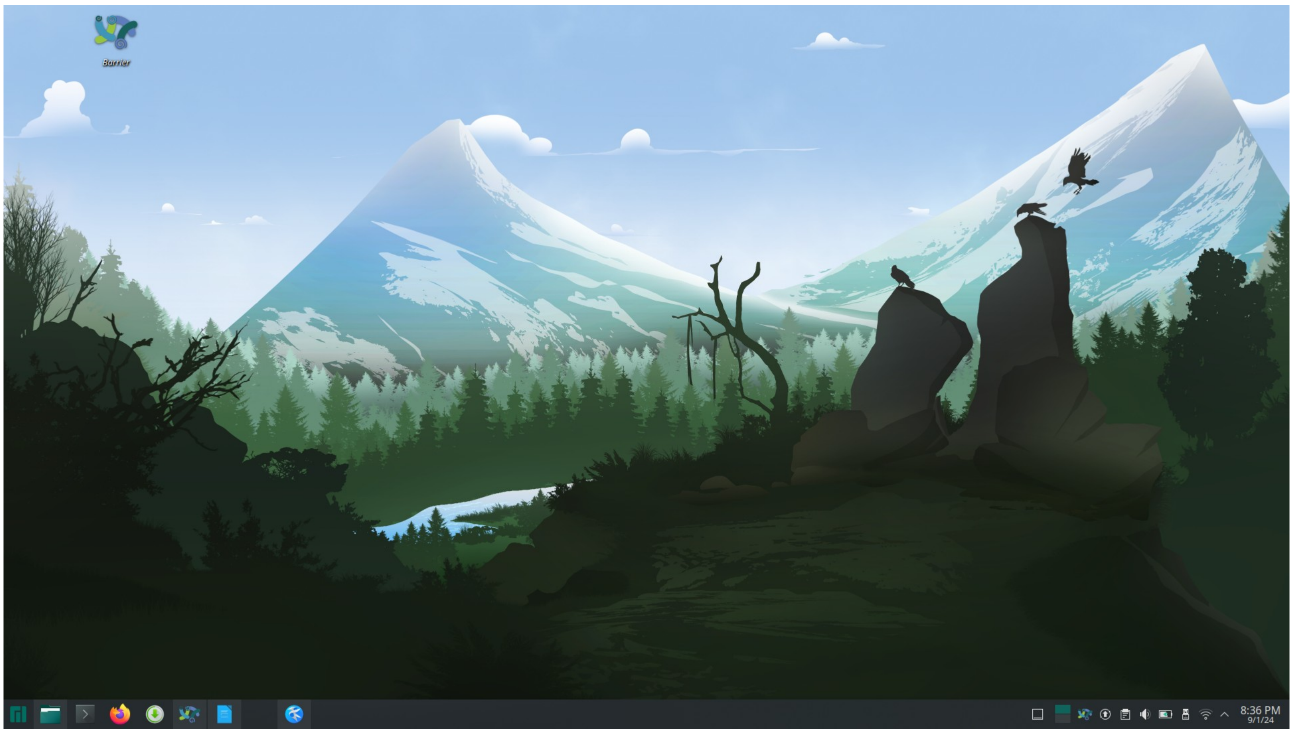
Manual

Setup your Environment:

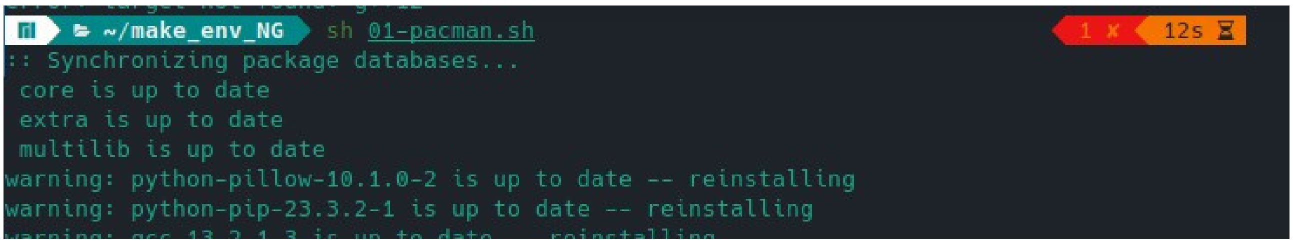
0. Fresh install Manjaro, set your mirrorlist straight, and grab some snacks



