Metrics to Evaluate your Semantic Segmentation Model

How do you know your segmentation model performs well? Find out here.

Ekin Tiu Aug 10, 2019 · 7 min read ★

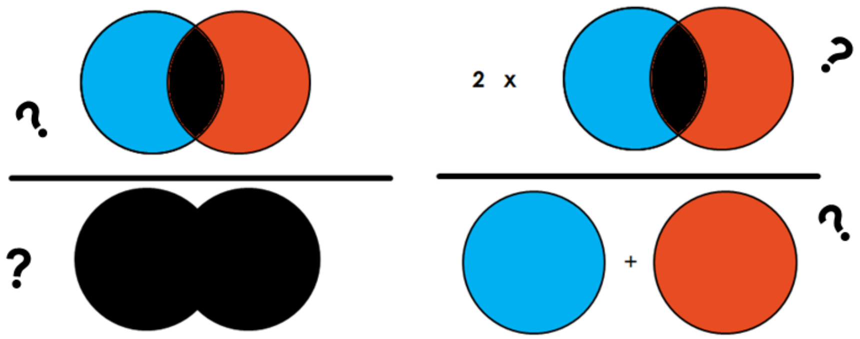


Illustration of IoU and Dice Coefficient.

Semantic segmentation. My absolute favorite task. I would make a deep learning model, have it all nice and trained… but wait. How do I know my model is performing well? In other words, what are the most common metrics for semantic segmentation? Here's a clear cut guide to the **essential metrics** that you need to know to ensure your model performs well. I have also included Keras implementations below.

If you want to learn more about Semantic Segmentation with Deep Learning, check out this Medium article by George Seif.

A guide and code

towardsdatascience.com

Contents:

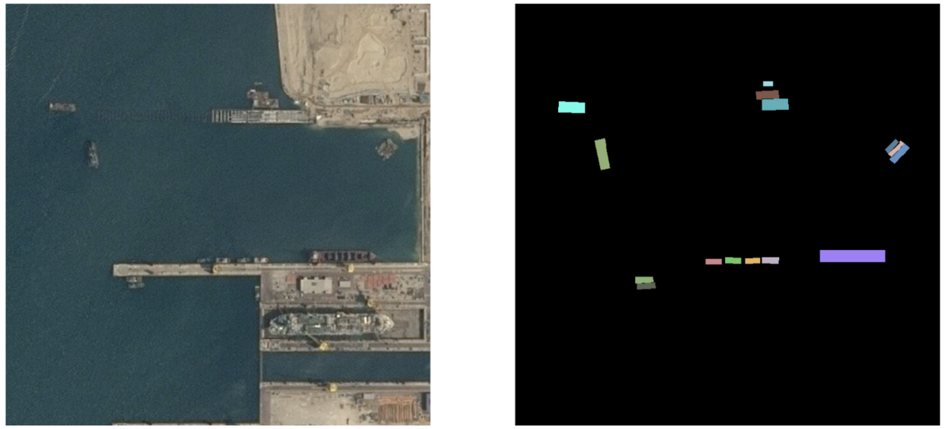
1. Pixel Accuracy
2. Intersection-Over-Union (Jaccard Index)
3. Dice Coefficient (F1 Score)
4. Conclusion, Notes, Summary

1. Pixel Accuracy

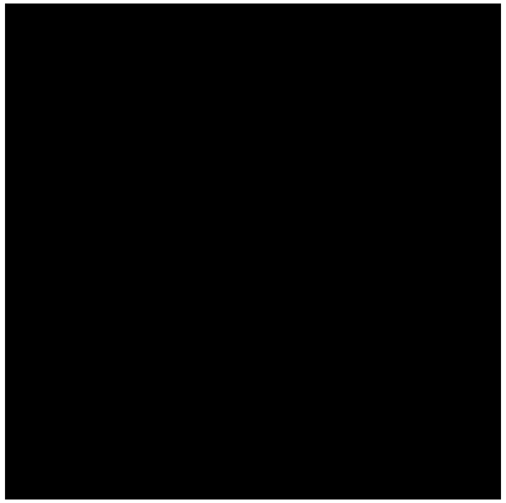
Pixel accuracy is perhaps the easiest to understand conceptually. *It is the percent of pixels in your image that are classified correctly.*

While it is easy to understand, it is in no way the best metric.

