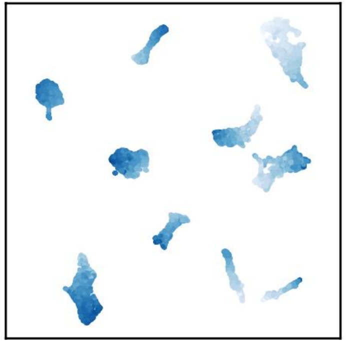
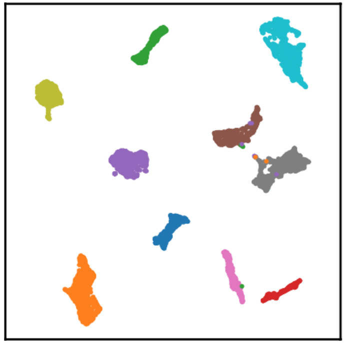
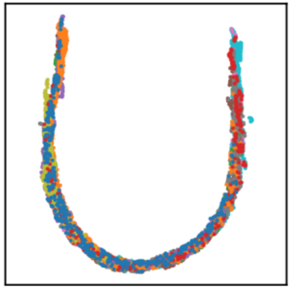




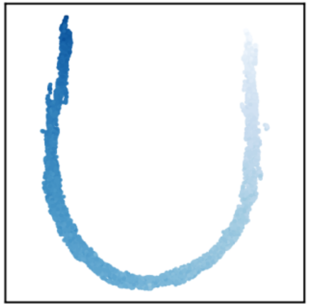
CS F425 - Deep Learning

Course Project, Group - 3



[Re] Rank-n-Contrast (NeurIPS’23 Spotlight)

Karan Bania (2021A7PS2582G)
Tejas Agrawal (2021A3PS2789G)
Rishav Mukherji (2020AAPS0330G)
Jinam Keniya (2021A7PS1622G)
Arnav Goyal (2021A7PS2596G)



Outline –

I. Reproduction of original results on AgeDB,

II. (Novel/Extra) Trying the loss on a Graph Regression task.

Part I. Reproduction of original results on AgeDB,

Dataset Used - AgeDB

1. Why **AgeDB**? Relatively smaller dataset for easier training
2. ~16k celebrity face images and age values
3. ~75:12.5:12.5 train:val:test split
4. Augmentation: RandomResizedCrop, RandomHorizontalFlip, RandomGrayscale, ColorJitter.
5. Obtained from here (i.bug UK)

Part I. Reproduction of original results on AgeDB,

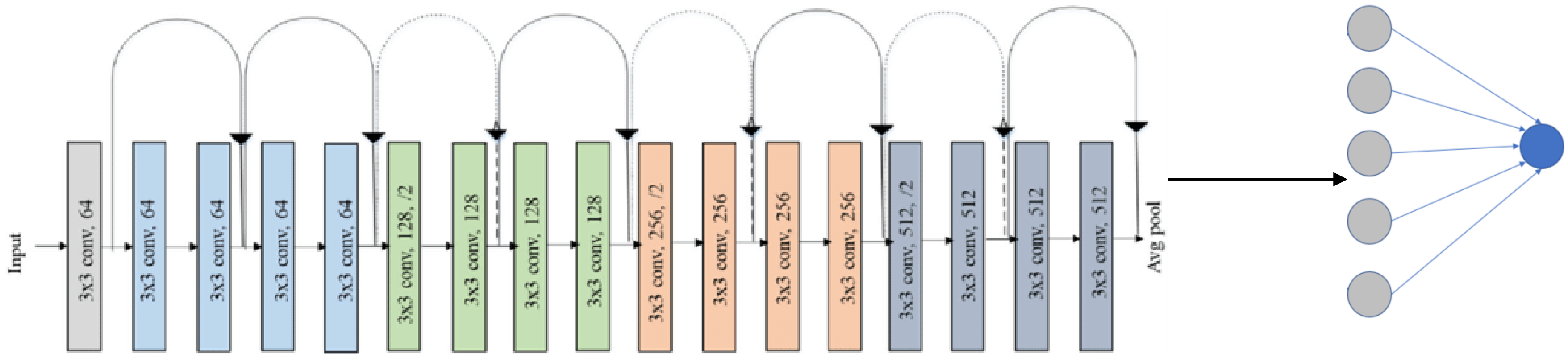
A few **examples**,

Reference : "AgeB: the first manually collected, in-the-wild age database"

