中图分类号: 学校代码: 10055
UDC: 密级: 公开

南開大學

博 士 学 位 论 文

视觉感知任务中神经网络损失函数设计研究

On the Design of Neural Network Loss Functions for Visual Perception

论文作者 赵凯 指导教师 程明明 教授
申请学位 工学博士 培养单位 计算机学院
学科专业 计算机科学与技术 研究方向 计算机视觉
答辩委员会主席 胡清华 评 阅 人 匿名评阅

南开大学研究生院

二〇二〇年六月

摘要

卷积神经网络作为计算机视觉领域的一个核心工具，已经被广泛运用于图像分类、语义分割、人脸识别、目标识别与跟踪、显著性物体检测等很多计算机视觉任务中。神经网络的训练高度依赖基于梯度的优化算法，在优化过程中，输出端在前向传播时计算神经网络预测的误差；在反向传播时计算误差对网络预测的梯度，然后通过反向传导算法将梯度逐层反传，并利用链式法则计算误差对各层参数的导数。最后用梯度下降法等算法将网络参数往误差下降的方向迭代更新，直至收敛。由于损失函数在神经网络优化中直接产生梯度，是梯度的来源，因此在网络的优化中处于十分重要的地位。

基于交叉熵的多分类损失函数最开始主要用于图像分类任务中，后来逐渐被应用到多个任务中用作损失函数，因为很多任务例如人脸识别，显著性检测，语义分割等都可以视为特殊的分类任务。作为分类任务的损失函数，交叉熵损失并不考虑考虑样本标注的置信度，也没有考虑多个分类类别之间的语义关联性等因素。同时应用到其它非分类任务中时，交叉熵损失只寻求模型在训练集上的分类误差最小化，并没有针对性优化特定任务的评价指标。例如人脸识别旨在于最大化不同类样本特征之间的距离的同时最小化同类样本的距离；而检测结果的对比度和完整性对显著性检测十分重要，最小化交叉熵损失并不能针对性地提高这些和任务紧密相关的评价指标。因此，针对不同任务对性能指标的不同需求来个性化地设计神经网络的优化目标对模型性能十分关键。

本文的研究内容以神经网络的优化目标为中心展开，以计算机视觉任务为落脚点，针对几个典型的计算机视觉任务对模型表现的不同期望，设计了任务相关的损失函数。针对显著性检测任务，本文针对当前主流显著性检测方法中物体边界模糊现象，设计了基于 F-measure的损失函数。该损失函数在整个可行域具有显著梯度，可以迫使网络输出两极分化的预测值，显著增强显著性检测结果的对比度，有效解决物体边界模糊问题。针对人脸识别任务，本文提出了“互斥正则”损失函数，该损失项可以显式地增大不同类别间特征的离散程度，从而增强模型对人脸照片的辨识度。针对卷积神经网络压缩任务，本文提出了一种自适应的 L1 稀疏正则损失，该正则损失函数可以在训练过程中自动控制模

型稀疏性，以便在压缩剪枝阶段最小化对模型性能的影响。

关键词： 卷积神经网络，计算机视觉，优化目标，损失函数，正则函数。

Abstract

As one of the fundamental tools in computer vision, Convolutional Neural Networks (CNNs) have been applied to various vision tasks including image classification, semantic segmentation, face recognition and object tracking. The training and optimization of CNNs highly depend on the gradient-based optimization algorithms *e.g.* stochastic gradient descent (SGD). During optimization, the loss function computes the costs of the model prediction w.r.t ground-truth (or label), and then pass the error back to intermediate layers using the back-propagation algorithm. Then the gradients of loss function w.r.t each model parameter is determined according to the chain-ruler. After that, SGD updates model parameters by moving them towards the direction where the loss function decreases. Since the loss function is the source of gradient and an indicator of model convergence, it plays an core role during optimization.

Cross-entropy loss is furstly applied to image classification task, and then quickly adopted to other tasks because many other tasks such as face recognition, saliency detection and semantic segmentation can be regarded as special cases of image classification. As a loss function for image classification, the cross-entropy does not take label confidences and the semantic relations between labels into consideration. When being applied to other tasks, cross-entropy loss only seeks the minimalization of classification error, while neglecting the evaluation metrics in a specific task. Take face recognition as an example, face recognition requires highly discriminative features rather than classification accuracy. While the contrast between foreground and background is critical to saliency detection which cannot be enhanced via cross-entropy loss. Therefore, designing loss functions specifically for tasks is critical to model performance.

This dissertation studies the research of loss functions in CNNs, the authors proposed 3 different loss functions for 3 different tasks of computer vision. Specifically, for saliency object detection, the authors propose to directly optimize the evaluation metric during training. By relaxing the F-measure, the authors derive soft version of F-measure that is differentiable w.r.t the model prediction, which can be directly ap-

pended to the back to CNN as the loss function. For face recognition, authors propose the ”exclusive regularization” that explicitly enlarges the distance between features of different classes. For model compression and pruning, the authors propose a adaptive sparse regularization loss that can automatically adjust the coefficient of sparse regularization loss, consequently minimizes the influence to the model performance during pruning.

Key Words: Convolutional Neural Networks, Computer Vision, Optimization target, Loss function, Regularization function.

目录

第一章 绪论

本章主要介绍相关的研究背景和本文的主要研究目标和贡献。

第一节 论文的背景

1.1.1 研究背景

视觉是人类从环境获取信息的主要来源，我们每天从睁开眼就从周围环境接收大量视觉信息。我们的视觉系统会分析和处理这些信息，根据这些视觉信息和环境进行交互，做出决策。图像和图形是日常生活中我们交流信息、表达情感最有效的方式之一。在朋友圈和推特等社交媒体上，图像和视频成为更受人们欢迎的媒介；在聊天应用中，人们更愿意用表情和符号来表达自己的喜怒哀乐。如何让计算机帮助和辅助人类进行视觉感知，通过采集图像和视频来认识和分析周围的环境，帮助人类进行决策，已经成为消防安防、智慧医疗、自动驾驶和智慧城市等领域的研究热点，并有着广泛的运用，有着巨大的经济和社会价值。

人类能够快速的从图像中理解并抽象出关键信息，并作出对应决策，但是对图像中具体某个像素值并不敏感。而图像在计算机中以像素值的形式存储和处理。计算机通常很难理解图像中的物体等概念，但是对像素值很敏感。通常计算机只能处理非常底层的图像数据，例如进行滤波，色彩增强，去噪等操作。如何从数值化后的图像数据中挖掘和学习更高级别的抽象的概念是计算机视觉领域的一个核心问题。最早人们大多手工设计特征提取算子从图像中提取梯度和纹理特征，然后进行特征聚合等操作，最后用简单的学习模型进行决策。受限于手工特征的表达能力，这类方法通常无法应对图像中物体的形变和复杂的环境，只能处理非常简单的场景。

随着计算机硬件，特别是图形处理器（GPU）性能的提升，使得训练超大规模的神经网络模型成为可能。 2012年 Krizhevsky 等人 [62] 使用一个七层卷积神经网络模型在 ImageNet [112] 大规模图像分类任务上取的了巨大突破。自此，深度卷积神经网络，又称深度学习，称为计算机视觉领域的研究热门，被广泛运用到图像分类 [39, 62]、人脸识别 [123, 114, 113]、目标检测 [28, 109]等众多计

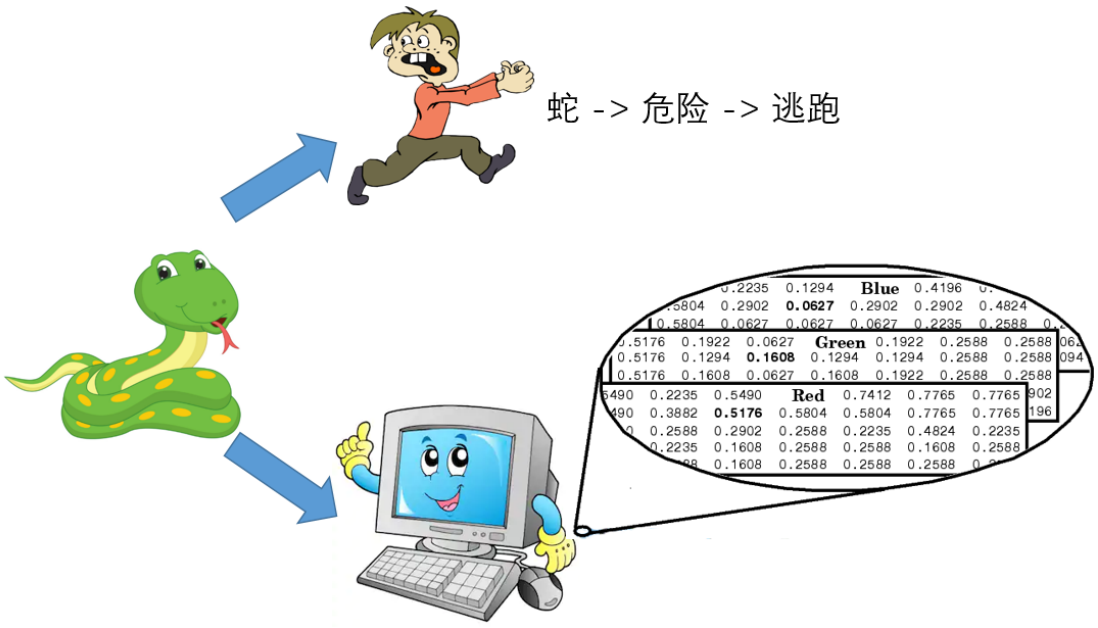


图 1.1 人类的视觉和认知系统能够迅速从视觉信号中抽象出概念，并作出决策。但是，对计算机而言一幅图片只是一个二维矩阵。让计算机能够跟人类一样从理解图像数据，具有十分重大的意义。

算机视觉的主要任务中，取得了远超传统方法的性能。

深度卷积神经网络的训练依赖于大量已标注的数据，通过在训练集上降低模型的预测误差，使模型在未知的测试数据上取得良好效果。这一过程和复习备考类似：在大量的训练集（平时作业，模拟考试）中反复训练模型（学生），降低在训练集上的误差（提高模拟考试的得分），以期望模型在测试集（最终测试）上取得良好成绩。这么做的理论支撑在于大数定律和独立同分布假设：我们假设训练样本和测试样本都是从同一个分布中采样出来的，因此他们是独立同分布的。因此如果模型在训练集上取得良好成绩，那么从统计上讲，模型也能在测试集上表现得不错。同时，为了保证测试数据和训练数据的同源，就必须保证训练数据是从真实世界中大量采样得到的，因此深度学习技术需要大量已标注的训练数据。我们用**损失函数**，来定量的描述神经网络的预测误差，通过在训练集上通过更新模型参数来不断降低预测误差，提升模型在训练数据上的预测准确度。同时我们往往对网络参数有一些先验假设，例如网络参数服从正态分布，网络参数在一个很小的区间不会太大等等。因此在损失函数之外我们会额外加上一项**正则化项**来控制网络参数。最终整个模型的**优化目标**由**损失**

函数和**正则化项**两项组成。

以分类任务为例，假设样本 x_i 的类别标签为 y_i，模型的输出为 $h(x_i) \in \mathbb{R}^C$，其中 C 为类别总数。那么目标函数 J_θ 为：

$$J_\theta = \sum_i -\log h(x_i)_{y_i} + \sum \frac{1}{2}||W||_2^2. \tag{1.1}$$

在 公式(1.1)的目标函数中，第一项与输入数据 x_i 有关，是损失函数；第二项与数据无关，只与网络参数有关，称为正则化项。

当前卷积神经网络的优化算法主要是**梯度下降法**及其变种：使用反向传导算法 [67, 111] 将损失函数的梯度从后向前传递到整个网络，然后用梯度下降法将网络参数朝损失下降的方向更新。优化目标函数对深度卷积神经网络的训练和收敛十分重要，这体现在：

- 在反向传导算法中，目标函数是产生梯度的源泉，整个网络得更新和收敛依赖于损失函数产生的梯度；
- 在训练过程中，损失函数是模型收敛的“指示剂”，通过观察训练集和测试集上损失函数的变化才能确定模型是否收敛，在必要的时候及时中断训练；
- 对于网络模型而言，损失函数指引着模型训练的方向，对于不同的任务要选取与该任务评价指标相容的损失函数，才能使得模型在测试机上有良好表现。

本文研究了深度学习技术在计算机视觉领域的几个代表性应用：显著性检测和分割、人脸识别和模型压缩剪枝，并详细分析了这几个任务中现有的目标函数所存在的问题，针对各任务的特性设计了损失函数。最后通过理论和实验来论证所提出损失函数的有效性。

1.1.2 国内外研究现状

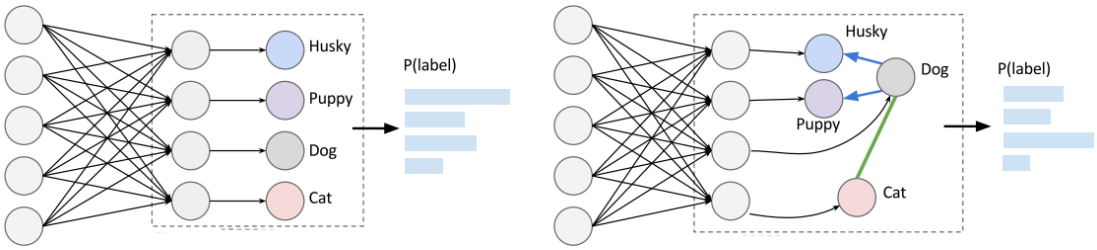
交叉熵（cross entropy）[1]在信息学中用于评估两个概率分布之间的差异信息。由于在机器学习中，模型的预测和样本的标签都可以看做是在所有标签上的概率分布，因此交叉熵损失函数被广泛运用于计算机视觉、自然语言处理的各种任务中。在计算机视觉领域，交叉熵损失函数（cross entropy loss）最开始被广泛用于训练图像分类网络 [62, 39]。后来，很多研究者将这以损失函数引入

[1] https://en.wikipedia.org/wiki/Cross_entropy

到语义分割 [89, 12]、显著性检测 [48]、目标检测 [29, 28, 109]等诸多任务中。因为广义上这些任务都可以看做是分类任务：语义分割和显著性检测可视为像素级别的分类任务，目标检测中的候选框识别是二分类任务，候选框中的物体分类是多分类任务。

另一方面，交叉熵损失用于分类任务也有一些缺点。首先，交叉熵损失不区分不同类别之间的误分类。例如将斑马误分类为马的损失与将斑马误分类为椅子的损失值一样，这显然与人类的认知不一致。其次，由于在交叉熵损失函数中样本标签通常是 onehot 向量 [2]，只在对应类别处有值，其他地方均为0。而这样的样本标签很容易造成模型的过拟合，因而要引入正则化项来控制模型复杂度，避免过拟合。此外，很多任务有着除了分类准确度之外的其他性能要求，因此单纯地使用交叉熵损失可能并不能满足这些性能上的要求。例如，显著性检测要求所检测结果对比度高；人脸识别要求所提特征区分度高等等。这些要求并不能通过最小化交叉熵损失函数达到。

近年来，针对交叉熵损失函数的这些不足，国内外研究者提出了很多基于交叉熵损失函数的改进，或者提出了新的损失函数。



(a) (b)

图 1.2 (a): 普通的交叉熵损失函数将所有类别独立看待，没有建模不同类别标签之间的结构关系。 (b): 文献 [21]所提出的基于**从属-互斥关系图**的损失函数能够建模标签之间的语义关系，得到和人类认知更符合的预测结果。图像来源于 [21]。

1.1.3 任务相关的损失函数

在分类任务，尤其是多分类任务中，类别标签之间有着互斥、从属等多种关系。如果能对这种关系进行建模，就能够规则模型的预测，使得模型预测更加符合标签语义之间的结构关系。邓等人 [21]使用一个树结构来表示样本标签之间的语义关系，并将这个结构关系定义为一种损失函数，来指导模型训练。

[2]onehot 向量表示向量中只有一个位置的值为1，其他位置的值都为0。假设5分类中的标注为2，经过 onehot 编码后变成 $[0,1,0,0,0]$

具体地，样本之间的语义关系可以分为从属（例如动物和斑马）、并列（斑马和马）以及互斥（动物和桌子）关系。这种层次关系可以很容易用一个树来表示。如 图 1.2所示，传统的交叉熵损失将各个类别标签独立看待，不考虑标签之间的依赖关系； Deng等人 [21]所提出的基于**从属-互斥关系图**的损失函数能够建模标签之间的语义关系，使得模型的预测结果和人的认知更加兼容。

但是，这种基于图的关系表示是一种硬指定（hard assign），是一种非此即彼的关系。同时，这种关系图需要手工构建，也需要大量的先验知识。Frogner等人 [26]提出用韦氏距离（wasserstein distance）代替交叉熵，在训练过程中最小化预测和标签之间的韦氏距离。如 图 1.3所示，韦氏距离度量了两个概率分布之间的“最优传输”，也就是将一个概率分布搬运到另外一个分布上的消耗。通过定义两个不同标签之间的“运输消耗”，可以将标签之间的关联性嵌入到损失之中。例如，斑马和马的运输消耗将小于斑马和桌子之间的运输消耗。因此将斑马预测成马的损失将小于将斑马预测成桌子的损失。

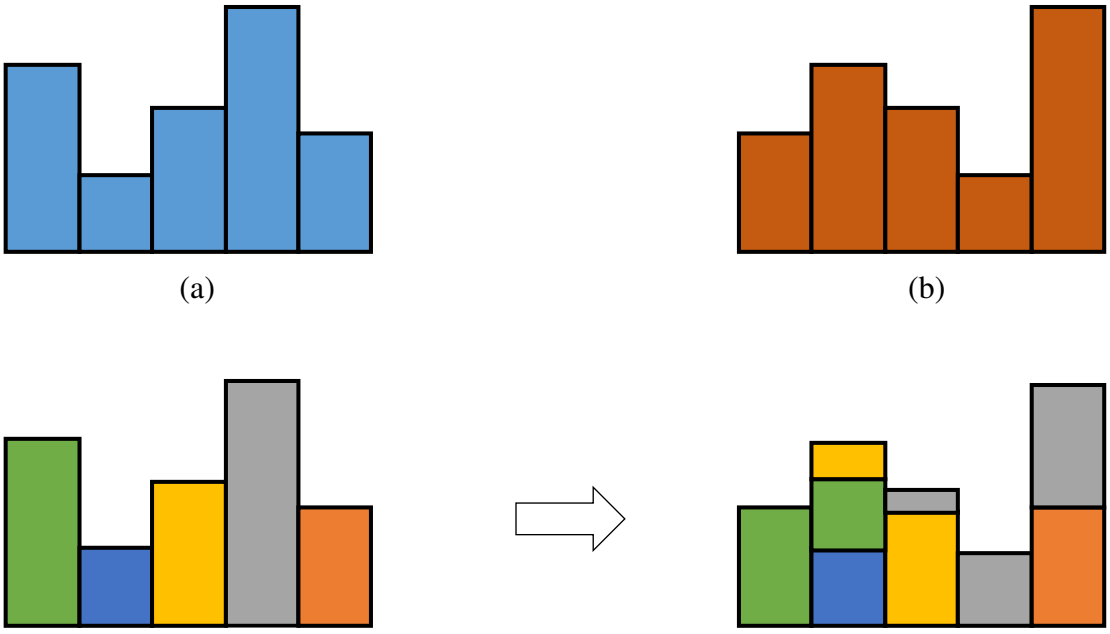
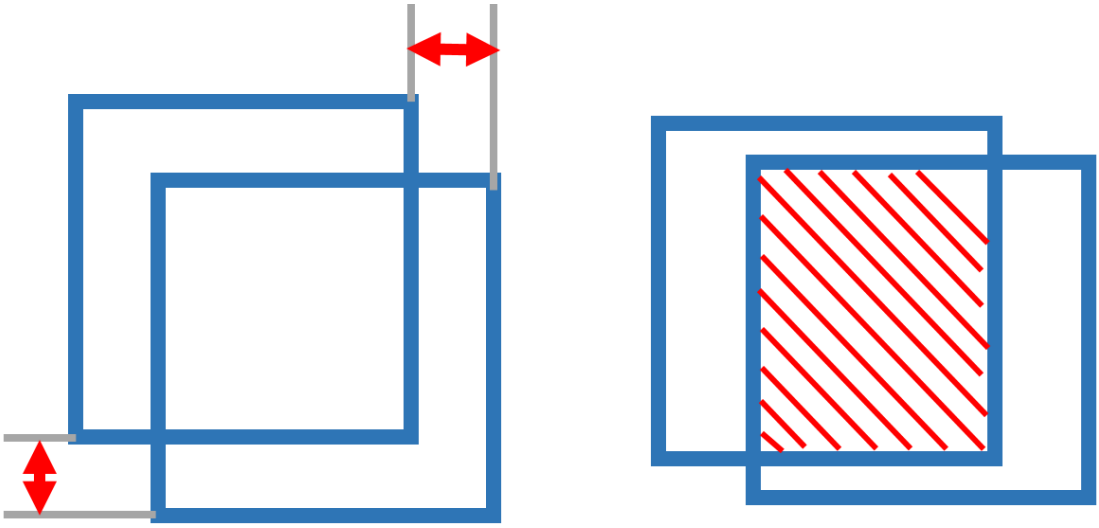


图 1.3 (a) (b): 两个概率分布 P_r 和 P_θ (c): 两个概率之间的最优传输。

在目标检测任务中，预测窗口和标签窗口之间的交并比（intersection over union）是评价包围盒（bounding box）的主要评价指标之一。而在训练阶段，目前的主流方法都是用预测的包围盒的四周与标签之间的误差距离作为损失函数。

因此损失函数并不能很好地反映包围盒的性能。使用预测包围盒和真实包围盒之间的交并比作为损失函数可以在训练过程中直接优化交并比，损失函数直接反映了包围盒的准确度。但是，交并比在预测和标定之间没有重合时始终为0，直接用交并比将导致损失函数不产生梯度，使得训练停滞。针对这一问题，文献 [110] 提出“泛化交并比”，泛化交并比即使在预测和标定之间没有重合区域时仍然产生梯度，指导模型训练。 在语义分割任务中，交并比也是评价模型质



(a) (b)

图 1.4 (a): 主流方法用包围盒上下和左右的预测误差距离作为损失函数。 (b): [110]提出直接在包围盒回归时直接以预测和标定之间的交并比作为优化目标。

量的重要指标，因此也有一部分研究用交并比作为训练语义分割模型的损失函数。为了解决交并比函数部分区域平坦的缺点，Rahman等人 [105] 用一个交并比的近似函数作为损失函数； Berman等人 [6] 则在 softmax 函数的基础上，使用一个放缩项使得损失函数称为交并比的近似。如 图 1.4所示，文献 [105]使用包围盒之间的交并比（IOU）作为损失函数。

在人脸识别任务中，所学到的参数的区分度是衡量模型质量的主要指标。具体地，我们希望同一个人的不同照片的特征尽可能相似，而不同人的照片特征尽可能具有很大的差异。而在交叉熵损失中，我们并没有显示地去最大化模型的区分度。在 CenterLoss [136] 中，作者为每个类别的人脸图片（每一类图片指的是来自同一个人的人脸图片）维护了一个特征中心（center），然后在交叉熵损失的基础上额外添加了一项来约束每个样本特征离该类别中心点的距离，

从而增强了同类样本的类内紧凑性。

$$\mathcal{L}_{center} = \frac{1}{2}\sum^{N}\left\|x - c_y\right\|_2^2, \tag{1.2}$$

公式(1.2)中，c_{y_i} 为类别为 y 的所有样本的均值点（中心），center loss $\mathcal{L}_{center}$ 为样本特征 x 到中心的欧氏距离。 如 图 1.5所示，center loss 可以显著增强同类样

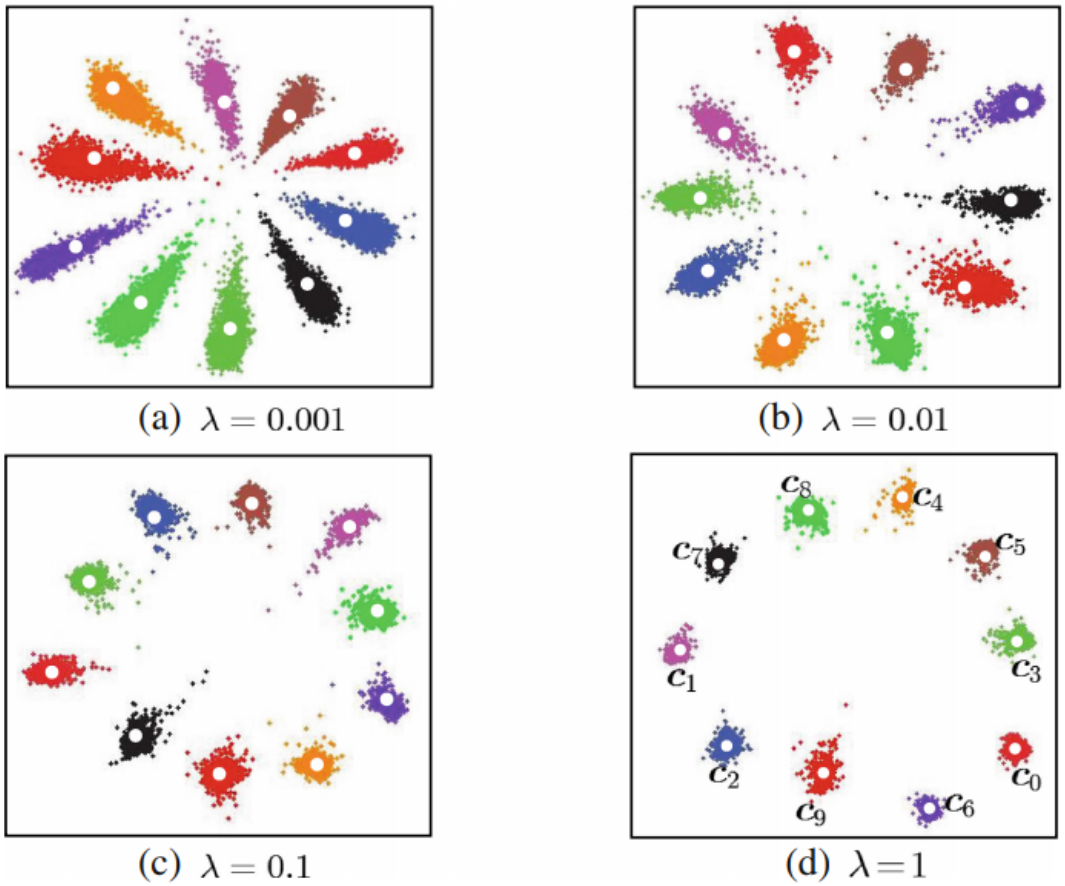


图 1.5 center-loss 在 MNIST 实验上的结果，λ 为 center loss的权重，同颜色的点表示同类别的样本特征，空心点为同类别样本的中心。随着 center loss权重 λ 的增大，同类别样本越来越紧凑。图像来源于 [136]。

本之间的紧凑度，从而提升了卷积神经网络模型的区分度。同时，也有一些工作 [86, 20] 在角度空间增强同类样本的紧凑度，取得了进一步的性能提升。但是，上述方法都只考虑了样本**类内的紧凑性**，却忽视了**类间的离散度**。因此，本文的一个研究重点就是通过改造人脸识别模型的优化目标，在增强模型类内

紧凑性的同时，达到增大特征的类间离散度。具体的说，不仅要使得同类别样本的特征之间距离小，而且要不同类别样本的特征之间距离大。

1.1.4 任务相关的正则项

正则项常用来约束模型参数，从而将先验知识嵌入到模型之中。例如，为了避免模型过于复杂而过拟合，通常使用 L2 正则化函数约束卷积神经网络的权重。由于 L1 正则更容易导致稀疏的模型参数，在很多模型稀疏化任务中会使用 L1正则作为优化目标的一部分。 在深度学习兴起之前，就已经有 LASSO

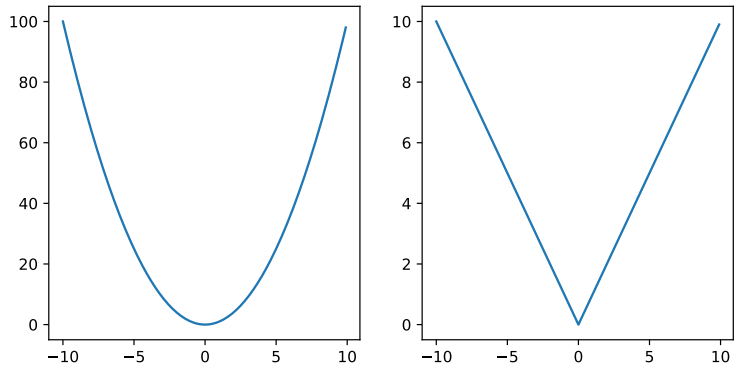


图 1.6 在零点处，L1 正则（右）比 L2 正则更加陡峭，有更大的梯度，因此 L1 正则更容易训练出稀疏的模型。

和 Group LASSO 等一批方法使用 L1 正则来获得稀疏的模型参数。在深度学习时代，L1正则被广泛运用于模型压缩。 Li等人 [77]在卷积神经网络的权重上施加 L1 正则，并在训练完成之后根据所得到网络参数的 L1 范数剔除部分参数，最终达到模型压缩的目的。在另外一个工作中，Liu等人 [87] 在批归一化层(BN: batch-norm)的缩放因子上施加 L1 正则。由于批归一化层对应每一个特征通道（也就是每一个卷积核）都有一个缩放因子，因此可以直接根据缩放因子的大小剔除整个卷积核。除了对模型参数进行剪枝，也有一类工作对网络连接进行剪枝。 Huang等人 [55] 在残差网络的残差连接上增加一个乘数并作为模型参数，然后对该系数施加 L1 正则，最后根据系数大小对残差连接进行剪枝。

除此之外，Louizos [91]等人提出在网络中使用 L_0 来得到稀疏的网络模型。但是 L_0 正则函数是一个可导的阶跃函数，无法直接作为优化目标。为了解决 L_0 不可导的问题，Louizos等人提出了一种在神经网络连接处的一种“门”装置，

这种门以一定概率开和关，这样开门的总数就是网络参数的 L_0 范数。这样一种设计模拟了 L_0 正则过程，并且使得整个过程可导，可以直接作为优化目标伴随网络的训练。

虽然 L1 和 L0 正则函数能够在一定程度上促进网络的稀疏，但是它们会对网络性能有较大的影响 [87]。当目标函数中正则化项的权重太大时，得到的模型性能下降；当权重太小时，又不能得到足够稀疏的模型。更重要的是，我们无法预先知道使用多大的正则化权重能够达到指定的稀疏比率。假设我们想在裁剪掉卷积神经网络中40%的网络参数的情况下，不引起较大的性能波动，我们无法确定一个合适的正则权重系数。上述的几种方法都是通过在不同数据集上进行搜索和试错，找到一个合适的系数，这样做显然不够智能。本文的另一个研究内容就是设计一种自适应的 L1正则机制，指定目标稀疏程度之后，本文所提出的算法能够在训练过程中自动调整 L1正则的权重，使得训练的网络经过指定比率剪枝之后不会引起较大的性能下降。

第二节　研究目标和主要贡献

本文的研究内容以卷积神经网络的优化目标函数为核心展开，为了推进卷积神经网络对不同计算机视觉任务的适应性，针对显著性检测、人脸识别和模型压缩三个典型任务设计了目标函数，提升了模型在这几个任务上的性能，特别是任务特性紧密相关的性能指标。具体地，本文在一下几个方面的问题进行了深入研究并提出了解决方案：

（1）针对显著性物体检测中当前方法普遍在训练阶段使用交叉熵损失（cross-entropy loss），但却在评测阶段使用 F-measure 评价预测质量的不一致现象，提出了直接以 F-measure 作为损失函数的解决方案。由于原始的 F-measure 不可导，本文提出对原始 F-measure 进行放缩，得到一个连续可导的近似。然后使用放缩的 F-measure 作为损失函数，指导模型在训练阶段直接最大化评测指标。相比于目前广泛使用的交叉熵损失，本文所提出的损失函数收敛速度非常快，通常在几百次迭代之后就能够有效预测出显著性物体的大致轮廓。同时，由于显著性检测任务存在显著的正负样本不均衡现象，交叉熵损失会导致检测结果准确率和召回率不平衡。由于 F-measure 是准确率和召回率的调和平均值，同时考虑了检测结果的准确率和召回率，因此广泛用于评测显著性检测结果的性能。本文采用 F-measure 作为优化目标，可以有效缓解检测结果中的准确

率/召回率失衡的情况。此外，由于所提出损失函数在饱和区域有很大的梯度，因此会迫使网络的输出两极分化，得到对比鲜明，不存在模糊区域的检测结果。在多个数据集上的定量对比表明，相比于交叉熵损失，本文所提出的损失函数能够显著提高模型预测的性能，平均 F-measure提升达到10%。此外，实验表明本文所提出的方法在基于深度图的显著性物体检测，图像中阴影检测中均有很好地表现。

（2）针对当前一些人脸检测算法主要侧重于增强所学习到的人脸特征的类内紧凑度（intra-class compactness），而忽略类间离散度（inter-class separateness）对特征区分度的影响，本文提出了一种可以显示地增强不同类别样本类间离散度的正则化项：互斥正则项。现有的一些方法例如 [136, 86]主要在欧氏空间或者是角度空间通过惩罚样本特征和类别中心之间的距离，使得样本特征向类别中心靠拢，从而达到增强类内紧凑度、提高模型区分度（discrimination）的目的。在另一方面，类间离散度，也就是不同类别的样本特征之间的疏远程度，也是决定模型区分度的重要因素。本文将卷积神经网络的分类层的权重作为各个类别的中心，并通过“互斥正则”函数显示地增大各类别中心与其最近邻类别中西之间的距离，达到增强特征类间离散度的目的。在多个公开数据集上的实验结果表明，本文所提出的正则项能够有效增强不同类别样本间的类间离散度，提升现有方法的人脸识别性能。

（3）针对现有神经网络剪枝方法的 L1 正则项权重系数需要通过大量实验手工调节的弊端，本文提出了一种自适应的正则函数。通过指定模型剪枝的比率，本文所提出的正则函数能够在训练过程中自动地调节权重系数。现有的方法通常针对不同的网络模型/数据集，搜索出一个合适的 L1 正则权重系数，然后在训练过程中使用固定的权重系数来规范模型参数。训练结束后，将一定比率绝对值比较小的参数从网络中剪除，从而得到压缩的模型。针对不同的数据集，不同的模型和不同的剪枝比率，正则项的权重系数要通过大量实验要选择，使得剪枝过程效率不高，不够自动化。本文所提出的“自适应正则”算法只需要给定期望的剪枝比率，就能够在训练过程中根据当前网络的稀疏性以及期望的剪枝比率自动的调节正则系数。在训练完成之后，按指定的比率剪出网络参数不会引起显著的性能损失，提升了压缩后模型的性能。在多个数据集上的实验表明，本文所提出的算法能够明显提升网络剪枝的效果。在和一个近期的研究结果 [87]的对比实验中，本文所提出的方法在不同数据集、网络模型、剪枝

比率下都有显著的优势。

第三节 本文的组织结构

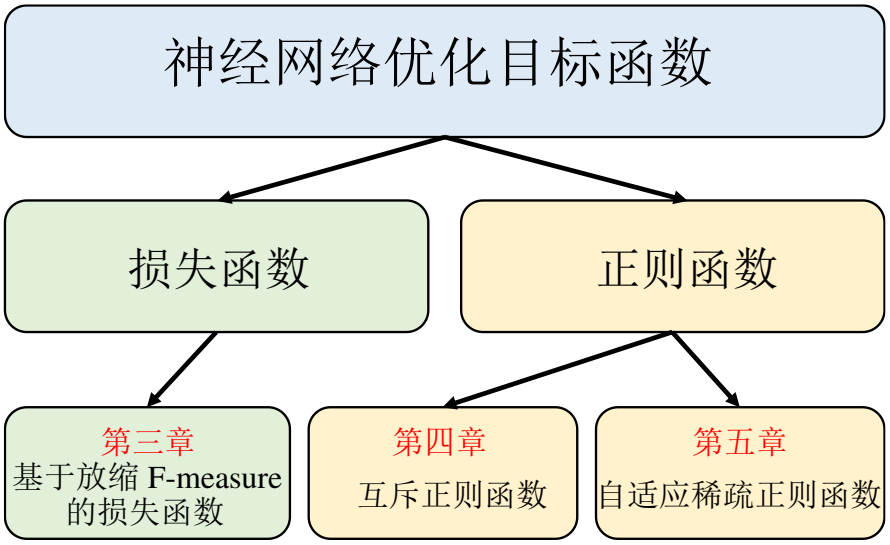


图 1.7 本文的组织结构和主要研究工作之间的相互关系。

本文组织结构如下：第一章介绍本文所涉及课题的背景、研究目标以及主要的研究成果和贡献；第二章介绍了在神经网络损失函数和正则化函数方面的几个代表性工作，包括在显著性检测、人脸识别和网络压缩等任务中针对任务特性设计优化目标例子；第三章详细介绍本文提出的基于“缩放 F-measure 的损失函数”及其在显著性检测任务中的应用，详细分析了显著性检测中现有损失函数的缺点和不足，并介绍了新损失函数的设计思路，最后实验验证了所提出算法的有效性。第四章介绍了本文所提出的“互斥正则”损失函数以及在人脸识别任务中的应用，详细介绍了该正则项的数学定义以及工作原理，并用大量实验证明了所提出方法的有效性第五章详细介绍了本文所提出的“自适应 L1 正则函数”以及在网络剪枝中的应用，以及在多个数据集上的大量试验结果。第六章总结全文，并对未来的研究作了展望。

第二章 相关工作

本文的主要研究对象为深度卷积神经网络的优化目标函数，因此本章介绍有关损失函数和正则函数设计的相关工作。

第一节 反向传导和损失函数

卷积神经网络的的历史可以追溯到日本研究人员福山提出的神经感知机（Neocognitron） [27]。受限于优化算法和计算能力，早期的神经网络往往只有几层，而且参数量也很少。直到 Rumelhart等人 [111]提出了反向传导算法（backpropagation），通过计算梯度的方法更新网络参数，才为训练大规模神经网络提供了理论基础。同时英伟达图形卡（GPU）计算能力为训练深度大规模神经网络提供了计算资源的基础。

这里我们用一个简单的三层神经网络（ 图 2.1）简述反向传导算法的工作原理。 **前向传导：**在前向传导（forward）阶段，输入层接收输入并逐层往后层

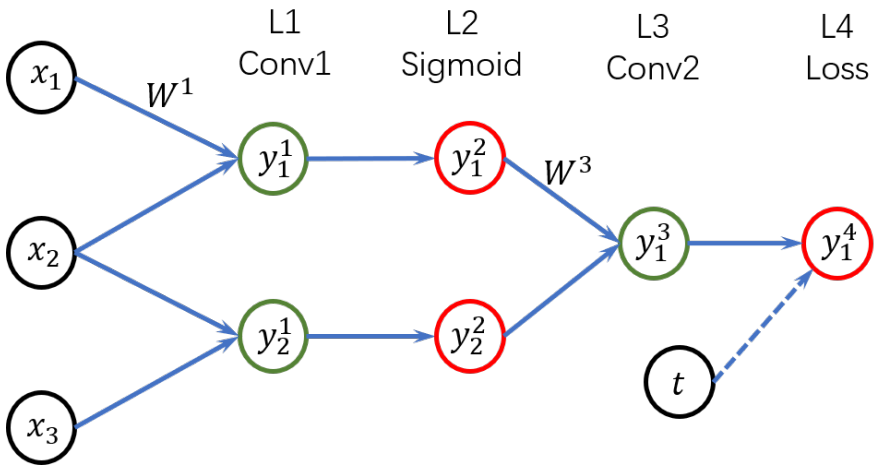


图 2.1 一个简单的 3 层神经网络，输入向量 $X \in \mathbb{R}^3$，输出预测 $\hat{y} \in \mathbb{R}$。

传递直到输出层。 x_i^l 表示“第 l 层的操作符的第 i 个输入”， y_i^l 表示“第 l 层的操作符的第 i 个输出”。在一次前向传导中，多个操作序贯进行：前一层的输出是下一层的输入： $y^l = x^{l+1}$。值得注意的是，神经网络中的每一个连接都带有

