

Browse experiment results ...

```
In [ ]:
import torch
import pickle
import glob
import os.path
import pandas as pd
from train_engine import __training_cfg, __exp_res_meta
import numpy as np
# from core.utils_ipynb import read_exp_result_files
import os

def read_exp_result_files(path):
    files = glob.glob(os.path.join(path, "*.pickle"))
    res = []
    for f in files:
        if os.path.basename(f) == 'errors.pickle':
            continue

        r = pickle.load(open(f, 'rb'))

        #older cfgs have no 'set_node_degree_uninformative' ...
        if 'set_node_degree_uninformative' not in r['exp_cfg']['model']:
            r['exp_cfg']['model']['set_node_degree_uninformative'] = False

        res.append(r)
    return res

def get_keychain_value_iter(d, key_chain=None):
    key_chain = [] if key_chain is None else list(key_chain).copy()

    if not isinstance(d, dict):
        yield tuple(key_chain), d
    else:
        for k, v in d.items():
            yield from get_keychain_value_iter(v, key_chain + [k])

def get_keychain_value(d, key_chain):

    try:
        for k in key_chain:
            d = d[k]

    except Exception as ex:
        raise KeyError() from ex

    return d
```

```
In [ ]:
kc = {k: k[-1] for k, v in list(get_keychain_value_iter(__exp_res_meta))}
kc
```

```
Out [ ]:
{('exp_cfg', 'dataset_name'): 'dataset_name',
 ('exp_cfg', 'training', 'lr'): 'lr',
 ('exp_cfg', 'training', 'lr_drop_fact'): 'lr_drop_fact',
 ('exp_cfg', 'training', 'num_epochs'): 'num_epochs',
 ('exp_cfg', 'training', 'epoch_step'): 'epoch_step',
 ('exp_cfg', 'training', 'batch_size'): 'batch_size',
 ('exp_cfg', 'training', 'weight_decay'): 'weight_decay',
 ('exp_cfg', 'training', 'validation_ratio'): 'validation_ratio',
 ('exp_cfg', 'model', 'model_type'): 'model_type',
 ('exp_cfg',
 'model',
 'use_super_level_set_filtration'): 'use_super_level_set_filtration',
 ('exp_cfg', 'model', 'use_node_degree'): 'use_node_degree',
 ('exp_cfg',
 'model',
 'set_node_degree_uninformative'): 'set_node_degree_uninformative',
 ('exp_cfg', 'model', 'pooling_strategy'): 'pooling_strategy',
 ('exp_cfg', 'model', 'use_node_label'): 'use_node_label',
 ('exp_cfg', 'model', 'gin_number'): 'gin_number',
 ('exp_cfg', 'model', 'gin_dimension'): 'gin_dimension',
 ('exp_cfg', 'model', 'gin_mlp_type'): 'gin_mlp_type',
 ('exp_cfg', 'model', 'num_struct_elements'): 'num_struct_elements',
 ('exp_cfg', 'model', 'cls_hidden_dimension'): 'cls_hidden_dimension',
 ('exp_cfg', 'model', 'drop_out'): 'drop_out',
 ('exp_cfg', 'tag'): 'tag',
 ('cv_test_acc',): 'cv_test_acc',
 ('cv_val_acc',): 'cv_val_acc',
 ('cv_indices_trn_tst_val',): 'cv_indices_trn_tst_val',
 ('cv_epoch_loss',): 'cv_epoch_loss',
 ('start_time',): 'start_time',
 ('id',): 'id'}
```

```
In [ ]:
COL_NAMES = {
    ('exp_cfg', 'dataset_name'): 'dataset_name',
    #('exp_cfg', 'tag'): 'tag',
    # ('exp_cfg', 'training', 'lr'): 'lr',
    # ('exp_cfg', 'training', 'lr_drop_fact'): 'lr_drop_fact',
    # ('exp_cfg', 'training', 'num_epochs'): 'num_epochs',
    # ('exp_cfg', 'training', 'epoch_step'): 'epoch_step',
    ('exp_cfg', 'training', 'batch_size'): 'batch_size',
