# Faster R-CNN: 使用候选区域生成网络向实时物体检测努力

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

## 1 摘要

最新的物体检测网络依赖候选区域生成算法来假定物体位置。例如 SPPnet[12] 和 Fast R-CNN[11] 的先进网络已经减少了检测网络的运行时间，暴露出候选区域生成是瓶颈。在这个工作中，我们提出了与检测网络分享全图卷积特征的候选区域生成网络 (RPN)，因此使得候选区域生成几乎毫无代价。RPN 是在每个位置同时预测物体边界和置信度的全卷积网络。RPN 通过端到端的方式训练来生成高质量的候选区域，这将会被 Fast R-CNN 用来检测。更进一步地，通过共享卷积特征，我们将 RPN 和 Fast R-CNN 合并为一个网络——使用最近流行的关于神经网络的术语 “注意力” 机制，RPN 组件告诉整个网络应该看向何处。对于非常深的 VGG-16[7] 模型，在每张图片使用 300 个候选区域的情况下，我们的检测系统在单块 GPU 上达到了 5fps 的速度，同时在 PASCAL VOC 2017 和 COCO 数据集上达到了最好的效果。在 ILSVRC 和 COCO2015 竞赛中，Faster R-CNN 和 RPN 是多个赛道第一名的基础。代码已经开源。

## 2 简介

近来，候选区域生成方法和基于区域的卷积神经网络的成功驱动了物体检测的进步（例如，[4]）。虽然在最初开发的 [6] 中基于区域的 CNN 是计算昂贵的，但是由于候选区域间的卷积共享，使得它们的开销被显著降低。最近的版本，Fast R-CNN[11] 使用非常深的网络，在忽略候选区域生成的情况下，几乎实现了实时速率。如今，候选区域生成是最优检测系统的测试时间计算瓶颈。

候选区域生成方法通常依赖于不昂贵的特征以及经济的推理策略。选择搜索 [4]，其中一种最流行的方法，基于设计好的低层次特征贪心地合并超像素。然而与高效的检测网络 [11] 相比，选择搜索比之慢一个数量级，在 CPU 实现中处理每个图片需要 2 秒。EdgeBoxes[10] 目前提供了候选区域质量和速度间的最好平衡，处理每张图片的速度达到了 0.2 秒。然而，候选区域生成步骤仍然和检测网络花费了相同的运行时间。

人们可能会注意到 fast R-CNN 借用了 GPU 的优势，然而候选区域生成方法是在 CPU 上实现的，这使得运行时间的比较并不公平。加速候选计算的一个显然方法便是为了 GPU 重新实现它。这可能是一个有效的工程解决方法，但是重新实现忽视了下游的检测网络并因此忽视了共享计算的重要机会。

在这片文章中，我们展示了一种算法方面的改变，使用深度卷积神经网络来计算候选区域，这引领了一种优雅且有效的解决方式——。给了检测网络的计算，候选计算几乎是免费的。作为结尾，我们提出了一个新的可以和最好的物体检测网络 [11, 6] 共享卷基层的候选区域生成网络 (RPN)。通过在测试时共享卷积，计算候选区域的额外开销很小（例如，每张图 10 毫秒）。

我们的发现是基于区域的检测器，例如 Fast R-CNN，使用的卷积特征图，也可以被用来生成候选区域。在这些卷积特征的上面，我们通过额外加入一些在常规网格的每个位置同时回归区域边界和置信度的卷积层的方式，构建了一个 RPN。RPN 因此是一种全卷积网络 (FCN)[13] 并可以针对生成检测候选区域任务端到端训练。

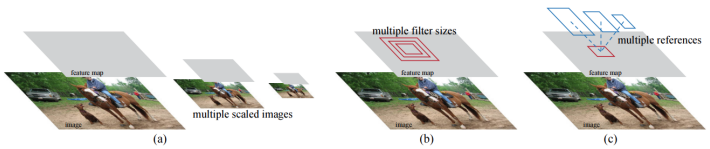


图 1: 解决多尺度和大小的不同方案。(a) 构建图片和特征图金字塔，并在所有尺度运行分类器。(b) 在特征图上运行多尺度/大小的过滤器。(c) 我们在回归函数中使用参考框金字塔。

