Twitter Thread by Vladimir Haltakov

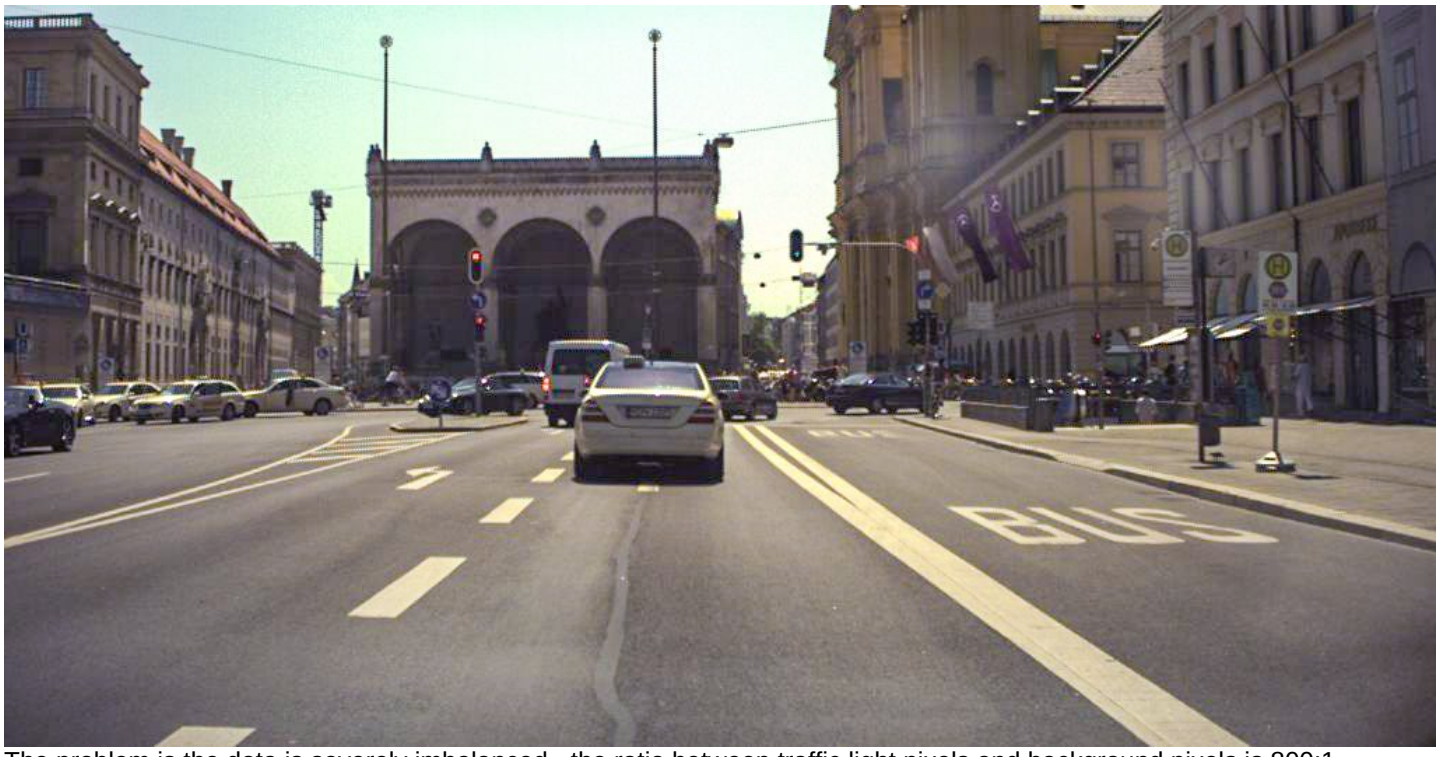
Vladimir Haltakov
@haltakov

Let's talk about a common problem in ML - imbalanced data ■■

Imagine we want to detect all pixels belonging to a traffic light from a self-driving car's camera. We train a model with 99.88% performance. Pretty cool, right?

Actually, this model is useless ■

Let me explain ■



The problem is the data is severely imbalanced - the ratio between traffic light pixels and background pixels is 800:1.

If we don't take any measures, our model will learn to classify each pixel as background giving us 99.88% accuracy. But it's useless!

What can we do? ■

Let me tell you about 3 ways of dealing with imbalanced data:

■■ Choose the right evaluation metric
■■ Undersampling your dataset
■■ Oversampling your dataset
■■ Adapting the loss

Let's dive in ■

1■■ Evaluation metrics

Looking at the overall accuracy is a very bad idea when dealing with imbalanced data. There are other measures that are much better suited:
■■ Precision
■■ Recall
■■ F1 score

I wrote a whole thread on that

https://t.co/MORy3Wehlu

> How to evaluate your ML model? \U0001f4cf
>
> Your accuracy is 97%, so this is pretty good, right? Right? No! \u274c
>
> Just looking at the model accuracy is not enough. Let me tell you about some other metrics:
> \u25aa\ufe0f Recall
> \u25aa\ufe0f Precision
> \u25aa\ufe0f F1 score
> \u25aa\ufe0f Confusion matrix
>
> Let's start \U0001f447
>
> — Vladimir Haltakov (@haltakov) August 31, 2021

2■■ Undersampling

The idea is to throw away samples of the overrepresented classes.

One way to do this is to randomly throw away samples. However, ideally, we want to make sure we are only throwing away samples that look similar.

