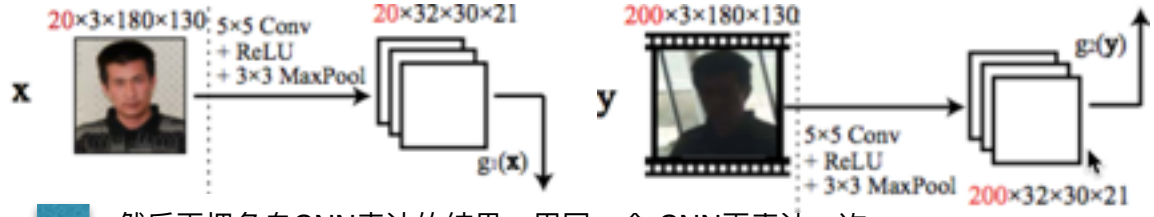
# Cross-Domain Visual Matching via Generalized Similarity Measure and Feature Learning

Liang Lin; Guangrun Wang; Wangmeng Zuo; Feng Xiangchu; Lei Zhang
IEEE Transactions on Pattern Analysis and Machine Intelligence
Year: 2016, Volume: PP, Issue: 99
Pages: 1 - 1, DOI: 10.1109/TPAMI.2016.2567386

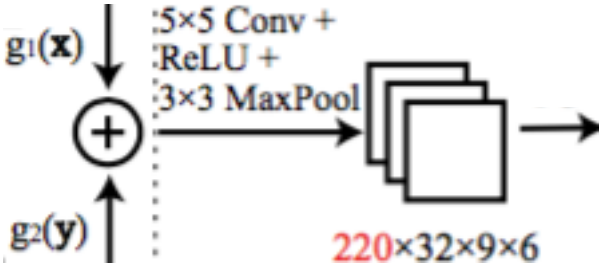
有道是世间人事物各从其类，但是偏偏有痴男怨女远跨越边界坠入爱河。那这种情况下如何衡量他们爱的多深呢？比如说现在有一张证件照的人脸照片，是端端正正坐着拍出来的，要跟一个视频监控环境下随意拍到的模糊照片做对比，怎么办呢？来自中山大学的Liang Lin教授说，好办，我们用深度学习就好啦！深度学习包治百病。

首先我们有个证件照X，还有一个视频监控拍到的模糊照片Y

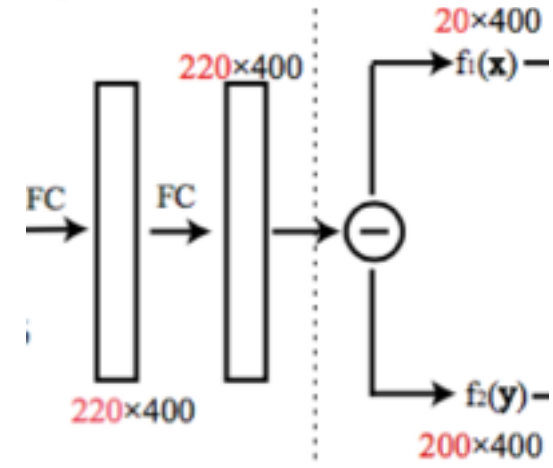

为了对比他们的相似度，我们先各自用一个CNN 网络对这两个图片进行表达




然后再把各自CNN表达的结果，用同一个 CNN再表达一次




然后把这个结果送入同样的两层全联接层，然后把X跟Y各自的结果拿出来




就得到了X跟Y 的深度表达了！

$$\mathbf{f}_1(\mathbf{x}), \mathbf{f}_2(\mathbf{y})$$

然后就是如何从这俩表达矢量中算出来他们的相似度呢？首先我们把他们俩矢量合成一个矢量，还加上一个1

$$[\mathbf{f}_1(\mathbf{x})^T \ \mathbf{f}_2(\mathbf{y})^T \ 1]$$

然后相似度就是这个大矢量跟一个矩阵相乘的结果了

$$S(\mathbf{f}_1(\mathbf{x}), \mathbf{f}_2(\mathbf{y})) = [\mathbf{f}_1(\mathbf{x})^T \ \mathbf{f}_2(\mathbf{y})^T \ 1] \begin{bmatrix} \mathbf{A} & \mathbf{C} & \mathbf{d} \\ \mathbf{C}^T & \mathbf{B} & \mathbf{e} \\ \mathbf{d}^T & \mathbf{e}^T & f \end{bmatrix} \begin{bmatrix} \mathbf{f}_1(\mathbf{x}) \\ \mathbf{f}_2(\mathbf{y}) \\ 1 \end{bmatrix}$$