RPN 被设计来高效地在广大的尺度范围以及高宽比范围预测候选区域。与之前使用图片金字塔 (2, a) 或过滤器金字塔 (2, b) 的流行方法 [3, 1, 6, 11] 相比，我们提出了全新的“锚”框来作为多尺度以及多高宽比的参照 (2, c)。我们的方案可以被视为回归参照金字塔，这避免了枚举多个尺度和

高宽比的图片或过滤器。当使用单一尺度图片进行训练和测试时，这个模型性能良好，因此对运行速度有益。

为了统一 RPN 和 Fast R-CNN[11] 物体检测网络，我们提出了一种候选区域生成任务微调和物体检测微调（保持候选固定）交替进行的训练方案。这个方案可以快速收敛并产生一个可以使得两个任务共享卷积特征的统一网络。

我们在 PASCAL VOC 检测基准上全面地评估了我们的方法，RPN 的 Fast R-CNN 方法得到的检测准确率好于选择搜索的 Fast R-CNN。同时，我们的方法几乎消除了测试阶段选择搜索带来的所有计算负担——候选区域生成运行的有效时间仅仅是 10 毫秒。在使用昂贵的很深的模型 [7] 的情况下，我们的检测方法依旧在单块 GPU 上达到了 5fps 的速率（包含所有步骤），因此这是就速度和准确率而言的实际物体检测系统。我们也汇报了在 MS COCO 数据集上的结果并探讨了使用 MS COCO 的数据提高在 PASCAL VOC 上的表现。数据已经在https://github.com/shaoqingren/faster_rcnn（MATLAB 版本）和https://github.com/rbgirshick/py-faster-rcnn（Python 版本）开源。

这个手稿的早期版本在之前已经发表。在那之后，RPN 和 Faster R-CNN 框架被采用并推广到其他方法，例如三维物体检测 [17]，基于部分的检测 [14]，实例分割 [15] 和图片字幕。我们的快速并有效的物体检测系统也在汇报了用户参与度改进的情况下，在例如 Pinterests 的商业系统中被构建。

在 ILSVRC 和 COCO 2015 竞赛中，Faster R-CNN 和 RPN 是在 ImageNet 检测，ImageNet 定位，COCO 检测和 COCO 分割的多个赛道中多个第一名条目的基础。RPN 王权从数据中学习生成候选区域，因此可以轻易从更深以及更有表达性的特征（例如在 [16] 中采用的 101 层残差网络）中受益。Faster R-CNN 和 RPN 也在那些竞赛中的领先条目中被使用。这些结果表明我们的方法不仅是对实际用法开销高效的方法，同时也是一种提升物体检测准确率的有效方式。

# 3 FASTER R-CNN

我们的物体检测系统，被称为 Faster R-CNN，由两个模块组成。第一个模块是一个生成候选区域的深层全卷积网络，第二个模块是使用候选区

域的 Fast R-CNN 检测器 [11]。整个系统是一个物体检测的单一统一网络（图3）。使用最近流行的神经网络术语“注意力”机制，RPN 模块告诉 Fast R-CNN 模块应该看向哪里。在 3.1 节中我们介绍了候选区域生成网络的设计和性质。在 3.2 节中我们开发了在特征共享的情况下训练两个模块的算法。

