卷积神经网络是一种强大的视觉模型,他可以产生分层的特征.本文表明了以端到端的方式训练的逐像素预测的卷积神经网络超过了以往的语义分割的SOTA模型.本文的核心就是构建了全卷积网络,全卷积网络以任意尺寸的图像作为输入,并且在经过推断和学习之后,全卷积网络会产生相同尺寸的输出.本文详细描述了全卷积网络的结构,说明了其与先前网络的关系,解释了全卷积网络的应用以及在空间密集预测任务上的应用.我们把同时期的卷积分类网络:AlexNet,VGGNet,GoogLeNet转换为全卷积网络,并且通过fine-tuning将分类网络学习到的表示迁移到了语义分割任务上.此外,我们提出了skip结构,他结合了来自于来自于浅层的密集的视觉外形也正和来自深层的稀疏的语义信息,最终产生了准确而又详尽的语义分割结果.我们的卷积网络在PASCAL VOC2012挑战赛上实现了SOTA的结果,比以往的SOTA方法有了20%的相对提升,IoU达到了62.2%.此外,在NYUDv2和SIFT-Flow数据集上都获得了SOTA的性能,而对于一个典型的图像推断时间只有不到0.2秒

Fully Convolutional Networks for Semantic Segmentation

CVPR2015,工作是2014年的语义分割奠基性的工作,虽然2012-1014年之前也有别的用神经网络做分割任务的,但是先前的方法效果都不是很好,FCN开创了新的时代,使得全卷积成为了范式

ResNet前的工作

感觉比较适合期刊文章
对自己的方法缺乏实验研究

文章结构有点像驳论文,先介绍了别人是怎么做分割的,然后diss他们的方法

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley
{jonlong,shelhamer,trevor}@cs.berkeley.edu

Abstract

Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, exceed the state-of-the-art in semantic segmentation. Our key insight is to build "fully convolutional" networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. We define and detail the space of fully convolutional networks, explain their application to spatially dense prediction tasks, and draw connections to prior models. We adapt contemporary classification networks (AlexNet [20], the VGG net [31], and GoogLeNet [32]) into fully convolutional networks and transfer their learned representations by fine-tuning [3] to the segmentation task. We then define a skip architecture that combines semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed segmentations. Our fully convolutional network achieves state-of-the-art segmentation of PASCAL VOC (20% relative improvement to 62.2% mean IU on 2012), NYUDv2, and SIFT Flow, while inference takes less than one fifth of a second for a typical image.

现有模型的全卷积版本可以输出任意尺寸输入的密集预测.全卷积网络的学习和推断都是通过密集前向传播和反向传播.全卷积网络中的上采样层使得模型可以实现逐像素的预测,并且加上下采样层,上采样层可以使得模型实现端到端的学习

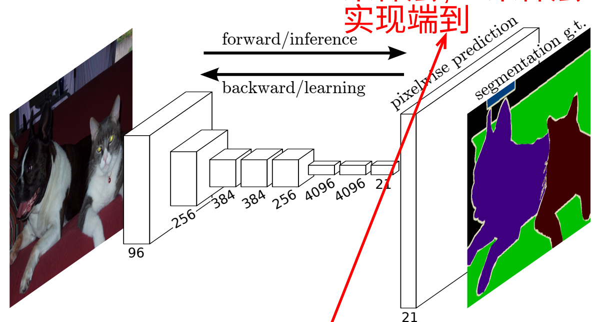


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

本文表明,使用端到端的逐像素预测的卷积网络超过以往的SOTA

卷积网络目前正在推动视觉识别领域的发展.他不仅推动了整张图像的分类任务的进步,也推动了具有固定结构输出的定位任务的进步,例如:物体检测,部分/关键点预测,定位匹配等等任务...

1. Introduction

Convolutional networks are driving advances in recognition. Convnets are not only improving for whole-image classification [20, 31, 32], but also making progress on local tasks with structured output. These include advances in bounding box object detection [29, 10, 17], part and keypoint prediction [39, 24], and local correspondence [24, 8].

The natural next step in the progression from coarse to fine inference is to make a prediction at every pixel. Prior approaches have used convnets for semantic segmentation [27, 2, 7, 28, 15, 13, 9], in which each pixel is labeled with the class of its enclosing object or region, but with shortcomings that this work addresses.

*Authors contributed equally

自然而然,卷积网络的下一步就是实现(从稀疏到)密集的预测,即对每个像素进行分类.而对每个像素进行分类的任务称为语义分割,即根据像素所在的物体获取区域对其赋予一个标签,而后我们的目标就是对每个像素准确的判断出他的类别.先前的一些工作已经在语义分割上有所成果,但是本文稍后会指出它们的问题.

We show that a fully convolutional network (FCN) trained end-to-end, pixels-to-pixels on semantic segmentation exceeds the state-of-the-art without further machinery. To our knowledge, this is the first work to train FCNs end-to-end (1) for pixelwise prediction and (2) from supervised pre-training. Fully convolutional versions of existing networks predict dense outputs from arbitrary-sized inputs. Both learning and inference are performed whole-image-at-a-time by dense feedforward computation and backpropagation. In-network upsampling layers enable pixelwise prediction and learning in nets with subsampled pooling.

以往的SOTA据我们所知,本文是第一个使用端到端的全卷积网络实现了:1.逐像素的预测,2.利用了有监督预训练学习到的知识.

This method is efficient, both asymptotically and absolutely, and precludes the need for the complications in other works. Patchwise training is common [27, 2, 7, 28, 9], but lacks the efficiency of fully convolutional training. Our approach does not make use of pre- and post-processing complications, including superpixels [7, 15], proposals [15, 13], or post-hoc refinement by random fields or local classifiers [7, 15]. Our model transfers recent success in classification [20, 31, 32] to dense prediction by reinterpreting classification nets as fully convolutional and fine-tuning from their learned representations. In contrast, previous works have applied small convnets without supervised pre-training [7, 28, 27].

本文提出的方法非常高效,不管是从渐进还是绝对分析,因此避免了其他方法带来的复杂性.我们的模型没有使用诸如超像素,提议等技术.我们的模型通过微调,将最近在分类任务上获得成功的分类方法学习到的知识迁移到了密集预测任务上去.