1. Download the environment making scripts

https://github.com/lancercat/make_envNG

2. execute 01-pacman.sh and give password for once or twice.



```
 ~/make_env_NG  sh 01-pacman.sh                                          1 ✘  12s ⌛
:: Synchronizing package databases...
 core is up to date
 extra is up to date
 multilib is up to date
warning: python-pillow-10.1.0-2 is up to date -- reinstalling
warning: python-pip-23.3.2-1 is up to date -- reinstalling
```

3 Reboot.

4. execute 02-pip.sh

5. Enjoy,

Code, model and data

Get the data:

Follow the data section here to get your data:

https://github.com/lancercat/VSDF/blob/master/manul.pdf

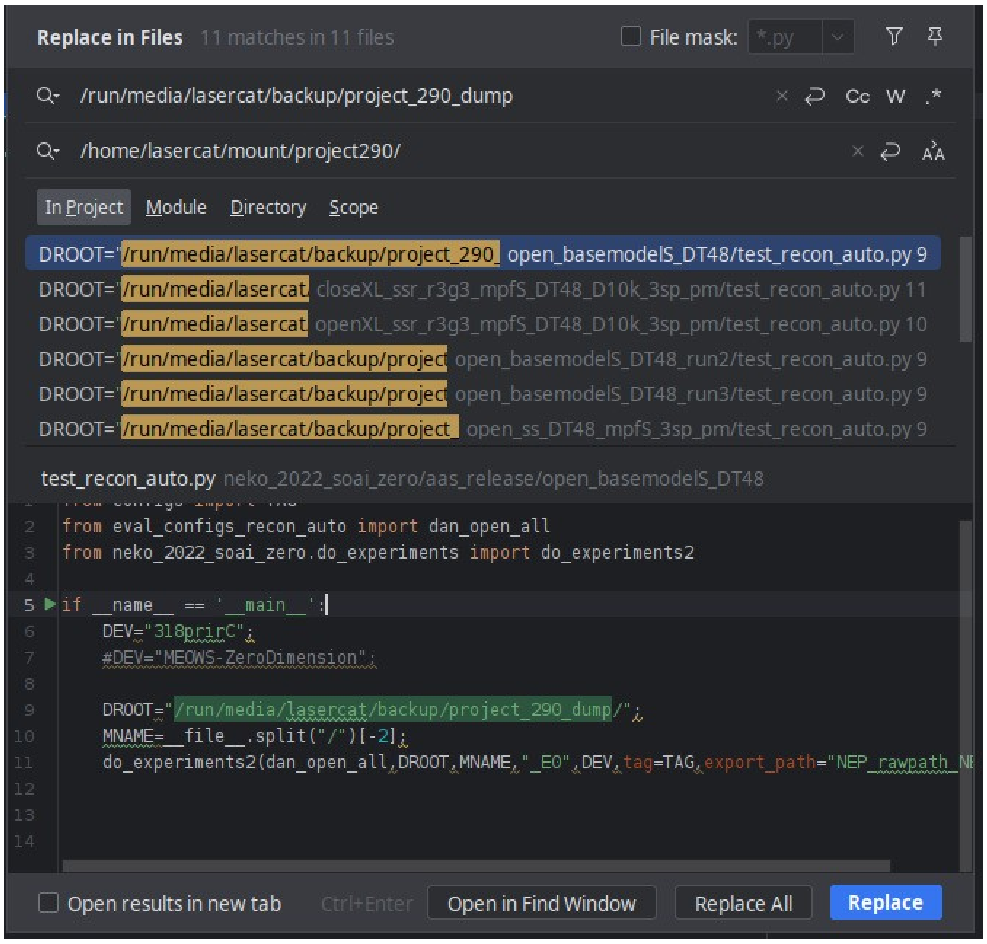
Download the code:

git clone https://github.com/lancercat/OAPR.git

Download the model:

https://drive.google.com/file/d/1Zw9ylDvOuJ7IP6vM1l8kvr94kbGOpahU/view?usp=sharing

and replace the DROOT with your model path.



```
/run/media/lasercat/backup/project_290_dump
/home/lasercat/mount/project290/
```

```
DROOT="/run/media/lasercat/backup/project_290_ open_basemodelS_DT48/test_recon_auto.py 9
DROOT="/run/media/lasercat/ closeXL_ssr_r3g3_mpfS_DT48_D10k_3sp_pm/test_recon_auto.py 11
DROOT="/run/media/lasercat openXL_ssr_r3g3_mpfS_DT48_D10k_3sp_pm/test_recon_auto.py 10
DROOT="/run/media/lasercat/backup/project open_basemodelS_DT48_run2/test_recon_auto.py 9
DROOT="/run/media/lasercat/backup/project open_basemodelS_DT48_run3/test_recon_auto.py 9
DROOT="/run/media/lasercat/backup/project_ open_ss_DT48_mpfS_3sp_pm/test_recon_auto.py 9
```

