AI on NVIDIA Jetson Nano (Day 2)

Updated at 31 Aug 2021

Outline

- AI and Deep Learning
  - A brief overview of Deep Learning and how it reates to AI
- ImageNet
  - Classifying Images with ImageNet
- Convolutional Neural Networks (CNNs)
  - An introduction to the dominant class of ANN for computer vision tasks
- Face Mask Project

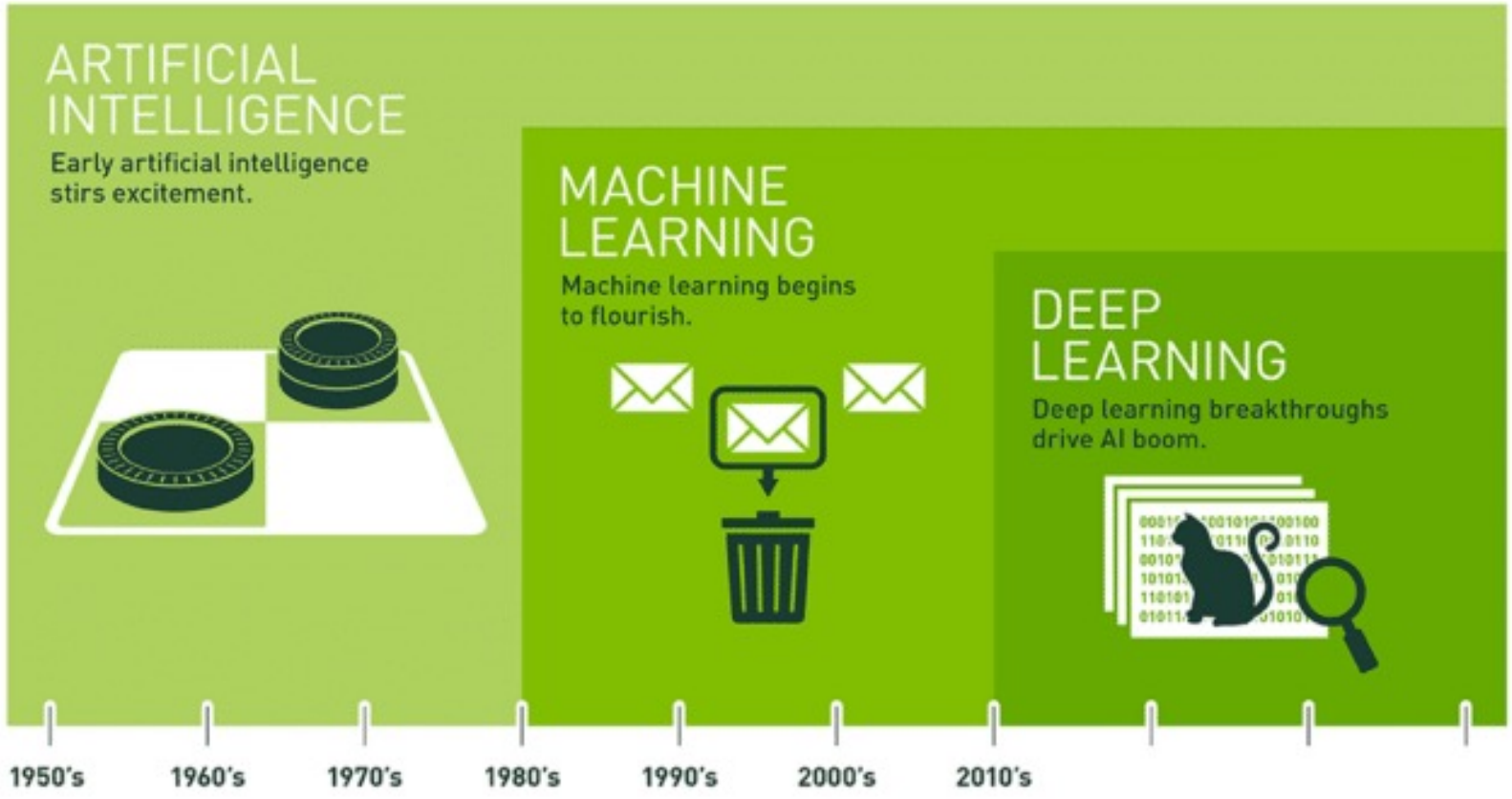
Prerequisites

- Jetson Nano Developer Kit
- Computer with Internet Access and SD card port
- microSD Memory Card (32GB UHS-I mininum)
- Compatible 5V 4A Power Supply with 2.1mm DC barrel connector
- 2-pin jumper
- USB cable (Micro-B to Type-A)
- Logitech C270 Webcam (Optional)

AI and Deep Learning

- As humans, we generalize what we see based on our experiences.
- In a similar way, we can use a branch of **AI** called **Machine Learning** to generalize and classify images based on experience in the form of lots of example data.
- In particular, we will use **Deep Neural Network** or **Deep Learning** to recognize relevant patterns in an image dataset, and ultimately match new images to correct answers.

