C964 Computer Science Capstone: Clothesifier

Andrew Staus

Western Governors University

010312389

Author Note

Thank you to all the people that supported me along the way—my professors, mentors, coworkers, employer, family, and especially my wife. I could not have done it without you.

Table of Contents

Abstract

The Computer Science Capstone is the final assignment before completing the Bachelor of Science in Computer Science degree program at Western Governors University. It requires a student to use the skills they gained through the course of the degree to develop an application that utilizes a machine learning algorithm to solve a problem for a business.

Clothesifier is the name of the application created for the fictitious e-commerce company Clothier, which allows a user to upload an image of a piece of clothing through a REST API and receive a category.

- The completed application can be accessed through this link.
- The EDA and model training notebook can be accessed at this link.
- The source code can be accessed on the GitHub repository at this link.

C964 Computer Science Capstone: Clothesifier

Part A: Project Proposal for Business Executives

Letter of Transmittal

December 1, 2022

William Hamilton, CTO
Clothier Company
2130 21 Street NE
Calgary, AB T2T 3P8

Dear Mr. Hamilton,

Clothier is a rapidly growing e-commerce business, serving tens of thousands of customers monthly with a wide selection of high-quality clothing items. As the company and its customer base have grown, so has the inventory. However, customers have reported difficulty finding the things they are interested in purchasing, leading to a high abandonment rate and lost sales.

The Clothier website has the functionality to display items by category. Still, it is currently unused due to the lack of categorization for the existing and growing inventory, and manually labeling each item would be cost-prohibitive. To address this issue, I propose implementing a machine learning system to automatically categorize all items in the Clothier inventory and, on an ongoing basis, for newly added items. The deployed system will provide an API that can be integrated directly into Clothier's inventory system, providing seamless usage for business users.

Categorizing clothing will improve customer experience and lead to more conversion as customers quickly find the items they seek. Categorization will also benefit the product and

marketing teams by providing the ability to generate better analyze which types of products are popular and which are not.

The development and deployment of the machine learning system will require a time investment of seventy-five working hours for one developer at a rate of one hundred dollars per hour, totaling seven thousand five hundred dollars. This investment represents a cost-effective solution for improving the Clothier business's inventory management and customer experience. The service development will use software development tools and libraries available at no cost and with no licensing fees, resulting in no additional expenses beyond the time investment for the developer. The service will be deployed on a serverless cloud platform, eliminating the need for Clothier to manage and maintain infrastructure, reducing costs, and improving the system's overall efficiency. The serverless platform does incur a usage fee of twenty-five cents per one million requests. However, the first one-million requests are free of charge, which exceeds Clothier's current needs. As a result, usage and maintenance costs for the foreseeable future will effectively be zero.

Overall, the development and deployment of the machine learning system will provide significant benefits for the Clothier business and its customers, with minimal costs and no maintenance required.

My education, culminating in a Bachelor of Computer Science and a decade of industry experience in software development, project management, and data science, has provided me with the skills to successfully develop and implement the automated categorization system for the Clothier e-commerce platform.

Sincerely,

Project Recommendation

Problem Summary. The Clothier e-commerce business is facing a significant challenge in inventory management. Users have reported difficulty navigating the website due to a lack of categorization, leading to high rates of abandonment and lost sales. The e-commerce platform Clothier uses can sort items by category. However, it is not being utilized due to the lack of category labels for the existing inventory.

Manually reviewing and categorizing each item in the inventory would be cost and time prohibitive due to the large volume of items. An automated system would provide a quick and accurate solution, allowing the entire inventory to be categorized in minutes rather than the months it would take humans to perform the same task.

By resolving the categorization problem, the Clothier business can improve the customer experience and provide business stakeholders with the ability to analyze relevant information.

Application Benefits. Implementing the Clothesifier automated categorization system will bring several benefits to the Clothier e-commerce business.

First, the categorization of items on an e-commerce site is an expected feature by customers. Research shows that 83% of customers browsing the Clothier website leave without purchasing an item (Doe, 2022), citing difficulties navigating the site and finding the desired products. By integrating the Clothesifier system directly into the inventory system, customers can search for products by type, leading to higher conversions and a better user experience.

Additionally, business stakeholders in the sales, marketing, and product teams have reported difficulty in understanding customer trends due to a lack of category information. The automated categorization system will provide these users with better insights, enabling them to create more targeted marketing campaigns and improve the overall customer experience.

Application Description. The Clothesifier automated categorization system will be able to accept an image of an inventory item and return the category for that item, along with a confidence metric indicating the model's level of certainty about the prediction. The system will be deployed as a web API, allowing seamless integration with the existing Clothier e-commerce systems, enabling Clothier to easily categorize its inventory and improve efficiency and accuracy in its business operations.

Data Description. Training the Clothesifier system will result in the ability to recognize the difference between clothing items using a freely available dataset called Fashion MNIST. The dataset consists of seventy-thousand examples of 28 by 28 images of clothing articles, along with categorical labels for each image. The data is of high quality and does not have any outliers. However, the pictures of the Clothier inventory may be of a different standard, which could limit the model's performance. The use of data augmentation practices will address the potential limitation, ensuring the dataset is more representative of real-world data, thus improving the model's ability to perform well in any circumstance.

Objectives and Hypothesis. Upon completion of this project, the Clothesifier system will be able to accept an image from the Clothier inventory and return the category for that item with 80% accuracy. In addition, the model's confidence levels for the classification will be made available for examination by business users.

The hypothesis is that a model trained on the dataset can generalize to the Clothier data and perform accurate image classification, allowing Clothier to categorize and manage their inventory, improving efficiency and accuracy in their business operations.

Methodology. Following a waterfall methodology, the development will proceed with a linear, sequential approach. This methodology divides the development process into distinct phases, each building upon the earlier one. The phases to be followed are:

- Requirements – Stakeholder engagement to gather the final system specifications. Integration specialists with knowledge of the Clothier e-commerce platform will supply input.
- Design – Interface, deployment architecture, model, and data will be detailed to ensure that all systems integrate.
- Implementation – The writing of code and the deployment and integration of systems.
- Verification – Evaluation that the system behaves correctly with valid and invalid input.
- Maintenance – Any bugs, issues, or additional features missed during the verification phase are assessed and resolved.

Funding Requirements. Implementation of the application will use a serverless architecture provided by AWS Lambda, eliminating the need for infrastructure management and maintenance. While there is a usage fee of $0.25 per one-million requests, the first one-million requests will be free. Based on the size of the Clothier inventory, it is doubtful that any costs will be associated with running the service.

Tools and software libraries that will be used are available at no cost and do not have associated licensing fees. The estimated development time for this project is 75 working hours for one software developer at a rate of $100 per hour, for a total cost of $7,500.

