VIGOR: Cross-View Image Geo-localization beyond One-to-one Retrieval 论文复现

张清望

摘要

基于图像的地理定位其目的是通过在带有 GPS 标记的参考数据库中找到最相似的图像来确定查询图像的位置。最近的工作在城市规模的数据集上取得了令人惊讶的高检索精度。但是，这些结果依赖于这样的假设，即存在精确地以任何查询图像的位置为中心的参考图像，这不适用于实际情况。VIGOR 缩短传统数据集与现实世界任务的差距，不再局限于简单的一对一跨视图图像匹配，而是采用一对多的模式。本工作在 VIGOR 的基础上，将官方采用 Tensorflow 实现的代码，使用 PyTorch 复现，并进行实验验证。同时，更换新的骨干网络让网络性能有所提升，基于训练得到的图像全局描述符矩阵，实现了一个简单的跨视图图像地理定位系统。

关键词：跨视图地理定位；定位系统；VIGOR；PyTorch

1 引言

基于图像的地理定位其目的是通过在带有 GPS 标记的参考数据库中找到最相似的图像来确定查询图像的位置。传统的基于图像的地理定位，查询数据集和参考数据集均是取自地面视图。这种方法的主要缺点之一是通常获得的数据库图像，没有对该区域的全面覆盖。这是因为图形集最有可能偏向著名的旅游区。因此，地对地地理定位方法往往在参考图像缺失的位置运行失败。相比之下，从具有鸟瞰图的设备拍摄的航拍图像，例如卫星和无人机图像，密集地覆盖着地球。因此，将地面照片与航拍图像匹配逐渐成为一种越来越流行的地理定位方法。该技术可以应用在自动驾驶，辅助导航，增强现实等领域。然而，由于地面和航空图像之间的视图发生了巨大变化，物体的外观特征差异明显，因此跨视图匹配仍然具有挑战性。

最近的一些工作，通过特征聚合和样本挖掘策略可以显著提高跨视图图像匹配的性能。当地面视图图像的方向信息可以获取时 (由基于手机 phone 的指南针提供)，最先进的方法可以实现超过 80% 的 top-1 检索精度[1]，这表明了在现实环境中准确地理定位的可能性。但是，现有数据集 CVUSA，CVACT 只是假设每个查询地面视图图像都具有一个相应的参考航空视图图像，其中心在查询图像的位置处精确对齐。但是作者认为这对于现实世界中的应用程序是不切实际的，因为查询图像可能出现在感兴趣区域中的任意位置，并且应在查询出现之前捕获参考图像。在这种情况下，不能保证完全对齐的一对一对应关系。因此，论文作者提出了一个新的基准数据集 (VIGOR)，以在更现实的环境中评估跨视图地理定位。新的数据集在这项工作和先前的研究之间产生了两个基本差异：超越一对一，超越检索。超越一对一，即不再局限于简单的一对一跨视图图像匹配，而是采用一对多的模式，在训练过程中利用多个参考图像计算损失。超越检索，图像检索只能提供图像级的定位，由于在作者的数据集中不保证中心对齐，因此在检索之后，进一步采用图像内校准来预测检索到的图像内部查询位置的偏移量。综上，基于该论文的对跨视图图像匹配的开创性，以及与本人研究方向的高度重合，选择复现这篇论文。

2 相关工作

2.1 跨视图地理定位

跨视图地理定位的早期作品[2-5]的检索准确性较低，这主要是因为两个视图之间存在明显的外观差距，并且度量学习技术较差。通过量身定制的特征提取器和修改的损失函数，Hu 等人[6]展示了使用端到端深度神经网络实现精确定位的可能性。最近的几种方法[1,7]旨在通过利用 GANs[8]和极变换[9]来减少域间隙。Regmi 等[7]提出使用条件 GAN 从地面视图查询生成合成的空中视图图像，并采用特征融合以实现更好的性能。SAFA[1]进一步利用几何先验知识，对查询图像应用极坐标变换，并用特征聚合块替换全局池化。如果给出方向信息，则 CVUSA[10] 上[1]的 top-1 精度几乎 90%。探索度量学习技术 (例如硬样本挖掘策略) 的其他方法[11-12]在流行的数据集上也显示出有希望的结果，并且它们不受几何假设的限制。但是，这些方法都没有考虑超出图像级别检索或多个参考图像进行训练的子图像级别的定位。

2.2 跨视图地理定位数据集

目前已经提出了许多用于跨视图地理定位的数据集[2-4,10,13]。Lin 等[2]考虑卫星图像和土地覆盖属性以进行跨视图地理定位。6,756 张地视图图像和 182,988 张航拍图像是从南卡罗来纳州查尔斯顿收集的。尽管航空图像是密集采样的，但它们迫使两个视图之间存在一对一的对应关系，并且无法根据距离进行评估。原始 CVUSA[4]是一个庞大的数据集，其中包含来自美国多个城市的 100 万多个地面和航空图像。Zhai 等[10]进一步利用相机的外部参数，通过扭曲全景图来生成对齐的对，从而产生 35,532 图像对用于训练，8,884 图像对用于测试。此版本的 CVUSA 是最近研究中使用最广泛的数据集[1,6-7,12,14]，如果未指定，作者将其称为 CVUSA。Vo[15]由来自美国 11 个城市的约 100 万个图像对组成。作者随机收集街景全景图，并从每个全景图中生成几种裁剪，以及从 Google 地图中空间对齐的航空图像。与 CVUSA 类似，CVACT[13]也由对齐的全景图和带有方向信息的航空影像组成。它有 35,532 图像对训练和 92,802 对测试。简而言之，所有这些数据集都考虑一对一的检索，并且没有一个数据集提供原始的 GPS 数据以米为单位进行定位评估。

