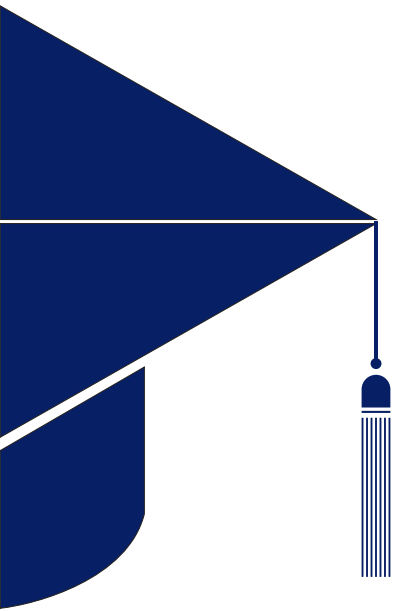


# 目标检测第一讲

# 什么叫目标检测

- 给定一张图片就可以识别出类别，就是**对象识别。**
- 而**目标检测**除了要识别类别外，还要找出它们的位置。
- 针对单个目标的任务，**识别就是给定一张图片，要让计算机告诉你图片中是什么。定位任务不仅要识别出图像中是什么，还要给出目标在图像中的位置信息。**简单的说，就是用一个矩形框把识别的目标框出来（有时候也有多个固定数量的目标）。

# 什么叫目标检测

Classification

CAT

Classification
+ Localization

CAT

# 相关专业术语介绍

- 交并比（Intersection Over Union , **IoU**）
- **Ground truth，候选区域**
- **AP**(Average Precision） ， **mAP**(mean Average Precision)
- **选择性搜索（selective search）**
- **非极大值抑制（Non-Maximum Suppression）**
- **自低向上（bottom-up)**
- **PASCAL VOC数据集**
- **微调（fine tune)**
- **预训练模型（pre-trained model)**
- **边框回归(Bounding Box Regression)**
- **Hard negative mining**

# 相关专业术语介绍

## 一、 PASCAL VOC数据集：

PASCAL VOC为图像识别和分类提供了一整套标准化的优秀的数据集，从2005年到2012年每年都会举行一场图像识别挑战。解压后的**VOCdevkit**目录下的**VOC2012**中。


其中在图像物体识别上着重需要了解的是**Annotations**、**ImageSets**和**JPEGImages**。

# 相关专业术语介绍

## 1.1PASCAL VOC数据集的三个主要文件夹：

### 1.1.1. JPEGImages文件夹

**JPEGImages**文件夹中包含了PASCAL VOC所提供的所有的图片信息，包括了训练图片和测试图片。这些图像都是以“**年份_编号.jpg**”格式命名的。

图片的像素尺寸大小不一，但是横向图的尺寸大约在**500*375**左右，纵向图的尺寸大约在**375*500**左右，基本不会偏差超过100。

（在之后的训练中，**第一步就是将这些图片都resize到300*300或是500*500**，所有原始图片不能离这个标准过远。）

# 相关专业术语介绍

**1.1.1JPEGImages文件夹：**

图像展示如下：

# 相关专业术语介绍

## 1.1PASCAL VOC数据集的三个主要文件夹：

### 1.1.2. Annotations文件夹

Annotations文件夹中存放的是xml格式的标签文件，每一个xml文件都对应于JPEGImages文件夹中的一张图片。存放了图片的信息。

# 相关专业术语介绍

```
1.<annotation>
2.    <folder>VOC2012</folder>
3.    <filename>2007_000392.jpg</filename>                          //文件名
4.    <source>
5.        <database>The VOC2007 Database</database>
6.        <annotation>PASCAL VOC2007</annotation>
7.        <image>flickr</image>
8.    </source>
9.    <size>                                                  //图像尺寸（长宽以及通道数）
10.        <width>500</width>
11.        <height>332</height>
12.        <depth>3</depth>
13.    </size>
14.    <segmented>1</segmented>
15.    <object>                                                    //检测到的物体
16.        <name>horse</name>                                    //物体类别
17.        <pose>Right</pose>
18.        <truncated>0</truncated>
19.        <difficult>0</difficult>
20.        <bndbox>                                              //bounding-box（包含左下角和右上角xy坐标）
21.<annotation>
22.    <folder>VOC2012</folder>
23.    <filename>2007_000392.jpg</filename>                          //文件名
24.    <source>
25.        <database>The VOC2007 Database</database>
```

……

# 相关专业术语介绍

## 1.1PASCAL VOC数据集的三个主要文件夹：

### 1.1.3. ImageSets文件夹

ImageSets存放的是每一种类型的challenge对应的图像数据。

在ImageSets下有四个文件夹：

# 相关专业术语介绍

## 1.1PASCAL VOC数据集的三个主要文件夹：

### 1.1.3. ImageSets文件夹

其中**Action**下存放的是人的动作（例如running、jumping等等，这也是VOC challenge的一部分）

**Layout**下存放的是具有人体部位的数据（人的head、hand、feet等等，这也是VOC challenge的一部分）。

**Main**下存放的是图像物体识别的数据，总共分为20类。

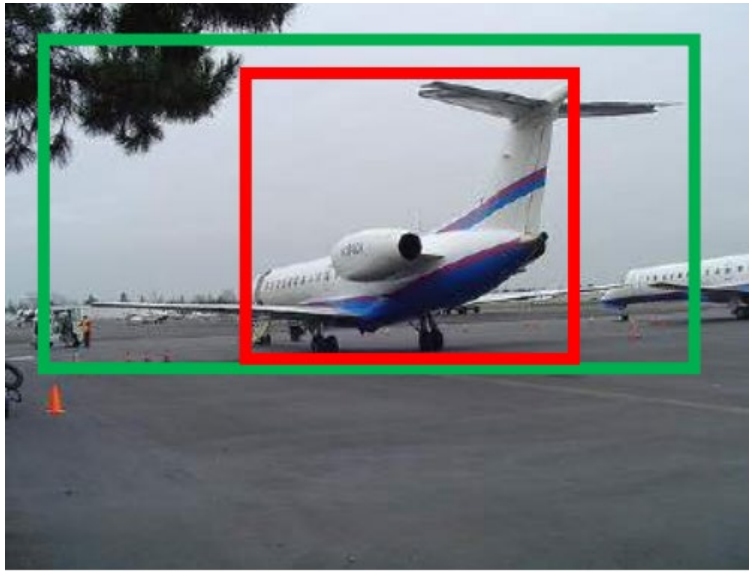
**Segmentation**下存放的是可用于分割的数据。

# 相关专业术语介绍

## 二、Ground Truth和候选区域：

- 候选区域(Region proposal)：通过某种方式在输入图片上生成一系列可能包含物体的区域，这些区域成为候选区域。
- Ground truth是摄影、测量与遥感学领域常用词汇，其解释就是字面意思：地面真值，地面实况；延伸到图像处理、机器学习等其他领域一般表示真实值，正确答案（或正确测量数据）。它是一个正确的基准值，一般用来进行误差估算和效果评价。

# 相关专业术语介绍



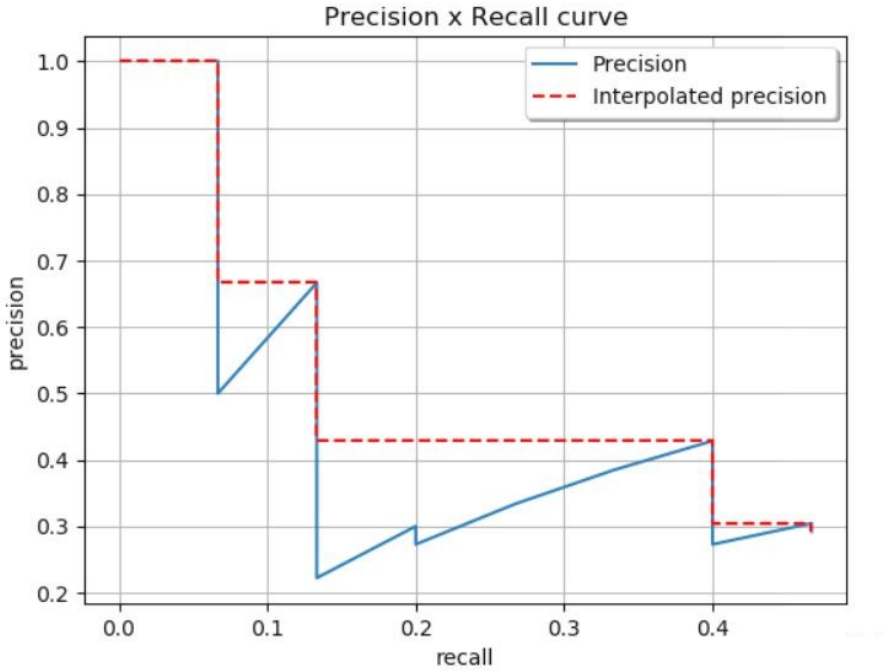
绿色框是我们的实际要达到的检测框（Ground Truth），红色框是我们生成的Region Proposal。

# 相关专业术语介绍

## 三、AP(Average Precision），mAP(mean Average Precision)：

- Precision-Recall曲线可以衡量目标检测模型的好坏，但不便于模型和模型之间比较。
- 在Precision-Recall曲线基础上，通过计算每一个recall值对应的Precision值的平均值，可以获得一个数值形式的评估指标：AP(Average Precision），用于衡量的是训练出来的模型在感兴趣的类别上的检测能力的好坏。
- 我们现在常用的算法是每个Recall的区间内我们只取这个区间内Precision的最大值然后和这个区间的长度做乘积，最后体现出来就是一系列的矩形的面积。

# 相关专业术语介绍




对于多个目标，我们计算所有目标AP的平均值作为目标检测最终的性能评价指标即 mean average precision (mAP)。

# 相关专业术语介绍

## 四、选择性搜索（selective search）

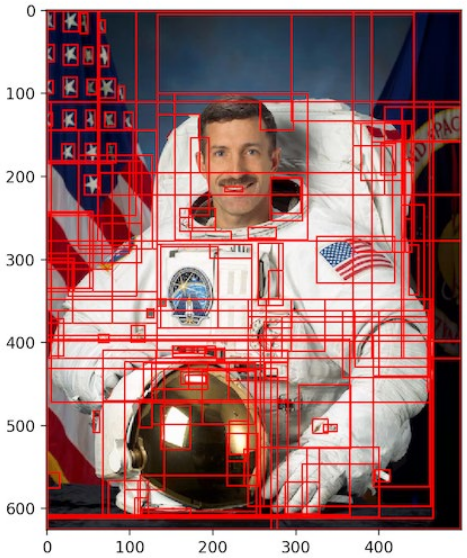
**参考论文：**Selective Search for Object Recognition

这篇论文的标题虽然也提到了 Object Recognition ，但就创新点而言，其实在 Selective Search 。

# 相关专业术语介绍

## 4.1 什么是选择性搜索？ （selective search）

Selective Search，说的简单点，就是从图片中找出物体可能存在的区域。




左边这幅宇航员的图片中，那些红色的框就是 Selective Search 找出来的可能存在物体的区域。

# 相关专业术语介绍

在进一步探讨它的原理之前，论文里提出了一个问题：如何判别哪些 region 属于一个物体？

(a) (b)

(c) (d)

作者在论文中用以上四幅图，分别描述了四种可能的情况：

1.图 a ，物体之间可能存在层级关系，比如：碗里有个勺；

2.图 b，我们可以用颜色来分开两只猫，却没法用纹理来区分；

3.图 c，我们可以用纹理来区分变色龙，却没法用颜色来区分；

4.图 d，轮胎是车的一部分，不是因为它们颜色相近、纹理相近，而是因为轮胎包含在车上。

# 相关专业术语介绍

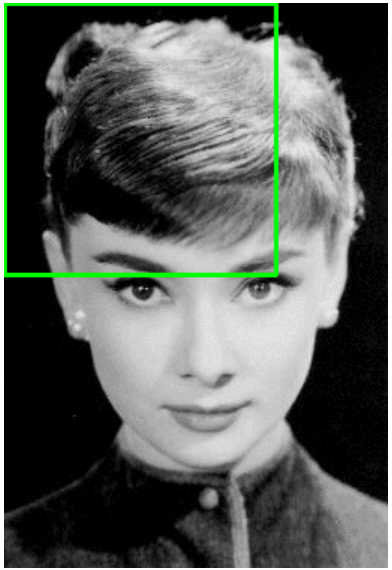
综上所述，我们没法用单一的特征来定位物体，需要综合考虑多种策略，

这一点是 Selective Search 精要所在。

# 相关专业术语介绍

## 4.2 候选框的查找方法

在此算法之前选择的常用方法是滑动窗口法：



滑动窗口(sliding window)，即遍历法，设计思想简单，计算量巨大，不实用。其问题在于：漫无目的性的搜索。

因为我不知道目标的大小，设置不同大小的窗口对候选框查找结果有很大的影响，而且滑动窗的步长太小会产生过多的候选框，带来很大的计算量，步长太大又容易错过精确的目标候选框，对于实时性和速度要求较高时不推荐使用滑窗法。

