# 深度迁移学习

## 一、 深度学习

### 1) ImageNet Classification with Deep Convolutional Neural Networks

**主要思想：**

该神经网络有 6000 万个参数和 650,000 个神经元，由五个卷积层，以及某些卷积层后跟着的 max-pooling 层，和三个全连接层，还有排在最后的 1000-way 的 softmax 层组成。使用了非饱和的神经元和一个非常高效的 GPU 关于卷积运算的工具。

1、采用了最新开发的正则化方法，称为“dropout”。

2、采用 ReLU 来代替传统的 tanh 引入非线性,;

3、采用 2 块显卡来进行并行计算，减少了更多显卡需要主机传递数据的时间消耗，在结构上，部分分布在不同显卡上面的前后层节点之间无连接，从而提高了训练速度;

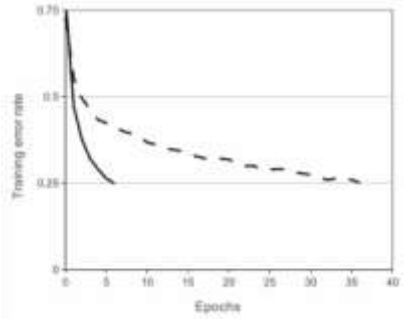
4、同层相邻节点的响应进行局部归一化提高了识别率（top5 错误率降低 1.2%);

5、有交叠的 pooling（top5 错误率降低 0.3%);

**体系架构：**

（1） ReLU

训练带 ReLUs 的深度卷积神经网络比带 tanh 单元的同等网络要快好几倍。如下图,带 ReLU 的四层卷积神经网络(实线)在 CIFAR-10 数据集上达到 25%训练误差率要比带 tanh 神经元的同等网络（虚线）快六倍。




（2） 在多个 GPU 上训练

（3） 局部响应归一化

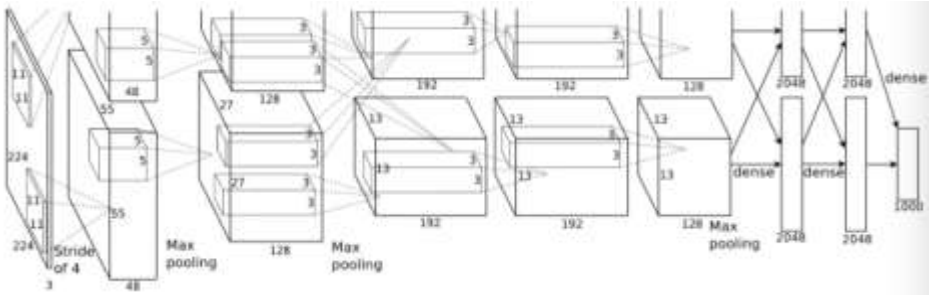
具体见 Very Deep Convolutional Networks for Large-Scale Image Recognition

（4） 重叠 Pooling

每个网格间隔距离为 s，而每一次进行降采样将从网格中心为中心，采样 z*z 个像素。如果 s=z，则与传统方法相同，而如果 s<z，则会进行重复采样。本文章将 s=2，z=3，成功的将 Top-1 和 Top-5 的错误率分别降低了 0.4%和 0.3%（与 s=2，z=2 相比)。而且，在实验中发现，采用重叠采样将会略微更难产生过拟合。

（5） 总体结构

该网络包括八个带权层；前五层是卷积层，剩下三层是全连接层。最后一个全连接层的输出被送到一个 1000-way 的 softmax 层，其产生一个覆盖 1000 类标签的分布。




响应归一化层跟在第一、第二个卷积层后面。最大 Pooling 层，跟在响应归一化层以及第五个卷积层之后。ReLU 非线性应用于每

个卷积层及全连接层的输出。第一个卷积层利用 96 个大小为 11×11×3、步长为 4 个像素的核，来对大小为 224×224×3 的输入图像进行滤波。第二个卷积层是 256 个大小为 5×5×48，第三、第四和第五个卷积层彼此相连，没有任何介于中间的 pooling 层与归一化层。第三个卷积层有 384 个大小为 3×3×256 的核，第四个卷积层拥有 384 个大小为 3×3×192 的核，第五个卷积层拥有 256 个大小为 3×3×192 的核。全连接层都各有 4096 个神经元。

（6）防止过拟合

- 数据增强

数据增强的第一种形式由生成图像转化和水平反射组成。该网络通过提取五个 224×224 的碎片（四个边角碎片和中心碎片）连同它们的水平反射（因此总共是十个碎片）做出了预测，并在这十个碎片上来平均该网络的 softmax 层做出的预测。

数据增强的第二种形式包含改变训练图像中 RGB 通道的强度。遍及整个 ImageNet 训练集的 RGB 像素值集合中执行 PCA。对于每个训练图像，我们成倍增加已有主成分，比例大小为对应特征值乘以一个从均值为 0，标准差为 0.1 的高斯分布中提取的随机变量。

- Dropout

（7）学习的详细过程

使用随机梯度下降法和一批大小为 128、动力为 0.9、权重衰减为 0.0005 的样例来训练。

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$$
$$w_{i+1} := w_i + v_{i+1}$$

我们用一个均值为 0、标准差为 0.01 的高斯分布初始化了每一层的权重。我们用常数 1 初始化了第二、第四和第五个卷积层以及全连接隐层的神经元偏差。在其余层用常数 0 初始化神经元偏差。

对于所有层都使用了相等的学习率，在整个训练过程中手动调整的。当验证误差率在当前学习率下不再提高时，就将学习率除以 10。学习率初始化为 0.01，在终止前降低三次。

## 2) Very Deep Convolutional Networks for Large-Scale Image Recognition

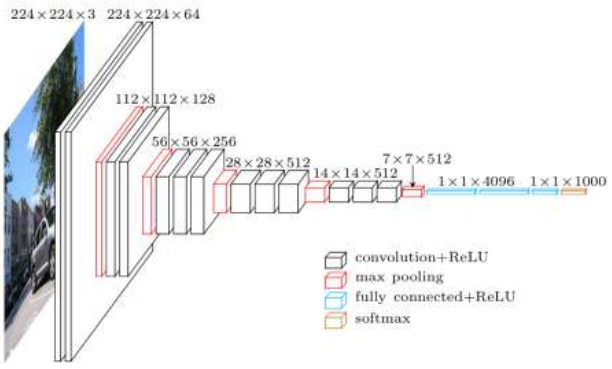
**主要思想：**

解决 ConvNet 结构设计中“深度”这个重要问题。提出用 3*3 的卷积增加网络的深度，当层数增加到 16-19 层时，分类准确度得到明显提升。网络具有很强的泛化能力。

**主要步骤：**

（1） 主要结构

- 输入是固定大小的 224*224RGB 图像。
- 唯一的预处理是减均值。
- 网络由 3*3 的卷积层栈组成，有些地方也利用了 1*1 的卷积过滤器。卷积层的 stride 固定为 1，对 3*3 的卷积层 padding 为 1，以保持空间分辨率不变。
- 通过 5 个 Max-pooling 通过 2*2 的窗口，stride 为 2 进行空间池化。
- 卷积层栈后面跟的是 3 个 Fully_connected 层。前 2 个是 4096 way，第 3 个是 1000 way。最后一层是 softmax。
- 隐藏层都用 ReLU 进行非线性化。
- 在所有的网路中，除一个用了 LRN 外，其他都没有用。因为在 ILSVRC 数据库上不能增强性能，但是导致了内存和计算时间的消耗。




注：

LRN(Local Response Normalization) 局部响应归一化

$$b_{x,y}^{i}=a_{x,y}^{i}/\left(k+\alpha\sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)}(a_{x,y}^{j})^{2}\right)^{\beta}$$

对任意的卷积核（kernel），选取左右临近（adjacent）的 n 个卷积核，将它们在同一个位置上卷积的结果进行求和。i 表示第 i 个核在位置(x,y)运用激活函数 ReLU 后的输出,n 是同一位置上临近的 kernal map 的数目,N 是 kernal 的总数。参数 K,n,alpha,belta 都是超参数，一般设置 k=2,n=5,aloha=1*e-4,beta=0.75。

（2） 配置

- 5 个网络（A-E），配置完全一样，除了深度，层数从 11 层到 16 层。conv 层的 channel 开始是 64，在每一个 max-pooling 后增加一倍，直到达到 512。

| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64 **LRN** | conv3-64 **conv3-64** | conv3-64 conv3-64 | conv3-64 conv3-64 | conv3-64 conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128 **conv3-128** | conv3-128 conv3-128 | conv3-128 conv3-128 | conv3-128 conv3-128 |
| maxpool | | | | | |
| conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 **conv1-256** | conv3-256 conv3-256 **conv3-256** | conv3-256 conv3-256 conv3-256 **conv3-256** |
| maxpool | | | | | |
| conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 **conv1-512** | conv3-512 conv3-512 **conv3-512** | conv3-512 conv3-512 conv3-512 **conv3-512** |
| maxpool | | | | | |
| conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 **conv1-512** | conv3-512 conv3-512 **conv3-512** | conv3-512 conv3-512 conv3-512 **conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

其中，文中使用 3*3 卷积核，2 个 3*3 的卷积层栈相当于一个 5*5 的卷积，而 3 个则相当于一个 7*7 的卷积。一是，用了 3 个非线性修正层,使得决策函数具有更强的判别能力;二是,减少了参数的数量,假设一个 3 层的 3*3 卷积层栈,输入输出都是 C 个 channel，则需要的参数是 3（3*3*C*C)=27*C*C，而一个 7*7 的卷积层，则需要参数 7*7*C*C=49*C*C。

1*1 卷积层是一种在不影响 conv 感受野的前提下增加决策函数非线性能力的一种方式。

（3） 训练

- 训练采用带 momentum 的 mini-batch 梯度下降优化多项式逻辑回归目标函数。batch size 256，momentum 0.9。
- 训练的正则化处理：权重衰减（L2 惩罚因子设置为 5*10^(-4)），前两个全连接层 dropout radio 设置为 0.5。学习率设置为 0.01，当验证精确度停止提高时，下降为原来的 0.1，一共下降 3 次，在 370k(74 epochs)迭代后停止。
- 浅层的网络（A）上采用随机初始化参数的方法进行训练，然后，当训练较深网络结构时，前 4 个卷积层和最后 3 个全连接层采用 net A 的参数。作者没有降低预初始化层的学习率，而是允许在学习过程中改变。
- 对于随机初始化，从 0 均值，方差为 0.01 的正态分布中采样。biase 初始化为 0。
- 第一种是针对单尺度训练,固定 S。(两个固定值 S=256,S=384)第二种方法是设置多尺度训练,让训练图像的 S 在[Smin,Smax]范围内随机，本文采用 Smin=256,Smax=512。

（4） 测试

采用 multi-crops 和 dense evaluation,multi-crops 比 dense evaluation 略好,两种方法的结合效果比单独使用任意一种都好,可能由于卷积边界的不同处理方式导致的互补，前者是 zero-padding，后者是邻域值 padding。

### 3) Going deeper with convolutions

**主要思想：**

目的：提升深度神经网络的性能。

一般方法带来的问题：增加网络的深度与宽度。

带来两个问题：

（1）参数增加，数据不足的情况容易导致过拟合

（2）计算资源要求高，而且在训练过程中会使得很多参数趋向于 0，浪费计算资源。

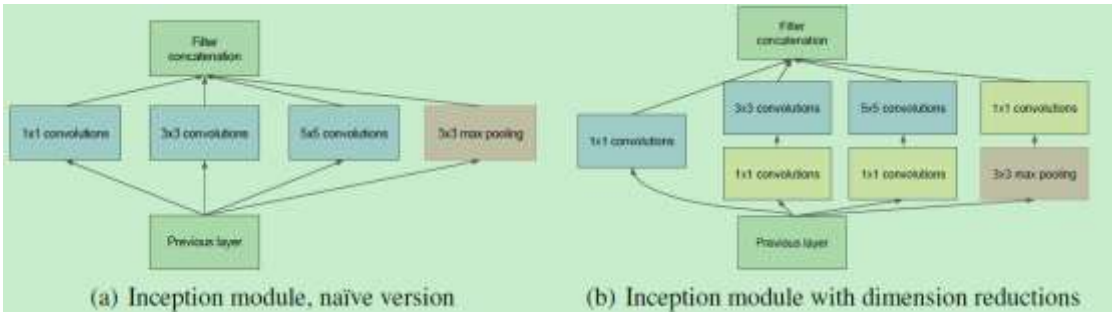
解决方法：使用稀疏连接替代稠密结构。

存在问题：计算机的基础结构在遇到稀疏数据计算时会很不高效，使用稀疏矩阵会使得效率大大降低。

目标：设计一种既能利用稀疏性，又可以利用稠密计算的网络结构。

**Inception 模型：**

主要思路是用密集成分来近似最优的局部稀疏结构。分析前一层的相关性，然后将高相关的神经元聚类。这些类作为下一层的神经元，并且和前一层的神经元相连接。




1、采用不同大小的卷积核意味着不同大小的感受野，最后拼接意味着不同尺度特征的融合；

2、卷积核大小采用 1、3 和 5，主要是为了方便对齐。

3、使用 5x5 的卷积核仍然会带来巨大的计算量。在 3*3 和 5*5 卷积前采用 1x1 卷积核来进行降维。

4、Inception 里面嵌入 pooling

5、高层次提取高抽象性的特征，空间集中性会降低，因此 3x3 和 5x5 的卷积核在更高层会比较多。

6、保持低层为传统卷积方式不变，只在较高的层开始用 Inception 模块。

**GoogLeNet（22 层）：**

网络中所有的卷积，包括 Inception 中的卷积， 都使用 ReLU 激活函数， 输入图像是 224*224 的 RGB 图像，做减均值预处理。#3×3 reduce 和 #5×5 reduce 代表 在 3×3 和 5×5 卷积之前进行的 1×1 的降维处理层，这些降维层也使用 ReLU 激活函数。

| type | patch size/ stride | output size | depth | #1×1 | #3×3 reduce | #3×3 | #5×5 reduce | #5×5 | pool proj | params | ops |
|---|---|---|---|---|---|---|---|---|---|---|---|
| convolution | 7×7/2 | 112×112×64 | 1 | | | | | | | 2.7K | 34M |
| max pool | 3×3/2 | 56×56×64 | 0 | | | | | | | | |
| convolution | 3×3/1 | 56×56×192 | 2 | | 64 | 192 | | | | 112K | 360M |
| max pool | 3×3/2 | 28×28×192 | 0 | | | | | | | | |
| inception (3a) | | 28×28×256 | 2 | 64 | 96 | 128 | 16 | 32 | 32 | 159K | 128M |
| inception (3b) | | 28×28×480 | 2 | 128 | 128 | 192 | 32 | 96 | 64 | 380K | 304M |
| max pool | 3×3/2 | 14×14×480 | 0 | | | | | | | | |
| inception (4a) | | 14×14×512 | 2 | 192 | 96 | 208 | 16 | 48 | 64 | 364K | 73M |
| inception (4b) | | 14×14×512 | 2 | 160 | 112 | 224 | 24 | 64 | 64 | 437K | 88M |
| inception (4c) | | 14×14×512 | 2 | 128 | 128 | 256 | 24 | 64 | 64 | 463K | 100M |
| inception (4d) | | 14×14×528 | 2 | 112 | 144 | 288 | 32 | 64 | 64 | 580K | 119M |
| inception (4e) | | 14×14×832 | 2 | 256 | 160 | 320 | 32 | 128 | 128 | 840K | 170M |
| max pool | 3×3/2 | 7×7×832 | 0 | | | | | | | | |
| inception (5a) | | 7×7×832 | 2 | 256 | 160 | 320 | 32 | 128 | 128 | 1072K | 54M |
| inception (5b) | | 7×7×1024 | 2 | 384 | 192 | 384 | 48 | 128 | 128 | 1388K | 71M |
| avg pool | 7×7/1 | 1×1×1024 | 0 | | | | | | | | |
| dropout (40%) | | 1×1×1024 | 0 | | | | | | | | |
| linear | | 1×1×1000 | 1 | | | | | | | 1000K | 1M |
| softmax | | 1×1×1000 | 0 | | | | | | | | |

通过训练来提升这些中间层的分类器效果，增加回传的梯度强度。在训练时，这些分类器的误差加入到系统总体误差，（乘以一个 0.3 的系数）。整个网络额外的结构（包括辅助分类器）如下：

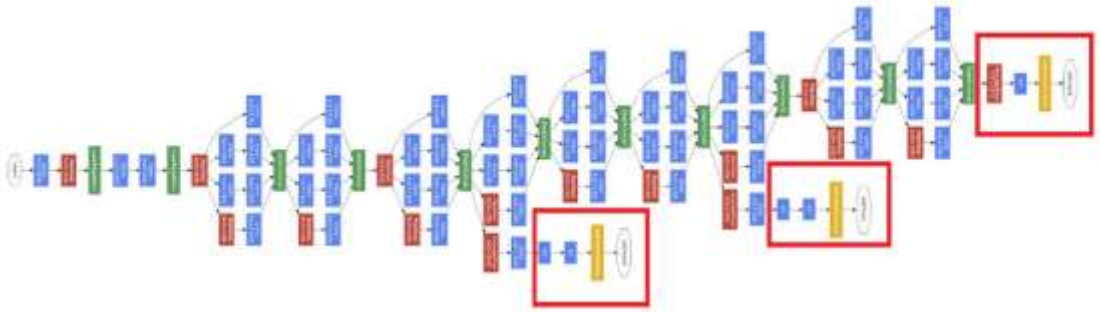
1）一个 5*5 的均值池化，步长 3， 这样网络（4a）的输出就是 4×4×512 ，网络（4d）的输出就是 4×4×528

2）128 个 1*1 卷积用于降维， 使用 ReLU 激活函数

3）一层 1024 个神经元的全链接层，使用 ReLU 激活函数

4）一个 dropout 层， 70% 概率

5）一个 softmax 线性层作为分类器



**训练：**

训练采用随机梯度下降，冲量 momentum：0.9，固定学习率每 8 个 epochs 减小 4%。

## 4) Deep Residual Learning for Image Recognition

**主要思想：**

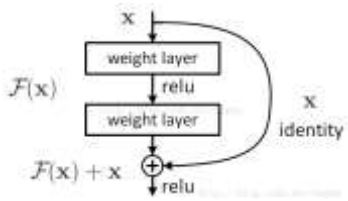
- 作者根据输入将层表示为学习残差函数，能够通过增加相当的深度来提高准确率。
- 核心是解决了增加深度带来的副作用（退化问题），这样能够通过单纯地增加网络深度，来提高网络性能。

**残差学习：**

将 H(x)看作一个由部分堆叠的层（并不一定是全部的网络）来拟合的底层映射，其中 x 是这些层的输入。假设多个非线性层能够逼近复杂的函数，这就等价于这些层能够逼近复杂的残差函数， H(x)-x （假设输入和输出的维度相同）。所以我们明确的让这些层来估计一个残差函数：F(x):=H(x)-x 而不是 H(x)。因此原始函数变成了：F(x)+x。

恒等映射：

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \quad (1)$$




图中的例子包含两层，$\mathcal{F} = \mathcal{W}_2\sigma(\mathcal{W}_1\mathrm{x})$，其中 σ 代表 ReLU，为了简化省略了偏置项。之后我们再执行另一个非线性操作(例如，σ)。

在公式 1 中，x 和 F 的维度必须相同。如果不相同（例如，当改变了输入/输出的通道），我们可以通过 shortcut 连接执行一个线性映射 Ws 来匹配两者的维度（Eq.2）：

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}. \quad (2)$$

但恒等映射已足够解决退化问题，并且是经济的，因此 Ws 只是用来解决维度不匹配的问题。

**网络结构：**

在传统网络的基础上，我们插入 shortcut 连接，将网络变成了对应的残差版本。如果输入和输出的维度相同时，可以直接使用恒等 shortcuts（Eq.1）（实线部分）。当维度增加时（Fig.3 中的虚线部分），考虑两个选项：

(A) shortcut 仍然使用恒等映射，在增加的维度上使用 0 来填充，这样做不会增加额外的参数；

(B) 使用 Eq.2 的映射 shortcut 来使维度保持一致（通过 1*1 的卷积）。

对于这两个选项，当 shortcut 跨越两种尺寸的特征图时，均使用 stride 为 2 的卷积。

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| conv2_x | 56×56 | 3×3 max pool, stride 2 | | | | |
| | | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

**结论：**

（1） 残差网络在加深深度的同时，可以解决训练精度下降的问题；

（2） 三种 shortcuts 结构中，

(A) 对增加的维度使用 0 填充，所有的 shortcuts 是无参数的；

(B) 对增加的维度使用映射 shortcuts，其它使用恒等 shortcuts；

(C) 所有的都是映射 shortcuts。

在 A、B、C 三个结果中细小的差距也表明了映射 shortcuts 对于解决退化问题并不是必需的。

（3）1202 层模型的测试结果比 110 层的结果要差，尽管它们的训练错误率差不多。是过拟合导致的。应用强大的正则化方法，如 maxout 或者 dropout，将会获得最好的结果。

### 5) Deep Learning Face Representation from Predicting 10,000 Classes（Deep ID1）

**主要思想：**

（1）Deep ID1 通过深度学习学习到一组高级特征表示集合用于人脸验证（对比两个人脸确认是否是同一个人）。

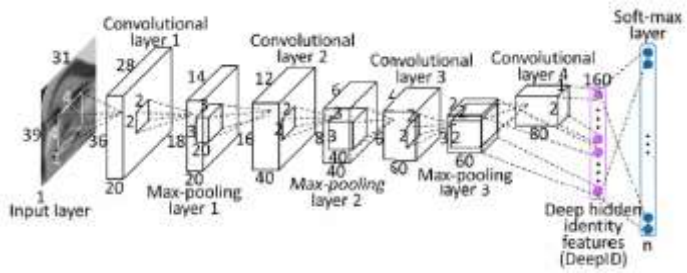
（2）用多分类代替二分类。

（3）关于网络。沿着特征抽取的层次，逐渐在高层形成紧致的和只有少量隐藏神经元的特征。从多个人脸区域中抽取特征，形成相互补充和超完备的表示。

（4）DeepID 和其他的人脸分类器（如 Joint Bayesian)相整合。

**主要步骤：**

（1）卷积网络架构




一共 4 个卷积层，其中前三个卷积层后都有一个 max-pooling layer。DeepID 层（160 维）同时全连接到最后一个卷积层和第三个卷积层（池化后），这样就可以同时学到高层特征和中层特征。

第三层卷积的神经单元的参数在 2*2 的局部区域内共享；第四层卷积则是全连接，参数在神经单元之间不共享。

最后的 soft-max 层（全连接）是学习类别分布的，若类别数是 10000，则该层维度是 10000。

其中，卷积公式为（采用 ReLU 激活器）：

$$y^{j(r)} = \max\left(0,\ b^{j(r)} + \sum_i k^{ij(r)} * x^{i(r)}\right), \quad (1)$$

池化公式为：

$$y_{j,k}^{i} = \max_{0 \le m,n < s} \left\{ x_{j \cdot s+m,\, k \cdot s+n}^{i} \right\}. \quad (2)$$

Deep ID 层公式为：

$$y_j = \max\left(0, \sum_i x_i^1 \cdot w_{i,j}^1 + \sum_i x_i^2 \cdot w_{i,j}^2 + b_j\right). \quad (3)$$

Softmax 输出：

$$y_i = \frac{\exp(y_i')}{\sum_{j=1}^{n} \exp(y_j')}. \quad (4)$$

（2）特征提取

通过 5 个 landmarks 对齐人脸，从人脸图像中选取 10 个区域，3 个尺度，RGB 和 gray 图像块共 60 个块，每个块及其水平翻转抽取 160 维的特征。这样整个 DeepID 的长度是 160*60*2。

（3）人脸验证

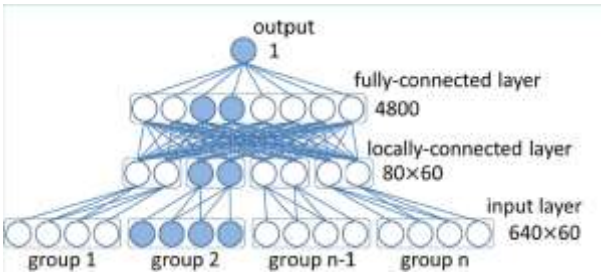
分别采用 Joint Bayesian 和构建深度网络分别进行。实验表明 Joint Bayesian 较好。

- Joint Bayesian

通过计算对数似然比 ratio 判断是否为同一人脸（具体可见论文 Bayesian Face Revisited A Joint Formulation）

$$r(x_1, x_2) = \log \frac{P(x_1, x_2 \mid H_I)}{P(x_1, x_2 \mid H_E)}. \quad (8)$$

- 深度网络




其中隐藏层采用 ReLU 函数，而输出层采用 sigmoid 函数。

### 6) Deeplearningfacerepresentationbyjointidentificationverification（Deep ID2）

**主要思想：**

同时使用 face identification 和 verification 信号进行监督学习。face identification 增大类间的变化，face verification 减少类内变化。在 LFW 数据库上得到了 99.15%人脸确认率。

原本的卷积神经网络最后一层 softmax 使用的是 Logistic Regression 作为最终的目标函数，也就是识别信号；但在 DeepID2 中，目标函数上添加了验证信号，两个信号使用加权的方式进行了组合。

**主要步骤：**

（1）特征提取。使用 SDM 算法抽取人脸上的 21 个标记对齐人脸。通过变化位置、尺度、颜色通道，得到 200 个 face patch，对每个 face patch，使用该 patch 及其水平反转的图像进行（400）特征学习。所以，一共需要 200 个深度卷积神经网络。

（2）学习特征。构建 200 个 DeepID2 来学习上一步得到的 patch。每个 DeepID2 都将输入图像表示成一个 160 维的向量。

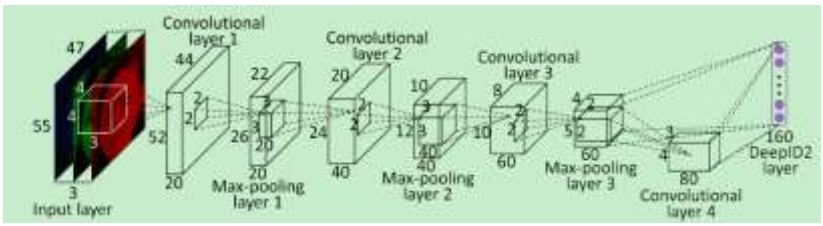
（3）使用前向后向贪心算法来选取一些有效且互补的 DeepID2 向量。

（4）选中 25 个向量后，每张图像的维度是 25*160=4000 维。使用 PCA 进行降维，降维后大约有 180 维。

（5）对于输出后的向量，联合贝叶斯模型来进行分类。

其中：

（1）卷积网络架构




因为添加了类间差距，所以最终层不能再成为是 softmax 层。

（2）两种监督信号

识别信号公式为：

$$\text{Ident}(f, t, \theta_{id}) = -\sum_{i=1}^{n} p_i \log \hat{p}_i = -\log \hat{p}_t \,. \tag{1}$$

验证信号公式为：

- L2 范数

$$\text{Verif}(f_i, f_j, y_{ij}, \theta_{ve}) = \begin{cases} \frac{1}{2}\|f_i - f_j\|_2^2 & \text{if } y_{ij} = 1 \\ \frac{1}{2}\max\left(0, m - \|f_i - f_j\|_2\right)^2 & \text{if } y_{ij} = -1 \end{cases}, \tag{2}$$

当两个样本相同时，则需要最小化它们之间的距离；当两个样本不同时，则需要最小化 m 与它们的距离值之差，m 是一个需要手动调整的参数。

- 余弦值

$$\text{Verif}(f_i, f_j, y_{ij}, \theta_{ve}) = \frac{1}{2}\left(y_{ij} - \sigma(wd + b)\right)^2 \,. \tag{3}$$

在最终组合目标函数时，将 Ident 与 Verif 加权。

（3）训练过程

**input**: training set $\chi = \{(x_i, l_i)\}$, initialized parameters $\theta_c$, $\theta_{id}$, and $\theta_{ve}$, hyperparameter $\lambda$, learning rate $\eta(t)$, $t \leftarrow 0$

**while** not converge **do**

$t \leftarrow t + 1$ sample two training samples $(x_i, l_i)$ and $(x_j, l_j)$ from $\chi$

$f_i = \text{Conv}(x_i, \theta_c)$ and $f_j = \text{Conv}(x_j, \theta_c)$

$\nabla\theta_{id} = \frac{\partial \text{Ident}(f_i, l_i, \theta_{id})}{\partial \theta_{id}} + \frac{\partial \text{Ident}(f_j, l_j, \theta_{id})}{\partial \theta_{id}}$

$\nabla\theta_{ve} = \lambda \cdot \frac{\partial \text{Verif}(f_i, f_j, y_{ij}, \theta_{ve})}{\partial \theta_{ve}}$, where $y_{ij} = 1$ if $l_i = l_j$, and $y_{ij} = -1$ otherwise.

$\nabla f_i = \frac{\partial \text{Ident}(f_i, l_i, \theta_{id})}{\partial f_i} + \lambda \cdot \frac{\partial \text{Verif}(f_i, f_j, y_{ij}, \theta_{ve})}{\partial f_i}$

$\nabla f_j = \frac{\partial \text{Ident}(f_j, l_j, \theta_{id})}{\partial f_j} + \lambda \cdot \frac{\partial \text{Verif}(f_i, f_j, y_{ij}, \theta_{ve})}{\partial f_j}$

$\nabla\theta_c = \nabla f_i \cdot \frac{\partial \text{Conv}(x_i, \theta_c)}{\partial \theta_c} + \nabla f_j \cdot \frac{\partial \text{Conv}(x_j, \theta_c)}{\partial \theta_c}$

update $\theta_{id} = \theta_{id} - \eta(t) \cdot \nabla\theta_{id}$, $\theta_{ve} = \theta_{ve} - \eta(t) \cdot \nabla\theta_{ve}$, and $\theta_c = \theta_c - \eta(t) \cdot \nabla\theta_c$

**end while**

**output** $\theta_c$

## 7) Bayesian Face Revisited: A Joint Formulation

（1）隐变量。一个人脸由两部分组成：μ 用来区分不同人，ε 是同一个人在不同姿态下的差异。

$$x = \mu + \varepsilon, \tag{2}$$

这两个潜在变量 μ 和 ε 分布服从两个高斯分布：N(0, Sμ)和 N(0, Sε)。

（2）用 *x1*, *x2* 分别表示两张图片，**Hi** 表示这两张图片为 intra-personal（同一个人），用 **He** 表示 extra-personal（不同人）。P(x1, x2|Hi) 和 P(x1, x2|He)分布的协方差矩阵为：

$$\Sigma_I = \begin{bmatrix} S_\mu + S_\varepsilon & S_\mu \\ S_\mu & S_\mu + S_\varepsilon \end{bmatrix} \Sigma_E = \begin{bmatrix} S_\mu + S_\varepsilon & 0 \\ 0 & S_\mu + S_\varepsilon \end{bmatrix}$$

（3）计算对数似然比 **r**(x1,x2)。，ratio 值很大，则判断为同一个人； ratio 值很小，则判断为不同人。

$$r(x_1, x_2) = \log \frac{P(x_1, x_2|H_I)}{P(x_1, x_2|H_E)} = x_1^T A x_1 + x_2^T A x_2 - 2x_1^T G x_2, \quad (4)$$

where

$$A = (S_\mu + S_\varepsilon)^{-1} - (F + G), \quad (5)$$

$$\begin{pmatrix} F+G & G \\ G & F+G \end{pmatrix} = \begin{pmatrix} S_\mu + S_\varepsilon & S_\mu \\ S_\mu & S_\mu + S_\varepsilon \end{pmatrix}^{-1}. \quad (6)$$

（4）构建一个类似 EM 算法估计两个协方差矩阵 Sμ 和 Sε 。

S-step:

h = [μ；ε1; ……; εm] and x = [x1; ……; xm]之间的关系为：

$$\mathbf{x} = \mathbf{P}\mathbf{h}, \quad \text{where } \mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \end{bmatrix}. \quad (7)$$

给定 x，则隐变量的期望值为：

$$E(\mathbf{h}|\mathbf{x}) = \Sigma_h \mathbf{P}^T \Sigma_x^{-1} \mathbf{x}.$$

M-step:

更新参数值Θ={Sμ，Sε}：

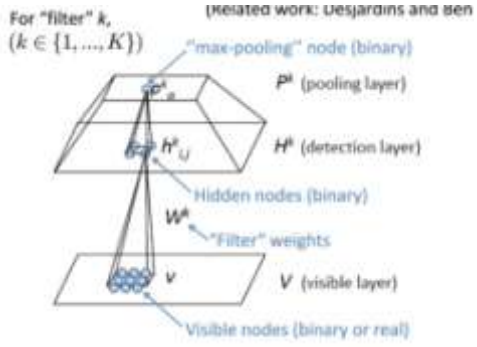
$$S_\mu = \mathbf{cov}(\mu)$$
$$S_\varepsilon = \mathbf{cov}(\varepsilon)$$

这里的 μ 和 ε 是 **E-step** 阶段的结果。不断重复 E-step 跟 M-step 过程，直到 Sμ，Sε 收敛

## 8) Learning hierarchical representations for face Verification with Convolutional Deep Belief Networks

**预备知识：（受限玻尔兹曼机）**

网络由一些可见单元和一些隐藏单元构成，可见变量和隐藏变量都是二元变量，亦即其状态取{0,1}。层间全连接，层内无连接。




RBM 是一种基于能量的模型，其可见变量 v 和隐藏变量 h 的联合能量为：

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{ij} W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j$$

其中 θ 是 RBM 的参数{W, a, b}，W 为可见单元和隐藏单元之间的边的权重，b 和 a 分别为可见单元和隐藏单元的偏置(bias)。

则 v 和 h 的联合概率：

$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\theta)} \exp\big(-E(\mathbf{v}, \mathbf{h}; \theta)\big)$$

其中：$Z(\theta) = \sum_{v,h} e^{-E(v,h|\theta)}$

1） 在给定可见单元的状态时，各隐藏层单元的激活状态之间是条件独立的。此时，第 j 个隐单元的激活概率为：

$$P(h_j = 1|v) = f(b_j + \sum_i v_i W_{ij})$$

2） 当给定隐单元的状态时，可见单元的激活概率同样是条件独立的：

$$P(v_i = 1|h) = f(a_i + \sum_j W_{ij} h_j)$$

最大似然函数为：

$$\ln \mathcal{L}_{\theta,\mathbf{S}} = \ln \prod_{i=1}^{n_s} P(\mathbf{v}^i) = \sum_{i=1}^{n_s} \ln P(\mathbf{v}^i).$$

梯度下降算法：

$$\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial \theta} = \sum_{m=1}^{n_s} \frac{\partial \ln P(\mathbf{v}^m)}{\partial \theta}$$

$$\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial w_{i,j}} = \sum_{m=1}^{n_s} \left[ P(h_i = 1|\mathbf{v}^m) v_j^m - \sum_{\mathbf{v}} P(\mathbf{v}) P(h_i = 1|\mathbf{v}) v_j \right],$$

$$\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial a_i} = \sum_{m=1}^{n_s} \left[ v_i^m - \sum_{\mathbf{v}} P(\mathbf{v}) v_i \right],$$

$$\frac{\partial \ln \mathcal{L}_{\mathbf{S}}}{\partial b_i} = \sum_{m=1}^{n_s} \left[ P(h_i = 1|\mathbf{v}^m) - \sum_{\mathbf{v}} P(\mathbf{v}) P(h_i = 1|\mathbf{v}) \right].$$

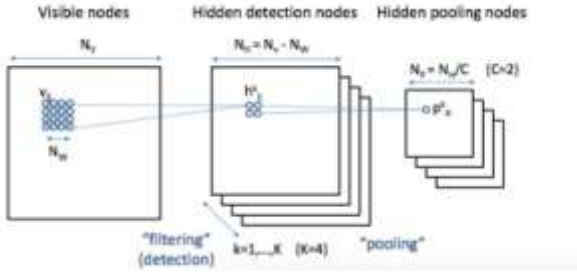
上述公式中关于$\sum_v$的计算复杂度为 O（$2^{n_v+n_h}$），因此通常采用 MCMC 方法如 Gibbs 采样方法进行采样，并用样本对$\sum_v$这个项进行估计。

**主要思想：**

构建了局部卷积限制波尔曼兹机（RBMS），一种新的卷积 RBM 可以适应全局结构，同时可以应用于高分辨率图像，对轻微的偏差具有鲁棒性。

（1）单层 CRBM 网络

输入为 Nv×Nv 的 2D 图像，和 CNN 一样 CRBM 可以设置多个特征滤波器(也称为卷积核)，假设有 K 个大小为 Nw×Nw 的特征滤波器。每个滤波器的计算分为 convolution 和 pooling 两个部分：




①是由可视层到隐层的计算②是隐层到下采样层 pooling 层的计算，这里采用了 max-pooling 方法，即按 pool 的大小 poolsize（如 poolsize=2，pool 区域大小则为 2x2）每个区域选取当中最大值。

CRBM 的能量函数公式如下：

$$\begin{aligned} P(\mathbf{v},\mathbf{h}) &= \frac{1}{Z}\exp(-E(\mathbf{v},\mathbf{h})) \\ E(\mathbf{v},\mathbf{h}) &= -\sum_{k=1}^{K}\sum_{i,j=1}^{N_H}\sum_{r,s=1}^{N_W} h_{ij}^k W_{rs}^k v_{i+r-1,j+s-1} \\ &\quad + \sum_{i,j=1}^{N_V}\frac{1}{2}v_{ij}^2 - \sum_{k=1}^{K} b_k \sum_{i,j=1}^{N_H} h_{ij}^k - c\sum_{i,j=1}^{N_V} v_{ij} \\ \text{s.t.} &\quad \sum_{(i,j)\in B_\alpha} h_{i,j}^k \le 1, \quad \forall k,\alpha \end{aligned} \tag{1}$$

下面的两个公式分别对应 convolution 和 pooling 的计算：

$$P(v_{ij} = 1|\mathbf{h}) = \mathcal{N}((\sum_k W^k *_f h^k)_{ij} + c, 1) \tag{2}$$

$$P(h_{i,j}^k = 1|\mathbf{v}) = \frac{\exp(I(h_{i,j}^k))}{1 + \sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))} \tag{3}$$

$$P(p_\alpha^k = 1|\mathbf{v}) = \frac{\sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))}{1 + \sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))} \tag{4}$$

