R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢

### 1. Image Semantic Segmentation

Image semantic segmentation is a computer vision task that groups pixels into meaningful or perceptually similar regions. Unlike image classification, which assigns a single label to an entire image, semantic segmentation classifies each pixel in an image into a predefined class (i.e. pixel-level classification). That is, each pixel in the image is assigned a label from a set of predefined categories. For example, in the following image, pixels might be labeled as "table", "chair", and "background".

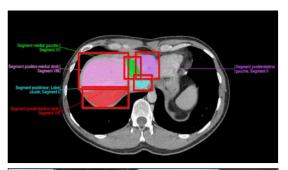




#### Applications:

- Medical Imaging: identifying and segmenting anatomical structures or pathological regions in medical scans
- Autonomous Driving: understanding and interpreting the environment by segmenting road scenes
- Agriculture: land use classification, urban planning, and disaster management







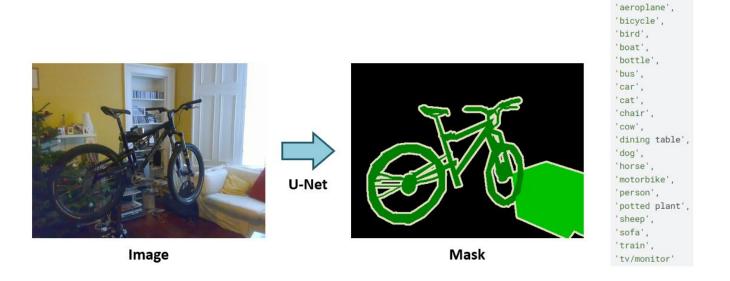
R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢

#### 2. Dataset

Pascal VOC 2012 Segmentation Dataset

• Input: RGB image

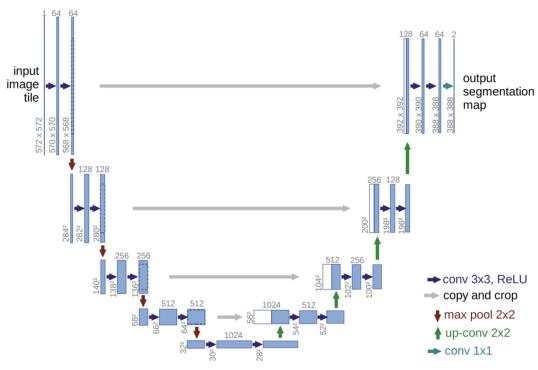
• Output: Mask with 21 possible class labels for each pixel



'background',

#### 3. U-Net Architecture

U-Net is a popular architecture for semantic segmentation. It consists of a contracting path (encoder) and an expansive path (decoder), with skip connections that combine feature maps from corresponding layers of the encoder and decoder. The key concept of the U-Net architecture is to consider features across different scales.



R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢

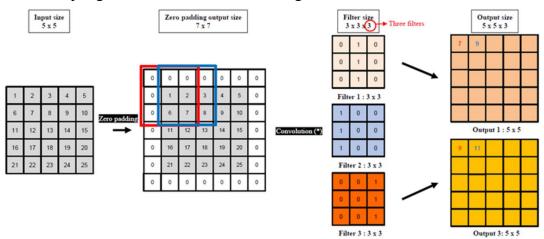
### 4. Model Training

- Develop and train the model by PyTorch
  - PyTorch is an open-source framework developed by Meta's AI research group
  - It offers an easy-to-use API and integrates seamlessly with the Python data
- Analyze the model structure and implement it
  - Down-sampling nn.Conv2d()
  - Concatenate path
    torch.cat()
  - Up-sampling
    nn.Upsample()
    nn.functional.interpolate()
    nn.ConvTranspose2d()

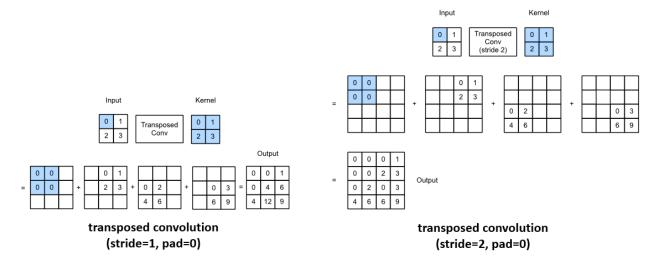
We use transposed convolution for up-sampling because it is a trainable layer.

### **Operations on U-Net**

• Down-sampling: use convolution to extract high-level information



• Up-sampling: use transposed convolution to increase the spatial resolution



R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢

#### **Model Profiling**

• Parameters: 486,813

MAC operations: 55,787,520
Input image size: 3 x 64 x 64
Output image size: 21 x 64 x 64

```
Total params: 486,613
Trainable params: 486,613
Non-trainable params: 0

Input size (MB): 0.05
Forward/backward pass size (MB): 6.84
Params size (MB): 1.86
Estimated Total Size (MB): 8.75

[INFO] Register count_convNd() for <class 'torch.nn.modules.conv.Conv2d'>.
[INFO] Register count_normalization() for <class 'torch.nn.modules.batchnorm.BatchNorm2d'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.activation.ReLU'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.conv.ConvTranspose2d'>.
INFO] Register count_convNd() for <class 'torch.nn.modules.conv.ConvTranspose2d'>.
macs:55787520.0, params:486613.0
```

