LANDMARK CLASSIFICATION USING DEEP LEARNING

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

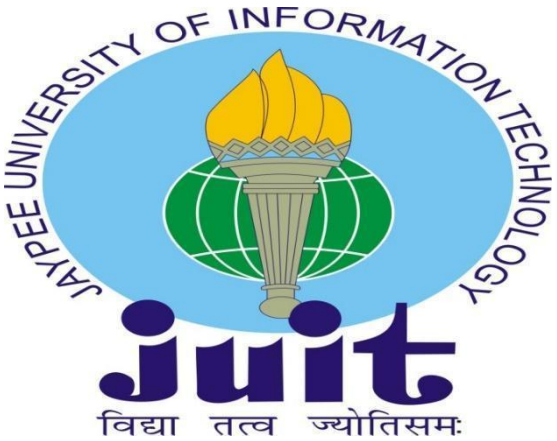
By

Nishant Attri 191512

Under the supervision of

Dr. Aman Sharma

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat,

Solan-173234, Himachal Pradesh

CERTIFICATE

I hereby declare that the work presented in this report entitled "LANDMARK CLASSIFICATION USING DEEP LEARNING" in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of Dr. Aman Sharma (Assistant Professor(SG)).
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Nishant Attri
191512

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Aman Sharma
Assistant Professor
Department of Computer Science & Engineering
Dated:

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, I express our heartiest thanks and gratefulness to almighty God for His divine blessing making it possible to complete the project work successfully.

I am really grateful and wish our profound indebtedness to the Supervisor Dr. Aman Sharma (Assistant Professor (SG), Department of Computer Science and Engineering). Deep knowledge & keen interest of our supervisor in the field of "Landmark classification" helped me to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to Dr. Aman Sharma, Assistant Professor (SG), Department of Computer Science and Engineering , for his kind help to finish this project. I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, I must acknowledge with due respect the constant support and patience of our parents.

Name : Nishant Attri

Roll No :191512

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S.No	Abbreviation	Full Form
1.	ANN	Artificial Neural Networks
2.	SVM	Support vector machines
3.	CNN	Convolutional Neural Networks
4.	KNN	K-Nearest Neighbour
5.	GPS	Global Positioning System
6.	PIL	Python Imaging Library
7.	GNU	GNU's Not Unix
8.	MATLAB	Matrix Laboratory
9.	BSD	Berkeley Software Distribution
10.	VGG	Visual Geometry Group

ABSTRACT

When we were looking at the diaries we kept as children, we would always try to identify all the places we had visited at least once during those early years. Despite this, we have no memory of their names. There is no doubt that Indians love to visit a variety of temples, but they often forget the names of the temples they have visited. In addition to that, it is also possible to experience a feeling of shyness when we are not even able to mention to our peers that we have also been there. We should never forget who built this monument. Due to this, Landmark Detection has become a powerful tool that helps us remember the names of these places in the future just so that we will be able to recognize them in the future. When it comes to Landmark Detection, it is the process of detecting pieces of artwork, buildings, and monuments that have been constructed by humans in an image by analysing the landmarks in that image. As a matter of fact, there is a visual symbol when it comes to buildings, which is known as a landmark, which is characterized by a distinctive visual form, which is not owned by other regions, and is very strong due to the fact that it has a characteristic that is unique. This classification of images can also be classified as landmark classifications as part of the Computer Vision field of study. The task of classifying images is one of the most difficult tasks for computers to handle, and it is often used to detect objects in images that are hidden within them. It is arguably very difficult for a computer to be able to classify images on its own. With the use of Deep Learning techniques, such as Convolutional Neural Networks (CNNs), we are able to simplify computer performance in classification

CHAPTER 1

INTRODUCTION

1.1 Introduction

There are several subfields in Machine Learning, such as Deep Learning, which uses artificial neural networks for the development of algorithms. The best results are observed by Artificial Neural Networks (ANNs) for classifying images, despite the fact that many other methods have proven to work as well. In comparison to these other methods, neural networks appear to have the best performance. Despite the fact that ANN was a relatively simple and straightforward machine learning method, it was not compatible with other machine learning techniques such as KNN and SVM, since different approaches and the complexity of the images made it lag behind other machine learning techniques. In recent years, Deep Learning has demonstrated exceptional performance. These performances have been attributed to strong computational factors, large datasets, and the use of techniques for training deeper networks. In short, CNN is an artificial neural network architecture that makes it much easier for image learning functions to implement as compared to other methods, since CNNs are primarily used to classify images. An image learning function can be made much more efficient to implement by using a CNN. One of the main problems with image classification is how accurate the results of the processing are and how much computer power is needed to process the images. The accuracy level is influenced not only by image diversity, but also by the size of the image, and by the type of hardware that processes the images. While there are a number of methods that have been discussed in relation to Deep Learning optimization, it is still difficult to determine which optimization offers the best accuracy level and the best time-saving results, as there have been numerous methods discussed. Generally speaking, the higher the level of computing power that is available to the CNN method, the better the performance that the CNN method will exhibit. This is particularly true for the implementation of the CNN method.

1.2 Problem Statement

The number of images uploaded to social media platforms is growing every day. As images are uploaded, these services can automatically identify their location based on the image metadata (such as GPS coordinates). However, if the image is not accompanied by GPS information, it will not be possible to locate the location within the image. As a part of our research, we plan to present an alternative method of identifying where a photograph was taken by identifying a recognizable landmark that can be identified. As a way of inferring the location of an image if there is no location metadata available, it can be done by detecting and classifying a discernible landmark within the image as a way to do so. Due to the sheer number of landmarks around the world and the enormous amount of images that are uploaded to photo sharing services on a daily basis, it would not be possible to classify these landmarks using human judgement, so we have decided to try and classify them through transfer learning.

1.3 Objectives

There are numerous land cover classes or themes that can be classified in a virtual picture and the purpose of the classification system is to categorize each pixel in the picture into one of these classes. As a result of this classified information, it is possible to also generate a thematic map describing the land cover of a picture based on the classified information. In most cases, multispectral information is used to categorize images and, indeed, the spectral pattern that is present in the information for each pixel is actually used as the numerical basis for categorizing them (Lillesand and Kiefer, 1994). The purpose of image classification is to discover and illustrate, as a completely distinct gray level (or hue), the functions occurring in a photo in terms of the item or kind of land cover these functions genuinely constitute at the bottom of a picture.

1.4 Methodology

There are three steps in the making of this project-

I. Download and Install Python Modules.
II. Create a CNN to Classify Landmarks (from Scratch).
III. Create a CNN to Classify Landmarks (using Transfer Learning).

Step I. Download and Install Python Modules.

We need to install the following python modules-

a) CV2
b) Matplotlib
c) Numpy
d) PIL
e) Torch
f) Torchvision

Let's understand all these modules and what they are.

a) CV2

A Python library called OpenCV is a tool for solving computer vision problems. OpenCV was developed initially in 1999 through Intel. Later in the year, it was supported by Willow Garage. In addition to C++, Python, and Java, OpenCV is compatible with a wide variety of programming languages. There are multiple systems that are supported by OpenCV Python, including Windows, Linux, and Mac OS. OpenCV Python is merely a wrapper elegance for a fully functional C++ library. It is possible to translate all of the OpenCV array systems to/from NumPy arrays using this method, which makes it easier to use OpenCV with libraries that use NumPy. A good example would be libraries alongside SciPy and Matplotlib.

b) Matplotlib

It is a plotting library similar to GNUplot with the benefit of being a Python module. One of the essential advantages of Matplotlib over GNUplot is that it is a Python module. Because Python is on the rise, Matplotlib's recognition is constantly increasing. In addition, Matplotlib's attraction is largely attributed to the fact that, when combined with Numpy and Scipy, it's taken into consideration to be a wonderful alternative to MATLAB. Whereas MATLAB is a highly-priced and closed supply, Matplotlib is a loose and open supply code. It is likewise item-orientated and may be utilized in an item-orientated way. Furthermore, it may be used with general-reason GUI toolkits like wxPython, Qt, and GTK+. There is

likewise a procedural "pylab", which is designed to intently resemble that of MATLAB. This could make it extraordinarily smooth for MATLAB customers to migrate to matplotlib. Matplotlib may be used to create first-class figures in plenty of hardcopy codecs and interactive environments throughout platforms. Another function of matplotlib is its steep studying curve, this means that customers normally make speedy development after having started. You can generate plots, histograms, energy spectra, bar charts, error charts, scatterplots, etc, with only some strains of code.

c) Numpy

There is no doubt that large-scale arrays and matrices are key components of Data Science strategies. Numpy is one of the most popular Python libraries that is used every single day. NumPy's array of diverse mathematical features makes it easier to extract useful records from those arrays and matrices with heavy numerical computation, which is made easier by using arrays or matrices as data storage containers, and heavy numerical computation is required to extract useful records from those arrays and matrices. In addition, a few other Python science libraries also rely on NumPy arrays as their primary inputs and outputs, so NumPy is not only the main Python science library but also one of the most essential. As an added bonus, it allows builders to use a single piece of code to perform simple and advanced statistical and mathematical functions on multidimensional arrays and matrices, even when the arrays or matrices are complex. Among the main capabilities of Numpy are its n-dimensional array statistics shape or ‘ndarray’. Numpy arrays are significantly faster than Python lists because they are homogeneous. As a result, all the elements of the array should be of the same type. It should be mentioned that Python lists are more flexible than NumPy arrays, in the sense that you can only keep the same type of statistics in each column. This makes them more versatile than NumPy arrays.

d) PIL

PIL is a Python imaging library that is very useful for processing photos. The library offers a number of functions for running on snapshots and for the use of Python as a programming language. Various Python photo-processing libraries, such as OpenCV, are used with this device as a photo-processing device.

e) Torch

The Python programming language and Torch library are both fundamental parts of the Python PyTorch machine learning framework. Torch is a fully open-source machine learning (ML) framework. Developed within the Lua scripting language, Torch is an open-source machine-learning library used for the development of deep neural networks. It's a favorite structure for deep gaining knowledge in research. Using this framework, the research prototyping and deployment process will be sped up.

