Visual Search: Fine-grained Recognition with Embedding Models for the Edge

Omid Azizi

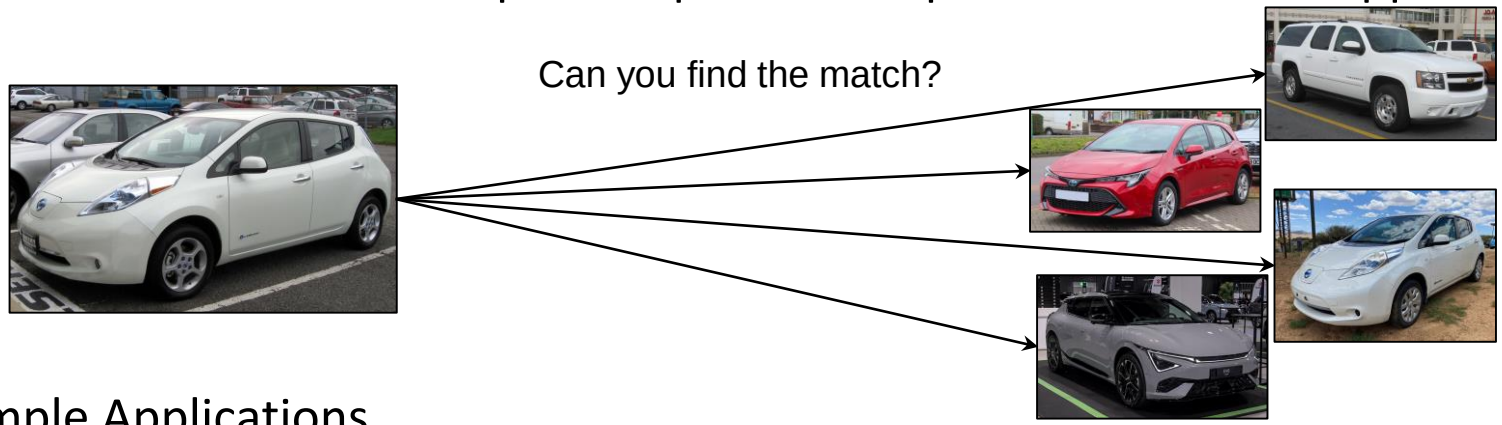
Co-founder

Gimlet Labs

gimlet

Visual search

Visual search is often an important part of computer vision based applications



Example Applications

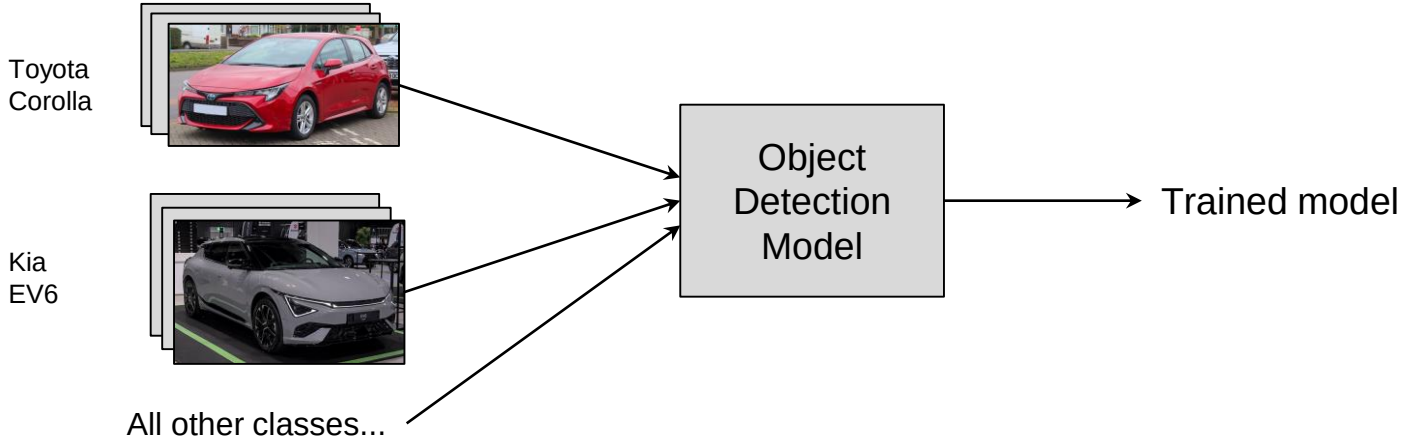
- Social media: person recognition (e.g., your photo library app)
- Retail: product recognition, self-checkout systems
- Manufacturing: defect detection

Approaches to visual search

1. Classical computer vision
   - SIFT/SURF - look for features like edges
2. Closed-set AI approaches
   - Example: classification models
   - Explicitly trained for a fixed set of classes
3. Open-set AI approaches (embedding models)
   - Not necessarily tied to specific classes
   - Model finds the relevant features to “characterize” an image

Using classification/object detection for visual search

A simple approach is to directly train for your classes:



But this assumes you know all the classes of interest

Open-set visual search

What if the search targets aren’t fixed? How can you handle new targets?

