# 基于图像检索的跨视角地理定位

李媛媛

**摘要**

在带有地理标记的航空图像和地面查询图像的地理定位任务上，由于航空和地面之间视角的巨大变化，传统的图像匹配不满足于现有的图像检索任务。得益于近年来深度学习的成功，提出了CVM-Net用于基于跨视角图像的地对空地理定位任务。具体来说，该网络是基于Siamese架构对匹配任务进行度量学习。首先使用全卷积层提取局部图像特征，然后使用NetVLAD将其编码为全局图像描述符。作为训练过程的一部分，还引入了一个简单有效的加权软间距排序损失函数，不仅加快了训练收敛速度，而且提高了最终的匹配精度。实验结果表明，在两个现有的基准数据集上，提出的网络显著优于最新的方法。

**关键词：**图像检索；跨视角；地理定位；Siamese架构

## 1 引言

基于图像的地理定位任务由于其在自动驾驶和增强现实等领域的潜在应用，在过去几年中引起了计算机视觉界的广泛关注。传统的基于图像的地理定位通常是在数据库中的查询和地理标记的参考图像都取自地面视图的情况下进行的。这类方法的主要缺点之一是通常从人口流量大的地方中获得的数据库图像，照片收集有可能偏向著名旅游区例，因而没有对区域进行全面的覆盖。因此，地对地的地理定位方法往往在参考图像不可用的位置失效。相比之下，从卫星和无人机等具有鸟瞰视角的设备上拍摄的航空图像则密集覆盖地球。因此，将地面图像与航空影像进行匹配逐渐成为一种越来越流行的地理定位方法[1]。然而，由于地面和航空影像之间的视角剧烈变化，跨视角匹配仍然具有挑战性。这导致使用传统手工特征如SIFT[2]和SURF[3]的跨视角匹配失败。

随着最近深度学习在许多计算机视觉任务中的成功，现有的跨视角图像匹配工作[4]大多采用卷积神经网络 (CNN) 学习表示，用于地面和航空图像之间的匹配。为了弥补较大的视角差异，Vo 和Hays[5]使用一个额外的网络分支来估计朝向，并利用航拍图像的多个可能方向来寻找两个视角之间的最佳匹配角度。这种方法在训练和测试中都会产生很大的开销。相比之下，作者的工作通过使用全局VLAD描述符来避免开销，该描述符在位置识别任务中表现出对大视角和场景变化的不变性。具体来说，作者在卷积神经网络上嵌入NetVLAD层[6]，以提取对较大视角变化具有不变性的描述符。图1展示了作者方法。其核心思想是NetVLAD将从CNN中获得的局部特征进行聚合，形成与局部特征位置无关的全局表示。

