# 1 绪论

## 1.1研究背景及意义

近年来深度学习研究取得了巨大进展，使得深度学习在图像识别、自然语言处理、语音识别等领域取得巨大的成就，尤其在图像识别、语音识别等模式领域，深度学习的分类准确率已经超越了人类。尽管深度学习在解决很多复杂问题时表现优异，但Szegedy等人发现深度神经网络（Deep Neural Networks，DNN）极易受对抗样本的攻击，这些对抗样本只是在原有的图像样本添加轻微的扰动，人类视觉系统几乎无法察觉这些扰动，就可导致基于深度学习的图像分类系统输出错误的分类结果，对抗样本攻击深度学习模型过程如图1所示。Kdnuggets等人还指出对抗样本不仅仅存在于计算机视觉领域，也普遍存在于语音识别、恶意软件检测、文本处理等深度学习领域。对抗样本的出现引发人们对深度学习的工作原理的质疑，也使得越来越多的研究人员开始对对抗样本的生成和防御进行研究。随着研究的不断深入，研究者们发现同一对抗样本可攻击不同网络结构的深度学习模型，这使得攻击者无需知道深度学习模型的具体网络结构就可对该模型进行攻击。此外，Sharif等人还发现将对抗样本打印、拍照处理后，对抗样本仍然可以欺骗深度学习图像识别系统，他们的发现也证明了物理世界也存在对抗样本。

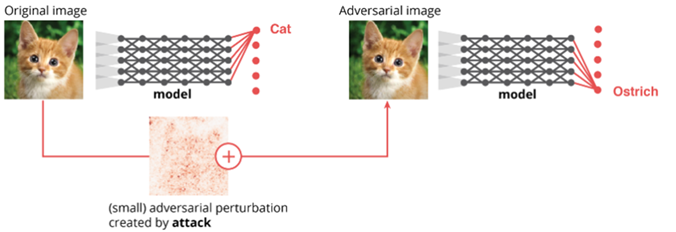


图1 对抗样本攻击举例

目前很多深度学习解决方案已经进入人们的日常生活，如人脸识别、自动驾驶、视频检测等，对抗样本的存在给这些解决方案的使用带来了巨大的风险。比如，在人脸识别系统中，恐怖分子可以利用对抗样本冒充他人身份，然后进行恐怖袭击，将造成不可估量的损失。又或者，在自动驾驶过程中，使用对抗样本覆盖真实的路标，车辆自动驾驶系统将无法对该路标做出正确的决策，从而引发严重的交通事故。因此设计安全性要求严苛的深度学习解决方案时，必须考虑如何防御对抗样本的攻击。

对抗样本的存在极大的限制了深度学习解决方案的使用，因此研究如何有效的防御对抗样本攻击具有非凡的现实意义。目前针对对抗样本的攻击，研究人员提出许多不同类型的防御方案，并取得了不错的防御效果，但随着对抗样本的不断升级，防御方案很容易被进化后的对抗样本攻破。另外，现有的防御方案大多是被动式防御，基本上只能防御一种对抗样本。鉴于目前对抗样本防御领域存在的问题，本文提出一种新的防御策略，利用降噪变分自动编码器(Denoising Variational Auto-Encoder，DVAE)结合生成式对抗网络(Generative Adversarial Networks，GAN)的对抗训练，设计并实现了一种专门针对对抗样去噪模型DVAE-GAN，通过去噪模型去除对抗样本中的扰动，以达到防御对抗样本的目的。

## 1.2国内外研究现状

自Szegedy等人提出对抗样本概率以来，研究人员们为了解释对抗样本存在原因，提出了各种各样的假设和推理，如模型正则化不足、输入高纬度等。根据这些假设和推理，研究者人员设计并实现多种不同的生成对抗样本算法，并尝试攻击不同的深度学习场景。在大量生成对抗样本算法提出的同时，深度学习应用领域也迎来了巨大的挑战，这也使得越来越多研究人员致力于研究如何有效的防御对抗样本。本节接下了将分别介绍目前对抗样本的攻击和防御国内外研究现状。

### 1.2.1生成对抗样本算法研究现状

目前对抗攻击分类方式有两种：一种是根据攻击者是否拥有深度学习模型信息，分为黑盒攻击和白盒攻击；另一种是根据攻击是否具有目标性，分为目标攻击和无目标攻击。黑盒攻击是指攻击者没有深度学习模型的信息，既不知道深度学习模型的网络结构也不知道深度学习模型的训练集；白盒攻击是指攻击者拥有深度学习模型的全部信息，既知道深度学习模型的网络结构也知道深度模型的训练数据集。目标攻击是指攻击者要求对抗样本能被深度学习模型识别成指定分类，无目标攻击是指攻击者只要求对抗样本能够使深度学习模型识别错误分类，不对具体分类目标作要求。本节选择从黑盒攻击和白盒攻击角度介绍目前国内外生成对抗样本的研究现状。

(1)白盒攻击

Szegedy等人首次提出对样本概念，证明了可以通过对图像添加小量的人类察觉不到的扰动误导神经网络做出误分类。他们尝试求解让深度学习模型做出误分类的最小扰动方程。由于求解该最小扰动程算法复杂度过高，他们将问题转化为寻找最小损失函数添加项方程，成功将问题转化成凸优化过程，并提出L-BFGS优化算法求解该问题，同时他们还提出对抗样本在不同模型中具有迁移性。

Goodfellow 等人指出对抗样本的存在可能跟深度神经网络的高度线性特性有关，基于上述假设他们提出了一种快速生成对抗样本算法——快速梯度符号法（FastGradient Sign Method，FGSM），同时他们还提出使用对抗样本进行对抗训练，可以有效的提高深度神经网络抵御对抗样本的能力。

Moosavi-Dezfooli等人为计算能使分类器模型分类产生错误的最小扰动提出了Deepfool算法，该算法生成对抗样本扰动会明显小于FGSM算法生成的对抗样本扰动，缺点是该算法生成对抗样本需要多步迭代，生成一个对抗样本需要花很多时间。

Carlini和 Wagner提出一种根据修改目标函数的生成对抗样本算法——C&W算法，经过实验证明该算法生成扰动可以攻破大多数基于目标函数的防御方法，该算法主要缺点是生成对抗样本需要大量迭代，生成一个对抗样本要花很多时间。

Kurakin 等人在原有的FGSM的基础上进行改进，将原有的单步生成对抗样本改进为迭代生成的方式，提出了基础迭代法(Basic Iterative Method，BIM)、最小可能类法 (Leastlikely Class Method，LCM)、迭代最小可能类法 (Iterative least-likely class method，ILCM) 等对抗样本生成方法，这些算法产生的对抗样本扰动比FGSM算法更小且攻击性更强。

(2)黑盒攻击

Sueta等人提出了一种极端情况下的生成对抗样本算法——OnePixl算法，该算法在只改变一个像素点的情况下，成功攻破三个不同深度的深度学习分类器模型。虽然该生成对抗样本算法仅仅只改变了原始图像的一个像素点，但其生成一个对抗样本时间很长，并且还不一定能成功生成对抗样本。清华大学 TSAIL 团队 Dong 等人提出了动量迭代法(Momentum Iterative Method，MIM)，该方法通过将动量项整合到BIM迭代过程中，极大提高了BIM的黑盒攻击性。Bose等人提出一种基于GAN攻击人脸识别系统的方法，该算法将人脸识别系统作为GAN的判别器，然后通过对抗训练生成对抗样本，通过实验证明该算法能在黑盒的条件下攻破人脸识别系统。

### 1.2.2 对抗样本防御研究现状

目前针对对抗样本防御方法主要分为完全防御和检测防御两类。完全防御目标是使深度学习模型能够正确的识别对抗样本的原始标签；检测防御的目标是能够正确的识别输入样本是否为对抗样本，如果输入的是对抗样本，则发出警报不对该样本进行分类。本接下来将根据这两个分类详细介绍目前国内外对抗样本防御现状。

(1)完全防御

基于对抗训练的防御方法：该类防御方法通过往深度学习模型训练过程中添加对抗样本的方式，以训练健壮性更好的模型，最终达到防御对抗攻击的目的。根据添加的对抗样本类型，对抗训练的防御方案可分为：基于FGSM对抗训练的防御策略、基于集成式对抗训练的防御策略及基于BIM的对抗训练的防御。基于FGSM对抗训练的防御通过在训练

## 1.3研究内容及章节安排

# 对抗样本攻防理论基础

### 2.1对抗样本的定义

2014年Szegedy等人首次提出对抗样本概念，他们发现深度神经网络很容易受到对抗样本的攻击，这些对抗样本只是在原始样本中添加轻微扰动，人眼甚至无法察觉这些扰动，就能使得分类器输出错误的分类结果。为了更好的描述对抗样本，他们对对抗样本作如下定义：

假设将图像映射为标签的分类器模型为：



分类器的损失函数为：



那么对于一个样本并且，其对抗样本生成过程可以表示为一个有界优化问题：



由于优化公式（2-3）算法复杂度太高不易计算，并且条件可以转化为，因此他们将该问题进行转化，转化后的公式为：



对于公式（2-4）他们提出用L-BFGS优化算法去求解该问题。在求解对抗样本的同时，他们还发现对抗样本对于低纬度的隐藏层输出影响较小，对于高纬度隐藏层输出影响较大。最后他们对于对抗样本存在性问题，提出了一个可能的解释（如图2-1所示）：现实世界出现对抗样本的概率很低，因此训练集和测试集中几乎不存在对抗样本或者说训练集无法覆盖所有的样本，又因为深度神经网络的高度非线性，导致过拟合只学习到非对抗样本的特征，并没有学习到泛化性特征。

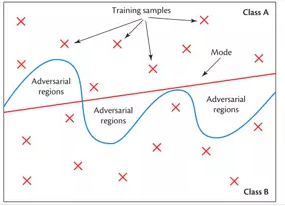


图2.1 深度神经网络模型训练与使用

如图2.1所示，蓝线为样本的真实决策边界，红线为模型训练得到的决策边界。可以从图中发现，模型训练后的最终决策边界和真实样本决策边界有较大的差别，这表明对抗样本是广泛存在的。

