

# How do I design a visual deep learning system in 2019?

An introductory exploration for curious people



Anthony Sarkis [Follow](#)

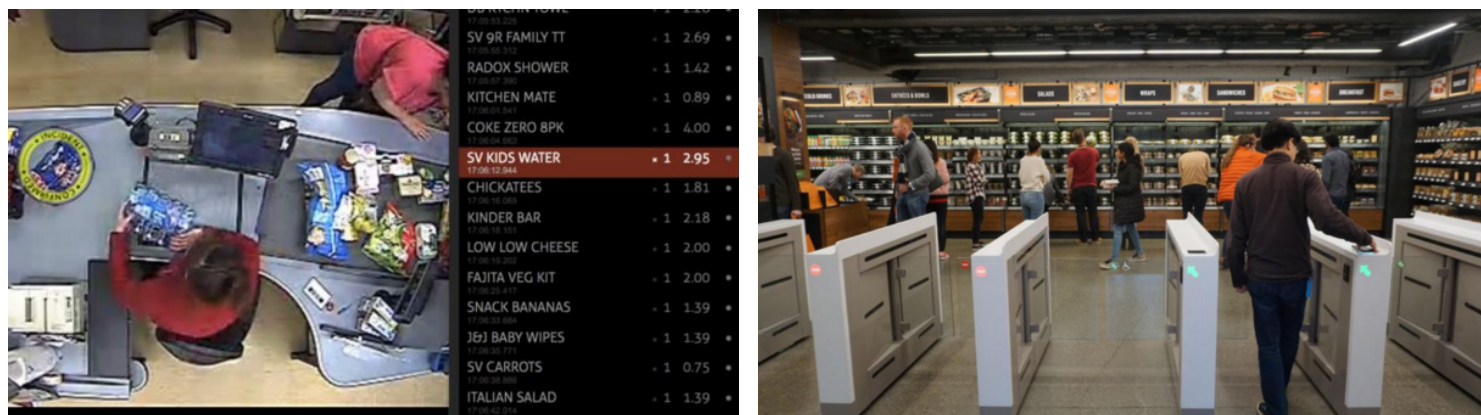
Mar 4 · 9 min read

Deep learning is one of the most important technical concepts of our time. There are already many general interest pieces and resources covering specific technical areas. Yet, a mildly technical introductory review of practical system and data design felt missing. Here I review that with a focus on unpacking often unstated assumptions. Enjoy! :)

## Part I—System design

**Are we integrating into an existing process, or creating a new process around this technology?**

While this could be asked of any new system, I think it's especially relevant for deep learning.



Left, approach example of "Integration", Right, example of "New Process"

Let's consider an example to guide our thoughts on this. Compare integrating deep learning into a checkout line, as opposed to creating a


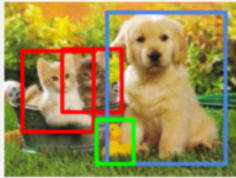

new type of store designed around deep learning, where there is no checkout.

Either approach may be appropriate, but it's worth pausing to consider which one is being pursued.

## Are we using the appropriate deep learning method?

*A survey of popular methods for vision tasks.*

We will explore full image classification, object detection, and segmentation.

	Classification	Object Detection	Instance Segmentation
			
	CAT	CAT, DOG, DUCK	CAT, DOG, DUCK
Compute Cost	🔥	🔥🔥	🔥🔥🔥🔥
Annotation Effort	✎	✎✎	✎✎✎✎✎
Location	✗	✓	✓✓
Research completeness	📖📖📖	📖📖	📖
Complexity of output	💎	💎💎	💎💎💎💎

Source: own work, except header images from Stanford CS231n

## Full Image Classification

This is the least expensive to setup, has been researched the most, and has the easiest output to deal with. Usually the final output is single string, *cat*. It's a building block for other methods. *Given how well covered and researched it is I won't spend any more time on it for the scope of this article, see appendix for resources on this.*

## Object detection

**Location** is the key difference that separates it from full image. The implicit assumption is that the object is also classified once detected. Object detection takes more work and has a more complex output to deal with.

