Lab 05

Semantic Segmentation and low-complexity model

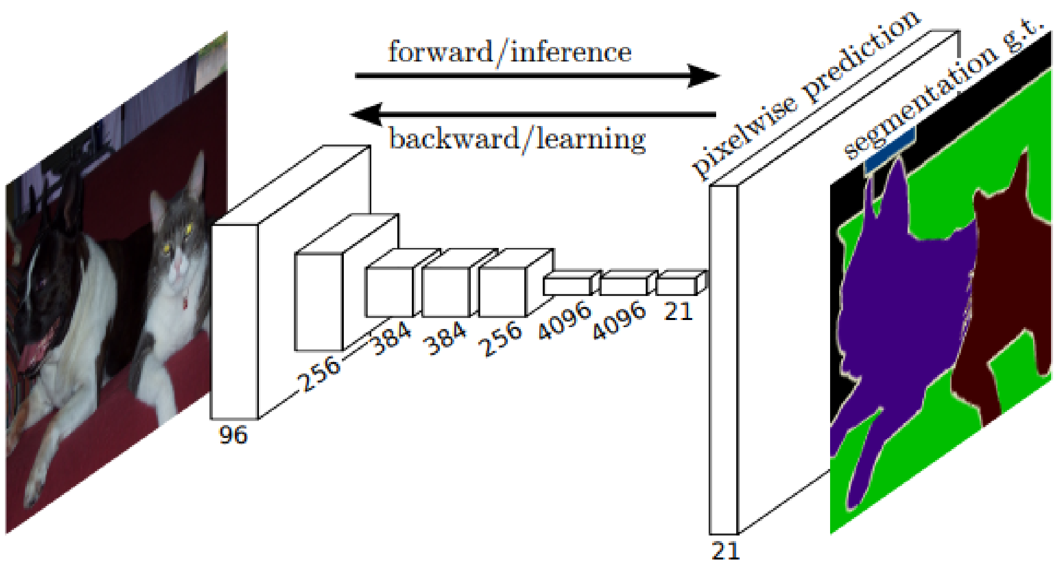
Outline

- Task 1
  - Semantic segmentation
- Task 2
  - Low complexity model
- Assignment rules

