IC Chipz

sdmay20-40

https://sdmay20-40.sd.ece.iastate.edu/

Advisor/Client - Dr. Henry Duwe

Team

Nicholas Dykhuizen - Integration Developer

Justin Elsbernd - Integration Developer

Joshua Heiser - Embedded Developer

Andrew Kicklighter - Mobile Developer

Paul Kiel - Embedded Developer

Alexander Weakland - Wildcard Developer

Problem Statement

In today's world, there is becoming a shortage of reliable skeet shooting judges. If this problem continues to be ignored, the sport of skeet shooting will eventually be out of options for judges.

The goal of IC Chipz is to solve this issue, do do this the team will implement an automatic scoring systems to fairly judge skeet shooting events using a NVIDIA Jetson board with an E-Con Systems camera, a mobile application, and machine vision.

Key Words and Technologies

1) Darknet: Convolutional Neural Network for training object detection models
2) OpenCV: A technology to integrate into recording devices as well as image manipulation
3) YOLO (You Only Look Once): Object Detection algorithm used to determining a given frame's result
4) Nvidia Jetson: Hardware platform used for field scoring
5) Xamarin Forms: Framework to develop mobile applications for both iOS and Android

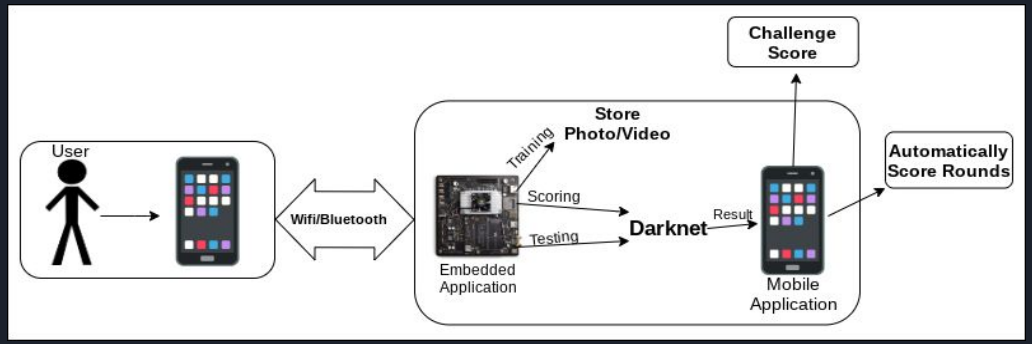
Conceptual Sketch

We envision IC Chipz to have three primary stages:

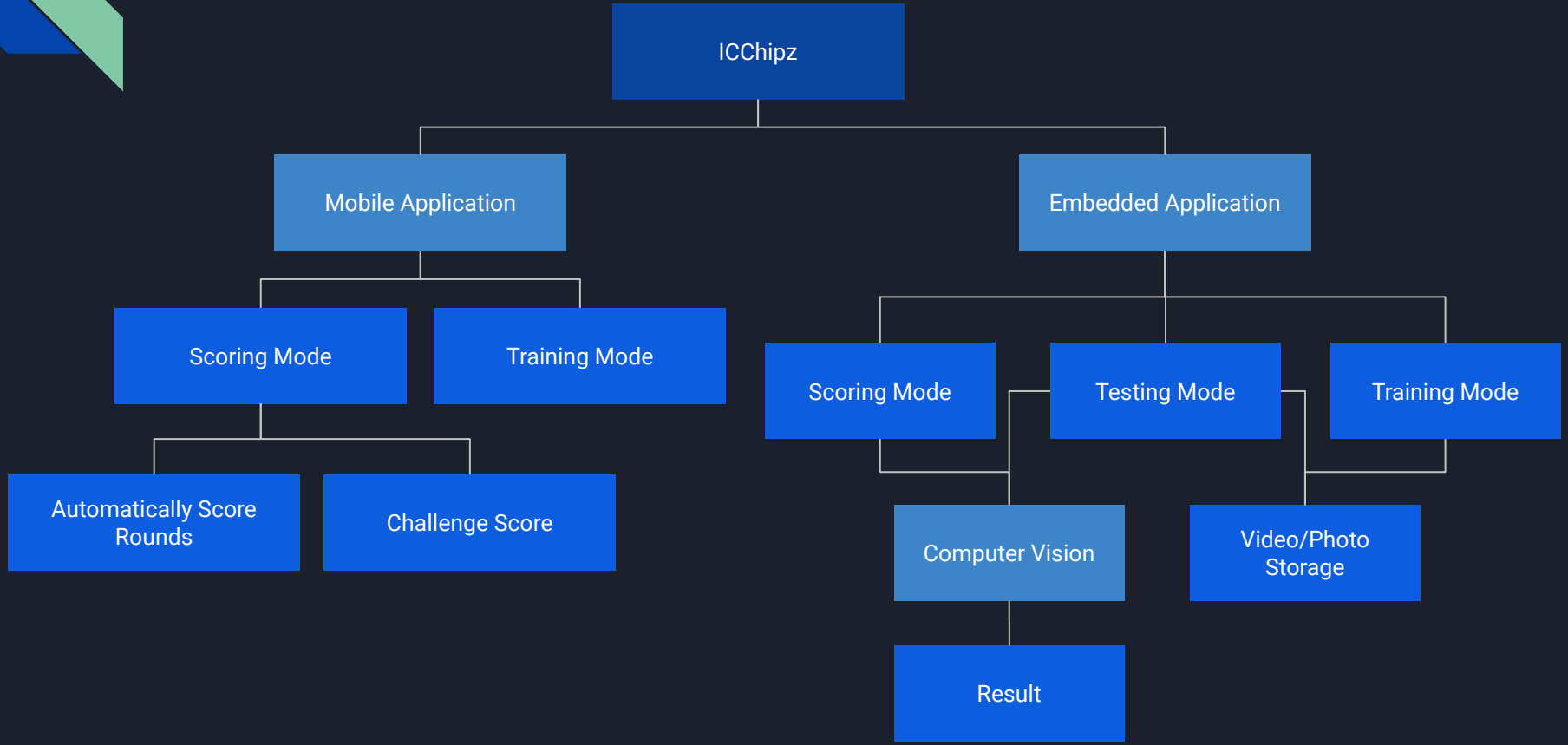
1) Mobile Application
   Handles user interaction with a round; provides front end experience

2) Nvidia Board Integration
   Creates a link between the mobile application and the board's camera and neural network

3) Machine Vision
   Underlying neural network for detecting hits and misses from a video source



Functional Decomposition



Milestones/Requirements Completed

Machine Vision

- Trained machine vision model
- 90+% accuracy of vision model

Nvidia Board Integration

- Data pipeline between video recording device and neural network
- Automate data collection and identification
- Ability to train, test, and score data

Mobile Application

- Connection to board via WiFi or Bluetooth
- Scoring page for in field scoring
- Front end for automated data collection and identification

Potential Risks/Risk Mitigation

- Weather
  - Team gathers data at the skeet shooting range, if it is raining/snowing then the team cannot go to the range to gather data to train the model
  - Mitigation: The team that worked on the project prior has datasets from going to the range, use these to train the model if unable to go to the range
- Safety Hazards
  - Going to a skeet shooting range to gather data by recording live shots is dangerous due to the firearms
  - Mitigation: The team will only be at the shotgun range with direct supervision and will follow all rules set forth at the range.

Online Instruction

While we were still able to meet our requirements, due to online Instruction there were a few parts of are project we were not able to fully explore including:

- Gathering more data from the skeet range for our testing and training models to improve the overall accuracy.
- Testing of the E-Con Systems camera and deciding if we need to implement a second camera or add a new lens to improve video quality and FPS.
- Testing the camera's efficiency in different environmental conditions.