Over two hundred special mathematical operations are available through PyTorch, an open-source framework for computing. The popularity of PyTorch continues to rise due to the simplicity of introducing synthetic neural community models. In particular, PyTorch is used by scientists for research as well as artificial intelligence. A changed BSD license is used to launch PyTorch.

f) TorchVision

Torchvison is a machine learning toolkit that offers a chain of modern algorithms, spanning from Gaussian Mixture Models to Hidden Markov Models, Support Vector Machines to Neural Networks. With Torchvision, you can control and manipulate pictures using common image processing methods with added functionality. Since Torchvision and Torch are completely integrated, the generated pictures may be used right away with Torch's machine learning algorithms. Both Torch and Torchvision are accessible to the public as GPL-licensed C++ versions under the terms of the Free-BSD licence.

Step II. Create a CNN to Classify Landmarks

Our next step will be to create a CNN that categorizes landmarks right at the beginning of the process. In this case, a 30% accuracy goal is our aim. Although 30% may at first appear low, it becomes a lot more reasonable when you understand how much hassle it is to perform this task. The most common way to photograph a landmark is to take an image of a structure or its surroundings that is quite unremarkable, such as in the example in the

following. When compared to the accuracy of 2% that could be achieved with random guessing, a 30% accuracy is much better.

Step III. Create a CNN to Classify Landmarks (using Transfer Learning).

Convolutional Neural Network (CNN) classification requires creating the network's architecture, training it on a dataset of landmark images, and assessing its performance on validation and testing sets. The CNN extracts features from raw photos automatically and learns to classify them into several landmark groups. To improve the mode's performance, techniques such as data augmentation, regularisation, and transfer learning can be applied.

Transfer Learning

The theory of transfer learning (TL) is one of many machine learning (ML) research areas that are concerned with storing the information that is obtained while resolving one problem in order to apply it in the same manner to a similar but unrelated issue in the future. Identifying trucks, for example, can be done using the same skills one would use to identify vehicles. The area of research in this area has some similarities with the extensive psychological literature on the transfer of learning, even though there are no formal connections between the two domains. In practice, a reinforcement learning agent's sampling efficiency may be substantially increased by reusing or transferring previous knowledge for learning new tasks in order to increase the sampling efficiency of the agent. It is therefore possible for training and test data to have different distributions, different labelling functions, different features, and even different classes depending on the data. There are two types of literature data: literature data following the same distribution, literature data that follows the same labelling function, and literature data whose features are the same are commonly referred to as data originating from the same source in transfer learning. Transfer learning is an approach to learning an algorithm for categorizing the target data that makes use of already available data from a variety of sources, e.g. data that has some similarities, but that is not necessarily the same as the target data, that is what is meant by transfer learning. In contrast to traditional supervised-learning algorithms that assume that the training and target data come from the same source, this approach is more approachable.

Three forms of transfer learning—inductive transfer learning, transductive transfer learning, and unsupervised transfer learning—have been identified in a review of the transfer learning literature by Pan and Yang [1]. The training and target data may have distinct labelling functions, feature distributions, prior distributions, or all three, depending on the kind of training and target data, which is covered in this paper's discussion of an inductive transfer learning technique. We presumptively have a small amount of labelled training samples from the target source, also known as the "same-distribution training data," and we aim to transfer knowledge from a much larger amount of labelled training samples, also known as the "different-distribution training data," that are available from sources other than the target data. Inductive transfer learning assumes that training and target sources' labelling functions are still fairly similar despite the fact that they come from different sources (training and target), so that sources with different distributions can add some extra information in regions of feature space where there is little to no training data with the same distribution.

The complete transfer learning process may be summed up as follows from a practical standpoint:

- Choose a pre-trained model : The initial stage in the transfer learning process is to choose a pre-trained model that is appropriate to the task at hand. This might be a model that was trained on an identical task or one that was pre-trained on a huge dataset like ImageNet. When choosing a pre-trained model, consider model architecture, accuracy, and the size of the pre-training dataset.

- Modify the pre-trained model: After choosing a pre-trained model, you must tweak it to fit your specific goal. Adding a new output layer and freezing the weights of the pre-trained layers is normal. The new output layer should contain the same number of classes as your target task, and the activation function should be chosen based on the task specifications. A sigmoid activation function, for example, may be utilised for a binary classification problem, whereas a softmax activation function may be better appropriate for a multiclass classification task.

- Train the model: After modifying the pre-trained model, you can train it on your chosen dataset. To avoid overfitting, assess the model's performance on a validation dataset during training and employ approaches such as early stopping. To increase

the model's performance, you should also try modifying hyperparameters such as the learning rate, batch size, and optimizer.

- Evaluate the performance: After training is complete, you should evaluate the performance of the model on a separate test dataset. This will give you an idea of how well the model generalizes to new data and how well it performs on your specific task.

- Iterate and refine: Depending on the model's performance, you may need to iterate and refine the process. To increase the model's accuracy, hyperparameters may be adjusted, a new pre-trained model used, or more data collected.

VGG 16 framework

The VGG16 model was discovered to have the greatest performance out of all the settings based on the performance of the ImageNet dataset. Let's examine this configuration's actual architecture in more detail.

A fixed 224 by 224 picture with three channels—R, G, and B, is considered the input to any network configuration. There is only one pre-processing that is carried out, and that is to normalize the RGB values of each pixel. This is done by subtracting the mean value from every pixel. Following activation of the ReLU, the image is sent to the first layer of two convolution layers that each have a very small receptive area of 3 x 3. In each layer, 64 filters are employed. This setup maintains the spatial resolution of the input picture, and the dimensions of the output activation map remain the same as those of the input picture, since the padding of one pixel is kept, while the convolution stride remains one pixel. As a result, the activations are then subjected to spatial max pooling with a stride of two pixels over a frame that measures 2x2, so that the activations are cut in half. The activations at the end of the first stack are 112 by 112 x 64 in size.
It is then necessary to pass the activations through a second stack that is identical to the first stack, except that the second stack has 128 filters instead of 64 in the first stack. In this case, the size is 56 x 56 x 128 after the second layer, and then a third layer is added to it, consisting of a max pool layer as well as three convolutional layers. Due to the 256 filters that were applied to this case, the output size of the stack in this case was 28 x 28 x 256, which is the output size of the stack. It is then followed by two more stacks of three

convolutional layers, each with 512 filters. The ultimate outputs of both of these stacks will be 7 7 x 512. The three fully connected layers that come after the convolutional layer stacks are separated by a flattening layer. One thousand neurons make up the output layer, the last fully connected layer, which represents the 1,000 potential classes in the ImageNet dataset. 4,096 neurons are found in each of the first two levels. After the output layer comes the Softmax activation layer, which is used for category classification.

As part of this project, we are going to use the VGG-16 model as a pretrained model, which is the result of a key turning point in humanity's effort to enable computers to "see" the world, in the form of a computer. There has been significant progress in the field of computer vision (CV) since several decades ago. The invention VGG16 was one of the key innovations that paved the way for further advancements in this field. Developed by Andrew Zisserman and Karen Simonyan at the University of Oxford, this model is based on a convolutional neural network (CNN). In 2013, the concept of the model was released, but in 2014 as part of the ILSVRC ImageNet Challenge, the actual model was presented. As part of ImageNet Large Scale Visual Recognition Challenge (ILSVRC), methods were evaluated on their ability to categorize (and recognize) large amounts of pictures. However, although they performed well on the task, they ultimately failed. Since our dataset is quite similar and we have a small number of classes, in order to output 50 classes, I only replaced the last fully connected layer of the model with one of our own problems.

1.5 Organization

The report's structure can be summarized as follows:

Part 1: Gives a general overview of the project, the venture points and the ambitions of the project.

Part 2: This section provides an overview of the topic's current research**.** A full explanation of all research, investigations, theories, and social gatherings involved in the project is given.

Part 3: Evaluates the project's architecture and strategy to determine the best result prediction.

Part 4: Results and screenshots are discussed.

Part 5: Finish the project and submit a proposal for further work.

CHAPTER 2
LITERATURE SURVEY

Filip Radenovic.[1] "Fine-tuning CNN Image Retrieval with No Human Annotation" published in 2018 presents a technique for fine-tuning convolutional neural networks (CNNs) for image retrieval tasks without the requirement for human annotations. A significant amount of weakly labelled data and a cutting-edge training method are used to achieve this. The authors begin by pointing out that CNNs can be employed as efficient feature extractors for image retrieval tasks when they have been trained beforehand on big image classification datasets. However, a large number of annotated images are typically needed to fine-tune these networks for particular retrieval tasks, which can be time-consuming and expensive to acquire. The authors suggest a strategy to get over this restriction by fine-tuning the CNNs for picture retrieval using a lot of weakly labelled images. The photos that are retrieved from the web using text-based queries make up the poorly labelled data used in this technique. Even though these images lack explicit annotations, it can be assumed that they fall under the same semantic category as the search term.

