GSoC 2020 - TMVA PyTorch Interface

TMVA PyTorch Interface

Handles

- Preprocessing
- Dataloading
- Validation
- Testing
- Comparision

Handles

- Model Definition
- Training
- Prediction

Why TMVA PyTorch Interface?

- Ease of use
- Ease of debugging
- Power & Flexibility to user.
- PyTorch prevalent in both industry and research.

Setup

```
void MethodPyTorch::SetupTorchModel(bool loadTrainedModel) {

// Load initial model or already trained model
      /*

       * Load pytorch model from file

       * Load pytorch user code methods
                * Optimizer
                * Loss Criterion
                * Train Method
                * Predict Method

       * load_model_custom_objects = {"optimizer": optimizer, "criterion":
                       criterion, "train_func": fit, "predict_func": predict}

       * Give option of using state dict or whole model to load, depending on what
         type of model is saved

       * Init variables and weights

       */
}
```

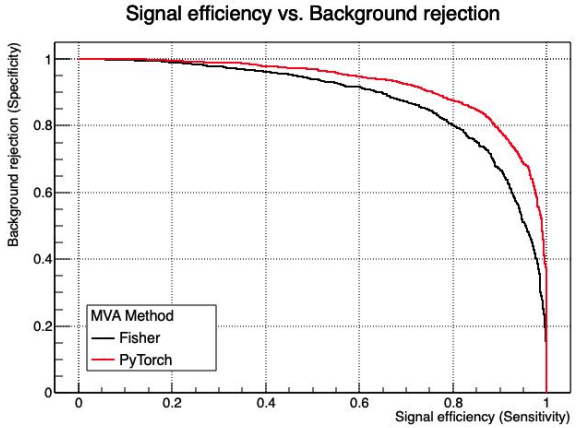
Train

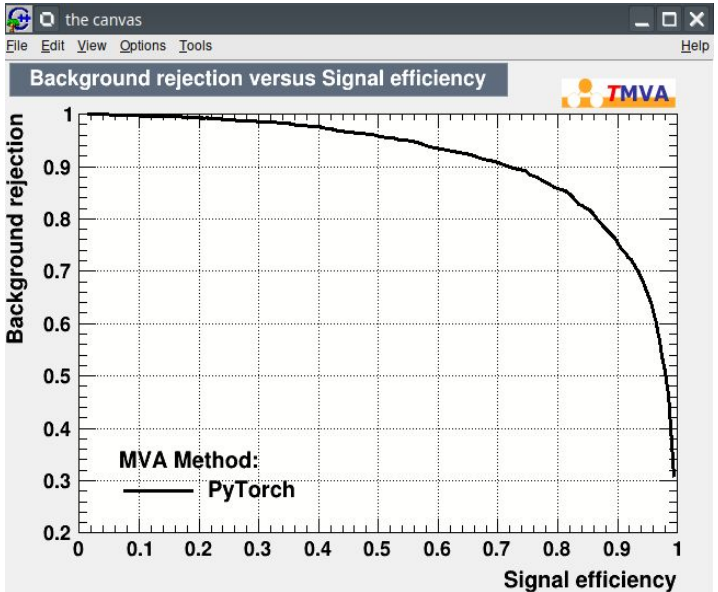
```
void MethodPyTorch::Train() {

  /* Setup parameters
   * Setup training alternatives to callbacks like keras
   *
   * Store trained model to file (only if option 'SaveBestOnly' is NOT activated,
   * because we do not want to override the best model checkpoint)
   *
   * Load PyTorch model from checkpoint .tar file or .pt/.pth file
   * Give option of using state dict or whole model to load depending on what type
     of model is saved
   */

   // Load initial model or already trained model

   // Start model training




}
```

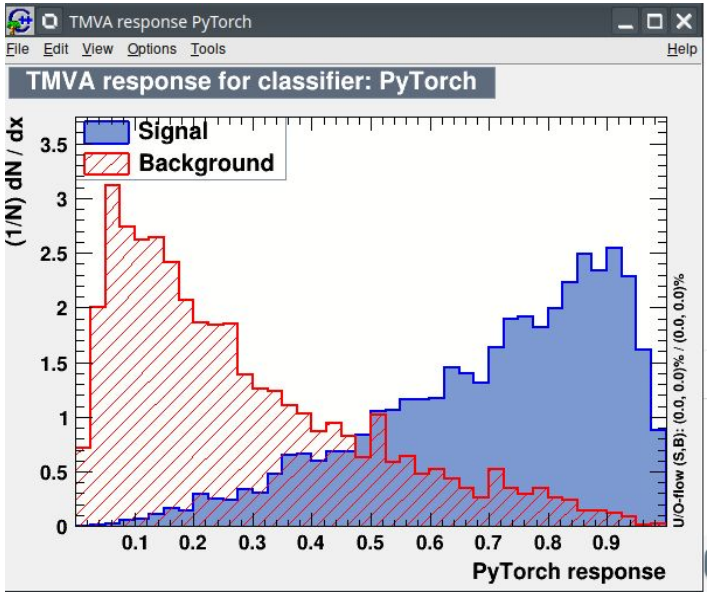
Tests

```
int testPyTorchClassification(){
   /*
    * Load PyTorch model & Build model from python file
    * Setup PyMVA and factory & Load data
    * Book and train method
    * Setup reader
    * Get mean response of method on signal and background events
    * Check whether the response is obviously better than guessing
    */

int testPyTorchMulticlass(){
   /*
    * Load PyTorch model & Build model from python file
    * Setup PyMVA and factory & Load data
    * Book and train method
    * Setup reader
    * Get mean response of method on signal and background events
    * Check whether the response is obviously better than guessing
    */

int testPyTorchRegression(){
   /*
    * Load PyTorch model & Build model from python file
    * Setup PyMVA and factory & Load data
    * Book and train method
    * Setup reader
    * Get mean response of method on signal and background events
    * Check whether the response is obviously better than guessing
    */
```

Results & Plots







Conclusion

- Achieved all the targets proposed before GSoC
- Developed a fully functional PyTorch Interface in TMVA
- Implemented tests
- Implemented tutorials
- PR #5757 & #6273 (Major GSoC Contributions)

Thank You

Blog: https://anirudhdagar.github.io/gsoc/