```python
# univariate normal
Normal(loc=0.0, scale=1.0)
# vector of 5 univariate normals
Normal(loc=tf.zeros(5), scale=tf.ones(5))
# 2 x 3 matrix of Exponentials
Exponential(rate=tf.ones([2, 3]))
```

Figure 1

```python
theta = Beta(1.0, 1.0)
x = Bernoulli(probs=tf.ones(50) * theta)
```

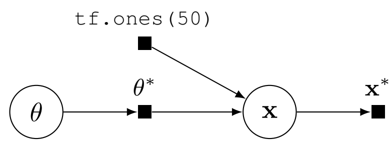


Figure 2

```
X = tf.placeholder(tf.float32, [N, D])
f = MultivariateNormalTriL(loc=tf.zeros(N),
                           scale_tril=tf.cholesky(rbf(X)))
y = Bernoulli(logits=f)
```

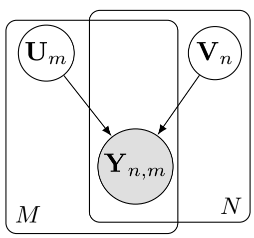
Figure 3

```
X = Normal(loc=tf.zeros([N, Q]), scale=tf.ones([N, Q]))
f = MultivariateNormalTriL(loc=tf.zeros([N, D]),
                           scale_tril=tf.cholesky(rbf(X)))
y = Bernoulli(logits=f)
```

Figure 4

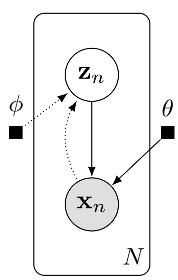
```
X = Normal(loc=tf.zeros([N, Q]), scale=tf.ones([N, Q]))
h1 = MultivariateNormalTriL(loc=tf.zeros([N, H1]),
                            scale_tril=tf.cholesky(rbf(X)))
h2 = MultivariateNormalTriL(loc=tf.zeros([N, H2]),
                            scale_tril=tf.cholesky(rbf(h1)))
f = MultivariateNormalTriL(loc=tf.zeros([N, D]),
                           scale_tril=tf.cholesky(rbf(h2)))
y = Bernoulli(logits=f)
```

Figure 5




```
N = 10
M = 10
K = 5  # latent dimension

U = Normal(loc=tf.zeros([M, K]), sigma=tf.ones([M, K]))
V = Normal(loc=tf.zeros([N, K]), sigma=tf.ones([N, K]))
Y = Normal(loc=tf.matmul(U, V, transpose_b=True), sigma=tf.ones([N, M]))
```

Figure 6




```
# Probabilistic model
z = Normal(loc=tf.zeros([N, d]), scale=tf.ones([N, d]))
h = Dense(256, activation='relu')(z)
x = Bernoulli(logits=Dense(28 * 28, activation=None)(h))

# Variational model
qx = tf.placeholder(tf.float32, [N, 28 * 28])
qh = Dense(256, activation='relu')(qx)
qz = Normal(loc=Dense(d, activation=None)(qh),
            scale=Dense(d, activation='softplus')(qh))
```

Figure 7

Inference method	Neg. log-likelihood
VAE [Kingma & Welling 2014]	≤ 88.2
VAE without analytic KL	≤ 89.4
VAE with analytic entropy	≤ 88.1
VAE with score function gradient	≤ 87.9
Normalizing flows [Rezende & Mohamed 2015]	≤ 85.8
Hierarchical variational model [Ranganath+ 2016]	≤ 85.4
Importance-weighted auto-encoders ($K = 50$) [Burda+ 2016]	≤ 86.3
HVM with IWAE objective ($K = 5$)	≤ 85.2
Rényi divergence ($\alpha = -1$) [Li & Turner 2016]	≤ 140.5

