Understanding Zero-Shot Learning— Making ML More Human

Source: https://scribe.rip/understanding-zero-shot-learning-making-ml-more-human-4653ac35ccab

Understanding Zero-Shot Learning — Making ML More Human

An intuitive overview of how a model can recognize what it hasn't seen.

✍Photo by Jason Leung → https://unsplash.com/@ninjason?utm_source=medium&utm_medium=referral on Unsplash → https://unsplash.com/?utm_source=medium&utm_medium=referral

Introduction — What is Zero-Shot Learning?

> Zero-shot learning allows a model to recognize what it **hasn't seen before.**

Imagine you're tasked with designing the latest and greatest machine learning model that can classify all animals. *Yes, all animals.*

Using your machine learning knowledge, you immediately understand that we need a labeled dataset with at least one example for every single animal. There's 1,899,587 described species in the world, so you're gonna need a dataset with roughly 2 million different classes.

Yikes.

✍Animals your model has to classify. Photos on Unsplash.

As you've probably noticed by now, getting large quantities of high quality labeled data is hard. Very hard.

It doesn't help when there are a gazillion different classes (i.e. animal species) that your model has to learn.

So how do we solve this problem?

One way is to **decrease our models' reliance on labeled data**. This is the motivation behind **zero-shot learning, *in which your model learns how to classify classes that it hasn't seen before.***

In the animal species classification example, your model may be able to predict that the image on the bottom right corner is a "Panda", even though it didn't explicitly see a labeled example of a "Panda" during training.

Crazy huh?!

In the next section, we'll learn how this seemingly magical method works through some examples of models that employ a zero-shot setup.

How does Zero-Shot Learning Work?

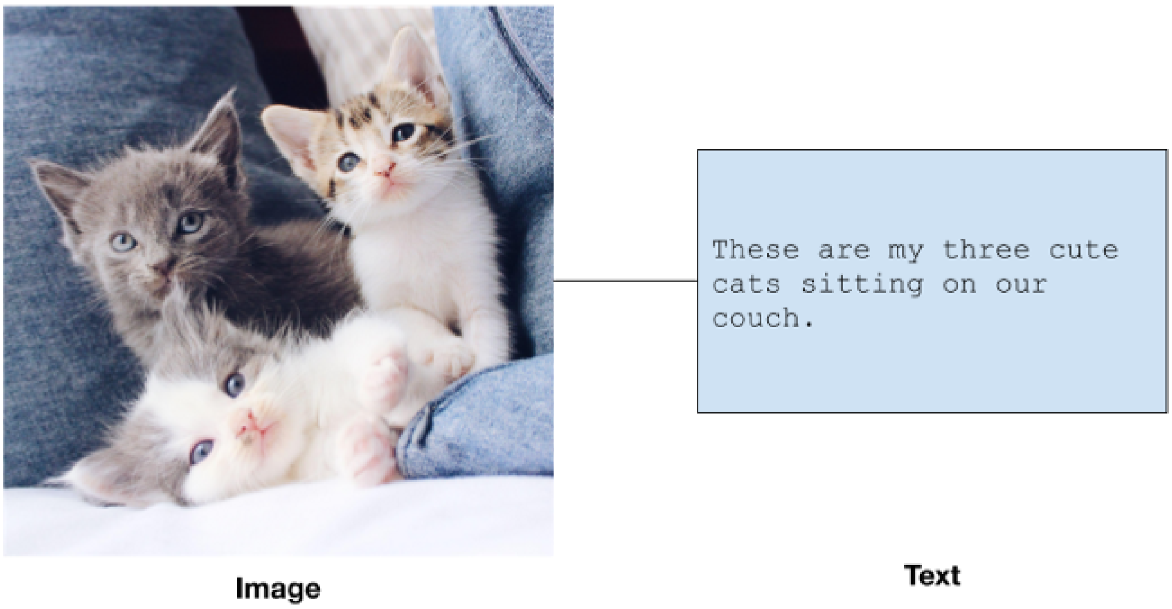
Although there are multiple approaches to zero-shot learning in literature, this article focuses on a recent method called Contrastive Language-Image Pretraining (CLIP) proposed by OpenAI that has performed well in a zero-shot setting [2] → https://arxiv.org/abs/2103.00020.

