Compression on MNIST dataset | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=Omd8LwiP4czT

Gmail Maps News Translate Disney+ Hotstar -... YouTube All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More resources
    - Featured examples
- + Section

[2] ✓ 3m

```
import tensorflow as tf
from tensorflow.keras import layers, Model
import numpy as np
import matplotlib.pyplot as plt

# ----------------------------
# Load and preprocess MNIST
# ----------------------------
(x_train, _), (x_test, _) = tf.keras.datasets.mnist.load_data()
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

x_train = x_train.reshape(-1, 784)
x_test = x_test.reshape(-1, 784)

latent_dim = 2  # Size of latent space

# ----------------------------
# Encoder
# ----------------------------
inputs = layers.Input(shape=(784,))
h = layers.Dense(256, activation='relu')(inputs)
h = layers.Dense(128, activation='relu')(h)
z_mean = layers.Dense(latent_dim)(h)
z_log_var = layers.Dense(latent_dim)(h)

# Sampling layer (reparameterization trick)
def sampling(args):
    z_mean, z_log_var = args
    epsilon = tf.random.normal(shape=(tf.shape(z_mean)[0], latent_dim))
```

Variables Terminal ✓ 00:31 Python 3

Compression on MNIST dataset | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=Omd8LwiP4czT

Gmail Maps News Translate Disney+ Hotstar -... YouTube All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all Copy to Drive

RAM Disk

Table of contents

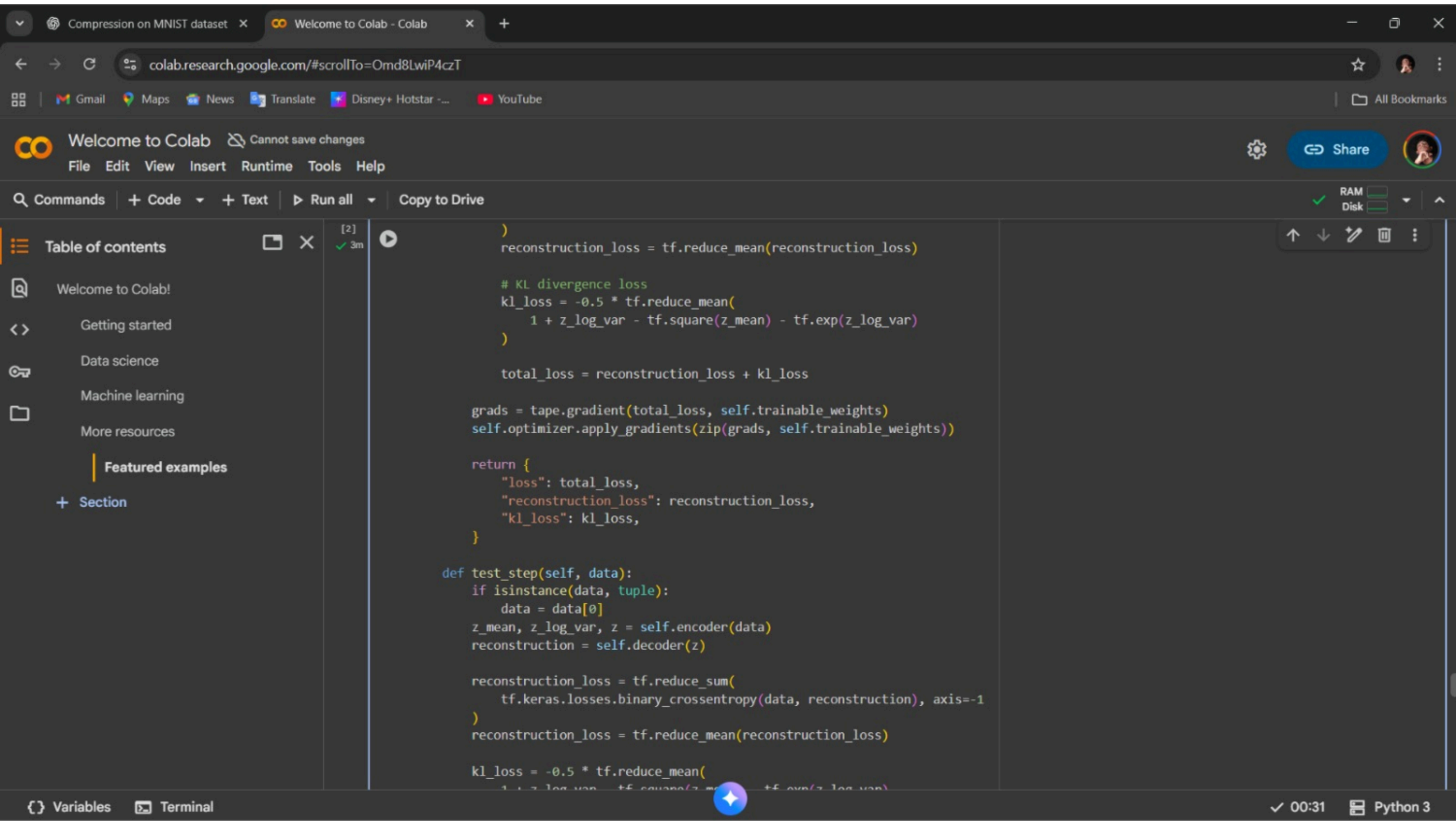
- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More resources
- Featured examples
- + Section

