- 注意事项：
  - 这**不是**一个逐行代码讲解的视频
  - 只讲关键部分的代码实现，可以一定程度上减轻复现或者读懂代码的难度
  - 深入理解DDPM
  - DDPM不算是很基础的深度学习内容，需要有基本的深度学习的知识和Pytorch的使用经验
- 比较简单的玩具代码：https://github.com/abarankab/DDPM 主要讲解
- 比较实用的代码：https://github.com/Janspiry/Image-Super-Resolution-via-Iterative-Refinement

- 为了复现DDPM，我们需要考虑哪些细节问题?
- （1）alpha、beta这些预先定义好的参数怎么获得? α_t β_t
- （2）训练过程? ✓
- （3）推理过程? ✓
- （4）UNet的结构?

$z = UNet(x_t, t)$

↑ 也可以换成别的网络 但UNet开销小

β_t $10^{-2} \rightarrow 2\times10^{-4}$

$T=2000$

- （1）alpha、beta这些预先定义好的参数怎么获得?
- https://github.com/abarankab/DDPM/blob/main/ddpm/diffusion.py

```
def generate_linear_schedule(T, low, high):
    return np.linspace(low, high, T)
```

```
alphas = 1.0 - betas
alphas_cumprod = np.cumprod(alphas)
```

α_t 连乘

把 np.array 转化成 tensor

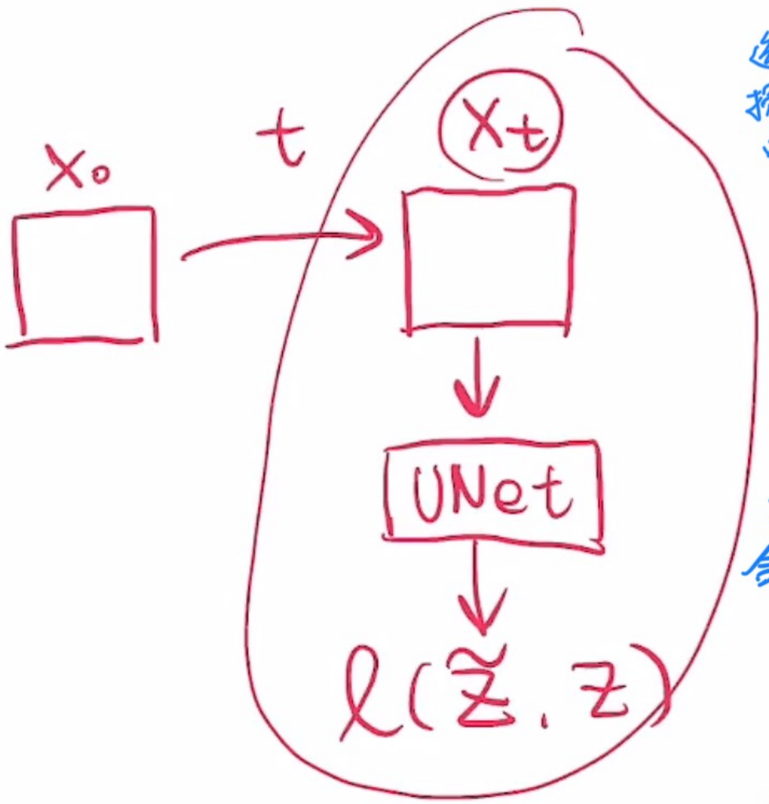
```
to_torch = partial(torch.tensor, dtype=torch.float32)
```

α_t ← t 索引

变成 tensor 保存起来

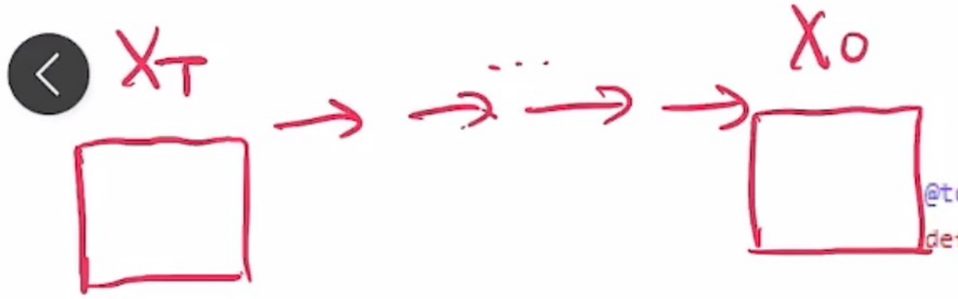
```
self.register_buffer("betas", to_torch(betas))
self.register_buffer("alphas", to_torch(alphas))
self.register_buffer("alphas_cumprod", to_torch(alphas_cumprod))

self.register_buffer("sqrt_alphas_cumprod", to_torch(np.sqrt(alphas_cumprod)))
self.register_buffer("sqrt_one_minus_alphas_cumprod", to_torch(np.sqrt(1 - alphas_cumprod)))
self.register_buffer("reciprocal_sqrt_alphas", to_torch(np.sqrt(1 / alphas)))

self.register_buffer("remove_noise_coeff", to_torch(betas / np.sqrt(1 - alphas_cumprod)))
self.register_buffer("sigma", to_torch(np.sqrt(betas)))
```

