SEOUL NATIONAL UNIVERSITY

LIST

LABORATORY FOR IMAGING SCIENCE AND TECHNOLOGY

Deep learning QSM tutorial

Woojin Jung

Ph.D Candidate

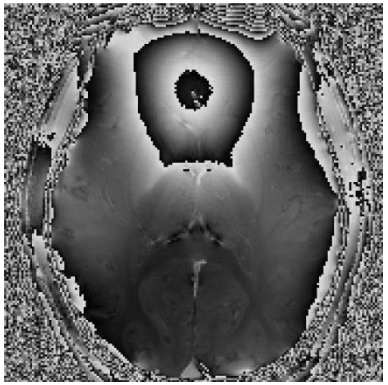
Laboratory for Imaging Science and Technology (LIST)

Seoul National University

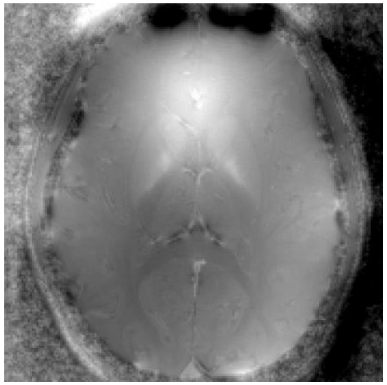
QSM reconstruction

Deep neural network

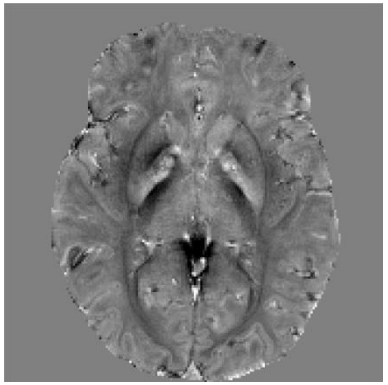
MRI phase



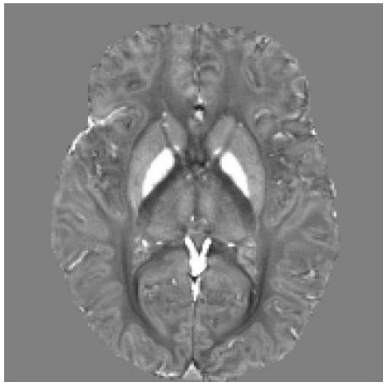
Unwrapped phase



Local field



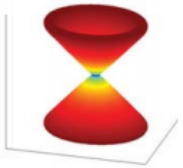
QSM



Phase unwrapping

Background field removal

Dipole deconvolution

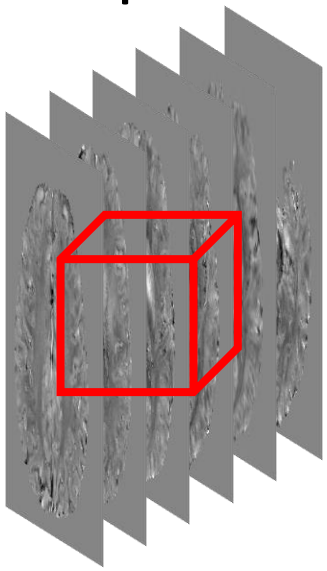


S. Wharton, MRM, 2010
K. Shmueli, MRM, 2009

ill-posed problem!

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Deep learning QSM

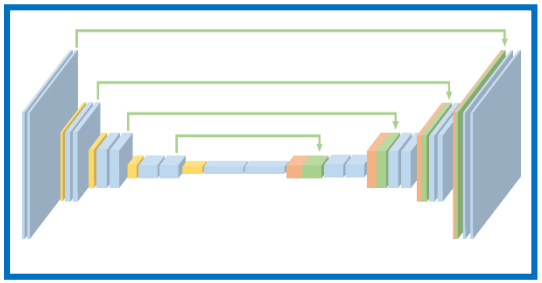


Patching field map
32 x 32 x 32

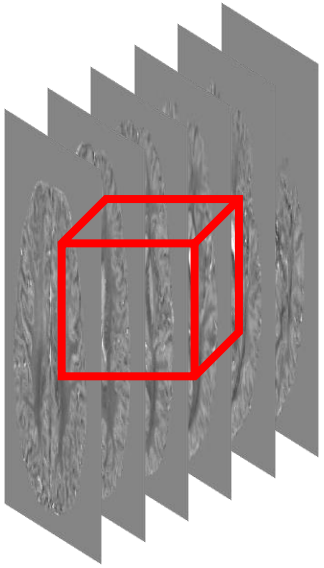
Training input

$$f = d * \chi$$

Toy deep neural network



Full deep neural network



Patching QSM map
32 x 32 x 32

Training label

Training

- Network model
- Loss function
- Optimizer

Evaluation

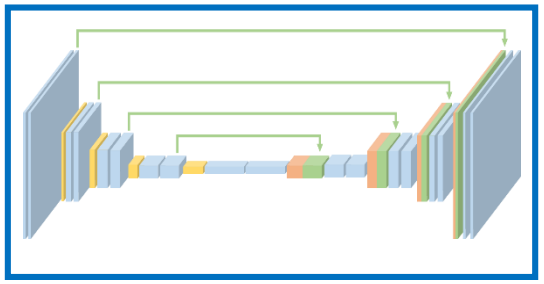
- Inference on testset

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Goal of Hands on

- 1) Build Toy deep neural network model
- 2) Programming training process
- 3) Load 'Full deep neural network' and inference on test set

Toy deep neural network

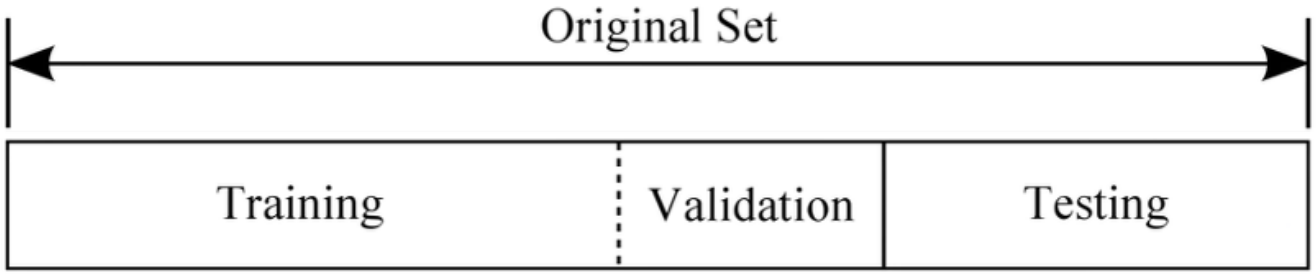


Full deep neural network

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

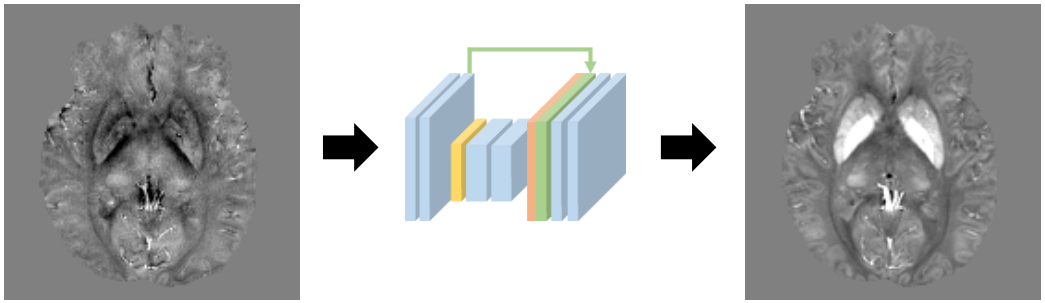
Dataset of Toy network

- Need to set training, validation, and testing data



- QSM challenge 2016 data utilized for this tutorial
  - Train input, Validation input, Test input – phs_tissue.mat
  - Train output, Validation ouput, Test output – chi_cosmos.mat

