KENTECH
Korea Institute of Energy Technology

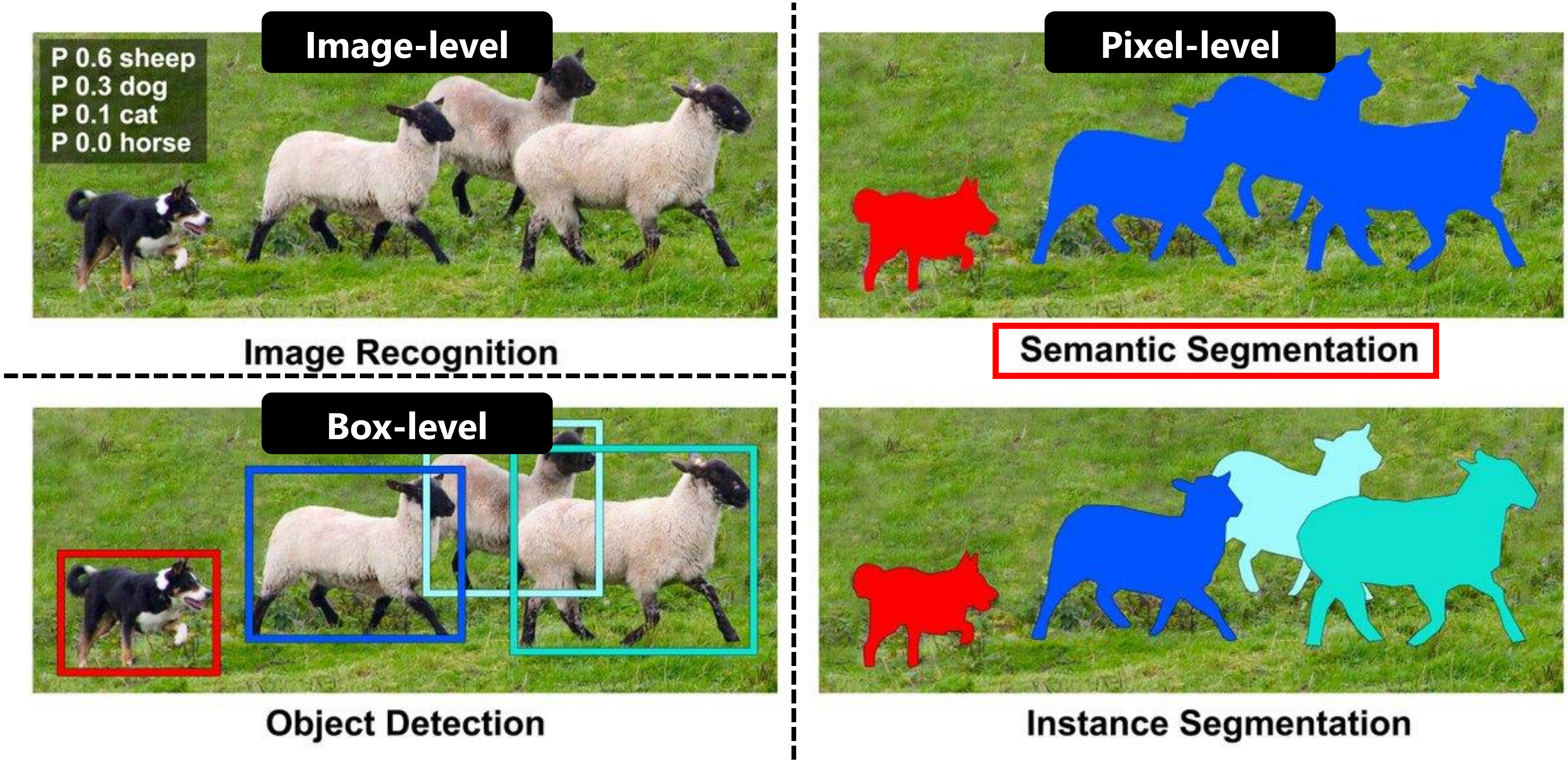
Visionary Course – Energy AI
Week 08

Apr. 26, 2022
Seokju Lee

KENTECH
Korea Institute of Energy Technology

Week 08b – Semantic Segmentation

Computer Vision Tasks



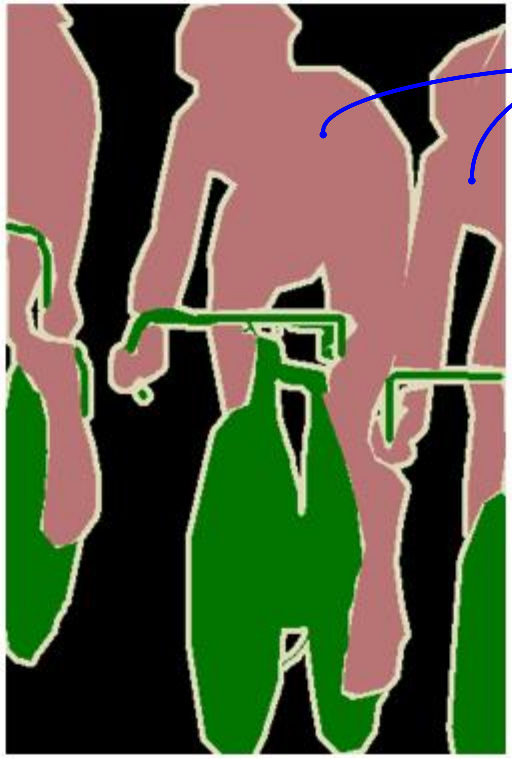
[1] Figure credits: John Wilson

Semantic Segmentation

: Task of assigning **a label** for **every pixel** of an image with a corresponding **class**

Dense prediction

predict



Different objects,
but same prediction
→ limitation

Person
Bicycle
Background

[1] Figures from Pascal VOC Challenge 2012

Semantic Segmentation: Input & Output

Groud Truth (GT) → supervised learning

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

Input

→ Input image resolution: $\boldsymbol{W} \times \boldsymbol{H} \times \mathbf{3}$

Semantic Labels

Note that this is a low-resolution prediction map for visual clarity.
In reality, the segmentation label resolution should match the original input's resolution ($\boldsymbol{W} \times \boldsymbol{H} \times \mathbf{3}$).

