**本科毕业设计论文**

**题 目** 基于变分自编码网络的图像分类

专业名称 软件工程

学生姓名 程钰翔

指导教师 柯乔

毕业时间 2020.6

摘 要

近年来在深度学习领域有两大模型受到了越来越多的关注，它们分别是变分自编码网络（Variational Auto Encoders，VAE）以及生成对抗网络（Generative Adversarial Networks，GANs），其中变分自编码网络在深度学习领域中的地位日益增加，应用也越来越广泛。目前，变分自编码技术已经在图像处理方面取得很大成就，这些应用包括通过变分自编码网络完成图像生成，根据图像内容分析完成给图片配上相应文字，小样本的图像分类等。

本文围绕变分自编码网络在图像分类方面的应用，首先对变分自编码网络的原理以及研究现状做了总体上的论述，根据变分自编码器的原理以及分类算法的原理对使用变分自编码器实现图像分类的过程进行综述。变分自编码器通过对原始数据进行编码，得到关于原始数据的隐特征，本文对隐特征进行分析，并根据隐特征的特点，使用KNN和决策树分类算法完成分类。并且论述了使用变分自编码器实现图像的分类的可行性以及优越性。

在实验过程中，本文使用多种包含图像数据的数据集，使用的数据集中包括MNIST数据集、PASCAL VOC数据集以及CelebA数据集，对变分自编码器进行训练，统计训练过程中的各类数据并进行分析，得到训练的结果以及各个数据集的特点，并且通过变分自编码器分析输入图像的各类特征，实现了图像的识别与分类。最后，我们通过对比变分自编码器与其他算法的实验结果，分析了各种算法的优劣，对变分自编码器的前景做了总结与展望。

关键词：自编码网络，变分推断，隐特征，图像分类

ABSTRACT

In recent years, two models in the field of deep learning have received more and more attention. They are Variational Auto Encoders (VAE) and Generative Adversarial Networks (GANs). Coding networks are increasing in the field of deep learning, and their applications are becoming more and more widespread. At present, the variational self-encoding technology has made great achievements in image processing. These applications include image generation through the variational self-encoding network, analysis of the image content, matching the pictures with corresponding text, and image classification of small samples.

This article focuses on the application of the variational self-encoding network in image classification. First, it makes a general discussion on the principle and research status of the variational self-encoding network. According to the principle of the variational self-encoder and the classification algorithm, the use of variational The process of image classification by self-encoder is reviewed. The variational autoencoder encodes the original data to obtain the hidden features of the original data. In this paper, the hidden features are analyzed, and according to the characteristics of the hidden features, KNN and decision tree classification algorithms are used to complete the classification. And discusses the feasibility and superiority of using variational autoencoders to achieve image classification.

During the experiment, this paper uses a variety of data sets containing image data. These data sets include the MNIST data set, the PASCAL VOC data set, and the CelebA data set. The variational autoencoder is trained, and various types of data in the training process are counted. And analyze it to get the training results and the characteristics of each data set, and analyze the various characteristics of the input image through the variational autoencoder to realize the image recognition and classification. Finally, by comparing the experimental results of the variational autoencoder with other algorithms, the advantages and disadvantages of various algorithms are analyzed, and the prospects of the variational autoencoder are summarized and prospected.

**Keywords:** self-coding network, variational inference, hidden features, image classification

目 录

**TOC \o "1-3" \h \u HYPERLINK \l \_Toc18567 第一章 绪论 PAGEREF \_Toc18567 5**

HYPERLINK \l \_Toc2418 1.1变分自编码器概述 PAGEREF \_Toc2418 5

HYPERLINK \l \_Toc8929 1.2国内外研究现状 PAGEREF \_Toc8929 5

**HYPERLINK \l \_Toc13266 第二章 变分自编码器原理综述 PAGEREF \_Toc13266 7**

HYPERLINK \l \_Toc19535 2.1 变分自编码原理 PAGEREF \_Toc19535 7

HYPERLINK \l \_Toc4759 2.2 变分自编码器图像分类原理 PAGEREF \_Toc4759 9

**HYPERLINK \l \_Toc31604 第三章 VAE应用图像分类实验环境 PAGEREF \_Toc31604 12**

HYPERLINK \l \_Toc3308 3.1 TensorFlow下实现变分自编码器 PAGEREF \_Toc3308 12

HYPERLINK \l \_Toc14026 3.2 实验数据集 PAGEREF \_Toc14026 12

HYPERLINK \l \_Toc5901 3.2.1 MNIST数据集 PAGEREF \_Toc5901 12

HYPERLINK \l \_Toc12958 3.2.2 PASCAL VOC数据集 PAGEREF \_Toc12958 13

HYPERLINK \l \_Toc26893 3.2.3 CelebA数据集 PAGEREF \_Toc26893 13

HYPERLINK \l \_Toc14026 3.3 分类算法 PAGEREF \_Toc14026 14

HYPERLINK \l \_Toc26893 3.3.1 KNN分类算法 PAGEREF \_Toc26893 14

HYPERLINK \l \_Toc26893 3.3.2 决策树分类算法 PAGEREF \_Toc26893 14

**HYPERLINK \l \_Toc12233 第四章 图像识别分类实验过程与结果 PAGEREF \_Toc12233 16**

HYPERLINK \l \_Toc6412 4.1 MNIST数据集上实现图像分类 16

HYPERLINK \l \_Toc9759 4.1.1 MNIST数据集上VAE训练过程 16

HYPERLINK \l \_Toc9759 4.1.2 MNIST数据集上VAE训练结果 17

HYPERLINK \l \_Toc9759 4.1.3 VAE在MNIST数据集上实现图像识别分类 21

HYPERLINK \l \_Toc9759 4.1.4 噪声因素对VAE训练结果的影响 PAGEREF \_Toc9759 23

HYPERLINK \l \_Toc6412 4.2 PASCAL VOC数据集上实现图像分类 PAGEREF \_Toc6412 24

HYPERLINK \l \_Toc22825 4.2.1 PASCAL VOC数据集上的训练过程 PAGEREF \_Toc22825 24

HYPERLINK \l \_Toc6739 4.2.2 VAE在PASCAL VOC数据集上的分类实现 PAGEREF \_Toc6739 25

HYPERLINK \l \_Toc11717 4.3 CelebA数据集上实现图像识别分类 PAGEREF \_Toc11717 29

HYPERLINK \l \_Toc7247 4.3.1 VAE在CelebA数据集上的训练过程 PAGEREF \_Toc7247 29

HYPERLINK \l \_Toc3327 4.3.2 VAE在CelebA数据集上的分类实现 PAGEREF \_Toc3327 32

**HYPERLINK \l \_Toc8504 第五章 VAE与其他算法的对比 PAGEREF \_Toc8504 33**

HYPERLINK \l \_Toc8226 5.1 自编码器 PAGEREF \_Toc8226 33

HYPERLINK \l \_Toc7853 5.2 AE与VAE分类精度上的比较 PAGEREF \_Toc7853 33

**HYPERLINK \l \_Toc20834 第六章 实验总结与论文总结 PAGEREF \_Toc20834 35**

HYPERLINK \l \_Toc29715 6.1 论文过程与总结 PAGEREF \_Toc29715 35

HYPERLINK \l \_Toc16388 6.1.1 理论过程 PAGEREF \_Toc16388 35

HYPERLINK \l \_Toc21204 6.1.2 数据总结 PAGEREF \_Toc21204 35

HYPERLINK \l \_Toc15984 6.2 变分自编码器的应用前景 PAGEREF \_Toc15984 37

HYPERLINK \l \_Toc25039 参考文献 PAGEREF \_Toc25039 39

HYPERLINK \l \_Toc26152 致 谢 PAGEREF \_Toc26152 40

HYPERLINK \l \_Toc14559 毕业设计小结 PAGEREF \_Toc14559 41

HYPERLINK \l \_Toc3938 附 录 PAGEREF \_Toc3938 42

第一章 绪论

1.1变分自编码器概述

近年来，随着机器学习的快速发展，许多先前未被考虑到的复杂场景都开始受到人们的关注。在这些场景中，进行推断和训练常常是困难且耗费巨大的。一方面，许多经典的算法在进行推断和训练时，常常有一些难以满足的条件或太强的限制，因此这些算法不能满足复杂场景的需求。另一方面，由于学术界逐渐出现了训练成本较低的高效方法。因此，一种能应用到复杂场景，且能够用小批量梯度下降训练的推断模型成为自然的发展需求，而变分自编码器（Variational Auto Encoders，VAE）正是这种发展需求的产物。

VAE成功地将神经网络应用到了推断问题中，同时也解决了连续数据的生成问题。在推断问题上，它具有训练快、成本低等优点；在生成问题上，它也具有训练快、高稳定性、高多样性、图片重建质量高等优点，但同时具有生成图片的清晰度低等缺点[1]。

如今VAE已经被应用到许多需要进行推断的场景中，比如在强化学习中用于推断对手的下一步动作上；作为生成模型VAE也同样得到广泛的应用，比如利用VAE开发出的快速的视频换脸工具。目前，变分自编码技术已经在图像处理方面取得很大成就，这些应用包括通过变分自编码网络完成图像生成、根据图像内容分析完成给图片配上相应文字、小样本的图像分类等。

1.2国内外研究现状

国内外在变分自编码网络的主要研究方向在图像处理方面，其中包括图像分类、目标检测识别、图像降噪、图文配字等主要技术。就国内外研究的趋势来看，变分自编码器在图像处理方面的技术要比在其他方面的技术更加成熟一点，并且图像处理领域涉及各种各样的技术，因此，变分自编码网络在图像处理方面具有很大的发展前景。

