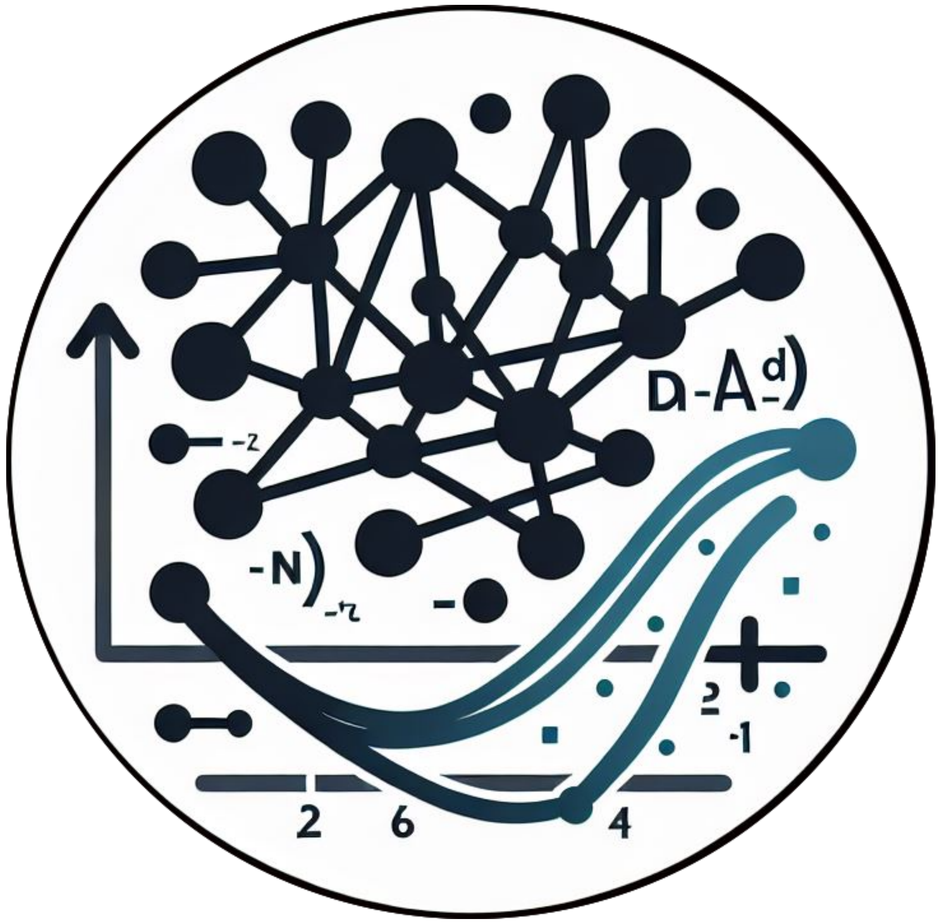
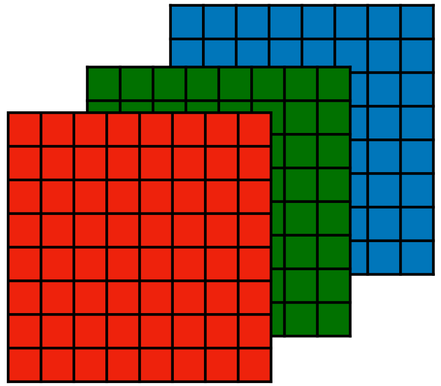
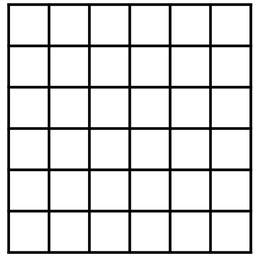
Convolutional Neural Nets (continued)



Deep Learning for Engineers
Andrew Ning
aning@byu.edu

1x1 convolution





8 x 8 x 3

1 x 1 x 3 x 1

8 x 8 x 1

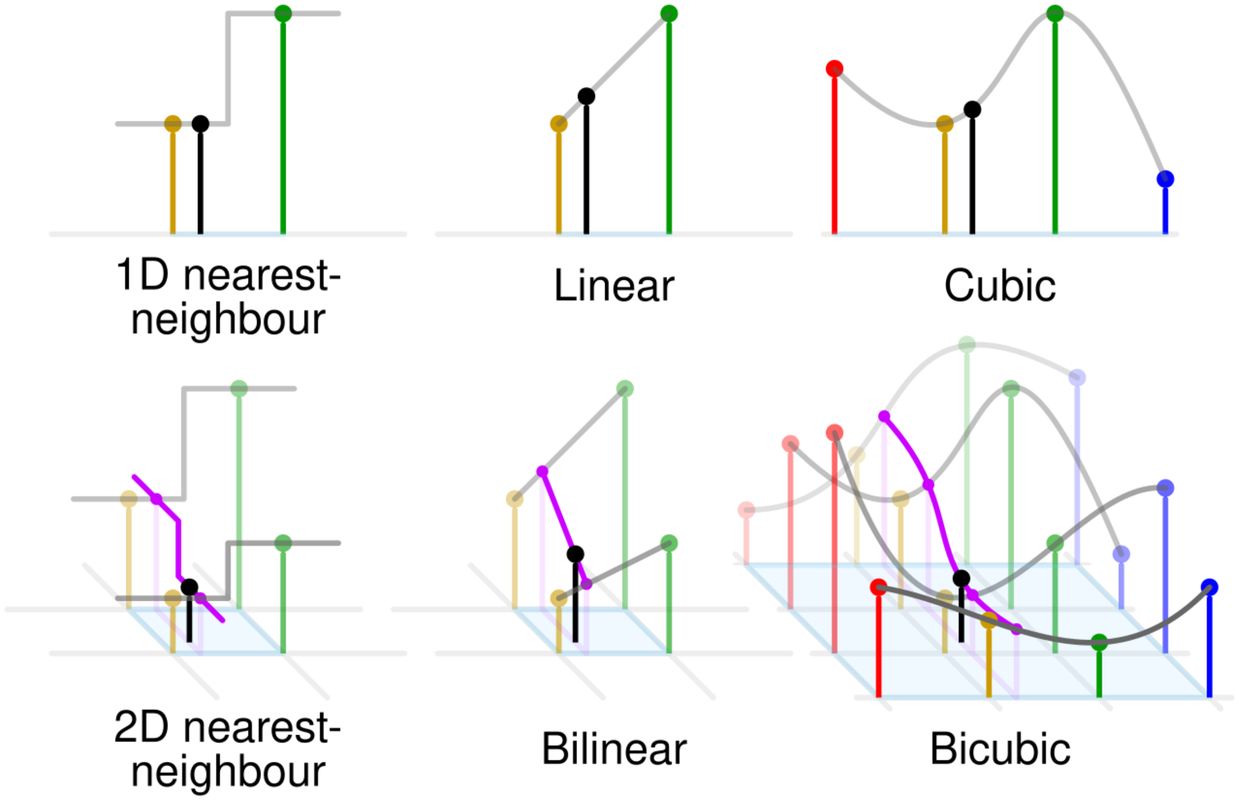
in channels

kernel size

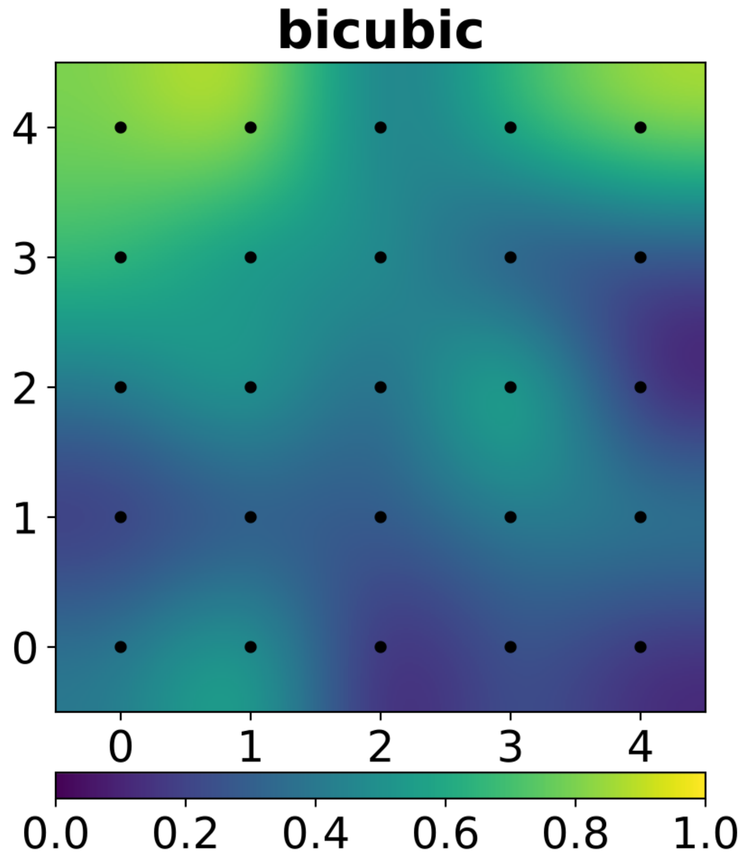
out channels

```
nn.Conv2d(in_channels=3, out_channels=1, kernel_size=1)
```

Upsample (bicubic interpolation)

