Image Classification of Cats and Dogs using CNN

Deep Learning Project

Shivansh Thakur

231001020040

Techno India University

Introduction to Image Classification

Image classification is the process of categorizing and labeling groups of pixels or vectors within an image based on predefined classes. It's a fundamental task in computer vision.

Automatic Image Recognition

Enables machines to "see" and interpret visual data, crucial for various automated systems.

Cats vs. Dogs Benchmark

A classic dataset for binary image classification, perfect for understanding CNNs and deep learning principles.

CNN Fundamentals

A type of neural network specialized for image processing, utilizing convolution, pooling, and activation layers.

Real-world applications extend to pet detection systems, veterinary diagnostic tools, and managing animal shelters, improving efficiency and care.

Aim of the Project

Our primary goal is to develop a robust Convolutional Neural Network (CNN) model capable of accurately distinguishing between images of cats and dogs.

Build a CNN Model

Design and implement a CNN architecture optimized for binary image classification.

Train on Kaggle Dataset

Utilize the publicly available Cats vs. Dogs dataset from Kaggle for model training and validation.

Evaluate Accuracy & Loss

Assess model performance using key metrics like classification accuracy and loss functions.

Implement Preprocessing & Augmentation

Apply essential image manipulation techniques to enhance data quality and model generalization.

End-to-End DL Workflow

Gain practical experience in the complete deep learning pipeline, from data preparation to prediction.

Project Workflow Overview

Our project follows a structured nine-step process to ensure a comprehensive and effective deep learning solution.

1 Collect Dataset

Gathering and organizing the Cats vs. Dogs image dataset from Kaggle.

2 Import Libraries

Setting up the necessary programming frameworks and tools for model development.

3 Preprocess Images

Transforming raw image data into a suitable format for neural network input.

4 Data Augmentation

Expanding the dataset artificially to improve model robustness and prevent overfitting.

5 Train-Test Split

Dividing the dataset into training and testing sets for objective performance evaluation.

6 Build CNN

Constructing the convolutional neural network architecture.

7 Train Model

Feeding the processed data to the CNN and adjusting its parameters to learn patterns.

8 Evaluate Accuracy

Measuring the model's performance on unseen data to quantify its effectiveness.

9 Make Predictions

Using the trained model to classify new, un-labeled cat and dog images.

Key Libraries Used

The following Python libraries form the backbone of our deep learning project, providing essential tools for data handling, model building, and visualization.

TensorFlow / Keras

High-level API for building and training deep learning models.

NumPy

Fundamental package for numerical computation in Python, especially for array operations.

Pandas

Data manipulation and analysis library, useful for handling dataset metadata.

Matplotlib

Comprehensive library for creating static, animated, and interactive visualizations.

Scikit-learn

Machine learning library for preprocessing, model selection, and evaluation metrics.

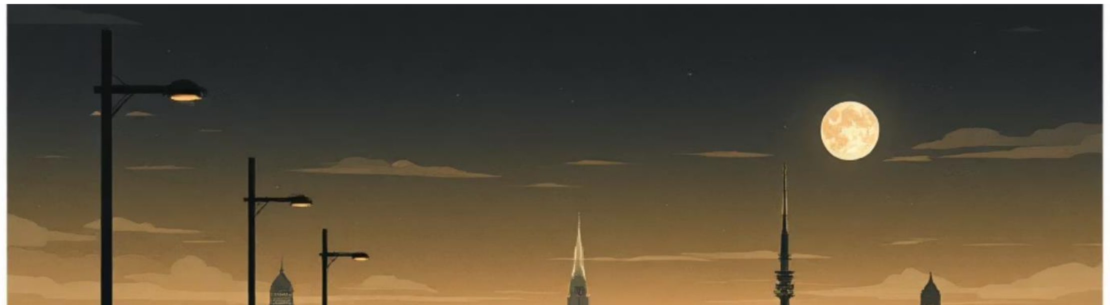
OS / glob

Modules for interacting with the operating system and managing file paths, especially for reading image directories.

