媒体与认知
图像实验开题报告

刘圣禹　　马啸阳　　袁健皓

2020 年 4 月 9 日

1　实验任务

本实验主要目标为对交通标志进行分类与检测，分为如下四个任务。

1. 基于传统机器学习方法进行交通标志分类；
2. 基于深度学习方法，实现并训练一个卷积神经网络；
3. 小样本分类；
4. 在给定的交通标志检测数据集上，实现一个检测模型。

2　现有工作

本节主要在课程中已习得的深度学习基础上，调研了若干现有表现较好的分类及检测方法，归纳总结各方法的思想及结构。

2.1　分类

2.1.1　HOG[1]

HOG（Histogram of Oriented Gradient，方向梯度直方图）是一种经典的图像特征提取方法，最初被用于行人识别领域。这种方法有其自身的优势：能够获取原始图片上很明显的边缘和梯度结构，且它在局部的特征表示具有几何不变性和光学形变的不变性。

HOG 的基本思想在于图像的局部特征可由梯度及边缘方向的分布刻画。利用梯度描绘图像边缘信息是图像处理中的经典方法，使用 Sobel 算子或其它不同算子可以提取不同的边缘特

征，而 HOG 则统计了梯度方向的分布，更好刻画了图像的局部特征。算法流程如图 1所示，先对输入图片进行归一化，再计算图像的梯度大小与方向，然后将图片分为小的区域。对于每个区域中的像素，求出梯度方向的 1 维直方图，再对其进行权重投影。对重叠块中的小区域进行对比度归一化，最后在将所有块中的直方图向量组合成一个 HOG 特征向量，得到的特征向量便可以通过线性 SVM 实现分类等任务。

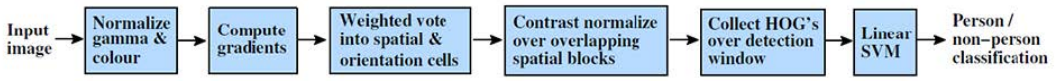


图 1: HOG 算法流程

总而言之，HOG 统计分割区域上的梯度方向直方图，用以刻画图像的局部特征，取得一维特征后，再使用 SVM 等经典分类器实现分类任务，这一经典算法也能在分类任务中获得较高的精度。

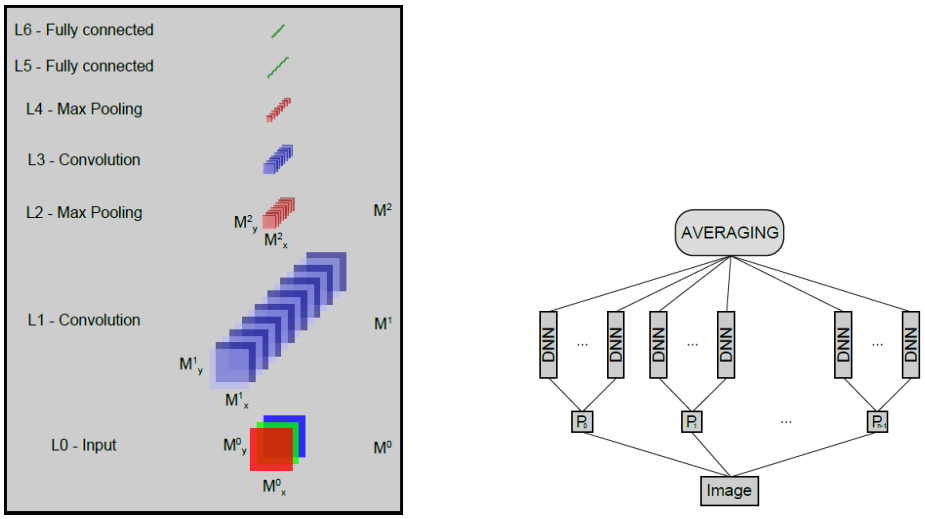
2.1.2 MCDNN[2]

