Analyzing and Modeling Forensic Data

Tasmia Rahman, Rosemary Dabbs, Zachary Randall, Sara Mousavi

Abstract— **Having a large collection of images taken from decomposing corpses has the potential to be a substantial source of forensic information for many different research purposes if the collections are accessible and can be easily utilized. For these images to be easily used by users, we need to have a platform that facilitates access, browsing, labeling and statistical analysis on the collection. In addition to the platform, an automated labeling process will reduce the expenses of time consuming manual labeling and will result in a very rich dataset that can be used by law enforcements, forensic researches, teachers and students. The objective of this work is to improve the efficiency and accuracy of detecting forensic features in the images obtained from the Knoxville Body Farm by cleaning and preprocessing the data and adding features to the current model. The unique and difficult content of the forensic image data makes this objective challenging.**

I. Introduction and Motivation

In this project we are going to automate the labeling process of a forensic dataset including one million images taken from donated corpses to the body farm located in Knoxville Tennessee. To do so, we use a work-flow of web-based, image processing and deep learning approaches. Using web development techniques an online platform is build to facilitate the browsing and tagging process of the images. In the image processing stage, we do cleaning on the images including removing the redundant images and we do image augmentation to increase the size of the labeled images. We use deep learning to automate the feature detection and annotation in the unlabeled images. In this proposal we plan on improving the current online platform, generating augmented images and use image processing approaches to clean the data, overcome insufficient labeled images and improve the accuracy of the current model built using Mask-RCNN. Later we would like to see how the weather impacts the decomposition process of the bodies by conducting statistical analytics to find the weather pattern prior to specific features.

II. Task Outcomes

A. *Refine Online Platform for Image Tagging and Browsing:*

The Image Cloud Platform for Use In Tagging and Research on Decomposition (ICPUTRD) was created to help facilitate browsing through images by enabling multiple users to access and label the images and providing tag-based categories to search through images. The platform also has domain export functionality. The annotated images can be downloaded as a .csv file with JSON-like objects and then supplied to the Mask R-CNN model for training and testing. The web development tasks for this project involve improving the current online platform that facilitates the manual labeling process for gathering training data.

1) Issues: Some of the issues that need to be address include improving the tag search client view, adding a carousel component to the image browsing, troubleshooting the functionality of the "search tags" option on the tags drop-down menu, debugging the CSS file loading error, and adding user tracking to load function.

2) Platform Overview: This platform was made using a meanJS web application boilerplate. A mean stack consists of four main technologies, MongoDB, ExpressJS, AngularJS, and NodeJS. MongoDb is the database that the server assesses when a request is made. ExpressJS is the back-end web framework for NodeJS that provides many HTTP utility methods. NodeJS is a JavaScript runtime environment that enables easy scalability and latency of applications by streamlining requests instead of creating multiple threads of execution for each request. NodeJs is built to handle high throughput. AngularJS is a JavaScript front-end framework that enables dynamic HTML web pages. It allows you to decorate your HTML that connects with your JavaScript. AngularJS is very useful for large applications because it allows reuse of repetitive HTML code. MeanJS is good to use for web application development because it allows quick set up for full stack web application for faster development and deployment, predefines the structure of the files to keep code organized and easily testable, helps make building heavy data intensive web applications easy and scalable by integrating MongoDB into the stack, and uses one language for both front and back end making.

3) Approach: In order to accomplish these tasks Rosemary Dabbs, with support from the team leader, obtained access to the online platform from Dr. Mockus via secure shell connection. After gaining access, she took time to familiarize herself with the content of all the files and the general organization of the current platform. The following location were found to be important:

TABLE I
MeanJS File Location Summary

Location: mean.js/	Importance	Libraries/Tools
public/scr.js	Load function location	JS, XHR
modules/tags/server/models/tags.server.model.js	Schemas	MongoDB
modules/tags/server/controllers/tags.server.controller.js	Controller	AngularJS
modules/tags/views/search-tags.client.view.html	Views	HTML, Bootstrap

After becoming knowledgeable with the structure of the current platform, she began troubleshooting. This task was the most time consuming. Debugging the mean stack was difficult because she was unfamiliar with how all the technologies worked together. Knowing JavaScript and the locations of the important parts of the code helped, but this was her first attempt at working on a multipage and complex full stack web application. The majority of her time was spent changing one thing expecting a certain result and then getting errors that were difficult to solve. Despite these difficulties, she learned a great deal from this project.

4) What Was Learned: She has a better understanding of how the technologies interact. It is all about request and response objects. An HTTP request is simple a collection of binary data that is sent to the server from a client. When a request is made, a router receives the request object and finds the correct controller to handle the request. The controller accesses the database and builds a model to pass to a view. The view builds the HTML page template and the corresponding model into a response object that gets sent back to the client to see.

5) Results: The HTML was updated by adding bootstrap to make a more modern view of the Tag Search page. The results of these changes are visually explained in Figure 1 and Figure 2.

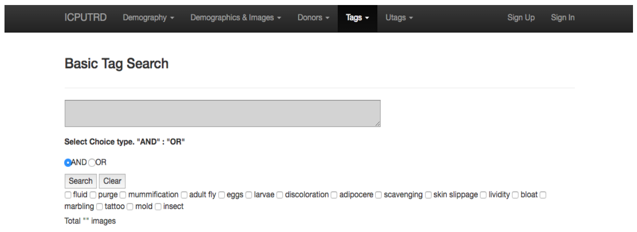

Fig. 1. Tag Search View Before Improvements

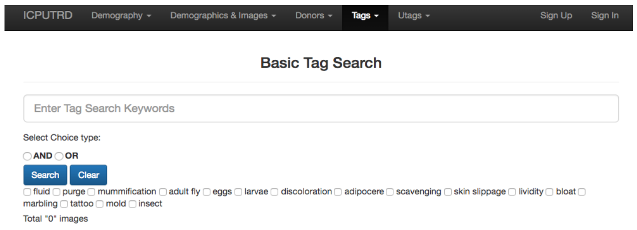

Fig. 2. Tag Search View After Improvements

B. Analytic tasks:

This part of the project involves analyzing the impact of weather on the likelihood of occurrence of forensic features. A mapping of Weather and forensic feature needs to be built. The map will provide the weather pattern prior to each feature and as a results will provide an understanding of the type of features that can happen in a specific season and weather. This results in eliminating the low accuracy resulted from searching for features in images that can not have those features. This id done through processing the data, analyzing the data and interpreting the results.

1) Processing Knoxville's weather data: The datasets used for this part of our analysis were: Image names, weather history of Knoxville and tagged dataset. Image names included information about donor's id and the date when this image was taken. The weather history starts from 2012-10-01(YYYY-MM-DD) and ends at 2018-06-19. It has the hourly weather for each day except for some missing days. There are some discrepancies in these datasets. The tagged dataset were for leg and mummification tags. These datasets had all the images that had leg or mummification tagged.

In the image names data set, there were inconsistencies in the format of the image names for example *UT14-12D_04_23_!2 (1)* where the part "!2" was incorrect. In addition to that, there were duplicate copies of images which made the dataset huge. However, only the unique ones were needed for the analysis. Hence, working with the large dataset was not time efficient.

Looking at the weather history datasets, we realized that most of the data have weather data available for hourly manner. However, there were some dates where some hourly data was missing. In fact, while doing the analysis, we realized that weather data did not have any data available for some days, even for some whole months. For example the weather dataset has no record available after 2014-08-08(YYYY-MM-DD) until 2015-06-12(YYYY-MM-DD).

Since the tagged datasets also had image names, the same inconsistencies within image names were found in those datasets as well.

The first task was to find out all the inconsistencies in the image name formats. My approach to this was to find out the most general pattern of image names by looking at the datasets. With the help of my python script, I analyzed that out of 1 million image names around 10,000 images had inconsistencies. The script narrowed down the inconsistencies and fixed most of them. After fixing them, I got rid of duplicate images which helped to minimize the observations number from 1M to around 26,000. This improved performance and runtime.

After fixing the inconsistencies, I wrote a script that will split the image names into donor id and date and generate a csv file with it which helped for further data analysis.

We also collected the extracted leg or mummification tagged data from the datasets of all tags and fixed the name inconsistencies as well.

2) Analyzing the weather data: The first task was to calculate ADD (Accumulated degree date) of temperature for each image and each donor. For each image this was accomplished by recording the current date of the image and minimum date(the very first image that was taken for this donor) of the that image. The script that was written for this analysis, generated all the in between dates of current date and minimum date and extracted weather history from the weather data for those dates. However, since the weather data missed observations for many hourly values as well as daily values, some ADDs came out 0 for some images. The way ADD was calculated was to take the average of the temperature for each day and then take the average of all in between days' average temperature.

Later we also generated ADD for humidity and wind speed for all the images.

Also, we wanted to compare the weather distribution of a specific feature against that of a general feature. For this, we chose mummification tag and leg tag since ideally leg label should be tagged in most of the images. Finally, I wrote a script for extracting prior weather history if given a image name.

3) Interpreting the results from weather data: We decided to see the weather distribution of leg tag and mummification tag for a specific donor. So we chose donor UT35-15D. We found out that for leg tag, the maximum and minimum date rage is of total 195 days and for mummification tag, the range is of total 123 days. The maximum date is same for both. So mummification started later which reflects our hypothesis that mummification date range should be a subset of leg data date range.

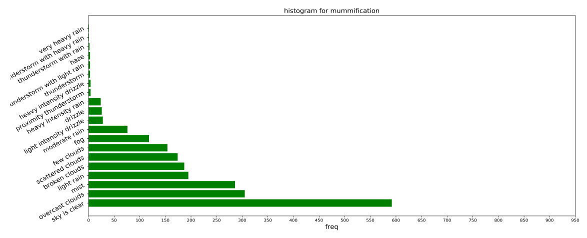

Fig. 3. Weather Distribution for Mummification

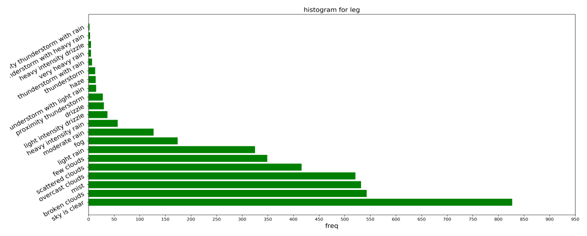

Fig. 4. Weather Distribution for Leg

We can see that for a donor mummification weather distribution is almost similar to that of leg data as their observation is almost during the same time period.

For two different donors we got the results plotted in Figure 5 and 6 :

Here we see some difference in weather distribution for example mummification has more clear sky. However since these are two different donors and their maximum and minimum dates are different, the difference in weather pattern could be due to different seasons of the bodies being exposed for these two donors.

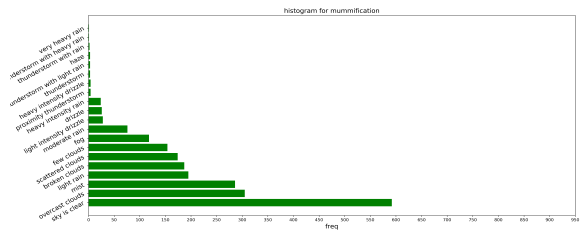

Fig. 5. Weather Distribution for Mummification

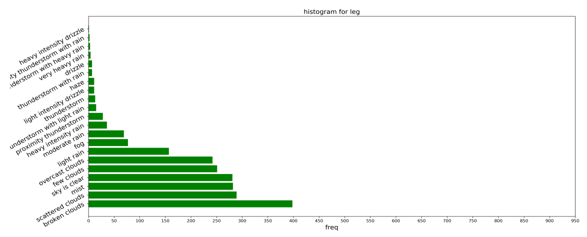

Fig. 6. Weather Distribution for Leg

Finally, the ADD calculation for temperature, humidity and wind speed for leg and mummification is as follows:

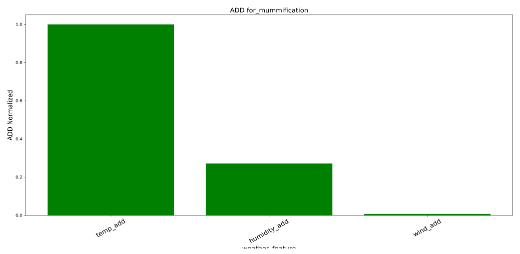

Fig. 7. ADD for Mummification

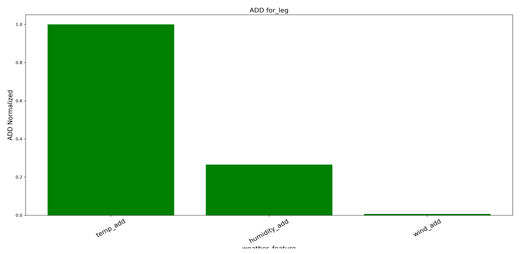

Fig. 8. ADD for leg

C. Image processing tasks:

The goals of the image processing tasks include matching crime scene images from Google or other sources to an image within our existing collection as well as separating the background from the foreground of an image. The Google images that would be used for matching would be from real crime scenes that have been reviewed by forensic experts. The expert information can then be associated with the matching image in our collection. This will help with identifying photos without the need to bring in an expert as well as provide more data for training the algorithm.The purpose of separating the background and foreground is to make it easier for the program to identify the crucial details required to correctly label a forensic image. Both of these goals are important for success of the application.

1) Image Hash: The method selected for removing duplicate images from the data set was image hashing. An image hash is a numerical value generated based on the pixel contents of an image. There are three methods of image hashing. Average hash is calculated by finding the difference between each pixel and an average gray scale value of the image. Difference hash is found by computing the difference between each pixel and its adjacent pixels. Perceptive hashing depends on the frequency of pixels rather than the color. We used average hashing for our testing. Images that have the same hash value are considered identical images. We first used image hashing to determine how many duplicate images were contained in a sample of 26,000 images. We found that in the 26,000 images, 2,300 had at least one duplicate image in the data set. We then ran our image hash comparisons on 989,838 images and determined that 63,468 images existed that had at least one duplicate image. It is crucial that duplicate images are removed from the data set because processing this type of data is expensive and resource intensive.

2) Background Removal: Objects in the background of an image can interfere with analysis and distort the results of the labeling process. Removing the background from images helps increase the accuracy of the system. The python library OpenCV and a web tool named LabelMe were both considered for this task. In the end, we chose to use OpenCV because it allowed for more options to automate the background removal process. The OpenCV background removal process works by using a mask to determine what is the foreground of the image and what is the background. The mask is created by finding large groups of similar pixels. When a large grouping is found, that grouping is determined to be the foreground. The pixels in that grouping are set to the same value. The rest of the pixels are set to be either 0 or 1. The mask is then multiplied with the original image. The result is the original image with the background removed. The mask generation still requires some manipulation when the background and foreground of images are extremely similar in color.

D. Literature Review:

The goal for the literature review is to have a better understanding of what the best practices are in the industry for improving deep learning environments for image classification. Rosemary looked at some of the different image segmentation methods in which the main goal is to classify objects in an image and define the objects boundaries. There are several image segmentation methods that can be used on images. Some of these include Thresholding, Edge Based, Region Based, Clustering, Watershed, PDE Based, and Artificial Neural Network (ANN) Based methods. Thresholding and Edge-based methods are useful for images with objects that are lighter than their background with high contrast. Region Based and Clustering methods are good if the image has regions that are similar that can be segmented. If this general approach was used, clustering would be more useful because it allows partial membership into regions. This makes this method better for more complex images. The Watershed method is topologically based and uses the images pixel gradients to define continuous boundaries. [3] After reviewing Various Image Segmentation Techniques: A Review and applying what we know about the type of images that are in our forensic data set, we predict that the Watershed Method would be most suitable.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has occurred since 2010. Over the years, the algorithms for object detection and classification has improved. According to the article A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN published in 2017, CNNs have started outperforming humans starting in 2015 for this specific challenge. [4] This improvement happened in just five years. It is difficult to image how these networks will be performing given another five years.

There are many Deep Learning Frameworks available to use. Knowing which one is the best to use can be a challenge. Jeff Hale, an experienced Data Scientist on Machine Learning, reported on the overall power of the different frameworks. Figure **??** was taken from from an article that Hale wrote about Deep Learning Frameworks in September 2018.

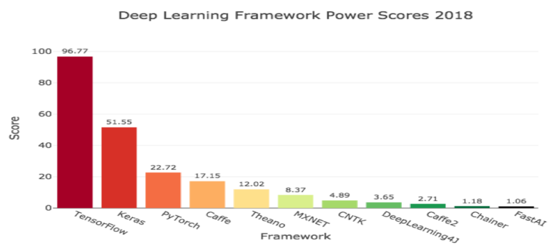

Fig. 9. A 2018 Comparison of Deep Learning Frameworks

E. Increasing the size of training data:

Currently we use Mask-RNNN to detect forensic features in the images. Mask-RCNN is the most accurate model for this purpose but it requires a large number of images to be trained on. For example COCO dataset that is used for this purpose is 163k images whereas we only have 3000 images labeled and used for the training and validation process. We believe that is the main reason of the current poor performance of Mask-RCNN on our data. In order to overcome this problem we use image augmentation.

1) Image augmentation: In order to provide a more divers range of images for the neural network and increase the size of the current training data set, we used image augmentation. There are many ways that one can use to augment images. Augmentation can be done by flipping, around x axis, y axis or x and y axis. Furthermore, images can be rotated, scaled or a combination of these. It is important to note that whatever changes that are made on the images, they have to be applied on the bounding boxes as well.

a) Flipping: Open CV provides built in functions for flipping images.

$$Void\ flip(InputArray\ src,\ OutputArray\ dest,\ int\ flipCode) \tag{1}$$

Where *src* is the image that is going to be flipped in an array format, *dest* the the flipped image in an array format. Depending of the value of the flipCode variables, the image will be flipped. Value 0 means flipping around the x axis and value 1, or any positive values, means flipping around y axis, value -1, or any negative values, means flipping around both axes. Let's consider row as the number of rows in src which would be the height of the image and col is the number of columns in src which would be width of the image. Each point $i,\ j$ in $dest$ is calculated using the following equation.

$$dest_{ij} = \begin{cases} src_{row-i-1,j} & if\ flipCode = 0 \\ src_{i,col-j-1} & if\ flipCode = 1 \\ src_{row-i-1,col-j-1} & if\ flipCode = -1 \end{cases} \tag{2}$$

In order to do the same thing on the bounding boxes we need to calculate the new coordinated of the start and end point of each bounding box as follows. Consider that row and col here are height and width

$$bbox\_flipped\_start_{ij} = \begin{cases} bbox_{i,row-j-h-1} & if\ flipCode = 0 \\ bbox_{col-i-w-1,j} & if\ flipCode = 1 \\ bbox_{col-i-w-1,row-j-h-1} & if\ flipCode = -1 \end{cases} \tag{3}$$

$$bbox\_flipped_end_{ij} = \begin{cases} bbox_{i,row-j+h-1} & if\ flipCode = 0 \\ bbox_{col-i+w-1,j} & if\ flipCode = 1 \\ bbox_{col-i+w-1,row-j+h-1} & if\ flipCode = -1 \end{cases} \tag{4}$$

where w and h are the width and height of the bounding box.

b) Scaling: Topically, scale for images means resizing images to different sizes. In this case we do not want to have different image size, but we want different scale of the images while they all have the same size. The means when we scale an images we zoom in in different areas and then resize the image to the original size. When the tags are generated from the database, mongo-db, each of them are stored in one line of the resulted csv file. Each line contains the user-id that has created the tag, the time of creation, the URL of the image in the ICPUTRD system, the coordinate of tag and the name of the image that the tag belongs to. The way that the scale function is implemented is based on the tags. A list of the scales is first created. For example $s = [0.3,\ 0.5,\ 0.8]$ means three different scales of the image are going to be created based on each tag. In another word, the information about tag x, which correspond to one line of the csv file, is given to the scale function. The function extracts/crops $s[i],\ i\ in\ len(s)$ piece of the image making sure that the bounding box of the current tag is included in the cropped piece. Then the cropped part will be resized to the size of the original image. The next step is to calculate the coordinate of the bounding box in the resized image. Although this part of the scaling function seems intuitive, there is one complications here. Since one image can contain multiple tags and bounding boxes, we need to make sure to check if any of the other tags belong to the current image will be in the cropped piece. If so, their coordinates need to be calculated and added to the csv file. As a result of out current implementation each tag will be augmented to 12 tags.

III. Limitations

The main limitations Rosemary was faced with for the online platform was restrictions on accessing the code and the npm crashing. After obtaining access to the code, she could not edit or debug the code because the files were read only. She first tried to solve this problem on her own by making a git server on her local machine with the mean stack, making edits to the code, and then pushing the changes back to the secure shell server, but when attempting to push, permission was denied. Since she could not debug the changes on her local machine because she did not have all the necessary packages installed to build and run the app, she was halted in her progress. She looked into using a meanJS docker to run the app on her local machine but was unable to figure this out in the limited amount of time. She approached Sara about this issue and together they determined that she was not a root user. Once the permissions were worked out, she was able to start making improvements, but then the npm started crashing when she tried to make edits on the code. Since she did not have access as a root user, she had to depend on Sara to restart the npm and redo permissions. This process was more time consuming than anticipated.

IV. Project management

Sara Mousavi was the team lead for this project. She was responsible for increasing the training data for the model with image augmentation. She was also be responsible for improving the current model. Rosemary Dabbs was responsible for improving the web-application interface and reviewing literature. Tasmia Rahman worked on the analytic tasks to add weather data to the collection of images. Zachary Randall worked on improving image processing and look into separating the background from the foreground. Each member of the team participated in the writing portions of the project and presentations.

V. Summary

A summary of the overall process for automatically labeling forensic images is outlined in Figure 10. The green boxes represent the areas that our team made effort toward improving the model.

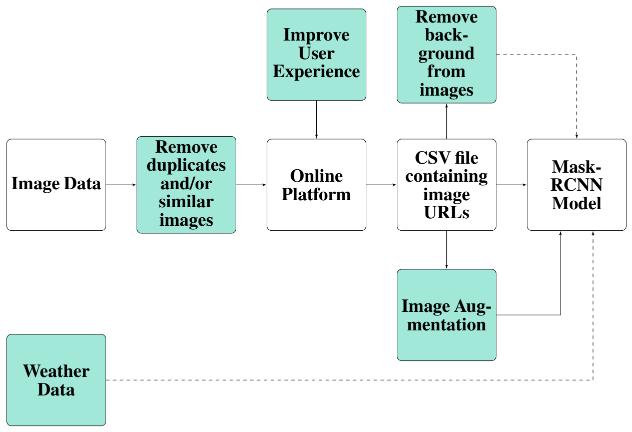
Improve User Experience
Remove back-ground from images
Image Data
Remove duplicates and/or similar images
Online Platform
CSV file containing image URLs
Mask-RCNN Model
Image Aug-mentation
Weather Data

Fig. 10. Summary Flow Chart

References

[1] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[2] Lin, T. Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B., & Belongie, S. J. (2017, July). Feature Pyramid Networks for Object Detection. In CVPR (Vol. 1, No. 2, p. 3).

[3] Kaur, D., Kaur, Y. (2014). Various image segmentation techniques: a review. International Journal of Computer Science and Mobile Computing, 3(5), 809-814.

[4] Parthasarathy, D. (2017). A brief history of CNNs in image segmentation: From R-CNN to Mask R-CNN. Computing Research Repository.

[5] Hale, Jeff. ”Deep Learning Framework Power Scores 2018”, 19 Sept. 2018, https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a