Cat vs. Cat Loaf Image Classification Using Convolutional Neural Networks

Farzana Chowdhury

***Abstract*—This paper investigates the classification of images into two distinct categories: "Cat" and "Cat Loaf," using Convolutional Neural Networks (CNNs). The project is inspired by the nuanced challenge of recognizing animal postures, specifically differentiating traditional cat poses from loaf-like compact positions. The dataset contains 646 RGB images, evenly split between the two classes. Images were preprocessed through resizing, normalization, and data augmentation to enhance generalization. The CNN architecture employed consists of convolutional, max-pooling, and dense layers, achieving a validation accuracy of approximately 66%. This project highlights the potential of CNNs in solving posture-based classification tasks, emphasizing the importance of feature extraction and augmentation in improving model performance. The study also discusses challenges such as subtle posture variations and dataset limitations, proposing future improvements to address these issues.**

I. Introduction

Image classification is one of the most fundamental tasks in computer vision, enabling machines to understand, categorize, and interpret visual data. The process involves assigning an image to one of several predefined categories by analyzing its visual patterns, textures, and features. Applications of image classification span a wide range of domains, including medical imaging [1], autonomous vehicles [2], and wildlife monitoring [4].

Traditional image classification tasks often involve clearly distinct object categories, such as differentiating between cats and dogs, cars and bicycles, or apples and oranges. In such cases, the classifier leverages high-level visual differences to distinguish one class from another. However, in certain scenarios, the challenge lies in detecting subtle differences within the same object class. This is particularly true for tasks requiring the identification of variations in posture, expression, or fine-grained details [3].

This project investigates such a nuanced classification challenge: distinguishing between images of cats and cats in a loaf-like posture, colloquially known as "cat loaves." Unlike traditional classification tasks, which rely on obvious differences between objects, this problem demands the model to learn subtle variations in body posture and spatial structure. For example, a cat loaf is characterized by a compact, tucked-in posture with legs hidden beneath the body, creating a loaf-like silhouette. In contrast, traditional cat postures often involve more visible legs and an extended body structure.

The "Cat vs. Cat Loaf" classification problem serves as a unique testbed for exploring the effectiveness of advanced feature extraction techniques in image classification. It demonstrates the capability of convolutional neural networks (CNNs) to learn hierarchical features, ranging from low-level edges and textures to high-level patterns, such as posture recognition [1]. Moreover, this challenge highlights the importance of leveraging preprocessing techniques and data augmentation [5] to create a robust model that can generalize well despite the inherent visual similarities between the two classes.

The significance of this study extends beyond its playful theme. Accurate recognition of animal postures has broader implications for fields like veterinary science, where posture analysis can help identify health issues in pets, and wildlife monitoring, where automated systems can classify animal behaviors in natural habitats [4]. This project also serves as a foundation for further research into nuanced classification tasks, showcasing how computer vision can handle complex and subtle distinctions.

By developing and evaluating a convolutional neural network for the "Cat vs. Cat Loaf" classification task, this project aims to contribute to the growing field of fine-grained image classification. It explores the potential of deep learning to handle subtle, posture-based distinctions, paving the way for applications in specialized computer vision problems. Through this study, the objective is not only to achieve high classification accuracy but also to understand the strengths and limitations of current techniques in addressing such nuanced challenges.

A. Inspiration for the Project

The inspiration for this project stems from a fascination with how computer vision models can perceive subtle visual distinctions. While cats in loaf postures are a lighthearted subject, the task has deeper implications for understanding pose estimation, animal behavior recognition, and specialized computer vision applications. Additionally, this project aligns with the broader goal of building models that can handle nuanced classification tasks, which are critical in fields such as medical imaging and wildlife monitoring.

The challenge of distinguishing between "Cat" and "Cat Loaf" images is both technically interesting and illustrative of the capability of CNNs to learn fine-grained features. This work serves as a foundation for understanding how neural networks can tackle similar problems in other domains.

II. Background: Image Classification and CNNs

A. Image Classification

Image classification involves assigning a label to an image based on its content. It is mathematically defined as:

$$f(I) \rightarrow y \quad \text{where } y \in \{C_1, C_2, \ldots, C_n\}$$

Here, f represents the classification function, I is the input image, and y is the predicted label. In this project, $n = 2$, with C_1 = Cat and C_2 = Cat Loaf. Unlike conventional classification tasks, this problem requires the model to identify subtle features that differentiate posture rather than object identity.

B. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning architectures specifically designed to process and analyze image data. They excel at automatically learning hierarchical features directly from raw pixel inputs, making them a cornerstone of modern computer vision tasks. CNNs are inspired by the biological processes of the visual cortex, where neurons are sensitive to specific regions of an image, enabling localized feature detection.

A CNN processes image data through a series of layers that extract increasingly complex features. The architecture typically consists of the following key components:

1) **Convolutional Layers**: Convolutional layers apply learnable filters (kernels) to the input image, sliding across spatial dimensions (height and width) to produce feature maps. These feature maps encode spatial information such as edges, textures, and shapes, which are crucial for understanding image content. The convolution operation is mathematically expressed as:

$$O(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i+m, j+n) \cdot K(m,n)$$

where I represents the input image, K is the kernel, and O is the output feature map. The kernel's size and stride determine how much the feature map is reduced, while the activation function (e.g., ReLU) introduces non-linearity, allowing the network to model complex patterns.

