Flowers 102 Classifier

Y3920141

***Abstract*—This report presents the development and evaluation of a Convolutional Neural Network (CNN) for classifying images in the Oxford 102 Category Flower Dataset (flowers-102). This dataset is provided by the Visual Geometry Group at the University of Oxford and consists of 102 flower categories. A custom Convolutional Neural Network (CNN) was designed with multiple convolutional, pooling, and fully connected layers, and the model was trained using cross-entropy loss and the Adam optimizer. The best model achieved a classification accuracy of 55.34% on the Flowers-102 test set.**

I. Introduction

The scenario for this task is to define the images into specific classes by constructing a neural network classifier that takes a given 2D input image and predicts a specific class as its output. Image classification is a fundamental task in computer vision, with applications spanning object and face recognition, medical image analysis, autonomous driving, and more. It is a crucial problem because it enables machines to interpret and understand visual information, which has become essential for many real-world applications. In recent years, Convolutional Neural Networks (CNNs) have established themselves as the dominant architecture in the image classification domain due to their effectiveness. These deep learning models learn hierarchical features directly from pixel data to identify patterns within images. A key milestone in the development of computer vision includes the creation of LeNet-5 by Yann LeCun, which became the foundation for CNNs [1]. The next breakthrough that drastically impacted this field was AlexNet, which significantly improved performance on the benchmark dataset ImageNet and demonstrated the potential of deep learning [2]. This success led to further development of other influential network architectures like VGG [3] and ResNet [4]. In this project, the flowers-102 classification task is tackled by designing and training a custom CNN architecture from scratch utilizing approaches like data augmentation, batch normalization, dropout, and the use of a learning rate scheduler to enhance the performance on the test set.

II. Method

In this section, we explore the approach taken to build the CNN for the flowers-102 classification task.

a) ***Data Augmentation:*** The Flowers 102 dataset contains 8169 images of 102 categories of flowers commonly occurring within the United Kingdom borders[5]. The training and validation sets each consist of 10 images per class, totaling 1020 images, while the test set consists of the 6149 images. This is a remarkably imbalanced split of the data, which significantly challenges creating a powerful model without leveraging techniques such as transfer learning. To mitigate these issues, data augmentation and normalization techniques were employed using PyTorch `v2` torchvision library. Augmentation techniques including random rotation, random resized crop, random horizontal and vertical flip, and color jittering were employed for the training set to help expand its diversity.

b) ***Network Architecture:*** CNNs are a specific type of deep neural network characterized by their renowned success in visual image analysis. The network takes as an input 3-channel RGB image and applies 64 filters of size 3x3 with padding to maintain spatial dimensions. This process is followed by a batch normalization layer to normalize the outputs, ensuring stable gradients during training[6]. Each convolution and batch normalization operation is succeeded by the Rectified Linear Unit (ReLU) activation function, introducing nonlinearity into the model and enabling the learning of complex input relationships. The final step is the pooling layer, which is responsible for dimensionality reduction, thereby reducing the computational load. In the model, a 2x2 Max Pooling filter is used, which selects the pixel with the maximum value to send to the output array, reducing the spatial dimensions of the input image. Subsequent deeper convolutional layers increase the number of filters to 128, 256, and 512 respectively. Post-convolution, the feature maps are flattened into a one-dimensional vector of size 512 * 7 * 7, representing 25088 features, and passed through the final two fully connected layers(FC) of the CNN. The first FC consists of 4096 neurons, followed by batch normalization, ReLU activation, and a 50% dropout rate. The second FC, consists of 102 neurons, which corresponds to the number of classes in the dataset.

c) ***Loss Function:*** To calculate the loss function, the Categorical Cross-Entropy Loss, which is a common choice for multi-class classification tasks, was used. The performance of the classifier is measured by quantifying the difference between the predicted probabilities and the actual class labels, which is aimed to be minimized.

d) ***Training Procedure:*** Initially, training was conducted using Stochastic Gradient Descent (SGD), which was replaced by the Adam optimizer due to its adaptive learning rate properties, leading to faster convergence. According to Dogo et al. (2018), Adam combines the advantages of both AdaGrad and RMSProp, making it suitable for a wide range of problems and ensuring efficient handling of sparse gradients and noisy data [7]. The learning rate was set to 0.0001 with a weight decay of 0.0001, and a Cosine Annealing Learning Rate Scheduler was employed to adjust the learning rate during training. In order to find a more optimal global minimum, scheduler gradually decreases the learning rate every 7 epochs following a cosine curve. The training was conducted over 150 epochs, achieving the best validation accuracy of 62.94%. Training over more epochs will potentially yield further improvements, as validation loss was consistently decreasing.

