# NTC与NTSCC入门

## NTC

NTC 的全称是 Nonlinear Transform Coding，即**非线性变换编码**，是一个借助神经网络进行图像压缩的框架。

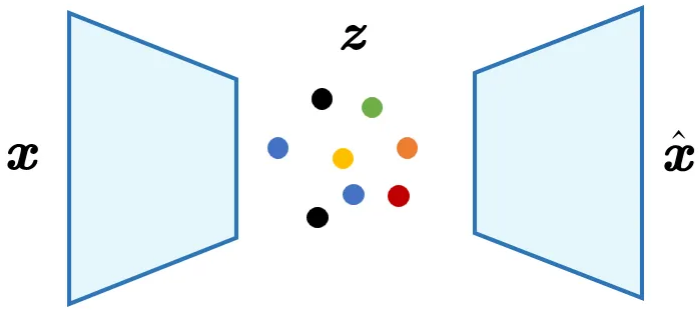
与传统的图像压缩方法一样，输入为原始图像，经过编码器后，得到**比特流**。比特流可以被解码器解码并恢复为图像。因为整个过程中存在量化操作，会引入失真，加上神经网络的非线性特性，因此 NTC 属于**有损压缩**的范畴。

NTC 是由法国学者 Balle 提出，分别在 2016、2018 和 2020 年发表了三篇论文，逐渐形成 NTC 的完整框架，其中以 2018 年的最为经典。该文档及代码都是基于 2018 年 Balle 发表在 ICLR 上的文章 ***Variational image compression with a scale hyperprior***。

> 论文链接：https://arxiv.org/abs/1802.01436

NTC 的设计很大程度受到了 VAE 的启发。与 VAE 一样，都由一个编码器和解码器组成。在 NTC 中，编码器也称作**解析变换（Analysis Transform）**，解码器也称作**综合变换（Synthesis Transform）**。由于编码器和解码器都是由神经网络构成的，而神经网络具有非线性的特点，因此解析变换与综合变换都是**非线性**变换。这就是“**非线性变换编码**”名称的来历。相比较之下，传统的图像压缩，所采取的变换往往是**线性**的（比如离散余弦变换 DCT 等）

变换的目的是**去除图像的相关性**。图像是一个非常高维的信源，存在着大量的冗余信息，正是因为这些冗余的存在，使得图像存在很大的压缩空间。冗余的本质就是相关性。如果能够尽可能的消除相关性，使高维的、相关的内容变成低维的、独立的内容，则达到了压缩的效果。因此，编码器–解码器（解析变换–解析变换）这样的自编码器结构，天然的完成了这个降维的作用。通过降维，并以图像的重建质量作为约束，起到了强制解除相关性的作用。




$$\mathcal{L} = \mathrm{MSE}\left(x, \hat{x}\right)$$

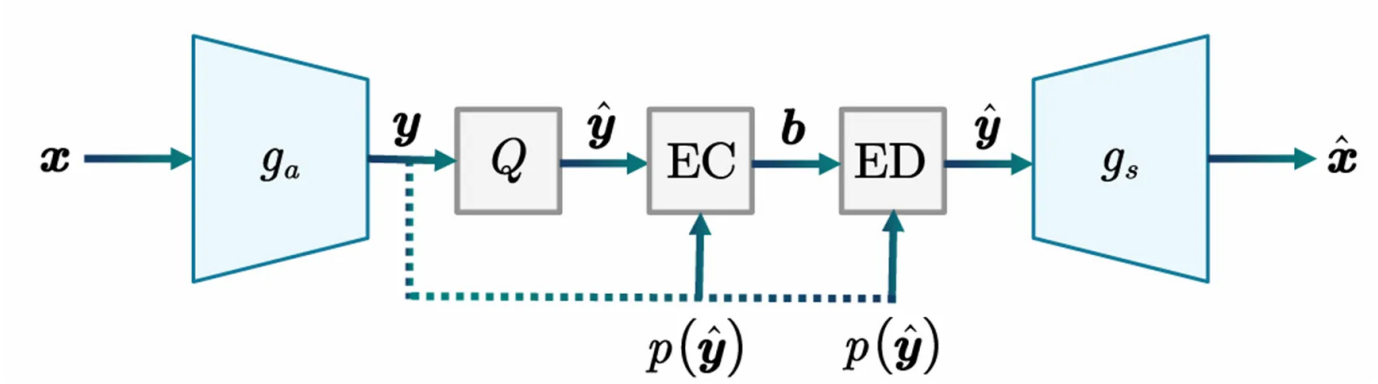
图像经过解相关变换后，依然是连续值（因为神经网络的输出值是连续的）。而只有离散值才可以编码为比特流。因此需要一步**量化**操作。最简单的量化就是均匀量化（这也是 NTC 中的量化方式），即取

