A black background with white text

AI-generated content may be incorrect. A black background with green and grey text

AI-generated content may be incorrect.

# 6CS012 – Artificial Intelligence and Machine Learning

**Title – Image Classification Using Convolutional Neural Networks (CNN)**

University Id: 2330520 Group: L6CG22

Tutor: Aatiz Ghimire

Submitted by: Shrota Ghimire

Submitted on: 2nd February 2025

# Abstract

The project concentrates on classifying images with the help of convolutional neural networks (CNN’s) that are known for sure to work well in the context of deep learning in image processing. An attempt is to create a fully image classification system using TensorFlow and Keras. Images are loading from a structured directory organization followed by preprocessing during which resizing and normalization of the content takes place. A CNN model that is very simple, which includes three convolutional and a pooling layer with dense layers for image classifying is developed and trained. Monitoring training allows sales monitoring in terms of Accuracy and Loss, and reviews of Classification Reports and Confusion Matrix to measure model performance. Performance of the model on the validation set is strong as indicated by evaluation. Through the successful application of deep learning to image classification, this piece sets a basis in terms of modeling extension, which could be a deeper model design or transfer of an existing model to a new task.

**Table of Contents**

[1. Introduction 1](#_Toc197897836)

[2. Dataset 1](#_Toc197897837)

[3. Methodology 1](#_Toc197897838)

[4. Experiments and Results 2](#_Toc197897839)

[4.1 Baseline Architecture Performance 2](#_Toc197897840)

[4.2 Computational Efficiency 2](#_Toc197897841)

[4.3 Optimizer Comparison 2](#_Toc197897842)

[4.4 Training Challenges 3](#_Toc197897843)

[5. Fine–Tuning or Transfer Learning 3](#_Toc197897844)

[6. Conclusion and Future Work 4](#_Toc197897845)

# Introduction

At the core of the applications of computer vision are image classification that drives technologies that are commonplace in medical imaging, autonomous vehicles, and facial identification. In the past, machine learning techniques hinged on manually implemented features, but developments in deep learning made them irrelevant. Convolutional Neural Networks (CNNs) can learn intricate features directly out of raw image pixels making them remarkably appropriate for image classification tasks. The objectives of this project include the elaboration of a CNN-based classifier for a custom set of images, evaluation of its performance, and analysis of training and classification process.

# Dataset

The dataset for this project that has been selected is titled “Flower classification” which is a custom dataset and consists of data that is divided into 2 main folders. train and test. The train set has 4,277 images with equal distribution to 5 classes, and the test set has only 5 images each also being a part of those 5 classes. In both folders the classes’ names are tulip, daisy, dandelion, sunflower and rose. To have consistency during training, all images are rescaled to a fixed size of 150 × 150 pixels. The preprocessing step consists in normalizing all pixel values to the 0-1 range using TensorFlow’s Rescaling (1/255) layer. To make sure that data is accurate, corrupted or unsupported image files are eliminated with the assistance of Python Imaging Library (PIL). Although this is a tailored dataset that’s excellent for supervised learning, its constraints like the small test set and training classes imbalances may affect the depth of evaluation of performance.

# Methodology

The key goal of this project is building customized CNN architecture to deal with multi-class image classification. The chosen architecture complies with a fixed format including three convolutional blocks, comprising of a Conv2D layer followed by a MaxPooling2D layer. The output of the convolutional blocks is fed into a Flatten layer which leads into two Dense layers of 128 and 64 units respectively. The ending Dense layer utilizes the SoftMax activation function to make probabilities for all five categories supporting the task of multi-class classification.

The training process picks the Sparse Categorical Cross entropy loss function appropriate for scenarios where multiple classes are to be classified using integer labels. The Adam optimizer is used to adjust weights in training while maintaining the configuration of the Adam optimizer to default settings. The training is implemented in totality of 15 epochs, with batch size of 32, and the learning rate implements Adam’s default value. This is of relevance, the baseline model will not have any regularization techniques such as dropout and batch normalization thus paving a possible direction for the enhancement of the model in its subsequent development phase.

TensorFlow’s image\_dataset\_from\_directory() is used in loading the dataset and makes input pipeline creation much easier. However, approaches such as shuffling, caching, and prefetching were also removed, which may improve the training rate and are applicable to further refinement.

# Experiments and Results

## 4.1 Baseline Architecture Performance

The baseline model was trained for 15 epochs and as its performance progressed, chart plots were used to track training and validation accuracy and loss. The model did well during training and validation, meaning it had indeed extracted meaningful information out of the data and not a lot on memorization. From this performance, the baseline system is good at image classification and can successfully generalize from the learned to unseen data.

## 4.2 Computational Efficiency

With the help of TensorFlow, training was successful with signs pointing towards training conducted in a CPU env instead of utilizing GPU or TPU resources from Colab. The compact model and the relatively low image resolution collectively contributed to high speeds of training, as well as fast experimentation and fine-tuning model.

## 4.3 Optimizer Comparison

The present experimental situation showed Adam optimizer only used in modeling. Adam performs efficient convergence as a rule within most tasks; however, the present study did not include the testing of Adam versus other approaches such as Stochastic Gradient Descent (SGD). It would be quite interesting to see how future experiments that are set to measure the effect of each of the different optimizers would help in understanding how the optimizers affect the convergence rate and final model accuracy.

## 4.4 Training Challenges

Stable learning was clear during the training process because there was a significant lack of overfitting. As much as that is the case, the limited use of data-augmentation strategies can inhibit the capabilities of the model to handle various real-world images appropriately. More so, such regularization methods as dropout were not considered which are common approaches to model stability and decrease in overfit that is evident. However, as regards accuracy and loss learning curves, a normal convergence pattern was observed, which implied an efficient and stable training process.

# Fine–Tuning or Transfer Learning

To increase the effectiveness of my model, I implemented transfer learning using a pre-trained VGG16 model. I dropped the preexisting levies of classification and created custom dense layers suited to the 5 flower categories of my dataset. First, I had the convolutional layers frozen, trained only the new dense ones – this way, the model was able to use the pre-trained knowledge about the ImageNet.

I then unfroze the last few layers of VGG16 to recalculate their weights using a lower learning rate for better adaptability to my flower dataset. This allowed the model to fine tune its top-level representations more appropriately to the specific features of my flower images which in turn gave rise to better performance. Accuracy of the model increased after fine-tuning and reports better performance than models that were trained without transfer learning. It correctly classified most of the test images and provided robust results out of sight data. This demonstrated a strong effectiveness for fine-tuning pre-trained models, particularly in use cases in which there are very few pieces of training data.

# Conclusion and Future Work

During this project a classification of the flowers I experimented with several CNN models: a simple baseline model, deeper model with introduced regularization, and a pre-trained model VGG16 further adapted for our dataset. The fine-tuned VGG16 Model topped in achieving the greatest accuracy, representing how with limited availability of data materials, transfer learning improves model performance. This method allowed me to build a highly effective model more quickly by leveraging existing knowledge from large image datasets such as ImageNet.

I will try to adjust the model further by using various pre-trained architectures such as – ResNet and EfficientNet – which may bring better results. Besides, I am interested in applying data augmentation and adaptive learning rate strategies to increase model generalization. My ambition is to implement the model within a simple web or mobile platform capable of providing immediate flower recognition.