| Conv2d-1   | 1                  | 0 to t 61           | D-1#    |
|--|--------------------|---------------------|---------|
| ReLU-3   | Layer (type)       | Output Shape        | Param # |
| ReLU-3   |                    | [-1, 8, 64, 64]     |         |
| Conv2d-4 [-1, 8, 64, 64] 576  BatchNorm2d-5 [-1, 18, 64, 64] 166  Conv2d-7 [-1, 116, 32, 32] 1,152  BatchNorm2d-8 [-1, 16, 32, 32] 32  ReLU-9 [-1, 16, 32, 32] 32  ReLU-10 [-1, 16, 32, 32] 32  BatchNorm2d-11 [-1, 16, 32, 32] 32  ReLU-12 [-1, 16, 32, 32] 32  ReLU-12 [-1, 16, 32, 32] 32  Conv2d-13 [-1, 16, 16] 4,608  BatchNorm2d-14 [-1, 32, 16, 16] 64  ReLU-15 [-1, 32, 16, 16] 9,216  BatchNorm2d-16 [-1, 32, 16, 16] 9,216  BatchNorm2d-17 [-1, 32, 16, 16] 64  ReLU-18 [-1, 32, 16, 16] 64  ReLU-19 [-1, 48, 8] 18,432  BatchNorm2d-20 [-1, 64, 8, 8] 128  ReLU-21 [-1, 64, 8, 8] 36,864  ReLU-24 [-1, 64, 8, 8] 36,864  BatchNorm2d-25 [-1, 128, 4, 4] 73,728  BatchNorm2d-26 [-1, 128, 4, 4] 73,728  BatchNorm2d-27 [-1, 128, 4, 4] 256  ReLU-27 [-1, 128, 4, 4] 256  ReLU-30 [-1, 128, 4, 4] 256  ReLU-31 [-1, 64, 8, 8] 32,768  BatchNorm2d-33 [-1, 64, 8, 8] 32,768  ReLU-34 [-1, 64, 8, 8] 32,768  BatchNorm2d-35 [-1, 128, 4, 4] 256  ReLU-36 [-1, 128, 4, 4] 256  ReLU-37 [-1, 128, 4, 4] 256  ReLU-38 [-1, 128, 4, 4] 256  ReLU-39 [-1, 128, 4, 4] 256  ReLU-39 [-1, 128, 4, 4] 256  ReLU-37 [-1, 64, 8, 8] 32,768  Conv2d-39 [-1, 128, 4, 4] 28  Conv2d-39 [-1, 128, 4, 4] 28  Conv2d-39 [-1, 128, 4, 4] 28  ReLU-41 [-1, 64, 8, 8] 36,864  ReLU-41 [-1, 64, 8, 8] 36,864  ReLU-44 [-1, 32, 16, 16] 9,216  Conv2d-42 [-1, 16, 32, 32] 32  ReLU-44 [-1, 32, 16, 16] 9,216  Conv2d-49 [-1, 16, 32, 32] 32  ReLU-40 [-1, 16, 32, 32] 32  ReLU-41 [-1, 64, 8, 8] 36,864  ReLU-44 [-1, 32, 16, 16] 9,216  ReLU-45 [-1, 16, 32, 32] 32  ReLU-46 [-1, 16, 32, 32] 32  ReLU-47 [-1, 16, 32, 32] 32  ReLU-48 [-1, 16, 32, 32] 32  ReLU-49 [-1, 16, 32, 32] 32  ReLU-49 [-1, 16, 32, 32] 32  ReLU-40 [-1, 16, 32, 32] 32  ReLU-41 [-1, 16, 32, 32] 32  ReLU-42 [-1, 16, 32, 32] 32  ReLU-43 [-1, 16, 32, 32] 32  ReLU-44 [-1, 32, 16, 16] 32  ReLU-45 [-1, 16, 46, 46] 35  ReLU-56 [-1, 16, 46, 46] 35  R |                    | [-1, 8, 64, 64]     |         |
| ReLU-6 [-1, 8, 64, 64] 16 Conv2d-7 [-1, 16, 32, 32] 1,152 RatchNorm2d-8 [-1, 16, 32, 32] 2,304 ReLU-9 [-1, 16, 32, 32] 0,0 Conv2d-10 [-1, 16, 32, 32] 0,0 RelU-12 [-1, 16, 32, 32] 0,0 RelU-12 [-1, 16, 32, 32] 0,0 Conv2d-13 [-1, 16, 32, 32] 0,0 Conv2d-13 [-1, 16, 32, 32] 0,0 Conv2d-13 [-1, 16, 32, 32] 0,0 RelU-12 [-1, 16, 32, 32] 0,0 Conv2d-13 [-1, 32, 16, 16] 64 RelU-15 [-1, 32, 16, 16] 64 RelU-15 [-1, 32, 16, 16] 64 RelU-18 [-1, 32, 16, 16] 9,216 RelU-18 [-1, 32, 16, 16] 64 RelU-19 [-1, 64, 8, 8] 18,432 RelU-21 [-1, 64, 8, 8] 128 RelU-22 [-1, 64, 8, 8] 128 RelU-23 [-1, 64, 8, 8] 128 RelU-27 [-1, 128, 4, 4] 73,728 RelU-27 [-1, 128, 4, 4] 256 RelU-27 [-1, 128, 4, 4] 147,456 RelU-30 [-1, 128, 4, 4] 147,456 RelU-31 [-1, 64, 8, 8] 32,768 RelU-32 [-1, 64, 8, 8] 32,768 RelU-34 [-1, 128, 4, 4] 147,456 RelU-35 [-1, 128, 4, 4] 147,456 RelU-36 [-1, 128, 4, 4] 147,456 RelU-37 [-1, 128, 4, 4] 147,456 RelU-38 [-1, 128, 4, 4] 147,456 RelU-39 [-1, 128, 4, 4] 147,456 RelU-30 [-1, 128, 4, 4] 147,456 RelU-31 [-1, 64, 8, 8] 32,768 RelU-32 [-1, 64, 8, 8] 32,768 RelU-34 [-1, 128, 4, 4] 147,456 RelU-37 [-1, 64, 8, 8] 32,768 RelU-39 [-1, 128, 4, 4] 147,456 RelU-39 [-1, 128, 4, 4] 147,456 RelU-30 [-1, 128, 4, 4] 147,456 RelU-31 [-1, 64, 8, 8] 36,864 RelU-37 [-1, 64, 8, 8] 36,864 RelU-38 [-1, 64, 8, 8] 36,864 RelU-41 [-1, 32, 16, 16] 64 RelU-44 [-1, 32, 16, 16] 64 RelU-45 [-1, 16, 32, 32] 2, 344 RelU-50 [-1, 16, 32, 32] 3, 34 RelU-50 [-1, 16, 32, 32] 3, 34 RelU-50 [-1, 16, 46, 46] 576 RelU-50 [-1,  |                    | [-1, 8, 64, 64]     |         |
| ReLU-6 CONV2d-7 RetCU-9 RetCU-9 RetCU-9 RetCU-9 RetCU-9 RetCU-9 RetCU-9 RetCU-9 RetCU-11 RetCU-12 RetCU-13 RetCU-13 RetCNorm2d-11 RetCU-15 RetCU-15 RetCU-15 RetCU-15 RetCU-15 RetCU-15 RetCU-15 RetCU-16 RetCU-16 RetCU-16 RetCU-17 RetCU-18 RetCU-18 RetCU-18 RetCU-18 RetCU-18 RetCU-18 RetCU-18 RetCU-18 RetCU-19 RetCU-19 RetCU-19 RetCU-19 RetCU-10 RetCU-10 RetCU-10 RetCU-11 RetCU-11 RetCU-11 RetCU-12 RetCU-21 RetCU-22 RetCU-24 RetCU-24 RetCU-24 RetCU-25 RetCU-27 RetCU-26 RetCU-27 RetCU-28 RetCU-27 RetCU-28 RetCU-28 RetCU-30 RetCU-31 RetCU-30 RetCU-31 RetCU-31 RetCU-31 RetCU-34 RetCU-34 RetCU-37 RetCU-36 RetCU-37 RetCU-38 RetCU-37 RetCU-30 RetCU-31 RetCU-31 RetCU-34 RetCU-34 RetCU-34 RetCU-35 RetCU-37 RetCU-38 RetCU-39 RetCU-31 RetCU-34 RetCU-44 RetCU-44 RetCU-44 RetCU-44 RetCU-44 RetCU-45 RetCU-49 RetCU-49 RetCU-49 RetCU-49 RetCU-49 RetCU-49 RetCU-40 RetCU-50 |                    | [-1, 0, 64, 64]     |         |