Fully Convolution Network

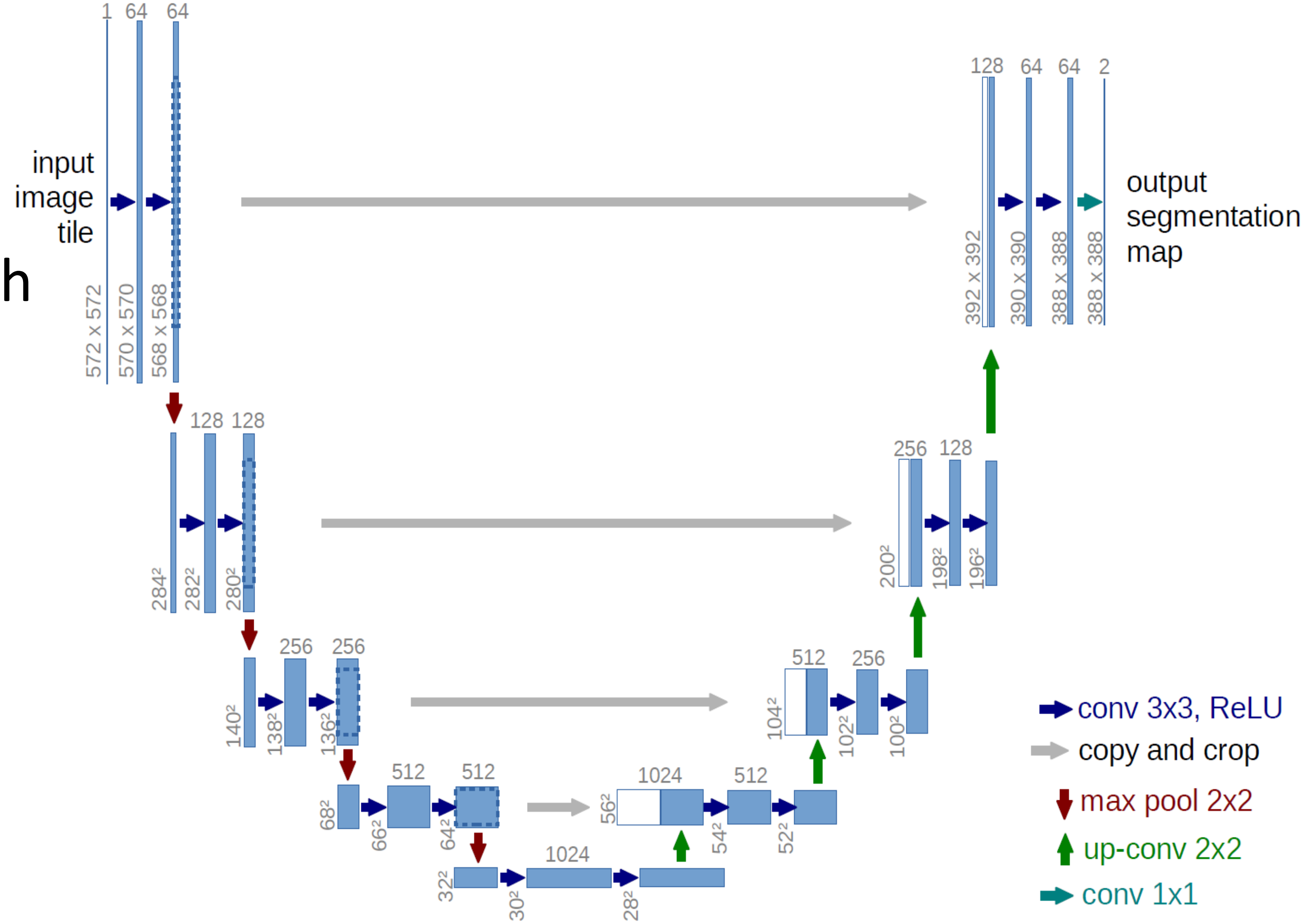
- How to inference lots of pixel?
  - Pixels to pixels training

- Encoder and decoder
  - Downsampling and upsampling



U-Net

- Based on fully convolution network
- Symmetric contracting/expanding path
- Skip connections

