基于生成多对抗网络的图像隐写模型

黄冬霞　骆伟祺　刘明林　唐伟轩　黄继武

摘要

自从 2014 年 Ian Goodfellow 等人提出生成对抗网络（GAN），已经在众多领域获得了广泛的应用。然而，到目前为止仅有少量 GAN 模型结合图像隐写的研究。现有基于 GAN 隐写方法主要集中在生成器的设计，在判别器中采用了较为简单的隐写分析器，不可避免的限制了模型的性能。在本文中，提出了一种基于生成多对抗网络（Steg-GMAN）的新型隐写方法来增强隐写安全性。具体来说，首先采用多个隐写分析器来增强，而不是像现有方法那样使用单个隐写分析器判别器的性能。此外，为了在训练阶段平衡生成器和判别器的能力，提出了一种自适应方法，根据不同隐写分析器能力更新提出的 GAN 模型的参数。在每次迭代中，只更新判别器中所有隐写分析器中最差的一个，同时使用最强隐写分析器的梯度更新生成器。这样可以逐步提高生成器和判别器的性能，从而避免梯度消散导致训练失败。大量比较结果表明，与传统的隐写术和现代基于 GAN 的隐写术方法相比，所提出的方法可以取得最先进的结果。此外，大量的消融实验验证了所提出模型的合理性。

关键词：隐写；生成对抗网络；隐写分析

1　引言

图像隐写旨在以难以察觉的方法将秘密信息隐藏在数字图像中。近年来，大多数隐写方法都是在失真最小化框架下设计的。在此框架内，嵌入成本的设计是一个关键方面。嵌入成本是指由于在图像的嵌入单元（例如像素或 DCT 系数）内隐藏秘密信息而导致的失真或破坏的程度。这些不同的嵌入成本显着影响图像隐写术的安全性能。定义了嵌入成本，就可以利用综合症网格码（STC）来嵌入秘密消息并最小化嵌入成本的总和。

与之相对的，图像隐写分析试图识别含有秘密信息的隐写图像。一些传统的隐写分析方法，如 SPAM 和 SRM，通常将输入图像转换到不同的残差域，然后提取高维统计特征来识别一些隐写方法的嵌入修改痕迹，最后使用集成分类器进行二分类。随着深度学习的出现，一些基于卷积神经网络的（CNNs）的隐写方法被提出，如 Xu-Net、SRNet 等，与传统的方法相比取得了很大的改进。

隐写和隐写分析是相互对立、相互促进的。为了提高隐写的安全性能，近年一些基于深度学习的隐写方法被提出。这些方法可以被分为两类：基于对抗攻击的方法和基于对抗网络的方法。基于对抗攻击的方法试图根据一些预训练的 CNN 隐写分析器的梯度调整原始的嵌入代价。基于对抗网络的隐写方法通过迭代训练生成器和判别器，生成高度安全的载密图像。现有的基于 GAN 的隐写分析方法主要集中在生成器的设计上，特别是用于生成概率图和嵌入仿真器。在训练阶段没有考虑到生成器和判别器之间关系。

本文提出一种新的基于 GAN 的隐写方法来增强安全性能，首先在判别器中使用多个隐写分析器而不是单个隐写分析器，然后提出一种自适应的方法更新生成器和判别器的参数，以平衡生成器和判别器的能力。通过这种方法，提高了模型的安全性能，同时避免因梯度消散而导致的训练失败。

2 相关工作

2.1 生成对抗网络

生成对抗网络由 Goodfellow 等人在 2014 年首次提出，它通过两个玩家之间的 minimax 博弈来生成任务模型。一方面，生成器试图生成与真实数据相似的假样本。具体来说，生成器 $G(z;\theta_g)$ 由参数为 θ_g 的多层感知机表示的可微函数，其输入是服从于简单分布 $p_z(z)$ 的噪声 z，输出是一个接近于真实数据分布 $p_{data}(x)$ 的假样本。另一方面，判别器试图区分生成器生成的假样本和真实数据。具体来说，判别器 $D(x;\theta_d)$ 由参数为 θ_d 的多层感知机，其输出一个 [0,1] 范围内的标量，代表输入 x 来自真实数据的概率。该模型通过生成器和判别器之间的相互对抗和学习，最终达到均衡状态，D 和 G 以下面的等式进行 minimax 博弈：

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{1}$$

其中 E 表示关于真实数据分布和噪声分布的期望，$G(x)$ 表示由噪声 z 通过生成器 G 生成的假样本。在 minimax 博弈中，训练 D 使 $[logD(x)+log(1-D(G(z))]$ 最大化以增加将真实样本与假样本正确分类的概率，同时训练 G 使 $log(1-D(G(z))$ 最小化使生成的假样本尽可能的接近真实样本数据分布。

2.2 基于 GAN 的隐写方法

隐写旨在以不可感知的方式将秘密信息嵌入到载体图像中，而隐写分析则试图从载体图像中识别经过隐写数据嵌入后得到的载密图像。隐写和隐写分析如同 GAN 中的生成器和判别器一样，相互对抗，相互促进。因此，一些隐写方法使用 GAN 架构来增强隐写的安全性。下面分别对三种经典的方法 ASDL-GAN、UT-GAN 和 SPAR-RL 进行简单介绍。

ASDL-GAN 首先将 GAN 引入到图像隐写中进行嵌入代价的学习，该模型中生成器包含一个 KV 高通滤波器，然后是 25 组卷积层，判别器是 Xu-Net。ASDL-GAN 使用预训练的三元嵌入仿真器（TES）子网络作为嵌入仿真器，将概率映射到修改，并在 GAN 的反向传播中提供帮助。生成器的损失函数与判别器中隐写分析器的不可检测性直接相关，判别器的损失函数定义为二分类的交叉熵。然而。基于文献中的实验，ASDL-GAN 的安全性与传统的隐写方法 HILL 相比，其安全性能相对较差。

UT-GAN 在生成器中使用了一个 U-Net 网络，在判别器中使用了具有 6 个高通滤波器的 Xu-Net。此外，设计了无需预训练的可微 Double-tanh 函数替代 TES 子网络作为嵌入仿真器，实现更短的训练时间。UT-GAN 中的损失函数定义方式与 ASDL-GAN 相同，实验结果表示，UT-GAN 的性能优于 ASDL-GAN 和 HILL。

SPAR-RL 是基于 GAN 的隐写方法的增强策略。在 SPAR-RL 中，GAN 方法中的生成器网络被视为策略网络，判别器网络被视为环境网络。策略网络旨在制定一个策略来生成载密图像欺骗环境网络，环境网络区分载体图像和载密图像，从而判断策略网络的有效性。在该方法中，使用了一个离散阶梯函数，此外，策略网络的损失函数通过从环境网络分配给每一个嵌入单元的像素级损失计算。实验表明 SPAR-RL 可以提高现有方法的安全性能。

3 本文方法

3.1 本文方法概述

如图 1 所示，本文方法的网络架构包括两个重要模块，生成器和判别器。生成器使用 U-Net 网络，使用无需预训练的可微双层 Tanh 函数作为嵌入仿真器，判别器使用多个先进的隐写分析器。

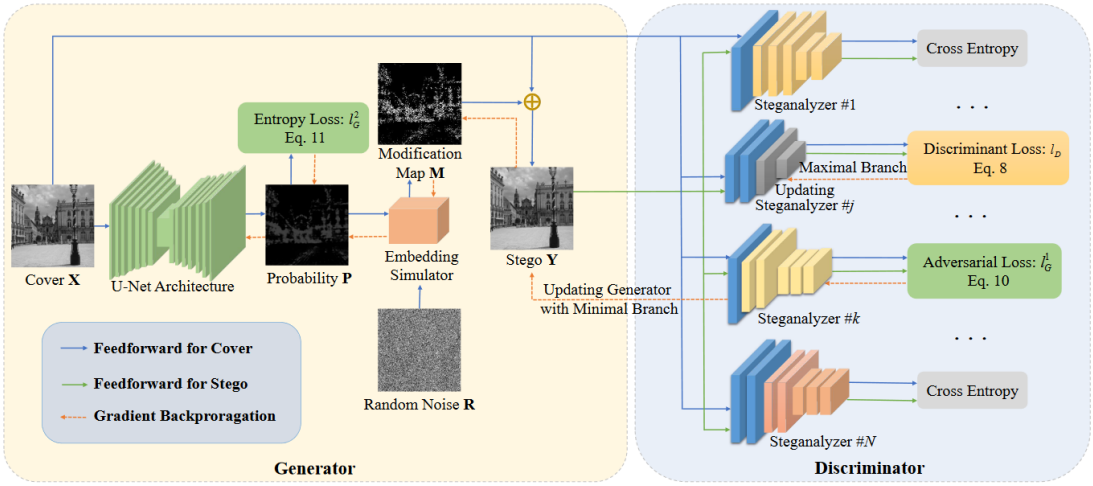


图 1: 本文提出的网络架构

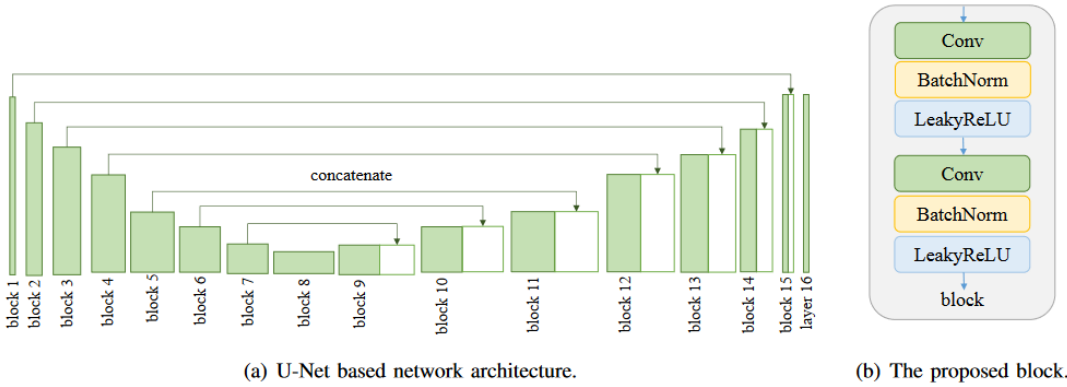
3.2 构造生成器

生成器旨在根据输入的载体图像和秘密消息生成相应的载密图像，为此，生成器包括两个步骤：

1、生成概率图。该步骤试图将一个大小为 H×W 的载体图像 $X = (x_{i,j})^{H\times W}$ 转化为嵌入概率图 $P = (p_{i,j})^{H\times W}$，其中 $p_{i,j}$ 表示像素点 $x_{i,j}$ 被修改 ±1 的概率。所提出的方法中，限制每个像素 $x_{i,j}$ 的 +1 和-1 的概率相等，表示为：

$$p_{i,j}^{+1} = p_{i,j}^{-1} = p_{i,j}/2 \tag{2}$$

如图 2 所示，采用基于 U-Net 的架构完成载体图像 X 到概率图像 P 的过程，它包含 15 个块和一个反卷积层。前 8 个块对图像进行下采样，后 7 个块对图像进行上采样。将第 i 个块（$i = 1, 2, ..., 7$) 的输出和第（16-i）个块的输出串联作为第（17-i）个块的输入。每个块包含 2 个卷积层，然后进行批处理归一化和 LeakyReLU。



(a) U-Net based network architecture. (b) The proposed block.

图 2: 基于 U-Net 网络生成概率图

2、模拟消息嵌入。通过 STC 模拟器从生成的概率图 P 中获得隐写图像。首先生成一个随机噪声 $R = (r_{i,j})^{H\times W}$，其中 $r_{i,j}$ 独立同分布且服从于 [0,1] 的均匀分布。通过比较 $p_{i,j}$ 和 $r_{i,j}$，定义 STC 模拟

器中的修改映射 $M=(m_{i,j})^{H\times W}$ 如下：

$$m_{i,j}=\begin{cases}-1 & r_{i,j}<p_{i,j}^{-1}\\ +1 & r_{i,j}>1-p_{i,j}^{+1}\\ 0 & otherwize\end{cases} \tag{3}$$

由于上述阶梯函数不可微，采用 Double-Tanh 函数作为仿真嵌入器，它用一个可微的近似来代替离散阶梯函数，使 GANs 可以反向传播，Double-Tanh 函数定义如下：

$$m_{i,j}=\frac{1}{2}tanh(\lambda(p_{ij}^{+1}-r_{i,j}))-\frac{1}{2}tanh(\lambda(p_{ij}^{-1}-(1-r_{i,j}))) \tag{4}$$

其中 $m_{i,j}$ 表示仿真嵌入的修改，区间范围是 [-1,1]，参数 λ 控制模拟精度，最终通过 $Y=X+M$ 得到载密图像。

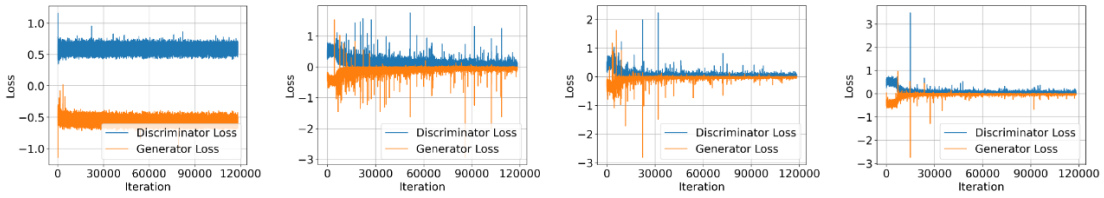
当训练完成后，输入图像的嵌入代价 ρ 可以通过如下式子计算：

$$\begin{cases}\rho_{i,j}^{+1}=\ln\left(\dfrac{1}{p_{i,j}^{+1}}-2\right)\\ \rho_{i,j}^{-1}=\ln\left(\dfrac{1}{p_{i,j}^{-1}}-2\right)\\ \rho_{i,j}^{0}=0\end{cases} \tag{5}$$

根据得到的嵌入代价，可以使用 STC 完成消息嵌入（而不是使用嵌入模拟器）得到实际的载密图像 Y，以最小化嵌入代价总和。

3.3 构造判别器

判别器试图判断输入图像时载体图像 X 还是有载密图像 Y，并为生成器和判别器中的网络参数更新提供有用信息。现有基于 GAN 的隐写方法仅在判别器中使用单个隐写分析器，限制了模型的最终性能。本文在判别器中使用多个隐写分析器，同时保持 UT-GAN 的核心框架完整，图 3 显示了随着训练迭代次数的增加，在判别器中分别使用三个隐写分析器的判别器和生成器的损失。从图 3 可以看出，对于 Xu-Nut，在训练阶段，生成器和判别器的损失可以保持动态平衡，在这种情况下最终可以得到一个相对较优的生成器，然而，对于另外三个更强的隐写分析器，判别器的损失迅速收敛到 0，这意味着鉴别器可以准确地从载体图像中识别出生成的隐写图像，在这种情况下，判别器的梯度将消失，因此不能为更新生成器提供任何有效信息。



