Image Classification Neural Network

Tsveta Ivanova

***Abstract*—The task for this project is to construct a machine-learning model that can classify images of flower species into name categories. The model learning the flower images dataset is a convolutional neural network comprising three main types of layers – convolutional layers, pooling layers and fully connected layers. The labelled images in the flowers-102 dataset are split into predefined varieties. They share a large between-class similarity [1], as flowers generally have similar structures, shapes and organisation of petals, stems, and leaves. In addition, the dataset exhibits small within-class similarity [1], due to lighting, angles, and colours. Consequently, this phenomenon contributes to the complexity of accurately categorising the numerous flowers. The achieved classification accuracy of the convolutional neural network model on the Oxford flowers-102 test set is 55.13%.**

I. Introduction

THE neural network model is trained on the pre-split Oxford flowers-102 dataset, which is provided by PyTorch's 'torchvision.datasets' module. The objective is to achieve categorisation accuracy without using pre-trained models. This is accomplished by fine-tuning the hyper-parameters, such as the epochs, learning rate, optimiser, batch size, weight decay, scheduler, and dropout rate. By controlling the number of input and output tensors, the stride and kernel size - the capture of unique intricate features of the flower categories improves. Each layer of the network learns to identify various aspects of the images and progressively extracts more complex and abstract representations. Image classification has been evolving rapidly as this task is key to computer vision's advancement. Computer vision is used widely in medicine for imaging analysis, in agriculture for crop monitoring, and in the autonomous vehicle industry for real-time image classification, ensuring safety and accurate surroundings interpretation for vehicles on the road. The proposed convolutional neural network's purpose is to classify images of flower species, which is an important undertaking for maintaining diverse species, natural pollination and bee health, essential for the balance in the ecosystem. Historically, image classification started with hand-engineered features that were fed into classical machine learning algorithms, while effective, they could not handle intricate details and complex images[3]. Research on Convolutional Neural Networks (CNN) started in the 1980s with Yann LeCun and his collaborators' research and started to be more widely applied in 2010. These models revolutionized image processing by automating the representation of low-level features on images, such as shapes, edges, and textures [3]. Currently, the convolutional neural network computes similarly to how the human brain digests visual inputs by extracting meaningful and low-dimensional features, filtering through the kernels of the convolutional layers and grouping similar features, the output of one layer is the input of the next. In the flower classification problem, the work begins with the multiclass dataset, which represents 102 categories of flowers, each category containing between 40 and 258 images. The data is split by the module 'torchvision' into a predefined train set – 1020 images total and 10 images per class, validation set - 1020 images total and 10 images per class and test set - 6149 images total and approximately 60 images per class. The technique of splitting datasets is widely utilised to strategically maximise the learning potential of the model by allowing the model to train, improve and correct on a portion of the dataset, then validate, without learning or influencing the parameters, but monitoring for potential overfitting and finally, testing on unseen images, to evaluate the generalisation capabilities.

II. Method

The approach to successfully creating and training a Convolutional Neural Network involves data preprocessing and augmentation. The torchvision flowers-102 dataset is split into predefined sets. In addition, the train and validation sets are over six times smaller in the number of images per set than the test set. Consequently, the reliance on data augmentation for the training set was a prerequisite for the model's effective training and avoiding overfitting maximally. The images were resized and cropped in the centre to retain the aspect ratio and ensure uniformly shaped and sized input, as the neural network requires images of the same size for the input data. As data quality and quantity are essential, considering the limited training set, applying the random horizontal flip and random rotation would increase the diversity and improve the generalisation of the model. Normalisation is applied to stabilise the training and help the optimisation process - gradient descent[13]. The images are normalised using the mean and standard deviation of the ImageNet [4] dataset that is used widely on nature-based images and proved to benefit the performance of the model better than with the custom mean and standard deviation on the flower images. The network has four convolutional layers, with a funnel approach preferred for the filter design, with 3 input channels and 64 output channels, the second convolutional layer has 64 input channels and 128 output channels, and the following two have 128 input channels and 256 output channels and 256 input channels and 512 output channels respectively. All the convolutional layers have a 3x3 kernel size and a padding of 1. As our dataset consists of complex in shape, colour and size elements, having more layers means a deeper network that can learn more details about the image and better detect patterns. The deeper the network the more details it can learn as every layer learns progressively more complex patterns and features. The loss function used is the CrossEntropyLoss, it is equivalent to applying a LogSoftmax on an input followed by NLLLoss [7]. CrossEntropyLoss[19] takes the probability prediction that

LogSoftmax produces and measures the distance from the truth values. Furthermore, as this is a loss function, the output of the LogSoftmax function is the prediction and we have the fact[4]. The formula for the CrossEntropyLoss follows next:

$$\text{CrossEntropyLoss} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log(p_{ic}) \tag{1}$$

Where:

- N is the number of samples.
- C is the number of classes.
- y_{ic} is a binary indicator (0 or 1) if class label c is the correct classification for sample i.
- p_{ic} is the predicted probability that sample i belongs to class c.

