mask R-CNN

Sung Man Cho

INDEX

Related Work

Related Work

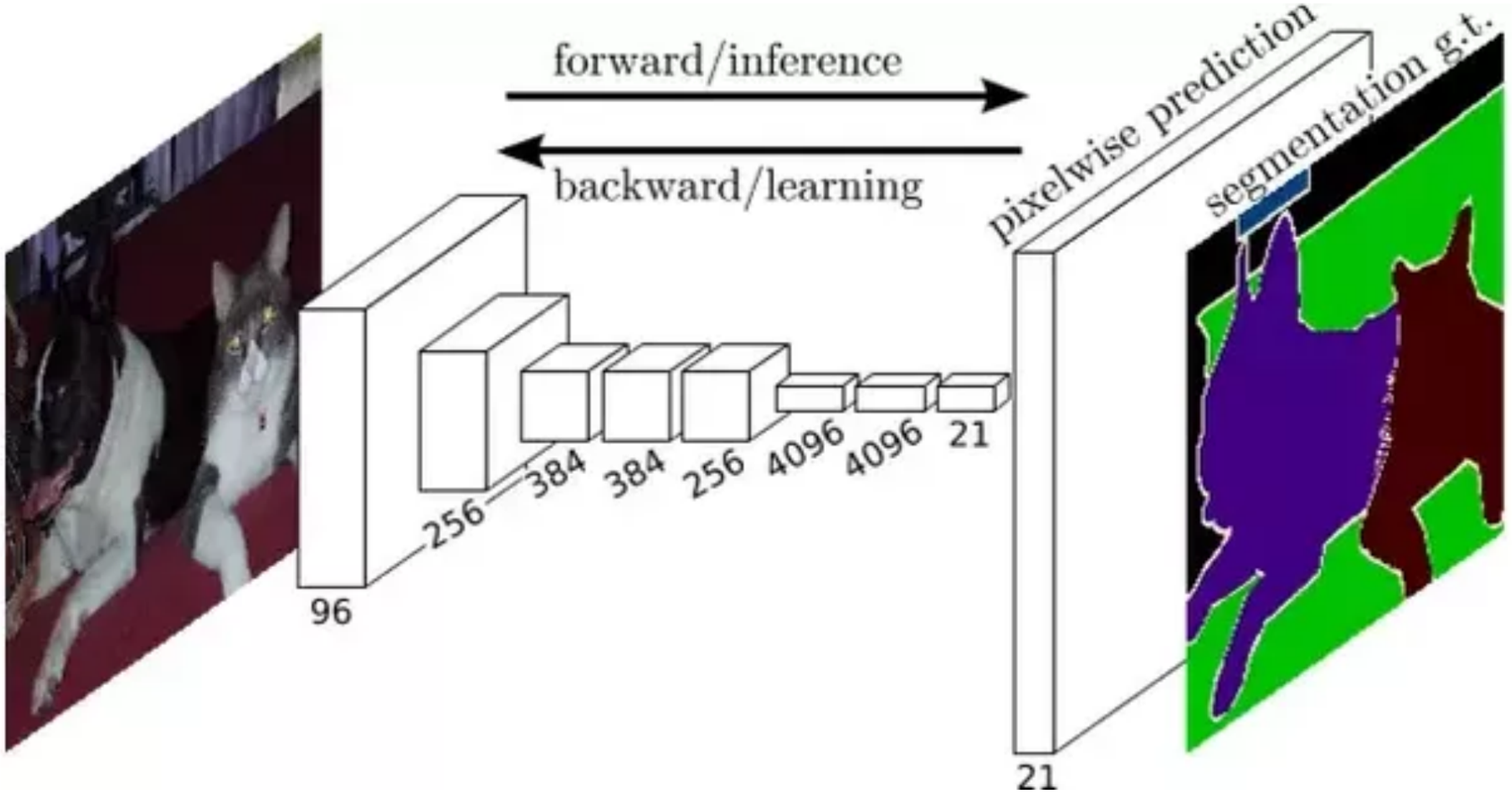
R-CNN Series

R-CNN	Fast R-CNN	Faster R-CNN	Mask R-CNN
Selective Search	**Selective Search** **RoI Pooling**	**RPN**	**Masking** **RoI Align**

Related Work

FCN

- Fully Convolutional Layer -> heat map output.

