Bringing Old Photos Back to Life

> 2020_CVPR_Bringing-Old-Photos-Back-to-Life.pdf arxiv 有CVPR，TPAMI 版本的pdf，TPAMI 版本的算法原理更多一些
> github repo
> 博客参考

Background

> 介绍老照片修复背景

老照片修复相比于 image restoration 任务更加复杂，可以简单归纳为一下两点：

1. 没有 degradation model 来模拟老照片的瑕疵（无法简单地模仿老照片的瑕疵，数据少）
2. 老照片为 一些列 degradations 的组合，需要不同的方法去修复。并且老照片瑕疵可以归纳为 **unstructured defects** (墨水，褪色)；**structured defects** (划痕)

并且老照片数据，**有对应的修复结果（clean image Ground Truth）的数据很少**，影响训练。以往的工作大多使用干净图像，对老照片进行合成来造数据，但会存在 **domain gap 的问题（造的数据和真实老照片差距很大）**：

- Prior DL methods 修复功能有限

  对划痕类似的 localized defects 检测，再用 inpainting 方法修复。缺点是无法修复 墨水，褪色等 spatially-uniform defects。

- DL-based image restoration methods 修复效果不好

  在合成图像上处理，但无法处理涂色 & inpaint 混合的情况。存在 domain gap，导致图像修复结果看起来还是很旧。

> 文章方法核心思想

作者借鉴 image translation 任务思想，将 `合成图 X、真实图 Y，老照片 R` 都映射到 latent space 来进行修复。将合成图和真实老照片通过共享的 *VAE* 映射到同一个 *latent space*，通过对 *VAE Encoder* 进行限制（*Gaussian Distribution & GAN Discriminator*），让合成图和老照片的 *latent code* 尽可能接近。然后在 latent space 学习合成图 X 到真实图 Y latent code 的映射，最后将映射结果通过 Decoder 恢复出真实图。如果老照片和合成图的 latent code 分布接近，则可以借鉴前面学到到的 mapping (合成图->真实图的映射)，得到修复结果的 latent code，再恢复出原图。

Contributions

- 映射到 latent space，使用合成图训练。降低 domain gap，避免使用老照片 GT （数据很少）进行训练
- 支持多种 degradation 混合的情况，能有效去除。

Related Work

- [ ] 2003_EUROCON_Towards-the-automated-restoration-of-old-photographic-prints--a-survey.pdf
- [ ] 2022_master_report_DSTT-MARB--Multi-scale-Attention-Based-Spatio-Temporal-Transformers-for-Old-Film-Restoration.pdf

- [ ] VAE
- [ ] baseline
  - [ ] Pix2Pix
  - [ ] CycleGAN
  - [ ] Deep image prior

- 矩阵乘法总结
  - Hadamard Product >> 矩阵对应元素乘积
  - Kronecker product >> 两个任意大小的矩阵间的运算

    如果A是一个$m\times n$的矩阵，而B是一个$p\times q$的矩阵，**克罗内克积**则是一个$mp\times nq$的分块矩阵
- Wasserstein Distance

  衡量概率分布差异，能够很自然地度量离散分布和连续分布之间的距离 & 给出了距离的度量，而且给出如何把一个分布变换为另一分布的方案 & 能够连续地把一个分布变换为另一个分布，在此同时，能够保持分布自身的几何形态特征；