Tobias Weyand.[2] "Google Landmarks Dataset v2 A Large-Scale Benchmark for Instance-Level Recognition and Retrieval", published in 2020. In the research, multiple cutting-edge approaches for instance-level recognition and retrieval are evaluated using a huge dataset of landmark photographs. Over 5 million images of more than 200,000 unique landmarks from around the world are included in the dataset. Along with other metadata like the GPS coordinates and the date the image was taken, each image has a distinctive identifier for the landmark. The photos were gathered from a variety of websites, including photo-sharing websites, social networking platforms, and image search engines. The authors employ a number of benchmark tasks, such as landmark retrieval, recognition, and verification, to assess instance-level recognition and retrieval techniques. The objective of the landmark retrieval challenge is to find photographs of a specific landmark from a query image. Classifying an image as belonging to a specific landmark is the aim of the landmark recognition challenge. The aim of the landmark verification task is to establish whether or not two photographs share the same landmark.

For these tasks, the authors assess a number of cutting-edge techniques, including deep neural networks and conventional computer vision algorithms. They demonstrate that deep neural networks perform noticeably better than conventional algorithms, and that this performance rises with the amount of the training dataset. In order to determine the most popular landmarks and the distribution of landmark photos across various countries and continents, the authors additionally analyse the dataset to find trends and patterns in the images. They demonstrate the diversity of the landmarks in the dataset and the variety of compositional and stylistic approaches used in the images. The study concludes by presenting a sizable dataset of landmark photos and evaluating a number of cutting-edge techniques for instance-level recognition and retrieval using this dataset. The work shows the possibility for additional research in this field and demonstrates the value of deep learning techniques for these tasks.

Hyeonwoo Noh.[3] "Large-Scale Image Retrieval with Attentive Deep Local Features" published in 2018. The solution for large-scale image retrieval presented in the research uses deep learning techniques and obtains cutting-edge results on a number of benchmark datasets. The approach put out in the research uses a deep neural network to extract local properties from photographs, which are then combined and matched to find related images from a sizable dataset. A convolutional neural network (CNN) that has been pre-trained on a large image classification dataset and fine-tuned on a smaller dataset of landmark images is used to extract the local features. The authors suggest using an attention mechanism, which enables the network to concentrate on the most crucial local features for the retrieval job, to enhance the method's performance. Based on a soft attention mechanism, the attention mechanism computes a weighted sum of the local characteristics according to their applicability to the query image. The Oxford5k, Paris6k, and Holidays datasets, which are frequently used for evaluating image retrieval algorithms, are among the benchmark datasets that the authors utilise to assess their methodology. They demonstrate that their approach outperforms both conventional methods based on hand-crafted features and other deep learning-based approaches to reach state-of-the-art performance on these datasets. The authors also do ablation studies to assess the effects of various elements of their methodology, such as the CNN's pre-training and the attention mechanism. They demonstrate the potency of their suggested strategy and explain how these elements contribute considerably to the method's efficacy. The research study concludes by presenting a solution for large-scale image retrieval that makes use of deep learning

methods and an attention mechanism. The technique highlights the value of pre-training and attention processes in deep learning-based picture retrieval while achieving state-of-the-art performance on a number of benchmark datasets. The technique makes a significant contribution to the field of picture retrieval and may find use in a variety of settings, including image search engines.

Jia Shijie.[4] "Research on Data Augmentation for Image Classification Based on Convolution Neural Networks" published in 2019. The study investigates how convolutional neural networks (CNNs) can perform better when data augmentation techniques are used. The significance of data augmentation for training CNNs is covered in the paper's opening section. To provide more training examples and broaden the diversity of the training data, data augmentation entails performing modifications to the training images, such as flipping, rotating, and scaling. As a result, CNN may be able to learn more robust and universal features, which will enhance its performance on brand-new, untried photos. The authors then suggest a number of fresh data augmentation methods, including colour jittering, random erasing, and random cropping. Random cropping, which includes picking an area of the image at random and resizing it to its original size, can aid the network in picking up more invariant information. Overfitting can be avoided by using random erasing, which entails randomly deleting a piece of the image with a random value. By randomly changing the image's brightness, contrast, and saturation, colour jittering can broaden the training set of data.

The authors do trials on a number of benchmark datasets, including CIFAR-10, CIFAR-100, and ImageNet, to see how well their suggested data augmentation strategies work. To assess how each data augmentation strategy affects the performance of the CNN, the authors also undertake ablation trials. They explain the relevance of integrating different data augmentation approaches for the best performance and show that each strategy considerably improves performance.

Ahmet Iscen.[5] "Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking" published in 2018. The research introduces a brand-new benchmark dataset for massive picture retrieval and tests a number of cutting-edge techniques on it.

The constraints of current benchmark datasets for image retrieval, such as the Oxford5k and Paris6k datasets, which are very small and have a small number of queries and ground-truth images, are covered in the first section of the work. The Oxford and Paris

(OxP) collection, which the authors suggest using, has over a million photos, thousands of searches, and ground-truth images. The OxP dataset is intended to present a greater challenge and to be more typical of actual picture retrieval situations. The authors then assess a number of cutting-edge methods for picture retrieval on the OxP dataset, including hybrid methods, hand-crafted feature methods, and deep learning feature methods. On the OxP dataset, they demonstrate that the deep learning-based methods perform better than the hand-crafted methods, with the top method achieving a mean average precision (mAP) of over 0.7. The size of the database and the number of queries are two other variables that the authors test to see how they affect how well the picture retrieval methods perform. It is important to develop scalable methods for large-scale image retrieval since they demonstrate that the performance of the approaches declines as the size of the database and the number of queries rise. The research underlines the promise of deep learning-based methods for this task and emphasises the value of creating scalable systems for large-scale image retrieval. Researchers working on image retrieval can benefit greatly from the OxP dataset, which has the potential to develop this area of study even further.

Shuhei Yokoo.[6] "Two-stage Discriminative Re-ranking for Large-scale Landmark Retrieval" published in 2019. The performance of massive landmark retrieval is improved by the unique two-stage discriminative re-ranking method proposed in the research.

The constraints of current methods for landmark retrieval are first discussed. These methods frequently have low retrieval accuracy and poor scalability as a result of the vast number of candidate photos and the substantial intra-class variation of landmark images. To overcome these constraints, the authors suggest a two-stage discriminative re-ranking strategy.

The query image and the candidate images are initially processed by the authors using a convolutional neural network (CNN) to extract global features. The candidate images are then ranked according to how similar they are to the query image using a pairwise similarity matrix. The authors employ a discriminative re-ranking strategy in the second stage to refine the initial rating achieved in the first stage. They employ a graph-based strategy to create a similarity graph between the candidate photos and the query image. They then use a random walk technique to propagate the similarity scores between the photos in the graph, which aids in the refinement of the candidate image ranking.

The authors undertake experiments on two benchmark datasets, Google Landmarks Dataset and Tokyo 24/7, to assess the efficacy of their suggested technique. They

demonstrate that their suggested method outperforms multiple state-of-the-art landmark retrieval algorithms, yielding significant gains in retrieval accuracy. The authors also conduct ablation studies to assess the influence of different components of their technique on landmark retrieval performance. They demonstrate that both global feature extraction with CNNs and discriminative re-ranking with the graph-based technique greatly enhance performance. The research shows how their suggested strategy works on two benchmark datasets and underlines the promise of graph-based methods for discriminative re-ranking. The suggested method is applicable in a variety of disciplines, including computer vision, image recognition, and deep learning.

Syed Kazim Raza.[7] "Visual Landmarks Recognition of Urban Structures using Convolutional Neural Network" published in 2019. Using convolutional neural networks (CNNs), the research provides a method for visual landmark recognition of metropolitan structures. The study opens by exploring the significance of visual landmarks in a variety of applications, including navigation, tourism, and urban planning. The authors present a CNN-based method for visual landmark recognition that is capable of dealing with the complex and diverse visual appearance of urban structures. The proposed approach is divided into three steps. The authors begin by extracting features from the input image using a pre-trained CNN. They utilise a clustering method in the second stage to organise the retrieved data into visual words. They employ a bag-of-visual-words (BoVW) model in the third step to represent the input image as a histogram of visual words. The authors conduct experiments on a dataset of urban structures to assess the effectiveness of their suggested strategy. They compare their method to other cutting-edge methods for visual landmark detection, such as classic feature-based methods and deep learning-based methods. In terms of accuracy and computational efficiency, they demonstrate that their proposed solution beats the other methods. The authors also undertake ablation studies to assess the influence of different components of their technique on visual landmark identification performance. They demonstrate that both feature extraction using CNNs and the BoVW model greatly enhance performance. The research shows the effectiveness of their suggested method on a dataset of urban structures and emphasises the potential of CNNs for dealing with the complex and diverse visual appearance of urban structures. The proposed method has a wide range of applications, including navigation, tourism, and urban planning.