  http://www.neuroimaging.at/pages/qsm.php

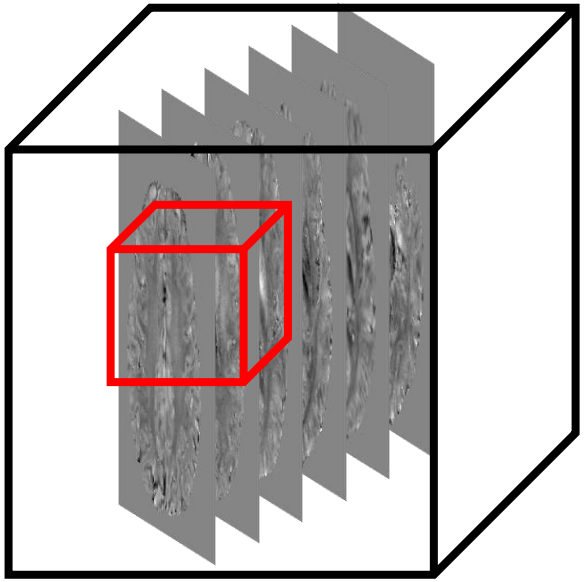


Input: phs_tissue.mat

Output: chi_cosmos.mat

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

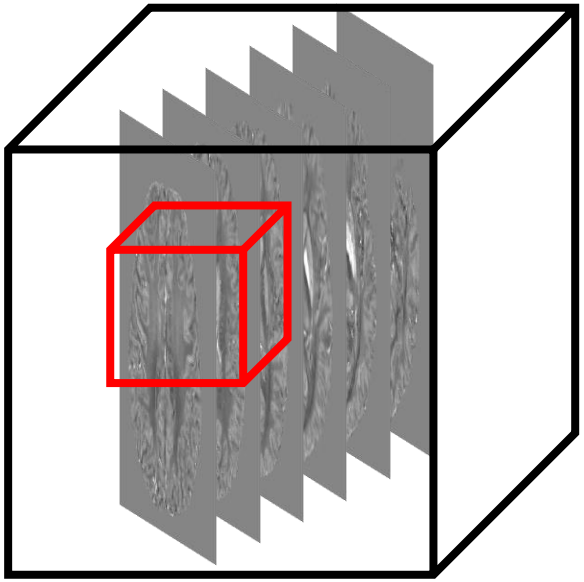
Dataset of Toy network



3D local field map
160 x 160 x 160

Patching field map
32 x 32 x 32

Training input



3D COSMOS QSM
160 x 160 x 160

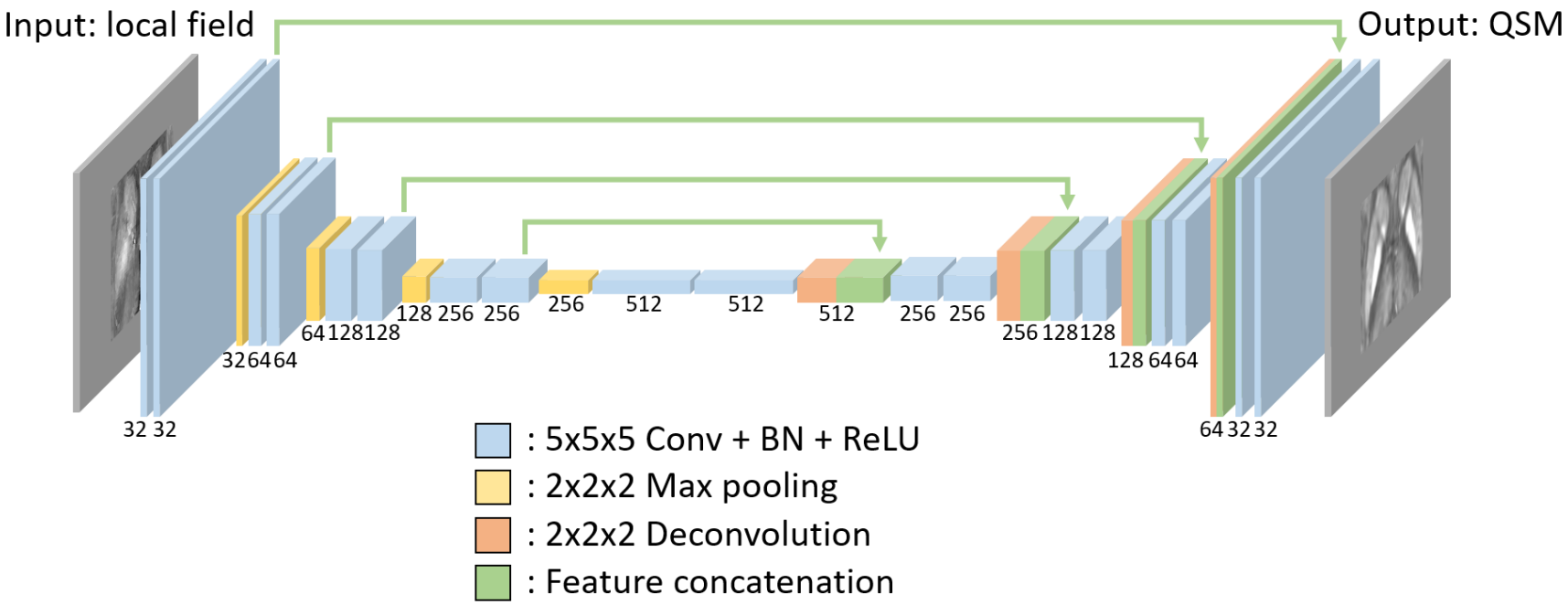
Patching QSM map
32 x 32 x 32

Training label

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Network model

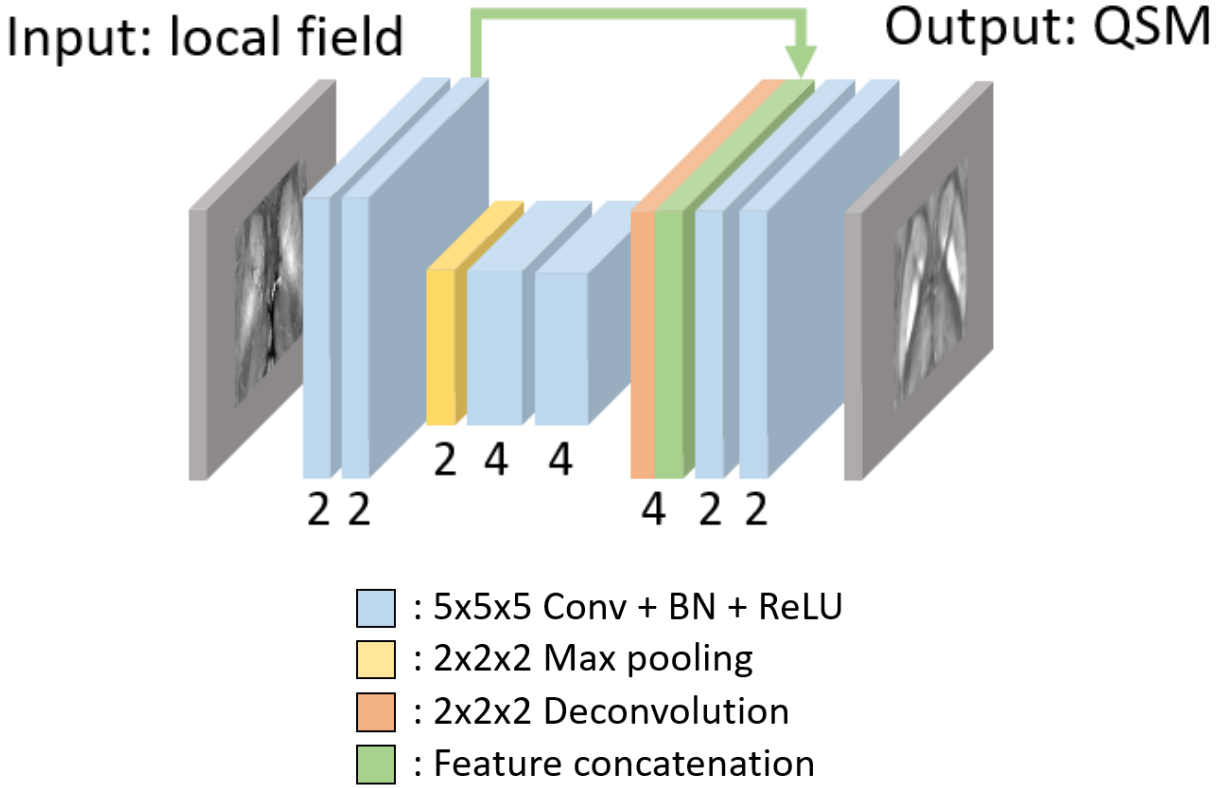
- Full 3D U-net



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Network model

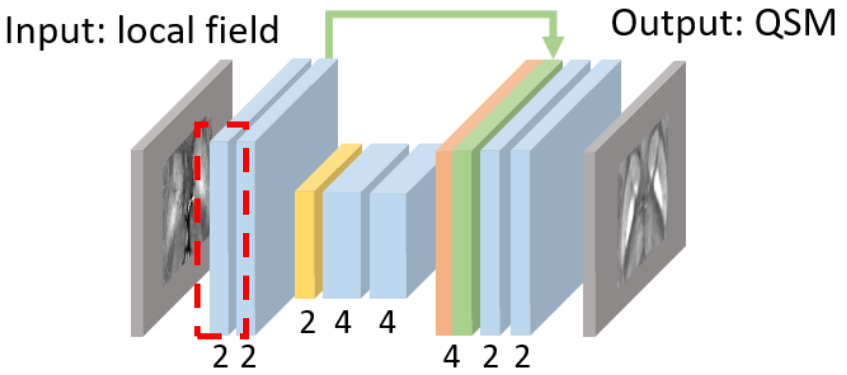
- Toy 3D U-net



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

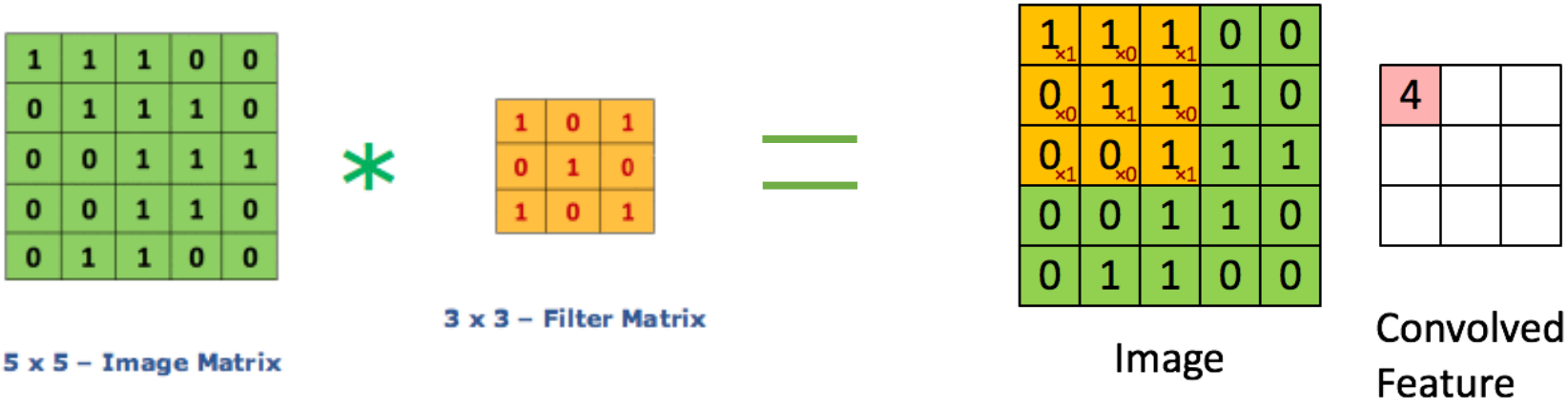
Network model

- Toy 3D U-net



- Convolution
  - Feature extraction

: 5x5x5 Conv + BN + ReLU

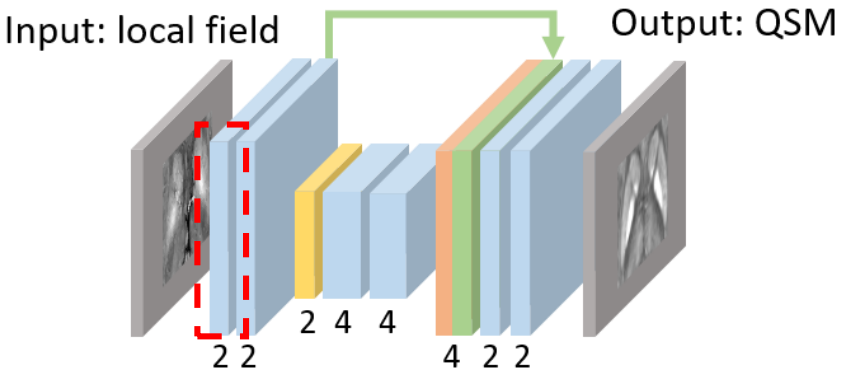


https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Network model

- Toy 3D U-net



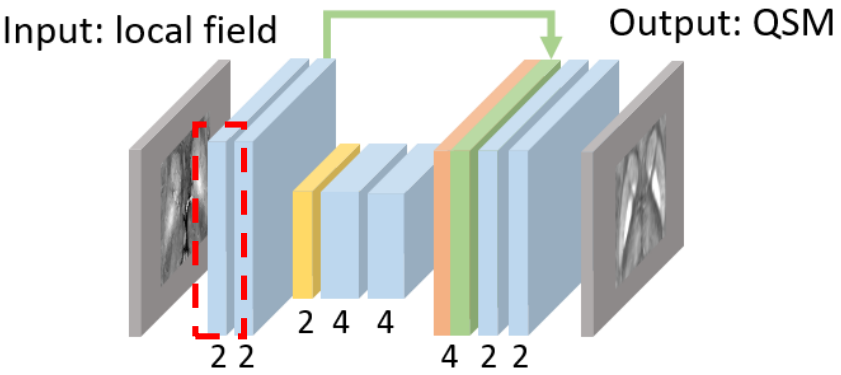
: 5x5x5 Conv + BN + ReLU

- Batch normalization
  - Better training performance

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

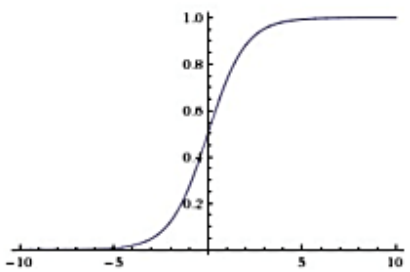
Network model

- Toy 3D U-net

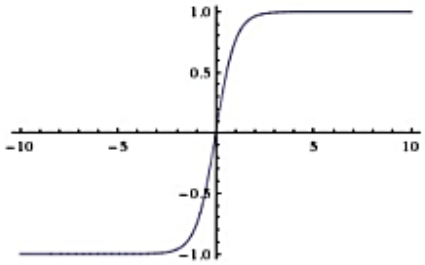


- Activation functions
  - Nonlinear property

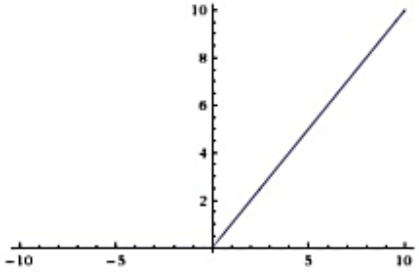
: 5x5x5 Conv + BN + ReLU



Sigmoid



tanh

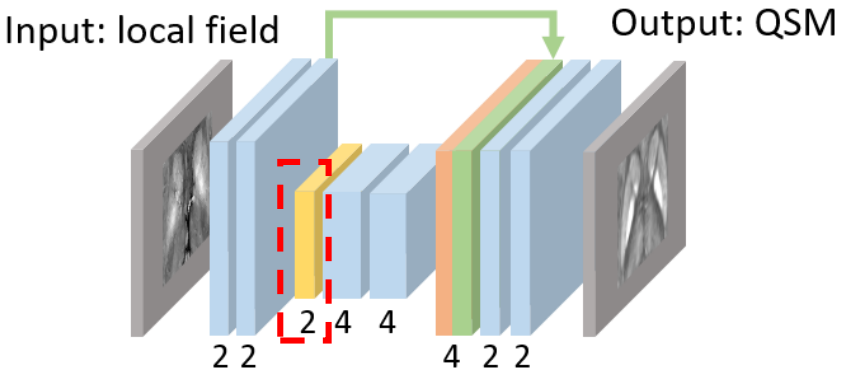


ReLU

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

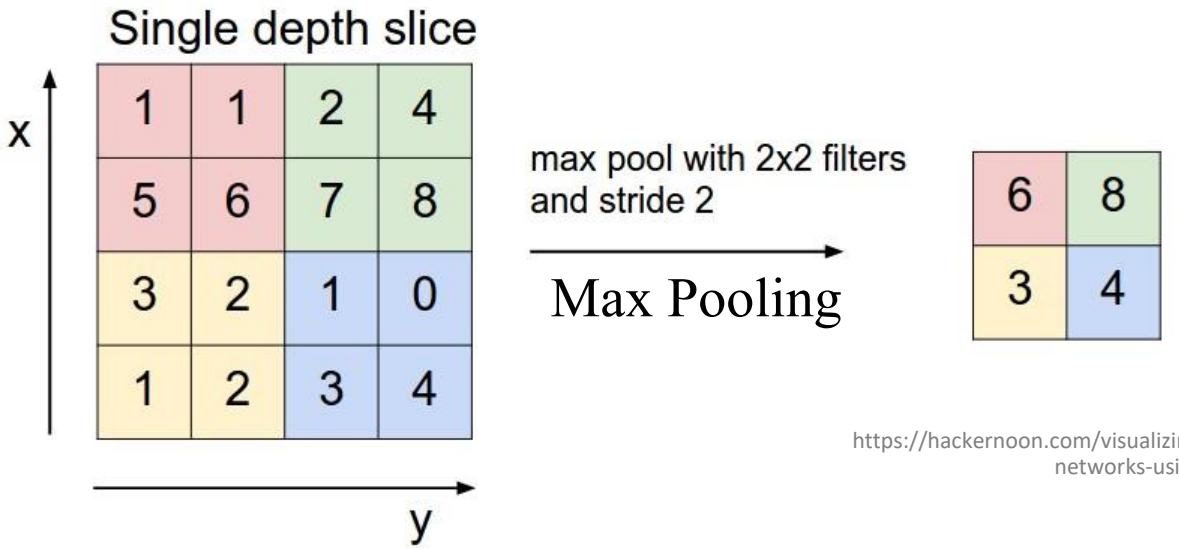