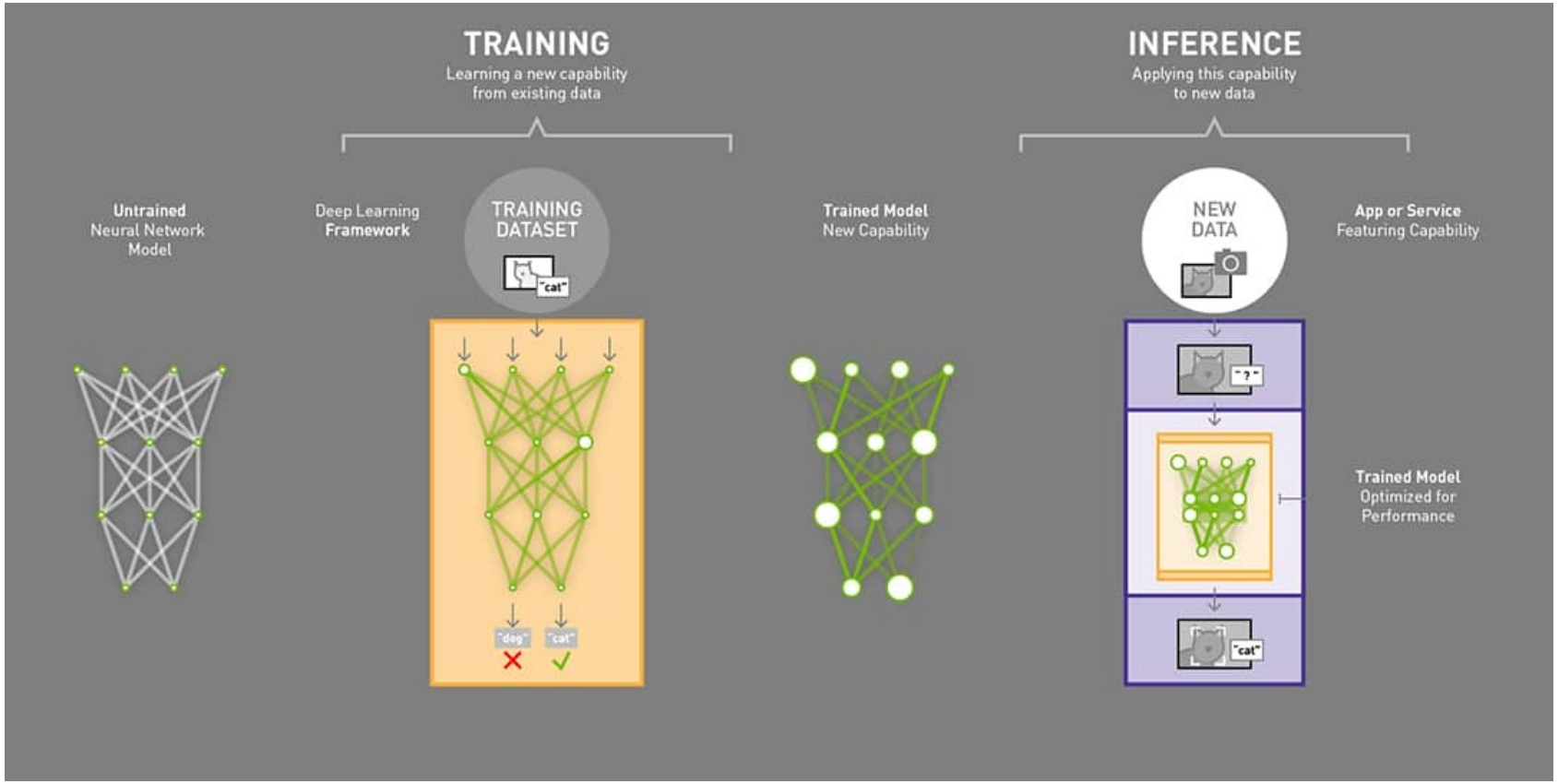
What's the Difference Between AI, ML, and DL?



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.

Deep Learning Models

- A Deep Learning model consists of a neural network with internal parameters, or weight, configured to map inputs to outputs.



ImageNet Data

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
  - ImageNet is an image database organized according to the WordNet hierarchy in which each node of the hierarchy is depicted by hundreds and thousands of images
  - The project has been intrumental in advancing computer vision and deep learning research
- This dataset spans 1,000 object classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images.
- Kaggle: https://www.kaggle.com/c/imagenet-object-localization-challenge/overview

Paper (2005): https://www.image-net.org/static_files/papers/imagenet_cvpr09.pdf

Using the ImageNet on Jetson Nano

- Open the terminal and located in the “build” directory with the following command: (compiling the project)

```
$ cd jetson-inference/build
$ make
$ sudo make install
$ sudo ldconfig
```

Using the ImageNet on Jetson Nano (cont'd)

- Verifying PyTorch

```
$ python3
$ >>> import torch
$ >>> print(torch.__version__)
$ >>> print('CUDA available: ' + str(torch.cuda.is_available()))
$ >>> a = torch.cuda.FloatTensor(2).zero_()
$ >>> print('Tensor a = ' + str(a))
$ >>> b = torch.randn(2).cuda()
$ >>> print('Tensor b = ' + str(b))
$ >>> c = a + b
$ >>> print('Tensor c = ' + str(c))
```

Using the ImageNet on Jetson Nano (cont'd)

- Open the terminal and create project with the following command:

```
$ mkdir Day1
$ cd Day1/ && code .
```

- Write the Python code: Live

Using the ImageNet on Jetson Nano (cont'd)

- Let's classify an example image with the my_imagenet.py

```
$ python my_imagenet.py images/your_picture.jpg images/output_your_picture.jpg
```

- Live camera recognition

```
$ python my_imagenet.py --camera=/dev/video0
```

Using the ImageNet on Jetson Nano (cont'd)