由两个单层 CRBM 构成一个深度置信网络。

（2）局部 CRBM 网络

将图像分割为许多重叠的区域，每个区域采用独立的权重。由于滤波权重不再是全局共享，局部使用可以避免隐藏的单位非指定地方伪激活。

局部 CRBM 能量函数为：

$$E(\mathbf{v},\mathbf{h}) = -\sum_{l=1}^{L}\sum_{k=1}^{K}\left(\mathbf{v}^{(l)} * \tilde{W}^{(l),k}\right) \odot \mathbf{h}^{(l),k} \qquad (6)$$
$$+ \sum_{i,j=1}^{N_V}\frac{1}{2}(\mathbf{v}_{ij}-c)^2 + \sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{r,s=1}^{N_H} b_k^{(l)}\mathbf{h}_{r,s}^{(l),k}$$

convolution 计算公式为：

$$P(\mathbf{h}_{r,s}^{(l),k}=1|\mathbf{v}^{(l)}) = \sigma((\mathbf{v}^{(l)} * \tilde{W}^{(l),k})_{r,s} + b_k^{(l)}). \qquad (7)$$

## 二、 传统迁移学习

### 1) A Survey on Transfer Learning

#### (1) 记法和定义

domain D 由两部分组成：一个特征空间 X 和一个边缘概率分布 P(X)，且 X={x1,x2,...,xn}∈X。如果两个 domain 不同，则他们有不同的特征空间或边缘概率分布。

给定 domian D={X,P(X)}，一个 task 由两部分组成：一个标签空间 Y 和一个目标预测函数 f(·)（由 T={Y,f(·)}表示）。task 不可被直观观测，但是可以通过训练数据学习得来。task 由 pair{xi,yi}组成，且 xi∈X,yi∈Y。函数 f(x)可用于预测对应标签。从概率学角度看，f(x)也可被写为 P(y|x)。

简化起见，本文中我们只考虑一个源域 DS 一个目标域 DT。更准确点，用 DS={(xS1,yS1)，...,(xSn,ySn)},xSi∈Xs,ySi∈Ys 来表示源域。以文档分类为例，DS 是文档对象向量及对应的 true 或 false 标签的集合。相似地有目标域记法 DT，一般有 0≤nT≪nS。

现在我们给出迁移学习的统一定义：

Definition 1 (Transfer learning)：给定源域 DS 和学习任务 TS，一个目标域 DT 和学习任务 TT，迁移学习致力于用 DS 和 TS 中的知识，帮助提高 DT 中目标预测函数 fT(·)的学习。并且有 DS≠DT 或 TS≠TT。

在上面定义中,D={X,P(X)},DS≠DT 意味着源域和目标域实例不同(XS≠XT)或者源域和目标域边缘概率分布不同(PS(X)≠PT(X))。同理 T={Y,P(Y|X)}，TS≠TT 意味着源域和目标域标签不同（YS≠YT）或者源域和目标域条件概率分布不同（P(YS|XS)≠P(YT|XT)）。当源域和目标域相同且源任务和目标任务相同，则学习问题是一个传统机器学习问题。

以文档分类为例，域不同有以下两种情况：

1. 特征空间不同，即 DS≠DT。可能是文档的语言不同。

2. 特征空间相同但边缘分布不同，即 P(XS)≠P(XT)。可能是文档主题不同。

给定域 DS 和 DT，学习任务不同可能有以下两种情况：

1. 域间标签空间不同，即 YS≠YT。可能是源域中文档需要分两类，目标域需要分十类。

2. 域间条件概率分布不同，即 P(YS|XS)≠P(YT|XT)。

除此之外，当两个域或者特征空间之间无论显式或隐式地存在某种关系时，我们说源域和目标域相关。

#### (2) 迁移学习分类

迁移学习主要有以下三个研究问题：1）迁移什么，2）如何迁移，3）何时迁移。

1. inductive transfer learning：目标任务和源任务不同，无论目标域与源域是否相同。

- 源域中大量已标注数据可用，和 multitask learning 类似。
- 源域中无已标注数据可用，和 self-taught learning 相似。

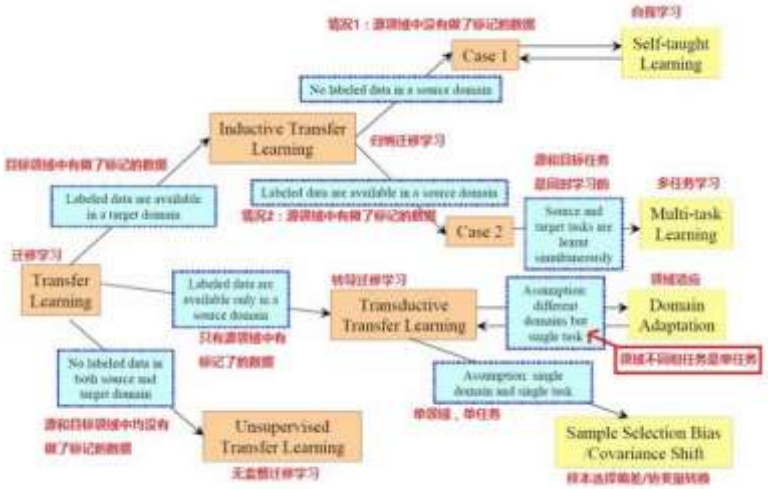
2. transductive transfer learning：源任务和目标任务相同，源域和目标域不同。目标域中无已标注数据可用，源域中有大量已标注数据可用。

- 源域和目标域中的特征空间不同，即 XS≠XT;
- 源域和目标域间的特征空间相同，XS=XT，但输入数据的边缘概率分布不同，即 P(XS)≠P(XT)。

3. unsupervised transfer learning：目标任务与源任务不同但相关。专注于解决目标域中的无监督学习问题，如聚类、降维、密度估计。训练中源域和目标域都无已标注数据可用。

| 学习方法 | | 源和目标领域 | 源和目标任务 |
|---|---|---|---|
| 传统机器学习 | | 相同 | 相同 |
| 迁移学习 | 归纳迁移学习/<br>无监督迁移学习 | 相同 | 不同但有关联 |
| | | 不同但有关联 | 不同但有关联 |
| | 转导迁移学习 | 不同但有关联 | 相同 |

| 迁移学习方法 | 关联领域 | 源资源标记 | 目标领域标记 | 任务 |
|---|---|---|---|---|
| 归纳迁移学习 | 多任务学习 | 有 | 有 | 回归，分类 |
| | 自我学习 | 无 | 有 | 回归，分类 |
| 转导迁移学习 | 领域适应，样本选择偏差，协变量转换 | 有 | 无 | 回归，分类 |
| 无监督迁移学习 | | 无 | 无 | 聚类，降维 |




上述三种迁移学习可以基于“迁移什么”被分为四种情况：

**迁移学习的不同的实现方法**

| 迁移学习实现方法 | 简要描述 |
|---|---|
| 立即迁移 | 调整源领域中一些有标记数据的权重以便在目标领域中使用 |
| 代表特征迁移 | 找到一个“足够好”的基本能同时适用于源领域和目标领域特征并减少分类和回归模型的错误 |
| 参数迁移 | 发现一个源领域和目标领域中相同或同样优先的，对迁移学习有好处的参数 |
| 相关知识迁移 | 建立源领域和目标领域间的相关知识地图，不仅领域之间是相关的，Ⅱ d中的假设在每个领域中也是满足的 |

**不同迁移学习实现方法用到的迁移学习原理**

| | 归纳迁移学习 | 转导迁移学习 | 无监督迁移学习 |
|---|---|---|---|
| 立即迁移 | √ | √ | |
| 代表特征迁移 | √ | √ | √ |
| 参数迁移 | √ | | |
| 相关知识迁移 | √ | | |

**（3） 具体例子**

1） INDUCTIVE TRANSFER LEARNING

- 基于实例迁移

AdaBoost 算法，具体详见 Boosting for Transfer Learning；

- 基于特征迁移

a） 有监督特征：

与 multitask learning 中使用的方法类似。基本想法是去构建一个可以跨相关任务的低维表示，可以通过一个优化问题来学习公共特征。

$$\arg\min_{A,U} \sum_{t\in\{T,S\}}\sum_{i=1}^{n_t} L(y_{t_i},\langle a_t, U^T x_{t_i}\rangle) + \gamma\|A\|_{2,1}^2 \quad (1)$$
$$s.t.\quad U\in \mathbf{O}^d.$$

S 和 T 表示源域和目标域中的任务，A=[aS,aT]∈Rd×2 是参数矩阵。U 是一个 d×d 的映射矩阵用于将高维数据映射成为低维表示。

$\|A\|_{r,p} = (\sum_{i=1}^{d} \|a^i\|_r^p)^p$。上式表达的优化问题同时估计了低维表示$U^T X_T, U^T X_S$和模型的参数 A，上式也可被等效转化为凸优化函数并被高效地解决。

b) 无监督特征：

以应用稀疏编码，在迁移学习中学习高维特征。由两部构成：第一步，通过在源域数据上求解(2)式得到更高层的偏置向量 b={b1,b2,...,bS}：

$$\min_{a,b} \sum_i \left\| x_{S_i} - \sum_j a^j_{S_i} b_j \right\|_2^2 + \beta \|a_{S_i}\|_1 \quad (2)$$
$$s.t. \quad \|b_j\|_2 \le 1, \quad \forall j \in 1,\ldots,s.$$

得到偏置向量 b 之后，第二步在目标域数据上应用(3)式以学习基于偏置向量 b 的更高维特征：

$$a^*_{T_i} = \arg\min_{a_{T_i}} \left\| x_{T_i} - \sum_j a^j_{T_i} b_j \right\|_2^2 + \beta \|a_{T_i}\|_1. \quad (3)$$

最后，目标域上，判别算法被应用到$a^*_{T_i}$和对应的标签以训练分类和回归模型。

- 基于参数迁移

基于 SVM，构建传递参数的正则化框架，假定每个任务的支持向量机中的参数 w 可分为两个项，一个是共同任务，另一个是特定任务。

$$w_S = w_0 + v_S \ \ and \ \ w_T = w_0 + v_T,$$

$$\begin{aligned} \min_{w_0, v_t, \xi_{t_i}} \quad & J(w_0, v_t, \xi_{t_i}) \\ = \quad & \sum_{t\in\{S,T\}} \sum_{i=1}^{n_t} \xi_{t_i} + \frac{\lambda_1}{2} \sum_{t\in\{S,T\}} \|v_t\|^2 + \lambda_2 \|w_0\|^2 \quad (4) \\ s.t. \quad & y_{t_i}(w_0 + v_t) \cdot x_{t_i} \ge 1 - \xi_{t_i}, \\ & \xi_{t_i} \ge 0, \ i \in \{1,2,\ldots,n_t\} \ and \ t \in \{S,T\}. \end{aligned}$$

通过优化上述公式，可求得参数$\omega_0$、$v_t$、$v_s$。

- 基于相关知识迁移

2) TRANSDUCTIVE TRANSFER LEARNING

- 基于实例迁移

通过降低 EMR，来优化参数θ

$$\begin{aligned} \theta^* &= \arg\min_{\theta\in\Theta} \sum_{(x,y)\in D_S} \frac{P(D_T)}{P(D_S)} P(D_S) l(x, y, \theta) \\ &\approx \arg\min_{\theta\in\Theta} \sum_{i=1}^{n_{S_e}} \frac{P_T(x_{T_i}, y_{T_i})}{P_S(x_{S_i}, y_{S_i})} l(x_{S_i}, y_{S_i}, \theta), \end{aligned} \quad (5)$$

## 2) Boosting for Transfer Learning

AdaBoost 算法中，起初给训练数据 T 中的每一个样例都赋予一个权重，当一个源域中的样本被错误的分类之后，我们认为这个样本是很难分类的，于是乎可以加大这个样本的权重，这样在下一次的训练中这个样本所占的比重就更大了，这一点和基本的 AdaBoost 算法的思想是一样的。如果辅助数据集中的一个样本被错误的分类了，我们认为这个样本对于目标数据是很不同的，我们就降低这个数据在样本中所占的权重，降低这个样本在分类器中所占的比重，下面给出 TradaBoost 算法的具体流程：

**算法 1** TrAdaBoost 算法描述

**输入** 两个训练数据集$T_a$和$T_b$（根据公式(3.1)，合并的训练数据集$T = T_a \cup T_b$），一个未标注的测试数据集$S$，一个基本分类算法**Learner**，和迭代次数$N$。

**初始化**

1. 初始权重向量$\mathbf{w}^1 = (w_1^1, \ldots, w_{n+m}^1)^\tau$。其中，

$$w_i^1 = \begin{cases} 1/n & \text{当} i = 1, \ldots, n \\ 1/m & \text{当} i = n+1, \ldots, n+m \end{cases}$$

2. 设置$\beta = 1/(1 + \sqrt{2\ln n/N})$。

**For** $t = 1, \ldots, N$

1. 设置$\mathbf{p}^t$满足

$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^{n+m} w_i^t}.$$

2. 调用**Learner**，根据合并后的训练数据$T$以及$T$上的权重分布$\mathbf{p}^t$和未标注数据$S$，得到一个在$S$的分类器$h_t : X \mapsto Y$。

3. 计算$h_t$在$T_b$上的错误率$\varepsilon_t$

$$\varepsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t |h_t(x_i) - c(x_i)|}{\sum_{i=n+1}^{n+m} w_i^t}.$$

4. 设置$\beta_t = \varepsilon_t/(1-\varepsilon_t)$。

5. 设置新的权重向量如下

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i) - c(x_i)|}, & \text{当} i = 1, \ldots, n \\ w_i^t \beta_t^{-|h_t(x_i) - c(x_i)|}, & \text{当} i = n+1, \ldots, n+m \end{cases}$$

**输出** 最终分类器

$$h_f(x) = \begin{cases} 1, & \prod_{t=\lceil N/2 \rceil}^{N} \ln(1/\beta_t) h_t(x) \geq \frac{1}{2} \sum_{t=\lceil N/2 \rceil}^{N} \ln(1/\beta_t) \\ 0, & \text{其他} \end{cases}$$

可以看到，在每一轮的迭代中，如果一个辅助训练数据被误分类，那么这个数据可能和源训练数据是矛盾的，那么我们就可以降低这个数据的权重。具体来说，就是给数据乘上一个$\beta^{|h_t(x_i)-c(x_i)|}$，其中β的值在 0 到 1 之间，所以在下一轮的迭代中，被误分类的样本就会比上一轮少影响分类模型一些，在若干次以后，辅助数据中符合源数据的那些数据会拥有更高的权重，而那些不符合源数据的权重会降低。极端的一个情况就是，辅助数据被全部忽略，训练数据就是源数据 Tb，这样这时候的算法就成了 AdaBoost 算法了。在计算错误率的时候，当计算得到的错误率大于 0.5 的话，需要将其重置为 0.5。

### 3) A Practical Transfer Learning Algorithm for Face Verification

**预备知识：（KL 散度）**

相对熵又称为 KL 散度、信息散度、信息增益，是两个概率分布 P 和 Q 差别的非对称性的度量。 对一个离散随机变量的两个概率分布 P 和 Q 来说，他们的 KL 散度定义为：

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

对于连续的随机变量，定义为：

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx.$$

注：

（1）$D_{KL}$(P||Q)是一个数值，而不是一个函数。相对熵的值为非负值，即$D_{KL}$(P||Q)≥0，只有 P(x)=Q(x) 几乎处处成立的时候，Dkl(P||Q)=0.

（2）KL 散度并不满足距离的概念，应为：1）KL 散度不是对称的，即 D(P||Q) ≠ D(Q||P)；2）KL 散度不满足三角不等式。

（3）P 通常指数据集，我们已有的数据集，Q 表示理论结果。

**主要思想：**

（1） 利用 KL 散度约束源领域和目标领域的分布，以最大化信息共享。

（2） 将 KL 散度和似然函数结合起来，并通过类 EM 算法求解。

基于贝叶斯模型，源领域参数为**Θs**={Sμ，Sε}，目标领域标记数据为$\mathcal{X}$，则目标就是学习新领域模型参数为**Θt**={Tμ，Tε}。在缺少源领域数据标记的情况下，通过最大化 logp(**Θt**|$\mathcal{X}$)来获得Θt。同时 KL 散度惩罚偏离源领域的目标领域的数据，将两者结合起来的优化目标如下：

$$\min_{\Theta_t} \quad -\log p(\mathcal{X}|\Theta_t) + \lambda \, \mathrm{KL}(p(\mathcal{X}|\Theta_t)||p(\mathcal{X}|\Theta_s)), \quad (3)$$

由于目标领域的实例都是独立的，所有实例的联合分布优化函数可以写为：

$$\min_{\Theta_t} -\sum_i \log p(X_i|\Theta_t) + \lambda \sum_i \mathrm{KL}(p(X_i|\Theta_t)||p(X_i|\Theta_s)).$$

类 EM 算法：

对于隐变量μ和ε可以将其表达成一个列向量，即 H=（$\mu, \varepsilon_1, \ldots\ldots \varepsilon_m$），而 X=PH。由于μ和ε是独立的零均值高斯变量，则 H 是零均值协方差为Ω的高斯分布。EM 算法的主要步骤为，先迭代计算两个隐变量的期望函数（E-step），然后通过最大化优化函数更新变量（M-step）。

E-step:

$$E(H|X) = \Omega P^T (P\Omega P^T)^{-1} X. \quad (5)$$

M-step: $\Theta t = \{T\mu, T\varepsilon\}$可依据以下公式更新

$$\begin{aligned} T_\mu &= w\, S_\mu + (1-w) n^{-1} \sum_i \mu_i \mu_i^T \\ T_\epsilon &= w\, S_\epsilon + (1-w) k^{-1} \sum_j \epsilon_j \epsilon_j^T, \end{aligned} \quad (6)$$

## 三、 深度迁移学习

### 1. Adversarial Discriminative Domain Adaptation

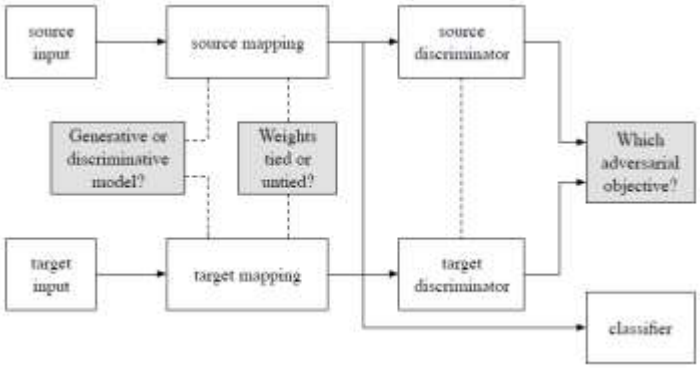
**主要思想：**

文章利用 GAN 网络的思想用于跨领域识别

- 先前的生成网络不适用于大的 domain shift
- 先前的辨别网络施加固定的权重，没有利用 GAN 的 loss

**一般框架：**

最近的 domain adaptation 方案：将两个域的 feature 投影到统一的特征空间。通过最小化一定的约束学习映射。这些方法总结为一些选择：




1） 是否使用生成器 generator

2） 使用什么 loss function

3） 是否权值共享

**生成对抗 adaptation:**

1） 利用源域标签，学习一个源域的映射 Ms 和分类器。

$$\min_{M_s,C} \mathcal{L}_{\text{cls}}(\mathbf{X}_s, Y_t) = \mathbb{E}_{(\mathbf{x}_s,y_s)\sim(\mathbf{X}_s,Y_t)} -\sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log C(M_s(\mathbf{x}_s)) \quad (1)$$

2) 把 Ms 和 Cs 迁移到目标域。其中，

- 分类器共享，即 Ct=Cs，相当于在映射后的子空间内，源域和目标域有着相同的分布。因此，只需要学习 Mt。
- 为获得 Mt，需要定义域分类器 D，借鉴 GAN 网络的思想，最小化 LadvD 可以得到当前的域分类器 D。

$$\begin{aligned}\mathcal{L}_{\text{adv}_D}(\mathbf{X}_s, \mathbf{X}_t, M_s, M_t) = \\ -\mathbb{E}_{\mathbf{x}_s\sim\mathbf{X}_s}[\log D(M_s(\mathbf{x}_s))] \\ -\mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}_t}[\log(1-D(M_t(\mathbf{x}_t)))]\end{aligned} \quad (2)$$

3) 得到 D 后，在 Mt 和 Ms 的限制条件下，通过生成器训练 Xt，希望 D 尽可能区分不出二者，即

$$\begin{aligned}&\min_{D} \ \mathcal{L}_{\text{adv}_D}(\mathbf{X}_s, \mathbf{X}_t, M_s, M_t) \\ &\min_{M_s,M_t} \ \mathcal{L}_{\text{adv}_M}(\mathbf{X}_s, \mathbf{X}_t, D) \\ &\text{s.t.} \ \ \psi(M_s, M_t)\end{aligned} \quad (3)$$

**源域和目标域的映射：**

- 全权值共享来说，Ms=Mt，对于

$$\psi_{\ell_i}(M_s^{\ell_i}, M_t^{\ell_i}) = (M_s^{\ell_i} = M_t^{\ell_i}). \quad (5)$$

- 无权值共享，这两者无需满足任何约束。
- 部分权值共享

**对抗性损失：**

- 原始的 GAN 的思想，容易发生梯度消失：

$$\mathcal{L}_{\text{adv}_M} = -\mathcal{L}_{\text{adv}_D}. \quad (6)$$

- 改进方法是：将 label 反转，最优化目标变为：

$$\mathcal{L}_{\text{adv}_M}(\mathbf{X}_s, \mathbf{X}_t, D) = -\mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}_t}[\log D(M_t(\mathbf{x}_t))]. \quad (7)$$

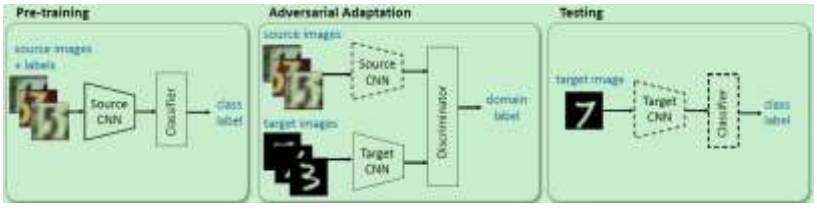
- 如果 Mt 和 Ms 都不固定，容易引发震荡，选择交叉熵：

$$\begin{aligned}\mathcal{L}_{\text{adv}_M}(\mathbf{X}_s, \mathbf{X}_t, D) = \\ -\sum_{d\in\{s,t\}} \mathbb{E}_{\mathbf{x}_d\sim\mathbf{X}_d}\Big[\frac{1}{2}\log D(M_d(\mathbf{x}_d)) \\ +\frac{1}{2}\log(1-D(M_d(\mathbf{x}_d)))\Big].\end{aligned} \quad (8)$$

**提出方法：**

1) 先通过 source domain 的样本和标签训练 Ms 和 C
2) 保持 Ms 和 C 不变，用 Ms 初始化 Mt，并交替优化第二项第三项，获得 D 和 Mt
3) testing stage: target domain 的 sample 直接通过 Mt 和 C 获得预测的标签。

$$\begin{aligned}&\min_{M_s,C} \mathcal{L}_{\text{cls}}(\mathbf{X}_s, Y_s) = \\ &\quad -\mathbb{E}_{(\mathbf{x}_s,y_s)\sim(\mathbf{X}_s,Y_s)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log C(M_s(\mathbf{x}_s)) \\ &\min_{D} \mathcal{L}_{\text{adv}_D}(\mathbf{X}_s, \mathbf{X}_t, M_s, M_t) = \\ &\quad -\mathbb{E}_{\mathbf{x}_s\sim\mathbf{X}_s}[\log D(M_s(\mathbf{x}_s))] \\ &\quad -\mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}_t}[\log(1-D(M_t(\mathbf{x}_t)))] \\ &\min_{M_s,M_t} \mathcal{L}_{\text{adv}_M}(\mathbf{X}_s, \mathbf{X}_t, D) = \\ &\quad -\mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}_t}[\log D(M_t(\mathbf{x}_t))].\end{aligned} \quad (9)$$



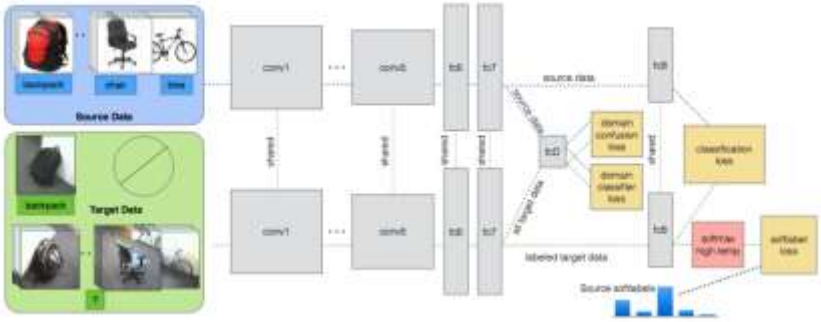
## 2. Simultaneous Deep Transfer Across Domains and Tasks

**主要思想：**

- 通过优化域混淆损失，寻找领域域不变性的特征表示。
- 利用 soft-label，优化目标域的结构，使得目标域中类别间关系与源域中的相同。

**实现途径：**

通过 CNN 学习源域和目标域的特征表示，其中 CNN 包括五个卷积层和三个全连接层，其中五个卷积层和 fc6、fc7 提取特征表示，fc8 输出分类结果。FcD 是域分类器，通过不断优化域混淆损失，调整 CNN 参数得到领域不变的特征表示。而通过 softlabel 损失，不断优化目标域的结构，使得目标域中类别间关系与源域中的相同。



- 分类损失

fc8 中类别分类器损失公式：

$$\mathcal{L}_C(x, y; \theta_{\text{repr}}, \theta_C) = -\sum_k \mathbb{1}[y = k] \log p_k \qquad (1)$$

其中 p 是 softmax 输出的分布函数，$p = \text{softmax}(\theta_C^{\mathrm{T}} f(x; \theta_{repr}))$。

- 域混淆损失

优化域分类器，使得能够区分出源域和目标域的特征：

$$\mathcal{L}_D(x_S, x_T, \theta_{\text{repr}}; \theta_D) = -\sum_d \mathbb{1}[y_D = d] \log q_d \qquad (3)$$

通过优化特征表示，使得域分类器无法分辨出源域和目标域：

$$\mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}}) = -\sum_d \frac{1}{D} \log q_d. \qquad (4)$$

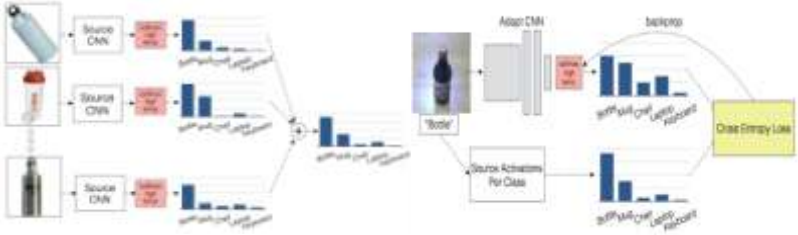
由于两者是互相矛盾的，所以迭代优化以上两个公式：

$$\min_{\theta_D} \mathcal{L}_D(x_S, x_T, \theta_{\text{repr}}; \theta_D) \qquad (5)$$

$$\min_{\theta_{\text{repr}}} \mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}}). \qquad (6)$$

- softlabel 损失

将属于同一类别的样本的 softmax 输出分布，进行平均，得到其他相似样本的概率分布。



计算出源域的 softlabel，同时计算目标领域的 softlabel，计算两者的交叉熵，并根据交叉熵结果调整权重。

$$\mathcal{L}_{\text{soft}}(x_T, y_T; \theta_{\text{repr}}, \theta_C) = -\sum_i l_i^{(y_T)} \log p_i \qquad (7)$$

其中，$p = \text{softmax}(\theta_C^T f(x_t; \theta_{repr})/\tau)$。

联合优化以上三者：

$$\mathcal{L}(x_S, y_S, x_T, y_T, \theta_D; \theta_{\text{repr}}, \theta_C) = \mathcal{L}_C(x_S, y_S, x_T, y_T; \theta_{\text{repr}}, \theta_C) + \lambda \mathcal{L}_{\text{conf}}(x_S, x_T, \theta_D; \theta_{\text{repr}}) + \nu \mathcal{L}_{\text{soft}}(x_T, y_T; \theta_{\text{repr}}, \theta_C). \quad (2)$$

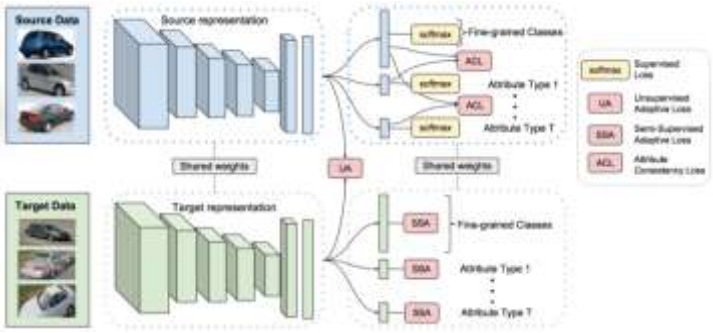
## 3. Fine-grained Recognition in the Wild: A Multi-Task Domain Adaptation Approach

**主要思想：**

- 在细颗粒度类别和属性的基础上，利用多任务适应损失，设计一个多任务适应的方法。（有时，虽然类别便签无法获得，但是属性标签是可以获得的，比如多种不同的汽车品种可能具有相同的体型或相同的制造方式。）
- 设计类别一致性损失，限制细颗粒度类别和属性在分类上的不一致性。
- 采用文献《Simultaneous deep transfer across domains and tasks》中的软标签转移和域混淆的方法进行细颗粒度和属性级别的迁移。

**主要架构：**

- 两个 CNN 神经网络共享的权重提取源图像和目标图像的特征。
- 由独立的 softmax 分类器分类每个属性和细粒度级的图像，分类器输出分类损失。
- 无监督自适应丢失，如域混淆（表示为 UA）都可以用于进一步改进自适应。
- 自适应的损失半监督（简称 SSA）如软标签的损失可以应用在属性以及细粒度的级别。
- 属性一致性损失（简称 ACL）鼓励细粒度和属性分类预测一致的标签。




（1） 分类损失

$$L_{a_n}(x, y; \theta_{rep}, \theta_{a_n}) = -\sum_{a_k=1}^{a_K} \mathbf{1}[y_a = a_k] \log p_{ak} \quad (1)$$

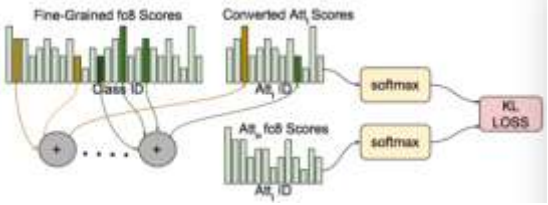
$$L_C(x, y; \theta_{rep}, \theta_C) = -\sum_{k=1}^{K} \mathbf{1}[y = k] \log p_k \quad (2)$$

$$L_{softmax} = \sum_{n=1}^{N_a} \alpha_n L_{a_n} + \alpha_c L_C \quad (3)$$

（1）式是属性分类损失，（2）式是细颗粒度类别分类损失，（3）式是将其加权求和，并作为最后 softmax 最后的分类损失。其中 C 为细颗粒度分类，a 为属性分类。

（2） 属性一致性损失

对于每个属性，先将不同类别的分数（同一属性）跨类别的转化为属性。设 f＝f 1，……k 由 k 类的分数组成。为了获得跨类别的属性分数，我们计算属于同一属性的类的 f 中平均值。然后对属性 a 计算一个 softmax 的分布，p＝$p_{a_1}$,……,$p_{a_k}$。




利用对称 KL 散度计算两者的差值

$$L_{con_{a_n}}(x,\theta_{rep},\theta_{a_n},\theta_c)=\frac{1}{2}D_{KL}(p_n||\hat{p}_n)+\frac{1}{2}D_{KL}(\hat{p}_n||p_n) \tag{4}$$

$$D_{KL}(p_n||\hat{p}_n)=\sum_{a_k=1}^{n_K} p_{nk}\log\frac{p_{nk}}{\hat{p}_{nk}} \tag{5}$$

$$L_{consistency}=\sum_{n=1}^{N_a}\beta_{a_n}L_{con_{a_n}}$$

（3） 自适应损失

使用文献《Simultaneous deep transfer across domains and tasks》中的方法，添加域混淆和 softlabel 损失。softlabel 损失仅用于半监督，且在细颗粒度分类和属性级别上同时使用。

## 4. Distilling the Knowledge in a Neural Network

**主要思想：**

训练阶段大量计算且不需实时响应。使用阶段需考虑计算资源限制、计算速度要求等。

- 大模型学习到的知识可以通过“提取”的方法转移到一个小模型上；
- 使用 softtargets。

使用 soft targets 原因：

通常我们使用 softmax 进行分类的时候，我们的 label 都是 one shot label，比如我们分三类：猫、虎和猪，那么一张猫的图片它的 label 就是[1 0 0]。这种标注方式意味着每一类之间都是独立的，完全没有任何联系。但是事实上，猫和虎的相似度应该高于猫和猪的相似度，这种丰富的结构信息，one shot label(hard target)描述不了。此外，正因为 one shot label 的 hard，导致了我们学习得到的概率分布也相对 hard。显然对于一张猫的图片强行学习一个[1 0 0]的分布，其难度要比学一个[0.65 0.3 0.05]的分布大得多。

**主要步骤：**

- 使用一个高的 T 值和 hard target 训练一个复杂的模型，得到训练样本对应的 soft target；
- 再用得到的 soft target 重新训练一个简单的模型（保持 T 为较高值）
- 将简单模型作为应用模型（应用时 T 值恢复为 1）

（1） soft target

首先自然是需要训练用的比较复杂的模型。文中使用的是一组神经网络，最后接了一个 softmax 层来得到概率输出。如下图所示：

$$q_i=\frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{1}$$

其中，qi 表示第 i 类的概率，zi，zj 表示上一层(logit)的输入，T 表示 temperature，通常为 1。但随着 T 的增大，q 会越来越“soft”。假如我们分三类，然后网络最后的输出是[1.0 2.0 3.0]，传统的 softmax(即 T=1)对此进行处理后得到的概率为[0.09 0.24 0.67]，而当 T=4 的时候，得到的概率则为[0.25 0.33 0.42]，分布更加平稳。

（2） 训练简单模型

有两个 loss，一个是 hard loss,即传统的 softmaxloss，使用 one shot label；另外一个是 soft loss，即 T>1 的 softmaxloss，使用我们第二步保存下来的 soft target label。

$$\mathrm{L}=\alpha L^{soft}+(1-\alpha)L^{hard}$$

其中，$L^{soft}=-\sum_{c=1}^{C}y_c^{soft}log\frac{e^{z_i/T}}{\sum_j e^{z_i/T}}$

由于 soft 项的残差大致乘以了$\frac{1}{T^2}$，因此和 hard 结合时，最好将 soft 项的残差都放大$T^2$倍以此来平衡二者。

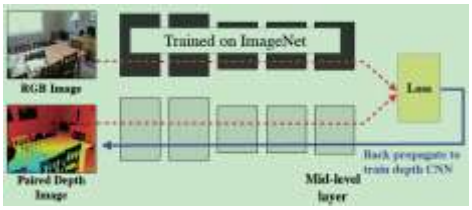
## 5. Cross Modal Distillation for Supervision Transfer

**主要思想：**

- 将一个模态（labeled）中学习到的特征迁移到另外一个模态（unlabeled）中。
- 此方案需要利用到两个模态中的 paired image 以及
- 利用从 labeled 模态中学习到的 mid-level 特征来监督 unlabeled 模态的特征学习。

**主要内容：**

利用已经从 ImageNet dataset 训练好的 CNN，利用其 mid-level 特征作为监督信号，监督 CNN 在 depth images 上的特征学习。




设定下环境，未标注数据的模态记为 U，标注大量样本的模态记为 L，对应的数据集分别记 Du 为 Dl 和，文中使用 CNN 来提取特征，#l 对应着层，图像的特征可记为$\Phi = \{\phi^i, \forall i \in \{1, \ldots\ldots, \#l\}$。假设我们希望能够从模态 U 中学习到一个较好的特征，再假设我们已经有了一个设计好的比较合适的模型$\Psi = \{\psi^i, \forall i \in \{1, \ldots\ldots, \#u\}$，此时任务就成了在不包含标注数据的模态 U 中学习相应的 CNN 参数了，需要学习的参数记为$W_u^{\{1,\ldots\ldots,\#u\}} = \{W_u^i, \forall i \in \{1, \ldots\ldots, \#u\}$。

合并有标注的模态 L 和未标注的模态 U，通过 paired image，可以得到在同一个场景下不同模态的图片数据集。训练模态 U 下的图片特征就是学习特征让其 match 模态 L 中 paired image 的某个中间层特征。文章采用的 L2 距离来衡量这种 match 关系。具体的 loss 见下面的公式：

$$\min_{W_u^{\{1,\ldots\ldots,\#u\}}} \sum_{(I_l, I_u) \in P_{l,u}} f\left(t\left(\psi^{\#u}(I_u)\right), \phi^{i^*}(I_l)\right) \qquad (2)$$

**注：**

**Hinton 等人提出的蒸馏方法是本模型的一个特例，其中一个）**

- **两模式 L 和 U 都是相同的**
- **监督转移发生在最后预测层，而不是在表示任意内部层。**

## 6. Cross-Modal Adaptation for RGB-D Detection

**主要思想：**

- CNN 底层网络属于类别无关，领域特定；顶层网络属于领域无关，类别特定。
- 融合深度图像和 RGB 图像的中层表示，可以使用 RGB-only 分类提高 RGB 检测器的质量。

**主要内容：**

通过给定标记的、一部分类别的深度图像，需要训练 RGB 对象检测器识别一个新的类别，使它可以在测试时，使用深度图像和 RGB 图像，产生更准确的检测。

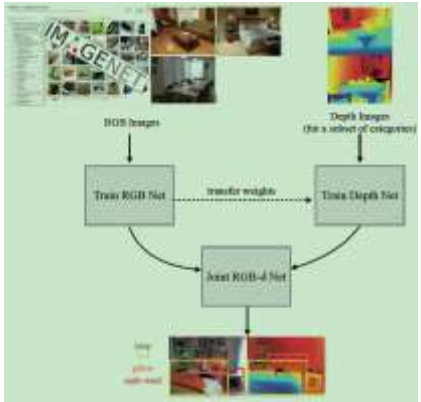
通过跨模式融合信息，并使用现有的标记深度数据提取中层深度表示，以提高识别性能。

最先进的目标检测模型遵循两阶段：

1）计算region proposals：与图像的物体有高度重叠的boundingboxes。

2）region proposals评分：通过使用神经网络得到。

- 对于计算region proposals 阶段，使用RGB-D MCG中的Edge Boxes来适应深度图像。
- 对于region proposals评分，通过训练具有不完备训练数据的一个领域模型，在此基础上获得一个多领域的CNN模型。