### 2.2生成对抗样本方法

在对抗样本概念被提出之后，越来多研究人员致力于研究生成对抗样本的方法。目前对抗样本攻击主要分为两类：一是白盒攻击，攻击者能够获取深度神经网络结构及参数，另一类是黑盒攻击，攻击者完全不知道深度神经网络网络结构及参数，只能通过模型输入获取输出。目前黑盒攻击主要通过对抗样本的迁移性来实现（用一个已知网络模型生产对抗样本，然后攻击未知模型），或者直接根据模型的输入输出建立生成对抗样本优化模型，通过求解该优化模型得到对抗样本。虽然黑盒攻击和白盒攻击从攻击的目的来说有很大的不同，但其生成对抗样本的本质并没有区别。与黑盒攻击相比，白盒攻击攻击效果要更好，并且目前的黑盒攻击大多是白盒攻击的迁移或者融合多种白盒攻击。此外，根据对抗样本攻击是否具有目标性可以将对抗攻击分为有目标攻击和无目标攻击，有目标攻击是指对抗样本能够使分类器以高置信度输出指定分类结果，无目标攻击只需使分类器输出错误结果即可。

目前生成对抗样本的方法数量众多，每一种生成对抗样本方法都有其独特之处，生成的对抗样本攻击强度也不尽相同，但它们生成对抗样本的本质没有什么区别，因此本文选择介绍目前较有代表性三种生成对抗样本方法：FGSM、BIM和PGD。

### 2.2.1快速梯度法-FGSM

2015年Goodfellow等人对生成对样本原因进行分析，他们发现扰动对深度神经网络影响会像雪球一样随着网络深度的增加越滚越大，他们称这种现象为扰动累加效应。由于神经网络的高度线性特性，导致这种效应相比非线性模型更加严重。基于深度神经网络高度线性及扰动累加效应，他们提出了一种基于梯度的快速生成对抗样本方法FGSM，FGSM中对抗样本扰动计算公式如下：



其中表示模型输入，表示结果标签， 表示损失函数，表示符号函数，表示控制扰动大小的的自定义参数，表示生成的对抗样本扰动。FGSM算法的主要思想:如果扰动的方向和梯度方向一致，损失函数就会增大，进而使得分类器分类结果产生最大的变化。扰动越大损失函数就会越大，分类器结果产生的变化就会越大。

该方法可以通过一步快速生成对抗样本，并且该方法未直接使用梯度值而选择梯度方向有效的控制了对抗样本扰动的大小。虽然该方法生成速度快，使用的计算资源少，但使用该方法有诸多的限制。首先该方法适合白盒攻击，生成的对抗样本迁移性较差。其次该方法对于线性分类器模型攻击效果较好，对于非线性分类器模型和不可微分类器模型攻击效果较差。最后使用FGSM生成的对抗样本扰动较大（相当于每个像素点都增加了），该扰动自身抗干扰能力不强，容易受到其他噪声影响。

### 2.2.2基础迭代法-BIM

针对于FGSM存在的两个问题：损失函数与输入图像之间近似线性和扰动相较于原始图像改动较大。Goodfellow等人提出进一步的假设，如果损失函数与输入图像之间是非线性的，是否会存在一个存在于之间的某个扰动，使得变得更大。基于上述新的假设，Goodfellow等人提出了一种以迭代的方式生成对抗样本的方法BIM，生成对抗样本公式如下：



其中截断函数为：



截断函数的目的是保证的每一个像素都在的临域内且每个像素都有意义。BIM每次迭代的含义是：每次迭代在上一步生成的对抗样本基础上每个像素增加（也可能减少），然后对新生成的对抗样本进行裁剪，保证新的对抗样本每个像素都在的各像素临域内。BIM生成的对抗样本攻击效果一般要优于FGSM生成的对抗样本，最差的情况生成的对抗样本攻击效果与FGSM相同。

BIM生成的对抗样本攻击效果虽然优于FGSM，但由于其生成对抗样本迭代过程固定且每次迭代都是对上一次生成的对抗样本进行裁剪，这导致其能够迭代的次数较少，同时生成的对抗样本也容易受到其他噪声的影响。

### 2.2.3投影梯度法-PGD

2017年Aleksander等人发现采用PGD方式求解对抗样本中的优化问题效果要好于Goodfellow他们提出的使用梯度下降方式。PGD算法与梯度下降算法比较类似，都是用于求解非线性规划问题，但PGD算法在求解带有约束的非线性规划问题时具有更好的效果。PGD在求解对抗样本优化问题时，先选取一个可行解，然后将梯度下降的方向投影到约束集上，然后连接可行解点和梯度下降点生成可行的下降方向，最后重复上述过程直到对抗样本满足设定要求的精度。PGD生成对抗样本算法如下：



其中为一个学习步长，表示投影算子，该算子能将当前可行解投影到约束集中向量长度最小的向量上。PGD相较于BIM有更多的迭代轮次，并且往对抗样本中添加了随机化点，使得PGD的攻击效果远好于BIM。PGD生成的对抗样本也是目前一阶损失函数攻击攻击效果最好的算法。

### 2.3对抗样本防御策略

针对以上的对抗样本攻击特点，目前的防御手段主要有三个方向：对输入图像进行预处理，改进神经网络模型，检测输入是否为对抗样本。

对图像进行预处理：因为对抗样本是在原有的样本上添加而外的扰动噪声，一般情况下噪声越大攻击效果越好，同时因为目前深度神经网络高度线性特点扰动对神经网络影响主要是靠累加效应，因此只要减少对抗样本中的扰动就可以减轻对抗样本攻击效果。目前对图像的预处理主要是通过图像降噪，图像压缩，图像重建。图像预处理一定程度能减少样本中的对抗扰动，但是代价是会降低整个模型的分类精度。

改进神经网络模型： Szegedy等人提出对抗本概念时也指出对抗样本会导致神经网络高层输出远高于普通样本。根据以上特点Goodfellow等人提出使用Lipschitz常数来控制神经网络的层与层输出比例，进而防止高层神经元输出爆炸。Goodfellow等人在提出FGSM时也提出使用对抗训练能够很好的提高深度神经网络模型的健壮性。此外还可以通过提高深度神经网络模型非线性度来提高深度神经网络模型的健壮性。

检测输入是否为对抗样本：虽然对抗样本与原始样本相比变化十分微小，但神经网络对这两种样本输出特性完全不同，也有学者发现对抗样本无法用重构网络进行重构。因此我们可以通过另一个模型观察神经网络分类器输出特性来区分样本是否为对抗样本，或者使用胶囊重构来辨别输入是否为对抗样本。

本文将选取三种具体的防御策略，进一步分析各方向防御策略的优缺点。

### 2.3.1图像压缩重建

图像压缩重建防御策略属于图像预处理防御策略的一种，其主要思路：在图像进入分类器之前将图像“净化”，得到不包含对抗扰动的干净图像后再送入分类器中；图像“净化”主要利用了图像压缩过程去除冗余信息的思想，图像压缩后只保留了主要信息，对抗样本扰动属于冗余信息，图像压缩重建后这部分冗余信息将被去除；同时压缩重建模型和分类器分开训练，并不会影响到已经训练好的模型。图像压缩重建模型图2.1所示：

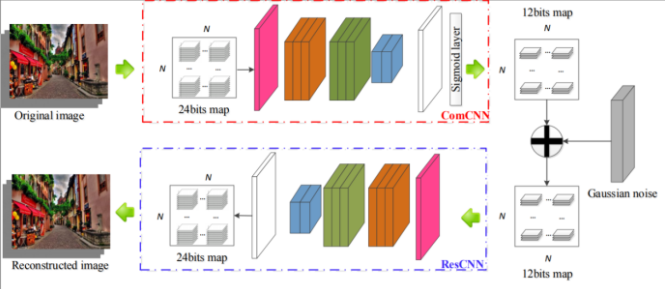


图2.1 图像压缩重建网络结构

图像压缩重建防御策略主要的优点是：不需要专门训练对抗样本，因为未改变分类器模型所以不需要重新训练分类器模型，“净化”后的图像在不同的分类器模型中都有较高的分类准确率。主要缺点是图像在经过压缩重建后图像质量会下降，这会导致分类器对无噪声图像分类准确率下降。

### 2.3.2对抗训练

对抗训练是指在分类器损失函数中添加一个正则项，该正则项是基于对抗样本引发损失的建模。分类器模型使用这种新的损失函数进行训练可以提高模型对对抗样本的泛化能力。实际操作是在原始模型训练集中添加对抗样本，进而提升模型对于对抗样本扰动的健壮性。使用FGSM作为对抗训练模型损失函数修改如下：



其中为模型原始损失函数，表示对抗样本引发的损失。对抗训练的好处是模型可以防御已知的对抗样本生成方法生成的对抗样本（只需向模型训练集中添加足够的该对抗样本生成方法生成的对抗样本），并且防御效果很好。缺点也很明显，对于新种类的对抗样本没有防御能力，并且每出现一种新的生成对抗样本方法，要想防御必须将该生成对抗样本生成的对抗样本加入到模型的训练集中然后重新训练，反复重新训练模型极大的提高了实施防御的成本。

### 2.3.3通过胶囊网络检测对抗样本

Frosst等人提出只需简单的改进胶囊网络训练方式就能使胶囊网络检测出对抗样本图像。他们除了正常训练图像分类任务之外，还训练胶囊网络根据顶层的姿态参数来重构原始图像。因为对抗样本与分类器分类结果类有较大的不同，所以会导致通过该类的顶层胶囊重构图像时会有较大的重构误差（图2.2给出了从真实数据和对抗样本生成的重构图像）。他们通过实验证明在输入图像和重构图像之间的距离上设置阈值可以有效的检测出对抗样本。

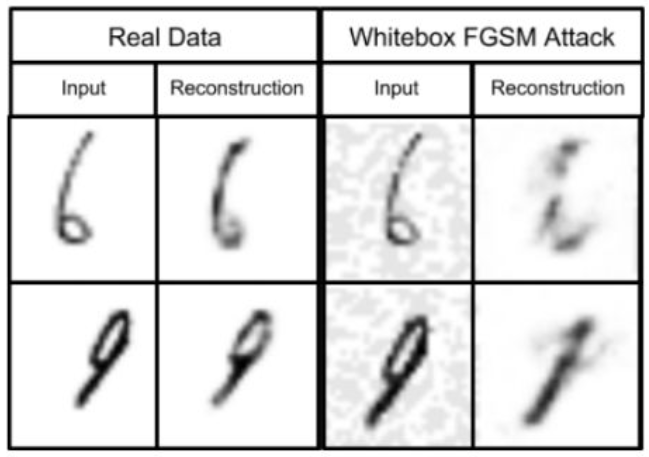


图2.2 真实数据与对抗样本生成的重构图像

该检测对抗样本方法主要适用于较小的数据集（如MNINST），在更复杂的数据中两张图像看起来相似，但两张图像的距离很大，这导致该方法并不能检测出图像是否为对抗样本。

