



PRESENTED BY  **Raytheon**  
Technologies

2021-2022 **FIRST**<sup>®</sup> Tech Challenge

# FTC Machine Learning Toolkit

## Sponsor Thank You

Thank you to our generous sponsors for your continued support of the *FIRST*® Tech Challenge!

***FIRST*® TECH CHALLENGE  
SEASON PRESENTING SPONSOR**



---

***FIRST*® TECH CHALLENGE  
PROGRAM SPONSOR**



---

***FIRST*® TECH CHALLENGE  
KEY SPONSOR**



Revision History		
Revision	Date	Description
1	10/24/2021	Initial Release

## Contents

Contents.....	3
1. Introduction.....	4
Sponsor Special Thanks .....	4
What is FIRST® Tech Challenge? .....	4
2. <i>Gracious Professionalism</i> ® .....	4
3. Machine Learning In a Nutshell .....	4
4. Logging in to the FIRST Tech Challenge Machine Learning tool .....	5
4.1 Preparing to log in to the tool.....	5
4.2 Logging into the ftc-ml tool.....	5
4.3 Changing the active team login session.....	6
4.4 Logging out .....	7
5. Managing the ftc-ml tool workflow.....	8
5.1 General ftc-ml workflow overview .....	8
5.2 Creating videos of objects to be recognized .....	9
5.3 Uploading videos to the ftc-ml tool.....	10
5.4 Adding labels to frames in a video .....	11
5.5 Producing Datasets .....	13
5.6 Training Models.....	14
5.7 Continuing Training on Models .....	16
5.8 Understanding Model Metrics .....	17
5.9 Canceling Training .....	20
5.10 Deleting a Model .....	21
5.11 Downloading Models .....	21
6. Using custom TensorFlow Models in Robot Code (by Uday Vidyadharan) .....	22
6.1 Android Studio.....	22
6.2 OnBot Java (OBJ) .....	27
6.3 Blocks .....	31
7. Optimizing Videos for increased TensorFlow Model Performance .....	34
7.1 Background Information .....	34
7.2 FAQ.....	36
Volunteer Special Thanks.....	37

## 1. Introduction

---

### **Sponsor Special Thanks**

Special thanks to FIRST® Strategic Partner Google for their support in making this tool available!

### **What is FIRST® Tech Challenge?**

FIRST® Tech Challenge is a student-centered program that focuses on giving students a unique and stimulating experience. Each year, teams engage in a new game where they design, build, test, and program autonomous and driver operated robots that must perform a series of tasks. To learn more about FIRST® Tech Challenge and other FIRST® Programs, visit [www.firstinspires.org](http://www.firstinspires.org).

## 2. Gracious Professionalism®

---

FIRST® uses this term to describe our programs' intent.

*Gracious Professionalism®* is a way of doing things that encourages high-quality work, emphasizes the value of others, and respects individuals and the community.

Watch Dr. Woodie Flowers explain *Gracious Professionalism* in this [short video](#).

## 3. Machine Learning In a Nutshell

---

What is Machine Learning? Machine learning is a branch of Artificial Intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way humans learn, gradually improving its accuracy. To borrow a description from TensorFlow's "[Intro to Machine Learning](#)" video series, traditional programming involves programming complex rules into a computer program that are used to analyze input data and output an answer. If the input data is an image of a flower, and if the programming/rules can recognize the flower, then it outputs the answer "flower." Having a traditional program recognize the differences between multiple different kinds of flowers would require significantly more complex programming, especially if the images are allowed to be at various angles and orientations, and not directly centered in the image. Instead, machine learning focuses on providing examples to a machine learning algorithm or "model" – providing data **and** answers – and allowing the model to build its own rules to determine the relationships between the examples provided to it. Just like a human, during each "step" of the training process the model makes a refined guess about the relationships between the known examples and then tests those guesses against examples not yet seen. By training a model over successive steps, the model attempts to improve its accuracy in correctly identifying previously unseen variations of the data. In this way, training a model to correctly recognize multiple types of data requires no more source code than recognizing a single type, it only requires creating more examples for the model to learn.

In FIRST Tech Challenge, the machine learning platform used is [TensorFlow](#). TensorFlow is an open source platform for machine learning with a comprehensive, flexible ecosystem of tools, libraries, and community resources to enable developers to create tools such as the FIRST Tech Challenge Machine Learning tool. TensorFlow has been utilized in FIRST Tech Challenge for a number of years, allowing teams to recognize individual game pieces and clusters of game pieces via pre-built models developed by FIRST Tech Challenge engineers. Now FIRST Tech Challenge is empowering teams to build their own custom models!

## 4. Logging in to the FIRST Tech Challenge Machine Learning tool

### 4.1 Preparing to log in to the tool

The FIRST Tech Challenge Machine Learning (ftc-ml) tool uses a Single Sign On (SSO) login through an individual's [FIRST Dashboard account](#) managed by the [ftc-scoring](#) platform, allowing the ftc-ml tool use a FIRST Dashboard login session for authentication through the ftc-scoring platform. One consequence of using FIRST Dashboard SSO is that all users of the ftc-ml tool MUST have a FIRST Dashboard account. The benefits of using the FIRST Dashboard SSO are that team affiliation and permission levels are automatically shared with the ftc-ml tool, allowing an individual's FIRST Dashboard account to be used for identity purposes and allows the team's roster to be the definitive source for team membership information.

Before logging into the ftc-ml tool, your browser (Chrome, Firefox, etc.) should be updated to the most recent version provided by the author of the browser. For example, older chromebooks that are limited and cannot update to the most recent version of the Chrome browser may not properly function within the ftc-ml tool. The only browser that has been fully tested with the ftc-ml tool is the Chrome browser, currently at version 94.0.4606.81 as of the writing of this document.

### 4.2 Logging into the ftc-ml tool

To log into the ftc-ml tool, go to the following URL: <https://ftc-ml.firstinspires.org>. If there is an active login session already being managed by the ftc-scoring platform, this URL will either take you directly to the team selection page (if you are present on the roster of multiple teams) or the main workflow page of the ftc-ml tool. Otherwise, this URL will temporarily redirect to the FIRST Dashboard login page as seen in Figure 1.

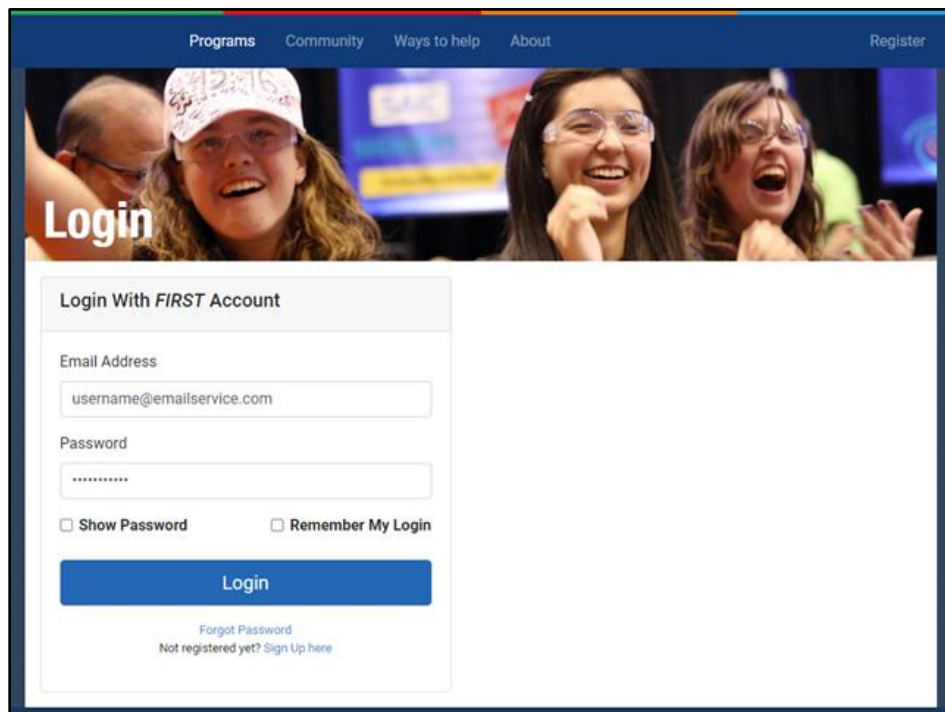


Figure 1: FIRST Dashboard Login Page

Enter login credentials for your FIRST Dashboard account here. If a password manager is being used, the password manager should recognize the domain being used and auto-fill the username and password for you.

Once login details are complete, click the Login button. Once the login credentials are accepted, you may be taken to one or more of these three pages:

1. If your *FIRST* Dashboard account is present on the roster for more than one team, you will be taken to the Team Selection page. On this page, clicking the “Select...” button under the “Team Number” header will provide a drop-down list of all team numbers for which you appear on the roster. Select the team number of the ftc-ml session you wish to enter, and click “Submit.”
2. If your *FIRST* Dashboard account is present on the roster of only one team, you will be taken to the main workflow page for the ftc-ml tool for your team. If your *FIRST* Dashboard account is present on the roster of more than one team, you will be taken to the main workflow page of the team selected on the Team Selection page.
3. If your *FIRST* Dashboard account is not associated with a team, or the associated/selected team does not have access to the ftc-ml tool, you will be taken to an error page as seen in Figure 2.

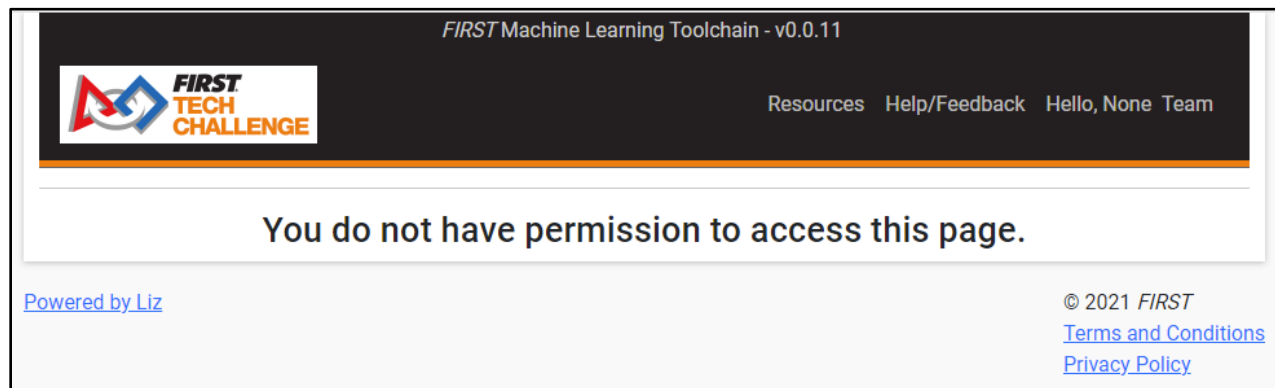


Figure 2: ftc-ml login permission denied error page

### 4.3 Changing the active team login session

If your *FIRST* Dashboard Account appears on the roster of multiple teams, and you're currently logged in to the ftc-ml tool and wish to change the active team session, **follow these steps exactly** to change teams:

1. Click on the “Hello, <NAME> Team <NUMBER>” text in the main header.
  - This will redirect you to the ftc-scoring accounts page. The simple act of clicking on “Hello...” and being redirected to the ftc-scoring login page will invalidate the Team Selection setting submitted when you first logged on.
2. **DO NOT** CLICK ON THE BROWSER'S BACK BUTTON once at the ftc-scoring accounts page, this will invalidate your entire session.
  - If you click on the browser's BACK button while on the ftc-scoring accounts page, you invalidate your entire SSO login session, and you will have to click the “Log Off” button on the [ftc-scoring accounts page](#) in order to completely clear your SSO session and try again.
3. In the browser's URL, go to <https://ftc-ml.firstinspires.org> to go back to the ftc-ml site.
4. Select the team from the “Select...” drop-down that you wish to enter the ftc-ml session for.
5. Click the “Submit” button.

#### 4.4 Logging out

When finished with an active ftc-ml session, it is advisable to log out of ftc-ml in order to ensure that the login session is closed and your *FIRST* Dashboard account is secure. To do this, follow these steps:

1. Click on the “Hello, <NAME> Team <NUMBER>” text in the main header.
  - This will redirect you to the ftc-scoring accounts page.
2. Click on the red “Logout” button on the ftc-scoring accounts page.
  - Pressing the “Logout” button closes the active authentication session with the *FIRST* Dashboard, cleans up session cookies, and prevents others from accessing your account.
3. Close the browser window.
  - This last step isn’t technically necessary, but it’s good practice.

## 5. Managing the ftc-ml tool workflow

### 5.1 General ftc-ml workflow overview

The ftc-ml tool is designed in such a way as to make TensorFlow model building simple and easy. It does not provide the myriad of user-accessible parameters to tweak that TensorFlow offers, so it's not meant as a general-purpose TensorFlow model building tool. However, teams will find that the parameters are sufficient for the vast majority of TensorFlow Object Detection (TFOD) use-cases used in *FIRST* Tech Challenge.

