

000
001 **Unsupervised Deep Domain Adaptation on**
002 **People Detection**

003
004 Anonymous ECCV submission
005

006 Paper ID ***
007
008

009 **Abstract.** 一方面，本课题将利用卷积神经网络+ 递归神经网络的方法
010 实现人头检测的算法，并与利用卷积神经网络+前馈神经网络实现的检测
011 算法相比较。另一方面，这篇论文提出了基于无监督场景迁移下的在密集
012 人群中人头检测的问题。也就是给定一个在原场景充分训练好的深度神经
013 网络模型，如何在待迁移场景没有任何标记数据的情况下，训练一个能够
014 识别待迁移场景中人物的模型。首先，我们利用迭代算法来自动标记待迁
015 移场景中的人物。由于自动标记出来的人物数据中存在错误数据，假阳性的
016 数据比例会随着训练迅速增加，影响到整个网络。因此我们将原场景中的
017 标记数据加入到训练数据中，来抑制错误数据。同时，我们将神经网络
018 最后的全连通层等价转化为两个子层。在新添加的子层上加入非监督的正
019 则化函数来防止训练过程中的过拟合。跟将原场景中训练的原模型在没有
020 利用待迁移场景的数据来微调得到的结果相比，我们提出的方法将待迁
021 移场景中人物检测的召回率提高了xx,同时precision只从xx掉到xx。与此
022 同时，我们将我们的算法运用到了标准的迁移学习基准数据库Office数据库，
023 我们的方法在监督场景迁移喝非监督场景迁移都取得了最好的结果。

024 **Keywords:** 无监督场景迁移，深度卷积神经网络，人物检测
025
026
027
028

029 **1 Introduction**
030

031 深度卷积神经网络在计算机视觉的任务上有非常突出的表现，例如语音识别
032 (speech recognition)，为视频配字 (video captioning)。本课题将利用卷积
033 神经网络+ 递归神经网络的方法实现人头检测的算法，并与利用卷积神经网
034 络+前馈神经网络实现的检测算法相比较。对于经典的物体检测算法，它们首
035 先得到所有可能的物体区域，然后对这些物体区域一一进行检测，并做出判
036 断。然后这样的方法有两个问题，一个在于提取所有的可能物体区域并一一检
037 测，这在上时间上花费巨大，对于监控场景来说，更是很难做到实时的检测。
038 另一方面这个检测算法也严重依赖与提前计算的所有可能的物体区域。这篇
039 文章采用的是端到端的监测网络。端到端就是将一张图片作为输入，网络直接
040 输出将所有可能的物体区域。同时我们利用了递归神经网络进行序列预测的功
041 能，可能很大程度的避免对同一个物体的重复预测。

042 另一方面，深度卷积神经网络同时也存在人工标记这个问题。大部分的任务
043 都需要大量的标记数据。像著名的PASCAL VOC和MS COCO，需要上千万的
044 带标记的图片数据来完成训练的过程。在监控的运用中，比如人物检测，这个

人工标记的过程就更加繁重了，因为需要标记到具体的位置。时至今日，世界上又有着无数的摄像头来担任监控的功能。然而，由于监控场景变化巨大，它受到背景，光线，视角，清晰度等等一系列的不稳定因素的影响，这就使得对于所有的监控场景进行人工标记这个工作更加的不可实施。在人物检测的任务中，需要上万的数据来训练一个好的检测神经网络。

当待迁移场景标记数据非常的少，甚至没有的情况下，场景迁移的任务就是帮忙减少所需要的有标记的数据的数量，当我们在原场景有大量的标记数据，那么这个迁移就需要考虑到原场景和待迁移场景之间的联系。大部分传统的场景迁移的算法[1-5]都尝试找到原场景和待迁移场景之间的共同特征，或者将特征投射到高维空间中和子空间中。最近，也有其他工作[6-8]提出了通过深度网络来学习基于特殊场景的检测工具。在我们的人物检测的任务中，我们尝试着将在原场景中充分训练的深度网络迁移到一个待迁移场景，这个待迁移场景并没有任何标记数据。

在这篇文章中，我们提出了基于人物检测的无监督场景迁移的新算法。使用原场景下的原模型进行初始化，我们首先利用迭代算法来自动标记待迁移场景的图片作为伪真实数据。在每一次迭代中，待迁移场景的模型被更新并用来自动标记下一个迭代需要的训练数据。然后，由于缺乏真阴性数据和伪阳性数据，自动标记数据中的错误将导致伪阳性的比例迅速上升，同时进一步影响到了召回率的提高。为了消除数据噪声的影响，我们提出了两个方法来正则化深度网络的训练。一方面，我们将来自原场景的有标记数据的图片加入了自动标记的数据中，来弥补真阴性数据的不足。另一方面，我们将深度网络中最后的全连通层的操作切分为两个子运算-元素点乘和求和。最后的全连通层也因此可以转化为两个子层，元素点乘层和求和层。基于此，一个无监督的正则化函数可以被添加到深度网络中作为无监督的损失函数来防止伪阳性结果的迅速增加以及增强召回率。

在人物检测的任务之外，我们也添加了基于标准迁移学习基准数据库的实验来评估我们的算法的适用性。我们的算法的迁移结果同时在监督和无监督的设置下超过了之前发布的其他人的工作，这证明了我们的迁移算法有足够的灵活度来处理检测和分类的任务。

这篇文章的贡献有这样三个方面：

- 我们提出了一个可行的解决方案来将在原场景充分训练的深度模型迁移到待迁移场景，而待迁移场景并不需要任何标记数据。这使得深度网络在不同监控情况下的部署变得更为容易。
- 由于大部分的算法都关注于最后一层的特征向量，我们往后一步来研究全连通层，并将最后一层全连通层转化为元素点乘层和求和层。因此一个无监督正则化函数可以加到元素点乘层，这样可以在抑制伪阳性和增加召回率方面取得更好的效果。
- 我们在标准迁移学习基准数据库的实验也表明了我们在其他深度场景迁移的任务中也可以取得好的结果。

这篇文章剩下的部分是这样组织的。首先章节2回顾了相关的工作，章节5给出了算法的细节部分，实验部分的结果在章节6 中做出介绍。章节7 总结了这篇论文。

090 2 Relate Work

091 检测多个互相遮挡的人物在计算机视觉领域是一个很棘手的问题。早期的工作
092 [9, 10]通过局部特征、霍夫曼投票等方法，但是仍然需要复杂的调试和多阶段的
093 算法，而且这些工作中使用的局部特征更是被现在的深度特征所取代。

094 在深度特征取代了传统的人工设计的特征之后，有些工作[11–13]便尝试利用
095 滑窗的方式或者检测所有可能物体物体区域的方式来获得可能的区域，然后通
096 过一个分类的卷积神经网络来对区域进行一一分类。这些方法对于没有遮挡的
097 少数物体的检测是有效的，但是在有遮挡的情况下就很难有好的表现。

098 另一方面，在许多检测的工作中，在原场景中的大量数据的训练下得到的模
099 型直接拿来检测待迁移场景的图片。他们假设待迁移场景的图片数据是原场景
100 中图片数据的子集。然而当待迁移场景的数据分布跟原场景的数据分布差别很大时，
101 原模型的表现会下降很多。场景迁移致力于得到更好的模型。

102 许多场景迁移的工作致力于找到原场景和待迁移场景之间的共同特征。Saenko
103 et al. [1, 2]提出了一个基于线性变换的算法和一个基础核变化的算法来减少场景
104 之间的变化。Gopalan et al. [3]将特征向量映射到Grassmann manifold而不是直
105 接在元数据的特征上进行操作。或者，Mesnil et al. [14]使用迁移学习的算法来
106 获得好的数据特征表示。然而这些方法都相当的局限，因为他们并没有学习基于
107 特殊场景的特征来增加准确率。我们的正则化函数受到了这些工作的启发。

108 另外一组场景迁移的工作[4, 5, 15]是将原场景的数据分布跟待迁移场景的数
109 据分布弄得尽量相似。在这些工作中，Maximum Mean Discrepancy (MMD)被
110 用到作为重新从原场景中挑选数据的评估依据，来使得其于待迁移场景的数据
111 分布尽量相似。在工作[16]中，MMD作为正则化工具加入到模型中以减少数据
112 分布之间的差别。

113 同时，在学习基于特殊场景学习的检测算法也有一些工作。Wang et al.[6,
114 7]利用了上下文的信息来计算检测物体的置信度，同时他也提出了学习待迁移
115 场景数据分布以及为特定场景的视觉模式设计了聚类层的算法。这些工作依赖于
116 重新给自动标记的数据以计算权重并加入最后的损失函数以及需要额外的上
117 下文设计以获得可靠的结果。同时，Hattori et al. [8]通过生成空间变化的行人
118 模型来学习特定场景的检测模型。Pishchulin et al. [17]利用了一个3D的形状模
119 型来生成训练数据。然而为了场景迁移来进行合成工作也很耗费人力。

