Landmark Classification and Social Media Tagging

1st Jyothir Adithya Mugireddy
Department of Sciences
Western University
London, Canada
jmugired@uwo.ca

2nd Muzhi Lyu
Department of Sciences
Western University
London, Canada
mlyu25@uwo.ca

3rd Wanxuan He
Department of Sciences
Western University
London, Canada
whe224@uwo.ca

4th Yiming Hu
Department of Sciences
Western University
London, Canada
yhu822@uwo.ca

***Abstract*—The ubiquitous practice of sharing and posting photographs has become an embedded aspect of daily life, while at the same time, privacy concerns are growing. Individuals are eager to take advantage of the efficacy of applications that can identify the content of images, provide relevant tags, or promote organized arrangements. However, a common problem is the hesitation to disclose GPS information in images for security concerns.To address this need, we propose a model that aims to extract information from the visual composition of images only. Our model strives to identify potential locations by examining background landmarks, and then proposing appropriate labels for the inferred locations. This innovative approach seeks to protect user privacy while pursuing efficient image understanding.Our methodology involved training our Convolutional Neural Network (CNN) model on a selection of the most representative landmarks within the dataset. Notably, we fine-tuned the network layers with the objective of achieving a foundational accuracy level of 50%. Subsequent to this initial training phase, our approach incorporates transfer learning techniques to further augment the model's accuracy, ensuring a robust and refined performance in landmark classification.**

***Index Terms*—CNN, Landmark Classification, Machine Learning**

I. Introduction

In the realm of photo sharing and storage services, the integration of location data with each uploaded image is pivotal for unlocking advanced functionalities. This inclusion not only facilitates the development of features such as relevant tag suggestions but also automates the organization of photo collections. However, a substantial portion of uploaded photos lacks location metadata, either due to the absence of GPS capabilities in the capturing device or the intentional removal of metadata for privacy preservation.

To address this challenge, our landmark classification project employs Convolutional Neural Networks (CNNs) to create a robust landmark classifier. The central objective is to deduce location information for images based solely on the landmarks depicted within them.

This innovative approach aims to seamlessly integrate the landmark classifier into photo sharing services, thereby enhancing user experiences by providing advanced organization and exploration capabilities for image collections. The project stands as a testament to the transformative potential of leveraging deep learning techniques to enrich and optimize photo-related services.

II. Background and Related Work

In the realm of landmark recognition, the exploration of this concept is far from novel, with the Google Landmark Dataset serving as a cornerstone in various Kaggle challenges, including ”Google Landmark Retrieval 2021” and ”Google Landmark Recognition 2021,” integral components of the ”Instance-Level Recognition Workshop.” Both challenges share the common objective of classifying landmarks, with the former focusing on retrieving similar landmarks within a database, and the latter aligning closely with our approach, divulging predicted landmark names.

It is noteworthy, however, that despite our shared utilization of the same dataset, our departure lies in the deliberate selection of a distinct training dataset size. This deliberate discrepancy in training dataset size introduces the possibility of divergent model behaviors, thereby contributing to the nuanced and differentiated outcomes of our approach.

III. Research Objectives

The primary goals of our project are as follows:

1) Optimal Model Architecture
Investigate and determine the most effective architectural configuration for the landmark classification model, with a focus on balancing computational efficiency and accuracy. This involves exploring different convolutional neural network (CNN) architectures and evaluating their impact on classification performance.
2) Transfer Learning Strategies
Evaluate and optimize transfer learning techniques, specifically leveraging pre-trained models like ResNet18, to boost the landmark classification model's accuracy. Investigate how knowledge transfer from broader image datasets, such as ImageNet, enhances the model's ability to recognize landmarks in natural scenes.
3) Evaluation Metrics and Generalization
Define and employ appropriate evaluation metrics to assess the model's performance comprehensively. Investigate the generalization capabilities of the model across different datasets and scenarios, ensuring that the trained model exhibits proficiency in recognizing landmarks beyond the training set.