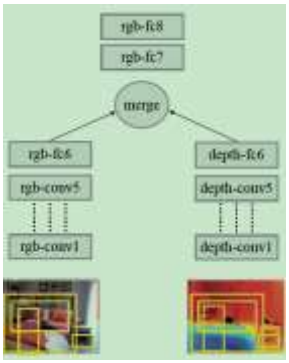

（1） 训练 RGB 网络（AlexNet 架构），使用所有标记的 RGB 数据。

（2） 深度图像产生一个相同的结构，采用深度输入的形式（HHA 编码）。深度图像 CNN 采用 RGB 网络初始化参数。

（3） 调整深度图像参数，使他更好的适应新模型。

（4） 微调 RGB 和深度图像模型之后，构建多领域模型。

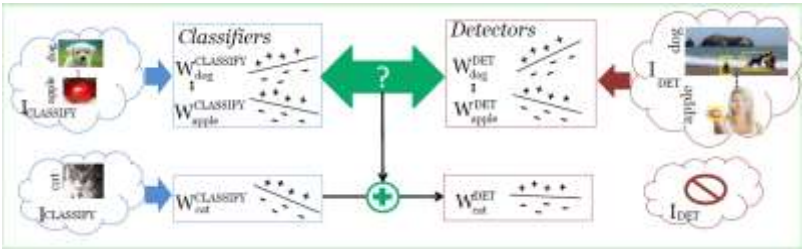
对于合并点之前的层，我们将 RGB 和深度网络的权重直接传递给体系结构的相应权重。对于合并点以上的所有层权重，我们使用 RGB 模型权重。




## 7. LSDA: Large Scale Detection through Adaptation

**主要思想：**

- 将一个 image classifier 转换成一个 object detector
- Image-level annotation 比较容易得到，object-level annotation 不太容易得到。对于一些类，具有 both image & object level annotation；另一些类，只有 image level annotation。
- 通过 CNN 将一个 image classifier 转换成一个 object detector
- transfer 前者的 knowledge 来对没有 object level annotation 的那些类，训练得到 object detector。




source domain：用来训练 classifier 的数据(image level annotation 的图像);target domain：用来训练 detector 的数据(both image & object level annotation 的图像)

**主要内容：**

a).对于所有 K 个类（有 image level annotations，A+B），学习 CNN classifiers；（A 只有 image level annotation，B 有 both）

- 其中，pre-training 用的是 ILSVRC 2012 的 classification 数据集（包含 120 万张图像，1000 个不同的类）。然后保留第 1-7 层，去掉了最后一层（soft-max 层），接一个新的 multinomial logistic regression 层，来对特定的这 K 个类进行识别。

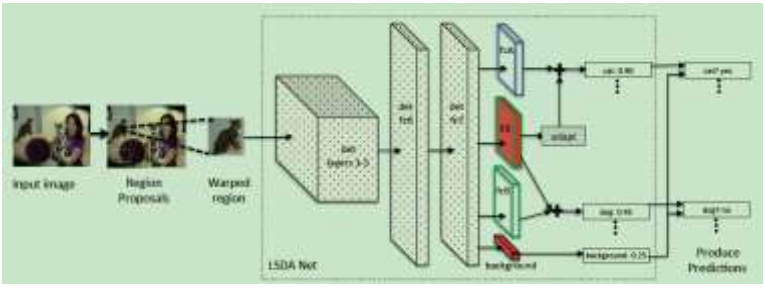
b).在 B 上 m 个类做 Fine-tuning（用 fine-tune RCNN 相同的方法，需要 B 的 bounding box），得到适用于 detection 的 network（加了一个背景类，图中 b 的红色部分），也就是类似 m 个 detectors；(detection adaptation layers，后来试验证明这一步对 mAP 的提高最大)

c). 把没有 object level annotation 的 A (K-m 个类)一起再输入这个 network,得到 adapted network。(output layer adaptation，实验证明，这一步可以提高 mAP，但是效果不是太明显)

- 由于每一类的权重变化是不一样的,对于 A 中的某个类,选出 B 中跟它最近的 k 个邻居类(fc7 层特征的最小欧氏距离,L2-norm)，计算这 k 个类的平均变化，把这个平均变化当做 A 的变化（加在 A 上）。

$W_i^d = W_i^c + \delta B_{i^*}$（对于 B 的变化权重）

$$\forall j \in A: W_j^d = W_j^c + \frac{1}{k}\sum_{i=1}^{k} \delta B_{N_B(j,i)} \quad (1)$$



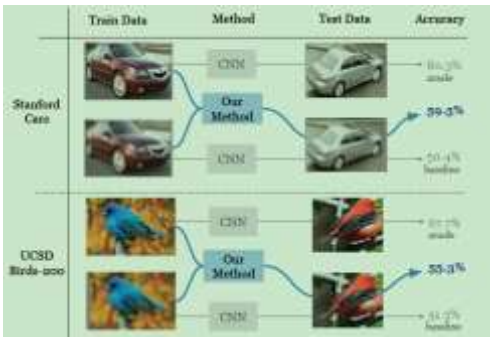

用自适应模型来做检测

给出一幅测试图像，先用 Selective Search 来生成 region proposals，和 R-CNN 一样 pad & warp 每个 region 成指定大小，输入 network。把每个 region 前向传入调整好的网络，得到 K+1 维的特征向量（target domain，也就是 detection 有 K 类），其中 K 个目标类，1 个是背景类。最后，对于每个类 i，作者计算一个 discriminative score。也就是说，对于每张测试图像，最后的输出是 K 个 score。

## 8. Fine-To-Coarse Knowledge Transfer For Low-Res Image Classification

**主要思想：**

对高分辨率数据进行训练，学习具有判别力的中层特征表示。然后，人为地降低了训练数据与测试域的分辨率，进行微调，得出适应目标分辨率的、具有鉴别力的特征。



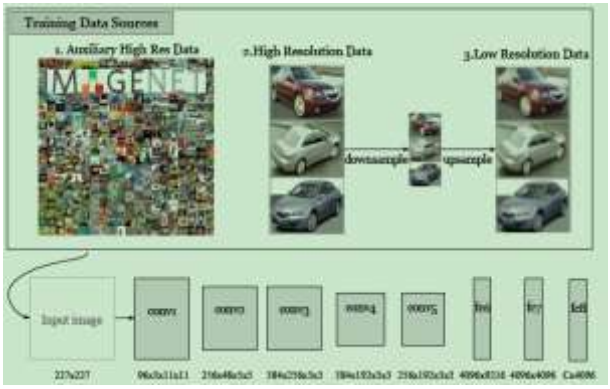

（1） 在一个较大的辅助数据集上训练以初始化模型，在一个大的辅助数据集训练

（2） 使用高分辨率的细粒度的分类训练数据，对模型进行继续微调（较低的学习率）

- 然后保留第 1-7 层，改变最后一层 1000 分类，来对特定的这 K 个类进行识别。并用标准化的高斯分布来初始化。

（3） 人为降低训练图像的分辨率，再度对模型进行微调。

- 将 227*227 的分辨率降低到 50*50




## 9. Learning and Transferring Mid-Level Image Representationsusing Convolutional Neural Networks

**主要思想：**

- 在 imagenet 等大数据集上面训练传统的 CNN 模型
- 共享 CNN 某些层的 weights
- 删除了 softmax 层，加上了两层自适应层

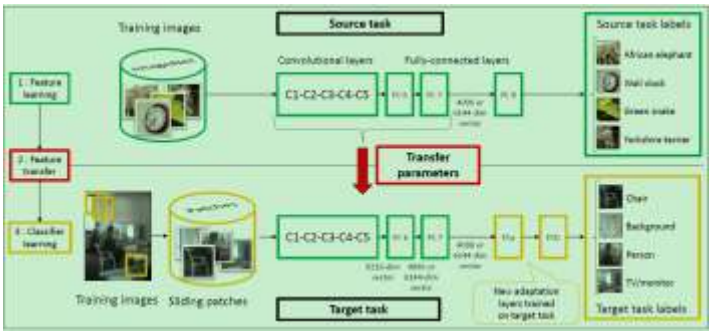
**主要方法：**

（1） 源域和目标域

预训练的数据集和特定任务的数据集的图像有很大的差异，如物体的种类、角度、图像的成像条件等，源域 ImageNet 的数据集图像中只包含一个对象，且基本占据图像中心位置，但目标域 Pascal VOC 的数据集图像中可能包含多个对象，且背景成分较多。


（2） 网络框架

- 在 imagenet 上面采用传统的 CNN 框架进行预训练
- 去掉最后一层 softmax 层，加上 FCa 和 FCb 两层自适应层
- 固定预训练模型的前面 7 层的参数不变，只训练自适应层的参数



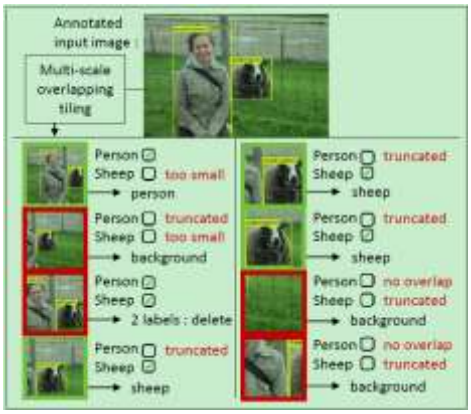

（3） 网络训练（准备训练样本）

- 采用 sliding window 的方法每张图像提取 500 个正方形图像块，每个块之间的重合比例至少为 50%；
- 给图像块打标签，假设图像块为 P，某一类正样本为 Bo 标为相应类别的正样本条件：

（A）P 和 Bo 的交集大于等于 P 的面积的 0.2 倍

（B）P 和 Bo 的交集大于等于 Bo 的面积的 0.6 倍

（C）P 中包含不多于一个物体




（4） 处理背景图像

通常这样得到的样本会导致训练样本不平衡的问题，大多数的图像块是背景图像，（可以采用重新改变损失函数的权重），本文采用随机选择背景图像的 10%；

（5） 分类

由于每一张图片由大概 500 个 Patch，所以采用下面的公式进行分类，得到每张图片具体的 20 个类别中的那 1 个类别。公式如下：

$$\text{score}(C_n) = \frac{1}{M}\sum_{i=1}^{M} y(C_n|P_i)^k, \qquad (1)$$

## 10. How transferable are features in deep neuralnetworks?

**主要思想：**

- 实验了深度神经网络中不同层神经元的泛化性能和特异性，

- 模型的迁移能力主要受到两个因素的影响：

  1）深度模型在越深的层，其专业性越强，即越只能完成特定任务，这使得深度模型学习到的深层特征迁移性很差；

  2）模型在优化的过程中，层与层之间的参数的优化是有关联性，当固定浅层的权值，来训练高层权值时，会打破这种关联性，使得模型的能力变差，泛化能力也变差。

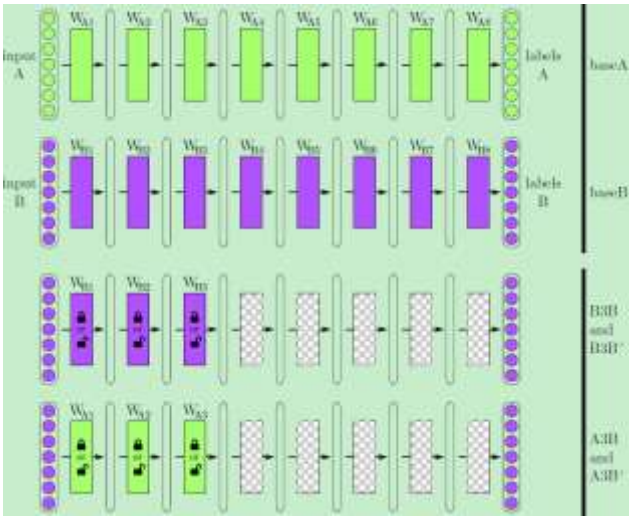
- 上述两个问题在深度神经网络的不同层发生占比不一样。

**主要内容：**

固定网络前 n 层的权值不变，利用目标数据集进行模型后几层权值的训练。如果目标数据集较小，模型的参数较多，那为了避免过拟合，可以将前面几层参数固定，只训练后几层；如果数据集较大，模型参数并不多，则可以全部进行训练。以上就是我们常见的 finetune 的过程。

分别用两个数据集训练了两个八层的卷积神经网络 baseA 和 baseB。这里先定义一些符号：

- BnB 表示利用 baseB 的前 n 层权值初始化一个新的网络，并且固定权值不变，随机初始化后几层网络的权值，在数据集 B 上进行训练；
- AnB 表示利用 baseA 的前 n 层权值初始化一个新的网络，并且固定权值不变，随机初始化后几层网络的权值，在数据集 B 上进行训练；
- BnB+和 BnB 唯一的区别是它不固定权值，全都可以进行训练；
- AnB+与 AnB 的唯一区别也是它的权值不固定。



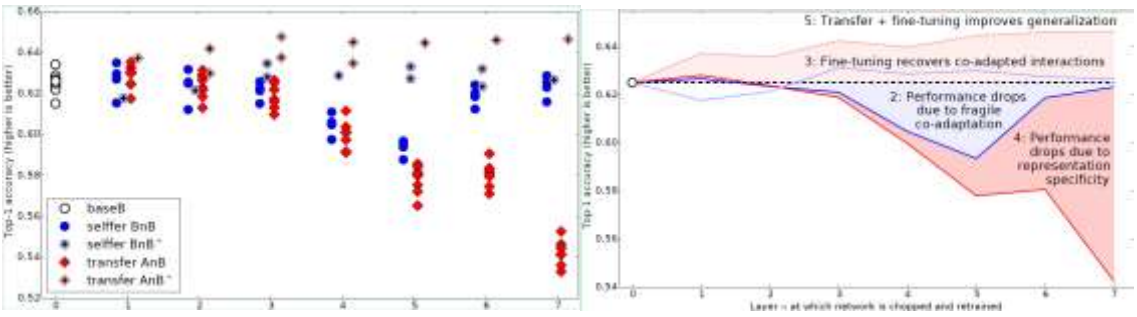
**实验结果：**

1、对于 BnB，说明神经网络层与层之间有着一种共同训练的关系；

2、对于 BnB+，由于全部 finetune 的存在，因此它的结果和 baseB 的结果很类似；

3、AnB 在较浅层时，和 BnB 一样，违反了共同训练的关系，使得效果下降；而到了更深的层次时，由于数据集 A 训练得到的网络泛化能力越来越小，从而使得效果下降；

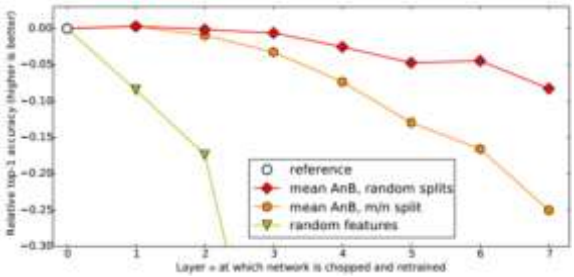
4、AnB+的效果都会比 baseB 好，而且保留的网络层数越多，效果越好。




不相似数据集键转移：

1、两个数据集的类别相差越大，特征的迁移性也就越差；

2、就算是两个没什么关系的数据集，从一个训练出特征，并迁移到另一个数据集上，也要比随机的权值效果好很多。




## 11. Progressive Neural Networks

**主要思想：**

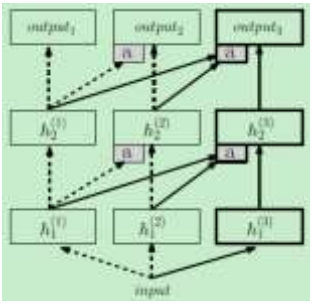
- 能够避免遗忘，可以利用先前知识通过横向连接先前学到的特征；
- 之前的训练都保留，不至于像 fine tune 那样更改原来的网络；
- 而且每一层的特征信息都能得到迁移，并且能够更好的具化分析。

**主要架构：**

Step 1：构造一个多层的神经网络，训练某一个任务，上图第一列

Step 2：构建第二个多层的神经网络，然后固定第一列也就是上一个任务的神经网络，将上一列的神经网络的每一层（注意是每一层）都通过 a 处理连接到第二列的神经网络的每一层作为额外输入。也就是第二个神经网络每一层除了原始的输入，还加上经过 a 处理的之前的神经网络对应层的输入。

Step 3：构建第三个多层神经网络，训练第三个任务，将前两列的神经网络固定，然后同上一样的方法连接到第三个神经网络中。



a 的作用其实主要是为了降维和输入的维度统一（与原始输入匹配）。

$$h_i^{(k)} = \sigma\left(W_i^{(k)} h_{i-1}^{(k)} + U_i^{(k:j)} \sigma(V_i^{(k:j)} \alpha_{i-1}^{(<k)} h_{i-1}^{(<k)})\right), \quad (2)$$

其中$h_{i-1}^{(<k)} = [h_{i-1}^{(1)} \ldots\ldots h_{i-1}^{(j)} \ldots\ldots h_{i-1}^{(k-1)}]$，其维数分别为$n_{i-1}^{(<k)}$，其中 a 并不是线性的横向连接，而是一个隐层的 MLP，非线性的适配器隐层映射到一个统一维度的子空间（对于卷积神经网络，相当于 1*1 的卷积核）。在进 MLP 之前，乘一个标量α，随机初始化（它的作用是对不同输入的不同尺度进行调整）。

## 12. Analysis of Representations for Domain Adaptation

**主要思想：**

- 学习一个分类器，通过寻找最佳的特征空间，当目标域和源域有不同分布的时候，也有较好的表现；
- 推倒出分类器在目标域中训练误差的上界；
- 最佳的特征空间，是最小化源域的训练误差和源域目标域的相似性。

**主要公式：**

源域为 S，目标域为 T，$\mathcal{D}_S$为源域样本的分布，$\widetilde{\mathcal{D}}_S$为特征空间 Z 上源域样本的分布。$f: \mathcal{X} \to [0,1], \tilde{f}$是特征 R 下的 f。源域的训练误差为如下式，目标域的训练误差类似：

$$\begin{aligned}\epsilon_S(h) &= \mathrm{E}_{\mathbf{z}\sim\hat{\mathcal{D}}_S}\left[\mathrm{E}_{y\sim\tilde{f}(\mathbf{z})}\left[y \neq h(\mathbf{z})\right]\right] \\ &= \mathrm{E}_{\mathbf{z}\sim\hat{\mathcal{D}}_S}\left|\tilde{f}(\mathbf{z}) - h(\mathbf{z})\right|.\end{aligned}$$

源域与目标域的特征空间的相似性公式如下，称为$\mathcal{A}$散度。

$$d_A(\mathcal{D},\mathcal{D}') = 2\sup_{A\in\mathcal{A}}\left|\Pr_{\mathcal{D}}[A] - \Pr_{\mathcal{D}'}[A]\right|$$

**则定理 1：**

源域$\mathcal{D}_S$有标签，则目标域的训练误差上界为：

$$\epsilon_T(h) \leq \hat{\epsilon}_S(h) + \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4}{\delta}\right)} + d_{\mathcal{H}}(\tilde{\mathcal{D}}_S,\tilde{\mathcal{D}}_T) + \lambda$$

**定理 2**

源域与目标域都无标签，则目标域的训练误差上界为：

$$\epsilon_T(h) \leq \hat{\epsilon}_S(h) + \frac{4}{m}\sqrt{\left(d\log\frac{2em}{d} + \log\frac{4}{\delta}\right)} + \lambda + d_{\mathcal{H}}(\tilde{\mathcal{U}}_S,\tilde{\mathcal{U}}_T) + 4\sqrt{\frac{d\log(2m') + \log(\frac{4}{\delta})}{m'}}$$

其中$\inf_{h\in\mathcal{H}}\left[\epsilon_S(h) + \epsilon_T(h)\right] \leq \lambda$

则通过目标域训练误差上界可以看出最佳的特征空间，是最小化源域的训练误差和源域目标域的$\mathcal{A}$散度。

## 13. Domain-Adversarial Training of Neural Networks

**主要思想：**

- 将域适应嵌入到学习表示的过程中，因此最终的分类决定是基于对域的变化具有区分性和不变性的特征的

a) 用于预测类标签的标签预测器，在训练和测试期间同时使用

b) 在训练期间区分源和目标域的领域分类器

**主要内容：**

无监督学习，共有N = n + $n'$，源域有标签共有 n 个样本，目标域无标签共有$n'$个样本。

$$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n} \sim (\mathcal{D}_{\mathrm{S}})^n; \quad T = \{\mathbf{x}_i\}_{i=n+1}^{N} \sim (\mathcal{D}_{\mathrm{T}}^{X})^{n'},$$

其中我们的目标是得到分类器 D，使得 Y 的训练误差最小。

$$R_{\mathcal{D}_{\mathrm{T}}}(\eta) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}_{\mathrm{T}}}\left(\eta(\mathbf{x}) \neq y\right)$$

**目标域和源域的散度（距离度量）：**

基于《Analysis of Representations for Domain Adaptation》的思想，得到源域和目标域的$\mathcal{K}$散度如下：

$$d_{\mathcal{H}}(\mathcal{D}_{\mathrm{S}}^{X},\mathcal{D}_{\mathrm{T}}^{X}) = 2\sup_{\eta\in\mathcal{H}}\left|\Pr_{\mathbf{x}\sim\mathcal{D}_{\mathrm{S}}^{X}}[\eta(\mathbf{x}) = 1] - \Pr_{\mathbf{x}\sim\mathcal{D}_{\mathrm{T}}^{X}}[\eta(\mathbf{x}) = 1]\right|.$$

同时，实际上$\mathcal{K}$散度可以如下计算

$$\hat{d}_{\mathcal{H}}(S,T) = 2\left(1 - \min_{\eta\in\mathcal{H}}\left[\frac{1}{n}\sum_{i=1}^{n} I[\eta(\mathbf{x}_i)=0] + \frac{1}{n'}\sum_{i=n+1}^{N} I[\eta(\mathbf{x}_i)=1]\right]\right), \quad (1)$$

**替代距离：**

**由于**$\mathcal{K}$散度不好计算，所以根据《Analysis of Representations for Domain Adaptation》的思想，可以进行如下计算，将源域目标化为标签 0，目标域样本化为标签 1：

$$U = \{(\mathbf{x}_i, 0)\}_{i=1}^{n} \cup \{(\mathbf{x}_i, 1)\}_{i=n+1}^{N}, \quad (2)$$

使用分类器进行分类，并计算分类误差$\epsilon$，则$\mathcal{A}$散度可如下计算：

$$\hat{d}_{\mathcal{A}} = 2(1 - 2\epsilon). \quad (3)$$

**1） 单隐层网络**

**类别分类器：**

$G_{\mathrm{f}}$函数将输入 x 映射到 D 维的特征空间：

$$G_f(\mathbf{x};\mathbf{W},\mathbf{b}) = \mathrm{sigm}(\mathbf{W}\mathbf{x} + \mathbf{b}), \quad (4)$$

$G_y$函数将特征向量映射到 Y 空间：

$$G_y(G_f(\mathbf{x}); \mathbf{V}, \mathbf{c}) = \text{softmax}\big(\mathbf{V}G_f(\mathbf{x}) + \mathbf{c}\big),$$

则需要最小化以下函数，从而得到合适的参数提取特征：

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}} \left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) + \lambda \cdot R(\mathbf{W},\mathbf{b})\right], \tag{5}$$

其中 $\mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i) = \log \frac{1}{G_y(G_f(\mathbf{x}))_{y_i}}$.

**域分类器：**

实际上$\mathcal{H}$散度为：

$$\hat{d}_{\mathcal{H}}\big(S(G_f), T(G_f)\big) = 2\left(1 - \min_{\eta\in\mathcal{H}}\left[\frac{1}{n}\sum_{i=1}^{n} I\big[\eta(G_f(\mathbf{x}_i))=0\big] + \frac{1}{n'}\sum_{i=n+1}^{N} I\big[\eta(G_f(\mathbf{x}_i))=1\big]\right]\right). \tag{6}$$

域分类器为：

$$G_d(G_f(\mathbf{x}); \mathbf{u}, z) = \text{sigm}\big(\mathbf{u}^\top G_f(\mathbf{x}) + z\big). \tag{7}$$

域分类损失：

$$\mathcal{L}_d\big(G_d(G_f(\mathbf{x}_i)), d_i\big) = d_i \log \frac{1}{G_d(G_f(\mathbf{x}_i))} + (1-d_i)\log\frac{1}{1-G_d(G_f(\mathbf{x}_i))},$$

优化目标则是：

$$E(\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c},\mathbf{u},z) \tag{9}$$
$$= \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) - \lambda\Big(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z) + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)\Big),$$

ere we are seeking the parameters $\hat{\mathbf{W}}, \hat{\mathbf{V}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}, \hat{\mathbf{u}}, \hat{z}$ that deliver a saddle point given by

$$(\hat{\mathbf{W}}, \hat{\mathbf{V}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}) = \operatorname*{argmin}_{\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c}} E(\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c},\hat{\mathbf{u}},\hat{z}),$$
$$(\hat{\mathbf{u}}, \hat{z}) = \operatorname*{argmax}_{\mathbf{u},z} E(\hat{\mathbf{W}},\hat{\mathbf{V}},\hat{\mathbf{b}},\hat{\mathbf{c}},\mathbf{u},z).$$

2） 深度网络

$$\mathcal{L}_y^i(\theta_f,\theta_y) = \mathcal{L}_y\big(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i\big),$$
$$\mathcal{L}_d^i(\theta_f,\theta_d) = \mathcal{L}_d\big(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), d_i\big).$$

优化目标为：

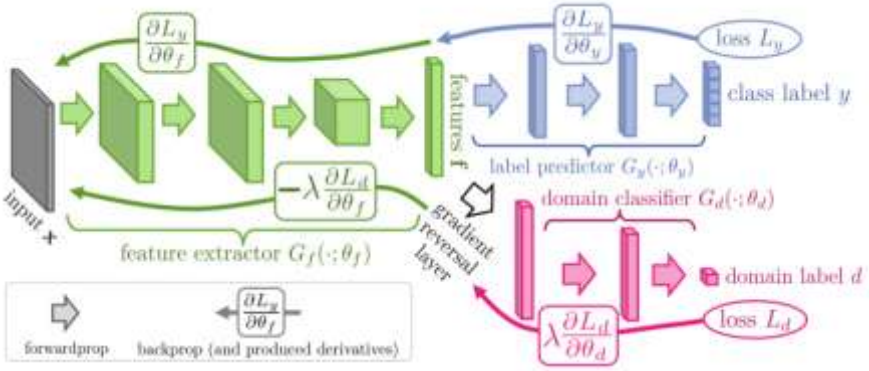
$$E(\theta_f,\theta_y,\theta_d) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\theta_f,\theta_y) - \lambda\Big(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\theta_f,\theta_d) + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\theta_f,\theta_d)\Big), \tag{10}$$

by finding the saddle point $\hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_d$ such that

$$(\hat{\theta}_f, \hat{\theta}_y) = \operatorname*{argmin}_{\theta_f,\theta_y} E(\theta_f,\theta_y,\hat{\theta}_d), \tag{11}$$
$$\hat{\theta}_d = \operatorname*{argmax}_{\theta_d} E(\hat{\theta}_f,\hat{\theta}_y,\theta_d). \tag{12}$$

梯度更新如下：

$$\theta_f \longleftarrow \theta_f - \mu\left(\frac{\partial\mathcal{L}_y^i}{\partial\theta_f} - \lambda\frac{\partial\mathcal{L}_d^i}{\partial\theta_f}\right), \tag{13}$$
$$\theta_y \longleftarrow \theta_y - \mu\frac{\partial\mathcal{L}_y^i}{\partial\theta_y}, \tag{14}$$
$$\theta_d \longleftarrow \theta_d - \mu\lambda\frac{\partial\mathcal{L}_d^i}{\partial\theta_d}, \tag{15}$$




## 14. Domain Adaptive Neural Networks for Object Recognition

**主要思想：**

- incorporates MMD measure as a regularization embedded in the supervised back-propagation training
- preceded by the denoising auto-encoder pretraining

**主要公式：**

**（1） MMD：**

Giventwo probability distributions p and q on X, MMD is defined as

$$\mathcal{MMD}(\mathcal{F}, p, q) = \sup_{f\in\mathcal{F}}(\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)]), \quad (1)$$

RKHS MMD：

$$\mathcal{MMD}_e(\mathbf{x}_s, \mathbf{x}_t) = \left\| \frac{1}{n_s}\sum_{i=1}^{n_s}\phi(\mathbf{x}_s^{(i)}) - \frac{1}{n_t}\sum_{i=1}^{n_t}\phi(\mathbf{x}_t^{(i)}) \right\|_{\mathcal{H}}. \quad (2)$$

$\mathcal{X} \to \mathcal{H}$is referred to as the feature space map

$$\begin{aligned}\mathcal{MMD}_e(\mathbf{x}_s, \mathbf{x}_t) &= \left( \frac{1}{n_s^2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s} k(\mathbf{x}_s^{(i)}, \mathbf{x}_s^{(j)}) + \frac{1}{n_t^2}\sum_{i=1}^{n_t}\sum_{j=1}^{n_t} k(\mathbf{x}_t^{(i)}, \mathbf{x}_t^{(j)}) \right. \\ &\quad \left. - \frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s} k(\mathbf{x}_s^{(i)}, \mathbf{x}_t^{(j)}) \right)^{\frac{1}{2}} \quad (3) \\ &= \left( \frac{\mathrm{Tr}(\mathbf{K}_{ss})}{n_s^2} + \frac{\mathrm{Tr}(\mathbf{K}_{tt})}{n_t^2} - 2\frac{\mathrm{Tr}(\mathbf{K}_{st})}{n_s n_t} \right)^{\frac{1}{2}}, \quad (4)\end{aligned}$$

where $[\mathbf{K}_{r\bullet\bullet}]_{ij} = k(\mathbf{x}_\bullet^{(i)}, \mathbf{x}_\bullet^{(j)})$

**（2） 前馈神经网络：**

three types of layer that arethe input, hidden, and output layers with weighted inter-layer connections：

$$\mathbf{h} = \sigma_1(\mathbf{W}_1^\top \mathbf{x} + \mathbf{b}), \quad (5)$$
$$\mathbf{o} = \sigma_2(\mathbf{W}_2^\top \mathbf{h} + \mathbf{c}), \quad (6)$$

the empirical log-likelihood loss function is given as

$$J_{\mathrm{NNs}} = -\frac{1}{n_s}\sum_{i=1}^{n_s}\sum_{k=1}^{I}([\mathbf{y}_s^{(i)}]_k \log([f(\mathbf{x}_s^{(i)})]_k))$$

**（3） Denoising Auto-encoder**

Unlabeled images from both source and target domainsare considered as inputs to the DAE pretraining

**（4） Domain Adaptive Neural Networks**

the loss function of a single layer DaNN is given by：

$$J_{\mathrm{DaNN}} = J_{\mathrm{NNs}} + \gamma\mathcal{MMD}_e^2(\mathbf{q}_s, \bar{\mathbf{q}}_t), \quad (8)$$

- $q_s = W_1^T X_S + b$（$U_1^T X_S$），$q_t = W_1^T X_t + b$（$U_1^T X_t$）
- choose the Gaussian kernel

$$\begin{aligned}\mathcal{MMD}_e^2(\mathbf{U}_1^\top \mathbf{X}_s, \mathbf{U}_1^\top \mathbf{X}_t) &= \frac{1}{n_s^2}\sum_{i,j=1}^{n_s}\exp\left(-\frac{(\mathbf{x}_s^{(i)} - \mathbf{x}_s^{(j)})^\top \mathbf{U}_1\mathbf{U}_1^\top(\mathbf{x}_s^{(i)} - \mathbf{x}_s^{(j)})}{2s^2}\right) \\ &\quad + \frac{1}{n_t^2}\sum_{i,j=1}^{n_t}\exp\left(-\frac{(\mathbf{x}_t^{(i)} - \mathbf{x}_t^{(j)})^\top \mathbf{U}_1\mathbf{U}_1^\top(\mathbf{x}_t^{(i)} - \mathbf{x}_t^{(j)})}{2s^2}\right) \\ &\quad - \frac{2}{n_s n_t}\sum_{i,j=1}^{n_s,n_t}\exp\left(-\frac{(\mathbf{x}_s^{(i)} - \mathbf{x}_t^{(j)})^\top \mathbf{U}_1\mathbf{U}_1^\top(\mathbf{x}_s^{(i)} - \mathbf{x}_t^{(j)})}{2s^2}\right) \quad (9)\end{aligned}$$

the gradient of $\mathcal{MMD}_e^2$:

$$\frac{\partial \mathcal{M}_{st}^2}{\partial \mathbf{U}_1} = \frac{1}{n_s^2}\sum_{i,j=1}^{n_s} G_{ss}(i,j) + \frac{1}{n_t^2}\sum_{i,j=1}^{n_t} G_{tt}(i,j) - \sum_{i,j=1}^{n_s,n_t} \frac{2}{n_s n_t} G_{st}(i,j). \quad (11)$$

其中：Let $G_{\bullet\bullet}(i,j)$ be the gradient of $k_G(\mathbf{U}_1^\top \mathbf{x}_\bullet^{(i)}, \mathbf{U}_1^\top \mathbf{x}_\bullet^{(j)})$

$$G_{\bullet\bullet}(i,j) = -\frac{1}{s^2} k_G(\mathbf{x}_\bullet^{(i)}, \mathbf{x}_\bullet^{(j)})(\mathbf{x}_\bullet^{(i)} - \mathbf{x}_\bullet^{(j)})(\mathbf{x}_\bullet^{(i)} - \mathbf{x}_\bullet^{(j)})^\top \mathbf{U}_1.$$

**Algorithm 1:** The DaNN supervised back-propagation algorithm.

**Data:**
$\mathbf{U}_1 \in \mathbb{R}^{(d+1)\times k}$ and $\mathbf{U}_2 \in \mathbb{R}^{(k+1)\times l}$ are the weight-bias matrices in the first and second layers, respectively.
$\mathbf{h} \in \mathbb{R}^k$ is the hidden layer vector.
$\mathbf{o} \in \mathbb{R}^l$ is the output layer vector.
$\alpha, \gamma$ are the learning rate and the MMD regularization constant.