VAE

> 苏剑林老师 VAE 讲解

nonlocal-block

methods

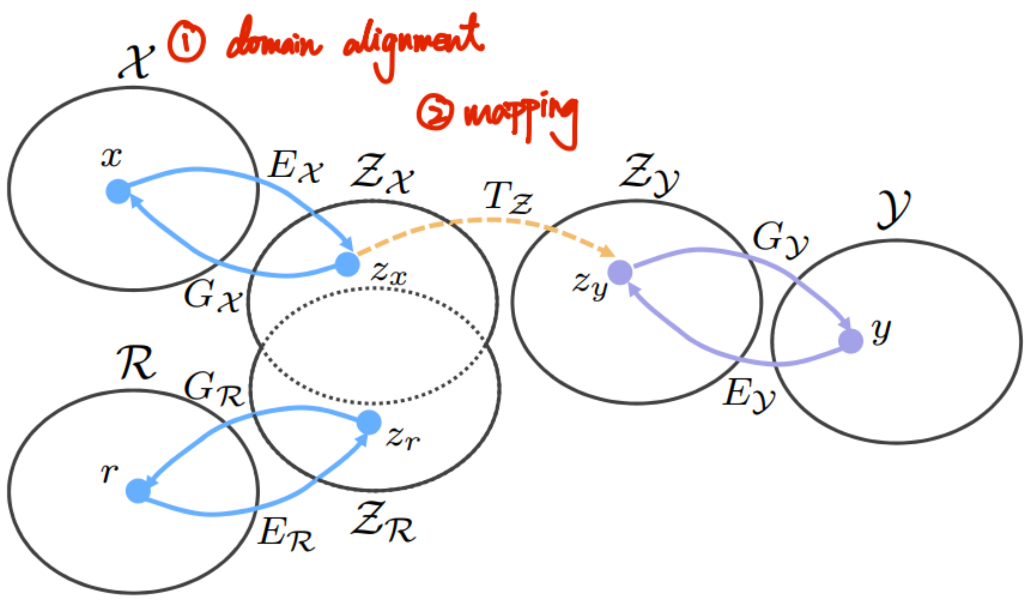
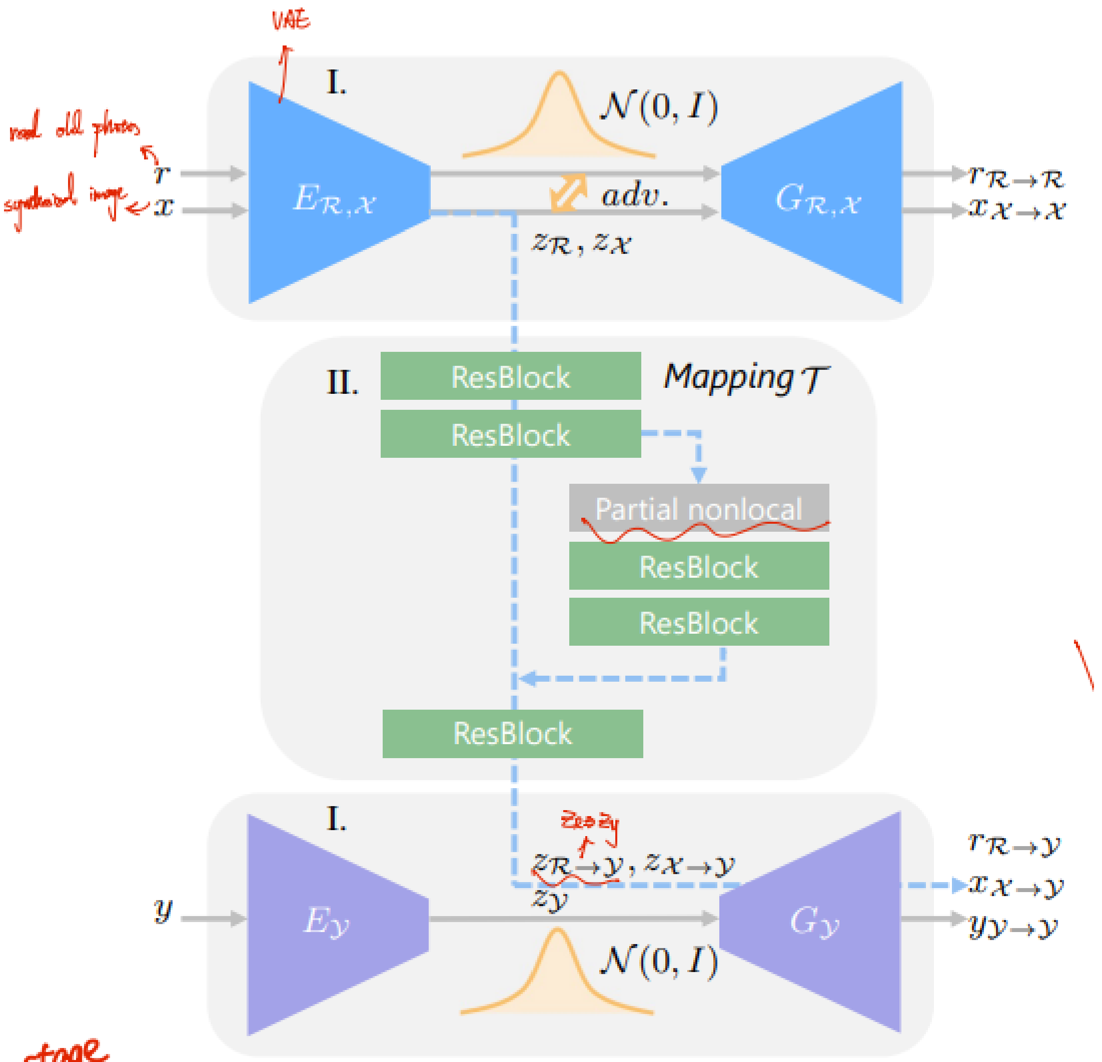


