

# 计算机视觉实验报告



姓名 学号

颜劭铭 162050127

杨铭 162050128

冉鹏 162050120

指导老师: 梁栋

南航·印象

计算机科学与技术学院/人工智能学院/软件学院

2023 年 5 月 18 日

# 目录

<b>1 低层视觉及传统方法: 暗通道先验及图像去雾霾</b>	<b>1</b>
1.1 简介 . . . . .	1
1.2 关键代码 . . . . .	4
1.3 实验设置 . . . . .	5
1.4 可视化及定量实验结果展示及说明 . . . . .	6
1.5 实验结果现象的原理性分析 . . . . .	7
1.6 结论与总结 . . . . .	8
<b>2 高层视觉及传统/深度学习方法: 语义分割 PSPNET/DeeplabV3+</b>	<b>8</b>
2.1 简介 . . . . .	8
2.2 关键代码 . . . . .	9
2.3 实验设置 . . . . .	9
2.4 PspNet 实验结果现象的原理性分析 . . . . .	9
2.5 DeeplabV3+ 实验结果现象的原理性分析 . . . . .	10
2.6 可视化及定量实验结果展示及说明 . . . . .	12
2.7 结论与总结 . . . . .	14
<b>3 OpenAI CLIP: Learning Transferable Visual Models From Natural Language Supervision (ICML-2021)</b>	<b>14</b>
3.1 简介 . . . . .	14
3.2 关键代码 . . . . .	14
3.3 实验设置 . . . . .	14
3.4 可视化及定量实验结果展示及说明 . . . . .	15
3.5 实验结果现象的原理性分析 . . . . .	16
3.6 结论与总结 . . . . .	18
<b>4 参考文献</b>	<b>21</b>

## 1 | 低层视觉及传统方法：暗通道先验及图像去雾

### 1.1 | 简介

雾霾是由空气中的灰尘和烟雾等小的漂浮颗粒产生的常见大气现象，这些漂浮的颗粒极大地吸收和散射光，导致拍摄图像质量下降。在雾霾影响下，视频监控，远程感应，自动驾驶等许多实际应用很容易受到威胁，检测和识别等高级计算机视觉任务很难完成。图像去雾的算法有很多种，但是主要分为两类：基于图像增强的去雾算法和基于图像复原的去雾算法。其中基于图像增强的去雾算法出发点是尽量去除图像噪声，提高图像对比度，从而恢复出无雾清晰图像，代表性方法有：直方图均衡化（HLE）、自适应直方图均衡化（AHE）等。而基于图像复原的去雾算法基本是基于大气退化模型，进行响应的去雾处理。这篇文章 Single Image Haze Removal Using Dark Channel Prior[3] 就是基于图像复原的去雾算法的代表性算法。

接下来对于论文中使用的主体方法进行介绍。

#### 1.1.1 | 暗通道先验

首先必须要提到的是暗通道先验的概念。这是通过对户外无雾图像的统计结果得来的，作者在论文中统计了 5000 多幅图像的特征，发现对于一幅彩色图像在绝大多数非天空的局部区域里，某一些像素总会有至少一个颜色通道具有很低的值。换句话说，该区域光强度的最小值是个很小的数。因此，对于任意的输入图像  $J$ ，它的暗通道可以用式子 1.1 表示：

$$J^{\text{dark}}(\mathbf{x}) = \min_{\mathbf{y} \in \Omega(\mathbf{x})} \left( \min_{c \in \{r, g, b\}} J^c(\mathbf{y}) \right) \quad (1.1)$$

其中式子中  $J^c$  表示彩色图像中的每个通道， $\Omega(\mathbf{x})$  表示以像素  $\mathbf{x}$  为中心的一个窗口。

而作者通过大量的统计得到的暗通道先验理论指出：

$$J^{\text{dark}} \rightarrow 0. \quad (1.2)$$

即对于无雾的非天空部分图像的暗通道值总是趋于 0。而相反的，对于有雾的图像来说其暗通道值不是趋于 0，如图 1.1 所示。

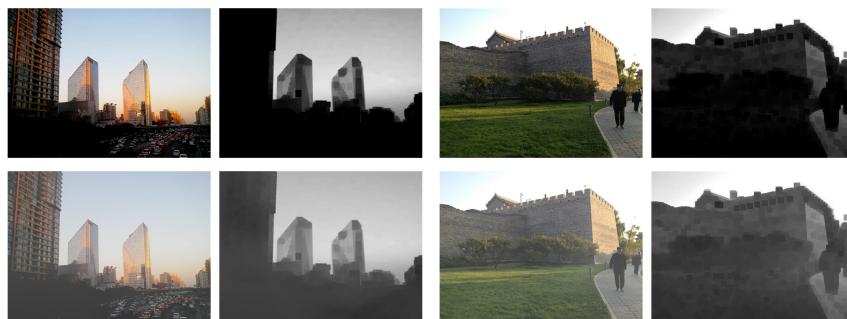


图 1.1：同一张图片有雾和无雾图像的暗通道

#### 1.1.2 | 利用暗通道先验去雾

其次，在计算机视觉和计算机图形中，广泛使用的雾图像生成模型如式子 1.3 所示：

$$\mathbf{I}(\mathbf{x}) = \mathbf{J}(\mathbf{x})t(\mathbf{x}) + \mathbf{A}(1 - t(\mathbf{x})) \quad (1.3)$$



其中,  $I(x)$  就是输入图像, 而  $J(x)$  则是要恢复的目标图像,  $A$  是全球大气光成分,  $t(x)$  则是透射率, 目前已知  $I(x)$ 。

将式1.3稍作处理, 变形为下式1.4:

$$\frac{I^c(x)}{A^c} = t(x) \frac{J^c(x)}{A^c} + 1 - t(x) \quad (1.4)$$

其中上标 C 表示  $R/G/B$  三个通道的意思。

首先假设在每一个窗口内透射率  $t(x)$  为常数, 定义他为  $\tilde{t}(x)$ , 并且  $A$  值已经给定, 然后对式1.4两边做两次最小值运算, 得到下式1.5:

$$\min_{y \in \Omega(x)} \left( \min_c \frac{I^c(y)}{A^c} \right) = \tilde{t}(x) \min_{y \in \Omega(x)} \left( \min_c \frac{J^c(y)}{A^c} \right) + 1 - \tilde{t}(x) \quad (1.5)$$

上式1.5中,  $J$  是目标的图像, 根据之前描述的暗通道先验理论有:

$$J^{\text{dark}}(x) = \min_{y \in \Omega(x)} \left( \min_c J^c(y) \right) = 0 \quad (1.6)$$

因此, 可推导出:

$$\min_{y \in \Omega(x)} \left( \min_c \frac{J^c(y)}{A^c} \right) = 0 \quad (1.7)$$

最后将式1.7带入式1.5中, 得到透射率  $\tilde{t}(x)$  的预估值如下式1.8所示

$$\tilde{t}(x) = 1 - \min_{y \in \Omega(x)} \left( \min_c \frac{I^c(y)}{A^c} \right). \quad (1.8)$$

作者在原论文中指出, 在现实生活中, 即使是晴天白云, 空气中也存在着一些颗粒, 因此, 看远处的物体还是能感觉到雾的影响, 另外, 雾的存在让人类感到景深的存在, 因此, 有必要在去雾的时候保留一定程度的雾, 这可以通过在式??中引入一个在  $[0,1]$  之间的调节因子  $\omega$ , 原论文中  $\omega = 0.95$ :

$$\tilde{t}(x) = 1 - \min_{y \in \Omega(x)} \left( \min_c \frac{I^c(y)}{A^c} \right) \quad (1.9)$$

对于式子1.4来说, 我们现在已知  $I(x)$ ,  $t(x)$ , 目标是  $J(x)$ , 但是我们还需要知道全球大气光成分值  $A$ 。作者在论文中指出, 我们可以借助暗通道图从输入图像中来获得该值。具体步骤如下所示:

- 1) 从暗通道图中按照亮度的大小取前 0.1% 的像素。
- 2) 在这些像素位置中, 对应原始有雾图像  $I$  中寻找对应的具有最高亮度的点的值, 作为  $A$  值。

