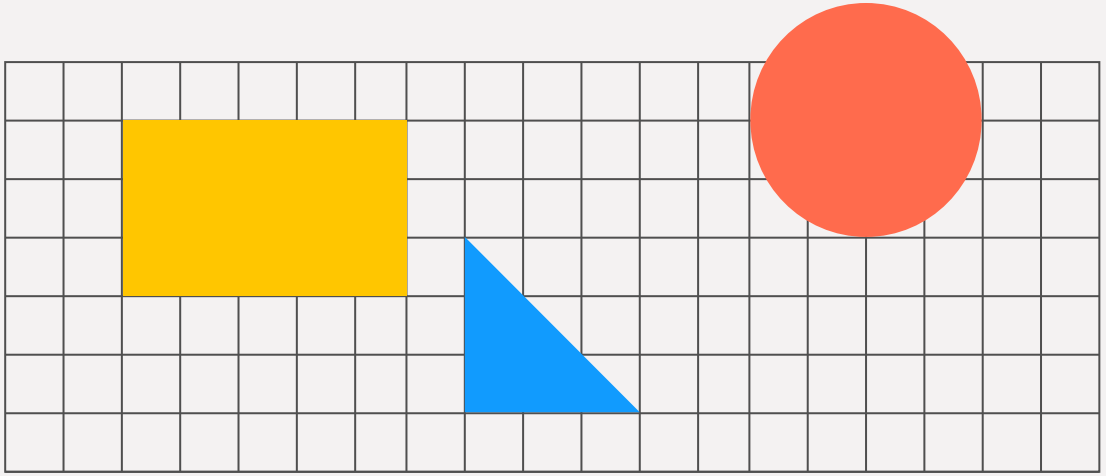
Image recognition with vector database

Lawrence Lin
Hengjie Zhan



Introduction

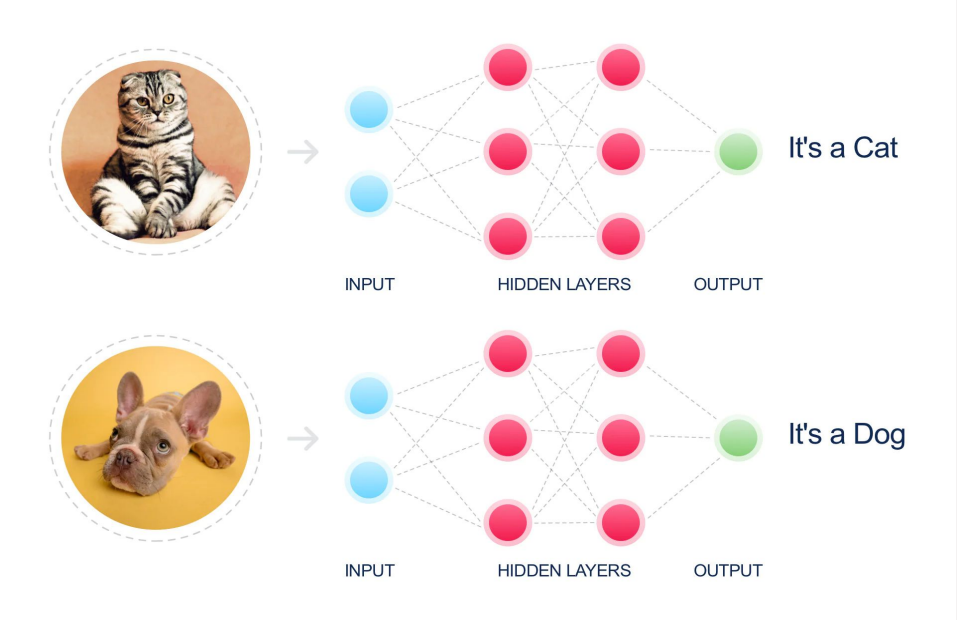
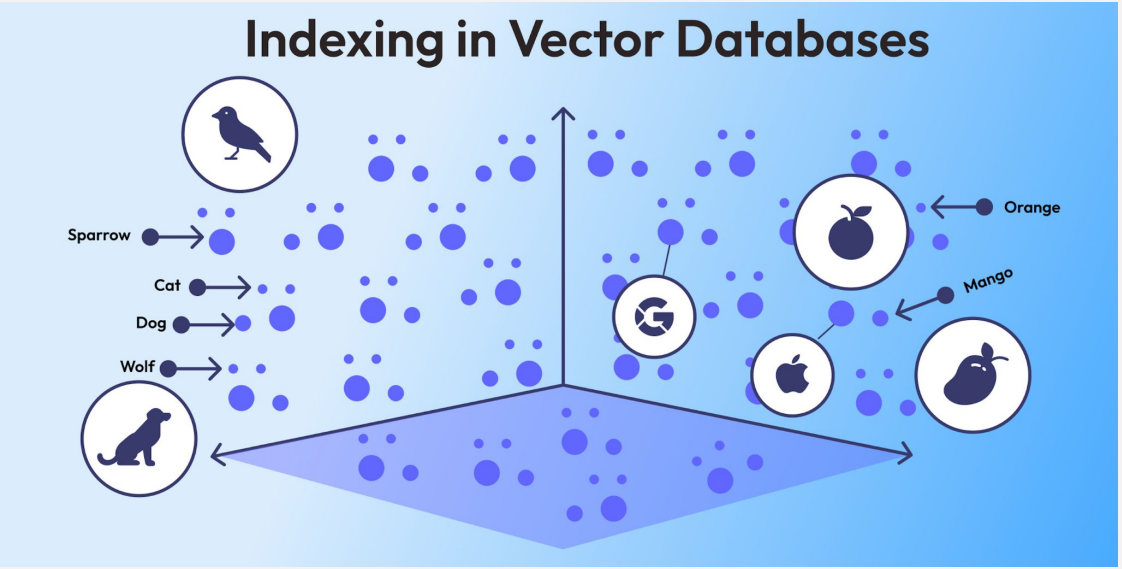