Figure 2: **Illustration of our translation method with three domains.**



2 stage

Figure 3: **Architecture of our restoration network.** (I.) We first train two VAEs: VAE$_1$ for images in real photos $r \in \mathcal{R}$ and synthetic images $x \in \mathcal{X}$, with their domain gap closed by jointly training an adversarial discriminator; VAE$_2$ is trained for clean images $y \in \mathcal{Y}$. With VAEs, images are transformed to compact latent space. (II.) Then, we learn the mapping that restores the corrupted images to clean ones in the latent space.

> 图中的符号解释
> `R`: 真实的老照片; `y`: 无损图像; `x`: 从 `y` 退化（人工退化后）的合成老照片图，x 和 y 是合成得到的退化-真值图像对。 `z_y`, `z_x`, `z_r` 为无损图 y，对应合成图 x 和老照片 r 的 `latent code`

核心思想：映射到 latent space，借鉴 x-> y 的映射来修复老照片 r ☆

- 作者将**希望将 R,X 映射得到的 latent code 接近** >> Domain alignment in the VAE latent space

  因此作者提出的方法为 2 stage 的，要先训好 x,r 的 VAE encoder；再去训 mapping 和 decoder
- 借鉴 `z_y`, `z_x` 之间的映射，作用于 `z_r` >> Restoration through latent mapping
- 提出 partial nonlocal block 在 mapping 修复的过程中 >> Multiple degradation restoration

Domain alignment

- 再从 GAN discriminator 角度使得 z_r z_x 接近

$$
\begin{aligned}
\mathcal{L}_{\mathrm{VAE}_1}(r) = {} & \mathrm{KL}(\underbrace{E_{\mathcal{R},\mathcal{X}}(z_r|r)}||\mathcal{N}(0, I)) \quad \text{求 latent code 分布距离} \\
& + \alpha \mathbb{E}_{z_r \sim E_{\mathcal{R},\mathcal{X}}(z_r|r)} \left[\| G_{\mathcal{R},\mathcal{X}}(\underbrace{r_{\mathcal{R}\to\mathcal{R}}}_{\text{output}}|\underbrace{z_r}_{\text{input}}) - r \|_1 \right] \\
& + \underbrace{\mathcal{L}_{\mathrm{VAE}_1,\mathrm{GAN}}(r)}_{\text{LSGAN loss} \Rightarrow \text{realism}}
\end{aligned}
\tag{2}
$$