3 本文方法

3.1 VIGOR 数据集

问题陈述。给定一个感兴趣的区域 (AOI)，作者的目标是通过将其与航空参考图像进行匹配来定位该区域中的任意街景查询。为了保证任何可能的查询被至少一个参考图像覆盖，参考航拍图像必须提供 AOI 的无缝覆盖。如图1(a) 所示，粗略采样的参考图像 (黑色方格) 不能提供 AOI 的完全覆盖，并且任意查询位置 (红星) 可能位于参考样本之间的区域中。即使查询位置 (黄星) 位于参考航空图像的边缘，该参考图像也仅与中心位于查询位置的图像共享部分 (最多一半) 场景，这可能无法提供足够的信息与其他负面参考图像区分开来。这些查询可以通过添加额外的重叠样本 (绿色框) 来覆盖。如图1(b) 所示，如果查询位置 (红星) 位于 $L\times L$ 航空图像的中心区域 (黑色虚线框)，则将查询和参考图像定义为彼此的正样本。中心区域以外的其他查询 (蓝星) 被定义为半阳性样本。为了保证任何任意查询具有一个正参考图像，作者建议对沿纬度和经度两个方向具有 50% 重叠的航空图像进行密集采样，如图1(c) 所示。通过这样做，AOI 中的任何任意查询位置 (红星) 都被四个参考图像 (尺寸 $L\times L$) 覆盖。

绿色框表示正参考，其他三个半正参考表示为蓝色框。正参考图像被认为是事实，因为它具有最接近查询的 GPS，并且包含与查询图像共享的最多对象。红色框表示完全对齐的航空图像。基于如图1(b)所示的正性和半正性的定义，可以很容易地看到，所有正性参考图像具有大于与完全对齐的参考 0.39 的 IOU(交集比并集)(参见图1(d))。0.62 典型阳性样本的 IOU(相对于中心的偏移等于 $(\pm L/8, \pm L/8)$)。半阳性样本和比对参考之间的 IOU 落在 $[1/7{\approx}0.14, 9/23{\approx}0.39]$。

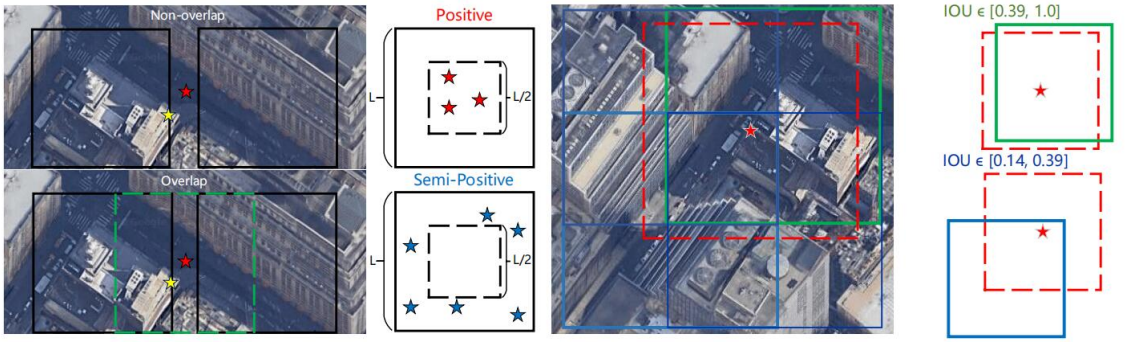

图 1: 提出数据集的抽样策略，星星表示查询位置

数据收集。如图2所示，作者使用 Google Maps 静态 API 收集了覆盖四个城市 (即纽约市 (曼哈顿)，旧金山，芝加哥和西雅图) 中心区域的 90,618 个航空图像，作为 AOI。然后使用 Google 街景静态 API 收集 238,696 街景全景图像，在大多数街道上的缩放级别 2。全景图的所有 GPS 位置在作者的数据集中都是唯一的，样本之间的典型间隔约为 30 m。作者对原始全景图进行数据平衡，以确保每个航空影像都有不超过 2 个正全景图。此过程为地理定位实验提供了 105,214 的全景图。此外，大约 4% 的航空图像不覆盖全景图。作者将它们保留为分散注意力的样本，以使数据集更加现实和具有挑战性。卫星图像的缩放级别为 20，地面分辨率约为 0.114 m。航空视图和地面视图的原始图像尺寸分别为 640×640 和 2048×1024。提供了用于鸟瞰图和地面视图图像的工业级 GPS 标签，用于米级评估。然后根据方向信息移动全景图，以使北部位于中间。图 3 示出了一对航空和街景图像之间的方向对应的示例。

