.multi_hand_landmarks[0].landmark[PINKY_FINGER_TIP].x

          msg.xindex = results.multi_hand_landmarks[0].landmark[INDEX_FINGER_TIP].x

          if self.hand_publisher.get_subscription_count() > 0:

              self.hand_publisher.publish(msg)

          else:

              self.get_logger().info('waiting for subcriber')
```

rosdep and Dependencies

https://docs.ros.org/en/humble/Tutorials/Intermediate/Rosdep.html

- Dependencies are listed in package.xml
- If dependency is not handled by package.xml
    - Can install it system-wide, or
    - Can install it in a python virtual environment and activate environment before using package

aisd_vision package.xml depend:

```
<depend>rclpy</depend>

<depend>image_transport</depend>

<depend>cv_bridge</depend>

<depend>sensor_msgs</depend>

<depend>std_msgs</depend>

<depend>python3-mediapipe-pip</depend>
```

aisd_vision setup.py entry_points

```
entry_points={
        'console_scripts': [
            'image_publisher = aisd_vision.image_publisher:main',
            'hands = aisd_vision.hands:main',
        ],
    },
```

aisd_motion setup.py entry_points

```
entry_points={
        'console_scripts': [
            'move = aisd_motion.move:main',
        ],
    },
```