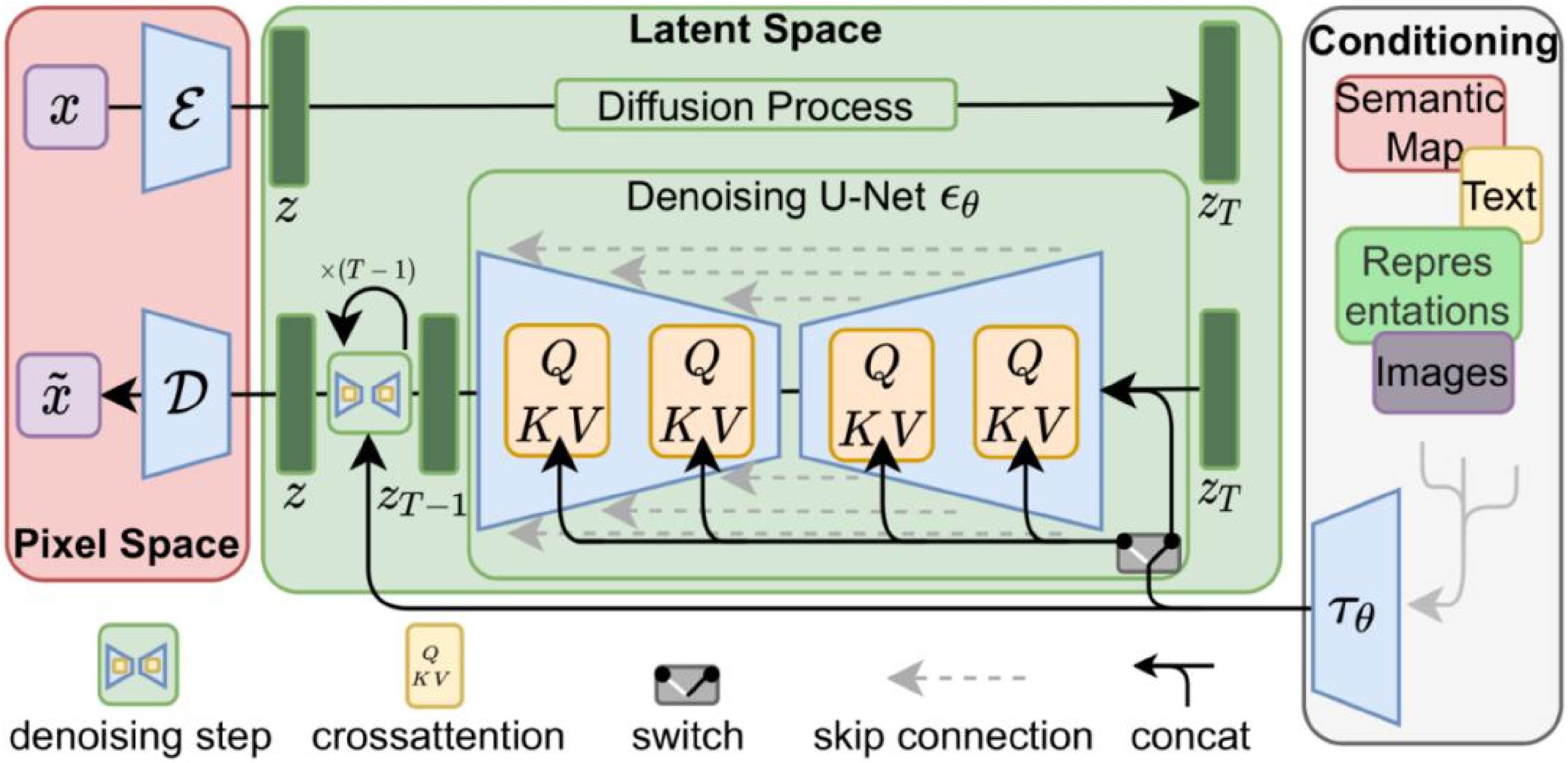
Code is “cheap”, show me the PROMPT!

Stable Diffusion基本原理与Keras简单示例

林嵩

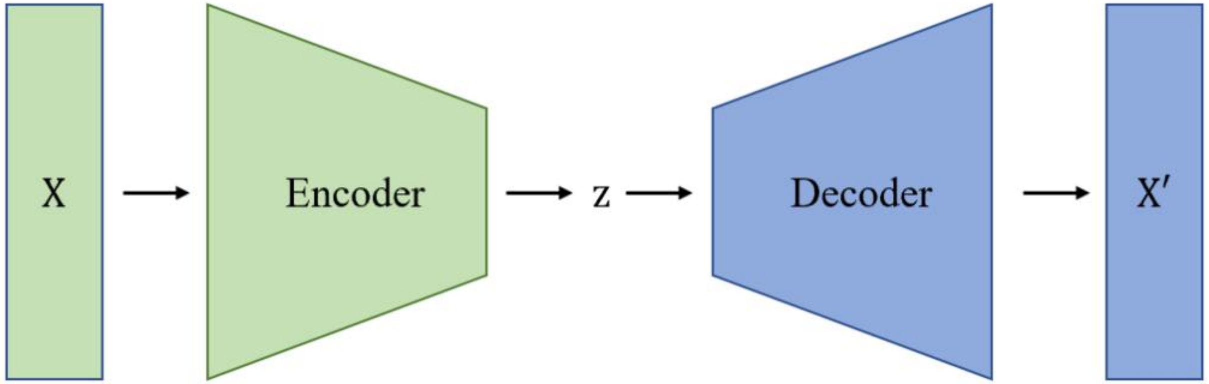
目录

Autoencoder
diffusion
U-net
text embedding
cross attention

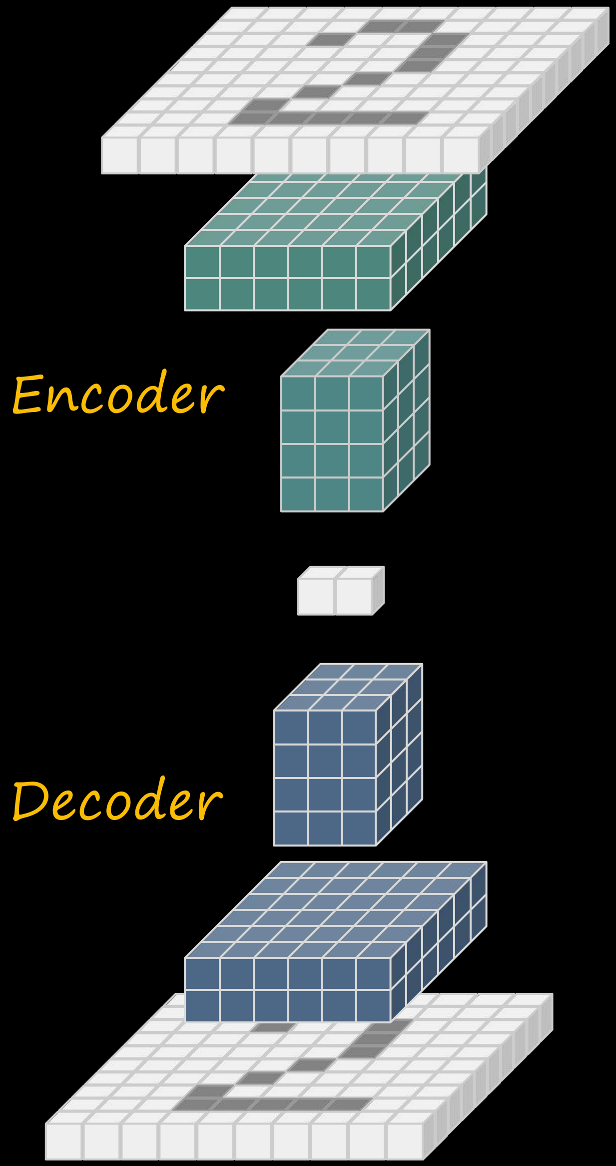


AE

自动编码机（Auto Encoder，AE）是早期较为简单的生成模型，通过一个编码器将输入编码成隐变量，再通过一个解码器解码成重构样本 。



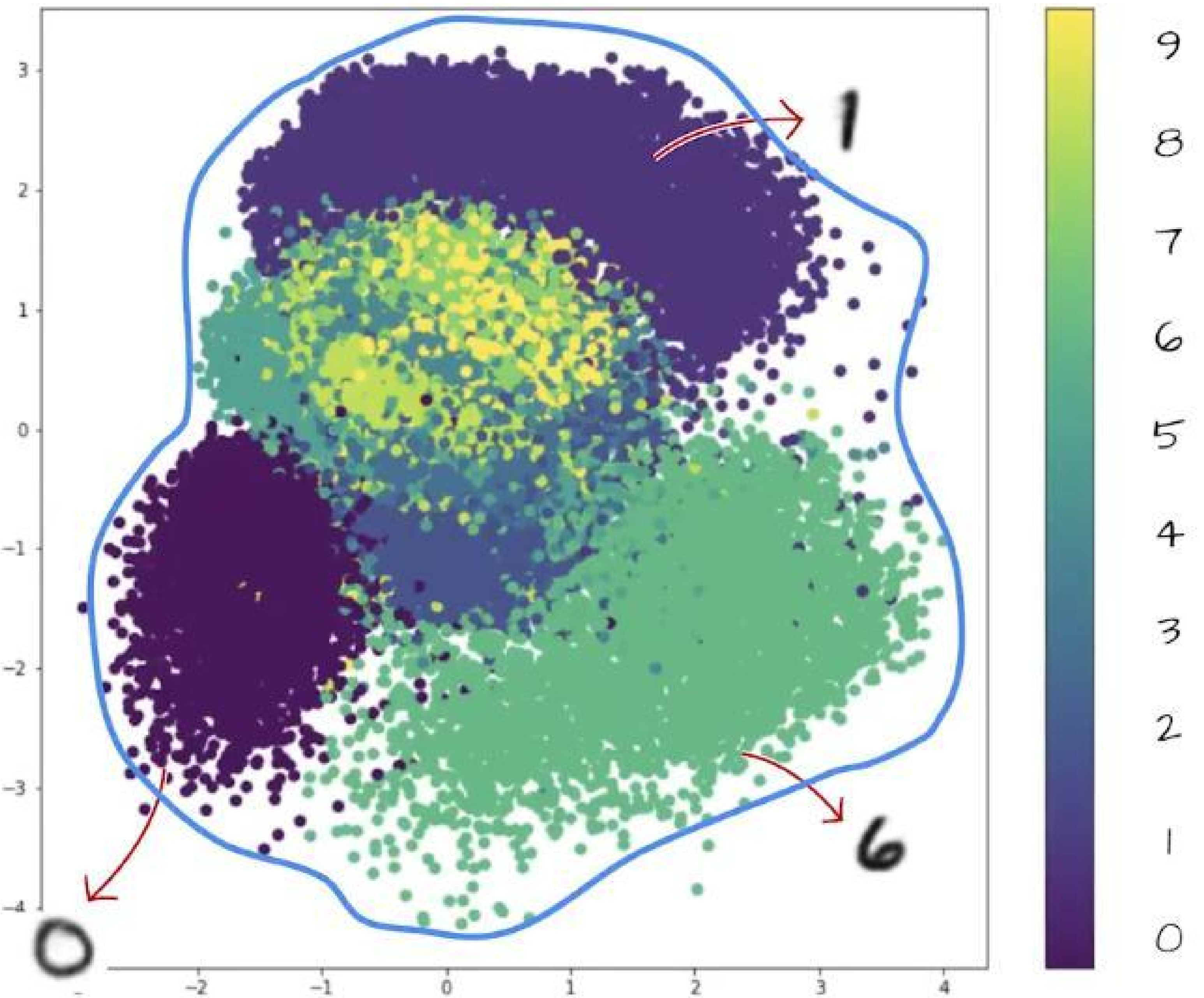
Autoencoder




```python
autoencoder = tf.keras.Sequential([

  layers.Conv2D(32, 3, strides=2,
                activation="relu", padding="same"),
  layers.Conv2D(64, 3, strides=2,
                activation="relu", padding="same"),
  layers.Flatten(),
  layers.Dense(latent_dim),  # latent_dim=2

  layers.Dense(7 * 7 * 64, activation="relu"),
  layers.Reshape((7, 7, 64)),
  layers.Conv2DTranspose(32, 3, strides=2,
                activation="relu", padding="same"),
  layers.Conv2DTranspose(1, 3, strides=2,
               activation="sigmoid", padding="same")
])
```