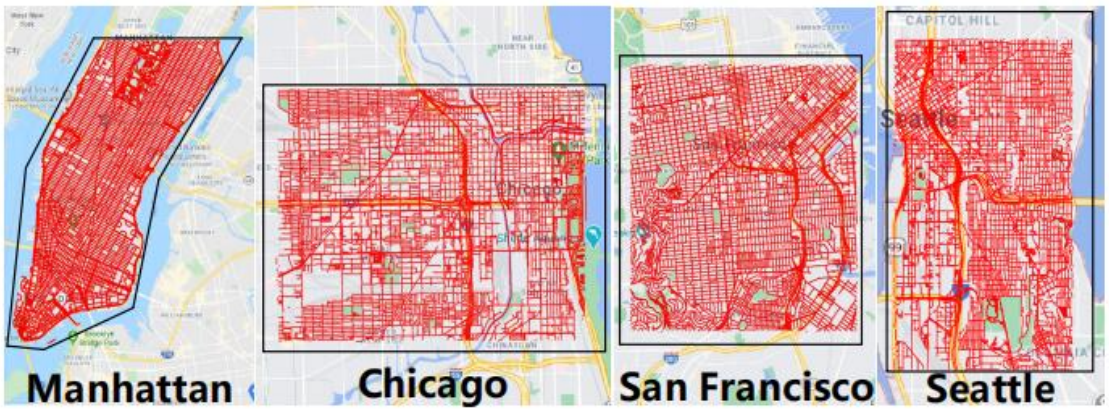

图 2: 四个城市的航空图像覆盖范围 (黑色多边形) 和全景图 (红点) 的分布

3.2 本文方法概述

为了在建议的数据集上获得令人满意的结果，重要的是采用最先进的技术来建立强大的基线。因此，作者采用了 SAFA (空间感知特征聚合)[1]的特征聚合模块，并采用了[12]的全局负挖掘策略。特征聚合。SAFA[1]是极坐标变换、Siamese 主干和特征聚合块的组合。但是，极坐标变换假定地面视图 GPS 位于相应的鸟瞰图参考图像的中心，这在作者的情况下不适用。因此，在框架中只采用特征聚合。特征聚合块的主要思想是根据嵌入的位置重新加权。当查询图像的方向信息可用时，空间感知块可提供

显著的性能增益。整体框架结构如图3所示。

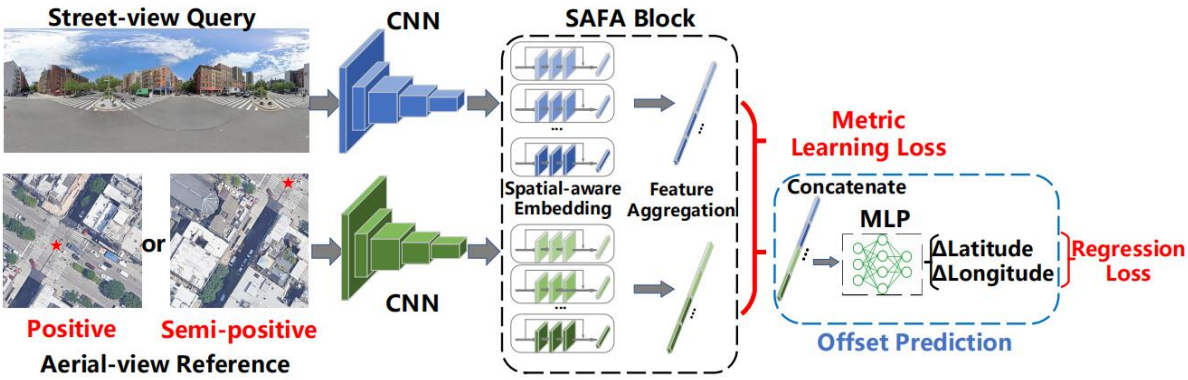

图 3: 端到端框架概述

挖掘策略。度量学习文献[16-18]已经揭示了在训练过程中挖掘硬样本的重要性，因为当大多数样本几乎不会对总损失做出贡献时，该模型将遭受收敛性差的困扰。对于跨视图地理定位,[12]进一步显示了挖掘全局硬样本而不是在小批次中挖掘的重要性。关键思想是建立一个先进先出的挖掘池，以缓存最难的样本的嵌入，并有效地刷新池以及反向传播。在小批次中，随机选择前半图像，并从挖掘池中选择相对于每个图像的全局硬样本，以形成批次的另一半。作者在基线中采用这种高效的全局挖掘策略[12]，以进一步提高其性能。

3.3 基于 IOU 的半正任务

如果仅考虑正样本，则可以通过标准度量学习来解决检索问题。对于基线，作者采用了[6]中提出的广泛使用的三元组损失：

$$L_{triplet} = \log(1 + e^{\alpha(d_{pos}+d_{neg})}) \tag{1}$$

其中 d_{pos} 和 d_{neg} 表示正负对的平方 l_2 距离。在具有 N 个地面视图和鸟瞰图图像对的小批处理中，作者使用穷举策略[17]来构建 $2N(N-1)$ 个三元组，从而充分利用所有输入图像。在[6]之后，作者对输出嵌入特征采用 l_2 归一化。除阳性样本外，还可以利用半阳性样本的监督信息。但是，简单地将半阳性样本分配为阳性会损害检索性能。对于街景查询，半正航拍图像仅包含查询位置的一小部分场景，因此半正样本与查询在特征嵌入空间上的相似性应该不会像正点样本那样高。一个直观的想法是根据参考图像和对齐图像之间的 IOU 分配相似性 (参见图1(d))。因此，基于 IOU 的半正分配损失表示为：

$$L_{IOU} = (\frac{S_{semi}}{S_{pos}} - \frac{IOU_{semi}}{IOU_{pos}})^2 \tag{2}$$

其中 S_{pos} 和 S_{semi} 表示嵌入空间中正对和半正对的余弦相似性。IOU_{pos} 和 IOU_{semi} 表示正对和半正对的 IOU。这种损失迫使嵌入空间中相似性的比率接近 $IOUs$ 的比率。在消融研究中还研究和评估了半阳性样本的其他分配策略。

