Skeggøx: Automatic Scoring for Accuracy of Thrown Axes

Mitchell Karchemsky
UC Berkeley
School of Information
mkarch@berkeley.edu

Lynn Marciano
UC Berkeley
School of Information
marcianolynn@berkeley.edu

Parham Motameni
UC Berkeley
School of Information
pmotameni@berkeley.edu

ABSTRACT

Visually judging the location of an objects position is a difficult challenge and one that is fraught with heuristics and human error. Skeggøx aims to be an automated machine learning computer-vision based approach which will automatically score a users thrown axe against a regulation board, and eliminate any ambiguities in points scored.

INTRODUCTION

The International Axe Throwing Federation (IATF) is an organization that standardizes a rule set for the competitive accuracy sport of axe throwing. With 10,000+ active members over 4 continents, the sport is amassing competitors at an ever-growing rate. The competitive portion of this sport is in which party can score a higher point value by scoring in more accurate zones on the board. These regulations are expanded upon in the Rules of the Sport Section . A key consideration is that currently human judgment does all scoring and may even require a third-party adjudicator to verify that the point value is legal. This system is slow, arduous, and can be filled with bias and misrepresentation. We aim to solve this problem by automating the scoring system with a computer vision and machine learning approach.

RULES OF THE SPORT

The International Axe Throwing Federation defines a set of Rules [7] that define fair play for axe throwing. We will focus specifically on the subset of rules relating to scoring an individual axe throw.

1. An axe that lands within the center black-outline ring is worth 5 points
2. An axe that lands within the middle red-outline ring is worth 3 points
3. An axe that lands within the middle blue-outline ring is worth 1 points
4. An axe that lands within either of the green-dots is worth 7 points
5. The paint of each defined sub-circle is considered inclusive to the higher point total. (i.e. The blade being completely in the black-outline paint would count for 5 points)
6. All scoring is based on where the majority of the blade lands and stays in the target
7. Scoring an axe in either of the green-dots is an exception to the majority rule and as long as any part of the axe blade is breaking the green paint, full points are awarded

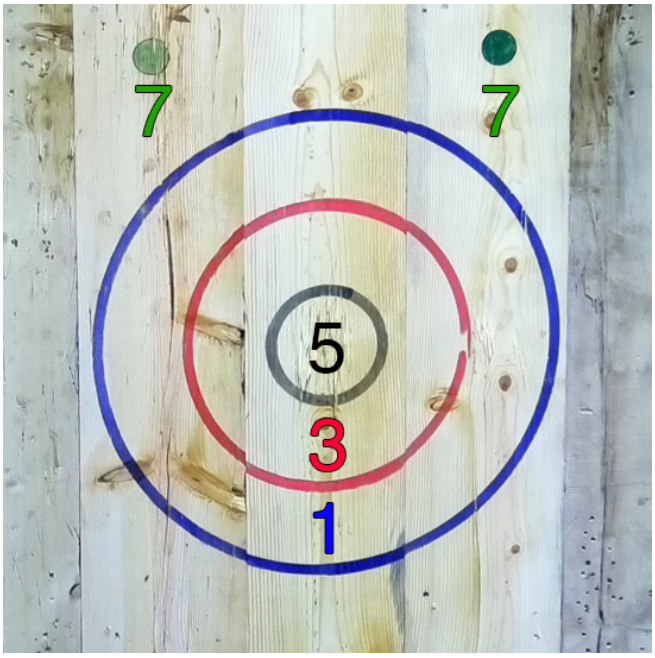

Figure 1. A standardized IATF board with their associated point values adjacent to their scoring areas. If an axe lands within a smaller subcircle, the higher point value is awarded.

8. Any axe that lands outside the shooting target or not inside the green paint, is worth 0 points

This subset of rules is non-exhaustive to all the various intricacies of the sport and focuses only on the portion of scoring which happens once an axe has landed within the target, we will not consider other scoring considerations such as foot-faults or other illegal throwing constraints.

Special Rules Considerations

As noted in the Rules 6 and 7, there are specific considerations for when an axe lands between two rings, or when aiming for the green "clutch" dots. If the axe lands between two sub-circles, wherever a majority of the axe blade lands determines the score of the axe. The amount of the axe blade which is submerged in the wooden board, and accounts for both the left and the right side of the blade calculates this majority.