在这个式子中还有一个问题是, 当透射率  $t$  的值较小时, 会导致目标图像  $J$  的值偏大, 从而导致图像整体曝光过高, 因此可以设置阈值  $t_0$ , 当  $t$  值小于  $t_0$  时, 令  $t = t_0$ , 因此, 最终的去雾公式如下1.10所示:

$$J(x) = \frac{I(x) - A}{\max(t(x), t_0)} + A \quad (1.10)$$

而如果简单地应用这个式子的话, 会发现恢复图像的局部会存在一些不协调或者没有进行去雾, 这是因为获得的透射率较为粗糙。而为了获得更为精细的透射率, 作者在文章中提出了 soft matting 的办法, 不过这个方法时间复杂度较高, 作者在后续又重新提出了新的方法 Guided Image Filtering 进行处理。



### 1.1.3 | 导向滤波

导向滤波是一种简单但是有着非常有效的边缘保留和平滑去噪能力的滤波器，它的定义如式子1.11所示：

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \quad (1.11)$$

其中  $I$  指的是引导图像， $p$  指的是输入图像， $q$  指的是输出图像。在刚刚的去雾算法中， $I$  即第一次得到的透射率图， $p$  即输入有雾图像， $q$  即输出去雾图像。

对式子1.11求导可得：

$$\nabla q = a \nabla I \quad (1.12)$$

由式子1.12可以发现，导向图  $I$  有梯度的地方，输出图像  $q$  就有梯度，即，他们拥有相同的边缘。在式子1.11的线性模型下，我们在保留边缘的同时还需要实现去噪，即：

$$q_i = p_i - n_i \quad (1.13)$$

其中  $n_i$  即噪声，因此由式子1.11和1.13可得到最小化的损失函数为，其中  $\epsilon a_k^2$  为惩罚项：

$$E(a_k, b_k) = \sum_{i \in \omega_k} \left( (a_k I_i + b_k - p_i)^2 + \epsilon a_k^2 \right) \quad (1.14)$$

对式1.14求解的结果如下：

$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon}, \quad (1.15)$$
$$b_k = \bar{p}_k - a_k \mu_k.$$

也就是说，当系数  $a_k, b_k$  如式子1.15所示时，整个导向滤波模型可以实现平滑去噪和边缘保留。

上述公式是针对一个像素点的，而当处理图像时，我们通常使用的是滑动窗口，因此将模型转变为以下形式：

$$q_i = \frac{1}{|\omega|} \sum_{k|i \in \omega_k} (a_k I_i + b_k) \quad (1.16)$$

而导向滤波同时还可以应用到图像去雾中代替 soft matting 精细化透射率  $t$ ，具体算法如图1.2所示。

```
Input: filtering input image p, guidance image I, radius r,  
       regularization ε  
Output: filtering output q.  
1: meanI = fmean(I)  
   meanp = fmean(p)  
   corrI = fmean(I.*I)  
   corrIp = fmean(I.*p)  
2: varI = corrI - meanI.*meanI  
   covIp = corrIp - meanI.*meanp  
3: a = covIp/(varI + ε)  
   b = meanp - a.*meanI  
4: meana = fmean(a)  
   meanb = fmean(b)  
5: q = meana*I + meanb  
/* fmean is a mean filter with a wide variety of O(N) time  
methods. */
```

图 1.2: Guided Filtering 运用到去雾图像



### 1.1.4 | 深度估计

由于雾会降低图像中的像素值，因此，暗通道先验认为当一幅图像受到空气散射的影响时，其暗通道与图像深度成反比。而整幅图像的暗通道描述了深度估计，即描述了场景中最远的像素被多少雾覆盖了。具体而言，这意味着估计场景中每个像素被遮盖的比例，从而估计每个像素的深度。

基于暗通道先验，可以推导出以下深度估计公式，其中  $t$  是透射率， $\beta$  是散射系数：

$$\text{depth} = \beta * \log(t) \quad (1.17)$$

### 1.2 | 关键代码

首先是暗通道计算的代码。

```
1 def get_dark_channel(self, img):
2     """计算暗通道"""
3     b, g, r = cv2.split(img)
4     min_rgb = cv2.min(cv2.min(r, g), b) # 计算每个像素的RGB通道中的最小值
5     kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2 * self.radius + 1, 2 * self.
radius + 1))
6     dark = cv2.erode(min_rgb, kernel) # 进行最小滤波以获取暗通道
7     return dark
8
```

然后是利用暗通道先验去雾的代码。

```
1 def estimate_atmospheric_light(self, img, dark):
2     """估计大气光A"""
3     [h, w] = img.shape[:2]
4     imgsize = h * w
5     Top_pixels = int(max(math.floor(h * w / 1000), 1)) # 计算暗通道中最亮的前0.1%像素数量
6
7     darkvec = dark.reshape(imgsize)
8     imgvec = img.reshape(imgsize, 3)
9
10    indexes = darkvec.argsort()
11    indexes = indexes[imgsize - Top_pixels::] # 选择对应于暗通道中最亮的前0.1%像素的索引
12
13    A = np.zeros(3)
14    for i in range(Top_pixels):
15        pixel = imgvec[indexes[i], :] # 获取所选索引处像素的RGB值
16        for channel in range(img.shape[2]):
17            if pixel[channel] > A[channel]:
18                A[channel] = pixel[channel] # 如果当前像素值大于存储值，用当前像素值更新存储值
19
20    return A
21
22 def estimate_transmission(self, img, A):
23     """估计透射率t"""
24     I_dark = np.empty(img.shape, img.dtype)
25
26     for channel in range(img.shape[2]):
27         I_dark[:, :, channel] = img[:, :, channel] / A[channel] # 将每个颜色通道除以相应的大气光值
28
29     transmission = 1 - self.omega * self.get_dark_channel(I_dark) # 根据公式，使用暗通道先验估计透射率
30     return transmission
```



```
30
31     def recover_image(self, img, t, A):
32         """恢复去雾图像"""
33         recovered_img = np.empty(img.shape, img.dtype)
34         t = cv2.max(t, self.t0) # 透射率取一个最小值
35
36         for channel in range(img.shape[2]):
37             recovered_img[:, :, channel] = (img[:, :, channel] - A[channel]) / t + A[channel]
38     ] # 恢复去雾图像
39     return recovered_img * 255
```

最后是利用导向滤波进行透射率精炼的代码。

```
1     def guided_filter(self, img, p):
2         """导向滤波"""
3         m_I = cv2.boxFilter(img, cv2.CV_64F, (self.r, self.r))
4         m_p = cv2.boxFilter(p, cv2.CV_64F, (self.r, self.r))
5         m_Ip = cv2.boxFilter(img * p, cv2.CV_64F, (self.r, self.r)) # 均值滤波
6         cov_Ip = m_Ip - m_I * m_p # 协方差
7
8         m_II = cv2.boxFilter(img * img, cv2.CV_64F, (self.r, self.r)) # 均值滤波
9         var_I = m_II - m_I * m_I # 方差
10
11        a = cov_Ip / (var_I + self.eps) # 系数 a
12        b = m_p - a * m_I # 系数 b
13
14        m_a = cv2.boxFilter(a, cv2.CV_64F, (self.r, self.r)) # 对系数 a 进行均值滤波
15        m_b = cv2.boxFilter(b, cv2.CV_64F, (self.r, self.r)) # 对系数 b 进行均值滤波
16
17        q = m_a * img + m_b # 计算导向滤波结果
18        return q
19
20    def refine_transmission(self, img, t):
21        """透射率精炼"""
22        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
23        gray = np.float64(gray) / 255
24        refined_t = self.guided_filter(gray, t) # 对透射率进行导向滤波
25        return refined_t
26
```

还有就是进行深度估计的代码

```
1     def recover_depth_map(self, transmission, A):
2         """深度估计"""
3         depth_map = -0.1 * np.log(transmission) * 255
4         depth_map_normalized = cv2.normalize(depth_map, None, 0, 1, cv2.NORM_MINMAX)
5         depth_map_normalized_gray = (depth_map_normalized * 255).astype('uint8')
6         depth_map_heatmap = cv2.applyColorMap(depth_map_normalized_gray, cv2.COLORMAP_HOT)
7         return depth_map_heatmap
8
```

### 1.3 | 实验设置

实验的超参数设置使用了论文中给出的实验建议，如表1.1所示。

实验使用了 REalistic Single Image DEhazing (RESIDE) 数据集。这是一个由合成雾图和真实世界的雾图组成的大规模数据集，并且根据不同的数据源和图像内容，分为五个子集，每个子集有不同的目的（训练或评估）或来源（室内或室外）：ITS (Indoor Training Set)，OTS (Outdoor Training