**begin**

1. Initialize $\mathbf{U}_1$ and $\mathbf{U}_2$ with small random real values;
2. Update $\mathbf{U}_2$ and $\mathbf{U}_1$ using the batched stochastic gradient descent by standard forward - backward pass w.r.t. $J_{\text{DNN}}$;
3. Update $\mathbf{U}_1$ by the offline gradient descent as follows
$$\mathbf{U}_1(t) := \mathbf{U}_1(t-1) - \alpha\gamma \frac{\partial \mathcal{M}_{st}^2}{\partial \mathbf{U}_1}$$
4. Repeat Steps 2 and 3 until the end of the epoch;

**end**

## 15. Deep Domain Confusion: Maximizing for Domain Invariance

**主要思想：**

- 被称为 DDC
- 同时减小分类损失和训练集与测试集的分布差异
- 分类损失利用深度网络的常用损失就可以很容易地进行表示
- 训练集和测试集的分布差异可以用最大均值差异（maximum mean discrepancy（MMD））

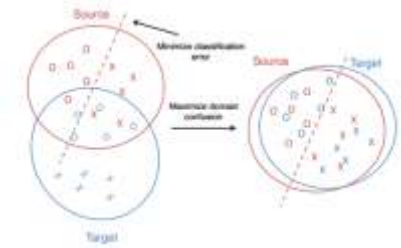
**主要方法：**

设训练集为 Xs，测试集为 Xt，定义一个映射函数，也可以叫核函数，它能够将样本 x 映射到合理的映射空间中，在这个空间中，最小化 MMD 可以写为：

$$\mathrm{MMD}(X_S, X_T) = \left\| \frac{1}{|X_S|}\sum_{x_s \in X_S} \phi(x_s) - \frac{1}{|X_T|}\sum_{x_t \in X_T} \phi(x_t) \right\| \quad (1)$$

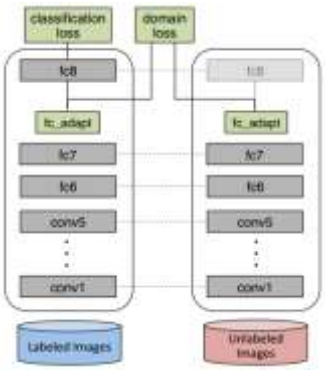
整体的损失函数可以定义为：

$$\mathcal{L} = \mathcal{L}_C(X_L, y) + \lambda \mathrm{MMD}^2(X_S, X_T) \quad (2)$$




将 MMD 结合如经典的深度网络中，则只需要增加一层自适应层（adaptation layer），在此，就需要

考虑两个问题：1）这个自适应层应该放在模型的什么地方；2）自适应层的维度应该设置为多大。为了解决这两个问题，作者测试了将自适应层放到各个位置，设置自适应层不同维度后 MMD 的值，该值越小越好，从而选择出了最佳的位置和维度。




## 16. Learning Transferable Features with Deep Adaptation Networks

**主要思想：**

一是 DDC 只适配了一层网络， DAN 就多适配几层；

二是 DDC 是用了单一核的 MMD，单一固定的核可能不是最优的核。DAN 用了多核的 MMD（MK-MMD），效果比 DDC 更好。

**主要方法：**

（1） 多核 MMD

对于两个概率分布p,q，MK-MMD 距离就是：

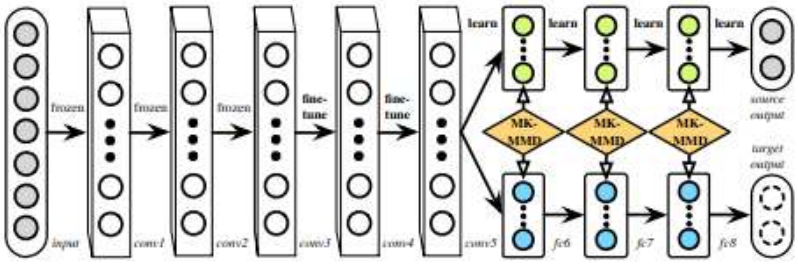
$$d_k^2(p,q) \triangleq \left\| \mathbf{E}_p[\phi(\mathbf{x}^s)] - \mathbf{E}_q[\phi(\mathbf{x}^t)] \right\|_{\mathcal{H}_k}^2. \quad (1)$$

这个多个核一起定义的 kernel 就是：

$$\mathcal{K} \triangleq \left\{ k = \sum_{u=1}^{m} \beta_u k_u : \sum_{u=1}^{m} \beta_u = 1, \beta_u \geqslant 0, \forall u \right\}. \quad (2)$$

（2） 多层适配

网络的迁移能力在这三层开始就会特别地 task-specific，所以要着重适配这三层。




（3） Deep Adaptation Networks (DAN)

优化目标由两部分组成：损失函数和分布距离。于是，DAN 的优化目标就是：

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(\mathbf{x}_i^a), y_i^a) + \lambda \sum_{l=l_1}^{l_2} d_k^2(\mathcal{D}_s^l, \mathcal{D}_t^l), \quad (4)$$

Θ表示网络的所有权重和 bias 参数。其中$l_1, l_2$分别是 6 和 8,表示网络适配是从第 6 层到第 8 层，前面的不进行适配。λ是惩罚系数。$J$就定义了一个损失函数，在深度网络中一般都是 cross-entropy。

（4） 学习策略(学习网络参数Θ和 MMD 的β)

- 网络参数Θ

对Θ的学习依赖于 MK-MMD 距离的计算。总是可以把 MK-MMD 展开成一堆内积的形式。然而，数据之间两两计算内积是非常复杂的，采用了 Gretton 在文章提出的对 MK-MMD 的无偏估计：$d_k^2(p,q) = \frac{2}{n_s} \sum_{i=1}^{n_s/2} g_k(\mathbf{z}_i)$，其中四元组$\mathbf{z}_i \triangleq (\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s, \mathbf{x}_{2i-1}^t, \mathbf{x}_{2i}^t)$，将 kernel 作用到$z_i$上以后，变成$g_k(\mathbf{z}_i) \triangleq k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s) + k(\mathbf{x}_{2i-1}^t, \mathbf{x}_{2i}^t) - k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^t) - k(\mathbf{x}_{2i}^s, \mathbf{x}_{2i-1}^t)$。

- MMD 的β

目标是：确保每个 kernel 生成的 MMD 距离的方差最小。也就是

$$\max_{k \in \mathcal{K}} d_k^2(\mathcal{D}_s^l, \mathcal{D}_t^l) \sigma_k^{-2}, \quad (7)$$

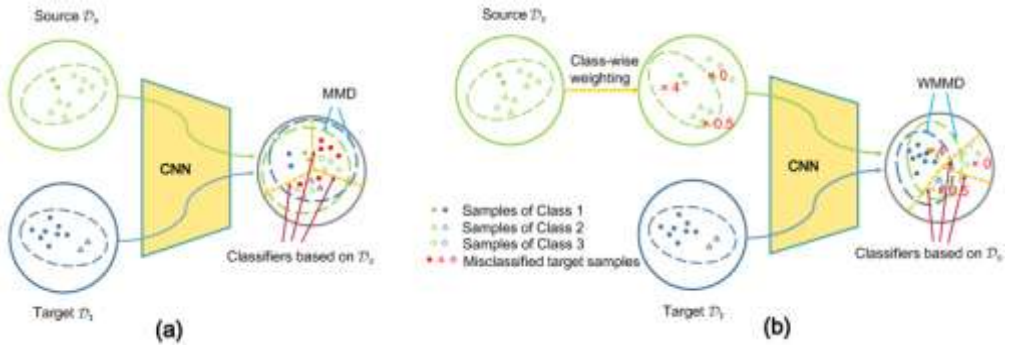
其中$\sigma_k^2 = \mathbf{E}_{\mathbf{z}} g_k^2(\mathbf{z}) - [\mathbf{E}_{\mathbf{z}} g_{\underline{k}}(\mathbf{z})]^2$是估计方差。实际求解的时候问题可以被规约成一个二次规划问题求解。

## 17. Mind the Class Weight Bias: Weighted Maximum Mean Discrepancy for Unsupervised Domain Adaptation

**主要思想：**

- learning transferable CNN features for unsupervised domain adaptation
- introduce class-specific auxiliary weights into the original MMD for exploiting the class prior probability on source and target domains
- weighted MMD model is defined by introducing an auxiliary weight for each class in the source domain
- a classification EM algorithm is suggested by alternating between assigning the pseudo-labels, estimating auxiliary weights and updating model parameters.

**主要方法：**




**1） MMD：**

An empirical estimate of MMD can be given by：

$$\mathrm{MMD}^2(\mathcal{D}_s, \mathcal{D}_t) = \left\| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathbf{x}_i^s) - \frac{1}{N}\sum_{j=1}^{N}\phi(\mathbf{x}_j^t) \right\|_{\mathcal{H}}^2, \quad (2)$$

based on pairwise similarity and is computed in quadratic time complexity, it is prohibitively time-consuming and unsuitable for using mini-batch stochastic gradient descent (SGD) in CNN-based domain adaptation methods.

$$\mathrm{MMD}_l^2(s,t) = \frac{2}{M}\sum_{i=1}^{M/2} h_l(\mathbf{z}_i), \quad (4)$$

where $h_l$ is an operator defined on a quad-tuple $\mathbf{z}_i = [\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s, \mathbf{x}_{2j-1}^t, \mathbf{x}_{2j}^t]$,

$$\begin{aligned} h_l(\mathbf{z}_i) =& k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s) + k(\mathbf{x}_{2j-1}^t, \mathbf{x}_{2j}^t) \\ & - k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2j}^t) - k(\mathbf{x}_{2i}^s, \mathbf{x}_{2j-1}^t). \end{aligned} \quad (5)$$

**2） Weighted-MMD：**

both p s (x s ) and p t (x t ) can be further represented as the mixtures of class conditional distributions：

$$\begin{aligned} p_u(\mathbf{x}^u) &= \sum_{c=1}^{C} p_u(y^u = c) p_u(\mathbf{x}^u | y^u = c) \\ &= \sum_{c=1}^{C} w_c^u p_u(\mathbf{x}^u | y^u = c), u \in \{s, t\}, \end{aligned}$$

where $w_c^s = p_s(y^s = c)$ and $w_c^t = p_t(y^t = c)$ note theclass prior probability.

$p_{s,a}(X_S)$has the same class weights with the target domain but owns the class conditional distributions in source domain.

$$p_{s,a}(\mathbf{x}^s) = \sum_{c=1}^{C} \alpha_c w_c^s p_s(\mathbf{x}^s | y^s = c).$$

the empirical estimation of weighted MMD$p_{s,a}(X_S)$ and p t (x t ) can be given by:

$$\mathrm{MMD}_w^2(\mathcal{D}_s,\mathcal{D}_t)=\left\|\frac{1}{\sum_{i=1}^{M}\alpha_{y_i^s}}\sum_{i=1}^{M}\alpha_{y_i^s}\phi(\mathbf{x}_i^s)-\frac{1}{N}\sum_{j=1}^{N}\phi(\mathbf{x}_j^t)\right\|_{\mathcal{H}}^2 \tag{8}$$

3) **Weighted Domain Adaptation Network**

to generalize CNN for domain adaptation, the weighted MMD-based regularizers are added to the higher layers of CNN.

ECM 算法

(i) E-step: estimating the class posterior probability of $\{x_j^t\}_{J=1}^N$

$$p(y_j^t=c|\mathbf{x}_j^t)=g_c(\mathbf{x}_j^t;\mathbf{W}). \tag{12}$$

(ii) C-step: assigning pseudo-labels$\{\hat{y}_j\}_{J=1}^N$（最大后验概率）and estimating auxiliary weights α

$$\hat{y}_j=\arg\max_c p(y_j^t=c|\mathbf{x}_j^t).$$

(iii) M-step: updating the model parameters W,updated with mini-batch SGD

$$\min_{\boldsymbol{W},\{\hat{y}_j\}_{j=1}^N,\boldsymbol{\alpha}}\frac{1}{M}\sum_{i=1}^{M}\ell(\mathbf{x}_i^s,y_i^s;\mathbf{W})+\gamma\frac{1}{N}\sum_{j=1}^{N}\ell(\mathbf{x}_j^t,\hat{y}_j^t;\mathbf{W})$$
$$+\lambda\sum_{l=l_1}^{l_2}\mathrm{MMD}_{l,w}(\mathcal{D}_s^l,\mathcal{D}_t^l), \tag{11}$$

## 18. Unsupervised Domain Adaptation with Residual Transfer Networks

**主要思想：**

1） 特征自适应的问题（DAN:MK-MMD）

2） 源任务与目标任务的分类器自适应问题(残差网络)

**主要方法：**

（1） 卷积神经网络

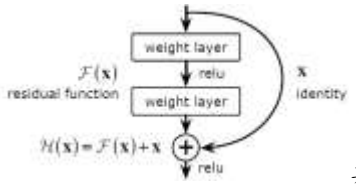
源域的损失函数（交叉熵）：

$$\min_{f_s\in\mathcal{F}}E(\mathcal{D}_s;f_s)=\frac{1}{n_s}\sum_{i=1}^{n_s}L(f_s(\mathbf{x}_i^s),y_i^s), \tag{1}$$

$$L(f_s(\mathbf{x}_i^s),y_i^s)=-\sum_{j=1}^{c}1\{y_i^s=j\}\log f_j^s(\mathbf{x}_i^s),$$
$$f_j^s(\mathbf{x}_i^s)=e^{h_{ij}^s}/\sum_{j'}e^{h_{ij'}^s}$$

（2） Classifier Adaptation

- 使目标分类器 ft 不能偏离源分类器 fs




其中，$\mathbf{x}\triangleq f_T(\mathbf{x}),\ \mathcal{H}(\mathbf{x})\triangleq f_S(\mathbf{x}),\ \text{and}\ \mathcal{F}(\mathbf{x})\triangleq\Delta f(f_T(\mathbf{x}))$.

则利用残差网络，得到源域分类函数：

$$f_S(\mathbf{x})=f_T(\mathbf{x})+\Delta f(f_T(\mathbf{x})), \tag{2}$$

- 保证 ft 适合目标域的特定结构

类别自适应损失函数

$$\min_{f_t\in\mathcal{F}}E(\mathcal{D}_t;f_t)=\frac{1}{n_t}\sum_{i=1}^{n_t}H(f_t(\mathbf{x}_i^t)), \tag{3}$$

其中，$H(f_t(\mathbf{x}_i^t))=-\sum_{j=1}^{c}f_j^t(\mathbf{x}_i^t)\log f_j^t(\mathbf{x}_i^t)$ $p(y_i^t=j|\mathbf{x}_i^t;f_t)=f_j^t(\mathbf{x}_i^t)$，$y_i^t$是根据后验概率得到的伪标签。

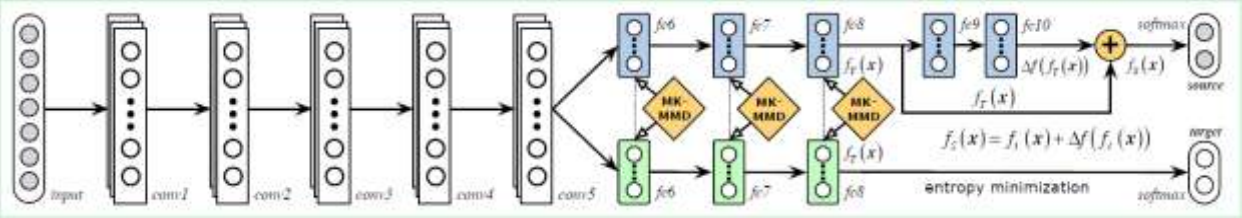
（3） Feature Adaptation

采用 DAN 方法中的，MK-MMD:

$$\min_{f_s,f_t\in\mathcal{F}} E\left(\mathcal{D}_s,\mathcal{D}_t;f_s,f_t,k\right)=\sum_{\ell\in\mathcal{L}} M_k^2\left(\mathcal{D}_s^\ell,\mathcal{D}_t^\ell\right), \quad (4)$$

（4） 整体网络架构

$$\min_{f_S=f_T+\Delta f(f_T)} \frac{1}{n_s}\sum_{i=1}^{n_s} L\left(f_s\left(\mathbf{x}_i^s\right),y_i^s\right)+\frac{\gamma}{n_t}\sum_{i=1}^{n_t} H\left(f_t\left(\mathbf{x}_i^t\right)\right)+\lambda\sum_{\ell\in\mathcal{L}} M_k^2\left(\mathcal{D}_s^\ell,\mathcal{D}_t^\ell\right),$$



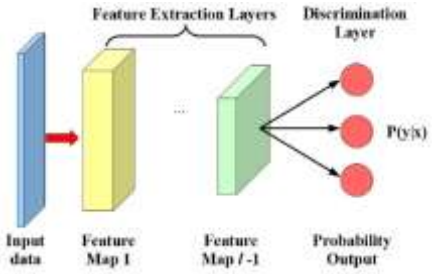

## 19. Deep Transfer Network: Unsupervised Domain Adaptation

**主要思想：**

1） 考虑 source 和 target dataset 之间的边缘分布和条件分布都不相同，即$\mathrm{P}(x^s)\neq\mathrm{P}(x^t)$且$\mathrm{P}(y^s|x^s)\neq\mathrm{P}(y^t|x^t)$;

2） 提出了一种用 MMD 来同时对两个域上的 marginal distribution 和 conditional distribution 进行约束的迁移网络。

- 用 MMD 来对两个域（源域和目标域）上的提取到的特征分布进行约束，从而使两个域上的特征分布尽可能相同，这个分布叫做 marginal distribution；同时对两个域上的 softmax 分类结果用 MMD 来进行约束，使得两个分类结果的分布尽可能相同，这个分布叫做 conditional distribution




**主要方法：**

（1） MMD

$$\begin{aligned}\mathrm{MMD}&=\left\|\frac{1}{n^s}\sum_{i=1}^{n^s}\mathbf{x}_i^s-\frac{1}{n^t}\sum_{j=1}^{n^t}\mathbf{x}_j^t\right\|_2^2\\&=\mathrm{Tr}(\mathbf{XMX}^T),\end{aligned} \quad (1)$$

$$M_{ij}=\begin{cases}1/(n^s)^2, & i\leqslant n^s, j\leqslant n^s\\ 1/(n^t)^2, & i>n^s, j>n^s\\ -1/(n^sn^t), & \text{otherwise}\end{cases}. \quad (2)$$

（2） Matching Marginal Distributions

特征提取层，也就是图中的前 l-1 层，最后一层是分类层，输出的是属于每个类的概率。本文在第 l-1 层以及分类器的输出衡量源域和目标域基于 MMD 的分布损失。在特征分布上，通过在目标函数中的 marginal MMD 来衡量两个域上的分布区别，具体如下：

$$\begin{aligned}\mathrm{MMD}_{mar}&=\left\|\frac{1}{n^s}\sum_{i=1}^{n^s}\mathbf{h}_i^s(l-1)-\frac{1}{n^t}\sum_{j=1}^{n^t}\mathbf{h}_j^t(l-1)\right\|_2^2\\&=\mathrm{Tr}(\mathbf{H}(l-1)\mathbf{MH}^T(l-1)),\end{aligned} \quad (4)$$

（3） Matching Conditional Distributions

先通过后验概率得到目标域的伪标签：

$$P(y=c|\mathbf{x})=\mathrm{softmax}_c(\mathbf{w}_c^T\mathbf{x})=\frac{e^{\mathbf{w}_c^T\mathbf{x}}}{\sum_j e^{\mathbf{w}_j^T\mathbf{x}}}. \quad (5)$$

在分类器的输出层添加 MMD 损失使得两个域在条件分布(conditional distribution)上尽可能一致。定义如下的 conditional MMD:

$$\begin{aligned}&\mathrm{MMD}_{con}\\&=\sum_{c=1}^{C}\|\frac{1}{n^s}\sum_{i=1}^{n^s}P(y^s=c|\mathbf{x}_i^s)-\frac{1}{n^t}\sum_{j=1}^{n^t}P(y^t=c|\mathbf{x}_j^t)\|^2\\&=\sum_{c=1}^{C}\mathbf{q}_c^T\mathbf{M}\mathbf{q}_c,\end{aligned}\tag{6}$$

（4）Final Objective Function

最后加上网络结构中标准的分类损失，得到整个网络的目标函数如下：

$$\begin{aligned}\mathbf{J}(\mathbf{W})&=-\mathcal{L}(\mathbf{W})+\lambda\mathrm{Tr}(\mathbf{H}(l-1)\mathbf{M}\mathbf{H}^T(l-1))\\&\quad+\mu\sum_{c=1}^{C}\mathbf{q}_c^T\mathbf{M}\mathbf{q}_c\\&=-\mathcal{L}(\mathbf{W})+\lambda\,\mathrm{MMD}_{mar}+\mu\,\mathrm{MMD}_{con},\end{aligned}\tag{8}$$

其中，$-\mathcal{L}(\mathbf{W})=-\sum_{i=1}^{n}\log\left(P(y=y_i|\mathbf{x}_i,\mathbf{W})\right).$

**Algorithm 1** Optimization of the Deep Transfer Network

**Input:** Source data with label $\mathbf{X}^s, Y^s$ and target data $\mathbf{X}^t$.
**Output:** Parameters of deep transfer network $\mathbf{W}$ and predicted labels of the target samples $\hat{Y}^t$.

1: **begin:**
2: Set $i=0$. Get $\hat{Y}_0^t$ by training a baseline neural network with $\mathbf{X}^s, Y^s$ and testing with $\mathbf{X}^t$.
3: **repeat**
4: $i=i+1$.
5: Make mini-batches.
6: Get $\mathbf{W}$ by optimizing Eq. 8 using source data $\mathbf{X}^s, Y^s$ and target data $\mathbf{X}^t, \hat{Y}_{i-1}^t$.
7: Predict $\mathbf{X}^t$ with the network to get $\hat{Y}_i^t$.
8: **until** $\hat{Y}_i^t=\hat{Y}_{i-1}^t$ **or** $i>T$.
9: **return** $\mathbf{W}, \hat{Y}^t=\hat{Y}_i^t$.

训练的时候采取了 mini batch 的方法，用一个 batch 来代替数据集的分布，mini batch 其实并不会影响实验结果，因为假设将数据集切分为 N 份，在 N 上的 MMD 会大于总的数据集上的 MMD，用公式表达如下：

$$\begin{aligned}\mathrm{MMD}&=\|\frac{1}{n^s}\sum_{i=1}^{n^s}\mathbf{x}_i^s-\frac{1}{n^t}\sum_{j=1}^{n^t}\mathbf{x}_j^t\|_2^2\\&\leqslant N\sum_{k=1}^{N}\|\frac{1}{n^s}\sum_{\mathbf{x}_i^s\in\mathcal{B}_k}\mathbf{x}_i^s-\frac{1}{n^t}\sum_{\mathbf{x}_j^t\in\mathcal{B}_k}\mathbf{x}_j^t\|_2^2\\&=N^2\sum_{k=1}^{N}\|\frac{1}{n_k^s}\sum_{\mathbf{x}_i^s\in\mathcal{B}_k}\mathbf{x}_i^s-\frac{1}{n_k^t}\sum_{\mathbf{x}_j^t\in\mathcal{B}_k}\mathbf{x}_j^t\|_2^2.\end{aligned}\tag{12}$$

## 20. Unsupervised Domain Adaptation by Backpropagation

**主要思想：**

利用对抗网络的框架来学习域不变特征

**主要方法：**

学习得到一个 domain classifier，它能对不同域进行区分。学习不变特征的假设就是，在训练好的 domain classifier 上，如果不同域上的特征在这个分类器上不能区分，也就是这个分类器的分类损失很大， 那么这个特征就可以看作是不变特征。

对于 label classification，我们要让学到的特征尽可能具有 label 的分类判别信息，也就是最小化 label classifier 的分类损失。

$$(\hat{\theta}_f,\hat{\theta}_y)=\arg\min_{\theta_f,\theta_y}E(\theta_f,\theta_y,\hat{\theta}_d)\tag{2}$$

$$\hat{\theta}_d=\arg\max_{\theta_d}E(\hat{\theta}_f,\hat{\theta}_y,\theta_d)\,.\tag{3}$$

其中，

$$E(\theta_f,\theta_y,\theta_d)=\sum_{\substack{i=1..N\\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y),y_i\right)-$$
$$\lambda\sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d),y_i\right)=$$
$$=\sum_{\substack{i=1..N\\ d_i=0}} L_y^i(\theta_f,\theta_y)-\lambda\sum_{i=1..N} L_d^i(\theta_f,\theta_d) \qquad (1)$$

其中 theta_f 表示特征提取的参数，theta_y 表示 label classifier 的分类器，theta_d 表示 domain classifier 的参数，L_y 表示 label classifier 的分类器，L_d 表示 domain classifier 的分类器。N 代表所有样本的数目，d_i 代表域标签，0 代表源域。

针对上面（2）和（3）中的问题，可以用下面的方法对网络参数进行更新：

$$\theta_f \longleftarrow \theta_f-\mu\left(\frac{\partial L_y^i}{\partial\theta_f}-\lambda\frac{\partial L_d^i}{\partial\theta_f}\right) \qquad (4)$$
$$\theta_y \longleftarrow \theta_y-\mu\frac{\partial L_y^i}{\partial\theta_y} \qquad (5)$$
$$\theta_d \longleftarrow \theta_d-\mu\frac{\partial L_d^i}{\partial\theta_d} \qquad (6)$$

为了使上面的式子符合标准的方向传播的表示，作者定义了一个中间函数，它在前向和反向过程中有两个不等价的表现形式：

$$R_\lambda(\mathbf{x})=\mathbf{x} \qquad (7)$$
$$\frac{dR_\lambda}{d\mathbf{x}}=-\lambda\mathbf{I} \qquad (8)$$

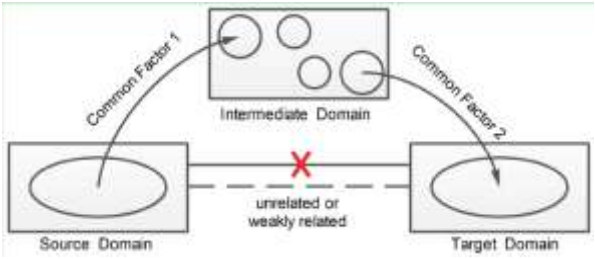
对应的损失函数表示为（这样就可以用标准的 SGD 方法进行反向传播）：

$$\tilde{E}(\theta_f,\theta_y,\theta_d)=\sum_{\substack{i=1..N\\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y),y_i\right)+$$
$$\sum_{i=1..N} L_d\left(G_d(R_\lambda(G_f(\mathbf{x}_i;\theta_f));\theta_d),y_i\right) \qquad (9)$$

## 21. Transitive Transfer Learning

**主要思想：**

- 两个 domain 之间如果相隔得太远，那么我们就插入一些 intermediate domains，一步步做迁移
- 首先选择一个或多个 intermediate domains 作为源域和目标域之间的桥梁，
- 通过 intermediate domains 完成知识的传递




**主要方法：**

**（1）Intermediate domain selection**

利用 Domain complexity 和$\mathcal{A}$-distance 估计领域难度和成对域距离。

- Domain complexity

  域复杂性可以由低频率的长尾特征的百分比表示。这些长尾特征带来长尾特征分布和显著的特征多样性，使机器学习变得困难。

$$cplx(D)=\frac{|\{x|c(x)<t\times n\}|}{m}, \qquad (1)$$

- $\mathcal{A}$-distance

$$dis_{\mathcal{A}}(\boldsymbol{D}_i,\boldsymbol{D}_j)=2(1-2\min_{h\in\mathcal{H}} error(h|\boldsymbol{D}_i,\boldsymbol{D}_j)), \qquad (2)$$

给定一组tr $=\{\mathcal{S},\mathcal{D},\mathcal{T}\}$，计算源域，目标域和中间域的 6 个特征（c1-c6），如下：

**Table 2: Domain characteristic features**

| feature | description |
|---|---|
| cplx_src ($c_1$) | source domain complexity |
| cplx_inter ($c_2$) | intermediate domain complexity |
| cplx_tar ($c_3$) | target domain complexity |
| $dis_{\mathcal{A}}^{si}$ ($c_4$) | a_distance between source and intermediate |
| $dis_{\mathcal{A}}^{st}$($c_5$) | a_distance between source and target |
| $dis_{\mathcal{A}}^{it}$ ($c_6$) | a_distance between intermediate and target |

从候选的 Intermediate domain 中，计算f(tr)，选择最大的作为最终的中间域：

$$f(\boldsymbol{tr}) = \delta(\beta_0 + \sum_{i=1}^{n} \beta_i c_i), \quad (3)$$

其中β通过最大化下面的公式得到：

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{t} l^{(i)} \log f(\boldsymbol{tr}_i) + (1 - l^{(i)}) \log(1 - f(\boldsymbol{tr}_i)), \quad (4)$$

（2） Transitive knowledge transfer

- 非负矩阵三因子分解

给定 x 可以将其进行矩阵分解如下：

$$\arg min_{F,A,G^T} \quad \mathcal{L} = \|X - FAG^T\|, \quad (5)$$

$F_{i,j}$表示第 i 个特征属于第 j 个特征聚类的概率;G 如果第 i 行最大的元素是位于第 j 列,这意味着以实例属于第 j 个标签;$A_{i,j}$是第 i 个特征聚类与第 j 个实例集群相关的概率。

- NMTF for Transfer Learning

$$\begin{aligned}\mathcal{L}_{ST} &= \|X_s - F_s A_s G_s\| + \|X_t - F_t A_t G_t\| \\ &= \left\|X_s - [F^1, F_s^2]\begin{bmatrix}A^1 \\ A_s^2\end{bmatrix} G_s^T\right\| + \left\|X_t - [F^1, F_t^2]\begin{bmatrix}A^1 \\ A_t^2\end{bmatrix} G_t^T\right\|,\end{aligned} \quad (6)$$

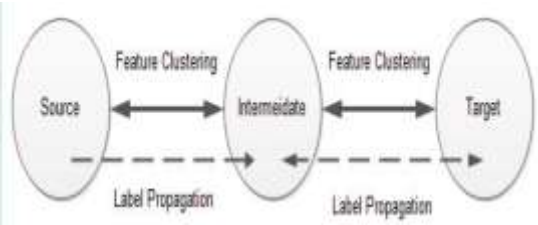
其中$F^1, A^1$是共有特征聚类，$F_S^2, F_t^2, A_S^2, A_t^2$是特有特征聚类。通过传递共有特征聚类$F^1, A^1$，从而得到 Gt。

- TTL Transfer Learning Algorithm

根据以下公式计算最终 Gt，其中前两行是源域到中间域的迁移，后两行是中间域到目标域的迁移。可见中间域的 F 和 A 被两次分解为不同的共有举证$\widehat{F^1}\widehat{A^1}$和$\widetilde{F^1}\widetilde{A^1}$。

$$\begin{aligned}\mathcal{L} &= \|X_s - F_s A_s G_s^T\| + \|X_I - F_I A_I G_I^T\| + \\ &\quad \|X_I - F_I' A_I' G_I^T\| + \|X_t - F_t A_t G_t^T\| \\ &= \left\|X_s - [\hat{F}^1, \hat{F}_s^2]\begin{bmatrix}\hat{A}^1 \\ \hat{A}_s^2\end{bmatrix} G_s^T\right\| + \left\|X_I - [\hat{F}^1, \hat{F}_I^2]\begin{bmatrix}\hat{A}^1 \\ \hat{A}_I^2\end{bmatrix} G_I^T\right\| + \\ &\quad \left\|X_I - [\tilde{F}^1, \tilde{F}_I^2]\begin{bmatrix}\tilde{A}^1 \\ \tilde{A}_I^2\end{bmatrix} G_I^T\right\| + \left\|X_t - [\tilde{F}^1, \tilde{F}_t^2]\begin{bmatrix}\tilde{A}^1 \\ \tilde{A}_t^2\end{bmatrix} G_t^T\right\|.\end{aligned} \quad (7)$$

通过传递 feature clustering 和 label information 得到最终的 Gt:




分解后的矩阵 F 包含隐含特征簇的信息，指示每个隐藏簇的特征分布。因此，每列 F 的总和等于 1。标签矩阵 G 表示每个实例的标签分布。因此，G 的每一行的总和必须等于 1。

$$\begin{aligned}&\arg\min_{F_s,A_s,F_I,A_I,G_I,F_I',A_I',F_t,A_t,G_t} \quad \mathcal{L} \\ &s.t. \\ &\sum_{i=1}^{m} \hat{F}^1(i,j) = 1, \quad \sum_{i=1}^{m} \hat{F}_s^2(i,j) = 1, \\ &\sum_{i=1}^{m} \hat{F}_I^2(i,j) = 1, \quad \sum_{i=1}^{m} \tilde{F}^1(i,j) = 1, \\ &\sum_{i=1}^{m} \tilde{F}_I^2(i,j) = 1, \quad \sum_{i=1}^{m} \tilde{F}_t^2(i,j) = 1, \\ &\sum_{j=1}^{c} G_I(i,j) = 1 \quad \sum_{j=1}^{c} G_t(i,j) = 1.\end{aligned} \quad (8)$$

**求解方法：**

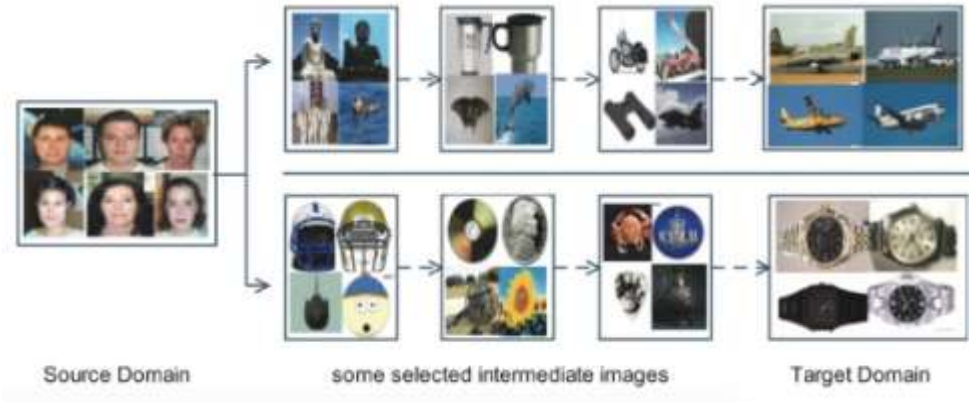
**Algorithm 1** The *TTL* Transfer Learning Algorithm

1: **Input**: Source, target, intermediate domains $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{D}$, the parameters $p$, and the number of iterations $Iter_{max}$.
2: Initialize the matrices $F_s, A_s, F_I, A_I, G_I, F_t, A_t, G_t$.
3: **while** $iter < Iter_{max}$ **do**
4: Update the sub-matrices of $F_s, A_s, F_I, A_I, F_t, A_t$ and label matrices $G_I$, $G_t$ according to the updating rules given in Eq. (9) and Eq. (12) of the Appendix section.
5: Normalize the sub-matrices of $F_s$, $F_I$, $F_t$, and label matrices $G_I$, $G_t$ according to the normalization rules given in Eq. (10) and Eq. (13) of the Appendix.
6: **end while**
7: **Output**: the predicted results of $G_t$.

## 22. Distant Domain Transfer Learning

**主要思想：**

- 通过逐步从中间域中选择多个样本子集，来搭建源域和目标域之间迁移的桥梁
- 源域有标签，目标域少量标签，中间域样本较多但无标签
- 利用重构误差作为衡量两个域之间的距离
- 逐步从源域剔除无用数据，从中间域选择有用数据，与目标域一起学起共同高层特征。




**主要方法：**

（1）Auto-Encoders and Its Variant

自动编码器是一个无监督的前馈神经网络，它有一个输入层，一个或多个隐藏层，以及一个输出层。fe（•）将其映射到一个隐式表示，然后通过解码函数 fd（•）对其进行解码以重建 x。

$$\text{encoding}: \boldsymbol{h} = f_e(\boldsymbol{x}), \text{ and decoding}: \hat{\boldsymbol{x}} = f_d(\boldsymbol{h}),$$

（2）Instance Selection via Reconstruction Error

为从中间选择有用的实例域,删除不相关的实例从源域为目标域，通过最小化重构错误我们学习一对编解码器。

$$\mathcal{J}_1(f_e, f_d, \boldsymbol{v}_S, \boldsymbol{v}_T) = \frac{1}{n_S}\sum_{i=1}^{n_S} v_S^i \|\hat{\boldsymbol{x}}_S^i - \boldsymbol{x}_S^i\|_2^2 + \frac{1}{n_I}\sum_{i=1}^{n_I} v_I^i \|\hat{\boldsymbol{x}}_I^i - \boldsymbol{x}_I^i\|_2^2 + \frac{1}{n_T}\sum_{i=1}^{n_T} \|\hat{\boldsymbol{x}}_T^i - \boldsymbol{x}_T^i\|_2^2 + R(\boldsymbol{v}_S, \boldsymbol{v}_T), \quad (1)$$

其中，Vs 和 Vi 是选择向量，如果选择该实例，则对应元素为 1，否则为 0。R 为正则项，防止 V 全部为 0。

$$R(\boldsymbol{v}_S, \boldsymbol{v}_T) = -\frac{\lambda_S}{n_S}\sum_{i=1}^{n_S} v_S^i - \frac{\lambda_I}{n_I}\sum_{i=1}^{n_I} v_I^i.$$

（3）Incorporation of Side Information

对于源和目标域，标记的数据可以用作 Side Information，而对于中间域，则没有标签信息。在本研究中，我们将对中间领域的预测视为 Side Information。利用其一起优化特征空间与分类器。

$$\mathcal{J}_2(f_c,f_e,f_d)=\frac{1}{n_S}\sum_{i=1}^{n_S}v_S^i\ell(y_S^i,f_c(\boldsymbol{h}_S^i))+\frac{1}{n_T}\sum_{i=1}^{n_T}\ell(y_T^i,f_c(\boldsymbol{h}_T^i))$$
$$+\frac{1}{n_I}\sum_{i=1}^{n_I}v_I^i g(f_c(\boldsymbol{h}_I^i)), \quad (2)$$

其中，g（）为交叉熵，g(z) = -z ln z - (1 - z) ln(1 - z)。

（4）总优化目标

$$\min_{\boldsymbol{\Theta},\boldsymbol{v}}\mathcal{J}=\mathcal{J}_1+\mathcal{J}_2, \quad \text{s.t. } v_S^i,v_I^i\in\{0,1\}, \quad (3)$$

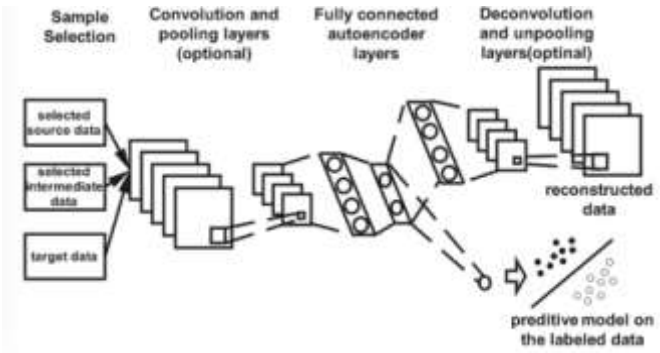
**Algorithm 1** The Selective Learning Algorithm (SLA)

1: **Input**: Data in $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{I}$, and parameters $\lambda_S$, $\lambda_I$, and $T$;
2: Initialize $\boldsymbol{\Theta}$, $\boldsymbol{v}_S=\mathbf{1}$, $\boldsymbol{v}_I=\mathbf{0}$; *// All source data are used*
3: **while** $t<T$ **do**
4: Update $\boldsymbol{\Theta}$ via the BP algorithm; *// Update the network*
5: Update $\boldsymbol{v}$ by Eqs. (4) and (5); *// Select "useful" instances*
6: $t=t+1$
7: **end while**
8: **Output**: $\boldsymbol{\Theta}$ and $\boldsymbol{v}$.

其中，通过（4）可以看出，在源域中的数据中，只有那些具有较低重构错误和低训练损失的实例才会在优化中选择，（5）同理。

$$v_S^i=\begin{cases}1 & \text{if } \ell(y_s^i,f_c(f_e(\boldsymbol{x}_S^i)))+\|\hat{\boldsymbol{x}}_S^i-\boldsymbol{x}_S^i\|_2^2<\lambda_S\\ 0 & \text{otherwise}\end{cases} \quad (4)$$

$$v_I^i=\begin{cases}1 & \text{if } \|\hat{\boldsymbol{x}}_I^i-\boldsymbol{x}_I^i\|_2^2+g(f_c(f_e(\boldsymbol{x}_I^i)))<\lambda_I\\ 0 & \text{otherwise}\end{cases} \quad (5)$$




## 23. Coupled Generative Adversarial Networks

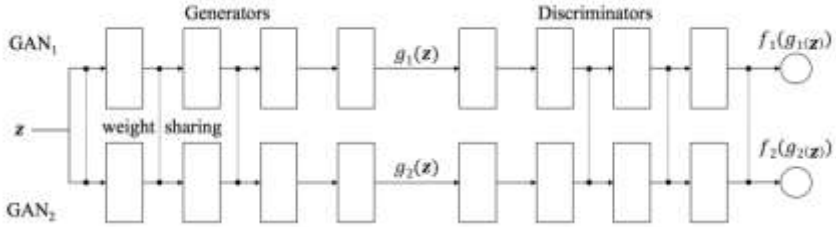
**主要思想：**

利用两个通过权值共享耦合的 GAN 网络，学习关于 multi-domain images 的联合分布，生成跨域样本。

**主要方法：**

CoGAN 的结构如图所示，由一对 GAN 构成，GAN1 和 GAN2 构成，每一个针对合成各自 domain 的图片。在训练过程中，我们让两个 GAN 共享一部分参数。

生成模型可以逐渐的从抽象的概念出来一点点解码出具体地细节出来。因此 generator 的前面几层解码的是高层的语义信息，比如目标的轮廓，而后面几层解码的是细节，比如边缘。由于我们需要保证高层信息的相关性，因此我们需要 g1 和 g2 的前几个层是共享权值的。后面几层的权值就不同了，这样可以根绝高层语义信息解码出不同的细节出来，用来 fool 各自的判别器。




（1）生成模型：

$$g_1(\mathbf{z}) = g_1^{(m_1)}(g_1^{(m_1-1)}(\dots g_1^{(2)}(g_1^{(1)}(\mathbf{z})))), \quad g_2(\mathbf{z}) = g_2^{(m_2)}(g_2^{(m_2-1)}(\dots g_2^{(2)}(g_2^{(1)}(\mathbf{z}))))$$

（2）判别模型：

$$f_1(\mathbf{x}_1) = f_1^{(n_1)}(f_1^{(n_1-1)}(\dots f_1^{(2)}(f_1^{(1)}(\mathbf{x}_1)))), f_2(\mathbf{x}_2) = f_2^{(n_2)}(f_2^{(n_2-1)}(\dots f_2^{(2)}(f_2^{(1)}(\mathbf{x}_2))))$$

（3）学习过程：

$$\max_{g_1,g_2}\min_{f_1,f_2} V(f_1,f_2,g_1,g_2), \text{ subject to } \begin{aligned} &\boldsymbol{\theta}_{g_1^{(i)}} = \boldsymbol{\theta}_{g_2^{(i)}}, \quad \text{for } i = 1,2,\dots,k \\ &\boldsymbol{\theta}_{f_1^{(n_1-j)}} = \boldsymbol{\theta}_{f_2^{(n_2-j)}}, \text{ for } j = 0,1,\dots,l-1 \end{aligned} \tag{2}$$

where the value function $V$ is given by

$$\begin{aligned} V(f_1,f_2,g_1,g_2) = &E_{\mathbf{x}_1\sim p_{\mathbf{X}_1}}[-\log f_1(\mathbf{x}_1)] + E_{\mathbf{z}\sim p_{\mathbf{Z}}}[-\log(1-f_1(g_1(\mathbf{z})))] \\ &+ E_{\mathbf{x}_2\sim p_{\mathbf{X}_2}}[-\log f_2(\mathbf{x}_2)] + E_{\mathbf{z}\sim p_{\mathbf{Z}}}[-\log(1-f_2(g_2(\mathbf{z})))]. \end{aligned} \tag{3}$$

（4）可以用在 domain adaption?