Andre´ Araujo.[8] "Detect-to-Retrieve: Efficient Regional Aggregation for Image Search" published in 2020. By performing regional aggregation on object detection outputs, the research provides a new method for efficient image search. The study begins by exploring the significance of picture search in a variety of applications, including e-commerce, multimedia retrieval, and augmented reality. To improve accuracy and efficiency, the authors suggest a new strategy to picture search that combines object detection and geographic aggregation. The proposed approach is divided into two steps. Object detection is conducted on the query image in the first stage to identify areas of interest (ROIs) that include objects. The ROIs are aggregated into a global representation in the second stage, which is used to retrieve similar images from a database. The authors test their suggested method on a variety of classic image retrieval datasets, including the Oxford Building and Paris Street View datasets. They compare their solution to numerous cutting-edge image search algorithms, including standard bag-of-words methods and deep learning-based methods. In terms of accuracy and computational efficiency, they demonstrate that their proposed solution beats the other methods. The authors also conduct ablation studies to assess the impact of various components of their technique on picture retrieval performance. They demonstrate that both item detection and regional aggregation considerably improve performance. The research shows the effectiveness of their proposed method on many typical image retrieval datasets and underlines the potential of object recognition and geographic aggregation for enhancing image search accuracy and efficiency. The proposed method has potential applications in e-commerce, multimedia retrieval, and augmented reality.

Agnieszka Mikołajczyk.[9] "Research on Data augmentation for improving deep learning in image classification problem" published in 2018. The research presents a new data augmentation strategy for deep learning models to improve their performance on picture classification issues. The paper starts by emphasising the significance of data augmentation in deep learning for image classification tasks. The authors suggest a new data augmentation strategy that involves randomly creating affine transformations on the training images, such as rotation, scaling, and translation. The authors test the effectiveness of their suggested data augmentation strategy on CIFAR-10 and CIFAR-100 image classification datasets. They compare their method to others for data augmentation, such as flipping, cropping, and colour jittering. In terms of accuracy, they demonstrate that their proposed data augmentation methodology surpasses the other methods. The authors also

run experiments to see how different parameters of their data augmentation technique affect image classification results. They demonstrate that the type and amplitude of affine transformations, as well as the likelihood of applying them, can have a considerable impact on the deep learning model's performance. The research shows the efficiency of their proposed strategy on multiple standard image classification datasets and underlines the potential of data augmentation for increasing deep learning model performance in image classification tasks. The proposed data augmentation method is applicable to a variety of deep learning models and picture classification problems.

CHAPTER 3
SYSTEM DESIGN & DEVELOPMENT

3.1 Dataset

The dataset used is a part of Google Landmarks Dataset v2 [3].

- Approximately 6000 images make up the dataset, which is divided up into 50 categories.
- The dataset was initially split in two, with an 80:20 ratio between the training and testing datasets.
- In the training dataset, there are 100 images/categories in total
- A total of 25 images and 25 categories are included in the testing dataset.

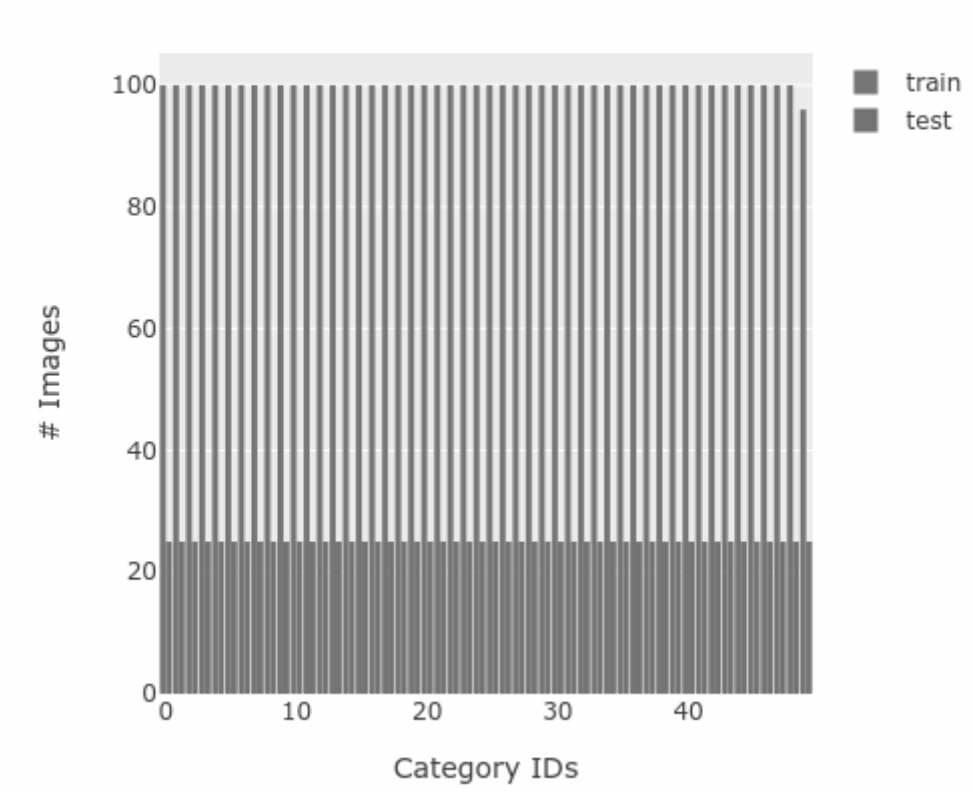


Fig 3.1 Images per Category

- Most images are < 200 kB.

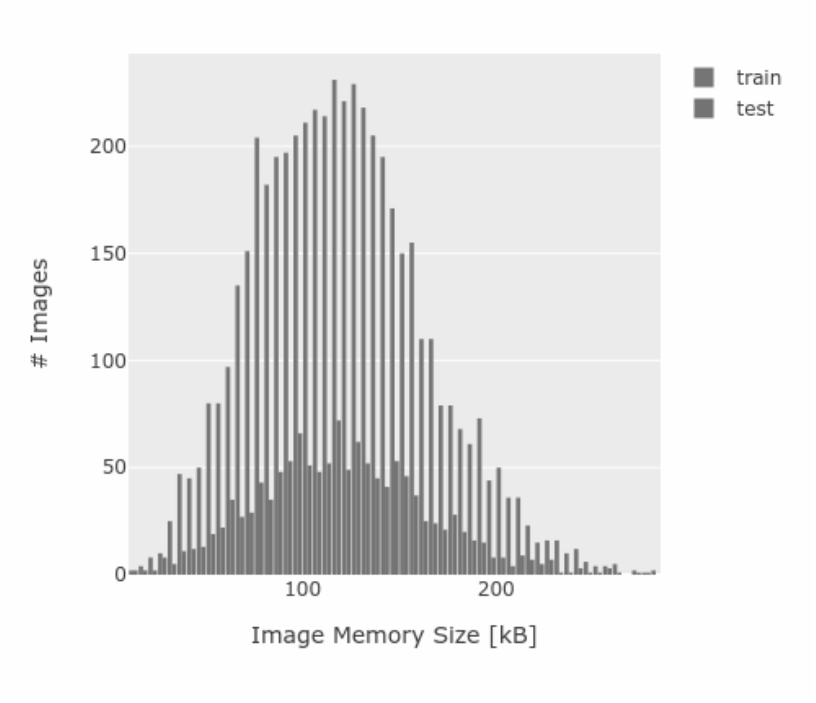


Fig 3.2 Images Memory Size

- The largest images are 800 x 800 pixels.
- Most images have one side which is 800 pixels.

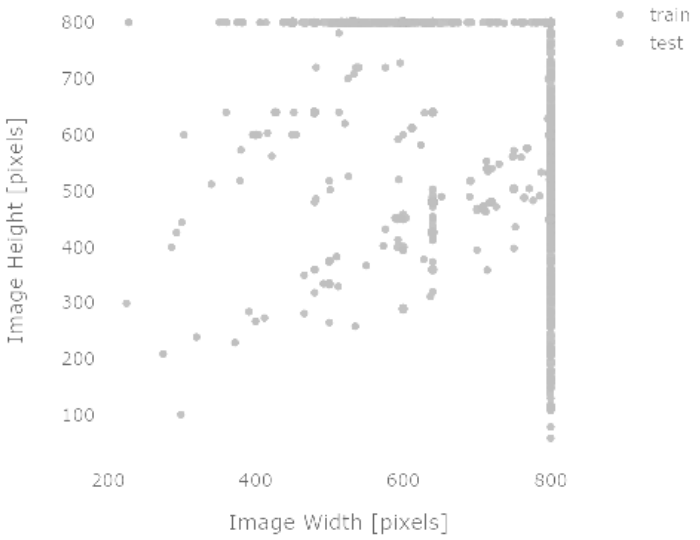


Fig 3.3 Images Width vs Height

- The training data set is then split into a ratio of 80:20 to form the training data set and validation data set.
- The final data set appears to be as follows.

Table 3.1 Data Set Distribution

Datasets	Total images	Total (%)
Train	3996	~64
Validate	1000	~16
Test	1250	~20

3.2. Model Development

3.2.1 Data Augmentation

In order to train the model, the images were randomly scaled and cropped to 224x224 pixels. However, in order to test the model, they were enlarged to 256x256 pixels, then cropped to 224x224 pixels at the center for testing. Due to the fact that 224x224 pixels looks very typical and reasonable when it comes to an image, and because it is inspired by the imagenet training image size of VGG-16, 224x224 pixels was chosen as the input image size. Once the image has been turned into a tensor, normalized to the imagenet normalization values [5].

Only a tiny portion of the dataset's training subset was subjected to random rotations of 10 degrees, random horizontal flips of the pictures, and random and varying colour jitters.

3.2.2 Create a CNN to Classify Landmarks

Creating a CNN that classifies landmarks from scratch is the first step in this process. We set a target accuracy of 30%. In spite of how low 30% may seem at first glance when you consider how difficult this problem is, it seems reasonable. Like in the following picture, an image taken at a landmark sometimes captures a mundane image of an animal or plant.