| Conv2d-7  BatchNorm2d-8  ReLU-9  ReLU-16, 32, 32]  ReLU-17  BatchNorm2d-11  Conv2d-10  BatchNorm2d-11  Conv2d-13  BatchNorm2d-14  ReLU-15  Conv2d-16  BatchNorm2d-14  ReLU-15  Conv2d-16  Conv2d-16  BatchNorm2d-14  Conv2d-16  ReLU-17  Conv2d-16  Conv2d-16  ReLU-18  Conv2d-17  ReLU-18  Conv2d-19  BatchNorm2d-17  ReLU-18  Conv2d-19  Conv2d-19  Conv2d-19  Conv2d-19  BatchNorm2d-20  Conv2d-19  Conv2d-21  BatchNorm2d-20  Conv2d-21  ReLU-21  Conv2d-22  BatchNorm2d-20  Conv2d-22  Lonuxd-23  ReLU-24  Conv2d-25  ReLU-27  Conv2d-28  BatchNorm2d-28  ReLU-27  Conv2d-28  BatchNorm2d-29  Conv2d-28  BatchNorm2d-29  Conv2d-28  BatchNorm2d-29  Conv2d-29  ReLU-30  Conv2d-31  Conv2d-31  Conv2d-31  Conv2d-32  ReLU-30  Conv2d-33  ReLU-30  Conv2d-34  ReLU-30  Conv2d-35  ReLU-30  Conv2d-35  ReLU-30  Conv2d-36  ReLU-30  Conv2d-37  ReLU-30  Conv2d-38  ReLU-30  Conv2d-39  ReLU-30  Conv2d-35  ReLU-30  Conv2d-35  ReLU-30  Conv2d-36  ReLU-37  ReLU-39  Conv2d-38  ReLU-30  Conv2d-39  ReLU-30  Conv2d-39  ReLU-30  Conv2d-39  ReLU-30  ReLU-41  Conv2d-42  ReLU-41  Lonuxd-44  Lonuxd-44  ReLU-37  Conv2d-49  ReLU-44  Conv2d-49  ReLU-41  ReLU-44  ReLU-46  ReLU-46  ReLU-47  ReLU-47  ReLU-48  ReLU-48  ReLU-49  ReLU |                    | 1-1. 8. 64. 641     |         |
| BatchNorm2d-8 ReLU-9 Conv2d-10 Conv2d-10 RetU-11 RetU-12 RetU-12 RetU-12 RetU-13 RetU-15 RetU-15 RetU-15 RetU-15 RetU-15 RetU-15 RetU-15 RetU-16 RetU-15 RetU-16 RetU-17 RetU-18 RetU-18 RetU-18 RetU-18 RetU-18 RetU-19 RetU-19 RetU-19 RetU-19 RetU-19 RetU-18 RetU-19 RetU-27 RetU-27 RetU-27 RetU-27 RetU-27 RetU-27 RetU-29 RetU-29 RetU-29 RetU-29 RetU-29 RetU-30 RetU-31 RetU-31 RetU-34 RetU-35 RetU-48 RetU-41 RetU-44 RetU-45 RetU-48 RetU-48 RetU-48 RetU-49 RetU-50 RetU- |                    | [-1, 16, 32, 32]    |         |
| ReLU-9 Conv2d-10 Conv2d-11 ReLU-12 ReLU-12 Conv2d-13 ReLU-15 Conv2d-13 ReLU-15 ReLU-16, 32, 32] Conv2d-13 ReLU-16, 16, 32, 32] Conv2d-13 RelU-17 RelU-18 RelU-18 Conv2d-16 RelU-18 Conv2d-19 RelU-18 Conv2d-19 RelU-11 Conv2d-19 RelU-21 RelU-21 RelU-11 RelU-21 RelU-11 RelU-21 RelU-24 RelU-24 RelU-24 RelU-24 RelU-24 RelU-27 RelU-27 RelU-27 RelU-27 RelU-30 RelU-30 Conv2d-28 RelU-30 Conv2d-29 RelU-30 Conv2d-29 RelU-30 Conv2d-29 RelU-30 Conv2d-32 RelU-30 Conv2d-32 RelU-30 Conv2d-32 RelU-30 Conv2d-32 RelU-30 Conv2d-33 RelU-34 RelU-34 RelU-34 RelU-35 RelU-34 RelU-36 RelU-37 RelU-37 RelU-38 RelU-38 RelU-38 RelU-39 RelU-39 RelU-39 RelU-30 Conv2d-39 RelU-30 RelU-30 RelU-30 RelU-31 RelU-31 RelU-31 RelU-32 RelU-34 RelU-34 RelU-35 RelU-36 RelU-37 RelU-37 RelU-38 RelU-38 RelU-38 RelU-39 RelU-39 RelU-39 RelU-30 RelU-31 RelU-31 RelU-31 RelU-31 RelU-32 RelU-34 RelU-34 RelU-35 RelU-36 RelU-48 RelU-41 RelU-48 RelU-48 RelU-48 RelU-48 RelU-48 RelU-48 RelU-48 RelU-48 RelU-48 RelU-50 RelU-51 RelU-51 RelU-52 RelU-55 RelU-57 RelU-58 RelU-56 RelU-57 RelU-58 RelU-57 RelU-58 RelU-58 RelU-58 RelU-59 R |                    | [-1, 16, 32, 32]    | 32      |
| BatchNorm2d-11   |                    | [-1, 16, 32, 32]    |         |
| ReLU-12 [-1, 16, 32, 32] 0 Conv2d-13 [-1, 32, 16, 16] 4,608 BatchNorm2d-14 [-1, 32, 16, 16] 64 ReLU-15 [-1, 32, 16, 16] 9,216 BatchNorm2d-17 [-1, 32, 16, 16] 64 ReLU-18 [-1, 32, 16, 16] 9,216 BatchNorm2d-20 [-1, 64, 8, 8] 18,432 BatchNorm2d-20 [-1, 64, 8, 8] 128 ReLU-21 [-1, 64, 8, 8] 128 ReLU-21 [-1, 64, 8, 8] 128 ReLU-24 [-1, 64, 8, 8] 128 ReLU-24 [-1, 64, 8, 8] 128 ReLU-25 [-1, 128, 4, 4] 73,728 BatchNorm2d-26 [-1, 128, 4, 4] 256 ReLU-27 [-1, 128, 4, 4] 256 BatchNorm2d-29 [-1, 128, 4, 4] 256 ReLU-30 [-1, 128, 4, 4] 256 ReLU-30 [-1, 128, 4, 4] 256 ReLU-30 [-1, 128, 4, 4] 0 ConvTranspose2d-31 [-1, 64, 8, 8] 32,768 Conv2d-32 [-1, 64, 8, 8] 73,728 BatchNorm2d-33 [-1, 64, 8, 8] 73,728 BatchNorm2d-36 [-1, 64, 8, 8] 73,728 ReLU-37 [-1, 128, 4, 4] 0 Conv2d-35 [-1, 64, 8, 8] 32,768 ReLU-36 [-1, 64, 8, 8] 32,768 ReLU-37 [-1, 128, 4, 4] 0 Conv2d-39 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 36,864 BatchNorm2d-36 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 36,864 BatchNorm2d-36 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 36,864 BatchNorm2d-36 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 36,864 BatchNorm2d-36 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 36,864 ReLU-41 [-1, 64, 8, 8] 36,864 ReLU-41 [-1, 64, 8, 8] 36,864 ReLU-41 [-1, 64, 8, 8] 36,864 ReLU-44 [-1, 32, 16, 16] 64 ReLU-44 [-1, 32, 16, 16] 64 ReLU-48 [-1, 32, 16, 16] 64 ReLU-48 [-1, 32, 16, 16] 64 ReLU-48 [-1, 16, 32, 32] 32 ReLU-50 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 ReLU-52 [-1, 8, 64, 64] 576 ReLU-58 [-1, 8, 64, 64] 576   |                    | [-1, 16, 32, 32]    |         |
| Conv2d-13  |                    | [-1, 16, 32, 32]    |         |
| BatchNorm2d-14 ReLU-15 ReLU-15 Conv2d-16 BatchNorm2d-17 ReLU-18 ReLU-18 Conv2d-19 ReLU-19 Conv2d-19 ReLU-21 ReLU-21 ReLU-21 ReLU-21 ReLU-21 ReLU-21 ReLU-21 ReLU-22 ReLU-24 ReLU-24 ReLU-24 ReLU-25 ReLU-25 ReLU-27 ReLU-27 ReLU-27 ReLU-27 ReLU-27 ReLU-30 Relu-40 Relu-41 Relu-41 Relu-44 Relu-48 Relu-44 Relu-48 Relu-44 Relu-48 Relu-49 Relu-48 Relu-49 Relu-48 Relu-49 Relu-48 Relu-49 Relu-49 Relu-48 Relu-49 Relu-49 Relu-49 Relu-49 Relu-49 Relu-50 Re |                    | [-1, 16, 32, 32]    |         |