采集 2000mnist 图构成集合 D1，从 USPS 中随机选取 1800 图片构成 D2 数据集。

CoGAN 用来实现图像生成的任务。对于数字分类，我们在判别器的最后一个隐层上加上一个 softmax 层。于是 CoGAN 的训练需要完成两个任务，一个任务是 mnist 的数字分类问题，D1 提供了图片和对应的 label。还有就是 D1 和 D2 上的图片生成问题。针对每一个 domain 定义有一个分类器:对于 MNIST,分类器是:$c_1(\mathbf{x}_1) \equiv c(f_1^{(3)}(f_1^{(2)}(f_1^{(1)}(\mathbf{x}_1))))$和 USPS 分类器 $c_2(\mathbf{x}_2) \equiv c(f_2^{(3)}(f_2^{(2)}(f_2^{(1)}(\mathbf{x}_2))))$ 因为权值共享$f_1^{(2)} = f_2^{(2)}$ and $f_1^{(3)} = f_2^{(3)}$ c 是 sofxmax 层，最后采用 c2 去判别 USPS 分类问题。

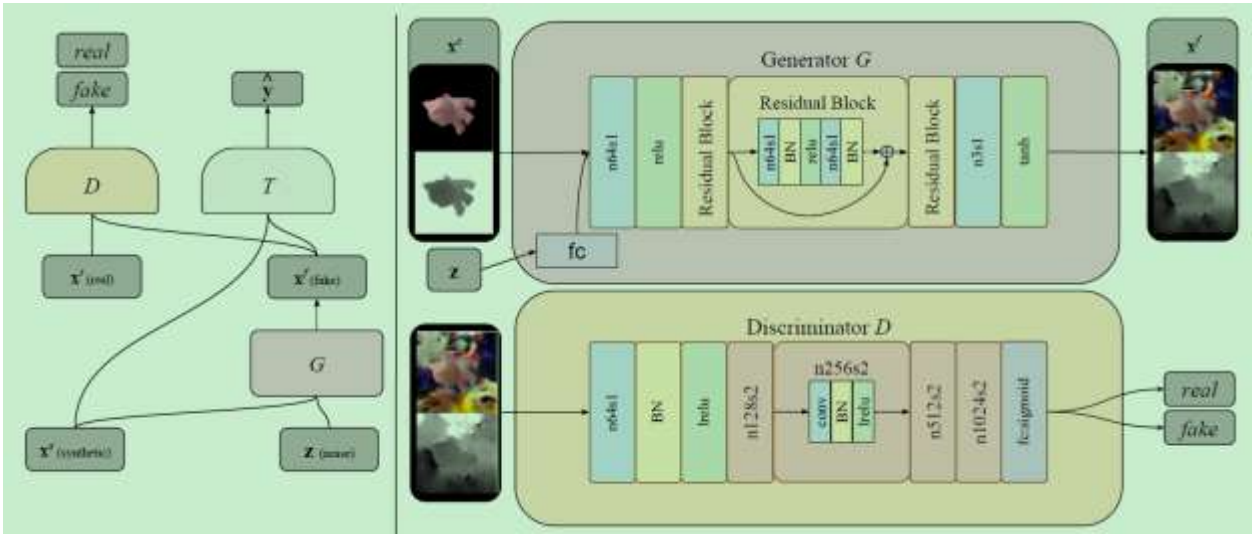
## 24. Unsupervised Pixel–Level Domain Adaptation with Generative Adversarial Networks

**主要思想：**

- 传统 unsupervised domain adaptation 在两个域之间对表征进行映射或者学习 domain-invariant 特征。
- 基于生成对抗网络（GAN）的方法能够使源域（source-domain）图像看起来就像是来自目标域（target domain）的一样。
- 解耦了领域自适应和类别判别过程，从而获得 Task-Specific 的架构

**主要方法：**

该方法将领域适应过程从特定任务的分类过程中分离出来。首先从源域生成图像，使他们看起来好像他们是从目标域采样。一旦生成器训练完成了，任何现成的可训练的分类器即可完成任务，没有领域适应需要。




- 源图像为 Xs 和噪声向量 z 生成适应图像$X^f$, $\mathrm{G}(X^s, \mathrm{z}; \theta_G) \rightarrow x^f$
- 鉴别器$\mathrm{D}(\mathrm{x}; \theta_D)$ 甄别真假图像
- 将$x^f$和 Xs 输入分类器$\mathrm{T}(\mathrm{x}; \theta_T)$，从而判断类别（实验证明同时输入$x^f$和 Xs 比只输入$x^f$效果好）
- n64s1 代表 stride 1 and 64 channels 的卷积； relu 代表非线性激活函数 ReLU；BN 是 batch normalization layer；FC 为全连接层。

最终优化目标：

$$\min_{\theta_G,\theta_T}\max_{\theta_D}\ \alpha\mathcal{L}_d(D,G)+\beta\mathcal{L}_t(G,T) \quad (1)$$

其中$L_d(D,G)$是领域自适应损失：

$$\mathcal{L}_d(D,G)=\mathbb{E}_{\mathbf{x}^t}[\log D(\mathbf{x}^t;\boldsymbol{\theta}_D)]+ \mathbb{E}_{\mathbf{x}^s,\mathbf{z}}[\log(1-D(G(\mathbf{x}^s,\mathbf{z};\boldsymbol{\theta}_G);\boldsymbol{\theta}_D))] \quad (2)$$

$L_t(G,T)$softmax 交叉熵损失：

$$\mathcal{L}_t(G,T)=\mathbb{E}_{\mathbf{x}^s,\mathbf{y}^s,\mathbf{z}}\big[-\mathbf{y}^{s\top}\log T(G(\mathbf{x}^s,\mathbf{z};\boldsymbol{\theta}_G);\boldsymbol{\theta}_T) -\mathbf{y}^{s\top}\log T(\mathbf{x}^s);\boldsymbol{\theta}_T\big] \quad (3)$$

1) 固定生成器参数$\theta_G$，更新鉴别器和分类器参数$\theta_T$和$\theta_D$

2) 固定鉴别器和分类器参数$\theta_T$和$\theta_D$，生成器参数$\theta_G$

- **Content－similarity loss**

为保证，前景相似，背景不同，加入 Content－similarity loss，优化 masked-PMSE 损失：

$$\min_{\theta_G,\theta_T}\max_{\theta_D}\ \alpha\mathcal{L}_d(D,G)+\beta\mathcal{L}_t(T,G)+\gamma\mathcal{L}_c(G) \quad (4)$$

$$\mathcal{L}_c(G)=\mathbb{E}_{\mathbf{x}^s,\mathbf{z}}\Big[\frac{1}{k}\|(\mathbf{x}^s-G(\mathbf{x}^s,\mathbf{z};\boldsymbol{\theta}_G))\circ\mathbf{m}\|_2^2 -\frac{1}{k^2}\left((\mathbf{x}^s-G(\mathbf{x}^s,\mathbf{z};\boldsymbol{\theta}_G))^\top\mathbf{m}\right)^2\Big] \quad (5)$$

## 25. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

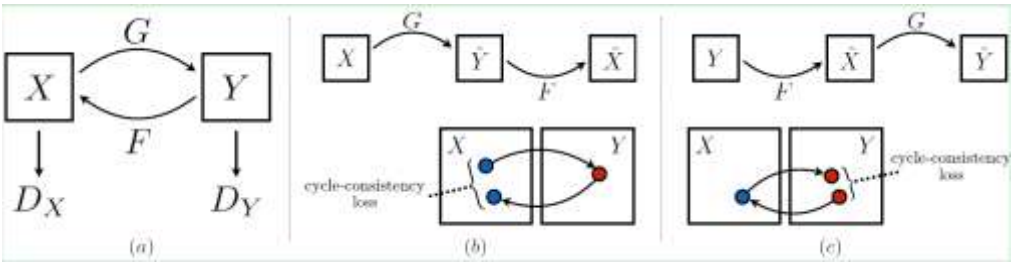
CycleGAN, DualGAN, DiscoGAN:

**主要思想（CycleGAN）：**

- image-to-image translation 中一对一的训练数据较难获得，
- 提出了一种在没有配对的情况下从来源域 X 到目标域 Y 进行图像转换的方式。
- 以 G:X→ Y 的方式建立映射，其中 G(X) 的图像分布与使用对抗性损失分布的 Y 难以区分
- 引入循环一致性损失函数来推动 F(G(X))≈X（反之亦然）。

**主要方法：**

CycleGAN 需要训练两个网络 G:X→Y 和 F:Y→X。相对地，有两个对抗损失 DX 和 DY，其中 DX 旨在区别出图像{x}和生成的{F(y)}，DY 同理。训练的目标分为两项：对抗损失是为了生成符合分布的图像，循环一致性损失是为了防止学习的 G 和 F 相互矛盾。




（1）Adversarial Loss

映射为 F。它就对应着 GAN 中的 生成器 ，F 可以将 X 中的图片 x 转换为 Y 中的图片 F(x)。对于生成的图片，我们还需要 GAN 中的 判别器 来判别它是否为真实图片，由此构成对抗生成网络。设这个判别器为$D_Y$。这样的话，根据这里的生成器和判别器，我们就可以构造一个 GAN 损失，表达式为：

$$\mathcal{L}_{\text{GAN}}(G,D_Y,X,Y)=\mathbb{E}_{y\sim p_{\text{data}}(y)}[\log D_Y(y)] +\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(1-D_Y(G(x))], \quad (1)$$

$$\mathrm{G}^*=argmin_G max_{D_y} L_{GAN}(G;D_Y;X;Y)，\ \mathrm{F}^*=argmin_F max_{D_X} L_{GAN}(G;D_X;X;Y)$$

（2）Cycle Consistency Loss

CycleGAN 同时学习 F 和 G 两个映射，并要求F(G(y)) ≈ y以及G(F(x)) ≈ x也就是说，将 X 的图片转换到 Y 空间后，应该还可以转换回来。这样就杜绝模型把所有 X 的图片都转换为 Y 空间中的同一张图片了

$$\mathcal{L}_{cyc}(G,F) = \mathbb{E}_{x\sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim p_{data}(y)}[\|G(F(y)) - y\|_1]. \quad (2)$$

（3）Full Objective

$$\mathcal{L}(G,F,D_X,D_Y) = \mathcal{L}_{GAN}(G,D_Y,X,Y) + \mathcal{L}_{GAN}(F,D_X,Y,X) + \lambda\mathcal{L}_{cyc}(G,F), \quad (3)$$

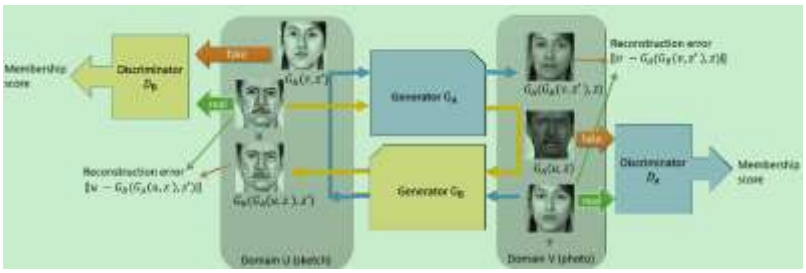
$$G^*,F^* = \arg\min_{F,G}\max_{D_X,D_Y}\mathcal{L}(G,F,D_X,D_Y). \quad (4)$$

## 26. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation

**主要思想（Dual GAN）：**

- 与 cycle GAN 思想相同。
- 生成器和判别器都和 pix2pix 一样（没有随机输入 z，但是有 dropout 的随机，而 cycle GAN 没有 dropout）。用了 wgan 来训练。cycle consistency 同样选用了 l1 距离。

**主要方法：**




$$l_A^d(u,v) = D_A(G_A(u,z)) - D_A(v), \quad (1)$$
$$l_B^d(u,v) = D_B(G_B(v,z')) - D_B(u), \quad (2)$$

$$l^g(u,v) = \lambda_U\|u - G_B(G_A(u,z),z')\| + \lambda_V\|v - G_A(G_B(v,z'),z)\| - D_A(G_B(v,z')) - D_B(G_A(u,z)), \quad (3)$$

- **WGAN**

原始的生成对抗网络，所要优化的目标函数为：

$$\min_G\max_D V(D,G) = E_{x\sim P_r}\log(D(x)) + E_{z\sim P_z(z)}[\log(1-D(G(z)))]$$

出现问题：判别器越好，生成器梯度消失越严重。

基于 Wasserstein 距离改进：

$$L = E_{x\sim P_r}[D(x)] - E_{x\sim P_g}[D(x)]$$

WGAN 的算法改动：

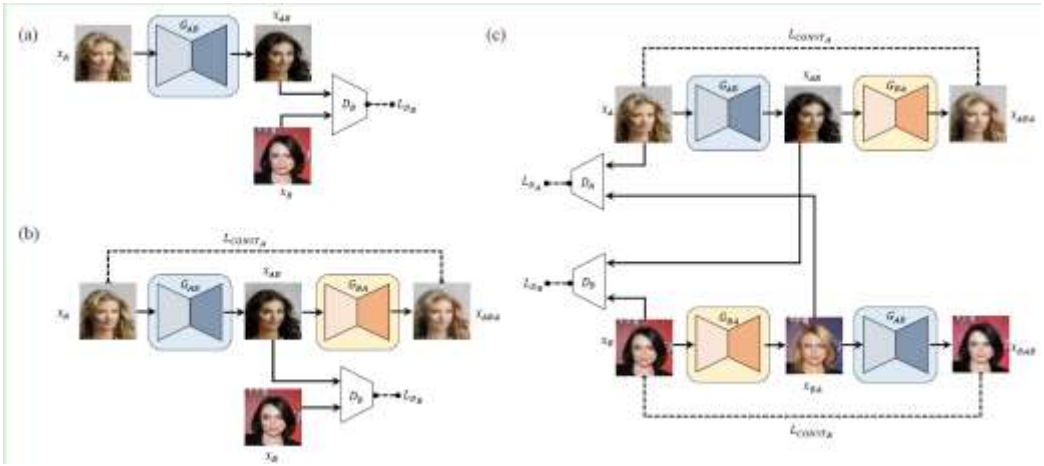
- 判别器最后一层去掉 sigmoid
- 生成器和判别器的 loss 不取 log
- 每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数 c
- 不要用基于动量的优化算法（包括 momentum 和 Adam），推荐 RMSProp，SGD 也行

## 27. Learning to Discover Cross-Domain Relationswith Generative Adversarial Networks

**主要思想（Disco GAN）：**

- 与 cycle GAN 和 Dual GAN 思想相同。
- 用了 conv,deconv 和 leaky relu 组成了生成器,然后一个 conv+leaky relu 作为判别器。他们用 l2 作为 cycle consistency。

**主要方法：**



$$x_{AB} = \mathbf{G}_{AB}(x_A) \quad (1)$$

$$x_{ABA} = \mathbf{G}_{BA}(x_{AB}) = \mathbf{G}_{BA} \circ \mathbf{G}_{AB}(x_A) \quad (2)$$

$$L_{CONST_A} = d(\mathbf{G}_{BA} \circ \mathbf{G}_{AB}(x_A), x_A) \quad (3)$$

$$L_{GAN_B} = -\mathbb{E}_{x_A \sim P_A}[\log \mathbf{D}_B(\mathbf{G}_{AB}(x_A))] \quad (4)$$

$$L_{D_B} = -\mathbb{E}_{x_B \sim P_B}[\log \mathbf{D}_B(x_B)] - \mathbb{E}_{x_A \sim P_A}[\log(1 - \mathbf{D}_B(\mathbf{G}_{AB}(x_A)))] \quad (6)$$

$$L_G = L_{G_{AB}} + L_{G_{BA}} = L_{GAN_B} + L_{CONST_A} + L_{GAN_A} + L_{CONST_B} \quad (7)$$

$$L_D = L_{D_A} + L_{D_B} \quad (8)$$

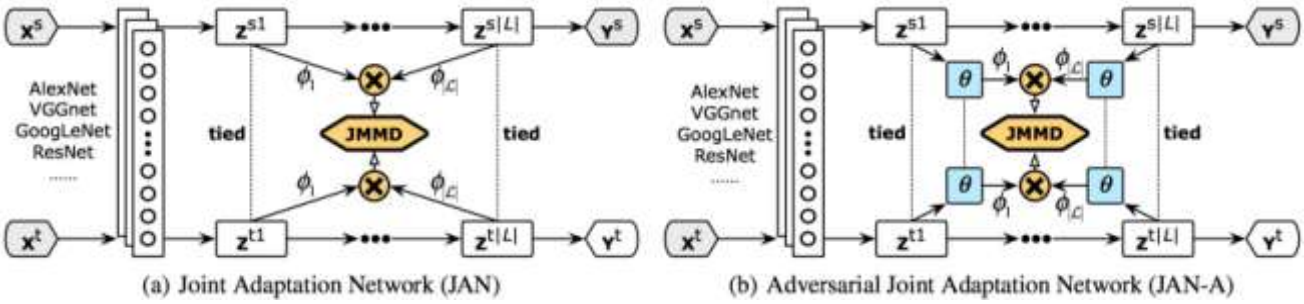
## 28. Deep Transfer Learning with Joint Adaptation Networks

**主要思想：**

- 之前的方法都调整边缘分布，并假设条件分布不改变，但没有研究联合分布 P(x,y)
- 联合分布的变化由边缘分布、条件分布、或者两者同时引起的
- 减少多个 task-specificl 层的联合分布 shift，通过这个优化操作近似减少输入特征和输出标签联合分布的变化。

**主要方法：**

JAN 使用一个 JMMD 来减少多个任务特定层的联合分布 shift，这反映了输入特性和输出标签的联合分布的变化;DAN 假设特征层和分类器层是独立的，需要多个 MMD，每个层都独立地减少每个层的边际分布差异。




(a) Joint Adaptation Network (JAN) (b) Adversarial Joint Adaptation Network (JAN-A)

- Hilbert Space Embedding

采用张量积 RKHS，Kernel embeddings 可以很容易地推广到两个或两个以上变量的联合分布：

$$\widehat{\mu}_{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i). \quad \widehat{C}_{\mathbf{X}^{1:m}} = \frac{1}{n}\sum_{i=1}^{n}\otimes_{\ell=1}^{m}\phi^{\ell}(\mathbf{x}_i^{\ell}). \quad (4)$$

其中，$\otimes_{\ell=1}^{m}\phi^{\ell}(\mathbf{x}^{\ell}) = \phi^{1}(\mathbf{x}^{1})\otimes\ldots\otimes\phi^{m}(\mathbf{x}^{m})$ $\langle\otimes_{\ell=1}^{m}\phi^{\ell}(\mathbf{x}^{\ell}), \otimes_{\ell=1}^{m}\phi^{\ell}(\mathbf{x}'^{\ell})\rangle = \prod_{\ell=1}^{m}k^{\ell}(\mathbf{x}^{\ell}, \mathbf{x}'^{\ell})$表示张量积。

- Joint Adaptation Networks

标准的 CNN 的深层特性最终从一般层转移到特定层，跨域的差异增加，特性和分类器的可转移性降低。则将源和目标域数据通过深层网络进行多层特征抽象，即联合分布$P(X^s,Y^s)$和$Q(X^t,Y^t)$间差异依然存在于高层$Z^1\cdots\cdots Z^l$。其中，$P(X^s)$和$Q(X^t)$特征差异存在于 fc6 和 fc7，而$P(Y^s)$和$Q(Y^t)$标签差异存在于 fc8。因此我们应该找到方法匹配$P(\mathbf{Z}^{s1},\ldots,\mathbf{Z}^{s|\mathcal{L}|})$ and $Q(\mathbf{Z}^{t1},\ldots,\mathbf{Z}^{t|\mathcal{L}|})$.

- Joint Maximum Mean Discrepancy

$$D_{\mathcal{L}}(P,Q) \triangleq \|\mathcal{C}_{\mathbf{Z}^{s,1:|\mathcal{L}|}}(P) - \mathcal{C}_{\mathbf{Z}^{t,1:|\mathcal{L}|}}(Q)\|^2_{\otimes_{\ell=1}^{|\mathcal{L}|}\mathcal{H}^\ell}. \quad (8)$$

其无偏估计为：

$$\begin{aligned}\widehat{D}_{\mathcal{L}}(P,Q) = &\frac{1}{n_s^2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}\prod_{\ell\in\mathcal{L}}k^\ell\left(\mathbf{z}_i^{s\ell},\mathbf{z}_j^{s\ell}\right)\\ &+\frac{1}{n_t^2}\sum_{i=1}^{n_t}\sum_{j=1}^{n_t}\prod_{\ell\in\mathcal{L}}k^\ell\left(\mathbf{z}_i^{t\ell},\mathbf{z}_j^{t\ell}\right) \quad (9)\\ &-\frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t}\prod_{\ell\in\mathcal{L}}k^\ell\left(\mathbf{z}_i^{s\ell},\mathbf{z}_j^{t\ell}\right).\end{aligned}$$

**1） JAN**

优化函数为：

$$\min_f \frac{1}{n_s}\sum_{i=1}^{n_s}J\left(f\left(\mathbf{x}_i^s\right),\mathbf{y}_i^s\right)+\lambda\widehat{D}_{\mathcal{L}}(P,Q), \quad (10)$$

其中，$\min_f \frac{1}{n_s}\sum_{i=1}^{n_s}J\left(f\left(\mathbf{x}_i^s\right),\mathbf{y}_i^s\right),$ (7) 是交叉熵。

**2） JAN-A**

一些被广泛使用的内核无法在高维空间中捕捉到非常复杂的距离。因此采用对抗的思想，引入参数为θ全连接层，通过优化参数来最大化两个域联合分布的距离。

$$\min_f \max_\theta \frac{1}{n_s}\sum_{i=1}^{n_s}J\left(f\left(\mathbf{x}_i^s\right),\mathbf{y}_i^s\right)+\lambda\widehat{D}_{\mathcal{L}}(P,Q;\theta). \quad (12)$$

## 29. Return of Frustratingly Easy Domain Adaptation

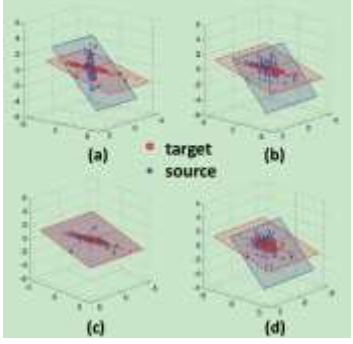
**主要思想：**

CORAL minimizes domain shiftby aligning the second-order statistics of source and targetdistributions, without requiring any target labels

**主要方法：**

- computing covariance statistics in each domain
- applyingthe whitening and re-coloring linear transformationto the source features.

（a）尽管特征被归零均值和标准偏差，源域和目标域仍然具有不同分布的协方差（b）source decorrelation，源域数据解相关后的两个域（c）Target re-correlation，将目标域的相关性添加到源特性。在这一步之后，源和目标分布很好地对齐，并且在调整后的源域上训练的分类器在目标域中工作得很好。（d）试图通过同时白化源和目标数据来调整分布。




为尽量减少源和目标数据二阶统计之间的距离(协方差),对源数据使用矩阵 A 进行线性变换,并采用 Frobenius 范数为距离度量：

$$\min_A \|C_S - C_T\|_F^2 = \min_A \|A^\top C_S A - C_T\|_F^2 \quad (1)$$

$$A^* = U_S E = (U_S \Sigma_S^{+\frac{1}{2}} U_S^\top)(U_{T[1:r]} \Sigma_{T[1:r]}^{\frac{1}{2}} U_{T[1:r]}^\top). \quad (2)$$

We can think of transformation $A$ in this way intuitively: thefirst part $(U_S \Sigma_S^{+\frac{1}{2}} U_S^\top)$ whitens the source data while the secondpart $(U_{T[1:r]} \Sigma_{T[1:r]}^{\frac{1}{2}} U_{T[1:r]}^\top)$ re-colors it with the targetcovariance.

传统的白化是协方差矩阵的对角元素添加一个小正则化参数 λ 让他达到满秩，然后乘以原始特征的逆平方根。

**Algorithm 1** CORAL for Unsupervised Domain Adaptation

**Input:** Source Data $D_S$, Target Data $D_T$
**Output:** Adjusted Source Data $D_S^*$
$C_S = cov(D_S) + eye(size(D_S, 2))$
$C_T = cov(D_T) + eye(size(D_T, 2))$
$D_S = D_S * C_S^{\frac{-1}{2}}$ % whitening source
$D_S^* = D_S * C_T^{\frac{1}{2}}$ % re-coloring with target covariance

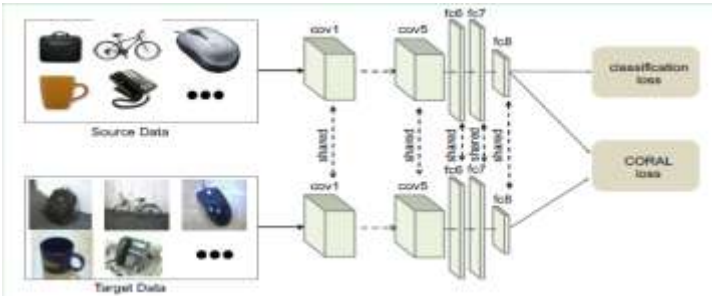
## 30. Deep CORAL: Correlation Alignment for DeepDomain Adaptation

**主要思想：**

- 基于 CORAL 方法，它是一种简单的无监督域自适应方法，通过线性变换对齐源域和目标域分布的二阶统计量
- 直接将 CORAL 引入深度学习，利用 CORAL loss 减少特征协方差的跨域距离，类似于 MMD

**主要方法：**

首先，初始化预先训练好的网络的参数，并利用有标记的源数据进行微调。然后，利用 CORAL 损失最小化源域和目标域特征值协方差的二阶统计。




优化函数为：

$$\mathcal{L} = \mathcal{L}_{CLASS} + \sum_{i=1}^{t} \lambda_i \mathcal{L}_{CORAL} \quad (6)$$

其中 t 为应用 CORAL loss 的层数，此文中只为 fc8。CORAL 损失定义如下：

$$\mathcal{L}_{CORAL} = \frac{1}{4d^2} \|C_S - C_T\|_F^2 \quad (1)$$

$C_S$和$C_T$为源域与目标域特征值的协方差矩阵，$\|.\|_F^2$是 squared matrix Frobenius norm。$D_S^{i,j}$($D_T^{i,j}$)是第 i 个源（目标）数据的 j 维元素。

$$C_S = \frac{1}{n_S - 1}(D_S^\top D_S - \frac{1}{n_S}(\mathbf{1}^\top D_S)^\top (\mathbf{1}^\top D_S)) \quad (2)$$

$$C_T = \frac{1}{n_T - 1}(D_T^\top D_T - \frac{1}{n_T}(\mathbf{1}^\top D_T)^\top (\mathbf{1}^\top D_T)) \quad (3)$$

CORAL 损失的导数为（可利用链式规则和梯度下降进行反向传播）

$$\frac{\partial \mathcal{L}_{CORAL}}{\partial D_S^{ij}} = \frac{1}{d^2(n_S - 1)}((D_S^\top - \frac{1}{n_S}(\mathbf{1}^\top D_S)^\top \mathbf{1}^\top)^\top (C_S - C_T))^{ij} \quad (4)$$

$$\frac{\partial \mathcal{L}_{CORAL}}{\partial D_T^{ij}} = -\frac{1}{d^2(n_T - 1)}((D_T^\top - \frac{1}{n_T}(\mathbf{1}^\top D_T)^\top \mathbf{1}^\top)^\top (C_S - C_T))^{ij} \quad (5)$$

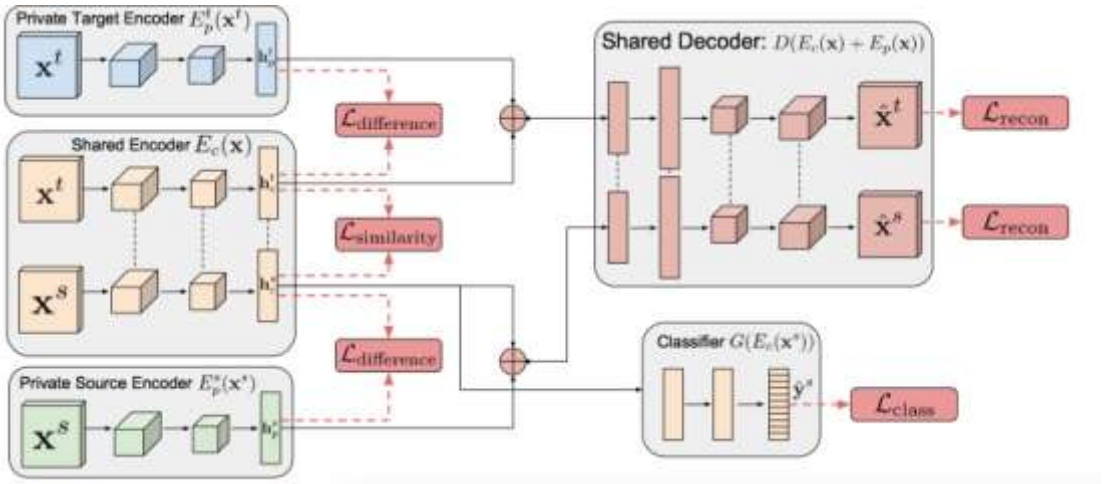
## 31. Domain Separation Networks

**主要思想：**

- 提取特征空间时，将其划分为两个子空间:一个是每个域私有的部分，另一个是跨域共享的部分
- 每个域的私有子空间，它捕获特定域的属性，例如背景和低层次的图像统计信息。通过使用自动编码器和重构损失函数捕获域共享的表示。
- 共享子空间与独有子空间正交

**主要方法：**

一个共享权重的编码器 E c(x)学习源域和目标域样本共享特征。一个私有编码器 E p(x)(每个域的一个)学习捕获每个领域特定的特征。一个共享权重的解码器通过使用私有和共有的特征重新构造源域和目标域的输入样本。




其中，$E_c(x;\theta_c)$是参数为$\theta_c$的编码器，提取输入图像 x 的共享高层特征。$E_p(x;\theta_p)$是参数为$\theta_p$的编码器，提取输入图像 x 的特有的高层特征。$\mathrm{D}(h;\theta_d)$是参数为$\theta_d$的解码器，将高层特征 h 重构为输入图像 x。$\mathrm{G}(h;\theta_g)$是参数为$\theta_g$的分类器，利用高层特征 h 得到标签 y。

- 优化函数

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \alpha\,\mathcal{L}_{\text{recon}} + \beta\,\mathcal{L}_{\text{difference}} + \gamma\,\mathcal{L}_{\text{similarity}}$$

其中

$$\mathcal{L}_{\text{task}} = -\sum_{i=0}^{N_s} \mathbf{y}_i^s \cdot \log \hat{\mathbf{y}}_i^s,$$

$$\mathcal{L}_{\text{recon}} = \sum_{i=1}^{N_s} \mathcal{L}_{\text{si\_mse}}(\mathbf{x}_i^s, \hat{\mathbf{x}}_i^s) + \sum_{i=1}^{N_t} \mathcal{L}_{\text{si\_mse}}(\mathbf{x}_i^t, \hat{\mathbf{x}}_i^t) \tag{3}$$

$$\mathcal{L}_{\text{si\_mse}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{k}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 - \frac{1}{k^2}([\mathbf{x} - \hat{\mathbf{x}}] \cdot \mathbf{1}_k)^2, \tag{4}$$

使用尺度不变的均值平方误差来表示重建损失，k 是像素的数量，1k 是长度是 k 的单位 1 向量。虽然平均平方误差损失在传统上用于重建任务，但它无法对于尺度变化做出惩罚。

$$L_{\text{difference}} = \left\|\mathbf{H}_c^{s\top}\mathbf{H}_p^s\right\|_F^2 + \left\|\mathbf{H}_c^{t\top}\mathbf{H}_p^t\right\|_F^2, \tag{5}$$

差异损失鼓励了每个域的共享和私有表示之间的正交性。

对于$\mathcal{L}_{similarity}$ 有两种形式：

一种是利用生成式对抗网络的思想：

$$\mathcal{L}_{\text{similarity}}^{\text{DANN}} = \sum_{i=0}^{N_s+N_t} \left\{ d_i \log \hat{d}_i + (1-d_i)\log(1-\hat{d}_i) \right\}. \tag{6}$$

另一种是利用 MMD 的思想：

$$\mathcal{L}_{\text{similarity}}^{\text{MMD}} = \frac{1}{(N^s)^2}\sum_{i,j=0}^{N^s} \kappa(\mathbf{h}_{ci}^s, \mathbf{h}_{cj}^s) - \frac{2}{N^s N^t}\sum_{i,j=0}^{N^s,N^t} \kappa(\mathbf{h}_{ci}^s, \mathbf{h}_{cj}^t) + \frac{1}{(N^t)^2}\sum_{i,j=0}^{N^t} \kappa(\mathbf{h}_{ci}^t, \mathbf{h}_{cj}^t), \tag{7}$$

其中利用生成时对抗网络比 MMD 效果要好，而且实验证明 MMD 只需要一层就行，多层对于效果没有提升。

## 32. Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation

**主要思想（DRCN）：**

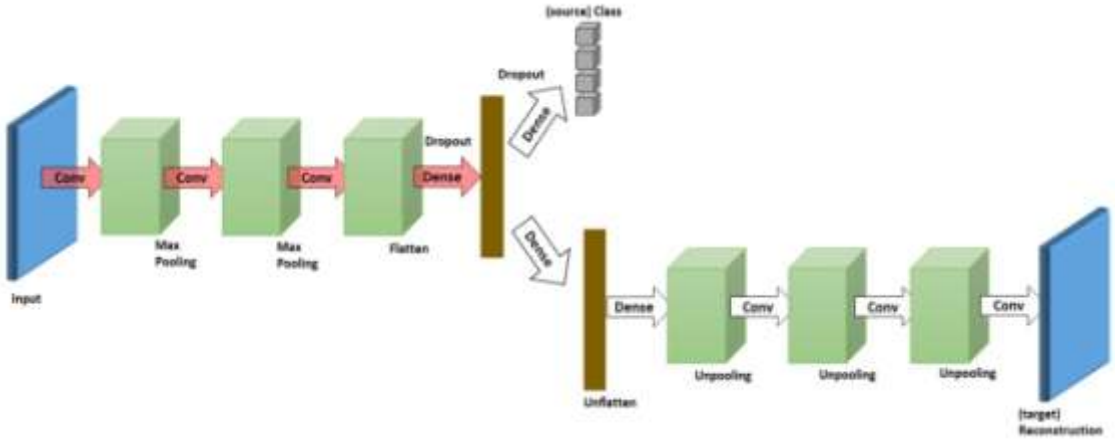
利用卷积神经网络完成两个任务：

- 监督源标签预测
- 无监督目标数据重建

DRCN 的编码参数是在两个任务之间共享的，而解码参数是独立的。其目的是，学习的标签预测函数能够很好地对目标域中的图像进行分类，因此数据重建可以被视为辅助任务，以支持对标签预测的适应。

**主要方法：**

我们假设一个域自适应表示应该满足两个标准：i)对标记数据进行分类 ii)重构目标域未标记的数据。模型是基于一个卷积的架构，有两条带有共享权重的管道。第一个管道是标准的卷积网络，用于源标签预测，而第二个是用于目标数据重建的卷积自动编码器。




$$f_c(x) = (g_{\text{lab}} \circ g_{\text{enc}})(x), \quad (1)$$
$$f_r(x) = (g_{\text{dec}} \circ g_{\text{enc}})(x). \quad (2)$$

其中，fc 表示源标签预测，fr 表示目标数据重建。编码器特征映射为$g_{enc}:\mathcal{X} \to \mathcal{F}$，解码器$g_{dec}:\mathcal{F} \to \mathcal{X}$，标签预测$g_{\text{lab}}:\mathcal{F} \to \mathcal{Y}$则优化目标为：

$$\min \lambda \mathcal{L}_c^{n_s}(\{\Theta_{\text{enc}}, \Theta_{\text{lab}}\}) + (1-\lambda)\mathcal{L}_r^{n_t}(\{\Theta_{\text{enc}}, \Theta_{\text{dec}}\}), \quad (5)$$

其中，

$$\mathcal{L}_c^{n_s}(\{\Theta_{\text{enc}}, \Theta_{\text{lab}}\}) := \sum_{i=1}^{n_s} \ell_c\left(f_c(\mathbf{x}_i^s; \{\Theta_{\text{enc}}, \Theta_{\text{lab}}\}), \mathbf{y}_i^s\right), \quad (3)$$

$$\mathcal{L}_r^{n_t}(\{\Theta_{\text{enc}}, \Theta_{\text{dec}}\}) := \sum_{j=1}^{n_t} \ell_r\left(f_r(\mathbf{x}_j^t; \{\Theta_{\text{enc}}, \Theta_{\text{dec}}\}), \mathbf{x}_j^t\right). \quad (4)$$

lc 是交叉熵$\sum_{k=1}^{m} y_k \log[f_c(\mathbf{x})]_k$，lr 是误差$\|\mathbf{x} - f_r(\mathbf{x})\|_2^2$.

