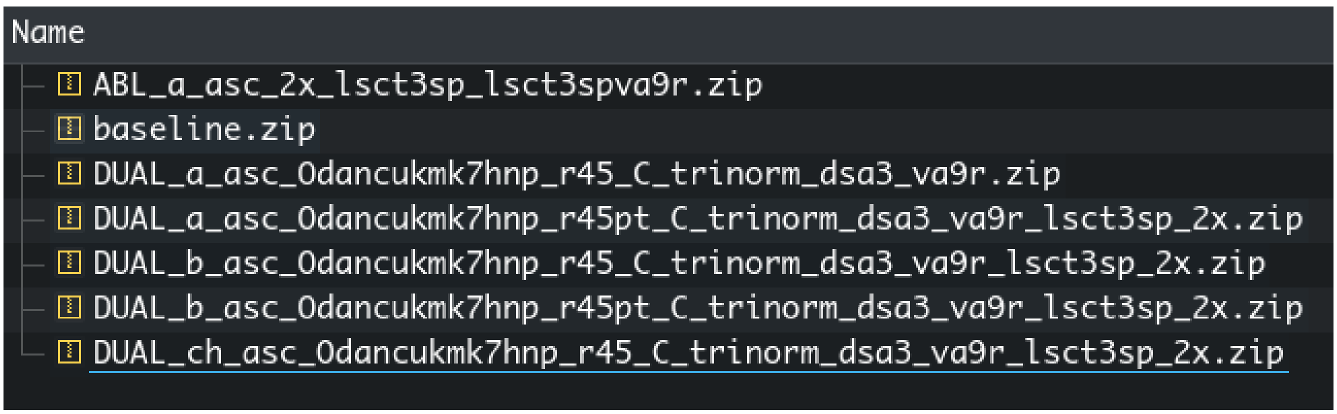
# Handbook

## Setup Environment

1. Make a fresh installation of Manjaro.

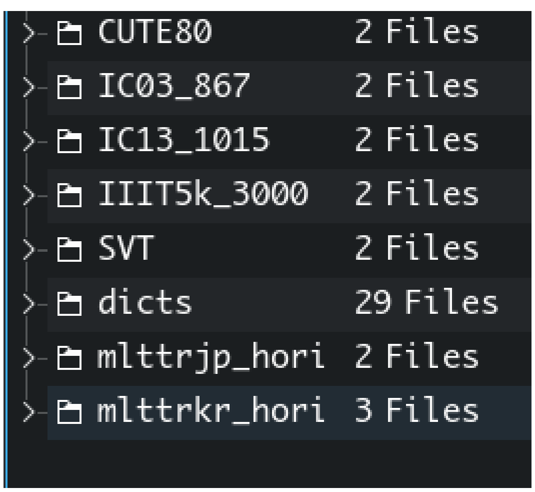
2. Download the repo from Github.

3. a) Download models from the github links, and put them in a folder, say /run/media/xxx/data/c491/c4-models-work/.




b) Run for i in $(ls); do unzip $i; done; to uncompress them

4. Download datasets from the github links and uncompress it to /run/media/xxx/data/xxx/ssddata/




5. Uncompress the Athena.zip file to /run/media/xxx/data/xxx/ssddata/

6. Install dependencies

a) Install packages on paclist.txt in the git repo

b) Install pip packages on the piplist.txt in the git repo

c) Install pylcs-laser.zip

pylcs modded by a Github user, adds support to return the exact LCS

Source: https://github.com/Meteorix/pylcs/issues/4,

7. Change the paths in neko_sdk/root.py to the data and models uncompressed. If you do NOT want intermediate results, return None for find_export_root.

```
sdanmk7_routine_cfg_va9.py  eval_configs.py  dual_lsct32.py  dual_no_lsct_32.py  test.py  root.py
import ...
def find_data_root():
    return "/run/media/xxxx/testing/home/xxxx/c491/ssddata/"
def find_model_root():
    return "/run/media/xxxx/testing/home/xxxx/c491/models-release/"
def find_export_root():
    return "/run/media/xxxx/testing/home/xxxx/c491/results/"
```