### 2.4对抗样本攻防总结

目前对抗样本主要特点有：对抗样本存在并不唯一，对抗样本攻击可以是有目标的也可以是无目标的；对抗样本具有迁移性，但对抗样本在用于未知模型时其攻击效果会降低；虽然生成对抗样本方法众多，但其本质是一样的，且目前的生成对抗样本算法大多是对FGSM算法进行改进；使用同一种生成对抗样本算法生成的对抗样本，对抗样本扰动越大其攻击效果越强同时抗干扰能力也越强；深度神经网络对于对抗样本与普通样本输出特性完全不同，这说明对抗样本与普通样本可能有着不一样的分布。

目前的防御策略主要存在三个问题：一是大部分防御策略是削弱对抗样本对神经网络模型的影响，二是许多防御策略需要重新训练分类器模型这导致防御成本极高，三是大部分防御策略会导致无噪声图像分类准确率下降。因此在制定对抗样本防御策略时不仅要考虑防御效果还要考虑防御成本及是否会影响普通样本分类准确率，由于现实生活中对抗样本存在的可能性较小，因此制定防御策略时要着重考虑防御策略是否会影响普通样本的分类准确率。此外，如果防御策略能够在不同模型之间迁移，该防御策略的防御成本将大量减少。

# 基于VAE-GAN的对抗样本防御策略

目前已有的对抗样本扰动大多十分微小（人眼难以察觉），然而深度神经网络模型对这些扰动十分敏感。目前许多防御手段目的是消除对抗样本中的扰动，使其恢复成原本的图像，这些方法对于部分扰动较大的对抗样本十分有效。但清华大学 TSAIL 团队 Dong 等人发现即使除去对抗样本的大部分噪声，部分攻击性强的对抗样本依然能够影响深度神经网络模型，使其输出错误的分类结果。Issaranon等人发现虽然对抗样本与原始样本十分相近，但深度神经网络模型对于两种样本输出特性完全不同，这表示对抗样本与原始样本的分布可能是不一样的。基于对抗样本以上两点特性，本文提出一种新的防御策略——基于VAE-GAN的对抗样本防御策略，该策略能在几乎不损害图像质量的情况，去除对抗样本的对抗扰动，将对抗样本还原成原始样本。

## 3.1 模型的基本原理

### 3.2.1 VAE的原理

变分自动编码器（Variational Auto-Encoder,VAE）是kingma等人在2012年提出的一种基于变分贝叶斯推断的生成式神经网络模型，主要用于数据压缩，图像降噪及图像重建等方面。VAE是完全的概率推断模型，能够近似的估算出输入样本的边缘概率，并通过最大化该边缘概率来优化模型的参数。如图3.2所示，VAE模型主要由两部分组成：一是编码器对输入数据进行变分推断，生成隐变量的变分概率分布；另一个是解码器将隐变量的变分概率分布还原成输入数据的近似概率分布。

如图3.2所示，表示真实样本的数据集,每个数据样本都是随机产生相互独立连续或者离散的分布变量，表示解码器解码生成的样本，表示隐变量，表示编码器中网络的参数，表示解码器网络中的参数。其中是一个高维空间可观测的随机向量，是一个相对低维的不可观测随机向量，VAE生成模型可以分为两个过程：Encoder推断得到隐变量的近似分布过程，Decoder将隐藏变量还原成近似于输入样本概率分布的过程。

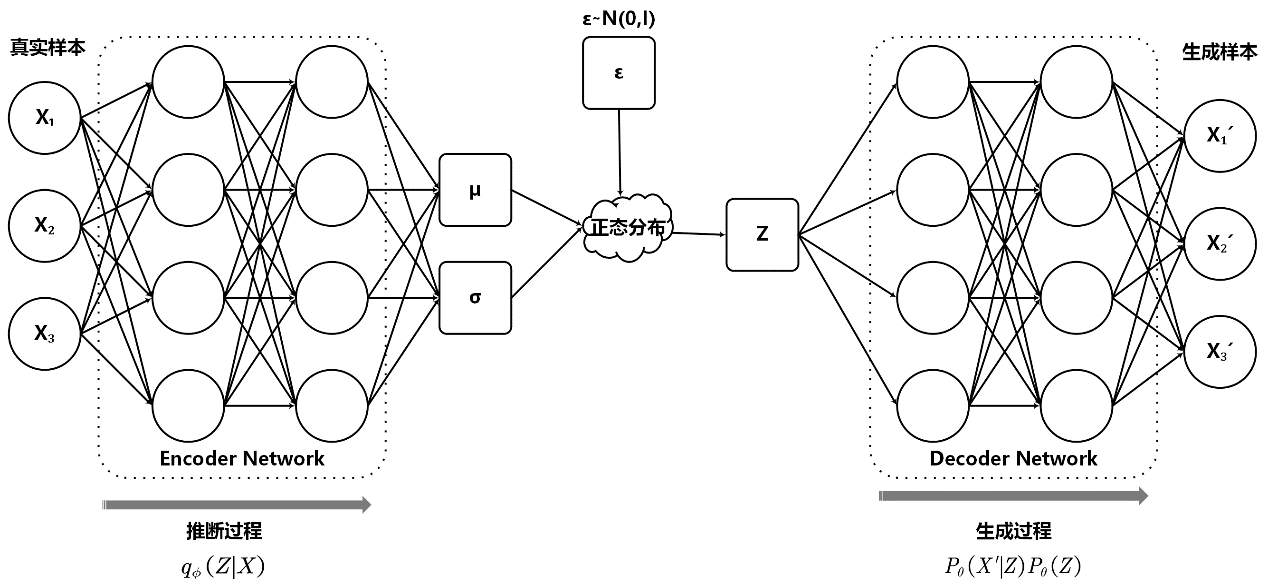


图3.1 VAE的结构示意图

在图3.2中，表示真实样本的数据集,每个数据样本都是随机产生相互独立连续或者离散的分布变量，表示解码器解码生成的样本，表示隐变量，表示编码器中网络的参数，表示解码器网络中的参数。其中是一个高维空间可观测的随机向量，是一个相对低维的不可观测随机向量，VAE生成模型可以分为两个过程：Encoder推断得到隐变量的近似分布过程，Decoder将隐藏变量还原成近似于输入样本概率分布的过程。

对于编码器，由于隐变量的概率分布无法被观察也难以计算，VAE引入了一个神经网络模型来代替隐藏变量的真实后验分布，并且假设服从普通正态分布，同时为了方便计算假设隐变量先验分布服从标准正态分布。为了使引入的识别模型尽可能与隐藏变量真实概率分布，VAE选择使用KL散度来评价这两个分布的相似度：



其中表示的边缘概率分布。

假设：



通过公式3-1和3-2可以得到：



因为恒成立，因此恒成立，也被称为的变分下界。结合公式3-1和公式3-3，编码器的优化目标可以转换为：



由于数据集的对数边缘似然等于各独立离散变量的对数边缘似然之和，即，因此编码器的最终优化目标可以转化为。

通过编码器我们可以得到隐变量的近似后验概率分布，然后我们可以从隐变量的近似后验分布中采样得到稳定的隐变量。解码器的目的是将隐变量恢复为原始输入样本的近似分布，解码器将隐变量恢复为输入样本近似分布可以表示为：



我们希望越大越好，结合公式3-3，可以将解码器的优化目标转化为：



结合公式3-4，解码器的损失函数也可以转化为。由于编码器和解码器的损失函数都是，所VAE整体优化目标为变分下界。由于公式3-2无法直接求解，我们需要对变分下界函数进行转化，变分下界函数可以最终转化为一个包含编码器和解码器结构的函数：



由于一开我们假设服从普通正态分布，同时也假设服从标准正态分布，因此变分下界函数3-2的前半部分可以进行转化：



对于变分下界的右半部分，该项是关于的后验分布，由于在VAE 没有对对解码器做太强的假设，因此该项不能通过解析的方式解出，kingma等人提出可以通过采样的方式对该项取近似值，因此该项可以转化为：



需要注意的是不是直接从分布中直接采样，而是使用了一种重参数技巧。该技巧将从直接采样，转变为先从采样得到，然后通过计算得到的采样。这是因为直接从采样，采样过程会参与模型参数的优化，而该采样过程不可微，这将导致整个模型无法训练。而从采样再计算的方式，采样过程不会参与模型参数的优化，所以不会影响模型的训练过程。最终采样得到的一般是伯努利分布或者高斯分布。假设采样分布为伯努利分布该项最终可以转化为：



其中表示解码器生成的结果，假设采样分布为高斯分布，则解码器解码的结果为高斯分布的均值，即解码器只关心均值（为未知数的值，且对优化目标没有影响），所以该项最终可以转化为：



综上所述，实际训练中VAE的优化目标可以表示为：



或者：



对应的VAE的损失函数可以表示为：



或者：



在训练过程中损失函数的第一项与解码器网络参数更新无关，解码器的参数更新只受损失函数第二项的影响，该项也被称为解码器的重构误差。

综上所述，VAE模型的最终结构如3.2所示：

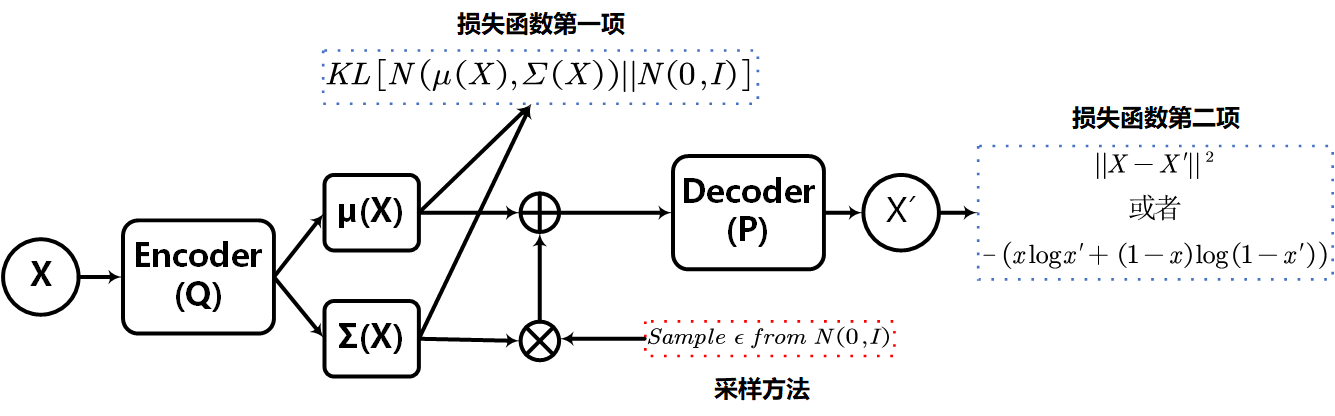


图3.2 VAE模型最终结构

3.1.2 DVAE的基本原理

降噪变分自编码器（Denoising Variational Auto-Encoder,DVAE），是将去噪准则（Denoising Criterion）和VAE结合在一起的一种自编码器。与VAE压缩重构过程不同，DVAE向输入的分布添加而外的随机噪声，然后将这个有噪声的分布作为VAE的输入，最后要求VAE的解码器重构出无噪声的分布。DVAE的结构如图3.3所示。

