Landmark Recognition

1 Introduction

Landmark recognition is a fundamental task in the field of computer vision, with significant applications in areas such as tourism, historical site identification, and augmented reality. It can be framed as an instance-level recognition problem, where the objective is to determine the specific instance of a broader object class—namely, a unique landmark—depicted in a given image.

2 Problem Statement

The task of landmark recognition involves accurately classifying and identifying landmarks depicted in query images. This is a challenging problem due to the wide variety of global landmarks, differences in lighting, angles, occlusions, and image quality. The goal is to build a system capable of recognizing and naming the landmark present in a given image with high accuracy. The project serves as an attempt to address the Google Landmark Recognition Competition hosted annually by Google on Kaggle platform. Leveraging recent advances in convolutional neural networks, the proposed solution utilizes transfer learning and model fine-tuning techniques to enhance recognition performance.

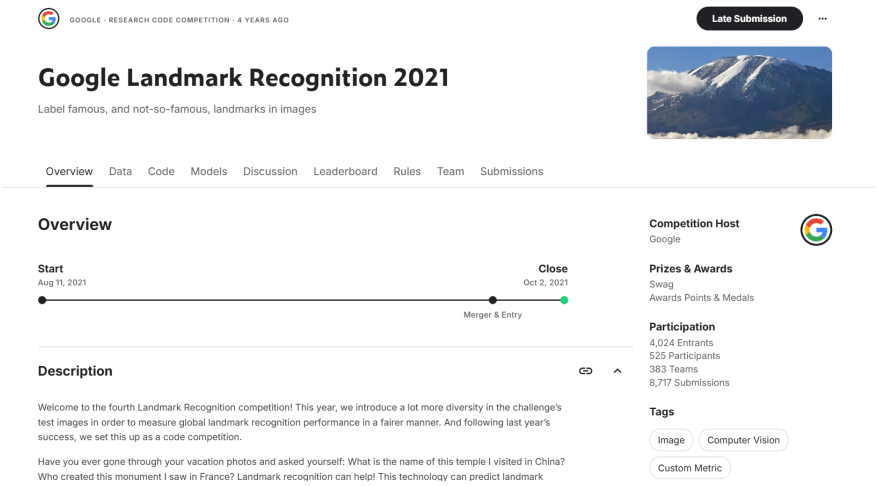


Figure 1: The competition on Kaggle

The system operates as follows: a user provides an input image, the model processes the image, predicts the most probable landmark depicted, and outputs the name of the recognized landmark. This approach combines state-of-the-art neural network techniques with a practical deployment strategy to tackle the real-world problem of large-scale landmark identification. This work contributes to the broader effort of advancing automated visual recognition systems capable of identifying real-world objects at a fine-grained level, a key challenge in contemporary computer vision research.

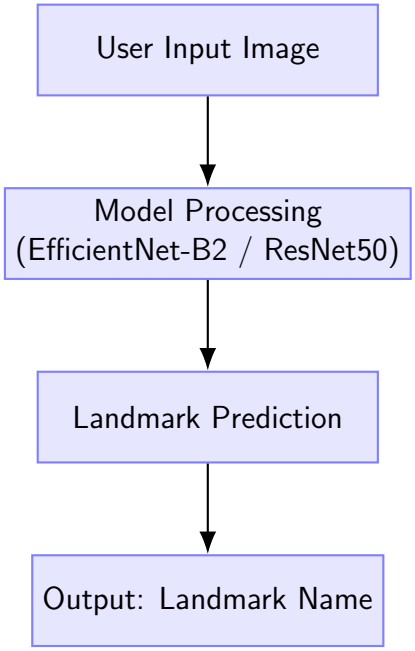


Figure 2: Pipeline for Landmark Recognition System

3 Dataset

In the third part, we discuss about the dataset we use in the project, as well as techniques we use to process the given dataset from the competition.

3.1 Original Dataset

In the context of the project, we mainly use the **Google Landmarks Dataset v2 (GLDv2)** as the dataset for landmark recognition. The original dataset is the largest dataset to date by a large margin, introduced as the new benchmark for large-scale, fine-grained instance recognition and image retrieval in the domain of human-made and natural landmarks. The images in the dataset are sourced from Wikimedia Commons, the world's largest crowd-sourced collection of landmark photos. The *GLDv2* dataset has 3 main properties that mimic the challenges of industrial landmark recognition systems:

- **Large-Scale:** GLDv2 contains over 5 million images spanning more than 200,000 classes of both human-made and natural landmarks. This scale presents significant

computational and modeling challenges.

- **Intra-Class Variability:** Images within the same landmark class can vary drastically. For instance, photos of a single landmark might include outdoor views, indoor scenes, artworks depicting the landmark (e.g., paintings in a museum), or even indirect elements (like a building's floor or a gym room in a mall). This visual diversity within classes increases the difficulty of learning consistent representations.

- **Long-Tailed Class Distribution:** The dataset mimics real-world frequency distribution. Popular landmarks have thousands of images, while obscure ones have very few. For example, the Eiffel Tower has over 400 images, whereas the Hanoi Opera House has only about 30. This imbalance requires strategies to handle underrepresented classes.

Moreover, images in GLDv2 are collected from Wikimedia Commons and other open sources, relying heavily on user-submitted content and metadata. Although reviewed by local experts, some labels are weak or ambiguous. For instance, an image of a painting of a landmark, or an interior scene loosely related to it, is still included in that landmark class. This contributes to the intra-class variability and potential label noise.

3.2 Our Dataset

As explained above, using the original GLDv2 dataset raises several significant challenges:

- **Scale and Computation:** Due to its enormous size, the dataset is extremely difficult and computationally expensive to train on, fine-tune, or even perform basic processing. With a total size exceeding 100GB, the dataset also poses considerable difficulties in terms of storage and cloud-based operations.

- **Noisy and Low-Quality Images:** The large number of images does not always translate to higher quality. Many of the images fail to highlight distinctive features of the landmarks, contributing instead to noise and reduced dataset clarity. A major challenge comes from out-of-domain (OOD) images, which provide little or no meaningful information for the task of recognition. In the original test set, only 1.1% of images actually depict landmarks, while 98.9% are out-of-domain, including human faces, animals, insects, and other unrelated content.

- **Lack of Generalization Support:** The dataset was not explicitly designed to evaluate the generalization capabilities of embedding models to unseen data. As such, the index and training sets do not contain disjoint class sets, limiting its usefulness for testing cross-class generalization.

Perhaps in recognition of the significant challenges posed by the original dataset, the organizers of the competition provided a curated subset known as the **Google Landmarks Dataset v2 Clean (GLDv2-Clean)**. This refined version contains approximately 1.5 million images across 80,000 landmark classes, representing an 80% reduction in both the number of images and classes compared to the original dataset. To construct *GLDv2-Clean*, the authors applied the **DELF (Deep Local Features)** technique combined with **k-Nearest**

Neighbors (k-NN) to filter the data. This approach aimed to retain only those images most relevant to each landmark—specifically, images that convey the most meaningful and distinctive features of the landmarks.

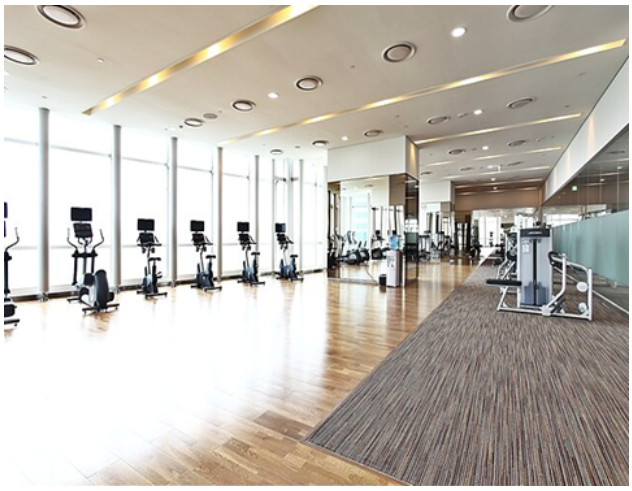


Figure 3: Some not very meaningful images

Despite this reduction, **GLDv2-Clean** remained too large for the scope of our project and the computational resources available to us. To address this, we performed an additional filtering step by selecting only the **top 100** most frequently occurring landmarks in the GLDv2-Clean dataset. This final reduced version, which we use for all our training and evaluation purposes, is referred to throughout this report as **GLDv2-Use**.

The **GLDv2-Use** dataset is further divided into three subsets: training, validation, and test sets. To ensure a balanced distribution across these subsets, **80%** of the images from each landmark class are allocated to the training set, while the validation and test sets each receive **10%**. This stratified split helps maintain class balance across all subsets, preventing any single class from dominating the validation or test evaluations.

- **Training set:** 52,824 images
- **Validation set:** 6,606 images
- **Test set:** 6,651 images

Each image is annotated with a unique landmark ID, ensuring clear identification of its corresponding class.

3.3 ADD THE ID OF EACH SET HERE

4 Exploratory Data Analysis (EDA)

In the fourth part of our report, we are going to provide some analysis to give insights into **GLDV2-Use**, the dataset which were used for training and testing in this project.

