**Product Look App**

An image recognition app for Canadian Tire

**Group Name:** Group 1

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **First name** | **Last Name** | **Student number** |
| Ajay | Dahiya | C0783939 |
| Akshita | Khatri | C0785493 |
| Angelica | Serrano | C0785370 |
| Jonatas | Aguiar | C0790419 |

**Submission date:** 14-August-2021

Contents

[Abstract 3](#_Toc79798368)

[Introduction 3](#_Toc79798369)

[Methods 3](#_Toc79798370)

[**A.** **Data collection** 3](#_Toc79798371)

[**B.** **Data pre-processing steps** 4](#_Toc79798372)

[**C.** **Machine Learning Model Architecture** 5](#_Toc79798373)

[**D.** **Web Application** 5](#_Toc79798374)

[**E.** **Deployment to Cloud** 7](#_Toc79798375)

[Results 8](#_Toc79798376)

[**A.** **Model experiment and result** 8](#_Toc79798377)

[**B.** **User Acceptance Test (UAT) result** 9](#_Toc79798378)

[Conclusions and Future Work 10](#_Toc79798379)

[References 10](#_Toc79798380)

# **Abstract**

Image recognition is a type of machine learning system that accepts photo as input data, analyzes and translates it into a certain class and returns the output in a human readable format. There are several applications of image recognition such as classifying objects, recognizing fake images and identifying handwritten words to name a few. In this project, image recognition was used to identify the category of any Canadian Tire product images uploaded into the web application called Product Look App.

# **Introduction**

Canadian Tire is a popular retail and ecommerce company in Canada. It offers wide selection of goods such as automotive, household supplies, tools, lightings, inflatable pools and even pet supplies. Due to the present situation where outside errands pose health risk, more customers prefer online purchases. This means more traffic and great timing for ecommerce website upgrades. In Canadian Tire website, the fastest option to search for products is the conventional search field. While this search field brings convenience to most users, some people often forget, or are unfamiliar about the product they are looking for. Hence, sometimes it causes annoyance than convenience. This project aims to bring ease and enhanced customer experience while users browse through Canadian Tire website.

Product Look App is a web application that can classify product images uploaded through the search field. Using Convolutional Neural Network (CNN) model and training images from Canadian Tire, app can identify the category where a specific product belongs to. Ultimately, users will be redirected to Canadian Tire website to browse for available brands and proceed to checkout.

# **Methods**

Agile methodology was adopted in implementing this project. The workflow on each stage was optimized to prevent 100% utilization of resources (group members). Additionally, a proper workplan was laid out initially to follow the proper sequence and promote collaboration between members. Deployment happens every iteration and the process started with Data collection. In this section, every stage will be discussed in a comprehensive report.

### **Data collection**

Images used for training the models were manually scrapped from Canadian Tire website. Initial plan was to use pre-processed images available on public repositories such as Kaggle, GitHub, or UCI. However, this approach may not work specifically for Canadian Tire. Hence, python codes for each five sections were developed separately to extract the train and test images.

#### **A.1 Data collection Tools**

The following tools were leveraged to scrape images from Canadian Tire website:

1. **Beautiful Soup -** is designed mainly for web scraping objects such as texts and images. In this project, Beautiful Soup was used in scraping (1) product images and (2) product URLs from Canadian Tire website.
2. **Selenium –** Selenium web driver module functions as navigator to a page given by the URL. This was used alongside Beautiful Soup and headless browsers (Chromium and Firefox) to automate the selection and click actions required to extract the objects needed from Canadian Tire.

#### **A.2 Dataset**

The final dataset consists of 103,500 train images designated across five product sections: automotive, household & pets, outdoor living, sports & recreation, and tools & hardware. A total of 207 categories (classes) ware used for this project.

Graphical user interface, application

Description automatically generated***Figure 1*** *– Sample images extracted per section*

### **Data pre-processing steps**

Image pre-processing was performed to improve the image features of our dataset. Aside from properly grouping and labeling each category into their proper class, image transformation also enhanced the model performance. While this stage was generally overlooked by most ML developers, this is one of the essential stages in this project. The steps associated in data pre-processing are discussed below.

#### **B.1. Read images**

Background pattern

Description automatically generated with medium confidenceIn this step, all the collected images were dumped into designated folders in Google Drive, where each folder pertains to their category names. Python script was created to count the number of original images and augmented images that needs to be generated on the next step.

