Classifying Urban sounds using Machine Learning and Deep Learning

Model Deployment Options

1. Generate C Code for embeded devices using MATLAB coder
2. Create standalone applcation using MATLAB compiler

Generate C Code for Prediction - Supported Models Only

If you train an SVM model using Classification Learner, you can generate C code for prediction.

C code generation requires:

- MATLAB Coder™ license
- SVM model (binary or multiclass)
- No categorical predictors or response in your data set

After you train an SVM model in Classification Learner, export the model to the workspace.Find the name of the classification model object in the exported struct. Examine the fields of the struct to find the model name, for example, `C.ClassificationSVM` where `C` is the name of your struct, e.g., `trainedModel`.Model name depends on what type of SVM you trained (binary or multiclass) and whether you exported a compact model or not.

Models can be

- `ClassificationSVM`
- `CompactClassificationSVM`
- `ClassificationECOC`
- `CompactClassificationECOC`

Use the function `saveLearnerForCoder` to prepare the model for code generation:

- `saveLearnerForCoder(Mdl,filename)`
- `saveLearnerForCoder(C.ClassificationSVM, 'mySVM')`

Create a function that loads the saved model and makes predictions on new data. For example:

function label = classifyX(X) %#codegen

Mdl = loadLearnerForCoder('bagtree');

label = predict(Mdl,X);

end

Save function in a script

Generate a MEX function from your function. For example:

```
x = dataTest{1,1:13};
```

```
codegen classifyX.m -args
```

The %#codegen compilation directive ixndicates that the MATLAB code is intended for code generation. To ensure that the MEX function can use the same input, specify the data in the workspace as arguments to the function using the `-args` option. `data` must be a matrix containing only the predictor columns used to train the model.

Use the MEX function to make predictions. For example:

```
labels = classifyX_mex(x)
```

```
labels = 1×1 cell array
    {'Dog Bark'}
```

TODO:

1. From the Instructions above use it for your choosen model
2. Use MATLAB Coder APP ignorning the "Generate a MEX Function from your function"
3. Compare Difference of a MEX function against MATLAB Function
4. Plot the difference in output
5. Display Execution time Using Tic Toc
6. Conclude

```
tic;
label_matlab=classifyX(x)
```

```
label_matlab = 1×1 cell array
    {'Dog Bark'}
```

```
a=toc;

tic;
label_mex = classifyX_mex(x);
b=toc;

comparison = table(["MATLAB","MEX"]',[label_matlab,label_mex]',[a,b]');
comparison.Properties.VariableNames = {'Name' 'Prediction' 'Prediction_Speed'}
```

```
comparison = 2×3 table
```

	Name	Prediction	Prediction_Speed
1	"MATLAB"	'Dog Bark'	2.8116
2	"MEX"	'Dog Bark'	0.0297

you can notice the performance of MEX file is faster than MATLAB significantly.

Deploy Predictions Using MATLAB Compiler - Any model will do

After you export a model to the workspace from Classification Learner, you can deploy it using MATLAB Compiler™.

Suppose you export the trained model to MATLAB Workspace based on the instructions in Export Model to Workspace, with the name `trainedModel`. To deploy predictions, follow these steps:

Save the `trainedModel` structure in a .mat file.

```
save mymodel trainedModel
```

Write the code to be compiled. This code must load the trained model and use it to make a prediction. It must also have a pragma, so the compiler recognizes that Statistics and Machine Learning Toolbox™ code is needed in the compiled application. This pragma could be any function in the toolbox.

function ypred = mypredict(tbl)

Mdl = loadLearnerForCoder('bagtree');

label = predict(Mdl,X);

end

save above script to mypredict.m

Compile as a standalone application.

```
mcc -m mypredict.m
```

You may use app designer to design your GUI, then later compile it as standalone desktop application

- you may refer to my designed app : app1.mlapp, open it in app designer

```
appdesigner
```