The process of building/training a TensorFlow model using the ftc-ml tool is summarized as follows:

1. Teams create short videos of the objects that they would like the model to be trained to recognize.
2. Videos are uploaded into the ftc-ml tool, and individual objects to be recognized in each video frame are labeled by the users.
3. Datasets composed of one or more labeled videos are created. Unlabeled videos, if used in a dataset, must be combined with labeled videos.
4. One or more datasets can be combined to create a model. The model is trained using Google TensorFlow cloud training services using the selected datasets as training resources.
5. The model is downloaded from the ftc-ml tool, and installed either onto the Robot Controller (for OnBotJava or Blocks) or within the Android Studio assets for use on the robot.
6. Robot code is modified to use the new model file and the labels created during the model creation process.

The ftc-ml main workflow page is designed to facilitate the model building/training process, and is demonstrated in Figure 3. The main body of the workflow page is designed to lead the user through a chronologic workflow of building/training TensorFlow models. This page is designed to be rendered minimally full-screen on a 1280x720 resolution monitor.

Date Uploaded	Description	Video Filename	File Size	Dimensions	Duration	Frames per Second	Number of Frames			
							In Video	Extracted	Labeled	Excluded
<input type="checkbox"/> 10/13/2021, 11:13:36 AM	<a href="#">FlashGordon5secvideo</a>	FlashGordon5sec.mp4	10,473,057	1920 x 1080	0:06	30	176	176	176	0
<input type="checkbox"/> 10/13/2021, 6:05:43 PM	<a href="#">FlashGordon5secVideoFlash</a>	FlashGordon5sec.mp4	10,473,057	1920 x 1080	0:06	30	176	176	176	0
<input type="checkbox"/> 10/18/2021, 5:21:54 PM	<a href="#">GhostBusterDuck</a>	GhostBusterDuck_10tps.mp4	1,350,860	1280 x 720	0:06	10	64	64	0	0

Figure 3: Example ftc-ml Main Workflow Page, showing sample **Videos** menu tab content



There are 3 primary areas of the main workflow page:

1. **Title Header** – The header has several important elements in it.
  - a. **Title and Version Information** – The title of the product, “FIRST Machine Learning Toolchain”, is shown alongside a version number indicator. Each time a new version of the software is deployed the version indicator will update.
  - b. **FIRST Tech Challenge Logo** – On the left of the header is a FIRST Tech Challenge logo. Clicking on the FIRST Tech Challenge Logo will always bring you back to the main workflow page, regardless of what menu or screen you are currently in, and will always restore the workflow Tab to the last selected Tab. There is no need to “save” any work or progress when using the ftc-ml tool, progress and work is saved automatically.
  - c. **Resources** – The Resources link will navigate to a page that contains resources such as the most recent copy of this ftc-ml manual and links to important or supplementary information.
  - d. **Help/Feedback** – The Help/Feedback link will navigate to the ftc-ml support forums. The support forums also use SSO login authentication, so to log in just click the “Login” button and if prompted just click the “Sign in with FIRST” button.
  - e. **Hello <NAME> Team <NUMBER>** - this link will take you to the ftc-scoring accounts page where you can log off when desired. This link also serves as the mechanism for invalidating a team selection (if your account is rostered on multiple FTC teams) so that a different team can be selected. See section 4.3 and section 4.4 for more information.
2. **Workflow Tabs** – The three main workflow tabs are **Videos**, **Datasets**, and **Models**. These workflow tabs are mostly chronologic from left to right through the TensorFlow model training process. Clicking on each tab will show the tab’s contents in the Tab Contents section of the page.
3. **Tab Contents** – Shows the specific actions and data for the currently selected workflow tab.
  - a. **Videos** – The **Videos** menu tab contains/displays action buttons for Videos. This includes Upload Videos, Produce Datasets (from selected Videos), and Delete Videos. A listing of all of the uploaded videos and a summary of the video contents is provided. Each video’s description, once processed, provides a link to the video labeling page for that video.
  - b. **Datasets** – the **Datasets** menu tab contains/displays action buttons for Datasets. This includes Download Datasets, Start Training models (with selected Datasets), and to Delete Datasets. A summary of each Dataset’s contents are displayed for each Dataset.
  - c. **Models** – the **Models** menu tab contains/displays action buttons for Models. This includes More Training (to continue training on an existing model), Download Model, Stop Training, and Delete Model. A summary of each Model’s training metrics is displayed for each Model. Each model’s description, once completed, shows in-depth details on the model including training performance metrics and a visual comparison of test data.

This section is meant to provide a basic explanation of the model creation process. For information regarding best practices for creating models, see Section 7 Optimizing Videos for increased TensorFlow Model Performance.

## 5.2 Creating videos of objects to be recognized

The ftc-ml tool uses videos instead of individual images because videos are an efficient package for managing potentially hundreds of images of the same object at slightly different angles/orientations and distances from the camera (poses). While video capture can be performed with any camera, it’s recommended that videos have exactly the same resolution as the camera being used on the robot. Ideally video capture should be done

with the exact camera being used on the robot at the estimated height from the surface of the floor that the camera will be at. By using the exact camera on the robot, specific artifacts of the camera used – such as lens distortion and other optical effects – can be reflected in the training images which result in a much better overall object detection rate. Programs such as the Windows 10 Camera Application can be used to capture video from a webcam while it's mounted on a robot and plugged into a laptop. It's recommended to use the lowest frames per second (fps) setting possible, only because with a higher framerate the likelihood of getting multiple frames of the exact same image are incredibly high, and that's just wasted frames that you have to label (or manually discard) and there's no extra benefit in model training with duplicate frames (it takes longer to train your model). There are multiple web-based tools that will allow you to change the frame rate by removing frames from the video free online. For tips and best practices for creating the best poses, see Section 7 Optimizing Videos for increased TensorFlow Model Performance.

### 5.3 Uploading videos to the ftc-ml tool

Once a video is ready for upload, select the Videos tab on the main workflow page. Click the Upload Video button, which will create a pop-up with the title, "Upload Video File." On this pop-up page, click the "Choose File" button to browse the local computer for the video file you wish to upload. Enter a description for the video in the box under the label "Description." Make the description meaningful, but short. Once completed, click the "Upload" button. If you wish to cancel the action and close the pop-up, click the "X" or the "Close" button.

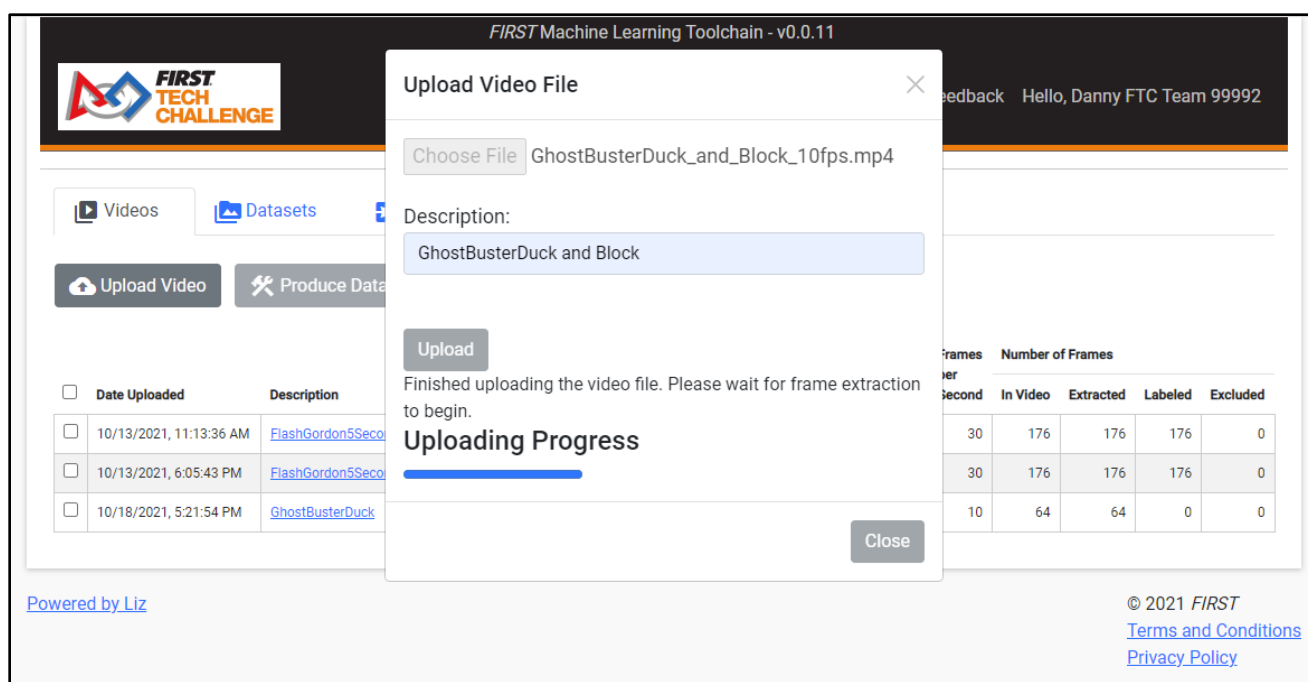


Figure 4: Uploading a video and preparing for frame extraction

Once the "Upload" button is clicked, the ftc-ml tool will begin the process of uploading the video. A progress bar will show the progress of the video upload. Once video upload process is complete, the ftc-ml tool will add the video to the Tab Contents area and prepare to extract the individual frames in the video for processing. Clicking the "Close" button, the "X" button, or clicking anywhere outside the pop-up will close the pop-up window, but the adding and extraction process is still being carried out by the server in the background. It may

take several seconds for the new video to show up in the Tab Contents area. Once the new video shows up in the list, it may take several seconds for the extraction process to begin, depending on server resources. As frames are extracted, the “Extracted” column will begin to count up. Once the “Extracted” column matches the “In Video” column, the description of the video will change to a link. Clicking on the link will navigate to the Video Labeling tool where objects in video frames may be labeled.

#### 5.4 Adding labels to frames in a video

The Video Labeling tool is the primary method for providing input to the Machine Learning process. Also known as Supervised Learning, a user selects regions of each video frame that contains objects that a model will be trained to recognize and provides categorization data in the form of a label. If done completely manually, this process can be time consuming and error prone – fortunately there are tools to help us do this. Because the input package is a video, each frame is more than likely sequential to one another. As the objects move around in the frame (such as a ball rolling by), or the camera pans around the objects (to get multiple pose angles at varying distances from the object) the objects are going to move between frames at relatively predictable increments. Tracking algorithms, such as the OpenCV Object Tracking API, can help track the movement of labeled objects from one frame to another with the help (from time to time) from a human supervising the process.

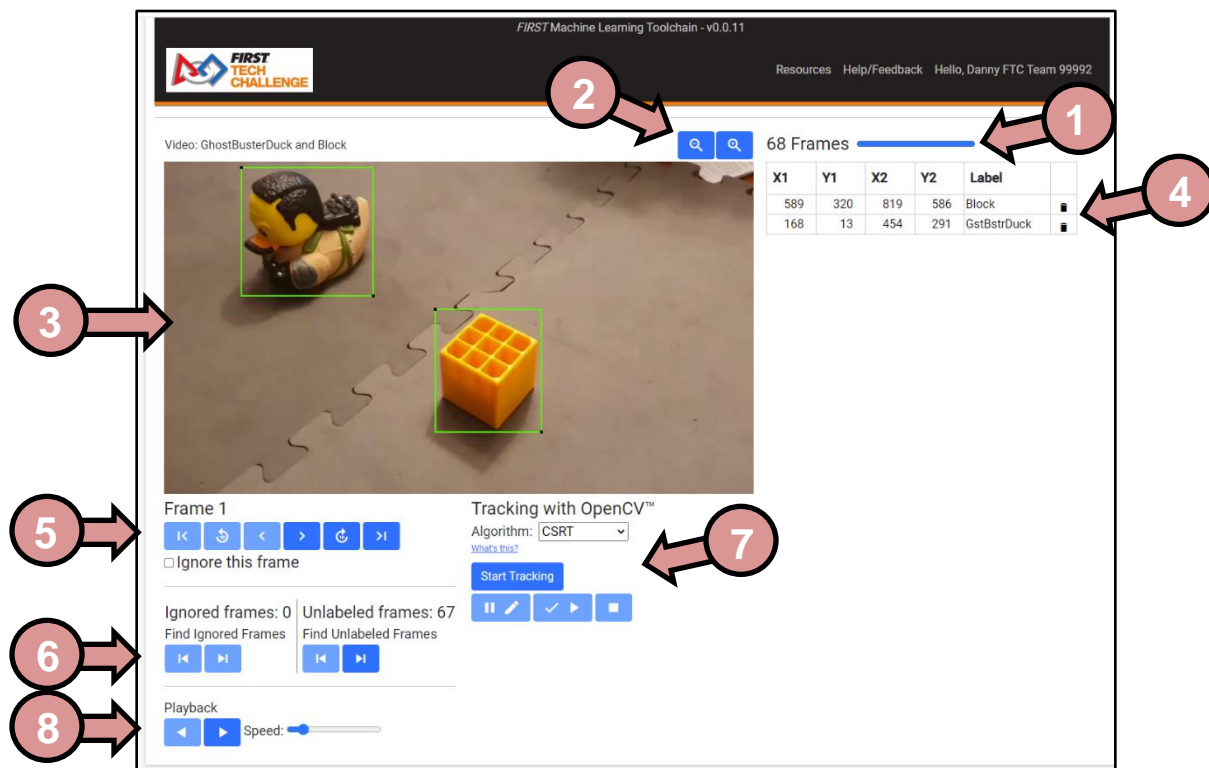


Figure 5: Video Labeling Tool main window.