test_recon_auto.py neko_2022_soai_zero/aas_release/open_basemodelS_DT48

```python
from eval_configs_recon_auto import dan_open_all
from neko_2022_soai_zero.do_experiments import do_experiments2

if __name__ == '__main__':
    DEV="318prirC";
    #DEV="MEOWS-ZeroDimension";

    DROOT="/run/media/lasercat/backup/project_290_dump/";
    MNAME=__file__.split("/")[-2];
    do_experiments2(dan_open_all,DROOT,MNAME,"_E0",DEV,tag=TAG,export_path="NEP_rawpath_NE
```

Ablative Experiments

Base model:

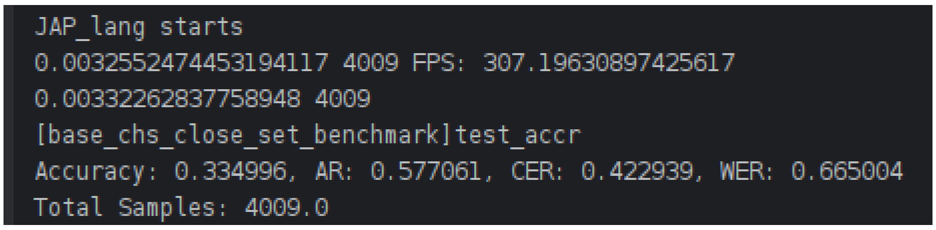
Run open_basemodelS_DT48/test_recon_auto.py

Run open_basemodelS_DT48_run2/test_recon_auto.py

Run open_basemodelS_DT48_run3/test_recon_auto.py

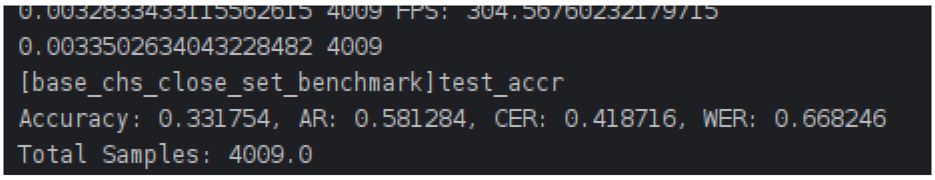
And collect the base_chs_GZSL-CHS-JP-KR_JAP section (the name got messy but nvm)

Run1: Accuracy: 0.334996



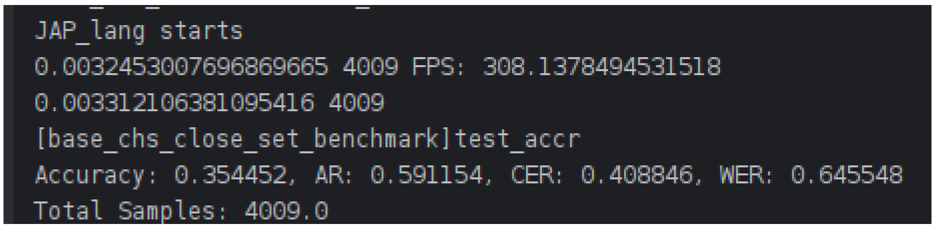
```
JAP_lang starts
0.0032552474453194117 4009 FPS: 307.19630897425617
0.00332262837758948 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.334996, AR: 0.577061, CER: 0.422939, WER: 0.665004
Total Samples: 4009.0
```

Run2: Accuracy: 0.331754

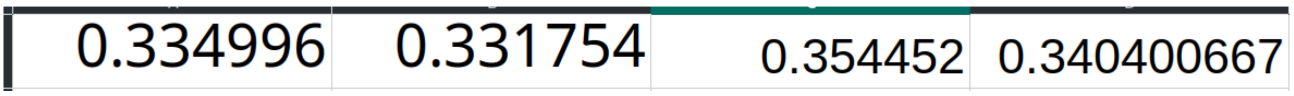


```
0.0032833433115562615 4009 FPS: 304.56760232179715
0.0033502634043228482 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.331754, AR: 0.581284, CER: 0.418716, WER: 0.668246
Total Samples: 4009.0
```