对于中间这个剧阵，我们可以对其中的各个子剧阵进行分解。首先这个相似度参数剧阵中的A和B是正定的，所以可以分解成

$$\mathbf{A} = \mathbf{L_A}^T \mathbf{L_A},$$
$$\mathbf{B} = \mathbf{L_B}^T \mathbf{L_B},$$

然后C未必是正定的，所以可以分解成

$$\mathbf{C} = -\mathbf{L_C^x}^T \mathbf{L_C^y}$$

把分解的结果代入，得到了

$$= \|\mathbf{L_A f}_1(\mathbf{x})\|^2 + \|\mathbf{L_B f}_2(\mathbf{y})\|^2 + 2\mathbf{d}^T \mathbf{f}_1(\mathbf{x})$$
$$- 2(\mathbf{L_C^x f}_1(\mathbf{x}))^T (\mathbf{L_C^y f}_2(\mathbf{y})) + 2\mathbf{e}^T \mathbf{f}_2(\mathbf{y}) + f.$$

干脆把这些跟x有关的都拼起来，做成一个矢量

$$\tilde{\mathbf{x}} \triangleq [\ \mathbf{L_A f}_1(\mathbf{x}) \quad \mathbf{L_C^x f}_1(\mathbf{x}) \quad \mathbf{d}^T \mathbf{f}_1(\mathbf{x})\ ]^T$$

对y来说也一样

$$\tilde{\mathbf{y}} \triangleq [\ \mathbf{L_B f}_2(\mathbf{y}) \quad \mathbf{L_C^y f}_2(\mathbf{y}) \quad \mathbf{e}^T \mathbf{f}_2(\mathbf{y})\ ]^T$$

然后定义三个剧阵，把这三项分别挑出来

$$\mathbf{P}_1 = [\ \mathbf{I}^{r \times r} \quad \mathbf{0}^{r \times (r+1)}\ ],$$
$$\mathbf{P}_2 = [\ \mathbf{0}^{r \times r} \quad \mathbf{I}^{r \times r} \quad \mathbf{0}^{r \times 1}\ ]$$
$$\mathbf{p}_3 = [\ \mathbf{0}^{1 \times 2r} \quad 1^{1 \times 1}\ ]^T,$$

那这个相似度函数就写成了

$$\tilde{S}(\mathbf{x}, \mathbf{y}) = (\mathbf{P}_1 \tilde{\mathbf{x}})^T \mathbf{P}_1 \tilde{\mathbf{x}} + (\mathbf{P}_1 \tilde{\mathbf{y}})^T \mathbf{P}_1 \tilde{\mathbf{y}} - 2(\mathbf{P}_2 \tilde{\mathbf{x}})^T \mathbf{P}_2 \tilde{\mathbf{y}}$$
$$+ 2\mathbf{p}_3^T \tilde{\mathbf{x}} + 2\mathbf{p}_3^T \tilde{\mathbf{y}} + f.$$

对于这个相似度，我们有很多参数要学习，包括CNN的参数，还有相似度参数剧阵，A, B, C。那为了学习这些参数呢，我们得有一些训练集，比如现在有一对训练样本，

$$\{\mathbf{x}_i, \mathbf{y}_i\}$$

是一个证件照，一个视频监控照片，然后我们还有一个 标号说明他们是不是同一个人。如果是同一个人的照片，就标记成－1，否则就是＋1。

$$\ell_i = \ell(\mathbf{x}_i, \mathbf{y}_i) = \begin{cases} -1, & c(\mathbf{x}) = c(\mathbf{y}) \\ 1, & \text{otherwise} \end{cases}$$