权重参数（weight）， 图 2.1中的每一根连线都有一个权重。

损失函数： 在前向传导得到网络输出值之后，我们要定量的评估网络的这次输出的预测的准确程度。在监督学习中，每个样本 x 对应都有它的标记（或者标签，label）y，损失函数 $\mathcal{L}(\cdot,cdot)$ 是模型输出 $\hat{y}$ 和样本标签 y 的函数。这里我们使用一个最简单的二次函数作为损失函数：

$$\mathcal{L} = (\hat{y} - y)^2. \tag{2.1}$$

反向传导：在得到网络预测的损失之后，我们用反向传导算法求得损失对网络参数的偏导数。反向传导算法包含以下两步：

1）梯度反传。对于最后一层（第 $l+1$ 层，也即损失函数层），其梯度为：

$$\delta^{l+1} = \frac{\partial \mathcal{L}(\hat{y},y)}{\partial \hat{y}}.$$

中间各层不直接产生梯度，但是接收后面层反向传递过来的梯度为

$$\Delta^l = f'(X^l) \cdot \Delta^{l+1}. \tag{2.2}$$

公式(2.2)中 Δ^l 为中间第 l 层的梯度，$f'(X^l)$ 为第 l 层输出堆输入的导数。对于全连接和卷积层而言，输出堆输入的导数就是网络权重参数本身，因此对于全连阶层和卷积层而言， 公式(2.2)可以写作：

$$\Delta^l = f'(X^l) \cdot \Delta^{l+1} = W^l \cdot \Delta^{l+1}, \tag{2.3}$$

其中 W 为 l 层的权重参数。

2）计算对参数的梯度。上一步反向传递回来的实际上是**损失函数对各层输出的导数**，而我们真正需要的是**损失函数对各层参数的导数**。这里以全连阶层为例，假设我们知道了损失函数对 l 层输出的导数 δ^{l+1}，那么根据链式法则，很容易得出：

$$\begin{aligned} \frac{\partial \mathcal{L}}{W^l} &= \frac{\partial \mathcal{L}}{X^{l+1}} \cdot \frac{\partial X^{l+1}}{W^l} \\ &= \frac{\partial \mathcal{L}}{X^{l+1}} \cdot X^l \\ &= \Delta^{l+1} \cdot X^l \end{aligned} \tag{2.4}$$

参数更新： 根据 公式(2.4)得出的损失函数对参数 W 得导数，我们根据梯度下降法更新网络参数：

$$W^{t+1} = W^t - \alpha \frac{\partial \mathcal{L}}{W}, \tag{2.5}$$

由于损失函数是反向传导算法中产生梯度的源泉，因此在神经网络的优化中处于很重要的地位。同时，为了将先验知识嵌入到网络参数，往往会使用使用正则函数作为优化目标的一部分。例如在训练卷积神经网络时，为了避免过拟合，通常会用 L2正则限制模型参数的大小。最终，神经网络的优化目标 $\mathcal{J}$ 为损失函数和正则函数的和：

$$\mathcal{J} = \mathcal{L} + \lambda ||W||_2^2, \tag{2.6}$$

其中 λ 是一个很小的权重系数。

第二节 神经网络损失函数的进化

神经网络广泛采用反向传导算法计算损失函数对参数的梯度，然后用梯度下降法更新模型参数。因此，神经网络的训练对损失函数的选取十分敏感。损失函数（loss function）是衡量神经网络在训练样本中表现标准，在优化过程中指引着网络参数更新的方向，因而损失函数的选取也和和任务密切相关。最初 Rumelhart 等人 [111] 提出用反向传导算法训练神经网络时，使用的损失函数是平方距离损失函数：

$$\mathcal{L} = (\hat{y} - y)^2.$$

其中 y 是样本的标签（label），$\hat{y}$ 为模型的预测（prediction）。很快有人发现，最小化交叉熵损失能够使得分类模型收敛的更快更好 [5, 71]。同时也有一些研究者提出了另外一些损失函数，取得了比交叉熵损失更好的性能 [25, 46]。

Glorot等人 [30]分析了为什么交叉熵损失会比二次函数损失收敛更快更好的原因。他们认为由于交叉熵损失在边缘地带（神经网络预测与样本标签相差很远）的梯度显著大于二次函数损失，而更大的梯度显然会加速模型收敛。因为梯度下降法训练网络模型就像随机在山丘上放置一枚小球，然后让它滑落到最低点。显然，如果山坡更陡峭，下降的速度会更快。同时，神经网络的训练采用了带有动量（momentum）的随机梯度下降法，历史梯度的积累会帮助模型跳过局部最优解（local minimum），得到效果更好的模型。

虽然收敛的比二次方损失更快，交叉熵损失在实际应用中仍存在一些不足。由于交叉熵损失不考虑各类别之间的关联，导致网络输出与人认知不一致的预测。例如，将斑马和马误分类的损失显然要比斑马和桌子的误分类，但是交叉熵损失并不考虑不同类别之间的语义关联，因此会得到相同的损失值。

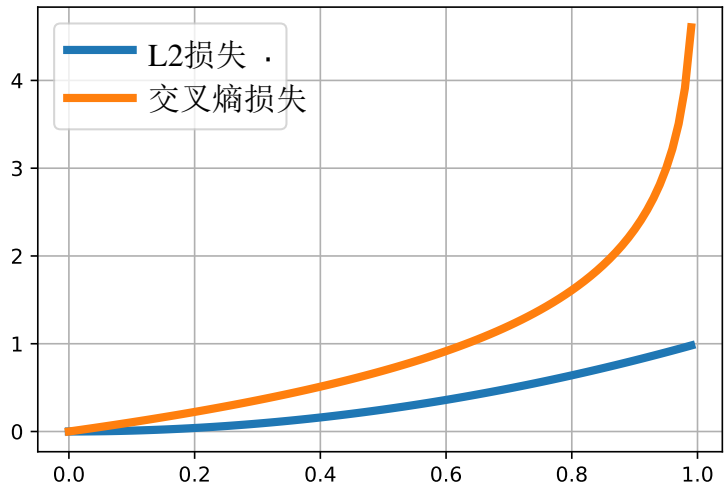


图 2.2 交叉熵损失（$l = -y \cdot \log(\hat{y})$）的梯度比 L2损失（$l = (y - \hat{y})^2$）要高，特别是当预测远离标注的时候。

Deng等人 [21]根据标签（label）的语义关系，在交叉熵损失的基础上构建了一个标签关系图（label relation graphs），并通过标签之间的互斥、包含和并列等关系来计算预测的损失。但是这种关系图仍然是以一种硬指定（hard assign）强行规定了各个标签之间的相容性，并且标签关系图的构建需要大量的人类先验知识和手工操作。

Frogner等人 [26]提出用韦氏距离（wasserstein distance）取代交叉熵，来度量模型预测和样本标签之间的误差。韦氏距离是最优传输（optimal transport）在离散情况下的特例，它评价两个概率分布之间的“最优传输问题”，也就是把一个概率分布搬运到另一个概率分布上消耗最小的方案。考虑两个概率分布 P 和 Q，以及概率 p,q 之间的一个传输 $T_{i,j}$，其中 $T_{i,j}$ 满足：

$$\begin{aligned} \sum_i T_{i,j} &= p_j \\ \sum_j T_{i,j} &= q_i. \end{aligned} \tag{2.7}$$

也即 T 的足边缘分布（marginal distribution）为 p 和 q。现在规定从 i 到 j 的传输消耗为 $G_{i,j}$，矩阵 G 称为 ground-metric。韦氏距离等于 p,q 之间传输的最好

的消耗：

$$W = \inf_{T} \langle T_{i,j} G_{i,j} \rangle_F, \tag{2.8}$$

其中 $\langle \cdot \rangle_F$ 是 Frobenius 内积[1]。现在把模型的预测和样本的标签分别当做两个概率分布，通过不同标签 i 和 j 之间的语义距离来设置 $G_{i,j}$。 Frogner等人 [26]首先使用自然语言处理中的 word2vec [94]技术计算每个标签的词向量，然后再计算不同标签之间词向量的距离。由于韦氏距离能够通过测地距离 ground-metric 来指定不同类别之间的距离，因而可以将不同类别之间的语义联系通过测地距离嵌入到损失函数中，让模型输出与人类认知更一致的预测。

在原始交叉熵损失中，样本标签是“硬指定”（hard assignment）的，这种硬指定的标签容易造成模型的过拟合，降低测试的准确度。 Szegedy等人 [127]提出了一种被称为标签平滑（label smoothing）的技术。原始交叉熵损失中的样本标签可以认为是一种 onehot向量 Y，如 图 2.3(a) 所示。标签平滑的样本标签 $\hat{Y}$ 为原始的 onehot向量 Y 和均匀分布 U 的加权平均：

$$\hat{Y} = \alpha U + (1 - \alpha) Y,$$

如 图 2.3(c)所示。有研究表明，标签平滑在分类任务中可以使同类样本的特征在表示层更加紧凑 [97]。 原始交叉熵损失计算模型输出和样本“硬标签”之间的交叉熵：

$$\mathcal{L}_{CE} = \sum_{k} Y_k \cdot \log(\hat{Y}_k) = -\log(\hat{Y}_y). \tag{2.9}$$

标签平滑损计算模型预测和平滑后的“软标签”之间的交叉熵：

$$\mathcal{L}_{CE} = \sum_{k} Y_k \cdot \log(\hat{Y}_k). \tag{2.10}$$

公式(2.9)和 公式(2.10)中，$\hat{Y}$ 是模型预测，$\hat{Y}_i$ 表示样本属于第 i 类的概率。 Y 是样本标签，在原始交叉熵损失中，$Y_y = 1, Y_{k,k\neq y} = 0$。

度量学习（metric learning）的目的是学习一个距离度量函数 $f(\cdot)$，使得相似的样本在特征空间彼此靠近，而不同的样本在特征空间彼此远离。由于度量学习与其他任务例如图像分类、语义分割、目标检测等任务有完全不同的目标，评价指标也全然不同，因此在传统任务中广泛使用的交叉熵损失在度量学习中并不是最好的选择。

[1] https://en.wikipedia.org/wiki/Frobenius_inner_product

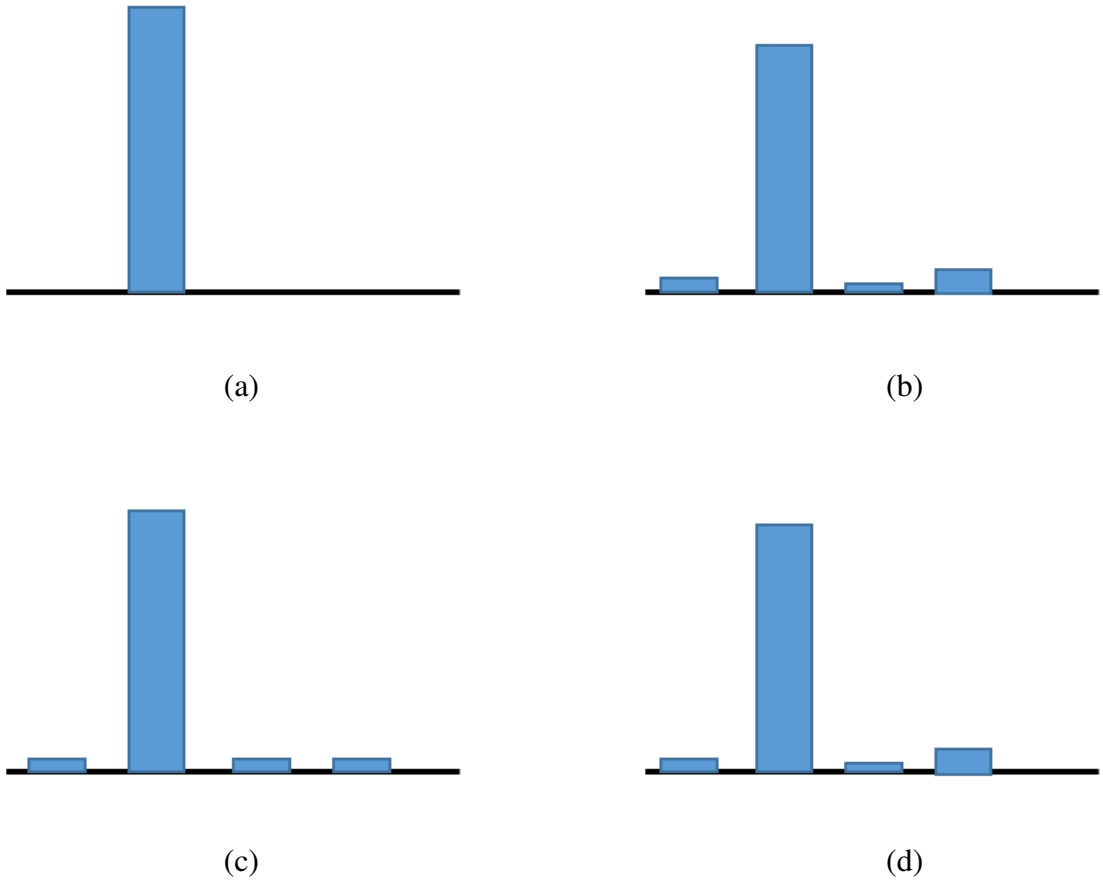


图 2.3 (a) 交叉熵损失的样本标签； (b) 交叉熵损失的模型预测； (c) 标签平滑损失函数中的样本标签； (d) 标签平滑损失函数中的模型预测。

孪生网络（Siamese network）被用在卷积神经网络中用于手写签名认证 [9]。如 图 2.4在孪生网络中，两个分支网络共享所有的网络参数。两张不同的手写签名分别输入到孪生网络中，并输出两个特征向量，最后在输出端对比两个特征向量来确定两个手写签名是否来自同一个人。

孪生网络通过接受成对的正例或者是负例来学习样本之间的距离度量，是一种十分成功的度量学习模型。共享两个子网络的参数对网络从图像中学习有用特征有正面影响，而且共享参数可以减少模型的存储开销。 Hadsell [33]等人提出 contrastive loss来训练孪生网络。给定任意一对样本，contrastive loss定义为：

$$\mathcal{L}_{contrastive}(X_i, X_j) = (1-Y)\cdot d(X_i, X_j)^2 + (Y)\cdot\{\max(0, m - d(X_i, X_j))\}^2. \quad (2.11)$$

其中，X_i, X_j 是一对输入样本的特征。 $Y(\cdot)$ 是示性函数，当输入样本属于同一

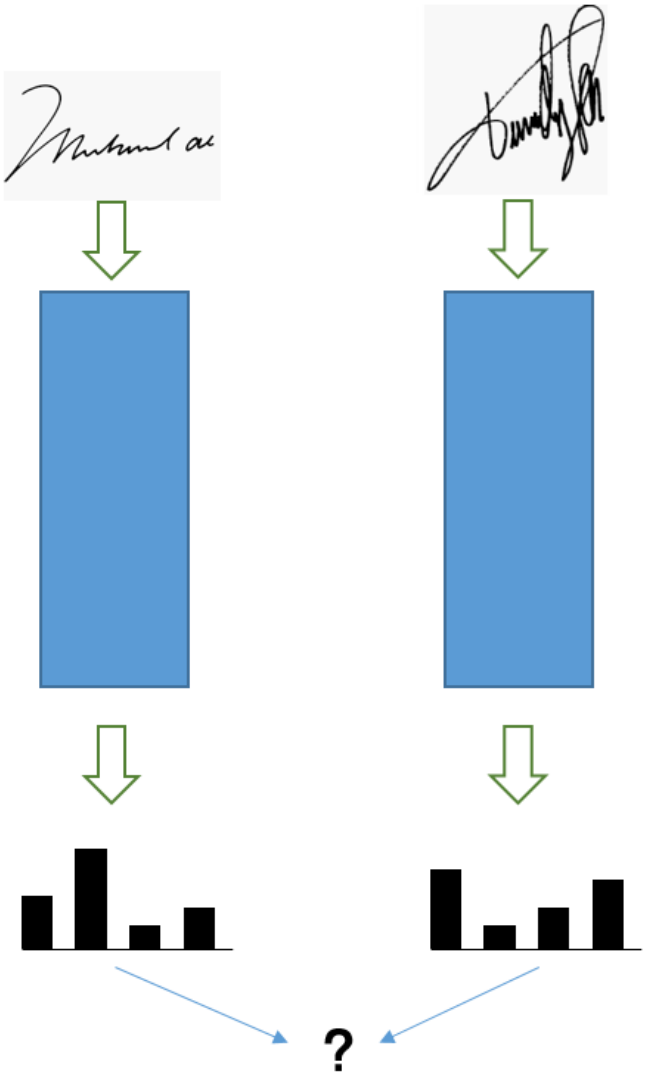


图 2.4 孪生网络中两个网络共享所有参数。两个不同手写签名分别输入到两个网络中，最后通过对比特征向量确定是否为同一个人的签名。

类时，$Y=1$，否则 $Y=0$。 m 是一个边界（margin）参数，表示不同类别样本特征之间边界的宽度，$d(\cdot,\cdot)$ 是某种距离度量，比如欧氏距离、角度距离等等Contrastive loss的设计理念十分简单：当一对样本来自同一类时，就惩罚他们之间的距离，这样使得同类样本的特征相互靠近，此时优化目标是 $d(X_i,X_j)^2$；当两个样本来自不同类别并且距离大于边界（margin）m 时，就迫使它们互相远

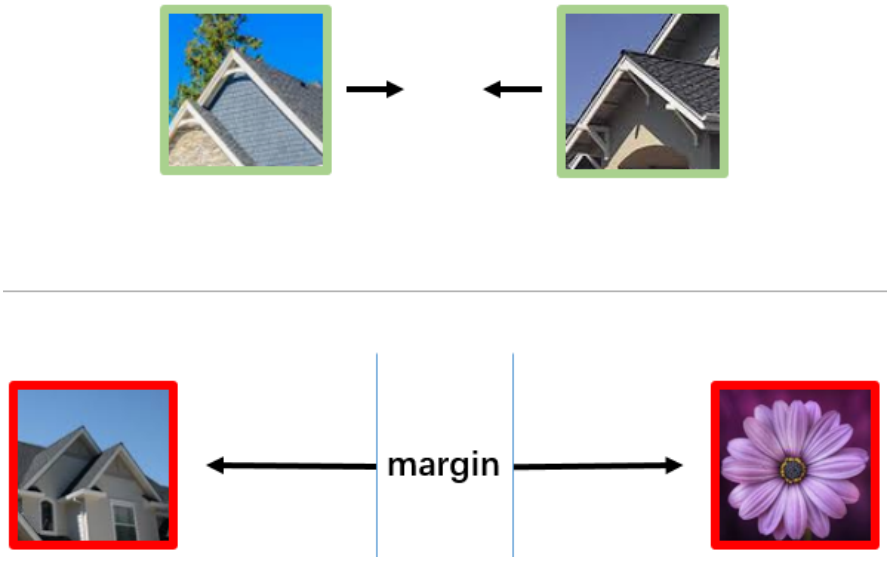


图 2.5 上：当两个样本来自同一类别时，contrastive loss 惩罚样本特征之间的距离。下：当样本来自不同类且之间距离小于边界参数（margin）m 时，contrast loss 迫使样本特征彼此远离。

离，此时优化目标是 $\{\max(0, m - d(X_i, X_j))\}^2$。虽然 Contrastive loss 收敛速度较慢，且收敛非常依赖输入样本对的选取，但是启发了后来的很多研究工作。

在 contrastive loss 的基础上，Hoffer等人 [47]提出了三元组损失函数（triplet loss）。如 图 2.6在 triplet损失中，模型的输入为一个三元组（triplet），分别是：

- 锚点样本（Anchor）；
- 正样本（Positive），与 Anchor同类或者很相似的样本；
- 负样本（Negative），与 Anchor不同类或者不相似的样本。

Triplet loss 在训练过程中比较样本之间的**欧氏距离**，该损失函数的设计思想直接来源于度量学习。如 公式(2.12)所示，triplet loss 计算一对来自相同类别和一对来自不同类别的样本特征之间的距离，而所有这些样本的特征都是用同一个网络提取的。最后的输出结果要通过对比两个样本对之间的距离而得出。Triplet loss 在优化中直接最大化 $d(A,N)$ 和最小化 $d(A,P)$：

$$\mathcal{L}_{triplet}(A,P,N) = \max(0, d(A,P)^2 - d(A,N)^2 + \alpha). \tag{2.12}$$

和 contrastive loss一样，α 也是一个边界参数（margin）。

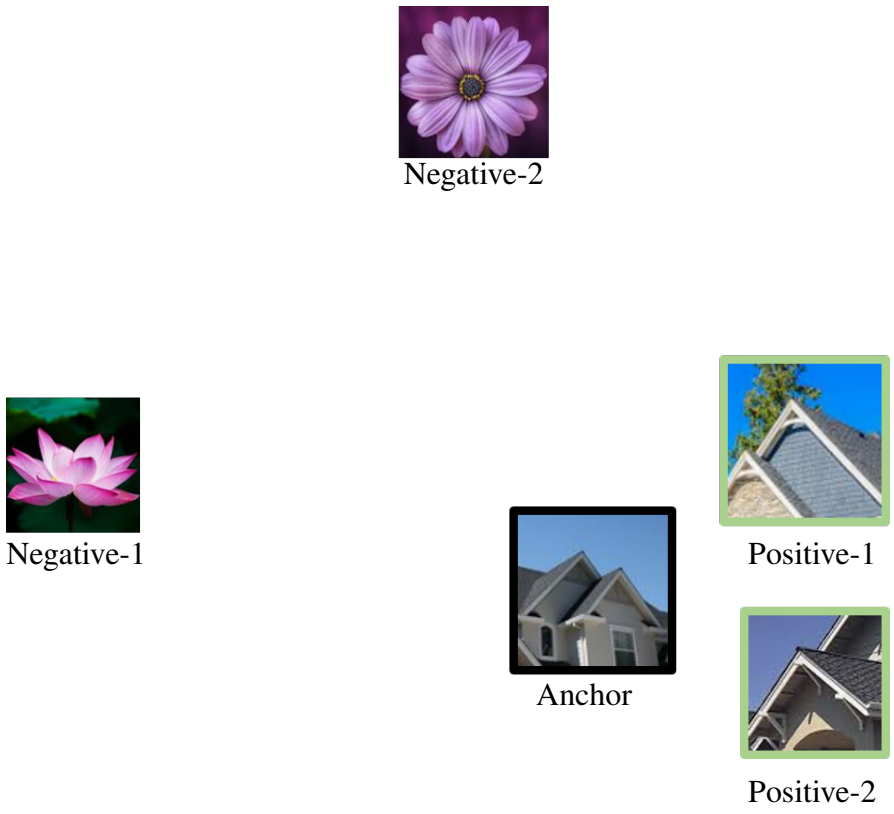


图 2.6 Triplet loss 同时最大化不同类样本之间的距离（$d(A,N)$）和最小化同类样本之间的距离（$d(A,P)$）。

在 contrastive loss 中，模型每次只能“看到”两个样本，通过样本是否匹配来确定两者的特征距离应该增大还是减小。在 triplet loss 中，模型每次能够“看到”三个样本：通过比对正样本和锚点，确定负样本远离锚点的更新方向；同时根据负样本和锚点，确定正样本靠近锚点的移动方向。因而，如 图 2.7所示，在 contrastive loss 和 trplet loss 中，样本位置更新的方向不一样。由于在一组训练样本中同时考虑了类间的离散性和类内的紧凑性，triplet loss 能够得到更有区分度（discrimination）的模型。

进一步，在多样本对损失函数（N-pair loss） [119]中，Sohn等人在三元损失（triplet loss）的基础上，在一组样本中包含多个负样本（ 图 2.8）。这样能够使得类别间的决策边界更加确定，样本更新的方向更一致：

$$\mathcal{L}_{N-pair}(x,x^{+},\{x_i^{-}\}_n=1^N)=\log(1+\sum_{i=1}^{N-1}\exp(x^T x_i^{-}-x^T x_i^{+})). \tag{2.13}$$

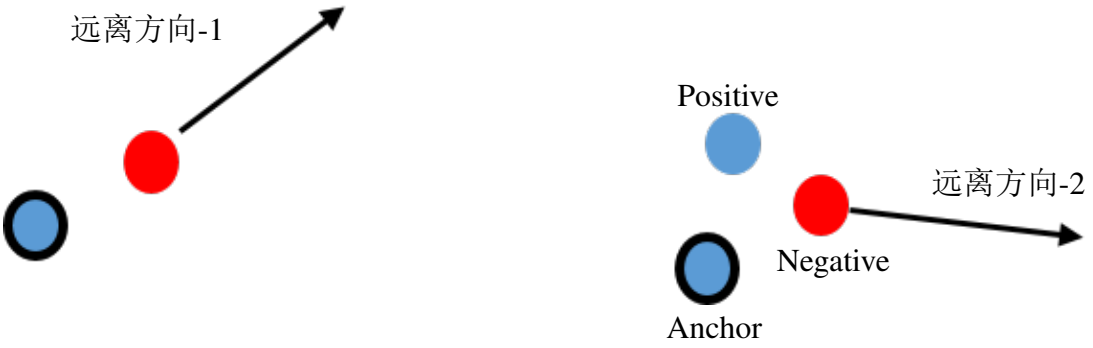


图 2.7 Contrastive loss 和 Triplet loss 中样本位置更新的方向不一样。左：在 contrastive loss 中，模型仅考虑两个样本是否属于同类，如果不属于，则两个样本朝相互远离的方向移动；右：在 triplet loss 中，在对比 **锚样本（Anchor）**和 **正样本（Positive）** 之后，**负样本（Negative）**朝“既远离 A，又远离 P”的方向移动。

公式(2.13) 中 x, x^+, x^- 分别代表 **锚样本（Anchor）**、**正样本（Positive）** 和 **负样本（Negative）**。 如 图 2.8 和 公式(2.13) 所示，N-pair loss 每组训练数据中包含

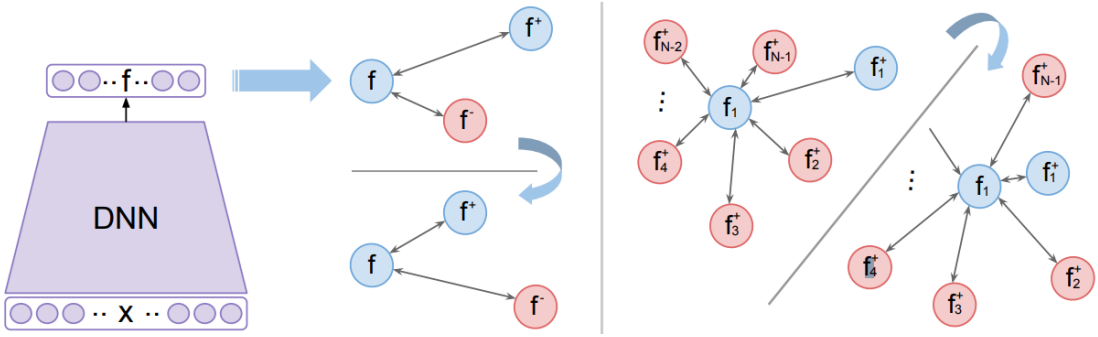


图 2.8 N-pair loss 损失函数中，模型通过观察锚点样本、正样本以及多个负样本的位置来确定正样本更新的方向。图片来自 [119]。

多个负样本，正样本在更新时移动的方向根据这些负样本综合确定。这样可以使得更新时的随机性减小，加快模型收敛速度和提高模型收敛时的性能。

2.2.1 人脸识别中的损失函数

人脸识别算法要求输入一对人脸照片，预测这两个照片是不是属于同一个

人，通常通过计算不同人脸图像对应特征的距离来进行判断。因此人脸识别和度量学习在性能需求上有一定相似。

DeepFace [128]和 DeepID [123]是率先使用使用深度学习技术进行人脸识别的工作。其中 DeepFace算法第一个在 LFW数据集 [54]上取的超过人类识别率的人脸识别算法（DeepFace: 97.35% v.s. 人类: 97.53%）。由于收到 AlexNet的影响，DeepFace 采用了类似的分类网络架构，在训练阶段训练人脸分类器；在测试阶段提取全连阶层特征。 DeepFace 算法用4百万张带有类别标定的图片训练了一个九层的卷积神经网络模型。在测试阶段将倒数第二层的输出向量作为图片的特征，并通过计算不同图像的特征之间的距离确定识别结果。

为了学习到更加具有区分度的特征，DeepID算法 [123]结合了分类的交叉熵损失和度量学习中的 contrastive loss 损失函数。如 公式(2.11)所示，如果一对样本来自同一类，那么 contrastive loss 损失函数迫使它们更接近；而如果来自不同的类别，则迫使它们远离彼此。这样，在交叉熵损失和 contrastive loss 的双重监督下，DeepID 能学到比 DeepFace 只用交叉熵损失更鲁棒的人脸特征。

紧接着，度量学习中的另外一个损失函数，triplet loss 损失函数，也被引入到人脸识别中来 [114, 113, 113]。在 triplet 三元组的选择上，FaceNet [114]选择“困难样本”（hard examples）作为负样本来学到更加鲁棒的特征。具体的，“困难样本”指的是那些在正确类别上输出概率低的，模型难以区分或者预测出错的样本。在 triplet 三元组中，困难样本就是锚点样本（anchor）最近的负样本。

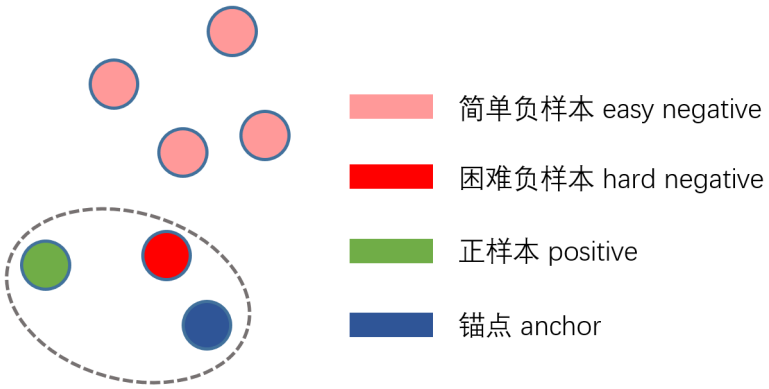


图 2.9 Triplet 三元组中的困难样本选择：选择不同类样本中离锚点（anchor）最近那个样本作为三元组中的负样本。

但是 contrastive loss 和 triplet loss 有时候由于不合适的训练样本选取，会遇

到训练不稳定的情况。有一些方法开始探索改进交叉熵损失，使得模型既能够稳定收敛，又能够提取出具有区分度的特征。 Wen等人 [136]在交叉熵损失之外提出了“中心损失”（center loss）来减小同种类别样本间的聚合度。 center loss 的定义如下：

$$\mathcal{L}_{center} = \frac{1}{2}\sum_{i=1}^{M} ||x_i - c_{y_i}||_2^2 \tag{2.14}$$

公式(2.14)中 x_i 为样本 i 的特征，y_i 为样本的类别标签， c_{y_i} 为所有类别为 y_i 的样本特征的中心（均值）。在训练中同时使用交叉熵损失和 center loss监督模型，文献 [136]中的总体损失函数为交叉熵损失和 center loss的加权和：

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda \cdot \mathcal{L}_{center}, \tag{2.15}$$

其中 λ 是 center loss的权重。这样网络不仅能够正确分类人脸照片，而且同一个人的照片在特征空间会更加聚集，因此特征会更鲁棒和有区分性。值得注意的是，由于网络在训练中参数会不断更新，导致样本特征也在不断更新，因此每类样本的中心 c 也必须随着训练不断更新。

之前的方法计算样本之间的距离大多在欧式空间（Euclidean space）计算。2017年后，陆陆续续一些研究表明，将样本特征归一化（normalize）之后使用样本特征之间的角度来衡量距离 [131, 86, 20, 156, 106]。主要的原因可能是人脸照片的模长（norm）会受到照片清晰度和该类样本数的影响 [106]，因而使用欧氏距离度量人脸相似度会受到模长的影响。如 图 2.11所示，清晰度高的正脸照具有较大（high）的模长，而清晰度低的侧面照模长较短。

假设 $x \in \mathbb{R}^D$ 为样本特征，$W \in \mathbb{R}^{D\times K}$ 为分类层的权重，其中 D 为样本特征维度（dimension）， K 为类别总数。则样本 x 属于类别 k 的概率为：

$$p_k = \frac{\exp(W_k^T \cdot x)}{\sum_{i=1}^{K}\exp(W_i^T \cdot x)}, \tag{2.16}$$

其中 W_j 为权重矩阵 W 的一列。 公式(2.16)又被称为“softmax”函数。最后，交叉熵损失为：

$$\mathcal{L}_{ce} = -\log(p_y), \tag{2.17}$$

式中 y 为样本标签。结合 公式(2.16)和 公式(2.17)：

$$\mathcal{L} = -\log(\frac{\exp(W_y^T \cdot x)}{\sum_{i=1}^{K}\exp(W_i^T \cdot x)}). \tag{2.18}$$

图 2.10 人脸图片特征的模长（norm）会受到图片质量（拍摄角度，清晰度等等）的影响。质量高的人脸照片模长更大，低质量的图片模长更小。因此使用欧氏距离度量人脸照片之间的相似度会受到模长的影响。

当样本特征 x 和权重 W 的每一列 W_i 都被归一化后，上式可以改写为：

$$\mathcal{L} = -\log\left(\frac{\exp(\cos\theta_y)}{\sum_{i=1}^{K}\exp(\cos\theta_i)}\right). \tag{2.19}$$

其中，θ_i 为样本特征 x 与第 i 类对应的权重向量 W_i 的夹角。 公式(2.19)将交叉熵损失引入角度空间，因此被称为**角度交叉熵损失**(angular cross entropy loss)。

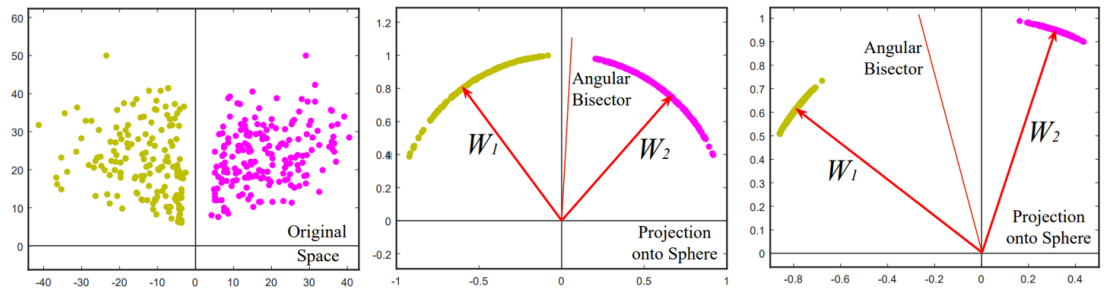
Liu等人 [86]提出在角度空间（angular space）来增强同类样本间的紧凑度（compactness）。这种方法被称为 SphereFace，因为权重被归一化后，整个损失函数的计算在一个单位球上进行。具体来说，对于标签为 y 的样本 x，SphereFace在计算 公式(2.19)中的损失时，将 θ_y 乘上一个系数 $m \geq 2$，这样模型会迫使 x 往更靠近 W_i 的方向移动。

$$\mathcal{L}_{\text{sphere\_face}} = -\log\left(\frac{\exp(\cos m\theta_y)}{\sum_{i=1,i\neq y}^{K}\exp(\cos\theta_i)+\exp(\cos m\theta_y)}\right). \tag{2.20}$$

SphereFace损失函数中的边界是通过乘以一个系数 m 来实现的。在另外一个研究中，Deng等人 [20]提出使用“加性”的边界，取得了更好的性能，这个方法被称为 ArcFace。

$$\mathcal{L}_{\text{arc\_face}} = -\log\left(\frac{\exp(\cos(m+\theta_y))}{\sum_{i=1,i\neq y}^{K}\exp(\cos\theta_i)+\exp(\cos(m+\theta_y))}\right). \tag{2.21}$$

Liu等人 [79]认为训练集中的每个类别都需要不同的边界参数，因此他们为每个类别都初始化一个 m 参数，然后用强化学习来自动学习这些边界参数。



(a) (b) (c)

图 2.11 二维空间内三种不同损失函数的样本特征分布图。(a):原始的交叉熵损失，样本点分布在整个欧式空间，同类样本聚成一簇；(b):角度交叉熵损失，样本点分布在单位圆上；(c):Sphereface [86]通过在角度空间中不同类之间建立边界（margin），使得同类样本更加紧凑。图像来源于 [86]。

第三节 正则函数

虽然正则函数也是优化目标的一部分，但是关于正则函数的相关研究没相对较少。这主要是因为正则函数在网络优化中在大部分情况下只是为了避免模型陷入过拟合（overfit），因而通常的 L2 正则就可以了。过拟合是指模型在训练集上的损失特别小，准确率特别高，但是在测试集上的表现却在下滑。由于深度神经网络模型复杂度高，参数量大，因此很容易拟合训练集并取得很低的训练损失。但是过拟合的模型的泛化能力（generation）往往很差，在新的、从未见过的测试数据上表现不好。

目前大多数主流的方法，不论是图像分类 [62, 39]，目标识别 [28, 109]，还是显著性检测 [48, 134]和人脸识别识别 [156, 20, 86, 79]，都广泛地使用 L2 正则来限制模型参数和防止过拟合。

近年来，由于深度学习算法移动端部署的需求，网络压缩和剪枝成为一个新的研究热点。神经网络参数的稀疏化是模型压缩和剪枝的前提：只有剔除那么些接近0的参数，才能在对模型性能影响尽可能低的情况下进行模型压缩和参数剪枝。而 L2正则相比 L1正则，由于在零点附近的梯度消失现象，不太容易产生稀疏的模型。

Li等人 [77]直接在卷积层权重参数上施加 L1 正则，并在训练完成之后剔除一部分参数绝对值比较小的卷积核，达到参数剪枝的目的。这么做的依据是参数绝对值比较小的卷积核，对该层应输出的特征也会比较小，总体而言对模型最后的预测也会影响比较小。但是在 resnet [39]、densenet [53]等网络

中使用了批归一化（batch normalization）操作，卷积层输出的特征会被归一化成满足均值为0，方差为1的状态。因此卷积层输出的特征的大小并不会影响后续卷积层输出的大小。在 Network slimming [77]中，作者以批处理层（batch normalization）的缩放因子作为卷积核重要性的依据进行剪枝。具体的，批处理层按一下公式处理输入特征：

$$y = \alpha \frac{x - U}{\sigma} + \beta \tag{2.22}$$

公式(5.1)中 U 是输入特征 x 的均值，σ 是方差，α 是缩放因子，β 是偏置项。因此用缩放因子 σ 作为特征重要性的标准，比直接用卷积层参数更合理。

除了 L1 正则之外，Louizos [91]等人还提出了使用更能促进网络稀疏性的 L0 正则函数进行网络压缩。由于 L0 正则函数本身并不可导，因此在实际使用中必须进行可导函数的近似，在实际运用中效率不高，因此在网络压缩和剪枝任务中最广泛采用的仍然是 L1 正则函数。

第三章 基于放缩 F-measure 损失函数

本章提出了基于放缩 F-measure 的损失函数，并研究了该损失函数在图像中显著性区域检测中的应用。本章 第一节介绍了显著性检测相关背景知识， 第二节介绍了显著性检测的相关工作， 第三节介绍了显著性检测的相关数据集。第四节介绍了显著性检测的评测标准，重点介绍了 F-measure 评测标准及其计算过程。 第五节详细推导了基于放缩 F-measure 的损失函数， 第六节用实验论证了所提出损失函数的有效性， 第七节对本章做了小结。