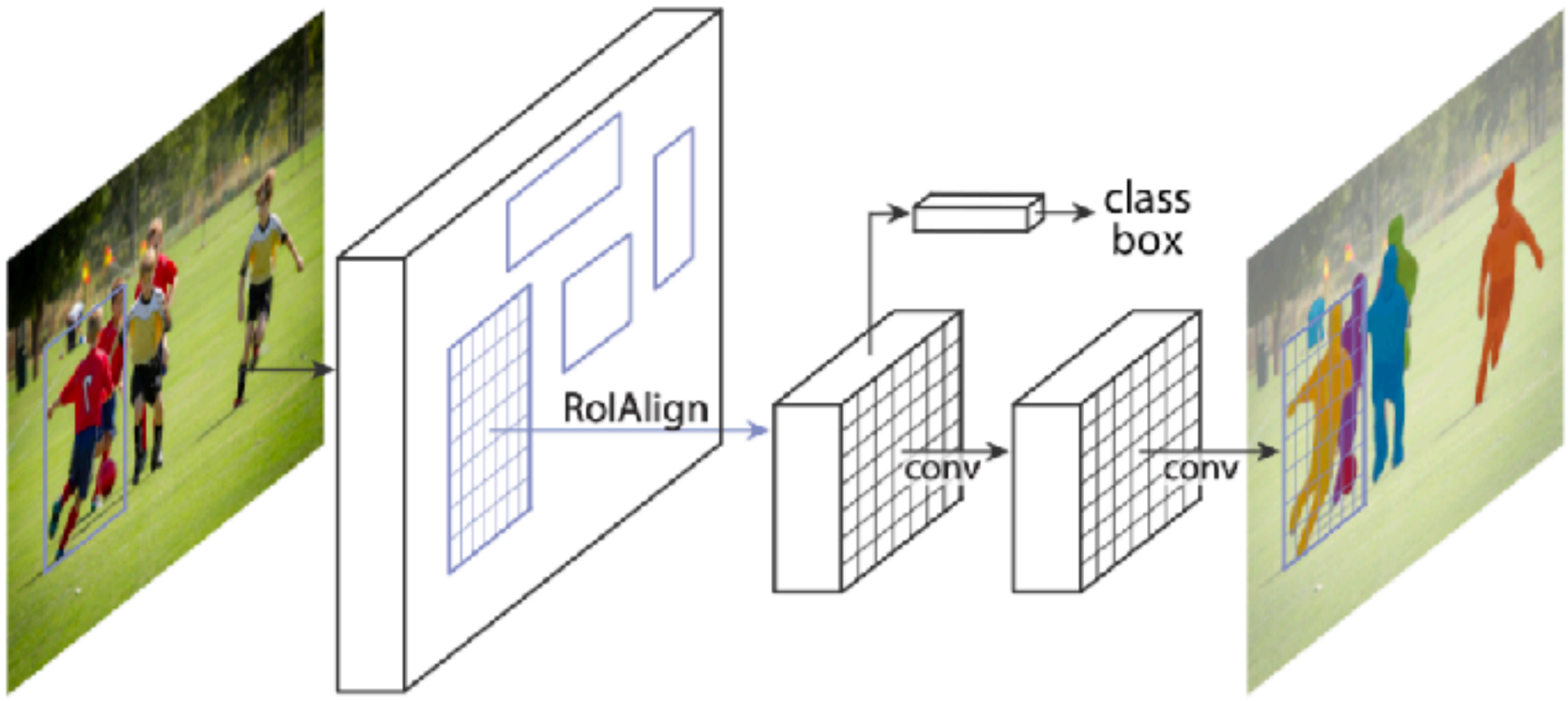


Mask R-CNN

Mask R-CNN

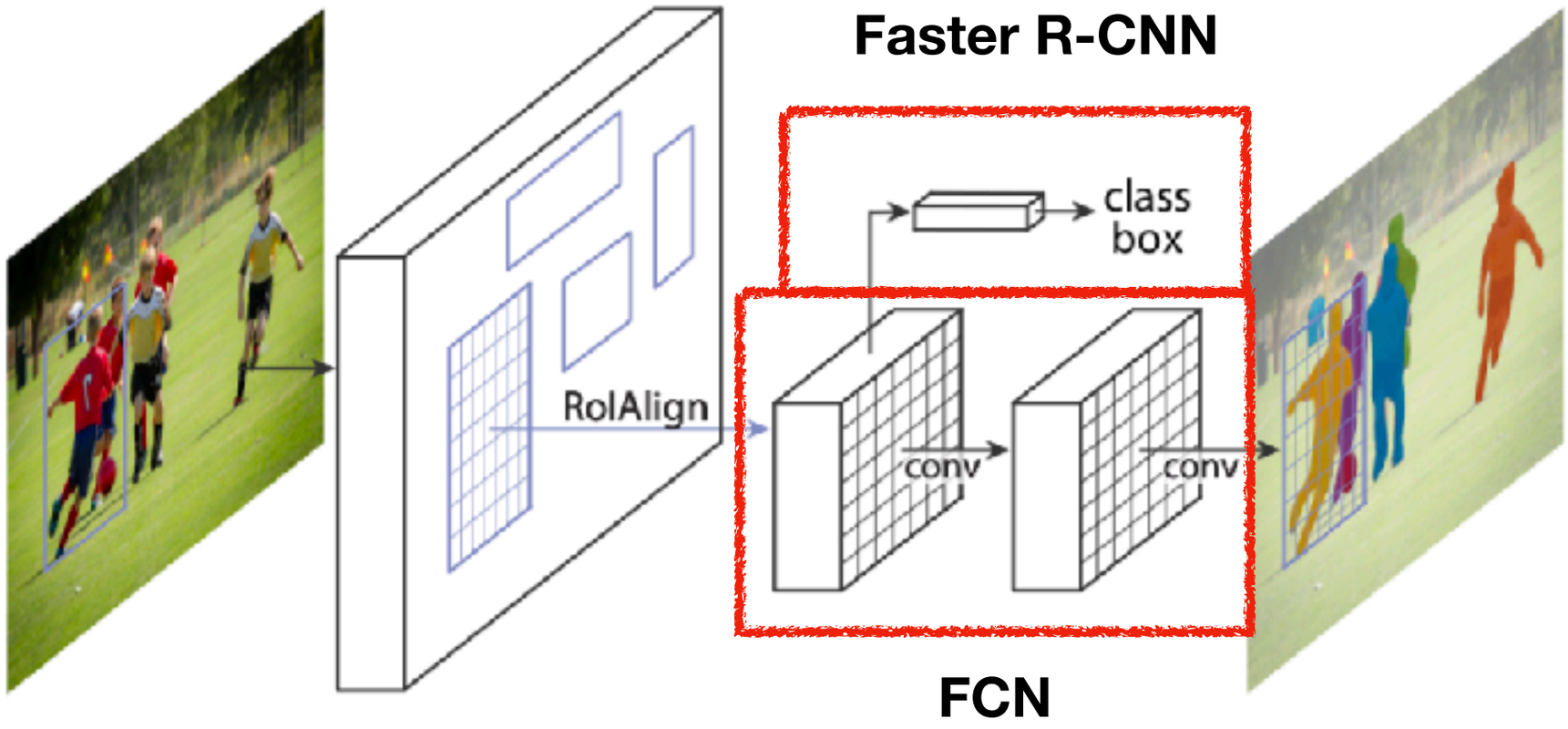
Architecture

- Goal is enabling framework for **instance segmentation.**



Mask R-CNN

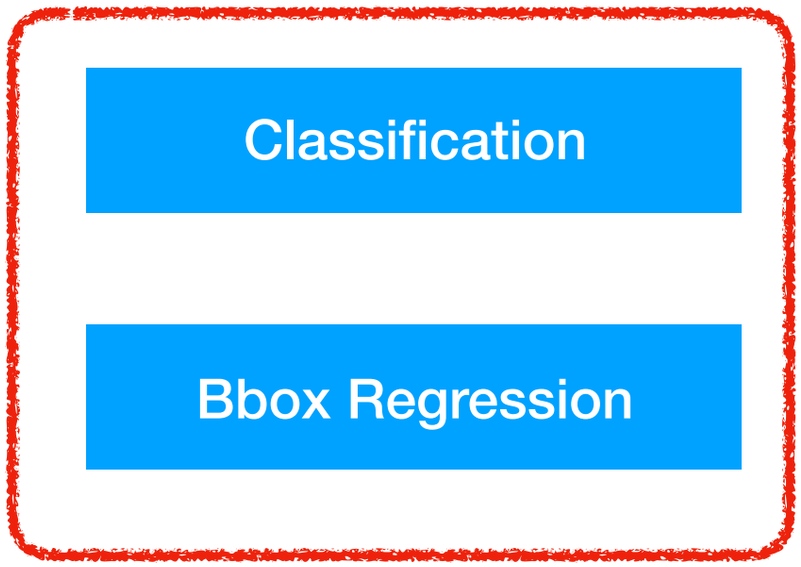
Architecture



net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	**36.7**	**59.5**	**38.9**

Mask R-CNN

Architecture



Faster R-CNN

- **Classification**
- **Instance Level RoI**

FCN

- ~~**Pixel-level Classification**~~
- **Per Pixel ~~Softmax (Multinomial)~~**
- ~~**Multi-Instance**~~

Mask R-CNN

Architecture

FCN

- ~~**Pixel-level Classification**~~
- **Per Pixel ~~Softmax (Multinomial)~~**
- ~~**Multi-Instance**~~
- **Per Pixel Sigmoid (Binary)**

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

Mask R-CNN

Architecture

DB

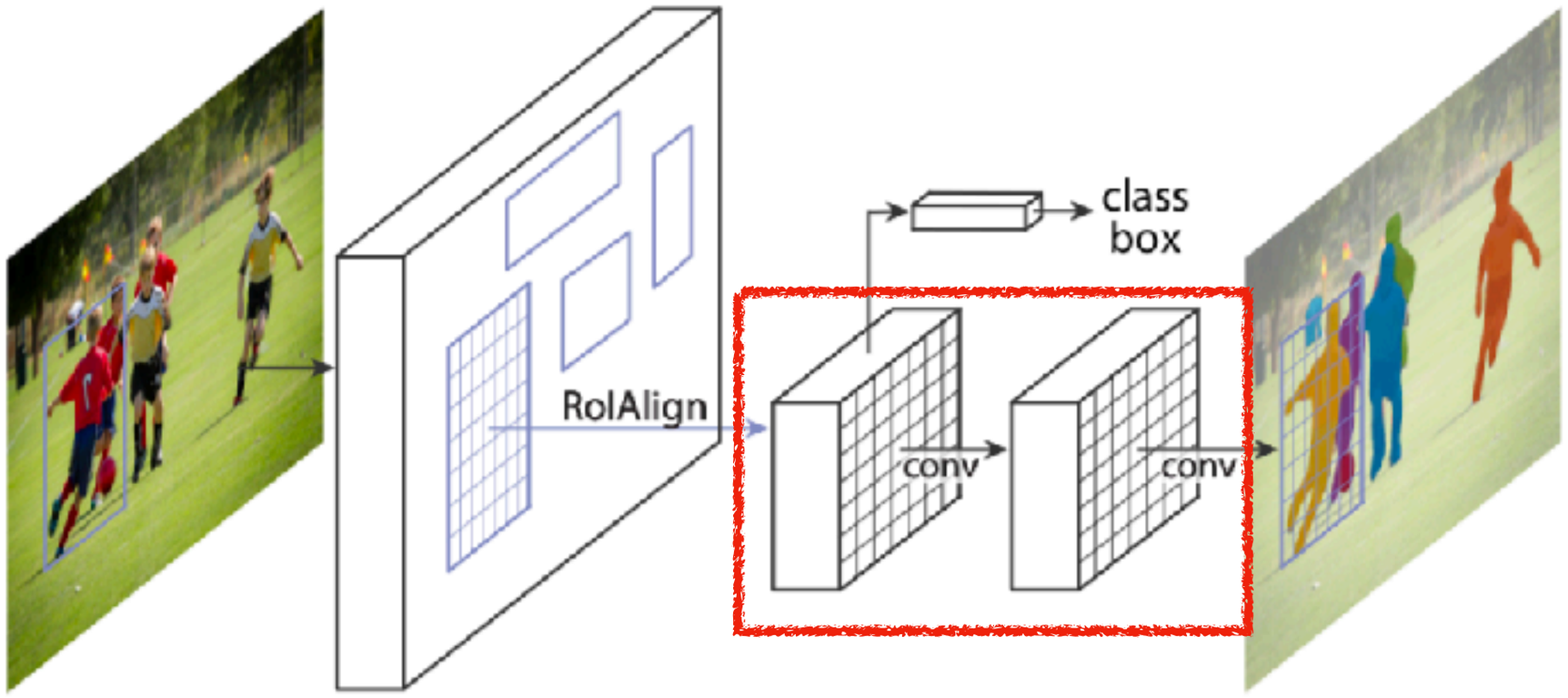
BBox + Class + Mask

$$L = L_{cls} + L_{box} + L_{mask}$$

L_{cls}: Softmax Cross Entropy
L_{box}: Regression
L_{mask}: Binary Cross Entropy

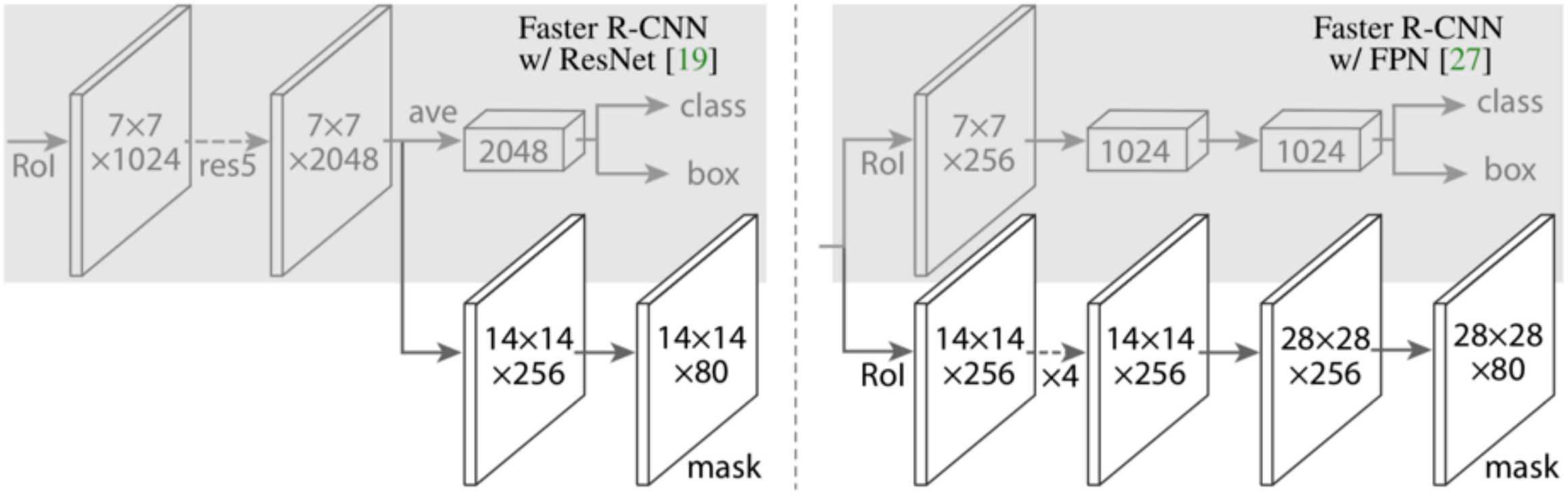
Mask R-CNN

Mask Branch



Mask R-CNN

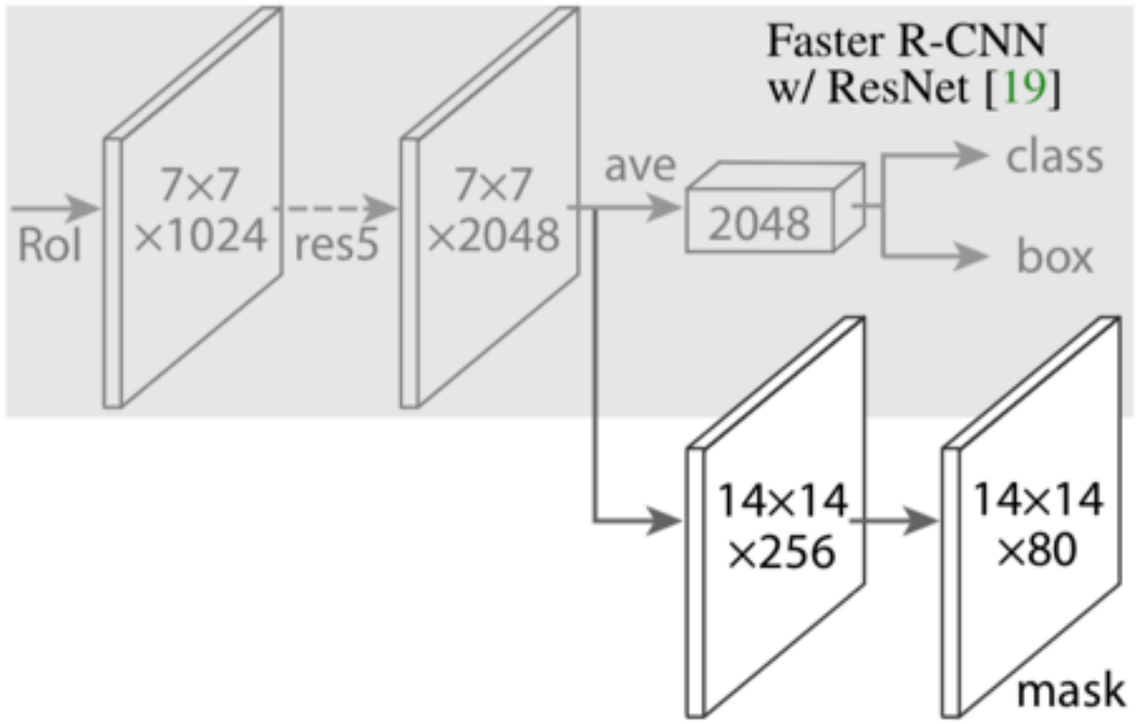
Mask Branch



while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.

