厦門大學

本 科 毕 业 论 文 （设 计）

（主修专业）

基于 SinGAN-Dehaze 单图像去雾的 MindSpore 实现

MindSpore Implementation of Single Image Dehazing via SinGAN-Dehaze

姓　　名：郭 龙

学　　号：22920182204175

学　　院：信息学院

专　　业：网络空间安全专业

年　　级：2018 级

校内指导教师：　　曲延云　　（教授）

二〇二二年五月八日

厦门大学本科学位论文诚信承诺书

本人呈交的学位论文是在导师指导下独立完成的研究成果。本人在论文写作中参考其他个人或集体已经发表的研究成果，均在文中以适当方式明确标明，并符合相关法律规范及《厦门大学本科毕业论文（设计）规范》。

该学位论文为（　　）课题（组）的研究成果，获得（　　）课题（组）经费或实验室的资助，在（　　）实验室完成（请在以上括号内填写课题或课题组负责人或实验室名称，未有此项声明内容的，可以不作特别声明）。

本人承诺辅修专业毕业论文（设计）（如有）的内容与主修专业不存在相同与相近情况。

学生声明（签名）：

年　　月　　日

致谢

又是凤凰花开之际，转瞬间在厦门大学便度过了近四年时光。大学本科这个阶段，是自己踏入成年世界的第一站，过程中喜悲参半、起起落落，在不断的选择和坚持中慢慢走出了属于我的道路。

象牙塔隔绝了社会的冲击，但我们的成长依旧伴随着这一代青年所要承担的压力，这也隐示了我们要完成的使命。在启程赶往下个目的地之前，我要感谢我的父母，在我看得到和看不到的地方默默付出，保障我的学习生活，关心我的成长，恩情难以回报；我要感谢我可爱的学校，她被赞为中国最美大学，她用开放包容的胸怀，向我展示了大学的无限可能；我要感谢我尽职尽责的老师们，用细致认真的教学态度为我打开网络空间安全专业的大门，在计算机学科中寻找自己的方向；我要感谢我真挚的朋友们，愿意分享我的欢乐与失意，关心与陪伴成为我一路进步的重要助推力；还有那些曾给予我帮助的人们，这些都是我与鹭岛的美好回忆。

这篇论文和我的大学本科生活都已经临近到片尾曲，最后郑重向我的本科毕设指导老师曲延云教授表示十分感谢，指导我们进行工作，给出切实可行的指导建议，让我能圆满完成考研和毕业工作；同时，也再次对课题组中给我带来支持、答疑解惑的唐靖钧等师兄师姐，以及项目组成员表示衷心的感谢！

厦门大学，我爱你，再见！

摘要

人工智能正在持续冲击着世界最前沿的科研技术，为了打造国产化的深度学习平台，华为自研发布了新一代人工智能开源计算框架 MindSpore，相比 Google 的 TensorFlow、Amazon 的 MXnet、Facebook 的 PyTorch，以及 Microsoft 的 CNTK 等四大架构，它具有开发态友好、运行态高效、部署态灵活等三大特点。

本文在 MindSpore 框架上实现 SinGAN-Dehaze 去雾模型，用以解决不成对图像去雾任务。监督图像去雾方法的前提是要有成对训练数据，人工标注代价很高，为此，现有图像去雾方法多在合成有雾图像集上进行训练，而利用这样的图像去雾模型在处理真实有雾图像时性能会下降。考虑到干净图像和有雾图像容易获得，使用不成对图像来训练模型，SinGAN-Dehaze 作为一种新的用于非配对图像去雾的单 GAN 模型，将循环一致性约束解耦为内容一致性和样式一致性，在客观指标和视觉效果方面都能取得优异的性能。

本项目将 PyTorch 下的训练网络移植到国内框架，推进深度学习代码的国产化，同时利用昇腾处理器的优异性能保证不成对图像转换的精度。主要研究工作如下：

1、分析 SinGAN 单图像对比转换的网络结构，了解源代码的技术细节；

2、学习并比较 MindSpore 与业内主流框架 PyTorch 的异同，对比 API 映射差异，确定并实施迁移方案；

3、调整网络训练的学习参数，尽量达到原代码的图片转换精度，进而部署到华为平台实现全栈共享。

关键词：图像去雾；深度学习；MindSpore；PyTorch

Abstract

Artificial intelligence is continuously impacting the world's most cutting-edge scientific research technology. In order to build a domestic in-depth learning platform, Huawei has self-developed and released a new generation of artificial intelligence open source computing framework MindSpore. Compared with TensorFlow, MXnet, PyTorch and CNTK, it has three characteristics: friendly development, efficient operation and flexible deployment.

In this paper, the SinGAN-Dehaze dehazing model is implemented on the MindSpore to solve the unpaired image dehazing task. The premise of supervised image dehazing method is to have paired training data, and the cost of manual annotation is very high. Therefore, most of the existing image dehazing methods are trained on the synthetic hazy image set, and the performance of using such image dehazing model will decline when processing the real hazy image. Considering that clean images and hazy images are easy to obtain, unpaired images are used to train the model. As a new single GAN model for unpaired image dehazing, SinGAN -Dehaze decouples the cycle-consistency constraint into content consistency and style consistency, and can achieve excellent performance in objective indicators and visual effects.

In this project, the training network under PyTorch is transplanted to the domestic framework to promote the localization of deep learning code. And the excellent performance of ShengTeng is used to ensure the accuracy of unpaired image conversion.

First, analyze the network structure of SinGAN single image contrast conversion, and understand the technical details of the source code;

Second, compare the similarities and differences between MindSpore and PyTorch, compare the API differences, determine and implement the migration scheme;

Third, adjust network training parameters to achieve the conversion accuracy of the original code as much as possible, and then deploy it to Huawei platform .

Key words: Image dehazing; Deep learning; MindSpore; PyTorch

目录

Contents

第一章 绪论

1.1 课题研究背景与意义

如今，人工智能发展迅速，广泛应用于各个领域，深刻影响着人们的日常生活模式，而作为其中最为火热的深度学习，其网络结构日趋复杂，给 AI 计算框架带来巨大挑战。目前业界主流的框架为国外的 TensorFlow、PyTorch、MXnet 和 CNTK，为实现易开发、高效执行和全场景部署的功能，华为技术有限公司推出了新一代深度学习框架 MindSpore[1]，从底层开发出三大技术优势，发展势头正盛，引领着国内 AI 计算领域的发展。

不成对图像去雾越来越受到人们的关注，因为监督去雾方法需要成对的训练数据作为前提，而捕获真实数据代价会很高，使用合成图片在真实的模糊场景上性能也会下降。现有的不成对图像去雾方法都是基于类 CycleGAN 框架，带有像素到像素的约束，导致模型繁琐，训练不稳定。对此，杨文锦[2]所在团队提出了一种新的用于非配对图像去雾的单 GAN 模型(SinGAN-Dehaze)，该模型消除了循环一致性约束。具体地说，循环一致性被解耦为内容一致性和样式一致性，其中像素到像素的映射被斑块到斑块的语义映射取代，通过捕获局部特征表示和全局上下文依赖关系来确保内容一致性，通过强迫去雾结果的高频信息分布与风格相似的清晰图像的高频信息分布接近来实现风格一致性。大量实验表明，在不成对去雾方法中，无论是在合成的还是真实的有雾场景里，这种方案在客观指标和视觉效果方面都能取得优异的性能。