# 相关专业术语介绍

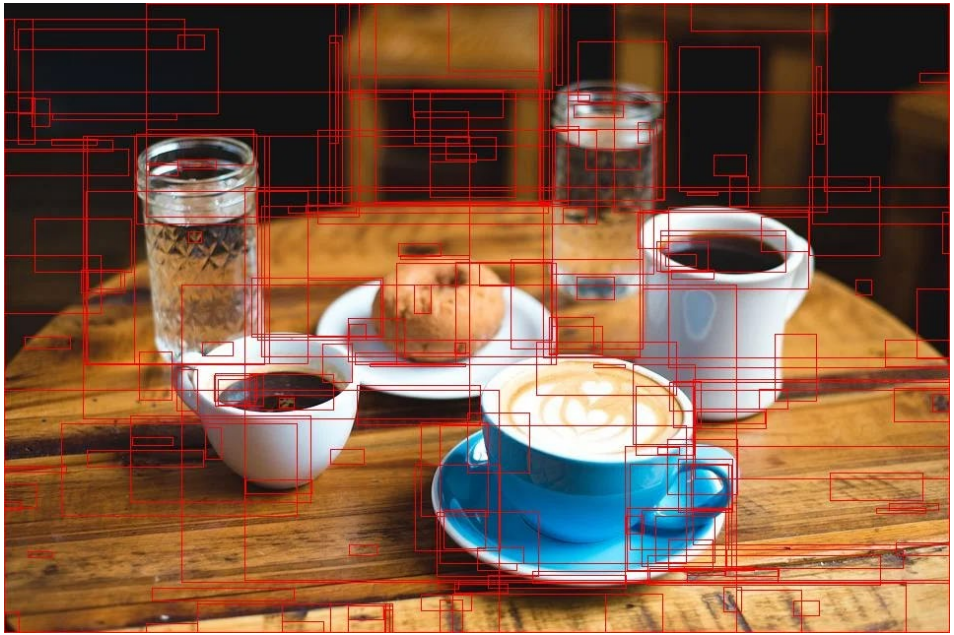
## 4.2 候选框的查找方法

选择性搜索(selective search)，根据相似程度（颜色，纹理，大小，形状等）来提出包含目标对象可能性更大的候选区域，候选框从数十万(滑动窗口法)降到几千(R-CNN中两千左右），从而极大的降低了计算量，提高了检测速度。

# 相关专业术语介绍

## 4.2 候选框的查找方法

selective search算法获得的候选框展示如下：

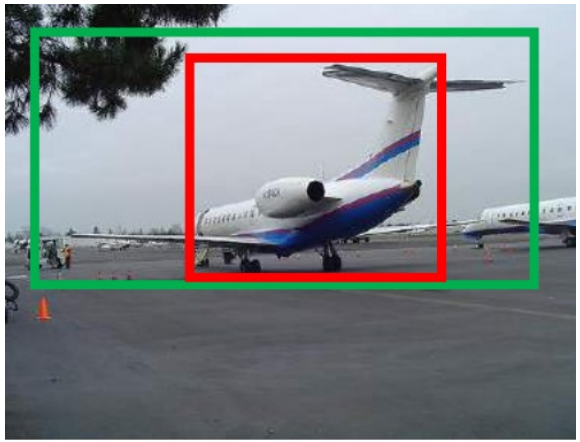


# 相关专业术语介绍

## 五、IoU交并比

为了计算算法提出的候选框的准确性，我们引入了IoU（Intersection Over Union）指标来计算候选框和目标实际标注边界框的重合度。

如下图所示，红色框是我们生成的候选框R，绿色是我们要达到的检测框G。



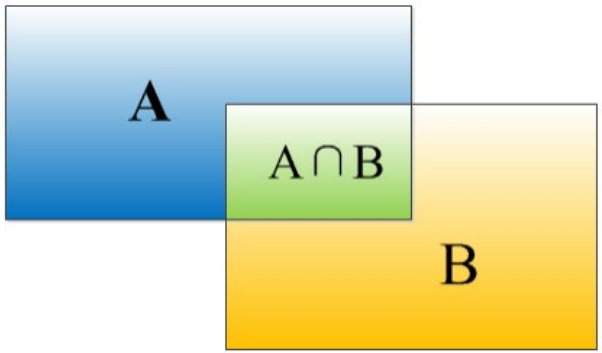
定义的计算公式为：

$$(R \cap G) / (R \cup G)$$

# 相关专业术语介绍

## IoU交并比：

为了更清晰明了的去得到计算结果，我们从数学的角度入手写出详细的计算过程来：




假如我们要计算两个矩形框A和B的IoU，就是它们的交集与并集之比。

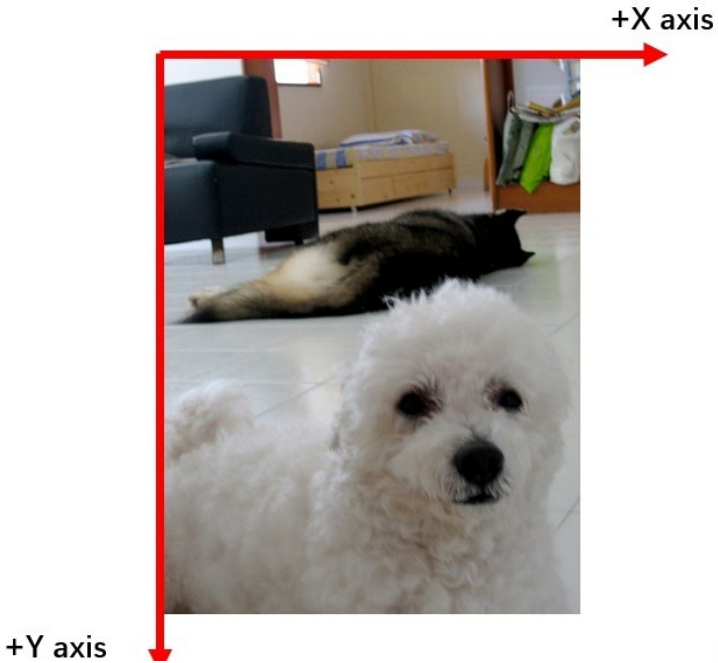
- IoU 为 0 时，两个框不重叠，没有交集。
- IoU 为 1 时，两个框完全重叠。
- IoU 取值为 0 ~ 1 之间的值时，代表了两个框的重叠程度，数值越高，重叠程度越高。

做检测任务时，会设置IoU的阈值(voc数据集通常是0.5)，为度量目标定位精度，当大于这个阈值时表示成功检测，否则表示漏检。

# 相关专业术语介绍

## IoU交并比的计算：

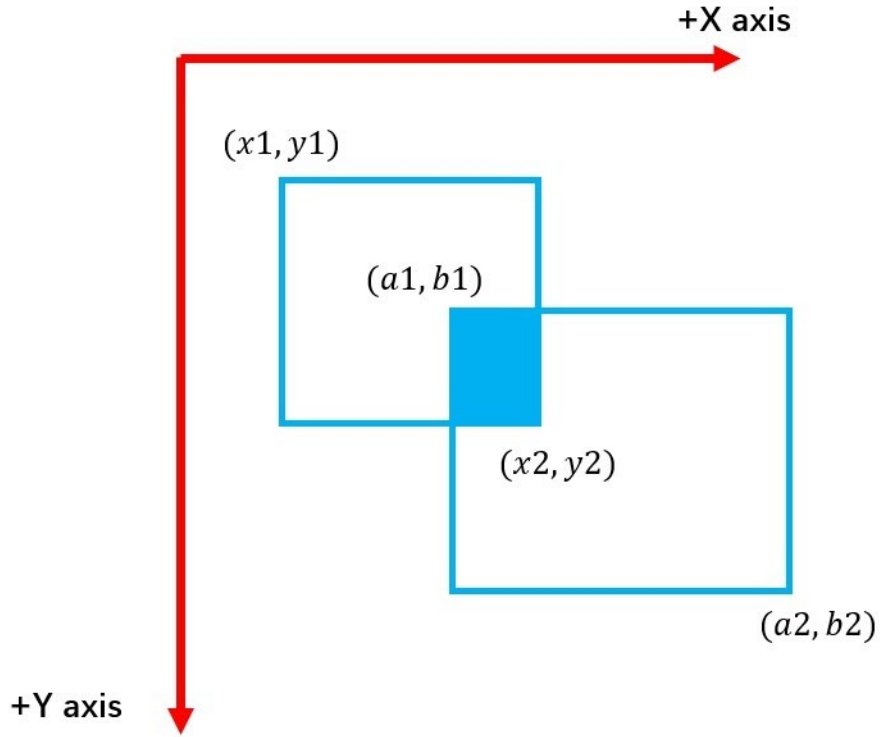
首先我们规定，以一张图像的左上角为原点建立一个坐标系，原点往右为X轴的正方向，原点往下为Y轴的正方向（这点很重要），如下图所示：



# 相关专业术语介绍

## IoU交并比的计算：

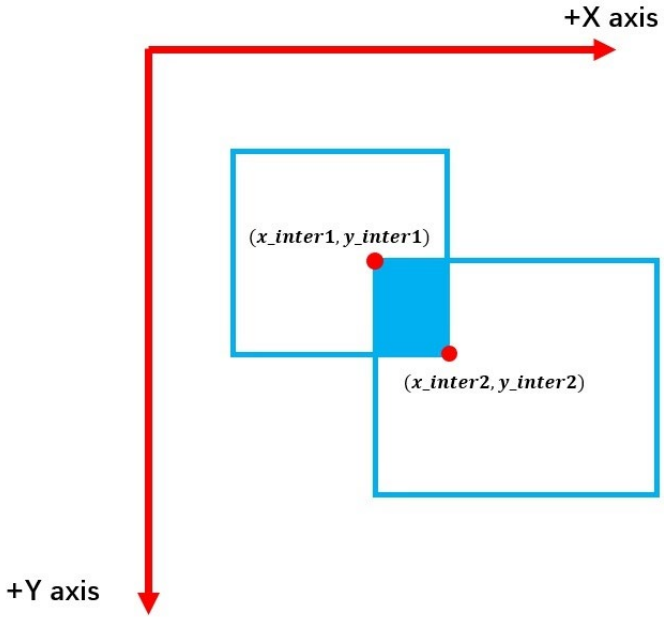
刚才添加图片是为了方便理解，现在我们将图片移开，假设在图上有两个bounding box：



# 相关专业术语介绍

## IoU交并比的计算：

我们在计算IoU的时候，需要得到overlap部分的左上角坐标和右下角坐标，也就是下图中的(x_inter1, y_inter1)和(x_inter2, y_inter2)。



# 相关专业术语介绍

## IoU交并比的计算：

对比一下原图，我们可以发现几个关系，在求解overlap部分左上角的坐标时，我们会对比两个bounding box的左上角坐标，此时我们可以发现，x_inter1的位置应该尽可能靠右，y_inter1的位置应该尽可能靠下。而向右和向下同时意味着相对于坐标系而言，我的x值和y值应该尽量地大，即：而向右和向下同时意味着相对于坐标系而言，我的x值和y值应该尽量地大，即：

$$x\_inter1 = max(x1,a1)$$

$$y\_inter1 = max(y1,b1)$$

# 相关专业术语介绍

## IoU交并比的计算：

在求解overlap部分右下角坐标时，我们会对比两个bounding box的右下角坐标，此时x_inter2应该尽可能靠左，y_inter2应该尽可能靠上，意味着x值和y值应该尽量地小，即：

**x_inter2 = min(x2,a2)**

**y_inter2 = min(y2,b2)**

overlap部分的面积求解如下:

**Width = (x_inter2-x_inter1)**

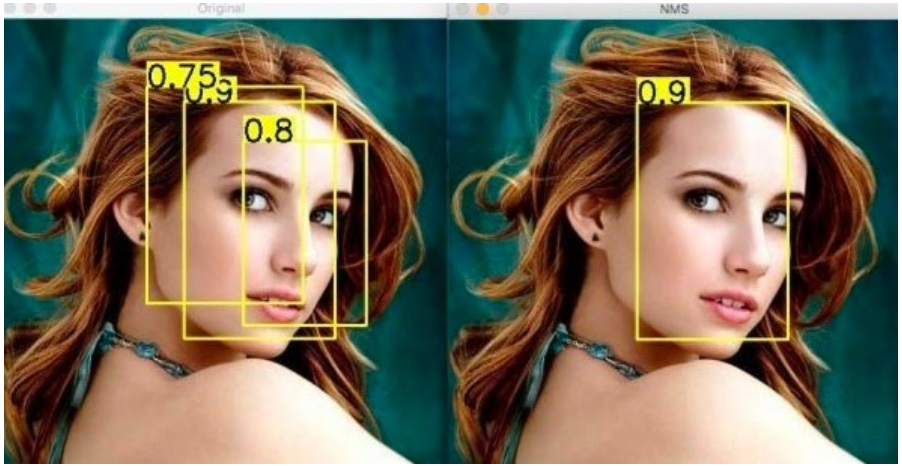
**Height = (y_inter2-y_inter1)**

**Area_overlap = Width*Height**

# 相关专业术语介绍

## 六、非极大值抑制（Non-Maximum Suppression）

在目标检测任务中，目标检测的过程中会在同一目标的位置上会产生大量的候选框，这些候选框相互之间可能会有重叠，此时需要利用非极大值抑制找到最佳的目标边界框，消除冗余的边界框。




左图是人脸检测的候选框结果，每个边界框有一个置信度得分(confidence score)，如果不使用非极大值抑制，就会有多个候选框出现。右图是使用非极大值抑制之后的结果，符合我们人脸检测的预期结果。

# 相关专业术语介绍

## 六、非极大值抑制（Non-Maximum Suppression）

非极大值抑制的流程如下：

- 根据置信度得分进行排序；
- 选择置信度最高的比边界框添加到最终输出列表中，将其从边界框列表中删除；
- 计算所有边界框的面积；
- 计算置信度最高的边界框与其它候选框的IoU；
- 删除IoU大于阈值的边界框；
- 重复上述过程，直至边界框列表为空。

# 相关专业术语介绍

## 七、自低向上（bottom-up）

首先来看三张图片：



# 相关专业术语介绍

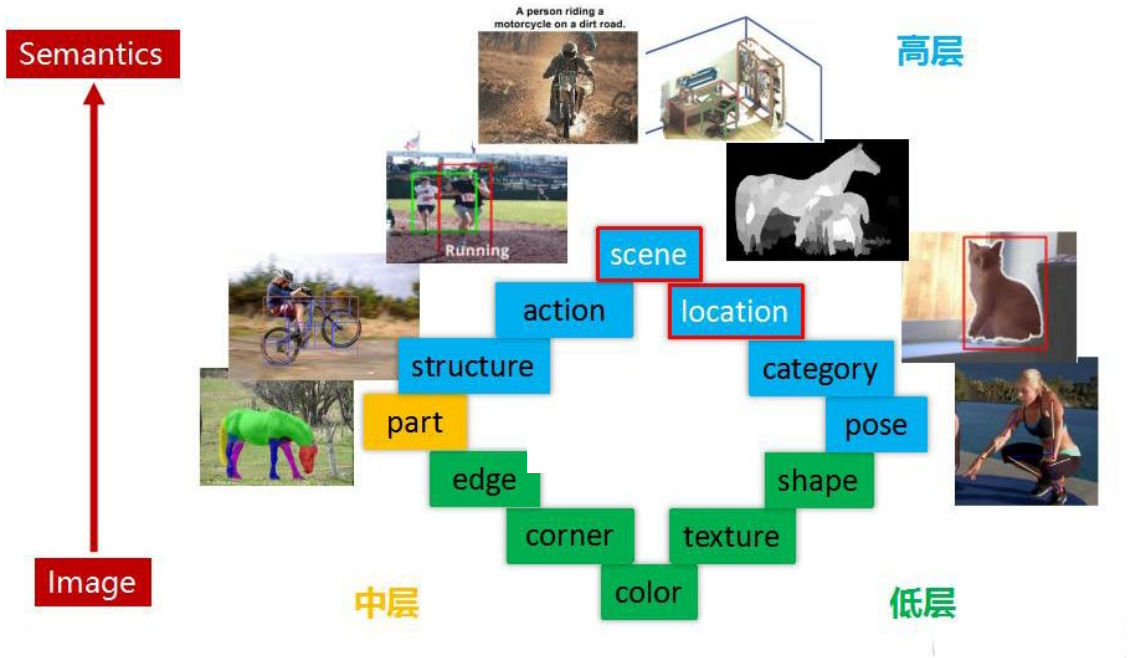
## 七、自低向上（bottom-up）

先看第一张，虽然前面云雾缭绕，大家依然知道这是古建筑，但如果让计算机来识别这张图就很困难。大家还能在这张图片上看到什么？是的，后面还有城市。哪个是显著性的目标？对图像本身来说，很多人会认为古建筑是显著的，因为它在图像的中间而且对比度最强。为什么云不是显著的？如果有人会关心云,云就应该是显著的。还有人关注的是城市，为什么后面的城市不是显著的呢？仅仅因为它在图像中表现比较弱，它就不显著了吗？下面我们来看第二张图，这个人带着彩烟跑，看样子还挺开心，然而却给监控带来了非常大的挑战。再看第三张图，在这种情况下，我们还能认出这是一辆汽车吗？

所以我们对于图像的认知应该从那些方面入手？

# 相关专业术语介绍

## 七、自低向上（bottom-up）



# 相关专业术语介绍

## 七、自低向上（bottom-up）

人对图像中语义的理解可分为中层、低层和高层。传统的图像识别是下面绿色的部分，其中包括边缘、角点、颜色、纹理、形状。这是我们的基石，也是过去几十年来取得成果最多的地方。低层特征的识别在深入研究的任务中显然是不够的，于是，近些年出现了很多关于可鉴别部件“PART”的研究工作，个人认为“PART”只是一个过渡，但是在我们找到认知的模型前还必须对它进行研究。“PART”之上是结构，结构是为了解决在识别时如何把“PART”组织在一起，还有动作、场景、定位、分类、姿态的问题。关于显著性物体检测，我认为涉及到场景和定位两方面。这也是我们拿到图以后要做的第一件事，首先要知道在哪里，然后再判断它是什么。

# 相关专业术语介绍

## 八、Hard negative mining

该思路源自于论文《Rich feature hierarchies for accurate object detection and semantic segmentation》，就是我们应该提到的两阶段目标检测的开山奠基之作。

**R-CNN中的hard negative mining：**

对于现在的我们，首先遇到难负例挖掘应该是R-CNN的论文，论文关于hard negative mining的部分引用了两篇论文:

- Object detection with discriminatively trained part based models
- Example-based learning for viewbased human face detection

# 相关专业术语介绍

## 八、Hard negative mining

1. Bootstrapping methods train a model with an initial subset of negative examples, and then collect negative examples that are incorrectly classified by this initial model to form a set of hard negatives. A new model is trained with the hard negative examples, and the process may be repeated a few times.
2. we use the following “bootstrap” strategy that incrementally selects only those “nonface” patterns with high utility value:
   1)Start with a small set of “nonface” examples in the training database.
   2)Train the MLP classifier with the current database of examples.
   3)Run the face detector on a sequence of random images. Collect all the “nonface” patterns that the current system wrongly classifies as “faces” (see Fig. 5b).Add these “nonface” patterns to the training database as new negative examples.
   4) Return to Step 2.

# 相关专业术语介绍

## 八、Hard negative mining

1.背景知识

难例挖掘与非极大值抑制 NMS 一样，都是为了解决目标检测老大难问题（样本不平衡+低召回率）及其带来的副作用。

首先，目标检测与图像分类不同，图像分类往往只有一个输出，但目标检测的输出个数却是未知的。除了Ground-Truth（标注数据）训练时，模型永远无法百分百确信自己要在一张图上预测多少物体。

所以目标检测问题的老大难问题之一就是如何提高召回率。

# 相关专业术语介绍

## 八、Hard negative mining

### 1.背景知识

召回率（Recall）是模型找到所有某类目标的能力（所有标注的真实边界框有多少被预测出来了）。检测时按照是否检出边界框与边界框是否存在，可以分为下表四种情况：

| | | True condition | | | |
|---|---|---|---|---|---|
| | Total population | Condition positive | Condition negative | Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$ | Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$ |
| Predicted condition | Predicted condition positive | True positive | False positive, Type I error | Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$ | False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$ |
| | Predicted condition negative | False negative, Type II error | True negative | False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$ | Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$ |
| | | True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$ | False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$ | Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$ | Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$ |
| | | False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$ | Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$ | Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$ | $F_1$ score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$ |

# 相关专业术语介绍

## 八、Hard negative mining

### 1.背景知识

为了提高这个值，很直观的想法是**“宁肯错杀一千，绝不放过一个”**。因此在目标检测中，模型往往会提出远高于实际数量的候选框。

但此时就会遇到一个问题，因为区域提议实在太多，导致在训练时绝大部分都是负样本，这导致了大量无意义负样本的梯度“淹没”了有意义的正样本。

根据Focal Loss论文的统计，通常包含少量信息的“easy examples”（通常是负例），与包含有用信息的“hard examples”（正例+难负例）之比为100000：100！这导致这些简单例的损失函数值将是难例损失函数的40倍！

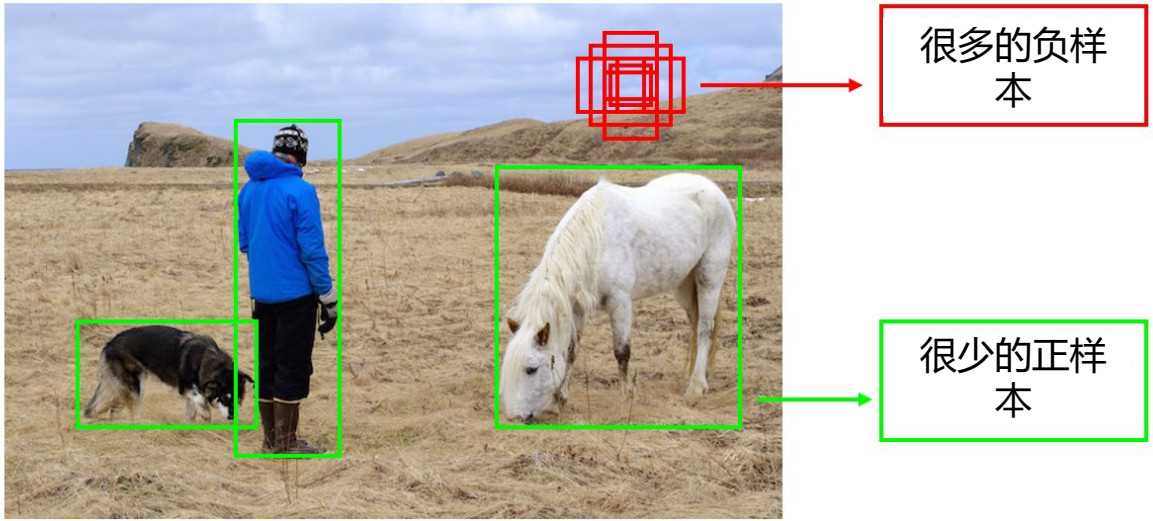
# 相关专业术语介绍

## 八、Hard negative mining

1.背景知识

100000 easy vs. 100 hard examples

**40x bigger loss** from easy examples!



# 相关专业术语介绍

## 八、Hard negative mining

1.背景知识

因此，为了让模型正常训练，我们必须要通过某种方法抑制大量的简单负例，挖掘所有难例的信息，这就是难例挖掘的初衷。

难负例挖掘（Hard Negative Mining）就是在训练时，尽量多挖掘些难负例（hard negative)加入负样本集，这样会比easy negative组成的负样本集效果更好。