Set), SOTS (Synthetic Objective Testing Set), RTTS (Real-world Task-Driven Testing Set), HSTS (Hybrid Subjective Testing Set), Unannotated Real-world Hazy Images (不包含在上述子集中)。其中测试使用的是 SOTS 数据集，挑选了 1000 张图片，分别包括了 500 张室内图片和 500 张室外图片。

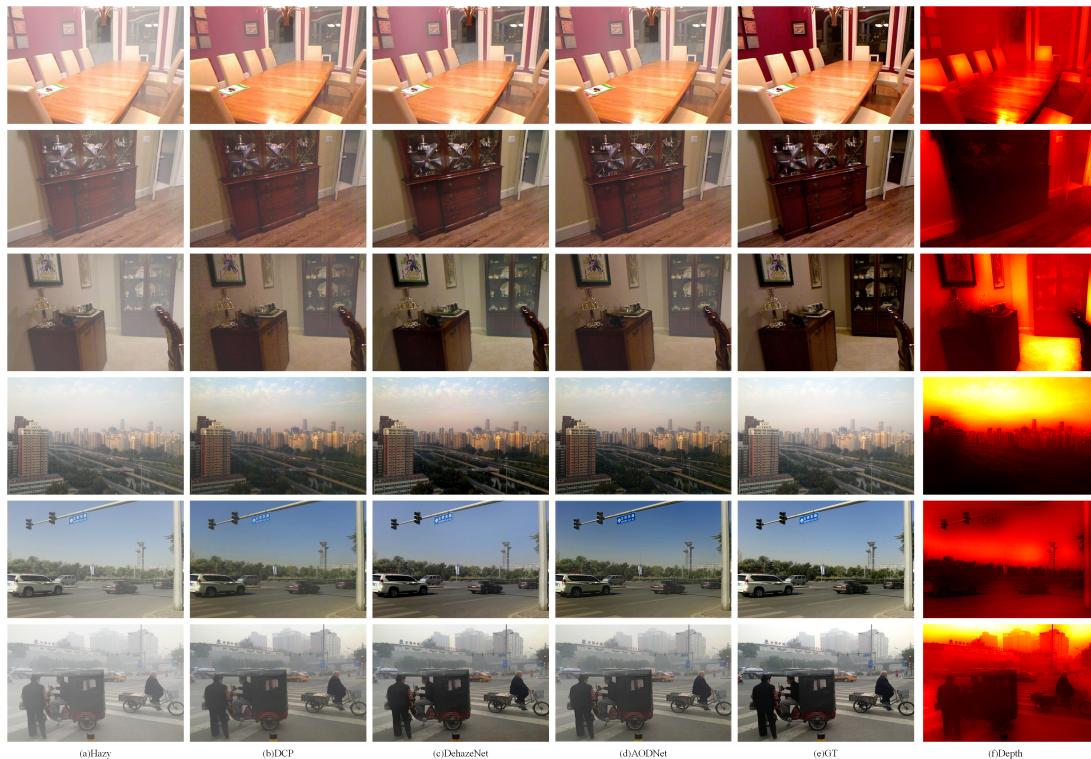
omega	t0	radius	r	eps
0.95	0.1	7	60	1e-4

**表 1.1:** 实验的超参数设置，其中 omega 是透射率估计的权重参数，t0 是最小透射值，radius 是暗通道计算半径，r 是导向滤波器半径，eps 是导向滤波器的 epsilon 值

## 1.4 | 可视化及定量实验结果展示及说明

为了更好地进行实验结果的对比，我们还选择了两种端到端神经网络的去雾方法，分别是 DehazeNet 和 AODNet，其中 DehazeNet 也参考了暗通道先验。可视化实验结果如图1.3所示，我们选择了数据集中六张图片进行可视化结果展示，其中三张室内图片，三张室外图片，可以更好地观察 DCP 的效果。

对于定量实验结果，如图1.2所示。我们采取了三种评价指标，分别是 PSNR,SSIM,Time。其中 PSNR 是峰值信噪比，该指标衡量了最大值信号和背景噪音之间的图像质量参考值，单位为 dB，其值越大，图像失真越少；SSIM 则是结构相似性指数，该指标从亮度、对比度以及结构量化图像的属性，用均值估计亮度，方差估计对比度，协方差估计结构相似程度，其值越大代表图像越相似。这三个指标都是对于所有图片的结果取平均值。



**图 1.3:** DCP, DehazeNet, AOD-Net 的可视化对比实验结果，其中 (a) 列是原始的雾图，(b) 列是该方法 Dark Channel Prior[3] 的去雾结果，(c) 列是 DehazeNet[1]，(d) 列是 AOD-Net[4] 的去雾结果，(e) 列是每一张雾图对应的 Ground Truth，(f) 列是每一张图利用 DCP 方法的深度估计结果

Method	PSNR	SSIM	Time(s)
DCP	18.00	0.8579	<b>0.0641</b>
DehazeNet	<b>24.48</b>	<b>0.9153</b>	0.3245
AOD-Net	21.08	0.8787	0.0685

**表 1.2:** DCP, DehazeNet, AOD-Net 在 SOT 数据集上的定量实验结果展示。其中 PSNR 与 SSIM 都是数值越高代表效果越好，而 Time (单位为秒) 则是越短越好。三种方法中最好的结果已经用粗体标注。

## 1.5 | 实验结果现象的原理性分析

通过定量实验结果我们可以发现，暗通道先验的方法虽然对大多数自然场景图像有较好的效果，并且实现原理简单，复原效果好，使用场景广泛，但是相比起后续兴起的深度学习方法还是有着一定的差距。从 PSNR 指标的角度观看可以发现 AOD-Net 的效果比 DCP 提高了 17%，而 DehazeNet 更是提高了 36%，不过由于深度学习需要的参数量多，训练时间长，DCP 的复原一张图片的平均时间反而是三种方法中最快的。

而通过可视化结果我们可以发现 DCP 方法在某些细节上是存在问题的，为了更好地观察这些问题，我们对于其他图片进行了可视化，如图1.4所示。

- 图像亮度降低。观察1.4.(b) 中第三张和第四张图片，可以发现整张图片的亮度降低了很多。
- 图像出现较多噪点。观察1.3.(b) 中第二张和第三张图片，可以发现整张图片都充斥着噪点。
- 当场景中出现光源时，违反雾图模型，导致了色斑色块效应，当像素点的强度接近大气光值时，去雾的图像出现局部的色斑、色偏效应。观察1.4.(b) 中第一张和第二张图片，可以发现天空的颜色是比较奇怪的。



**图 1.4:** DCP 部分图片可视化实验结果，其中 (a) 列是原始的雾图，(b) 列是该方法 Dark Channel Prior[3] 的去雾结果，(c) 列是每一张雾图对应的 Ground Truth



需要注意的是，这个算法我是用 python 复现的，而如果使用 C++ 复现的话，时间可以缩短一倍，如果使用 GPU 进行加速的话，应该可以达到实时处理。

在实验结果中，我觉得有两点需要注意：

- 这个去雾算法对于低对比度的天空或者水面背景的去雾效果会产生块效应，去雾效果不好，而且这种效应并不能通过调参来避免。
- 暗通道去雾使得图像整体的亮度有所降低，所以在最后可以自适应的提高亮度来减轻这种现象。

对于第二个问题，我们可以使用一些简单的自适应提高亮度方法来解决；但是对于第一个问题中低对比度的天空或者水面会出现失真或块效应的问题没有想到比较合适的解决方法。

## 1.6 | 结论与总结

对于暗通道先验这个方法，作者通过统计无雾图像的特征时发现了在无雾图像中，每一个局部区域都很有可能会有阴影，或者是纯颜色的东西，又或者是黑色的东西。因此，每一个局部区域都很有可能有至少一个颜色通道会有很低的值，也就是 Dark Channel Prior(DCP)。直观来说，DCP 认为每一个局部区域都总有一些很暗的东西。这个规律很简单，但在去雾问题上却是本质的基本规律。由于雾总是灰白色的，因此一旦图像受到雾的影响，那么这些本来应该很暗的东西就会变得灰白。不仅如此，根据计算机视觉中常用的雾图形成公式，还能根据这些东西的灰白程度来判断雾的浓度。总而言之，在 14 年前那个深度学习还没有开始盛行的年代，通过一个简单的方法，或者说发现了一个基本的自然规律并且应用在实际的问题中，可以使当时的实验结果有着重大提升是一件让人感觉非常不可思议的事情。