122 3 检测网络的构建

123 监控场景在现代生活的应用已经越来越广泛，而对监控场景中的人物进行检
124 测也有着重要的应用价值。监控视频中的人物头部检测大致可以分为两类，一
125 类借助于动态的视频，先提取出固定的背景图片，然后减去背景进行检测；另一
126 类直接从静态的一帧帧图片中，检测出人物头部的区域。然而检测速度慢，
127 误检率高，部署成本高仍然是目前面临的难题。卷积神经网络在计算机视觉领
128 域的巨大成功使得这个任务变得更加可行。我们从静态的图片入手，运用卷积
129 神经网络（Convolutional Neural Network）进行图片检测。卷积神经网络作为
130 编码器将关于图片的特征并表达为特征图时，就需要再加一个解码器将特征图
131 表达为最后的输出。关于将特征图解码有两个可行的方向，一个是直接利用前
132 一个解码器将特征图解码为最后的输出，另一个是先将特征图解码为中间的
133 图像，然后再将中间的图像解码为最后的输出。关于将特征图解码有两个可行的
134 方向，一个是直接利用前一个解码器将特征图解码为最后的输出，另一个是先将
135 特征图解码为中间的图像，然后再将中间的图像解码为最后的输出。

135 馈神经网络将特征图转化为特征向量并最后表达为输出，另一个就是利用递归
136 神经网络将特征图的信息解码为一系列的输出。在物体检测的任务中，卷积神
137 经网络+前馈神经网络的方法被广泛采用，然后我们的任务为监控场景下的人
138 头检测，这有别于普通的物体检测。首先监控场景下存在这密集的场景，在某
139 一个时刻可能存在着多达几十个人物出现在场景中；另一个方面，监控场景下
140 的人物相比于普通相机拍到的人物都要更小。这种从物体数量到物体尺寸都跟
141 经典的物体检测的任务有所出处，因此卷积神经网络+前馈神经网络的方法会
142 遇到一些问题。卷积神经网络+递归神经网络的方法可以解决存在多个物体需
143 要检测的情况。

144 3.1 卷积神经网络+前馈神经网络的方法

145 卷积神经网络+前馈神经网络的方法被广泛的应用到物体检测当中。在卷积
146 神经网络+前馈神经网络的方法中，卷积神经网络将输入图片编码为高维的
147 特征图，然后前馈神经网络直接将特征图直接解码为最后的输入。在这众多的算
148 法中，我们选择了经典的Fast R-CNN[18]算法作为此类检测算法的代表，并利
149 用其算法在我们的数据上做训练。然而由于传统物体检测任务的物体大小跟本
150 文章的任务中的需要检测的任务的大小存在较大差异，因此并不一定可以取得
151 好的效果。

152 选择性搜索

153 Fast R-CNN同R-CNN一样，都需要提前提取所有可能的预测图片区域，在
154 实验中，我们选择了Fast R-CNN推荐的选择性搜索算法来提取每张图片的可以
155 预测区域。对于一张图片，选择性搜索需要提取一千甚至两千个可能的预测区
156 域，才能做到尽量覆盖所有的可能的物体。这对于Fast R-CNN原始的数据集来
157 说是足够的，因为Fast R-CNN所训练和测试的数据集中的物体都是比较明显且
158 在图片中面积占比比较大。然而在我们的任务中，人物的头部小而密集，那么
159 选择性搜索很难有效的所有的可能人头区域都提取出来，而且如果改变选择性
160 搜索的参数使得尽量给出更多的可能区域，那么大量的可能区域也给训练和测
161 试时间带来了极大的麻烦。在我们的实验中，选择性搜索需要对每张图片给出
162 近三千个可能的区域，以尽量覆盖所有的人头区域。同时，选择性搜索本身
163 的算法时间也不少，对于一张图片来说标准的i7 处理器需要近1s的时间才能提取
164 完所有的可能区域。

165 算法框架

166 对于Fast R-CNN的训练过程，首先算法利用选择性搜索算法提取出图片中
167 所有可能的预测区域。然后对输入图片进行图片金字塔的操作，以减少物体大
168 小不同对结果产生的影响，当然在我们的应用中人物头部的大小超出了图片金
169 字塔的作用。对于得到的图片金字塔，通过卷积神经网络编码为最后的包含了
170 丰富的图片信息的特征图。这个时候对于之前选择性搜索得到的每一个可能的
171 预测区域，从特征图中裁剪出相应的子特征图。子特征图再利用pooling层来统
172 一到相同的大小。最后连上两个不同的全连通层，一个全连通层直接输出这个
173 区域中物体的种类，另一个全连通层进行物体位置的细致化回归。最后的损失
174 函数直接跟物体的类别和对应的更加的细致的区域坐标相关联。

在Fast R-CNN的测试过程中，同样需要先提取所有可能的预测区域、图片金字塔、提取特征图、pooling、全连通层输出等步骤。不同的是：在训练过程中，由于选择性搜索提取出的非人物区域占大多数，而真正能够匹配上真是物体区域的占少数，因此不能无差别的选择需要训练的数据，那么会导致训练数据的失衡，因此在每次训练的过程中，需要有25%的预测区域与真实的物体区域有超过0.5的重合度。但是在测试的过程中，没法预知真实物体的区域，因此选择性搜索计算出的所有可能物体区域都需要进行测试以得到最后的输出。

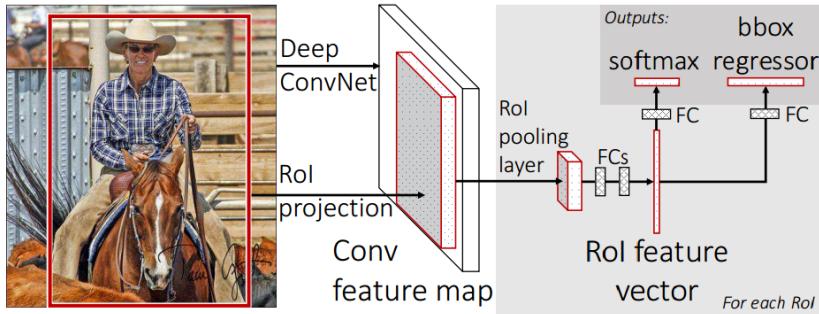


Fig. 1. Fast R-CNN框架图。一张输入图片和多个可能的区域被输入到网络中，每一个可能的区域都通过pooling层得到一个固定大小的特征图，然后将特征图连接到全连通层来输出物体的类别和细致化的区域坐标

存在的问题

我们在密集场景人物检测的数据集上测试了该算法，然后并不能得到很好的效果，我们利用超过一万张图片对Fast R-CNN进行训练，在收敛之后，网络的召回率和准确率都不高，其表现甚至比不上直接通过滑动窗口加上分类的卷积神经网的效果，示意图如图x 所示。经过分析，我们认为有以下几个原因：

- 首先，通过选择性搜索的算法计算所有可能的物体区域在我们的场景中的表现比较有效。我们的场景为监控视频中的密集场景，存在大量的人物互相遮挡的情况，同时人物的头部区域也很小，选择性搜索很难有效的得到所有的区域，就算框到了，也存在区域大小的问题。
- Fast R-CNN利用图片金字塔来减少图片大小不同引起的错误问题，然而对于我们的场景来说，这种人工引入的方法并无法从根本上解决这个问题。
- 在卷积神经网络之后的特征图，Fast R-CNN利用pooling层来裁剪出每个区域对应的子特征图。然而对于我们的任务来说，一个小的人头对应的子特征图的宽度和高度都为一，在这个基础上把其统一到 $x \times x$ 的大小的特征图显然没有意义。
- 最后，虽然我们想要让神经网络直接去预测人物的头部，然后人物的身体部分对于预测来说也是至关重要的信息，否则单单依靠人头在有些情况下是很难跟背景分离开的，特别是在昏暗和像素低的情况下，就算是人也很难只依靠人物头部直接定位到这个任务。而Fast R-CNN直接裁剪了人物头部对应

225 的特征图，那么人物身体部分的重要信息无疑会排除掉，那么这样训练过来
 226 的网络没有考虑到身体的信息，那么能不能完成这个识别任务也是一个问
 227 题。

228 在这种卷积神经网络+前馈神经网络的检测算法没有办法取得好的效果的情
 229 下，我们将注意力转移到卷积神经网络+递归神经网络的方法上。卷积神经网
 230 络+前馈神经网络的检测算法的不佳表现的根本原因在于训练数据物体大小的
 231 差异太大，在Fast R-CNN得到的特征图中，每一个 1×1 的子特征图都包含了多
 232 个人头，因此卷积神经网络+递归神经网络的方法对每一个 1×1 的子特征图进行
 233 序列的人物预测可以取得更好的效果，具体见下一节中的内容。



236
 237 Fig. 2. Fast R-CNN的实验结果图。从图中可以看出算法的召回率和准确率都不高。
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249

250 3.2 卷积神经网络+递归神经网络的方法

251 递归神经网络

252 在前馈神经网络中，没有办法有效果的对序列的信息进行编码和预测，导致
 253 了其在语音处理等序列预测领域有很大的限制。而递归神经网络的出现很好的
 254 解决了这个问题。若图所示便是递归神经网络的示意图。在递归神经网络中，
 255 输入为可以为一个序列，输出也可以为一个序列。递归神经网络在文字预测方
 256 面有着很好的效果，举一个单词预测的例子，当我们输入一个单词的前几个字
 257 母，那么如果给出最可能的下一次字母呢？假设我们的词汇量中只有4个可能
 258 的字母“he^{llo}”，然后我们想要训练一个递归神经网络能够预测“hello”。这个训
 259 练的过程其实就是4个分开的步骤：首先当给定输入为“h”时，那么字母“e”
 260 输出的概率应该是最高的，其次给定“he”，那么字母“l”输出的概率应该
 261 是最高的，然后给定“hel”，那么字母“l”输出的概率应该是最高的，最后给定
 262 “hell”，那么字母“o”输出的概率应该是最高的。需要注意的是，当我们第
 263 一次输入“l”时，给出的预测是“l”，当我们第二次输入“l”时，这个时候
 264 给出的输出为“o”。由此可以看出递归神经网络并不是只依赖于当前的输入，
 265
 266
 267
 268
 269