Figure 5 shows a sample of the Video Labeling Tool main window. The tool is composed of multiple segments, labeled as follows:

1. **Loading Progress Bar** - In the upper-right-hand area of the window, a frame loading progress bar will show the frame load progress; for large videos this might take a while. While the image frames are loading, loaded frames may begin to be examined and labeled in the main image window.

*Gracious Professionalism® - "Doing your best work while treating others with respect and kindness - It's what makes FIRST, first."*

2. **Zoom Tools** - If necessary or desired, these zoom buttons may be used to increase and decrease the size of elements in the Video Labeling tool window.
3. **Main Image Window** – The main image window is for viewing the current frame of the video, and where the labeling for the video happens. To create a bounding box left-click on the location for one corner of the box, drag the mouse to the opposite diagonal corner for the box, and then release the mouse button. Once a bounding box is shown, a label can be added to the Region Labels area or the bounding box can be deleted using the Trash Can icon in the Region Labels area.
4. **Region Labels** – When dragging a bounding box within the Main Image Window, the coordinates of the bounding box are stored here. Each bounding box needs a label. Labels must be exact – just like in a password, capitalization matters! Keep labels short and to the point.
5. **Frame Navigation Buttons** – To navigate deliberately between frames, and show which frame is currently being viewed, the frame navigation area can help out. The current Frame number is shown above the navigation toolbar. To ignore the current frame (completely eliminate the frame from being used in training) the “Ignore this frame” checkbox below the frame navigation toolbar must be checked for each frame to be ignored.
6. **Special Frame Navigation** – This area is used to view/verify ignored frames and to find unlabeled frames in the video. All instances of objects in frames should be labeled, but unlabeled frames (also known as “negative frames”) can be useful if your objects are typically hiding something that you would like for your model to ignore if the objects are not present.
7. **OpenCV Object Tracking tools** – Use this to begin, pause, or stop object tracking using the selected object tracking algorithm. Click the “Start Tracking” button once the first frame is fully labeled, and monitor each frame to ensure the bounding boxes are correct
8. **Playback Menu** - The Playback menu can play the video (with or without labels) at varying speeds. The Left and Right buttons indicate direction, and the Speed slider controls the speed of the playback. Click a direction once to begin playing the video in that direction. This is useful when reviewing bounding box selections between frames.

When a video is first loaded into the Video Labeling tool, it may take several seconds for the image data to be loaded into the tool. Bounding Boxes should be drawn around objects in the Main Image Window and each bounding box needs to be labeled. If the bounding box needs to be modified, click and drag the corners with a black “dot” in them to adjust the bounding box. It’s okay if bounding boxes overlap slightly in training data. If multiple “blocks” are in the image, then each should get the same “block” label. If you want the model to classify an object within a bounding box as a “duck”, then add the “duck” label, and so on. You should not have more than 10 labeled objects within a single frame (so keep that in mind when creating videos).

Once the first frame has been fully labeled, click the “Start Tracking” button on the OpenCV Object Tracking tools. It may take several seconds for the tracking process to begin. Once started, OpenCV will progress frame-by-frame, attempting to track the bounded labeled object as it moves for you. If you need to pause the OpenCV tracking and correct a bounding box that becomes too large, too small, or loses the object, do so. You may resume tracking at any time.

To review the bounding boxes throughout the video, use the Playback Menu to show each frame in sequence in the desired direction.

Once the labeling process has completed, click on the FIRST Tech Challenge logo to return back to the ftc-ml main workflow page. Note that there is no “save” button, actions are saved each time a browser action occurs, and there is no way to “undo” or “redo” actions.

## 5.5 Producing Datasets

Video frames, bounding boxes, and labels are the core inputs to the TensorFlow model training platform. In order to package this data together for TensorFlow, these inputs are converted into Datasets. Datasets are then submitted to the TensorFlow API to create models.

To create a dataset, one or more videos should be selected (checking the box to the left of each video to be combined into a single dataset) and the “Produce Dataset” action button pressed. This will open a pop-up dialog to select the number of frames for training and evaluation. The standard is to take 80% of the frames for training the model, and saving 20% for validation/evaluation/testing. Frames are randomized and separated into the two pools (Training vs Evaluation) based on this percentage. It’s not recommended to change this. Enter a descriptive name in the “Description” field, as this will be the description for the dataset. Keep it short and to the point. When ready, press “Produce Dataset” – the ftc-ml tool will extract the frame, label, and bounding box information and build the dataset. Don’t worry if you close your window or the pop-up goes away before it’s done, when the dataset is completed it will show up in your “Datasets” Tab Content area.

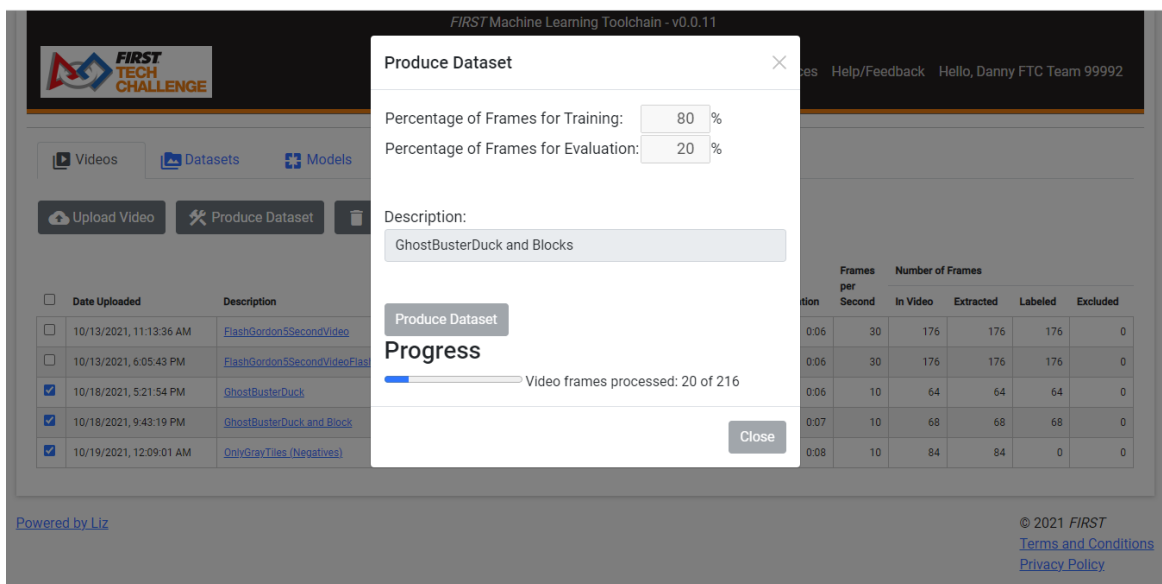


Figure 6: Creating a Dataset with the "Produce Dataset" video action

The most important thing to consider when creating a dataset is the final list of labels. There are several rules to datasets that must be adhered to:

- Datasets must contain AT LEAST one label. In other words, a dataset cannot contain only negative frames (frames that are unlabeled, because no actual objects being detected are present).
- Datasets should be considered “whole” by themselves. While it’s possible to create datasets for individual labels, datasets cannot be “combined” to train models unless they contain exactly the same labels. For example, a dataset containing only the label “Bird” cannot later be combined with a dataset containing both labels “Bird” and “Bee” to form a model. However, a single dataset may be created out

of multiple labeled videos that contain only “Bird”, multiple videos that contain both “Bird” and “Bee”, and videos that only contain negative frames all with the Video “Produce Dataset” action.

When creation of datasets is complete, check the dataset in the “Dataset” tab. Look at the labels used to create the dataset and make sure they’re spelled correctly. If one of the videos had a misspelling, it might be necessary to find and correct the video and create the data set again. See Figure 7 for an example of a dataset made from one or more videos with a misspelled label.

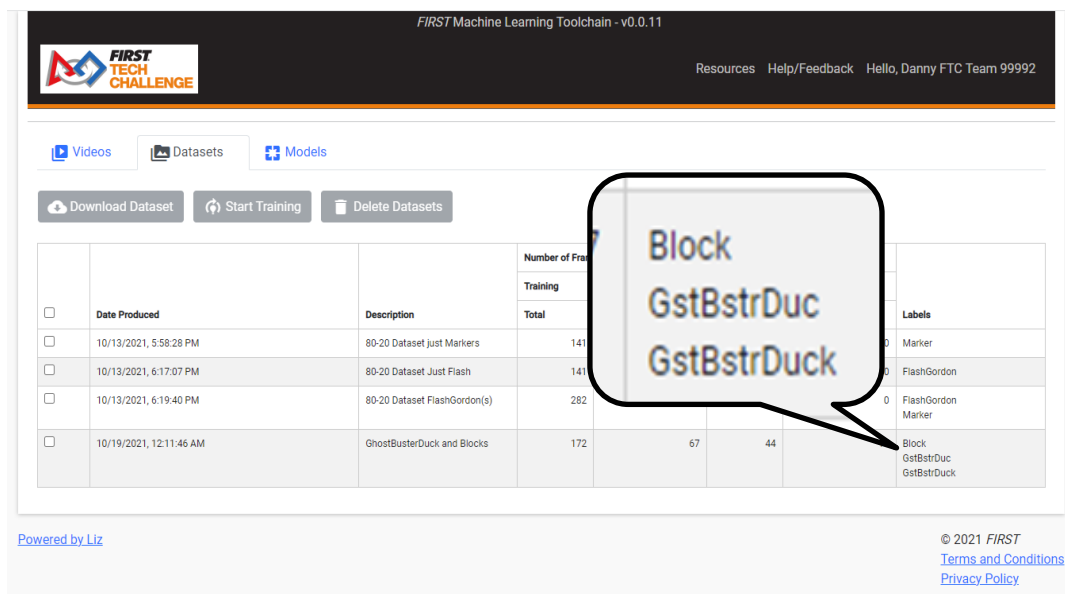


Figure 7: Whoops! Dataset made with videos containing misspelled labels

## 5.6 Training Models

Once a Dataset is created, you’re almost ready to start training your model! From the Dataset tab, one or more datasets may be selected to use as training input for a TensorFlow model. Remember, if selecting multiple datasets the datasets must be 100% label identical in order to be combined into a model, or else the “Start Training” button will not be enabled. No more, no less!

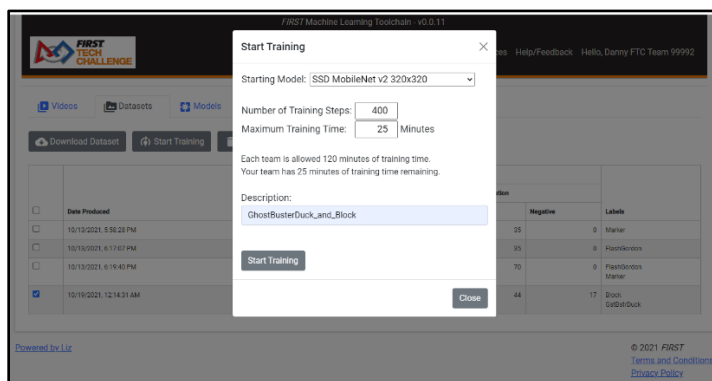


Figure 8: Configuring a Model Training Session



Once you've selected the dataset(s) you wish to use to train a model, clicking on the "Start Training" button brings up a pop-up window as seen in Figure 8. Here you're able to tweak several options:

- **Starting Model** – The typical starting model size is the SSD MobileNet v2 320x320, and this is the recommended model type for FTC. Please don't waste your allocated time specifically experimenting with model types.
- **Number of Training Steps** – Training steps are done in batches of 100, and checkpoints are only saved after every 100 steps, so keep this number as a multiple of 100. You have no idea how many training steps it will take for your model training to converge, and you have no idea how long that will take, and there's no way to predict it except "try it and see how it goes." It's completely dependent upon the number of training frames in your data, the model's training accuracy with that data, and the myriad of parameters you can't tweak.
- **Maximum Training Time** – If you specify 500 steps the model will continue to train until 500 steps have been completed, or until the maximum training time is reached, whichever comes first. If your model trains for 499 steps, and is forced to quit because it reached its maximum training time, the extra 99 steps will be wasted training because only the last model checkpoint is used and checkpoints are only saved every 100 steps. Unfortunately we cannot track how many ACTUAL steps the model trains for, we only get the last checkpoint. Therefore, set your number of training steps low and your maximum training time high to ensure you don't lose training time. Try for 300 steps for 30 minutes, and see if you get all those steps in that amount of time. If the training metrics are not satisfactory, and the total training time was way less than 30 minutes, you can always continue training on the model to see if more time will get you better results (AND you know how long it took, generally). If you allocate 60 minutes for a training session, and it only takes 20 minutes to complete training, you get the remaining 40 minutes back once the training session has completed.
- **Description** – this will be used for the description of your Model. Keep it short and succinct.