    # ('exp_cfg', 'training', 'weight_decay'): 'weight_decay',
    # ('exp_cfg', 'training', 'validation_ratio'): 'validation_ratio',
    ('exp_cfg', 'model', 'model_type'): 'model_type',
    ('exp_cfg', 'model', 'use_super_level_set_filtration'): 'use_super_level_set_filtration',
    ('exp_cfg', 'model', 'use_node_degree'): 'use_node_degree',
    ('exp_cfg', 'model', 'use_node_label'): 'use_node_label',
    ('exp_cfg', 'model', 'gin_number'): 'gin_number',
    ('exp_cfg', 'model', 'gin_dimension'): 'gin_dimension',
    #('exp_cfg', 'model', 'gin_mlp_type'): 'gin_mlp_type',
    ('exp_cfg', 'model', 'set_node_degree_uninformative'): 'set_node_degree_uninformative',
    ('exp_cfg', 'model', 'num_struct_elements'): 'num_struct_elements',
    ('exp_cfg', 'model', 'drop_out'): 'drop_out',
    ('exp_cfg', 'model', 'pooling_strategy'): 'pooling_strategy',
    # ('cv_test_acc',): 'cv_test_acc',
    # ('cv_val_acc',): 'cv_val_acc',
    # ('cv_indices_trn_tst_val',): 'cv_indices_trn_tst_val',
    # ('cv_epoch_loss',): 'cv_epoch_loss',
    # ('start_time',): 'start_time',
    # ('id',): 'id',
    ('finished_training',): 'finished_training'
}
```

```
In [ ]:
def pd_frame(path):

    f = read_exp_result_files(path)

    data_frames = []
    for i, res in enumerate(f):
        row = {}

        cv_acc_last = [x[-1] for x in res['cv_test_acc'] if len(x) > 0]

        row['acc_last_mean'] = np.mean(cv_acc_last)
        row['acc_last_std'] = np.std(cv_acc_last)

        cv_acc_validated = []
        for test, val in zip(res['cv_test_acc'], res['cv_val_acc']):
            if not len(test) == res['__exp_cfg']['training']['num_epochs']:
                continue
            n = len(test)//2
            test = torch.tensor(test[n:])
            val = torch.tensor(val[n:])
            #test = torch.tensor(test)
            #val = torch.tensor(val)

            _, i_max = val.max(0)
            cv_acc_validated.append(test[i_max].item())

        row['acc_val_mean'] = np.mean(cv_acc_validated)
        row['acc_val_std'] = np.std(cv_acc_validated)

        cv_folds_available = sum([1 for cv in res['cv_test_acc'] if len(cv) == res['__exp_cfg']['training']['num_epochs']])
        row['cv_folds_available'] = cv_folds_available

        for k, v in COL_NAMES.items():
            try:
                row[v] = get_keychain_value(res, k)
            except KeyError:
                pass

        f = pd.DataFrame(row, index=[i])

        data_frames.append(f)

    return pd.concat(data_frames, sort=True)
```

```
In [ ]:
path = './experiment_logs/'
RES = pd_frame(path)
RES = RES.sort_values(by=['dataset_name', 'model_type'], ascending=False)
# RES[RES['dataset_name'].str.contains('REDDIT') & (RES['gin_number'] == 3)]
RES
```

Out []:	acc_last_mean	acc_last_std	acc_val_mean	acc_val_std	batch_size	cv_folds_available	dataset_name	drop_out	finished_training	gin_dimension	gi
20	51.611182	8.204332	53.771182	1.631303	64	10	REDDIT-MULTI-5K	0.1	True	64	
43	53.851503	2.601825	54.311302	2.472776	64	10	REDDIT-MULTI-5K	0.1	True	64	
0	53.771583	1.873286	54.991904	2.479941	64	10	REDDIT-MULTI-5K	0.1	True	64	
