C964 Computer Science Capstone: Clothesifier

Andrew Staus

Western Governors University

010312389

Table of Contents

[Part A: Project Proposal for Business Executives 5](#_Toc122911328)

[Letter of Transmittal 5](#_Toc122911329)

[Project Recommendation 8](#_Toc122911330)

[Problem Summary 8](#_Toc122911331)

[Application Benefits 8](#_Toc122911332)

[Application Description 9](#_Toc122911333)

[Data Description 9](#_Toc122911334)

[Objectives and Hypothesis 9](#_Toc122911335)

[Methodology 10](#_Toc122911336)

[Funding Requirements 10](#_Toc122911337)

[Data Precautions 11](#_Toc122911338)

[Developer Expertise 11](#_Toc122911339)

[Part B: Project Proposal 12](#_Toc122911340)

[Problem Statement 12](#_Toc122911341)

[Customer Summary 12](#_Toc122911342)

[Existing System Analysis 12](#_Toc122911343)

[Data 13](#_Toc122911344)

[Project Methodology 13](#_Toc122911345)

[Project Outcomes 14](#_Toc122911346)

[Implementation Plan 15](#_Toc122911347)

[Evaluation Plan 16](#_Toc122911348)

[Model Module Testing 16](#_Toc122911349)

[API Module Testing 16](#_Toc122911350)

[Webpage Module Testing 16](#_Toc122911351)

[System Testing 17](#_Toc122911352)

[Post-deployment Evaluation 17](#_Toc122911353)

[Resources and Costs 17](#_Toc122911354)

[Timeline and Milestones 18](#_Toc122911355)

[Part C: Application 20](#_Toc122911356)

[Application Files 20](#_Toc122911357)

[Part D: Post-implementation Report 21](#_Toc122911358)

[Business Vision 21](#_Toc122911359)

[Datasets 23](#_Toc122911360)

[Data Product Code 25](#_Toc122911361)

[Objective and Hypothesis Verification 25](#_Toc122911362)

[Effective Visualization and Reporting 28](#_Toc122911363)

[Accuracy Analysis 31](#_Toc122911364)

[Accuracy Against the Validation Set 32](#_Toc122911365)

[Misclassification Analysis 32](#_Toc122911366)

[Testing Against Real-World Examples 32](#_Toc122911367)

[Application Testing 32](#_Toc122911368)

[User Guide 34](#_Toc122911369)

[Summation of Learning Experience 38](#_Toc122911370)

# 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 1T2

Dear Mr. Hamilton,

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

The Clothier website already includes the functionality to sort items by category, but this feature is not being utilized due to the lack of categorization for the existing and growing inventory, and manually labelling each item in the inventory would be cost-prohibitive. To address this issue, I propose implementing a machine learning system to automatically categorize all items in the Clothier inventory, as well as provide categorization for new items as they are added. A machine learning system can be implemented to automatically categorize all items in the Clothier inventory and provide the ability to also provide categorization for new items as they are added. The system will be deployed as an API that can be integrated directly into Clothier’s existing inventory system, providing seamless integration for business users.

The categorization of clothing will provide a simpler customer experience, and lead to more conversion as customers easily find the items they are looking for. 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 75 working hours for one developer at a rate of $100 per hour, totalling $7,500. This investment represents a cost-effective solution for improving inventory management and customer experience for the Clothier business. The service will be developed using software development tools and libraries that are available at no cost and with no licensing fees. This means that Clothier will not incur any additional costs 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. This will further reduce costs and improve the overall efficiency of the system. The serverless platform does incur a minimum usage fee of $0.20 per one million requests, but this limit far exceeds Clothier's current needs. As a result, 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.

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

Sincerely,

## Project Recommendation

### Problem Summary

**.** The Clothier e-commerce business is facing a significant challenge in terms of 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 being used by Clothier includes the ability to sort items by category, but this feature is not being utilized due to the lack of category labels for the existing and future 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 will be able to improve the customer experience and provide business stakeholders with the ability to analyze relevant information.

### Application Benefits

**.** The implementation of 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 expected by customers. Research shows that 80% of customers browsing the Clothier website leave without purchasing an item, citing difficulties in navigating the site and finding the desired products. By integrating the Clothesifier system directly into the inventory system, customers will be able to 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 for seamless integration with the existing Clothier e-commerce systems. This will allow the Clothier to easily categorize their inventory and improve efficiency and accuracy in their business operations.

