Flower Classification using a custom Convolutional Neural Network

Y3918296

***Abstract*—With the aim to correctly classify flowers from the VGG 102 Category Flower Dataset, the experimentation of image classification and data augmentation techniques led to the creation of a Convolutional Neural Network. This was a challenge to create due to the varying class sample sizes and a small training/test data split. After the fine-tuning of the model's hyperparameters, it achieves an accuracy of 52.9%.**

***Index Terms*—convolutional neural network, machine learning, flower classification**

I. Introduction

IMAGE classification is a fundamental computer vision task of assigning a label or class to a given image. It is becoming the base for a variety of applications in different fields; in healthcare, it has taken the forefront of health informatics, enabling the classification of different imaging modalities [1]. Additionally, autonomous driving systems incorporate image classification techniques to classify and detect objects such as cars or pedestrians [2]. The aim is to construct and train a neural network classifier for the image classification of flower species from the *102 Category Flower Dataset* produced by the Visual Geometry Group of the University of Oxford [3].

LeCun *et al.* developed *LeNet5* [4], arguably the earliest known convolutional neural network architecture. This model tackles the image classification problem, specifically hand-written digit recognition, with only 5 layers. In recent years, newer models have emerged with varying number of layers and architectures. One example is *EfficientNetV2*, training faster than new, high performance models whilst also being up to 6.8x smaller [5].

Machine learning is defined as "the capacity of computers to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and infer from patterns in data" [6]. Image classification models independently identify patterns in data, forming their own decisions for problems. Correctly classifying inputs is crucial for many modern and safety-critical systems; loss of life being an extreme yet possible outcome [7]. This machine problem is tackled through the development of a popular image classification architecture: Convolutional Neural Network.

II. Method

A Convolutional Neural Network (CNN) seems a fitting choice for this problem due to its ability to extract features from images such as edges or shapes that can be built upon to create more complex features and to make more accurate decisions. The proposed architecture consists of three types of layers: a convolution layer, a pooling layer, and a fully connected layer. These layers allow for the extraction of features, summarization of data neighbourhoods, and the mapping of inputs (with the extracted features) to outputs.

A. Data Augmentation

Due to the small dataset size, data augmentation is essential to avoid overfitting and so various augmentation techniques are necessary. When applying various augmentation techniques, it became clear some were more effective than others. As a flower cannot be inverted, it seems intuitive that vertical flips of the training images significantly decrease the model's accuracy. As a result of this, the probability of vertically flipping an image has been reduced, as including the transformation could still provide some generalisation to the model. On the other hand, rotating the image would be effective in generalising the model as flowers often sway in the wind. Random Erasing [8] is also used as it is shown to increase a model's accuracy.

B. Architecture

The model takes an input of the de facto size $224 \times 224 \times 3$. It consists of 10 convolutional layers, which are split into blocks of 2. This is shown in Figure 1. Each convolutional layer is followed by an activation function and batch normalisation to help generalise the model. After the convolutional blocks, the feature maps are flattened into a 1-dimensional vector before applying 3 fully connected layers $2048 \rightarrow 512 \rightarrow 102$ to return a vector $y \in \mathbb{R}^{102}$, the number of flower species.

1) Convolutional Layer: A convolutional layer performs convolutions on the image's channels. Due to the small training set, a small filter size ensures all necessary features and information is collected and not lost. Therefore, all convolutional layers in the architecture have 3×3 filters with both the stride and padding set to 1.

2) Activation Function: After each convolutional layer, a Rectified Linear Unit (ReLU) activation function is applied, introducing non-linearity into the network [9, p.170]. It does so through the following function: $\text{ReLU}(x) = \max(0, x)$. Using an activation function reduces the model's complexity, making it easier to optimise due to its linear behaviour.

3) Batch Normalisation: Normalising the inputs of each convolutional layer with Batch Normalisation address the problem of Internal Covariate Shift [10]. This speeds up the training process of multiple convolutional layers before they are pooled. It also has the benefit of regularising the model further, improving its generalisation and performance.

4) Pooling Layer: After a convolutional block, a pooling layer is applied. The aim of a pooling layer is to reduce the spatial size of a feature, making the model less computationally expensive when processing the data. In the architecture, Max

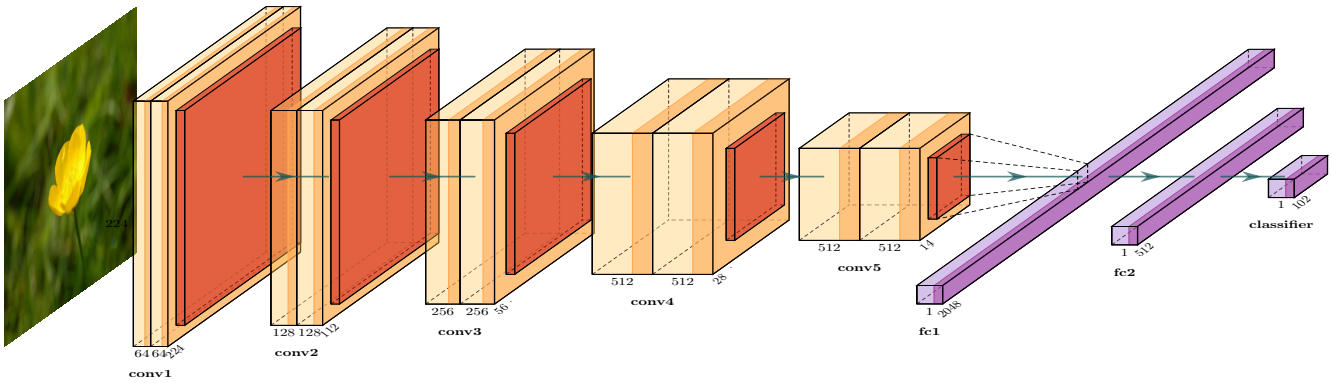

Fig. 1. A diagram showing the designed CNN architecture

Pooling layers are used, which find the maximum value from a specified filter over the data. Using a Max Pooling layer over others (Average Pooling layer) gives the benefit of ignoring noisy data as well as dimension reduction.