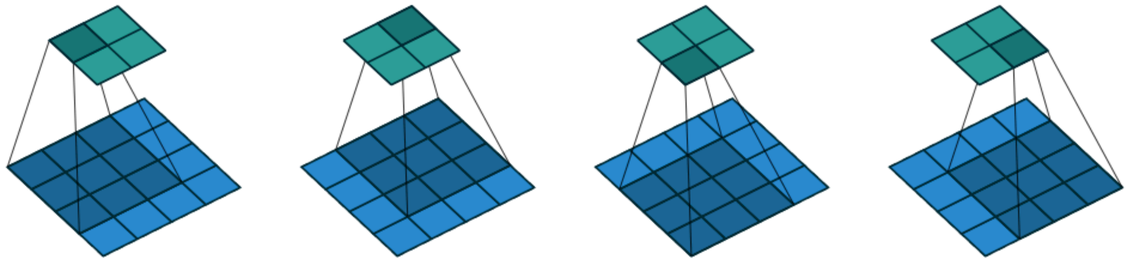


Cmglee, Wikimedia Commons



Zykure, Wikimedia Commons

Upsample (Transpose Convolution)



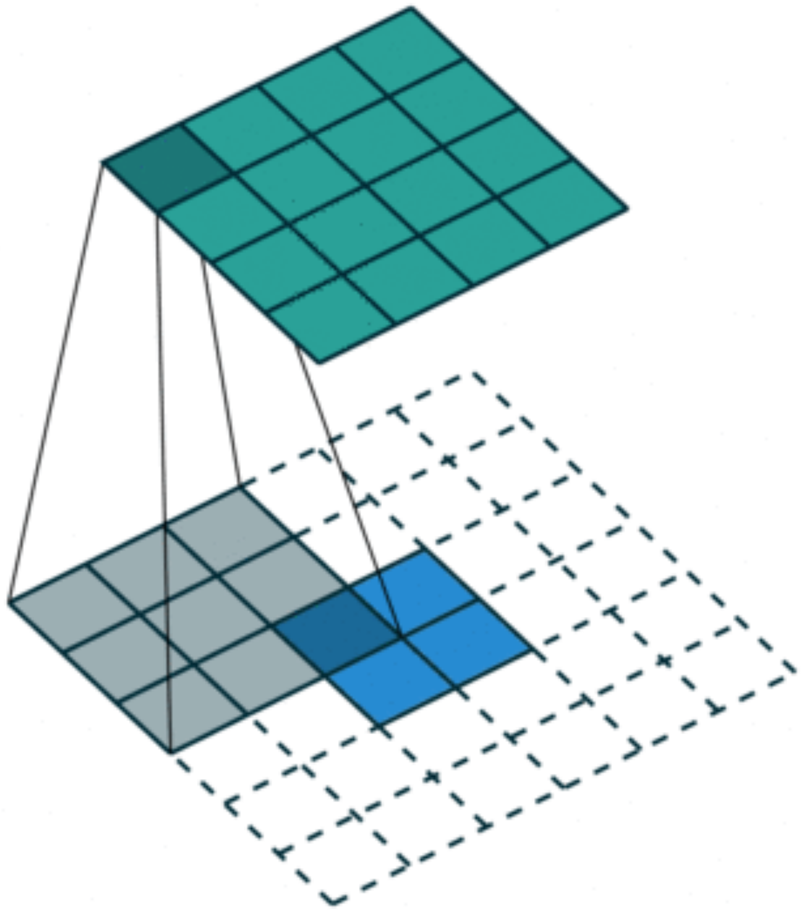
$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

A guide to convolution arithmetic for deep learning, Dumoulin and Visinn

convolution: 16 dim input (4 x 4) → 4 dimensional output (2x2)

transpose conv: 4 dimensional input (2x2) → 16 dim output (4 x 4)