[1] Figure credits: Jeremy Jordan

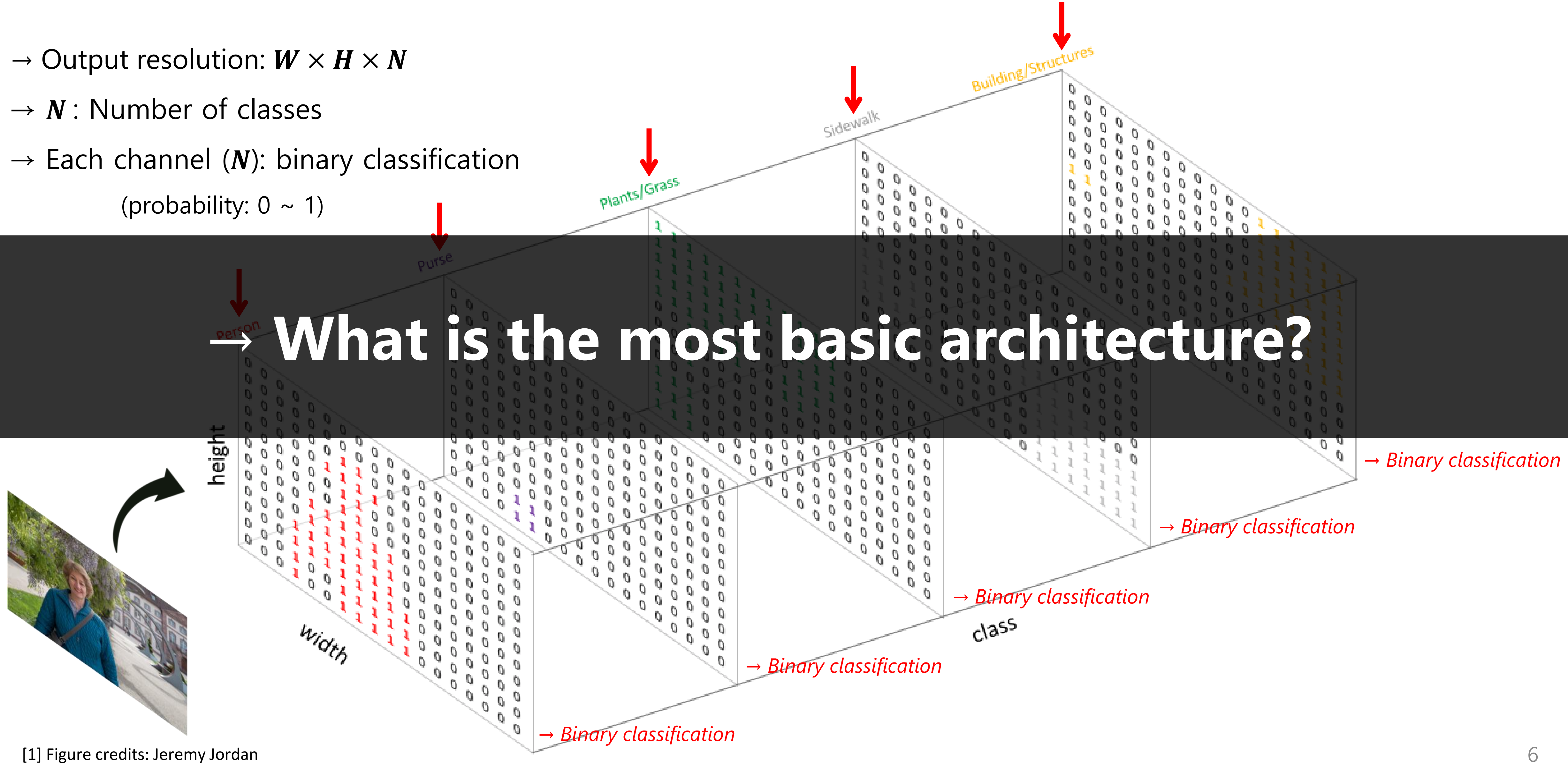
Semantic Segmentation: Input & Output

→ Output resolution: $\boldsymbol{W} \times \boldsymbol{H} \times \boldsymbol{N}$

→ $\boldsymbol{N}$: Number of classes

→ Each channel ($\boldsymbol{N}$): binary classification

(probability: 0 ~ 1)