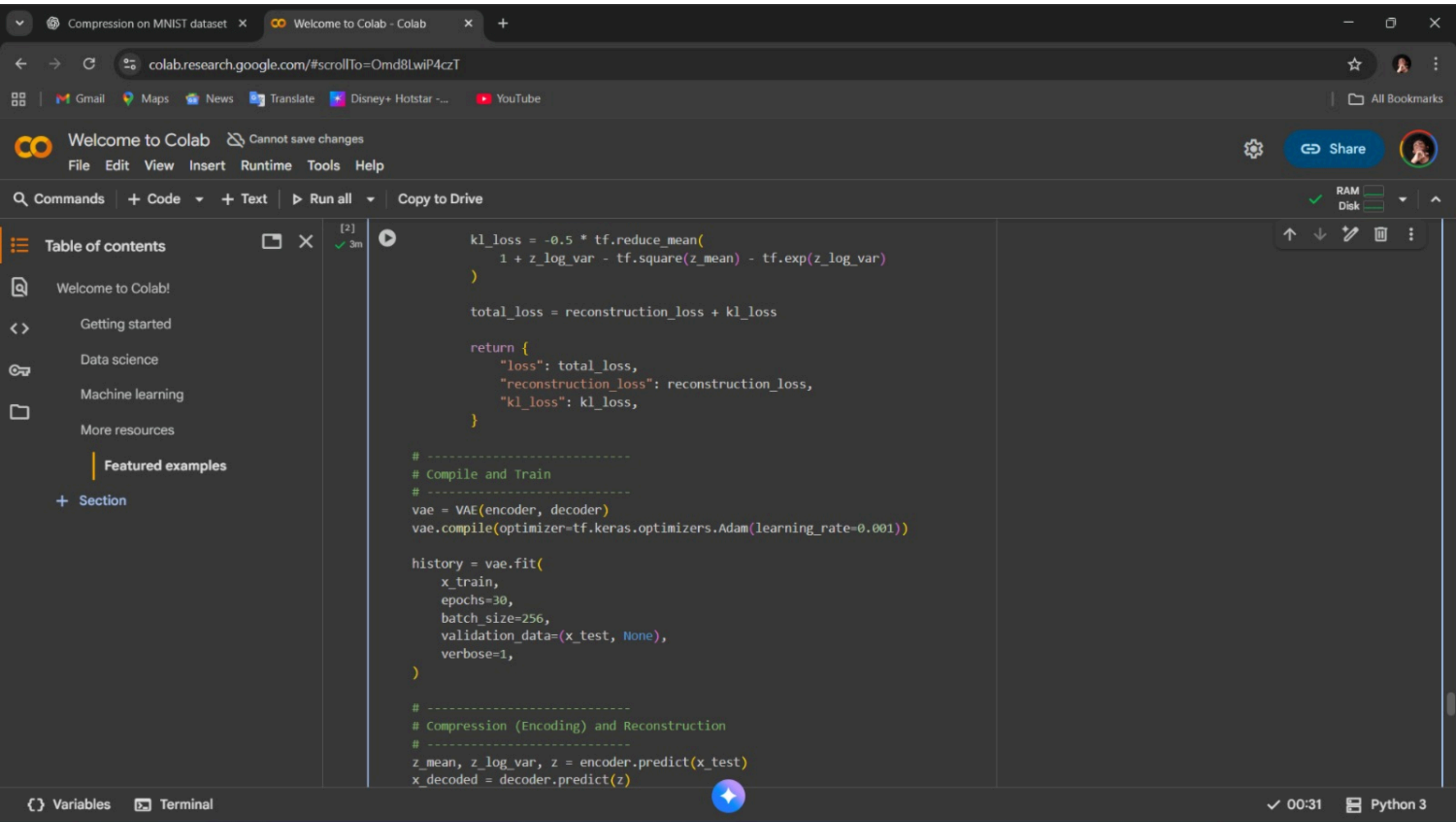
[2] 3m

```
z = layers.Lambda(sampling)([z_mean, z_log_var])
encoder = Model(inputs, [z_mean, z_log_var, z], name="encoder")

# ---------------------------
# Decoder
# ---------------------------
latent_inputs = layers.Input(shape=(latent_dim,))
x = layers.Dense(128, activation='relu')(latent_inputs)
x = layers.Dense(256, activation='relu')(x)
outputs = layers.Dense(784, activation='sigmoid')(x)
decoder = Model(latent_inputs, outputs, name="decoder")

# ---------------------------
# Custom VAE Model
# ---------------------------
class VAE(Model):
    def __init__(self, encoder, decoder, **kwargs):
        super(VAE, self).__init__(**kwargs)
        self.encoder = encoder
        self.decoder = decoder

    def train_step(self, data):
        if isinstance(data, tuple):
            data = data[0]
        with tf.GradientTape() as tape:
            z_mean, z_log_var, z = self.encoder(data)
            reconstruction = self.decoder(z)

            # Reconstruction loss
            reconstruction_loss = tf.reduce_sum(
                tf.keras.losses.binary_crossentropy(data, reconstruction), axis=-1
            )
```