Fig 3.4 Man in Haleakalā National Park

Having a 30% accuracy is substantially better than having a 2% accuracy from random guessing. The neural network was developed "from the ground up." The architecture employed by this model, which is a condensed form of VGG16, is as follows.

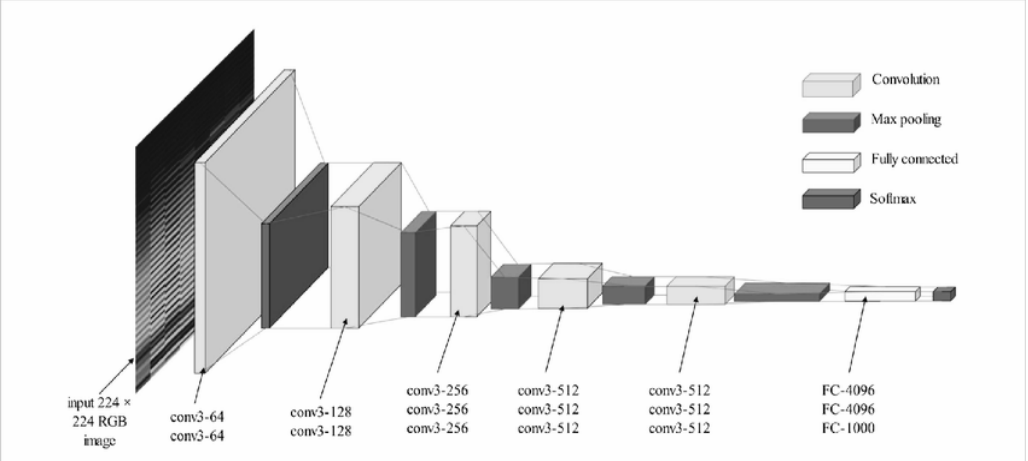


Fig 3.5 Structure of CNN Used

Table 3.2 Model Used (Similar to VGG16)

Component	Layers	Size
Input	RGB Image	224*224*3
Feature extractor	2 Conv layer + 1 max pool	
	2 Conv layer + 1 max pool	
	2 Conv layer + 1 max pool	
Classifier	Flatten layer + dropout	
	1 fully connected + ReLU	
	1 fully connected + ReLU	
Output	Log Softmax	50

3.2.3 Create a CNN to Classify Landmarks (using Transfer Learning)

Using the same classifier that was used in the above CNN, the second neural network uses a pre-trained version of the VGG-16 neural network. Classifier weights are the only thing that needs to be retrained since-

i. In comparison, VGG16 was trained with over 1.2 million images while the new dataset contains only 6000 images.
ii. VGG16 was trained on the same dataset as the new dataset that was used in the training of this model.

The final fully connected layer of the model was also replaced with one of our own to produce 50 classes. Additionally, the parameters were frozen for all the feature layers of the model, leaving the parameters of the classifier part for training.

VGG - 16 framework

Table 3.3 VGG-16 Architecture Used

	A	B	C
1	Parameter	Method	Value / Function
2	Error function	Cross Entropy lo	nn.NLLLoss()
3	Metrics	Accuracy	> 60 %
4	Batching	Mini-batches	64 data points
5	Optimizer	Stochastic Gradi	nn.optim.SGD()
6	Epochs	-	20
7	Learning rate	-	0.01

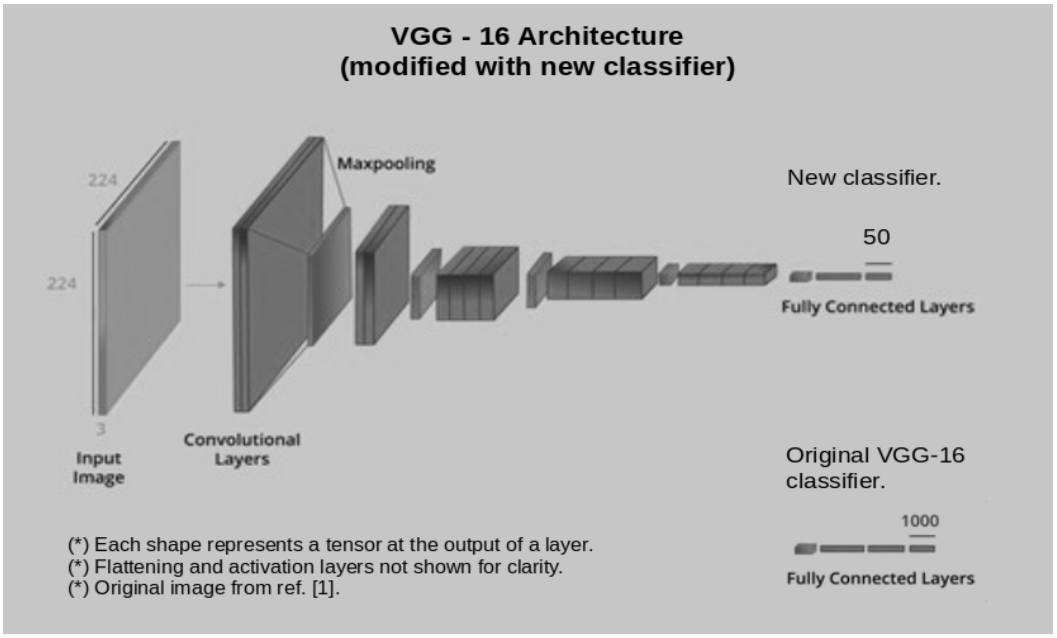


Fig 3.6 VGG-16 Model

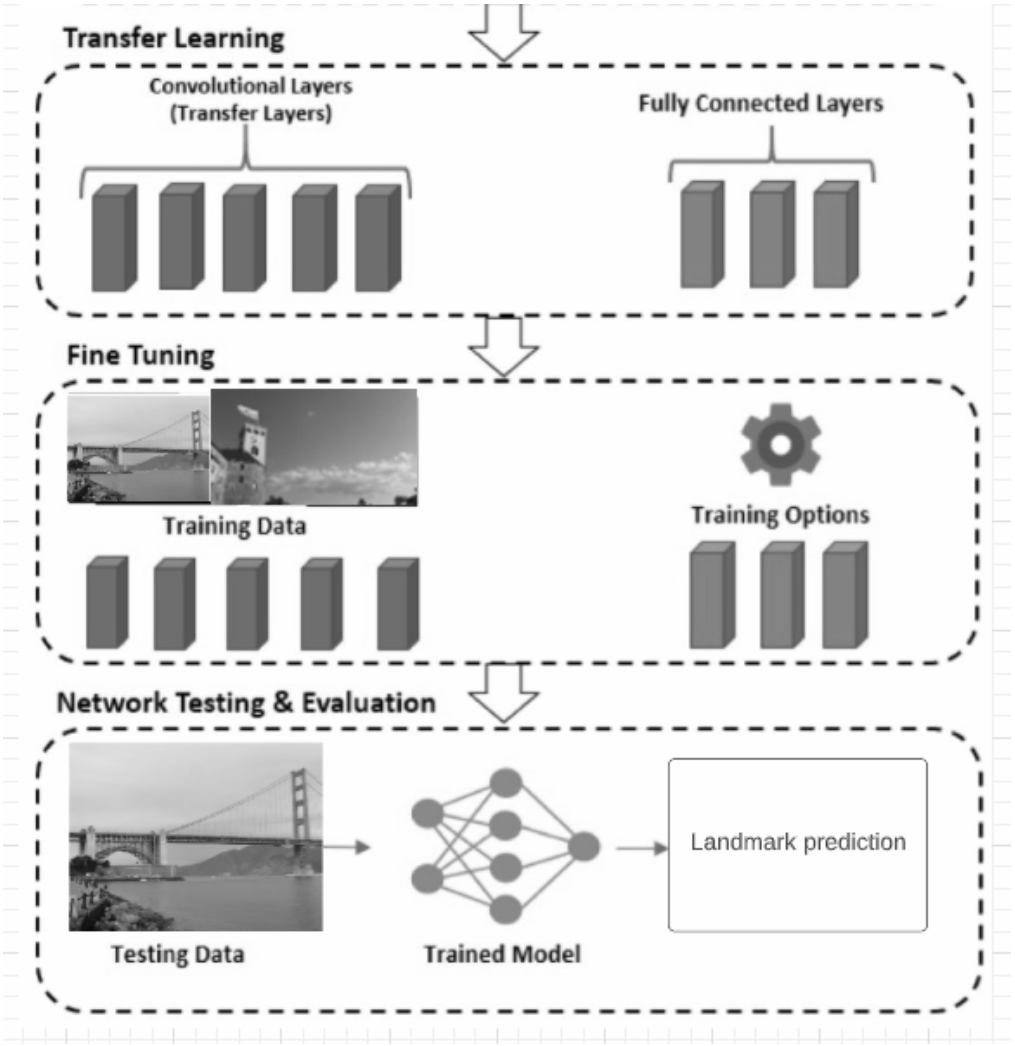


Fig 3.7 Transfer Learning Model

```
[ ]  from torchvision import models

     model_transfer = models.vgg16(pretrained=True)

     # Freeze training for all "features" layers
     for parameter in model_transfer.features.parameters():
         parameter.requires_grad = False


     n_input = model_transfer.classifier[6].in_features
     model_transfer.classifier[6] = nn.Linear(n_input, n_classes) # n_classes = 50


     if use_cuda:
         model_transfer = model_transfer.cuda()
```

Fig 3.8 Implementation of the Model

Loss function

Based on the assumptions we make, our error function measures and calculates the deviation between our predictions and the predictions that are accurate based on the assumptions we make.