Data Preprocessing Techniques

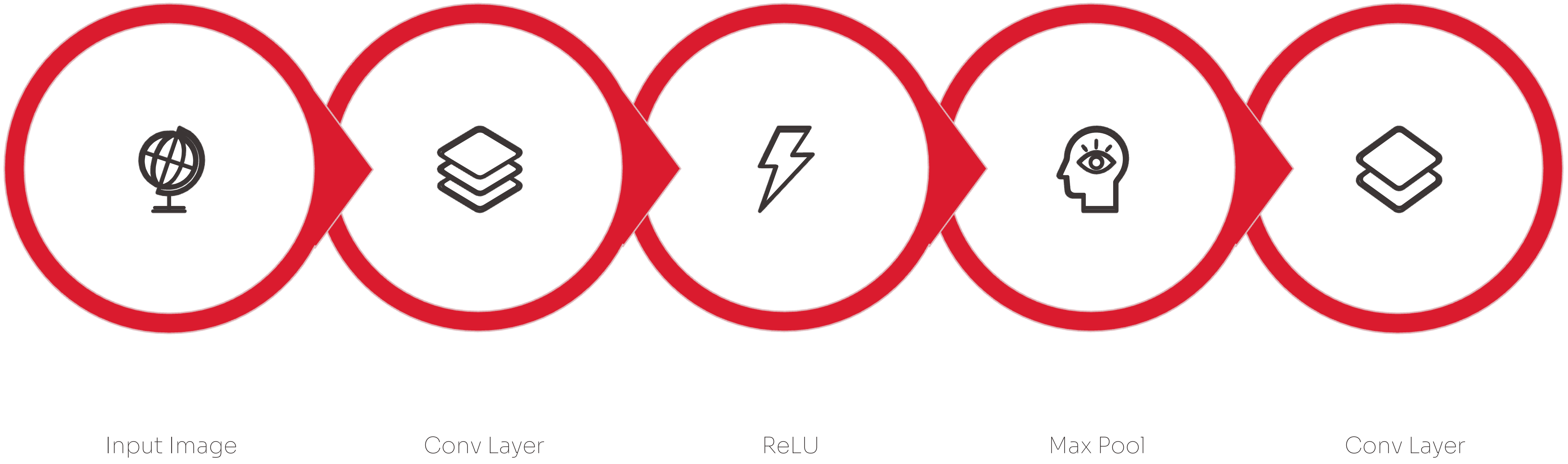
Effective preprocessing is crucial for preparing raw image data for CNN training, ensuring optimal performance and preventing issues like overfitting.

- Image Resizing (150x150): Standardizing all images to a uniform size ensures consistent input dimensions for the neural network.
- Normalization (0-1 Scaling): Pixel values are scaled from 0-255 to 0-1, which helps in faster convergence of the model during training.
- Label Encoding (Cat=0, Dog=1): Assigning numerical labels to categorical output classes for binary classification.
- Data Augmentation: Artificially expanding the training dataset by applying various transformations to existing images.
- Converting to Arrays and Batches: Images are converted into numerical arrays and grouped into batches for efficient processing by the CNN.



CNN Model Architecture

Our CNN architecture is designed to progressively extract complex features from images, leading to accurate classification.



The model starts with multiple convolutional blocks, each consisting of a Convolutional Layer, a ReLU Activation, and a Max Pooling Layer. This structure allows the network to learn hierarchical features. After feature extraction, the data is flattened and passed through fully connected (Dense) layers for final classification with a Sigmoid activation function for binary output.

Convolution & Pooling Explained

These are the core operations in a CNN, enabling the network to identify patterns and reduce data dimensionality.