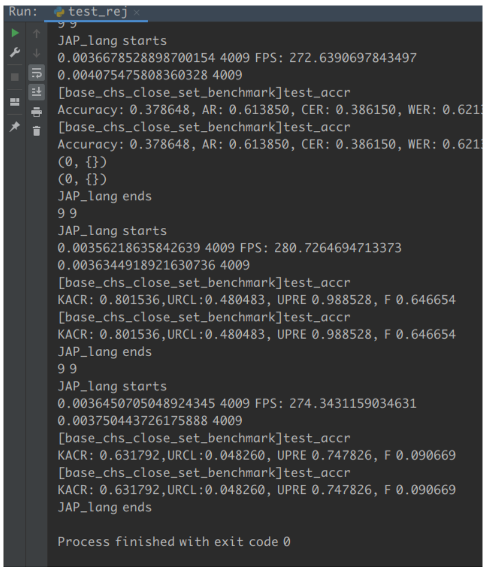
All done! You can now go to the next steps to check out if our framework lives up to its name.

# Evaluation:

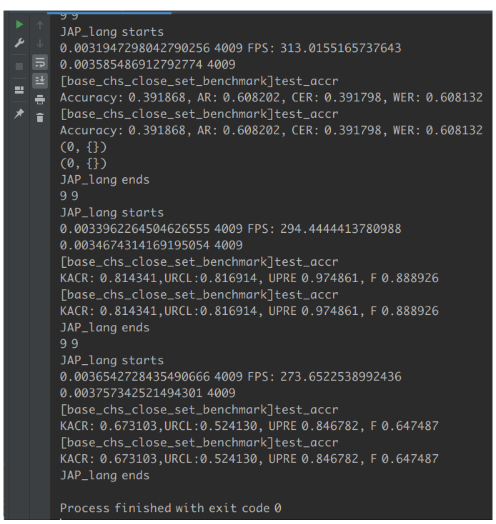
Going ~100 FPS on a laptop single batched is what we reported in the paper, now let's go 280+ FPS on that laptop going multi-batched.  We actually liked the OSOCR way of testing on laptops so we also perform evaluations on laptops consuming around 230w (120w GPU+65W CPU + a little bit extra). This gives an insight into how the model may behave when deployed on a mobile platform that fits into vehicles or other power-consumption-sensitive computing platforms.

## Reproducing Ablative Study

Base model: Run test_rej.py under DUAL_a_asc_Odancukmk7hnp_r45_C_trinorm_dsa3




CIL only: Run test_rej.py under DUAL_a_asc_Odancukmk7hnp_r45_C_trinorm_dsa3_va9r





I think it can run even faster, but nevermind.

ICL only and full model: Run test_rej.py under ABL_a_asc_2x_lsct3sp_lsct3spva9r

GZSL

```
9 9
JAP_lang starts
0.0034094652531301032 4009 FPS: 293.30112664498847
0.003800137972528425 4009
[base_chs_lsctsp_2x_close_set_benchmark]test_accr
Accuracy: 0.407333, AR: 0.612478, CER: 0.387522, WER: 0.592667
[base_chs_lsctsp_2x_close_set_benchmark]test_accr
Accuracy: 0.407333, AR: 0.612478, CER: 0.387522, WER: 0.592667
(0, {})
(0, {})
JAP_lang ends
9 9
JAP_lang starts
0.0034650819026255256 4009 FPS: 288.5934670814824
0.0035795830645505315 4009
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.415565, AR: 0.617967, CER: 0.382033, WER: 0.584435
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.415565, AR: 0.617967, CER: 0.382033, WER: 0.584435
(0, {})
(0, {})
JAP_lang ends
9 9
JAP_lang starts
```