DESIGN RATIONAL

Skeggøx is a hybrid-compute implementation where we use both cloud and edge systems to create a harmonic implemen-

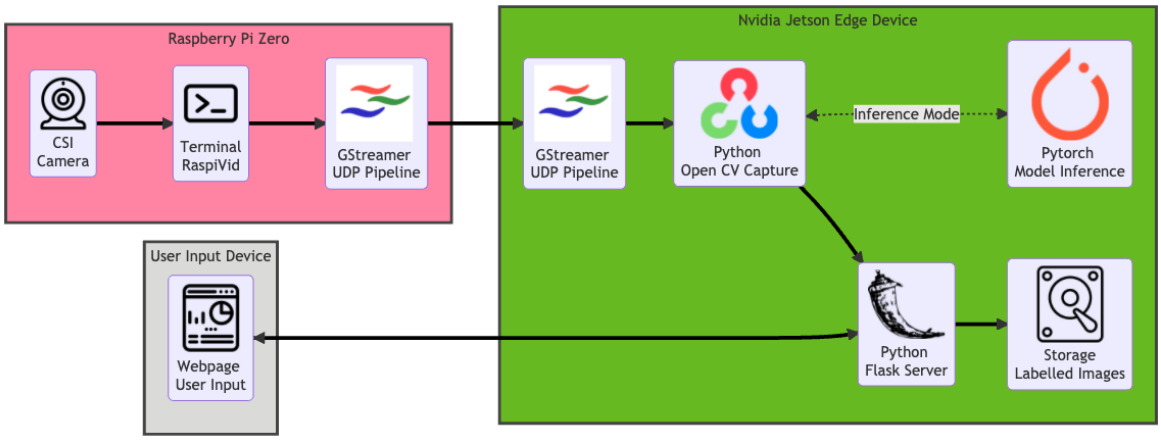

Figure 2. The "edge" inference and data collection pipeline. Each colored rectangle describes a separate computing resource that is utilized. When in Data-collection mode, the User Input device allows live data labelling and capture; Inference-mode provides a live view and inferred score from our model.

tation. In the following section, we describe the affordances, motivations, and deciding factors in our end-to-end model.

No Interventions

Skeggøx aims to be a system completely free of interventions where the players will not need to accommodate for any external factors to use this system. This means that no requirements on the player's axes, style, board setup, etc. will be stipulated such that the most intuitive system can be created. These challenges are further described in the Data Capture Challenges and Considerations Section.

Single Persepctive Camera

To minimize barriers to entry, we will use a single camera for this task of automatic classification. We are aware that there are obvious challenges to not having a full-depth perception (See Figure 3), but seek to evaluate the efficacy of a single perspective system in this paper.

Live Clasisfication

The end goal of Skeggøx is to be given an input video stream of an axe-throwing lane and be able to automatically classify frames from the video stream. This should be done with no external involvement other than accessing the user interface for the inference view of the model.

Cloud Computation

Training a deep learning neural network model and optimizing its performance is a substantially resource-intensive and computationally expensive operation. Since our system is reliant on the processing of images, a convolutional neural network with multiple layers will also add to the complexity of the training process. We use Amazon Web Services (AWS) to instantiate an Elastic Cloud Computing resource with a dedicated Nvidia T4 GPU for model training. With this more powerful machine, we can train a neural network and save the appropriate weights and values, which can then be used for inference at the edge.

Edge Systems

To run inference in real-time, we utilize a variety of edge systems to process information in real-time. Our main processing interface is the Nvidia Jetson Nano which serves as the model inference gateway in our pipeline. We describe this systems implementation in the Implementation section, but we provide a summary visualization in Figure 2

DATA

Our data comprises approximately 3,000 labeled images of target boards, collected over the course of 8 weeks. As shown in Figure 2, there is a sophisticated layout to the systems that are involved in the collection and inference pipeline. In this section, we will discuss the data collection and aggregation portion of the system and some challenges faced.

Data Capture Challenges and Considerations

There are many considerations that are at play when considering the challenges of recognizing and correctly labeling our targets. The sections below list some difficulties but are not an all-encompassing enumeration of the issues we faced.

Real-time Data Capture

We created a front-end interface for our system which allows for us to not only capture images of the axes in the board but label them with their appropriate values in real-time (See Figure 7). This is essential, as we can only discover scoring discrepancies at the time of the event; that if the axe is disputed (see Rules of the Sport section), it would be difficult to distinguish from the photo alone. If the decision was to instead record video and then present it to some crowd-labeling system, someone could incorrectly label many of the axes which are between two scoring sections.