Variables Terminal

00:31 Python 3


Compression on MNIST dataset
Welcome to Colab - Colab
colab.research.google.com/#scrollTo=Omd8LwiP4czT
Gmail
Maps
News
Translate
Disney+ Hotstar -...
YouTube
All Bookmarks
Welcome to Colab
Cannot save changes
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
Copy to Drive
RAM
Disk
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More resources
Featured examples
Section
[2]
3m
)
reconstruction_loss = tf.reduce_mean(reconstruction_loss)
# KL divergence loss
kl_loss = -0.5 * tf.reduce_mean(
1 + z_log_var - tf.square(z_mean) - tf.exp(z_log_var)
)
total_loss = reconstruction_loss + kl_loss
grads = tape.gradient(total_loss, self.trainable_weights)
self.optimizer.apply_gradients(zip(grads, self.trainable_weights))
return {
"loss": total_loss,
"reconstruction_loss": reconstruction_loss,
"kl_loss": kl_loss,
}
def test_step(self, data):
if isinstance(data, tuple):
data = data[0]
z_mean, z_log_var, z = self.encoder(data)
reconstruction = self.decoder(z)
reconstruction_loss = tf.reduce_sum(
tf.keras.losses.binary_crossentropy(data, reconstruction), axis=-1
)
reconstruction_loss = tf.reduce_mean(reconstruction_loss)
kl_loss = -0.5 * tf.reduce_mean(
Variables
Terminal
00:31
Python 3


