# Model Development and Optimization Report
# Dog vs Cat vs Bird

Abdul Lateef Abro - 29636

December 31, 2024

**Abstract**

This report covers the development and optimization of a Convolutional Neural Network (CNN) for image classification on a custom dataset. It includes dataset preprocessing, baseline model development, the application of optimization techniques, and transfer learning. Finally, the results, performance metrics, and insights are presented.

## 1 Dataset Preprocessing and Exploration

### 1.1 Dataset Overview

We are working with a dataset containing images categorized into three classes: `dog`, `cat`, and `bird`. The dataset consists of images of these animals, each image labeled accordingly.

### 1.2 Data Loading and Exploration

The dataset was loaded into a pandas DataFrame containing two columns: the 'filename' and its corresponding 'label'.

### 1.3 Label Encoding

The categorical string labels (`dog`, `cat`, `bird`) were converted into integer labels (0, 1, 2) using a dictionary-based mapping:

```
# Create a mapping from string labels to numerical indices
class_mapping = {'dog': 0, 'cat': 1, 'bird': 2}

#Reverse mapping for later use
reverse_mapping = {0: 'dog', 1: 'cat', 2: 'bird'}
```

### 1.4 Train-Test Split

The dataset was initially split into just 30% of the original, this is due to the limited resources at our disposal, however, to ensure its integrity the classes were equally preserved.

The dataset was then split into training and testing sets using an 80-20 split, ensuring that the model was trained on a subset of the data while being evaluated on unseen data.

## 2 Baseline Model Development

### 2.1 Model Architecture

The baseline model was implemented as a simple Convolutional Neural Network (CNN) with the following architecture:

- **Conv1**: 32 filters, kernel size of 3x3, stride 1, padding 1
- **Conv2**: 64 filters, kernel size of 3x3, stride 1, padding 1
- **MaxPooling**: 2x2 pooling
- **Fully Connected Layers**: 128 neurons and the final output layer with 3 units corresponding to the number of classes.

The CNN was designed to learn basic features like edges and textures and progressively capture more complex patterns.

### 2.2 Model Training

The model was trained using the `Adam` optimizer with a learning rate of 0.001 and the `CrossEntropyLoss` function. Training was performed for 30 epochs with a batch size of 32. During training, the loss and accuracy were tracked.

The final results from the baseline model were, sadly, un-found:

$$\text{Training Accuracy} = -$$

$$\text{Training Loss} = -$$

# 3 Optimization Techniques

## 3.1 Regularization Techniques

To enhance generalization and avoid overfitting, we applied the following optimization techniques:

- **Data Normalization**: The pixel values were normalized using mean and standard deviation values from the ImageNet dataset (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]). This step ensures that the input data has a consistent scale and helps the model converge faster.
- **Learning Rate Scheduling**: A learning rate scheduler was applied to decrease the learning rate by a factor of 0.1 every 3 epochs.

## 3.2 Model Performance Metrics

We evaluated the model using accuracy, precision, recall, and F1-score. Sadly, the baseline model could not be evaluated in time, but the final classification report from the baseline CNN model was as follows:

```
---------------------------------
Evaluating Train data accuracy
---------------------------------
              precision    recall  f1-score   support

         dog       1.00      1.00      1.00        96
         cat       1.00      1.00      1.00        96
        bird       1.00      1.00      1.00        96

    accuracy                           1.00       288
   macro avg       1.00      1.00      1.00       288
```

```
weighted avg       1.00      1.00      1.00       288

----------------------------------
Evaluating Test data accuracy
----------------------------------
```

# 4 Transfer Learning

## 4.1 Transfer Learning Overview

Transfer learning leverages pre-trained models, which have learned useful features from a large dataset like ImageNet, and fine-tunes them on our specific dataset. For this task, we used `ResNet18`, a pre-trained residual network, and replaced its final fully connected layer to accommodate the three output classes.

## 4.2 Implementation of Transfer Learning

The pre-trained ResNet18 model was modified by replacing its last layer to match the number of classes. We then fine-tuned only the newly added fully connected layer while keeping the pre-trained layers frozen.

## 4.3 Transfer Learning Training

The transfer learning model was trained with the same parameters as the baseline CNN model but with the new fully connected layers fine-tuned on the dataset.

# 5 Optimization of Transfer Learning

## 5.1 Learning Rate and Regularization

To improve performance, the following optimizations were applied to the transfer learning model:

- **Learning Rate Fine-Tuning**: A learning rate of 0.0001 was used for pre-trained layers, and 0.001 for the new fully connected layers.
- **Batch Normalization**: Batch normalization layers were added after each convolutional layer to stabilize and speed up training.

## 5.2 Model Training and Evaluation

After applying these optimizations, the transfer learning model showed a significant improvement over the baseline CNN. The classification report for the transfer learning model was as follows:

```
---------------------------------
Evaluating Train data accuracy
---------------------------------
Accuracy: 1.0000
Recall: 1.0000
F1-Score: 1.0000
Classification Report:
              precision    recall  f1-score   support

        bird       1.00      1.00      1.00        96
         cat       1.00      1.00      1.00        96
         dog       1.00      1.00      1.00        96

    accuracy                           1.00       288
   macro avg       1.00      1.00      1.00       288
weighted avg       1.00      1.00      1.00       288

---------------------------------
Evaluating Test data accuracy
---------------------------------
```

```
Accuracy: 0.8152
Recall: 0.8152
F1-Score: 0.8144
Classification Report:
              precision    recall  f1-score   support

        bird       0.80      0.77      0.79       864
         cat       0.77      0.76      0.76       864
         dog       0.87      0.91      0.89       864

    accuracy                           0.82      2592
   macro avg       0.81      0.82      0.81      2592
weighted avg       0.81      0.82      0.81      2592
```

# 6 Results, Metrics, and Insights

## 6.1 Results

The baseline CNN model achieved an accuracy of ?%, with a unkown performance across classes. On the other hand, the transfer learning model achieved an accuracy of 81.5%, showcasing the advantage of using pre-trained networks for feature extraction.

## 6.2 Metrics

The key performance metrics for both models are summarized as follows:

- **Baseline CNN**: Accuracy = ?%, Precision = ?, Recall = ?, F1-Score = ?.
- **Transfer Learning (ResNet18)**: Accuracy = 81.5%, Precision = 0.81, Recall = 0.81, F1-Score = 0.81.

## 6.3 Insights

- **Transfer Learning**: The transfer learning model performed better, particularly in classifying `bird` and `dog` images.

- **Optimization Techniques**: Techniques like batch normalization improved generalization and reduced overfitting.
- **Future Work**: Further improvements could be made by experimenting with more complex architectures, such as VGG16 or ResNet50, or by combining models through ensembling methods.

Link to Python Notebook: `https://colab.research.google.com/drive/1O-8tVISVR29cZ-bqgnvmU4pDK9T9lgFr?usp=sharing`