Fig. 1. Sample image in Train Sets

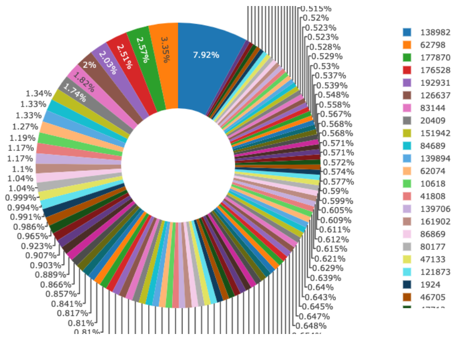

Fig. 2. Distribution of number of images in train set, where the largest category is 138982 - a photo library contributed by the ETH-Bibliothek

IV. Methodology - Data Selection

A. *Data Preparing*

We used a subset of Google Landmarks Dataset V2 as the dataset to train our model. This version of the dataset contains approximately 5 million images, split into 3 sets of images: train, index and test. Each image has been classified by the landmark category, super category, hierarchical and whether it's a human-made or natural landmark.

Identify Target Classes: Our objective is to focus on substantial classes that comprise more than 600 images, as the image count serves as an indicator of the landmark's significance. To achieve this, we organized the dataset based on their IDs and retained only those classes with more than 600 images. This process yielded 115 classes, totaling 65,000 images. Given the considerable size of over 70GB, which is impractical for efficient execution, we opted to limit each class to 300 images. Consequently, we retained 84 crucial landmark classes, resulting in a reduced dataset of 25,000 images.

Data Crawling and Classification: The CSV data exclusively contains image URLs. After several attempts, it became apparent that attempting to load data directly from these URLs was impractical. Consequently, we devised a Python script to download the images and organize them into folders based on their respective categories.

During the initial execution, we encountered a challenge where the script became unresponsive due to a corrupted URL. To address this issue, we implemented `try` and `except` blocks in the script. Additionally, we incorporated a mechanism to track the number of downloaded images. This adjustment allowed us to resume the process from the point of failure, rather than restarting it entirely in case of any interruptions. The complete execution of this process spanned approximately 1 day.

B. *Data Processing*

Load and Transform: We loaded the data using PyTorch's DataLoader. To prepare the dataset for training, we implemented a function to compute the mean and standard deviation. Prior to utilization, the data will go through a series of transformations. Initially, we resized the images to 256 pixels, followed by a random crop to 224 pixels and a random horizontal flip. These augmentations introduce noises into the dataset, mitigating the risk of overfitting. Subsequently, we converted the images to tensors and normalized them using the calculated mean and standard deviation.

Train Test Split: Before initiating training, we partitioned the dataset into training, testing, and validation sets, allocating 0.6, 0.2, 0.2 of the data to each, respectively. The training and testing sets were employed for evaluating accuracy rates and loss metrics. Meanwhile, the validation set served the purpose of model validation, exempting it from the random crop and flip transformations to ensure unbiased evaluation. All three datasets were loaded and transformed accordingly, rendering them prepared for the training phase.

Batch Visualization: We are verifying the effectiveness of our transformation and loading processes by visualizing a batch of images. Initially, we de-normalize the images, converting them to numpy format. Subsequently, we iterate through a single batch, confirming that the images are loaded and transformed as anticipated.

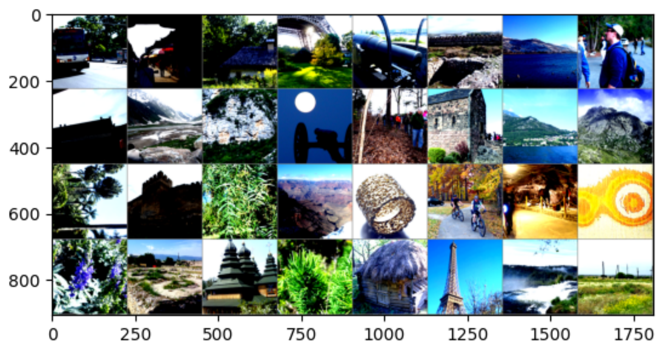

Fig. 3. Batch Visualization

Now we are ready to build models.

V. Methods - Model Building

Throughout our project, we employed a total of three models. The initial model was based on our designed architecture. Following testing with a subset of only 50 classes and a few thousand images, we trained it for 80 epochs, yet the accuracy rate plateaued at around 50%. Consequently, we developed two additional models: one utilizing ResNet and the other employing ResNeXt. Both ResNet and ResNeXt models are reaching 60% accuracy rate, and we intend to utilize these two models for making predictions.

In this section, we will discuss about the methodology of training, validation, optimization, loss Functions. Then we will provide the details of each model.