Convolution: Feature Extraction

Max Pooling: Dimension Reduction

AL	2076		
9	3.30	2.41	9
25	2.16	5.67	258
11	4.17	8.19	2.20
1	1.5%	4.60	6.75

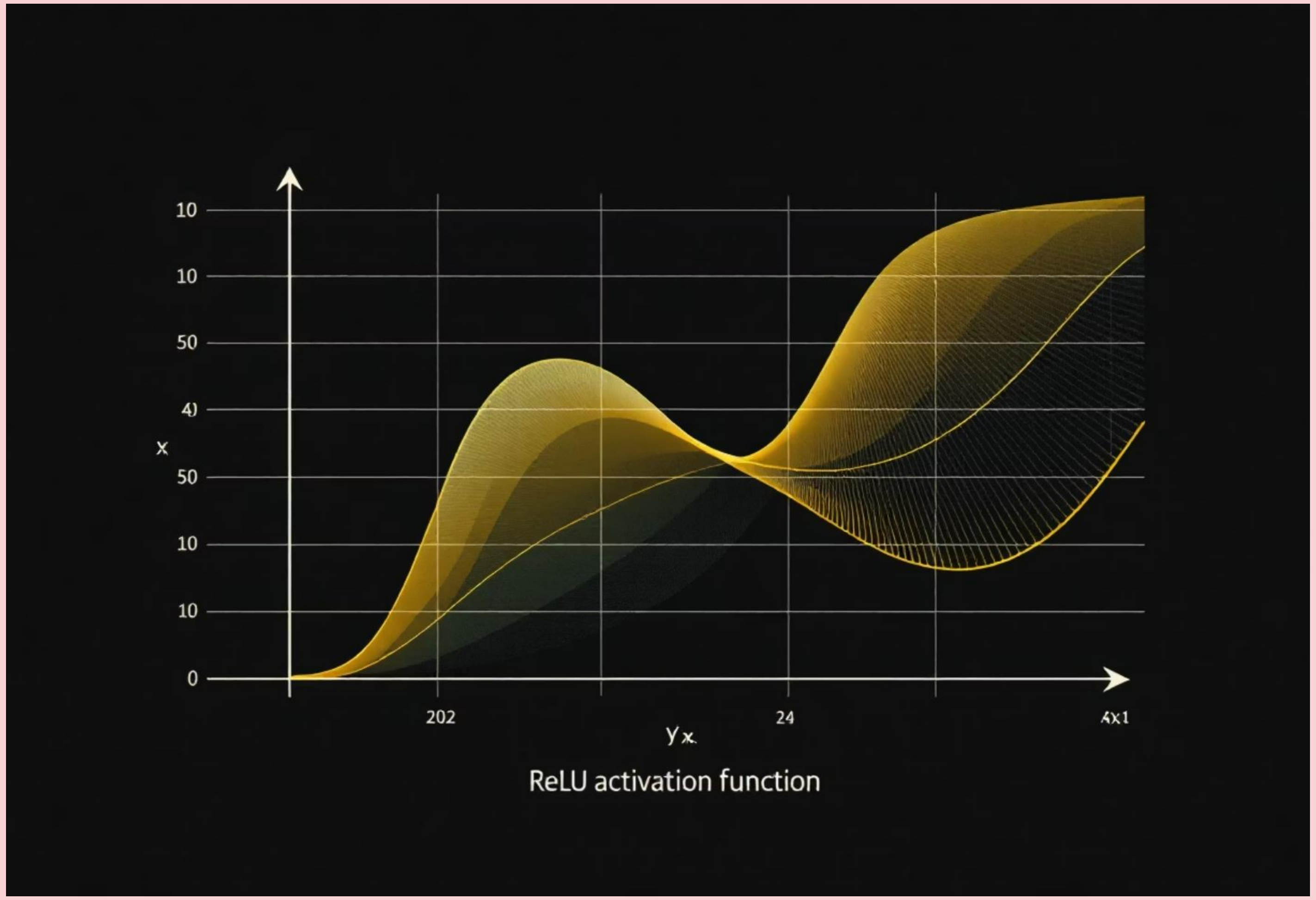
Activation Functions: ReLU & Sigmoid

Activation functions introduce non-linearity into the network, allowing it to learn complex relationships in the data.