Model Definition

ResNet-18

Linear layer



Encoder

Losses used to train encoder
L1, RnC

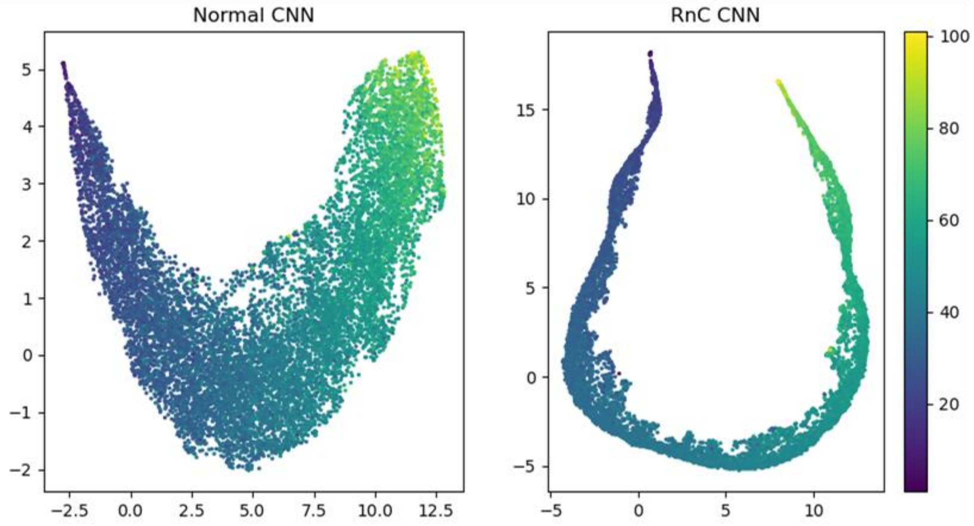
Regressor

Losses used to train regressor
L1, L2, Huber

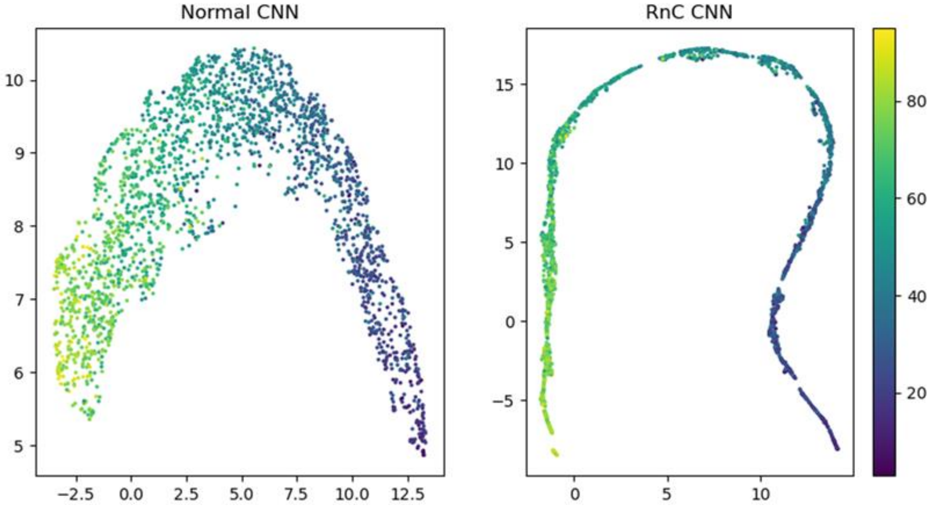
Experiments

Experiment	type	Encoder trained using	Regressor trained using
L1	1-stage	L1	L1
L2	1-stage	L2	L2
Huber	1-stage	huber	huber
RnC(L1)	2-stage	RnC	L1
RnC(L2)	2-stage	RnC	L2
RnC(Huber)	2-stage	RnC	huber