第一节 引言

3.1.1 背景知识

人类视觉系统具有在复杂环境中进行实时场景理解的能力，这一能力是在自然界长期适应环境和进化中得来的。对于计算机而言，实时快速的场景理解和响应依然是一个很难的任务。认知科学和神经科学的研究表明，人的视觉和神经系统在视觉信号的传输和处理过程中，能够选择性过滤掉一部分视觉信号，优先处理重点区域或者重点物体相关的视觉信号 [130, 61]。例如，在原始森林中，人类能够快速识别草丛中的毒蛇并作出逃跑和躲避的行动，而忽略毒蛇旁边的其他物体和区域。这种“无意识的选择重点区域而忽略无关区域”是人类和其他很多动物视觉系统的一个重要功能，是在长期进化中能够保存自身的重要武器。这种“重要区域”通常被称为显著性区域（saliency region）或者是显著性物体（saliency object）。在一些近期的研究中，显著性物体或区域检测被简称为显著性检测（saliency detection） [134, 153, 7, 142, 73, 82]。

早期的显著性检测工作主要集中在人眼睛的视点预测（fixation prediction），也就是预测人类眼睛视线集中的一个点的位置。 Cheng等人 [14]提出使用图像中的全局对比度信息来挖掘图像中的显著性物体，而不是仅仅预测视线集中的某一个点。这个阶段的工作主要利用显著性物体区域和图像背景的对比来进行显著性检测，例如亮度，纹理或者直方图统计等等。

Liu等人 [83]使用机器学习技术，通过标定显著性物体数据集，训练机器学习模型来检测显著性显著性物体。之前的工作将人类对显著性物体/区域的先验

知识通过对比度计算等方式显示地嵌入到模型。在基于机器学习的显著性物体检测中，关于显著性物体的知识通过训练数据的方式隐式地嵌入到模型中，模型通过不断地从数据中学习来理解什么是显著物体。使用深度学习技术进行显著性检测的工作始于 [133]。 Liu等人 [82]使用深度学习技术，训练大型的卷积神经网络来检测显著性物体，使得显著性物体检测的性能得到了飞跃式的提升。由于全卷积神经网络 [89]可以端到端的快速进行像素界别的预测，因此被广泛运用于显著性物体检测 [48, 155, 153]。

随着深度传感器的普及化，很多移动设备都配备了深度摄像机。有一些研究人员开始结合彩色图像和深度图来检测场景中的显著性物体和区域 [108, 104, 34]。这些方法将深度图（depth map）作为图像数据的补充，在不同的阶段将深度图提取的特征和深度图提取的特征进行融合。

基于深度学习的显著性检测方法主要依赖于大量已标注的显著性分割数据集来训练深度卷积神经网络模型，然后在测试阶段预测每个像素的显著性。Long等人 [89]提出了全卷积神经网络（FCN，fully connected neuralnetworks），可以端对端的（end-to-end）进行快速的像素级预测，因此有大量的后续研究试图改进全卷积神经网络的结构，使得其能生成更精细的显著性图 [48, 153]

第二节 相关工作

近十年研究人员提出了大量的显著性检测方法。这些方法有的基于全局或者局部对比度，有的基于稀疏编码理论的，也有的方法使用图模型（graphcal model），还有一些方法在频域检测显著性物体。最近几年（2015-），随着深度学习的兴起，涌现了一批基于深度学习的显著性检测方法。

3.2.1 非深度学习的显著性检测方法

由于通常情况下属于显著性区域（物体）的像素在颜色和亮度上与图片上的其它像素有一定的区别，也就是说和背景部分的对比度较高，因此计算每个像素与其它像素的颜色差异是一种比较直观的计算显著性的方法。

2006年 Zhai [149]等人提出任一像素的显著性可以通过计算该像素和其它所有像素的颜色差异而得到。具体地，每个像素的颜色可以表示为三维空间中的一个点（RGB），两种颜色的差异即为它们三维空间中的欧氏距离。 2008年，Achanta等人 [2]通过在不同尺度上比较局部图像区域与其领域在亮度和色彩上

的区别，从而计算局部区域的显著性。之前的方法都指在一个领域内比较图像块的对比度，2015年 Cheng等人 [14]提出通过计算全局对比度来确定某一个区域的显著性。

值得一提的是，2017年，上海交通大学的 Xiaodi Hou和 Liqing Zhang [49]利用显著性区域和背景区域在频域（spectral domain）的对比度，提出了频域残差法（spectral residual）在频域内检测显著性区域。具体地，频域残差法将图像变换到频域，然后根据图像中非显著性区域频域幅度谱的尖峰部分，使用低通滤波器滤掉尖峰部分。受到这个工作的启发，复旦大学的 Guo等人 [31]提出在频域的相位谱上检测显著性区域。

通过计算稀疏和稠密重建误差来预测图像区域的显著性。每个图像区域的稠密重建误差（dense reconstruct error）通过以背景区域为基的主成分分析（pca）得到；而稀疏重建误差（sparse reconstruct error）定义为图像稀疏表示的残差。这两种重建误差通过临近的图像区域传导至整幅图像，并最终融合成整幅图像的显著性响应图。这个方法从一个全新的角度处理显著性检测任务，并且也取得了不错的实验结果。

图模型（graphical model）是一种利用图数据结构来建模因素之间相互关系的模型。在图像中，不同的图像区域之间可以用图模型来建模。在基于图模型的图像建模中，一般用图像过分割 [15]或者超像素分割 [3]等技术将原图分割成若干个不规则区域。然后以每个区域为一个图节点（graph node），在节点之间进行推理（inference）。由于图模型可以很好地建模图像区域之间的空间相关性，因此被广泛应用在图像分割和显著性检测等任务中。 Hare等人在2017年的论文中 [37]提出了基于图模型的显著性检测方法（GBVS，graph-based visual saliency）。通过构造一个随机马尔可夫链，链上的每一个节点对应图像的一个区域，然后计算节点的显著度并通过马尔可夫链传播，最后将显著性归一化得到整个链上的显著性输出。 Jiang等人 [57]将图（graph）上的显著性检测表示为一个吸收马尔可夫链的吸收过程。每个节点的显著性等于转移节点到吸收节点的平均时间。

3.2.2 基于深度学习的显著性检测方法

基于图像块的深度学习显著性检测方案

随着深度卷积神经网络（deep convolutional neural networks）技术的兴起，

有很多研究者也使用深度学习模型来检测显著性区域。

在正式介绍具体的基于深度学习的显著性方法之前，我们先给出深度学习显著性检测的一般过程。给定一张输入图片 $I \in \mathbb{R}^{H\times W\times 3}$，卷积神经网络模型 $f(\cdot)$ 输出一个显著性响应图（saliency map）$S = f(I) \in \mathbb{R}^{H\times W}$，每个像素的值 $S_{i,j} \in [0,1]$ 表示该像素的显著性，或者属于显著性区域的概率。

对于深度学习显著性检测模型而言，$f(\cdot)$ 通过大量数据的训练而得到。我们假设模型的参数为 θ，也即 $S = f_\theta(I)$，模型在训练过程中不断地调整参数 θ 来使得预测的更加准确。假设我们有数据集 $\mathcal{D} = \Big\{\{I_1, Y_1\}, \{I_2, Y_2\}, ..., \{I_N, Y_N\}\Big\}$，每一组数据中 I 为图像，$Y \in \Delta\{0,1\}^{H\times W}$ 为人工标注的显著性图（ground-truth）标注，一般以二值图（binary map）的形式存在。训练的目的就是找到合适的 θ，使得模型在整个训练集上的损失最小：

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \ell(f_\theta(I_i), Y_i). \tag{3.1}$$

其中 $\ell l(\cdot)$ 是某种距离度量，例如交叉熵损失。接下来我们介绍几种最具代表性的深度学习显著性检测模型。

最初使用深度学习检测显著性的方法都是将输入图像使用超像素分割或者是过分割技术技术分割成若干个小的区域，然后在每个区域内独立的去预测显著性。如图 3.1所示，使用 SLIC [3]超像素分割算法将图像分割成若干个小的

图 3.1 使用 SLIC 算法进行图像超像素分割的例子。输入图像被分割为若干个小区域，每个区域内部的纹理、颜色等信息相似。以超像素为单位进行图像处理比逐像素处理高效很多。

超像素区域，区域内像素点具有相似的颜色和纹理等底层特征。因此，以超像素为单位进行显著性检测笔逐像素的预测要高效很多。

Zhao等人在2015年提出多重上下文深度学习（MCDL，multi-context deep learning）模型 [158]，利用两路神经网络分别提取局部和全局信息。其中局部特征提取支路提取一个超像素内的特征，全局特征提取支路提取全图的上下文信息（context information）。最后这两个支路提取的特征将送到一个多层感知机中（MLP，multi-layer percentron）。 Lee等人 [70]将一个由底层特征计算出的距离图（distance map）和由神经网络提取的超像素特征拼接，最后用一个线性模型预测每个超像素的显著性。 Li等人 [75]使用一个预训练好的分类网络来提取每个超像素分割的多尺度特征，最后用一个多层感知机模型预测超像素级别的显著性。 He等人 [42]先将图像进行超像素分割，然后在每个超像素内分别提取两个与颜色相关的特征，并将这两个特征合并后输入到一个一维的卷积网络（Conv1D）中得到该超像素的预测结果。由于该方法对图像做了多个尺度的超像素分割，因此最后的多个预测结果要经过上采样（upsample）才能合并到一起。

除了以超像素为单位进行显著性检测推断之外，有一些方法还试图在物体候选区域或包围盒上检测显著性物体。 Zhang等人 [150]首先用深度卷积神经网络模型产生大量候选窗口（bounding box proposals），然后用最大后验概率准备（Maximum a Posteriori）从这些候选窗口中检测显著性区域。值得注意的是，该方法的输出也是检测窗口的形式。 文献 [59]先生成一些候选区域，然后用一个

图 3.2 文献 [150] 预测显著性物体的包围盒（bounding boxes），对于不存在显著性物体的输入图像，该方法输出为空。

卷积网络模型来将这些区域归类为一些提前预设的形状模板。最终的检测结果为所有这些二值的模板的均值。

基于全卷积网的端到端显著性检测

2015年，Long等人 [89]提出了用于图像语义分割的全卷积网络模型。区别于传统的卷积神经网络在末尾用若干个全连阶层做预测，全卷积网络在网络最后对特征进行上采样（upsample）到输入图像的尺寸，然后用 1×1 卷积层进行**逐像素**预测。 全卷积网络可以在一次前向传播中预测所有像素的输出，因此被

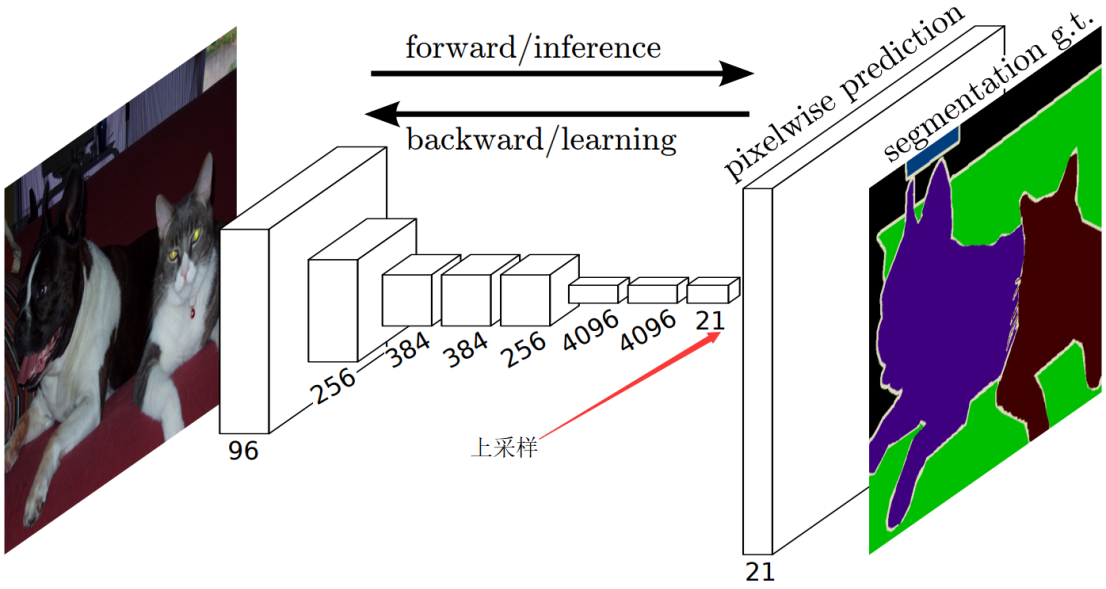


图 3.3 全卷积神经网络（FCN） [89]最出被用在语义分割任务上。将普通卷积神经网络最后一个卷积层输出的特征图进行上采样，然后用 1×1 的卷积层直接做像素级的预测。图像来自 [89]。

广泛运用到语义分割 [89, 12] 、边缘检测 [141]，物体骨架检测 [115, 157]、显著性检测 [153, 48] 等需要逐像素预测的任务上，取得了非常优异的性能和效率。

同时，如 图 3.3所示，全卷积网络最后的输出是由低分辨率的特征图上采样得到的，因此最后的检测结果往往分辨率不高，很多物体边缘的细节部分不够精细。

许多早期基于全卷积网络的方法采取了“逐步精炼（refine）”方法来克服全卷积网络分辨率低的缺点。 2017年，Wang [134]等人提出了一种递归的全卷积神经网络（RFCN，recurrent fully convolutional networks），在递归运算

中，后一个网络的输入是前一个网络的输出和原始的输入图像，经过数次这样的精炼（refine），最终得到的显著性检测结果会比单独的全卷积网络更加细致。 Kuen [64]等人也提出了一种逐步精炼（refine）的显著性检测方案，不同于 [134]，Kuen等人的方法在后续的过程中只选择性精炼物体周围的一部分显著性图。 Hu等人提出的方法 [51]提出在超像素层面进一步优化得到精细的显著性图。

Zhang等人 [151]使用空洞卷积 [12]来提升全卷积网络输出的分辨率，并且通过同时建模显著性和预测中噪声来优化预测的精细程度。文献 [72]通过训练一个变分自编码器（VAE，variational autoencoder） [60] 来重建输入图片的背景部分，并通过重建误差得到显著性检测结果。

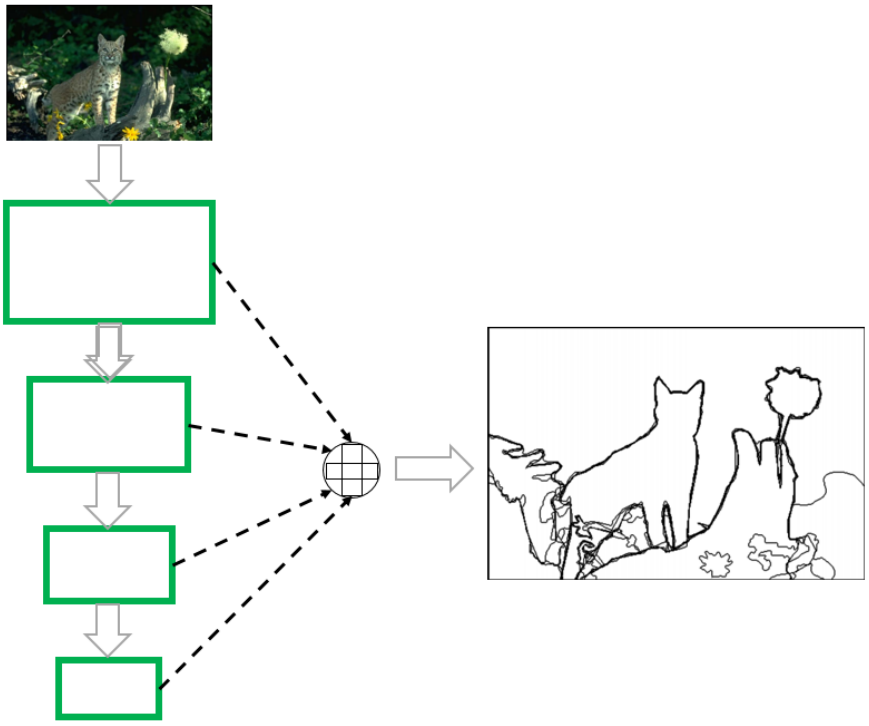


图 3.4 HED（Holistically-Nested Edge Detection）算法将神经网络中间层的特征抽出，得到边输出（side-output）。最终模型的输出为多个不同尺度边输出的融合。

后来，有一些方法尝试融合多种尺度的显著性检测结果。 2017年，Li等人 [76]用三个网络分别处理三种不同尺度的输入图片，最终三个输出通过一个一个可以学习的权重自适应的合并到一起。 Wang [135]也采取了多尺度网络逐步优化的策略。该方法首先通过一次预测产生粗略的显著性图，然后逐渐用局

部的上下文特征来优化这个粗略的显著性图。最终多个尺度的输出通过空间金字塔池化层来合并成最终的输出。 2019年，大连理工大学的 Zeng 等人 [147]分别使用了全局（global）和局部（local）两个网络分别提取全局显著信息和局部图像细节，从而实现了高分辨率显著性检测。同时他们还构建了一个高分辨率的显著性检测数据集。 Zhuge [160]等人将多尺度的深度特征投影到一个度量空间（metric space），然后在度量空间将多尺度的特征进行自底向上的融合。

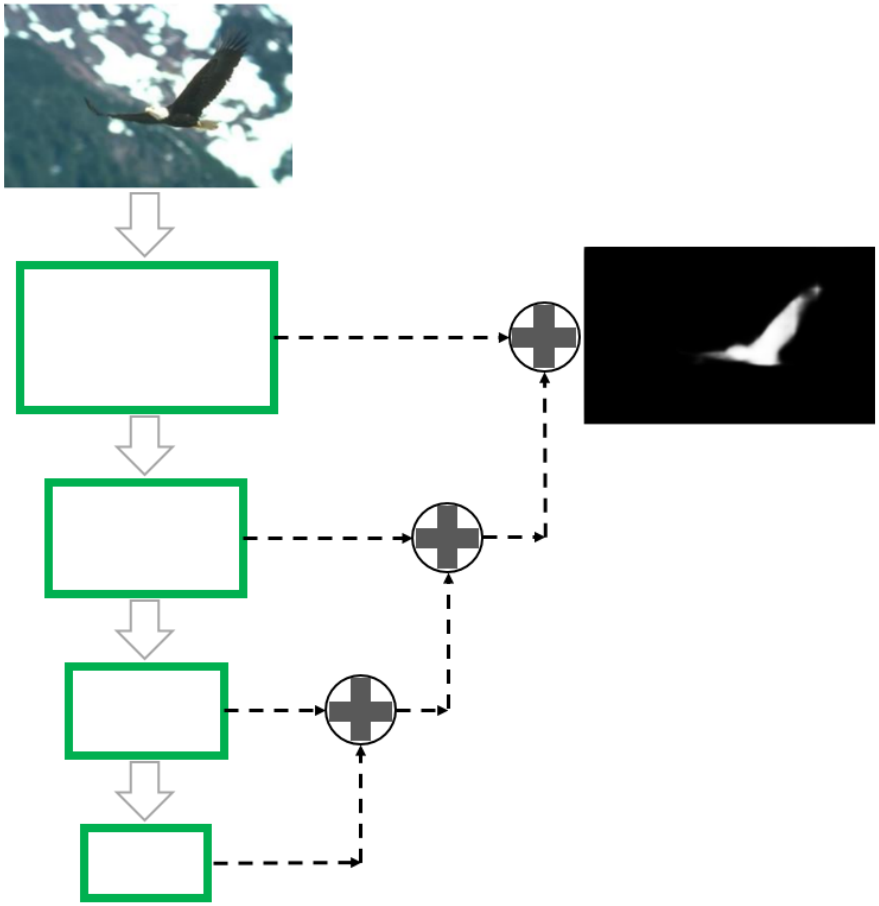


图 3.5 在 HED [141]的基础上，Hou等人 [48]提出了一种自底向上的表输出优化策略，通过递归地将深层的输出与邻近的浅层输出融合，从而让底层特征得到顶层特征的语义信息。

基于深浅层深度特征融合的显著性检测方法

2017年，UCSD 的 Xie等人 [141]提出了一种融合不同阶段深度特征进行边缘检测的方法。该方法被简称为 HED（holistically-nested edge detection）。低层

特征语义特征不丰富，但是分辨率高；而高层特征语义信息丰富，而分别率低下。如果合理的融合不同阶段的深度特征，就能得到语义信息丰富而且分辨率高的特征，进而得到精确的预测结果。具体地，HED 从卷积网络地不同阶段提取出语义信息/分辨率各不相同的深度特征和输出，并独立的监督这些输出。这些卷积神经网络中间各层的输出被称为“边输出”。最后总的输出是各边输出的加权平均。该研究成果获得了2017年的马尔奖提名[1]

受到 HED 的启发，有很多研究工作在简述性检测中也尝试融合不同阶段的深度特征，得到更强的特征，从而提升显著性检测的性能。 2017年，南开大学的 Hou 等人 [48]提出一种自上而下、从深到浅的逐步精炼（refine）的策略。具体来说，在 HED 中各边输出都是独立预测，然后平均到一起。由于底层特征语义信息不足，因此底层的边输出会含有很多噪声；同时来自深层的边输出则有分辨率不足的缺点。 Hou 等人 [48] 提出递归地将深层边输出上采样并和相邻的浅层边输出想融合，最终会极大地缓解浅层边输出噪声过大的现象，因为浅层特征得到了深层特征中的信息。

同年，Zhang等人 [153]提出了一个类似的网络结果，该网络结构先进行多尺度的特征融合，然后进行自顶向下的特征精炼，使得精炼后的特征兼具高分辨率和高语义信息。值得一提的是，该方法在特征融合之前使用了物体的边缘信息来帮助特征更准确地区分边界地区的显著性物体和背景。

Hu等人 [52]将多个边输出累积到一起，然后用若干个卷积层反复的优化累积的边输出以得到更加细致的显著图。 Zeng等人 [148]使用多个空洞卷积 [12]产生四个尺度不一的特征图，然后再通过上采样到原图尺寸，得到多个显著性图。

第三节 显著性检测相关数据集

除了方法之外，数据集也会对深度学习模型的性能有较大影响。 Hou等人 [48]等人指出，在仅包含 1000 个样本的 ECSSD [142] 数据集上训练得到的模型性能比包含 10,000 个样本的 MSRA10K [14] 数据集更好。本节介绍显著性检测相关的数据集，本文所涉及的数据集均有完整的像素级别的标注。相关数据集的统计信息，包括数据集发表年份、发表的刊物、数据集复杂程度、数据集包含样本数目等信息记录在 表格 3.1。

图 3.6列举了几个 MSRA-B数据集 [84]数据集中的样本图片和对应的标注。

[1] https://tc.computer.org/tcpami/iccv-best-paper-award/

数据集	样本数	发表年份	发表刊物	对比度
MSRA-B [84]	5000	2011	TPAMI	高
ECSSD [142]	1000	2013	CVPR	高
HKU-IS [75]	1447	2015	CVPR	低
PASCALS [78]	850	2014	CVPR	中
SOD [96]	300	2010	CVPRW	低
DUT-OMRON [143]	5168	2013	CVPR	低

表 3.1 本章涉及到的几个显著性检测数据集的统计信息。“对比度”表示样本中前景部分（显著性区域）和背景的对比度，对比度越高数据集越简单。

图 3.6 MSRA-B [84] 数据集中的图例和标注。 MSRA-B 数据集中的样本比较简单，通常一张图中只包含一个与背景区域对比明显的显著性区域。

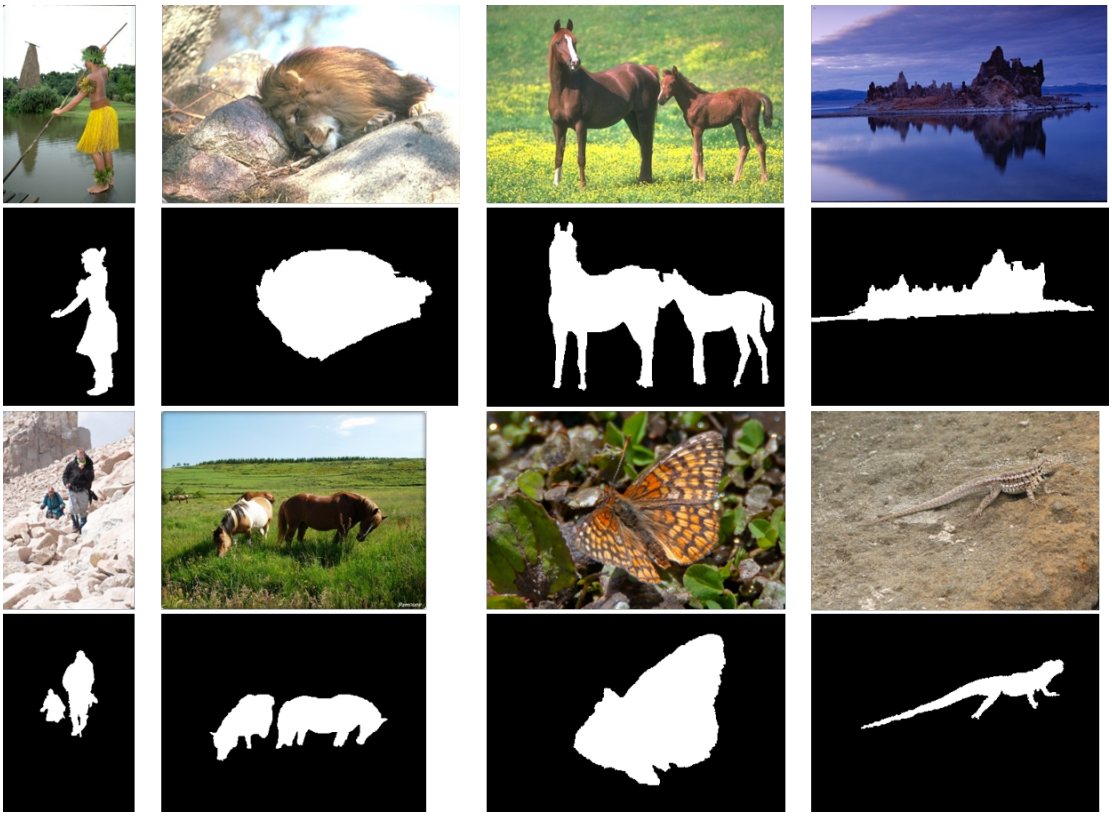


图 3.7 ECSSD [142] 数据集中的图例和标注。 ECSSD 数据集中的样本更多样化，同一张图中有时会包含多个显著性物体。

MSRA-B数据集 [84]中每张图片仅有一个显著性物体，且该显著性物体与背景的对比度往往比较大，而且通常处于图像的中心位置。总体来说 MSRA-B是一个相对简单的显著性检测数据集。

如图 图 3.7所示，ECSSD数据集中一个样本中可能有多个显著性物体。虽然 ECSSD数据集中的图片可能包含多个显著性物体，相比 MSRA-B数据集更难，但是数据集中的显著性物体和背景的对比度比较大。

近年来，研究人员开始收集背景复杂，前背景对比度低的困难数据集来增强显著性检测方法在复杂场景中的鲁棒性。例如 DUT-OMRON [143] 收集了5168 张背景复杂的图片，有些图片中有3-4个显著性物体（区域）， HKU-IS [75]数据集也收集了1447张场景复杂的样本图片。 PASCALS [78]从 PASCAL VOC数据集中挑选了 850 张图片，并从原始数据集的语义分割标注得到逐像素的显著性区域标注。这些数据集背景更复杂多变，更具有挑战性。 图 3.8是

图 3.8 DUT-OMRON [143]数据集中的样本图片和对应标注。 DUT-OMRON 数据集中样本可能包含多个显著性物体，背景复杂，对比度较低。

DUT-OMRON 数据集中的样本示例。

第四节 显著性检测方法的评测

显著性检测的结果一般是一个与输入图像大小一致的显著性图（saliency map）$\hat{Y} \in [0,1]^{H\times W}$，其中 $\hat{y}_{i,j} \in [0,1]$ 表示原图中 i,j 位置的像素属于显著区域的概率。

假设显著性检测结果为 $\hat{Y} \in [0,1]^{H\times W}$，对应的人工标注为 $Y \in \{0,1\}^{H\times W}$，目前大多数方法采用以下几种评价指标（evaluation metric）来定量评价检测结果的质量：

- 平均绝对误差（MAE，mean absolute error）；
- 准确率（Precision）
- 召回率（Recall）

3.4.1 平均绝对误差（MAE）

平均绝对误差定义十分简单：

$$\mathrm{MAE}(\hat{Y},Y) = \frac{\sum_{i,j}|\hat{Y}_{i,j} - Y_{i,j}|}{H\times W}. \tag{3.2}$$

虽然 MAE定义简单实现也容易，而且有十分直观的解释，但是在实际中并不能完全反映检测结果 $\hat{Y}$ 的质量。特别是当有一些样本显著性物体区域占比很小的时候（在一些数据集中这种情况很常见），即便模型将全部的样本预测为背景，也会有很小的平均绝对误差。

3.4.2 准确率（Precision）、召回率（Recall）和 F-measure

给定连续的显著性检测结果 $\hat{Y} \in [0,1]^{H\times W}$，我们先用一个阈值 τ 将预测结果二值化，得到二值化的显著性检测结果 $\hat{Y}^{\tau}\{0,1\}^{H\times W}$。然后我们定义真阳（true positive），假阳（false positive）和假阴（false negative）：

$$\begin{aligned} TP(\dot{Y}^t,Y) &= \sum\nolimits_i 1(y_i == 1 \text{ and } \dot{y}_i^{\tau} == 1), \\ FP(\dot{Y}^t,Y) &= \sum\nolimits_i 1(y_i == 0 \text{ and } \dot{y}_i^{\tau} == 1), \\ FN(\dot{Y}^t,Y) &= \sum\nolimits_i 1(y_i == 1 \text{ and } \dot{y}_i^{\tau} == 0). \end{aligned} \tag{3.3}$$

其中 i 为图像上的位置索引。 TP 为“标注为正（属于显著性区域），预测也为正的像素的个数”，FP 为“标注为负（不属于显著性区域），预测却为正的像素的个数”， FN 为“标注为正（属于显著性区域），预测却为正的像素的个数”。

图 3.9 当显著性区域占比较小时，平均绝对误差（MAE）评价指标对显著性区域的变化不敏感。“样本2-预测-1”漏掉了途中显著性物体–马–的半个身子，“样本2-预测-2”甚至将整个图像都预测为背景，然而它们的平均绝对值误差均小于“样本1-预测1”–仅仅是稍微漏掉了狮子头部的一小部分。从这个例子可以看出，在显著性占比对比悬殊的情况下，平均绝对误差（MAE）并不能很好地评价显著性检测的质量。

然后我们分别定义准确率（precision）和召回率（recall）为：

$$\begin{aligned} p(\hat{Y},Y) &= \frac{TP}{TP+FP}, \\ r(\hat{Y},Y) &= \frac{TP}{TP+FN}. \end{aligned} \tag{3.4}$$

单一的准确率和召回率对模型性能的评价都十分片面，因为准确率和召回率是一对相互矛盾的评价指标：准确率上升意味着召回率下降，而召回率上升意味着准确率下降。如 图 3.10所示，对于同一个连续的网络输出 $\hat{Y}$，使用不同的阈值 τ 能得到不同的二值显著性图。随着阈值 τ 增大（从左到右），二值显著性图的前景区域变小，准确率上升，召回率上升。

在实践中，我们往往用准确率和召回率的调和平均值–F-measure–来评价显

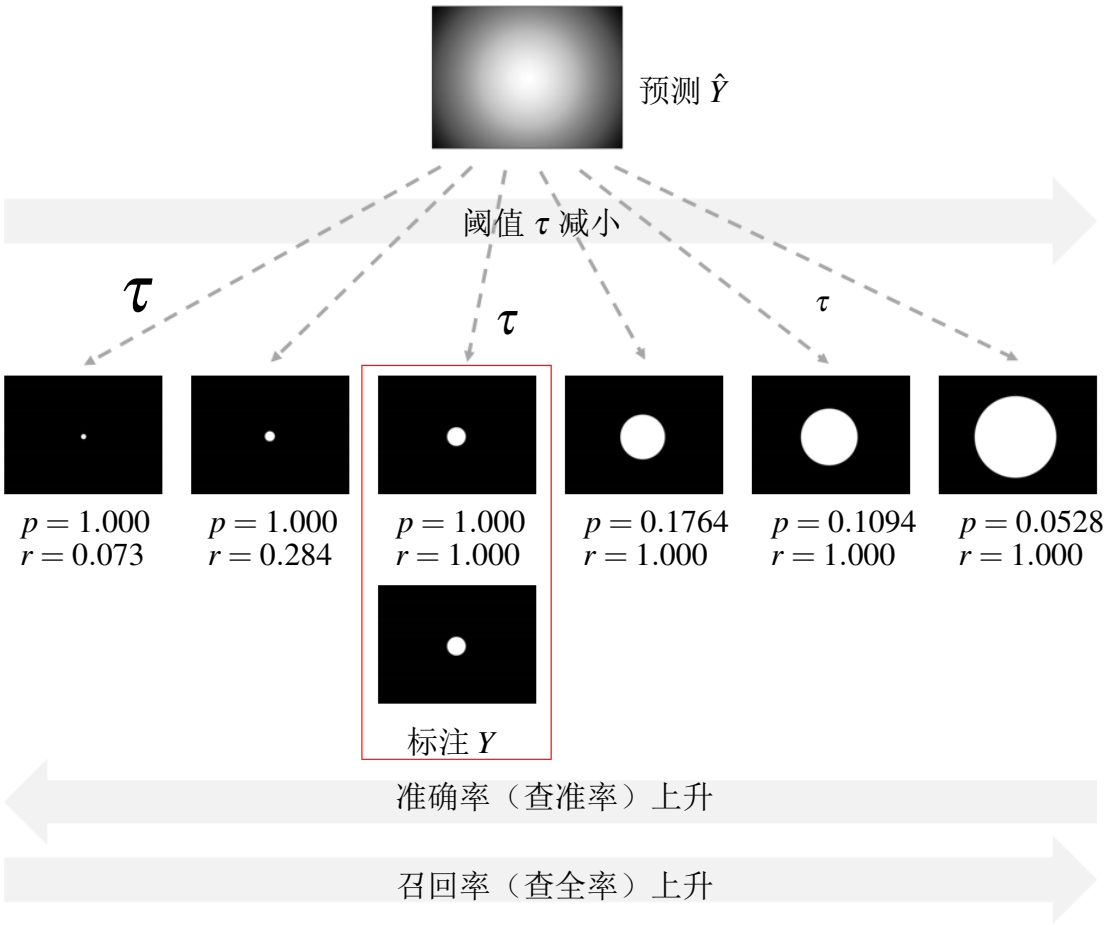


图 3.10 对于同一个预测 $\hat{Y}$，使用不同的阈值 τ 二值化，能得到不同的二值显著性图 $\hat{Y}^{\tau}$，它们的准确率与召回率各不相同。随着阈值 τ 增大（从左到右），二值显著性图的前景区域变小，准确率上升，召回率上升。因此，准确率和召回率是一对相互矛盾的评价指标。

著性图的质量。 F-measure的定义为：

$$\begin{aligned} F(\hat{Y},Y) &= \frac{(1+\beta^2)p\cdot r}{\beta^2 p + r}, \\ &= \frac{(1+\beta^2)TP}{\beta^2(TP+FN)+(TP+FP)}, \\ &= \frac{(1+\beta^2)TP}{H}, \end{aligned} \tag{3.5}$$

公式(3.5)中 $\beta^2 > 0$ 是一个超参数，当 $\beta^2 > 1$ 时，F-measure 更倾向于召回率；当 $0 < \beta^2 < 1$ 时，F-measure 更倾向于准确率。目前的显著性检测方法普遍使用 $\beta^2 = 0.3$ [1, 48, 74, 82, 134, 52, 160]。

在实际中，现有的方法都是比较的在某个数据集上的最大 F-measure值（Max F-measure）。当得到模型在某个数据集所有的测试图片上的预测结果 $\{\hat{Y}_1,\hat{Y}_2,...,\hat{Y}_N\}$ 之后，此时的 F-measure是一个只跟二值化阈值 τ 有关的函数。根据 公式(3.5)当 τ 很大或者很小时都不利于 F-measure。因此最后的最大 F-measure（Max F-measure）为：

$$MaxF = \max_{\tau} F(\{\hat{Y}_1,\hat{Y}_2,...,\hat{Y}_N\},\{Y_1,Y_2,...,Y_N\}). \tag{3.6}$$

此时的阈值称为最佳阈值：

$$\tau^* = \operatorname*{argmax}_{\tau} F(\{\hat{Y}_1,\hat{Y}_2,...,\hat{Y}_N\},\{Y_1,Y_2,...,Y_N\}). \tag{3.7}$$

第五节 基于放缩 F-measure 损失函数的显著性物体检测

3.5.1 交叉熵损失在显著性检测中的缺点

目前基于深度学习的显著性检测方法在训练阶段普遍是用交叉熵损失（cross entropy loss）作为损失函数。假设 $\hat{Y} \in (0,1)^{H\times W}$ 为模型的输出，$Y \in \{0,1\}^{H\times W}$ 为样本的标注，交叉熵损失为：

$$\mathcal{L}_{CE}(\hat{Y},Y) = -\sum_i^{|Y|}\left(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right), \tag{3.8}$$

公式(3.8)与 公式(2.9)中的定义有一些不一样。因为在分类任务中，每一个图片对应一个输出的向量，表示图片属于每一类的概率。而在显著性检测中，每一个像素都可以认为是一个样本，对应一个输出。并且显著性检测是一个二分类问题，因此对于每个像素模型只需要输出一个值 y_i 表示该像素属于显著性区域的概率，而该像素点属于背景的概率则为 $1-y_1$。

本文指出，使用 公式(3.8)中的损失函数训练显著性物体检测模型可能会有以下几个问题：

- 首先，我们在训练过程中最小化模型输出与样本标注之间的交叉熵，而测试的时候却以输出结果的 F-measure值作为主要衡量标准，在训练阶段和测试阶段的优化目标不一致；

- 其次， 公式(3.8)中的交叉熵损失在饱和区域（预测值与标注很接近时）会出现梯度下降现象，导致模型在物体边界的预测值偏小，边界不清晰，对比度低；

- 由于在优化空间中的局部区域梯度过小，交叉熵损失收敛速度也比较慢
- 最后，由于显著性检测训练数据集中普遍存在显著性区域占比过小的问题，这样训练出来的模型具有显著的偏见（bias），会倾向于将一个未知的像素预测为背景，导致最后在评测时召回率称为制约模型性能的主要因素，我们将在后文中用实验证明这点。

3.5.2 基于放缩 F-measure 的损失函数

本节我们介绍本文所提出的基于放缩 *F-measure* 的损失函数。首先介绍标准 F-measure 的不可微分性，然后通过对标准 F-measure 进行一个近似的放缩，得到一个放缩的可导的 F-measure 函数，并定义了两个基于可导 F-measure 的损失函数。

标准 ***F-measure*** 的不可导性

在标准的 F-measure 中，True Positive，False Positive 和 False Negative 定义为对应像素点的**个数**：

$$\begin{aligned} TP(\hat{Y}^{\tau},Y) &= \sum_i 1(y_i == 1 \text{ and } \hat{y}^{\tau} == 1), \\ FP(\hat{Y}^{\tau},Y) &= \sum_i 1(y_i == 0 \text{ and } \hat{y}^{\tau} == 1), \\ FN(\hat{Y}^{\tau},Y) &= \sum_i 1(y_i == 1 \text{ and } \hat{y}^{\tau} == 0). \end{aligned} \tag{3.9}$$

其中 Y 是样本标注，$\hat{Y}^{\tau}$ 是模型输出 $\hat{Y}$ 经过阈值 τ 二值化之后的显著性图。$1(\cdot)$ 是一个示性函数，只有条件满足的时候为 1，否则为0. 根据 公式(3.9)中的 TP, FP, FN，结合 公式(3.5)可得到标准 F-measure。