How can you distinguish across fine-grained differences?

Humans recognize novel objects with just a few examples — without re-learning the knowledge base

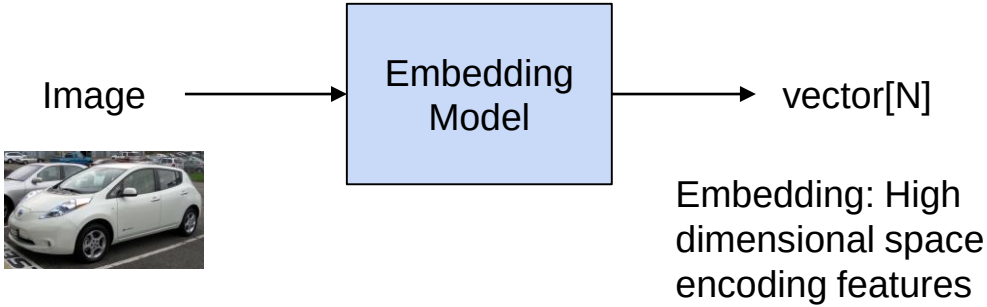
- If I give you a few examples of a new product, you’ll probably be able to recognize it

Goal: An AI model where the classes are not fixed

Embedding models are the answer

Instead of teaching a model specific classes, why don't we teach it to organize visual features?

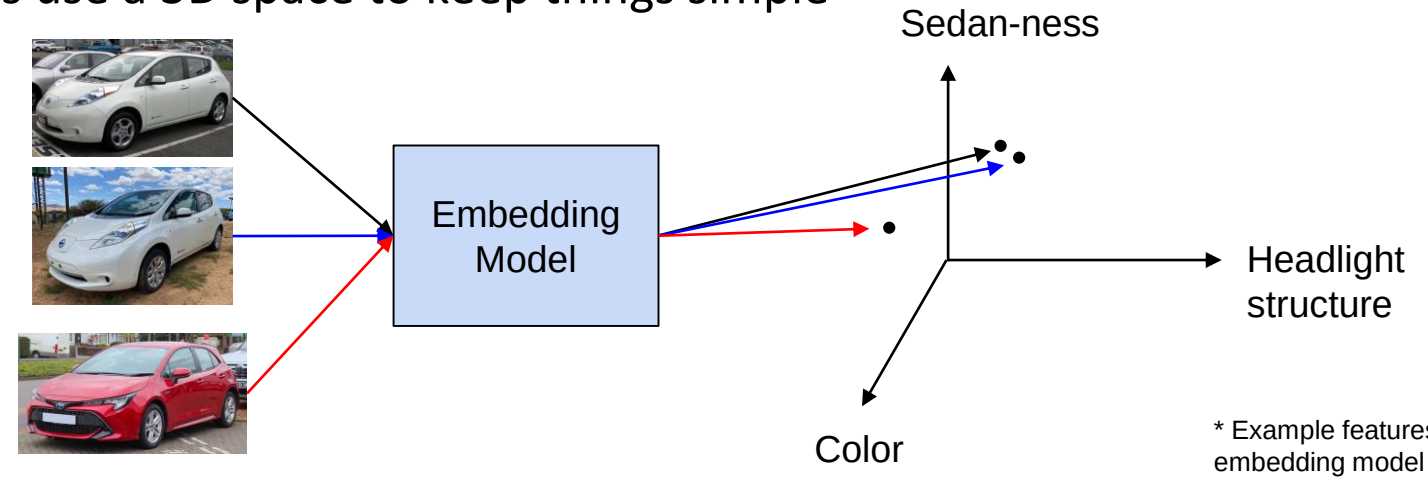
Features can be encoded in an ***embedding***: A numerical representation of data—like images or text—that captures its essential features in a continuous vector space.