目前，中国智能制造业领域所采用的重要核心产业软件系统仍较高依靠进口，面临自主可控开发的“瓶颈”问题，亟待解决。联系课程组与华为科技有限公司的合作项目，将作为深度学习前沿领域的图像识别与处理方向，同国产开源计算框架结合，可以推动更多业界人士关注和使用 MindSpore，进而推进版本完善，加快功能部署，让代码国产化向前走一步，有利于逐步跟上 AI 技术的国际化趋势，让 MindSpore 框架能和其他深度学习框架同台竞技，为未来技术和经济的发展打下磐石般的基础。同时，将 SinGAN-Dehaze 从 PyTorch 框架移植到 MindSpore 上，能将此图像处理方法应用在华为全栈硬件平台，充分利用昇腾处理器[3]的强大算力，保证此方法的有效性、可应用性。

1.2 MindSpore 框架概述

MindSpore 是全场景的深度学习框架，API 使用以及调试复杂度的降低使得 AI 开发过程较为简便，数据分析和计算、数据预处理和分布式训练都能够做到高效率进行，云、边缘以及端侧协同支持从而实现了全场景覆盖，如图 1 所示。

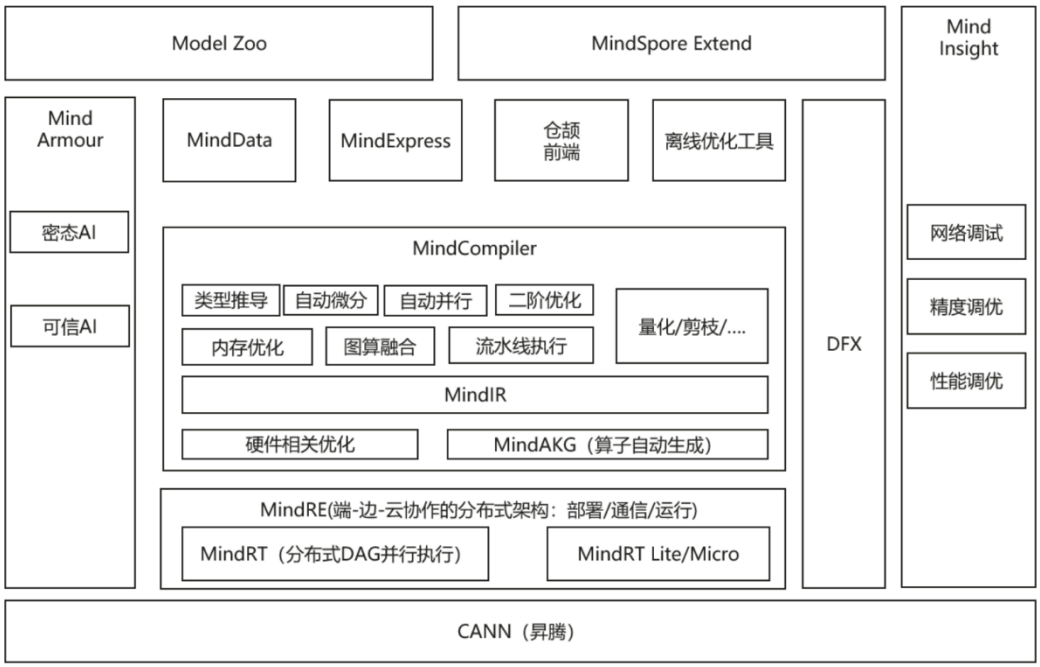


图 1 MindSpore 架构[4]

与市面上目前最主要的深度学习框架相比，作为华为整个产业的最佳实践，MindSpore 支持了 Python 程序设计范式，开发人员能够利用 Python 的原生控制编程逻辑，在简单的 AI 编程下实现繁杂的神经网络模型的建立；同时为了防止开发者混淆，MindSpore 介绍了尽可能少的接口和概念。具体来说，有以下几大优势：

首先，MindSpore 统一了动、静态图的编码方式，大大提高了二者的可兼容性，不管是容易调试但不易高效执行的动态图，或是训练性能较高但调试困难的静态图，MindSpore 均给予支持，同时模式之间的切换只需要一行代码。

其次，MindSpore 引入了函数型可微分编程框架，采用了源码转换的函数自动微分机制，精简了神经网络模型的梯度下降算法进行复杂手动求导的过程，开发者可只关心模型算法的数学原生表达式，而无需手动求导此步骤。

最后，MindSpore 规整了单机和分布式训练的编码方法，相比其他框架中繁

杂的分布式策略，开发人员可以在原来的单机代码上进行少量调整实现分布式训练，提高神经网络训练效率，AI 开发不再困难重重无从下手，模型思路随之清晰易得。

1.3 SinGAN-Dehaze 网络简介

不成对图像去雾技术是计算机视觉应用领域的一项科研热点，其目的是将模糊图像还原为无雾图像，对于目标辨识、情景认知等高级计算机视觉任务都有着重要意义。随着深度学习的发展，图像去雾的研究取得了很大的进展。目前主流的图像去雾方法[5-9]属于监督学习方法，需要成对的训练数据，即清晰图像和模糊图像。然而，真实世界的配对数据，即雾霾图像及其对应的清晰图像，几乎是不可得的。因此，大多数流行的去雾方法都依赖基于物理散射模型生成的模糊图像。

$$I(x) = J(x)t(x) + A(1 - t(x)) \tag{1}$$

其中x为像素位置，$I(x)$为朦胧图像，$J(x)$为无雾图像，A为全球大气光，$t(x)$为透射图像。

这些使用合成成对训练数据的去雾模型虽然取得了显著的改进，但由于真实数据和合成数据之间不一致的雾霾分布导致的性能下降，它们难以很好地推广到实际的模糊图像中。因此，不成对图像去雾技术受到了越来越多的关注。

当下，非配对图像去雾研究仍处于初级阶段，现有的非配对去雾方法大都基于提供了循环一致性监督的 CycleGAN[10]框架，例如，根据有雾输入生成相应的去雾图像，然后再将去雾图像转回模糊图像。然而，它也存在着两个不可忽视的局限性。首先，类 CycleGAN 方法关注像素到像素的约束，在特征表示上低于斑块到斑块的约束，与成对的去雾方法相比，容易产生意料之外的伪影。其次，类 CycleGAN 方法中的双 GAN 通常会导致计算复杂度高和训练不稳定。如图 2 所示，对比基于 CycleGAN 和 SinGAN-Dehaze 的去雾网络在架构上的方法，前者一般由两个生成器和两个鉴别器组成，用于跨域转换模糊和清晰的图像，这加重了计算资源的负担，同时训练阶段缺乏足够的监督信息导致模型优化困难。

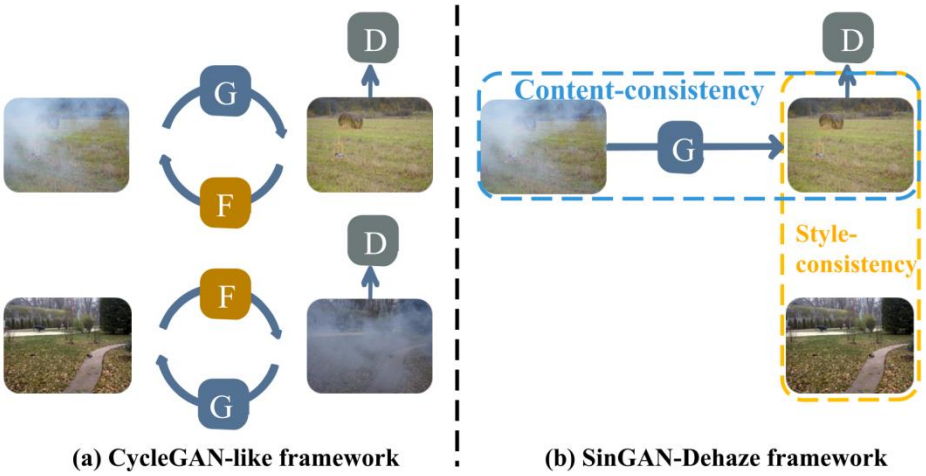


图 2 CycleGAN-like 和 SinGAN-Dehaze 方法的原理图比较[2]