总体来说，国内外变分自编器的研究已经趋近成熟，根据变分自编码器的原理，国内外学者将VAE应用到了各个方面，比如图像识别、图像分类、图文配字、图像生成以及不在图像处理领域的机械故障评估，战斗形势判断，异常检测，轨迹规划等。

而且对于变分自编码器的改进方案也在层出不穷，有基于条件的变分自编码器（CVAE），变分公平自编码器（VFAE），以及变分自编码器与生成对抗网络相结合融合网络（VAE-GAN）[1]。变分自编码技术正在快速的发展，国内外学者所做的贡献将会对这次的课题研究带来很大的帮助。

变分自编码器在各个领域内都有很广泛的应用，由于变分自编码器可以通过训练生成不同类型的数据，在音乐的乐谱生成中，我们可以通过VAE实现乐谱的自动生成。通过变分自编码器实现的乐谱生成系统，不仅满足了非专业人员进行音乐创作的需求，与此同时该技术也可以大幅降低技术门槛，节约制作成本。由于乐谱中的信息是人们通过对各种音乐符号的定义来进行解释的，因此对于机器来说我们需要一套特定的数据定义方法来表示不同的音乐符号。在传统的学习方法中，由于缺乏成熟的定义手段，在乐谱的自动生成这一方面是具有一定发展限制的。但是在变分自编码器的生成模型中，VAE并不需要对原始的乐谱数据有具体的解析以及定义，VAE需要做的是根据原始的乐谱图像，抓取该乐谱的主要特征信息，然后人们就可以通过对这些信息的加工或处理，生成新的乐谱[2]。

变分自编码技术也可以应用于高光谱的图像分类中去。在日常生活中，高光谱图像对于人们的很多项目活动都有重要的作用，所谓高光谱图像就是有多个波段组成的一种遥感图像，其主要特点是包含的波段数量多，并且为遥感数据。高光谱图像在地质监测与各种农业活动中具有很广泛的应用。由于高光谱图像属于遥感图像并且包含的大量的波段数据，因此在传统的分类方法中，想要实现对于此类图像的精确分类，需要耗费大量的物力以及财力。高光谱图像在数据空间具有较高的维度，如果强行追求精确分类，可能会导致机器同时处理的数据过多，造成“维度爆炸”。与此同时，当处理的数据维度过高时，对计算机的性能要求也会十分苛刻，此时运算的速度会明显降低，效率就会不足。因此，我们需要新的技术来处理此类图像，由于VAE在数据处理的过程中有降维的过程，并且可以分析图像的特征实现图像的分类，解决了传统分类方法对高光谱图像处理能力有限的问题，因此VAE在高光谱的图像分类过程中是具有很广泛的应用前景的[3]。

除了传统的图像处理，变分自编码器还可以通过对图像的分析，得到图像中的信息，对图像的内容进行判断以及决策。其中VAE在空战态势分析上也得到了具体的应用。所谓空战态势评估，就是机器通过对空战中的得到的图像进行分析和处理，为下一步的行动提供最优的决策，同时对战斗的形势作出判断。传统的态势评估方法对于数据的处理主体还是在用户身上，这种态势评估的主观性太强，并且在面对大量的数据时，其处理能力就会远远下降。VAE在战斗态势的评估中就可以避免这些问题，由于VAE可以根据人们定义的基本战斗策略与其分析的图片进行匹配，并且可以不停通过训练来寻求最优的策略，大大提高了数据分析的效率。现代化空战中，作战双方在战前以及战中能够得到的信息量是非常大的，这也就要求态势评估的机器要在短时间内得到优秀的策略，如果耗时过长，那么得到的策略信息就是不及时的，当然这些策略信息也就失去了一定的价值。在VAE中，我们可以对其进行大量的数据训练，满足了现代化空战中的大数据的特点，因此具有很好的应用前景[4]。

变分自编码器原理综述

2.1 变分自编码原理

变分自编码器是由自编码器衍伸而来，自编码器（Auto Encoders，AE）是一种的学习模型，它主要由编码器（Encoder）和解码器（Decoder）组成，其中编码器挖掘数据的隐含特征，解码器通过数据的隐含特征实现最小化生成数据和原始数据的误差。变分自编码器在自编码器的基础上，对自编码器的编码和解码阶段进行改进，加入了一定的参数约束而不是直接通过特定的编码来完成这个过程。

在传统的变分自编码器中，原始的输入数据一般会经过编码器进行降维处理，我们知道原始的图像数据是具有一定维度的。在VAE中编码器的目的就是对原始的输入数据进行处理，VAE将原始的图像数据分布通过编码器转换为正态分布，这也就体现了变分的思想。正态分布需要两个值来进行确定，均值和方差，编码器通过计算均值与方差将原始的数据映射到了低维的空间上，这个空间我们称为隐空间。在隐空间内，每一个正态分布都对应于一个原始图像数据的分布。隐空间的数据也映射出了原始数据分布的特征。

接下来VAE在隐空间内对生成的正态分布做采样处理，得到了采样变量，该采样变量与原始数据变量为一一对应的关系。此时，解码器通过对采样变量的分析，分析其在隐空间内的特征结构，尝试去重构原始变量，最终得到新的图像数据。训练VAE的过程就是迭代执行上述流程的过程，当VAE完成训练之后，解码器就可以识别出与原始变量相关的特征，尽量准确地生成新的数据了。

图2-1 变分自编码器数据处理流程

我们假设现在有一批初始数据样本{X1...Xk},变分自编码器的基本实现过程是通过编码器得到一个关于初始数据样本的隐变量{Z1...Zk}，然后构建一个生成模型X = g（Z），生成新的概率分布。具体来说，对于初始的数据样本Xk，我们假设生成一个专属于Xk的概率分布Zk，在变分自编码器中，该分布为正态分布。在这里，因为每个Zk都是专属于一个初始样本Xk的正态分布，所以我们只要对Zk进行采样，再通过生成器还原为新的Xk就可以了。

首先，我们介绍隐变量Z的构建过程，也就是VAE中编码器的原理与功能，我们知道，每个生成隐变量的分布是专属于一个初始变量的，并且我们希望生成的隐变量为正态分布，正态分布具有两个主要的参数特征：均值μ和方差σ2。因此，生成器需要根据初始变量的分布计算出每个初始变量对应隐变量分布的均值与方差，对应计算公式[5]：

EMBED Equation.KSEE3 \\* MERGEFORMAT （1-1）

EMBED Equation.KSEE3 \\* MERGEFORMAT （1-2）

均值和方差计算函数由编码器的神经网络计算实现，实际上，VAE在重构隐变量Z的过程中并不知道Z的具体分布，而是通过采样来得到关于Z的分布，这样也就导致了在重构过程中是有误差存在的。这样的话解码器在生成过程中就会收到重构误差，也就是噪声的影响。我们知道，构建过程需要计算的一个值为方差，方差的大小会影响到解码器生成过程中的噪声程度。当方差处于较低水平时，此时由于采样导致的误差就会比较小，那么此时的重构的难度会降低。反之，方差处于较高水平时，重构误差就会增大，重构的难度会提高。由此，我们可以知道，得到方差越小的那一组初始数据在重构过程中解码器的生成效果会越好。

实际上，方差的大小的确会影响到生成效果，但是理想状态却并不是方差为0的时候，当编码器计算方差时，如果方差为0，那么生成的隐变量Z将不再符合我们所假设的中间变量Z为正态分布的条件，此时的编码器便不再具有VAE的特性，退化成为普通的AE编码器。因此，在编码器生成的过程中我们应该避免上述情况的出现，因此，在VAE的编码生成隐变量的过程中，编码器采取了一定的措施避免了这种情况的出现。在编码过程中，编码器对均值和方差的计算过程做了改进，在重构误差的基础上添加了额外的损失函数。在这里，作者给出了一种添加额外损失函数的方法[5]：

EMBED Equation.KSEE3 \\* MERGEFORMAT (1-3)

EMBED Equation.KSEE3 \\* MERGEFORMAT (1-4)

其中，d表示输入样本数据的维度，公式1-3的计算结果为重构误差在均值上的额外增加，公式1-4的计算结果是重构误差在方差上的额外增加。通过上述方法，计算出来的均值与方差会使隐变量Z的分布更加接近于标准正态分布，也就避免了因为方差为0而导致的编码器退化。

其次，在解码过程中，我们可以通过调节噪声来调节重构误差的方法来完成对解码器生成能力的训练，当解码器的生成能力较低时，我们可以调节噪声变小，使生成隐变量的方差处于较小的状态，这样解码器的重构难度就会降低，有利于解码器生成图像，当解码器的生成能力达到一定水平时，我们可以调节噪声变大，此时隐变量的方差就会增加，随之重构难度增加，以此达到训练解码器重构能力的目的。

2.2 变分自编码器图像分类原理

变分自编码在实际的应用场景中也具有比较大价值，图像处理是变分自编码器的一个主要应用场景，图像处理方面，VAE可以通过输入图像的数据样本分布，通过解码器得到关于输入数据的隐变量，通过隐变量的特点，变分自编码器可以完成图像的降噪，图像的分类，图像的匹配文字，以及图像的优化等各种功能，在本论文中，我们主要讨论变分自编码器在图像分类方面的应用。

