

Jeremy Jordan – An overview of semantic image segmentation.



15

DATA SCIENCE

An overview of semantic image segmentation.



JEREMY JORDAN

21 MAY 2018 • 12 MIN READ

In this post, I'll discuss how to use convolutional neural networks for the task of **semantic image segmentation**. Image segmentation is a computer vision task in which we label specific regions of an image according to what's being shown.

"What's in this image, and where in the image is it located?"

Jump to:

- Representing the task
- Constructing an architecture
 - Methods for upsampling
 - Fully convolutional networks

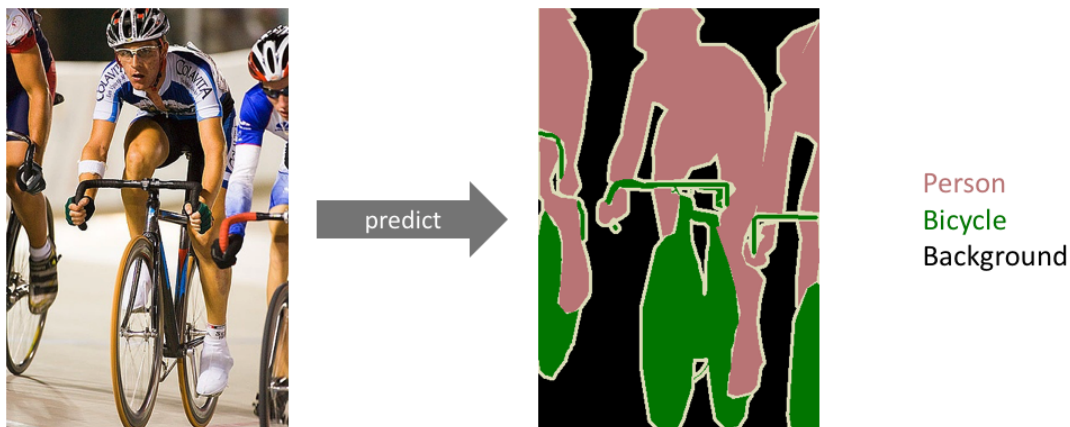
You've successfully subscribed to Jeremy Jordan!

- Dilated convolutions

Jeremy Jordan – An overview of semantic image segmentation.

- Common datasets and segmentation competitions
- Further reading

More specifically, the goal of semantic image segmentation is to label *each pixel* of an image with a corresponding **class** of what is being represented. Because we're predicting for every pixel in the image, this task is commonly referred to as **dense prediction**.



An example of semantic segmentation, where the goal is to predict class labels for each pixel in the image. [\(Source\)](#)

One important thing to note is that we're not separating *instances* of the same class; we only care about the category of each pixel. In other words, if you have two objects of the same category in your input image, the segmentation map does not inherently distinguish these as separate objects. There exists a different class of models, known as *instance segmentation* models, which *do* distinguish between separate objects of the same class.

You've successfully subscribed to Jeremy Jordan!

- **Autonomous vehicles**

Jeremy Jordan – An overview of semantic image segmentation.

their environment so that self-driving cars can safely integrate into our existing roads.



A real-time segmented road scene for autonomous driving. [\(Source\)](#)

- **Medical image diagnostics**

Machines can augment analysis performed by radiologists, greatly reducing the time required to run diagnostic tests.



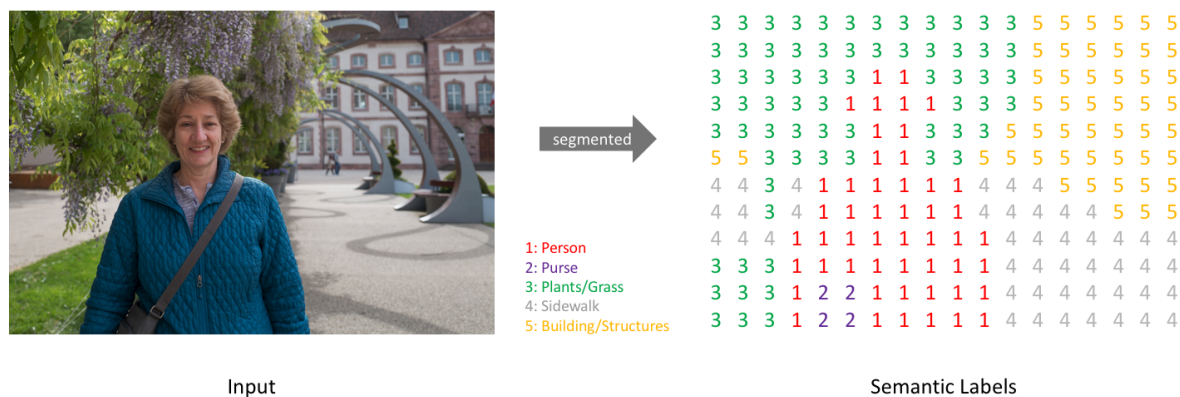
You've successfully subscribed to Jeremy Jordan!

Input Image

Segmented Image

Representing the task

Simply, our goal is to take either a RGB color image ($height \times width \times 3$) or a grayscale image ($height \times width \times 1$) and output a segmentation map where each pixel contains a class label represented as an integer ($height \times width \times 1$).