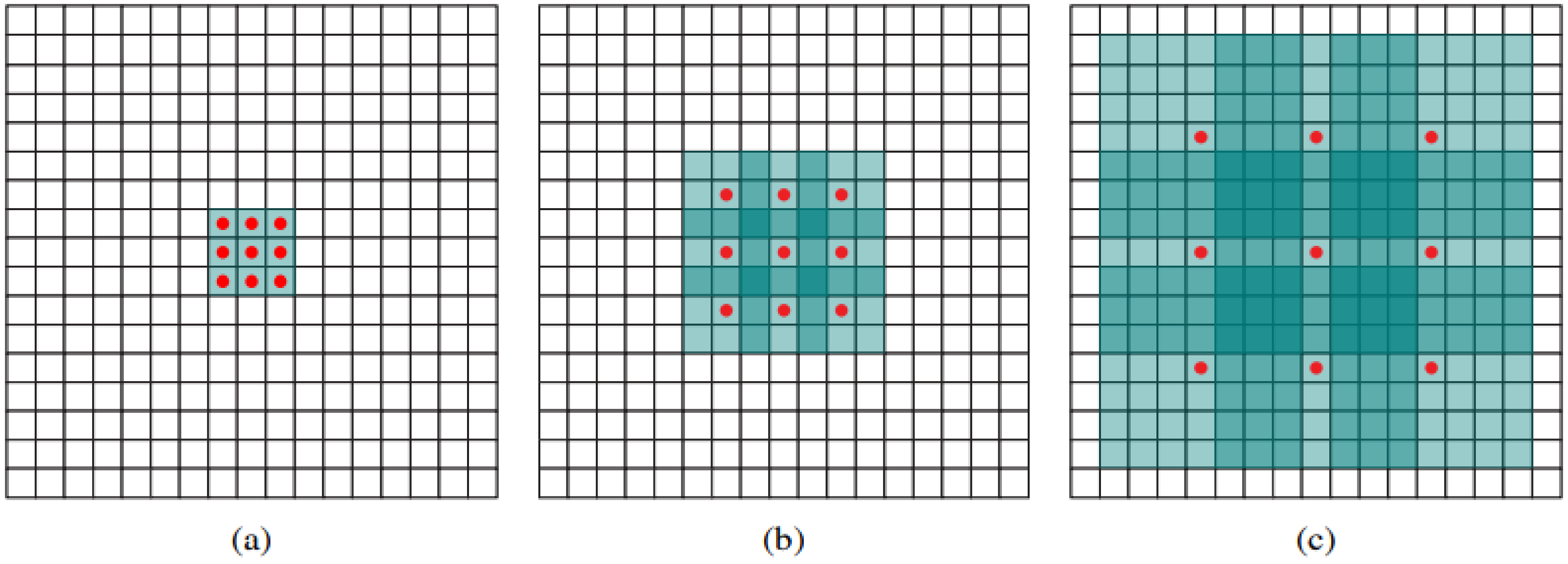


How to Upsample?

- Use traditional upsampling method
  - Bilinear, nearest, ….
- Use transpose convolution layer(deconvolution)
  - Learn how to upsample

Problems of Vanilla CNN

- Vanilla CNN uses pooling layer to reduce computation and increase the receptive field
- Is there any way to increase the receptive field without information loss?
  - Dilated convolution



Other semantic segmentation model

- FCN
- UNet
- SegNet
- Deeplab v1 ~ v3
- PSPNet
- SegFormer

Other Tips

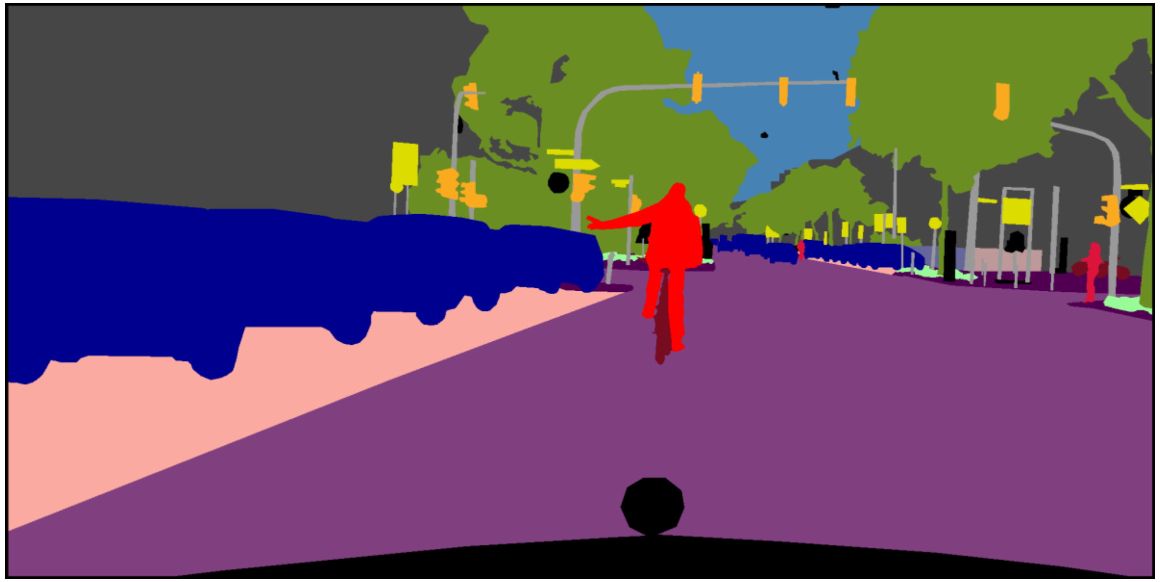
- Data augmentation
  - RandomHorizontalFlip
  - RandomCrop , ….
- Multi-scale
- Loss function
- Skip connection