由于反向传导算法依赖于损失函数产生的梯度，因此损失函数必须可导。但是，由于**二值化**这个操作，标准的 F-measure 对网络输出 $\hat{Y}$ 的导数不存在。

为了能够在神经网络中直接优化 F-measure，我们必须对 F-measure 做一个连续可导的逼近，构造一个连续可导的函数来近似 F-measure。基于这个出发点，我们重新定义了 True Positive，False Positive 和 False Negative：

$$\begin{aligned} TP(\hat{Y},Y) &= \sum_i \hat{y}_i \cdot y_i, \\ FP(\hat{Y},Y) &= \sum_i \hat{y}_i \cdot (1-y_i), \\ FN(\hat{Y},Y) &= \sum_i (1-\hat{y}_i) \cdot y_i \ . \end{aligned} \tag{3.10}$$

公式(3.10) 是对 公式(3.9) 的一个逼近与 公式(3.9)不同的是， 公式(3.10) 中并没有用到二值化之后的显著图，而是直接使用网络的输出 $\hat{Y}$ 计算。因此 公式(3.10)中的 TP，FP 和 FN 对 $\hat{Y}$ 的导数是存在的。

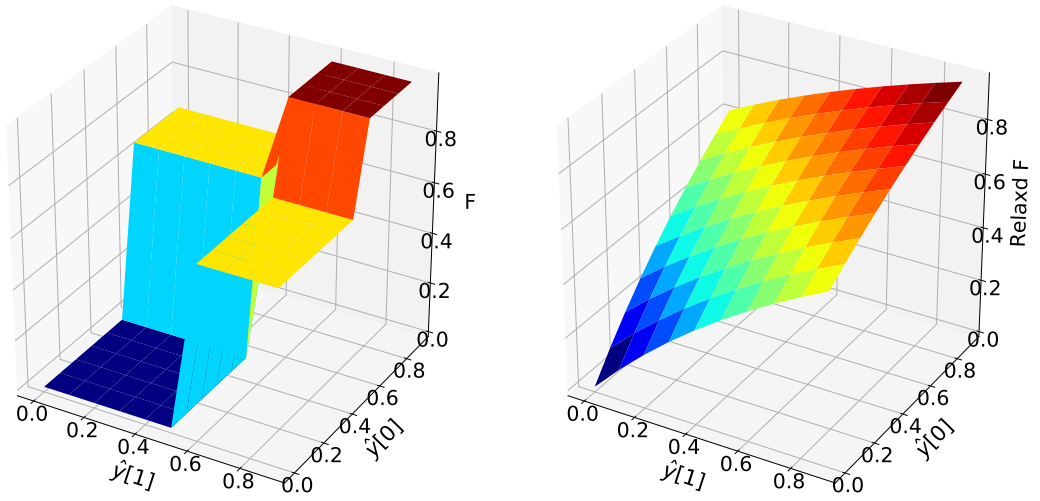
进一步的，根据 公式(3.10)中的 TP，FP 和 FN，我们定义准确率（precision）和召回率（recall）：

$$p(\hat{Y},Y)=\frac{TP}{TP+FP},\quad r(\hat{Y},Y)=\frac{TP}{TP+FN}. \tag{3.11}$$

最后，放缩的 F-measure 为：

$$\begin{aligned} F(\hat{Y},Y)&=\frac{(1+\beta^2)p\cdot r}{\beta^2 p+r},\\ &=\frac{(1+\beta^2)TP}{\beta^2(TP+FN)+(TP+FP)},\\ &=\frac{(1+\beta^2)TP}{H}, \end{aligned} \tag{3.12}$$

其中 $H=\beta^2(TP+FN)+(TP+FP)$。由于 公式(3.10) 是对标准形式 公式(3.9)的逼近，因此 公式(3.12) 中的 F-measure也是对标准 F-measure 的一种逼近。图 3.11 描述了两个像素情况下 Relaxed F-measure 对 F-measure 的逼近情况。



(a) (b)

图 3.11 当两个像素的标注为[0,0]时预测值 $\hat{Y}[0],\hat{Y}[1]$ 的值和 F-measure的变化图。左边：标准的 F-measure（F）；右边：本文所提出的放缩的 F-measure（Relaxed F）。

图中两个像素的标注均为0，也就是说这两个点均为背景，标准 F-measure 二值

化时使用的阈值 $\tau = 0.5$。从图中可以看出，标准 F-measure 随着预测 $\hat{Y}$ 的变化是阶跃的，不连续不可导的。而本文所提出的放缩的 F-measure（Relaxed F， 公式(3.12)）随着预测的变化连续变化，且变化趋势和标准 F-measure 一致。

进一步，为了在网络训练中能够最大化 F-measure，我们定义了基于缩放的 F-measure 的损失函数（FLoss）为：

$$\mathcal{L}_F(\hat{Y}, Y) = 1 - F = 1 - \frac{(1+\beta^2)TP}{H}. \tag{3.13}$$

同时，我们也定义了另一种基于 F-measure 的对数函数的损失函数，简称为 LogFLoss：

$$\mathcal{L}_{\log F}(\hat{Y}, Y) = -\log(F), \tag{3.14}$$

之后我们会从理论和实验对比两个方面证明 FLoss 比 LogFLoss 更能促进模型产生高对比的显著性检测结果。

3.5.3 损失函数的梯度对比

根据 公式(3.13)，FLoss 对网络预测 $\hat{Y}$ 的梯度为：

$$\begin{aligned}
\frac{\partial \mathcal{L}_F}{\partial \hat{y}_i} &= -\frac{\partial F}{\partial \hat{y}_i} \\
&= -\left(\frac{\partial F}{\partial TP} \cdot \frac{\partial TP}{\partial \hat{y}_i} + \frac{\partial F}{\partial H} \cdot \frac{\partial H}{\partial \hat{y}_i}\right) \\
&= -\left(\frac{(1+\beta^2)y_i}{H} - \frac{(1+\beta^2)TP}{H^2}\right) \\
&= \frac{(1+\beta^2)TP}{H^2} - \frac{(1+\beta^2)y_i}{H}.
\end{aligned} \tag{3.15}$$

而 LogFLoss 对网络预测 $\hat{Y}$ 的梯度为：

$$\frac{\partial \mathcal{L}_{\log F}}{\partial \hat{y}_i} = \frac{1}{F}\left[\frac{(1+\beta^2)TP}{H^2} - \frac{(1+\beta^2)y_i}{H}\right]. \tag{3.16}$$

另外，交叉熵损失 $\mathcal{L}_{CE}$ 对网络输出 $\hat{Y}$ 的梯度为：

$$\frac{\partial \mathcal{L}_{CE}}{\partial \hat{y}_i} = \frac{y_i}{\hat{y}_i} - \frac{1-y_i}{1-\hat{y}_i}. \tag{3.17}$$

下面我们仍然以一个两点的显著性检测为例，对比不同损失函数的梯度。假设我们输入的图片只有两个像素，我们在一个二维平面中画出不同输出 $\hat{y}[0], \hat{y}[1]$ 的情况下不同损失函数的值，从损失函数曲面的陡峭我们可以看出各损失函数的梯度情况。

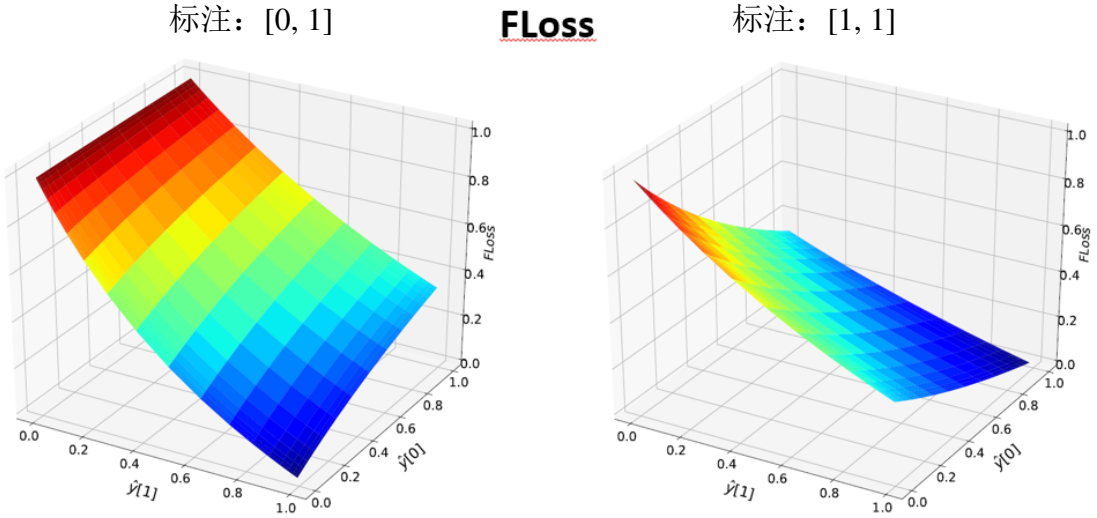


图 3.12 两个像素的情况下模型输出 $\hat{y}[0],\hat{y}[1]$ 与损失函数的关系。左边两个像素的真实标注为 [0, 1]，右边的为 [1, 1]。可以看出，当模型预测趋向于标注的时候，FLoss（公式(3.13)）仍然有很大的梯度，这回迫使模型产生更加靠近标注的预测值，使得显著性检测结果对比度更鲜明。

图 3.12表明，Floss（公式(3.13)）即便在模型预测趋向于标注时仍然具有相当的梯度，因此可以持续促使模型产生更靠近标注（0或1）的输出。也就是说，Floss训练得到的显著性图具有**对比度高**的特点。我们将在下一节中用实验证明这个结论。

图 3.13和 图 3.14说明，当预测 $\hat{Y}$ 靠近标注 Y 时，交叉熵损失和 Log-Floss都出现一定程度的**梯度消失**现象。在这种情况下，$\hat{Y}$ 失去继续靠近 Y 的动力，导致得到的显著性图在物体边缘地带置信度不高，对比度不明显。

第六节 实验验证

本节我们用实验数据证明本文所提出的 Floss的有效性。我们测试了3种不同的深度学习显著性检测模型，用我们所提出的损失函数替换原始模型中的交叉熵损失。在5个公开数据集上的评测表明，本文所提出的基于放缩F-measure的损失函数（Floss）能够显著提升显著性检测模型的性能，同时加快模型收敛速度。更重要的是，本文所提出的方法得到的显著性图不需要任何后续处理就具有十分高的对比度，有较大的实际应用价值。

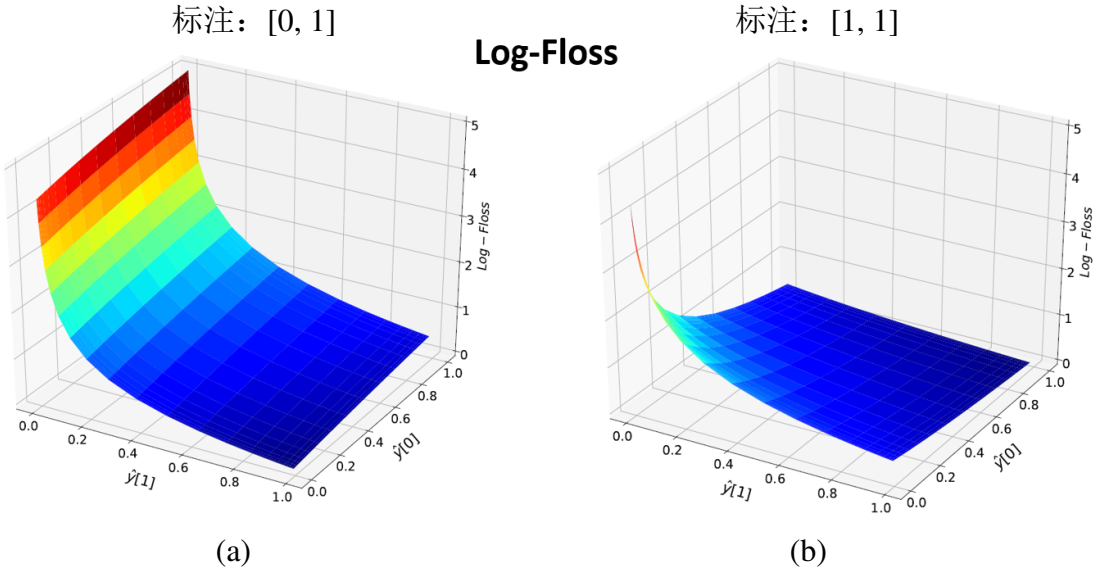


图 3.13 两个像素的情况下模型输出 $\hat{y}[0],\hat{y}[1]$ 与损失函数的关系。左边两个像素的真实标注为 [0, 1]，右边的为 [1, 1]。当模型输出趋近于标注时，Log-Floss（公式(3.14)）的变化变得平缓，损失函数梯度变小甚至消失。体现在输出结果上，就是在一些物体的边缘区域模型会输出不够置信的显著性得分，使得整体显著性图对比度不够鲜明。

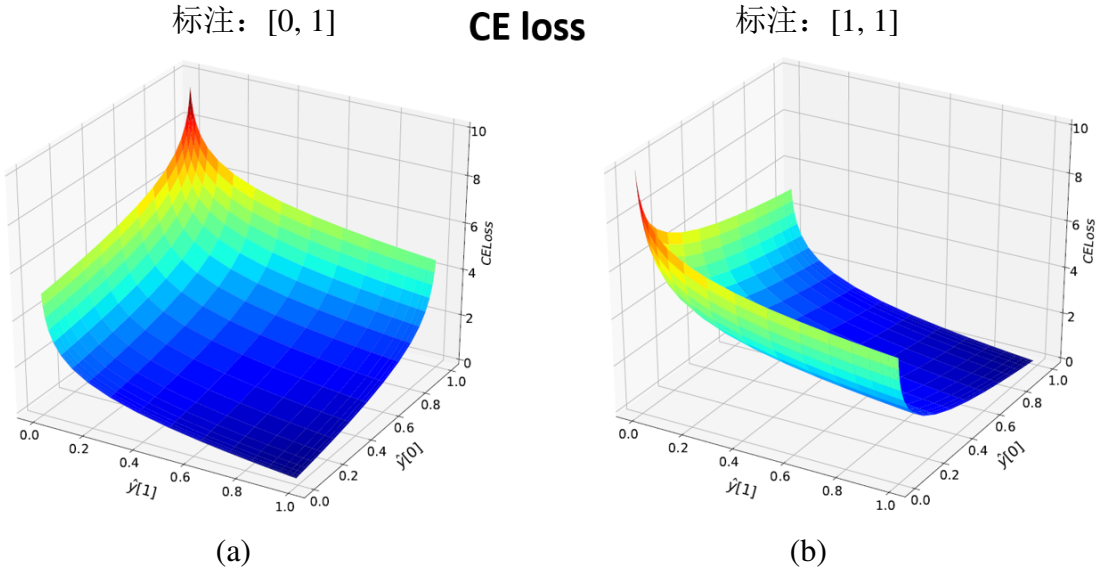


图 3.14 两个像素的情况下模型输出 $\hat{y}[0],\hat{y}[1]$ 与损失函数的关系。左边两个像素的真实标注为 [0, 1]，右边的为 [1, 1]。和 Log-Floss 公式(3.14)相似，当模型输出趋近于标注时，交叉熵损失（公式(3.8)）的变化变得平缓，损失函数梯度变小甚至消失。体现在输出结果上，就是在一些物体的边缘区域模型会输出不够置信的显著性得分，使得整体显著性图对比度不够鲜明。

3.6.1 实验配置和环境

数据集和数据增强 本章的实验环境为一个装有 Titan-XP 显卡的服务器，CPU 为 Intel core i7，操作系统为 Ubuntu 16.04。为了公平比较，在本节的实验中，所有的模型，包括本文所提出的方法及其他对比模型，都使用 MSRA-B [84]数据集作为训练集。 MSRA-B数据集有5000张图片，他们被均匀地分为训练集和测试集。我们在五个数据集上测试模型的性能： ECSSD [142], HKU-IS [75], PASCALS [78], SOD [96], and DUT-OMRON [96]. 更多关于数据集的介绍请参考本章的 第三节和 表格 3.1。

数据增强对深度学习模型的性能有很大的作用。对于所有的模型，我们使用相同的数据增强策略。对于 DSS [48] 和 DHS [82]，我们仅仅左右翻转图片（和标注）。 Amulet [153] 只支持 256×256 的图片输入，因此我们随机地裁剪数据集中的原始图片，然后 resize到固定尺寸，最后输入到模型中。

网络结构和超参数 本章的实验在三个不同的深度学习显著性检测模型上测试了所提出的损失函数，这三个模型分别是 Amulet [153], DHS [84] 和 DSS [48]. 为了验证 FLoss (Eq. 3.13)的有效性，, 本文仅将原始方法中的交叉熵损失替换为本文新提出的损失函数，其它的所有配置保持不变。在上一节曾提到，相比交叉熵损失，Floss的允许更大的学习率，因此在实验中使用 Floss训练时学习率为对应交叉熵损失的 10^4 倍。例如，DSS模型在使用交叉熵损失训练时学习率为 10^{-8}，当用 Floss训练时候，学习率调整为 10^{-4}。为了公平比较，其它的所有超参数均和原文保持一致。

评测指标我们使用最大 F-measure (MaxF), 平均 F-measure (MeanF) 和平均绝对误差 (MAE = $\frac{1}{N}\sum_i^N |\hat{y}_i - y_i|$)和评测不同显著性检测结果的性能。关于这些评测指标的详细说明请参考本章的 第四节。

3.6.2 Floss 和 Log-Floss

本章的 3.5.3中的损失函数梯度分析指出，由于在饱和区域的高梯度特性，Floss能够取得比 Log-Floss对比度更加强的显著性检测结果，本节用实验证明这一结论。本节分别使用 Floss 公式(3.13)和 Log-Floss 公式(3.14)训练 DSS [48]模型。 图 3.15是 Floss 和 Log-Floss 两种损失函数训练出的模型在 ECSSD数据集 [142]上的部分测试结果， 表格 3.2 是两种损失函数在 ECSSD 数据集上的定量评测结果。 从 figreffig:examples-floss-vs-logfloss可以明显的看出，Floss得到的

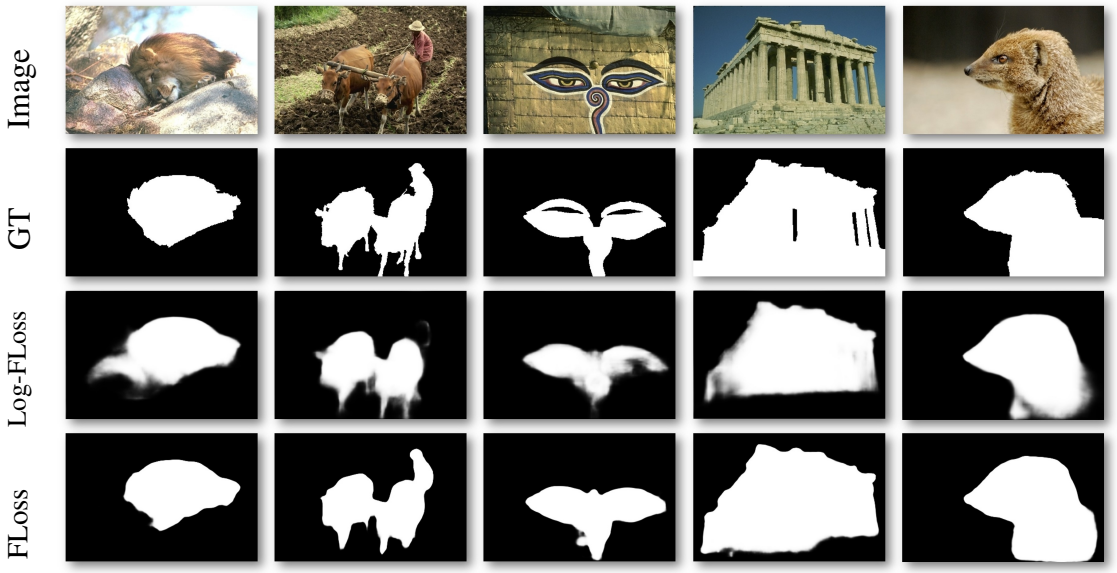


图 3.15 Floss 和 Log-Floss 两种损失函数得到的显著性检测结果对比。 Floss的结果边界更清晰，对比度更高。

显著性图在土体边缘部位具有很强的对比度，几乎没有置信度低的过渡带。而同样的模型，使用 Log-Floss训练得到的结果在一些物体的边缘处比较模糊，对比度不高。

表格 3.2对比了 Floss和 Log-Floss两种损失函数在多个数据集上的定量评测结果。 表格 3.2的结果同样表明，Floss能取得比 Log-Floss更好的结果，特别是

	Training data		ECSSD [142]			HKU-IS [75]			PASCALS [78]			SOD [96]			DUT-OMRON [96]		
Model	Train	#Images	MaxF	MeanF	MAE	MaxF	MeanF	MAE	MaxF	MeanF	MAE	MaxF	MeanF	MAE	MaxF	MeanF	MAE
Log-FLoss	MB [84]	2.5K	.909	.891	.057	.903	.881	.043	.823	.808	.101	.838	.817	.122	.770	.741	.062
FLoss	MB [84]	2.5K	.914	.903	.050	.908	.896	.038	.829	.818	.091	.843	.838	.111	.777	.755	.067

表 3.2 Log-FLoss (公式(3.14)) 和 FLoss (公式(3.13)) 在 ECSSD 数据集上的性能对比。 Floss取得比 Log-Floss 更高的 F-measure 和更小的平均绝对误差（MAE）。

在平均 F-measure（MeanF）评测指标下，Floss性能优势十分明显。

3.6.3 在公开数据集上的实验对比

本节对比了 Floss和交叉熵损失在3个不同的显著性检测模型，5个不同的显著性检测数据集上的性能比较。 表格 3.3和 表格 3.4记录了本文所提出的 Floss和交叉熵损失在三种不同显著性检测模型，五个不同的显著性检测数据集

上的性能比较。 表格 3.3和 表格 3.4的结果表明，本文所提出的 Floss能显著提

	Training data		ECSSD [142]			HKU-IS [75]		
Model	Train	#Images	MaxF	MeanF	MAE	MaxF	MeanF	MAE
RFCN [134]	MK [14]	10K	.898	.842	.095	.895	.830	.078
DCL [74]	MB [84]	2.5K	.897	.847	.077	.893	.837	.063
DHS [82]	MK [14]+D [96]	9.5K	.905	.876	.066	.891	.860	.059
Amulet [153]	MK [14]	10K	.912	.898	.059	.889	.873	.052
DHS [82]	MB	2.5K	.874	.867	.074	.835	.829	.071
DHS+FLoss [82]	MB	2.5K	**.884**	**.879**	**.067**	**.859**	**.854**	**.061**
Amulet [153]	MB	2.5K	.881	.857	.076	.868	.837	.061
Amulet-FLoss	MB	2.5K	**.894**	**.883**	**.063**	**.880**	**.866**	**.051**
DSS [48]	MB	2.5K	.908	.889	.060	.899	.877	.048
DSS+FLoss	MB	2.5K	**.914**	**.903**	**.050**	**.908**	**.896**	**.038**

表 3.3 不同显著性检测方法在 ECSSD [142]和 HKU-IS [75]数据集上的比较。

	Training data		PASCALS [78]			SOD [96]			DUT-OMRON [96]		
Model	Train	#Images	MaxF	MeanF	MAE	MaxF	MeanF	MAE	MaxF	MeanF	MAE
RFCN [134]	MK [14]	10K	.829	.784	.118	.807	.748	.161	-	-	-
DCL [74]	MB [84]	2.5K	.807	.761	.115	.833	.780	.131	.733	.690	.095
DHS [82]	MK [14]+D [96]	9.5K	.820	.794	.101	.819	.793	.136	-	-	-
Amulet [153]	MK [14]	10K	.828	.813	.092	.801	.780	.146	.737	.719	.083
DHS [82]	MB	2.5K	.782	.777	.114	.800	.789	.140	.704	.696	**.078**
DHS+FLoss [82]	MB	2.5K	**.792**	**.786**	**.107**	**.801**	**.795**	**.138**	**.707**	**.701**	.079
Amulet [153]	MB	2.5K .775	.753	.125	.791	.776	.149	.704	.663	.098	
Amulet-FLoss	MB	2.5K	**.791**	**.776**	**.115**	**.805**	**.800**	**.138**	**.729**	**.696**	**.097**
DSS [48]	MB	2.5K	.824	.806	.099	.835	.815	.125	.761	.738	.071
DSS+FLoss	MB	2.5K	**.829**	**.818**	**.091**	**.843**	**.838**	**.111**	**.777**	**.755**	**.067**

表 3.4 不同显著性检测方法在 PASCALS [78]、SOD [96]和 DUT-OMRON [96] 三个数据集上的定量比较结果。

升基于交叉熵损失的显著性检测模型的性能。特别是当用平均 F-measure作为评价指标时，Floss对原始方法的相对性能提高有的高达5%。在平均绝对误差（MAE）指标下，Floss也有非常明显的性能提高。

图 3.16展示了几个在不同测试数据集上检测结果的可视化对比，本文所提出的 Floss 得到的显著图相比基于交叉熵损失的模型具有对比度高的特点。

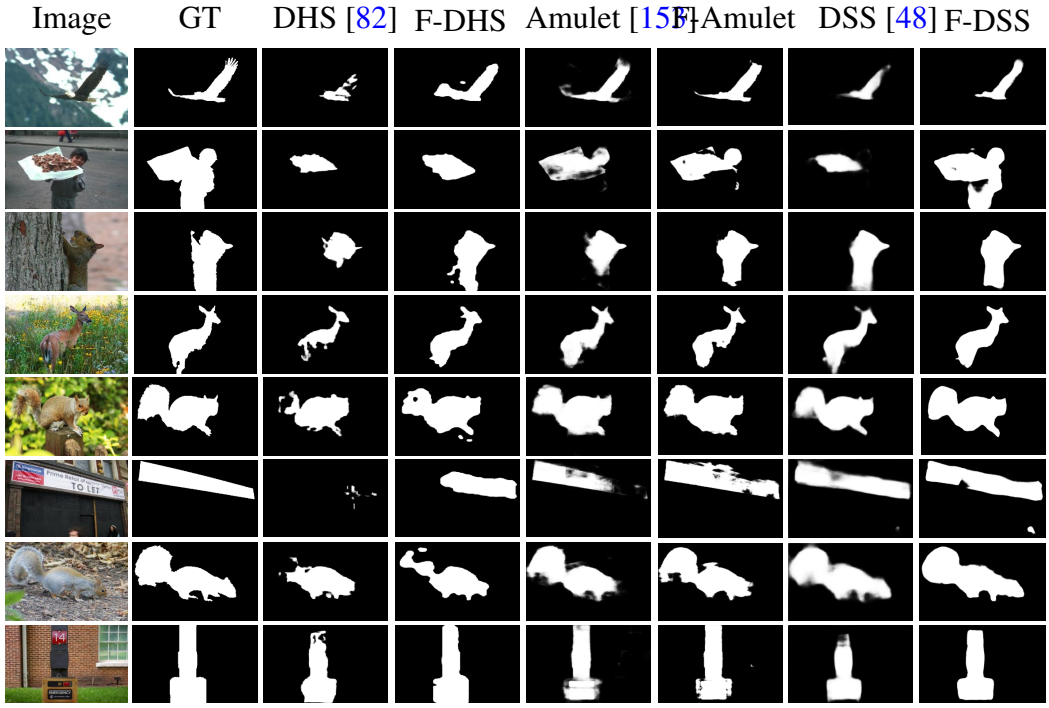
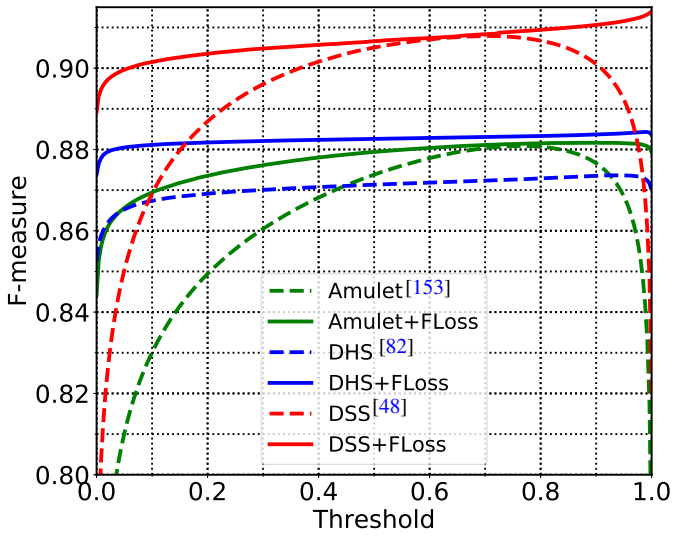


图 3.16 基于 Floss 和交叉熵损失的显著性检测结果对比。 Floss 产生的显著图对比更鲜明，在物体边界处更清晰。

3.6.4 对阈值的敏感度对比

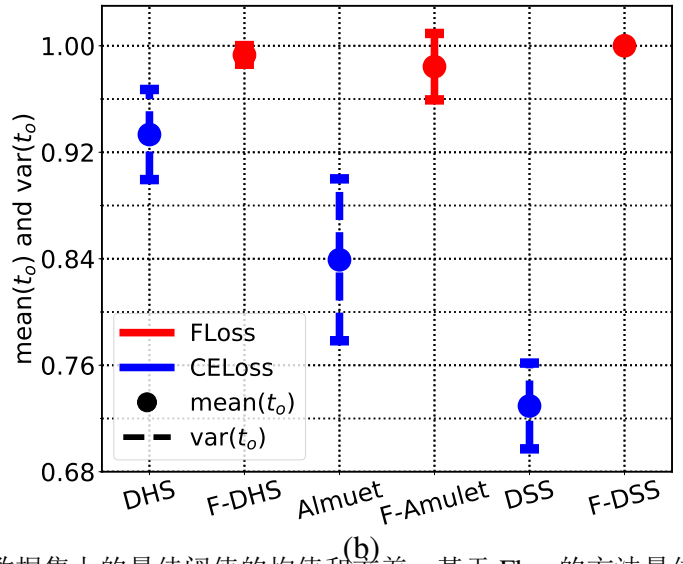
目前的一些主流显著性检测算法均是在测试集上找到一个最佳的二值化阈值 τ，最后用该阈值产生二值化的显著性图进行评测，计算出最佳的 F-measure。在测试集上调试得到超参数有可能在其他数据集并不是最优的。换句话说，现有的一些显著性检测方法可能对二值化阈值的选取很不稳定。本文所提出的基于 F-measure的损失函数的显著性检测方法对阈值十分鲁棒，这体现在：

- 本文所提出的方法在不同阈值 τ 均能取得良好的性能；
- 在一个数据集上找到的最佳阈值在其他数据集上也有很好的性能，换句话



(a)

图 3.17 不同阈值情况下模型的性能（用 F-measure衡量）。基于 Floss的方法的性能几乎不受阈值选取的影响，在不同阈值下都有很高的 F-measure。



(b)

图 3.18 在不同数据集上的最佳阈值的均值和方差。基于 Floss的方法最佳阈值的方差十分小，这说明用 Floss训练出的模型在不同数据集上的最佳阈值很接近，很容易互相迁移。

说，不同数据集上的最佳阈值很相似。

首先 图 3.17显示了在不同二值化阈值 τ 下各模型检测结果的 F-measure）。从图中我们可以看出，使用 Floss训练的模型对阈值的选取十分鲁棒，模型的性能几乎不受阈值的影响

其次， 图 3.18统计了不同方法在不同数据集上最佳阈值的分布情况。其中

条状图的长度表示同一个模型在不同数据集上最佳阈值的方差，越长说明方法越大，该方法在不同数据集上的最佳阈值差异大。条状图的中心表示在不同数据集上最佳阈值的均值。

从 图 3.18很容易得出结论：基于 Floss的显著性检测模型在不同数据集上的最佳阈值几乎不变。甚至不同的网络结构，只要是使用本文所提出的 Floss训练，在各数据集上的最佳阈值都十分接近。 图 3.17和 图 3.18的实验结果表明，相比交叉熵损失，本文所提出的方法对阈值的选取十分鲁棒，具体体现在以下三个方面：

- 对同一个模型在单个数据集上，模型的性能几乎不受阈值的影响；
- 对一个模型在不同数据集上的最佳阈值方差很小，因此在一个数据集上找到的最佳阈值可以很容易迁移到其他数据集上；
- 基于不同网络结构的模型，只要使用 Floss 作为损失函数，在各数据集上的最佳阈值就比较相近。

第七节 小结

本章提出了基于“放缩 F-measure”的损失函数并将该损失函数应用到显著性检测任务中。在多个公开数据集上基于多个主流深度学习显著性检测模型的评测结果表明，本文所提出的损失函数能够取得比广泛使用的交叉熵损失更好的性能。而且本章所提出的损失函数能够在不使用条件随机场等后处理技术的情况下产生前背景对比度十分鲜明的显著性图，具有很大的实用前景。

在未来的工作中，作者计划将该损失函数应用到图像中倒影检测，基于深度图的显著性检测等逐像素二分类任务中。同时也可以将该损失函数的拓展到多类的情况下并运用到语义分割等逐像素的多分类任务中。本章节相关内容已经发表在计算机视觉领域的国际顶级会议 IEEE ICCV2019（IEEE International Conference on Computer Vision）上。

第四章 互斥正则损失函数

本章主提出了“互斥正则损失函数”，该损失函数能够在训练过程中显示地增大不同类别样本特征之间的距离，从而促进所学到的特征的“类间离散度”。在人脸识别任务上的实验证明，本章所提出的损失函数能够有效增强模型对不同人脸的区分度，提升人脸检测的准确度。

本章第一节节首先介绍人脸识别相关的背景知识，第二节节介绍了本章的研究动机，主要包含以往方法大多将注意力集中在“类内紧凑度”上，而忽略了“类间离散度”，因此本文提出新的损失函数用于增强“类间离散度”。第三节介绍相关工作，主要是之前以“类内紧凑度”为核心的损失函数设计相关的工作。第四节详细推导了本章所提出的互斥正则损失函数。第五节用实验论证了所提出方法的有效性，第六节对全章做了简单总结。

第一节 引言

4.1.1 背景知识

由于在安防、反恐、电子身份认证等领域的广泛运用，近年来人脸识别是计算机视觉领域的研究热点之一。通常根据使用场景的不同，人脸识别可以分为以下两个类别：

- 限制条件下人脸识别；
- 非限制条件下人脸识别。

限制条件下人脸识别场景下，输入一张人脸照片，模型输出的该照片对应数据库中的哪一个个体（identity，通俗讲就是对应哪一个人），也就是说限制条件下人脸识别在训练和测试时的的身份是重合的，测试图片对应的个体要求属于训练样本的个体集合。非限制条件下人脸识别对训练和测试样本没有任何限制。模型在训练完成之后，输入一对人脸照片，要求模型判断这两张照片是否属于同一个人。由于非限制性人脸识别对数据的要求更低，对场景的限制更小，更容易应用到实际生活中。本章主要研究非限制条件下的人脸识别，除非特别说明，本文所说的“人脸识别”均指的是“非限制条件下人脸识别”。

如 图 4.1所示，一个典型的人脸识别系统工作流程如下：首先采集人脸图

片，并检测图片中的人脸关键点。然后根据检测到的关键点校正图片中的人脸姿态，使人脸的关键点（双眼，鼻子，嘴角）在固定的位置。最后，使用特征提取模型提取校正后的人脸照片的人脸特征，并根据所提取的特征输出识别结果。

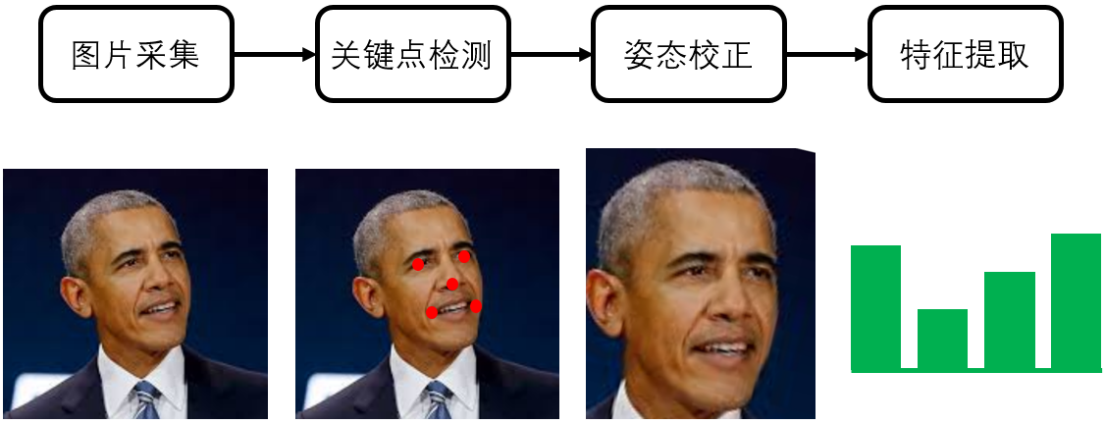


图 4.1 人脸识别中处理输入图片的基本流程：首先采集图片并定位输入图片中人脸的关键点，然后根据定位到的关键点校正人脸。最后将标准姿态下的人脸图片输入到模型中得到人脸特征，最后根据人脸特征确定识别结果。

人脸识别系统最后需要确定输入的两张人脸图片是不是来自同一个个体（identity），因此目前普遍的做法是在一个度量空间计算两个图片的特征之间的距离，如果特征距离小于某个阈值，则认为两张图片来自同一个个体，否则来自不同的个体。与分类任务不同的是，人脸识别并不强调分类的准确性，反而对特征的“区分度”有很高的要求。因为人脸识别的核心问题就是确定输入的两张照片是否属于同一个个体（identity），这就要求人脸特征具有高度的可区分性，这体现在两点：

- 同一个个体的不同照片的特征要尽可能接近；
- 不同个体的人脸照片的特征要尽可能远离。

由于目前很多大型的人脸识别数据集例如 CASIA-Webface [145]、MS-CELE [32]数据集都是通过从互联网上使用人名作为关键词搜索和下载名人照片来收集数据的。这些数据集和图像分类数据集的标注形式并没有区别，都是某个类别有若干张图片，或者是收集到某个名人的若干张人脸照片。因此目前有很多基于深度学习的人脸识别方法和分类模型一样，使用交叉熵损失来训练模型。虽然使用交叉熵损失也可以训练模型并得到性能十分优异的人脸识别模型，

但是交缠上损失本身并不强调特征的“区分度”，而是单纯地在降低分类本身的误差。因此，使用交叉熵损失未必是人脸识别算法中的最好的损失函数。

近年来有很多研究尝试在交叉熵损失的基础上增加额外的损失项，以达到增强特征区分度的目的。2016年，Wen等人 [136]使用了一个额外的损失项来约束同一类别的样本特征离该类样本特征的中心的欧式距离，来增强特征的**类内紧凑度**。2017年，Liu等人则提出在角度空间使用一个乘子约束同一类样本和样本中心的角度，以达到在角度空间增强类内的紧凑度的目的。2019年，Deng等人 [20]提出用一个加性边界（additive margin）来约束同类样本在角度空间中的紧凑度，取得了很好地性能提升。

但是，几乎目前所有试图在交叉熵损失的基础上提升特征区分度的方法都只关注了“类内紧凑度”，也就是说只是让同类样本的特征在特征空间更加紧凑。而特征区分度的另外一个方面，“类间离散度”，也就是不同类别的样本之间最好是互相远离，却在目前的一些研究中被忽视了。

第二节 研究动机

目前很多基于交叉熵损失的改进方案都是提出了新的损失函数以提升特征的类内紧凑度，而忽视了类间离散度。本章的研究试图设计一种新的损失函数项，在该损失项的约束下使得不同类别的样本特征在特征空间尽可能彼此远离，以达到增强类间离散度和区分度的目的。

在基于交叉熵的分类网络中，我们使用网络倒数第二层的输出 $\mathbf{X} \in \mathbb{R}^K$ 为样本的特征向量，并将网络的最后一个全连阶层（fully connected layer）视为一个线性分类器，接收特征 $\mathbf{X}$ 并输出对应各类别的预测 $\mathbf{Y}$：

$$Y = W \cdot X.$$

其中 $W \in \mathbb{R}^{K,N}$ 为全连阶层的权重参数，K 为特征维度，N 为训练集类别总数。最后使用 softmax[1]函数将输出 y 转化为概率形式：

$$p_i = \frac{\exp(y_i)}{\sum_j \exp(y_j)}. \tag{4.1}$$

目前已经有一部分研究工作在交叉熵损失的基础上提出了一些改进以增强特征的区分度。但是这些工作都只强调了类内紧凑度对特征区分度的影响，忽略了类间离散度。

[1] https://en.wikipedia.org/wiki/Softmax_function

本文的研究动机在于通过对交叉熵损失进行改进，提出新的损失项来增加不同类别样本间的离散度。在分类层中，参数 W 的每一列 $W_k, k = 1, ..., K$ 可以当做每一个类别的中心，在交叉熵损失的优化下，样本特征将向类别中心方向移动。本章提出一种正则损失函数，通过惩罚类别中心间的距离，首先让不同类的类中心彼此远离。然后在交叉熵损失的共同作用下，不同类别的样本特征也将朝彼此远离的方向移动。

第三节 相关工作

本章主要关注深度学习的人脸检测方法损失函数设计，特别是基于交叉熵损失函数的改进方案。本节主要介绍一些有代表性的深度学习人脸识别方法，特别是有关人脸识别损失函数和正则损失项设计的一些工作。

4.3.1 基于度量学习的损失函数

由于人脸识别和度量学习（metric learning）有着相似的性质，都是为了让语义相似的样本在特征空间也互相靠近，因此有一些人脸识别方法直接借用了一些度量学习中的损失函数。

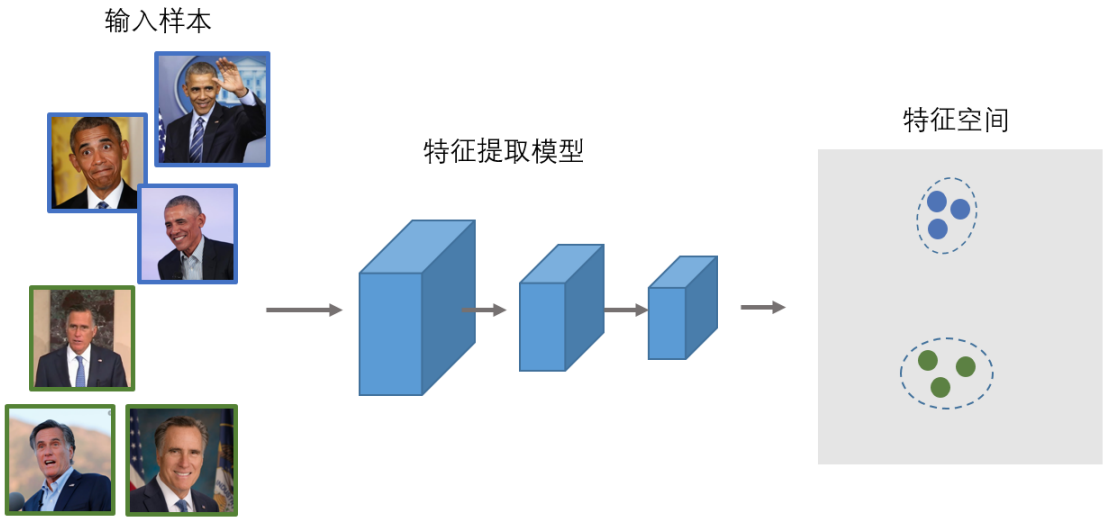


图 4.2 人脸识别（face recognition）和度量学习（metric learning）类似，都期望对于语义相同的样本（同一个人的人脸照片，纹理相似的图像块）经过特征提取之后在特征空间也彼此接近。

文献 [123, 124, 125, 146] 使用对比损失（contrastive loss）来训练深度人脸

识别模型。 Contrastive loss 的思想十分简单：如果一对样本语义类别相同，就让它们的特征距离更小；如果这对样本的语义类别不同，则让它们的特征距离增大。具体地，对于一对输入图片 $\{I_i, I_j\}$，contrastive loss 的定义为：

$$\begin{aligned}\mathcal{L}_{contrastive}(I_i, I_j) = &\mathbb{1}(y_i = y_j) \cdot \max(0, \left\| f(I_i) - f(i_j)) \right\|_2 - m^+) + \\ &(1 - \mathbb{1}(y_i = y_j)) \cdot \max(0, m^- - \left\| f(I_i) - f(i_j)) \right\|_2).\end{aligned} \tag{4.2}$$

其中 y_i, y_j 为图片的语义类别，$\mathbb{1}(\cdot)$ 为示性函数（indicator function），只有当条件成立时为 1，否则为 0。 $f(\cdot)$ 表示特征提取器，在基于深度学习的方法中则为深度卷积神经网络。 m^+ 和 m^- 分别是两个边界参数（margin），表示同类样本特征之间的最大距离和不同类样本之间的最小距离。 DeepID [123]方法同时使用 contrastive loss 和交叉熵分类损失来学习鲁棒的人脸识别特征。在 DeepID的基础上，DeepID2方法 [124]通过提高特征维度进一步提高了模型的性能。同时，受到深度监督网络（deeply supervised nets） [69]的启发，DeepID2给中间层特征也接上了损失函数，使得中间层的特征也得到了监督。 DeepID3 [125]通过使用更加先进的网络结构（GoogleNet [126]，VGGNet [116]）.