MCDNN（Multi-Column Deep Neural Network）在德国交通标志检测识别数据集（German traffic sign recognition benchmark）上获得了 99.46% 的准确率，达到了人类识别的水准，较为符合本实验的目标。

MCDNN 中，基础网络结构如图 2(a) 所示，含两轮交替的卷积与最大池化层，以及两个全连接层，是典型的 DNN 网络结构。而 MCDNN 的主要贡献在于引入了多道 DNN 机制（如图 2(b)）。

多道 DNN 的思想在于，将原始图片经过不同的输入增强与预处理，作为多个 DNN 通道的输入，每个 DNN 通道被随机初始化（服从不同正态分布）后进行训练，训练过程中也不断对输入图片进行随机的扭曲。最后对所有 DNN 通道的输出结果做平均，得到输出。并且实验表明，进行算术平均比不同通道的加强平均具有更好的泛化能力。

MCDNN 的数据增强部分是对整个数据集的操作，非随机，包括 Image Adjustment（图像调整）、Histogram Equalization（直方图均衡）、Adaptive Histogram Equalization（自适应直方图均衡）、Contrast Normalization（对比度归一化）等基本的图像预处理方式。经过不同的数据增强会作为不同的通道输入。图像扭曲（在训练过程中也每次迭代执行）包括小幅度平移、缩放、旋转。不同的图片经过切割、数据增强与扭曲，最终通过双线性插值得到最终固定大小的图片数据。



(a) DNN 网络结构 (b) 多道机制

图 2: MCDNN 结构

文中训练交通标志所使用的通道结构是训练了 25 个 DNN，每 5 个 DNN 对应一种图像预处理（包括原始图像和 4 种归一化方法），即每种预处理连接 5 个通道，最终执行算术平均，最终得到了 99.46% 的准确率，同时具有较好的泛化能力，是非常优秀的深度学习方法。

2.1.3 VGG[3]

VGG 表明增大网络深度能有效提升网络的性能。其主要思想为在 AlexNet 中用较小卷积核的叠加来取代较大的卷积核，增加网络深度。VGG 基础结构为卷积与最大池化的交替，如图 3所示。图像预处理中将每个像素减去训练集上的 RGB 均值。

实验表明，AlexNet 中采用的 LRN 层（local response normalization，局部响应归一化）没有带来性能的提高，因此 VGG 中去除 LRN 层，采用更简单的网络。同时，多个小卷积核叠加比单个大卷积核性能好，并且深度更深的网络分类性能更好。因此，VGG 的优点在于，它以简单、便于设置的网络结构达成了较好的性能。同时，它的缺点在于它的全连接层过多导致参数过多难以训练。

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					

图 3: VGG 网络结构

2.1.4 ResNet[4]

ResNet（Residual Network，深度残差网络）提出的主要动机是为了解决深度网络的退化问题。实验中，当网络的层数增加到一定程度时，继续增加反而会导致准确率的降低（出于梯度消失或其它原因）。但是理论上更深的网络不会比更浅的网络准确率低，因为可以将多余的层学习为恒等映射，出现这种问题说明了用非线性层表示恒等映射是很困难的。基于这种想法，ResNet 中令 $\mathcal{H}(x)$ 为需要学习的目标函数，该网络不学习 $\mathcal{H}(x)$，改而学习 $\mathcal{H}(x)-x$，令 $\mathcal{F}(x)=\mathcal{H}(x)-x$，构造如下结构：

$$y=\mathcal{F}(x,W_i)+x$$

其中 x，y 分别表示输入、输出，$\mathcal{F}(x,W_i)$ 表示需要被学习的映射。这样在遇到先前所述难以学习恒等映射的问题时，该网络只需将 $\mathcal{F}(x)$ 学习为 0，降低了学习的难度和资源消耗。

这里 x 是短路直连到输出的，其中假设 x 与维数相同，如果维数不同就对 x 作投影变换。$\mathcal{F}(x)$ 在此网络中为两层或三层网络需要拟合的函数。这样的基础结构即残差块，如图 4所示，