γ

differentiates $\mathcal{Z}_\mathcal{R}$ and $\mathcal{Z}_\mathcal{X}$, whose loss is defined as,

$$
\begin{aligned}
\text{max} \quad \mathcal{L}^{\text{latent}}_{\mathrm{VAE}_1,\mathrm{GAN}}(r, x) = {} & \mathbb{E}_{x \sim \mathcal{X}}[\underbrace{D_{\mathcal{R},\mathcal{X}}(E_{\mathcal{R},\mathcal{X}}(x))}_{\to 1}{}^2] \\
& + \mathbb{E}_{r \sim \mathcal{R}}[(1 - \underbrace{D_{\mathcal{R},\mathcal{X}}(E_{\mathcal{R},\mathcal{X}}(r))}_{\to 0})^2].
\end{aligned}
\tag{3}
$$

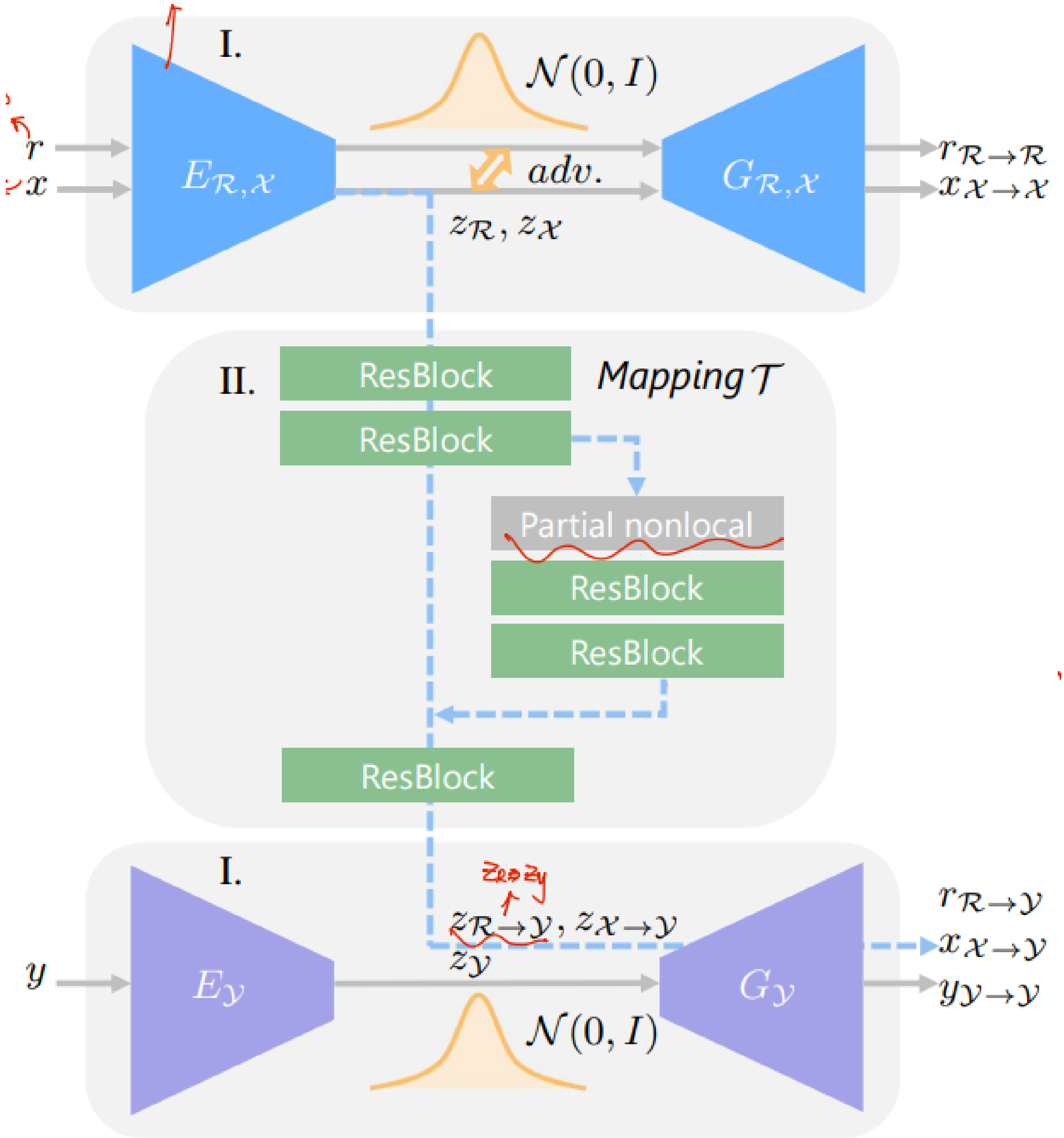
Meanwhile, the encoder $E_{\mathcal{R},\mathcal{X}}$ of VAE_1 tries to fool the discriminator with a contradictory loss to ensure that $\mathcal{R}$ and $\mathcal{X}$ are mapped to the same space. Combined with the latent adversarial loss, the total objective function for VAE_1 becomes, domain gap ↓