Encoder output on training data

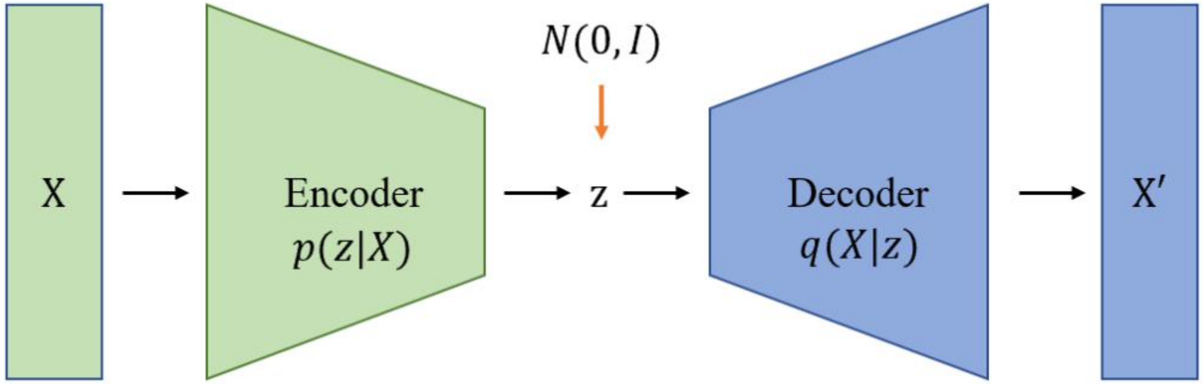
Decoder output from latent space

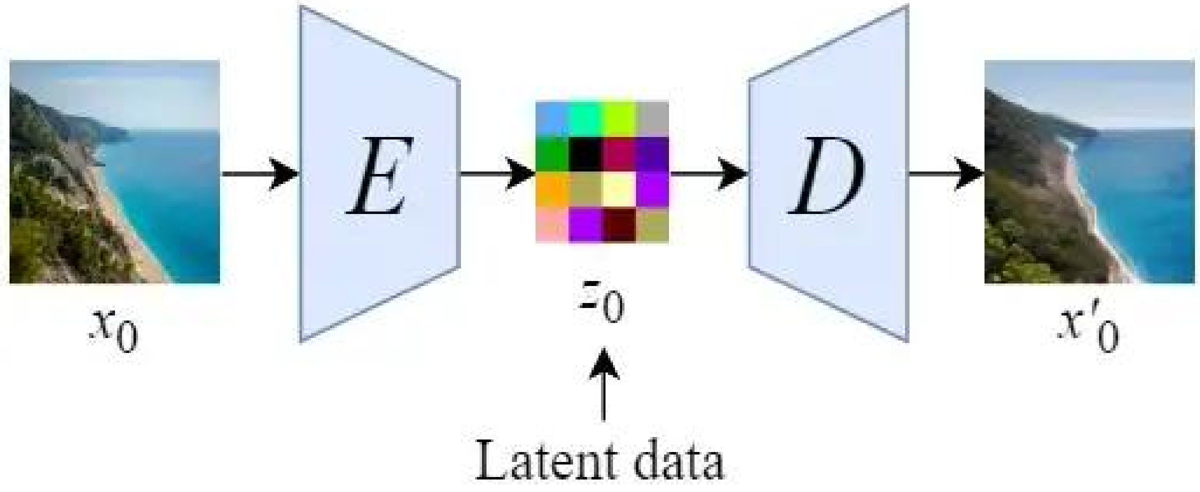


VAE - 有点像但不多

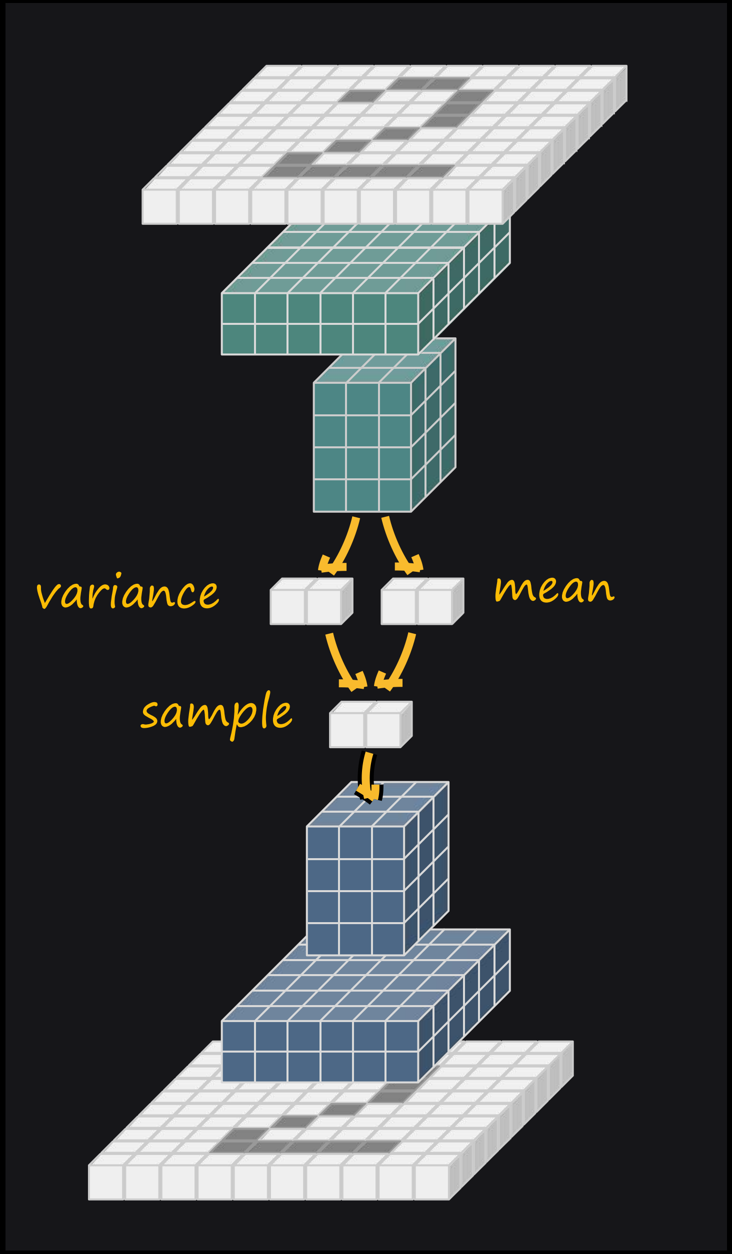
VAE（Varient Auto Encoder）生成类似于输入样本，但是不完全一样的新东西，在合理范围内变花样。

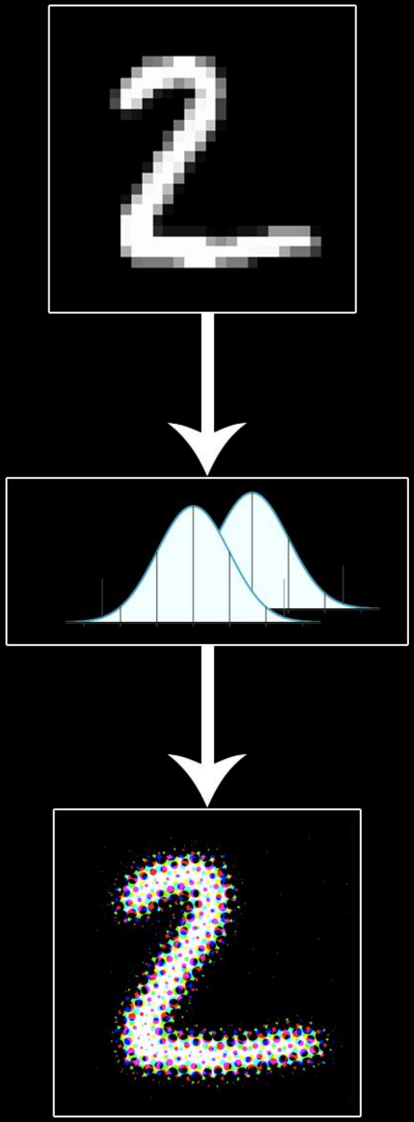
在latent上操作而不需要encoder，直接得到decoder的输出。





Variational Autoencoder





Input image

Probabilistic representation

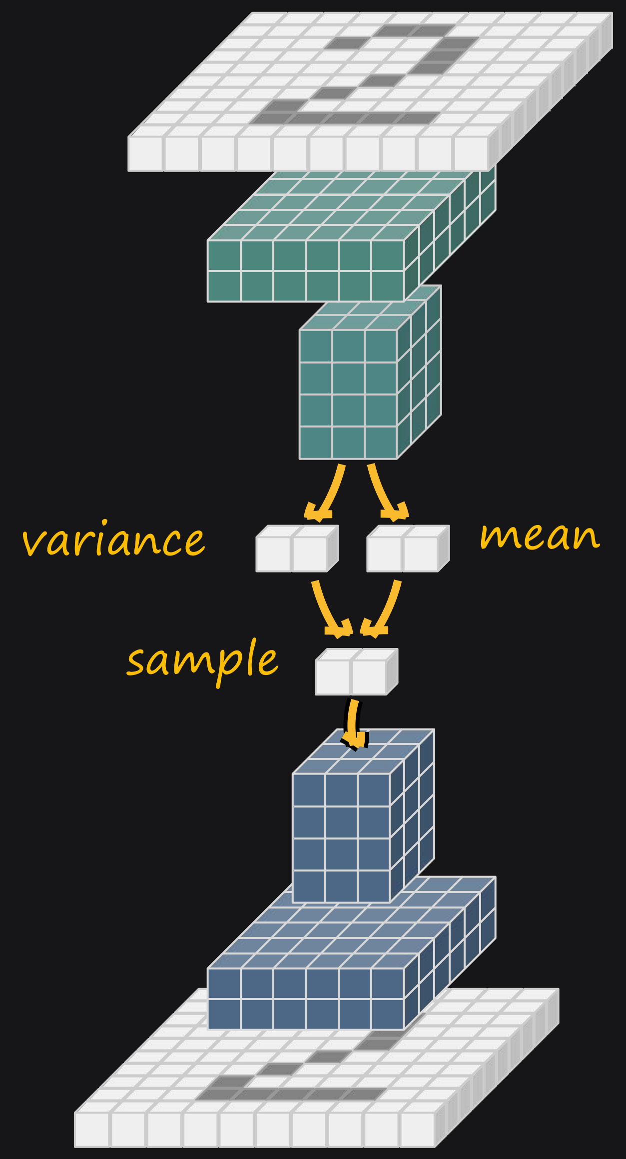
Reconstructed image

VAE part #1
sampling layer

VAE part #2
variational encoder

VAE part #3
regularization loss

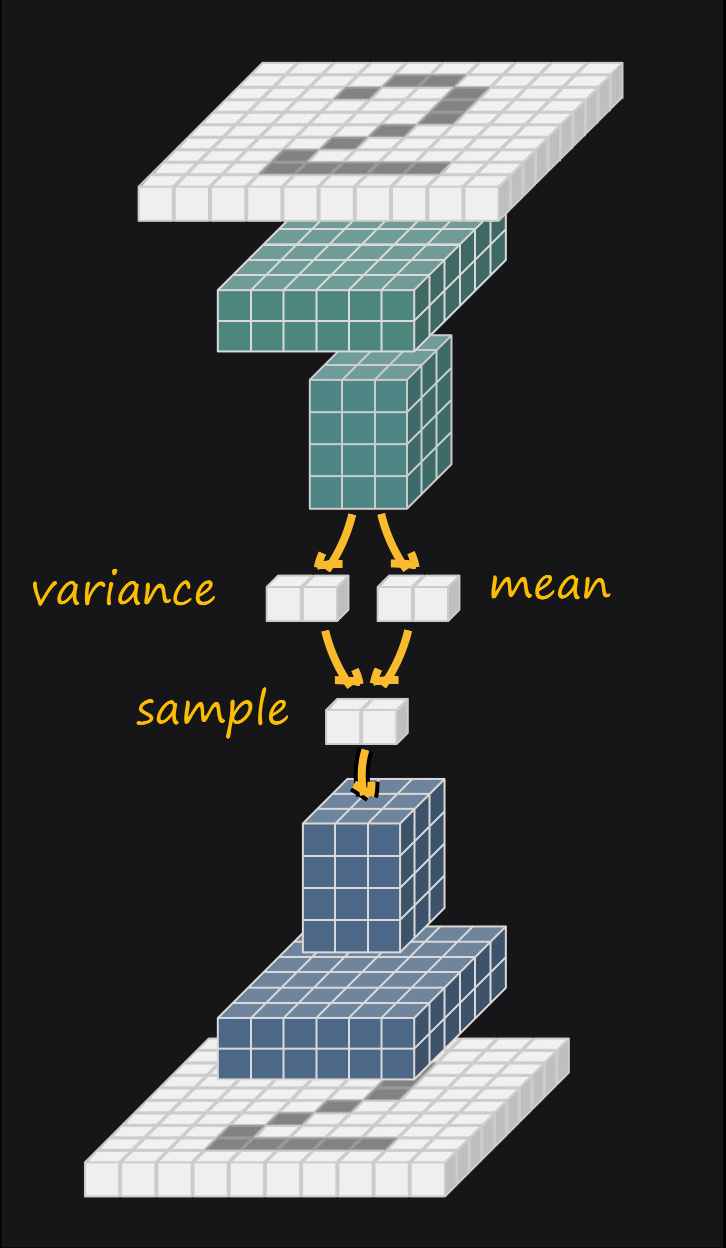
VAE part #4
reconstruction loss




```
encoder_input = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(32, 3, strides=2,
                       activation="relu",
padding="same")(encoder_input)
x = layers.Conv2D(64, 3, strides=2,activation="relu",
padding="same")(x)
x = layers.Flatten()(x)


# latent_dim = 2
z_mean = layers.Dense(latent_dim, name="z_mean")(x)
z_log_var = layers.Dense(latent_dim, name="z_log_var")(x)
z = Sampling()(z_mean, z_log_var)

encoder = keras.Model(encoder_input, [z_mean, z_log_var, z])
```

