

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/`.

```
Name
- ABL_a_asc_2x_lsct3sp_lsct3spva9r.zip
- baseline.zip
- DUAL_a_asc_0dancukmk7hnp_r45_C_trinorm_dsa3_va9r.zip
- DUAL_a_asc_0dancukmk7hnp_r45pt_C_trinorm_dsa3_va9r_lsct3sp_2x.zip
- DUAL_b_asc_0dancukmk7hnp_r45_C_trinorm_dsa3_va9r_lsct3sp_2x.zip
- DUAL_b_asc_0dancukmk7hnp_r45pt_C_trinorm_dsa3_va9r_lsct3sp_2x.zip
- DUAL_ch_asc_0dancukmk7hnp_r45_C_trinorm_dsa3_va9r_lsct3sp_2x.zip
```

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/`

```
> CUTE80      2 Files
> IC03_867    2 Files
> IC13_1015   2 Files
> IIIT5k_3000 2 Files
> SVT         2 Files
> dicts       29 Files
> mlttrjp_hori 2 Files
> mlttrkr_hori 3 Files
```

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

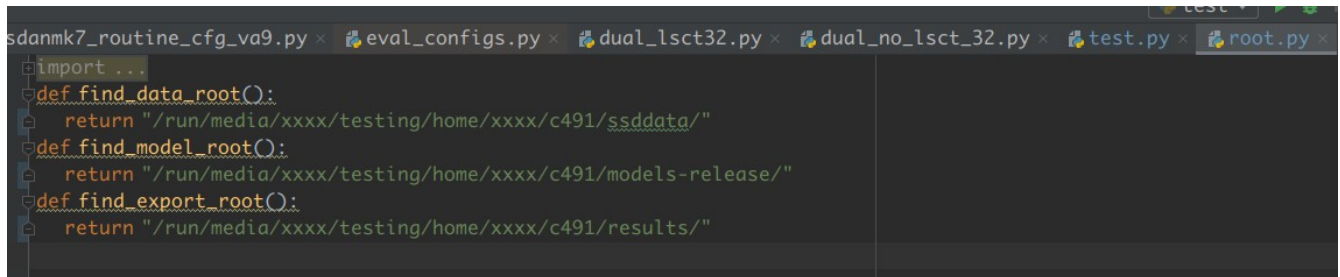
6. Install dependencies

- 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 x eval_configs.py x dual_lsct32.py x dual_no_lsct_32.py x test.py x root.py x
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

```
Run: test_rej
JAP_lang starts
0.0036678528898700154 4009 FPS: 272.6390697843497
0.004075475808360328 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.378648, AR: 0.613850, CER: 0.386150, WER: 0.621
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.378648, AR: 0.613850, CER: 0.386150, WER: 0.621
(0, {})
(0, {})
JAP_lang ends
9 9
JAP_lang starts
0.00356218635842639 4009 FPS: 280.7264694713373
0.0036344918921630736 4009
[base_chs_close_set_benchmark]test_accr
KACR: 0.801536, URCL: 0.480483, UPRE 0.988528, F 0.646654
[base_chs_close_set_benchmark]test_accr
KACR: 0.801536, URCL: 0.480483, UPRE 0.988528, F 0.646654
JAP_lang ends
9 9
JAP_lang starts
0.0036450705048924345 4009 FPS: 274.3431159034631
0.003750443726175888 4009
[base_chs_close_set_benchmark]test_accr
KACR: 0.631792, URCL: 0.048260, UPRE 0.747826, F 0.090669
[base_chs_close_set_benchmark]test_accr
KACR: 0.631792, URCL: 0.048260, UPRE 0.747826, F 0.090669
JAP_lang ends
Process finished with exit code 0
```