Dataset: Cityscapes

- An RGB image with an RGB semantic segmentation image
  - Original size is 1024x2048
  - I will give you original size!
  - In default training settings, I will resize the image to 256x512 for training time issue

image

label



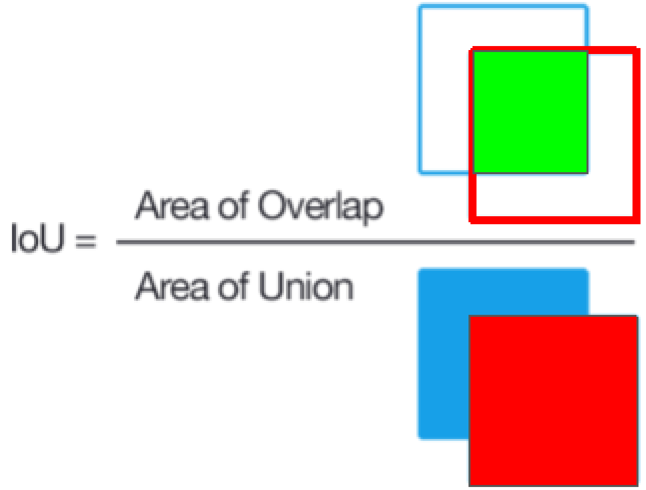
Classes of Label

- There are 34 classes and 8 kinds of categories

```
labels = [
    #       name                     id    trainId   category            catId     hasInstances   ignoreInEval   color          outColor
    Label(  'unlabeled'            ,  0 ,        0 , 'void'            , 0       , False        , True         , (  0,  0,  0) , (  0,  0,  0) ),
    Label(  'ego vehicle'          ,  1 ,        0 , 'void'            , 0       , False        , True         , (  0,  0,  0) , (  0,  0,  0) ),
    Label(  'rectification border' ,  2 ,        0 , 'void'            , 0       , False        , True         , (  0,  0,  0) , (  0,  0,  0) ),
    Label(  'out of roi'           ,  3 ,        0 , 'void'            , 0       , False        , True         , (  0,  0,  0) , (  0,  0,  0) ),
    Label(  'static'               ,  4 ,        0 , 'void'            , 0       , False        , True         , (  0,  0,  0) , (  0,  0,  0) ),
    Label(  'dynamic'              ,  5 ,        0 , 'void'            , 0       , False        , True         , (111, 74,  0) , (  0,  0,  0) ),
    Label(  'ground'               ,  6 ,        0 , 'void'            , 0       , False        , True         , ( 81,  0, 81) , (  0,  0,  0) ),
    Label(  'road'                 ,  7 ,        1 , 'flat'            , 1       , False        , False        , (128, 64,128) , (244, 35,232) ),
    Label(  'sidewalk'             ,  8 ,        1 , 'flat'            , 1       , False        , False        , (244, 35,232) , (244, 35,232) ),
    Label(  'parking'              ,  9 ,        1 , 'flat'            , 1       , False        , True         , (250,170,160) , (244, 35,232) ),
    Label(  'rail track'           , 10 ,        1 , 'flat'            , 1       , False        , True         , (230,150,140) , (244, 35,232) ),
    Label(  'building'             , 11 ,        2 , 'construction'    , 2       , False        , False        , ( 70, 70, 70) , ( 70, 70, 70) ),
    Label(  'wall'                 , 12 ,        2 , 'construction'    , 2       , False        , False        , (102,102,156) , ( 70, 70, 70) ),
    Label(  'fence'                , 13 ,        2 , 'construction'    , 2       , False        , False        , (190,153,153) , ( 70, 70, 70) ),
    Label(  'guard rail'           , 14 ,        2 , 'construction'    , 2       , False        , True         , (180,165,180) , ( 70, 70, 70) ),
    Label(  'bridge'               , 15 ,        2 , 'construction'    , 2       , False        , True         , (150,100,100) , ( 70, 70, 70) ),
    Label(  'tunnel'               , 16 ,        2 , 'construction'    , 2       , False        , True         , (150,120, 90) , ( 70, 70, 70) ),
    Label(  'pole'                 , 17 ,        3 , 'object'          , 3       , False        , False        , (153,153,153) , (153,153,153) ),
    Label(  'traffic light'        , 19 ,        3 , 'object'          , 3       , False        , False        , (250,170, 30) , (153,153,153) ),
    Label(  'traffic sign'         , 20 ,        3 , 'object'          , 3       , False        , False        , (220,220,  0) , (153,153,153) ),