3.4 偏移预测

通过检索到的 top-1 参考航拍图像，作者采用了一个辅助任务来在统一的框架中进一步完善航拍图像内部的定位 (参见图 4)。对于图像检索，在作者的数据集中检索到的参考图像之间的最小间隔是航空图像宽度的一半 $(L/2)$。为了实现更细粒度的定位，作者应用 MLP (多层感知器) 来预测查询位置相对于检索到的参考图像中心的偏移。如图 4 所示，辅助 MLP 由两个全连接的层组成，并将串联的嵌入特征作为输入。在这里，作者使用回归来生成预测，同时作者在实验中还提供了与分类的比较。

偏移回归损失公式为：

$$L_{offset} = (lat - lat^*)^2 + (lon - lon^*)^2 \tag{3}$$

其中 lat 和 lon 表示查询 GPS 位置相对于参考 GPS 在纬度和经度方向上的预测偏移，而 $lat^{\square}$ 和 $lon^{\square}$ 表示真实偏移。它们都被转换成米，并在训练时用 L 归一化。最终的混合损失函数由下式给出：

$$L_{hybrid} = L_{triplet} + L_{IOU} + L_{offset} \tag{4}$$

4 复现细节

4.1 与已有开源代码对比

原始开源代码 github 链接：https://github.com/Jeff-Zilence/VIGOR。主要参考的代码 github 链接：https://github.com/Jeff-Zilence/TransGeo2022。参考其数据集加载，网络训练模块及文件组织方式。主要贡献包括：将原本 Tensorflow1 实现的代码，使用 PyTorch 重新实现，填补该工作在 PyTorch 实现的空白；通过更换骨干网络在实验部分略有提升；实现一个简单的跨视图地理定位系统。

PyTorch 重新实现 VIGOR 模型。 SAFA[1]是极坐标变换、Siamese 主干和特征聚合块的组合。但是，极坐标变换假定地面视图 GPS 位于相应的鸟瞰图参考图像的中心，这在本文方法的情况下不适用。因此，在框架中只采用特征聚合如图。特征聚合块的主要思想是根据嵌入的位置重新加权。当查询图像的方向信息可用时，空间感知块可提供显著的性能增益。该模块部分核心代码对比如图4，图5所示。

```
def SAFA(x_sat, x_grd, dimension=8, trainable=True, original=False):
    with tf.variable_scope('vgg_grd'):
        vgg_grd = Vgg16()
        grd_local = vgg_grd.build(x_grd)
        grd_local_out = grd_local
        grd_local = tf.nn.max_pool(grd_local, [1, 2, 2, 1], [1, 2, 2, 1], padding='SAME')
    batch, g_height, g_width, channel = grd_local.get_shape().as_list()

    grd_w = spatial_aware(grd_local, dimension, trainable, name='spatial_grd')
    grd_local = tf.reshape(grd_local, [-1, g_height * g_width, channel])

    grd_global = tf.einsum('bic, bid -> bdc', grd_local, grd_w)
    grd_global = tf.reshape(grd_global, [-1, dimension * channel])

    with tf.variable_scope('vgg_sat'):
        vgg_sat = Vgg16()
        sat_local = vgg_sat.build(x_sat)
        sat_local_out = sat_local
        sat_local = tf.nn.max_pool(sat_local, [1, 2, 2, 1], [1, 2, 2, 1], padding='SAME')
    batch, s_height, s_width, channel = sat_local.get_shape().as_list()

    sat_w = spatial_aware(sat_local, dimension, trainable, name='spatial_sat')
    sat_local = tf.reshape(sat_local, [-1, s_height * s_width, channel])

    sat_global = tf.einsum('bic, bid -> bdc', sat_local, sat_w)
    sat_global = tf.reshape(sat_global, [-1, dimension * channel])
    if original:
        return tf.nn.l2_normalize(sat_global, dim=1), tf.nn.l2_normalize(grd_global,
                                                                         dim=1), sat_global, grd_global, sat_local_out, grd_local_out
    return tf.nn.l2_normalize(sat_global, dim=1), tf.nn.l2_normalize(grd_global, dim=1), sat_local, grd_local
```

图 4: 原文开源代码 (Tensorflow)

```
def forward(self, x, atten=None, indexes=None):

    output = self.conv1(x)
    output = self.maxpool(output)
    output = self.conv2_x(output)
    output = self.conv3_x(output)
    output = self.conv4_x(output)
    output = self.conv5_x(output)
    x = output
    x_dist = self.max_pooling_along_channels(x)
    x_dist = x_dist.reshape(x.shape[0], x.shape[2] * x.shape[3])
    x_dist = self.spatial_aware_importance_generator[0](x_dist)
    x_dist = x_dist.reshape(x_dist.shape[0], x.shape[2], x.shape[3])
    x_dist = x_dist.view(x_dist.shape[0], 1, x_dist.shape[1], x_dist.shape[2])
    x_dist = x * x_dist
    x_dist = x_dist.reshape(x.shape[0], x.shape[1] * x.shape[2] * x.shape[3])
    for i in range(1, 4):
        x_mid = self.max_pooling_along_channels(x)
        x_mid = x_mid.reshape(x.shape[0], x.shape[2] * x.shape[3])
        x_mid = self.spatial_aware_importance_generator[i](x_mid)
        x_mid = x_mid.reshape(x.shape[0], x.shape[2], x.shape[3])
        x_mid = x_mid.view(x_mid.shape[0], 1, x_mid.shape[1], x_mid.shape[2])
        x_mid = x * x_mid
        x_mid = x_mid.reshape(x.shape[0], x.shape[1] * x.shape[2] * x.shape[3])
        x_dist = torch.cat((x_dist, x_mid), dim=1)
    x = self.head(x_dist)
    return x
```

