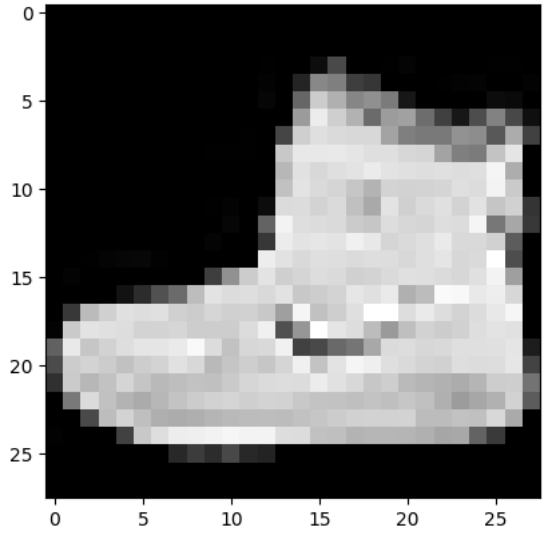
https://colab.research.google.com/drive/1UM1nNZkAJm7t_MmVNLJm9zYQBRevCPX-?usp=sharing

```
# импорт библиотек
import numpy as np
import matplotlib.pyplot as plt

from tensorflow import keras
from keras.datasets import fashion_mnist
import keras.backend as K
from keras.layers import Dense, Flatten, Reshape, Input, Lambda, BatchNormalization, Dropout

(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

x_train = x_train / 255
x_test = x_test / 255

x_train = np.reshape(x_train, (len(x_train), 28, 28, 1))
x_test  = np.reshape(x_test,  (len(x_test),  28, 28, 1))


import matplotlib.pyplot as plt

plt.imshow(x_train[0], cmap='gray')
plt.show()
```




```
x_train.shape
```

```
(60000, 28, 28, 1)
```

```
hidden_dim = 2
batch_size = 60


input_img = Input((28, 28, 1))
x = Flatten()(input_img)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
z_mean = Dense(hidden_dim)(x)
z_log_var = Dense(hidden_dim)(x)


def noiser(args):
  global z_mean, z_log_var
  z_mean, z_log_var = args
  N = K.random_normal(shape=(batch_size, hidden_dim), mean=0., stddev=1.0)
  return K.exp(z_log_var / 2) * N + z_mean


h = Lambda(noiser, output_shape=(hidden_dim,))([z_mean, z_log_var])


