Ally: Designing Interfaces for Human + AI Collaborative Creativity for Computer Aided Design (CAD) Applications

by

Isabelle Chong

B.S. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author . . . .
Department of Electrical Engineering and Computer Science
May 6, 2022

Certified by . . . .
Pattie Maes
Professor of Media Arts and Sciences, MIT Media Lab
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Ally: Designing Interfaces for Human + AI Collaborative Creativity for Computer Aided Design (CAD) Applications

by

Isabelle Chong

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Creativity is an essential component to design and is something that is seen as intrinsically human. As the world continues to become more and more digital, computers become ever more present in the world of creativity. However, the relationship between human and technology does not have to be adversarial. My research for Ally builds on the work of Paper Dreams, an adaptive drawing canvas platform with the objective of augmenting creativity using machine learning and multimodal inputs. In collaboration with PTC, Ally expands Paper Dreams, taking digitally drawn sketches using a Sketch-A-Net model or webcam images using a YOLOv5 model to recognize user input and build a Computer Aided Design (CAD) scene that has been collaboratively created by a human and an algorithm. The devised model is ultimately able to correctly identify the desired part for the user's design in one of the top five most similar results at a rate better than by chance on a test set of possible user input data. Using the CAD capabilities of PTC Onshape and the Sketch-A-Net/YOLOv5 models, an extension to Paper Dreams has been built that will hopefully be usable in industry to create "digital twins" and allow humans and machines to design together.

Thesis Supervisor: Pattie Maes
Title: Professor of Media Arts and Sciences, MIT Media Lab

Acknowledgments

I would like to thank my direct supervisor Guillermo Bernal, Professor Pattie Maes, and the rest of the Fluid Interfaces group at the Massachusetts Institute of Technology Media Lab. I would also like to thank professors Fredo Durand and Wojciech Matusik for their extremely helpful advice on how to approach many of the problems encountered over the course of this project. Finally, I would like to thank my partner, friends and family for supporting me as I completed this thesis.

Contents

1 Introduction **9**

2 Related Work **13**

3 Methods **17**

3.1 Application Infrastructure 17

3.1.1 Onshape and User Interface 17

3.1.2 Database 19

3.2 Recognition Models 20

3.2.1 Three dimensional sketch feature recognition via Sketch-A-Net 20

3.2.2 Webcam image object recognition via YOLOv5 20

3.2.3 Experimental setup and data collection 23

4 Results **29**

4.1 Sketch-A-Net 29

4.2 YOLOv5 30

4.3 In-App Performance 31

5 Further Work **35**

6 Conclusions **37**

Chapter 1

Introduction

Many great developments in many different fields are the products of collaborations. In the modern era, a lot of collaboration is digital. Onshape is a cloud-based product development platform owned by PTC that allows teams of users to collaborate on Computer Aided Design (CAD) models [8]. Similarly to how multiple users might collaborate on a single Onshape project, Paper Dreams allows a human user to have a machine collaborator. On top of collaborative benefits, machine learning can also be mundanely used for design applications to query for parts to place into a model. In a lot of situations, people might want to place certain parts into a design project, but might not be able to remember the name of the part or where it is located in the document. Human-machine collaboration may be used to fill in these gaps. This situation often occurs when people want to create "digital twins," or digital CAD model copies of real-world 3D models. Previous iterations of Paper Dreams have allowed users to create 2D drawn scenes in collaboration with their computers, using a Generative Adversarial Network (GAN) [1].

Ally attempts to extend the functionality of Paper Dreams to 3D CAD models in Onshape, building both an infrastructure a user can interact with to collaborate with a machine to produce a CAD sketch and a model that can recognize a 3D component from a user's 2D drawing or webcam image.

One of the unique difficulties of trying to build a model to recognize a 3D part is that a 3D object can technically be viewed in infinite directions. In order to be able to successfully train a model that can identify sketches regardless of the angle at which the user has decided to sketch them, the model is trained using sketches and webcam images of the object from multiple angles.

Moreover, each of the potential user inputs comes with its own unique challenges. One of the challenges of recognition on a sketch-based model is the variation of quality of user sketches and the abstract nature of free-hand representations. To deal with this, a Sketch-A-Net model is utilized. Furthermore, adding in webcam images as a possible input type raises the issue of background filtration. Because webcam images will also capture extraneous information within the user's environment in the backgrounds of the photographs, it is necessary to identify the part without being confused by the extra information. In order to be able to filter out the backgrounds of images, a YOLOv5 model [2][5] is used, which trains using tight bounding boxes to be able to distinguish objects from their environments.

To allow a user to interface with this model, an Onshape application has been created. This application is able to accept input from either a drawing canvas or the user's webcam to be recognized by the Sketch-A-Net or YOLOv5 models, respectively. Once the user's input is submitted, the model is able to recognize the sketch or image as one of the elements from a preliminary set of Lego bricks, the CAD files for which are included in the Onshape document upon which the application is called. The CAD part corresponding to the classification is then able to be placed onto the modeling stage for the user to rearrange as they see fit in the larger design.

The created Onshape interface can help users discover new model configurations and increase performance and efficiency while reducing their cognitive load. This Onshape application offers users up to five possible part suggestions per input, letting them also see components similar to the one they input that might better fit their design. Furthermore, since for applications such as design or manufacturing, the part libraries tend to be quite huge, it can be difficult for a user to be able to search through the large dataset for the proper part by hand. By allowing users to query by sketch or image, the application lets them build models more quickly and with greater ease.

The devised model used in the application is ultimately able to correctly identify the desired part for the user's design in one of the top five most similar results for the user's sketch or webcam input at a rate better than by chance. Using the suggestion model, the user is able to see not only the model's guess at which part they would like to incorporate into their design, but also a few other parts the model deems as "similar" to allow for a more collaborative method of designing between human and machine.

Chapter 2

Related Work

Paper Dreams uses digital sketching as a platform to explore the potential dynamics of human-machine interaction for the purpose of augmenting creativity [1]. By taking live inputs such as pen strokes, the current 2-D Paper Dreams system is able to drive the creative thinking process forward by adding colors based on the recognized sketch, suggesting elements for the scene, and allowing the user to control the serendipity of suggestions. Within a matter of seconds and a few strokes, users are able to create a colorful digital painting whose elements are influenced by the suggestions given by the interface.

In contrast, because querying 3D models using 2-D images usually utilizes a more rudimentary quick sketch, these sketches do not typically contain color. Thus, most approaches that try to recognize 3D models via sketch will perform some variety of pre-processing to turn the 3D model into a collection of 2D line drawings from multiple perspectives. As previously stated, 3D model recognition approaches can require an infinite number of views since a user may technically sketch a 3D model from any angle, and there is no one view which can be said to be the definitive "view" of the model. Some approaches use a Bag of Features (BoF) model in order to query for 3D models using sketches, but these still fall victim to requiring training using specific viewpoints [3]. In order to ameliorate this, one can use a Siamese Convolutional Neural Network (SCNN), which requires comparatively few views to train. The SCNN has been successfully used in order to recognize sketches of 2D photographs in Koch et al. 2015 [6] and of 3D models in Wang et al. 2015 [11].