Target Board Variances

As the target board is made out of natural materials, there are natural patterns and deformations which are non-uniform. Qualities such as grain pattern and direction, natural discoloration, imperfections in the wood itself, such as knots, are some of the different factors which make the wooden boards

Label	Count
5 Points	1588
3 Points	610
0 Points	291
1 Point	148
Right Clutch	82
Empty Board	82
Left Clutch	74

Table 1. A summary of the number of labelled image data that was used in creation and training of the model

different from each other; no boards will ever be identical. These non-homologous characteristics add more challenges to our system.

Alongside natural variances with the boards, there are also natural degradation factors to the matter of play that must be considered. For example, the boards are only replaced after they have become so worn out that an axe penetrates through the entire depth of a board. This means that a new board and a deplenished board will look visually different and lead to other variability in our study. An additional factor to consider is that boards will go through discoloration processes as the boards are allowed to be sprayed with water periodically, creating a different visual characteristic of the board.

Physical Axe Variances

In the IATF there are some explicit regulations [7] over the type and qualities of a player's axe but there is much leeway and flexibility to allow players to develop their own preference and techniques including but not limited to the axe haft and head shapes. This means that unlike other sports where the objects of interest are relatively uniform [1], our system needs to generalize to the idea of an axe as opposed to tracking a more uniform shape or color key mapping

Axe Throw Depth and Angle

As described in the Rules of the Sport section, as long as a majority of the axe blade is within a subsection, it is awarded that many points. This means that the blade of the axe can find itself in a variety of positions and angles which all influence how large the majority/share of the axe is within the appropriate scoring zone. This is visualized in Figure 3.

Judging Variances

Each axe is scored by the opponent of the thrower; this means that a human will look at the axe and determine whether or not they believe it is worth whichever point values. While there is not an antagonistic culture and there are mechanisms in place to challenge decisions, we only called a neutral adjudicator upon when there is a disagreement between the two players on a perceived score. This means that sometimes an axe may seem to be incorrectly labelled or inversely, an axe that looked clearly to be in one point range was actually in another when measured (See Figure 3). As per the principals we laid out in the Design Rational, we did not want to impede on the players and ask for arbitration or interfere with any of the calls which were made on the field as we were capturing data.

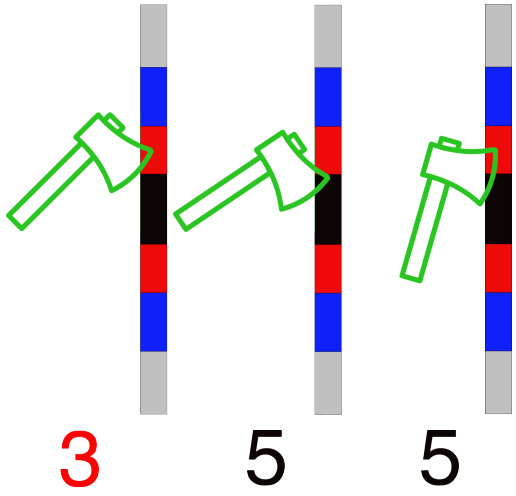

Figure 3. A crossesctional comparison of different scores based on angle and depth of landed blade of axe. In the leftmost axe, all portions of the blade are fully within the red section, denoting that it is worth 3 points. In the middle, the axe is barely touching the red section with a large majority in the center bullseye. Far Right, the axe has a slight majority (2%) favor to the bullseye, meaning this axe would be labelled as 5 points.

Data Labels

There are 7 mutually exclusive and collectively exhaustive classes to our data; their distributions are described in Table 1. Images of axes are labeled at the time of image capture as according to the visual in Figure 1 and stipulations outlined in the Rules of the Sport section. Although the top two "clutches" are labeled separately as `Left Clutch` and `Right Clutch`, the point values they describe are the same; this is to help distinguish the locations for modeling purposes. We also label `empty` images of boards that do not have any presence of an axe on the board. This ensures that our system will not incorrectly attempt to label an image with a score when there is no value possible.