通常情况下，变分自编码器通过对输入样本图像的分析得到相应的隐特征，也就是专属于图像数据分布的一个正态分布。然后对需要实验的数据进行同样的步骤得到各种图像的隐特征，当我们输入了足够多的数据样本后，此时的变分自编码网络就已经学习到了各类数据图像的隐特征。这个时候我们对VAE学习得到的图像隐特征进行分析和总结。使用分类算法构建分类器，每次输入不同类型的图像时，分类器就可以直接判断与该图像的隐特征最相似的图像特征，就可以实现对图像的分类[6]。

INCLUDEPICTURE \d "http://rtt.5read.com/pdgpath/15/155e/155ec247283cf97f88c1c68874158779/807ee92f10e9b63faa790fc085d6b13c.jpg" \\* MERGEFORMATINET

图2-2 变分自编码处理图像的过程

VAE可以通过训练达到对图像识别分类的效果，在实际应用中，我们可以将其应用于各种不同的场景，这里我们举例几个应用场景：

变分自编码技术可以应用于简单图片的图文匹配，VAE经过一定的训练可以识别图像的特征，那么我们只需要在解码过程中根据我们的需要定义不同的图片种类。然后解码器在输出的时候就会根据图片的特征对应到不同的种类，作者给出了相应的实验结果[7]，如图2-3所示。

图2-3 VAE图像分类的应用

通过训练，VAE可以输出狮子和狗的图像，此时解码器已经能够识别这两种图像的区别，然后其只需要与我们定义的图片种类对应，我们就可以通过VAE比不同的动物图片配上动物的名字。

变分自编码技术可以应用于车牌的识别，在基于MNIST数据集的实验中我们可以训练得到能够识别数字的解码器。通过图2-4我们可以看到，在经过了若干次迭代后，VAE的解码器已经可以分辨不同手写的数字图像的特征，并将它们输出。机动车车牌的数字由于是印刷体，每个相同数字它们的图像都是完全一致的，VAE可以通过训练识别具有不同特征的手写数字，因此也可以通过训练实现对车牌印刷体数字图像的识别。在其他文献中，作者已经将其应用于实践中结果如下图所示[8]。

图2-4 VAE在车牌识别中的应用

通过对VAE做改进，我们还可以实现更多功能，国外已经有研究学者通过改进之后的半监督式的VAE实现了更加复杂的图文配字，如图2-5所示。

图2-5 VAE在更加复杂的图像分类中的应用

图2-5中描述的实验结果来自于国外学者的实验结果，经过训练后的VAE可以直接识别图中人物，地点，活动等众多因素，实现了更加强大的图像分类识别功能[9]。

第三章 VAE应用图像分类实验环境

3.1 TensorFlow下实现变分自编码器

变分自编码器是基于神经网络的生成模型，本实验使用Tensorflow为实验环境，Tensorflow是广泛使用的实现机器学习以及其它涉及大量数学运算的算法库之一。Tensorflow由Google开发，是GitHub上最受欢迎的机器学习库之一。Google几乎在所有应用程序中都使用Tensorflow来实现机器学习。

Tensorflow主要处理的数据对象为多维数据，处理的过程可以被视为流，也就是可以使用数据流图的方式进行数值计算，在深度学习领域有很多算法模型都符合Tensorflow的计算模式，我们可以从第一章中得出，Tensorflow是适合变分自编码器进行运算的环境。

Tensorflow分为GPU版本与CPU版本，它们的主要运算不同在于运算方式上。GPU拥有更多的可以用于计算的寄存器，并且cache的数量较少，可以同时进行比CPU更多线程的运算，更适合图像的处理。因此，GPU版本的Tensorflow更加适合变分自编码器的运算。本次GPU版本的Tensorflow的环境如下：

Python3.6版本（或者Anaconda）

NVIDIA GPU Computing Toolkit CUDA

cudnn-10-windows10-x64（与CUDA版本相对）

3.2 实验数据集

3.2.1 MNIST数据集

MNIST数据集是一个手写数字图片的数据集。该数据集包含60000个用于训练的示例和10000个用于测试的示例。这些数字已经过尺寸标准化并位于图像中心，图像是固定大小(28x28像素)，其值为0到1。在VAE的图像处理应用方面，已有部分学者使用该数据集。

INCLUDEPICTURE \d "C:\\Users\\程钰翔\\AppData\\Roaming\\Tencent\\Users\\1520479617\\QQ\\WinTemp\\RichOle\\@JO%NXQCC[T3WXH01G@P8XG.png" \\* MERGEFORMATINET

图3-1 MNIST数据集示意图

3.2.2 PASCAL VOC数据集

Pascal VOC 2012数据集作为基准数据之一，在对象检测、图像分割网络对比实验与图像分类中被频频使用，主要有四个大类别，分别是人、常见动物、交通车辆、室内家具用品。在这四个大类别之下一共包含了20个小类别，对应20种不同类型的图像。该数据集主要提供图像分类、对象检测识别、图像分割三类服务。

INCLUDEPICTURE \d "https://ss1.bdstatic.com/70cFuXSh\_Q1YnxGkpoWK1HF6hhy/it/u=844067022,3960075027&fm=26&gp=0.jpg" \\* MERGEFORMATINET

图3-2 PASCAL VOC数据集示意图

3.2.3 CelebA数据集

CelebA是CelebFaces Attribute的缩写，意为名人人脸属性数据集，其包含10,177个名人身份的202,599张人脸图片，每张图片都做好了特征标记，包含人脸bbox标注框、5个人脸特征点坐标以及40个属性标记，CelebA由香港中文大学开放提供，广泛用于人脸相关的计算机视觉训练任务，可用于人脸属性标识训练、人脸检测训练以及landmark标记等。

INCLUDEPICTURE \d "https://ss0.bdstatic.com/70cFvHSh\_Q1YnxGkpoWK1HF6hhy/it/u=3620251550,3863126198&fm=26&gp=0.jpg" \\* MERGEFORMATINET

图3-3 CelebA数据集示意图

3.3 分类算法

3.3.1 KNN分类算法

KNN算法的分类原理就是通过分析一个样本在特征空间中最近的几个样本的标签属性来进行分类的。其工作原理如下：首先将要分类的图像特征映射到特征空间中去，然后在特征空间中找到k个与其最近的特征值，查看这些特征值中出现次数最多的特征值，最后查看该特征值的标签，则要分类的图像就属于该类别。具体的分类过程如下图所示：

图3-4 KNN分类算法示意图

图中绿色的点就是我们要预测的那个点，假设K=3。那么KNN算法就会找到与它距离最近的三个点（在这里我们用圆圈把它圈起来），看看哪种类别多，该图中是蓝色三角形多，新来的绿色点就应该归类到蓝色三角形。

当K=5的时候，判定就变成不一样了。这次变成红色圆形多一些，所以该绿点被归类成红色圆形。从这个例子中，我们就能看得出K的取值是很重要的。

3.3.2 决策树分类算法

通过树形的逻辑结构一步一步进行分类，最开始通过根结点的逻辑判断进入哪一个分支，到达左子树或右子树的结点进行下一次判断，直到到达根结点输出分类标签。便完成了对图像的分类。

举例说明，我们现在有一批苹果，根据属性对其分类，属性图如下：

表3-1 苹果分类的属性图

样本中有2个属性，A0表示是否红苹果。A1表示是否大苹果。

INCLUDEPICTURE \d "https://img-blog.csdn.net/20141119133743342" \\* MERGEFORMATINET

图3-6 决策树算法示意图

以上图的决策树为例，我们可以通过逻辑结构来判断出要分类的苹果属于好还是坏。在VAE实现图像分类的过程中，我们根据VAE学习到的数据特征来设计决策树，从而实现对图像的分类。

第四章 图像识别分类实验过程与结果

4.1 MNIST数据集上实现图像分类

4.1.1 MNIST数据集上VAE训练过程

首先，VAE需要在数据集上进行运算，因此实验中我们使用MNIST数据集，该数据集是一个存储正常人手写数字图片的数据集。MNIST 图像的维度是 28\*28 像素，只有单色通道。我们的输入数据 X\_in 是一批一批的 MNIST 字符，变分自编码网络会学习如何重建它们。然后在一个占位符 Y 中输出它们，输出和输入具有相同的维度。

图4-1 导入MNIST数据集

根据变分自编码器对图像的处理过程，我们可以把变分自编码器分为两个部分，即编码器和解码器。在论文第二章中，我们知道输入的样本数据通过编码器的神经网络的计算，得到两组数据，分别是均值与方差。然后通过均值方差计算得到隐变量Z，对Z采样之后的结果作为解码器的输入。编码器函数（encoder）的输入参数中，X\_in表示输入的字符图像。编码器函数的功能就是通过对输入数据的计算得到均值与方差。其中均值（mean）我们用参数mn表示，方差这里我们用标准差代替，标准差（std）我们用sd来表示。然后，我们利用tensorflow的multiply()函数计算出隐变量z，然后通过对z采样得到sampled\_z，该变量作为解码器的输入参数。

图4-2 VAE的编码器

在论文第二章中，我们已经知道如果不对编码器生成的方差做处理，那么在训练过程中方差可能出现为0的情况，此时模型就会退化为自编码器。因此在这里我们处理标准差的时候，需要添加额外的损失函数对标准差进行优化。添加一个损失函数来处理标准差，使得到的隐变量z的分布更接近于标准正态分布。