- （2）训练过程?
- https://github.com/abarankab/DDPM/blob/main/ddpm/diffusion.py




```
def get_losses(self, x, t, y):
    noise = torch.randn_like(x)

    perturbed_x = self.perturb_x(x, t, noise)
    estimated_noise = self.model(perturbed_x, t, y)

    if self.loss_type == "l1":
        loss = F.l1_loss(estimated_noise, noise)
    elif self.loss_type == "l2":
        loss = F.mse_loss(estimated_noise, noise)

    return loss

def forward(self, x, y=None):
    b, c, h, w = x.shape
    device = x.device

    if h != self.img_size[0]:
        raise ValueError("image height does not match diffusion parameters")
    if w != self.img_size[0]:
        raise ValueError("image width does not match diffusion parameters")

    t = torch.randint(0, self.num_timesteps, (b,), device=device)
    return self.get_losses(x, t, y)
```

- （3）推理过程?
- https://github.com/abarankab/DDPM/blob/main/ddpm/diffusion.py



采样方式

```python
@torch.no_grad()
def remove_noise(self, x, t, y, use_ema=True):
    if use_ema:
        return (
            (x - extract(self.remove_noise_coeff, t, x.shape) * self.ema_model(x, t, y)) *
            extract(self.reciprocal_sqrt_alphas, t, x.shape)
        )
    else:
        return (
            (x - extract(self.remove_noise_coeff, t, x.shape) * self.model(x, t, y)) *
            extract(self.reciprocal_sqrt_alphas, t, x.shape)
        )
```

去噪 $\tilde{z}$

```python
@torch.no_grad()
def sample(self, batch_size, device, y=None, use_ema=True):
    if y is not None and batch_size != len(y):
        raise ValueError("sample batch size different from length of given y")

    x = torch.randn(batch_size, self.img_channels, *self.img_size, device=device)

    for t in range(self.num_timesteps - 1, -1, -1):
        t_batch = torch.tensor([t], device=device).repeat(batch_size)
        x = self.remove_noise(x, t_batch, y, use_ema)

        if t > 0:
            x += extract(self.sigma, t_batch, x.shape) * torch.randn_like(x)

    return x.cpu().detach()
```