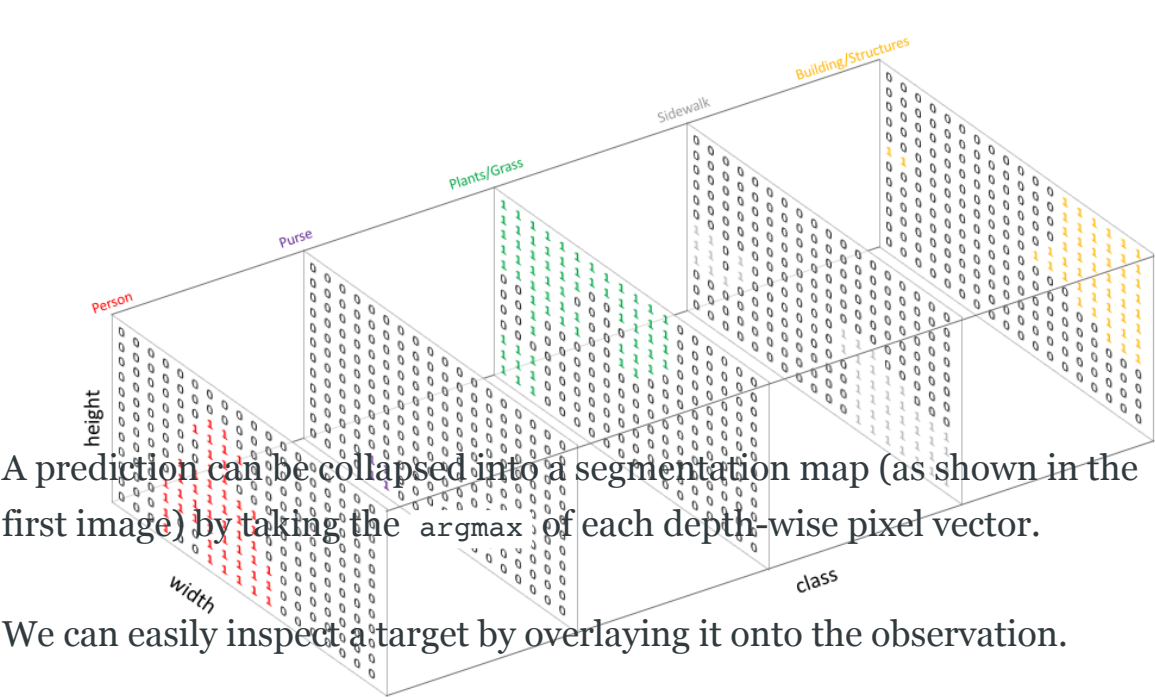


Note: For visual clarity, I've labeled a low-resolution prediction map. In reality, the segmentation label resolution should match the original input's resolution.

Similar to how we treat standard categorical values, we'll create our **target** by one-hot encoding the class labels - essentially creating an output channel for each of the possible classes.

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.



When we overlay a *single channel* of our target (or prediction), we refer to this as a **mask** which illuminates the regions of an image where a specific class is present.

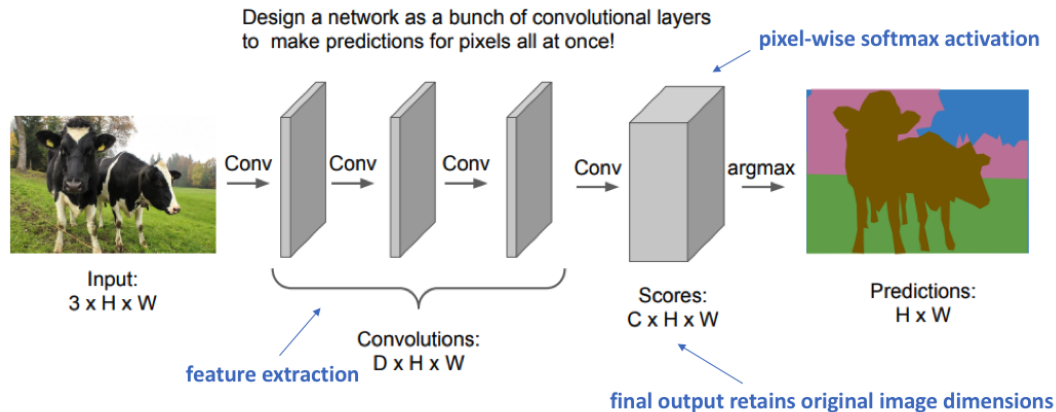


You've successfully subscribed to Jeremy Jordan!

this task is to simply stack a number of convolutional layers (with same padding to preserve dimensions) and output a final segmentation map.

Jeremy Jordan – An overview of semantic image segmentation.

however, it's quite computationally expensive to preserve the full resolution throughout the network.



Downside: Preserving image dimensions throughout entire network will be computationally expensive.

Image credit

Recall that for deep convolutional networks, earlier layers tend to learn low-level concepts while later layers develop more high-level (and specialized) feature mappings. **In order to maintain expressiveness, we typically need to increase the number of feature maps (channels) as we get deeper in the network.**

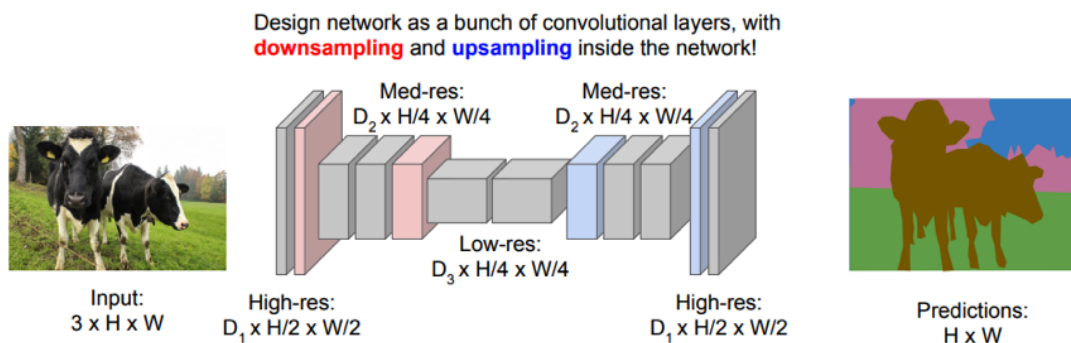
This didn't necessarily pose a problem for the task of image classification, because for that task we only care about *what* the image contains (and not where it is located). Thus, we could alleviate computational burden by periodically downsampling our feature maps through pooling or strided convolutions (ie. compressing the spatial resolution) without concern. However, for image segmentation, we would like our model to produce a *full-resolution* semantic prediction.

