ChallengesToOvercome

November 23, 2024

3 Challenges to overcome

3.1 Computational time

During the training process, the model exhibited a consistent training time of approximately 70 seconds per epoch. This seems unexpected due to the relatively small size of the dataset, however, the utilization of data augmentation techniques significantly increased the training data's size. This is most likely the main contributer to the large training duration.

To address these performance bottlenecks, I employed caching and prefetching mechanisms to optimize image loading speed. Despite these optimizations, the overall training time remained moderate but, in my opinion, still in acceptable limits.

Challenge Analysis: While data augmentation is crucial to improving model generalization, it introduces a computational overhead as the model needs to process a larger volume of data. We are then faced with a trade-off between training time and model performance. I don't mind the 10 minute wait for my model to train if it will be a perfect fit, so I chose the model performance route.

Future Work: In the future, I would definetly plan to further explore methods for improving computational efficiency. One thing I could do would be to expierement with different batch sizes.

3.2 Overfitting

Initially my model had high training accuracy and low validation accuracy, this demonstrated tendency towards overfitting. This suggests, my model was memorizing the training examples instead of learning generalized patterns. This would result in poor performance on unseen data. The model was also fluctuating accuracy values significantly indicating the same problem.

Mitigation Strategies: To mitigate overfitting and improve the model's generalization capability, several techniques were eployed. These can be categorized broadly into techniques implemented proactively and those added in response to the observed problem.

Proactive Measures: 1. Data Augmentation - Artificial expansion of the dataset through the introduction of variations of existing training images. This exposes the model to a wider range of potential patterns. 2. Normilization - Pixel values were normalized using Z-score normilization. This ensured stable and efficient training by preventing features with larger values from dominating the learning process

Reactive Measures: 1. L2 Regularization - Introduced to penalize large weights within the model, encouraging it to learn simpler patterns and reducing complexity of the learned function. This helps prevent the model from fitting the noise in the training data. 2. Dropout - Randomly dropped neurons during training, preventing over-reliance 3. Early Stopping - Prevents the model from training for too long by stopping when the validation accuracy plateaus or starts to decrease

As a result, I succeeded in improving my validation accuracy, reducing overfitting and having more stabilized learning.

3.3 Low Performance

After applying L2 regularization and dropout I noticed a decrease in model performance/accuracy. To fix this I addressed the values used, I : - Lowered the regularizer from `0.01` to `0.001` - Decreased the dropout of the dense layers from `0.5` to `0.25` - Decreased the dropout of conv layers from `0.25` to `0.15`.

These adjustments successfully mitigated the performance drop and improved overall accuracy.

3.4 Undesirable Evaluation Metrics

Initially, the model exhibited high test accuracy but surprisingly low scores for evaluation metrics like precision, recall, and F1-score. These scores were all around `0.50` whill the test accuracy was `99%`. This discrepancy was perplexing and led to extensive troubleshooting. After investigating various potential solutions, the root cause was identified as an error in the code. Correcting this error resolved the problem and the evalution metrics were more aligned with the observed test accuracy.