**Input:**
• Labeled source data: $S^s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$;
• Unlabeled target data: $S_u^t = \{\mathbf{x}_j^t\}_{i=j}^{n_t}$;
• Learning rates: $\alpha_c$ and $\alpha_r$;

1: Initialize parameters $\Theta_{\text{enc}}, \Theta_{\text{dec}}, \Theta_{\text{lab}}$
2: **while** not stop **do**
3: **for each** source batch of size $m_s$ **do**
4: Do a forward pass according to (1);
5: Let $\Theta_c = \{\Theta_{\text{enc}}, \Theta_{\text{lab}}\}$. Update $\Theta_c$:

$$\Theta_c \leftarrow \Theta_c - \alpha_c \lambda \nabla_{\Theta_c} \mathcal{L}_c^{m_s}(\Theta_c);$$

6: **end for**
7: **for each** target batch of size $m_t$ **do**
8: Do a forward pass according to (2);
9: Let $\Theta_r = \{\Theta_{\text{enc}}, \Theta_{\text{dec}}\}$. Update $\Theta_r$:

$$\Theta_r \leftarrow \Theta_r - \alpha_r (1-\lambda) \nabla_{\Theta_r} \mathcal{L}_r^{m_t}(\Theta_r).$$

10: **end for**
11: **end while**

**Output:**
• DRCN learnt parameters: $\hat{\Theta} = \{\hat{\Theta}_{\text{enc}}, \hat{\Theta}_{\text{dec}}, \hat{\Theta}_{\text{lab}}\}$;

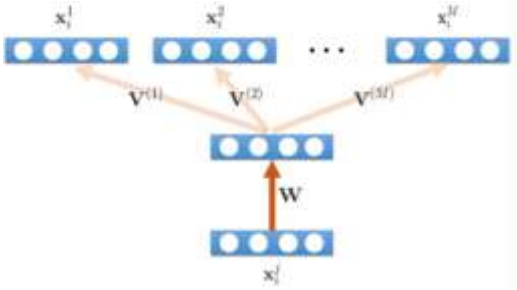
## 33. Domain Generalization for Object Recognition with Multi-task Autoencoders

**主要思想:**

- 引入反向传播的自动编码器，使用输入作为标签，并学习以最小的失真重构它们
- 联合学习从相关域获取的多个数据，并利用重建任务调整网络
- 多个源域之间是自然发生的转换，如：观察角度的旋转，物体的大小伸缩，和照明条件的变化等

**主要方法:**

多任务自动编码器体系结构由三个具有多层网络组成;每个输出对应一个任务或域。输入-隐藏的权重表示共享参数，而隐藏-输出权重表示特定领域的参数。这一体系结构与多任务神经网络类似。主要的区别在于，MTAE 的输出层对应于不同的域，而不是不同的类标签。




两种类型的重建任务在 MTAE 训练:1)自域重建和 2)域间重建。给定 M 源域，有 M*M 个重构任务，其中 M 任务是自域重构，剩下的 M*(M-1)任务是域间重构。

$$\begin{aligned} \bar{\mathbf{X}} &= [\mathbf{X}^1; \mathbf{X}^2; \ldots; \mathbf{X}^M], \\ \bar{\mathbf{X}}^l &= [\mathbf{X}^l; \mathbf{X}^l; \ldots; \mathbf{X}^l] \end{aligned} \tag{3}$$

其中，$\bar{\mathrm{X}}$是来自所有域的数据点的矩阵，而$\bar{\mathrm{X}}^l$是从第 l 域获取的复制数据集的矩阵。矩阵每列表示第 l 域的 dx 维数据。

the feed-forward MTAE reconstruction is

$$\begin{aligned} \mathbf{h}_i &= \sigma_{\text{enc}}(\mathbf{W}^\top \bar{\mathbf{x}}_i), \\ f_{\Theta^{(l)}}(\bar{\mathbf{x}}_i) &= \sigma_{\text{dec}}(\mathbf{V}^{(l)\top} \mathbf{h}_i), \end{aligned} \tag{4}$$

W 是共享编码器的权重，$V^{(l)}$是每个域特有的解码器的权重。

重构损失为:

$$J(\Theta^{(l)}) = \sum_{i=1}^{N} \mathcal{L}\left(f_{\Theta^{(l)}}(\bar{\mathbf{x}}_i), \bar{\mathbf{x}}_i^l\right). \tag{5}$$

$$\hat{\Theta}^{(l)} := \arg\min_{\Theta^{(l)}} \sum_{l=1}^{M} J(\Theta^{(l)}) + \eta \mathcal{R}(\Theta^{(l)}), \tag{6}$$

其中$\mathcal{R}$（$\Theta^{(l)}$）是正则项，等于 $\|\mathbf{W}\|_2^2+\|\mathbf{V}^{(l)}\|_2^2$.

## 34. Lightweight Unsupervised Domain Adaptationby Convolutional Filter Reconstruction

**主要思想：**

- 只使用少量的目标样本，分析和重构被 domain shift 影响的滤波器的输出。
- 重建的目的是使目标图像的滤波器响应类似于源图像的响应图。
- 使用基于 Lasso 的优化方法，同时选择和重建不良滤波器的响应图，并用 KL 散度指导滤波器选择过程。

**主要方法：**

第一层比后一层更容易受到 domain shift 的影响。

不同的过滤器之间的差异较大，而在后层的过滤器的 H-散度服从正态分布。

域适应问题应该从第一层开始，以便在正确的层次上纠正每个移位，而不是等到最后一层，然后尝试匹配这两个特征空间。

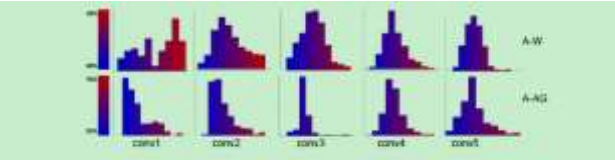


Fig. 1: The histograms of the filters' H-divergences for different layers.

- Divergence measure

采用 KL 散度衡量源域和目标域分布的距离，因为使用 H 散度需要训练的领域分类器，同时也需要有有足够的数据量。

- Filter selection

在同一步中选择一个受影响的滤波器并重建它。

$$B^* = argmin_B\{\sum_{i=1}^{n}(f_{y_i}-\beta_0-\sum_{j=1}^{P}f_{ij_i}\beta_j)^2+\lambda\sum_{j=1}^{P}|\Delta_j^{KL}\cdot\beta_j|\} \qquad (2)$$

$B^*$是滤波器的权重组，每个元素$\beta_j$对应一个滤波器$f_j$。如果滤波器有一个非零的重量，这意味着它是一种很好的选择，我们将保持它的价值。另一方面，如果滤波器具有零重量将被标记为重建，同时非零权重滤波器被用于此目的。

- Reconstruction

对应需要重建的滤波器 fb，利用给定的非零的滤波器的响应，我们使用线性回归方法预测滤波器的输出$f_{b_y}$，得到最终的$B_b$。

## 35. One-Shot Adaptation of Supervised DeepConvolutional Models

**主要思想：**

- 利用 CNN 提取特征，然后在使用传统的分类器和迁移方法进行分类和迁移
- 对比了使用 CNN 特征的非监督和监督的传统迁移方法
- 对比了采用不同层（DeCAF6，DeCAF7，DeCAF8）提取的特征

**主要方法：**

- 非监督

1） CA

通过最小化$\|UM-\hat{U}\|_F^2$，求得转化矩阵 M。求解得$M^*=U^T\hat{U}$。

- 监督

1） 混合法

最大混合：$v_{\text{adapt}}=\max(v_s,v_t)$，线性插值混合：$v_{\text{adapt}}=(1-\alpha)v_s+\alpha v_t$。其中 v 是分类器。

2） 数据增强

将源数据和目标数据强化为$x'=(x;x;0)$，$\hat{x}'=(\hat{x};0;\hat{x})$并采用 SVM 进行分类。

3） PMT

$$\mathcal{L}_{PMT}(\tilde{\theta}) = \frac{1}{2}\|\tilde{\theta}\|_2^2 + \frac{\Gamma}{2}\|\tilde{\theta}\|_2^2 \sin^2\alpha(\tilde{\theta},\theta) + \ell_{hinge}(\tilde{X},\tilde{Y};\tilde{\theta}) \ ,$$

，其中θ和θ'是SVM分类器参数

4) MMDT

$$\mathcal{L}_{MMDT}(\theta, A) = \frac{1}{2}\|\theta\|_2^2 + \frac{1}{2}\|A - I\|_F^2 + C_s\ell_{hinge}(X, Y;\theta) + C_t\ell_{hinge}(A\tilde{X}, \tilde{Y};\theta) \ , \quad (2)$$

，其中A为转换矩阵

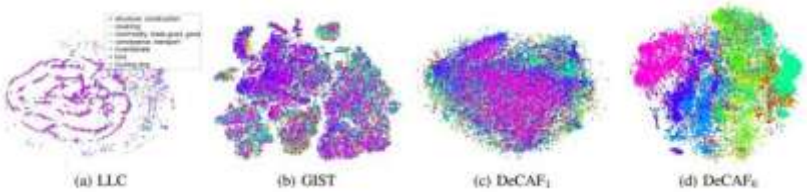
## 36. DeCAF: A deep convolutional activation feature for generic visual recognition

**主要思想：**

- 探究能否将一个由ImageNet（一个很大的数据集)训练来的卷积神经网络DeCAF泛化到其他可用数据较少的任务

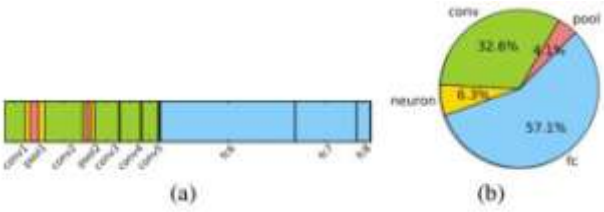
**主要结论：**

1) 使用起始层到中间的某层的网络作为特征提取。使用t-SNE来visualize特征的区分能力。DeCAF就是本文的方法，下标表示网络层数。可以看出深层的CNN特征在区分特征的能力上要远胜于手工特征(LLC和GIST)和浅层CNN特征。



(a) LLC (b) GIST (c) $\text{DeCAF}_1$ (d) $\text{DeCAF}_6$

2) 时间上说，全连接层占用的时间最长，卷积其次，应该是因为这两者牵扯到矩阵乘法。而pooling和RELU之类的开销则非常小，如下图：




(a) (b)

3) 使用不同层的CNN+logit回归或者SVM， 6层时的效果最好，分类器的话线性SVM和Logit回归的效果差不多。

| | $\text{DeCAF}_5$ | $\text{DeCAF}_6$ | $\text{DeCAF}_7$ |
|---|---|---|---|
| LogReg | $63.29 \pm 6.6$ | $84.30 \pm 1.6$ | $84.87 \pm 0.6$ |
| LogReg with Dropout | - | $86.08 \pm 0.8$ | $85.68 \pm 0.6$ |
| SVM | $77.12 \pm 1.1$ | $84.77 \pm 1.2$ | $83.24 \pm 1.2$ |
| SVM with Dropout | - | $\mathbf{86.91 \pm 0.7}$ | $85.51 \pm 0.9$ |
| Yang et al. (2009) | | 84.3 | |
| Jarrett et al. (2009) | | 65.5 | |

## 37. Rich feature hierarchies for accurate object detection and semantic segmentation

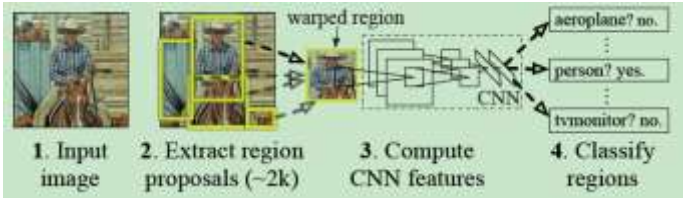
**主要思想：**

采用了迁移学习的思想。较大的**识别库**（ImageNet ILSVC 2012)：标定每张图片中物体的类别。一千万图像，1000类。→较小的**检测库**（PASCAL VOC 2007)：标定每张图片中，物体的类别和位置。一万图像，20类。

**主要方法：**

RCNN算法分为4个步骤

- 一张图像生成1K~2K个**候选区域**
- 对每个候选区域，使用深度网络**提取特征**
- 特征送入每一类的SVM **分类器**，判别是否属于该类
- 使用回归器**精细修正**候选框位置




（1）候选区域

使用了 Selective Search 方法从一张图像生成约 2000-3000 个候选区域。基本思路如下：

- 使用一种过分割手段，将图像分割成小区域
- 查看现有小区域，合并**可能性最高**的两个区域。重复直到整张图像合并成一个区域位置
- 输出所有曾经存在过的区域，所谓候选区域

（2）提取特征

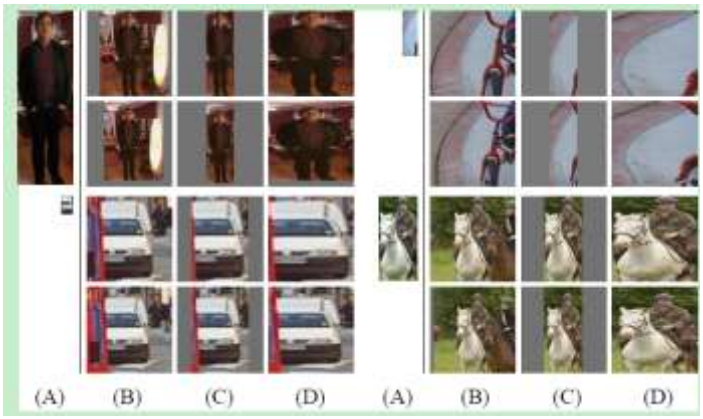
1） 预处理

使用深度网络提取特征之前，首先把候选区域归一化成同一尺寸 227×227。

- 各向异性缩放：不管图片的长宽比例，管它是否扭曲，进行缩放就是了，全部缩放到 CNN 输入的大小 227*227，如下图(D)所示；
- 各向同性缩放：图片扭曲后，估计会对后续 CNN 的训练精度有影响，这个有两种办法

A、直接在原始图片中，把 bounding box 的边界进行扩展延伸成正方形，然后再进行裁剪；如果已经延伸到了原始图片的外边界，那么就用 bounding box 中的颜色均值填充；如下图(B)所示；

B、先把 bounding box 图片裁剪出来，然后用固定的背景颜色填充成正方形图片(背景颜色也是采用 bounding box 的像素颜色均值),如下图(C)所示；




2） 标记样本

人工标注的数据一张图片中就只标注了正确的 bounding box，搜索出来的 2000 个矩形框也不可能会出现一个与人工标注完全匹配的候选框。因此我们需要用 IOU 为 2000 个 bounding box 打标签，以便下一步 CNN 训练使用。在 CNN 阶段，如果用 selective search 挑选出来的候选框与物体的人工标注矩形框的重叠区域 IoU 大于 0.5，那么我们就把这个候选框标注成物体类别（正例），否则我们就把它当做背景类别（负例）。IOU 定义如下：


$$IOU = (A \cap B)/(A \cup B)$$

3） 有监督预训练

直接用 Alexnet 的网络做图像分类的预训练，作为初始的参数值，然后再 fine-tuning 训练。

4） fine-tuning 训练

采用 selective search 搜索出来的候选框，然后处理到指定大小图片，继续对上面预训练的 cnn 模型进行 fine-tuning 训练。假设要检测的物体类别有 N 类,那么我们就需要把上面预训练阶段的 CNN 模型的最后一层给替换掉,替换成 N+1 个输出的神经元(加 1，表示还有一个背景)，然后这一层直接采用参数随机初始化的方法，其它网络层的参数不变；接着就可以开始继续 SGD 训练了。开始的时候，SGD 学习率选择 0.001。

（3）SVM 分类

经过训练之后，每个 SVM 输出一个该候选框中的物体是否属于该类的概率(不属于这个类别和定位不准确都会影响最后概率大小)。

（4）非极大抑制

Non-Maximum Suppression 就是需要根据 score 矩阵和 region 的坐标信息，从中找到置信度比较高的 bounding box（将相似的框保留概率最大的）

先假设有 6 个矩形框，根据分类器类别分类概率做排序，从小到大分别属于车辆的概率分别为 A、B、C、D、E、F。

(1)从最大概率矩形框 F 开始，分别判断 A~E 与 F 的重叠度 IOU 是否大于某个设定的阈值；

(2)假设 B、D 与 F 的重叠度超过阈值，那么就扔掉 B、D；并标记第一个矩形框 F，是我们保留下来的。

(3)从剩下的矩形框 A、C、E 中，选择概率最大的 E，然后判断 E 与 A、C 的重叠度，重叠度大于一定的阈值，那么就扔掉；并标记 E 是我们保留下来的第二个矩形框。

就这样一直重复，找到所有被保留下来的矩形框。

## 38. Subspace Alignment Based Domain Adaptation for RCNN Detector

**主要思想：**

在没有获得监督的情况下，针对局部目标检测 bounding－box，考虑一种基于子空间对齐的域适应技术，来调整源和目标子空间之间的特性子空间

通过从在源上训练的 RCNN 检测器和并应用到目标上，来获得目标的子空间。再利用子空间对齐技术实现迁移。

**主要方法：**

- Subspace Alignment

我们对数据向量进行标准化，并对源数据向量和目标数据向量进行单独的 PCA。选择 d 个最大的特征值，认为这些特征向量分别作为源和目标子空间的基础。用 Xs 表示源子空间和 Xt 表示目标子空间。使用一个变换矩阵 M 来将源子空间与目标子空间对齐：

$$F(M) = \|X_S M - X_T\|_F^2 \qquad M^* = \underset{M}{\operatorname{argmin}}(F(M)). \qquad (1)$$

- Subspace Alignment for Adapting RCNN

1) 一些通过selective search获得的bounding box与ground truth边界框重叠面积大于一个特定的阈值，则利用它们在获得特定的源域子空间时。
2) 我们在目标数据集上运行RCNN检测器，它最初是在源数据集上训练的。运行RCNN检测器为目标数据集的图像提供每个搜索窗口的得分。分数大于某个阈值的特定类的所有搜索窗口都是子空间生成的正样本。
3) 一旦我们有了目标子空间的正样本，我们就会生成类特定的目标子空间，并分别在每个类上应用子空间对齐方法。

**Algorithm 1** Subspace Alignment based Domain Adaptation for RCNN Detector

**procedure** SA BASED RCNN ADAPTATION(Source Data S,Target Data T)
  **for** each image $j \in$ *Source* and *Target Image* **do**
    $Windows(j) \leftarrow ComputeSelectiveSearchWindows(j)$
    $feat(j) \leftarrow ComputeCaffeFeat(Windows(j))$
  **end for**
  $InitRCNNdetector \leftarrow TrainRCNNonSource(SourceData)$
  **for** each class $i \in$ *Object Class* **do**
    $PosSrc(i) = ()$ and $PosTgt(i) = ()$
    **for** each image $j \in$ *Source* and *Target Image* **do**
      $ol(j) = ComputOverlap(gTBbox(j,i), Windows(i))$ ▷ For source images
      $PosSrc(i) = \text{Stack}(PosSrc(i), feat(i)(ol(j) \geq \gamma)$ ▷ For source images
      $score(i,j) = runInitRCNNdetector(image(j))$ ▷ For target images
      $PosTgt(i) = \text{Stack}(PosTgt(i), feat(i)(score(i,j) \geq \sigma)$ ▷ For target images
    **end for**
    $X_{source}(i) \leftarrow PCA(PosSrc(i))$
    $X_{target}(i) \leftarrow PCA(PosTgt(i))$
  **end for**
  **for** each class $i \in$ *Object Class* **do**
    $ProjectMat(i) \leftarrow SubspaceAlign(X_{source}(i), X_{target}(i))$
  **end for**
  $AdaptedRCNNdetector \leftarrow TrainRCNNonSource(ProjectedSrcData)$
  $boxes \leftarrow runAdaptedRCNNdetector(ProjectedTgtData)$
  $predictBbox \leftarrow runNonMaximumSupression(boxes)$
  **return** *predictBbox*
**end procedure**

## 39. Weakly Supervised Localization of Novel Objects Using Appearance Transfer

**主要思想：**

- 在学习新的类别检测器时，利用语义相似度得出相似度较高的类别，将新类别的检测器的权重与相似类别检测器的权重加权平均，从而得到新类别的检测器

**主要方法：**

新对象只有图像级别的标签，其他图像具有对象级别标签。利用与新对象相似类别的图像训练新对象的检测器。

- 在每个图像中生成一组 object proposals
- 选择高评分作为新对象的正面例子。然后，随机生成包围盒来组成一组反例。鉴于这些正面和负面的例子，我们学习一个外观模型 CNN+SVM
- 使用与新对象和熟悉对象相关联的词向量来建立它们的语义相关性

1） 一个新的对象词向量，可以用熟悉的物体的词向量线性组合表示，其中θ是线性组合参数：

$$\mathbf{v} \approx \theta_1 \mathbf{v}_1 + \theta_2 \mathbf{v}_2 + ... \theta_K \mathbf{v}_K \quad (1)$$

2） 通过优化以下公式，得到最佳的θ

$$\min_{\Theta>0} ||\mathbf{v} - (\theta_1 \mathbf{v}_1 + \theta_2 \mathbf{v}_2 + ... \theta_K \mathbf{v}_K)||_2^2 + \lambda ||\Theta||_1 \quad (2)$$

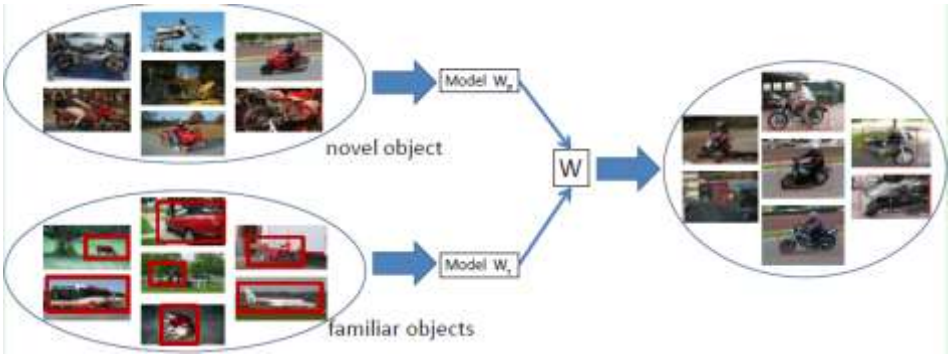
- 根据熟悉的对象与新对象的关联转移外观模型

1） 如果假设对象类的语义关联（词向量词）与外观模型相似，我们可以使用同一θ来表示新对象的外观模型：

$$\mathbf{w}_t = \theta_1 \mathbf{u}_1 + \theta_2 \mathbf{u}_2 + ... \theta_K \mathbf{u}_K \quad (3)$$

2） 最后的 w 是这两个对象的线性组合

$$\mathbf{w} = \gamma \mathbf{w}_p + \mathbf{w}_t \quad (4)$$



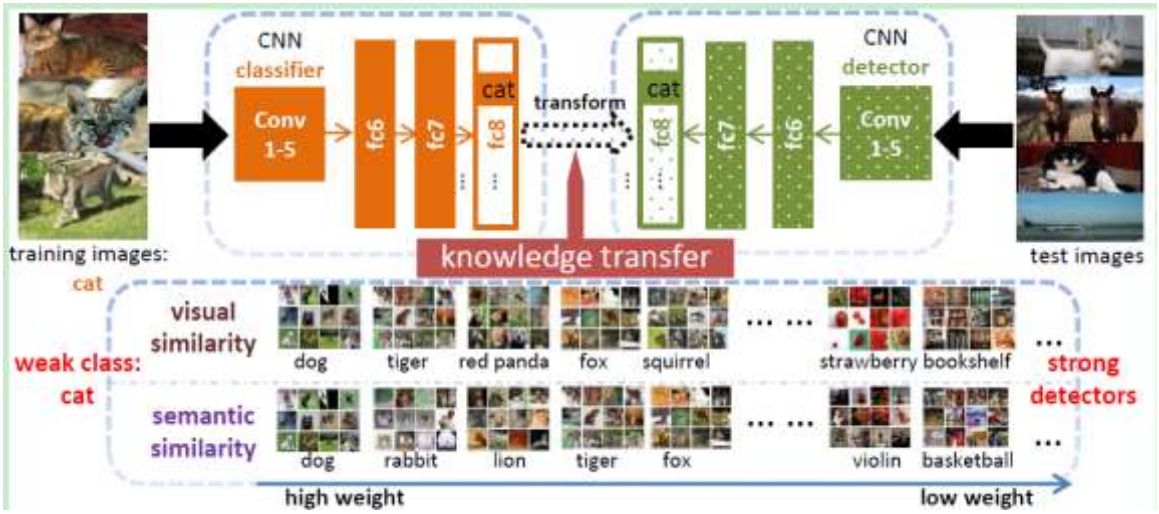

## 40. Large Scale Semi-supervised Object Detection using Visual and SemanticKnowledge Transfer

**主要思想：**

- 基于 LSDA，在差异的基础上，加上相似度不同的类别的权重的影响
- 在完成猫分类器和猫检测器的转换时，采用相似类别（如狗），而不是非相似类别（如小提琴、草莓）的差异，效果更好
- 采用视觉相似度和语义相似度

**主要方法：**

在我们的半监督学习案例中，我们假设我们有一组“完全标记”类别和“弱标记”类别。对于“完全标记”类别，有大量的具有图像级标签和包围盒注释的训练图像可用于学习对象检测器。对于每一个“弱标记”类别，我们有许多包含目标对象的训练图像，但是我们不能访问对象的确切位置。




- LSDA

先预训练分类器，再利用 K 个完全标记的图片（B 域）对分类器进行微调。在利用 K 个完全标记的图片利用 RCNN 算法将分类器微调 fc8 成探测器，其中权重值的变量为$\Delta\mathcal{B}$。选择弱标记图像（A 域），与类别 j 相似的 B 中的 k 个邻近图片的$\Delta\mathcal{B}$取平均值来微调 A 域探测器。

$$\forall j \in \mathcal{A} : \overrightarrow{w_j^{\mathcal{A}}} = \overrightarrow{w_j^{\mathcal{E}}} + \frac{1}{k}\sum_{i=1}^{k}\Delta_{B_i^j} \qquad (1)$$

- visual similarity

CNN 容易混淆视觉上相似的类别，它们有可能 softmax 预测评分较高。视觉相似性（简记$s_v$） “弱标记”类$j \in \mathcal{A}$和“完全标记”类$i \in \mathcal{B}$被定义为：

$$s_v(j,i) \propto \frac{1}{N}\sum_{n=1}^{N} CNN(I_n)_i \qquad (2)$$

$I_n$是$\mathcal{A}$域中类别 j 的图像，$CNN(I_n)_i$是$\mathcal{A}$域的图片通过 CNN 网络之后，softmax 输出其属于$\mathcal{B}$域类别 i 的概率。

利用视觉相似性进行加权平均：

$$\forall j \in \mathcal{A} : \overrightarrow{w_{j_v}^{\mathcal{A}}} = \overrightarrow{w_j^{\mathcal{E}}} + \sum_{i=1}^{m} s_v(j,i)\Delta_{B_i^j} \qquad (3)$$

- semantic relatedness

1) 我们将每个的 K 类作为一个词向量， 300 维 Word2vec 嵌入。从而计算词的相似性 s（i，j）。

2) 我们假设每个弱标签类别$j \in \mathcal{A}$都可以表示成$\mathcal{B}$域中 m 个词向量的线性组合：$v_j \approx \Gamma_j V$, where $V = [v_1; v_2; \ldots; v_i; \ldots; v_m]$，

$\Gamma_j = [\gamma_j^1, \gamma_j^2, \ldots, \gamma_j^i, \ldots, \gamma_j^m]$，则通过优化以下公式，得到一个稀疏向量Γ：

$$\Gamma_j^* = \arg\min_{\Gamma_j \geq 0}(\|v_j - \Gamma_j V\|_2 + \lambda\|\Gamma_j\|_0) \qquad (4)$$

则利用语义相似度进行加权平均：

$$\forall j \in \mathcal{A} : \overrightarrow{w_{j_s}^{\mathcal{A}}} = \overrightarrow{w_j^{\mathcal{E}}} + \sum_{i=1}^{m} s_s(j,i)\Delta_{B_i^j} \qquad (5)$$

其中，$s_s(j,i) = \gamma_j^i$

- Mixture transfer model

根据交叉验证，α取 0.6 较好。

$$s = intersect[\alpha s_v + (1-\alpha)s_s] \qquad (6)$$

## 41. Learning Transferrable Knowledge for Semantic Segmentationwith Deep Convolutional Neural Network

**主要思想：**

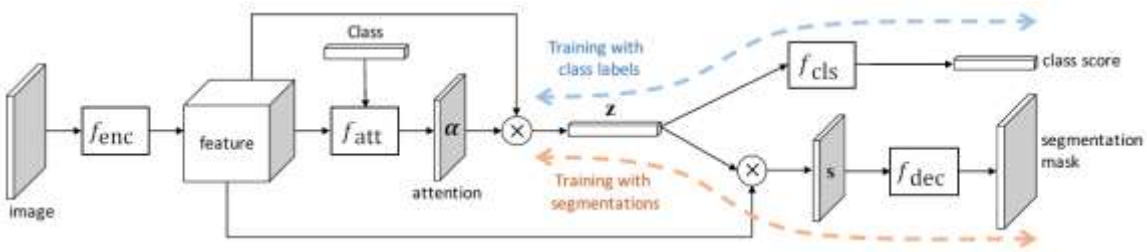
- 利用不同类别的辅助分割注释来指导只有图像级标签的图像分割
- 一种基于注意力模型的解耦编码器解码器结构
- 使用注意模型生成图像中每个类别的空间亮点，并随后使用解码器为每个高亮区域产生前景分割

**主要方法：**

该网络由三部分组成：编码器、解码器和编码器与解码器之间的注意模型。

在这种体系结构中，输入图像首先被编码器转换成多维特征向量，注意力模型识别与图像相关的每个类别的显著区域。注意模型的输出揭示了每个类别在粗特征图中的位置信息，其中每个类别的稠密和详细的前景分割掩模随后被解码器获得。

利用源域的分割注释优化解码器和注意力模型，利用目标域和源域的图像分类标签优化注意力模型。




- 编码器

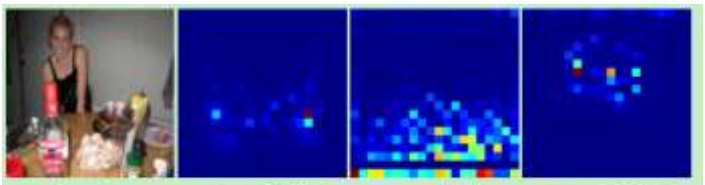
$$\mathbf{A} = f_{\text{enc}}(\mathbf{x};\theta_e), \quad \mathbf{A} \in \mathrm{R}^{M\times D} \tag{1}$$

- Attention model

给定从编码器提取的特征标识符 A∈ $R^{M\times N}$和他相关的类别标签$\mathcal{L}^*$，注意力模型的目标是学习权重向量$\{\alpha^l\}_{\forall l\in\mathcal{L}^*}$，其中$\alpha^l \in R^M$表示每个 l 类别的物体的特征位置相关性。

$$\mathbf{v}^l = f_{\text{att}}(\mathbf{A}, \mathbf{y}^l;\theta_\alpha), \quad \mathbf{v}^l \in \mathrm{R}^M \tag{2}$$

$$\alpha_i^l = \frac{\exp\left(v_i^l\right)}{\sum_i \exp\left(v_i^l\right)}, \quad \boldsymbol{\alpha}^l \in \mathrm{R}^M, \tag{3}$$




《Learning to relate images》中使用 a 3-way tensor 来表示特征与向量的关系，而在这里进行改进：

$$\mathbf{v}^l = \mathbf{W}^{att}\left(\mathbf{W}^{feat}\mathbf{A} \odot \mathbf{W}^{label}\mathbf{y}^l\right) + \mathbf{b}, \tag{4}$$

- 分类器

根据聚集在空间区域的特征，根据类别特定的注意提取特征如下，zl 表示在特征映射的所有通道上定义的特定于类别的特性。

$$\mathbf{z}^l = \mathbf{A}^{\mathrm{T}}\boldsymbol{\alpha}^l, \quad \mathbf{z}^l \in \mathrm{R}^D. \tag{5}$$

$$\min_{\theta_\alpha,\theta_c} \sum_{i\in\mathcal{T}\cup\mathcal{S}} \sum_{l\in\mathcal{L}_i^*} e_c\left(\mathbf{y}_i^l, f_{\text{cls}}(\mathbf{z}_i^l;\theta_c)\right), \tag{6}$$

- 解码器

通过下列公式将注意力模型由稀疏变为稠密：

$$\mathbf{s}^l = \mathbf{A}\mathbf{z}^l, \quad \mathbf{s} \in \mathrm{R}^M \tag{7}$$

致密注意力向量$s^l$作为输入，解码训练由下面的目标函数最小化分割损失：

$$\min_{\theta_\alpha,\theta_s} \sum_{i\in\mathcal{S}} \sum_{l\in\mathcal{L}_i^*} e_s\left(\mathbf{d}_i^l, f_{\text{dec}}(\mathbf{s}_i^l;\theta_s)\right), \tag{8}$$

## 42. Best Practices for Fine-tuningVisual Classifiers to New Domains

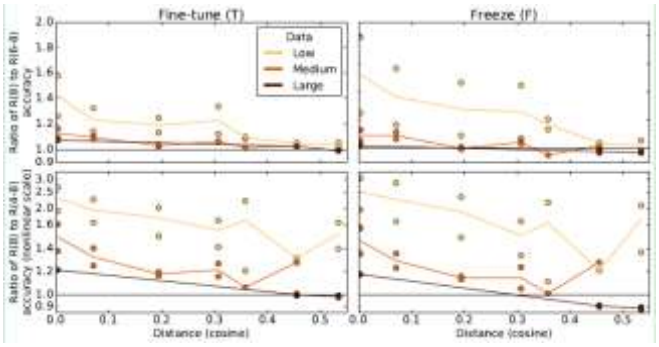
**主要思想：**

- 考虑（1）源数据和目标数据之间的距离（2）目标数据量的大小，选择微调的最佳策略

**主要结论：**

**（1）复制除了最后一层之外的所有层通常是对新数据集进行微调的最佳实践。**

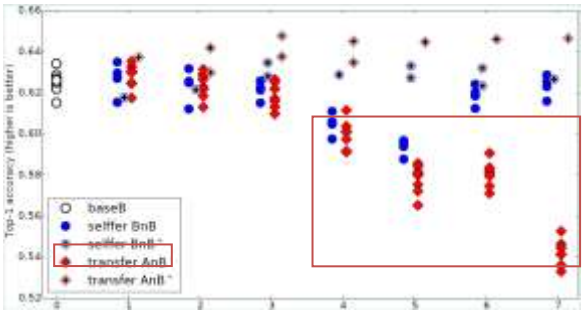
T（A-B）表示层 A-B 复制并微调，而 F（A-B）表示层 A-B 复制并 freezing。R（A-B）表示层 A-B 随机初始化。如 T（1-7）R（8）。



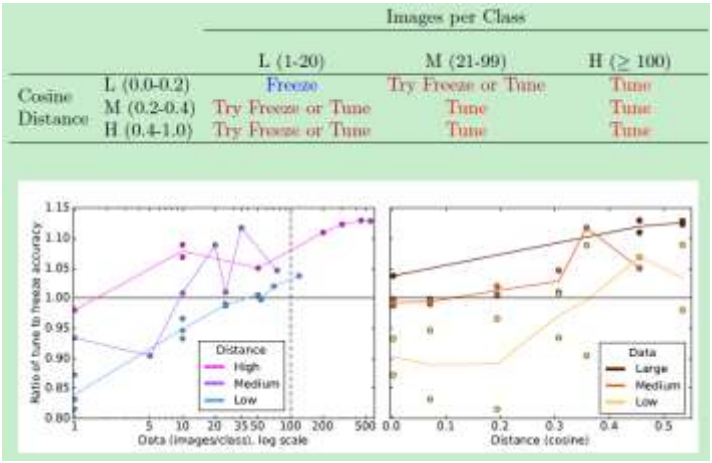

上图是上部是 R（8）的准确率与 R（6-8）的准确度之比。底部：R（8）和 R（4-8）之间的比率。每一点代表一对实验与其他条件固定。左边是微调（T）实验，右边是 Freezing（F）实验。1 以上的值意味着复制优于随机初始化。低于 1 的值意味着复制劣于随机初始化。

可以看到大部分数据都在 1 以上，因此，复制除了最后一层之外的所有层通常是对新数据集进行微调的最佳实践。

这与《How transferable are features in deep neuralnetworks?》结论相悖，因为本实验是在小目标数据集上进行的。




**大量的目标数据，微调是最好低或中等数量的训练数据时，源和目标之间的距离起着更重要的作用。随着距离的增加，微调相对于 Freezing 更好。在低数据和低距离设置，Freezing 优于微调。在中等数据和中到高距离设置，微调优于 Freezing。**