Data Bias

There is an inherent bias in our dataset as we captured images from skilled players who attend a weekly league. This means these players are far more accurate and will have a disproportionately larger bias towards hitting the optimal point values and less of the 0 points or 1 point sections of the board. While we acknowledge this as a point of concern, we will show in the Data Bias Concerns that we are not concerned about the imbalanced data classes

IMPLEMENTATION

In this section we will describe the variety of hardware and software systems which were utilized in the execution of this project.

Hardware

Three main hardware devices were utilzed in order to make a seamless experience both for data-capture and inference. A portion of the full system can be seen in Figure 5 and their features afforded are described below.

Figure 4. An annotated view of an axe throwing lane. In pink is the Raspberry Pi camera, in green, the distance between the camera and targets; approximately 117 inches.

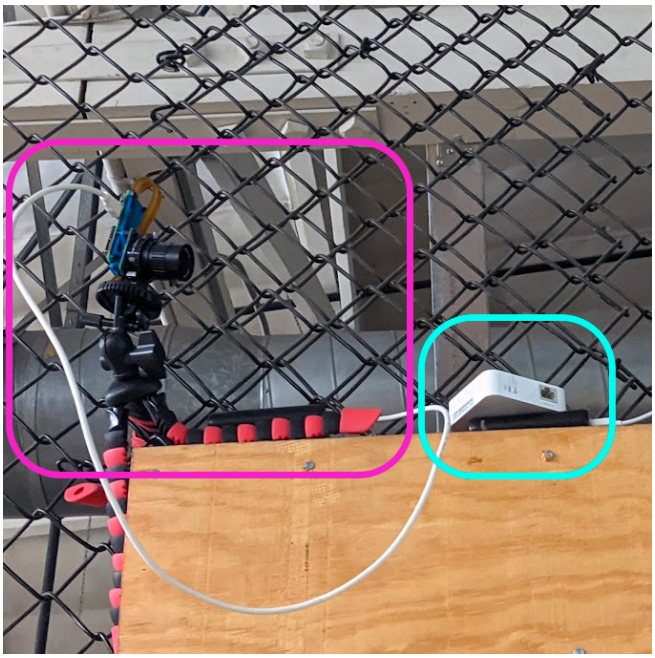

Figure 5. An annotated view of the Raspberry Pi Camera (pink) and WiFi router (teal)

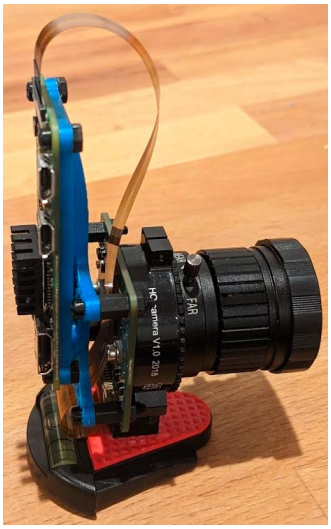

Figure 6. The Raspberry Pi based image capturing device. The portrait-oriented circuitboard with the black heatsink is the Raspberry Pi Zero. Connected with a flexible flat cable to the High Quality Camera module on the far right. In-between in blue plastic is the custom designed and 3d printed mount.

Camera Capture: Raspberry Pi Zero

Because of the relatively hazardous nature of axes being thrown in any camera vicinity, we use a secondary device to serve solely as the video capturing mechanism. A Raspberry Pi Zero with an HQ Camera[12] was used to stream live video data. The camera has a 12mp resolution with a 16mm focal length lens. We capture a 1024x768 resolution image due to space and network constraints. To make our system more compact, we 3d printed a custom mount which allowed for all components to be fixed relative to each other. We can see the physical setup of this system in Figure 6. We instantiate a video stream from the device through the command-line program `raspivid` [11], this is then piped to a GStreamer [5] pipeline which serves an h264 encoded video stream via UDP to an IP address. The device which then retrieves this video stream is described in the next subsection.

Video Processing: Nvidia Jetson Nano

As processing and running inference on images is a computationally intensive task, we utilized a powerful edge device in the Nvidia Jetson Nano [10] to receive, infer, and serve our video stream. A GStreamer pipeline accepts a UDP stream and transforms it to be captured and processed with OpenCV [2]. OpenCV captures individual frames from the GStreamer pipeline and serves individual JPG images to a python-based Flask server [13].