- Check the USB webcam

```
$ video-viewer /dev/video0
```

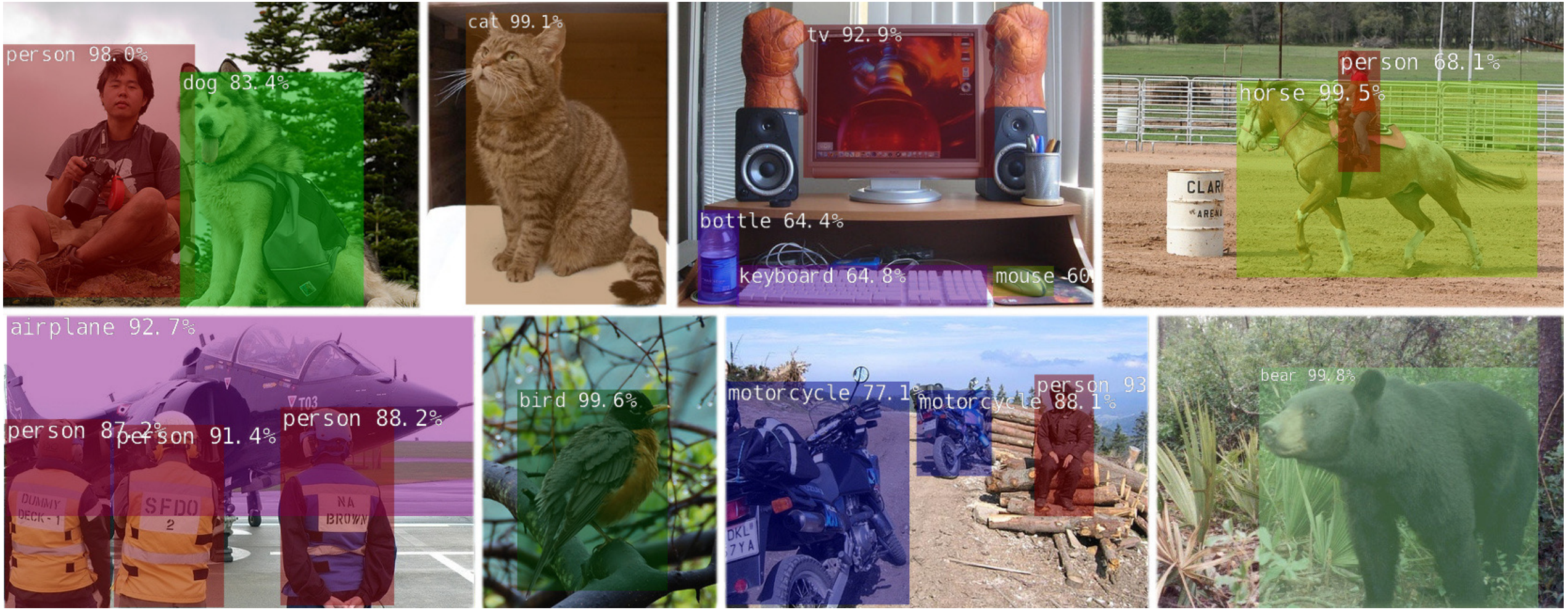
- Using different classification models

```
$ python my_imagenet.py --network=resnet-18 images/jellyfish.jpg \
images/test/output/output_jellyfish.jpg
```

Using the ImageNet on Jetson Nano (cont'd)

- Networks (CLI argument)
  - Alexnet: alexnet
  - GoogleNet: googlenet
  - GoogleNet-12: googlenet-12
  - ResNet-18: resnet-18
  - ResNet-50: resnet-50
  - ResNet-101: resnet-101
  - ResNet-152: resnet-152
  - VGG-16: vgg-16
  - VGG-19: vgg-19
  - Inception-v4: inception-v4

Locating Objects with DetectNet



Locating Objects with DetectNet (cont'd)

- Create new file is my_detectnet.py

```
$ touch my_detectnet.py
```

- Write the Python code: Live

Locating Objects with DetectNet (cont'd)

- Pre-trained Dtection Models Available
  - SSD-Mobilenet-v1: ssd-mobilenet-v1
  - SSD-Mobilenet-v2: ssd-mobilenet-v2
  - SSD-Inception-v2: ssd-inception-v2
  - DetectNet-COCO-Dog: coco-dog
  - DetectNet-COCO-Bottle: coco-bottle
  - DetectNet-COCO-Chair: coco-chair
  - DetectNet-COCO-Airplane: coco-airplane
  - ped-100: pednet
  - multiped-500: multiped
  - Facenet-120: facenet

Semantic Segmentation with SegNet

Semantic Segmentation with SegNet (cont'd)

- Create new file is my_segnet.py

```
$ touch my_segnet.py
```

- Write the Python code: Live

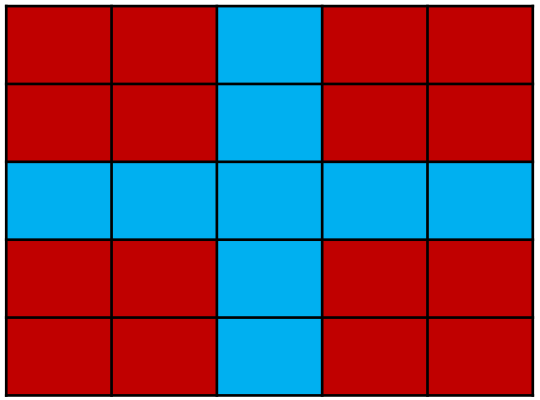
Semantic Segmentation with SegNet (cont'd)

- Pre-trained Dtection Models Available
  - Cityscapes (512 x 256): fcn-resnet18-cityscapes-512x256
  - Cityscapes (1024 x 512): fcn-resnet18-cityscapes-1024x512
  - Cityscapes (2048 x 1024): fcn-resnet18-cityscapes-2048x1024
  - DeepScene (576 x 320): fcn-resnet18-deepscene-576x320
  - DeepScene (864 x 480): fcn-resnet18-deepscene-864x480
  - Multi-Human (512 x 320): fcn-resnet18—mhp-512x320
  - Multi-Human (640 x 360): fcn-resnet18—mhp-640x360
  - Pascal VOC (320 x 320): fcn-resnet18-voc-320x320
  - Pascal VOC (512 x 320): fcn-resnet18-voc-512x320
  - SUN RGB-D (512 x 400): fcn-resnet18-sun-512x400
  - SUN RGB-D (640 x 512): fcn-resnet18-sun-640x512

Convolutional Neural Networks (CNNs)

- This is a cross picture?