而是必须一直保留着前文的信息才能够做到这个预测，而前馈神经网络是没有办法做到的。在检测阶段，我们将一个字母输入到递归神经网络，然后得到一个关于输出的概率分布。然后我们根据概率从这个分布中采样得到一个可能的输出字母并反馈作为接下来的输入。重复这个过程来得到一系列的字母直到终止。图x给出了递归神经网络的示意图。

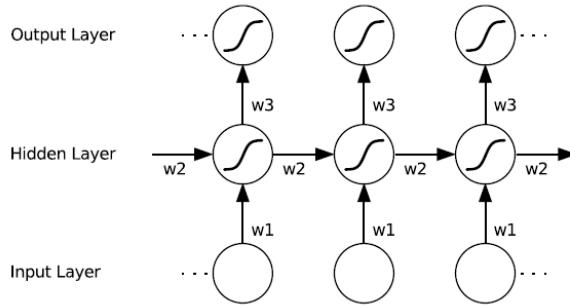


Fig. 3. 递归神经网络

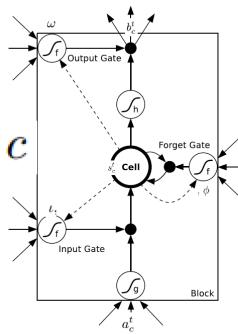
递归神经网络的前传跟普通的前馈神经网络差不多，除了当信息到达隐含层时，需要同时把当前的输入和前一个时间点的隐含层的输出一起作为输入。为了保持结构上的一致，递归神经网络的第一个时间点可以生成一个无效的前一个时间点的隐含层作为输入。

递归神经网络的问题存在于于长序列进行预测。如果我们只需要根据最近的信息来完成现在的预测，比如上文提到的预测字母的例子，当我们尝试预测“hello worl”的“d”字母时，我们并不需要久之前的信息，而只需要“worl”这几个字母即可，那么我们知道接下来很可能是“d”字母。在这个例子中当前预测所需要的上文信息离当前很近，因此并不存在太大的问题，递归神经网络也可以完成这个任务。然而有些情况下我们需要更多的上文信息，比如我们想要根据“the first program we learned is to print hello w”来预测这边应该是输出“world”。从前文“hello”的信息来看，有可能是接一个含“w”的名词，当我们追溯到更远的“the first program”那么我们知道这很可能是经典的“hello world”，但是对于递归神经网络来说，“the fist program”这个信息来影响当前的预测是比较难得。如果考虑到需要依据的信息和当前的预测更远时，那么递归神经网络几乎无法完成这个任务。有不同的方法尝试着解决这个问题，包括改变误差反传的机制，或者将信息进一步压缩。在这些方法中下过最好的就是下一节讲到的LSTM单元。

LSTM

在上一个章节中，我们提到了使用递归神经网络所带来的对于序列预测的帮助，能够将前文的信息也编码到隐含层中，以作为下一个预测的依据。然后标准的递归神经网络框架也面临了极大的问题。这个问题来源于神经网络前传和误差反传的机制，当递归神经网络的所要预测的序列太长时，前文的信息对于

315 后面的预测的影响将是指数级的上升或者下降。解决这个问题最有效的办法就
 316 是利用LSTM来取代传统的神经元。图x给出了LSTM的示意图。
 317
 318



330 Fig. 4. lstm
 331
 332

333 一个标准的LSTM网络跟标准的递归神经网络是一样的，除了其中隐含层的
 334 神经元都被替换为LSTM单元。一个基本的LSTM包含了一小个集合的递归子
 335 网络，称为记忆单元。对于每一个记忆单元，都包含了几个自连通的记忆细胞
 336 和三个门，即输入输出和忘记门。这三门也分别对应了对记忆读、写和重置的
 337 功能。这三个门是LSTM的核心所在。它们可以在时间推移的过程中，储存或
 338 者读取这些记忆，甚至抹除无用的记忆，这样就减轻了随着时间的推移信息难
 339 以传递的问题。例如我们通过控制某个LSTM输入门关闭，这样该LSTM中的
 340 信息就不会受到新输入的影响，在将来的某一个时刻，保存下来的信息便有可
 341 能通过开启输出门使得信息被读取并发挥作用。当忘记门开启式，LSTM便可
 342 以忘掉无用的信息，在将来的某一个时刻，通过打开输入门来保存新的记忆信
 343 息。在这样一个过程中，LSTM很好的模拟了人脑记忆的思维模式。如果用某
 344 一个时刻的隐含层比作人脑的话，那么隐含层中的LSTM单元就对应了人脑的
 345 记忆细胞，LSTM单元某一个时刻的值也对应了人脑在某一个时刻的记忆。对于
 346 人脑来说，有时候发生了可能影响到未来的事情，比如你在地铁上想到了一
 347 个新算法而手头有没有电脑可以实验，那么你就利用自己的记忆在这个新算法
 348 记忆下来并在将来有电脑可以使用的情况下触发这个记忆并做相关实验，在实
 349 验完成之后发现并没有起效果，于是你就忘了这个算法。对应于LSTM网络，
 350 那么当新算法作为输入输入到网络中，LSTM单元及时开启输入门并保存算法
 351 信息，当后来的某一个时刻有电脑可以使用时，触发了输出门的开启，于是算
 352 法被输出到当前时刻，失败的实验结果也反馈给了LSTM网络，于是它开启了
 353 忘记门将这个算法忘记。因此，LSTM很好的模拟了人脑的记忆读取写入和忘
 354 记的功能，因此对于长序列的信息也可以做到很好的保存和处理。

355 3.3 基于卷积神经网络+递归神经网络的方法的检测网络

356 在这个章节中我们将介绍我们使用的检测网络的结构。就像上文提到的那
 357 样，监控场景下存在这密集的场景，在某一个时刻可能存在着多达几十个人物

出现在场景中以及监控场景下的人物相比于普通相机拍到的人物都要更小。这种从物体数量到物体尺寸都跟经典的物体检测的任务有所出处，因此卷积神经网络+前馈神经网络的方法会遇到一些问题。卷积神经网络+递归神经网络的方法可以解决存在多个物体需要检测的情况。在这边我们使用了一个比较新颖的检测网络。这是一个端到端预测的算法。所谓的端到端就是我们将一张图片作为输入，那么网络输出将所有可能的物体区域记忆其置信度。这种端到端直接预测的算法适用于我们的监控场景，相比于其他先提前计算所有可能的区域，然后将这些区域一一通过神经网络来判断做分类要更加的有效果。因为我们场景中的人头存在多而小的问题。同时我们利用了递归神经网络进行序列预测的功能，可能很大程度的避免对同一个物体的重复预测。

检测模型的构建

这个部分将介绍CNN与递归神经网络的结合所得到的监测网络。将CNN与递归神经网络为基础的解码器结合在一起有如下优点：首先就是能够直接将包含了丰富的图片信息的GoogLeNet 特征向量直接表达为输出，其次就是能够对一个子区域中可能的人物头部做序列预测。这样我们的网络就能够记住之前已经预测过了的人物，避免了对同一个物体的重复预测。

我们使用的检测网络首先通过深度卷积网络将一张图片编码为一个高位特征向量，然后将这个特征向量表达为一系列的预测区域。为了做序列预测，那么我们就需要使用LSTM的记忆功能来完成这个任务。更具体的，我们将一张图片通过CNN编码为 15×20 的网格，每个网格内都包含了1024 维的GoogLeNet 高纬特征向量。同时每个网格对应了输出图片的一个子区域，这个1024 维的GoogLeNet 高纬特征向量包含了丰富高度抽象的图片信息，其中便携带了能够作出位置预测的信息。对于每个网格，子区域中间的 64×64 区域是鼓励网络取预测的，在这中心区域之外，并不鼓励网络去预测。而 64×64 的大小已经足够用于捕捉监控场景中人物头部的大小。更大的区域并不能在我们的这个任务中取得更好的效果。对于每一个网格，1024维的向量被分别输入到不同的递归神经网络网络中，之间并不互相影响，但是在最后的结果中将这300 个LSTM 网络的结果合并在一起。在递归神经网络中的每一步，LSTM输出一个新的预测区域以及其对应的置信度，这个新的区域在之前并没有被输出过人物。

这里利用到的递归神经网络网络中，每个隐含层包含250个LSTM单元，不带有bias项。在每一个递归神经网络的时间点上，我们将1024维的GoogLeNet 特征向量和上一层的隐含层输出拼接在一起作为当前的输入输入给当前的隐含层。在训练的过程中，我们通过损失函数的设计鼓励一系列的预测区域能够按照从大到小的置信度输出。当LSTM无法再找到一个置信度能够高于一个提前设定的阈值的预测区域，那么LSTM便停止预测。300个分离的LSTM网络的输出被总结在一起，作为输入图片最后的预测结果。

