Mini Project Report

on

Data-Centric AI VS Model-Centric AI

Submitted in partial fulfillment of the requirements for the VI semester

Bachelor of Technology

in

Data Science

of

Garden City University

By

Done By:
Tharun SP (22BTDS144)
Section: "D" (Data Science)

Introduction

For this mini-project, we chose Option 1: Improving a Simple Classifier with Data Augmentation to explore how data-centric approaches can enhance model performance. The objective is to investigate the impact of data augmentation on the performance of a simple image classifier distinguishing between Asian Elephants and African Elephants.

This project contrasts Data-Centric AI, which focuses on improving data quality and quantity, with Model-Centric AI, which primarily focuses on enhancing the model architecture and tuning. We demonstrate how meaningful improvements in model performance can be achieved by focusing solely on augmenting and diversifying the dataset.

Methodology / Approach

Dataset:

We created a binary classification dataset consisting of around 100 images per class (Asian and African Elephants). Images were sourced from a small public dataset available in ZIP format and manually curated for quality and relevance.

Data Augmentation Techniques:

We used TensorFlow's ImageDataGenerator to implement the following augmentations:

- Random Rotation (up to 30°)
- Horizontal Flip
- Zoom Range (up to 20%)

These augmentations simulate real-world variations in elephant images, such as different orientations, lighting, and scale, without altering semantic content.

Dataset Creation:

The augmented dataset included 3 additional variants per original image, increasing the size and diversity of the dataset by 300%.

Model Architecture:

We implemented a basic Convolutional Neural Network (CNN) using TensorFlow:

- 3 convolutional layers (32, 64, 128 filters)
- ReLU activations and max pooling
- 1 dense layer with 512 units
- Final sigmoid output layer for binary classification

Training Setup:

- Image size: 150×150
- Batch size: 32
- Epochs: 10
- Optimizer: Adam
- Loss: Binary Crossentropy
- Train-Test Split: 80:20 (held out before augmentation)
- Evaluation Metrics: Accuracy, Precision

Results And Findings

Dataset	Training Accuracy	Validation Accuracy
Original	0.85	0.86
Augmented	0.98	0.98

Visualizations:

- Side-by-side comparison of original vs. augmented images
- Training/validation accuracy curves showed better generalization with augmentation

Observations:

- The model trained on the original dataset overfit quickly and had lower performance.
- Augmentation led to significantly improved generalization, accuracy, and robustness.

<u>Discussion And Conclusion</u>

This project illustrates the power of Data-Centric AI: without modifying the model architecture, we improved accuracy from 85% to 98% simply by enhancing the dataset.

Key Takeaways:

Model-Centric efforts may hit diminishing returns when working with small, limited datasets.

Data augmentation provides a scalable, practical approach to boosting performance.

Data-Centric AI promotes better real-world performance by simulating diverse scenarios during training.

Challenges:

Choosing meaningful augmentation without distorting key features was critical.

Ensuring class balance post-augmentation was necessary to avoid bias.

Future Work:

Use automated augmentation strategies (e.g., AutoAugment).

Expand dataset with more species and deploy the classifier in a real-time mobile app.

Code

```
# -------------------------
# STEP 1: Upload & Extract ZIP
# -------------------------
from google.colab import files
import zipfile
import os

# Upload the ZIP file (you will be prompted to upload your file)
uploaded = files.upload()

# Extract the ZIP file (assumes only one ZIP file was uploaded)
for filename in uploaded.keys():
    zip_ref = zipfile.ZipFile("/content/archive.zip", 'r')
    zip_ref.extractall()
    zip_ref.close()

print("ZIP extracted successfully.")

# -------------------------
# STEP 2: Import Libraries
# -------------------------
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
# -------------------------
# STEP 3: Define Paths & Settings
# -------------------------
IMG_SIZE = (150, 150)
BATCH_SIZE = 32

train_dir = '/content/dataset/test'
val_dir = '/content/dataset/test'

# -------------------------
# STEP 4: Data Preprocessing
# -------------------------
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

# -------------------------
# STEP 5: Build CNN Model
# -------------------------
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Binary classification
])

# -------------------------
# STEP 6: Compile the Model
# -------------------------
model.compile(
    loss='binary_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
# -------------------------
# STEP 7: Train the Model
# -------------------------
EPOCHS = 10
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=EPOCHS
)

print(f" Final Training Accuracy: {history.history['accuracy'][-1]:.2f}")

print(f" Final Validation Accuracy: {history.history['val_accuracy'][-1]:.2f}")

# -------------------------
# STEP 8: Save the Model
# -------------------------
model.save('binary_classifier_model.h5')
print(" Model saved as binary_classifier_model.h5")
```

References

Kaggle Dataset: https://www.kaggle.com/datasets/vivmankar/asian-vs-african-elephant-image-classification/data