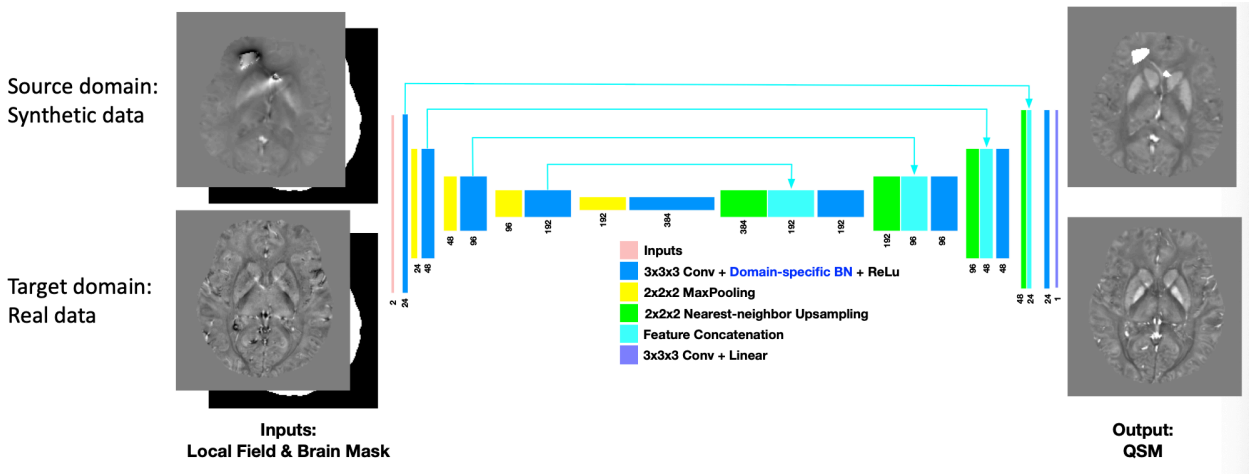
QSMInvNet Example

Juan Liu -- Marquette University and
Kevin Koch -- Medical College of Wisconsin

Please cite using:
Liu, Juan, and Kevin M. Koch. "Non-locally encoder-decoder convolutional network for whole brain QSM inversion." arXiv preprint arXiv:1904.05493 (2019).

Training data: A single QSM dataset, the COSMOS result of 2016 QSM challenge dataset is used to generate the training data cohort through substantial levels of data augmentation, including elastic transforms, contrast adjustments, and the addition of high susceptibility localized sources. Induced field maps are then calculated using conventional dipole convolution to create training input (field map) and label (QSM) pairs.

Neural network training: The applied neural network is a 3D convolutional neural network with encoder decoder architecture. The inputs are local tissue field and brain mask, and the output is QSM. During training, the network has four inputs (local field and brain mask from the synthetic data and real data) and two outputs (QSM outputs). Domain-specific batch normalization is used to address the domain shift problem. All other layers have shared parameters. The loss function includes the L1 loss of source domain QSM $L_{Sx} = \left\| x_s - x_{s\_Label} \right\|_1$, data consistency loss of target domain QSM $L_{Tx} = \left\| W \cdot (e^{jy_t} - e^{jd*x_t}) \right\|_2$ and data regularization loss of target domain QSM $L_{Tx\_TV} = \|G_x(x_t)\|_1 + \left\|G_y(x_t)\right\|_1 + \|G_z(x_t)\|_1$, $Loss = L_{Sx} + \lambda_1 L_{Tx} + \lambda_2 L_{Tx\_TV}$. In the data consistency loss, the dipole inversion is utilized, and W is the data weighting matrix which utilizes a magnitude image.



Network fine-tuning: After network training, the network parameters (weights and biases) are fine-tuned. Fine tuning can happen for each individual dataset, or can be applied for a batch of data (with a domain shift relative to the training data). For this case, we performed the fine tuning on an individual dataset basis. This fine-tuning effectively optimizes the QSM inverse problem on each individual dataset.

First, the model weights from saved pre-trained network are loaded, in which the batch normalization layers are derived from the target domain. The fine-tuning network then only accepts the local field and brain mask from the target domain to do network fine-tuning based on the physical QSM inversion model. The loss function includes a data consistency term $L_{Tx} = \left\| W \cdot (e^{jy_t} - e^{jd*x_t}) \right\|_2$ and data regularization loss of target domain QSM $L_{Tx\_TV} = \|G_x(x_t)\|_1 + \left\|G_y(x_t)\right\|_1 + \|G_z(x_t)\|_1$, $Loss = L_{Tx} + \lambda\, L_{Tx\_TV}$.

This code was constructed using python 2.7, tensorflow 1.12, and keras 2.2.2

How to use the code:

1. use ./training_data/genChi.py to generate synthetic susceptibility maps
2. use ./training_data/qsmFmapCal_gpu.py to do dipole convolution
3. use ./genRDFPatchesSyn.py to get patches of the synthetic data for network training due to memory limitations of hardware required for network training
4. do network training by calling ./network_model/train_model/train.py. In ./network_model/train_model/train.py, change config["datasets_path"], config["targetdata_path"] to the path of the synthetic training data and real target data.
5. After network training, fine-tune the network on individual dataset by calling ./network_model/model_finetune/ train.py.