以这样的结构构成了 ResNet，如图 7所示（左图为 VGG）。

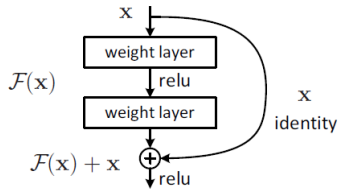


图 4: 残差块

ResNet 在 ImageNet 上的实验结果如图 5所示。对照网络在网络层数为 34 层时的错误率要高于 18 层的网络，说明出现了退化；而 ResNet 在 34 层时的错误率要低于 18 层，说明退化现象被较好地解决了，即可以通过增加网络深度来提高准确率了。同时发现 ResNet 的收敛速度也要更快。

然而使用 ResNet 也并不能无限制地增加深度，如图 6所示，发现在网络过深 (1000 余层) 时，仍然会因为过拟合而出现准确率降低的情况。

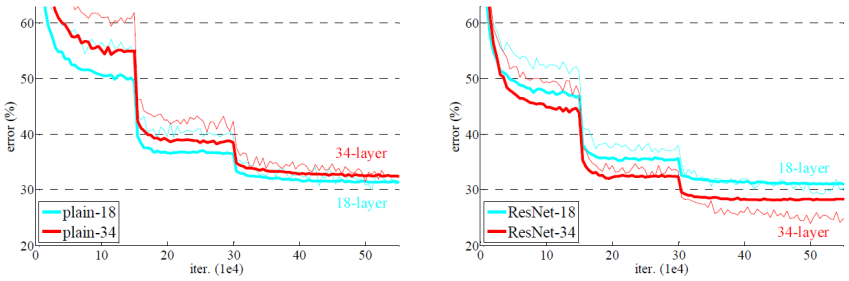


图 5: ResNet 实验结果

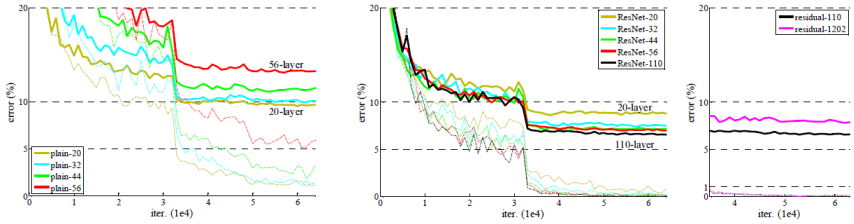


图 6: 深层 ResNet 实验结果

总体而言，ResNet 通过引入残差块提高网络深度，获得更强的学习能力，近来有诸多网络变体，是实现分类网络的较好选择。

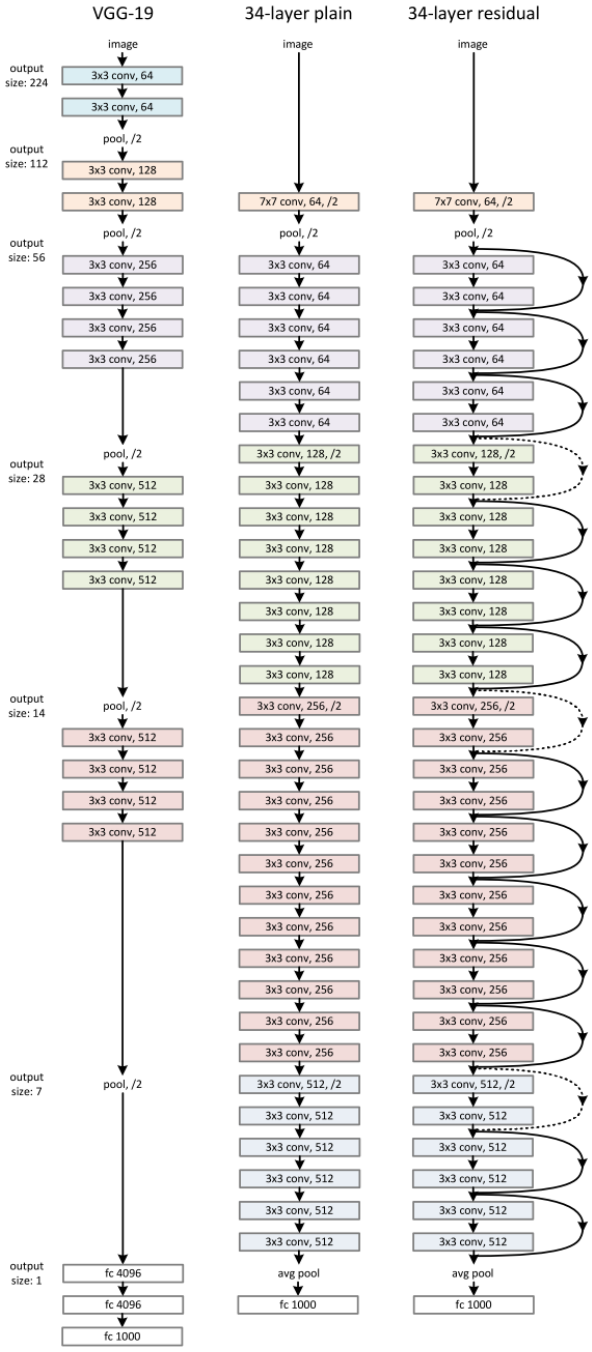


图 7: ResNet 网络结构

2.1.5 匹配网络 [5] 与原型网络 [6]