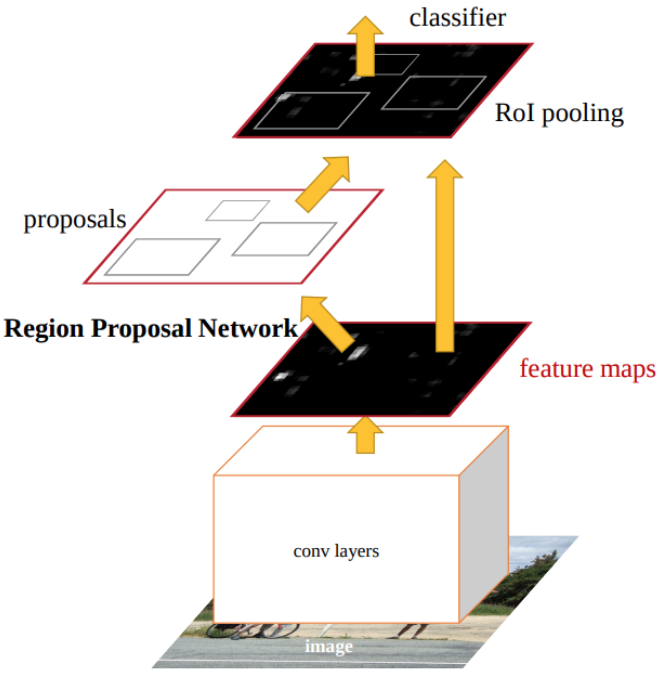


图 2: Faster R-CNN 是一个物体检测的单一统一网络。RPN 模块作为这个统一网络的“注意力”。

### 3.1 候选区域生成网络

候选区域生成网络（RPN）使用一张图片（任意大小）作为输入并输出一个矩形物体候选的集合，每个候选都有一个置信度。我们使用全卷积网络 [13] 为这个过程建模，这也是我们将在这一章描述的内容。由于我们最终的目标是和 Fast R-CNN 物体检测网络 [11] 共享计算，因此我们假设两个网络共享一些共同的卷积层。在我们的实验中，我们探讨了有 5 层共享卷积层的 Zeiler 和 Fergus 模型 [9]（ZF）和有 13 层共享卷积层的 Simonyan 和

Zisserman 模型 [7]（VGG-16）。

为了生成候选区域，我们我们在有最后的共享卷积层输出的卷积特征图上滑动一个小网络。这个小网络使用一个输入卷积特征图的 $n \times n$ 滑动窗口作为输入。每个滑动窗口被映射至一个更低维度的特征（ZF 是 256 维，VGG 是 512 维，并通过 ReLU[2]）。这个特征被送入两个孪生全连接层——一个框回归层（*reg*）和一个框分类层（*cls*）。我们在这片文章中使用 $n = 3$，注意输入图片的有效接受域是很大的（ZF 和 VGG 分别是 171 和 228 像素）。这个迷你网络在图3.1的左边被单独阐述。注意由于迷你网络通过滑动窗口的方式操作，所以全连接网络被所有空间位置共享。这种架构自然地被通过 $n \times n$ 卷积层紧接两个 $1 \times 1$ 卷积层实现（分别为了 *reg* 和 *cls*）。

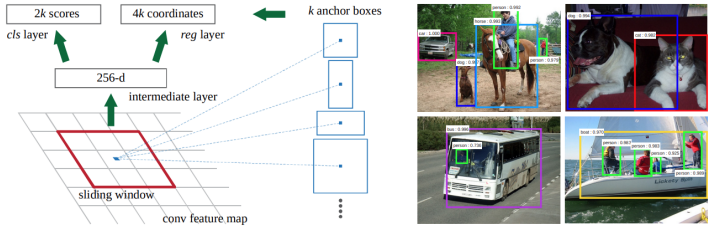


图 3: **左**：候选区域生成网络（RPN）。**右**：在 PASCAL VOC 2007 test 上使用 RPN 候选区域的检测例子。我们的方法在大范围的尺度和高宽比上进行了检测。

### 3.1.1 锚

在每一个滑动窗口位置，我们同时预测多个候选区域，其中每一个位置的最大可能候选区域数量记为 $k$。所以 *reg* 层有 $4k$ 个输出来编码 $k$ 个框的坐标，*cls* 层输出 $2k$ 个值来为每个候选估计是否存在物体的概率。$k$ 个候选区域相对于 $k$ 个参考框被参数化，我们将之称为锚。锚的中心在滑动窗口的某个位置，并与尺寸和高宽比关联（图3.1左）。我们默认使用 3 个尺寸和 3 个高宽比，在每个滑动位置得到 $k = 9$ 个锚。对于一个尺寸为 $W \times H$（典型大致为 2400）的卷积特征图，共有 $WHk$ 个锚。