整/四舍五入。经过量化后，取值变成离散值，就可以通过熵编码（在 NTC 中具体为算数编码），就可以得到比特流。若想解码，则先进行熵译码，将比特流恢复为离散值，再送入解码器（综合变换）升维，重建原始图像。

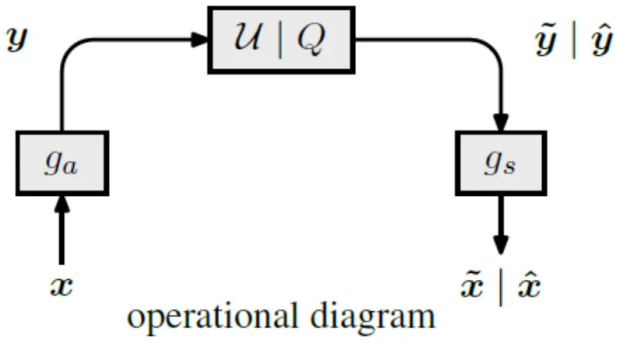
$$\mathbf{x} \xrightarrow{f_\phi(\cdot)} \mathbf{y} \xrightarrow{[\cdot]} \hat{\mathbf{y}} \xrightarrow{\text{转换为比特流}} \mathbf{b} \xrightarrow{\text{传输}} \hat{\mathbf{b}} \xrightarrow{\text{转换为量化特征符号流}} \hat{\mathbf{y}} \xrightarrow{g_\theta(\cdot)} \hat{\mathbf{x}}.$$

Balle 2016 系统框架

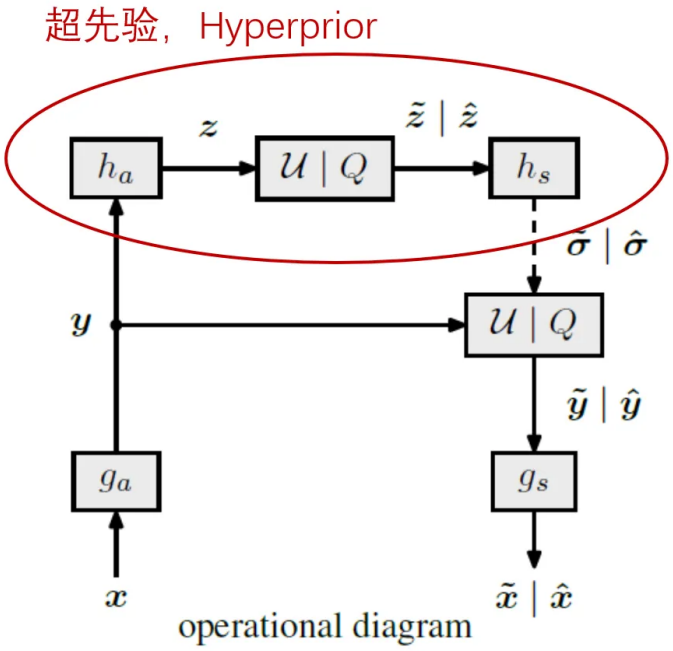
在 Balle 2016 年发表的论文中，已经构建了上面的完整流程。而 2018 年的论文则进一步创造性的又引入了一对神经网络，称作**超先验解析变换**和**超先验综合变换**（本质上也是一对编码器和解码器）。这样做的目的是，提取一个编解码器可以共享的内容（称作边信息），利用边信息可以让编解码器共享更精准估计的概率分布，从而提升编码效率。因为根据信息论，估计出的概率分布越贴近真实分布，熵编码的效率就越高（具体可以参考信息论中交叉熵的定义）。由于超先验网络还起到了估计概率分布的作用，这与 VAE 的思想也是很贴合的，这就是文章标题中“**Variational**”的来历。




Balle 2018 系统框架




Balle 2016




Balle 2018

NTC 训练的损失函数形式如下：

$$\mathcal{L}_{\mathrm{NTC}} = R + \lambda D(\boldsymbol{x}, \hat{\boldsymbol{x}})$$