虽然 contrastive loss 具有十分直观的解释，通常性能也不错，但是边界参数（margins）m^+, m^- 对模型和数据敏感，难以找到合适的边界参数。再者，contrastive loss 的收敛依赖于训练数据对的选取，样本对的顺序、组合对模型收敛和性能也有影响。总的来说 contrastive loss 原理清晰，也有直观的解释，性能也不错，但是在训练中不够稳定，还有两个比较难调的超参数。

很多研究工作 [114, 102, 113, 113, 81, 23] 将度量学习中的另外一个损失函数，三元组损失函数（triplet loss），引入到人脸识别领域中。与 contrastive loss 不同，Triplet 损失函数 [47]接收三个样本，并根据三个样本间的相对距离来计算损失。假设 I, I^+, I_- 为三个样本组成的三元组，其中 I 为锚样本（anchor），I^+ 为正样本，I^- 为负样本。其中正样本 I^+ 与锚样本语义类别相同，负样本与锚样本类别不同：$y = y^+, y \neq y^-$。 Triplet loss 希望同一个三元组中锚样本和正样本的特征尽可能接近，而锚样本和负样本的特征彼此远离。 FaceNet [114]使用如下形式的 triplet loss：

$$\mathcal{L}_{triplet} = \max(0, \left\| f(I) - f(I^+) \right\|_2 - \left\| f(I) - f(I^-) \right\|_2 + m), \tag{4.3}$$

其中 $f(\cdot)$ 为特征提取器，$\|\cdot - \cdot\|_2$ 表示两点间的欧氏距离，m 是边界（margin）参数。在 FaceNet的基础上，文献 [113, 113] 用一个线性映射 W 对特征进行变

换，然后再计算 triplet loss，其中变换参数 W 是可以自动学习的。

三元组损失（triplet loss）和对比损失（contrastive loss）都面临这训练时的稳定性问题。其中一个重要的原因就是这两个损失函数的收敛都依赖于样本对（三元组）的选取。因此有一些工作尝通过挖掘数据集中的困难样本（hard examples） [139, 118] 来构建更鲁棒的三元组，使得损失函数收敛得更加稳定。

4.3.2 基于交叉熵的损失函数

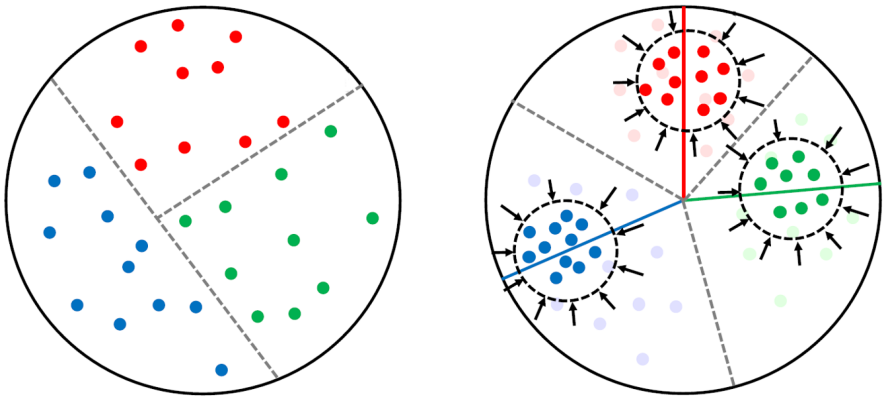
虽然交叉熵损失没有显式地增强特征的区分度，但是由于其收敛稳定、易于优化，所以有一部分方法试图以交叉熵损失为基础，增加额外的损失项来增强特征的区分度，这样既能利用交叉熵损失训练稳定的优点，又能得到区分度明显的特征。

不同的类别间间距（***margin***）

Center Loss. 2015年，Wen等人在交叉熵损失的基础上提出了“中心损失”，中心损失通过惩罚同类样本特征到该类别中心的距离，让各类样本向中心靠拢，最后得到类间紧凑的特征表示。假设 $\mathbb{X}_i$ 为样本特征，对应的类别标签为 y_i，中心损失定义为：

$$\mathcal{L}_{center} = \frac{1}{2} \cdot \left\| \mathbf{X}_i - \mathbf{C}_{y_i} \right\|_2 , \tag{4.4}$$

其中 $\mathbf{C}_{y_i}$ 为第 y_i 类样本特征的中心。



(a) (b)

图 4.3 二维空间的特征分布。 (a) 交叉熵损失得到特征可以用一条直线分开，保证分类准确度。 (b) 中心损失（center loss）通过惩罚样本到同类样本中心的距离，促成同类样本类间紧凑。

Large Marge Loss. 2016年，Liu等人 [85]提出了“大间距交叉熵损失”（large margin loss），通过在经过 softmax 函数后的后验概率上施加一个类间间距（margin），使得类间特征更加紧凑。同样，令 $\mathbf{X} \in \mathbb{R}^{\mathbf{K}}$ 为样本特征，$W \in \mathbb{R}^{K,N}$ 为线性分类层的特征，模型输出为：

$$\mathbf{Y} = \mathbf{X}^T \cdot W. \tag{4.5}$$

上式中的向量乘法可以转化为：

其中对应第 n 个类别的输出 $y_n = \mathbf{X}^T \cdot W_n$，其中 $W_n \in \mathbb{R}^K$ 为矩阵 W 的第 n 列。

$$y_n = \|\mathbf{X}\|_2 \|W_n\|_2 \cdot \cos(\theta_{\mathbf{X},W_n}) \tag{4.6}$$

最后通过 公式(4.1)中的 softmax 函数将输出转化为概率形式 p_n：

$$\begin{aligned} p_n &= \frac{\exp(y_n)}{\sum_j \exp(y_j)} \\ &= \frac{\exp(\|\mathbf{X}\|_2 \|W_n\|_2 \cdot \cos(\theta_{\mathbf{X},W_n}))}{\sum_j \exp(\|\mathbf{X}\|_2 \left\|W_j\right\|_2 \cdot \cos(\theta_{\mathbf{X},W_j}))} \end{aligned} \tag{4.7}$$

Large margin loss 通过在角度空间增加一个类间的间距来让类内的特征更加紧凑：

$$p_n = \frac{\exp(\|\mathbf{X}\|_2 \|W_n\|_2 \cdot \cos(\psi(\theta_{\mathbf{X},W_n})))}{\exp(\|\mathbf{X}\|_2 \|W_n\|_2 \cdot \cos(\psi(\theta_{\mathbf{X},W_n}))) + \sum_{j \neq n} \exp(\|\mathbf{X}\|_2 \left\|W_j\right\|_2 \cdot \cos(\theta_{\mathbf{X},W_j}))} \tag{4.8}$$

其中

$$\psi(\theta) = \cos(m \cdot \theta),$$

$m \geq 2$ 是一个整数。

公式(4.8)和 公式(4.7)的区别在于在对应类别的夹角上乘以一个大于1的整数，迫使特征与对应类别的权重向量 W_y 更加接近。由于交叉熵损失最后惩罚的是在对应类别上的概率 p_y:

$$\mathcal{L} = -\log(p_y),$$

所以为了在正确类别 y 上取得足够大的概率，相比标准的交叉熵损失，Large margin 损失必须保持 $\mathbf{X}$ 与权重向量 W_y 的夹角更小，因为夹角 $\theta_{\mathbf{X},W_y}$ 要乘上 m。

归一化特征和权重向量会规避这些因素带来的影响。

在 Large margin loss的基础上，Liu等人 [86]将每一类对应的权重向量 W_k 归一化到一个单位球上，这样类别之间的边界就转换到了一个单位球（unit

sphere）上，因此这个方法也被称为 Sphereface。 Sphereface 中样本 X 属于类别 n 的概率为：

$$p_n = \frac{\exp(\|\mathbf{X}\|_2 \cdot (\cos(m\theta_{\mathbf{X},W_n}) - m))}{\exp(\|\mathbf{X}\|_2 \cdot (\cos(m\theta_{\mathbf{X},W_n}))) + \sum_{j \neq n} \exp(\|\mathbf{X}\|_2 \cdot \cos(\theta_{\mathbf{X},W_j}))} \tag{4.9}$$

相比 公式(4.8)， 公式(4.9)中少了 $\|W_n\|_2$，因为在 Sphereface 中权重向量都被归一化到单位球上，因此 $\|W_n\|_2 = 1, \forall n = 1, ..., N$。 如 图 4.4所示，Sphereface将权

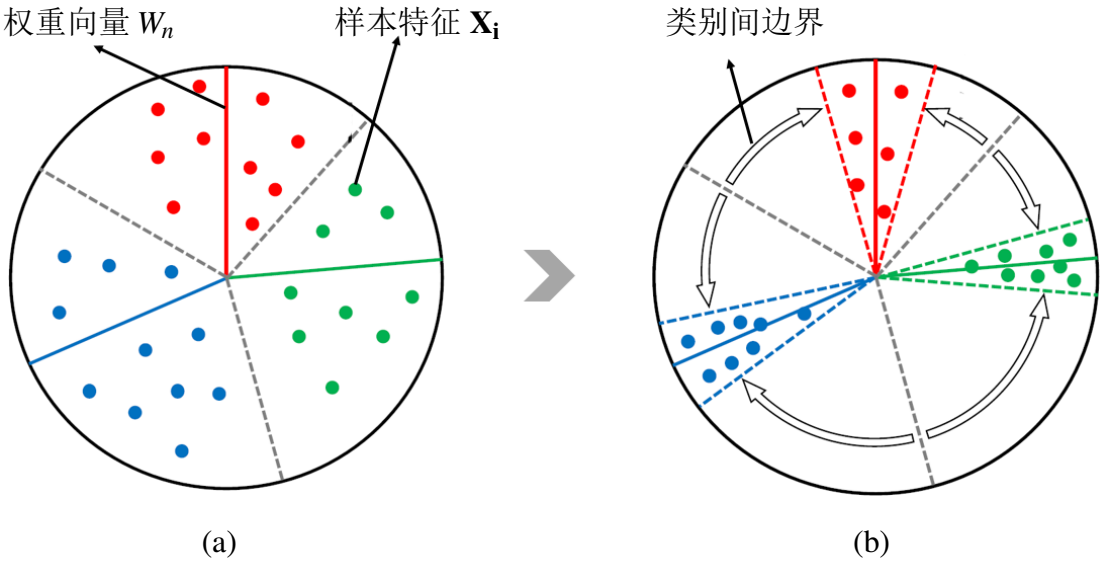


图 4.4 (a) sphereface将权重向量归一化到单位球上； (b) 然后在单位球内对样本特征施加类内的边界，使同类样本间的角度更小。

重向量 W_i 归一化到一个单位圆上，然后压缩同类样本之间的角度距离。

Wang等人 [132]通过在余弦值上减去一个常数来施加类别间的间距（margin），该方法又被称为**CosFace**。 Cosface中的后验概率为：

$$p_n = \frac{\exp(\|\mathbf{X}\|_2 \cdot (\cos(\theta_{\mathbf{X},W_n}) - m))}{\exp(\|\mathbf{X}\|_2 \cdot (\cos(\theta_{\mathbf{X},W_n}) - m)) + \sum_{j \neq n} \exp(\|\mathbf{X}\|_2 \cdot \cos(\theta_{\mathbf{X},W_j}))} \tag{4.10}$$

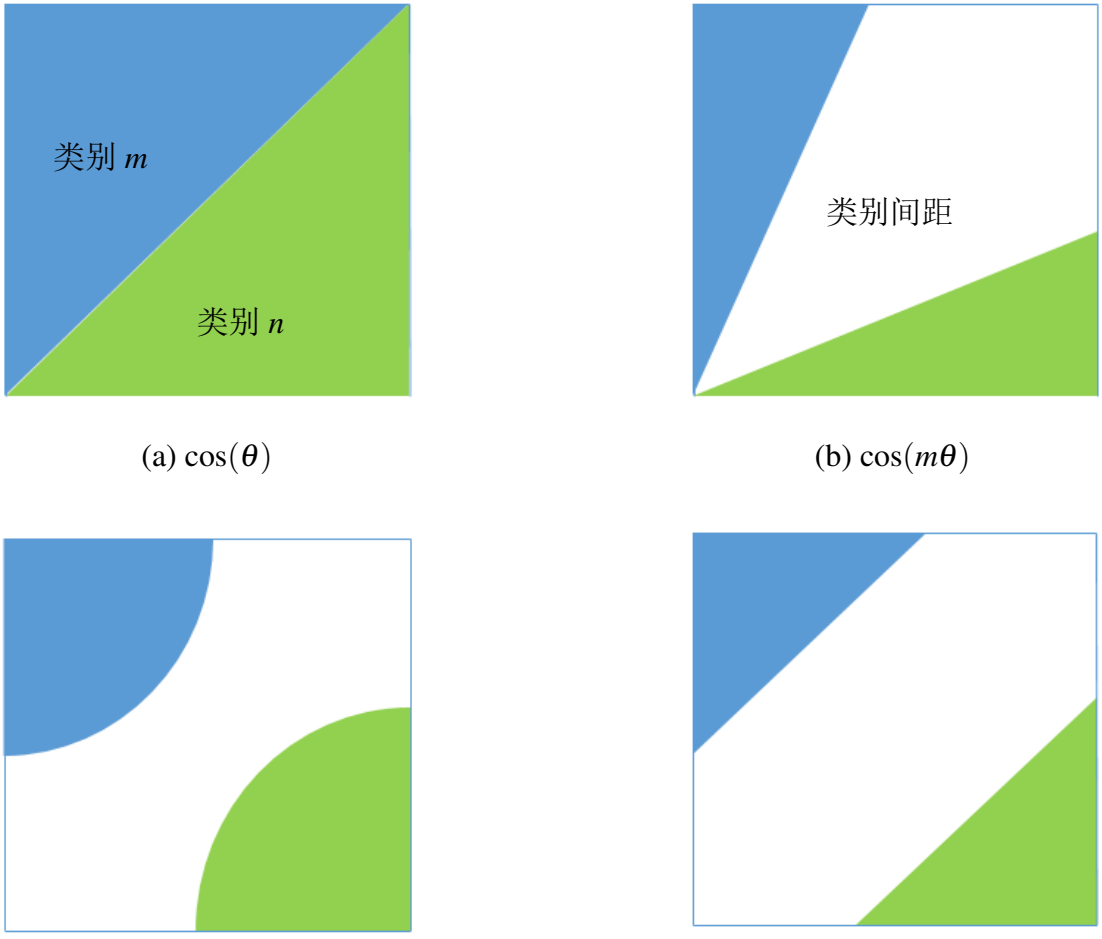
Liu等人提出的**Arcface** [20]将 公式(4.8)中的 $m \cdot \theta$ 改为 $m + \theta$，形成了一个加性（additive）的边界。在 Arcface中，给定图像特征 $\mathbf{X}$，该样本属于类别 n 的概率为：

$$p_n = \frac{\exp(\|\mathbf{X}\|_2 \|W_n\|_2 \cdot \cos(m + \theta_{\mathbf{X},W_n}))}{\exp(\|\mathbf{X}\|_2 \|W_n\|_2 \cdot \cos(m + \theta_{\mathbf{X},W_n})) + \sum_{j \neq n} \exp(\|\mathbf{X}\|_2 \|W_j\|_2 \cdot \cos(\theta_{\mathbf{X},W_j}))} \tag{4.11}$$

Liu等人 [79]提出了“Fair loss”，该损失函数为每个类别设置了不同的间距参数，并且这些间距可以在训练过程中自动调整。 Faire loss 将 [86, 132, 20] 中的边界参数纳入到同一套公式中：

$$p_n = \frac{\exp(\|\mathbf{X}\|_2 \|W_n\|_2 \cdot \cos(m_1 + m_2\theta_{\mathbf{X},W_n}) - m_3)}{\exp(\|\mathbf{X}\|_2 \|W_n\|_2 \cdot \cos(m_1 + m_2\theta_{\mathbf{X},W_n}) - m_3) + \sum_{j \neq n} \exp(\|\mathbf{X}\|_2 \|W_j\|_2 \cdot \cos(\theta_{\mathbf{X},W_j}))} \tag{4.12}$$

然后在训练过程中用强化学习模型去不断调整 m_1, m_2, m_3，针对每一类得到最佳的类别间距。



(a) $\cos(\theta)$ (b) $\cos(m\theta)$

(c) $\cos(\theta) - m$ (d) $\cos(m + \theta)$

图 4.5 (a) 标准的交叉熵损失，类别间可分（separateable）；(b), (c) 和 (d) 分别代表 Sphereface [86]，CosFace [132]和 [20] 中的类别间间隔。

图 4.5中展示了原始交叉熵损失、CosFace、ArcFace 和 SphereFace 几种方法的类间间距（margin）。

特征和权重归一化

早期基于交叉熵损失的人脸识别算法例如 Center Loss [136]用欧氏距离来度量样本特征-样本特征、样本特征-权重向量之间的距离。后来 SphereFace [86]发现在计算后验概率 p_n 时归一化权重向量（公式(4.9)）会有更好的性能。

Wang等人 [131]通过同时将权重向量 W_n 和特征 $\mathbf{X}$ 归一化，将样本间的相似度量转化成角度。由于该方法同时归一化（normalize）特征和权重，因此称为 NormFace。由于在 NormFace中权重和特征都被归一化，相似度的度量都可以用向量间的夹角表示，因此对应的损失函数被称为“角度交叉熵损失”（ACE，angular cross entropy loss）：

$$\mathcal{L}_{ACE}(\mathbf{X},y) = -\log \frac{\exp(\cos(\theta_y))}{\sum_j \exp\cos(\theta_j)} \tag{4.13}$$

其中 θ_j 为样本特征 $\mathbf{X}$ 与权重向量 W_j 之间的夹角。

用角度度量样本-样本，样本-权重向量之间的距离会带来更好的性能，[106]也提出了相同的方法，并指出原因可能是图像特征模长会受到图像质量的影响，而权重向量模长会受到该类样本数目的影响。因此后续的方法 ArcFace [20]、CosFace [132]、FairLoss [79]都选择同时归一化样本特征和权重向量。 图 4.6展现了三种不同的归一化策略：

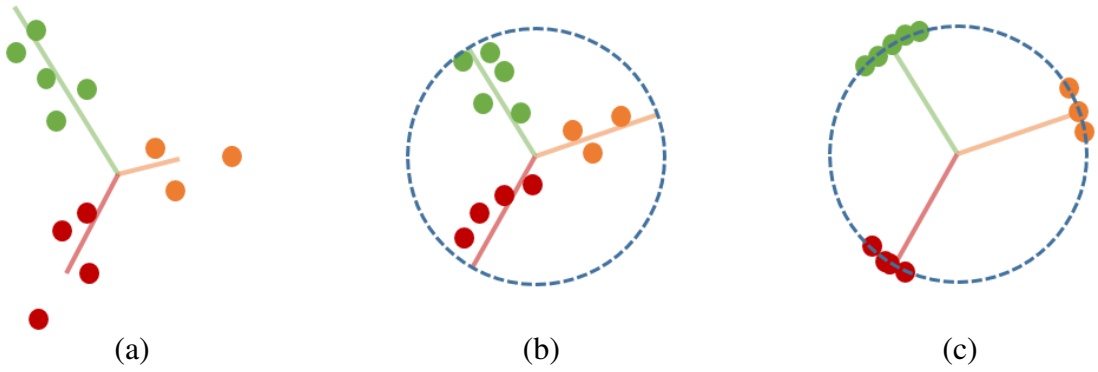


图 4.6 不同的归一化策略。 (a) 权重向量和特征都不归一化； (b) 只归一化权重，不归一化特征； (c) 同时归一化特征和权重。

表格 4.1 统计了近几年几种基于交叉熵损失的人脸识别方法，包括它们的发表刊物，Margin 形式，特征和权重归一化情况等。

方法	出版物	Margin	权重归一化	特征归一化
Large Margin [85]	ICML2016	$\cos(m\theta)$	✗	✗
NormFace [131]	ACMM2017	✗	✓	✓
Sphereface [86]	CVPR2017	$\cos(m\theta)$	✓	✗
CosFace [132]	CVPR2018	$\cos(\theta)-m$	✓	✓
ArcFace [20]	CVPR2019	$\cos(\theta+m)$	✓	✓
FairLoss [79]	ICCV2019	$\cos(m_1+m_2\theta)-m_3$	✓	✓

表 4.1 几种基于交叉熵损失的人脸识别方法汇总。包括发表出处，类别间间距（margin）形式，权重、特征归一化。

第四节 互斥正则损失

虽然 SphereFace [86]和 Center Loss [136]等方法通过增强类内紧凑度度提高了特征的区分度，并且显著提升了人脸识别的性能，但是类间离散度作为特征区分度的重要因素之一却在当前一些方法中被忽略了。

本节提出了一种新的正则损失项，该正则项通过显示地增大不同类别样本特征对应的权重向量之间的角度，然后权重向量带动样本特征与其他类别特征彼此远离，从而达到增强样本特征类间离散度的目的。由于该正则项使得不同类的权重向量之间彼此排斥和远离，因此被称为“互斥正则项”（exclusive regularization）。

4.4.1 互斥正则化的数学定义

假设 $G_\theta(\cdot)$ 为卷积神经网络特征提取器，输入图片 I_i，输出特征 $X_i \in \mathbb{R}^K$，该网络的参数为 θ。

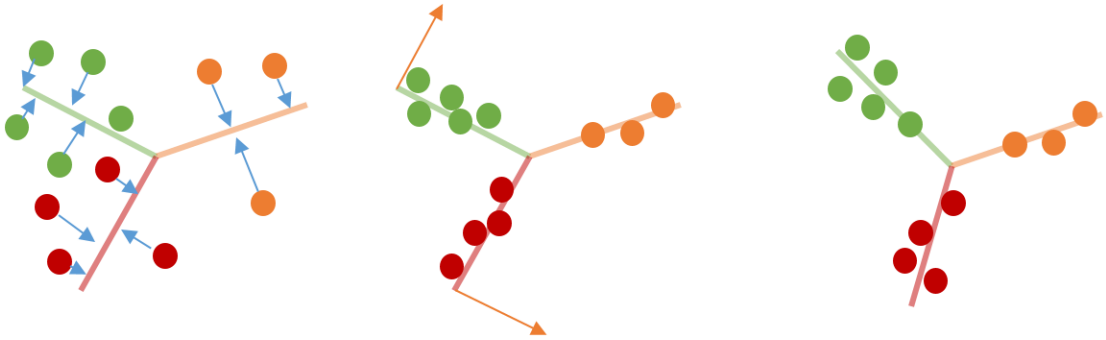
矩阵 $W \in \mathbb{R}^{K\times N}$ 为线性分类器(图 4.10中的 FC2)的权重，将输入的特征映射成对应每个类别的输出，其中 K 为特征维度，N 为类别总数，W 的每一列 W_n 为对应第 n 类的权重向量。给定输入图片 I_i，该图像对应的特征为：

$$\mathbf{X}_i = G_\theta(I_i). \tag{4.14}$$

根据 公式(4.13)，角度交叉熵损失（angular cross entropy loss）定义为：

$$\mathcal{L}_{ACE}(\mathbf{X}, y) = -\log \frac{\exp(\cos(\theta_y))}{\sum_{n=1}^{N} \exp\cos(\theta_n)} \tag{4.15}$$

其中 θ_n 为样本特征 $\mathbf{X}$ 与权重向量 W_n 之间的夹角。在训练过程中模型为了减小 公式(4.15)中的损失函数，会不断减小样本特征 $\mathbf{X}$ 与对应类别的权重向量 W_y 之间的夹角 θ_y。因而我们可以将权重向量 W_n 看多对应类别的类别中心，在训练过程中，样本特征会不断向对应类别中心靠拢。这样，如果对分类层的参数 W 施加一个正则损失，使得在训练过程中 W 的每一列彼此远离，这样在交叉熵损失的作用下，对应类别的特征也会互相远离。如 图 4.7所示，交叉熵损失迫使样本特征往类别中心移动；互斥正则函数则让类别中心彼此远离，在交叉熵损失和互斥正则函数的双重作用下，同类样本向类别中心靠拢，不同类样本往远离的方向移动。



(a) (b) (c)

图 4.7 （a）在交叉熵损失（ 公式(4.15)）的作用下，样本特征（圆点）像权重特征（实线）的方向移动；（b）互斥正则函数将权重特往向彼此远离的方向推动；（c）在交叉熵损失和互斥正则函数的双重作用下，同类样本向类别中心靠拢，不同类样本往远离的方向移动。

为了使得不同类别的中心 W_n 彼此远离，互斥正则函数（exclusive regularization）定义为：

$$\begin{aligned}\mathcal{L}_{exclusive}(W) &= \frac{1}{N}\sum_{i=1}^{N}\max_{j\neq i}\cos(W_i, W_j) \\ &= \frac{1}{N}\sum_{i=1}^{N}\max_{j\neq i}\frac{W_i \cdot W_j}{\|W_i\| \cdot \|W_j\|}.\end{aligned} \tag{4.16}$$

$\max_{j\neq i}\cos(W_i, W_j)$ 实际上就是与 W_i 角度最小的 $W_j, j \neq i$，因此 公式(4.16)中的互斥正则函数实际上是在最小化权重向量 W_i 与其最近邻之间的 cos距离，最小化 cos也就相当于最大化两者之间的角度。

在训练过程中同时最小化交叉熵损失和互斥正则损失，最终整个模型的损失函数为：

$$\mathcal{L}(\theta,W)=\mathcal{L}_s(\theta,W)+\lambda\mathcal{L}_{exclusive}(W), \tag{4.17}$$

其中 λ 是一个用于调节两项损失函数比例的系数。

这样，在交叉熵损失和互斥正则损失的共同作用下，样本特征就能既保证类内紧凑，又保持类间离散。同时 公式(4.17)中的交叉熵损失可以用其它类似的损失函数代替，实现互斥正则函数与其他方法的协同。

4.4.2 用投影梯度下降法优化互斥正则函数

优化公式(4.17)中的损失函数可以表示为:

$$(\theta^*,W^*)=\underset{(\theta,W)}{\operatorname{argmin}}\mathcal{L}(\theta,W). \tag{4.18}$$

其中 W 为线性分类器（ 图 4.10中的 F2）的参数 θ 为卷积神经网络特征提取器（ 图 4.10中的其它层）的参数。参数 θ 可以使用普通的梯度下降法来更新：

$$\theta^{t+1}=\theta^t-\alpha\frac{\partial\mathcal{L}_s(\theta^t,W)}{\partial\theta^t}, \tag{4.19}$$

其中 α 是学习率。

由于参数 W 的每一列必须被归一化到一个单位圆上，而普通的梯度下降法则会将参数 W 更新到单位圆之外，不满足角度交叉熵损失的条件。本文采用投影梯度下降法（projected stochastic gradient descent） [8]来更新 W：

$$\begin{cases}\hat{W}^{(t+1)}=W^t-\alpha\frac{\partial\mathcal{L}}{\partial W^t}\\ W^{(t+1)}=Normalize(\hat{W}^{(t+1)}).\end{cases} \tag{4.20}$$

投影梯度下降法的第一步按照普通的梯度下降法规则来更新参数，第二步将更新完的参数“投影”到约束区域内，在本文中，也就是将更新后的参数 W 重新投影回单位圆。

4.4.3 互斥正则函数的梯度

公式(4.20)需要互斥正则函数对参数 W 的梯度（gradient）。假设 W_j 为 W 的第 j 列，并且收到 $|W_j|_2^2=1$ 的约束. 那么 $\mathcal{L}_{exclusive}$ 对 W_j 的偏导数为:

$$\frac{\partial\mathcal{L}_{exclusive}(W)}{\partial W_j}=W_{j'}+\sum_{W_i\in\mathbb{C}}W_i \tag{4.21}$$

上式中 $W_{j'}$ 是 W_j 的最近邻:

$$j' = \underset{i\in\{1,\ldots,C\},i\neq j}{\operatorname{argmax}} W_j \cdot W_i.$$

$\mathbb{C}$ 是所有最近邻为 W_j 的列向量的集合:

$$\forall W_i \in \mathbb{C}, \quad \underset{k\in\{1,\ldots,C\},k\neq i}{\operatorname{argmax}} W_i \cdot W_k = j.$$

第五节 实验论证

4.5.1 MNIST数据集上的示意性试验

为了说明本问题提出的**互斥正则损失函数**的作用原理，本研究先在MNIST数据集上做一个简单的示例试验。在目前已有的一些方法中，类别总数相对于特征维度来说偏少，因此无法体现出类别中心在特征空间的分布不均。例如在一些典型的人脸识别算法[85, 86, 54]提取了512维的人脸特征，但是训练集的类别数仅在 10^4 数量级，因此这些类别中心可以很容易在高维空间中离散分布，不存在挤兑。

为了清晰地展现类别中心在特征空间的挤兑现象，本实验挑选了 MNIST数据集 [65]中的三类，同时为了可视化的方便，将特征限制在2D空间中。实验网络采用一个略作修改的 LeNet[68]网络结构。

本实验一共使用三种损失函数训练：

1. 交叉熵损失,
2. 交叉熵损失 + **中心损失**[136]（ 公式(4.4)），
3. SphereFace[86]（ 公式(4.9)），
4. SphereFace[86] + 本章提出的**互斥正则损失**（ 公式(4.16)）。

这三种损失函数训练得到的特征分布如图 4.8所示。

如 图 4.8 (a)所示，由交叉熵损失训练得到的样本特征类别间界限明确，同类样本特征集聚成簇。 图 4.8 (b)展示了加上**中心损失**后的特征分布。通过惩罚样本特征到该类样本特征中心的距离，**中心损失**能让同类样本特征更加紧凑，类间紧凑更强。与中心损失在欧式空间压缩同类样本特征之间的距离，SphereFace在角度空间中施加一个类间间距，使得同类样本在角度空间更加紧凑（ 图 4.8 (c)）。

中心损失和 SphereFace都是在交叉熵损失的基础上尝试压缩同类样本特征间的距离，增强类间的紧凑度。在互斥正数则损失的作用下，图 4.8 (d)中不同类别的特征分布更加均匀，类别间特征的距离更大。

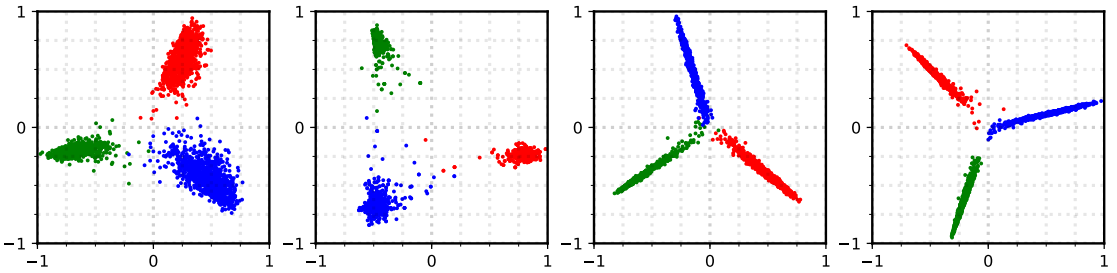


图 4.8 3 类 MNIST 手写数字实验的特征分布，同种颜色的点表示同类别的样本特征。(a) 交叉熵损失能学到类间可分开的样本分布，这样可以保证分类准确性； (b) 中心损失（Center loss） [136] 在欧氏空间内增强类内紧凑度； (c) Sphereface [86] 将同类样本在夹在一个很小的角度内，在角度空间增强类内紧凑度。 (d) 本章所提出的互斥正则损失能指引不同类的样本互相远离，增强类间离散度。

4.5.2 定量比较

本节定量对比了所提出的**互斥正则损失**和其它方法。

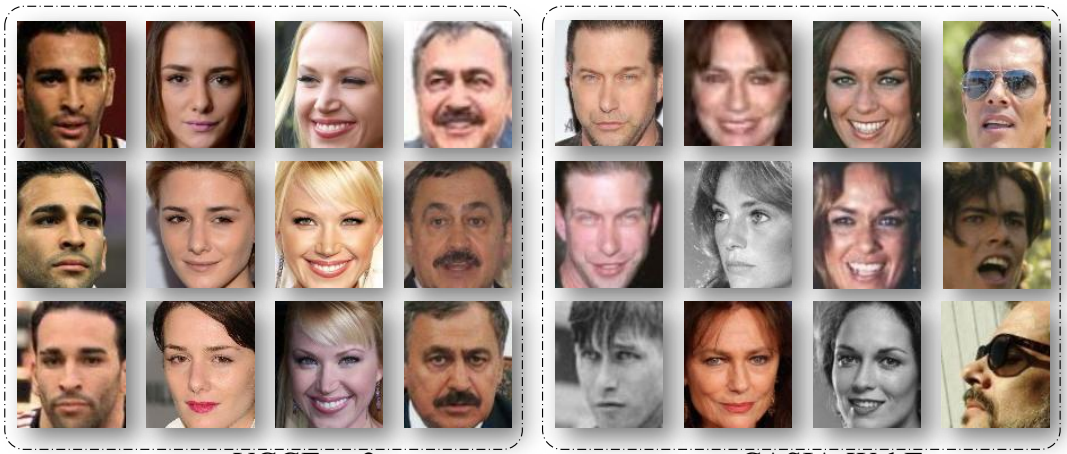


图 4.9 VGGFace2[10] (左) 和 CASIA-WebFace[145] (右) 数据集中的样本，同一列的样本属于同一个个体（identity）。两个数据集中的人脸都包含不同的光照条件、表情、年龄等。总的来说，VGGFace 数据集中的样本图像质量要略高于 WebFace 数据集，因为 WebFace 数据集中有一些侧脸照、低分辨率照片等影响模型识别的样本。

4.5.3 实现细节

网络结构 近几年与卷积神经网络结构设计的相关工作 [127, 39, 62]都指出一个事实，那就是更深的网络往往会带来更好的性能。

在 SphereFace [86] 中作者测试了五种不同深度（4，10，20，36，64）的深度残差网络，更深的网络性能更好，但是占用更多的计算资源也消耗更多的内存。为了平衡模型性能和内存、算力消耗，本节的所有试验都使用20层的残差网络，该网络结构同时也在文献 [136, 20]中使用。如 图 4.10所示，卷积神经网络接收 112×96 输入图片，在经过4个残差结构之后，输出的特征图大小为 $7 \times 6 \times 512$，最后由一个全脸阶层（FC1）将特征图转换为 512 维的特征向量，该特征向量即为 公式(4.13)和 公式(4.15)中的 $\mathbf{X}$。在训练阶段，线性分类层（FC2）将特征 $\mathbf{X}$ 转换为对应每个类别的输出，然后计算交叉熵损失。

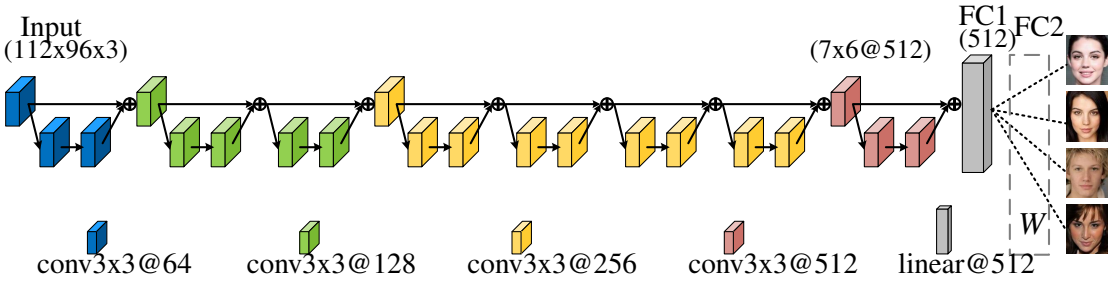


图 4.10 本节实验所用的 ResNet20网络。 'conv3x3@$\mathcal{X}$'表示 3×3 的卷积层，其输出的特征图为 $\mathcal{X}$, $\bigoplus$ 表示逐元素的乘法。 W 是线性分类层的参数，负责将特征 $\mathbf{X}$ 映射为对应每个类别的预测输出。

数据集 本节的实验使用 CASIA-WebFace[145]和 VGGFace2 [10]数据集作为训练数据，并剔除了测试集中存在的样本和个体。 CASIA-WebFace数据集包含494,414张人脸照片，分别属于 10,575 个不同的个体。 VGGFace2 数据集有 310 万张人脸图片，分别属于 8,631 个个体. 图 4.9展示了一部分训练数据中的图片。如 图 4.9中所示，WebFace数据集包含相当多的低分辨率或者是侧脸的图片；而VGGFace数据集的图片分辨率和人脸完整度都比较好。总的来说，VGGFace数据集的质量要比 WebFace数据集高。

数据预处理 人脸对齐是人脸识别中被广泛采用的一个预处理步骤。和之前的一些工作 [128, 114, 123, 136, 86]一样，本节的实验使用 MTCNN [152]来检测数

据集中的人脸关键点（双眼，鼻，双嘴角），然后根据所检测到的关键点将人脸对齐到标准姿态，保证所有人脸的关键点在图像上的同一个位置。

如果图像中有多个人脸则选取离图像中心最近的一个。如果训练集中的图片上没有人脸，则丢弃该训练样本；如果测试集中的图片中没有人脸，则截取图片中央的一个区域作为测试样本。

定量评测规则 为了验证所提出算法的有效性，本文在 LFW [54]、YTF [137] 和 MegaFace [58] 三个数据集上定量对比了本文所提出的方法和其他不同人脸识别的性能。试验中分别提取了测试集中样本的512维特征（图 4.10中 FC2的输出），同时将测试图片左右翻转后提取特征并与原始特征拼接在一起。最终，对每个测试样本试验中都提取到1024维的特征。

对 LFW 和 YouTubeFace 数据集，试验中使用余弦距离（cosine distance）来度量两个特征的相似度，然后使用标准的 10-fold 交叉验证来测试识别精度，最终的精度为 10 次试验的平均。具体来说，测试集中的所有样本被随机地分为10份，然后其中9份作为验证集，在验证集上找出最佳的相似度阈值，该阈值可以在验证集上取得最大的识别精度。然后使用该阈值在1份测试集上得到测试精度。

MegaFace数据集官方提供了测试的程序 [2]，因此试验中仅提取 MegaFace 测试集上的人脸特征，然后用官方提供的测试代码得到评测结果。

不同损失函数的组合 前文曾提到，公式(4.17)中的交叉熵损失可以被其他类似的分类损失函数所替代，实现和互斥正则损失的协同作用。本节的实验将互斥正则损失使用到三种现有的方法中，构成了三种不同的组合： Softmax Loss + $\mathcal{L}_r$ (RegularFace + SM), Softmax Loss + Center Loss + $\mathcal{L}_r$ (RegularFace + [136]), Softmax Loss + angular margin + $\mathcal{L}_r$ (RegularFace + [86]).

