titu1994 / **Super-Resolution-using-Generative-Adversarial-Networks**

An implementation of SRGAN model in Keras

titu1994 Remove second Leaky Relu after BN in residual blocks from generator — Latest commit `fde5f80` 17 days ago

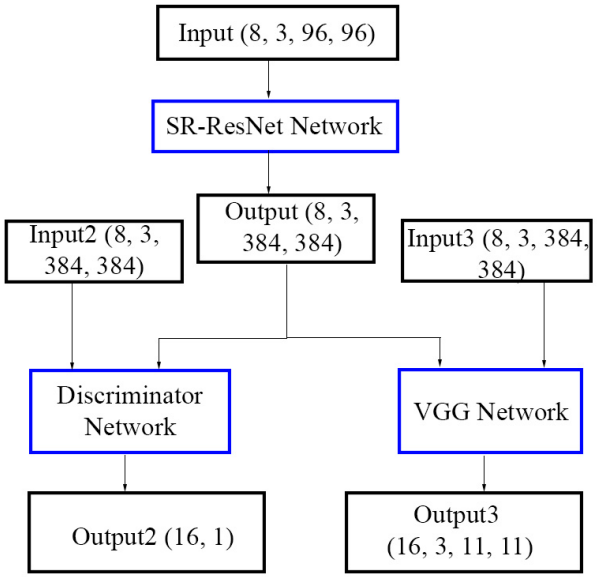
.idea	Remove second Leaky Relu after BN in residual blocks from generator	17 days ago
architecture	Fixed SRGAN full model training	3 months ago
tests	COCO dataset trained sr_resnet_weights	5 months ago
weights	Upscaling now uses Nearest Neighbour upscaling.	3 months ago
README.md	Updated README to reflect changes	3 months ago
keras_ops.py	Fix bug in keras 1.2.1 (TypeError standardize_input_data got an unexp…	2 months ago
layers.py	Normalization now applies tanh scaling when mode='gan' [0, 255] -> [-…	3 months ago
loss.py	Update regularizers losses to new method for keras 1.2.0	3 months ago
models.py	Remove second Leaky Relu after BN in residual blocks from generator	17 days ago
visualize.py	Updated visualization and loss saving procedure	5 months ago

README.md

Super Resolution using Generative Adversarial Networks

This is an implementation of the SRGAN model proposed in the paper Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network in Keras. Note that this project is a work in progress.

A simplified view of the model can be seen as below:



Implementation Details

The SRGAN model is built in stages within models.py. Initially, only the SR-ResNet model is created, to which the VGG

network is appended to create the pre-training model. The VGG weights are freezed as we will not update these weights.

In the pre-train mode:

1. The discriminator model is not attached to the entire network. Therefore it is only the SR + VGG model that will be pretrained first.
2. During pretraining, the VGG perceptual losses will be used to train (using the ContentVGGRegularizer) and TotalVariation loss (using TVRegularizer). No other loss (MSE, Binary crosss entropy, Discriminator) will be applied.
3. Content Regularizer loss will be applied to the VGG Convolution 2-2 layer
4. After pre training the SR + VGG model, we will pretrain the discriminator model.
5. During discriminator pretraining, model is Generaor + Discriminator. Only binary cross entropy loss is used to train the Discriminator network.

In the full train mode:

1. The discriminator model is attached to the entire network. Therefore it creates the SR + GAN + VGG model (SRGAN)
2. Discriminator loss is also added to the VGGContentLoss and TVLoss.
3. Content regularizer loss is applied to the VGG Convolution 5-3 layer. (VGG 16 is used instead of 19 for now)

Usage

Currently, models.py contains most of the code to train and create the models. To use different modes, uncomment the parts of the code that you need.

Note the difference between the *_network objects and *_model objects.

- The *_network objects refer to the helper classes which create and manage the Keras models, load and save weights and set whether the model can be trained or not.

- The *_models objects refer to the underlying Keras model.

Note: The training images need to be stored in a subdirectory. Assume the path to the images is `/path-to-dir/path-to-sub-dir/*.png`, then simply write the path as `coco_path = /path-to-dir`. If this does not work, try `coco_path = /path-to-dir/` with a trailing slash (/)

To just create the pretrain model:

```
srgan_network = SRGANNetwork(img_width=32, img_height=32, batch_size=1)
srgan_model = srgan_network.build_srgan_pretrain_model()

# Plot the model
from keras.utils.visualize_util import plot
plot(srgan_model, to_file='SRGAN.png', show_shapes=True)
```

To pretrain the SR network:

```
srgan_network = SRGANNetwork(img_width=32, img_height=32, batch_size=1)
srgan_network.pre_train_srgan(iamges_path, nb_epochs=1, nb_images=50000)
```

** NOTE **: There may be many cases where generator initializations may lead to completely solid validation images. Please check the first few iterations to see if the validation images are not solid images.

To counteract this, a pretrained generator model has been provided, from which you can restart training. Therefore the model can continue learning without hitting a bad initialization.

To pretrain the Discriminator network:

```
srgan_network = SRGANNetwork(img_width=32, img_height=32, batch_size=1)
srgan_network.pre_train_discriminator(iamges_path, nb_epochs=1, nb_images=50000, batchsize=16)
```

To train the full network (Does NOT work properly right now, Discriminator is not correctly trained):

```
srgan_network = SRGANNetwork(img_width=32, img_height=32, batch_size=1)
srgan_network.train_full_model(coco_path, nb_images=80000, nb_epochs=10)
```

Benchmarks

Currently supports validation agains Set5, Set14 and BSD 100 dataset images. To download the images, each of the 3 dataset have scripts called download_*.py which must be run before running benchmark_test.py test.

Current Scores (Due to RGB grid and Blurred restoration):

SR ResNet:

- Set5 : Average PSNR of Set5 validation images : 22.1211430348
- Set14 : Average PSNR of Set5 validation images : 20.3971611357
- BSD100 : Average PSNR of BSD100 validation images : 20.9544390316

Drawbacks:

- Since keras has internal checks for batch size, we have to bypass an internal keras check called check_array_length(), which checks the input and output batch sizes. As we provide the original images to Input 2, batch size doubles. This causes an assertion error in internal keras code. For now, we rewrite the fit logic of keras in keras_training_ops.py and use the bypass fit functions.
- For some reason, the Deconvolution networks are not learning the upscaling function properly. This causes grids to form throughout the upscaled image. This is possibly due to the large (4x) upscaling procedure, but the Twitter team was able to do it.

Plans

The codebase is currently very chaotic, since I am focusing on correct implementation before making the project better. Therefore, expect the code to drastically change over commits.

Some things I am currently trying out:

- [x] Training the discriminator model separately properly.
- [x] Training the discriminator using soft labels and adversarial loss.
- [x] Properly train SRGAN (SR ResNet + VGG + Discriminator) model.
- [x] Fix the pixel grid formation when upscaling the image. (With Nearest Neighbour Upscaling).
- [x] Replacing the 2 deconv layers for a nearest neighbour upsampling layers.
- [ ] Improve docs & instructions

Discussion

There is an ongoing discussion at https://github.com/fchollet/keras/issues/3940 where I detail some of the outputs and attempts to correct the errors.

Requirements

- Theano (master branch)
- Keras 1.2.0 +