Click the "Start Training" button and your dataset is shipped off to the Google TensorFlow platform for training!

**KNOWN BUG:** Sometimes once you press the "Start Training" button the pop-up will eventually go away but the page is still grayed and disabled. If this happens, press the browser's Refresh button to reload the page.

To monitor model training, a user may monitor the status on the Models tab or they can click on the description for the model. The main status indicators are "Job State", "Steps Completed", and "Training Time." Steps Completed will update each time a model checkpoint is reached, and Training Time will update while the Job is in the RUNNING state. A full list of Job States is as follows:

Table 1: Job State possible values

Name	Description
SUCCEEDED	The model has been trained successfully. Check metrics for performance.
FAILED	The model training has failed.
CANCELLED	The user cancelled the job prior to any checkpoints being created.
STATE_UNSPECIFIED	This means that the model is in an unpredicted state. Contact Support.
QUEUED	The job has been queued but has not yet started, is waiting for resources.
PREPARING	The job is preparing to run.
RUNNING	The job is running.

STOP REQUESTED	The user pressed the stop button, but the job hasn't been CANCELLED yet.
STOPPING	The job is in the process of being stopped.
STOPPED	The user cancelled the job after checkpoints were created, can train more.
TRY_AGAIN_LATER	The job cannot be queued immediately. Try to queue again at another time.

### 5.7 Continuing Training on Models

Once a model has been created, and its training and evaluation metrics have been analyzed, it's possible to use that model as a basis for continued training. You must continue to use the same dataset(s) that the model was originally trained with, but it's possible to add datasets if they are completely label identical. If this is the case, additional checkboxes will appear in the "More Training" pop-up to allow you to add datasets for training. There are benefits to NOT adding more datasets – you now have a good estimation of how long your model takes to perform minimal training, and perhaps you can accurately determine training efficiency.

To continue training a model, select the Models tab, select the model you wish to continue training for, and click the "More Training" action button. A pop-up will allow you to specify the number of Training Steps to continue with, the Maximum Training Time, additional datasets if any are compatible, and a new Description for the new model.

Figure 9: Example of continuing training and adding additional datasets

Note: Models share a parent/child relationship, much like Datasets and Models. You cannot delete a dataset that a model used (without deleting the model first) just in case the model wants to continue training, and you cannot delete a parent model without deleting its children first.



## 5.8 Understanding Model Metrics

Models are essentially prediction engines, or weighted algorithms that are designed to predict a future value given a set of input values. When a model is being trained, data from the Training pool is being used to train the model along with a set of weights which can be tuned to help the model with its predictions. For each step in training, data from the Testing pool is used to measure the cumulative model's ability to make predictions. Loss functions are used to determine how far the predicted values of the model deviate from the actual values in the Testing data; this deviation is known as "Loss". Optimization functions use the loss to help adjust model weights between each training step to minimize that loss, so that each step the model prediction is closer to the actual data. This is, in effect, what training is all about.

Each training checkpoint (100 training steps), model metrics are saved – among these metrics include the loss values for various properties of the model. Metrics can be analyzed to get a general sense for how model training is going, and whether or not the model is reaching convergence (reaching the point where additional training yields little to no benefits). Several of these metrics are described below:

Training Metrics (metrics taken as the model is being trained)

- **learning\_rate** – The learning\_rate refers to the average update rate at which the model's weights are changing in order to fit the data. Really small values means the model will take a long time to adjust the weights to fit the prediction to the data, and really large values means the model might overshoot as it's trying to adjust the weights. Since ftc-ml doesn't provide user tweakable parameters, this metric is purely informational.
- **Loss/classification\_loss** – This is the loss for the classification of detected objects into various classes (Labels), such as Block, Ball, Duck, etc. During training, this graph should trend downward as the classification improves. Values closer to zero are better.
- **Loss/localization\_loss** – This is the loss for the bounding box regressor, which is the function for determining the bounding box for detected objects. This graph should trend downward as the prediction for the bounding box moves closer to the labeled bounding boxes. Values closer to zero are better.
- **Loss/regularization\_loss** – This is the loss for a larger set of "global" optimization metrics that help drive the model in desired directions. Since parameter tweaks aren't available to users in ftc-ml, this metric is generally meaningless for analyzing model behavior during training.
- **Loss/total\_loss** – This is an overall summary of the loss metrics for the model as a whole. Values closer to zero are better.
- **steps\_per\_sec** – This shows the average model training speed at each checkpoint.

Evaluation Metrics (metrics taken as the model is being tested/evaluated)

- **DetectionBoxes\_Precision/mAP** – This is the "mean average precision", which is an overall precision of detection/classification across all frames, labels, and bounding box thresholds and taking the average. This gives a view of how well the model is generally performing; values closer to 1.0 are better.
- **DetectionBoxes\_Precision/mAP (large, medium, small)** – This filters and separates the mAP metrics into three buckets based on the average pixel size of the detected objects and bounding boxes with respect to the model size. Values of -1 indicate that no objects met the size constraints for that bucket.

- **DetectionBoxes\_Precision/mAP(@.50IOU, @.75IOU)** – IOU stands for “Intersection Over Union”, also referred to as the Jaccard index, and is essentially a statistic used for gauging the similarity and diversity of sample sets. Normally an IOU  $>.50$  is considered a good prediction, and  $>.75$  is considered a really good prediction. These metrics are the average precision using only the specified IOU (but still going over all frames and labels). The idea of this metric is to give you a rough sense of accuracy of object detection if you are not super strict about the position of your bounding boxes (for example, in the .50IOU case, you only require at least IOU=0.5 to count as a match). Values closer to 1.0 are better.
- **DetectionBoxes\_Recall/AR(@1, @10, @100)** – These are “mean average recalls”, or a metric for specifically measuring object detection performance, bucketed by the maximum number of detections within the image (objects with only one detection would be in the @1 bucket, objects with at most 10 detections would be in the @10 bucket, and so on). The Recall metric is a metric that compares “true data” with “predicted data”, and provides an indication of the number of misdetections. A value of 1.0 means “all perfect detections”, and the more “misdetections” in the model the closer the value is to zero.
- **DetectionBoxes\_Recall/AR@100(large, medium, small)** – these are average recalls bucketed by the size of the detected bounding box. Notice the AR@100 in the metric – this means only images with at most 100 detections are used (typically this will mean all images for fml-tc). Buckets are equal to that of the /mAP metric above. Values of -1 indicate that no objects met the size constraints for that bucket.
- **Loss/classification\_loss** – Same as Loss/classification\_loss in Training Metrics, except this is for the Evaluation/Testing data.
- **Loss/localization\_loss** – Same as Loss/localization\_loss in Training Metrics, except this is for the Evaluation/Testing data.
- **Loss/regularization\_loss** – Same as Loss/regularization\_loss in Training Metrics, except this is for the Evaluation/Testing data.
- **Loss/total\_loss** – Same as Loss/total\_loss in Training Metrics, except this is for the Evaluation/Testing data.

To view model metrics, click on the Description link for the model in the Models tab you wish to view. This will open the “Monitor Training” viewer for that model.

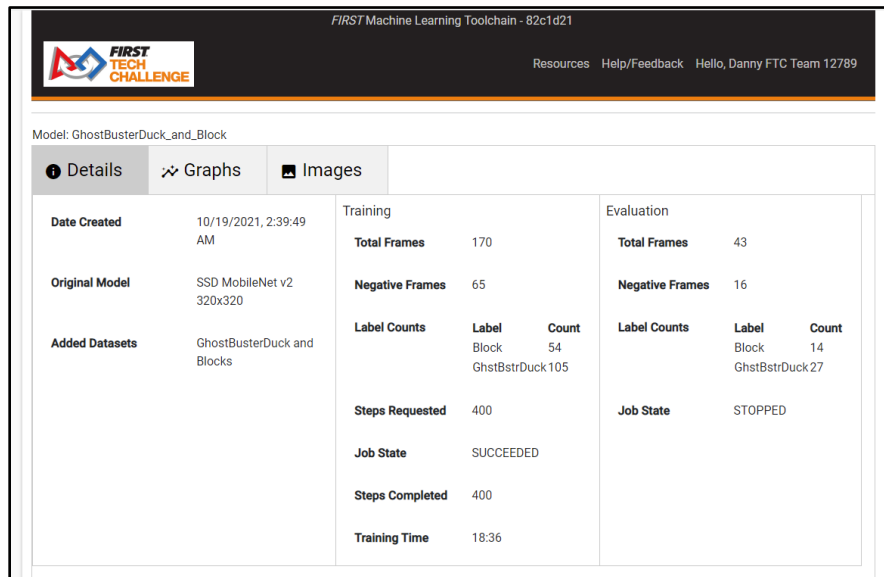


Figure 10: Viewing the model details in the "Monitor Training" viewer

The Monitor Training Viewer, seen in Figure 10, has 3 separate "tabs" within the viewer.

1. **Details** – Here the general training details are listed for the model. This includes which datasets were used to create the model, which model originated this model, training details, and evaluation details. This is the default tab when the Monitor Training viewer for the model is opened.
2. **Graphs** – This provides a scrollable viewer to see the graphs of specific performance metrics (discussed above). When the Monitor Training viewer is opened, the graphs may take several seconds to load – a rotating icon will show as the metric graphs are loaded.
3. **Images** – In the Images tab, you are able to see how well the model performed on each evaluation image at each 100-step checkpoint for each of the evaluation images in your data set. When the viewer is first opened, the images may need to load; a spinning icon in the images tab will be shown while loading. An example of the Images Tab can be seen in Figure 11 below. There are two copies of the same image side by side – each image represents one evaluation image in the Dataset. The image on the right always shows the bounding box labeled by the user, and always has a 100% detection shown on the bounding box. The image on the left shows the bounding box and detection percentage as predicted by the model at a specific checkpoint. In the example 400 steps were run and the images are showing the bounding boxes and detection rate of the 400-step checkpoint. Move the slider above each image to select a different checkpoint. The images are small, but to view the images larger right-click the image you wish to view and select "Open image in new tab" to open the image at full resolution in a new tab.

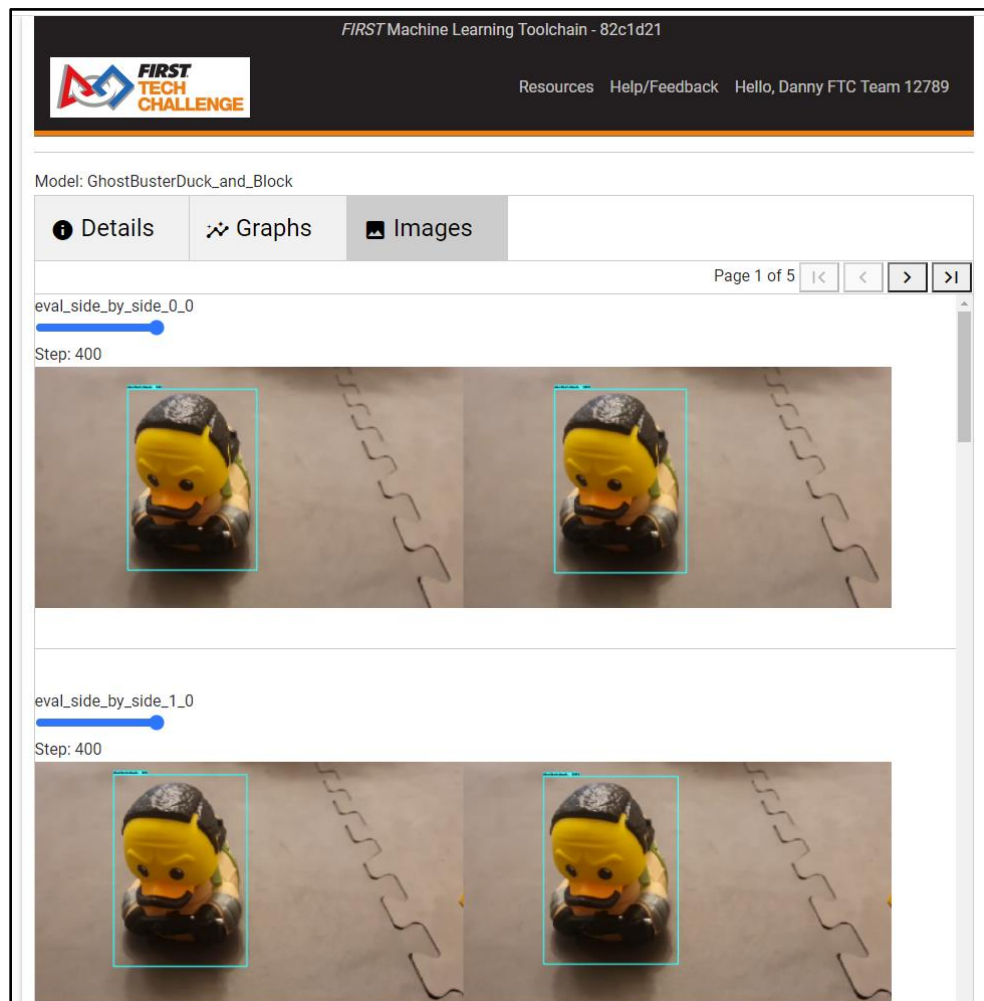


Figure 11: Viewing Training Image Performance in the Monitor Model viewer

## 5.9 Canceling Training

If one model is selected and that model's training is not finished and has not already been cancelled, the Cancel Training button is enabled. If a model has a checkpoint, the checkpoint can still be downloaded.