Human Interface: Web-based User Interface

As we utilize a web framework through Flask, we are given the flexibility of a web-based front end which allows for all modern mobile devices to connect to our service and either capture or see inference in real-time. In "Data-capture" Mode (Figure 7), Flask serves a webpage that displays the video stream in a window alongside a set of labeling buttons which will capture the frame, label the data with the selected button,

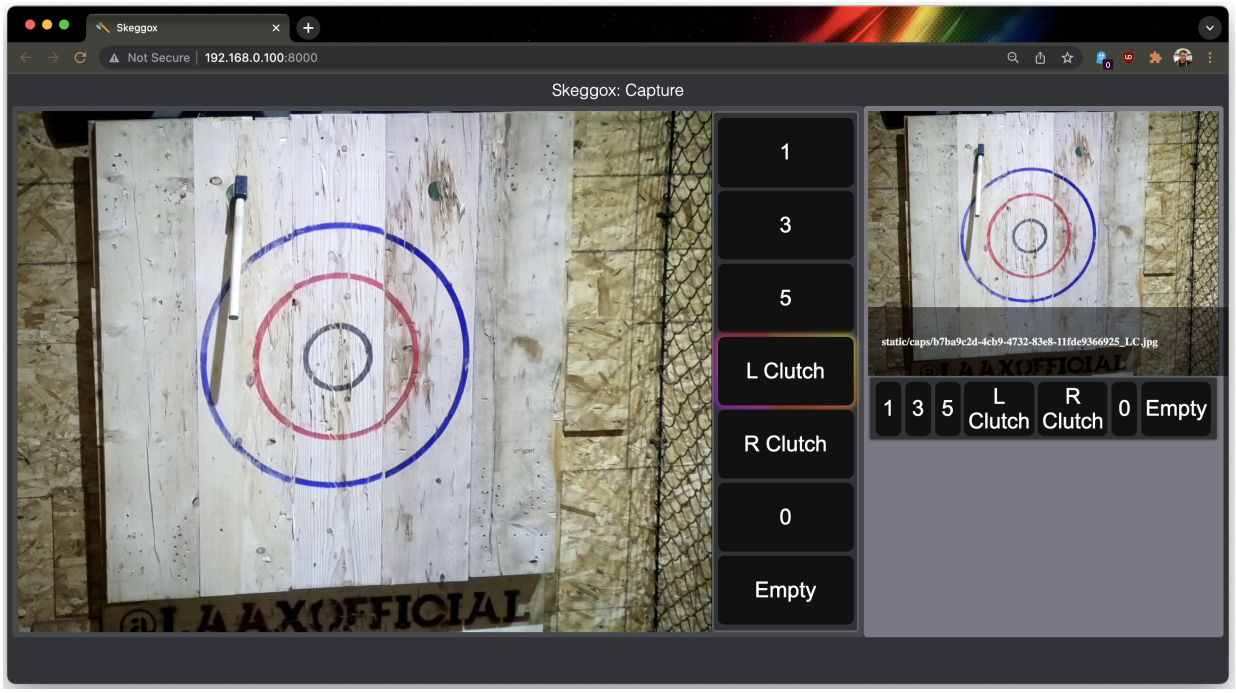

Figure 7. The web-based data capture interface. Live-streamed images are displayed on the left half of the screen, with large buttons which allow a user to simultaneously capture and label an image. On the right side, is the last captured image and a set of buttons which allow for the value to be relabelled in case of a redetermination of the proper score.

and store it for future retrieval, and if necessary, there is an ability for the user to re-label the previously captured data. In the "Inference" Mode, OpenCV will send the captured frame to our trained model for the prediction, and retrieve the predicted label; the inferred value is then augmented onto the captured frame and served to Flask. Flask serves as a webpage for a user to access from any mobile device. This pipeline can be seen in Figure 2

Networking: WiFi Router

To network all of our devices, we used a TP-Link mobile router [16]. This router was chosen as it was a standalone device that could be powered via a USB power bank. All networked devices communicate via 2.4ghz wifi as the Raspberry Pi's built-in wireless adapter is limited to 2.4ghz. As we worked in an environment that had many wireless devices and conflicting access points, the router was placed as close as possible to the Raspberry Pi as this minimized wireless RSSI loss (Figure 5). Other connecting devices had stronger antennae than the PCB antennae of the Raspberry Pi and only experienced moderate degraded signal quality by the distance from the router.

