Image Classification of Hand-Drawn Sketches using Convolutional Neural Networks

Members:

- 2702368223 - Helena Aurelia Sanjaya
- 2702331100 - Reinhart Gian Widjaja
- 2702210432 - William Sebastian Liman

Final Project

-

Deep Learning

01

Introduction

Background

Computer vision excels at classifying realistic photos but struggles with abstract sketches. Sketches lack texture/color and rely solely on stroke lines, varying wildly by user style

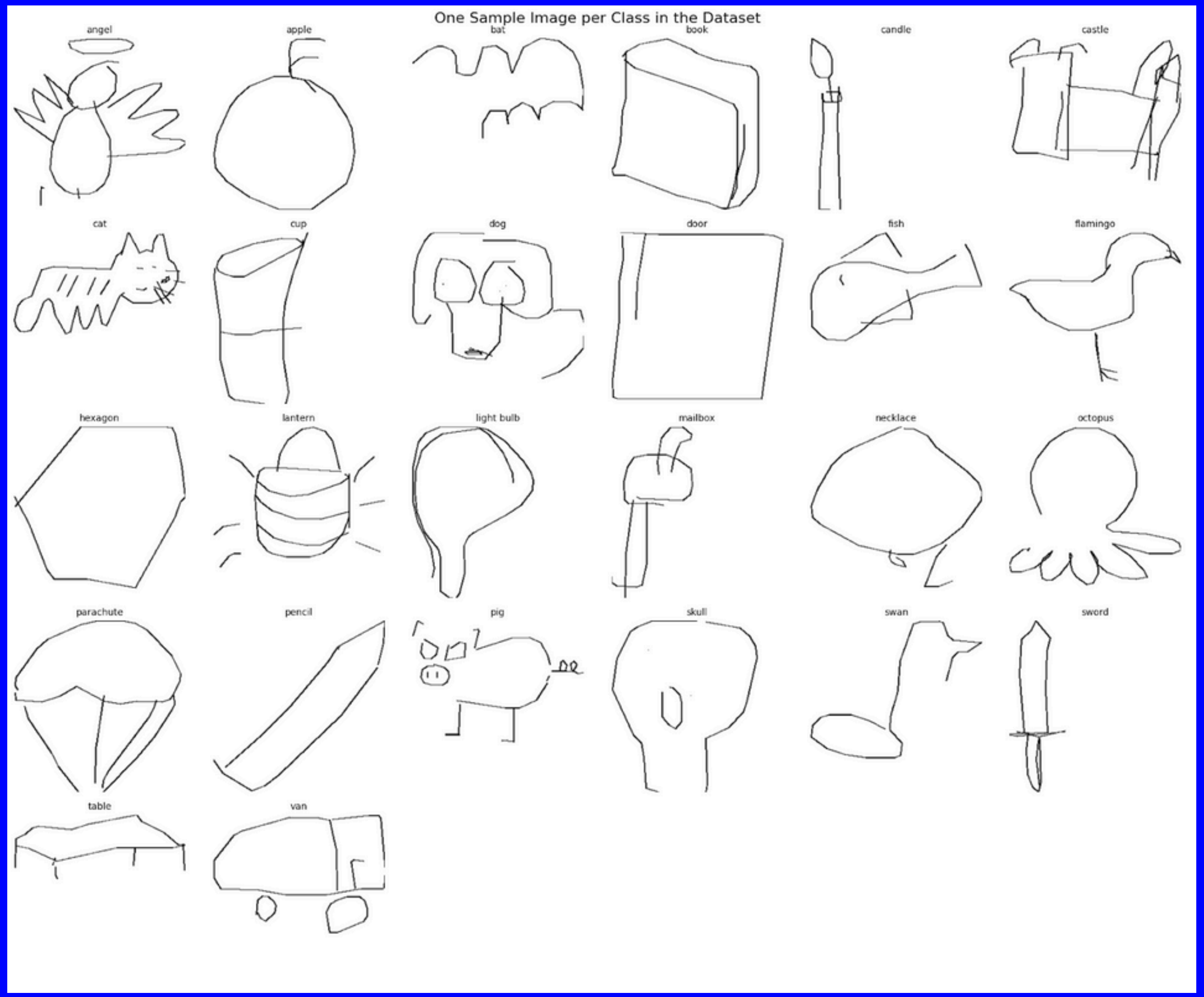
Objectives

- Develop a Deep Learning model (CNN) to classify sketches.
- Curate and preprocess a subset of the QuickDraw dataset.
- Deploy the model for real-time inference on a web app.

Dataset

Google Quick, Draw! Subset

- **Source:** Subset of the massive Quick, Draw! dataset (50M drawings).
- **Selection:** 26 distinct classes (e.g., Apple, Cat, Sword, Hexagon).
- **Volume:** 130,000 total images (5,000 per class) to ensure balance.
- **Format:** 255×255 pixel raster images.



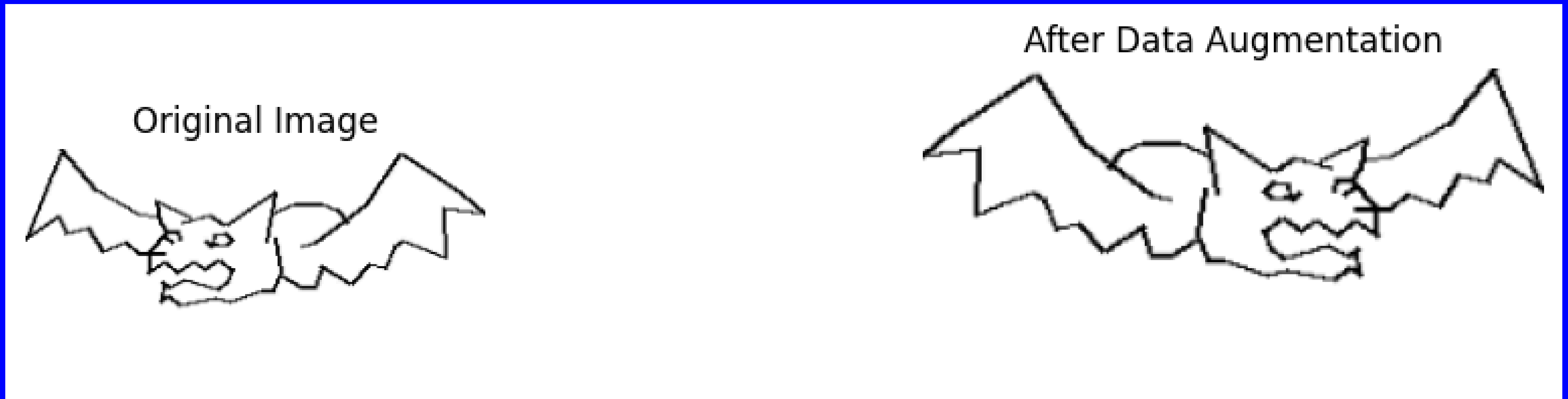
Sample Image for every Class

Final Presentation - Deep Learning

Data Preprocessing Pipeline

Resizing: Uniform resolution of 224×224 pixels.
Grayscale Conversion: Reduced to 1 channel to focus on stroke information.
Normalization: Scaled pixel values to [0, 1].

Augmentation Techniques: Random horizontal flipping, brightness adjustment, contrast variation.



Data Augmentation

02

03

Model Architecture

Layer Type	Parameters	Output Shape
Input Layer	-	(224, 224, 1)
Conv Block 1	2× Conv2D (32 filters, 3×3), BatchNorm, ReLU	(224, 224, 32)
Max Pooling 1	Pool Size: (2, 2)	(112, 112, 32)
Dropout	Rate: 0.5 × dropout	(112, 112, 32)
Conv Block 2	2× Conv2D (64 filters, 3×3), BatchNorm, ReLU	(112, 112, 64)
Max Pooling 2	Pool size: (2, 2)	(56, 56, 64)
Dropout	Rate: 0.5 × dropout	(56, 56, 64)
Conv Block 3	2× Conv2D (128 filters, 3×3), BatchNorm, ReLU	(56, 56, 128)
Max Pooling 3	Pool size: (2, 2)	(28, 28, 128)
Dropout	Rate: 0.5 × dropout	(28, 28, 128)
Conv Block 4	2× Conv2D (256 filters, 3×3), BatchNorm, ReLU	(28, 28, 256)
Max Pooling 4	Pool size: (2, 2)	(14, 14, 256)
Dropout	Rate: 0.5 × dropout	(14, 14, 256)
Global Avg Pooling	-	(256)
Dense Layer 1	Units: 256, Activation: ReLU	(256)
Dropout	Rate: dropout	(256)
Dense Layer 2	Units: 128, Activation: ReLU	(128)
Dropout	Rate: dropout	(128)
Output Layer	Units: 26, Activation: Softmax	(26)

Training Strategy

Optimizer: Adam with adaptive learning rate (ReduceLROnPlateau).