Another challenge to the use of sketches to query a dataset is that quality of sketches can vary greatly. The Sketchy Dataset was developed in order to try to create a dataset of sketches large enough for a network to learn based off of. Networks trained on this dataset had success using a Triplet loss function and two GoogLeNet networks for sketches and the photos they were paired to respectively, also incorporating each network's classification loss [9]. The current 2-D Paper Dreams sketch recognition model deals with this using a network based off of Sketch-a-Net [13], a deep neural network model that beats human recognition performance by 1.8% on the TU-Berlin dataset, which is another large scale benchmark dataset of sketch images. This neural network for Paper Dreams was trained on a combined dataset comprised of the TU-Berlin dataset, the Sketchy Dataset, and an independently gathered created dataset of sketches, augmented with the Augmentor Python library. However, since for Ally the models are being trained on a unique dataset of parts that are not included within the larger Sketchy or TU-Berlin datasets, there is a unique set of challenges as well.

Because this extension adds the ability to recognize components from webcam image inputs as well as sketches, it is also necessary to find a way to deal with the issues that come from this type of input. Namely, a model is needed that is able to recognize components within an image regardless of content in the background: an object detector. Object detectors are usually composed of a Convolutional Neural Network (CNN) backbone feature network, which compresses an input image into a collection of features, a neck that combines backbone feature layers, and a head that performs the actual detection [10].

Traditional object detectors typically use Mean Square Error (MSE) to perform regression on the center point, height, and width of the bounding boxes within an image annotation. However, more recently, YOLO models typically use Intersection of Union (IoU) loss to train. IoU loss will also take into account the area of intersection between the predicted bounding box and ground truth bounding box [12]. There are further variants on IoU loss as well, for example DIoU loss, which considers distance to the center of the object, and CIoU loss, which considers overlapping area, distance between center points, and aspect ratio [14].

Augmentations can also be applied to object detection datasets to raise the variety of situations upon which an object detection model is trained. Of particular interest are methods to stitch multiple images together, such the Mosaic method. Mosaic stitches together four images, which helps an object detection model get used to images with more variegated backgrounds for object detection.

Prior approaches to object detection include the YOLO (You Only Look Once) networks, which recognize objects within images based on bounding boxes [2]. YOLO is a one-stage object detector with many different versions, the most recent of which is YOLOv5 [5].

Chapter 3

Methods

The Ally extension consists of two main components: a prototype web application created to interface with PTC Onshape and two CNN models for sketch and webcam image recognition and model retrieval: a modified Sketch-A-Net model for sketch recognition and a YOLOv5 model for webcam image recognition.

3.1 Application Infrastructure

An overview of the architecture of the Ally application created can be seen in Figure 3-1. There are four parts to the application: the User Interface (UI) component that interacts directly with Onshape, a Python Flask application to handle GET and POST requests as well as host the webcam window, which cannot be accessed directly from the Onshape application due to privacy settings, the recognition and retrieval models, and a Google Firebase database that holds user sketch/webcam image inputs.

3.1.1 Onshape and User Interface

The Onshape application infrastructure is composed of a Node JS application that is hosted on Heroku and interfaces directly with Onshape documents as an add-on Onshape application. Specifically, the application is a sidebar app that can be opened

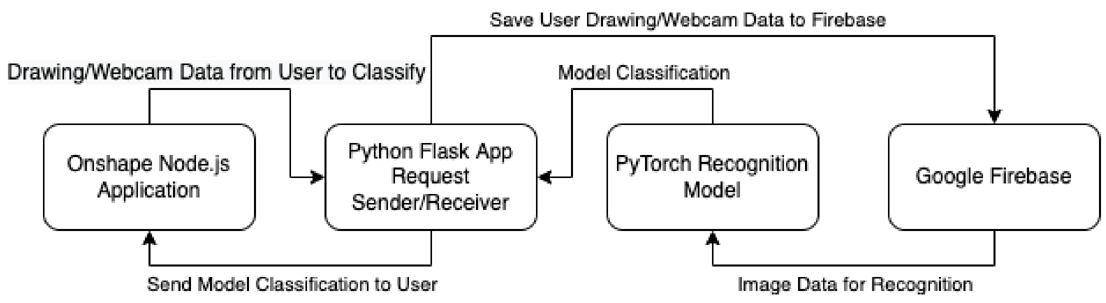
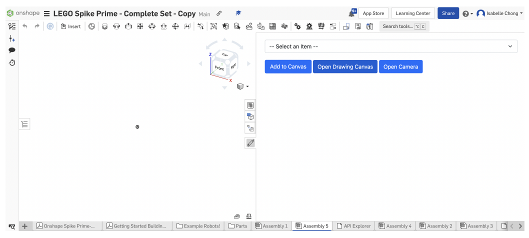
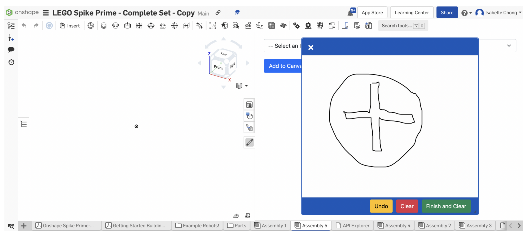

Save User Drawing/Webcam Data to Firebase
Drawing/Webcam Data from User to Classify
Model Classification
Onshape Node.js Application
Python Flask App Request Sender/Receiver
PyTorch Recognition Model
Google Firebase
Send Model Classification to User
Image Data for Recognition

Figure 3-1: General overview of the Onshape application infrastructure

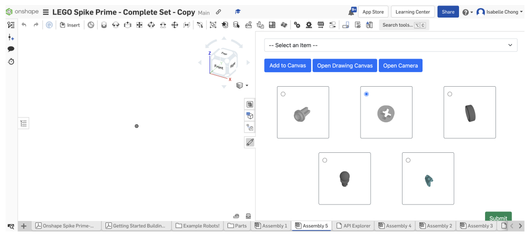
aside any assembly or part file to add components to the user's work space. A walk through of the process from app opening to sketch recognition and insertion of the part into the work space is shown in Figure 3-2, and a walk through of the same process for a webcam image input is shown in Figure 3-3. This app was based off of a base provided by PTC for building Onshape applications [7] in order to obtain the proper Onshape permissions to access document data.



