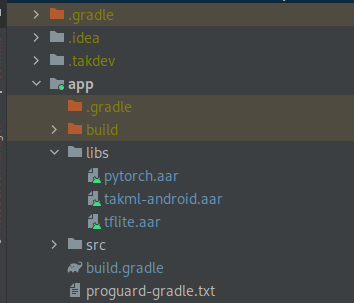
|  |  |
| --- | --- |
| TAK ML MLA Plugin Development  Model Learning Application plugin development with TAK ML  June 8th, 2023 |  |

**Definition**

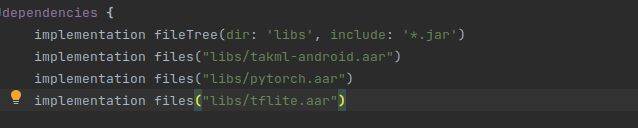
The MLA Plugin, or Machine Learning Application plugin, is a plugin that uses TAK ML and MX Plugins to run ML models on ATAK. It is user facing. This guide explains how to use TAK ML in ATAK plugins.

**1. Create the Android ATAK Project**

Each TAK ML MLA plugin needs the TAK ML AAR dependency and any Mx Plugin AAR dependencies they wish to use. To add the AAR, create a libs folder in your TAK plugin and copy the TAK ML Android AAR there along with any Mx Plugin dependency.

The below screenshot shows TAK ML AAR with two MX Plugin Dependencies (Pytorch and TF Lite).  


Next, declare the dependencies in your build.gradle:



The TAK ML Android AAR requires no additional dependencies, however the MX Plugin AARs can. Please see the documentation for each MX Plugin to see which you will need to include. For example, the Pytorch and TFLite MX Plugins require the following:



**3. Using the TAK ML libraries**

First, you will need to instantiate a **Takml** object, passing the ATAK plugin context:

|  |
| --- |
| Takml takml = new Takml(pluginContext) |

Next, you may optionally choose to add a **TakmlInitializationListener**. This is particularly useful to have your application aware when TAK ML has finished reading all the TAK ML models from the sd card and has finished initializing. To do so:

|  |
| --- |
| takml.addInitializationListener(new TakmlInitializationListener() {  @Override  public void finishedInitializing() {  }  }); |

TAK ML Models are automatically read from disk and available to the user (please see Section 3):

|  |
| --- |
| // Get all available TAK ML Models takml.getModels()  // Get TAK ML Model given friendly name: takml.getModel(friendlyName) |

Additionally, TAK ML Models can be imported programmatically. If the persistToDisk flag is set, it will be written to disk (and will be available to use even on restart of ATAK). This may be particularly useful if desiring to embed models with an apk install.

|  |
| --- |
| takml.addTakmlModel(takmlModel, persistToDisk); |

Next, you can create a **TakmlExecutor**. This allows executing a TAK ML Model:

|  |
| --- |
| TakmlExecutor takmlExecutor = takml.createExecutor(takmlModel); |

You can then invoke a prediction, passing the input data and an **MXExecuteModelCallback**. The MXExecuteModelCallback returns a **TakmlResult**, whether the execution was successful or not, and the output model type (e.g. ModelTypeConstants.IMAGE\_CLASSIFICATION).

|  |
| --- |
| TakmlExecutor.executePrediction(inputData, new MXExecuteModelCallback() {  @Override  public void modelResult(List<? extends TakmlResult> takmlResults, boolean success, String modelType) {  } }); |

Note the takmlResults, which returns a list of prediction results. The type of TakmlResult depends on the type of TAK ML Model.  
  