Regularization:

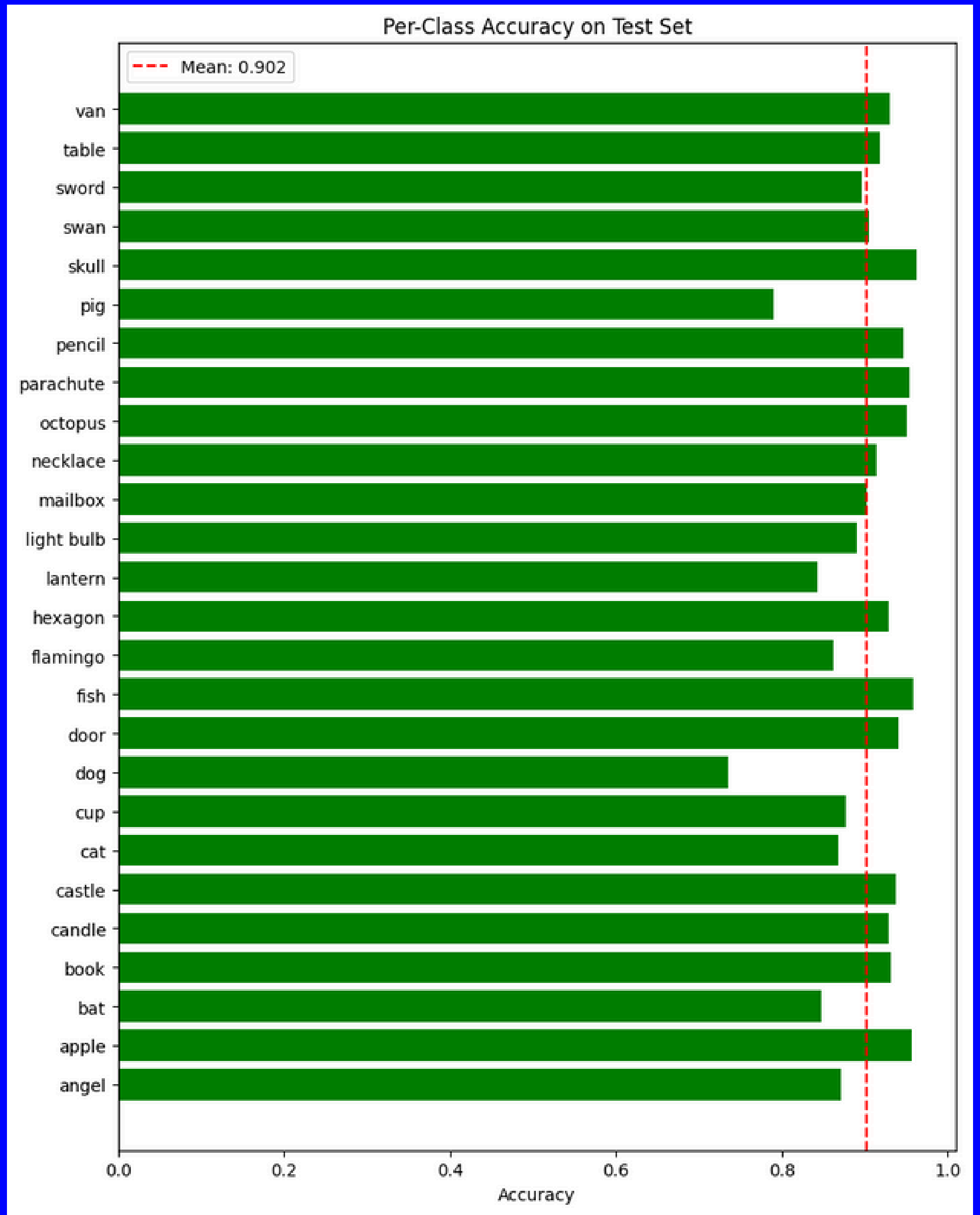
- Batch Normalization for stability.
- Dropout (Rate 0.5) to prevent overfitting.
- Early Stopping based on validation loss.