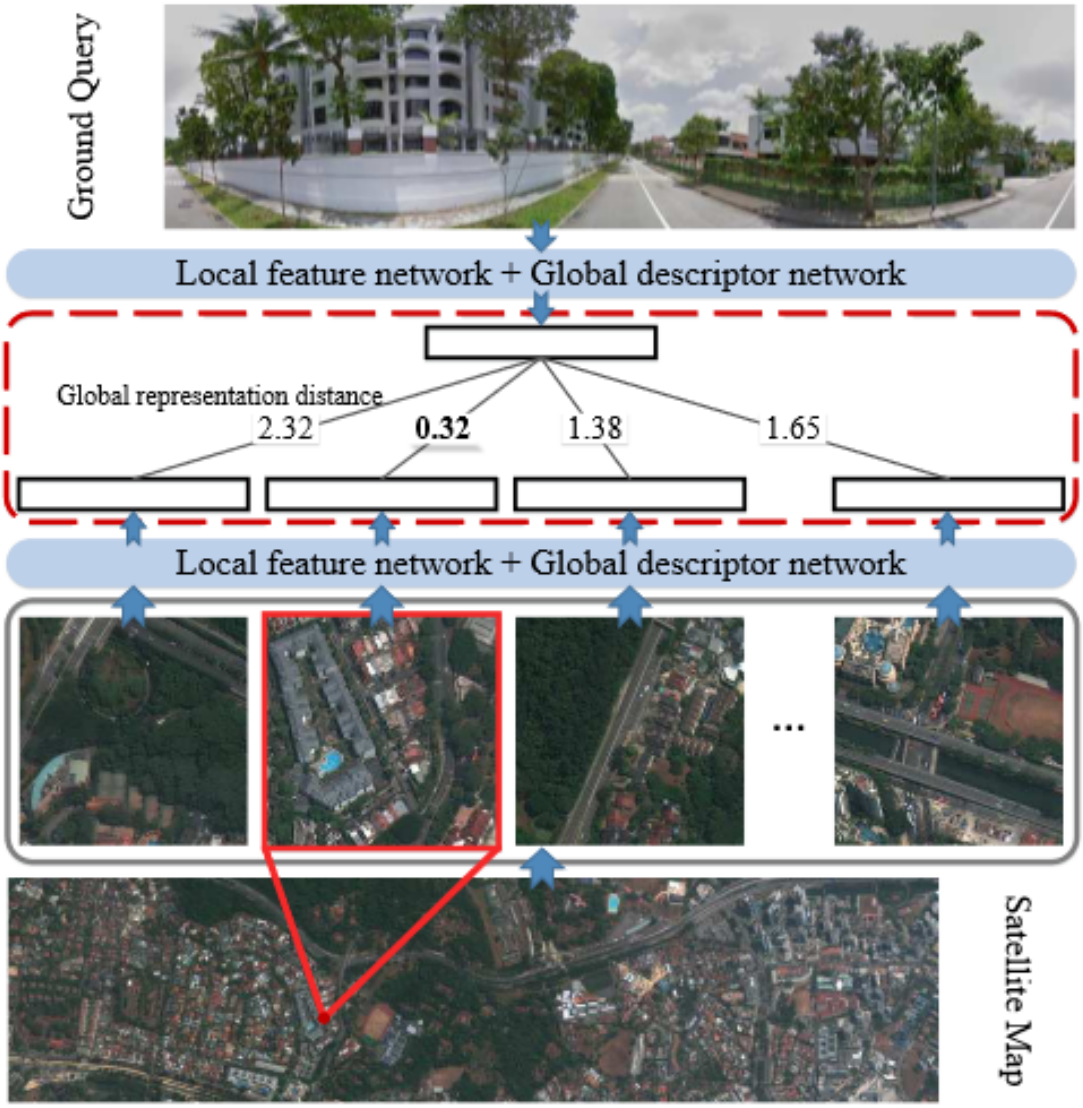


图 1: 基于图像的地对空地理定位问题的图例说明，以及提出的框架

在本文中，作者提出了一个强大的网络架构：CVM-Net，用于基于跨视角图像的地对空地理定位。具体来说，将 NetVLAD 层与 Siamese 网络结合，共同学习跨视角图像匹配的鲁棒表示。CVM-Net 学习局部特征并形成对大视角变化具有不变性的全局描述符，用于地对空地理定位。作为训练过程的一部分，还引入了一种新的加权软间距排序损失，不仅加快了训练收敛，而且提高了最终的检索精度。此外，这种新的加权软边际可以嵌入到三重和四重损失中。大量实验结果表明，提出的框架显著优于所有最先进的方法，特别是在全景 CVUSA 数据集[7]上。

## 2 相关工作

现有的关于估计查询地面图像地理位置的工作大多采用图像匹配或图像检索技术。这些工作可以根据特征的类型进行分类。

### 2.1 手工特征

在早期阶段，使用计算机视觉领域常用的传统特征进行跨视角图像匹配[8]。然而，由于视点的巨大差异，同一位置的航拍图像和地面视图图像出现了很大的差异。这导致与传统局部特征的直接匹配失效。因此，许多方法将地面影像弯曲到自顶向下的视角，以改进特征匹配。在航拍图像倾斜且建筑物立面可见的情况下，通过立面图斑匹配可以实现地理定位。

### 2.2 可学习特征

由于深度学习方法在图像/视频分类和识别任务中被证明是非常成功的，一些应用被引入到跨视角图像匹配和检索领域。Workman 和 Jacobs 在 ImageNet 和 Places 上训练的 AlexNet 模型上进行了实验。结果表明，用于普通图像分类的深度特征明显优于手工设计的特征。随后，Workman 等人通过在航拍枝干上训练卷积神经网络，进一步提高了匹配精度。Vo 和 Hays 对现有的分类和检索网络进行了深入的实验，包括二分类网络、Siamese 网络和 Triplet 网络。通过新颖的软间隔三元组损失和耗尽的小批量数据训练策略，他们在检索精度上取得了显著的提升。另一方面，Zhai 等人提出了一种弱监督训练网络来获取卫星图像的语义布局。这些布局被用作图像描述符从数据库中进行检索。

图像检索最重要的部分是找到一个好的图像描述子，该描述子具有判别性和快速的比较性。Sivic 和 Zisserman 提出视觉词袋描述符，将一组局部特征聚合成视觉单词的直方图，即全局描述符。表明该描述符具有部分视点和遮挡不变性，并且优于局部特征匹配。Nister 和 Stewenius 创建了树形结构词汇来支持更多的视觉单词。Jegou 等人提出了 VLAD 描述符[9]。它们不是直方图，而是将局部特征的残差聚合到聚类中心。在此基础上，Arandjelovic 等人提出了 VLAD 的可学习层，即 NetVLAD 可以嵌入到深度网络中进行端到端的训练。在他们的扩展文章中，说明了 NetVLAD 优于多个全连接层、最大池化和 VLAD。由于 NetVLAD 的性能优越，网络采用 NetVLAD 层。

### 2.3 图像检索损失

本文的工作也与通过深度网络进行度量学习有关。在图像检索任务中应用最广泛的损失函数是最大间隔三元组损失[10]，它强制正对的距离小于负对的距离，这个阈值需要仔细选择。为了克服这个问题，Vo 和 Hays 提出了一种被证明是有效的软间隔三元组损失。由于三元组损失对不相关对没有约束，在训练过程中减少类内变异时会导致类间变异较小。为了缓解这个问题，提出了四元组和角度损失，进一步改进三元组网络的训练和图像检索的性能。

## 3 本文方法

与基于图像的地对空地理定位[11]的现有工作类似，目标是从给定的地理标记卫星图像 (见图 1) 数据库中找到与查询地面图像最接近的匹配，即跨视角图像检索。为此，提出了 CVM-Net。

### 3.1 方法概述

为了学习卫星和地面图像之间的联合关系，采用了在图像匹配和检索任务中被证明非常成功的 Siamese-like 架构。特别地，框架包含两个相同架构的网络分支。每个分支由局部特征提取和全局描述子生成两部分组成。在第一部分中，CNNs 用于提取局部特征。在第二部分中，将局部特征编码为一个全局描述符，该描述符在大视角变化时具有不变性。为此，采用 VLAD 描述符，在每个 CNN 分支上嵌入 NetVLAD 层。

### 3.2 局部特征提取

使用全卷积网络 (FCN)$f^L$ 来提取图像的局部特征向量。对于卫星图像 $I_s$，其局部特征的集合由 $U_s = f^L(I_s;\Theta_s^L)$ 给出，其中 $\Theta_s^L$ 为卫星分支 FCN 的参数。对于一幅地面图像 $I_g$，其局部特征集合 $U_g = f^L(I_g;\Theta_g^L)$，其中 $\Theta_g^L$ 为地面视图分支 FCN 的参数。在这项工作中，使用 AlexNet[12]和 VGG16[13]的卷积部分作为 $f^L$ 来比较网络的结果。

