

DeepPhoto

Real-time fine-grained image editing for social applications

Roberta CONRAD | Ching-yu LIN | Jørgen LUND | Fernando PÉREZ | Jiahao WANG

- 01** Our team
- 02** Business case
- 03** Semantic segmentation
- 04** Light weight RefineNet
- 05** (Neural Style Transfer)
- 06** Conclusion and examples

The team that has been working on the project



Fernando Pérez



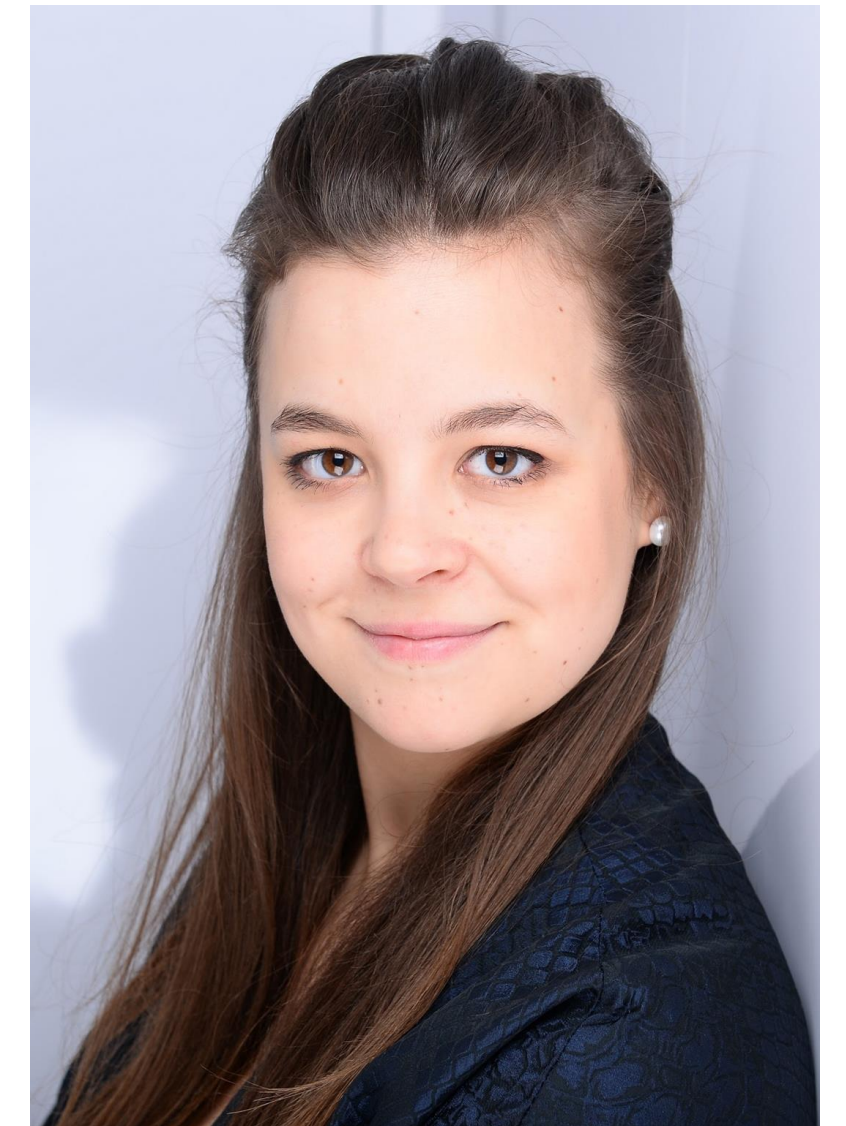
Jørgen Lund



Jiahao Wang



Ching-Yu Lin



Roberta Conrad

Nowadays, there is a **boom** of applications that allow **users to apply different filters to a picture to make it cooler**. However, these applications don't allow to apply filters **only to specific part of the picture**.



What?



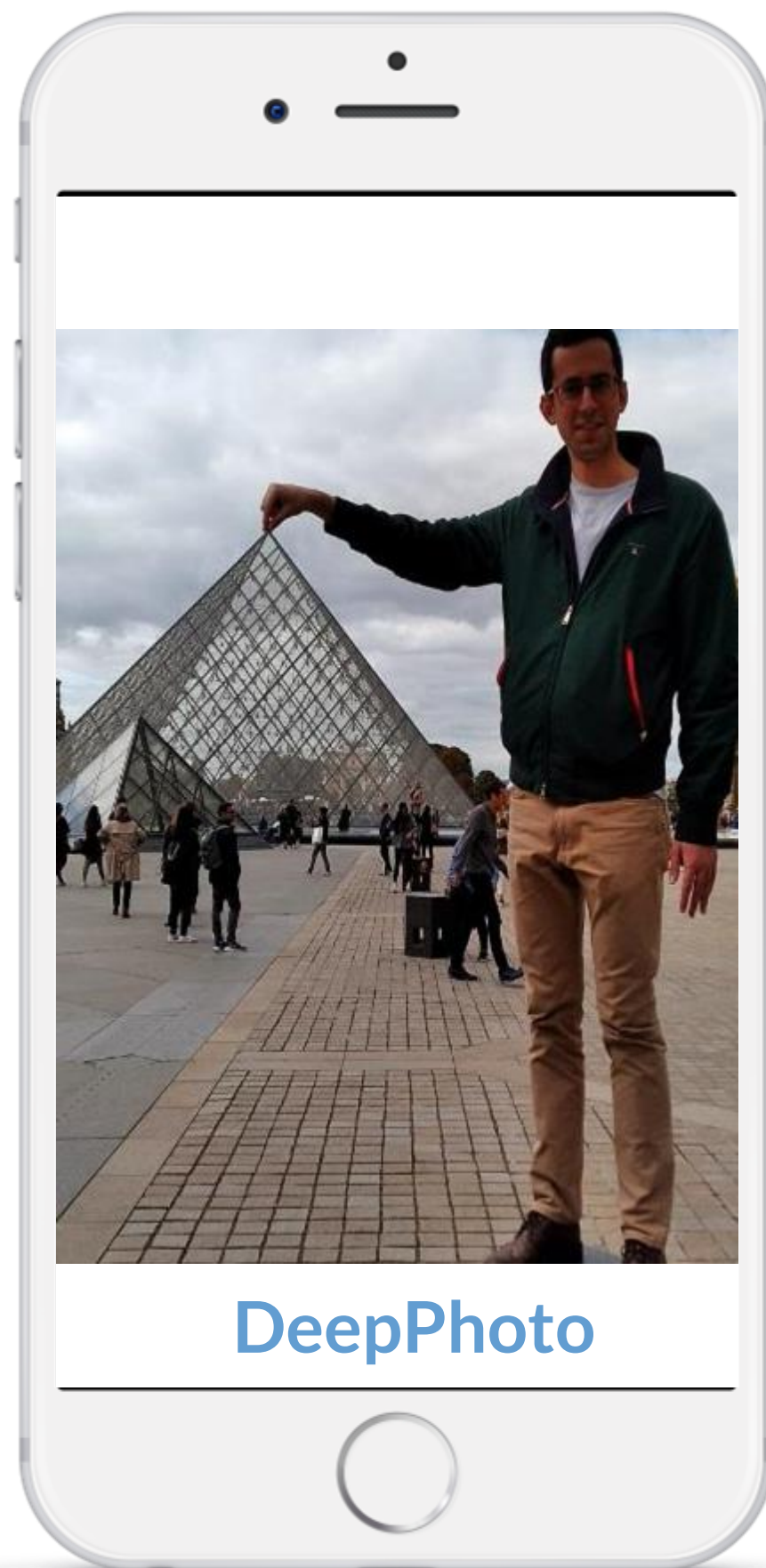
- Application that uses latest computer vision techniques to allow users to edit specific semantic parts of a picture.
- Users can select the style and decide whether change the style to the **whole picture, or just the background or body**.

Why?

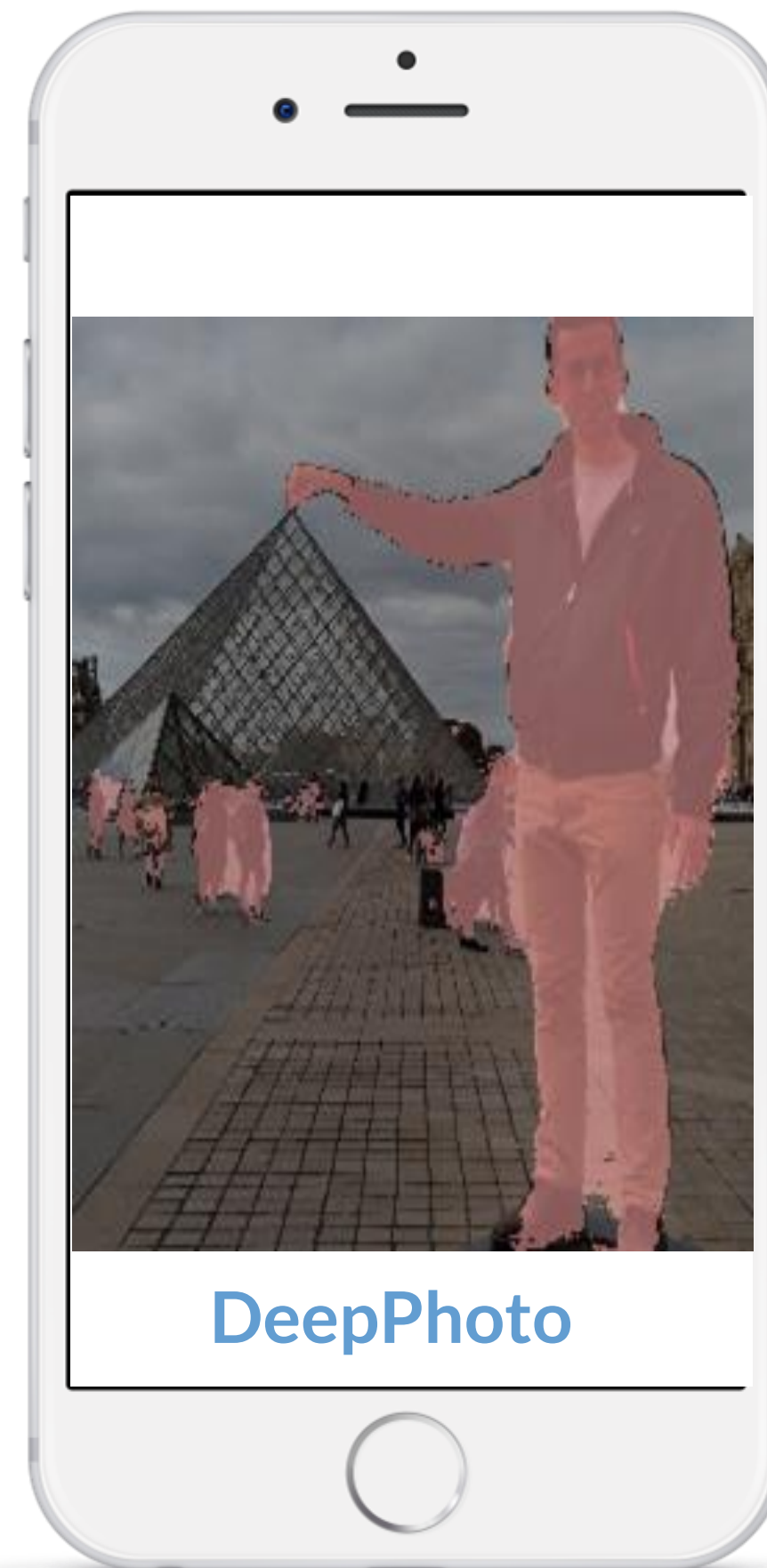
- Applications in the market don't allow to specify to which part of the picture you want to change the style and have a very limited number of different styles.
- Take advantage of the latest advances in computer vision techniques.