Mask R-CNN

Mask Branch



while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.

```
def _mask_network(self):
    def f(x):
        target_metadata, output_features, output_proposal_bounding_boxes = x

        mask_features = keras_rcnn.layers.RegionOfInterest(
            extent=(14, 14),
            strides=2
        )([
            target_metadata,
            output_features,
            output_proposal_bounding_boxes
        ])

mask_features = keras.layers.TimeDistributed(
    keras.layers.Conv2D(
        activation="relu",
        filters=256,
        kernel_size=(3, 3),
        padding="same"
    )
)(mask_features)

mask_features = keras.layers.TimeDistributed(
    keras.layers.Conv2D(
        activation="relu",
        filters=256,
        kernel_size=(3, 3),
        padding="same"
    )
)(mask_features)
```

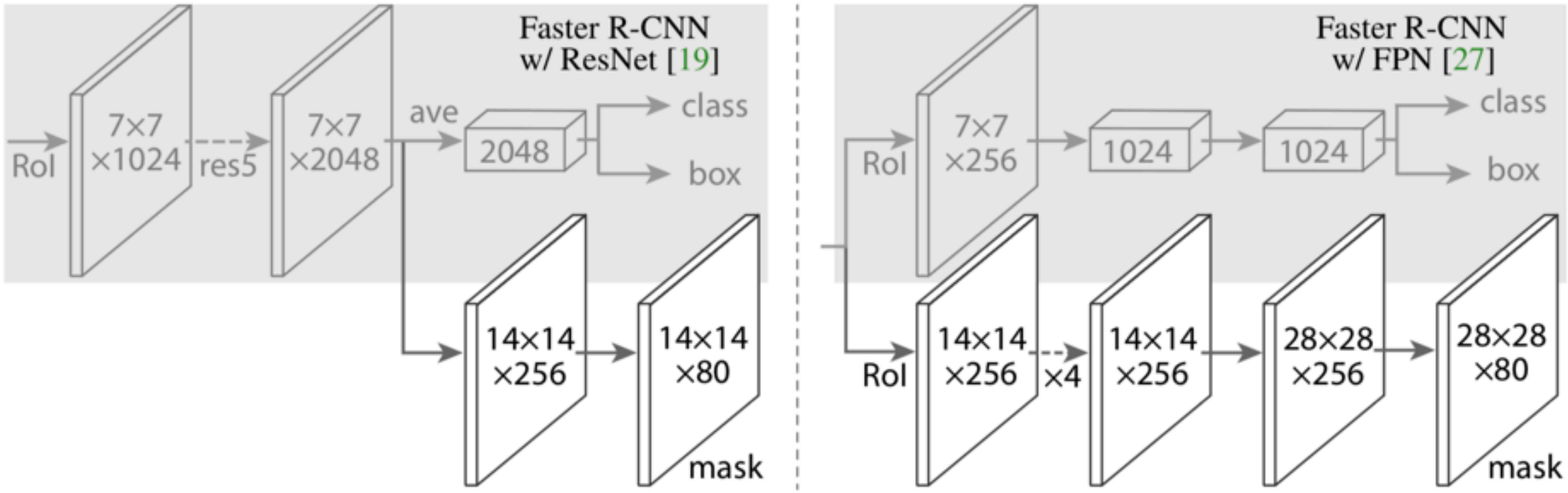
Mask R-CNN

Mask Branch

```
def _mask_network(self):
    def f(x):
        target_metadata, output_features, output_proposal_bounding_boxes = x

        mask_features = keras_rcnn.layers.RegionOfInterest(
            extent=(14, 14),
            strides=2
        )([
            target_metadata,
            output_features,
            output_proposal_bounding_boxes
        ])

        mask_features = keras.layers.TimeDistributed(
            keras.layers.Conv2D(
                activation="relu",
                filters=256,
                kernel_size=(3, 3),
                padding="same"
            )
        )(mask_features)

        mask_features = keras.layers.TimeDistributed(
            keras.layers.Conv2D(
                activation="relu",
                filters=256,
                kernel_size=(3, 3),
                padding="same"
            )
        )(mask_features)

        mask_features = keras.layers.TimeDistributed(
            keras.layers.Conv2D(
                activation="relu",
                filters=256,
                kernel_size=(3, 3),
                padding="same"
            )
        )(mask_features)

        mask_features = keras.layers.TimeDistributed(
            keras.layers.Conv2DTranspose(
                activation="relu",
                filters=256,
                kernel_size=(2, 2),
                strides=2
            )
        )(mask_features)

        mask_features = keras.layers.TimeDistributed(
            keras.layers.Conv2D(
                activation="sigmoid",
                filters=self.n_categories,
                kernel_size=(1, 1),
                strides=1
            )
        )(mask_features)

        return mask_features

    return f
```

Mask R-CNN

Mask Branch

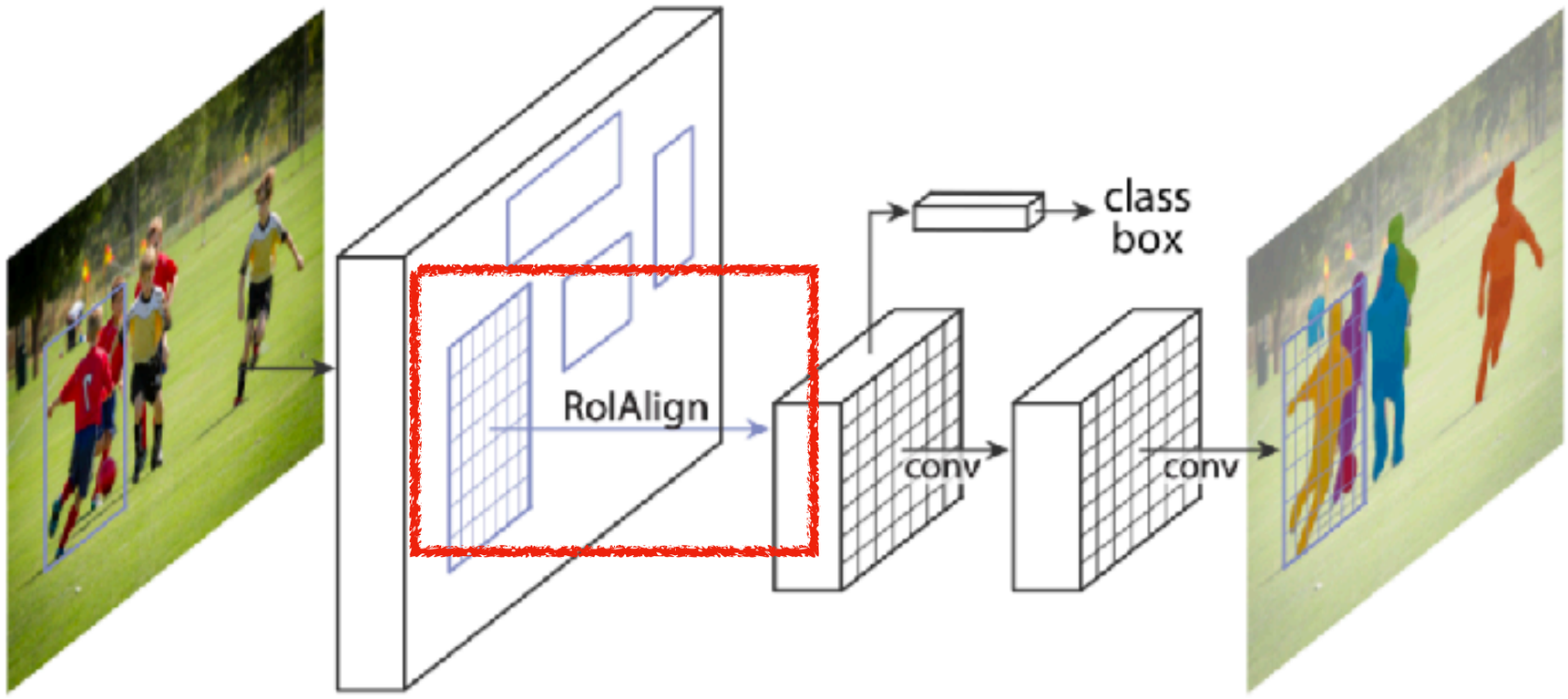


	mask branch	AP	AP_{50}	AP_{75}
MLP	fc: 1024→1024→80·28^2	31.5	53.7	32.8
MLP	fc: 1024→1024→1024→80·28^2	31.5	54.0	32.6
FCN	conv: 256→256→256→256→256→80	**33.6**	**55.2**	**35.3**

(e) **Mask Branch** (ResNet-50-FPN): Fully convolutional networks (FCN) *vs.* multi-layer perceptrons (MLP, fully-connected) for mask prediction. FCNs improve results as they take advantage of explicitly encoding spatial layout.

