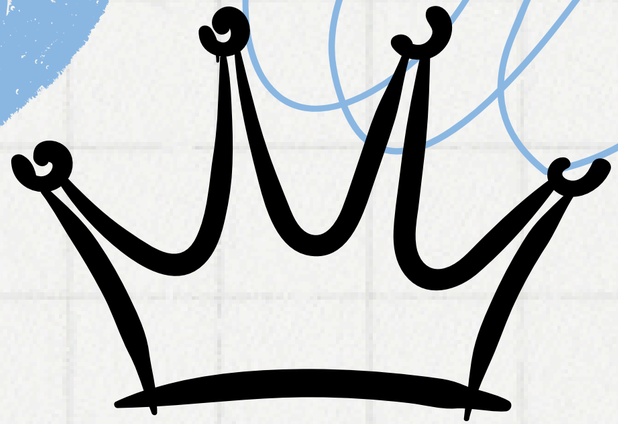
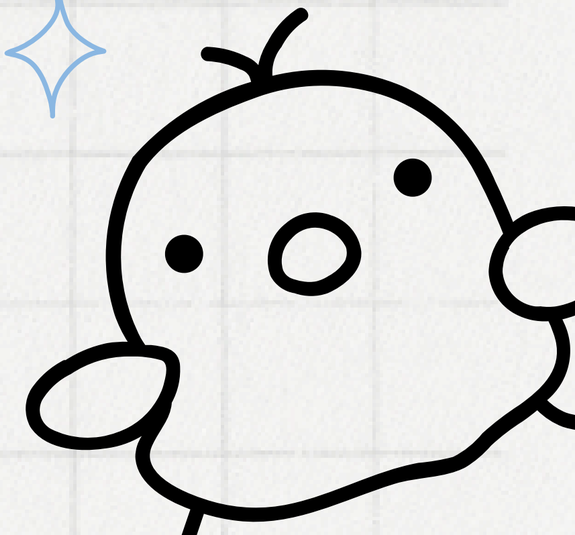


Doodle Classification

Presented by Aryan and Manpreet



Abstract

This project involves developing a deep learning model using Convolutional Neural Networks (CNNs) for automated image classification. The dataset was preprocessed through resizing, normalization, and data augmentation to improve model performance and reduce overfitting.

The CNN architecture includes convolutional, pooling, and dense layers, achieving 96.93% accuracy on test data.

Key challenges like overfitting and class imbalance were addressed, demonstrating the model's robustness and potential for real-world applications. Future improvements could include transfer learning and larger datasets to further enhance accuracy and generalization.