How the DeepPhoto works?

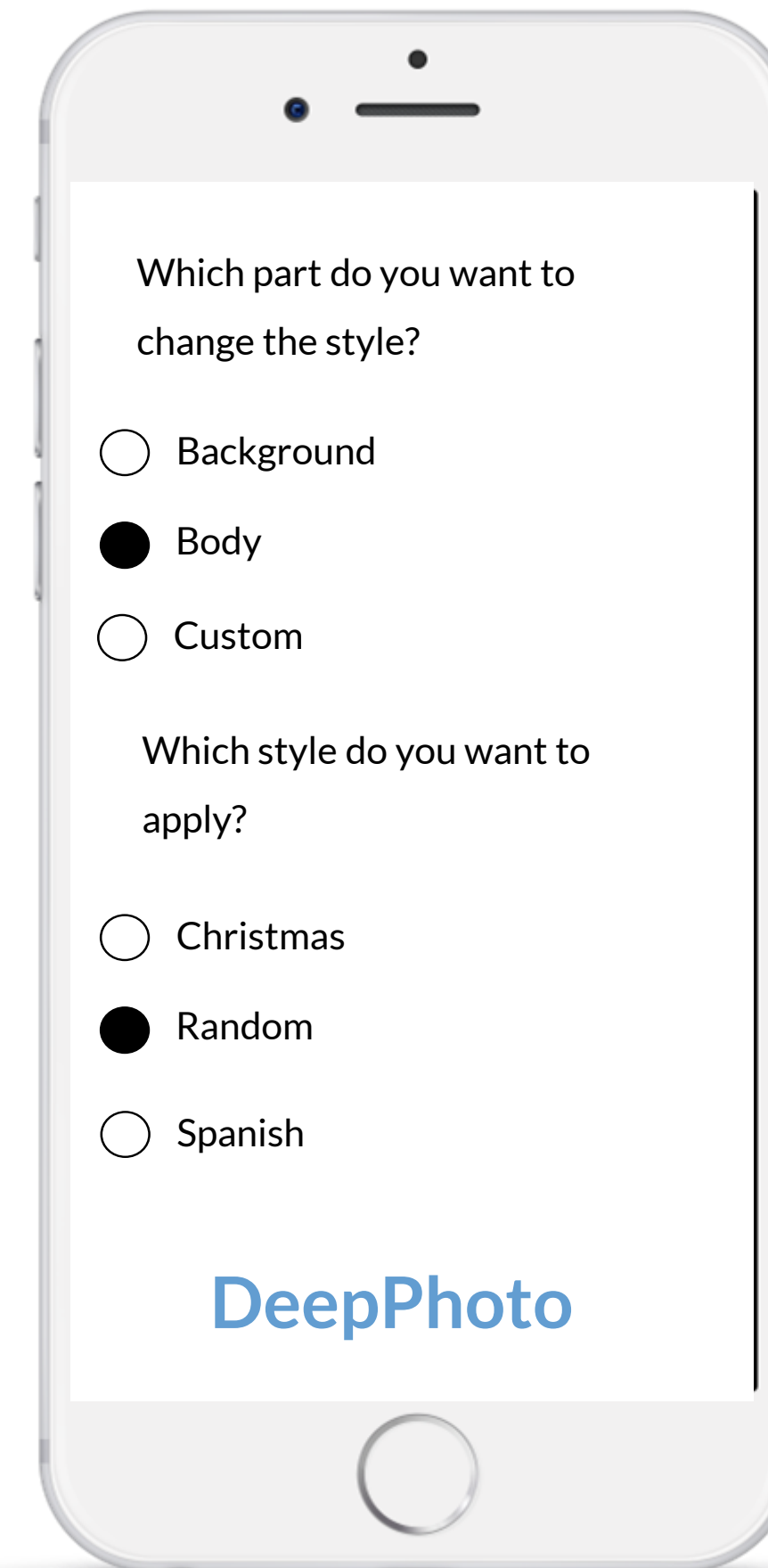
5



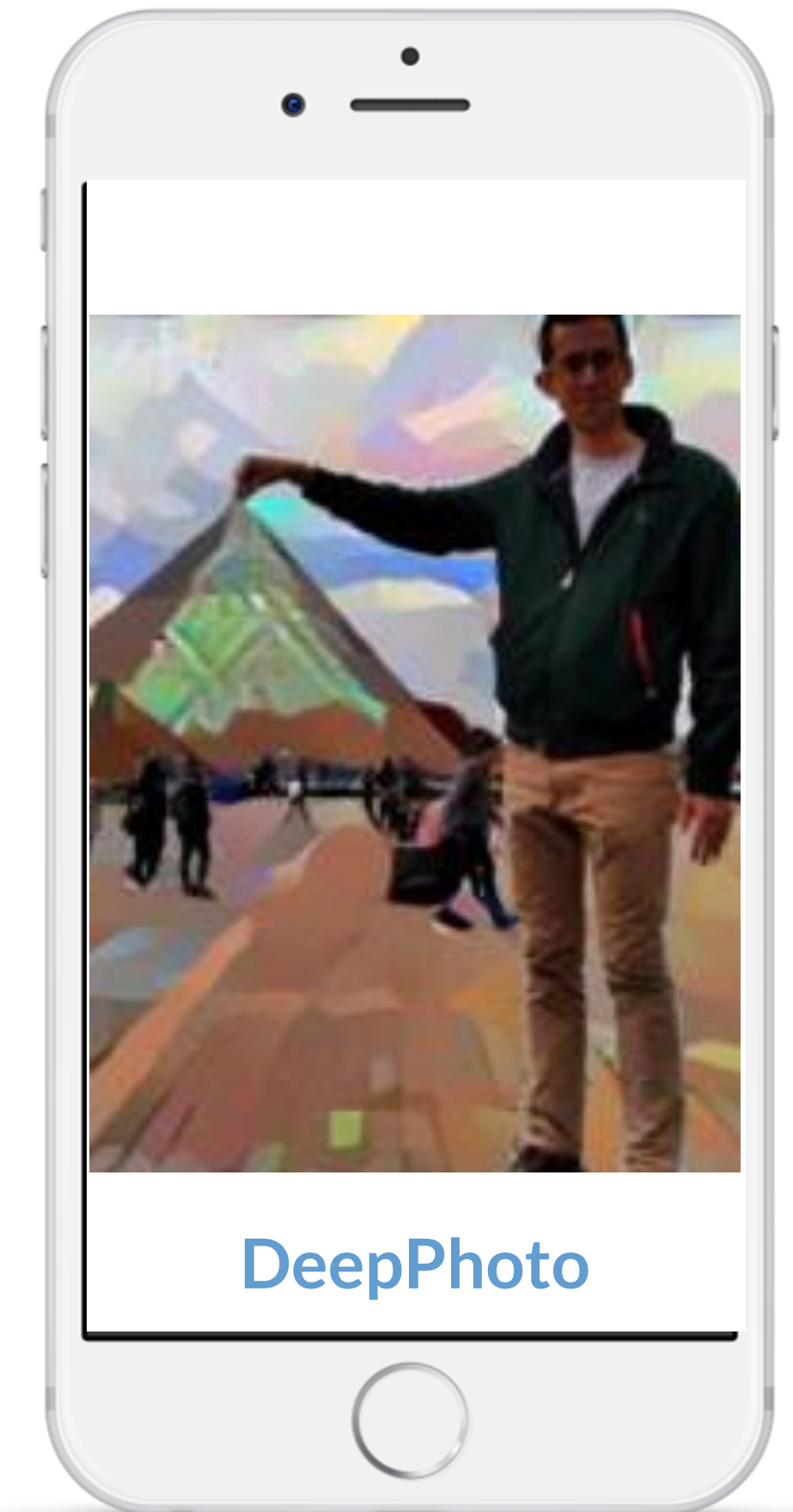
First, users take a great picture or upload to the app one previously taken.



Our algorithm detect the different components inside picture.



User select how they want to personalize the picture.



Users download the final crazy picture.



First step is to detect the different elements of the picture with semantic segmentation

6

Tasks

- **Classification.** It's a process of grouping pixels into several classes.
- **Object detection.** It deals with identifying and locating object of certain classes in the image.
- **Semantic segmentation.** It's the task of classifying every pixels in an image into a class.
- **Instance segmentation.** Instance segmentation is the task of detecting and delineating each distinct object of interest appearing in an image.

Example of input and output

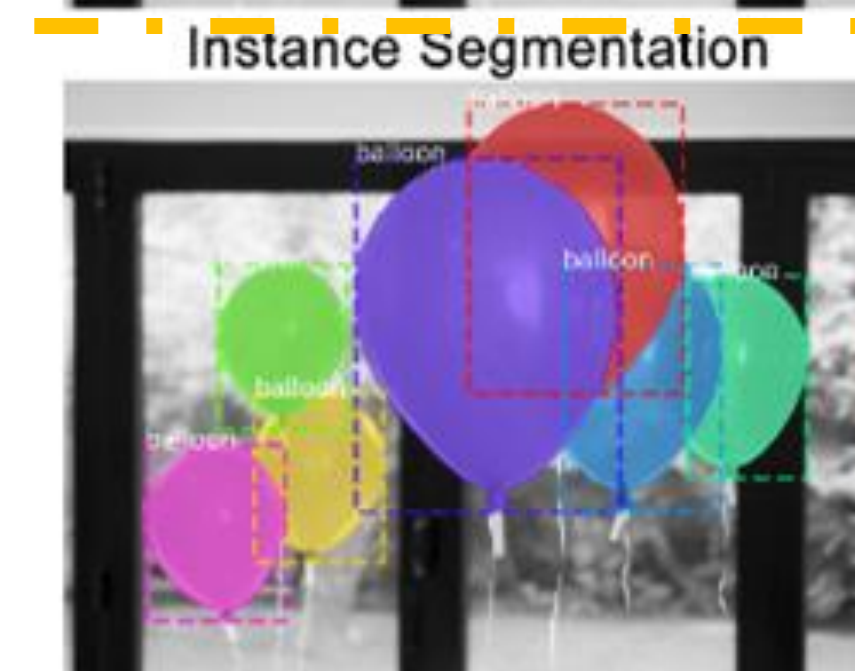
$256 \times 256 \times 3 \rightarrow \{0,1\}^n$, with
 $n = \# \text{ of classes}$

$256 \times 256 \times 3 \rightarrow 2 \text{ points } \# \text{ Objects}$

$256 \times 256 \times 3 \rightarrow 256 \times 256 \times$
 $\# \text{ classes}$