[Videos](#)
[Datasets](#)
[Models](#)

[More Training](#)
[Download Model](#)
[Cancel Training](#)
[Delete Models](#)

<input type="checkbox"/>	Date Created	Description	Starting Model	Steps Requested	Job State	Steps Completed	Training Time
<input type="checkbox"/>	10/12/2021, 12:52:57 PM	<a href="#">BallsModel</a>	SSD MobileNet v2 320x320	2,000	STOPPED	1,100	1:02:20
<input checked="" type="checkbox"/>	10/12/2021, 3:16:08 PM	<a href="#">BallModel Isolated</a>	SSD MobileNet v2 320x320	2,000	PREPARING	0	

Figure 12: Canceling Training on a model

### 5.10 Deleting a Model

If one or more models is selected and those models' training is finished, the Delete Models button is enabled.

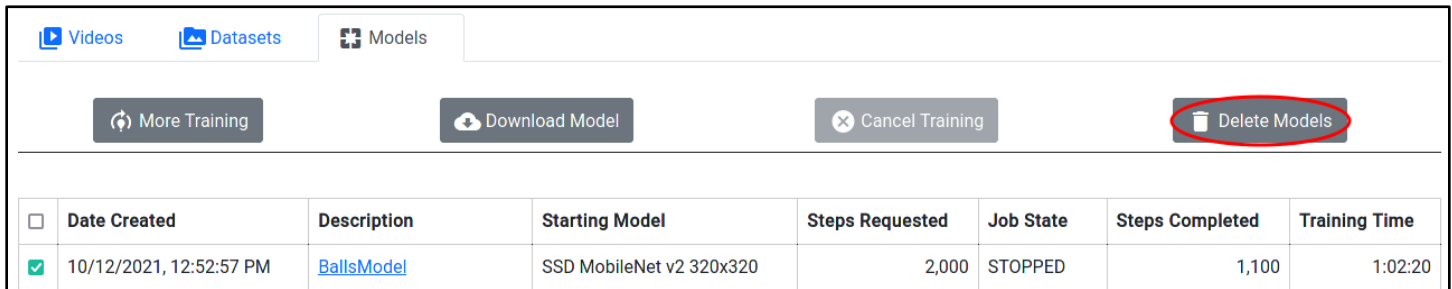


Figure 13: Deleting a model

When the user clicks Delete Models, the system determines whether the selected models can be deleted. Models that have been used as a starting point for more training cannot be deleted until after the other model is deleted.

A confirmation dialog is shown after the delete button has been pressed. If the users clicks Yes, the selected models will be deleted. If the selected models cannot be deleted, a dialog explaining why is shown:

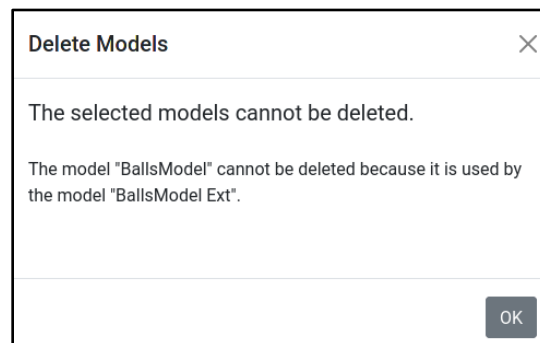


Figure 14: Cannot delete model because of dependencies

### 5.11 Downloading Models

In order to integrate models into your robot code, the models need to be downloaded first. If a model is selected and that model's training has finished and saved a checkpoint, the Download Model button is enabled.

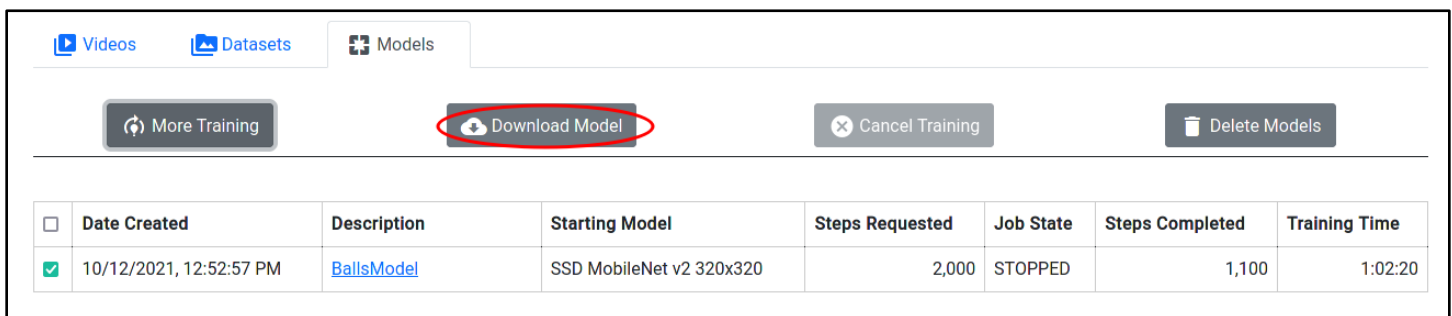


Figure 15: Downloading a Model

## 6. Using custom TensorFlow Models in Robot Code (by Uday Vidyadharan)

The basis of this tutorial will be the sample opmodes provided by FIRST in the [FTC 7.0 SDK](#). The process for testing a custom TensorFlow model is quite simple. To do this the general process flow is as follows:

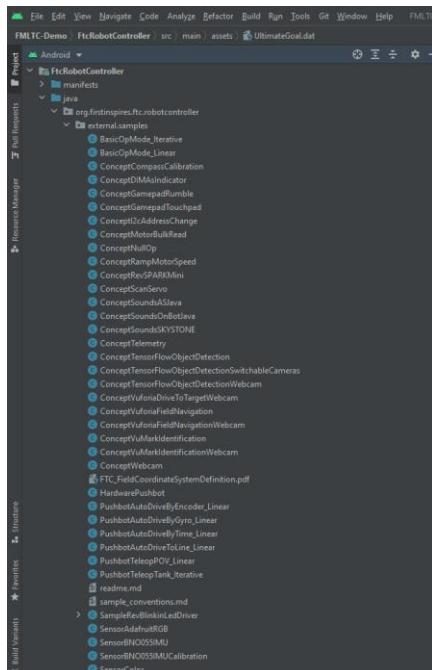
1. Use ftc-ml to build your custom TensorFlow model.
2. Create a new OpMode based on an appropriate sample OpMode.
3. Make relatively small changes to the new OpMode.
4. Add your model (.tflite file)

### 6.1 Android Studio

It is assumed that you already know how to use Android Studio. If not, be sure to check out the [Android Studio Guide for FIRST Tech Challenge](#) document before proceeding.

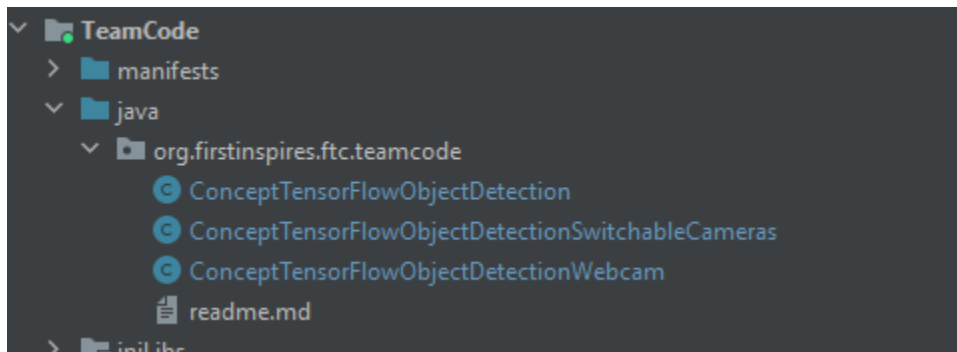
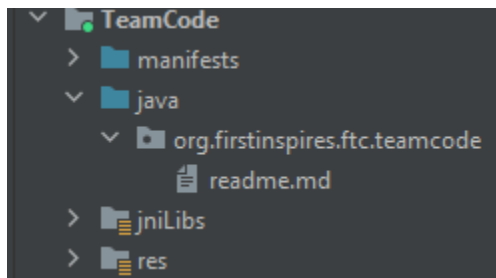
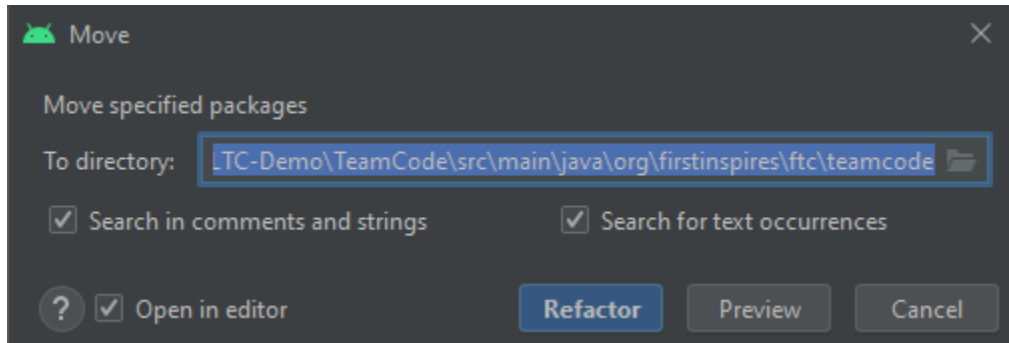
1. In Android Studio, there are three sample Opmodes to choose from. Each has its own use case. Depending on your use case choose one (or more) of the following.
  - `ConceptTensorFlowObjectDetection`
    - This opmode is designed to use the robot controller's internal camera, if it has one, for the input for TFOD.
  - `ConceptTensorFlowObjectDetectionSwitchableCameras`
    - This opmode is designed to be able to switch between two different webcams for the input for TFOD.
  - `ConceptTensorFlowObjectDetectionWebcam`
    - This opmode is designed to use an external webcam as its input for TFOD
2. We will now have to move the sample opmode from the samples folder to the teamcode folder. To do this we will first have to go to the project view on the right. Then we will need to navigate to
  - `FtcRobotController -> Java -> org.firstinspires.ftc.robotcontroller -> external.samples`

It should look like this:



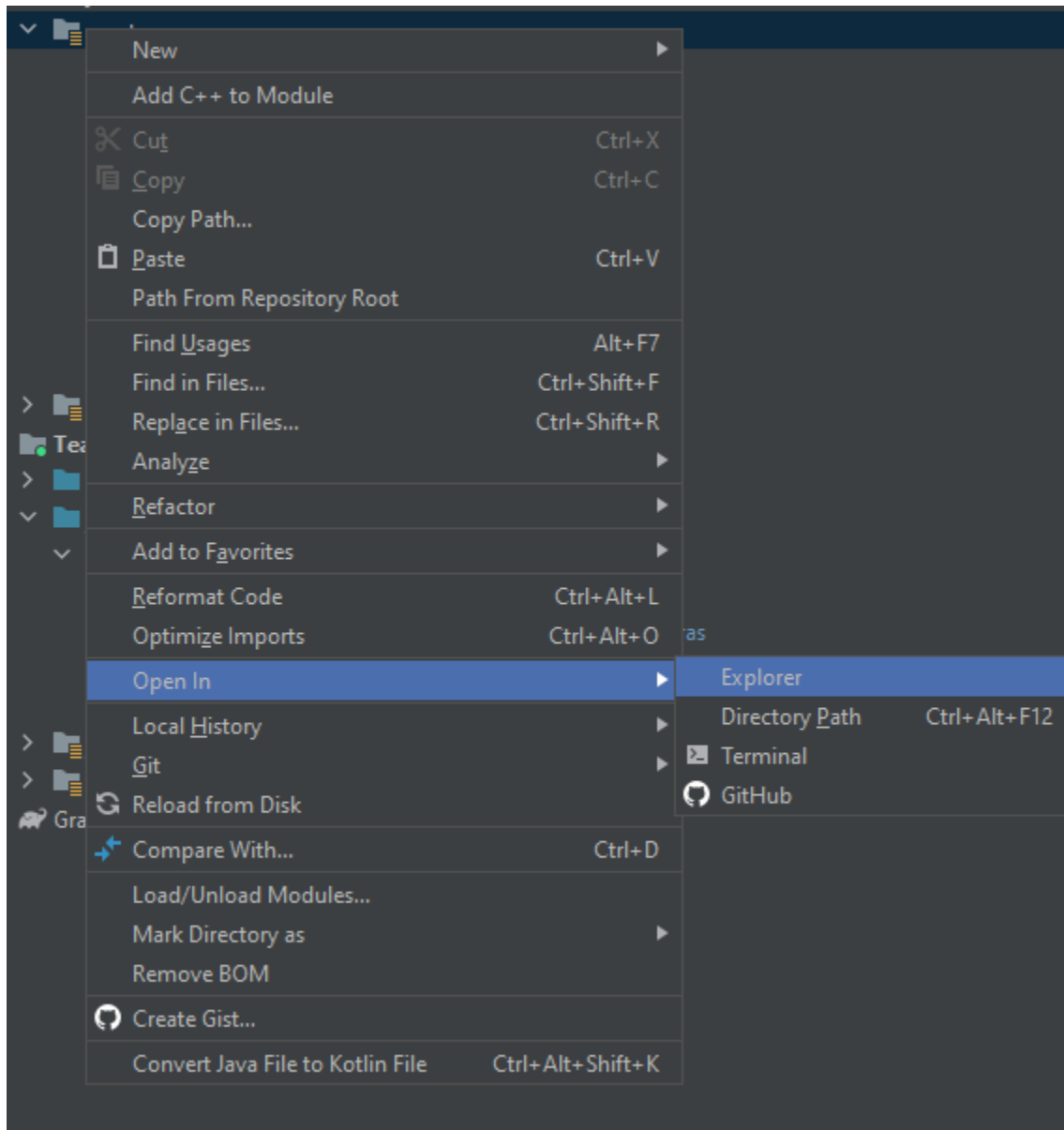
3. Select the opmode that you chose earlier and drag and drop it into the following folder:
  - TeamCode -> Java -> org.firstinspires.ftc.teamcode

When prompted to refactor select “refactor”. In the image below I moved all three opmodes into my teamcode folder but that is not necessary. You only need one opmode.



4. You will then have to open the opmode that you moved and remove the @Disabled should be near the top of the code.
5. The next step is to add your model (.tflite) to your assets folder. You will find this folder in your project view on the left by going to:
  - FtcRobotController -> assets.

By left- clicking on the assets folder you go to the “Open In” sub menu from which you can choose your file explorer of choice. From there you need to copy your model (.tflite) into the assets folder that you have just opened.



- Look for the line below in your opmode of choice. Change “FreightFrenzy\_BCDM.tflite” to the name of the model you just added to the assets folder.

```
private static final String TFOD_MODEL_ASSET = "FreightFrenzy_BCDM.tflite";
```

- Next we will have to acquire a Vuforia Key. There are detailed instructions in each of the opmodes on acquiring such a key.
- After following these instructions replace the following string with your key:  
`-- YOUR NEW VUFORIA KEY GOES HERE ---`
- (optional) You may find it necessary to adjust the zoom. By default it is set to 2.5 but feel free to adjust this to your needs. Note that this is digital zoom not optical zoom. This means that in essence you are simply telling TFOD to ignore outer sections of the image.



10. If you are using `ConceptTensorFlowObjectDetectionWebcam` or `ConceptTensorFlowObjectDetectionSwitchableCameras` you will also need to update the device name of the camera. Simply change the device name to correspond with your config. In the image below the name of the webcam is "Webcam 1".

```
private void initVuforia() {
    /*
     * Configure Vuforia by creating a Parameter object, and passing it to the Vuforia engine.
     */
    VuforiaLocalizer.Parameters parameters = new VuforiaLocalizer.Parameters();

    parameters.vuforiaLicenseKey = VUFORIA_KEY;
    parameters.cameraName = hardwareMap.get(WebcamName.class, deviceName: "Webcam 1");

    // Instantiate the Vuforia engine
    vuforia = ClassFactory.getInstance().createVuforia(parameters);

    // Loading trackables is not necessary for the TensorFlow Object Detection engine.
}
```

```
private void initVuforia() {
    /*
     * Configure Vuforia by creating a Parameter object, and passing it to the Vuforia engine.
     */
    VuforiaLocalizer.Parameters parameters = new VuforiaLocalizer.Parameters();

    parameters.vuforiaLicenseKey = VUFORIA_KEY;

    // Indicate that we wish to be able to switch cameras
    webcam1 = hardwareMap.get(WebcamName.class, deviceName: "Webcam 1");
    webcam2 = hardwareMap.get(WebcamName.class, deviceName: "Webcam 2");
    parameters.cameraName = ClassFactory.getInstance().getCameraManager().nameForSwitchableCamera(webcam1, webcam2);

    // Instantiate the Vuforia engine
    vuforia = ClassFactory.getInstance().createVuforia(parameters);

    // Set the active camera to Webcam 1.
    switchableCamera = (SwitchableCamera) vuforia.getCamera();
    switchableCamera.SetActiveCamera(webcam1);

    // Loading trackables is not necessary for the TensorFlow Object Detection engine.
}
```

11. You will next have to update the labels. To do this find the section of code shown below. This should match the order of the labels in the dataset(s) that your model is based on. There should be at least one label.

```
private static final String[] LABELS = {
    "Ball",
    "Cube",
    "Duck",
    "Marker"
};
```

(Before)

**Gracious Professionalism®** - "Doing your best work while treating others with respect and kindness - It's what makes FIRST, first."

```
private static final String[] LABELS = {  
    "FlashGordon",  
    "RedMarker",  
};
```

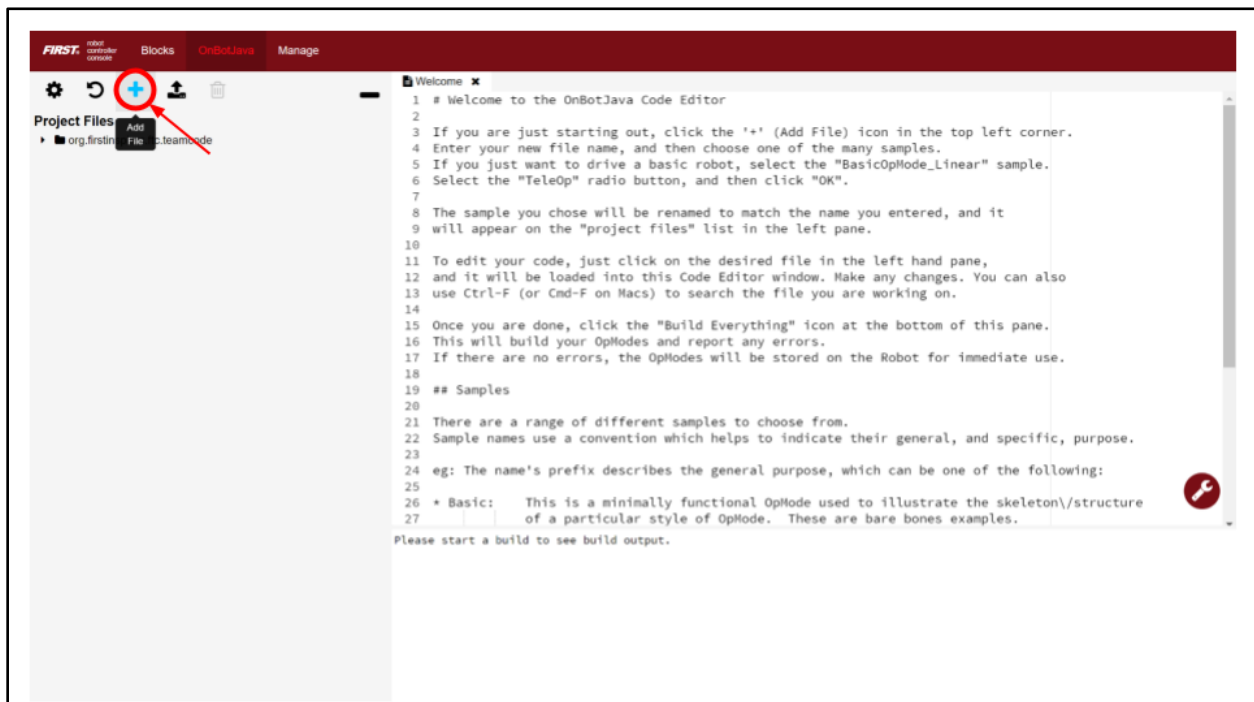
(After)

12. And you are all set to test. Just open the opmode as you would any other opmode and select initialize and play.

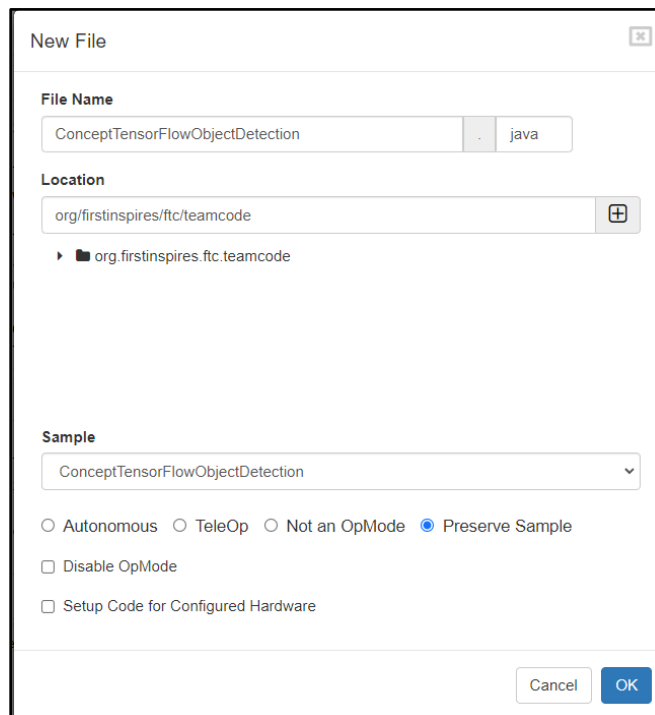
## 6.2 OnBot Java (OBJ)

It is assumed that you already know how to use OnBot Java. If not, be sure to check out the [OnBot Java Guide](#) document before proceeding.

- In On Bot Java, there are three sample Opmodes to choose from. Each has its own use case. Depending on your use case choose one (or more) of the following.
  - ConceptTensorFlowObjectDetection
    - This opmode is designed to use the robot controller's internal camera, if it has one, for the input for TFOD.
  - ConceptTensorFlowObjectDetectionSwitchableCameras
    - This opmode is designed to be able to switch between two different webcams for the input for TFOD.
  - ConceptTensorFlowObjectDetectionWebcam
    - This opmode is designed to use an external webcam as its input for TFOD
- Click the “Add Files” button in the top left corner.



- In the bottom half of the popup click the drop down below “Sample”. Then select the opmode that you chose earlier in Step 1. Then in the “File Name” text box enter the desired name of your opmode. Then click the blue “OK” button.



