单位代码：_____ 密　级：公开

学　　号：XXX 分类号：_____

合肥工业大学

Hefei University of Technology

硕士学位论文

MASTER'S DISSERTATION

（学术硕士）

论文题目：XXX

XXX

学科专业：信号与信息处理

学生姓名：XXX

导师姓名：XXX　教授

完成时间：2020 年 4 月

合　肥　工　业　大　学

学历硕士学位论文

XXX
XXX

作者姓名：XXX

指导教师：XXX　教授

学科专业：信号与信息处理

研究方向：XXX

2020 年 4 月

A Dissertation Submitted for the Degree of Bachelor

XXX

By

XXX

Hefei University of Technology

Hefei, Anhui, P.R.China

4 Month, 2020 Year

毕业设计（论文）独创性声明

本人郑重声明：所呈交的毕业设计（论文）是本人在指导教师指导下进行独立研究工作所取得的成果。据我所知，除了文中特别加以标注和致谢的内容外，设计（论文）中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 合肥工业大学 或其他教育机构的学位或证书而使用过的材料。对本文成果做出贡献的个人和集体，本人已在设计（论文）中作了明确的说明，并表示谢意。

毕业设计（论文）中表达的观点纯属作者本人观点，与合肥工业大学无关。

毕业设计（论文）作者签名：　　签名日期：　　年　　月　　日

毕业设计（论文）版权使用授权书

本学位论文作者完全了解 合肥工业大学 有关保留、使用毕业设计（论文）的规定，即：除保密期内的涉密设计（论文）外，学校有权保存并向国家有关部门或机构送交设计（论文）的复印件和电子光盘，允许设计（论文）被查阅或借阅。本人授权 合肥工业大学 可以将本毕业设计（论文）的全部或部分内容编入有关数据库，允许采用影印、缩印或扫描等复制手段保存、汇编毕业设计（论文）。

（保密的毕业设计（论文）在解密后适用本授权书）

学位论文作者签名：　　指导教师签名：

签名日期：　　年　　月　　日　　签名日期：　　年　　月　　日

致谢

这部分是致谢

作者：XXX

2020 年 4 月 6 日

摘　要

这部分是摘要

关键词：

ABSTRACT

This is English abstract.

KEYWORDS:

目　录

插图清单

表格清单

第一章 这是第一章

1.1 这是第一节

这部分写你的文章，section 为一级标题，subsection 为二级标题，subsubsection 为三级标题，依次类推

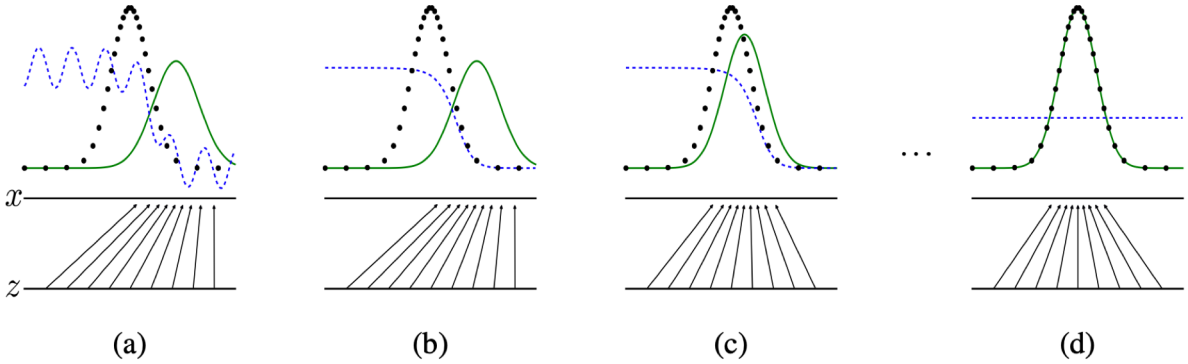

图 1.1 测试图

Fig 1.1 test figure

这是你的第一张图片 [1]

$$\mathcal{L}_D == \mathbb{E}_{x\sim p_data(x)}[\log\left(D(x)\right)] + \mathbb{E}_{z\sim p_z(z)}[\log\ \left(1-D(G(z))\right)] \tag{1.1}$$