**具有转变不变形的锚** 我们的方法的一个重要性质就是它的转变不变性，就锚和计算关于锚的候选的函数来说都是如此。如果对图片中的一个物体进行转变，那么候选也应该转变，同时同样的函数应该可以在一个位置预测候

选区域。这个转变不变性性质由我们的方法保证。与之对比的是，MultiBox 方法 [5, 8] 使用 k-means 来生成 800 个锚，这不是转变不变性的。所以当一个物体转变时，MultiBox 并不能保证生成同样的候选区域。

转变不变性也减少了模型大小。MultiBox 有一个 $(4+1) \times 800$ 维的全连接输出层，然而我们的方法在 $k = 9$ 时仅有一个 $(4+2) \times 9$ 维的卷积输出层。作为结果，我们的输出层有 $2.8 \times 10^4$ 个参数 (对于 VGG-16 来说，$512 \times (4+2) \times 9$)，比有 $6.1 \times 10^6$（对于 MultiBox 中的 GoogleNet，$1536 \times (4+1) \times 800$）个参数的 MultiBox 的输出层小两个数量级。如果考虑特征映射层，我们的候选区域生成层参数数量仍然比 MultiBox 的参数数量小一个数量级。我们期待我们的模型在小数据集上，例如 PASCAL VOC，过拟合的风险更小。

**作为回归参考的多尺度锚** 我们对于锚的设计展示了一种创新的解决多尺度（和高宽比）问题的方案。如图2所示，存在两种多尺度预测的方式。第一种方式基于图片/特征金字塔，例如，DPM[1] 和基于 CNN 的方法 [3, 6, 11]。图片被重塑为多个尺寸，并未每个尺寸计算特征图（HOG[1] 或者深度卷积特征 [3, 6, 11]）（图2(a)）。这种方式通常是有效的，但是耗费时间。第二种方式是使用多个尺寸（和/或高宽比）的滑动窗口。例如，在 DPM[1] 中，不同高宽比的模型使用不同的过滤器尺寸分别进行训练（例如 $5 \times 7$ 和 $7 \times 5$）。如果使用这种方式来攫夺多尺度问题，它可以被想象为“过滤器金字塔”（图2(b)）。第二种方式通常与第一种方式结合使用。

作为比较，我们的基于锚的方式是在锚金字塔上构建的，同时也是最花费高效的。我们的方法使用多个尺寸和高宽比的锚作为参考来分类并回归边界框。它仅依赖于单一尺寸的图片和特征图，并使用单一尺寸的过滤器（特征图上的滑动窗口）。我们通过实验展示了这种方案对于解决多尺度和多大小的作用。

由于这个多尺度设计基于锚，所以我们可以简单地使用在单一尺度图片上计算得到的卷积特征，正如 Fast R-CNN[11] 中所做的。多尺度锚的设计是在无额外开销的情况下共享特征来解决尺度问题的关键组成成分。

### 3.1.2 损失函数

为了训练 RPN，我们为每一个锚分配了一个二进制类别标签（是否是一个物体）。我们为两种锚分配正标签：(i) 与 ground-truth 框有最大 IoU

的锚（们）或 (ii) 与任意 ground-truth 框的 IoU 高于 0.7 的锚。注意一个 ground-truth 框可能会为多个锚分配正标签。通常第二种情况对于决定正样本来说已经足够了；但是在某些罕见情况下，第二个条件可能会找不到正样本，所以我们任然采用了第一个条件。如果锚与任意 ground-truth 框的 IoU 都低于 0.3，我们为其分配负标签。既不是正例也不是负例的锚不会对训练目标做出贡献。

有了这些定义，我们最小化一个遵循 Fast R-CNN[11] 中的多任务损失的目标函数。我们对于单张图片的损失函数定义为：

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i L_{reg}(t_i, t_i^*) \tag{1}$$