During training, the model figures out how to organize the embedded information into the various dimensions

Embedding models analogy

Let’s use a 3D space to keep things simple



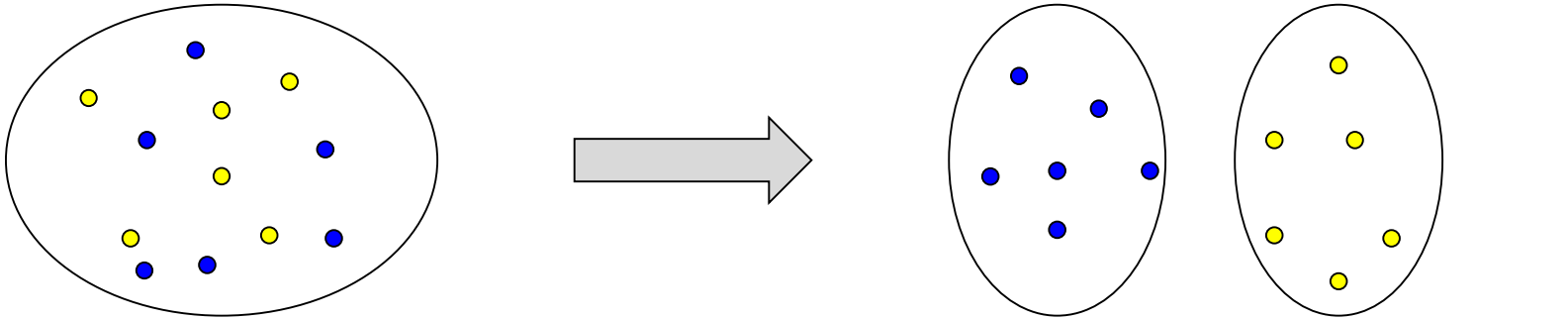
* Example features are not real; the embedding model will learn visual features during training

Typical image embedding space ranges in dimensions from 256 to 2048

What about distinguishing similar objects?

They may appear close to each other in the embedding space, but not too close

- Include similar, but differently labeled items in your training data set
- For example, model year 2018 vs model year 2023 versions of the same car



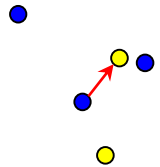
close, but disparate in the embedding space

With the right training data and objective, the embedding model can learn relevant features to distinguish similar objects

Choosing the right loss function

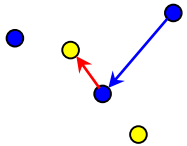
Think of training an embedding model as a process of organizing the data.

Contrastive Loss
(independent pairs)



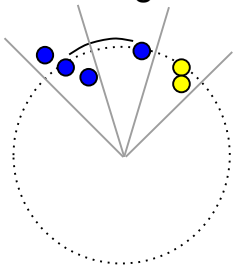
Pull similar pairs close, push dissimilar ones apart.

Triplet Loss
(one positive, one negative)



Anchor must be *relatively closer* to the positive than the negative.

Angular Margin Loss



Ensures the angle between embeddings of different classes is sufficiently large

Triplet loss is a good starting point for visual search

- Relative loss function yields good organizational structure for the search

Fine-tuning your own embedding model

Usually, you'll start with a pre-trained model

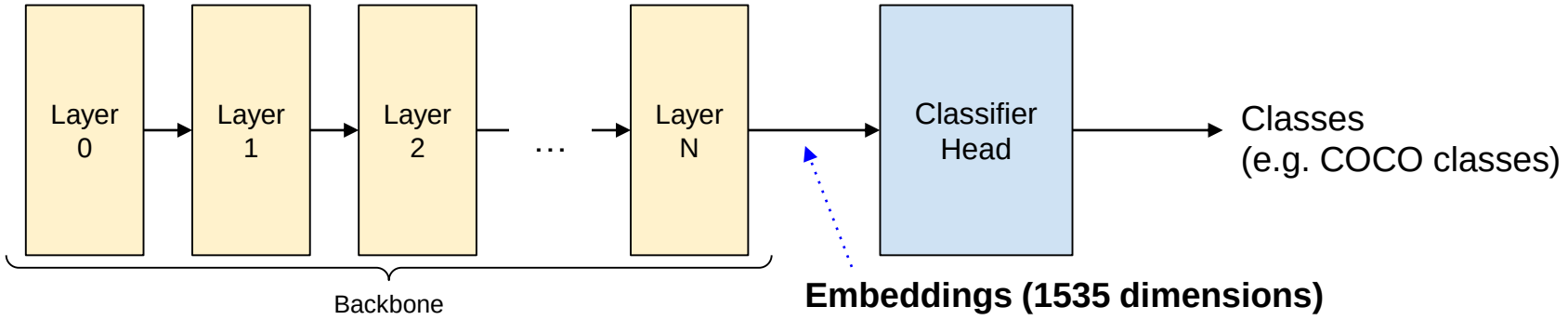
- Something that has seen a wide variety of images before