The input image passes through a convolutional layer, each layer is followed by a batch normalisation, ReLU activation function and max pooling. The output of the final pooling layer is flattened into a single vector. The flattened vector is then passed through two fully connected layers with ReLU activation, batch normalisation and two dropouts. The input image has three dimensions – RGB - or height, width and depth. The kernel will move along the receptive image pixels and there will be a dot product computation between the filter and the image pixel[12]. The stride is the number of pixels the filter will move over the matrix. After each convolution computation, a Rectified Linear Unit is applied that introduces non-linearity to the model. The max pooling layer reduces the number of parameters in the input by selecting the pixel with the maximum value and sending it to the output array. Pooling helps reduce overfitting, as well as increasing efficiency. The fully connected layers operate by establishing a connection between an output node and an input node, where all nodes are connected [10].

III. Convolutional Neural Network

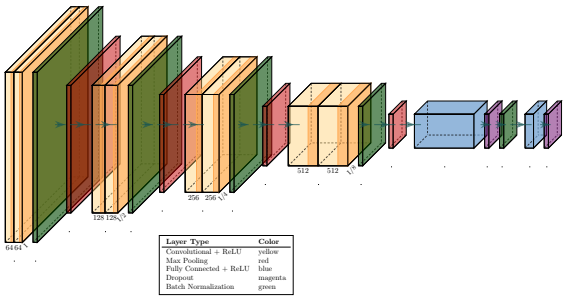

Fig. 1. Convolutional Neural Network Architecture

IV. Results & Evaluation

The final test accuracy of the model is 55.13% after running for four astronomical hours on 1000 epochs, using a CUDA GPU in a Google Colab notebook. The main approach is to adjust the hyperparameters gradually and sequentially, so that it is possible to pinpoint which hyperparameters improved the performance of the model and how. In the first experiment, the three splits of the dataset relied on the same transform pipeline[6]. This cannot be allowed as the training dataset relies on image augmentation, and the validation and test do not. In the second experiment, the transforms for train and validation were the same. This is not a traditional approach and even though the results were not vastly different to the final best model it would be recommended to keep the test and validation transforms the same[5],[7]. The validation is there to monitor the test data, and not to influence the model directly. The size of the images affected the speed with which the model was training, from 128x128 pixels, 256x256 px and 512x512 px. Whereas with the 128-pixel image, the model trained rapidly with lower accuracy. With the 512-pixel data input, the model was not computationally efficient. The layers involved continuous changes to be able to understand how the model could generalise better. The CNNs of the three experiments were diverse, with many layers, with max pooling and without, in addition, a second dropout improved the overfitting tendency by 5%. One of the most impactful adjustments is the switch from 'v1'.transforms to 'v2'.transforms. 'v2' transforms are recommended by PyTorch promising a 5-10% improvement in performance. The Adam optimiser is known to be more robust and adaptive, as well as less computationally demanding, yet it did not outperform Stochastic Gradient Descent. The several experiments that were performed throughout were immensely intriguing, considering that adding too many layers or building with too few layers resulted in lowered accuracy and overfitting. The suspicion is that this is due to the number of images in the dataset, their unique similarities and differences, as well as the pre-split train, validation and test sets quantities. If the model was overly complex, it did not perform well. Similarly, using less than three convolutional layers led to poor performance. The funnel approach, where the number of filters increases with depth outperformed the wizard's hat, where the number of filters decreases with depth. Max pooling was effective in down sampling the feature maps. For the optimisation, the parameters that were chosen are a learning rate of 0.001, a momentum of 0.9 and a weight decay of 0.001 with a scheduler Step LR with optimiser, step size of 50 and gamma=0.1. StepLR outperformed Reduced LR On Plateau and Cosine Annealing. In addition, different combinations were tried between optimisers and schedulers.

TABLE I
Comparison between three models of image classifiers.

	Model 1	Model 2	Model 3
Validation Accuracy %	54.41	53.80	53.82
Validation Loss	1.88	1.95	1.94

V. Conclusion & Further work

The performance on the test set is satisfactory considering that there are no pre-built models used and the dataset is not very easy to classify due to its between class similarities and within class similarities. The Convolutional Neural Network Architecture is managing to classify with over 50%, however, it is overfitting during 90% of the epochs, if this is overcome then the accuracy would grow incrementally.

VI. REFERENCES

[1] M.-E. Nilsback and A. Zisserman, "Automated Flower Classification over a Large Number of Classes," University of Oxford, 2008. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/publications/2008/Nilsback08/nilsback08.pdf. [Accessed: 26-May-2024].

[2] S. Amidi, "Evolution of Image Classification Explained," Stanford University, [Online]. Available: https://stanford.edu/~shervine/blog/evolution-image-classification-explained. [Accessed: 26-May-2024].

[3] F. Shoukkathali, "The Evolution of Image Classification: From Handcrafted Features to Convolutional Neural Networks," Medium, 2018. [Online]. Available: https://medium.com/@fathimathul.shoukkathali/the-evolution-of-image-classification-from-handcrafted-features-to-convolutional-neural-networks-4814607bbb39. [Accessed: 26-May-2024].

[4] PyTorch, "CrossEntropyLoss," [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html. [Accessed: 26-May-2024].

[5] PyTorch, "Data Transformation for Training and Validation Data," [Online]. Available: https://discuss.pytorch.org/t/data-transformation-for-training-and-validation-data/32507. [Accessed: 26-May-2024].

[6] PyTorch, "Transforms," [Online]. Available: https://pytorch.org/vision/main/transforms.html. [Accessed: 26-May-2024].

[7] PyTorch, "Transforms RST," [Online]. Available: https://github.com/pytorch/vision/blob/main/docs/source/transforms.rst. [Accessed: 26-May-2024].

[8] S. Hung, "Flower102 Transfer Learning," [Online]. Available: https://shannonhung.github.io/en/posts/flower102-transfer-learning/#Use-Transfer-Learning. [Accessed: 26-May-2024].

[9] Rumn, "Ultimate Guide to Fine-Tuning in PyTorch (Part 3): Deep Dive to PyTorch Data Transforms," Medium, [Online]. Available: https://rumn.medium.com/ultimate-guide-to-fine-tuning-in-pytorch-part-3-deep-dive-to-pytorch-data-transforms-53ed29d18dde. [Accessed: 26-May-2024].

[10] IBM, "Convolutional Neural Networks," [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks. [Accessed: 26-May-2024].

[11] PyTorch, "The denormalization for imshow," [Online]. Available: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html. [Accessed: 26-May-2024].

[12] S. Amidi, "CS 230: Deep Learning Cheatsheet," Stanford University, [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#:~:text=Fully%20Connected%20(FC)%20The%20fully,objectives%20such%20as%20class%20scores. [Accessed: 26-May-2024].

[13] Stack Overflow, "Why PyTorch Officially Use Mean 0.485, 0.456, 0.406 and std 0.229, 0.224, 0.225," [Online]. Available: https://stackoverflow.com/questions/58151507/why-pytorch-officially-use-mean-0-485-0-456-0-406-and-std-0-229-0-224-0-2. [Accessed: 26-May-2024].

[14] "arXiv:1608.06993v1," [Online]. Available: https://arxiv.org/pdf/1608.06993. [Accessed: 26-May-2024].

[15] "arXiv:1912.11370v3," [Online]. Available: https://arxiv.org/pdf/1912.11370v3. [Accessed: 26-May-2024].

[16] M. Awrangjeb, "Flower Image Classification Using Deep Convolutional Neural Network," ResearchGate, [Online]. Available: https://www.researchgate.net/profile/Mohammad-Awrangjeb/publication/352093979_Flower_Image_Classification_Using_Deep_Convolu

tional_Neural_Network/links/610b6af41ca20f6f86000d73/Flower-Image-Classification-Using-Deep-Convolutional-Neural-Network.pdf. [Accessed: 26-May-2024].

[17] F. Shoukkathali, "The Evolution of Image Classification: From Handcrafted Features to Convolutional Neural Networks," Medium, [Online]. Available: https://medium.com/@fathimathul.shoukkathali/the-evolution-of-image-classification-from-handcrafted-features-to-convolutional-neural-networks-4814607bbb39. [Accessed: 26-May-2024].

[18] IBM, "Convolutional Neural Networks," [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks#:~:text=It%20requires%20a%20few%20components,to%20RGB%20in%20an%20image. [Accessed: 26-May-2024].

[19] Unpack AI, "Cross-Entropy Loss in ML," Medium, [Online]. Available: https://medium.com/unpackai/cross-entropy-loss-in-ml-d9f22fc11fe0. [Accessed: 26-May-2024].