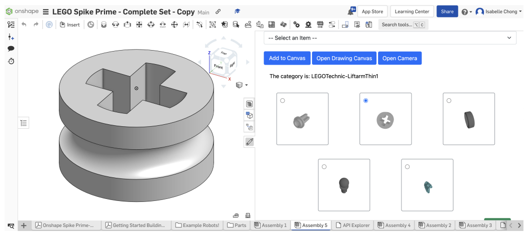
(a) The user opens the application alongside a blank Onshape assembly



(b) The user inputs a sketch of a part they would like to add to their assembly

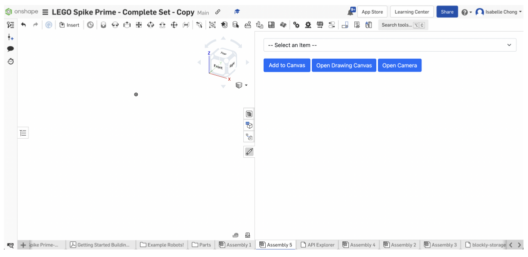


(c) The application provides a list of possible part suggestions for the user to choose from

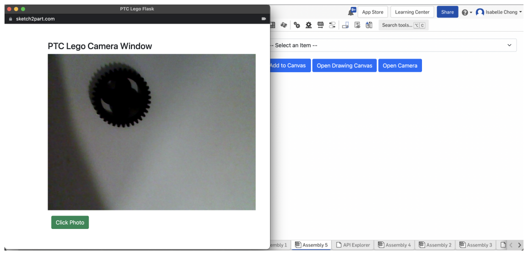


(d) The user has chosen a part and the application adds it to the user's work space to be incorporated into their model

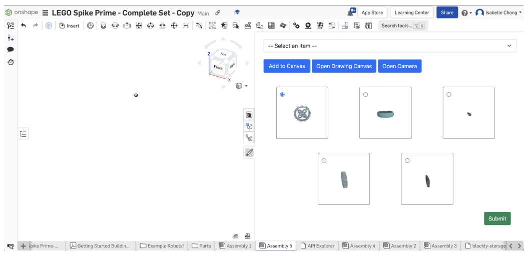
Figure 3-2: A walk through of the sketch path of the Onshape application



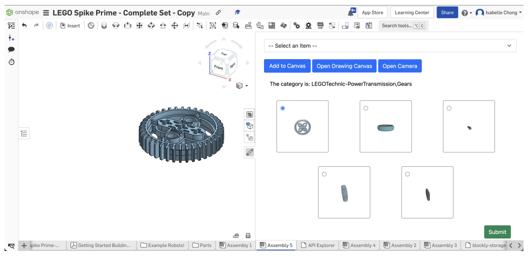
(a) The user opens the application alongside a blank Onshape assembly



(b) The user inputs an image of a part they would like to add to their assembly



(c) The application provides a list of possible part suggestions for the user to choose from


The category is: LEGOTechnic-PowerTransmission,Gears

(d) The user has chosen a part and the application adds it to the user's work space to be incorporated into their model

Figure 3-3: A walk through of the webcam image path of the Onshape application

3.1.2 Database

The database back end of the Onshape application is held in Google Firebase storage. The database provides an intermediate between the recognition models and Onshape application. When a user enters an input via Onshape application, the input data is sent via POST as a PNG file to Google Firebase storage, tagged with a unique ID based on when the request was sent. The Onshape application then sends a GET request to the Python Flask application, which in turn issues a GET request to the Google Firebase to retrieve the input corresponding to the desired ID. While the extensiveness of this process does anecdotally seem to result in somewhat slowed application performance, the decrease in speed is not such that it is detrimental to application function, and using this process also allows us to preserve user sketch data for potential future training applications.

3.2 Recognition Models

Ally attempts to recognize 2 different types of inputs: sketches and webcam images. To do this, data of sketches and photographs of the parts in the dataset must be collected from multiple angles. However, sketch and photograph inputs are very different in nature. To deal with this, two different models are used: a Sketch-A-Net model to recognize and retrieve parts from sketch data, and a YOLOv5 model to recognize and retrieve parts from webcam image data.

3.2.1 Three dimensional sketch feature recognition via Sketch-A-Net

The three dimensional sketch feature recognition algorithm that has been chosen to use for the Ally extension is a variant of the Sketch-A-Net model. Sketch-A-Net is a deep learning CNN primarily used to identify sketches and classify them into different categories. As previously stated, sketch recognition can be a difficult problem because quality of sketches can vary greatly. Furthermore, sketches are usually quite abstract representations of the objects upon which they are based, and may share very few features in common with them. Sketch-A-Net was chosen due to the model's prior successes at sketch identification. The Sketch-A-Net approach uses a model with larger first filter layers, no local response normalization, larger pooling size, higher dropout, and less parameters to achieve good performance on sketch datasets. A modified version of a Sketch-A-Net network is used, the architecture for which is shown in Table 3.1. This Sketch-A-Net is then trained using a cross entropy loss and an Adam optimizer.

3.2.2 Webcam image object recognition via YOLOv5

The webcam image object recognition algorithm that has been chosen to use for the extension is a modified YOLOv5 object detection model. As previously stated, the YOLOv5 network is the latest in the series of YOLO networks, which are one-stage

Table 3.1: Custom Sketch-A-Net architecture

Index	Layer	Type	Kernel Size	Filter Number	Stride	Pad
0	Input	-	-	-	-	-
1	L1	Conv	16x16	64	3	0
2		ReLU	-	-	-	-
3		MaxPool	5x5	-	2	0
4	L2	Conv	7x7	128	1	0
5		ReLU	-	-	-	-
6		MaxPool	5x5	-	2	0
7	L3	Conv	3x3	256	1	1
8		ReLU	-	-	-	-
9	L4	Conv	3x3	256	1	1
10		ReLU	-	-	-	-
11	L5	Conv	7x7	256	1	1
12		ReLU	-	-	-	-
13		MaxPool	3x3	-	2	0
14	L6	Conv	7x7	512	1	0
15		ReLU	-	-	-	-
16		Dropout	-	-	-	-
17	L7	Conv	1x1	512	1	0
18		ReLU	-	-	-	-
19		Dropout	-	-	-	-
19	L8	Linear Fully Connected	-	250	-	-

object detector networks that recognize objects when trained on images with labeled bounding boxes. This model was chosen because of its ability to isolate objects and recognize them, regardless of background content. The YOLOv5 model is composed of a backbone of a CSPDarknet53 with an SPP layer, PANet as the Neck, and YOLO detection as the head [15]. A more detailed diagram of YOLOv5's architecture is shown in Figure 3-4. CSPDarknet53 is chosen as the backbone layer as opposed to common alternative CSPResNeXt50 because it contains 29 convolutional layers 3×3, a 725×725 receptive field and 27.6M parameters, as opposed CSPResNeXt50's only 16 convolutional layers 3×3, a 425×425 receptive field and 20.6M parameters. The SPP layer that follows it up is chosen because it allows us to increase the receptive field and separate out significant context features [2]. Further alterations are performed on the Non-Maximum Suppression (NMS) method on the inference results to make it such that a user receives multiple suggestions per part detected in the input webcam image, not just one. The YOLOv5 model uses Complete Intersection Over Union (CIoU) loss and a SGD optimizer.