24	53.291583	2.777682	52.691543	2.634461	64	10	REDDIT-MULTI-5K	0.1	True	64	
30	87.100000	3.006659	88.850000	2.366960	64	10	REDDIT-BINARY	0.1	True	64	
41	87.350000	2.775338	87.400000	3.128898	64	10	REDDIT-BINARY	0.1	True	64	
46	86.350000	3.090712	85.900000	2.662705	64	10	REDDIT-BINARY	0.1	True	64	
21	90.300000	1.307670	90.500000	1.596872	64	10	REDDIT-BINARY	0.1	True	64	
39	89.400000	1.609348	89.650000	1.162970	64	10	REDDIT-BINARY	0.1	True	64	
44	83.950000	2.953388	85.600000	2.517936	64	10	REDDIT-BINARY	0.1	True	64	
9	71.874196	3.222069	72.679376	3.779476	64	10	PROTEINS	0.1	True	64	
17	67.563124	2.880053	70.889186	0.780713	64	5	PROTEINS	0.1	True	64	
27	69.097490	3.779691	70.803571	3.887956	64	10	PROTEINS	0.1	True	64	
42	74.213227	2.086025	74.662062	1.734282	64	5	PROTEINS	0.1	True	64	
2	71.159051	2.319949	71.967033	2.904664	64	5	PROTEINS	0.1	True	64	
10	71.602317	3.920555	72.772683	3.411210	64	10	PROTEINS	0.1	True	64	
15	70.712641	3.020149	73.046904	1.737326	64	5	PROTEINS	0.1	True	64	
23	73.138205	2.533780	73.497356	2.226057	64	5	PROTEINS	0.1	True	64	
12	67.658462	3.916730	70.435168	3.939531	64	10	PROTEINS	0.1	True	64	
3	89.327485	4.813917	86.754385	7.486808	64	10	MUTAG	0.1	True	64	
28	83.976608	6.046060	82.426900	4.201684	64	10	MUTAG	0.1	True	64	
37	83.513514	2.581469	83.513513	1.973298	64	5	MUTAG	0.1	True	64	
25	88.805121	3.601307	89.345662	1.780311	64	5	MUTAG	0.1	True	64	
1	88.264580	3.704449	88.805121	4.904055	64	5	MUTAG	0.1	True	64	
45	84.035088	7.854783	85.643274	4.730069	64	10	MUTAG	0.1	True	64	
13	86.695906	6.356876	86.666666	6.025379	64	10	MUTAG	0.1	True	64	
31	50.333333	2.586289	51.466667	2.454927	64	10	IMDB-MULTI	0.1	True	64	
11	49.600000	3.028751	49.400001	2.943165	64	10	IMDB-MULTI	0.1	True	64	
36	74.200000	2.821347	74.700000	3.163858	64	10	IMDB-BINARY	0.1	True	64	
5	73.700000	3.348134	75.500000	2.801785	64	10	IMDB-BINARY	0.1	True	64	
4	72.600000	1.462874	73.700000	2.014944	64	5	IMDB-BINARY	0.1	True	64	
38	73.100000	2.200000	74.200000	1.503330	64	5	IMDB-BINARY	0.1	True	64	
33	74.900000	3.039737	73.100000	3.389690	64	10	IMDB-BINARY	0.1	True	64	
7	74.900000	2.981610	74.400000	3.555278	64	10	IMDB-BINARY	0.1	True	64	
35	78.463158	5.802750	78.721053	6.219219	64	10	DHFR	0.1	True	64	
19	79.770175	6.721874	80.701752	5.814899	64	10	DHFR	0.1	True	64	
32	75.662252	3.081428	75.528932	0.728321	64	5	DHFR	0.1	True	64	
16	78.702510	2.356774	76.725340	2.764108	64	5	DHFR	0.1	True	64	
29	79.643860	6.298231	78.454385	4.668299	64	10	DHFR	0.1	True	64	
34	78.046358	2.908635	76.589403	3.105531	64	5	DHFR	0.1	True	64	
26	77.782456	4.499432	77.380701	4.591231	64	10	DHFR	0.1	True	64	
6	79.222942	4.891501	80.291397	2.560826	64	10	COX2	0.1	True	64	
8	78.802035	4.885647	80.712305	2.813944	64	10	COX2	0.1	True	64	
14	80.301990	1.574569	81.372685	0.802416	64	5	COX2	0.1	True	64	
18	78.805765	2.136796	78.798900	0.502047	64	5	COX2	0.1	True	64	
22	82.656143	1.838411	81.585449	2.656718	64	5	COX2	0.1	True	64	
47	80.527290	3.307624	78.820537	3.389812	64	10	COX2	0.1	True	64	
40	79.449584	3.676848	80.721555	2.563945	64	10	COX2	0.1	True	64	

The following cells contain some utility for messing around with results, i.e., deleting etc.