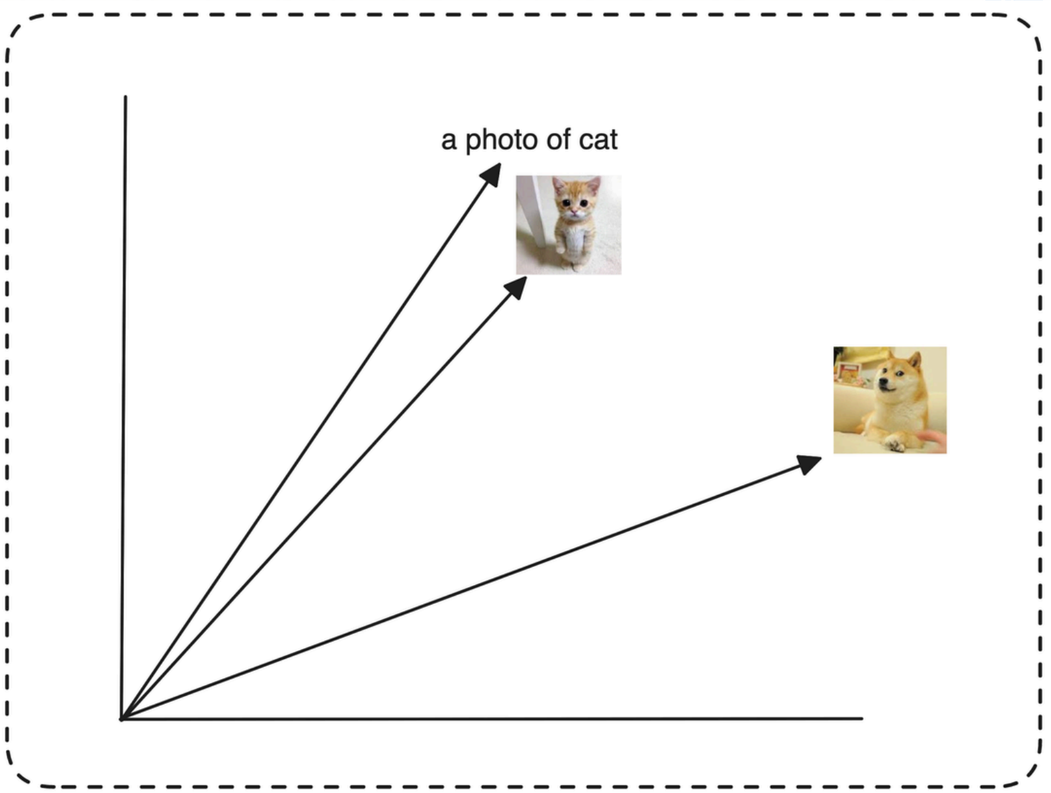
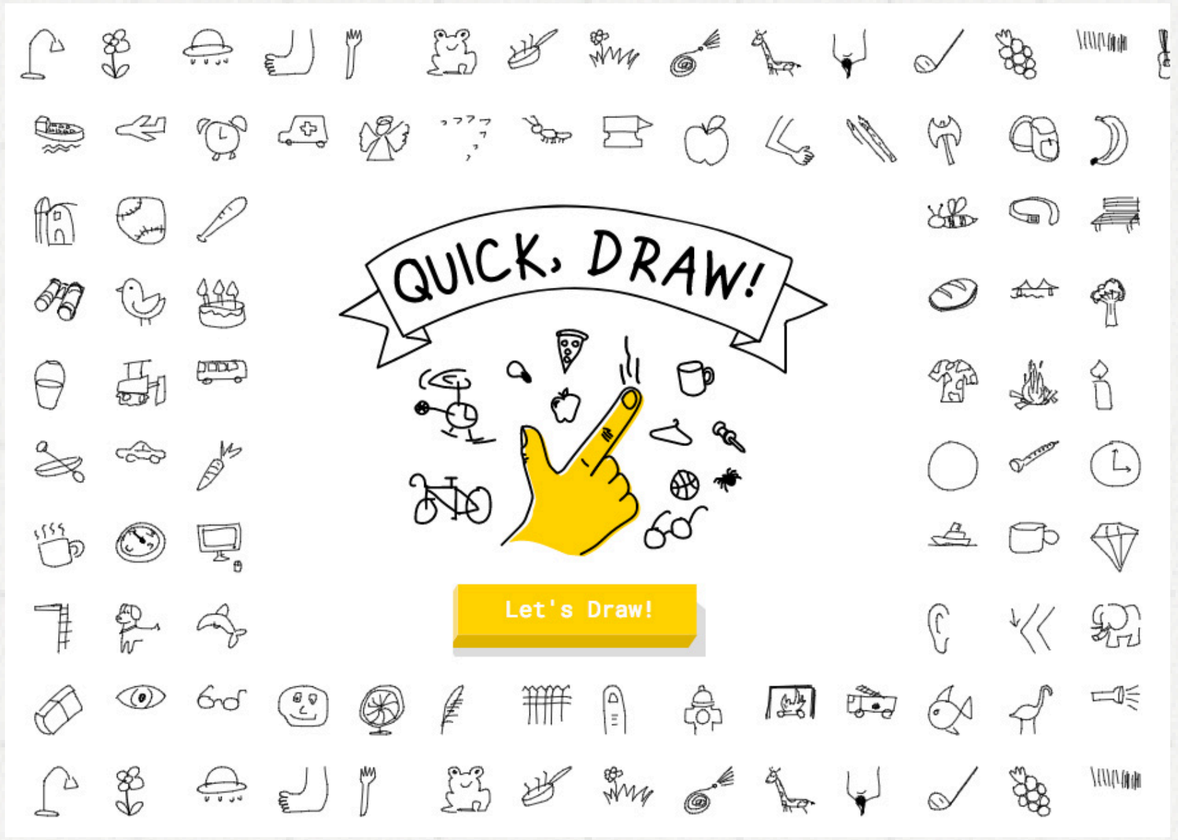


Image of the "cat" is nearer to it's relevant description in the Vector space

Dataset Description

The Quick Draw dataset is a collection of 50 million doodle drawings of 300+ categories. The drawings drawn by the players were captured as 28 x 28 grayscale images in .npy (numpy array) format with respect to each category.