$256 \times 256 \times 3 \rightarrow 256 \times 256 \times$
 $\# \text{ classes} \times \# \text{ objects}$

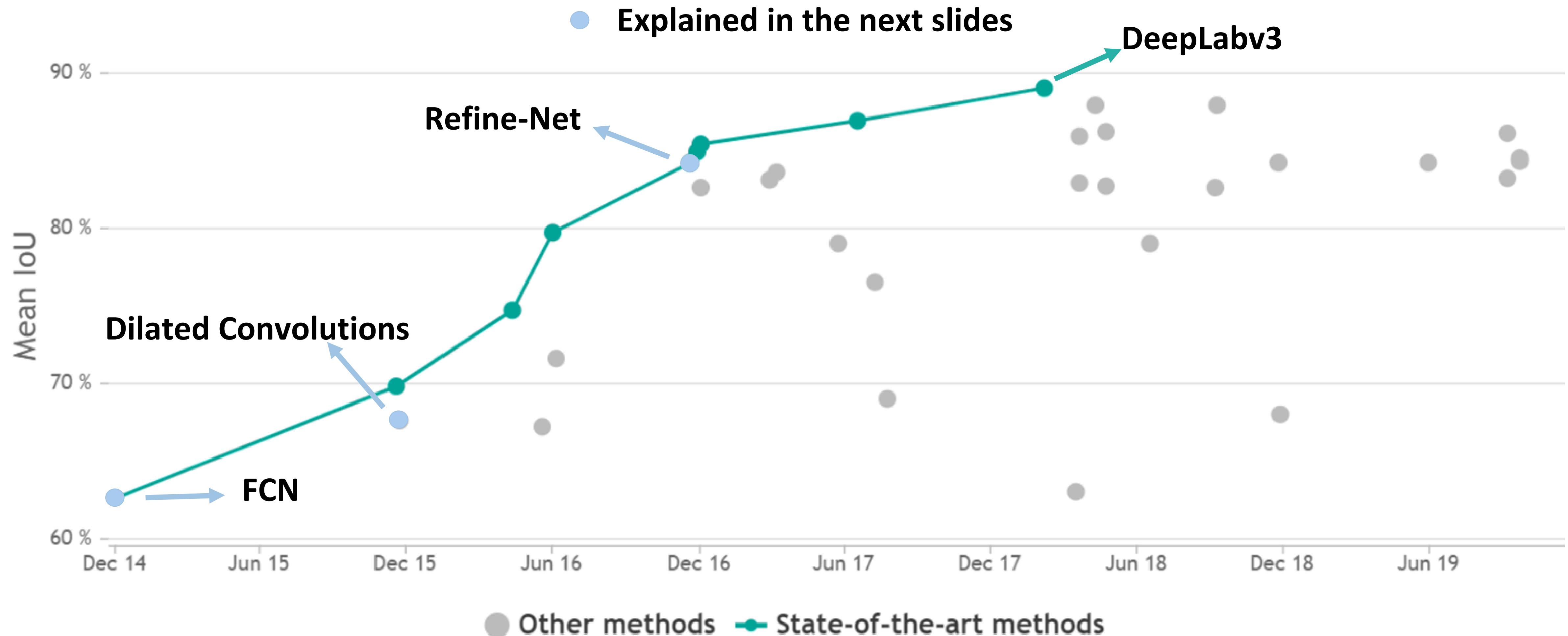
 Focus of the project



Complexity

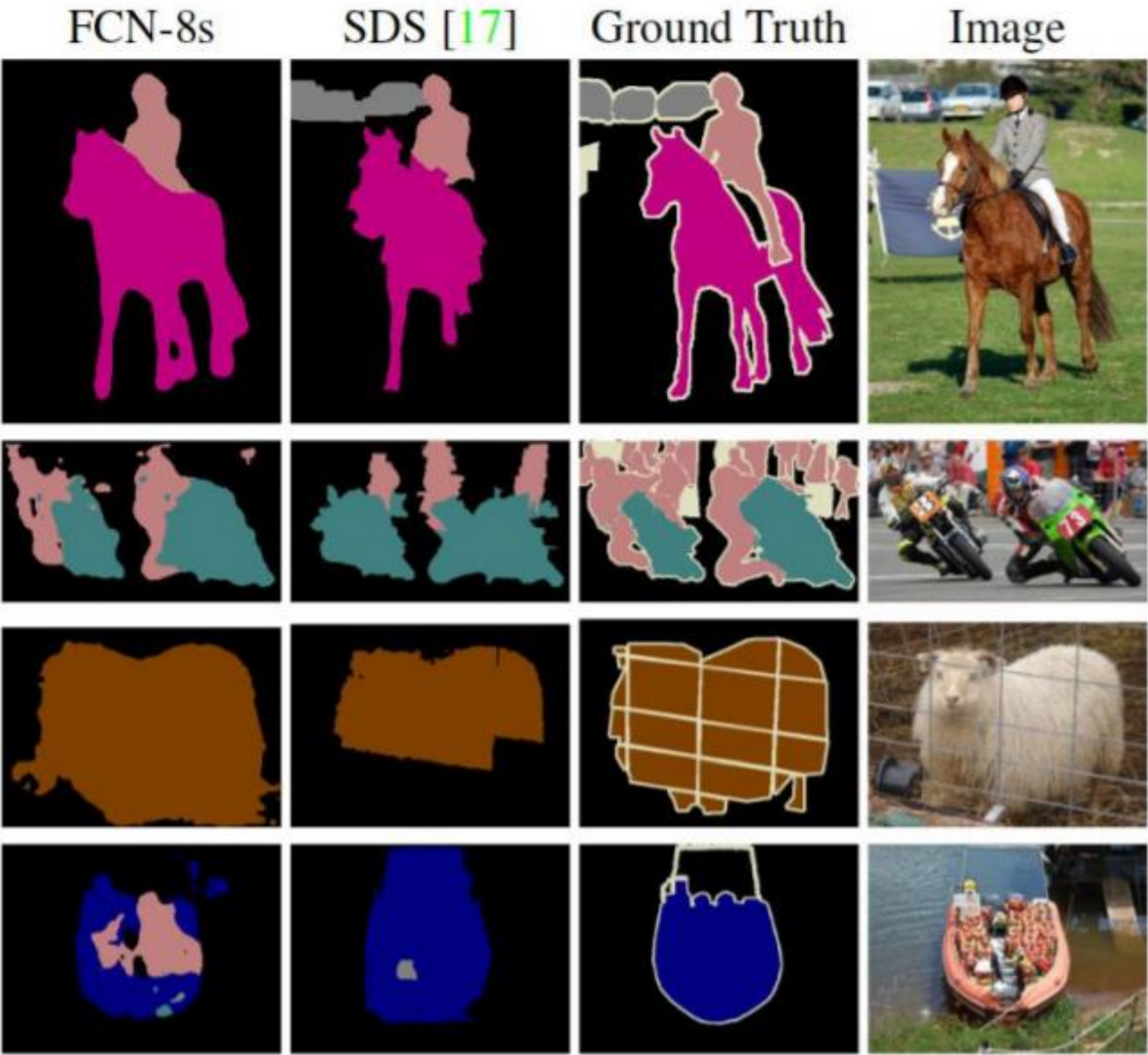
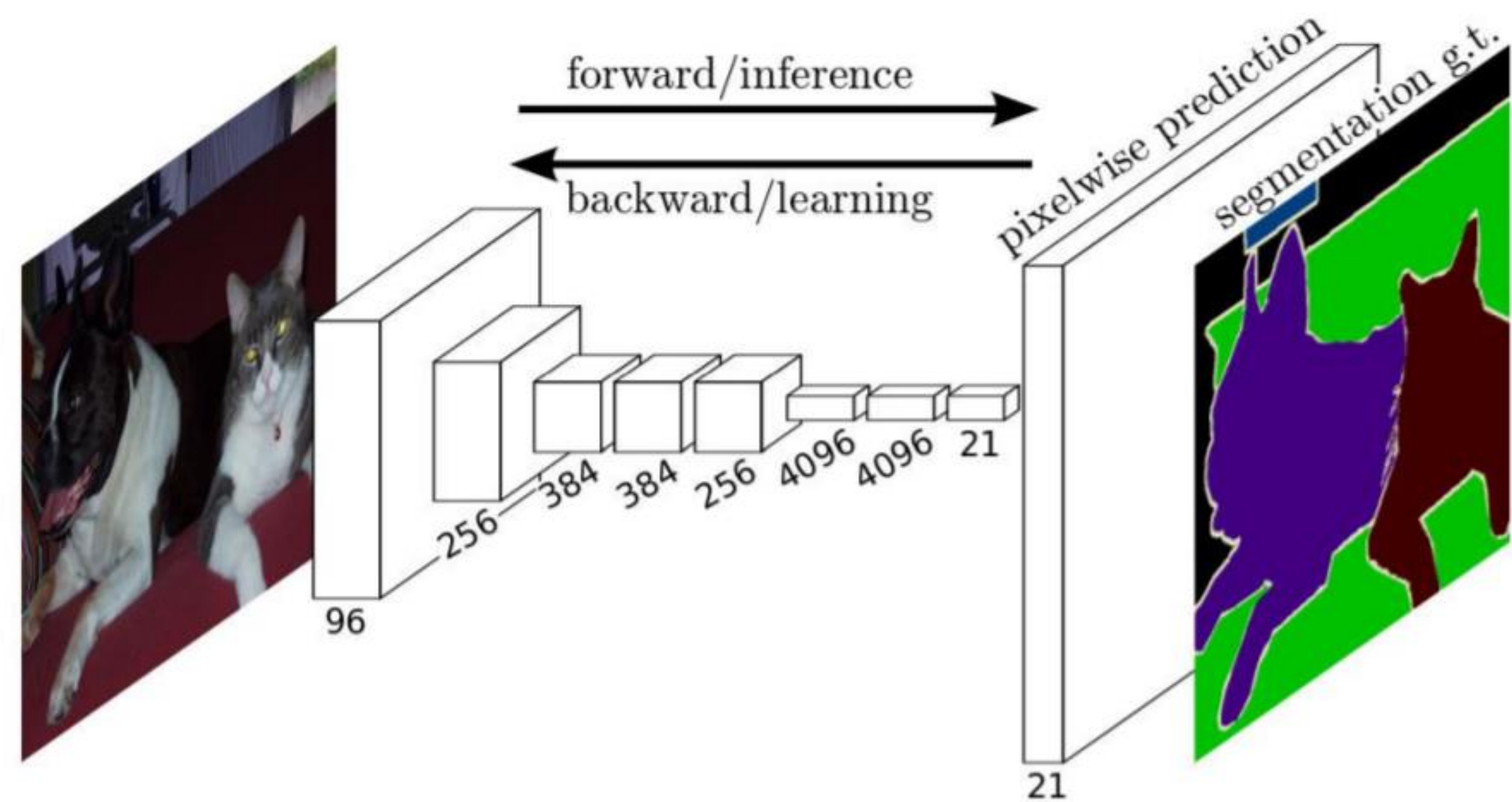
Since the launched of FCN in 2014, Semantic Segmentation has evolved a lot in the past few years.