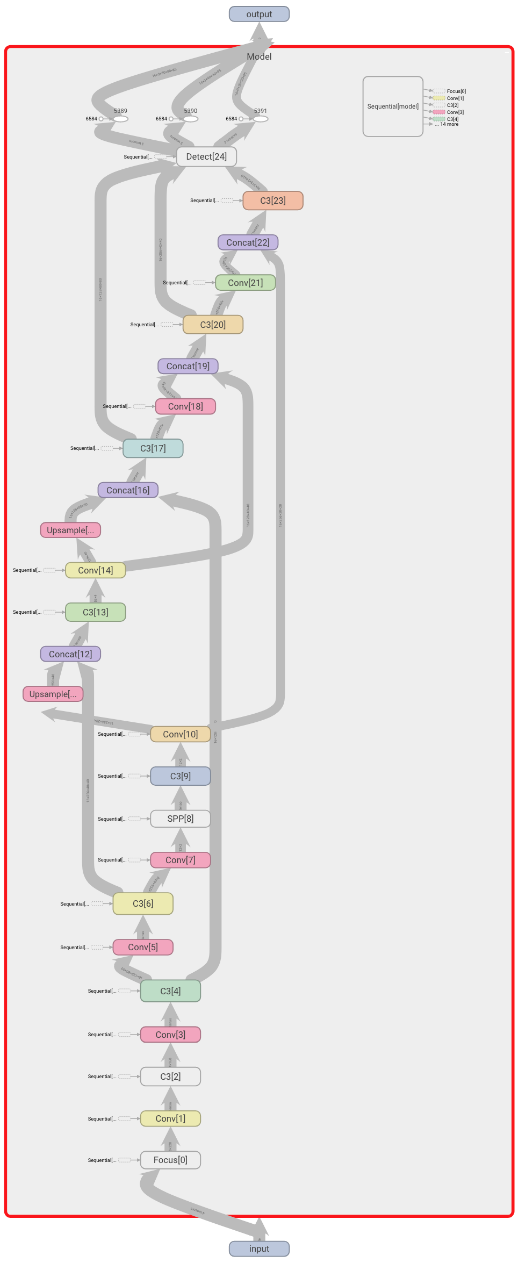


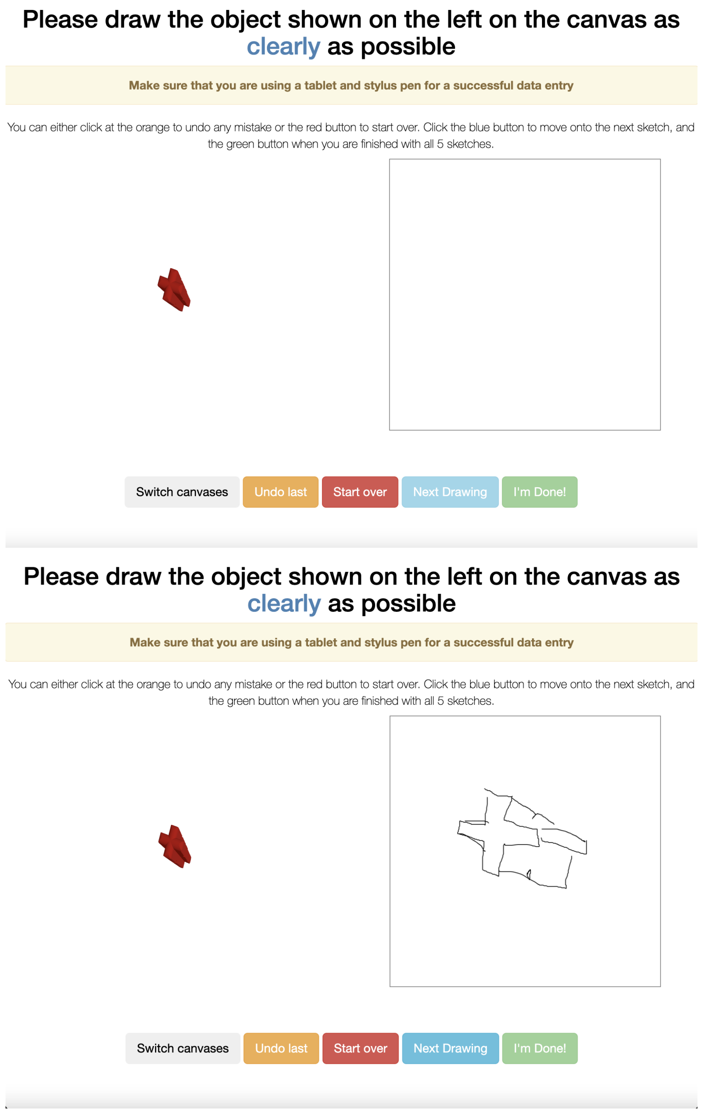
Figure 3-4: Architecture diagram of YOLOv5 model

3.2.3 Experimental setup and data collection

Experiments on the Ally application were run on a pre-existing Onshape CAD dataset of models from the Lego Education Spike Prime kit [4], chosen because it was a dataset from which modular parts could be combined into larger structures and because the models for each part were already readily available on Onshape as CAD models. When attempting to design recognition models to retrieve 3D objects as opposed to 2D images, the main difference, as previously stated, is that a 3D model, unlike a 2D image, may be viewed from any number of dimensions, making the amount of data needed to correctly recognize a 3D model from a 2D representation larger than that needed to correctly recognize a 2D image from a 2D representation. However, a significant challenge to this project was that there were no pre-existing sketch or photo datasets for this particular group of Lego Spike Prime parts. To try to ameliorate this, number of data augmentations and expansions have been implemented in order to try to expand the dataset.

Sketch-A-Net sketch data

To collect data for user sketches of the different parts in the Lego Spike Prime dataset, an Amazon Mechanical Turk (MTurk) experiment (Figure 3-5) was created. This experiment presents users with a selection of five random parts from the dataset, with each part presented at a random angle selected from a list of possible angle presets. The user is then invited to sketch the part as they see it in the canvas provided. Using MTurk within the limited time frame, an average of about 20 sketches per part was aggregated. However, given that larger sketch databases such as the TU-Berlin dataset have about 80 sketches per category, steps were taken to try to extend the dataset. One way this was done was through sketch generation. For each CAD part in the Lego Spike Prime set, using the Blender 3D graphics toolset and a Python script, a series of images of the part at different angles was generated by moving a camera in a circle along a randomized axis. Both the depth and normal views of the models at these angles were then extracted as well (Figure 3-6). Using the Sobel kernel, the gradients for the X and Y directions of both the depth and normal images were found. Taking the magnitudes of the gradients for both of these images, if either magnitude was greater than a certain threshold (in this case, a threshold of 0.1), that pixel would be considered a location of discontinuity and marked as part of the line drawing of the sketch.