The internal representation of the location is often quite complex<sup>5</sup>. Usually the final output of the location is expressed as the minimum information need to draw a box. Either as the (top left point x,y) and (width, height), or as an (x\_min, y\_min) and (x\_max, y\_max) pair of points.

Another assumption is that we want locate all objects in the image. So the combined output is a list with coordinates and a class for each. Note the algorithm may have an effective limit of say 100 objects. Generally medium to large objects can be detected disproportionately better than small objects<sup>8</sup>.

To illustrate the difference location makes consider this image:



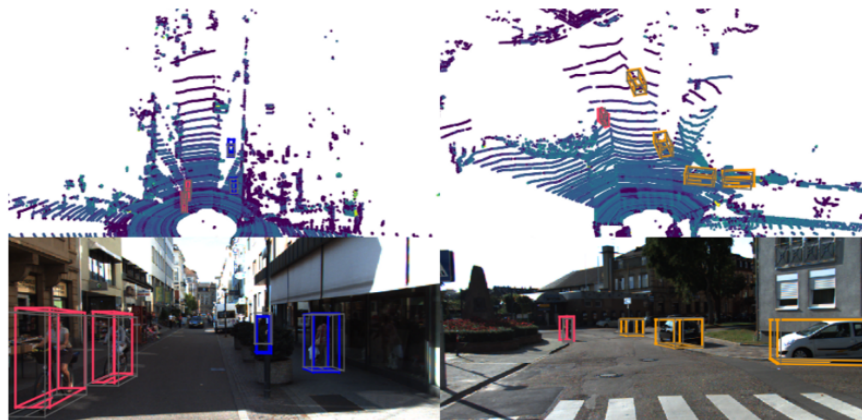
Source: real time traffic light detection<sup>2</sup>

Here the traffic light of the lane we are in is different from the other lanes, as well as the red light in the distance is further from the closer lights. It would be hard to get a useful output out of a full image classification here.

Benefits of location information:

- The object of interest does **not need to be the focus of the image**.
- **Context**, which can be used to do further processing such as, making measurements.
- Can perform classification on **multiple** objects from one image.

Within object detection you have two main choices, “one stage” or “two stage” methods. One stage directly guesses the location and two stage which makes a bunch of guesses and then refines them<sup>6</sup>. Naturally the method that does more work is generally more accurate and slower. *See references in appendix for information beyond the scope of this article.*



Source: PointPillars, Lang et. al. <https://arxiv.org/pdf/1812.05784.pdf>

A somewhat related method is Cuboid detection, for example in [LiDAR scans](#) shown above.

If you are starting a new project and aren't reasonably sure which method fits best, object detection is the current “default” choice for cost / performance trade off.

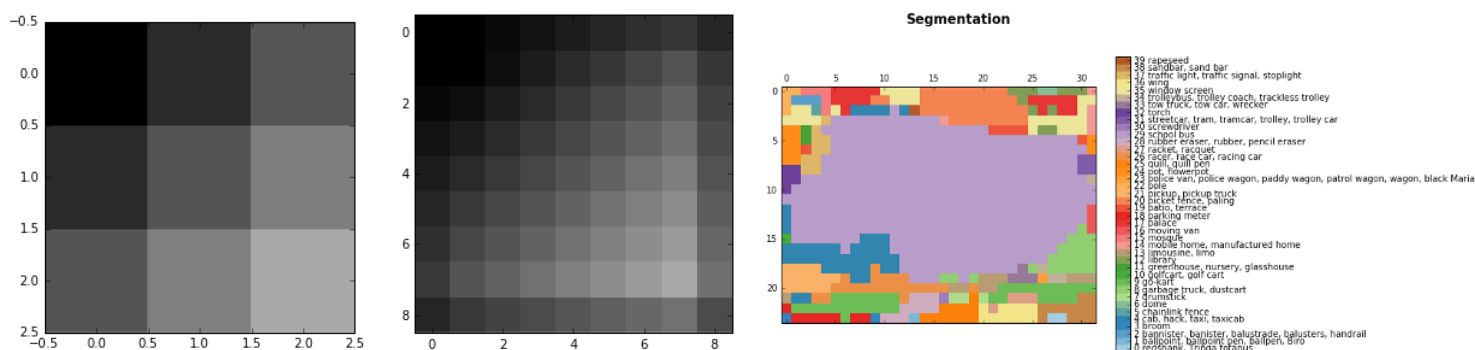
## (Semantic) Segmentation

The most effort of the three.