→ **What is the most basic architecture?**

[1] Figure credits: Jeremy Jordan

Image Classification

This image is CC0 public domain

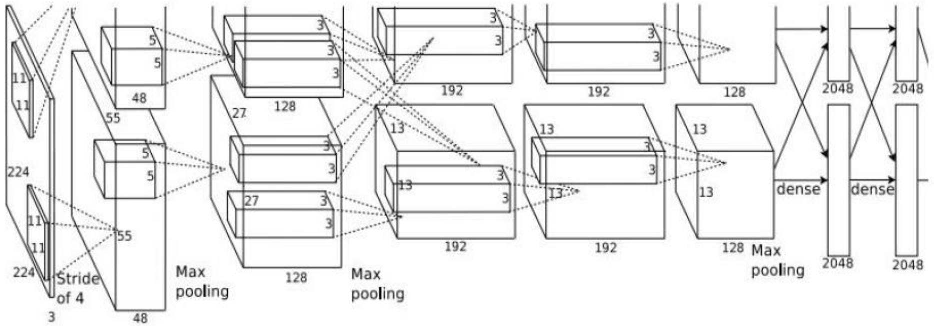


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

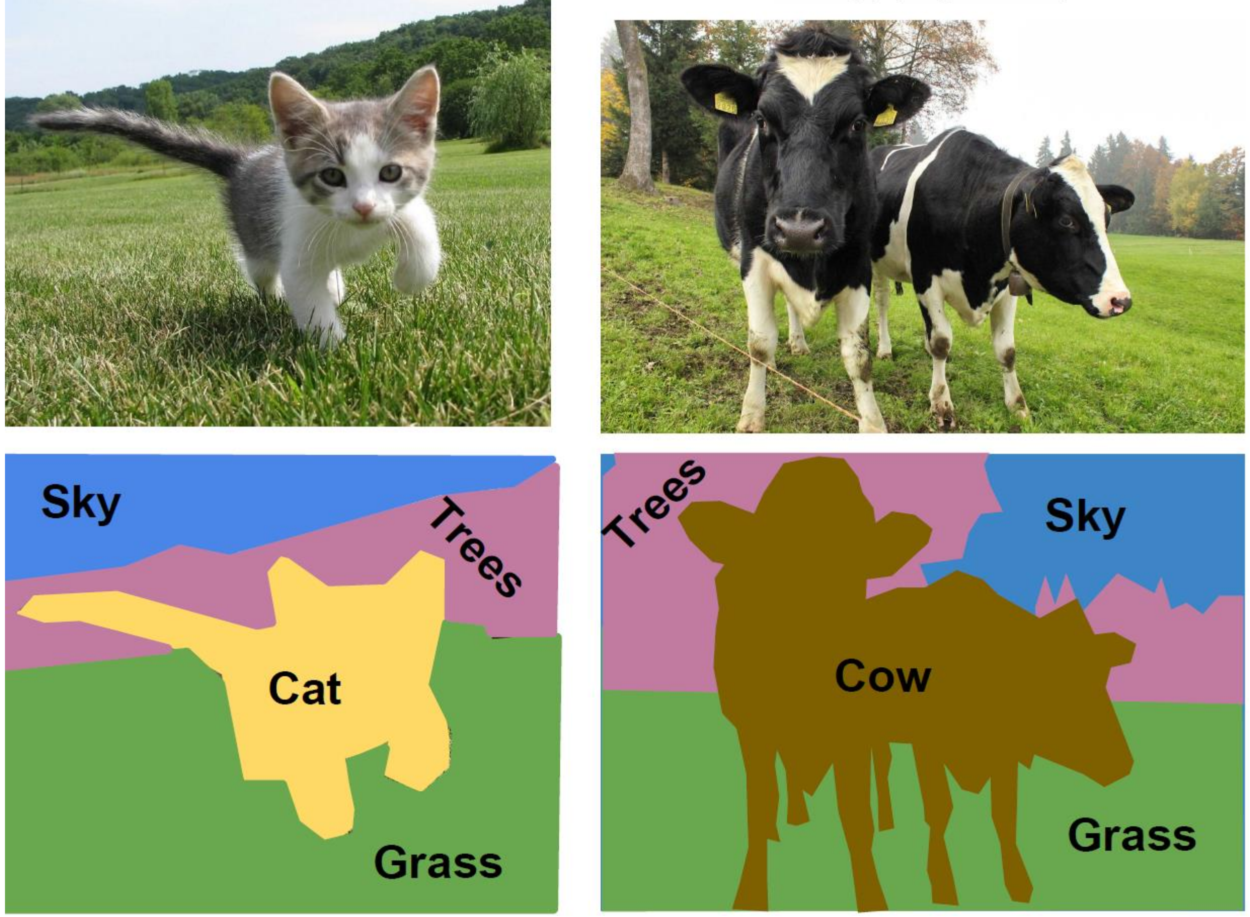
[1] Stanford university cs231n lecture slide