Table 1

```
N = 1000  # number of data points
d = 10  # latent dimensionality

# DATA
x_data = np.loadtxt('mnist.txt', np.float32)

# MODEL
z = Normal(loc=tf.zeros([N, d]), scale=tf.ones([N, d]))
h = Dense(256, activation='relu')(z)
x = Bernoulli(logits=Dense(28 * 28, activation=None)(h))

# INFERENCE
qx = tf.placeholder(tf.float32, [N, 28 * 28])
qh = Dense(256, activation='relu')(qx)
qz = Normal(loc=Dense(d, activation=None)(qh),
            scale=Dense(d, activation='softplus')(qh))

inference = ed.KLqp({z: qz}, data={x: x_data, qx: x_data})
inference.run()
```

```
N = 1000  # number of data points
d = 10  # latent dimensionality

# DATA
x_data = np.loadtxt('mnist.txt', np.float32)

# MODEL
z = Normal(loc=tf.zeros([N, d]), scale=tf.ones([N, d]))
h = Dense(256, activation='relu')(z)
x = Bernoulli(logits=Dense(28 * 28, activation=None)(h))

# INFERENCE
T = 10000  # number of samples
qz = Empirical(params=tf.Variable(tf.zeros([T, N, d])))

inference = ed.HMC({z: qz}, data={x: x_data})
inference.run()
```

Figure 8

```
inference = ed.Inference({beta: qbeta, z: qz}, data={x: x_train})
```

```
qbeta = Normal(loc=tf.Variable(tf.zeros([K, D])),
               scale=tf.exp(tf.Variable(tf.zeros([K, D]))))
qz = Categorical(logits=tf.Variable(tf.zeros([N, K])))

inference = ed.VariationalInference({beta: qbeta, z: qz}, data={x: x_train})
```

```
T = 10000  # number of samples
qbeta = Empirical(params=tf.Variable(tf.zeros([T, K, D])))
qz = Empirical(params=tf.Variable(tf.zeros([T, N])))

inference = ed.MonteCarlo({beta: qbeta, z: qz}, data={x: x_train})
```

Figure 9

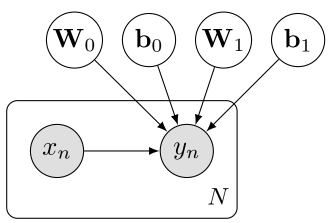
Probabilistic programming system	Runtime (s)
Handwritten NumPy (1 CPU)	534
Stan (1 CPU) [Carpenter+ 2016]	171
PyMC3 (12 CPU) [Salvatier+ 2015]	30.0
Edward (12 CPU)	**8.2**
Handwritten TensorFlow (GPU)	5.0
Edward (GPU)	**4.9** (35x faster than Stan)

Table 2

```python
def generative_network(eps):
  h = Dense(256, activation='relu')(eps)
  return Dense(28 * 28, activation=None)(h)

def discriminative_network(x):
  h = Dense(28 * 28, activation='relu')(x)
  return Dense(h, activation=None)(1)

# Probabilistic model
eps = Normal(loc=tf.zeros([N, d]), scale=tf.ones([N, d]))
x = generative_network(eps)

inference = ed.GANInference(data={x: x_train},
    discriminator=discriminative_network)
inference.run()
```

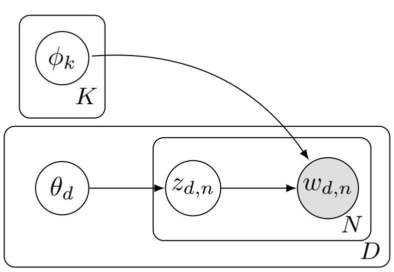
```python
def generative_network(eps):
  h = Dense(256, activation='relu')(eps)
  return Dense(28 * 28, activation=None)(h)

def discriminative_network(x):
  h = Dense(28 * 28, activation='relu')(x)
  return Dense(h, activation=None)(1)

# Probabilistic model
eps = Normal(loc=tf.zeros([N, d]), scale=tf.ones([N, d]))
x = generative_network(eps)

inference = ed.WGANInference(data={x: x_train},
    discriminator=discriminative_network)
inference.run()
```

Figure 10



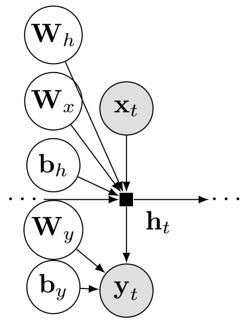

```python
W_0 = Normal(loc=tf.zeros([D, H]), sigma=tf.ones([D, H]))
W_1 = Normal(loc=tf.zeros([H, 1]), sigma=tf.ones([H, 1]))
b_0 = Normal(loc=tf.zeros(H), sigma=tf.ones(H))
b_1 = Normal(loc=tf.zeros(1), sigma=tf.ones(1))

x = tf.placeholder(tf.float32, [N, D])
y = Bernoulli(logits=tf.matmul(tf.nn.tanh(tf.matmul(x, W_0) + b_0), W_1) + b_1)
```