Compression on MNIST dataset
Welcome to Colab - Colab
colab.research.google.com/#scrollTo=Omd8LwiP4czT
Gmail
Maps
News
Translate
Disney+ Hotstar -...
YouTube
All Bookmarks
Welcome to Colab
Cannot save changes
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
Copy to Drive
RAM
Disk
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More resources
Featured examples
+ Section
[2]
3m
kl_loss = -0.5 * tf.reduce_mean(
1 + z_log_var - tf.square(z_mean) - tf.exp(z_log_var)
)
total_loss = reconstruction_loss + kl_loss
return {
"loss": total_loss,
"reconstruction_loss": reconstruction_loss,
"kl_loss": kl_loss,
}
# ----------------------------
# Compile and Train
# ----------------------------
vae = VAE(encoder, decoder)
vae.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001))
history = vae.fit(
x_train,
epochs=30,
batch_size=256,
validation_data=(x_test, None),
verbose=1,
)

Compression (Encoding) and Reconstruction
----
z_mean, z_log_var, z = encoder.predict(x_test)
x_decoded = decoder.predict(z)
Variables
Terminal
00:31
Python 3

Compression on MNIST dataset | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=Omd8LwiP4czT

Gmail Maps News Translate Disney+ Hotstar -... YouTube All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More resources
    - Featured examples
- + Section

[2] ✓ 3m

```
# ----------------------------
# Visualization
# ----------------------------
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # Original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
    plt.title("Original")
    plt.axis("off")

    # Reconstructed
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(x_decoded[i].reshape(28, 28), cmap="gray")
    plt.title("Reconstructed")
    plt.axis("off")
plt.show()
```

```
Epoch 1/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 10s 32ms/step - kl_loss: 2.4209 - loss: 69.4352 - reconstruction_loss: 67.0143 - val_kl_loss: 2.0936 - val_loss
Epoch 2/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 10s 28ms/step - kl_loss: 1.9075 - loss: 57.1722 - reconstruction_loss: 55.2647 - val_kl_loss: 2.0989 - val_loss
Epoch 3/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 27ms/step - kl_loss: 2.0547 - loss: 55.3771 - reconstruction_loss: 53.3224 - val_kl_loss: 2.1453 - val_loss
Epoch 4/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 7s 30ms/step - kl_loss: 2.1609 - loss: 54.2927 - reconstruction_loss: 52.1318 - val_kl_loss: 2.2014 - val_loss
Epoch 5/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 27ms/step - kl_loss: 2.2781 - loss: 53.3850 - reconstruction_loss: 51.1070 - val_kl_loss: 2.2363 - val_loss
Epoch 6/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 11s 32ms/step - kl_loss: 2.3838 - loss: 52.4831 - reconstruction_loss: 50.0993 - val_kl_loss: 2.3522 - val_loss
Epoch 7/30
```

Variables Terminal ✓ 00:31 Python 3

Compression on MNIST dataset | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=Omd8LwiP4czT