You've successfully subscribed to Jeremy Jordan!

encoder/decoder structure where we *downsample* the spatial resolution of the input, developing lower-resolution feature mappings

Jeremy Jordan – An overview of semantic image segmentation.

and the *upsample* the feature representations into a full-resolution segmentation map.



Solution: Make network deep and *work at a lower spatial resolution* for many of the layers.

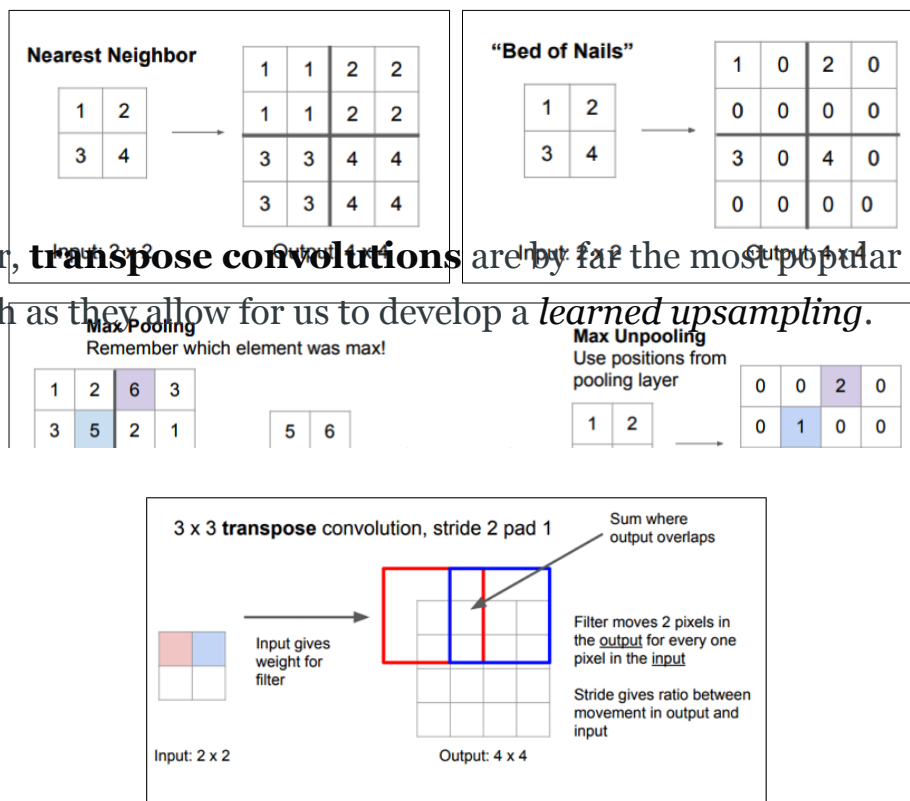
Image credit

Methods for upsampling

There are a few different approaches that we can use to *upsample* the resolution of a feature map. Whereas pooling operations downsample the resolution by summarizing a local area with a single value (ie. average or max pooling), "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.



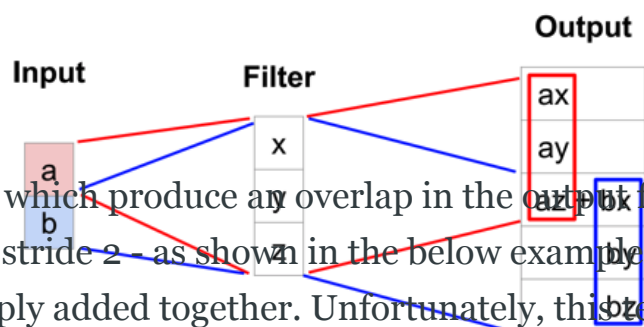
However, **transpose convolutions** are by far the most popular approach as they allow for us to develop a *learned upsampling*.

Image credit

Whereas a typical convolution operation will take the dot product of the values currently in the filter's view and produce a single value for the corresponding output position, a transpose convolution essentially does the opposite. For a transpose convolution, we take a single value from the low-resolution feature map and multiply all of the weights in our filter by this value, projecting those weighted values into the output feature map.

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.



For filter sizes which produce any overlap in the output feature map (eg. 3x3 filter with stride 2 – as shown in the below example), the overlapping values are simply added together. Unfortunately, this tends to produce a checkerboard artifact in the output and is undesirable, so it's best to ensure that your filter size does not produce an overlap.