# 2 | 高层视觉及传统/深度学习方法: 语义分割 PSPNET/DeeplabV3+

## 2.1 | 简介

图像分割是一个将图像划分为不重叠的区域或片段的过程，每个片段代表一个不同的对象或感兴趣的区域。图像分割的目的是识别图像中不同物体或区域之间的边界，并将图像中的每个像素分配到所识别的片段中。当在图像中的物体具有类似的颜色、形状或纹理的情况下式，这个任务将会非常具有挑战性。

图像分割在各个领域都有广泛的应用，包括医学成像、机器人、计算机图形、视频监控等等。例如，在医学成像中，图像分割可用于识别和定位图像中的特定结构，如肿瘤、血管或器官。在机器人领域，图像分割可用于实时识别和跟踪物体，这对自主导航和操纵至关重要。

一些传统的图像分割技术包括阈值处理、边缘检测和区域增长。这些方法通常依靠手工制作的特征和启发式方法来识别图像中不同区域之间的边界。而近年来，基于深度学习的方法在图像分割中越来越受欢迎，因为它们有能力学习分层特征并捕获图像的高级语义。卷积神经网络 (CNN) 通常用于图像分割，该网络在一组有注释的图像上进行训练，学习识别图像中不同区域之间的边界。用于图像分割的一个流行的 CNN 架构是 U-Net，它被设计为同时执行下采样和上采样操作，以捕捉图像的全局和局部特征。



而相比起图像分割，语义分割是在像素级别上的分类，属于同一类的像素都要被归为一类，因此语义分割是从像素级别来理解图像的。

综上所述，我们认为语义分割是一个非常具有挑战性和有趣的任务，所以选择了在 CityScspe 对比两种经典模型 PSPNet 和 DeeplabV3+ 来完成实验。

## 2.2 | 关键代码

使用的是 Github 上非原创性代码，其中 PspNet 链接如下：SegmenTron，DeeplabV3+ 链接如下：DeepLabV3Plus-Pytorch，我们的代码和数据集结果链接如下：CV-Final。

## 2.3 | 实验设置

为了尽量进行定量实验，我们这里汇总两种模型的部分重要训练参数细节设定，如下图。其中，在数据集 cityscape 上，我们限于 GPU 资源和时间的限制，并没有采用全部的完整数据集，而是通过随机采样选择出  $\frac{2}{3}$  的数据进行训练。此外，由于 DeepLabV3+ 提供了两种 backbone 架构，所以我们也进行保持两种 backbone 结构的参数一致性，从而讨论 backbone 对效果的影响。

Model	DeeplabV3+		PSPNET
Backbone	resnet101	mobilenet_v2	resnet101
Dataset	cityscape	cityscape	cityscape
Dataset Mean	[0.485, 0.456, 0.406]	[0.485, 0.456, 0.406]	[0.485, 0.456, 0.406]
Dataset Std	[0.229, 0.224, 0.225]	[0.229, 0.224, 0.225]	[0.229, 0.224, 0.225]
Train	Batch size	8	4
	Learning rate	0.02	0.01
	Epochs	400	400
	Crop size	769	713
	auxiliary loss	False	True (AUX_WEIGHT = 0.4)
Test	Batch size	4	4
	Crop size	(1025, 2049)	(1025, 2049)
	Epochs	400	400

## 2.4 | PspNet 实验结果现象的原理性分析

PSPNet (Pyramid Scene Parsing Network) 是一种用于图像语义分割的深度学习模型，它的提出主要采用了 4 种新的思路：

1. 金字塔池化 (Pyramid Pooling)：PSPNet 中引入了一种新的金字塔池化模块，可以在不同尺度上提取图像特征。这种模块可以有效地捕捉不同尺度下的上下文信息，并将这些信息整合到一个固定长度的向量中，这样可以使模型更好地识别图像中的物体。
2. 多尺度特征融合：PSPNet 在对图像特征进行处理时，引入了多个并行的卷积层，每个卷积层在不同尺度下处理图像特征。这样可以提高模型对物体的检测和分割能力。
3. 空洞卷积 (Dilated Convolution)：PSPNet 在模型中采用了空洞卷积，这种卷积可以在不增加参数和计算量的情况下，增加感受野的大小，提高模型对物体的识别和分割能力。

4. 融合多个卷积网络：PSPNet 将多个卷积网络融合在一起，包括 ResNet、VGG 和 GoogleNet 等。这样可以充分利用不同卷积网络的优点，提高模型的性能和准确度。

#### 2.4.1 | 模型结构

PSPNet 的模型结构可以分为两个主要部分：特征提取模块和金字塔池化模块。

1. 特征提取模块：PSPNet 使用了预训练的卷积神经网络（例如 ResNet）来提取图像的特征。这些特征在不同的层级上具有不同的语义信息，因此 PSPNet 通过采用多个尺度的特征图来捕获不同层级的信息。这个过程被称为金字塔池化（Pyramid Pooling）。
2. 金字塔池化模块：金字塔池化模块用于捕获不同层级特征的上下文信息。它由四个并行的池化操作组成，每个池化操作对输入特征图进行不同的空间池化操作，然后将结果上采样到相同的尺寸。这四个操作分别是全局池化、 $1/2$ 、 $1/3$  和  $1/6$  的平均池化。全局池化用于捕获全局上下文信息，而不同比例的平均池化则用于捕获局部的细节信息。

在金字塔池化模块中，每个池化操作的输出都经过一个卷积层和一个反卷积层，以减少特征图的通道数，并将其上采样到与输入特征图相同的尺寸。然后，这些特征图通过连接操作合并在一起，形成了最终的上下文感知特征图。

3. 解码器：在特征提取和金字塔池化后，PSPNet 使用一个解码器网络来将上下文感知特征图转换为与输入图像相同分辨率的语义分割结果。解码器通常由卷积层、上采样层和分类器组成。上采样层用于将特征图的尺寸上采样到与输入图像相同的尺寸，以便进行像素级的语义分割。分类器使用卷积层将上采样后的特征图映射到最终的语义类别数量。

通过特征提取模块和金字塔池化模块，PSPNet 能够同时捕获全局上下文信息和局部细节信息，从而提高语义分割的准确性。这个结构使得 PSPNet 成为一种在各种场景中表现出色的语义分割模型。

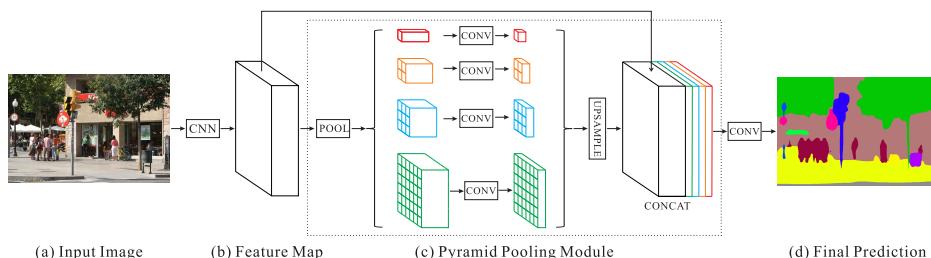


图 2.1: PSPNet 的特征金字塔结构

#### 2.4.2 | 训练细节

我们在 A5000 设备上训练 PspNet，得到如图2.2所示 loss 图：

### 2.5 | DeeplabV3+ 实验结果现象的原理性分析

DeepLab v3+ 是一种用于语义分割任务的深度卷积神经网络模型，是 DeepLab 系列模型的改进版本。它在 DeepLab v3 的基础上引入了编码器-解码器结构和多尺度信息融合，以进一步提高语义分割的性能。其创新点主要有：