Transpose Convolution



output

kernel

input

Transpose Convolution

input

1	0
2	2

kernel

1	3
0	2

=

output

1	3	
0	2	

+

	0	0
	0	0

+

2	6	
0	4	

+

	2	6
	0	4

=

1	3	0
2	10	6
0	4	4

Sizing

$$H_{out} = (H_{in} - 1) \times \text{stride} + \text{kernel}$$

input

1	0
2	2

kernel

1	3
0	2

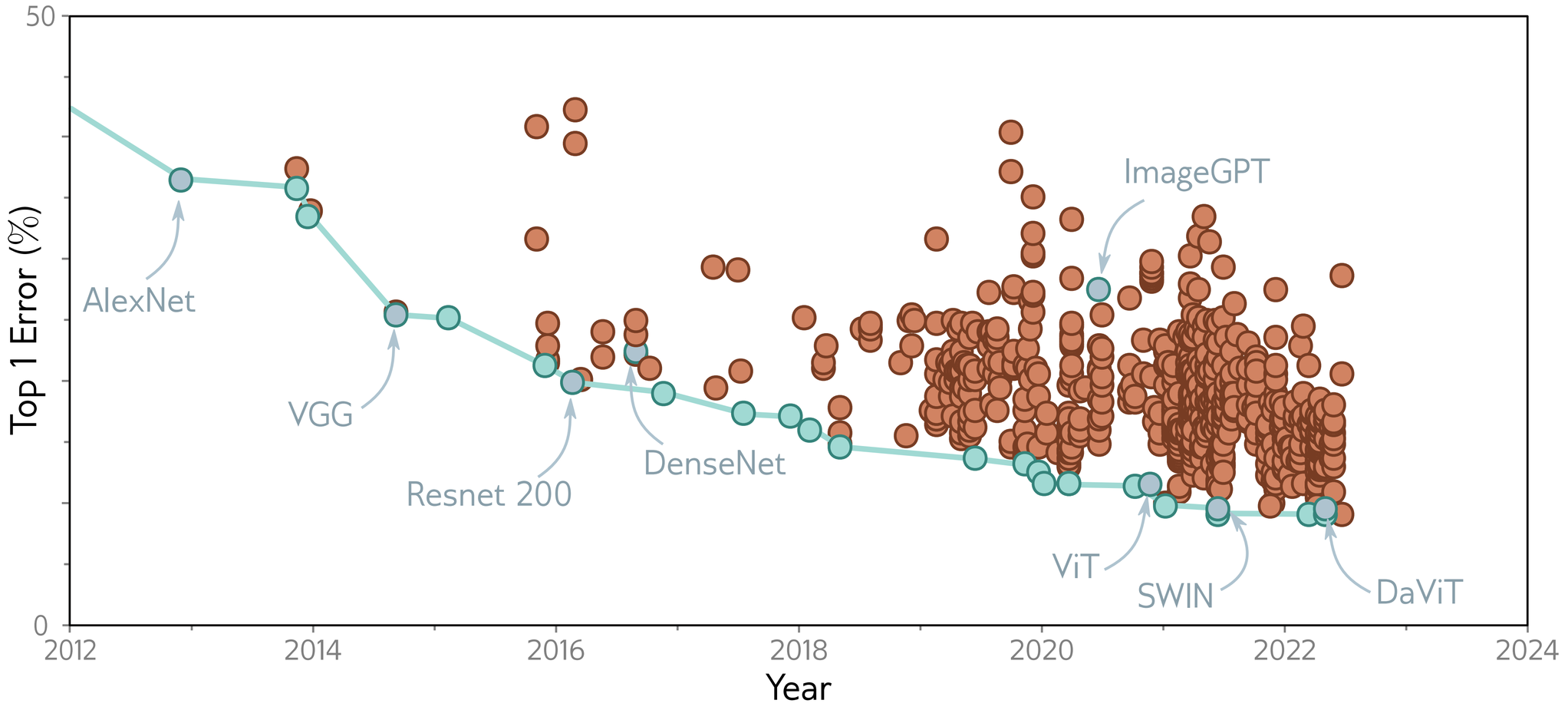
output

1	3	0
2	10	6
0	4	4

torch

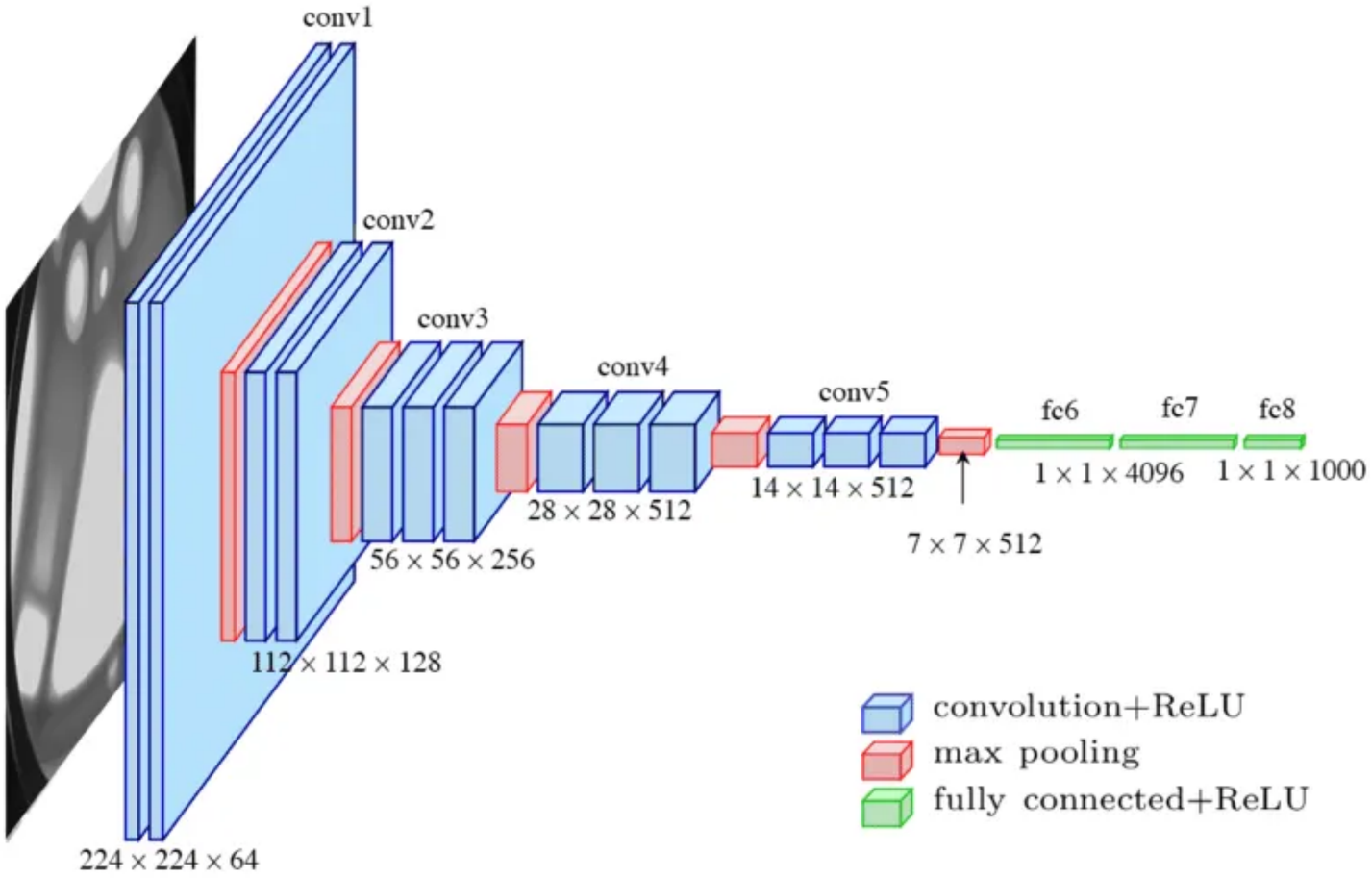
```
nn.ConvTranspose2d(in_channels, out_channels,
    kernel_size, stride=1, padding=0)
```

ImageNet

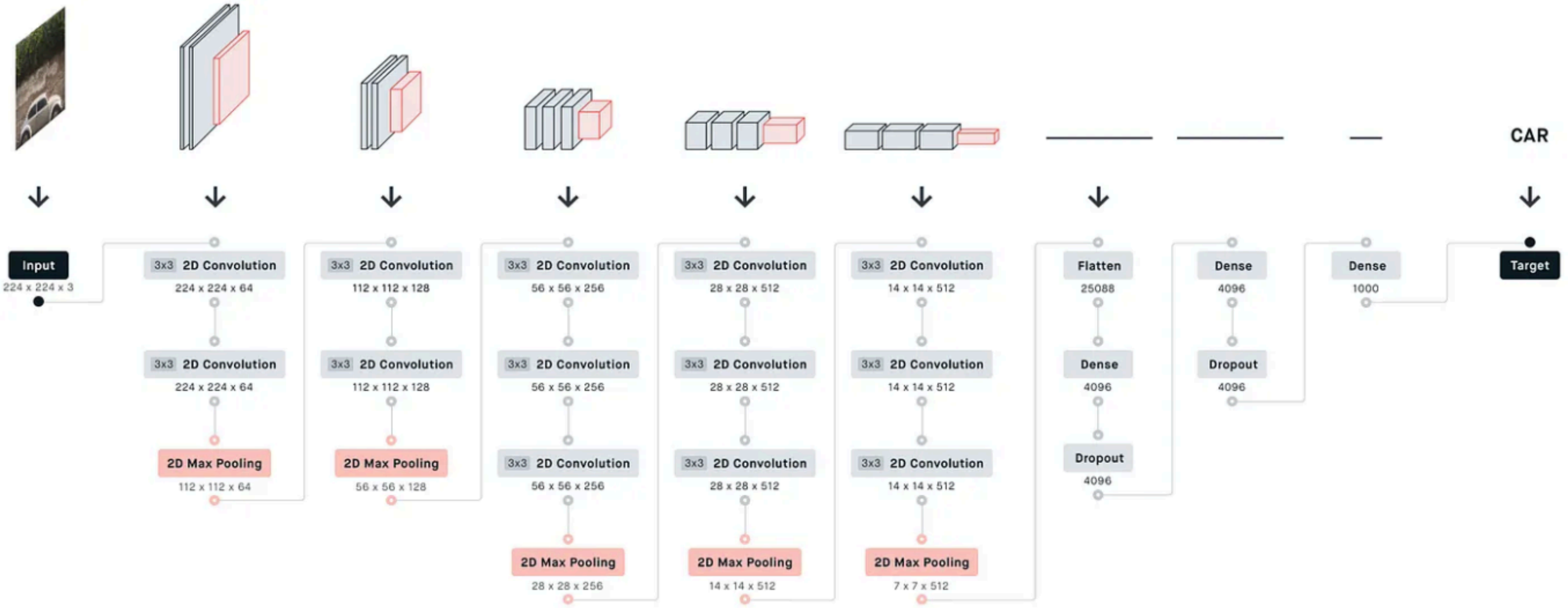


Understanding Deep Learning, Simon J. D. Prince, 2023

VGG



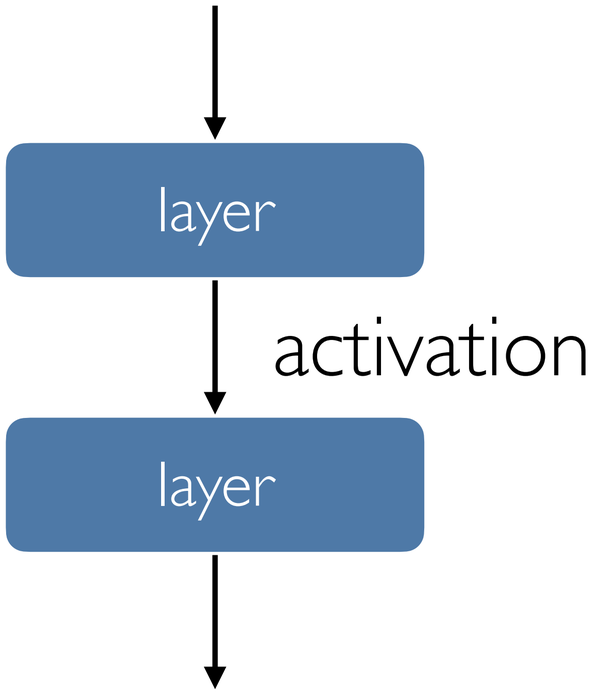
VGG-Net Architecture Explained, Siddhesh Bangar