Training Time: ~160 minutes on a single GPU (50 epochs max).

Evaluation Results

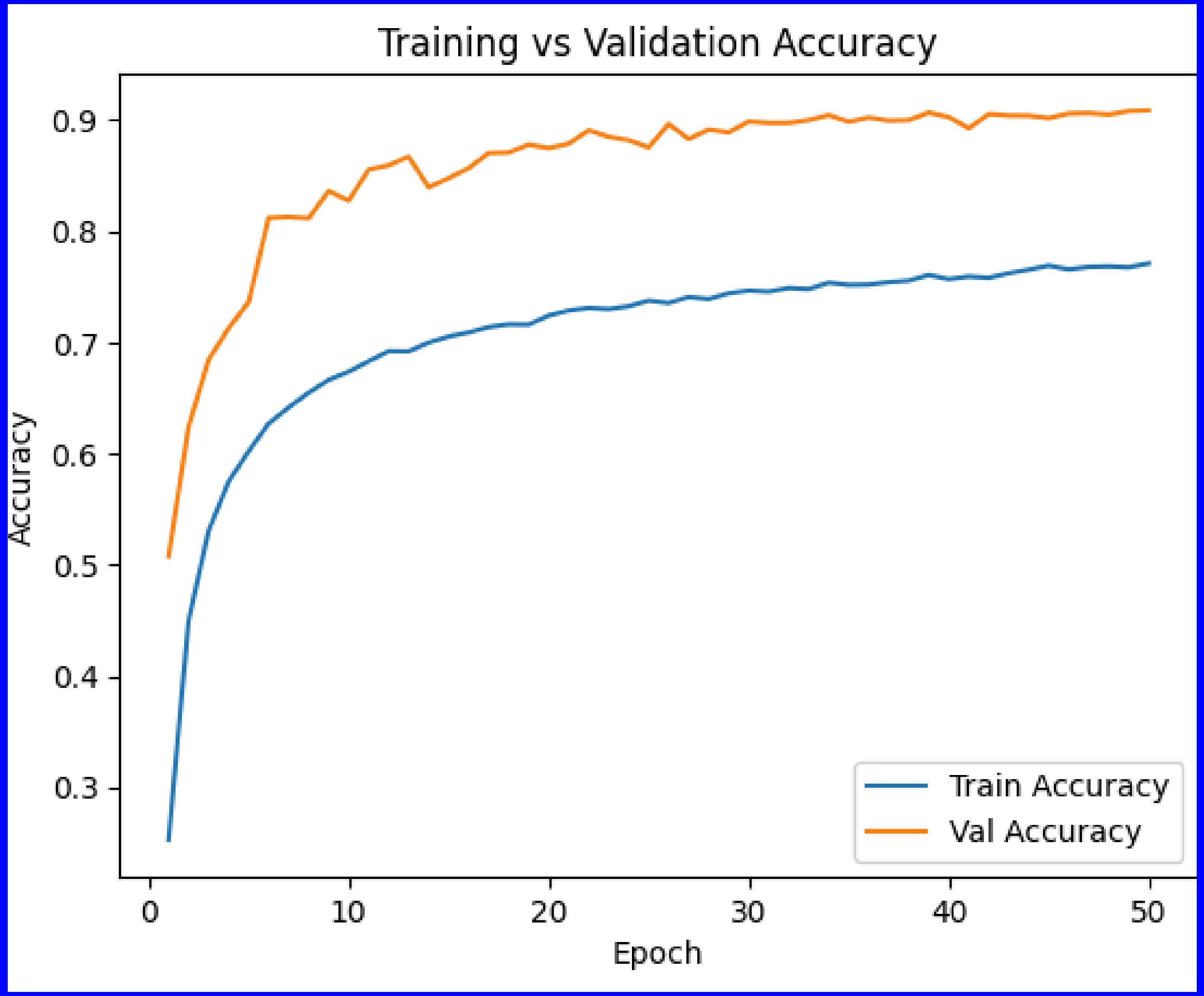
Final Test Accuracy: 90.13%
Loss: 1.0267