图 5: 本文复现代码 (PyTorch)

更换骨干网络进行实验。原文采用 VGG 作为其骨干，复现过程尝试 ResNet 作为骨干网络，核心代码如图6所示。

```
def __init__(self, block, num_block, num_classes=100):
    super().__init__()

    self.in_channels = 64

    self.conv1 = nn.Sequential(
        nn.Conv2d(3, 64, kernel_size=7, padding=3, stride=2, bias=False),
        nn.BatchNorm2d(64),
        nn.ReLU(inplace=True))
    self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2)
    self.conv2_x = self._make_layer(block, 64, num_block[0], 1)
    self.conv3_x = self._make_layer(block, 128, num_block[1], 2)
    self.conv4_x = self._make_layer(block, 256, num_block[2], 2)
    self.conv5_x = self._make_layer(block, 512, num_block[3], 2)
    self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))
```

图 6: ResNet 骨干网络部分核心代码

跨视图地理定位系统。在原文中，只关注于对模型的相关实现，而没有对于定位系统的实现。在本文复现工作中，从零开始搭建跨视图地理定位系统，通过模型获得图像数据集的全局描述符后，设计简单易用的 UI 界面，使得用户可以通过几次点击操作完成跨视图地理定位的体验。相关核心代码

如图7所示。

```
def init():
    ybar=tk.Scrollbar(root,orient='vertical')
    tree = ttk.Treeview(root,height=12,selectmode='browse',yscrollcommand=ybar.set)
    parent=[]
    for name in grd_path:
        parent.append(data_root + "NewYork/panorama/" + name)
    parent.reverse()
    for i in range(len(parent)):
        tree.insert("",0,parent[i],text=(parent[i][parent[i].rfind("/")+1:]),values=parent[i])
    def selectTree(event):
        for item in tree.selection():
            item_text = tree.item(item, "values")
            global gpath
            gpath=item_text[0]
            if(gpath.find('.')>=0):
                img = Image.open(gpath)
                img = img.resize((350, 250),Image.ANTIALIAS)
                photo = ImageTk.PhotoImage(img)
                label=tk.Label(root,image=photo)
                label.place(x=30,y=320)
                root.mainloop()
    # 选中行
    tree.bind('<<TreeviewSelect>>', selectTree)
    tree.place(x=30,y=15,width=350,height=250)
    ybar['command']=tree.yview
    ybar.place(x=380,y=15,height=250)
    comboxlist=ttk.Combobox(root,textvariable=comvalue) #初始化
    comboxlist["values"]=("VIGOR","SAFA")
    comboxlist.current(0)  #选择第一个
    comboxlist.bind("<<ComboboxSelected>>",select_dist)  #绑定事件,(下拉列表框被选中时，绑定go()函数)
    comboxlist.place(x=30,y=280)
```

图 7: 跨视图地理定位系统部分核心代码

4.2 实验环境搭建

本文实验所需硬件环境如表1所示。所需软件环境如表2所示。PyTorch 1.8.1 GPU 版本是本文进行相关实验的相关组件，其安装步骤只要包括：(1) 通过 anaconda 官网安装 anaconda3；(2) 为本文项目新建 Python3.8 的虚拟环境，命令为：conda create python=3.8；(3) 输入命令：conda install pytorch==1.8.1 torchvision==0.9.1 torchaudio==0.8.1 cudatoolkit=10.2 -c pytorch 安装 PyTorch 1.8.1 GPU 版本；(4) 若步骤 3 失败，可以通过离线方式安装，在链接：https://download.pytorch.org/whl/torch_stable.html。下载所需的 PyTorch 版本，然后上传到服务器进行安装；(5) 最后通过 Python 命令：print(”torch 版本:”, torch.__version__) 输出 PyTorch 版本号，验证是否安装正确。

表 1: 实验硬件环境说明

名称	详细参数
操作系统	Linux 4.15.0-156-generic
内存大小	500GB
硬盘容量	3TB
CPU	Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz
GPU	Tesla V100-SXM2-32GB

4.3 界面分析与使用说明

本文实现的跨视图地理定位系统主界面如图8所示。

表 2: 实验软件环境说明

软件包名称	版本号
Python	3.8
PyTorch	1.8.1
tensorboard	2.10.1
torchvision	0.9.1
tqdm	4.64.1

图 8: 跨视图地理定位系统主界面

如上图所示，对界面模块进行相应说明。界面区域 1：用户可以在此处选择文件上传至系统，完成定位操作；复位选项可以对系统进行复位操作增强系统鲁棒性；帮助选项可以跳转到帮助页码，用户可以获取用户手册。界面区域 2：展现所有的查询图像名称，用户可以直接选择想要定位的图像。界面区域 3：左侧选择框中，用户可以选择不同的方法进行定位任务；右侧定位按钮，用户通过点击对当前所选图片进行地理定位。界面区域 4：用于实时展示当前所选的查询图像。界面区域 5：用于实时展示定位得到的最相似的前 50 张参考图像，并且在每张图像下方显示实际的距离。