A. *Optimization*

In our pursuit of an effective loss metric for the model, we leveraged the CrossEntropyLoss, a well-suited choice for classification scenarios. This metric serves as a pivotal tool to quantify the dissimilarity between the predicted and actual classifications, crucial for guiding the model towards accurate landmark identification. To further optimize computational efficiency, we judiciously examined GPU availability, and if present, seamlessly transferred the loss computation to the GPU, thereby harnessing parallel processing capabilities.

For optimization metrics, we implemented a versatile and adaptable optimizer function, providing the flexibility to choose from a range of optimizers, such as 'SGD', 'Adam', 'Nadam', or 'Adagrad'. Each optimizer brings distinct advantages to the table. The optimizer instantiation process is intricately tailored based on the selected option, encompassing crucial parameters such as learning rate, momentum, and weight decay for regularization. This dynamic approach allows us to fine-tune the optimization strategy according to the specific requirements of the landmark classification model. Whether prioritizing speed, convergence, or adaptive learning rates, our optimization framework accommodates diverse preferences, enhancing the adaptability and performance optimization capabilities of our landmark classification model.

B. *Model Training*

The whole training process is encapsulated within the 'train model' function. This function takes inputs such as the model architecture, data loaders, loss function, optimizer, learning rate scheduler, and the number of training epochs. Performance metrics are initialized, such as the start time of training (since), a copy of the best model weights (best-model-wts), and the best accuracy achieved (best-acc). These metrics will be utilized to monitor and assess the model's performance throughout training.The model is moved to the specified computing device (CPU or GPU), and if multiple GPUs are available, the DataParallel module is employed for parallel training across GPUs. This step optimizes training efficiency.

The training process is organized into epochs, and each epoch consists of both training and validation phases. During each epoch, the model parameters are updated based on the training data, and the model's performance is evaluated on a separate validation dataset. For each phase (training or validation), the model is set to the appropriate mode (training or evaluation). Gradients are zeroed to avoid accumulation, and forward and backward passes are conducted. During training, gradients are calculated, and the optimizer updates the model's weights accordingly.

The DataLoader is utilized to iterate through batches of training or validation data. The forward pass calculates model predictions, and in the training phase, the backward pass computes gradients and updates the model's parameters.Running statistics, such as the running loss and the number of correct predictions, are maintained for each batch. These statistics are crucial for calculating the average loss and accuracy at the end of each epoch. A learning rate scheduler adjusts the learning rate during training, improving convergence and model performance. The learning rate scheduler is activated during the training phase.

For each epoch, average loss and accuracy are computed by dividing the running statistics by the total number of samples in the dataset. These metrics provide a concise summary of the model's performance during the entire epoch.The loss and accuracy for each phase (training or validation) are printed to the console, allowing for real-time monitoring of training progress.

During the validation phase, if the current accuracy surpasses the best accuracy achieved so far, a deep copy of the model's weights is stored. This ensures that the best-performing model is retained. Once all epochs are completed, the total training duration is calculated and printed. The model is then loaded with the best-performing weights before being returned.

*1) **Validation**:* Similar to training phase, the validation function assesses the performance and generalizes capabilities of our landmark classification model at the conclusion of each epoch. The function meticulously navigates through the validation dataloader. It systematically evaluates the model's performance on the validation set, offering insights into its ability to generalize beyond the training data.

Within the function, the model undergoes a seamless transition to evaluation mode, ensuring that its behavior aligns with the validation context. Then the validation loss is computed for each batch, offering a quantitative measure of the model's accuracy in predicting landmarks on unseen data.

The validation loop is executed iteratively, and the average validation loss is continuously updated, providing a nuanced metric for assessing the model's performance. This comprehensive approach ensures that our landmark classification model undergoes rigorous evaluation, fostering a robust understanding of its efficacy and potential areas for refinement. This validation process is fundamental in gauging the model's ability to generalize to real-world scenarios beyond the training data, validating its proficiency and enhancing its adaptability.

C. *Loss Function*

We have opted for the Cross Entropy Loss as our choice of loss function. This decision was motivated by the nature of our task, which involves classifying images into multiple categories. The Cross Entropy Loss is specifically designed for multi-class classification, making it well-suited to our objective. Internally, it applies the Log Soft max activation, converting raw output logits into probability distributions over classes. This not only simplifies the representation of target labels by accepting class indices directly but also ensures stable and efficient gradient computation during back propagation.Its robust handling of class imbalance further supports its suitability for our project, contributing to the overall effectiveness of our neural network training process.