2) **Pooling Layers**: Pooling layers downsample the spatial dimensions of feature maps while retaining the most significant information. This operation reduces computational complexity, minimizes overfitting, and improves feature invariance to translations. Common pooling methods include:
   - **Max-Pooling**: Retains the maximum value in each pooling window, emphasizing strong activations.
   - **Average-Pooling**: Computes the average value within the pooling window, offering a smoother representation.

   Max-pooling is often preferred for its ability to highlight prominent features while discarding background noise.

3) **Fully Connected Layers**: After the convolutional and pooling layers extract spatial features, fully connected layers (dense layers) process this information to make predictions. These layers interpret the high-level features by learning complex decision boundaries. The final fully connected layer typically outputs class probabilities using a softmax (for multi-class classification) or sigmoid activation function (for binary classification).

C. Hierarchical Feature Learning

CNNs are particularly effective because of their ability to learn hierarchical features:

- **Low-Level Features**: The initial layers capture basic elements such as edges, corners, and textures.
- **Mid-Level Features**: Intermediate layers detect more complex patterns, such as shapes or motifs.
- **High-Level Features**: The deeper layers identify task-specific features, such as objects or postures, which are crucial for classification tasks.

This hierarchical approach enables CNNs to generalize well across various image classification tasks, including nuanced ones like the "Cat vs. Cat Loaf" problem.

D. Advantages of CNNs

CNNs offer several advantages over traditional machine learning techniques:

- **Automatic Feature Extraction**: Unlike traditional approaches requiring handcrafted features, CNNs learn features directly from data.
- **Parameter Sharing**: Convolutional layers reuse the same kernel weights across the input, significantly reducing the number of trainable parameters compared to fully connected networks.
- **Spatial Invariance**: Pooling layers make CNNs robust to minor translations, rotations, and distortions in the input image.
- **Scalability**: CNNs perform well on large-scale datasets and are highly adaptable to complex tasks like object detection and segmentation.

E. Relevance to This Project

In this study, CNNs are used to distinguish between subtle posture differences in "Cat" and "Cat Loaf" images. The convolutional layers extract posture-specific features, while pooling layers ensure these features are invariant to minor variations in pose or alignment. The fully connected layers interpret these features, producing a probability score for binary classification. By leveraging the hierarchical learning capabilities of CNNs, this project effectively addresses the challenges of posture-based image classification.

The convolution operation is key to feature extraction and is mathematically defined as:

$$O(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i+m, j+n) \cdot K(m,n)$$

where I is the input image, K is the kernel (filter), and O is the output feature map. Pooling layers, such as max-pooling, summarize feature maps by retaining the most significant values within pooling windows, improving feature representation.

III. Dataset Overview

A. Dataset Description

The "Cat vs. Cat Loaf" dataset consists of 646 RGB images, evenly divided into two classes:

- **Cat**: 323 images of cats in traditional poses.
- **Cat Loaf**: 323 images of cats in a loaf posture, characterized by a compact body with paws tucked underneath.

Images were resized to 100×100 pixels to standardize input dimensions for the CNN. This uniformity simplifies the learning process and ensures that the model focuses on feature extraction rather than data variability.

B. Relevance of the Dataset

This dataset presents a unique challenge in computer vision, as the classification task relies on detecting subtle differences in posture rather than entirely different objects. The problem highlights the importance of spatial feature extraction in neural networks and serves as a testbed for evaluating the robustness of CNN architectures in nuanced classification tasks.

IV. Methodology

A. Data Preprocessing

To prepare the dataset for training, the following preprocessing steps were applied:

- **Resizing and Normalization**: All images were resized to 100×100 pixels, and pixel values were scaled to the range $[0, 1]$ to ensure uniform input and faster model convergence.
- **Data Augmentation**: Random transformations, including rotation (up to $30°$), width and height shifts (up to 20%), and horizontal flips, were applied. These augmentations increased dataset diversity, reduced overfitting, and improved generalization.

B. Model Architecture

The CNN architecture consists of:

- Three convolutional layers with filter sizes of 32, 64, and 128, each followed by ReLU activation and max-pooling.
- A fully connected layer with 128 neurons and ReLU activation.
- A dropout layer with a rate of 0.5 to prevent overfitting.
- A final dense layer with sigmoid activation for binary classification.

C. Training and Validation

The dataset was split in an 80:20 ratio, with the majority reserved for training and a smaller subset for validation. The model was trained for 10 epochs using the Adam optimizer and binary cross-entropy loss. Batch size was set to 32, balancing training efficiency and memory requirements.

V. Experimentation and Evaluation

A. Evaluation Metrics

The model's performance was assessed using accuracy, precision, recall, and F1-score. These metrics provided a comprehensive understanding of the model's efficacy. A confusion matrix was used to visualize the distribution of predictions across the "Cat" and "Cat Loaf" classes, highlighting strengths and potential areas for improvement.