4.4 创新点

（1）使用 PyTorch 复现官方提供的 Tensorflow 版本代码，填补该工作在 PyTorch 实现的空白。

（2）探究使用新的骨干网络在模型性能上取得提升。

（3）实现一个简单的跨视图地理定位系统，用户可以通过几次点击操作和系统交互，体验跨视图地理定位。

5 实验结果分析

实施细节。所有实验都是基于 PyTorch 部署的。在馈送到网络之前，将地面全景和鸟瞰图图像的大小分别调整为 640×320 和 320×320。采用 VGG-16[19]作为骨干特征提取器，下面[1]使用 8 个 SAFA 块。挖掘策略参数的设置与[12]中的相同。在[6]之后，作者将 $L_{triplet}$ 损失中的 α 设置为 10。Adam 优化器[20]的学习率为 10^{-5}。作者的方法首先用 $L_{triplet}$ 进行训练。然后，对于相同区域设置，在 30 个时期之后切换到混合损耗，对于跨区域设置，在 10 个时期之后切换到混合损耗。用于比较的基线仅用 $L_{triplet}$ 训练。

评估指标。在之前的工作[1,6]之后，作者首先以 top-k 的召回精度评估检索性能。对于每个测试查询，将检索其在嵌入空间中最接近的 k 个参考邻居作为预测。如果将地面对应航空图像包含在 top-k 个检索图像中，则认为一次检索是正确的。如果检索到的 top-1 参考图像覆盖了查询图像 (包括 ground-truth)，则将其视为命中，并且命中率也提供用于检索评估。此外，作者将 top-1 预测位置与地面真相查询 GPS 之间的真实距离计算为米级评估。

主要结果。在 VIGOR 数据集上，将原文中提出的 Tensorflow 实现与本文实现的 PyTorch 版本在相同区域和跨区域设置下进行了比较，用来验证本文 PyTorch 实现的正确性，实验结果如表3所示。其中 VIGOR 是原文的实验结果，Ours 是本文使用 PyTorch 复现得到的实验结果。由实验结果可以看出，本文复现得到的实验结果与原文报告的实验结果十分接近（因为深度神经网络初始化优化方法等实现细节的不确定性，所有存在合理的误差），可以判断本文使用 PyTorch 复现的代码是正确的。使用 ResNet 作为新的骨干网络替换原文采用的 VGG 网络得到实验结果如表4所示。由表中实验结果可以看出，采用 ResNet 作为新的骨干网络，在各项指标上较原文中的实验结果略有提升。将 VIGOR 方法在检索和米级评估方面都优于先前的方法，如图9所示。与基线相比，本文方法对 10 米级精度的相对改进 (见图9) 在相同区域设置中 124% (11.4%→25.5%)，而在跨区域设置中 121% (2.8%→6.2%)。

表 3: 复现实验结果与原文对比

	Same-Area				**Cross-Area**			
	Top-1	Top-5	Top-10	Top-1%	Top-1	Top-5	Top-10	Top-1%
VIGOR	41.1	65.8	-	98.4	11.0	23.6	-	80.2
Ours	41.07	65.81	74.05	98.37	11.02	23.56	30.76	80.22

表 4: 复现实验结果与原文对比

	Same-Area				**Cross-Area**			
	Top-1	Top-5	Top-10	Top-1%	Top-1	Top-5	Top-10	Top-1%
VIGOR	41.1	65.8	-	98.4	11.0	23.6	-	80.2
Ours(ResNet)	41.93	66.41	76.09	98.83	11.51	24.82	32.38	80.16

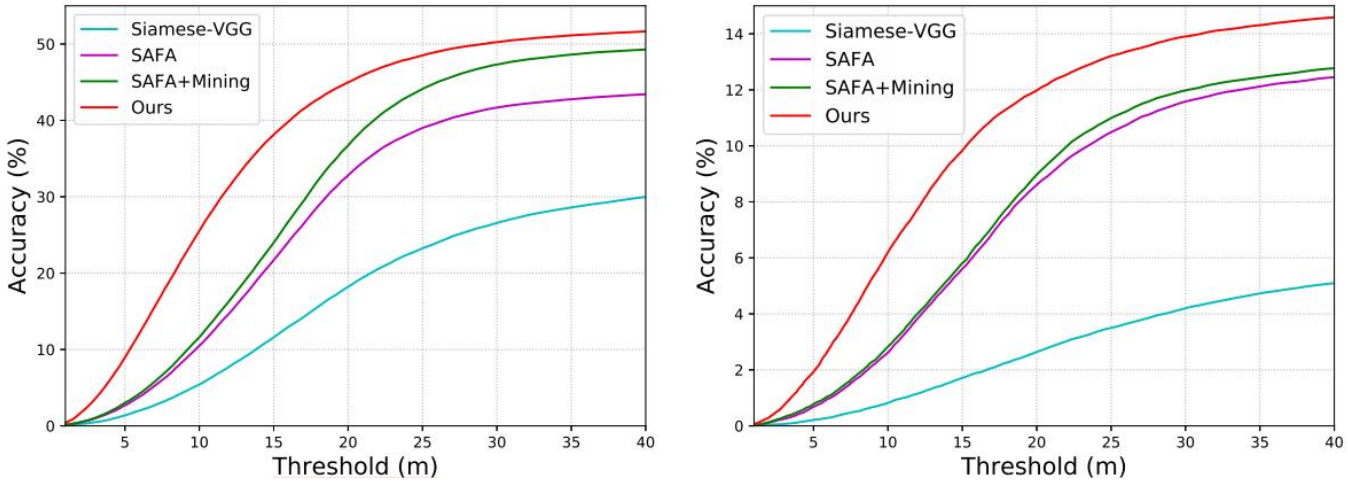

图 9: 不同的方法在同区域 (左) 和跨区域 (右) 米级定位精度实验结果

可视化分析。对本文实现的模型进行可视化化分析如图6所示。用户的检索示例如图11所示（更多检索示例参见附带的演示视频）。通过直观的可视化结果可以看出，本文实现的模型可以得到较好的检索结果。