D. Models

Group Designed Architecture

1) Layer Selection: The genesis of our model construction involved the establishment of a foundational structure, commencing with the configuration of three input channels corresponding to RGB images. We then incrementally expanded the convolutional layers, starting with 32 output channels, followed by successive increments to 128 and 512. This step-by-step augmentation allowed for a methodical evolution of our model, providing a structured foundation for the following subsequent layers.

2) Channels: The exploration of optimal channel configurations was a meticulous process involving extensive experimentation and evaluation. We tested various combinations of channel numbers, with a comprehensive analysis encompassing both computation time and accuracy metrics. After a thorough assessment of multiple configurations, the empirical findings converged towards a combination of 64, 128, and 256 channels. This particular configuration emerged as a balanced compromise, showcasing favorable computational efficiency without compromising on the model's classification accuracy. The empirical scrutiny of channel combinations underscores the iterative and data-driven nature of our approach in fine-tuning the CNN architecture to achieve an optimal balance between computational resource utilization and model performance.

3) ReLU: To infuse non-linearity into the model and facilitate the discernment of complex patterns, Rectified Linear Unit (ReLU) activation functions were judiciously applied after each convolutional layer. This activation choice is rooted in its proven efficacy in enabling the network to learn intricate representations, vital for the discernment of nuanced features within the dataset.

4) Max Pooling: Crucial to the spatial dimension reduction within the network, max pooling layers were strategically introduced after each convolutional layer. Characterized by a 2x2 kernel size and a stride of 2, these layers played a pivotal role in downsizing the spatial dimensions while retaining salient features. This reduction not only aids in computational efficiency but also ensures the retention of essential information for subsequent layers.

5) Fully Connected Layers: Transitioning from the convolutional layers to the classification phase, fully connected layers were seamlessly integrated. The meticulous calculation of the input size for the first linear layer, 256 x 28 x 28, was achieved by flattening the feature maps derived from the last convolutional layer. This process ensures compatibility between the preceding convolutional layers and the ensuing classification layers, thereby maintaining a coherent flow of information.

6) ReLU and Dropout: Post the first linear layer, the introduction of a ReLU activation function served to reintroduce non-linearity into the model. This deliberate choice contributes to the network's ability to capture intricate dependencies and relationships within the data. Simultaneously, a dropout layer with a probability of 0.7 was strategically inserted into the architecture to enhance the model's generalization capacity by randomly deactivating connections during the training phase, thereby mitigating the risk of overfitting.

7) Output Layer: Culminating in the output layer, a linear layer with output features aligning with the specified number of classes was introduced. This final layer produced classification logits, embodying the culmination of the model's learning process and enabling it to make informed predictions based on the learned features.

8) Result: This architecture only achieved an accuracy rate of 50% after 80 epochs running using a small dataset. So we decided to discard it and move on to stronger Models

Resnet Model

ResNet, short for Residual Networks, is a type of deep neural network architecture designed to address the vanishing gradient problem in very deep networks. It was introduced by Kaiming He, et al., in their paper titled "Deep Residual Learning for Image Recognition," presented at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Key features of the ResNet model include:

Residual Blocks: The core innovation of ResNet is the use of residual blocks. Traditional neural networks stack layers on top of each other, and during training, as the network gets deeper, the gradient may diminish, leading to difficulties in updating weights and learning. Residual blocks introduce a "skip connection" or "shortcut" that allows the gradient to bypass one or more layers, mitigating the vanishing gradient problem.

Identity Mapping: The skip connection in a residual block enables the learning of an identity mapping, making it easier for the network to learn the residual (difference) between the input and the output of a block. This simplifies the learning process for deep networks.

Applications: ResNet architectures have achieved state-of-the-art results in various computer vision tasks, particularly image classification, object detection, and image segmentation. The residual learning concept has been influential and has inspired the design of subsequent neural network architectures.

Pre-trained Models: Pre-trained versions of ResNet on large datasets like ImageNet are available, allowing researchers and practitioners to use transfer learning for tasks with limited labeled data.

We trained using parallel GPUs and after 3 epochs of running. The accuracy rate is more than our own designed model.