### Data Description

**.** The Clothesifier system will be trained 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, it is possible that the images in the Clothier inventory may not be of the same standard, which could limit the performance of the model. To address this potential limitation, we will perform data augmentation to make the dataset more representative of real-world data and improve 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 will be able to generalize to the Clothier data and perform accurate image classification. This will allow the Clothier to more easily categorize and manage their inventory, improving efficiency and accuracy in their business operations.

### Methodology

**.** A waterfall methodology will be followed to develop the system, following a linear, sequential approach. This methodology involves dividing the development process into distinct phases, with each phase building upon the earlier one. The phases to be followed are:

* Requirements – Stakeholders will be engaged to gather the specifications needed for the final system. 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 code will be written, and the systems deployed and integrated.
* Verification – The system will be evaluated to ensure that it behaves correctly with both valid and invalid input.
* Maintenance – Any bugs, issues, or additional features missed during verification will be assessed and resolved.

### Funding Requirements

**.** The application will be implemented using a serverless architecture provided by AWS Lambda. This will eliminate the need for infrastructure management and maintenance. While there is a usage fee of $0.20 per one-million requests, the first one-million requests will be free. Based on the size of the Clothier inventory, it is highly unlikely that there will be any costs associated with running the service.

For the project development, we will use development tools and software libraries that are available at no cost and do not require any licensing fees. The estimated development time for this project is 75 working hours for two software developers at a rate of $100 per hour, for a total cost of $7,500.

### Data Precautions

**.** The data used to train the model is publicly available and does not contain any sensitive or proprietary information. In addition, the nature of the system being developed ensures that the data is never stored in an accessible database.

### 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. We believe that 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 inadequate categorization system for the products it offers. To address this issue, we propose the implementation of an automated classification system deployed as a REST API. 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 current categorization system is inadequate, leading 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 can be easily integrated into the platform's inventory pipeline, allowing for the automatic categorization of newly added items, and streamlining the inventory management process for the platform's developers. Additionally, the API can be used by Clothier's analysis team to accurately categorize the current inventory, improving the overall user experience and conversion rates on the platform.

## Existing System Analysis

The Clothier e-commerce platform currently lacks a functional categorization system, leading to poor customer experiences and low conversion rates. While the platform does support the integration of REST APIs, the lack of a categorization system remains a significant issue. Our proposed solution, a data product in the form of a REST API endpoint, aims to address this problem by supplying a comprehensive and automated method for categorizing both new and existing inventory on the platform. The API can 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 labelled 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 in a suitable manner 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. This involves a sequential and linear progression through the following phases: requirements gathering, design, implementation, testing, and maintenance.

The Waterfall methodology is a cost-effective and efficient approach for small projects and those where requirements are expected to remain stable. 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. This will involve discussions about the desired features and functionality, as well as any specific requirements or constraints.

Once the requirements have been gathered and clarified, we will move into the design phase, where we will decide on the architecture of the application. This will involve deciding on the technologies and frameworks to be used, as well as the overall structure and design of the application.

Next, we will move into the implementation phase, where we will set up the programming environment, and begin 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 that are discovered.

Finally, we will move into the maintenance phase, where we will address any remaining bugs or regulatory concerns and ensure that the application is running optimally. Ongoing maintenance will be performed as needed to ensure the continued success of the Clothesifier application.

## Project Outcomes

Upon completion of the project, the API endpoint will be available for use by the customer. To show the functionality of the API, a browser-based GUI will be provided that allows 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 architectural style and will include documentation on its proper use. The customer will be able to access the API endpoint to integrate its functionality into their systems and applications. The API endpoint and associated documentation will supply clear instructions on how to use the API consistently and predictably.

## Implementation Plan

The proposed approach for implementing this project is a top-down approach, in which each step is completed in order. This approach is being adopted to ensure that functionality is verified after each step, ensuring that all modules will be able to integrate into a viable product. The steps to be followed 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 provide visualizations of 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 images, we will assess its accuracy on the validation set to determine its performance on items that the model was not trained on. 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 any 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 the upload of an image file and displays the classification results from the API to the user. Appropriate error alerts should be displayed to the user if any 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 is functioning correctly and seamlessly as a whole. This will involve evaluating the integration and communication between the different modules, as well as the overall user experience and performance.

