Project Report: Flower Classification using Multi-Layer Perceptron (MLP)

Team Details

- **Team Members**:
    - Sanshrav Arora (ID: 2021A7PS2690P)

Table of Contents

---

1. Introduction

The task was to classify 60 categories of flowers using a dataset of 256x256 RGB images. Due to restrictions on using convolutional or recurrent neural networks, we designed a **Residual MLP Model** to tackle this classification problem. This architecture relies on multi-layer perceptrons (MLPs) and residual blocks with affine transformations to process image patches, making it suitable for image classification tasks where convolutional layers are not permitted.

We utilized token-mixing and feedforward residual connections to capture spatial and pattern-related information across patches of the input images. Positional embeddings were used to retain spatial information of the patches, while global average pooling helped in deriving final classification predictions.

2. Dataset Description

The dataset used for this task contains 3000 images for training and 600 images for validation. Each image has a resolution of 256x256 pixels and represents one of the 60 flower classes. The dataset is divided into:

- **Training Set**: 50 images per class (3000 total images)
- **Validation Set**: 10 images per class (600 total images)

3. Model Architecture and Design Choices

The model used in this project is a **Residual MLP Model**, carefully designed to meet the specific requirements of flower classification, and restricted to using a multi-layer perceptron (MLP) without convolutional layers. Below are the components of the model and the reasoning behind the choices made.

1. Patch Embedding

- **Description**: Images are split into patches of size 16x16, and these patches are flattened and linearly projected into a higher-dimensional space.
- **Reasoning**: Since convolutional layers were not allowed, splitting the images into patches enabled the model to handle high-resolution images by breaking them down into smaller, manageable segments. This approach mimics the effect of convolutional layers by allowing the model to process smaller portions of the image independently before projecting them into a higher-dimensional space. The linear projection step helps transform the flattened patches into a feature-rich representation that the model can process.

2. Positional Embeddings

- **Description**: A learnable positional embedding is added to retain spatial information of the patches, ensuring the model can track spatial relationships between patches.
- **Reasoning**: Since the model splits the image into patches, it loses the spatial relationships between patches (which are naturally preserved in convolutional networks). The positional embedding reintroduces this lost spatial information, allowing the model to distinguish between patches based on their positions in the image. This is essential for image classification tasks as spatial arrangement is critical in recognizing patterns and textures, especially in flowers.

3. Residual MLP Blocks

The core of the model consists of several **Residual MLP Blocks**. Each block includes the following components:

- **Learned Affine Transformations**:
    - **Description**: Scaling (alpha) and shifting (beta) parameters are applied to the input features via a learned affine transformation.
    - **Reasoning**: The affine transformation allows the model to scale and shift feature maps, effectively normalizing the features. This step helps the network learn faster by stabilizing the range of input values, making it easier for downstream layers to process the data.
- **Token Mixing**:

    - **Description**: This layer mixes information across patches by rearranging and linearly projecting the data.
    - **Reasoning**: Token mixing is used to enable interactions between different patches, capturing relationships across the image. This is crucial because the classification task depends not only on individual patch features but also on how they relate to each other spatially. Token mixing emulates the spatial context provided by convolutional layers, which is absent in a pure MLP.
- **Feedforward Networks**:
    - **Description**: Two fully connected layers with GELU activations, allowing complex feature learning.
    - **Reasoning**: Feedforward layers are essential for learning hierarchical feature representations. By adding non-linearity through GELU activation, the model can capture more complex patterns. The dropout mechanism is introduced here to regularize the network, reducing the chance of overfitting.
- **Residual Connections**:
    - **Description**: Both the token-mixing and feedforward layers have residual connections to enable efficient training and prevent degradation.
    - **Reasoning**: Residual connections help prevent the vanishing gradient problem and improve gradient flow during training. They ensure that earlier learned features are retained and not overwritten, which is particularly important for deep models. This also allows the network to learn more effectively by using identity shortcuts that bypass one or more layers.

4. Global Average Pooling

- **Description**: The outputs from all patches are averaged before classification.
- **Reasoning**: Global average pooling is applied to create a summary of the learned features from the patches. By averaging, the model focuses on the most prominent features, reducing the risk of overfitting by avoiding reliance on specific locations or features from individual patches. This approach is computationally efficient and effective for classification tasks.

5. Classification Head

- **Description**: A fully connected layer maps the pooled features to the 60 flower categories.
- **Reasoning**: The final fully connected layer serves as the classifier, mapping the feature representations from the global average pooling layer to the 60 flower categories. This standard setup ensures that the output is a probability distribution over the 60 classes, which is used for the classification decision.

Summary of Design Choices

The model was designed to balance simplicity and efficiency, with a focus on stability and performance. **Residual connections** were incorporated to prevent gradient degradation and enhance training stability, while **affine transformations** enabled better feature scaling and normalization throughout the network. The model's architecture is optimized to handle the complex flower classification task without the use of convolutional layers, leveraging token mixing, patch embeddings, and global average pooling to extract meaningful features from image patches and make accurate classifications.

Model Hyperparameters:

- **Image Size**: 256x256 pixels
- **Patch Size**: 16x16 pixels
- **Number of Residual MLP Blocks**: 5
- **Batch Size**: 32
- **Dimension of MLP**: 512
- **Dropout Rate**: 0.5
- **Weight Decay**: 0.0001
- **Optimizer**: Adam
- **Activation Function**: GELU
- **Learning Rate**: 0.001
- **Loss Function**: Categorical Cross-Entropy

4. Training Process

We trained the MLP using the following parameters:

- **Dropout (0.5)**: A dropout rate of 0.5 was applied in the fully connected layers to regularize the model. Dropout helps prevent overfitting by randomly deactivating neurons during training, ensuring the model does not rely too heavily on specific neurons.
- **Learning Rate (0.001)**: Different learning rates were experimented with to find the optimal value for faster convergence while avoiding overshooting the optimal solution. A learning rate of 0.001 was found to work well in conjunction with the Adam optimizer.
- **Weight Decay (0.0001)**: Weight decay (L2 regularization) with a value of 0.0001 was used to penalize large weights, which helps prevent overfitting by ensuring the model doesn't become too complex.
- **Batch Size (32)**: Various batch sizes were experimented with to balance memory usage and training speed. A batch size of 32 was selected as it provided stable training dynamics and convergence without causing memory overflow.
- **Model Depth (5)**: The depth of the model (number of residual MLP blocks) was another parameter that was experimented with. Different depths were tested to find the optimal balance between model complexity and performance. After experimentation, a depth of 5 residual MLP blocks was found to perform well, capturing the necessary complexity without overfitting.
- **Number of Epochs**: 100 (early stoping at epoch 54)
- **Early Stopping**: Early stopping was implemented to prevent overfitting by halting the training process if the validation loss did not improve for 10 consecutive epochs. This allowed the model to generalize better and saved training time.

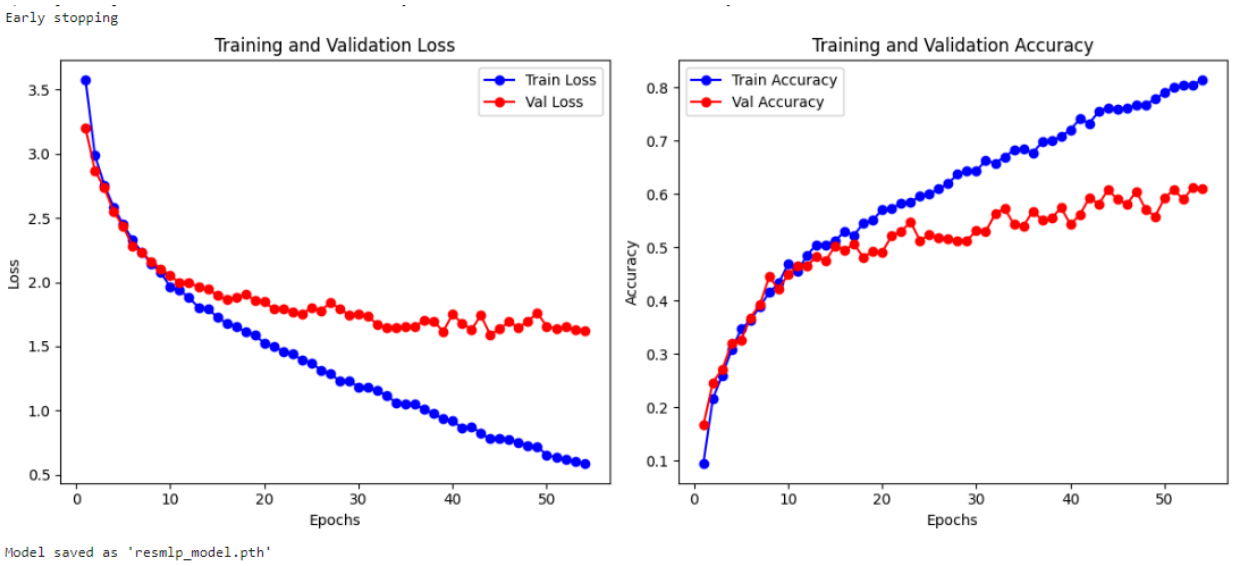
Preprocessing and Data Augmentation:

- All images were resized to 256x256 pixels.
- Images were normalized to have pixel values between 0 and 1.
- Data augmentation techniques were applied to enhance generalization by artificially increasing the dataset size and variability. These techniques included:
    - **RandomResizedCrop**: Images were randomly resized and cropped to 80-100% of their original size, ensuring they maintained the target resolution of 256x256 pixels.
    - **RandomRotation**: Images were randomly rotated by ±20 degrees to simulate different viewing angles and improve the robustness of the model to rotational variations.
    - **RandomHorizontalFlip**: A 50% probability of randomly flipping the images horizontally was applied to simulate different orientations.

- **Normalization**: After applying the above augmentations, images were normalized using the mean and standard deviation from ImageNet statistics. This step ensures that the pixel values are standardized, making training more efficient.
- **Color Jittering** was deliberately not applied in this project because the primary distinguishing features of the flowers are their colors and textures. Altering these features through color jittering could have distorted key information essential for accurate classification. By preserving the natural color distribution, the model was better able to focus on the important visual characteristics of the flowers, such as color patterns and textures, which are critical for distinguishing between different flower species.

These data augmentation techniques helped prevent overfitting and allowed the model to learn more generalized patterns.

Plots for training loss vs epochs and validation accuracy vs epochs are provided below.



5. Results and Analysis

The performance of the model was evaluated based on accuracy. The following are the results obtained:

- **Training Accuracy**: **81.50%**
- **Validation Accuracy**: **61.00%**

The model performed well on the training set, but the validation accuracy was slightly lower, indicating some overfitting. Regularization techniques such as dropout were explored to address this issue. While experimenting with values of Learning Rate, Weight Decay, Batch Size, Data Augmentation and Model Depth.

Evaluation Metric:

The evaluation metric used was **accuracy**, i.e., the number of correct predictions out of all predictions. This metric was tracked throughout the training process to monitor the model's performance on both the training and validation datasets.

Additionally, **categorical cross-entropy loss** was used as a measure of how well the model's predictions matched the true labels. The loss function played a key role in guiding the optimization process, with lower loss values indicating better performance. The training process aimed to minimize this loss while simultaneously maximizing the accuracy of predictions.

Both accuracy and loss were monitored during training and validation to ensure that the model was converging properly and not overfitting to the training data.

6. Conclusion

In this project, we successfully developed and implemented a **Residual MLP Model** for classifying 60 categories of flowers, overcoming the limitations imposed by the restriction of not using convolutional or recurrent architectures. By breaking the images into patches and leveraging token mixing, positional embeddings, and residual connections, the model was able to capture essential spatial and texture-related features for flower classification.

Key hyperparameters such as **learning rate**, **dropout**, **weight decay**, and **batch size** were fine-tuned through extensive experimentation, while **regularization techniques** like dropout and data augmentation were employed to reduce overfitting and improve generalization. Additionally, the depth of the model was carefully optimized to achieve a balance between complexity and performance.

The use of **categorical cross-entropy loss** and **accuracy** as evaluation metrics allowed us to track and measure the model's performance throughout the training process, ensuring effective learning and model stability.

Although this model performed well within the given constraints, there is potential for further improvement through more advanced augmentation techniques or deeper architectural exploration. The results demonstrate that even without the use of convolutional networks, a carefully designed MLP-based architecture can perform complex image classification tasks effectively.

7. Future Work

To improve the model, future work could focus on:

- **Ensemble of Classifiers**: To reduce overfitting and improve generalization, we plan to experiment with an **ensemble of classifiers**. This approach involves training multiple models (each with slightly different configurations or initializations) and combining their predictions through techniques such as **majority voting** or **weighted averaging**. Ensemble methods are known to reduce variance and can lead to more robust and accurate predictions by leveraging the strengths of different models.
- **Regularization Techniques**: Further experimentation with regularization techniques such as **label smoothing**, **early stopping with patience**, and **dropout rate optimization** could help reduce overfitting and improve the model’s performance.