其中，$R$ 为码率（对应代码里的 `bpp_y` 和 `bpp_z`），$D$ 为图像端到端的失真，比如均方误差 MSE（对应提升重建图像的 PSNR，即峰值信噪比），也可以替换成更复杂的指标（比如多尺度结构相似性 MS-SSIM 或者更贴近人眼主观感知质量的 LPIPS 等）。

在对 NTC 进行训练的时候，由于前文提到的量化操作是一个不可导的操作，这使得神经网络训练进行误差反向传播的时候梯度无法回传。为了解决这一问题，训练的时候，量化操作采用添加均匀噪声来替代（因为量化操作就可以看作引入均匀的量化噪声），从而将不可到的操作变为可导的操作。（或者采用梯度直通估计（STE）的方式，即把不可导的阶跃函数变为导数为 1 的函数）。

下面是我复现的 Balle 2018 年论文的代码，已上传至 GitHub。

> 仓库链接：https://github.com/HJWZWW/NTC

代码实现了 NTC 的训练和测试。仓库包含 `readme.md`，里面有代码的使用教程。

## NTSCC

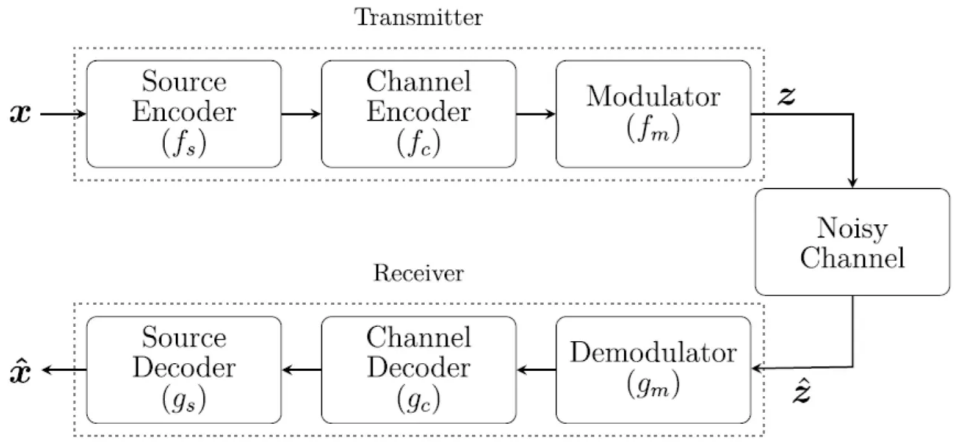
NTSCC 的全称为：Nonlinear Transform Source-Channel Coding，即基于非线性变换的联合信源信道编码。来自戴老师论文：***Nonlinear Transform Source-Channel Coding for Semantic Communications***

> 论文链接：https://arxiv.org/abs/2112.10961

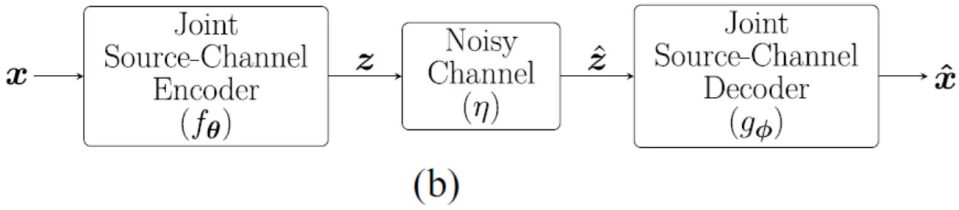
NTSCC 本质上是一种 Deep JSCC。所谓 Deep JSCC（下面简称 DJSCC）就是利用神经网络作为编解码器，直接将信源映射为模拟信号进行传输。传统的信源编码+信道编码的分离范式被神经网络所替代。从整体上来看，非常类似一个去噪自编码器。