The table below summarizes the key metrics for each epoch:

Epoch	Train Loss / Acc	Validation Loss / Acc
0	2.3474 / 0.4268	1.9441 / 0.5048
1	1.6668 / 0.5678	1.7284 / 0.5494
2	1.4436 / 0.6177	1.6677 / 0.5598

TABLE I
TRAINING RESULTS

The training loss steadily decreased from 2.3474 in the first epoch to 1.4436 in the third epoch, indicative of the model's learning progress. Simultaneously, training accuracy improved, reaching 61.77 % in the third epoch. Notably, the validation set mirrored this trend, with both loss and accuracy showing positive trends. The highest validation accuracy, 55.98%, was achieved in the final epoch, suggesting that the model generalized well to unseen data. The complete training process spanned 41 minutes and 10 seconds, highlighting the computational resources required for training. Overall, the observed trends in loss reduction and accuracy improvement demonstrate the effectiveness of the training strategy in enhancing the model's predictive capabilities.

ResNeXt Model

Deep learning has revolutionized the field of computer vision, and the ResNeXt model stands out as a significant milestone in this journey. Building on the foundation laid by the ResNet architecture, known for its deep neural networks and skip connections, ResNeXt introduces an innovative concept: cardinality.

ResNet marked a breakthrough with its residual blocks, simplifying the training of deep networks. However, ResNeXt takes this a step further by incorporating cardinality, a measure of the number of paths within a block. This addition allows for richer feature representations, a crucial factor in the effectiveness of image classification tasks. This model is distinctive for its integration of the cardinality concept, essentially an expansion of the group idea in network design. In ResNeXt, different cardinalities represent various subspaces, each capable of learning more diverse representations. Compared to the original ResNet or Single-head Attention mechanisms, ResNeXt, along with Multi-head Attention, demonstrates superior representational capabilities.

We initiated the ResNeXt model by integrating pre-trained weights sourced from a comprehensive image dataset. The inclusion of pre-trained weights is pivotal as it allows the model to utilize knowledge garnered from a wide range of images, establishing a robust foundation for our specific task of image classification. The architectural design of the ResNeXt model is intricate, encompassing a deep neural network that consists of convolutional layers and an array of residual blocks. To tailor the model to our dataset's needs, we adapted the final fully connected layer to align with the number of classes present in the dataset.

For the training procedure, we opted for the Adam optimizer, setting the learning rate at 0.001. We implemented a learning rate scheduler to fine-tune the learning rate during training, with a step size configured at seven epochs and a gamma value of 0.1. The training leveraged the cross-entropy loss function, an optimal choice for multi-class classification tasks. We conducted the training over five epochs, with each epoch encompassing a complete iteration through the entire training dataset. To efficiently monitor the training progression, we employed a tqdm progress bar, which offered real-time insights into batch-wise accuracy and loss.

In the evaluation phase, after the completion of each epoch, we assessed the model using the validation set. This step was crucial to evaluate the model's ability to generalize and perform on unseen data. We diligently recorded key metrics like accuracy and loss during both the training and validation phases to meticulously track the model's performance.

The results and analysis revealed significant findings. During the training phase, the ResNeXt model exhibited impressive improvements in accuracy, ultimately achieving a final accuracy of 65.5% after the completion of five epochs. We noticed a consistent decrease in training loss over the epochs, a clear indication of the model's effective learning in minimizing prediction errors.

Regarding the validation performance, the model's accuracy on the validation set displayed a similar upward trend, reaching 59.3% after five epochs. This slight disparity between the training and validation accuracies is not uncommon, and it actually demonstrates the model's capability to generalize effectively to new, unseen data.

The decreasing pattern observed in the validation loss is a strong indicator that the ResNeXt model is not overfitting to the training data. This trend is a positive sign, as it implies that the model is learning generalizable patterns rather than memorizing the training dataset.

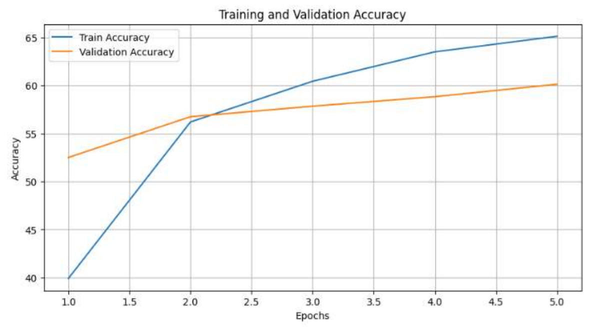

Fig. 4. Training and Validation Accuracy

During the training of our ResNeXt model, we observed a significant progression in loss reduction across the training and validation phases. Starting at a higher training loss of 2.50 and a validation loss of 1.84 in the initial epoch, the model initially demonstrated room for improvement in handling both datasets. As training proceeded, a steady decline in both loss metrics was evident. This descending trend is indicative of the model's increasing proficiency in discerning and learning relevant patterns within the training data, thereby enhancing its predictive accuracy. By the time we reached the concluding epoch, the training loss had substantially diminished to 1.27, and the validation loss had registered at 1.61, confirming the model's continued advancement and its capability to generalize well from the training to the validation dataset.