OSR

```
(0, {})
(0, {})
JAP_lang ends
9 9
JAP_lang starts
0.00331193338043525 4009 FPS: 301.93844052158477
0.0033741620568808074 4009
[base_chs_lsctsp_2x_close_set_benchmark]test_accr
KACR: 0.805378,URCL:0.771995, UPRE 0.974580, F 0.861538
[base_chs_lsctsp_2x_close_set_benchmark]test_accr
KACR: 0.805378,URCL:0.771995, UPRE 0.974580, F 0.861538
JAP_lang ends
9 9
JAP_lang starts
0.0034653331074342374 4009 FPS: 288.57254670689036
0.0035378199082772076 4009
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
KACR: 0.827145,URCL:0.870818, UPRE 0.977739, F 0.921186
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
KACR: 0.827145,URCL:0.870818, UPRE 0.977739, F 0.921186
```

GOSR

```
JAP_lang ends
9 9
JAP_lang starts
0.0034190572447002484 4009 FPS: 292.4782852203081
0.0035231323365886836 4009
[base_chs_lsctsp_2x_close_set_benchmark]test_accr
KACR: 0.662775,URCL:0.484287, UPRE 0.829011, F 0.611406
[base_chs_lsctsp_2x_close_set_benchmark]test_accr
KACR: 0.662775,URCL:0.484287, UPRE 0.829011, F 0.611406
JAP_lang ends
9 9
JAP_lang starts
0.0033245114026506156 4009 FPS: 300.7960806519434
0.0034288189065637777 4009
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
KACR: 0.676695,URCL:0.687991, UPRE 0.819519, F 0.748017
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
KACR: 0.676695,URCL:0.687991, UPRE 0.819519, F 0.748017
JAP_lang ends

Process finished with exit code 0
```

# Reproducing Results On English, Japanese, and Korean

Regular model: Run test.py under ABL_a_asc_2x_lsct3sp_lsct3spva9r

```
9 9
CUTE starts
/usr/lib/python3.10/site-packages/torch/utils/data/dataloader.py:
  create. Please be aware that excessive worker creation might get Dat
  warnings.warn(_create_warning_msg(
0.010705261594719358 288 FPS: 93.41200970682262
0.014507010579109192 288
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.729167, AR: 0.870846, CER: 0.129154, WER: 0.270833
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.729167, AR: 0.870846, CER: 0.129154, WER: 0.270833
(0, {})
(0, {})
CUTE ends
IIIT5k starts
0.009958661953608195 3000 FPS: 100.41509639130614
0.009964030663172405 3000
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.832000, AR: 0.936669, CER: 0.063331, WER: 0.168000
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.832000, AR: 0.936669, CER: 0.063331, WER: 0.168000
(0, {})
(0, {})
IIIT5k ends
9 9
JAP_lang starts
0.003196817587426429 4009 FPS: 312.8110918599649
0.0033238392037981375 4009
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.415565, AR: 0.617967, CER: 0.382033, WER: 0.584435
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.415565, AR: 0.617967, CER: 0.382033, WER: 0.584435
(0, {})
(0, {})
JAP_lang ends
9 9
KR_lang starts
Corrupted image for 5171
0.0037015587281636723 5171 FPS: 270.15645932925554
0.0037829781426979526 5171
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.195320, AR: 0.438824, CER: 0.561176, WER: 0.804680
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
Accuracy: 0.195320, AR: 0.438824, CER: 0.561176, WER: 0.804680
(0, {})
(0, {})
KR_lang ends
```

The FPSs are incoherent due to batch size.

Also note the running time includes prototype caching time, hence CUTE(288) has a lower FPS than IIIT5k(3000). In practical use, the FPS will be close to IIIT5k performance.

Large: Run test.py under DUAL_a_asc_Odancukmk7hnp_r45pt_C_trinorm_dsa3_va9r_lsct3sp_2x

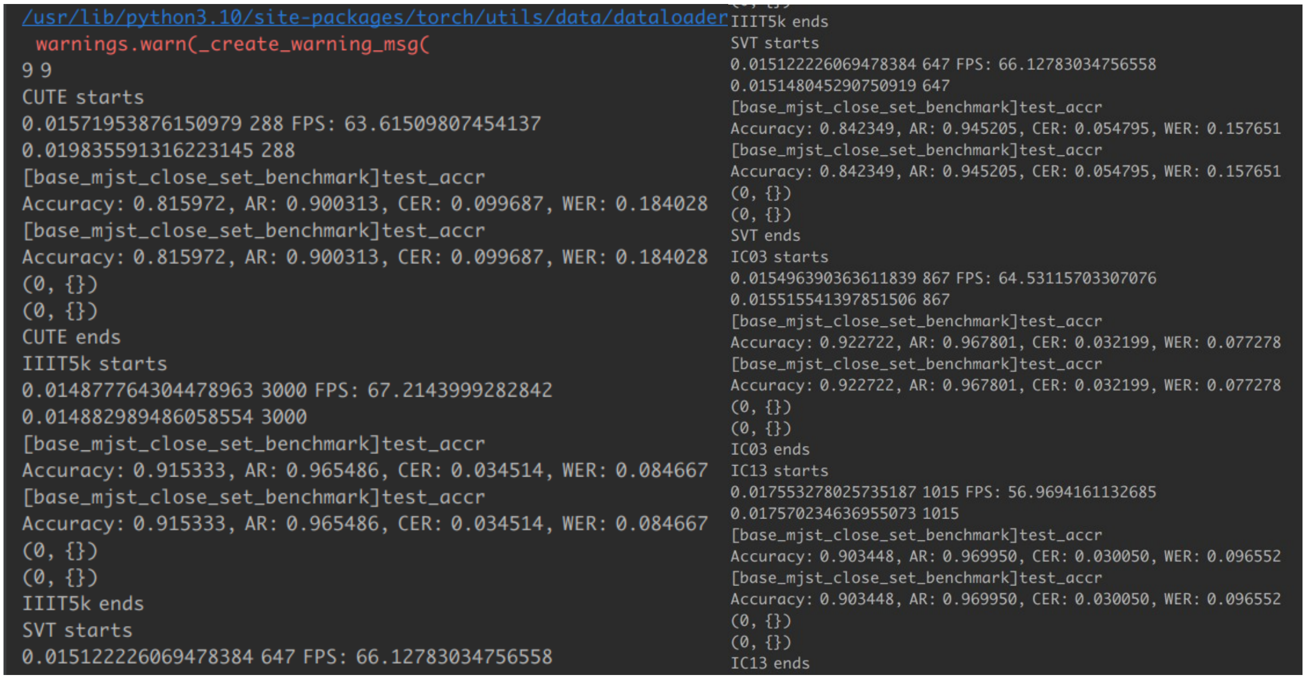
```
Run: test
Accuracy: 0.784722, AR: 0.894671, CER: 0.105329, WER: 0.215278
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.784722, AR: 0.894671, CER: 0.105329, WER: 0.215278
(0, {})
(0, {})
CUTE ends
IIIT5k starts
0.015140331427256267 3000 FPS: 66.04875228819348
0.015145817677179972 3000
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.850667, AR: 0.945118, CER: 0.054882, WER: 0.149333
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.850667, AR: 0.945118, CER: 0.054882, WER: 0.149333
(0, {})
(0, {})
IIIT5k ends
9 9
JAP_lang starts
0.005023633610075268 4009 FPS: 199.0591029557622
0.005359705798671738 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.444749, AR: 0.649319, CER: 0.350681, WER: 0.555251
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.444749, AR: 0.649319, CER: 0.350681, WER: 0.555251
(0, {})
(0, {})
JAP_lang ends
9 9
KR_lang starts
Corrupted image for 5171
0.0051070246791452304 5171 FPS: 195.80872676874773
0.005276509766144348 5171
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.221427, AR: 0.454455, CER: 0.545545, WER: 0.778573
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.221427, AR: 0.454455, CER: 0.545545, WER: 0.778573
(0, {})
(0, {})
KR_lang ends
Git  Find  Run  TODO  Problems  Debug  Python Packages  Termina
```

# Reproducing Close-set Benchmarks

Regular: run test.py under DUAL_b_asc_Odancukmk7hnp_r45_C_trinorm_dsa3_va9r_lsct3sp_2x

```
Run: test (2)
/usr/lib/python3.10/site-packages/torch/utils/data/dataloader.py:487
  warnings.warn(_create_warning_msg(
9 9
CUTE starts
0.012077465653419495 288 FPS: 82.79882789125298
0.015858748720751867 288
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.795139, AR: 0.875235, CER: 0.124765, WER: 0.204861
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.795139, AR: 0.875235, CER: 0.124765, WER: 0.204861
(0, {})
(0, {})
CUTE ends
IIIT5k starts
0.010423674662907919 3000 FPS: 95.93545772859218
0.010429205020268758 3000
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.892333, AR: 0.955465, CER: 0.044535, WER: 0.107667
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.892333, AR: 0.955465, CER: 0.044535, WER: 0.107667
(0, {})
(0, {})
IIIT5k ends
SVT starts
0.010439500558136788 647 FPS: 95.79002313674641
(0, {})
IIIT5k ends
SVT starts
0.010439500558136788 647 FPS: 95.79002313674641
0.010465028297035182 647
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.822257, AR: 0.928346, CER: 0.071654, WER: 0.177743
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.822257, AR: 0.928346, CER: 0.071654, WER: 0.177743
(0, {})
(0, {})
SVT ends
IC03 starts
0.010752115832618089 867 FPS: 93.00495042718535
0.010770864552692558 867
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.925029, AR: 0.967801, CER: 0.032199, WER: 0.074971
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.925029, AR: 0.967801, CER: 0.032199, WER: 0.074971
(0, {})
(0, {})
IC03 ends
IC13 starts
0.010621431425874456 1015 FPS: 94.14926857823879
0.010637677479260074 1015
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.903448, AR: 0.967724, CER: 0.032276, WER: 0.096552
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.903448, AR: 0.967724, CER: 0.032276, WER: 0.096552
```

Large: run test.py under DUAL_b_asc_Odancukmk7hnp_r45pt_C_trinorm_dsa3_va9r_lsct3sp_2x