损失函数的设计

在上一个章节中提到，我们这个网络需要预测一个序列的区域以及相对应的置信度。在递归神经网络的每一个时间点，LSTM输出一个区域以及置信度，我们将这个区域标记为 $b = b_{pos} + b_c$ ，其中 $b_{pos} = (b_x, b_y, b_w, b_h) \in R^4$ 是预测区域的相对的位置、高度和宽度。置信度 b_c 的取值范围是 $b_c \in [0, 1]$ 。在测试阶段，当置信度的值低于提前预测的阈值比如0.9时，递归神经网络将停止预测区域。当置信度高于提前预测的阈值比如0.9时，我们将其看成一个正样例。我们

将对应的真实的人物区域标记为 $G = \{b^i | i = 1, \dots, M\}$, 而由递归神经网络预测的一系列可能区域为 $C = \{\tilde{b}^j | j = 1, \dots, N\}$ 。

接下来我们将如何利用损失函数来引导和鼓励神经网络来序列预测区域。一个典型的预测错误包含不正确的定位, 多个预测区域同时预测同一个区域, 以及不正确的置信度。不同的错误就需要不同的反馈。对于不正确的定位, 我们需要对其位置进行微调, 对于多个预测区域预测了同一个区域, 就需要保留其中一个区域, 对于不正确的预测, 我们就需要降低其置信度已达到停止预测的效果。为了能够正确引导网络训练, 就需要设计一个预测区域与真实人物区域一一匹配的匹配算法。假设存在这样一个匹配方案 $f : G \rightarrow C$, 即 C 中的第 $f(i)$ 个预测区域对应于 G 中的第 i 个真实人物区域。给定匹配翻案 $f : G \rightarrow C$, C , G , 我们将损失函数定义为如下形式:

$$L(G, C, f) = \alpha * \sum_{i=1}^{|G|} l_{pos}(b_{pos}^i, \tilde{b}_{pos}^{f(i)}) + \sum_{j=1}^{|C|} l_c(\tilde{b}_c^j, y_j) \quad (1)$$

其中 $l_{pos} = \|b_{pos}^i - \tilde{b}_{pos}^{f(i)}\|$, 是预测区域与真实人物区域之间的差值, l_c 是cross-entropy函数惩罚预测区域的置信度, α 用来权衡置信度和位置不匹配错误之间的影响。当匹配方案确定下来之后, 我们便可以通过 $L(G, C, f)$ 计算损失方程来进行误差反传。

对于匹配算法, 我们考虑最简单的基于固定顺序的匹配算法来匹配预测区域和真实人物区域。我们将真实人物区域根据坐标位置从上到下从左到右排序, 然后将预测区域按照先后顺序依序排给真实人物区域。我们将这个匹配方程称为固定顺序匹配方程。然而固定匹配算法存在着局限性, 当递归神经网络输出伪阳性或者伪阴性预测区域时, 它可能将预测区域不正确的匹配给真实人物区域。这是固定顺序匹配算法没法解决的问题。因此我们需要一个更加动态的匹配算法来解决这个问题。

考虑到我们想要通过损失函数鼓励我们的网络给出一系列的预测区域, 当输出的置信度低于一个提前预定的阈值比如0.9时, 我们停止进一步输出预测区域。为了能够让网络达到这样一个效果, 我们必须鼓励引导我们的网络在优先在序列预测中给出正确的预测区域, 将置信度低的预测区域放到置信度高的预测区域之后。因此当两个预测区域预测了同一个真实人物区域时, 我们更倾向于将序列中出现更早的区域匹配给真实人物区域。除了预测到同一个区域, 还有其他的因素影响到我们的匹配算法, 包括预测区域与真实人物区域的交集面积、预测区域是在递归神经网络的哪一个时间点给出的、预测区域与真实人物区域之间的距离。因此, 当预测区域与真实人物区域的交集面积不同时, 我们偏向于选择交集面积大的预测区域来匹配当前真实人物区域; 当交集面积相同时, 我们考虑预测区域与真实人物区域之间的距离更近的来匹配当前真实人物区域; 如果预测区域与真实人物区域之间的距离相同时, 我们考虑预测区域在递归神经网络中给出的时间点给最早的预测区域来匹配当前真实人物区域。

4 数据集与有监督学习的实验结果

4.1 数据集

同时，我们通过摄像头采集了一个特定场景的监控视频。这个监控视频是在一个大学餐厅拍摄，具体环境为餐厅的正门口，时间为中午学生集中用餐时段，采集时长为30分钟。这个场景中包含了具有各种变化的人物表现：1. 不同人物的正面侧面等各个角度的照片，2. 随着摄像头的拍摄区域由远及近人头的大小也有明显的变化，3. 这些人物之间经常会出现明显的身体遮挡，包括头部也有可能会出现遮挡现象，当处于高峰期时，会出现人群混杂的情况，难度更大，4. 人物的明暗变化也不尽相同，有广场上的比较明亮的区域也有位于屋檐下的比较暗的区域，甚至包括了上行电梯口的相当昏暗的区域，这个时候人物可能会与电梯融为一体难以分辨。同时为了为神经网络的训练提供训练数据，我们从视频中抽取了412张图片并对其进行人物标记，总共有4014个人物。其中最多的一张图片中包含了23个人头。因此这是一个相对密集的场景。我们通过annotorious工具对人物进行标记。annotorious是一个基于JavaScript的网页标注工具，我们对该进行改造使其适用于我们的项目。通过annotorious工具，我们可以获得更加精确的标记数据。上文提到的标记数据来自监控视频的前二十分钟，最后的十分钟作为检测用，并不提供标记数据用以训练。该数据集中的示例图如图x所示。

4.2 有监督学习的实验结果

首先我们将公开的Brainwash数据集用于训练我们的基于卷积神经网络+递归神经网络的方法的检测网络直到网络收敛。然后我们将412张标记过的数据随机分成三个数据集，包括训练数据集300张图片，验证数据集42张图片以及测试数据集70张图片。我们使用深度学习框架Caffe作为我们的训练平台，我们将网络的学习率设为0.01，动量设为0.5。在10000次迭代之后，我们模型收敛，其准确率为92%，召回率为82%。

同时我们也测试了通过基于卷积神经网络+前馈神经网络的方法的检测网络的方法(fast r-cnn)来训练，然而效果并不好。因此我们对比了这两个网络之间的区别：

1. 从两个算法提特征向量的角度来看，他们都应用了卷积神经网络。从这个方面来看是没有太大差异的；
2. 在得到特征向量之后，前者利用递归神经网络作序列预测，最多检测到的人物头部为5个，而后者则直接通过全连通层做二分类；
3. 两个算法能够适用的场景不一样，前者适用于面积小的物体的检测，例如监控视频中的人物头部，而后者适用于面积大的物体的检测。

从上述原因可以看出，fast r-cnn的算法在处理本课题的检测人头的任务时并不能有好的表现。

在选择了卷积神经网络和递归神经网络相结合的算法来检测人物头部之后，我们尝试了迁移学习方面的算法研究，即如何将只利用了咖啡厅的图片来训练的神经网络在不标记任何新场景图片的情况下识别新场景的人物头部。关于这个任务我们首先考虑到了一下三点：

1. 虽然旧场景和新场景的背景不同，但是需要检测的人物头部却有着相同或相似的特征；

- 495 2. 如果是利用已经标记的新场景图片进行训练，仅需要三百多张就可以取得很
496 好的效果，说明新训练的神经网络和旧神经网络之间的差异并不是非常大；
497 3. 如果将旧场景训练的神经网络直接检测新场景，那么其precision为91%， recall为118%
498 说明虽然新场景并没有训练过，但是它的precision已经很高了。
499

500 结合上述三个点，我们初步提出了适用于该任务的场景迁移的方法：1) 将新
501 场景的图片输入旧神经网络，得出检测的结果ri；2) 假定ri为ground truth，并
502 利用ri训练旧神经网络，得到新神经网络wi；3) 将新神经网络wi当成旧神经网
503 络，重复步骤1、2、3，直到达到人为设定的迭代次数位置。实现上述算法，取
504 得了precision 50%， recall 49% 的结果。基于此，我们进一步研究了表现更好的
505 场景迁移算法。

5 我们的迁移方法

508 在这个部分中，我们将介绍在人物检测的任务中的无监督场景迁移算法。我
509 们将来自原场景的训练图片标记为 $\mathbf{X}^S = \{x_i^S\}_{i=1}^{N^S}$ ，把待迁移场景中的训练图
510 片标记为 $\mathbf{X}^T = \{x_j^T\}_{j=1}^{N^T}$ 。对于原场景中的图片，我们将其对应的标记标记
511 为 $\mathbf{B}_i^S = \{b_{i,k}^S\}_{k=1}^{N_i^S}$ ，其中 $b_{i,k}^S = (x, y, w, h) \in R^4$ 。然而待迁移场景中的图片的标
512 记是自动标记的，我们标记为 $\tilde{\mathbf{B}}_j^T = \{\tilde{b}_{j,k}^T\}_{k=1}^{N_j^T}$ ，其中 $\tilde{b}_{j,k}^T = (\tilde{x}, \tilde{y}, \tilde{w}, \tilde{h}) \in R^4$ 。在
513 迁移的过程中，待迁移场景的标记随着每次迭代而变化。在待迁移场景中，人
514 工标记的数据仅仅用来做迁移算法的评估用。
515