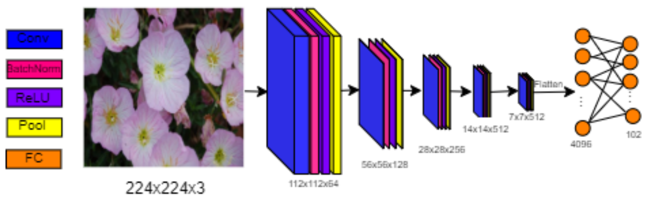

Fig. 1. Visualisation of the network architecture

Dropout Rate	FC Layer	Train Loss	Train Acc	Val Loss	Val Acc
0.2	(2048, 102)	1.6488	0.6196	2.1372	0.4882
0.5	(2048, 102)	2.0818	0.5176	2.3555	0.4431
0.2	(4096, 102)	0.9777	0.7843	1.8759	0.5461
0.5	(4096, 102)	1.1774	0.7520	1.9000	0.5333

TABLE I
IMPACT OF FULLY CONNECTED LAYER SIZE AND DROPOUT RATE ON PERFORMANCE

III. RESULTS AND EVALUATION

In order to fine-tune the network architecture and hyper-parameters, a series of experiments were conducted. The classifier underwent numerous experimental changes by altering values and observing the consequences. Initially, the architecture consisted of three convolution layers. To capture more of the fine-grained details, two additional convolution layers were incorporated, increasing the output channels to 512.

Experimentally, the learning rate scheduler was changed from StepLR to Cosine Annealing due to its smoother and more gradual learning rate decay, which led to better convergence on this small and imbalanced dataset.

Upon testing different batch sizes, a batch size of 128 was found to be the most suitable, even though it was memory-consuming. Various learning rates were tried, ranging from 0.01 to 0.000001, with 0.0001 was the best fit for the given task, balancing good convergence speed and stability. This selection aligns with findings about relationship between batch sizes and optimal learning rates, particularly for Adam-style optimizers [8].

A. Evaluation Metrics

The primary metric used to evaluate the performance of the CNN was classification accuracy. This was calculated as the ratio of correctly predicted images to the total number of images in the set. This metric applies to both validation sets during training and test sets during the evaluation stage. Additionally, training and validation losses were monitored to provide insights on how well the model was learning the patterns. For these calculations, the Categorical Cross-Entropy Loss function was used.

The final improvements of the model were achieved by experimenting with different configurations of the second fully connected layer and dropout rates. The table below represents the results of the experiments:

Increasing the size of the fully connected layer from 2048 to 4096 improved all the monitored metrics, indicating that larger fully connected layers are beneficial for learning more complex patterns, although it is not always the case for small datasets due to the high risk of overfitting. Higher dropout rates helped mitigate overfitting of the model, although higher rates may also degrade performance. The use of dropout has been widely researched and well-known for its generalization improvement [9].

B. Reproducibility

All the results can be reproduced by preserving the defined architecture, following the approaches utilized in this classifier to enhance its performance, including data augmentation and usage of the learning rate scheduler, and maintaining the values of the hyper parameters.

C. Final Test Accuracy

The final accuracy achieved by the model was 55.34%. The training of the model was executed on Google Compute Engine, utilizing a T4 GPU with 15.0 GB of GPU RAM and 12.7 GB of system RAM. The programming environment used was Python 3 and PyTorch was the deep learning framework employed. Training the network over 150 epochs took 52 minutes and 16 seconds.

IV. CONCLUSIONS

The aim of this project was to classify the images from the Flowers-102 dataset, which was achieved with an accuracy of 55.34% on the test set. This might be considered as a reasonable level of success, given the fine-grained nature of the dataset and its complexity. The Oxford 102 Flowers can be used as a benchmark for image classification techniques due to its high intra-class variability and imbalanced data distribution.

The designed architecture seems to be a good choice for this task, especially after introducing modifications during the experimentation phase. The training loss consistently decreased and the training accuracy reached up to 91.76% by the final epoch. The validation loss also showed a downward trend over many epochs, with some fluctuations. The best validation accuracy was 62.94%, indicating the model's success at predicting unseen data, although the improvement plateaued during certain periods. Training the model for a longer period of time could have potentially led to better results. However, due to hardware limitations, such as the memory consumption of larger batch size, significant GPU RAM and system RAM were required.

For further work, several approaches could be explored, including the use of more sophisticated data augmentation techniques, experimenting with other loss functions, automated hyperparameter tuning, early stopping methods, and experimenting with more advanced architectures like ResNet[4] or VGG[3].

References

1) LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE.
2) Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems.
3) Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
4) He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR).
5) Nilsback, M. E., Zisserman, A. (2008). Automated flower classification over a large number of classes. In 2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing (pp. 722-729). IEEE.
6) Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning* (pp. 448-456).
7) Dogo, E. M., Osho, O., Sanuade, O., Afolabi, O., Ayo, F. E., & Ajayi, G. O. (2018). Adaptive learning rate optimization strategies in artificial neural network model training for medical data sets. *MDPI Electronics*, *7*(3), 56.
8) Li, S., Zhao, P., Zhang, H., Sun, X., Wu, H., Jiao, D., Wang, W., Liu, C., Fang, Z., Xue, J., Tao, Y., Cui, B., Wang, D. (2024). Surge Phenomenon in Optimal Learning Rate and Batch Size Scaling. arXiv preprint arXiv:2405.
9) Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929-1958. Available: https://www.cs.toronto.edu/ rsalakhu/papers/srivastava14a.pdf