The goal of CLIP is to learn how to **classify images** without any explicit labels.

Intuition

Just like traditional supervised models, CLIP has two stages: **the training stage (learning)** and the **inference stage (making predictions)**.

In the *training stage*, CLIP learns about images by "reading" auxiliary text (i.e. sentences) corresponding to each image like in the example below.



These are my three cute cats sitting on our couch.

✍Example of a possible input to the CLIP architecture. Photo by The Lucky Neko → https://unsplash.com/@theluckyneko on Unsplash → https://unsplash.com/photos/rplhB9mYF48

As a human (assuming you've never seen a cat before), you can read this text, and probably decipher that the three *things* in the image are "cats". If you saw enough pictures of cats paired with captions with the word "cat" in it, chances are, you'd get really good at determining whether or not there are cats in an image.

Similarly, by seeing 400 million image-text pairings of different objects, the model is able to understand how certain phrases and words correspond to certain patterns in the images. Once it has this understanding, the model can use this accumulated knowledge to extrapolate to other classification tasks.

But wait a second.

You may be wondering, isn't this "auxiliary text" technically a type of label, thereby rendering this process *not* the label-free learning that I promised at the beginning?

While the auxiliary information (i.e. captions) *are* a form of supervision, they *are not labels!* Through this auxiliary information, we are able to use *information-rich* **unstructured data** instead of having to parse the unstructured data ourselves to handcraft a single label (i.e. "these are my three cute cats…" → "cats").

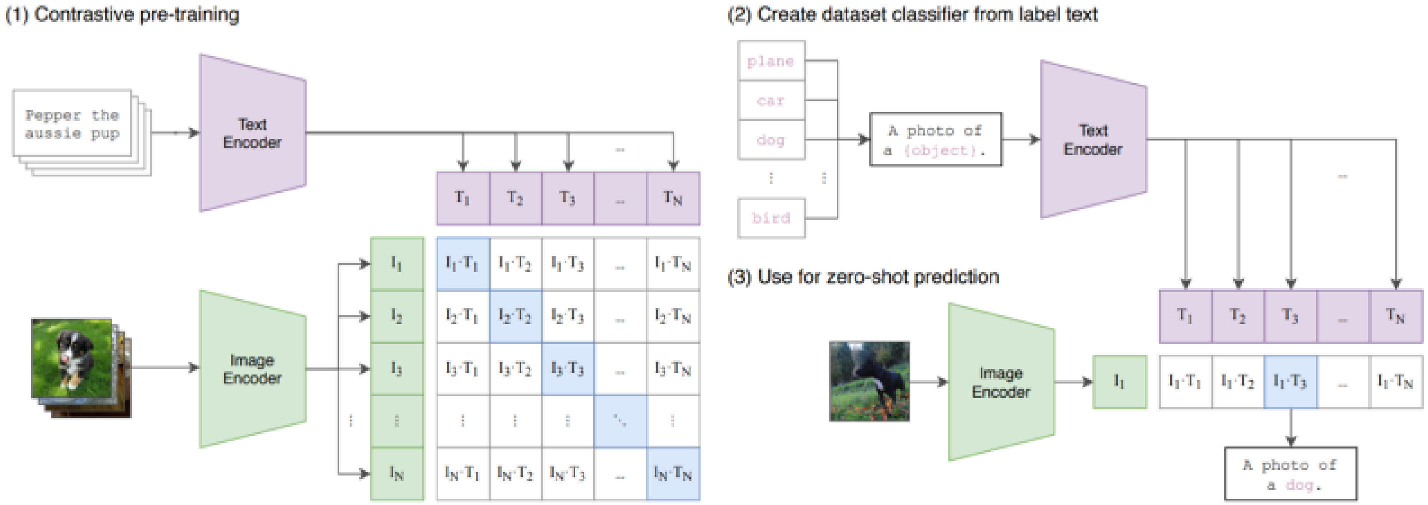
Designing a label takes time and prunes potentially useful information. By using CLIP's methodology, we get to bypass this bottleneck and maximize the amount of information the model has access to.

Diving Deeper — Zero-Shot Training

How *exactly* is the model able to learn from these auxiliary texts?