Physical Setup

Repeatable setup is key to producing the most reliable dataset for any machine learning problem. By separating the camera capture system as described in Section Camera Capture: Raspberry Pi Zero, we can place our smaller setup in a more optimal, but more remote location. Our Camera was positioned on a flexible tripod attached to an existing fence approximately 117 inches from the target and angled at approximately 25°downwards, facing the target. We can see this setup in Figure 4. A portable USB power bank powered the Raspberry Pi system. While we did our best to set up the system in the most repeatable way, we were not able to perfectly replicate the pitch, yaw, roll, as well as x, y, and z positions when visiting the facility week over week. To offset this variation, we describe a data normalization technique as in the Image Pre-processing section.

AUTOMATIC CLASSIFICATION SYSTEM

Datasets

We created two separate training environments as per our two different types of data: "Unmodified" and "Normalized". We consider an image to be "Normalized" if it has undergone the automated scale and crop procedure as noted in the Image Pre-processing section. We used both datasets in separate training instances; their performance is evaluated and recorded below.

Data Splits

Our full dataset was split into an 75-10-15 split where 75% of our data was used for training the model, 10% of the dataset was left for validation during model training, and a final 15% was left for testing accuracy outside of training.

Image Pre-processing

Because of the physical limitations of the setup, we implemented an automatic position normalization technique to help create more consistent data from our images (See Figure 8) This technique implements an automated perspective scale and crop which corrects for any rotation and skew or other

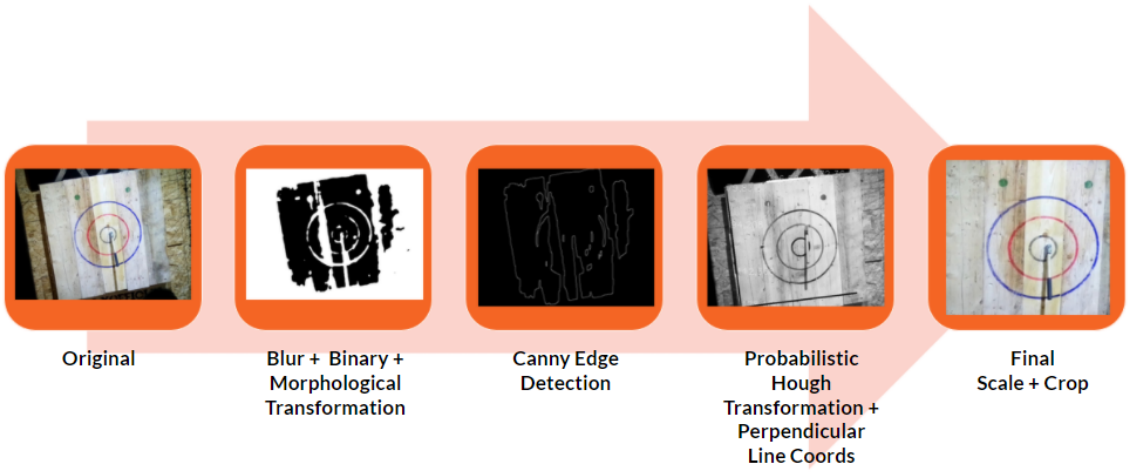

Figure 8. A visualization of the steps taken to transform a "Raw" image capture to a "Normalized" scaled and cropped image.

geometric distortions which are inherited from the positioning of the camera system.

The area of interest is the target board, and we consider the background as noise. In 2021, A. Issa et al. [8] proposed a pre-processing framework to apply morphological procedure on a captured image to build the thickness of the object's limits. In 2020, J. Du et al. [9] described a technique used the edges to determine the coordinates of a target board to apply the perspective transform. We use these frameworks to distinguish the shape of the target board to determine its edges and extract its four corner coordinates to apply a perspective transformation with the steps detailed below.

Morphological transformations are typically performed on binary images. We first process our captured image to be gray-scale and apply a Gaussian blur to reduce the noise in the image to convert the image to be binary and apply the morphological transformations. The Canny Edge Detection algorithm, developed by John F. Canny in 1986 [3], is used to extract the structural information of the board. We then apply Probabilistic Hough Transform [4] to determine the line segments of the edge. However, because of the natural target board variances and the background noise, line segments found are not complete See Figure 9.