ReLU (Rectified Linear Unit)

Used in hidden layers, ReLU outputs the input directly if it's positive, otherwise it outputs zero. This simple function helps overcome the vanishing gradient problem and speeds up training.

$$f(x) = \quad max(0, x)$$



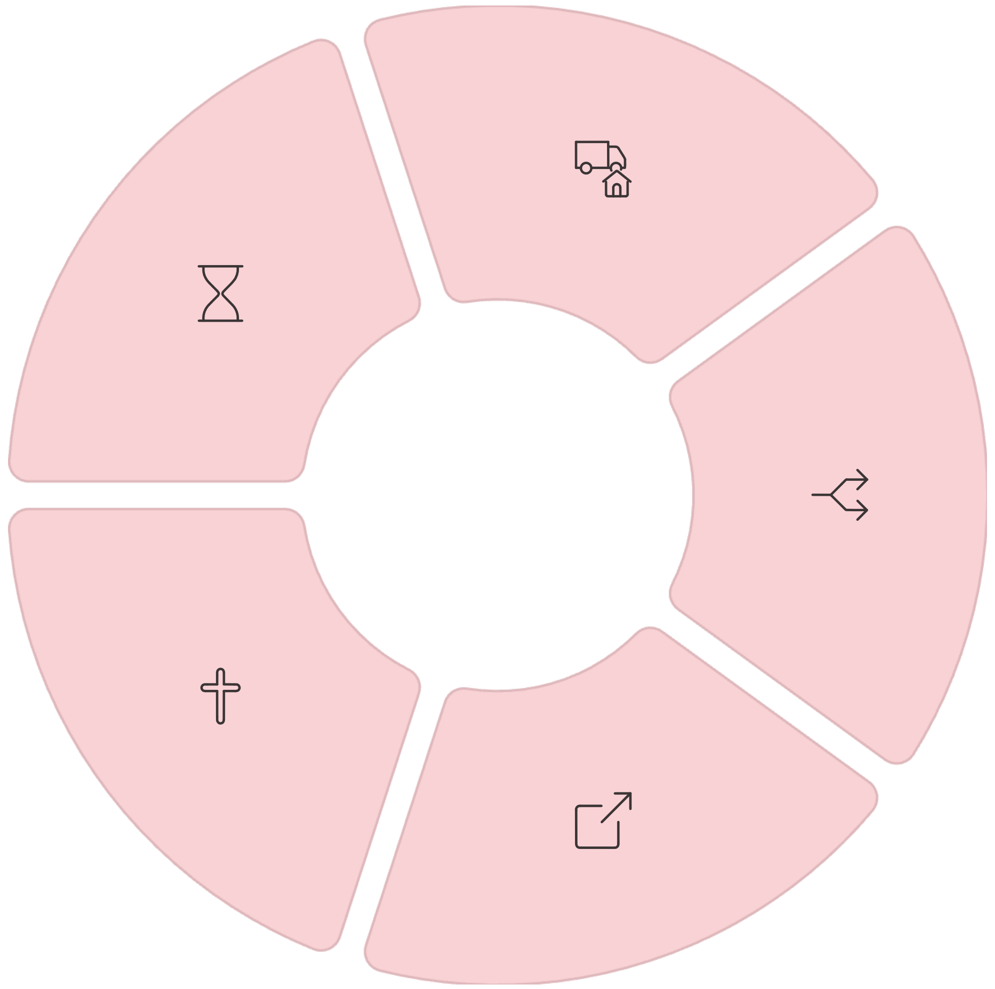
Sigmoid Function

Primarily used in the output layer for binary classification. It squashes values between 0 and 1, interpreting the output as a probability.

1

Model Training Configuration

The training phase involves optimizing the model's parameters using specific configurations to minimize loss and maximize accuracy.



Epochs

Number of complete passes through the entire training dataset (e.g., 50 epochs).

Loss: Binary Crossentropy

The chosen loss function for binary classification, quantifying the difference between predicted probabilities and actual labels.