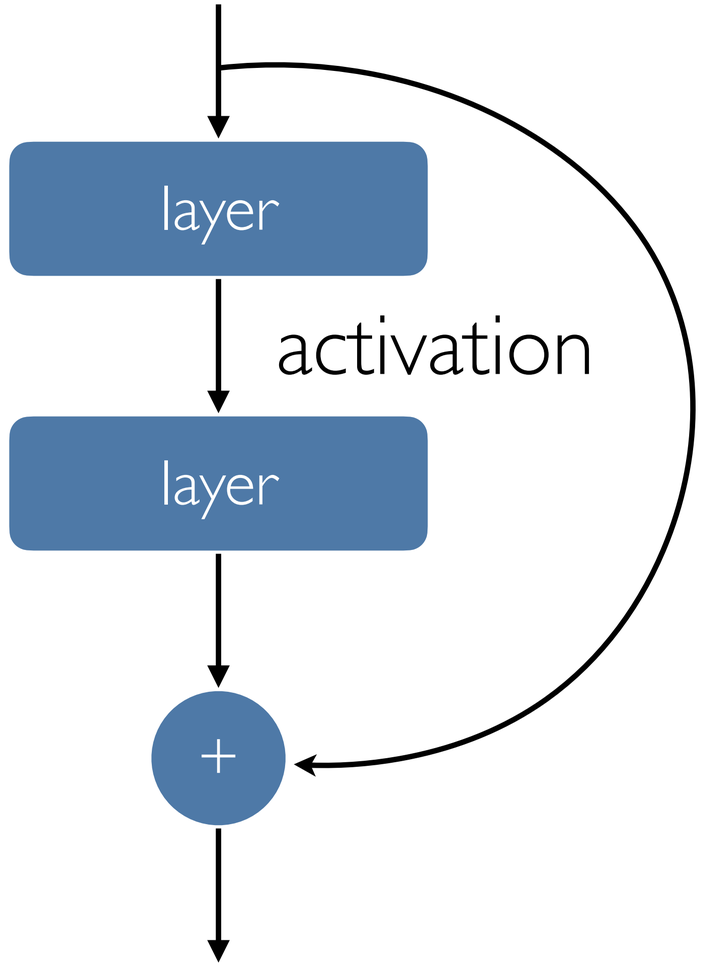


VGG 16 Architecture

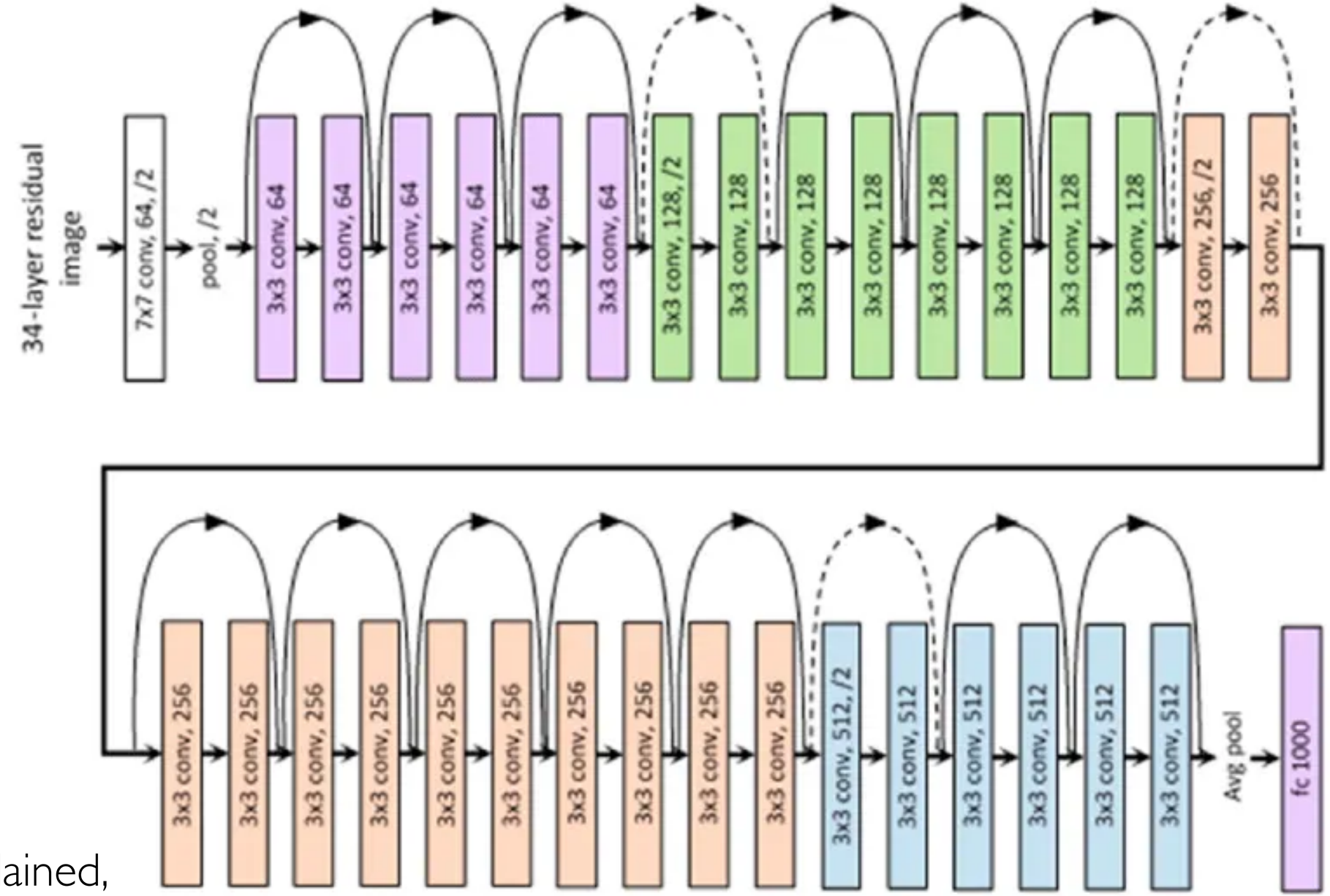
VGG-Net Architecture Explained, Siddhesh Bangar