Network model

- Toy 3D U-net



- Max pooling
  - Decrease number of features

: 2x2x2 Max pooling

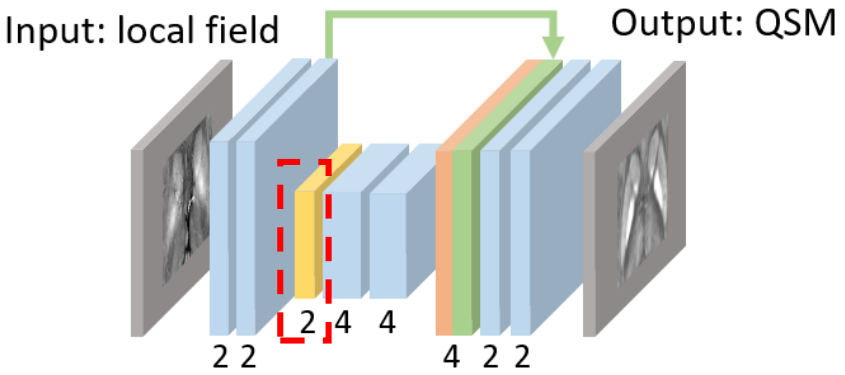


https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

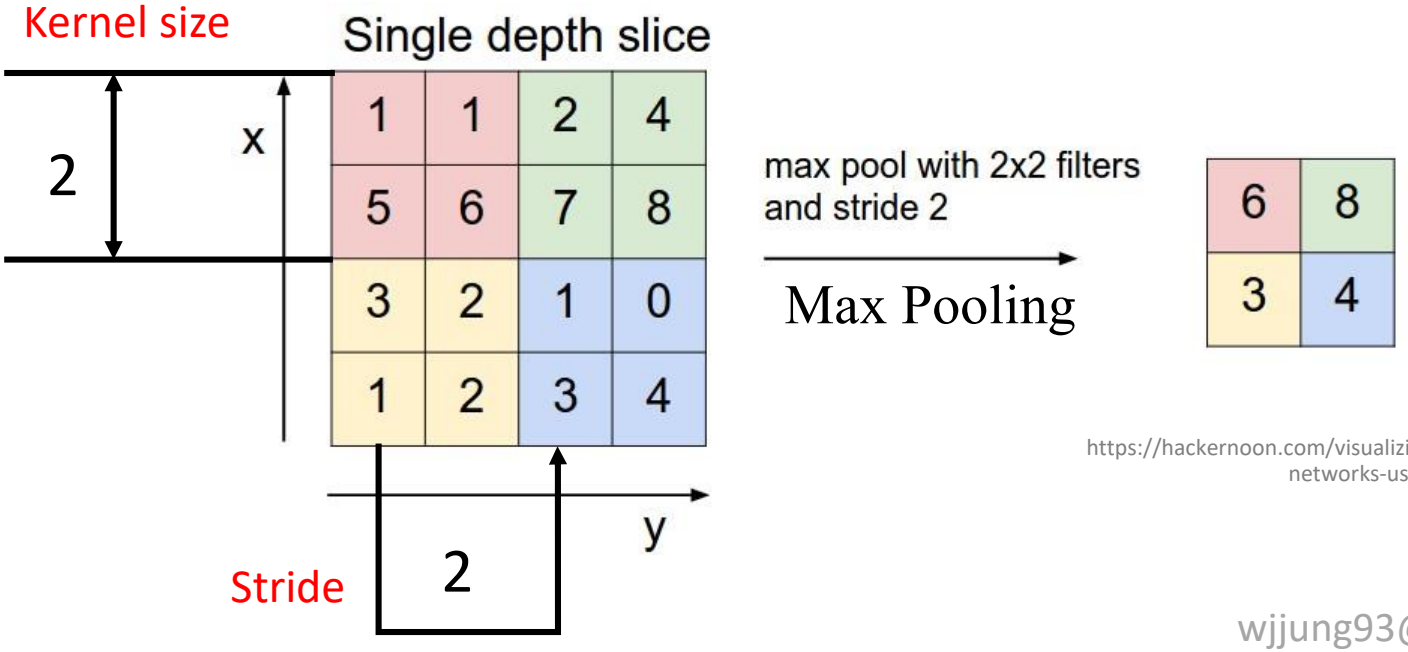
Network model

- Toy 3D U-net



- Max pooling
  - Decrease number of features

: 2x2x2 Max pooling

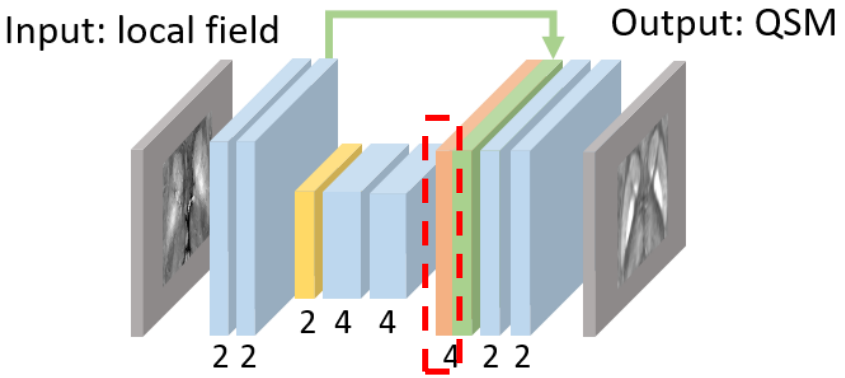


https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

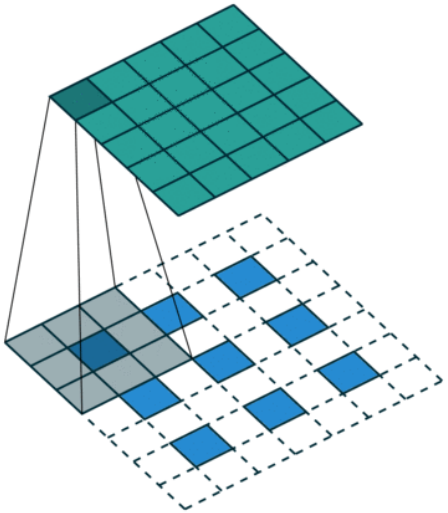
Network model

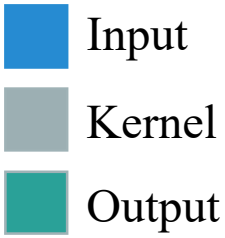
- Toy 3D U-net



- Deconvolution

: 2x2x2 Deconvolution



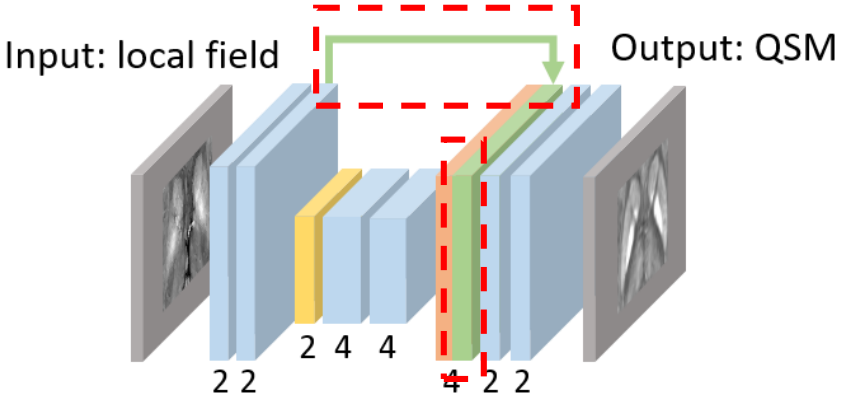


https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

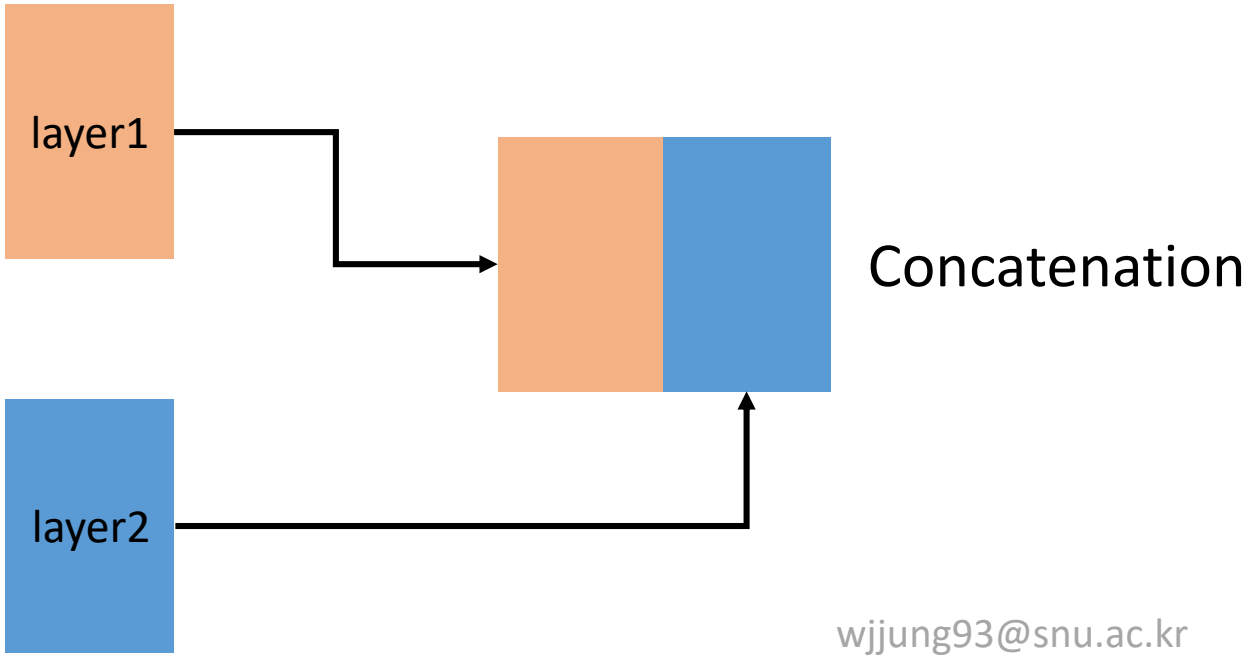
Network model

- Toy 3D U-net



- Feature concatenation
  - Feature propagation

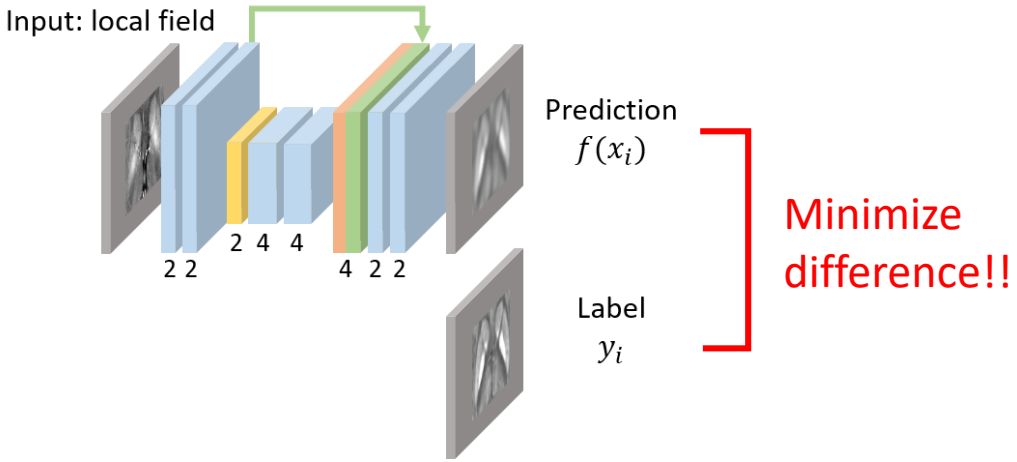
: Feature concatenation



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Network model

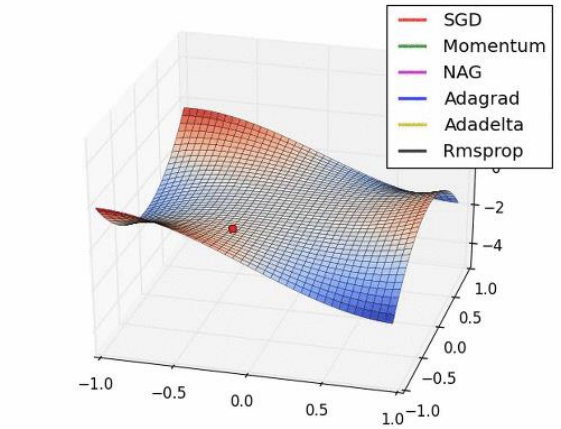
- Loss function - L1 loss



$$L = \sum_{i=1}^{n} |y_i - f(x_i)|$$

$x: input, y: label$

- Optimizer - Gradient Descent based
  - Adam optimizer

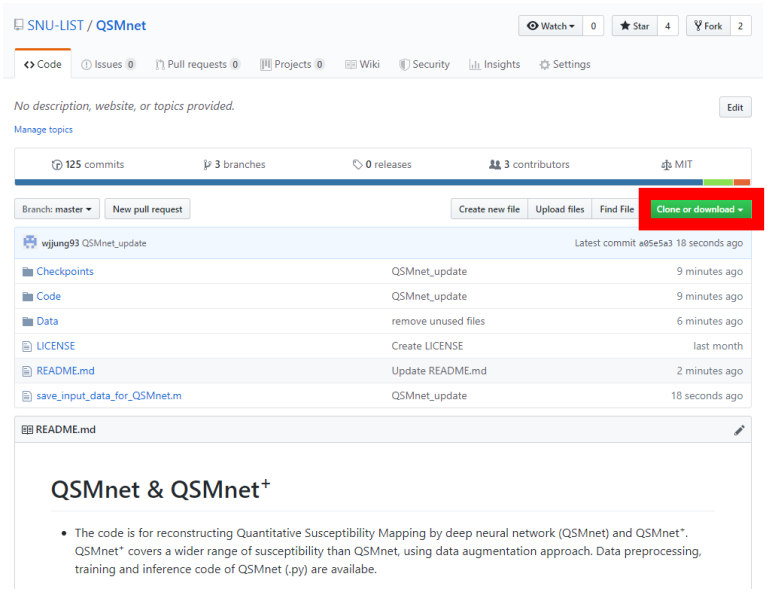


https://gomguard.tistory.com/187

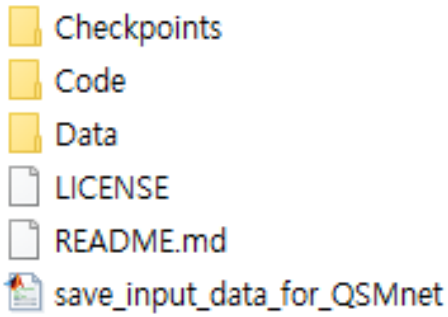
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Hands on

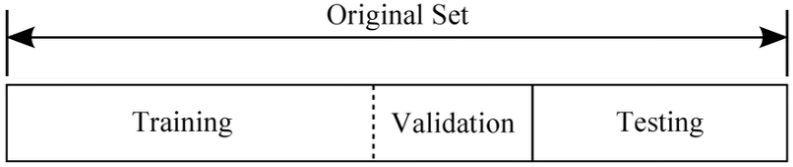
QSMnet directory



QSMnet-master/

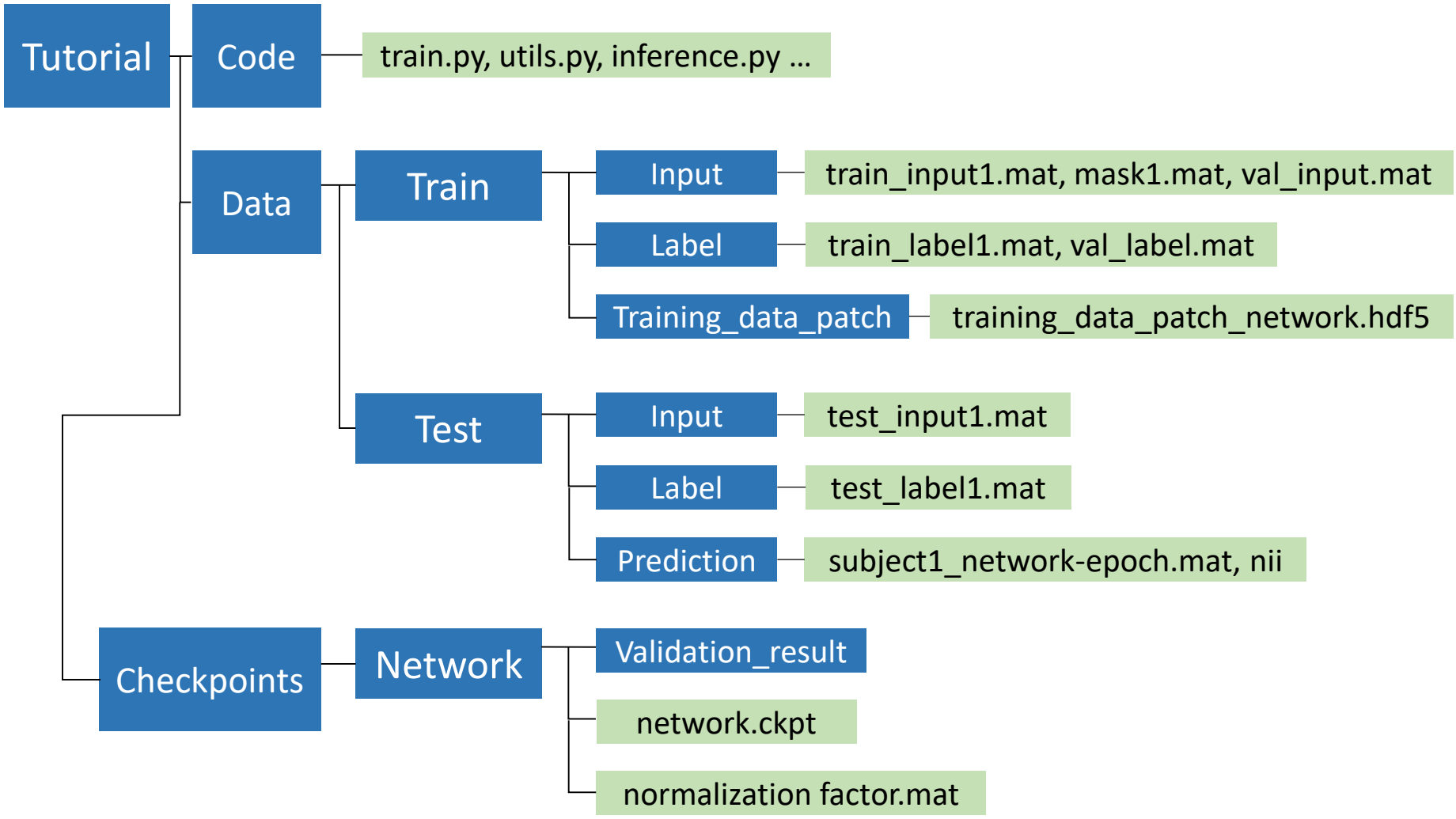


- Checkpoints – Network file
- Code – Training & inference code in python
- Data – Training, validation, test data