### 3.3 全局描述符生成

将 FCN 得到的局部特征向量集输入到 NetVLAD 层，得到全局描述符。NetVLAD 是 VLAD 的可训练深度网络版，它将局部特征向量的残差聚合到各自的聚类质心，生成全局描述符。质心和距离度量是 NetVLAD 中可训练的参数。在本文中，尝试了两种策略，即 CVM-Net-I 和 CVM-Net-II，将卫星和地面图像的局部特征向量聚合成各自的全局描述符，在一个公共空间进行相似性比较。

**CVM-Net-I**

如图 2 所示，为每个分支使用单独的 NetVLAD 层来生成卫星和地面图像各自的全局描述符。图像的全局描述符可以表示为 $v_i = f^G(U_i; \Theta_i^G),$ ，其中 $i \in \{s, g\}$ 表示卫星或地面分支。$\Theta_i^G$ 中有两组参数：(1) K 个簇的质心 $C_i = \{c_{i,1}, \ldots, c_{i,K}\}$；(2) 每个簇的距离度量 $Wi, k$。两个 NetVLAD 中的聚类数目均设置为相同。每个 NetVLAD 层对处于同一空间的视图 $v_s$ 和 $v_g$ 分别生成一个 VLAD 向量，即全局描述符，用于直接进行相似性比较。为了保持低计算复杂度，在将 VLAD 向量送入损失函数进行端到端训练之前对其进行降维，或者将其用于相似性比较。

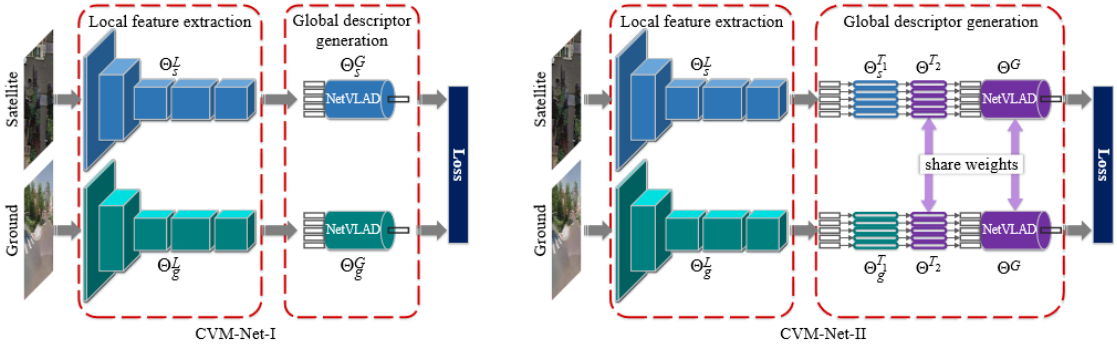


图 2: CVM-Nets 概述

除了判别能力外，在 Siamese-like 架构中共同训练的具有相同簇数的两个 NetVLAD 层，能够输出两个位于共同空间的 VLAD 向量。给定一组局部特征向量 $U = \{u_1, \ldots, u_N\}$，则 VLAD 向量 v 的第 k 个元素为 $V(k) = \sum_{j=1}^{n} \bar{a}_k u_j (u_j - c_k)$，其中 $Bara_k u_j$ 为由距离度量参数和输入局部特征向量确定的软分配权重。如上式所示，每个质心的描述向量是残差对质心的求和。两个视图质心的残差在一个新的公共空间中，独立于两个质心的定义域。因此，它们可以看作是在一个共同的“残差”空间中，关于两个视图中的质心对。卫星和地面视图描述符的比较是质心比较。这使得两个视图的 VLAD 描述符具有可比性。图 3 展示了这一概念的说明。