    Label(  'vegetation'           , 21 ,        4 , 'nature'          , 4       , False        , False        , (107,142, 35) , (107,142, 35) ),
    Label(  'terrain'              , 22 ,        4 , 'nature'          , 4       , False        , False        , (152,251,152) , (107,142, 35) ),
    Label(  'sky'                  , 23 ,        5 , 'sky'             , 5       , False        , False        , ( 70,130,180) , ( 70,130,180) ),
    Label(  'person'               , 24 ,        6 , 'human'           , 6       , True         , False        , (220, 20, 60) , (220, 20, 60) ),
    Label(  'rider'                , 25 ,        6 , 'human'           , 6       , True         , False        , (255,  0,  0) , (220, 20, 60) ),
    Label(  'car'                  , 26 ,        7 , 'vehicle'         , 7       , True         , False        , (  0,  0,142) , (  0,  0,142) ),
    Label(  'truck'                , 27 ,        7 , 'vehicle'         , 7       , True         , False        , (  0,  0, 70) , (  0,  0,142) ),
    Label(  'bus'                  , 28 ,        7 , 'vehicle'         , 7       , True         , False        , (  0, 60,100) , (  0,  0,142) ),
    Label(  'caravan'              , 29 ,        7 , 'vehicle'         , 7       , True         , True         , (  0,  0, 90) , (  0,  0,142) ),
    Label(  'trailer'              , 30 ,        7 , 'vehicle'         , 7       , True         , True         , (  0,  0,110) , (  0,  0,142) ),
    Label(  'train'                , 31 ,        7 , 'vehicle'         , 7       , True         , False        , (  0, 80,100) , (  0,  0,142) ),
    Label(  'motorcycle'           , 32 ,        7 , 'vehicle'         , 7       , True         , False        , (  0,  0,230) , (  0,  0,142) ),
    Label(  'bicycle'              , 33 ,        7 , 'vehicle'         , 7       , True         , False        , (119, 11, 32) , (  0,  0,142) ),
]
```

Model Evaluation

- Evaluation metric: Mean Intersection over Union(mIoU)
  - For each class, IoU is determined as :
  - $IoU = \frac{True\ Positive}{True\ Positive+False\ Positive+Flase\ Negative}$
  - mIoU is calculated by averaging the over all classes
  - mIoU calculation has done by TA, DO NOT change it



Task-1: Semantic segmentation

- You should build your model in ./network/network.py
  - You can choose the model whatever you want

```
import torch
import torch.nn as nn
import torch.nn.functional as F

### Write your model architecture



### End

def load_model(MODEL_PATH):
    #call model
    model =
    state_dict = torch.load(MODEL_PATH)
    model.load_state_dict(state_dict)
    return model
```

Task-1 Training

- The image will be resize to 256x512 by default
- You should implement the training settings by yourself
- IoU computation has been completed by TA

Main

```python
# define yout model
Net = 
Net = Net.to(device)
# define your optimizer

# define your loss
```

Training

```python
for i, (data, target) in enumerate(data_loader):
    data , target = data.to(device),target.to(device)
    ### yourself
    # zero grad

    # input data to network

    # calculate loss

    # backward

    # step LR

    ### End
    training_loss = loss.item()
    pbar.set_postfix(**{'loss (batch)': training_loss})
    pred = pred.data.max(1)[1]
    Meter['metric'].update(target.data.cpu().numpy(), pred.data.cpu().numpy())
    Meter['loss'].update(training_loss,data.size()[0])
    pbar.update(data.shape[0])
```