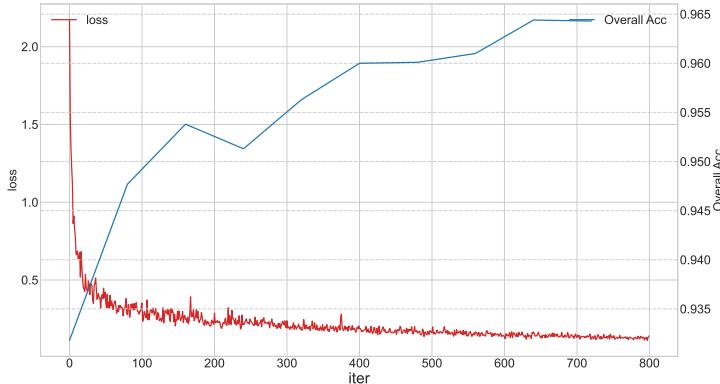


图 2.2: pspnet loss

1. 空洞卷积 (Dilated Convolution): DeepLab v3+ 使用了空洞卷积来扩大感受野。通过在卷积层中引入跳跃的空洞率 (dilation rate)，模型可以捕获更广阔的上下文信息，从而更好地理解图像中的语义信息。
2. Atrous Spatial Pyramid Pooling (ASPP): DeepLab v3+ 引入了 ASPP 模块来捕获不同尺度下的上下文信息。ASPP 模块由多个并行的空洞卷积层组成，每个层的空洞率不同，以捕获不同尺度的特征。这样可以在不引入额外的上采样过程的情况下，提高网络对多尺度物体的分割能力。
3. 编码器-解码器结构: DeepLab v3+ 引入了解码器网络来将高级特征图进行上采样，以生成与输入图像相同分辨率的分割结果。解码器包含了一个上采样层和一个特征融合模块。通过解码器的引入，DeepLab v3+ 可以更好地恢复分割结果的空间细节和边缘锐利度。

这些创新点使得 DeepLab v3+ 在语义分割任务中取得了出色的性能，能够准确地将图像中的每个像素分类为不同的语义类别，从而在许多计算机视觉应用中发挥重要作用，例如图像分割、自动驾驶和医学图像分析等。

### 2.5.1 | 模型结构

DeepLab V3+ 结合了空洞卷积 (Dilated Convolution)、多尺度信息融合和编码器-解码器结构，以提高语义分割的性能。其详细的模型结构如图2.3所示：

1. 特征提取 (编码器): DeepLab V3+ 使用一个预训练的卷积神经网络 (通常是 ResNet) 作为特征提取网络。该网络负责从输入图像中提取高级语义特征。在 ResNet 的基础上，使用空洞卷积 (Dilated Convolution) 来扩大感受野，以捕获更广阔的上下文信息。空洞卷积通过在卷积层中引入跳跃的空洞率 (dilation rate)，以增加卷积核在输入特征图上的感受野。
2. 多尺度信息融合: DeepLab V3+ 引入了一种称为 Atrous Spatial Pyramid Pooling (ASPP) 的模块，用于捕获不同尺度下的上下文信息。ASPP 模块由多个并行的空洞卷积层组成，每个层的空洞率不同，以捕获不同尺度的特征。这样可以在不引入额外的上采样过程的情况下，提高网络对多尺度物体的分割能力。
3. 解码器: DeepLab V3+ 使用解码器网络来将高级特征图进行上采样，以生成与输入图像相同分辨率的分割结果。解码器包含了一个上采样层和一个特征融合模块。

- 上采样层：DeepLab V3+ 使用双线性插值或转置卷积（Transposed Convolution）来将特征图的尺寸上采样到与输入图像相同的尺寸。
- 特征融合模块：为了提高分割结果的细节性和边缘锐利度，DeepLab V3+ 在上采样的特征图和原始输入图像之间进行了特征融合。具体来说，它使用了一个具有 1x1 卷积的残差连接（Residual Connection）来融合不同分辨率的特征。

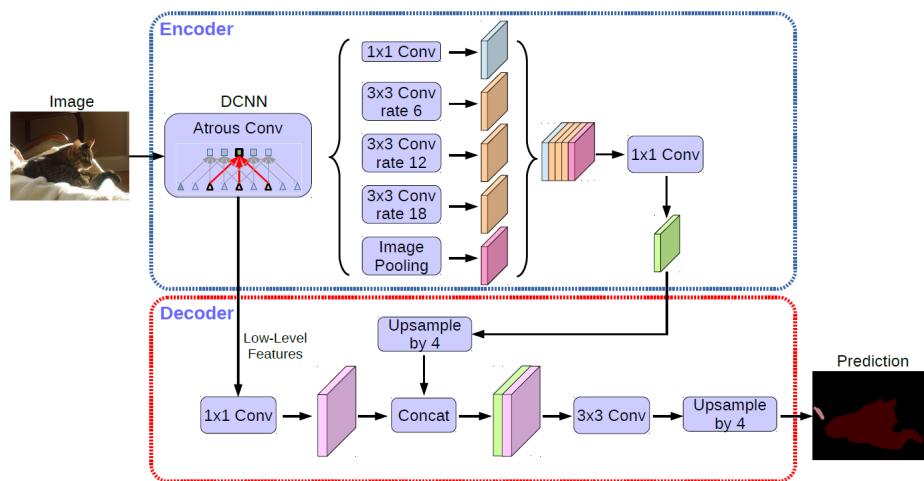
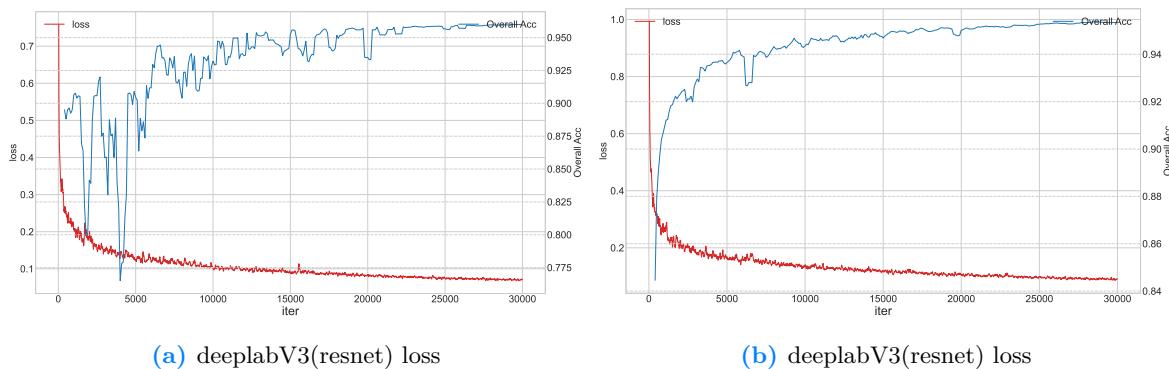


图 2.3: DeeplabV3+ 模型结构

## 2.5.2 | 训练细节

同样在 GPU A5000 设备上分别训练，得到如图2.4a和2.4b所示的 loss 细节：

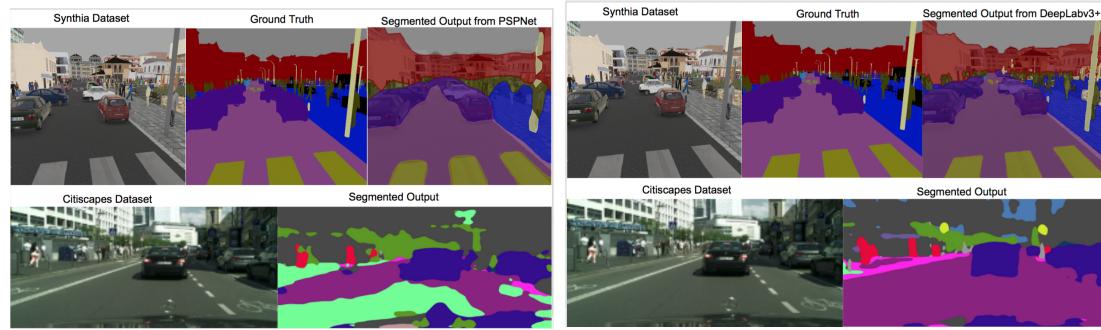


## 2.6 | 可视化及定量实验结果展示及说明

对于同一张输入图片，我们得到了 PspNet 和 DeepLabV3+(resnet101) 的效果图如图2.5a和2.5b所示：

从图中可以看出，在几乎相同的训练细节的设置下，DeepLab v3+ 的效果要明显好于 PspNet，尤其是在道路的分割上，DeepLab v3+ 能比较明显地分割出完整的道路，但是 PspNet 的道路则杂乱不清。

