Environment

Env1

OS: Ubuntu 18.04

GPU: GTX2080Ti

Nvidia-driver: 384.111

Cuda: 9.0.176

Cudnn: 7.3.1

Python: 3.6.6

Pip: 19.0.3

Env2

OS: Ubuntu 18.04

GPU: GTX2080Ti

Nvidia-driver: 418.39

Cuda: 10.1.243

Cudnn: 7.6.5

Python: 3.6.9

Pip: 19.0.3

The requirements for Env1 and Env2 are shown in pip_list and pip_list2, which are derived by `pip freeze` in our well-done environment. You can use `pip install --upgrade -r pip_list` or `pip install --upgrade -r pip_list2` to install these requirements. Most of packages can be easily installed by pip automaticall, except for few third party packages that are not open-source.

The `dataset_path` in `ood_regularizer/experiment/datasetes/overall.py` should be specified as the absolute path of the datasets provided in supplemental material.

Scripts

The experiments in our paper can be reproduced by following script. The default hyper-parameters are shown in each script. You can use `--in_dataset={} --out_dataset={}` at the end of command to specify any dataset pair in datasets. For example, `--in_dataset=mnist28 --out_dataset=fashion_mnist28`. The name of dataset is same as the directory name in `dataset_path`.

Before the expeirments, you should go into the `code` directory, and then use following command.

```
export PYTHONPAYH=$PYTHONPATH:`pwd`
cd ood_regularizer/experiment
```

Use `python` to starting following experiments:

In Env1

Model VAE

```
python models/likelihood/vae.py --self_ood=True
```

The experiments for Recon, ELBO, ELBO - Recon, MCMC Recon, MCMC $\log p_\theta(x)$, $\log p_\theta(x)$, $T_perm(x)$, $\|\nabla_x \log p_\theta(x)\|$, $LLR(x)$ and $S(x)$.

```
python models/likelihood/vae.py --use_transductive=False
```

The experiments for $\log p_\theta(x) - \log p_\omega(x)$.

```
python models/likelihood/vae.py --use_transductive=False --mixed_ratio=0.2
```

The experiments for $\log p_\theta(x) - \log p_\omega(x)$ when only 20% data in out-of-distribubtion can be used for training.

```
python models/likelihood/vae.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$.

```
python models/likelihood/vae.py  --pretrain=True
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when γ is pretrained.

```
python models/singleshot/vae.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ where γ is trained by single-shot fine-tune.

```
python models/likelihood/vae.py --mixed_ratio=0.2
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when only 20% data in mixture distribution can be used for training.

```
python models/increment/vae.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is streaming with size 4096.

```
python models/increment/vae.py --mixed_train_skip=64
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is streaming with 64 data is coming at the same time.

```
python models/increment/vae.py --retrain_for_batch=True
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is split to several blocks with size 4096.

```
python models/ensemble/vae.py
```

The experiments for $WAIC(x)$ and $Var_\theta[\log p_\theta(x)]$.

```
python models/conditional/vae.py
```

The experiments for $\log p(x|y)$.

```
python models/batch_norm/vae.py
```

The experiments for $T_{b,r_1,r_2}(x)$ and $\log p_\theta(x)$ with BN.

```
python models/likelihood/vae_pretrain_diagram.py --pretrain=True
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when γ is pretrained. This experiment will print the performance after each epoch.

```
python models/likelihood/vae_pretrain_diagram.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when γ is unpretrained. This experiment will print the performance after each epoch.

Model PixelCNN

```
python models/likelihood/pixelcnn.py --self_ood=True
```

$\log p_\theta(x)$, $T_perm(x)$, $LLR(x)$ and $S(x)$.

```
python models/likelihood/pixelcnn.py --use_transductive=False
```

The experiments for $\log p_\theta(x) - \log p_\omega(x)$.

```
python models/likelihood/pixelcnn.py --use_transductive=False --
mixed_ratio=0.2
```

The experiments for $\log p_\theta(x) - \log p_\omega(x)$ when only 20% data in out-of-distribubtion can be used for training.

```
python models/likelihood/pixelcnn.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$

```
python models/likelihood/pixelcnn.py --pretrain=True
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when γ is pretrained.