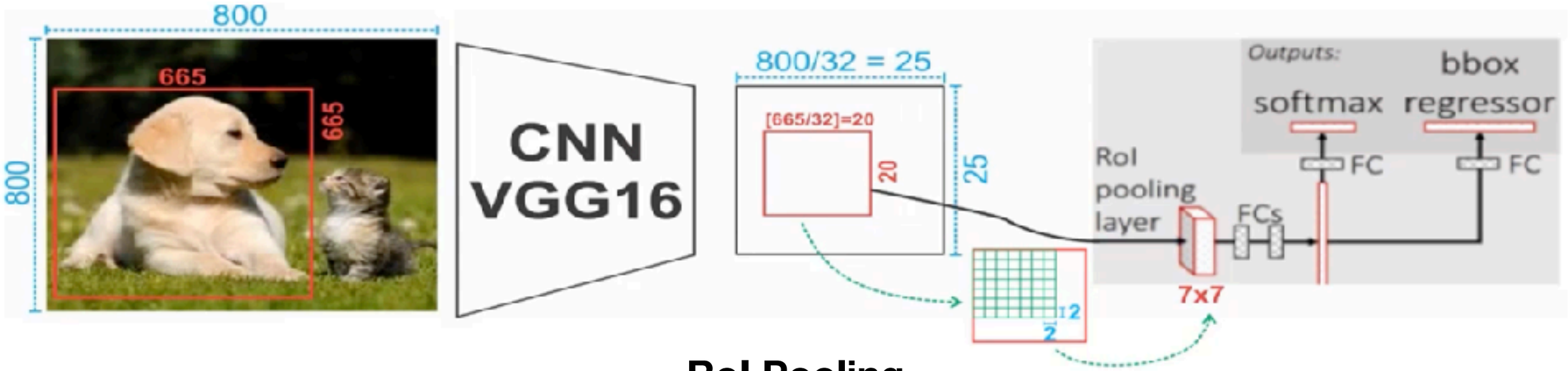
Mask R-CNN

RoIAlign

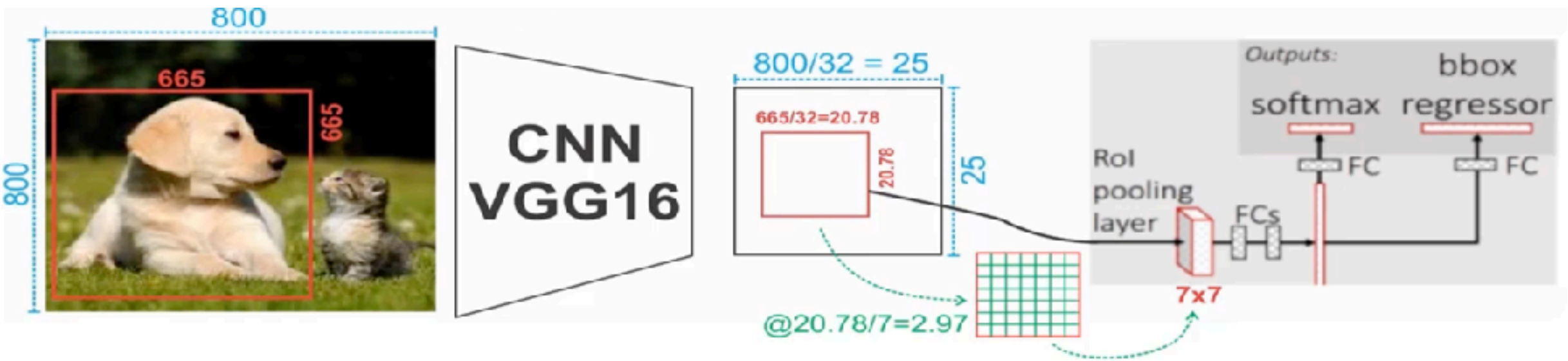


Mask R-CNN

RoIAlign



RoI Pooling



RoI Align

Mask R-CNN

RoIAlign

```
x1, y1, x2, y2 = torch.split(boxes, 1, dim=1)
image_height, image_width = featuremap.size()[2:4]

if self.transform_fpcoor:
    spacing_w = (x2 - x1) / float(self.crop_width)
    spacing_h = (y2 - y1) / float(self.crop_height)

    nx0 = (x1 + spacing_w / 2 - 0.5) / float(image_width - 1)
    ny0 = (y1 + spacing_h / 2 - 0.5) / float(image_height - 1)
    nw = spacing_w * float(self.crop_width - 1) / float(image_width - 1)
    nh = spacing_h * float(self.crop_height - 1) / float(image_height - 1)

    boxes = torch.cat((ny0, nx0, ny0 + nh, nx0 + nw), 1)
else:
    x1 = x1 / float(image_width - 1)
    x2 = x2 / float(image_width - 1)
    y1 = y1 / float(image_height - 1)
    y2 = y2 / float(image_height - 1)
    boxes = torch.cat((y1, x1, y2, x2), 1)

boxes = boxes.detach().contiguous()
box_ind = box_ind.detach()
return CropAndResizeFunction(self.crop_height, self.crop_width, self.extrapolation_value)(featuremap, boxes, box_ind)
```

Pytorch

Mask R-CNN

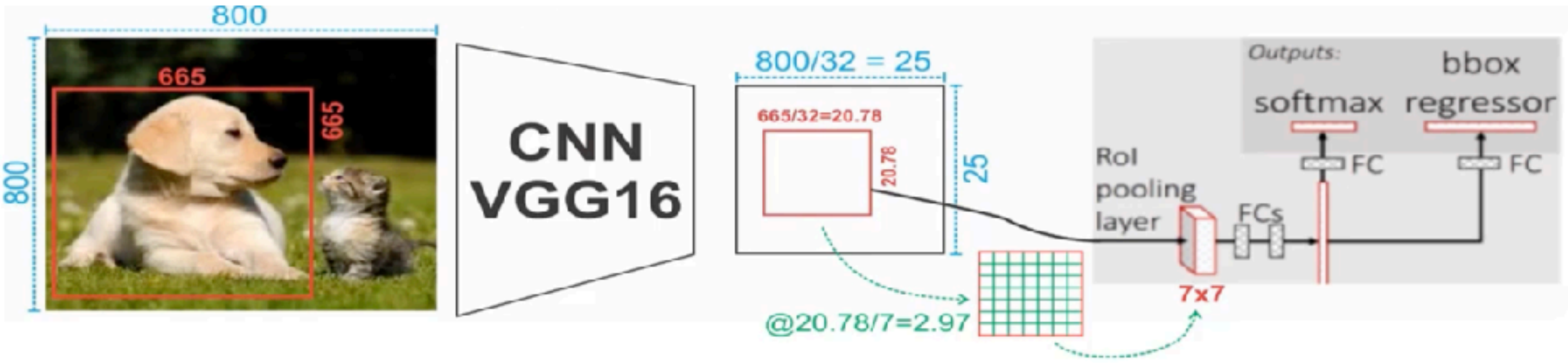
RoIAlign

```
# convert regions from (x, y, w, h) to (x1, y1, x2, y2)
boxes = keras.backend.cast(boxes, keras.backend.floatx())

boxes = boxes / self.stride

x1 = boxes[..., 0]
y1 = boxes[..., 1]
x2 = boxes[..., 2]
y2 = boxes[..., 3]

# normalize the boxes
shape = metadata[0]

h = keras.backend.cast(shape[0], keras.backend.floatx())
w = keras.backend.cast(shape[1], keras.backend.floatx())

x1 /= w
y1 /= h
x2 /= w
y2 /= h

x1 = keras.backend.expand_dims(x1, axis=2)
y1 = keras.backend.expand_dims(y1, axis=2)
x2 = keras.backend.expand_dims(x2, axis=2)
y2 = keras.backend.expand_dims(y2, axis=2)

boxes = keras.backend.concatenate([y1, x1, y2, x2], axis=2)
boxes = keras.backend.reshape(boxes, (-1, 4))

slices = keras_rcnn.backend.crop_and_resize(image, boxes, self.extent)

return keras.backend.expand_dims(slices, axis=0)
```

Keras

Mask R-CNN

RoIAlign



	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

(d) **RoIAlign** (ResNet-50-**C5**, *stride 32*): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in big accuracy gaps.

Fully Convolutional Networks

Jeon Ji Hye

Index

Before Presentation

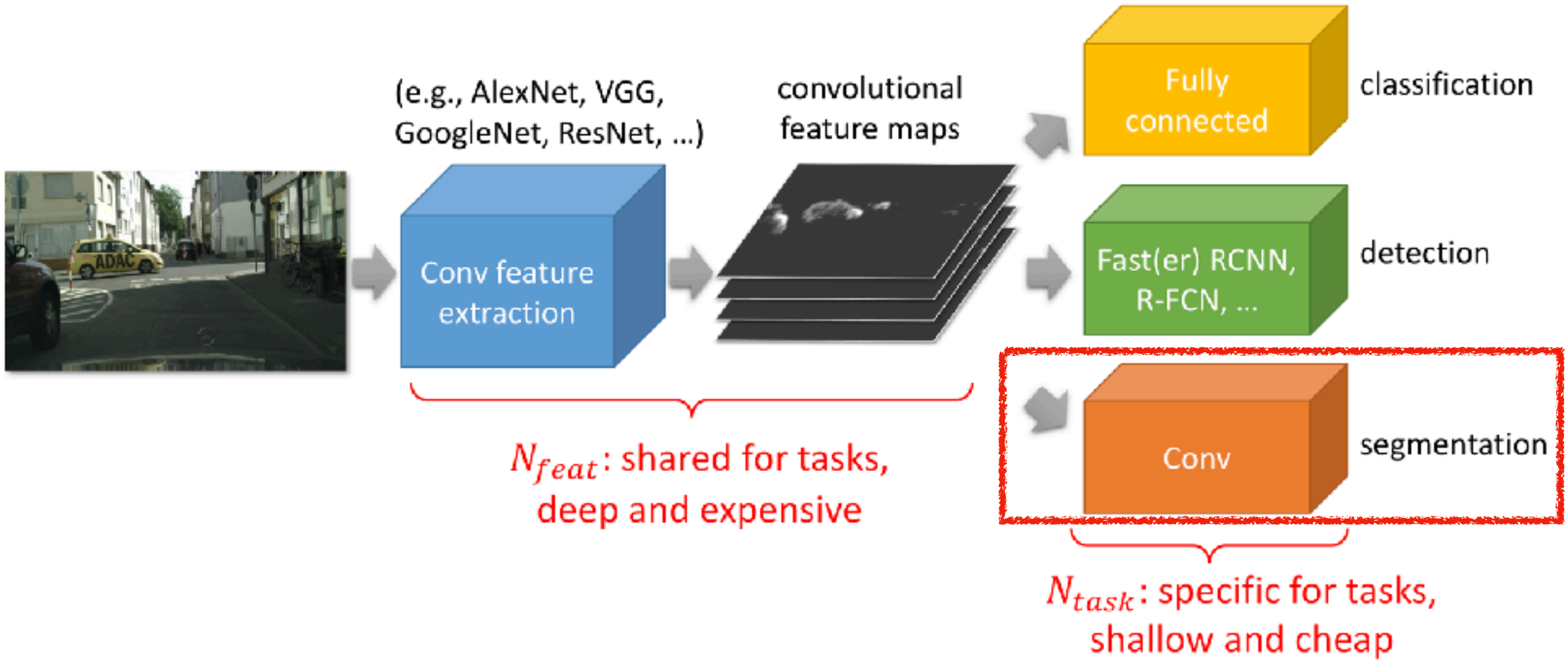
Two Important Things..