At first glance, it might be difficult to see the issue with this metric. To expose this metric for what it really is, here's a scenario: Let's say you ran the following image(*Left*)through your segmentation model. The image on the right is the ground truth, or annotation (what the model is supposed to segment). In this case, our model is trying to segment ships in a satellite image.



You see that your segmentation accuracy is **95%**. That's awesome! Let's see how your segmentation looks like!

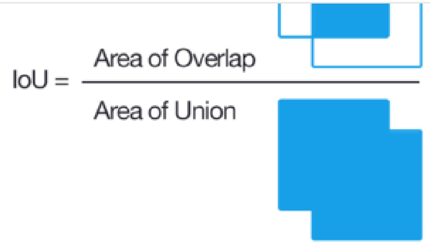


Not exactly what you were hoping for, huh. Is there something wrong with our calculation? Nope. It's exactly right. It's just that one class was 95% of the original image. So if the model classifies all pixels as that class, 95% of pixels are classified accurately while the other 5% are not. As a result, although your accuracy is a whopping 95%, your model is returning a completely useless prediction. This is meant to illustrate that high pixel accuracy doesn't always imply superior segmentation ability.

This issue is called **class imbalance**. When our classes are extremely imbalanced, it means that a class or some classes dominate the image, while some other classes make up only a small portion of the image. Unfortunately, class imbalance is prevalent in many real world data sets, so it can't be ignored. Therefore, I present to you two alternative metrics that are better at dealing with this issue:

2. Intersection-Over-Union (IoU, Jaccard Index)

The Intersection-Over-Union (IoU), also known as the Jaccard Index, is one of the most commonly used metrics in semantic segmentation… and for good reason. The IoU is a very straightforward metric that's extremely effective.



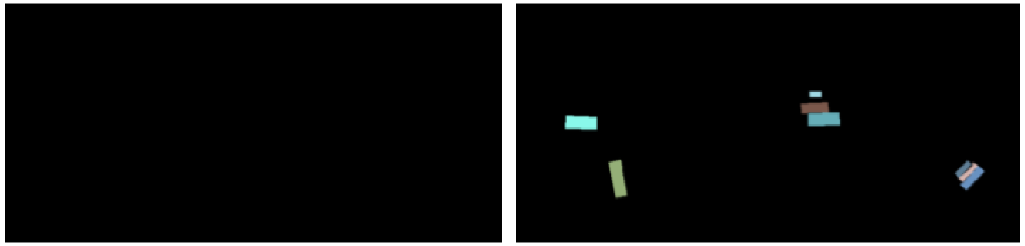
IoU calculation visualized. Source: Wikipedia

Before reading the following statement, take a look at the image to the left. Simply put, the **IoU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth**, as shown on the image to the left. This metric ranges from 0–1 (0–100%) with 0 signifying no overlap and 1 signifying perfectly overlapping segmentation.

For **binary** (two classes) or **multi-class segmentation**, the mean IoU of the image is calculated by **taking the IoU of each class and averaging them**. (It's implemented slightly differently in code).

Now let's try to understand why this metric is better than pixel accuracy by using the same scenario as section 1. For the sake of simplicity, let's assume all the ships (colored boxes) are part of the same class.

But wait, *what exactly is overlap and union in our context?* The image above shows a pretty clear picture, but I found it a bit difficult to understand in the context of predicted vs. ground truth because they aren't necessarily overlapping like the image depicts above. Let's take a look at the predicted segmentation and the ground truth side-by-side.



Predicted segmentation (Left) Ground truth annotation (Right)

Let's calculate the ship IoU first. We assume the total area of the image is 100 (100 pixels). First, let's think about the ships' overlap. We can pretend that we move the predicted segmentation (left) directly above the ground truth (right), and see if there are any ship pixels that overlap. Since there are no pixels that are classified as ships by the model, there are 0 overlapping ship pixels.

Union consists of all of the pixels classified as ships from **both** images, minus the overlap/intersection. In this case, there are 5 pixels (this is an arbitrary number choice) that are classified as ships total. Subtract the overlap/intersection which is 0 to get 5 as the area of union. After doing the calculations, we learn that the IoU is merely **47.5%!** See the calculation below.

