In [17]:
```
import keras
from keras import layers
from tensorflow.keras import backend as K
from keras.models import Model
from keras.datasets import mnist
import matplotlib.pyplot as plt
from scipy.stats import norm
import numpy as np


img_shape = (28, 28, 1)
batch_size = 16
latent_dim = 2

input_img = keras.Input(shape = img_shape)
x = layers.Conv2D(32, 3, padding = 'same', activation = 'relu')(input_img)
x = layers.Conv2D(64, 3, padding = 'same', activation = 'relu', strides = (2, 2))(x)
x = layers.Conv2D(64, 3, padding = 'same', activation = 'relu')(x)
x = layers.Conv2D(32, 3, padding = 'same', activation = 'relu')(x)
shape_before_flattening = K.int_shape(x)

x = layers.Flatten()(x)
x = layers.Dense(32, activation = 'relu')(x)

z_mean = layers.Dense(latent_dim)(x)
z_log_var = layers.Dense(latent_dim)(x)
```

In [18]:
```
def sampling(args):
    z_mean, z_log_var = args
    epsilon = K.random_normal(shape = (K.shape(z_mean)[0], latent_dim), mean = 0., stddev = 1.)
    return z_mean + K.exp(z_log_var) * epsilon
z = layers.Lambda(sampling)([z_mean, z_log_var])
```

In [19]:
```
decoder_input = layers.Input(K.int_shape(z)[1:])

x = layers.Dense(np.prod(shape_before_flattening[1:]), activation = 'relu')(decoder_input)

x = layers.Reshape(shape_before_flattening[1:])(x)

x = layers.Conv2DTranspose(32, 3, padding = 'same', activation = 'relu', strides = (2, 2))(x)

x = layers.Conv2D(1, 3, padding = 'same', activation = 'sigmoid')(x)

decoder = Model(decoder_input, x)

z_decoded = decoder(z)
```

In [20]:
```
class CustomVariationalLayer(keras.layers.Layer):

    def vae_loss(self, x, z_decoded):
        x = K.flatten(x)
        z_decoded = K.flatten(z_decoded)
        xent_loss = keras.metrics.binary_crossentropy(x, z_decoded)
        kl_loss = -5e-4 * K.mean(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis = -1)
        return K.mean(xent_loss + kl_loss)

    def call(self, inputs):
        x = inputs[0]
        z_decoded = inputs[1]
        loss = self.vae_loss(x, z_decoded)
        self.add_loss(loss, inputs = inputs)
        return x
y = CustomVariationalLayer()([input_img, z_decoded])
```

In [23]:

```
vae = Model(input_img, y)
vae.compile(optimizer = 'rmsprop', loss = None)
vae.summary()

(x_train, _), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255.
x_train = x_train.reshape(x_train.shape + (1,))
x_test = x_test.astype('float32') / 255
x_test = x_test.reshape(x_test.shape + (1,))

vae.fit(x = x_train, y = None, shuffle = True, epochs = 10, batch_size = batch_size, validation_data = (x_tes
t, None))
```