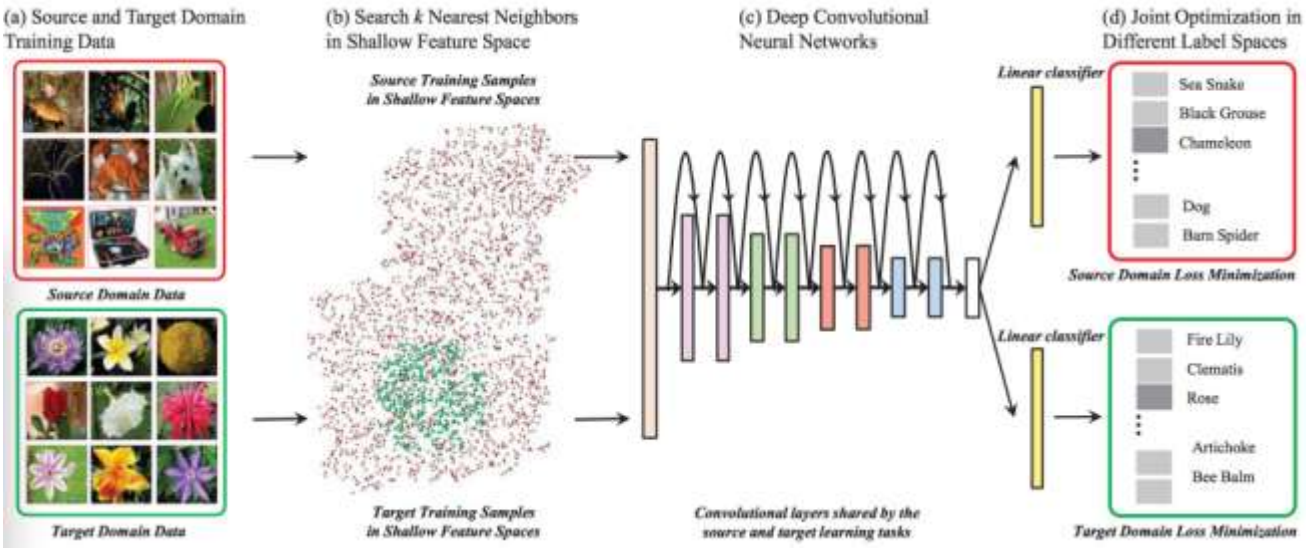

## 43. Borrowing Treasures from the Wealthy: Deep Transfer Learning through Selective Joint Fine-Tuning

**主要思想：**

- 从源域中选择一个与目标域中层特征相似的子集，然后利用目标域全部实例和源域选中的实例去微调
- 其中中层特征使用 Gabor 线性滤波器或 CNN 一层和二层非线性滤波器提取
- 源域和目标域共享 CNN，最后分类使用 softmax 进行分类，源域只用选中实例，目标域是全部实例

**主要方法：**

a）源域和目标域中的数据集。(b）通过低级别特征空间选择源域中目标域训练样本的最近 k 邻。(C）深度卷积神经网络权值初始化和预 train 的 ImageNet or Places。(d）共同优化其自有标签空间中的源域和目标域成本函数。




- Nearest Neighbor Ranking

给定目标域中训练图像 x 的基于直方图的描述符，我们在源域中搜索其最近的邻居。在不同的层次结构中，不同的卷积层的内核可能是不同的。为了确保在最近的邻居搜索过程中，不同的卷积层之间的权重相等，每个内核响应的直方图都被相应层的内核总数标准化。

$$\mathcal{H}(\boldsymbol{x}_i^t, \boldsymbol{x}_j^s) = \sum_{h=1}^{D} w_h[\kappa(\boldsymbol{\phi}_h^{i,t}, \boldsymbol{\phi}_h^{j,s}) + \kappa(\boldsymbol{\phi}_h^{j,s}, \boldsymbol{\phi}_h^{i,t})], \quad (1)$$

$w_h = 1/N_h, N_h$是卷积核在相应的层的数量，$\phi_h^{i,t}$和$\phi_h^{j,s}$是图像$x_i^t$和$x_j^s$h 层的直方图， κ（•, •）是 KL 散度。

- Hard Samples in the Target Domain

我们为这个目的提出一个迭代的方案。在第 m 次迭代之后，我们计算信息熵来度量目标域中训练样本的分类不确定度。

$$\mathcal{H}_i^m = -\sum_{c=1}^{U} p_{i,c}^m \log(p_{i,c}^m), \qquad (2)$$

在下一次迭代中，我们增加了训练样本的最近邻数。继续对在当前迭代中训练的模型进行微调。对于目标域中的一个训练样例 x，在下一次迭代中它的最近邻的数量定义如下。

$$\mathcal{K}_i^{m+1} = \begin{cases} \mathcal{K}_i^m + \sigma_0, & \widehat{y_i^t} \neq y_i^t \\ \mathcal{K}_i^m + \sigma_1, & \widehat{y_i^t} = y_i^t \quad and \quad \mathcal{H}_i^m \geq \delta \\ \mathcal{K}_i^m, & \widehat{y_i^t} = y_i^t \quad and \quad \mathcal{H}_i^m < \delta \end{cases} \quad (3)$$

其中，$\sigma_0$，$\sigma_1$，δ是常数。$\widehat{y_i^t}$是预测标签，$\mathcal{K}_i^m$是第 m 次迭代的最邻近数。这里取$\sigma_0 = 4\mathcal{K}_0$，$\sigma_1 = 2\mathcal{K}_0$，$\mathcal{K}_0$是初始化最邻近数。

## 44. DLID: Deep Learning for Domain Adaptationby Interpolating between Domains

**主要思想：**

- 受测地线路径中的“中间表示”启发，提出了一种基于深度模型的域间插值方法。
- 通过在中间数据集上使用预测稀疏分解（PSD），非监督训练深度非线性特征提取器来实现的，在该数据集中，源数据的量逐渐被目标样本所取代。

**主要方法：**

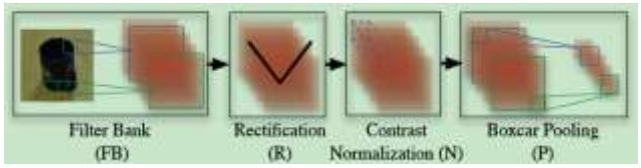
1） Interpolatingpath

从所有源数据样本 DS 开始，我们生成中间采样数据集，在每一个连续数据集中，我们逐步增加从 DT 中随机抽取的样本的亲部分，并从 DS 中抽取样本的比例降低。让$p\epsilon[1,\ldots\ldots, P]$是我们生成的 P 数据集的索引。$D_p = D_S$，当 p= 1；$D_p = D_T$，当 p = P 。$p\epsilon[2,\ldots\ldots, P-1]$，是从源域和目标域抽取一定比例的样本，并满足$D_p$中目标域样本的比例小于$D_{p+1}$。每一个数据集都可以看作是 S 和 t 之间特定类型的插值路径的一个点。

2） 特征提取器

对于每一个中间数据$D_p$，训练很深的非线性特征提取器$F_{W_p}$，参数为$W_p$。采用预测稀疏分解（PSD）的方法进行无监督训练。

架构如下：




每层特征提取器由四部分组成的级联在一起，即滤波器单元（FB）、Rectication Units (R)，Contrast Normalization Unit (N)和 Boxcar Pooling Unit (P)。

$$y_j = g_j \tanh(\sum_i k_{ij} * x_i)$$

PSD 算法如下：

$$E_{PSD} = \|X - WZ\|_2^2 + \lambda\|Z\|_1 + \|Z - C(X,K)\|_2^2 \quad (4)$$

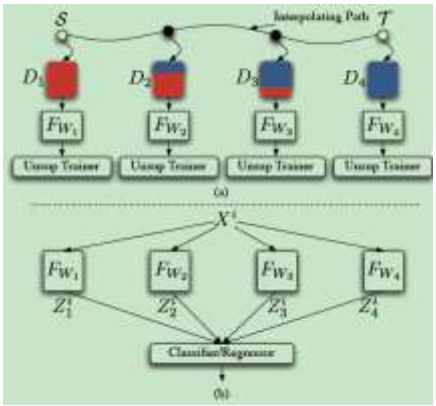
$$Z^* = \arg\min_Z E_{PSD}(X,Z,W,K) \quad (5)$$

3) 训练分类器

一旦特征提取器相应路径上的所有点都训练，任何输入样本然后通过连接所有的输出的特征提取器共同创建的输入路径的特征值。

$$Z^i = [F_{W_1}(X^i)\ F_{W_2}(X^i) \ldots F_{W_P}(X^i)] = [Z_1^i\ Z_2^i \ldots Z_P^i].$$

分类器在源数据 DS 上训练，通过训练组（$Z^i$，$Y^i$），最小化一个适当的损失函数。




## 45. PSD：Domain Adaptation for Large-Scale Sentiment Classification:A Deep Learning Approach

标准的稀疏编码（Sparse Coding）问题是最小化下面这个目标函数：

$$\mathcal{L}(X,Z;D) = \sum_i \frac{1}{2}\|X^{(i)} - DZ^{(i)}\|_2^2 + \lambda\|Z^{(i)}\|_1. \quad (2\text{-}1)$$

其中，λ是正则化超参数，X(i) ∈ Rn 是输入变量，Z(i) ∈ Rm 是想要求得的稀疏表示，而矩阵 D ∈ Rn×m 由一组基函数构成，通常 m > n 以获得超完备表示。然而，如果对 D 没有额外限制，很容易得到一个不需要的解：只要用同一个常量分别放大 D 和缩小 Z，这样就减小了公式 2-1 中的第二项，同时保证了第一项的大小不变。因此，为了避免出现这种解，D 的各列一般要求保证单位 L2 范式。

可预测稀疏分解算训练了一个参数为Θ的前向非线性回归变量 F(X(i)；Θ)，它直接将输入 X(i) 映射成它的稀疏表示 Z(i)。新的目标函数如下所示：

$$\mathcal{L}(X,Z;D,\Theta) = \sum_i \frac{1}{2}\|X^{(i)} - DZ^{(i)}\|_2^2 + \lambda\|Z^{(i)}\|_1 + \alpha\|Z^{(i)} - F(X^{(i)};\Theta)\|_2^2. \quad (2\text{-}9)$$

其中超参数λ和α是正则化权重，而 F(X(i)；Θ）通常定义如下：

$$F(X^{(i)};G,W,B) = G\tanh(WX^{(i)} + B) \quad (2\text{-}10)$$

其中的参数包括一个过滤矩阵 W ∈ Rm×n，一个偏斜向量 B ∈ Rm 和一个对角矩阵 G ∈ Rm×m。tanh 是双曲线函数。G 是用来放大 tanh 的有限值域范围的，并且Θ = {G, W, B}。如果原始问题存在多个解，那么 PSD 尝试从中找到一个可以用非线性回归变量来近似的解，这样就能获得更快的推断速度。

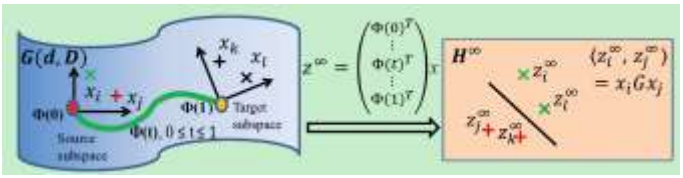
## 46. Geodesic Flow Kernel for Unsupervised Domain Adaptation

**主要思想：**

- 源域数据全部有标注，目标域数据全部无标注
- SGF 把 source 和 target 分别看成高维空间（Grassmann 流形）中的两个点，在这两个点的测地线距离上取 n 个中间点构成了一条测地线的路径。只需要找到合适的每一步的变换，就能从 source 变换到 target 了。
- GFK（1）确定 source 和 target 路径上中间点的个数（无限个）。（2）它通过提出一种 kernel 方法，利用路径上的所有点的积分。（3）当有多个 source 的时候，提出 Rank of Domain 度量，用跟 target 最近的 source 进行迁移

**主要方法：**

GFK 方法有以下几个步骤：选择最优的子空间维度进行变换；构建测地线；计算测地线流式核；以及构建分类器。




1） 选择最优的子空间维度

D 维数据，降低到 d 维？这个 d 怎么选择？提出 subspace disagreement measure（SDM），这个度量可以反映出两个 domain 在多少维的子空间下能够保持最大一致性。分别计算两个 domain 和 S+T 的空间的夹角：记$\sin\alpha_d$和$\sin\beta_d$分别为 source 和 target 与 S+T 的夹角。这两个角度如果很小的话，表示两个 domain 距离很小。

$$\mathcal{D}(\mathrm{d}) = 0.5\left[\sin\alpha_{\mathrm{d}} + \sin\beta_{\mathrm{d}}\right] \quad (7)$$

2） 构建测地线流

$\Phi(0)=P_S, \Phi(1)=P_T$，对于定义域在[0,1]之间的一个点 t，$\Phi(t)\in G(d,D)$。这里的$G(d,D)$就是一个 D 维向量空间中所有 d 个向量构成的 Grassmann 流形。这时，测地线函数在点 t 处的函数值就为

$$\boldsymbol{\Phi}(t) = \boldsymbol{P_S U_1 \Gamma}(t) - \boldsymbol{R_S U_2 \Sigma}(t). \quad (1)$$

其中，$P_S^{\mathrm{T}} P_T = U_1 \Gamma V^{\mathrm{T}}, \quad R_S^{\mathrm{T}} P_T = -U_2 \Sigma V^{\mathrm{T}}. \quad (2)$，Γ（t）和Σ（t）是对角矩阵，元素为$\cos t\theta_i$何$\sin t\theta_i$。

3） 计算测地线流式核

$$\langle z_i^{\infty}, z_j^{\infty} \rangle = \int_0^1 (\boldsymbol{\Phi}(t)^{\mathrm{T}} \boldsymbol{x}_i)^{\mathrm{T}} (\boldsymbol{\Phi}(t)^{\mathrm{T}} \boldsymbol{x}_j)\, dt = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{G} \boldsymbol{x}_j \quad (4)$$

其中，$$\boldsymbol{G} = [\boldsymbol{P_S U_1} \;\; \boldsymbol{R_S U_2}] \begin{bmatrix} \Lambda_1 & \Lambda_2 \\ \Lambda_2 & \Lambda_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{U_1^{\mathrm{T}} P_S^{\mathrm{T}}} \\ \boldsymbol{U_2^{\mathrm{T}} R_S^{\mathrm{T}}} \end{bmatrix} \quad (5)$$

4） 构建分类器

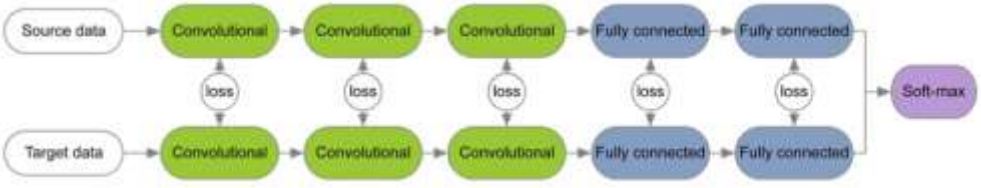
## 47. Beyond Sharing Weights for Deep Domain Adaptation

**主要思想：**

- 深度学习的域适应方法不应侧重于对域转移不变量的学习，这使它们的区别性更小。相反，我们应该显式地对域转移建模。
- 引入了一个双流的 CNN 架构，两者之间的权重没有被共享，而是利用损失函数，限制这些权重保持线性相关的关系

**主要方法：**

一个流在源数据上运行，另一个流在目标一个上运行。他们的权重没有共享，引入损失函数，防止相应的权重彼此差异太大。另外，我们在学习源和目标表示之间使用 MMD



$$L(\theta^s,\theta^t|\mathbf{X}^s,Y^s,\mathbf{X}^t,Y^t)=L_s+L_t+L_w+L_{MMD}, \quad (1)$$

$$L_s=\frac{1}{N^s}\sum_{i=1}^{N^s}c(\theta^s|\mathbf{x}_i^s,y_i^s) \quad (2)$$

$$L_t=\frac{1}{N_l^t}\sum_{i=1}^{N_l^t}c(\theta^t|\mathbf{x}_i^t,y_i^t) \quad (3)$$

$$L_w=\lambda_w\sum_{j\in\Omega}r_w(\theta_j^s,\theta_j^t) \quad (4)$$

$$L_{MMD}=\lambda_u r_u(\theta^s,\theta^t|\mathbf{X}^s,\mathbf{X}^t), \quad (5)$$

（2）和（3）为源域和目标域的分类损失，（4）是权重的距离损失，（5）式是 MMD。

- **权重距离损失**

最简单的是$\|\theta_j^s-\theta_j^t\|_2^2$，这将会对权重之间的细微差别进行惩罚，使整个系统过于僵化。为了增加更多的灵活性，我们使用指数损失函数：

$$r_w(\theta_j^s,\theta_j^t)=\exp\left(\|\theta_j^s-\theta_j^t\|^2\right)-1. \quad (6)$$

它仍然倾向于保持两种流的权重非常接近，当域有显著差异时，这可能是非常严格的。因此，我们建议通过允许一个流中的权重来进行线性变换，从而进一步放松这一点。

$$r_w(\theta_j^s,\theta_j^t)=\exp\left(\|a_j\theta_j^s+b_j-\theta_j^t\|^2\right)-1, \quad (7)$$

**注意：**

然而，哪个层应该或不应该共享它们的权重的最佳选择是依赖于应用程序。在 UAV 案例中，允许前两层中的权重不同，后几层权重共享，从而获得最高性能，我们理解这意味着域转移是由低级的变化引起的。

相比之下，对于 Office 数据集，最好只允许最后两层中的权重有所不同。

## 48. Learning Deep Feature Representations with Domain Guided Dropout for Person Re-identification

**主要思想：**

- 根据它在这个领域的有效性分配每个神经元的一个特定的 drop out 率。
- 有两种方案，deterministic scheme 和 stochastic scheme。
- 在所有领域的数据集共同训练之后，我们将标准的 drop out 替换为 deterministic Domain Guided Dropout，并重新开始训练。

**主要方法：**

利用多领域身份认证作为例子，假设有 D 个领域，每个领域包含$N_i$个图像有$M_i$个不同的人。让$\mathrm{M}=\sum_{l=1}^{D}M_i$,则新的人的身份标签为$y'\in\{1,2,\ldots,\mathrm{M}\}$,则构造一个单任务目标函数，即学习一个 softmax 分类器 f 和特征提取器 g:

$$\arg\min_{f,g}\sum_{i=1}^{D}\sum_{j=1}^{N_i}\mathcal{L}\left((f\circ g)(x_i^{(j)}),y'^{(j)}_i\right). \quad (2)$$

我们定义了一个特定神经元对这个例子的影响，当我们移除神经元时，我们就会得到损失函数：

$$s_i=\mathcal{L}(g(x)_{\setminus i})-\mathcal{L}(g(x)), \quad (3)$$

其中 g（x）\i 是我们将 i-th 神经元的响应设为 0 之后的特征向量。对于每个域 D，我们将计算 si 的期望取其所有样本值以获得平均影响分数。为加速计算，我们采用近似泰勒展开：

$$s_i\approx-\frac{\partial\mathcal{L}}{\partial g(x)_i}g(x)_i+\frac{1}{2}\frac{\partial^2\mathcal{L}}{\partial g(x)_i^2}g(x)_i^2. \quad (4)$$

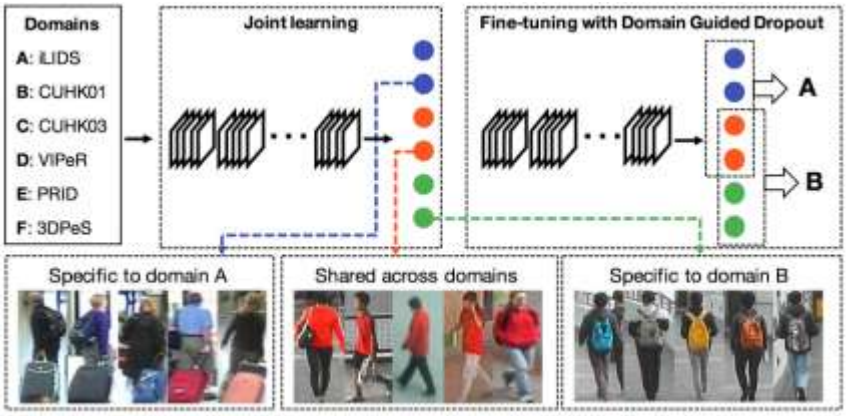
- deterministic

$$m_i=\begin{cases}1 & \text{if } s_i>0\\ 0 & \text{if } s_i\le 0\end{cases} \quad (5)$$

- stochastic

$$p(m_i=1)=\frac{1}{1+e^{-s_i/T}}, \quad (6)$$

当 T→0，它相当于第一种方案；当 T→∞，落到的比例 0.5 的标准 dropout。



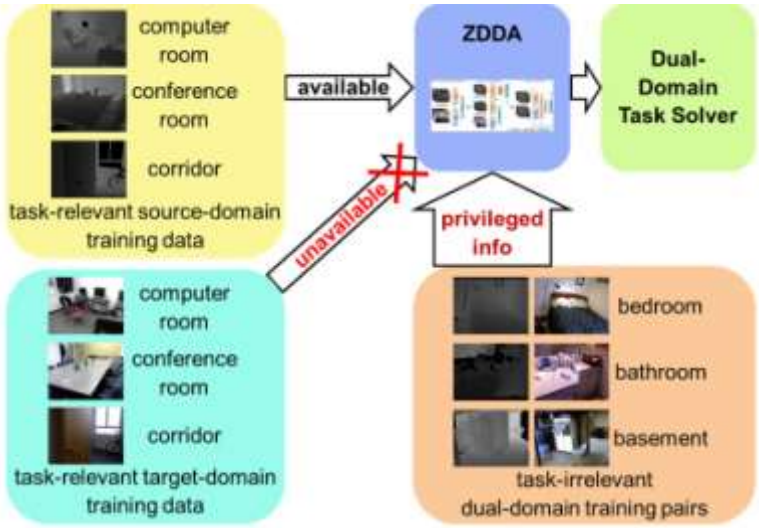

## 49. Zero-Shot Deep Domain Adaptation

**主要思想：**

使用任务相关的源域数据和任务无关的双域数据对，学习任务相关的目标域表示

**主要方法：**

在这个 3-场景分类任务的例子中，源和目标域分别是深度和 RGB。与任务无关的 rgb-d 数据对对与任务相关的数据有不同的标签。



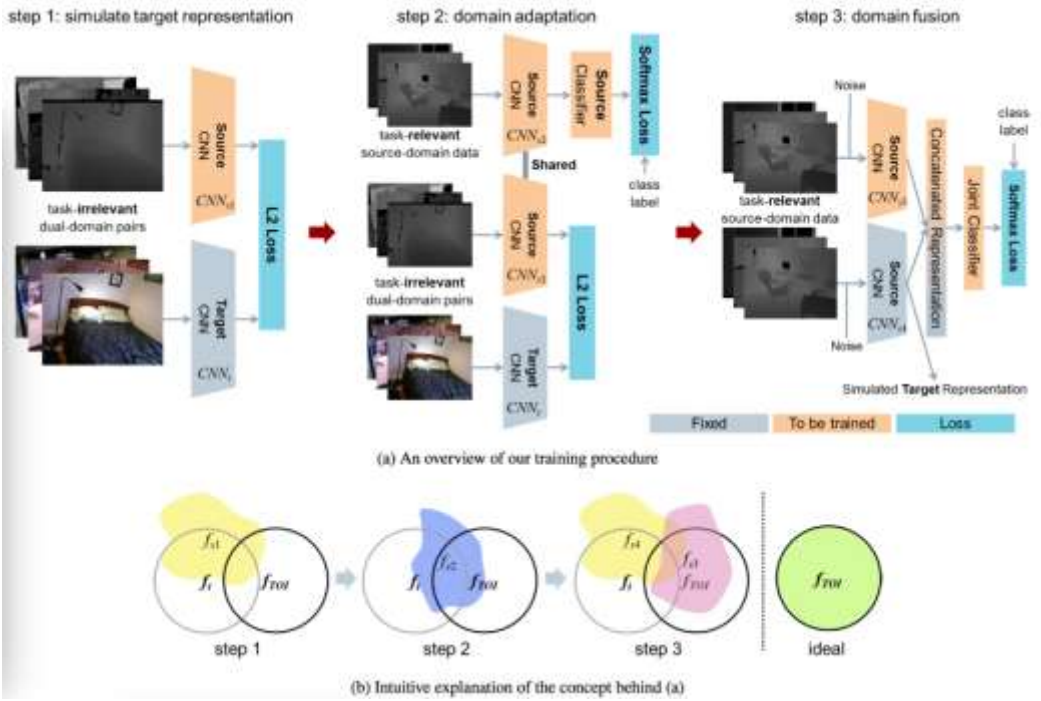

**领域适应：**当任务相关的数据在训练时间无法获得时，可以解决与任务相关的 Ds 和 D t 的问题。我们假设在训练时，我们可以访问 Ds 中的任务相关的标记数据，以及 Ds 和 Dt 中任务无关的双域对。

**域融合：**给定上述假设，当测试数据在 Ds 和 D t 都可用，但有噪声，解决与任务相关的问题。

Step1：创建了两个卷积神经网络(CNN)，CNN s1 和 CNN t，来处理与任务无关的 RGB-D 对。这一步的目的是训练 CNN s1，这样我们就可以通过将对应的深度图像输入到 CNNs1 中来模拟一个 RGB 图像的表示。

Step2：我们添加了 CNNs2(与 CNN s1 相同的网络架构)，以及对网络的一个 classifier。新添加的 CNN 将任务相关的深度图像作为输入，并与原始源 CNN 共享所有的权重。训练 CNNs2 和源分类器，使得 softmax 损耗和 L2 损失的加权总和被最小化。

Step3：创建两个 CNN、CNN s3 和 CNN s4(每个都与 CNN s1 相同的网络架构)，并在上面添加一个连接层来连接它们的输出表示。连接的表示连接到一个联合分类器。CNN s3 和 CNN s4 都将任务相关的深度图像作为输入。我们随机选择 CNN s3 和 CNN s4 的一些输入，并添加噪声。在联合分类器上采用 softmax 损失，并利用深度数据进行监督训练。




(a) An overview of our training procedure

(b) Intuitive explanation of the concept behind (a)

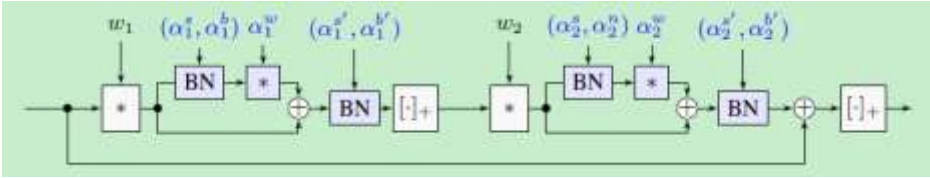
## 50. Learning multiple visual domains with residualadapters

**主要思想：**

- 提出一个可调的深的网络架构，通过adapterresidual modules，可以同时应用在多个视觉领域
- 网络的调整相当于，根据领域调整卷基层的参数，又相当于添加适当的卷积层到网络中
- 通过在整个体系结构中调整批处理和实例标准化层中的某些参数来吸收单个神经网络中的不同域

**主要方法：**

提出基于神经网络去预测另一个网络的参数。$\alpha_d = Ae_{d_x}$，其中$e_{d_x}$是领域$d_x$中图像x的指示函数，A矩阵的每列是参数向量$\alpha_d$。则通过以上公式，很容易构造一个辅助网络，通过x预测d，则参数α = ψ（x）可以表示为一个神经网络的输出。如果d已知，则$\alpha_d$ = ψ（x,d）。网络$\phi_{\varphi(x)}(x)$的结构是由φ(x)决定的。




为了限制特定域的参数的数量，α选择的是1×1过滤器。

$$g(x;\alpha) = x + \alpha * x.$$

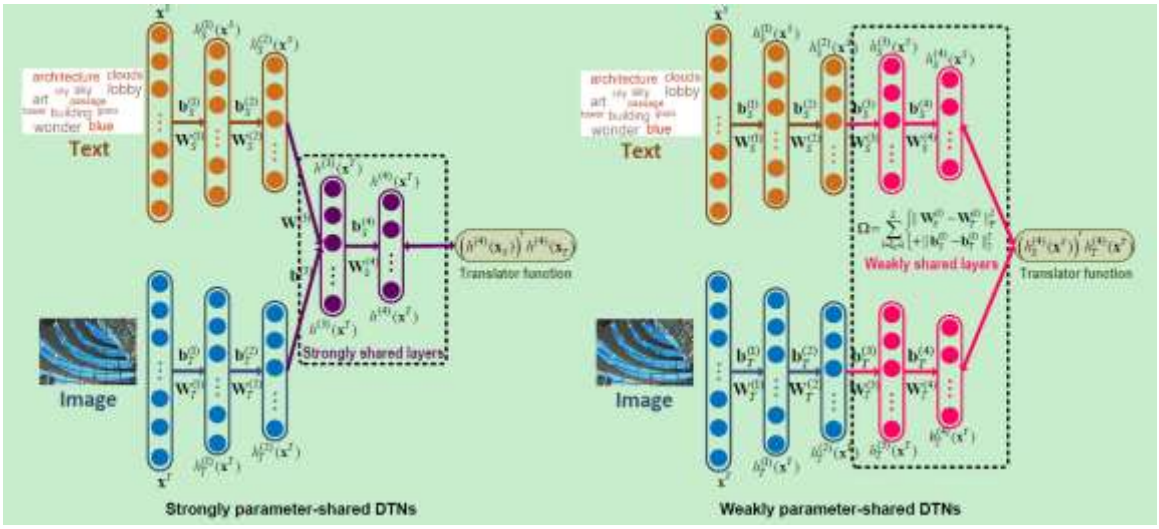
## 51. Weakly-Shared Deep Transfer Networks forHeterogeneous-Domain Knowledge Propagation

**主要思想：**

- 能够将标签信息通过不同的领域，特别是从文本域转移到图像域
- 构建一对 SAE(Stacked Auto-Encoders)，采取对文字和图像作为输入，共享高层参数
- 采用弱共享而不是强共享

**主要方法：**

图中左边，DTNs 使用共享参数层（层 3 和 4）为两域共享一组特征。相反，在右边的图，弱共享参数 DTNs 放松约束，允许两域使用不同层提取特征。这些弱共享参数层通过施加一个正则化，决定共享的程度。这种结构非常灵活，能够代表域特定的特性和跨域的共享特性。




给定文本数据集 $\mathcal{D}_S = \{(\bar{\mathbf{x}}_j^S, \bar{y}_j^S)\}_{j=1}^{N_S}$，和文本图像数据对 $\mathcal{C} = \{(\mathbf{x}_i^S, \mathbf{x}_i^T)\}_{i=1}^{N_C}$

- 特征共享

  第一个 L1 层分别学习文本和图像的表示，而最后的 L2 层提供共享表示：

$$\Omega = \sum_{l=L_1+1}^{L} (\|\mathbf{W}_S^{(l)} - \mathbf{W}_T^{(l)}\|_F^2 + \|\mathbf{b}_S^{(l)} - \mathbf{b}_T^{(l)}\|_2^2). \quad (2)$$

- Deeply Transfer Mechanism

  对于成对图像数据 $x^T$ 和文本数据 $\bar{x}_j^S$，转换函数用于将源域的标签传播到目标域，如下所示（‘表示转置）：

$$f(\mathbf{x}^T) = \sum_j^{N_S} \bar{y}_j^S \left(h_S^{(L)}(\mathbf{x}_j^S)\right)' h_T^{(L)}(\mathbf{x}^T), \quad (3)$$

我们学习的参数，提出通过最小化的 DTNs 上面的标签转移过程中发生的损失，以及最大化的一组共生对文本和图像捕捉跨域的对齐信息之间的一致性

- Empirical loss on the auxiliary set

在目标域我们有个小的辅助集，$A_T = \{(\bar{\mathbf{x}}_t^T, \bar{y}_t^T)\}_{t=1}^{N_T}$，则损失函数为

$$\underset{\Theta}{\operatorname{argmin}} J_1 = \sum_{t=1}^{N_T} \ell\left(\bar{y}_t^T, f(\bar{\mathbf{x}}_t^T)\right), \quad (4)$$

- Empirical loss on co-occurrence pairs

$$\underset{\Theta}{\operatorname{argmin}} J_2 = \sum_{i=1}^{N_C} \chi\left(\left(h_S^{(L)}(\mathbf{x}_i^S)\right)' h_T^{(L)}(\mathbf{x}_i^T)\right), \quad (5)$$

其中 $\chi = exp(-x)$。

最终优化函数：

$$\underset{\Theta}{\operatorname{argmin}} J = J_1 + \eta J_2 + \frac{\gamma}{2}\Omega + \frac{\lambda}{2}\Psi, \quad (6)$$

其中 $\text{where } \Psi = \sum_{l=1}^{L} (\|\mathbf{W}_S^{(l)}\|_F^2 + \|\mathbf{b}_S^{(l)}\|_2^2 + \|\mathbf{W}_T^{(l)}\|_F^2 + \|\mathbf{b}_T^{(l)}\|_2^2)$ 为正则项。

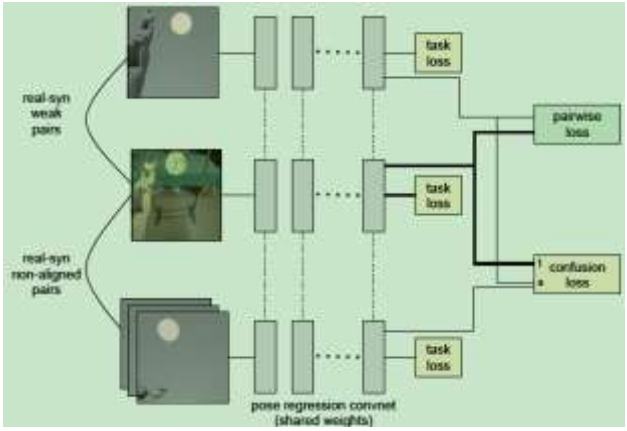
## 52. Adapting Deep VisuomotorRepresentations with Weak PairwiseConstraints

**主要思想：**

- 提出结合分布和成对图像对齐的方法，并通过在源域和目标域中使用弱对齐的图像来消除对标签的需求
- 先学会一种卷积网络从原图像预测关键位置，然后应用 guided policy search 将关键位置映射为行动
- 提出一个模型，增强了任务的损失之外，两个适应损失函数进行对齐特征空间

**主要方法：**

在确定一个弱配对的源和目标图像之后，模型将一个任务的损失，在分布水平上两个域的域混淆损失，源图像和目标图像的配对损失。这三个的损失，确保我们的模型学习准确而稳健的执行任务。




- Task loss

我们定义的问题，找到图像的特征$f(x,\theta_{repr})$,这表示允许从一个大的数据集 XS 的源图像和目标图像的 XT 学习视觉运动政策。寻求输入图像 X 并直接输出一些标签ϕ。

$$\mathcal{L}_\phi(x,\phi;\theta_\phi,\theta_{\text{repr}})=\frac{1}{2K}\sum_{i=1}^{K}||\theta_\phi^T f(x^{(i)};\theta_{\text{repr}})-\phi^{(i)}||_2^2 \quad (1)$$

- Domain confusion loss

域分类器参数$\theta_D$是保证将图片按照其域分类。同时通过优化卷积神经网络的参数$\theta_{\text{repr}}$，使得域分类器无法进行明确分类。

$$\mathcal{L}_{\text{conf}}(x_S,x_T,\theta_D;\theta_{\text{repr}})=-\sum_{x\in(x_S\cup x_T)}\sum_d\frac{1}{D}\log q_d(x,\theta_D,\theta_{\text{repr}}). \quad (2)$$

Here, $q$ corresponds to the domain classifier activations:

$$q(x,\theta_D;\theta_{\text{repr}})=\text{softmax}(\theta_D^T f(x;\theta_{\text{repr}})) \quad (3)$$

- Pairwise loss

寻找具有相似标签的源和目标图像对，并在表示空间中对齐它们。

$$\mathcal{L}_{\text{pairwise}}(x_S,x_T;P,\theta_{\text{repr}})=\sum_{(i,j)\in P}\left[\frac{1}{2}\rho\left(x_S^{(i)},x_T^{(j)};\theta_{\text{repr}}\right)^2\right], \quad (4)$$

$$\rho\left(x_S^{(i)},x_T^{(j)};\theta_{\text{repr}}\right)=\left\|f(x_S^{(i)};\theta_{\text{repr}})-f(x_T^{(j)};\theta_{\text{repr}})\right\|_2. \quad (5)$$

- Complete objective

① 首先找到一个初始表示$\theta_{repr}$，利用源图像最大限度地减少任务损失$\mathcal{L}_\phi$和和域混淆误差$\mathcal{L}_{conf}$

② 在源和目标集中提取图像特征表示，源图像最近邻的目标域图像构成弱配对 p，这些配对的标签可以被用来计算目标图像的任务损失$\mathcal{L}_\phi$

③ 一旦 P 已经确定，我们把它固定，优化$\mathcal{L}_{pairwise}$ 相对于表示确保配对图像在特征空间中足够接近。

## 53. Unsupervised Domain Adaptation for Face Recognition in Unlabeled Videos

**主要思想：**

- 利用大规模无标记的视频数据来缩小不同领域之间的差距，将知识从大规模的标签上转移
- 自适应实现：（i）通过特征匹配从网络提取知识到视频自适应网络；（ii）通过合成数据增强进行特征恢复；（iii）通过域对抗鉴别器学习域不变特征。
- 提出了一种具有识别功能的特征融合方法，可以有效地将多个帧的特征集合起来，并根据它们对人脸识别的适应性进行有效的排序，剔除不想似的帧。

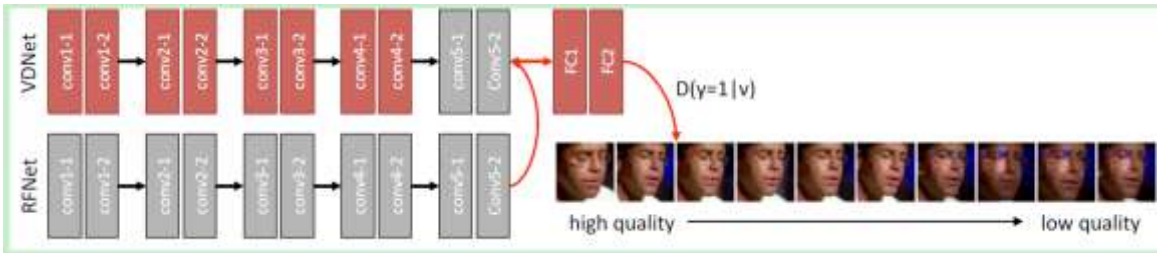
**主要方法：**

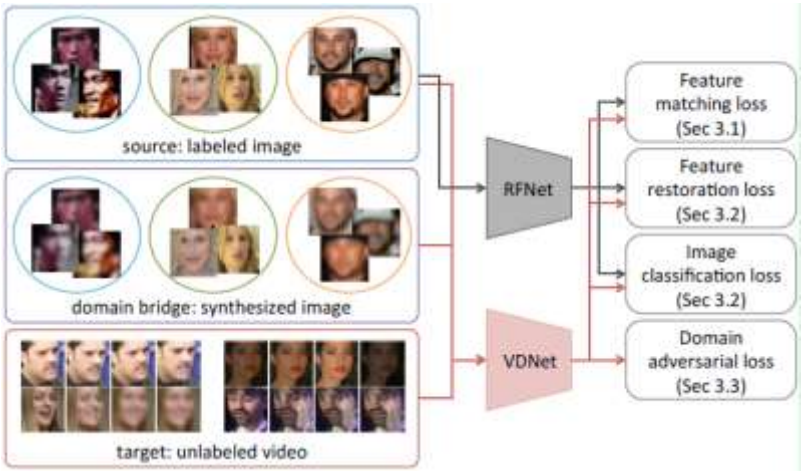
由于视频中的模糊、压缩、运动和其他伪影，静止图像和视频之间还是具有一定的差距。

为了利用有标签的网络人脸图像，我们通过一个人脸识别引擎来训练 VDNet，该引擎预先利用 web 人脸数据集预训练,，我们称它为参考网络(RFNet)。

红色和灰色的块分别表示可训练和固定的模块。VDNet 不仅与 RFNet 共享网络体系结构，而且使用相同的网络参数进行初始化。一旦经过训练，鉴别器 D 就可以在视频序列中对帧进行排序，通过指示帧是否与人脸识别引擎相匹配，并拒绝那些极其不适合人脸识

