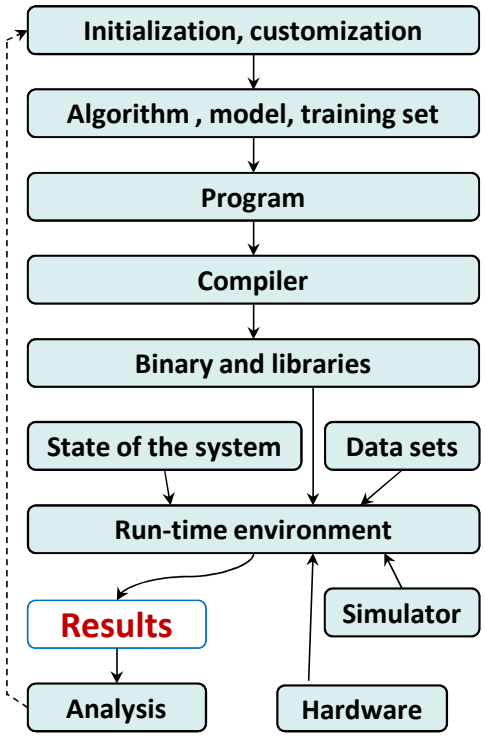
Typical directory structure of an ad-hoc experimental pack shared for Artifact Evaluation

Experiment workflow

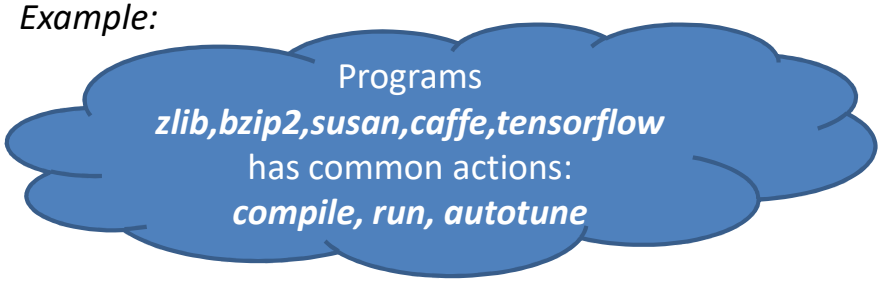



```
scripts/ download_dataset.sh
         download_dnn_model.bat
         init_setup_rpi3_gcc7.1.0.sh
         init_setup_windows10.bat
         init_setup_android.sh
         list_programs.sh
         compile_program.py
         run_program.py
         analyze_results.sh
         build_predictive_model.bat
         plot_graph.sh
programs/ bzip2
         classify-image
         decode-video-stream
datasets/ jpg-images/ 1.jpg, 2.jpg, 3.jpg
         png-images
         videos
third-party-tools/ gcc-7.1.0
         llvm-4.0
         opencl-profiler , cuda-profiler
         arch-simulator
         caffe, caffe2, tensorflow, cntk,
         mxnet, clbast, openblas, libdnn
some-meta/ gcc-7.1.0-compiler-flags.txt
         llvm-4.0-compiler-flags.txt
         rpi3-hw-description.txt
some-results/ reference-speedups.txt
         predictions.cvs
         graph-autotuning-rpi3.xls
```

(a)

Reorganize objects with common actions and meta info into 2 level directories with auto-generated Unique ID, Python module (object API) and JSON meta information

Example:



Convert code and data into CK repo (2 main dir. levels)

1 level (API)	2 level (entry)	
.ckr.json		- repo desc. including deps on other repos
module	/program	/module.py – unified CK JSON API (functions: compile, run, autotune)
		/.cm/meta.json – JSON description of a program module
program	/zlib	/*.c … - program sources, build files
		/.cm/meta.json - JSON desc of zlib
	/.cm	/* - UID for zlib
.cm	/*	- UIDs for module and program

must be the same

(b)