| ReLU-15 Conv2d-16 BatchNorm2d-17 ReLU-18 ReLU-18 Conv2d-19 [-1, 32, 16, 16] Conv2d-19 [-1, 64, 8, 8] BatchNorm2d-20 [-1, 64, 8, 8] ReLU-21 Conv2d-22 [-1, 64, 8, 8] ReLU-21 [-1, 64, 8, 8] ReLU-24 [-1, 64, 8, 8] ReLU-24 [-1, 64, 8, 8] ReLU-25 ReLU-24 [-1, 64, 8, 8] ReLU-26 Conv2d-27 ReLU-27 Conv2d-27 ReLU-27 Conv2d-28 ReLU-27 Conv2d-28 ReLU-27 ReLU-30 Conv2d-30 ReLU-30 Conv2d-31 ReLU-30 Conv2d-32 ReLU-30 Conv2d-32 ReLU-34 Conv2d-33 ReLU-34 ReLU-35 ReLU-36 ReLU-37 Conv2d-38 ReLU-37 Conv2d-38 ReLU-38 ReLU-39 Conv2d-39 ReLU-30 Conv2d-30 ReLU-30 Conv2d-30 ReLU-30 Conv2d-30 ReLU-31 ReLU-32 ReLU-34 ReLU-35 ReLU-37 Conv2d-36 ReLU-37 Conv2d-38 ReLU-37 Conv2d-39 ReLU-39 Conv2d-39 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-44 ReLU-45 ReLU-48 ReLU-48 ReLU-49 ReLU-50  |                    | [-1, 32, 16, 16]    |         |
| Conv2d-16 BatchNorm2d-17 RetU-18 RetU-18 Conv2d-19 Conv2d-19 BatchNorm2d-20 RetU-21 Conv2d-22 [-1, 64, 8, 8] Conv2d-22 [-1, 64, 8, 8] RetU-21 Conv2d-23 BatchNorm2d-23 RetU-24 Conv2d-25 BatchNorm2d-26 RetU-27 Conv2d-26 BatchNorm2d-26 RetU-27 Conv2d-28 BatchNorm2d-29 RetU-30 Conv2d-29 RetU-30 Conv2d-31 BatchNorm2d-29 RetU-30 Conv2d-31 BatchNorm2d-33 RetU-30 Conv2d-32 BatchNorm2d-33 RetU-34 Conv2d-35 BatchNorm2d-35 RetU-36 Conv2d-37 BatchNorm2d-38 RetU-37 Conv2d-38 BatchNorm2d-38 RetU-34 Conv2d-35 BatchNorm2d-36 BatchNorm2d-36 RetU-37 Conv2d-38 BatchNorm2d-36 BatchNorm2d-39 RetU-37 Conv2d-39 RetU-37 Conv2d-39 BatchNorm2d-39 RetU-41 Conv2d-40 RetU-41 Conv2d-40 RetU-41 Conv2d-40 RetU-41 Conv2d-40 RetU-41 Conv2d-40 RetU-44 RetU-45 RetU-48 RetU-48 RetU-48 RetU-48 RetU-48 RetU-48 RetU-48 RetU-48 RetU-55 RetU-56 RetU-56 RetU-56 RetU-56 RetU-57 RetU-58 RetU-56 RetU-57 RetU-58 RetU-58 RetU-56 RetU-57 RetU-58 RetU-58 RetU-57 RetU-58 |                    | 1-1-32-16-161       |         |
| BatchNorm2d-17 ReLU-18 Conv2d-19 BatchNorm2d-20 ReLU-21 Conv2d-21 ReLU-21 Conv2d-22 Conv2d-22 Conv2d-22 Conv2d-23 ReLU-24 Conv2d-25 ReLU-25 ReLU-27 Conv2d-25 ReLU-27 Conv2d-26 ReLU-77 Conv2d-28 ReLU-77 Conv2d-28 ReLU-78 ReLU-79 Conv2d-29 ReLU-30 Conv2d-29 ReLU-30 Conv2d-30 ReLU-30 Conv2d-30 ReLU-30 Conv2d-32 ReLU-30 Conv2d-32 ReLU-30 Conv2d-32 ReLU-34 ReLU-34 ReLU-35 ReLU-36 ReLU-37 Conv2d-38 ReLU-38 ReLU-39 Conv2d-39 ReLU-30 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-44 ReLU-45 ReLU-46 ReLU-47 ReLU-48 ReLU-48 ReLU-49 ReLU-50  | Conv2d-16          | 1-1, 32, 16, 161    | 9,216   |
| ReLU-18 Conv2d-19 BatchNorm2d-20 ReLU-21 Conv2d-22 [-1, 64, 8, 8] ReLU-21 Conv2d-22 [-1, 64, 8, 8] ReLU-24 [-1, 64, 8, 8] ReLU-24 [-1, 64, 8, 8] ReLU-25 ReLU-26 ReLU-27 [-1, 64, 8, 8] ReLU-27 [-1, 64, 8, 8] ReLU-27 [-1, 64, 8, 8] ReLU-28 ReLU-29 Conv2d-25 ReLU-27 [-1, 128, 4, 4] ReLU-27 [-1, 128, 4, 4] ReLU-30 Conv2d-28 ReLU-30 Conv2d-28 ReLU-30 Conv2d-31 ReLU-30 Conv2d-32 ReLU-34 ReLU-34 ReLU-34 ReLU-34 ReLU-34 ReLU-34 ReLU-37 Conv2d-35 ReLU-37 Conv2d-35 ReLU-37 Conv2d-38 ReLU-37 Conv2d-39 ReLU-37 Conv2d-39 ReLU-37 Conv2d-39 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-44 ReLU-48 ReLU-49 ReLU-49 ReLU-49 ReLU-49 ReLU-49 ReLU-49 ReLU-40 ReLU-40 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-42 ReLU-43 ReLU-44 ReLU-44 ReLU-44 ReLU-44 ReLU-45 ReLU-46 ReLU-47 ReLU-48 ReLU-48 ReLU-48 ReLU-49 ReLU-51 ReLU-51 ReLU-51 ReLU-55 ReLU-56 ReLU-56 ReLU-56 ReLU-57 ReLU-58 ReLU-57 ReLU-58 ReLU-58 ReLU-58 ReLU-59 Re |                    | [-1, 32, 16, 16]    | 64      |
| Conv2d-22  |                    | [-1, 32, 16, 16]    |         |
| Conv2d-22  |                    | [-1, 64, 8, 8]      |         |
| Conv2d-22  |                    | [-1, 64, 8, 8]      |         |
| BatchNorm2d-23 ReLU-24 [-1, 64, 8, 8] Conv2d-25 [-1, 128, 4, 4] ReLU-27 [-1, 128, 4, 4] Conv2d-26 ReLU-27 [-1, 128, 4, 4] Conv2d-28 BatchNorm2d-29 [-1, 128, 4, 4] ReLU-30 Conv2d-28 [-1, 128, 4, 4] ReLU-30 [-1, 128, 4, 4] ReLU-30 Conv7ranspose2d-31 [-1, 64, 8, 8] Conv2d-32 [-1, 64, 8, 8] ReLU-34 [-1, 64, 8, 8] ReLU-34 [-1, 64, 8, 8] ReLU-34 [-1, 64, 8, 8] ReLU-37 Conv2d-35 [-1, 64, 8, 8] ReLU-37 Conv2d-39 [-1, 64, 8, 8] ReLU-37 Conv2d-39 [-1, 32, 16, 16] ReLU-41 ReLU-44 [-1, 32, 16, 16] ReLU-48 ReLU-49 ReLU-49 ReLU-49 ReLU-49 ReLU-49 ReLU-49 ReLU-49 ReLU-40 ReLU-40 ReLU-41 ReLU-41 ReLU-41 ReLU-42 ReLU-43 ReLU-44 ReLU-44 ReLU-45 ReLU-45 ReLU-46 ReLU-47 ReLU-48 ReLU-48 ReLU-49 ReLU-51 ReLU-51 ReLU-51 ReLU-51 ReLU-51 ReLU-55 ReLU-55 ReLU-56 ReLU-56 ReLU-57 ReLU-57 ReLU-58 ReLU-57 ReLU-58 ReLU-58 ReLU-59 ReLU |                    | [-1, 04, 0, 0]      |         |
| ReLU-24 Conv2d-25  |                    | [-1, 64, 8, 8]      |         |
| Conv2d-25  |                    | 1 - 1 - 64 - 8 - 81 |         |