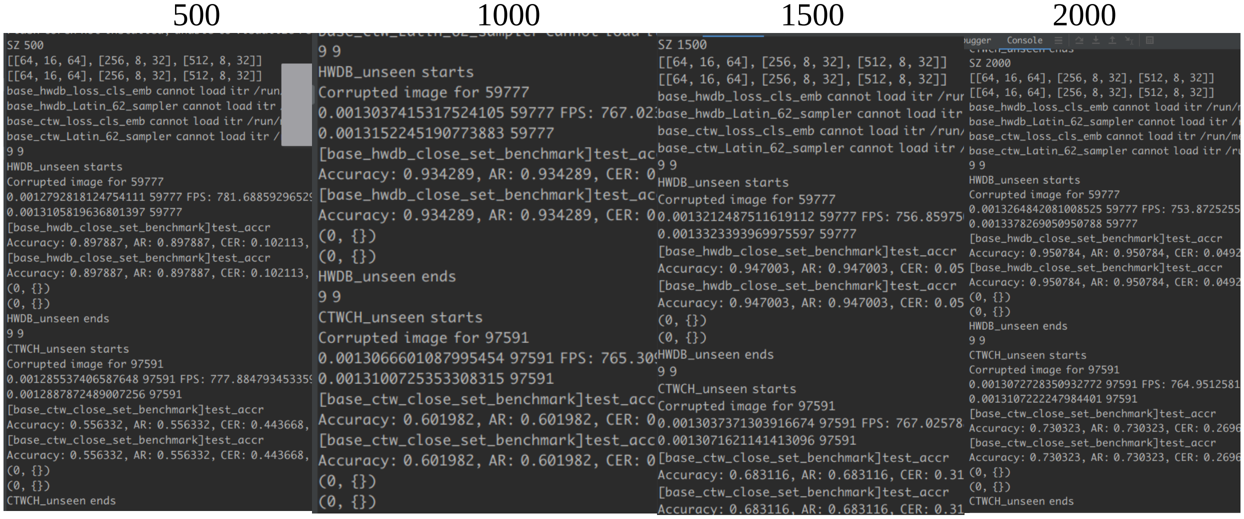



## Reproducing Zero-shot Character Benchmarks

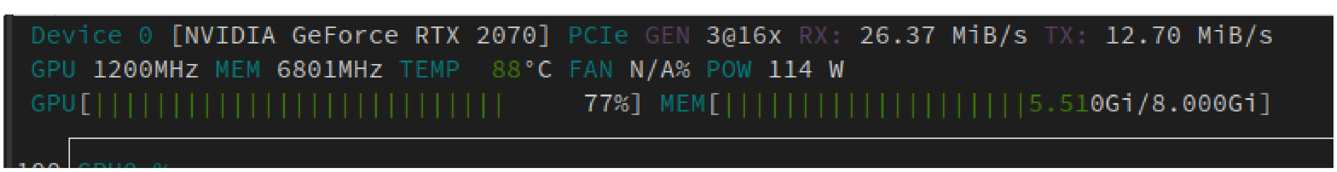
Run test.py under DUAL_ch_asc_Odancukmk7hnp_r45_C_trinorm_dsa3_va9r_lsct3sp_2x

Note the datasets are not uploaded for review due to the size limitation.