***Table 1 –*** *Sample data from read\_img table*

#### **B.2 Augment the images**

In Convolutional Neural Network, the main goal of data augmentation is to avoid overfitting. Overfitting happens when the model is trained with images with high variance. Since Canadian Tire product images have huge variation, augmentation technique was applied.

An image augmentation library named *AugLy* was utilized to perform random transformations on the train images. This is a newly released data augmentation library by Facebook and supports four modalities including image. These functions were used primarily to transform the dataset:

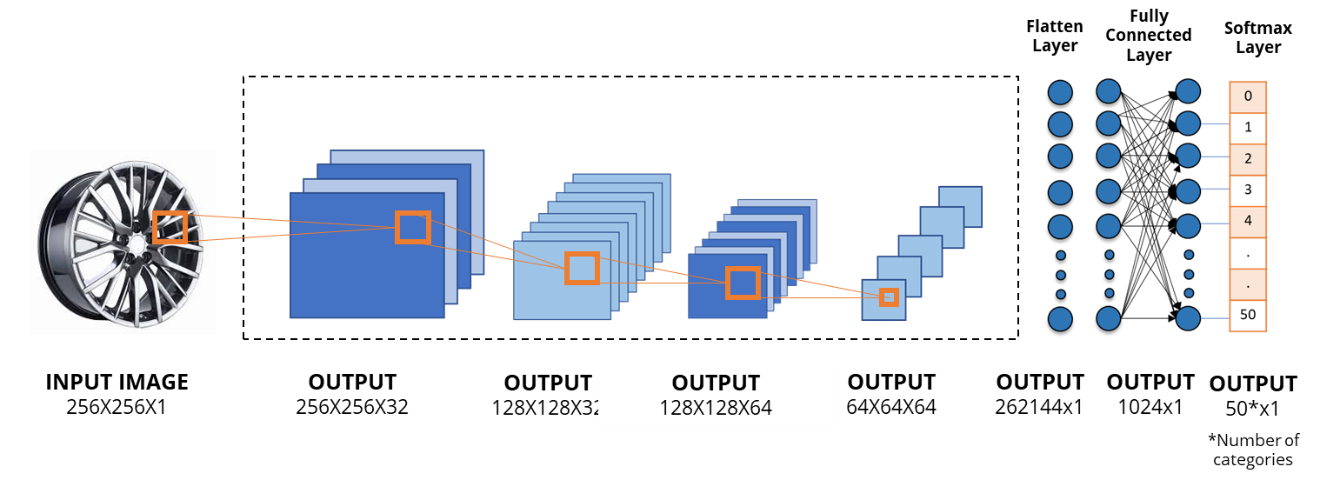
1. Scale – to randomly resize the image.
2. Saturate – to increase the intensity of image colors.
3. Shuffle – to shuffles the pixels of the image.
4. Pixelization – to pixelates an image.
5. Rotate – to randomly rotates the image to a certain degree angle.
6. Blur – to changes the sharpness of an image.
7. Change Aspect Ratio – to randomly changes the height and width of an image.
8. Sharpen – to changes the sharpness of an image.
9. Perspective Transform – to change the perspective of an image.

#### **B.3 Label the dataset**

Lastly, dataset features (X) and labels (y) were saved respectively as pickle files. Pickle is a python module that allows user to save an array into disk by serializing the object structure before saving it.

### **Machine Learning Model Architecture**

Five multi-class classification models were built using Convolutional Neural Network (CNN) - a popular Deep Learning algorithm that is commonly used for image classification. There are several libraries available to implement CNN, such as Tensorflow, Keras and Pytorch. For this project, Keras sequential model was applied, where each model consists of input layer, hidden layers, and output layer. Figure 2 shows the custom CNN architecture implemented across all five sections.

***Figure 2*** *– CNN architecture*

A series of experiments were implemented to come up with a model that generates highest accuracy and lowest training loss. Refer to *Result* section for more comprehensible details about the final model.

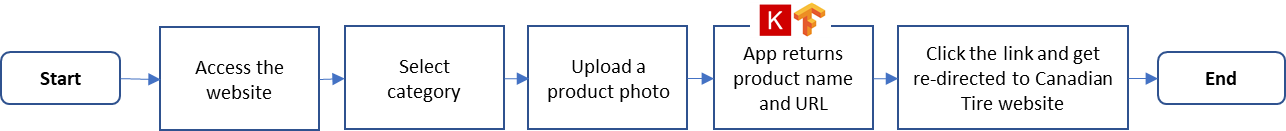
The image dimension used for training each model was 256x256. Hence, a single generated h5 file, as expected, reached 3GB. To reduce the file size, pruning technique was applied. Pruning is a process of eliminating the unused parameters in a neural network model. This process has reduced the size of each model files down to 1GB.

