Convolutional networks → trained end-to-end, pixels to pixels, exceed state of the art in semantic segmentation.

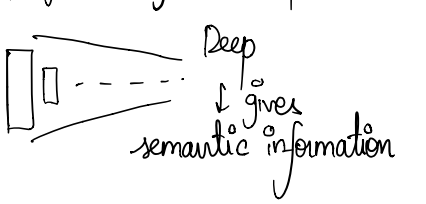
↳ means study of meaning

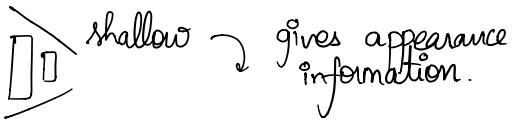
Abstract

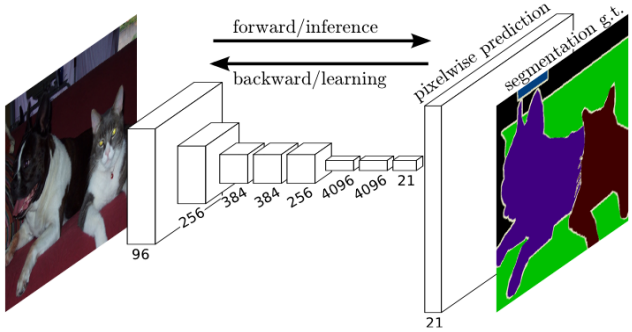
(•) Target: Key insight is to build "fully convolutional networks" that take in arbitary size and produce correspondingly-sized output with efficient inference and learning.

(•) We adapt contemporary classification networks like Alexnet, the VGG net and Googlenet into fully convolutional networks and transfer their learned representations by fine-tuning to the segmentation task.

(•) 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.







Introduction

(•) The Natural next step in the progression from coarse to fine inference is to make prediction at every pixel.

(•) This paper shows. FCNs exceed State of the art.

→ Fully convolutional versions of existing networks predict dense outputs from arbitary-sized input.
→ 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.

In this paper Patchwise training is mentioned. Question How does it differ from Fully convolutional Network?

from StackExchange → A FCN. takes a whole image MxM image and produces outputs for all sub-images in a single convnet forward pass.

Patchwise training explicitly crops out the subimages and produces outputs for each subimage in independent forward passes. Therefore FCNs are substantially faster than patchwise training.

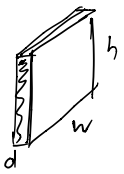
Now while this is fast, it restricts your training sampling process compared to patchwise training.

(•) 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 non-linear local-to-global pyramid.

3. Fully Convolutional Networks.

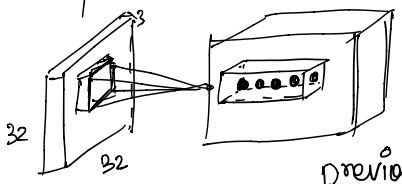
Each layer of data in a convnet is 3-D



h x w x d.

h, w → spatial dimensions, d is the feature or channel dimension.

(•) Locations in higher layers correspond to the locations in the image they are path-connected to, which are called receptive fields.



for high dimensional input it is impractical to connect neurons to all neurons in the previous volume. Instead, we connect each neuron to only a local region of the input volume. The spatial extent of this connectivity is a hyperparameter called receptive field of the neuron (Equivalently this the filter size).

ex: 1. Suppose input volume is [32x32x3]. If the receptive field or filter size is 5x5, then each neuron in the conv. layer will have weights to a [5x5x3] region in the input volume.

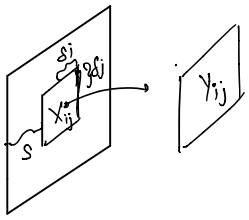
So per neuron → 5x5x3 + 1 (bias)

75+1 = 76 parameters.

(•) 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 co-ordinates.

X_{ij} → Data vector at location (i,j) in a particular layer, and Y_{ij} for the following layer, $Y_{ij} = f_{ks}(\{X_{si+\delta i, sj+\delta j}\}\ 0 \le \delta i, \delta j \le k)$.

k is the kernel size, S → stride or subsampling factor.

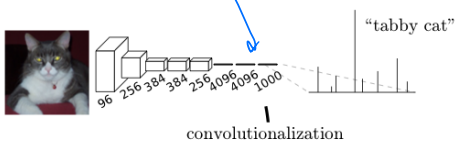


f_{ks} determines the type of layer

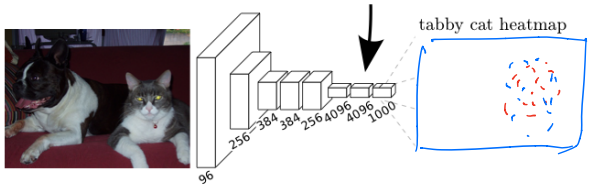
↳ a matrix multiplication for convolution or avg pooling
a spatial max for max pooling
an elementwise non-linearity for an activation function.

3.1. Adapting classifiers for dense prediction.

(•) Typical recognition nets, including AlexNet, LeNet take fixed-sized inputs and produce non-spatial outputs.



(•) The fully connected layers of these nets have fixed dimensions and throw away spatial co-ordinates.



(•) However, these fully connected layers can also be viewed as convolutions with kernels 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.

(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.

(4.) · Segmentation architecture.

(•) Cast ILSVRC classifier into FCNs and augment them for dense prediction with in-network upsampling and pixelwise loss.

(ImageNet Large scale Visual Recognition challenge)

(•) Next we add skip 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.

(•) We train with a per-pixel multi-nomial logistic loss and validate with the standard metric of mean pixel intersection over union, with the mean taken over all classes, including background.

(4.1). From classifier to dense FCN.

Start by convolutionalizing proven classification architectures. AlexNet → won ILSVRC12

VGG

GoogLeNet → ILSVRC14

The pick → VGG16 performed similar to VGG19

(*) For GoogLeNet, use only the final loss layer and improve performance by discarding the final average pooling layer.

(*) Decapitate each net by discarding the final classifier layer, and convert all fully connected layers to convolution.

(*) We append a 1×1 convolution with channel dimension 21 to to predict the number of PASCAL classes (including backgrd) at each of the coarse output locations, followed by a deconvolution layer to bi-linearly upsample the coarse outputs to pixel-dense outputs.

Table 1 compares validation results along with basic characteristic of each network.

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

The best results after 175 epochs (fixed learning rate).

- Fine tuning from classification to segmentation gave reasonable predictions for each net. Even the worst model achieved state of the art performance (~ 75%).
- 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. Training on extra data raises FCN-VGG16 to 59.4 mean IU. and FCN-AlexNet to 48. mean IU.

4.2 Combining what and where

- While fully convolutionized classifiers can be fine tuned to segmentation, and even score highly on the standard metric, their output is dissatisfyingly coarse.
- The 32 pixel stride at the final prediction layer limits the scale of detail in the upsampled output.
- This is addressed by adding skips that combine the final prediction layer with lower layers with finer strides.

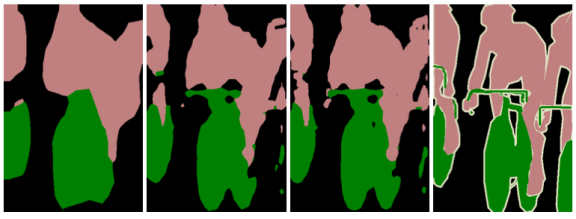


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).