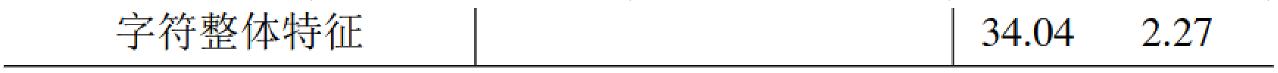
Run3: 0.354452



```
JAP_lang starts
0.0032453007696869665 4009 FPS: 308.1378494531518
0.0033121063810954016 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.354452, AR: 0.591154, CER: 0.408846, WER: 0.645548
Total Samples: 4009.0
```



0.334996	0.331754	0.354452	0.340400667



字符整体特征		34.04	2.27

See, I shit you not.

Hella fast, eh? bcs it has a small backbone, and a faster device, when you batch it and run on SOTA hardwares (1240P + 4060ti on TB4), boom.

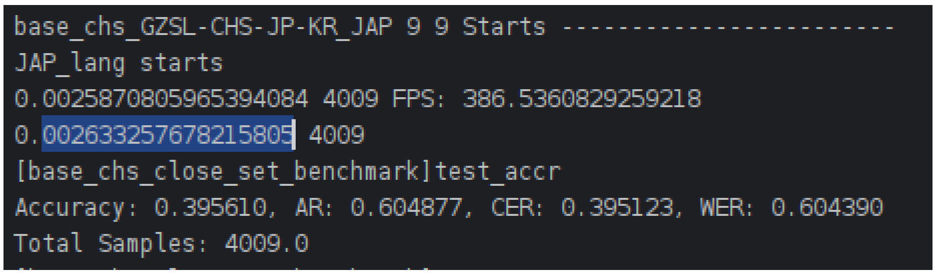
Part representation only:

Run open_ss_DT48_mpfS_3sp_pm/test_recon_auto.py

Run open_ss_DT48_mpfS_3sp_pm/test_recon_auto.py

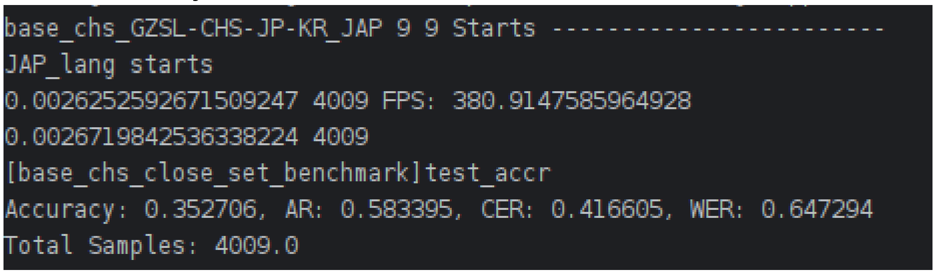
Run open_ss_DT48_mpfS_3sp_pm/test_recon_auto.py

Run1: Accuracy: 0.395610



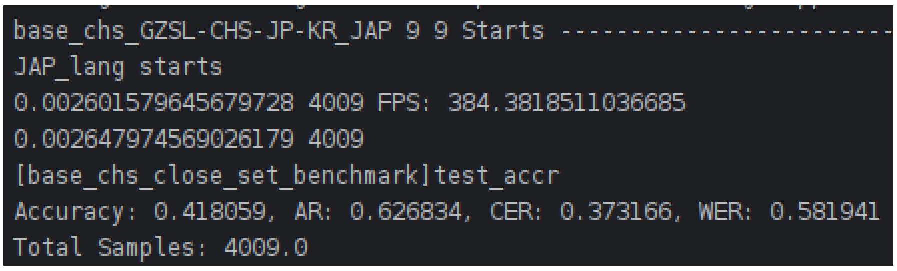
```
base_chs_GZSL-CHS-JP-KR_JAP 9 9 Starts -----------------------
JAP_lang starts
0.0025870805965394084 4009 FPS: 386.5360829259218
0.002633257678215805 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.395610, AR: 0.604877, CER: 0.395123, WER: 0.604390
Total Samples: 4009.0
```