别的帧。




- feature matching (FM) loss

VDNet 的特征提取函数为$\phi(\cdot)$: $R^D \to R^K$,RFNet 的特征提取器为$\varphi(\cdot)$: $R^D \to R^K$，静止图像数据集为$\mathcal{L}$，视频数据集为$\mathcal{V}$。

$$\mathcal{L}_{\text{FM}} = \frac{1}{|\mathcal{I}|}\sum_{x\in\mathcal{I}} \|\phi(x) - \psi(x)\|_2^2 \qquad (1)$$

- feature restoration (FR) loss

训练 VDNet“还原”源域图像的特征表示（没有数据增强）

$$\mathcal{L}_{\text{FR}} = \frac{1}{|\mathcal{I}|}\sum_{x\in\mathcal{I}} \mathbb{E}_{B(\cdot)}\left[\|\phi(B(x)) - \psi(x)\|_2^2\right] \qquad (2)$$

其中 B 是转换，在这项工作中，我们考虑以下三种类型的图像转换：

① 线性运动模糊：内核长度随机选择（5, 15）和内核角选择在（10, 30）

② 尺度变化：我们重新调整图像 6 1 的原始图像尺寸小

③ JPEG 压缩：质量参数是随机设置的（30, 75）

- Image classification loss (N-pair loss)

$$\mathcal{L}_{\text{IC}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(\phi(B_i(x_i^+))^\top \psi(x_i))}{\sum_{n=1}^{N}\exp(\phi(B_i(x_i^+))^\top \psi(x_n))} \qquad (3)$$

- Domain Adversarial Loss

① Two-way D

使用 two-way softmax 分类器 D 区分源域（y=1）和目标域的合成图像和视频（y=2）。原始图像来自图像域，但图像的合成图像和随机视频帧都被训练成属于同一域的

$$\begin{aligned}\mathcal{L}_{\mathcal{D}} = &-\mathbb{E}_{x\in\mathcal{I}}\left[\log \mathcal{D}(y=1|\phi(x))\right] \\ &-\mathbb{E}_{x\in B(\mathcal{I})\cup\mathcal{V}}\left[\log \mathcal{D}(y=2|\phi(x))\right] \qquad (6)\\ \mathcal{L}_{\text{Adv}} = &-\mathbb{E}_{x\in B(\mathcal{I})\cup\mathcal{V}}\left[\log \mathcal{D}(y=1|\phi(x))\right] \qquad (7)\end{aligned}$$

② Three-way D

使用 three-way softmax 分类器 D 区分源域（y=1）和合成图像（y=2）和视频（y=3）

$$\begin{aligned}\mathcal{L}_{\mathcal{D}} = &-\mathbb{E}_{x\in\mathcal{I}}\left[\log \mathcal{D}(y=1|\phi(x))\right] \\ &-\mathbb{E}_{x\in B(\mathcal{I})}\left[\log \mathcal{D}(y=2|\phi(x))\right] \qquad (8)\\ &-\mathbb{E}_{x\in\mathcal{V}}\left[\log \mathcal{D}(y=3|\phi(x))\right] \\ \mathcal{L}_{\text{Adv}} = &-\mathbb{E}_{x\in B(\mathcal{I})\cup\mathcal{V}}\left[\log \mathcal{D}(y=1|\phi(x))\right] \qquad (9)\end{aligned}$$

- 总体目标

$$\mathcal{L} = \mathcal{L}_{\text{FM}} + \alpha\mathcal{L}_{\text{FR}} + \beta\mathcal{L}_{\text{IC}} + \gamma\mathcal{L}_{\text{Adv}} \qquad (10)$$

- Discriminator-Guided Feature Fusion

鉴别器已经被训练来区分静止的图像和模糊的图像或视频帧，因此它的输出可能已经被用来作为一个高质量图像的分数。

有了鉴别器的得分，视频 V 的聚合特征向量被表示为特征向量的加权平均值，如下所列

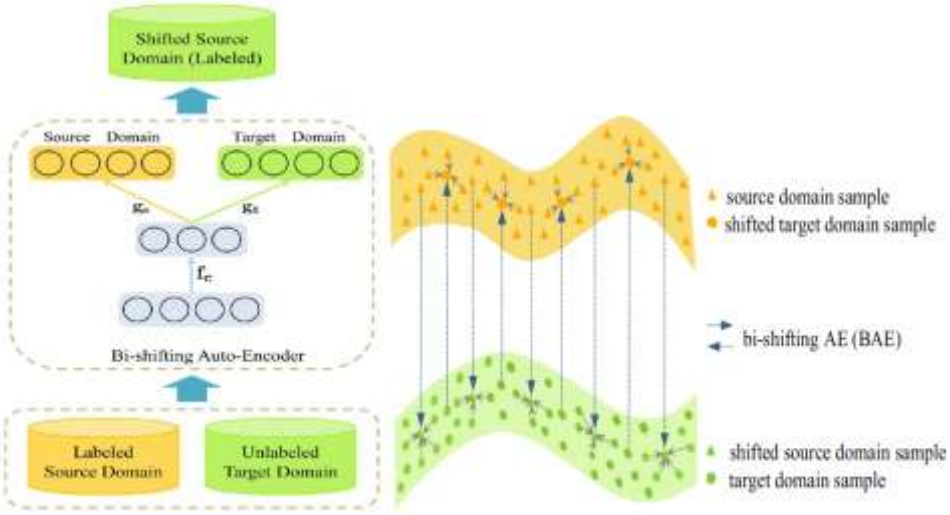
$$\phi_V = \frac{\sum_{v\in V} \mathcal{D}(y=1|\phi(v)) \cdot \phi(v)}{\sum_{v\in V} \mathcal{D}(y=1|\phi(v))}. \qquad (11)$$

## 54. Bi-shifting Auto-Encoder for Unsupervised Domain Adaptation

**主要思想：**

- 编码器和解码器的非线性映射函数保证了将样本从一个域转移到另一个域的可行性
- 稀疏重构约束则保证移位域和理想域遵循相似分布

**主要方法：**




- Auto-Encoder

通过 BAE，源域样本可以转换为目标域样本，并通过稀疏和线性重建的目标域样本表示；同样的，目标域样本可以转化为源域，也可以由源域样本进行稀疏线性重建。

双移位自动编码器网络由一个编码器$f_c$和两个解码器，即$g_s$和$g_t$组成，它们可以分别将输入样本转换为源域和目标域。

$$\mathbf{z} \triangleq \mathbf{f}_c(\mathbf{x}) = \sigma(\mathbf{W}_c\mathbf{x} + \mathbf{b}_c) \qquad (4)$$

$$\begin{aligned} \mathbf{g}_s(\mathbf{z}) &= \sigma(\mathbf{W}_s\mathbf{z} + \mathbf{b}_s), \\ \mathbf{g}_t(\mathbf{z}) &= \sigma(\mathbf{W}_t\mathbf{z} + \mathbf{b}_t), \end{aligned} \qquad (5)$$

- sparely reconstructed

如果每个移动源域样本（绿色三角）可以由目标域（绿色圈）的几个局部邻居稀疏重构，它们往往遵循相似的局部结构，这意味着整个相似分布。同样，每个移动目标域样本（黄色圆圈）被限制为稀疏重构的源域邻居（黄色三角形），强制他们遵循类似的分布。

$$\mathbf{g}_t(\mathbf{f}_c(\mathbf{x}_i^s)) = \mathbf{X}_t\beta_i^t, \quad s.t., |\beta_i^t|_0 < \tau, \qquad (6)$$

$g_t(f_c(x_i^s))$表示源域通过编码器$f_c$和解码器$g_t$重构的目标域样本，$\beta_i^t$是一个与邻域相关的非零值的稀疏向量。

则最终优化目标为：

$$\begin{aligned} \min_{\mathbf{f}_c,\mathbf{g}_s,\mathbf{g}_t,\mathbf{B}_s,\mathbf{B}_t} & ||\mathbf{X}_s - \mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_s))||_2^2 + ||\mathbf{X}_t\mathbf{B}_t - \mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_s))||_2^2 \\ & + ||\mathbf{X}_s\mathbf{B}_s - \mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_t))||_2^2 + ||\mathbf{X}_t - \mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_t))||_2^2 \\ & + \gamma\left(\sum_{i=1}^{n_s}|\beta_i^s|_1 + \sum_{i=1}^{n_t}|\beta_i^t|_1\right). \end{aligned} \qquad (9)$$

其中，γ 是控制稀疏性的一个参数，即较大的 γ 导致稀疏重建所需的样本较少，而较小的 γ 导致更多的样本被选择用于稀疏重建。

因此，标记的源域样本（$X_s, y_s$）可以转移到目标域$(\mathbf{G}_t, \mathbf{y}_s), \mathbf{G}_t \triangleq \mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_s))$。映射的源域样本$G_t$与目标域有相似的分布，所以任何监督的方法可以应用到学习在目标域分类的分类器。

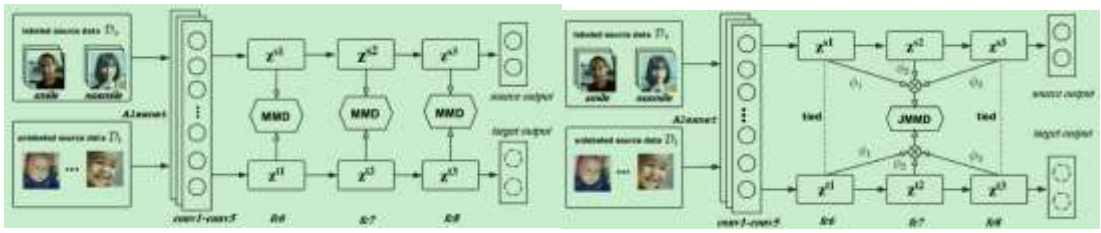
## 55. Detecting Smiles of Young Children via Deep Transfer Learning

**主要思想：**

- 该方法借鉴了成人表情识别（笑或者不笑）的知识，并成功地迁移到了婴儿表情识别上，在非常有限的标记数据进行训练。
- 利用 DAN 和 JAN 进行迁移学习在婴儿表情识别上的应用

**主要方法：**

DAN 和 JAN 的架构：




自己构造了一个婴儿人脸的数据集，namely Baby and ChildSmile (BCS)。并利用CelebFaces Attributes (CelebA) and FotW 两个数据集的知识进行迁移学习，年龄分布不均匀，对婴儿和儿童的年龄小于5岁的图像的比例只有约3.8%。

| Race | Smile | Non-Smile | All |
|---|---|---|---|
| Caucasian | 222 | 227 | 449 |
| African-American | 152 | 160 | 312 |
| Asian | 247 | 227 | 484 |
| All | 621 | 624 | 1245 |

- Discrepancy Analysis

  成人和婴儿的人脸结构存在差异，使得识别结果不理想。

| Method | Accuracy (%) | Recall (%) |
|---|---|---|
| *on FotW Validation* | | |
| AlexNet | 86.51 | 77.08 |
| ResNet | 87.03 | 79.32 |
| *on BCS* | | |
| AlexNet | 59.12 | 25.60 |
| ResNet | 69.00 | 45.89 |

- Contribution Assessment of Deep Transfer Learning

  利用DAN和JAN后，效果提升

| Method | Accuracy (%) | Recall (%) |
|---|---|---|
| AlexNet [15] | 59.12 | 25.60 |
| DAN_AlexNet | 77.03 | 63.77 |
| JAN_AlexNet | 83.85 | 74.23 |
| ResNet [12] | 69.00 | 45.89 |
| DAN_ResNet | 80.16 | 69.89 |
| JAN_ResNet | **85.06** | **78.10** |

## 56. Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning

**主要思想：**

- 匹配概率分布的高阶中心距
- 与 KL 散度相比，KL 散度只匹配一阶矩，我们的方法匹配了更高阶的矩。与 MMD 相比，我们的方法匹配了每一阶的高阶矩，而且它不需要任何计算昂贵的距离和内核矩阵的组合。
- 因为在紧凑区间上的分布通过矩序列具有等价的表示。
- 与经典方法 MMD、DANN、DDC、DAN、COARL、DLID 相比都取得了更好的性能

**主要方法：**

$$CMD(p,q) = \frac{1}{|b-a|}\|\mathbb{E}(X) - \mathbb{E}(Y)\|_2 + \sum_{k=2}\frac{1}{|b-a|^k}\|c_k(X) - c_k(Y)\|_2 \quad (5)$$

where $\mathbb{E}(X)$ is the expectation of $X$, and

$$c_k(X) = \left(\mathbb{E}\left(\prod_{i=1}^{N}(X_i - \mathbb{E}(X_i))^{r_i}\right)\right)_{r_1+\ldots+r_N=k,\ r_1,\ldots,r_N\geq 0}$$

定义 the final central moment discrepancy regularizer为CMD的经验估计。这一定义包括三个近似步骤:(a)仅计算边际中心的计算量，(b)通过参数K计算K个有限的中心矩，和(c)通过用经验期望替换期望值的概率分布的抽样

$$CMD_K(X,Y) = \frac{1}{|b-a|}\|\mathbf{E}(X) - \mathbf{E}(Y)\|_2 + \sum_{k=2}^{K}\frac{1}{|b-a|^k}\|C_k(X) - C_k(Y)\|_2 \quad (6)$$

其中 $\mathbf{E}(X) = \frac{1}{|X|}\sum_{x\in}C_k(X) = \mathbf{E}((x-\mathbf{E}(X))^k)$

将 CMD 与分类损失联合起来，利用反向传播来优化参数，优化公式如下:

$$\min_{\theta\in\Theta}\ \mathbf{E}(l(\theta, X_S, Y_S)) + \lambda \cdot d(A_H(\theta, X_S), A_H(\theta, X_T)) \quad (2)$$

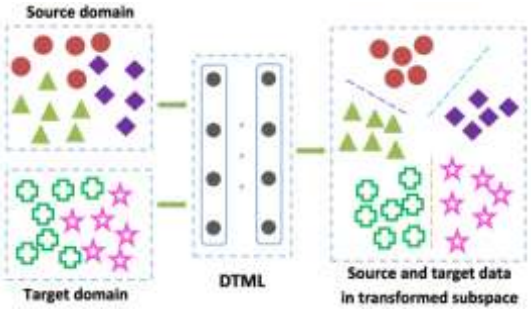
## 57. Deep Transfer Metric Learning

**主要思想:**

- DTML通过最大化类间距离和最小化内部类距离，并最小化源域和目标域在网络顶层的分布差异，从而学习了一个深度度量网络
- 进一步开发一个深度监督的转移度量学习(DSTML)方法，方法是在DTML中添加一个额外的目标，其中隐藏层和顶层的输出都是联合优化的

**主要方法:**

对于来自源域和目标域的训练集的每个样本，我们将其传递给开发的深层神经网络。我们执行两个约束条件在所有训练样本的输出的网络:1)内部类变化的差异最大化和最小化,和 2)分布源域和目标域之间的分歧在顶部层网络的最小化。




对于输入 x，对应第 i 层输出为:

$$f^{(m)}(\mathbf{x}) = \mathbf{h}^{(m)} = \varphi\left(\mathbf{W}^{(m)}\mathbf{h}^{(m-1)} + \mathbf{b}^{(m)}\right) \in \mathbb{R}^{p^{(m)}}, \quad (1)$$

在顶层，提出了一种强监督的深度度量学习方法

$$\min_{f^{(M)}} J = S_c^{(M)} - \alpha S_b^{(M)} + \gamma\sum_{m=1}^{M}\left(\|\mathbf{W}^{(m)}\|_F^2 + \|\mathbf{b}^{(m)}\|_2^2\right), \quad (3)$$

其中α（α＞0）是一个自由的参数,它可以平衡类间紧性和类间性之间的重要关系;$\|Z\|_F$表示矩阵Z的Frobenius范数;γ（γ＞0）是一个可调的正正则化参数; S c和S b定义了类内紧性和类间可分性，定义如下

$$S_c^{(m)} = \frac{1}{Nk_1}\sum_{i=1}^{N}\sum_{j=1}^{N}P_{ij}\, d^2_{f^{(m)}}(\mathbf{x}_i, \mathbf{x}_j), \quad (4)$$

$$S_b^{(m)} = \frac{1}{Nk_2}\sum_{i=1}^{N}\sum_{j=1}^{N}Q_{ij}\, d^2_{f^{(m)}}(\mathbf{x}_i, \mathbf{x}_j), \quad (5)$$

其中 $d^2_{f^{(m)}}(\mathbf{x}_i, \mathbf{x}_j) = \left\|f^{(m)}(\mathbf{x}_i) - f^{(m)}(\mathbf{x}_j)\right\|_2^2$. (2)

- deeply supervised transfer metric learning (DSTML)

我们进一步提出了一种深度监督的迁移度量学习(DSTML)方法，以便更好地利用所有层的输出的差别信息。我们制定了以下优化问题：

$$\min_{f^{(M)}} J = J^{(M)} + \sum_{m=1}^{M-1} \omega^{(m)} h\left(J^{(m)} - \tau^{(m)}\right), \quad (12)$$

$$J^{(m)} = S_c^{(m)} - \alpha S_b^{(m)} + \beta D_{ts}^{(m)}(\mathcal{X}_t, \mathcal{X}_s) + \gamma \left(\|\mathbf{W}^{(m)}\|_F^2 + \|\mathbf{b}^{(m)}\|_2^2\right), \quad (13)$$

其中 是第 m 层的 DTML 的函数。$J^{(M)}$是最顶层的损失，$J^{(m)}$是第 m 层隐层的损失。h(x)=max（x,0），$\tau^{(m)}$是阈值，控制$J^{(m)}$是否有效。$\omega^{(m)}$是顶层和隐层损失的权值。

## 58. Unsupervised Cross-Domain Image Generation

**主要思想：**

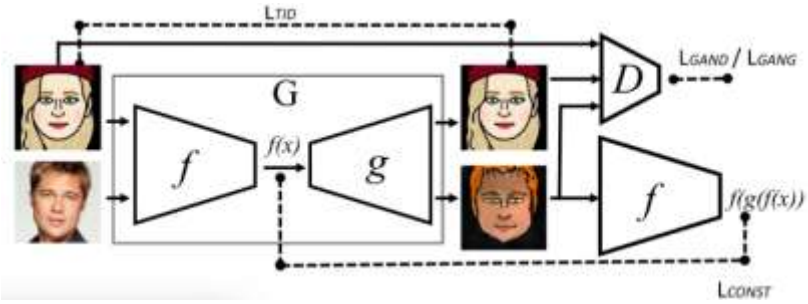
使用一个复合损失函数，其中包括一个 a multiclass GAN loss, an f-constancy component, and a regularizing component, 该组件鼓励 G 将样本从 T 映射到自己。

**主要方法：**

利用了一个特殊结构的深层神经网络，其中函数 G 是由输入函数 f 和一个学习函数 g 组成。f 为特征提取卷积网络，g 为反卷积网络。

复合损失函数包括：一项是一个生成对抗网络(GAN)项,鼓励 G(x)函数可生成与真实的目标域不可区分样本,无论 x 或者 x t . 第二损失项执行每个源域样本 x，‖f(x) − f(G(x))‖很小。第三个损失鼓励 G 将样本从 T 映射到自己

应用了不同域的损失项：目标域的像素损失和源域的特征损失。




优化 LD 和 LG=$L_{GANG} + \alpha L_{CONST} + \beta L_{TID} + \gamma L_{TV}$

$$L_D = -\mathbb{E}_{x\in s} \log D_1(g(f(x))) - \mathbb{E}_{x\in t} \log D_2(g(f(x))) - \mathbb{E}_{x\in t} \log D_3(x) \quad (3)$$

$$L_{GANG} = -\mathbb{E}_{x\in s} \log D_3(g(f(x))) - \mathbb{E}_{x\in t} \log D_3(g(f(x))) \quad (4)$$

$$L_{CONST} = \sum_{x\in s} d(f(x), f(g(f(x)))) \quad (5)$$

$$L_{TID} = \sum_{x\in t} d_2(x, G(x)) \quad (6)$$

$L_{TV}$是 anisotropic total variation loss，以使结果图像稍微平滑。$z = [z_{i,j}] = G(x)$:

$$L_{TV}(z) = \sum_{i,j} \left((z_{i,j+1} - z_{ij})^2 + (z_{i+1,j} - z_{ij})^2\right)^{\frac{\beta}{2}}, \quad (7)$$

## 59. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization

**主要思想：**

- 给定一个内容输入和一个风格输入，AdaIN 调整内容输入的均值和方差，使其与风格输入的相似。
- 能够同时完成多种风格转换
- 计算时间更少

**主要方法：**

- Batch Normalization

在网络的每一层输入的时候，又插入了一个归一化层，也就是先做一个归一化处理，然后再进入网络的下一层。

如果我们直接对网络某一层 A 的输出数据做归一化然后送入网络下一层 B，这样会影响到本层所学习到的特征。）怎么办？论文引入了可学习的参数 γ、β（变换重构）来保留其学习到的特征

$$\mathrm{BN}(x)=\gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta \tag{1}$$

$$\mu_c(x)=\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}x_{nchw} \tag{2}$$

$$\sigma_c(x)=\sqrt{\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw}-\mu_c(x))^2+\epsilon} \tag{3}$$

- Instance Normalization

不同于 BN 层，这里 μ（x）和 σ（x）在空间维度上独立的每个通道和每个样品进行计算：

$$\mathrm{IN}(x)=\gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta \tag{4}$$

$$\mu_{nc}(x)=\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}x_{nchw} \tag{5}$$

$$\sigma_{nc}(x)=\sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw}-\mu_{nc}(x))^2+\epsilon} \tag{6}$$

- Conditional Instance Normalization

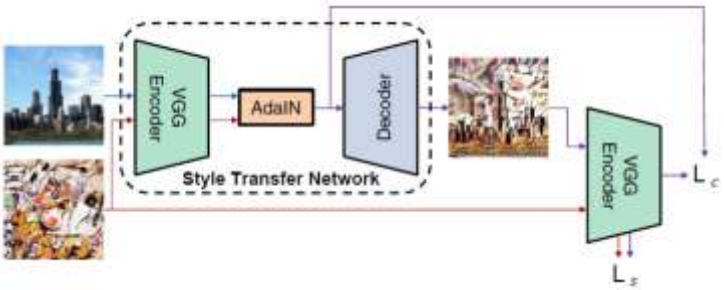
提出了一个条件实例规范化层，它为每种样式学习一组不同的参数 γ s 和 β s，来实现多种风格的转换：

$$\mathrm{CIN}(x;s)=\gamma^s\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta^s \tag{7}$$

- Adaptive Instance Normalization

AdaIN 没有可学习的 affine parameters。相反，它自适应地从样式输入中计算 affine parameters：

$$\mathrm{AdaIN}(x,y)=\sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\mu(y) \tag{8}$$




在将内容和风格的图片编码到特征空间，我们将内容和风格图像的特征都输入到 AdaIN 层对齐均值和方差，产生目标特征映射空间 t：

$$t=\mathrm{AdaIN}(f(c),f(s)) \tag{9}$$

随机初始化的解码器 g 被训练成将 t 映射回图像空间，生成程序化图像 T（c，s）：

$$T(c,s)=g(t) \tag{10}$$

计算损失函数，训练解码器

$$\mathcal{L}=\mathcal{L}_c+\lambda\mathcal{L}_s \tag{11}$$

$$\mathcal{L}_c=\|f(g(t))-t\|_2 \tag{12}$$

$$\mathcal{L}_s=\sum_{i=1}^{L}\|\mu(\phi_i(g(t)))-\mu(\phi_i(s))\|_2+\sum_{i=1}^{L}\|\sigma(\phi_i(g(t)))-\sigma(\phi_i(s))\|_2 \tag{13}$$

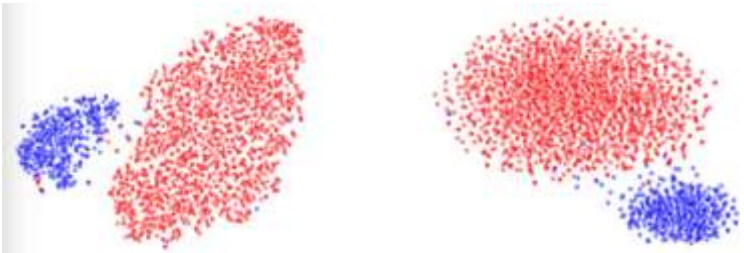
## 60. Revisiting Batch Normalization For Practical Domain Adaptation

**主要思想：**

- 在网络所有的 BN 层中，通过将源域的统计数据调整到目标域，实现了对域适应任务的深度适应效果
- 用目标域的统计数据替换源域中每个 BN 层的统计数据

**主要方法：**

不同数据集下浅层和深层的 mini-batch BN 特征向量分布的可视化。每个点代表一个小批量中的 BN 统计数据。不同颜色的点来自不同数据集。显然，来自不同领域的 BN 统计数据被分为不同集群



(a) Shallow layer distributions　　(b) Deep layer distributions

1) DNN 的浅层和深层都受域转移的影响，仅仅通过操纵输出层来适应域是不够的
2) BN 层的统计数据包含了数据域的特征

**Algorithm 1** Adaptive Batch Normalization (AdaBN)

**for** neuron $j$ in DNN **do**
  Concatenate neuron responses on all images of target domain $t$: $\mathbf{x}_j = [\ldots, x_j(m), \ldots]$
  Compute the mean and variance of the target domain: $\mu_j^t = \mathbb{E}(\mathbf{x}_j^t)$, $\sigma_j^t = \sqrt{\mathrm{Var}(\mathbf{x}_j^t)}$.
**end for**
**for** neuron $j$ in DNN, testing image $m$ in target domain **do**
  Compute BN output $y_j(m) := \gamma_j \frac{\left(x_j(m) - \mu_j^t\right)}{\sigma_j^t} + \beta_j$
**end for**

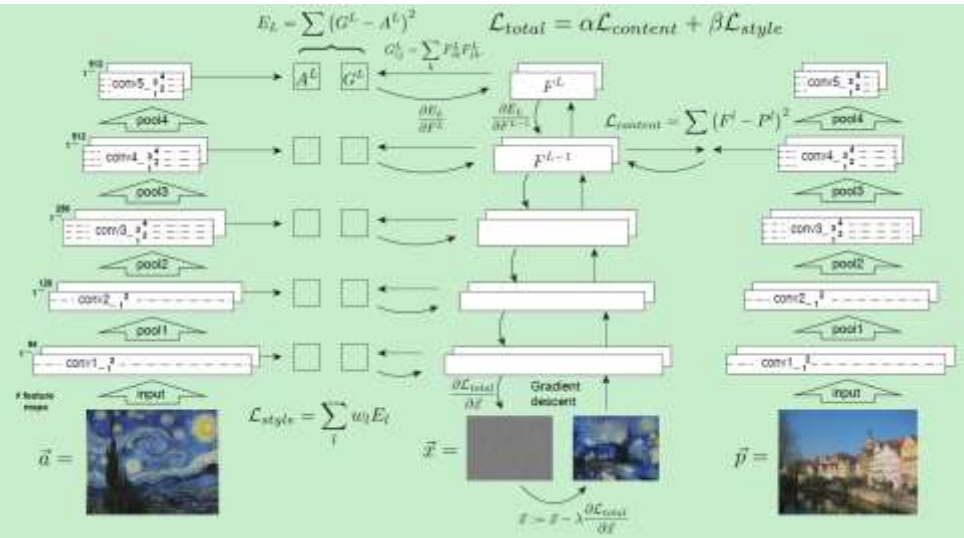
## 61. Image Style Transfer Using Convolutional Neural Networks

**主要思想：**

- 使用 CNN 进行特征提取，然后使用这些提取到的特征进行 reconstruct。不同的 CNN 的 conv layer 提取到特征是不一样的，低层次的偏向于点 线等特征，高层次的更加偏向于 texture 信息
- 作者使用 VGG19 的网络结构来做 feature extractor，其最终是将 conv4_1 作为 contentlayer，conv1_1，conv2_1，conv3_1，conv4_1 和 conv5_1 作为 style layer

**主要方法：**

1) 首先提取和存储 content 和 style 特征。style 图像 a 通过网络，计算并存储在所有层上的 stylerepresentation $A^l$（左）。content 图像 p 通过网络，并存储在一层中的 content representation $P^l$（右）。
2) 然后将随机白噪声图像 x 通过网络，计算其 style 特征$G^l$和 content 特征$F^l$。
3) 在每一层中，计算均方差$G^l$和$A^l$之间的风格损失$\mathcal{L}_{style}$（左）。同时计算$P^l$和$F^l$的平方差-内容损失$\mathcal{L}_{constent}$（右）。总损失$\mathcal{L}_{total}$是内容和形式的损失之间的线性组合。




- Content representation

给定一个图像 x，卷积神经网络每层使用滤波对图像进行编码。一个层有$N_l$个滤波器会产生$N_l$个大小为$M_l$的特征图，$M_l$是特征图的高度乘以宽度。所以，l 层的响应可以存储在一个矩阵$F^l \in \mathcal{R}^{N_l \times M_l}$

$$\mathcal{L}_{content}(\vec{p},\vec{x},l) = \frac{1}{2}\sum_{i,j}(F_{ij}^l - P_{ij}^l)^2 . \qquad (1)$$

- Style representation

为了获得输入图像风格的表示，我们使用一个特征空间来捕获纹理信息。此特性空间可以在网络的任何层上的过滤器响应之上构建。它由不同的滤波器响应之间的相关性组成，有 Gram 矩阵(至今为止为什么 Gram 矩阵能够表示风格信息，还未知)表示：

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l. \qquad (3)$$

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G_{ij}^l - A_{ij}^l)^2 \qquad (4)$$

$$\mathcal{L}_{style}(\vec{a},\vec{x}) = \sum_{l=0}^{L} w_l E_l. \qquad (5)$$

$$\mathcal{L}_{total}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{content}(\vec{p},\vec{x}) + \beta\mathcal{L}_{style}(\vec{a},\vec{x}) \qquad (7)$$

## 62. Demystifying Neural Style Transfer

**主要思想：**

- 从理论上证明了匹配的 Gram 矩阵等价于 MMD 过程。因此，神经风格转换中的风格信息本质上是由 CNN 的激活分布来表示的，可以通过分布对齐实现风格转换。
- 我们利用其他几种分布对准方法，发现这些方法都能产生很好的转移结果。
- 提出风格转换是一种特殊形式的 domain adaption，将特征映射中每个位置的特征作为一个单独的数据样本对待，而在传统的域适应问题将每个图像作为一个样本

**主要方法：**

Gram 矩阵表示风格损失：

$$\mathcal{L}_{style}^l = \frac{1}{4N_l^2 M_l^2}\sum_{i=1}^{N_l}\sum_{j=1}^{N_l}(G_{ij}^l - A_{ij}^l)^2. \qquad (6)$$

$$G_{ij}^l = \sum_{k=1}^{M_l} F_{ik}^l F_{jk}^l. \qquad (7)$$

通过公式推导可得，Gram 矩阵表示风格损失（6）式，可等价于二阶多项式核 k（x，y）$=x^T y^2$的 MMD.

$$\mathcal{L}_{style}^{l} = \frac{1}{4N_l^2 M_l^2} \sum_{k_1=1}^{M_l} \sum_{k_2=1}^{M_l} \Big( k(\mathbf{f}_{k_1}^{l}, \mathbf{f}_{k_2}^{l}) + k(\mathbf{s}_{k_1}^{l}, \mathbf{s}_{k_2}^{l}) - 2k(\mathbf{f}_{k_1}^{l}, \mathbf{s}_{k_2}^{l}) \Big) \quad (9)$$
$$= \frac{1}{4N_l^2} \mathrm{MMD}^2[\mathcal{F}^l, \mathcal{S}^l],$$

$f_k^l$表示$F^l$矩阵的第 K 列，$s_k^l$表示$S^l$矩阵的第 K 列。

- MMD with Different Kernel Functions

(1) Linear kernel: $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$;
(2) Polynomial kernel: $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + c)^d$;
(3) Gaussian kernel: $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}\right)$.

- BN Statistics Matching

BN 层的统计量（即均值和方差）包含不同域的特征。受此观察启发，他们使用不同领域的独立 BN 统计数据。这个简单操作有效地对齐不同的域分布。作为一个特殊的域适应问题，我们相信一个特定层的 BN 统计也可以表示样式。因此，我们通过调整两幅图像中两幅特征图的 BN 统计量（平均值和标准差）来构造另一种样式损失：

$$\mathcal{L}_{style}^{l} = \frac{1}{N_l} \sum_{i=1}^{N_l} \left( (\mu_{F^l}^{i} - \mu_{S^l}^{i})^2 + (\sigma_{F^l}^{i} - \sigma_{S^l}^{i})^2 \right), \quad (11)$$

$$\mu_{F^l}^{i} = \frac{1}{M_l} \sum_{j=1}^{M_l} F_{ij}^{l}, \quad \sigma_{F^l}^{i\,2} = \frac{1}{M_l} \sum_{j=1}^{M_l} (F_{ij}^{l} - \mu_{F^l}^{i})^2, \quad (12)$$

## 63. Synthetic to Real Adaptation withGenerative Correlation Alignment Networks

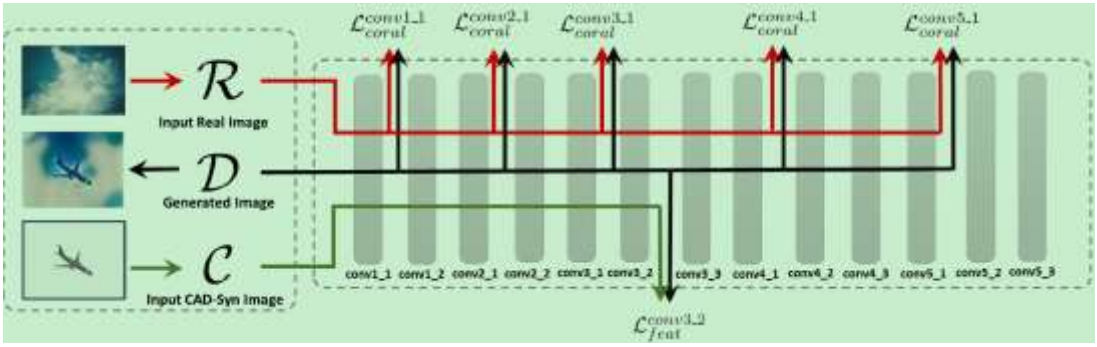
**主要思想：**

- 提出并实现了一种 Deep Generative Correlation AlignmentNetwork （ DGCAN)，通过生成具有自然特征统计特征的图像，使 CAD 合成域适应实际领域
- 利用 l2 损失来保存 CAD 模型在特征空间中的内容，并应用二阶 CORAL 损失减少区域差异，使该合成图像具有与真实图像相似的低层统计信息。

**主要方法：**

以 CAD 合成图像 C 和现实背景图像 R 作为输入并生成新的图像$\mathcal{D}$，$\mathcal{D}$与 CAD 图像包含相同的对象形状，且从背景图像中获取更加自然低层的特征分布。网络结构是基于 VGG-16，其中包括 13 卷积层，分为五组。在以下的层中使用$l^2$ loss $\mathcal{L}_{feat}^{\mathcal{X}_f}$和 CORAL loss $\mathcal{L}_{CORAL}^{\mathcal{X}_c}$：

$\mathcal{X}_f = \{conv3-2\}, \mathcal{X}_c = \{conv1-1,\ conv2-1,\ conv3-1,\ conv4-1,\ conv5-1\}$




- Shape Preserving loss

$\mathcal{H}^l(.)$是网络中第 l 层的特征矩阵，$\mathcal{H}_i^l(.)$是$\mathcal{H}^l(.)$的 i 维向量，$\mathcal{H}_{ij}^l(.)$是$\mathcal{H}_i^l(.)$的第 j 个值。

$$\mathcal{L}_{feat}^{\mathcal{X}_f} = \sum_{l \in \mathcal{X}_f} \left( \frac{\omega_f^l}{2\alpha^l} \sum_i \|\mathcal{H}_i^l(\mathcal{D}) - \mathcal{H}_i^l(\mathcal{C})\|_2^2 \right). \quad (1)$$

其中$w_f^l$是权重，$\alpha^l = N^l F^l$

- Naturalness loss

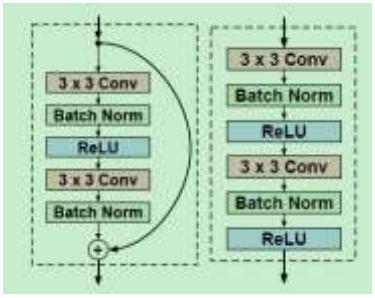
$Covs, Covt$是源域和目标域协方差的矩阵特征向量

$$\mathcal{L}_{coral}^{\mathcal{X}_c} = \sum_{l \in \mathcal{X}_c} \left( \frac{\omega_c^l}{4\alpha^{l^2}} \| Cov(\mathcal{H}^l(\mathcal{D})) - Cov(\mathcal{H}^l(\mathcal{R})) \|_F^2 \right)$$

$$Cov(\mathcal{H}^l(\mathcal{M})) = \frac{1}{N^l} \{ \mathcal{H}^l(\mathcal{M})^\top \mathcal{H}^l(\mathcal{M}) - \frac{1}{N^l} [(\mathbf{1}^\top \mathcal{H}^l(\mathcal{M}))^\top (\mathbf{1}^\top \mathcal{H}^l(\mathcal{M}))] \} \quad (4)$$

## 64. Perceptual Losses for Real-Time Style Transferand Super-Resolution: Supplementary Material

残差网络训练速度更快，但这两个网络最终达到类似的训练损失。图 2 还显示了经过训练的残差和非剩余网络的样式转换示例；两者都类似于对输入图像应用类似的转换。


$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{y \sim p_{data}(y)}[\log D(y)] + \mathbb{E}_{x \sim p_{data}(x), z \sim p_z(z)}[\log(1 - D(G(x, z))]. \quad (2)$$