516 我们的迁移算法框架分为两个框架流—愿框架流和待迁移框架流。图片x展示了
517 这个框架。愿框架流将原场景的数据作为输入，待迁移框架流将待迁移场景的
518 数据作为输入。这两个框架流可以利用任何的端到端的深度检测网络作为它们的
519 模型。这里我们使用了在章节5.1中介绍的网络作为两个框架流中的模型。
520 在初始化阶段，我们首先利用足够的原场景的标记数据来训练原框架流中的模
521 型，有监督的损失函数来在预测区域进行回归。在训练收敛之后，原框架流的
522 权重用来初始化待迁移框架流中的待迁移模型。在迁移阶段，迭代算法用来作
523 为训练的方法。待迁移框架流中的待迁移模型在迭代过程中被不断更新，同时
524 原框架流中的原模型权重不再更新。
525

526 我们的算法同时利用了监督的损失函数和非监督的损失函数来训练学习基于
527 特定场景的检测模型，同时避免训练中的过拟合。对于监督的损失函数，我们
528 利用自动标记的数据作为训练数据。由于自动标记的数据存在数据错误，我么
529 需要利用非监督的损失函数来正则化整个网络。我们将愿框架的原模型作为数
530 据分布的参考。因此我们在迁移过程中结合的无监督和监督的损失函数可以被
531 这样描述：

$$L(\theta^T | \mathbf{X}^S, \mathbf{B}^S, \mathbf{X}^T, \tilde{\mathbf{B}}^T, \theta^S) = L_S + \alpha * L_U \quad (2)$$

$$L_S = \sum_{j=1}^{N^T} \sum_{k=1}^{N_j^T} (r(\theta^T | x_j^T, \tilde{b}_{j,k}^T) + c(\theta^T | x_j^T, \tilde{b}_{j,k}^T)) \quad (3)$$

$$L_U = L_{MMD}(\theta^T | \mathbf{X}^S, \mathbf{X}^T, \theta^S) \quad (4)$$

532 其中 L_S 是有监督的损失函数来学习基于场景的检测模型， L_U 是无监督的损
533 失函数。 $r(\cdot)$ 是为了预测区域定位的回归函数，比如norm-1损失函数， $c(\cdot)$ 是
534

为了预测区域置信度的分类函数，比如cross-entropy损失函数。 $L_{MMD}(\cdot)$ 是基于MMD的无监督正则化函数。其中常数 α 用来平衡监督损失函数和无监督损失函数的影响。在我们的实验中，它对于不同的场景具有鲁棒性，我们取 $\alpha = 10$ 作为所有实验的 α 值。

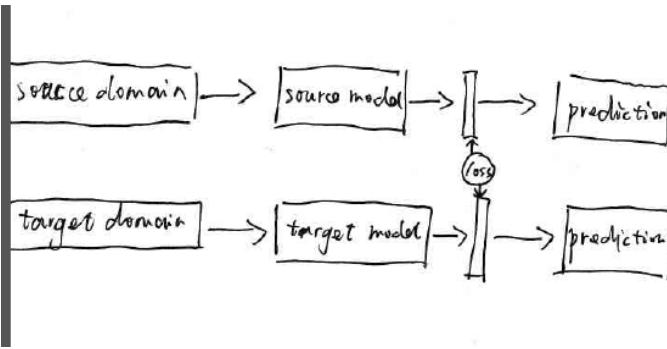


Fig. 5. One kernel at x_s (dotted kernel) or two kernels at x_i and x_j (left and right) lead to the same summed estimate at x_s . This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption. The last sentence of a figure caption should generally end without a full stop

5.1 检测网络

我们利用Russel et al.提出的模型来作为我们迁移框架流中的模型，这是一个端到端的人物检测模型，它不需要任何提前计算好的可能人物区域。这个检测网络包含了一个GoogLeNet [19]来将一张图片编码为一个(15x20x1024)的特征图，其中每一个1024维都是对应感应区域的数据表征，每一个感应区域对应了原图中的一个子区域。然后一个基于递归神经网络的层将批量大小为300的数据表征解码为具体的300*5个输入，包括预测区域的位置和其对应的置信度。最后所有子图上的输出总结在一起作为最好的检测结果。当我们在原场景的大量数据充分训练之下，得到的深度网络在待迁移场景有较高的准确度，然后它的召回率并不高。同时，不同于其他需要提前计算可能的预测区域并一一给出置信度的检测网络[18, 20]，这个检测网络直接输出了所有的具有高置信度的预测区域。因此，负样本中可能包含人头区域也可能包含非人头区域，这并不能作为迁移时的训练数据。

关于这个网络的损失函数，我们在迁移的过程中做了一些处理，而不是直接使用。首先我们把待迁移场景的数据输入到原场景训练的原模型中，那么我们就得到了一些自动标记的数据。那么如果我们直接将这些数据作为训练数据，而且不更改任何的网络的设定的情况下会怎么样？下面我们进行简单的分析。这些自动标记的数据是根据置信度高于一定阈值得到的，因此它们在网络最后的输出并不是完全的百分百置信度，比如0.93这样，因此它跟1之间还

有0.07的误差，可以作为损失对网络进行误差反传并更新，那么这种情况下网络有两种发展的可能，可能性一：网络参数发生变化，因此有可能会提高整体的表现，可能性二：网络对于自动标记图片产生过拟合，同时自动标记的数据中存在的错误也使得网络在识别上产生了疑惑。比较这两种可能性，那么可能性二更有可能在网络中发生，因为可能性一中，虽然网络参数发生了改变，但是未必是往好的方向，如果可能性二的问题没法办法减轻或者解决，那么单纯粗暴的把自动标记的数据丢到网络中去做训练并不能得到更好的结果。我们的实验也验证了这一点。当我们将自动标记的数据作为训练数据，同时网络的设定也没有任何改变的情况下，准确率和召回率都没有提高，反而因为过拟合和错误数据的加入使得网络的表现更加糟糕。

在网络设定不改变的情况下，没办法取得效果，那么我们考虑对网络最后的损失函数进行调整。首先分析自动标记数据，这些自动标记数据具有准确率高召回率低的特点，因此在这些标记数据中，真阳性样本的可能行很大，但是自动标记区域剩下的其他区域中，有可能有包含人物区域，也可能不包含人物区域。因此我们可以鼓励网络大胆预测自动标记的区域，而对于剩下的其他区域是否有人头保持保守的态度。对应到网络中，我们通过调整损失函数来解决这个问题。在将自动标记数据拿来训练的过程中，最后序列预测的区域中，如果自动标记的标记是1，那么进行误差反传，如果自动标记的标记是0，那么不进行误差反传。通过这种方式，我们引导网络来预测更多的人头。实验的结果表明这种方式确实可以大大的增加召回率。然后由此引来的另一个问题是准确率也大幅的下降。其中最重要的一个原因是关闭了负样本的误差反传，导致整个网络无法对非人物进行正常的反应。下面的章节将介绍如何解决这些问题。

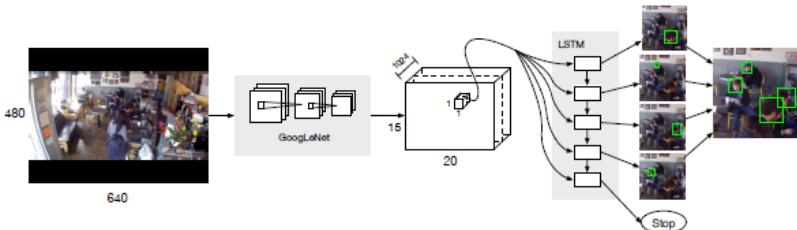


Fig. 6. One kernel at x_s (dotted kernel)

5.2 迭代算法

这个章节中我们将介绍迁移过程中的迭代算法。自动检测工具将待迁移场景中置信度高的图片作为下一次迭代需要的训练数据。为了生成第一次迭代的自动标记数据，我们利用在原场景充分训练好的原模型来生成待迁移场景的训练数据。就像章节5.1中提到的那样，由原模型自动标记得到的待迁移场景下的训练

630 数据具有低的召回率和高的准确率。在随后的迭代中，每次迭代需要的训练数
 631 据都有上一次迭代得到的更新的待迁移模型自动标记得到。在这些迭代中，这
 632 些自动标记的图片就作为训练数据来更新待迁移模型。

633 迭代算法在不断产生新的训练数据方面有重要的作用。这里包含两个方面
 634 增益。一方面，我们的目标是无监督的场景迁移。在这个设定下，有两种可行
 635 的办法，第一种就是直接学习原场景和待迁移场景数据之间的共性，例如映射
 636 到同一个高维空间或者提取到共享的特征向量，这些也在相关工作中提到，然
 637 后这种方法有很大的局限性，一则如果去找到两个场景之间数据的共性是一个
 638 需要人的智力的过程，一则这种途径学到的特征并不是适应于特殊场景的检测
 639 模型。另一种方法就是通过自动标记的方法来产生伪的真实数据，这样就能够
 640 通过通用的算法和模型来训练针对特定场景的检测网络。另一方面，如果只对
 641 第一次或者某一次自动标记的数据进行训练，那么在训练的后阶段不仅效果下
 642 降，而且很容易产生过拟合。迭代的算法便可以通过不断产生新的自动标记数据
 643 来更新训练数据集。当然这边也要提到迭代算法的局限性，那就是没有办法
 644 自动的找到停止迭代的最好时机，目前可以简单的人为设定最大迭代次数来终
 645 止迭代。同时下一个章节中的无监督正则化也提供了一个可行的方案，具体将
 646 下面的章节。