| ReLU-27 [-1, 128, 4, 4] 256 ReLU-27 [-1, 128, 4, 4] 147,456 ReLU-30 [-1, 128, 4, 4] 256 ReLU-30 [-1, 64, 8, 8] 32,768 Conv2d-32 [-1, 64, 8, 8] 73,728 ReLU-34 [-1, 64, 8, 8] 128 ReLU-34 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 36,864 ReLU-37 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 9 Conv2d-39 [-1, 32, 16, 16] 8,192 Conv2d-39 [-1, 32, 16, 16] 64 ReLU-41 [-1, 32, 16, 16] 64 ReLU-41 [-1, 32, 16, 16] 64 ReLU-41 [-1, 32, 16, 16] 64 ReLU-44 [-1, 32, 16, 16] 64 ReLU-48 [-1, 16, 32, 32] 2,048 BatchNorm2d-47 [-1, 16, 32, 32] 32 ReLU-48 [-1, 16, 32, 32] 32 ReLU-48 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 Conv2d-53 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 ReLU-52 [-1, 8, 64, 64] 1,152 ReLU-55 [-1, 8, 64, 64] 576 ReLU-56 [-1, 8, 64, 64] 576 ReLU-57 [-1, 8, 64, 64] 576 ReLU-58 [-1, 8, 64, 64] 64   | Conv2d-25          | [-1, 128, 4, 4]     | 73,728  |
| ReLU-27 Conv2d-28 BatchNorm2d-29 ReLU-30 ConvTranspose2d-31 Conv2d-32 BatchNorm2d-33 BatchNorm2d-35 ReLU-34 Conv2d-35 BatchNorm2d-36 BatchNorm2d-36 BatchNorm2d-38 BatchNorm2d-39 Conv2d-39 Conv2d-39 BatchNorm2d-40 ReLU-41 Conv2d-42 BatchNorm2d-40 ReLU-41 Conv2d-42 BatchNorm2d-43 BatchNorm2d-44 ReLU-44 Conv2d-45 BatchNorm2d-46 BatchNorm2d-47 Conv2d-47 BatchNorm2d-47 BatchNorm2d-47 BatchNorm2d-47 BatchNorm2d-47 Conv2d-48 BatchNorm2d-47 BatchNorm2d-49 BatchNorm2d-47 Conv2d-48 BatchNorm2d-47 Conv2d-48 BatchNorm2d-47 Conv2d-49 BatchNorm2d-47 Conv2d-48 BatchNorm2d-47 Conv2d-48 BatchNorm2d-47 Conv2d-49 BatchNorm2d-47 Conv2d-49 BatchNorm2d-47 Conv2d-49 BatchNorm2d-50 BatchNorm2d-50 BatchNorm2d-50 Conv2d-50 Conv2d-50 BatchNorm2d-50 Conv2d-51 BatchNorm2d-50 Conv2d-53 BatchNorm2d-54 ReLU-55 Conv2d-56 BatchNorm2d-57 ReLU-55 Conv2d-56 BatchNorm2d-57 ReLU-58 BatchNorm2d-59 BatchNorm2d-59 BatchNorm2 |                    | [-1, 128, 4, 4]     |         |
| ReLU-30 ConvTranspose2d-31 Conv2d-32 ReLU-34 ReLU-34 Conv2d-35 ReLU-36 ReLU-37 ReLU-37 ReLU-38 ReLU-39 Conv2d-39 ReLU-39 Conv2d-39 ReLU-39 ReLU-39 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-44 Relu-48 Relu-50 Relu-50 Relu-51 Relu-51 Relu-55 Relu-55 Relu-55 Relu-55 Relu-55 Relu-55 Relu-55 Relu-56 Relu-57 Relu-58 Relu-57 Relu-58 Relu-58 Relu-58 Relu-57 Relu-58 Relu-58 Relu-58 Relu-58 Relu-58 Relu-59 Rel | ReLU-27            | [-1, 128, 4, 4]     |         |
| ReLU-30 ConvTranspose2d-31 Conv2d-32 ReLU-34 ReLU-34 Conv2d-35 ReLU-36 ReLU-37 ReLU-37 ReLU-38 ReLU-39 Conv2d-39 ReLU-39 Conv2d-39 ReLU-39 ReLU-39 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-41 ReLU-44 Relu-48 Relu-50 Relu-50 Relu-51 Relu-51 Relu-55 Relu-55 Relu-55 Relu-55 Relu-55 Relu-55 Relu-55 Relu-56 Relu-57 Relu-58 Relu-57 Relu-58 Relu-58 Relu-58 Relu-57 Relu-58 Relu-58 Relu-58 Relu-58 Relu-58 Relu-59 Rel |                    | [-1, 128, 4, 4]     |         |
| ConvTranspose2d-31   |                    | [-1, 128, 4, 4]     |         |
| Conv2d-32  |                    | [-1, 120, 4, 4]     |         |
| BatchNorm2d-33   |                    | [-1, 64, 8, 8]      |         |
| ReLU-34 [-1, 64, 8, 8] 0 Conv2d-35 [-1, 64, 8, 8] 36,864  BatchNorm2d-36 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 128 ReLU-37 [-1, 64, 8, 8] 128 Conv2d-39 [-1, 32, 16, 16] 8,192 Conv2d-39 [-1, 32, 16, 16] 18,432 BatchNorm2d-40 [-1, 32, 16, 16] 64 ReLU-41 [-1, 32, 16, 16] 9,216 BatchNorm2d-43 [-1, 32, 16, 16] 64 ReLU-44 [-1, 32, 16, 16] 64 ReLU-44 [-1, 32, 16, 16] 64 ConvTranspose2d-45 [-1, 16, 32, 32] 2,048 Conv2d-46 [-1, 16, 32, 32] 4,608 BatchNorm2d-47 [-1, 16, 32, 32] 32 ReLU-48 [-1, 16, 32, 32] 32 ReLU-48 [-1, 16, 32, 32] 32 ReLU-49 [-1, 16, 32, 32] 32 Conv2d-49 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 Conv2d-50 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 Conv2d-53 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 1,152 ReLU-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 ReLU-57 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 0  |                    | 1 64 8 81           |         |
| Conv2d-35  |                    | [-1, 64, 8, 8]      |         |
| BatchNorm2d-36 ReLU-37 [-1, 64, 8, 8] 0 ConvTranspose2d-38 [-1, 32, 16, 16] Conv2d-39 [-1, 32, 16, 16] BatchNorm2d-40 ReLU-41 [-1, 32, 16, 16] Conv2d-42 [-1, 32, 16, 16] BatchNorm2d-43 ReLU-44 [-1, 32, 16, 16] Conv7ranspose2d-45 ReLU-44 [-1, 32, 16, 16] Conv7ranspose2d-45 [-1, 16, 32, 32] Conv2d-46 BatchNorm2d-47 [-1, 16, 32, 32] ReLU-48 Conv2d-49 [-1, 16, 32, 32] Conv2d-49 BatchNorm2d-50 ReLU-51 Conv7ranspose2d-52 [-1, 16, 32, 32] ReLU-51 Conv7ranspose2d-53 ReLU-51 Conv7ranspose2d-54 [-1, 16, 32, 32] ReLU-51 Conv7ranspose2d-55 [-1, 8, 64, 64] ReLU-55 Conv2d-56 ReLU-55 [-1, 8, 64, 64] BatchNorm2d-57 ReLU-58 [-1, 8, 64, 64] ReLU-57 ReLU-58 [-1, 8, 64, 64] ReLU-58 [-1, 8, 64, 64] ReLU-59 ReLU-57 ReLU-58 [-1, 8, 64, 64] |                    | i-1, 64, 8, 8i      |         |