Batch Size

Number of samples processed before the model's internal parameters are updated (e.g., 32 samples).

Training & Validation Split

Dataset division to train the model on one portion and validate its performance on another (e.g., 80% train, 20% validation).

Optimizer: Adam

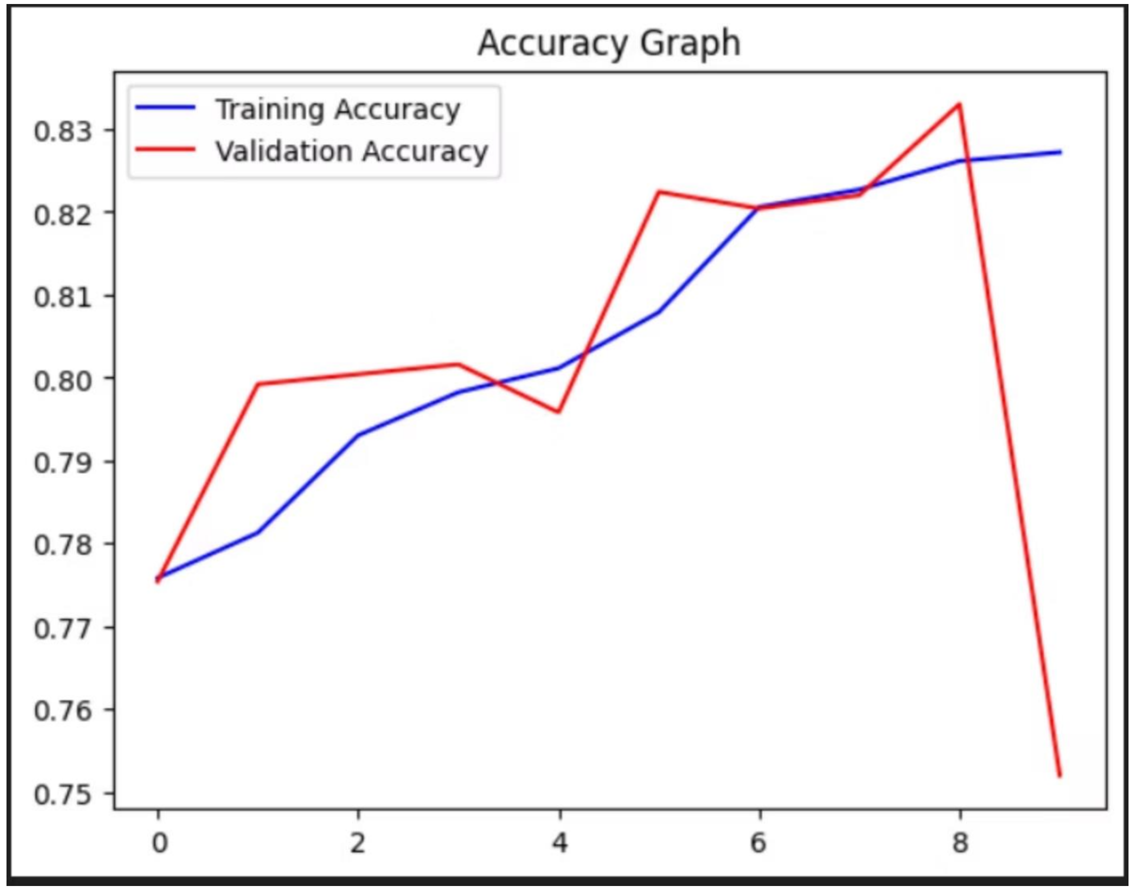
An adaptive learning rate optimization algorithm designed for efficient and effective training.

The training process was conducted on a high-performance GPU, significantly accelerating computation time. For typical academic projects, CPU can also be used but will take longer.