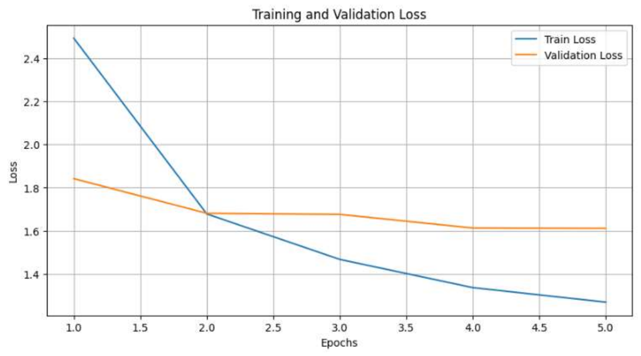

Fig. 5. Training and Validation Loss

In order to improve time efficiency, the function dynamically transfers the model to the GPU when available, capitalizing on parallel processing capabilities for expedited computations. Following this, the model is adeptly set to training mode, signaling the commencement of the intricate training procedure.

Within the training loop, each batch from the designated dataloader is systematically traversed, with progress visually tracked using the tqdm library. At the core of each iterations, the data goes through sequence of operations. Firstly, gradients of all optimized variables are expunged to forestall any inadvertent accumulation from preceding iterations. A forward pass through the model ensues, generating predictions that are subsequently juxtaposed with actual target values to compute the loss. Post this, gradients of the loss with respect to model parameters are diligently calculated, paving the way for the subsequent crucial step— a single optimization iteration that systematically updates the model parameters based on these gradients.

This meticulous process unfolds iteratively for each batch, progressively honing the model's parameters and elevating its predictive acumen. Throughout the epoch, the average training loss serves as a dynamic metric, undergoing continuous updates to offer a comprehensive view of the evolving performance landscape of our landmark classification model. This intricate choreography ensures a methodical and dynamic training trajectory, positioning our model for optimal performance in subsequent classification tasks.

E. ***Final Decision***

After comparing the three models, we have chosen ResNext as our final model. While Resext exhibits a similar accuracy rate, ResNext was selected due to better performance. Despite not being able to train the ResNet model for more epochs, the result is tending to overfitting, so we do not think it will have better performance after several more epochs.

VI. Methodology - Transfer Learning

In the context of our landmark classification project, the incorporation of transfer learning emerges as a pivotal strategy to augment model accuracy. Transfer learning involves leveraging knowledge gained from pre-trained models on vast datasets and applying it to a target task'– for now, landmark classification. This methodology is particularly potent when confronted with limited labeled data for the specific task at hand.

Transfer learning facilitates the initialization of our Convolutional Neural Network (CNN) with weights and features learned from a pre-existing model, typically trained on a diverse dataset. These pre-existing models, often trained on large-scale image datasets, have acquired a wealth of knowledge in discerning intricate patterns and features. By initializing our model with these learned weights, we equip it with a foundation that already possesses a nuanced understanding of visual elements and hierarchical representations.

A. *Methods - ResNet*

For the base of model, we used ResNet18, the architecture that stands out for its commendable performance on ImageNet, showcasing the ability to discern intricate visual features while maintaining a relatively compact size. The choice of ResNet18 aligns with the principle of efficiency—its 18-layer architecture strikes a balance between computational complexity and effectiveness in image classification tasks.

Crucially, ResNet18's prior training on ImageNet proves advantageous for our landmark classification endeavor. Both ImageNet and our landmark task involve images of natural scenes, sharing contextual similarities. By leveraging the knowledge embedded in ResNet18 from its ImageNet training, we harness a pre-trained model that already possesses a nuanced understanding of diverse visual elements in natural scenes. This transfer of knowledge accelerates our model's learning process, making it adept at recognizing landmarks within the specific context of natural scenes.

