IMAGE CLASSIFICATION

1. Introduction

The task of this project is binary image classification, where the input is an image of either a dog or a cat, and the output is the predicted class label ("dog" or "cat"). The goal is to train a convolutional neural network (CNN) that can accurately distinguish between the two categories.

The dataset consists of labeled dog and cat images (balanced between the two classes). The images vary in size and resolution, so preprocessing steps are required before training.

2. Dataset

- Source: Kaggle: Cat VS Dog Dataset.
- Size: 24,961 images (50% cats, 50% dogs).
- Format: JPEG images with varying resolutions.
- Split: 70% training, 15% validation and 15% testing.

3. Pipeline

3.1 Preprocessing & Data Augmentation

To ensure consistency and improve generalization, the following preprocessing steps were applied:

- Data Cleaning:
    - Removed corrupted/error images that could not be loaded.
    - Filtered out images with height < 50 pixels or width < 50 pixels, since such small images provide too little information for classification.
    - Resulted in 24919 images
- Resize: All images resized to a fixed resolution (e.g., 224×224).
- Normalization: Pixel values scaled to [0,1] and standardized per channel.

- Random augmentations during training:
    - Random horizontal flip
    - Random crop

3.2 Model

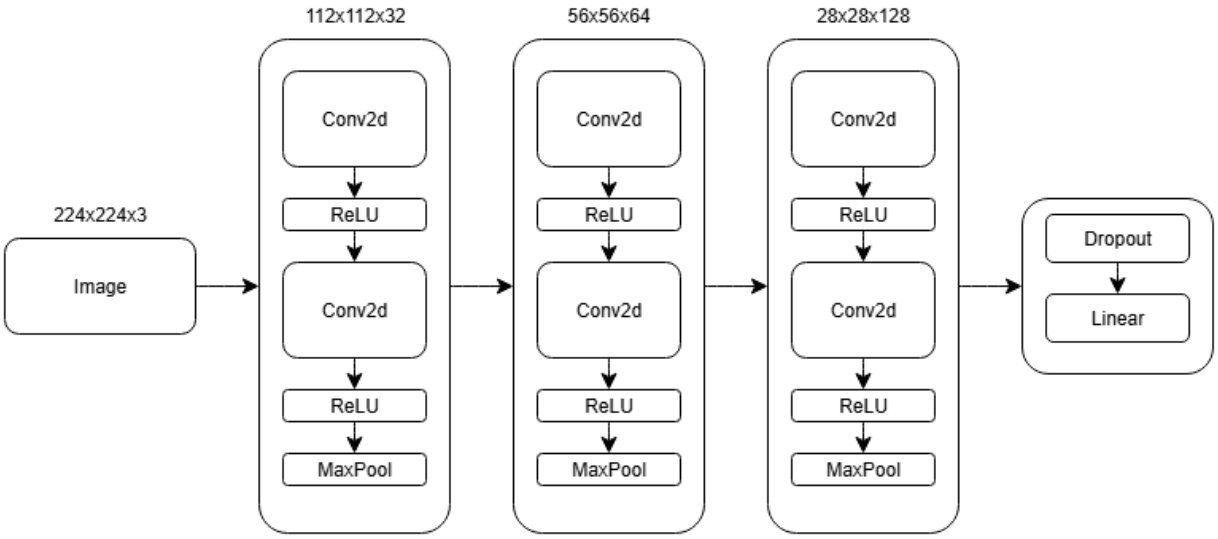

Figure 1: Model Architecture

The classification model was designed as a Convolutional Neural Network (CNN) with three convolutional blocks followed by fully connected layers for classification. The overall architecture is illustrated in Figure X (see diagram).

- Convolutional Blocks (Feature Extraction): Each block consists of two convolutional layers with ReLU activations, followed by a max-pooling layer.
    - Conv layers: extract spatial features such as edges, textures, and patterns.
    - ReLU activations: introduce non-linearity and speed up training.
    - MaxPooling: reduces the spatial dimensions, making the network more computationally efficient and robust to small translations.
    - After the three blocks, the feature maps are reduced from the input resolution (224×224×3) to a compact representation of size (28×28×128).
- Fully Connected Layers (Classification Head):

- The output feature maps are flattened and passed through a dropout layer to reduce overfitting.
- A fully connected layer with 128 neurons and ReLU activation learns high-level combinations of features, followed by the final linear layer with 2 outputs corresponding to the two classes (dog vs cat).
  A softmax activation is applied to obtain class probabilities.