647 由于自动标记的待迁移场景的数据包含低的召回率。同时人物区域和非人
 648 物区域都混杂在负样本中，我们忽略了低置信度的区域的误差反传。也就是在
 649 训练的过程中我们鼓励网络更加大胆的预测正样本，而对于负样本保持保守的
 650 态度。这个策略无疑将导致许多非人物的区域都被预测为人物。当错误积累到
 651 一定程度，进一步的训练也无法提高召回率，反而适得其反。为了弥补真阴性
 652 数据的不足，我们将原场景中人工标记的图片数据加入到训练数据中。在我们的
 653 实验中，当训练这些来自原场景的标记图片时，我们只对那些具有低置信度
 654 的区域进行反传。同时章节5.3中的无监督损失函数加入了网络中进行正则化约
 655 约束。完整的迁移算法在算法1中给出。在一个提前指定的迭代次数极限到达之
 656 后，我们就获得了最后的在迁移场景下学习到的检测模型。

659 **Algorithm 1** Deep domain adaptation algorithm (to be completed)

```

660 1: procedure DEEP DOMAIN ADAPTATION
661 2: Train source model on source stream with abundant annotated data
662 3: Use well-trained source model on source stream to initialize model on target stream
663   as  $M_0$ 
664 4:   for  $i = 0:N^I$  do
665 5:      $M_i$  generate "fake ground truth"  $G_i$  of target domain
666 6:     balbal
667 7:     balbabla
668 8:      $G_i = G_i +$  random samples from source domain
669 9:     Take  $G_i$  as training data to upgrade  $M_i$  into  $M_{i+1}$ 
670 10:   end for
671 11:  $M_{N^I}$ : final model.
672 12: end procedure

```

675 5.3 Unsupervised weights regularizer on Element-wise Multiply 676 Layer

677 Element-wise Multiply Layer

679 在深度神经网络中，最后一层的特征向量被作为输入图片的重要表征。然
680 而，在这篇文章中我们往后一步，关注于最后一层的全连通层。这一层作为解
681 码器将最后一层特征向量中包含的丰富信息表达为最后的输出层。由于原模型
682 在充分的原场景数据下训练得到的，因此其最后一层全连通层的参数也很好的
683 收敛过。我们认为相比于在最后一层的特征向量加入正则化限制，如果在最
684 后这一层全连通层加入了正则化的约束将会取得更好的效果。首先，我们标记
685 最后一层特征向量、最后一层全连通层的参数和最后的输出对应为 $\mathbf{F}_{(N^B \times N^D)}$ ，
686 $\mathbf{C}_{(N^D \times N^O)}$ 和 $\mathbf{P}_{(N^B \times N^O)}$ 。最后一层全连通层的操作可以被表达为矩阵乘法：

$$687 \quad \mathbf{P} = \mathbf{F} * \mathbf{C} \quad (5)$$

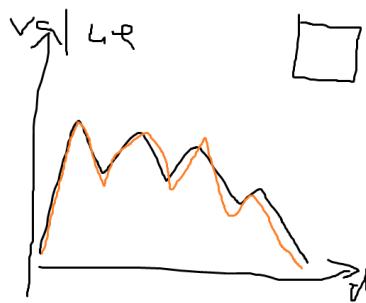
$$688 \quad P_{b,o} = \sum_d F_{b,d} * K_{d,o} \quad (6)$$

691 从中得到可以得到启发，我们将上述共识切分为两个子操作—元素点乘和求
692 和，同样这两个操作可以描述为：

$$694 \quad M_{b,o,d} = F_{b,d} * C_{d,o} \quad (7)$$

$$695 \quad P_{b,o} = \sum_d M_{b,o,d} \quad (8)$$

698 其中 $\mathbf{M}_{(N^B \times N^O \times N^D)} = [\mathbf{m}_{b,o}]$ 是元素点乘操作的参数张量。 $\mathbf{m}_{b,o}$ 是一个具
699 有 N^D 维的向量，这将作为无监督正则化约束的基础。最后，我们可以将最
700 后一层特征向量和最后的输出之间的最后的全连通层转化一个元素点乘层和
701 求和层。这个转化来的元素点乘层是最后一个在输出层之前的带有参数权重的
702 层。图片x展示了这个转化。



716 **Fig. 7.** One kernel at x_s (dotted kernel) or two kernels at x_i and x_j (left and right)
717 lead to the same summed estimate at x_s .

Unsupervised weights regularizer on Element-wise Multiply Layer

在一些工作中[decaf][10]，最后一层的特征向量被当做图片最后的表征。在场景迁移的任务中，当原模型在在大量原场景数据的训练之下，章节5.3中提到的元素点乘层也包含了丰富且重要的信息，直接导致了最后的输出。对于输出层的特定节点来说，它的最后输出值取决于被转化来的最后的元素点乘层，并且元素点乘层每一维对其的贡献并不是随机确定的。在这篇文章中，我们认为原场景和待迁移场景中数据在最后这层元素点乘层的分布应该是相似的。虽然在最后一层元素点乘层中，每一维可能对最后的输出产生影响，但是由于原模型是在大量的原场景的数据的训练之下得到的，那么它在元素点乘层上的分布应该具有代表性，且待迁移模型在迁移的过程中也应该保持一致。这里我们利用MMD (maximum mean discrepancy)来编码原场景数据和待迁移场景数据在元素点乘层上的分布的相似性。

$$L_{MMD}(\theta^T | \mathbf{X}^S, \mathbf{X}^T, \theta^S) = \frac{1}{N^O} \sum_{o=1}^{N^O} \left\| \frac{1}{N^B} \sum_{b=1}^{N^B} \mathbf{m}_{b,o}^T - \frac{1}{N^B} \sum_{b=1}^{N^B} \mathbf{m}_{b,o}^S \right\|^2 \quad (9)$$

这个公示也可以解释为对于所有输出节点的 $\mathbf{m}_{b,o}^T$ 和 $\mathbf{m}_{b,o}^S$ 的中心之间的欧式距离的平均值。在实验的过程中，将所有的数据集都拿进来计算数据分布是不现实的，然后数据太少得到的数据分布不稳定，难以做正则化约束，因此在我们的实验中， $L_{MMD}(\cdot)$ 损失函数是由每一次的批量训练中得到的。 $\mathbf{m}_{b_i,o}^S$ 和 $\mathbf{m}_{b_j,o}^S$ 的比较可以在图x中看到。从图中可以看到原场景不同的数据在最后一层元素点乘层的分布是很接近的。

对于将最后一层全连通层转化为元素点乘层和求和层以及在最后一层元素点乘层上做正则化约束是经过一系列的启示时候得到的。首先我们在实验过程中尝试了分别在GoogLeNet特征向量和递归神经网络层的隐含层加入正则化约束，实验结果表明递归神经网络层的隐含层加入正则化约束这一举动比在GoogLeNet特征向量加入正则化约束有更好的效果，一个明显的原因就是误差反传的机制，即误差反传只能从后往前传递，因此在GoogLeNet特征向量上加的正则化约束就没法对递归神经网络层产生影响，而递归神经网络又是直接对最后的输出产生影响的解码器，因此递归神经网络层的隐含层加入正则化约束可以取得更好的效果。其次我们尝试了除了求取 $\mathbf{m}_{b_i,o}$ 的中心并求欧式距离之外其他的可能的正则化形式。例如在 $\mathbf{m}_{b_i,o}$ 每一纬度的数据分布方差上也加入正则化或者将 $\mathbf{m}_{b_i,o}$ 的中心根据预测的置信度分为两个中心在加入正则化，然而效果并不好。最后考虑到如果将最后一层全连通层转化为元素点乘层和求和层，那么我们就可以将全连通层的参数也集中到元素点乘层中，在对元素点乘层进行正则化约束，那么不仅可以调整卷积神经网络和递归神经网络的参数，还可以调整最后一层全连通层的参数。我们的实验结果也表明了转化来的元素点乘层上的无监督正则化函数取得了比特征向量层更好的效果。

6 实验结果

在这个章节中，我们将介绍我们在监控场景中的表现和在标准场景迁移基准数据集上的表现。由于我们最早做场景迁移的动机在于更方便的部署深度检测网络到监控场景中，因此我们首先展示我们在密集场景中人物检测的表现。然后我们将我们的迁移算法应用到标准场景迁移基准数据集上来验证我们的算法的适用性。

Fig. 8. One kernel at x_s (*dotted kernel*) or two kernels at x_i and x_j (*left and right*) lead to the same summed estimate at x_s .