CIL only: Run test_rej.py under DUAL_a_asc_Odancukmk7hnp_r45_C_trinorm_dsa3_va9r

```
JAP_lang starts
0.0031947298042790256 4009 FPS: 313.0155165737643
0.003585486912792774 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.391868, AR: 0.608202, CER: 0.391798, WER: 0.608132
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.391868, AR: 0.608202, CER: 0.391798, WER: 0.608132
(0, {})
(0, {})
JAP_lang ends
9 9
JAP_lang starts
0.0033962264504626555 4009 FPS: 294.4444413780988
0.0034674314169195054 4009
[base_chs_close_set_benchmark]test_accr
KACR: 0.814341, URCL: 0.816914, UPRE 0.974861, F 0.888926
[base_chs_close_set_benchmark]test_accr
KACR: 0.814341, URCL: 0.816914, UPRE 0.974861, F 0.888926
JAP_lang ends
9 9
JAP_lang starts
0.0036542728435490666 4009 FPS: 273.6522538992436
0.003757342521494301 4009
[base_chs_close_set_benchmark]test_accr
KACR: 0.673103, URCL: 0.524130, UPRE 0.846782, F 0.647487
[base_chs_close_set_benchmark]test_accr
KACR: 0.673103, URCL: 0.524130, UPRE 0.846782, F 0.647487
JAP_lang ends
Process finished with exit code 0
```



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

OSR

GOSR

```
9 9
JAP_lang starts
0.0034094652531301832 4009 FPS: 293.30112664498847
0.00330013972528445 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
(0, {})
(0, {})
JAP_lang ends
9 9
JAP_lang starts
0.00331183338043525 4009 FPS: 301.93844052158477
0.0033741620568808074 4009
[base_chs_lsctsp_2x_close_set_benchmark]test_accr
KACR: 0.805378, URLC: 0.771995, UPRE 0.974580, F 0.861538
[base_chs_lsctsp_2x_close_set_benchmark]test_accr
KACR: 0.805378, URLC: 0.771995, UPRE 0.974580, F 0.861538
JAP_lang ends
9 9
JAP_lang starts
0.0034653331074342374 4009 FPS: 288.57254670689036
0.0035378199082722076 4009
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
KACR: 0.827145, URLC: 0.870818, UPRE 0.977739, F 0.921186
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
KACR: 0.827145, URLC: 0.870818, UPRE 0.977739, F 0.921186
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, URLC: 0.687991, UPRE 0.819519, F 0.748017
[base_chs_lsctsp_2x_va9r_close_set_benchmark]test_accr
KACR: 0.676695, URLC: 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 Data
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
```

Reproducing Close-set Benchmarks

Regular: run test.py under DUAL_b_asc_Odancukmk7hnp_r45_C_trinorm_dsa3_va9r_lsct3sp_2x

```
Run: test (2) x
/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
0.010465028297035182 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
```