这里，$i$ 是迷你批中的一个锚的索引，$p_i$ 是预测的锚 $i$ 是一个物体的概率。当锚是正例是 ground-truth 的标签为 1，如果锚是负例则为 0。$t_i$ 表示预测边界框的 4 个参数化坐标，$t_i^*$ 则是与一个正例锚相关联的 ground-truth 框。分类损失 $L_{cls}$ 是两个类别（是否有物体）的对数损失。对于回归损失，我们使用 $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$，其中 $R$ 是 [11] 中定义的鲁棒损失函数（平滑 $L_1$）。$p_i^* L_{reg}$ 项表示回归损失仅在正例锚（$p_I^* = 1$）作用，而在其他情况不作用。*cls* 和 *reg* 层的输出分别组成了 $\{p_i\}$ 和 $\{t_i\}$。

对于边界框回归，我们遵循 [6]，采用了如下所示的四个坐标的参数化：

$$\begin{aligned} t_x &= (x - x_a)/w_a, \quad & t_y &= (y - y_a)/h_a, \\ t_w &= \log(w/w_a), & t_h &= \log(h/h_a) \\ t_x^* &= (x^* - x_a)/w_a, & t_y^* &= (y^* - y_a)/h_a, \\ t_w^* &= \log(w^*/w_a), & t_h^* &= \log(h^*/h_a), \end{aligned} \tag{2}$$

其中 $x, y, w$ 和 $h$ 表示框的中心坐标以及它的宽度和高度。变量 $x, x_a$ 和 $x^*$ 分别表示预测框，锚框和 ground-truth 框（$y, w, h$ 类似）。这可以被想象为从锚框到附近 ground-truth 框的边界框回归。

不仅如此，我们的方法通过不同于以往基于 ROI（Region of Interest，感兴趣区域）方法的方式 [12, 11] 来实现边界框回归。在 [12, 11] 中，边界框回归是在从任意尺寸 ROI 池化得到的特征上进行的，同时所有区域尺寸共享回归的权重。在我们的提法中，用于回归的特征在特征图上都有相同的空间大小（$3 \times 3$）。为了考虑变化的尺寸，会习得 $k$ 个边界框回归器。每个回归器为一个尺寸和高宽比负责，同时 $k$ 个回归器不会共享权重。这样一

来，即使特征是固定尺寸，感谢锚的设计，我们仍然有可能预测不同尺寸的框。

#### 3.1.3 训练 RPN

我们可以通过反向传播和随机梯度下降（SGD）训练 RPN。我们遵循 [11] 中的“以图片为中心”的采样策略来训练网络。每个来自单一图片的迷你批包含多个正例和负例锚。虽然优化考虑所有锚的损失函数是可能的，但是由于负例样本占绝大多数，这会向负例样本偏移。取而代之的是，我们在一张图中随机采样 256 个锚来计算迷你批的损失函数，其中正例和负例的锚的比例最多为 1 : 1。如果图片中的正例样本比 128 少，我们将会使用负例来填充。

我们使用来自均值为 0、标准差为 0.01 的高斯分布的权重来初始化所有的新层。正如标准做法 [6]，所有其他的层（也就是，共享的卷积层）通过在 ImageNet 分类任务上训练得到的模型初始化。我们调整 ZF 网络的所有层和 VGG 网络 conv3_1 及其后续层来节省内存 [11]。在 PASCAL VOC 数据集上，我们对前 60k 个迷你批使用 0.001 的学习率，接下来 20k 个迷你批则使用 0.0001 的学习率。我们使用 0.9 作为动量以及 0.0005 的权重衰减 [**alexnet**]。我们使用 Caffe 实现。

### 3.2 RPN 和 Fast R-CNN 的共享特征

到此为止我们已经描述了如何训练生成候选区域的网络，但是还没有考虑将会使用这些候选区域的基于区域的物体检测 CNN。对于检测网络，我们采用 Fast R-CNN[11]。接下来我们描述学习一个由共享卷积层的 RPN 和 Fast R-CNN 组成的统一网络（图3）的算法。

独立训练的 RPN 和 Fast R-CNN 都会通过不同的方式修改它们的卷积层。因此我们需要开发一种允许在两个网络间共享卷积层而不是学习两个分开网络的技术。我们讨论三种训练共享特征网络的方法：