Semantic segmentation faces an inherent tension between semantics and location: global information resolves what while local information resolves where. Deep feature hierarchies encode location and semantics in a nonlinear

语义分割任务本质上面临着语义和位置的平衡:全局信息解决了物体是什么,而局部信息解决了物体在那里

卷积网络能够以深度分层的,非线性的方式构建位置-全局的特征金字塔.我们定义了跳跃结构,很好了利用了这一系列分层的特征,他可以结合深层的稀疏语义信息和浅层的精细的外形信息

local-to-global pyramid. We define a skip architecture to take advantage of this feature spectrum that combines deep, coarse, semantic information and shallow, fine, appearance information in Section 4.2 (see Figure 3).

在接下来的章节中,我们回顾了最近使用深度网络来处理分类任务的文章,全卷积网络以及处理语义分割任务的深度网络.此外,我们介绍了FCN的结构,以及在密集预测上的权衡.介绍了网络内的上采样和多层结合方法.最后介绍了测试的设置,以及我们在PASCAL VOC 2011-2,NYUDv2和SIFT Flow数据集上的表现

In the next section, we review related work on deep classification nets, FCNs, and recent approaches to semantic segmentation using convnets. The following sections explain FCN design and dense prediction tradeoffs, introduce our architecture with in-network upsampling and multi-layer combinations, and describe our experimental framework. Finally, we demonstrate state-of-the-art results on PASCAL VOC 2011-2, NYUDv2, and SIFT Flow.

2. Related work

Our approach draws on recent successes of deep nets for image classification [20, 31, 32] and transfer learning [3, 38]. Transfer was first demonstrated on various visual recognition tasks [3, 38], then on detection, and on both instance and semantic segmentation in hybrid proposal-classifier models [10, 15, 13]. We now re-architect and fine-tune classification nets to direct, dense prediction of semantic segmentation. We chart the space of FCNs and situate prior models, both historical and recent, in this framework.

这些都是远古时期的文章了翻译也没啥用,看一遍就行

Fully convolutional networks To our knowledge, the idea of extending a convnet to arbitrary-sized inputs first appeared in Matan *et al.* [26], which extended the classic LeNet [21] to recognize strings of digits. Because their net was limited to one-dimensional input strings, Matan *et al.* used Viterbi decoding to obtain their outputs. Wolf and Platt [37] expand convnet outputs to 2-dimensional maps of detection scores for the four corners of postal address blocks. Both of these historical works do inference and learning fully convolutionally for detection. Ning *et al.* [27] define a convnet for coarse multiclass segmentation of *C. elegans* tissues with fully convolutional inference.

现代的卷积网络也对全卷积网络,有Sermanet等人做的Sliding Window检测,Pinheiro等人做的语义分割,Tompson等人做的姿态估计

Fully convolutional computation has also been exploited in the present era of many-layered nets. Sliding window detection by Sermanet *et al.* [29], semantic segmentation by Pinheiro and Collobert [28], and image restoration by Eigen *et al.* [4] do fully convolutional inference. Fully convolutional training is rare, but used effectively by Tompson *et al.* [35] to learn an end-to-end part detector and spatial model for pose estimation, although they do not exposit on or analyze this method.

Alternatively, He *et al.* [17] discard the non-convolutional portion of classification nets to make a feature extractor. They combine proposals and spatial pyramid pooling to yield a localized, fixed-length feature for classification. While fast and effective, this hybrid model cannot be learned end-to-end.

Dense prediction with convnets Several recent works have applied convnets to dense prediction problems, including semantic segmentation by Ning *et al.* [27], Farabet *et al.* [7], and Pinheiro and Collobert [28]; boundary prediction for electron microscopy by Ciresan *et al.* [2] and for natural images by a hybrid convnet/nearest neighbor model by Ganin and Lempitsky [9]; and image restoration and depth estimation by Eigen *et al.* [4, 5]. Common elements of these approaches include

最近有一些工作开始使用卷积网络处理密集预测问题,例如Ning等人,Farabet等人,Pinherio等人使用卷积网络处理语义分割问题,Ciersan等人使用卷积网络处理医学影像边界预测问题,Ganin等人针对自然图像,Eigen等人利用卷积网络处理图像恢复问题

为了得到密集输出,这些针对密集预测问题的卷积网络中使用到的方法包括:1.使用小模型限制模型容量和感受野,2.逐块训练,3.后处理,包括:超分辨率投影,随机场正则化,滤波,以及局部分类,4.输入偏移和输出集成,5.多尺度特征金字塔处理,6.饱和tanh非线性,7.集成学习等方法

- small models restricting capacity and receptive fields;
- patchwise training [27, 2, 7, 28, 9];
- post-processing by superpixel projection, random field regularization, filtering, or local classification [7, 2, 9];
- input shifting and output interlacing for dense output [29, 28, 9];
- multi-scale pyramid processing [7, 28, 9];
- saturating tanh nonlinearities [7, 4, 28]; and
- ensembles [2, 9],

虽然上面的这些技巧本文提出的方法统统都没有用,但是本文还是在3.2,3.3,3.4节讨论了上面的这些方法

whereas our method does without this machinery. However, we do study patchwise training 3.4 and "shift-and-stitch" dense output 3.2 from the perspective of FCNs. We also discuss in-network upsampling 3.3, of which the fully connected prediction by Eigen *et al*. [5] is a special case.

与现有的方法不同,本文利用了迁移学习,讲分类网络视为有监督预训练,而后在全卷积魔性的基础上进行fine-tune来学习整个图像的分割

Unlike these existing methods, we adapt and extend deep classification architectures, using image classification as supervised pre-training, and fine-tune fully convolutionally to learn simply and efficiently from whole image inputs and whole image ground thruths.

Hariharan等人和Gupta等人同样也是利用深度分类网络来实现语义分割,但是不同的是,他们是利用了混合proposal-classifier的模型,更具体的来说,他们的方法fine-tune了R-CNN来实现语义分割.因为他们的方法在PASCALVOC2012上得到了SOTA的效果,我们的模型将会和他们进行比较

Hariharan *et al.* [15] and Gupta *et al.* [13] likewise adapt deep classification nets to semantic segmentation, but do so in hybrid proposal-classifier models. These approaches fine-tune an R-CNN system [10] by sampling bounding boxes and/or region proposals for detection, semantic segmentation, and instance segmentation. Neither method is learned end-to-end. They achieve state-of-the-art segmentation results on PASCAL VOC and NYUDv2 respectively, so we directly compare our standalone, end-to-end FCN to their semantic segmentation results in Section 5.