Functional model

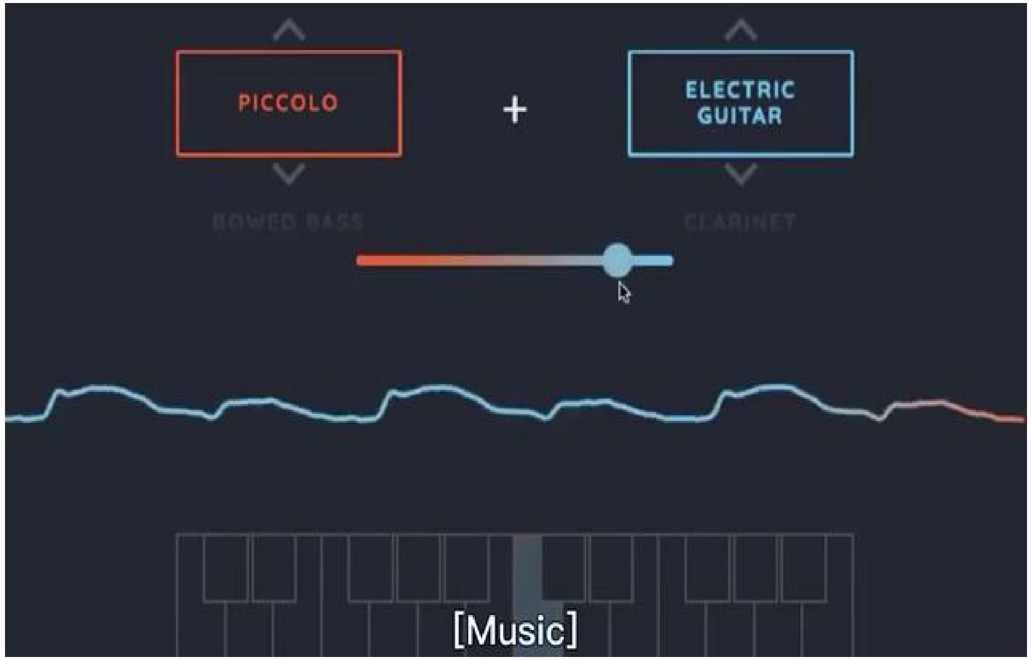
VAE part #2
variational
encoder



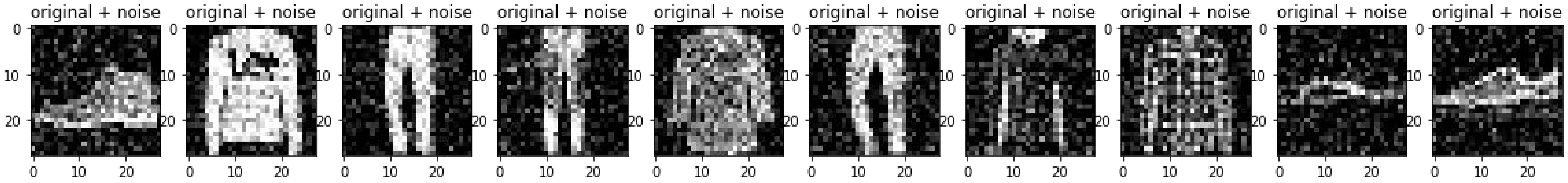
Model Composition

```
latent_input = keras.Input(shape=(latent_dim,))
y = layers.Dense(7 * 7 * 64, activation="relu")(latent_input)
y = layers.Reshape((7, 7, 64))(y)
y = layers.Conv2DTranspose(32, 3, strides=2,
                           activation="relu", padding="same")(y)
y = layers.Conv2DTranspose(1, 3, strides=2,
                           activation="sigmoid", padding="same")(y)

decoder = keras.Model(latent_input, y)

z_mean, z_log_var, z = encoder(encoder_input)
decoder_output       = decoder(z)

vae = keras.Model(encoder_input, decoder_output)
```

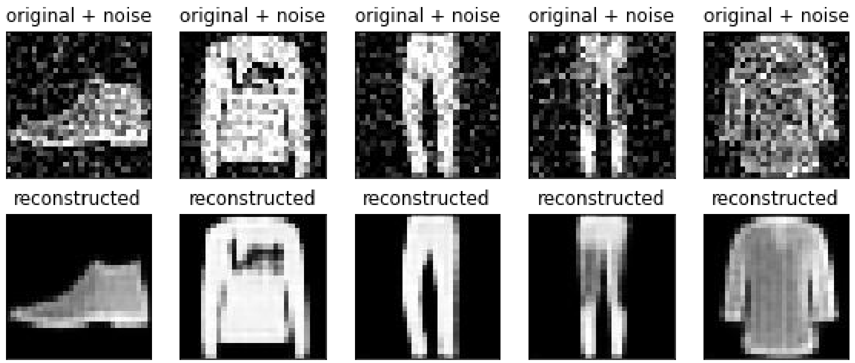
应用场景：
- 降噪
- 异常检测
- renlian生成
- 曲风变化



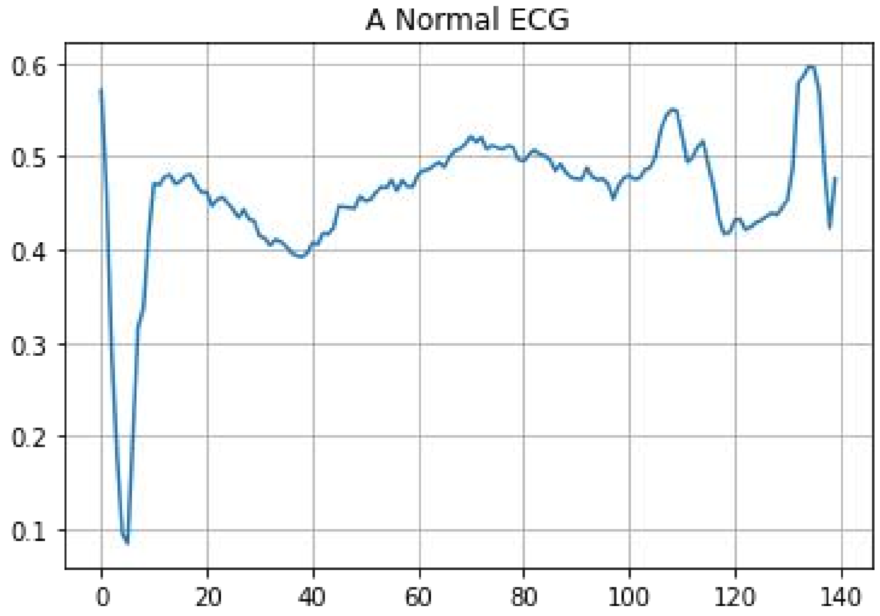
```
(x_train, _), (x_test, _) = fashion_mnist.load_data()

x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]

noise_factor = 0.2
x_train_noisy = x_train + noise_factor * tf.random.normal(shape=x_train.shape)
x_test_noisy = x_test + noise_factor * tf.random.normal(shape=x_test.shape)

x_train_noisy = tf.clip_by_value(x_train_noisy, clip_value_min=0., clip_value_max=1.)
x_test_noisy = tf.clip_by_value(x_test_noisy, clip_value_min=0., clip_value_max=1.)
```

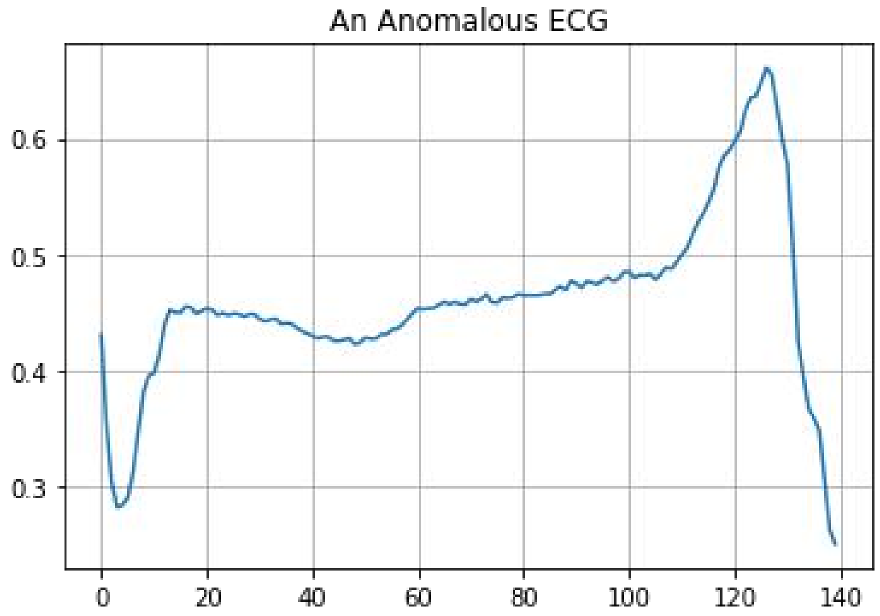


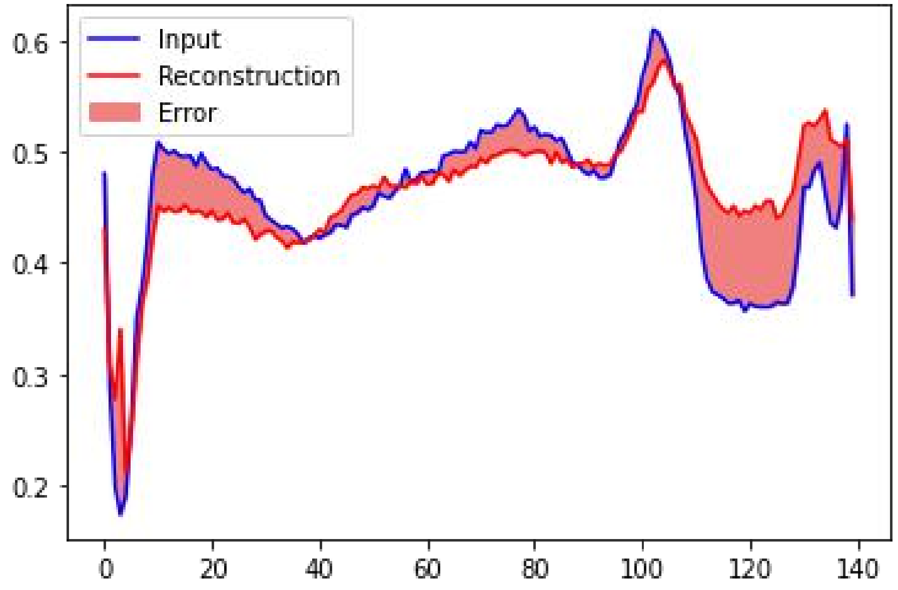
```
class Denoise(Model):
  def __init__(self):
    super(Denoise, self).__init__()
    self.encoder = tf.keras.Sequential([
      layers.Input(shape=(28, 28, 1)),
      layers.Conv2D(16, (3, 3), activation='relu', padding='same',
strides=2),
      layers.Conv2D(8, (3, 3), activation='relu', padding='same',
strides=2)])

    self.decoder = tf.keras.Sequential([
      layers.Conv2DTranspose(8, kernel_size=3, strides=2,
activation='relu', padding='same'),
      layers.Conv2DTranspose(16, kernel_size=3, strides=2,
activation='relu', padding='same'),
      layers.Conv2D(1, kernel_size=(3, 3), activation='sigmoid',
padding='same')])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder = Denoise()

autoencoder.fit(x_train_noisy, x_train,                epochs=10,
                shuffle=True,                validation_data=(x_test_noisy,
x_test))
```

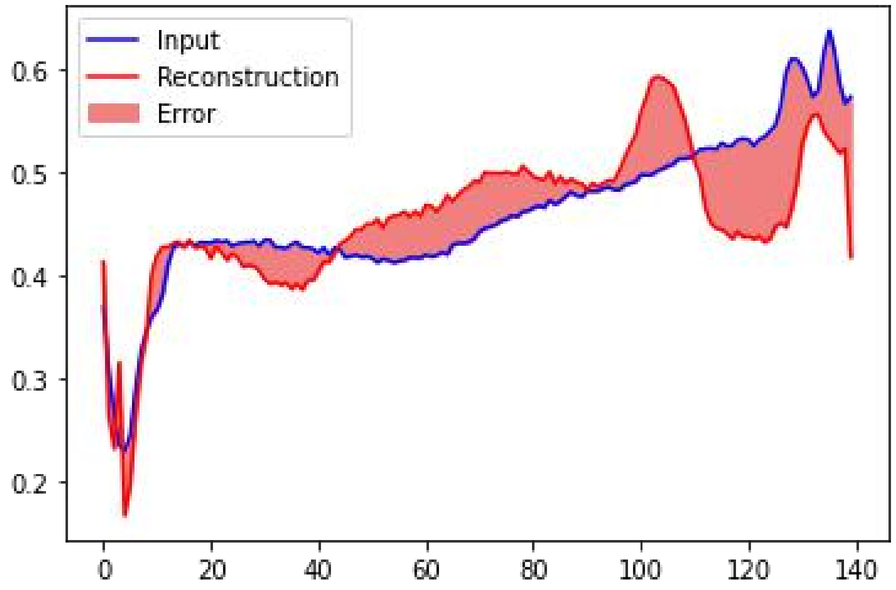


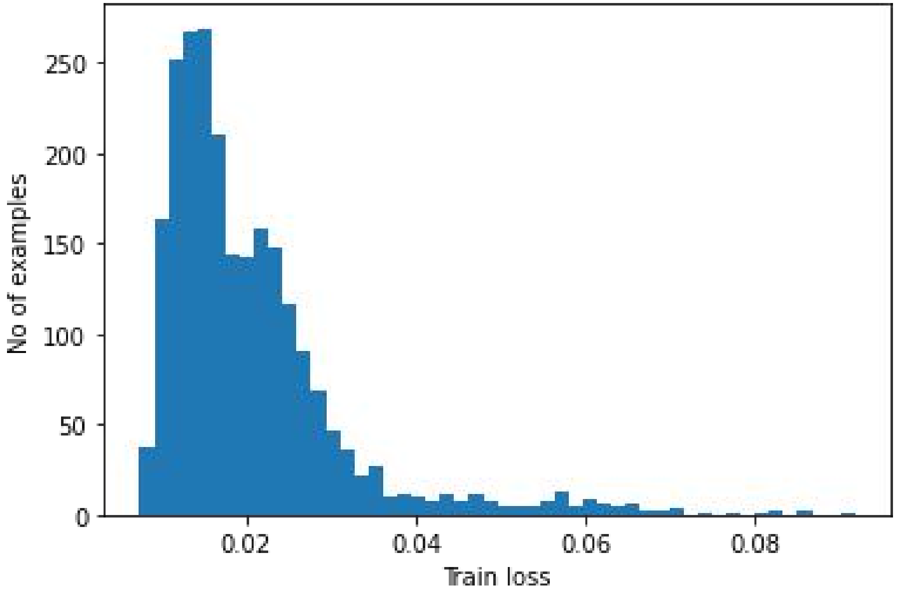
```
class AnomalyDetector(Model):
  def __init__(self):
    super(AnomalyDetector, self).__init__()
    self.encoder = tf.keras.Sequential([
      layers.Dense(32, activation="relu"),
      layers.Dense(16, activation="relu"),
      layers.Dense(8, activation="relu")])

    self.decoder = tf.keras.Sequential([
      layers.Dense(16, activation="relu"),
      layers.Dense(32, activation="relu"),
      layers.Dense(140, activation="sigmoid")])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder = AnomalyDetector()
autoencoder.compile(optimizer='adam', loss='mae')

history = autoencoder.fit(normal_train_data, normal_train_data,
          epochs=20,  batch_size=512,

          validation_data=(test_data, test_data), shuffle=True)
```