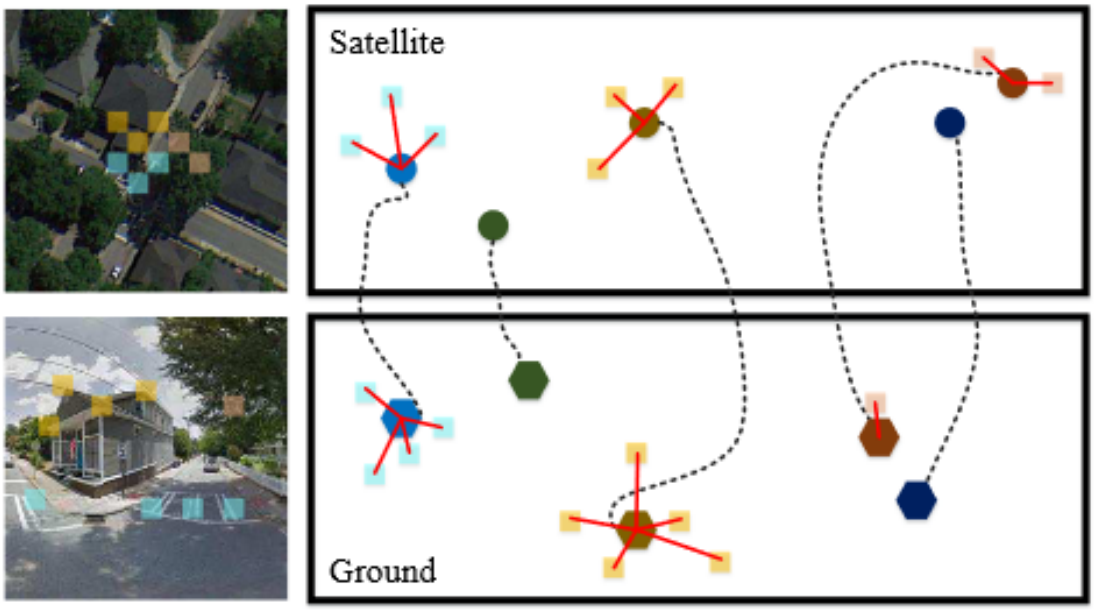


图 3: NetVLAD 实现跨视图匹配的说明

CVM-Net-I 的完整模型如图 2 所示。卫星图像的全局描述符由 $\upsilon_s = f^G(f^L(I_s;\Theta_s^L);\Theta_s^G)$ 给出，地面图像由 $\upsilon_g = f^G(f^L(I_g;\Theta_g^L);\Theta_g^G)$ 给出。两个分支具有相同的结构和不同的参数。最后，通过全连接层降低两个视图全局描述符的维度。

**CVM-Net-II**

与 CVM-Net-I 中结构相似的两个独立网络不同，作者提出了第二个网络——CVM-Net-II，该网络在 Siamese 架构中具有一些共享权重。图 2 展示了 CVM-Net-II 的架构。具体来说，提取局部特征 $U_s$ 和 $U_g$ 的 CNN 层数保持不变。然后将这些局部特征通过两个全连接层——第一层具有独立权重 $\Theta_s^{T1}$ 和 $\Theta_g^{T1}$，第二层具有共享权重 $\Theta^{T1}$。两个全连接层后的特征 $U'_s$ 和 $U'_g$ 由 $\mu'_{s,j} = f^T(\mu_{s,j};\Theta_s^{T1},\Theta^{T2});\mu'_{g,j} = f^T(\mu_{g,j};\Theta_g^{T1},\Theta^{T2})$ 给出。其中 $\mu_{s,j} \in U_s,\ \mu_{g,j} \in U_g\ and\ \mu'_{s,j} \in U'_s,\ \mu'_{g,j} \in U'_g$。最后，将变换后的局部特征送入共享权重为 $\Theta^G$ 的 NetVLAD 层。卫星和地面图像的全局描述符由 $\upsilon_s = f^G(U'_s;\Theta^G);\upsilon_g = f^G(U'_g;\Theta^G$ 给出。CVM-Net-II 的完整模型如图 2 所示。在 CVM-Net-II 网络中采用了权重共享，因为在许多 Siamese 网络架构[14]中，权重共享已经被证明可以改善度量学习。