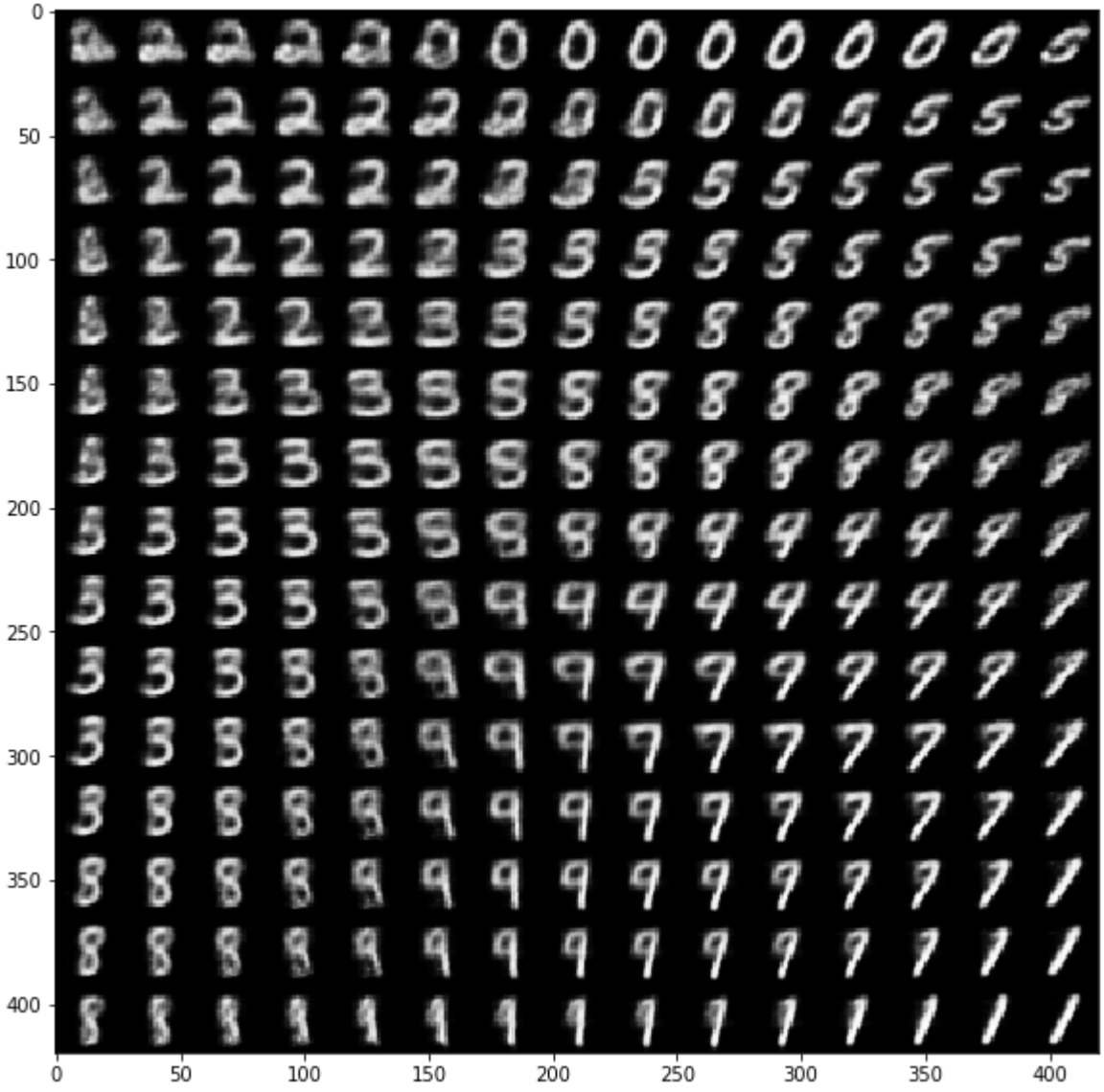
```
Model: "model_6"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_9 (InputLayer)            (None, 28, 28, 1)    0
__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 28, 28, 32)   320         input_9[0][0]
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 14, 14, 64)   18496       conv2d_15[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 14, 14, 64)   36928       conv2d_16[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 14, 14, 32)   18464       conv2d_17[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 6272)         0           conv2d_18[0][0]
__________________________________________________________________________________________________
dense_13 (Dense)                (None, 32)           200736      flatten_4[0][0]
__________________________________________________________________________________________________
dense_14 (Dense)                (None, 2)            66          dense_13[0][0]
__________________________________________________________________________________________________
dense_15 (Dense)                (None, 2)            66          dense_13[0][0]
__________________________________________________________________________________________________
lambda_4 (Lambda)               (None, 2)            0           dense_14[0][0]
                                                                 dense_15[0][0]
__________________________________________________________________________________________________
model_3 (Model)                 (None, 28, 28, 1)    28353       lambda_4[0][0]
__________________________________________________________________________________________________
custom_variational_layer_3 (Cus [(None, 28, 28, 1),  0           input_9[0][0]
                                                                 model_3[1][0]
==================================================================================================
Total params: 303,429
Trainable params: 303,429
Non-trainable params: 0
__________________________________________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 252s 4ms/step - loss: 0.2123 - val_loss: 0.1953
Epoch 2/10
60000/60000 [==============================] - 295s 5ms/step - loss: 0.1928 - val_loss: 0.1890
Epoch 3/10
60000/60000 [==============================] - 288s 5ms/step - loss: 0.1889 - val_loss: 0.1877
Epoch 4/10
60000/60000 [==============================] - 290s 5ms/step - loss: 0.1866 - val_loss: 0.1860
```

```
Epoch 5/10
60000/60000 [==============================] - 287s 5ms/step - loss: 0.1849 - val_loss: 0.1844
Epoch 6/10
60000/60000 [==============================] - 278s 5ms/step - loss: 0.1834 - val_loss: 0.1834
Epoch 7/10
60000/60000 [==============================] - 262s 4ms/step - loss: 0.1824 - val_loss: 0.1817
Epoch 8/10
60000/60000 [==============================] - 264s 4ms/step - loss: 0.1814 - val_loss: 0.1809
Epoch 9/10
60000/60000 [==============================] - 271s 5ms/step - loss: 0.1810 - val_loss: 0.1801
Epoch 10/10
60000/60000 [==============================] - 338s 6ms/step - loss: 0.1807 - val_loss: 0.1820
```

Out[23]: <keras.callbacks.callbacks.History at 0x11acf5a2080>

In [27]:

```
import os
from pathlib import Path

current_dir = Path(os.getcwd()).absolute()
results_dir = current_dir.joinpath('results')

results_dir.mkdir(parents=True, exist_ok=True)

vae_dir = results_dir.joinpath('vae')
vae_dir.mkdir(parents=True, exist_ok=True)



n = 15
digit_size = 28
figure = np.zeros((digit_size * n, digit_size * n))
grid_x = norm.ppf(np.linspace(0.05, 0.95, n))
grid_y = norm.ppf(np.linspace(0.05, 0.95, n))
for i, yi in enumerate(grid_x):
    for j, xi in enumerate(grid_y):
        z_sample = np.array([[xi, yi]])
        z_sample = np.tile(z_sample, batch_size).reshape(batch_size, 2)
        x_decoded = decoder.predict(z_sample, batch_size = batch_size)
        digit = x_decoded[0].reshape(digit_size, digit_size)
        figure[i * digit_size: (i + 1) * digit_size, j * digit_size: (j + 1) * digit_size] = digit
plt.figure(figsize = (10, 10))
plt.imshow(figure, cmap = 'Greys_r')
plt.savefig(vae_dir.joinpath('results.png'))
plt.show()
```



In [26]:

```
<Figure size 432x288 with 0 Axes>
```

In [ ]: