Allison DeCicco
Billy Miller
Lian Showl

The Average Joe Classifier
Design Homework Two

Description of Product

Our product is a classifier for someone who does not know/understand machine learning. The user will be able to upload their dataset, label the object(s) they want to classify, and then create a classifier without a large time commitment or an understanding of what is happening. The gap our product is filling is for someone who is not educated or familiar with topics such as machine learning and computer vision. Also, we want to create a classifier that does not need hundreds of images and require a large time commitment. Our user will still be able to create a classifier without the knowledge or the time commitment typically needed for this type of product.

Possible Users

The intended user for the Average Joe Classifier is anyone who feels that they could benefit from using computer vision, but may not have the skills to implement it themselves. This can include researchers, law enforcement, a suburban dad, and more. The users will interact with the product by navigating to our website on their personal computer. From there they will be able to follow our process to obtain a trained classifier.

Possible Use Cases

1. Law Enforcement: This is a possible use case because law enforcement spends hours a day analyzing video surveillance footage to solve crimes. If they could train our system on images of suspects or victims they are looking for in the footage, then they wouldn't have to watch the footage themselves. This would save them hours doing a task that is very time consuming.
2. Biology Research: Camera traps can be used to see the daily lives of animals. The Average Joe Classifier is a better solution over other classifiers because the researcher can create it based off of its own footage. Many researchers have classifiers made for them to meet their needs, but with the Average Joe Classifier they can create it themselves.
3. The everyday dad: For example, if a dad has a bird feeder and wants to see what bird food is the most popular he could create a classifier for birds and track which ones come depending on the food. This is one example of how an average person would utilize our classifier.

Components:

The components in the User Interface container are the Python, JavaScript, and the YOLO command line interface. For the JavaScript to interact with Python, it is being done as a command line interface. The Python script is written so the JavaScript will run it with certain arguments to get the results it needs. The YOLO CLI is written in the same way as the Python, so the Python script then calls the YOLO CLI for the actual training and testing. The Python script is also responsible for data manipulation. It will communicate with the MongoDB database using the PyMongo Python package. The script will need to load the images into memory for training and testing and then store its results back to the database. The MongoDB database is a nonrelational database.

The UI is interacting with our database by first creating a connection with the database. It then makes requests based on what the user inputs. One example is it creates a request to store the image. When it does this, the program creates a file path. The image is being split up so that it can be properly stored in the database. The UI code is an ejs file. The ejs allows us to have JavasScript embedded into our HTML. Another request example is in the ejs file. There is a “form action” where it submits a POST request. The request is submitted when a picture’s delete button is clicked. The picture’s id is submitted and then deleted from the database based on its ID. Both the UI and the database are deployed on the server. The user can then access the product by navigating to the appropriate URL using their web browser.

The annotation tool uses Annotorious to use its event handling functions. Since the user draws bounding boxes on the image, the event handling is necessary to receive the coordinates of the desired selection in real-time. The functions are called within the UI to work together to show the user's selections. The coordinates are stored in the database by the JavaScript. The multiple bounding boxes allow the user to label many instances of the object in the same file (as needed). The JavaScript will then run the Python script, passing “train” and the list of image IDs as parameters. The image IDs are the lookup keys for the training images in the database.

Interfaces:

The three components which will be interacting with each other are the JavaScript, Python, and MongoDB. Both the Python and JavaScript will connect to the MongoDB using a regular TCP/IP socket. They will then use MongoDB’s APIs for Python and Javascript respectively to interact with the database. The API calls to interact with the database are designed to mimic the function you would use when using the MongoDB command line interface. For example, we are using functions like insert() and get() to insert and read from the database. The Python script is designed to be used as a command line interface so it can be called by the JavaScript. The two functions the Python script has are train and test. Test will later be changed to retrain once our full pipeline is working. To use these functions the JavaScript uses the parameters train or test and then the list of MongoDB IDs of the images to train/test on. The Python does not send any information back to the JavaScript, it simply stores the results of the training or testing in the database in a predefined location so the JavaScript can then display them.

Critical Formats and Structures:

We use the gridfs node library to be able to stream photos from our UI to the database. This allows us to communicate directly with the database. We also are storing the coordinates of what was labeled in the images.

To communicate between the HTML and JavaScript of the UI, we send POST requests with express, which is a node library. Express parses incoming requests with JSON payloads and is based on a body-parser. This allows the pictures to be uploaded and deleted and the coordinates to be uploaded to and from the database.

Javascript is used for the client side of the server. It is an object oriented scripting language. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects at run time, as opposed to the syntactic class definitions common in compiled languages like C++ and Java.

Core Algorithm:

The core algorithm we will be using is the You Only Look Once (YOLO) computer vision algorithm. We chose this algorithm because it is the fastest computer vision algorithm available. This allows our product's latency to be as low as possible while the user training their model. YOLO is able to achieve such impressive speeds by using a different method than other computer vision algorithms. Most algorithms require several passes through a neural network in order to get a classification for an image, but YOLO is able to do it in just one pass. The YOLO algorithm is run in the Python script, which is called by the JavaScript when it is time for training or testing.

Primary Data Structures:

The primary data structure we are using is a list. The data is being stored in a list because YOLO, our core algorithm, requires the data to be stored that way. A list is an ordered sequence of elements stored together which can be changed. Each element in a list is called an item. A list is beneficial because each item can be used individually or all of the items can be used together.

Timeline:

Our goal for the timeline of our project is at the 40% mark we will have a full working pipeline, which means a user will submit an image, which then will be stored in the database. The user can label the image and the coordinates of the box will also be stored in the database. The system will train on a model. Then there will be a separate page that uses the model previously created to show the results. For 70%, we will be able to upload multiple images to be trained on and edit the bounding boxes on the images that we trained on after the first set of testing is done. Another milestone between 70% and 100% would training on the new set of labels. The user will be able to give their feedback if the previous round of training was successful or not. At 100%, the user will be able to train on the images as many times as they choose to and receive the results. We will test our different milestones with test images of birds. We will also use videos from youtube and parse those as test data. Most of the different modules for our project are will already be integrated, since in development, it will be difficult to do only one part of the project independently.