We fuse features across layers to define a nonlinear local-to-global representation that we tune end-to-end. In contemporary work Hariharan *et al.* [16] also use multiple layers in their hybrid model for semantic segmentation.

3. Fully convolutional networks

Each layer of data in a convnet is a three-dimensional array of size $h \times w \times d$, where h and w are spatial dimensions, and d is the feature or channel dimension. The first layer is the image, with pixel size $h \times w$, and d color channels. Locations in higher layers correspond to the locations in the image they are path-connected to, which are called their *receptive fields*.

Convnets are built on translation invariance. Their basic components (convolution, pooling, and activation functions) operate on local input regions, and depend only on *relative* spatial coordinates. Writing $\mathbf{x}_{ij}$ for the data vector at location (i, j) in a particular layer, and $\mathbf{y}_{ij}$ for the follow-

下面这一段讲了不同的全卷积网络结构(其实我个人觉得这段更应该称为如何利用网络获得密集输出),首先回顾了分类网络,然后讲本文的方法通过重新理解分类层来获得密集输出,接下来讨论了其他的获得密集输出的方法,包括:Shift-stitch是滤波器稀疏化,上采样是反向步长卷积,逐块学习是损失采样,类似于驳论文

这段回顾了卷积网络，给出了本文的Notation

x_ij表示了位置(i,j)处的像素，y_ij表示下一层对应位置的像素，k是卷积核大小，f表示了不同的操作

ing layer, these functions compute outputs $\mathbf{y}_{ij}$ by

$$\mathbf{y}_{ij} = f_{ks}\left(\{\mathbf{x}_{si+\delta i, sj+\delta j}\}_{0\le \delta i, \delta j \le k}\right)$$

where k is called the kernel size, s is the stride or subsampling factor, and f_{ks} determines the layer type: a matrix multiplication for convolution or average pooling, a spatial max for max pooling, or an elementwise nonlinearity for an activation function, and so on for other types of layers.

This functional form is maintained under composition, with kernel size and stride obeying the transformation rule

$$f_{ks} \circ g_{k's'} = (f \circ g)_{k'+(k-1)s', ss'}.$$

While a general deep net computes a general nonlinear function, a net with only layers of this form computes a nonlinear *filter*, which we call a *deep filter* or *fully convolutional network*. An FCN naturally operates on an input of any size, and produces an output of corresponding (possibly resampled) spatial dimensions.

注意，对于全卷积网络来说，损失是每一个位置上的损失的和，梯度也是一样

A real-valued loss function composed with an FCN defines a task. If the loss function is a sum over the spatial dimensions of the final layer, $\ell(\mathbf{x};\theta) = \sum_{ij} \ell'(\mathbf{x}_{ij};\theta)$, its gradient will be a sum over the gradients of each of its spatial components. Thus stochastic gradient descent on ℓ computed on whole images will be the same as stochastic gradient descent on ℓ', taking all of the final layer receptive fields as a minibatch.

When these receptive fields overlap significantly, both feedforward computation *and* backpropagation are much more efficient when computed layer-by-layer over an entire image instead of independently patch-by-patch.

接下来我们解释了如何把分类网络转变为全卷积网络来得到输出的图像。对于逐像素的预测，我们需要把分类网络的稀疏预测重新对应到图像上去

We next explain how to convert classification nets into fully convolutional nets that produce coarse output maps. For pixelwise prediction, we need to connect these coarse outputs back to the pixels. Section 3.2 describes a trick, fast scanning [11], introduced for this purpose. We gain insight into this trick by reinterpreting it as an equivalent network modification. As an efficient, effective alternative, we introduce deconvolution layers for upsampling in Section 3.3. In Section 3.4 we consider training by patchwise sampling, and give evidence in Section 4.3 that our whole image training is faster and equally effective.

这段讲解了这篇文章最核心的部分：重新理解分类器

3.1. Adapting classifiers for dense prediction

Typical recognition nets, including LeNet [21], AlexNet [20], and its deeper successors [31, 32], ostensibly take fixed-sized inputs and produce non-spatial outputs. The fully connected layers of these nets have fixed dimensions and throw away spatial coordinates. However, these fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions. Doing so casts them into fully convolutional networks that take input of any size and output classification maps. This transformation is illustrated in Figure 2.

传统的卷积网络，包括LeNet，AlexNet和GoogLeNet，VGGNet这些模型，都需要以固定尺寸图像作为输入，而后得到非空间的输出。这些网络中的线性层需要固定尺寸的输入，然后丢弃掉空间坐标信息。然而，最后的第一个线性层可以视为是一个卷积核等于最后feature map大小的卷积层，因此从这个角度理解，就可以得到一个全卷积网络

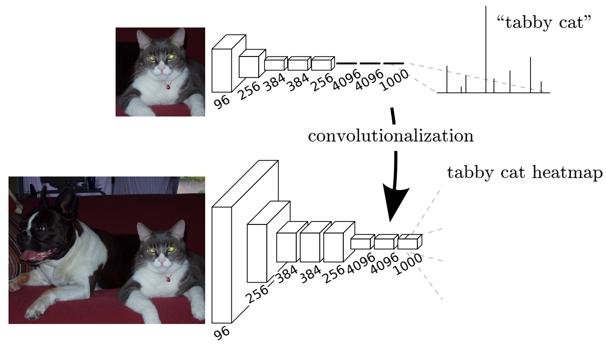


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Furthermore, while the resulting maps are equivalent to the evaluation of the original net on particular input patches, the computation is highly amortized over the overlapping regions of those patches. For example, while AlexNet takes 1.2 ms (on a typical GPU) to infer the classification scores of a 227×227 image, the fully convolutional net takes 22 ms to produce a 10×10 grid of outputs from a 500×500 image, which is more than 5 times faster than the naïve approach[1].

The spatial output maps of these convolutionalized models make them a natural choice for dense problems like semantic segmentation. With ground truth available at every output cell, both the forward and backward passes are straightforward, and both take advantage of the inherent computational efficiency (and aggressive optimization) of convolution. The corresponding backward times for the AlexNet example are 2.4 ms for a single image and 37 ms for a fully convolutional 10×10 output map, resulting in a speedup similar to that of the forward pass.