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

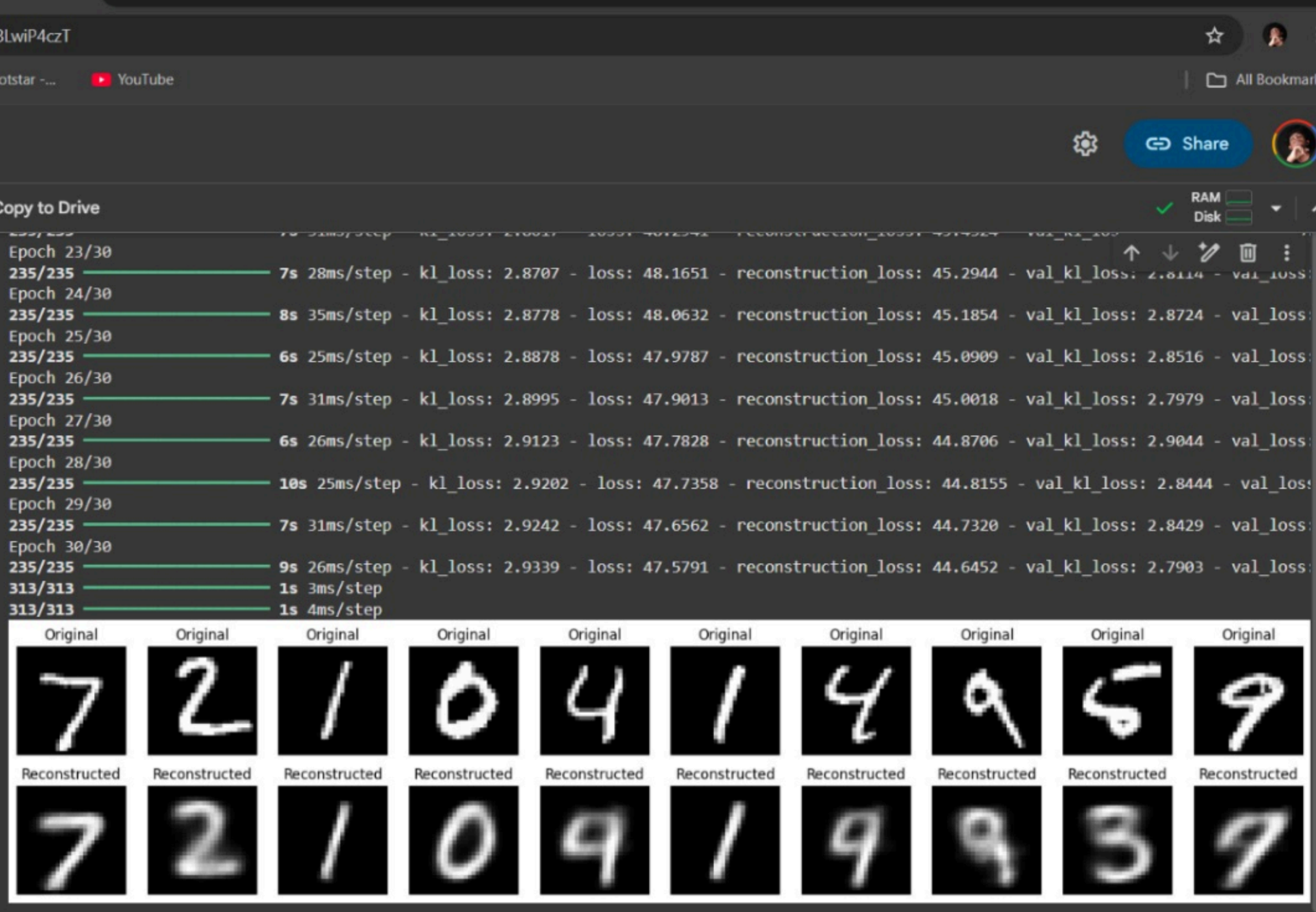
Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More resources
    - Featured examples
- + Section

```
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 7s 28ms/step - kl_loss: 2.8707 - loss: 48.1651 - reconstruction_loss: 45.2944 - val_kl_loss: 2.8114 - val_loss:
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 8s 35ms/step - kl_loss: 2.8778 - loss: 48.0632 - reconstruction_loss: 45.1854 - val_kl_loss: 2.8724 - val_loss:
Epoch 25/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 25ms/step - kl_loss: 2.8878 - loss: 47.9787 - reconstruction_loss: 45.0909 - val_kl_loss: 2.8516 - val_loss:
Epoch 26/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 7s 31ms/step - kl_loss: 2.8995 - loss: 47.9013 - reconstruction_loss: 45.0018 - val_kl_loss: 2.7979 - val_loss:
Epoch 27/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 26ms/step - kl_loss: 2.9123 - loss: 47.7828 - reconstruction_loss: 44.8706 - val_kl_loss: 2.9044 - val_loss:
Epoch 28/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 10s 25ms/step - kl_loss: 2.9202 - loss: 47.7358 - reconstruction_loss: 44.8155 - val_kl_loss: 2.8444 - val_loss:
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 7s 31ms/step - kl_loss: 2.9242 - loss: 47.6562 - reconstruction_loss: 44.7320 - val_kl_loss: 2.8429 - val_loss:
Epoch 30/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 26ms/step - kl_loss: 2.9339 - loss: 47.5791 - reconstruction_loss: 44.6452 - val_kl_loss: 2.7903 - val_loss:
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
```



Variables | Terminal | 00:31 | Python 3

11. Experiments using Variational Autoencoder.

Aim :- To implement a Variational Autoencoder (VAE) for compressing and reconstructing the MNIST handwritten digit images and to analyze its ability to learn a smooth latent space representation.

objectives :-

1. To understand the working of variational. Autoencoders as probabilistic extension of standard autoencoders
2. to learn a latent distribution for efficient image compression.
3. To reconstruct MNIST digits from the latent space
4. To visualize and interpret the latent space and reconstructed outputs
5. To compare the performance of VAE with.