Image recognition is a field of artificial intelligence that enables systems to analyze and interpret visual data. By using advanced algorithms, particularly in **computer vision**, image recognition identifies objects, patterns, scenes, and even emotions within digital images or videos.

Vector databases can store image features as **high-dimensional vectors**, enabling fast similarity searches and scalable retrieval. By integrating vector databases with image recognition, industries like **e-commerce** can provide advanced visual search tools.

What is vector database?



Vector databases are purpose-built for storing and searching **high-dimensional vectors**, which are numerical representations of data such as images, text, or audio. In image recognition, models like convolutional neural networks (CNNs) generate feature embeddings—compact vector representations that encode the essence of an image. By leveraging vector databases, these embeddings can be efficiently indexed, searched, and compared.

How to transform image to vector?

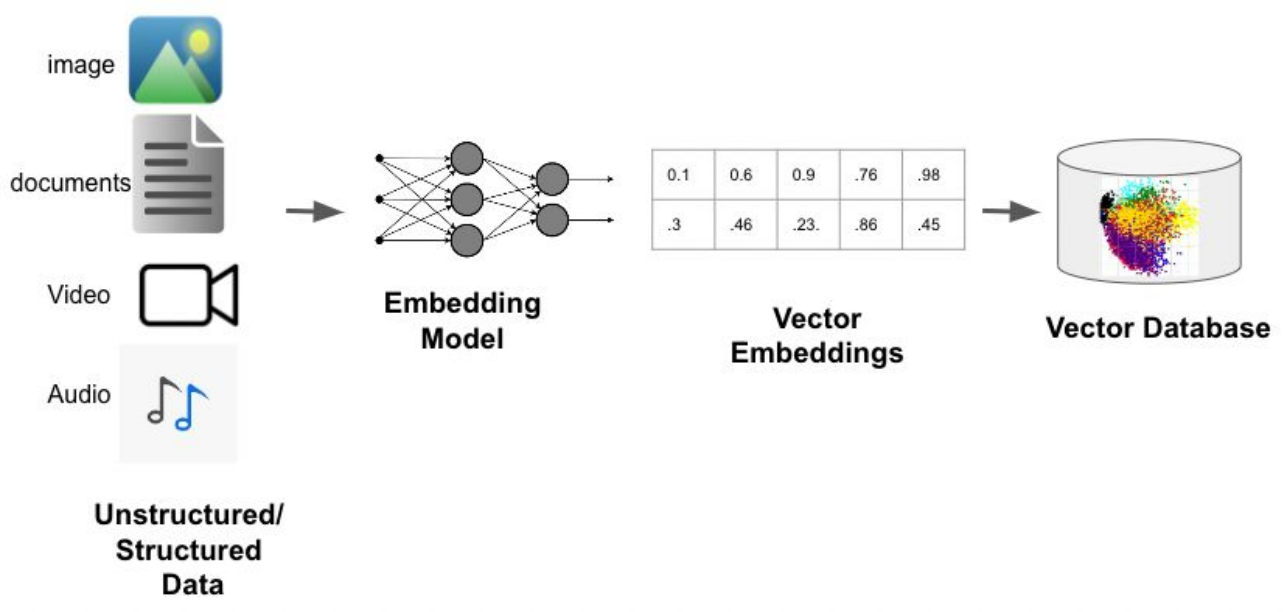
We use **MobileNetV2**, which is a deep learning architecture designed to enable efficient image classification and object detection on resource-constrained devices like smartphones and embedded systems.

How does it work?

1. Image preprocessing:
   Resize: Adjust the image size to the required dimensions of the model (typically 224x224).
   Pixel Value Normalization: Scale the pixel values to the range [-1, 1].