```
python models/singleshot/pixelcnn.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ where γ is trained by single-shot fine-tune.

```
python models/likelihood/pixelcnn.py --mixed_ratio=0.2
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when only 20% data in mixture distribution can be used for training.

```
python models/increment/pixelcnn.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is streaming with size 4096.

```
python models/increment/pixelcnn.py --mixed_train_skip=64
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is streaming with 64 data is coming at the same time.

```
python models/increment/pixelcnn.py --retrain_for_batch=True
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is split to several blocks with size 4096.

```
python models/ensemble/pixelcnn.py
```

The experiments for $WAIC(x)$ and $Var_\theta[\log p_\theta(x)]$.

```
python models/conditional/pixelcnn.py
```

The experiments for $\log p(x|y)$.

```
python models/batch_norm/pixelcnn.py
```

The experiments for $T_{b,r_1,r_2}(x)$ and $\log p_\theta(x)$ with BN.

Model WGAN

```
python models/wgan/wasserstein.py --use_transductive=False
```

The experiments for $D_\omega(x)$.

```
python models/wgan/wasserstein.py --use_transductive=False --mixed_ratio=0.2
```

The experiments for $D_\omega(x)$ when only 20% data in out-of-distribution can be used for training.

```
python models/wgan/wasserstein.py
```

The experiments for $D_\gamma(x)$, $\|\nabla_x D_\theta(x)\|$.

```
python models/wgan/wasserstein.py --pretrain=True
```

The experiments for $D_\gamma(x)$ when γ is pretrained.

```
python models/wgan/wasserstein.py --mixed_ratio=0.2
```

The experiments for $D_\omega(x)$ when only 20% data in mixture distribution can be used for training.

```
python models/increment/wasserstein.py
```

The experiments for $D_\gamma(x)$ when data is streaming with size 4096.

```
python models/increment/wasserstein.py --mixed_train_skip=64
```

The experiments for $D_\gamma(x)$ when data is streaming with 64 data is coming at the same time.

```
python models/increment/wasserstein.py --retrain_for_batch=True
```

The experiments for $D_\gamma(x)$ when data is split to several blocks with size 4096.

Other models

```
python models/likelihood/vib.py
```

The experiments for H, R in VIB.

```
python models/conditional/generalized_odin.py
```

The experiments for $DeConf-C, DeConf-C*$ in generalized odin.

```
python models/conditional/pure_classifier.py --use_transductive=False
```

The experiments for Perfect classifier.

In Env2

Model RealNVP

```
python models/likelihood/glow.py --self_ood=True
```

$\log p_\theta(x)$, $T_perm(x)$, $LLR(x)$, Volume, $p_\theta(z)$ and $S(x)$.

```
python models/likelihood/glow.py --use_transductive=False
```

The experiments for $\log p_\theta(x) - \log p_\omega(x)$.

```
python models/likelihood/glow.py --use_transductive=False --mixed_ratio=0.2
```

The experiments for $\log p_\theta(x) - \log p_\omega(x)$ when only 20% data in out-of-distribubtion can be used for training.

```
python models/singleshot/glow.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ where γ is trained by single-shot fine-tune.

```
python models/likelihood/glow.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$.

```
python models/likelihood/glow.py --pretrain=True
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when γ is pretrained.

```
python models/likelihood/glow.py --mixed_ratio=0.2
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when only 20% data in mixture distribubtion can be used for training.

```
python models/increment/glow.py
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is streaming with size 4096.

```
python models/increment/glow.py --mixed_train_skip=64
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is streaming with 64 data is coming at the same time.

```
python models/increment/glow.py --retrain_for_batch=True
```

The experiments for $\log p_\theta(x) - \log p_\gamma(x)$ when data is split to several blocks with size 4096.

```
python models/ensemble/glow.py
```

The experiments for $WAIC(x)$ and $Var_\theta[\log p_\theta(x)]$.

```
python models/conditional/glow.py
```

The experiments for $\log p(x|y)$.

```
python models/batch_norm/glow.py
```

The experiments for $T_{b,r_1,r_2}(x)$ and $\log p_\theta(x)$ with BN.

Other models

```
python models/conditional/odin.py
```

The experiments for ODIN, Mahalanobis, Entropy of $p(y|x)$.

```
python models/ensemble/classifier.py
```

The experiments for Disagreement.