7



One of the first development in semantic segmentation was Fully Convolutional Network (FCN)

FCN is the first algorithm for doing semantic segmentation. This model uses various blocks of convolution and max pool layers to first decompress an image to 1/32th of its original size. It then makes a class prediction at this level of granularity. Finally it uses up sampling and deconvolution layers to resize the image to its original dimensions.



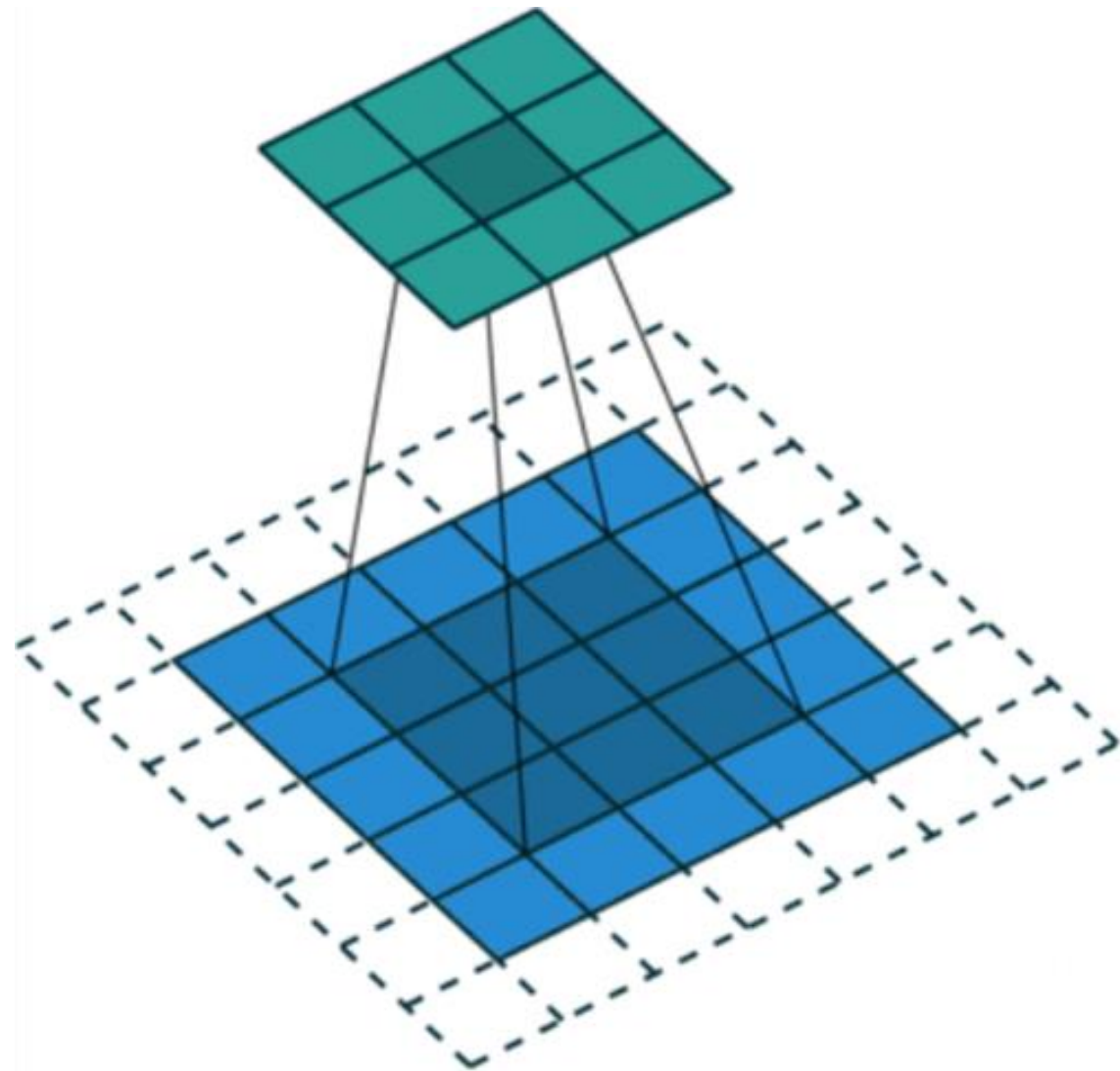
FCN resolution can be improved by the use of dilated convolution:

Dilated-FCN

9

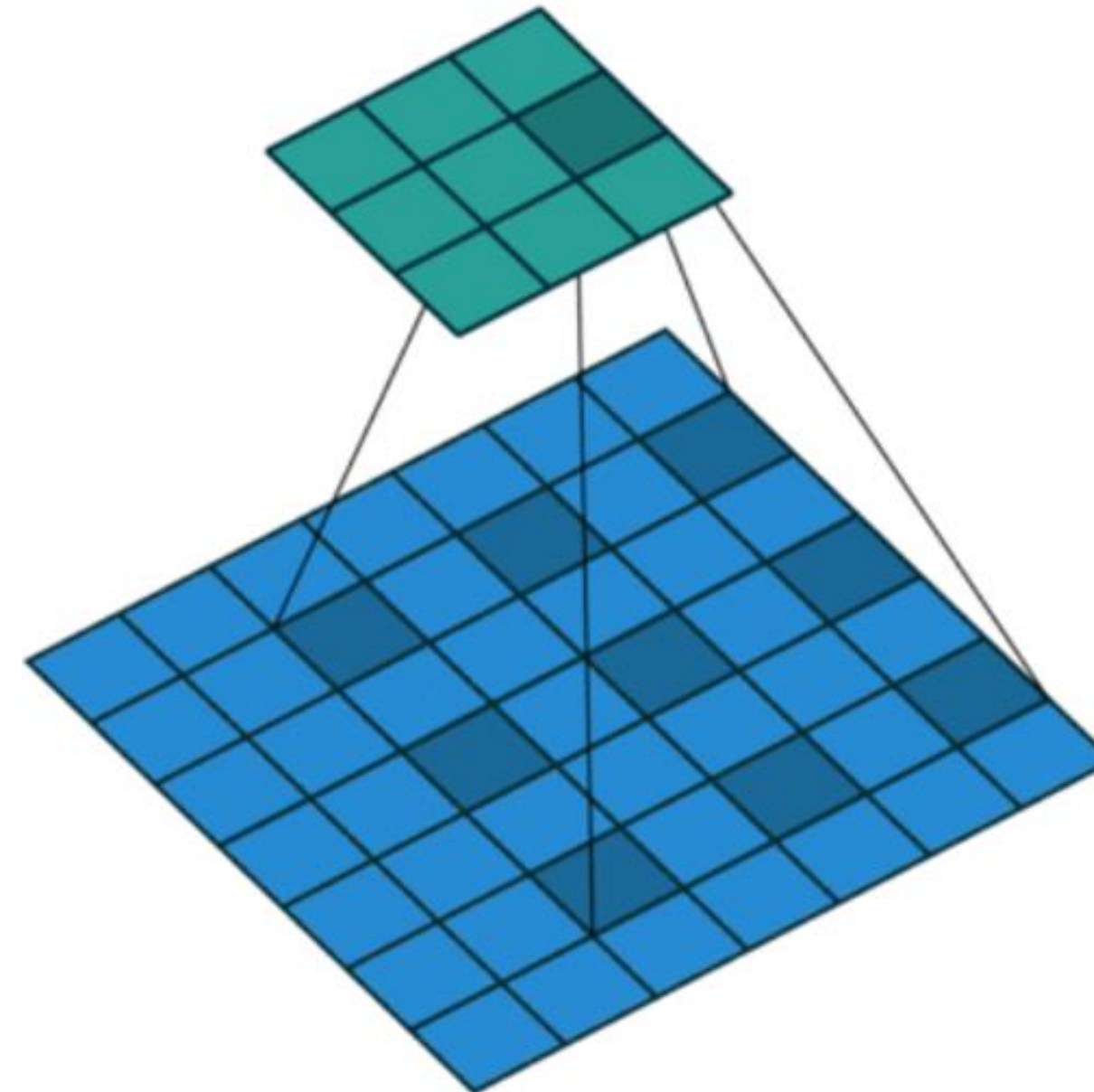
Normal convolution

$$(F * k)(p) = \sum_{s+t=p} F(s) k(t).$$



Dilated convolution

$$(F *_l k)(p) = \sum_{s+lt=p} F(s) k(t).$$



Dilated convolution
with $l = 2$

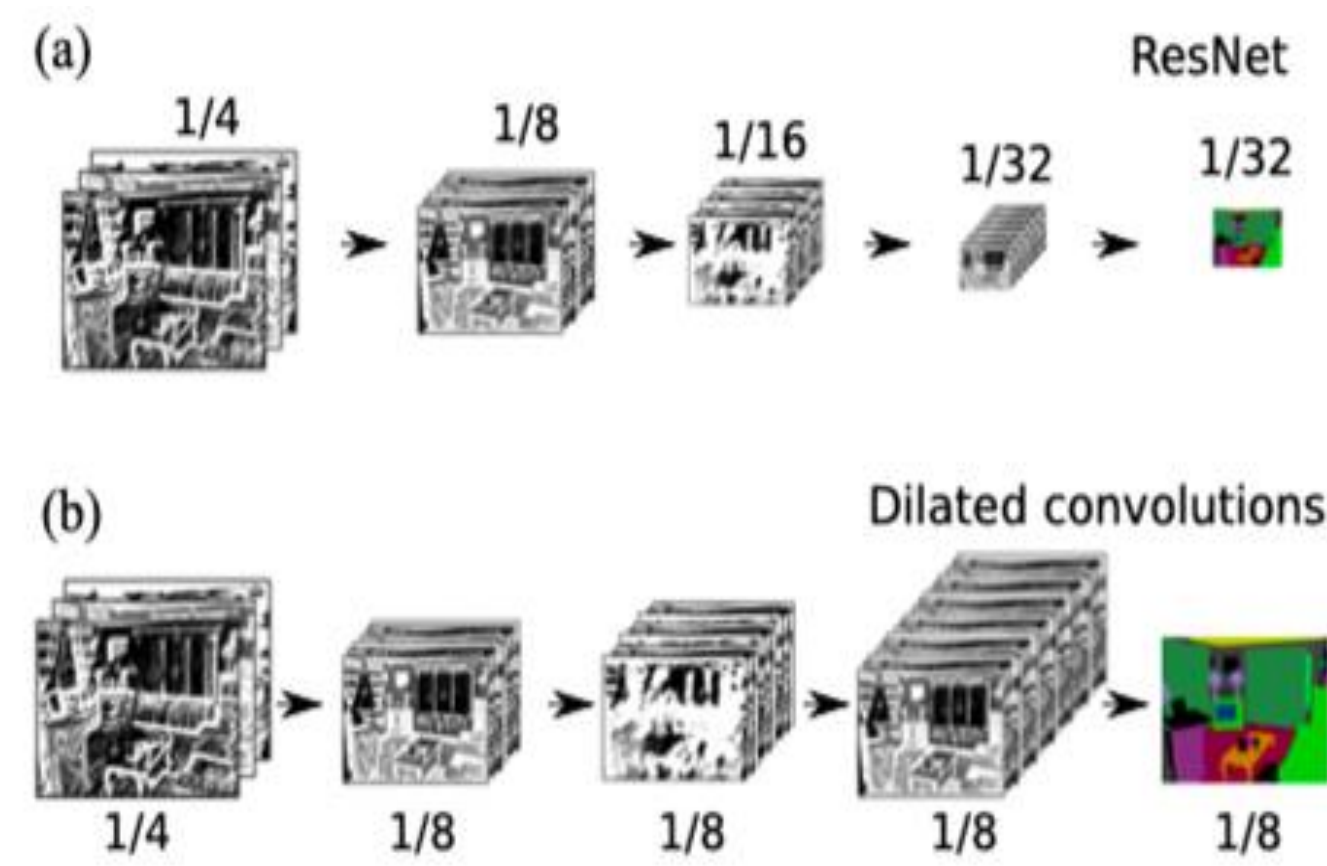
When $l=1$, it is standard convolution.

