MULTICLASS (CARS AND TREES) IMAGE CLASSIFCATION USING DENSENET

Contents

Introduction

Image classification is an important application in the area of computer vision that includes assigning categories to input pictures based on the visual content of those images. Image classification is an essential part of the discipline. It is vital in a variety of sectors, including as surveillance, autonomous driving, and medical diagnostics, to have the capacity to effectively identify pictures. The DenseNet model was going to be used in this project, and the goal was to categorise photos of cars and trees [2]. This report produced the dataset by utilising the aerial view of Google Maps and capturing screenshots of photographs of cars and trees in the cities of Stirling, Scotland, and Edinburgh. The code is implemented to put the DenseNet model through its paces on the training dataset before analysing how well it performed on the test dataset.

Dataset

The dataset includes 50 images of cars and trees each from Edinburgh and 50 images of cars and trees each from Stirling obtained by taking screenshots from google maps aerial view.

Sample of the dataset

Tree

Figure 1: Dataset samples

Description of the Proposed Solutions with Justifications

The report proposed using the DenseNet model for classifying car and tree images. The term "dense net" refers to densely linked convolutional networks. It is quite similar to a ResNet, although with significant essential distinctions. An additive approach is used by ResNet, which means it utilizes an output from a previous layer as an input for a subsequent layer. On the other hand, the DenseNet algorithm uses every output from a previous layer as an input for a future layer [2]. DenseNet was specifically designed to increase accuracy in high-level neural networks created by the vanishing gradient. This gradient occurs when there is a large distance across the input and output layers, and as a result, information is lost before it can get at its final destination [3].

To implement the DenseNet model, this report used the pre-trained DenseNet121 model provided by the Keras API. The code started the system with pre-trained weights using the ImageNet dataset, then added a dense layer to the model with two output nodes so that it could identify photos of cars and trees. When it came time to construct the model, turned to

the Adam optimizer and the SparseCategoricalCrossentropy loss function [1]. Then provided the model ten repetitions of training when it was in the training phase, and then used the validation dataset to assess how well it functioned.

```
Epoch 1/10
C:\Users\haris\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\backend.py:5582: UserWarning: "`sparse_categorical_crossentropy`
  output, from_logits = _get_logits(
5/5 [==============================] - 152s 26s/step - loss: 0.4938 - accuracy: 0.8264 - val_loss: 28.0079 - val_accuracy: 0.3889
Epoch 2/10
5/5 [==============================] - 116s 22s/step - loss: 0.1778 - accuracy: 0.9444 - val_loss: 108.8827 - val_accuracy: 0.3889
Epoch 3/10
5/5 [==============================] - 113s 22s/step - loss: 0.1075 - accuracy: 0.9653 - val_loss: 717.3874 - val_accuracy: 0.3889
Epoch 4/10
5/5 [==============================] - 113s 22s/step - loss: 0.0163 - accuracy: 0.9931 - val_loss: 1004.2118 - val_accuracy: 0.3889
Epoch 5/10
5/5 [==============================] - 107s 21s/step - loss: 0.0193 - accuracy: 0.9931 - val_loss: 1244.7076 - val_accuracy: 0.3889
Epoch 6/10
5/5 [==============================] - 135s 26s/step - loss: 0.0478 - accuracy: 0.9931 - val_loss: 1511.9205 - val_accuracy: 0.3889
Epoch 7/10
5/5 [==============================] - 121s 24s/step - loss: 0.0044 - accuracy: 1.0000 - val_loss: 1279.2593 - val_accuracy: 0.3889
Epoch 8/10
5/5 [==============================] - 122s 25s/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 927.1909 - val_accuracy: 0.3889
Epoch 9/10
5/5 [==============================] - 114s 22s/step - loss: 0.0670 - accuracy: 0.9792 - val_loss: 590.7307 - val_accuracy: 0.3889
Epoch 10/10
5/5 [==============================] - 116s 21s/step - loss: 0.0448 - accuracy: 0.9861 - val_loss: 509.0646 - val_accuracy: 0.3889
```

Figure 2: Model training

Results

The model is just making random guesses at the class labels of the photos if it achieves an accuracy of 0.5 on the test dataset. In other words, the model is unable to distinguish accurately between photographs of vehicles and images of trees.

There are several possible explanations for the low accuracy, including the fact that the dataset in issue is relatively small. It is indeed conceivable that a small dataset does not contain enough instances of each class, making it challenging for the model to acquire important characteristics that differentiate the classes.

In order to obtain a high degree of accuracy when employing deep learning, it is necessary for the model to be capable of acquiring relevant information. If the dataset is too small, the model may not be able to learn all of the essential features, resulting in poor performance.

```
1/1 [==============================] - 4s 4s/step - loss: 463.2037 - accuracy: 0.5000
Test F1 Score: 0.5
```

Figure 3: F1 score

Actual vs Predicted Classes for Images

In figure 4, the actual vs predicted classes for images are displayed. This shows that there are very less predictions made by the model.

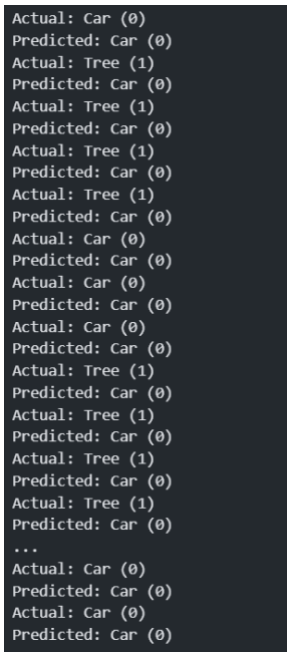

Figure 4: Actual vs predicted

Discussion of the results

There may be a number of reasons why the suggested solution performs poorly. To begin, the model would have been unable to acquire sophisticated patterns because of the tiny size of the dataset used for training. In addition, the dataset only comprises aerial views of automobiles and trees, which may not be reflective of the variety of pictures the model may experience in practise.

The use of the DenseNet model framework is another potential cause of the subpar results. DenseNet might not be the best design for this challenge, despite its proven success on other picture classification tasks. It is indeed possible that the best-performing model can only be determined by extensive testing of other models like ResNet, Inception, and MobileNet.

Conclusion

In this work, it described how to apply the DenseNet model to the problem of categorising images of automobiles and trees. The suggested approach was applied to a dataset consisting of aerial photographs of vehicles and trees in Stirling, Scotland, and Edinburgh. The findings indicated that the model performed no better than chance (at 50% accuracy).

The tiny dataset size, the lack of variety in the training data, and the use of the DenseNet model architecture are all possible causes of the subpar results. Experimentation with alternative models and data augmentation strategies may be required to increase performance.

References

[1] Tensorflow, “API Documentation | TensorFlow Core v2.4.1,” *TensorFlow*, 2023. https://www.tensorflow.org/api_docs

[2] K. Zhang, Y. Guo, X. Wang, J. Yuan, and Q. Ding, “Multiple Feature Reweight DenseNet for Image Classification,” *IEEE Access*, vol. 7, pp. 9872–9880, 2019.

[3] Z. Abai and N. Rajmalwar, “DenseNet Models for Tiny ImageNet Classification,” *arXiv:1904.10429 [cs]*, Jun. 2020.