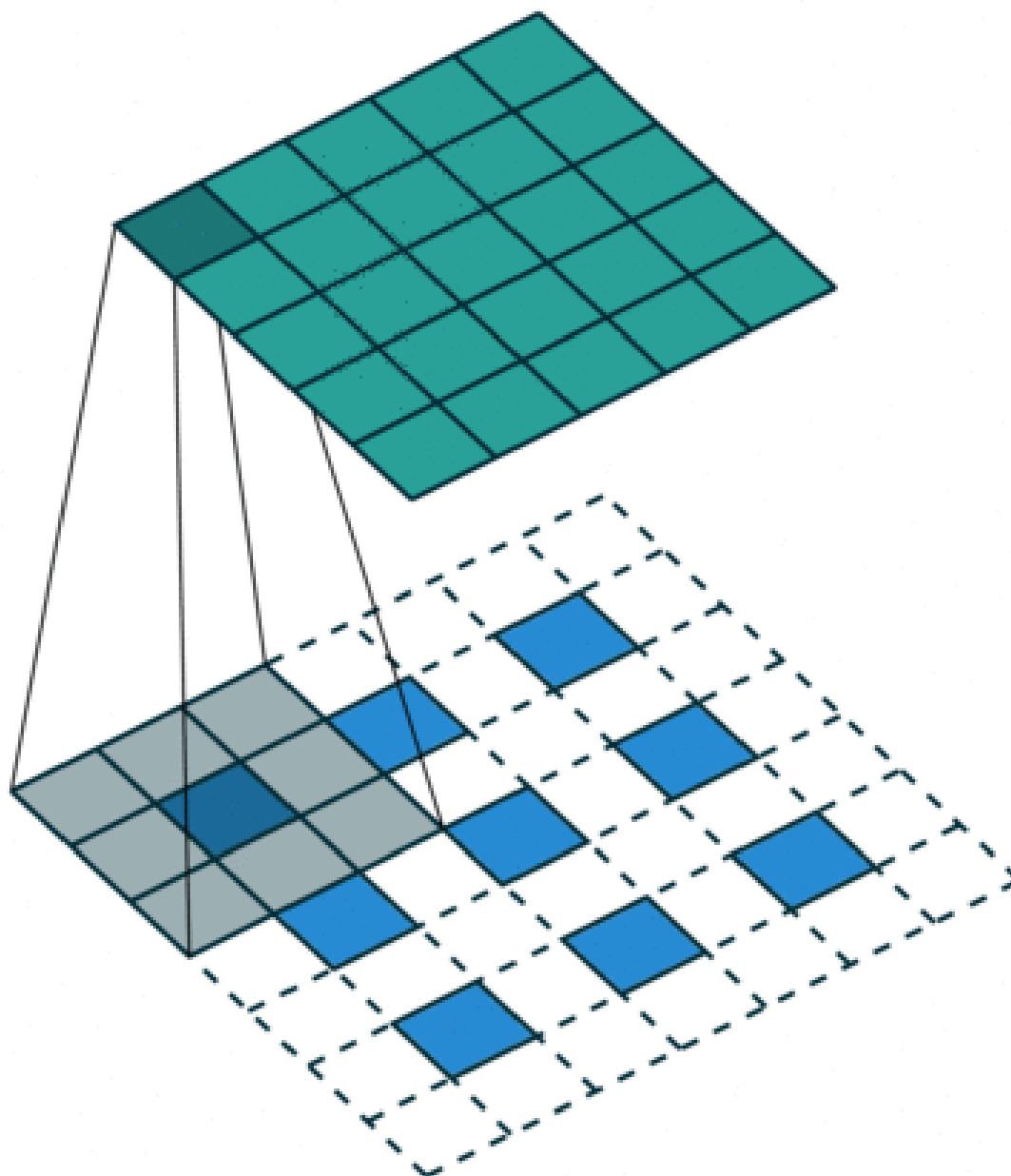
A simplified 1D example of upsampling through a transpose operation. [\(Source\)](#)

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.



15



Fully convolutional networks

Input in blue, output in green. (Source)

The approach of using a "fully convolutional" network trained end-to-end, pixels-to-pixels for the task of image segmentation was introduced by Long et al in late 2014. The paper's authors propose adapting existing

You've successfully subscribed to Jeremy Jordan!

transpose convolutional layers to upsample the coarse feature maps into a full-resolution segmentation map.

Jeremy Jordan – An overview of semantic image segmentation.

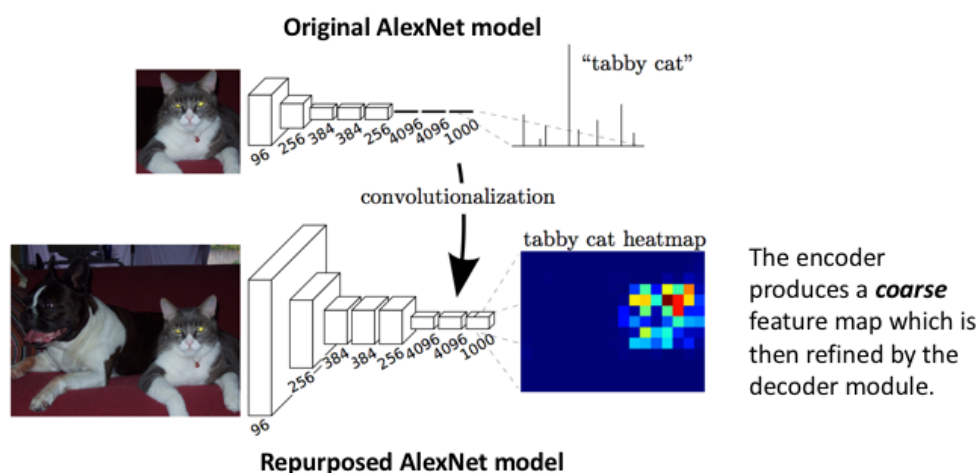


Image credit (with modification)

The full network, as shown below, is trained according to a pixel-wise cross entropy loss.

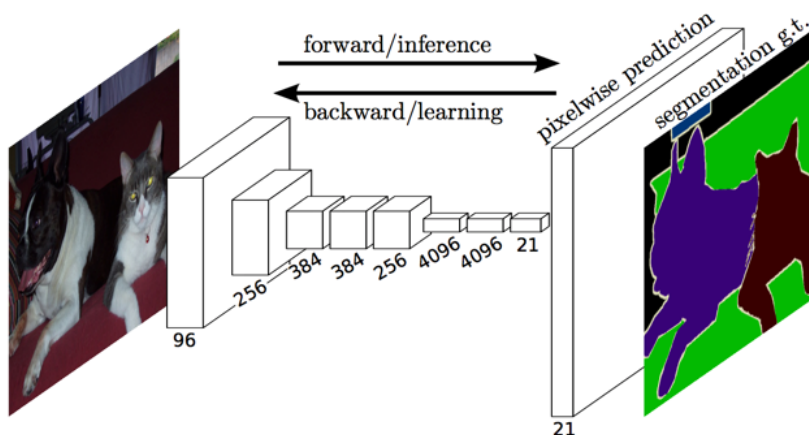


Image credit

You've successfully subscribed to Jeremy Jordan!

grained segmentations (as shown below).

Jeremy Jordan – An overview of semantic image segmentation.