这两种方法都是用以进行小样本（few shot）分类的。用于小样本分类任务的方法主要有基于 Fine-tune、基于度量与基于图神经网络等多类思路。本小组主要调研了基于度量的小样本分类方法，其中主要有匹配网络（matching network）与原型网络（prototypical network）两种方法。

匹配网络使用注意力机制，在少样本预测（或本实验中单样本预测）中，以简单的加权平均预测未见过的样本 $y = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$，其中 a 为一个注意力机制，$(x_i, y_i)_{i=1}^{k}$ 为 k 对样本-标签对组成的训练集（支持集），$\hat{x}$ 是待预测的样本。

模型的能力很大取决于注意力机制的选取。匹配网络中采用如下的 softmax 模型计算注意力：

$$a(\hat{x}, x_i) = e^{c(f(\hat{x}, g(x_i)))} / \sum_{j=1}^{k} e^{c(f(\hat{x}, g(x_j)))}$$

其中 c 是余弦距离，而 f 和 g 分别为查询集和测试集所嵌入的网络，这两个网络可以相同。图像分类任务中，这两个网络通常为 CNN。原论文中介绍了一些 f 和 g 网络的设计优化方法，但不影响匹配网络的核心思想。算法的流程如图 8所示。

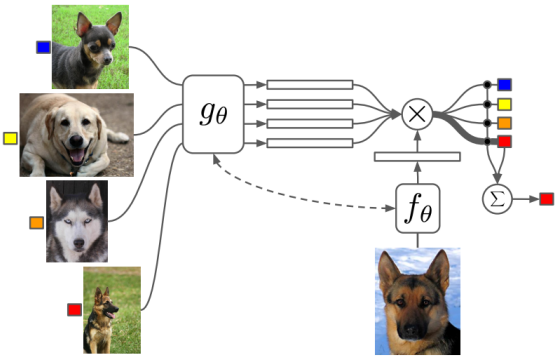
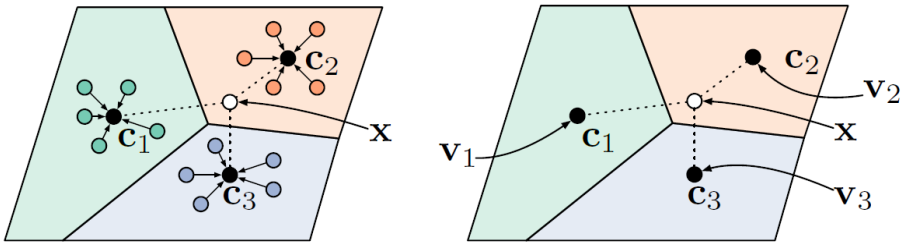