### **Web Application**

A web application made with Streamlit was created as user interface. Streamlit is an opensource framework that can be used to run machine learning models in a graphical user interface such as web browser. Based on the several trials made using different platforms (Flask, Django) - Streamlit was found to be the most efficient and lightweight tool for this project.

#### **D.1 Front-end process**

The web application workflow is straightforward. Initially, user needs to select the category of their product, then upload a photo, and wait until the app detects the correct product name.

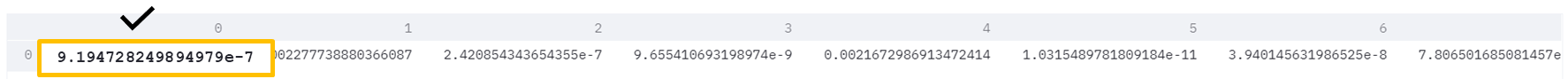
******

***Figure 2*** *– Website workflow*

#### **D.2 Back-end process**

One of the advantages of using Streamlit was the availability of pre-built components that comes with it, such as input image field and selection buttons. Thus, integrating the CNN model was easier than using other libraries. The following items describe the back-end processing steps:

1. After uploading a valid photo (jpg or png format), application calls the model file for specific selected section (automotive, tools & hardware, home and pets, sports & recreation, or outdoor living).
2. The uploaded image will be pre-processed through -
   1. Conversion to grayscale
   2. Resizing to 256 x 256 (trainset dimension)
   3. Converting to array
   4. Reshaping the array to (-1, 256, 256, 1)
3. Prediction output returns a set of arrays and will be parsed to get the class with highest prediction percentage.

******

***Figure 3*** *– Sample prediction output*

1. To display the output in a human-readable format, predicted class will be mapped against major\_category\_id column of data source table. For further details about the data source table, refer to *Database* section

Graphical user interface, text

Description automatically generated

***Figure 4 –*** *Sample predicted class in human readable format*

#### **D.3 Database**

Table

Description automatically generatedThe database used in this project is in a form of csv file imported to Pandas dataframe. Since the content isn’t relatively huge, csv file is sufficient to hold all the data needed for the application to work as designed. The data source table contains a set of data with category names, Canadian Tire URL for each product, category section and the product’s array position from the prediction output described in *D.2 Back-end process.*

***Table 2 –*** *Preview of data source table*

### **Deployment to Cloud**

Production deployment was conducted right after completing the development and functional testing phase.

#### **E.1 Deployment Tools**

One of the goals of this project, is to build a simple interface with straightforward and well-structured architecture so future changes can easily be implemented. Hence, the following tools were leveraged in this stage.

1. **GitHub** – is a popular source code repository and versioning tool.
2. **Streamlit** – to build a web interface that is integrated with CNN model. Refer to section *D. Web Application* for further details about this tool.
3. **Heroku** – is a platform-as-a-service (PaaS) cloud host that offers free tier for web applications with size not more than 500MB. Heroku was utilized during development phase, when each of the model was tested separately.
4. **Azure** – is a cloud computing service from Microsoft. Like Heroku, Azure also offers free tier account but with a wider choice of Virtual Machine (VM) types. Azure hosts the Product Look App production server, with a standard 4 virtual CPUs, 16 GB RAM and 30GB storage OS disk. It runs in Linux CentOS 7.9 that can be accessed via SSH through a PEM key file.
5. **Docker** – is an opensource platform used for containerization. It enabled swift and effective deployment process of our application because of its capability to generate a portable and lightweight docker image.
6. **Docker Hub** – is the central meeting place for developers where compiled images can be uploaded to be stored or shared to community. Product Look App has been published to docker hub for public use.

#### **E.2 Deployment Architecture**

Figure 5 shows the application design workflow, from local developer machine until deployment to production environment hosted in Azure. Refer to detailed steps below:

1. Developers commit and push the updated codes to GitHub.
2. Compile the updated version into docker image and push to docker hub.
3. Connect to production environment via SSH and pull the latest image file from docker hub.
4. Run the latest docker image using TMUX or Terminal Multiplexer to keep the application running at port 8085 even after closing the Linux terminal window from our local machine.

