Neural Networks

Exam Number: Y3917254

***Abstract*—This report details the development and evaluation of a CNN for classifying the Flowers-102 dataset. The architecture features two convolutional layers with ELU activation, batch normalization, max pooling, dropout, and fully connected layers for final classification. Data augmentation techniques were applied, and the Adam optimizer with a cyclic learning rate scheduler was used for training. The CNN achieved a 48.74% accuracy on the Flowers-102 test set.**

I. Introduction

Classifying images with Convolutional Neural Networks (CNNs) involves training a model to identify the categories of new images based on a set of labeled training images. In the past, this task depended on manually crafted features and traditional machine learning methods such as SVMs and k-NN, which often lacked the ability to adapt to various datasets. The rise of deep learning, especially CNNs, revolutionized this domain by enabling automatic learning of feature hierarchies directly from the data, leading to notable gains in precision and robustness, as evidenced by the results on comprehensive datasets like ImageNet.The significance of image classification spans critical sectors, including medical diagnostics, self-driving cars, and the sorting of digital images, where precise models are crucial for dependability and operational efficiency. This report focuses on the Flowers-102 classification task, which involves distinguishing between 102 different flower species based on their images. I made a CNN with convolutional layers for feature extraction, batch normalization for consistent training, dropout layers to prevent overfitting, and dense layers for classification. To enhance generalization, I used data augmentation techniques such as random flips, rotations, and color variations.

II. Method

This section describes the architecture of the convolutional neural network (CNN) developed for classifying images from the FLowers-102 dataset, along with the training procedure and justification for chosen methods.

A. Network Architecture

The CNN architecture consists of two convolutional layers, each followed by batch normalization and max-pooling layers, to extract hierarchical features from the input images. The architecture also includes dropout layers to prevent overfitting and fully connected layers to facilitate the final classification.

The detailed architecture is as follows:

- **Convolutional Layer 1:** 32 filters of size 3x3, with padding of 1, followed by batch normalization and ELU activation.
- **Max Pooling Layer 1:** 2x2 pooling.
- **Convolutional Layer 2:** 64 filters of size 3x3, with padding of 1, followed by batch normalization and ELU activation.
- **Max Pooling Layer 2:** 2x2 pooling.
- **Fully Connected Layer 1:** 256 units, followed by batch normalization, ELU activation, and a dropout rate of 0.4.
- **Output Layer:** 102 units (corresponding to the number of flower categories), with softmax activation for classification.

B. Data Augmentation

To enhance the generalization capability of the model, extensive data augmentation techniques were employed during the training phase. These techniques included:

- Random horizontal and vertical flips
- Random rotations up to 70 degrees
- Color jittering (brightness, contrast, saturation, hue)
- Random affine transformations (shear, scale)
- Random grayscaling with a probability of 0.4

C. Training Procedure

The training dataset was loaded and augmented using the torchvision library. The Adam optimizer was used for training, with a learning rate (lr) of 0.00001 and weight decay of 0.0005. To further improve convergence, a cyclic learning rate scheduler was employed, varying the learning rate between 0.00001 and 0.0001 over 50 iterations.

The loss function used was cross-entropy loss, defined as:

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

where y is the true label and $\hat{y}$ is the predicted probability distribution.

1) Evaluation: The model's performance was evaluated using accuracy on the validation and test datasets by calculating the ratio of correct predictions to total samples. Training and validation losses and accuracies were monitored to ensure generalization. The final test accuracy was obtained by comparing the model's predictions to actual labels on the test set.

III. Results & Evaluation

A. Experiment Description

The experiments used a CNN for the Flowers-102 image classification task. The architecture includes convolutional layers for feature extraction, batch normalization for stability, dropout layers for regularization, and fully connected layers for classification.

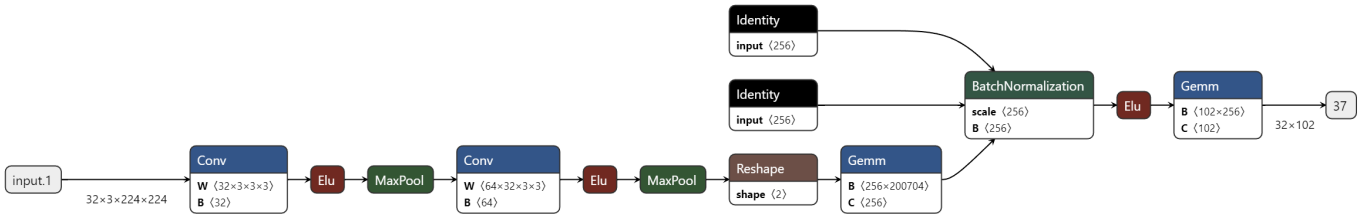

Fig. 1. Network Architecture Diagram

B. Evaluation Metrics

The primary performance metric was classification accuracy, calculated as the ratio of correct predictions to total images, for both validation and test datasets. Training and validation loss values were also monitored to assess model convergence.

C. Results & Initial Conclusions

Summary of the results and conclusions from various experiments conducted during this project:

- **Experiment 1:** The initial simple model with only one CNN layer and ReLU activation achieved a test accuracy of 6.0%, indicating insufficient complexity.
- **Experiment 2:** dding data augmentation techniques like `RandomRotation()`, `ColorJitter()`, and `RandomGrayscale()` increased the test accuracy to 8.6%, enhancing the model's ability to generalize.
- **Experiment 3:** Increasing the number of convolutional layers to 2 improved the test accuracy to 17.8%, allowing the model to capture more detailed features.
- **Experiment 4:** Adding batch normalization further improved the test accuracy to 23.0%, stabilizing and accelerating the training process.
- **Experiment 5:** Changing the activation function to ELU addressed the dying ReLU problem, resulting in a significant improvement to 41.0%.
- **Experiment 6:** Switching the scheduler from OneCycleLR to CyclicLR provided better learning rate adjustments and led to the final test accuracy of 48.74%, demonstrating improved convergence and generalization.

D. Justification of Choices

The CNN was chosen for its proven effectiveness in image classification. The Adam optimizer was used for its adaptive learning rate, aiding faster convergence. The CyclicLR scheduler dynamically varied the learning rate to avoid local minima.

E. Optimization Algorithm and Hyperparameters

The Adam optimizer was used with the following parameters:

- Learning rate: 0.00001
- Weight decay: 0.0005

The CyclicLR scheduler parameters were:

- Base learning rate: 0.00001
- Max learning rate: 0.0001
- Step size up: 50
- Step size down: 50

The hyperparameters used for training included:

- Batch size: 32
- Dropout rate: 0.4
- Number of epochs: 1200

F. Classification Test Accuracy and Evaluation Setup

The final classification test accuracy achieved by our model was 48.74%.

1) Evaluation Setup:

a) Procedure:

- The test dataset was loaded and preprocessed by resizing the images to 256x256 and normalizing them using mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225].
- The preprocessed images were fed into the trained CNN model to obtain predictions.
- The model's predictions were compared against the true labels of the test dataset to compute the accuracy.

G. Hardware and Environment

Used T4 GPU on Google Colab and departmental compute server

H. Training Time

The total training time for the network was approximately 10 hours, the model was trained for 1200 epochs to ensure convergence and optimal performance.

IV. Conclusion & Further work

In conclusion, the CNN achieved a test accuracy of 48.74% on the Flowers-102 dataset, indicating a moderate level of performance. The chosen architecture, with its convolutional layers, ELU activation, batch normalization, max pooling, dropout, and fully connected layers, proved effective but has room for improvement. Data augmentation and the cyclic learning rate scheduler enhanced generalization and training efficiency. While the model performed reasonably well, there were challenges in achieving higher accuracy, likely due to the complexity of the dataset and the need for more advanced techniques. For further work, exploring deeper architectures or using transfer learning from pre-trained models could improve performance. Additionally, fine-tuning hyperparameters and enhancing data augmentation strategies might yield better results.