Research in this area is exciting but it is still an open question. The output is a “dense” pixel mask ie for every pixel in the image, it’s assigned a class. Given this density of information usually more post processing is required for useful interpretation.

The current go to method here is Mask R-CNN<sup>4</sup>, which combines object detection and segmentation.

The key component of segmentation is the use of an up sampling technique, like a transpose operator like `conv2d_transpose_in` tensorflow. The images below give a general stylistic idea, see source for in depth treatment.



Source: Upsampling and Image Segmentation with Tensorflow and TF-Slim by Daniil Pakhomov. As an indication of how much there is to be researched, the community has yet to collectively agree on a name for this technique, hence the (semantic) segmentation. If you think that’s silly try bringing up a “deconvolution” there’s even a whole paper about one aspect of it.

Sub variants include instance segmentation and panoptic segmentation.

## Specialized methods:

Other visual methods are usually less popular and/or more specialized. For example key points is fairly general, but usually is mentioned in the context of human pose and motion.



Source: Open Pose, Hidalgo<sup>3</sup>

. . .

## Part II—Data design overview

Research is trending towards making the non-data portion of deep learning system design easier. Given that a deep learning network learns from data, this puts pressure on the data set creation and validation.

In research settings the goal is usually to constrain the dataset in such a way so as to test the algorithm. In production use cases there is more freedom to engineer the data.

Data design is a complex topic. Here we will cover 2 assumptions and a concept called Dynamic Data Design.

### Assumptions

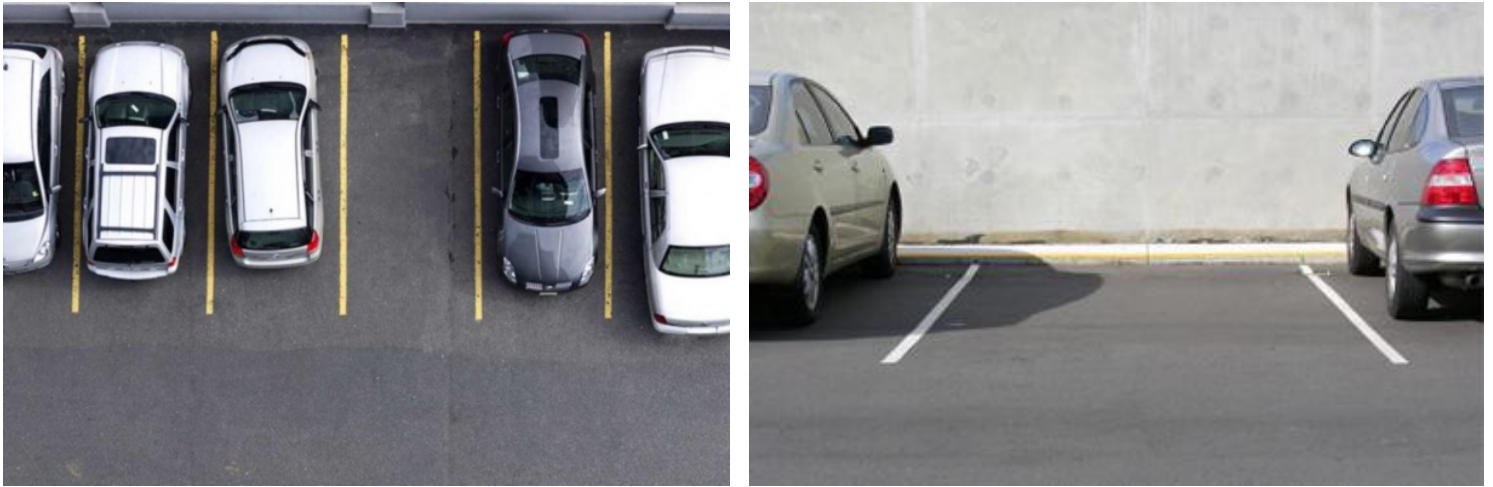
**“Assumption: We can just find an existing one”**

A deep learning based system works when it’s prediction data (or “domain”) is similar to it’s training data. This is probably the most common expectation / reality gap. Plan for it!