# 相关专业术语介绍

## 九、预训练模型（pre-trained model）

预训练模型就是已经用数据集训练好了的模型。

现在我们常用的预训练模型就是他人用的常用模型，比如VGG16/19，并用大型数据集来做训练集，比如imagenet，COCO等训练好的模型参数。

# 相关专业术语介绍

## 十、微调（Fine Tune）

### 10.1什么是微调?

1.假设我们的神经网络符合形式：y=w*x；

2. 现在我们要找到一个w，使得当输入x=2时，输出y=1，也就是希望w=0.5；

3.首先对w进行初始化，初始化的值均服从均值为0，方差为1的分布，假设w初始化为0.1：y=0.1*x；

4.当输入x=2时，w=0.1，输出y=0.2，这个时候实际值和目标值1的误差是0.8；

5.0.8的误差经过反向传播去更新权值w，假如这次更新为w=0.2，输出为0.4，与目标值的误差为0.6；

5.可能经过十次或二十次反向传播，w终于等到我们想要的0.5；

# 相关专业术语介绍

**10.1什么是微调?**

7.如果在更新模型最开始有人告诉你，w的值应该在0.47附近：y=0.47*x；

8.那么从最开始训练，你的目标值的误差就只有0.06了，那么可能只需要一步两步就能把w训练到0.5。

总结：第七步就是相当于给我们一个预训练模型（pre-trained model），第八步就是基于这个模型微调。相当于我们从头开始训练，微调为我们省去了大量计算计算资源和计算时间，提高了计算效率，甚至提高准确率。

# 相关专业术语介绍

## 10.2为什么要进行微调?

普通预训练模型的特点是:

用了大型数据集做训练，已经具备了提取浅层基础特征和深层抽象特征的能力。

**不做微调:**

1）从头开始训练，需要大量的数据、计算时间和计算资源;

2）存在模型不收敛，参数不够优化、准确率低、模型泛化能力低，容易过拟合等风险。

**使用微调:**

有效避免了上述可能存在的问题。

# 相关专业术语介绍

## 十一、边界框回归

相比传统的图像分类，目标检测不仅要实现目标的分类，而且还要解决目标的定位问题，即获取目标在原始图像中的位置信息。

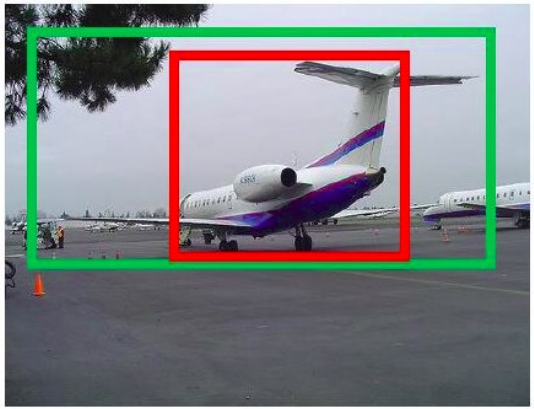
在不管是最初版本的R-CNN，还之后的改进版本——Fast R-CNN和Faster R-CNN都需要利用边界框回归来预测（矫正）物体的目标检测框，以提高最终的检测精度。因此掌握边界框回归（Bounding-Box Regression）是极其重要的，这是熟练使用R-CNN系列模型的关键一步，也是代码实现中比较重要的一个模块。

# 相关专业术语介绍

## 11.1为什么要做Bounding-box regression?

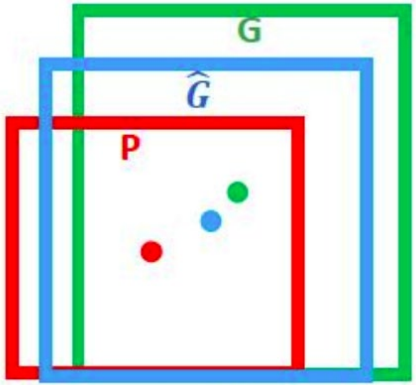
绿色的框为飞机的Ground Truth，红色的框是Selective Search提取的Region Proposal。那么即便红色的框被分类器识别为飞机，但是由于红色的框定位不准（IoU<0.5），那么这张图相当于没有正确的检测出飞机。如果我们能对红色的框进行微调，使得经过微调后的窗口跟Ground Truth 更接近，这样岂不是定位会很准确。

确实，Bounding-box regression就是用来微调这个窗口的。



# 相关专业术语介绍

## 11.2问题数学表达




对于窗口，一般使用四维向量（x，y，w，h）来表示，分别表示窗口的中心点坐标和宽高。对于左图，红色的框代表原始的Proposal候选目标框对于窗口，蓝色的框代表边界框回归算法预测的目标框，绿色的框代表目标的Ground Truth真实目标框的中心框。三种颜色的点分别带别对应颜色的中心点。

# 相关专业术语介绍

## 11.2问题数学表达

边框回归的目的既是：给定 $(P_x, P_y, P_w, P_h)$ 寻找一种映射 $f$， 使得 $f(P_x, P_y, P_w, P_h) = (\tilde{G}_x, \tilde{G}_y, \tilde{G}_w, \tilde{G}_h)$ 并且 $(\tilde{G}_x, \tilde{G}_y, \tilde{G}_w, \tilde{G}_h) \approx (G_x, G_y, G_w, G_h)$

## 11.3边框回归怎么做的?

那么经过何种变换才能从图 2 中的窗口 P 变为窗口呢？ 比较简单的思路就是: *平移+尺度放缩*

1. 先做平移$(\Delta x, \Delta y)$， $\Delta x = P_w d_x(P), \Delta y = P_h d_y(P)$ 这是R-CNN论文的:

$$\hat{G}_x = P_w d_x(P) + P_x, (1)$$

$$\hat{G}_y = P_h d_y(P) + P_y, (2)$$

2. 然后再做尺度缩放$(S_w, S_h)$, $S_w = exp(d_w(P)), S_h = exp(d_h(P))$, 对应论文中:

$$\hat{G}_w = P_w exp(d_w(P)), (3)$$

$$\hat{G}_h = P_h exp(d_h(P)), (4)$$

# 相关专业术语介绍

## 11.2问题数学表达

有了这四个变换我们就可以直接得到 Ground Truth，这里还有个问题，根据(1)~(4)我们可以知道，P经过变换得到的并不是真实值G，而是预测值。的确，这四个值应该是经过 Ground Truth 和 Proposal 计算得到的真正需要的平移量和尺度缩放 。

$$t_x = (G_x - P_x)/P_w, (6)$$

$$t_y = (G_y - P_y)/P_h, (7)$$

$$t_w = \log(G_w/P_w), (8)$$

$$t_h = \log(G_h/P_h), (9)$$

# 相关专业术语介绍

## 11.2问题数学表达

那么目标函数可以表示为，是输入Proposal的特征向量，是要学习的参数（*表示 x，y，w，h， 也就是每一个变换对应一个目标函数），是得到的预测值。我们要让预测值跟真实值差距最小，得到损失函数为：

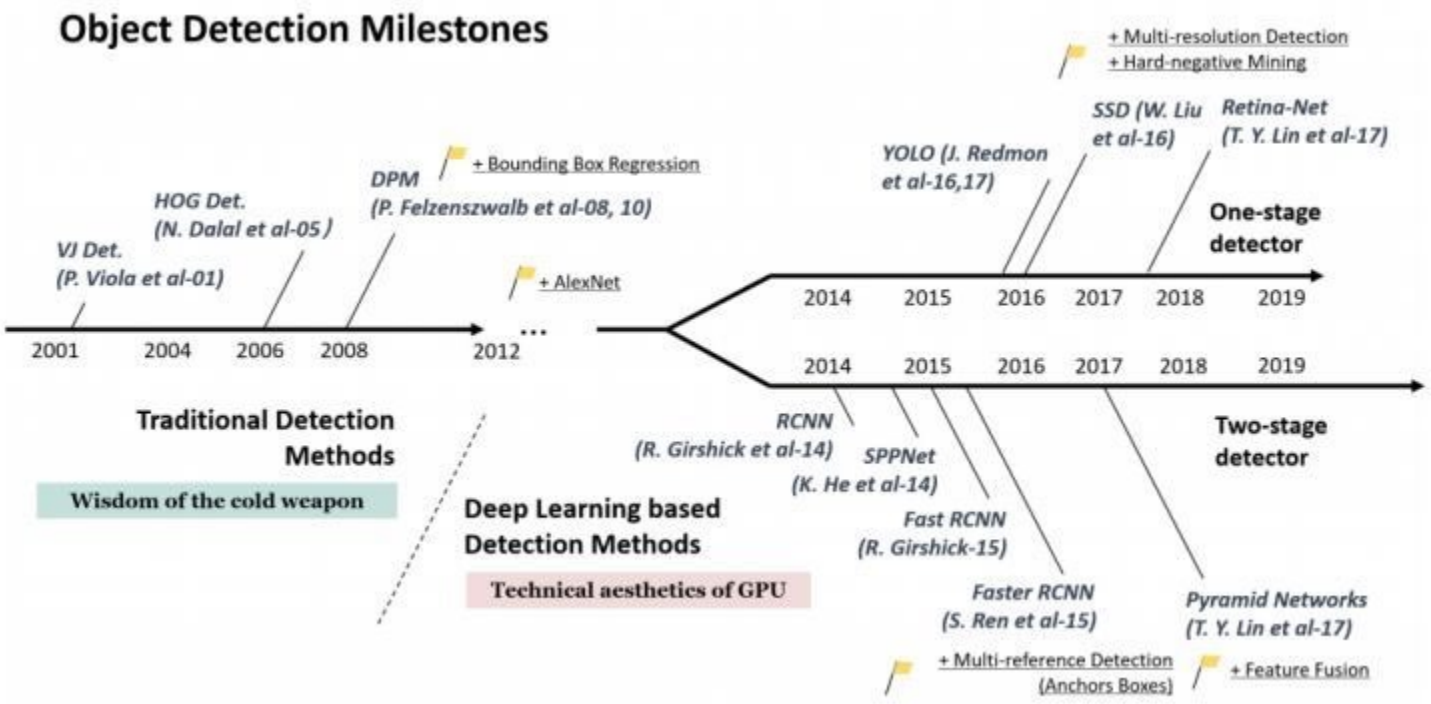
$$Loss = \sum_{i}^{N}(t_{*}^{i} - \hat{w}_{*}^{T}\phi_{5}(P^{i}))^{2}$$

函数的优化目标为：

$$W_{*} = argmin_{w_{*}}\sum_{i}^{N}(t_{*}^{i} - \hat{w}_{*}^{T}\phi_{5}(P^{i}))^{2} + \lambda||\hat{w}_{*}||^{2}$$

# 相关算法

## 目标检测算法分类：



# 相关算法

## 目标检测算法分类：

目标检测的发展可以划分为两个周期：

（1）传统目标检测算法（1998-2014）：传统目标检测算法主要基于手工提取特征，具体步骤可概括为：选取感兴趣区域->对可能包含物体的区域进行特征分类->对提取的特征进行检测。虽然传统目标检测算法经过了十余年的发展，但是其识别效果并没有较大改善且运算量大，此处不再详细介绍。

（2）基于深度学习的目标检测算法（2014-）：该方法主要分为Anchor-based的Two Stage和One Stage和Anchor-free两种思路。

# 相关算法

## 目标检测算法分类：

a）Two Stage：先预设一个区域，该区域称为Region Proposal，即一个可能包含待检测物体的预选框（简称RP），再通过卷积神经网络进行样本分类计算。

该算法的流程是：特征提取 -> 生成RP -> 分类/回归定位。

常见的Two Stage算法有：R-CNN、SPP-Net、Fast R-CNN、Faster R-CNN、R-FCN等，其特点是检测精度高，但是检测速度不如One Stage方法高。

# 相关算法

## 目标检测算法分类：

b）One Stage：直接在网络中提取特征值来分类目标和定位。

该算法的流程是：特征提取 -> 分类/回归定位。

常见的One Stage算法有：OverFeat、YOLOv1、YOLOv2、YOLOv3、YOLOv5、SSD、RetinaNet等，其特点是检测速度快，但是精度没有Two Stage方法高。

c）Anchor-free：该方法通过确定关键点的方式来完成检测，大大减少了网络超参数的数量。

# R-CNN论文讲解

**R-CNN在前人的肩膀上前行**

在过去的十多年时间里，传统的机器视觉领域，通常采用特征描述子来应对目标识别任务，这些特征描述子最常见的就是STFT和HOG。OpenCV里有现成的API可供大家实现相关的操作。