4.1 Distribution of number of images per class

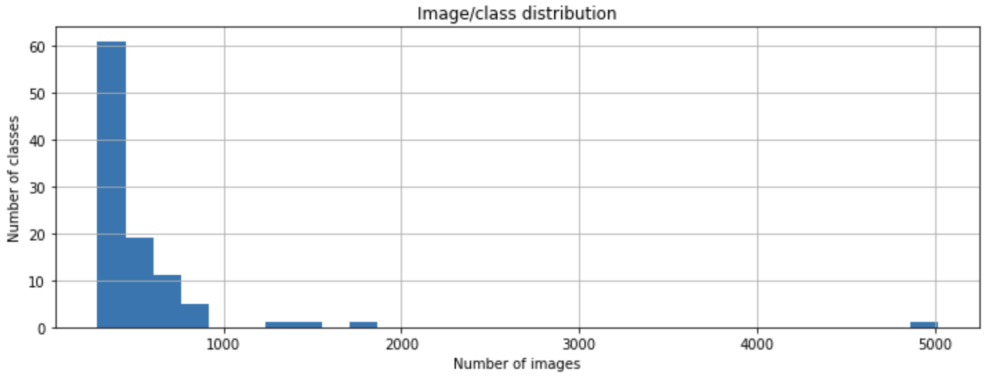


Figure 4: Distribution of number of images per class

The histogram illustrates a significant imbalance in the number of images per class. The majority of landmark classes contain fewer than 1,000 images, with a large concentration of classes having between 400 and 600 images. Only a few classes exceed this range, and a very small number have more than 2,000 images, with one class approaching 5,000 images.

4.2 Brightness and Contrast Analysis

Figure 5 illustrates the distribution of brightness and contrast values across a sample of images in the dataset.

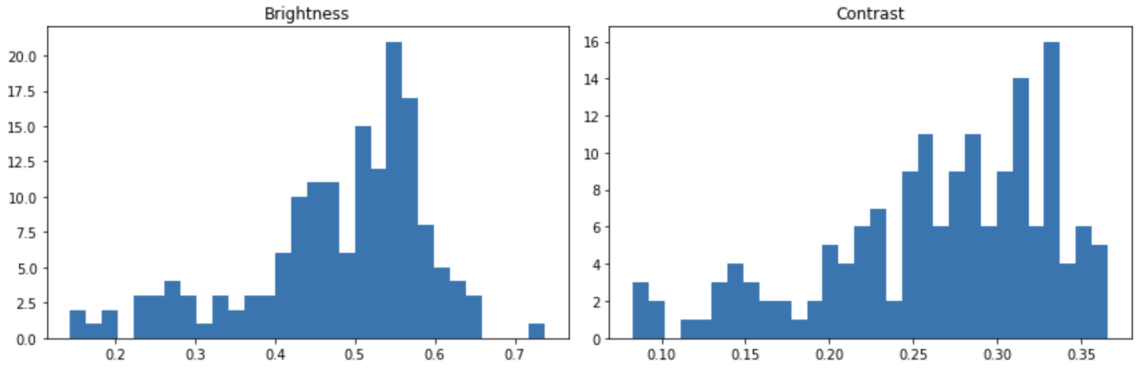


Figure 5: Distribution of brightness (left) and contrast (right) across images.

- **Brightness:** The brightness values are primarily concentrated between **0.45 and 0.65**. The distribution is slightly skewed to the left, with relatively few images exhibiting very low brightness levels (below 0.3). The peak occurs around **0.55**, indicating that most images are moderately bright. Only a small number of images fall at the extremes (below 0.2 or above 0.7).

- **Contrast:** The contrast distribution is skewed toward higher values. Most images fall within the **0.20 to 0.35** range, with a gradual increase from low contrast values starting at 0.10. The distribution peaks near **0.32–0.33**, then tapers off. Very few images have contrast values lower than 0.15.

The dataset exhibits generally consistent brightness and contrast characteristics, which is favorable for training a deep learning model. However, some variation still exists. To mitigate any potential issues caused by lighting inconsistencies.

4.3 Average RGB Channel Analysis

Figure 10 shows the distribution of average pixel intensities for each of the RGB channels across the dataset.

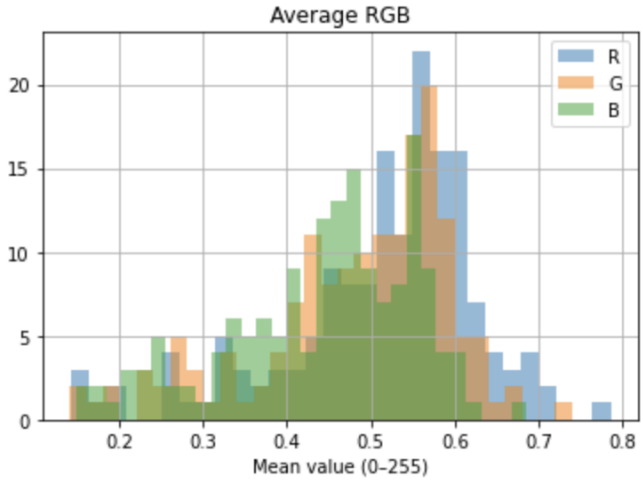


Figure 6: Distribution of average RGB values per image.

- **Red (R):** The red channel shows a strong peak between **0.55 and 0.60**, with fewer images having values below 0.3 or above 0.7.
- **Green (G):** The green channel closely follows the red channel, with a peak also near **0.55** and a slightly broader spread in the lower range.
- **Blue (B):** The blue channel appears more evenly distributed, with a gradual increase from 0.3 to 0.55 and a peak slightly lower than R and G. There are more images with lower average blue values, suggesting slightly warmer tones overall.

The RGB distributions suggest a well-balanced color profile across the dataset, though with a slight dominance of warmer hues (higher R and G values). This can affect color sensitivity during training, and normalization across channels is essential for consistent model performance.

4.4 Image Sharpness Analysis (Laplacian Variance)

Figure 7 presents the distribution of blur scores measured using Laplacian variance. This metric quantifies the sharpness of an image, where lower values indicate more blurriness.

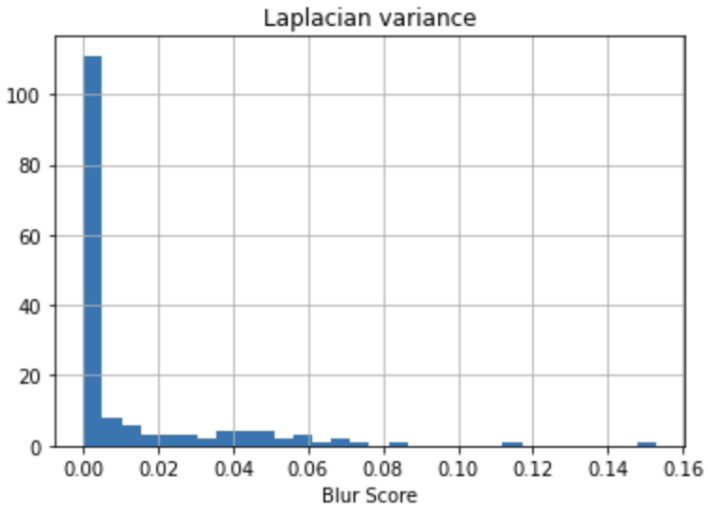


Figure 7: Distribution of image blur scores (Laplacian variance).

- The majority of images exhibit extremely low Laplacian variance scores (below **0.01**), indicating a high concentration of blurry images in the dataset.
- A small number of images have scores between **0.02 and 0.06**, suggesting moderate sharpness.
- Only a few images exceed a blur score of **0.10**, which reflects high sharpness.

The dataset includes a significant proportion of blurry images, which could negatively impact model performance. It may be beneficial to apply filtering or thresholding based on blur score to improve data quality or use data augmentation techniques that simulate focus and defocus conditions for robustness.

5 Data Preprocessing

Based on data analysis above, we purpose 2 images preprocessing pipelines:

5.1 Pipeline 1

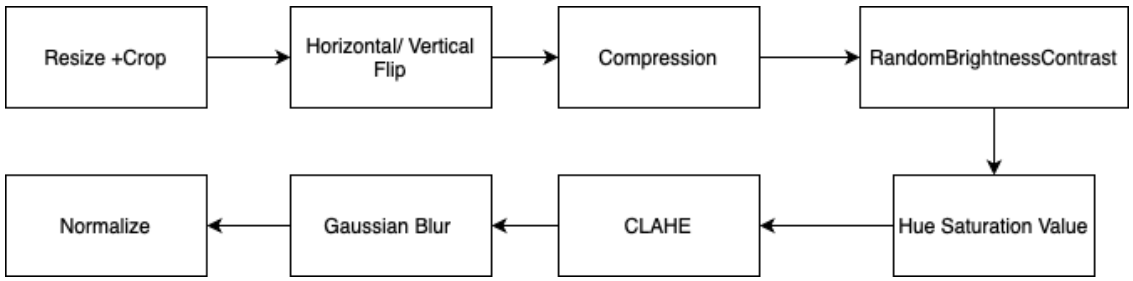


Figure 8: Pipeline 1 flow

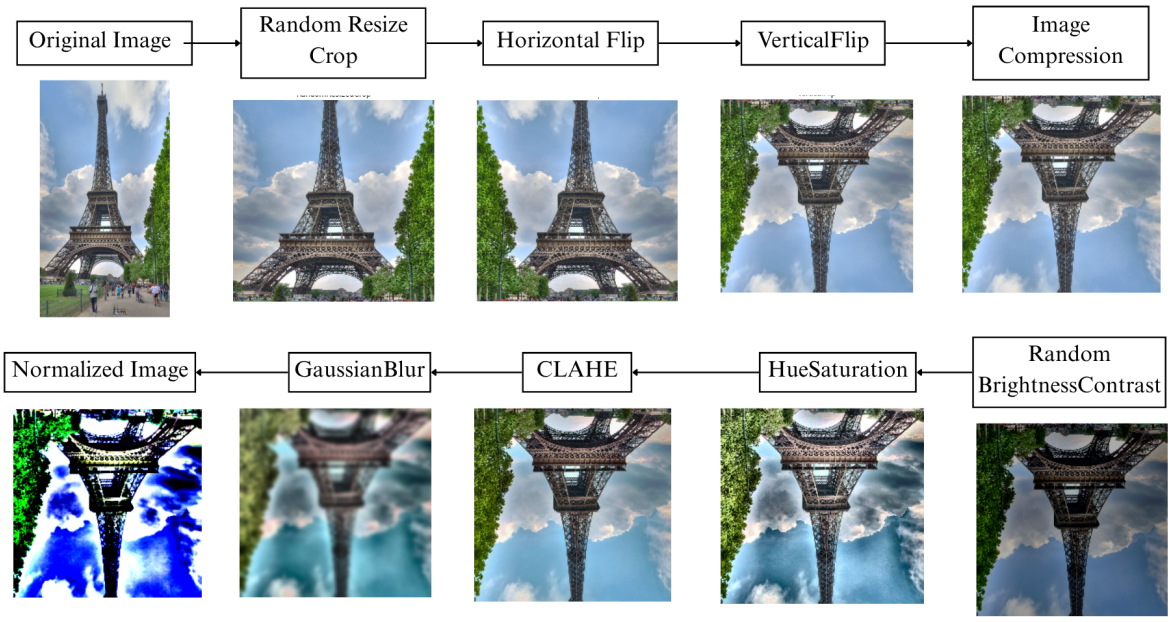


Figure 9: Pipeline 1 results

5.2 Pipeline 2

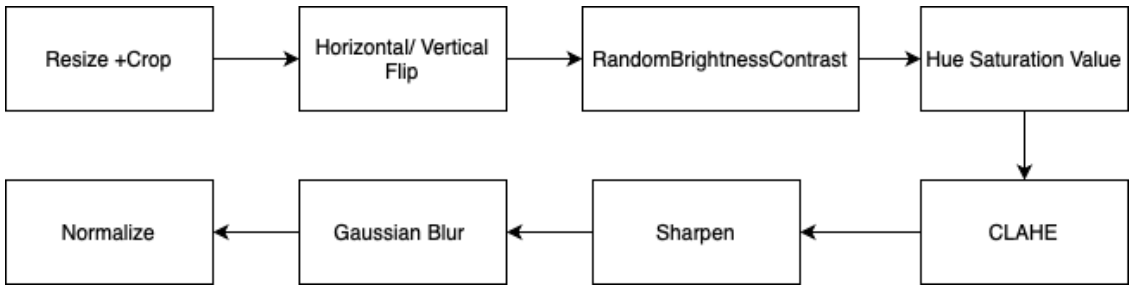


Figure 10: Pipeline 1 flow

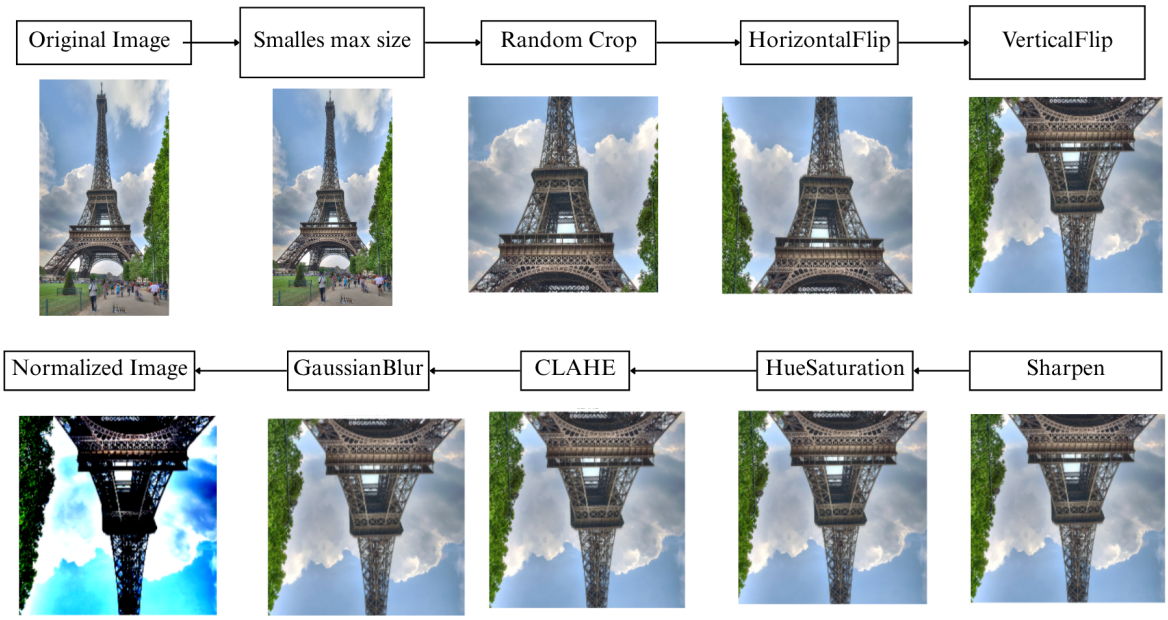


Figure 11: Pipeline 2 results

5.3 Comparisons

Although the 2 pipelines might seems similar, there are still some major differences:

Component	Pipeline 1	Pipeline 2
Resize + Crop	`RandomResizedCrop` (scale-based cropping)	`SmallestMaxSize(288)` + `RandomCrop` *Separates resize and crop*
Compression	`ImageCompression` (quality_lower=99, quality_upper=100)	Not included
BrightnessContrast	`p = 0.2`	`p = 0.3`
Sharpen	Not included	`Sharpen(alpha=(0.1, 0.3), p=0.2)`
Normalize	Default normalization (no mean/std specified)	Normalization with ImageNet statistics
Flip, Hue/Saturation, CLAHE, Blur	Included	Included

Table 1: Comparison between the two data augmentation pipelines.

Pipeline 1 employs a unified *RandomResizedCrop* operation, which combines resizing and cropping into a single step, alongside very high-quality image compression and a conservative brightness/contrast augmentation. This pipeline emphasizes minimal compression loss and simpler augmentation transformations, potentially preserving more natural image features.

In contrast, **Pipeline 2** separates resizing and cropping into two distinct operations (**SmallestMaxSize** followed by **RandomCrop**), enabling finer control over image dimensions. It introduces additional augmentations such as sharpening and slightly stronger brightness/contrast adjustments. Moreover, Pipeline 2 normalizes images using **ImageNet statistics**, which may help align the data distribution with common pretrained models.

Both pipelines include standard augmentations like **flipping, hue/saturation adjustments, CLAHE**, and **blur**, aiming to improve robustness against varied visual conditions.The choice between these pipelines involves a trade-off between augmentation complexity, computational overhead, and potential impact on model generalization. Because landmarks are not usually be pictured upside-down so vertical flipping is used carefully with low probability. Evaluating their effects on validation performance will guide the selection of the optimal approach for landmark recognition.

6 Methodologies

In this section, we present the methodologies implemented for our landmark recognition project. We explore both baseline and enhanced models, leveraging state-of-the-art convolutional neural networks and advanced feature aggregation techniques:

- **ResNet-50**
- **EfficientNet-B2**
- **ResNet-50 + Deep Orthogonal Local Gradient (DOLG)**
- **EfficientNet-B2 + Deep Local Features (DELF)**

The first two approaches—**ResNet-50** and **EfficientNet-B2**—utilize pretrained backbone models widely adopted for image classification tasks. The latter two methods combine these backbones with local feature extraction and aggregation techniques: **DOLG**, which enhances **ResNet-50** by capturing fine-grained local details while preserving global context, and **DELF**, which augments EfficientNet-B2 by focusing on semantically meaningful local descriptors.

We selected both the **ResNet** and **EfficientNet** families due to their relatively compact model sizes, proven performance on large-scale vision tasks, and compatibility with limited computational resources. These architectures offer a good balance between efficiency and accuracy, making them suitable candidates for experimentation in our project setting.

6.1 ResNet-50

ResNet-50 is a Convolutional Neural Network (CNN) architecture that belongs to the ResNet (Residual Networks) family, a series of models designed to address the challenges with training deep neural networks, developed by researchers at Microsoft Research Asia, **ResNet-50** is renowned for its depth and efficiency in image classification tasks. ResNet architectures come in various depths, such as ResNet-18, ResNet-32, and so forth, with ResNet-50 being a mid-sized variant. Since its release, the architecture has served as a benchmark for Computer Vision task, revolutionize how deep learning models are used nowadays.