### 3.4 加权软间距排序损失

三元组损失常被用作目标函数来训练用于图像匹配和检索任务的深度网络。三元组损失的目标是学习一个比负例更接近所选锚点的正例的网络。最简单的三元组损失是最大间距三元组损失：$\mathcal{L}_{max} = \max(0, m + d_{pos} - d_{neg})$，其中 $d_{pos}$ 和 $d_{neg}$ 是所有正例和负例到所选锚点的距离。m 是间间距，文献[15]表明，为了得到最好的结果，需要仔细选择 m。软间距三元组损失的提出，避免了在三元组损失中需要确定间距：$\mathcal{L}_{soft} = \ln(1 + e^d)$，其中 $d = d_{pos} - d_{neg}$。使用软间距三元组损失来训练 CVM-Nets，但是注意到这种损失会导致收敛速度变慢。为了提高收敛速度，作者提出了一个加权软间距排序损失，它通过一个系数 $\alpha$ 来衡量 $\mathcal{L}_{soft}$ 中的 d：$\mathcal{L}_{weighted} = \ln(1 + e^{\alpha d})$。当 $\alpha = 1$ 时，加权软间距排序损失变为软间距三元组损失。通过实验观察到，随着 $\alpha$ 的增加，收敛速度和结果都有所改善。损失的梯度随着 $\alpha$ 的增加而增加，这可能导致网络更快地提高权重，从而减少较大的误差。

作者提出的损失也可以嵌入到其他具有三元组损失成分的损失函数中。四元组损失[16]是三元组损失的改进版本，它也试图使不相关的负对进一步远离正对。四元组损失为 $\mathcal{L}_{quad} = \max(0, m_1 +$

$d_{pos} - d_{neg}) + \max(0, m_2 + d_{pos} - d^*_{neg})$。其中 m1 和 m2 为边距，$d^*neg$ 为选择的正例、负例和锚例集合之外的另一个例子的距离。同时也注意到，加权软间距分量不再需要间距。加权四元组损失为：$\mathcal{L}_{quad\square weighted} = \ln(1 + e^{\alpha(d_{pos} - d_{neg})}) + \ln(1 + e^{\alpha(d_{pos} - d^*_{neg})})$。

# 4 复现细节

## 4.1 与已有开源代码对比

在已有开源代码的基础上，改进原有的模型，对其骨干网络进行替换，将原开源代码中使用的 VGG16 网络分别替换为 ResNet 和 GoogleNet。为了新的骨干网络能够与原有的模型进行适配，需要先查看原网络参数的配置，具体查看原骨干网络第一层的输入大小和原骨干网络最后一层的输出大小，并修改有关参数，修改完之后再进行骨干网络的替换，最后再检查新的骨干网络的输入输出是否与原骨干网络的输入输出相匹配。部分代码如图所示：

```
def __fc(self, name, net, units, with_activation=True):
    with tf.variable_scope(name):
        input_units = net.get_shape()[-1]
        w = tf.get_variable('w', shape=[input_units, units])
        b = tf.get_variable('b', shape=[units])
        net = tf.add(tf.matmul(net, w), b)
        if with_activation:
            net = tf.nn.relu(net)
        return net

def __conv2d(self, name, net, output_channels, window_size, stride_size, padding='SAME', with_activation=True):
    with tf.variable_scope(name):
        input_channels = net.get_shape()[-1]
        filter = tf.get_variable('w', shape=[window_size, window_size, input_channels, output_channels])
        bias = tf.get_variable('b', shape=[output_channels])
        net = tf.nn.bias_add(tf.nn.conv2d(input=net, filter=filter,
                                          strides=[2, stride_size, stride_size, 2],
                                          padding=padding), bias)
        if with_activation:
            net = tf.nn.relu(net)
        return net
```