At the time of this writing, the following TakmlResult options are available:  
**Recognition** – For Classification and Object Detection results  
**Regression** – For Linear Regression  
  
Example:

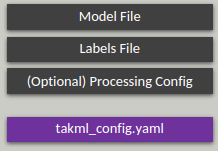
|  |
| --- |
| takmlExecutor.executePrediction(bitmapImageBytes, new MXExecuteModelCallback() {  @Override  public void modelResult(List<? extends TakmlResult> takmlResults, boolean success, String modelType) {  for (TakmlResult takmlResult : takmlResults){  if(modelType.equals(ModelTypeConstants.IMAGE\_CLASSIFICATION) || modelType.equals(GENERIC\_RECOGNITION)){  Recognition recognition = (Recognition) takmlResult;  String label = recognition.getLabel();  float confidenceScore = recognition.getConfidence();  }else if(modelType.equals(ModelTypeConstants.OBJECT\_DETECTION)){  Recognition recognition = (Recognition) takmlResult;  String label = recognition.getLabel();  float confidenceScore = recognition.getConfidence();  float leftImageCoord = recognition.getLeft();  float rightImageCoord = recognition.getRight();  float topImageCoord = recognition.getRight();  float bottomImageCoord = recognition.getBottom();  }else if(modelType.equals(ModelTypeConstants.LINEAR\_REGRESSION)){  // is linear regression  Regression regression = (Regression) takmlResult;  float result = regression.getPredictionResult();  }  }  }  }); |

Note: An Mx plugin developer can also create their own type of TakmlResult. See the documentation for each Mx plugin. Hence why this was designed to be generic.

Finally, to shut down the TAKML Executor:

|  |
| --- |
| takmlExecutor.shutdown(); |

**4. Importing a TAK ML Model (Optional)**

To use TAK ML, **TAK ML Models** must be defined. A TAK ML Model is a wrapper that includes the trained ML model and accessory files. The format is the following:  


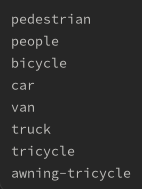
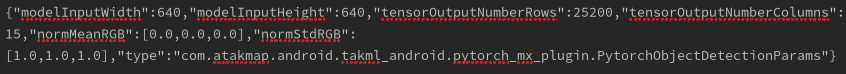
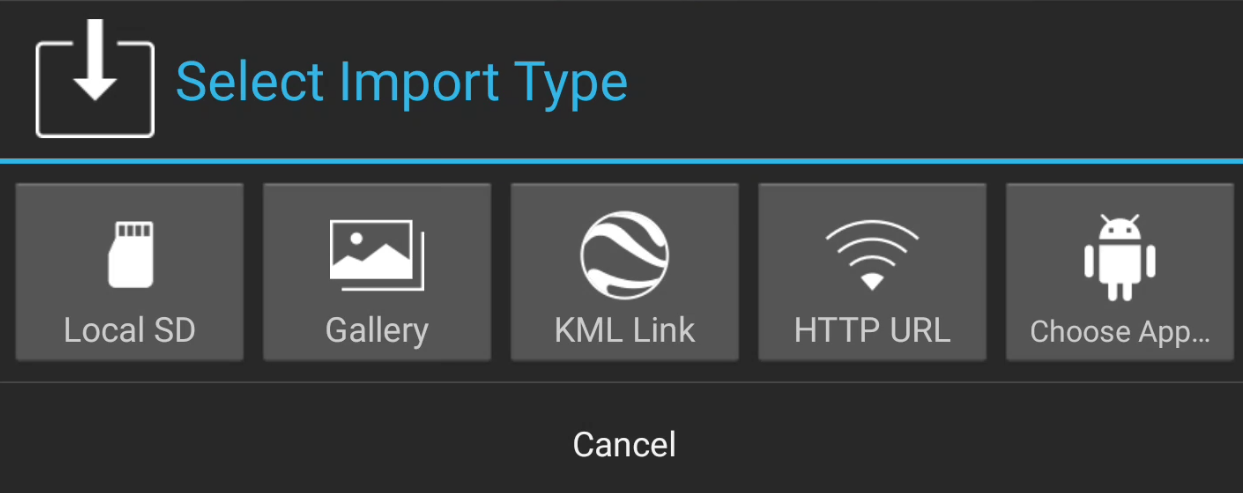
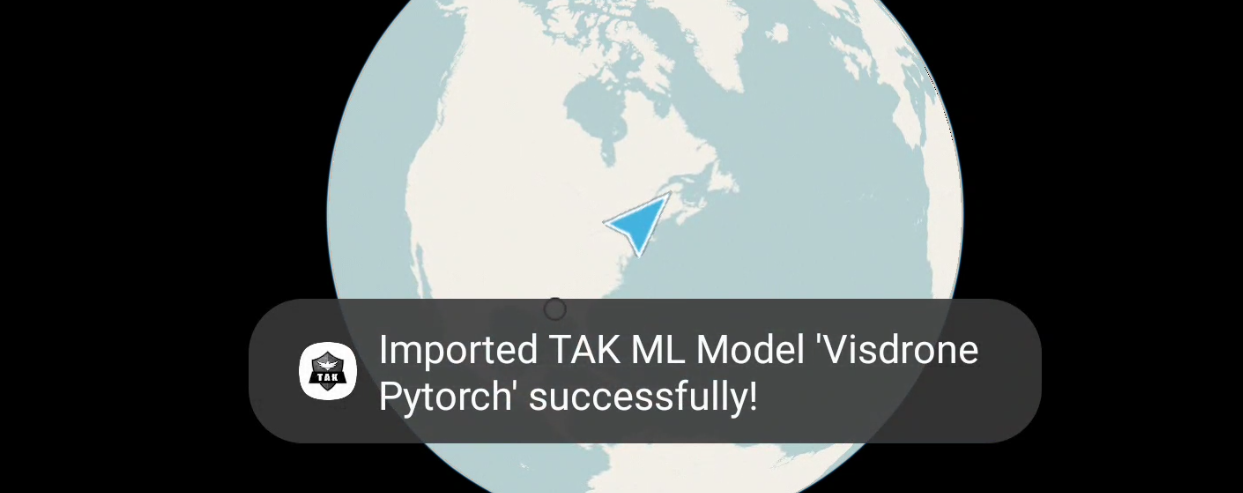
There are two methods of creating a TAK ML Model:

1. Via importing a Zip File in ATAK
2. Programmatically (see step 3 for further detail)

Creating a TAK ML Model Zip file is straightforward:

1. **Assure that your MLA plugin in installed and activated**.
2. Create an empty folder and create a takml\_config.yaml file. Note, it must have this name. This Yaml file is a pointer to all the related files. It’s structure is as follows:

|  |
| --- |
| **friendlyName**: [friendly name for your TAK ML Model]  **modelType**: [Model Type, e.g. IMAGE\_CLASSIFICATION, OBJECT\_DETECTION, GENERIC\_RECOGNITION, or LINEAR\_REGRESSION]  **modelName**: [name of your trained model file]  **labelsName**: [if applicable (e.g. Object Detection or Classification) name of your class labels file, if applicable]  **processingConfig**: [if applicable (e.g. Pytorch Object Detection), name of your Processing Config, which provides specific info on how to process the model] |

1. Add your model file
2. Add your labels file, if applicable (e.g. not Linear Regression). This is a line-separated list of classes. For example:  
   
3. If applicable, add your Processing Config. Currently, this is only applicable to Pytorch Object Detection. For example:  
   Please see the ”TAK ML Pytorch - Object Detection with Yolov5 Developer Guide” for more information.
4. Zip up the file and copy to your Android device
5. Use the ATAK Import Tool to import the Zip File:
   1. Open Hamburger Menu and find the Import Tool  
      
   2. Select “Local SD”  
      
   3. Finally locate your TAK ML Model Zip file
   4. It may take up to several seconds, but a notification stating the model has been imported should appear  
      
   5. The model is now copied to /sdcard/takml and imported into your TAK ML MLA application.