***Short Report***

***BSc Computer Science***

A logo for a company

Description automatically generated

**Author Name:** Dhyan Nilesh Patel

**Assignment:** 6COM2000 Advanced AI Computer Vision Mini Project Specification

**Student ID:** 21008521

**Professor:** Christopher Roadknight

**Problem Statement**

Object recognition, a crucial component of computer vision, involves the identification and categorization of objects within the images. This mini project specifically aims to utilize object recognition techniques to analyse images and deduce the objects depicted in them. The primary objective is to create a model capable of accurately classifying images into distinct categories, with a focus on identifying whether the images contain People, Buildings, Food, or fall into an 'Other' category. A web-based company envisions enhancing its online platform to better cater to users originating from TripAdvisor, a renowned travel and restaurant review website. To achieve this, the company seeks to tailor its advertising strategies based on the content of images uploaded by users. The image classification will empower the company to target advertising more effectively, optimizing advertising revenue by aligning content with user interests.

**Dataset Description**

The dataset has pictures of different things, organized into groups. For example, there's a group for pictures of food, another for people, and so on. The computer will learn from these pictures to recognize what's in new pictures. But there are some things we need to be careful about. The dataset might have more pictures of one thing than another. For example, there might be lots of pictures of food, but fewer of something else. We need to make sure our program doesn't just get good at one thing and ignore the others. To help our program learn, we're also making the pictures all the same size. This way, the program can focus on the important parts of the pictures without getting confused. In simple terms, the goal is to make the computer understand and recognize different things in pictures, even when the pictures are a bit tricky or not the same. The dataset we're using has a mix of pictures to teach the computer about different things. (1)

**Model Selection**

Selecting a **Convolutional Neural Network (CNN)** for my object recognition task, given the limited dataset of 47 images, is a sensible decision for a few straightforward reasons. CNN excel at breaking down the image into different parts like edges, shapes, and colors – essentially solving the puzzle of what's in the picture. This innate ability makes them highly suitable for recognizing objects. CNNs are like quick learners that adapt well to small datasets. Given our limited set of 47 images, I needed a model that can make the most out of a handful of examples.

Handling variations in pictures is another strength of CNNs. They're like detectives that don't get easily confused by different angles, lighting conditions, or styles in my images. This is crucial for my task, as the pictures might have these variations, and CNNs are adept at managing them. (5)

**Data Preprocessing**

The code involves several key data preprocessing steps, each contributing to the improved performance of the Convolutional Neural Network (CNN) model for object recognition:

* **Normalization:** A normalization layer is applied to scale pixel values to the range [0,1]. This standardization ensures that all input values are within a consistent numerical range, preventing certain features from dominating others. Normalization helped stabilize and accelerate the training process by ensuring that the model is sensitive to variations in all input features.
* **Grayscale Conversion:** The images are converted to grayscale using a lambda layer. Grayscale conversion reduced the input dimensionality by eliminating color channels, potentially reducing model complexity and training time. It also simplified the learning task for the model, focusing on the luminance information in the images.
* **Data Augmentation:** Data augmentation introduced variations in the training dataset by applying transformations such as horizontal flips, rotations, and zooms. This helped the model generalize better to diverse orientations and scales of objects, improving its robustness. Augmenting the dataset with diverse perspectives of the same objects enhanced the model's ability to recognize features under different conditions, reducing overfitting and promoting better performance on unseen data.
* **Histogram Equalization:** Histogram equalization was applied as a part of data preprocessing, specifically for contrast adjustment. It enhanced the visibility of details in images by redistributing pixel intensities. In this case, it was used within the grayscale conversion lambda layer to adjust the contrast of the images, potentially aiding the model in capturing important features.
* **Performance Configuration:** To optimize data loading and training speed, the code includes configuration steps such as caching, shuffling, and prefetching. Caching ensures that the dataset is stored in memory for faster access, shuffling randomizes the order of training samples to prevent learning biases, and prefetching overlaps data loading with model training to minimize idle time.

These preprocessing techniques collectively contribute to enhancing the model's performance, enabling it to learn meaningful representations from diverse and augmented training data.

**Training and Testing Accuracy**

**Training Accuracy:** The training accuracy is observed to increase gradually over the epochs. This indicates that the model is effectively learning from the training data and improving its ability to correctly classify the training samples.

**Testing Accuracy:** The testing accuracy shows an initial increase, but then it fluctuates, possibly indicating that the model faces challenges in generalizing to new, unseen data. The fluctuations might suggest overfitting, where the model is too tailored to the training data and struggles to perform well on unfamiliar examples.