Imagine designing a system to detect open parking spots. These two perspectives are very different. If you are planning to use a birds eye view like the image to the left in production, your training data should



be from a similar view point. Using the image on the right for training would be unlikely to help and may make the network worse!

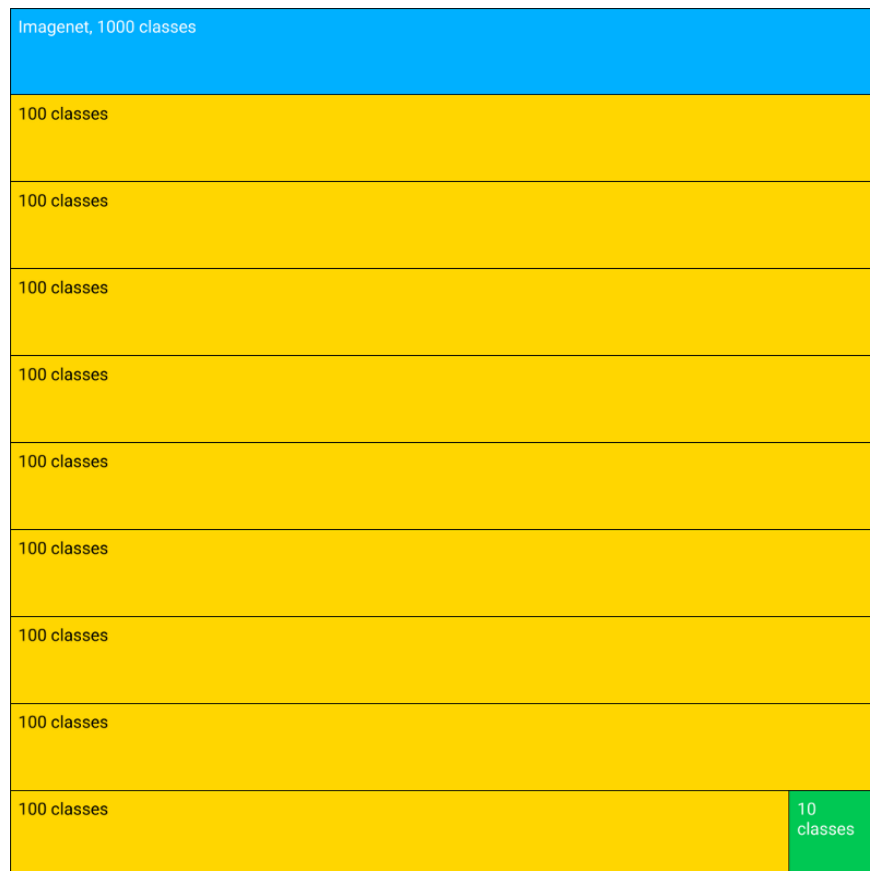


If the left is your training data and right is your use case you are in trouble!

***“Assumption: It’s expensive create our own”***

Imagenet has more than 20,000<sup>1</sup> categories, and the standard challenge benchmark “trims” it to 1,000 classes. It would be very expensive to create your own.

However! We are looking **use the technology**, not explore it’s frontiers in research. For a production use case how many classes do you need? Probably orders of magnitude less than 1,000. 10 classes is likely more than enough to reasonably represent an average problem.



To scale illustration. 1x1 inch is 10 classes. Source: own work.

This reduction in classes, plus the improvement of the algorithms<sup>7</sup>, means it's relatively inexpensive to create a starter dataset.

## Dynamic data design (D3 principle)

One of the most common assumptions with a *dataset* is right in its name. That it's a set. That it's static. This approach is a built in assumption to research data, but doesn't really reflect the changing face of deep learning systems and exception handling. Motivation:

- Data design **choices** have a critical impact on system performance
- The domain of production data should **be expected to regularly change**<sup>10</sup>
- It's more effective to build data within the **context** of your specific deep learning system.