Please draw the object shown on the left on the canvas as clearly as possible

Make sure that you are using a tablet and stylus pen for a successful data entry

You can either click at the orange to undo any mistake or the red button to start over. Click the blue button to move onto the next sketch, and the green button when you are finished with all 5 sketches.

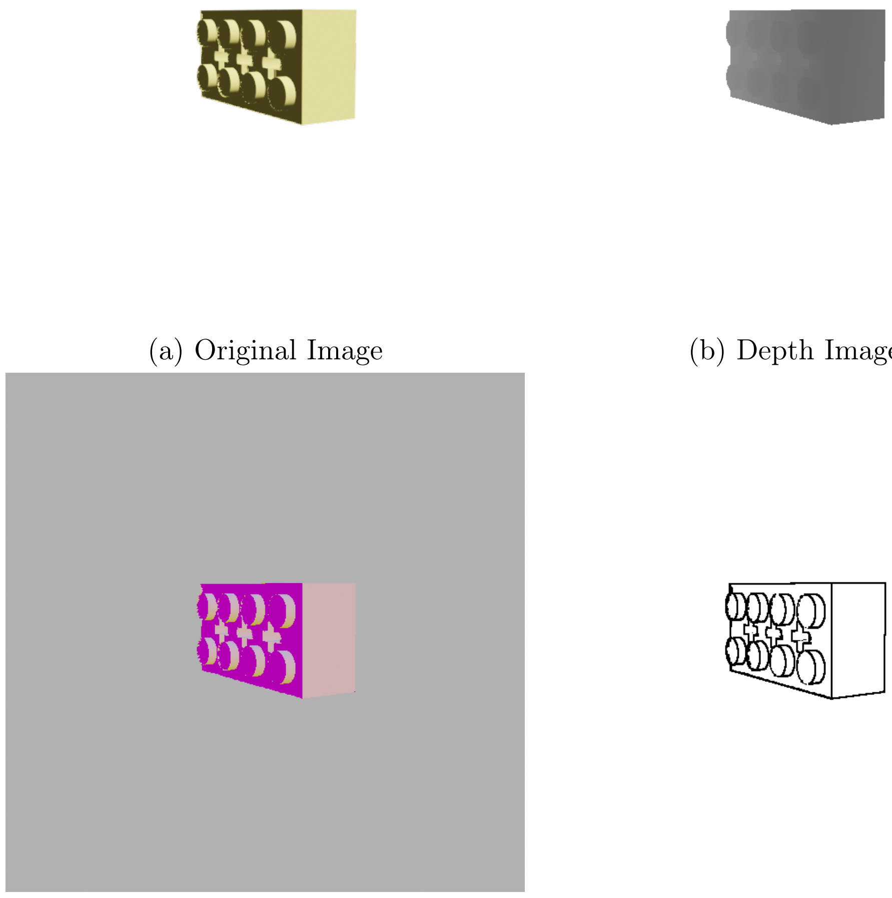
Please draw the object shown on the left on the canvas as clearly as possible

Make sure that you are using a tablet and stylus pen for a successful data entry

You can either click at the orange to undo any mistake or the red button to start over. Click the blue button to move onto the next sketch, and the green button when you are finished with all 5 sketches.

Figure 3-5: The Amazon Mechanical Turk experiment setup

To further expand the contents of the sketch dataset, a series of augmentations was also applied to each sketch, both MTurk gathered and CAD generated, using the Augmentor Python library. Table 3.2 shows which augmentations used, as well as their magnitude and frequency. Augmentated images were generated until each



(a) Original Image (b) Depth Image

(c) Normal Image (d) Generated "Sketch"

Figure 3-6: The "LegoBricks-Rectangular" part shown in its original 3D model, its depth image, its normal image, and its generated "sketch"

category had a total of 2000 sketch images (Figure 3-7).

Table 3.2: Sketch augmentations

Augmentation Type	Probability	Amount
Zoom	0.5	1.0x-1.1x
Random Distortion	0.7	4x4 Grid, Magnitude 8
Rotate Random 90	0.5	-
Rotate	0.5	25

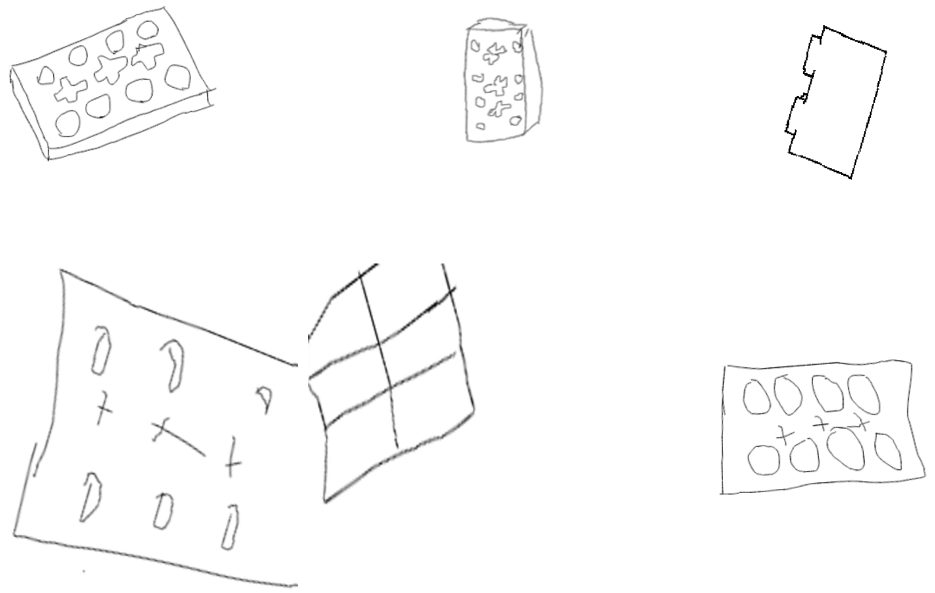


Figure 3-7: A series of training sketches for the "LegoBricks-Rectangular" part post-augmentation

Before any augmentations or sketch generation, 20% of the initial sketch images were partitioned off to serve as test images. After augmentation and sketch generation, 20% of the altered images were partitioned off to serve as validation images. The remaining 80% served as training data.

YOLOv5 webcam image data

To collect webcam image data, 20 images were taken of each part in the Lego Spike Prime dataset from a variety of angles on a plain white background. Proper bounding boxes were then defined for each image and labeled with the correct category by hand using the Roboflow software. Generated "photos" were then also added using the aforementioned 3D CAD model images. This set of generated images was simple to annotate, since the backgrounds of the generated webcam images are completely plain and thus easy to parse via Python script. A comparison of the generated data vs. the photo data is shown in Figure 3-8. These images were then also added to the Roboflow dataset for augmentation. The Roboflow built in augmentor adds 3 augmented outputs per input image, and the selection of augmentations chosen is shown in Table 3.3.