This function calculates the adverse consequences associated with an incorrect prediction caused by a particular error size or direction using a loss function influenced by those consequences. The loss function we use is a cross entropy loss function, which is affected by the consequences of specific error sizes or directions.

Cross Entropy loss function-

Random variable X's entropy measures the level of uncertainty associated with its possible outcomes. Entropy is defined as the following if p(x) is the probability distribution and X is a random variable.

$$H(X) = \begin{cases} -\int_x p(x) \log p(x), & \text{if } X \text{ is continous} \\ -\sum_x p(x) \log p(x), & \text{if } X \text{ is discrete} \end{cases}$$

Fig 3.9 Definition of Entropy

The projected probability and the real class's intended outcome, which may either be 0 or 1, are used to calculate a score or loss based on how much the probability deviates from the actual expected value. A penalised difference close to 1 obtains the highest score possible, but a difference close to 0 receives a low score.

As a result of cross-entropy loss, the weights of the model are adjusted during training. A better model results in a lower loss. Models with a zero cross-entropy loss are flawless.

Cross-entropy is defined as

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \text{ for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

Fig 3.10 Mathematical definition of Cross-Entropy

Gradient descent optimizer

This approach locates a local minimum of a differential equation using first-order iterative optimization. Theoretically, because this is thought to be the sharpest descending direction, repeated steps should be made in the opposite direction of the gradient at the present level (or a rough gradient) [7]. On the other hand, if you go in the gradient's direction, a local maximum will happen. In this instance, the method is known as gradient ascent.

SGD (Stochastic Gradient Descent)

In It has been demonstrated that SGD algorithms are efficient at optimizing large-scale deep learning models [8]. Stochastic describes a mechanism or method that is based on some random possibility; thus, for each iteration, only a few samples are selected at random [9]. In SGD, the network structure is changed after each training stage in order to find the global minimum. Instead of determining the gradient of the full dataset, the error is minimised by estimating the gradient of a randomly chosen batch. Rather than random selection, random shuffling is done stepwise through batches of data. Based on SGD, significant variations in the objective function occur frequently [11], Figure 3.11 provides an example.

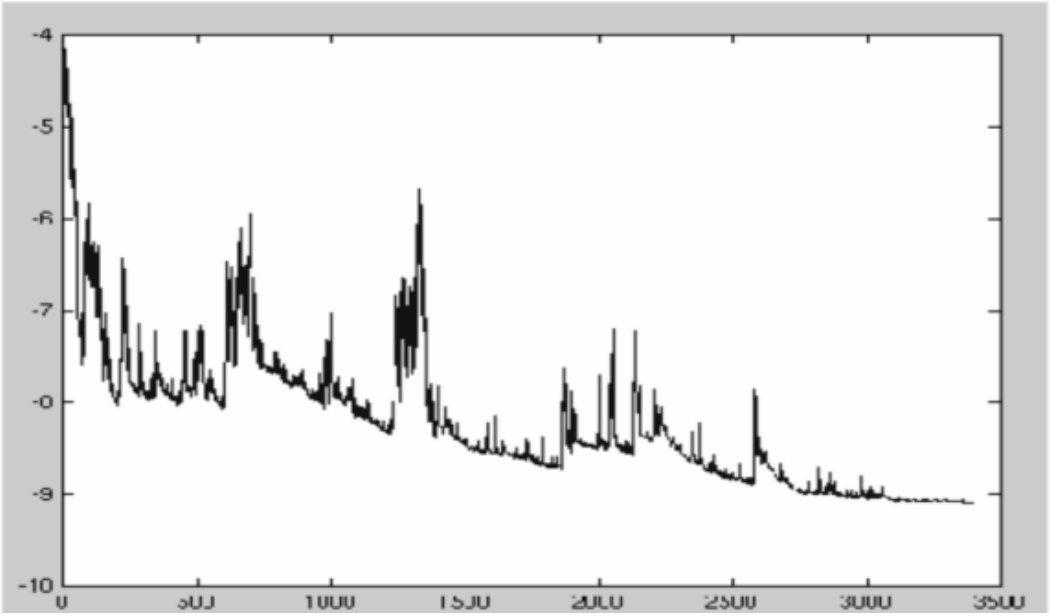


Fig 3.11 Stochastic Gradient Descent

Implementing cross entropy and SGD.

```
# loss function
criterion_transfer = nn.CrossEntropyLoss()


def get_optimizer_transfer(model):
  #Optimizer

    optimizer = optim.SGD(model.classifier.parameters(), lr=0.01)
    return optimizer
```

Fig 3.12 Implementation of Cross Entropy and SGD

CHAPTER 4
EXPERIMENT AND RESULT ANALYSIS

4.1 Results and Outputs

4.1.1 CNN made from scratch

After applying all the data augmentation techniques, visualised a batch of 8 images.

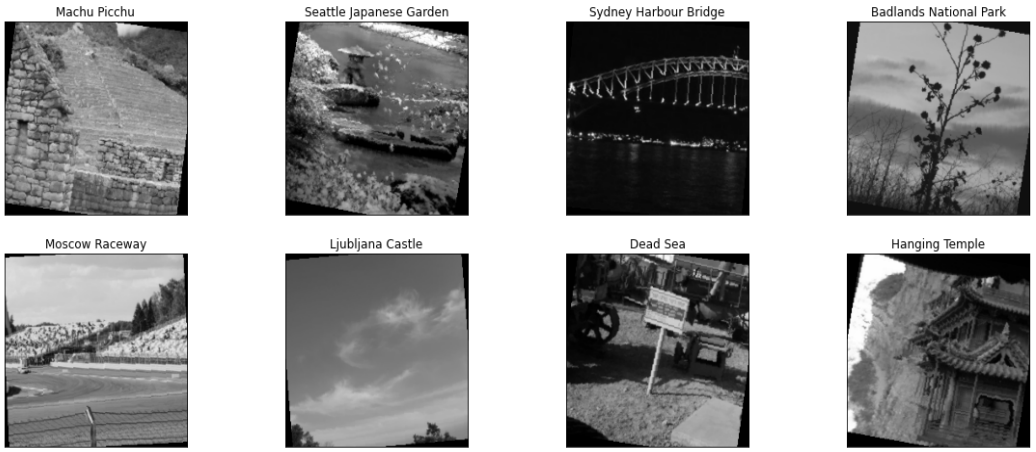


Fig 4.1 Visualised Batch of 8 Images

```
Epoch: 1        Training Loss: 3.911506         Validation Loss: 3.450699
Validation loss decreased (inf --> 3.450699).  Saving model ...
Epoch: 2        Training Loss: 3.642568         Validation Loss: 3.268413
Validation loss decreased (3.450699 --> 3.268413).  Saving model ...
Epoch: 3        Training Loss: 3.501564         Validation Loss: 3.205350
Validation loss decreased (3.268413 --> 3.205350).  Saving model ...
Epoch: 4        Training Loss: 3.435972         Validation Loss: 3.088995
Validation loss decreased (3.205350 --> 3.088995).  Saving model ...
Epoch: 5        Training Loss: 3.364526         Validation Loss: 3.008727
Validation loss decreased (3.088995 --> 3.008727).  Saving model ...
Epoch: 6        Training Loss: 3.298379         Validation Loss: 3.003557
Validation loss decreased (3.008727 --> 3.003557).  Saving model ...
Epoch: 7        Training Loss: 3.252809         Validation Loss: 2.940330
Validation loss decreased (3.003557 --> 2.940330).  Saving model ...
Epoch: 8        Training Loss: 3.193659         Validation Loss: 2.856374
Validation loss decreased (2.940330 --> 2.856374).  Saving model ...
Epoch: 9        Training Loss: 3.165934         Validation Loss: 2.965969
Epoch: 10       Training Loss: 3.135950         Validation Loss: 2.887387
Epoch: 11       Training Loss: 3.075298         Validation Loss: 2.806339
Validation loss decreased (2.856374 --> 2.806339).  Saving model ...
Epoch: 12       Training Loss: 3.042089         Validation Loss: 2.800764
Validation loss decreased (2.806339 --> 2.800764).  Saving model ...
Epoch: 13       Training Loss: 3.042000         Validation Loss: 2.843068
Epoch: 14       Training Loss: 2.968982         Validation Loss: 2.781700
Validation loss decreased (2.800764 --> 2.781700).  Saving model ...
Epoch: 15       Training Loss: 2.958057         Validation Loss: 2.703547
Validation loss decreased (2.781700 --> 2.703547).  Saving model ...
Epoch: 16       Training Loss: 2.966427         Validation Loss: 2.640817
Validation loss decreased (2.703547 --> 2.640817).  Saving model ...
Epoch: 17       Training Loss: 2.893807         Validation Loss: 2.664074
Epoch: 18       Training Loss: 2.930810         Validation Loss: 2.665594
Epoch: 19       Training Loss: 2.889585         Validation Loss: 2.605282
Validation loss decreased (2.640817 --> 2.605282).  Saving model ...
Epoch: 20       Training Loss: 2.868078         Validation Loss: 2.698078
```