**A graph of a training and testing loss

Description automatically generated with medium confidence**

**Training and Testing Loss**

**Training Loss:** The training loss is observed to decrease steadily over the epochs, indicating that the model is fitting the training data well and minimizing errors during training.

**Testing Loss:** The testing loss initially decreases, but at some point, it starts increasing. This behavior suggests that the model is overfitting to the training data, resulting in higher loss when faced with new, unseen data. (7)

**Test on Validation Images**

The model's difficulty in correctly predicting all validation images provided may be due to several factors. Firstly, the model's architecture might not be complex enough to grasp intricate features in the validation set. To address this, trying a more complex model or adjusting the existing one could help. Secondly, a small or non-diverse training dataset could limit the model's ability to generalize to new images. Obtaining a more diverse dataset or using transfer learning methods might improve performance. The model may have overfit the training data, meaning it has learned to perform well on the training set but does not generalize well to new data. This can happen if the model is too complex or if there isn't enough regularization. This can be mitigated by adding regularization techniques. Additionally, ensuring balanced class representation, refining data augmentation, and carefully tuning hyperparameters can contribute to better predictions. (4)

**VGG16 Model and K-means clustering**

During the project, I contemplated alternative approaches, including utilizing a pretrained model like VGG16 and employing K-means clustering. While VGG16, a well-established pretrained model, is powerful for image recognition, it was not chosen due to its relatively high complexity. Given the dataset size and computational resources, a simpler architecture was favored to expedite training and accommodate the project's scope. On the other hand, K-means clustering, often used for unsupervised learning, was considered for its simplicity and interpretability. However, it relies on grouping data points without leveraging the hierarchical features crucial for image recognition tasks. The lack of inherent understanding of image structures and patterns made K-means less suitable for this specific project, where capturing intricate features is essential. Ultimately, the chosen approach involved building a convolutional neural network (CNN) from scratch, providing a balance between model complexity and performance. This tailored CNN was designed to suit the dataset's characteristics, optimizing for both accuracy and efficiency in the given project constraints. (8)

**Improvement Strategies**

To enhance the training results and overall performance of the Convolutional Neural Network (CNN), several strategies can be implemented. First and foremost, increasing the diversity of the dataset is crucial. This can be achieved by collecting more data, thereby exposing the model to a broader range of patterns and variations. Experimenting with the model architecture is another avenue for improvement. Adjusting the complexity of the CNN architecture based on observed overfitting or underfitting is essential. Fine-tuning the model's architecture, exploring different layer configurations, and considering the integration of pre-trained models like VGG16 as the dataset grows can contribute to improved performance. To enhance the training results of the neural network, we can consider incorporating "Early Stopping," a technique that interrupts training when the model's performance on the validation set plateaus, preventing potential overfitting. Another valuable adjustment involves implementing a "Learning Rate Schedule," dynamically adapting the learning rate during training to optimize convergence speed and avoid overshooting.

A group of buildings with a railing

Description automatically generated

**References**

1. *ImageNet: A large-scale hierarchical image database* (2009). <https://ieeexplore.ieee.org/document/5206848>.
2. Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017) 'ImageNet classification with deep convolutional neural networks,' Communications of the ACM, 60(6), pp. 84–90. <https://doi.org/10.1145/3065386>.
3. *Image classification from scratch*. <https://keras.io/examples/vision/image_classification_from_scratch/>.
4. Nelson, D. (2023) *Image Recognition and Classification in Python with TensorFlow and Keras*. <https://stackabuse.com/image-recognition-in-python-with-tensorflow-and-keras/>.
5. Rowe, W. (no date) How to Use Keras to Solve Classification Problems with a Neural Network. <https://www.bmc.com/blogs/keras-neural-network-classification/>.
6. Singh, A. and Singh, P. (2020) 'Image classification: a survey,' *Journal of Informatics Electrical and Electronics Engineering (JIEEE)*, 1(2), pp. 1–9. <https://doi.org/10.54060/jieee/001.02.002>.
7. *Computer vision with TensorFlow* (no date). <https://www.tensorflow.org/tutorials/images/>.
8. Flomo, G. (2021) 'How to cluster images based on visual similarity - Towards Data Science,' *Medium*, 16 December. <https://towardsdatascience.com/how-to-cluster-images-based-on-visual-similarity-cd6e7209fe34>.