Validation

```python
for i, (data, target) in enumerate(data_loader):
    data, target = data.to(device), target.to(device)
    timeStart = time.time()
    ### Yourself
    # input data to network


    timeEnd = time.time()
    # calculate loss

    ## End
    pred = pred.data.max(1)[1]
    Meter['metric'].update(target.data.cpu().numpy(), pred.data.cpu().numpy())
    Meter['loss'].update(validation_loss,data.size()[0])
    Meter['time'].update(timeEnd-timeStart,1)
```

Testing flow for task-1

- You need to prepare
  - ./network/network.py：your model architecture
  - model_task1.pth：saved model file
- Command
  - python test_task1.py <MODEL_PATH> <DATA_PATH>
  - If you don't change the model and dataset storage location, you can simply run python test_task1.py
- Example output

```
Final IoU = 0.779713
```

- For task-1, mIoU should >= 0.72
- Please make sure you can successfully run the test_task1.py and have the results.
- **Do not modify the test_task1.py!**

Reminder

- DO NOT use the validation set for training
  – TA has reserved some of the dataset for testing

- Only use the data we provide to train your model
  – Without other data (augmentation is allowed)
  – Without pretrained model
  – Without calling any full network package

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
```

Outline

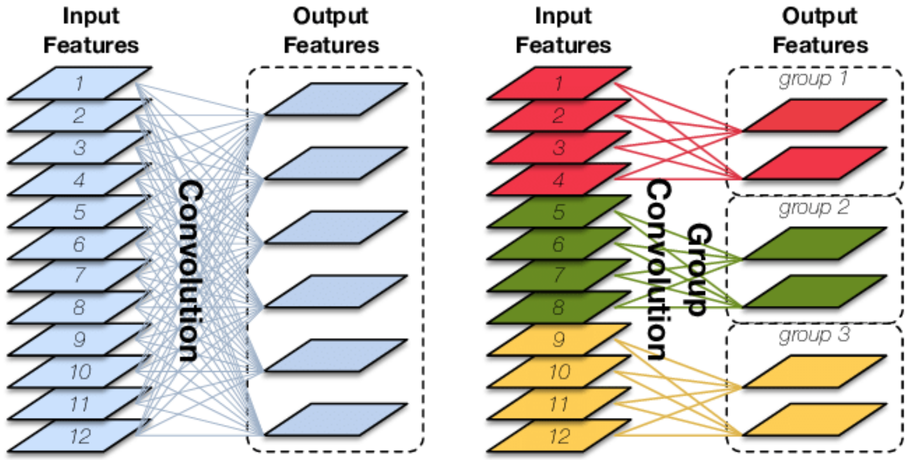
- Task 1
  - Semantic segmentation
- Task 2
  - Low complexity model
- Assignment rules

Why do we need low-complexity model

- Semantic segmentation is a computationally intensive task
  - We need to find ways to reduce its complexity.
- How?
  - Reduce execution time
    - Include computation time and memory access time
  - Reduce parameters
    - Reduce memory access
  - Reduce FLOPs
    - Reduce computation time
- **Objective in this Task: fewer parameters, lower FLOPs, maintain accuracy!**

Decomposition

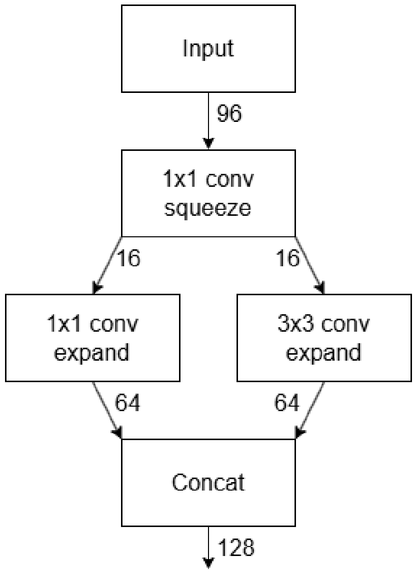
- Convolution has high computational and memory demands.
  - Decomposition can reduce both FLOPs and parameters



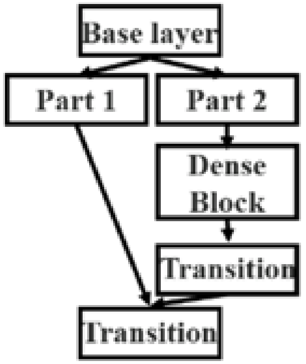
- Group convolution
  - Example: ResNeXt
- Separable convolution
  - Depth-wise convolution + point-wise convolution
  - Example: MobileNet, Xception, EfficientNet

Channel reduction and partition

- Reducing the channel might also be a good method to reduce parameters and FLOPs



- Channel reduce
  - Reduce the channel by performing 1x1 convolution
  - Example: SqueezeNet
- Channel partition
  - Calculate only on part of the channel, and then concatenate the result
  - Example: CSPNet



(b) CSPDenseNet

Other methods

- The method used in Lab04 can also be effectively applied in this lab, alongside the approaches mentioned above.
  - Pruning
  - ~~Quantization~~

Accuracy Restore

- The former methods might cause a drop in accuracy. How could we restore it?
  - create or reuse gradient diversity as much as possible

- Attention
  - Example: Squeeze-Excitation block
- Multiple gradient path
  - Example: Dense layer in DenseNet
- Skip connection
  - Example: U-Net
- Training
  - Example: Multi-scale, Data augmentation, loss function

Task-2: Low complexity

- Using your model architecture in task-1 as backbone
- Build and adjust it in **./network2/network.py**
- **Any method** can be used as long as it effectively reduces model complexity