| ReLU-37 [-1, 64, 8, 8] 0 ConvTranspose2d-38 [-1, 32, 16, 16] 8,192 Conv2d-39 [-1, 32, 16, 16] 18,432 BatchNorm2d-40 [-1, 32, 16, 16] 64 ReLU-41 [-1, 32, 16, 16] 9,216 BatchNorm2d-43 [-1, 32, 16, 16] 64 ReLU-44 [-1, 32, 16, 16] 64 ReLU-44 [-1, 32, 16, 16] 64 ConvTranspose2d-45 [-1, 16, 32, 32] 2,048 Conv2d-46 [-1, 16, 32, 32] 4,608 BatchNorm2d-47 [-1, 16, 32, 32] 32 ReLU-48 [-1, 16, 32, 32] 32 ReLU-48 [-1, 16, 32, 32] 32 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 ConvTranspose2d-52 [-1, 16, 32, 32] 32 ReLU-51 [-1, 16, 32, 32] 32 Conv2d-53 [-1, 16, 32, 32] 32 BatchNorm2d-54 [-1, 16, 32, 32] 32 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 1,152 BatchNorm2d-56 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 16 BatchNorm2d-57 [-1, 8, 64, 64] 16 ReLU-57 [-1, 8, 64, 64] 16 ReLU-58 [-1, 8, 64, 64] 16 ReLU-59 [-1, 8, 64, 64] 16   |                    | [-1, 64, 8, 8]      |         |
| ConvTranspose2d-38   |                    | [-1, 64, 8, 8]      |         |
| Conv2d-39 [-1, 32, 16, 16] 18,432  BatchNorm2d-40 [-1, 32, 16, 16] 64  ReLU-41 [-1, 32, 16, 16] 0  Conv2d-42 [-1, 32, 16, 16] 9,216  BatchNorm2d-43 [-1, 32, 16, 16] 64  ReLU-44 [-1, 32, 16, 16] 0  ConvTranspose2d-45 [-1, 16, 32, 32] 2,048  BatchNorm2d-47 [-1, 16, 32, 32] 4,608  BatchNorm2d-47 [-1, 16, 32, 32] 32  ReLU-48 [-1, 16, 32, 32] 32  Conv2d-49 [-1, 16, 32, 32] 2,304  BatchNorm2d-50 [-1, 16, 32, 32] 2,304  BatchNorm2d-50 [-1, 16, 32, 32] 32  Conv2d-53 [-1, 16, 32, 32] 32  Conv2d-53 [-1, 8, 64, 64] 512  Conv2d-54 [-1, 8, 64, 64] 1,152  BatchNorm2d-54 [-1, 8, 64, 64] 1,152  ReLU-55 [-1, 8, 64, 64] 576  BatchNorm2d-57 [-1, 8, 64, 64] 576  BatchNorm2d-57 [-1, 8, 64, 64] 0  ReLU-58 [-1, 8, 64, 64] 0   | ConvTranspose2d-38 | [-1, 32, 16, 16]    | 8,192   |
| Ret.U-48 [-1, 16, 32, 32] 0 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 16  | Conv2d-39          | [-1, 32, 16, 16]    | 18,432  |
| Ret.U-48 [-1, 16, 32, 32] 0 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 16  | BatchNorm2d-40     | [-1, 32, 16, 16]    | 64      |
| Ret.U-48 [-1, 16, 32, 32] 0 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 16  |                    | [-1, 32, 16, 16]    |         |
| Ret.U-48 [-1, 16, 32, 32] 0 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 16  |                    | [-1, 32, 16, 16]    |         |
| Ret.U-48 [-1, 16, 32, 32] 0 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 16  |                    | [-1, 32, 16, 16]    |         |
| Ret.U-48 [-1, 16, 32, 32] 0 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 16  |                    | [-1, 32, 16, 16]    |         |
| Ret.U-48 [-1, 16, 32, 32] 0 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 16  |                    | [-1, 16, 32, 32]    |         |
| Ret.U-48 [-1, 16, 32, 32] 0 Conv2d-49 [-1, 16, 32, 32] 2,304 BatchNorm2d-50 [-1, 16, 32, 32] 32 Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 16  |                    | [-1, 16, 32, 32]    |         |
| Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 0   |                    | [-1, 16, 32, 32]    |         |
| Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 0   |                    | [ 1 16 32 32]       |         |
| Ret.U-51 [-1, 16, 32, 32] 0 ConvTranspose2d-52 [-1, 8, 64, 64] 512 Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 Ret.U-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 Ret.U-58 [-1, 8, 64, 64] 0   |                    | [1, 16, 32, 32]     |         |
| ConvTranspose2d-52 [-1, 8, 64, 64] 512<br>Conv2d-53 [-1, 8, 64, 64] 1,152<br>BatchNorm2d-54 [-1, 8, 64, 64] 16<br>ReLU-55 [-1, 8, 64, 64] 0<br>Conv2d-56 [-1, 8, 64, 64] 576<br>BatchNorm2d-57 [-1, 8, 64, 64] 16<br>ReLU-58 [-1, 8, 64, 64] 0   |                    | [-1, 16, 32, 32]    |         |
| Conv2d-53 [-1, 8, 64, 64] 1,152 BatchNorm2d-54 [-1, 8, 64, 64] 16 ReLU-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 ReLU-58 [-1, 8, 64, 64] 0  |                    | [-1, 8, 64, 64]     |         |
| BatchNorm2d-54 [-1, 8, 64, 64] 16 ReLU-55 [-1, 8, 64, 64] 0 Conv2d-56 [-1, 8, 64, 64] 576 BatchNorm2d-57 [-1, 8, 64, 64] 16 ReLU-58 [-1, 8, 64, 64] 0  |                    | [-1, 8, 64, 64]     |         |
| ReLU-55 [-1, 8, 64, 64] 0<br>Conv2d-56 [-1, 8, 64, 64] 576<br>BatchNorm2d-57 [-1, 8, 64, 64] 16<br>ReLU-58 [-1, 8, 64, 64] 0   |                    | 1. 8. 64. 641       |         |
| Conv2d-56 [-1, 8, 64, 64] 576<br>BatchNorm2d-57 [-1, 8, 64, 64] 16<br>ReLU-58 [-1, 8, 64, 64] 0  |                    | [-1, 8, 64, 64]     |         |
| BatchNorm2d-57 [-1, 8, 64, 64] 16<br>ReLU-58 [-1, 8, 64, 64] 0   |                    | [-1, 8, 64, 64]     |         |
| ReLU-58 [-1, 8, 64, 64] 0  |                    | [-1, 8, 64, 64]     |         |
| Conv2d-59 [-1, 21, 64, 64] 189   |                    | [-1, 8, 64, 64]     |         |
|  | Conv2d-59          | [-1, 21, 64, 64]    | 189     |
|  |                    |                     |         |