ResNet

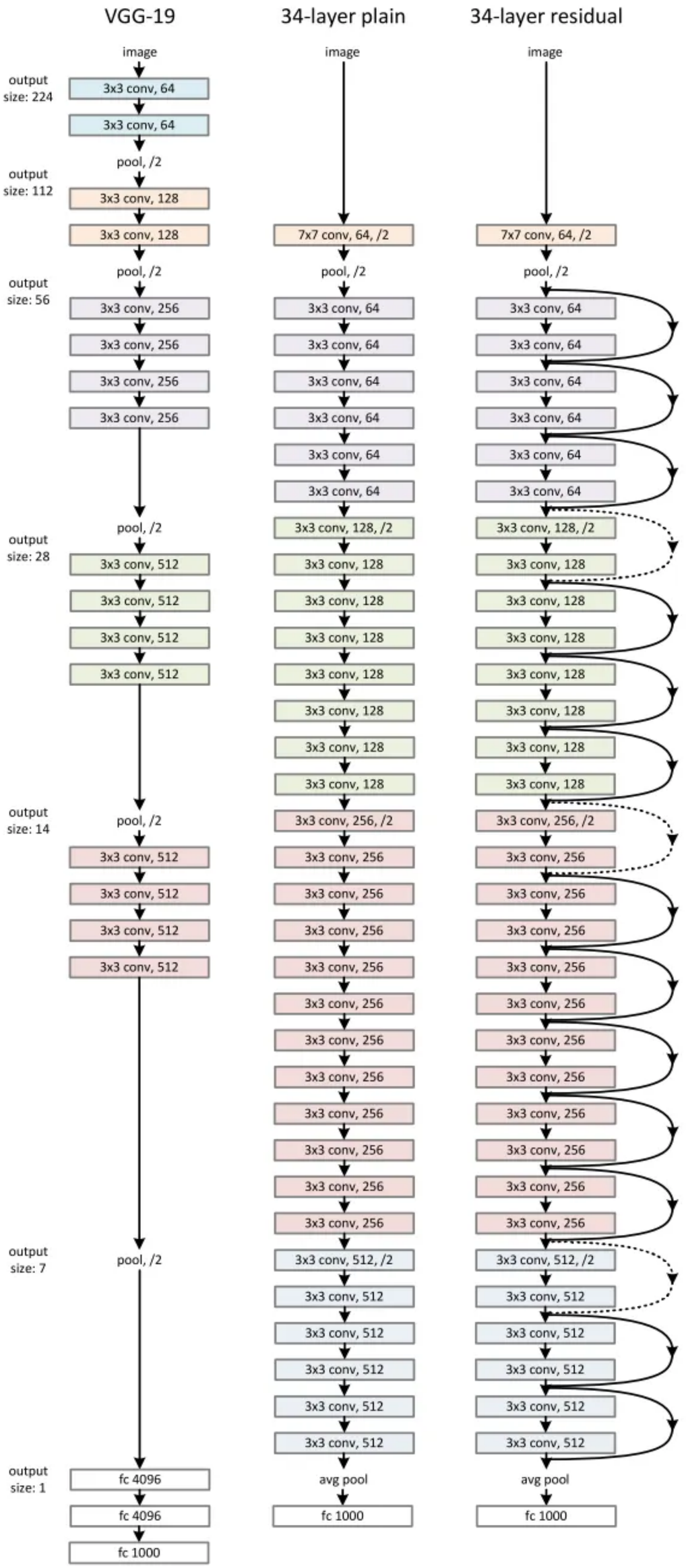




ResNet-34

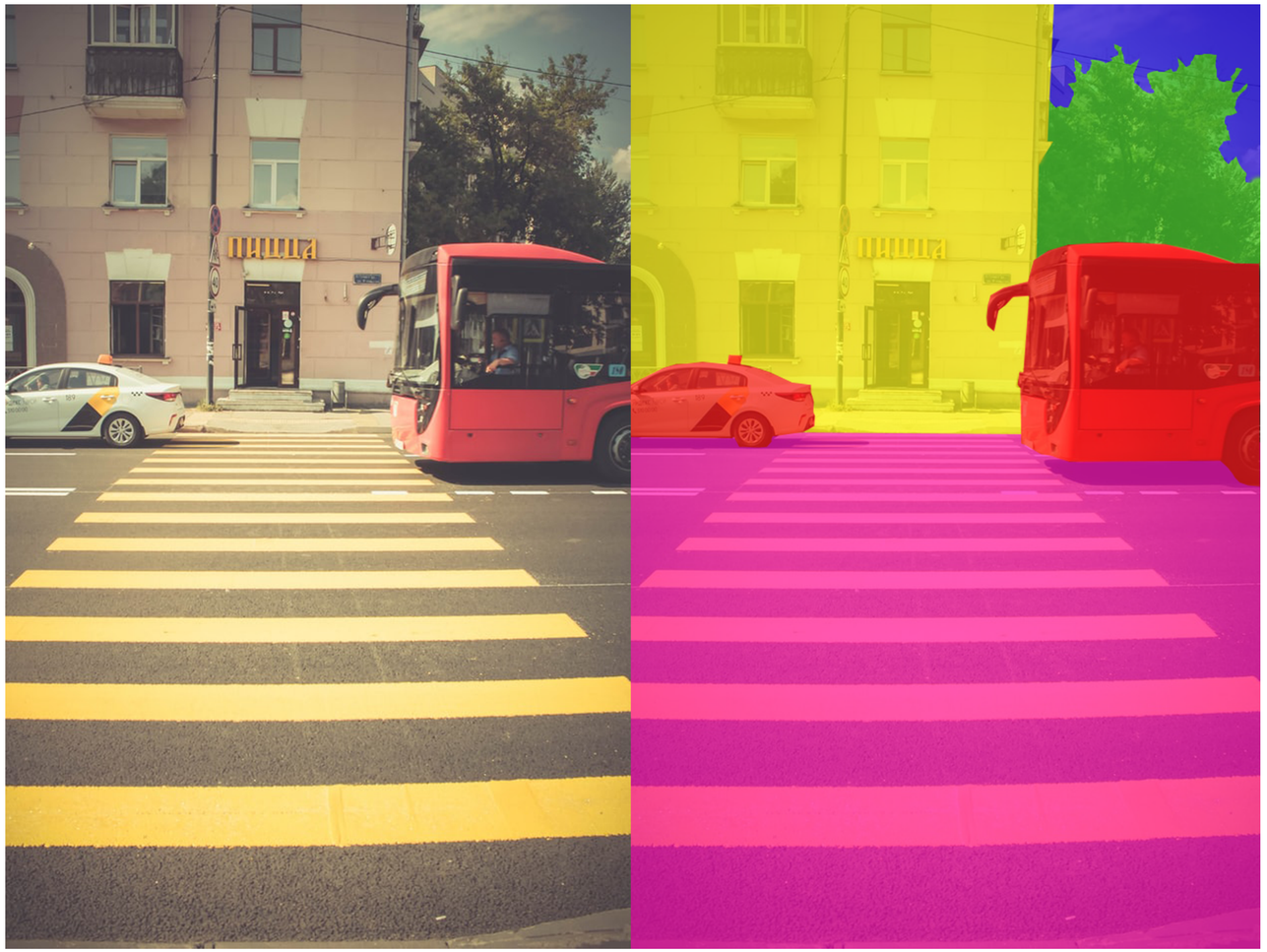


Resnet Architecture Explained,
Siddhesh Bangar

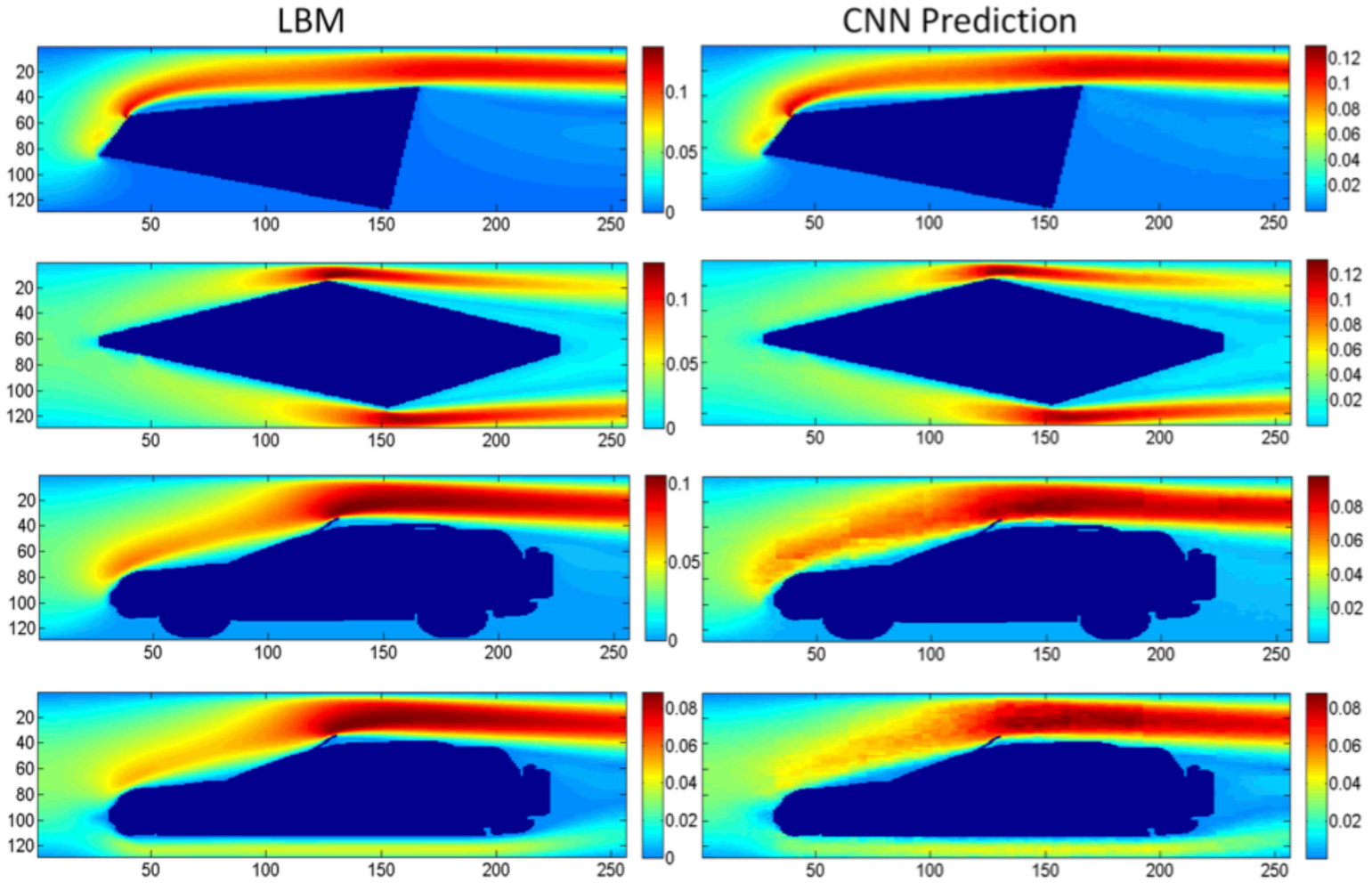


Resnet Architecture Explained, Siddhesh Bangar

Image Segmentation

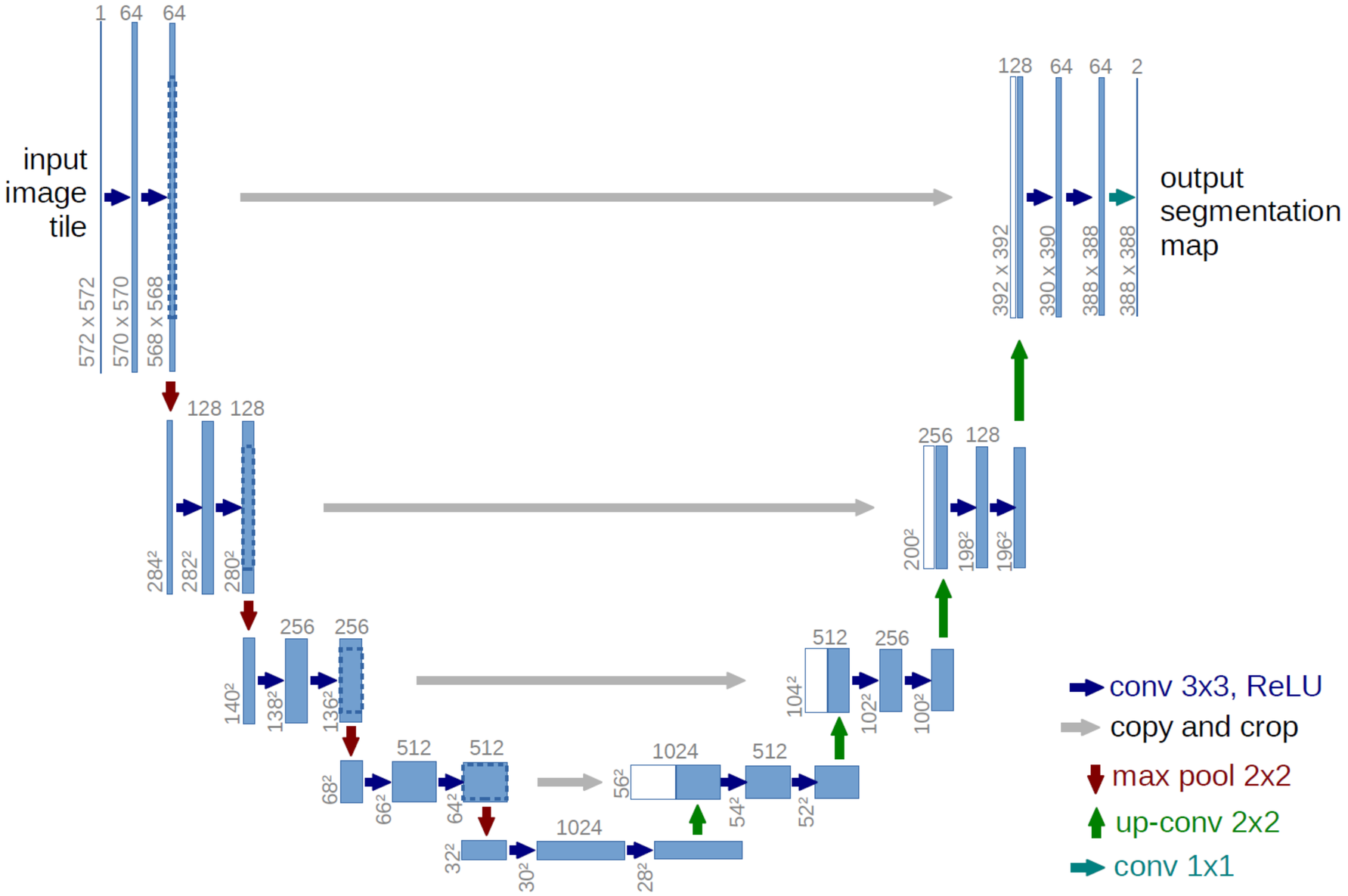


B. Palac, Wikimedia Commons

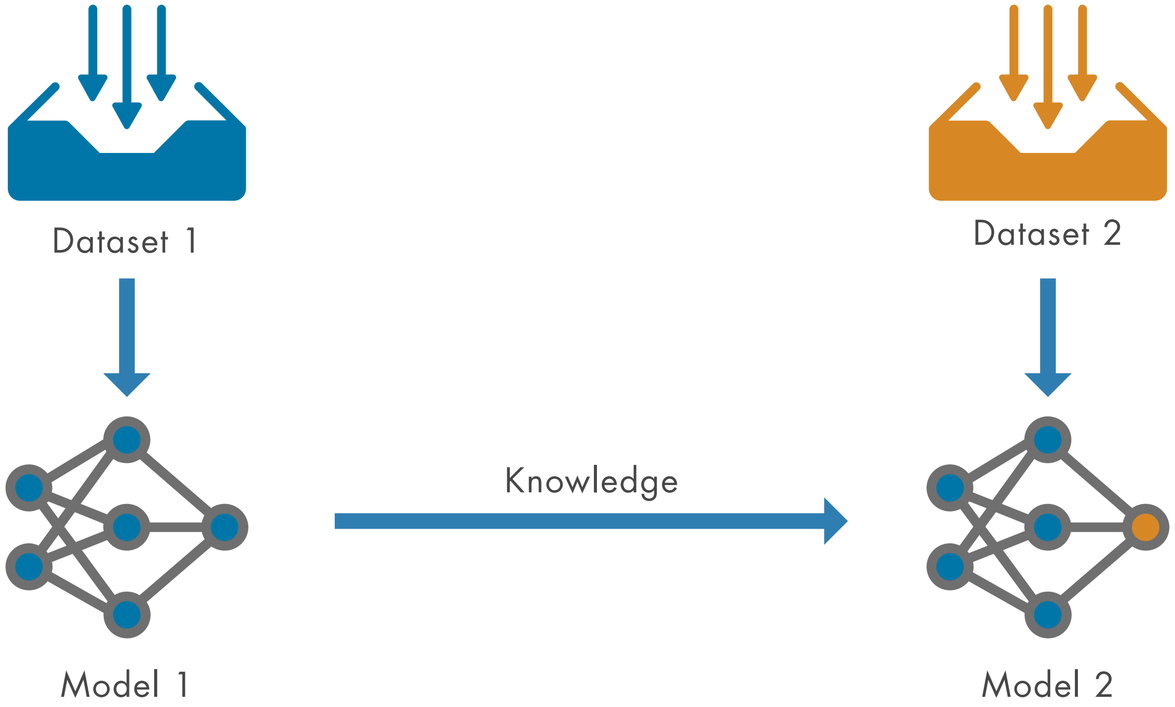