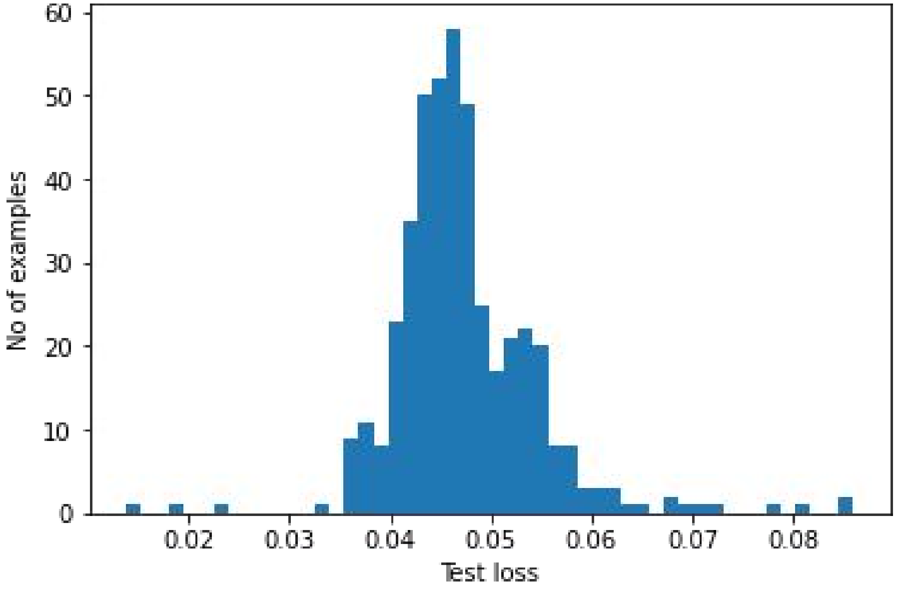


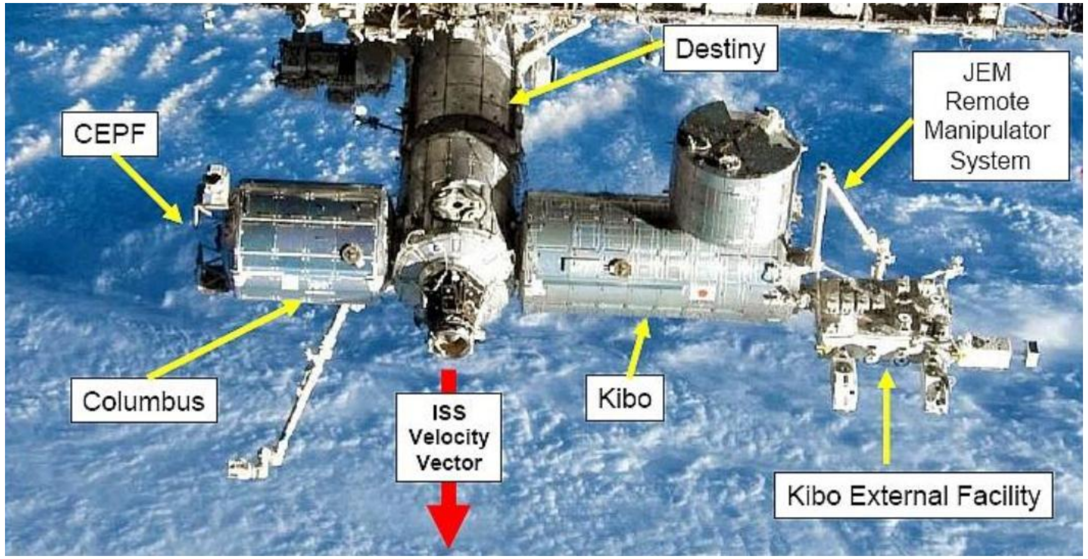


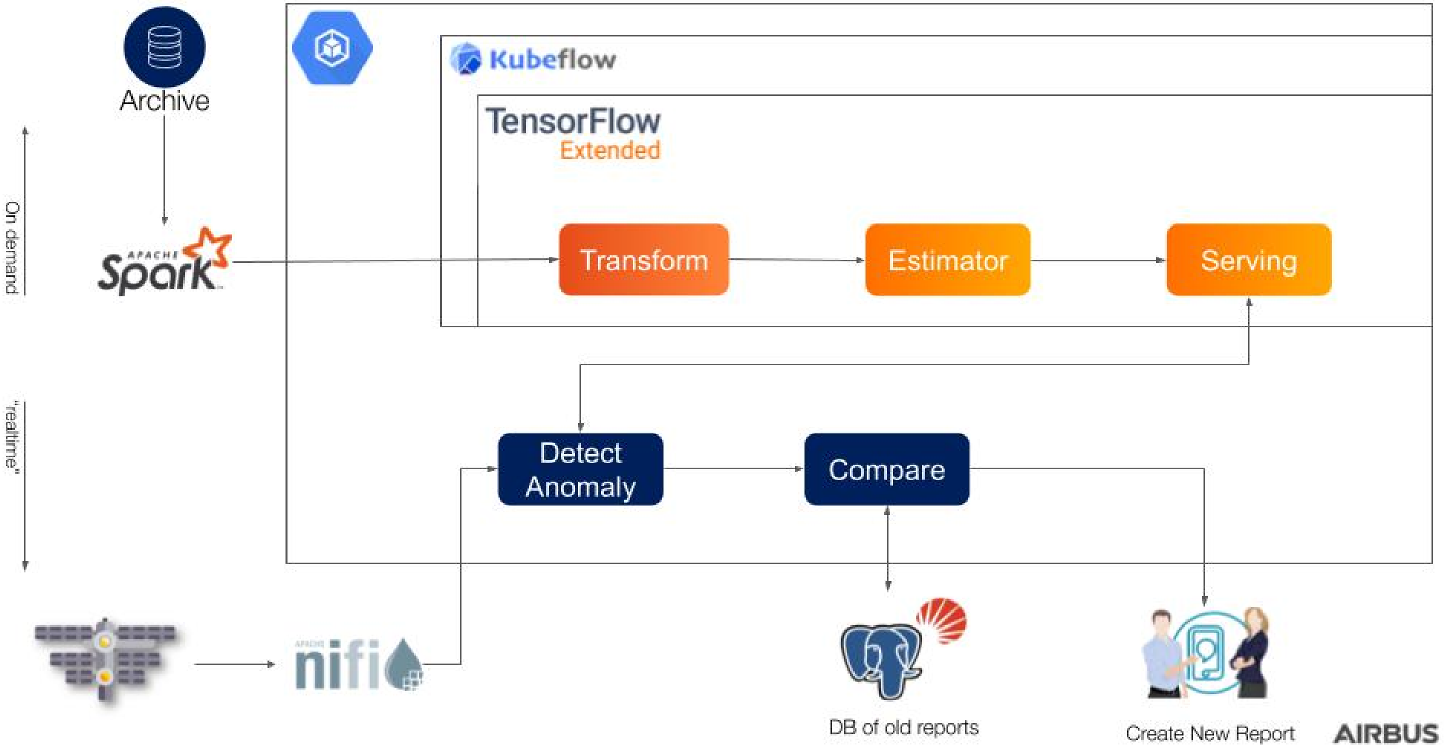

Input
Reconstruction
Error
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100
120
140


Input
Reconstruction
Error
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100
120
140

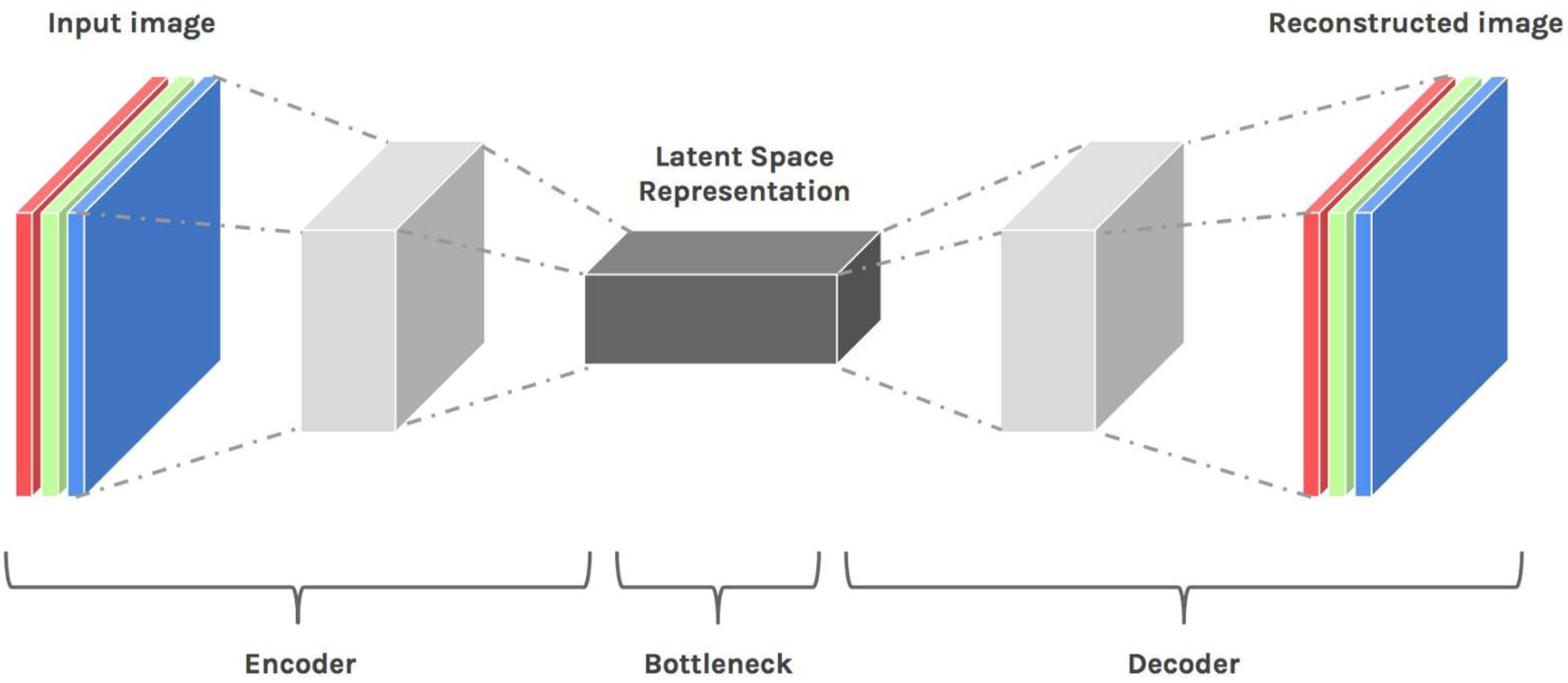

No of examples
0
50
100
150
200
250
0.02
0.04
0.06
0.08
Train loss

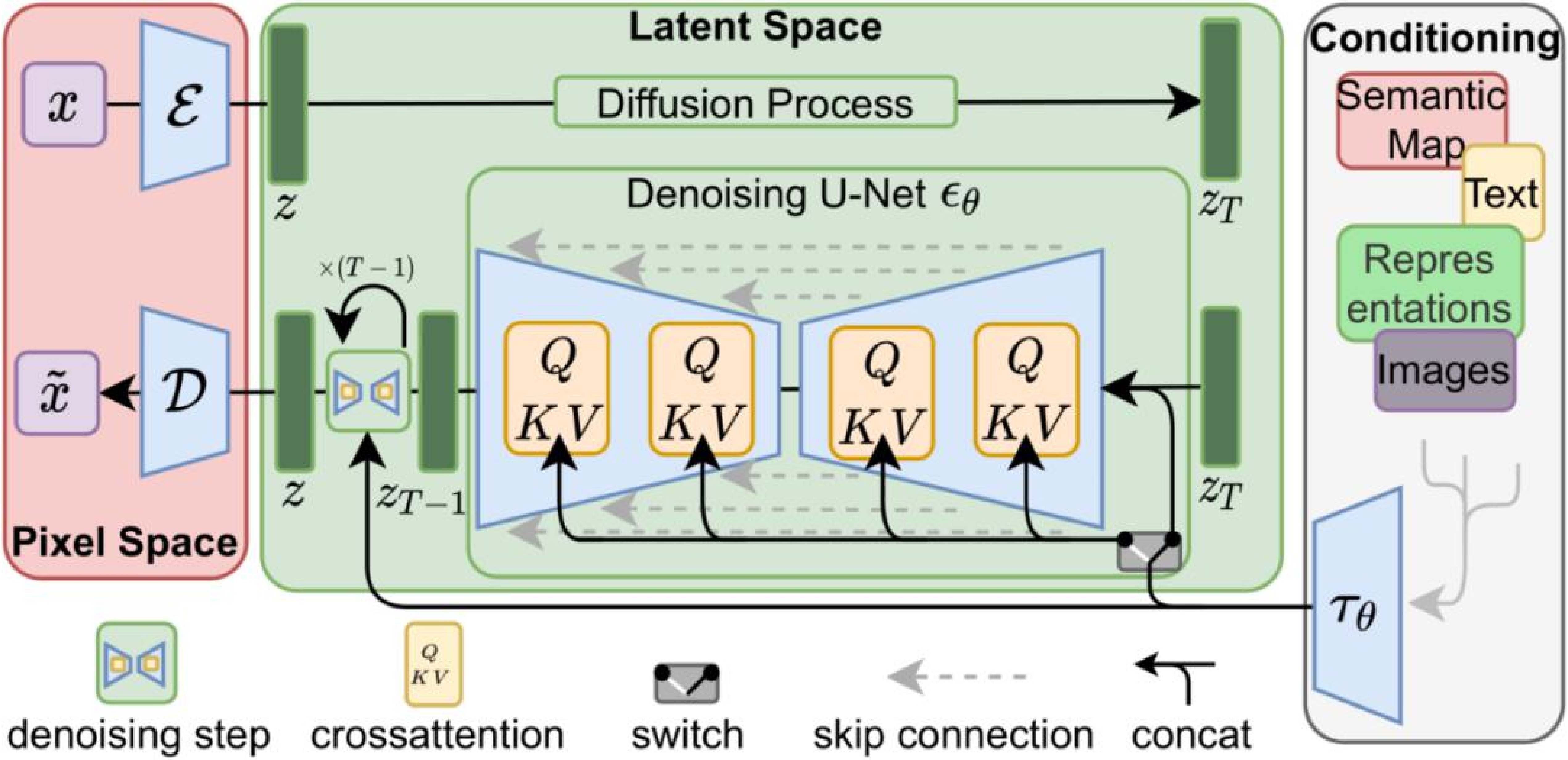

No of examples
0
10
20
30
40
50
60
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Test loss