Input feature map



5 x 5 pixels

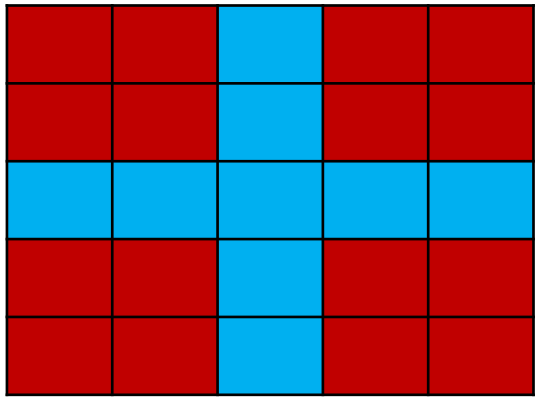
1	1	5	1	1
1	1	5	1	1
5	5	5	5	5
1	1	5	1	1
1	1	5	1	1

Red: 1, Blue: 5

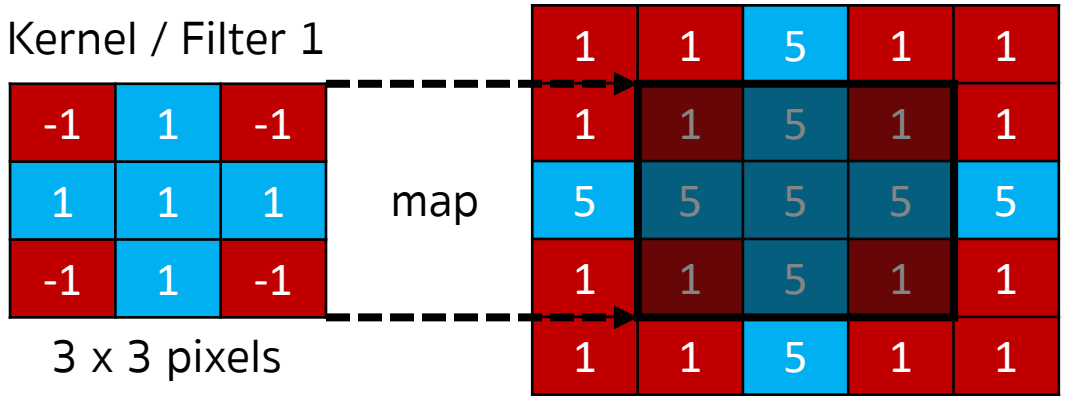
CNN / ConvNet (cont'd)

- Kernel / Filter

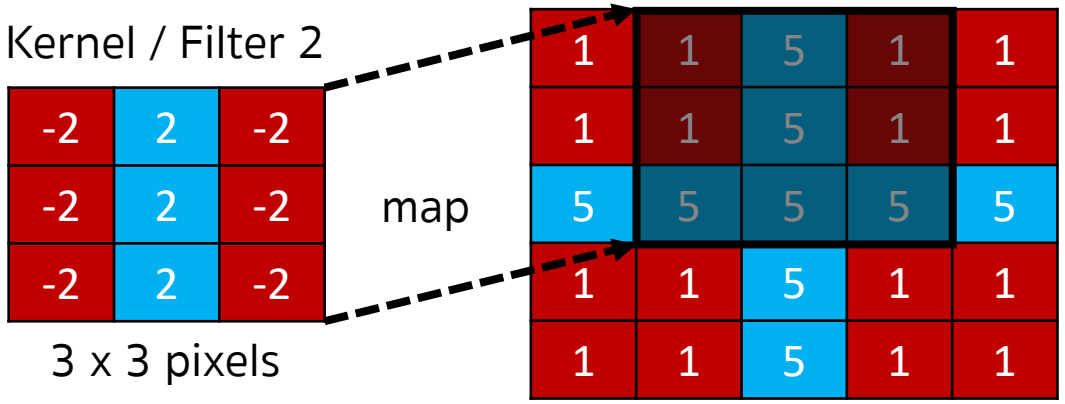
This is a cross picture?