Semantic Image Segmentation

Label each pixel in the image with a category label

- **Pixel-level** classification task
- Requires **spatial** information

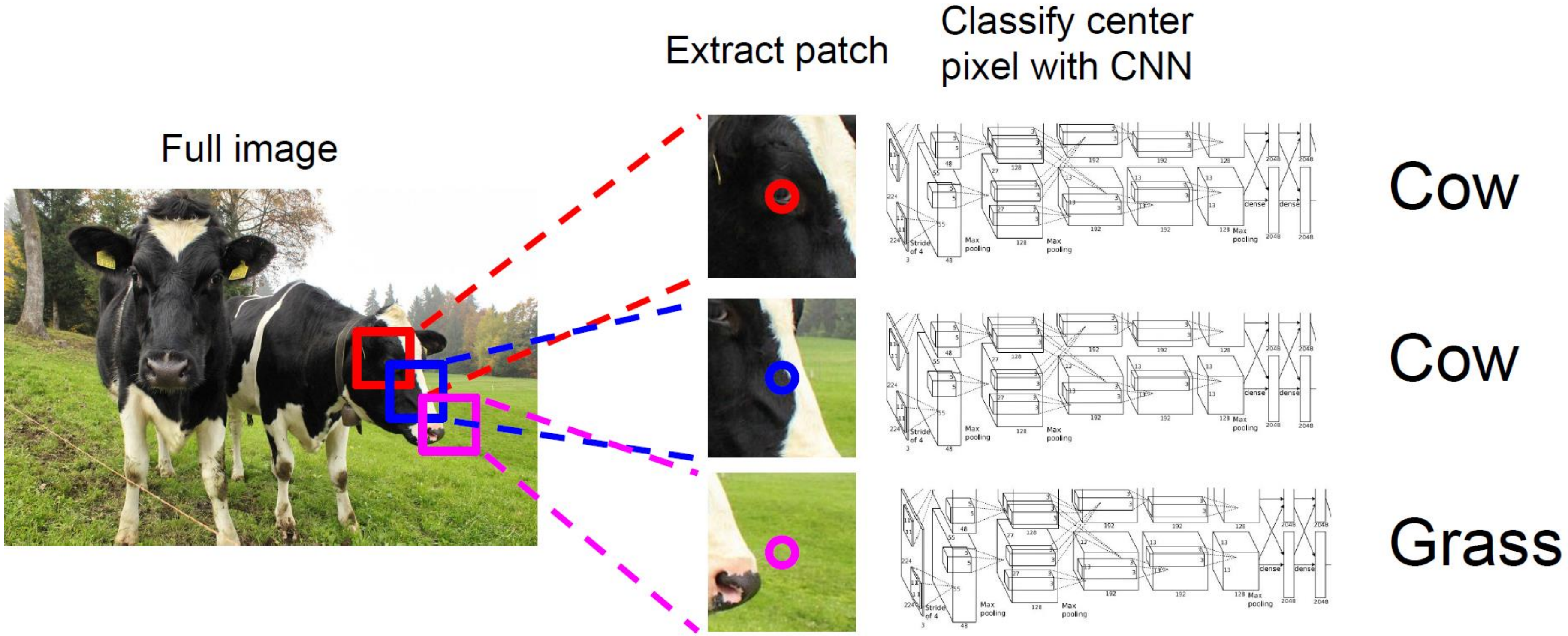
This image is CC0 public domain



[1] Stanford university cs231n lecture slide

Sliding Window Approach

Perform classification by **sliding window**



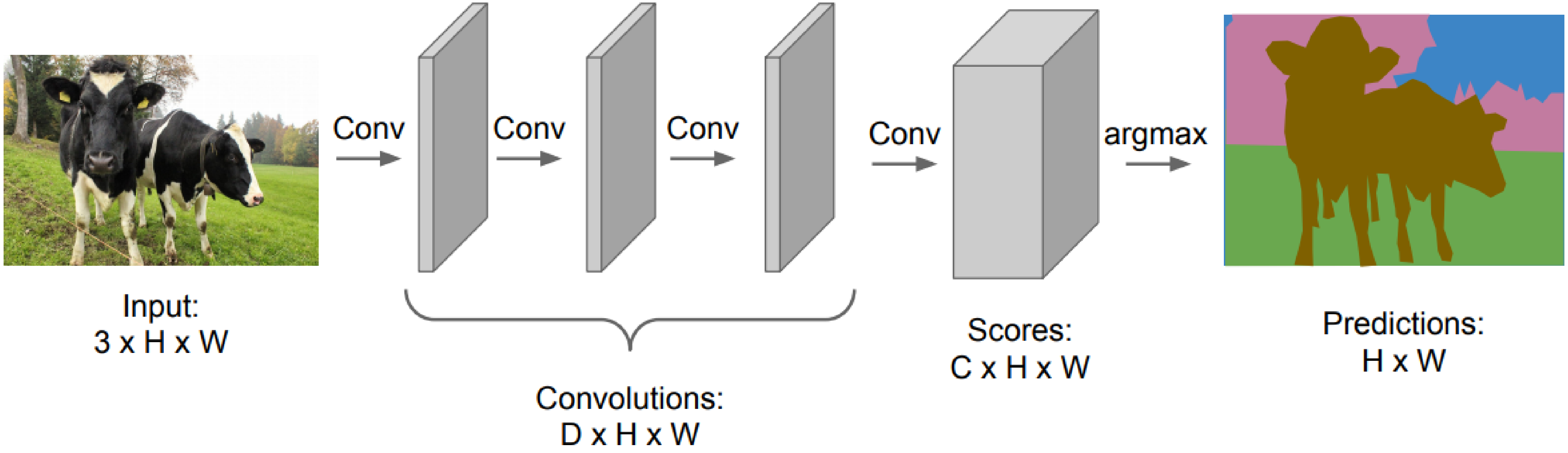
→ Computationally **inefficient**

= Not reusing features for shared image region

[1] Stanford university cs231n lecture slide

Fully Convolutional Approach

Design a network as **a bunch of convolutional layers** (preserving spatial information) to make predictions for pixels **all at once**!

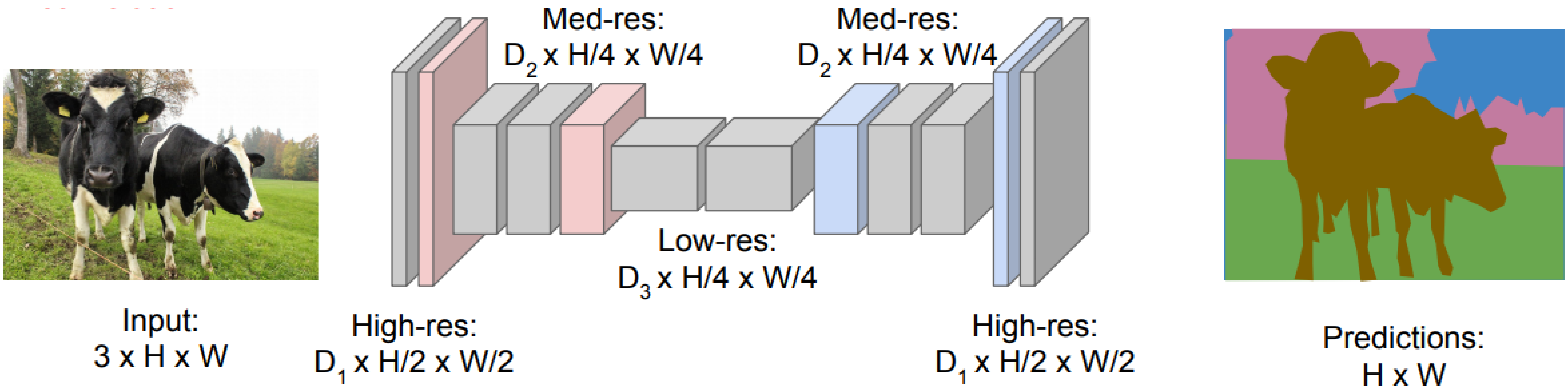


→ Convolutions at original image resolution will be **very expensive!**

[1] Stanford university cs231n lecture slide

Fully Convolutional Approach

Design a network as a bunch of convolutional layers,
with **downsampling** and **upsampling** inside the network!

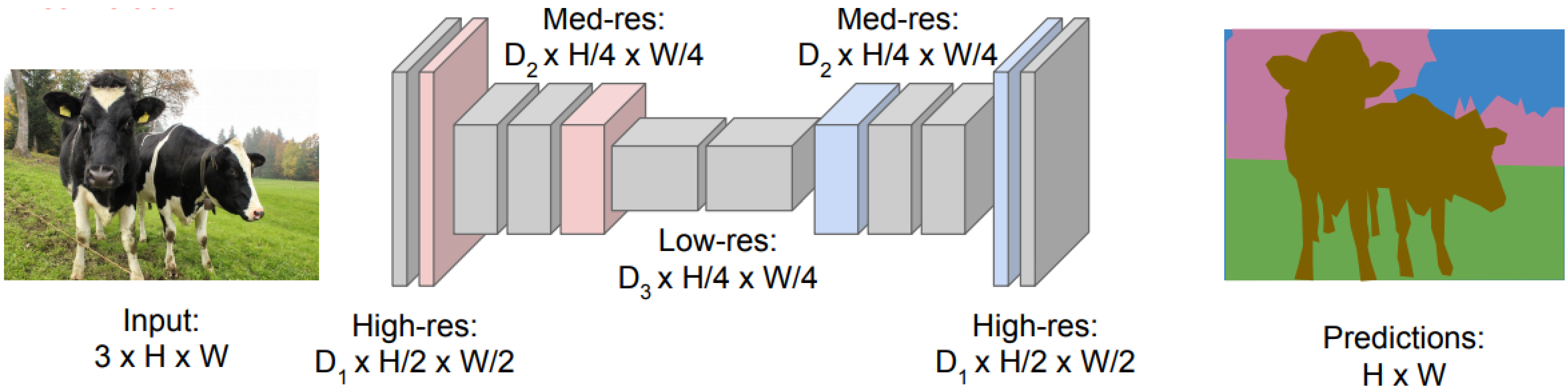


Downsampling: pooling, strided convolution (encoder)
Upsampling: unpooling, strided transpose convolution (decoder)

[1] Stanford university cs231n lecture slide

Fully Convolutional Approach

Design a network as a bunch of convolutional layers,
with **downsampling** and **upsampling** inside the network!



→ This is the **basic structure (Encoder-Decoder)** for segmentation task.