Run2: Accuracy: 0.352706

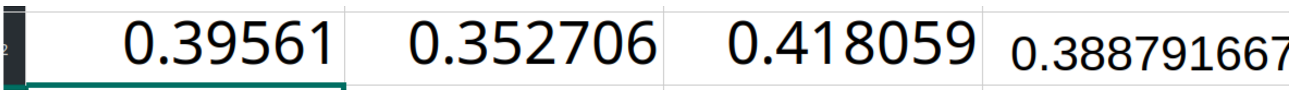


```
base_chs_GZSL-CHS-JP-KR_JAP 9 9 Starts -----------------------
JAP_lang starts
0.0026252592671509247 4009 FPS: 380.9147585964928
0.0026719842536338224 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.352706, AR: 0.583395, CER: 0.416605, WER: 0.647294
Total Samples: 4009.0
```

Run3 Accuracy: 0.418059



```
base_chs_GZSL-CHS-JP-KR_JAP 9 9 Starts -----------------------
JAP_lang starts
0.002601579645679728 4009 FPS: 384.3818511036685
0.002647974569026179 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.418059, AR: 0.626834, CER: 0.373166, WER: 0.581941
Total Samples: 4009.0
```



0.39561	0.352706	0.418059	0.388791667

仅自适应字符部件表示	✓	38.91	6.54

A 0.02 difference, marginable, should be caused by how different softwares process float numbers.

See, it's ~80fps FASTER the the prev one! Again I shit you not on the faster claim.

Any one haz a 4090? Tell me if this goes 1000FPS!

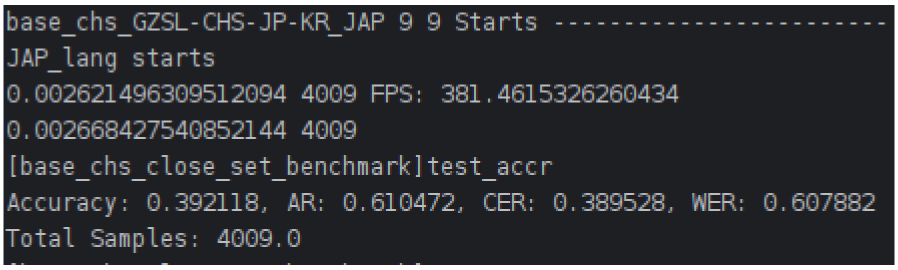
Full model:

Run open_ssr_r3g3_mpfS_DT48_D10k_3sp_pm/test_recon_auto.py

Run open_ssr_r3g3_mpfS_DT48_D10k_3sp_pm/test_recon_auto.py

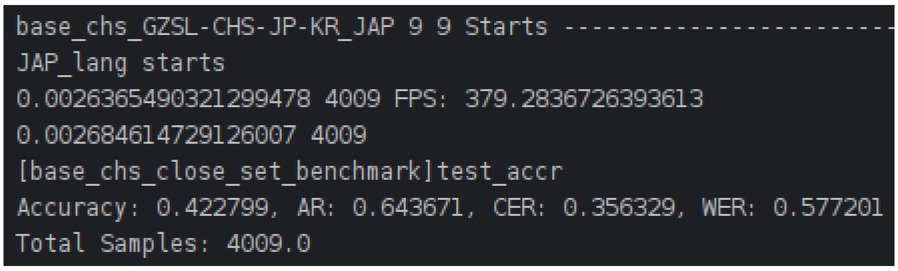
Run open_ssr_r3g3_mpfS_DT48_D10k_3sp_pm/test_recon_auto.py

Run1: 0.392118



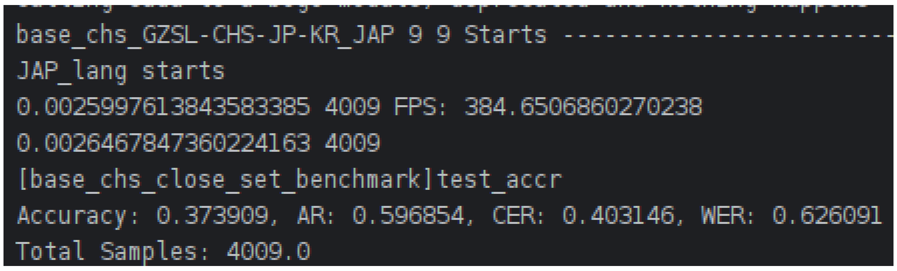
```
base_chs_GZSL-CHS-JP-KR_JAP 9 9 Starts -----------------------
JAP_lang starts
0.002621496309512094 4009 FPS: 381.4615326260434
0.002668427540852144 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.392118, AR: 0.610472, CER: 0.389528, WER: 0.607882
Total Samples: 4009.0
```