While our reinterpretation of classification nets as fully convolutional yields output maps for inputs of any size, the output dimensions are typically reduced by subsampling. The classification nets subsample to keep filters small and computational requirements reasonable. This coarsens the output of a fully convolutional version of these nets, reducing it from the size of the input by a factor equal to the pixel stride of the receptive fields of the output units.

需要注意的是，我们重新理解分类网络最终得到的全卷积网络最后得到的feature map的尺寸被极大地减少了，按照pool的尺寸和步长指数级减少，以维持卷积核比较小，从而减少参数量和计算量

3.2. Shift-and-stitch is filter rarefaction

Dense predictions can be obtained from coarse outputs by stitching together output from shifted versions of the input. If the output is downsampled by a factor of f, shift the input x pixels to the right and y pixels down, once for every (x, y) s.t. $0 \le x, y < f$. Process each of these f^2 inputs, and interlace the outputs so that the predictions correspond to the pixels at the *centers* of their receptive fields.

这段没必要看，已经是不用的方法了

[1] Assuming efficient batching of single image inputs. The classification scores for a single image by itself take 5.4 ms to produce, which is nearly 25 times slower than the fully convolutional version.

Although performing this transformation naïvely increases the cost by a factor of f^2, there is a well-known trick for efficiently producing identical results [11, 29] known to the wavelet community as the à trous algorithm [25]. Consider a layer (convolution or pooling) with input stride s, and a subsequent convolution layer with filter weights f_{ij} (eliding the irrelevant feature dimensions). Setting the lower layer's input stride to 1 upsamples its output by a factor of s. However, convolving the original filter with the upsampled output does not produce the same result as shift-and-stitch, because the original filter only sees a reduced portion of its (now upsampled) input. To reproduce the trick, rarefy the filter by enlarging it as

$$f'_{ij} = \begin{cases} f_{i/s, j/s} & \text{if } s \text{ divides both } i \text{ and } j; \\ 0 & \text{otherwise,} \end{cases}$$

(with i and j zero-based). Reproducing the full net output of the trick involves repeating this filter enlargement layer-by-layer until all subsampling is removed. (In practice, this can be done efficiently by processing subsampled versions of the upsampled input.)

Decreasing subsampling within a net is a tradeoff: the filters see finer information, but have smaller receptive fields and take longer to compute. The shift-and-stitch trick is another kind of tradeoff: the output is denser without decreasing the receptive field sizes of the filters, but the filters are prohibited from accessing information at a finer scale than their original design.

Although we have done preliminary experiments with this trick, we do not use it in our model. We find learning through upsampling, as described in the next section, to be more effective and efficient, especially when combined with the skip layer fusion described later on.

这段也没有必要看了,就是转置卷积

3.3. Upsampling is backwards strided convolution

Another way to connect coarse outputs to dense pixels is interpolation. For instance, simple bilinear interpolation computes each output y_{ij} from the nearest four inputs by a linear map that depends only on the relative positions of the input and output cells.

In a sense, upsampling with factor f is convolution with a *fractional* input stride of $1/f$. So long as f is integral, a natural way to upsample is therefore *backwards convolution* (sometimes called *deconvolution*) with an *output* stride of f. Such an operation is trivial to implement, since it simply reverses the forward and backward passes of convolution. Thus upsampling is performed in-network for end-to-end learning by backpropagation from the pixelwise loss.

Note that the deconvolution filter in such a layer need not be fixed (e.g., to bilinear upsampling), but can be learned. A stack of deconvolution layers and activation functions can even learn a nonlinear upsampling.

In our experiments, we find that in-network upsampling is fast and effective for learning dense prediction. Our best segmentation architecture uses these layers to learn to upsample for refined prediction in Section 4.2.

3.4. Patchwise training is loss sampling

这段也不用看了,也是不用的方法了

In stochastic optimization, gradient computation is driven by the training distribution. Both patchwise training and fully convolutional training can be made to produce any distribution, although their relative computational efficiency depends on overlap and minibatch size. Whole image fully convolutional training is identical to patchwise training where each batch consists of all the receptive fields of the units below the loss for an image (or collection of images). While this is more efficient than uniform sampling of patches, it reduces the number of possible batches. However, random selection of patches within an image may be recovered simply. Restricting the loss to a randomly sampled subset of its spatial terms (or, equivalently applying a DropConnect mask [36] between the output and the loss) excludes patches from the gradient computation.

If the kept patches still have significant overlap, fully convolutional computation will still speed up training. If gradients are accumulated over multiple backward passes, batches can include patches from several images.[2]

Sampling in patchwise training can correct class imbalance [27, 7, 2] and mitigate the spatial correlation of dense patches [28, 15]. In fully convolutional training, class balance can also be achieved by weighting the loss, and loss sampling can be used to address spatial correlation.

We explore training with sampling in Section 4.3, and do not find that it yields faster or better convergence for dense prediction. Whole image training is effective and efficient.

我们把ILSVRC的分类模型转变为全卷积网络,然后通过网络内上采样和逐像素loss来fine-tune模型,最后得到分割的模型.最后通过添加skip结构来融合粗粒度的语义信息和细粒度的位置信息

4. Segmentation Architecture

We cast ILSVRC classifiers into FCNs and augment them for dense prediction with in-network upsampling and a pixelwise loss. We train for segmentation by fine-tuning. Next, we add skips between layers to fuse coarse, semantic and local, appearance information. This skip architecture is learned end-to-end to refine the semantics and spatial precision of the output.

For this investigation, we train and validate on the PASCAL VOC 2011 segmentation challenge [6]. We train with a per-pixel multinomial logistic loss and validate with the standard metric of mean pixel intersection over union, with the mean taken over all classes, including background. The training ignores pixels that are masked out (as ambiguous or difficult) in the ground truth.