图 8: 匹配网络算法流程

原型网络则较为接近 K 近邻算法，基本思想是将样本投射到一个空间，使得同类较近，异类较远，最后根据测试样本到各类中心的距离分类，如图 9所示。样本主要学习投影函数 $f_\phi : \mathbb{R}^D \to \mathbb{R}^M$。而一个类的原型 $\mathbf{c}_k$ 由类中所有点投影均值决定。再根据到各个类原型的距离，可以通过 softmax 生成概率分布

$$p_\phi(y = k|\mathbf{x}) = \frac{\exp\left(-d(f_\phi(\mathbf{x}), \mathbf{c}_k)\right)}{\sum_{k'} \exp\left(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'})\right)}$$



(a) Few-shot (b) Zero-shot

图 9: 原型网络示意

在训练过程中，以负对数 $J(\phi) = -\log p_\phi(y = k|\mathbf{x})$ 作为损失函数进行训练，具体损失函数计算方法如图 10所示。论文中也考察了不同距离函数的选取（认为使用 Bregman 散度比余弦距离更为有效）以及训练时的样本数的影响，总体而言，原型网络是一个简单有效，准确率较高的少样本分类算法。

Algorithm 1 Training episode loss computation for Prototypical Networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. RANDOMSAMPLE(S, N) denotes a set of N elements chosen uniformly at random from set S, without replacement.

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.
Output: The loss J for a randomly generated training episode.

$V \leftarrow$ RANDOMSAMPLE$(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for episode
for k in $\{1, \ldots, N_C\}$ **do**
  $S_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k}, N_S)$ ▷ Select support examples
  $Q_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k} \setminus S_k, N_Q)$ ▷ Select query examples
  $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\boldsymbol{\phi}}(\mathbf{x}_i)$ ▷ Compute prototype from support examples
end for
$J \leftarrow 0$ ▷ Initialize loss
for k in $\{1, \ldots, N_C\}$ **do**
  for $(\mathbf{x}, y)$ in Q_k **do**
    $J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_{k'})) \right]$ ▷ Update loss
  end for
end for

图 10: 原型网络损失函数算法

2.2 检测

2.2.1 R-CNN 与 Fast R-CNN[7]

R-CNN（region with CNN features）在权威数据集 PASCAL 达到了 53.3% 的水平，将 mAP 在 VOC2012 的结果上提高了 30% 以上。

如图 11所示，R-CNN 的物体检测系统由三部分构成。首先是进行区域推荐（region proposal），论文使用了产生类别无关的选择性搜索（selective search）；其次进行特征提取（feature extraction），具体是在图像转换的基础上，使用一个大型卷积神经网络，通过五个卷积层和两个全连接层进行前向传播，最终实现对每个推荐区域抽取 4096 维度的特征向量；第三个是一个指定类别的线性 SVM。