为克服以上问题，杨文锦[2]提出了一种新颖的只有单一 GAN 的不成对图像去雾网络(SinGAN-Dehaze)，它打破了循环一致性约束，设计了内容一致性和样式一致性约束。在内容一致性约束下，通过对比学习最小化原始斑块与去雾斑块之间的距离。具体地说，通过建立一个前置任务来比较去雾图像和有雾图像，其目标是使去雾图像中的锚点与对应块之间的差异最小化，同时使其与其他受扰动块之间的差异最大化。在风格一致性约束方面，将高频信息作为风格特征，增强去雾图像与未配对的清晰图像之间高频分布的一致性。在实现时，采用域传输模块来缩小高频信息分布的差距。经多次实验表明，此方案可以在合成数据集和真实数据集上达到与最先进的方法相比的卓越性能。

第二章 相关工作

2.1 MindSpore 编程概述

MindSpore 网络运算的基本数据结构为张量，其与数组和矩阵非常相似，属性包括形状、数据类型、转置张量、单个元素大小、占用字节数量、维数、元素个数和每一维步长，基本运算包括算术、线性代数、矩阵处理、采样等。张量的创建方式有多种，构造张量时，可以传入 Tensor、float、int、bool、tuple、list 和 NumPy.array 类型。与 PyTorch 相比，MindSpore 优化了数据对象的设计逻辑，仅保留了 PyTorch 的两种数据对象：Tensor 和 Parameter，其中 Tensor 对象仅参与运算，不需要对其进行梯度求导和参数更新，而 Parameter 数据对象和 PyTorch 的 Parameter 意义相同，会根据其属性 requires_grad 来决定是否对其进行梯度求导和参数更新。在网络迁移时，在 PyTorch 中未进行参数更新的数据对象，在 MindSpore 中均可声明为 Tensor。在建立张量时，MindSpore 对网络传播过程中的数据类型有非常严格的要求，参数的数据类型也不会自动修改，而在 Tensor 的参数定义上值得注意的一点是，mindspore.Tensor.size 用来查看元素的总数，而 torch.Tensor.size 查看的是 Tensor 的形状。

在网络训练和推理过程中，由于原始数据通常存储在磁盘和数据库中，需要先通过数据加载接口将其读取到内存空间，转换成框架通用的张量格式，然后通过数据处理和增强步骤，将其映射到更加易于学习的特征空间，同时增加样本的数量和泛化性，最后输入到网络进行计算。常用数据集和标准格式的数据集加载可以使用 mindspore.dataset 接口快速进行数据处理操作。Dataset 将数据加载到内存后，数据按 Tensor 形式进行组织。

使用 mindspore.nn 提供的各种网络基础模块可以创建由多个数据操作层组成的神经网络模型，MindSpore 的 Cell 类是构建所有网络的基类，也是网络的基本单元，网络中的元素定义在__init__函数中并对其初始化，网络的图结构表达定义在 construct 函数中，通过调用这些类的对象完成整个模型的构建和训练。与之对应的，在 PyTorch 框架上构建神经网络时，需要继承 nn.Module 类，并重写__init__方法和 forward 方法。

在网络结构建立的微分角度上，MindSpore 先构建的是基于反向自动微分的

GradOperation 方法，并在该方法的基础上实现正向微分。自动微分可以计算可导函数在某点处的导数值，是对反向传播算法的一般化，它解决了复杂的数学运算问题，将其分解为一系列简单的基本运算，该功能对开发人员屏蔽了大量的求导细节和步骤，大大降低了框架的使用门槛。对于反向微分的实现，首先取出自动微分流程中需要进行处理的函数，并作为自动微分模块的输入，为每个原函数中的节点生成其所对应的梯度函数的节点，再按照反向自动微分的规则将这些节点连接在一起，从原函数对象转换到梯度函数对象，生成梯度函数图[11]。

网络创建后离不开模型训练与调优，MindSpore 的网络优化逻辑封装在 Optimizer 对象中，mindspore.nn 提供了诸多常用的优化器函数来帮助完成网络优化，如 Adam、SGD、Momentum 等。优化器函数会计算和更新梯度，模型优化算法的选择直接关系到最终模型的训练效果，因此有时候最终模型效果较差，未必是特征或者模型设计的问题，而很有可能是优化算法的问题。

训练网络模型的过程中希望保存中间和最后结果，用于微调和后续模型部署和推理，这时可以使用 MindSpore 提供的 save_checkpoint 直接传入网络和保存路径保存模型，或者使用 model.train 里的 callbacks 参数传入保存模型的对象 ModelCheckpoint（一般与 CheckpointConfig 配合使用）保存模型参数，生成 CheckPoint(即 ckpt)文件。要加载模型权重，需要先创建相同模型的实例，然后使用 load_checkpoint 和 load_param_into_net 方法加载参数[11]。

2.2 图像去雾深度模型

作为深度学习的代表算法之一，卷积神经网络(CNN)的兴起为图像去雾的进一步发展铺平了道路，例如，在 AOD-Net[5]中，重新设计了一个大气散射模型，在一个统一的模型中估计所有参数。在 EPDN[6]中，设计了一个增强的 pix2pix 去雾网络。在 FFA-Net[7]中，提出了一种特征融合注意网络，为处理不同类型的信息提供了额外的灵活性。但是这些方法都依赖于成对的训练数据，其中的模糊图像是通过物理散射模型合成的。因为合成数据和真实数据之间的领域偏移，这些受监督模型在真实应用程序中的性能可能会大大降低。

现有的无监督去雾方法大致可以分为传统的基于先验的方法和类

CycleGAN 方法。传统的去雾方法通过手工设计各种先验来估计大气光和透射图，然后根据大气散射模型[12]恢复出清晰图像。这些方法只有在满足前提条件的情况下才能发挥良好的性能，这本身就限制了它们的应用。类 CycleGAN 去雾模型打破了成对训练对监督去雾方法的约束。如，CycleDehaze[13]引入了一个循环感知一致性损失来增强内容约束。循环一致性损失通常是用 l1 范数计算，因此又称为 L1loss，当把输入 x 送入生成器 G 后，得到假的 y 图，再把这张假的 y 图送入到生成器 F 中，得到更假的 x 图。理想情况下，此时更假的 x 图应该与原始的 x 图相差无几，这样也便构成了一个循环，这个差值就是循环一致性损失。然而，这些类 CycleGAN 的方法只考虑全局信息，而忽略了局部数据的变化。此外，循环一致性假设两个域之间的关系是一个双边映射，这导致两个成对的生成器和鉴别器增加了内存资源的负担。针对上述问题，SinGAN-Dehaze 提出了具有内容一致性和风格一致性约束的不成对图像去雾网络，来消除循环一致性约束的限制。

对比学习[14]是一种在无标签情况下通过学习数据点间相似或不同来区别数据集一般特征的自监督学习方法，可以获得图像的鲁棒特征表示，因此获得了广泛的应用。鲁棒性的特征，即在环境变化时，仍然能够对信息进行过滤。在找到鲁棒特征的前提下，即使狗的照片被标记为猫，使用这样错误的训练数据训练出的模型，也能够将猫正确分类。对比学习方法在高级视觉任务中有广泛的应用，而在近年，Kyungjune[15]提出了真正无监督的图像翻译方法，证明了对比学习也适用于低级视觉任务。CUT[16]对图像翻译采用对比学习，但不需要严格的内容一致性约束。在此基础上，SinGAN-Dehaze 设计了一种新的去雾网络，消除了循环一致性约束，并设计了内容保持模块和样式传递模块。