Figure 11




```python
D = 4  # number of documents
N = [11502, 213, 1523, 1351]  # words per doc
K = 10  # number of topics
V = 100000  # vocabulary size

theta = Dirichlet(alpha=tf.zeros([D, K]) + 0.1)
phi = Dirichlet(alpha=tf.zeros([K, V]) + 0.05)
z = [[0] * N] * D
w = [[0] * N] * D
for d in range(D):
  for n in range(N[d]):
    z[d][n] = Categorical(pi=theta[d, :])
    w[d][n] = Categorical(pi=phi[z[d][n], :])
```

Figure 12

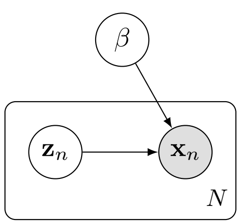



```
def rnn_cell(hprev, xt):
  return tf.tanh(tf.dot(hprev, Wh) + tf.dot(xt, Wx) + bh)

Wh = Normal(loc=tf.zeros([H, H]), scale=tf.ones([H, H]))
Wx = Normal(loc=tf.zeros([D, H]), scale=tf.ones([D, H]))
Wy = Normal(loc=tf.zeros([H, 1]), scale=tf.ones([H, 1]))
bh = Normal(loc=tf.zeros(H), scale=tf.ones(H))
by = Normal(loc=tf.zeros(1), scale=tf.ones(1))

x = tf.placeholder(tf.float32, [None, D])
h = tf.scan(rnn_cell, x, initializer=tf.zeros(H))
y = Normal(loc=tf.matmul(h, Wy) + by, scale=1.0)
```

Figure 13

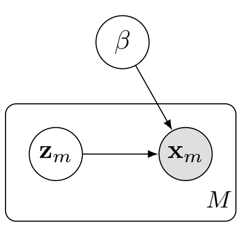
```
qbeta = PointMass(params=tf.Variable(tf.zeros([K, D])))
qz = Categorical(logits=tf.Variable(tf.zeros([N, K])))

inference_e = ed.VariationalInference({z: qz}, data={x: x_train, beta: qbeta})
inference_m = ed.MAP({beta: qbeta}, data={x: x_train, z: qz})
...
for _ in range(10000):
  inference_e.update()
  inference_m.update()
```

Figure 14




```
N = 10000  # number of data points
D = 2  # data dimension
K = 5  # number of clusters

beta = Normal(loc=tf.zeros([K, D]), scale=tf.ones([K, D]))
z = Categorical(logits=tf.zeros([N, K]))
x = Normal(loc=tf.gather(beta, z), scale=tf.ones([N, D]))
```

Figure 15




```
beta = Normal(loc=tf.zeros([K, D]), scale=tf.ones([K, D]))
z = Categorical(logits=tf.zeros([M, K]))
x = Normal(loc=tf.gather(beta, z), scale=tf.ones([M, D]))

qbeta = Normal(loc=tf.Variable(tf.zeros([K, D])),
               scale=tf.nn.softplus(tf.Variable(tf.zeros([K, D]))))
qz = Categorical(logits=tf.Variable(tf.zeros([M, D])))

inference = ed.VariationalInference({beta: qbeta, z: qz}, data={x: x_batch})
inference.initialize(scale={x: float(N)/M, z: float(N)/M})
```

Figure 16

```
loc = DirichletProcess(0.1, Normal(tf.zeros(D), tf.ones(D)), sample_shape=N)
x = Normal(loc=loc, scale=tf.ones([N, D]))
```

```
def dirichlet_process(alpha):
  def cond(k, beta_k):
    flip = Bernoulli(p=beta_k)
    return tf.equal(flip, tf.constant(1))

  def body(k, beta_k):
    beta_k = beta_k * Beta(a=1.0, b=alpha)
    return k + 1, beta_k

  k = tf.constant(0)
  beta_k = Beta(a=1.0, b=alpha)
  stick_num, stick_beta = tf.while_loop(cond, body, loop_vars=[k, beta_k])
  return stick_num
```

Figure 17