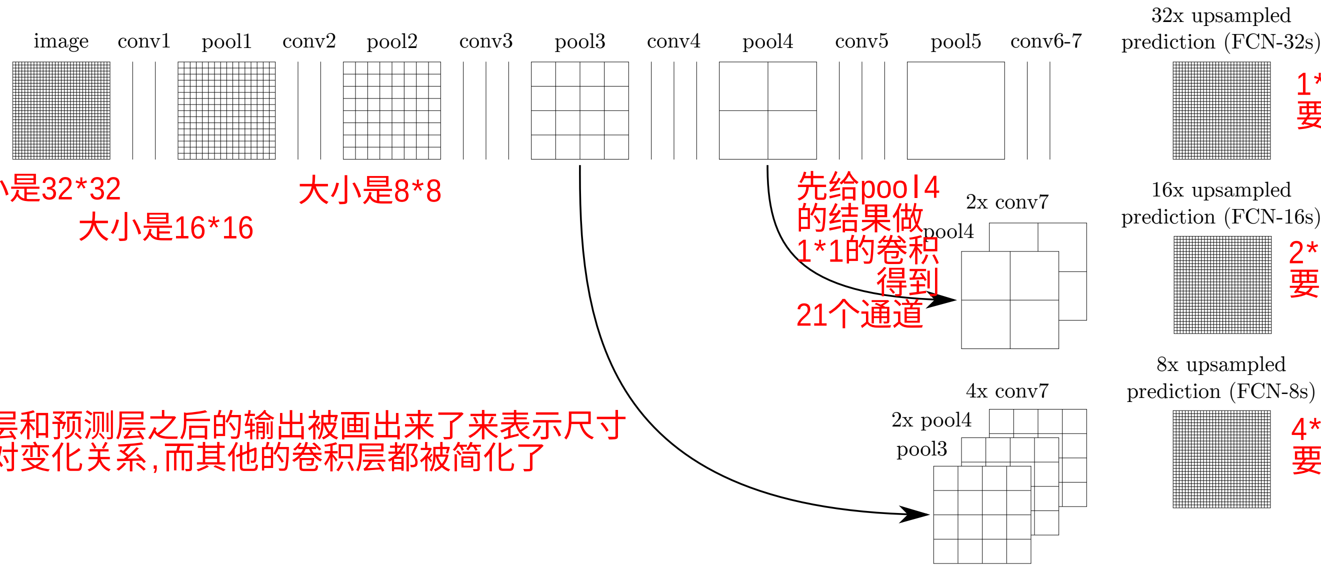
注意,训练阶段把边界给忽略了

[2] Note that not every possible patch is included this way, since the receptive fields of the final layer units lie on a fixed, strided grid. However, by shifting the image right and down by a random value up to the stride, random selection from all possible patches may be recovered.

这段之后开始介绍模型,主要分了两部分,第一部分是介绍如何对现有得模型进行修改得到FCN,第二部分是提出了含有skip结构的FCN

关于转置卷积的原理和介绍，参考李沐的课程：
https://www.bilibili.com/video/BV17o4y1X7Jn?spm_id_from=333.999.0.0
https://www.bilibili.com/video/BV1CM4y1K7r7?spm_id_from=333.999.0.0

写网络的时候得写一个函数自动计算输出的feature map的大小



1*1的上采样需要32的stride

大小是32*32

大小是16*16

大小是8*8

先给pool4的结果做1*1的卷积得到21个通道

2*2的上采样需要16的stride

池化层和预测层之后的输出被画出来了来表示尺寸的相对变化关系，而其他的卷积层都被简化了

4*4的上采样需要8的stride

Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

这一段主要讲如何从其它分类网络得到FCN

4.1. From classifier to dense FCN

实现细节：把最后的线性分类层丢弃，然后把所有的全连接层换成卷积层

We begin by convolutionalizing proven classification architectures as in Section 3. We consider the AlexNet[3] architecture [20] that won ILSVRC12, as well as the VGG nets [31] and the GoogLeNet[4] [32] which did exceptionally well in ILSVRC14. We pick the VGG 16-layer net[5], which we found to be equivalent to the 19-layer net on this task. For GoogLeNet, we use only the final loss layer, and improve performance by discarding the final average pooling layer. We decapitate each net by discarding the final classifier layer, and convert all fully connected layers to convolutions. We append a 1×1 convolution with channel dimension 21 to predict scores for each of the PASCAL classes (including background) at each of the coarse output locations, followed by a deconvolution layer to bilinearly upsample the coarse outputs to pixel-dense outputs as described in Section 3.3. Table 1 compares the preliminary validation results along with the basic characteristics of each net. We report the best results achieved after convergence at a fixed learning rate (at least 175 epochs).

然后在最后一层前用1*1的卷积让输出的通道称为21个通道，这21个通道就是最后每一类的得分

Fine-tuning from classification to segmentation gave reasonable predictions for each net. Even the worst model achieved $\sim 75\%$ of state-of-the-art performance. The segmentation-equipped VGG net (FCN-VGG16) already appears to be state-of-the-art at 56.0 mean IU on val, compared to 52.6 on test [15]. Training on extra data raises FCN-VGG16 to 59.4 mean IU and FCN-AlexNet to 48.0 mean IU on a subset of val[7]. Despite similar classification accuracy, our implementation of GoogLeNet did not match the VGG16 segmentation result.

[3]Using the publicly available `CaffeNet` reference model.

[4]Since there is no publicly available version of GoogLeNet, we use our own reimplementation. Our version is trained with less extensive data augmentation, and gets 68.5% top-1 and 88.4% top-5 ILSVRC accuracy.

[5]Using the publicly available version from the Caffe model zoo.

使用上面的方法来修改之前的模型，然后只训练了175个epoch，就得到了非常好的性能，最差的模型达到了SOTA 75%的性能，而最好的模型性能已经和SOTA持平

Table 1. We adapt and extend three classification convnets. We compare performance by mean intersection over union on the validation set of PASCAL VOC 2011 and by inference time (averaged over 20 trials for a 500×500 input on an NVIDIA Tesla K40c). We detail the architecture of the adapted nets with regard to dense prediction: number of parameter layers, receptive field size of output units, and the coarsest stride within the net. (These numbers give the best performance obtained at a fixed learning rate, not best performance possible.)

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

接下来我们定义了新提出的FCN的网络，他把分层的特征结合起来，并且提高了输出的空间准确率

4.2. Combining what and where

We define a new fully convolutional net (FCN) for segmentation that combines layers of the feature hierarchy and refines the spatial precision of the output. See Figure 3.

While fully convolutionalized classifiers can be fine-

虽然转换为卷积之后的分类网络能够在标准的衡量方法下获得非常高的性能,但从可视化的角度看到的结果依旧是无法让人满意的.而造成最后结果很差的原因,从直观上来看就是最后一层的32为步长的转置卷积限制了上采样后的输出的细节信息.