15

Ground truth target



Predicted segmentation

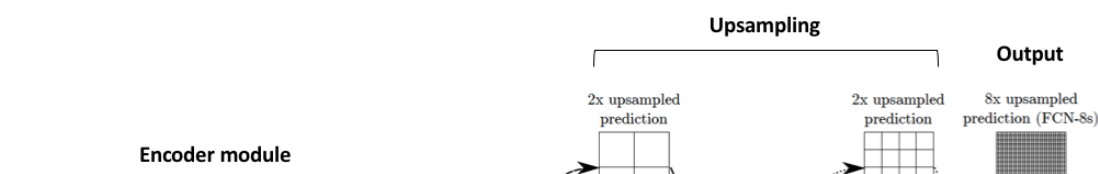


The paper's authors comment eloquently on this struggle:

*Semantic segmentation faces an inherent tension between semantics and location: global information resolves **what** while local information resolves **where**... Combining fine layers and coarse layers lets the model make local predictions that respect global structure. – Long et al.*

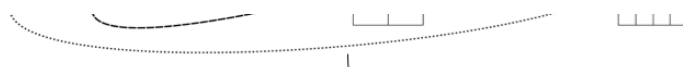
Adding skip connections

The authors address this tension by slowly upsampling (in stages) the encoded representation, adding "skip connections" from earlier layers, and summing these two feature maps.



You've successfully subscribed to Jeremy Jordan!

Input



Skip connections

Jeremy Jordan – An overview of semantic image segmentation.

Image credit (with modification)

These skip connections from earlier layers in the network (prior to a downsampling operation) should provide the necessary detail in order to reconstruct accurate shapes for segmentation boundaries. Indeed, we can recover more fine-grain detail with the addition of these skip connections.

Ground truth target



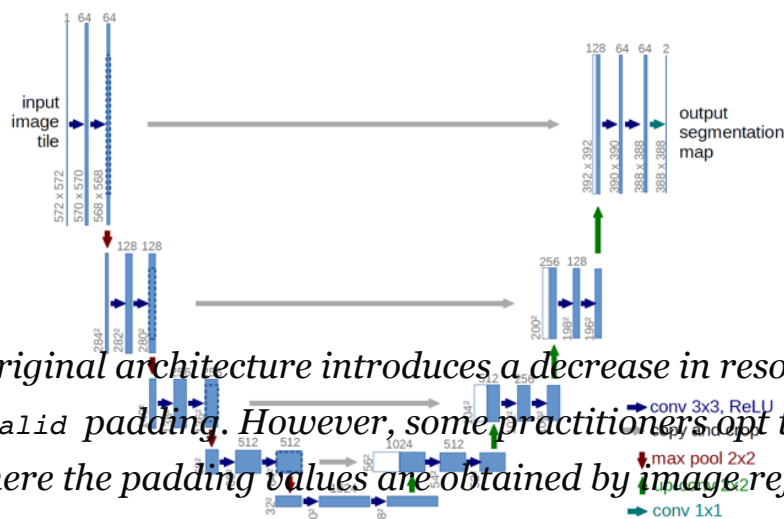
Predicted segmentation



Ronneberger et al. improve upon the "fully convolutional" architecture primarily through **expanding the capacity of the decoder** module of the network. More concretely, they propose the **U-Net architecture** which "consists of a contracting path to capture context and a **symmetric** expanding path that enables precise localization." This simpler architecture has grown to be very popular and has been adapted for a variety of segmentation problems.

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.



Note: The original architecture introduces a decrease in resolution due to the use of `valid padding`. However, some practitioners opt to use `same padding` where the padding values are obtained by image reflection at the border.

Image credit

Whereas [Long et al.](#) (FCN paper) reported that data augmentation ("randomly mirroring and "jittering" the images by translating them up to 32 pixels") did not result in a noticeable improvement in performance, [Ronneberger et al.](#) (U-Net paper) credit data augmentations ("random elastic deformations of the training samples") as a key concept for learning. It appears as if **the usefulness (and type) of data augmentation depends on the problem domain.**

Advanced U-Net variants

The standard U-Net model consists of a series of convolution operations for each "block" in the architecture. As I discussed in my post on [common convolutional network architectures](#), there exist a number of more advanced "blocks" that can be substituted in for stacked convolutional layers.

[Drozdzal et al.](#) swap out the basic stacked convolution blocks in favor of

You've successfully subscribed to Jeremy Jordan!

the corresponding feature maps of encoder and decoder modules) found in the standard U-Net structure. They report that the short skip connections

Jeremy Jordan – An overview of semantic image segmentation.

allow for faster convergence when training and allow for deeper models to be trained. Expanding on this, Jegou et al. proposed the use of **dense blocks**, still following a U-Net structure, arguing that the "characteristics of DenseNets make them a very good fit for semantic segmentation as they *naturally induce skip connections and multi-scale supervision.*" These dense blocks are useful as they carry low level features from previous layers directly alongside higher level features from more recent layers, allowing for highly efficient feature reuse.

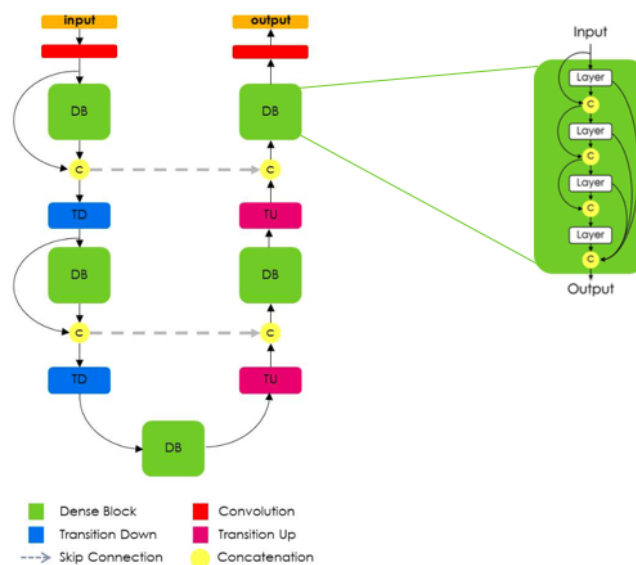


Image credit (with modification)

One very important aspect of this architecture is the fact that the upsampling path *does not* have a skip connection between the input and output of a dense block. The authors note that because the "upsampling path *increases* the feature maps spatial resolution, the linear growth in the number of features would be too memory demanding." Thus, only the

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.