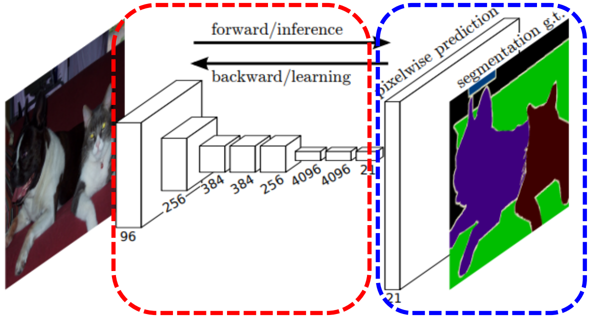
[1] Stanford university cs231n lecture slide

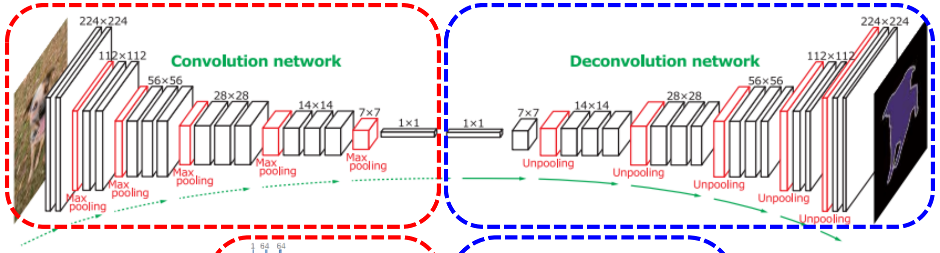
Encoder–Decoder Architectures

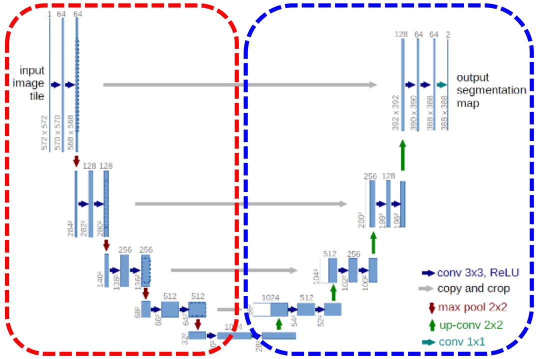
References

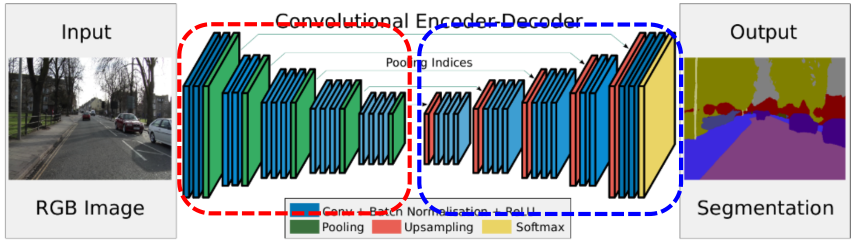
Encoder Decoder

- J. Long, et al., "**Fully Convolutional Networks.**" *CVPR*, 2015.
  → Will be deployed in Jetson Nano
- H. Noh, et al., "**Learning Deconvolution Network for Semantic Segmentation.**" *ICCV*, 2015.
- O. Ronneberger, et al., "**U-Net: Convolutional Networks for Biomedical Image Segmentation.**" *MICCAI*, 2015.
- V. Badrinarayanan, et al., "**SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.**" *T-PAMI*, 2016.









*Back to Basics: FC & Convolution

Fully-connected (FC) layer

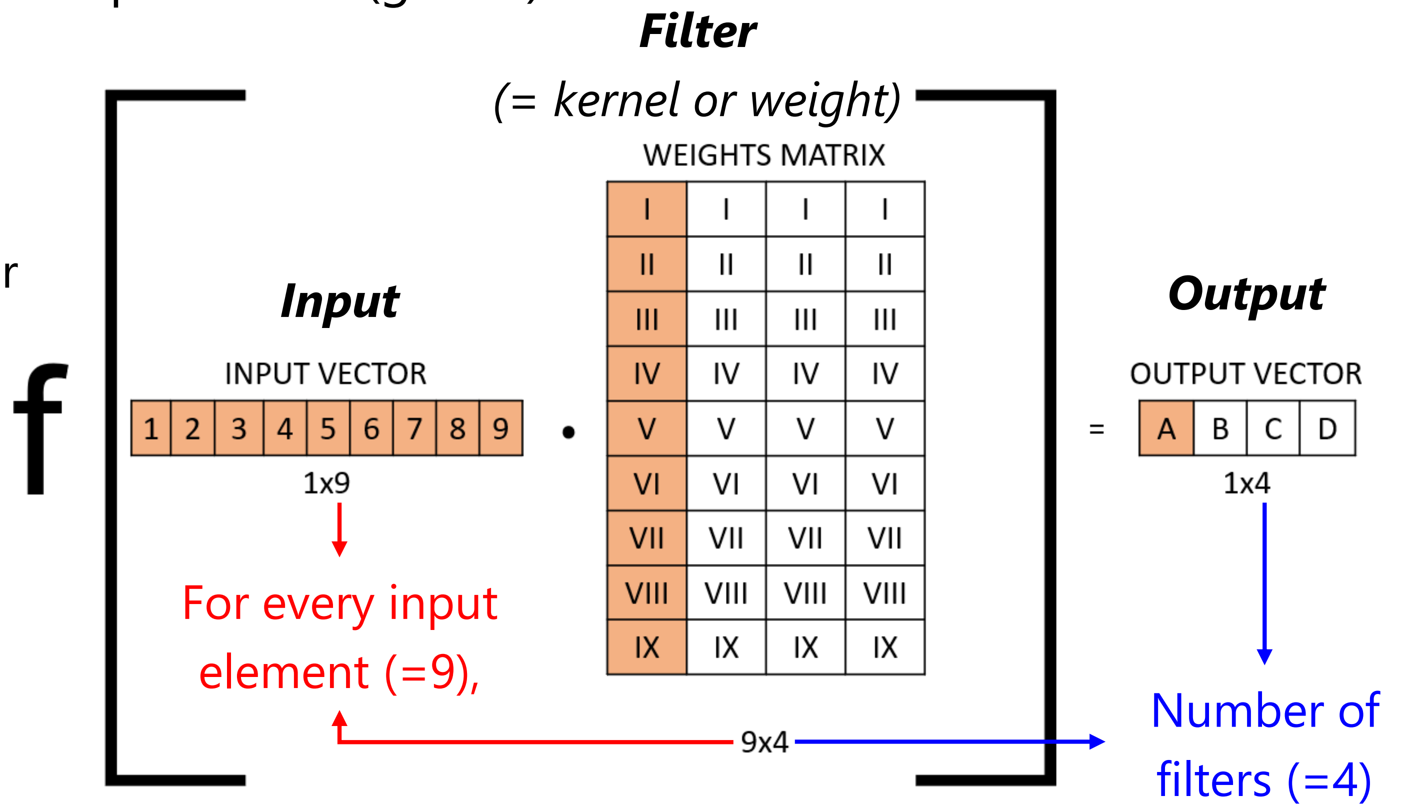
→ A.k.a. multi-layer perceptron (MLP)