The complete dataset is huge (~73GB) and so we have used only a subset of the complete data (6 categories).



Methodology

01

Data Loading and Preprocessing

02

Model Architecture

03

Compilation and Training

04

Evaluation and Visualization

05

Prediction

Dataset and Preprocessing

- **Dataset Details –** The dataset used in this project consists of images labeled into distinct categories. Each image has been tagged to aid in supervised learning.
- **Dataset Preprocessing –** Preprocessing steps included resizing images, converting them to grayscale or RGB if necessary, and normalizing pixel values to make the model converge faster.
- **Data Augmentation –** To prevent overfitting and enhance generalization, data augmentation techniques like rotation, flipping, and scaling were applied to create additional training samples from existing images.

Model Architecture

Introduction to CNN - Convolutional Neural Networks (CNNs) are deep learning models particularly suited for visual tasks. They can automatically learn patterns and features from images, such as edges, textures, and shapes.

Model Layers - The model consists of several layers:

- Convolutional Layers to extract features
- Pooling Layers to reduce dimensionality and computation
- Dense Layers for classification tasks
- Dropout Layers to prevent overfitting.

Layer Details – Our model uses ***ReLU*** (Rectified Linear Activation) for non-linearity and Softmax in the output layer to provide class probabilities.

Advantages of ReLu

ReLU, short for Rectified Linear Unit, is one of the most popular activation functions in deep learning, particularly for Convolutional Neural Networks (CNNs).

ReLU is widely preferred due to its simplicity, computational efficiency, and ability to avoid vanishing gradients in deep networks, leading to faster and more effective training.

01. Computational Efficiency

02. Sparsity

03. Avoiding the vanishing gradient problems

04. Biological Plausibility

Training the Model

- **Model Compilation –** The model was compiled with categorical cross-entropy as the loss function, Adam as the optimizer, and accuracy as the performance metric.

- **Training Strategy –** We trained the model over multiple epochs, using a validation split to monitor overfitting and adjust the training process.

- **Hyperparameter Tuning –** Several hyperparameters, such as learning rate, batch size, and epoch count, were fine-tuned to optimize model performance.