这是你的第一个公式

$$\begin{aligned}
C(G) &= \max_D V(D,G) \\
&= \max_D \int_x p_{data}(x)\log\left(D(x)\right) + p_g(x)\log\left(1-D(x)\right)dx \\
&= \int_x p_{data}(x)\log D_G^*(x) + p_g(x)\log\left(1-D_G^*(x)\right)dx \\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right) + p_g(x)\log\left(1-\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right)dx \\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right) + p_g(x)\log\left(\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right)dx \\
&= \int_x p_{data}(x)\log\left(\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right) + p_g(x)\log\left(\frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right)dx - \log 4 \\
&= KL[p_{data}(x)\|\frac{p_{data}(x)+p_g(x)}{2}] + KL[p_g(x)\|\frac{p_{data}(x)+p_g(x)}{2}] - \log 4
\end{aligned} \tag{1.2}$$

这是你的第一个长公式

这是你的第一张表格

表 1.1　生成对抗网络生成 $28 \times 28 \times 1$ 的黑白图像的网络详细设计

Tab 1.1　Detailed Network Design for Generating Adversarial Networks to Generate $28 \times 28 \times 1$ Black and White Images

Generator		**Discriminator**	
Layer	Shape	Layer	Shape
input layer	(B, 100)	input layer	(B, 28, 28, 1)
MLP-(100, 3136), BN, ReLU	(B, 3136)	CONV-(N64, K3, S2, P1), BN, LReLU	(B, 14, 14, 64)
Reshape	(B, 7, 7, 64)	CONV-(N64, K3, S2, P1), BN, LReLU	(B, 7, 7, 64)
DeCONV-(N64, K3, S2, P1), BN, ReLU	(B, 14, 14, 64)	CONV-(N128, K3, S2, P1), BN, LReLU	(B, 4, 4, 128)
CONV-(N128, K3, S1, P1), BN, ReLU	(B, 14, 14, 128)	CONV-(N128, K3, S1, P1), BN, LReLU	(B, 4, 4, 128)
DeCONV-(N64, K3, S2, P1), BN, ReLU	(B, 28, 28, 128)	Flatten	(B, 2048)
CONV-(N64, K3, S1, P1), BN, ReLU	(B, 28, 28, 64)	MLP-(2048, 1), BN, Sigmoid	(B, 1)
CONV-(N64, K3, S1, P1), BN, ReLU	(B, 28, 28, 64)		
CONV-(N1, K3, S1, P1), BN, ReLU	(B, 28, 28, 1)		

```python
def adaptive_instance_layer_norm(x, gamma, beta, smoothing=True, scope='instance_layer_norm'):
    with tf.variable_scope(scope):
        ch = x.shape[-1]
        eps = 1e-5
        # 计算 Instance mean, sigma and ins
        ins_mean, ins_sigma = tf.nn.moments(x, axes=[1, 2], keep_dims=True)
        x_ins = (x - ins_mean) / (tf.sqrt(ins_sigma + eps))

        # 计算 Layer mean, sigma and ln
        ln_mean, ln_sigma = tf.nn.moments(x, axes=[1, 2, 3], keep_dims=True)
        x_ln = (x - ln_mean) / (tf.sqrt(ln_sigma + eps))

        # 给定 rho 的范围, smoothing 控制 rho 的弹性范围
        if smoothing:
            rho = tf.get_variable("rho", [ch], initializer=tf.constant_initializer(0.9),
                                  constraint=lambda x: tf.clip_by_value(x,
                                  clip_value_min=0.0, clip_value_max=0.9))
        else:
            rho = tf.get_variable("rho", [ch], initializer=tf.constant_initializer(1.0),
                                  constraint=lambda x: tf.clip_by_value(x,
                                  clip_value_min=0.0, clip_value_max=1.0))

        # rho = tf.clip_by_value(rho - tf.constant(0.1), 0.0, 1.0)

        x_hat = rho * x_ins + (1 - rho) * x_ln

        x_hat = x_hat * gamma + beta

        return x_hat
```

参考文献

[1] AVRIEL M. Nonlinear programming: analysis and methods[M]. [S.l.] : Courier Corporation, 2003.

[2] HINTON G E. A practical guide to training restricted Boltzmann machines[G] // Neural networks: Tricks of the trade. [S.l.] : Springer, 2012 : 599 – 619.

[3] BOURLARD H, KAMP Y. Auto-association by multilayer perceptrons and singular value decomposition[J]. Biological cybernetics, 1988, 59(4-5) : 291 – 294.