需要注意的是，DJSCC 是传输模型，更具体来说是一种模拟传输的通信系统。而前文提到的 NTC 则是一种压缩模型，两者在功能上有本质的不同。

**分离式**编码方案
（传统无线通信系统）

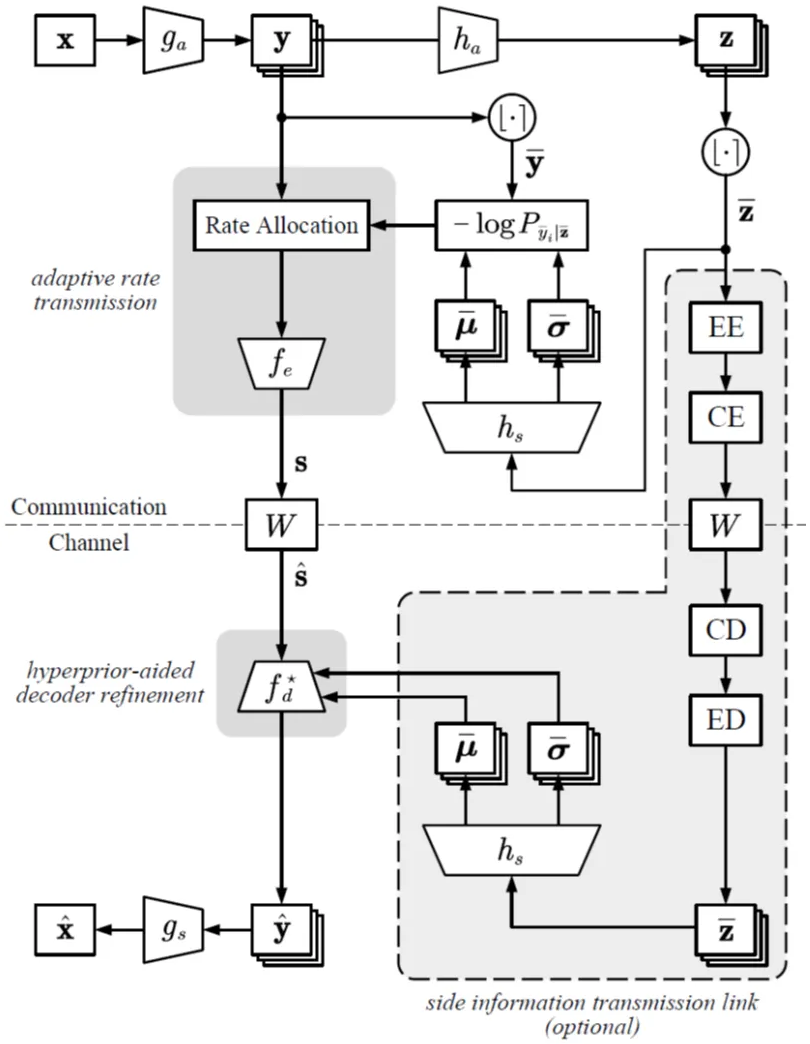



**联合信源信道编码方案**
**JSCC**




分离式编码方案与联合信源信道编码方案对比

而 NTSCC 区别于一般的 DJSCC，正是受到了 NTC 的启发。因为一般的 DJSCC 的维度（或者可以看作是带宽）是固定死的，这使得带宽无法自适应传输内容，比如复杂的图像块和简单的图像块都采用同样的带宽传输，导致对复杂的图像块带宽不足，对简单的图像块带宽过剩。而 NTSCC 的思想正是利用 NTC 可以估计码率这一特点，用 NTC 估计出的码率作为指导，按照比例对每个图像块占用的带宽进行分配，这样就变成了一种可变带宽的 DJSCC。




NTSCC 框架图

NTSCC 训练的损失函数如下：

$$\mathcal{L}_{\mathrm{NTSCC}} = \mathcal{L}_{\mathrm{NTC}} + D_{\mathrm{NTSCC}}(\boldsymbol{x}, \hat{\boldsymbol{x}})$$

其中， $\mathcal{L}_{\mathrm{NTC}}$ 为 NTC 的损失函数，如上一节所示。 $D_{\mathrm{NTSCC}}(\boldsymbol{x}, \hat{\boldsymbol{x}})$ 为端到端的图像失真，同样可以是 MSE 或者其他客观/主观指标。

训练 NTSCC 时，为了训练稳定，最好先训练 NTC，然后把训练好的 NTC 的权重载入之后，再联合端到端训练。

NTSCC 的代码存放在王思贤师兄的 GitHub 仓库中。

> 仓库链接：https://github.com/wsxtyrdd/NTSCC_JSAC22

仓库包含 `readme.md`，里面有代码的使用教程。

需要注意的是，NTSCC 的项目结构与 NTC 的项目结构稍有区别，具体细节请参照两者的 `readme.md`。