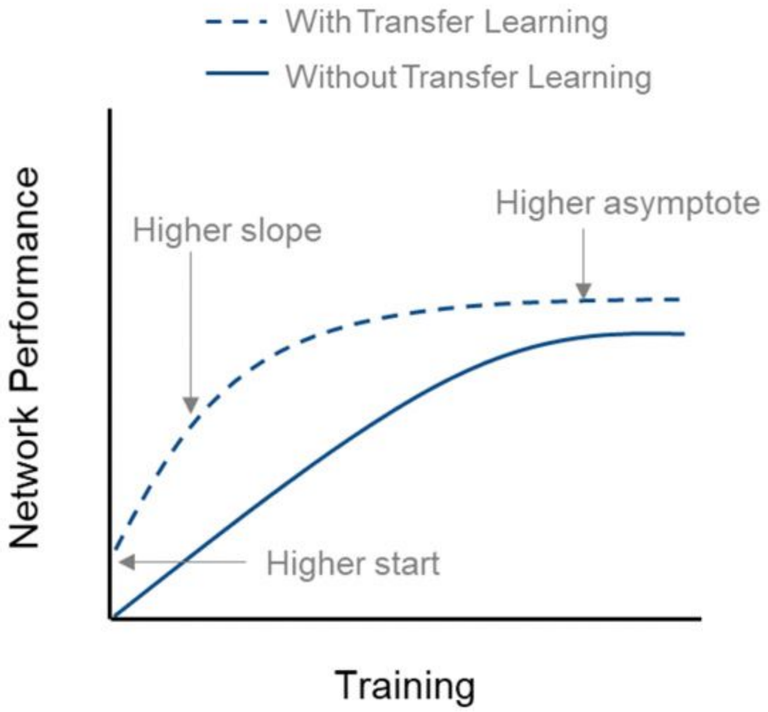
Convolutional Neural Networks for Steady Flow Approximation, Xiaoxiao Guo, Wei Li, Francesco Iorio

U-Net


input image tile
572 x 572
570 x 570
568 x 568
1 64 64
128 128
284²
282²
280²
256 256
140²
138²
136²
512 512
68²
66²
64²
1024
32²
30²
28²
1024 512
56²
54²
512 256
104²
102²
100²
256 128
200²
198²
196²
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