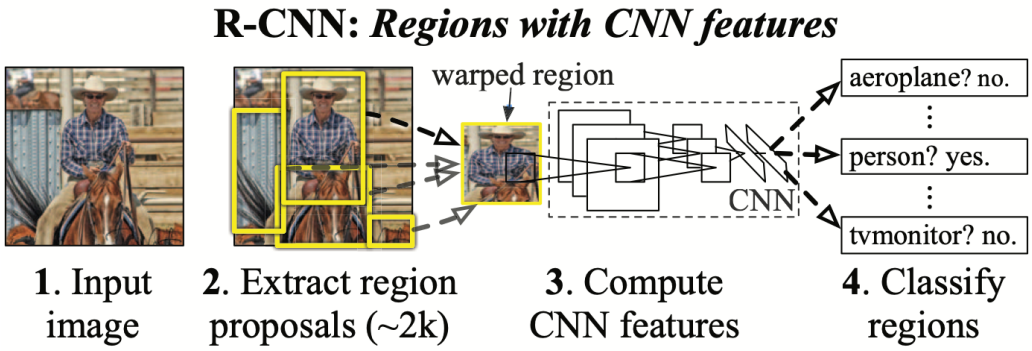


图 11: R-CNN 算法流程

作者认为，本方法取得较高性能的原因主要有两个方面。首先是应用自底向上与区域推荐结合的高容量卷积神经网络进行物体定位与分割，另外一个是在标签匮乏情况下训练神经网络的一个方法。在有监督的情况下进行网络的预训练（supervised pre-training）是很有效的，之后采用稀少数据在特定领域进行定位任务的调优（domain-specific fine-tuning），从而获得了很高的检测精度，如图 12所示。

VOC 2011 test	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
R&P [2]	83.4	46.8	18.9	36.6	31.2	42.7	57.3	47.4	44.1	8.1	39.4	**36.1**	36.3	49.5	48.3	50.7	26.3	47.2	22.1	42.0	43.2	40.8
O_2P [4]	**85.4**	**69.7**	22.3	45.2	**44.4**	46.9	66.7	57.8	56.2	**13.5**	**46.1**	32.3	41.2	**59.1**	55.3	51.0	**36.2**	50.4	**27.8**	46.9	**44.6**	47.6
ours (*full+fg* R-CNN fc_6)	84.2	66.9	**23.7**	**58.3**	37.4	**55.4**	**73.3**	**58.7**	**56.5**	9.7	45.5	29.5	**49.3**	40.1	**57.8**	**53.9**	33.8	**60.7**	22.7	**47.1**	41.3	**47.9**

图 12: R-CNN 实验结果

然而，R-CNN 有着显著的缺陷。R-CNN 的训练过程是一个多级流水线，训练的时间较长，

需要大量的磁盘空间缓存，目标检测速度很慢。而 Fast-RCNN 修正了 R-CNN 的不足，并提高速度与准确性。

Fast R-CNN 的结构如图 13所示，输入图像与多个感兴趣区域（RoI）被输入到几个卷积层网络中，之后将每个 RoI 池化到特征图中，在此基础上抽取了固定长度的特征向量，送入一系列全连接层（fc）中。输出两个向量：softmax 概率和回归偏移量。