SIFT和HOG的王者地位被卷积神经网络撼动。

2012年Krizhevsky等人在ImageNet举办的ILSVRC目标识别挑战大赛中一战成名，豪夺当年的第一名，Top5错误率15%，而他们团队提出来的网络结构以第一作者Alex Krizhevsky命名，它就是AlexNet。

# R-CNN论文讲解

**R-CNN在前人的肩膀上前行**

因为AlexNet的出现，世人的目光重回神经网络领域，以此为契机，不断涌现出各种各样的网络比如：VGG，GoogleNet，ResNet等等。

2014年，对于目标检测领域是一个极具里程碑意义的年份。当年，Ross Girshick大神使用Region Proposal + CNN的方式代替传统目标检测研究中使用的滑动窗口+手工设计特征的方法，推出全新的R-CNN架构，正式掀起基于深度学习方法的目标检测研究的浪潮。

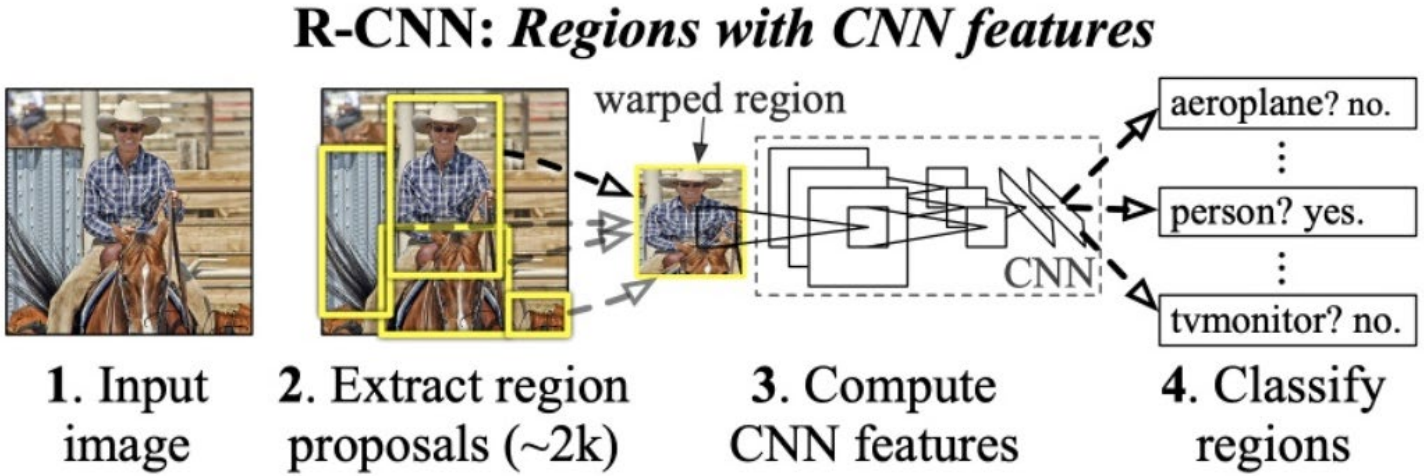
# R-CNN论文讲解

**R-CNN取得的贡献**

- 这篇论文证明了可以将神经网络应用在自低向上的候选区域，这样就可以进行目标分类和目标定位。传统的目标检测算法使用滑动窗口法依次判断穷举出的所有候选区域，而R-CNN则采用Selective Search算法预先提取一系列可能包含物体的候选区域，然后在候选区域上使用卷机神网络提取特征，大幅减少了计算量。
- 这篇论文也带来了一个观点，那就是当我们缺乏大量的标注数据时，比较好的手段是，进行神经网络的迁移学习，采用在其他大型数据集上训练过的神经网络，然后在小规模特定的数据集中进行微调。

# R-CNN论文讲解

R-CNN的核心流程



# R-CNN论文讲解

## R-CNN的核心流程

前面的内容提到，R-CNN系统分为3个阶段，反映到架构上由3个模块完成：

- 生成类别独立的候选区域，这些候选区域其中包含了R-CNN最终定位的结果。
- 神经网络去针对每个候选区域提取固定长度的特征向量。
- 一系列的SVM分类器

# R-CNN论文讲解

## R-CNN的核心流程

R-CNN利用候选区域与CNN结合来做目标定位。

具体是：

- 给定一张输入图片，从图片中提取2000个类别独立的候选区域；
- 对于每个区域利用CNN抽取一个固定长度的特征向量；
- 再对每个区域利用SVM进行目标分类；
- 使用回归器精细修正候选框位置。

# R-CNN论文讲解

## R-CNN的候选区域

能够生成候选区域的方法有很多，比如：

- Objectness
- Selective search
- Category-independent object proposals
- Constrained parametric min-cuts(CPMC)
- Multi-scale combinatorial grouping
- Ciresan

R-CNN采用的是Selective Search算法。

# R-CNN论文讲解

## R-CNN的候选区域

从一张图像生成约2000个候选区域。基本思路如下：

- 使用一种过分割手段，将图像分割成小区域；
- 查看现有小区域，合并可能性最高的两个区域。重复直到整张图像合并成一个区域位置；
- 输出所有曾经存在过的区域，所谓候选区域

候选区域生成和后续步骤相对独立，实际可以使用任意算法进行。

# R-CNN论文讲解

## R-CNN的候选区域

合并规则：

优先合并以下四种区域：

1.颜色（颜色直方图）相近的

2.纹理（梯度直方图）相近的

3.合并后总面积小的

4.合并后，总面积在其BBOX中所占比例大的

**其中第三条**:保证合并操作的尺度较为均匀，避免一个大区域陆续“吃掉”其他小区域。

**其中第四条:**保证合并后形状规则。

# R-CNN论文讲解

## R-CNN的特征提取

R-CNN抽取了一个4096维的特征向量，采用的是AlexNet，基于Caffe进行代码开发。

解下来简单介绍一下AlexNet结构：

AlexNet由输入层，5个卷积层，3个全连接层组成(其中最后一个全连接层也是softmax输出层）。网络是分布在2个GPU上的，部分层只跟同一个GPU中的上一层连接。其网络结构如下：

# R-CNN论文讲解

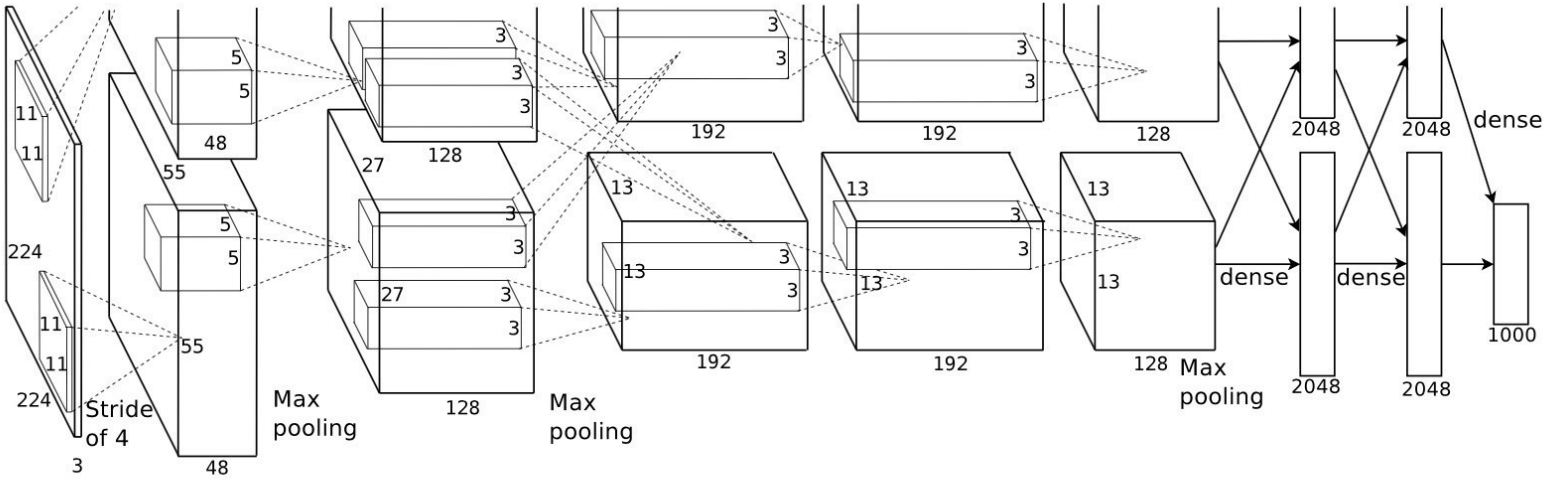


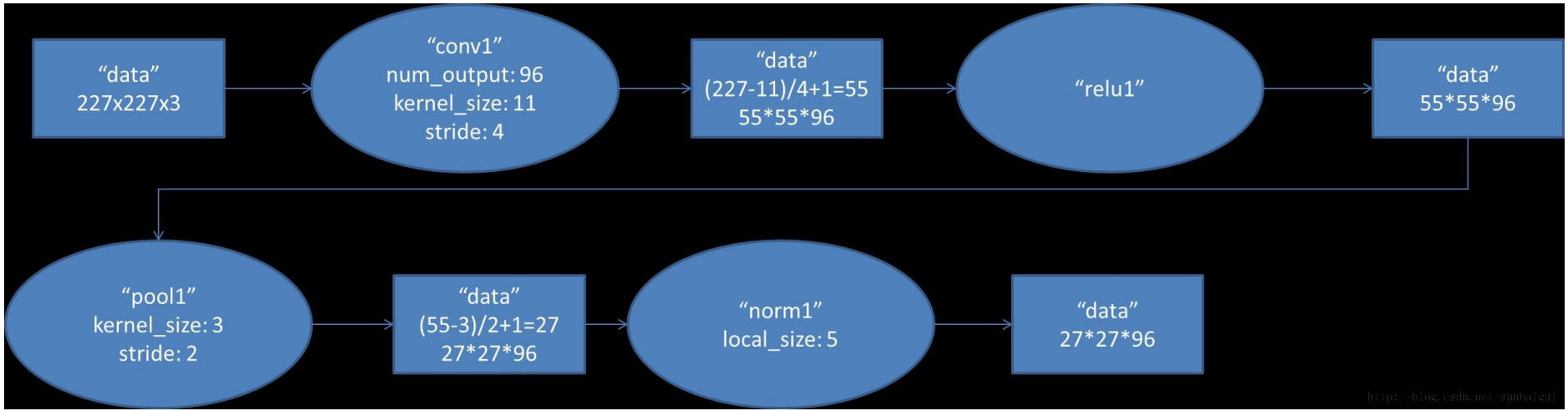

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

# R-CNN论文讲解

第一层输入数据为原始的227* 227* 3的图像（输入图像的尺寸是224* 224，在进行第一次卷积的时候会padding 3个像素变成227* 227），这个图像被11* 11* 3的卷积核进行卷积运算，卷积核对原始图像的每次卷积都生成一个新的像素。卷积核沿原始图像的x轴方向和y轴方向两个方向移动，移动的步长是4个像素。因此，卷积核在移动的过程中会生成(227-11)/4+1=55个像素(227个像素减去11，正好是54，即生成54个像素，再加上被减去的11也对应生成一个像素)，行和列的55*55个像素形成对原始图像卷积之后的像素层。共有96个卷积核，会生成55*55*96个卷积后的像素层，96个卷积核分成2组，每组48个卷积核。对应生成2组55* 55* 48的卷积后的像素层数据。这些像素层经过relu1单元的处理，生成激活像素层，尺寸仍为2组55* 55* 48的像素层数据。

这些像素层经过pool运算(池化运算)的处理，池化运算的尺度为3*3，运算的步长为2，则池化后图像的尺寸为(55-3)/2+1=27。即池化后像素的规模为27*27*96；然后经过归一化处理，归一化运算的尺度为5*5；第一卷积层运算结束后形成的像素层的规模为27*27*96。分别对应96个卷积核所运算形成。这96层像素层分为2组，每组48个像素层，每组在一个独立的GPU上进行运算。