Integration Team
Board Application

- C++ based application
- Integrates with mobile application via wireless technologies
  - Bluetooth
  - WiFi
- Integrates with Darknet and OpenCV
- Application between board's features and mobile application
  - Handles data transfer of images, scoring, and photos
  - Records, stores, and classifies video footage for machine vision model
  - Sends and receives commands from mobile application

Integration Team
Board Application Design and Modes

Design

- Designed with multi threading for footage recording, user input, and network communication
- Singleton design pattern with global resources accessible to all threads.
- Loads machine vision network from config file and locks camera on application boot for increased efficiency

Modes

- **Training** - Gather video footage, recording from user input, and classifying the footage as hit, miss, or no bird
- **Testing** - Gather video footage, and match user marked hit, miss, or no bird, against the machine vision model
- **Scoring** - Automated data scoring by detecting targets in frame sending the results over the network to the mobile application

Embedded Team
Why choose Computer Vision/Neural Networks?

- From the original goals laid out by our client, we needed the following things
  - A system that can find an object
  - A system to keep track of the status of the object
  - A way to to score based on the status of the object
- This is usually done a couple of ways
- We felt that object detection/Neural networks is the easiest way to solve these issues
  - Object Detection allows us to find objects in frames
  - The Neural Network is used to help assist in the status of the object
  - These two working in tandem will allow us to automate this process as our Embedded board can do this work while the camera is recording/operating.

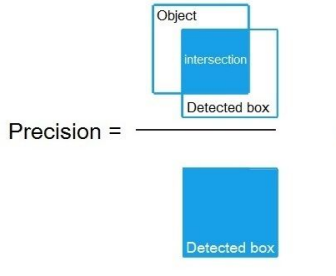
Why Darknet/YOLO?

- We want object detection
- Algorithm that is easily trainable
- Time to train (and run) is much faster than other respected algorithms

Coco Dataset

The COCO dataset is a standard image data set for benchmarking computer vision applications and models for speed and precision

mAP - mean average precision



Performance on the COCO Dataset

Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244
SSD321	COCO trainval	test-dev	45.4	-	16
DSSD321	COCO trainval	test-dev	46.1	-	12
R-FCN	COCO trainval	test-dev	51.9	-	12
SSD513	COCO trainval	test-dev	50.4	-	8
DSSD513	COCO trainval	test-dev	53.3	-	6
FPN FRCN	COCO trainval	test-dev	59.1	-	6
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20

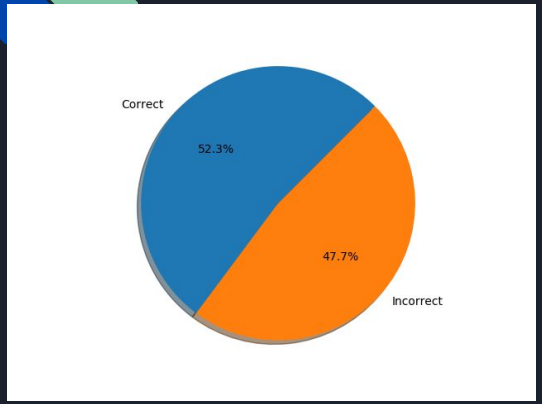
Chart from: https://pjreddie.com/darknet/yolo/
Data in chart provided by multiple studies at Cornell

Model testing

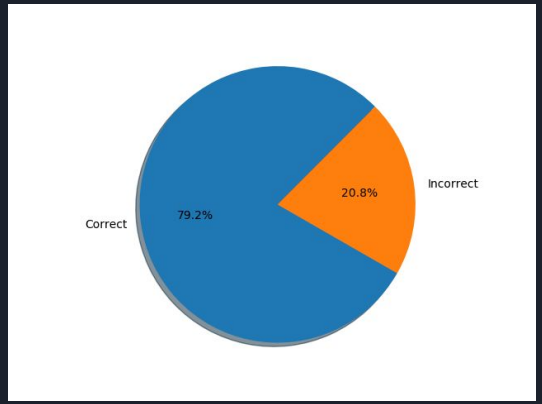
- Mutually exclusive test and train sets.
- Compare known with cv model output.
- Isolate and save filenames of images that are read incorrectly.
- View results in a confusion matrix format.
- Found that nothing less than 7000 training iterations gave accurate enough results.
- 7000-10000 iterations shows small increases but 83-88% model accuracy is viable for our application
- 14000 seemed to overtrain model and it tested poorly on mutually exclusive test set

Iteration amount seen above Graph

1000



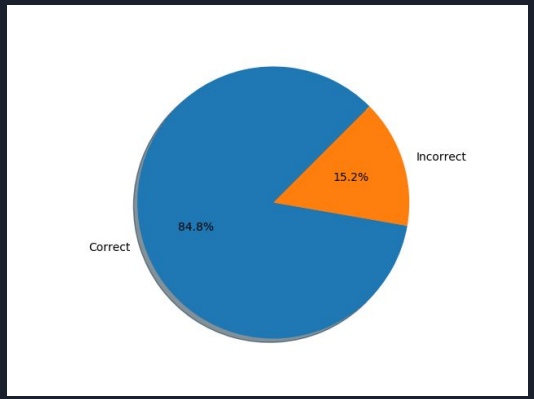
4000



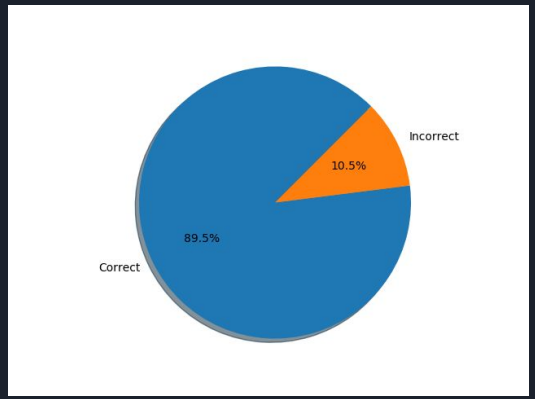
BLUE = Correct Prediction

ORANGE = Incorrect Prediction

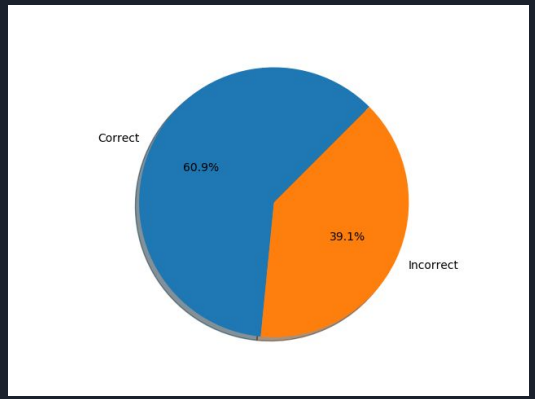
8000



10000



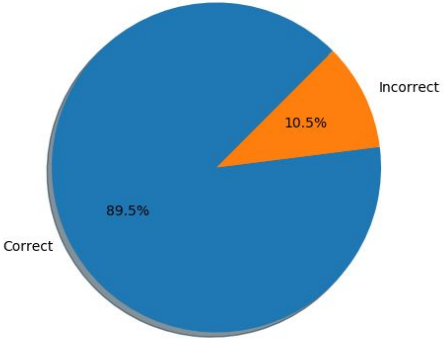
14000 (overtrained)