The FC-DenseNet103 model achieves state of the art results (Oct 2017) on the CamVid dataset.

Dilated/atrous convolutions

One benefit of downsampling a feature map is that it *broadens the receptive field* (with respect to the input) for the following filter, given a constant filter size. Recall that this approach is more desirable than increasing the filter size due to the parameter inefficiency of large filters (discussed here in Section 3.1). However, this broader context comes at the cost of reduced spatial resolution.

Dilated convolutions provide alternative approach towards gaining a wide field of view while preserving the full spatial dimension. As shown in the figure below, the values used for a dilated convolution are spaced apart according to some specified *dilation rate*.

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.

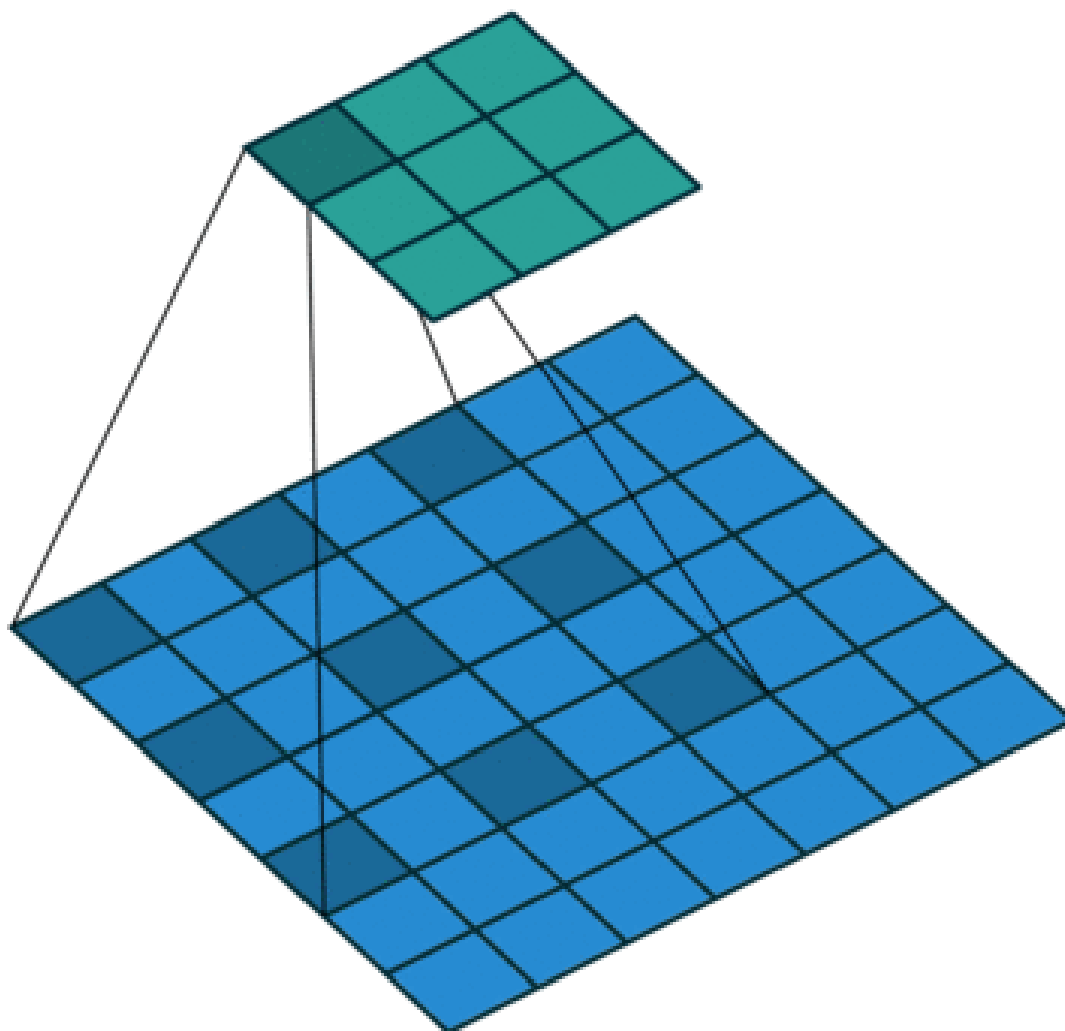


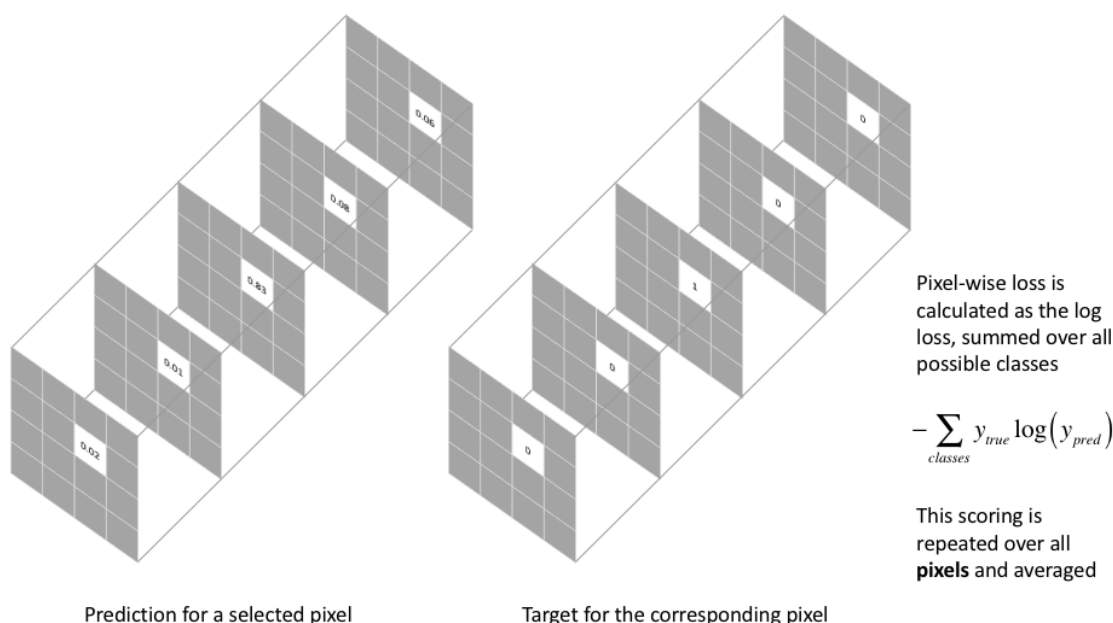
Image credit

You've successfully subscribed to Jeremy Jordan!