When $l > 1$, it is dilated convolution.

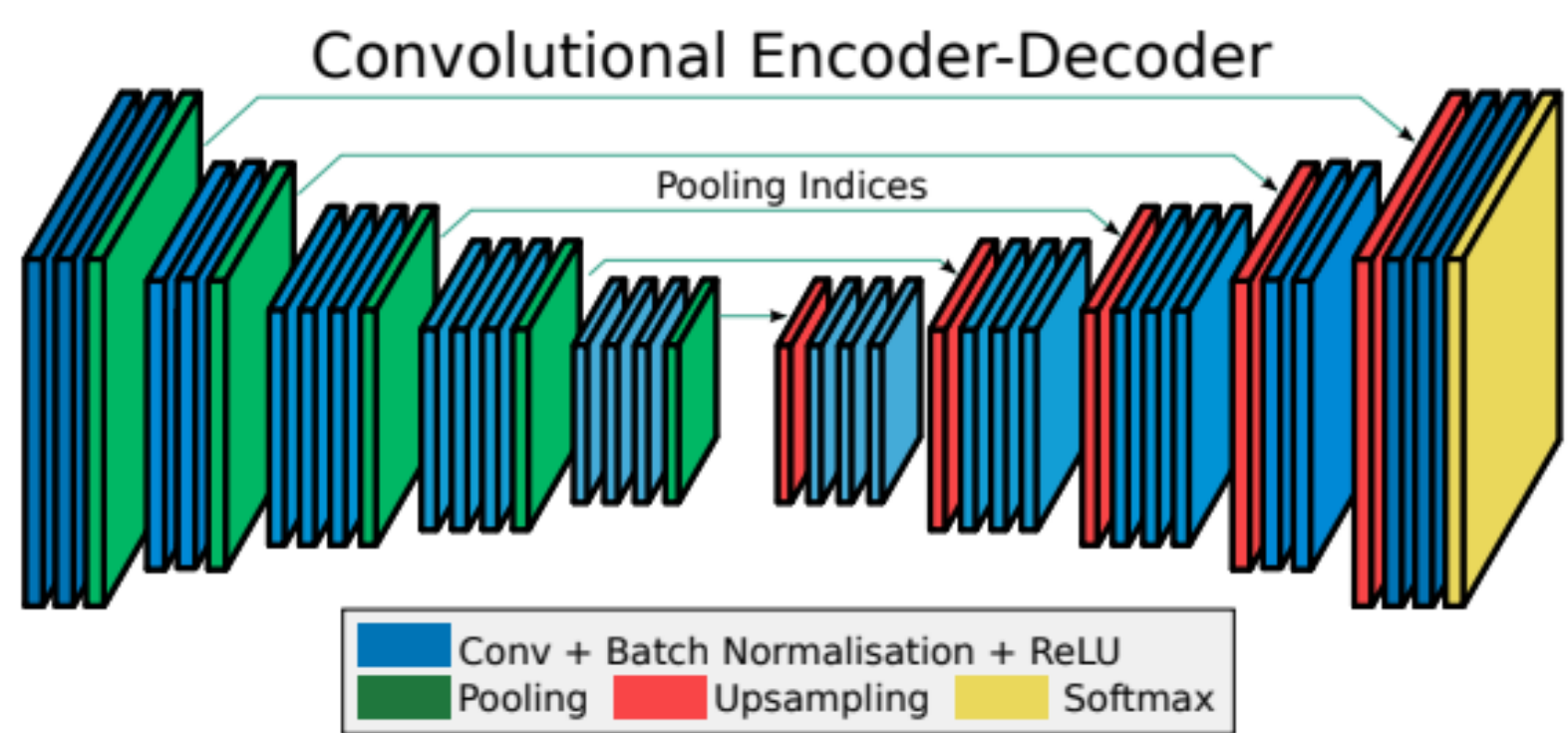
Dilated-FCN include **atrous filters** to **increase prediction resolution** but are **computationally expensive to train** and require much more memory.

Encoder-Decoder was introduced to mitigate the problems of FCN (low resolution) and Dilated-FCN (computationally expensive)

