16th December 2022

CS 552: COMPUTER NETWORKS

Project: Image Classification Using Network Protocol

TEAM MEMBERS:
Dheekshith Dev Manohar Mekala (*dm1653*)
Teja Ranga Rao Paladugu (*tp577*)

ABSTRACT

We developed a robust machine learning classifier that transfers images through Transmission Control Protocol (TCP) and classifies those images as either animated or real images by performing various classification operations along with relative accuracies for each operation. We programmed three distinct image classifiers where one performs on byte-array, and the other two classifiers use Computer Vision (CV). Each respective operation had a trade-off between reliability and speed of operation.

Keywords: TCP (Transmission Control Protocol), Machine Learning, Computer Vision, Color Counting, Bilateral Filtering, Byte-Array, One-Hot Encoding, Base64 String, Hexadecimal Byte-Array.

***Programming Language*:** Python 3.10.

INTRODUCTION

Distinguishing animated images from real-life photos is critical for many applications, including web browsers and image searches in multimedia documents. The initial technique we implemented is more significant if robustness and security are a primary concern than accuracy because we operated on byte arrays rather than pixels, unlike many fundamental ML models. Image classification applications are used in many areas, such as medical imaging, object identification in satellite images, traffic control systems, brake light detection, machine vision, and more. This application we built would classify as an intermediate technology where one of its objectives is to serve as a fundamental basis for future machine learning models that would classify images on byte-array using pattern recognition than image processing.

IMPLEMENTATION LOGIC

One way to discriminate between cartoon and natural scene images is to compare a given image to its "smoothed" self. The motivation behind this is that a "smoothed" cartoon image statistically will not change much, whereas a natural scene image will. In other words, take an image, “cartoonify” (i.e., smooth) it, and subtract the result from the original:

isNotACartoonIndex = mean (originalImage - smooth(originalImage))

This difference (i.e., taking its mean value) will give the level of change caused by the smoothing. The index should be high for non-smooth original (natural scene) images and low for smooth original (cartoony) images.

Bilateral filtering can be done with OpenCV using the **cvSmooth** function with the **CV_BILATERAL** parameter. As for subtracting the “cartoonyfied” image from the original, we did that using the Hue

channel of the HSV images. This means we need to first convert both images from RGB to HSV. We computed the entropy of the color histogram of the image. Cartoon-like images should have fewer shades of different colors and thus a lower entropy. Also, sometimes the number of colors over the number of pixels may not be enough. There may be jpeg artifacts, for instance, that artificially increase the number of colors in the image, but only for a few pixels. In this case, most pixels in the image would still have very few colors, which would correspond to low entropy.

We have an RGB image, for each pixel, the R, G, and B values range from 0 to 255. We divided this range into n bins. Our algorithm counts how many pixels fall into each one of these 3-dimensional bins and divides the values of the bins by the total number of pixels so that the histogram sums up to 1; then computes the entropy, which is –

sum_i p_i * log(p_i), where p_i is the value of the i'th bin.

MOTIVATION

Traditional machine learning methods (such as multilayer perception machines, support vector machines, etc.) mostly use shallow structures to deal with a limited number of samples and computing units. When the target objects have rich meanings, the performance and generalization ability of complex classification problems are obviously insufficient. The convolution neural networks (CNN) developed in recent years have been widely used in image processing because they are good at dealing with image classification and recognition problems and have greatly improved the accuracy of many machine learning tasks. But CNN is prone to malware and DDOS attacks and also computationally intensive. Our proposed model performs the classification on the core-level byte array rather than the top-level compressed image and is considered robust but has trade-offs between speed and reliability.

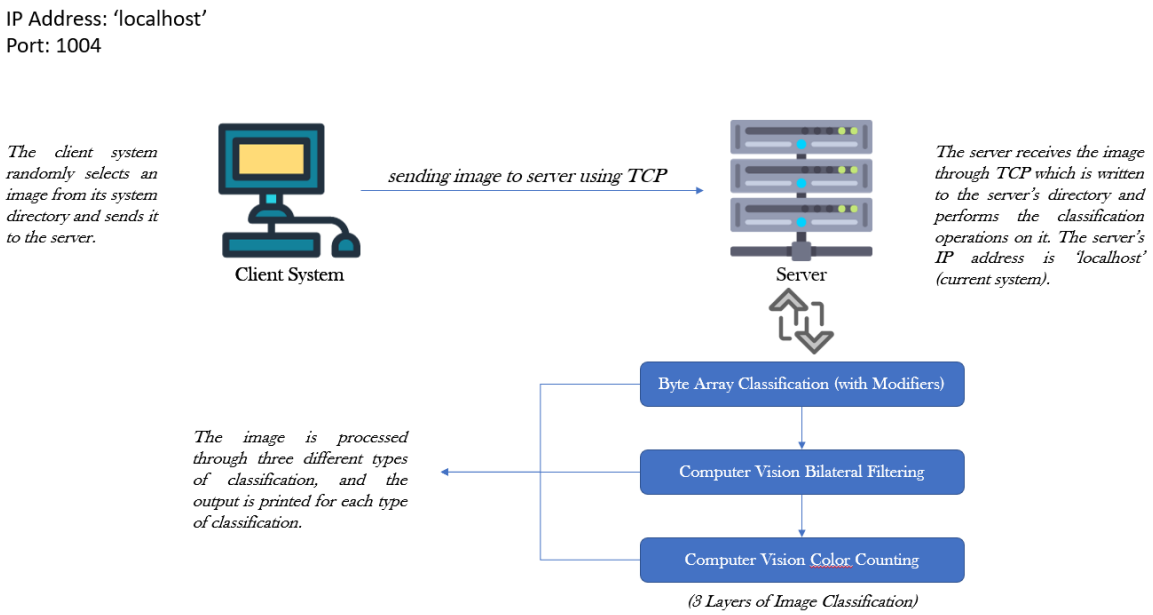
If this project is successful, it will be deemed helpful for domains where speed and security are the primary factors for image classification. This model is a good trade-off for speed vs. accuracy factors. Also, when images are transferred over the network in the form of packets, we can identify what type of multimedia data is being transmitted by using this model on the byte streams of data being transferred.

METHODOLOGY

We used a socket programming module to set up TCP and an IPv4 server. In Sockets, **socket.AF_INET** specifies IP version 4 and **socket.SOCK_STREAM** specifies TCP.
The byte-array image classification is performed on three parameters: The image is converted into a byte array, and its total length is calculated. The byte-array is again converted into a hexadecimal byte-array, and its total length is calculated for further classification.
The image is also converted into a Base-64 string for more efficient classification. These three parameters' observation metrics are One-Hot encoded (specifying exact ranges) for improved accuracy.
Computer Vision Bilateral Filtering is a technique that blurs the image intentionally and compares the blurred image against the original image using their respective RGB histograms. Color Counting is a technique that counts the total number of colors used from the majority of the image by taking the vertical 512 pixels of the image.

We evaluate the model efficiency with a confusion matrix by classifying the image as true, false positives, and negatives. Our model classifies images into two classes: Real and Animated images classes.

ARCHITECTURAL DIAGRAM



Existing models - All the existing models till now use TensorFlow and Keras ML libraries to conduct image classifications. These models are very slow and computationally intensive when used on bulk data.

Our models - The model we built would serve as an excellent fundamental starting point for other future ML models; if appropriately tweaked, a well-developed and trained pattern recognition ML model performing its operations on byte streams rather than intensive image processing can effectively replace the existing image ML models.

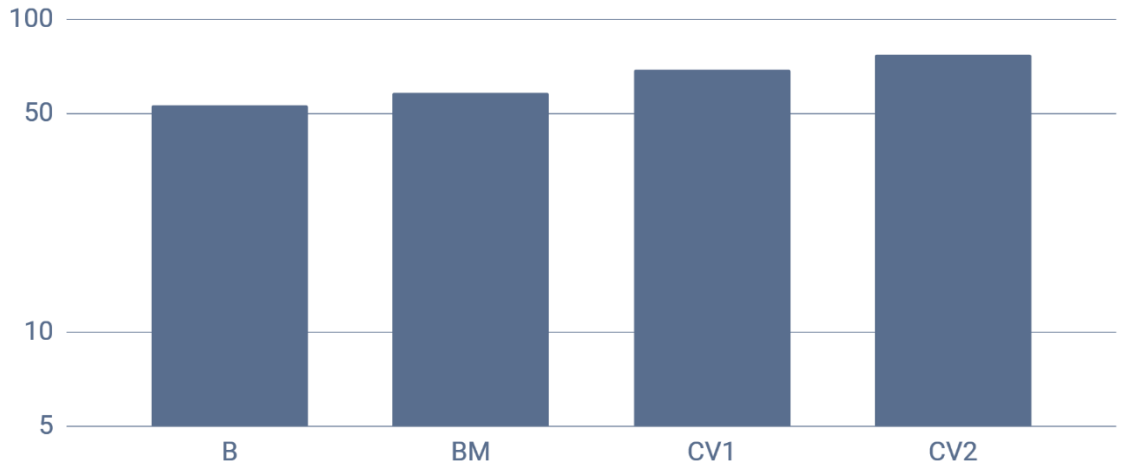
Why TCP?

Although User Datagram Protocol (UDP) is quick for sending data, TCP guarantees successful image transfer. In contrast, once a packet is lost or corrupted in UDP, it's really difficult to reform the image again. Protocols cannot allow for a complete transfer of an entire image directly if the size is large. They can only send images in chunks. The buffer size we used for image chunks is 2048 bytes. As TCP is a

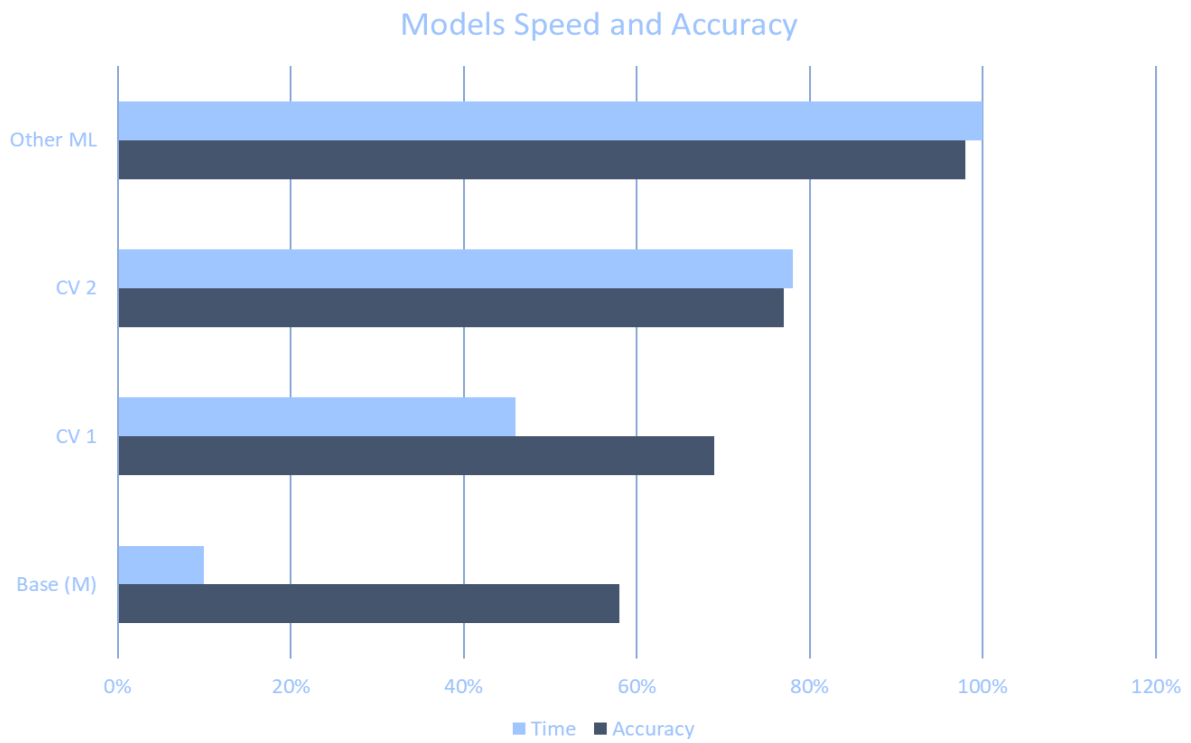
continuous data stream, it will efficiently handle the data without any data loss. The handling of packets in UDP is also very inefficient (like resending the entire image data for a single lost/corrupted packet) compared to TCP.

RESULTS

We compared our base model with other practical image classification models in the market. We tested the speed and accuracy of all the image classification models and accumulated the data below.

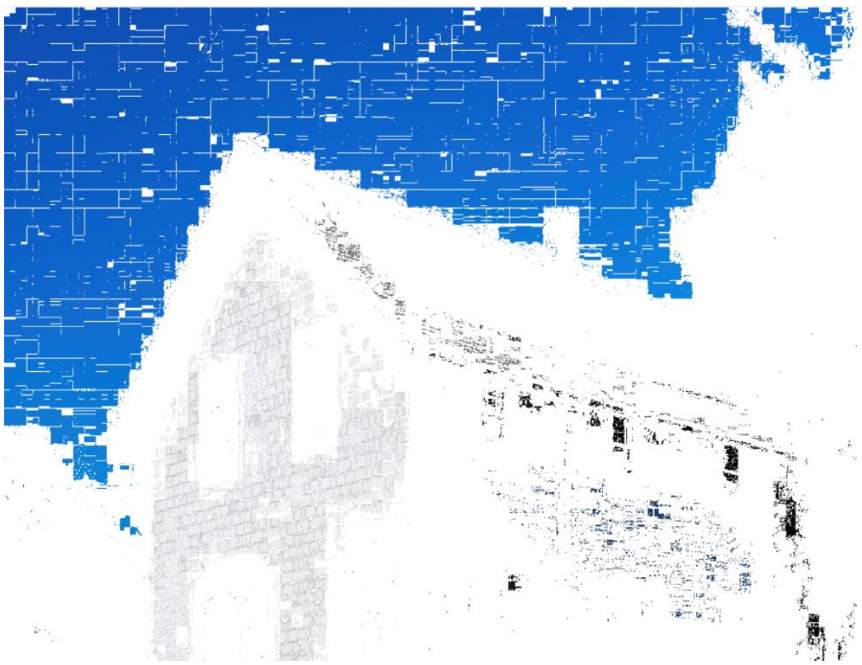


We got 53% accuracy for our base byte array model, and with equipped with modifiers, we pushed the accuracy to 58%. For the traditional Computer Vision models, the accuracies are 69% for the bilateral filtering technique and 77% for the color counting technique. The reason we chose to implement these two CV models out of all other models is because of their excellent, consistent trade-off between speed vs. accuracy. All the other models give up speed for increased accuracy or vice-versa.



We can see how the computation time increases as its accuracy increases. Here the percentage of time is scaled down based on the longest time of a model being equated to 100% as the max threshold.

This is a preview of how the Computer Vision color counting algorithm works on real images.



This is a preview of how the Computer Vision color counting algorithm works on animated images.

CHALLENGES

- ✓ We learned that the byte-array length primarily depends on the image's resolution. So, we tested 83 images of equal resolution for precise comparison and found that animated images have more byte count than real images.
- ✓ The initial data has many outliers, so we devised the One-Hot Encoding technique to narrow down the total range of bytes length, effectively ignoring the outliers. With this, we were able to improve the accuracy by a bit.
- ✓ For the CV bilateral filter, we observed that the level of the blur of the image had a slight impact on accuracy. Still, if the blur level crosses a certain threshold, it negatively affects the model's accuracy. We tested the application many times to come up with a reasonable threshold for the intensity of the blur to push the accuracy as best as possible.

- ✓ Initially, we seldom got corrupted images through TCP. Later, we figured out that it was due to the time-out limit of the TCP socket connection.

CONTRIBUTIONS

Dheekshith Dev Manohar Mekala (dm1653)

- Devised the initial workflow design for the project.
- Conceptualized and implemented color counting and bilateral filtering approaches, improving the accuracy and speed of the model.
- Generated hexadecimal and base64 byte arrays, making the model robust and reliable.
- Implemented TCP connection between client and server, which provides a pipeline for image delivery for classification.
- Ran multiple tests for identifying proper thresholds for algorithms as well as for the overall model evaluation.
- Debugged the entire code and fixed it wherever necessary.
- Performed output validation.
- Validated the test data set for the project.

Teja Ranga Rao Paladagu (tp577)

- Data Aggregation and Pre-processing, Conceptualized and developed One hot Encoding approach, Model validation: Accuracy, Precision.
- Performed data pre-processing and saved and converted the image packets into a series of byte arrays. Evaluated the byte array and extracted features based on identified patterns/sequences.
- Implemented the One-Hot Encoding concept.
- Confusion Matrix - TP, TN, FP, FN & Recall, and Precision.
- Initialized the test data set for the project.

CONCLUSION

Finally, we strongly believe the model we came up with might provide a fundamental skeleton for future ML Image Classification models that do not necessarily use intensive image processing techniques but instead use other ways to classify images like the new methodology we implemented.

Also, there are not a lot of great pattern recognition models to explicitly use on any form (Base64, Hexadecimal) of byte-array data. So, if we were to experiment again, we would like to implement an ultimately better pattern recognition ML model from scratch that can classify images accurately, unlike the pre-built pattern recognition models out there.

To advise successors who would want to draw on this project as their basis, we would like to point out that it is worth delving into the concept of pattern recognition on byte array for image classification, which would inspire a whole new set of ideas and techniques in the field of Machine Learning.

FUTURE SCOPE

The default TCP for image transfer can also be replaced with other protocols for an even faster file transfer rate in the future. For the ML approach - We observed a trade-off between speed and accuracy. So, exploring more pattern recognition ML models is wise than image processing models, which would increase the overall accuracy. The world of AI is still uncharted, so there could be many possibilities of an even better model out there for image classification, and we explored that a bit with our project.

REFERENCES & ACKNOWLEDGEMENT

We would like to sincerely thank our professor **RICHARD MARTIN** for constantly providing feedback and redirecting us whenever we faced a roadblock in our project. We would also like to thank him for structuring the entire computer network graduate course in a way that benefitted us greatly as we have learned all the fundamentals of computer networks.