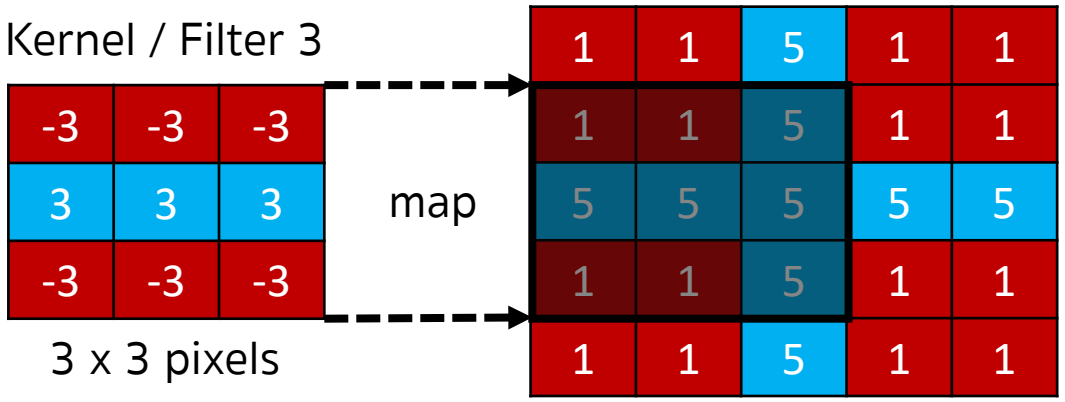
Kernel / Filter 1



Kernel / Filter 2

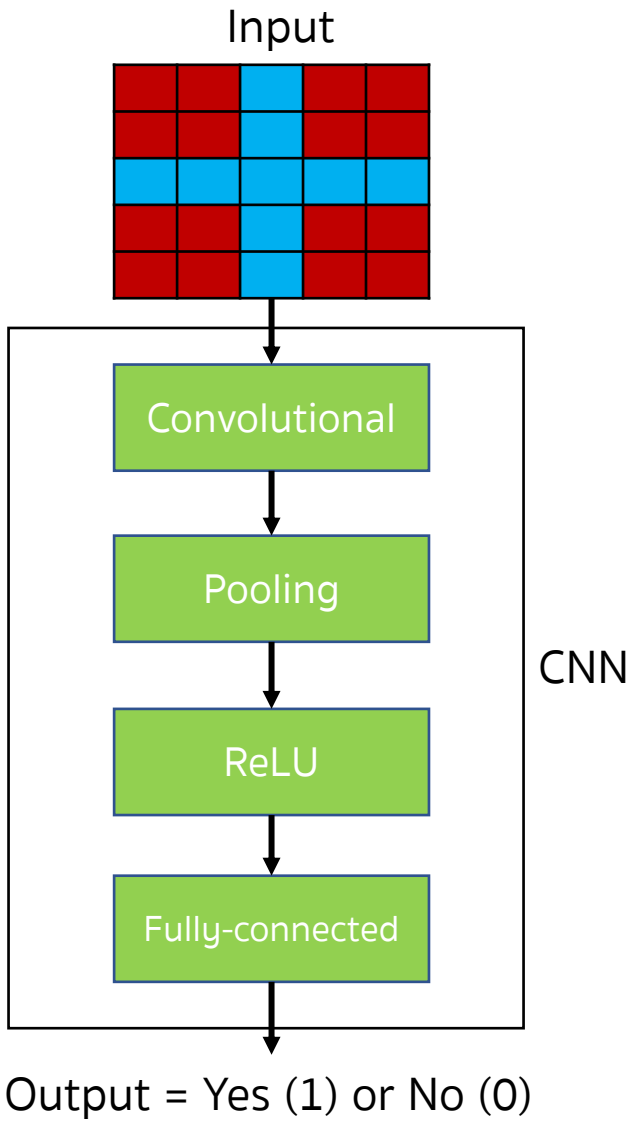


Kernel / Filter 3



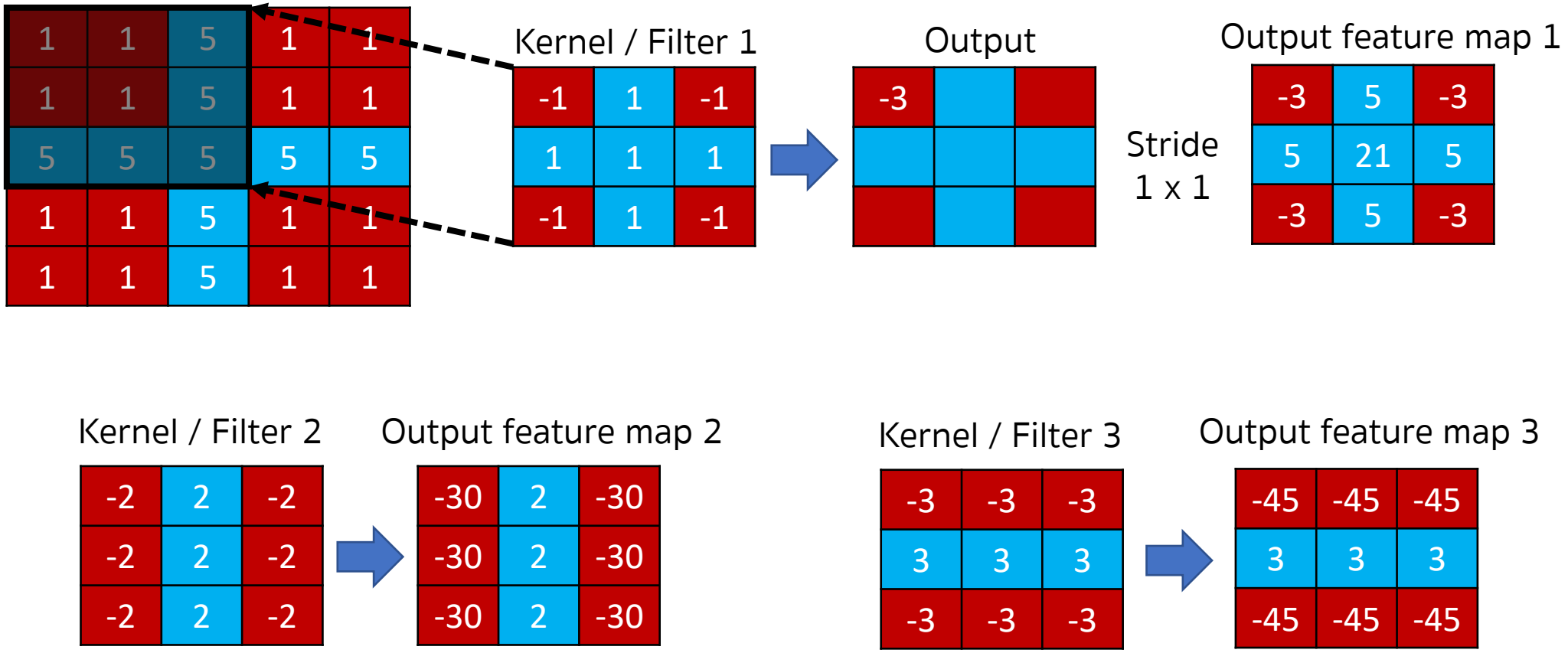
CNN / ConvNet (cont'd)