In deep neural networks, a primary problem while training models is the Degradation problem. As networks become deeper, their accuracy saturates and then degrades rapidly. This degradation is not caused by overfitting, but rather the difficulty of optimizing the training process.ResNet solved this problem using Residual Blocks that allow for the direct flow of information through the skip connections, mitigating the vanishing gradient problem. The residual block used in ResNet-50 is called the Bottleneck Residual Block. This block has the following architecture:

The original ResNet-50 model comprises 50 layers and is pre-trained on the ImageNet dataset by default. However, in our implementation, we initialized the model with pretrained=False to allow training from scratch using our specific dataset, which consists of 100 distinct target classes.

Base Architecture: ResNet-50 includes a series of convolutional and identity blocks that incorporate residual connections — a technique where the input to a layer is added to the

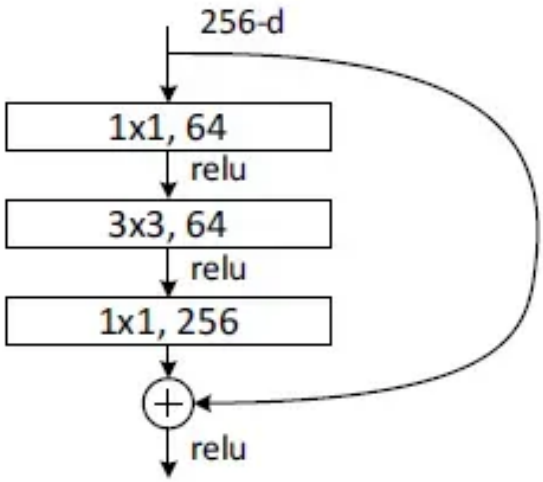


Figure 12: ResNet Residual Block

output of a deeper layer. This design mitigates the vanishing gradient problem and enables the training of deeper networks by allowing gradients to propagate directly through the skip connections.

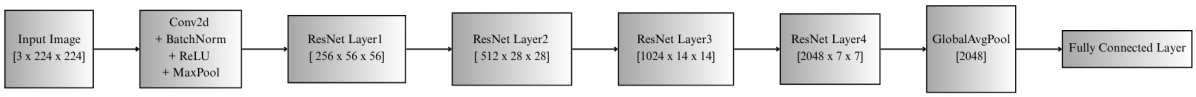


Figure 13: ResNet Architecture

To assess the influence of data preprocessing on model performance, we trained two versions of the ResNet-50 architecture, each corresponding to one of the distinct augmentation pipelines described earlier.

Both versions utilize the same modified classification head, replacing the original fully connected layer with a three-layer multilayer perceptron (MLP) composed of linear layers interleaved with ReLU activations and dropout regularization. This modification is designed to enhance the model's capacity for learning discriminative features while reducing overfitting:

- **Final fully connected (FC):** which maps the feature representation to 1,000 ImageNet classes, is replaced in our model to match our classification task with 100 output classes.
- **Linear Layer (infeatures $\rightarrow$ 512)** This layer takes the flattened output of the final convolutional block of ResNet-50 and projects it into a 512-dimensional feature space. The infeatures is determined dynamically from the original model, ensuring compatibility

- **ReLU Activation** A Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity, enabling the network to learn more complex decision boundaries.
- **Dropout** ($p = 0.5$): Dropout is used as a regularization technique to prevent overfitting by randomly deactivating 50 percent of the neurons during each training iteration.
  - Learning Rate: 1e-4
  - Batch Size: 32
  - Epochs: 20 (starting from a checkpoint at epoch 12)

The original fully connected (FC) layer at the end of ResNet-50, which maps the feature representation to 1,000 ImageNet classes, is replaced in our model to match our classification task with 100 output classes.

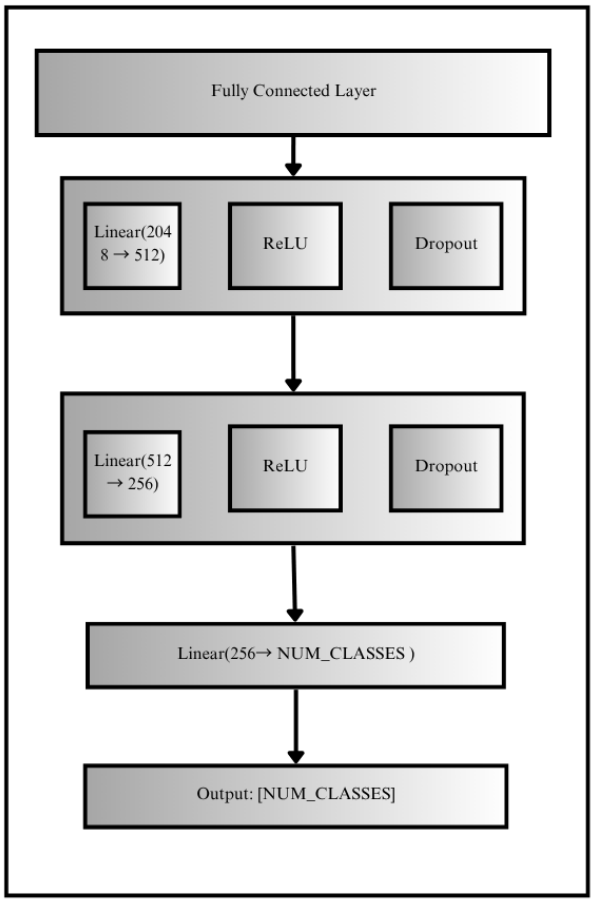


Figure 14: ResNet Fully Connected Layer

6.2 EfficientNet-B2

For our second methodogly, we use the base architecture **EfficientNet-B2**, a state-of-the-art CNN architecture known for its balance between accuracy and computational efficiency. Effi-

cientNet achieves high performance by uniformly scaling depth, width, and resolution using a compound scaling method. This enables the network to achieve strong results on image classification tasks with fewer parameters and lower computational cost compared to traditional architectures. We instantiated the model using the *EfficientNet.fromname('efficientnet-b2')* method, which constructs the architecture with pre-defined scaling parameters. The original classification layer of the network, designed for the ImageNet task with 1,000 classes, was modified to suit our specific task.

- **Architecture:** EfficientNet-B2 pretrained on ImageNet
- **Loss Function:** CrossEntropyLoss
- **Optimizer:** AdamW
- **Hyperparameters:**
  - Learning Rate: 1e-4
  - Batch Size: 32
  - Epochs: 20 (starting from a checkpoint at epoch 12)

Model training and performance tracking were handled using Weights & Biases (W&B), allowing for detailed analysis across epochs. This EfficientNet-B2 model offers a lightweight yet powerful alternative to traditional deep CNNs. By customizing the final classification layer, we adapted it for our 100-class classification task. The use of pre-trained weights from prior training enables better convergence and potentially higher accuracy during evaluation or additional training. EfficientNet's design also allows it to generalize well with fewer parameters, making it particularly suitable for applications where computational efficiency is essential.

6.3 EffieicentNet-B2 wtih DEep Local Features (DELF)

In this work, we integrate EfficientNet-B2 with **Deep Local Features (DELF)** to enhance the model's ability to recognize fine-grained spatial patterns crucial for tasks like landmark recognition. This hybrid design captures both global semantic information and localized discriminative features, resulting in a more robust and context-aware model. Unlike traditional local descriptors such as SIFT, DELF is learned end-to-end and tailored for visual recognition tasks. The model itself, is available on TensorFlowHub, was trained with Google-Landmark dataset, and additional query images optimized for landmark recognition. It identifies and encodes regions in the image that are both spatially distinct and semantically relevant. For each image input to the model, DELF will return featrue descriptors and feature locations,

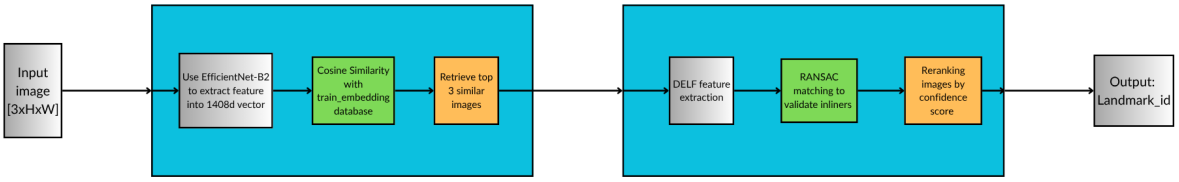


Figure 15: Architecture of the method EfficientNet + DELF with 2 phases Embedding and Re-ranking

Embedding Stage: The embedding phase involves extracting features from an image and measuring the similarities between the input image and existing training images.

- First, apply the trained EfficientNet-B2 model. The input image is decomposed, and its features are extracted into a 1408-dimensional vector.
- Next, extract features from all images in the training dataset to create an embedding database containing the 1408-dimensional vectors of the training images.
- Compute the cosine similarity between the input image vector and the vectors in the database. Cosine similarity quantifies the similarity between two vectors based on the cosine of the angle between them.
- Retrieve the top 3 most similar images and use these as input for the re-ranking phase.

Reranking Stage In the re-ranking phase, further processing of the top 3 retrieved images takes place.

