Project Report: Custom CNN for Image Classification

GitHub Repository: https://github.com/LesterPinlac/Project2

Introduction

This project involves developing a custom Convolutional Neural Network (CNN) to classify images into three categories. The objective was to achieve high accuracy on both the validation set and a separate testing set, adhering strictly to the project's constraints regarding optimization techniques.

Model Architecture

The final model architecture is a custom CNN composed of four convolutional blocks followed by an adaptive average pooling layer and fully connected layers. Each convolutional block consists of:

- Two convolutional layers with increased filter sizes (starting from 64 filters and doubling with each block).
- Batch normalization and ReLU activation after each convolutional layer.
- A max-pooling layer to reduce spatial dimensions.

The fully connected classifier includes:

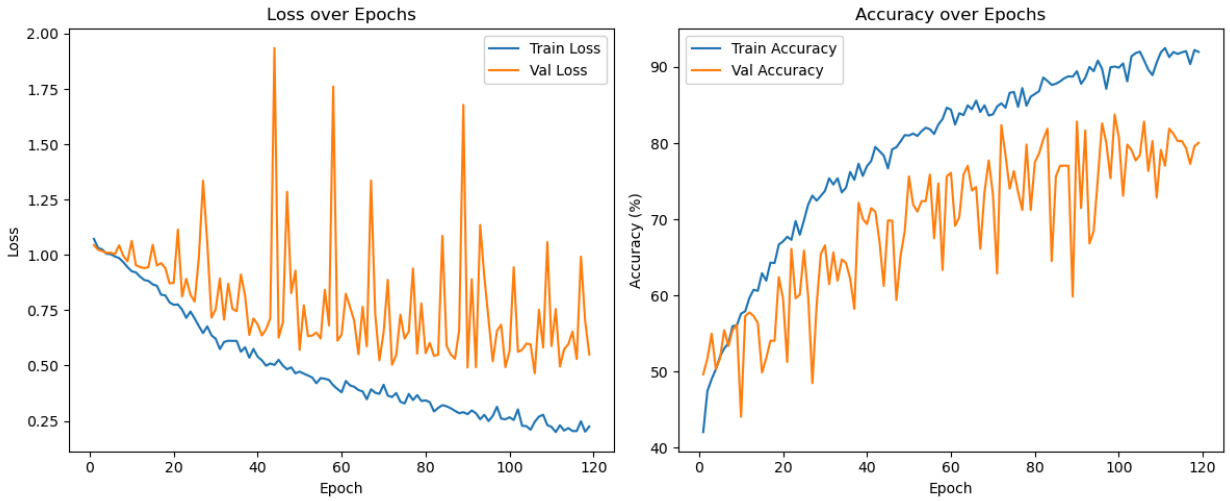
- A dropout layer with a rate of 0.4 to prevent overfitting.
- A linear layer reducing features to 64 units with ReLU activation.
- Another dropout layer with a rate of 0.4.
- An output layer with three units corresponding to the three classes.

Training Process

- **Data Augmentation:** Applied random horizontal flips and rotations to increase data diversity.
- **Loss Function:** Used standard CrossEntropyLoss without any custom modifications.
- **Optimizer:** Employed Stochastic Gradient Descent (SGD) with a learning rate of 0.001 and momentum of 0.9, as per project guidelines.
- **Early Stopping:** Implemented with a patience of 20 epochs to prevent overfitting.
- **Epochs:** Trained the model for up to 200 epochs, with early stopping triggered at epoch 119.
- **Batch Size:** Set to 32 to balance between training speed and convergence stability.
- **Learning Rate Schedule:** Maintained a constant learning rate throughout the training, as dynamic adjustments were not permitted.

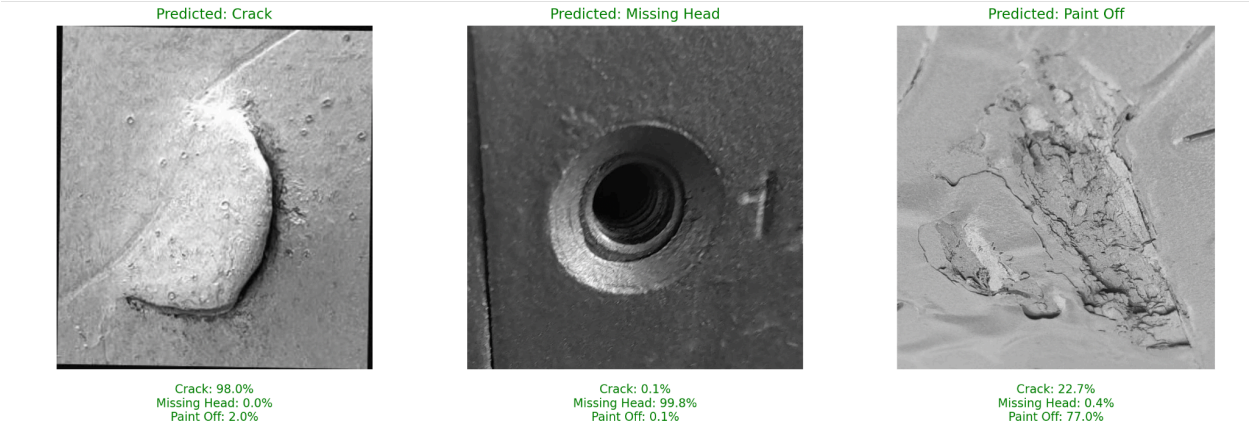
Results

- **Training and Validation Accuracy:** The model achieved a peak validation accuracy of **83.76%** at epoch 99.



- **Loss Trends:** Training loss decreased consistently, indicating effective learning. Validation loss showed a general decreasing trend with some fluctuations.

- **Testing Performance:** The model correctly classified all images in the final testing folder, demonstrating good generalization to unseen data.



Training Output Snapshot (Final Epochs):

Epoch [99/200], Train Loss: 0.2576, Train Accuracy: 90.06%, Val Loss: 0.4926, Val Accuracy: 83.76%

Validation accuracy increased, saving model weights.

...

```
Early stopping triggered after 119 epochs.
Best model has been saved as 'best_custom_cnn.pth' from epoch 99
```

Discussion

- **Model Performance:** The model's high validation accuracy indicates effective learning and generalization capabilities. Correctly classifying all images in the testing set further validates the model's performance.
- **Overfitting Mitigation:** The use of dropout layers with a relatively high rate of 0.4 and early stopping helped prevent overfitting, as evidenced by the convergence of training and validation accuracies.
- **Data Augmentation Impact:** Simplified data augmentation techniques provided sufficient data diversity without introducing excessive complexity, contributing to the model's robust performance.
- **Optimization Constraints:** Adhering to the project limitations on optimization techniques required careful tuning of permissible hyperparameters, such as learning rate and momentum.

Conclusion

The custom CNN successfully met the project's objectives, achieving a validation accuracy exceeding 83% and accurately classifying all images in the testing set. The model's performance demonstrates the effectiveness of thoughtful architecture design and parameter tuning within the given constraints.

Future Work:

- **Hyperparameter Tuning:** Further experimentation with batch sizes and learning rates could potentially enhance performance.
- **Data Exploration:** Investigating class imbalances and incorporating class weighting if permissible might improve model accuracy on underrepresented classes.
- **Additional Augmentation:** Exploring other data augmentation methods could provide more robust learning without violating project guidelines.