Destiny
CEPF
JEM Remote Manipulator System
Columbus
ISS Velocity Vector
Kibo
Kibo External Facility


On demand
"realtime"
Archive
Spark
Kubeflow
TensorFlow Extended
Transform
Estimator
Serving
Detect Anomaly
Compare
nifi
DB of old reports
Create New Report
AIRBUS

AE机制的核心-latent

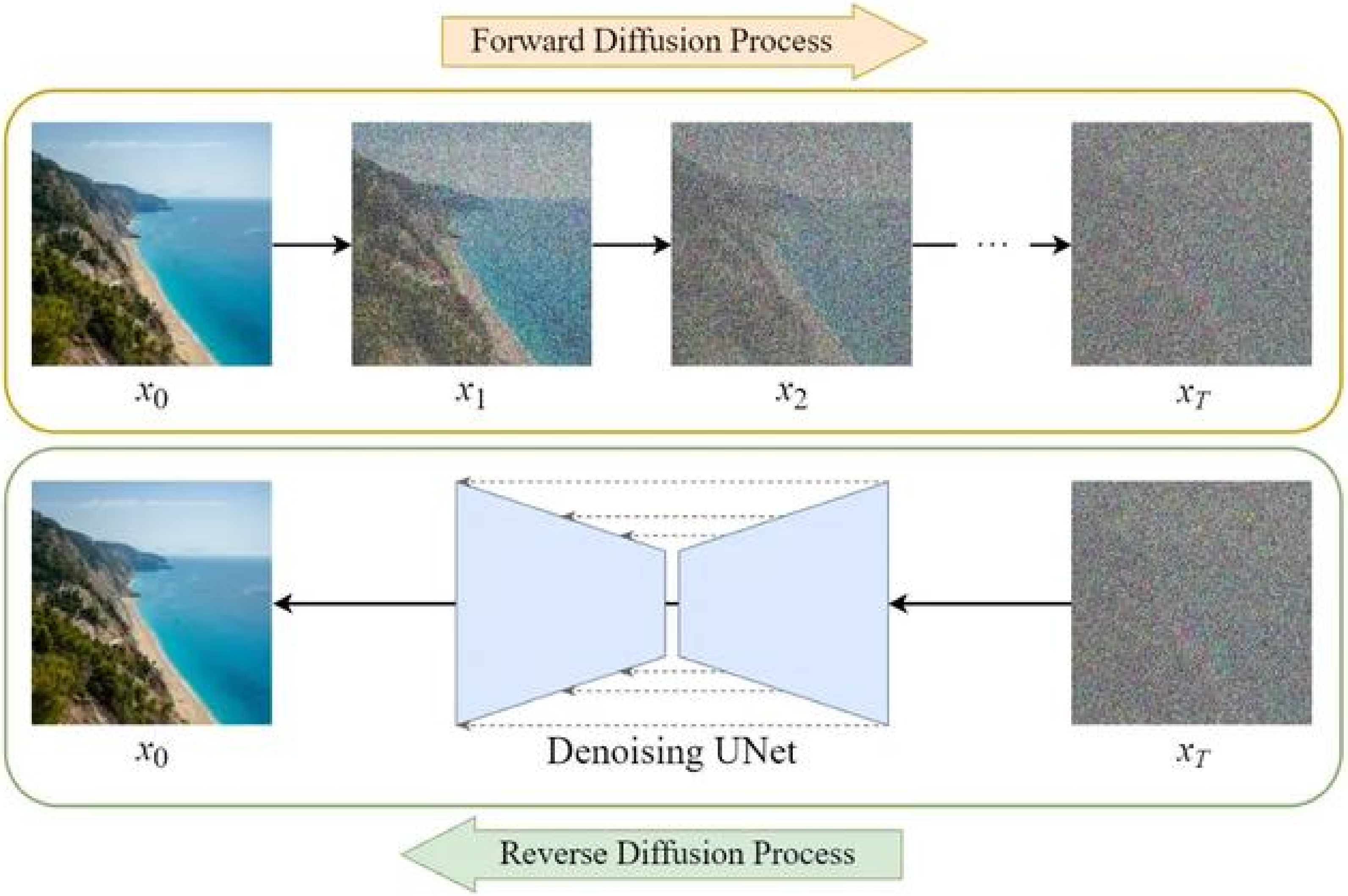


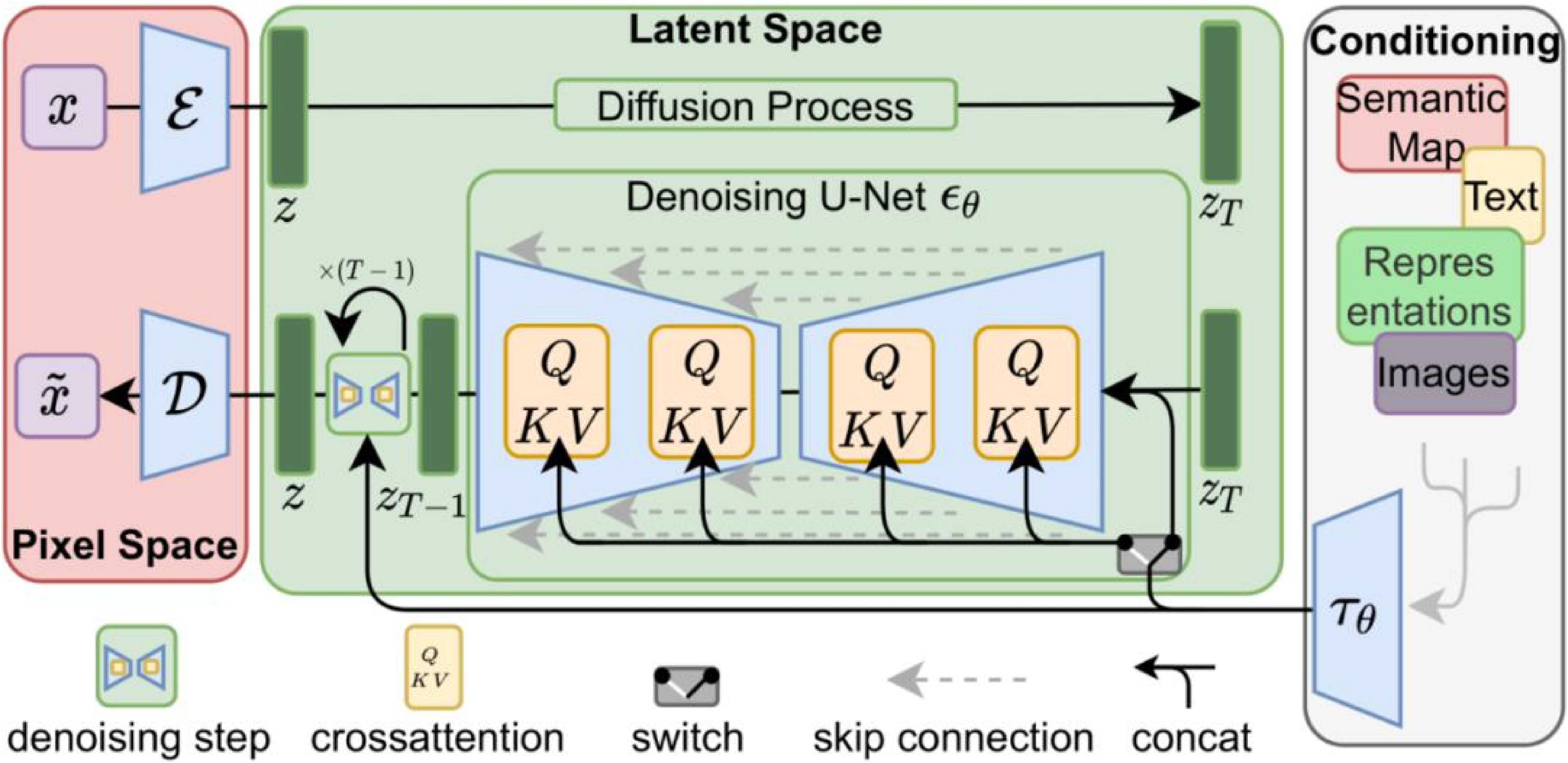

Latent Space
Conditioning
Diffusion Process
Semantic Map
Text
Denoising U-Net ϵ_θ
Repres entations
Images
x
$\mathcal{E}$
z
z_T
$\times(T-1)$
$\tilde{x}$
$\mathcal{D}$
z_{T-1}
Q
$K\,V$
τ_θ
Pixel Space
denoising step
crossattention
switch
skip connection
concat

diffusion

正向：水里加盐
逆向：慢镜头分步倒放

一个模型的状态转移如果符合马尔科夫链的状态转移矩阵，当状态转移到一定次数时，模型状态最终收敛于一个平稳分布。

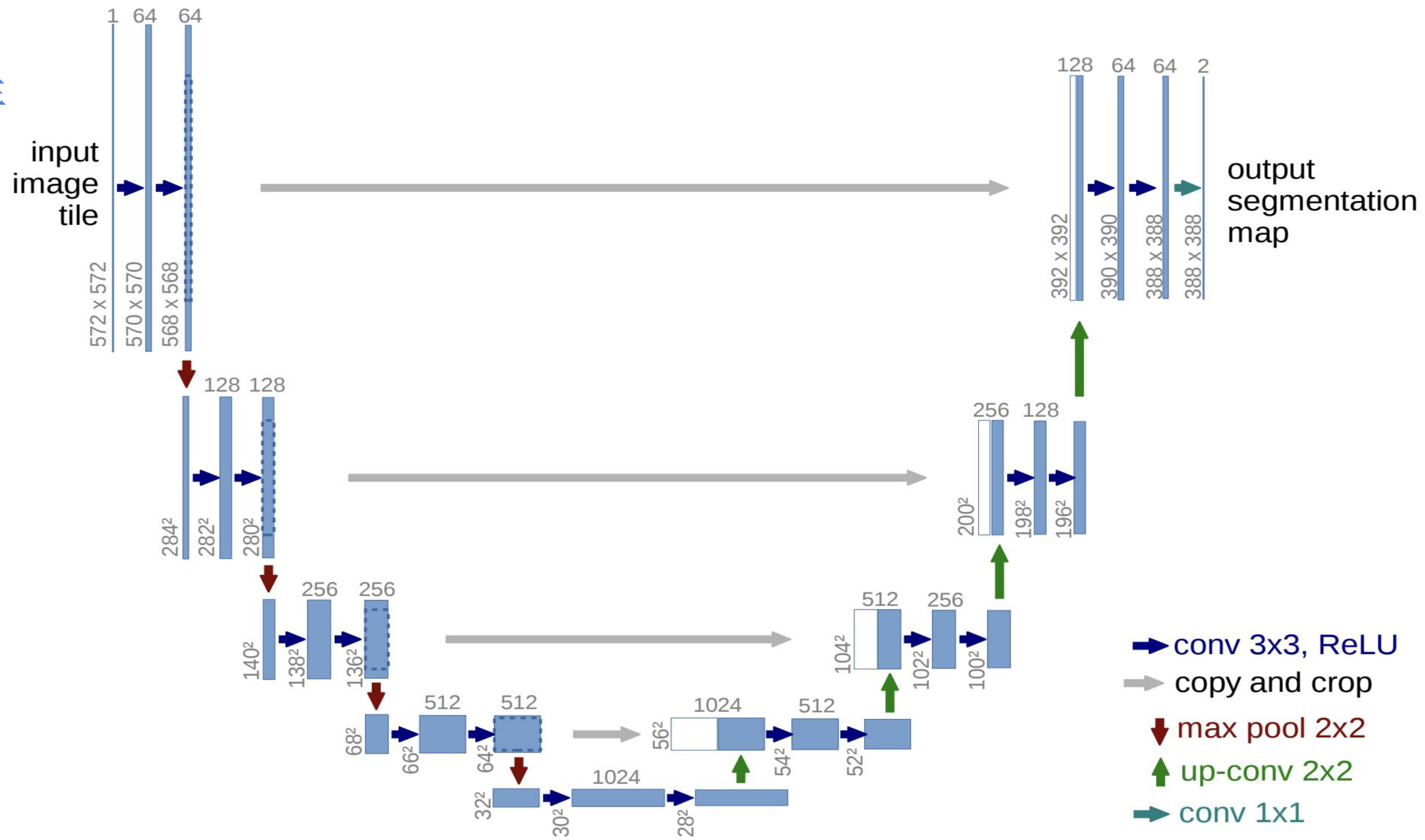



Latent Space
Conditioning
Semantic Map
Text
Repres entations
Images
Pixel Space
x
$\mathcal{E}$
z
Diffusion Process
z_T
Denoising U-Net ϵ_θ
$\times(T-1)$
Q $K\,V$
$\tilde{x}$
$\mathcal{D}$
z_{T-1}
τ_θ
denoising step
crossattention
switch
skip connection
concat