Data Precautions. The data to be used in training the model is publicly available and does not contain sensitive or proprietary information. In addition, the nature of the deployed system does not require access to a dataset, so there will not be any privacy concerns if the data becomes sensitive in the future.

Developer Expertise. We propose to assign the project to a developer with a decade of experience in developing and deploying software, managing projects, and data science. This developer holds a Bachelor of Science in Computer Science, which complements their practical experience. This individual is an excellent candidate for developing the system, given their combination of education and experience.

Part B: Project Proposal

Problem Statement

The Clothier e-commerce platform is experiencing low conversion rates and suboptimal customer experiences due to its lack of a product categorization system. Implementing an automated classification system deployed as a REST API to address this issue is recommended. This system will accept images of clothing as input and accurately return the corresponding category for the item.

Customer Summary

The Clothier e-commerce platform serves as a retail destination for customers looking to purchase clothing and accessories. However, the platform's lack of a categorization system leads to poor customer experiences and low conversion rates. Our proposed application – a data product in the form of a REST API endpoint – aims to address this problem by supplying a reliable and accurate method for categorizing both new and existing inventory on the platform. The API will be easily integrated into the platform's inventory pipeline, automatically categorizing newly added items and streamlining the process for the platform's managers. Additionally, the API can be used by Clothier's analysis team to accurately classify the current inventory, improving the overall user experience and conversion rates on the platform.

Existing System Analysis

The Clothier e-commerce platform needs item categorization to support higher conversion rates and sales. A categorization system must be implemented to take advantage of the systems existing API integration capability. Our proposed solution, a data product in the form of a REST API endpoint, aims to address this problem by supplying an automated method for categorizing both new and existing inventory on the platform. The API will be easily integrated

into the platform's existing infrastructure, allowing for the seamless implementation of a reliable categorization system and improving the overall user experience on the platform.

Data

The proposed application will use the Fashion MNIST dataset – a publicly available and free source – to train the classification model. This dataset consists of sixty-thousand training images and ten-thousand test images of clothing and accessories, each labeled with the corresponding category.

Any outliers or incomplete data within the Fashion MNIST dataset will be found and managed during the preprocessing phase. Outliers may be removed from the dataset or treated suitably depending on the specific circumstances and impact on model performance.

Project Methodology

Throughout the development of this project, we will follow the Waterfall project management methodology. The methodology involves a sequential and linear progression through the following phases: requirements gathering, design, implementation, testing, and maintenance.

The methodology is a cost-effective and efficient approach for small projects and those with stable requirements. It allows for a clear and structured development process, with each phase building upon the earlier one.

During the requirements gathering phase, we will engage with the Clothier IT team to understand their needs and requirements for the Clothesifier application, including discussions about the desired features and functionality and any specific requirements or constraints.

The design phase will follow requirements gathering, providing decisions on the application's architecture, technologies, frameworks, and overall structure and design.

Next, we will move into the implementation phase, setting up the programming environment and coding the application. We will follow industry best practices for coding and testing to ensure that the application is of high quality and meets the requirements outlined in the earlier phases.

Once the implementation is complete, we will move into the testing phase, where we will verify the proper behavior of the application across both intended and unintended inputs. We will allow for extra time in this phase to resolve any bugs.

Finally, we will move into the maintenance phase, addressing any remaining bugs or regulatory concerns and ensuring that the application runs optimally.

Project Outcomes

Upon completion of the project, the API endpoint will be available for use by the customer. To demonstrate the functionality of the API, a provided browser-based GUI will allow a user to upload a single image and evaluate the results. The demonstration GUI will include documentation on its proper usage.

The API endpoint will adhere to the constraints of the REST architecture, which is well documented and understood, alleviating the need for documentation on its proper use. The customer will have access to the API endpoint to integrate its functionality into their systems and applications.

Implementation Plan

The proposed approach for implementing this project is a top-down approach, with each step completed in order. This approach ensures functionality verification after each step and that all modules can integrate into a viable product. The steps are as follows:

1. Conduct exploratory data analysis (EDA) using the Pandas, Matplotlib, Sci-Kit Learn, and Plotly Python libraries to understand the characteristics of the Fashion MNIST dataset and decide the best approach for developing the machine learning model.
2. Reshape the data using the Numpy library and augment it with additional examples using Numpy and Sci-Kit Learn.
3. Compile a convolutional neural network (CNN) model using the Keras TensorFlow interface.
4. Train the CNN on the augmented Fashion MNIST dataset for 100 epochs to achieve sufficient accuracy for the requirements.
5. Analyze model performance using Matplotlib and Sci-Kit Learn to verify that accuracy meets the requirements.
6. Export the model in a TensorFlow Lite (tflite) format.
7. Write the API code using the FastAPI, Pydantic, Numpy, Pillow, and TFLite_Runtime Python libraries. Use the Mangum library to wrap the app in an event handler for compatibility with AWS Lambda.
8. Package and deploy the API code as a function on AWS Lambda.
9. Compile the required libraries for the API to run on an Amazon Linux environment, package them, and deploy them as a layer on AWS Lambda.
10. Develop a basic frontend for API functionality testing using HTML5, CSS, and JavaScript. Use the jQuery library to access the API hosted on the AWS Lambda site, the Bootstrap CSS and JavaScript libraries to supply a responsive grid interface, and the Chart.JS library to visualize the results.
11. Store the project on a GitHub repository and deploy the webpage using GitHub Sites.

Evaluation Plan

To ensure the successful and accurate functioning of the Clothesifier application, we will implement a thorough evaluation plan to confirm the performance of each module at multiple stages throughout the development process.

Model Module Testing. Before training, ten-thousand clothing images will be reserved for validation. After training the model on the remaining sixty-thousand photos, we will assess its accuracy on the validation set to determine its performance on items the model has not seen before. A successful test will have the model score over 80% accuracy on the validation set.

API Module Testing. We will evaluate the API to confirm that it responds with a proper classification when provided with a valid Base64 image file at the correct endpoint. In case of exceptions, the API should return a message containing the exception information for debugging purposes.

Webpage Module Testing. The webpage will be evaluated to ensure that it accepts an image file upload and displays the classification results from the API to the user. Appropriate error alerts should be displayed to the user if exceptions arise during the classification process.

System Testing. In addition to evaluating the individual modules, we will also conduct system testing to ensure that the entire Clothesifier application functions correctly and seamlessly. These tests will assess the integration and communication between the different modules and the overall user experience and performance.

Post-deployment Evaluation. After the Clothesifier application is deployed and used, we will conduct ongoing evaluations to assess its performance and effectiveness, including monitoring the accuracy of the classification results and tracking any issues or errors that may arise in the API or webpage. We will also solicit feedback from users of the application to gather

insights on its usability and user experience. Based on the results of these evaluations, we will make any necessary updates or improvements to the Clothesifier application to ensure its continued success and value for the Clothier e-commerce platform.