```
import torch
import torch.nn as nn
import torch.nn.functional as F

### Write your model architecture



### End

def load_model(MODEL_PATH):
    #call model
    model =
    state_dict = torch.load(MODEL_PATH)
    model.load_state_dict(state_dict)
    return model
```

Task-2 training

- Totally same as task-1 (if you doesn’t apply pruning or quantization)
- The image will be resize to 256x512 by default
- You should implement the training settings by yourself
- IoU computation has been completed by TA

Main

```python
# define yout model
Net = 
Net = Net.to(device)
# define your optimizer

# define your loss
```

Training

```python
for i, (data, target) in enumerate(data_loader):
    data , target = data.to(device),target.to(device)
    ### yourself
    # zero grad

    # input data to network

    # calculate loss

    # backward

    # step LR

    ### End
    training_loss = loss.item()
    pbar.set_postfix(**{'loss (batch)': training_loss})
    pred = pred.data.max(1)[1]
    Meter['metric'].update(target.data.cpu().numpy(), pred.data.cpu().numpy())
    Meter['loss'].update(training_loss,data.size()[0])
    pbar.update(data.shape[0])
```

Validation

```python
for i, (data, target) in enumerate(data_loader):
    data, target = data.to(device), target.to(device)
    timeStart = time.time()
    ### Yourself
    # input data to network


    timeEnd = time.time()
    # calculate loss

    ## End
    pred = pred.data.max(1)[1]
    Meter['metric'].update(target.data.cpu().numpy(), pred.data.cpu().numpy())
    Meter['loss'].update(validation_loss,data.size()[0])
    Meter['time'].update(timeEnd-timeStart,1)
```

Testing flow for task-2

- You need to prepare
  - ./network2/network.py：Your task-2 model architecture
  - model_task2.pth：saved model file
- Command
  - python test_task2.py <MODEL_PATH> <DATA_PATH>
  - If you don't change the model and dataset storage location, you can simply run python test_task2.py
- Example output