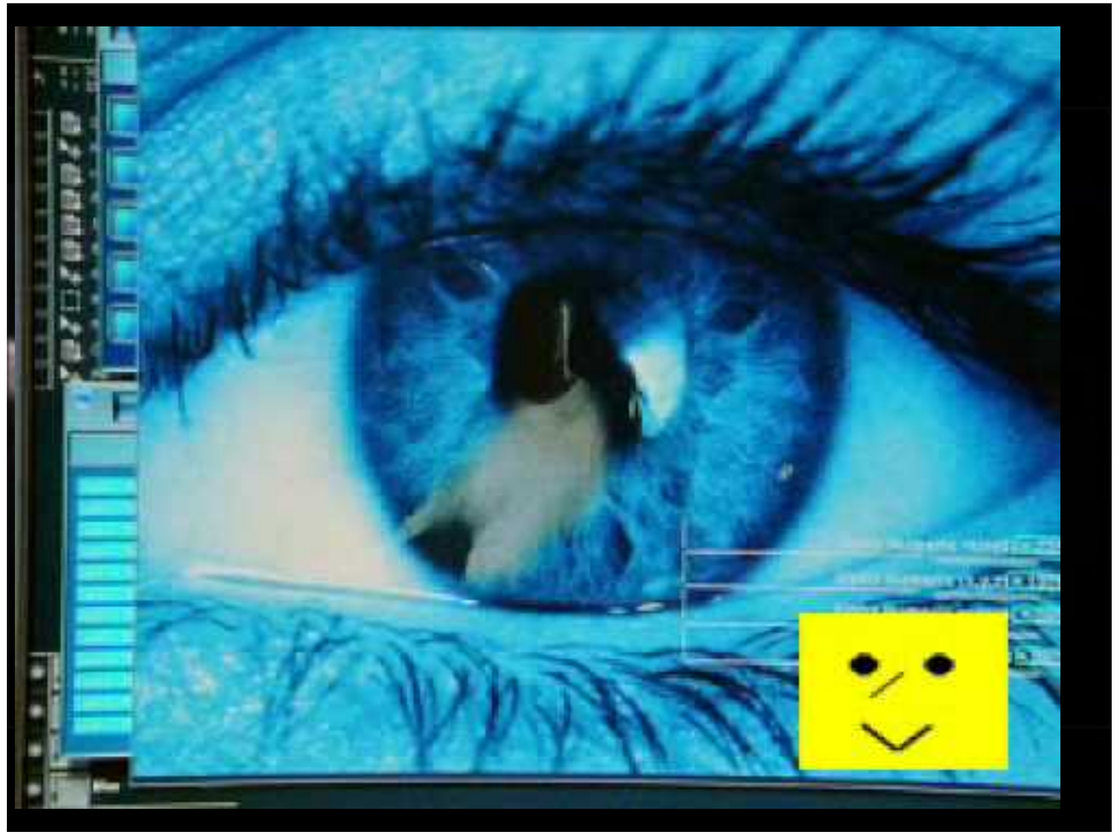
Transfer Learning



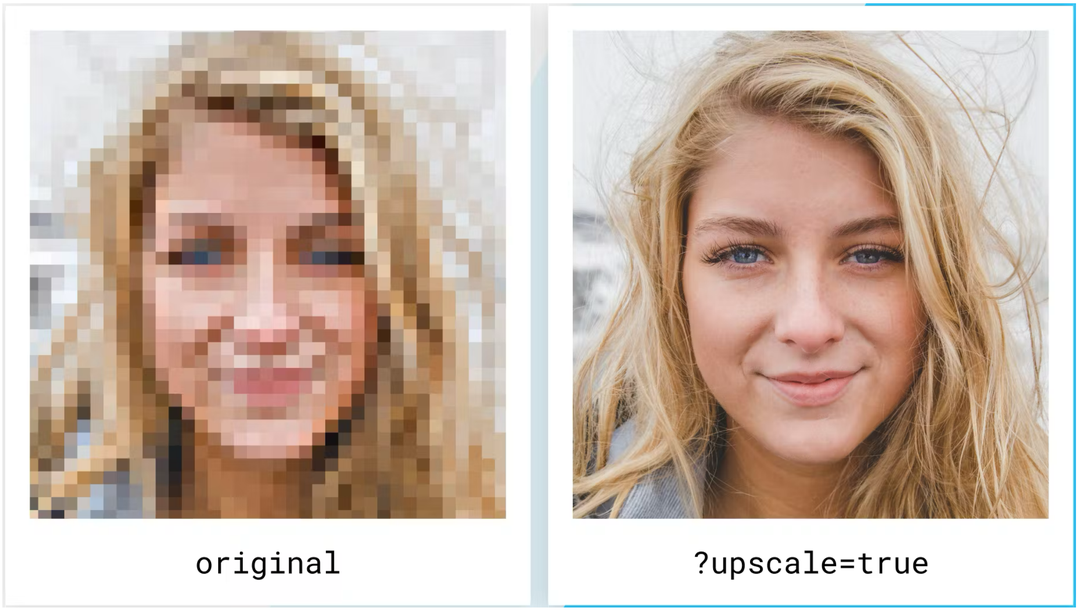


Mathworks

Super Resolution



6 Best AI Image Enhancer You Must Try In 2026, William Bollson



AI Super Resolution Is Now Available on imgix, Eric Sanchez

Physics of Fluids ARTICLE scitation.org/journal/phf

Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels

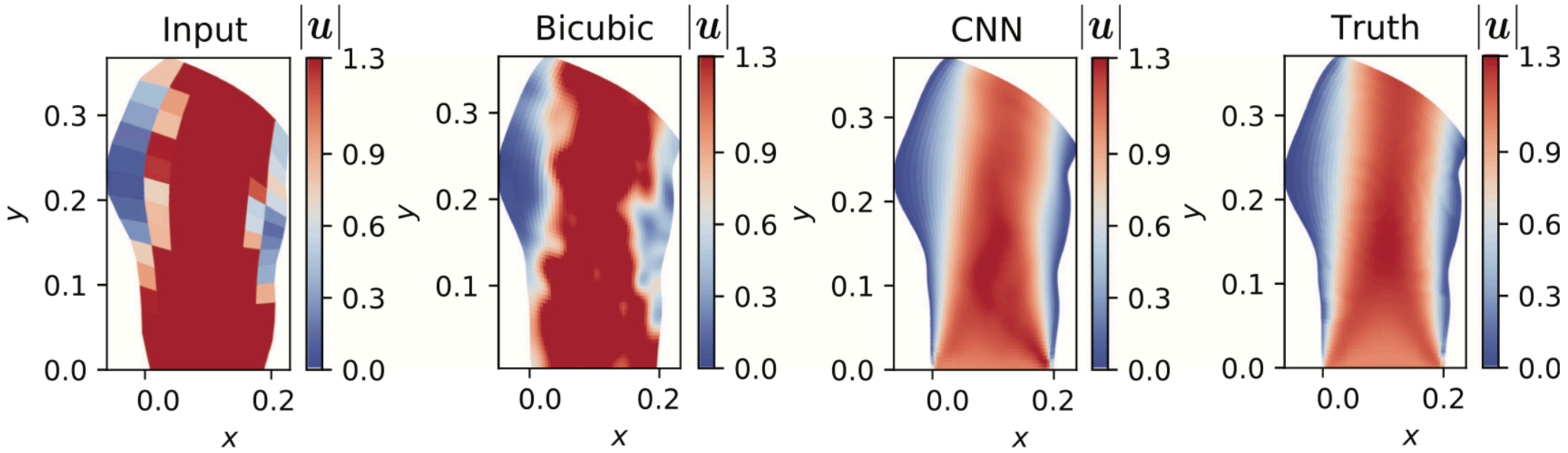
Cite as: Phys. Fluids **33**, 073603 (2021); doi: 10.1063/5.0054312
Submitted: 16 April 2021 · Accepted: 1 June 2021 ·
Published Online: 7 July 2021

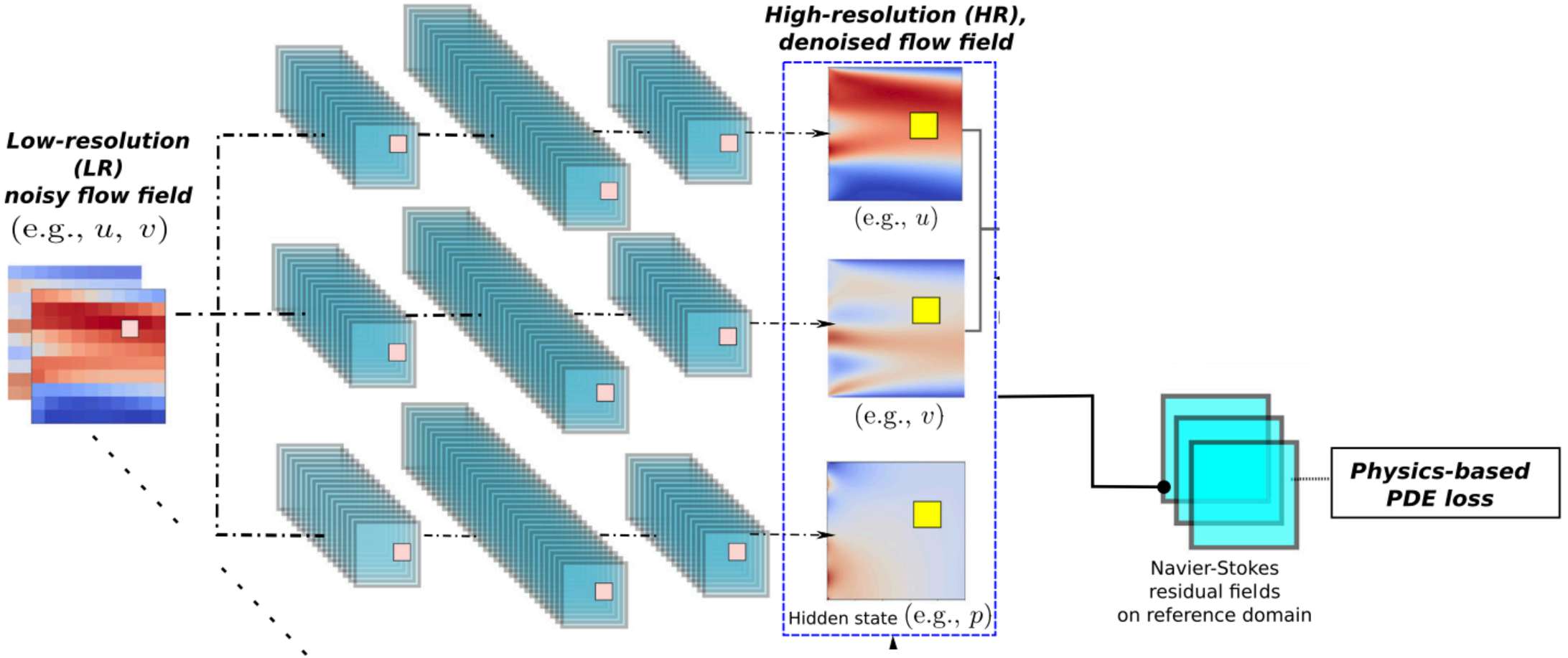
View Online Export Citation CrossMark

Han Gao,[a] Luning Sun,[b] and Jian-Xun Wang[c]