① 映射到 Gaussian Distribution 要接近 ② GAN Discriminator 分辨 zr, zx

$$
\min_{E_{\mathcal{R},\mathcal{X}}, G_{\mathcal{R},\mathcal{X}}} \max_{D_{\mathcal{R},\mathcal{X}}} \mathcal{L}_{\mathrm{VAE}_1}(r) + \mathcal{L}_{\mathrm{VAE}_1}(x) + \mathcal{L}^{\text{latent}}_{\mathrm{VAE}_1,\mathrm{GAN}}(r, x).
$$

Restoration through latent mapping

在 latent space 通过映射的方式实现修复。老照片的退化有比较复杂，需要利用更大范围的信息进行填充以保证全局图像结构的一致性。因此作者**提出了一个全局分支，该分支包含一个partial nonlocal block和两个残差块**，如下图所示



- partial nonlocal block 作为 global branch 去修复待 inpainting 区域

  在 ECCV2018 提出的 nonlocal block 改造而来：借用完好区域（源区域）的信息来修补损坏的区域（目标区域）。

  - $S_{i,j}$ 表示中间层输入的 Feature Map F 中**完好区域元素** i, j 的亲和度。文章中用 U-net 去分割出裂痕等 mask

    最后模块输出 O 理解：对于受损元素 i，用亲和的完好区域 j 元素来加权修复 ☆

$$s_{i,j} = (1 - m_j) f_{i,j} / \sum_{\forall k} (1 - m_k) f_{i,k}, \tag{7}$$

where,

$$f_{i,j} = \exp(\theta(F_i)^T \cdot \phi(F_j)) \tag{8}$$

gives the pairwise affinity with embedded Gaussian. θ and ϕ project F to Gaussian space for affinity calculation. According to the affinity $s_{i,j}$ that considers the holes in the mask, the partial nonlocal finally outputs

$$O_i = \nu \left(\sum_{\forall j} s_{i,j} \mu(F_j) \right), \tag{9}$$

- U-net mask 分割

  作者自己标注了 783 张有划痕的老照片，去 finetune U-net(现在合成的瑕疵上训)