R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢

#### **Model Training Summary**

Testing accuracy: 94.5%

```
2012/VOCtrainval 11-May-2012.tar to ./dataset/VOCtrainval_11-May-2012.tar
100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 100%| 
                                                                                                                                                                                                      super(UNet, self).
self.channel_size1
                                                                                                                                                                                                      self.channel_size2 =
                       enc1 = self.enc1(x)
enc2 = self.enc2(F.max_pool2d(enc1, 2))
enc3 = self.enc3(F.max_pool2d(enc2, 2))
                                                                                                                                                                                                      self.channel_size3 = 32
self.channel_size4 = 64
                                                                                                                                                                                                    self.enc1 = self.conv_block(self.channel_size1, self.channel_size2)
self.enc2 = self.conv_block(self.channel_size2, self.channel_size3)
self.enc3 = self.conv_block(self.channel_size2, self.channel_size3)
self.enc4 = self.conv_block(self.channel_size3, self.channel_size4)
                        bottleneck = self.bottleneck(F.max pool2d(enc4, 2))
                        dec4 = self.upconv4(bottleneck)
                         dec4 = torch.cat((dec4, enc4), dim=1)
                                                                                                                                                                                                      self.bottleneck = self.conv_block(self.channel_size4, self.channel_size5)
                        dec3 = self.upconv3(dec4)
                                                                                                                                                                                                      self.upconv4 = nn.ConvTranspose2d(self.channel_size5, self.channel_size4, kernel_size=2, stride=2, bias=False)
                         dec3 = self. dec3(dec3)
                                                                                                                                                                                                      self.dec3 = self.conv_block(self.channel_size4, self.channel_size3)
self.upconv2 = nn.ConvTranspose2d(self.channel_size3, self.channel_size2, kernel_size=2, stride=2, bias=False)
self.dec2 = self.conv_block(self.channel_size3, self.channel_size2)
                         dec2 = self. dec2(dec2)
                         dec1 = self.dec1(dec1)
                                                                                                                                                                                                       self.outconv = nn.Conv2d(self.channel_sizel, out_channels, kernel_size=1)
```

#### 5. Program Translation: from Python to C

• Conv2d() in PyTorch

torch nn Conv2d(in channels

torch.nn.Conv2d(in\_channels, out\_channels, kernel\_size, stride=1, paddin
g=0, dilation=1, groups=1, bias=True, padding\_mode='zeros', device=None,
dtype=None)

R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢

• ConvTranspose2d() in PyTorch

torch.nn.ConvTranspose2d(in\_channels, out\_channels, kernel\_size, stride=
1, padding=0, output\_padding=0, groups=1, bias=True, dilation=1, padding
\_mode='zeros', device=None, dtype=None)

 ReLU() in PyTorch torch.nn.ReLU(inplace=False)