AFFILIATIONS

Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, Indiana 46556, USA


Input
Bicubic
CNN
Truth
$|u|$
1.3
0.9
0.6
0.3
0.0
0.0
0.1
0.2
0.3
y
x



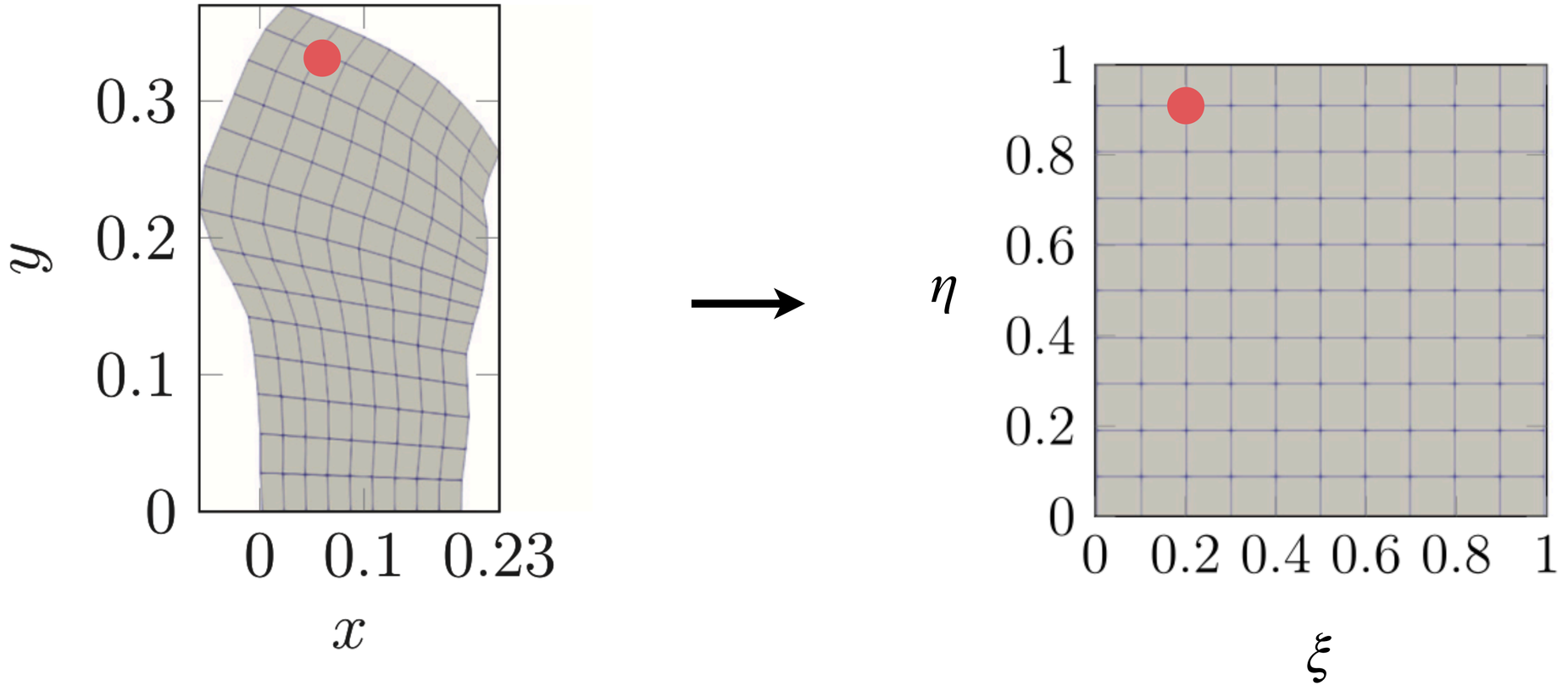
rather than 3 separate networks each producing 1 output channel
you could use 1 network with 3 output channels

Physics Loss

$$\frac{du}{dx} + \frac{dv}{dy} = 0$$

$$u\frac{du}{dx} + v\frac{du}{dy} + \frac{dp}{dx} - 0.01\left(\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2}\right) = 0$$

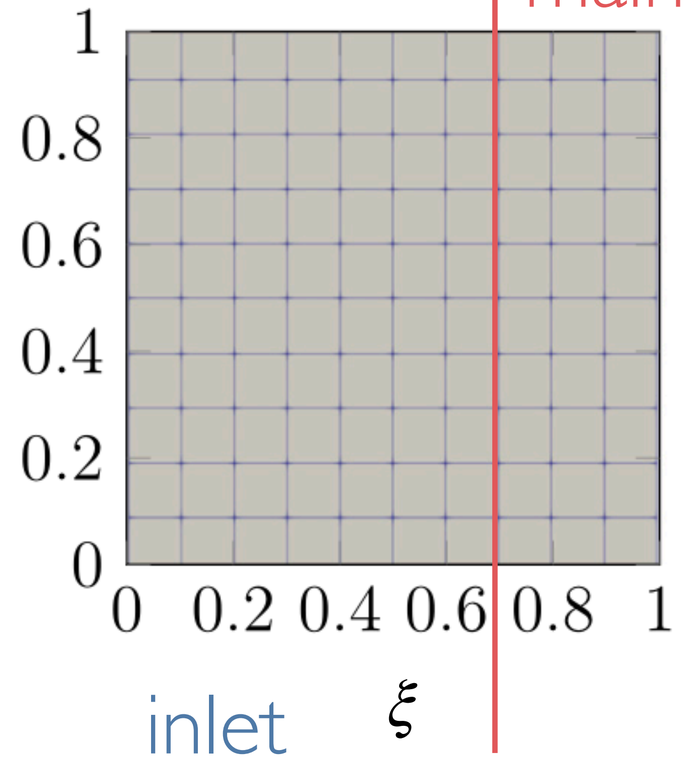
$$u\frac{dv}{dx} + v\frac{dv}{dy} + \frac{dp}{dy} - 0.01\left(\frac{d^2v}{dx^2} + \frac{d^2v}{dy^2}\right) = 0$$


0.3
0.2
0.1
0
y
0
0.1
0.23
x
1
0.8
0.6
0.4
0.2
0
η
0
0.2
0.4
0.6
0.8
1
ξ

Boundary Conditions

outlet $du/d\eta = 0, dv/d\eta = 0, p = 0$

main flow direction



wall

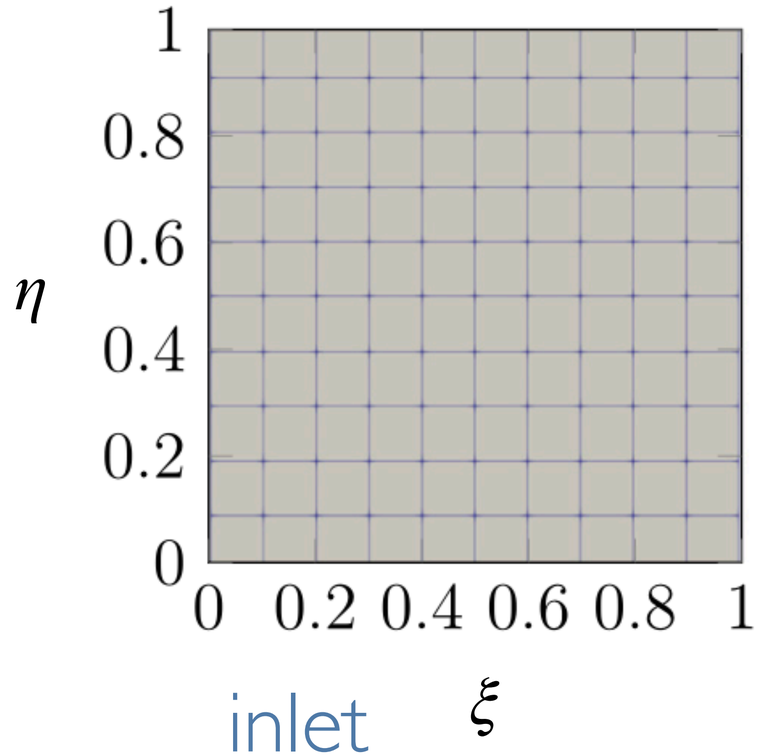
wall

$u = 0, v = 0, dp/d\xi = 0$

inlet

$u = 0, v = 1, dp/d\eta = 0$

Boundary Conditions



inlet

$u = 0, v = 1, dp/d\eta = 0$

height x width

u, v, p each of size $n_\eta \times n_\xi$

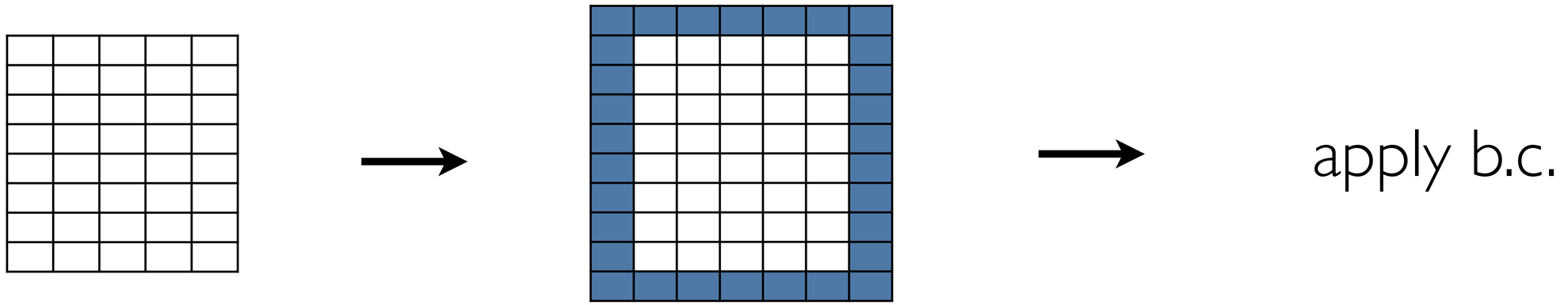
u[0, :] = 0
v[0, :] = 1
p[0, :] = p[1, :]

Boundary Conditions

or have NN output 77x49
and just overwrite boundary values

NN output: 75 x 47

zero pad: 77 x 49



apply b.c.

```
zeropad = nn.ConstantPad2d(1, 0.0)
u = zeropad(u)
v = zeropad(v)
p = zeropad(p)
```

might be useful depending on how you choose to do things

say a was nbatch x nchan x ny x nx and I wanted the first channel without dropping the dimension

`a[:, [0], :, :]` is simpler than: `a[:, 0, :, :].unsqueeze(1)`

```
torch.cat((a, b, c), dim=1) # concatenate on channel dimension
```