图 4: 代码片段 1

```
def __max_pool(self, name, net, window_size, stride_size, padding='SAME'):
    with tf.variable_scope(name):
        net = tf.nn.max_pool(net, ksize=[2, window_size, window_size, 2],
                             strides=[2, stride_size, stride_size, 2],
                             padding=padding)
        return net

def __avg_pool(self, name, net, window_size, stride_size, padding='SAME'):
    with tf.variable_scope(name):
        net = tf.nn.avg_pool(net, ksize=[2, window_size, window_size, 2],
                             strides=[2, stride_size, stride_size, 2],
                             padding=padding)
        return net
```

图 5: 代码片段 2

```
def inception(self, name, net,
              branch1_output_channels,
              branch2_reduce_output_channels, branch2_output_channels,
              , branch3_output_channels,):
    with tf.variable_scope(name):
        with tf.variable_scope("branch1"):
            net1 = self.__conv2d('conv1', net, branch1_output_channels, 2, 1)
        with tf.variable_scope("branch2"):
            tmp_net = self.__conv2d('conv1', net, branch2_reduce_output_channels, 1, 1)
            net2 = self.__conv2d('conv2', tmp_net, branch2_output_channels, 3, 1)
        with tf.variable_scope("branch3"):
            tmp_net = self.__max_pool('pool1', net, 3, 1)
            net4 = self.__conv2d('conv1', tmp_net, branch3_output_channels, 2, 2)
        with tf.variable_scope("concat"):
            net = tf.concat([net1, net2, net3, net4], axis=-1)
        return net
```

图 6: 代码片段 3

```
class Resnet():
    def __init__(self, filters, strides=1, residual_path=False):
        super(Resnet, self).__init__()
        self.filters = filters
        self.strides = strides
        self.residual_path = residual_path
        self.c1 = Conv2D(filters, (2, 2), strides=strides, padding="same", use_bias=False)
        self.b1 = BatchNormalization()
        self.a1 = Activation("sigmoid")
        if residual_path:
            self.down_c1 = Conv2D(filters, (1, 1), strides=strides, padding="same", use_bias=False)
            self.down_b1 = BatchNormalization()
        self.a2 = Activation("sigmoid")

    def call(self, inputs):
        residual = inputs
        x = self.c1(inputs)
        x = self.b1(x)
        x = self.a1(x)
        if self.residual_path:
            residual = self.down_c1(inputs)
            residual = self.down_b1(residual)
        out = self.a2(y + residual)
        return out
```

图 7: 代码片段 4

## 4.2 实验环境搭建

| Name | Version |
|---|---|
| Python | 3.6 |
| tensorflow-gpu | 1.12.0 |
| CUDA | 9.0 |
| CUDNN | 7.3.1 |

图 8: 实验环境

### 4.3 创新点

（1）在已有的开源代码上进行实验，通过调整网络模型有关参数后，检索性能与原文相比有所提升。

（2）编写 ResNet 代码块，进行骨干网络的替换并调整有关参数，检索性能与原文相比有所提升。

（3）编写 GoogleNet 代码块，进行骨干网络的替换并调整有关参数，检索性能与原文相比有所提升。

## 5 实验结果分析

将原有的 VGG16 网络和替换的 ResNet 和 GoogleNet 分别在 CVUSA 数据集上进行实验，实验结果如图所示：

| Backbone | CVM-NET-I | CVM-NET-II |
|---|---|---|
| VGG16 | 91.40% | 87.20% |
| VGG16 (ours) | 91.60% | 87.50% |
| ResNet (ours) | 91.90% | 87.80% |
| GoogleNet (ours) | 92.10% | 87.60% |

图 9: 实验结果

（1）对于 VGG16 网络，通过调整学习率等参数，召回率有一定的提升，但总体效果与原文相差不大。

（2）对于 ResNet，引入了更深层的神经网络结构，提取到了更深层的图像特征，使得特征描述符更好地应用于图像匹配中，因而在 CVM-NET-I 和 CVM-NET-II 两个方案中，召回率都有所提升。

（3）对于 GoogleNet，神经网络结构更为复杂，在 CVM-NET-I 方案中召回率进一步提升，但在 CVM-NET-II 方案中，召回率比 ResNet 稍有下降。