Model	Comment
EfficientNet	Good for embedded execution
MobileNet	Good for embedded execution
ViTs	Typically larger, but don't discount them
CLIP	Embedding space trained on text and images

EfficientNet

EfficientNet has a backbone and a classification head

- In the process of training, the backbone has learned to create embeddings



Original model may be trained on different classes, but has still learned to “see”

- Remove the head and fine-tune the backbone for your classes

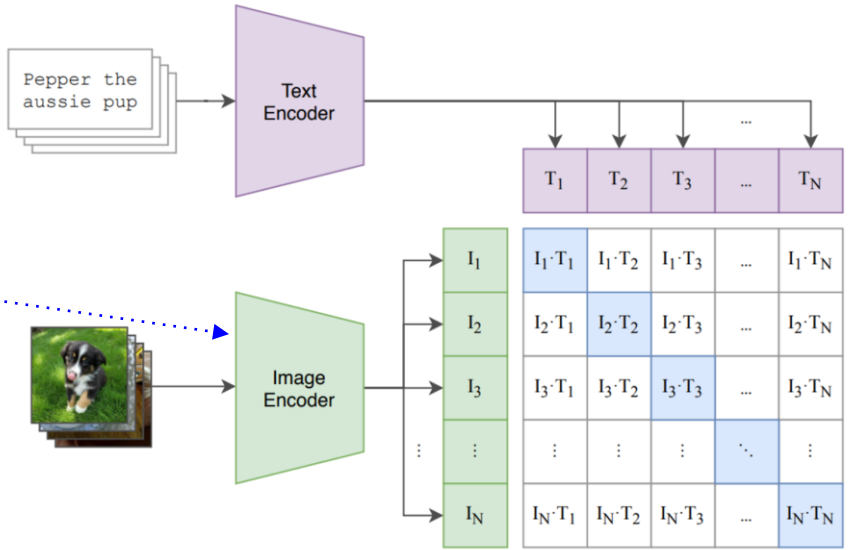
OpenCLIP

CLIP models use a joint text-vision embedding

- CLIP has learned to organize visual data
- Trained with lots of data

Why not just use the image encoder?

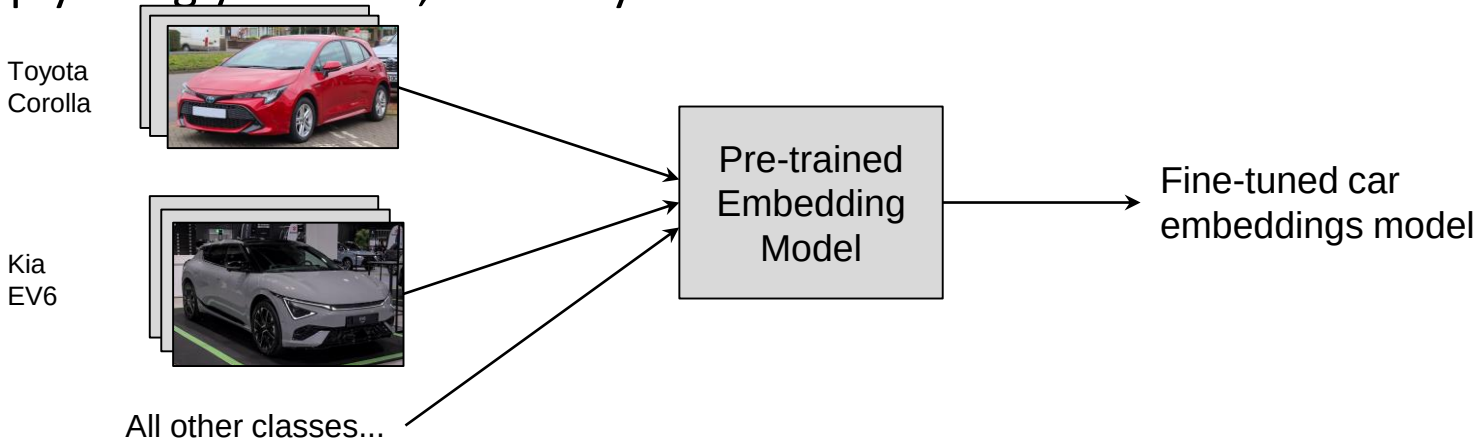
- Powerful 768-dim embedding space



Source: Learning Transferable Visual Models From Natural Language Supervision [https://arxiv.org/pdf/2103.00020]