As suggested by the architecture's name, CLIP uses a technique called **contrastive learning** in order to understand the relationship between image and text pairings. If you're unfamiliar with contrastive learning, I suggest checking out this article on contrastive learning before continuing.

> **Understanding Contrastive Learning → https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607** *Learn how to learn without labels.*towardsdatascience.com → https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607


(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T₁ T₂ T₃ ... T_N
I₁ I₂ I₃ ⋮ I_N
I₁·T₁ I₁·T₂ I₁·T₃ ... I₁·T_N
I₂·T₁ I₂·T₂ I₂·T₃ ... I₂·T_N
I₃·T₁ I₃·T₂ I₃·T₃ ... I₃·T_N
I_N·T₁ I_N·T₂ I_N·T₃ ... I_N·T_N
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
T₁ T₂ T₃ ... T_N
(3) Use for zero-shot prediction
Image Encoder
I₁
I₁·T₁ I₁·T₂ I₁·T₃ ... I₁·T_N
A photo of a dog.

✍Summary of the CLIP approach. Figure from [2].

In essence, CLIP aims to *minimize the difference between the encodings of the image and it's corresponding text.* In other words, the model should learn to make the encodings of the images and the encodings of its corresponding text *as similar as possible.*

Let's break down this idea a bit more.

What are encodings? Encodings are simply lower-dimensional representations of data (green and purple boxes in the figure above). Ideally, encodings for an image or a text should represent *the most important, distinguishable, information* of that image or text respectively.

For instance, all images of cats should have *similar encodings* since they all have cats in them, while they should be *distinct* from encodings of dogs.

Notice that in this ideal world where encodings of similar objects are also similar and encodings of different objects are also different, that it becomes really easy to *classify the images.* If we feed an image into the model and the encoding is similar to some other "cat" encodings the model has seen, it can say that it's a "cat"!

It seems like the key to good image classification then is learning *ideal image encodings*. This is actually the entire premise behind CLIP (and most of deep learning)! We start with terrible encodings (i.e. random encodings per image), and we want the model to *learn* ideal encodings (i.e. cat images have similar encodings).

For more intuition behind this idea of learning representations of data, I recommend the following article:

> **Understanding Latent Space in Machine Learning → https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d** *Learn a fundamental, yet often 'hidden,' concept of deep learning*towardsdatascience.com → https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d

Why should the image encoding be as similar as possible to it's corresponding text encoding? Now that we know what an encoding is and why it's important to learn good encodings, we can explore why we are forcing the model to make image and text encodings similar.

Recall that our ultimate goal is to learn how to classify *images*, and we therefore need to learn good image representations (encodings). When we had labels, we were able to learn good encodings by minimizing the difference between the *model output* and the *expected output* (the label).

However, in the case of CLIP, we don't have a single *model output*, and we don't have a single *expected output*. Instead, we can treat the image encodings of the training images as the model output, and the text encodings of the corresponding captions as the expected output.