B. Experimentation Steps

The experimentation process involved:

1) Loading and preprocessing the dataset, including normalization and augmentation.
2) Training the CNN on the augmented dataset with an 80:20 split for training and validation.
3) Hyperparameter tuning to optimize learning rates, batch sizes, and network depth.
4) Monitoring performance across epochs to identify potential overfitting or underfitting.
5) Testing on unseen data to evaluate real-world applicability.

VI. Results and Discussion

After completing the training process, the model achieved a validation accuracy of 66.15% and a validation loss of 0.62. Figures 1 and 2 show the training and validation accuracy and loss curves.

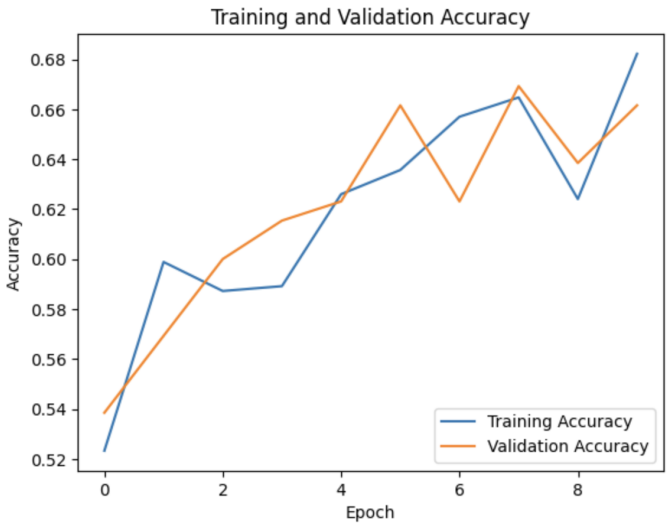

Fig. 1. Training and Validation Accuracy. The accuracy plot illustrates the model's learning progress over the epochs, showing improvement in both training and validation accuracy.

Fig. 2. Training and Validation Loss

The results demonstrate the model's ability to distinguish between the two classes, though subtle posture differences and a limited dataset size present challenges. Future improvements, such as increasing the dataset size or using transfer learning, could enhance performance.

VII. Concept to Code

This project's code is in google collab, and here is code includes:

- **Data Preprocessing**: Scripts for resizing, normalizing, augmenting, and splitting the dataset into training and validation sets.
- **Model Architecture**: A Python script defining the Convolutional Neural Network (CNN) model used for the classification task.
- **Training and Evaluation**: Scripts for training the model, visualizing performance metrics, and saving the trained model.
- **Execution Instructions**: A detailed text file outlining how to run the code step-by-step, including required dependencies and configurations.

A. How to Run the Code

To replicate the results and use the provided code, follow these steps:

1) **Set Up the Environment**:
   - Install Python (version 3.8 or later) and necessary libraries such as TensorFlow, NumPy, Matplotlib, and Pillow.
2) ** add the Dataset from kaggle**:
   - The dataset is available at the repository or via the Kaggle dataset link provided in the documentation.
3) **Run the Preprocessing Script**:
   - Execute the preprocessing script to resize images to 100×100 pixels, normalize pixel values, and split the data into training and validation sets.
4) **Train the Model**:
   - Use the script to train the CNN on the preprocessed dataset.
   - Training progress, including accuracy and loss metrics, will be logged and visualized during execution.
5) **Evaluate the Model**:
   - Evaluate the trained model on validation data using the following command:
   - This will output key performance metrics, such as validation accuracy, precision, recall, and F1-score, as well as a confusion matrix.
6) **Visualize Results**:
   - Visualizations of training and validation accuracy/loss curves can be generated by running the script.
7) **Save and Use the Model**:
   - The trained model is saved in `.h5` format for future inference.
   - Use this model to make predictions on new data by running the script.

B. Next Steps for Technical Implementation

While the current implementation achieves reasonable performance, future steps include:

- **Expanding the Dataset**: Adding more images to improve generalization.
- **Using Transfer Learning**: Employing pre-trained models to enhance feature extraction.
- **Incorporating Advanced Augmentations**: Simulating more diverse image variations to make the model more robust.
- **Fine-Tuning Hyperparameters**: Exploring optimal settings for learning rate, batch size, and architecture depth.

VIII. Conclusion

This study demonstrates the potential of CNNs for nuanced image classification tasks, such as posture recognition. The results underscore the importance of feature extraction and augmentation in achieving robust performance. Future work will focus on increasing dataset size, exploring transfer learning, and employing attention mechanisms to improve classification accuracy.

References

[1] Y. LeCun et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.
[2] A. Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks," *NIPS*, 2012.
[3] K. He et al., "Deep Residual Learning for Image Recognition," *CVPR*, 2016.
[4] R. Szeliski, "Computer Vision: Algorithms and Applications," Springer, 2011.
[5] T. DeVries, G. Taylor, "Dataset Augmentation in Feature Space," *ICLR*, 2017.