Fig 4.2 Different Weight Initialization

Training and validating the model

```
Epoch: 1        Training Loss: 3.715440         Validation Loss: 3.405643
Validation loss decreased (inf --> 3.405643).  Saving model ...
Epoch: 2        Training Loss: 3.473488         Validation Loss: 3.236045
Validation loss decreased (3.405643 --> 3.236045).  Saving model ...
Epoch: 3        Training Loss: 3.388526         Validation Loss: 3.148500
Validation loss decreased (3.236045 --> 3.148500).  Saving model ...
Epoch: 4        Training Loss: 3.283857         Validation Loss: 3.084732
Validation loss decreased (3.148500 --> 3.084732).  Saving model ...
Epoch: 5        Training Loss: 3.228807         Validation Loss: 3.017459
Validation loss decreased (3.084732 --> 3.017459).  Saving model ...
Epoch: 6        Training Loss: 3.180959         Validation Loss: 2.945568
Validation loss decreased (3.017459 --> 2.945568).  Saving model ...
Epoch: 7        Training Loss: 3.133376         Validation Loss: 2.918783
Validation loss decreased (2.945568 --> 2.918783).  Saving model ...
Epoch: 8        Training Loss: 3.093122         Validation Loss: 2.894541
Validation loss decreased (2.918783 --> 2.894541).  Saving model ...
Epoch: 9        Training Loss: 3.038492         Validation Loss: 2.868679
Validation loss decreased (2.894541 --> 2.868679).  Saving model ...
Epoch: 10       Training Loss: 3.009516         Validation Loss: 2.768383
Validation loss decreased (2.868679 --> 2.768383).  Saving model ...
Epoch: 11       Training Loss: 2.970611         Validation Loss: 2.854443
Epoch: 12       Training Loss: 2.923759         Validation Loss: 2.755418
Validation loss decreased (2.768383 --> 2.755418).  Saving model ...
Epoch: 13       Training Loss: 2.912466         Validation Loss: 2.692363
Validation loss decreased (2.755418 --> 2.692363).  Saving model ...
Epoch: 14       Training Loss: 2.897586         Validation Loss: 2.775196
Epoch: 15       Training Loss: 2.875578         Validation Loss: 2.687246
Validation loss decreased (2.692363 --> 2.687246).  Saving model ...
Epoch: 16       Training Loss: 2.831820         Validation Loss: 2.605873
Validation loss decreased (2.687246 --> 2.605873).  Saving model ...
Epoch: 17       Training Loss: 2.809957         Validation Loss: 2.630216
Epoch: 18       Training Loss: 2.776156         Validation Loss: 2.569144
Validation loss decreased (2.605873 --> 2.569144).  Saving model ...
Epoch: 19       Training Loss: 2.765365         Validation Loss: 2.634003
Epoch: 20       Training Loss: 2.759795         Validation Loss: 2.621263
```

Fig 4.3 Training and Validating the Model

On plotting the loss plot for training and validating the model. We get the following graph.

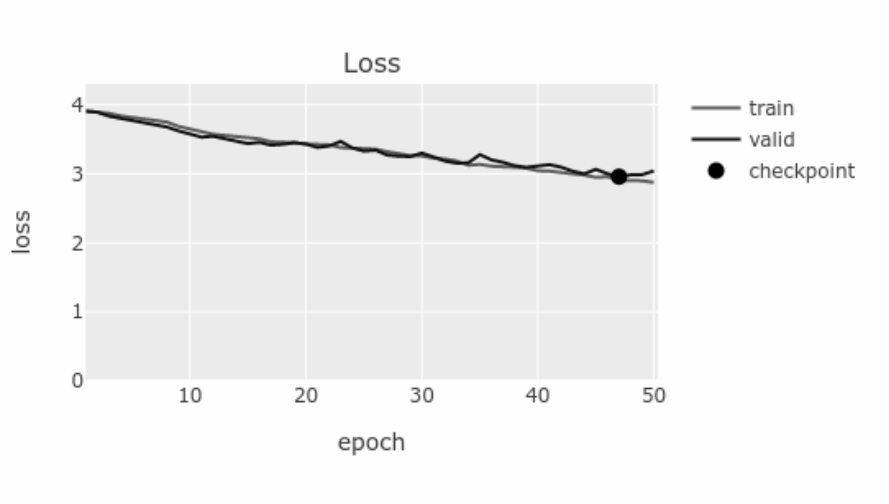


Fig 4.4 Model learning curve

Testing the model

Further testing of the model reveals a test loss of 2.50411 and a test accuracy of 38%. This accuracy is way high than the expected accuracy of 30%. Altogether this model is a success. Now moving on to the next steps of training the above model using transfer learning.

```
Test Loss: 2.450411


Test Accuracy: 38% (480/1250)
```

Fig 4.5 Test Loss and Test Accuracy

4.1.2 CNN made using Transfer Learning

This model has been made according to the parameters defined in the system development section

Training and validating the model

After training and validation of the model we get the training loss of 0.93 and validation loss of 1.002 after 14 epochs.

```
Epoch: 1        Training Loss: 2.577775         Validation Loss: 1.509118
Validation loss decreased (inf --> 1.509118).  Saving model ...
Epoch: 2        Training Loss: 1.735742         Validation Loss: 1.303752
Validation loss decreased (1.509118 --> 1.303752).  Saving model ...
Epoch: 3        Training Loss: 1.521258         Validation Loss: 1.248257
Validation loss decreased (1.303752 --> 1.248257).  Saving model ...
Epoch: 4        Training Loss: 1.402849         Validation Loss: 1.120429
Validation loss decreased (1.248257 --> 1.120429).  Saving model ...
Epoch: 5        Training Loss: 1.339077         Validation Loss: 1.085235
Validation loss decreased (1.120429 --> 1.085235).  Saving model ...
Epoch: 6        Training Loss: 1.238962         Validation Loss: 1.055327
Validation loss decreased (1.085235 --> 1.055327).  Saving model ...
Epoch: 7        Training Loss: 1.185147         Validation Loss: 1.048260
Validation loss decreased (1.055327 --> 1.048260).  Saving model ...
Epoch: 8        Training Loss: 1.131207         Validation Loss: 0.975124
Validation loss decreased (1.048260 --> 0.975124).  Saving model ...
Epoch: 9        Training Loss: 1.111621         Validation Loss: 1.002886
Epoch: 10       Training Loss: 1.050350         Validation Loss: 0.985795
Epoch: 11       Training Loss: 1.022418         Validation Loss: 0.966452
Validation loss decreased (0.975124 --> 0.966452).  Saving model ...
Epoch: 12       Training Loss: 0.988365         Validation Loss: 0.993472
Epoch: 13       Training Loss: 0.969868         Validation Loss: 0.974029
Epoch: 14       Training Loss: 0.934713         Validation Loss: 0.966944
```

Fig 4.6 Training and validation after transfer learning

Plotting a graph for the same results in the following graph shows that the loss decreases significantly after every epoch.

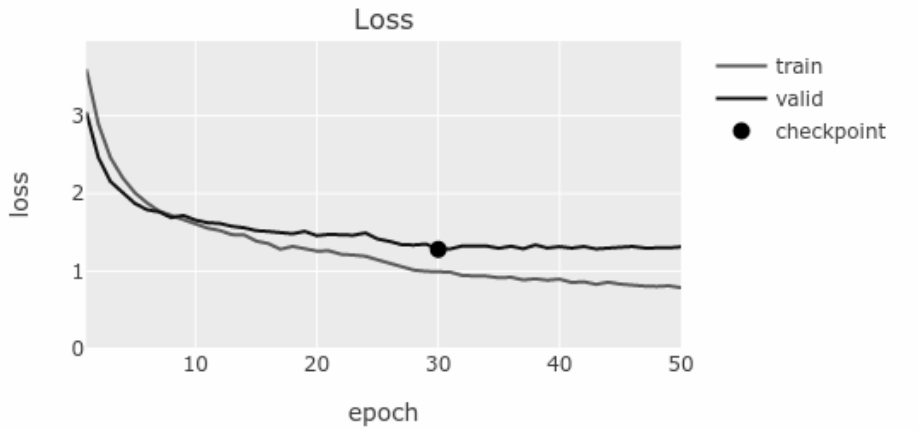


Fig 4.7 Train and validation loss after each epoch

Testing the model

The testing of the model results in an accuracy of 78% after 20 epochs.

```
Test Loss: 0.817717

Test Accuracy: 78% (976/1250)
```

Fig 4.8 Test Accuracy

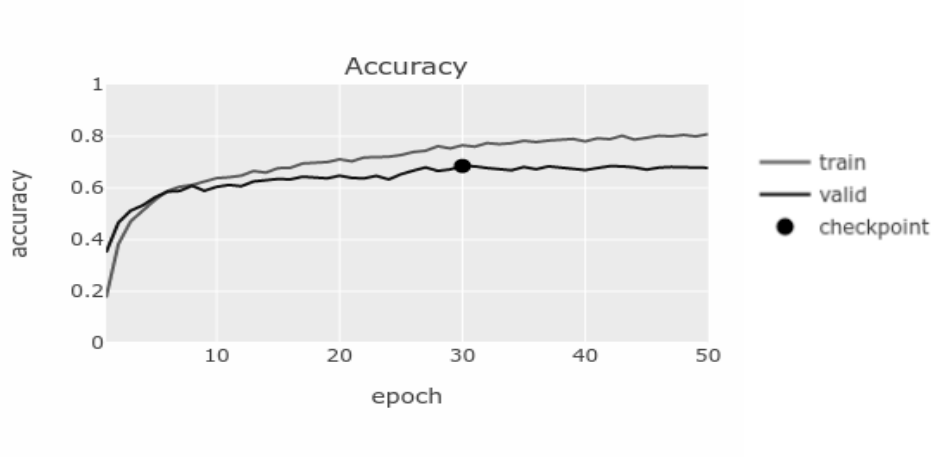


Fig 4.9 Accuracy vs epoch graph

The above graph shows that the increase in accuracy is significant as compared to the direct implementation of the VGG 16 model.

