

Super Resolution

Network Architecture

I build 3-layered network for Super Resolution tasks(2x), 3x3 conv with ReLU as activation. (see **SuperResolutionNetwork** at `lib/network.py`)

Detail hyper parameter setup as below:

- Use Adam optimizer with `lr=.001` and `lr_decay=.0`.
- Use **Mean Squared Error** as loss
- Use PSNR as evaluation metric
- Initialize kernel(*random uniform*) and bias(*zeros*) in convolution layers.

Dataset

Use all `91` and `291` dataset. Create target image, random crop from image after random scale down and create source image using scale down and up by half.

Evaluation

I use PSNR as evaluation metric, (see **SuperResolutionNetwork.metric** at `lib/network.py`) using TensorFlow implementation, `tf.image.psnr`.

Implement **Custom Callback** (see **CustomCallback** at `utils/callbacks.py`) for logging loss, accuracy and sample images. For each interval, write inferenced image summary for one train set and all test set.

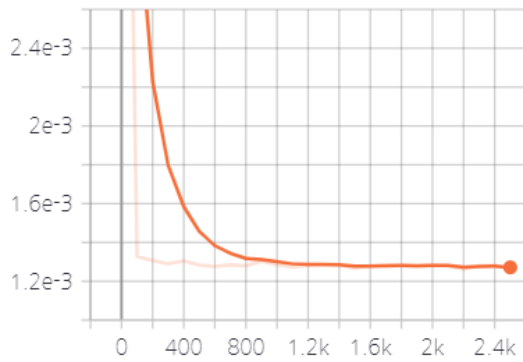
Results

In terms of time, it was only 2500 epoch. However, PSN and loss is saturated after 1000 epoch. Model parameters is saved on `model.hdf5` using keras model save method.

See `./assets` directory for results. Train, prediction and test image is merged for easy compare (re-scale image, prediction image, ground truth image).

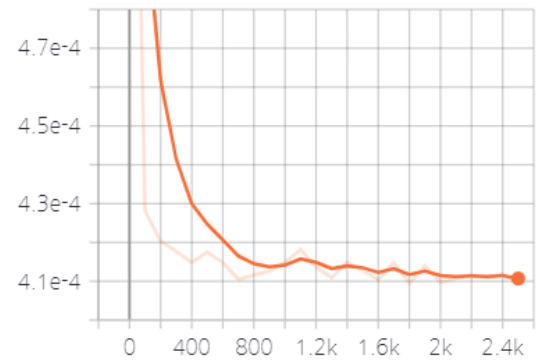
loss

loss



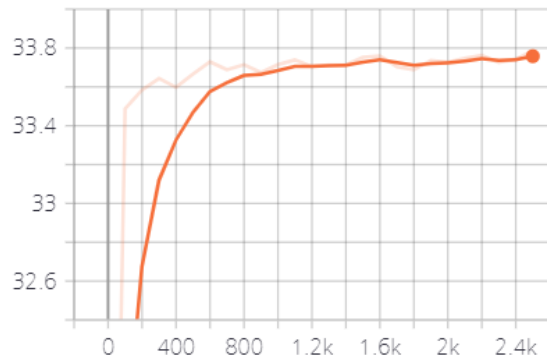
val_loss

val_loss



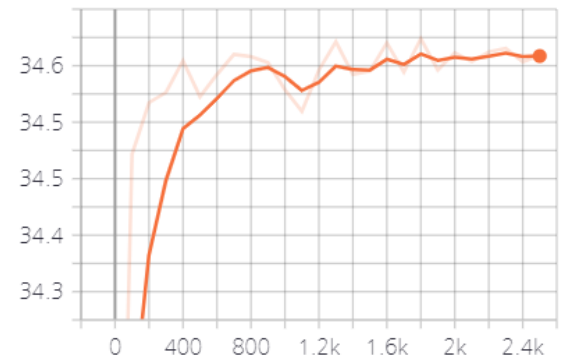
metric

metric



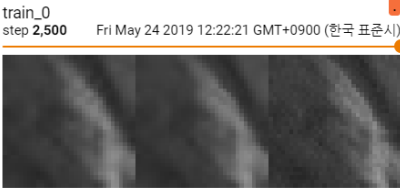
val_metric

val_metric



This is some train samples for check train is running well.

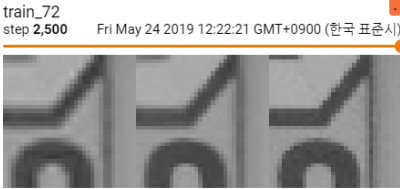
train_0



train_10



train_72



train_1



train_11



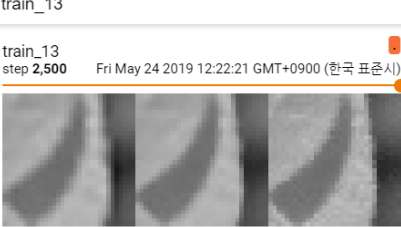
train_73



train_2



train_12



train_74