正态分布随机采样 X_T

X_t

一共做T步上推过程

转换成 tensor

得到 x_{t-1} 的均值部分

X_{t-1}

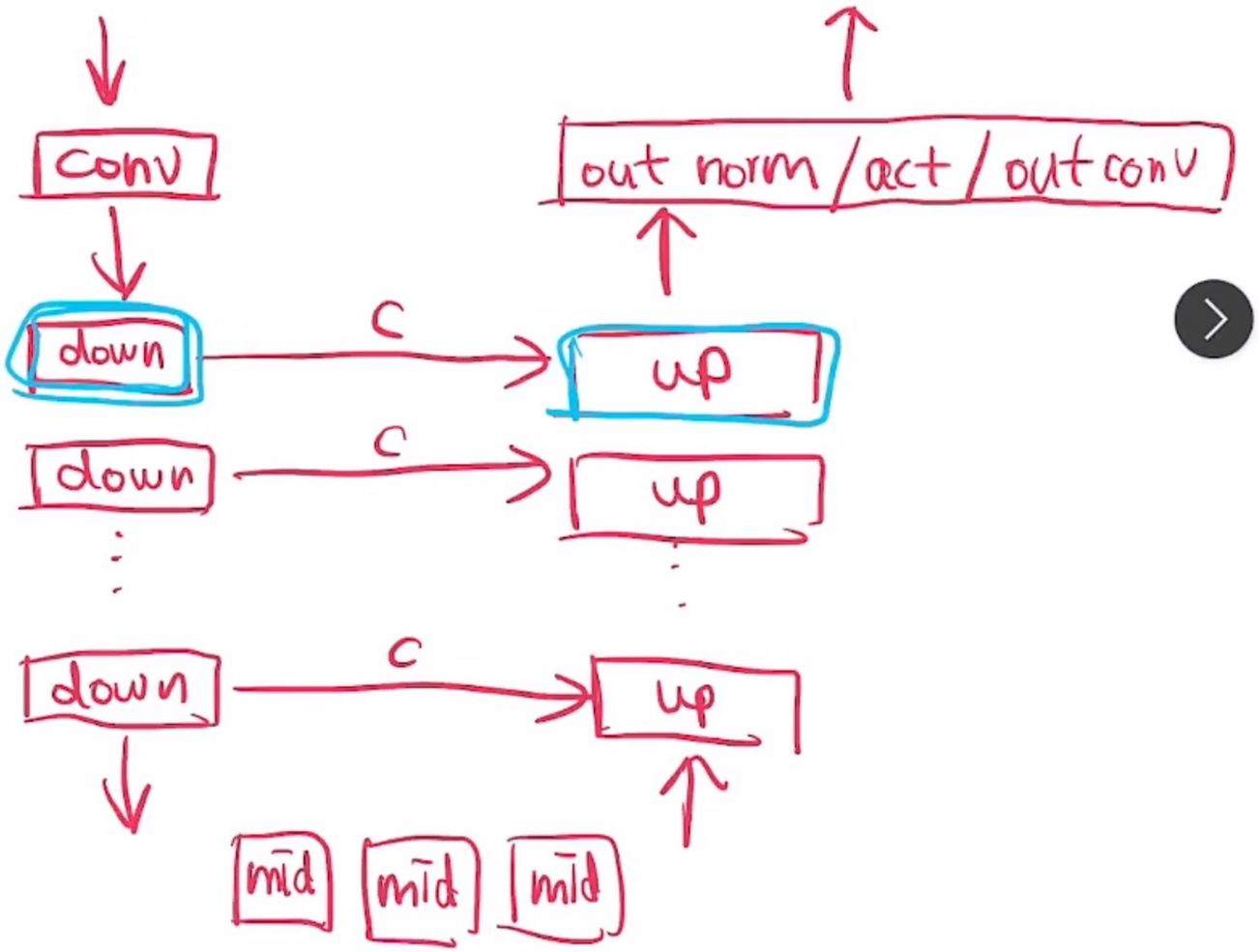
t>0 要加噪声

- （4）UNet的结构?
- https://github.com/abarankab/DDPM/blob/main/ddpm/unet.py

```
x = self.init_conv(x)

skips = [x]

for layer in self.downs:
    x = layer(x, time_emb, y)
    skips.append(x)

for layer in self.mid:
    x = layer(x, time_emb, y)

for layer in self.ups:
    if isinstance(layer, ResidualBlock):
        x = torch.cat([x, skips.pop()], dim=1)
    x = layer(x, time_emb, y)

x = self.activation(self.out_norm(x))
x = self.out_conv(x)
```

concate

这是关键



- （4）UNet的结构?
- https://github.com/abarankab/DDPM/blob/main/ddpm/unet.py

```
x = self.init_conv(x)

skips = [x]

for layer in self.downs:
    x = layer(x, time_emb, y)
    skips.append(x)

for layer in self.mid:
    x = layer(x, time_emb, y)

for layer in self.ups:
    if isinstance(layer, ResidualBlock):
        x = torch.cat([x, skips.pop()], dim=1)
    x = layer(x, time_emb, y)

x = self.activation(self.out_norm(x))
x = self.out_conv(x)
```

向前去找

```
if self.time_mlp is not None:
    if time is None:
        raise ValueError("time conditioning was specified but tim is not passed")

    time_emb = self.time_mlp(time)
else:
    time_emb = None
```