Epochs & Loss

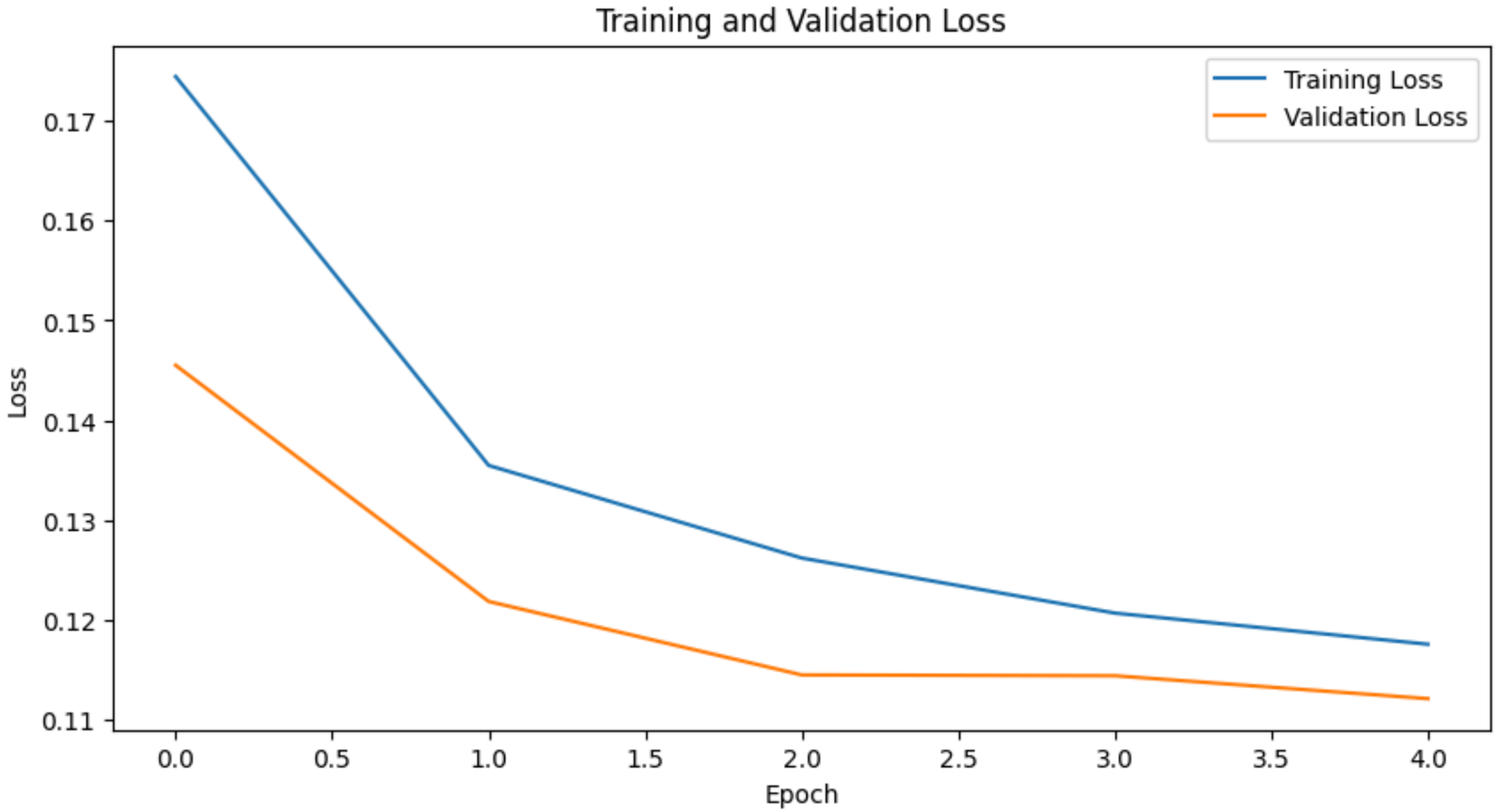
In deep learning, an **epoch** is a complete pass through the entire training dataset by the model. During each epoch, the model sees every data point in the training set once, adjusts its parameters (weights and biases), and moves a step closer to minimizing the error.

- Loss is a measure of how well the model’s predictions match the actual data labels. Lower loss values indicate that the model is performing better.
- At the beginning of training, the loss is usually high, as the model’s parameters are randomly initialized and have not yet been optimized.
- As training progresses over multiple epochs, the model's parameters are adjusted through backpropagation, which minimizes the loss. The loss typically decreases over epochs if the model is learning effectively.