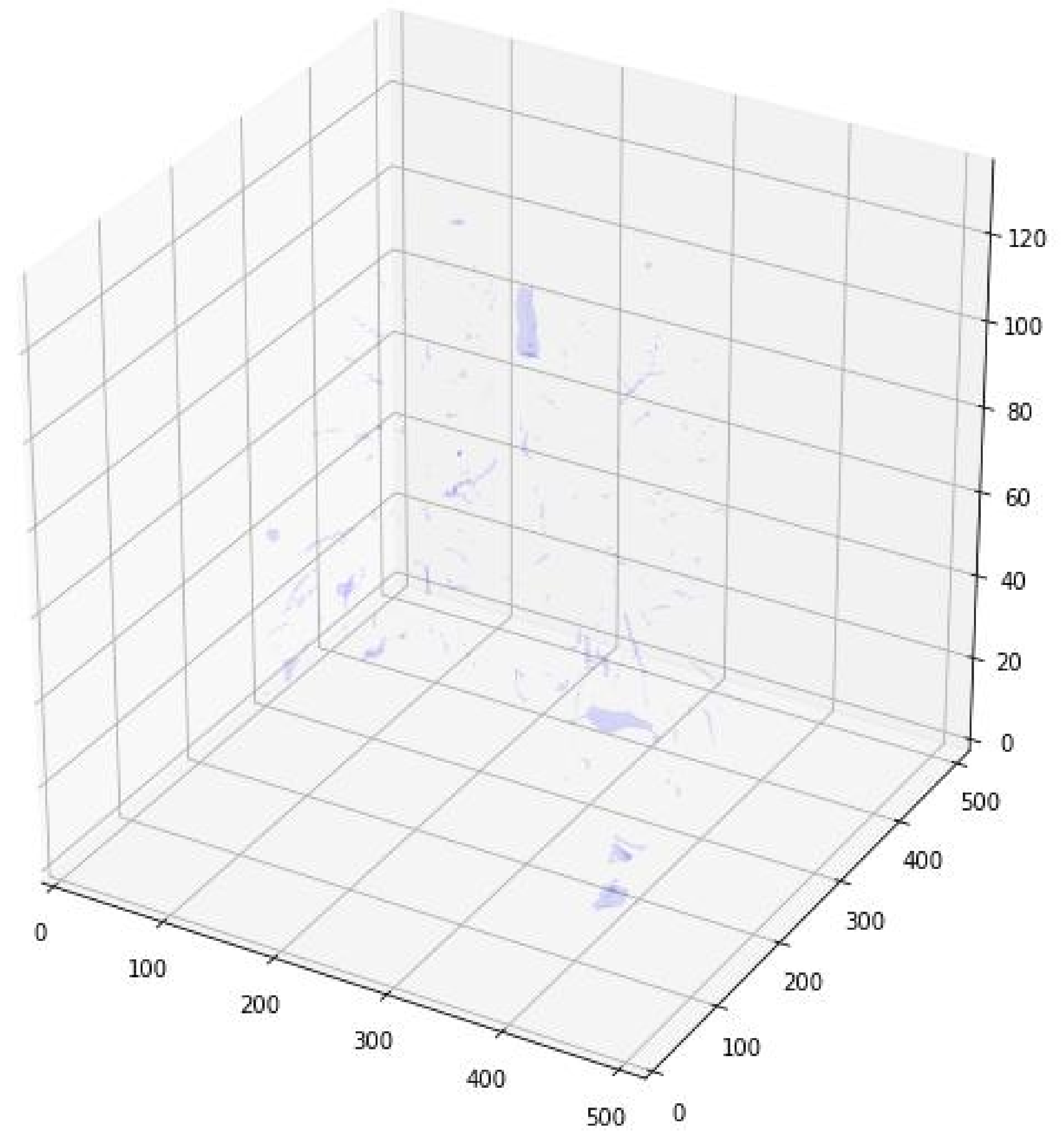
U-net

下采样提取目标特征，上采样对其像素点进行分类。

skip connection帮助解决下采样丢失掉的细节损失。




0.0
0.2
0.4
0.6
0.8
1.0


0
20
40
60
80
100
120
0
100
200
300
400
500
0
100
200
300
400
500

(a) 实验原图

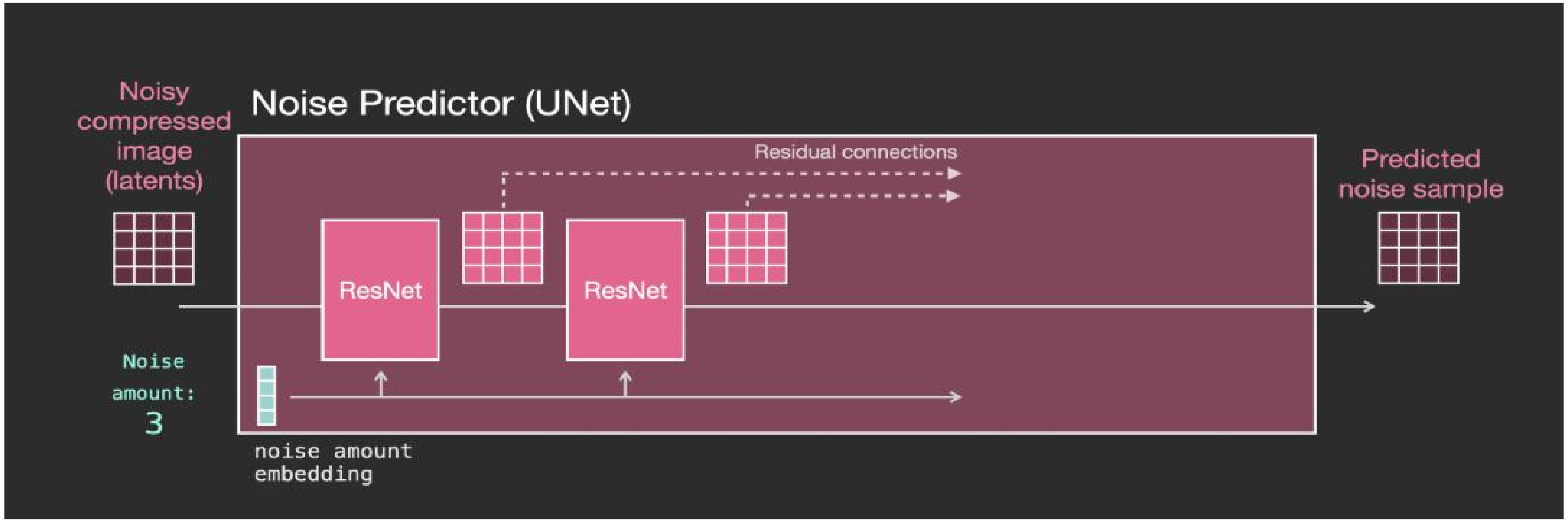
(b) 目视解译图

(c) 道路提取图

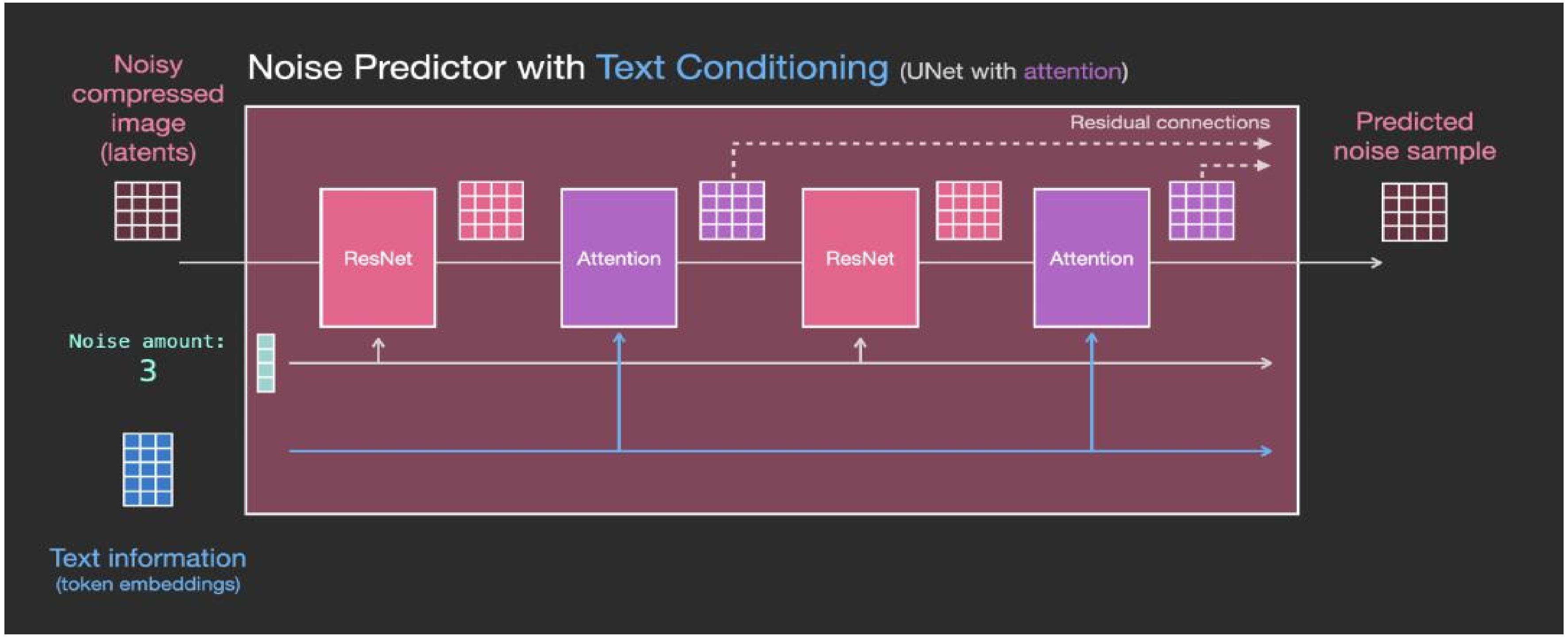
(d) 道路叠加图

U-net

一般降噪：
ResNet块添加噪声。



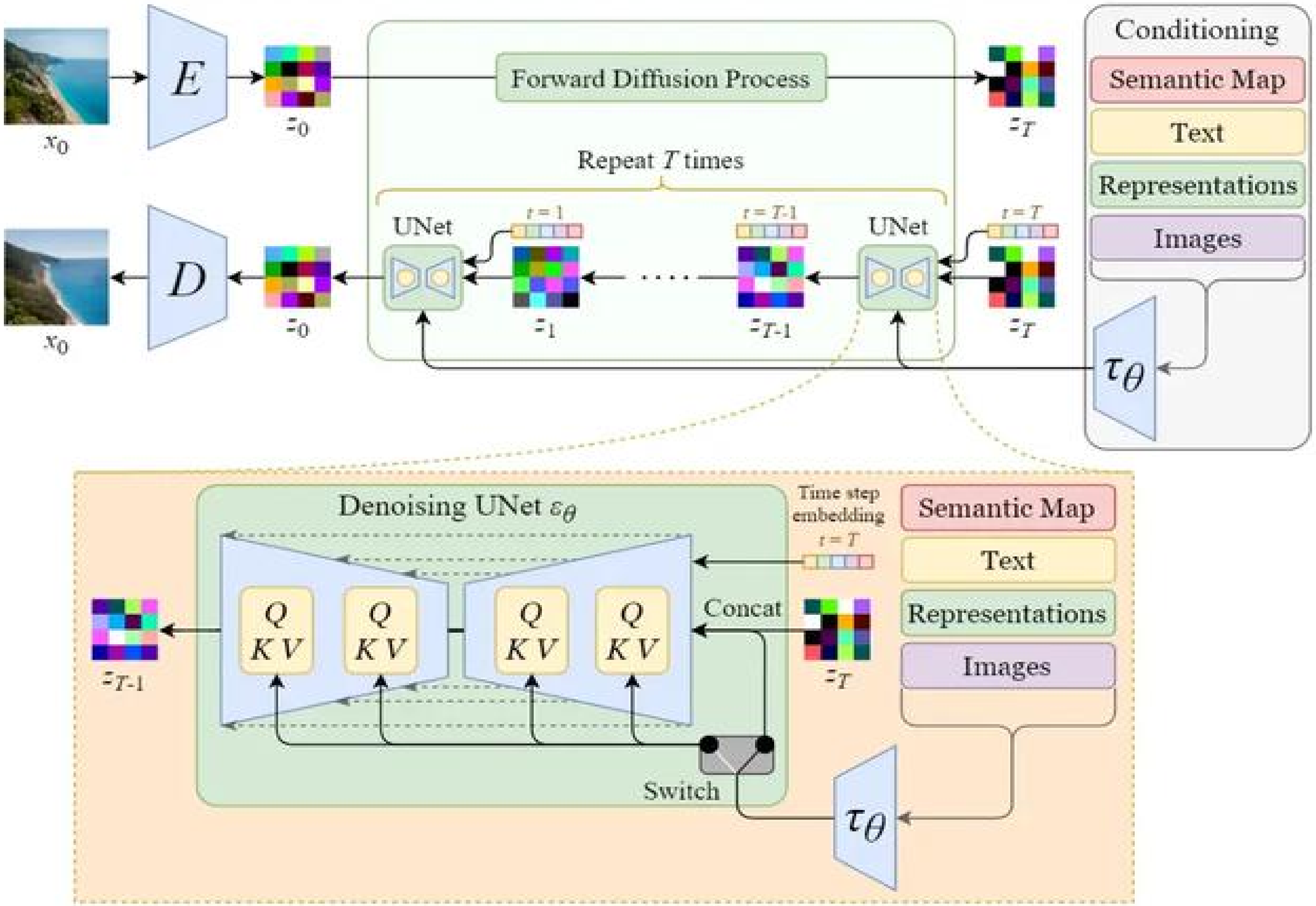
带文本降噪：
ResNet块之间添加
Attention层。



U-net

训练过程：
文本引导下的
latent生成

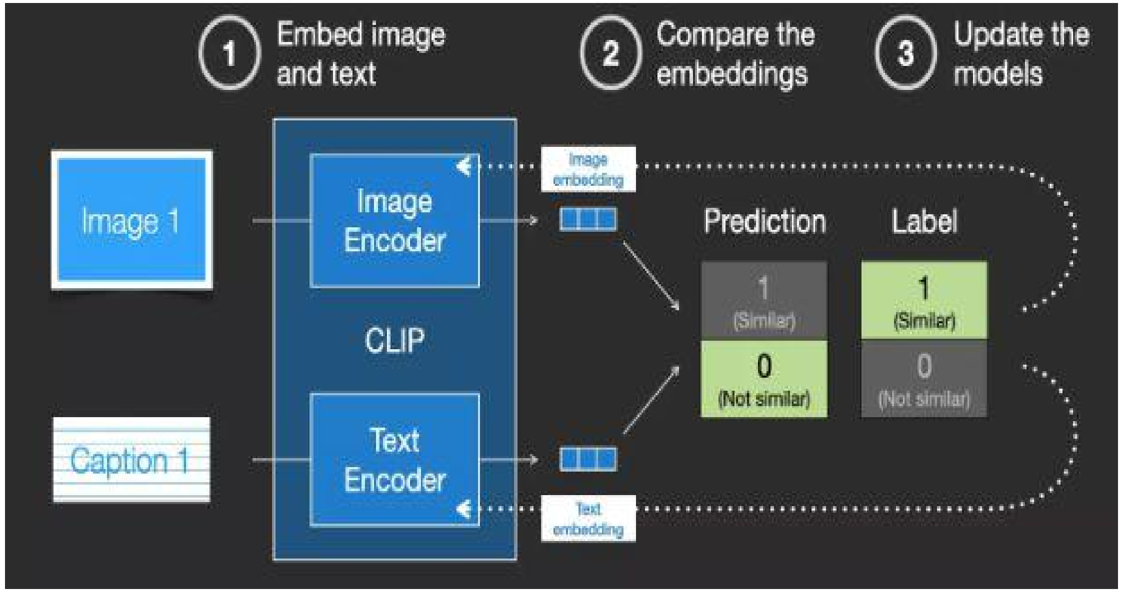
主打参数占比大



text embedding

维度匹配：以OpenAI所开源的CLIP模型clip-vit-large-patch14为例，CLIP的text encoder是一个只有encoder模块的transformer模型，层数为12，特征维度为768。