Fully Convolutional Networks
: 1x1 Convolution Layer

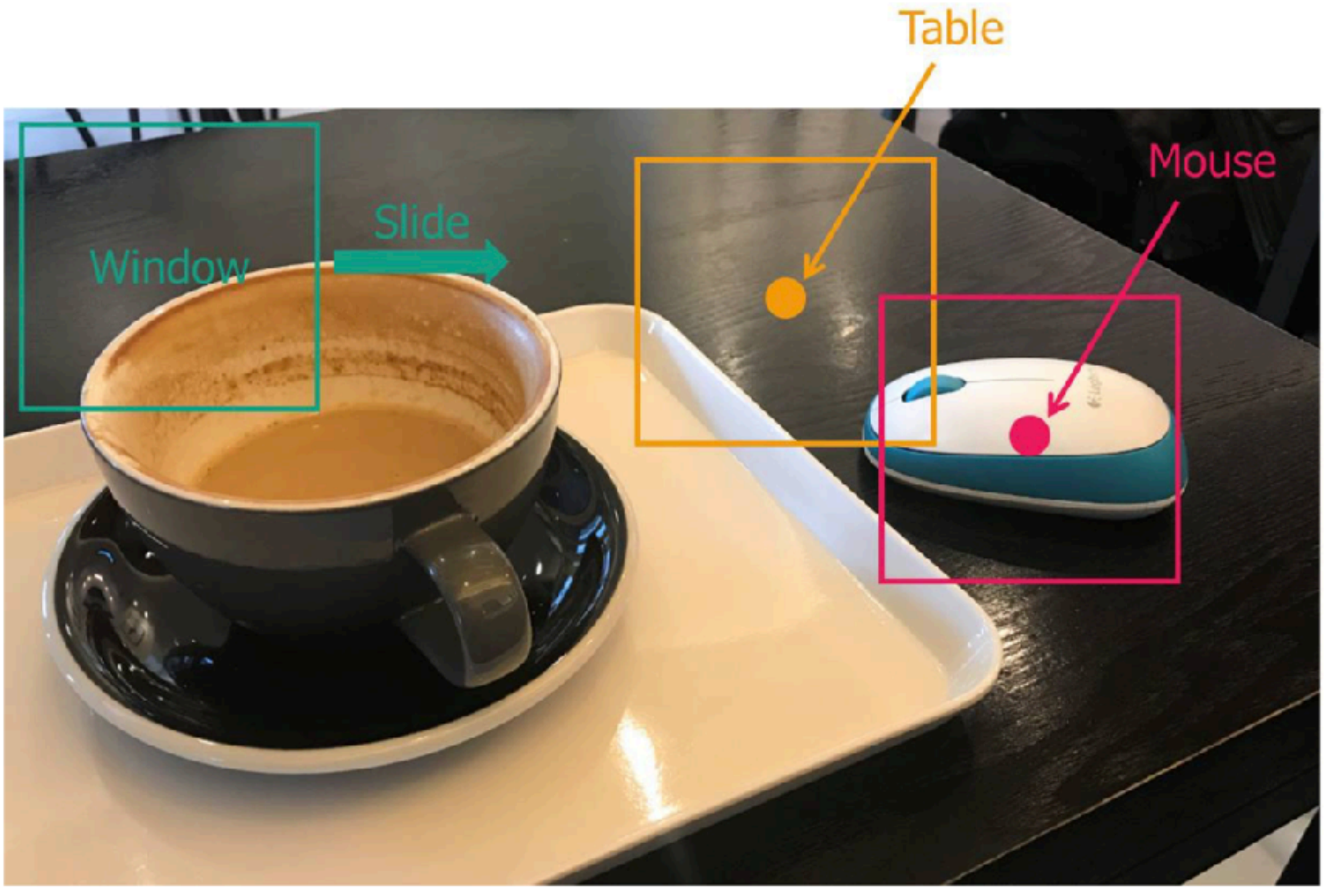
Skip Connection

Introduction

Introduction

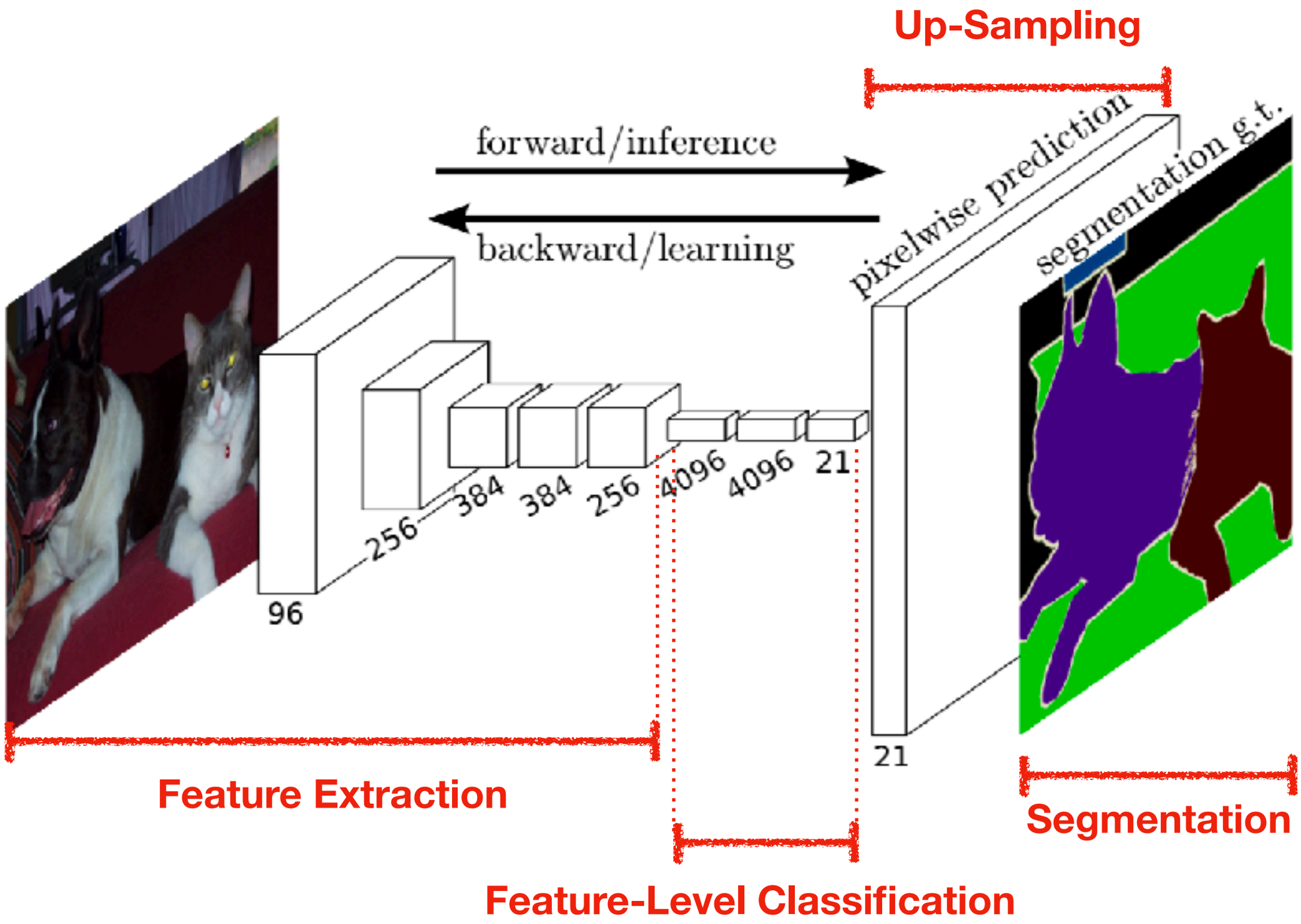


Previous Work

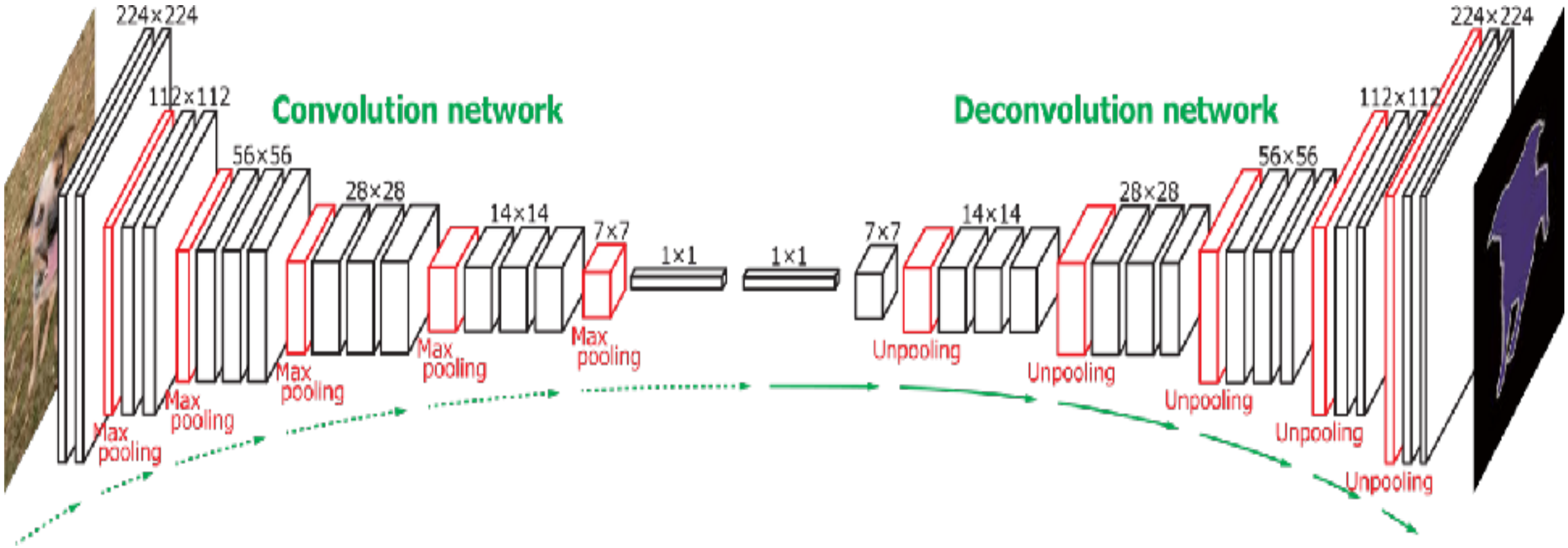


Architecture

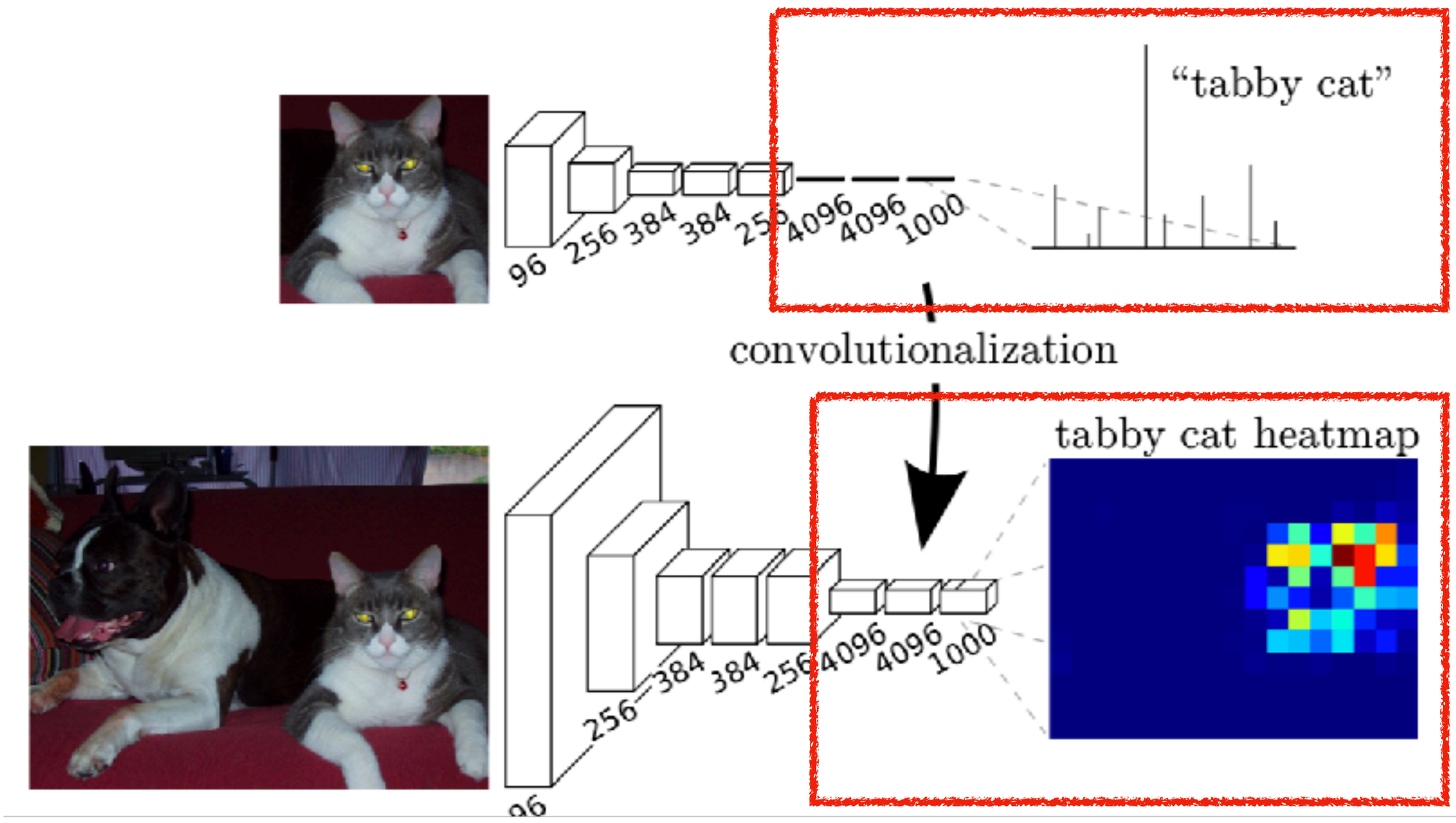
Architecture



Architecture

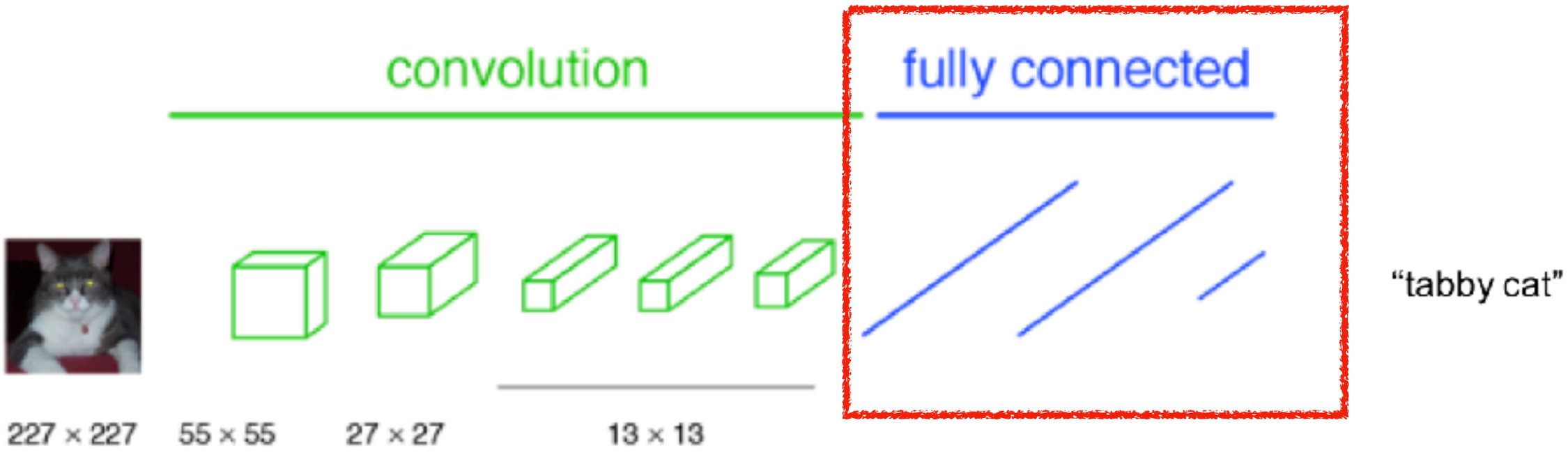