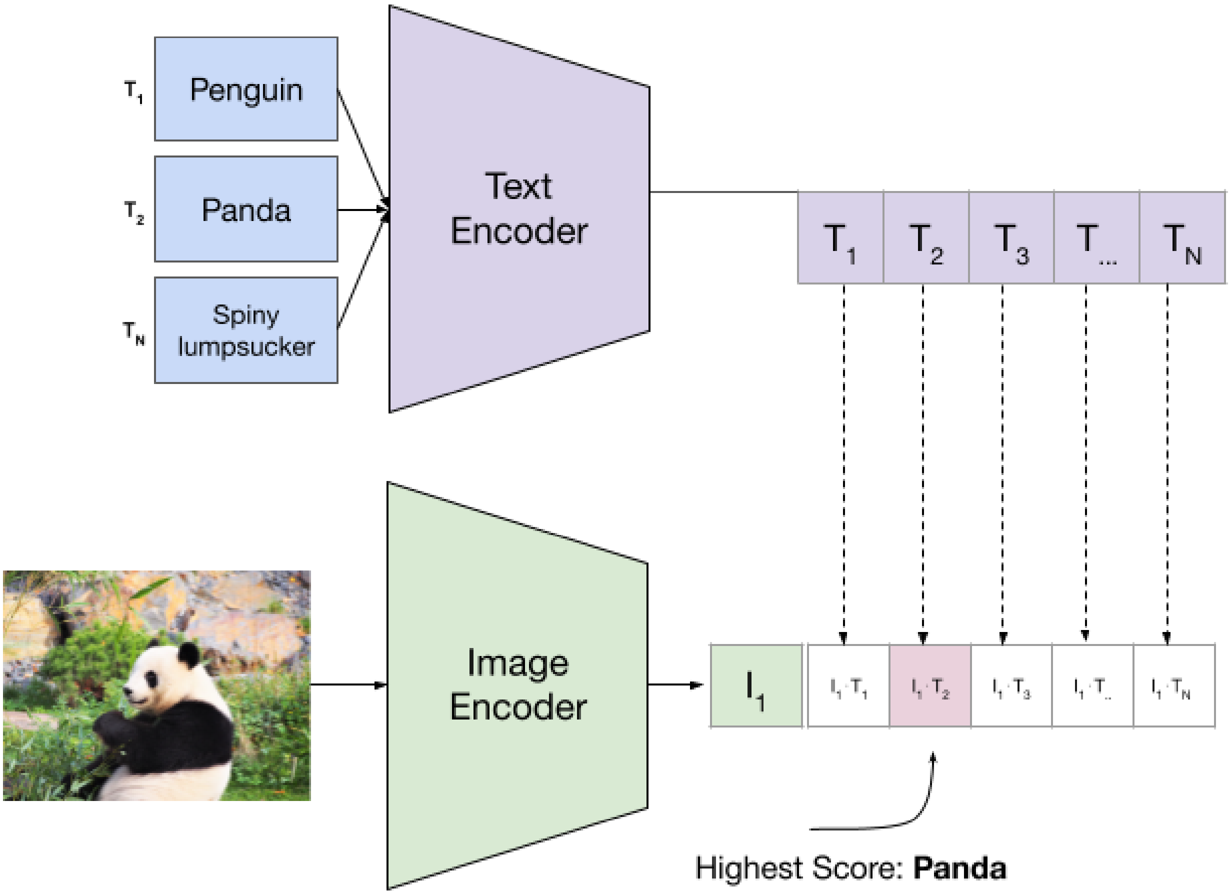
In fact, we can imagine that the **model is learning how to create good labels for us.** Since the text encoder is also updated in the process, over time, the model learns how to extract *more important information* from the text, thereby giving us a better text encoding (expected output).

With this in mind, it makes sense that we should try to minimize the difference between the image and text encodings. *It's because we know that images that are similar will likely have similar text encodings*, just like in the labeled setup where images that are similar will have the same label. As a result, the model will learn to generate similar encodings for similar images.

Diving Deeper — Zero-Shot Inference

Once the model is trained on enough image-text pairings, it can be used for *inference (use the model to make predictions on unseen data).* This is where the setup gets really clever.

In the *inference* stage, we setup the typical *classification task* by first obtaining a list of *all possible labels*. So if we were predicting animal species, we would need a list of all animal species (i.e. Penguins, Pandas, Spiny lumpsucker[1], etc.)(Step 2 in figure).



✍Photo by Polina Razorilova → https://unsplash.com/@bolotohodova on Unsplash → https://unsplash.com/photos/rlDlMH07nRY. Figure by Author adapted from figure in [2].

Each label will then be encoded by the pretrained text encoder from Step 1.

Now that we have the label encodings, T_1 to T_n, we can take the image that we want to classify, feed it through the pretrained image encoder, and compute how *similar* the image encoding is to *each text label encoding* using a distance metric called **cosine similarity → https://towardsdatascience.com/understanding-cosine-similarity-and-its-application-fd42f585296a.**

We now classify the image as the label *with the greatest similarity* to the image. We can do this since we know the model has learned to generate encodings for images that are as similar as possible to it's textual counterpart, most of which likely contained the label we are trying to classify.

And voila! We've achieved zero-shot capabilities![2]

Interestingly, CLIP → https://github.com/openai/CLIP is a relatively new method that has a unique level of simplicity compared to more traditional zero-shot learning approaches. Some of these concepts, including the **embedding based approach** and the **generative approach** are explained intuitively in the article below.