In order to address the incomplete line segments, we use a point-slope-formula based on the initially detected segments' slope to elongate the line segment edge. With a complete line segment edge, we can determine the perpendicular coordinates of the board and use the 3 found coordinates to determine the last coordinate and apply the four-point perspective transformation. The automated perspective scale and crop was successful in transforming 94 percent of the 3031 captured images in our dataset. However, because of the physical limitations, we still had to manually scale and crop about 200 images.

Figure 9. A visualization of the parallel lines detected by the Probablistic Hough Transform. Pixel coordinates of the start and endpoints of the linesegments (display in thick whites stroke) are overlayed on the image.

Image Transformations

In both datasets (raw and normalized), we used randomized image transformations to help expand our existing dataset to help train our model to generalize its learning; these transformations included a random probability to do the following:

- Random rotation [-10,+10]°
- Horizontal Flip
- Blur with a random-sized kernel

An important note is that some of our data is positionally encoded (Left and right clutch). In a scenario where training with transformations occurred (namely horizontal flips), we relabelled all data to remove positional information. Before being inputted into the model for training or evaluation, all images are resized and cropped to 224 by 224 pixels.

Model Architecture

We tested the efficacy of various existing convolutional neural network model architectures they consisted of Mixnet-XL [14], Resnet50 [6], and EfficientNet B2 [15]. Each model performed well and we summarize the results in Table 2. When training on a smaller set of data (approximately 2700 images), we found better accuracy against the testing dataset and settled on it for the use of inference.

Performance Improvement and Preventing Gradient Vanishing

In order to improve model performance, we used autocasting. Autocast only wrapped the forward passes of the network. As a result of autocasting, the gradient values with small magnitudes may not be representable in a float 16 representation, so their values will flush to zero. This uderflowing event means the updates for the corresponding parameters will be lost and the gradients vanish. To prevent this, we utilized gradient scaling for backward passes which multiplies the network losses by a scale factor.

Learning Rate

We tried several learning rate schedulers including but not limited to Cosine Annealing, Linear, and Step Scheduling. We found the most optimal results came with a Step scheduler.

Transfer Learning

As well as testing different architectures, we investigated using pre-trained weights from their models. We found that starting with pre-trained values lead to faster convergence and higher accuracy. Conversly, fine-tuning the model did not result in any noticeable improvement when we unfroze the base model.

Performance Evaluation

As shown in Table 2, we enjoy high accuracy values for a variety of different model architectures. The highest accuracy of the "unmodified" dataset was 91.00%; when using the "Normalized" dataset, we reached a slightly higher accuracy of 91.43%. A notable point about our highest accuracy "unmodified" model is that we created this model when there were relatively less data available to us, and it used approximately 2700 data points as opposed to the full dataset.

Architecture	Dataset	Validation	Test
MixNet-XL	**Unmodified 2700**	**93.18%**	**91.00%**
MixNet-XL	Unmodified Full	90.70%	87.91%
ResNet	Unmodified Full	92.25%	90.11%
EfficientNet B2	Unmodified Full	90.25%	89.23%
ResNet50	**Full (Normalized)**	**91.47%**	**91.43%**
MixNet-XL	Full (Normalized)	90.31%	89.45%
EfficientNet B2	Full (Normalized)	89.53%	87.47%

Table 2. A summary of the performance of different model architectures including the dataset that they used. Test Accuracy represents a separate set of data which was never seen by the models and only for evaluation. The two rows bolded represent the best test accuracy dependant on the type of dataset used.

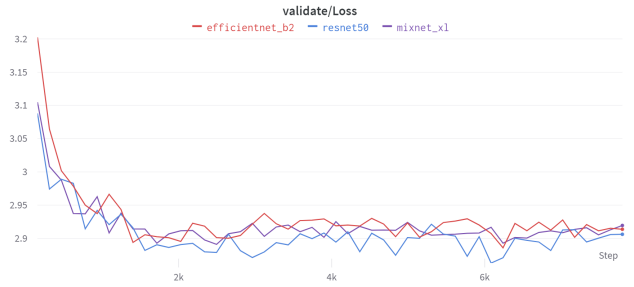

Figure 10. A comparison of the different model architectured validation loss against the "Raw" dataset. No strong indication of overfitting.

Another evaluation which is important for us to gauge the worthiness of our model is the likelihood of overfitting. We can see in the validation loss graph (Figure 10)that there are no strong indications of overfitting.