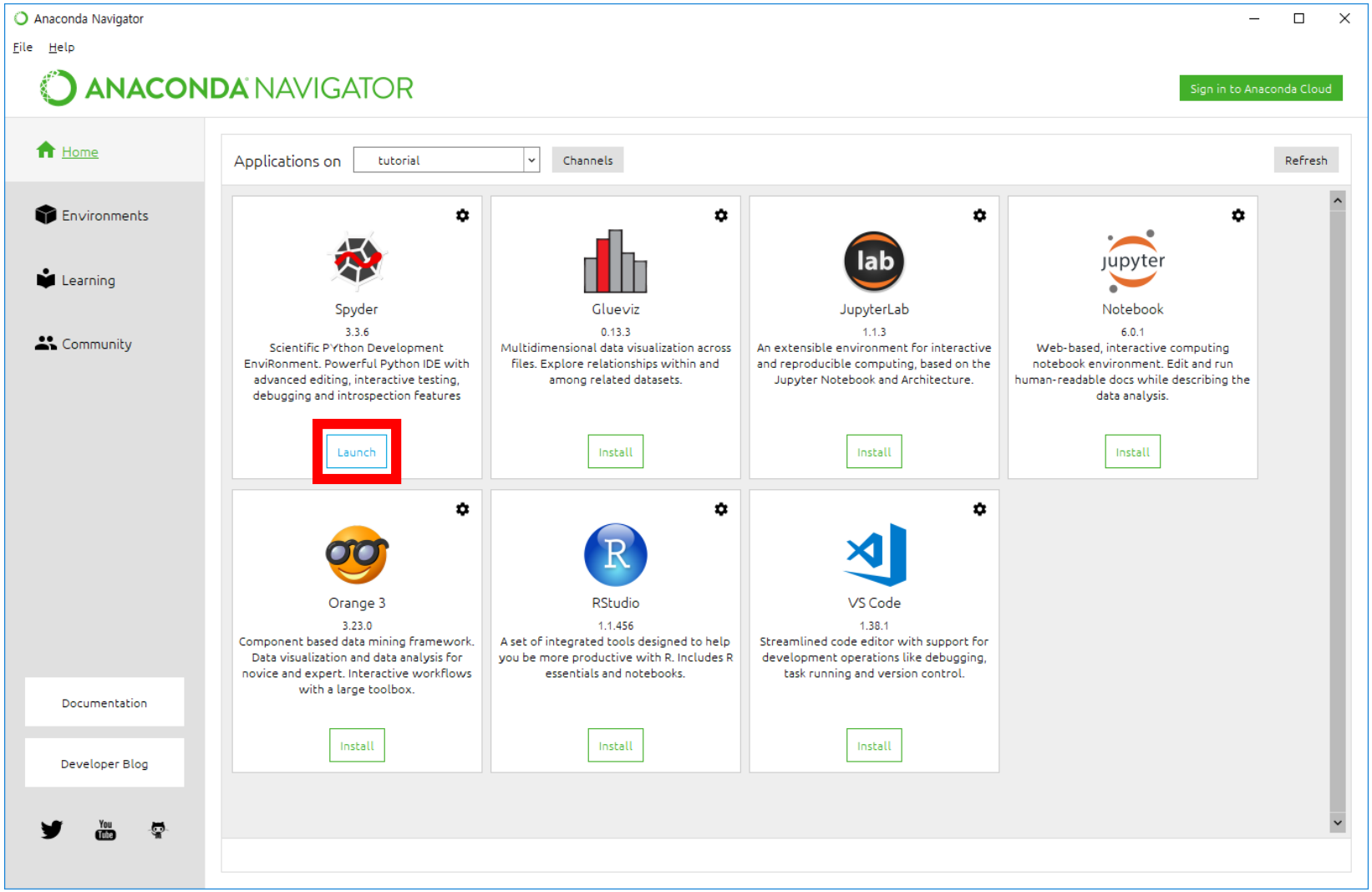
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Structure Overview



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

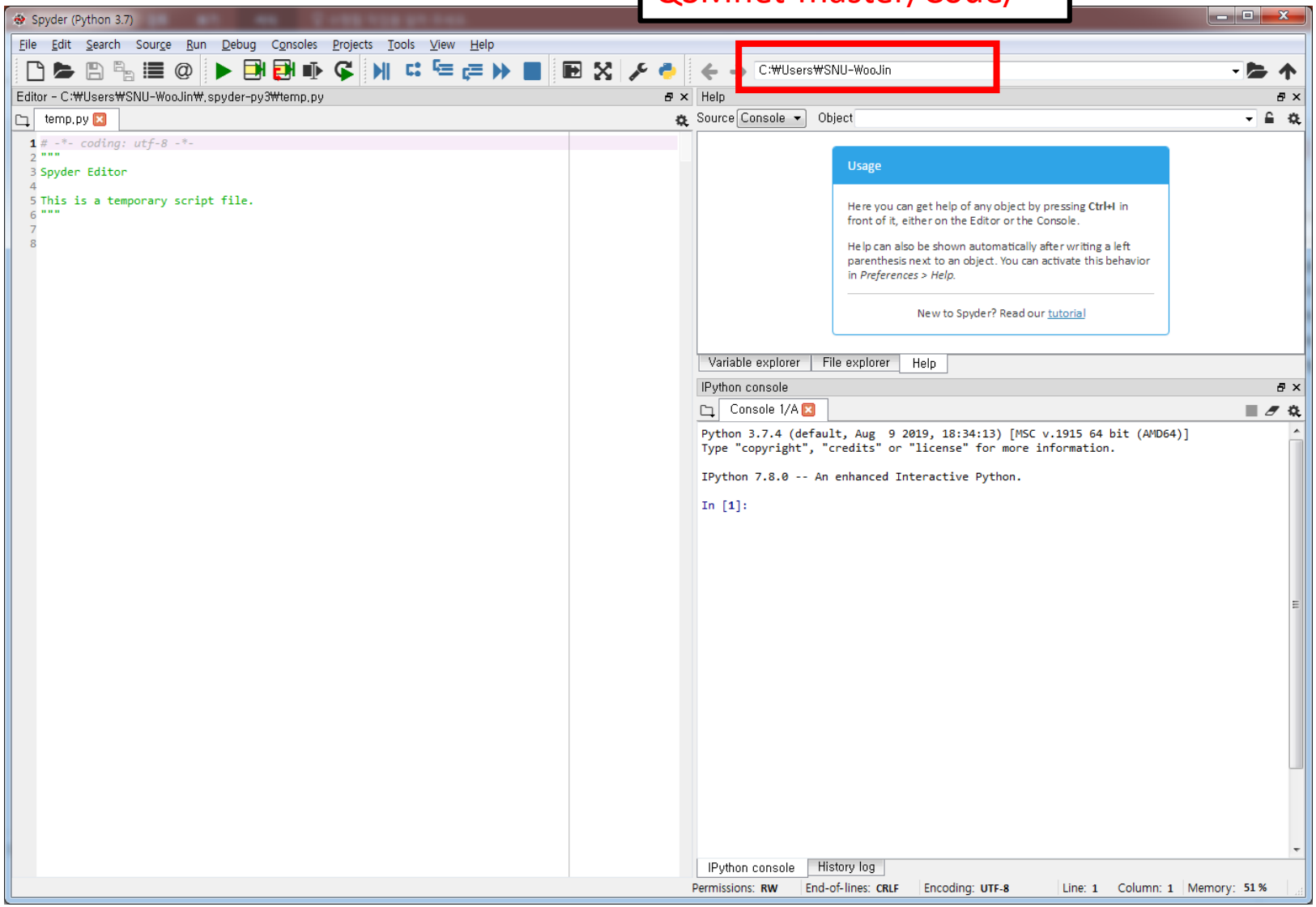
Open Spyder



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Training code

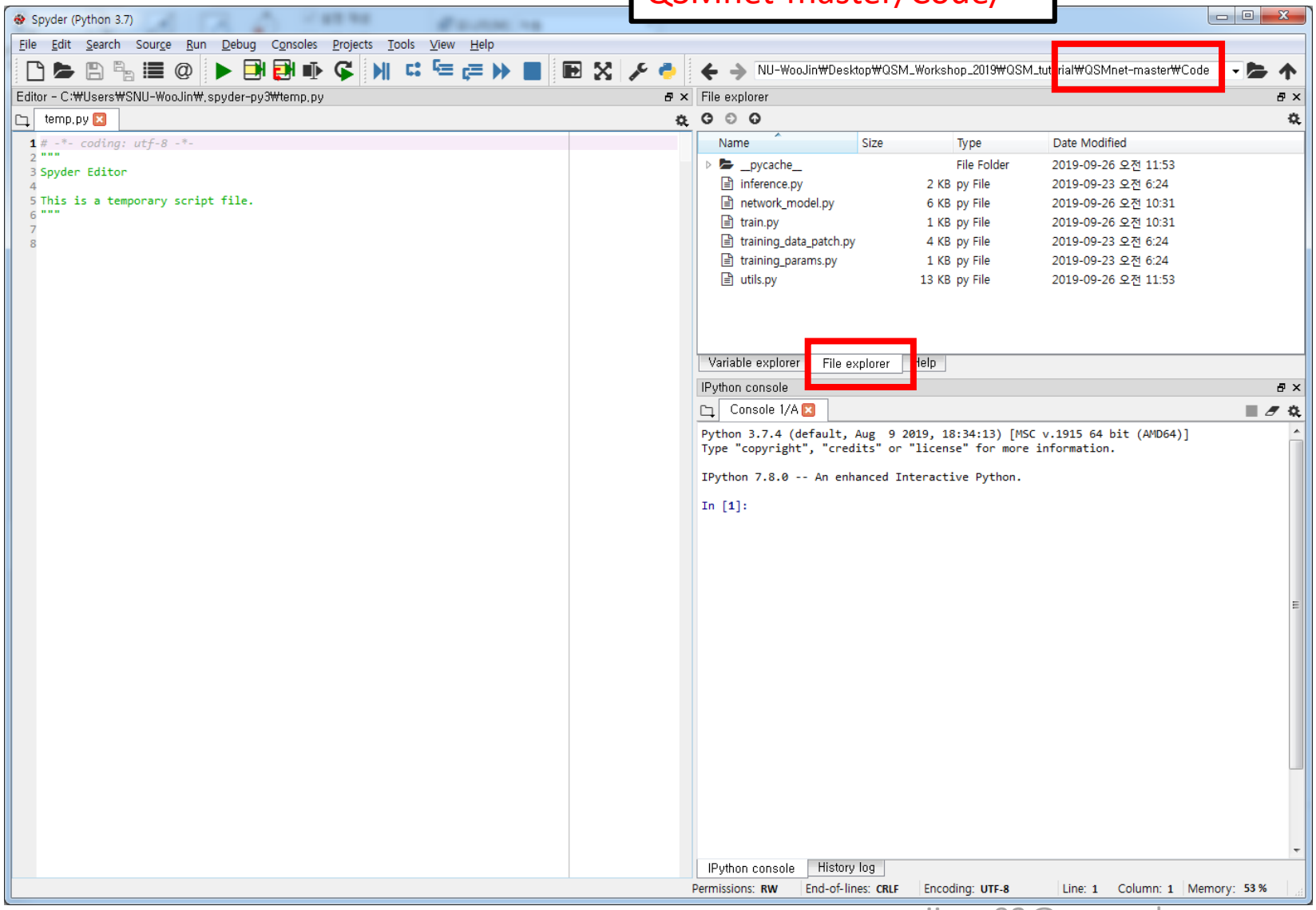
QSMnet-master/Code/



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Training code

QSMnet-master/Code/

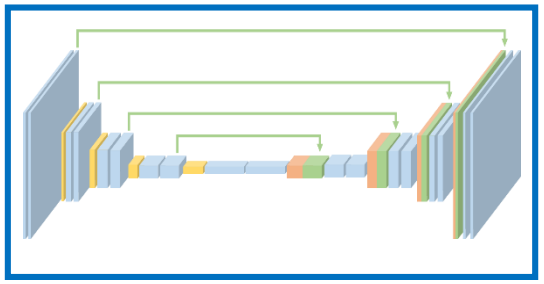


wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Goal of Hands on

- 1) Build Toy deep neural network model
- 2) Programming training process
- 3) Load 'Full deep neural network' and inference on test set

Toy deep neural network



Full deep neural network

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Goal of Hands on

- 1) Build Toy deep neural network model

Open network_model.py!!

Toy deep neural network

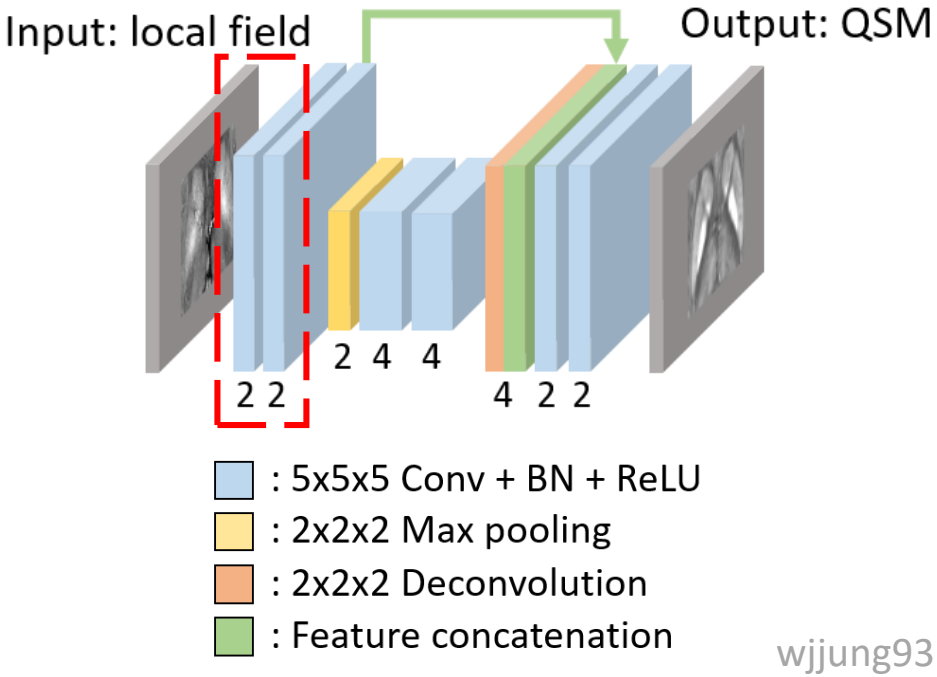
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Build Toy 3D U-net

network_model.py

```
15      conv11 = Conv3d('conv11', x, 2, [5, 5, 5], act_func, reuse, isTrain)
16      conv12 = Conv3d('conv12', conv11, 2, [5, 5, 5], act_func, reuse, isTrain)
```

- Convolution: $conv3d(layer\ name, input, output\ channel, kernel\ size, activation\ function, reuse, isTrain)$
- 15: input dim (batch size, 32, 32, 32, 1) -> output dim (batch size, 32, 32, 32, 2)
- 16: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 32, 32, 32, 2)