Architecture



Fully Convolutional Networks

Fully Connected Problem



Need Fixed Size

Lose Spatial Information

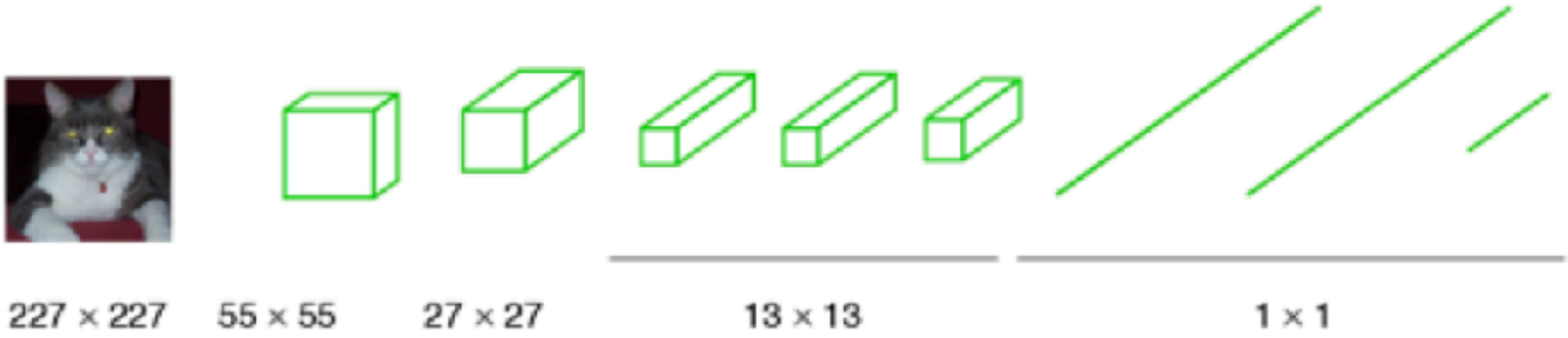
FC Layer - Too many Param

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

VGG 16

1x1 Conv-Layer



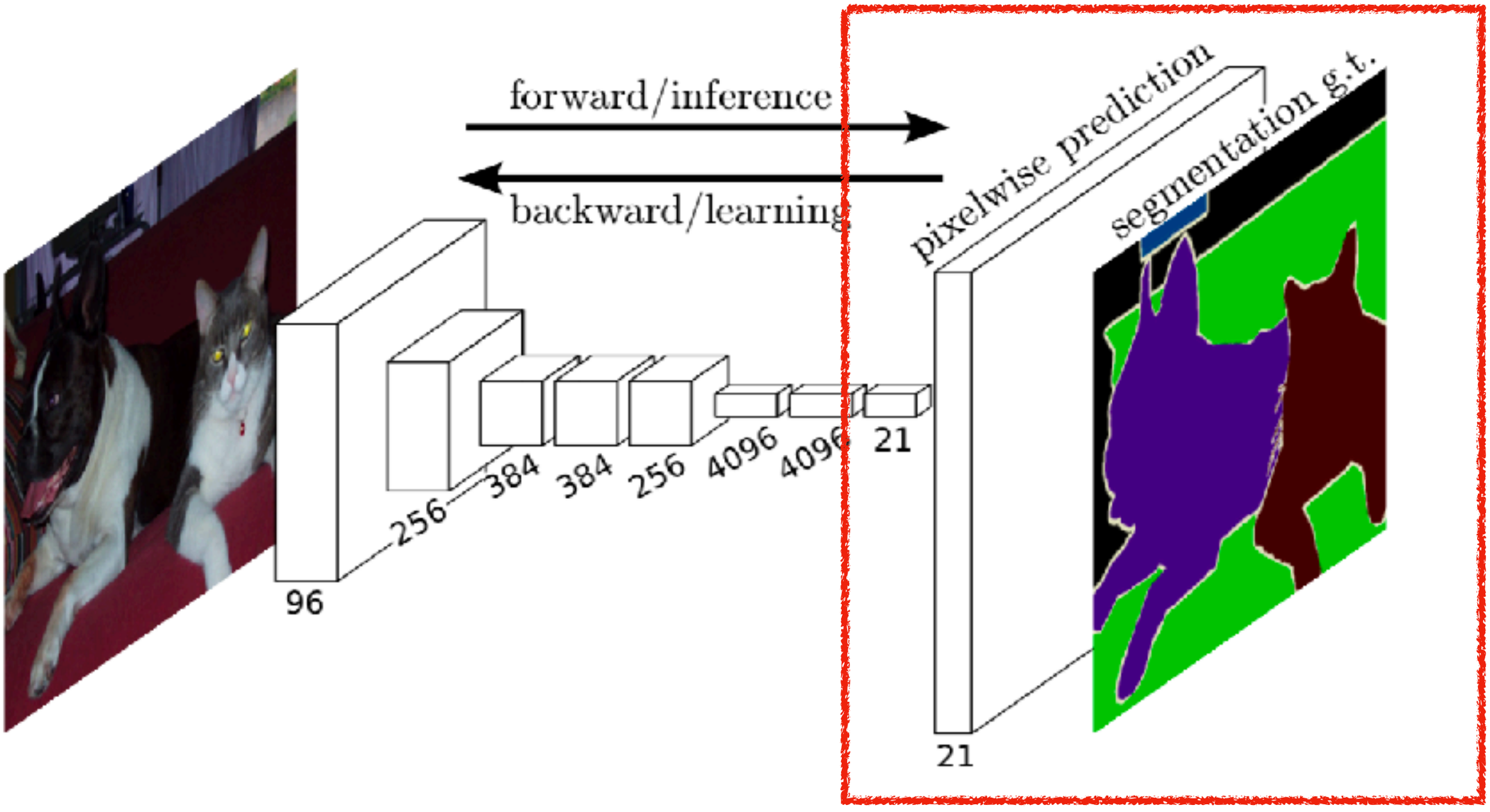
Remain Spatial Information

per patch -> whole image