在 LFW、YoutubeFace（表格 4.2）和 MegaFace数据集上的定量对比（表格 4.3）结果说明，在原始方法的基础上，本研究所提出的互斥正则损失函数能够显著提升人脸识别的性能，证明了所提出方法的有效性。其中，在 softmax loss 基础上的提升最为明显。由于 Center Loss [136] 以及 SphereFace [86] 两种方法在 softmax 损失函数的基础上已经将人脸识别性能提升到一定程度，在这两种

[2] http://megaface.cs.washington.edu/

方法上进一步提升模型能较为困难。因此本文所提出的方法在 Center Loss [136] 和 SphereFace [86] 基础上的提升比较微弱。

4.5.4 实验结果

LFW和 YouTubeFace（YTF）数据集 LFW数据集[54]包含13,233张测试图片，分别来自5,749个不同的个体。 YTF 数据集 [137]包含3,424段来自http://youtube.com的视频，视频中一共包含1,595个不同的个体，平均每个人有2.15段视频。这些视频帧数分布不均，最多的有6,070帧，少的只有48帧，平均每个视频有181.3帧。 LFW 和 YoutubeFace 数据集中的人脸姿态以及背景都十分复杂，变化很大，因此这两个数据集上的人脸识别比较困难。本节的实验主要对比了三种不同的方法以及这些方法加上互斥正则函数后的结果，实验结果记录在 表格 4.2中。 表格 4.2中的实验结果表明，互斥正则函数能够提升原始方法的性

方法	训练数据	LFW	YTF
DeepFace [128] (3)	4M	97.35	91.4
FaceNet [114]	4M	**99.65**	**95.1**
DeepID2+ [123]	4M	98.70	-
DeepID2+ [123] (25)	4M	99.47	93.2
中心损失 [136]	0.7M	99.28	94.9
交叉熵损失	WebFace	97.88	90.1
中心损失 [136]		98.91	93.4
大间距交叉熵损失 [85]		99.01	93.0
SphereFace [86]		99.26	94.1
互斥正则损失+交叉熵损失		99.02	91.9
互斥正则损失+中心损失		99.18	93.7
互斥正则损失+SphereFace		**99.33**	**94.4**
交叉熵损失	VGGFace2	98.55	93.4
中心损失 [136]		99.31	94.3
大间距交叉熵损失 [85]		99.35	94.1
SphereFace [86]		99.50	95.9
互斥正则损失+交叉熵损失		99.32	94.7
互斥正则损失+中心损失		99.39	95.1
互斥正则损失+SphereFace		**99.61**	**96.7**

表 4.2 LFW和 YTF数据集上的性能比较。 A+B表示使用 A方法的损失函数和 B方法的损失函数共同训练模型。

能，特别是当和 SphereFace [86]一起使用时，取得了所有方法中最好的性能。

4.5.5 MegaFace数据集上的测试结果

MegaFace [58] 是一个相对比较新的大型人脸识别测试集。 MegaFace 数据集被分为两个子集： (1) gallery 子集包含属于 690K 个体的的 1百万人脸图片，以及 (2) probe 集由两个现有的数据集组成：Facescrub [100] and FGNet [101]。

根据训练集的大小，MegaFace 共有两种评测规则。当训练集样本数小于50万张图片时，视为“小”数据集训练，否则视为“大”数据集训练。本实验同时评测了“大”和“小”两种训练模式下各模型的性能。在大数据集模式下使用 VGGFace 数据集 [10]作为训练集，在小数据集模式下使用 WebFace 数据集 [145]进行训练。

实验结果记录在 表格 4.3中。 表格 4.3中“Rank1 Acc”表示取相似度最高的样本作为预测时的准确率，“Ver”表示在误接收率（False accept rate）为 10^{-6} 时的人脸验证正确率。

Method	训练数据（大/小）	Rank1 Acc	Ver.
交叉熵损失	Small	52.86	65.93
大间距交叉熵损失 [85]		67.13	80.42
中心损失 [136]		65.23	76.52
SphereFace [86]		69.62	83.16
互斥正则损失+交叉熵损失		65.91	78.21
互斥正则损失+中心损失		68.37	81.25
互斥正则损失+SphereFace		**70.23**	**84.07**
交叉熵损失	Large	61.72	70.52
中心损失 [136]		70.29	87.01
SphereFace [86]		74.82	89.01
互斥正则损失+交叉熵损失		72.91	88.37
互斥正则损失+中心损失		73.27	89.14
互斥正则损失+SphereFace		**75.61**	**91.13**

表 4.3 MegaFace 数据集 [58] 上的人脸验证性能比较。 challenge 1. “Rank1 Acc”表示取相似度最高的样本作为预测时的准确率，“Ver”表示在误接收率（False accept rate）为 10^{-6} 时的人脸验证正确率。

4.5.6 参数讨论

试验中唯一一个可以调节的参数就是 公式(4.17)中整体损失函数中的权重系数 λ。较大的 λ 使得互斥正则函数主导了训练，会影响模型分类性能；而过小的 λ 不能起到增大类间离散度的作用。为了测试不同 λ 下模型的性能，本

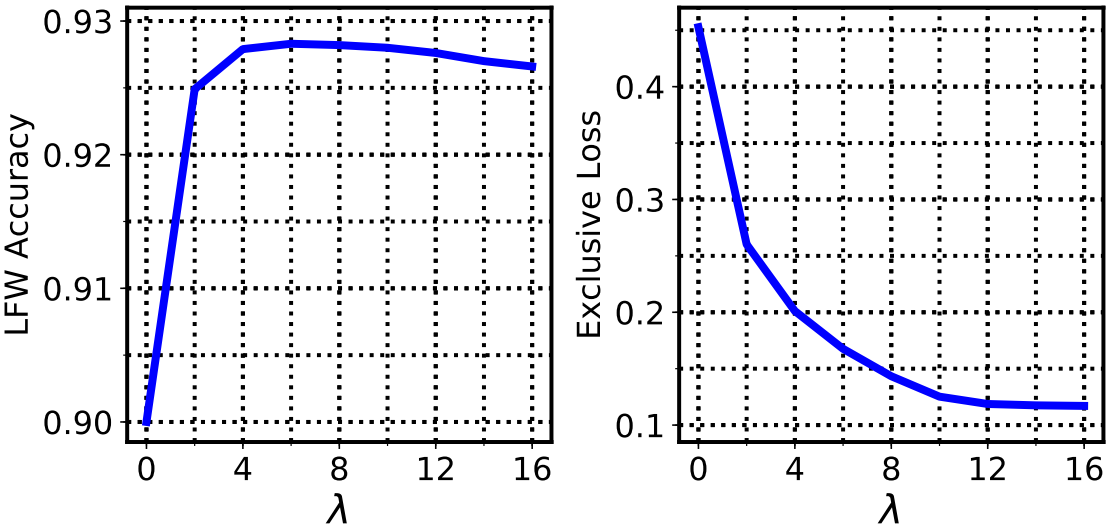


图 4.11 在不同的互斥正则损失权重 λ 下 LFW数据集 [54]上的人脸识别准确度以及收敛后的互斥正则损失。

研究使用不同的 λ 训练了多个模型，然后在 LFW数据集上测试模型的性能。如 图 4.11 (a)所示，随着 λ 变大，模型的性能迅速上升并在当 $\lambda = 6$ 时达到峰值，随后开始下降。 图 4.11 (b)则显示，随着 λ 的增大，互斥损失项（exclusive loss）持续下降。

4.5.7 退火策略

为了进一步研究交叉熵损失和互斥正则损失之间的关系，本节在 图 4.12本节记录了两种损失在训练过程中的变化情况。 图 4.12 (a) 中模型只接受交叉熵损失的监督($\lambda = 0$)，在图 4.12 (b) $\lambda = 1$。在试验中发现，交叉熵损失和互斥正则损失在训练初期互相“抗衡”，因为当同时使用交叉熵损失和互斥正则损失时，在训练初期互斥正则损失出现大幅度的震动，影响模型收敛后的性能。如图 4.12 (b)所示，同时使用交叉熵损失和互斥正则损失的时候，互斥正则损失在训练初期不稳定，先是快速上升，然后突然上升，最后缓慢下降直至稳定。

为了使得训练过程更加稳定，本文采样了**退火策略**（annealing strategy），在训练初期线性地从0开始增大互斥正则损失的权重 λ，待到稳定后再使用固定的 λ 训练。同样的方法在一些近期的研究中[90, 86]也被用来平衡不同的损失函数项。假设 λ 是互斥正则损失的权重，一共退火 N 个 epochs ，那么在训练过程

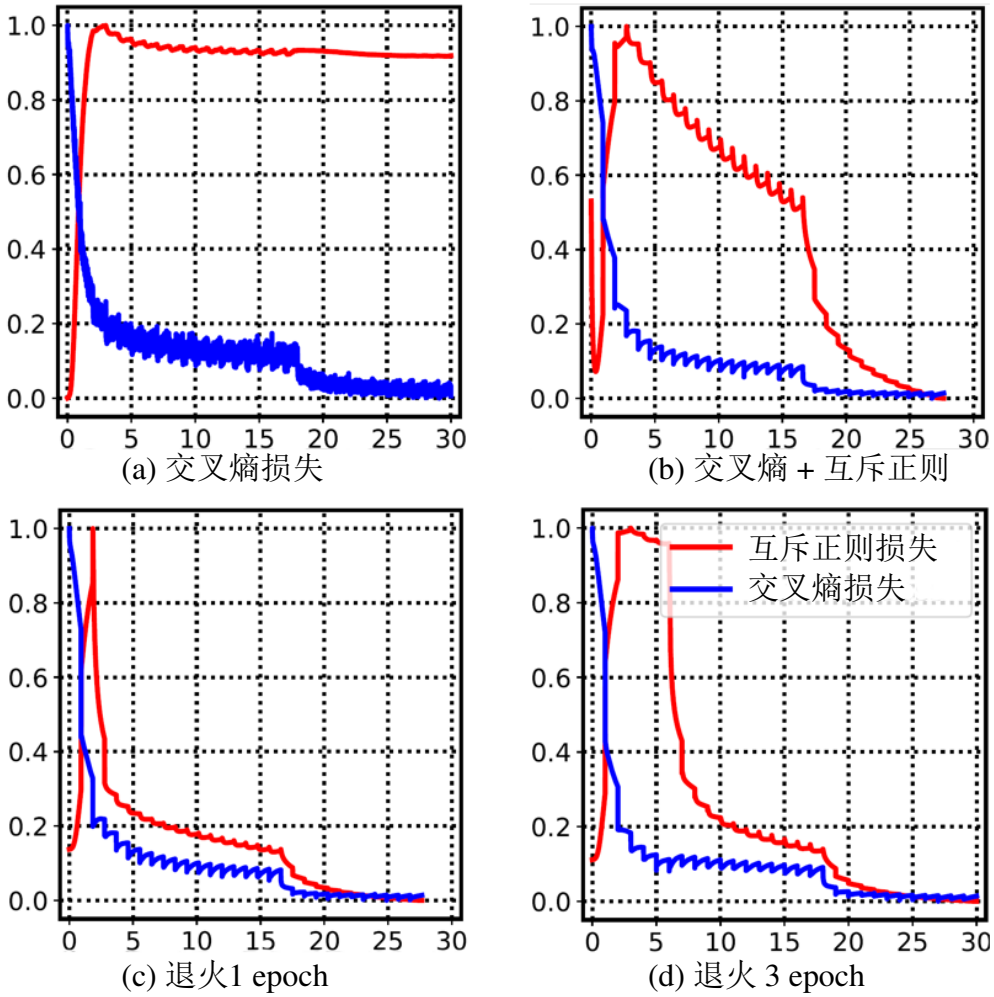


图 4.12 不同情况下交叉熵损失和互斥正则损失的变化曲线。(a): 只有交叉熵损失的情况下（ 公式(4.17)中 $\lambda = 0$），互斥正则损失（红线）在训练初期快速上升，之后保持不变。(b): 同时使用交叉熵损失和互斥正则损失的时候，互斥正则损失在训练初期不稳定，先是快速下降，然后突然上升，最后缓慢下降直至稳定。这种训练过程中的不稳定会影响模型性能。 (c): 在同时使用交叉熵损失和互斥正则损失的情况下，对互斥正则损失进行 1 个 epoch 的退火（在1个 epoch 内从0开始缓慢上升互斥正则损失的权重 λ）。 (d): 在同时使用交叉熵损失和互斥正则损失的情况下，对互斥正则损失进行 3 个 epoch 的退火。

中第 t 个 epoch 时互斥正则函数的有效权重为：

$$\lambda^*(t) = \begin{cases} \frac{t}{N} \cdot \lambda, & t <= N \\ \lambda, & otherwise. \end{cases} \tag{4.22}$$

第六节 小结

本章提出了针对人脸识别的“互斥正则”损失函数。该正则项通过在训练过程中显示地增大不同类别样本特征中心之间的距离，从而增强不同类别特征之间的类间离散度，促进了特征的区分度，增强了人脸识别算法的性能。

在 LFW、YTF 和 MegaFace 三个大型的测试集上的实验表明，本文所提出的正则损失项能够显著提升基准方法的人脸识别性能，验证了本章所提出的方法的有效性。本章相关的研究成果已经被计算机视觉与模式识别领域的顶级会议 IEEE CVPR2019（IEEE conference on Computer Vision and Patterm Recognition）录用。

第五章 自适应的疏正则损失函数

本章提出了一种自适应地稀疏正则损失函数，该损失函数在训练中通过自动控制损失函数中稀疏正则项的系数，促使网络达到指定的稀疏度。在按照稀疏度所指定的比率进行模型压缩后，可以最大限度地保留原始模型的精度。同时还提出了一种基于网络中前后层关系的滤波器重要性估计方法，该估计方法能够根据卷积层和批归一化层（batch normalization）的前后依赖关系准确定位对模型输出贡献更低的滤波器，从而实现无伤剪枝。

本章第一节首先介绍模型剪枝和压缩的相关背景知识，第二节介绍模型剪枝的相关工作，第三节详细说明本章所提出的自适稀疏应正则损失函数，第四节介绍基于前后相关性的滤波器重要性估计算法，第五节介绍实验结果，包括实验细节以及在多个公开数据及上的定量比较结果，第六节对本章做个小结。

第一节 引言

5.1.1 背景知识

神经网络在图像理解、语音识别、自然语言处理等领域的广泛使用，取得了远优于传统方法的性能, 越来越多的应用在采用基于神经网络的解决方案。目前神经网络的结构朝着更深、参数更多、拓扑结构更复杂、计算复杂度更高的方向发展，这给神经网络在很多嵌入式设备和移动设备上的部署带来了挑战。网络计算量和参数体积是制约神经网络在移动端等低功耗设备上部署的制约因素之一。网络压缩（network compression）技术通过剔除网络中的不重要参数或者连接，在一定程度上保持网络性能的基础上降低模型的计算开销，因而具有广阔的实用前景。

广义上讲网络压缩技术可以分为以下几个种类：

1. **网络剪枝**（network pruning）通过在训练过程中对网络参数施加稀疏约束，待训练结束得到稀疏网络之后，根据一定规则去除（剪枝）稀疏的参数，得到压缩后的模型；
2. **轻量级网络结构设计**（network architecture design）通过手工设计轻量级的网络结构，得到计算开销和存储开销更轻量级的网络模型；

3. **知识蒸馏**（knowledge distillation）先训练一个大模型，然后用大模型作为“老师”来指导小模型（学生）的学习，通过将大网络的知识“蒸馏”到小网络中而得到性能优异的轻量级网络模型；
4. **自动网络结构搜索**（NAS，neural architecture search）通过强化学习、进化算法等技术在给定计算开销或者是内存开销约束的情况下自动搜索出合适的网络结构。

网络剪枝方法操作简单，不像轻量级网络设计那样需要大量的人类经验，也不需要像训练知识蒸馏那样训练一大一小两个模型，同时效率也较高，不像自动网络结构搜索方法那样动辄需要上千 GPU 时的计算量才能搜索到一个网络模型。因此，网络剪枝受到了国内外很多研究者的关注，提出了很多的网络剪枝方法 [43, 44, 77, 45, 36, 87, 95, 4, 24, 11, 159, 88, 92, 144]。

如 图 5.1目前主流的网络剪枝方法大致分为三步： 1）第一步先训练一个完整的大型网络，譬如 ResNet50 [39]。在训练过程中有时会使用稀疏正则损失，使得训练到的网络参数稀疏化。 2）根据第一步得到的网络模型，将其中重要性最低的一部分参数和连接去掉。 3）由于剪枝过程中去掉了一部分参数，导致了模型性能下降，第三步通过对剪枝后的网络进行小学习率的训练（微调，finetune）补偿下降的模型性能。

图 5.1 网络剪枝的三个步骤：训练初试网络 → 剪枝 → 调优（finetune）剪枝后的小网络。

一般情况下很多任务中都使用 L2 正则 $L2 = ||W||^2$ 来约束模型参数，避免过拟合 [62, 39, 89, 109, 48]。在模型剪枝任务中，为了在训练初始网络时得到稀疏的参数，目前一些方法往往使用 L1 正则（$L1 = |W|$）作为损失函数的一部分，在训练过程中促使网络参数往 0 移动。 L1 正则函数比 L2 正则更能促使模型稀疏的原因，一个比较粗糙的结束就是 L1 正则函数在 0 点左右的梯度更大。如 图 5.2 所示，L2 正则函数在 0 点附近的梯度逐渐减小，因此在模型参数 W 接近 0 时得不到足够的像 0 移动的动力。而 L1 正则无论在什么位置的梯度都是 ± 1，因此当 W 接近 0 时会得到足够的梯度朝 0 更新。

以分类任务为例，网络的损失函数由用于指导模型分类的交叉熵损失和促

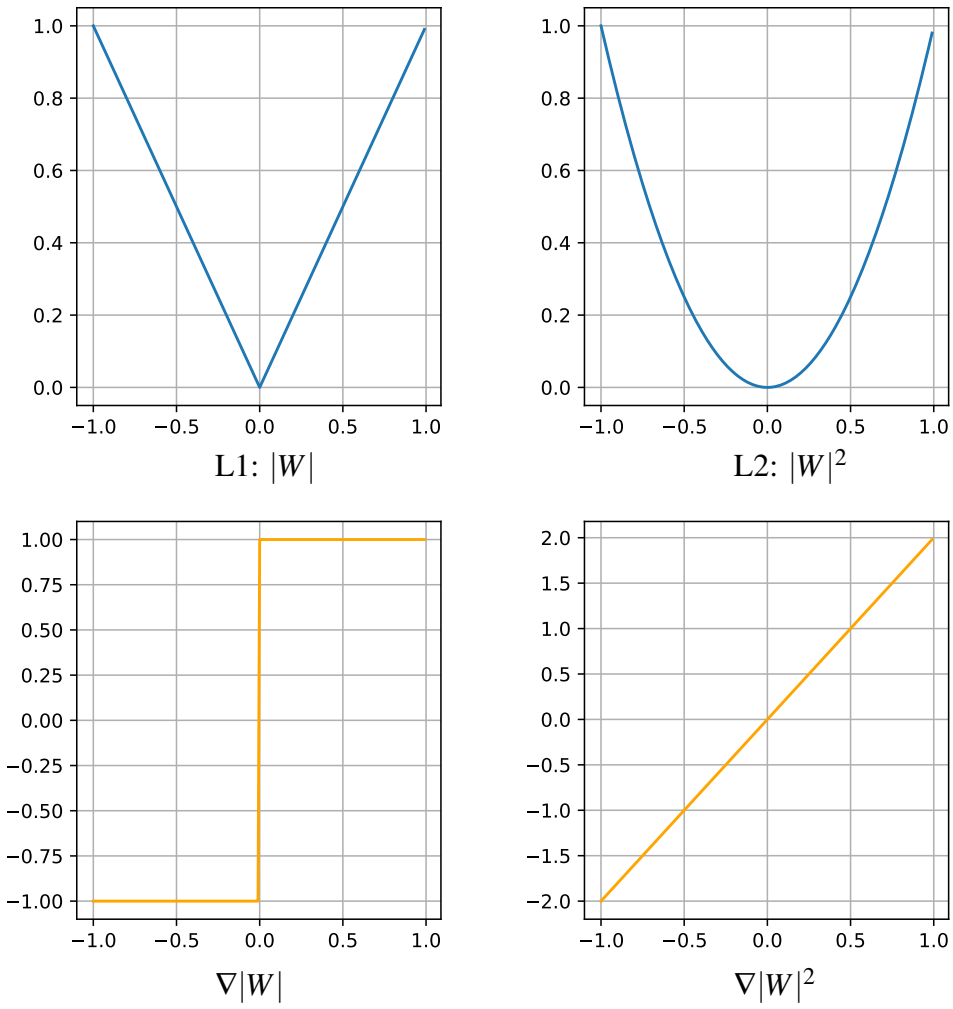


图 5.2 L1 正则函数（左上）的梯度（左下）比 L2 正则函数（右上）的梯度（右下）在 0 点附近更大。

进模型稀疏的 L1 正则函数组成：

$$\mathcal{L}_{overall} = \mathcal{L}_{CE} + \lambda||W||$$

其中 λ 为正则函数的比例系数。在损失函数中，如果 λ 太小，则训练得到的初始模型稀疏程度不够，剪枝得到的模型参数量仍然很大。过大的正则函数比例会影响初始模型的性能，最后调优（finetune）之后得到的小模型性能也受到影响。为了在模型稀疏性和性能之间达到平衡，有些方法 [87, 44, 77, 92] 通过人工搜索为不同模型在不同数据集上分别找到最佳的 λ。在训练初始模型时使用

固定的 λ，第一轮训练完成之后根据模型参数的重要性，将每一层中的一部分滤波器（filters）从模型中剔除，得到剪枝后的模型。这样做虽然能够得到稀疏的初始模型，但是却无法精确控制模型的稀疏性。例如假设事先确定剪枝比率为50%（剔除总共滤波器中的 50%），使用固定的 λ 无法保证初始模型训练完成之后恰好有那么多滤波器参数处于稀疏状态，如果强行剪枝掉 50% 的参数，可能会导致剪枝完之后的模型性能受到很大影响，最终调优（finetune）之后的模型性能也不能达到预期。

因此，如果在训练过程中**动态调整损失函数中 L1 正则的系数 λ**，就可使初始模型训练完成后恰好达到所需要的稀疏度，同时保证剪枝后模型的性能。

第二节 相关工作

本章主要研究模型压缩和剪枝任务中的稀疏正则损失函数。本节简单回顾以往的深度神经网络剪枝相关的工作，特别是在训练中使用稀疏正则损失来促进模型稀疏性的相关工作。粗略地讲，模型剪枝可分为两类：

1. 结构化模型剪枝。这类剪枝方法剔除卷积层中的整个滤波器甚至是整个卷积层的所有参数，剪枝完之后的模型仍然可以直接使用现有的深度学习框架和相关硬件直接进行推理和训练。
2. 非结构化剪枝。这类方法剔除模型参数中的某一些稀疏的值，剪枝完之后的模型参数不具备特定的结构，可以是任意的，因此不方便用现有的深度学习框架和硬件直接推理和训练。

5.2.1 结构化剪枝

2017年，Li等人 [77]根据卷积层中权重的模长来整个剔除卷积核中的滤波器（filters）。 Hu等人 [50]根据网络输出的特征图（feature maps）中 0 所占的百分比来确定被剪枝的滤波器。这两种方法之所以选择用权重的模长或者是输出特征的 0 占比来确定滤波器的重要性，就是因为如果输出的特征数值上很小，那么对后续层的影响也就很小。

由于很多网络结构中卷积层都紧跟批归一化层（BN，batch normalization），由于 BN 层会对特征进行归一化（减均值、除方差）：

$$\mathbf{X} \to \mathbf{Y} = \text{conv}(\mathbf{X}) \to \mathbf{Z} = \text{BN(Y)} = \frac{\mathbf{Y} - \mu}{\sigma}.$$

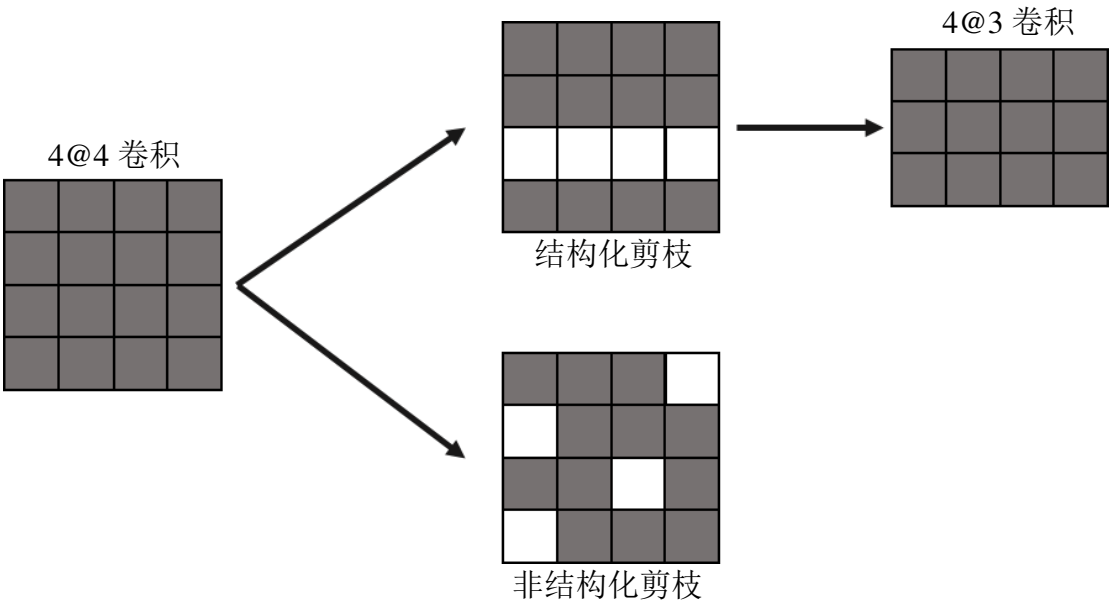


图 5.3 以一个 4@4 的卷积层（输入4个特征图，输出4个特征图）为例，结构化剪枝得到的网络参数仍然可以当做一个完备的卷积层，局有卷积层的参数结构。而非结构化剪枝得到的参数事任意的，无法使用现有的深度学习软件和硬件进行推理和训练。

因此即便卷积层输出的某个特征图数值上很小，经过 BN 层的处理之后所有特征图均变得均值为 0，方差为 1，因此不具有区分滤波器重要性的能力。

Liu等人 [87]提出 Network Slimming 方法，用 BN 层的缩放系数来确定滤波器的重要性。假设输入特征 $\mathbf{X} \in \mathbb{R}^{N \times C_{in} \times H_{in} \times W_{in}}$，$N$ 是 batch size，C_{in} 为输入特征的通道数；输出特征 $\mathbf{X} \in \mathbb{R}^{N \times C_{out} \times H_{out} \times W_{out}}$，其中 C_{out} 是输出特征的通道数。BN 层对输入的特征 $\mathbf{X}$ 作如下变换：

$$\mathbf{Y} = \frac{\mathbf{X} - \mu}{\sigma} \cdot \gamma + \beta. \tag{5.1}$$

公式(5.1)中 $\mu \in \mathbb{R}^{C_{out}}$ 是输出特征的均值，$\sigma \in \mathbb{R}^{C_{out}}$ 是输出特征的方差，$\gamma \in \mathbb{R}^{C_{out}}$ 是可以学习的缩放因子，$\beta \in \mathbb{R}^{C_{out}}$ 是可以学习的偏置项。文献 [87]中使用 γ 作为滤波器（或者是特征图）重要性的标准。值得注意的是，该方法在训练初试网络时（ 图 5.1中的第一步）对 BN 层的缩放系数 γ 使用了 L1 正则函数来促使缩放系数变得稀疏。根据 公式(5.1)，当 BN 层的缩放系数 γ 接近 0 时，对应的输出特征图也会变为 0，对后续层的影响也变得很小。因此 γ 可以作为特征图（或者是滤波器）重要性的标准（图 5.4）。

Huang [55]在残差网络的短连接上增加了一个缩放系数，然后在该缩放系数

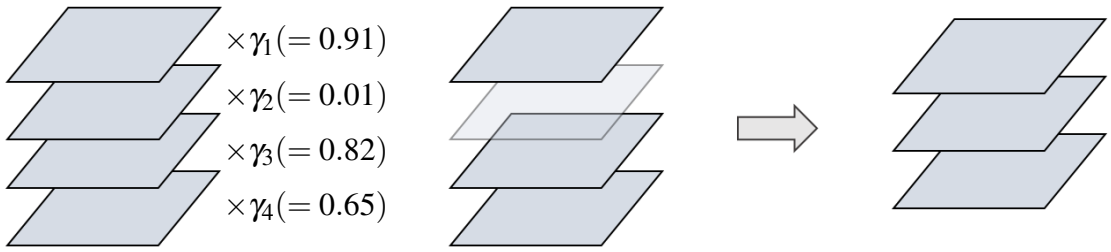


图 5.4 文献 [87] 用批归一化层（BN 层）的缩放系数作为输出特征图的重要性，从而确定哪些滤波器应该被剪枝掉。

上施加 L1 正则。在训练完成之后，根据缩放系数的大小来剪枝一部分短连接，从而减小网络的计算开销和内存开销（图 5.5）。

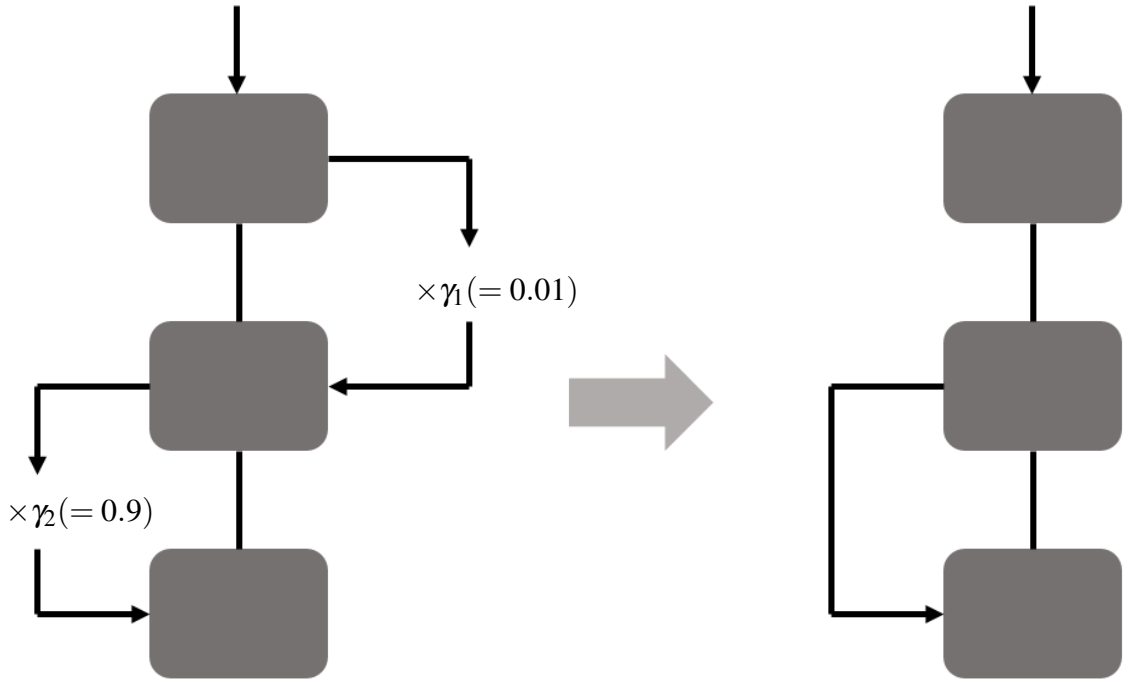


图 5.5 文献 [55] 根据连接中的缩放系数对短连接进行剪枝。

He等人 [45]用输出的特征图的重建误差（reconstruction error）来确定哪些滤波器应该被剪枝。文献 [95]通过泰勒展开（taylor expansion）来计算每个输出特征图通道对模型最终损失的影响，从而确定对应滤波器的重要性。 Suau等人 [122]通过主成分分析计算滤波器参数之间的相关系数，从而找出冗余的滤波器。文献 [16]逐层计算每个滤波器的“补偿系数”来估计滤波器被移除之后对模型的影响。 2019年，He [43]等人将同一层的每个滤波器参数当做高维空间中

的一个点，然后通过滤波器参数和同层参数几何中心点（geometric median）之间的距离来确定滤波器的重要性，离几何中心点越近的滤波器越容易被其它滤波器代替，重要性越低。

以上方法的一个主要的研究热点在于如何确定哪些滤波器应该没剪枝，也就是确定每个滤波器的重要性（importance）。有的方法通过参数的幅度值来确定滤波器的重要性 [55, 87, 77]，有的方法使用重建误差来确定每一个特征图通道的重要性 [92, 45]，有的方法通过几何平均点来确定哪些滤波器是可以被取代的 [43]。

5.2.2 非结构化剪枝

非结构化剪枝方法直接单独去掉单独的参数值，而不是整个滤波器，因此得到的模型参数十分不规则，无法用现有软件库或者是硬件加速。

非结构化的模型剪枝可以赘述到1990年代。 1990年，LeCun等人 [66]通过损失函数对网络参数的 Hessian 矩阵来计算每个参数值对损失函数的贡献，从而确定哪些参数可以被去掉。文献 [38]采用了类似的思路去掉模型中的冗余参数。在最近的2015年，Han等人 [36]通过幅值来确定每个参数的重要性来对单独的参数点进行剪枝。进一步，在文献 [35]中对剪枝后的参数使用哈夫曼编码来压缩模型参数。 Srinivas等人 [120]使用一种迭代策略逐步去除模型中的参数。Louizos等人 [91]对模型参数施加 L0 正则，并通过随机的 0/1 门（gate）来裁剪模型参数。

由于非结构化剪枝方法得到的模型无法直接应用到现有的软件库上，也不能使用现有的硬件设备加速，因此结构化的剪枝方法更有实际的应用前景。

5.2.3 其它模型压缩方法

除了参数剪枝之外，也有一些研究尝试从其它方面入手展开模型压缩研究。

低秩分解

文献 [22, 117, 154] 对网络全连接层（fully connected layers）的参数进行矩阵分解，例如奇异值分解（SVD） [22]，然后在矩阵分解中寻找原参数矩阵的低秩重建，从而压缩网络参数。但是这类方法只能压缩全连接层的参数，无法对占模型大多数的卷积层参数进行压缩，因此压缩比例比较低。

参数量化

目前深度学习模型普遍在推理和训练的时候使用单精度的浮点数（float 32）表示数据和参数。如果将模型参数量化成用整数（例如 int8），或者甚至布尔值（0/1）来表示模型参数和数据，那么将节省巨大的计算开销和内存开销。文献 [13, 18, 107, 140]用低比特位的整数或者是布尔数来表示模型中的参数和中间特征来压缩模型参数。例如，Courbariaux等人 [18]将网络参数和特征二值化； Rastegari等人 [107]用三值数来表示模型参数和特征，模型参数和特征处于（0-1,0,1）三种状态之一。尽管将模型参数量化能够很大程度上减轻模型的计算和内存开销，但是由于量化过程中模型性能丢失严重，因此当前这些模型量化方法并不能应用到实际之中。