***Graphical user interface, application

Description automatically generatedFigure 5 –*** *Application design architecture*

# **Results**

**An array of experiments and evaluation were performed to produce the final version of the application. In this section - Model experiment and UAT result will be comprehensively discussed.**

### **Model experiment and result**

Seven separate models were built and executed for each category. Subsequently, the generated validation loss and accuracy of each model was compared to determine which one performed best.

Graphical user interface, application

Description automatically generated

***Figure 6*** *– Loss and Accuracy graph for six experimental models for Home and Pets section*

Figure 6 shows six experimental models trained for Home and Pets section where the first few models show high training loss and low accuracy. To reduce overfitting, the following steps were applied:

1. Regularization – which regularized or reduced the coefficient estimates towards zero. This technique made the learning simpler during training phase.
2. Weight initialization – by defining the initial value of weights prior model training.
3. Weight constraints – by constantly checking and updating the weight sizes. It applies a condition that if weight exceeded its pre-defined limit, it will be reduced to lower size or within the range limit.
4. Image augmentation – by transforming the training data using random augmentation functions such as blur, sharpen, aspect ratio and so on. For further details about image augmentation, refer to Section *B. Data pre-processing.*

Graphical user interface, application, table

Description automatically generated

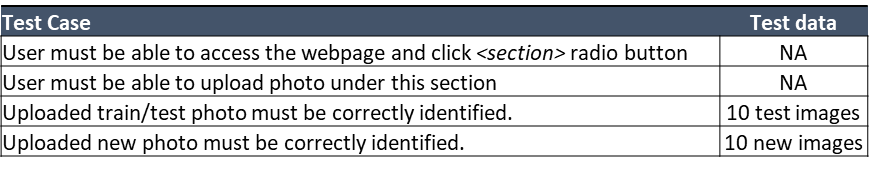
***Figure 7*** *– Final model output for Home and Pets section*

The final model shown for Home and Pets at Figure 7 has nine layers with full details as follows:

1. **Input layer** - A convolutional layer of 32 activation maps, using a 3x3 filter size, and ReLU activation function was applied against the 256x256 grayscale image.
2. **Max Pooling layer 1** - to downsize the activation maps from 256x256 to 128x128.
3. **Dropout layer 1** - with a rate of 0.4 to minimize overfitting.
4. **Hidden convolutional layer** - with 64 filters of 3x3 size and ReLU activation function was applied. The output was 64 activation maps with 128x128 size.
5. **Max Pooling** **layer** **2** - to downsize the 64 activation maps from 128x128 to 64x64.
6. **Dropout layer 2** - with a rate of 0.4 to minimize overfitting.
7. **Flatten layer** - to squeeze the activation maps into one dimension.
8. **Fully connected layer** - with 1024 nodes and ReLU activation function.
9. **Output layer** - where nodes refer to the number of categories. The SoftMax activation function was applied as we are dealing with multi-class classification.

### **User Acceptance Test (UAT) result**

Test Plan was created, and each section were distributed among members. Some of the steps are described below:



***Table 3 –*** *Preview of Test Plan*

All items passed, where *<section>* pertains to: Automotive, Tools & Hardware, Home & Pets, Sports and Recreation and Outdoor Living.

# **Conclusions and Future Work**

In this project, dataset was collected by scrapping product images from Canadian Tire website using python code, with Beautiful Soup and Selenium web driver as primary modules. A total of 103,500 images were collected, pre-processed, and labeled with 207 categories. CNN was leveraged in creating the multi-class classification model. Seven experimental models were built for each five sections (Automotive, Tools & Hardware, Home & Pets, Sports and Recreation and Outdoor Living). Various steps were conducted to reduce overfitting such as regularization, weight constraints and image augmentation. Streamlit, an opensource web framework was utilized to create a web application that runs the model files. The whole application including codes, configuration set-up and software dependencies were compiled into one Docker image file. Heroku and Azure were used as Development and Production environment, respectively. The existing version of the application is now running in production environment and can be accessed through this link: http://productlookapp.eastus.cloudapp.azure.com:8501/.

For the next version, model and training set enhancement will be the primary focus. Since the models only yield 50-65% accuracy during evaluation, further training, and hyper-parameter tuning needs to be done to improve the model performance. Moreover, having an adequate and balanced dataset is crucial in implementing this project. Hence, data collection is also something that needs to be planned thoroughly for a longer timeline, to obtain suitable and sufficient dataset.

# **References**

// Complete citations for any articles or other materials referenced in the text of the article.