### Post-deployment Evaluation

**.** After the Clothesifier application is deployed and in use, we will conduct ongoing evaluations to assess its performance and effectiveness. This will include monitoring the accuracy of the classification results, as well as 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 $0.00, as it will be hosted on the serverless cloud platform AWS Lambda. AWS Lambda offers a free tier for the first 1 million requests per month, 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, it is expected that users will remain within the free tier of 1 million requests per month. However, if the inventory size increases and additional requests are made exceeding the free tier, the cost for each additional 1 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 expected to require 75 hours over the course of 2 weeks. This schedule is the result of constraints of the assigned software engineer’s schedule. Completion of the project is dependent 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, which will be performed autonomously outside of business hours.*

# Part C: Application

## Application Files

The application files will be organized within the three development modules, (a) Model Training; (b) API; (c) Webpage; as described in the table below (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 front-end 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 catalogue of clothing items for sale, but no system in place for categorizing them. This made it difficult for customers to find specific items and for the company to effectively manage and organize its inventory. The Clothesifier system was developed 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

A picture containing diagram

Description automatically generated

*Image retrieved from the* [*training Jupyter notebook*](https://github.com/AndrewStaus/Clothesifier/blob/main/model%20training/notebook.ipynb)

The trained model was integrated into a REST API service that was created using the Python programing language, and open-source libraries then 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

Text

Description automatically generated

*Image captured from the* [*API root endpoint*](https://xxlkbgor75nvr7qw256z2xnrdm0ppqai.lambda-url.us-east-2.on.aws/)

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

To use the Clothesifier system, users can visit the front-end 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

Graphical user interface, application, website

Description automatically generated

*Image captured from the* [*Clothesifier webpage*](https://andrewstaus.github.io/Clothesifier/webpage/index.html)

## 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 also 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

A picture containing text, white, several

Description automatically generated

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. However, the final application does not use this dataset, as the model is deployed in a trained state on the API.

**Figure** 5

Images Representations Extracted from the Augmented Data

Calendar

Description automatically generated with medium confidence

## Data Product Code

TODO

## Objective and Hypothesis Verification

The first objective of the Clothesifier system is the ability to accept an image from the Clothier inventory and return the category for that item with 80% accuracy. The objective was accomplished with validation through 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

Graphical user interface, application

Description automatically generated

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

Figure 7

Confusion Matrix Showing Classification Successes and Errors

Graphical user interface, application

Description automatically generated

The hypothesis was that a model would be able to generalize to the Clothier data and perform accurate image classification. This was validated as a correct hypothesis by the model correctly 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

Graphical user interface

Description automatically generated

## Effective Visualization and Reporting

### Visualization and reporting played a key 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.

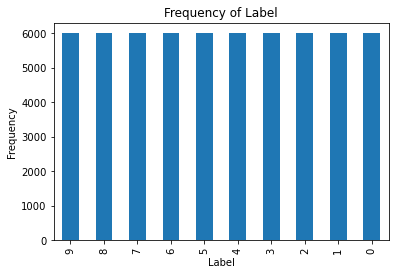
### Exploratory Data Analysis

**.** Since the data used in our convolutional neural network (CNN) was made up of flattened image files, heatmaps were used to plot these images pixel values to get a better understanding of 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

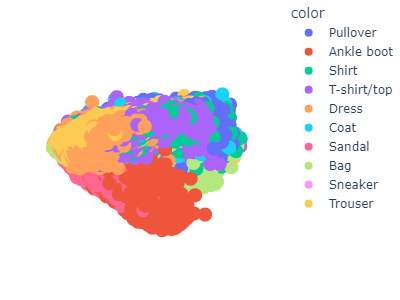
Bar Chart Showing Frequency of Labels



To better understand the similarities between the images, we used principal component analysis (PCA) 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



