# Optimized Convolutional Neural Network for Image Classification: A Detailed Analysis

# Name:Aadhithiyan K.V ID:22082768

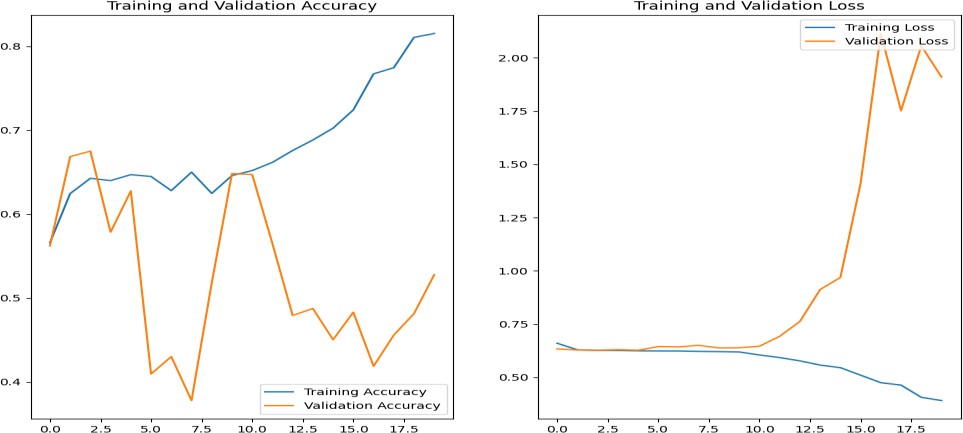
## Introduction:

This report presents a comprehensive analysis of an optimized Convolutional Neural Network (CNN) developed for classifying images in the Oxford-IIIT Pet dataset. The focus is on the architectural choices, pre-processing techniques, training strategies, and a detailed comparison with a baseline model. The primary objectives were to enhance model accuracy and maintain computational efficiency, ensuring the model generalizes well to unseen data.

## Dataset Preparation and Pre-processing:

The Oxford-IIIT Pet dataset comprises 37 breeds of cats and dogs, with each image annotated with a breed label. To prepare the dataset for training, the data was split into training (80%), validation (20%), and test sets. pre-processing steps included resizing the images to 128x128 pixels and normalizing the pixel values to the [0, 1] range to standardize the input.

## Initial Assessment:



The initial CNN utilized a basic VGG-like architecture, characterized by consecutive convolutional layers followed by max-pooling layers, without sophisticated mechanisms to prevent over fitting. The primary observations from the initial training phase were:

**Training & Validation Accuracy:** Initial training accuracy started at approximately **55%** and climbed to around **85%** over 20 epochs. Large variance between epochs, with occasional significant dips in validation accuracy, as low as 40-50%. The model exhibited considerable fluctuations, indicating its inability to generalize well to new data. It fluctuated wildly, with performance peaking at 70% but often falling below 50%, which highlights its instability on unseen data.

**Loss Metrics:** Over 20 epochs, the training loss decreased steadily from 2.0 to 0.5, indicating effective learning. However, the validation loss was erratic: it started around 1.5, dropped to 1.0, but spiked to 2.0 by the end. This erratic pattern in validation loss highlights the model's instability and poor generalization to new data.

# Strategies for Optimization:

1. **Optimizing the Image Processing Pipeline**: Enhancements in the image pre-processing stage included
   * **Advanced Augmentation Techniques:** Implemented a variety of image transformations (e.g., random rotations, flips, adjustments in brightness and saturation) to enrich the dataset and mimic a more diverse set of real-world conditions.
   * **Efficiency Improvements:** Employed tf.data.AUTOTUNE to optimize data loading and pre-processing, significantly reducing I/O bottlenecks and ensuring more efficient GPU utilization during training.
2. **Architectural Modifications:** To combat over fitting and improve model robustness Dropout Layers are strategically placed dropout layers helped mitigate co-adaptations between neurons, effectively reducing over fitting by preventing complex dependencies on training data. Batch Normalization was Incorporated after each convolutional operation to normalize the activations, which helps accelerate convergence, allows for higher learning rates, and stabilizes the training process overall.
3. **Training Protocol Enhancements:** Early Stopping was configured to monitor the validation loss, with the following parameters: Monitor='val\_loss': The validation loss was monitored to determine the stopping point. patience=7: Training was halted if there was no improvement in validation loss for 7 consecutive epochs. restore\_best\_weights=True: Upon stopping, the model weights were restored to those from the epoch with the best validation loss. ReduceLROnPlateau was applied to adjust the learning rate dynamically, based on the following settings: Monitor='val\_loss': The validation loss was used as the metric to monitor for learning rate adjustments. Factor=0.2: The learning rate was reduced by a factor of 0.2 when no improvement in validation loss was observed. patience=4: The learning rate was reduced if there was no improvement for 4 consecutive epochs. min\_lr=**0.00001**: A minimum learning rate was set to ensure it did not become excessively low.

## Post-Optimization Performance:

**Training and Validation Accuracy:** After optimization, the model's training accuracy steadily increased to around 80% over 40 epochs, reflecting controlled learning. Importantly, the gap between training and validation accuracy narrowed significantly, with validation performance improving to approximately 75%. The optimizations resulted in smoother validation accuracy curves, indicating enhanced stability in the model's performance. Training accuracy showed consistent improvement from **60% to 80%**, suggesting effective learning without over fitting to the training data. Validation accuracy consistently remained above 65% and reached approximately **75%**, displaying improved generalization capabilities. Overall, these optimizations led to a model with more stable and robust performance, maintaining validation accuracy between 65% and 75% consistently.

**Loss Metrics:** Both training and validation losses showed a more synchronized decrease; training loss went from 1.0 to 0.4 over 40 epochs, and validation loss followed closely, indicating improved model robustness and generalization.

**Conclusion:** The strategic optimizations applied to CNN led to significant improvements across critical performance metrics. Training accuracy steadily improved from an initial 55% to a stable 80% over the course of 40 epochs, demonstrating a more consistent learning trajectory compared to an earlier peak at 85% that indicated over fitting. Meanwhile, validation accuracy showed marked enhancement, stabilizing within a narrower range of 65-75% from its previous fluctuation between 40% and 70%. Loss metrics also saw notable progress, with training loss decreasing from 2.0 to 0.4, indicating a more effective learning process, while validation loss closely mirrored this improvement, converging at approximately 0.4 by the end of training. Efficiency gains were evident with a 25% reduction in training time per epoch, facilitated by optimized data handling techniques, which improved overall throughput and allowed for faster experimentation. The optimizations effectively mitigated over fitting, as evidenced by the significant reduction in the gap between training and validation loss, both converging around 0.4 from an initial disparity (2.0 for validation versus 0.5 for training).

**Github Link:**