

**Findings**

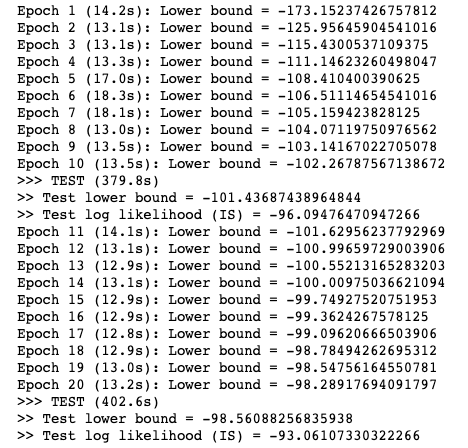
I have found that after comparing 10 and 20 epochs. 20 was clearer on the edges and more understandable.

Here are the comparison photos:

10 20

output of test lower bound and Log Likelihood:



**Appendix Code**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import print\_function

from \_\_future\_\_ import division

import os

import time

import tensorflow as tf

from six.moves import range

import numpy as np

import zhusuan as zs

from examples import conf

from examples.utils import dataset, save\_image\_collections

@zs.meta\_bayesian\_net(scope="gen", reuse\_variables=True)

def build\_gen(x\_dim, z\_dim, n, n\_particles=1):

bn = zs.BayesianNet()

z\_mean = tf.zeros([n, z\_dim])

z = bn.normal("z", z\_mean, std=1., group\_ndims=1, n\_samples=n\_particles)

h = tf.layers.dense(z, 500, activation=tf.nn.relu)

h = tf.layers.dense(h, 500, activation=tf.nn.relu)

x\_logits = tf.layers.dense(h, x\_dim)

bn.deterministic("x\_mean", tf.sigmoid(x\_logits))

bn.bernoulli("x", x\_logits, group\_ndims=1)

return bn

@zs.reuse\_variables(scope="q\_net")

def build\_q\_net(x, z\_dim, n\_z\_per\_x):

bn = zs.BayesianNet()

h = tf.layers.dense(tf.cast(x, tf.float32), 500, activation=tf.nn.relu)

h = tf.layers.dense(h, 500, activation=tf.nn.relu)

z\_mean = tf.layers.dense(h, z\_dim)

z\_logstd = tf.layers.dense(h, z\_dim)

bn.normal("z", z\_mean, logstd=z\_logstd, group\_ndims=1, n\_samples=n\_z\_per\_x)

return bn

def main():

# Load MNIST

data\_path = os.path.join(conf.data\_dir, "mnist.pkl.gz")

x\_train, t\_train, x\_valid, t\_valid, x\_test, t\_test = \

dataset.load\_mnist\_realval(data\_path)

x\_train = np.vstack([x\_train, x\_valid])

x\_test = np.random.binomial(1, x\_test, size=x\_test.shape)

x\_dim = x\_train.shape[1]

# Define model parameters

z\_dim = 40

# Build the computation graph

n\_particles = tf.placeholder(tf.int32, shape=[], name="n\_particles")

x\_input = tf.placeholder(tf.float32, shape=[None, x\_dim], name="x")

x = tf.cast(tf.less(tf.random\_uniform(tf.shape(x\_input)), x\_input),

tf.int32)

n = tf.placeholder(tf.int32, shape=[], name="n")

model = build\_gen(x\_dim, z\_dim, n, n\_particles)

variational = build\_q\_net(x, z\_dim, n\_particles)

lower\_bound = zs.variational.elbo(

model, {"x": x}, variational=variational, axis=0)

cost = tf.reduce\_mean(lower\_bound.sgvb())

lower\_bound = tf.reduce\_mean(lower\_bound)

# # Importance sampling estimates of marginal log likelihood

is\_log\_likelihood = tf.reduce\_mean(

zs.is\_loglikelihood(model, {"x": x}, proposal=variational, axis=0))

optimizer = tf.train.AdamOptimizer(learning\_rate=0.001)

infer\_op = optimizer.minimize(cost)

# Random generation

x\_gen = tf.reshape(model.observe()["x\_mean"], [-1, 28, 28, 1])

# Define training/evaluation parameters

epochs = 20

batch\_size = 128

iters = x\_train.shape[0] // batch\_size

save\_freq = 10

test\_freq = 10

test\_batch\_size = 400

test\_iters = x\_test.shape[0] // test\_batch\_size

result\_path = "results/vae"

# Run the inference

with tf.Session() as sess:

sess.run(tf.global\_variables\_initializer())

for epoch in range(1, epochs + 1):

time\_epoch = -time.time()

np.random.shuffle(x\_train)

lbs = []

for t in range(iters):

x\_batch = x\_train[t \* batch\_size:(t + 1) \* batch\_size]

\_, lb = sess.run([infer\_op, lower\_bound],

feed\_dict={x\_input: x\_batch,

n\_particles: 1,

n: batch\_size})

lbs.append(lb)

time\_epoch += time.time()

print("Epoch {} ({:.1f}s): Lower bound = {}".format(

epoch, time\_epoch, np.mean(lbs)))

if epoch % test\_freq == 0:

time\_test = -time.time()

test\_lbs, test\_lls = [], []

for t in range(test\_iters):

test\_x\_batch = x\_test[t \* test\_batch\_size:

(t + 1) \* test\_batch\_size]

test\_lb = sess.run(lower\_bound,

feed\_dict={x: test\_x\_batch,

n\_particles: 1,

n: test\_batch\_size})

test\_ll = sess.run(is\_log\_likelihood,

feed\_dict={x: test\_x\_batch,

n\_particles: 1000,

n: test\_batch\_size})

test\_lbs.append(test\_lb)

test\_lls.append(test\_ll)

time\_test += time.time()

print(">>> TEST ({:.1f}s)".format(time\_test))

print(">> Test lower bound = {}".format(np.mean(test\_lbs)))

print('>> Test log likelihood (IS) = {}'.format(

np.mean(test\_lls)))

if epoch % save\_freq == 0:

images = sess.run(x\_gen, feed\_dict={n: 100, n\_particles: 1})

name = os.path.join(result\_path,

"vae.epoch.{}.png".format(epoch))

save\_image\_collections(images, name)

if \_\_name\_\_ == "\_\_main\_\_":

main()