# R-CNN论文讲解

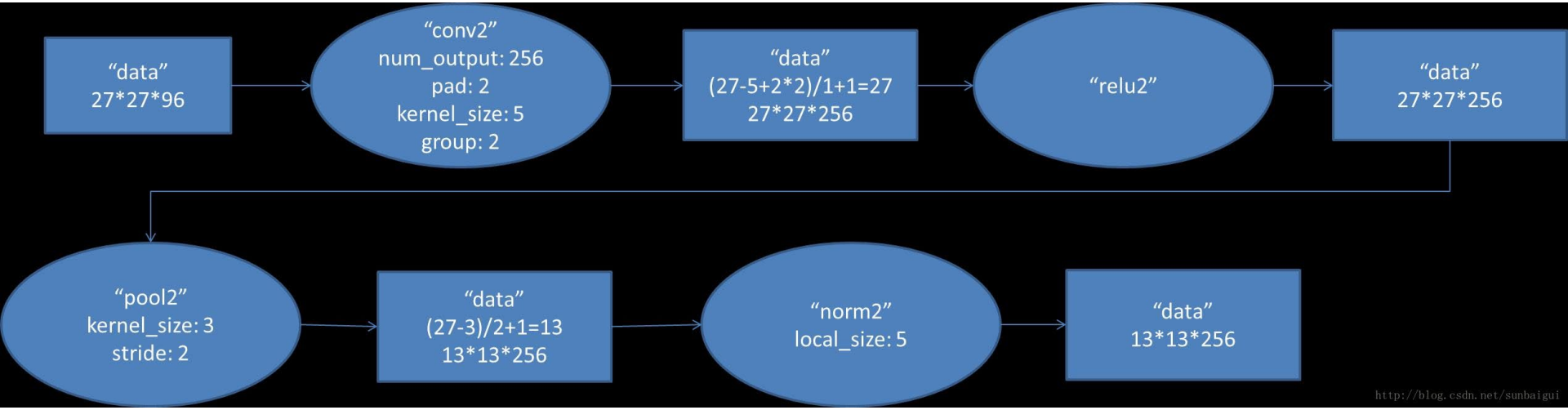

“data”
227x227x3
“conv1”
num_output: 96
kernel_size: 11
stride: 4
“data”
(227-11)/4+1=55
55*55*96
“relu1”
“data”
55*55*96
“pool1”
kernel_size: 3
stride: 2
“data”
(55-3)/2+1=27
27*27*96
“norm1”
local_size: 5
“data”
27*27*96

# R-CNN论文讲解

第二层输入数据为第一层输出的27*27*96的像素层，为便于后续处理，每幅像素层的左右两边和上下两边都要填充2个像素；27*27*96的像素数据分成27*27*48的两组像素数据，两组数据分别再两个不同的GPU中进行运算。每组像素数据被5*5*48的卷积核进行卷积运算，卷积核对每组数据的每次卷积都生成一个新的像素。卷积核沿原始图像的x轴方向和y轴方向两个方向移动，移动的步长是1个像素。因此，卷积核在移动的过程中会生成(27-5+2*2)/1+1=27个像素。(27个像素减去5，正好是22，在加上上下、左右各填充的2个像素，即生成26个像素，再加上被减去的5也对应生成一个像素)，行和列的27*27个像素形成对原始图像卷积之后的像素层。共有256个5*5*48卷积核；这256个卷积核分成两组，每组针对一个GPU中的27*27*48的像素进行卷积运算。会生成两组27*27*128个卷积后的像素层。这些像素层经过relu2单元的处理，生成激活像素层，尺寸仍为两组27*27*128的像素层。

这些像素层经过pool运算(池化运算)的处理，池化运算的尺度为3*3，运算的步长为2，则池化后图像的尺寸为(57-3)/2+1=13。 即池化后像素的规模为2组13*13*128的像素层；然后经过归一化处理，归一化运算的尺度为5*5；第二卷积层运算结束后形成的像素层的规模为2组13*13*128的像素层。分别对应2组128个卷积核所运算形成。每组在一个GPU上进行运算。即共256个卷积核，共2个GPU进行运算。

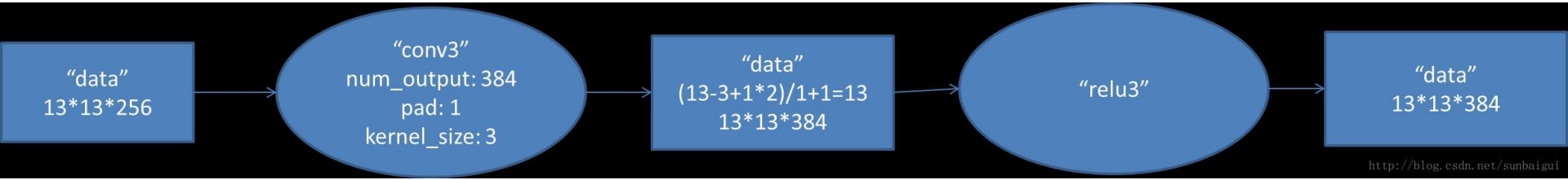
# R-CNN论文讲解


“data”
27*27*96
“conv2”
num_output: 256
pad: 2
kernel_size: 5
group: 2
“data”
(27-5+2*2)/1+1=27
27*27*256
“relu2”
“data”
27*27*256
“pool2”
kernel_size: 3
stride: 2
“data”
(27-3)/2+1=13
13*13*256
“norm2”
local_size: 5
“data”
13*13*256
http://blog.csdn.net/sunbaigui

# R-CNN论文讲解

第三层输入数据为第二层输出的2组13*13*128的像素层；为便于后续处理，每幅像素层的左右两边和上下两边都要填充1个像素；2组像素层数据都被送至2个不同的GPU中进行运算。每个GPU中都有192个卷积核，每个卷积核的尺寸是3*3*256。因此，每个GPU中的卷积核都能对2组13*13*128的像素层的所有数据进行卷积运算。卷积核对每组数据的每次卷积都生成一个新的像素。卷积核沿像素层数据的x轴方向和y轴方向两个方向移动，移动的步长是1个像素。因此，运算后的卷积核的尺寸为(13-3+1*2)/1+1=13（13个像素减去3，正好是10，在加上上下、左右各填充的1个像素，即生成12个像素，再加上被减去的3也对应生成一个像素），每个GPU中共13*13*192个卷积核。2个GPU中共13*13*384个卷积后的像素层。这些像素层经过relu3单元的处理，生成激活像素层，尺寸仍为2组13*13*192像素层，共13*13*384个像素层。

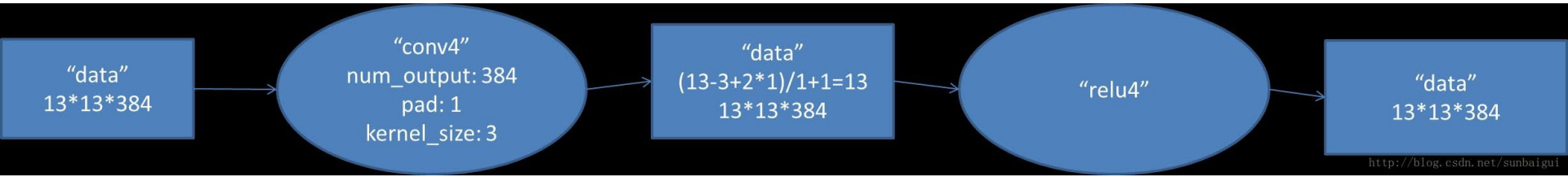
# R-CNN论文讲解


“data”
13*13*256
“conv3”
num_output: 384
pad: 1
kernel_size: 3
“data”
(13-3+1*2)/1+1=13
13*13*384
“relu3”
“data”
13*13*384
http://blog.csdn.net/sunbaigui

# R-CNN论文讲解

第四层输入数据为第三层输出的2组13*13*192的像素层；为便于后续处理，每幅像素层的左右两边和上下两边都要填充1个像素；2组像素层数据都被送至2个不同的GPU中进行运算。每个GPU中都有192个卷积核，每个卷积核的尺寸是3*3*192。因此，每个GPU中的卷积核能对1组13*13*192的像素层的数据进行卷积运算。卷积核对每组数据的每次卷积都生成一个新的像素。卷积核沿像素层数据的x轴方向和y轴方向两个方向移动，移动的步长是1个像素。因此，运算后的卷积核的尺寸为(13-3+1*2)/1+1=13（13个像素减去3，正好是10，在加上上下、左右各填充的1个像素，即生成12个像素，再加上被减去的3也对应生成一个像素），每个GPU中共13*13*192个卷积核。2个GPU中共13*13*384个卷积后的像素层。这些像素层经过relu4单元的处理，生成激活像素层，尺寸仍为2组13*13*192像素层，共13*13*384个像素层。

# R-CNN论文讲解

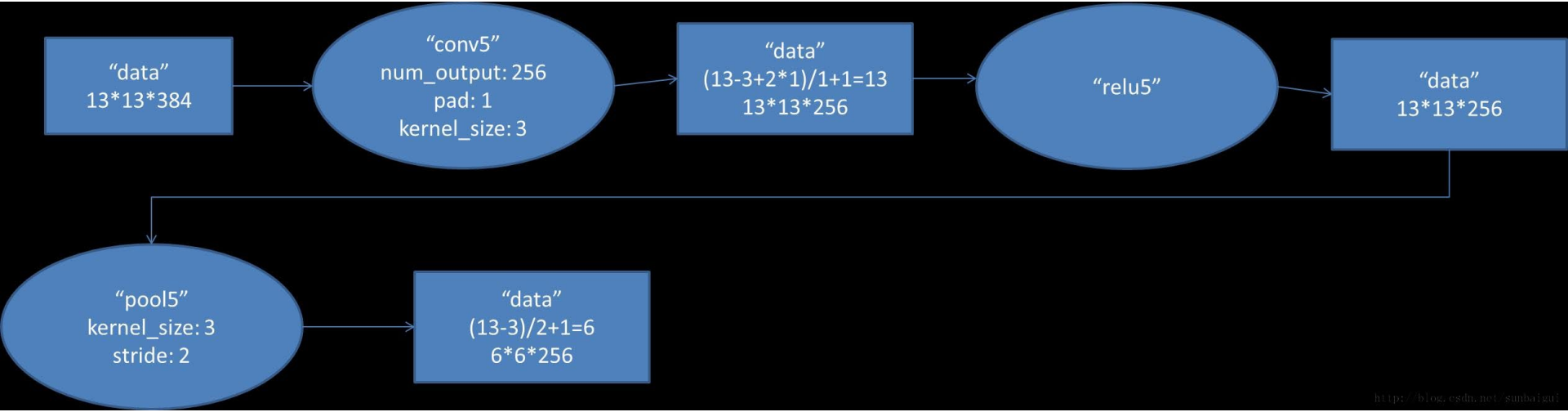

“data”
13*13*384
“conv4”
num_output: 384
pad: 1
kernel_size: 3
“data”
(13-3+2*1)/1+1=13
13*13*384
“relu4”
“data”
13*13*384
http://blog.csdn.net/sunbaigui

# R-CNN论文讲解

第五层输入数据为第四层输出的2组13*13*192的像素层；为便于后续处理，每幅像素层的左右两边和上下两边都要填充1个像素；2组像素层数据都被送至2个不同的GPU中进行运算。每个GPU中都有128个卷积核，每个卷积核的尺寸是3*3*192。因此，每个GPU中的卷积核能对1组13*13*192的像素层的数据进行卷积运算。卷积核对每组数据的每次卷积都生成一个新的像素。卷积核沿像素层数据的x轴方向和y轴方向两个方向移动，移动的步长是1个像素。因此，运算后的卷积核的尺寸为(13-3+1*2)/1+1=13（13个像素减去3，正好是10，在加上上下、左右各填充的1个像素，即生成12个像素，再加上被减去的3也对应生成一个像素），每个GPU中共13*13*128个卷积核。2个GPU中共13*13*256个卷积后的像素层。这些像素层经过relu5单元的处理，生成激活像素层，尺寸仍为2组13*13*128像素层，共13*13*256个像素层。

2组13*13*128像素层分别在2个不同GPU中进行池化(pool)运算处理。池化运算的尺度为3*3，运算的步长为2，则池化后图像的尺寸为(13-3)/2+1=6。 即池化后像素的规模为两组6*6*128的像素层数据，共6*6*256规模的像素层数据。

# R-CNN论文讲解


“data”
13*13*384
“conv5”
num_output: 256
pad: 1
kernel_size: 3
“data”
(13-3+2*1)/1+1=13
13*13*256
“relu5”
“data”
13*13*256
“pool5”
kernel_size: 3
stride: 2
“data”
(13-3)/2+1=6
6*6*256