Resources and Costs

The deployment and operation of the Clothesifier API will have an initial cost of zero dollars, as it will be hosted on the serverless cloud platform AWS Lambda. AWS Lambda offers a free tier for the first one million monthly requests, with a tiered pricing system for additional requests.

The Clothesifier application is lightweight and has a normal compute duration of fewer than 0.002 GB-seconds (see Table 1 for pricing), using the minimum memory tier of 128 MB (see Table 2 for pricing). Given the current size of the Clothier inventory, usage will remain within the free tier of one million requests per month. However, if the inventory size increases and requests exceed the free tier, the cost for each additional one million classifications is expected to be less than $0.25.

Table 1

AWS Lambda Compute Pricing

Usage Cost Per: Billion GB-seconds / month	Duration (GB-second)	1M requests
First 6	$0.0000166667	$0.20
Next 9	$0.000015	$0.20
Over 15	$0.0000133334	$0.20

Table 2

AWS Lambda Memory Pricing

Memory (MB)	Price per 1ms
128	$0.0000000021
512	$0.0000000083
1024	$0.0000000167

Timeline and Milestones

Development and deployment are estimated to require 75 hours over the course of 2 weeks. This timeline results from the constraints of the assigned software engineer's schedule. The project's completion depends on the milestones described below (see Table 3).

Table 3

Timeline and Milestones

Mile-stone	Prere-quisites	Activity	Resource Assigned	Hours	Start	End
1	-	Requirements approval	Project Manager	8	2022-12-05	2022-12-05
2	1	Architecture Design	Software Engineer	8	2022-12-06	2022-12-06
3	1	Development Environment configuration	Software Engineer	2	2022-12-07	2022-12-07
4	3	Exploratory Data Analysis	Software Engineer	3	2022-12-07	2022-12-07
5	4	Model Creation	Software Engineer	12*	2022-12-07	2022-12-08
6	5	Model Module Testing	Software Engineer	1	2022-12-08	2022-12-08
7	2	API Development	Software Engineer	8	2022-12-08	2022-12-09
8	6, 7	API Deployment	Software Engineer	2	2022-12-09	2022-12-09
9	8	API Module Testing	Software Engineer	1	2022-12-12	2022-12-12
10	2	Frontend Website Creation	Software Engineer	8	2022-12-12	2022-12-13
11	10	Frontend Website Deployment	Software Engineer	2	2022-12-13	2022-12-13
12	11	Website Module Testing	Software Engineer	1	2022-12-13	2022-12-13
13	8, 12	Module Integration	Software Engineer	1	2022-12-13	2022-12-13
14	13	System Testing	Quality Assurance	20	2022-12-14	2022-12-16
15	14	Final project delivery	Software Engineer	8	2022-12-19	2022-12-19

**Model creation will only require 2 hours of developer time; the remaining 10 hours consist of training time, performed autonomously outside business* hours.

Part C: Application

Application Files

We will organize the application files into three folders relating to the development modules, (a) Model Training; (b) API; (c) Webpage; (see Table 4).

Table 4

List of Application Artifacts

Directory/Filename	*Description*
\clothesifier	
API	*Files related to the back-end API*
\layers*	*AWS Linux compiled libraries for main.py*
aws-lambda-fuction.zip	*Packaged API code and model for AWS Lambda*
main.py	*API Source Code*
model.tflite	*The Frozen CNN Model in .tflite format*
requirements.txt	*Required libraries for API code*
model training	*Files related to the training of the model*
\data	*Data for training and testing*
boot.jpg	*An image of a boot*
Fashion MNIST.zip	*The Fashion MNIST dataset*
pullover.jpg	*An image of a sweater*
\model*	*The CNN model in .pb format*
notebook.ipynb	*The notebook for EDA and Model training*
requirements.txt	*The required libraries for the notebook*
webpage	*Files to run the frontend webpage*
/test images/*	*Images used for evaluating the webpage functionality*
index.html	*The homepage of the website*
scripts.js	*The JavaScript*
shirt.svg	*A vector image of a shirt*
styles.css	*The style sheet*

Part D: Post-implementation Report

Business Vision

The Clothier e-commerce site had a large and diverse catalog of clothing items for sale but no system for categorizing them, which made it difficult for customers to find specific items and for the company to manage and organize its inventory effectively. We developed the Clothesifier system to address this problem.

Using the Python programming language and open-source libraries, a convolutional neural network was trained in a Jupyter notebook that could accurately assign categories to clothing items (see Figure 1).

Figure 1

A representation of the CNN model

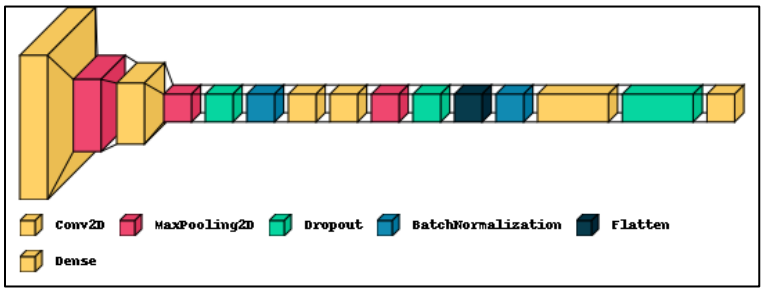


Image retrieved from the training Jupyter notebook

The model was integrated into a REST API service created using the Python programing language and open-source libraries. The module was deployed on a serverless AWS Lambda platform with an x86 instruction set, running Amazon Linux for integration into other systems (see Figure 2).

Figure 2

The Welcome Message on the Root Endpoint of the API

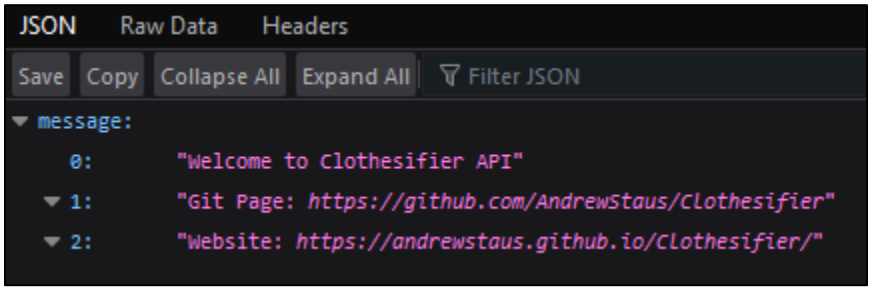


Image captured from the [API root endpoint.]

While the API was the primary deliverable for the customer, a website was developed using HTML5, CSS, and JavaScript languages, along with open-source libraries. The website allows users to evaluate the system's functionality effectively by uploading an image of clothing and receiving the predicted category in return.