> **Zero-Shot Learning: Can you classify an object without seeing it before? - KDnuggets → https://www.kdnuggets.com/2021/04/zero-shot-learning.html** *Developing machine learning models that can perform predictive functions on data it has never seen before has become an…*www.kdnuggets.com → https://www.kdnuggets.com/2021/04/zero-shot-learning.html

> Regardless of which approach is used, a common theme of zero-shot learning is that we can use some **auxiliary information (i.e. textual descriptions)** that aren't explicit labels as a weak form of supervision.

Potential Applications of Zero-Shot Learning

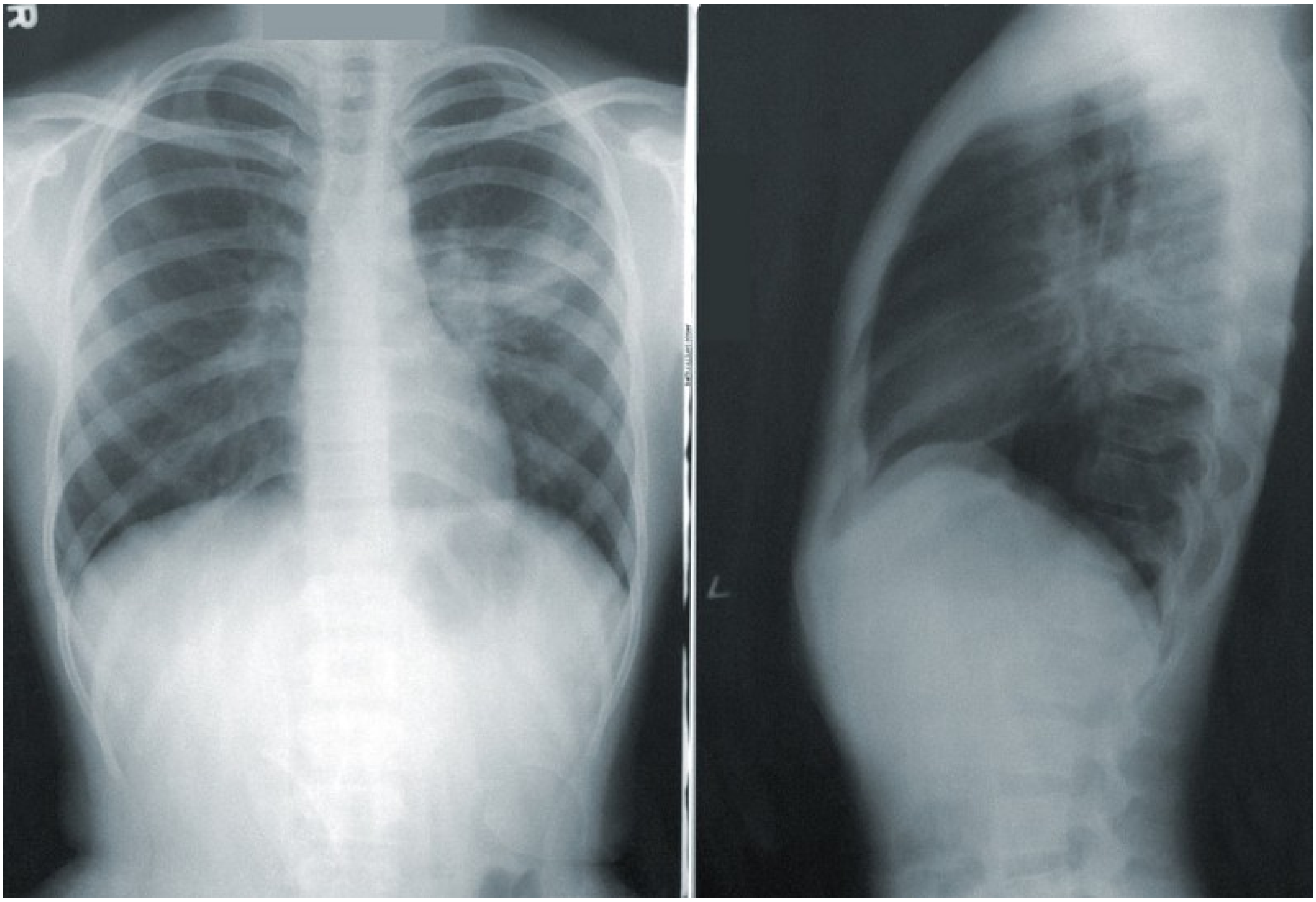
In general, zero-shot learning is most useful in scenarios where large annotated datasets are necessary, but not easy to obtain.