网络结构搜索

目前经典的神经网络模型，例如 AlexNet [62]、VGGNet [116]、ResNet [39]都依靠人工设计来确定层与层之间的连接以及每一层有多少个滤波器（filters）。很多研究试图用学习算法来自动搜索合适的网络结构，使得搜索到的网络结构既满足性能要求，又只有较低的计算和内存开销 [19, 80, 103, 129, 138, 161]，这类方法被称作“自动网络结构搜索”（NAS，neural architecture search）。例如 ChamNet [19] 通过贝叶斯方法来优化一个高斯过程来预测模型的精确度，该模型在不同层有各不相同的滤波器个数。 FBNet [138]使用基于梯度的方法来同时训练网络模型和搜索每一层的滤波器个数。这类方法也可以看做是一种滤波器层面的剪枝，除了没有在参数上施加稀疏化正则。同时，也有一些方法使用强化学习等算法通过一次训练直接搜索处包括网络连接和各层滤波器个数在内的整个网络结构 [161]。需要指出的是，基于强化学习的方法通常十分耗时，有时甚至需要上千 GPU 时才能搜索到网络结构。

第三节 自适应的 L1 正则损失函数

Network Slimming [87] 等一些方法在模型参数上施加固定权重的 L1 正则来促进模型稀疏性，然后在第一轮训练完成后按照参数幅值剪枝掉一定比例的滤波器，实现网络参数压缩和剪枝。然后，在训练中找到一个合适的 L1 正则权重 λ 并不容易，不同数据集、网络结构下权重都不一样。例如，Network Slimming 通过在不同数据集和网络结构上做多次重复的网格化搜索（grid search）来确定

合适的 λ，这样做无疑是低效和不自动化的。而且，不同的剪枝比率也需要不同的稀疏正则权重，剪枝比率高的情况下稀疏正则权值显然要更大。

为了避免在各种实验配置（网络结构，数据集，剪枝比率）上通过手工搜寻来寻找合适的 $L1$ 正则函数的权重，本文提出了一种自动控制损失函数中正则函数权重的方法。与 Network Slimming 方法 [87]一样，本文也将 $L1$ 正则施加在批归一化层（BN）的缩放因子 γ 上（ 公式(5.1)）。

5.3.1 滤波器筛选

Network Slimming [87] 通过全局排序，将所有层的滤波器排序，然后按照滤波器重要性从低到高筛选出需要被剪枝的滤波器。由于不同层的滤波器参数内在的统计规律不一样，放在一起比较并不合适，在之后的实验部分会证明这种剪枝策略会导致不稳定的模型结构和较差的性能。

例如，假设有两个层中的滤波器重要性分别是 $\{0.10, 0.01, 0.03, 0.15\}$ 和 $\{1, 100, 2, 200\}$，按照 Network Slimming 的做法，假设要剪枝 50% 的参数，那么第一个卷积层的所有滤波器都将被裁掉，而第二个卷积层的所有滤波器都将被保留，最后的网络结构将无法训练。因此这种不考虑层之间滤波器重要性分布的本征差异的滤波器筛选策略有可能带来不稳定的模型结构。

本文提出一种基于局部比较的滤波器筛选规则，通过比较同一层的滤波器的重要性确定哪些滤波器应该被剪枝，保留了不同层之间参数幅值的特有的分布。假设第 l 层第 i 个滤波器的重要性（importance）就是第 l 层的第 i 个批归一化层（BN）缩放因子 γ_i^l。并且，如果 γ_i^l 满足一下条件，即认为该滤波器是“稀疏的”：

$$\gamma_i^l < \tau \cdot \max_j \gamma_j^l. \tag{5.2}$$

其中 $\tau = 0.05$ 是一个很小的阈值。 公式(5.2) 表明，如果一个滤波器的重要性（γ_i^l）小于同层重要性最大的那个滤波器的 5%，那么就说明这个滤波器是稀疏的。

5.3.2 自适应 λ 调整

为了避免在各种实验配置（网络结构，数据集，剪枝比率）上通过手工搜寻来寻找合适的 $L1$ 正则函数的权重，本文提出了一种自动控制 λ 的方法。在

训练中的 t 时候，定义模型的稀疏度（sparsity）为：

$$P = \frac{\sum_l |\mathcal{F}^l|}{\sum_l C^l}. \tag{5.3}$$

其中 $\mathcal{F}^l$ 是 l 层所有满足公式(5.2)条件的稀疏滤波器的集合。因此 公式(5.3)中的模型稀疏性 P 为**模型中的系数滤波器的占比**。

给定一个目标模型稀疏性 P，本文提出的自适应 L1 正则函数权重算法根训练中模型的稀疏度（ 公式(5.3)）自动调节 L1 稀疏正则的权重系数 λ。在训练中，给定训练过程的总 epochs （模型遍历整个数据集称为一个 epoch）N 以及预定的模型稀疏度（剪枝比率）r，如果模型的稀疏度不够，例如 $P_t - P_{t-1} < (r - P_{t-1})/(N - t + 1)$，则增大 λ: $\lambda \leftarrow \lambda + \Delta_\lambda$；如果模型稀疏度过高（$P_t > r$），则降低 λ：$\lambda \leftarrow \lambda - \Delta_\lambda$。上述过程已经总结到算法 5.6中。

Initialize $\lambda_1 = 0, P_1 = 0, N =$ #epochs
for $t := 1$ *to* N **do** *train for 1 epoch*
  $P_t = \frac{\sum_l |\mathcal{F}^l_{\text{pruned}}|}{\sum_l C^l}$
  if $P_t - P_{t-1} < \frac{r - P_{t-1}}{N - t + 1}$ **then**
    $\lambda_{t+1} = \lambda_t + \Delta_\lambda$
  else if $P_t > r$ **then**
    $\lambda_{t+1} = \lambda_t - \Delta_\lambda$
end

图 5.6 自适应稀疏正则调整算法

第四节 考虑前后关系的滤波器重要性估计

5.4.1 前后相关性分析

Network Slimming [87]方法中只用批归一化层（batch normalization）的缩放因子 γ 来确定某个滤波器的重要性。但是，在卷积神经网络中，层与层之间是序贯的，连续的，连续的两层之间是紧密联系的。因此，只单独地考虑某一层参数的大小来确定滤波器的重要性是不全面的。

例如，l 层的 BN 层缩放因子 γ^l_i 虽然可能很小，但是紧接着的 $l+1$ 层的卷积核中对应的滤波器可能幅值很大，因此将 l 层的第 i 个滤波器剪掉就会对模型

输出带来很大的影响。基于以上分析，本文提出了基于前后依赖关系的滤波器重要性评判准则。**同时考虑 BN 层缩放因子大小以及后续卷积层对应滤波器参数的大小，共同确定滤波器的重要性。**

通常我们假设一个卷积神经网络包含多个卷积层(Conv layers)，批归一化层（batch normalization）[56]和非线性激活层例如 ReLU。这些层按次序处理输入数据，并最终输出模型预测的结果。特别地，在 BN 层中，每个输入/输出特征图通道都是独立处理的，每个特征图通道都对应一个均值、方差、缩放因子和偏置项 公式(5.1)。为了识别出对网络影响最小的滤波器，以便于减小剪枝对模型的影响以及提高剪枝后模型的性能，本节对每个滤波器（特征图通道）在网络中的贡献（contribution）作如下分析。

假设 $\boldsymbol{X}^l \in \mathbb{R}^{C^l \times H^l \times W^l}$ 是经过第 l 个 BN 层归一化之后（但是还没有乘以缩放因子）的特征，因此经过缩放因子之后的特征 $\boldsymbol{Y}^l$ 可以表示为[1]：

$$\boldsymbol{Y}_c^l = \gamma_c^l \boldsymbol{X}_c^l, \tag{5.4}$$

其中 $\gamma^l \in \mathbb{R}^{C^l}$ 表示第 l^{th} 个 BN 层的缩放因子，$\boldsymbol{X}_c^l$ $(resp.$ $\boldsymbol{Y}_c^l)$ 表示 $\boldsymbol{X}^l$ 的第 c 个通道。最后，一个利普希茨连续（Lipschitz-continuous）的非线性变换（non-linearity）σ 作用到 $\boldsymbol{Y}^l$ 上：

$$\boldsymbol{Z}^l = \sigma(\boldsymbol{Y}^l). \tag{5.5}$$

最后，$\boldsymbol{Z}^l$ 的所有通道都通过一个卷积操作被融合到 $\boldsymbol{F}^{l+1} \in \mathbb{R}^{C^{l+1} \times H^{l+1} \times W^{l+1}}$ 中，其中不同的通道对最终的输出 $\boldsymbol{F}^{l+1}$ 贡献也不同。

令 $\boldsymbol{W}^{l+1} \in \mathbb{R}^{C^{l+1} \times C^l \times k \times k}$ 表示 $(l+1)$ 层的卷积核, 其中 k 为卷积核大小。卷积操作可以表示为：

$$\boldsymbol{F}^{l+1} = \boldsymbol{W}^{l+1} \circledast \boldsymbol{Z}^l, \tag{5.6}$$

其中 $\circledast$ 表示卷积操作符。实际上卷积操作为一个放射变换，因此这里可以将 公式(5.6) 中的卷积运算重写为：

$$\widetilde{\boldsymbol{F}}^{l+1} = \widetilde{\boldsymbol{W}}^{l+1} \widetilde{\boldsymbol{Z}}^l, \tag{5.7}$$

其中 $\widetilde{\boldsymbol{F}}^{l+1} \in \mathbb{R}^{C^{l+1} \times H^{l+1} W^{l+1}}$, $\widetilde{\boldsymbol{W}}^{l+1} \in \mathbb{R}^{C^{l+1} \times k^2 C^l}$, 以及 $\widetilde{\boldsymbol{Z}}^l \in \mathbb{R}^{k^2 C^l \times H^{l+1} W^{l+1}}$ 是 $\boldsymbol{F}^{l+1}$、$\boldsymbol{W}^{l+1}$ 和 $\boldsymbol{Z}^l$ 展开之后的结果。

[1]为了方便，本文忽略公式(5.6)中的偏置项。

将 $\widetilde{\boldsymbol{F}}^{l+1}$ 沿着通道这个维度分解：

$$\widetilde{\boldsymbol{F}}^{l+1} = \sum_{c=1}^{C^l} \widetilde{\boldsymbol{W}}_c^{l+1} \widetilde{\boldsymbol{Z}}_c^l, \tag{5.8}$$

其中 $\widetilde{\boldsymbol{W}}_c^{l+1} \in \mathbb{R}^{C^{l+1} \times k^2}$， $\widetilde{\boldsymbol{Z}}_c^l \in \mathbb{R}^{k^2 \times H^{l+1} W^{l+1}}$。

下面分析卷积层输出特征图中每个通道对最终模型输出的贡献（或者是影响）[2]

$$\begin{aligned} ||\widetilde{\boldsymbol{F}}^{l+1}|| &\leq \sum_{c=1}^{C^l} ||\widetilde{\boldsymbol{W}}_c^{l+1} \widetilde{\boldsymbol{Z}}_c^l|| \leq \sum_{c=1}^{C^l} ||\widetilde{\boldsymbol{W}}_c^{l+1}|| \cdot ||\widetilde{\boldsymbol{Z}}_c^l|| \\ &\leq \sum_{c=1}^{C^l} ||\widetilde{\boldsymbol{W}}_c^{l+1}|| \cdot \mathcal{L} ||\widetilde{\boldsymbol{Y}}_c^l|| \\ &= \mathcal{L} \sum_{c=1}^{C^l} |\gamma_c^l| \cdot ||\widetilde{\boldsymbol{W}}_c^{l+1}|| \cdot ||\widetilde{\boldsymbol{X}}_c^l||, \end{aligned} \tag{5.9}$$

其中 $\mathcal{L}$ 表示函数 σ 的利普希茨常数（Lipschitz constant）, $\widetilde{\boldsymbol{X}}^l$、$\widetilde{\boldsymbol{Y}}^l$ 是 $\boldsymbol{X}^l$ 和 $\boldsymbol{Y}^l$ 展开后的结果。因为 BN 层中的归一化操作将输出特征 $\boldsymbol{X}_c^l$ 中每个通道都“一致化”了，本文用以下公式量化第 c^{th} 个通道的贡献：

$$S_c^l = |\gamma_c^l| \cdot ||\widetilde{\boldsymbol{W}}_c^{l+1}||, \tag{5.10}$$

公式(5.10)中的 S_c^l 将作为本文中第 l 层的第 c 个滤波器的重要性因子。

第五节 实验结果

本节首先介绍相关的实验细节（5.5.1），然后在 CIFAR [63]，SVHN [99] 和 ImageNet [112] 三个公开数据及上对比本文所提出的自适应正则损失函数方法和之前一些模型剪枝方法的图像分类准确率。

5.5.1 实现细节

本节的所有实验都是修改 Network Slimming [87] 的官方代码[3]实现的，使用了 PyTorch [121] 框架。剪枝实验采取 图 5.8中的“训练（train）,、剪枝（prune）、调优（finetune）”三个步骤。

[2]这里假设非线性激活层在输入 0 时输出也是0.事实上几乎所有常用的激活函数例如 ReLU [98]和其变种 [93, 17, 40]均满足这一假设。

[3]https://github.com/Eric-mingjie/rethinking-network-pruning

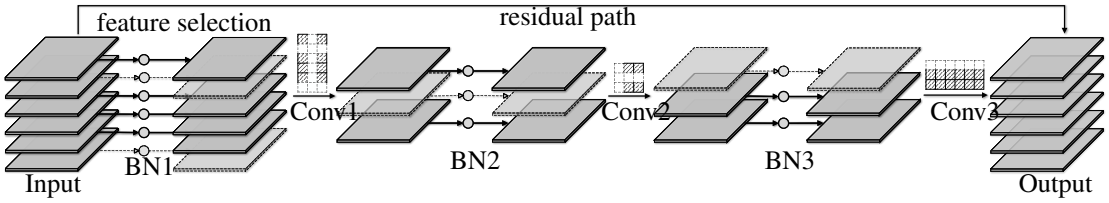


图 5.7 对 ResNet [39]网络中的瓶颈结构（bottleneck structure）进行剪枝。平面和网格表示特征图（feature maps）和卷积核（convolutional kernels）。虚线包围的平面和空白网格单元表示被剪枝的特征图和滤波器参数。在第一个批归一化层（batch norm）之后使用特征选择（feature selection）将冗余的特征图剔除，同时对于最后一个卷积层只剪枝其输出特征图。因此，整个结构的输入、输出特征图通道数在剪枝前后保持不变。

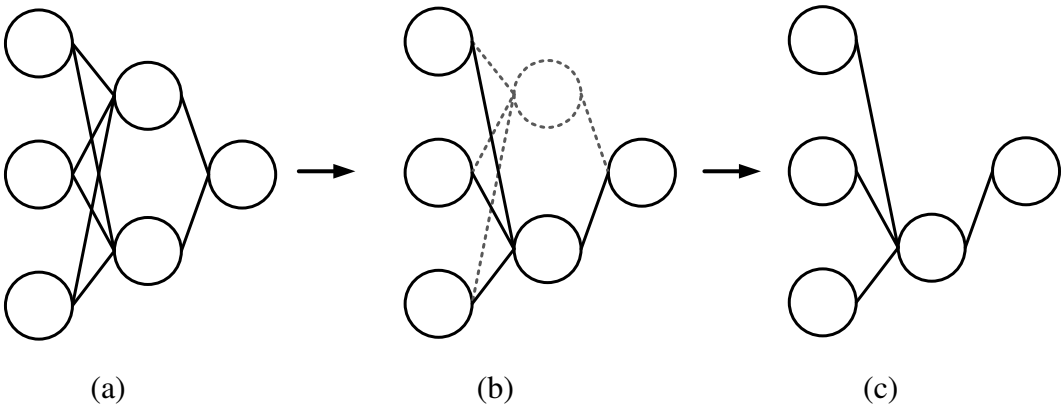


图 5.8 剪枝过程的三个主要步骤：(a) 训练一个大型过参数化的网络，一般在训练过程中用 L1 稀疏正则促使模型参数变得稀疏；(b) 剔除不重要的滤波器，得到剪枝后的模型；(c) 调优（finetune）剪枝后的模型以补偿兼职过程中丢失的性能。

数据集和数据增强 本节主要是在 CIFAR [63], SVHN [99], 和 ImageNet [112] 三个数据集上做图像分类实验来证明所提出方法的有效性。

对于 CIFAR 和 SVHN 数据集，实验根据很多文献的通用规则进行数据增强：首先在原始图像四周增加 4 个像素的边界，然后随机裁剪出 32×32 的图像块；对 CIFAR 数据集的图像做一次随机的左右翻转，由于 SVHN 数据集上是街景数字，因此不做左右翻转。

ImageNet 数据集上的数据增强稍微复杂一些。根据之前主流方法 [40, 39, 41, 116]的数据集增强策略：首先 resize 原图像，使得 resize 之后的最短边长度为 256，随后在 resize 之后的图像上随机裁剪出 224×224 的图像块。最后对裁剪之后的图像做一次随机的左右翻转。

虽有数据集上的数据在输入网络模型之前都要减去该数据集所有数据的均值以及除以方差进行归一化。

基础网络架构 本节的实验主要在 VGGNet [116] 和 ResNet [39] 网络上进行。根据 Network Slimming [87] 的实践经验，本届采用“预激活”（pre-act-resnet）网络结构 [41]。在 pre-act-resnet 网络中，激活函数在卷积层之前，网络中各层以“批归一化（BN，batch normalization），激活，卷积”的顺序展开。具体网络结构请参考 图 5.7。

超参数 除非特别说明， 公式(5.2) 中的阈值为 0.01，在所有实验中 $\Delta_\lambda = 10^{-5}$。网络使用随机梯度下降法（SGD）进行优化，动量参数（momentum）为 0.9，参数衰减因子（weight decay，也就是 L2 正则）系数为 10^{-4}。初试学习率为 0.1，在特定的阶段学习率下降十倍。在 CIFAR 数据集上一共训练 100 个 epochs，在 SVHN 数据集上训练 40 个 epochs。学习率分别在总 epochs 50% 和 75% 下降。在 ImageNet 数据集上总共训练 100 个 epochs，学习率每 30 个 epochs 下降一次。

ImageNet 上的半精度训练 由于 ImageNet 数据集数据量大，训练十分耗时, 本节所有的 ImageNet 实验都使用 APEX 软件库[4]进行半精度（float16）训练。除了 BN 层的参数用 float32 表示之外，其他所有层的参数以及输入输出数据均使用 float16 表示。在半精度加速下，在 4 个 Titan-2080 GPUs 上训练一个 ResNet-50 网络只需要 40 个小时。尽管使用半精度训练，试验中并没有发现明显的性能丢失。例如， 表格 5.4 中 Pre-act-resnet-50 网络取得了 76.27% 的 top-1 精确度，这个数字非常接近原文 [41] 以及文献 [95]给出的参考数据。

训练 → 剪枝 → 调优 本实验采用“训练 → 剪枝 → 调优”的三步法 图 5.8，该方法被很多相关的文献所采用的[44, 43, 87, 92]。

在试验中发现，在第一阶段的训练中，当学习率大的时候模型稀疏度 P 上升的很快。而当学习率下降之后，稀疏度上升的很慢，除非使用非常大的 L1 正则权重（λ）。因而，为了在第一阶段训练中快速促进模型稀疏，第一阶段训练中不降低学习率，仅在调优（finetune）阶段降低学习率。

[4] https://github.com/NVIDIA/apex

Model	**Methods**	**ratio r**	**Baseline accuracy**	**Finetune accuracy**
VGG11	Network Slimming [87]	0.5	92.13 (±0.18)	91.91 (±0.01)
	Ours		92.02 (±0.20)	92.17 (±0.18)
VGG16	Network Slimming	0.6	93.73 (±0.06)	93.65 (±0.04)
	Ours		93.57 (±0.26)	93.70 (±0.05)
VGG19	Network Slimming (from [88])	0.7	93.53 (±0.16)	93.60 (±0.16)
	Ours		93.66 (±0.26)	93.53 (±0.24)
Res56	SFP	0.4	93.59 (±0.58)	92.26 (±0.31)
	FPGM [43]		93.59 (±0.58)	92.93 (±0.49)
	Network Slimming		93.56 (±0.19)	93.33 (±0.14)
	Ours		93.73 (±0.10)	93.86 (±0.19)
	Network Slimming	0.5	93.56 (±0.19)	92.90 (±0.14)
	Ours		93.73 (±0.10)	93.62 (±0.16)
	Network Slimming	0.6	93.56 (±0.19)	91.94 (±0.10)
	Ours		93.73 (±0.10)	92.68 (±0.15)
Res110	SFP	0.4	93.68 (±0.32)	93.38 (±0.30)
	FPGM [43]		93.68 (±0.32)	93.73 (±0.23)
	Network Slimming		94.61 (±0.01)	94.49 (±0.12)
	Ours		94.43 (±0.13)	94.75 (±0.12)
	Network Slimming	0.5	94.61 (±0.01)	94.24 (±0.13)
	Ours		94.43 (±0.13)	94.52 (±0.26)
	Network Slimming	0.6	94.61 (±0.01)	93.47 (±0.15)
	Ours		94.43 (±0.13)	94.57 (±0.04)
Res164	Network Slimming (from [88])	0.4	95.04 (±0.16)	94.77 (±0.12)
	Ours		94.86 (±0.10)	95.01 (±0.15)
	Network Slimming	0.5	95.04 (±0.16)	94.52 (±0.09)
	Ours		94.86 (±0.10)	94.83 (±0.05)
	Network Slimming (from [88])	0.6	95.04 (±0.16)	94.23 (±0.21)
	Ours		94.86 (±0.10)	94.53 (±0.35)

表 5.1 CIFAR10 数据集上的实验结果。不论是在 VGGNet 还是 ResNet 网络上，本章所提出的“自适应稀疏正则损失”方法都能提升基准方法 Network Slimming (Network Slimming) [87] 的性能。

短连接结构的剪枝 在前激活残差网络（Pre-Act-ResNet）结构中，各运算符以“批归一化（BN）”→“ReLU”→“卷积（Conv）”的顺序执行。正如图 5.7，给定输入的特征图，在第一个 BN 层之后首先进行一次特征图筛选（feature selection）操作来剔除不重要的特征通道。此处各特征通道的重要性根据公式(5.10)确定。

对于第一个和第二个卷积层（Conv），本方法即对输入通道进行剪枝，也对输出通道剪枝（在图 5.7中，被点填充的方框表示剪枝的滤波器）。

为了保证残差结构输入/输出通道数的稳定，对于第三个卷积层（Conv），只对齐输入通道进行剪枝，输出通道保持不变。这样，整个残差结构在剪枝完之后输入通道不变，输出通道也不变，可以无缝插入到原网络中使用。值得注意的是，当计算模型的稀疏度（公式(5.3)）的时候，第三个卷积层并没有计入其中，因为它的输出通道并没有被剪枝。例如 图 5.7的情况下就只有 2 个滤波器被剪枝了，第三个卷积层不算入其中。

5.5.2 CIFAR 数据集的实验结果

本研究首先在 CIFAR10 和 CIFAR100 数据集上验证所提出的算法。 CIFAR 上的实验主要用不同深度的 VGGNets 和 ResNets 两种网络结构。由于 CIFAR 实验上的实验结果不稳定，本实验记录每 10 次实验结果的均值和方差。正如前文所介绍的，Network Slimming[87]由于使用全局排序来筛选被剪枝的滤波器，因此得到的结果十分不稳定。因此对于 Network Slimming 方法，如果精度低于均值的 10% 则丢弃该试验的数据并重新做一次实验。 CIFAR 数据集上的定量对比记录在 表格 5.1 和 表格 5.2 中。另外，本文对比了不同剪枝比率下 Network Slimming 和本文所提出方法的性能，分类准确率和剪枝比率的曲线图记录在 图 5.9

VGGNet 网络 首先做的是 VGGNet 网络上的实验。 VGGNet 网络是顺序结构，没有跳层连接或者是短连接，因此很方便剪枝。实验发现，在 VGGNet 网络中即便是用很大的剪枝比率也只带来了微小的性能下降。以 VGG-19 网络为例，在 CIFAR10 数据集上即便是剪掉 70% 的滤波器，Network slimming 和本文提出的方法均在原模型基础上提高了性能。并且，提升模型深度并不一定意味着性能的提升，因为在 CIFAR100 数据集上的结果中，VGG16 取得了比 VGG19 更好的（或者相当的）性能。以上实验结果表明：

Model	Methods	ratio r	Baseline accuracy	Finetune accuracy
VGG11	Network Slimming [87]	0.3	69.33 (±0.26)	66.54 (±0.14)
	Ours		68.24 (±0.11)	67.84 (±0.11)
VGG16	Network Slimming	0.3	73.50 (±0.18)	73.36 (±0.28)
	Ours		72.16 (±0.23)	73.59 (±0.37)
	Network Slimming	0.4	73.50 (±0.18)	N/A
	Ours		72.16 (±0.23)	73.59 (±0.23)
VGG19	Network Slimming (from [88])	0.5	72.63 (±0.21)	72.32 (±0.28)
	Ours		71.19 (±0.54)	72.48 (±0.28)
Res164	Network Slimming (from [88])	0.4	76.80 (±0.19)	76.22 (±0.20)
	Ours		76.43 (±0.26)	77.74 (±0.17)
	Network Slimming	0.6	76.80 (±0.19)	74.17 (±0.33)
	Ours		76.43 (±0.26)	76.28 (±0.27)

表 5.2 CIFAR100 数据及上的实验结果。 'N/A' 表示该设置下所有的实验都失败了，因此不记录实验数据。 Experimental results on the CIFAR100 dataset. 本文所提出的方法在几乎所有的设置条件下均优于 Network Slimming 方法，特别是当网络为 ResNet-164 时，本文所提出方法比 Slimming 方法高出近 2%。

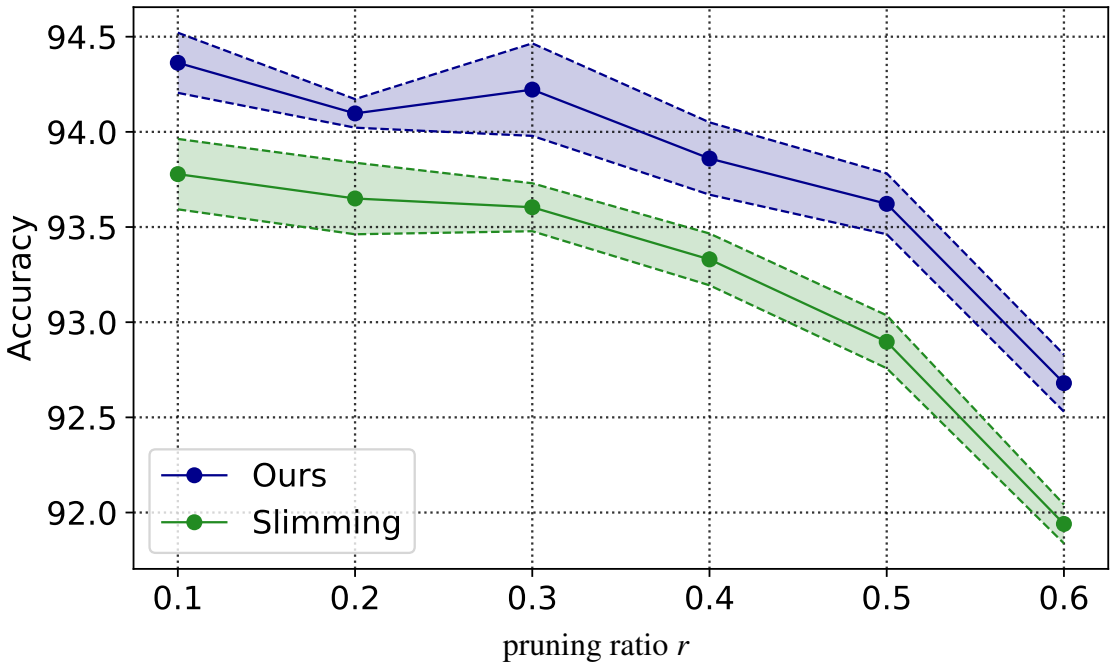


图 5.9 不同剪枝比率 r 下 ResNet-56 的性能对比。曲线中心表示均值，粗度表示方差。本文所提出方法在各项对比中均要优于 Network Slimming。

1. 在 CIFAR 数据集上由于数据及简单，而 VGG 网络参数相对较大，因此 VGG 网络在 CIFAR 数据集上处于过参数化（over parameterized）状态；
2. 因此在 CIFAR 数据集上 VGGNet 网络可以承受很大的滤波器剪枝比率。

ResNet 网络的实验结果 在 ResNet 网络结构上剪枝要比 VGGNet 网络更复杂，主要是 ResNet 网络要处理层的残差连接。根据 5.5.1和 图 5.7，在处理 resnet 时只对瓶颈结构（bottleneck structure）内部的卷积层进行剪枝，保证剪枝前后瓶颈结构的输入、输出特征图通道数不变。在保持相同的剪枝比率的情况下，基于自适应正则损失函数的方法都要显著优于 Network Slimming [87] 方法。

5.5.3 SVHN 数据集上的结果

本节对比 ResNet 网络在 SVHN 数据集上的剪枝结果。从 表格 5.3中的实验可以看出，本文所提出基于“自适应稀疏正则损失函数”的剪枝方法在不同设置条件和网络深度写均优于基准方法 [87]

同时，Network Slimming [87] 在剪枝比率很高比如 $r = 80\%$ 的时候经常会得不到正常的剪枝后模型，或者是剪枝后模型性能非常差。但是本章所提出的方法即使在剪枝比率很高的情况下也能够得到保持一定分类准确率的压缩后模型。例如，在 ResNet-56 网络中，即便是 80% 的滤波器被剪枝掉了，相比原始模型本章所提出的方法只有 0.10% 的精度丢失。

另外，与 CIFAR 数据及集上的实验类似，剪枝掉一定比例的滤波器甚至可能带来额外的精度提升（例如剪枝比率为 20% 或 40% 时）。这说明 ResNet56 网络相对于 SVHN 数据集来说也“过参数化”了，网络参数过多，因此合适的剪枝比率能够减轻这种过参数化，带来性能提升。

5.5.4 ImageNet 数据集的结果

本节在大型数据分类数据集 ImageNet [112] 上评测本章所提出方法和其他模型剪枝方法的性能。

Network Slimming [87] 和本章所提出方法的结果都是重新复现得到的，其他的结果都由原文献中获取。本节对比了一些最近提出来的模型剪枝方法，比如用参数幅值作为剪枝依据的文献 [77]，另外还有2019年提出的数据相关的剪枝方法 [95]，以及几个基于批归一化层缩放因子的剪枝方法 [87, 144]。本节的所有实验结果都总结在 表格 5.4，并根据剪枝比率，或者是计算开销（FLOPs）

Model	Methods	ratio r	Baseline accuracy	Finetune accuracy
Res20	Network Slimming [87]	0.2	95.85 (±0.07)	95.82 (±0.18)
	Ours		95.85 (±0.07)	96.18 (±0.09)
	Network Slimming [87]	0.4	95.85 (±0.07)	95.77 (±0.13)
	Ours		95.85 (±0.07)	96.20 (±0.11)
	Network Slimming [87]	0.6	95.85 (±0.07)	95.66 (±0.07)
	Ours		95.85 (±0.07)	96.15 (±0.05)
	Network Slimming [87]	0.8	95.85 (±0.07)	N/A
	Ours		95.85 (±0.07)	95.49 (±0.13)
Res56	Network Slimming [87]	0.2	96.87 (±0.04)	96.62 (±0.05)
	Ours		96.87 (±0.04)	97.04 (±0.08)
	Network Slimming [87]	0.4	96.87 (±0.04)	96.56 (±0.07)
	Ours		96.87 (±0.04)	97.00 (±0.02)
	Network Slimming [87]	0.6	96.87 (±0.04)	N/A
	Ours		96.87 (±0.04)	97.03 (±0.02)
	Network Slimming [87]	0.8	96.87 (±0.04)	N/A
	Ours		96.87 (±0.04)	96.77 (±0.05)

表 5.3 SVHN 数据集上的图像分类实验结果。“N/A”表示在 10 次试验中所有的实验都无法得到可靠的模型。相比 Netwoek Slimming (Network Slimming) [87]，本章提出的方法能够接受很高的剪枝比率（例如 80%），并且在各种剪枝比率和配置中表现都优于 Netwoek Slimming 方法。

以及模型参数数目（Params）分组比较。

表格 5.4的结果表明，在计算开销和模型参数类似的条件下，本文所提出的方法性能显著优于基准方法 Network Slimming [87]，同时也高于其它剪枝方法例如 [144, 77]。即便是和数据相关的剪枝方法 [95]相比，本文所提出的方法也取得了与之相当的性能。

5.5.5 消融实验

本节通过消融实验（ablation studies）来验证本章所提出算法中的每一个模块的作用。本节的所有实验都是在 CIFAR100 数据集上完成的。

考虑前后相关性的滤波器重要性估计

本消融实验主要为了证明本文所提出的“考虑前后关系的滤波器重要性估计算法”的有效性。该方法的有效性可以通过两个方面辅证：

模型	**Methods**	**ratio** r	**Acc.** (%)	**#Params** (10^7)	**#FLOPs** (10^6)
VGG11	Baseline	-	70.84	3.18	7.61
	Network Slimming [87]	0.50	68.62	1.18	6.93
	Ours	0.50	69.12	1.18	6.97
Res50	Baseline	-	76.27	2.56	4.13
	ThiNet [92]	0.50	71.01	1.24	3.48
	ThiNet	0.70	68.42	0.87	2.20
	Li *et al.* [77]	N/A	72.04	1.93	2.76
	Network Slimming	0.50	71.99	1.11	1.87
	Ours	0.50	72.41	1.07	1.86
	Taylor [95]	0.19	75.48	1.79	2.66
	Network Slimming	0.20	75.12	1.78	2.81
	Ours	0.20	75.37	1.76	2.82
Res101	Baseline	-	77.37	4.45	7.86
	Ye *et al.* [144]-v1	N/A	74.56	1.73	3.69
	Ye *et al.* [144]-v2	N/A	75.27	2.36	4.47
	Taylor [95]	0.45	75.95	2.07	2.85
	Network Slimming	0.50	75.97	2.09	3.16
	Ours	0.50	76.54	2.17	3.23
	Taylor [95]	0.25	77.35	3.12	4.70
	Ours	0.20	77.36	3.18	4.81

表 5.4 在 ImageNet 数据集上的图像分类实验结果。本章所提出的方法在不同剪枝比率，模型参数量和计算开销下的比较中都取得了高于基准方法的性能，同时也优于一些主流剪枝方法 [92, 144, 87, 77], 即便与数据相关的剪枝方法 [95]相比，本章所提出的方法也取得了与之相当的性能。

1. 剪枝之后的性能下降的程度。如果滤波器重要性估计方法更好，那么就能更精准地找打不重要的滤波器，在剪枝之后对模型性能的影响也就更小。
2. 调优（finetune）之后的性能。滤波器重要性估计算法越精准，越能找到真正贡献低的滤波器，在剪枝之后能得到更好的模型结构，因此调优之后的模型性能也会更高。

在同一个剪枝比率 $r = 0.5$ 下，假设滤波器重要性算法越精准，模型剪枝后（调优前）的性能就越高，因此同样剪枝比率下“剪枝后调优前”模型性能可以说明重要性估计的优劣。本消融实验对比一下三种不同的方法：

1. （a）原始 Network Slimming [87] 方法只用 BN 层缩放系数度量滤波器重要性；

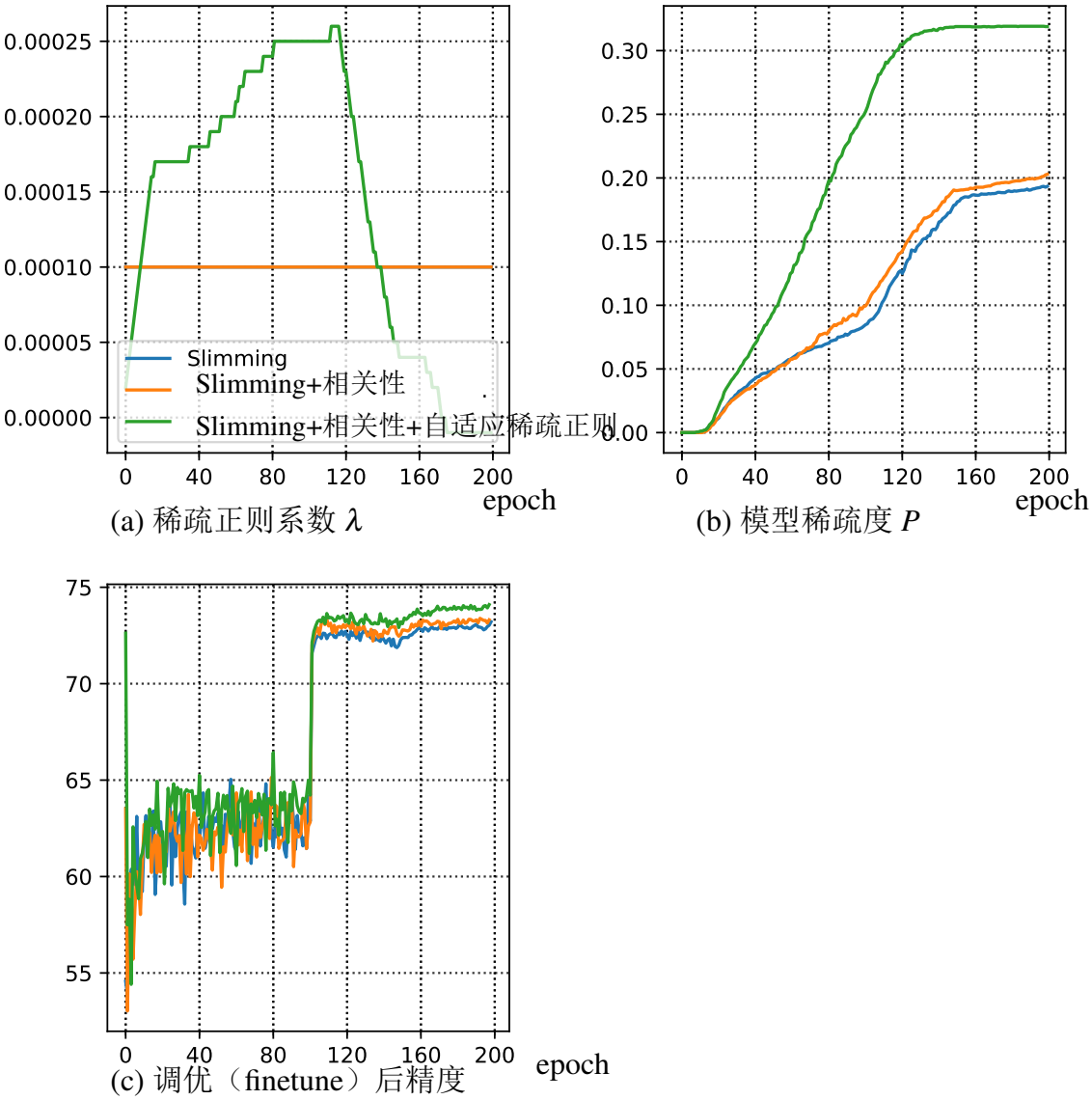


图 5.10 三种不同方法训练中的动态曲线图。（a）为第一阶段训练过程中稀疏正则系数 λ 的变化情况，（b）为模型稀疏度（公式(5.3)中的 P）的变化情况；（c）为调优（finetune）过程中的模型精确度。相比原始的 Network Slimming，使用考虑前后相关性的滤波器重要性估计算法能够更精准地找到不重要的滤波器，这体现在第一阶段训练完成并剪枝后的模型性能更高。在此基础上使用了本文提出的**自适应稀疏正则函数控制**之后，不仅在剪枝后得到的模型性能最高，而且调优后的模型性能也是最好的。这说明本章所提出的两种策略均对剪枝性能有有效提升。

2. （b）Network Slimming 方法 + 考虑前后关系的滤波器重要性度量（公式(5.10)）；
3. （c）Network Slimming 方法 + 考虑前后关系的滤波器重要性度量（公