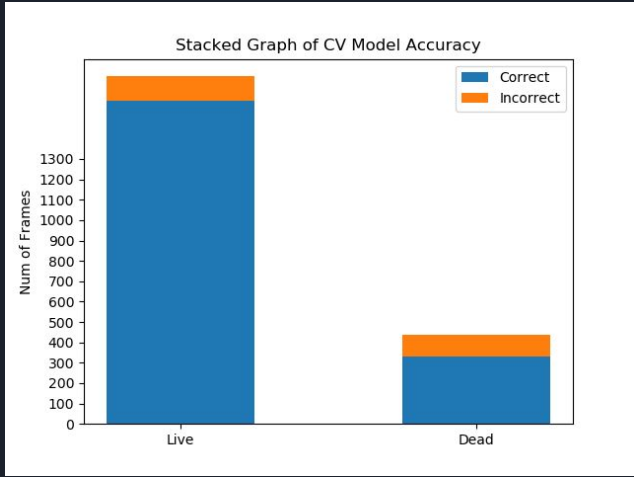


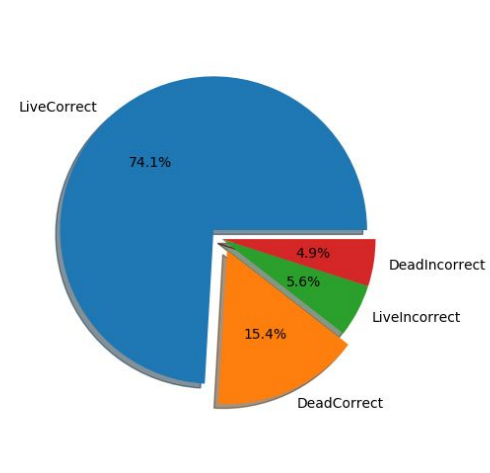
Model Accuracy

(at 10000 Iterations)

```
total number of images pulled is 2142
Darknet guessed correctly for live 93.02461899179367% of the time
Darknet guessed incorrectly for live 6.97538100820633% of the time
Darknet guessed correctly for Dead 75.68807339449542% of the time
Darknet guessed incorrectly for Dead 24.31192660550459% of the time
Overall Accuracy of the Darknet algorithm is 89.49579831932773%
```







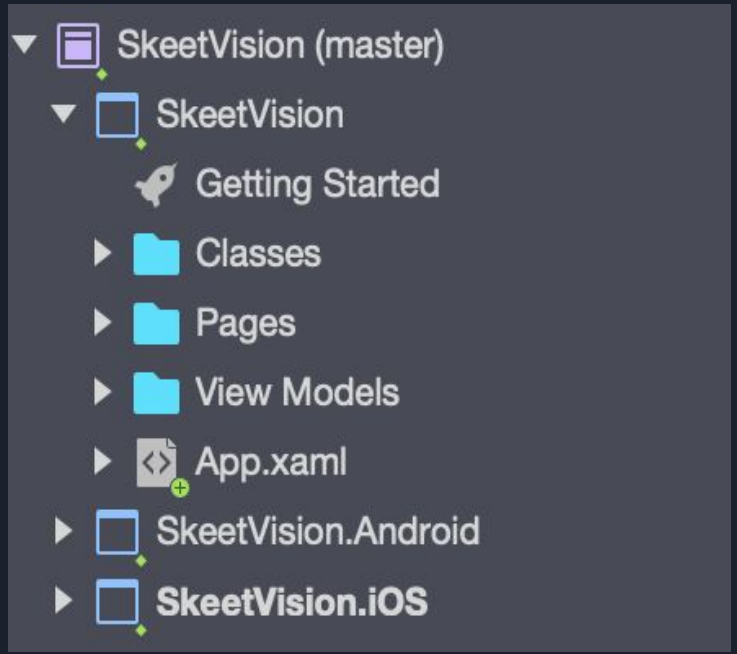
Computer Vision to Score Frames

- Written in C++
- Loads labels and vision data file from config file
- Receives video frames from OpenCV integrated camera
- Get a frame and pass it into Darknet
    - Darknet will return the number of objects detected that meet our labels in a frame
- Go through each object that Darknet returns and check their status
    - If an object is detected is hit for multiple frames, we know that the pigeon has been hit
    - If a object was detected earlier and is now no longer detected, we know that the pigeon has been missed
    - If there was never a pigeon in the frame, then return no pigeon

Mobile Design

- **Platforms**
  - Xamarin Forms through Visual Studios
  - Allows for cross platform development for iOS and Android
- **Design**
  - Minimize platform specific code
  - Split the code into 3 parts: Classes, Pages, and View Models
    - Classes are general code to be used throughout the app (i.e. code to connect to WiFi or define a shooter)
  - Pages deal with the UI design
  - View Models are the backend code for each page and handle things such as scoring rounds

Xamarin

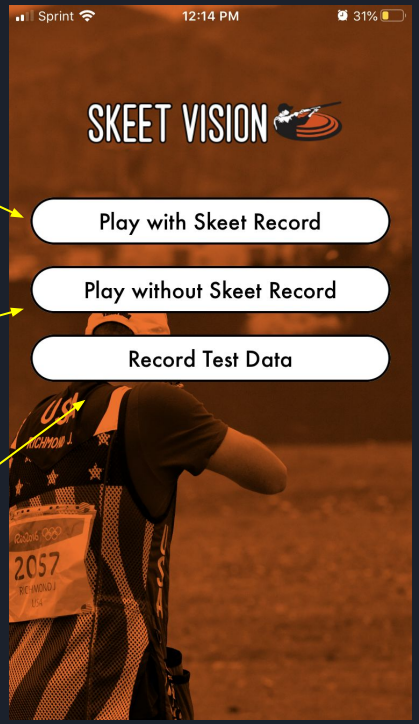


Mobile Design

Used to score a round with the board once connected

Used for scoring a round if you do not have the board

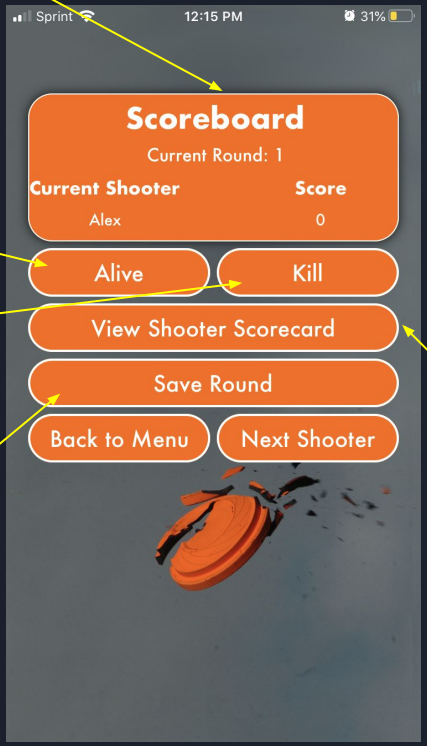
Used for gathering data for the testing set, will not be in the production app



- Mark shot as a hit or a miss
- Used for training the device
  - Production application will automatically mark the skeet as alive or dead

- Allows user to save their round with all its data, this can be revisited at any time

- Displays the current round, the current shooter and their score

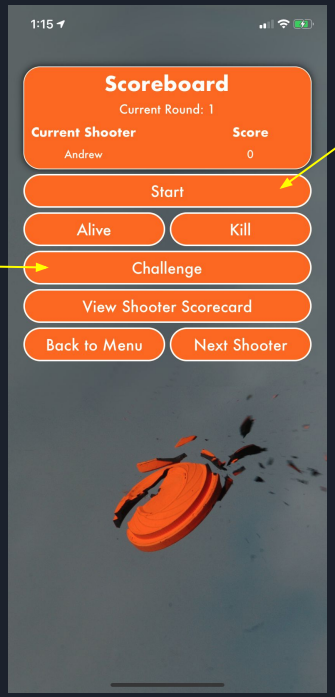


- View the shooter's scorecard
- Can also view every other shooter's scorecard who is currently in the round

Mobile Design

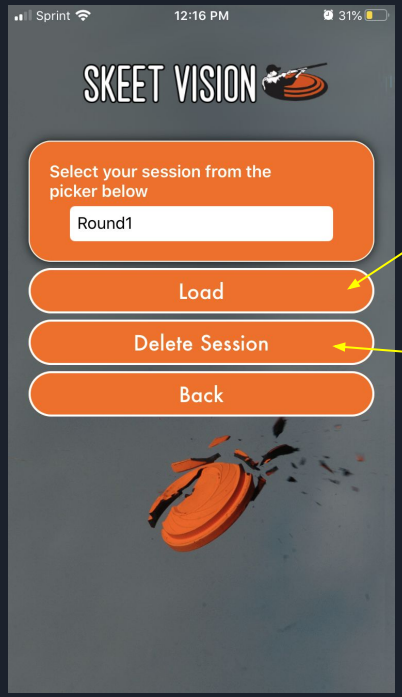
- Once connected to the board the user will have the option to Challenge the ruling of the device
- Will play the video the device took
  - Official reviews the video and marks the shot as alive or dead

UI when connected to board



- Start recording video on the device
- Changes to stop when pressed

Save State



- Load saved session from its name
- Ability to delete a session by its name if it is no longer needed

Testing

- Mobile Team
  - Made sure that all functionalities required for the mobile app works as expected
  - Used Apple's Test Flight so that other team members were able to utilize the application we reported bugs/crashes to the mobile team as they occurred
- Integration Team
  - Made sure that the mobile app and the embedded board can communicate as needed
  - Made sure that file transfers are successful
  - Made sure that file storage is done correctly based on inputs
- Embedded Team
  - Made sure that the scoring algorithm works
  - Created proper test and training sets
  - Added the training sets to Darknet to increase its accuracy
  - Used Python scripts to verify the accuracy meets the requirements set out at the beginning of the semester (90+ percent)

Engineering Standards and Design Practices

- IEEE 1012-2016 - Standard for System, Software, and Hardware Verification and Validation
- IEEE 1220-2005 - Standard for Application and Management of the Systems Engineering Process
- IEEE 829 - Documentation Standards

Conclusion

- **What we have…**
  - A working model that will be able score with 90%+ accuracy during a night-time game.
  - The mobile application and portable board device have a data collection mode allowing them to add additional data points to a data sets
  - A mobile application that communicates back and forth with the device
  - Mobile application has a score review process to allow invalid scores to be challenged and reversed if necessary as well a save state functionally
  - Post round statistics information that is displayed after each round
- **Our plan for the future...**
  - Work will likely continue next year with a new group to continue increasing vision model accuracy, allowing it work under more conditions like daylight and lens flare, adding features, and maintaining the app on both iOS and Android.

Contributions

Andrew Kicklighter

- Mobile Application Development
- Bug Testing
- Integration between application and board

Josh Heiser

- Wrote test scripts for DarkNet model testing
- Integration of DarkNet API in C++
- Object detection

Nick Dykhuizen

- Integration design
- Network communication
- Pipeline Implementation
- OpenCV Integration
- DarkNet API in C++

Paul Kiel

- DarkNet Development/Implementation
- Object Detection
- Model Generation

Justin Elsbernd

- On-Board Integration
- Pipeline Implementation
- Integration of DarkNet in C++

Alex Weakland

- Analyzed Training and Testing data for vision model
- Bug Testing
- Assisted teams with various tasks and bugs (wildcard)

Special Thanks

Thank you to Dr. Henry Duwe for his guidance with the IC Chipz project.

Demonstration

A short demonstration showing how the Embedded Board would pass data into Darknet and the algorithm that scores/places the results in the correct directory on the Embedded System.

Questions?