BatchNorm2d in PyTorch

torch.nn.BatchNorm2d(num\_features, eps=1e-

05, momentum=0.1, affine=True, track\_running\_stats=True, device=None, dt ype=None)

```
void batch_norm(float* input, float* output, float* gamma, float* beta, int channels, int height, int width, float epsilon) {
   int num_elements = height * width;

   for (int c = 0; c < channels; c++) {
        float mean = 0.0;

        float var = 0.0;

        // Calculate mean
        for (int i = 0; i < num_elements; i++) {
            mean += input[c * num_elements + i];
        }
        mean /= num_elements;

        // Calculate variance
        for (int i = 0; i < num_elements; i++) {
            var += (input[c * num_elements + i] - mean) * (input[c * num_elements + i] - mean);
        }
        var /= num_elements;

        // Normalize, scale, and shift
        for (int i = 0; i < num_elements; i++) {
            int idx = c * num_elements + i;
                output[idx] = gamma[c] * ((input[idx] - mean) / (float)sqrt((float) (var + epsilon))) + beta[c];
        }
    }
}

void relu(float* input, float* output, int channels, int height, int width) {
    int num_elements = channels * height * width;

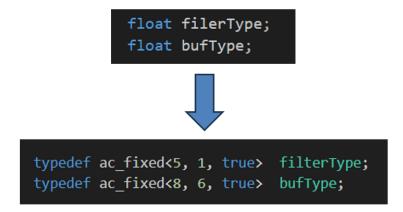
for (int i = 0; i < num_elements; i++) {
        output[i] = (float) fmax((float)0, (float) input[i]);
    }
}</pre>
```

R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢

### 6. Hardware Implementation by HLS

### Parameter and Data Quantization

- Original python model use float32 for compute
- Need to quantize as fixed point to reduce hardware complexity and storage
- Parameter: 5 bits, Pixel: 8 bits
- Can reduce 32 bits to 5, 8 bits (fixed point, W5A8)

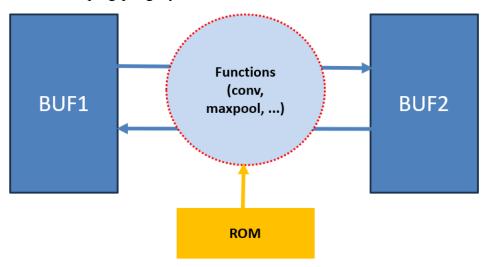


### **HLS Implementation**

• Use RAM as data buffer, ROM as parameter storage



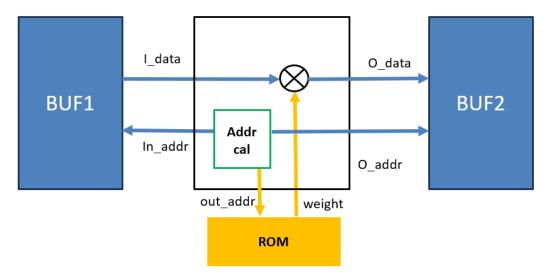
• 2 set of data buffer, ping-pong style



### **Implementation of Conv2d**

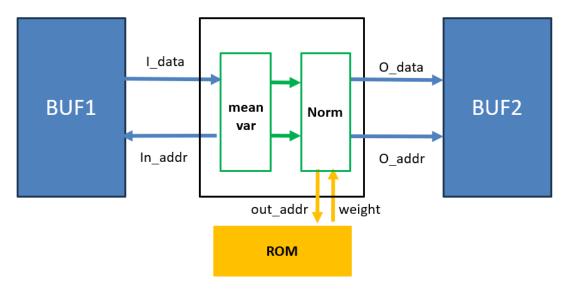
- Minimum hardware resource
- Compute address every cycle, communicate with storage

R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢



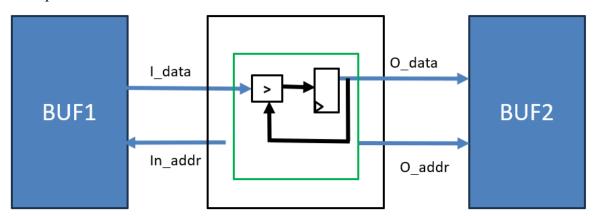
### **Implementation of Batchnorm**

- Compute mean, variance for each channel
- Multiply by gamma and beta



### **Implementation of Maxpool**

• Compare with the candidate max value in the filter



R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢

### **HLS Implementation Results**

Hardware resources by modules

| conv2d | batchnorm | maxpool | transconv | ROM    | top    |
|--------|-----------|---------|-----------|--------|--------|
| 6647   | 9894      | 3650    | 7194      | 393216 | 423251 |

- ROM occupies a large portion
- Reported RAM areas are 0

```
Bill of Materials (Datapath)

Component Name

Area Score Area(DSP) Area(LUTs) Area(MUX_CARRYS) Delay Post Alloc Post Assign

[Lib: Xilinx_RAMS]

BLOCK_1R1W_RBW(10,12,12,4096,1,4096,12,1)

BLOCK_1R1W_RBW(4,15,12,32768,1,32768,12,1)

BLOCK_1R1W_RBW(4,15,12,32768,1,32768,12,1)

BLOCK_1R1W_RBW(4,15,12,32768,1,32768,12,1)

BLOCK_1R1W_RBW(4,15,12,32768,1,32768,12,1)
```

### 7. FSIC Integration

AXI-Lite configuration register address map

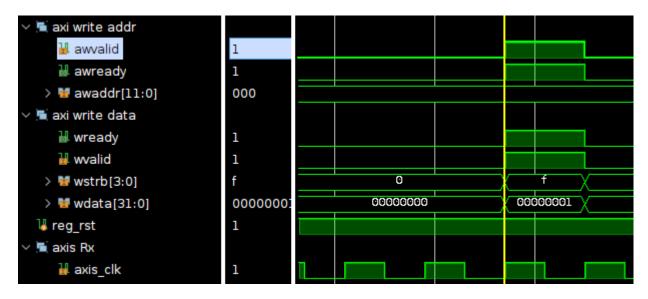
| Base Address | Offset | Description                  |
|--------------|--------|------------------------------|
| 0x3000_0000  | 0x000  | {ap_start, ap_done, ap_idle} |
|              | 0x010  | {height, width}              |
|              | 0x020  | {kernel_size, padding}       |
|              | 0x030  | {in_channels, out_channels}  |

- AXI-Stream
  - Input image to data RAM
  - Output results from data RAM

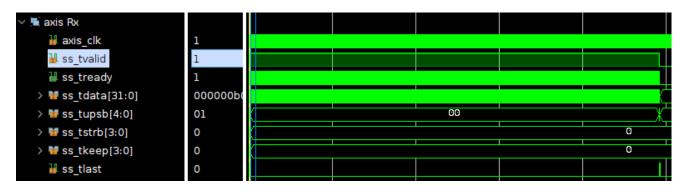
Integrate a UNET IP into the user project 0:

#### **Program ap start:**

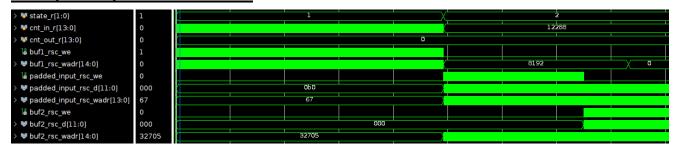
R12943003 劉冠亨 R12943016 謝言鼎 R11943006 張力元 R11525063 李佳螢



### **Input stream:**



### Pad input and perform calculation:



### 8. GitHub Link

 $https://github.com/alexding 1226/ASOC\_final$