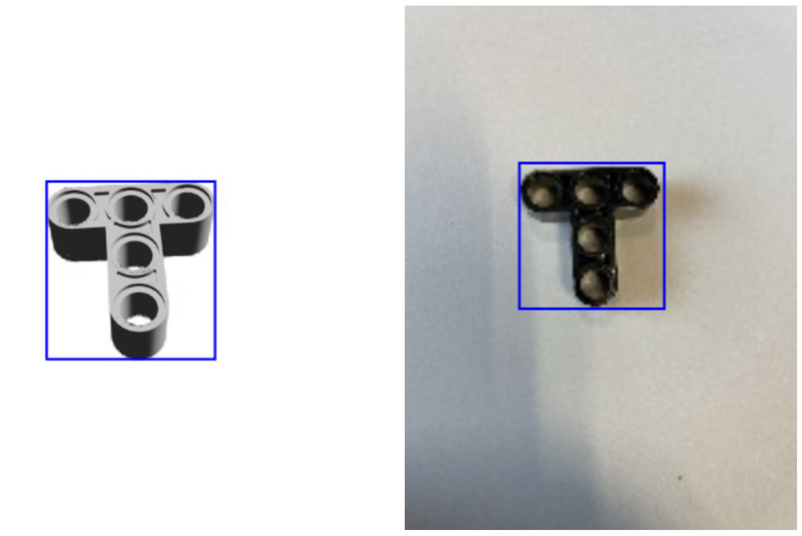

(a) A generated part (b) A photo of the actual part

Figure 3-8: A comparison of the generated part vs. the actual photo data of the part, annotated in Roboflow

Roboflow partitions 10% of the pre-augmentation input images to be test images and 20% of the remaining pre-augmentation input images to be validation images, augmenting the remaining 70% to use as training data (Figure 3-9).

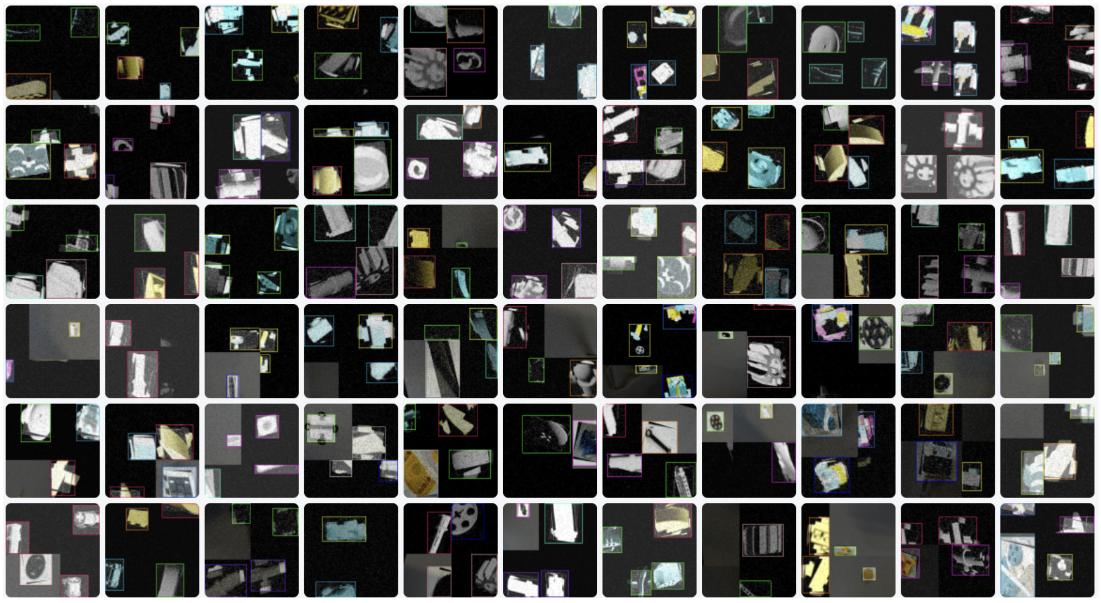


Figure 3-9: A selection of the YOLOv5 training images post-augmentation

Table 3.3: Webcam image augmentations

Augmentation Type	Details
Flip	Horizontal, Vertical
90° Rotate	Clockwise, Counter-Clockwise
Crop	0% Minimum Zoom, 20% Maximum Zoom
Rotation	Between -15° and +15°
Shear	±15° Horizontal, ±15° Vertical
Saturation	Between -25% and +25%
Brightness	Between -25% and +25%
Exposure	Between -25% and +25%
Blur	Up to 5px
Noise	Up to 5% of pixels
Mosaic	Applied
Bounding Box: Flip	Horizontal
Bounding Box: 90° Rotate	Clockwise, Counter-Clockwise
Bounding Box: Crop	0% Minimum Zoom, 20% Maximum Zoom
Bounding Box: Rotation	Between -15° and +15°
Bounding Box: Shear	±15° Horizontal, ±15° Vertical
Bounding Box: Brightness	Between -25% and +25%
Bounding Box: Exposure	Between -25% and +25%
Bounding Box: Blur	Up to 10px
Bounding Box: Noise	Up to 5% of pixels

Chapter 4

Results

After augmenting and annotating the sketch and webcam image data, the two models were then trained and tested. The results of those experiments are shown below.

4.1 Sketch-A-Net

Running the Sketch-A-Net model for 20 epochs, a top-one best training error of 3% and a top-one best validation error of 17% are achieved. The loss and errors over time for the Sketch-A-Net model are shown in Figures 4-1 and 4-2. However, after training with the test set data, the percent error increases to a top-five error of 78%, as compared to a by-chance error of 90%, for an accuracy of 22% as compared to 10% by chance. This is an improvement, but accuracy is likely limited due to the smaller number of unique sketches per category, an average of 20 per part. As previously stated, the TU-Berlin database has 80 sketches per category. Due to time constraints on this thesis, the amount of data able to be gathered via MTurk was limited. With a longer time frame, more user sketches could be aggregated, which would improve model performance by increasing the varieties of sketches that are available per part for the model to train on.