- CNN Layers
  - Convolutional layer
  - Pooling layer
  - ReLU layer
  - Fully-connected Layer



CNN / ConvNet (cont'd)

- Convolutional layer



CNN / ConvNet (cont'd)

- Padding

0	0	0	0	0	0	0
0	1	1	5	1	1	0
0	1	1	5	1	1	0
0	5	5	5	5	5	0
0	1	1	5	1	1	0
0	1	1	5	1	1	0
0	0	0	0	0	0	0

Kernel / Filter 1

-1	1	-1
1	1	1
-1	1	-1

2	2	10	2	2
2	-3	5	-3	2
10	5	21	5	10
2	-3	5	-3	2
2	2	10	2	2

5 x 5

Kernel / Filter 2

-2	2	-2
-2	2	-2
-2	2	-2

0	-20	12	-20	0
0	-30	2	-30	0
0	-30	2	-30	0
0	-30	2	-30	0
0	-20	12	-20	0

5 x 5

Kernel / Filter 3

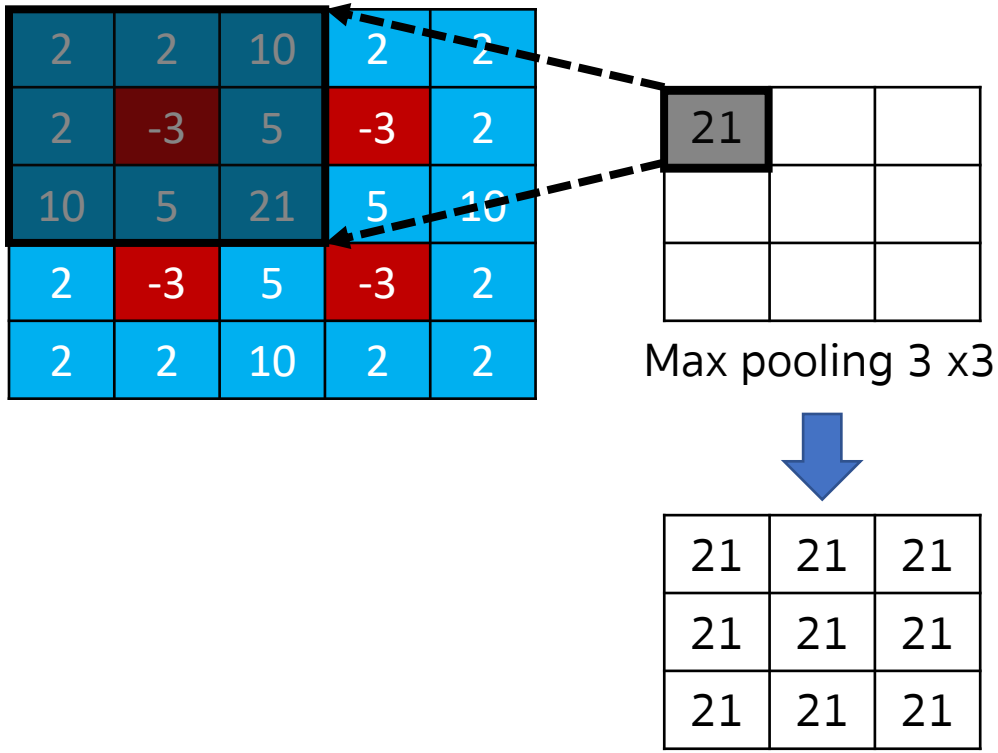
-3	-3	-3
3	3	3
-3	-3	-3

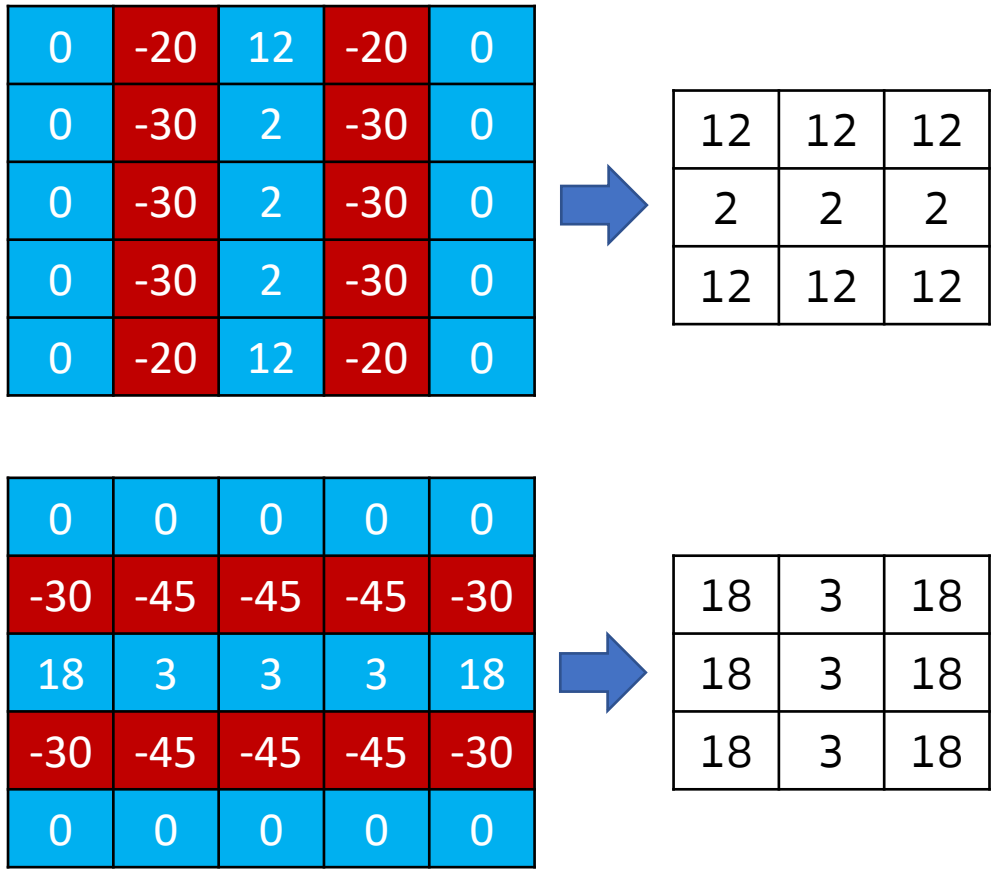
0	0	0	0	0
-30	-45	-45	-45	-30
18	3	3	3	18
-30	-45	-45	-45	-30
0	0	0	0	0