为了更加定量比较，我们汇总了三种网络的指标对比表格，如表2.1所示：



(a) PspNet 的处理结果

(b) deeplabV3+(resnet101) 的处理结果

	Metrics	PSPNET	DeeplabV3+(resnet101)	DeeplabV3+(mobilenet_v2)
class-level	Mean IOU	35.2	49.3	48.2
	Road	90.8	94.6	90.6
	Sidewalk	54.3	65	55.8
	Building	71.6	81.2	72.1
	Wall	14.5	28	27.1
	Fence	13.4	28.8	25.1
	Pole	22	31.4	25.9
	Trafficlight	10.9	26.5	15.9
	Trafficsign	15.7	36.8	30.7
	Vegetation	78.1	82.3	80.3
	Terrain	35.5	46	41.3
	Sky	83.6	86.5	82.6
	Person	33.5	51.6	45.6
	Rider	9	30.9	25.7
	Car	72.6	82.9	75.6
	Truck	10.8	32.8	29.8
	Bus	12.3	38.2	33.1
	Train	0.8	27.5	25.4
	Motorcycle	4.6	15.3	14.2
	Bicycle	35.1	49.6	45.9
category-level	Mean IOU	65.7	74.1	71.2
	Flat	95	96.5	95.2
	Nature	78	82.4	78.6
	Object	22.9	36.2	33.8
	Sky	83.6	86.5	82.5
	Construction	71.6	81.2	80.6
	Human	37.4	54.6	52.5
	Vechicle	71.1	81.3	75.3

表 2.1: 三种网络测试结果对比



## 2.7 | 结论与总结

从 DeeplabV3+ 和 PspNet 的对比中可以发现，DeeplabV3+ 的效果要优于 PspNet，空洞卷积和编码器-解码器的结构的引入有着非常好的效果。同时，语义分割的可视化效果图是让我们感到惊讶的，因为语义是一种图像的高层信息，而在可视化结果中，Deeplabv3+ 可以很好地将汽车，楼房，天空，道路等信息用不同颜色进行分割。相信随着技术的发展，语义分割在工业界的应用会让我们感到惊艳。

## 3 | OpenAI CLIP: Learning Transferable Visual Models From Natural Language Supervision (ICML-2021)

### 3.1 | 简介

在之前的计算机视觉领域中存在几个问题，包括数据集制作成本高（大多是人工标注）；大多数任务都是基于预先定义的标签来训练，虽然说如今监督学习在很多任务上都达到了让人惊叹的结果，但其限制是：往往需要足够多的样本才能训练出足够好的模型，并且限制了模型的泛化能力和实用性，比如说利用猫狗训练出来的分类器，就只能对猫狗进行分类，其他的物种它都无法识别。这样的模型显然并不符合我们对人工智能的终极想象，我们希望机器能具有通过推理，识别新类别的能力。针对这个问题产生了一种新的研究方向——Zero-shot Learning，也就是希望我们的模型能够对其从未见过的类别进行分类，让机器具有推理能力，实现真正的智能。其中 Zero-shot 是指对于要分类的类别对象，一次也不学习。假设我们的模型已经能够识别马，老虎和熊猫了，现在需要该模型也识别斑马，那么我们需要告诉模型，怎样的对象才是斑马，但是并不能直接让模型看见斑马。所以模型需要知道的信息是马的样本、老虎的样本、熊猫的样本和样本的标签，以及关于前三种动物和斑马的描述。

而 CLIP[6] 是 OpenAI2021 年提出的结合图像和文本的多模态模型，是一个用文本作为监督信号来训练可迁移的视觉模型，实现了 zero-shot 分类，并且后续有许多新的模型能够根据基础的 CLIP 进行变种，应用在不同的任务场景中。

具体来说，Contrastive Language-Image Pre-training(CLIP)，是一种基于对比文本-图像对的预训练方法或者模型。CLIP 是一种基于对比学习的多模态模型，与 CV 中的一些对比学习方法不同的是，CLIP 的训练数据是文本-图像对：一张图像和它对应的文本描述，并且希望通过对比学习，模型能够学习到文本-图像对的匹配关系。

### 3.2 | 关键代码

由于 CLIP 过大，无法训练，因此我们直接调用 OpenAI 官方提供的训练好的权重进行下游任务。在 3.5 中给出了部分代码，具体的代码可访问如下链接：CV-Final。

### 3.3 | 实验设置

为了训练 CLIP，OpenAI 从互联网收集了共 4 个亿的文本-图像对。论文中 Text Encoder 固定选择一个包含 63M 参数的 text transformer 模型，而 Image Encoder 采用了两种的不同的架构，一是常用的 CNN 架构 ResNet，二是基于 transformer 的 ViT，其中 ResNet 包含 5 个不同大小的模型：ResNet50，ResNet101，RN50x4，RN50x16 和 RNx64（后面三个模型是按照 EfficientNet 缩放规则对 ResNet 分别增大 4x，16x 和 64x 得到），而 ViT 选择 3 个不同大小的模型：ViT-B/32，



ViT-B/16 和 ViT-L/14。所有的模型都训练 32 个 epochs，采用 AdamW 优化器，而且训练过程采用了一个较大的 batch size: 32768。对于 ViT-L/14，还在 336 的分辨率下额外 finetune 了一个 epoch 来增强性能，论文发现这个模型效果最好，记为 ViT-L/14@336，论文中进行对比实验的 CLIP 模型也采用这个。

因此，我们进行的分类任务也采用了 VIT-L/14@336 模型，但是对于其中两个数据集，我们的算力不够，只能选择较小的模型 ViT-B/32，模型信息如表3.1所示。

Model	模型参数量	输入图片尺寸	文本长度	词表大小
ViT-L/14@336px	427944193	336	77	49408
ViT-B/32	151277313	224	77	49408

表 3.1: 模型信息

实验使用了多种不同的数据集，有传统的图像分类数据集 CIFAR10, CIFAR100，运动分类数据集 archive, ImageNet 数据集的非官方子集 miniImagenet 和自己的 NBA 数据集，数据集具体情况如表3.2所示。

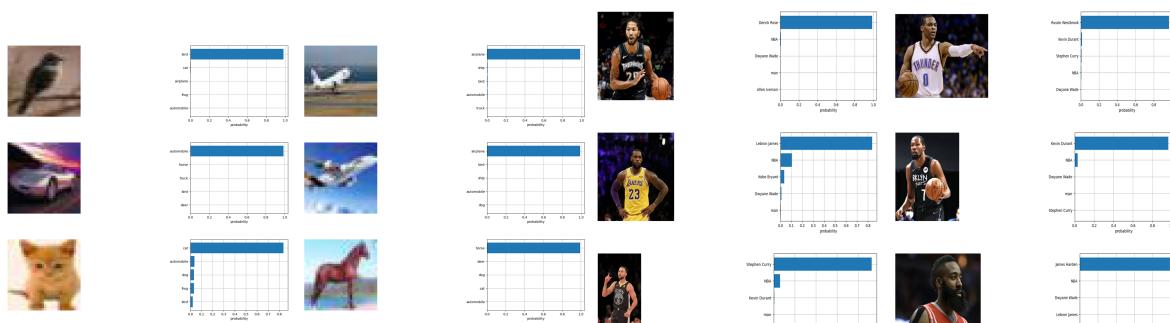
Dataset	CIFAR10	CIFAR100	archive	miniImagenet	NBA
Classes	10	100	101	100	12
Num	10000	10000	505	500	12

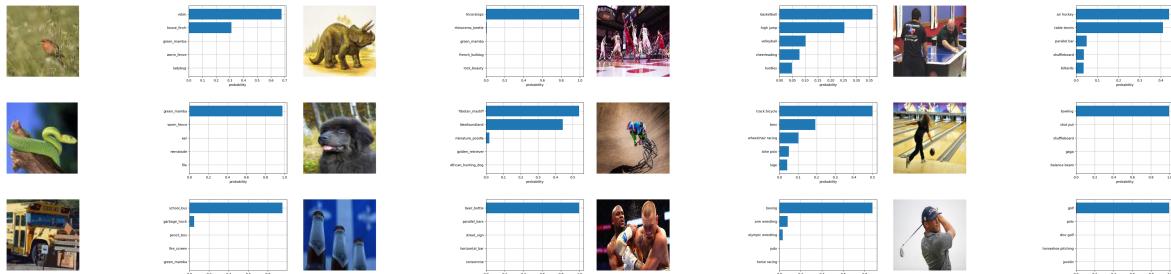
表 3.2: 数据集具体情况

### 3.4 | 可视化及定量实验结果展示及说明

对于上述的五个数据集，我们在 CIFAR10 和 CIFAR100 的测试集上做了定量实验，如表3.3所示，在除了 CIFAR100 以外的数据集上做了可视化结果，如图3.1a、3.1b、3.2a和3.2b所示。