This architecture provides a balance between model complexity and computational efficiency. It serves as a strong baseline for binary classification before experimenting with deeper or pre-trained networks (e.g., ResNet, VGG).

3.3 Experiment Setup

The model was trained on Google Colab using an NVIDIA T4 GPU. The training configuration was as follows:

- Loss Function: CrossEntropyLoss (suitable for multi-class classification).
- Optimizer: Adam optimizer.
- Learning Rate: 1×10^{-3} (0.001).
- Scheduler: StepLR scheduler for learning rate decay after fixed steps.
- Batch Size: 32.
- Epochs: Maximum of 100 epochs.
- Early Stopping: Monitored validation accuracy with patience = 5. Training stopped at epoch 21 when validation accuracy plateaued.

4. Results

The model was trained with a maximum of 100 epochs but early stopping halted training at epoch 21 when validation accuracy plateaued.

- Final Accuracy: The model achieved 80.18% accuracy on the validation set.
- Training Dynamics**:** Training and validation accuracy increased steadily, stabilizing around epoch 11. Validation accuracy did not improve further after epoch 21, and early stopping occurred at epoch 26.

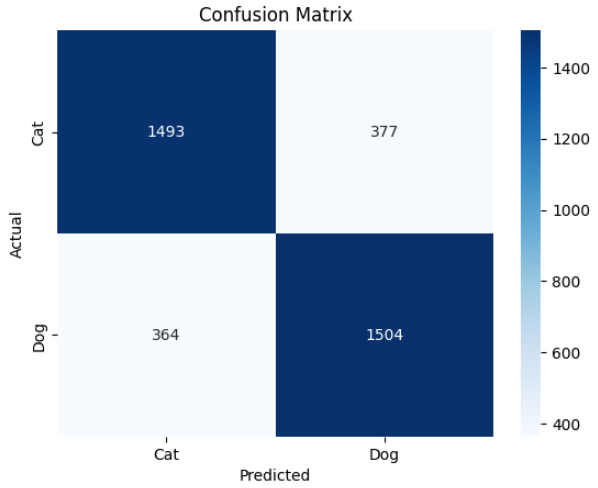

Figure 2: Confusion Matrix

The model correctly classified most cats (1493/1870) and dogs (1504/1868).

Misclassifications are balanced:

- 377 cats were predicted as dogs.
- 364 dogs were predicted as cats.

This suggests that the model does not strongly favor one class over the other, but struggles with visually similar cases (e.g., small dogs resembling cats or dark-colored cats resembling dogs).

5. Error Analysis

Figure 3: Misclassified Images

To better understand model limitations, I examined misclassified images (Figure 3). The confusion matrix shows that errors are balanced between cats and dogs, with roughly 20%

misclassification for each class. Closer inspection of individual samples reveals several common issues:

- Image A (black dog in snow): The dark dog blends into the snowy background, producing low-contrast features. The model may not have seen similar lighting conditions during training, causing confusion with cat-like silhouettes.
- Image B (white cat with green eye reflections): Strong glare and partial occlusion from the metal bar obscure key facial features, making it difficult for the network to capture discriminative cat features.
- Image C (kitten with occlusion): The kitten appears small in the frame, partially occluded by another animal's tail.

Since we applied Resize(256) followed by CenterCrop(224), much of the relevant information (e.g., face, ears) was reduced or cropped out, leading to loss of critical cues.

6. Improvements / Future Work

To improve performance beyond 79%, the following steps could be explored:

1. Deeper Models: Add more blocks to the model or use pre-trained CNNs (ResNet, VGG) for better feature extraction.
2. More Augmentation: Add rotations, brightness/contrast adjustments, and random erasing, use bigger resolution.
3. Hyperparameter Tuning: Experiment with different learning rates, batch sizes, and schedulers.
4. Regularization: Use weight decay or dropout more effectively.
5. Transfer Learning: Fine-tune a model pre-trained on ImageNet.