- Extract local features from the top 3 images.
- Re-rank the images to find the one that is closest to the input image.
- Return the corresponding landmark ID of the closest image.
- Apply DELF to extract key features from the images.
- Use RANSAC (Random Sample Consensus) to find the most reliable key features that represent the landmark.

float

Figure 16: Extracting keypoints in an image

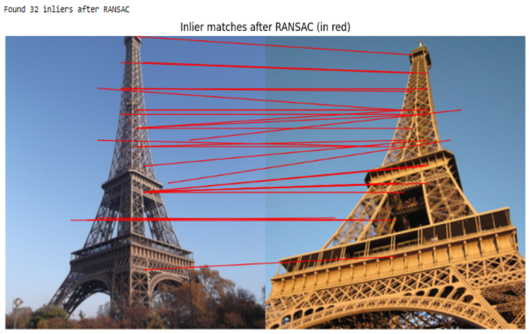


Figure 17: Images Inliers after RANSAC

6.4 ResNet50 with Deep Orthogonal Local and Global Features

To enhance the discriminative power of the backbone network, we augment ResNet-50 with a Deep Orthogonal Local and Global Features(DOLG) module. DOLG is a technique designed to combine global semantic features and local fine-grained features in an orthogonal manner, improving performance in visual recognition tasks. asks, especially those involving landmark or fine-grained instance recognition.. It consists of a local and a global branch for learning two types of features jointly and an orthogonal fusion module to combine them. In detail, the local components orthogonal to the global feature are decomposed from the local features. Subsequently, the orthogonal components are concatenated with the global feature as a complementary part. Finally, it is aggregated into a compact descriptor. With our orthogonal fusion, the most critical local information can be extracted and redundant components to the global information are eliminated, such that local and global components can be mutually reinforced to produce final representative descriptor with objective-oriented training. To enhance local feature learning, inspired by lessons from prior research, the

local branch is equipped with multi-atrous convolutions and self-attention mechanisms to attentively extract representative local features.

1. Global features that capture high-level semantic information across the entire image.

2. Local features that focus on spatially localized, detailed patterns such as corners, textures, or edges.

3. Orthogonal Fusion of Local and Global branch: The key innovation in DOLG lies in orthogonalizing these two feature types to reduce redundancy and ensure that local features do not overlap with global semantics. This is achieved through orthogonal regularization, enforcing independence between the local and global representations.

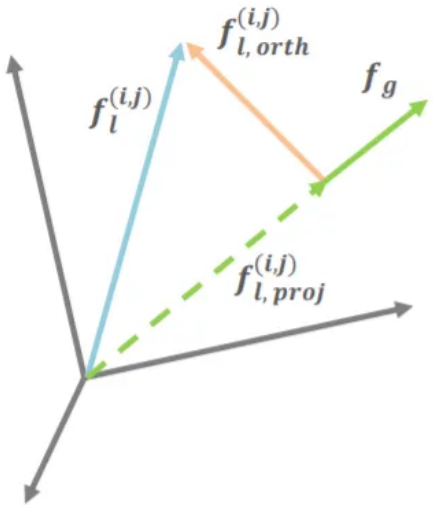
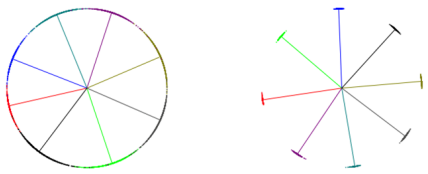


Figure 18: Orthogonal Fusion of Local and Global branch

We utilize ResNet50, a deep residual convolutional neural network, as the feature extraction backbone. ResNet50 is pretrained on ImageNet and is known for its ability to learn rich semantic representations through the use of residual connections. The convolutional layers in ResNet50 extract hierarchical features from input images that capture both low-level textures and high-level semantic patterns.

- **A local feature attention block** convolution + ReLU + normalization) to capture discriminative patches.
- **A global average pooling layer** to summarize high-level semantic information.
- **Both branches (local and global)** are projected to a shared embedding space, and an orthogonality constraint is applied during training.
- **Hyperparameters:** Finally, the two orthogonalized vectors are fused (e.g., via concatenation or weighted sum) and passed to the classification head.
- **ArcFace Head for Classification (Additive Angular Margin Loss) :** Instead of a conventional linear classifier, we employ an ArcFace head, a metric learning loss

function that introduces an additive angular margin to the softmax function. ArcFace, improved from softmax loss function, projects both features and weights to a hypersphere and optimizes angular distances between them. This approach improves inter-class separability and intra-class compactness, which is especially valuable for landmark recognition where different classes may share similar visual patterns. Formally, ArcFace modifies the decision boundary of softmax by enforcing a margin m in the angle space:



(a) Softmax (b) ArcFace

Figure 19: Visualization differences between Arcface and Softmax loss

Integrating DOLG with ResNet-50 allows the model to learn more expressive, disentangled representations, significantly boosting its effectiveness in complex image recognition tasks such as landmark classification

7 Handling Out-of-Distribution (OOD) Images

Building an effective landmark recognition system requires not only accuracy but also robustness and reliability. The models described in the previous section are trained under the assumption that every input image contains a landmark—possibly occupying only a small portion of the image. However, in real-world scenarios, this assumption may not always hold true. Input images may not always include landmarks, and may contain irrelevant content or completely unrelated scenes.

This issue is particularly relevant to the Google Landmark Recognition competition, where the public test set includes a significant number of non-landmark images mixed in with landmark images. To address this challenge, we propose two methods aimed at distinguishing between images that contain landmarks and those that are irrelevant (i.e., out-of-distribution or non-landmark images)

7.1 Out-of-distribution dataset

As we mention above, we want our system to be able to distinguish between an image possibly containing a landmark and those which possibly do not contain one. In order to do that, we also collect non-landmark data using 3 different datasets: **Caltech101**, **CelebA**, and **SUN397**. All of threes are famous, frequently used dataset for computer vision tasks:

- **Caltech101**: The Caltech-101 dataset contains 9,146 images across 101 object categories (e.g., “helicopter”, “elephant”, and “chair”) along with a ”background” cate-

gory. Most classes include about 50 images, with some ranging from 40 to 800. Image resolution is approximately 300×200 pixels. All images were used in this work.

- **CelebA**: A curated subset of the CelebFaces Attributes Dataset (CelebA), commonly used for facial recognition and synthesis. We used 10,000 images of celebrity faces, varying in pose, lighting, and background.
- **SUN397**: The SUN397 (Scene UNderstanding) dataset includes 397 scene categories with a total of 108,754 images. We selected a subset of categories relevant for non-landmark classification, totaling approximately XXXX images.

Dataset	**Description**	**Number of Images Used**
Caltech101	Contains 9,146 images from 101 object categories (e.g., "helicopter", "elephant", and "chair") plus a background category. Most classes have around 50 images.	9,146 images used as non-landmark data.
CelebA	Subset of the CelebA dataset for face recognition tasks. Includes images from various identities with diverse backgrounds, poses, and facial attributes.	10,000 face images used as non-landmark data.
SUN397	Large-scale scene classification dataset with 397 scene categories. We selected a subset of categories such as `amusement_arcade`, `art_school`, `beauty_salon`, `parking_lot`, `playground`, and `street`.	673 images used from selected categories.
Total	–	Approximately [INSERT TOTAL] images

Table 2: Datasets used to build the Non-Landmark dataset for Out-of-Distribution detection.

To balance between the number of landmark and non-landmark images, we also use a similar number of landmark - contained images: with 19783 images (About 70 images/landmark) with 19380 non-landmark images.

7.2 Binary Classifier for Landmark Detection

Our first approach involves training a dedicated binary classifier to separate landmark images from non-landmark images. By framing this as a binary classification problem, the model learns to identify whether a given image contains a landmark or not. This pre-filtering step can then be used to improve the overall system's reliability by rejecting irrelevant inputs before attempting landmark recognition.

We trained the binary model using a pretrained **MobileNetv2** with the classifier removed for binary classification. We use BCEWithLogitsLoss that inlcudes a Sigmoid internally.

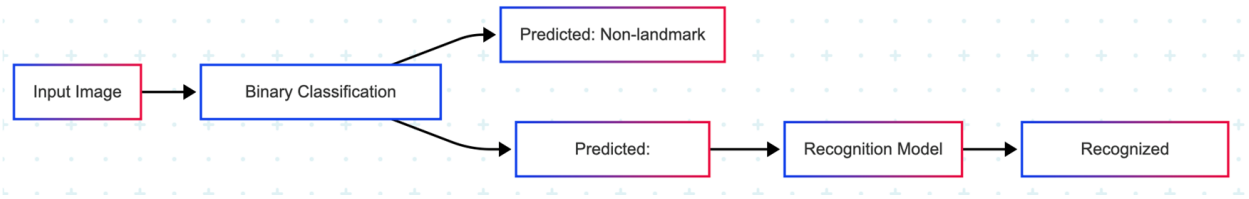


Figure 20: Binary Classification Architecture

7.3 Implementing ODIN: Out-of-Distribution Detector for Neural Networks

As a complementary approach, we implement **ODIN (Out-of-Distribution detector for Neural networks)**, a state-of-the-art method designed to detect whether an input sample belongs to the training distribution or not. ODIN leverages temperature scaling and input perturbations to enhance the confidence scores of in-distribution samples while reducing those of OOD samples, thus providing an effective mechanism to identify non-landmark images.

Figure 21: Binary Classification Architecture

7.4 Feature Matching using Cosine Similarity

The third and final approach based on the process of feature extraction that has been used in the above 2 methods of DOLG and DELF.