长度约束：对于输入text，送入CLIP text encoder后得到最后的hidden states（即最后一个transformer block得到的特征），其特征维度大小为77x768（77是token的数量）
输入text的tokens数量超过77后，将进行截断，如果不足则进行paddings。



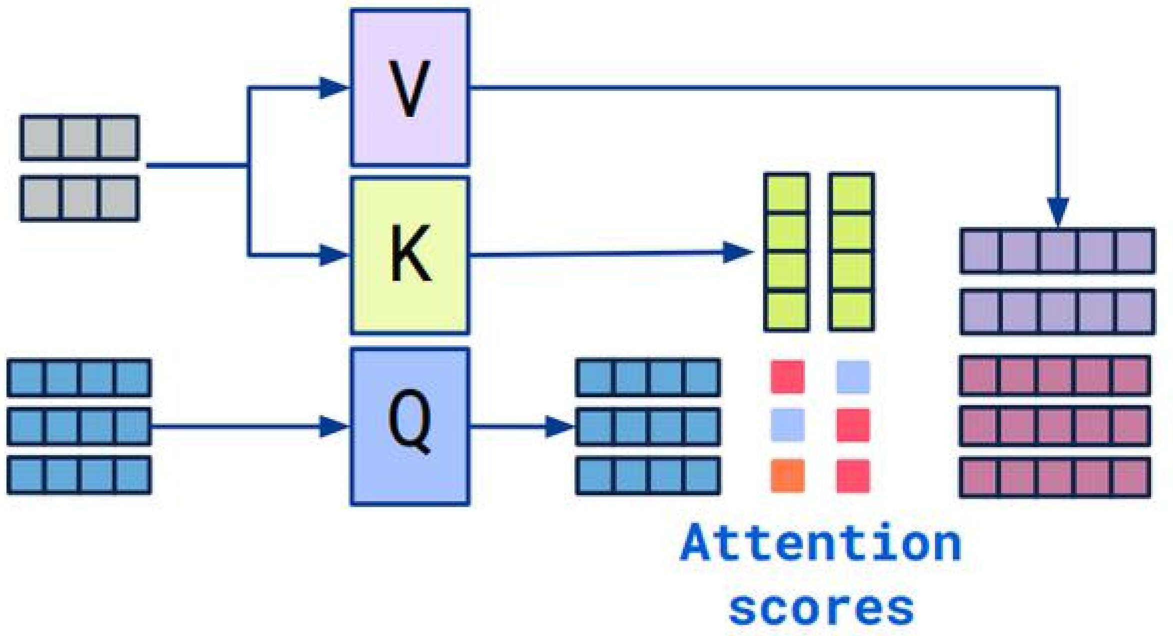
cross attention

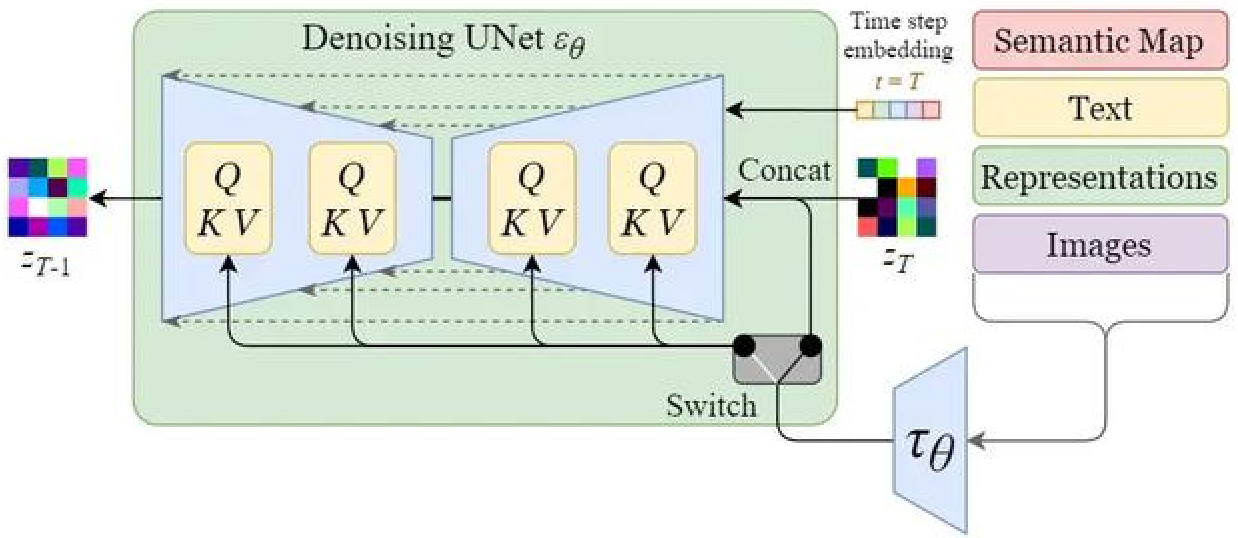
混合两个不同嵌入序列的注意机制

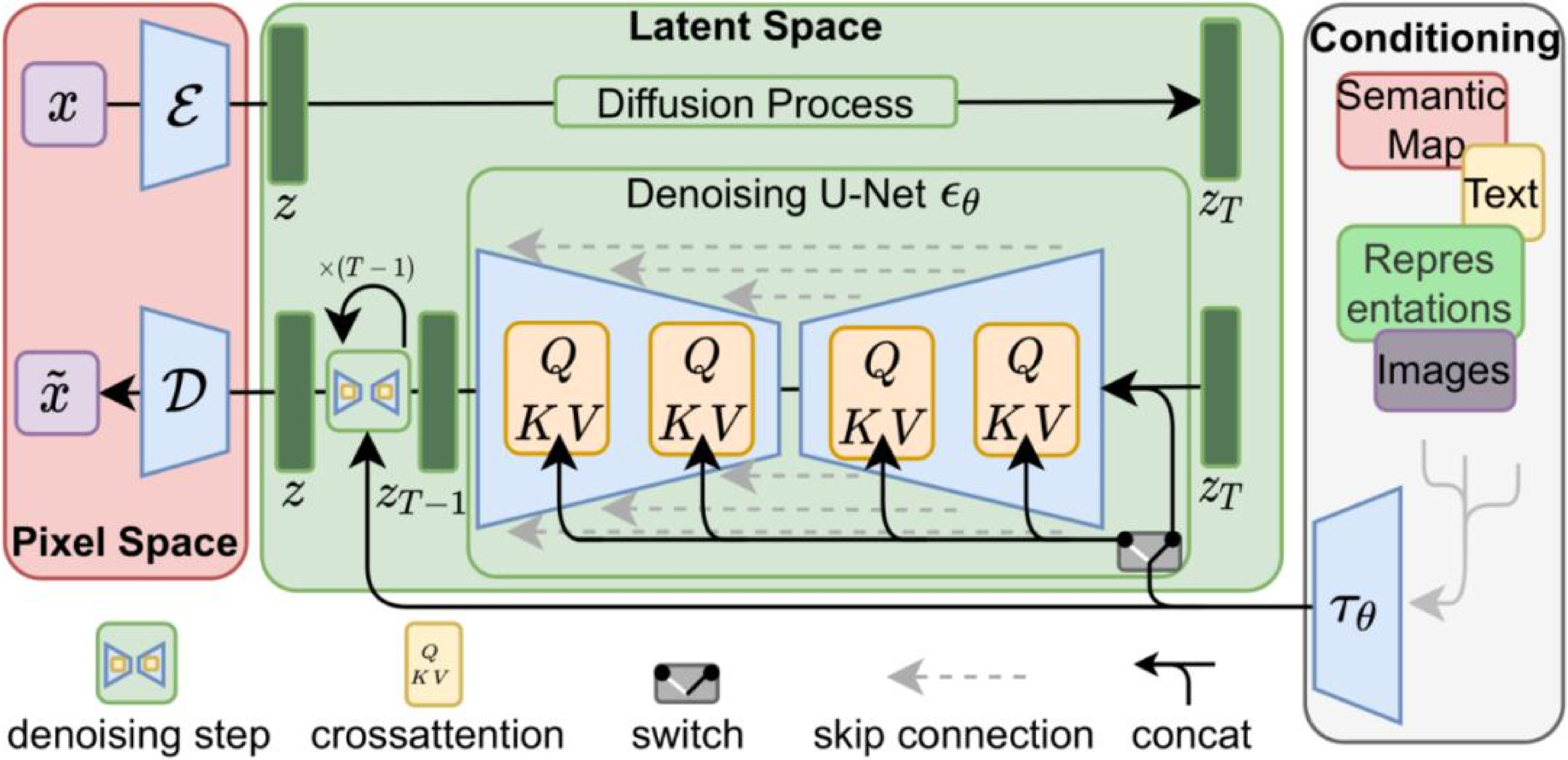
两个序列必须具有相同的维度

两个序列可以是不同的形式（文本、图像、声音）

concat（三角函数）







Latent Space
Conditioning
Pixel Space
x
$\mathcal{E}$
z
Diffusion Process
z_T
Semantic Map
Text
Repres entations
Images
Denoising U-Net ϵ_θ
$\times(T-1)$
$\tilde{x}$
$\mathcal{D}$
z
z_{T-1}
Q
$K\,V$
z_T
τ_θ
denoising step
crossattention
switch
skip connection
concat

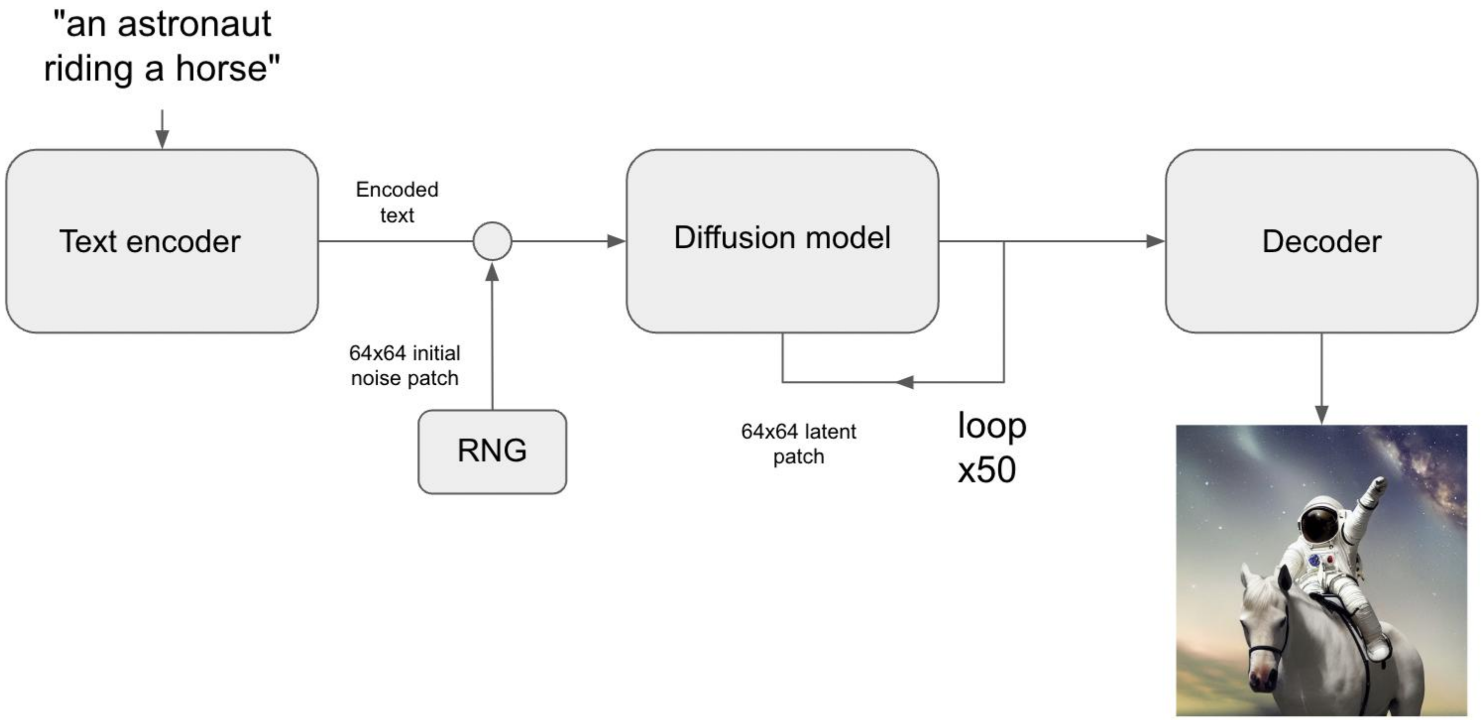
推理流程

根据输入text用text encoder提取text embeddings，

初始化随机噪音noise（维度匹配），

送入U-net中生成去噪后的latent，

最后送入decoder得到生成的图像。



已有改进

SD本身

Fine-tuning: Textual Inversion, Aesthetic Embedding, Dreambooth, Hypernetworks, LoRA, ControlNet