Encoder representation Visualisation



Train



Test

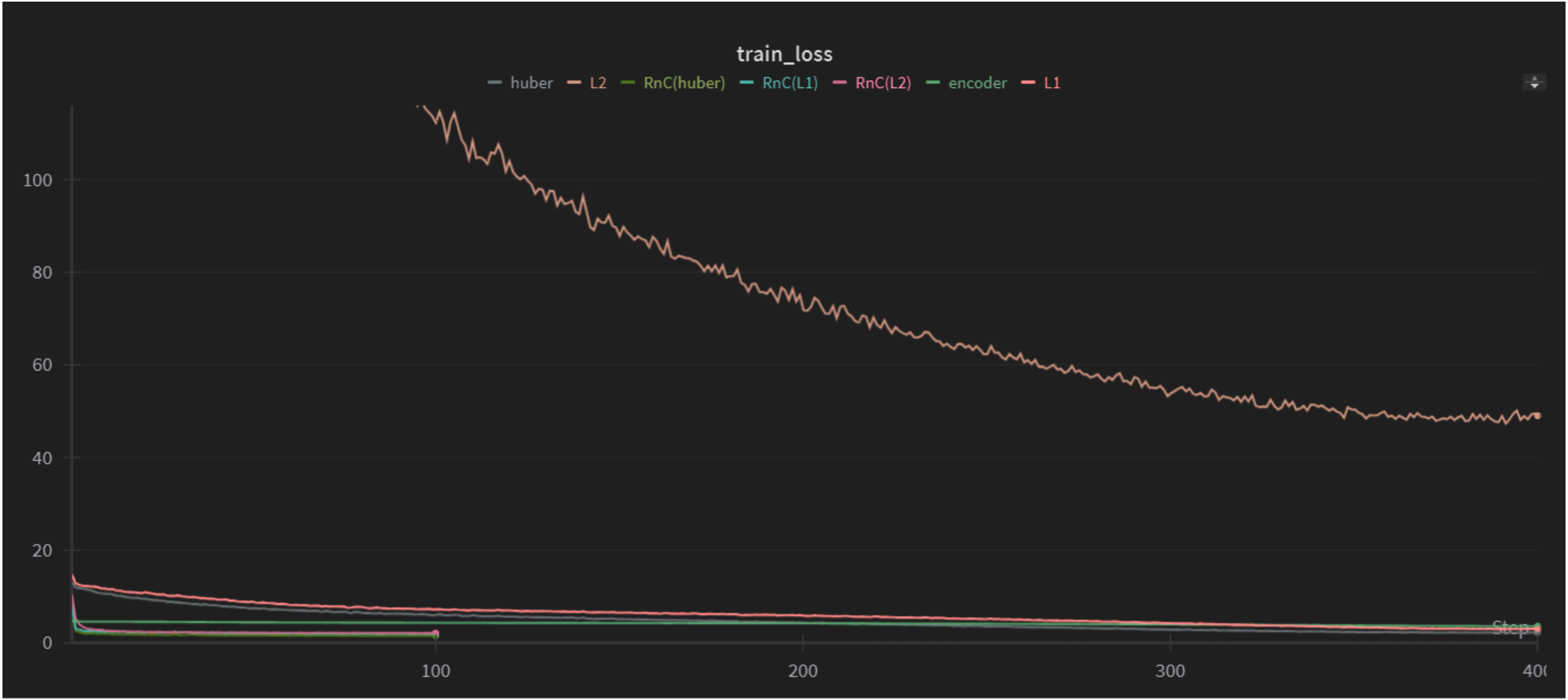
Hyperparams

Experiment	Batch size	Epochs	Learning Rate	Learning Decay	Weight Decay
L1	128	400	0.2	0.1	0.0001
L2	128	400	0.005	0	0.0001
Huber	128	400	0.05	0.1	0.0001
RnC(L1)	128	400(encoder) + 100(regressor)	0.5(encoder) + 0.05(regressor)	0.1(encoder) + 0.2(regressor)	0
RnC(L2)	128	400(encoder) + 100(regressor)	0.5(encoder) + 0.01(regressor)	0.1(encoder) + 0.2(regressor)	0