t 20 30 or 2000

mlp 输出表示时间的向量

- （4）UNet的结构?
- https://github.com/abarankab/DDPM/blob/main/ddpm/unet.py

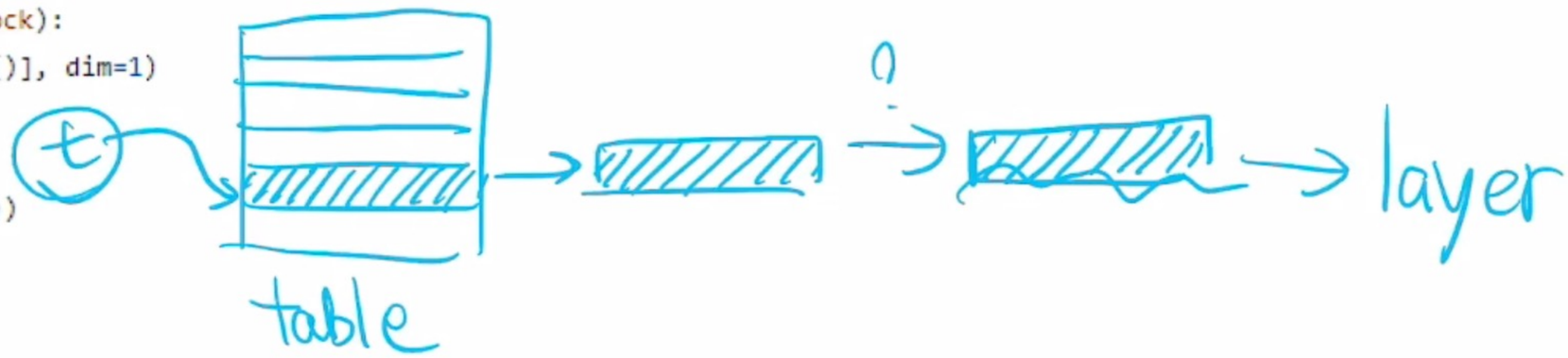
```
x = self.init_conv(x)

skips = [x]

for layer in self.downs:
    x = layer(x, time_emb, y)
    skips.append(x)

for layer in self.mid:
    x = layer(x, time_emb, y)

for layer in self.ups:
    if isinstance(layer, ResidualBlock):
        x = torch.cat([x, skips.pop()], dim=1)
    x = layer(x, time_emb, y)

x = self.activation(self.out_norm(x))
x = self.out_conv(x)
```

是怎么送进模型的呢?

```
if self.time_mlp is not None:
    if time is None:
        raise ValueError("time conditioning was specified but tim is not passed")

    time_emb = self.time_mlp(time)
else:
    time_emb = None
```

```
self.time_mlp = nn.Sequential(
    PositionalEmbedding(base_channels, time_emb_scale),
    nn.Linear(base_channels, time_emb_dim),
    nn.SiLU(),
    nn.Linear(time_emb_dim, time_emb_dim),
) if time_emb_dim is not None else None
```

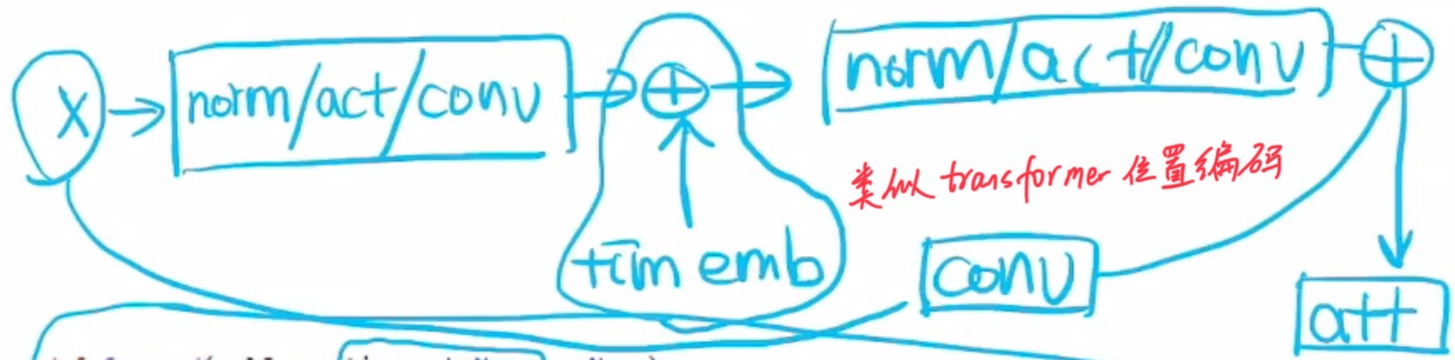
非线性过程



- （4）UNet的结构?
- https://github.com/abarankab/DDPM/blob/main/ddpm/unet.py

```
x = self.init_conv(x)

skips = [x]

for layer in self.downs:
    x = layer(x, time_emb, y)
    skips.append(x)

for layer in self.mid:
    x = layer(x, time_emb, y)

for layer in self.ups:
    if isinstance(layer, ResidualBlock):
        x = torch.cat([x, skips.pop()], dim=1)
    x = layer(x, time_emb, y)

x = self.activation(self.out_norm(x))
x = self.out_conv(x)
```