→ **Vector** operation

→ Every input element affects output vector (global)

→ Input size is **fixed**

Input feature $3 \times 3 \times 1$
→ stretch to 1×9 vector



[1] Stanford university cs231n lecture slide

*Back to Basics: FC & Convolution

Convolutional layer

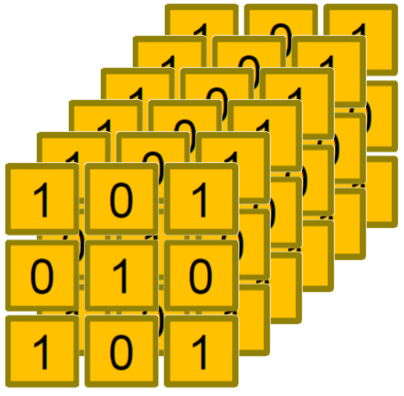
→ **Matrix** (2D or 3D) operation

→ Aggregate **local** & **spatial** features (flexible input resolution)

→ 1 × 1 convolution = FC layer

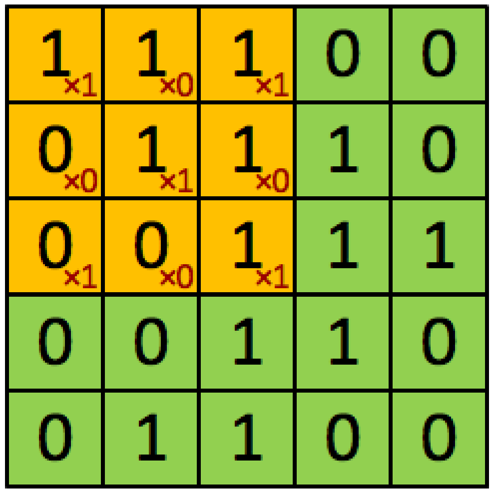
Real image sample

→ Sliding window operation

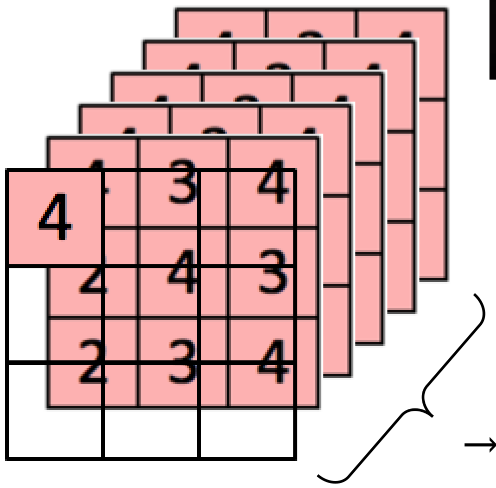


Filter size 3 × 3

Input image 5 × 5



Image



Convolved Feature

→ **If the number of Filters = 6**

→ **Number of output features = 6**

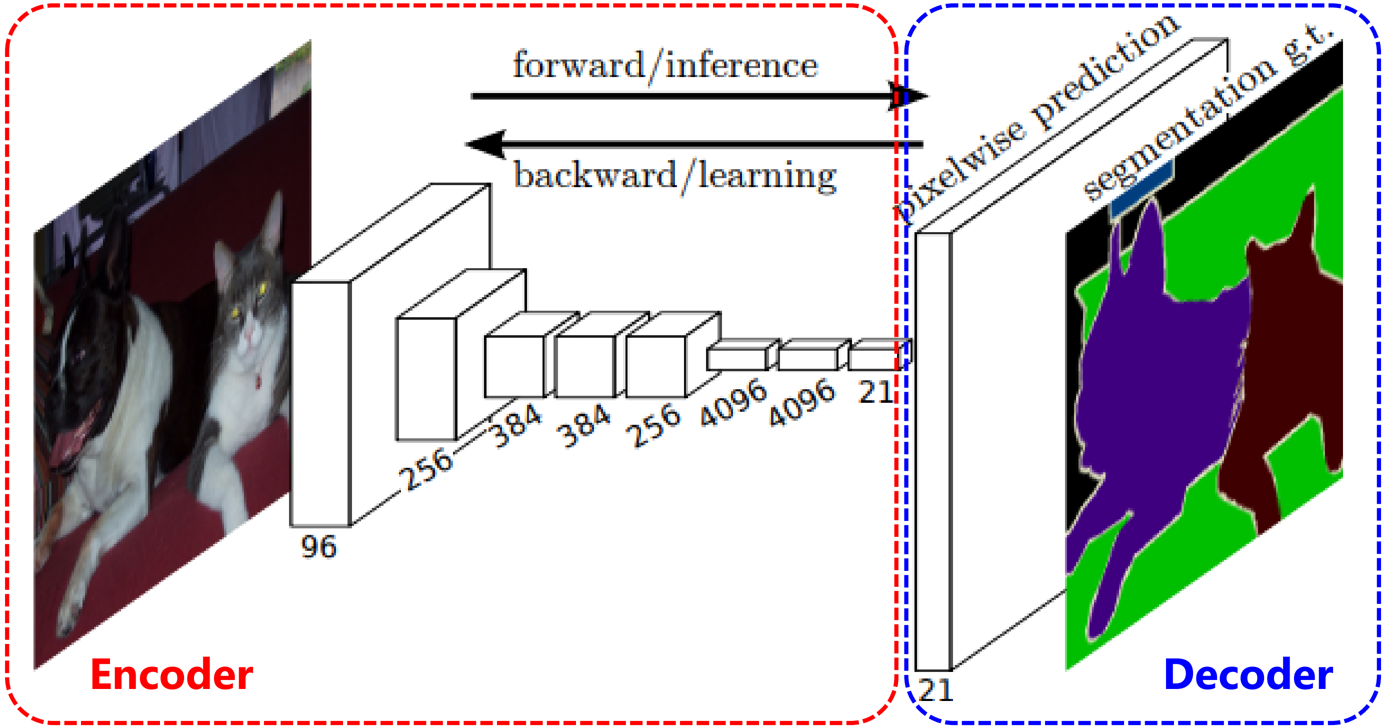
[1] Stanford university cs231n lecture slide