500 1000 1500 2000

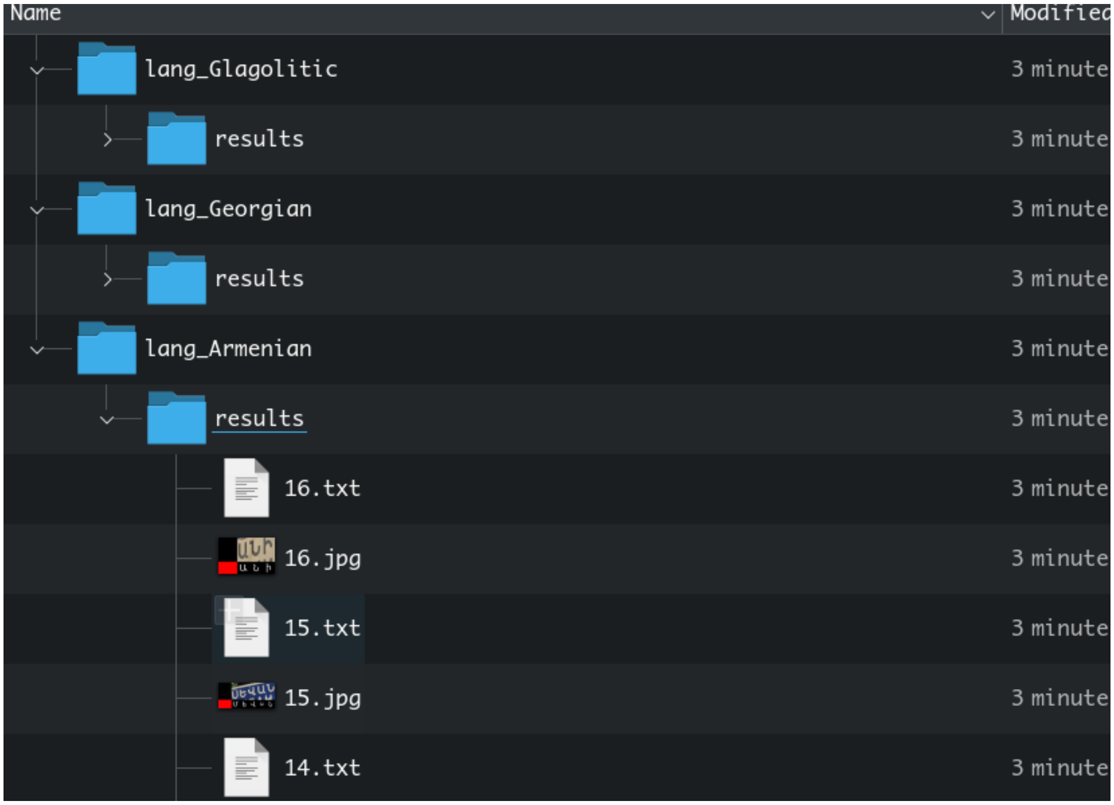



Speed drops after the card overheats, poor little card.



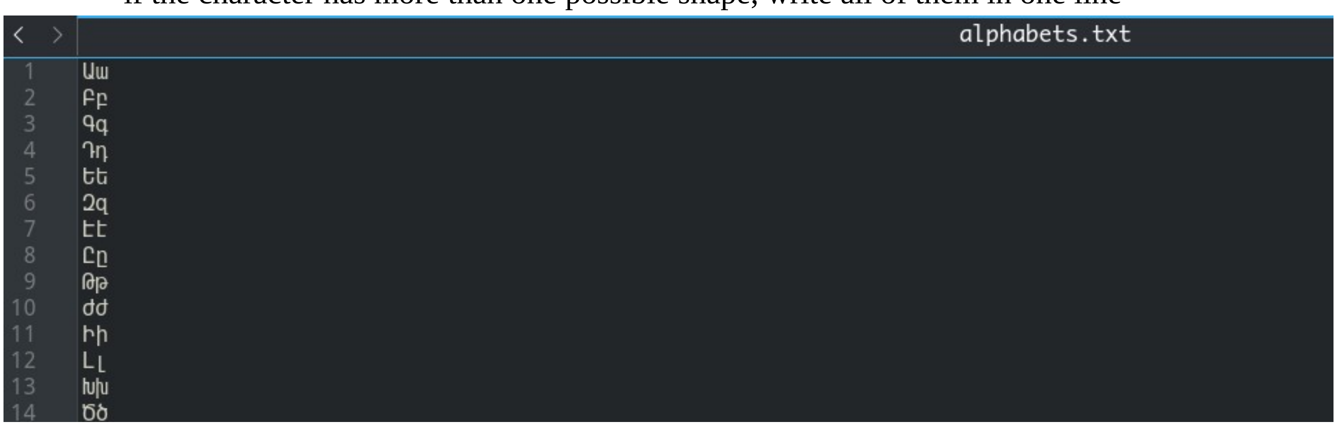
# Recognition of some Greek Family Languages

Run test_athena.py under DUAL_a_asc_Odancukmk7hnp_r45pt_C_trinorm_dsa3_va9r_lsct3sp_2x