图4-3 VAE的中间变量标准差的优化函数

在解码器中，我们以变量sampled\_z作为输入，该变量是通过对隐变量z采样得到的。这里采样得到的结果与被采样的变量是不一样的，采样得到的结果往往是离散变量的而不是连续的变量。解码器的功能是尝试从采样得到的数据中恢复输入样本原有的数据信息，在这里我们可以通过调节解码器中的参数来改变解码器还原数据的难度，当解码器训练次数较少时，其复原数据的能力不够，需要降低复原难度。随着训练次数增加，解码器会增加复原难度。通过tensorflow的reshape()函数重塑得到输出数据img。然后可以通过plt.imshow()函数展示生成的图像。

图4-4 VAE的解码器

迭代过程一共迭代2000次，每400次作为一个节点输出得到的数字。然后我们统计每次输出的图像，在经过这次训练之后，解码器就可以根据隐变量sampled\_z的特征识别不同的数字，输出不同的数字。

图4-5 主函数训练解码器

4.1.2 MNIST数据集上VAE训练结果

实验中我们的主要目的是为了让VAE通过训练来分类识别从1到9的手写数字。首先针对不同的数字，我们希望解码器通过一定次数的训练能够达到高度还原初始的图像。解码器的训练次数与输出得到的结果如下。图中我们可以得到，在迭代次数400次左右时，图像的还原程度并不高，VAE的输出结果还不满足能够进行图像分类识别的预期。随着迭代次数的增加，图像的输出与原始图像相似程度越来越高，在迭代2000次左右时，VAE基本能够还原原始图像。

表4-1 部分实验数据迭代过程

当解码器初步完成训练时，我们可以增加损失函数对计算出来的方差的影响。在有噪音的干扰下，解码器的还原效果可能会比较差。此时，我们可以进行第二次迭代训练，当解码器经过一定次数的迭代训练后，其就具备了识别分类图像的功能。

当解码器训练到一定程度时，我们可以直接将具有不同手写数字的图像作为输入数据，解码器此时已经具备一定的识别能力，通过分析图像的隐特征，输出不同的数字，完成对从1到9不同数字的分类识别。训练结果如图3-8所示，可以看出训练完成的VAE输出的结果已经和原数据差别很小了。

图4-6 训练后的VAE图像输出结果

变分自编码器在训练过程中，编码器需要计算的是满足每一个输入变量的均值与方差，方差可以用标准差来代替，每一组计算得到的均值与方差与原始输入变量的关系是一一对应的。均值与方差对应了该输入图像数据在隐空间上的变量分布，对于输入图像来说，这些数据隐含了该图像在隐空间上的主要特征，在一定程度上可以体现不同图像的特点，也体现了VAE完成训练后对图像分类的效果。

隐空间的数据是否在分布上具有明显的分类，在一定程度上体现了VAE是否能够正确的对图像进行分类，我们在实验过程中对中间过程产生的均值与标准差，也就是代表隐变量的特征值进行了统计。在经历了足够多次的训练结果后，我们分别输入多个代表不同手写数字的图像，得到了以下的输出结果。

图4-7 完成训练的VAE在MNIST数据集实验输出结果

在输出结果中，我们可以看到训练完成的VAE已经可以生成肉眼可以识别的数字图像了，但是从输出结果我们并不能分析训练完成后的VAE对于不同的数字图像是否具有较强的识别能力，为了避免单一输出结果带来的误差，我们可以在图像的生成过程中采集中间变量，也就是上文中提到的有编码器计算的均值与标准差变量。由于中间变量只有两个种类，我们分别用坐标系的横纵坐标代替这两种变量，用不同的颜色标注不同的数字，最终得到以下结果。在该结果中我们可以准确的看到输入变量在隐空间的分布。

图4-8 训练完成的VAE中间变量特征分布散点图

中间变量一共有两个特征点，分别是均值和标准差，图中以横坐标代表均值，纵坐标代表标准差，这里我们对均值和标准差做了处理，将其变化范围映射到了区间[-4,4]上，这样我们可以通过散点图分析训练结果。不同的颜色从下往上分别代表了不同的数字，每一个散点代表图4-9中的一个数字图像。

分析结果如下：

通过中间变量的散点特征分布，我们可以看到不同颜色的点的分布基本满足了分布在较为集中的一点上，不同颜色的散点有交叉的部分，通过交叉的部分我们可以分析MNIST数据集上不同手写数字图像可能存在的相似度，总体上来看，此时的VAE已经具备了识别图像、分类图像的能力。

图中明显具有交集的是绿色代表的数字4和棕色代表的数字9，通过观察我们发现，在MNIST数据集中，手写数字4与9在特征上存在较大的相似之处，这也是在训练过程中出现不同图像中间变量的特征点有重合之处的原因。与此同时，具有交集的散点还有手写数字0和6，手写数字2和8等等。我们对具有交集的数字进行了统计，使用以下邻接矩阵来表示，其中Y表示两个数字有交集，N表示两个数字没有交集。

表4-2 MNIST数据集上VAE训练结果散点图中数字的交集结果

从结果表格中我们可以看到在手写数字中，具有相似特征的数字往往在隐空间上具有一定的交集，这也说明了VAE最终训练的结果已经接近了我们肉眼的识别能力。即能够识别出特征差异较大的数字图像，也能够对不同数字图像的相似性进行判断，完成对图像的分类。

通过图中所有散点的分布我们可以发现，所有输入图像经过编码器生成的中间变量的分布是较为集中的，所有变量的分布大约呈圆形，圆心的位置大约在坐标原点处。在这里我们虽然对中间变量的结果进行了处理，但是这些中间变量依旧保持了较为集中的分布，因此我们可以分析出MNIST数据集的图像在特征上具有相似之处，不同的图像虽然特征不同，但是其在隐空间中的特征分布总保持在一定的范围内。

总结：MNIST数据集的数据在特征上具有一定的集中性，但是不同数字对应到的特征也有其一定的差别，这一点可以在上文的散点图中展示。总体来说数据集的数据特征与手写数字的基本特征是吻合的。

4.1.3 VAE在MNIST数据集上实现图像识别分类

在传统的机器学习问题中有很多种算法用于图像的分类，在这里我们主要使用两种算法对图像进行分类，主要是K-近邻算法（k-NearestNeighbor，KNN）以及决策树算法（Decision Tree）。

在KNN算法中，我们需要定义k的值。因为KNN算法的分类原理就是通过分析一个样本在特征空间中最近的几个样本的标签属性来进行分类的。其工作原理如下：首先将要分类的图像特征映射到特征空间中去，然后在特征空间中找到k个与其最近的特征值，查看这些特征值中出现次数最多的特征值，最后查看该特征值的标签，则要分类的图像就属于该类别。在这里我们以图4-9为特征空间，取k的值为10，得到的分类结果如下：

表4-3 基于KNN算法分类结果

从图中我们可以看出，手写数字的总体分类精度满足我们完成分类的条件，但是其中数字4、数字5和数字9的分类精度较低，与其他的结果并没有保持一致。这一点我们可以追溯到图4-9中去，我们可以在图中看到数字4和数字9的特征点有着大量的重合，这会影响到KNN算法的分类结果。同时，数字5的特征点分布具有两个集中的区域，这会导致KNN算法在判断时有时只能取到一个区域内的特征点，从而产生分类误差，这也是数字5的分类精度明显较低的原因。

在决策树算法中，我们需要通过树形的逻辑结构一步一步进行分类，通过根结点的逻辑给需要分类的图像指定一个方向，到达左子树或右子树进行下一次分类，直到到达根结点就算是完成了对图像的分类。在这里我们采用如下决策树：

图4-9 决策树算法的决策树逻辑结构

通过决策树，我们可以通过一步一步确定坐标的方式逐步锁定待分类图像在特征空间的位置，从而锁定其类型。最终得到的分类结果如下：

表4-4 基于决策树算法分类结果

从图中我们可以看出，即使我们换了一种分类的方法，最终数字4和数字9的分类精度还是没有太大的提高。这一点我们可以从特征散点图中看出，但是数字5的分类精度却提高了。分析其原因主要是因为在KNN算法中，我们只关心k个最相似的特征值，但是数字5的分布有两个集中的区域，在决策树算法中，我们可以通过逻辑定位，找到这两个区域分类识别到数字5，这样分类的精度就会有所提高，但是决策树的方法也有缺陷，对于分布并不是很集中的特征值分类精度会下降。总体看来两种算法的分类精度没有太大差别。

4.1.4 噪声因素对VAE训练结果的影响

在论文的第一章中，我们可以知道在VAE的解码器训练过程中，编码器生成的隐变量可能出现方差为0的情况，然后若不加噪声处理，VAE会退化为普通的自编码器。我们可以动态的调整噪声的大小从而达到训练解码器的目的。当解码器刚开始工作时，调整噪声处于较小的状态可以更好的生成数据，当解码器具备一定能力时，调整噪声变大可以更好的训练解码器。

我们以迭代50次为一个记录点，分别得到在加入噪声和没有加入噪声时解码器的输出结果。如下图所示，我们可以看出，同样是迭代50次，加入噪声后的VAE生成图像的效果远远不如没有加入噪声。我们知道，适当的噪声有利于生成器的训练，因此我们打算继续迭代，观察VAE的生成效果。

图4-10 噪声对VAE初始生成效果的影响

为了探索噪声对VAE的生成效果的影响具体是怎样的，我们分别设置了3组实验，在第一组实验中，我们对VAE不做任何加入噪声的处理；第二组实验，我们在VAE的训练过程中加入了适量的噪声；第三组实验中，我们在VAE的训练过程中加入了较大的噪声，我们只需要比对三组实验的训练结果，就可以分析噪声在VAE训练过程中的影响了。