Run2: 0.422799

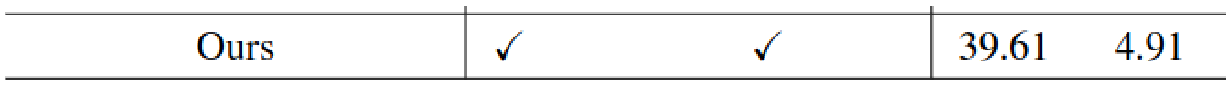


```
base_chs_GZSL-CHS-JP-KR_JAP 9 9 Starts -----------------------
JAP_lang starts
0.0026365490321299478 4009 FPS: 379.2836726393613
0.002684614729126007 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.422799, AR: 0.643671, CER: 0.356329, WER: 0.577201
Total Samples: 4009.0
```

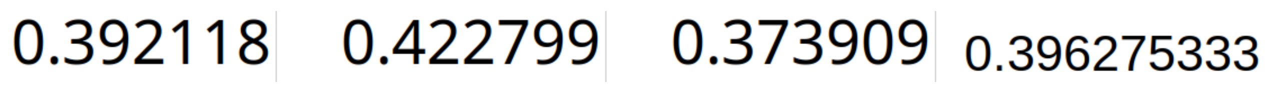
Run3: 0.373909



```
base_chs_GZSL-CHS-JP-KR_JAP 9 9 Starts -----------------------
JAP_lang starts
0.0025997613843583385 4009 FPS: 384.6506860270238
0.0026467847360224163 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.373909, AR: 0.596854, CER: 0.403146, WER: 0.626091
Total Samples: 4009.0
```



Ours	✓	✓	39.61	4.91



0.392118	0.422799	0.373909	0.396275333

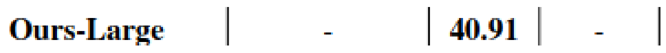
A 0.01 difference, marginable, should be caused by how different softwares process float numbers.

High five! The ablative reproduced!

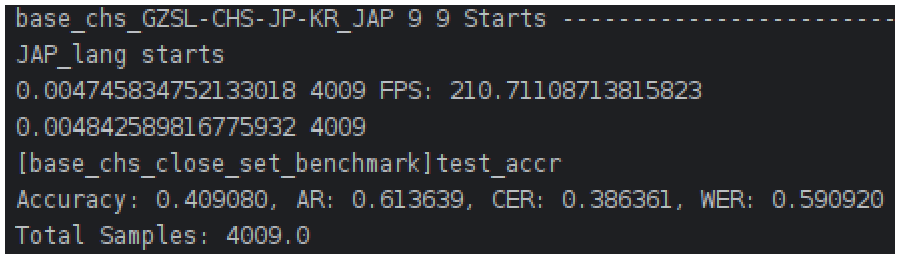
Open-Set benchmark

Large model:Run openXL_ssr_r3g3_mpfS_DT48_D10k_3sp_pm/test_recon_auto.py

GZSL

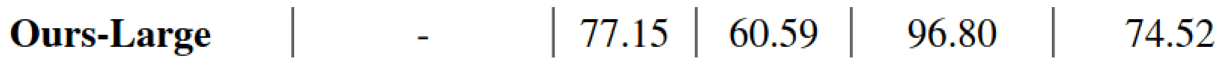


Ours-Large	-	**40.91**	-

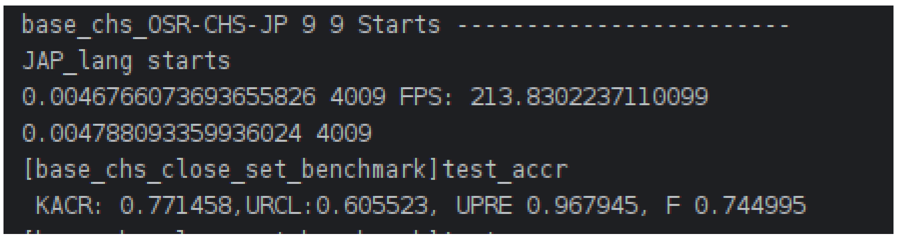


```
base_chs_GZSL-CHS-JP-KR_JAP 9 9 Starts ------------------------
JAP_lang starts
0.004745834752133018 4009 FPS: 210.71108713815823
0.004842589816775932 4009
[base_chs_close_set_benchmark]test_accr
Accuracy: 0.409080, AR: 0.613639, CER: 0.386361, WER: 0.590920
Total Samples: 4009.0
```