为了解决这个问题,我们提出了skip结构.skip结构把来自底层的输出和最终的上采样层进行了结合.而语义分割需要精细尺度的信息,因此结合前几层的,看了比较少得像素的输出是合理的

tuned to segmentation as shown in 4.1, and even score highly on the standard metric, their output is dissatisfyingly coarse (see Figure 4). The 32 pixel stride at the final prediction layer limits the scale of detail in the upsampled output.

We address this by adding skips [1] that combine the final prediction layer with lower layers with finer strides. This turns a line topology into a DAG, with edges that skip ahead from lower layers to higher ones (Figure 3). As they see fewer pixels, the finer scale predictions should need fewer layers, so it makes sense to make them from shallower net outputs. Combining fine layers and coarse layers lets the model make local predictions that respect global structure. By analogy to the jet of Koenderick and van Doorn [19], we call our nonlinear feature hierarchy the *deep jet*.

We first divide the output stride in half by predicting from a 16 pixel stride layer. We add a 1×1 convolution layer on top of `pool4` to produce additional class predictions. We fuse this output with the predictions computed on top of `conv7` (convolutionalized `fc7`) at stride 32 by adding a $2\times$ upsampling layer and summing[6] both predictions (see Figure 3). We initialize the $2\times$ upsampling to bilinear interpolation, but allow the parameters to be learned as described in Section 3.3. Finally, the stride 16 predictions are upsampled back to the image. We call this net FCN-16s. FCN-16s is learned end-to-end, initialized with the parameters of the last, coarser net, which we now call FCN-32s. The new parameters acting on `pool4` are zero-initialized so that the net starts with unmodified predictions. The learning rate is decreased by a factor of 100.

实现的细节:额外添加的层从0开始初始化,其他层的参数利用先前的FCN-32s的结果

Learning this skip net improves performance on the validation set by 3.0 mean IU to 62.4. Figure 4 shows improvement in the fine structure of the output. We compared this fusion with learning only from the `pool4` layer, which resulted in poor performance, and simply decreasing the learning rate without adding the skip, which resulted in an insignificant performance improvement without improving the quality of the output.

如果没有最后的信息,只从pool4的结果学习输出的话,性能就会很差

We continue in this fashion by fusing predictions from `pool3` with a $2\times$ upsampling of predictions fused from `pool4` and `conv7`, building the net FCN-8s. We obtain a minor additional improvement to 62.7 mean IU, and find a slight improvement in the smoothness and detail of our output. At this point our fusion improvements have met diminishing returns, both with respect to the IU metric which emphasizes large-scale correctness, and also in terms of the improvement visible e.g. in Figure 4, so we do not continue fusing even lower layers.

然后我们把pool3的结果拿出来进行融合.最后得到的性能只有一点点的提升

Refinement by other means Decreasing the stride of pooling layers is the most straightforward way to obtain finer predictions. However, doing so is problematic for our VGG16-based net. Setting the `pool5` stride to 1 requires our convolutionalized `fc6` to have kernel size 14×14 to maintain its receptive field size. In addition to their computational cost, we had difficulty learning such large filters. We attempted to re-architect the layers above `pool5` with smaller filters, but did not achieve comparable performance; one possible explanation is that the ILSVRC initialization of the upper layers is important.

虽然没有办法再利用更加精细的feature map了,但是我们认为在VGG16最后一层的步长太大了,因此损失了精细的空间信息.然而保持最后的输出的feature map形状不变的话,如果改小步长的话,那么就需要增加卷积核,因此最后的卷积核就成了14*14大小的卷积核.而我们没法训练这么大的卷积核

最后,本文尝试着修改了最后输出的形状,结果最后发现分类的结果不如原始的性能,而这可能是因为在ILSVRC上训练的问题

Another way to obtain finer predictions is to use the shift-and-stitch trick described in Section 3.2. In limited experiments, we found the cost to improvement ratio from this method to be worse than layer fusion.

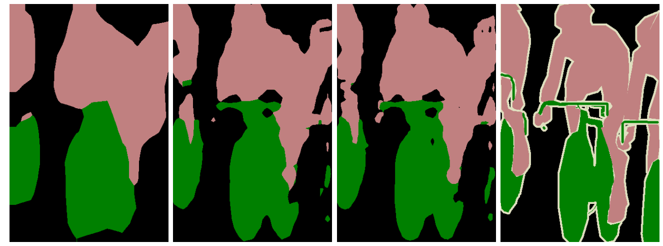


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

Table 2. Comparison of skip FCNs on a subset[7] of PASCAL VOC 2011 segval. Learning is end-to-end, except for FCN-32s-fixed, where only the last layer is fine-tuned. Note that FCN-32s is FCN-VGG16, renamed to highlight stride.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

4.3. Experimental framework

Optimization We train by SGD with momentum. We use a minibatch size of 20 images and fixed learning rates of 10^{-3}, 10^{-4}, and 5^{-5} for FCN-AlexNet, FCN-VGG16, and FCN-GoogLeNet, respectively, chosen by line search. We use momentum 0.9, weight decay of 5^{-4} or 2^{-4}, and doubled learning rate for biases, although we found training to be sensitive to the learning rate alone. We zero-initialize the class scoring layer, as random initialization yielded neither better performance nor faster convergence. Dropout was included where used in the original classifier nets.

Fine-tuning We fine-tune all layers by back-propagation through the whole net. Fine-tuning the output classifier alone yields only 70% of the full fine-tuning performance as compared in Table 2. Training from scratch is not feasible considering the time required to learn the base classification nets. (Note that the VGG net is trained in stages, while we initialize from the full 16-layer

[6] Max fusion made learning difficult due to gradient switching.

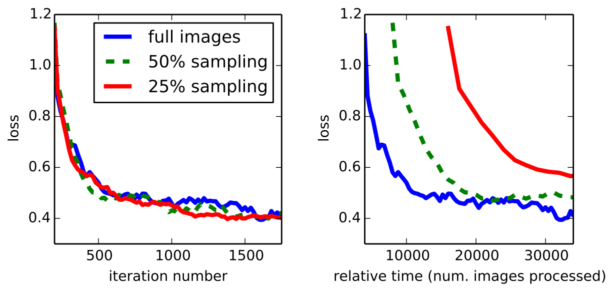


Figure 5. Training on whole images is just as effective as sampling patches, but results in faster (wall time) convergence by making more efficient use of data. Left shows the effect of sampling on convergence rate for a fixed expected batch size, while right plots the same by relative wall time.

version.) Fine-tuning takes three days on a single GPU for the coarse FCN-32s version, and about one day each to upgrade to the FCN-16s and FCN-8s versions.