5 x 5

CNN / ConvNet (cont'd)

- Pooling Layer





CNN / ConvNet (cont'd)

- Rectified Linear Units (ReLU) layer
  - Activation function

ReLU (z) = 0 when z < 0
z when z >= 0

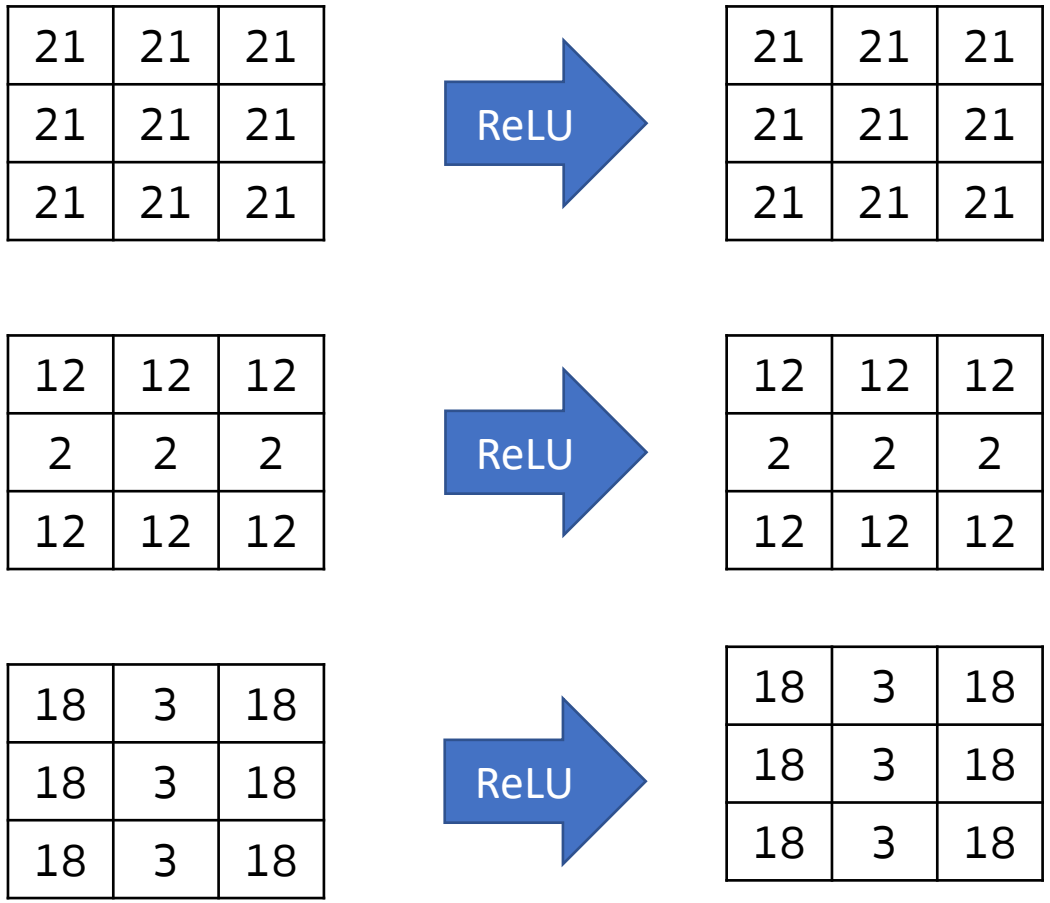
  - Example

Input function = 100 → max(0, 100) → Output = 100

Input function = -70 → max(0, -70) → Output = 0

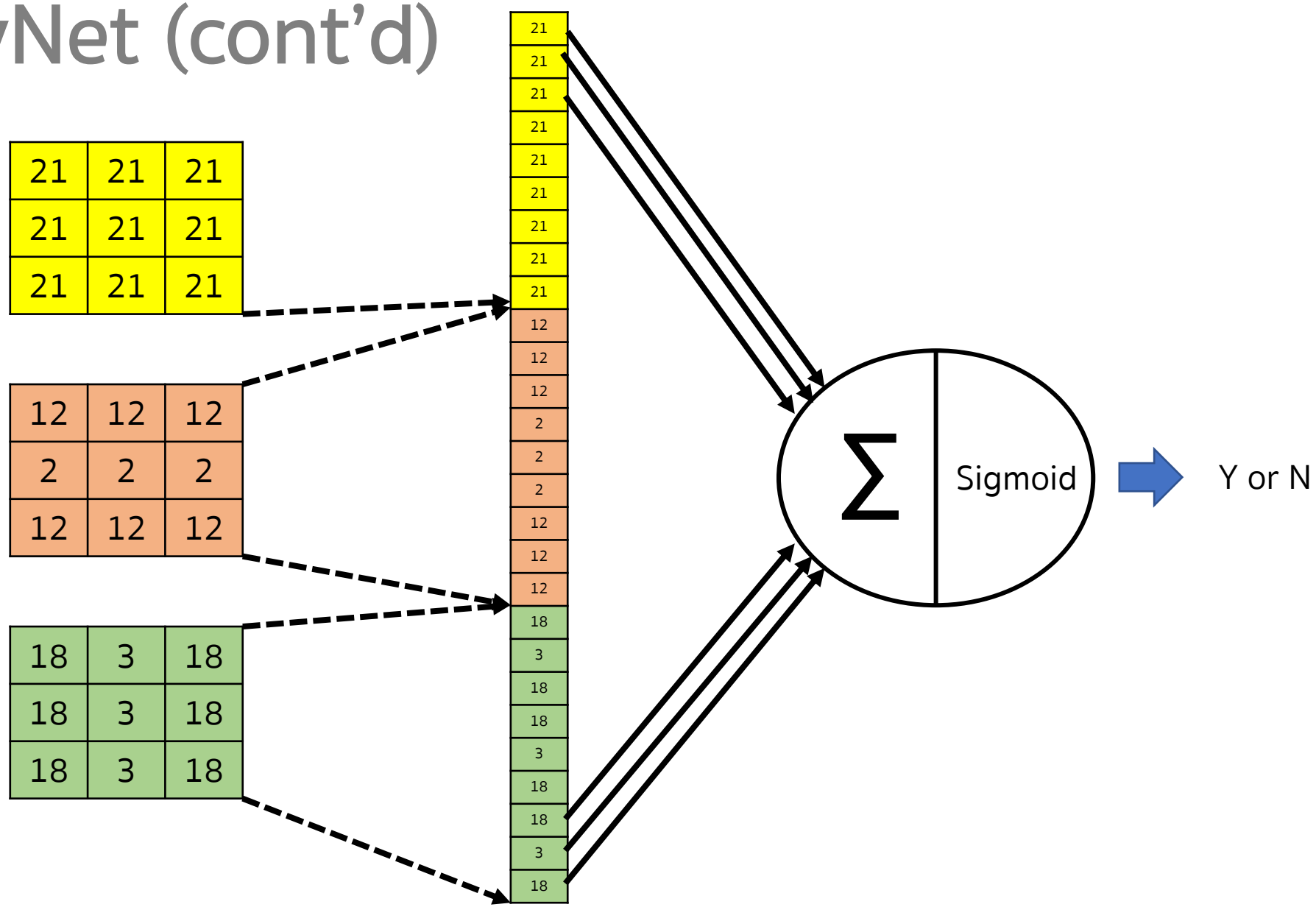
CNN / ConvNet (cont'd)

- ReLu layer