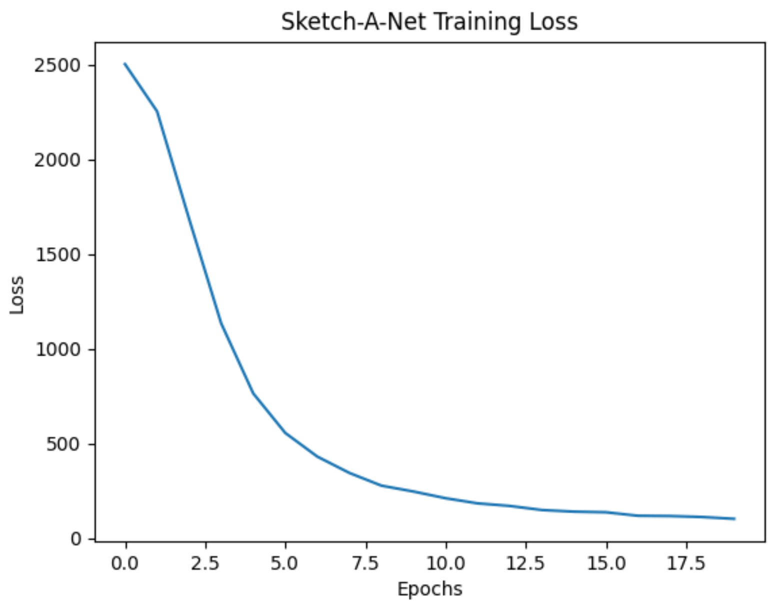


Figure 4-1: Loss for the Sketch-A-Net model over 20 epochs

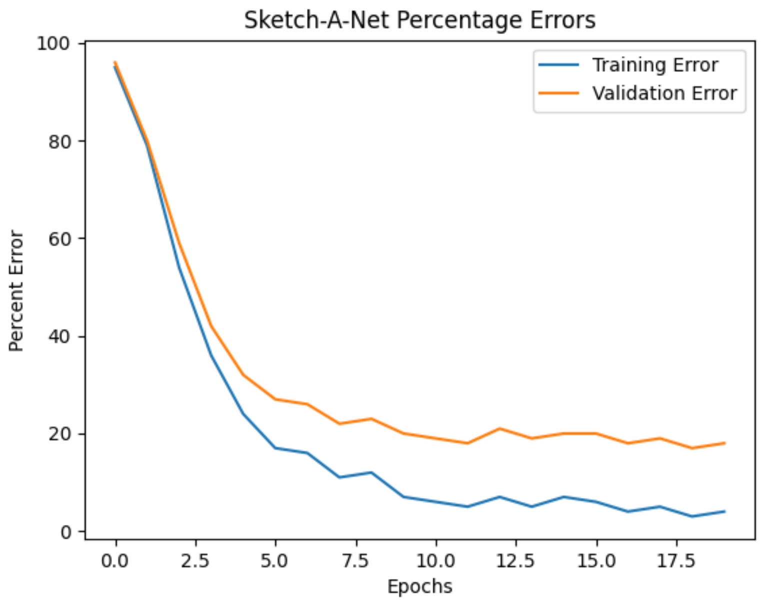


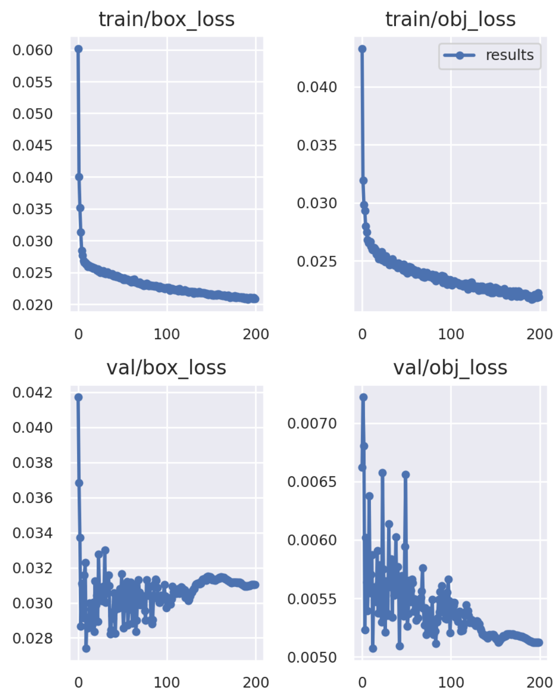
Figure 4-2: Percentage errors for the Sketch-A-Net model over 20 epochs for training and validation sets

4.2 YOLOv5

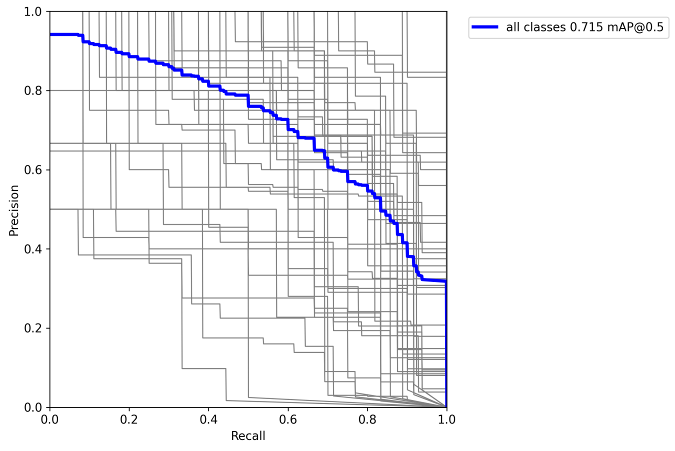
Running the YOLOv5 model for 200 epochs at an image size of 608x608, the loss graphs, PR curve, and compatibility matrix shown in Figure 4-3 are achieved. As can be seen from the PR curve, the mean average precision with an Intersection Over Union (IoU) threshold of 0.5 of the YOLOv5 model was 0.715. The confusion matrix also shows a relatively low rate of false identifications for each category. Overall, the top-five accuracy of the YOLOv5 model on the set of test data was 94%.

4.3 In-App Performance

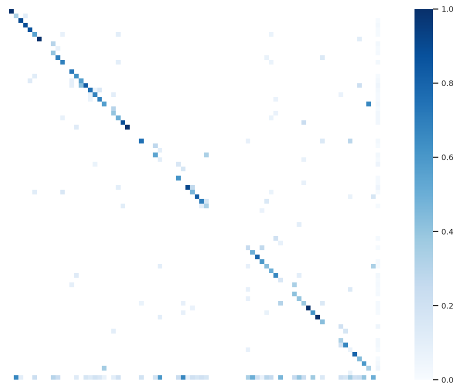
In-app, both the Sketch-A-Net model and YOLOv5 model are able to seamlessly integrate with the Python Flask application. While there is a slight time delay between when the user makes the request and when the model and application are able to output the results of the user's query, this delay is not unreasonable, and is most likely mainly due to the large number of requests between applications that need to be made in order for the application workflow to function. For instance, if the Onshape web application API could be reconfigured such that the webcam could be directly activated from within an app, an additional request to the Python Flask App could be removed.