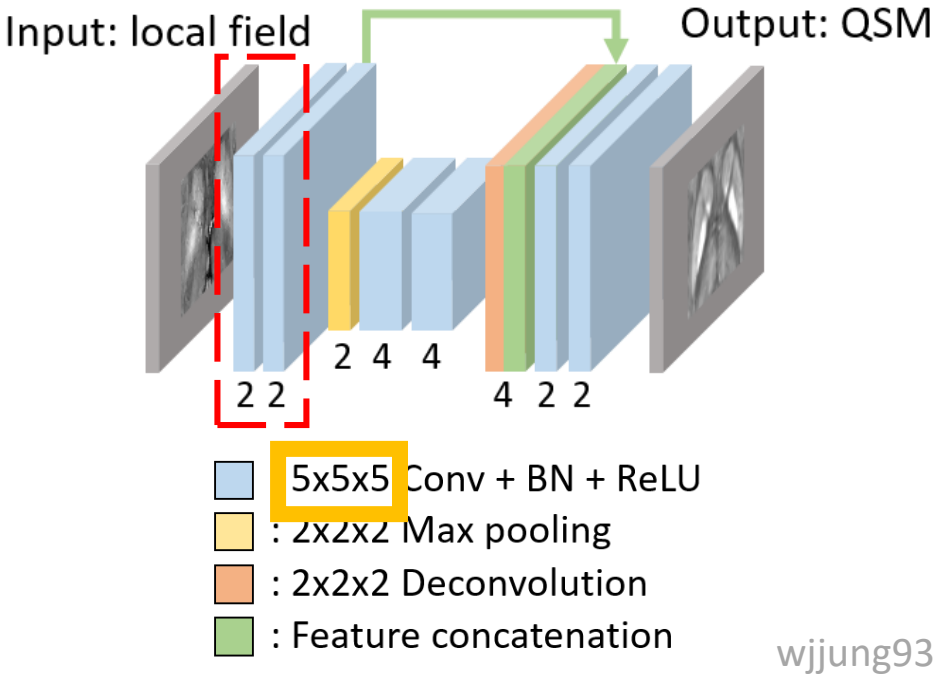
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Build Toy 3D U-net

network_model.py

```
15      conv11 = Conv3d('conv11', x, 2, [5, 5, 5] act_func, reuse, isTrain)
16      conv12 = Conv3d('conv12', conv11, 2, [5, 5, 5], act_func, reuse, isTrain)
```

- Convolution: $conv3d(layer\ name, input, output\ channel, kernel\ size, activation\ function, reuse, isTrain)$
- 15: input dim (batch size, 32, 32, 32, 1) -> output dim (batch size, 32, 32, 32, 2)
- 16: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 32, 32, 32, 2)



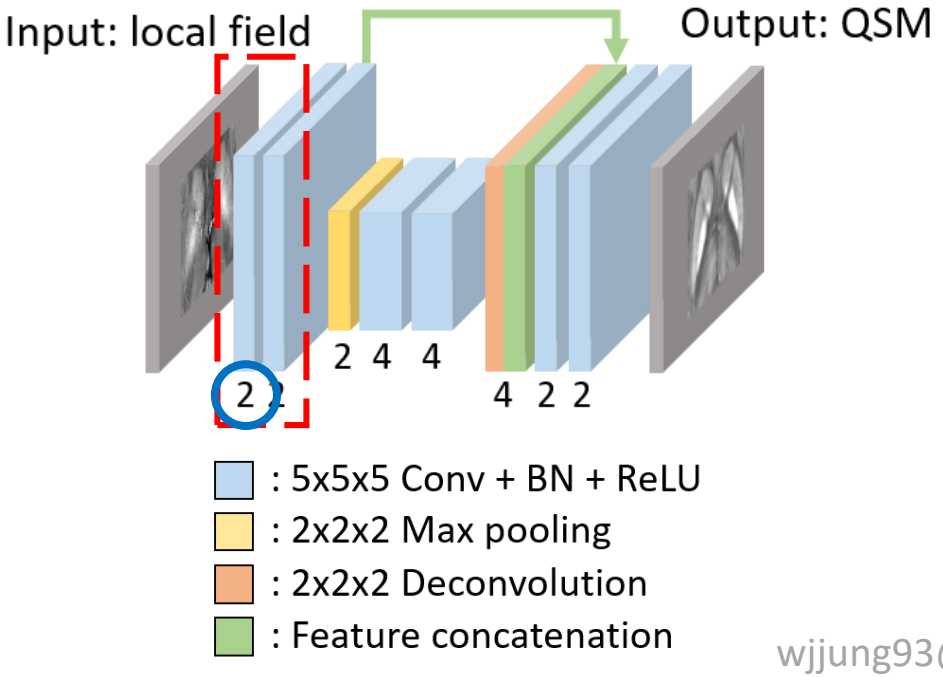
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Build Toy 3D U-net

network_model.py

```
15      conv11 = Conv3d('conv11', x, 2, [5, 5, 5], act_func, reuse, isTrain)
16      conv12 = Conv3d('conv12', conv11, 2, [5, 5, 5], act_func, reuse, isTrain)
```

- Convolution: $conv3d(layer\ name, input, output\ channel, kernel\ size, activation\ function, reuse, isTrain)$
- 15: input dim (batch size, 32, 32, 32, 1) -> output dim (batch size, 32, 32, 32, 2)
- 16: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 32, 32, 32, 2)



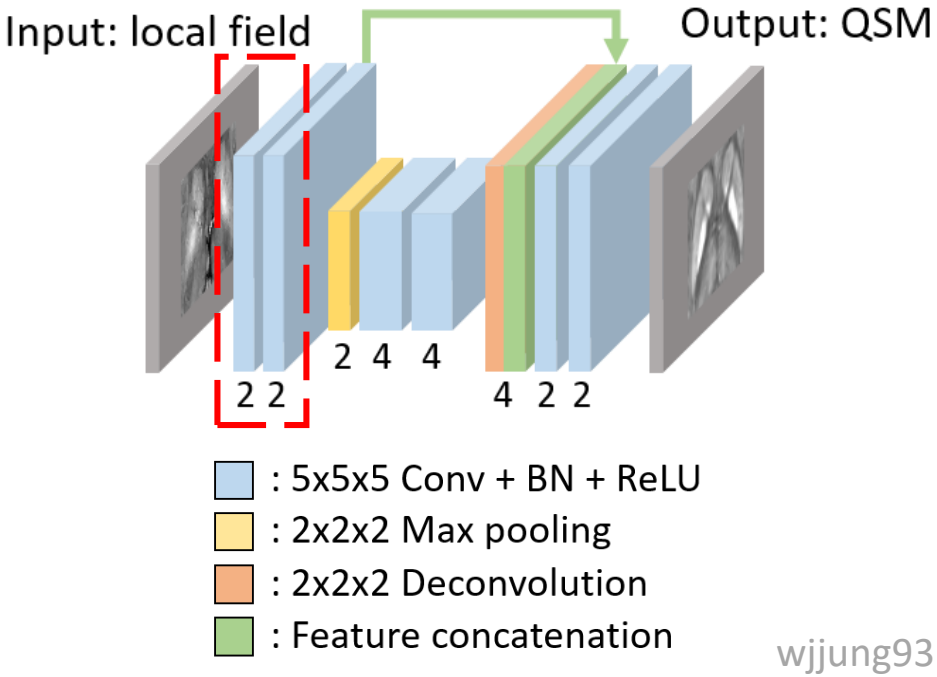
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Build Toy 3D U-net

network_model.py

```
15      conv11 = Conv3d('conv11', x, 2, [5, 5, 5], act_func, reuse, isTrain)
16      conv12 = Conv3d('conv12', conv11, 2, [5, 5, 5], act_func, reuse, isTrain)
```

- act_func: activation function - 'relu'
- reuse: False for first time
- isTrain: True for training, otherwise, freeze parameters



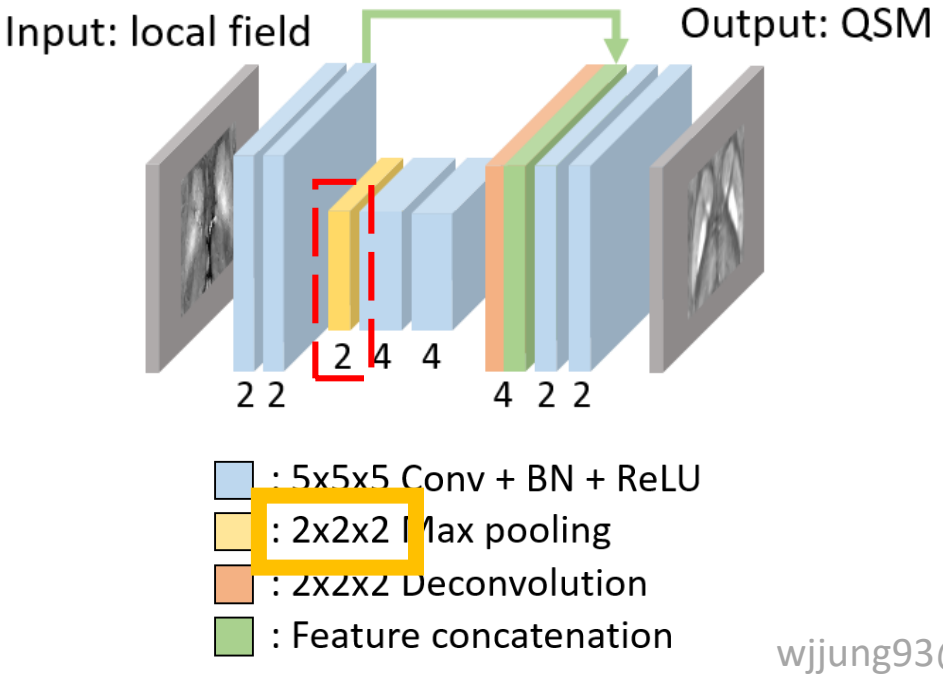
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

network_model.py

```
17      pool1 = Max_pool('maxpool1', conv12, [2, 2, 2], reuse)
```

- Max pooling: $Max\_pool(layer\ name, input, kernel\ size, reuse)$
- 17: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 16, 16, 16, 2)



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

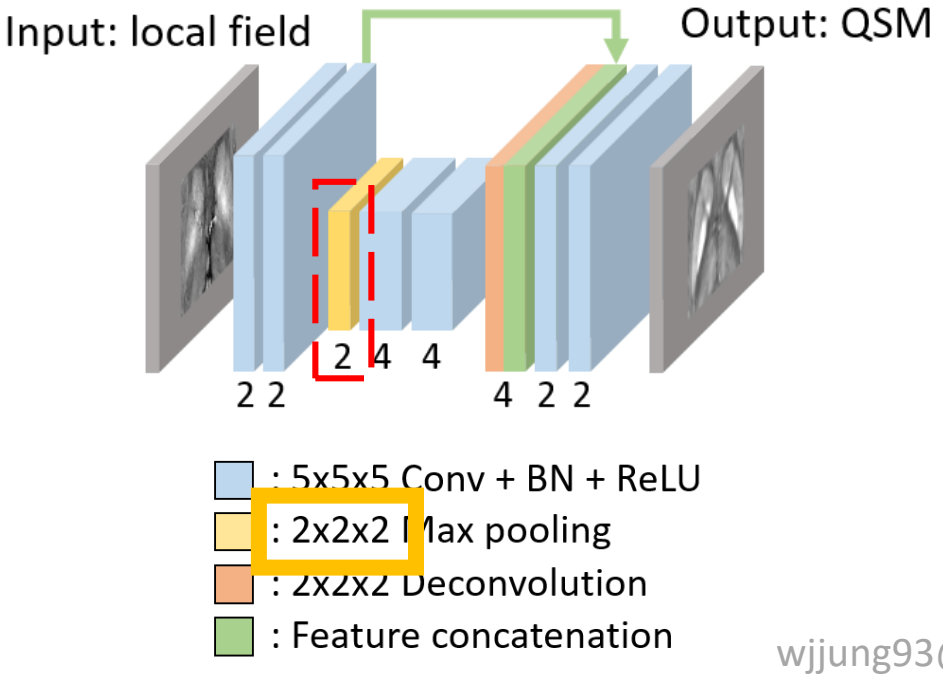
Toy 3D U-net

network_model.py

```
17      pool1 = Max_pool('maxpool1', conv12, [2, 2, 2], reuse)
```

stride

- Max pooling: $Max\_pool(layer\ name, input, kernel\ size, reuse)$
- 17: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 16, 16, 16, 2)



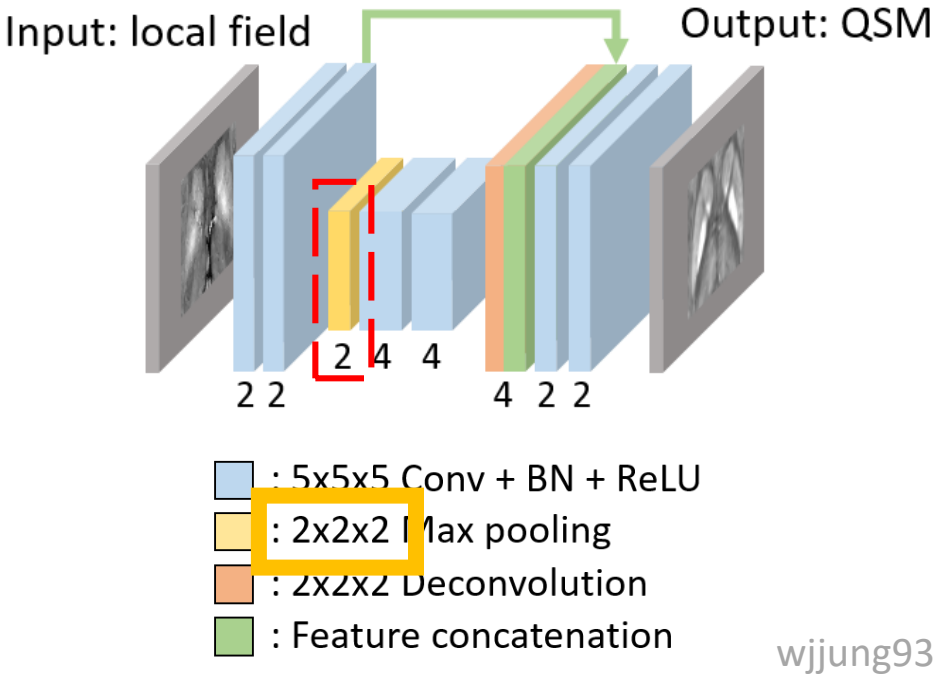
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

network_model.py

```
17      pool1 = Max_pool('maxpool1', conv12, [2, 2, 2], reuse)
```

- Max pooling: $Max\_pool(layer\ name, input, kernel\ size, reuse)$
- 17: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 16, 16, 16, 2)