convolutions with successively higher dilation rates to maintain the same field of view while preventing loss of spatial detail. However, it is often still

Defining a loss function

The most commonly used loss function for the task of image segmentation is a **pixel-wise cross entropy loss**. This loss examines *each pixel individually*, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector.



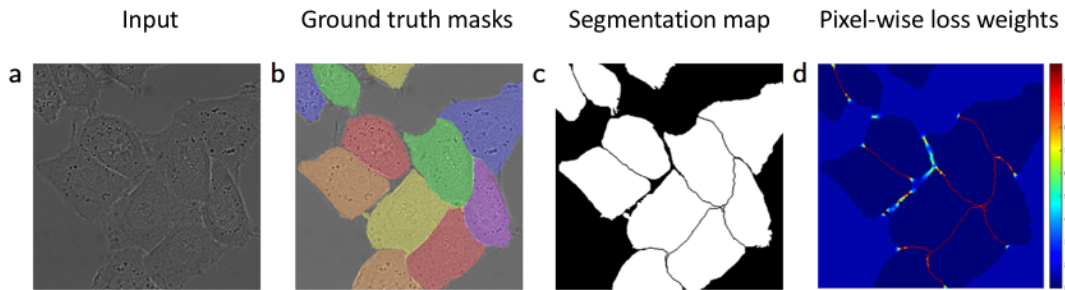
Because the cross entropy loss evaluates the class predictions for each pixel vector individually and then averages over all pixels, we're essentially asserting equal learning to each pixel in the image. This can be a problem if your various classes have unbalanced representation in the image, as training can be dominated by the most prevalent class. Long et al. (FCN

You've successfully subscribed to Jeremy Jordan!

Meanwhile, Ronneberger et al. (U-Net paper) discuss a loss weighting

Jeremy Jordan – An overview of semantic image segmentation.

segment cells in biomedical images in a *discontinuous* fashion such that individual cells may be easily identified within the binary segmentation map.



Notice how the binary segmentation map produces clear borders around the cells. [\(Source\)](#)

Another popular loss function for image segmentation tasks is based on the Dice coefficient, which is essentially a measure of overlap between two samples. This measure ranges from 0 to 1 where a Dice coefficient of 1 denotes perfect and complete overlap. The Dice coefficient was originally developed for binary data, and can be calculated as:

$$Dice = \frac{2 |A \cap B|}{|A| + |B|}$$

where $|A \cap B|$ represents the common elements between sets A and B, and

You've successfully subscribed to Jeremy Jordan!

for the case of evaluating a Dice coefficient on predicted segmentation masks, we can approximate $|A \cap B|$ as the element-wise multiplication

Jeremy Jordan – An overview of semantic image segmentation.

$$|A \cap B| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \underset{\text{prediction}}{*} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \underset{\text{target}}{\xrightarrow{\text{element-wise multiply}}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.41$$

Because our target mask is binary, we effectively zero-out any pixels from our prediction which are not "activated" in the target mask. For the remaining pixels, we are essentially penalizing low-confidence predictions; a higher value for this expression, which is in the numerator, leads to a better Dice coefficient.

In order to quantify $|A|$ and $|B|$, some researchers use the simple sum whereas other researchers prefer to use the squared sum for this calculation. I don't have the practical experience to know which performs better empirically over a wide range of tasks, so I'll leave you to try them both and see which works better.

$$|A| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.82 \quad \text{[optional squaring]} \\ |B| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{sum}} 8 \quad \text{[optional squaring]}$$

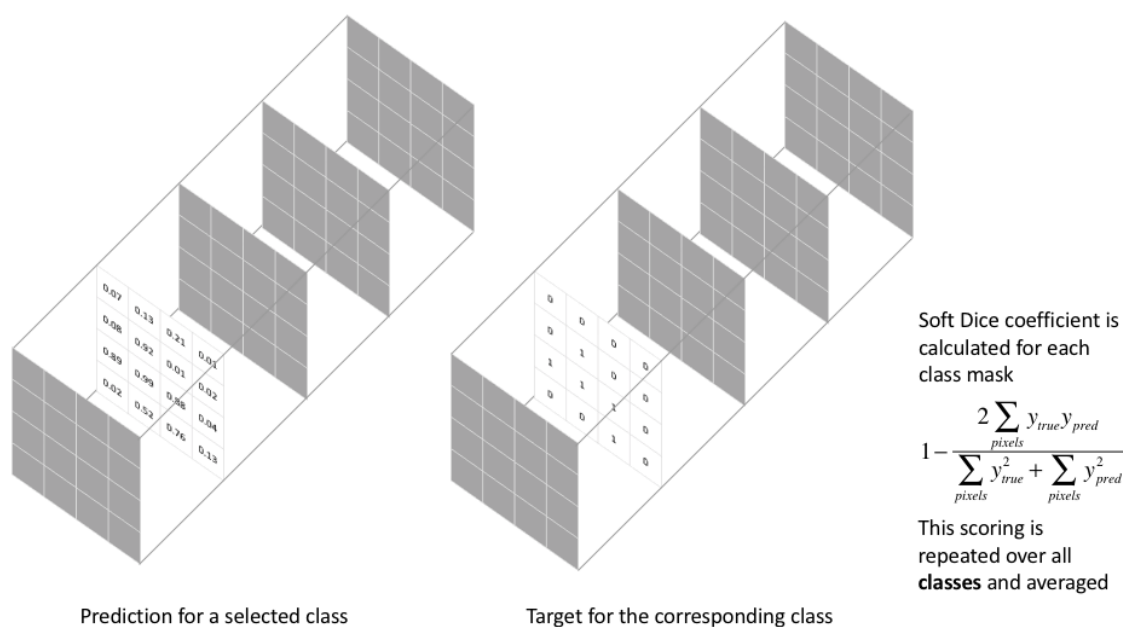
In case you were wondering, there's a 2 in the numerator in calculating the