**New File**

**File Name**

**Location**

▶ **org.firstinspires.ftc.teamcode**

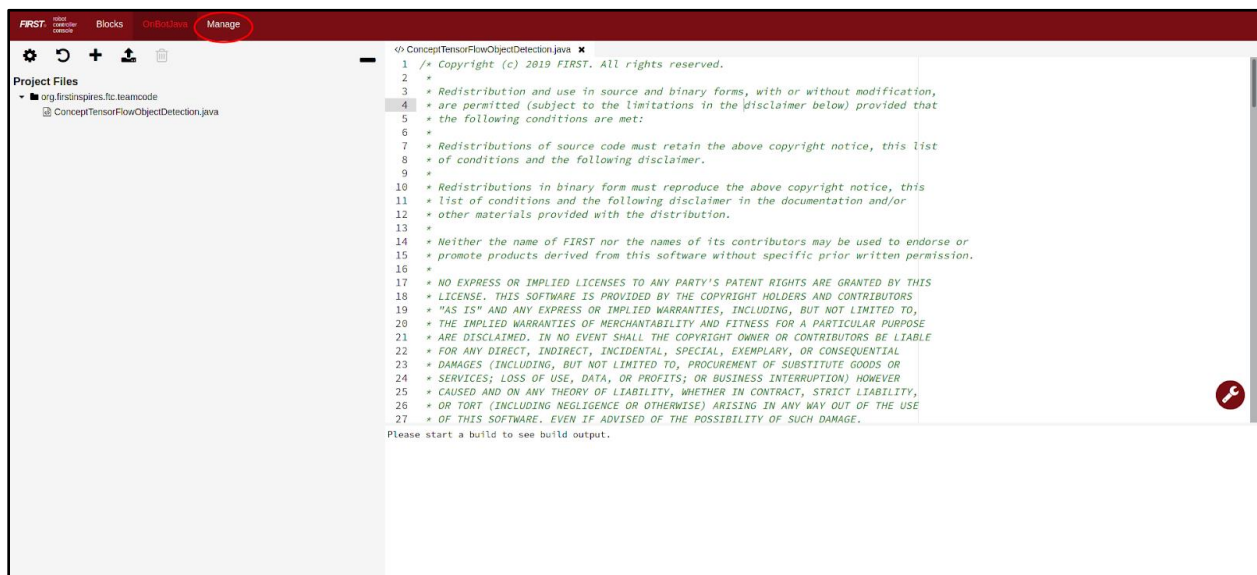
**Sample**

☐ Autonomous
 ☐ TeleOp
 ☐ Not an OpMode
 ☒ Preserve Sample

☐ Disable OpMode

☐ Setup Code for Configured Hardware

- The next step is to upload the TFOD model that you created using FTC-ML. To do this go to the ribbon on the top of your windows and select “Manage”.



- Next select the “Upload TensorFlow Lite Model File” button and select the model (.tflite). Then click upload.

6. Then navigate back to the “OnBotJava” page. You will now have to modify the name of the model that the opmode uses. To do this change the following line such that “FreightFrenzy\_BCDM.tflite” is replaced with the name of the file that you uploaded earlier.

```
private static final String TFOD_MODEL_ASSET = "FreightFrenzy_BCDM.tflite";
```

7. Change from “loadModelFromAsset” to “loadModelFromFile” in the line below in your opmode.

```
private void initTfod() {
    int tfodMonitorViewId = hardwareMap.appContext.getResources().getIdentifier(
        "tfodMonitorViewId", "id", hardwareMap.appContext.getPackageName());
    TFObjectDetector.Parameters tfodParameters = new TFObjectDetector.Parameters(tfodMonitorViewId);
    tfodParameters.minResultConfidence = 0.8f;
    tfodParameters.isModelTensorFlow2 = true;
    tfodParameters.inputSize = 320;
    tfod = ClassFactory.getInstance().createTFObjectDetector(tfodParameters, vuforia);
    tfod.loadModelFromFile(TFOD_MODEL_ASSET, LABELS);
}
```

8. Modify the labels of the model such that it matches those of your model. You need at least one label. The order should be alphabetical and be composed of the labels in the dataset(s) used to make the model.

```
private static final String[] LABELS = {
    "Ball",
    "Cube",
    "Duck",
    "Marker"
};
```

9. You will now need to acquire a Vuforia License Key. To do this follow the instructions provided in the opmode. They will be found above the line shown below. The final step will be to replace " -- YOUR NEW VUFORIA KEY GOES HERE --- " with your actual key.

10. (optional) You may find it necessary to adjust the zoom. By default it is set to 2.5 but feel free to adjust this to your needs. Note that this is digital zoom not optical zoom. This means that in essence you are simply telling TFOD to ignore outer sections of the image.

```
tfod.setZoom(2.5, 16.0/9.0);
```

11. If you are using `ConceptTensorFlowObjectDetectionWebcam` or `ConceptTensorFlowObjectDetectionSwitchableCameras` you will also need to update the device name of the camera. Simply change the device name to correspond with your config. In the image below the name of the webcam is "Webcam 1".

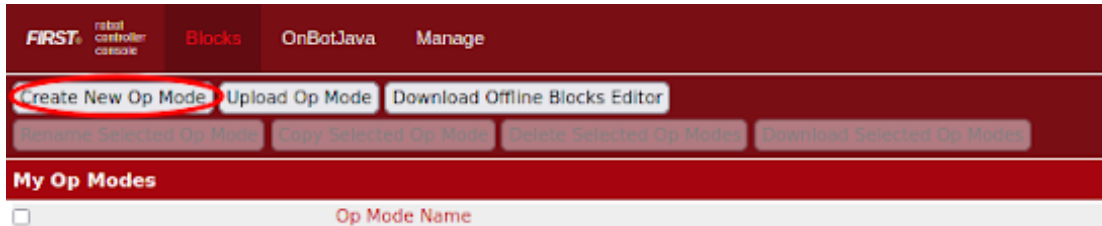
```
private void initVuforia() {  
    /*  
     * Configure Vuforia by creating a Parameter object, and passing it to the Vuforia engine.  
     */  
    VuforiaLocalizer.Parameters parameters = new VuforiaLocalizer.Parameters();  
  
    parameters.vuforiaLicenseKey = VUFORIA_KEY;  
    parameters.cameraName = hardwareMap.get(WebcamName.class, "Webcam 1");  
  
    // Instantiate the Vuforia engine  
    vuforia = ClassFactory.getInstance().createVuforia(parameters);  
  
    // Loading trackables is not necessary for the TensorFlow Object Detection engine.  
}
```

12. And you are all set to test. After clicking the build button just open the opmode as you would any other opmode and select initialize and play. Have fun testing.

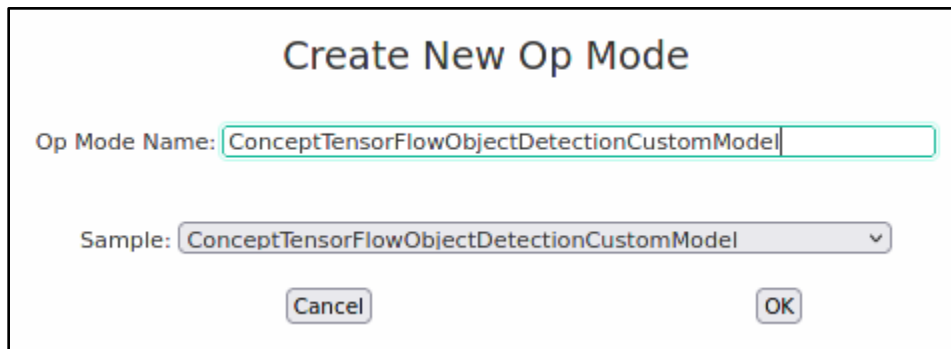
### 6.3 Blocks

It is assumed that you already know how to use Blocks. If not, be sure to check out the [Blocks Programming Guide](#) document before proceeding.

1. First click the “Create New Op Mode” button in the Blocks main management interface.



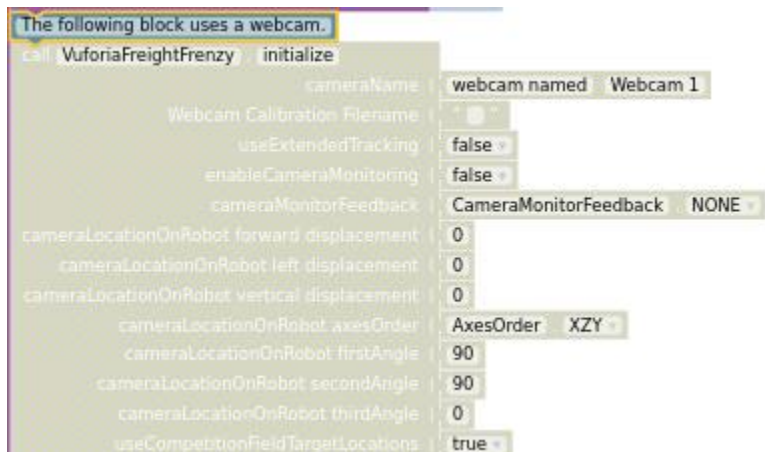
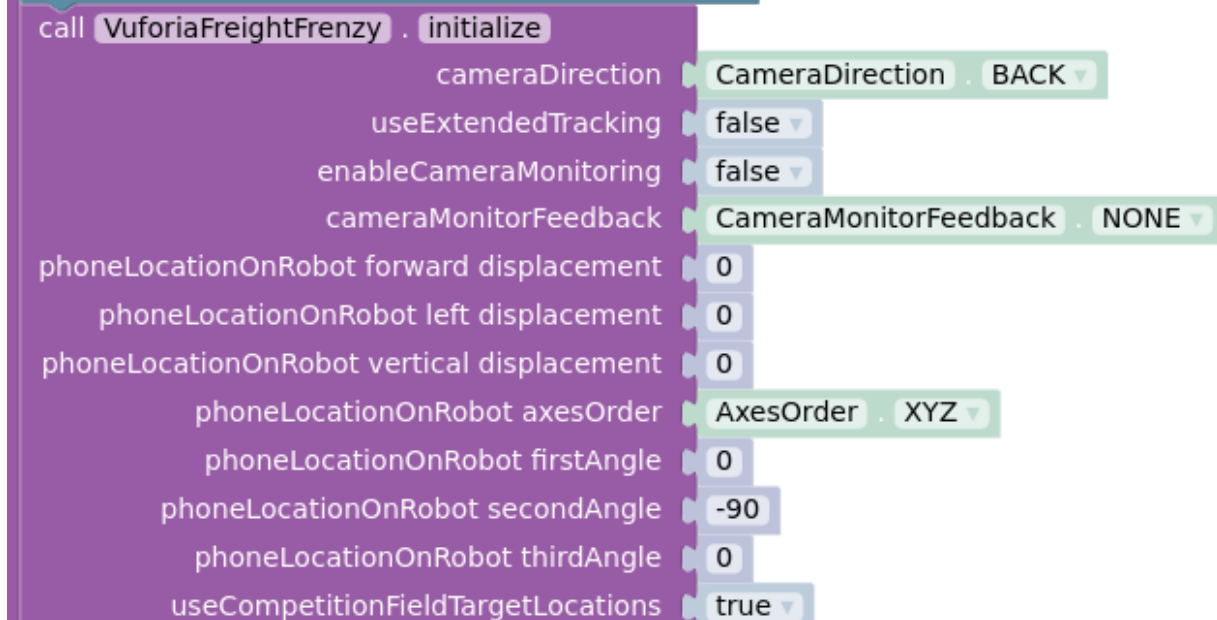
2. Select “ConceptTensorFlowObjectDetectionCustomModel” in the sample drop down and choose whatever name you find appropriate. Then click OK.



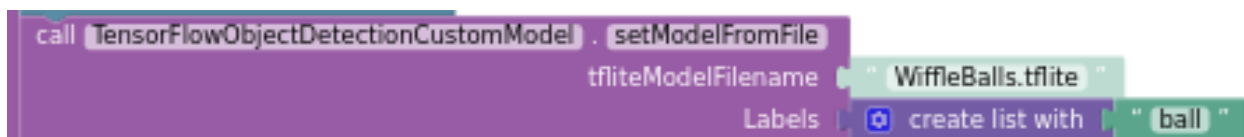
**NOTE:** Be careful of a common pitfall – as of FTC SDK 7.0, normal TensorFlow blocks cannot load custom tensorflow models. There are TWO kinds of TensorFlow blocks, one type for regular models and one type for custom models, and ALL of the blocks in each category MUST be used together (so old programs that load regular models would need ALL of the TensorFlow blocks replaced, not just the ones that load the model). If you choose to not create a new Op Mode for loading the Custom Models, and you do not replace ALL of the “normal” TensorFlow blocks, be aware that we warned you here.

3. You can either use a webcam or the built in camera for your video input. If you are using the control hub you, do not have a built in camera. If you are using the phone’s built in camera do not change the disabled block. If you are using a webcam, disable the block of code below the comment “The following block(first image below) uses the device's back camera.” To disable the block right click the block and select “Disable Block.” Then enable the following block (second image below). Do so by right clicking the block and clicking “Enable Block”.

The following block uses the device's back camera.