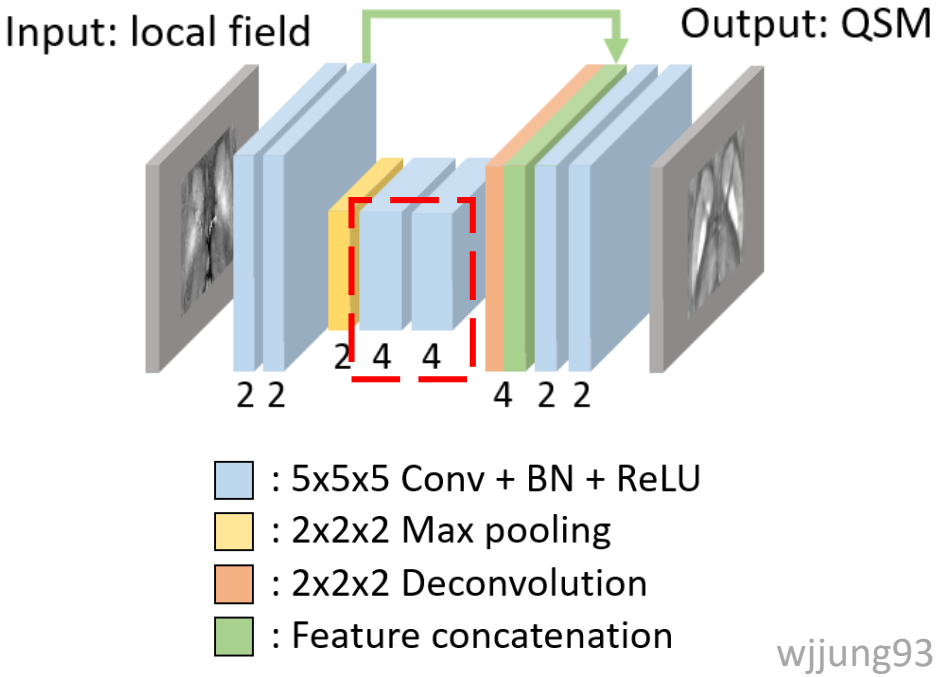
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

network_model.py

```
19      conv21 = Conv3d('conv21', pool1, 4, [5, 5, 5], act_func, reuse, isTrain)
20      conv22 = Conv3d('conv22', conv21, 4, [5, 5, 5], act_func, reuse, isTrain)
```

- 19: input dim (batch size, 16, 16, 16, 2) -> output dim (batch size, 16, 16, 16, 4)
- 20: input dim (batch size, 16, 16, 16, 4) -> output dim (batch size, 16, 16, 16, 4)



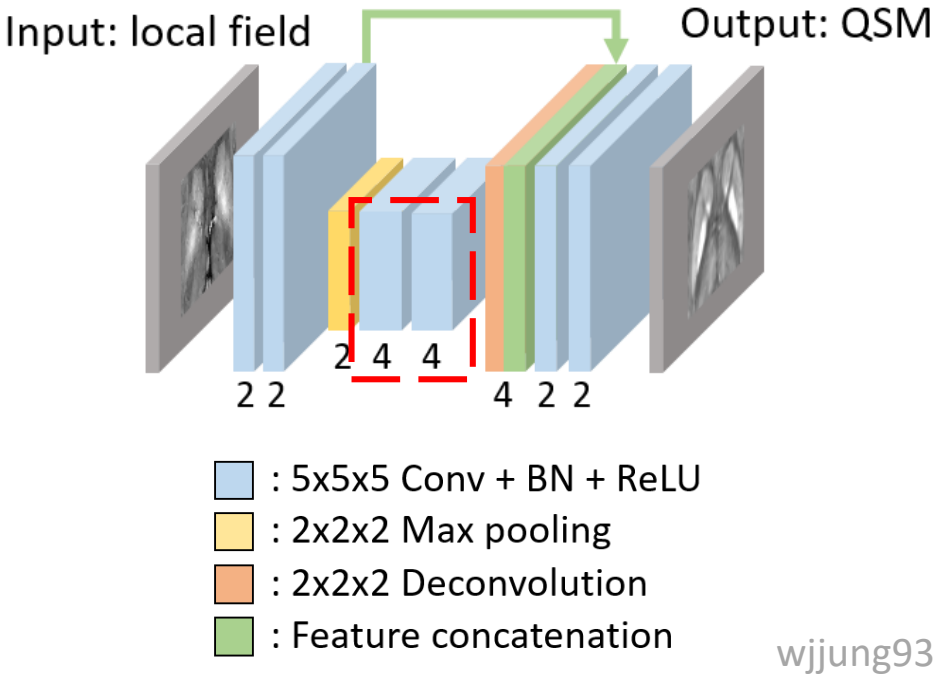
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

network_model.py

```
19      conv21 = Conv3d('conv21', pool1, 4, [5, 5, 5], act_func, reuse, isTrain)
20      conv22 = Conv3d('conv22', conv21, 4, [5, 5, 5], act_func, reuse, isTrain)
```

- 19: input dim (batch size, 16, 16, 16, 2) -> output dim (batch size, 16, 16, 16, 4)
- 20: input dim (batch size, 16, 16, 16, 4) -> output dim (batch size, 16, 16, 16, 4)



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

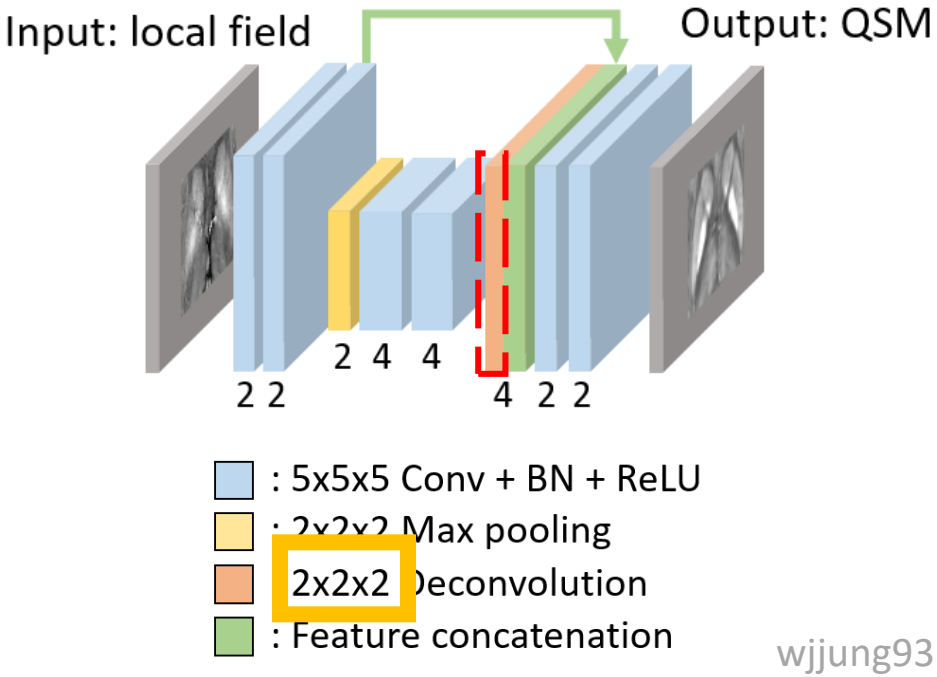
network_model.py

```
22      deconv1 = Deconv3d('deconv1', conv22, 2, [2, 2, 2], [2, 2, 2], reuse, isTrain)
23      concat1 = Concat('concat1', conv12, deconv1, reuse)
```

- Deconvolution:

  $Deconv3d(layer\ name, input, output\ channel, kernel\ size, stride, reuse, isTrain)$

- 22: input dim (batch size, 16, 16, 16, 4) -> output dim (batch size, 32, 32, 32, 2)



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

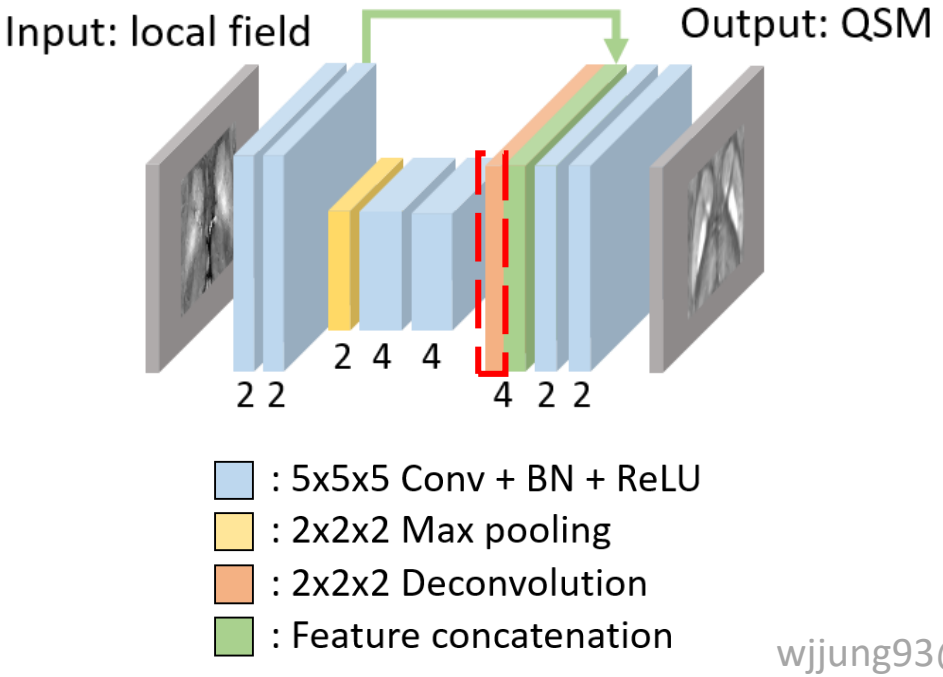
network_model.py

```
22      deconv1 = Deconv3d('deconv1', conv22, 2, [2, 2, 2], [2, 2, 2], reuse, isTrain)
23      concat1 = Concat('concat1', conv12, deconv1, reuse)
```

- Deconvolution:

  $Deconv3d(layer\ name, input, output\ channel, kernel\ size, stride, reuse, isTrain)$

- 22: input dim (batch size, 16, 16, 16, 4) -> output dim (batch size, 32, 32, 32, 2)



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

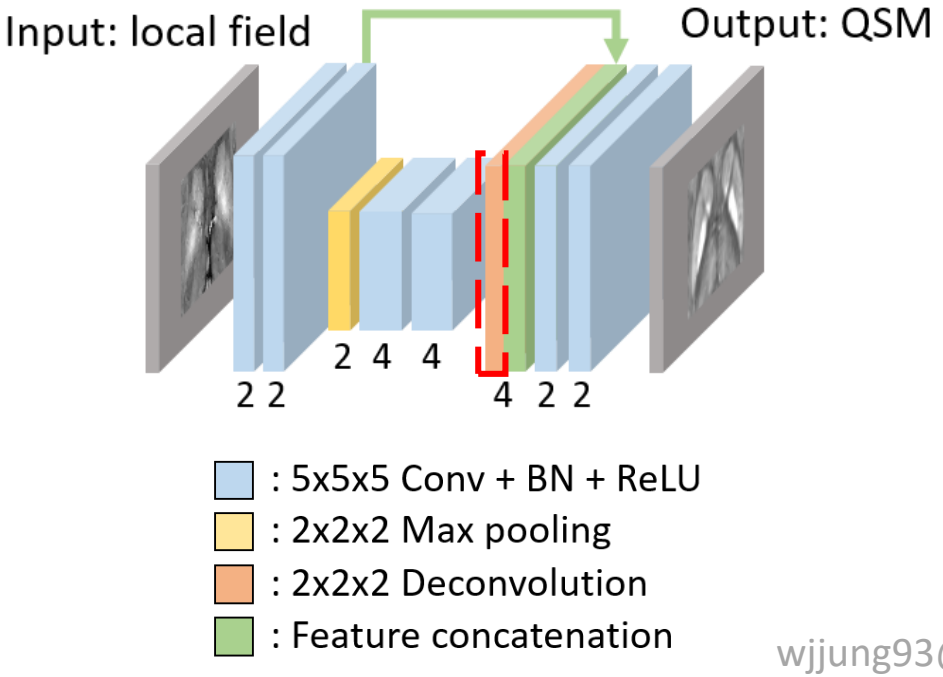
network_model.py

```
22      deconv1 = Deconv3d('deconv1', conv22, 2, [2, 2, 2], [2, 2, 2], reuse, isTrain)
23      concat1 = Concat('concat1', conv12, deconv1, reuse)
```

- Deconvolution:

  $Deconv3d(layer\ name, input, output\ channel, kernel\ size, stride, reuse, isTrain)$

- 22: input dim (batch size, 16, 16, 16, 4) -> output dim (batch size, 32, 32, 32, 2)



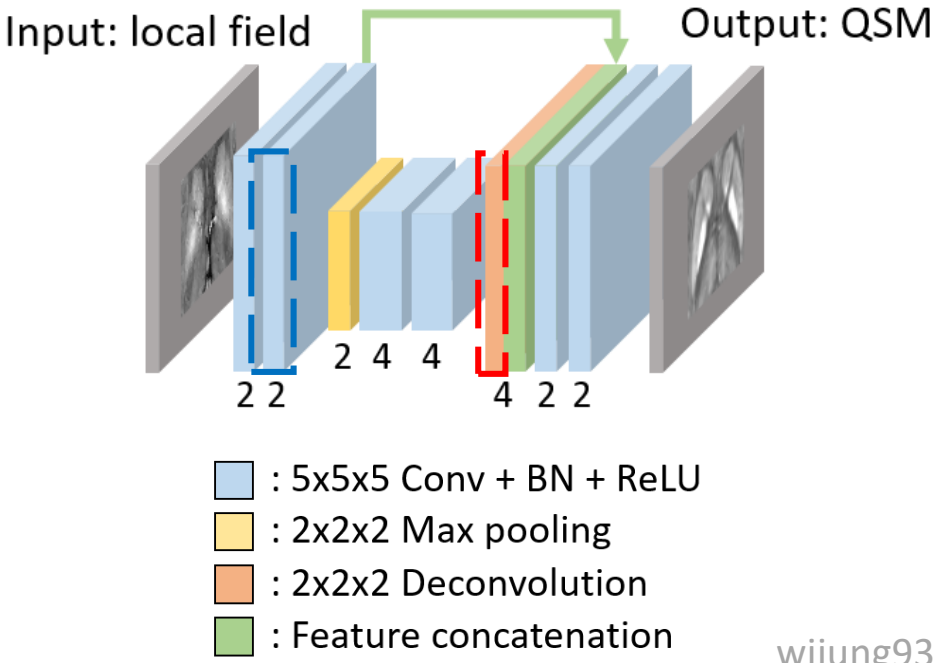
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

network_model.py

```
22      deconv1 = Deconv3d('deconv1', conv22, 2, [2, 2, 2], [2, 2, 2], reuse, isTrain)
23      concat1 = Concat('concat1', conv12, deconv1, reuse)
```

- Concatenation: $Concat(layer\ name, x, y, reuse)$
- 37: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 32, 32, 32, 4)



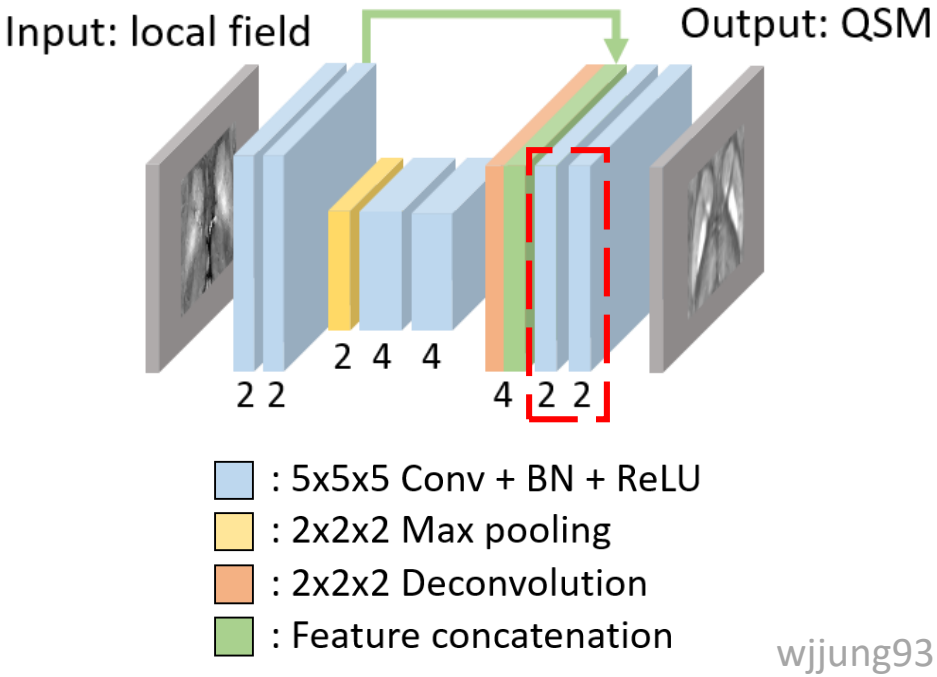
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

network_model.py

```
25      conv31 = Conv3d('conv31', concat1, 2, [5, 5, 5], act_func, reuse, isTrain)
26      conv32 = Conv3d('conv32', conv31, 2, [5, 5, 5], act_func, reuse, isTrain)
```