To test the Clothesifier system, users can visit the website and upload an image of a piece of clothing. The system will then process the image and return the predicted category for the item (see Figure 3).

Figure 3

Clothesifier Classifying an Image of a Pullover

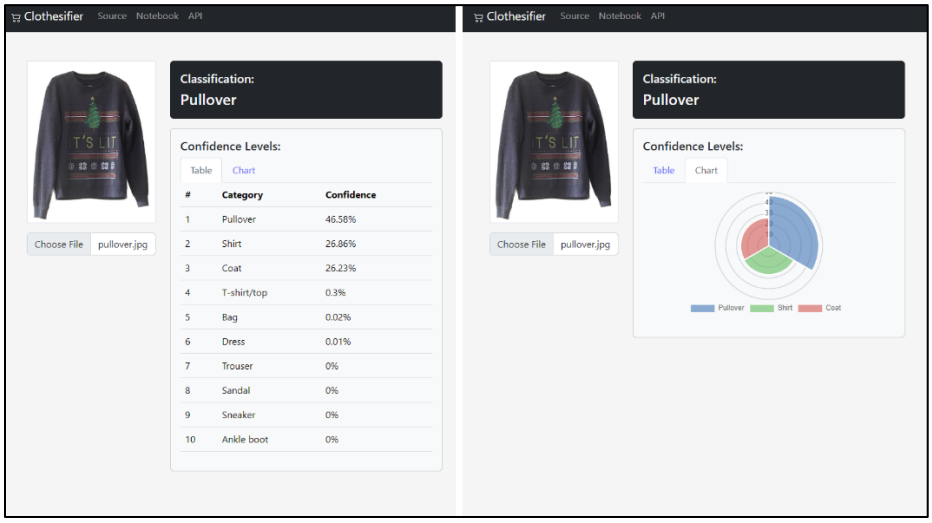


Image captured from the [Clothesifier webpage.]

Datasets

The convolutional neural network (CNN) used in the application was trained using the Fashion MNIST dataset, which was sourced from Kaggle.com. This dataset is provided in the project's GitHub repository in the "\model training\data\Fashion MNIST.zip" file and is available on Kaggle. The dataset consists of two CSV files, "fashion-mnist_train.csv" and "fashion-mnist_test.csv," containing image data and labels for sixty-thousand training examples and ten-thousand validation examples, respectively. These files are considered the raw datasets for the CNN (see Table 5 and Figure 4).

Table 5

Example of Raw Data from Fashion-mnist_train.csv

label	pixel1	pixel2	pixel3	…	pixel784
2	0	0	0	…	0
9	0	0	0	…	0
6	0	0	0	…	0
…	…	…	…	…	…
7	0	0	0	…	0

Note: the images have a flat background with values of zero.

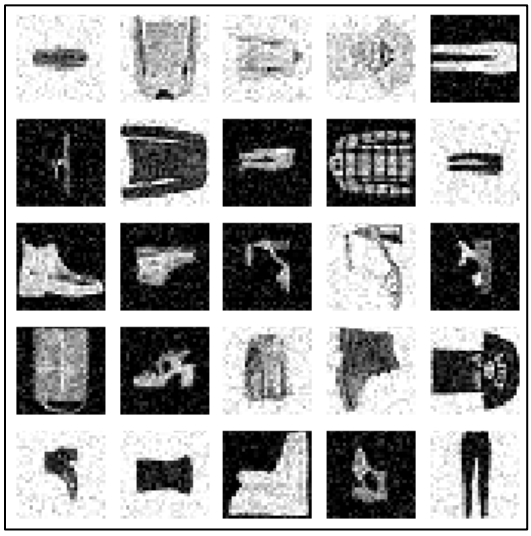
Figure 4

Images Representations Extracted from the Raw Data

To further enrich the dataset, data augmentation was performed using multiple Python libraries in a Jupyter notebook (see Figure 5). The augmented dataset was then used to train the CNN model. The final application does require access to the dataset during regular operation, as the model is deployed in a trained state on the API.

Figure 5

Images Representations Extracted from the Augmented Data



Data Product Code

The application is divided into three distinct modules that work together to classify items accurately for the user. The separation of these modules helps to ensure that each one has a clear and specific responsibility, which helps to make the application more organized, efficient, and maintainable.

The first module was built using a Jupyter notebook running a Python kernel utilizing the TensorFlow python library. It is responsible for training the convolutional neural network (CNN) model to classify items. This module involves preparing and processing the data, defining the

model architecture, and training the model. Once the model is trained, it is frozen and serialized so that the other modules can use it.

The second module is the API, which was developed using the FastAPI Python library and provides an interface for interacting with the trained model. It accepts raw data in image files, processes it for the model, runs the model on the data, and returns the classification result to the user. The API module is designed to be lightweight and efficient, using a model interpreter (tflite_runtime) to load the trained model and handle requests quickly.

The third module is the webpage, developed using HTML, CSS, and JavaScript, which provides a user-friendly interface for interacting with the API. It allows users to upload images and receive results without knowing how to use the API directly. The webpage is designed to be intuitive and easy to use, making it accessible to a wide range of stakeholders who may not have technical expertise.

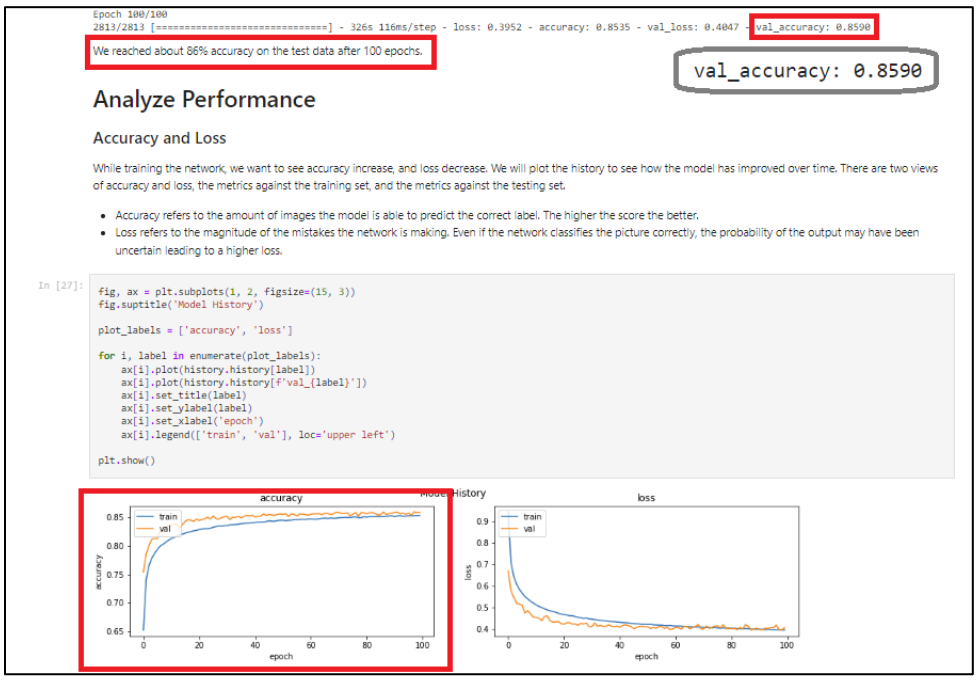
Together, these three modules work seamlessly to deliver accurate classification results to the user. The logical separation of responsibilities among the modules helps to make the application more organized, efficient, and maintainable.

Objective and Hypothesis Verification

The Clothesifier system's objective is to accept an image from the Clothier inventory and return the category for that item with 80% accuracy. The objective was validated by testing the CNN model against the verification data reserved for testing, reaching close to 86% accuracy, as demonstrated in the training notebook (see Figure 6).

Figure 6

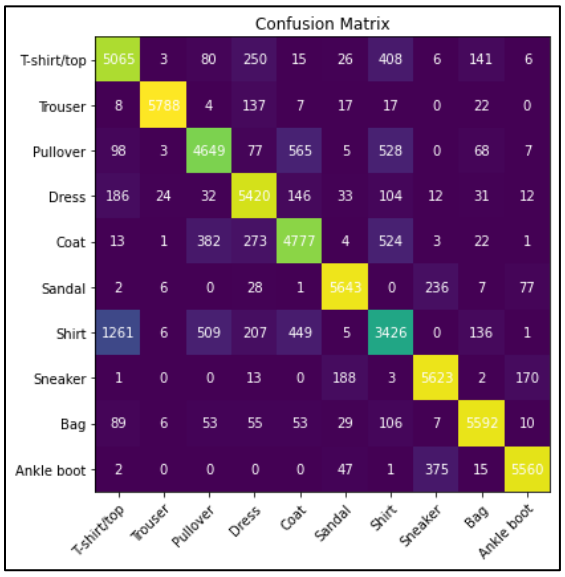
Model Accuracy Score on the Validation Set After Final Training Epoch



Further validation was performed using a confusion matrix to understand which classes were responsible for the misclassification (see Figure 7).

Figure 7

Confusion Matrix of Successes and Errors



The hypothesis was that a model could generalize to the Clothier data and perform accurate image classification. The hypothesis was validated as correct by the model accurately predicting curated items from the Clothier inventory, as demonstrated in the notebook (see Figure 8).

Figure 8

Classification of an Item from the Clothier Inventory

Effective Visualization and Reporting

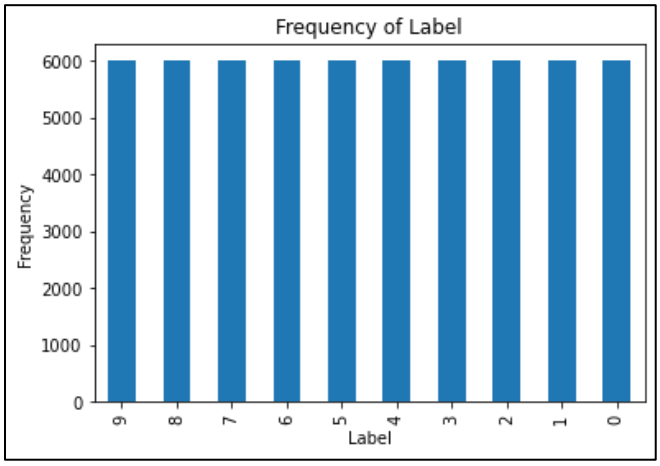
Visualization and reporting played a crucial role in various stages of the project. They were used to explore the data, train the model, and present the results to the user on our website.

Exploratory Data Analysis. Since the data used in our convolutional neural network (CNN) was made up of flattened image files, heatmaps were used to plot these pixel values to understand better what the model was being trained on (see Figure 4).

To ensure that our training set was balanced, meaning it had equal amounts of each label, we used a bar chart to visualize the frequency of each label in the training data (see Figure 9).

Figure 9

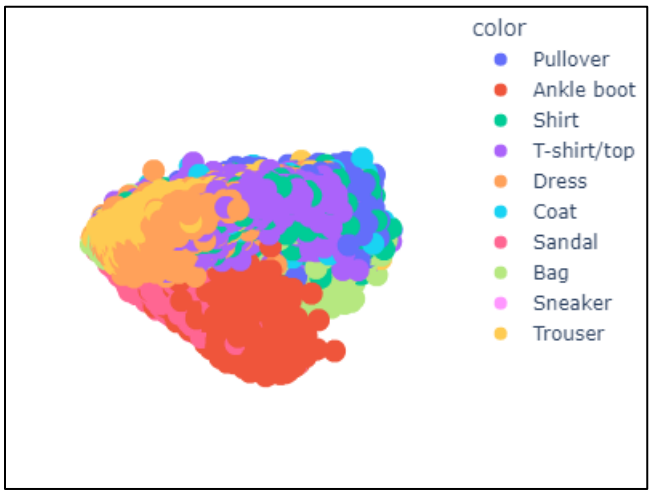
Bar Chart Showing Frequency of Labels



To better understand the similarities between the images, principal component analysis (PCA) was used to reduce the dimensionality of the data to three dimensions. The results were then plotted using a three-dimensional scatter plot (see Figure 10).

Figure 10

Three-Dimensional Scatter Plot of Fashion MNIST Principal Components

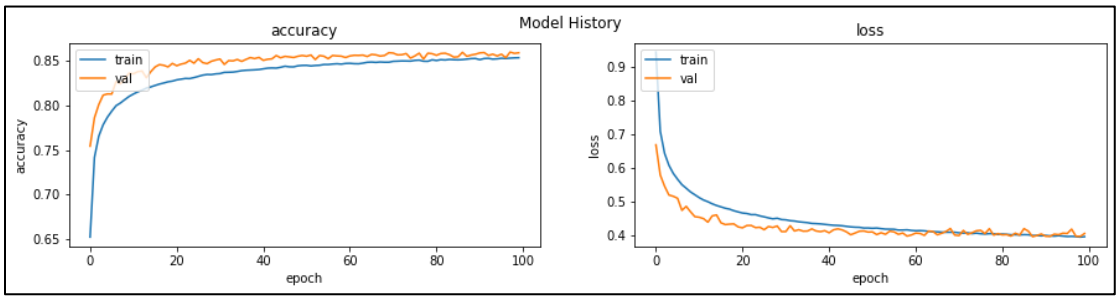


Model Training. Before training the model, the dataset was augmented to create more variation in the images used to train the model. The augmented data was visualized using heatmaps for inspection (see Figure 5). We also visually represented the model's architecture for review (see Figure 1).