: reduce calculation

1x1 Conv-Layer

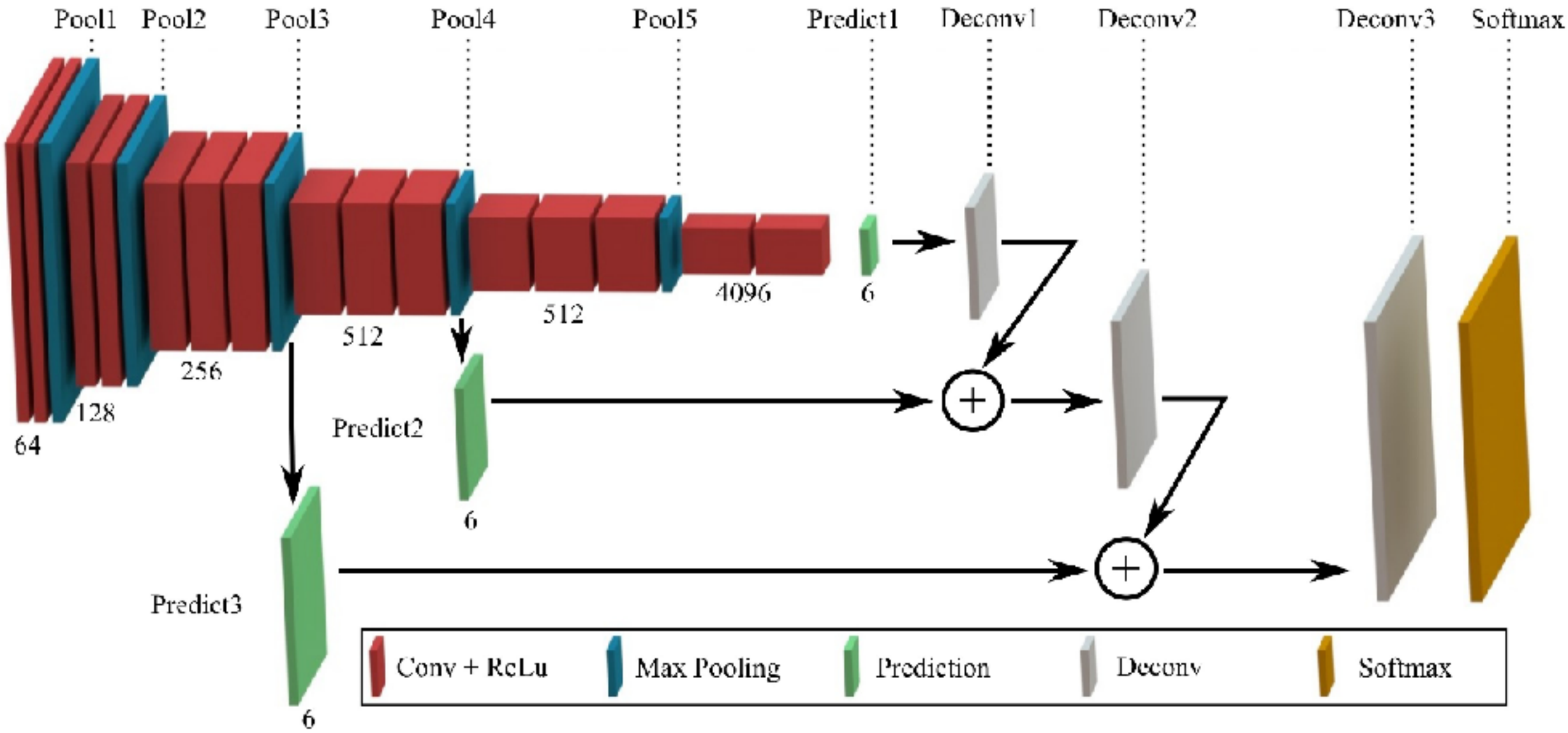
Up-Sampling

Coarse To Dense



Do not use shift-and-stitch

Coarse To Dense



Up-Sampling Problem

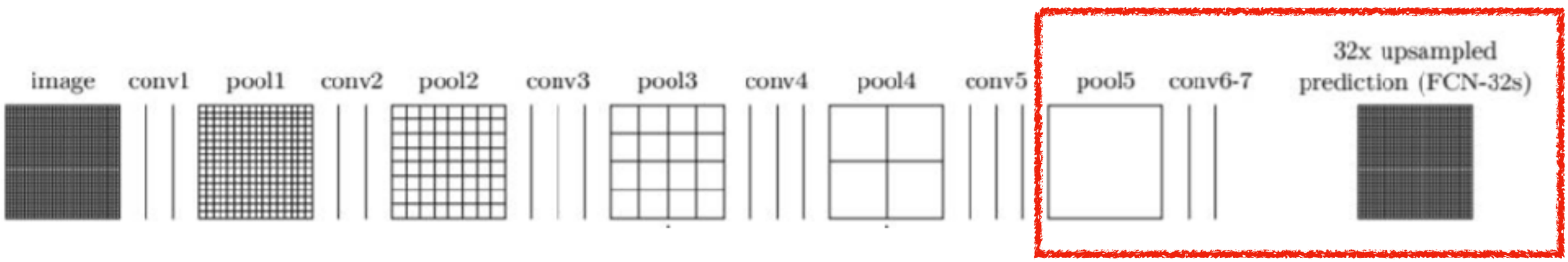
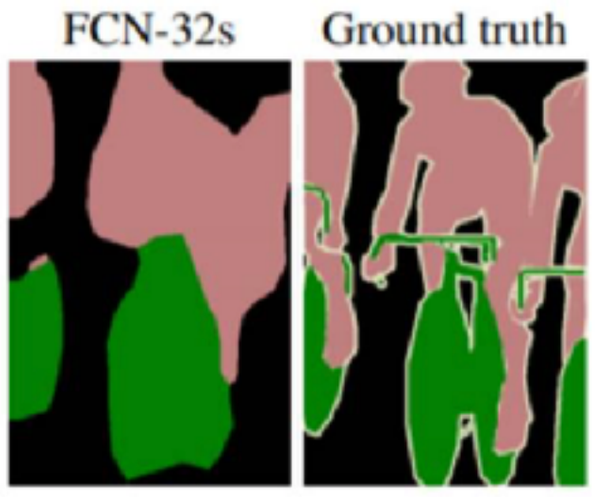


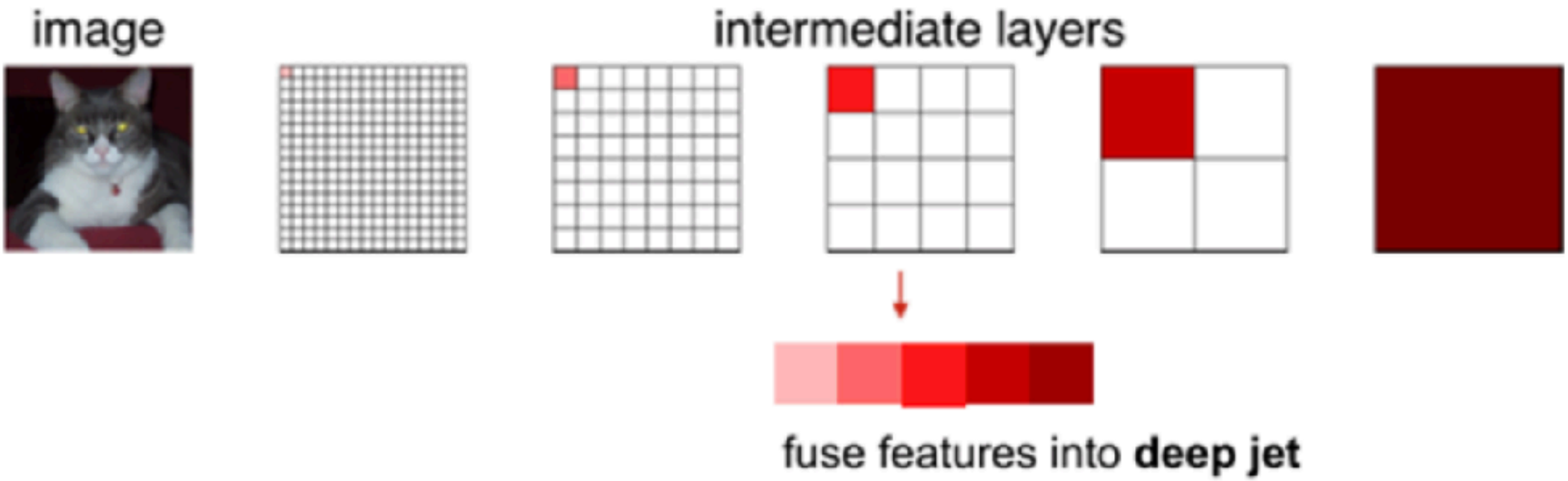
Figure 6. (Conventional) Fully Convolutional Networks : FCN-32s