- 25: input dim (batch size, 32, 32, 32, 4) -> output dim (batch size, 32, 32, 32, 2)
- 26: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 32, 32, 32, 2)



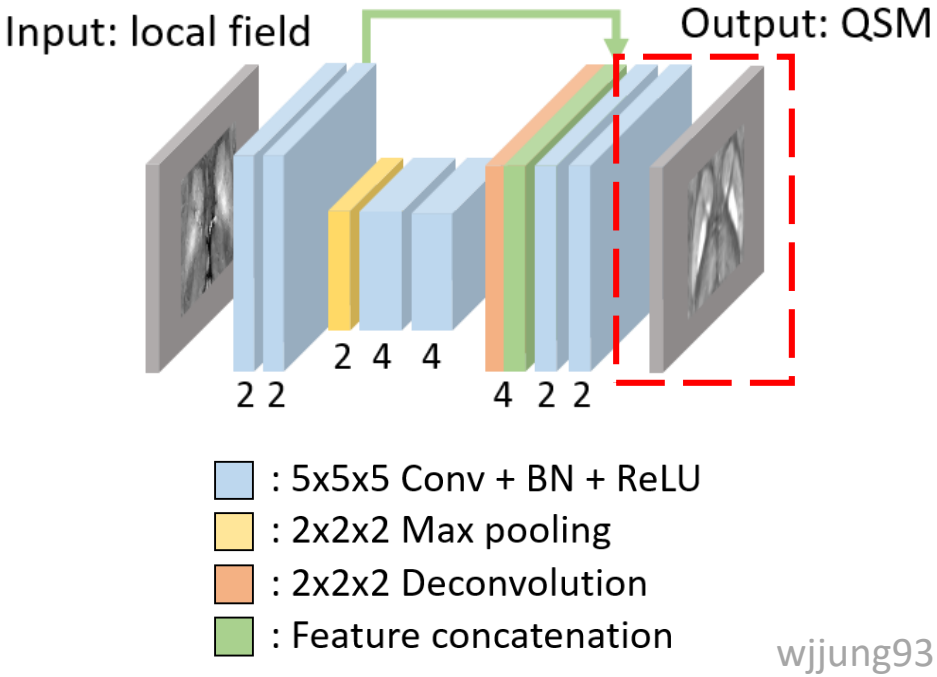
wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Toy 3D U-net

network_model.py

```
28      out_image = Conv('out_image', conv32, 1, [1, 1, 1], reuse, isTrain)
```

- 28: input dim (batch size, 32, 32, 32, 2) -> output dim (batch size, 32, 32, 32, 1)



wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Goal of Hands on

- 1) Build Toy deep neural network model
- 2) Programming training process
- 3) Load ‘Full deep neural network’ and inference on test set

Toy deep neural network

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Training_params.py

```
data_folder = "../Data/"
net_name = 'QSMnet'
PS = '32' # patch_size


C = {
    'data': {
        'data_folder': data_folder,
        'train_data_path': data_folder + 'Train/Training_data_patch/training_data_patch_' + net_name + '_' + PS + '.hdf5',
        'val_input_path': data_folder + 'Train/Input/val_input.mat',
        'val_label_path': data_folder + 'Train/Label/val_label.mat',
        'save_path': '../Checkpoints/' + net_name + '_'+ PS + '/'
    },

    'train': {
        'batch_size': 20, # batch size
        'learning_rate': 0.001, # initial learning rate
        'train_epochs': 25, # The number of training epochs
        'save_step': 5 # Step for saving network
    },
    'validation': {
        'display_step': 2, # display step of validation images
        'display_slice_num': [52,72,92,112], # slice number of validation images for displaying
    }
}
```

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Train.py

train.py | inference.py | network_model.py | utils.py | training_params.py

```python
import tensorflow as tf
import numpy as np
import time

from training_params import *
from utils import *
from network_model import *

#
# Description:
#  Training code of QSMnet and QSMnet+
#
#  Copyright @ Woojin Jung & Jaeyeon Yoon
#  Laboratory for Imaging Science and Technology
#  Seoul National University
#  email : wjjung93@snu.ac.kr
#

#%% Train
def train():
    tf.compat.v1.reset_default_graph()
    #%% Loading dataset
    train_dataset = dataset() # Training set, validation set

    #%% Declaration of tensor
    X = tf.compat.v1.placeholder("float", [None, PS, PS, PS, 1]) # Training input
    Y = tf.compat.v1.placeholder("float", [None, PS, PS, PS, 1]) # Training label

    N = np.shape(train_dataset.tefield) # matrix size of validation set
    X_val = tf.compat.v1.placeholder("float", [None, N[1], N[2], N[3], 1]) # Validation input
    Y_val = tf.compat.v1.placeholder("float", [None, N[1], N[2], N[3], 1]) # Validation label
    keep_prob = tf.compat.v1.placeholder("float") #dropout rate

    #%% Definition of model
    predX = qsmnet_toy(X, 'relu', False, True)
    predX_val = qsmnet_toy(X_val, 'relu', True, False)

    #%% Definition of loss function
    loss = l1(predX, Y)
    loss_val = l1(predX_val,Y_val)

    #%% Definition of optimizer
    train_op = tf.compat.v1.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)

    #%% Generate saver instance
    qsm_saver = tf.compat.v1.train.Saver()

    #%% Running session
    Training_network(train_dataset, X, Y, X_val, Y_val, predX_val, loss, loss_val, train_op, keep_prob, qsm_saver)

if __name__ == '__main__':
    start_time = time.time()
    train()
    print("Total training time : {} sec".format(time.time() - start_time))
```

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Train.py

```
22  #%% Loading dataset
23  train_dataset = dataset() # Training set, validation set
24
25  #%% Declaration of tensor
26  X = tf.compat.v1.placeholder("float", [None, PS, PS, PS, 1]) # Training input
27  Y = tf.compat.v1.placeholder("float", [None, PS, PS, PS, 1]) # Training label
28
29  N = np.shape(train_dataset.tefield) # matrix size of validation set
30  X_val = tf.compat.v1.placeholder("float", [None, N[1], N[2], N[3], 1]) # Validation input
31  Y_val = tf.compat.v1.placeholder("float", [None, N[1], N[2], N[3], 1]) # Validation label
32  keep_prob = tf.compat.v1.placeholder("float") #dropout rate
```

- 23 : Data loading – training set, validation set

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Train.py

```
22  #%% Loading dataset
23  train_dataset = dataset() # Training set, validation set
24
25  #%% Declaration of tensor
26  X = tf.compat.v1.placeholder("float", [None, PS, PS, PS, 1])
27  Y = tf.compat.v1.placeholder("float", [None, PS, PS, PS, 1])
28
29  N = np.shape(train_dataset.tefield) # matrix size of validation set
30  X_val = tf.compat.v1.placeholder("float", [None, N[1], N[2], N[3], 1])
31  Y_val = tf.compat.v1.placeholder("float", [None, N[1], N[2], N[3], 1])
32  keep_prob = tf.compat.v1.placeholder("float") #dropout rate
```

Training set

Validation set

- 23 : Data loading – training set, validation set
- 26 ~ 32 : Declaration of tensor by placeholder

- 26 ~ 27 : Training input & label dimension
  - [Batch size, Patch size_x, Patch size_y, Patch size_z, channel = 1]
- 30 ~ 31 : Validation input & label dimension
  - [Batch size, x, y, z, channel = 1]

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Train.py

```
34  #%% Definition of model
35  predX = qsmnet_toy(X, 'relu', False, True)
36  predX_val = qsmnet_toy(X_val, 'relu', True, False)
37
38  #%% Definition of loss function
39  loss = l1(predX, Y)
40  loss_val = l1(predX_val,Y_val)
41
42  #%% Definition of optimizer
43  train_op = tf.compat.v1.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)
44
45  #%% Generate saver instance
46  qsm_saver = tf.compat.v1.train.Saver()
47
48  #%% Running session
49  Training_network(train_dataset, X, Y, X_val, Y_val, predX_val, loss, loss_val, train_op, keep_prob, qsm_saver)
```

- 35 ~ 36: Definition of model
  - Training inputs: X
  - Activation function: ‘relu’ (‘relu’ or ‘leaky_relu’)
  - Reuse: ‘False’ for training, ‘True’ for validation
  - IsTrain: ‘True’ for training, ‘False’ for validation

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Train.py

```
34    #%% Definition of model
35    predX = qsmnet_toy(X, 'relu', False, True)
36    predX_val = qsmnet_toy(X_val, 'relu', True, False)
37
38    #%% Definition of loss function
39    loss = l1(predX, Y)
40    loss_val = l1(predX_val,Y_val)
41
42    #%% Definition of optimizer
43    train_op = tf.compat.v1.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)
44
45    #%% Generate saver instance
46    qsm_saver = tf.compat.v1.train.Saver()
47
48    #%% Running session
49    Training_network(train_dataset, X, Y, X_val, Y_val, predX_val, loss, loss_val, train_op, keep_prob, qsm_saver)
```

- 39 ~ 40: Loss function, custom defined l1 loss in utils.py
  - Prediction image: predX
  - Label: Y
- 43: Adam optimizer
  - learning rate
  - minimize(loss)

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Train.py

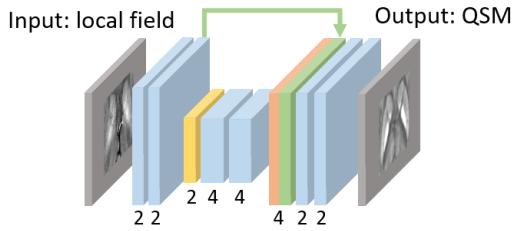
```
34  #%% Definition of model
35  predX = qsmnet_toy(X, 'relu', False, True)
36  predX_val = qsmnet_toy(X_val, 'relu', True, False)
37
38  #%% Definition of loss function
39  loss = l1(predX, Y)
40  loss_val = l1(predX_val,Y_val)
41
42  #%% Definition of optimizer
43  train_op = tf.compat.v1.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)
44
45  #%% Generate saver instance
46  qsm_saver = tf.compat.v1.train.Saver()
47
48  #%% Running session
49  Training_network(train_dataset, X, Y, X_val, Y_val, predX_val, loss, loss_val, train_op, keep_prob, qsm_saver)
```

- 46: Save instance for network
- 49: Training code in utils.py

    - hyper parameters (batch size, learning rate, epochs ..) defined in "Training_params.py"

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

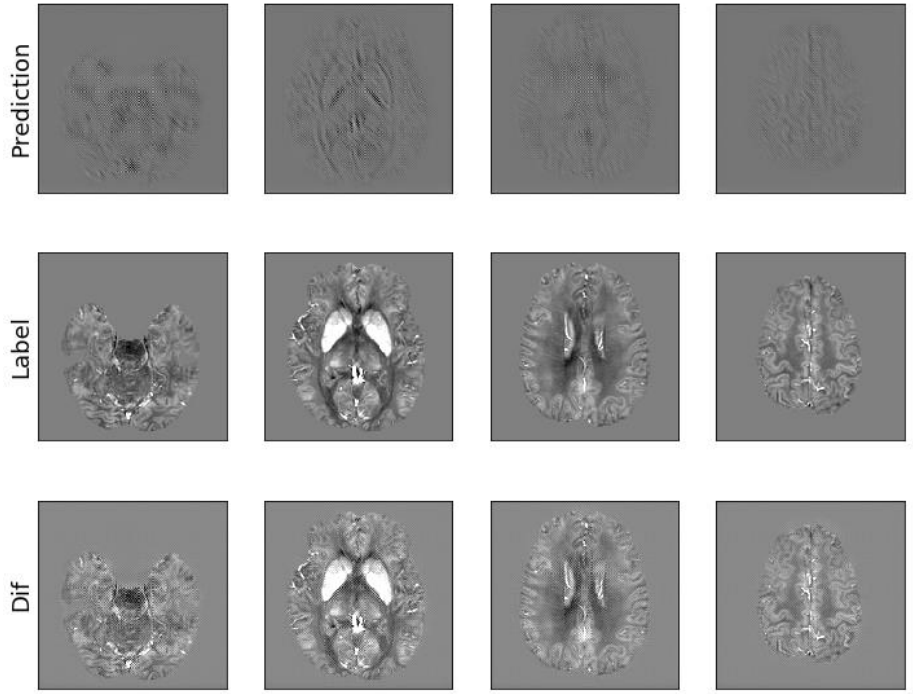
Toy 3D U-net results



- Press F5 in train.py

"Training failed"

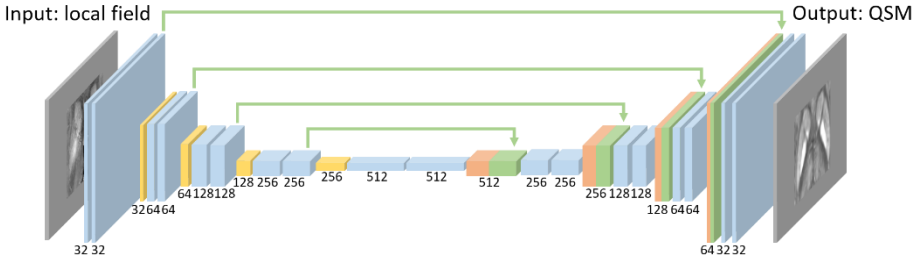
```
In [1]: runfile('D:/QSM_tutorial/QSMnet-master/Code/train.py', wdir
WARNING: Logging before flag parsing goes to stderr.
W0922 15:04:15.745205 43364 lazy_loader.py:50]
The TensorFlow contrib module will not be included in TensorFlow 2.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/201809
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an iss

Training Start!
100%|██████████| 50/50 [05:13<00:00,  6.91s/it]
Epoch: 0001 Training_cost= 7.88741
100%|██████████| 50/50 [05:12<00:00,  5.87s/it]
Epoch: 0002 Training_cost= 7.76975
Epoch: 0002 Validation_cost= 0.71658
```