Below, we examine some potential real-world applications outlined in [3] → https://arxiv.org/abs/2004.14143.

COVID-19 Chest X-Ray Diagnosis

COVID-19 chest x-ray diagnosis is a perfect (and topical if you're reading this in 2021) example of how zero-shot learning may be applied in a medically relevant and low-label setting.

Since there were few COVID-19 positive chest x-rays in the initial stages of the pandemic, it was difficult to create a high performing model that could distinguish COVID from non-COVID, or COVID from a more common respiratory disease such as Pneumonia.



✍Example of Chest X-Rays. Photo by CDC → https://unsplash.com/@cdc on Unsplash → https://unsplash.com/photos/NMZdj2Zu36M

To solve this issue of few labels, we want a model that can learn about COVID-19 having only seen images of other diseases and some auxiliary information about COVID-19 chest x-rays — no explicit labels of COVID.

In this case, we may be able to use text descriptions of COVID-19 chest x-rays as a form of auxiliary info. An example is included below.

> "Bilateral multifocal patchy GGOs and consolidation can be seen." — [3] → https://arxiv.org/abs/2004.14143

To the best of my knowledge, there are no formally published works that demonstrate zero-shot applied to COVID classification yet. However, the availability of auxiliary text information highlights the potential of zero-shot learning to be applied to such a medically important task.

Autonomous Vehicles

✍Image of Tesla. Photo by Alexander Zahn → https://unsplash.com/@alexzahn on Unsplash → https://unsplash.com/photos/re8FigEQ4eQ

Another crucial example of an AI that will almost certainly see something it hasn't seen before is the *perception system* of an **autonomous vehicle.**

In fact, even humans see new objects on the road that they don't know how to react to. If we want to design safe and robust self-driving cars, we need to give them the ability to adapt to unseen circumstances.

Again, to the best of my knowledge, there are few works that have applied zero-shot approaches to self-driving perception systems. However, some interesting ideas include using textual descriptions of concept cars[3] to teach a model to differentiate from regular cars [3] and transferring policies learned in simulation to a city in a zero-shot fashion [1].

Key Takeaways

- **Zero-shot Learning** is a setup in which a model can learn to recognize things that it hasn't explicitly seen before in training.
- There are different **zero-shot learning** approaches, but a commonality is that auxiliary information such as textual descriptions are used or encoded during the training process instead of explicit labels.

- There are several potential applications of zero-shot learning in the real world — from COVID-19 disease diagnosis to self-driving cars.

The works explained in this article are only the tip of the iceberg! As the field progresses, we steadily work towards the AI that many of us envision: one that's not strictly limited by the structure of labeled data, one that can understand things it hasn't seen before, and one that's more human.

References

[1] Jang, K., Vinitsky, E., Chalaki, B., Remer, B., Beaver, L., Malikopoulos, A. A., & Bayen, A. (2019, April). Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles. In → https://arxiv.org/abs/1812.06120*Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems* (pp. 291–300). → https://arxiv.org/abs/1812.06120

[2] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., … & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. → https://arxiv.org/abs/2103.00020*arXiv preprint arXiv:2103.00020*. → https://arxiv.org/abs/2103.00020

[3] Rezaei, M., & Shahidi, M. (2020). Zero-shot learning and its applications from autonomous vehicles to covid-19 diagnosis: A review. → https://arxiv.org/abs/2004.14143*Intelligence-based medicine*, 100005. → https://arxiv.org/abs/2004.14143

Notes

1. Yes, the "spiny lumpsucker" is an actual thing. See https://www.treehugger.com/animals-with-completely-ridiculous-names-4864307
2. This is a simplified explanation of CLIP. I suggest taking a look at the paper → https://arxiv.org/pdf/2103.00020.pdf from OpenAI.
3. According to Wikipedia → https://en.wikipedia.org/wiki/Concept_car#:~:text=A%20concept%20car%20(also%20known,may%20not%20be%20mass%2Dproduced., A **concept car** is "a car made to showcase new styling and/or new technology". In other words, they look wack enough to be distinguishable from regular cars.