B. *Methods - ADAM Optimizer*

we have opted for 'Adam' as the selected optimizer, as it adapts the learning rates of individual parameters, providing an adaptive and efficient approach to gradient-based optimization. The key strength of Adam lies in its ability to dynamically adjust learning rates for each parameter based on the historical gradients, offering robustness across various types of data and architectures.

The decision to use Adam is driven by several factors. Firstly, Adam often requires less manual tuning of hyperparameters compared to traditional optimizers like SGD. This adaptability is especially beneficial in scenarios where hyperparameter tuning may be resource-intensive or time-consuming.

Moreover, Adam tends to perform well in practice across a diverse range of tasks. Its incorporation of momentum, which helps accelerate convergence, and adaptive learning rates contribute to its effectiveness in training deep neural networks.

VII. Predictions and Results

In the process of evaluating our image classification model, we performed inference on an image obtained from a specified URL. The image was preprocessed using a series of transformations, including resizing and center cropping, to ensure compatibility with our model. Subsequently, the model, operating in evaluation mode, provided predictions for the top-K classes along with their associated probabilities.

The image, accessible through the provided URL, triggered predictions for the top-5 classes. The model's output, in the form of softmax probabilities, was post-processed to map class indices back to human-readable class names using a predefined mapping (class_to_idx). To enhance the interpretability of the predictions, we incorporated a visual representation by displaying the original image alongside the top-5 predicted class labels and their corresponding probabilities.

For the specific image at the provided URL, the top-5 predictions are as follows:

URL:https://www.gettyimages.ca/detail/photo/eiffel-tower-royalty-free-image/98038911.

These predictions showcase the model's ability to recognize and rank potential classes associated with the input image. The visual representation serves as a valuable tool for comprehending the model's outputs and assessing the confidence levels assigned to each prediction. This iterative process of inference and visual feedback contributes to a deeper understanding of the model's performance and can be important in refining its capabilities. Adjusting parameters, such as the input image or the top-K parameter, enables a flexible exploration of the model's predictive behavior.

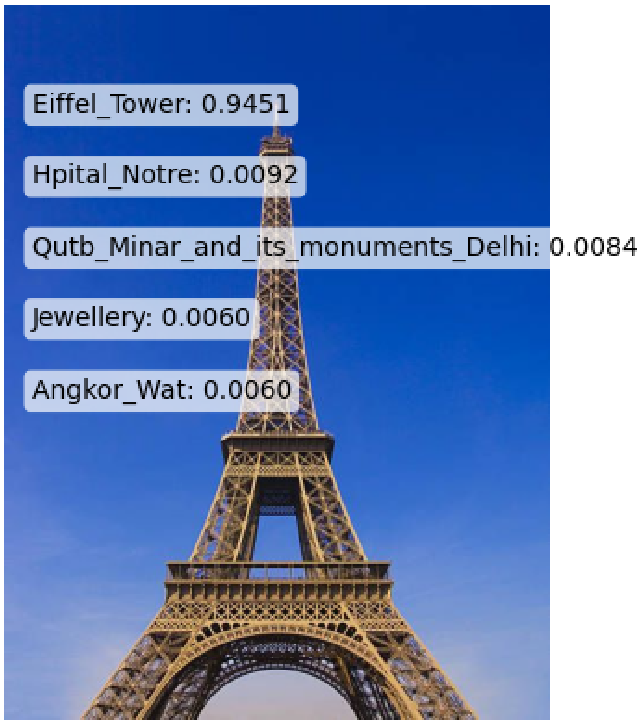

Fig. 6. Eiffle Tower

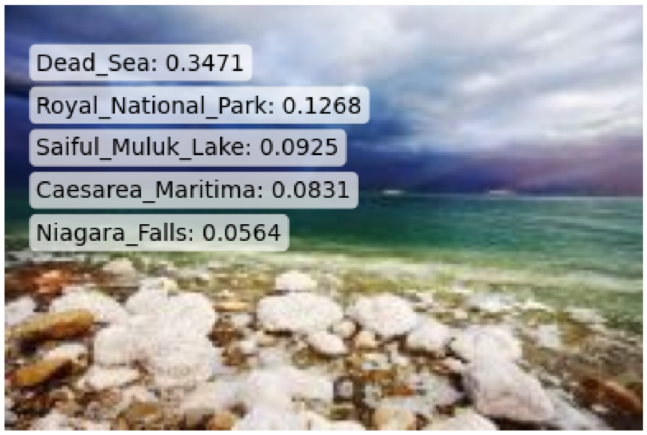
Dead_Sea: 0.3471
Royal_National_Park: 0.1268
Saiful_Muluk_Lake: 0.0925
Caesarea_Maritima: 0.0831
Niagara_Falls: 0.0564