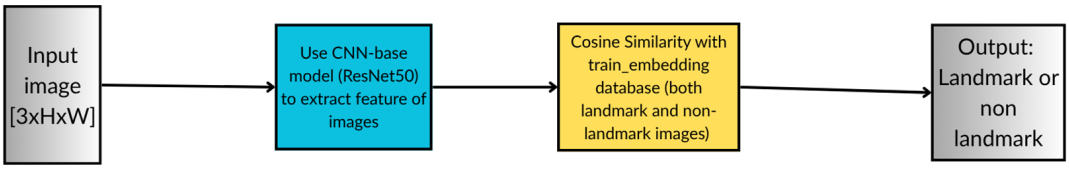


Figure 22: Feature Matching pipeline

Implementation: From the input image, we use ResNet50 (a CNN base model) to extract the features of the image into a vector space. Then calculate the cosine similarity of the input vector with the train_embedding database, where we previously extracted the features of landmark and non-landmark images. The results is an image we believe that its features have the closest similarity with the input image, and we use that image to identify which class the input image belongs to.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 23: Cosine Similarity Formula

8 Training & Evaluation

In this section, we discuss about our training process, how we train the models, and the evaluation methods to determine the best model configurations for the task.

8.1 Training process

Training settings: We trained our models on the Kaggle platform, not only because the competition was hosted on Kaggle, but also because we can transport the large dataset more conveniently. Kaggle provided two GPU accelerator choices.

8.1.1 ResNet50

We trained the ResNet50 model on both augmentation pipelines

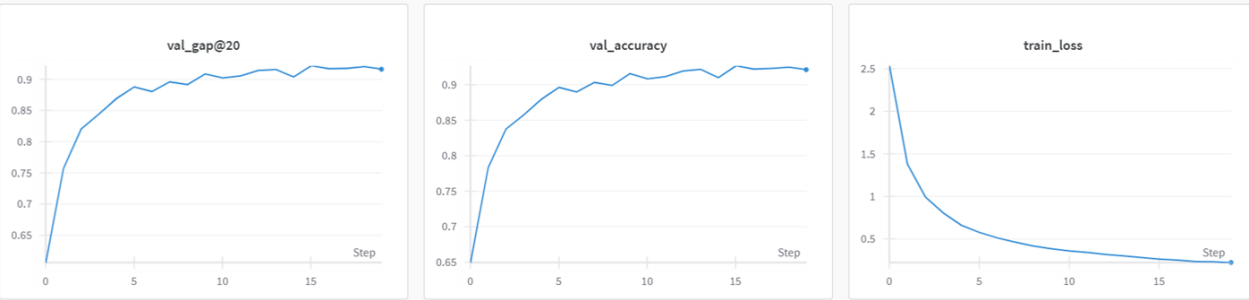


Figure 24: ResNet50 pipeline 1 training process

8.1.2 EfficientNetB2

Training proceeded from an existing checkpoint, starting at epoch 12. During each epoch:

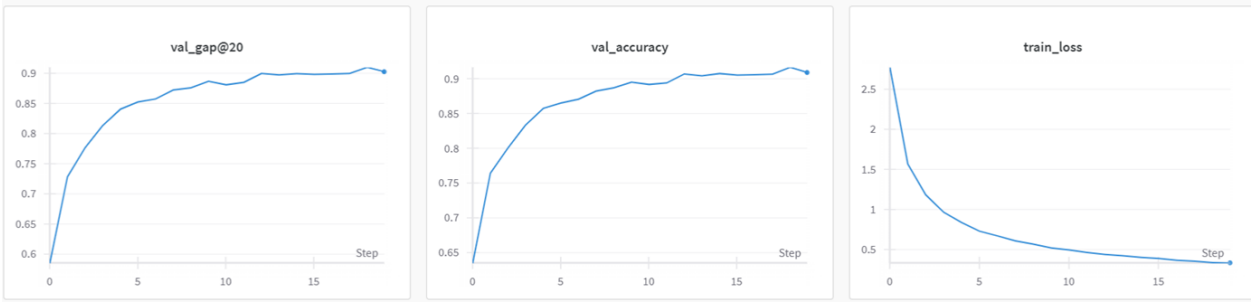


Figure 25: ResNet50 pipeline 2 training process

- **Training:** Loss was computed and backpropagation was applied.
- **Validation:** Predictions were collected to compute accuracy and GAP@20.

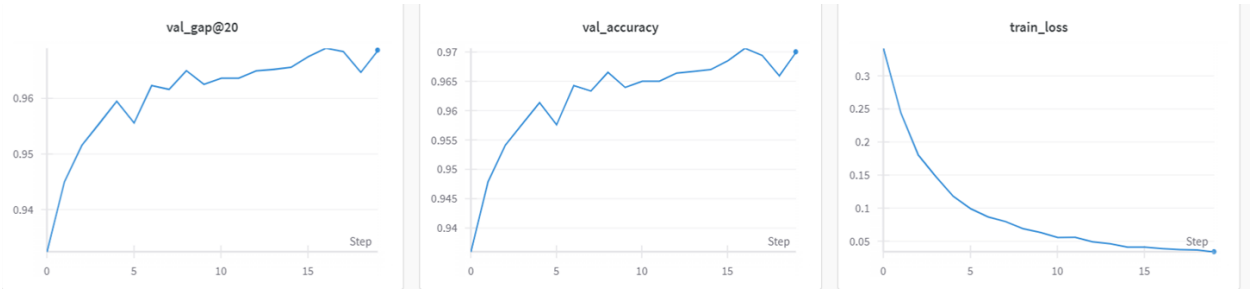


Figure 26: EfficientNet pipeline 1 training process

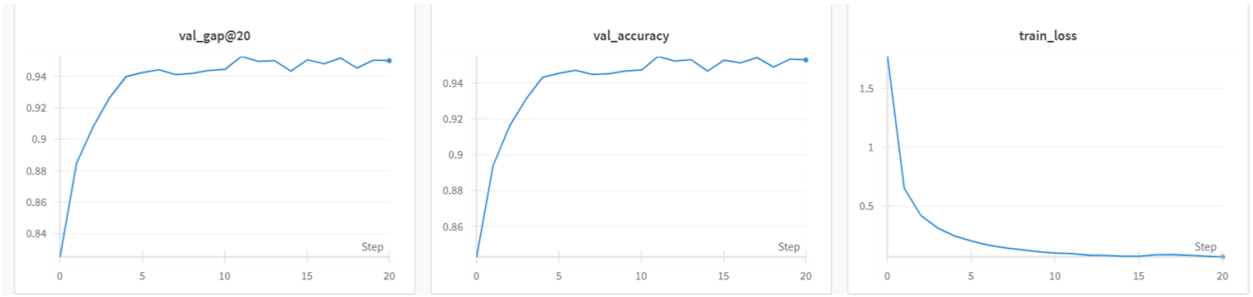


Figure 27: EfficientNet pipeline 2 training process

8.1.3 EfficientNetB2 with DELF

For the feature extraction base EfficientNetB2 model, we implemented our trained EfficientNet as described in the previous section. By doing so, we not only reduce computational

resource usage, but also highlight the potential improvements DELF can provide for a given model.

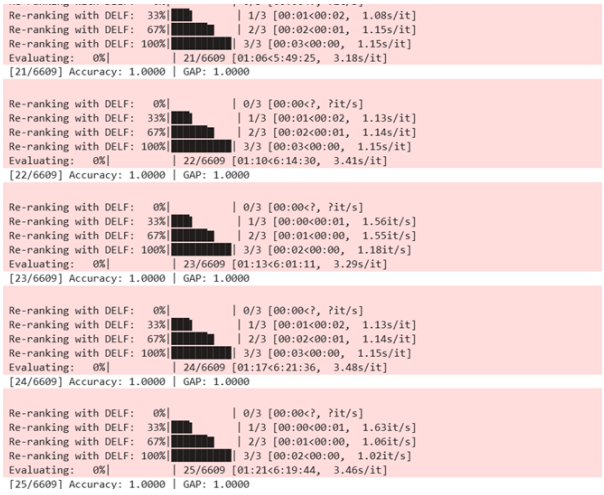


Figure 28: DELF reranking process

8.1.4 ResNet50 with DOLG

8.2 Evaluation

For the fine-grained landmark recognition task, we evaluate model performance using two main metrics: **Accuracy** and **Global Average Precision at 20** (GAP@20). These metrics provide insight into both exact matches and ranked prediction relevance.

- **Accuracy:** Accuracy measures the proportion of correctly predicted labels among the total number of predictions. It is defined as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

  This metric gives a direct sense of how often the model selects the correct landmark class as the top prediction.

- **GAP@20 (Global Average Precision at 20):** GAP@20 is a ranking-based metric used in retrieval and classification tasks to measure how well the top-20 predictions are ordered by confidence. This metric is the primary evaluation metric provided by the competition. It is calculated as:

$$\text{GAP@20} = \sum_{i=1}^{N} p(i) \cdot \Delta r(i)$$

  Where:

  – N is the number of predictions (up to 20 per sample),

- $p(i)$ is the precision at rank i,
- $\Delta r(i)$ is the change in recall at rank i.

GAP@20 is especially valuable in this task because test samples may have multiple plausible landmark predictions. The metric rewards models that not only predict correctly, but also rank relevant predictions higher.

The model was checkpointed at every improvement in GAP score, ensuring that the best-performing version was retained.

9 Results and Discussion

Overall, all the methods perform quiet decently, achieve high results both in accuracy and GAP@20 score. The accuracy of Efficient-net family model is higher than Res-net family model and the use of DELF and DOLG algorithm does increase the performance of the base model itself.