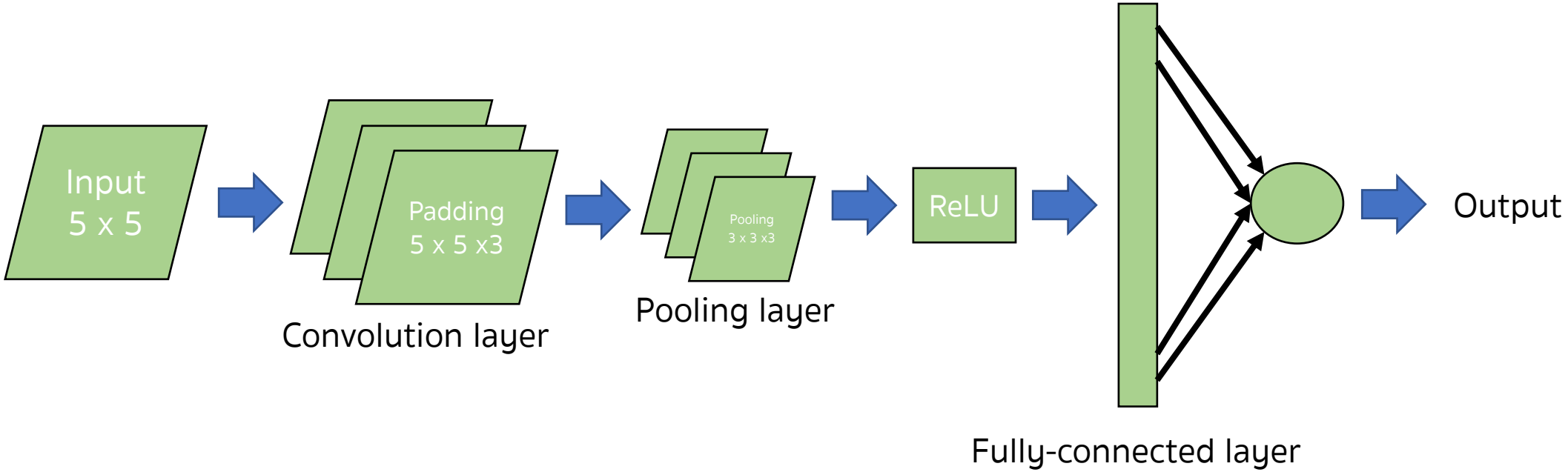
CNN / ConvNet (cont'd)

- Fully-connected layer



CNN / ConvNet (cont'd)

- CNN Overview



Face Mask Detection Project

- Write the python code: Live

References

- ImageNet Data
  - https://www.image-net.org/update-mar-11-2021.php
- What's Difference Between Artificial Intelligence, Machine Learning and Deep Learning?
  - https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/