5) Dropout: Dropout is applied before each fully connected layer and after the last convolutional block to help further prevent overfitting. This is done by giving neurons and their connections a probability $p = 0.5$ of dropping out during the training process [11].

6) Fully Connected: Fully connected layers are implemented after the convolutional blocks to map the neurons to the eventual number of classifiers. The layers are staggered from 2048 down to 102 as these features appear to give the best results.

C. Loss Function

The Cross Entropy Loss Function is effective when performing multi-class image classification. With this being the problem at hand, it felt to be a fitting choice for training the architecture. This is calculated by the following:

$$\ell(x, y) = \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} l_n$$

Where C is the number of classes (102), x is the input, y is the target, w is the weight, N is the mini-batch dimension and

$$l_n = -w_{y_n} \log \frac{\exp\left(x_{n,y_n}\right)}{\sum_{c=1}^{C} \exp\left(x_{n,c}\right)}$$

which is equivalent to applying the $\log(\text{Softmax}(x))$ function on the output (the probabilities of being a given class) with the Negative Log Likelihood Loss applied after.

III. Results & Evaluation

To maximise the training dataset accuracy, the model's hyperparameters must be fine-tuned. Experiments were therefore conducted and recorded in Table I. The experiments were performed on Viking Computer Cluster [12], using either an NVIDIA A40 or H100 GPU for 300 epochs. A model's performance will be based on the categorisation accuracy of the model, a straightforward yet effective evaluation metric analysing how well the models predict the flower species.

TABLE I
Accuracy of model hyperparameter variations after 300 epochs

	Batch Size	Learning Rate	Accuracy %
Variation 1	32	0.001	12.2%
Variation 2	64	0.001	36.4%
Variation 3	128	0.001	38.9%
Variation 4	32	0.0001	50.1%
Variation 5	64	0.0001	52.9%
Variation 6	128	0.0001	52.0%
Variation 7	32	0.00001	44.1%
Variation 8	64	0.00001	41.86%
Variation 9	128	0.00001	38.97%

The Adam optimiser [13] was used as it handles 'noisy' data well and has an adaptive learning rate. This is useful for a small training dataset and ensures the network's accuracy is maximised by changing the learning rate when needed.

Fine-tuning Variation 5 (500 epochs with a weight decay of $1e^{-4}$) gave worse results, hence Variation 5 is the best model with an accuracy of 52.9%, taking 1 hour and 40 minutes.

IV. Conclusion & Further Work

In conclusion, it appears the model has limited success, producing a low accuracy of 52.9%. However, for a relatively shallow CNN architecture compared to deep networks such as ResNet152 [14], it could still correctly predict some flowers.

It is believed the data augmentation had a large positive impact on the model accuracy; precise adjustments to the data transformations allowed for effective generalisation and a higher model accuracy. On the other hand, I do not believe the designed architecture is an effective one. It appears to converge relatively quickly and still overfit the data. For a very small dataset and a significantly larger test set, this is to be expected. The fine-tuning of the hyperparameters was also successful, boosting the model's accuracy from 12.2%.

Looking to the future, it would be interesting to see the effectiveness of different data augmentation techniques on specific Image Classification models and the how much each augmentation contributes to a model's accuracy.

REFERENCES

[1] E. Miranda, M. Aryuni, and E. Irwansyah, "A survey of medical image classification techniques," in *2016 International Conference on Information Management and Technology (ICIMTech)*, 2016, pp. 56–61.

[2] P. Ramakrishnan, K. Dhanavel, K. Deepak, and R. Dhinakaran, "Autonomous Vehicle Image Classification using Deep Learning," in *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*. Erode, India: IEEE, Mar. 2023, pp. 97–104. [Online]. Available: https://ieeexplore.ieee.org/document/10104830/

[3] "Visual Geometry Group - University of Oxford." [Online]. Available: https://www.robots.ox.ac.uk/~vgg/data/flowers/102/

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[5] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," Jun. 2021, arXiv:2104.00298 [cs]. [Online]. Available: http://arxiv.org/abs/2104.00298

[6] Oxford English Dictionary, "machine learning, n." Mar. 2024. [Online]. Available: https://doi.org/10.1093/OED/2166790335

[7] "Cruise self-driving cars investigated after two accidents," *BBC News*, Oct. 2023. [Online]. Available: https://www.bbc.com/news/technology-67133409

[8] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," 2017, _eprint: 1708.04896.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[10] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015, _eprint: 1502.03167.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[12] "Viking Computing Cluster - IT Services, University of York." [Online]. Available: https://www.york.ac.uk/it-services/tools/viking-computing-cluster/

[13] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017, _eprint: 1412.6980.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015, _eprint: 1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385