After training, the training history was plotted using line graphs to show the model's evolution of accuracy and loss over the training period (see Figure 11). These graphs helped us analyze the network's performance and determine if the model has overfitted or needs more training. A confusion matrix was also used to analyze misclassifications and identify which labels the model had difficulty differentiating between (see Figure 7).

Figure 11

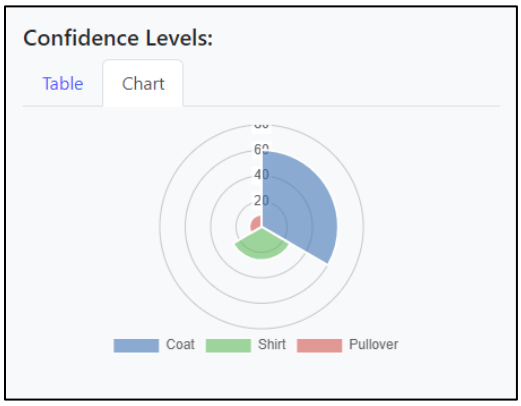
Accuracy and Loss



User Interface. A radar chart was used (see Figure 12) to provide the user with a visual representation of the confidence levels.

Figure 12

A Radar Chart



Accuracy Analysis

As described in the Objective and Hypothesis Verification section, accuracy was assessed based on the performance of the CNN to classify an image of clothing correctly. Model performance was evaluated during the Model Module Testing phase in three ways:

Accuracy Against the Validation Set. The Fashion MNIST data was split such that ten-thousand images were set aside for validation. The model was not trained on these images, so they are appropriate for assessing its accuracy. After the final training epoch, the model performed with close to 86% accuracy on the validation set (see Figure 6).

Misclassification Analysis. The validation data generated a confusion matrix to visualize where the model was making classification mistakes (see Figure 7). The confusion matrix showed that errors were happing between similar items. For example, the largest misclassifications were happening between images of Shirts and T-shirts/tops. With this insight, the model was deployed with more confidence that even when an item is incorrectly classified, it is still likely to be put in a category that will be similar and better than no category.

Model Testing On Real-World Examples. The model was trained on a dataset from a different source than the data the model will be classifying. Some items from the Clothier inventory were evaluated to ensure that the model could generalize to real-world data examples (see Figure 8). The model could classify the small real-world data set with 100% accuracy.

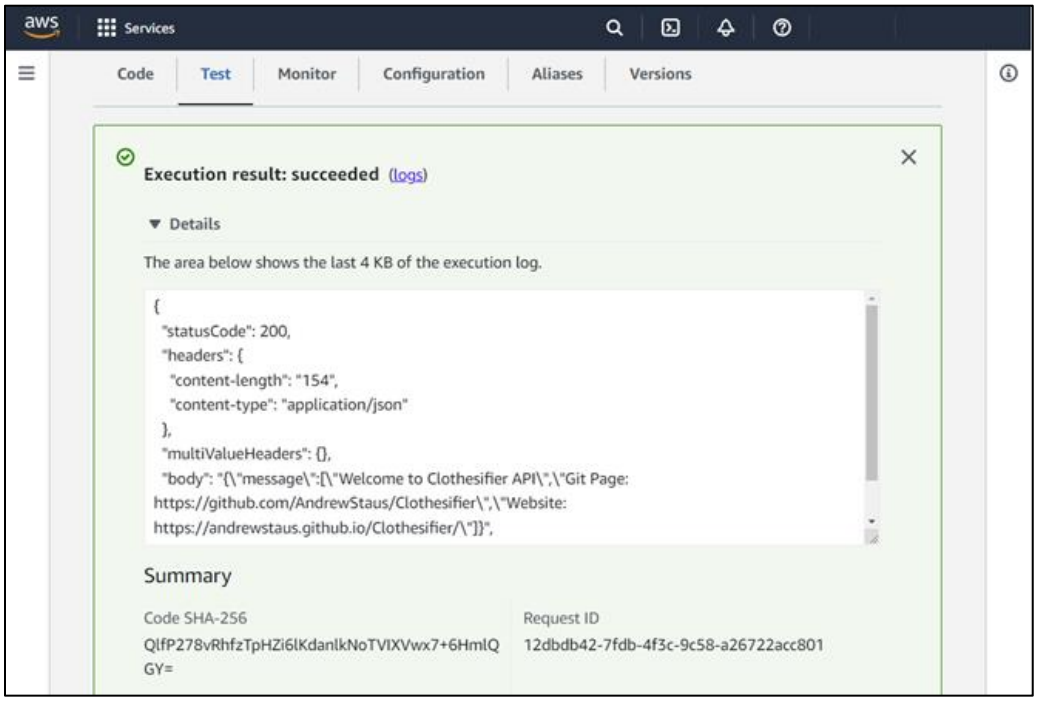
Application Testing

Throughout the development process, testing was conducted on each module as it was completed to verify that it was producing the expected outputs for both valid and invalid inputs.

API Module Testing. The operation of the API module, which was deployed on AWS Lambda, was verified using test cases. These tests simulated the submission of a base64 encoded image file and an invalid request, and the resulting JSON response was checked to ensure that it met project requirements (see Figure 9).

Figure 9

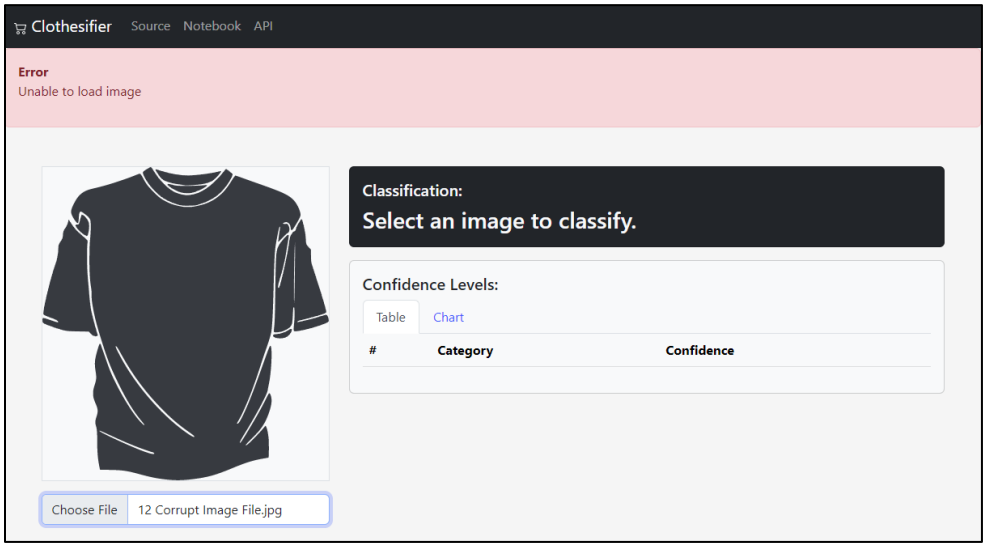
AWS Lambda Test



Website Module Testing. The Clothesifier website was designed to catch exceptions within its functions and display descriptive error alerts to the user. Testing was done to verify that the appropriate error messages were displayed for all error events (see Figure 10). However, due to the API not yet being integrated with the webpage at the testing time, it was impossible to validate a successful test case at this time.