(i) *依次训练*。在这种方案中，我们先训练 RPN，并使用候选区域训练 Fast R-CNN。被 Fast R-CNN 调整的网络接着被用来初始化 RPN，接着进行迭代这个过程。这是在论文所有实验中使用的方法。

(ii) *近似联合学习*。在这种方案中，如图3所示，RPN 和 Fast R-CNN 网络将会在训练中合并如一个网络。在每一个 SGD 迭代中，前向传播会生成在训练 Fast R-CNN 检测器被视为固定的、事先计算好的候选区域的候

选区域。反向传播照常进行，其中共享层的传播信号将来自 RPN 和 Fast R-CNN 的损失结合。这种方案实现起来是简单的。但是这种方案忽视了作为网络响应的候选区域坐标的衍生物，所以是近似的。在我们的实验中，我们经验性地发现这种解决方案产生近似的结果，然而与依次训练相比，减少了 $25-50\%$ 的训练时间。我们发布的 Python 代码中包含了这种解决方案。

(iii) 非近似联合学习。正如上面讨论的，RPN 预测的边界框也是函数的输入。Fast R-CNN 中的 ROI 池化层接受卷积特征以及预测的边界框作为输入，所以一个理论可行的反向传播解决方案应该也考虑到框坐标的梯度。这些梯度在上述的近似联合学习中被忽视。在非近似联合学习中，我们需要一个对框坐标可导的 RoI 池化层。这并不是一个轻松的问题，可以通过 [**15**] 提出的 "ROI 包裹" 层给出一种解决方案，但是这超出了本文的讨论范围。

**4 步依次训练**。在这片论文中，我们采用了使用的 4 步训练算法来通过依次优化习得共享特征。在第一步中，我们如 3.1.3 节中所描述的那样训练一个 RPN。这个网络使用 ImageNet 预训练模型初始化并在候选区域生成任务上微调。在第二步中，我们使用第一步得到的 RPN 生成的候选区域训练一个单独的 Fast R-CNN 检测器。这个检测网络也通过 ImageNet 预训练模型初始化。在这个节点，两个网络并没有共享卷积层。在第三步，我们使用检测网络来初始化 RPN 训练，但是我们固定共享的卷积层并只微调 RPN 特有的层。现在两个网络共享一些卷积层了。最终，保持共享的卷积层固定，我们微调 Fast R-CNN 的特有卷积层。这样一来，两个网络共享相同的卷积层并形成了一个统一网络。类似的依次训练可以重复多次，但是我们发现收效甚微。

### 3.3 实现细节

# References

[1] Pedro F Felzenszwalb et al. "Object detection with discriminatively trained part-based models". In: *IEEE transactions on pattern analysis and machine intelligence* 32.9 (2009), pp. 1627–1645.

[2] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Icml*. 2010.

[3] Pierre Sermanet et al. “Overfeat: Integrated recognition, localization and detection using convolutional networks”. In: *arXiv preprint arXiv:1312.6229* (2013).

[4] Jasper RR Uijlings et al. “Selective search for object recognition”. In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[5] Dumitru Erhan et al. “Scalable object detection using deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 2147–2154.

[6] Ross Girshick et al. “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.

[7] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[8] Christian Szegedy et al. “Scalable, high-quality object detection”. In: *arXiv preprint arXiv:1412.1441* (2014).

[9] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[10] C Lawrence Zitnick and Piotr Dollár. “Edge boxes: Locating object proposals from edges”. In: *European conference on computer vision*. Springer. 2014, pp. 391–405.

[11] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[12] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[14] Jun Zhu, Xianjie Chen, and Alan L Yuille. "DeePM: A deep part-based model for object detection and semantic part localization". In: *arXiv preprint arXiv:1511.07131* (2015).

[15] Jifeng Dai, Kaiming He, and Jian Sun. "Instance-aware semantic segmentation via multi-task network cascades". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3150–3158.

[16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[17] Shuran Song and Jianxiong Xiao. "Deep sliding shapes for amodal 3d object detection in rgb-d images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 808–816.