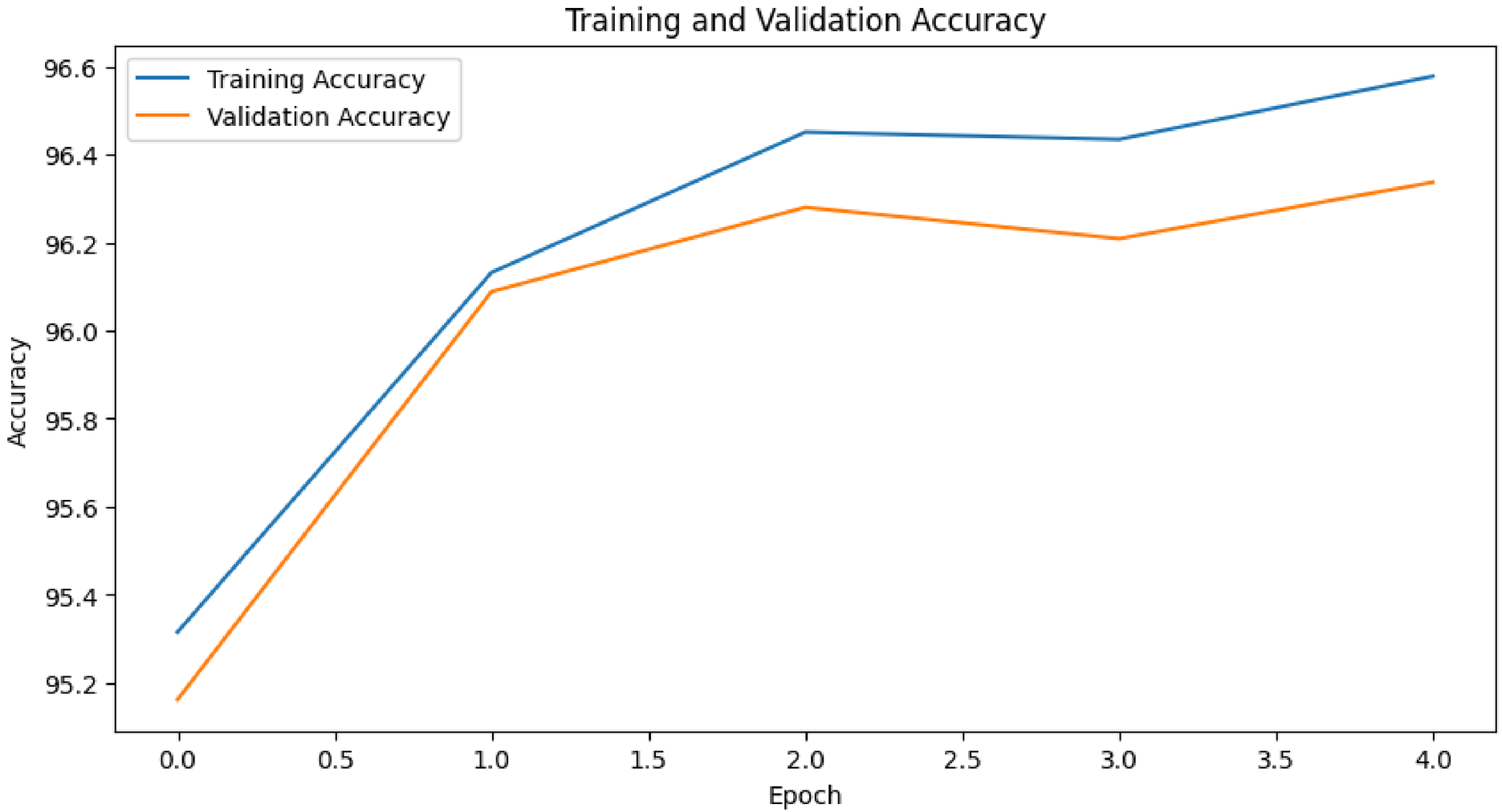
Training Loss vs Validation Loss

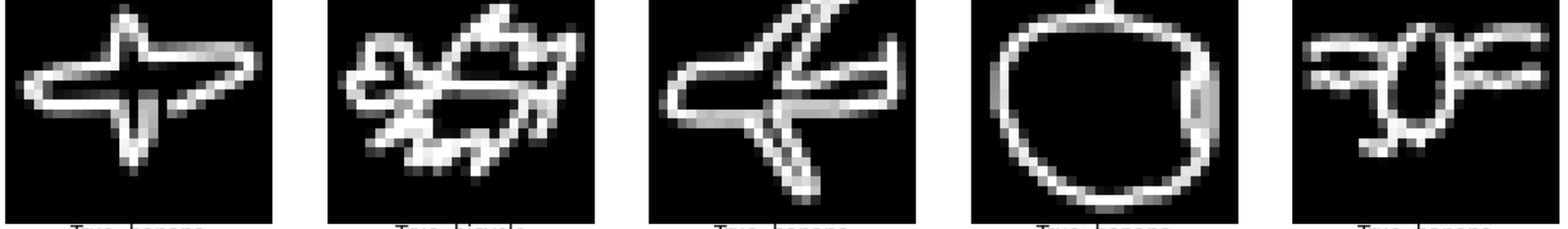
- Training loss is computed on the data the model is trained on.
- Validation loss is calculated on a separate set of data to evaluate the model's performance on unseen data.