OSR

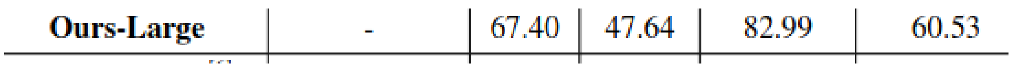


Ours-Large	-	77.15	60.59	96.80	74.52

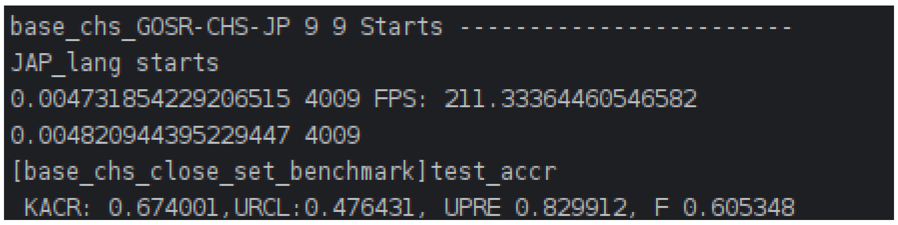


```
base_chs_OSR-CHS-JP 9 9 Starts ------------------------
JAP_lang starts
0.0046766073693655826 4009 FPS: 213.8302237110099
0.004788093359936024 4009
[base_chs_close_set_benchmark]test_accr
 KACR: 0.771458,URCL:0.605523, UPRE 0.967945, F 0.744995
```

GOSR

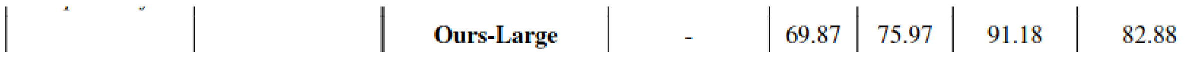


Ours-Large	-	67.40	47.64	82.99	60.53

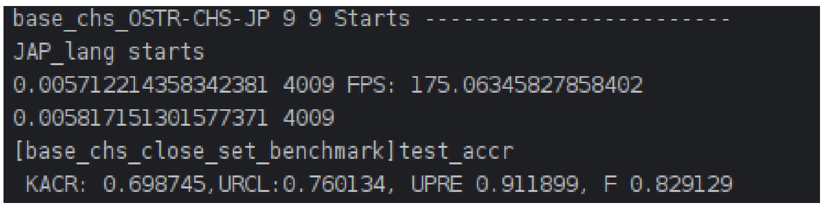


```
base_chs_GOSR-CHS-JP 9 9 Starts ------------------------
JAP_lang starts
0.004731854229206515 4009 FPS: 211.33364460546582
0.004820944395229447 4009
[base_chs_close_set_benchmark]test_accr
 KACR: 0.674001,URCL:0.476431, UPRE 0.829912, F 0.605348
```

OSTR



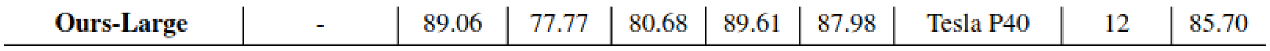
Ours-Large	-	69.87	75.97	91.18	82.88



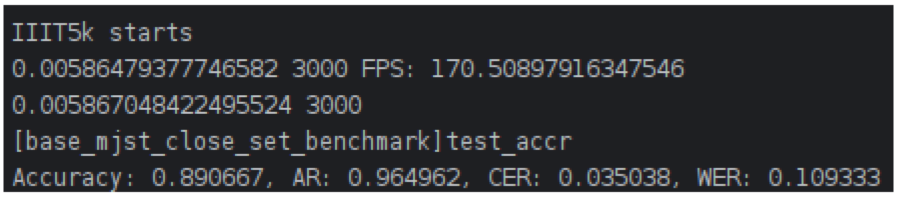
```
base_chs_OSTR-CHS-JP 9 9 Starts ------------------------
JAP_lang starts
0.005712214358342381 4009 FPS: 175.06345827858402
0.005817151301577371 4009
[base_chs_close_set_benchmark]test_accr
 KACR: 0.698745,URCL:0.760134, UPRE 0.911899, F 0.829129
```