2.3 从 PyTorch 到 MindSpore 代码迁移

为实现 SinGAN-Dehaze 在 MindSpore 上的网络迁移，首先需要对原网络脚本进行分析，理解原网络构建方式，包括数据集加载、预处理，正向网络结构，loss 损失函数定义，优化器定义，数据集对象迭代方式，反向传播梯度计算方法以及参数更新等，综合评估缺失算子并补齐框架功能；然后着手进行网络脚本开发，对数据做好处理，构建对应的深度学习网络；紧接着调试执行网络，通过单

步和多轮迭代调试，打通单卡、多卡流程；之后调节网络参数，对网络精度性能进行调优，将网络转移到云平台训练；最后，当网络精度性能达标，即可交付迁移网络。根据 MindSpore 迁移文档的 API 映射，大部分算子的实现在参数、输入、输出、逻辑功能和特定场景等方面与 PyTorch 语法存在差异，因此清楚待实现网络的具体构建及各 API 的具体使用是需要攻克的重难点。

第三章 SinGAN-Dehaze 图像去雾方法

在 SinGAN-Dehaze 图像去雾方法中，循环一致性解耦到内容一致性和样式一致性约束的体系结构[2]如图 3 所示，由 GAN 网络、内容保持模块(CPM)和风格传递模块(STM)三部分组成。CPM 通过局部和全局内容一致性约束，强制生成的图像保留与模糊输入相同的低级内容信息。利用 STM 将生成结果的高频分布拉至清晰图像附近，单独约束可以进一步提高去雾性能。需要注意的是，整个网络只包含一个 GAN，即只有一个发生器和一个鉴别器(由于空间限制，图 3 中省略了鉴别器)，其基本结构类似于 CycleDehaze[13]。生成器是一个编码器－解码器结构。对输入信号进行了步长为 2 的卷积层 4×下取样。然后在网络末端进行相应的上采样。中间特征映射由 9 个残差块[17]组成。在训练过程中，首先将一幅模糊图像通过 GAN，输出去雾后的图像，然后将其送入 CPM 和 STM。前者对去雾结果进行内容保持约束，后者对去雾结果进行样式传递约束。

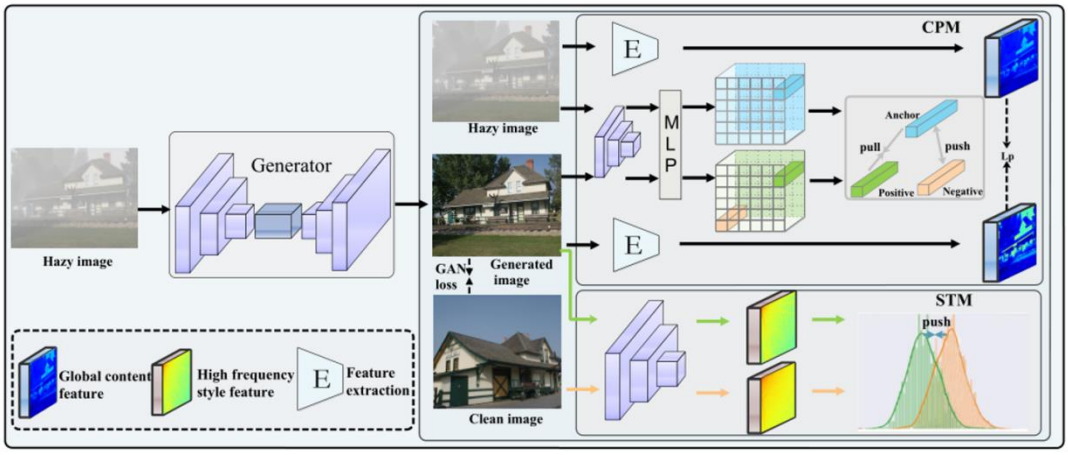


图 3 SinGAN-Dehaze 的网络结构[2]

为了加强内容一致性约束，SinGAN-Dehaze 方法进行了局部扰动对比学习(LPCL)和全局上下文感知学习(GCPL)，共同确保去雾结果的内容保真度。CPM 对输入的模糊图像和生成的去雾结果之间的全局和局部纹理信息进行约束。而 LPCL 则先采用了类似于 GAN 的共享编码器，随后采用了具有一个隐藏层的共享多层感知器(MLP)提取特征，再进行对比学习。GCPL 在去雾—模糊的成对图像上增强感知损失。对比学习包含两个重要因素：如何构建有效的“正”和“负”样本，以及如何度量样本之间的潜在特征分布。因此，SinGAN-Dehaze 设计了一

个局部扰动对比学习机制来捕获斑块级的特征表示。如图 3 所示，将雾霾图像和生成的去雾结果分别输入到共享编码器和共享 MLP 中，获取其特征向量。在 MLP 输出特性上进行对比学习。为了将模糊的输入和输出联系起来，定义输入的雾霾斑块作为锚点，生成的图像对应位置作为正样本。然后，在输出的其他位置对负样本进行采样。根据 InfoNCE[18]，局部对比损失公式为：

$$L_{cl} = -log\frac{\exp\left(s \cdot s^{+}/\tau\right)}{\exp(s \cdot s^{+}/\tau) + \sum_{i=1}^{N}\exp\left(s \cdot s_{i}^{-}/\tau\right)} \tag{2}$$

其中s，s^{+}，s_{i}^{-}分别表示本地编码的查询特征，对应的正特征和负特征。τ为温度参数，如果在训练的过程中将温度参数设置比较大，那么预测的概率分布会比较平滑，会导致损失变大，从而梯度也越大，参数更新步长就会较大，避免陷入局部最优解，根据经验设定为 0.07。N是负样本个数。全局特征编码为 256 个局部向量。为了充分挖掘内部数据特征，正负样本数量之比为 1:255。考虑到图像的内在自相似性，所采样的正样本可以与其他负样本内容一致。对比学习的目的是将具有正向特征的查询特征拉到锚点附近，同时将具有负向特征的查询特征推到锚点附近。相似斑块引起的自相似性会进一步阻碍模型的表示。因此，通过在负样本特征中加入随机高斯噪声来进行扰动，以达到稳定训练的目的。

一个小斑块仅仅包含局部的邻域线索，而图像去雾过程还包含全局的上下文和语义信息，这是其所不能包含的。因此，全局内容的保留在去雾中也起着至关重要的作用。由于缺少成对的清晰图像作为监督，考虑利用模糊输入的内在特征。因此，通过预训练的 VGG 模型来测量感知相似度，以缩小模糊输入与生成的图像之间的特征空间距离，在生成的图像与对应的原始模糊输入之间添加一个感知损失：

$$L_{p} = \sum_{i=1}^{M}\frac{\lambda_{i}}{C_{i}H_{i}W_{i}}\left\| \phi^{i}(X) - \phi^{i}(\hat{Y}) \right\|_{1} \tag{3}$$

式中X、$\hat{Y}$分别为雾天图像及相应的生成器去雾结果。M为 VGG16 提供的特征层数量，H_{i}和W_{i}为第 i 个特征图的高度和宽度，C_{i}为特征通道的数量，$\phi(\bullet)$代表在 VGG16 网络中所获得的所有特征。底层特征倾向于提取高分辨率的详细信息，