04

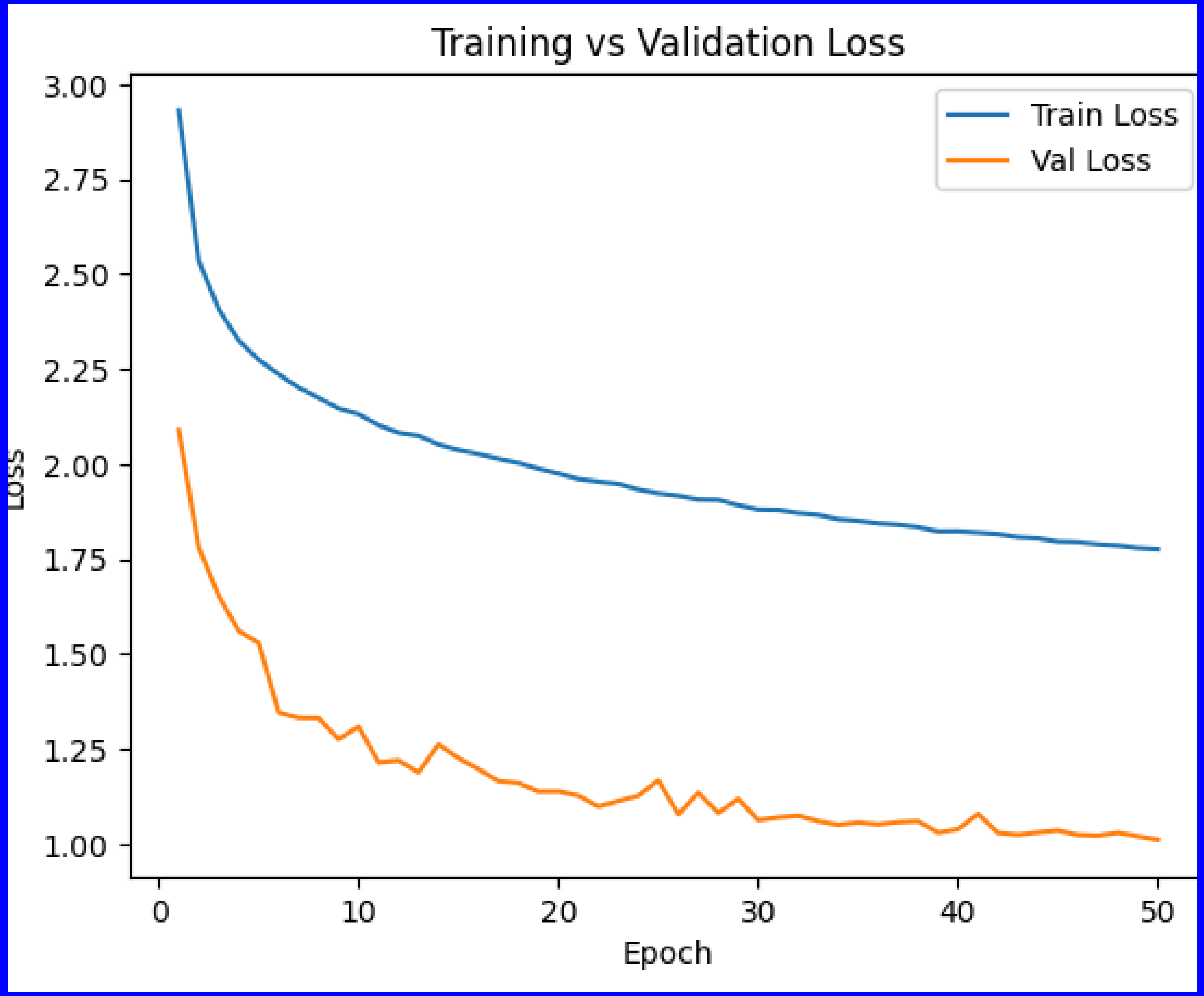


Per-Class Accuracy on Test Set

04

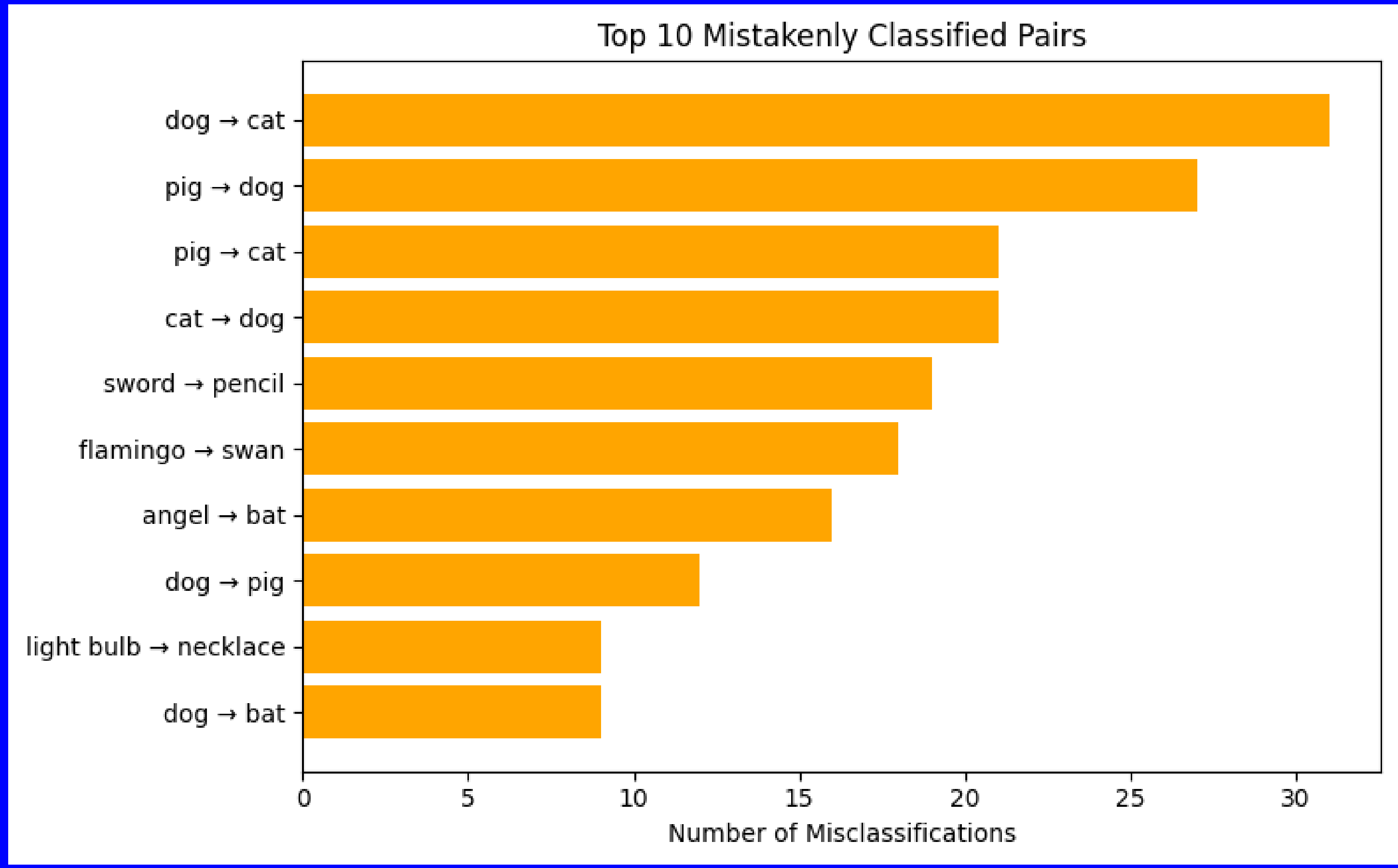