Here is the detailed calculation:

Ships: Area of Overlap = 0, Area of Union = (5+0)-0 =5

Area of Overlap/Area of Union = 0%

Now for the black background, we do the same thing.

Background: Area of Overlap = 95, Area of Union = (95+100)–95 = 100

Area of Overlap/Area of Union =95%

Mean IoU = (Ships + Background)/2 = (0%+95%)/2 = **47.5%**

Wow. That's a lot lower than the 95% pixel accuracy we calculated. It is obvious that 47.5 is a much better indication of the success of our segmentation, or more appropriately, the lack thereof.

Here is a great Keras implementation that I used in my own projects:

```
def iou_coef(y_true, y_pred, smooth=1):
  intersection = K.sum(K.abs(y_true * y_pred), axis=[1,2,3])
  union = K.sum(y_true,[1,2,3])+K.sum(y_pred,[1,2,3])-intersection
  iou = K.mean((intersection + smooth) / (union + smooth), axis=0)
  return iou
```

Both y_true and y_pred are **m** x **r** x **c** x **n** where m is the number of images, r is the number of rows, c is the number of columns, and n is the number of classes.

3. Dice Coefficient (F1 Score)

Simply put, the **Dice Coefficient is 2 * the Area of Overlap divided by the total number of pixels in both images.** (See explanation of area of union in section 2).

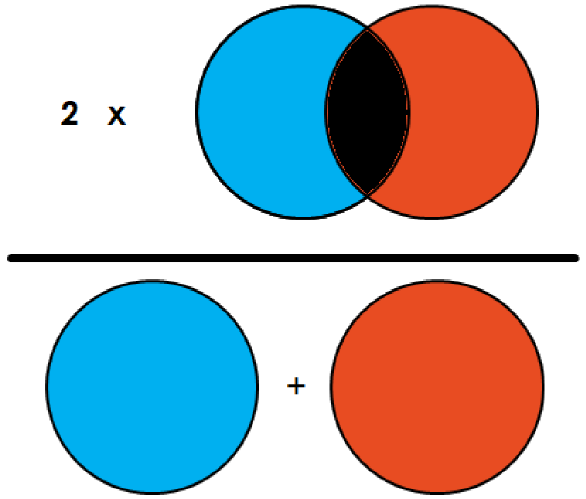


Illustration of Dice Coefficient. 2xOverlap/Total number of pixels

So for the same scenario used in 1 and 2, we would perform the following calculations:

Total Number of Pixels for both images combined = 200

Ships: Area of Overlap = 0

*(2 * Area of Overlap)/(total pixels combined) = 0/200 = 0*

(2 * Area of Overlap)/(total pixels combined) = 95*2/200 = 0.95

Dice = (Ships + Background)/2 = (0%+95%)/2 = **47.5%**

In this case, we got the same value as the IoU, but this will not always be the case.

The Dice coefficient is very similar to the IoU. They are positively correlated, meaning if one says model A is better than model B at segmenting an image, then the other will say the same. Like the IoU, they both range from 0 to 1, with 1 signifying the greatest similarity between predicted and truth.

To better understand the differences between them, I recommend reading the following Stack Exchange answer:

F1/Dice-Score vs IoU

begingroup$ You're on the right track. So a few things right off the bat. From the definition of the two metrics, we...

stats.stackexchange.com

Here's an implementation for the Dice Coefficient with the same input conditions specified in section 2:

```
def dice_coef(y_true, y_pred, smooth=1):
  intersection = K.sum(y_true * y_pred, axis=[1,2,3])
  union = K.sum(y_true, axis=[1,2,3]) + K.sum(y_pred, axis=[1,2,3])
  dice = K.mean((2. * intersection + smooth)/(union + smooth),
axis=0)
  return dice
```

4. Conclusion, Notes, Summary

In conclusion, the most commonly used metrics for semantic segmentation are the IoU and the Dice Coefficient. I have included code implementations in Keras, and will explain them in greater depth in an upcoming article.

As some of you may have noticed, I have purposely excluded discussions and explanations regarding true positives, true negatives, false positives and false

I hope this article imparts a greater insight about each metric so that you may have a deeper understanding during implementation or when reading literature.

Machine Learning   Segmentation   Metrics   Evaluation   Deep Learning