Here is a strategy to achieve that ■



Clever Undersampling

■■ Compute image features for each sample using a pre-trained CNN
■■ Cluster images by visual appearance using k-means, DBSCAN etc.
■■ Remove similar samples from the clusters (check out for example the Near-Miss or the Tomek Links strategies)

■

3■■ Oversampling

The idea here is to generate new samples from underrepresented classes. The easiest way to do this is of course to repeat the samples. However, we are not gaining any new information with this.

Some better strategies ■

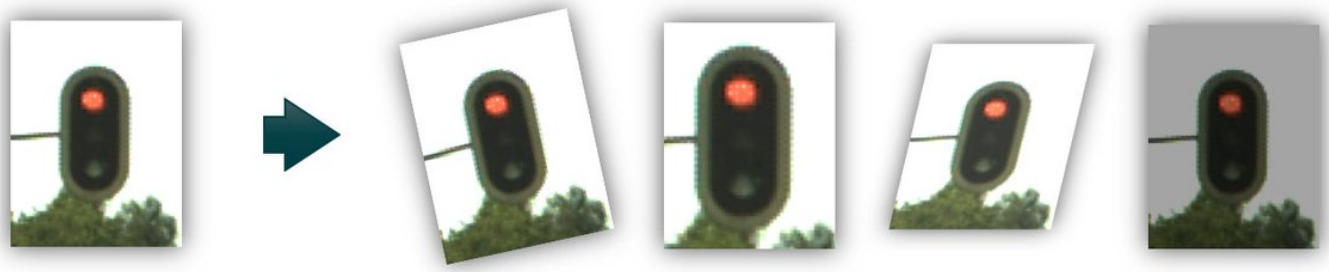


Data Augmentation

Create new samples by modifying the existing ones. You can apply many different transformations like for example:

■■ Rotation
■■ Flipping
■■ Zooming
■■ Skewing
■■ Color changing

We can do some even more advanced stuff ■



SMOTE

The idea is to create new samples by combining two existing ones.

This technique is more common when working with tabular data, but can be used for images as well. For that, we can combine the images in feature space and reconstruct them using an autoencoder.

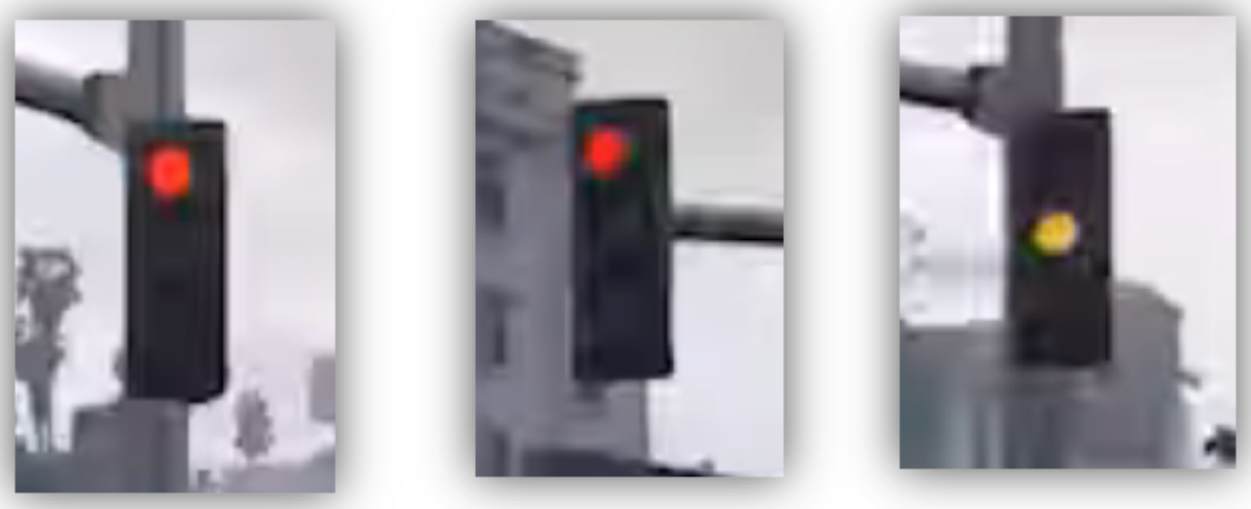


Synthetic Data

Another option is to generate synthetic data to add to our dataset. This can be done either using a GAN or using a realistic simulation to render new images.

There are even companies that specialize in this, like https://t.co/nOGdc0cNBb (not affiliated)

■



4■■ Adapting the loss function

Finally, an easy way to improve the balance is directly in your loss function. We can specify that samples of the underrepresented class to have more weight and contribute more to the loss function.

Here is an example of how to do it in the code.

```
# PyTorch - set the weight variable
torch.nn.CrossEntropyLoss(weight=None, ...)

# TensorFlow - set the class_weight variable
tf.keras.Model.fit(class_weight=None, ...)
```

So, let's recap the main ideas when dealing with imbalanced data:

■■ Make sure you are using the right evaluation metric
■■ Use undersampling and oversampling techniques to improve your dataset
■■ Use class weights in your loss function

This week I'm reposting some of my best threads from the past months, so I can focus on creating my machine learning course.

Next week I'm back with some new content on machine learning and web3, so make sure you follow me @haltakov.