More Training Data The PASCAL VOC 2011 segmentation training set labels 1112 images. Hariharan *et al.* [14] collected labels for a larger set of 8498 PASCAL training images, which was used to train the previous state-of-the-art system, SDS [15]. This training data improves the FCN-VGG16 validation score[7] by 3.4 points to 59.4 mean IU.

Patch Sampling As explained in Section 3.4, our full image training effectively batches each image into a regular grid of large, overlapping patches. By contrast, prior work randomly samples patches over a full dataset [27, 2, 7, 28, 9], potentially resulting in higher variance batches that may accelerate convergence [22]. We study this tradeoff by spatially sampling the loss in the manner described earlier, making an independent choice to ignore each final layer cell with some probability $1 - p$. To avoid changing the effective batch size, we simultaneously increase the number of images per batch by a factor $1/p$. Note that due to the efficiency of convolution, this form of rejection sampling is still faster than patchwise training for large enough values of p (e.g., at least for $p > 0.2$ according to the numbers in Section 3.1). Figure 5 shows the effect of this form of sampling on convergence. We find that sampling does not have a significant effect on convergence rate compared to whole image training, but takes significantly more time due to the larger number of images that need to be considered per batch. We therefore choose unsampled, whole image training in our other experiments.

Class Balancing Fully convolutional training can balance classes by weighting or sampling the loss. Although our labels are mildly unbalanced (about $3/4$ are background), we find class balancing unnecessary.

Dense Prediction The scores are upsampled to the input dimensions by deconvolution layers within the net. Final layer deconvolutional filters are fixed to bilinear interpolation, while intermediate upsampling layers are initialized to bilinear upsampling, and then learned.

Augmentation We tried augmenting the training data by randomly mirroring and "jittering" the images by translating them up to 32 pixels (the coarsest scale of prediction) in each direction. This yielded no noticeable improvement.

Implementation All models are trained and tested with Caffe [18] on a single NVIDIA Tesla K40c. Our models and code are publicly available at `http://fcn.berkeleyvision.org`.

5. Results

We test our FCN on semantic segmentation and scene parsing, exploring PASCAL VOC, NYUDv2, and SIFT Flow. Although these tasks have historically distinguished between objects and regions, we treat both uniformly as pixel prediction. We evaluate our FCN skip architecture on each of these datasets, and then extend it to multi-modal input for NYUDv2 and multi-task prediction for the semantic and geometric labels of SIFT Flow.

Metrics We report four metrics from common semantic segmentation and scene parsing evaluations that are variations on pixel accuracy and region intersection over union (IU). Let n_{ij} be the number of pixels of class i predicted to belong to class j, where there are n_{cl} different classes, and let $t_i = \sum_j n_{ij}$ be the total number of pixels of class i. We compute:

- pixel accuracy: $\sum_i n_{ii} / \sum_i t_i$
- mean accuraccy: $(1/n_{\text{cl}}) \sum_i n_{ii}/t_i$
- mean IU: $(1/n_{\text{cl}}) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$
- frequency weighted IU: $(\sum_k t_k)^{-1} \sum_i t_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$

PASCAL VOC Table 3 gives the performance of our FCN-8s on the test sets of PASCAL VOC 2011 and 2012, and compares it to the previous state-of-the-art, SDS [15], and the well-known R-CNN [10]. We achieve the best results on mean IU[8] by a relative margin of 20%. Inference time is reduced $114\times$ (convnet only, ignoring proposals and refinement) or $286\times$ (overall).

Table 3. Our fully convolutional net gives a 20% relative improvement over the state-of-the-art on the PASCAL VOC 2011 and 2012 test sets and reduces inference time.

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [10]	47.9	-	-
SDS [15]	52.6	51.6	$\sim$ 50 s
FCN-8s	**62.7**	**62.2**	$\sim$ **175 ms**

NYUDv2 [30] is an RGB-D dataset collected using the

[7] There are training images from [14] included in the PASCAL VOC 2011 val set, so we validate on the non-intersecting set of 736 images.

[8] This is the only metric provided by the test server.

Table 4. Results on NYUDv2. *RGBD* is early-fusion of the RGB and depth channels at the input. *HHA* is the depth embedding of [13] as horizontal disparity, height above ground, and the angle of the local surface normal with the inferred gravity direction. *RGB-HHA* is the jointly trained late fusion model that sums RGB and HHA predictions.

	pixel acc.	mean acc.	mean IU	f.w. IU
Gupta *et al.* [13]	60.3	-	28.6	47.0
FCN-32s RGB	60.0	42.2	29.2	43.9
FCN-32s RGBD	61.5	42.4	30.5	45.5
FCN-32s HHA	57.1	35.2	24.2	40.4
FCN-32s RGB-HHA	64.3	44.9	32.8	48.0
FCN-16s RGB-HHA	**65.4**	**46.1**	**34.0**	**49.5**

Microsoft Kinect. It has 1449 RGB-D images, with pixelwise labels that have been coalesced into a 40 class semantic segmentation task by Gupta *et al.* [12]. We report results on the standard split of 795 training images and 654 testing images. (Note: all model selection is performed on PASCAL 2011 val.) Table 4 gives the performance of our model in several variations. First we train our unmodified coarse model (FCN-32s) on RGB images. To add depth information, we train on a model upgraded to take four-channel RGB-D input (early fusion). This provides little benefit, perhaps due to the difficultly of propagating meaningful gradients all the way through the model. Following the success of Gupta *et al.* [13], we try the three-dimensional HHA encoding of depth, training nets on just this information, as well as a "late fusion" of RGB and HHA where the predictions from both nets are summed at the final layer, and the resulting two-stream net is learned end-to-end. Finally we upgrade this late fusion net to a 16-stride version.

SIFT Flow is a dataset of 2,688 images with pixel labels for 33 semantic categories ("bridge", "mountain", "sun"), as well as three geometric categories ("horizontal", "vertical", and "sky"). An FCN can naturally learn a joint representation that simultaneously predicts both types of labels. We learn a two-headed version of FCN-16s with semantic and geometric prediction layers and losses. The learned model performs as well on both tasks as two independently trained models, while learning and inference are essentially as fast as each independent model by itself. The results in Table 5, computed on the standard split into 2,488 training and 200 test images,[9] show state-of-the-art performance on both tasks.