## 6 总结与展望

作者提出了两种跨视角匹配网络 CVM-NET-I 和 CVM-NET-II，能够将地面视角图像与卫星图像进行匹配，从而实现跨视角图像定位。并且引入了加权软间距排序损失，并证明它显著加快了训练速度，提高了网络的性能。通过在大型数据集上的实验，虽然 CVM-NET-I 方案性能显著提升，但 CVM-NET-II 方案的性能仍有待提升。

作者提出的 CVM-Nets 也可以训练用于其他跨领域图像检索任务，例如将手绘草图匹配到自然照片、白天和夜晚的图像、不同风格的绘画等。此外，提出的网络可以扩展到一般的跨领域信息检索任务。例如，在单词到图像的检索中，也可以用 Word2Vec 模型分支替换局部图像特征提取组件。

## 参考文献

[1] M.Bansal, K.Daniilidis, H.Sawhney. Ultra-wide baseline facade matching for geo-localization[J]. European Conference on Computer Vision, 2012: 1.

[2] D.G.Lowe. Distinctive image features from scale-invariant keypoints[J]. International Journal of Computer Vision, 2004, 60(2): 1.

[3] H.Bay, T.Tuytelaars, L.Van.Gool. Surf: Speeded up robust features[J]. European Conference on Computer Vision, 2006: 1.

[4] S.Workman, N.Jacobs. On the location dependence of convolutional neural network features[J]. IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015: 1, 2.

[5] N.N.Vo, J.Hays. Localizing and orienting street views using overhead imagery[J]. European Conference on Computer Vision, 2016: 1, 2, 3, 4, 5, 6, 8.

[6] R.Arandjelovic, P.Gronat, A.Torii, et al. Netvlad:Cnn architecture for weakly supervised place recognition[J]. IEEE Conference on Computer Vision and Pattern Recognition, 2016: 2, 3, 4.

[7] M.Zhai, Z.Bessinger, S.Workman, et al. Predicting ground-level scene layout from aerial imagery[J]. IEEE Conference on Computer Vision and Pattern Recognition, 2017: 1, 2, 3, 5, 6, 7, 8.

[8] M.Noda, T.Takahashi, D.Deguchi, et al. Vehicle ego-localization by matching in-vehicle camera images to an aerial image[J]. Asian Conference on Computer Vision Workshops, 2010: 2.

[9] H.Jegou, M.Douze, C.Schmid, et al. Aggregating local descriptors into a compact image representation [J]. IEEE Conference on Computer Vision and Pattern Recognition, 2010: 2, 3.

[10] A.Hermans, L.Beyer, B.Leibe. In defense of the triplet loss for person re-identification[J]. CoRR,abs/1703.07737, 2017: 2, 4.

[11] S.Workman, R.Souvenir, N.Jacobs. Wide-area image geolocalization with aerial reference imagery[J]. IEEE International Conference on Computer Vision, 2015: 1, 2, 3, 5, 6.

[12] A.Krizhevsky, I.Sutskever, G.E.Hinton. Imagenet classification with deep convolutional neural networks[J]. Advances in Neural Information Processing Systems, 2012: 2, 3, 5, 6.

[13] K.Simonyan, A.Zisserman. Very deep convolutional networks for large-scale image recognition[J]. CoRR, abs/1409.1556, 2014: 3, 5, 6.

[14] S.Chopra, R.Hadsell, Y.LeCun. Learning a similarity metric discriminatively, with application to face verification[J]. IEEE Conference on Computer Vision and Pattern Recognition, 2005: 4.

[15] J.Hays, A.A.Efros. Im2gps: estimating geographic information from a single image[J]. IEEE Conference on Computer Vision and Pattern Recognition, 2008: 1.

[16] W.Chen, X.Chen, J.Zhang, et al. Beyond triplet loss: A deep quadruplet network for person re-identification[J]. IEEE Conference on Computer Vision and Pattern Recognition, 2017: 2, 4, 6.