RnC(Huber)	128	400(encoder) + 100(regressor)	0.5(encoder) + 0.05(regressor)	0.1(encoder) + 0.2(regressor)	0.0001(encoder) + 0(regressor)

Reproduction of original results on AgeDB

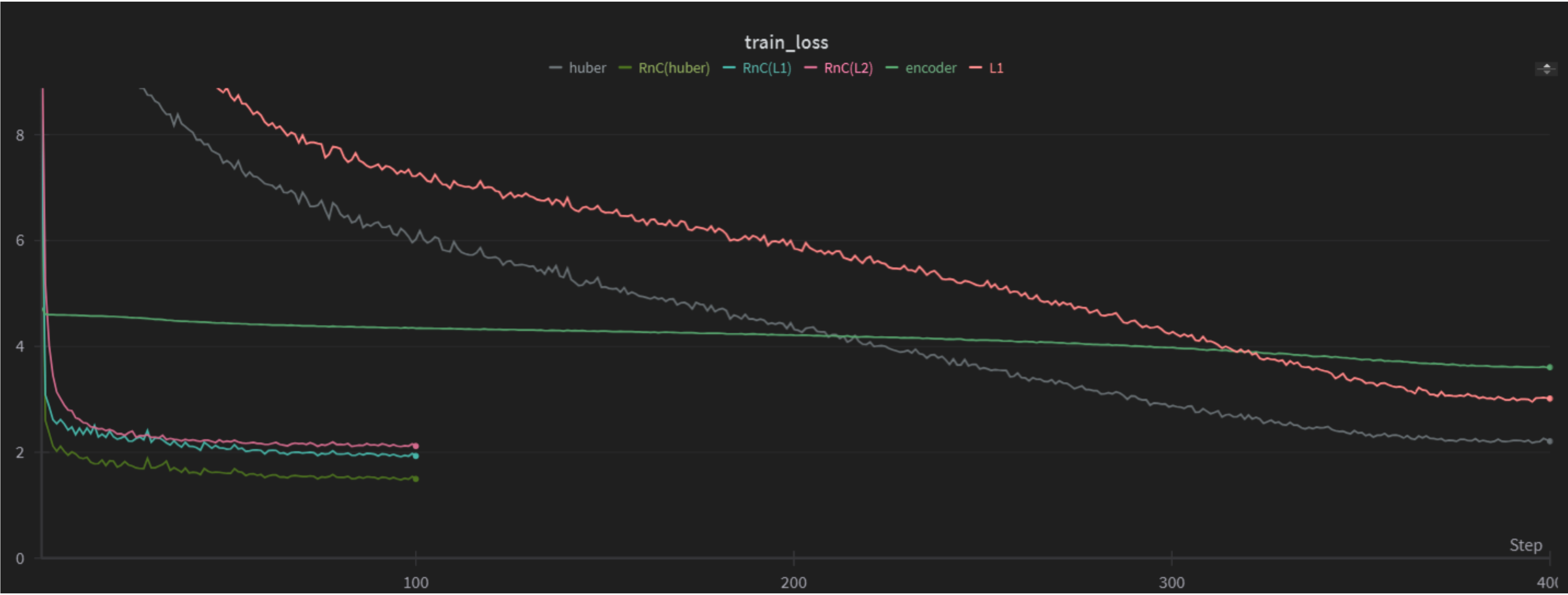
	Results on AgeDB (batch_size = 128)		
Experiments	Test MAE	Reported in paper (bs=256)	Error in replication(%)
L1	6.544	6.63	-1.297
L2	6.852	6.57	4.292
Huber	6.448	6.54	-1.407
RnC(L1)	6.154	6.14	0.228
RnC(L2)	6.439	6.19	4.023
RnC(Huber)	6.182	6.15	0.520

MAE Reduction by using RnC over standard loss (%)		
Experiments	Test Reduction	Reduction reported in Paper
RnC(L1)	-5.960	-7.391
RnC(L2)	-6.027	-5.784
RnC(Huber)	-4.125	-5.963

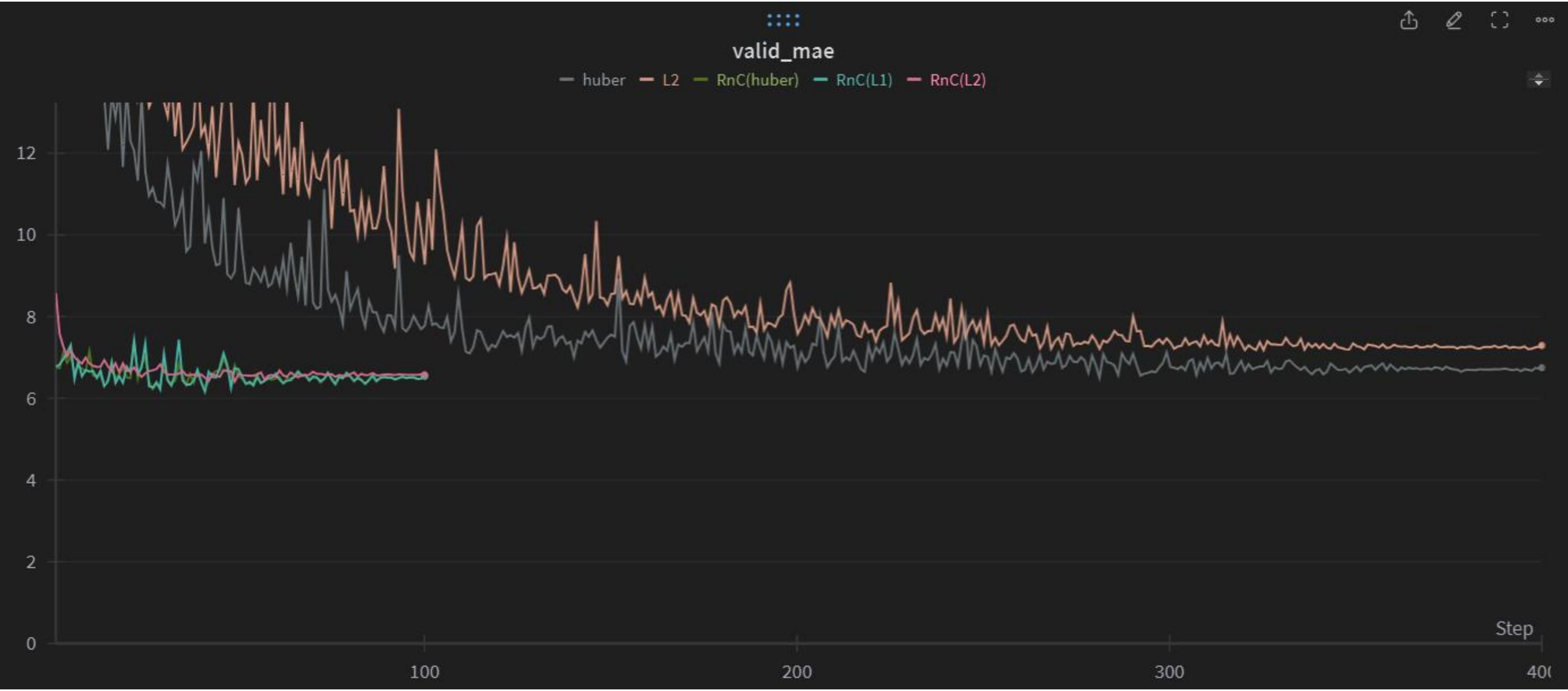


train_loss
huber
L2
RnC(huber)
RnC(L1)
RnC(L2)
encoder
L1
100
80
60
40
20
0
100
200
300
Step

Train Loss



Train Loss (L2 plot removed)



Validation Loss

Part II. (Novel/Extra) Trying the loss on a Graph Regression task,

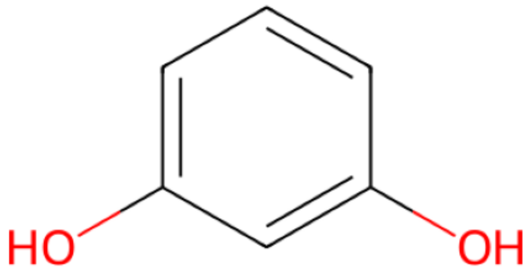
The **Dataset**,

1. ESOL (**E**stimated **SOL**ubility),
2. **1128** molecules and their solubility values,
3. Why **ESOL**? Small enough to experiment quickly, Relevant enough to be challenging.
4. **80:10:10** train:val:test split.

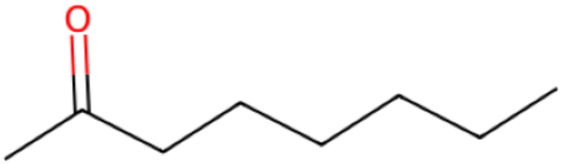
Part II. **(Novel/Extra)** Trying the loss on a Graph Regression task,

A few **examples**, (plotted using RDkit)

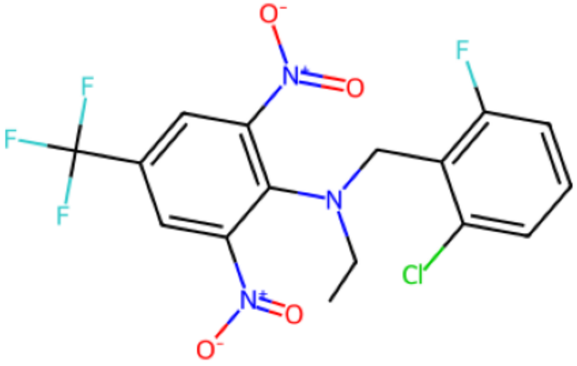
Molecule: Oc1cccc(O)c1
log(Solubility): 0.8100000023841858 mol/L



Molecule: CCCCCCC(=O)C
log(Solubility): -2.049999952316284 mol/L

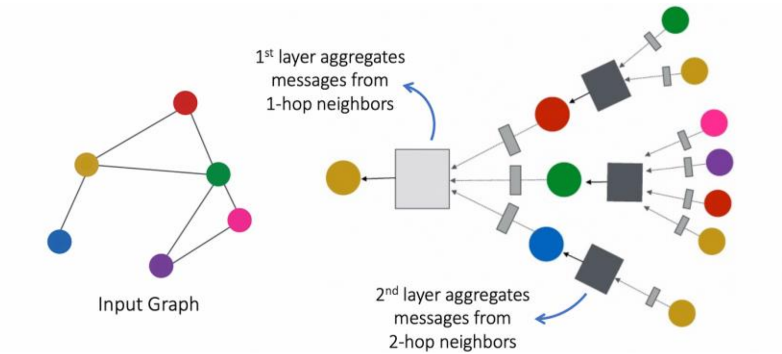


Molecule: CCN(Cc1c(F)cccc1Cl)c2c(cc(cc2N(=O)=...
log(Solubility): -6.78000020980835 mol/L