FCN: The First CNN-Based Segmentation Network

Key idea

→ **Fully convolutional layers** to extract **spatial** features

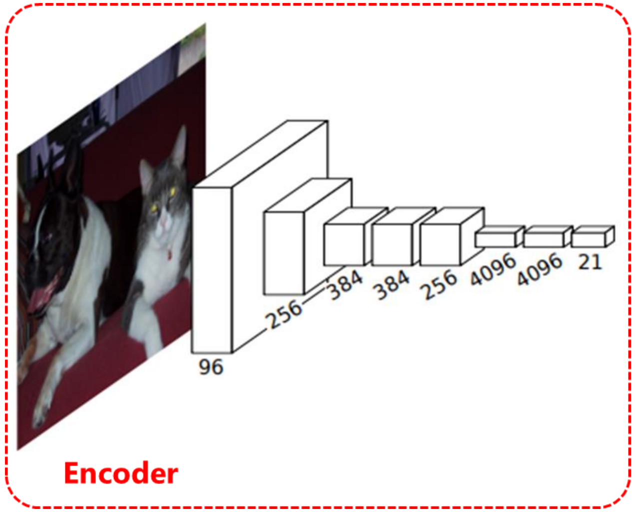
Specifically for VGG-16 model, the last three FC layers are replaced by conv layers.



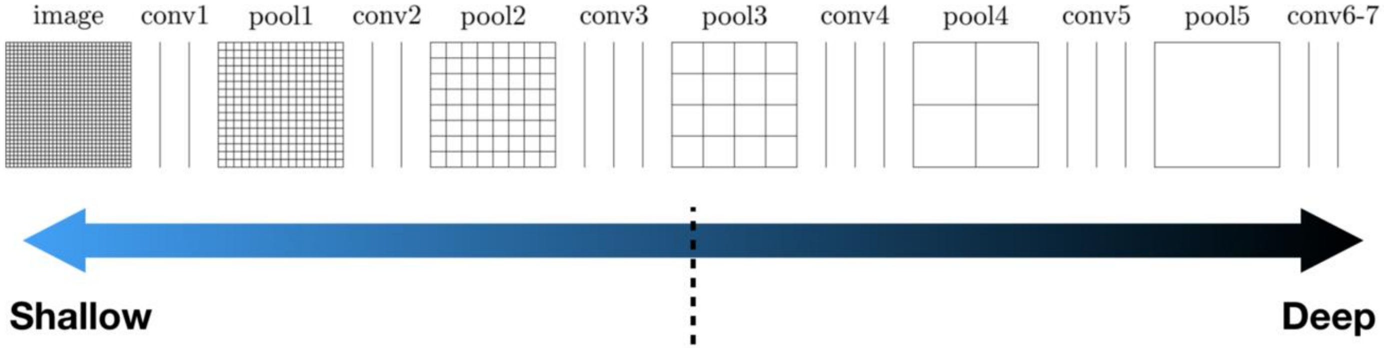
[1] J. Long, et al., “Fully Convolutional Networks.” CVPR, 2015.

Encoding = Extracting Abstract Features

Encoding stage



=



Fine	Coarse
Location	Semantic
Local	Global
Detail	Abstract

[1] J. Long, et al., "Fully Convolutional Networks." CVPR, 2015.

Decoding = Aggregating Details

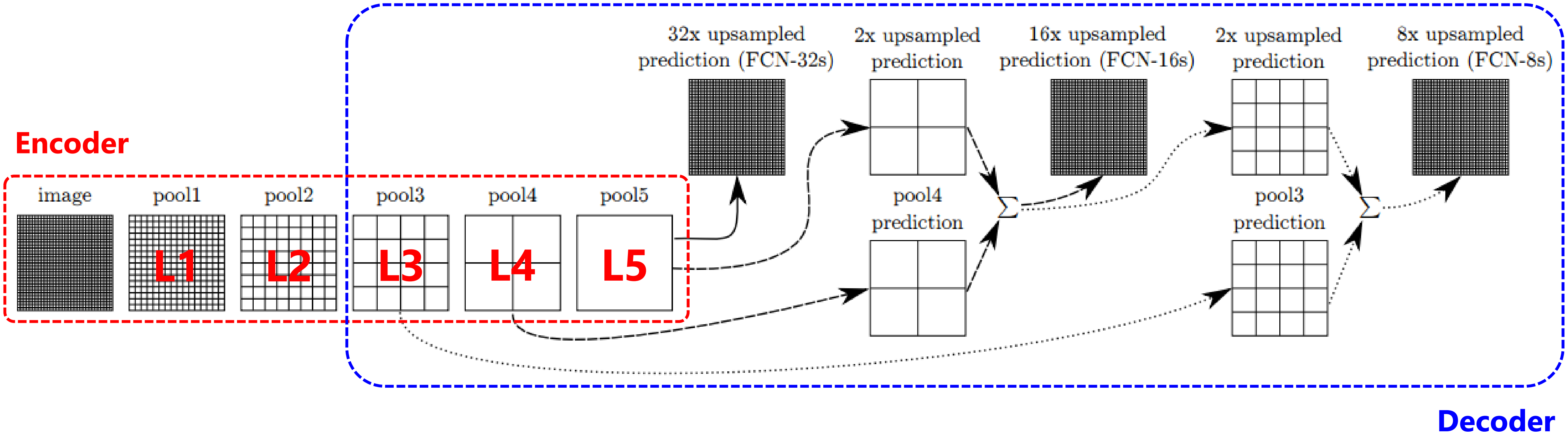
Decoding stage

→ Skip-connections to aggregate **multi-scale** spatial features.

Specifically, it takes details from **L3**, **L4**, and **L5**

→ Upsampling using a bilinear interpolation

Also differentiable for backpropagation



[1] J. Long, et al., "Fully Convolutional Networks." CVPR, 2015.