RB 残差模块

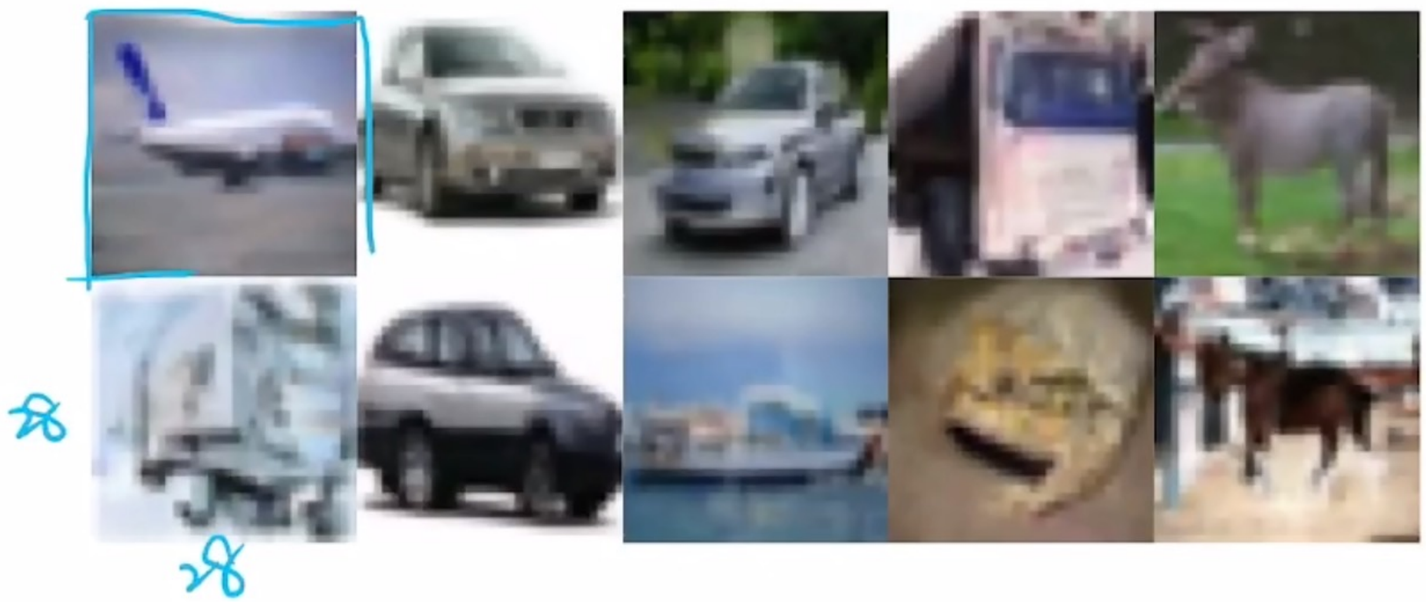



```
def forward(self, x, time_emb=None, y=None):
    out = self.activation(self.norm_1(x))
    out = self.conv_1(out)

    if self.time_bias is not None:
        if time_emb is None:
            raise ValueError("time conditioning was specified but time_emb is not passed")
        out += self.time_bias(self.activation(time_emb))[:, :, None, None]

    if self.class_bias is not None:
        if y is None:
            raise ValueError("class conditioning was specified but y is not passed")

        out += self.class_bias(y)[:, :, None, None]

    out = self.activation(self.norm_2(out))
    out = self.conv_2(out) + self.residual_connection(x)
    out = self.attention(out)

    return out
```

加入 time_emb

自注意力

➤ CIFAR10训练150K次迭代的随机生成结果



- 建议：
  - 想要掌握DDPM，看代码，跑实验，出结果
  - 想要看更好的生成结果，参考https://github.com/Janspiry/Image-Super-Resolution-via-Iterative-Refinement
  - 看不懂代码？先跑实验再说

- 更新计划（大概吧）：
  - 第一章：DDPM介绍
  - 第二章：DDPM的应用
  - 第三章：score-based model介绍
  - 第四章：score-based model的应用

SR3 ddpm DDIM Diffusion CLIP ILVR SDEdit MRI ...

扩散模型
- score-based ✓ 偏理论多
- DDPM ✓ 偏应用多