Performance metrics:

- **Best Validation Accuracy:** 0.9549
- **Best GAP@20:** 0.9527

Model	**Accuracy (%)**	**GAP@20**
EfficientNet pipeline 1 + DELF	[0.9845]	[0.983]
ResNet pipeline 2 + DOLG	[0.9300]	[0.9320]
EfficientNet pipeline 1	[0.97]	[0.9687]
EfficientNet pipeline 2	[0.9549]	[0.9527]
ResNet pipeline 1	[0.9300]	[0.9320]
ResNet pipeline 2	[0.9088]	[0.9161]

Table 3: Comparison of model performance on the landmark recognition task.

Observations:

- Data augmentation significantly improved generalization.
- GAP@20 proved to be a more reliable metric than accuracy due to the multi-class nature and imbalance.
- The use of W&B enabled precise tracking of overfitting and performance fluctuations.

Model	Accuracy	Precision	Recall
MobileNet Binary Classifier	0.8915	0.8492	0.9520
Odin Score	0.5400	0.5400	0.5500
Feature Extraction	0.9585	0.9600	0.9600

Table 4: Comparison of different models on classification performance metrics.

For the classification task, the use of Feature matching provides the best results with a trade off of high computational resources. The use of ODIN performs the least effective could be due to the fact the ODIN was originally used for multi-class implementation, not for binary

10 Conclusion

This project successfully applied a number of methods model for landmark recognition. Key takeaways include:

- Transfer learning is highly effective for complex visual tasks.
- GAP@20 is a suitable metric for retrieval-like classification.
- Future work could explore ensemble models, advanced metric learning, or multimodal data (e.g., GPS + image).

The approach and results offer a strong baseline for future improvements and real-world deployment in landmark identification systems.

11 Demonstration

We have a live demonstration of our landmark recognition models and binary classification models on Gradio, a platform to deploy machine learning models live on local host and a https url, which results in example below

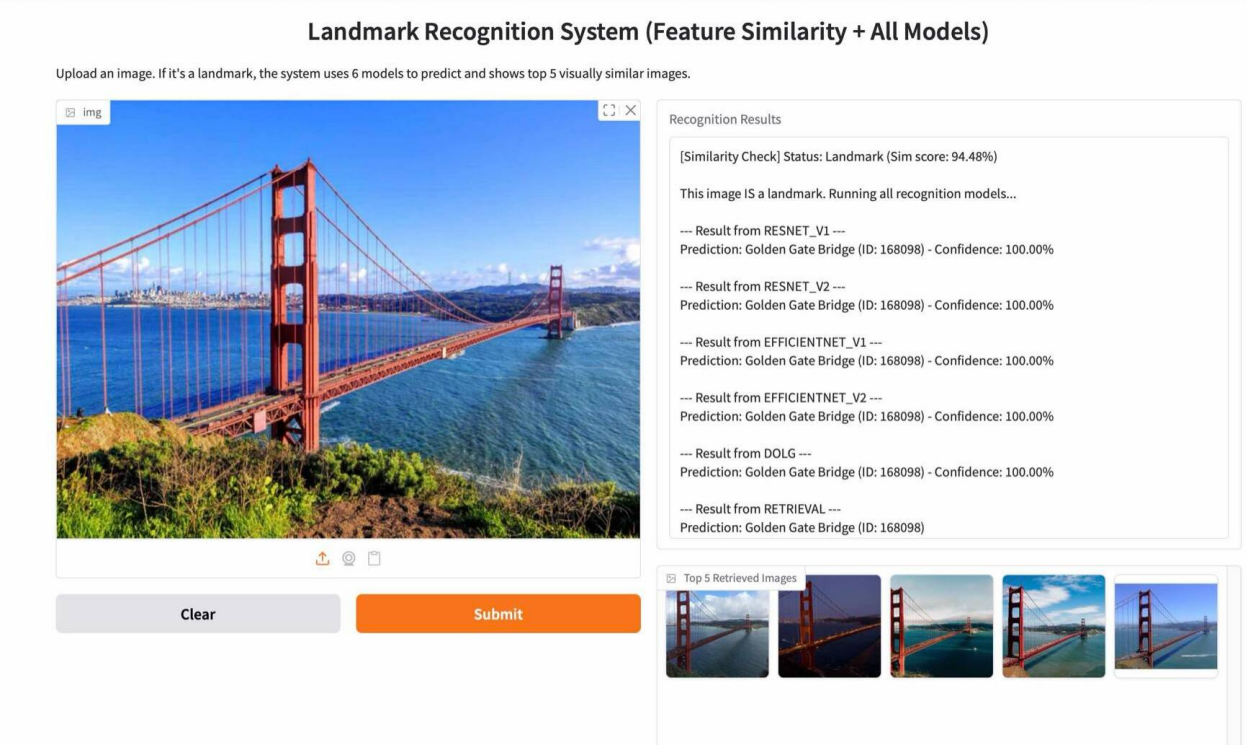


Figure 29: Our Demo Interface

Our demo interface has 2 main parts: *Input image* and *Output results*. The input section has a submit button where user can post the required image for the recognition process, while the Output section shows the results of the classification and recognition process. We also include the top 5 retrieved images (which is the direct results from the *DELF* method)

11.1 Classification Demo results

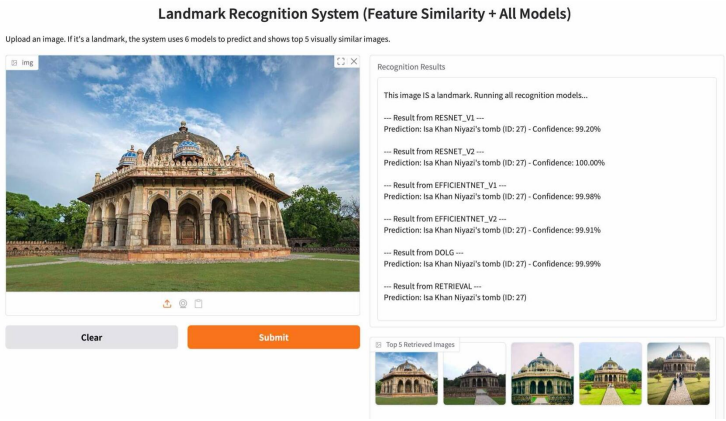


Figure 30: The results of Binary classification method

When the user submit an image, it gone through the classification progress to identify whether the image is a landmark or not. The above image is the result of the *Binary classification method*, which is our first method, and because the landmark is in our training set so it can be easily recognized as a landmark with the recognition progress on the right hand side.

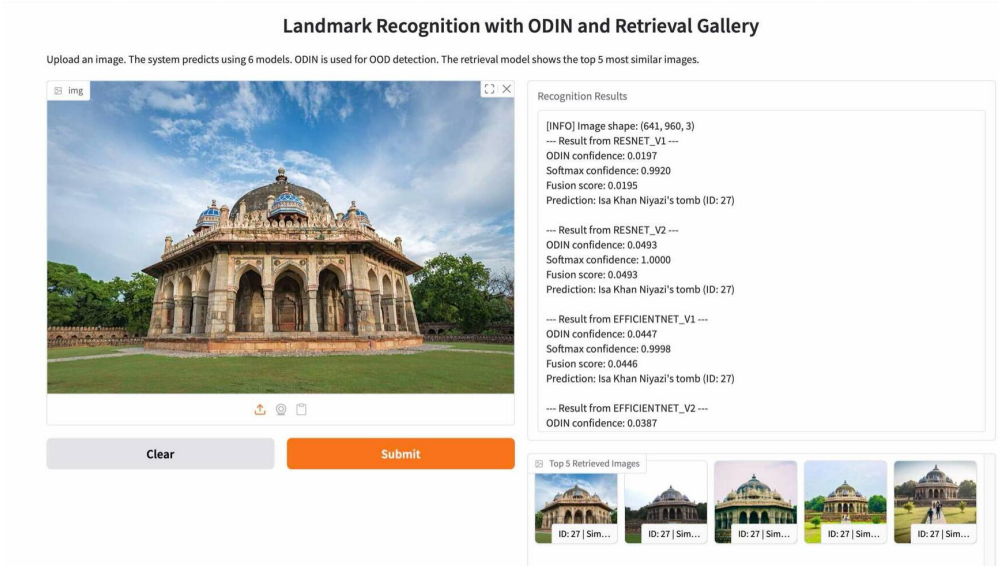


Figure 31: The results of the ODIN method

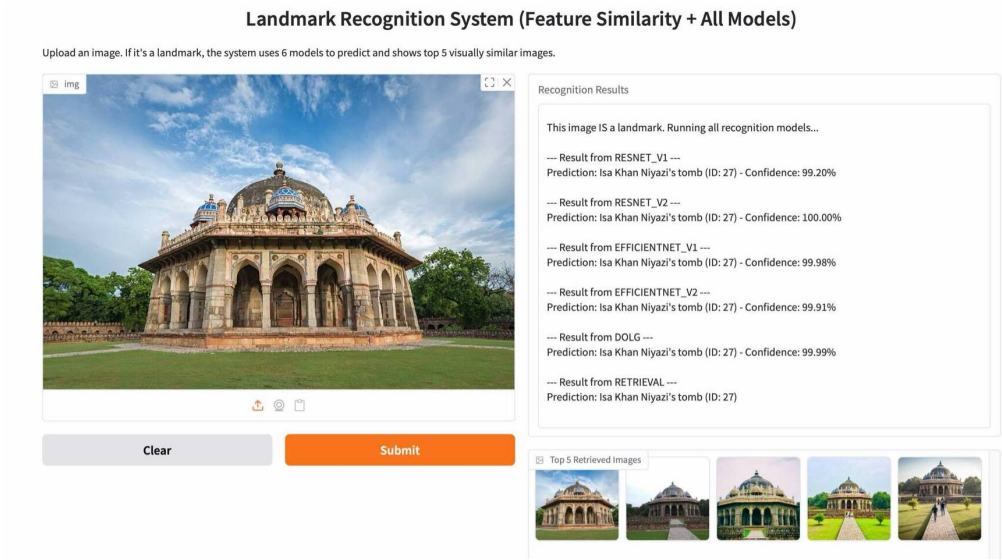


Figure 32: The results of the feature matching method

Overall, all the methods can recognize the landmarks in the dataset well. But what about cases when a landmark not in the training set, a new landmark that our model has never seen before . To test that, we try *Hanoi Opera House*, a landmark that is not available

in our training set to test the performance of our methods, and it turns out only the third method (Feature matching) could recognize it and classify it as a valid landmark.