与VAE的优化目标类似，DVAE的优化目标为DVAE的变分下界函数，由于两个模型的变分下界函数推导过程基本一致，本文不在对DVAE的变分下界函数进行推导直接给出DVAE的优化目标：



或者：



其中和表示由带有噪声的样本编码得到的隐变量的近似后验概率分布的均值和方差。对比VAE和DVAE的优化目标，两者优化目标形式上基本上是一样的，其主要区别是在实际训练过程中VAE的重构分布是编码器输入的分布，而DVAE重构分布不是其编码器输入的分布。

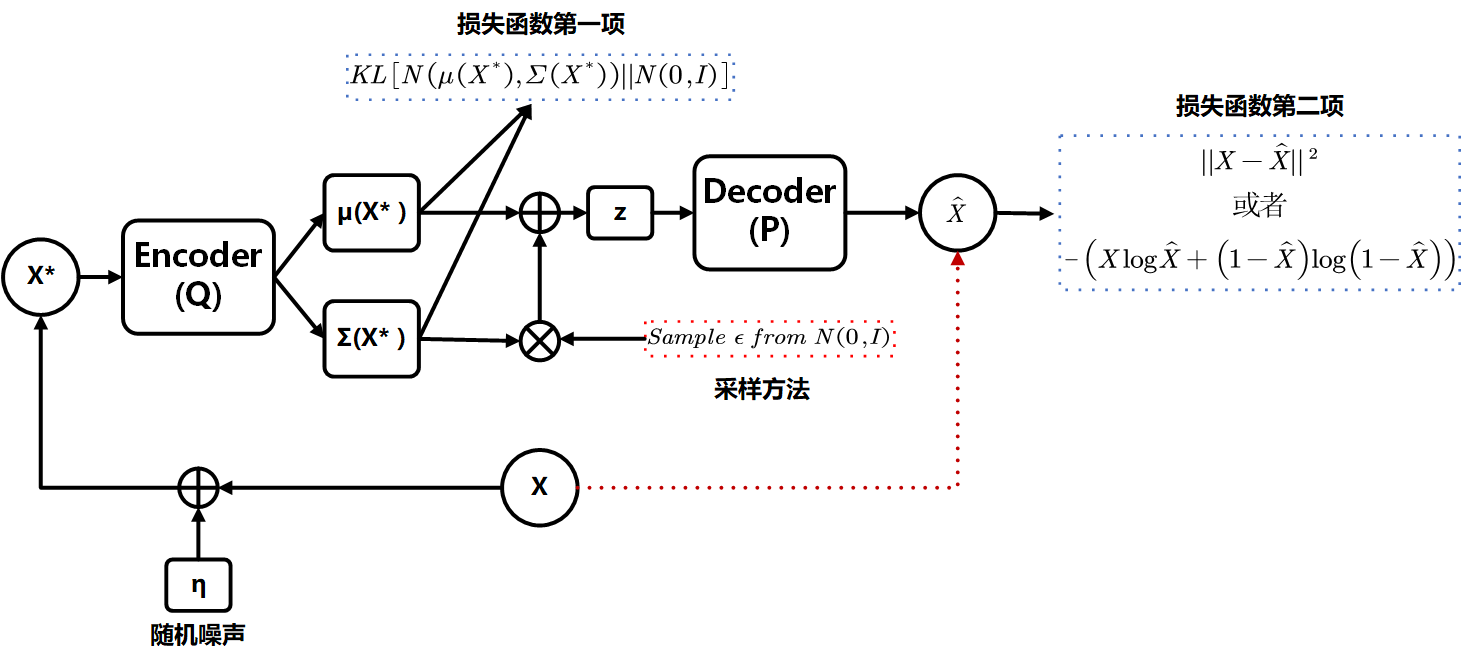


图3.3 DVAE结构示意图

3.1.3 GAN的基本原理

生成式对抗网络（Generative Adversarial Networks，GAN）是一种无监督生成式学习模型，由Goodfellow等人于2014年提出，GAN对抗学习的思想不仅广泛的运用于图像的生成领域，也为其许多其他领域提供了新思路。

如图3.4所示，GAN包含两个网络结构，一个生成网络G（Generator）和一个判别网络D（Discriminatior），其中判别网络D与我们普通分类网络类似。GAN的生成网络G输入为一个随机噪声Z，输出为生成的图像样本G(Z)。判别网络D的输入为G(Z)和真实图像样本，然后判别网络D会对输入辨别分类输出一个概率值用来表示与真实图像样本的相似度，比如判别网络D的输入是G(Z)时输出为0.3，那么表示判别网络D认为G(Z)有30%的概率是真实图像样本。然后生成网络G和判别网络D会根据D的输出不断的优化自己，生成网络G尽可能的提高自己的输出G(Z)与真实图像样本的相似度，即尽可能的欺骗判别网络D，而判别网络D通过学习尽可能的分辨出谁是真实图像样本谁是生成的图像样本。生成网络G和判别网络D相当于在做一个极大极小的博弈过程，该过程也被称为零和博弈。

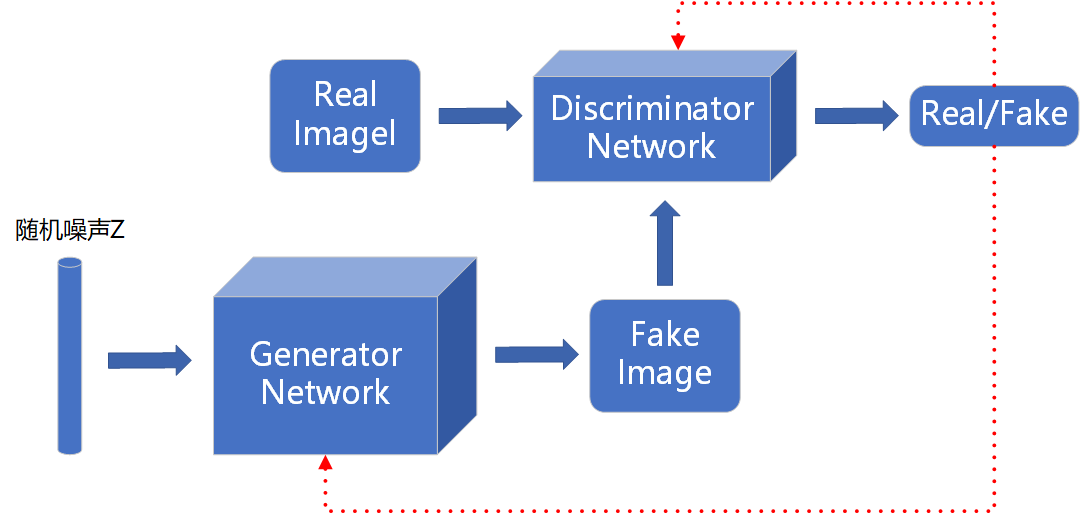


图3.4 GAN网络示意图

与传统深度学习模型训练不同，GAN训练过程中两个网络模型交替训练。在训练判别网络D时，固定生成网络G，相反在训练生成网络G时，固定判别网络。通过反复的对抗训练，最终使生成网络G生成的图像样本与真实图像样本的分布一致，判别网络D无法辨别输入的图像是真实图像样本还是生成图像样本。GAN的优化目标定义如下：



其中D代表判别网络，G代表生成网络，代表生成网络的输入随机噪声，服从分布，表示真实图像样本服从分布，表示期望。对于判别网络D，其目标是最大化优化目标函数，使得接近并且接近0。对于生成网络，其目标是使优化目标函数尽可能的小，也就是要尽可能的接近1。

GAN的生成网络和判别网络通过反复的对抗训练，最终会达到纳什均衡，即生成网络和判别网络都达到最优。如果固定生成网络中的参数，假设真实图像样本分布为，生成图像样本的分布为，则判别网络最优为：



最终解得最佳判别网络为：



在判别网络最优时，生成网络最优为：



引入JS散度公式：



结合公式3-和3-可得：



当真实图像样本分布与生成图像样本分布相同时，生成网络优化目标达到最优为，同时将该条件带入最佳判别网络中，得最优辨别网络输出为。也就是说最终判别网络辨别真实样本的准确率只有50%时，生成网络和判别网络达到纳什均衡。

从上述分析可知，GAN生成的图像不是单纯的记忆数据集中的图像样本，而是通过特征学习生成了新图像。GAN对抗训练有很多优点，但也存在一个比较大的缺点：初始的判别网络训练的不好，将会导致生成网络的损失函数值很小，导致生成网络无法学习。比如在最优判别网络条件下，生成网络的损失函数值将变为和的散度，两个分布相同的时散度值最小为0，相反当两个分布完全不同时散度值将收敛为一个常数。由于初始条件下生成网络生成的图像样本分布与真实图像样本的分布几乎不可能重叠，此时和的散度值为，将该值代入公式3-可得生成网络损失函数值为0，也就是说生成网络的梯度值为0，生成网络将无法学习到如何有效信息。

## 3.2基于DVAE-GAN的对抗样本防御策略

针对对抗样本的噪声特性和扰动累加效应，本文提出一种基于DVAE-GAN的对抗样本防御策略，防御策略主要思路为：在对抗样本图像进入分类器之前通过DVAE将图像进行压缩重建去除对抗样本中的扰动，得到降噪后的图像样本再送入分类器中；为了保证通过DVAE压缩重建后的图像不是对抗样本，使用GAN辅助VAE训练。本文提出防御策略分类器模型和防御模型分开训练，无需重新训练分类器模型。