FCN: low resolution. Dilated-FCN: slow

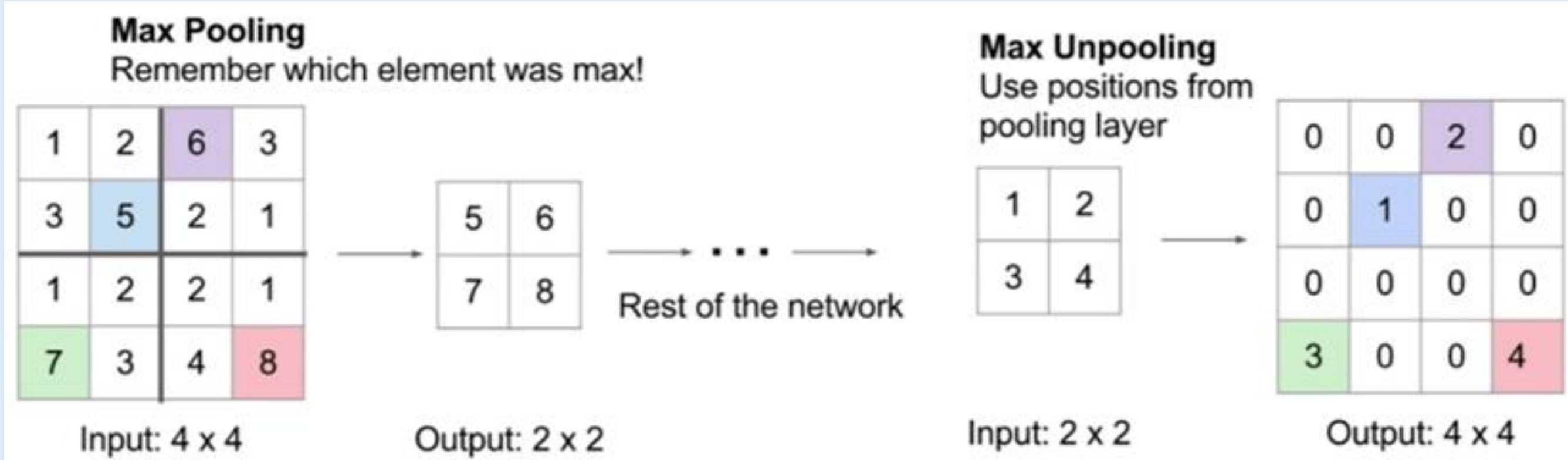


These problems can be mitigated by using a Encoder-Decoder

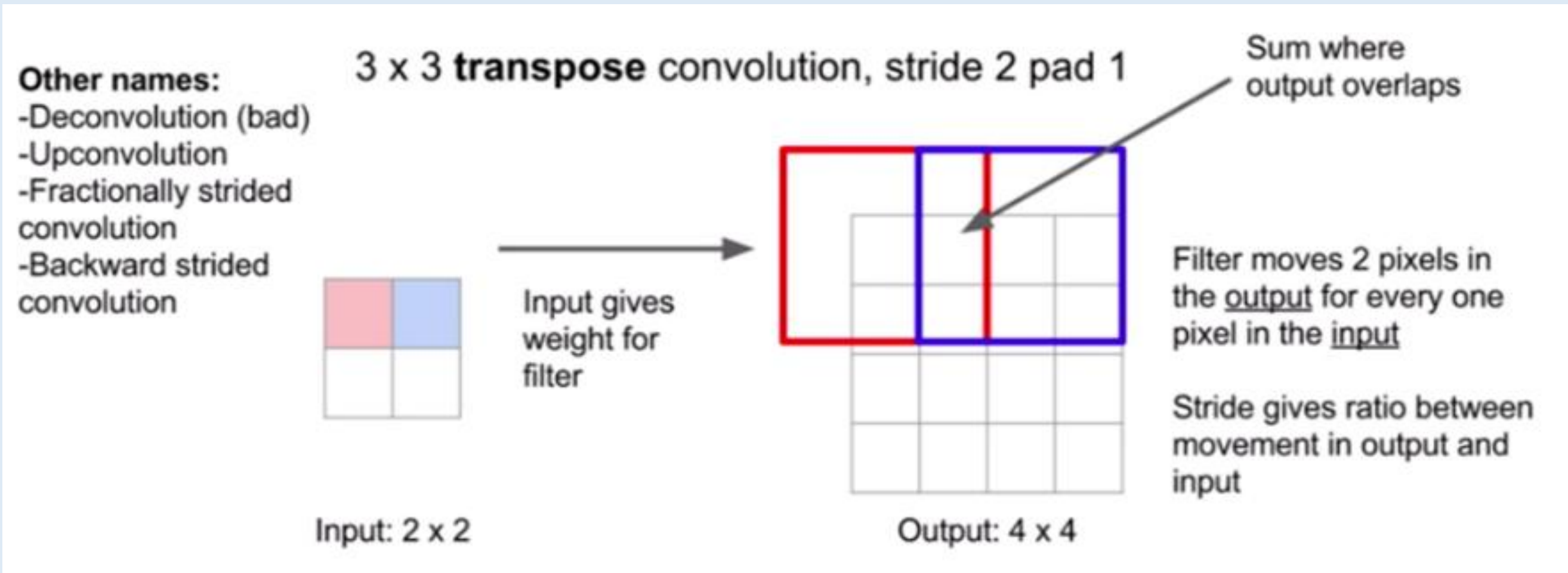


Decoder: Convolutions + Upsampling

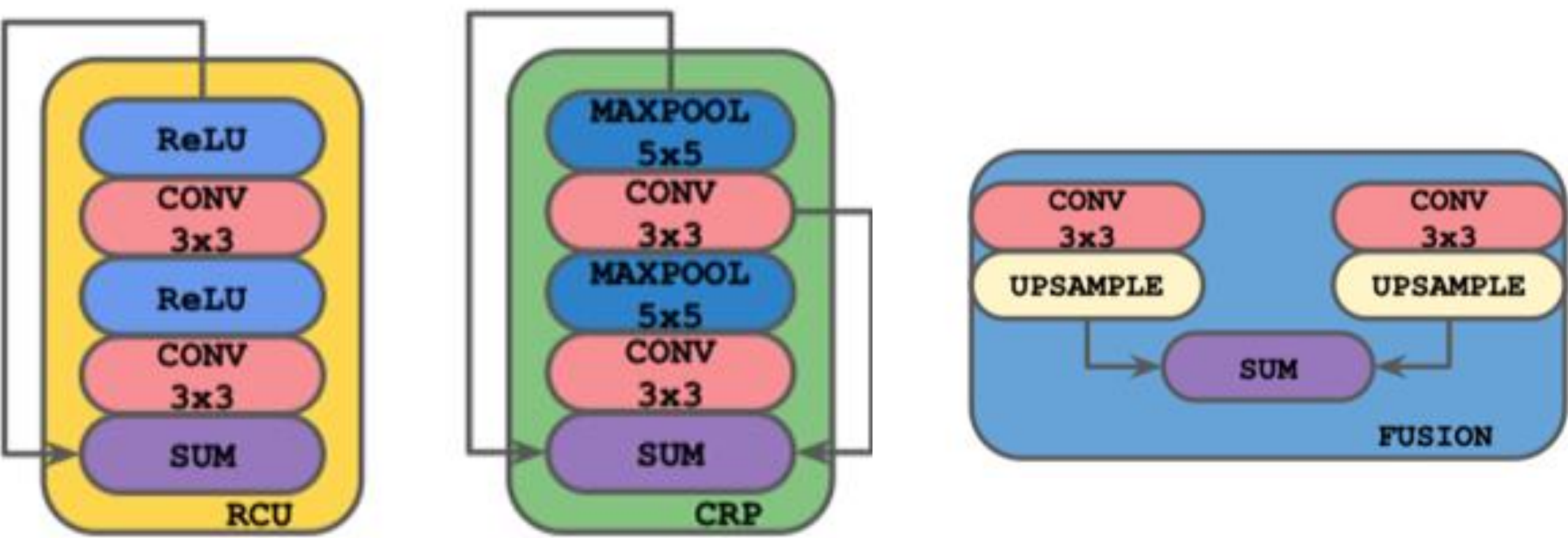
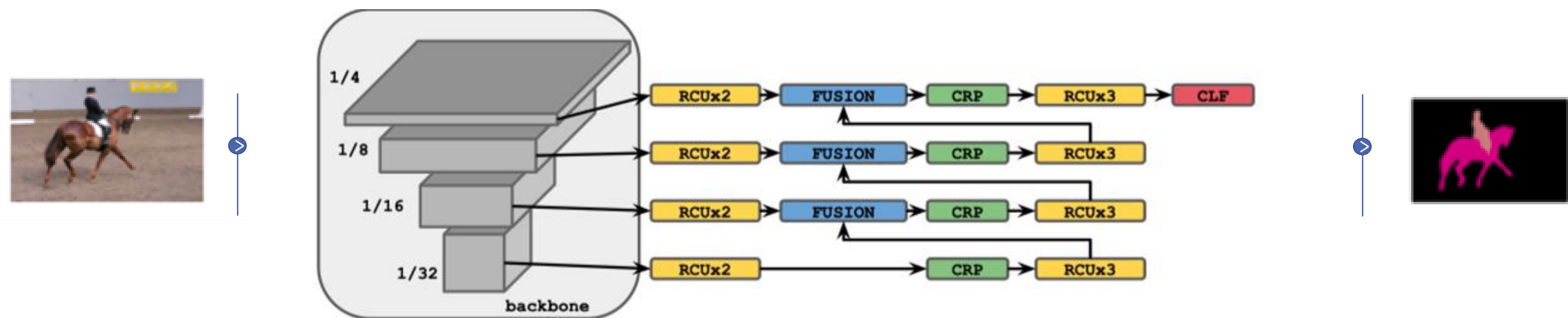
Upsampling: Max Unpooling



Upsampling: Transpose Convolution



RefineNet: EncoderDecoder+ Residual Units



- 1. Long range residual connections
- 2. Multi-path refinement: to fuse coarse high-level semantic features with finer-grained low-level features to generate high-resolution semantic feature maps
- 3. Less memory and parameters than Dilated FCN
- 4. Backbone is compatible for most of the pre-trained models

The models previously explained don't fit well into our use case due to size and speed constraints.

12

Our goal is to find a model that is **fast and small** enough to be used on a **mobile phone**, while maintaining a **good quality**

RefineNet offers a good quality but the size is **more than 200MB**.

Additionally, pre-trained models are trained with classes that are **not need** for our business case.

Smaller setup

1. Replace Conv3x3 to Conv1x1 in CRP and Fusion blocks
2. Remove RCU blocks
3. $\frac{1}{2}$ of parameters remained

Fewer classes

Relabel data to only distinguish **persons** and **background**.

Experiment - LightNet : Pruning the RefineNet

13

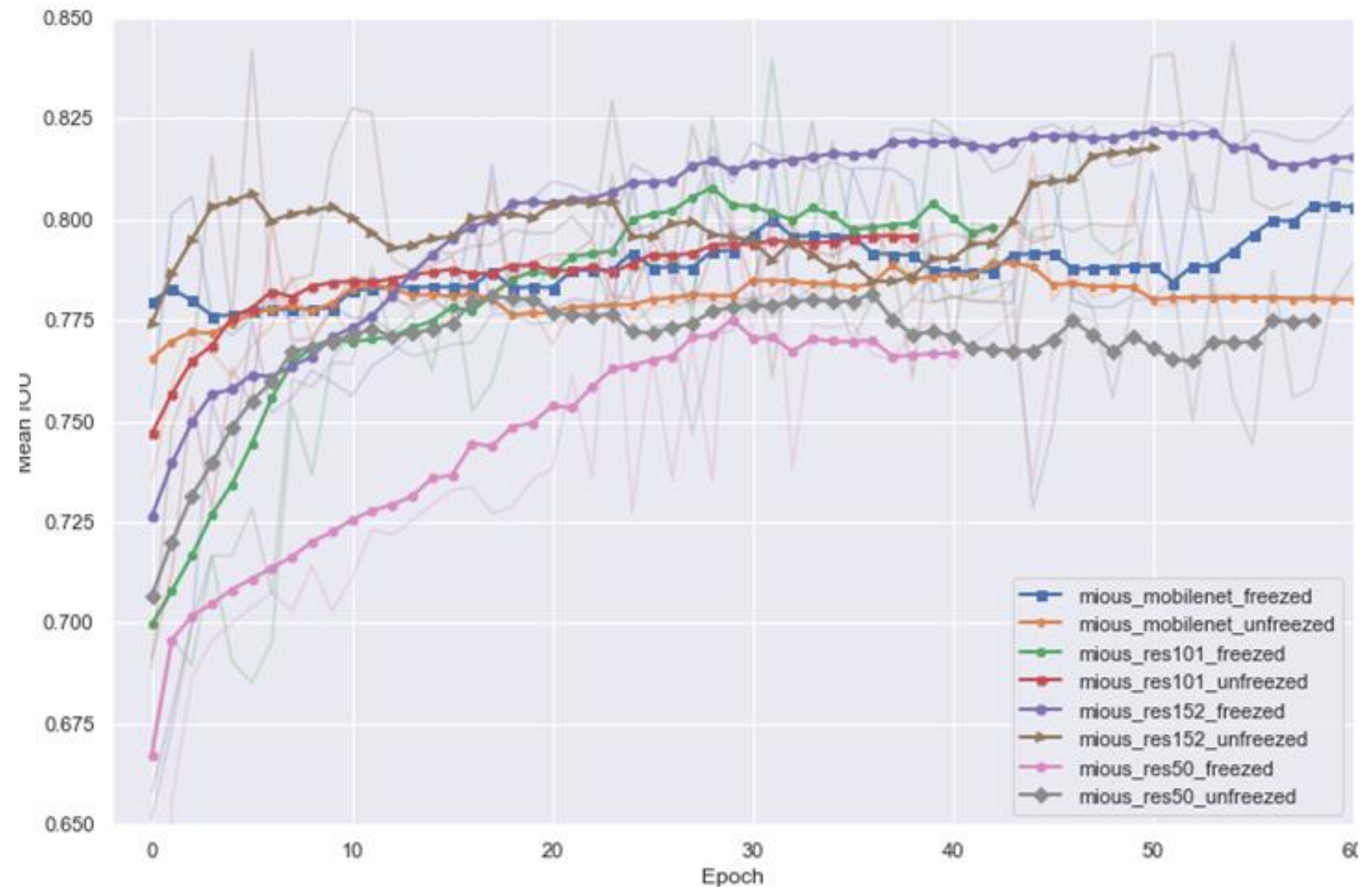
Backbone options

- 1.MobileNetV1
- 2.ResNet50
- 3.ResNet101
- 4.ResNet152

Key findings:

1. Complexity of the model is positive related to its performance
2. Difference in the evolution of mIoU for networks with frozen and unfrozen layers.

mIoU curve for validation set





Experiment - quantitative evaluation on testset

Two metrics:

- 1. mIoU: mean intersection over union (quality)
- 2. Average inference speed per picture

Our goal is to find a model that is **fast** and **small** enough to be used in a mobile phone, while maintaining a **good quality**. We choose **LightNet-MobileNet** because it offers the better balance between the three components.

 Model to benchmark  Model used in our app

| Models | mIoU | Average speed (s) | Size (MB) |
|--|--------------|-------------------|-----------|
|  LightNet-MobileNet | 0.809 | 0.02334 | 13 |
| LightNet-MobileNet-frozen | 0.808 | 0.02247 | 13 |
| LightNet-ResNet-50 | 0.813 | 0.02386 | 104 |
| LightNet-ResNet-50-frozen | 0.815 | 0.02254 | 104 |
| LightNet-ResNet-101 | 0.776 | 0.03978 | 177 |
| LightNet-ResNet-101-frozen | 0.847 | 0.03860 | 177 |
| LightNet-ResNet-152 | 0.859 | 0.05156 | 237 |
| LightNet-ResNet-152-frozen | 0.805 | 0.05128 | 237 |
|  DeepLabV3-ResNet-101 | 0.806 | 0.0388 | 233 |

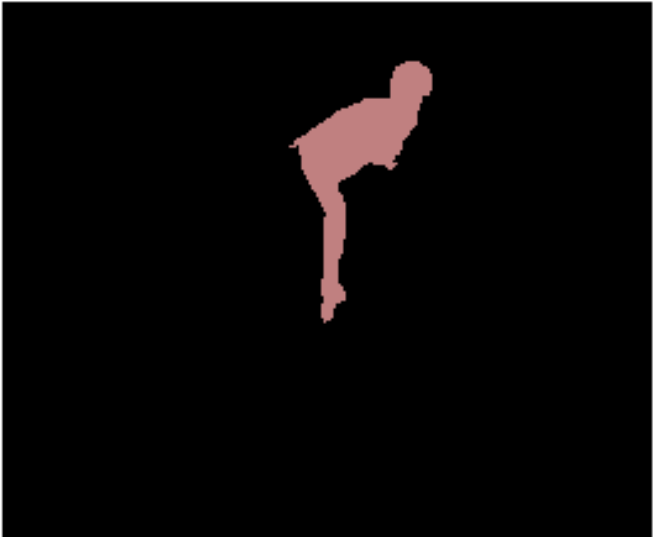
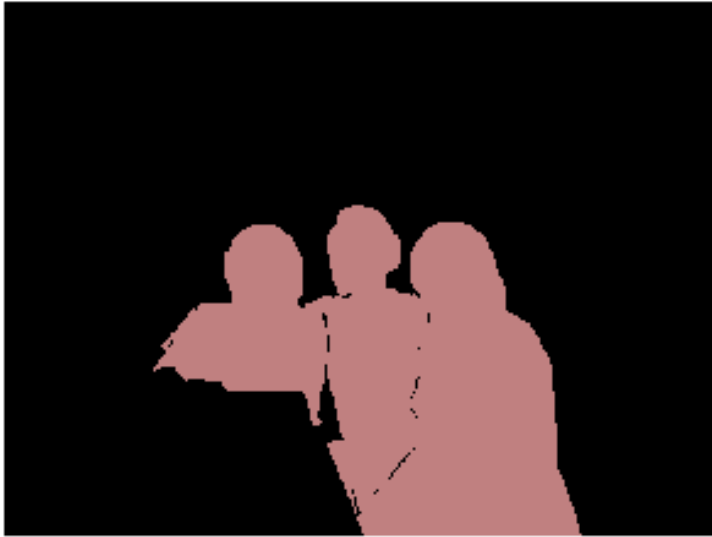
Test set: 104 pictures