You've successfully subscribed to Jeremy Jordan!

can be minimized, we can simply use $1 - \text{Dice}$. This loss function is known as the **soft Dice loss** because we directly use the predicted probabilities

Jeremy Jordan – An overview of semantic image segmentation.

With respect to the neural network output, the numerator is concerned with the *common activations* between our prediction and target mask, where as the denominator is concerned with the quantity of activations in each mask *separately*. This has the effect of normalizing our loss according to the size of the target mask such that the soft Dice loss does not struggle learning from classes with lesser spatial representation in an image.



A soft Dice loss is calculated for each class separately and then averaged to yield a final score. An example implementation is provided below.

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.

```
def soft_dice_loss(y_true, y_pred, epsilon=1e-6):
```

```
...
```

Soft dice loss calculation for arbitrary batch size, number of classes, and number of spatial dimensions.

Assumes the `channels_last` format.

Arguments

y_true: b x X x Y (x Z...) x c One hot encoding of ground truth

y_pred: b x X x Y (x Z...) x c Network output, must sum to 1 over c channel (such as after softmax)

epsilon: Used for numerical stability to avoid divide by zero errors

References

V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation

<https://arxiv.org/abs/1606.04797>

More details on Dice loss formulation

<https://mediatum.ub.tum.de/doc/1395260/1395260.pdf> (page 72)

Adapted from <https://github.com/Lasagne/Recipes/issues/99#issuecomment-347775022>

```
...
```

skip the batch and class axis for calculating Dice score

```
axes = tuple(range(1, len(y_pred.shape)-1))
```

```
numerator = 2. * np.sum(y_pred * y_true, axes)
```

```
denominator = np.sum(np.square(y_pred) + np.square(y_true), axes)
```

```
return 1 - np.mean((numerator + epsilon) / (denominator + epsilon)) # average over classes and batch
```

```
# thanks @mfernezir for catching a bug in an earlier version of this implementation!
```

[ce_loss.py](#) hosted with ❤ by [GitHub](#)

You've successfully subscribed to Jeremy Jordan!

Common datasets and segmentation competitions

Below, I've listed a number of common datasets that researchers use to train new models and benchmark against the state of the art. You can also explore previous Kaggle competitions and read about how winning solutions implemented segmentation models for their given task.

Datasets

- [PASCAL VOC 2012 Segmentation Competition](#)
- [COCO 2018 Stuff Segmentation Task](#)
- [BDD100K: A Large-scale Diverse Driving Video Database](#)
- [Cambridge-driving Labeled Video Database \(CamVid\)](#)
- [Cityscapes Dataset](#)
- [Mapillary Vistas Dataset](#)
- [ApolloScape Scene Parsing](#)

Past Kaggle Competitions

- [2018 Data Science Bowl](#)
 - Read about the [first place solution.](#)
- [Carvana Image Masking Challenge](#)
 - Read about the [first place solution.](#)

You've successfully subscribed to Jeremy Jordan!

Jeremy Jordan – An overview of semantic image segmentation.

Further Reading

Papers

- [Fully Convolutional Networks for Semantic Segmentation](#)
- [U-Net: Convolutional Networks for Biomedical Image Segmentation](#)
- [The Importance of Skip Connections in Biomedical Image Segmentation](#)
- [The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation](#)
- [Multi-Scale Context Aggregation by Dilated Convolutions](#)
- [DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs](#)
- [Rethinking Atrous Convolution for Semantic Image Segmentation](#)
- [Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images](#)

Lectures

- [Stanford CS231n: Detection and Segmentation](#)
 - [Lecture Slides](#)

Blog posts

- [Mat Kelcey's \(Twitter Famous\) Bee Detector](#)

You've successfully subscribed to Jeremy Jordan!

- [U-Net Case Study: Data Science Bowl 2018](#)

Jeremy Jordan – An overview of semantic image segmentation.

Image labeling tools

- [labelme: Image Polygonal Annotation with Python](#)

Useful Github repos

- [Pytorch implementations](#)



15

Subscribe to Jeremy Jordan

Get the latest posts delivered right to your inbox

Subscribe

MORE IN DATA SCIENCE

A simple solution for monitoring ML systems.

2 Jan 2021 – 10 min read

Effective testing for machine learning systems.

19 Aug 2020 – 9 min read

You've successfully subscribed to Jeremy Jordan!

[See all 47 posts →](#)

Jeremy Jordan – An overview of semantic image segmentation.

DATA SCIENCE

Evaluating image segmentation models.

When evaluating a standard machine learning model, we usually classify our predictions into four categories: true positives, false positives, true negatives, and false negatives. However, for the dense prediction task of image segmentation, it's not immediately clear what counts as a "true positive&



JEREMY JORDAN

30 MAY 2018 • 5 MIN READ

STARTUPS

Lessons learned from attempting to launch a startup.

In Q4 of 2017, I made the decision to walk down the entrepreneurial path and dedicate a full-time effort towards launching a startup venture. I secured a healthy seed round of funding from a local angel investor and recruited three of my peers to



JEREMY JORDAN

9 MAY 2018 • 5 MIN READ

Jeremy Jordan © 2021

[Latest Posts](#) [Twitter](#) [Ghost](#)