Final testing of the model with input pictures

Fig 4.10 Output on Input

CHAPTER 5
CONCLUSION

5.1 Conclusion

Deep learning techniques for landmark categorization were demonstrated in this study. We achieved an accuracy of 78% by creating a convolutional neural network (CNN) and training it on a collection of landmark photos. While this is a promising result, more work needs to be done in the future. The architecture of the CNN is one feature that could be improved. While we experimented with various network configurations, other architectures may achieve better performance.

Furthermore, we could investigate the use of transfer learning techniques to improve the performance of pre-trained models. Another area that may be improved is the dataset itself. Despite the fact that our dataset featured a wide variety of landmark photos, it was still very small, which may have hindered the model's ability to generalise to new data. We could potentially improve performance by gathering and annotating a larger and more diverse sample.

Despite these limitations, our findings suggest that deep learning techniques for landmark classification can be effective. This has practical consequences for a variety of applications, including tourism, navigation, and the protection of cultural heritage. Deep learning could become a strong tool for automatically detecting and categorising landmarks with more research and development, benefiting a wide range of fields and sectors.

5.2 Future Scope

The Performance can be improved using several factors such as -

- Learning rate

  To improve the performance of a machine learning model using the learning rate, an appropriate learning rate schedule must be chosen. A learning rate schedule is a training function that modifies the learning rate over time. Fixed, step decay, exponential decay, and polynomial decay are all examples of learning rate schedules.

  Fixed learning rate plans maintain a constant learning rate throughout training, whereas other schedules gradually lower the learning rate over time. Exponential

and polynomial decay schedules, respectively, decrease the learning rate exponentially or polynomially, allowing the model to converge faster and achieve greater performance.

- Epochs and batch size

  The number of epochs is the number of times during training that the full training dataset is processed through the neural network. Setting the number of epochs too low can cause underfitting, which occurs when the model fails to capture the underlying pattern in the data. Setting the number of epochs too large, on the other hand, can result in overfitting, in which the model learns to fit the training data too well and performs badly on fresh, unknown data.

  The batch size is the number of samples that the neural network processes in each training cycle. If the batch size is too small, the training process will be slower and more noisy. Setting the batch size too large, on the other hand, can result in faster convergence but also a higher memory requirement and poor generalisation performance.

- Early-stopping

  Early stopping is a prominent approach for boosting deep learning model performance. It entails monitoring the model's performance on a validation dataset during training and terminating the training process when the performance begins to deteriorate. Early stopping can improve generalisation performance and accuracy on fresh, unseen data by avoiding the model from overfitting to the training data. We can increase the performance of our deep learning model and avoid overfitting to the training data by adopting early stopping. This method is especially beneficial when dealing with large and complicated datasets, where overfitting is a prevalent issue. Early stopping was employed in our landmark classification project to reduce overfitting and increase model accuracy. We were able to get good generalisation performance and high accuracy on new, unseen data by carefully tweaking the patience parameter and monitoring validation performance.

- Regularization

Overfitting occurs when the neural network learns to fit the training data too well and becomes too complex, resulting in poor performance on new, unseen data. Dropout helps to address this problem by randomly dropping out some of the neurons during training. This forces the remaining neurons to learn more robust features that are useful for the classification task, even when some of the neurons are missing.

The dropout rate is a hyperparameter that determines the percentage of neurons that are dropped out during training. A high dropout rate can lead to underfitting, where the model is not complex enough to capture the underlying pattern in the data. A low dropout rate can result in overfitting, where the model becomes too complex and starts fitting the noise in the data.

As a result of this model, it is possible to develop an app that will predict the location of an image after the image has been taken from the camera by utilizing this model. The scope of this work goes beyond what the project currently entails and therefore it has not been implemented at present.

CHAPTER 6
REFERENCES

[1] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. "Fine- tuning cnn image retrieval with no human annotation. TPAMI, 2018".

[2] T. Weyand, A. Araujo, B. Cao and J. Sim, "Google Landmarks Dataset v2 – A Large-Scale Benchmark for Instance-Level Recognition and Retrieval," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 2572-2581.

[3] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large Scale image retrieval with attentive deep local features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3456–3465, 2018.

[4] J. Shijie, W. Ping, J. Peiyi and H. Siping, "Research on data augmentation for image classification based on convolution neural networks," 2019..

[5] Filip Radenovic, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In CVPR, pages 5706–5715, 2018.

[6] Shuhei Yokoo, Kohei Ozaki, Edgar Simo-Serra, Satoshi Iizuka, "Two-stage Discriminative Re-ranking for Large-scale Landmark Retrieval", 2019.

[7] Syed Kazim Raza, Syed Ammar Ahmed, Dr. Muhammad Khurram, "Visual Landmarks Recognition of Urban Structures using Convolutional Neural Network", 2020.

[8] Marvin Teichmann, Andre´ Araujo, Menglong Zhu, Jack Sim, "Detect-to-Retrieve: Efficient Regional Aggregation for Image Search", 2020.

[9] Cui, X., Zhang, W., Tüske, Z., & Picheny, M. (2018). Evolutionary Stochastic Gradient Descent for Optimization of Deep Neural Networks. In S. Bengio, H.

[10] Agnieszka Mikołajczyk, Michał Grochowski, "Data augmentation for improving deep learning in image classification problem", 2018.

[11] Wang, B. and Ye, Q., 2020. Stochastic Gradient Descent with Nonlinear Conjugate Gradient-Style Adaptive Momentum. arXiv preprint arXiv:2012.02188.

[12] Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of optimization techniques based on gradient descent algorithm: A review. PalArch's Journal of Archaeology of Egypt/Egyptology, 18(4), 2715–2743.

[13] Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), Advances in Neural Information Processing Systems (Vol.31). Curran Associates, Inc.

Appendix

Appendix A

Major Codes of the Project

```
import torch
from torchvision import datasets, transforms
import numpy as np
import splitfolders

mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
# Create Transforms
train_transform = transforms.Compose([
    transforms.RandomResizedCrop(224),
    # Augmentation
    transforms.RandomHorizontalFlip(p=0.3),
    transforms.RandomRotation(10),
    transforms.RandomChoice([
            transforms.ColorJitter(hue=0.1),
            transforms.ColorJitter(brightness=0.2),
            transforms.ColorJitter(saturation=0.2),
            transforms.ColorJitter(contrast=0.2),
        ]),
    # -------------
    transforms.ToTensor(),
    transforms.Normalize(mean, std) # Imagenet standards
])

test_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean, std) # Imagenet standards
])
```

Loading the Data

```
# Split the training folder to new training and validation folders
splitfolders.ratio("landmark_images/train", output="train_valid", seed=1337, ratio=(.8, .2), group_prefix=None)

# Load the image data
train_data = datasets.ImageFolder('./train_valid/train', transform=train_transform)
valid_data = datasets.ImageFolder('./train_valid/val', transform=test_transform)
test_data = datasets.ImageFolder('landmark_images/test', transform=test_transform)

# Save classes names
n_classes = len(train_data.classes)
classes = [class_.split(".")[1].replace("_", " ") for class_ in train_data.classes]

print('\n Num training images: ',   len(train_data))
print('Num validation images: ', len(valid_data))
print('Num test images: ',       len(test_data))

batch_size = 20
# Create data loaders
train_loader = torch.utils.data.DataLoader(train_data, batch_size, shuffle=True)
valid_loader = torch.utils.data.DataLoader(valid_data, batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_data, batch_size, shuffle=True)


loaders_scratch = {'train': train_loader, 'valid': valid_loader, 'test': test_loader}
```

Split the Data into loaders

```
[ ]

    loaders_transfer = loaders_scratch.copy()
```

Specify Loss Function and Optimizer

```
# loss function
criterion_transfer = nn.CrossEntropyLoss()


def get_optimizer_transfer(model):
  #Optimizer

    optimizer = optim.SGD(model.classifier.parameters(), lr=0.01)
    return optimizer
```

Loss Function and Optimizer

Model Architecture

Use transfer learning to create a CNN to classify images of landmarks.

```
from torchvision import models

model_transfer = models.vgg16(pretrained=True)

# Freeze training for all "features" layers
for parameter in model_transfer.features.parameters():
    parameter.requires_grad = False


n_input = model_transfer.classifier[6].in_features
model_transfer.classifier[6] = nn.Linear(n_input, n_classes) # n_classes = 50


if use_cuda:
    model_transfer = model_transfer.cuda()
```

Model Architecture (Transfer learning)

```
# train the model and save the best model parameters at filepath 'model_transfer.pt'
train(15, loaders_transfer, model_transfer, get_optimizer_transfer(model_transfer), criterion_transfer,
      use_cuda, 'model_transfer.pt')

# load the model that got the best validation accuracy
model_transfer.load_state_dict(torch.load('model_transfer.pt'))
```

Train and validate the model (Transfer learning)

Test the Model

```
[ ] test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)

    Test Loss: 0.817717


    Test Accuracy: 78% (976/1250)
```

Test the model (Transfer learning)