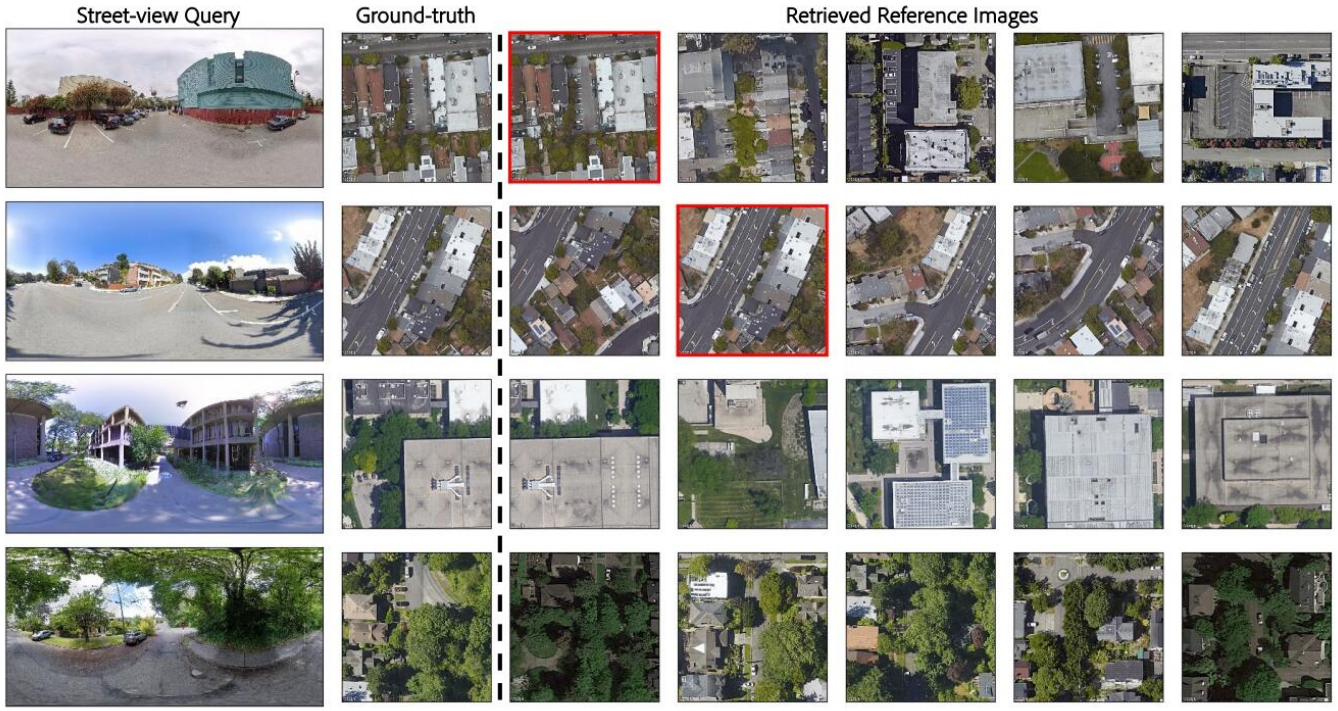

图 10: 可视化分析

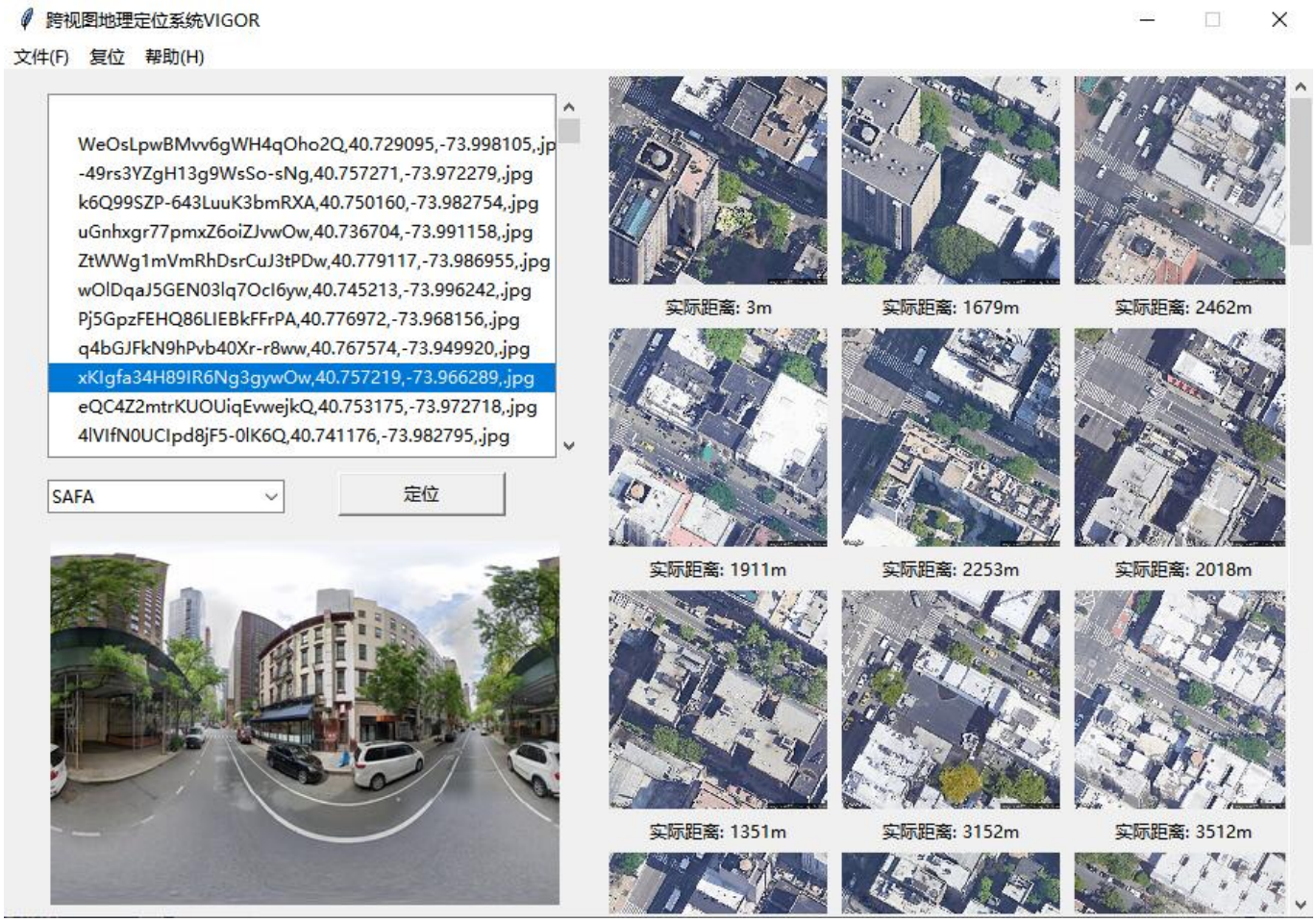

图 11: 用户界面检索示例

6 总结与展望

在本文中使用 PyTorch 完整复现了官方提供的 Tensorflow 版本的 VIGOR 模型，并进行相关实验验证了其正确性。同时探索更换新的骨干网络，对模型性能带来的增益。最后，实现了一个简单易用的跨视图图像地理定位系统。本文的复现依旧存在一些不足，如：在代码效率和可读性方面不高；代码结构之间耦合性较高。在下一步的工作中，希望提高代码效率，为代码增加注释提高其可读性，为代码结构解耦提高其灵活性。同时，希望探究更多的骨干网络或者尝试改进现有的网络结构，以待进一步提高网络性能。进一步美化界面设计及用户使用体验，使得系统更加易用好用。

参考文献

[1] SHI Y, LIU L, YU X, et al. Spatial-aware feature aggregation for image based cross-view geo-localization[J]. Advances in Neural Information Processing Systems, 2019, 32.

[2] LIN T Y, BELONGIE S, HAYS J. Cross-view image geolocalization[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013: 891-898.

[3] TIAN Y, CHEN C, SHAH M. Cross-view image matching for geo-localization in urban environments [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 3608-3616.

[4] WORKMAN S, SOUVENIR R, JACOBS N. Wide-area image geolocalization with aerial reference

imagery[C]//Proceedings of the IEEE International Conference on Computer Vision. 2015: 3961-3969.

[5] ZAMIR A R, SHAH M. Accurate Image Localization Based on Google Maps Street View[C]// DANIILIDIS K, MARAGOS P, PARAGIOS N. Computer Vision – ECCV 2010. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010: 255-268.

[6] HU S, FENG M, NGUYEN R M, et al. Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 7258-7267.

[7] REGMI K, SHAH M. Bridging the domain gap for ground-to-aerial image matching[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 470-479.

[8] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial networks[J]. Communications of the ACM, 2020, 63(11): 139-144.

[9] SHI Y, YU X, CAMPBELL D, et al. Where am i looking at? joint location and orientation estimation by cross-view matching[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 4064-4072.