x32 Up-Sampling



Lose Detail

Deep jet



Skip Layer

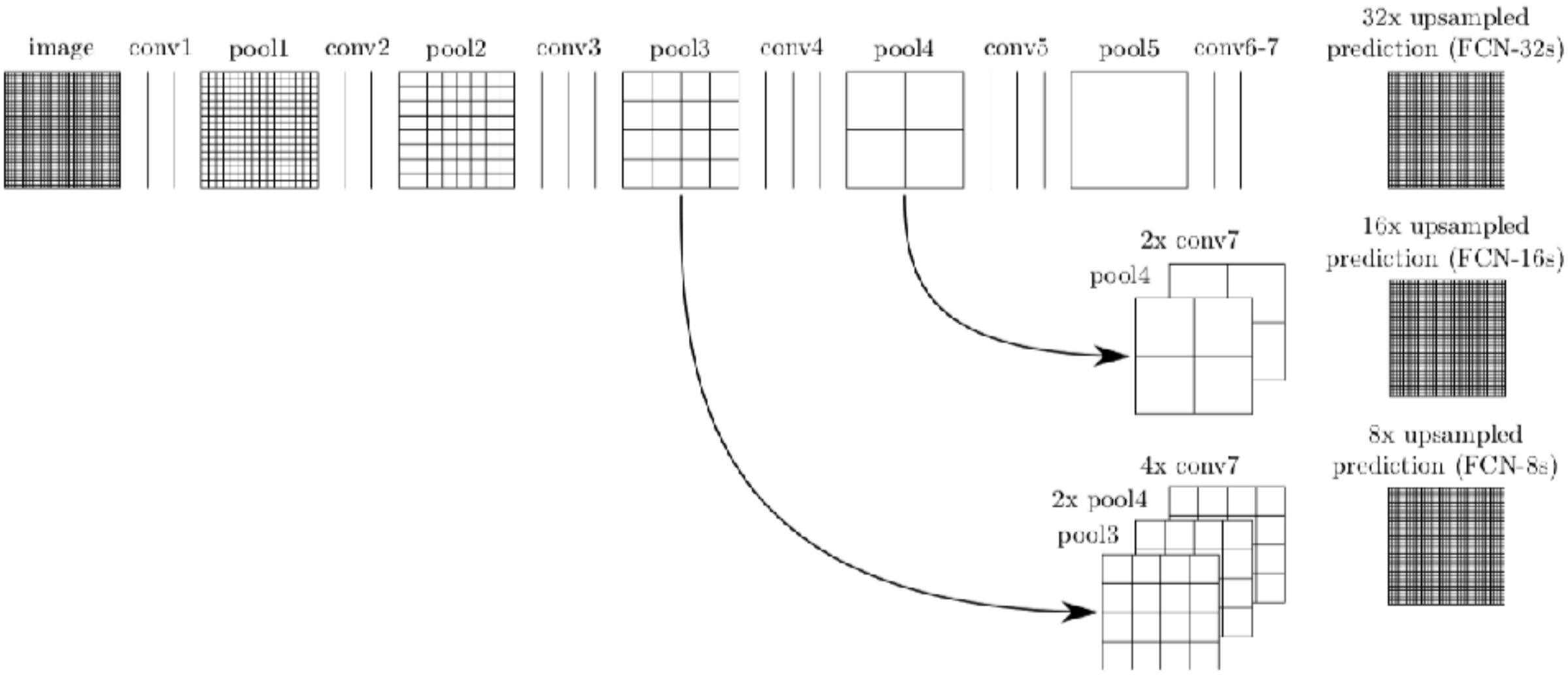
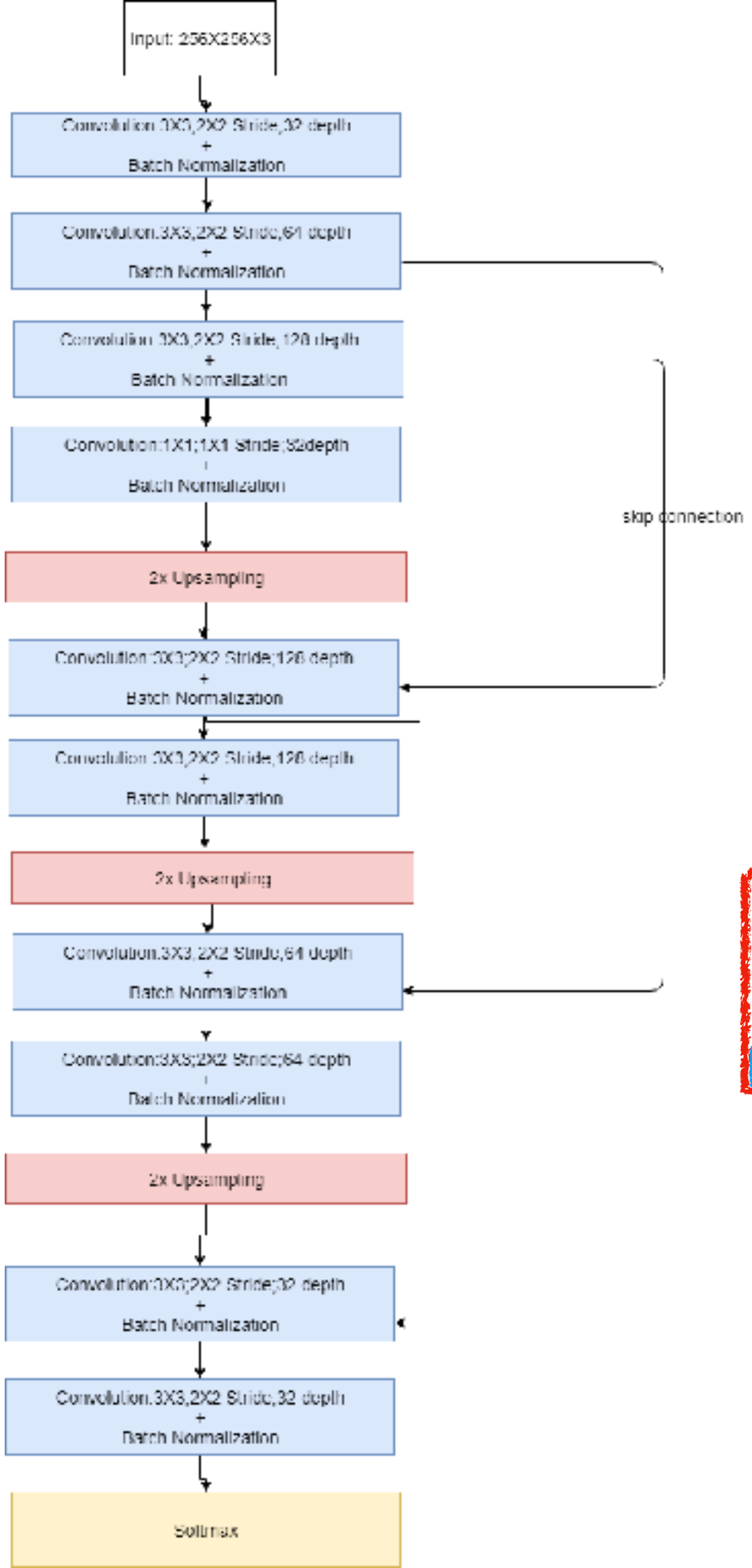


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled
prediction (FCN-32s)
2x conv7
pool4
16x upsampled
prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled
prediction (FCN-8s)

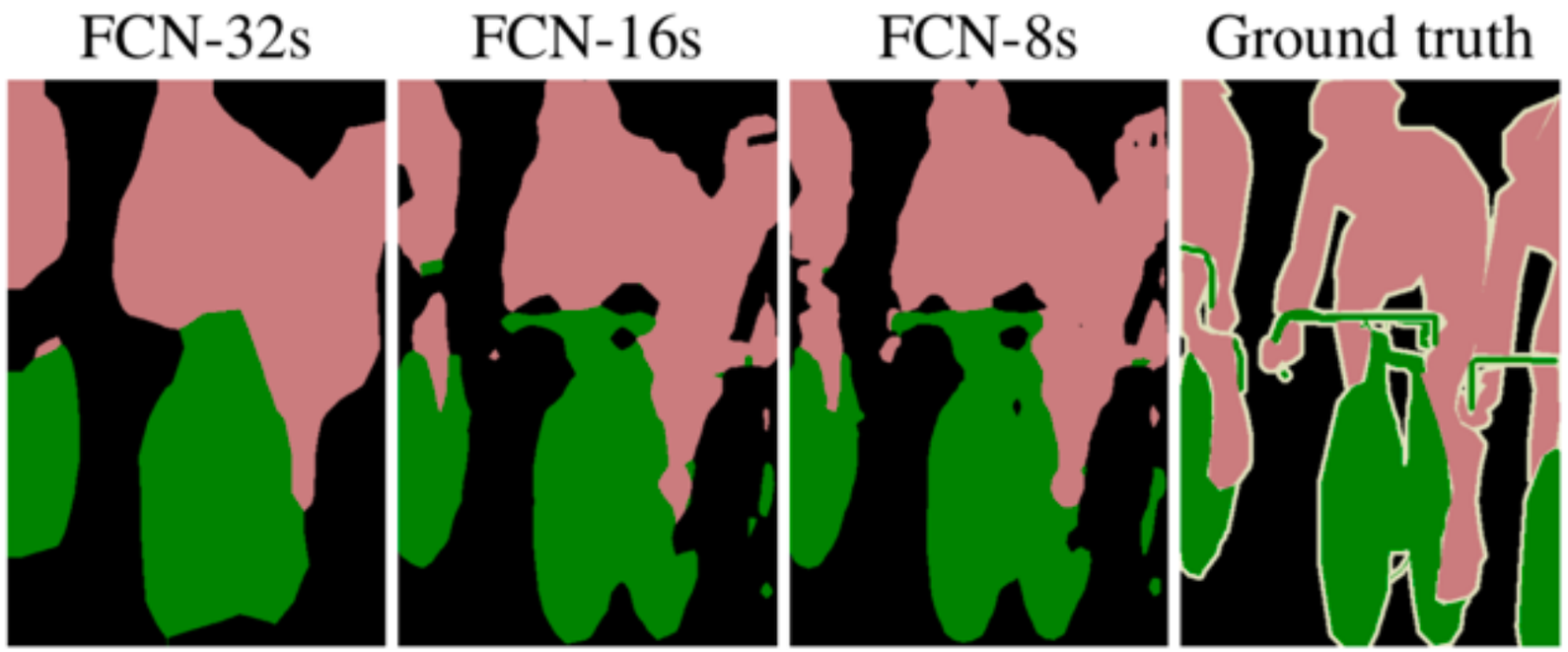



```
image_net = vgg_net(weights, processed_image)
conv_final_layer = image_net["conv5_3"]

pool5 = utils.max_pool_2x2(conv_final_layer)

W6 = utils.weight_variable([7, 7, 512, 4096], name="W6")
b6 = utils.bias_variable([4096], name="b6")
conv6 = utils.conv2d_basic(pool5, W6, b6)
relu6 = tf.nn.relu(conv6, name="relu6")
if FLAGS.debug:
    utils.add_activation_summary(relu6)
relu_dropout6 = tf.nn.dropout(relu6, keep_prob=keep_prob)

W7 = utils.weight_variable([1, 1, 4096, 4096], name="W7")
b7 = utils.bias_variable([4096], name="b7")
conv7 = utils.conv2d_basic(relu_dropout6, W7, b7)
relu7 = tf.nn.relu(conv7, name="relu7")
if FLAGS.debug:
    utils.add_activation_summary(relu7)
relu_dropout7 = tf.nn.dropout(relu7, keep_prob=keep_prob)

W8 = utils.weight_variable([1, 1, 4096, NUM_OF_CLASSESS], name="W8")
b8 = utils.bias_variable([NUM_OF_CLASSESS], name="b8")
conv8 = utils.conv2d_basic(relu_dropout7, W8, b8)
# annotation_pred1 = tf.argmax(conv8, dimension=3, name="prediction1")
```

```
# now to upscale to actual image size
deconv_shape1 = image_net["pool4"].get_shape()
W_t1 = utils.weight_variable([4, 4, deconv_shape1[3].value, NUM_OF_CLASSESS], name="W_t1")
b_t1 = utils.bias_variable([deconv_shape1[3].value], name="b_t1")
conv_t1 = utils.conv2d_transpose_strided(conv8, W_t1, b_t1, output_shape=tf.shape(image_net["pool4"]))
fuse_1 = tf.add(conv_t1, image_net["pool4"], name="fuse_1")

deconv_shape2 = image_net["pool3"].get_shape()
W_t2 = utils.weight_variable([4, 4, deconv_shape2[3].value, deconv_shape1[3].value], name="W_t2")
b_t2 = utils.bias_variable([deconv_shape2[3].value], name="b_t2")
conv_t2 = utils.conv2d_transpose_strided(fuse_1, W_t2, b_t2, output_shape=tf.shape(image_net["pool3"]))
fuse_2 = tf.add(conv_t2, image_net["pool3"], name="fuse_2")

shape = tf.shape(image)
deconv_shape3 = tf.stack([shape[0], shape[1], shape[2], NUM_OF_CLASSESS])
W_t3 = utils.weight_variable([16, 16, NUM_OF_CLASSESS, deconv_shape2[3].value], name="W_t3")
b_t3 = utils.bias_variable([NUM_OF_CLASSESS], name="b_t3")
conv_t3 = utils.conv2d_transpose_strided(fuse_2, W_t3, b_t3, output_shape=deconv_shape3, stride=8)

annotation_pred = tf.argmax(conv_t3, dimension=3, name="prediction")
```

Just Add

Skip Layer



Pascal VOC 2011 Results

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

Results

Results

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

PASCAL VOC 2011

Results