Experiment - qualitative evaluation on testset

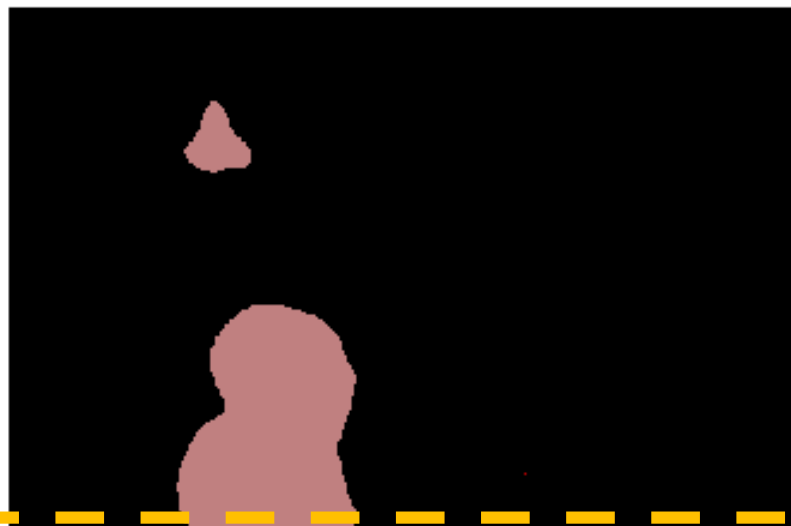
Image



Ground truth



DeepLabv3



LightNet-MobileNet (our)



After detecting the different components of the picture we apply Neural Style Transfer

16

Neural style transfer is an optimization technique used to take two images: a **content image** and a **style reference image** and **blend them together** so the output image looks like the content image, but “painted” in the style of the style reference image.

The principle is to define **two distance functions**, one that describes how different the content of two images are, L_{content} , and one that describes the **difference between the two images in terms of their style**, L_{style} .

We'll transform the base input image by **minimizing the content and style distances (losses) with backpropagation**, creating an image that matches the content of the content image and the style of the style image.

$$L_{\text{content}} = \frac{1}{2} \sum_{i,j} (A_{ij}^l(g) - A_{ij}^l(c))^2$$

$$L_{\text{style}} = \sum_l w^l L_{\text{style}}^l \text{ where,}$$

$$L_{\text{style}}^l = \frac{1}{M^l} \sum_{ij} (G_{ij}^l(s) - G_{ij}^l(g))^2 \text{ where,}$$

$$G_{ij}^l(I) = \sum_k A_{ik}^l(I) A_{jk}^l(I).$$



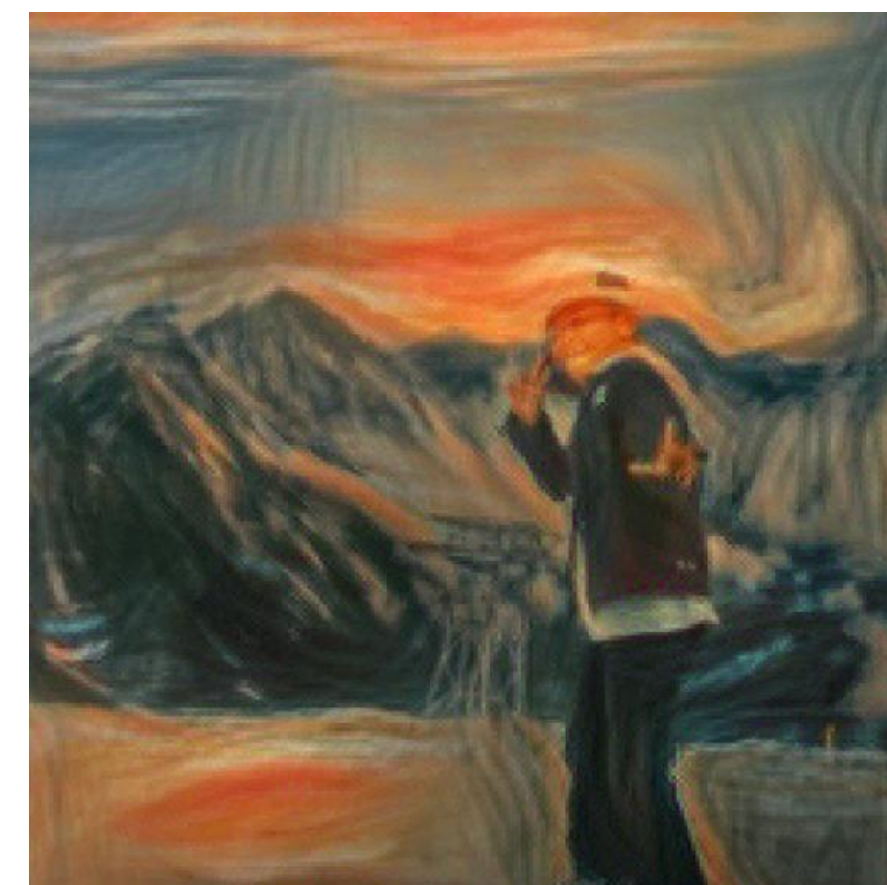
Content picture

+



Style picture

>



Output picture

Conclusions:

1. We identified the business opportunity and built a prototype to demonstrate the feasibility
2. Experiment shows the **small model can perform as well as big model** (237MB) while keeping size small (13Mb) and **increase the inference speed.**

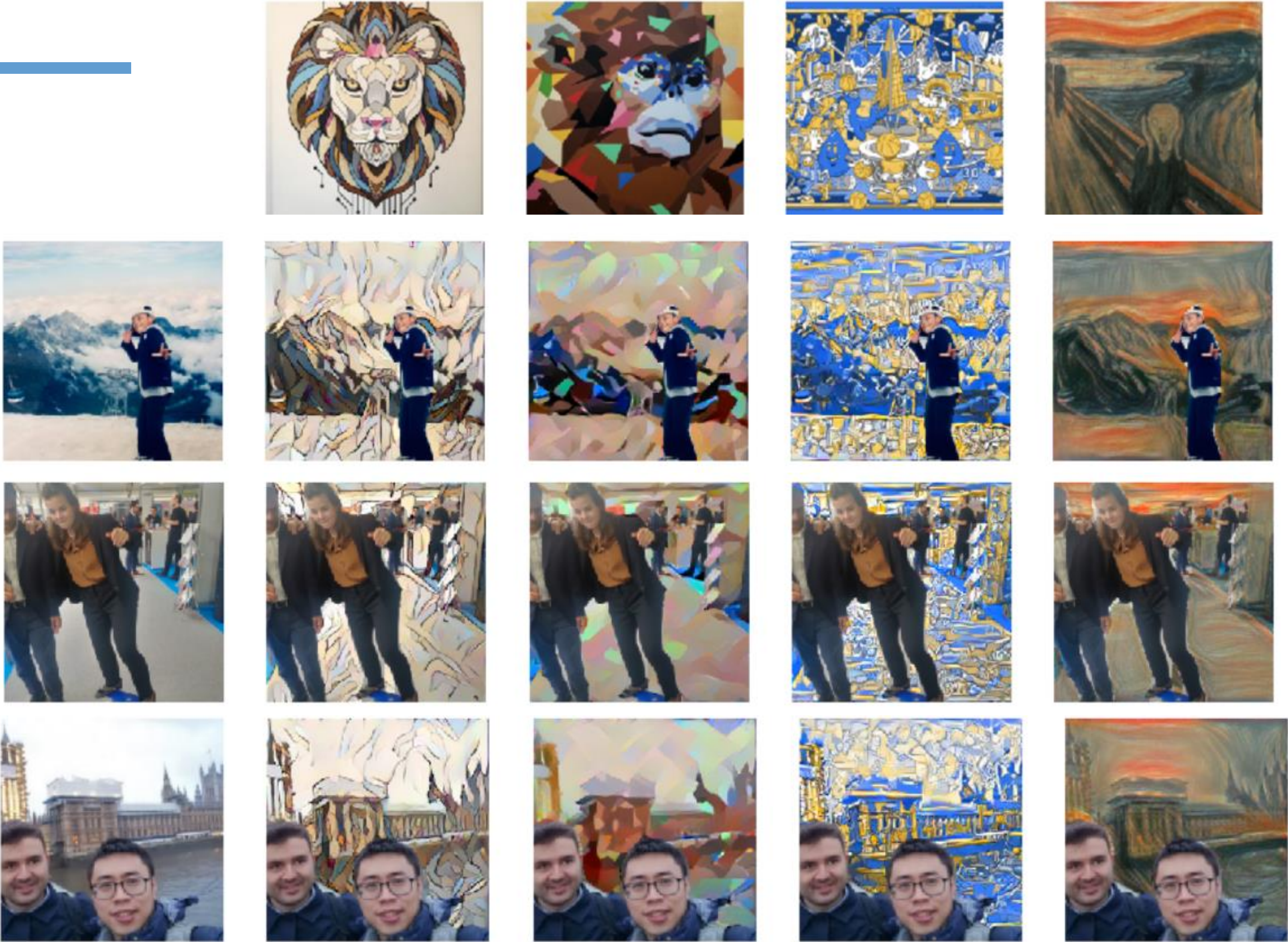
Next steps:

1. **Instance segmentation:** allows users to have finer image editing options.
2. **Neural style transfer:** design a light weight architecture to reduce the pipeline inference speed to have better user experience.