Loss Change over epochs



Accuracy over Epochs



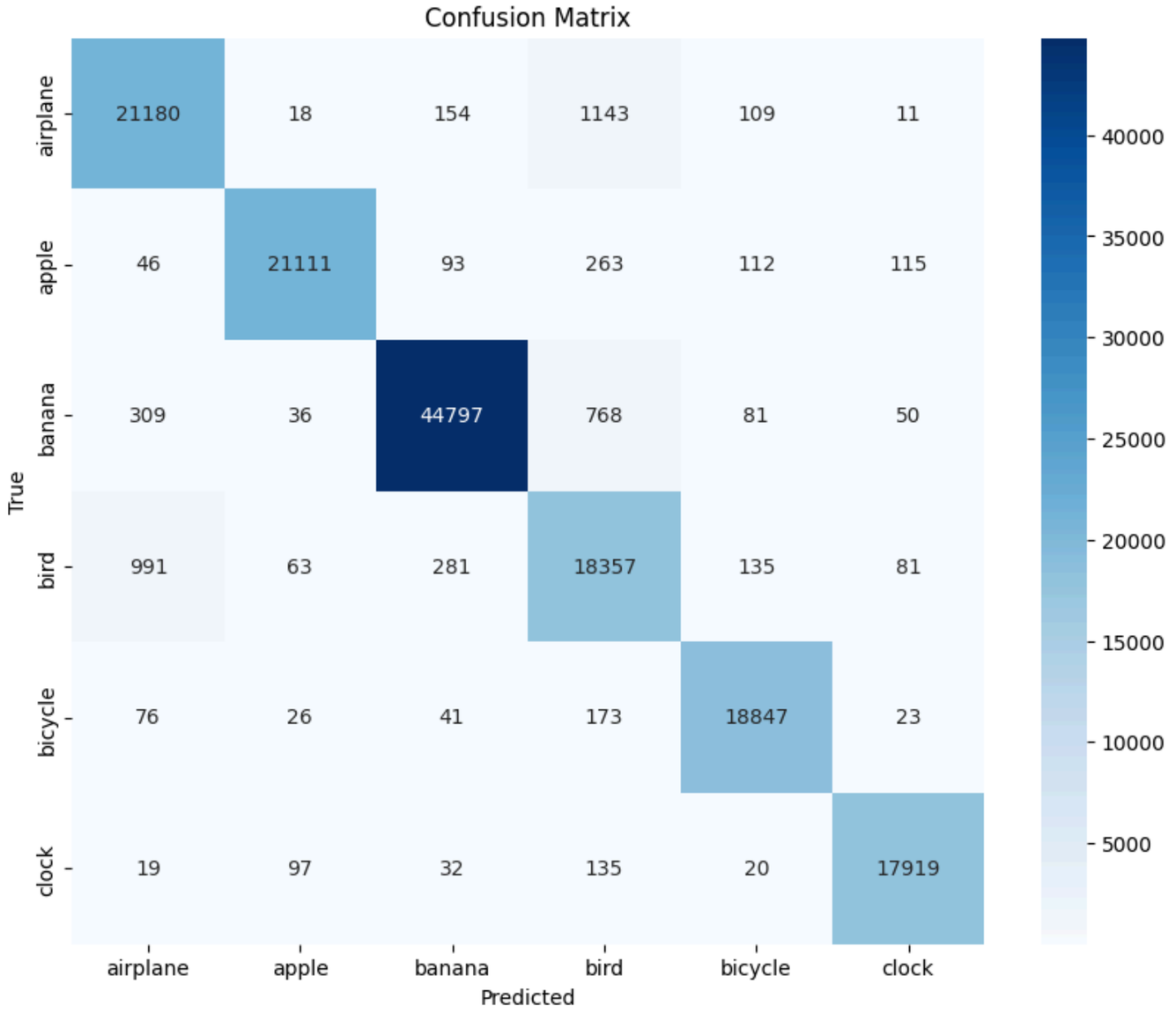

True: banana
Pred: banana
True: bicycle
Pred: bicycle
True: banana
Pred: banana
True: banana
Pred: banana
True: banana
Pred: banana
True: banana
Pred: banana
True: banana
Pred: banana
True: banana
Pred: banana
True: airplane
Pred: airplane
True: apple
Pred: apple
True: banana
Pred: banana
True: bicycle
Pred: bicycle
True: banana
Pred: banana
True: banana
Pred: banana
True: banana
Pred: banana
True: banana
Pred: banana
True: banana
Pred: banana
True: banana
Pred: banana
True: airplane
Pred: airplane
True: apple
Pred: apple