Training vs Validation Accuracy



Training vs Validation Loss

Error Analysis



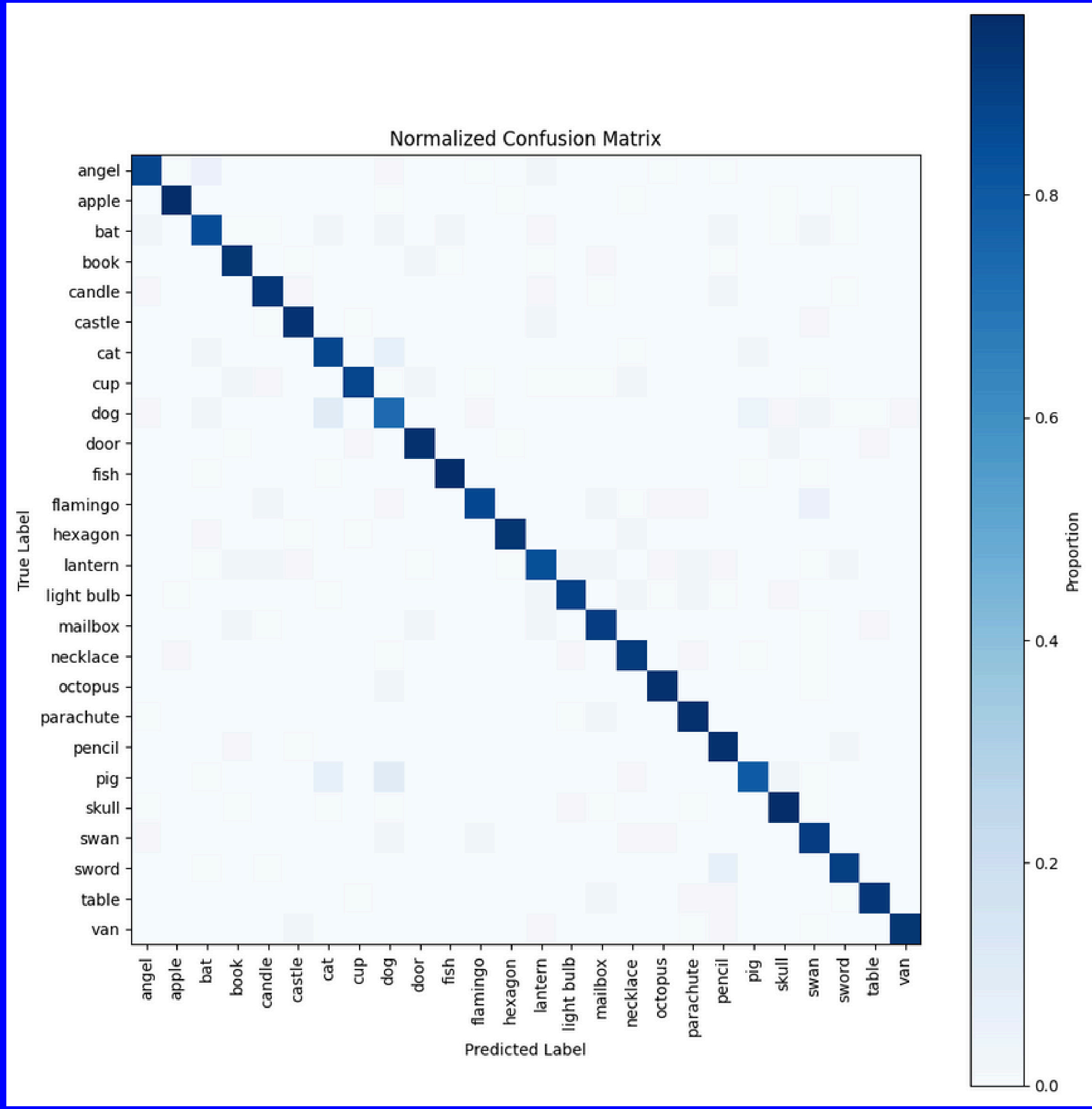
Top 10 Mistakenly Classified Pairs

High Accuracy: Distinct shapes (e.g., Hexagon, Sword).
Common Confusions: Visually similar pairs found in the Confusion Matrix.