Pseudo code :-

1. Import TensorFlow and MNIST dataset.
2. Normalize and flatten the dataset
3. Define encoder:
   Input → Dense (512, relu)
   ...
   Dense (latent_dim*2) → outs mean (μ)
   log variance (log σ^2)
4. sampling function :-
   $z = \mu + \exp(0.5 * \log\sigma^2) * E$

5) Define decoder
Input(z) → Dense(512, relu) → Dense(784, sigmoid)

6) Define VAE model = Encoder + Sampling + Decoder

7) loss = Reconstruction loss + KL divergence loss.

8) Train model for 50 epochs

9) Visualize original and reconstructed images

10) plot 2D latent space.

Result:-

- The variational Autoencoder successfully compressed MNIST digit images into a 2-D latent space.
- The KL divergence ensured a smooth latent distribution enabling interpolation b/w digits.
- Reconstruction loss ≈ 0.02 - 0.03 depending on training
- The model demonstrates that VAEs can learn meaningful continuous and structured latent representation useful for data compression and generation.

epoch 1/30 :-

KL loss :- 2.4709 - loss :- 69.6352.

reconstruction - loss :- 67.0143 - Val KL loss :- 2.0936.

epoch 10/30 :-

KaL loss :- 2.5976 - loss :- 50.5912.

reconstruction - loss :- 47.9936 - Val KL loss :- 2.6866

epoch 20/30 :-

KaL loss :- 2.8249 - loss :- 48.5557 -

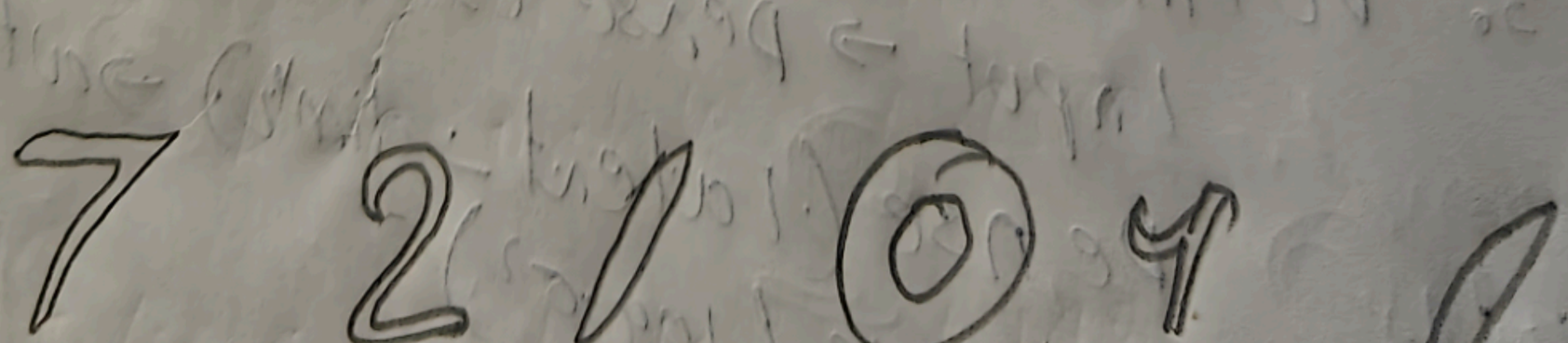
reconstruction - loss. 45.9307 - Val KL loss : 2.7994

epoch 30/30 :-

KaL - loss :- 2.9339 - loss :- 47.5791 -

reconstruction - loss :- 44.6452 - val loss 2.2903

Original



Reconstructed