模型	方法	剪枝比率 r	调优（finetune）前	调优（finetune）后
VGG16	Network Slimming	0.3	52.19 (±6.82)	73.36 (±0.28)
	Network Slimming+前后相关	0.3	61.19 (±6.18)	73.57 (±0.31)
	Network Slimming+前后相关+自适应稀疏正则	0.3	72.83 (±0.26)	73.59 (±0.37)
Res56	Network Slimming	0.5	1.41 (±0.25)	71.13 (±0.26)
	Network Slimming+前后相关	0.5	5.29 (±1.01)	73.62 (±0.14)
	Network Slimming+前后相关+自适应稀疏正则	0.5	55.29 (±1.92)	74.53 (±0.10)

表 5.5 不同策略下剪枝前、后的模型性能。

式(5.10)）+ 自适应稀疏正则损失（算法 5.6）。

首先在 VGGNet-16 网络结构上做一个展示性实验，剪枝比率为 $r = 0.3$，实验结果记录在 图 5.10。 图 5.10的三种方法的曲线表明，相比原始的 Network Slimming，使用考虑前后相关性的滤波器重要性估计算法能够更精准地找到不重要的滤波器，这体现在第一阶段训练完成并剪枝后的模型性能更高。在此基础上使用了本文提出的**自适应稀疏正则函数控制**之后，不仅在剪枝后得到的模型性能最高，而且调优后的模型性能也是最好的。这说明本章所提出的两种策略均对剪枝性能有有效提升。

除此之外，本节还分别以 VGGNet-16 和 ResNet-56 网络为基础模型在 CIFAR100 数据集上进行了 10-fold 交叉验证试验，实验统计结果在 表格 5.5中。表格 5.5 中的结果表明： 1）考虑相关性的滤波器重要性估计算法能够更有效地找到冗余的滤波器，剪枝后的模型性能更高； 2）自适应稀疏正则函数能够在第一阶段训练过程中有效控制模型稀疏度，使得训练完成之后恰好有指定数目的滤波器处于稀疏状态；因此剪枝对模型的性能影响十分小，剪枝后能得到更鲁棒的网络结构，调优后得到更好的精确度。

自适应稀疏正则函数 **v.s.** *自适应剪枝阈值* τ

为了保证第一阶段训练完成之后能达到指定的模型稀疏度，有两种不同的方法：

1. 固定稀疏性阈值 τ，然后在训练过程中通过自适应地调整稀疏正则损失函数的系数 λ 来达到指定的稀疏度；
2. 第一阶段训练使用固定的稀疏正则系数 λ，然后在训练完成之后搜寻一个

合适的稀疏性阈值 τ 使得模型达到指定的稀疏度。

本章采用了 a) 方法，为了证明 a) 方法比 b)方法更有效，本节的消融实验对比这两种方法的性能。根据 Network Slimming [87], 在 b) 方法中，稀疏正则函数的系数 λ 设定为 10^{-5}，本实验的剪枝比率设定为 0.5。 表格 5.6中的实验结果表明，a) 方法表现更好，这体现在： a)方法剪枝后的性能更高，这说明剪枝对原模型的影响小，找到了更稀疏的滤波器； a)方法调优后性能更佳，这说明剪枝得到的模型更鲁棒，模型结构更好。

更好的剪枝后模型结构

在 5.3.1节中曾提到，由于 Network Slimming [87]对滤波器重要性作全局排序来确定滤波器去留，有时候同一层中的大量滤波器被剪枝，因此可能得到非常不稳定的网络模型。本消融实验验证本文所提出的剪枝策略能够得到更稳定鲁棒的网络模型。为了只测试剪枝后的模型结构的优劣，本实验将剪枝后的网络参数重新随机初始化，这样不同方法仅仅是网络结构不一样。实际上这种训练策略和文献 [88] 中的“scratch-E”一样，即从头开始训练剪枝后的模型。

表格 5.7中的结果表明，本文所提出的滤波器筛选策略能够得到更稳定的网络结构，因为调优后的网络性能远高于 Network Slimming方法的性能。

剪枝的稳定性

5.3.1中曾提到 Network Slimming 有时会剪枝得到很不稳定，性能很低的网络模型，特别是当剪枝比率很高的时候。本实验验证这一点，并证明本文所提出的方法在即便是剪枝比率很高的情况下也不会，或者是很少出现极端不稳定模型结构的情况。

本节一共设计了两个实验。第一个试验中给出一个典型的 Network Slimming 得到的不稳定模型的例子。该模型由 VGG16 网络以 $r = 0.4$ 的剪枝比率得到， 图 5.11展示了该模型中不同层剩余滤波器个数的分布。 如 图 5.11所示，Network Slimming 剪枝得到的模型最小的层只保留了 1 个滤波器，也就是说将

Method	剪枝前 （第一阶段训练完）	阈值 p	调优前 （剪枝后）	调优后
(a)	60.86	0.01	60.86	75.24
(b)	73.59	0.41	1.53	74.36

表 5.6 两种不同达到指定剪枝比率方法 a) 和 b) 的对比。

模型	方法	准确率 (%)		
		Baseline	调优	从头训练 （随机初始化参数）
Res164	Network Slimming	76.80 (±0.19)	74.17 (±0.33)	75.05 (±0.08)
	Ours	76.43 (±0.26)	76.43 (±0.27)	76.41 (±0.32)

表 5.7 从头训练剪枝后模型的性能对比。所有模型在剪枝之后参数都被随机初始化，这样最后的性能差异仅由模型结构差异造成。本文所提出的方法性能优于 Network Slimming，这意味着剪枝得到的模型结构更稳定。

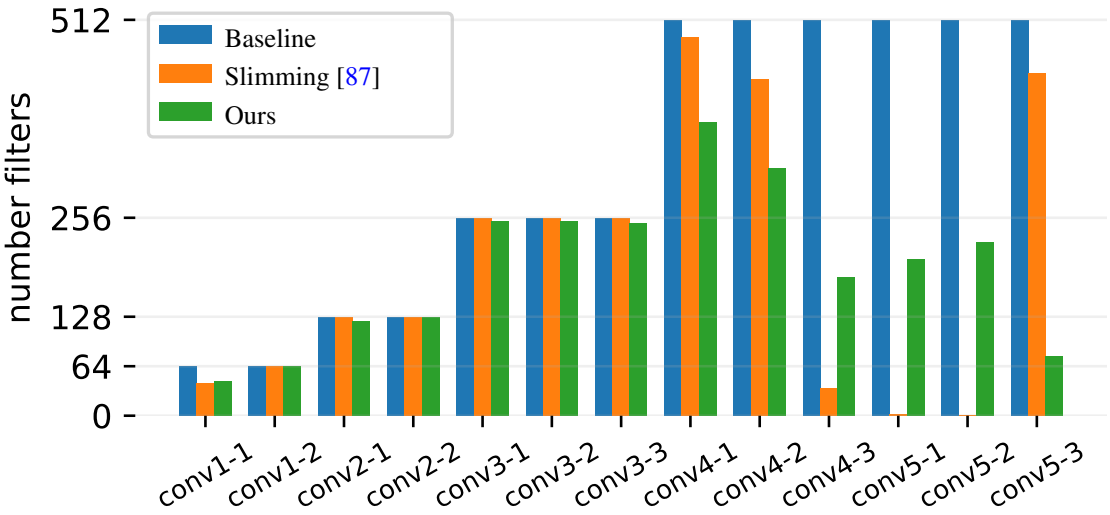


图 5.11 VGG-16 网络剪枝后各层滤波器分布情况。 Network Slimming [87] 方法得到的剪枝后模型十分不稳定，部分层只剩下 1-2 个滤波器。

高维特征图压缩到一维空间，这极大地损失了深度特征中的语义信息，最终模型的性能会受到很大的影响。而同样的剪枝比率下，用本章所提出方法得到的剪枝后模型滤波器分布比较均匀，并没有出现极端的分布不均情况。

第二个实验将同样的剪枝过程重复五次，并将每次的结果记录在表格 5.8中。 表格 5.8中的 (X/Y)表示剪枝后各层中最少的滤波器有 X 个，最终模型的分类准确度为 Y。 表格 5.8中的实验结果表明，即使在很高的剪枝比率下，本章所提出的“自适应正则损失函数”方法仍然能够得到完整的剪枝后模型，并且剪枝后模型的分类准确度也保持的不错。但是 Network Slimming 方法在所有的试验中都无法得到完整的剪枝后模型，这体现在所有的实验都存在某一层所有的滤波器都被剪枝掉的情况，因此网络模型被破坏，无法进行预测和训练，分类精度等于随机猜测。

数据集	方法	剪枝比率 r	run-1	run-2	run-3	run-4	run-5
CIFAR10	Network Slimming	0.7	10.00 / 0	10.00 / 0	10.00 / 0	10.00 / 0	10.00 / 0
	Ours		93.93 / 24	93.66 / 25	93.94 / 27	93.70 / 23	93.89 / 27
CIFAR100	Network Slimming	0.4	1.00 / 0	1.00 / 1	1.00 / 0	1.00 / 0	1.00 / 0
	Ours		73.24 / 29	73.60 / 37	73.92 / 35	73.47 / 37	73.71 / 37

表 5.8 VGGNet-16 网络在 CIFAR100 数据集上的 5 次实验结果记录。(·/·) 表示调优之后的模型性能和**各层中保留的最少的滤波器数目**。

第六节 小结

本章提出了一种自适应的 L1 正则损失函数，在模型剪枝过程中自动调整模型的稀疏正则系数，以便模型在第一阶段训练完成之后能够达到指定的稀疏度，降低剪枝后对模型性能的影响。与此同时，还提出了一个基于模型中前后层关系的滤波器重要性估计算法，与文献[87] 中只是用批归一化层的缩放系数来确定滤波器重要性相比，本章所提出的重要性估计算法能够更精准地找到冗余的滤波器，使得剪枝后的模型精度更高。在多个数据集上的大量实验表明，本章所提出的方法能够显著提升模型剪枝方法的性能，在与文献[87] 的对比试验中取得优异的实验结果。本章节相关内容已经投稿到 IEEE Transaction on Cybernetics。

第六章 总结与展望

本文研究工作的核心是卷积神经网络优化目标函数的设计，包括针对具体任务评价指标的损失函数设计，以及针对应用场景的正则函数设计和自适应调整。主要的研究方法是通过分析不同计算机视觉应用中对模型表现和输出结果的不同要求，设计合适的损失函数代替原模型中广泛使用的交叉熵损失，并用实验结果证明所提出优化目标函数的有效性。主要的研究成果包括：

1. 基于放缩 F-measure 的损失函数。将该损失函数应用到显著性检测任务重，可以统一显著性检测中训练和测试之间的优化目标，同时前背景对比鲜明的显著性检测结果；
2. 基于中心间角度距离的互斥正则函数。该正则函数可以显示地增强神经网络不同类别特征之间的类间离散度；
3. 自适应的稀疏正则损失函数。该稀疏正则函数可以在训练过程中根据预定的稀疏比率自动调节稀疏正则损失函数的系数。

作者将所提出的损失函数/正则函数分别应用到显著性检测、人脸识别和卷积神经网络模型剪枝中，取得了良好的结果。

第一节 工作总结

随着图像传感器的普及，人类每天都要获取大量的图像和视频数据。如何让机器智能地从这些图像和视频数据中提取有用信息，帮助人类进行分析、决策和预警，是当今人工智能和计算机视觉研究的重点之一。深度学习技术作为人工智能和计算机视觉的核心工具，已经广泛运用于语音识别、计算机视觉、自然语言处理等多个领域，取得了远超以往方法的性能。然而深度学习算法依赖于大量的有标注的数据，深度神经网络只有从大量的已标注数据中学习和不断更新参数，才能从一个随机系统变成能够做出准确预测和决策的智能系统。损失函数在深度神经网络的优化过程中主要负责量化模型预测与真实标签之间的误差，并将此误差反向回传到网络各层，指导参数更新。因此，损失函数是神经网络优化过程中的“老师”，告诉神经网络什么时候预测对了，什么时候预测错了，应该往什么方向更新参数。

本文的研究主要以神经网络优化目标函数（objective target function）为核心展开。由于深度学习率先在图像分类（image classification）任务中发力，并迅速被应用到计算机视觉的方方面面，因此很多其他计算机视觉任务中仍然借用了图像分类中的“交叉熵损失”。然而不同任务对模型的表现是不一样的，有的任务并不以分类误差为评价指标。因此针对不同任务的特征设计类别相关的个性化损失函数，可以使得训练过程中的优化方向和模型应用和测试阶段的目标一致，提升模型的性能。

本文在第一章简单介绍了相关的研究背景，第二章介绍了几个典型的任务相关的损失函数设计的相关工作。在第三章，本文针对目前显著性检测方法训练和测试阶段优化目标不一致、测试阶段需要在测试集合上调试超参数以及输出的显著性图在物体边缘地带比较模糊的几个缺点，将显著性检测的评测指标F-measure 放缩并使其成为可导函数，并在训练阶段直接以 F-measure 作为优化目标，提出在训练阶段直接以评测指标作为目标函数。由于显著性检测的评测指标 F-measure 是一个不可导函数，本文设计了一个对 F-measure 的近似函数，该函数求得 F-measure 的一个连续的下确界。因此可以利用该函数在训练过程中直接最大化 F-measure，实现训练和测试阶段的目标函数统一化。在对所提出的损失函数进行梯度分析时发现，该损失函数在饱和区域仍然具有很大的梯度，因此在训练阶段可以持续促进模型产生两极分化的预测结果。因此本文所提出的方法在不需要任何后处理步骤的情况下能得到前背景对比非常鲜明的显著性图。同时，本文所提出的损失函数收敛速度块，只需要交叉熵损失大约1/3的迭代次数即可收敛到很好的结果，具有一定的实用价值。

在第四章，作者针对目前的人脸识别算法中使用的损失函数都只强调了同类特征之间的“类间紧凑性”，而忽略了类间样本应该保持一定的“类间离散度”，本文设计了“互斥正则损失函数”。该损失函数通过显示地将不同类别特征的中心往互相远离的方向推动，增大不同类别样本特征之间的距离，这个过程就像同极磁铁互相排斥一样，因此称为“互斥正则损失函数”。具体地，互斥正则损失函数以分类层各类的权重向量为类别中心，在训练过程中惩罚每一类的中心与其最近邻之间的夹角。因此在每次迭代过程中类别中心都将往远离最近邻的方向移动一小步，最终各类别的特征将均匀地分布在角度空间中，增大了不同类别样本特征之间的距离。实验表明互斥正则损失能够在现有方法基础上提升特征的区分度，特别是样本特征类间的离散度，从而提升现有人脸识

别模型的性能。

第五章，本文设计了一种“自适应的稀疏正则损失函数”，该损失函数可以在训练过程中根据事先确定的稀疏率自动调节稀疏正则损失的系数，使得训练完的模型满足特定的稀疏性，同时保持模型的性能。这样，在模型压缩过程中只需要实现确定压缩比率，在训练中损失函数就会自动根据当前模型的稀疏度和目标压缩比自动调节损失函数中稀疏正则损失的权重，训练完成之后将稀疏的滤波器直接剔除，将对模型性能的影响最小化，实现无伤剪枝。实验证明作者提出的自适应正则损失函数相比使用固定的稀疏正则系数能够显著保留原始模型的性能，并获得更稳定的模型结构，最终提升调优（finetune）后模型的性能。

第二节 展望

神经网络的优化和损失函数研究是一个很大的话题，其中损失函数、正则函数以及优化器的设计都对模型最后的表现有很大的影响。本文针对几个特定任务中的损失函数和正则函数的研究仅仅是这个领域很小的一个方面，同时也只是一个开端。作者认为相关领域有以下几点可以持续挖掘和拓展：

1）第三章涉及的基于 F-measure 的损失函数虽然能够使得模型产生前后背景分明的显著图，一定程度上缓解测试阶段最佳阈值不可通用的问题，但是并没有从根源上避免在测试集上搜索超参数的。如果损失函数能够模拟测试阶段的阈值二值化和 F-measure 计算过程，就可以在训练阶段学习到测试时需要的所有超参数，彻底避免在测试阶段还需要搜寻最佳阈值，同时完全统一训练和测试过程的计算流程。

2）第四章涉及的互斥正则损失函数虽然通过惩罚类别间的角度距离保证了类别间的距离，但是不同类别间的类间离散度有可能是不一样的。因为类别间的语义关系是有远有近的，因此针对不同的类间关系允许不同的类间离散度，或许能学习到更具有区分度、和人类认知更一致的特征。

3）第五章涉及到的自适应稀疏正则损失函数是以 L1 正则函数为基础涉及的，从优化上来讲，L0 norm 或者是 L0.5 norm 比 L1 正则更能促使模型的稀疏化，如果能对 L0 norm或者其他非凸约束作连续近似，就能够在更低的正则系数的情况下得到更稀疏的模型参数，增大剪枝比率的同时保持模型性能。

4）随着智能化以及端到端（end-to-end）思想的进一步深化，深度学习系

统中很多原本需要人手工设定的参数都可以用数据驱动的方法自动学习到。例如之前很多经典的网络结构例如 VGGNet、ResNet都是由人类专家根据经验手工设计层数，层内的连接以及每层有多少个滤波器。现在的发展潮流是使用智能算法，例如强化学习或者是进化算法自动学习和确定这些参数。因此将来的一个可以拓展的点就是设计学习算法从不同任务的数据中自动学习损失函数，或者确定损失函数的超参数。一个可行的方案就是设计一个“母损失函数”，该损失函数具有很多的超参数，不同的超参数组合可以特化成特定的一种损失函数。然后在训练过程中用智能算法自动学习这些超参数，从而实现损失函数设计的自动化。以上所述第 2）点就可以采取这种方法，在训练过程中自动确定不同类别间的类间离散度。

参考文献

[1] Achanta R, Hemami S, Estrada F, et al. Frequency-tuned salient region detection. In: CVPR, 2009: 1597 $\sim$ 1604.

[2] Achanta R, Estrada F, Wils P, et al. Salient region detection and segmentation. In: International conference on computer vision systems, 2008: 66 $\sim$ 75.

[3] Achanta R, Shaji A, Smith K, et al. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence, 2012, 34(11): 2274 $\sim$ 2282.

[4] Anwar S, Hwang K and Sung W. Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2017, 13(3): 32.

[5] Baum E B, Wilczek F. Supervised learning of probability distributions by neural networks. In: Neural information processing systems, 1988: 52 $\sim$ 61.

[6] Berman M, Rannen Triki A and Blaschko M B. The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 4413 $\sim$ 4421.

[7] Borji A, Cheng M.-M, Hou Q, et al. Salient object detection: A survey. Computational Visual Media, 2019, 5(2): 117 $\sim$ 150.

[8] Boyd S, Xiao L and Mutapcic A. Subgradient methods. Lecture notes of EE392o, Stanford University, 2003.

[9] Bromley J, Guyon I, LeCun Y, et al. Signature verification using a" siamese" time delay neural network. In: Advances in neural information processing systems, 1994: 737 $\sim$ 744.

[10] Cao Q, Shen L, Xie W, et al. Vggface2: A dataset for recognising faces across pose and age. In: IEEE International Conference on Automatic Face & Gesture Recognition, 2018: 67 $\sim$ 74.

[11] Carreira-Perpinán M A, Idelbayev Y. Learning-Compression: Algorithms for Neural Net Pruning. In: IEEE conf Computer Vision and Pattern Recognition. 2018: 8532 $\sim$ 8541.

[12] Chen L.-C, Papandreou G, Kokkinos I, et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 2017, 40(4): 834 $\sim$ 848.

[13] Chen W, Wilson J, Tyree S, et al. Compressing neural networks with the hashing trick. In: International conference on machine learning, 2015: 2285 $\sim$ 2294.

[14] Cheng M.-M, Mitra N J, Huang X, et al. Global contrast based salient region detection. IEEE PAMI, 2015, 37(3): 569 $\sim$ 582.

[15] Cheng M.-M, Liu Y, Hou Q, et al. HFS: Hierarchical feature selection for efficient image segmentation. In: European conference on computer vision, 2016: 867 ∼ 882.
[16] Chin T.-W, Zhang C and Marculescu D. Layer-compensated pruning for resource-constrained convolutional neural networks. arXiv preprint arXiv:1810.00518, 2018.
[17] Clevert D.-A, Unterthiner T and Hochreiter S. Fast and accurate deep network learning by exponential linear units (elus). In: 2016.
[18] Courbariaux M, Hubara I, Soudry D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830, 2016.
[19] Dai X, Zhang P, Wu B, et al. Chamnet: Towards efficient network design through platform-aware model adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019: 11398 ∼ 11407.
[20] Deng J, Guo J, Xue N, et al. Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019: 4690 ∼ 4699.
[21] Deng J, Ding N, Jia Y, et al. Large-scale object classification using label relation graphs. In: European conference on computer vision, 2014: 48 ∼ 64.
[22] Denton E L, Zaremba W, Bruna J, et al. Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems, 2014: 1269 ∼ 1277.
[23] Ding C, Tao D. Robust face recognition via multimodal deep face representation. IEEE Transactions on Multimedia, 2015, 17(11): 2049 ∼ 2058.
[24] Ding X, Ding G, Guo Y, et al. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In: IEEE conf Computer Vision and Pattern Recognition. 2019: 4943 ∼ 4953.
[25] Fahlman S E. An empirical study of learning speed in back-propagation networks. Carnegie Mellon University, Computer Science Department, 1988.
[26] Frogner C, Zhang C, Mobahi H, et al. Learning with a Wasserstein loss. In: Advances in Neural Information Processing Systems, 2015: 2053 ∼ 2061.
[27] Fukushima K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural networks, 1988, 1(2): 119 ∼ 130.
[28] Girshick R. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, 2015: 1440 ∼ 1448.
[29] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014: 580 ∼ 587.
[30] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010: 249 ∼ 256.

[31] Guo C, Ma Q and Zhang L. Spatio-temporal saliency detection using phase spectrum of quaternion fourier transform. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008: 1 ∼ 8.

[32] Guo Y, Zhang L, Hu Y, et al. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In: European Conference on Computer Vision. 2016: 87 ∼ 102.

[33] Hadsell R, Chopra S and LeCun Y. Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006: 1735 ∼ 1742.

[34] Han J, Chen H, Liu N, et al. CNNs-based RGB-D saliency detection via cross-view transfer and multiview fusion. IEEE transactions on cybernetics, 2017, 48(11): 3171 ∼ 3183.

[35] Han S, Mao H and Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.

[36] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems. 2015: 1135 ∼ 1143.

[37] Harel J, Koch C and Perona P. Graph-based visual saliency. In: Advances in neural information processing systems, 2007: 545 ∼ 552.

[38] Hassibi B, Stork D G. Second order derivatives for network pruning: Optimal brain surgeon. In: Advances in neural information processing systems, 1993: 164 ∼ 171.

[39] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016: 770 ∼ 778.

[40] He K, Zhang X, Ren S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: IEEE International Conference on Computer Vision. 2015: 1026 ∼ 1034.

[41] He K, Zhang X, Ren S, et al. Identity mappings in deep residual networks. In: European Conference on Computer Vision. 2016: 630 ∼ 645.

[42] He S, Lau R W, Liu W, et al. Supercnn: A superpixelwise convolutional neural network for salient object detection. International journal of computer vision, 2015, 115(3): 330 ∼ 344.

[43] He Y, Liu P, Wang Z, et al. Filter pruning via geometric median for deep convolutional neural networks acceleration. In: IEEE conf Computer Vision and Pattern Recognition. 2019: 4340 ∼ 4349.

[44] He Y, Kang G, Dong X, et al. Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. In: 2018: 2234 ∼ 2240.

[45] He Y, Zhang X and Sun J. Channel Pruning for Accelerating Very Deep Neural Networks. In: IEEE International Conference on Computer Vision. 2017: 1398 ∼ 1406.

[46] Hertz J A. Introduction to the theory of neural computation. CRC Press, 2018.

[47] Hoffer E, Ailon N. Deep metric learning using triplet network. In: International Workshop on Similarity-Based Pattern Recognition, 2015: 84 $\sim$ 92.
[48] Hou Q, Cheng M.-M, Hu X, et al. Deeply supervised salient object detection with short connections. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017: 3203 $\sim$ 3212.
[49] Hou X, Zhang L. Saliency detection: A spectral residual approach. In: 2007 IEEE Conference on computer vision and pattern recognition, 2007: 1 $\sim$ 8.
[50] Hu H, Peng R, Tai Y.-W, et al. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint arXiv:1607.03250, 2016.
[51] Hu P, Shuai B, Liu J, et al. Deep level sets for salient object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017: 2300 $\sim$ 2309.
[52] Hu X, Zhu L, Qin J, et al. Recurrently aggregating deep features for salient object detection. In: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[53] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks. In: IEEE conf Computer Vision and Pattern Recognition. 2017: 4700 $\sim$ 4708.
[54] Huang G B, Ramesh M, Berg T, et al. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments, Oct. 2007.
[55] Huang Z, Wang N. Data-driven sparse structure selection for deep neural networks. In: Proceedings of the European conference on computer vision (ECCV), 2018: 304 $\sim$ 320.
[56] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: 2015.
[57] Jiang B, Zhang L, Lu H, et al. Saliency detection via absorbing markov chain. In: Proceedings of the IEEE international conference on computer vision, 2013: 1665 $\sim$ 1672.
[58] Kemelmacher-Shlizerman I, Seitz S M, Miller D, et al. The megaface benchmark: 1 million faces for recognition at scale. In: IEEE conf Computer Vision and Pattern Recognition. 2016: 4873 $\sim$ 4882.
[59] Kim J, Pavlovic V. A shape-based approach for salient object detection using deep learning. In: European Conference on Computer Vision, 2016: 455 $\sim$ 470.
[60] Kingma D P, Welling M. Auto-encoding variational bayes. arXiv preprint arXiv: 1312.6114, 2013.
[61] Koch C, Ullman S. Shifts in selective visual attention: towards the underlying neural circuitry. In: Matters of intelligence. Springer, 1987: 115 $\sim$ 141.
[62] Krizhevsky A, Sutskever I and Hinton G E. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, 2012: 1097 $\sim$ 1105.
[63] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images, 2009.

[64] Kuen J, Wang Z and Wang G. Recurrent attentional networks for saliency detection. In: Proceedings of the IEEE Conference on computer Vision and Pattern Recognition, 2016: 3668 ∼ 3677.
[65] LeCun Y, Cortes C and Burges C. MNIST handwritten digit database. AT&T Labs, 2010, 2.
[66] LeCun Y, Denker J S and Solla S A. Optimal brain damage. In: Advances in neural information processing systems, 1990: 598 ∼ 605.
[67] LeCun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989, 1(4): 541 ∼ 551.
[68] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. 1998, 86(11): 2278 ∼ 2324.
[69] Lee C.-Y, Xie S, Gallagher P, et al. Deeply-supervised nets. In: Artificial intelligence and statistics, 2015: 562 ∼ 570.
[70] Lee G, Tai Y.-W and Kim J. Deep saliency with encoded low level distance map and high level features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 660 ∼ 668.
[71] Levin E, Fleisher M. Accelerated learning in layered neural networks. Complex systems, 1988, 2: 625 ∼ 640.
[72] Li B, Sun Z and Guo Y. Supervae: Superpixelwise variational autoencoder for salient object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2019: 8569 ∼ 8576.
[73] Li G, Yu Y. Visual Saliency Based on Multiscale Deep Features. In: CVPR, 2015: 5455 ∼ 5463.
[74] Li G, Yu Y. Deep contrast learning for salient object detection. In: CVPR, 2016: 478 ∼ 487.
[75] Li G, Yu Y. Visual saliency based on multiscale deep features. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015: 5455 ∼ 5463.
[76] Li G, Xie Y, Lin L, et al. Instance-level salient object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017: 2386 ∼ 2395.
[77] Li H, Kadav A, Durdanovic I, et al. Pruning Filters for Efficient ConvNets. In: 2017.
[78] Li Y, Hou X, Koch C, et al. The secrets of salient object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014: 280 ∼ 287.
[79] Liu B, Deng W, Zhong Y, et al. Fair Loss: Margin-Aware Reinforcement Learning for Deep Face Recognition. In: Proceedings of the IEEE International Conference on Computer Vision, 2019: 10052 ∼ 10061.
[80] Liu C, Zoph B, Neumann M, et al. Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision (ECCV), 2018: 19 ∼ 34.

[81] Liu J, Deng Y, Bai T, et al. Targeting ultimate accuracy: Face recognition via deep embedding. arXiv preprint arXiv:1506.07310, 2015.
[82] Liu N, Han J. Dhsnet: Deep hierarchical saliency network for salient object detection. In: CVPR, 2016: 678 $\sim$ 686.
[83] Liu T, Yuan Z, Sun J, et al. Learning to detect a salient object. IEEE Transactions on Pattern analysis and machine intelligence, 2010, 33(2): 353 $\sim$ 367.
[84] Liu T, Yuan Z, Sun J, et al. Learning to detect a salient object. IEEE PAMI, 2011, 33(2): 353 $\sim$ 367.
[85] Liu W, Wen Y, Yu Z, et al. Large-margin softmax loss for convolutional neural networks. In: ICML, 2016: 7.
[86] Liu W, Wen Y, Yu Z, et al. Sphereface: Deep hypersphere embedding for face recognition. In: IEEE conf Computer Vision and Pattern Recognition. 2017.
[87] Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming. In: IEEE International Conference on Computer Vision. 2017: 2736 $\sim$ 2744.
[88] Liu Z, Sun M, Zhou T, et al. Rethinking the value of network pruning. In: 2019.
[89] Long J, Shelhamer E and Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015: 3431 $\sim$ 3440.
[90] Loshchilov I, Hutter F. Sgdr: Stochastic gradient descent with warm restarts. In: ICLR, 2017.
[91] Louizos C, Welling M and Kingma D P. Learning Sparse Neural Networks through L_0 Regularization. arXiv preprint arXiv: 1712.01312, 2017.
[92] Luo J.-H, Wu J and Lin W. Thinet: A filter level pruning method for deep neural network compression. In: IEEE International Conference on Computer Vision. 2017: 5058 $\sim$ 5066.
[93] Maas A L, Hannun A Y and Ng A Y. Rectifier nonlinearities improve neural network acoustic models. In: 2013.
[94] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, 2013: 3111 $\sim$ 3119.
[95] Molchanov P, Mallya A, Tyree S, et al. Importance Estimation for Neural Network Pruning. In: IEEE conf Computer Vision and Pattern Recognition. 2019: 11264 $\sim$ 11272.
[96] Movahedi V, Elder J H. Design and perceptual validation of performance measures for salient object segmentation. In: CVPR-workshop, 2010: 49 $\sim$ 56.
[97] Müller R, Kornblith S and Hinton G E. When does label smoothing help? In: Advances in Neural Information Processing Systems, 2019: 4696 $\sim$ 4705.
[98] Nair V, Hinton G E. Rectified linear units improve restricted boltzmann machines. In: 2010: 807 $\sim$ 814.
[99] Netzer Y, Wang T, Coates A, et al. Reading Digits in Natural Images with Unsupervised Feature Learning. In: NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

[100] Ng H.-W, Winkler S. A data-driven approach to cleaning large face datasets. In: IEEE Conf Image Process (ICIP), 2014: 343 $\sim$ 347.
[101] Panis G, Lanitis A, Tsapatsoulis N, et al. Overview of research on facial ageing using the FG-NET ageing database. Iet Biometrics, 2016, 5(2): 37 $\sim$ 46.
[102] Parkhi O M, Vedaldi A and Zisserman A. Deep face recognition. 2015.
[103] Pham H, Guan M Y, Zoph B, et al. Efficient neural architecture search via parameter sharing. In: 2018.
[104] Qu L, He S, Zhang J, et al. RGBD salient object detection via deep fusion. IEEE Transactions on Image Processing, 2017, 26(5): 2274 $\sim$ 2285.
[105] Rahman M A, Wang Y. Optimizing intersection-over-union in deep neural networks for image segmentation. In: International symposium on visual computing, 2016: 234 $\sim$ 244.
[106] Ranjan R, Castillo C D and Chellappa R. L2-constrained softmax loss for discriminative face verification. arXiv preprint arXiv:1703.09507, 2017.
[107] Rastegari M, Ordonez V, Redmon J, et al. Xnor-net: Imagenet classification using binary convolutional neural networks. In: European conference on computer vision, 2016: 525 $\sim$ 542.
[108] Ren J, Gong X, Yu L, et al. Exploiting global priors for RGB-D saliency detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015: 25 $\sim$ 32.
[109] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, 2015: 91 $\sim$ 99.
[110] Rezatofighi H, Tsoi N, Gwak J, et al. Generalized intersection over union: A metric and a loss for bounding box regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019: 658 $\sim$ 666.
[111] Rumelhart D E, Hinton G E and Williams R J. Learning representations by back-propagating errors. nature, 1986, 323(6088): 533 $\sim$ 536.
[112] Russakovsky O, Deng J, Su H, et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 2015, 115(3): 211 $\sim$ 252.
[113] Sankaranarayanan S, Alavi A, Castillo C D, et al. Triplet probabilistic embedding for face verification and clustering. In: 2016 IEEE 8th international conference on biometrics theory, applications and systems (BTAS), 2016: 1 $\sim$ 8.
[114] Schroff F, Kalenichenko D and Philbin J. Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015: 815 $\sim$ 823.
[115] Shen W, Zhao K, Jiang Y, et al. Object Skeleton Extraction in Natural Images by Fusing Scale-associated Deep Side Outputs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2016: 222 $\sim$ 230.
[116] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: ICLR, 2015.

[117] Sindhwani V, Sainath T and Kumar S. Structured transforms for small-footprint deep learning. In: Advances in Neural Information Processing Systems, 2015: 3088 ∼ 3096.

[118] Smirnov E, Melnikov A, Oleinik A, et al. Hard example mining with auxiliary embeddings. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018: 37 ∼ 46.

[119] Sohn K. Improved deep metric learning with multi-class n-pair loss objective. In: Advances in neural information processing systems, 2016: 1857 ∼ 1865.

[120] Srinivas S, Babu R V. Data-free parameter pruning for deep neural networks. arXiv preprint arXiv:1507.06149, 2015.

[121] Steiner B, DeVito Z, Chintala S, et al. PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. 2019.

[122] Suau X, Zappella L, Palakkode V, et al. Principal filter analysis for guided network compression. arXiv preprint arXiv:1807.10585, 2018, 2.

[123] Sun Y, Wang X and Tang X. Deeply learned face representations are sparse, selective, and robust. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015: 2892 ∼ 2900.

[124] Sun Y, Chen Y, Wang X, et al. Deep learning face representation by joint identification-verification. In: Advances in neural information processing systems, 2014: 1988 ∼ 1996.

[125] Sun Y, Liang D, Wang X, et al. Deepid3: Face recognition with very deep neural networks. arXiv preprint arXiv:1502.00873, 2015.

[126] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015: 1 ∼ 9.

[127] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016: 2818 ∼ 2826.

[128] Taigman Y, Yang M, Ranzato M, et al. Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014: 1701 ∼ 1708.

[129] Tan M, Chen B, Pang R, et al. Mnasnet: Platform-aware neural architecture search for mobile. In: IEEE conf Computer Vision and Pattern Recognition. 2019: 2820 ∼ 2828.

[130] Treisman A M, Gelade G. A feature-integration theory of attention. Cognitive psychology, 1980, 12(1): 97 ∼ 136.

[131] Wang F, Xiang X, Cheng J, et al. NormFace: L2 Hypersphere Embedding for Face Verification. In: Proceedings of the 25th ACM international conference on Multimedia, 2017.

[132] Wang H, Wang Y, Zhou Z, et al. Cosface: Large margin cosine loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 5265 ∼ 5274.

[133] Wang L, Lu H, Ruan X, et al. Deep networks for saliency detection via local estimation and global search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 3183 ∼ 3192.
[134] Wang L, Wang L, Lu H, et al. Saliency detection with recurrent fully convolutional networks. In: ECCV, 2016: 825 ∼ 841.
[135] Wang T, Borji A, Zhang L, et al. A stagewise refinement model for detecting salient objects in images. In: Proceedings of the IEEE International Conference on Computer Vision, 2017: 4019 ∼ 4028.
[136] Wen Y, Zhang K, Li Z, et al. A discriminative feature learning approach for deep face recognition. In: European Conference on Computer Vision. 2016: 499 ∼ 515.
[137] Wolf L, Hassner T and Maoz I. Face recognition in unconstrained videos with matched background similarity. In: IEEE conf Computer Vision and Pattern Recognition. 2011: 529 ∼ 534.
[138] Wu B, Dai X, Zhang P, et al. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: IEEE conf Computer Vision and Pattern Recognition. 2019: 10734 ∼ 10742.
[139] Wu C.-Y, Manmatha R, Smola A J, et al. Sampling matters in deep embedding learning. In: Proceedings of the IEEE International Conference on Computer Vision, 2017: 2840 ∼ 2848.
[140] Wu J, Leng C, Wang Y, et al. Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 4820 ∼ 4828.
[141] Xie S, Tu Z. Holistically-nested edge detection. In: ICCV, 2015: 1395 ∼ 1403.
[142] Yan Q, Xu L, Shi J, et al. Hierarchical saliency detection. In: CVPR, 2013: 1155 ∼ 1162.
[143] Yang C, Zhang L, Lu H, et al. Saliency detection via graph-based manifold ranking. In: CVPR, 2013: 3166 ∼ 3173.
[144] Ye J, Lu X, Lin Z, et al. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In: 2018.
[145] Yi D, Lei Z, Liao S, et al. Learning Face Representation from Scratch. arXiv preprint arXiv:1411.7923, 2014.
[146] Yi D, Lei Z, Liao S, et al. Learning face representation from scratch. arXiv preprint arXiv:1411.7923, 2014.
[147] Zeng Y, Zhang P, Zhang J, et al. Towards High-Resolution Salient Object Detection. In: Proceedings of the IEEE International Conference on Computer Vision, 2019: 7234 ∼ 7243.
[148] Zeng Y, Zhuge Y, Lu H, et al. Multi-source weak supervision for saliency detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019: 6074 ∼ 6083.
[149] Zhai Y, Shah M. Visual attention detection in video sequences using spatiotemporal cues. In: Proceedings of the 14th ACM international conference on Multimedia, 2006: 815 ∼ 824.

[150] Zhang J, Sclaroff S, Lin Z, et al. Unconstrained salient object detection via proposal subset optimization. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016: 5733 ∼ 5742.

[151] Zhang J, Zhang T, Dai Y, et al. Deep unsupervised saliency detection: A multiple noisy labeling perspective. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 9029 ∼ 9038.

[152] Zhang K, Zhang Z, Li Z, et al. Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 2016, 23(10): 1499 ∼ 1503.

[153] Zhang P, Wang D, Lu H, et al. Amulet: Aggregating Multi-level Convolutional Features for Salient Object Detection. ICCV, 2017.

[154] Zhang X, Zou J, He K, et al. Accelerating very deep convolutional networks for classification and detection. IEEE transactions on pattern analysis and machine intelligence, 2015, 38(10): 1943 ∼ 1955.

[155] Zhao J, liu J, Fan D, et al. EGNet:Edge Guidance Network for Salient Object Detection. In: ICCV, Oct. 2019.

[156] Zhao K, Xu J and Cheng M.-M. RegularFace: Deep Face Recognition via Exclusive Regularization. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[157] Zhao K, Shen W, Gao S, et al. Hi-Fi: Hierarchical Feature Integration for Skeleton Detection. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, July 2018: 1191 ∼ 1197. http://kaizhao.net/hifi.

[158] Zhao R, Ouyang W, Li H, et al. Saliency detection by multi-context deep learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1265 ∼ 1274.

[159] Zhou Y, Zhang Y, Wang Y, et al. Accelerate CNN via Recursive Bayesian Pruning. In: IEEE International Conference on Computer Vision. 2019.

[160] Zhuge Y, Zeng Y and Lu H. Deep embedding features for salient object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2019: 9340 ∼ 9347.

[161] Zoph B, Le Q V. Neural architecture search with reinforcement learning. In: 2017.