- **Top Error: Dog → Cat.**
- **Other Errors: Pig → Dog, Flamingo → Swan.**

Confusion Matrix

04



Normalized Confusion Matrix

Final Project – Deep Learning

05

Deployment

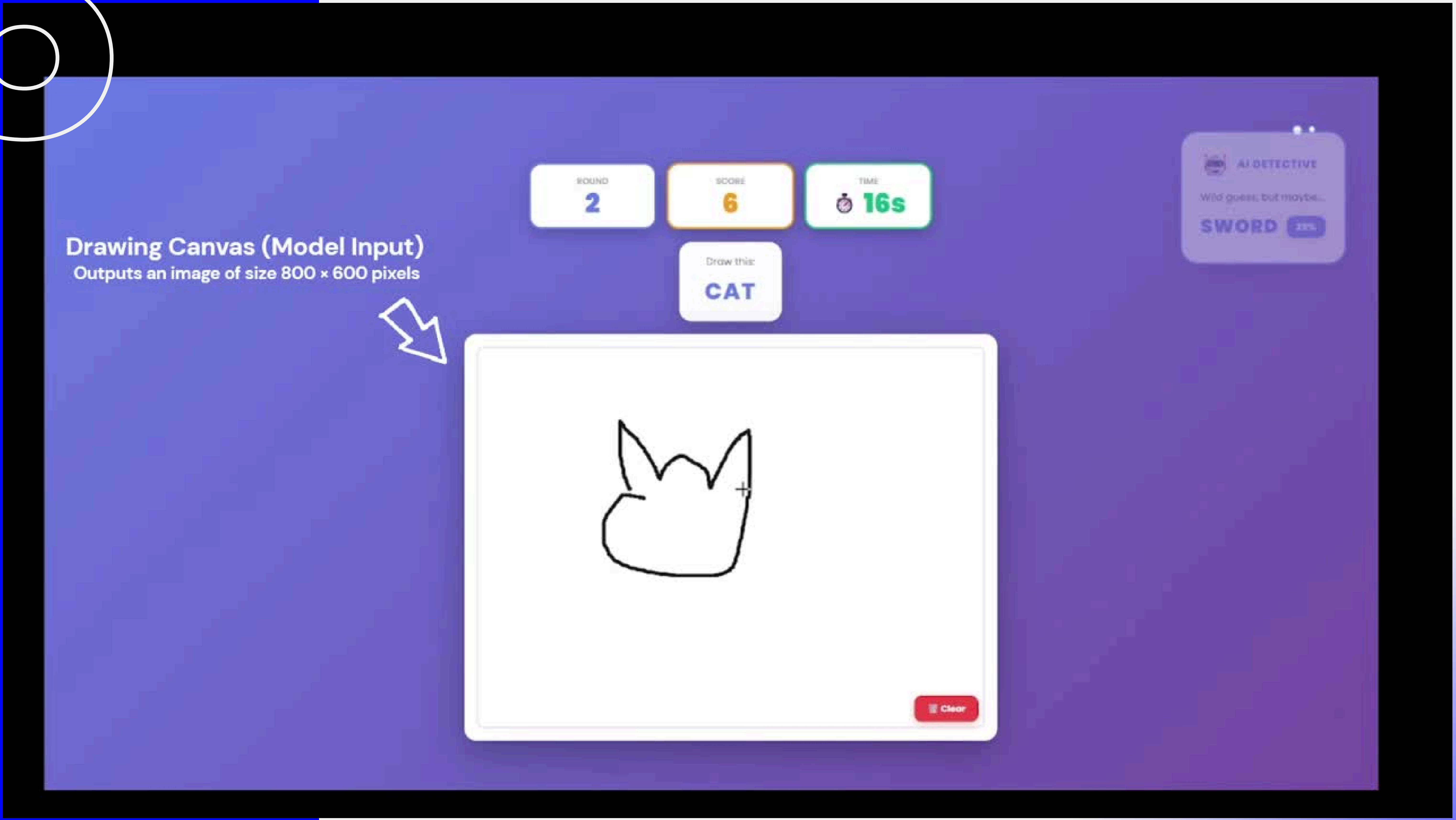
Tech Stack

- Frontend: React
- Backend: FastAPI
- Model: Keras

Workflow

1. User draws on HTML5 Canvas.
2. Image sent to backend after 0.5s inactivity.
3. Preprocessing (Crop, Resize 224×224, Normalize).
4. Real-time inference display.

05



ROUND
2
SCORE
6
TIME
16s
AI DETECTIVE
Wild guess, but maybe...
SWORD
Drawing Canvas (Model Input)
Outputs an image of size 800 × 600 pixels
Draw this:
CAT
Clear

06

Reflection & Conclusion

Strengths

Strong accuracy (90.13%) on unseen test data and effective regularization prevented overfitting

Limitations

Hardware constraints limited the study to 26 out of 345 possible classes

Thank You