Fine-tuning with your data

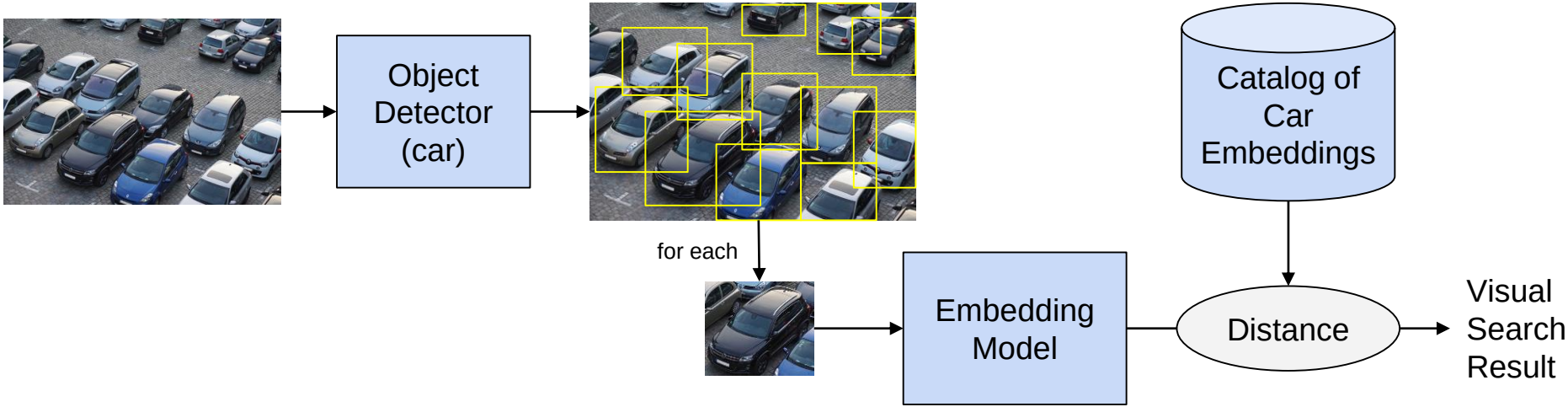
Simply bring your data, choose your loss function and train



A few epochs are often enough to get good results

Using embedding models in a practical setup

Putting things together: real use cases often need more than an embedding model



Embedding models for embedded devices

You don’t need a discrete GPU to run powerful embedding models

	Parameters	NUC with 12th Gen Intel Core i7	Jetson Orin Nano
OpenCLIP ViT-B-32	~87M	120 ms	56 ms
EfficientNet B3	~10M	37 ms	54 ms

* Measurements on Gimlet platform, and include a full pipeline (including image resizing and other work)

Summary

The next time you have to visually search or detect an object, ask yourself

- Should I use an embedding model instead?

An open-set approach like an embedding model might just fit the bill

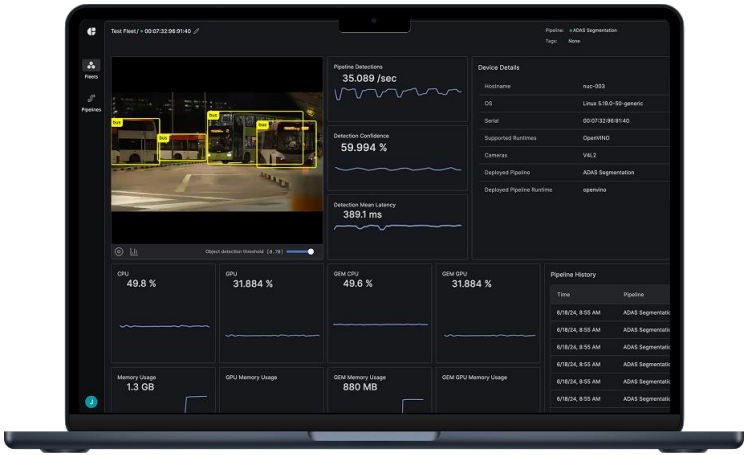
- Ability to add new classes, without retraining
- Ability to search fine-grained where classes are not known up-front

Unless you're sure you have a fixed set of objects to detect, consider embedding models for your next embedded vision project!

Resources

Gimlet: Run AI on the edge and the cloud

- Come visit the Gimlet booth #721!
- Waitlist: gimletlabs.ai



Train your own visual search embedding space

- https://github.com/gimletlabs/embedding-tuning

Visual search publications

- FaceNet: A Unified Embedding for Face Recognition and Clustering
- Learning Fine-grained Image Similarity with Deep Ranking
- ArcFace: Additive Angular Margin Loss for Deep Face Recognition

Thanks!

Contact: oazizi@gimletlabs.ai