96%

Accuracy

To ensure the accuracy was meaningful, we used a validation split during training, monitoring both training and validation accuracy across epochs. This helped in detecting overfitting, where the model might perform well on the training data but poorly on new data. Additionally, techniques like data augmentation and dropout layers were employed to improve accuracy and generalization.

Confusion Matrix



Challenges

Overfitting

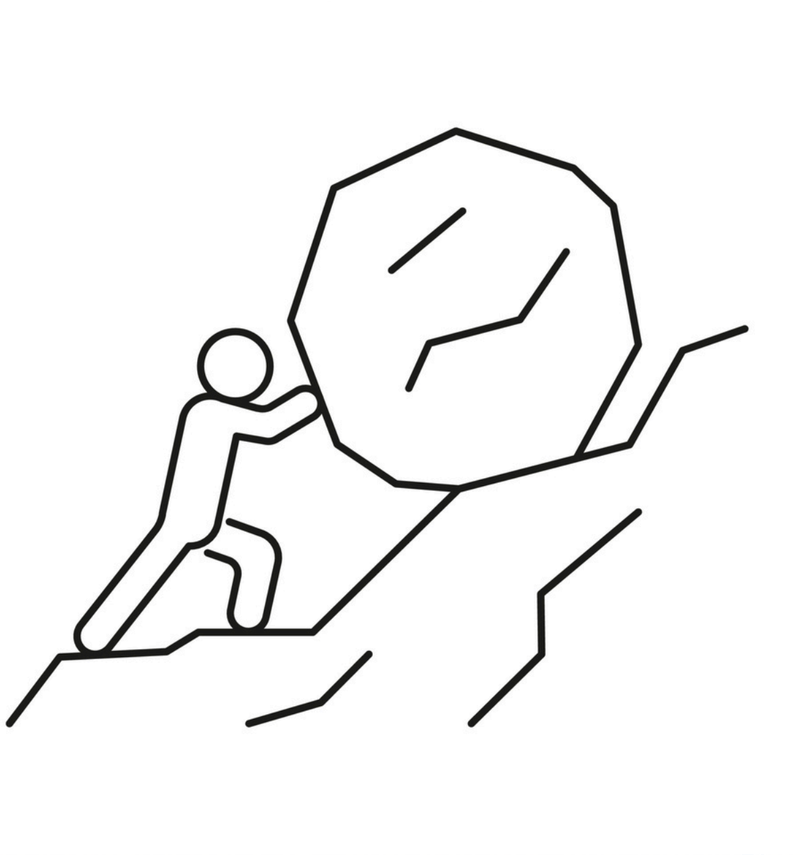
Overfitting was managed by using techniques like dropout layers and data augmentation to improve generalization.

Class Imbalance

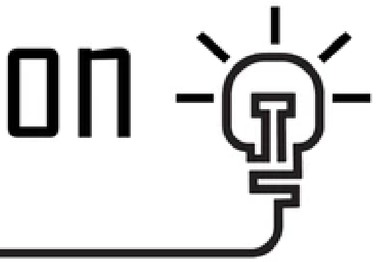
For datasets with class imbalance, we used weighted loss or over-sampling techniques to ensure each class was well-represented during training.

Computational Requirements

Training CNNs can be computationally expensive. We used GPU acceleration to optimize training speed and efficiency.



Conclusion



- Summary:
  - This project demonstrates how deep learning models, specifically CNNs, can effectively classify images into categories. Our model achieved promising accuracy, showing potential for real-world applications.
- Final Thoughts:
  - With further tuning and larger datasets, models like this can be deployed in various fields, transforming how we approach classification problems in both industry and academia.

Thank you very much!

github.com/Saint-Potato/doodleClassification