Keras实现

XLA compilation via jit_compile =True 进行 XLA 编译

Support for mixed precision computation 混合精度运算

混合精度 + XLA 编译设置

```
keras.mixed_precision.set_global_policy("mixed_float16")
model = keras_cv.models.StableDiffusion(jit_compile=True)
images = model.text_to_image(
    "Teddy bears conducting machine learning research",
    batch_size=3,
)
```

架构小结

Autoencoder

diffusion

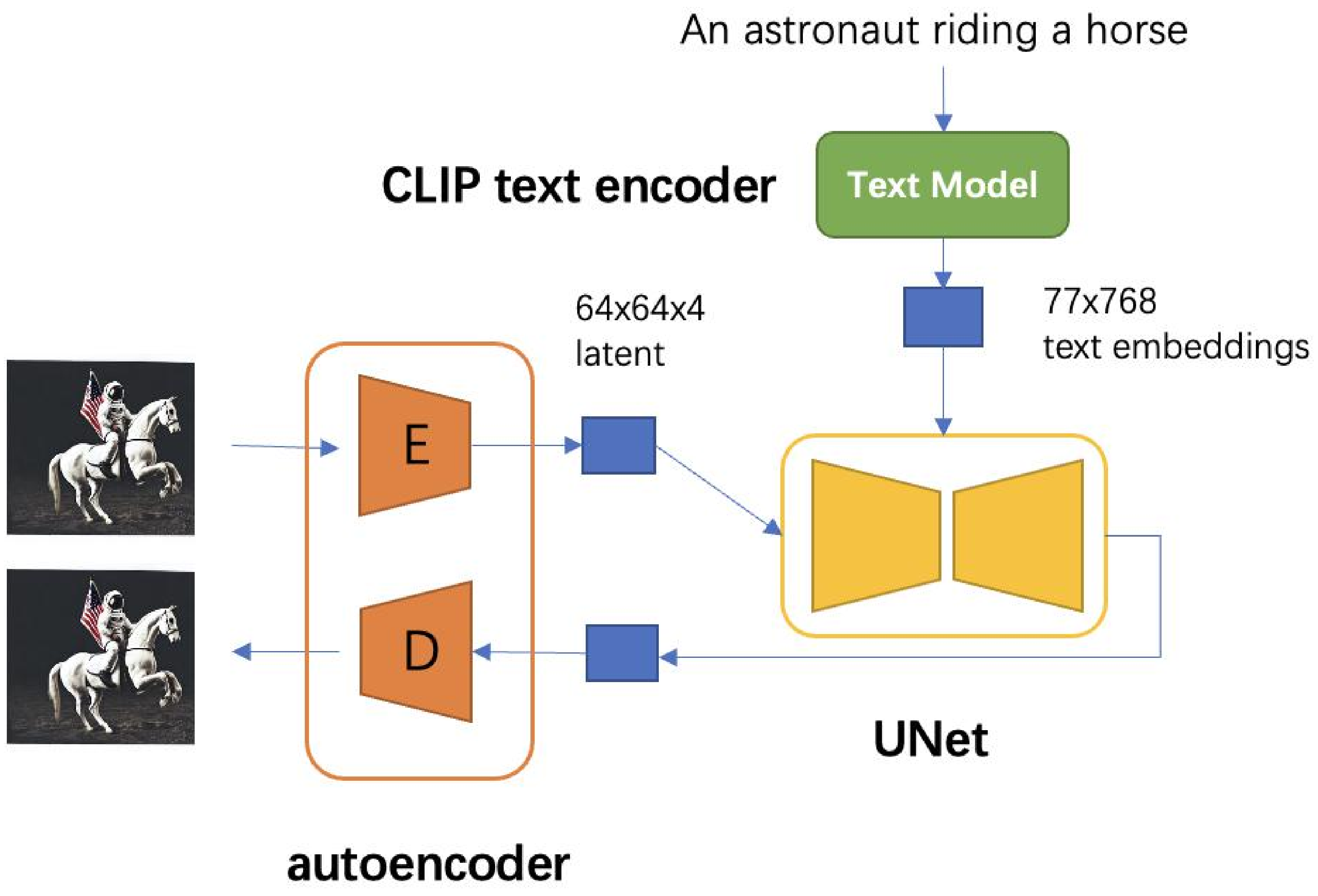
U-net

text embedding

cross attention

启示:

- 集成创新
- code的内涵并不chea



涌现能力（Emergent Abilities）

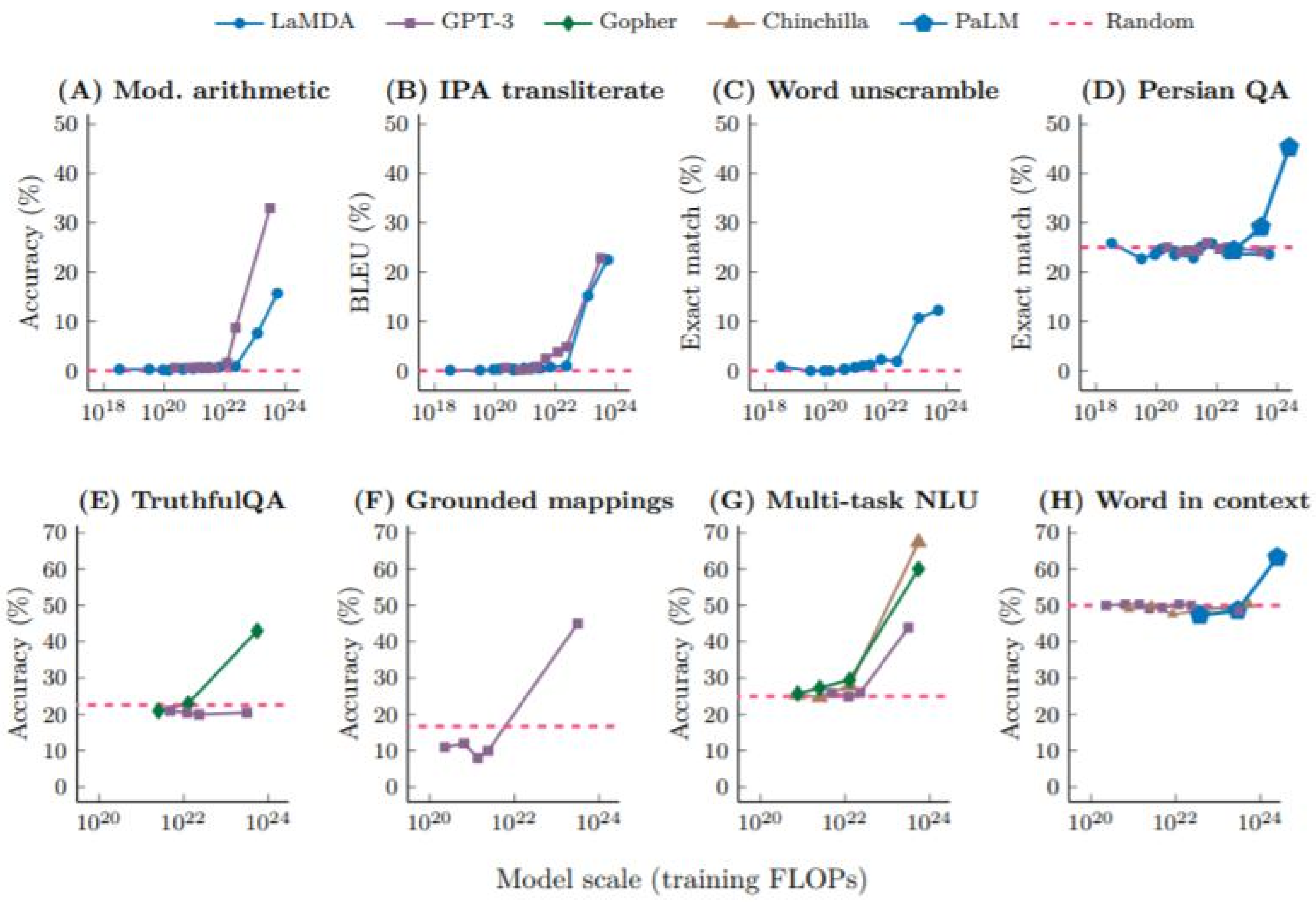
模型大

+

语料库大

=

厚积薄发



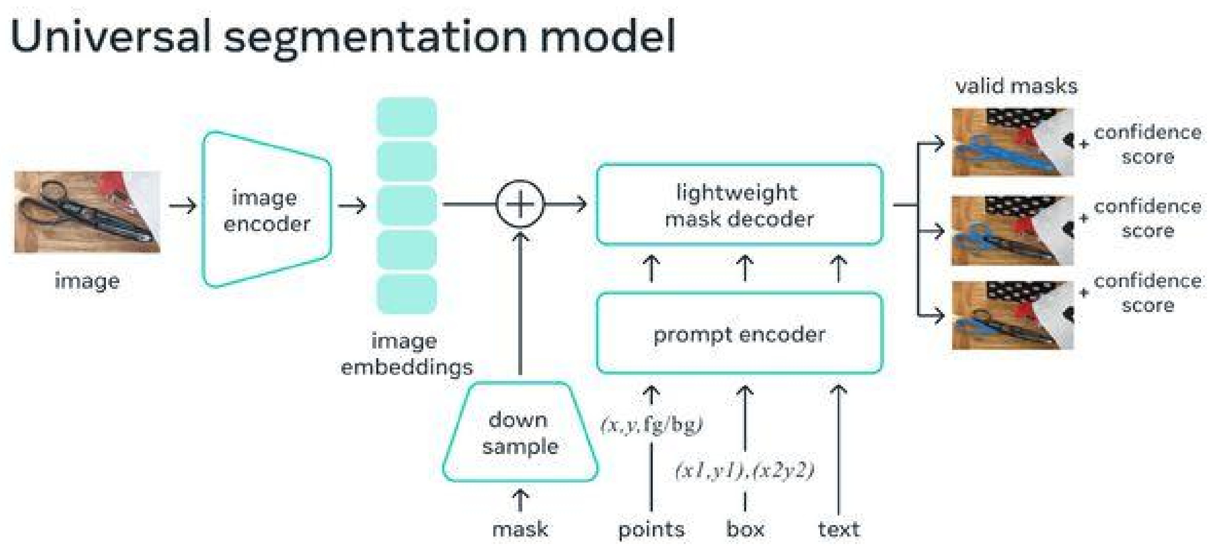
可能的发展方向

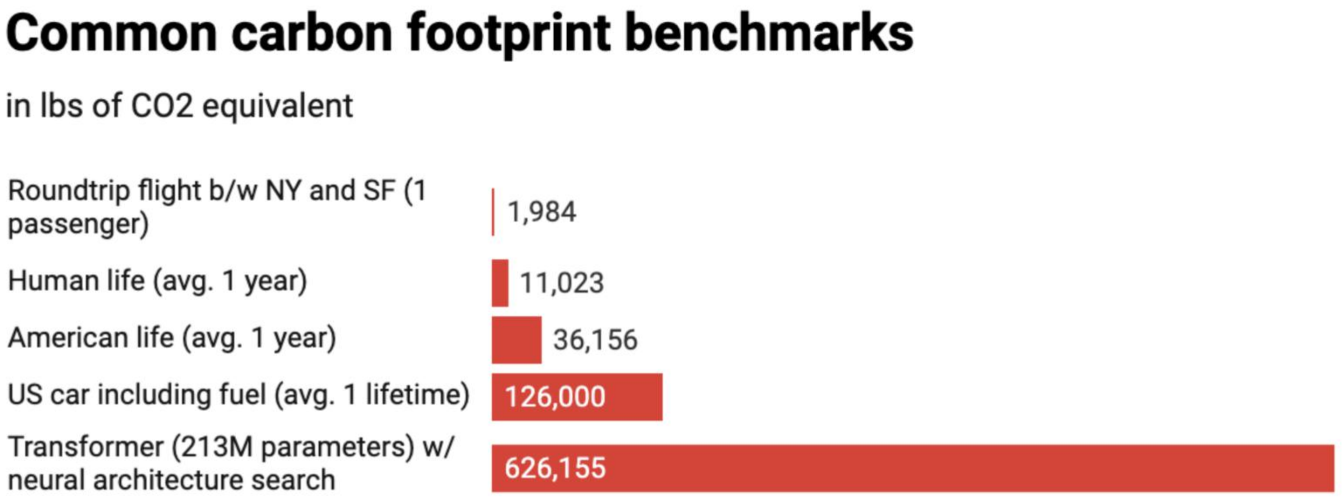
通用

- （可以实现的）模型更大？
- （实现某个功能的）模型更小？
- 实时在线获取新知识？
- 输入字数更多？
- 输入数据格式更丰富？
- 节能？
- 同时发生！

专用

- 许多细分领域





郑重提示

利用生成式人工智能生成的内容应当体现社会主义核心价值观

--《生成式人工智能服务管理办法（征求意见稿）》

http://www.cac.gov.cn/2023-04/11/c_1682854275475410.htm tf.yyds.thanks!