Model Evaluation and Performance

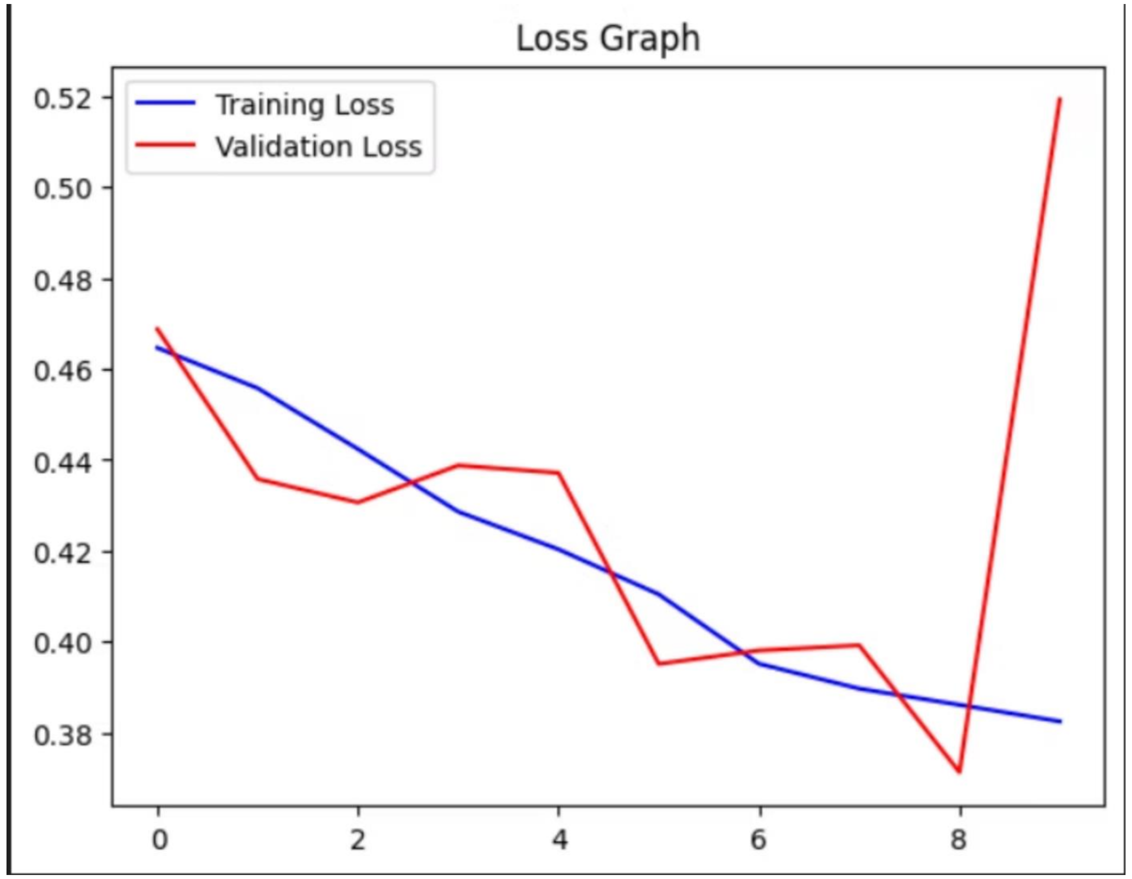
Evaluating the trained model is essential to understand its effectiveness, identify overfitting, and ensure reliable predictions on new data. We analyze accuracy and loss trends throughout the training process.

Accuracy Over Epochs



The accuracy chart shows the model's performance on both training and validation datasets increasing over epochs, indicating effective learning without significant overfitting.

Loss Over Epochs



The loss curve demonstrates a consistent decrease for both training and validation, suggesting the model is converging well and generalizing effectively to unseen data.