input_dec = Input(shape=(hidden_dim,))
d = Dense(128, activation='relu')(input_dec)
d = Dense(256, activation='relu')(input_dec)
```

```
d = Dense(28*28, activation='sigmoid')(d)
decoded = Reshape((28, 28, 1))(d)


encoder = keras.Model(input_img, h, name='encoder')
decoder = keras.Model(input_dec, decoded, name='decoder')
vae = keras.Model(input_img, decoder(encoder(input_img)), name="vae")


def vae_loss(x, y):
  x = K.reshape(x, shape=(batch_size, 28*28))
  y = K.reshape(y, shape=(batch_size, 28*28))
  loss = K.sum(K.square(x-y), axis=-1)
  kl_loss = -0.5 * K.sum(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
  return loss + kl_loss


vae.compile(optimizer='adam', loss=vae_loss)



import random
random.shuffle(x_train)
x_validation = x_train[54000:]
x_train = x_train[:54000]


vae.summary()
```

```
Model: "vae"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_15 (InputLayer)       [(None, 28, 28, 1)]       0

 encoder (Functional)        (60, 2)                   234372

 decoder (Functional)        (None, 28, 28, 1)         202256

=================================================================
Total params: 436,628
Trainable params: 436,628
Non-trainable params: 0
_________________________________________________________________
```

```
history = vae.fit(x_train, x_train, epochs=5, batch_size=batch_size, validation_data=(x_validation, x_validation))
```

```
Epoch 1/5
900/900 [==============================] - 8s 8ms/step - loss: 40.9119 - val_loss: 34.0365
Epoch 2/5
900/900 [==============================] - 10s 11ms/step - loss: 32.9120 - val_loss: 33.0248
Epoch 3/5
900/900 [==============================] - 6s 7ms/step - loss: 31.7586 - val_loss: 32.0336
Epoch 4/5
900/900 [==============================] - 7s 8ms/step - loss: 30.9753 - val_loss: 31.1254
Epoch 5/5
900/900 [==============================] - 7s 7ms/step - loss: 30.4291 - val_loss: 30.7706
```

```
history.history
```

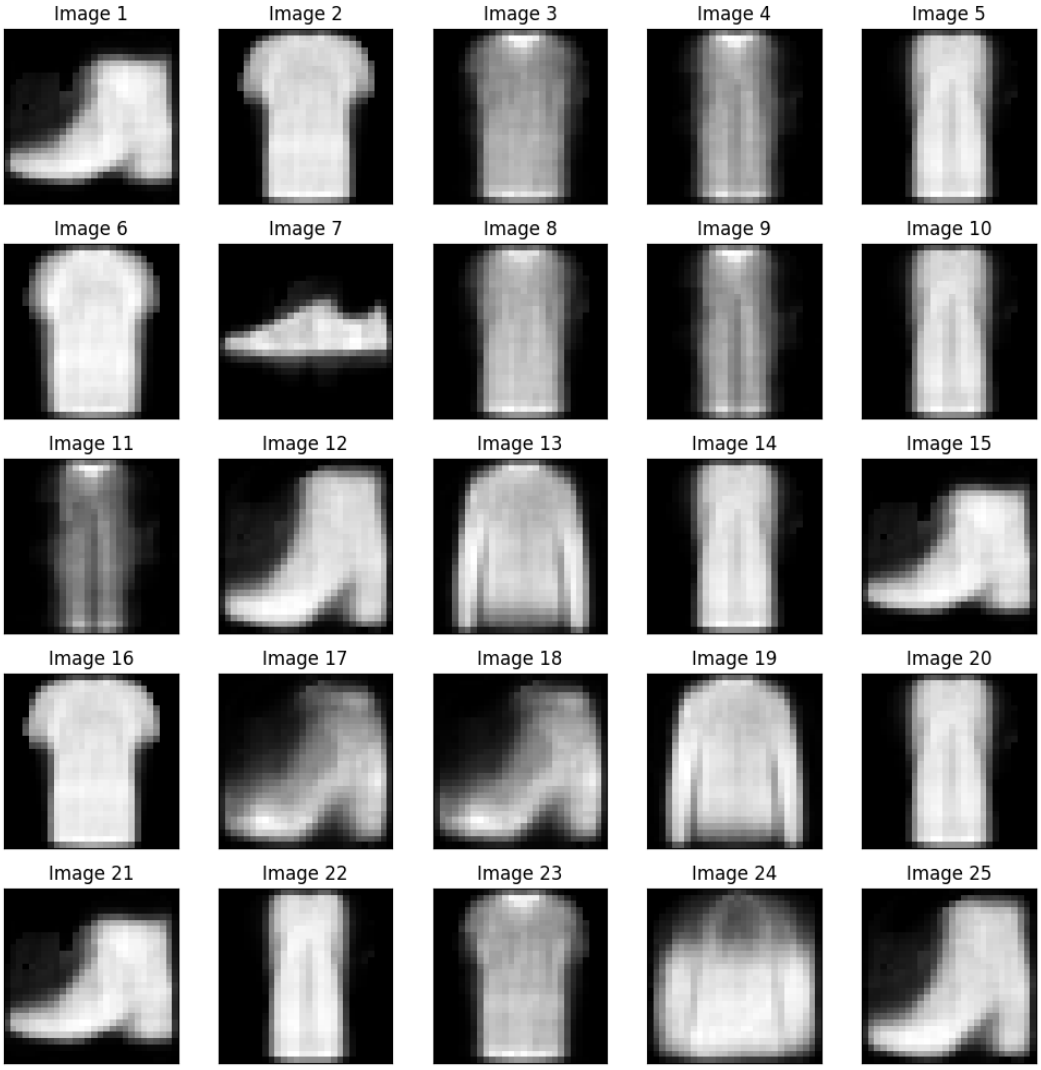
```
{'loss': [40.91185760498047,
  32.91195297241211,
  31.758567810058594,
  30.97525405883789,
  30.429147720336914],
 'val_loss': [34.03645706176758,
  33.024810791015625,
  32.03363037109375,
  31.12542152404785,
  30.770627975463867]}
```

```
from re import VERBOSE
def pred(test):
  h = encoder.predict(test[:60], batch_size=batch_size, verbose=0)
  pred = decoder.predict(h, verbose=0)