表4-5 噪声对VAE迭代次数的影响

通过结果我们可以知道，在不加入噪声的情况下，VAE一样可以完成训练，适当的加入噪声可以缩短训练时间，当加入过量的噪声时，VAE可能会训练失败。（这里噪声过大导致每次输出结果很差，一直没有的到理想的训练结果）。分析原因，首先噪声可以促进解码器生成能力的训练，当编码器生成的隐变量被加入了一定的噪声，那么解码器要想办法提高自己的生成能力，在这个过程中，解码器会不断的训练自己提高生成能力。但是当我们加入的噪声比较大时，我们知道中间隐变量会产生较大的变化，解码器的输入是根据隐变量采样而得来的，此时隐变量由于较大噪声的影响已经失去了大部分特征，解码器的生成难度很高。这样就会导致生成结果一直都很差，随机性很强。

我们知道噪声会对VAE的训练结果会有影响，此时的VAE学习得到的图像隐特征也不同，此时的分类结果也会受到影响。我们通过实验来进行分析。分别对没有加入噪声，加入一定噪声以及加入过量噪声的VAE进行分类实验，分类的方法采用KNN算法，得到的结果如下：

表4-6 噪声对图像分类结果的影响

通过结果我们可以看到，加入了一定噪声和没有加入噪声训练完成的VAE在图像分类中分类精度没有太大差距，但是在训练过程中加入了过量噪声的VAE由于没有学习到图像的主要隐特征，在图像分类时的分类精度只有11.13%,说明此时VAE还没有训练完成，不具备图像分类的能力。总结：适当的加入噪声可以加快VAE的训练速度，但是并不能提高分类的精度，但是一旦加入了过量的噪声会导致训练速度变慢甚至训练失败，分类精度也会大幅度下降。

4.2 PASCAL VOC数据集上实现图像分类

4.2.1 PASCAL VOC数据集上的训练过程

PASCAL VOC数据集是一款专门用作图像分类的数据集，在该数据集中一共定义了20种不同类型的图像，这些图像包括人类、动物、交通工具以及室内物品。这些图像种类的具体内容如下表所示。

表4-7 PASCAL VOC数据集上的图像类别

中文名称

英文名称

1

人类

Person

2

鸟类

Bird

3

猫

Cat

4

奶牛

Cow

5

狗

Dog

6

马

Horse

7

绵羊

Sheep

8

飞机

Aeroplane

9

单车

Bycicle

10

船只

Boat

11

公车

Bus

12

汽车

Car

13

摩托车

Motorbike

14

火车

Train

15

瓶子

Bottle

16

椅子

Chair

17

桌子

Table

18

盆栽

Plant

19

沙发

Sofa

20

电视

Tv

PASCAL VOC数据集一共分为四个部分，第一部分为Annotation，该部分主要包括了每一个图像的基本信息，主要为图像的位置、尺寸以及图像中的信息等。第二部分为ImageSets，主要内容与第一部分相似，只是存储的格式与第一部分不同。第三部分为JEPGImages部分，这里主要存储图像的具体数据。最后一部分为SegmentationClass部分，该部分主要涉及图像的语义分割。

图像的处理过程与上文中在MNIST数据集上完成训练的过程是一样的，由编码器对输入图像进行编码得到原始数据在隐空间上的特征，然后解码器对这些特征进行解码分析生成图像。我们对VAE进行迭代训练，当解码器可以顺利的完成对输入图像的识别分类并将其输出时，我们便可以认为训练完成，这时我们就可以输入一组图像来检验最终VAE的训练成果了。

4.2.2 VAE在PASCAL VOC数据集上的分类实现

由于PASCAL VOC数据集对于图像的各种类型已经定义好了，因此我们不用对图像的类型进行定义就可以进行图像的分类了。VAE在训练的过程中，解码器经过一次次的迭代已经可以识别不同图像对应的特征了，在这里我们一共对6种图片的类型进行分类。

我们对于每一种类型的图像分别输入8张，经过训练完成后的VAE得到最后的输出。然后我们分析隐空间中关于隐变量的数据来检验分类成果。我们知道隐变量的分布为正态分布，相关的值为均值和方差。在这里，我们对均值和方差做了规范化处理，根据映射函数，将其分别映射到区间[0,5]上。所得结果如下：

表4-8 PASCAL VOC数据集上训练VAE得到图像的隐空间特征

图像类别横坐标纵坐标Dog32.9Dog3.73.5Dog3.43.1Dog3.74.1Dog3.23.5Dog4.63.8Dog4.13.5Dog3.23Cat2.74.2Cat3.14.5Cat3.94.3Cat2.94.1Cat2.63.9Cat3.44.4Cat3.64.7Cat3.34.5Bus1.52.4Bus1.23.1Bus2.22.4Bus1.73.2Bus2.12.8Bus1.92.6Bus1.51.9Bus2.52.5表4-9 PASCAL VOC数据集上训练VAE得到图像的隐空间特征

图像类别

横坐标

纵坐标

Car

0.8

3.7

Car

1.2

3.3

Car

0.6

3.6

Car

0.5

3.7

Car

1.5

4.1

Car

0.6

3.1

Car

0.3

2.9

Car

0.2

3.8

Horse

2.9

1.1

Horse

2.8

0.9

Horse

3

1.7

Horse

3.1

1.6

Horse

3.2

2.2

Horse

3.3

1.9

Horse

3.2

2.1

Horse

3.4

1.8

TV

0.2

1.9

TV

0.4

1.8

TV

0.5

1.7

TV

0.6

1.6

TV

0.8

1.7

TV

0.3

2.2

TV

0.7

2.3

TV

1.3

2.1

参与训练的图像每一种类型有8张，一共有6种类型的图像。共计48张，分别展示如下：

图4-11 PASCAL VOC数据集上VAE实现图像分类的结果

通过以上的表项我们可以得到关于图像在隐空间上特征分布的散点图，通过该图我们可以分析各个图像的特征以及检验训练完成的VAE的图像分类结果。图中每一种类型的图像分别有8张，在图中分别代表一组颜色不同的散点。

图4-12 PASCAL VOC数据集图像在隐空间的特征分布

通过散点图我们可以看到，不同类型的图像分布集中的位置不同，每种图像对应8个散点，并且特征有相似的图像在分布上会有一定程度的交集。上图说明此时的VAE已经可以对狗、猫、公车、汽车、马以及电视机这6种不同的图像进行分类和识别了。

在这里，针对隐空间的特征分布图，我们继续采用KNN以及决策树算法来进行分类。分类的结果如下：

KNN算法：

表4-10 基于KNN算法的分类结果

从结果来看，图片分类的精度已经达到了70.0%的水平，是较好的分类结果。满足我们实验对图像分类的要求。

决策树算法：

首先，我们需要针对特征图中的不同图像的特征值设计决策树，保证能够锁定待分类图像的位置从而实现图像分类。

图4-13 决策树逻辑结构

根据以上决策树执行算法得到的结果如下：

表4-11 基于决策树算法的分类结果

结合两种算法的结果，我们可以得到两种算法分类图像的平均精度都在75.0%左右，两种算法的平均分类精度处于同一水平。此时，VAE已经完成了训练，能够识别不同类型图像的隐特征，实现图像的分类了。

4.3 CelebA数据集上实现图像识别分类

4.3.1 VAE在CelebA数据集上的训练过程

在这里图像的处理过程与上文中在MNIST数据集上完成训练的过程是一样的，由编码器对输入图像进行编码得到原始数据在隐空间上的特征，然后解码器对这些特征进行解码分析生成图像。我们对VAE进行迭代训练，当解码器可以顺利的完成对输入图像的识别分类并将其输出时，我们便可以认为训练完成，这时我们就可以输入一组图像来检验最终VAE的训练成果了。

在实验过程中，我们发现当迭代次数超过3000时，训练的结果便开始趋于一种稳定的状态，该状态下无论怎样增加迭代次数，生成的图像与原始图像相比相似度不再增加。这时训练过程已经完成。如下图所示，在迭代过程达到稳定后VAE的输出结果。

图4-14 CelebA数据集上VAE的训练结果

从训练结果来看，训练完成后的VAE的确对人脸图片具有一定的识别能力，但是其生成图像的效果却远远不如在MNIST数据集上的训练结果。这里我们可以分析得到变分自编码器具有的弊端，在面对MNIST这种图像数据特征比较单一的数据集时，VAE以其简短的构建过程完成了图像分类识别的工作。但是当数据集的原始数据图像逐渐趋于复杂时，VAE的弊端便暴露了出来。在这里我们将VAE在不同数据集的训练结果做了一个对比。

图4-15 VAE在不同数据集训练结果对比

我们可以明显地看到，在CelebA数据集下，VAE经过训练之后生成的图像与原始图像的相似度明显不如在MNIST数据集下的训练结果。这里我们可以对其进行分析。

首先，VAE对于原始数据的处理过程是通过变分的方法，其并没有像生成对抗网络算法那样的对抗式训练，逐步增强的过程。我们只是通过不停的数据进行迭代，一步一步训练解码器，但是并不能给解码器关于原始图像的定义。因此在面对复杂的输入图像时，VAE的训练完成后的输出图像还是比较模糊的。

其次，我们从上文中已经得到了关于MNIST数据集分布的特点，该数据集的图像数据特征分布较为集中，并且我们可以在隐空间中看出不同的类型的图像数据是有明显差别的。在CelebA数据集中，我们如果要实现对图像的分类，我们对于分类的类型定义是比较多元的，比如当我们需要对人脸数据进行性别上的分类时，我们需要对解码器定义其特征，当我们以人种为分类标识时，我们就需要改变其特征的定义。因此在面对较为复杂图像的分类时，VAE的效果就显得并不是那么的好了。