Here's one example of a dynamic approach:



## 1. Define optimal data in context of your deep learning system

A general rule of thumb: If a human can reasonably see it, a deep learning system can probably see it too.

- Do we need to collect new data, or do we have existing data that fits the system design?
- For integration into existing processes: What are **humans** involved in this process *literally* looking at? Not necessarily existing sensor placements as even if the process isn't changing, sensor placement may!
- For new processes: How can we setup data collection (ie camera sensors) in such a way so as to maximize value to our deep learning context?

## 2. Collect a small sample of data

The goal is to discover and/or confirm the initial limits of the system design. Let's not try to collect the whole dataset upfront! Instead collect a small sample, and then test the waters with it.

Annotate the samples (ie ~100 ) and review, with questions like:

- How well does a baseline model learn on this data?
- Knowing what we know now, could we adapt the the collection process in a way that would make deep learning and/or annotation easier?

The theme here is instead of *challenging* the deep learning system, we learn the limits and design in consideration of them.

## 3. Collect one order of magnitude more data

At this point we have a inkling of an idea of the limits of:

- Performance per class. ie Which classes need more data. Let's not assume that the data should be balanced 1:1, ie should more difficult classes have a different ratio of samples?
- Expected number of samples needed. This may have been difficult to predict at onset. Sample answers: We have already collected

enough, we likely need 2–3x more, or we need to review if this is a workable class.

- General level of model performance. If you have collected over 200 samples for an object detection problem, and still aren't getting designed performance, it's likely worth reconsidering the definition of the class.

Throughout this process we are looking for directional change. For example, if you add 100 examples and the model does not seem to improve, perhaps this is going in the wrong direction. Around this point you can turn on active labeling approaches.

A reminder here that strategy is more about **variety** than volume. More of the same won't provide much model performance improvement.

*Repeat as needed.*

## One last thing—exception handling question

Exception handling for deep learning systems is an open area of research. One general directional thing to cover is the assumption:

**“Deep learning systems aren't ready for prime time, they don't work all the time.”**

When's the last time you ran into a computer bug? Something didn't quite work right, or some glitch? It's pretty common. Not just in every day software, but even mission critical systems<sup>9</sup>.

And olive jars:



Exception handling for olive pitting

The questions should really be: In the context of the overall system design, how good does the deep learning part need to be to be useful? Could we modify the overall system design to better support deep learning?

Thanks for reading!

. . .

Are you interested in, or already working with deep learning systems? At Diffgram we are building infrastructure to power the next generation of real world applications. [Try it out](#) and/or [consider joining us](#). :)

. . .

## Resources

1, <https://www.nytimes.com/2012/11/20/science/for-web-images-creating-new-technology-to-see-and-find.html>

2 [https://medium.com/@anthony\\_sarkis/self-driving-cars-implementing-real-time-traffic-light-detection-and-classification-in-2017-7d9ae8df1c58](https://medium.com/@anthony_sarkis/self-driving-cars-implementing-real-time-traffic-light-detection-and-classification-in-2017-7d9ae8df1c58)