Figure 10

Error Alert After Attempted Corrupt Image File Upload



System Testing. After all of the modules were integrated, the whole system was tested to ensure that it correctly handled valid inputs, invalid inputs, and system errors – such as an unresponsive API. This final round of testing confirmed the overall functionality of the completed system.

User Guide

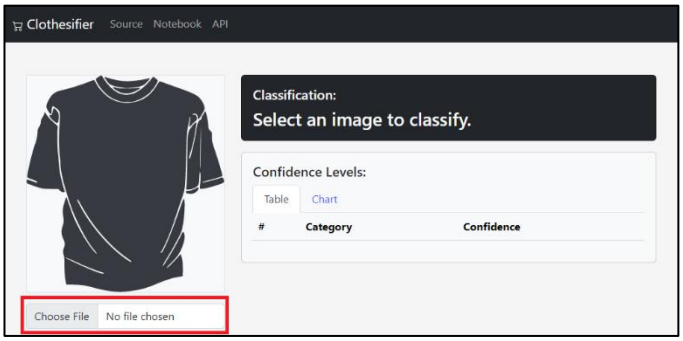
The Clothesifier API and frontend website are deployed in production environments and are ready for use. No installation or setup is required. Instructions on deploying a service on AWS Lambda or integrating an API into an application are outside the project's scope. The user

guide (see Table 6) will focus on using the Clothesifier website, which has all modules deployed and integrated.

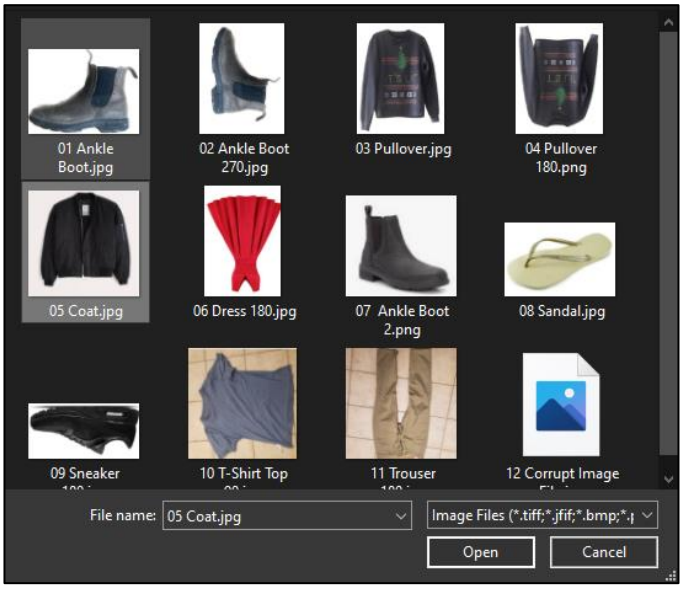
The complete source code for the project is available in the GitHub repository: https://github.com/AndrewStaus/WGU-Capstone-Clothesifier

Table 6

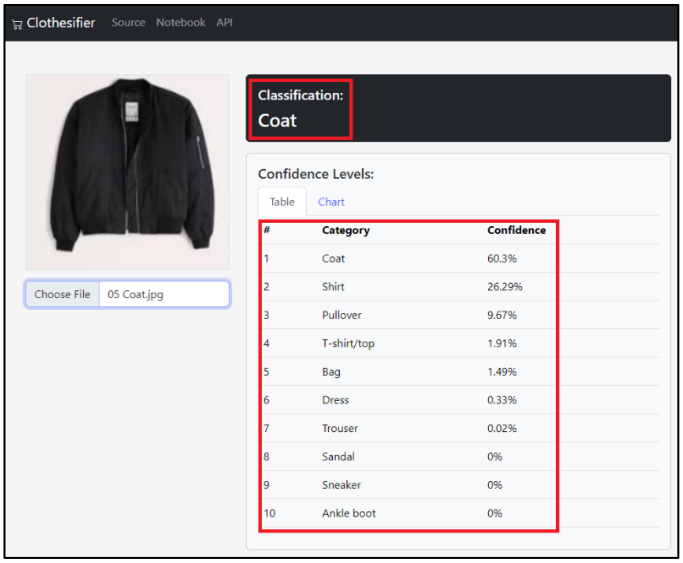
Clothesifier User Guide

1. Download and extract the test images.	https://andrewstaus.github.io/WGU-Capstone-Clothesifier/webpage/test_images/test_images.zip
2. Navigate to the Clothesifier website.	https://andrewstaus.github.io/WGU-Capstone-Clothesifier
3. Press the "Choose File" button.	

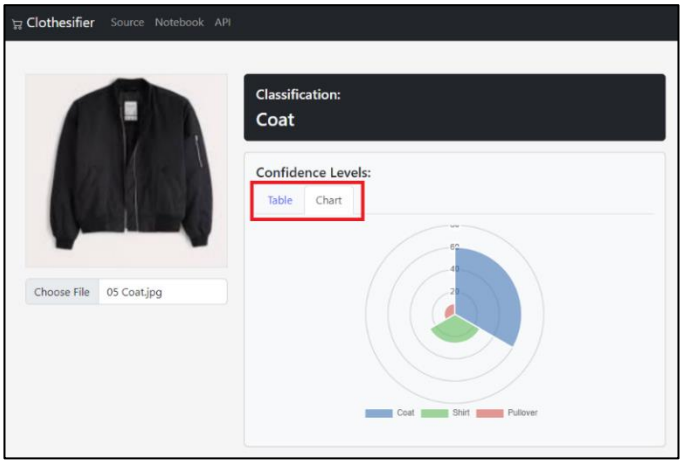
4. Select an image from the downloaded test images and press "Open."