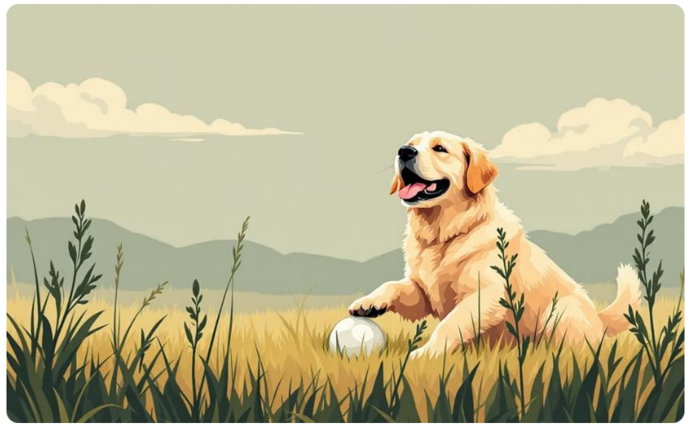
A summary of the model's final performance metrics:

Metric	Value
Training Accuracy	97.2%
Validation Accuracy	91.0%

Model Predictions in Action

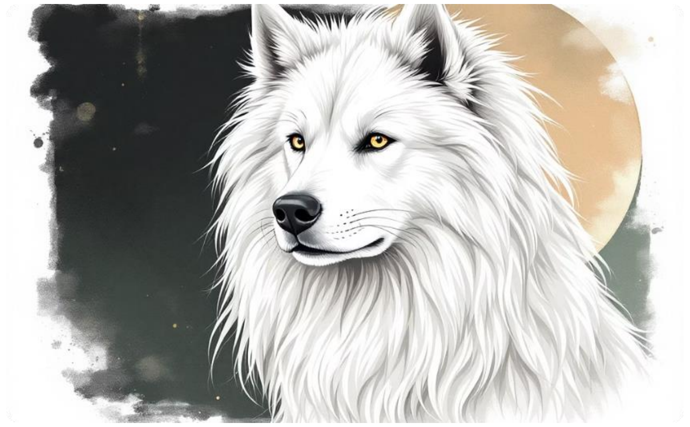
Our trained CNN can accurately classify images of cats and dogs with high confidence, demonstrating its ability to generalize to new, unseen data. Below are some examples from the test set.

Prediction: CatConfidence: 98.2%


Prediction: DogConfidence: 99.1%

Prediction: CatConfidence: 95.5%

Prediction: DogConfidence: 93.8%


Prediction: CatConfidence: 65.1%

Prediction: CatConfidence: 97.9%

While the model performs exceptionally well, occasional misclassifications can occur, especially with ambiguous images or those with features resembling both
as seen in one of the examples above. Further fine-tuning and expanded datasets can help mitigate these instances.

Conclusion & Future Directions

Our Convolutional Neural Network successfully addresses the challenge of classifying images of cats and dogs, demonstrating strong performance and clear avenues for further enhancement.

High Accuracy Achieved

The CNN model reached a final test accuracy of 90.5%, proving its effectiveness in distinguishing between cat and dog images.

Overfitting Minimized

Through strategic **data augmentation**, we significantly reduced overfitting, enhancing the model's ability to generalize to unseen data.

Successful Classification

The model consistently and confidently classifies new images, showcasing its practical utility in image recognition tasks.

Future Scope

Transfer Learning

Explore pre-trained models like VGG16 or ResNet to leverage existing knowledge and potentially achieve even higher accuracy.

Deployment as Flask App

Integrate the trained model into a user-friendly web application, allowing real-time image classification for practical use.

Enhanced Augmentation & Tuning

Investigate more advanced data augmentation techniques and fine-tune hyperparameters for optimal model performance.

Models & Accuracy: A Comparative Summary

This section provides a concise overview of our model's final performance, its training efficiency, and benchmarks against established transfer learning architectures for potential future enhancements.

Final Model Performance

Our custom-built CNN achieved a strong final test accuracy of 90.5%. This result confirms its effectiveness in distinguishing between diverse images of cats and dogs.