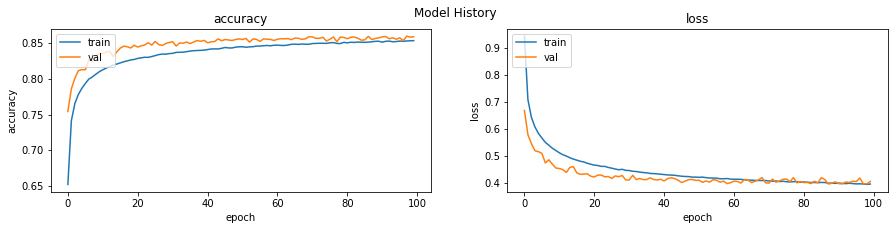
### Model Training

**.** Before training the model, the dataset was augmented to create more variation in the images the model was being trained on. The augmented data was visualized using heatmaps for inspection (see Figure 5). We also created a visual representation of the model's architecture for inspection (see Figure 1).

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

Figure 11

Accuracy and Loss



### User Interface

**.** To provide the user with a visual representation of the confidence levels of each predicted label for an uploaded image, a radar chart was used (see Figure 12)

Figure 12

A Radar Chart

Chart, radar chart

Description automatically generated

## Accuracy Analysis

As described in the Objective and Hypothesis Verification section, accuracy was assessed based on the performance of the CNN to correctly classify an image of clothing. Model performance was assessed during the Model Module Testing phase in 3 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 use in assessing the accuracy of the model. After the final training epoch, the model performed with close to 86% accuracy on the validation set (see Figure 6).

### Misclassification Analysis

**.** A confusion matrix was then generated from the validation data to visualize where the model was making classification mistakes (see Figure 7). The confusion matrix showed that mistakes 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 with the knowledge that even when an item is incorrectly classified, it is still being put in a category that will be similar and will be better than no category at all.

### Testing Against Real-World Examples

**.** The model was trained on a dataset that did not come from the same source as the data the model will be classifying. Some items from the Clothier inventory were evaluated to ensure that the model would be able to generalize to real-world data examples (see Figure 8). The model was able to classify real-world data 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

Graphical user interface, text, application, email

Description automatically generated

### Website Module Testing

**.** The Clothesifier website was designed to catch exceptions within its functions and display descriptive error alerts to the user. Testing was carried out 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 time of testing, it was not possible to validate a successful test case.

Figure 10

Error Alert After Attempted Corrupt Image File Upload

Graphical user interface, website

Description automatically generated

### System Testing

**.** After all modules were integrated, the full system was tested to ensure that it was handling valid inputs, invalid inputs, and system errors such as the API being offline correctly. This final round of testing confirmed the overall functionality of the completed system.

## User Guide

The Clothesifier API and front-end website are deployed in production environments and are ready for use. No installation or setup is required. Instructions on how to integrate API into an application are outside of the scope of the project, so instead the user guide (see Table 6) will focus on the use of the Clothesifier website which has the API already integrated.

Table 6

Clothesifier User Guide

|  |  |
| --- | --- |
| 1. Download and extract the test images | <https://andrewstaus.github.io/Clothesifier/webpage/test_images/test_images.zip> |
| 1. Navigate to the Clothesifier website | <https://andrewstaus.github.io/Clothesifier/> |
| 1. Press the “Choose File” button | Graphical user interface, website  Description automatically generated |
| 1. Select an image from the downloaded test images and press Open | Graphical user interface, application  Description automatically generated |
| 1. View the classification result and confidence levels | Graphical user interface, website  Description automatically generated |
| 1. Toggle between the table and chart view of confidence levels by selecting the respective tab | Graphical user interface  Description automatically generated |
| 1. Alternatively, users can take advantage of the mobile-friendly webpage design by uploading their images directly using their phone or tablets camera by navigating to the site on their mobile device | Graphical user interface  Description automatically generated |
| 1. The EDA, and model training Jupyter notebook can be viewed by clicking the “Notebook” link on the navigation bar | Graphical user interface, website  Description automatically generated |
| The notebook holds most of the details for the descriptive, and non-descriptive methods and visualizations | Graphical user interface  Description automatically generated |
| 1. The root endpoint for the API can be accessed by clicking the “API” link on the navigation bar | Graphical user interface, website  Description automatically generated |
| Integrating an API into an application is out of the scope of this document, however, more information can be found here: <https://blog.hubspot.com/website/application-programming-interface-api> | Text  Description automatically generated |

## Summation of Learning Experience

TODO