### 3.2.1模型整体框架

基于DVAE-GAN的对抗样本防御策略整体框架如图3-4所示，防御策略一共包含两个部分：DVAE模块和GAN模块。DVAE模块的主要功能是通过对抗样本进行压缩重建去除对抗样本中的扰动，DVAE模块的输入为对抗样本，输出为去噪后的图像。GAN模块的主要功能是辅助DVAE模块的训练，GAN的生成器和DVAE模块的解码器共用一个神经网络结构，GAN的判别器输入为生成器生成的图像样本和真实的图像样本，GAN的输出为输入样本是真实样本的概率。

本节接下来将详细的介绍整个模型各模块的功能及其模型的整体优化目标。

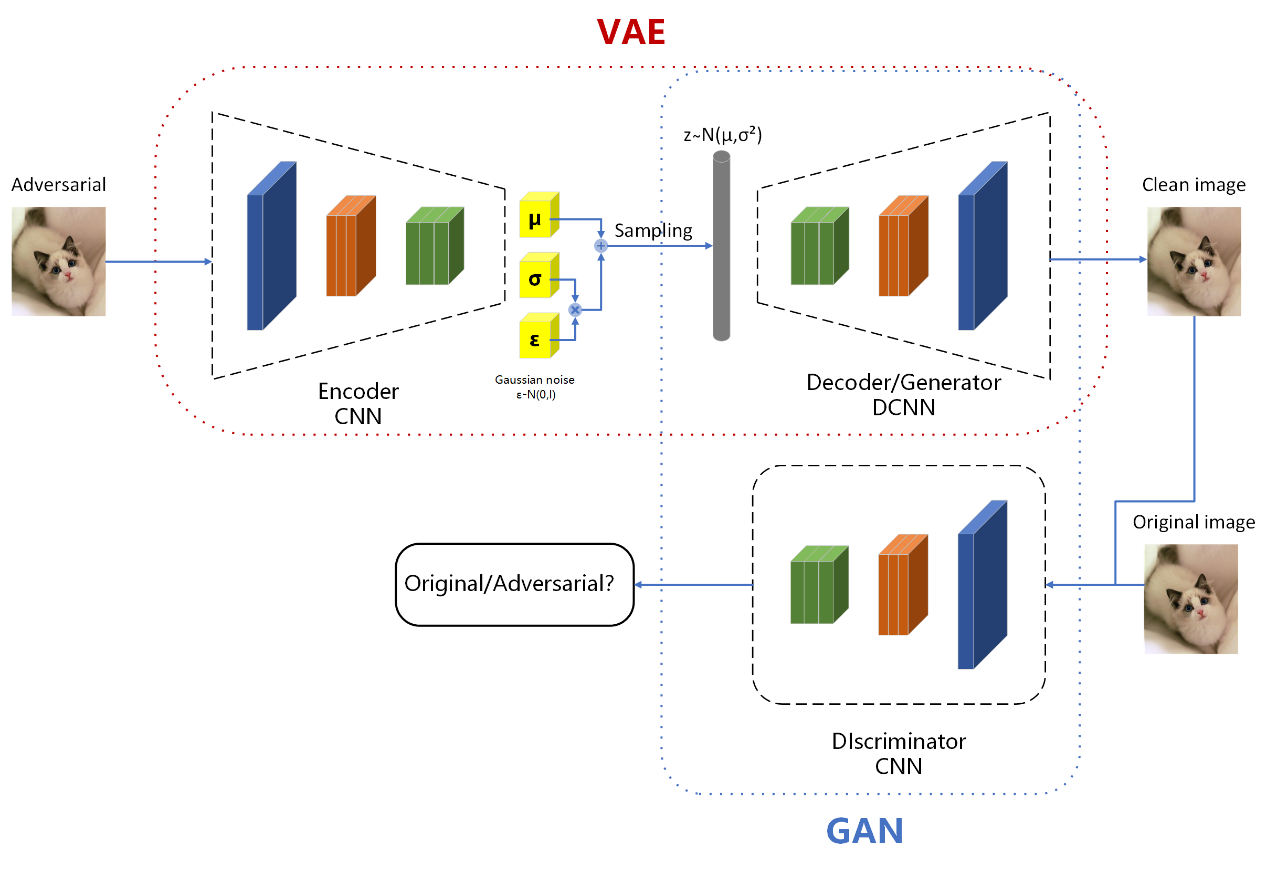


图3.4 基于DVAE-GAN的对抗样本防御策略整体框架

3.2.2 DVAE模块

对抗样本是在原始自然图像中加入特定噪声生成的，这部分噪声相对整个图片来说很小，但并不是完全不可见，如果将图片进行局部放大还是可以发现对抗样本和原始自然样本有很大的不同。由于图像局部结构中的相邻像素有很强的相关性和相似性，图像压缩可以有效的保留图像的显著性信息，同时也能够减少这部分的冗余信息，而对抗样本中的噪声对于原始图像来说是一些额外的冗余信息，因此通过图像压缩重建可以有效对的减少对抗样本中的对抗扰动。为了在图像压缩重建过程尽可能的保证图像的质量，本文选择使用DVAE模型通过压缩重建的方式去除对抗样本中的噪声。

DVAE模块的编码器主要功能是将输入的图像转化为隐变量的近似后验概率分布，然后对近似后验分布进行采样得到稳定的隐变量。编码器的输入为对抗样本或者原始样本，输出是两组维向量：一个组均值向量，另一组是标准差向量，然后我们可以通过这两组维向量构建隐变量的后验近似概率分布。最后我们再从隐变量近似概率分布采样获得隐变量，采样方式为，其中表示从标准正态分布中随机取样的结果。DVAE的解码器主要功能将隐变量恢复为无噪声图像，解码器的输入为隐变量，输出为降噪后的图像。

3.2.3 GAN模块

虽然使用DVAE模块能够去除对抗样本的中的绝大多数的对抗扰动，但去噪后的图像的分类准确率并没有增加多少，仅使用DVAE防御对抗样本结果如表3.1所示，从表3.1中我们可以发现仅使用DVAE对对抗样本进行除噪并不能完全防御对抗样本的攻击，因此DVAE的压缩重建并不能保证去噪后的图像不是对抗样本。通过表3.1还可以发现，普通样本通过压缩重建后分类准确率会下降，那是因为DVAE解码器的损失函数是原始图像与生成图像均方差，如果生成的图像的高频和低频有相同的噪声，那么DVAE解码器所得到的损失函数值是一样的，但相同的噪声在图像的高频部分和低频部分对图像的分布影响是不一样的，通常情况下噪声在图像的高频部分对图像的分布影响较小，在图像低频部分对图像的影响较大，DVAE使用均方差作为解码器的损失函数，导致解码无法学习到图像的分布特性，最后导致DVAE生成的图像相较于原始图像比较模糊。

表3.1 仅使用DVAE防御对抗样本

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | FGSM() | FGSM() | FGSM() |
| Classifier | 96.43% | 24.32% | 16.45% | 11.55% |
| DVAE+Classifier | 92.15% | 52.56% | 47.38% | 42.22% |

针对DVAE存在的上述两个问题，本文使用GAN模型辅助DVAE模型的训练提高DVAE解码器生成图像的质量，同时保证该生成图像不是对抗样本。GAN的生成器和DVAE的解码器共用一个网络，这相当于给DVAE的解码器添加了一个GAN生成器的损失，从本章3.1.3节可知GAN的损失函数是关于生成图像分布和真实图像分布的散度，生成图像分布与真实图像分布越接近该值越小，因此GAN辅助DVAE训练可以让DVAE的解码器在学习到图像的像素特点的同时学习到图像的分布特性。GAN的生成器在给DVAE解码器提供额外的损失函数的同时DVAE的解码器也相对的给GAN的生成器提供了一个均方差损失函数，而这个损失函数恰好可以提高GAN生成器前期生成图像的质量，从而解决了GAN训练难的缺点。GAN的判别器是一个二分类的分类器网络，如果输入的图像是对抗样本，由于对抗样本扰动累加效应判别器的输出会与真实图像的输出不同，这说明在GAN的判别器眼中对抗样本和真实样本属于两种分布，因此通过GAN的生成器与判别器的对抗训练，可以保证DVAE解码器生成的图像不是对抗样本。

GAN的生成器与DVAE的解码器共用一个网络，所以GAN的生成器输入输出与DVAE的输入输出一致。GAN的判别器的输入是生成器生成的去噪后的图像和该图像的原始图像，输出为输入图像是真实图像的概率。

3.2.3模型的优化目标

基于DVAE-GAN的对抗样本防御模型由DVAE和GAN两个模型组成，一共有三个网络结构：DVAE的编码器网络，DVAE解码器和GAN生成器共用网络和GAN的判别器网络。对于DVAE的编码器，其优化目标就是最大化DVAE的变分下界函数：



对应的DVAE编码器的损失函数为：



由于DVAE的解码器和GAN的生成器共用一个网络，因此DVAE解码器和GAN共用的网络的优化目标包含DVAE的变分下界函数和GAN生成器的损失函数：



由于DVAE变分下界函数前半部分与DVAE的解码器无关，同时GAN的优化目标的第一项在GAN的生成器训练时是固定值，因此可以将这两项舍弃。然后DVAE解码器和GAN共用网络的优化目标可以转化为：



对应的DVAE解码器和GAN共用网络可以表示为：



GAN的判别器网络优化目标和普通GAN判别器优化目标一样为：



对应的GAN的判别器网络的损失函数可以表示为：



综上所述，模型整体优化目标为：



对应的模型整体损失函数为：



注意由GAN对抗训练的存在，模型整体的损失函数无法通过一个函数进行描述，公式3-只是一个抽象的损失函数。

3.3 模型的训练流程

在训练过程中，由于GAN对抗训练的存在，我们无法直接根据模型的整体优化目标直接更新全部网络的参数，最后本文选择分步更新模型中各网络结构中的参数，网络参数更新顺序为：先更新DVAE编码器网络，然后更新DVAE解码器和GAN生成器共用网络，最后更新GAN判别器网络。DVAE-GAN训练流程如表3.2所示。