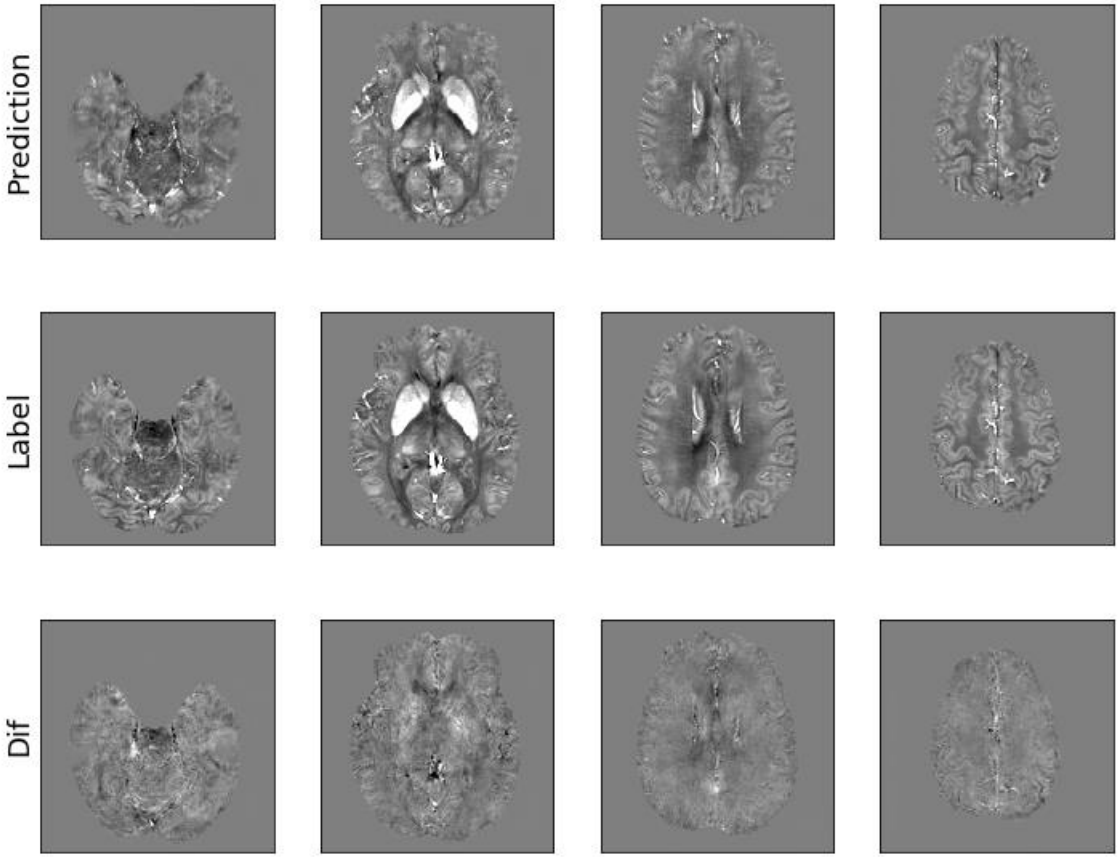


Need more layers and data!

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Full deep neural network results



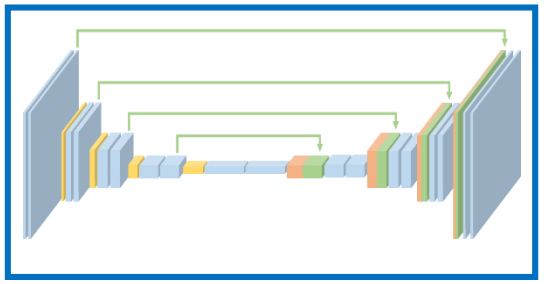


Successfully reconstructed!

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Goal of Hands on

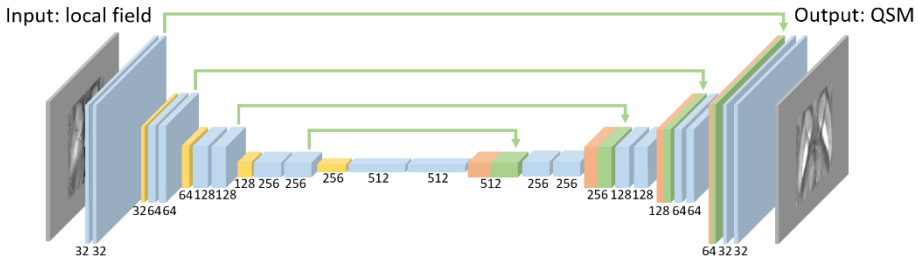
- 1) Build Toy deep neural network model
- 2) Programming training process
- 3) Load 'Full deep neural network' and inference on test set



Full deep neural network

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Inference.py

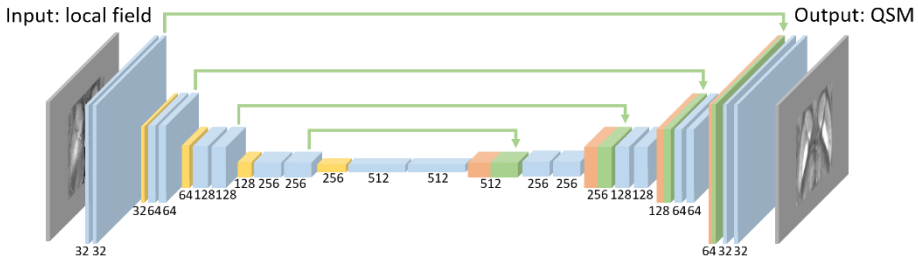



```
21 '''
22 Network model
23 '''
24 network_name = 'QSMnet+_64'
25 net_model = 'qsmnet_deep'
26 sub_num = 1 #number of subjects in testset
27
28 '''
29 File Path
30 '''
31 FILE_PATH_INPUT = '../Data/Test/Input/test_input'
32 FILE_PATH_PRED = '../Data/Test/Prediction/'
```

- 24: network_name - (network name)_(patch size in training)
- 25: net_model - qsmnet_toy, qsmnet_deep …
- 26: sub_num - number of subjects stored in Test/input/test_input.mat
- 32: FILE_PATH_PRED - result of network saved in this directory

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Inference.py



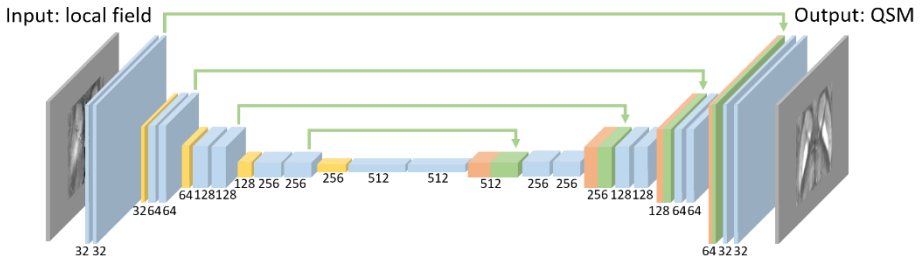

```
56        feed_result = net_func(Z, 'leaky_relu', False, False)

64            print('###########Restore Network###########')
65            saver.restore(sess, '../Checkpoints/'+ network_name + '/' + network_name + '-25')
```

- 56: network_model (Z, activation function, reuse, isTrain)

- 65: saver.restore - load network model & weighting parameters

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Inference.py



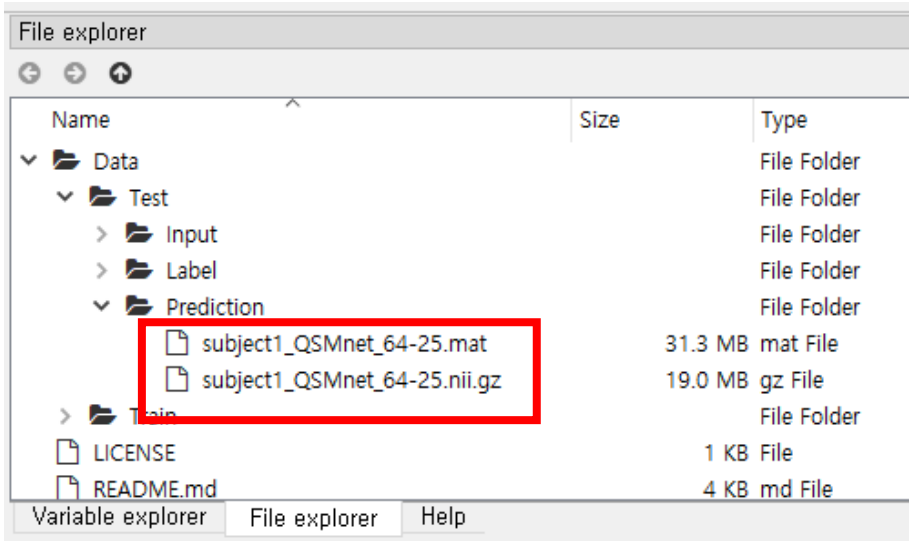

```
68          display_slice_inf([52,72,92,112], result_im)
69          print('##########Saving MATLAB & NII file...##########')
70          scipy.io.savemat(FILE_PATH_PRED + '/subject' + str(i) + '_' + str(network_name) + '-25.mat', mdict={'sus': result_im})
71          save_nii(result_im, FILE_PATH_PRED, 'subject' + str(i) + '_' + str(network_name)+'-25')
```

- 68: display_slice_inf(*slice number array, prediction image*)

  - display example slices in console

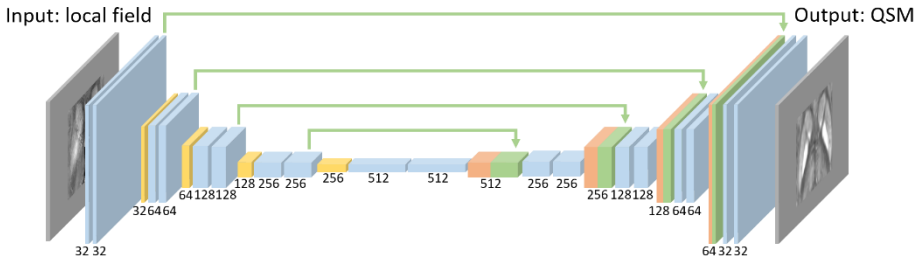
- 70 ~ 71: save result of network in ‘.mat’ & ‘.nii’ format

  - results saved in FILE_PATH_PRED

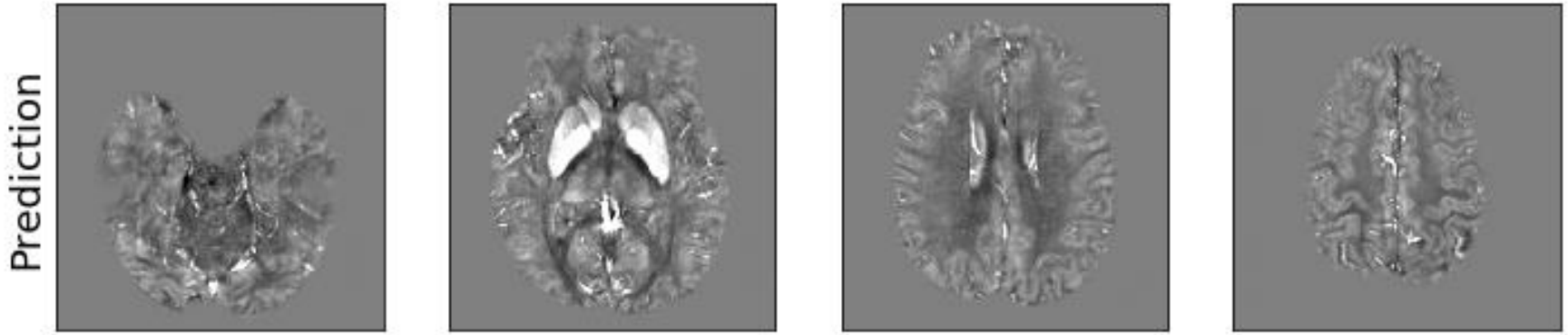


wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Inference.py




```
##########Restore Network##########
Done!
##########Inference...##########
```




```
##########Saving MATLAB & NII file...##########
All done!
Total inference time : 62.31426787376404 sec
```

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

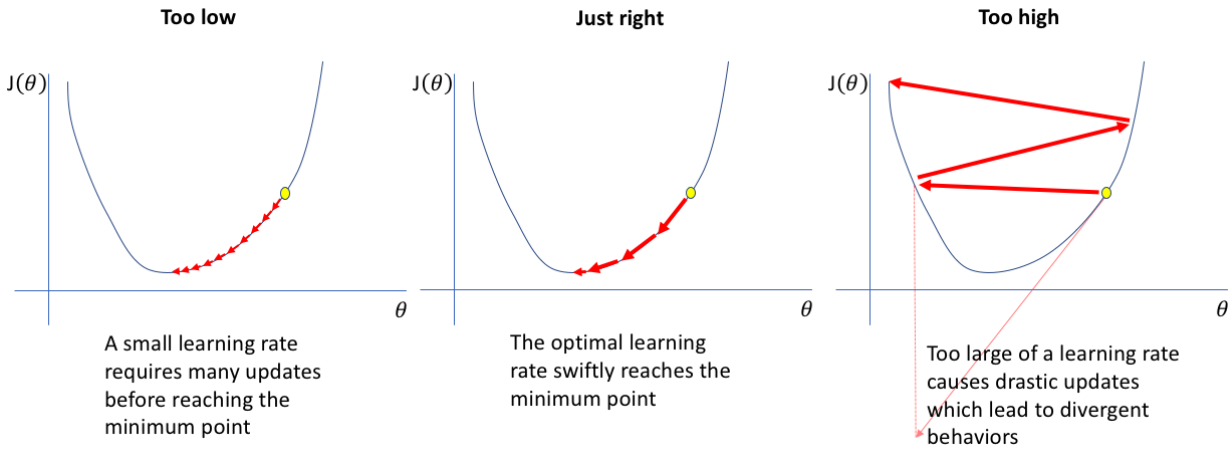
Discussion

- How to decide hyper parameter?
  - Batch size : GPU memory vs. Accurate gradient estimation
  - Optimizer : SGD? RMSProp? Adam?
  - Learning rate
  - Epoch stop point?

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

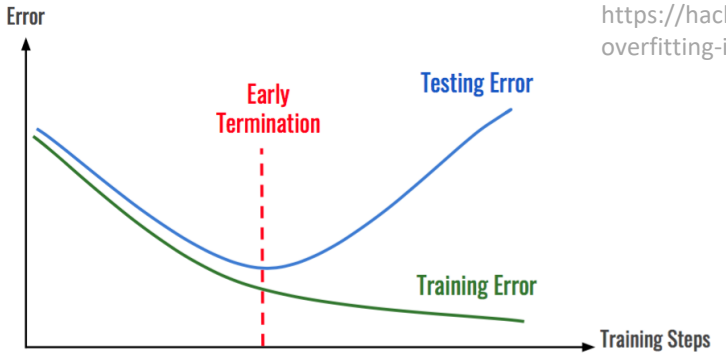
Discussion

- Keep your eyes on loss
  - Learning rate



https://www.jeremyjordan.me/nn-learning-rate/

  - Epoch stop point?



https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Conclusion

- Develop deep neural network trained QSM!
  - Training Toy 3D U-net
  - Inference Full 3D U-net
- Hyper parameter – batch size, optimizer, learning rate, early stop...
- Reference paper
  - Yoon, Jaeyeon, et al. "Quantitative susceptibility mapping using deep neural network: QSMnet." *NeuroImage* 179 (2018): 199-206.
  - Jung, Woojin, et al. "Exploring linearity of deep neural network trained QSM: QSMnet+." *arXiv preprint arXiv:1909.07716* (2019).

wjjung93@snu.ac.kr
https://github.com/SNU-LIST/QSMnet

Thank you!

wjjung93@snu.ac.kr

Laboratory for Imaging Science and Technology @ Seoul National University