> 优化

1. 优化 mapping 结果 `T(z_x)` 与 `z_y` 距离
2. LSGAN >> realism
3. perceptual loss

Let $r_{\mathcal{R}\to\mathcal{Y}}$, $x_{\mathcal{X}\to\mathcal{Y}}$ and $y_{\mathcal{Y}\to\mathcal{Y}}$ be the final translation outputs for r, x and y, respectively. At this stage, we solely train the parameters of the latent mapping network $\mathcal{T}$ and fix the two VAEs. The loss function $\mathcal{L}_{\mathcal{T}}$, which is imposed at both the latent space and the end of generator $G_{\mathcal{Y}}$, consists of three terms,

zx, zy 1范数距离 realism perceptual loss

$$\mathcal{L}_{\mathcal{T}}(x, y) = \lambda_1 \mathcal{L}_{\mathcal{T},\ell_1} + \mathcal{L}_{\mathcal{T},\text{GAN}} + \lambda_2 \mathcal{L}_{\text{FM}} \tag{5}$$

where, the latent space loss, ① $\mathcal{L}_{\mathcal{T},\ell_1} = \mathbb{E} \left\| \mathcal{T}(z_x) - z_y \right\|_1$, penalizes the ℓ_1 distance of the corresponding latent codes. We introduce the adversarial loss ② $\mathcal{L}_{\mathcal{T},\text{GAN}}$, still in the form of LSGAN [47], to encourage the ultimate translated synthetic image $x_{\mathcal{X}\to\mathcal{Y}}$ to look real. Besides, we introduce

- 融合一下全局操作修复 inpainting，局部操作修复原本就完好的区域

$$F_{fuse} = (1 - m) \odot \rho_{\text{local}}(F) + m \odot \rho_{\text{global}}(O), \qquad (10)$$

对应元素乘积

where operator $\odot$ denotes Hadamard product, and ρ_{local} and ρ_{global} denote the nonlinear transformation of residual blocks in two branches. In this way, the two branches con-

> ❓ local 修复和 全局 inpaint 结果直接相加?

Experiment

- 数据
  - training set Pascal VOC
  - 在 DIV2K 上合成数据，真实老照片数据 test
  - 自己收集了 5718 张老照片但没 release 出来

  训练时随机 crop 256x256 区域
- 量化比较

  PSNR，SSIM，FID
- 效果比较

  由于无GT，直接看图
- User study >> 找人看，几个方法的结果排名

Summary✨

> learn what & how to apply to our task

Code

- install https://github.com/microsoft/Bringing-Old-Photos-Back-to-Life/issues/248#issuecomment-1344901139

Train Domain_A

> Data Preparation `class UnPairOldPhotos_SR(BaseDataset):  ## Synthetic + Real Old`

随机 50% 概率取 Pascal VOC 或真实老照片。对于合成噪声

1. `cv2.GaussianBlur` P=70%
2. 加噪声 `synthesize_gaussian, synthesize_speckle, synthesize_salt_pepper` 转化为 numpy 采样 gaussian
3. 降低分辨率 + 在还原原分辨率

```
new_w=random.randint(int(w/2),w)
new_h=random.randint(int(h/2),h)
img=img.resize((new_w,new_h),Image.BICUBIC)
img=img.resize((w,h),Image.NEAREST)  # P=0.5
```

4. JPEG 压缩

```
def convertToJpeg(im,quality):
    with BytesIO() as f:
        im.save(f, format='JPEG',quality=quality)
        f.seek(0)
        return Image.open(f).convert('RGB')
```

- <10% 概率，转为灰度图，再转回 RGB
- resize 到 256
- flip & normalize

```
input_dict = {'label': A_tensor, 'inst': is_real_old,  # Union[0,1]
              'image': A_tensor,
              'feat': feat_tensor,  # 0
              'path': path}
return input_dict
```

Domain alignment loss

`input_dict['label']` 进入 Encoder 获取 hidden tensor, 加上噪声还原 >> 作为 fake 数据给判别器

```
hiddens = self.netG.forward(input_concat, 'enc')
noise = Variable(torch.randn(hiddens.size()).cuda(hiddens.data.get_device()))
# This is a reduced VAE implementation where we assume the outputs are
multivariate Gaussian distribution with mean = hiddens and std_dev = all ones.
# We follow the the VAE of MUNIT
(https://github.com/NVlabs/MUNIT/blob/master/networks.py)
fake_image = self.netG.forward(hiddens + noise, 'dec')
```