表3.2 DVAE-GAN训练流程

|  |
| --- |
|  |
|  |

训练具体流程为：

1.初始化DVAE编码器网络，DVAE解码器和GAN生成器共用网络和GAN判别器网络中的参数；

2.如果剩余训练周期大于0并且模型各模块的损失函数不是无穷大执行步骤3，否者训练结束；

3.从训练集中选取一批图像样本；

4.随机向图像样本中添加对抗扰动，得到；

5.通过DVAE的编码器网络计算隐变量后验概率分布的均值和方差和；

6.从隐变量的后验概率分布采样得到隐变量；

7. 通过DVAE解码器和GAN生成器共用网络将隐变量还原成无噪声图像；

8.计算DVAE编码器网络的损失函数；

9.计算DVAE解码器和GAN生成器共用网络的损失函数；

10.计算GAN判别器网络的损失函数；

11.更新DVAE编码器网络中的参数；

12.更新DVAE解码器和GAN生成器共用网络中的参数；

13.更新GAN的判别器网络中参数；

14.训练周期减1，跳转到步骤2。

3.4 本章小结

本章针对对抗样本的噪声特性和扰动累加效应，提出了一种基于DVAE-GAN的对抗样本防御策略。首先简单介绍了该防御策略的防御思路和模型的整体框架，然后详细介绍了防御模型中的DVAE模块和GAN模块，并解释了各模块对于防御对抗样本的作用。然后分析了各个模块的优化目标及模型的整体目标。最后详细介绍了整个防御的算法流程。

# 4实验及结果分析

4.1实验环境

本文使用的实验硬件环境为：英特尔Xeon E-5-2678v3处理器，64G内存，GPU型号为英伟达RTX2080，显存大小为8G。实验使用的软件环境为：Ubuntu16.04操作系统，编程语言为python3，开发环境为PyCharm，使用的机器学习框架为TensorFlow 1.15.1。本文使用的实验数据集为：MNIST数据集和CIFAR-10数据集。

MNIST数据集是计算机视觉领域使用最多的数据集之一，该数据包含有70000张不同的手写数字图片，每张图片包含28\*28个灰度像素点和一个对应的0-9的标签。MNIST数据集一共分为三个部分：训练集，交叉验证集和测试集。训练集包含55000张图片，交叉验证集包含5000张图片，测试集包含10000张图片。

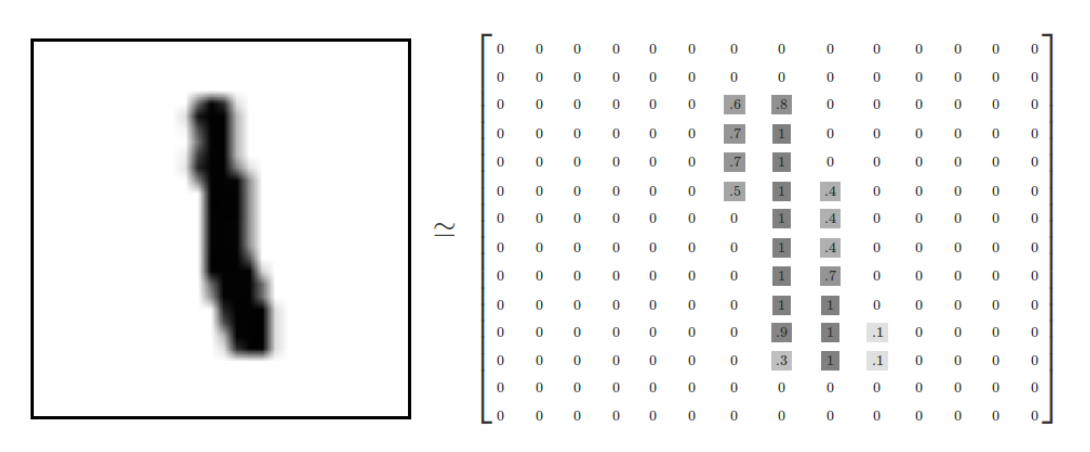


图4.1 MNIST数据集示例



图4.2 CIFAR-10数据集示例

CIFAR-10数据集由6000张彩色图片组成，每张图片包含32\*32个RGB彩色像素点。CIFAR-10数据集包含十个互斥的类别：飞机、鸟、马、青蛙、猫、狗、鹿、卡车、船、汽车。每个分类都包含6000张图片。训练时随机从每类6000张图片中选取1000张图片作为测试集，剩余50000张图片作为训练集。

4.2对抗样本攻防实验设计

4.2.1 攻防目标模型设计

为了验证本文提出的基于DVAE-GAN对抗样本防御策略的有效性，同时检验使用不同网络生成的对抗样本训练成的防御模型是否具有迁移性，本文在MNIST数据集和CIFAR-10数据集中分别实现两个不同分类精度的分类器模型作为对抗样本攻击和防御模型，分类器模型网络结构如表4-1、表4-2、表4-3和表4-4所示。

表4-1 MNIST数据集分类器模型MNIST\_A网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| Conv+ReLU | filters:64, filter\_size:, strides:2, padding:SAME |
| Conv+ReLU | filters:128, filter\_size:, strides:2, padding:SAME |
| Conv+ReLU | filters:128, filter\_size:, strides:1, padding:SAME |
| Flatten | Min\_ndim:3 |
| FC+ReLU | 200 units |
| Softmax | 10 units |

表4-2 MNIST数据集分类器模型MNIST\_B网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| Conv+ReLU | filters:64, filter\_size:, strides:1, padding:SAME |
| Conv+ReLU | filters:64, filter\_size:, strides:1, padding:SAME |
| Max Pooling | pool\_size: |
| Conv+ReLU | filters:128, filter\_size:, strides:1, padding: SAME |
| Conv+ReLU | filters:128, filter\_size:, strides:1, padding: SAME |
| Max Pooling | pool\_size: |
| Flatten | Min\_ndim:3 |
| FC+ReLU | 100 units |
| FC+ReLU | 100 units |
| Softmax | 10 units |

如表4-1所示，MNIST数据集分类器模型MNIST\_A一共有6层网络，第一层是卷积层包含64个卷积核，每个卷积核大小为，卷积核的移动步长为2，激活函数为ReLU函数，边缘填充模式为SAME；第二层也是卷积层包含128个卷积核，每个卷积核大小为，卷积核移动步长为2，激活函数为ReLU函数，边缘填充模式为VALID，第三层也是卷积层包含128个卷积核，每个卷积核大小为，卷积核移动步长为1，激活函数为ReLU函数，边缘填充模式为VALID；第四层是扁平化层用于将多维输入转化为一维输出；第五层是全连接层一共包含100个神经元，激活函数为ReLU函数；最后一是Softmax层作用是输出分类结果。如表4-2所示，MNIST数据集分类器模型MNIST\_B一共有10层网络，第一层和第二层都是卷积层都拥有64卷积核，卷积核大小都是，卷积核的移动步长都是1，边缘填充模式都是SAME；第三层是池化层，池化方法为最大池化法，池化核大小为；第四层和第五层都是卷积层都拥有64卷积核，卷积核大小都是，卷积核的移动步长都是1，边缘填充模式都是VALID；第七层是池化层，池化方法为最大池化法，池化核大小为；第八层和第九层都是全连接层分别包含100个神经元和100个神经元，激活函数都是ReLU函数；最后一层为Softmax层作用是输出分类结果。

表4-3 CIFAR-10数据集分类器模型CIFAR\_A网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| Conv+ReLU | filters:64, filter\_size:, strides:1, padding:SAME |
| Conv+ReLU | filters:64, filter\_size:, strides:1, padding:SAME |
| Max Pooling | pool\_size: |
| Conv+ReLU | filters:128, filter\_size:, strides:1, padding: VALID |
| Conv+ReLU | filters:128, filter\_size:, strides:1, padding: VALID |
| Max Pooling | pool\_size: |
| Flatten | Min\_ndim:3 |
| FC+ReLU | 256 units |
| FC+ReLU | 256 units |
| Softmax | 10 units |

如表4-3所示，CIFAR-10数据集分类器模型CIFAR\_A网络结构与MNIST数据集分类器模型MNIST\_B类似，唯一区别是CIFAR\_A网络的第八层和第九层是256个神经元的全连接层，MNIST\_B网络的第八层和第九层是100个神经元的全连接层。

表4-4 CIFAR-10数据集分类器模型CIFAR\_B网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| Conv+ReLU | filters:64, filter\_size:, strides:1, padding: SAME |
| Conv+ReLU | filters:64, filter\_size:, strides:1, padding: SAME |
| Max Pooling | pool\_size: |
| Conv+ReLU | filters:128, filter\_size:, strides:1, padding: SAME |
| Conv+ReLU | filters:128, filter\_size:, strides:1, padding: SAME |
| Max Pooling | pool\_size: |
| Conv+ReLU | filters:256, filter\_size:, strides:1, padding: SAME |
| Conv+ReLU | filters:256, filter\_size:, strides:1, padding: SAME |
| Conv+ReLU | filters:256, filter\_size:, strides:1, padding: SAME |
| Max Pooling | pool\_size: |
| Conv+ReLU | filters:512, filter\_size:, strides:1, padding: SAME |
| Conv+ReLU | filters:512, filter\_size:, strides:1, padding: SAME |
| Conv+ReLU | filters:512, filter\_size:, strides:1, padding: SAME |
| Max Pooling | pool\_size: |
| Conv+ReLU | filters:512, filter\_size:, strides:, padding:SAME |
| Conv+ReLU | filters:512, filter\_size:, strides:, padding: SAME |
| Conv+ReLU | filters:512, filter\_size:, strides:, padding: SAME |
| Flatten | Min\_ndim:3 |
| FC+ReLU | 4096 units |
| FC+ReLU | 4096 units |
| Softmax | 10 units |

如表4-4所示，CIFAR-10数据集分类器模型CIFAR\_A一共拥有21层网络，前17层除了第三层、第六层、第十层和第十四层之外全是卷积层，除了拥有的卷积核的数量不同，卷积核的大小、激活函数、卷积核的移动步长和边缘填充模式都相同。第三层、第六层、第十层和第十四层都是池化层，池化方式都是最大池化法，池化核大小都是；第十八层是扁平化层用于将多维输入转化为一维输出；第十九层和第二十层都是全连接层分别拥有4096个神经元和10个神经元；最后一层是Sofmax层作用是输出分类结果。

构建完分类器模型后，对各个分类器模型进行训练，训练参数如表4-3所示。对于MNIST\_A和MNIST\_B模型，训练时学习率都设置为0.001，每批次都并行训练100张图片，训练周期分别为15和20。对于CIFAR\_A和CIFAR\_B模型，训练学习率都设置为0.0005，每批次都并行训练100张图片，训练周期分别为50和100。训练完的结果如表4-4所示，MNIST数据集两个分类器模型经过训练后，在测试集的分类准确率分别为97.43%和98.72%；CIFAR-10数据集两个分类器模型经过训练后，在测试集的分类准确率分别为76.92%和89.53%。