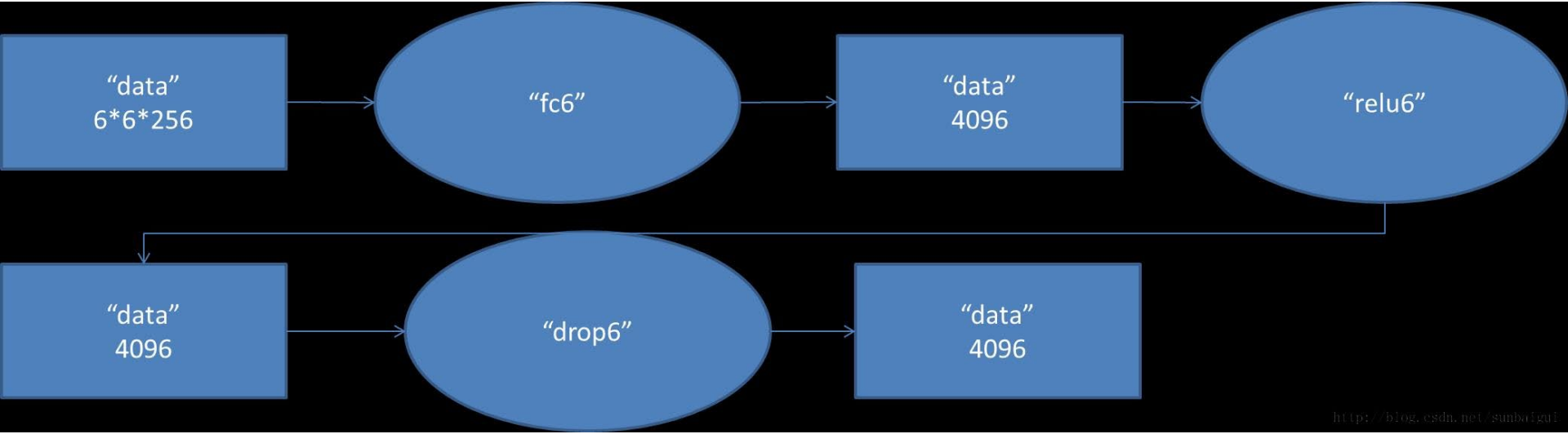
# R-CNN论文讲解

第六层输入数据的尺寸是6*6*256，采用6*6*256尺寸的滤波器对第六层的输入数据进行卷积运算；每个6*6*256尺寸的滤波器对第六层的输入数据进行卷积运算生成一个运算结果，通过一个神经元输出这个运算结果；共有4096个6*6*256尺寸的滤波器对输入数据进行卷积运算，通过4096个神经元输出运算结果；这4096个运算结果通过relu激活函数生成4096个值；并通过drop运算后输出4096个本层的输出结果值。

由于第六层的运算过程中，采用的滤波器的尺寸(6*6*256)与待处理的feature map的尺寸(6*6*256)相同，即滤波器中的每个系数只与feature map中的一个像素值相乘；而其它卷积层中，每个滤波器的系数都会与多个feature map中像素值相乘；因此，将第六层称为全连接层。

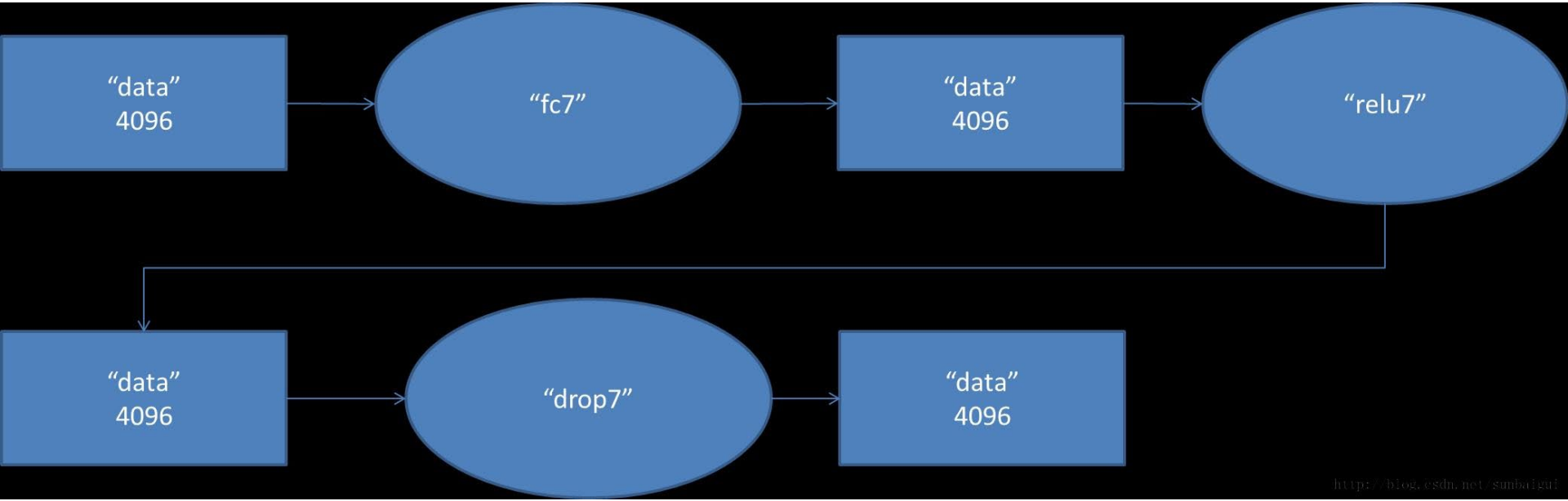
第五层输出的6*6*256规模的像素层数据与第六层的4096个神经元进行全连接，然后经由relu6进行处理后生成4096个数据，再经过dropout6处理后输出4096个数据。

# R-CNN论文讲解



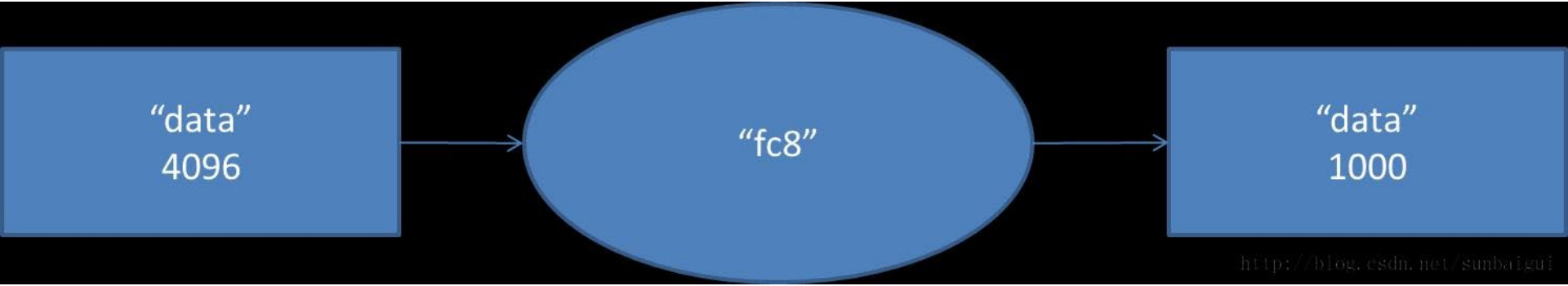
# R-CNN论文讲解

第六层输出的4096个数据与第七层的4096个神经元进行全连接，然后经由relu7进行处理后生成4096个数据，再经过dropout7处理后输出4096个数据。第六层输出的4096个数据与第七层的4096个神经元进行全连接，然后经由relu7进行处理后生成4096个数据，再经过dropout7处理后输出4096个数据。



# R-CNN论文讲解

第七层输出的4096个数据与第八层的1000个神经元进行全连接，经过训练后输出被训练的数值。第七层输出的4096个数据与第八层的1000个神经元进行全连接，经过训练后输出被训练的数值。



# R-CNN论文讲解

# R-CNN论文讲解

回归我们学习的R-CNN论文。

需要注意的是AlexNet的输入图像大小是227*227。

而我们通过Selective Search产生的候选区域大小不一，为了兼容，R-CNN采取了非常暴力的手段，那就是无视候选区域的大小和形状，统一变换到相同的尺寸：227*227。

在对region进行变换的时候，首先对这些区域进行扩充处理，在其框周围添加了p个像素，也就是人为添加了边框，在这里p=16。

论文中，作者如此解释：

Of the many possible transformations of our arbitrary-shaped regions, we opt for the simplest.

# R-CNN论文讲解

**测试阶段的目标检测：**

在测试阶段，R-CNN在每张图片上抽取近2000个候选区域。

然后讲每个候选区域进行尺寸的修改变换，送进神经网络以读取特征，然后用SVM进行类别的识别，并产生计数。

候选区域有2000个，所以会有很多的重叠部分。针对每个类，通过计算IoU指标，采取非极大值抑制，以最高分的区域为基础，剔除掉那些重叠位置的区域。

# R-CNN论文讲解

**目标检测运行时的分析：**

作者提出，两个因素可以使得目标识别变得高效：

- CNN的参数是所有类别共享的
- R-CNN生成的特征向量参数维度较少，论文拿应用在UVA采用的空间金字塔技术相比，它们生成的维度是360K，而R-CNN是4K多。参数变少了，所以比传统的高效。

# R-CNN论文讲解

## 训练：

前面的论文中已经提到使用R-CNN采用迁移学习。

提取在ILSVRC2012的模型和权重，然后在VOC上进行fine-tune。

需要注意的是，这里在ImageNet上训练的是模型识别物体类型的能力，而不是预测bbox位置的能力。

ImageNet的训练中需要预测1000个类别，而R-CNN在VOC上进行迁移学习的时候，神经网络只需要识别21个类别。（VOC规定的20个类别+背景）

R-CNN将候选框与Ground Truth中的box标签相比较，如果IoU>0.5，说明两个对象重叠的位置比较多，于是就可以认为这个候选框是Positive，否则就是Negetive。

训练策略是：采用SGD训练，初始学习率为0.001，mini-batch大小为128。

# R-CNN论文讲解

## 训练中的识别相关介绍：

对待一个二分类器，它的结果分为正和负。

举个论文中的例子：有一个汽车分类器，一个方框中包含了汽车，就是正，否则负。

这里有一个比较难确认的问题，如果一个方框，只有一部分与汽车重叠，那么该如何标记?