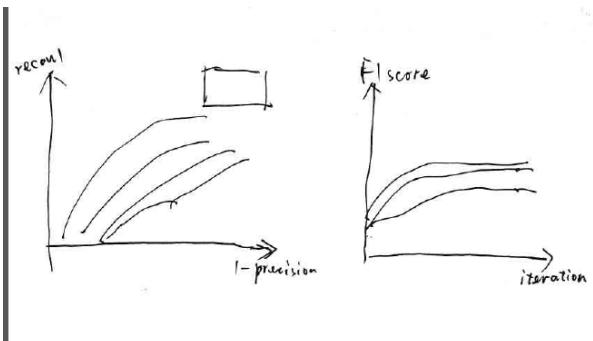
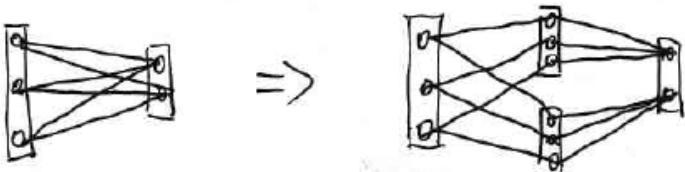


Fig. 9. One kernel at x_s (*dotted kernel*) or two kernels at x_i and x_j (*left and right*)

6.1 监控场景中的场景迁移

数据集和评估标准

为了展示我们的场景迁移算法在人物检测任务中的效果，我们收集了一个包含了3个待迁移场景的数据集。这三个待迁移场景分别包含了1308, 1213和331张没有标记的图片。对于每个场景，我们额外标记了100张图片作为最后的测试数据。我们在人工标记的工程中并不是将整个人的身体都标记出来，而是将其头部标记出来。这样做的原因是在室内的字符检测或者密集场景中的人物检测，由于遮挡等原因，一个人的身体部分很可能是看不到的，选择人物头部作为人物的代表可以解决这个问题。原场景的数据集是来自Brainwash数据集（<http://d2.mpi-inf.mpg.de/datasets>）。这个数据集包含了超过11917张来自三个场景的带标记的图片。原场景和待迁移场景的示意图如图x所示。

我们对于检测结果的评估方法来自于PASCAL VOC [21]中所定义的标准。为了判断一个预测区域是不是能够匹配一个真实的人物区域，他们之间的交集面积必须超过他们的合集面积的50%。如果多个检测结果预测了同一个人物，那么也只能当作一个正确率的预测。我们在图x中画出了准确率-召回率曲线图，同时，在迁移算法的迭代过程中的F1值 $F1 = 2 * precision * recall / (precision + recall)$ 也展示在图x中。

实验设定

我们使用了深度学习框架Caffe [22]作为我们的迁移学习平台。在场景迁移的过程中，我们将网络的学习率设为0.01，动量设为0.5。在初始化阶段，GoogLeNet的权重首先用来初始化愿框架流的原模型，同时递归神经网络层的参数根据均一分布随机初始化。在每一次迭代中，100张自动标记的待迁移场景的图片和1000张原场景的标记图片交互的用来训练待迁移场景的网络。检测网络的输出包括预测区域的坐标及其对应的置信度，因此在最后一层特征向量和最后的输出之间有两个全连通层。经过我们的实验，发现当无监督的MMD正则化函数已经加到输出预测区域置信度的元素点乘层时，再将无监督的MMD正则化函数加到输出预测区域坐标的元素点乘层并不能更好的表现。我们的迁移算法在3个待迁移场景分别做了实验。

对比方法

为了展示算法的有效性，我们比较了4个不同的方法，其中方法4是我们最后的迁移算法：

1. 只有待迁移场景自动标记的数据做训练，不加任何的无监督正则化函数。
 2. 只有待迁移场景自动标记的数据做训练，无监督的MMD正则化函数加到最后一层全连通层转化的元素点乘层。
 3. 待迁移场景自动标记的数据和来自原场景的标记数据做训练，无监督的MMD正则化函数加到最后一层特征向量层。
 4. 待迁移场景自动标记的数据和来自原场景的标记数据做训练，无监督的MMD正则化函数加到最后一层全连通层转化的元素点乘层。

图x画出了上诉几个比较方法的在待迁移场景1中的准确率-召回率曲线。同时，在迁移过程中每一次迭代的F1值的变化也在图x中显示出来。表格x给出了4个比较方法在三个迁移场景下，当F1的值达到最大时相应的准确率和召回率。迁移算法的结果示意图如图y所示。

	Scene 1			Scene 2			Scene 3			
	1-Pr	Re	F1	1-Pr	Re	F1	1-Pr	Re	F1	
method 0	0.101	0.187	0.309	0.059	0.599	0.732	0.190	0.476	0.599	855
method 1	0.245	0.408	0.530	0.632	0.905	0.524	0.176	0.778	0.800	856
method 2	0.284	0.476	0.572	0.012	0.837	0.906	0.078	0.653	0.764	857
method 3	0.109	0.496	0.637	0.002	0.721	0.838	0.044	0.611	0.746	858
method 4	0.140	0.530	0.656	0.006	0.811	0.893	0.097	0.778	0.836	859

实验结果

从表格x和图x中，我们有以下结论：

- 使用了迭代算法来更新待迁移场景中的模型的方法1、2、3、4的召回率数值比直接使用在原场景训练的原模型的方法0更高。这表现了我们的迭代算法在其自动标记和迭代式训练的有效性。
- 对比于方法1，方法2有着更高的F1值。他们区别在于是否在损失函数里面添加了无监督的正则化函数，证明了我们的算法可以抑制数据错误同时提高最终的召回率。
- 方法4对比于方法2有着更高的F1值，这表明了增加的来自原场景的标记数据有助于减少在迁移过程中针对待迁移场景负样本不反传的错误。
- 对比于方法3，方法4的召回率更高了，这表明了转化来的元素点乘层上的无监督正则化函数取得了比特征向量层更好的效果。

通过上述三个不同待迁移场景的实验，可以看我们的算法都取得了很好的效果，因此这使得将该检测网络在没有任何标记数据的情况下迁移到不同的待迁移场景更加容易。在实际的部署中，我们只需要首先从特定的监控场景中采集一定量的无标记图片，然后将这些图片送到特定的服务器进行场景迁移算法的训练，在一定的迭代次数之后，将训练好的模型返回给该待迁移场景，就可以直接使用。这边还存在两个工程方面的问题，一个是如何决定停止迭代的次数，在迁移算法迭代的过程中，模型的表现呈现J性，即刚刚开始F1的值会越来越高，然而到了一定程度之后F1的值便会开始下降，F1值下降的直接结果就是每次自动标记的数据的质量也会下降，最终影响到模型的表现。在实验的过程中，我们发现加了MMD正则化约束之后，模型的F1值并不会产生大的震动，网络的表现也不会突然急剧下降，因此在这一训练过程中可以找到一个相对较优的模型。另一个问题就是虽然文章的迁移算法可以取得不错的效果，但是有时候监控应用需要极高的准确率，那在这种情况下，没法避免的需要标记的数据来保证准确率，但是我们迁移算法也同样可以在有监督的数据的场景迁移中，帮忙模型取得更好的表现。更具体的例子在下面对于标准迁移学习基准数据库的实验可以看出。

6.2 Domain Adaptation on Standard Classification Benchmark

Office dataset

Office数据集[1]包含了来自三个场景(Amazon, DSLR, Webcam)的31个分类。图x给出了样例图片。由于Amazon场景有着最多的2817张标记图片，我们把它作为原场景，将Webcam场景作为待迁移场景。训练数据的使用遵从了标准的有监督和无监督的设定。具体来说，对于无监督学习，我们在原场

900 景Amazon每个类别随机采样了20张图片作为训练数据，对于有监督学习，我们
 901 额外在待迁移场景中每类选取了3个标记图片作为训练数据。在两种实验设定
 902 中，剩下的图片都用来作为评估以及用于无监督的MMD正则化计算。

903 Office数据集的实验跟上面一节的实验存在着相似和不同的地方。相似的地
 904 方在于：Amazon数据集中的图片包含了大量的抠出来的图片，而且图片的色泽
 905 也没有收到背景的影响，同一类中的图片虽然差异很大，但是恰恰包含了同一
 906 种物体的大量变种和变化，这使得将Amazon场景的数据集迁移到其他特定场景
 907 更加容易。Amazon很适合作为原场景。不同的地方在于：这三个场景整体的数
 908 据量都不多，有些场景的某个类的数量甚至只有7张，而我们的MMD正则化函
 909 数需要通过足够量的数据来计算出一个稳定的值，因此MMD正则化函数在总体
 910 图片数据量少的情况下，其表现也会受到影响。



911
 912
 913
 914 Fig. 6. Some examples from three domains in the Office dataset.
 915
 916
 917
 918
 919

920 Fig. 10. One kernel at x_s (dotted kernel) or two kernels at x_i and x_j (left and right)
 921
 922

923 Experimental settings and network design

924 在有监督的实验设定中，我们服用了在人物检测中的框架流。我们利
 925 用AlexNet [23]作为两个框架流的基本模型。首先，我们利用Amazon场景提
 926 供的训练数据训练愿框架流的原模型。然后在章节5.2中的迭代算法作为场景迁
 927 移的基本训练方法。自动标记工具将待迁移场景中置信度高于0.9的作为下一次
 928 迭代的训练数据。在每一次迭代过程中，随机从训练数据中挑选100张来更新模
 929 型。无监督的MMD正则化函数加到最后一层全连通层转化的元素点乘层，损失
 930 函数4中常数 α 设为10，这同人物检测中利用的常数设定相同。在这一过程中，
 931 需要注意到待迁移场景的人工标记数据以及待迁移场景的自动标记数据之
 932 间的比例问题。首先如果人工标记数据过多，那么一则自动标记的数据没有办法发
 933 挥作用，二则人工标记的数据在不断训练过程中也容易过拟合。其次如果自动
 934 标记数据过多，那么人工标记数据的作用就没法发挥出来，而人工标记的数据
 935 其正确率应该是百分百的，更有利于网络的训练。因此在实验过程中我们采取
 936 了这样的数据选取方式：由于每次迭代都是从数据集中选出100张图片，因此
 937 我们关注于人工标记数据和自动标记数据之间的比例，在第一次迭代中，我们
 938 将比例设为1:1进行训练，在之后的迭代中，我们将比例调为1:1.1，以此类推，
 939 我们每一次新的迭代都将自动标记数据的比例提高0.1，知道所有的待迁移场
 940 领的数据都被自动标记工具选取过一遍。这样做的优势在于刚刚开始时，我们充
 941 足利用了自动标记的数据，同时在之后的迭代中逐渐引入人工标记的数据，以
 942 便模型的泛化能力。在实验中，我们使用了Adam优化器，学习率为0.001，动
 943 学量为0.9，权重衰减率为0.0005，批处理大小为100，每张图片的尺寸为224x
 944 224。我们在TensorFlow上实现了整个框架，使用了NVIDIA GeForce GTX 980显
 945 卡和Intel i7-6700K CPU。整个实验耗时约10小时，得到了一个准确率为85%的
 946 模型。该模型在测试集上的表现与文献[23]中报道的85.5%相当。