表4-3 训练参数表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 训练参数 | MNIST\_A | MNIST\_B | CIFAR\_A | CIFAR\_A |
| Learning Rate | 0.001 | 0.001 | 0.0005 | 0.0005 |
| Batch Size | 100 | 100 | 100 | 100 |
| Epochs | 15 | 20 | 50 | 100 |

表4-4 分类器训练结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MNIST\_A | MNIST\_B | CIFAR\_A | CIFAR\_A |
| 准确率 | 96.43% | 98.72% | 76.92% | 89.53% |

4.2.2攻击方设计

为了测试和定量分析本文提出的基于DVAE-GAN对抗样本防御策略的防御性能，本文选取第二章所介绍的FGSM、BIM和PGD作为攻击方来生成对抗样本，同时使用对生成对抗样本扰动大小进行描述。由于本文提出的对抗样本防御策略未对分类器模型进行改进，选择白盒攻击和黑盒攻击对防御性能评价影响不大，因此本文选取白盒攻击作为攻击方的攻击手段。攻击方具体的攻击方案为：使用FGSM、BIM和PGD生成扰动大小为0.02、0.04、0.06、0.08和0.1的对抗样本，分别使用这些对抗样本攻击无防御分类器模型、。

表4-5 各分类器对不同扰动大小的对抗样本分类准确率

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 生成对抗样本方法 | | 扰动大小 | | | | |
|  |  |  |  |  |
| MNIST\_A | FGSM | 24.32% | 19.53% | 16.45% | 14.26% | 11.55% |
| BIM | 5.1% | 3.15% | 1.53% | 1.12% | 0.85% |
| PGD | 3.24% | 2.34% | 1.62% | 1.03% | 0.78% |
| MNIST\_B | FGSM | 32.33% | 27.62% | 22.53% | 20.34% | 15.43% |
| BIM | 8.26% | 6.36% | 6.52% | 4.32% | 2.37% |
| PGD | 4.47% | 3.81% | 2.75% | 2.17% | 1.81% |
| CIFAR\_A | FGSM | 18.34% | 15.4% | 13.52% | 13.02% | 11.63% |
| BIM | 4.39% | 3.2% | 2.55% | 1.23% | 0.62% |
| PGD | 2.68% | 2.23% | 1.06% | 0.74% | 0.32% |
| CIFAR\_B | FGSM | 26.23% | 21.37% | 16.58% | 14.32% | 13.8% |
| BIM | 8.32% | 7.53% | 4.89% | 3.13% | 2.99% |
| PGD | 7.77% | 6.68% | 5.37% | 2.69% | 1.82% |

在表4.4中每一行数据表示该行分类器对该行生成对抗样本方法生成的对抗样本的分类准确率。通过对比不同生成对抗样本方法对同一个分类器模型攻击结果可以发现：PGD生成的对抗样本攻击性最强，FGSM生成的对抗样本攻击性最弱。通过对比同一个生成对抗样本方法生成不同尺度的扰动攻击同一分类器的结果可以发现：生成扰动越大攻击性越强。此外我们还可以发现分类器模型的复杂程度也会影响对抗样本攻击效果，分类器模型越复杂其防御对抗样本能力要好于简单的分类器模型。

4.2.3防御方设计

针对MNIST数据集和CIFAR-10数据集，本文构建了两个不同深度的基于DVAE-GAN对抗样本防御模型，同时在构建模型过程中本文假设两个防御模型的隐变量维数都是128，构建完后的MNIST数据集DVAE-GAN防御模型各模块网络结构如表4-4、表4-5和表4-6所示，构建完后的CIFAR-10数据集DVAE-GAN防御模型各模块网络结构如表4-7、表4-8和表4-9所示。

如表4-4所示，MINST数据集上DVAE-GAN模型的编码器一共有六层网络，前四层网络是卷积神经网络，分别有64、128、256、256个卷积核，卷积核大小都是，前两层的卷积核步长为2，后两层的卷积核步长为1，边缘填充模式都是SAME，在激活函数激活前都进行归一化（Batch Normalization，BN）处理，激活函数都是LReLU；第五层为扁平化层作用是将多维输入转化为一维输出；最后一层为全连接层作用是输出隐变量的先验分布均值和方差（注意均值和方差是并列连接该全连接层），激活函数为LReLU。

表4-4 MNIST数据集DVAE-GAN编码器网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| BN+Conv+LReLU | filters:64, filter\_size:, strides:2, padding:SAME |
| BN+Conv+LReLU | filters:128, filter\_size:, strides:2, padding:SAME |
| BN+Conv+LReLU | filters:256, filter\_size:, strides:1, padding:SAME |
| BN+Conv+LReLU | filters:256, filter\_size:, strides:1, padding:SAME |
| Flatten | Min\_ndim:3 |
| FC+LReLU | 128 units |

如表4-4所示，MINST数据集上DVAE-GAN模型的生成器（解码器）一共有六层网络，第一层是输入重塑层作用是将输入的隐藏变量形状重塑为，然后将重塑后的结果归一化处理，最后使用ReLU激活函数对归一后的结果进行非线性变换；第二、三、四、五层都是反卷积层，分别有512、256、128、64个卷积核，卷积核大小都是，第二层和第三层的卷积核步长为1，第三层和第四层的卷积核步长为2，边缘填充模式都是SAME，在激活函数激活前都进行数据的归一化处理，激活函数都是ReLU函数；最后一层是输出层使用的激活函数为Tanh激活函数。

表4-5 MNIST数据集DVAE-GAN生成器（解码器）网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| BN+Reshape+ReLU | reshape\_size: |
| BN+Deconv+ReLU | filters:512, filter\_size:, strides:1, padding:SAME |
| BN+Deconv+ReLU | filters:256, filter\_size:, strides:1, padding:SAME |
| BN+Deconv+ReLU | filters:128, filter\_size:, strides:2, padding:SAME |
| BN+Deconv+ReLU | filters:64, filter\_size:, strides:2, padding:SAME |
| Tanh | output |

如表4-6所示MINST数据集上DVAE-GAN模型的判别器一共有五层网络，前四层都是卷积层，分别有64、128、256、512个卷积核，卷积核大小都是，前两层的卷积核步长为2，第三层和第四层的卷积核步长为1，边缘填充模式都是SAME，在激活函数激活前都进行归一化处理，激活函数都是LReLU函数；最后一层是重塑层将输入的多维数据重塑为一维数据，然后再对重塑后的结果进行线性变换得到形状为的数据作为sigmod函数的输入，最后sigmod函数输出就是判别器输入图像是真实图像的概率。

表4-6 MNIST数据集DVAE-GAN判别器网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| BN+Conv+LReLU | filters:64, filter\_size:, strides:2, padding:SAME |
| BN+Conv+LReLU | filters:128, filter\_size:, strides:2, padding:SAME |
| BN+Conv+LReLU | filters:256, filter\_size:, strides:1, padding:SAME |
| BN+Conv+LReLU | filters:512, filter\_size:, strides:1, padding:SAME |
| Reshape+Liner+sigmod | reshape\_size: |

表4-7 CIFAR-10数据集DVAE-GAN编码器网络结构

|  |  |  |
| --- | --- | --- |
| 各层类型 | | 具体结构 |
| BN+Conv+LReLU | filters:64, filter\_size:, strides:1, padding: SAME | |
| BN+Conv+LReLU | filters:64, filter\_size:, strides:2, padding: SAME | |
| BN+Conv+LReLU | filters:128, filter\_size:, strides:1, padding: SAME | |
| BN+Conv+LReLU | filters:128, filter\_size:, strides:2, padding: SAME | |
| BN+Conv+LReLU | filters:256, filter\_size:, strides:1, padding: SAME | |
| BN+Conv+LReLU | filters:256, filter\_size:, strides:2, padding: SAME | |
| BN+Conv+LReLU | filters:512, filter\_size:, strides:1, padding: SAME | |
| BN+Conv+LReLU | filters:512, filter\_size:, strides:2, padding: SAME | |
| Flatten | | Min\_ndim:3 |
| FC+LReLU | | 128 units |

如表4.7所示，CIFAR-10数据集DVAE-GAN编码器一共有十层网络，前八层都是卷积层，分别有64、64、128、128、256、256、512、512个卷积核，卷积核大小都是，奇数层的卷积核步长为1，偶数层的卷积核的步长为2，边缘填充模式都是SAME，激活函数激活前都进行归一化处理，激活函数都是LReLU，第九层是扁平化层作用是将输入的多维数据转化为一维数据，最后一层为全连接层作用是输出隐变量的先验分布均值和方差（注意均值和方差是并列连接该全连接层），激活函数为LReLU。

表4-8 CIFAR-10数据集DVAE-GAN生成器网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| Reshape+ReLU | reshape\_size: |
| BN+Deconv+ReLU | filters:512, filter\_size:, strides:2, padding:SAME |
| BN+Deconv+ReLU | filters:512, filter\_size:, strides:1, padding:SAME |
| BN+Deconv+ReLU | filters:256, filter\_size:, strides:2, padding:SAME |
| BN+Deconv+ReLU | filters:256, filter\_size:, strides:1, padding:SAME |
| BN+Deconv+ReLU | filters:128, filter\_size:, strides:2, padding:SAME |
| BN+Deconv+ReLU | filters:128, filter\_size:, strides:1, padding:SAME |
| BN+Deconv+ReLU | filters:64, filter\_size:, strides:2, padding:SAME |
| BN+Deconv+ReLU | filters:64, filter\_size:, strides:1, padding:SAME |
| Tanh | output |

如表4-8所示，CIFAR-10数据集上DVAE-GAN模型的生成器（解码器）一共有十层网络，第一层是输入重塑层作用是将输入的隐藏变量形状重塑为，然后将重塑后的结果归一化处理，最后使用ReLU激活函数对归一后的结果进行非线性变换；第二、三、四、五、六、七、八、九层都是反卷积层，分别有512、512、256、256、128、128、64、64个卷积核，卷积核大小都是，偶数层的卷积核步长为2，奇数层的卷积核步长为1，边缘填充模式都是SAME，在激活函数激活前都进行数据的归一化处理，激活函数都是ReLU函数；最后一层是输出层使用的激活函数为Tanh激活函数。

如表4-9所示MINST数据集上DVAE-GAN模型的判别器一共有十层网络，前八层都是卷积层，分别有64、64、128、128、256、256、512、512个卷积核，卷积核大小都是 ，奇数层的卷积核步长为2，偶数层的卷积核步长为1，边缘填充模式都是SAME，在激活函数激活前都进行归一化处理，激活函数都是LReLU函数；最后一层是重塑层将输入的多维数据重塑为一维数据，然后再对重塑后的结果进行线性变换得到形状为的数据作为sigmod函数的输入，最后sigmod函数输出就是判别器输入图像是真实图像的概率。