从定量实验结果可以发现 CLIP 虽然与论文中展示的结果有轻微的差距，但是依旧是非常惊人的。对于可视化结果，我们在每个数据集中分别给出了六张图片进行展示，并且在每张图片旁边绘制了 CLIP 对于该图片前五个标签的预测概率，可以发现虽然有些标签是存在干扰性的，但是结果是基本正确的。





(a) miniImagenet 可视化结果

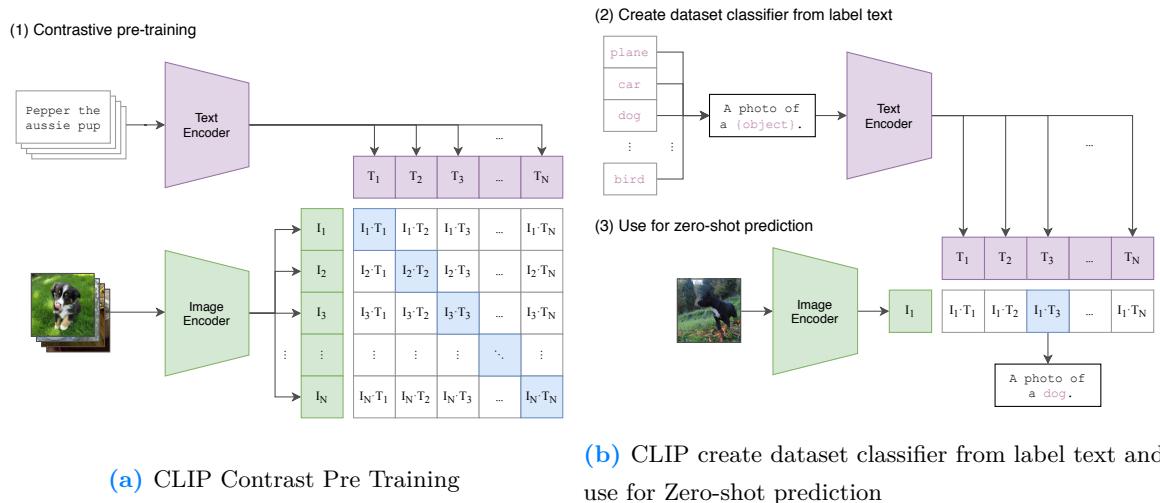
(b) archive 可视化结果

Model	CIFAR10	CIFAR100
ViT-L/14@336px	94.97%	72.00%
ViT-B/32	88.78%	61.71%

表 3.3: CLIO 在 CIFAR100 和 CIFAR10 测试集的定量实验结果

### 3.5 | 实验结果现象的原理性分析

如图3.3a所示，CLIP 包括两个模型：Text Encoder 和 Image Encoder，其中 Text Encoder 用来提取文本的特征，可以采用 NLP 中常用的 text transformer 模型；而 Image Encoder 用来提取图像的特征，可以采用常用 CNN 模型或者 vision transformer。



模型对提取的文本特征和图像特征进行对比学习。对于  $N$  个包含个文本-图像对的训练 batch，将  $N$  个文本特征和一个图像特征两两组合，CLIP 模型会预测出  $N^2$  个可能的文本-图像对的相似度，这里的相似度直接计算文本特征和图像特征的余弦相似性 (cosine similarity)，即图3.3a所示的矩阵。这里共有  $N$  个正样本，即真正属于一对的文本和图像 (矩阵中的对角线元素)，而剩余的  $N^2 - N$  个文本-图像对为负样本，那么 CLIP 的训练目标就是最大  $N$  个正样本的相似度，同时最小化  $N^2 - N$  个负样本的相似度。

可以看到训练后的 CLIP 其实是两个模型，除了视觉模型外还有一个文本模型，那么接下来就是对预训练好的视觉模型进行迁移。与我们正常的计算机视觉工作中常用的先预训练然后微调不同，CLIP 可以直接实现 zero-shot 的图像分类，即不需要任何训练数据，就能在某个具体下游任务上实现分类，这也是该工作的亮点和强大之处。用 CLIP 实现 zero-shot 分类很简单，只需要简单的两步：



- 如图3.3b所示，根据任务的分类标签构建每个类别的描述文本：A photo of label，然后将这些文本送入 Text Encoder 得到对应的文本特征，如果类别数目为 N，那么将得到 N 个文本特征；
- 将要预测的图像送入 Image Encoder 得到图像特征，然后与 N 个文本特征计算缩放的余弦相似度，然后选择相似度最大的文本对应的类别作为图像分类预测结果，进一步地，可以将这些相似度看成 logits，送入 softmax 后可以得到每个类别的预测概率。

可以看到，作者利用 CLIP 的多模态特性为具体的任务构建了动态的分类器，其中 Text Encoder 提取的文本特征可以看成分类器的 weights，而 Image Encoder 提取的图像特征是分类器的输入。

接下来给出了一个基于 CLIP 的实例：

在这个实例中，我们选取了 skimage 中的六张图片，首先我们创建文本描述，输入图像，然后将图片和文字 encode 生成图片特征和文本特征：

```
1     descriptions = {  
2         "astronaut": "A color image of astronaut Eileen Collins.",  
3         "camera": "A famous black and white photograph of a man with a camera.",  
4         "chelsea": "A color image of a tabby cat.",  
5         "coffee": "A color photograph of a cup of coffee.",  
6         "motorcycle_left": "A left view of a red motorcycle standing in a garage.",  
7         "rocket": "A color image of a rocket launch."  
8     }  
9  
10    image_input = torch.tensor(np.stack(images)).cuda()  
11  
12    text_tokens = clip.tokenize(['This is ' + desc for desc in texts]).cuda() # 对文本描述进  
行 tokenization  
13  
14    with torch.no_grad(): # 使用模型对图像和文本特征进行编码  
15        image_features = model.encode_image(image_input).float()  
16        text_features = model.encode_text(text_tokens).float()  
17
```

然后对于图像和文本特征进行归一化，并计算余弦相似度，结果如图3.4所示，可以看到对于要预测的 6 个图像，按照最大相似度，其不同图像均能匹配到正确的文本标签：

```
1     # 对图像和文本特征进行 L2 归一化  
2     image_features /= image_features.norm(dim = -1, keepdim=True)  
3     text_features /= text_features.norm(dim = -1, keepdim=True)  
4  
5     similarity = text_features.cpu().numpy()@image_features.cpu().numpy().T # 计算文本和图像  
特征之间的余弦相似度  
6
```

进一步地，我们可以对得到的余弦相似度计算 softmax，得到每个预测类别的概率值，如图3.5所示，可以看到对于 6 个图像，CLIP 模型均能够以绝对的置信度给出正确的分类结果：

```
1     # 计算文本特征和相似度得分  
2     with torch.no_grad():  
3         text_features = model.encode_text(text_tokens).float()  
4         text_features /= text_features.norm(dim = -1, keepdim = True)  
5  
6         text_probs = (100.0 * image_features @ text_features.T).softmax(dim = -1)  
7         top_probs, top_labels = text_probs.cpu().topk(5, dim = -1) # 获取概率值和对应的前 5 个标  
签  
8
```