而顶层特征则偏向于获取较低分辨率的语义信息。所以，采用加权 VGG16 特征对损耗进行优化，对于第 2、第 7、第 14 层，λ_1、λ_2、λ_3分别为 1、1/2、1/4。

局部扰动对比学习可以增强获取局部内容一致性的能力。通过感知一致性约束获得全局上下文依赖关系。通过这两个约束，生成的结果可以有效地保持内容的一致性。

除了对生成的图像保留内容外，对于去雾，生成的图像与清晰图像之间的风格间隙的缩小也很重要。清晰数据集需要满足模糊数据集的类似场景。例如，清晰和朦胧的图像都是户外场景。将生成的去雾图像定义为源域，将无雾图像定义为目标域。最终的目标是找到一个转换函数来缩小生成的数据与无雾数据之间的差异。在 SinGAN-Dehaze 方法中，STM 只包含一个编码器，与生成器的对应部分相同，并以图像的高频分布作为风格，滤除低频内容。如何衡量两种风格之间的差距是个重要的问题。SinGAN 采用最大均值差异(Maximum Mean difference, MMD)来测量两种风格之间的差距，因为 MMD 被证明更适合测量分布的差异，即生成图像的风格和干净图像的风格。为了约束生成的图像和清晰图像的风格，首先通过高斯滤波器将低频内容从高频语义风格中分离出来。然后，将两幅高频图输入编码器进行特征提取。编码器的重用主要出于两个考虑。首先，它可以作为一个正则化器来约束编码器的训练，这意味着编码器不仅学习内容特定的特征，而且还学习领域不变的特征。其次，通过权值共享，降低模型的复杂性。因此，风格间隙的收缩率损失公式为：

$$\begin{aligned} L_{mmd}\left(g(Y), g(\hat{Y})\right) &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n} k\left(g(y_i), g(y_{i'})\right) \\ &- \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k\left(g(y_i), g(\hat{y}_j)\right) \\ &+ \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m} k\left(g(\hat{y}_j), g(\hat{y}_{j'})\right) \end{aligned} \tag{4}$$

其中$k(u,v)$=$e^{\frac{-\|u-v\|^2}{\sigma}}$为高斯核映射函数。$g$（•）为编码器提取的特征。$Y$和$\hat{Y}$是没有雾霾的图像和生成的图像。$n$和$m$分别为有雾样本数和清晰样本数。

为了拟合生成结果与目标图像的分布，首先采用 GAN 的对抗性损失。发生器的主要目的就是把模糊图像转换为无雾图像，而鉴别器则被训练来区分图像的“真”与“假”。为改善生成图像的品质和训练的稳定性，采用 LSGAN[19]的目标：

$$L_{adv} = \mathbb{E}_{y\sim P_{data}(Y)}[logD(y)] + \mathbb{E}_{x\sim P_{data}(X)}\left[\log\left(1 - D\left(G(x)\right)\right)\right] \quad (5)$$

其中X和Y分别表示来自另一个相似风格数据集的模糊图像和清晰图像。

因此，优化模型的总损失函数可进一步表示为：

$$L = L_{cl} + L_p + L_{mmd} + L_{adv} \quad (6)$$

这也就是 SinGAN-Dehaze 网络实现的技术目标与难点。

第四章 SinGAN-Dehaze 的 MindSpore 实现

4.1 数据集对象

要证实 SinGAN-Dehaze 方法的准确性，在合成数据集和真实数据集中分别加以评估。Dhazy[20]是一个公共可用数据集，用于去雾方法的定量评价。它提供了 1449 幅无雾/有雾图像，是常用于无监督去雾的数据集。数据集按 8:2 的比例随机分为训练集和测试集，训练集随机打乱成不配对的模式。此外，RESIDE[21]是一个广泛应用的图像雾化数据集，它包含五大子集：室内训练集(ITS)、室外训练集(OTS)、综合目标检测集(SOTS)、混合主观测试集(HSTS)和真实任务驱动检测集(RTTS)。对于合成去雾，选择 1500 张不配对的合成模糊图像进行训练，分别来自 ITS 和 SOTS。为评估对真实数据集的鲁棒性，采用了 NTIRE2018 挑战赛中提出的 O-HAZE[22]数据集。它包含 45 对真实世界的模糊和无模糊图像，使用最后 10 幅图像作为测试数据集。

对非常用数据集，MindSpore 提供了 GenerateDataset 来自定义 Python 数据源，通过迭代该数据源构造数据集，虽然功能类似，但它和 PyTorch 的 Dataloader 并不是同一个概念。GenerateDataset 加载数据集后生成最基本的数据流，输出为单个样本数据，还可以继续使用 MindData 提供的接口进行其他预处理的数据增强操作，例如 map、batch、shuffle、repeat 等。而 Dataloader 通常是数据处理的最终出口，输出 batch_size 个样本，然后直接送入网络。

4.2 网络脚本开发

使用 MindSpore 框架搭建神经网络流程与其他框架类似，但支持的算子存在差异，在进行从 PyTorch 到 MindSpore 的代码移植时需要找出缺失的算子。MindSpore API 由各种 Python/C++API 算子组成，大致分为：数据框架算子，包括张量、基本数据类型、训练梯度、优化器算子；数据预处理算子，包括图片读取、数据类型转化算子；网络结构算子，包括网络构建中使用到的卷积、归一化算子。根据 API 映射，相比原网络框架，MindSpore 算子至少存在以下缺失[11]：

（1）图像增强部分，如 torchvision.transforms.Lambda；

（2）学习率调整，如 torch.optim.lr_scheduler.ReduceLROnPlateau，在 mindspore.nn 无对应；

（3）数据采样，如 torch.nn.functional.interpolate 可以实现多种采样，而对应到 mindspore.nn.ResizeBilinear 只有 bilinear 模式。

对于缺失算子，有以下处理办法：考虑用其他算子替换，分析算子实现公式，审视是否可以用现有 MindSpore 算子叠加达到预期目标；考虑临时规避方案，如某个 loss 不支持，可以替换为同类已支持的 loss 算子；以及提交开发人员，等待新版本的发布。对于 SinGAN-Dehaze 网络迁移遇到的问题，采用的方法主要是算子规避和小算子拼接。

对于典型算子和接口，MindSpore 和 PyTorch 也存在不小差别。PyTorch 在构建模型时，通常会利用 torch.device 指定模型和数据绑定的设备是 CPU 还是 GPU，以及绑定的具体的 GPU 序号。而在 MindSpore 中，只需要通过 context 中的 device_target 参数指定模型绑定的设备，用 device_id 指定设备序号。一旦设备设置成功，输入数据和模型就会拷贝到指定的设备中执行，不需要也无法改变数据和模型所运行的设备类型，这与 PyTorch 绑定相应设备后仍要将模型和数据部署到对应设备上的方式不同。MindSpore 中的 nn.Cell 类用来构建图结构，其中提供了丰富的 API 供开发者使用，常用的功能也可以找到映射，其网络的图结构表达定义在 construct 函数中。此外，最需要注意的问题是对应算子的初始化含义，不仅包括算子参数意义，还要理解算子行为。在网络迁移时常用的 API 映射变换[23]及需要注意的差异如下：

（1）mindspore.ops.Concat 对应 torch.cat，作用是在指定轴上拼接输入 Tensor，要求输入 tensor 的数据类型保持一致，不能像 PyTorch 一样低精度 tensor 自动转为高精度 tensor，不一致时可通过 ops.Cast 转换精度后再调用 Concat 算子。