  fig, axes = plt.subplots(nrows=5, ncols=5, figsize=(10,10))
  image_shape = (28, 28, 1)
  for i, ax in enumerate(axes.flat):
      ax.imshow(pred[i].reshape(image_shape), cmap='gray')
      ax.set_xticks([])
      ax.set_yticks([])
      ax.set_title(f"Image {i+1}")
  plt.tight_layout()
  plt.show()
```

```
test_loss = vae.evaluate(x_test[:6000], x_test[:6000], batch_size=batch_size)
print("Test Loss:", test_loss)
```

```
100/100 [==============================] - 0s 2ms/step - loss: 30.3793
Test Loss: 30.37934112548828
```

```
pred(x_train)
```




```
pred(x_validation)
```

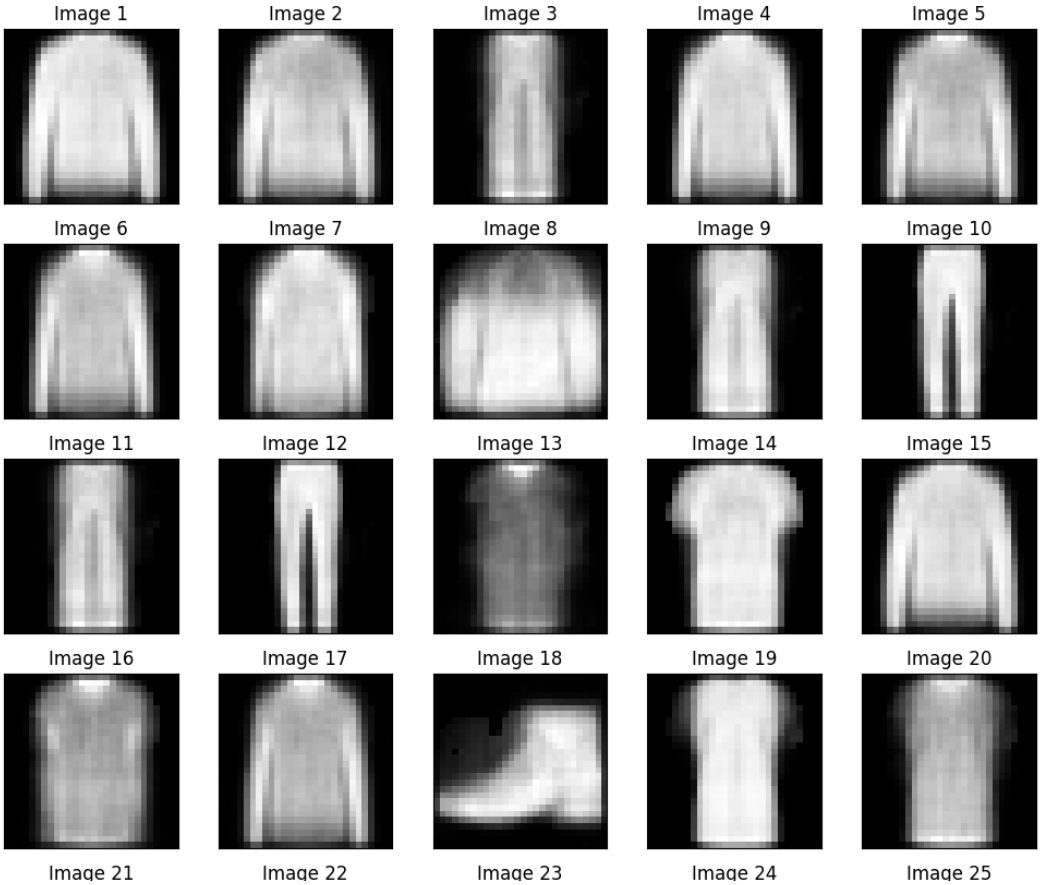
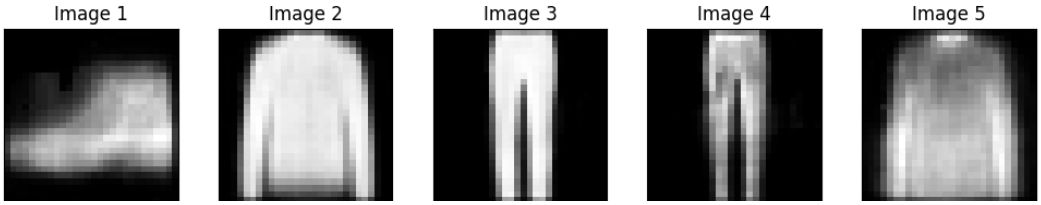


Image 1
Image 2
Image 3
Image 4
Image 5
Image 6
Image 7
Image 8
Image 9
Image 10
Image 11
Image 12
Image 13
Image 14
Image 15
Image 16
Image 17
Image 18
Image 19
Image 20
Image 21
Image 22
Image 23
Image 24
Image 25

```
pred(x_test)
```

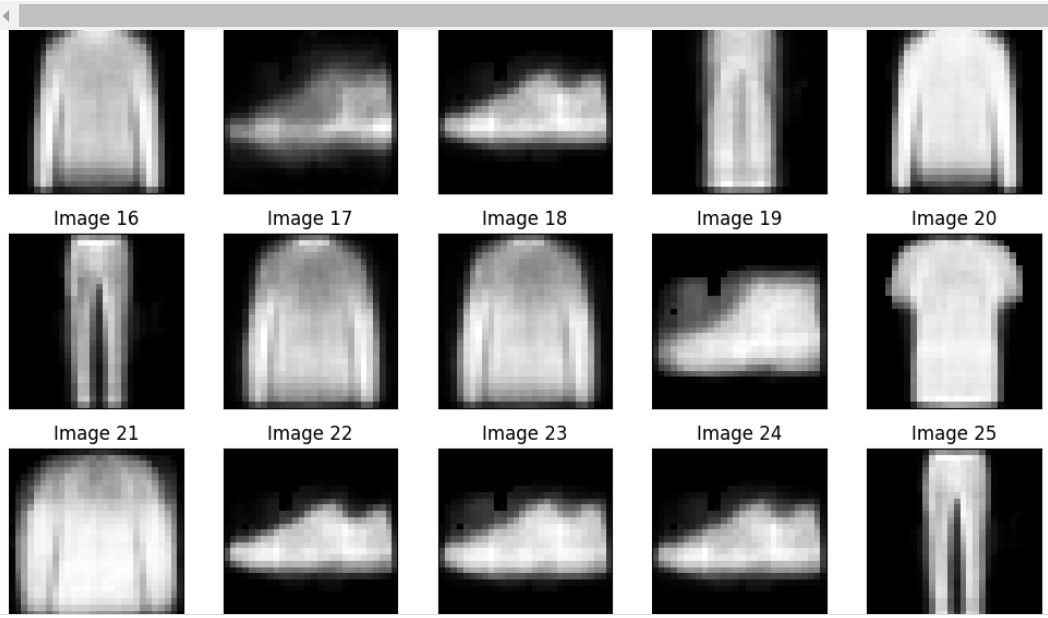



```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
vae.save('/content/drive/MyDrive/my_model/vae')
```

```
WARNING:absl:Found untraced functions such as _update_step_xla while saving (showing 1 of 1). These functions will not be directly
```



✓ 1 сек. выполнено в 19:08