3 <https://github.com/CMU-Perceptual-Computing-Lab/openpose>

4 <https://arxiv.org/abs/1703.06870>

5 Image from: <https://arxiv.org/pdf/1902.07296v1.pdf>

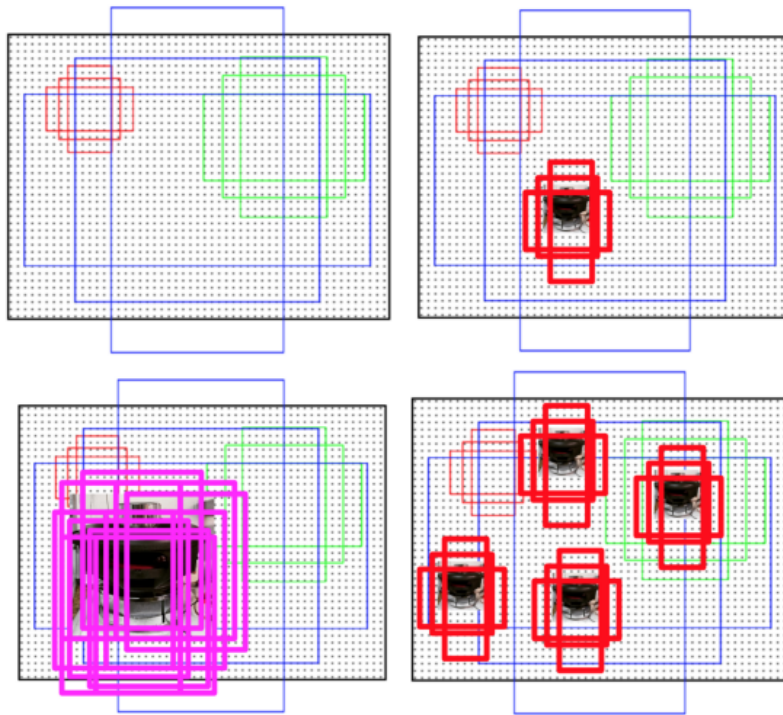


Fig. 4: Schematic illustration of anchors of different scales matching the ground truth objects. Small objects have much less anchors matched. To overcome the problem, we propose to artificially augment the images by copy-pasting small objects, so that there is more anchors positively matched with small objects during training.

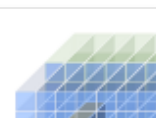
Kisantant et al <https://arxiv.org/pdf/1902.07296v1.pdf>

Showing example of complex representation of “anchors” in network (not final output).

6 Some ROI articles

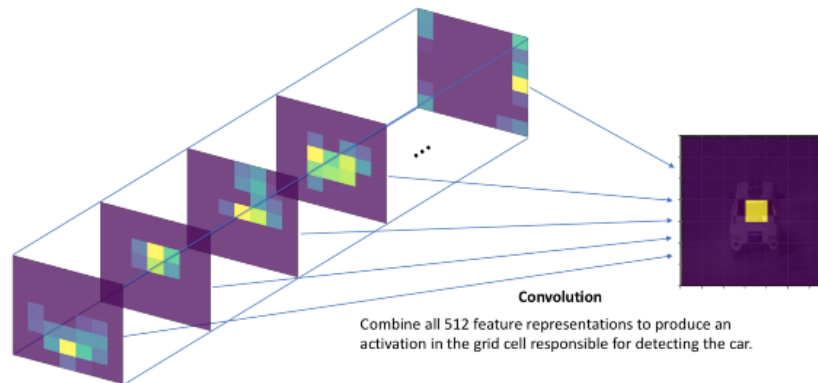
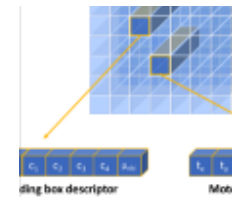
Optimizing the Trade-off between  
Single-Stage and Two-Stage Deep Object Detectors  
using Image Difficulty Prediction  
<https://arxiv.org/pdf/1803.08707.pdf>

An overview of object detection: one-stage methods.



In this post, I'll discuss an overview of deep learning techniques for object detection using...

[www.jeremyjordan.me](http://www.jeremyjordan.me)



From <https://www.jeremyjordan.me/object-detection-one-stage/>

<- Side note this is the clearest description of a convolution operation I have seen.

7, [https://medium.com/@anthony\\_sarkis/the-age-of-the-algorithm-why-ai-progress-is-faster-than-moores-law-2fb7d5ae7943](https://medium.com/@anthony_sarkis/the-age-of-the-algorithm-why-ai-progress-is-faster-than-moores-law-2fb7d5ae7943)

8, Augmentation for small object detection  
<https://arxiv.org/pdf/1902.07296v1.pdf>

9, [https://en.wikipedia.org/wiki/List\\_of\\_software\\_bugs](https://en.wikipedia.org/wiki/List_of_software_bugs)

10, <https://machinelearningmastery.com/gentle-introduction-concept-drift-machine-learning/> <- Not a huge fan of this article but seemed to be best general primer on the topic

Full image classification resources

- <http://cs231n.github.io/classification/>
- Classification in general: <https://playground.tensorflow.org/>