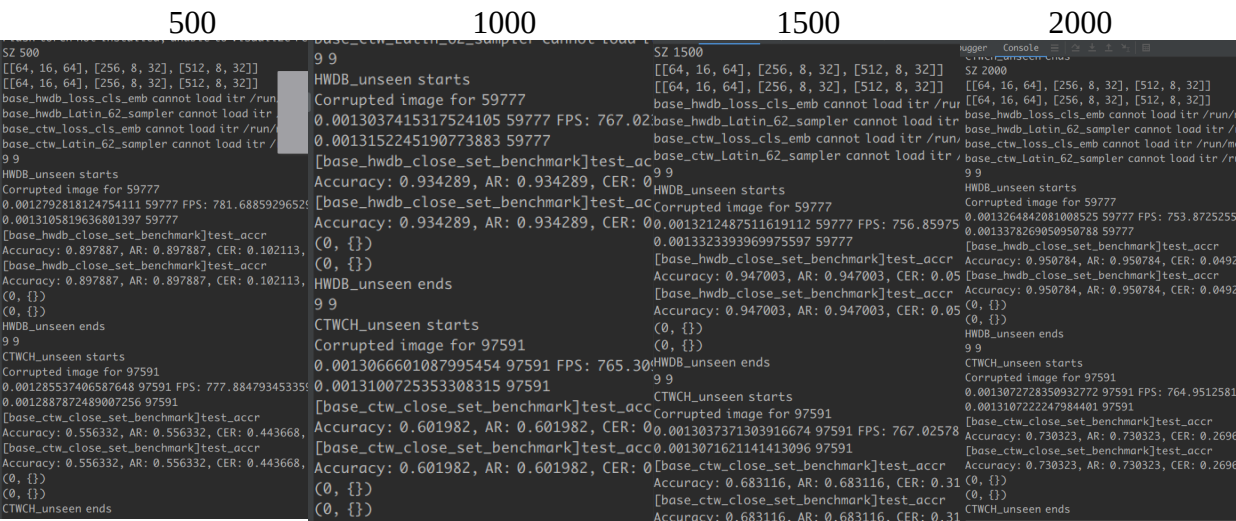

Large: run test.py under DUAL_b_asc_Odancukmk7hnp_r45pt_C_trinorm_dsa3_va9r_lsct3sp_2x

```
/usr/lib/python3.10/site-packages/torch/utils/data/dataloader.py:155: UserWarning: The dataloader is running on a GPU device but the data is on the CPU. This is inefficient. Consider moving the data to the GPU device.
warnings.warn(_create_warning_msg(
9 9
CUTE starts
0.01571953876150979 288 FPS: 63.61509807454137
0.019835591316223145 288
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.815972, AR: 0.900313, CER: 0.099687, WER: 0.184028
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.815972, AR: 0.900313, CER: 0.099687, WER: 0.184028
(0, {})
(0, {})
CUTE ends
IIIT5k starts
0.014877764304478963 3000 FPS: 67.2143999282842
0.014882989486058554 3000
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.915333, AR: 0.965486, CER: 0.034514, WER: 0.084667
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.915333, AR: 0.965486, CER: 0.034514, WER: 0.084667
(0, {})
(0, {})
IIIT5k ends
SVT starts
0.015122226069478384 647 FPS: 66.12783034756558
0.015148045290750919 647
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.842349, AR: 0.945205, CER: 0.054795, WER: 0.157651
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.842349, AR: 0.945205, CER: 0.054795, WER: 0.157651
(0, {})
(0, {})
SVT ends
IC03 starts
0.015496390363611839 867 FPS: 64.53115703307076
0.015515541397851506 867
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.922722, AR: 0.967801, CER: 0.032199, WER: 0.077278
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.922722, AR: 0.967801, CER: 0.032199, WER: 0.077278
(0, {})
(0, {})
IC03 ends
IC13 starts
0.017553278025735187 1015 FPS: 56.9694161132685
0.017570234636955073 1015
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.903448, AR: 0.969950, CER: 0.030050, WER: 0.096552
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.903448, AR: 0.969950, CER: 0.030050, WER: 0.096552
(0, {})
(0, {})
IC13 ends
```

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.



Speed drops after the card overheats, poor little card.

```
Device 0 [NVIDIA GeForce RTX 2070] PCIe GEN 3@16x RX: 26.37 MiB/s TX: 12.70 MiB/s
GPU 1200MHz MEM 6801MHz TEMP 88°C FAN N/A% POW 114 W
GPU[||||||||||||||||||||||||| 77%] MEM[|||||||||||||||||||||5.510Gi/8.000Gi]
```