6. Conclusion

Fully convolutional networks are a rich class of models, of which modern classification convnets are a special case. Recognizing this, extending these classification nets to segmentation, and improving the architecture with multi-resolution layer combinations dramatically improves the state-of-the-art, while simultaneously simplifying and speeding up learning and inference.

Acknowledgements This work was supported in part by DARPA's MSEE and SMISC programs, NSF awards IIS-1427425, IIS-1212798, IIS-1116411, and the NSF GRFP, Toyota, and the Berkeley Vision and Learning Center. We gratefully acknowledge NVIDIA for GPU donation. We

[9]Three of the SIFT Flow categories are not present in the test set. We made predictions across all 33 categories, but only included categories actually present in the test set in our evaluation.

Table 5. Results on SIFT Flow[9] with class segmentation (center) and geometric segmentation (right). Tighe [33] is a non-parametric transfer method. Tighe 1 is an exemplar SVM while 2 is SVM + MRF. Farabet is a multi-scale convnet trained on class-balanced samples (1) or natural frequency samples (2). Pinheiro is a multi-scale, recurrent convnet, denoted RCNN$_3$ ($\circ^3$). The metric for geometry is pixel accuracy.

	pixel acc.	mean acc.	mean IU	f.w. IU	geom. acc.
Liu *et al.* [23]	76.7	-	-	-	-
Tighe *et al.* [33]	-	-	-	-	90.8
Tighe *et al.* [34] 1	75.6	41.1	-	-	-
Tighe *et al.* [34] 2	78.6	39.2	-	-	-
Farabet *et al.* [7] 1	72.3	50.8	-	-	-
Farabet *et al.* [7] 2	78.5	29.6	-	-	-
Pinheiro *et al.* [28]	77.7	29.8	-	-	-
FCN-16s	**85.2**	**51.7**	39.5	76.1	**94.3**

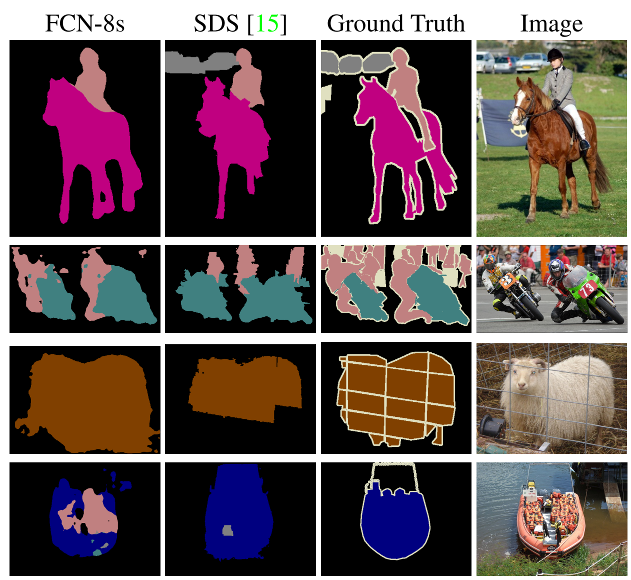


Figure 6. Fully convolutional segmentation nets produce state-of-the-art performance on PASCAL. The left column shows the output of our highest performing net, FCN-8s. The second shows the segmentations produced by the previous state-of-the-art system by Hariharan *et al.* [15]. Notice the fine structures recovered (first row), ability to separate closely interacting objects (second row), and robustness to occluders (third row). The fourth row shows a failure case: the net sees lifejackets in a boat as people.

thank Bharath Hariharan and Saurabh Gupta for their advice and dataset tools. We thank Sergio Guadarrama for reproducing GoogLeNet in Caffe. We thank Jitendra Malik for his helpful comments. Thanks to Wei Liu for pointing out an issue wth our SIFT Flow mean IU computation and an error in our frequency weighted mean IU formula.

References

[1] C. M. Bishop. *Pattern recognition and machine learning*, page 229. Springer-Verlag New York, 2006. 6

[2] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, pages 2852–2860, 2012. 1, 2, 4, 7

[3] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 1, 2

[4] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 633–640. IEEE, 2013. 2

[5] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014. 2

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html. 4

[7] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2013. 1, 2, 4, 7, 8

[8] P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor matching with convolutional neural networks: a comparison to SIFT. *CoRR*, abs/1405.5769, 2014. 1

[9] Y. Ganin and V. Lempitsky. N^4-fields: Neural network nearest neighbor fields for image transforms. In *ACCV*, 2014. 1, 2, 7

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014. 1, 2, 7

[11] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. In *ICIP*, 2013. 3, 4

[12] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, 2013. 8

[13] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*. Springer, 2014. 1, 2, 8

[14] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011. 7

[15] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision (ECCV)*, 2014. 1, 2, 4, 5, 7, 8

[16] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Computer Vision and Pattern Recognition*, 2015. 2

[17] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1, 2

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 7

[19] J. J. Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biological cybernetics*, 55(6):367–375, 1987. 6

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 3, 5

[21] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to hand-written zip code recognition. In *Neural Computation*, 1989. 2, 3

[22] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 1998. 7

[23] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):978–994, 2011. 8

[24] J. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *NIPS*, 2014. 1

[25] S. Mallat. *A wavelet tour of signal processing*. Academic press, 2nd edition, 1999. 4

[26] O. Matan, C. J. Burges, Y. LeCun, and J. S. Denker. Multi-digit recognition using a space displacement neural network. In *NIPS*, pages 488–495. Citeseer, 1991. 2

[27] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano. Toward automatic phenotyping of developing embryos from videos. *Image Processing, IEEE Transactions on*, 14(9):1360–1371, 2005. 1, 2, 4, 7

[28] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *ICML*, 2014. 1, 2, 4, 7, 8

[29] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 1, 2, 4

[30] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 7

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 2, 3, 5

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.

Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 1, 2, 3, 5

[33] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *ECCV*, pages 352–365. Springer, 2010. 8

[34] J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *CVPR*, 2013. 8

[35] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *CoRR*, abs/1406.2984, 2014. 2

[36] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013. 4

[37] R. Wolf and J. C. Platt. Postal address block location using a convolutional locator network. *Advances in Neural Information Processing Systems*, pages 745–745, 1994. 2

[38] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014. 2

[39] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014. 1