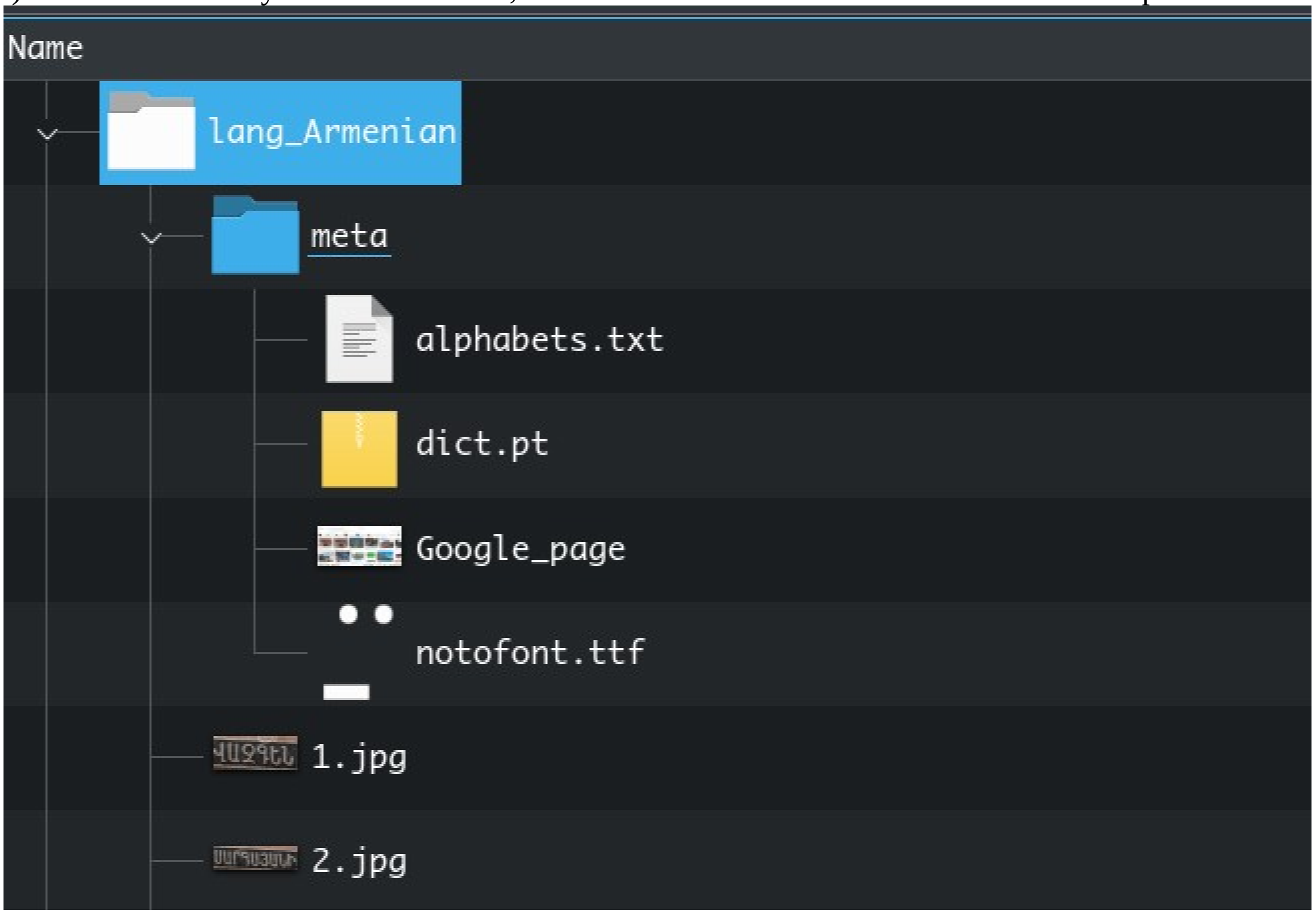



If you have your own language in mind, say xxxx language, you can test it with the following steps:
a) add a folder named lang_xxxx in the Athena folder (the lang_xxxx part is mandatory for the testing script to pick it up)
b) add the images in question to the folder
c) Add a folder name meta (again it has to be named as `meta')
d) Download Noto sans regular font in question to the meta folder and name it as `notofont.ttf'
e) add the  alphabets file, each character per line

if the character has more than one possible shape, write all of them in one line



f) Done! Don’t worry about the dict file, the framework will build it from notofont and alphabets




## System information

```
 Class           Description
===========================
 system          Computer
 bus             Motherboard
 memory          24GiB System memory
 processor       Intel(R) Core(TM) i5-9400 CPU @ 2.90GHz
 bridge          8th Gen Core Processor Host Bridge/DRAM Registers
 bridge          6th-10th Gen Core Processor PCIe Controller (x16)
 display         TU106M [GeForce RTX 2070 Mobile]
 multimedia      TU106 High Definition Audio Controller
 display         CoffeeLake-S GT2 [UHD Graphics 630]
```