其实这个相似度函数是个差异度的函数。如果是同一个人，我们自然希望这个差异度得到的值是小的，小到比－1还小。否则，是同一个人，我们就希望他是大的，大到比＋1还大

$$\tilde{S}(\mathbf{x}_i, \mathbf{y}_i) \begin{cases} < -1, \text{ if } \ell_i = -1 \\ \geq 1, \quad \text{otherwise} \end{cases}$$

$$1 - \ell_i \tilde{S}(\mathbf{x}_i, \mathbf{y}_i) < 0$$

这样的话，我们可以搞一个惩罚函数，惩罚那些没满足这些条件的。

$$(1 - \ell_i \tilde{S}(\mathbf{x}_i, \mathbf{y}_i))_+$$

有很多对这样的样本，就有了训练集合 $\mathcal{D} = \{(\{\mathbf{x}_i, \mathbf{y}_i\}, \ell_i)\}_{i=1}^N$
那我们就吧他们的损失函数加起来，得到目标函数

$$\min \sum_{i=1}^{N} (1 - \ell_i \tilde{S}(\mathbf{x}_i, \mathbf{y}_i))_+$$

然后为了避免过学习的问题,我们还要让参数的norm2 也最小

$\mathbf{W}$ 是深度网络的参数

$\mathbf{\Phi} = (\mathbf{L_A}, \mathbf{L_B}, \mathbf{L_C^x}, \mathbf{L_C^y}, \mathbf{d}, \mathbf{e}, f)$ 是相似度函数的参数

$$\min \lambda \|\mathbf{W}\|^2 + \mu \|\mathbf{\Phi}\|^2$$

加起来就得到了最终的优化问题

$$\min_{\mathbf{\Omega} = (\mathbf{W}, \mathbf{\Phi})} \sum_{i=1}^{N} (1 - \ell_i \tilde{S}(\mathbf{x}_i, \mathbf{y}_i))_+ + \lambda \|\mathbf{W}\|^2 + \mu \|\mathbf{\Phi}\|^2$$

$$\Omega = \min_{(\mathbf{W},\Phi)} \sum_{i=1}^{N} \{ 1 - \ell_i [ (\mathbf{P}_1 \tilde{\mathbf{z}}^{j_{i,1}})^T \mathbf{P}_1 \tilde{\mathbf{z}}^{j_{i,1}} + (\mathbf{P}_1 \tilde{\mathbf{z}}^{j_{i,2}})^T \mathbf{P}_1 \tilde{\mathbf{z}}^{j_{i,2}} - 2(\mathbf{P}_2 \tilde{\mathbf{z}}^{j_{i,1}})^T \mathbf{P}_2 \tilde{\mathbf{z}}^{j_{i,2}} + 2\mathbf{p}_3^T \tilde{\mathbf{z}}^{j_{i,1}} + 2\mathbf{p}_3^T \tilde{\mathbf{z}}^{j_{i,2}} + f ] \}_+ + \lambda \|\mathbf{W}\|^2 + \mu \|\Phi\|^2$$

为了解这个问题呢，就用梯度下降算法。

$$\Omega = \Omega - \alpha \frac{\partial}{\partial \Omega} H(\Omega),$$

函数层数太多，那就用求导数对法则，从外层到内层一层层的求进去。
先从求 $\tilde{\mathbf{z}}^{j_{i,x}}$ 的导数入手

$$\frac{\partial L}{\partial \tilde{\mathbf{z}}^{j_{i,x}}}$$

求这个导数的一个障碍就是，这个损失函数是个hinge loss, 他妈的不可导啊。好办，就用先算一下是不是为零

$$\mathbf{1}_{\mathbf{z}^{j_{i,x}}}(\mathbf{z}^{j_{i,y}}) = 1 \quad \text{when} \quad \ell_{j_{i,x},j_{i,y}} \tilde{S}(\mathbf{z}^{j_{i,x}}, \mathbf{z}^{j_{i,y}}) < 1$$

$$\mathbf{1}_{\mathbf{z}^{j_{i,x}}}(\mathbf{z}^{j_{i,y}}) = 0 \quad \text{Otherwise}$$

然后就求这个损失函数内部的导数就好了。内部是二次的

$$\frac{\partial L}{\partial \tilde{\mathbf{z}}^{j_{i,x}}} = -\sum 2\mathbf{1}_{\mathbf{z}^{j_{i,x}}}(\mathbf{z}^{j_{i,y}}) \ell_{j_{i,x},j_{i,y}} (\mathbf{P}_1^T \mathbf{P}_1 \tilde{\mathbf{z}}^{j_{i,x}} - \mathbf{P}_2^T \mathbf{P}_2 \tilde{\mathbf{z}}^{j_{i,y}} + \mathbf{p}_3)$$

然后依次的求进去就可以了。

启发：

1. 不同模态的数据也可以直接做比较，用CNN映射到同一个空间就可以了。
2. 没映射到一个空间，也可以比较，设计一个metric matrix就可以了。
3. 不同模态的数据，可以先分别用CNN，然后用一个一样的CNN, 然后分别拿出来做为表达。