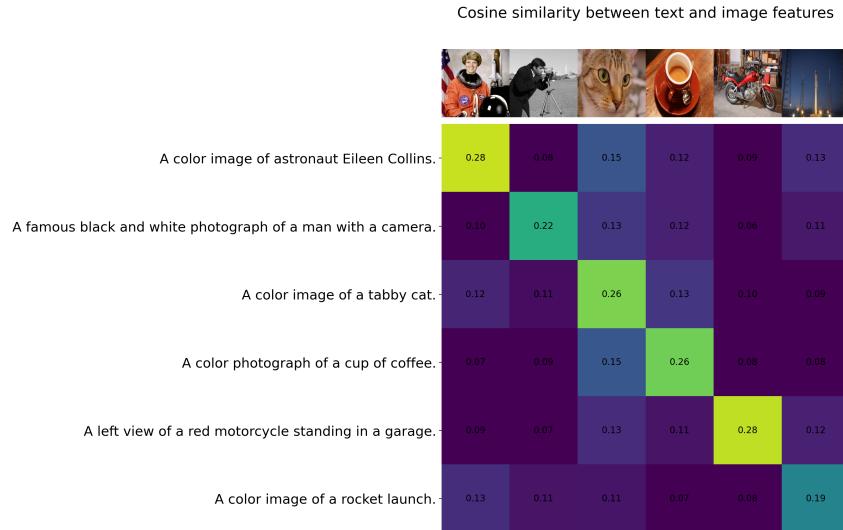


图 3.4: 文本和图像的余弦相似度



图 3.5: 预测类别概率值

### 3.6 | 结论与总结

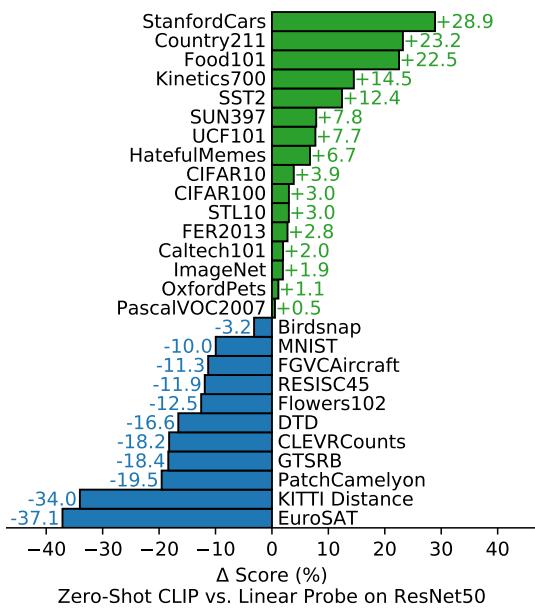
通过对选择特定领域的数据集和特定图片上进行测试，可以看出 CLIP 取得了不错的效果。此外，作者对比了 zero-shot CLIP 和在 ImageNet 数据上预先训练的 ResNet50 linear probing（加上线性分类层进行 finetune）在 27 个数据集上的表现，如图3.5所示。结果显示在 16 个数据集上 CLIP 的表现超过了 ResNet50。然而，在一些特别的、复杂的或抽象的数据集上，例如卫星图像分类、淋巴结转移检测和合成场景中计数等任务，CLIP 的效果不如全监督的 ResNet50，这表明 CLIP 仍有改进的空间。值得注意的是，CLIP 表现较差的数据集竟然包括 MNIST，分类准确度只有 88%。这是不可思议的，因为 MNIST 任务太简单了。通过对 CLIP 训练数据进行分析，作者发现 4 亿的训练数据中基本上没有和 MNIST 相似的数据，因此在这个任务上，CLIP 的表现十分不理想。这说明 CLIP 仍然无法解决域外泛化这个深度学习难题。

除此以外，论文还发现 CLIP 在自然分布漂移上表现更鲁棒，如图3.6a所示。例如，在 ImageNet 验证集上，CLIP 和在 ImageNet 上有监督训练的 ResNet101 都能达到 76.2% 的准确度。在 ImageNetV2

数据集上，CLIP 的表现超过了 ResNet101。在另外的 4 个分布漂移的数据集上，ResNet101 的性能下降得比较厉害，但是 CLIP 能依然保持较大的准确度。例如，在 ImageNet-A 数据集上，ResNet101 的准确度只有 2.7%，而 CLIP 能达到 77.1%。

总之，相比以往的迁移学习方法或者基于 MoCo 和 SimCLR 的对比学习方法以及基于 MAE 和 BeiT 的图像掩码方法，这些方法都无法实现 zero-shot，这些模型在新的数据集上需要定义新的分类器来重新训练。而 CLIP 通过文本-图像对数据集训练，最后实现了 zero-shot。然而，CLIP 仍存在一定的局限性：

- CLIP 的 zero-shot 性能虽然和有监督的 ResNet50 相当，但是还不是 SOTA，作者估计要达到 SOTA 的效果，CLIP 还需要增加 1000x 的计算量，这是难以想象的；
- CLIP 的 zero-shot 在某些数据集上表现较差，如细粒度分类，抽象任务等；
- CLIP 在自然分布漂移上表现鲁棒，但是依然存在域外泛化问题，即如果测试数据集的分布和训练集相差较大，CLIP 会表现较差；
- CLIP 并没有解决深度学习的数据效率低下难题，训练 CLIP 需要大量的数据。



(a) CLIP 和 RESNET50 在 27 个数据集上的对比

Dataset Examples	ImageNet		Δ Score
	ResNet101	CLIP	
ImageNet	76.2	76.2	0%
ImageNetV2	64.3	70.1	+5.8%
ImageNet-R	37.7	88.9	+51.2%
ObjectNet	32.6	72.3	+39.7%
ImageNet Sketch	25.2	60.2	+35.0%
ImageNet-A	2.7	77.1	+74.4%

(b) Zero-shot CLIP is much more robust to distribution shift than standard ImageNet models

虽然论文中只对用 CLIP 进行 zero-shot 分类做了实验，但实际上 CLIP 的应用价值远不止此。CLIP 之后出现了很多基于 CLIP 的其他下游任务的应用研究，这里我们列出一些应用场景：

- zero-shot 检测：CLIP 可以应用在目标检测任务上，实现 zero-shot 检测，即检测训练数据集没有包含的类别，比如谷歌提出的 ViLD[2] 基于 CLIP 实现了开放词汇的物体检测，其基本思路和 zero-shot 分类相似，只不过这篇论文中是用文本特征和 ROI 特征来计算相似度。
- Meta AI 的最新工作 Detic[12] 可以检测 2000 个类，背后也用到了 CLIP。
- 视频理解：CLIP 是基于文本-图像对来做的，但是它可以扩展到文本-视频，比如 VideoCLIP[11] 就是将 CLIP 应用在视频领域来实现一些 zero-shot 视频理解任务。



- 图像编辑：CLIP 可以用在指导图像编辑任务上，HairCLIP[10] 这篇工作用 CLIP 来定制化修改发型。
- 图像生成：CLIP 还可以应用在图像生成上，比如 StyleCLIP[5] 这篇工作用 CLIP 实现了文本引导的 StyleGAN。
- CLIP-GEN[8] 这篇工作基于 CLIP 来训练文本生成图像模型，训练无需直接采用任何文本数据。
- 自监督学习：最近华为的工作 MVP[9] 更是采用 CLIP 来进行视觉自监督训练。
- VL 任务：CLIP 本身就是多模态模型，所以它也可以用在用图像-文本多模态任务，如图像描述和视觉问答，这篇论文 CLIP Models are Few-shot Learners: Empirical Studies on VQA and Visual Entailment[7] 评估了 CLIP 在 vqa 任务中的 zero-shot 性能，并展示了 CLIP 在视觉问答任务中的 zero-shot 跨模态迁移能力。



## 4 | 参考文献

- [1] Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. Dehazenet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, page 5187–5198, Aug 2016.
- [2] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation, Apr 2021.
- [3] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 2341–2353, Sep 2010.
- [4] Boyi Li, Xiulan Peng, Zhangyang Wang, Xu Jizheng, and Dan Feng. An all-in-one network for dehazing and beyond, Jul 2017.
- [5] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery, Mar 2021.
- [6] Alec Radford, JongMin Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, JackA. Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, Feb 2021.
- [7] Haoyu Song, Li Dong, Wei-Nan Zhang, Ting Liu, and Furu Wei. Clip models are few-shot learners: Empirical studies on vqa and visual entailment.
- [8] Wang Wang, Zihao, Liu Liu, Wei, He He, Qian, Wu Wu, Xinglong, Yi Yi, and Zili. Clip-gen: Language-free training of a text-to-image generator with clip, Mar 2022.
- [9] Longhui Wei, Lingxi Xie, Wengang Zhou, Houqiang Li, and Qi Tian. Mvp: Multimodality-guided visual pre-training.
- [10] Tianyi Wei, Dongdong Chen, Wenbo Zhou, Jing Liao, Zhentao Tan, Lu Yuan, Weiming Zhang, and Nenghai Yu. Hairclip: Design your hair by text and reference image.
- [11] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding, Sep 2021.
- [12] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Phillip Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision.