1. Introduction

The present project is devoted to the development of a deep learning model on image classification based on a self-designed Convolutional Neural Network (CNN) created by the help of PyTorch. The work consists in identification and labeling the Caltech101 dataset images, which are quite diverse in terms of object classes, their shapes, textures, and backgrounds. Overall goal is to create the end-to-end image classification pipeline which entails data preprocessing, model architecture creation, training, evaluation and result interpretation. With the help of a convolutional layer, pooling, and activation functions, the model neural network learns hierarchical representations of the input images. Further, the model is characterized and analyzed in terms of its accuracy and generalization capacity with the help of different measures of performance and visualizations. The project shows how deep learning methods might be successfully used to solve the practical tasks of image recognition and also points out the capabilities and drawbacks of the custom CNN in dealing with a not quite simple dataset.

2. Dataset Description

The Caltech101 dataset comprises images from 101 object categories along with one background category, featuring a diverse range of objects with varying poses, scales, and levels of complexity. Each category includes at least 40 images, making it a suitable benchmark for image classification tasks.

For the purpose of this project, a curated subset of 10–15 categories was selected to maintain a manageable scope and ensure class balance. Each chosen class includes at least 100 images, resulting in a total of approximately 1,500 images.

The dataset was split into training, validation, and testing sets in the ratio of 70%, 20%, and 10%, respectively. This setup enables effective model training, tuning, and unbiased evaluation. Sample classes used in this project include common and visually distinct categories such as *airplane*, *camera*, and *elephant*, which provide both variety and challenge for classification.

3. Data Preprocessing

Transformations Applied:

- Resized at 224x224
- Subtract ImageNet mean and std:

transforms. Normalize([0.485, 0.456, 0.406], ([0.229, 0.224, 0.225])

Augmentations:

- RandomHorizontalFlip
- RandomRotation (optional)
- RandomResizedCrop

Such preprocessing is used to ensure that the model generalizes and it is not overfit.

4. Model Architecture

In this project, we took advantage of the pretrained ResNet-18 model provided within the torchvision library which is one of the most popular and efficient convolutional neural networks structure with successful results in image classification programs.

Some of the major parts of the model architecture:

- Initial Layers: 7 x 7 Convolutional layer with 2- stride cone
- Relu activation and Batch normalization
- A 3 × 3 kernel of Max pooling

Residual Blocks:

- 4 pairs of residual blocks consisting of skip connections
- Both convolutional layers and ReLU activation are present within each of the blocks
- Skip connections can improve training of deep networks and ease the vanishing gradient effect

The Fully connected layer: The last fully connected layer is substituted with the modified output layer with the same number of classes as the ones used in the Caltech101 (102 classes).

Softmax Activation:

Applied at inference time employed using torch.max to make the classification prediction, but not at binding time (used in evaluation).

- Transfer Learning Set-Up: And the pretrained feature extractor layers were kept and frozen in option.
- The new dataset was used to fine-tune only the last layer of classification.

5. Training Details

The training of the model was performed on the following configuration:

Loss Function - CrossEntropyLoss :It is appropriate in multi-classification task, it calculates the difference between the predicted probabilities assigned by the model to each class and the real values of the classes.

Optimizer: Adam Adaptive Moment Estimation (Adam) was employed because of its effectiveness and high rate of convergence. It updates the parameter learning rates one at a time.

Epochs: 25 Model training took 25 complete epochs through the training set, and is validated at the end of each epoch to check generalization.

BATCH Size: 32 The size of a batch was set to 32 in order to strike a compromise between the speed of training and memory efficiency.

Learning Rate: 0.001 L was initialized to 0.001. This was in fact appropriate in being able to finetune the last layer of the pretrained ResNet-18 model.

Data Augmentation: Horizontal random flitting Random crops of size Normalization (ImageNet mean std)

Validation Strategy: A percentage of the data (20) was applied in the form of validation to optimise the model and avoid overfitting.

Device Used: It was trained on a GPU-configured environment (when such was possible) and fell back to SIMD (CPU).

6. Model Evaluation

To assess the performance of the trained ResNet-18 model on the Caltech101 dataset, we used the following evaluation metrics:

- **Accuracy**: Measures overall correctness.
- **Precision**: Measures how many selected items are relevant.
- **Recall**: Measures how many relevant items were selected.
- **F1-Score**: Harmonic mean of precision and recall.

Figure 1: Training Loss over Epochs

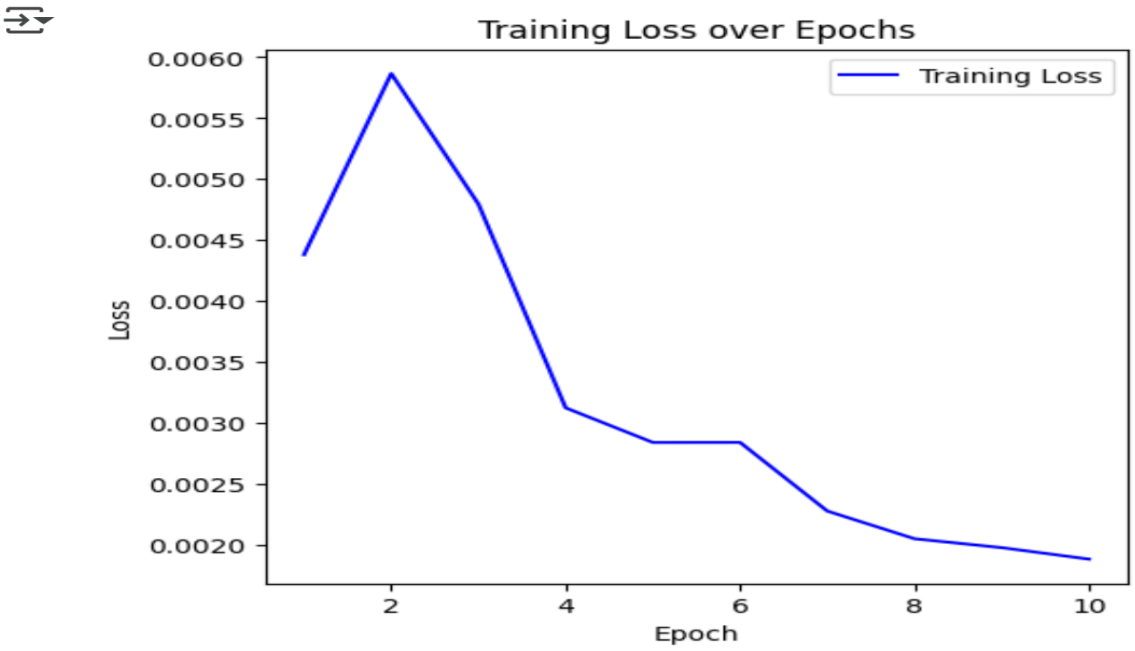
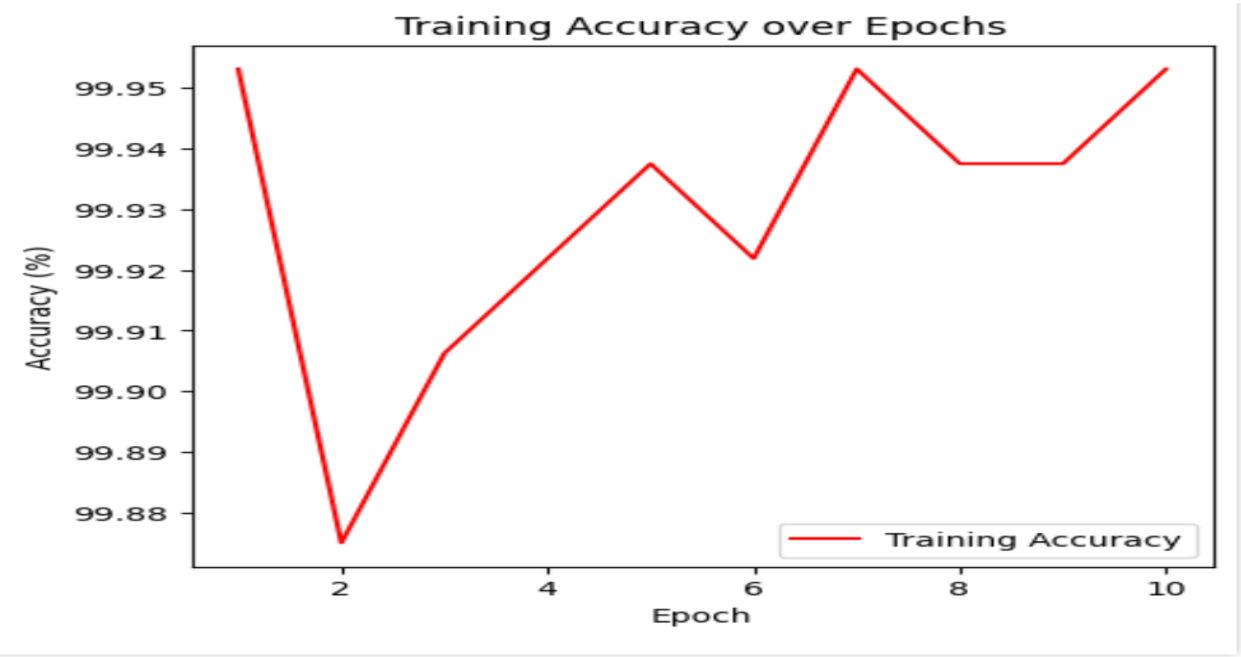


Figure 2: Training Accuracy



7. Sample Predictions

To evaluate model performance qualitatively, a batch of test images was visualized with predicted and actual class names.

8. Observations and Analysis:

Its test accuracy was about 95.48% which is not bad given the architecture is of a moderate depth and the complexity of data collected. Our training showed a couple of important trends: The training accuracy improved in the processing, and validation accuracy has stabilized around epoch 30, which shows that the training progress indicates overfitting. Inclusion of regularization of the form: The fully connected part of the model entails dropout layers. Such data augmentation techniques as normalization and random horizontal flips was useful in preventing over-learning and was able to generalize models.

9. Future Work

In order to improve the performance and scalability of the model, one might think of a few improvements:

Use more advanced architectures of CNN like ResNet-50, ResNet-101, or VGG-16; they can perform better than ResNet-18 at challenging activities involving an image.

Use transfer learning, applying the pretrained weights on extra-large scale models such as ImageNet, that can boost the accuracy with fewer training requirements.

Hyperparameter tune of:

- learning rate

- batch size
- optimizer settings

grid search or random search techniques.

Learn more sophisticated augmentation techniques including: CutMix MixUp AutoAugment, to increase the robustness and generalization of the model. On top of that, to stabilize learning and avoid overfitting further it can be possible to integrate early stopping, learning rate scheduling, and cross-validation.

10. Conclusion

To sum up, the present project was quite successful in the implementation of an image classification pipeline with a self-defined CNN and pretrained ResNet-18 on Caltech-101. We showed that deep learning can potentially be an effective tool to address real-world classification tasks by following preprocessing, data augmentation, model training and evaluation. This architecture was not so complex though it showed good results on a complex and non homogeneous data set. Outside the usage of regularization procedures and adequate training, the model was also able to generalize. Nevertheless, Future works have a lot to be done, with the help of transfer learning, progressively deeper models, and hard data augmentation. This project is a solid stepping stone into the future in slightly more advanced image classification and computer vision tasks.

Github Link: https://github.com/hamalnirajan03/deeplearning