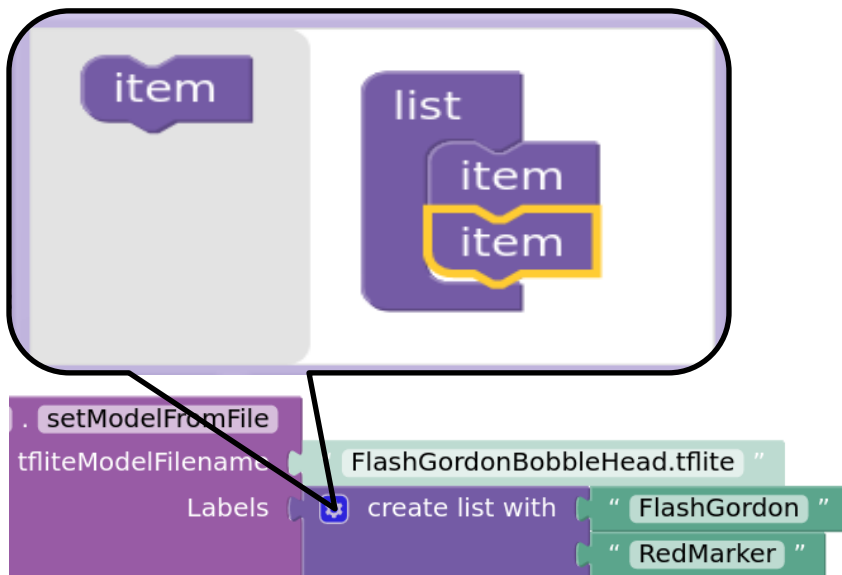


- If you are using a webcam change the cameraName parameter in the block that you just enabled to the name of your webcam. Currently it is called "Webcam 1."
- Then scroll down to the following block.



- Change tfliteModelFilename from "WiffleBalls.tflite" to the name of your model.
- Next you will have to update the labels. This should match the order of the labels in the dataset(s) that your model is based on. To do so click on the gear icon and add the needed number of items. Then add the items in your model to the labels list (seen below). This should be in alphabetical order and match the labels in the dataset(s) that your model is made from.

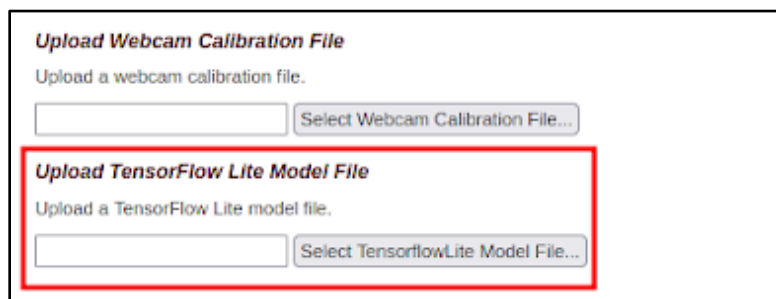




8. The next step is to upload the TFOOD model that you created using FTC-ML. To do this go to the ribbon on the top of your windows and select "Manage".



9. Next select the "Upload TensorFlow Lite Model File" button and select the model (.tflite). Then click upload.



10. And you are all set to test. After building the opmode as you would any other opmode and select initialize and play. Have fun testing.

## 7. Optimizing Videos for increased TensorFlow Model Performance

### 7.1 Background Information

Before diving into creating your first videos for TensorFlow training, it's important to cover a bunch of topics under the header of, "Things you should really know about TensorFlow for ftc-ml and were hopefully probably about to ask anyway". Here they are, in an order that hopefully makes some sense. Please read this in its entirety:

1. AI and Machine learning are **incredibly** resource-hungry operations.
  - For high-end performance, machine learning applications can run on special Google-designed Artificial Intelligence (AI) accelerator hardware chips known as Tensor Processing Units (TPU). These advanced chips are specialized for the high-volume low-precision computations required for AI processing, and are Google proprietary. TPUs generally consume large amounts of power when running in the Google datacenters. A TPU designed to consume far less power, known as the Pixel Neural Core, was introduced in the Pixel 4 smartphone in 2019 for machine learning applications.
  - For far less performance, machine learning applications can run on traditional Graphical Processing Units (GPU) typically found on graphics cards or embedded systems. Most modern cell phones contain GPUs, such as the Qualcomm Adreno 308 GPU found on the Moto E5 phone. However, performance is relative – the performance of GPUs found in cell phones is dwarfed by GPUs found in graphics cards or desktop systems.
  - For incredibly unreasonably low performance, machine learning applications can run on a general Central Processing Unit (CPU). Let's say no more about this and move along.
2. Building a TensorFlow model from scratch can take months of TPU time to train and refine the model properly. However, pre-trained models can be used as starting points to relatively quickly add novel (new) datasets. Therefore, Google provides a [TensorFlow Detection Model Zoo](#) that contains pre-trained models using the [COCO 2017 dataset](#) composed of over 120,000 images of common everyday objects classified into [81 different labels](#). The ftc-ml tool uses the SSD MobileNet v2 320x320 model as its default starter model from this Zoo – the TensorFlow models released in the [FTC 7.0 SDK](#) are based on this model too, so with the simple addition of labels to your Op Mode your "stock" TensorFlow models could recognize 81 additional categories of objects with no additional training needed. TRY IT OUT!
3. The performance of a TensorFlow model using Object Detection, even on TPU hardware, is completely dependent upon the core resolution of the model it's working with. The larger the core resolution, the more processing the model must perform. As an optimization, the core models in the TensorFlow Detection Model Zoo are trained on square (meaning equal width and height) resolutions of varying sizes. For TensorFlow models designed for Mobile applications, the core resolution is intentionally kept small. A 640x640 core model requires at least 4x the processing effort of a 320x320 core model; not all mobile devices can keep up with even 1-2 frames per second (fps) processing rates even on a 320x320 model!
4. Modern webcams have very high resolutions. The minimum resolution for an "acceptable" modern webcam is 720p. When scaling 720p, 1080p, or higher resolution images to a core model resolution of 320x320, fine details in the image are lost. The 16:9 aspect ratio source image is squeezed to a 1:1

aspect ratio image, making wide objects narrow (this is part of the reason why a webcam trained in a landscape orientation has poorer detection in portrait orientation). Small yellow objects in the source image suddenly turn into tiny indistinguishable blocky yellow blobs. The effects of the scaling process can be brutal.

5. To combat the effects of scaling, varying the “pose” of an object (orientation, angle, and distance from the camera) is incredibly important.
  - It is vital that the size of the object in the image be as large as possible when the camera is at maximum detection distance from the object. The larger the object is in the image, the more likely that scaling effects will have a lessened impact on the scaled image.
  - TensorFlow models are able to be more generically trained (that’s a good thing) when the objects are different sizes in different images. For example, including poses with the object at different distances from the camera is ideal. Building a labeled dataset with the object at different sizes helps the model recognize the objects better when they are different sizes.
  - If the object should still be recognized when it is rotated in any way, rotational variations are also important.
  - I hope you’ve realize this by now, but TensorFlow models follow the garbage-in garbage-out concept in model training. The more variations in size, rotation, angle, and orientation you can supply of the target object the more the model is going to be able to recognize/predict that target object.
6. TensorFlow Object Detection is not the best at recognizing geometries. Yes, this might run contrary to conventional wisdom in human object detection. Because a machine learning model is usually trained to be as general as possible, yellow circles and yellow octogons (depending on size) could be difficult to differentiate from each other (and from a generic yellow blob) depending on how the model is trained. Therefore, don’t expect TensorFlow to be really good at recognizing subtle differences in geometry.
7. Even though TensorFlow isn’t the best at recognizing geometries, it’s incredibly good at recognizing textures. No, probably not the kinds of textures you’re thinking about – we’re talking visual textures like zebra stripes, giraffe spots, neon colors, and so on. Colored patterns are TensorFlow’s strength. Careful Team Shipping Element design beforehand may yield great benefits later.
8. When creating videos for TensorFlow training, be very careful about the backgrounds being used. Machine Learning involves passing data and answers to a model, and letting the model determine the rules for detecting objects. If the background of an object is always consistent – let’s say the object is a duck on dark gray tiles – the model may include in its rules that the object must always be on a dark gray background, and will not recognize the duck on light gray tiles. In order to create a more generic model, the object would need to be found on multiple different backgrounds. “Negative Frames”, or frames that have no labels in them and are just of the background, can be used to help the model intentionally recognize “what is in the background” and that those elements of the background should be ignored; TensorFlow does this by adding the background patterns to an internal “background” label that is never shown to the user. It’s not typically necessary to include “Negative Frames”, however, unless there is content in the background that is only seen when the object is not present, and you feel it’s advantageous to ignore that content,. TensorFlow and modern Machine Learning algorithms isolate portions of each frame that do not include bounding boxes and add those portions of the image to the “background” label.
9. Related to backgrounds, lighting effects can cause issues with Object Detection. If the model is only trained with frames with objects that are extremely well lit, the model may not be very good when the

objects are not so well lit. It's important to get videos/frames of different lighting conditions if it's possible that the lighting conditions could differ between training and competition venues. One [classical urban legend about tank detection](#) in the early 1990's gives a pretty good warning about dataset bias.

10. If multiple similar-looking objects could possibly be in a frame and you only want the model to ever recognize one of them (for example you could have Yellow Blocks and Yellow Ducks in the same frame, but you ONLY want the model to detect Ducks) it is advised that yellow blocks be present but unlabeled in multiple frames. This allows the background detector to pick up the yellow blocks as background items, and be trained (covertly) to not recognize blocks as ducks by accident. There is no need to label objects in a model unless you want TensorFlow to specifically learn them.
11. Play like you train, and train like you play. This is just a poor way of saying, "try your best to video how your robot will see the objects in competition, and try your best in competition to make sure that your robot only sees the objects like you trained the model". This has been said in different ways multiple times, but it needs to be repeated. The most likely reason a model will have poor performance in competition is because something has changed – whether that be the lighting is different, more/different objects are in the background, the pose of the objects are too different from those during training, and so on.
12. This might not need to be said, but avoid "floppy" or "non-rigid" objects. For example fabric that can be folded or bunched up, flexible objects with joints that can move, or structures that can easily bend. Models still might be able to differentiate some of the possible variations, but the likelihood that it doesn't when it matters is too great.

## 7.2 FAQ

1. Why is TensorFlow called "TensorFlow"?
  - The name TensorFlow is derived from the single- and multi-dimensional arrays that neural networks perform operations on, known as "*tensors*". Data in a neural network "flows" through the network as its being classified, passing through weighted nodes. Hence TensorFlow. There were apparently multiple projects known as "TensorFlow" that sprung up at the same time.
2. How many frames of our object is enough to ensure a good model?
  - Thanks for asking, come back and let me know when you find the answer. To be honest, that's going to be completely dependent upon the object, poses you're trying to account for, backgrounds, and lighting conditions. As long as the base model originated from TensorFlow's Model Zoo, additional novel objects don't require thousands of training frames, 200 frames could possibly get the job done, if optimized. Exceeding 1,000 frames for a single object is likely overkill.
3. How do I know if my model is trained well?
  - There are a number of metrics in Section 5.8 that can help you determine when a model begins to converge (where additional training will likely lead to no benefit). Pay special attention to mAP metrics and Loss metrics, you should see those metrics begin to "level out" after 400-500 steps on average.
4. Why does my team only get 120 minutes of model training time?
  - Training in the Google TensorFlow network on TPU resources is quite expensive. Each team is allocated an amount of time based on the costs of using the TPU resources. Our hope is that a

team who is cognizant of their training time should be able to get 2-3 models and additional training time on one model with that allocation.

- It is not possible for a team to “purchase” additional training time for their account. We’re hoping teams will give us feedback on what they feel a reasonable amount of time could be (and let us figure out how to allocate those resources).
5. Why can’t I seem to get a 100% object detection prediction?
    - Model predictions are never perfect, and attempting to strive for that makes for a really specific and non-generic model. If object detection probability is really high (in the 90-99% range), it might be pointing out that your model may not be as generic as it could be; it depends on the datasets and what you’re trying to do. Generally after training if your model is predicting all objects above 50% all the time you’re actually doing really well.
  6. I read somewhere about a parameter I can tweak...
    - There are no parameters to tweak in ftc-ml, sorry. It was designed to be simple and easy to use. If you want, feel free to clone your own [fmltc repository](#), modify the code, and deploy it to your own Google Cloud Project instance! However, swim at your own risk.
  7. Can object bounding boxes overlap?
    - Sure, but if you have “blocks” in front of a “ball” such that objects are obscuring each other, just label the parts that are not obscured. Don’t include areas in your bounding box where “there would be the rest of the ball here if it wasn’t obscured by these blocks”
  8. What are the maximum limitations imposed within the ftc-ml tool for various actions? (PER TEAM)
    - Max # of Datasets: 20 (you can delete datasets to make more)
    - Max # of Videos: 50 (you can delete videos to upload more)
    - Max # of Videos performing tracking at once: 3 (for multiple logins doing tracking)
    - Max # Bounding Boxes per frame: 10
    - Max Video Limits: 2 Minutes, 1000 frames, 3840 x 2160 resolution, 100MB

## Volunteer Special Thanks

---

The *FIRST* Tech Challenge staff would like to extend a special thanks to the following volunteers for their hard work and dedication toward this project:

- Liz Looney, Google – FIRST Machine Learning Toolchain lead developer
- Mr. Phil Malone – Model designer and platform tester
- Uday Vidyadharan – Platform tester and Contributor
- Jacob Burroughs – Platform configuration and FTC Scoring SSO
- Richard Lester – Platform UI improvements