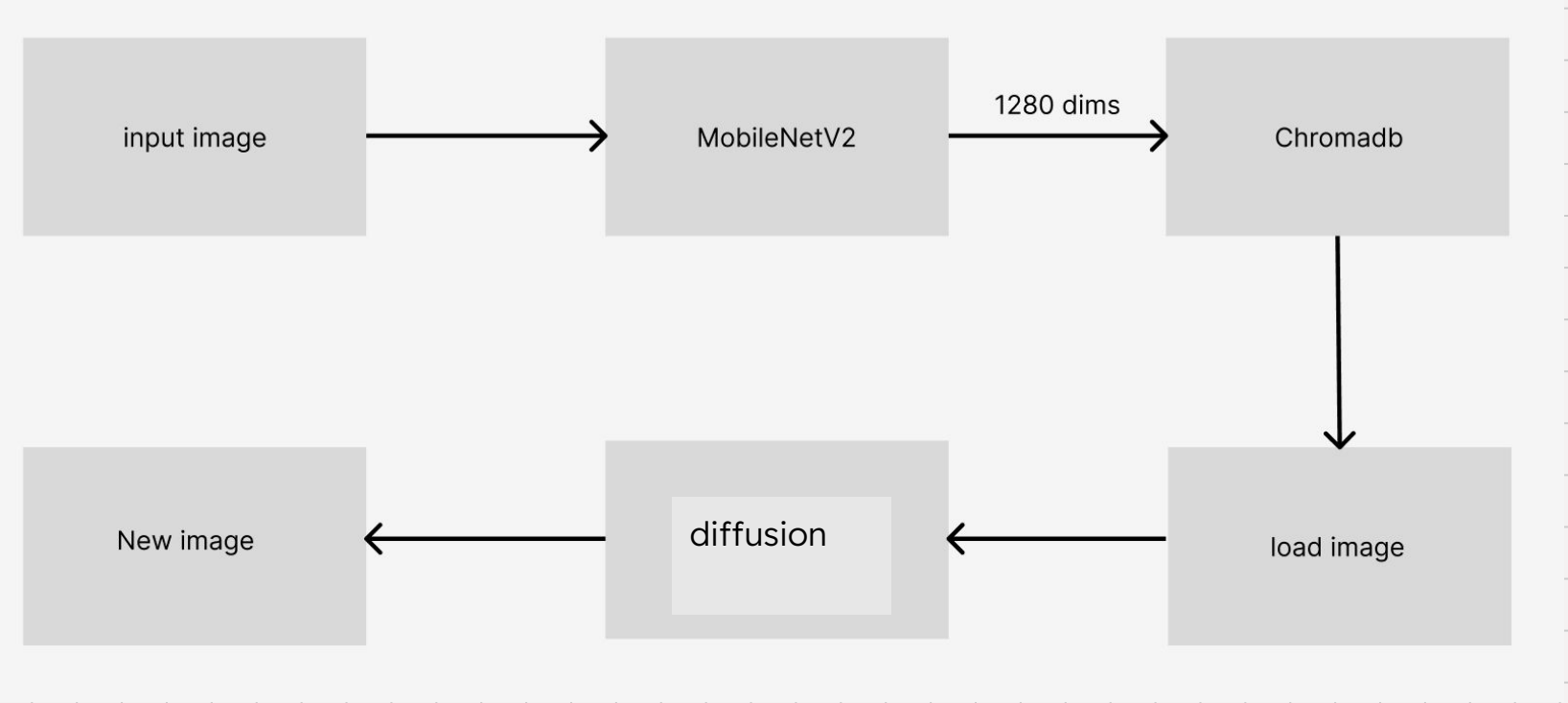
2. Feature Vector Extraction:
   MobileNetV2 outputs a fixed-size high-dimensional feature vector, typically of dimension 1280 (depending on the model configuration). This vector represents the semantic features of the image.

Vector Database: ChromaDB



ChromaDB is the open-source AI application database, which is designed for efficiently storing and retrieving vector embeddings.

Project Architecture



Methodology

1. Preprocessing:

- Images resized to 224x224 pixels.
- Normalized using MobileNetV2's preprocessing pipeline.

2. Feature Extraction:

- Utilized **MobileNetV2** for generating high-dimensional feature vectors.
- Extracted 1280 dims array embeddings represent unique image features.

3. Data Storage:

- **ChromaDB** stores image embeddings as vectors.
- Metadata, including file location, saved for easy reference.

Methodology

4. Image Query:

- Use K-Nearest Neighbors (KNN) query to search from ChromaDB.
- Finds visually similar images based on embeddings.

5. Stable Diffusion

- Create an new image with img2img function base on the image from the ChromaDB.

6. User Interface:

- CustomTkinter UI enables image upload, embedding, and querying.
- Results displayed with thumbnails of matching images.

Conclusion

- Successfully implemented a pipeline for image embedding and retrieval.
- Leveraged TensorFlow and ChromaDB for efficient backend processing.
- Intuitive UI allows non-technical users to interact with the system.

Future work

- **Use different model like YOLO**
- **Use pytorch to implement img2img**
- **Run the model on the GPU**

Demo