More examples

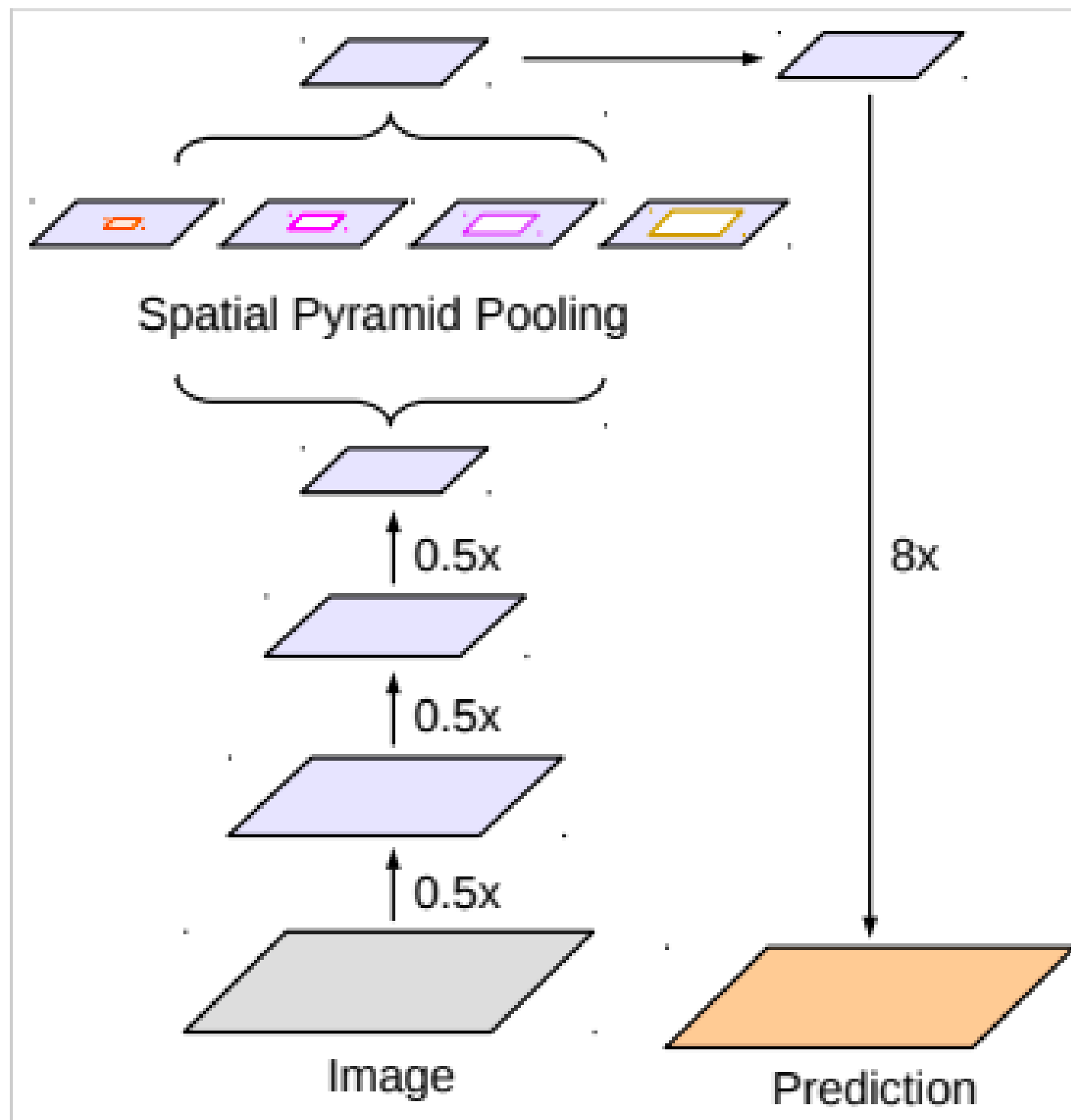
DeepPhoto



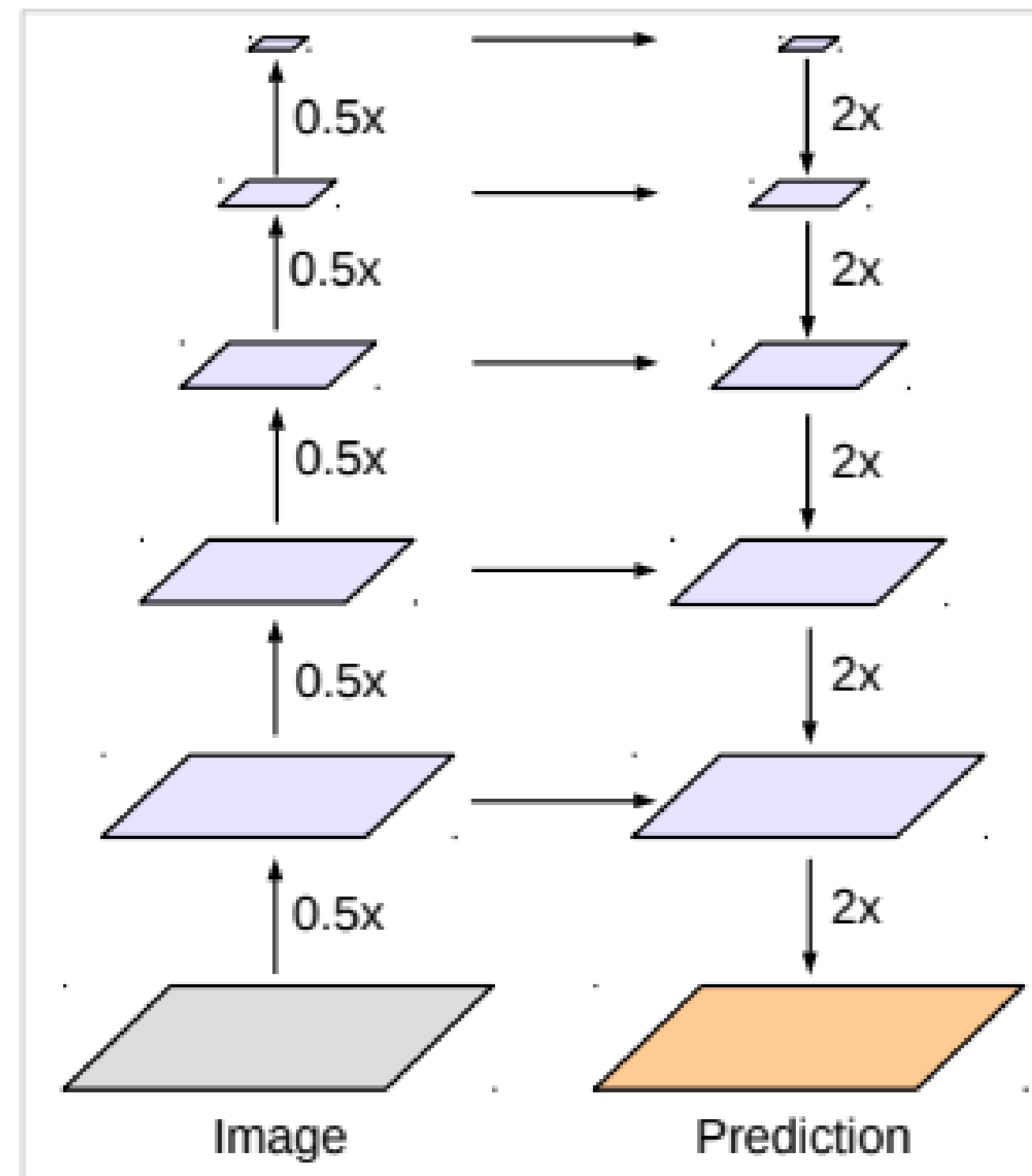
Note. All pictures has been taken by Akshay Sundar. All rights reserved. Copyright own by the author. More info at akshay.sundar@hec.edu.

Appendix – DeepLabV3 architecture

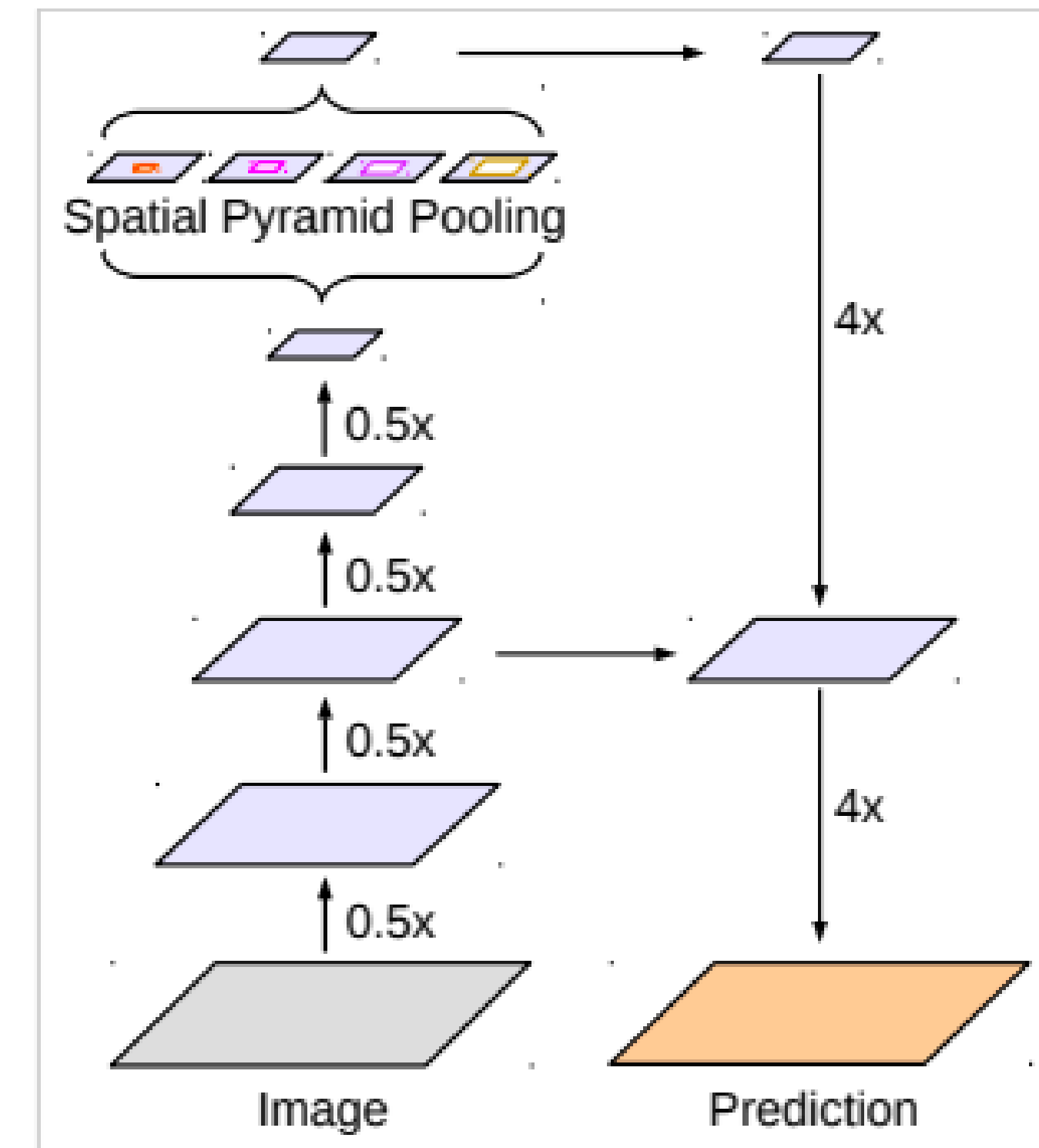
19



(a) Spatial Pyramid Pooling



(b) Encoder-Decoder



(c) Encoder-Decoder with Atrous Conv