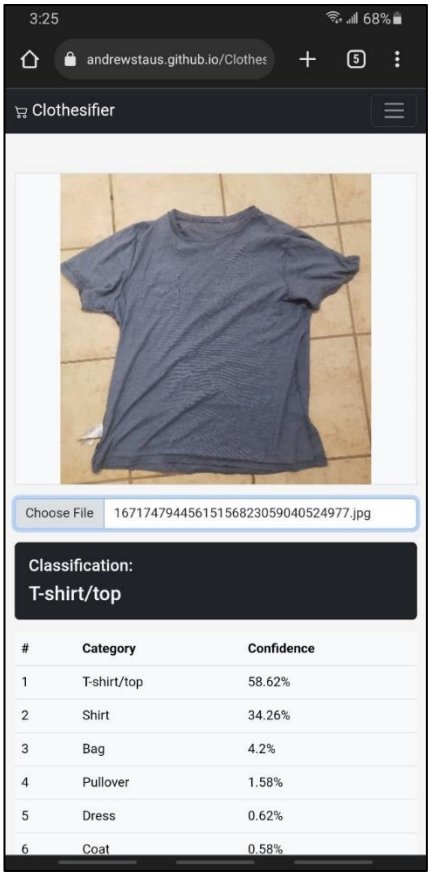
5. View the classification result and confidence levels.



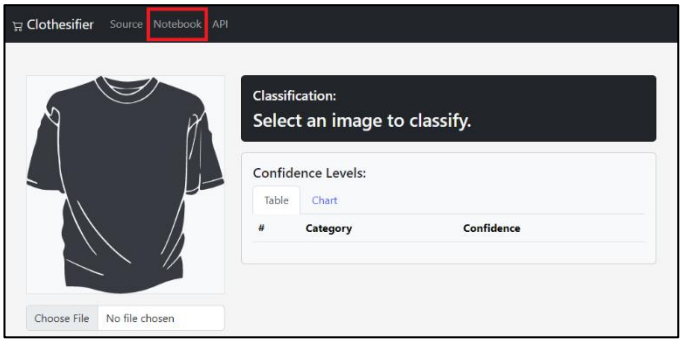
6. Toggle between the table and chart view of confidence levels by selecting the respective tab.



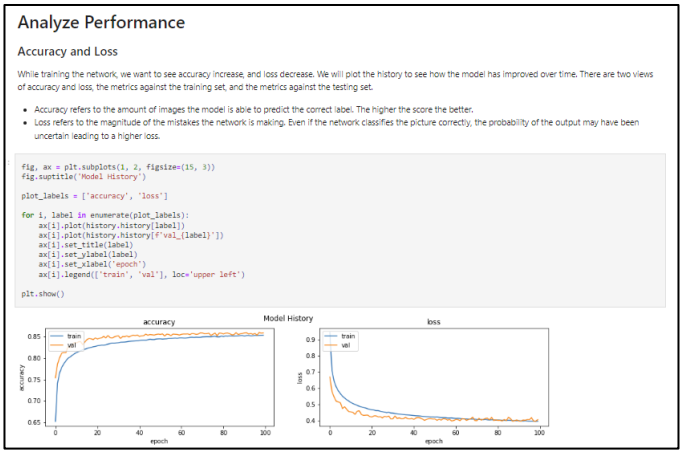
7. Alternatively, users can take advantage of the mobile-friendly webpage design by uploading their images using their phone or tablet camera by navigating to the site on their mobile device.



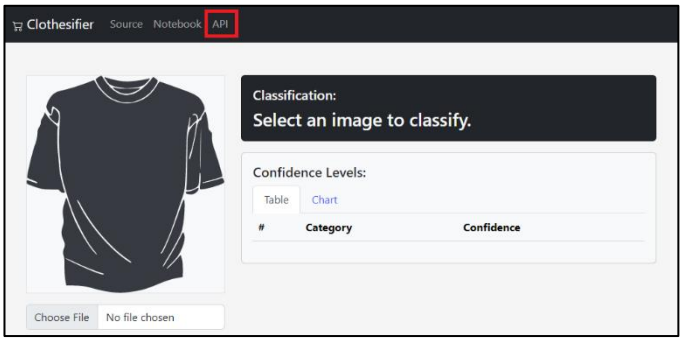
8. The EDA and model training Jupyter notebook can be reviewed by clicking the "Notebook" link on the navigation bar.



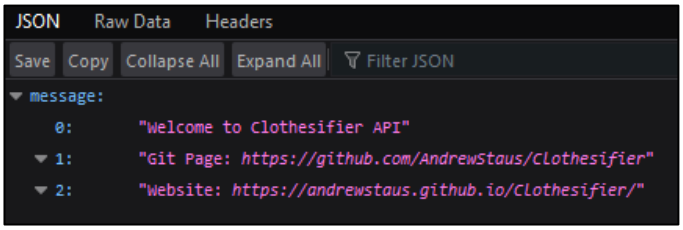
The notebook holds the most details for the descriptive and non-descriptive methods and visualizations.



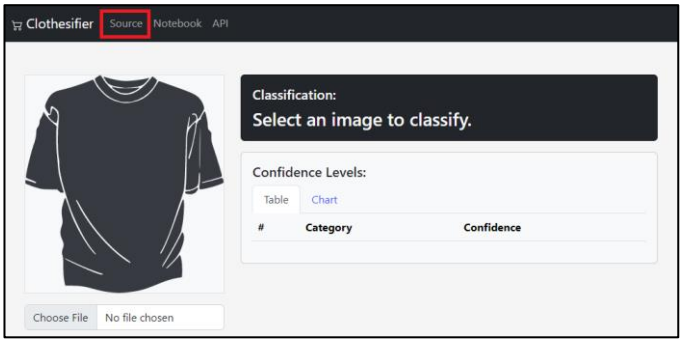
9. The root endpoint for the API can be accessed by clicking the "API" link on the navigation bar.



Integrating an API into an application is out of the scope of this document. However, more information can be found here: https://www.w3schools.in/restful-web-services/intro



10. The complete source code for the project can be found in the GitHub repository by clicking the "Source" link on the navigation bar.



Summation of Learning Experience

While working on this assignment, I faced the challenge of completing an open-ended project without defined success metrics. However, my previous education and desire to produce a useful and practical application motivated me to deliver a solution that a business could apply in real-world scenarios. The project management skills I gained from ITIL certification helped me understand the processes involved in developing and deploying an application, including planning and execution. The experience of building multiple applications from scratch and creating functional interfaces during my education gave me the confidence to tackle a more complex development challenge.

I built this application primarily using Python and JavaScript because I was familiar with these languages from my studies and had gained proficiency in using them through previous projects. I also took several machine learning courses and was proficient in using the TensorFlow library to solve image classification problems. To further challenge and expand my knowledge, I selected FastAPI and AWS Lambda as technologies because I desired to familiarize myself with them. I wanted to learn how to deploy efficient and scalable solutions for real-world applications using these technologies.

Overall, this project has been a valuable learning experience and has encouraged me to continue expanding my programming skills and seek certification. The knowledge and skills gained from this project will be helpful in my future endeavors and help me deliver high-quality solutions in various contexts.

References

Doe, J. (2022). *Fictitious Citation Simulating Internal Document.* Calgary: Clothier.