(a) YOLOv5 losses



(b) YOLOv5 PR curve



(c) YOLOv5 confusion matrix

Figure 4-3: YOLOv5 result metrics

Table 4.1: YOLOv5 Categories, listed in order of Confusion Matrix input (Figure 4-3C); Note that some of these names are repeats for formatting purposes

Part Category	Part Category
CrankshaftAssembly	LEGOTechnic-Connectors-Axle
DEMO	LEGOTechnic-Connectors-Liftarm
Example1-SwayingRobot	LEGOTechnic-Connectors-Pins
Example2-TinyDogBot	LEGOTechnic-Liftarm
Example3-DriveBot	LEGOTechnic-LiftarmThin
LEGOBricks-Angled	LEGOTechnic-LiftarmThin1
LEGOBricks-Circular	LEGOTechnic-LiftarmThin2
LEGOBricks-Circular,ClearShield	LEGOTechnic-Liftarms,Angled
LEGOBricks-Circular,Eyes	LEGOTechnic-Liftarms,Buildplate
LEGOBricks-Circular-ClearShield	LEGOTechnic-Liftarms,Buildplates
LEGOBricks-Circular-Eyes	LEGOTechnic-Liftarms,Frames
LEGOBricks-Curved	LEGOTechnic-Liftarms,H
LEGOBricks-Curved,Windscreens	LEGOTechnic-Liftarms,L
LEGOBricks-Curved-Windscreens	LEGOTechnic-Liftarms,PinConnectors
LEGOBricks-Rectangular	LEGOTechnic-Liftarms,T
LEGOMotors-LargeAngular	LEGOTechnic-Liftarms-Angled
LEGOMotors-SmallAngularMotor	LEGOTechnic-Liftarms-Buildplate
LEGOSensors-Color	LEGOTechnic-Liftarms-Buildplates
LEGOSensors-Distance	LEGOTechnic-Liftarms-Frames
LEGOSystem-SpikePrimeHub	LEGOSensors-Force
LEGOTechnic-Liftarms-H	LEGOTechnic-Liftarms-L
LEGOTechnic-Axle,DoubleConnector	LEGOTechnic-Liftarms-PinConnectors
LEGOTechnic-Axle-DoubleConnector	LEGOTechnic-Liftarms-T
LEGOTechnic-AxleAngleConnector	LEGOTechnic-Panel,Curved
LEGOTechnic-AxleBush	LEGOTechnic-Panel,Plate
LEGOTechnic-Axles,Linkw-oStoppers	LEGOTechnic-Axles
LEGOTechnic-Axles,Linkw/oStoppers	LEGOTechnic-Panel-Curved
LEGOTechnic-Axles-Linkw-oStoppers	LEGOTechnic-Panel-Plate
LEGOTechnic-Connector,AxleFlexible	LEGOTechnic-Pins
LEGOTechnic-Connector,BionicleTooth	LEGOTechnic-Pinsw-Axle
LEGOTechnic-Connector-AxleFlexible	LEGOTechnic-Pinsw-AxleConnector
LEGOTechnic-Connector-BionicleTooth	LEGOTechnic-Pinsw-AxleThin
LEGOTechnic-Connectors,Axle	LEGOTechnic-Pinsw-BallHitch
LEGOTechnic-Connectors,Liftarm	LEGOTechnic-PowerTransmission,Gears
LEGOTechnic-Connectors,Pins	LEGOTechnic-PowerTransmission,Pulleys

Table 4.2: Table 4.1 Continued

Part Category	Part Category
LEGOTechnic-PowerTransmission,Rack	LEGOTechnicCaster
LEGOTechnic-PowerTransmission-Gears	RadialEngine
LEGOTechnic-PowerTransmission-Pulleys	SpikePrime-StandardizedCopy1
LEGOTechnic-PowerTransmission-Rack	TechnicBeamTriangleThin
LEGOTechnic-Wheels	

Chapter 5

Further Work

In general it seems that the biggest challenge to model accuracy for both Sketch-A-Net and YOLOv5 is lack of enough variegated data. Due to time constraints on this thesis, a truly large number of sketches and images comparable to well-known datasets such as Sketchy, TU-Berlin or COCO MS were unable to be compiled. However, performing augmentations to data to expand the dataset has been shown to be promising, and current model accuracy shows both Sketch-A-Net and YOLOv5 provide improvements to query accuracy over pure chance. For the sketch inputs, simply running the MTurk experiment for longer to accrue more data could be sufficient. Another thing that could be helpful for the sketch inputs could be to specifically choose a number of "representative angles" for the MTurk from which users could draw their training sketches to better represent what a typical user thinking of a part and then drawing it to query the model would create.

More extensive data would also be useful to allow the YOLOv5 model to function at the current accuracy rate in a wider variety of environments. Currently, the YOLOv5 model functions best on images with plain backgrounds. However, in the use case of a user capturing images to query the dataset, the background may not always be consistently plain. Thus, collecting data of the different parts within the Lego Spike Prime dataset against differing backgrounds could help improve model accuracy in a greater variety of locations.

In addition to improvements on the current models, future research might focus on improving the functionality of Ally by adding features such as partial sketch recognition, which is also proposed as a part of Paper Dreams.

Finally, the current Ally application is specifically tailored for the Lego Spike Prime dataset. Future research could be done into allowing users with different Onshape datasets to be able to train and use the Ally app for their particular Onshape documents. This would increase the application's usability in industry and as a wider design application.

Chapter 6

Conclusions

Ally is a working prototype of an application by which human users can query a set of Onshape CAD parts in order to collaboratively build a model with a computer. Using a Sketch-A-Net and a YOLOv5 model, initial results are achieved with an accuracy rate better than chance for both sketch and image querying, the performance of both which are suspected would improve with an increased amount of training data input. The applications for Ally include, most prominently, the creation of "digital twins," which are digital copies of real-life 3D builds. This application is useful in both the design and manufacturing industries. Future research might include adding partial sketch recognition capabilities or re-configuring the application for usability on a wider variety of Onshape documents and CAD datasets. Overall, Ally has proven to be a promising prototype application for human-AI collaborative design.

Bibliography

[1] Guillermo Bernal, Lily Zhou, Erica Yuen, and Pattie Maes. Paper dreams: Real-time human and machine collaboration for visual story development. *Generative Art Conference XXII*, 2019.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[3] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):31:1–31:10, 2012.

[4] O. Gervais. *LEGO Spike Prime - Complete Set.* Onshape, 2020.

[5] Glenn Jocher. ultralytics/yolov5. https://github.com/ultralytics/yolov5, 2022.

[6] Gregory R. Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[7] Onshape. app-gltf-viewer. https://github.com/onshape-public/app-gltf-viewer, 2021.

[8] Product Development Platform. Onshape, a p. t. c. b.

[9] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Trans. Graph.*, 35(4), jul 2016.

[10] Jacob Solawetz. Yolov4 - an explanation of how it works, Mar 2022.

[11] Fang Wang, Le Kang, and Yi Li. Sketch-based 3d shape retrieval using convolutional neural networks. *CoRR*, abs/1504.03504, 2015.

[12] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas S. Huang. Unitbox: An advanced object detection network. *CoRR*, abs/1608.01471, 2016.

[13] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales. *Sketch-a-net that beats humans.* In British Machine Vision Conference (BMVC, 2015.

[14] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. volume 34, pages 12993–13000, 02 2020.

[15] Xingkui Zhu, Shuchang Lyu, Xu Wang, and Qi Zhao. Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios. *CoRR*, abs/2108.11539, 2021.