[10] ZHAI M, BESSINGER Z, WORKMAN S, et al. Predicting ground-level scene layout from aerial imagery[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 867-875.

[11] CAI S, GUO Y, KHAN S, et al. Ground-to-aerial image geo-localization with a hard exemplar reweighting triplet loss[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 8391-8400.

[12] ZHU S, YANG T, CHEN C. Revisiting street-to-aerial view image geo-localization and orientation estimation[C]//Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021: 756-765.

[13] LIU L, LI H. Lending orientation to neural networks for cross-view geo-localization[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 5624-5633.

[14] VERDE S, RESEK T, MILANI S, et al. Ground-to-aerial viewpoint localization via landmark graphs matching[J]. IEEE Signal Processing Letters, 2020, 27: 1490-1494.

[15] VO N N, HAYS J. Localizing and orienting street views using overhead imagery[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. 2016: 494-509.

[16] MUSGRAVE K, BELONGIE S, LIM S N. A metric learning reality check[C]//Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16. 2020: 681-699.

[17] SCHROFF F, KALENICHENKO D, PHILBIN J. Facenet: A unified embedding for face recognition and clustering[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 815-823.

[18] SUH Y, HAN B, KIM W, et al. Stochastic class-based hard example mining for deep metric learning [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 7251-7259.

[19] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition [J]. arXiv preprint arXiv:1409.1556, 2014.

[20] KINGMA D P, BA J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.