我们了解到VAE在面临较为复杂的图像时，训练的结果并没有像在MNIST数据集上的训练结果好。但是，我们依旧可以通过在隐空间中寻找变量的相互关系来实现图像的特征提取，有了这些我们定义的特征，我们不仅可以实现对图像的分类识别，也可以通过图像的特征来做一些特征转换。

我们首先完成对图像的识别分类，因为CelebA数据集是以人脸数据为基础的，因此我们这里以性别作为分类的一个标准。在上文中我们完成了对VAE的训练，在这里我们直接输入原始图像，得到输出结果即可。训练完成的VAE可以直接生成图像，我们可以将其与原始图像数据进行对比，如果生成的图像也满足原始图像在分类标准上的分类特征，我们就认为解码器已经拥有了分类的能力。最终输出的结果如下图所示。

图4-16 训练完成后的VAE在CelebA数据集上的训练结果

从结果来看，生成的数据依旧如上文所说的缺少相似度。我们可以看到，此时的VAE可能具有对图片中人物性别的识别能力。在上文关于VAE在MNIST数据集的实验中，我们知道在隐空间中的特征如果被分割了，那么VAE的解码器就会根据不同的隐特征生成不同的图像，也就是说此时的VAE是具备了图片分类的能力的。但是由于人脸的彩色图像数据在隐空间的维度比较高，我们不方便用图表的方式直接验证其结果。因此，我们需要对隐空间生成的数据向量做特殊的处理来进行验证。

在这里我们采用了隐空间中特征值变化的方法来验证训练完成后的VAE是否已经具备了识别的能力。验证的步骤如下：

更具数据集的图片，经历足够多次的迭代，完成对VAE的训练。

使用第一步得到的VAE对训练集计算出其在隐空间中关于性别属性向量。步骤为首先把所有男性人脸图片输入编码器，在隐空间中算出一个关于性别属性向量的平均值，再把所有女性人脸输入编码器，同样的方法算出一个平均值，两者的差就是男女属性的向量。

把原图输入编码器，把得到的向量减去（如果向相反的方向变化就加上）属性向量，再输入到解码器，可以得到一个输出的图像。

检验输出图像的特点，在隐空间中我们完成了对属性变量的转换，在输出的结果中理想的数据变化应该是男性图片与女性图片的特征出现了明显的翻转，如果输出图像结果符合以上的预期，那么就说明此时的VAE已经可以分辨出人脸图片中的性别特征，从而实现图像的分类了。

在实验中，我们采用以上的步骤得到的结果如下：

图4-17 通过训练VAE在CelebA数据集上实现性别特征翻转

通过实验结果我们可以发现，在输出的图像中，由于我们对编码器的输出结果做了一定的修改，输入的原始图像变量在隐空间内的特征被修改了，输出的图像与原始图像相比表现出了性别上了翻转。原来具有性别特征的脸部图像在经过VAE的生成处理之后，其表现出的性别特征是相反的，从这一点我们可以看出此时的VAE在训练完成后，已经具备了对原始人脸图像数据进行性别识别的能力。其可以自主的分析输入图像的特征，然后根据分类标准对图像进行识别分类。

4.3.2 VAE在CelebA数据集上的分类实现

通过以上的实验，我们已经完成了对VAE的训练过程，我们以人脸图像的性别为分类的基础，在这里我们使用KNN算法对输入的图片进行分类。得到的结果如下：

表4-12 CelebA数据集上的图像分类结果

通过图表我们可以得到，在面临图像数据复杂的数据集时，VAE的学习效果并没有在简单数据集上的好，但是图像的分类精度还是达到了70%，基本实现了对于不同图像的分类。

通过对输入图像在隐空间中特征向量的分析和处理，我们可以实现关于图像的很多种应用，也可以根据对特征的定义实现在不同的范畴内对图像进行分类。因此对于人脸图像的识别分类以及各种处理，变分自编码器的发展前景还是很广阔的。

第五章 VAE与其他算法的对比

5.1 自编码器

自编码器（AE）的工作原理与变分自编码器类似，自编码器也由编码器与解码器构成，并且与VAE一样，AE也有隐空间。输入的原始数据通过编码器编码，隐空间内得到关于原始数据的隐特征，然后解码器再根据这些隐特征分析原始数据结构尝试还原原始的数据，得到输出结果。总体来讲就是AE有一个隐藏层的神经网络，输入输出都是x，并且输入维度一定要比输出维度大，属于无监督学习。它先将输入压缩成潜在空间表征，然后通过这种表征来重构输出得到新的重构数据。

自编码器能从数据样本中进行无监督学习，这意味着可将这个算法应用到某个数据集中，不需要任何新的特征工程，只需要适当地训练数据。但是，自编码器在图像压缩方面表现得不好。由于在某个给定数据集上训练自编码器，因此它在处理与训练集相类似的数据时可达到合理的压缩结果，但是在压缩差异较大的其他图像时效果不佳。这里，像JPEG这样的压缩技术在通用图像压缩方面会表现得更好[10]。

图5-1 自编码器结构图

与VAE不同的是，自编码器缺少了变分的过程，在VAE的编码过程中，我们假设中间变量是满足正态分布的，并且在编码的过程中VAE会让编码器生成的中间变量尽量接近标准正态分布。自编码器就没有类似的过程，并且VAE的实现过程中还有避免计算的方差为0的情况，因为如果不去避免VAE就会逐渐退化为AE。

5.2 AE与VAE分类精度上的比较

同样在MNIST数据集下，我们采用AE算法对生成模型进行了训练，同样以手写数字作为识别的训练内容，自编码网络算法在训练的迭代过程中展现出了与VAE不同的特点。图中展示了AE算法最终完成训练的输出结果[11]。

图5-2 自编码器训练结果

完成训练之后，我们就可以开始实现图像的分类了，在这里我们继续采用KNN算法对测试图像进行分类。在上文中，我们已经使用VAE在MNIST数据集上完成了图像的识别分类，这里我们用训练好的AE完成分类结果如下：

表5-1 AE在MNIST数据集上的图像分类结果

从图中我们可以看出，通过AE实现的图像分类在每一个数字上分类精度都略低于使用VAE实现图像的分类。使用VAE实现图像分类的分类精度信息可以参考上一章，总体的分类精度也是有一定差别的，具体比较如下图所示：

表5-2 VAE与AE在分类精度上的对比

我们可以看到，相比于AE，VAE在图像分类方面具有一定的优势，并且VAE在训练过程中，一般情况下的训练速度也是比AE要快的。综上，VAE在图像分类识别方面具有一定的优势，有很好的发展前景。

第六章 实验总结与论文总结

6.1 论文过程与总结

6.1.1 理论过程

首先，论文从变分自编码器的基本理论出发，通过变分自编码器的基本理论，得到VAE可以应用于图像分类识别的可行性。通过第一章的概述，我们可以知道VAE从变分的角度出发，通过对输入图像数据的变换，得到满足图像特征的隐变量，然后通过解码器分析隐变量的特征，从而生成新的图片数据。

图像分类也就是在一个数据集中通过分析不同图像的特征分出属于不同类型的图像，在MNIST数据集中，数字的种类便可以成为一种分类的标准。由于MNIST数据集包含了不同的手写数字，每个手写数字的图像在数据分布上可能表现不同，但是这些图像的数据分布在某个隐空间内是具有相同的特点的，我们在原始的数据图像中不能直接得到关于该类图像的特点，因此我们需要变分自编码器解码得到隐变量，从而分析出属于同一类型图像的特点。

在第一章的分析中，我们可以知道，我们不能从VAE的编码器中直接得到生成的隐变量的具体数据，但是我们知道解码器可以分析隐变量的数据从而尝试去生成图像，因此解码器的结果从一定程度上可以反映出隐变量的具体分布。当我们去实现图像的分类时，我们只需要训练解码器，然后尝试让解码器分析不同数据图像的隐变量，如果解码器最终生成了与原始数据图像类型相同的结果，我们就可以认为解码器达到了训练的结果。

每当我们完成一次解码器的训练之后，我们可以继续输入下一个类型的手写数字，过程中可以记录不同数字的训练的迭代次数，然后直到所有类型的数字都训练完成之后，我们就可以调整输入了。在编码器一端输入不同类型的数字图像，然后观察解码器的输入结果，如果此时我们可以从结果中明显的分辨出不同的手写数字，此时便说明我们已经完成了对解码器的训练。其已经具备了在该数据集下实现图像分类的功能。

6.1.2 数据总结

实验过程的数据主要分为两种，一种为训练的迭代结果，一种为图像分类的输出结果。首先，我们对每个代表不同数字的图像样本作为输入，对解码器进行训练，得到关于每种类型图像大致的训练迭代次数，接下来我们可以计算得到VAE在MNIST数据集进行训练的平均迭代次数，我们可以得到在MNIST数据集上，VAE的训练迭代次数大约在1600到2000次。此时解码器已经可以通过分析输入图像的隐特征，解码生成对应的图像了。

图6-1 训练完成后的VAE输出结果

然后我们对编码器生成中间隐变量的特征的散点图进行了分析，从图4-8与4-9中我们可以看出，在MNIST数据集上不同的手写数字的特征关系，例如数字4与数字9这样的我们看来有相似特征的数字在VAE中也具有相似的特征，我们可以在4-9的散点图中看出，代表这两个数字的点是具有明显交集的，这也是解码器实现图像分类的一个关键因素，我们只要找到在隐空间中关于不同图像的特征分布，就可以通过解码器实现不同图像的分类了。