表4-9 MNIST数据集DVAE-GAN判别器网络结构

|  |  |
| --- | --- |
| 各层类型 | 具体结构 |
| BN+Conv+LReLU | filters:64, filter\_size:, strides:1, padding:SAME |
| BN+Conv+LReLU | filters:64, filter\_size:, strides:2, padding:SAME |
| BN+Conv+LReLU | filters:128, filter\_size:, strides:1, padding:SAME |
| BN+Conv+LReLU | filters:128, filter\_size:, strides:2, padding:SAME |
| BN+Conv+LReLU | filters:256, filter\_size:, strides:1, padding:SAME |
| BN+Conv+LReLU | filters:256, filter\_size:, strides:2, padding:SAME |
| BN+Conv+LReLU | filters:512, filter\_size:, strides:1, padding:SAME |
| BN+Conv+LReLU | filters:512, filter\_size:, strides:2, padding:SAME |
| Reshape+Liner+sigmod | reshape\_size: |

在构建完两个DVAE-GAN模型网络结构后需要对两个模型进行训练，具体的训练过程为：使用FGSM、BIM和PGD生成对抗样本方法对本文提出的4个攻防分类器模型进行攻击，对应的每个模型生成五组扰动大小为0.02、0.04、0.06、0.08、0.1的对抗样本，每组对抗样本数量为1000个；然后将这15组对抗样本加上35000张普通训练集中的图像作为训练DVAE-GAN的一个训练数据集，一共能生成2个MNIST数据集防御模型的训练集和2个CIFAR-10数据集防御模型的训练集；最后使用这4个训练集对两个防御模型进行训练，训练时使用的参数如表4-10所示，训练完成后能得到2个MNIST数据防御模型和2个CIFAR-10数据集模型。为了更好的区分这四个防御模型，本文将这4个防御模型命名为：DGMA、DGMB、DCMA和DCMB。DGMA表示使用MNIST\_A分类器生成的对抗样本训练成的MNIST数据集防御模型，DGMB表示使用MNIST\_B分类器生成的对抗样本训练成的MNIST数据集防御模型，DCMA表示使用CIFAR\_A分类器生成的对抗样本训练成的CIFAR-10数据集防御模型，DCMB表示使用CIFAR\_B分类器生成的对抗样本训练成的CIFAR-10数据集防御模型。

表4-10 MNIST数据集和CIFAR-10数据集防御模型训练参数

|  |  |  |
| --- | --- | --- |
| 训练参数 | DVAE-GAN(MNIST) | DVAE-GAN(CIFAR-10) |
| Learning Rate | 0.001 | 0.0005 |
| LReLU | 0.02 | 0.02 |
| Batch size | 100 | 100 |
| Epoch | 100 | 200 |

如表4-10所示，对于MNIST数据集防御模型训练时学习率设置为0.001，激活函数LReLU负数梯度值设置为0.02，每批次并行训练100张图片，训练周期数为100；对于CIFAR-10数据集防御模型训练时学习率设置为0.0005，激活函数LReLU负数梯度值设置为0.02，每批次并行训练100张图片，训练周期为200。

4.3实验结果及分析

为了更好的评价本文提出的防御模型防御性能，本文选取4个主流的同时防御模型作为对比实验组，选取的对比防御模型为：Label Smoothing、Feature Squeezing和PiexlDefend。

4.3.1 MNIST数据集结果

表4-11 在MNIST\_A模型上防御FGSM对抗样本结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 防御策略 | 扰动大小 | | | | | |
|  |  |  |  |  |  |
| Normal | 96.43% | 24.32% | 19.53% | 16.45% | 14.26% | 11.55% |
| Label Smoothing | 95.86% | 88.63% | 75.34% | 65.73% | 53.67% | 52.26% |
| Feature Squeezing | 90.37% | 87.52% | 72.37% | 68.92% | 59.28% | 52.72% |
| PiexlDefend | 92.78% | 90.36% | 85.23% | 84.33% | 84.57% | 83.28% |
| DGMA | 95.27% | 92.74% | 92.82% | 92.34% | 91.37% | 91.21% |
| DGMB | 94.78% | 91.52% | 89.79% | 90.53% | 88.78% | 87.96% |

表4-12 在MNIST\_A模型上防御BIM生成的对抗样本结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 防御策略 | 扰动大小 | | | | | |
|  |  |  |  |  |  |
| Normal | 96.43% | 5.1% | 3.15% | 1.53% | 1.12% | 0.85% |
| Label Smoothing | 95.86% | 40.12% | 28.06% | 18.37% | 13.67% | 12.26% |
| Feature Squeezing | 90.37% | 57.52% | 52.37% | 48.92% | 45.28% | 39.72% |
| PiexlDefend | 92.78% | 85.36% | 83.23% | 80.33% | 80.57% | 78.28% |
| DGMA | 95.27% | 87.74% | 86.43% | 85.83% | 83.37% | 82.21% |
| DGMB | 94.78% | 74.52% | 75.37% | 75.25% | 73.37% | 71.32% |

表4-13 在MNIST\_A模型上防御PGD生成的对抗样本结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 防御策略 | 扰动大小 | | | | | |
|  |  |  |  |  |  |
| Normal | 96.43% | 3.24% | 2.34% | 1.62% | 1.03% | 0.78% |
| Label Smoothing | 95.86% | 10.24 | 5.25% | 3.17% | 2.22% | 5.21% |
| Feature Squeezing | 90.37% | 48.72% | 45.32% | 43.92% | 39.28% | 40.72% |
| PiexlDefend | 92.78% | 69.24% | 68.62% | 65.33% | 60.18% | 58.68% |
| DGMA | 95.27% | 68.14% | 64.32% | 62.34% | 58.82% | 57.32% |
| DGMB | 94.78% | 60.34% | 62.14% | 58.12% | 57.78% | 56.96% |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 防御策略 | | 扰动大小 | | | | | |
|  |  |  |  |  |  |
| FGSM | Normal | 98.72% | 32.33% | 27.62% | 22.53% | 20.34% | 15.43% |
| DGMA | 95.47% | 91.57% | 89.24% | 88.76% | 88.23% | 88.52% |
| DGMB | 96.27% | 94.37% | 92.67% | 91.74% | 91.52% | 90.89% |
| BIM | Normal | 98.72% | 8.26% | 6.36% | 6.52% | 4.32% | 2.37% |
| DGMA | 95.47% | 80.76% | 78.23% | 76.32% | 72.36% | 71.42% |
| DGMB | 96.27% | 90.32% | 87.14% | 84.12% | 83.23% | 80.2% |
| PGD | Normal | 98.72% | 4.47% | 3.81% | 2.75% | 2.17% | 1.81% |
| DGMA | 95.47% | 63.26% | 60.34% | 58.37% | 56.31% | 55.41% |
| DGMB | 96.27% | 70.34% | 67.17% | 66.38% | 65.25% | 62.78% |

表4-13 在CIFAR\_A模型上防御FGSM生成的对抗样本结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 防御方法 | 扰动大小 | | | | | |
|  |  |  |  |  |  |
| Normal | 76.92% | 24.32% | 19.53% | 16.45% | 14.26% | 11.55% |
| Label Smoothing | 75.83% | 69.03% | 65.32% | 55.83% | 42.37% | 35.64% |
| Feature Squeezing | 68.34% | 50.52% | 42.37% | 36.92% | 28.28% | 23.72% |
| PiexlDefend | 72.27% | 65.23% | 61.47% | 58.23% | 57.37% | 56.02% |
| DGMA | 74.37% | 67.89% | 64.37% | 62.53% | 57.32% | 54.17% |
| DGMB | 72.38% | 64.23% | 62.76% | 57.34% | 55.68% | 54.76% |

表4-13 在CIFAR\_A模型上防御BIM生成的对抗样本结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 防御方法 | 扰动大小 | | | | | |
|  |  |  |  |  |  |
| Normal | 76.92% | 4.39% | 3.2% | 2.55% | 1.23% | 0.62% |
| Label Smoothing | 75.83% | 18.23% | 14.32% | 14.83% | 8.37% | 6.64% |
| Feature Squeezing | 68.34% | 30.12% | 33.46% | 25.73% | 16.28% | 8.72% |
| PiexlDefend | 72.27% | 56.88% | 54.67% | 49.13% | 47.37% | 44.34% |
| DGMA | 74.37% | 62.32% | 58.14% | 56.78% | 53.12% | 50.88% |
| DGMB | 72.38% | 55.23% | 52.77% | 50.84% | 46.37% | 45.58% |

表4-15 在CIFAR\_A模型上防御PGD对抗样本结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 防御方法 | 扰动大小 | | | | | |
|  |  |  |  |  |  |
| Normal | 76.92% | 2.68% | 2.23% | 1.06% | 0.74% | 0.32% |
| Label Smoothing | 75.83% | 8.27% | 4.3% | 3.66% | 2.37% | 1.23% |
| Feature Squeezing | 68.34% | 25.57% | 23.42% | 15.23% | 12.67% | 7.37% |
| PiexlDefend | 72.27% | 46.75% | 42.27% | 38.45% | 35.67% | 33.33% |
| DGMA | 74.37% | 54.32% | 52.24% | 49.08% | 48.12% | 46.88% |
| DGMB | 72.38% | 50.37% | 47.32% | 45.63% | 42.37% | 42.58% |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 防御策略 | | 扰动大小 | | | | | |
|  |  |  |  |  |  |
| FGSM | Normal | 89.53% | 26.23% | 21.37% | 16.58% | 14.32% | 13.8% |
| DGMA | 82.36% | 73.62% | 72.32% | 70.28% | 69.34% | 67.15% |
| DGMB | 84.32% | 80.23% | 76.32% | 74.32% | 74.02% | 71.08% |
| BIM | Normal | 89.53% | 8.32% | 7.53% | 4.89% | 3.13% | 2.99% |
| DGMA | 82.36% | 70.34% | 65.68% | 63.07% | 61.21% | 60.31% |
| DGMB | 84.32% | 75.86% | 73.63% | 74.32% | 72.63% | 71.63% |
| PGD | Normal | 89.53% | 7.77% | 6.68% | 5.37% | 2.69% | 1.82% |
| DGMA | 82.36% | 64.07% | 60.06% | 58.53% | 56.47% | 54.68% |
| DGMB | 84.32% | 71.75% | 68.34% | 66.83% | 65.34% | 64.52% |