Fig. 7. Dead Sea

A. Some more Results:

VIII. Conclusion

This project employed diverse methodologies and models, resulting in the successful implementation of this model capable of providing accurate classifications for user-input images. However, the model has certain limitations. It excels in classifying distinctive landmarks like the Eiffel Tower with high accuracy but faces challenges when categorizing less unique landmarks such as parks, castles, and seas, leading to lower accuracy rates. The constrained device capacity prevented the use of larger datasets, which could significantly enhance the model's performance. Despite these limitations, the current model overall achieves commendable performance.

References

[Weyand et al.(2020)] Weyand, T., Araujo, A., Cao, B., Sim, J. 2020, *Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

[Ramzi et al.(2023)] Ramzi, E., Audebert, N., Rambour, C., Araujo, A., Bitot, X., Thome, N. 2023, *Optimization of Rank Losses for Image Retrieval*, in submission to IEEE Transactions on Pattern Analysis and Machine Intelligence

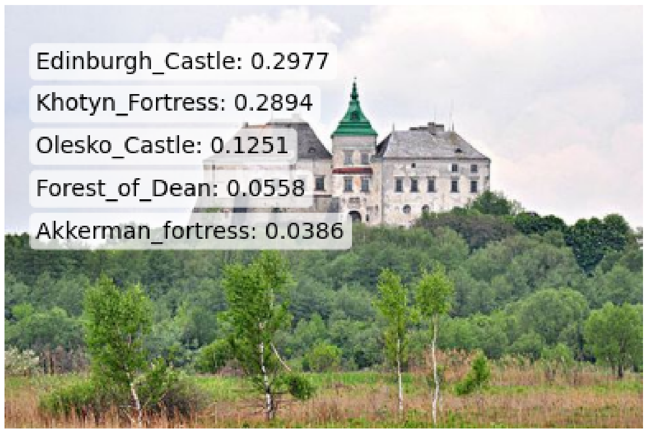

Fig. 8. Olesko Castle

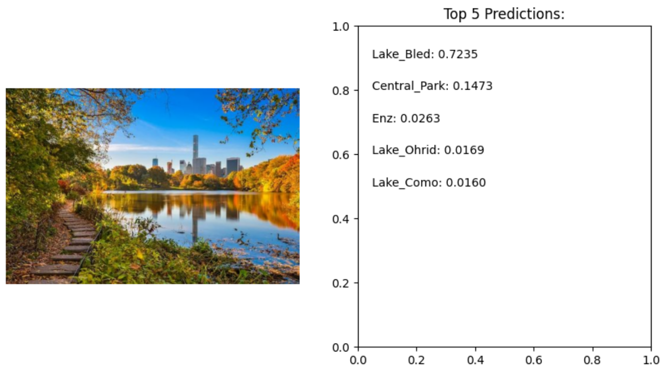

Fig. 9. Park Classification

[He et al.(2016)] He, K., Zhang, X., Ren, S., Sun, J. 2016, *Deep Residual Learning for Image Recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

[Xie et al.(2017)] Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K. 2017, *Aggregated Residual Transformations for Deep Neural Networks*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), https://arxiv.org/abs/1611.05431

[Sarkar (2018)] Sarkar, D. 2018, *Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook*, Blog post, Analytics Vidhya, https://www.analyticsvidhya.com/blog/2018/11/introduction-implementation-neural-network-architectures-tensorflow/

[ResNet Documentation] PyTorch. *ResNet Documentation*, https://pytorch.org/docs/stable/torchvision/models.html#id3

[ResNeXt Documentation] PyTorch. *ResNeXt Documentation*, https://pytorch.org/docs/stable/torchvision/models.html#id4

[ResNet Explained - YouTube] *ResNet Explained - YouTube*, https://www.youtube.com/watch?v=GWt6Fu05voI

[ResNeXt Overview - YouTube] *ResNeXt Overview - YouTube*, https://www.youtube.com/watch?v=CIfsB_EYsVI