```
flops =1953431552
Model parameter size =1522.402 KB
Final IoU = 0.678319
```

- For task-2, mIoU should >= 0.66
- Please make sure you can successfully run the test_task2.py and have the results.
- **Do not modify the test_task2.py!**

Reminder

- DO NOT use the validation set for training
  - TA has reserved some of the dataset for testing
- Only use the data we provide to train your model
  - Without other data (augmentation is allowed)
  - Without pretrained model
  - Without calling any full network package

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
```

Outline

- Task 1
  - Semantic segmentation
- Task 2
  - Low complexity model
- Assignment rules

Download files

- If you are using the 414 server
  - Run the command: tar -xvf ../shared/DL_Lab5_414.tar
  - You don't need to download the dataset

- If you are not using the 414 server
  - Download DL_Lab5.tar from new E3
  - Download dataset from the link provided by TA

Files

- DL_Lab5
  - data: dataset(image)
    - If you are using the 414 server, you don’t need this folder
  - dataset
    - dataset.py: load data
      - Do not change this file
  - network
    - network.py: model for task1
  - network2
    - network.py: model for task2
  - utils
    - labels_cat.py: label definition
    - metric.py: acc/IoU calculation
      - Do not change files in this folder
  - Lab05_segmentation.ipynb
    - Task-1 notebook
  - Lab05_segmentation_task2.ipynb
    - Task-2 notebook
  - test_task1.py
    - Test script for task-1
  - test_task2.py
    - Test script for task-2

```
drwxr-xr-x  6 dl2025f_ta_5 dl2025f_tas 4096 Nov  9 14:53 ./
drwx------ 10 dl2025f_ta_5 dl2025f_tas 4096 Nov  9 14:53 ../
drwxr-xr-x  2 dl2025f_ta_5 dl2025f_tas 4096 Nov  9 14:39 dataset/
-rw-r--r--  1 dl2025f_ta_5 dl2025f_tas 7530 Nov  9 14:37 Lab05_segmentation_task1.ipynb
-rw-r--r--  1 dl2025f_ta_5 dl2025f_tas 7571 Nov  9 14:38 Lab05_segmentation_task2.ipynb
drwxr-xr-x  2 dl2025f_ta_5 dl2025f_tas 4096 Nov  9 08:08 network/
drwxr-xr-x  2 dl2025f_ta_5 dl2025f_tas 4096 Nov  9 08:08 network2/
-rw-r--r--  1 dl2025f_ta_5 dl2025f_tas 1819 Nov  9 14:35 test_task1.py
-rw-r--r--  1 dl2025f_ta_5 dl2025f_tas 2641 Nov  9 14:35 test_task2.py
drwxr-xr-x  2 dl2025f_ta_5 dl2025f_tas 4096 Nov  8 08:50 utils/
```

Grading Policy

- Task-1 **(25 %)**
  - mIoU should >= 0.72
    - TA will run test_task1.py by using your ./network/network.py & model_task1.pth
- Task-2 **(55 %)**
  - mIoU should >= 0.66 (25%)
    - TA will run test_task2.py by using your ./network2/network.py & model_task2.pth
  - Performance (30%)
    - $FoM = (FLOPs)^2 \times (model\ size)^2 \times \frac{1}{(val.\, mIoU - 0.65)}$
    - The smaller, the better
- Report **(20 %)**

Reminder

- Submit Deadline : 2 weeks (2025/11/24 23:59)
- Please strictly follow the naming rules !!!
  - Any naming error will result in 5 points deduction
- Upload 7 files to new e3
  - Lab05_task1_studentID.ipynb
  - Lab05_task2_studentID.ipynb
  - network_task1.py (just rename ./network/network.py to network_task1.py)
  - network_task2.py (just rename ./network2/network.py to network_task2.py)
  - model_task1.pth
  - model_task2.pth
  - report_studentID.pdf (example: report_313555555.pdf)

Report

- Task-1
  - Show the model architecture you use, and how do you improve your model
    - Pros and cons of your model
  - Show how do you improve accuracy
    - Loss, Optimizer, overcome overfitting

- Task-2
  - Which methods do you apply to lower complexity and bridge the accuracy gap
  - Why do you apply the methods
  - How do you implement

Have Fun !!!!!!