从4-9的散点图中我们分析出了关于MNIST数据集图像的特点，我们可以看到所有代表图像的点是分布在一个固定的区域内的，整体来看其分布是较为集中的，这也和数据集中的图像数据有关系，书写数字图像在一定程度上是有共同特征的，在一定程度上也有差别，这一点我们可以从图4-9的分布中看出。

然后，我们根据我们训练完成的VAE结合KNN以及决策树的分类算法实现了对不同类型数字的图像的分类。并且分析了不同数字分类精度不同的原因，以及两种分类算法的差别。

接下来，我们分析了实验中的数据因素对于VAE训练过程中迭代次数以及分类精度的影响，这里主要分析了加入噪音的强度对以上两者的影响。

在噪音强度对迭代速度的分析中，我们得到如图4-15的结果。我们分别在VAE的训练过程中选择了不加入噪音，加入适量噪音以及加入过量噪音三个选项。

从结果分析，我们可以得到，适量的噪音可以稍微破坏生成隐变量的结构，对于生成器来说生成难度上升，可以加快生成器的训练速度，此时VAE的训练速度会比不加入任何噪声的情况要快。当我们加入了过量的噪声时，生成器在不停的迭代过程中一直没有完成训练，此时由于过量的噪声严重破坏了隐变量的结构，生成器没有办法还原生成原始数据。

表6-1 不同噪声下VAE的训练结果

在噪音强度对分类精度的影响中，我们得到了结论：当加入一定噪音时，只会影响VAE的训练速度，而不会影响分类精度。但是一旦加入了过量的噪音，VAE就会失去对图像的识别能力，从而分类精度降低。

表6-2 噪音对分类精度的影响

接下来，我们在其他的两个数据集上进行了同样的实验，分析了实验结果，虽然VAE都完成了分类的功能，但是在MNIST数据集上表现的更好，这也和该数据集的图像数据分布单一有一定关系。

最后，我们通过实验以及资料对变分自编码器和其他算法做了对比，这里我们用来对比的算法为自编码网络算法，也就是AE。通过最终的结果我们可以发现，在同一数据集和同一分类算法的情况下，变分自编码器的图像分类精度是高于自编码器的，因此VAE在图像分类方面有很好的应用前景。

6.2 变分自编码器的应用前景

首先，针对本次实验，我们完成了训练VAE达到识别分类手写数字的目的，通过变分自编码器对数字的识别分类我们可以将其应用与许多方面。我们可以将其应用于车牌识别，通过解码器识别生成不同类型的数字，直接通过处理就可以作为数据存储，节约了人工成本，VAE也可以应用于试卷以及各种表格等手写数据的机器识别，可以通过表格等内容直接分析得到其中的数字数据以及字体，这种应用可以直接进行各种数据的机器统计，提高了统计效率。

当前社会的各个领域数据都处于一种极速膨胀的环境之中，通过变分自编码器的图像分类可以利用机器快速的实现对各个数字的统计。对于时代的应用前景而言，VAE的图像分类是非常契合时代步伐的。

VAE除了可以实现对于手写数字的分类，我们也可以对其加以改进，令其具有更加强大的分类功能完成其他各种图像的分类。例如对于人脸图像的分类可以令警察快速的锁定失踪的人，对于家具工具的分类可以应用于家庭机器人的程序中等等。

总体来说，变分自编码器的应用领域是很广的，国内外学者将 VAE 应用到了各个方面，比如，图像识别，图像分类，图文配字，图像生成以及不在图像处理领域的机械故障评估，战斗形势判断，异常检测，轨迹规划等。而且对于变分自编码器的改进方案也在层出不穷，有基于条件的变分自编码器（CVAE），变分公平自编码器（VFAE），以及变分自编码器与生成对抗网络相结合融合网络（VAE-GAN）。变分自编码技术正在快速的发展，未来各个领域可能会看到跟多关于VAE技术的应用[12]。

参考文献

[1]Kingma D P, Welling M. Auto-Encoding Variational Bayes[J]. international conference on learning representations, 2014.

[2]范纯龙等. 基于变分自编码器的交互式旋律生成方法[J]. 《计算机应用研究》, 2020.4.21.

[3]徐朋磊等. 一种基于变分自编码器的高光谱图像分类方法[J]. 《现代测绘》，2020年1月. 第43卷第一期.

[4]杨任农等. 变分自编码器结合聚类算法在空战态势评估问题上的应用[J]. 《国防科技大学学报》，2019年4月.

[5]Blei D M, Kucukelbir A, Mcauliffe J, et al. Variational Inference: A Review for Statisticians[J]. Journal of the American Statistical Association, 2017,112(518): 859-877.

[6]翟正利,梁振明,周炜等. 变分自编码器模型综述[J]《计算机工程与应用》 2019, 55(3): 1-9.

[7]郑欣悦,黄永辉. 基于VAE和注意力机制的小样本图像分类方法《计算机应用与软件》， 2019年第10期.

[8]张明军等. 一种基于机器学习的车牌识别系统的设计[J]. 《计算机技术与发展》，2020年3月.

[9]Pu Y, Gan Z, Henao R, et al. Variational Autoencoder for Deep Learning of Images, Labels and Captions[J]. arXiv: Machine Learning, 2016.

[10]Gulrajani I, Ahmed F, Arjovsky M, et al. Improved Training of Wasserstein GANs[J]. arXiv: Learning, 2017.

[11]spectre. GANs入门系列之用GAN生成MNIST数据集之pytorch实现2018-6 HYPERLINK "https://blog.csdn.net/weixin\_41278720/article/details/80861284" https://blog.csdn.net/weixin\_41278720/article/details/80861284.

[12]张蕾,钱峰,赵姝等. 利用变分自编码器进行网络表示学习.《计算机科学与探索》 2019年第10期.

致 谢

感谢大学四年以来对我的培养，感谢我的导师对本论文从选题、构思、资料收集到最后定稿的各个环节给予细心的指引和教导。在毕设论文的工作开展过程中，我的导师对我的思路给予了非常好的指导。在开题时，由于我的理解疏忽，导致我对变分自编码器的原理有一定程度上的误解，这时我的导师及时给我指导，并且帮我寻找了可用的资源，在中期完成实验过程时，由于缺乏数据我无从下手，这时我的导师帮我找到了可用的数据集。最重要的是，实验过程的思路以及流程都是在我导师的指导下完成的，导师的指导是我完成此次毕业设计最重要的一个因素。

同时，我还要感谢大学以来各个老师对我的培养与教导，在你们的教育下，我完成了大学基础课程和专业课程的学习，为我的毕业设计的完成打下了基础。你们的教导是我大学四年最为宝贵的财富。

最后，我还要感谢我的同学在论文写作过程当中的给予我很大的鼓励和帮助，在与他们的交流中拓展了我的视野和思路，使我能更好的完成这次毕业论文。

毕业设计小结

本次毕业设计的主要内容为变分自编码在图像分类中的应用，变分自编码就是通过编码器编码和解码器解码完成对一种数据的重构。本次毕设主要由理论过程和实验过程组成。

理论过程中，我从变分自编码器的原理出发，引用了论文中的成果，描述了变分自编码器在实现过程中的每一步流程以及这些流程中需要的数据。与此同时，我对每一步中需要的数据做出了解释，对数据来源以及数据含义给出了阐述。最后，我从变分自编码器的特点出发，阐述了变分自编码器在各行各业的广泛应用，其中以变分自编码器在图像分类中的应用为主要内容，引用论文中的成果，展示了应用的具体内容和数据。

实验过程中，我以最基础的手写体数据集开始，首先完成变分自编码器的训练过程，然后根据训练结果我在变分自编码器的隐空间中提取到了原始数据的隐特征，然后我们使用KNN算法以及决策树算法，对隐特征进行分类实验，最终得到了各个类型图像的分类精度。其次，我在同样的数据集上对自编码器算法做了同样的实验，通过实验结果与变分自编码器进行对比，得到两种算法的特点以及区别，并且分析了变分自编码器网络在图像分类方面的优点。最后，我在其他的数据集上使用了变分自编码算法，得到了该算法在不同数据集的特点，同时也分析出了不同数据集的特点以及这些数据集对变分自编码器的影响。

附 录

实验过程用到的代码

1.数据集代入与处理：

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

import gzip

import os

import numpy

from scipy import ndimage

from six.moves import urllib

import tensorflow as tf

SOURCE\_URL = 'http://yann.lecun.com/exdb/mnist/'

DATA\_DIRECTORY = "data"

# Params for MNIST

IMAGE\_SIZE = 28

NUM\_CHANNELS = 1

PIXEL\_DEPTH = 255

NUM\_LABELS = 10

VALIDATION\_SIZE = 5000 # Size of the validation set.

# Download MNIST data

def maybe\_download(filename):

"""Download the data from Yann's website, unless it's already here."""

if not tf.gfile.Exists(DATA\_DIRECTORY):

tf.gfile.MakeDirs(DATA\_DIRECTORY)

filepath = os.path.join(DATA\_DIRECTORY, filename)

if not tf.gfile.Exists(filepath):

filepath, \_ = urllib.request.urlretrieve(SOURCE\_URL + filename, filepath)

with tf.gfile.GFile(filepath) as f:

size = f.size()

print('Successfully downloaded', filename, size, 'bytes.')

return filepath

# Extract the images

defextract\_data(filename,num\_images,norm\_shift=False, norm\_scale=True):

"""Extract the images into a 4D tensor [image index, y, x, channels].