945 分发挥人工标记数据的作用，在之后的迭代中，提高自动标记数据的比例以让
 946 其发挥作用，同时避免了人工标记数据不断训练产生的过拟合。我们的实验也
 947 表明了这种策略相比与固定比例的策略可以取得更好的效果。就Amazon场景
 948 到DSLR场景的有监督场景迁移实验来说，准确率从77%提高到84%。
 949

950 对于无监督情况，我们使用了于有监督相同的实验设定和网络模型，除了在
 951 迁移的过程中，待迁移场景没有提供任何的可以添加到训练数据的人工标记数
 952 据。无监督的设定更适合与我们算法，因为从有监督的情况来看，还需要对人
 953 工标记数据和自动标记数据之间的比例进行动态调整才能取得好的效果，而无
 954 监督的情况下除了框架流中的模型，我们并不需要对自己的迁移学习的算法进
 955 行修改就可以取得很好的效果。
 956

Performance evaluation

957 在表格x中，我们跟6个最近发布的在有监督和无监督设定下的工作的结果。
 958 我们的方法在两种设定下都超过了其他方法的结果。这更加证实了我们的迭代
 959 算法以及无监督MMD正则化函数加到最后一层全连通层转化的元素点乘层在不
 960 同运用下的效果。尤其是在无监督的情况下，一方面我们通过迭代算法不断产
 961 生新的自动标记图片来为训练提供训练数据，另一方面我们利用MMD无监督正
 962 则化工具很好的抑制了自动标记数据中的错误所对网络产生的影响。通过上一
 963 节中的实验以及这一节的实验，表明了我们提出的场景迁移的算法可以适用于
 964 多种任务包括检测和分类，以及在有监督和无监督的设定之下都可以取得不错
 965 的效果。再次证明了方法的有效性。
 966

	$A \rightarrow W$	
	Supervised	Unsupervised
GFK(PLS,PCA)[24]	46.4	15.0
SA [25]	45.0	15.3
DA-NBNN [26]	52.8	23.3
DLID [27]	51.9	26.1
DeCAF ₆ S [28]	80.7	52.2
DaNN [16]	53.6	35.0
Ours	84.3	66.3

7 Conclusions

982 在这篇文章中，首先我们通过对卷积神经网络+递归神经网络的方法实现人
 983 头检测的算法和利用卷积神经网络+前馈神经网络实现的检测算法相比较，找
 984 到更适合于实际监控场景中部署人物检测功能的算法，包括了从检测速度到检
 985 测的准确率，我们选择了卷积神经网络+递归神经网络的方法来进行人头检测
 986 的算法。
 987

988 同时由于监控场景的大量变化以及人工标记工作的繁重，我们提出了无监督
 989 场景迁移的算法，即如何在需要提供任何的待迁移场景的标记数据的情况下，
 将原场景训练的神经网络应用于新的待迁移场景。我们首先提出了通过迭代

990 算法来不断产生自动标记数据，其次我们通过对检测网络的损失函数的修改，
991 使其减少自动标记数据没有负样本的影响。由于自动标记数据中缺少真阴性数
992 据和伪阳性数据，直接训练会导致伪阳性的比例迅速上升，同时影响到了召回
993 率的进一步提高。为了减少错误数据的影响，我们提出了一下两个方法来约束
994 深度网络的训练。一方面，我们加入了原场景的标记数据，另一方面，添加无
995 监督正则化函数的方法来约束网络。其中我们需要对最后一层全连通层进行转
996 化，产生新的元素点乘层和求和层，然后在元素点乘层上加入MMD无监督正则
997 化函数来防止伪阳性结果的迅速增加以及增强召回率。
998

999 我们在自己建立的待迁移场景数据库上进行了无监督场景迁移的实验，以及
1000 在标准的场景迁移基准数据集上进行了有监督和无监督的场景迁移实验。我们的
1001 算法在这两部分实验的上的表现表明了我们的算法可以适用于多种任务包括
1002 检测和分类，以及在有监督和无监督的设定之下都可以取得不错的效果。
1003

References

1. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Computer Vision–ECCV 2010. Springer (2010) 213–226
2. Kulis, B., Saenko, K., Darrell, T.: What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE (2011) 1785–1792
3. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 999–1006
4. Huang, J., Gretton, A., Borgwardt, K.M., Schölkopf, B., Smola, A.J.: Correcting sample selection bias by unlabeled data. In: Advances in neural information processing systems. (2006) 601–608
5. Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., Schölkopf, B.: Covariate shift by kernel mean matching. Dataset shift in machine learning **3**(4) (2009) 5
6. Wang, X., Wang, M., Li, W.: Scene-specific pedestrian detection for static video surveillance. Pattern Analysis and Machine Intelligence, IEEE Transactions on **36**(2) (2014) 361–374
7. Zeng, X., Ouyang, W., Wang, M., Wang, X.: Deep learning of scene-specific classifier for pedestrian detection. In: Computer Vision–ECCV 2014. Springer (2014) 472–487
8. Hattori, H., Naresh Boddeti, V., Kitani, K.M., Kanade, T.: Learning scene-specific pedestrian detectors without real data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3819–3827
9. Leibe, B., Seemann, E., Schiele, B.: Pedestrian detection in crowded scenes. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Volume 1., IEEE (2005) 878–885
10. Barinova, O., Lempitsky, V., Kholi, P.: On detection of multiple object instances using hough transforms. Pattern Analysis and Machine Intelligence, IEEE Transactions on **34**(9) (2012) 1773–1784
11. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 (2013)



Fig. 11. One kernel at x_s (*dotted kernel*) or two kernels at x_i and x_j (*left and right*)

- 1080 12. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for ac-
1081 curate object detection and semantic segmentation. In: Proceedings of the IEEE
1082 conference on computer vision and pattern recognition. (2014) 580–587
1083 13. Zhang, S., Benenson, R., Schiele, B.: Filtered channel features for pedestrian de-
1084 tection. In: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Con-
1085 ference on, IEEE (2015) 1751–1760
1086 14. Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I.J., Lavoie,
1087 E., Muller, X., Desjardins, G., Warde-Farley, D., et al.: Unsupervised and transfer
1088 learning challenge: a deep learning approach. ICML Unsupervised and Transfer
1089 Learning **27** (2012) 97–110
1090 15. Gong, B., Grauman, K., Sha, F.: Connecting the dots with landmarks: Discrimi-
1091 natively learning domain-invariant features for unsupervised domain adaptation.
1092 In: Proceedings of The 30th International Conference on Machine Learning. (2013)
1093 222–230
1094 16. Ghifary, M., Kleijn, W.B., Zhang, M.: Domain adaptive neural networks for object
1095 recognition. In: PRICAI 2014: Trends in Artificial Intelligence. Springer (2014)
1096 898–904
1097 17. Pishchulin, L., Jain, A., Wojek, C., Andriluka, M., Thormählen, T., Schiele, B.:
1098 Learning people detection models from few training samples. In: Computer Vision
1099 and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE (2011) 1473–
1100 1480
1101 18. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on
1102 Computer Vision. (2015) 1440–1448
1103 19. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D.,
1104 Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings
1105 of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1–9
1106 20. Vu, T.H., Osokin, A., Laptev, I.: Context-aware cnns for person head detection.
1107 In: Proceedings of the IEEE International Conference on Computer Vision. (2015)
1108 2893–2901
1109 21. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisser-
1110 man, A.: The pascal visual object classes challenge: A retrospective. International
1111 Journal of Computer Vision **111**(1) (2015) 98–136
1112 22. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama,
1113 S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding.
1114 arXiv preprint arXiv:1408.5093 (2014)
1115 23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep
1116 convolutional neural networks. In: Advances in neural information processing systems.
1117 (2012) 1097–1105
1118 24. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised
1119 domain adaptation. In: Computer Vision and Pattern Recognition (CVPR), 2012
1120 IEEE Conference on, IEEE (2012) 2066–2073
1121 25. Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T.: Unsupervised visual do-
1122 main adaptation using subspace alignment. In: Proceedings of the IEEE Interna-
1123 tional Conference on Computer Vision. (2013) 2960–2967
1124 26. Tommasi, T., Caputo, B.: Frustratingly easy nbnn domain adaptation. In: Proceed-
1125 ings of the IEEE International Conference on Computer Vision. (2013) 897–904
1126 27. Chopra, S., Balakrishnan, S., Gopalan, R.: Dlid: Deep learning for domain adap-
1127 tation by interpolating between domains. In: ICML workshop on challenges in
1128 representation learning. Volume 2. (2013)

- 1125 28. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.:
1126 Decaf: A deep convolutional activation feature for generic visual recognition. arXiv
1127 preprint arXiv:1310.1531 (2013)

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169