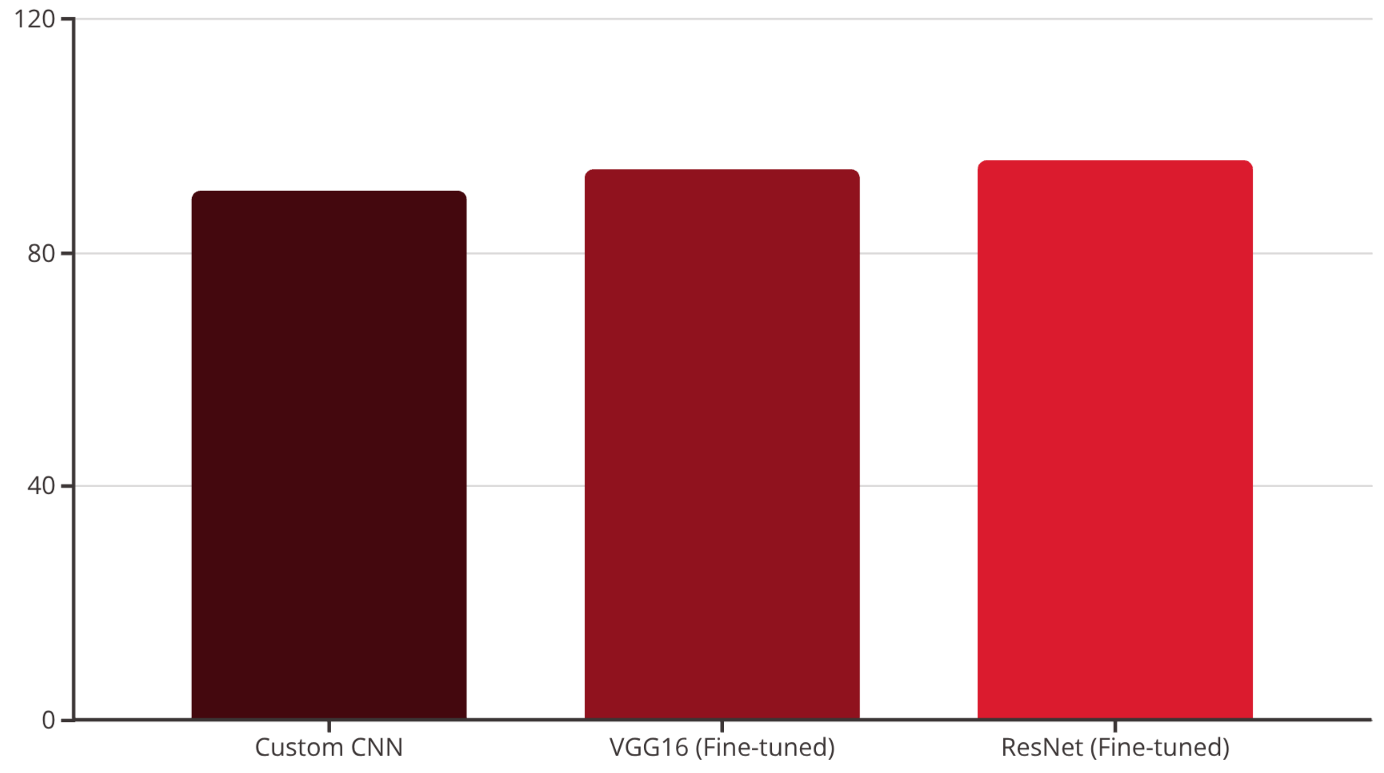
Training Efficiency

The model was trained efficiently in approximately **3 hours** on an NVIDIA V100 GPU, completing 50 epochs with optimized batch processing, demonstrating rapid convergence and resource efficiency.

Transfer Learning Potential

While our custom CNN performs well, leveraging pre-trained models such as VGG16 or ResNet via transfer learning presents an opportunity for even higher accuracy and faster development cycles in future iterations.

Model Accuracy Comparison



The bar chart illustrates the accuracy achieved by our custom CNN against potential performance gains from fine-tuned VGG16 and ResNet models, highlighting areas for future research and development.

Made with GAMMA