Values are rescaled from [0, 255] down to [-0.5, 0.5].

"""

print('Extracting', filename)

with gzip.open(filename) as bytestream:

bytestream.read(16)

buf = bytestream.read(IMAGE\_SIZE \* IMAGE\_SIZE \* num\_images \* NUM\_CHANNELS)

data = numpy.frombuffer(buf, dtype=numpy.uint8).astype(numpy.float32)

if norm\_shift:

data = data - (PIXEL\_DEPTH / 2.0)

if norm\_scale:

data = data / PIXEL\_DEPTH

data = data.reshape(num\_images, IMAGE\_SIZE, IMAGE\_SIZE, NUM\_CHANNELS)

data = numpy.reshape(data, [num\_images, -1])

return data

# Extract the labels

def extract\_labels(filename, num\_images):

"""Extract the labels into a vector of int64 label IDs."""

print('Extracting', filename)

with gzip.open(filename) as bytestream:

bytestream.read(8)

buf = bytestream.read(1 \* num\_images)

labels = numpy.frombuffer(buf, dtype=numpy.uint8).astype(numpy.int64)

num\_labels\_data = len(labels)

one\_hot\_encoding = numpy.zeros((num\_labels\_data,NUM\_LABELS))

one\_hot\_encoding[numpy.arange(num\_labels\_data),labels] = 1

one\_hot\_encoding = numpy.reshape(one\_hot\_encoding, [-1, NUM\_LABELS])

return one\_hot\_encoding

# Augment training data

def expend\_training\_data(images, labels):

expanded\_images = []

expanded\_labels = []

j = 0 # counter

for x, y in zip(images, labels):

j = j+1

if j%100==0:

print ('expanding data : %03d / %03d' % (j,numpy.size(images,0)))

# register original data

expanded\_images.append(x)

expanded\_labels.append(y)

# get a value for the background

# zero is the expected value, but median() is used to estimate background's value

bg\_value = numpy.median(x) # this is regarded as background's value

image = numpy.reshape(x, (-1, 28))

for i in range(4):

# rotate the image with random degree

angle = numpy.random.randint(-15,15,1)

new\_img = ndimage.rotate(image,angle,reshape=False, cval=bg\_value)

# shift the image with random distance

shift = numpy.random.randint(-2, 2, 2)

new\_img\_ = ndimage.shift(new\_img,shift, cval=bg\_value)

# register new training data

expanded\_images.append(numpy.reshape(new\_img\_, 784))

expanded\_labels.append(y)

# images and labels are concatenated for random-shuffle at each epoch

# notice that pair of image and label should not be broken

expanded\_train\_total\_data = numpy.concatenate((expanded\_images, expanded\_labels), axis=1)

numpy.random.shuffle(expanded\_train\_total\_data)

return expanded\_train\_total\_data

# Prepare MNISt data

def prepare\_MNIST\_data(use\_norm\_shift=False, use\_norm\_scale=True, use\_data\_augmentation=False):

# Get the data.

train\_data\_filename = maybe\_download('train-images-idx3-ubyte.gz')

train\_labels\_filename = maybe\_download('train-labels-idx1-ubyte.gz')

test\_data\_filename = maybe\_download('t10k-images-idx3-ubyte.gz')

test\_labels\_filename = maybe\_download('t10k-labels-idx1-ubyte.gz')

# Extract it into numpy arrays.

train\_data = extract\_data(train\_data\_filename, 60000, use\_norm\_shift, use\_norm\_scale)

train\_labels = extract\_labels(train\_labels\_filename, 60000)

test\_data = extract\_data(test\_data\_filename, 10000, use\_norm\_shift, use\_norm\_scale)

test\_labels = extract\_labels(test\_labels\_filename, 10000)

# Generate a validation set.

validation\_data = train\_data[:VALIDATION\_SIZE, :]

validation\_labels = train\_labels[:VALIDATION\_SIZE,:]

train\_data = train\_data[VALIDATION\_SIZE:, :]

train\_labels = train\_labels[VALIDATION\_SIZE:,:]

# Concatenate train\_data & train\_labels for random shuffle

if use\_data\_augmentation:

train\_total\_data = expend\_training\_data(train\_data, train\_labels)

else:

train\_total\_data = numpy.concatenate((train\_data, train\_labels), axis=1)

train\_size = train\_total\_data.shape[0]

return train\_total\_data, train\_size, validation\_data, validation\_labels, test\_data, test\_labels

2.VAE训练过程：

import tensorflow as tf

# Gaussian MLP as encoder

def gaussian\_MLP\_encoder(x, n\_hidden, n\_output, keep\_prob):

with tf.variable\_scope("gaussian\_MLP\_encoder"):

# initializers

w\_init = tf.contrib.layers.variance\_scaling\_initializer()

b\_init = tf.constant\_initializer(0.)

# 1st hidden layer

w0 = tf.get\_variable('w0', [x.get\_shape()[1], n\_hidden], initializer=w\_init)

b0 = tf.get\_variable('b0', [n\_hidden], initializer=b\_init)

h0 = tf.matmul(x, w0) + b0

h0 = tf.nn.elu(h0)

h0 = tf.nn.dropout(h0, keep\_prob)

# 2nd hidden layer

w1 = tf.get\_variable('w1', [h0.get\_shape()[1], n\_hidden], initializer=w\_init)

b1 = tf.get\_variable('b1', [n\_hidden], initializer=b\_init)

h1 = tf.matmul(h0, w1) + b1

h1 = tf.nn.tanh(h1)

h1 = tf.nn.dropout(h1, keep\_prob)

# output layer

# borrowed from https: // github.com / altosaar / vae / blob / master / vae.py

wo = tf.get\_variable('wo', [h1.get\_shape()[1], n\_output \* 2], initializer=w\_init)

bo = tf.get\_variable('bo', [n\_output \* 2], initializer=b\_init)

gaussian\_params = tf.matmul(h1, wo) + bo

# The mean parameter is unconstrained

mean = gaussian\_params[:, :n\_output]

# The standard deviation must be positive. Parametrize with a softplus and

# add a small epsilon for numerical stability

stddev = 1e-6 + tf.nn.softplus(gaussian\_params[:, n\_output:])

return mean, stddev

# Bernoulli MLP as decoder

def bernoulli\_MLP\_decoder(z, n\_hidden, n\_output, keep\_prob, reuse=False):

with tf.variable\_scope("bernoulli\_MLP\_decoder", reuse=reuse):

# initializers

w\_init = tf.contrib.layers.variance\_scaling\_initializer()

b\_init = tf.constant\_initializer(0.)

# 1st hidden layer

w0 = tf.get\_variable('w0', [z.get\_shape()[1], n\_hidden], initializer=w\_init)

b0 = tf.get\_variable('b0', [n\_hidden], initializer=b\_init)

h0 = tf.matmul(z, w0) + b0

h0 = tf.nn.tanh(h0)

h0 = tf.nn.dropout(h0, keep\_prob)

# 2nd hidden layer

w1 = tf.get\_variable('w1', [h0.get\_shape()[1], n\_hidden], initializer=w\_init)

b1 = tf.get\_variable('b1', [n\_hidden], initializer=b\_init)

h1 = tf.matmul(h0, w1) + b1

h1 = tf.nn.elu(h1)

h1 = tf.nn.dropout(h1, keep\_prob)

# output layer-mean

wo = tf.get\_variable('wo', [h1.get\_shape()[1], n\_output], initializer=w\_init)

bo = tf.get\_variable('bo', [n\_output], initializer=b\_init)

y = tf.sigmoid(tf.matmul(h1, wo) + bo)

return y

# Gateway

def autoencoder(x\_hat, x, dim\_img, dim\_z, n\_hidden, keep\_prob):

# encoding

mu, sigma = gaussian\_MLP\_encoder(x\_hat, n\_hidden, dim\_z, keep\_prob)

# sampling by re-parameterization technique

z = mu + sigma \* tf.random\_normal(tf.shape(mu), 0, 1, dtype=tf.float32)

# decoding

y = bernoulli\_MLP\_decoder(z, n\_hidden, dim\_img, keep\_prob)

y = tf.clip\_by\_value(y, 1e-8, 1 - 1e-8)

# loss

marginal\_likelihood = tf.reduce\_sum(x \* tf.log(y) + (1 - x) \* tf.log(1 - y), 1)

KL\_divergence = 0.5 \* tf.reduce\_sum(tf.square(mu) + tf.square(sigma) - tf.log(1e-8 + tf.square(sigma)) - 1, 1)

marginal\_likelihood = tf.reduce\_mean(marginal\_likelihood)

KL\_divergence = tf.reduce\_mean(KL\_divergence)

ELBO = marginal\_likelihood - KL\_divergence

loss = -ELBO

return y, z, loss, -marginal\_likelihood, KL\_divergence

def decoder(z, dim\_img, n\_hidden):

y = bernoulli\_MLP\_decoder(z, n\_hidden, dim\_img, 1.0, reuse=True)

return y

**本科毕业设计（论文）**

PAGE i

PAGE 9

INCLUDEPICTURE "http://202.117.83.12/view/vi/pic/xb.gif" \\* MERGEFORMATINET

INCLUDEPICTURE "http://202.117.83.12/view/vi/pic/bzz.gif" \\* MERGEFORMATINET