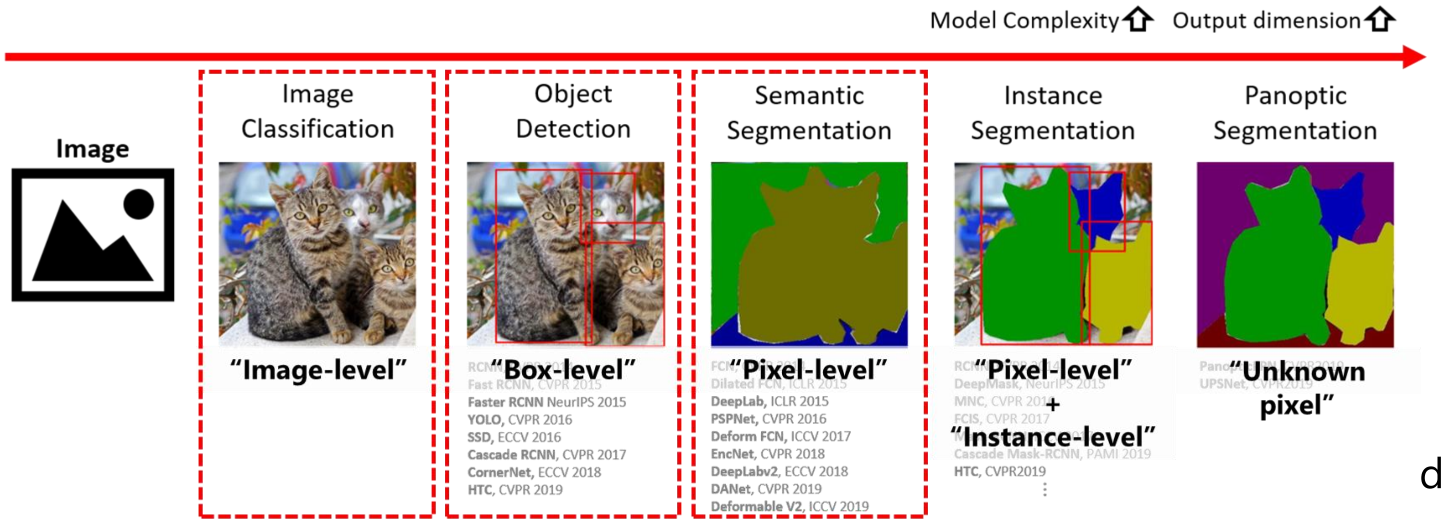
Summary

Basic computer vision tasks (semantic tasks)

→ Image classification (AlexNet, VGG, GoogleNet, ResNet): basic deep neural networks

→ Object detection (R-CNN, SSD): Trade-off between two-stage & one-stage detectors

→ Semantic segmentation (Fully Convolutional Network): Fully-connected vs. Convolutional layers



→ Combinatorial works of object detection and semantic segmentation

→ We will move on to **geometric tasks** in the next class!

KENTECH
Korea Institute of Energy Technology

Week 08b – Semantic Segmentation on Jetson Nano

Experiments

Before starting

*Your basic workspace is here: "cd ~/jetson-inference/build/aarch64/bin" Every code is pre-built in this path.

Live semantic segmentation with visualization

Q1. Run "python segnet.py --flip-method=rotate-180". What is on the screen? What is different from detectnet in the previous class? Can you guess the meaning of the color on the screen?

Try different models for semantic segmentation

Q2.1. Run "python segnet.py --flip-method=rotate-180" again with objects and scenery (If possible, include as many objects as possible). Observe the default network and segmentation.

Q2.2. Go to the linked page (https://github.com/dusty-nv/jetson-inference/blob/master/docs/segnet-console-2.md). Please read the list of segmentation models. Try to use different models, trained on different datasets. Please specify the list of classes for each dataset.

Q2.3. (optional) Check your installed models through below commands.
"cd ~/jetson-inference/build/aarch64/bin/networks"
"ls"
you can see the model folder installed.

Experiments

Q2.4. (optional) After Q2.3., follow the command to download models.
"cd ../" move back to ~/jetson-inference/build/aarch64/bin/
"cd ../../../" move to ~/jetson-inference/
"cd tools/" move to ~/jetson-inference/tools/
"./download-models.sh"
After download,
"cd ../" move to ~/jetson-inference/
"cd build/aarch64/bin/" move to the basic workspace.

Q2.5. Run specified model "python segnet.py --network=fcn-resnet18-cityscapes-512x256 --flip-method=rotate-180". And then try same name's model but pixel size is different. "python segnet.py --network=fcn-resnet18-cityscapes-1024x512 --flip-method=rotate-180".

Q2.6. Which model derives a more elaborate result? How about FPS? (Which model segments quickly when you change the angle?)

Q2.7. Try different models with the same view.
Run "python segnet.py --network=fcn-resnet18-sun-512x400 --flip-method=rotate-180". How the segmentation of the same object different is per model?

Experiments

Effect of illumination

Q3.1. With less illumination (e.g., make a shadow in front of the target object), run "python segnet.py --network=fcn-resnet18-cityscapes-1024x512 --flip-method=rotate-180" How does the segmentation work?

Q3.2. With more illumination (e.g., use a smartphone flash or control the classroom's light intensity), run the same command in 3.1. How does the illumination condition affect the segmentation?

Segmentation on your own images

Q4.1. Download arbitrary scenery images, people images, animals images. Move the images to the ~/jetson-inference/build/aarch64/bin/images folder.

Q4.2. Segmentation on each photo with three different models.
Run "python segnet.py --network=fcn-resnet18-mhp-640x360 images/photo.jpg images/test/out_seg_photo_mhp640.jpg",
"python segnet.py --network=fcn-resnet18-sun-512x400 images/photo.jpg images/test/out_seg_photo_sun512.jpg", …

Experiments

Q4.3. Observe the results of segmentation which are saved in ~/jetson-inference/build/aarch64/bin/images/test folder.

Try using options for semantic segmentation

Q5.1. Try Q4.2 and 4.3 with visualize option (Default is overlay). Run "python segnet.py --network=fcn-resnet18-deepscene-576x320
--visualize=mask images/photo.jpg images/test/out_seg_photo_deepscene576_mask.jpg".

Q5.2. In the results, which model is the best to recognize each photo?

Q5.3. Try one of the images with filter-mode (Default is linear). Run "python segnet.py --network=fcn-resnet18-voc-512x320 --filter-mode=point images/photo.jpg images/test/out_seg_photo_voc512.jpg".

Q5.4. Control alpha(Default alpha=120). Run "python segnet.py --network=fcn-resnet18-voc-512x320 --alpha=50 --filter-mode=point images/photo.jpg images/test/out_seg_photo_voc512_alpha50.jpg".

Experiments

Some useful tips while debugging

*Sometimes, the python process does not respond. In this case, please terminate the process with `ctrl+c`. If it still does not respond at all, forcibly stop the process with `ctrl+z`, and check the running process name with the `ps -a` command, and then type `sudo pkill -9 [name-of-process]` command to kill the process. If you don’t shut it down, it will remain as a 🧟zombie🧟 and keep occupying the processor (CPU or GPU) in the background.

*Sometimes, the best solution for resolving an issue is just rebooting the system.

“cd” = “change directory”

“ls” = “list segment (files & directories)”