CS 554 PROJECT - IMAGE CLASSIFIER (ML)

TEAM - Teja Ranga Rao Paladagu (tp577@scarletmail.rutgers.edu)
Dheekshith Dev Manohar Mekala (dm1653@scarletmail.rutgers.edu)

ABSTRACT

I propose building a Machine Learning classifier as a stand-alone application with possible integration that classifies images into two classes (Animated/Normal). We would also distinguish why the proposed model on byte array (data in the form of bytes (packets)) is better than any Vision-based approach, and for image transfer, we will use the network protocols (TCP/UDP (whichever suits best)) switching between based on performance and quality.

Future scope - Increase the number of classes of images to make the application more robust and scalable. We can also extend this project's implementation to the classification of videos (animated/gifs/real-life etc.)

IMPORTANCE

Distinguishing cartoon images from real-life images is essential for many applications, such as web browsers, or when the user searches for images in multimedia documents. The particular approach is significant as we operate on byte stream/array rather than the number of pixels that many of the novel ML approaches work on.

IMPLEMENTATION LOGIC

The input dataset is validated, and data analysis and class segregation are done. The data is pre-processed to nullify bias, and inappropriate inputs are removed. The images are forwarded using the network protocols (TCP/UDP) in the form of packets, and these packets from protocols are saved to a series of byte arrays/streams. The byte array of the packages is then calculated for every image, the inherent patterns are identified, and relationships are established. A key difference could be that the byte arrays for animated images could be more extensive compared to real-life pictures. There could be patterns as the data in animated images are in compressed format.

```
with open("image.jpeg", "rb") as image:
    f = image.read()
    b = bytearray(f)
```

CS 554 PROJECT - IMAGE CLASSIFIER (ML)

WORKFLOW

Take an input image from the user or image.random()

Perform data pre-processing and save and convert the image packets into a series of byte arrays.

Evaluate the byte array and extract features based on identified patterns/sequences.

Code implementation 1.) Logistic Regression 2.) One-Hot Encoding

Model evaluation – UDP - Confusion Matrix - TP, TN, FP, FN & Recall, and Precision.

Outputs Expected:

Class A - Animated Image

Class B - Real image // binary classification

Tech Stack - Python, Machine Learning, Visual Studio Code, Git version control, Computer Networks, Bit Manipulation.

CS 554 PROJECT - IMAGE CLASSIFIER (ML)

ACKNOWLEDGEMENT

we want to thank our professor, Dr. Richard Martin, for this opportunity to execute the theoretical ideas in this final project.