Data Bias Concerns

As mentioned in the Data Capture Challenges and Considerations Section, we have an imbalanced dataset which might cause our model to overfit and only answer the majority class. This concern is not substantiated in our trained model and is seen through the confusion matrix in Figure 11. Had our model only been predicting the majority class, we would see a vertical bar on the class of 5, but here we see a strong diagonal showing that the model was not only predicting with variety but also correctly predicting the true classes.

Model Selection for Inference

With our highest two models showing similar performance, we settled for using the model which was trained on the "Raw" dataset. Even though the "Normalized" model performs better, the incremental gain comes at the cost of needing to preprocess all the frames at the edge as part of our streaming pipeline. We do not see the relatively small incremental gain as worthwhile to our real-time processing system, where any additional encumberments can make the interface feel unusable. Our belief in why the "Raw" model performs adequately is that the convolutional neural network learns to generalize the axe placements even with the variations in the original dataset and even more so with the transformation described in the Image Transformations Section.

Figure 11. Confusion Matrix of the final model. The model does a fair job at predicting the correct classes and is not mistakingly predicting the majority class for highest accuracy results. The class 10 represents an Empty board, and the class 11 represents a Clutch hit; all other numeric values refer to their relative point values.

Real-time Inference

With a trained model, we are now able to slightly modify our existing capture pipeline (See Figure 2) to make predictions from frames of a live video stream. After every 10th received frame, we feed the image to the model and received a predicted value. This predicted value is then drawn on top of the video frame and served back to the user in the Web Interface.

Skeggøx demonstrates it is possible to do automatic score classification of non-traditional accuracy sports such as axe throwing. Even with the challenges we faced and described in prior sections, we were able to create a reliable, intuitive, and replicable model. In the future, we would like to explore other mechanisms which can help us get an even more accurate system including but not limited to: stereographic or alternating view cameras, time-of-flight sensors for depth perceptions, more robust networking, more readily accessible physical setup, and using robust preprocessing systems.

With a model accuracy above 90% and a robust infrastructure, we firmly believe that our work can be extended to other activities.

ACKNOWLEDGMENTS

This work was supported in part by the UC Berkeley: School of Information. We thank LA-AX and their staff for use of their facilities and the incredible support they provided. Special thanks to Michael Werner for providing calibration materials and Jacob Lewallen for networking equipment.

REFERENCES

[1] M. Archana and M. Kalaisevi Geetha. 2015. Object Detection and Tracking Based on Trajectory in Broadcast Tennis Video. *Procedia Computer Science* 58 (2015), 225–232. DOI:http://dx.doi.org/https://doi.org/10.1016/j.procs.2015.08.060 Second International Symposium on Computer Vision and the Internet (VisionNet'15).

[2] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[3] John Canny. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), 679–698.

[4] Richard O. Duda and Peter E. Hart. 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Commun. ACM* 15, 1 (jan 1972), 11–15. DOI: http://dx.doi.org/10.1145/361237.361242

[5] GStreamer. 2021. GStreamer. (2021). https://gstreamer.freedesktop.org/

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. (2015).

[7] IATF. 2021. Rules & guidelines. (2021). https://iatf.com/rules-guidelines

[8] Abbas Issa, Sundus Dhamad, and Wisam Ali. 2021. Automation of Real-time Target Scoring System Based on Image Processing Technique. *Journal of Mechanical Engineering Research and Developments* 44 (11 2021), 316–323.

[9] Jian Li Jiarong Du, Ru Lai. 2020. Vision-based Automatic Archery Target Reporting. *The 9th International Symposium on Computational Intelligence and Industrial Applications* (10 2020).

[10] NVIDIA. 2021. Nvidia Jetson Nano. (Apr 2021). https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[11] Raspberry Pi. 2021a. Raspberry Pi Documentation. (2021). https://www.raspberrypi.com/documentation/accessories/camera.html

[12] Raspberry Pi. 2021b. Raspberry Pi High Quality Camera. (2021). https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/

[13] Pallets Projects. 2021. Flask. (2021). https://flask.palletsprojects.com/en/2.0.x/

[14] Mingxing Tan and Quoc V. Le. 2019. MixConv: Mixed Depthwise Convolutional Kernels. (2019).

[15] Mingxing Tan and Quoc V. Le. 2020. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. (2020).

[16] TP-Link. 2021. AC750 Wireless Travel Router. (2021). https://www.tp-link.com/us/home-networking/wifi-router/tl-wr902ac/