- KL loss

```
hiddens = self.netG.forward(input_concat, 'enc')

loss_G_kl = torch.mean(torch.pow(hiddens, 2)) * self.opt.kl
```

> `Global/models/networks.py`

`class Pix2PixHDModel(BaseModel)` 中定义了 Generator，Discriminator

- `self.netG` >> generator
  `netG = GlobalGenerator_DCDCv2(input_nc, output_nc, ngf, k_size, n_downsample_global, norm_layer, opt=opt)`

  定义 Encoder Decoder >> U-Net 结构，具体参考 TAPAMI 版本pdf Page6
- `self.netD` >> discriminator

`netD = MultiscaleDiscriminator(input_nc: 3, opt, ndf: 64, n_layers_D: 3, norm_layer, use_sigmoid, num_D, getIntermFeat)`

三层 PatchGAN discriminator，每层之间加个 AveragePooling

```
class NLayerDiscriminator(nn.Module)
- Conv2d & LeakyReLU  # side = side / 2
- [Conv2d & norm & LeakyReLU] * 2  # 下采样 side = side / 2
- [Conv2d & norm & LeakyReLU]  # 不下采样
```

NonLocal Block

> https://github.com/tea1528/Non-Local-NN-Pytorch

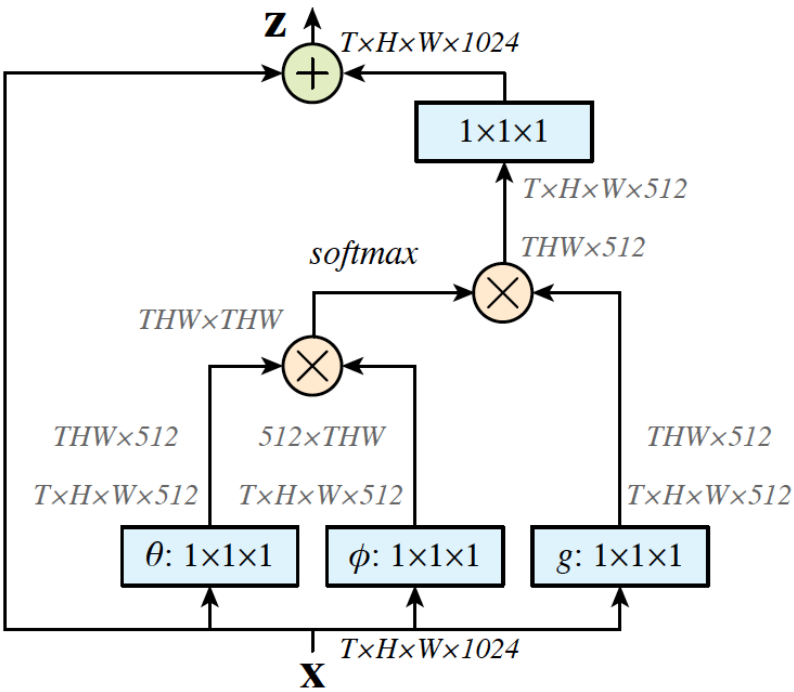


Figure 2. A spacetime **non-local block**. The feature maps are shown as the shape of their tensors, *e.g.*, $T\times H\times W\times 1024$ for 1024 channels (proper reshaping is performed when noted). "$\otimes$" denotes matrix multiplication, and "$\oplus$" denotes element-wise sum. The softmax operation is performed on each row. The blue boxes denote $1\times1\times1$ convolutions. Here we show the embedded Gaussian version, with a bottleneck of 512 channels. The vanilla Gaussian version can be done by removing θ and ϕ, and the dot-product version can be done by replacing softmax with scaling by $1/N$.

`class NonLocalBlock2D_with_mask_Res(nn.Module)`