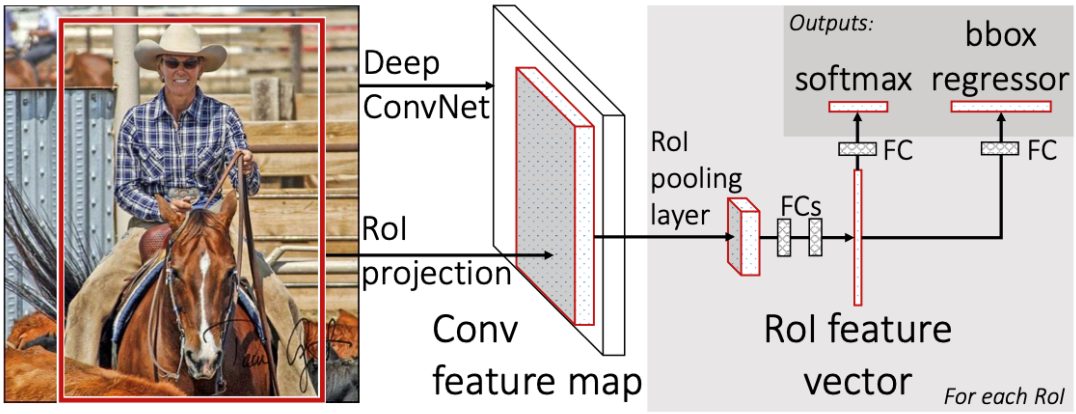


图 13: Fast R-CNN 算法流程

在训练的过程中使用反向传播算法将网络的所有权重进行微调是 Fast R-CNN 的重要能力，具体的过程包括多任务损失、小批量采样、RoI 池化层的反向传播和 SGD 超参数。在微调的同时，Fast R-CNN 还能联合优化 softmax 分类器和检测框回归。网络训练完毕，使用前向传播进行检测，由于计算全连接层花费时间较多，可以使用截断的 SVD 压缩来进行检测加速。

Fast R-CNN 在 VOC12 获得了最高精度；同时，相对于 R-CNN，Fast R-CNN 能进行更快速的训练和检测，实验结果如图 14所示。

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

(a) 速度对比

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	**82.3**	**78.4**	**70.8**	**52.3**	38.7	**77.8**	**71.6**	**89.3**	**44.2**	**73.0**	**55.0**	**87.5**	**80.5**	**80.8**	**72.0**	35.1	**68.3**	**65.7**	**80.4**	**64.2**	**68.4**

(b) 精度对比

图 14: Fast R-CNN 实验结果

2.2.2 YOLO[8]

YOLO（You Only Look Once），顾名思义“只看一次”，使用单个神经网络直接预测物品边界和类别概率。与 R-CNN 方法采用推荐区域生成 bounding box 进而用分类器进行识别相比，YOLO 使用单个神经网络直接提取多个 bounding boxes 和类别概率（如图 15所示），流程简单，速度更快，适合用于实时检测。与分类器不同，YOLO 对直接影响检测性能的损失函数进行训练，并且对整个模型进行整体训练。

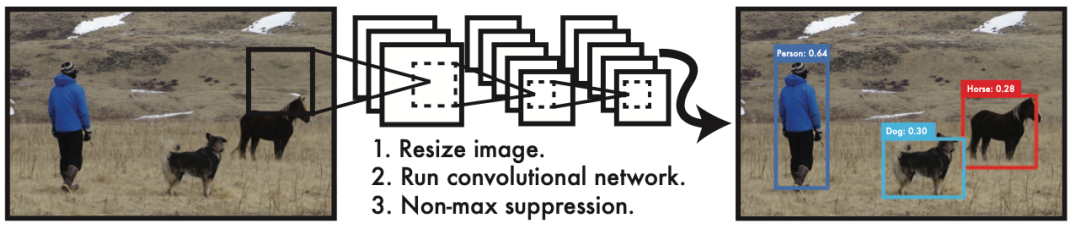


图 15: YOLO 算法流程

考虑到本次实验的要求，与非实时检测系统（如 R-CNN）相比，YOLO 能获得更快的速度和精度（如图 16所示），因而更加适合本实验。

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

图 16: YOLO 实验结果

YOLO 也存在某些方面的不足：每个网格只预测两个边界框和一个类别，距离较近的目标可能影响检测的准确度；在边界框较小的情况下产生的误差对准确度影响较大。

3 研究计划

对于分类任务，本小组计划在任务一中采用 HOG+SVM 方式，其中 SVM 采用网格搜索进行调参。对于任务二，本小组计划采用 ResNet 网络进行训练，通过交叉验证进行模型评价，根据验证集上的训练效果调整网络结构以及预处理机制。如有必要，还可以进行数据增强。任务三中，本小组计划采用原型网络进行单样本学习。在分类任务中，训练后再根据验证集上出错的样本，考察分类出错的主要类别，调整预处理方法，以更有效反映样本的特征。

检测任务中，本小组计划基于当前流行的几种检测算法，如 Faster R-CNN、YOLO 等，对给定数据集中的交通标志进行检测。对于每一个检测结果，期望给出所检测标志的 bounding box 与概率。对实验结果进行统计分析，通过计算 type I error 和 type II error 来评价不同检测算法的性能优劣。最后将表现较好的模型检测结果用作之前实验分类器的输入数据，并进行分类。通过统计分类准确率来评估整个系统的性能优劣。