（2）mindspore.ops.ReduceMean 对应 torch.std_mean，作用是输出 Tensor 各维度上的平均值来对所有维度进行归约，或者对指定维度进行求平均值归约，但不会像 PyTorch 一样同时计算标准差。

（3）mindspore.ops.GradOperation 对应 Pytorch 的 torch.autograd.grad 以及 torch.autograd.backward，为输入函数生成梯度函数，此方法生成的梯度函数可以通过

构造参数自定义，同时参数设置更为简单。

（4）mindspore.context.set_context 对应 torch.cuda.set_device，不仅可以像 Pytorch 般设置当前使用的 device 卡号，还设置模式 mode，运行环境 device_target，是否保存图 save_graphs 等。

（5）mindspore.nn.BatchNorm2d 对应 torch.nn.BatchNorm2d，在四维输入上应用批归一化处理避免内部协变量偏移，特别注意其 momentum 参数的默认值为 0.9，与 Pytorch 的 momentum 关系为 1-momentum。

（6）mindspore.nn.Conv2d 对应 torch.nn.Conv2d，对输入 Tensor 计算二维卷积，默认对输入进行填充，使输出与输入维度一致，默认 has_bias 为 False，这些与 PyTorch 都是相反的。

（7）mindspore.nn.Dense 对应 torch.nn.Linear，应用在全连接层，对传入数据应用线性变换，在输出数据之前可以选择应用激活函数 activation，默认权重矩阵由标准正态分布初始化，偏移矩阵初始化为 0，而 PyTorch 默认权重矩阵和偏移矩阵都由均匀分布初始化。

（8）mindspore.nn.MaxPool2d 对应 torch.nn.MaxPool2d，对输入的多维数据进行二维的最大池化运算，没有 padding 参数来调整输出的 shape，仅通过 pad_mode 参数控制 pad 模式。

（9）mindspore.nn.Pad 对应 torch.nn.functional.pad，根据 shape 为(n, 2)的元组 paddings 参数对输入进行填充，其中 n 为输入数据的维度。要使 MindSpore 的输出 shape 与 Pytorch 的一致，需要对应 PyTorch 的 pad 参数设定 paddings。

（10）mindspore.nn.Optimizer 对应 torch.optim.optimizer，用于参数更新的优化器基类，入参设置存在较大差异。

（11）mindspore.Tensor.sum 对应 torch.Tensor.sum，返回给定轴上的张量元素之和，可以通过入参 initial 配置求和的起始值，其他入参两接口设定相同。

整个网络最关键的部分是损失函数的定义与使用，包括$\boldsymbol{L_{adv}}$、$\boldsymbol{L_{cl}}$、$\boldsymbol{L_{p}}$和$\boldsymbol{L_{mmd}}$。在 MindSpore 中，以上 loss 分别作如下定义：

```
class GeneratorLoss(nn.Cell):
    """
    SinGAN-Dehaze generator loss.
    """
    def __init__(self, args, generator, D_A, D_B):
        super(GeneratorLoss, self).__init__()
        self.lambda_A = args.lambda_A
        self.lambda_B = args.lambda_B
        self.lambda_idt = args.lambda_idt
        self.use_identity = args.lambda_idt > 0
        self.dis_loss = GANLoss(args.gan_mode)
        self.rec_loss = nn.L1Loss("mean")
        self.generator = generator
        self.D_A = D_A
        self.D_B = D_B
        self.true = Tensor(True, mstype.bool_)

    def construct(self, img_A, img_B):
        """If use_identity, identity loss will be used."""
        fake_A, fake_B, rec_A, rec_B, identity_A, identity_B = self.generator(img_A, img_B)
        loss_G_A = self.dis_loss(self.D_B(fake_B), self.true)
        loss_G_B = self.dis_loss(self.D_A(fake_A), self.true)
        loss_C_A = self.rec_loss(rec_A, img_A) * self.lambda_A
        loss_C_B = self.rec_loss(rec_B, img_B) * self.lambda_B
        if self.use_identity:
            loss_idt_A = self.rec_loss(identity_A, img_A) * self.lambda_A * self.lambda_idt
            loss_idt_B = self.rec_loss(identity_B, img_B) * self.lambda_B * self.lambda_idt
        else:
            loss_idt_A = 0
            loss_idt_B = 0
        loss_G = loss_G_A + loss_G_B + loss_C_A + loss_C_B + loss_idt_A + loss_idt_B
        return (fake_A, fake_B, loss_G, loss_G_A, loss_G_B, loss_C_A, loss_C_B, loss_idt_A, loss_idt_B)
```

图 4 SinGAN-Dehaze 方法的L_{adv}定义

```
class PatchNCELoss(nn.Cell):
    def __init__(self, opt):
        super().__init__()
        self.opt = opt
        self.cross_entropy_loss = nn.SoftmaxCrossEntropyWithLogits(reduction='none')
        self.mask_dtype = ms.uint8 if version.parse(ms.__version__) < version.parse('1.2.0') else ms.bool_
        uniform_range = 0.3
        self.uni_dist = Uniform(-uniform_range, uniform_range)

    def feature_based_noise(self, x):
        noise_vector = self.uni_dist.sample(x.shape[1:]).to(x.device).unsqueeze(0)
        x_noise = x.mul(noise_vector) + x
        return  x_noise

    def feature_dropout(self, x):
        attention = ops.ReduceMean(keep_dims=True)(x, axis=1)
        _, max_val = ops.ArgMaxWithValue(axis=1, keep_dims=True)(attention.view(x.size(0), -1))
        threshold = max_val*np.random.uniform(0.7,0.9)
        threshold = threshold.view(x.shape(0), 1, 1, 1).expand_as(attention)
        drop_mask = (attention < threshold).float()
        return x.mul(drop_mask)

    def construct(self, feat_q, feat_k):
        # print(feat_q.shape,feat_k.shape)
        batchSize = feat_q.shape[0]
        dim = feat_q.shape[1]
        feat_k = feat_k.detach()
```

图 5 SinGAN-Dehaze 方法的L_{cl}定义

```
# --- Perceptual loss network  --- #
class LossNetwork(nn.Cell):
    def __init__(self, vgg_model):
        super(LossNetwork, self).__init__()
        self.vgg_layers = vgg_model
        self.layer_name_mapping = {
            '3': "relu1_2",
            '8': "relu2_2",
            '15': "relu3_3"
        }

    def output_features(self, x):
        output = {}
        for name, module in self.vgg_layers._modules.items():
            x = module(x)
            if name in self.layer_name_mapping:
                output[self.layer_name_mapping[name]] = x
        return list(output.values())

    def construct(self, pred_im, gt):
        loss = []
        pred_im_features = self.output_features(pred_im)
        gt_features = self.output_features(gt)
        for pred_im_feature, gt_feature in zip(pred_im_features, gt_features):
            loss.append(nn.MSELoss(pred_im_feature, gt_feature))

        return sum(loss)/len(loss)
```

图 6 SinGAN-Dehaze 方法的L_p定义

```
def mmd(source, target, kernel_mul=2.0, kernel_num=5, fix_sigma=None):
    batch_size = int(source.size()[0])
    kernels = guassian_kernel(source, target,
                              kernel_mul=kernel_mul,
                              kernel_num=kernel_num,
                              fix_sigma=fix_sigma)
    XX = kernels[:batch_size, :batch_size]  # Source<->Source
    YY = kernels[batch_size:, batch_size:]  # Target<->Target
    XY = kernels[:batch_size, batch_size:]  # Source<->Target
    YX = kernels[batch_size:, :batch_size]  # Target<->Source
    loss = ops.ReduceMean(keep_dims=True)(XX + YY - XY - YX)
    # 这里是假定X和Y的样本数量是相同的
    # 当不同的时候，就需要乘上上面的M矩阵
    return loss
```

图 7 SinGAN-Dehaze 方法的L_{mmd}定义

最终 G 的整体 loss 为四项之和。

4.3 模型训练

网络模型在 CPU 平台下 1.6 版本的 MindSpore 框架上试运行，然后将代码及数据集文件上传至华为云计算平台 ModelArts[24]上，使用其上配置 1*Ascend-910(32GB)+CPU:24 核 96GB 完成后续模型训练与参数调试。训练样本被裁剪为 256×256 的小块， 每次将 1 个小块作为输入，采用 Adam 优化器训练 SinGAN-

Dehaze，动量β_1=0.9，β_2=0.999，初始学习率设为2×10^{-4}。对室外数据集，训练迭代次数默认设置 200 个 epoch。对室内数据集，训练使用 400 个 epoch。

第五章 网络训练及性能测试

5.1 线上部署

5.1.1 ModelArts 平台

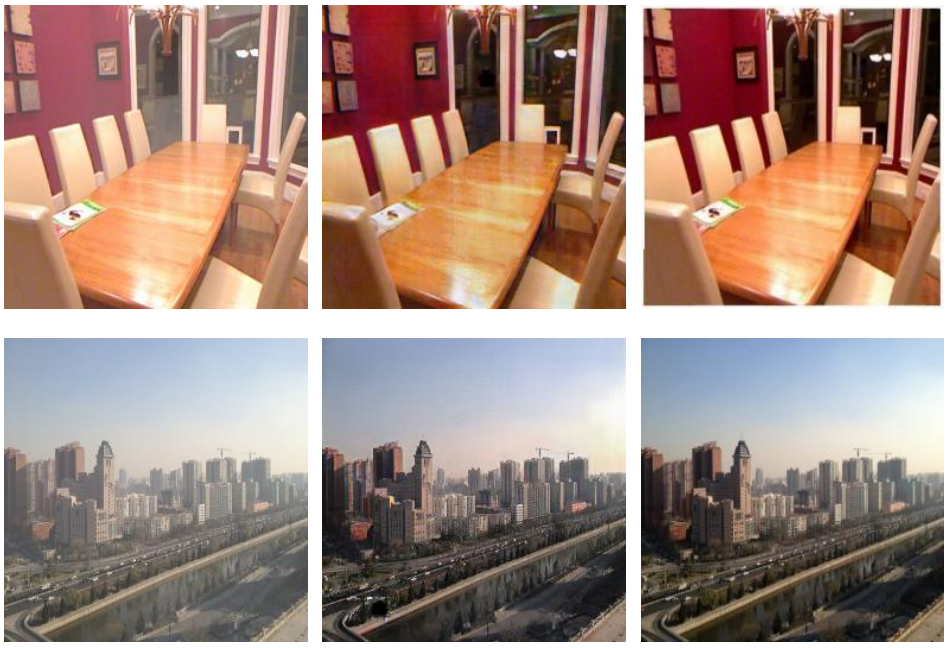
ModelArts 开发平台[24]，是由华为技术有限公司推出的 AI 一站式云端训练产品，使用者能够完成全流程的人工智能技术开发服务，包含大数据分析处理、模型训练、模型管理、模型部署等操作，并同时提供 AI Gallery 功能，可与其他开发人员共享模型。为了降低开发难度，提升开发人员的模型训练效率及训练性能，ModelArts 实现了可视化作业信息管理、资源信息管理、版本信息管理等功能，基于机器学习算法及强化学习的模型训练自动超参调优，如 learning rate、batch size 等自动的调参策略；预置和调优常用模型，简化模型开发和全流程训练管理。ModelArts 的 MoXing 进行了全栈优化，极大减少了训练收敛时间。在数据读取和预处理方面，MoXing 可以通过输入多级并发流水线保证数据 IO 不形成阻碍；在模型计算方面，MoXing 对上层模型提供半精度和单精度构成的混合精度计算，采用自适应的尺度缩小由于精度计算带来的损失；在超参调优方面，采用动态超参策略使模型收敛所需 epoch 个数最小化；在底层的优化设计上，MoXing 通过与底层华为服务器和通信计算库结合，使得分布式加速进一步增强。

5.1.2 部署流程

根据华为云 ModelArts 使用文档，通过课题组项目取得账号权限，首先使用对象存储服务将数据上传至创建好的 OBS 桶中，上传的数据包括数据集对象及代码文件；然后创建 Notebook 开发环境，修改代码从而适配云端训练；紧接着创建训练作业，设置网络参数及输出位置，开始训练生成模型；最后进行模型管理及部署上线。

5.2 模型性能测试与分析

用测试集来预测去雾结果，比较生成图像与真实图像的差异，如图 8 所示。



Hazy　　SinGAN-Dehaze　　GT

图 8 SinGAN-Dehaze 图像去雾结果

从峰值信噪比（PSNR）、结构相似性（SSIM）和色差比较 CIEDE 几个方面评价图像去雾水平。

PSNR 是目前应用最为广泛且常用的评价画质的客观量测法，一般用于评估图像处理程序是否令人满意，其计算方法是原图像与被处理图像之间的均方误差对应$(2^n-1)^2$的对数值(信号最大值的平方，n是每个采样值的比特数，MSE是原图像（声音）与处理图像（声音）之间均方误差)，它的单位是 dB。

$$PSNR = 10 \times log_{10}(\frac{(2^n-1)^2}{MSE}) \tag{7}$$

SSIM 也是图像辨别通常使用的确定两张图片相似程度的方法，当两幅图片完全相同时，SSIM 的值为 1。给定两张图片 x 和 y，这两张图片的结构相似性可根据以下方法求出（μ_x，μ_y分别是x，y的平均值，σ_x^2和σ_y^2分别是x，y的方差，σ_{xy}是x，y的协方差。$c_1 = （k_1L）^2$，$c_2 = （k_2L）^2$是用来保持稳定的常量，$k_1 = 0.01$，$k_2 = 0.03$，L是像素值的动态范围）：

$$SSIM(x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \tag{8}$$

CIEDE2000 是目前最新的色差方法，对比上代公式，准确度有较大提升，理论来说是目前和人的视觉体验最直接相符的公式，公式表达如下（在颜色空间内计算各参数）：

$$\Delta E_{2000}=\sqrt{(\frac{\Delta L'}{K_L S_L})^2+(\frac{\Delta C'_{ab}}{K_C S_C})^2+(\frac{\Delta H'_{ab}}{K_H S_H})^2+R_T(\frac{\Delta C'_{ab}}{K_C S_C})(\frac{\Delta H'_{ab}}{K_H S_H})} \tag{9}$$

经过实验比较，SinGAN-Dehaze 网络在 MindSpore 上的综合性能与在 PyTorch 上的训练结果具有可比性且相差不大。

表 1 与最先进的方法在 SOTS 数据集上进行定量比较

	AOD-net	LDP	EPDN	Cycle Dehaze	SinGAN-Dehaze（PyTorch）	SinGAN-Dehaze（MindSpore）
Paired	✓	✓	✓	×	×	×
PSNR	20.49	20.63	22.00	19.35	22.47	22.23
SSIM	0.8579	0.7891	0.8786	0.7544	0.8756	0.8391
CIEDE	7.18	7.46	7.11	9.92	6.50	7.22

表 2 在 O-HAZE 数据集上的定量比较

	AOD-net	LDP	EPDN	SinGAN-Dehaze（PyTorch）	SinGAN-Dehaze（MindSpore）
Paired	✓	✓	✓	×	×
PSNR	15.30	14.67	16.26	16.25	16.10
SSIM	0.5430	0.4821	0.5519	0.7005	0.5658
CIEDE	18.09	19.99	14.35	14.94	15.09

表 3 在 DHazy 数据集上的定量比较

	CycleGAN	GLCGAN	PSD	SinGAN-Dehaze（PyTorch）	SinGAN-Dehaze（MindSpore）
PSNR	13.39	15.98	10.85	19.84	17.03
SSIM	0.5223	0.8208	0.6497	0.7970	0.7232
CIEDE	17.61	10.39	21.65	7.69	10.63

将 MindSpore 上实现的 SinGAN-Dehaze 方法与在 PyTorch 框架下建立的最先进的方法进行比较，如表 1，表 2 与表 3 所示，SinGAN-Dehaze（PyTorch）在 SOTS 数据集上取得了最高的 PSNR，在 O-HAZE 数据集上取得最高的 SSIM 和最好的 CIEDE 效果，在 DHazy 数据集上有最高的 PSNR 和 CIEDE，而其他指数测试结果也得到较为理想的表现，可见此方法综合性能最好，而 SinGAN-Dehaze（MindSpore）的测试结果有一定的性能下降，在室内数据集上的表现不如室外数据集，一方面是由 MindSpore 框架算子的不完善所导致，比如优化器算子的缺失，另一方面是室内数据集的事物细节更丰富，稍许差异就会导致较大的结果偏差，这些在实验可容忍的误差范围之内。

第六章 总结与展望

6.1 总结

本文根据一种新的用于非配对图像去雾的 SinGAN-Dehaze 网络，在 MindSpore 框架上进行了实现。这种去雾方法打破了目前解决非配对图像去雾问题的主流 CycleGAN-like 框架的限制，将循环一致性约束解耦为内容一致性约束和样式一致性约束。具体来说，通过局部扰动对比学习和全局语境感知学习来保证内容一致性约束，同时将高频特征作为风格信息，使去模糊结果与清晰图像之间的风格距离最小化。因此，只使用一个 GAN 来解决未配对图像去雾问题。在合成数据和实际图像上的大量试验显示，这种非配对去雾技术可以达到非常有效的作用，甚至超过了一些监督去雾方法。而 MindSpore 作为华为科技公司最新发布的深度学习框架，具有易开发、高效执行、全栈覆盖等特点，引领国内 AI 发展。结合 SinGAN-Dehaze 图像去雾和 MindSpore 框架，有利于推动实现代码国产化，产品国有化，打破国内人工智能开发瓶颈，最终的实验结果表明，SinGAN-Dehaze 方法在 MindSpore 框架上的性能略低于 PyTorch 框架，但也能取得很理想的效果。

6.2 工作展望

本项目是 SinGAN-Dehaze 去雾方法在 MindSpore 框架上的初次实现，在网络定义和训练结果等方面还存在一些不足，在以下方面可做出改进：

1、在 SinGAN-Dehaze 方法中，要求清晰无雾的图像与模糊的图像具有相似的风格，比如都是室内图像。当雾霾数据集和清晰数据集的域间差距完全不同时，无法达到很好的效果。因此，要求训练所使用的数据集具有类似的风格，由于 ModelArts 文件上传存在的限制以及训练的不稳定，图像转换效果不是很好；

2、MindSpore 框架的 API 开发仍然任重道远，进行网络迁移时，由于算子缺失导致网络精度下降是无法避免的，在之后发布 MindSpore 新版本增添新算子后，网络精度还可以得到更好的优化。

参考文献

[1] 于璠.新一代深度学习框架研究[J].大数据，2020,6(04):69-80.

[2] Yanyun Qu, Weniin Yang. Farewell to CycleGAN: Single GAN with Decoupled Constraint for Unpaired Image Dehazing. CVPR 2022 Submission.

[3] 华为发布 AI 处理器昇腾 910[J].办公自动化，2019,24(19):25.

[4] MindSpore 技术白皮书[EB/OL]. https://www.mindspore.cn/docs/programming_guide/zh-CN/r1.6/design/technical_white_paper.html. [2022-05-08]

[5] Boyi Li, Xiulian Peng, Zhangyang Wang, Jizheng Xu, and Dan Feng. Aod-net: All-in-one dehazing network. In ICCV, 2017. 1, 2, 7, 8

[6] Yang Liu, Jinshan Pan, Jimmy Ren, and Zhixun Su. Learning deep priors for image dehazing. In ICCV, 2019. 1, 2, 7, 8

[7] Xu Qin, Zhilin Wang, Yuanchao Bai, Xiaodong Xie, and Huizhu Jia. Ffa-net: Feature fusion attention network for single image dehazing. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 11908–11915,2020. 1, 2, 7

[8] Yanyun Qu, Yizi Chen, Jingying Huang, and Yuan Xie. En-hanced pix2pix dehazing network. In CVPR, 2019. 1, 2, 7,8

[9] Haiyan Wu, Yanyun Qu, Shaohui Lin, Jian Zhou, Ruizhi Qiao, Zhizhong Zhang, Y uan Xie, and Lizhuang Ma. Contrastive learning for compact single image dehazing. CVPR,2021. 1

[10] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei AEfros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In ICCV, 2017. 1, 6, 7

[11] MindSpore 编程指南[EB/OL]. https://www.mindspore.cn/docs/programming_guide/zh-CN/r1.6/index.html .[2022-05-08]

[12] Robby T Tan. Visibility in bad weather from a single image.In CVPR, 2008. 2

[13] Deniz Engin, Anil Genc ¸, and Hazim Kemal Ekenel. Cycle-dehaze: Enhanced cyclegan for single image dehazing. In CVPRW, 2018. 1, 2, 3, 6, 7, 8

[14] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimension-ality reduction by learning an invariant mapping. In CVPR,2006. 3

[15] Kyungjune Baek, Y unjey Choi, Y oungjung Uh, JaejunY oo, and Hyunjung Shim. Rethinking the truly un-supervised image-to-image translation. arXiv preprintarXiv:2006.06500, 2020. 3

[16] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In ECCV, 2020. 3

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.Deep residual learning for image recognition. In CVPR,2016. 3

[18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Repre-sentation learning with contrastive predictive coding. arXivpreprint arXiv:1807.03748, 2018. 3, 4

[19] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, ZhenWang, and Stephen Paul Smolley. Least squares generative adversarial networks. In ICCV, 2017. 5

[20] Cosmin Ancuti, Codruta O. Ancuti, and Christophe DeVleeschouwer. D-HAZY: A dataset to evaluate quantita-tively dehazing algorithms. In ICIP, 2016. 5, 6

[21] Boyi Li, Wenqi Ren, Dengpan Fu, Dacheng Tao, Dan Feng,Wenjun Zeng, and Zhangyang Wang. Benchmarking single-image dehazing and beyond. TIP, 2018. 5

[22]Codruta O Ancuti, Cosmin Ancuti, Radu Timofte, and Christophe De Vleeschouwer. O-haze: a dehazing bench-mark with real hazy and haze-free outdoor images. In CVPRW, 2018. 5, 7

[23]PyTorch 与 MindSpore[EB/OL]. https://www.mindspore.cn/docs/zh-CN/r1.7/note/api_mapping/pytorch_api_mapping.html. [2022-05-08]

[24]ModelArts 使用简介[EB/OL]. https://support.huaweicloud.com/qs-modelarts/modelarts_06_0006.html. (2022-04-15) [2022-05-08]