Close-Set benchmark

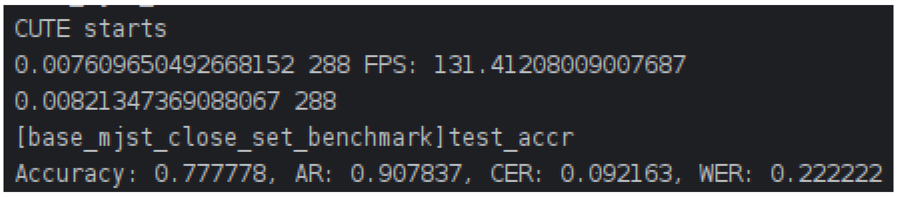
Run closeXL_ssr_r3g3_mpfS_DT48_D10k_3sp_pm/speedbench.py



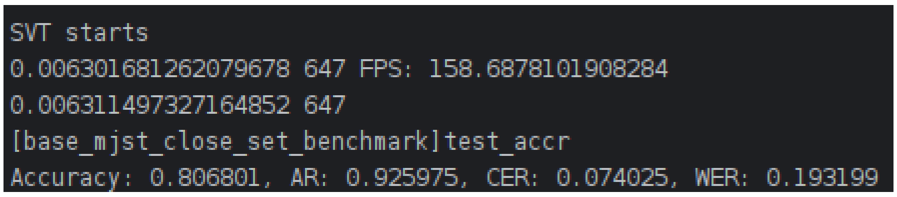
Ours-Large	-	89.06	77.77	80.68	89.61	87.98	Tesla P40	12	85.70



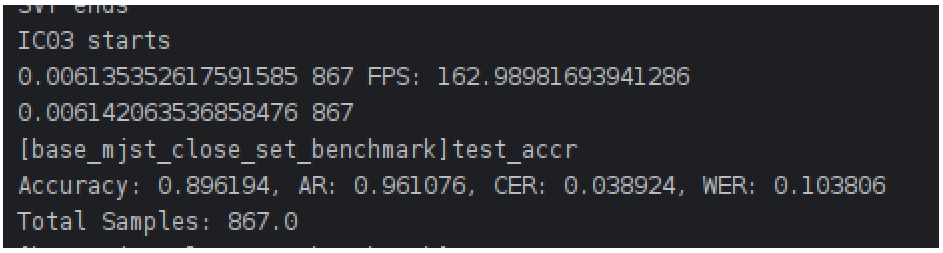
```
IIIT5k starts
0.005864793777746582 3000 FPS: 170.50897916347546
0.005867048422495524 3000
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.890667, AR: 0.964962, CER: 0.035038, WER: 0.109333
```



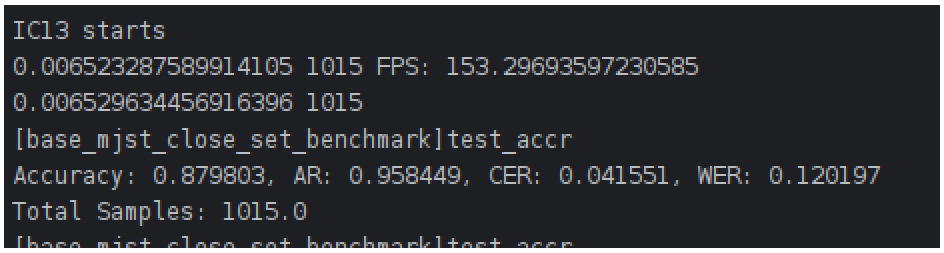
```
CUTE starts
0.007609650492668152 288 FPS: 131.41208009007687
0.00821347369088067 288
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.777778, AR: 0.907837, CER: 0.092163, WER: 0.222222
```



```
SVT starts
0.006301681262079678 647 FPS: 158.6878101908284
0.006311497327164852 647
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.806801, AR: 0.925975, CER: 0.074025, WER: 0.193199
```



```
IC03 starts
0.006135352617591585 867 FPS: 162.98981693941286
0.006142063536858476 867
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.896194, AR: 0.961076, CER: 0.038924, WER: 0.103806
Total Samples: 867.0
```



```
IC13 starts
0.006523287589914105 1015 FPS: 153.29693597230585
0.006529634456916396 1015
[base_mjst_close_set_benchmark]test_accr
Accuracy: 0.879803, AR: 0.958449, CER: 0.041551, WER: 0.120197
Total Samples: 1015.0
```

It's much faster on 4060ti due to its high freq.

The NG framework will likely come this year.

See you in the next manual