Recognition of some Greek Family Languages

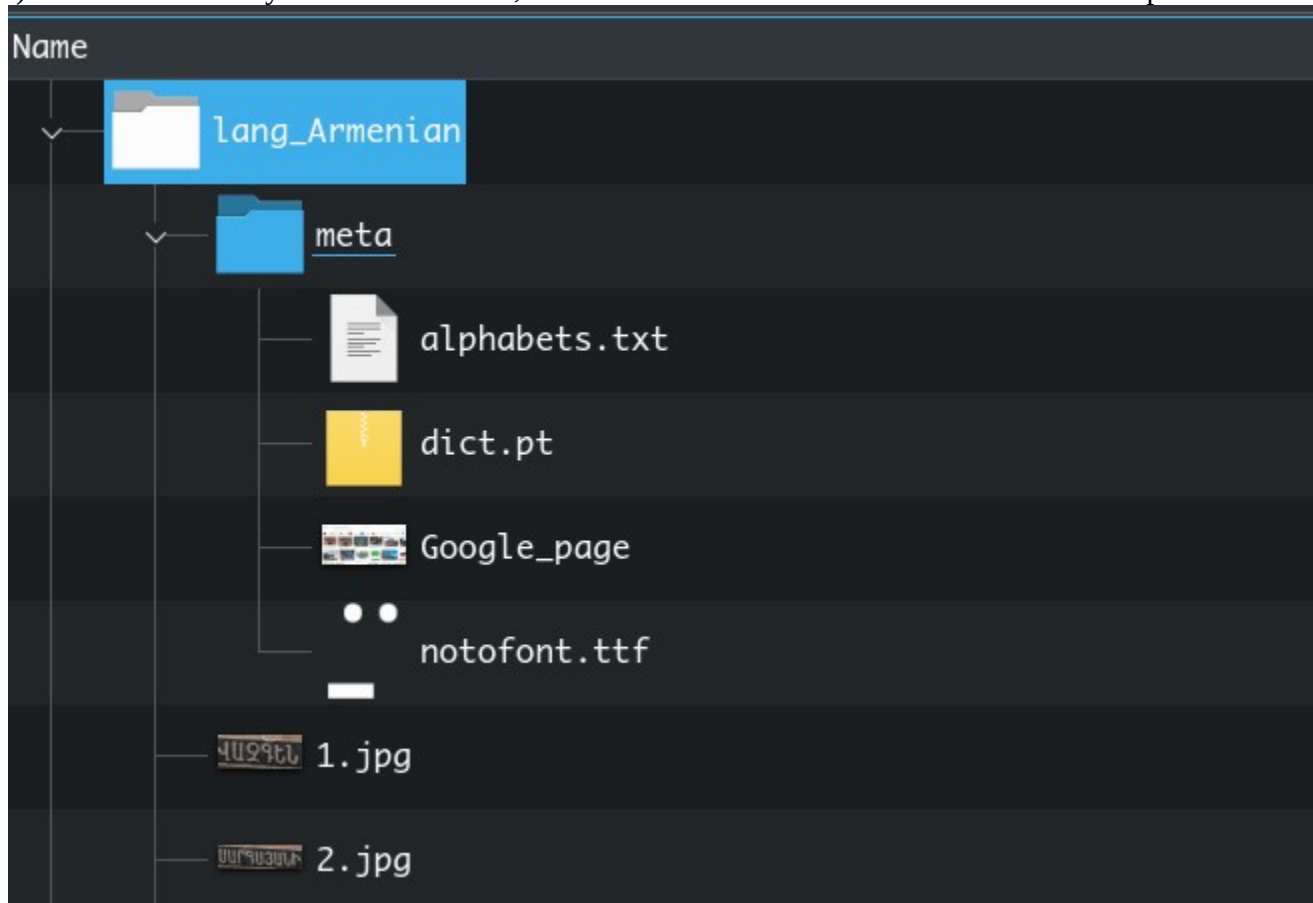
Run test_athena.py under DUAL_a_asc_Odancukmk7hnp_r45pt_C_trinorm_dsa3_va9r_lsct3sp_2x

Name	Modified
lang_Glagolitic	3 minute
results	3 minute
lang_Georgian	3 minute
results	3 minute
lang_Armenian	3 minute
results	3 minute
16.txt	3 minute
16.jpg	3 minute
15.txt	3 minute
15.jpg	3 minute
14.txt	3 minute

- 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

<	>	alphabets.txt
1		Աա
2		Բբ
3		Գգ
4		Դդ
5		Եե
6		Զզ
7		Էէ
8		Ըը
9		Թթ
10		Ճճ
11		Իի
12		Լլ
13		Խխ
14		Օօ

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]