论文作者引用了IoU的阈值，文中取0.3，如果一个区域与Ground Truth的IoU值低于设定的阈值，则标记为负。

注意的是，阈值不是胡乱取的，来自{0，0.1，0.2，0.3，0.4，0.5}的数值组合的。

# R-CNN论文讲解

## 训练中的识别相关介绍：

阈值的选取非常重要。

引用论文中的实验：如果阈值选取为0.5，mAP指标直接下降5个点。如果取值为0，则下降4个点。

特征抽取成功后，R-CNN会用SVM去识别每个区域的类别，但这里需要优化。

因为训练的数据量实在太大，不可能一次性填充到电脑内存中。

解决办法就是Hard negative mining。

# R-CNN论文讲解

## R-CNN在PASCAL VOC2010-2012上的表现：

R-CNN是在PASCAL VOC2012上进行最终的微调，也就是在VOC2012的训练集上优化SVM。

然后，还与当时4个强劲的目标检测算法进行了比较。

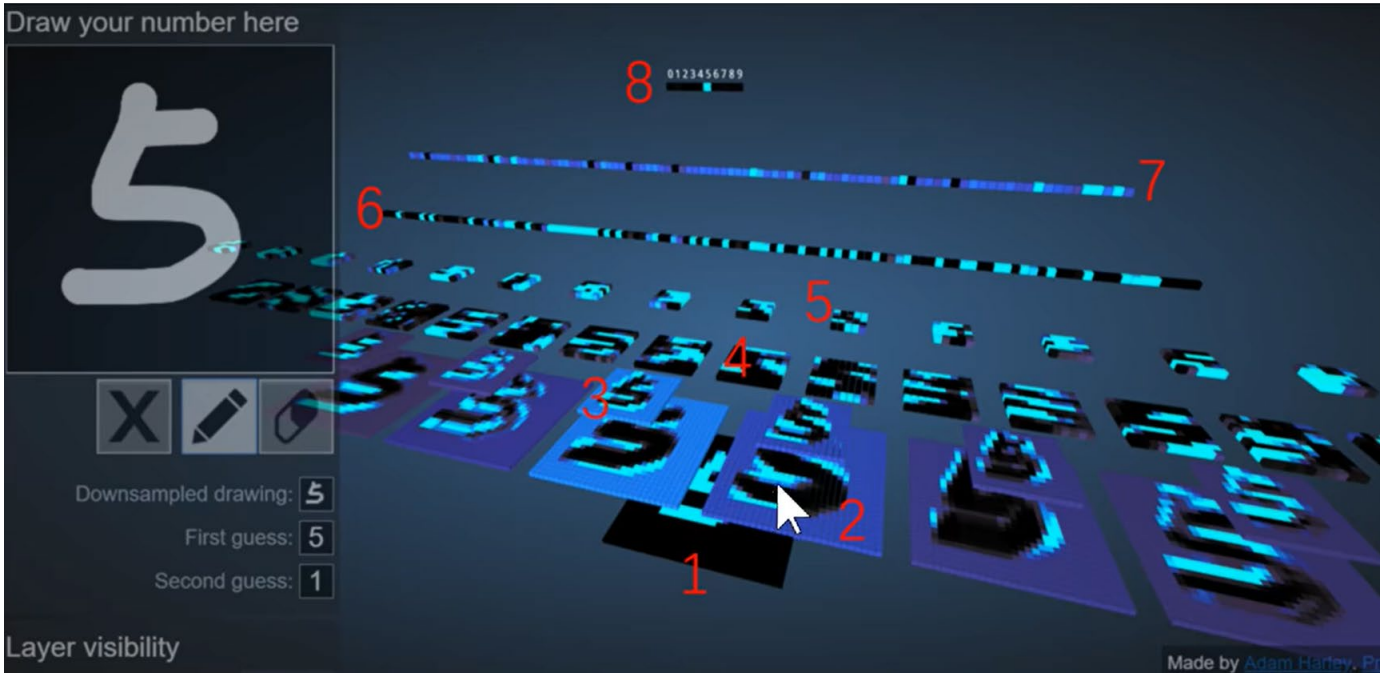
| VOC 2010 test | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DPM v5 [17]† | 49.2 | 53.8 | 13.1 | 15.3 | 35.5 | 53.4 | 49.7 | 27.0 | 17.2 | 28.8 | 14.7 | 17.8 | 46.4 | 51.2 | 47.7 | 10.8 | 34.2 | 20.7 | 43.8 | 38.3 | 33.4 |
| UVA [32] | 56.2 | 42.4 | 15.3 | 12.6 | 21.8 | 49.3 | 36.8 | 46.1 | 12.9 | 32.1 | 30.0 | 36.5 | 43.5 | 52.9 | 32.9 | 15.3 | 41.1 | 31.8 | 47.0 | 44.8 | 35.1 |
| Regionlets [35] | 65.0 | 48.9 | 25.9 | 24.6 | 24.5 | 56.1 | 54.5 | 51.2 | 17.0 | 28.9 | 30.2 | 35.8 | 40.2 | 55.7 | 43.5 | 14.3 | 43.9 | 32.6 | 54.0 | 45.9 | 39.7 |
| SegDPM [15]† | 61.4 | 53.4 | 25.6 | 25.2 | 35.5 | 51.7 | 50.6 | 50.8 | 19.3 | 33.8 | 26.8 | 40.4 | 48.3 | 54.4 | 47.1 | 14.8 | 38.7 | 35.0 | 52.8 | 43.1 | 40.4 |
| R-CNN | 67.1 | 64.1 | 46.7 | 32.0 | 30.5 | 56.4 | 57.2 | 65.9 | 27.0 | 47.3 | 40.9 | 66.6 | 57.8 | 65.9 | 53.6 | 26.7 | 56.5 | 38.1 | 52.8 | 50.2 | 50.2 |
| R-CNN BB | **71.8** | **65.8** | **53.0** | **36.8** | **35.9** | **59.7** | **60.0** | **69.9** | **27.9** | **50.6** | **41.4** | **70.0** | **62.0** | **69.0** | **58.1** | **29.5** | **59.4** | **39.3** | **61.2** | **52.4** | **53.7** |

值得注意的是，UVA也采用了相同的候选框选取算法，但是很明显R-CNN的表现更显著。

# R-CNN论文讲解

## 可视化：

在CNN里面，第一层可以直接显示，肉眼都可以分辨，主要捕捉的是物体的边缘、颜色等底层特征。但是越往后卷积层越抽象，这个时候进行可视化就是一个巨大的挑战。



# R-CNN论文讲解

## 可视化：

Ziler和Fergus提出了一种基于反卷积手段的可视化研究，但论文作者提出了一个简单直接的方法，没有参数。

思路如下：

挑选一个特征出来，把它当成一个物体分类器，然后计算它们处理不同候选框时，activation的值，这个值文中解释为：特征对这块区域的响应情况。然后将activation的值排名，作为分数，取前几位，然后显示这些候选框。

由此，可以讲明白，这个特征大概是什么。

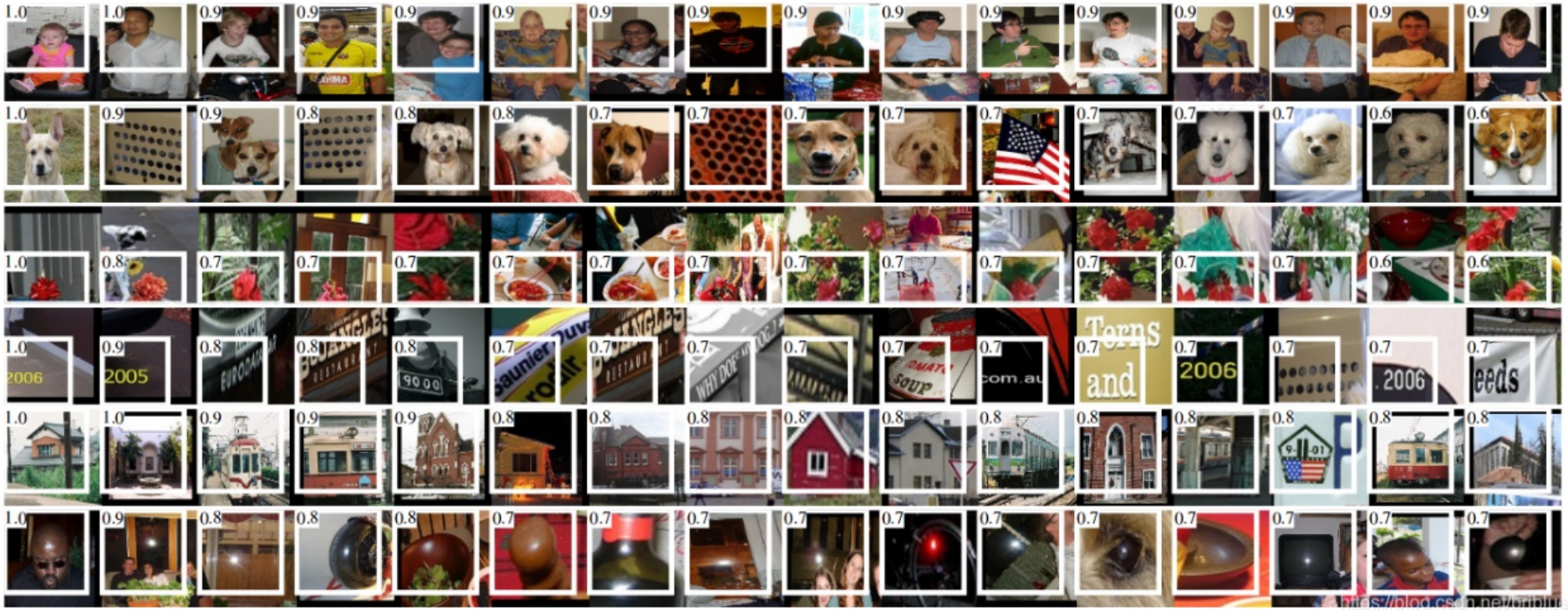
# R-CNN论文讲解

## 可视化：

作者将pool5作为可视化的对象，它的feature map是6*6*256的规格，可以理解为有256个小方块，每个方块对应一个特征。

下面的图像展示了可视化的效果，这里只显示了256个特征中的6个，每个特征取activation值最高的16个区域。

# R-CNN论文讲解


1.0 1.0 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
1.0 0.9 0.9 0.8 0.8 0.8 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.6 0.6
1.0 0.8 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.6 0.6
1.0 0.9 0.8 0.8 0.8 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7
1.0 1.0 0.9 0.9 0.9 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8
1.0 0.9 0.8 0.8 0.8 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7

# R-CNN论文讲解

上图应该很直观了，对于同一类特征，activation的值相差不大，这也是CNN能够准确识别物体的直观体现。

# R-CNN论文讲解

## 框架简单：

AlexNet有七层，那么那些层是关键指标呢？是否有可以删除的层?

pool5在可视化里面已经讨论过了，那么fc6和fc7就成了研究的对象。

fc6与pool5构成全连接，为了计算特征会乘以一个4096*9216的权重矩阵，然后再与一组偏置单元相加，所以有3700多万的参数。

fc7是最后一层，它的权重矩阵是4096*409，参数有1678万。

# R-CNN论文讲解

## 框架简单：

但经过作者在PASCAL上不做微调，直接测试，可以发现fc7的意义没有fc6大，甚至移除它之后，对于mAP的结果没有影响。而移除fc7就表示可以减少将近1800万个参数。

作者进一步探究，更惊喜的是，同时移除fc6和fc7并没有多大的损失，甚至结果还要好上一点点。

综上所述：CNN最神奇的地方来自卷积层，不是全连接层。

# R-CNN论文讲解

## 框架简单：

如果经过微调会怎么样？

结果证明：微调后fc6和fc7提升的效果明显。

# R-CNN论文讲解

## 框架简单：

综上所述：

pool5从ImageNet训练集中学习到了物体识别的泛化能力，而能力的提升则是通过特定领域的微调。

举个例子，神经网络在ImageNet数据集中学习到了100种猫的特征，而我们自己的数据集有两种猫，经过微调训练后，这个神经网络能更准确的识别这两种猫。

# R-CNN论文讲解

## 目标检测错误分析：

论文作者采用了Hoiem提出的目标检测分析工具，能够直观的揭露出错误的模型，论文通过这个工具针对性的进行微调。

## 边界框回归：

Bbox的值就是物理方框的位置，预测它是回归问题。

受DPM算法启发，作者训练了一个线性的回归模型，这个模型能够针对区域中的pool5数据预测一个新的box位置。

# R-CNN论文讲解

## 语义分割：

候选框分类是语义分割的标准做法，所以R-CNN也可以做语义分割。作者拿它和O2P来比较。

R-CNN进行语义分割分为3个阶段：

- 利用CPMC生成候选框，然后将其调整为227*227，送到神经网络中，这是full阶段，区域中有前景也有背景；
- 这个阶段只处理候选框的前景，将背景用输入的平均值代替，然后背景就变成了0，这个阶段称为fg；
- full+fg阶段，将背景和前景简单的拼接。

# R-CNN论文讲解

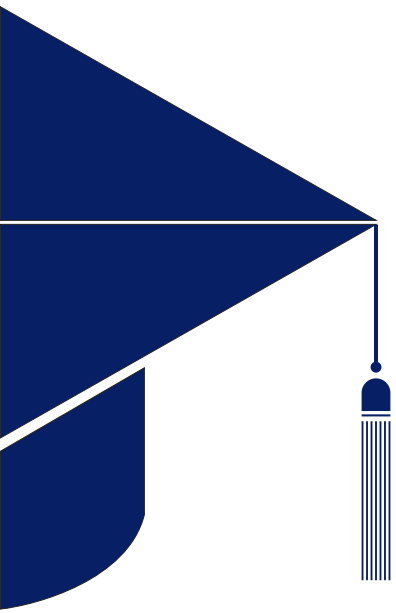
**总结全文：**

- R-CNN采用AlexNet
- R-CNN采用Selective Search技术生成候选框
- R-CNN在ImageNet上先进行预训练，然后利用成熟的权重参数在PASCAL VOC数据集上进行微调
- R-CNN用CNN抽取特征，然后用一系列的SVM做类别预测
- R-CNN的bbox位置回归基于DPM的算法，自己训练了一个线性回归模型
- R-CNN的语义分割采用CPMC生成Region

# R-CNN论文讲解

**总结全文：**

在2022年的今天，R-CNN早已经不是最先进的目标检测模型，亦不是最先进的语义分割模型，但这篇论文最大的意义在于展示了作者如何在资源匮乏的情况下整合现有的先进技术去解决问题。



# 演示完毕 请多指点