4 目前进展

目前本小组已初步完成一个任务一的模型，在验证集上获得了 95% 的准确率。我们采用 HOG+SVM 方式，划分 20% 验证集，先在默认参数下对 HOG 参数进行调整，发现当输入图片调整至 64*64 像素，每个区域 4*4 时效果最好。这个归一化图像大小略大于训练图像的平均大小（约 47*42）。随后使用 5-fold 网格搜索调整 SVM 参数，发现 C=100，gamma=0.1 时精度最高，达到 95%，具体如 17所示。目前还没有加入调试 test.json 的接口，代码也将会继续完善。

```
The parameters of the best model are:
{'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
Classification report for classifier GridSearchCV(cv=5, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=12,
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'gamma': [10, 1, 0.1, 0.01, 0.001],
                         'kernel': ('linear', 'rbf')},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='f1_weighted', verbose=1):
              precision    recall  f1-score   support

          i2       0.88      0.86      0.87        51
          i4       0.97      0.95      0.96       117
          i5       1.00      0.98      0.99       251
          io       0.94      0.95      0.94       139
          ip       0.96      1.00      0.98        44
         p11       0.94      0.97      0.96       270
         p23       0.96      0.96      0.96        26
         p26       0.95      0.93      0.94       136
          p5       0.98      0.95      0.96        42
         p30       0.94      0.93      0.93        67
         p40       0.96      0.98      0.97       211
          pl5       1.00      0.97      0.98        62
         pl50       0.94      0.94      0.94       160
         pl60       0.98      0.95      0.97       123
         pl80       0.97      0.95      0.96       132
           pn       0.93      0.98      0.95       480
          pne       0.99      0.98      0.99       341
           po       0.91      0.81      0.86       189
          w57       1.00      0.96      0.98        52

    accuracy                           0.95      2893
   macro avg       0.96      0.95      0.95      2893
weighted avg       0.95      0.95      0.95      2893
```

图 17: 任务一实验结果

代码如下：

```
from skimage.feature import hog
from sklearn import svm, metrics
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
import joblib
from PIL import Image
import numpy as np
import os


def read_data(data_dir):
    datas = []
    labels = []
    for label in os.listdir(data_dir):
        print(label)
        class_path = os.path.join(data_dir, label)
        for img_name in os.listdir(class_path):
            img = Image.open(os.path.join(class_path, img_name))
            out = img.resize((64, 64), Image.ANTIALIAS)
            fd = hog(out, orientations=9, pixels_per_cell=(4, 4), cells_per_block=(16, 16), block_norm='
                L2',
                    feature_vector=True, multichannel=True)
            datas.append(fd)
            labels.append(label)

    datas = np.array(datas)
    labels = np.array(labels)
    np.savez('hog', datas=datas, labels=labels)
    return datas, labels


datas, labels = read_data('data/Classification/Data/Train')

svr = svm.SVC()
parameters = {'kernel': ('linear', 'rbf'), 'C': [0.1, 1, 10, 100, 1000], 'gamma': [10, 1, 0.1, 0.01,
    0.001]}
classifier = GridSearchCV(svr, parameters, scoring='f1_weighted', n_jobs=12, cv=5, verbose=1)

X_train, X_test, y_train, y_test = train_test_split(datas, labels, test_size=0.2)

classifier.fit(X_train, y_train)
print('The parameters of the best model are: ')
print(classifier.best_params_)
joblib.dump(classifier, 'svm.m')
predicted = classifier.predict(X_test)
print("Classification report for classifier %s:\n%s\n" % (classifier, metrics.classification_report(
```

```
        y_test, predicted)))
```

参考文献

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[2] D. CireşAn, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural networks*, vol. 32, pp. 333–338, 2012.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[6] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, 2017, pp. 4077–4087.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.