(a) Xu-Net (b) Yedroudj-Net (c) SRNet (d) CovNet

图 3: UT-GAN 框架使用 4 种不同的隐写分析器时判别器与生成器的损失对比

基于上述实验和分析得出结论，应该引入更强的隐写分析器来提高判别器的性能，同时设计新的训练策略来保持判别器和生成器之间的动态平衡。为此，作者在判别器中使用多个隐写分析器，然后采用一种自适应的方式来更新所提出的 GAN 模型中的参数，以避免梯度消失，具体分为以下三个步骤：

1、计算交叉熵。对于判别器中每个隐写分析器 D_i 输入一幅图像，其输出是隐写分析网络中 softmax

层对输入图像的判别结果，表示为二维归一化向量。隐写分析器 D_i 的判别性能由二元交叉熵 E_i 来衡量，定义如下：

$$E_i = -Z_0 log(D_i(X)) - Z_1 log(D_i(Y)) \tag{6}$$

式中 $D_i(X)$ 和 $D_i(Y)$ 表示第 i 个图像分别被判别为载体图像和载密图像的结果，i = 1，2，.. N，向量 $z_0 = (1 \quad 0)^T$ 和 $z_1 = (0 \quad 1)^T$ 是 X 和 Y 的真实标签。在所提出的模型中，第 i 个隐写分析器 D_i 的判别性能由二元交叉熵 E_i 来判断。二元交叉熵越大，隐写分析器越弱。假设第 j 个隐写分析器是最弱的，第 k 个隐写分析器是最强的。

2、更新判别器。为了增强判别器的性能，在每次迭代中只更新判别器中最弱的隐写分析器（假设为第 j 个），保持其他 N-1 个更强的隐写分析器不变。通过这种方式，隐写分析器可以缓慢渐进的方式进行增强。

3、更新生成器。为了增强生成器的性能，使用最强的隐写分析器更新生成器（假设为第 k 个），主要原因是最强的隐写分析器可以给生成器提供相对更有效的反馈。本文提出的 GAN 模型的判别器使用了两种隐写分析器（Xu-Net 和 Yedroudj-Net），在隐写安全性和训练效果之间取得了较好的折中。

3.4 设计损失函数

所提出的模型中有两个损失函数，即判别器损失 l_D 和生成器损失 l_G，分别对两个损失函数进行描述。

1、判别器损失：判别器损失旨在判断一幅图像是否含有秘密信息，在提出的判别器中使用多个隐写分析器，并在每次迭代中更新最弱的隐写分析器，因此判别器的损失 l_D 定义如下：

$$l_D = \underset{i\in\{1,2,...N\}}{max}\{-z_0 log(D_i(X)) - z_1 log(D_i(Y))\} \tag{7}$$

2、生成器损失：生成器的目的是以不易察觉的方式将秘密信息嵌入到载体图像中，从而使生成的载密图像难以被判别器中的隐写分析器检测，因此生成器的损失定义如下：

$$l_G = -\alpha \cdot l_G^1 + \beta \cdot l_G^2 \tag{8}$$

其中，第一项 l_G^1 表示每一次迭代过程中最强的隐写分析器的对抗损失，第二项 l_G^2 是熵损失，用于保证嵌入负载。α、β 分别代表二者的权重。

$$l_G^1 = \underset{i\in\{1,2,...N\}}{min}\{-z_0 log(D_i(X)) - z_1 log(D_i(Y))\} \tag{9}$$

$$l_G^2 = (-(\sum_{\forall(i,j)}\sum_{\forall m} p_{i,j}^{(m)} \cdot log_2(p_{i,j}^{(m)})) - (H{\times}W{\times}q))^2 \tag{10}$$

其中 H 和 W 表示输入图像的高度和宽度，$1{\le}i{\le}H, 1{\le}j{\le}W$,m 表示嵌入修改，q 表示嵌入载荷。在训练阶段，损失函数 l_D 和 l_G 分别用于计算梯度和更新判别器和生成器的模型参数，下面是模型的训练步骤算法伪代码。

Procedure 1 Training Steps of Steg-GMAN

Input: cover image X, random noise R, *epochs,* generator loss parameter α and β, number of discriminators *N,* label of cover z_0,label of stego z_1,learning rate η

Output: generator parameter θ_g

for *epoch* ***in*** *{1,⋯,epochs }* **do**

  $\mathbf{P} \leftarrow G(\mathbf{X};\theta_g)$

  Compute **M** by Double-Tanh function

  $\mathbf{Y} \leftarrow \mathbf{X} + \mathbf{M}$

  for *i in {1,...,N }* **do**

    $E_i \leftarrow -z_0 log(D_i(X)) - z_1 log(D_i(Y))$

  end

  $l_D \leftarrow max_{i\in\{1,2,...,N\}}\{E_i\}$

  $Update\ Discriminator\ by \theta_d \leftarrow \theta_d - \eta\nabla_{\theta_g} l_G$

  $l_G^1 \leftarrow min_{i\in\{1,2,...,N\}}\{E_i\}$

  Compute l_G^2

  $l_G \leftarrow -\alpha \cdot l_G^1 + \beta \cdot l_G^2$

  $Update\ Generator\ by \theta_g \leftarrow \theta_g - \eta\nabla_{\theta_g} l_G$

  Learning rate $\eta decay$

end

return θ_g

4 复现细节

4.1 与已有开源代码对比

本篇论文在 Github 上存在基于 python 的开源模型代码，作者给出了载密图像的生成过程，包括对抗网络的训练以及修改成本的计算。对论文模型的复现工作首先使用作者给出的开源代码训练 steg-GMAN 模型，根据 GAN 的生成器计算图像像素点修改成本，之后使用 matlab 得到载密图像。

作者提供的代码中仅包含模型的训练部分，没有对结果的测试部分，复现过程加入训练隐写分析器对生成的载密图像进行测试工作。首先使用 BOWS2 载体图像和 UNIWARD 隐写生成的载密图像组合作为训练数据集，训练性能良好的隐写分析器，对 steg-GMAN 网络生成器生成的 Bossbase 数据集载密图像进行判别分析，根据错误率评估模型的安全性。

4.2 实验环境搭建

复现本篇论文使用 vscode 远程连接服务器，1080tiGPU 训练模型，使用 3.8 版本的 python、2.1 版本的 pytorch，在掩密图像生成过程使用 matlab2023b。

4.3 复现过程

复现总体流程如图所示：

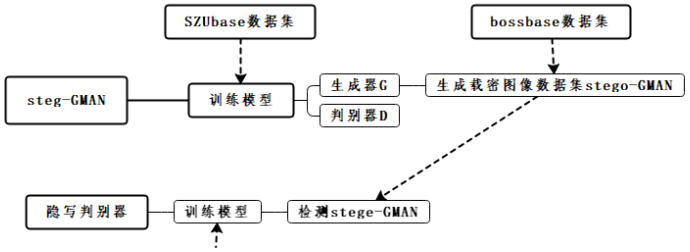


图 4: 复现工作

在训练 steg-GMAN 阶段使用 SZUbase 的 40000 张载体图片，使用交叉熵损失函数，Adam 优化器来更新神经网络参数。生成器和判别器的训练过程如图所示。比较两个隐写判别器的损失值大小，以

较大的损失值反向传播，更新隐写分析器，将较小的损失值作为 g_{l1}，与 g_{l2} 组合更新生成器参数。

```
for i,sample in enumerate(dataloader,0):

    #生成器生成载密图像过程
    optimizerG.zero_grad()

    data, rand_ud, label = sample['data'], sample['rand_ud'],sample['label']
    cover, n, label = data.cuda(), rand_ud.cuda(), label.cuda()

    p = netG(cover)          #使用生成器网络得到修改概率图
    p_plus = p/2.0 + 1e-5    #计算像素点±1的概率
    p_minus = p/2.0 + 1e-5

    m =  - 0.5 * torch.tanh(TS_PARA * (p - 2 * n)) + 0.5 * torch.tanh(TS_PARA * (p - 2 * (1 - n)))
                             #仿真嵌入模拟器得到修改图
    stego = cover + m        #将载体图像和修改图相加得到载密图像

    C = -(p_plus * torch.log2(p_plus) + p_minus*torch.log2(p_minus)+ (1 - p+1e-5) * torch.log2(1 - p+1e-5))
                             #计算像素点嵌入成本
    d_input = torch.zeros(opt.batchSize*2,1,256,256).cuda() #张量，大小（批大小，1，256，256）
    d_input[0:opt.batchSize*2:2,:] = data                   #偶数索引存放cover
    d_input[1:opt.batchSize*2:2,:] = stego                  #奇数索引存放生成的stego
                                                            #label = np.array([0, 1], dtype='int32')
    label = label.reshape(-1)                               #将label降为1维，分别表示载体图像和载密图像


    optimizerD1.zero_grad()                  #分别通过netxu、netyed，得到图片为0类1类的概率，计算损失值
    errD1 = criterion(netXu(d_input.detach()), label)

    optimizerD2.zero_grad()
    errD2 = criterion(netYed(d_input.detach()), label)
```

```
    if errD1.item() > errD2.item():
        errD1.backward()
        optimizerD1.step()
        iteration_yed += 1
        g_l1 = criterion(netYed(d_input), label)

    else:
        errD2.backward()
        optimizerD2.step()
        iteration_xu += 1
        g_l1 = criterion(netXu(d_input), label)

    g_l2 = torch.mean((C.sum(dim = (1,2,3)) - 256 * 256 * PAYLOAD) ** 2)

    errG = -g_l1 + 1e-7*g_l2
    if epoch > 0:
        train_hist['G_loss'].append(errG.item())
        train_hist['D_loss_Xu'].append(errD1.item())
        train_hist['D_loss_Yed'].append(errD2.item())

    errG.backward()
    optimizerG.step()
```

图 5: 对抗网络迭代过程

使用迭代训练后的模型得到 baoobase 数据集的修改概率图，使用 matlab 采用 STC 编码得到隐写图像。

在论文作者提供的源代码基础上，使用 bows2 数据集和 UNIWARD 隐写方法生成的掩密图像组合训练隐写分析器，用于评估本次复现生成的掩密图像的安全性。尝试使用 yed-net 网络结构，训练隐写分析器。下面是训练 yed-net 的界面，代码部分是读取数据集训练网络模型部分，在训练过程中采用了作者提供源代码中隐写分析器 yed-net 网络结构。

图 6: yed-net 网络训练

此外，使用 SRM 特征提取结合集成分类算法训练分类器测试 steg-GMAN 隐写安全性。首先使用 bows2 数据集使用 UNIWARD 隐写方法训练隐写分类器，检测 steg-GMAN 在 bossbase 数据集上面生成的掩密图片，评估模型的抗隐写分析能力。

5 实验结果分析

5.1 steg-GMAN 训练结果

下图展示了对抗网络训练前几轮和最后几轮的生成器和判别器的损失值，随迭代次数的变化，生成器和判别器在训练过程中损失值处于动态平衡的状态，判别器的损失不会迅速收敛，可以在迭代过程中反馈用于更新生成器，不断提高生成器的性能。最终 xunet 迭代更新了 78268 次，yudnet 迭代更新了 45860 次。

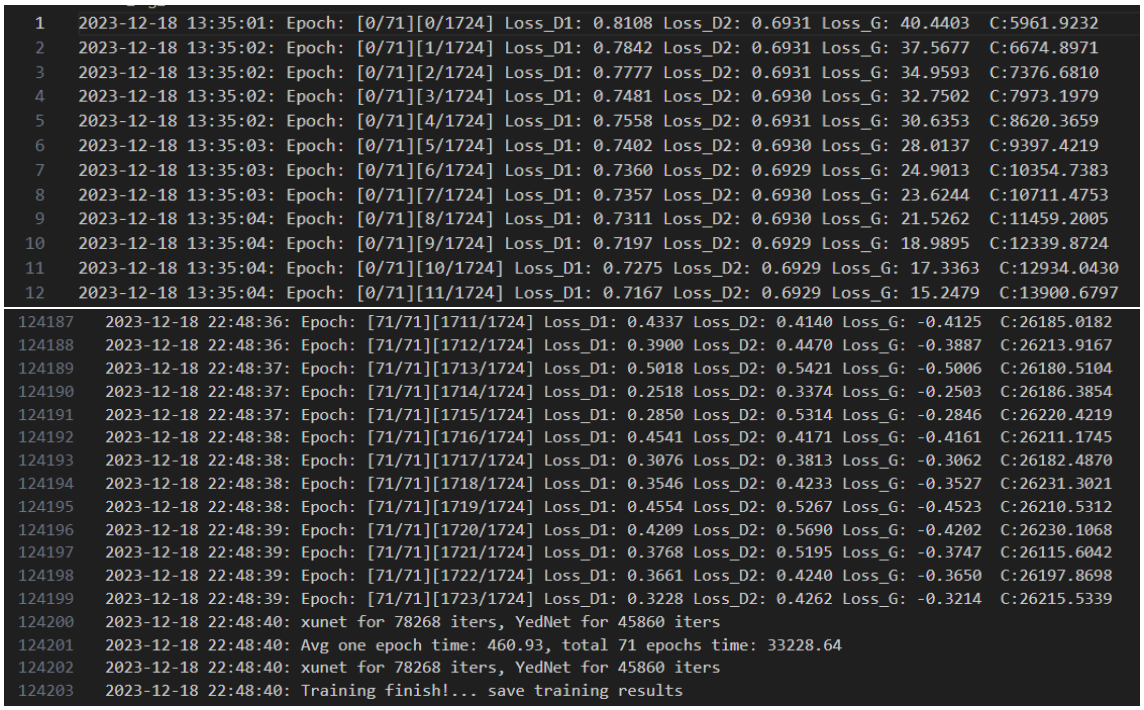


图 7: 训练 steg-GMAN 模型损失函数随迭代次数的变化图

下图是训练 steg-GMAN 时生成器和两种隐写分析器的损失值变化。

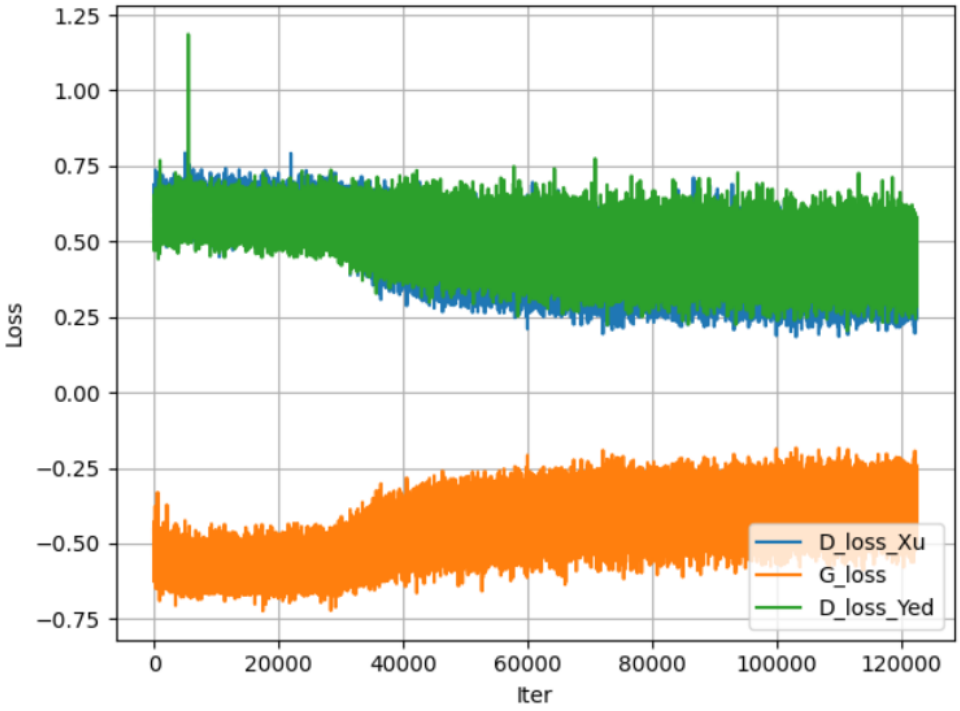


图 8: 训练 steg-GMAN 模型损失函数随迭代次数的变化图

5.2 steg-GMAN 结果

下图展示了使用 bossbase 数据集分别在 0.1bpp 和 0.4bpp 嵌入负载率下的生成的载体图像的结果以及修改图，可以看出以网络的生成器模型对图片的修改主要集中在纹理区域，使得修改难以被检测出。

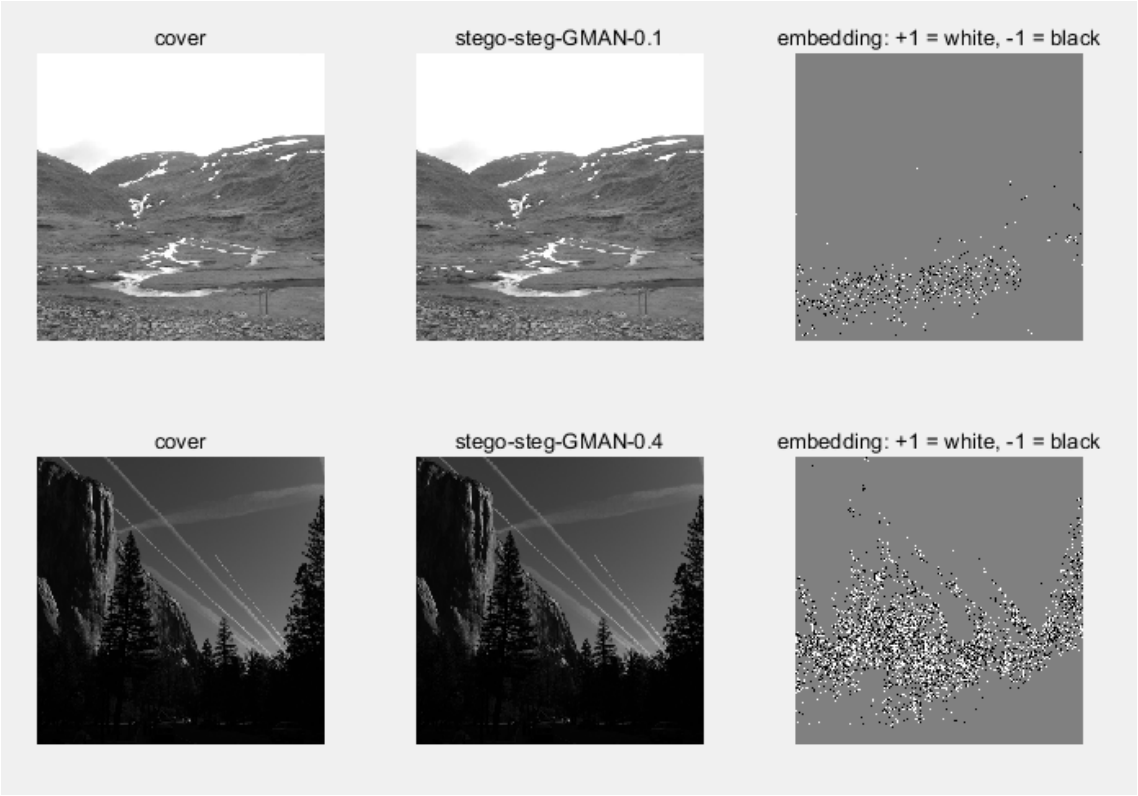


图 9: 使用载体图像（bossbase 数据集 1.pgm 和 9012.pgm）生成的载密图像以及对应修改图

5.3 生成隐写图像的检测结果

为了检测生成图像的安全性，首先使用 BOWS2 预先训练出性能良好的隐写分析器，再使用隐写分析器对 steg-GMAN 生成器在 bossbase 上生成的隐写图像进行判别检测，根据隐写分析器的判别错

误率评估隐写方法的安全性能。

实验分别使用了 SRM 和 yednet 隐写分析，结果均表示论文提出的方法具有更好的隐写安全性。

6 总结与展望

本次实验对 Steg-GMAN 进行复现，实现了一个生成式多对抗网络。生成器采用 U-Net 框架生成修改概率图，并使用 Double-tanh 函数模拟 STC 信息嵌入过程，得到修改图，将修改图与与原始图像相加得到隐写图像，判别器中使用 XuNet 和 YedNet 隐写分析器，通过迭代更新生成器和判别器提高生成器生成的隐写图像的安全性。实验在作者提供的开源代码基础上，加入了对生成载密图像的安全性测试的工作，使用根据其他隐写方法（本次实验使用 UNIWARD）生成数据集训练当前性能较好的隐写分析器来检测生成的载密图像，根据检测错误率评估隐写方法的安全性。

在未来的研究中，可以考虑几个方面进行研究。首先，本文对于像素点的修改嵌入成本认为是对称的，即像素点 ±1 的概率相等，实际情况可能是非对称的。其次，在生成器中可以采用性能更加良好的网络架构，或使用多个网络生成概率图，增加载密图像的多样性，使用多个网络时，同时考虑更新参数的策略，实现生成器和判别器更好的动态平衡。此外，在训练网络过程中，如何在保持性能基本不变的情况下减少模型训练时间也是值得研究的方向。

参考文献

[1] YEDROUDJ M, COMBY F, CHAUMONT M. Yedrouj-Net: An efficient CNN for spatial steganalysis [J]. arXiv e-prints, 2018.

[2] TANG W, LI B, TAN S, et al. CNN Based Adversarial Embedding with Minimum Alteration for Image Steganography[J]. 2018.

[3] YANG J, RUAN D, HUANG J, et al. An Embedding Cost Learning Framework Using GAN[J]. IEEE Transactions on Information Forensics and Security, 2019, 15(99): 839-851.

[4] SINGH B, SUR A, MITRA P. Multi-Contextual Design of Convolutional Neural Network for Steganalysis[J]. 2021.

[5] BOROUMAND M, CHEN M, FRIDRICH J. Deep Residual Network for Steganalysis of Digital Images [J]. IEEE Transactions on Information Forensics and Security, 2019, 14(5): 1181-1193.

[6] ZHANG R, ZHU F, LIU J, et al. Depth-Wise Separable Convolutions and Multi-Level Pooling for an Efficient Spatial CNN-Based Steganalysis[J]. IEEE, 2020.

[7] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative Adversarial Nets[C]//Neural Information Processing Systems. 2014.

[8] RONNEBERGER O, FISCHER P, BROX T. U-Net: Convolutional Networks for Biomedical Image Segmentation[C]//International Conference on Medical Image Computing and Computer-Assisted Intervention. 2015.

[9] FILLER T, JUDAS J, FRIDRICH J. Minimizing Additive Distortion in Steganography Using Syndrome-Trellis Codes[J]. IEEE Transactions on Information Forensics & Security, 2011, 6(3): 920-935.

[10] LI B, WANG M, HUANG J, et al. A new cost function for spatial image steganography[C]/ /2014 IEEE International Conference on Image Processing (ICIP). 2015.

[11] HOLUB V, FRIDRICH J, DENEMARK T. Universal distortion function for steganography in an arbitrary domain[J]. Eurasip Journal on Information Security, 2014, 2014(1): 1.

[12] SEDIGHI V, COGRANNE R, FRIDRICH J. Content-Adaptive Steganography by Minimizing Statistical Detectability[J]. IEEE Transactions on Information Forensics and Security, 2015, 11(2): 1-1.

[13] TANG W, LI B, BARNI M, et al. An Automatic Cost Learning Framework for Image Steganography Using Deep Reinforcement Learning[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 952-967.

[14] TANG W, TAN S, LI B, et al. Automatic Steganographic Distortion Learning Using a Generative Adversarial Network[J]. IEEE Signal Processing Letters, 2017, 24(99): 1547-1551.