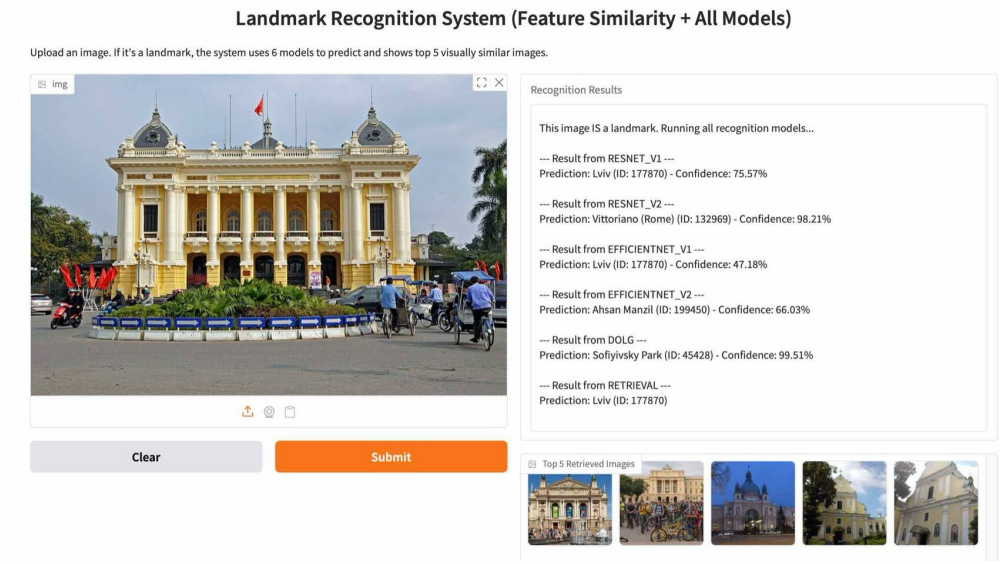


Figure 33: The results of the feature matching method for Out-of-distribution landmarks

In the above image, although by using Feature matching, the method can still recognize enough inliners point so it can classify the input image as a landmark-related image. This can due to the fact that, building-relevant landmarks could share similar architecture or color-scheme, which makes them can be connected to each other. Therefore, feature matching method can perform better than the two methods due to the fact that it could compare with the related ones.

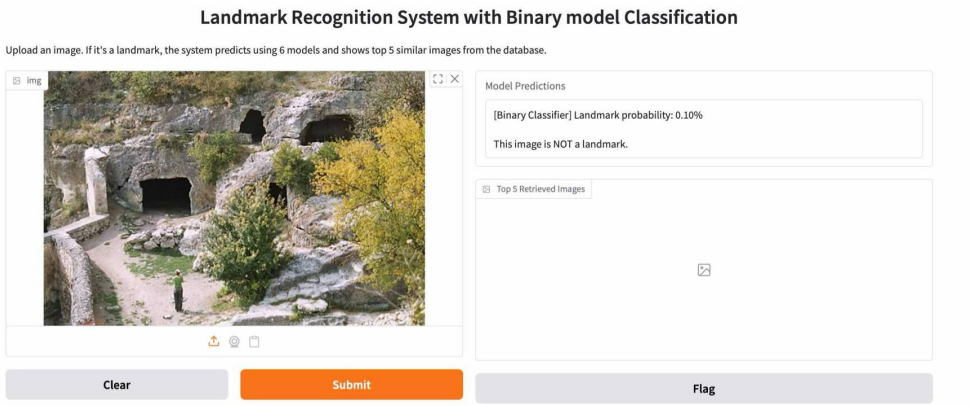


Figure 34: A fail example of Binary classification

In the above image, when we try another landmark that is not in our training set, the binary classifier fail to detect it as a landmark. It could be that the image contains a lot

of trees, surrounding objects that makes it difficult to recognize the landmark itself, hide it from out model.

11.2 Recognition Demo results

Most of non-landmark images were filtered in the classification process, so the recognition process cam show the true performance of our models.

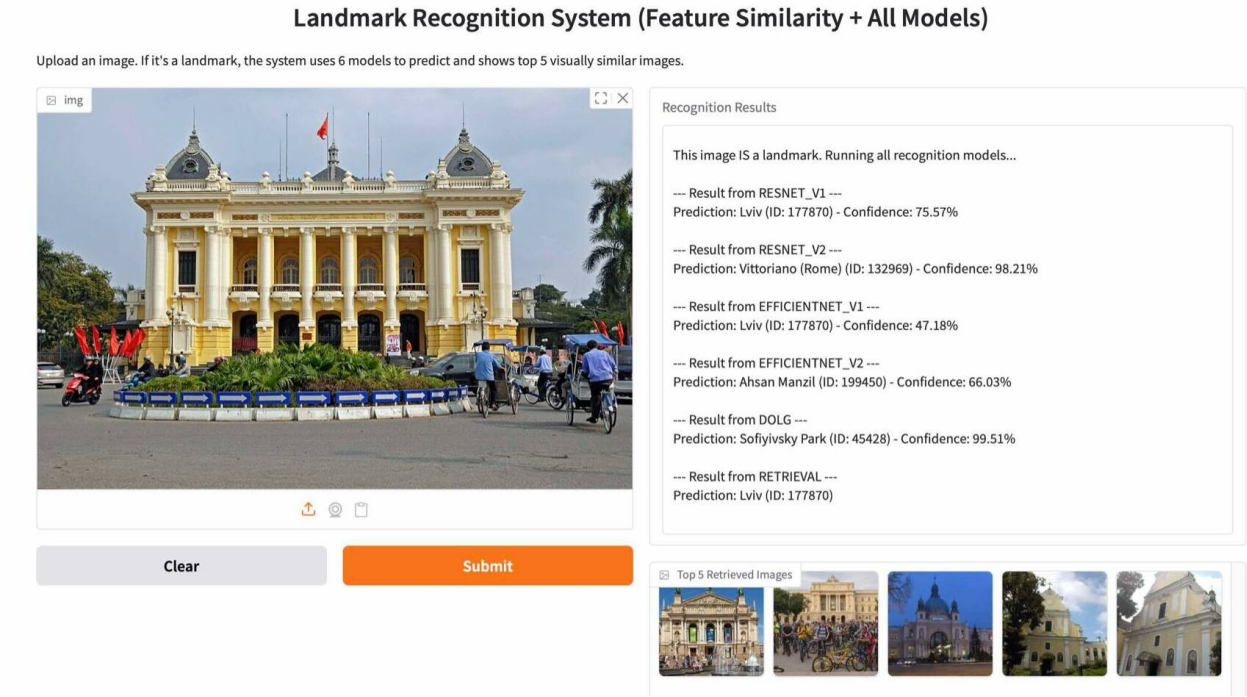


Figure 35: Out-of-distribution landmark mistakenly classify to most similar landmarks in the dataset

In the case when a out-of-distribution landmark was introduced to our model, again it was misclassify to other landmarks that share some architecture similarity, we show the top 5 images that by using feature matching, is closest to the input image. In the image, Hanoi Opera House was mistakenly classify to other landmarks such as Lviv, Ahsan Manzil, etc all have similary color scheme and architecture.

12 References

1) **Large-Scale Image Retrieval with Attentive Deep Local Features** by Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, Bohyung Han. *ICCV 2017*. `https://doi.org/10.48550/arXiv.1612.06321`.
2) **DOLG: Single-Stage Image Retrieval with Deep Orthogonal Fusion of Local and Global Features** by Min Yang, Dongliang He, Miao Fan, Baorong Shi, Xuetong Xue, Fu Li, Errui

Ding, Jizhou Huang. *ICCV 2021*. https://arxiv.org/abs/2108.02927.
3) **Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks** by Shiyu Liang, Yixuan Li, R. Srikant. *ICLR 2018*. https://arxiv.org/abs/1706.02690.
4) **A Data Geek's Guide to Recognize Landmarks** by Abhinaya Ananthakrishnan https://medium.com/@abhinaya08/google-landmark-recognition-274aab3c71ae.