Part II. (Novel/Extra) Trying the loss on a Graph Regression task,

Model Definition



$$\mathbf{x}_i' = h_{\Theta}\left((1+\epsilon) \cdot \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \right)$$

```
class Encoder(nn.Module):

    def __init__(self, args : ModelArgs):
        super(Encoder, self).__init__()
        self.convs = nn.ModuleList()
        self.bns = nn.ModuleList()
        self.num_layers = args.num_layers

        self.convs.append(GINConv(nn.Sequential(nn.Linear(args.in_channels, 4*args.hidden_channels),
                                                nn.ReLU(),
                                                nn.Linear(4*args.hidden_channels, args.hidden_channels))))
        self.bns.append(nn.BatchNorm1d(args.hidden_channels))
        for _ in range(self.num_layers-2):
            self.convs.append(GINConv(nn.Sequential(nn.Linear(args.hidden_channels, 4*args.hidden_channels),
                                                    nn.ReLU(),
                                                    nn.Linear(4*args.hidden_channels, args.hidden_channels))))
            self.bns.append(nn.BatchNorm1d(args.hidden_channels))

        self.convs.append(GINConv(nn.Sequential(nn.Linear(args.hidden_channels, args.out_channels),
                                                nn.ReLU(),
                                                nn.Linear(args.out_channels, args.out_channels))))
```

Part II. (Novel/Extra) Trying the loss on a Graph Regression task,

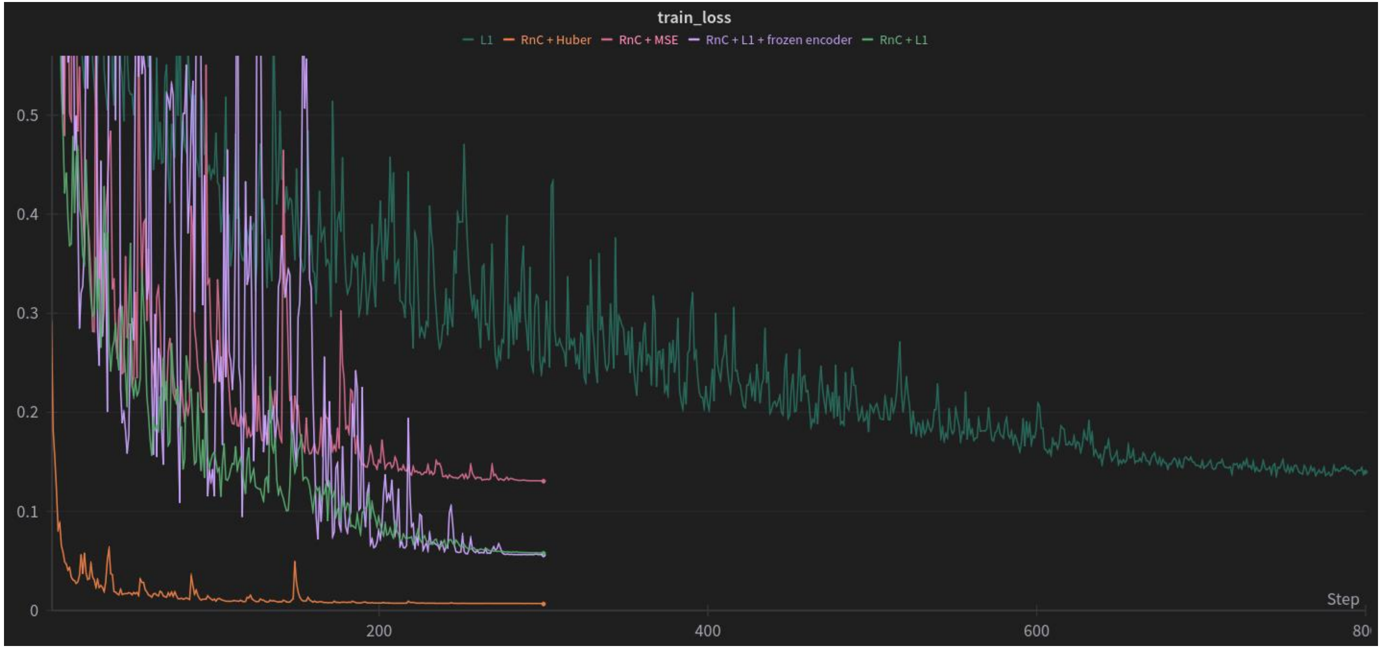
First, results and then the approach. [NUMBERS]

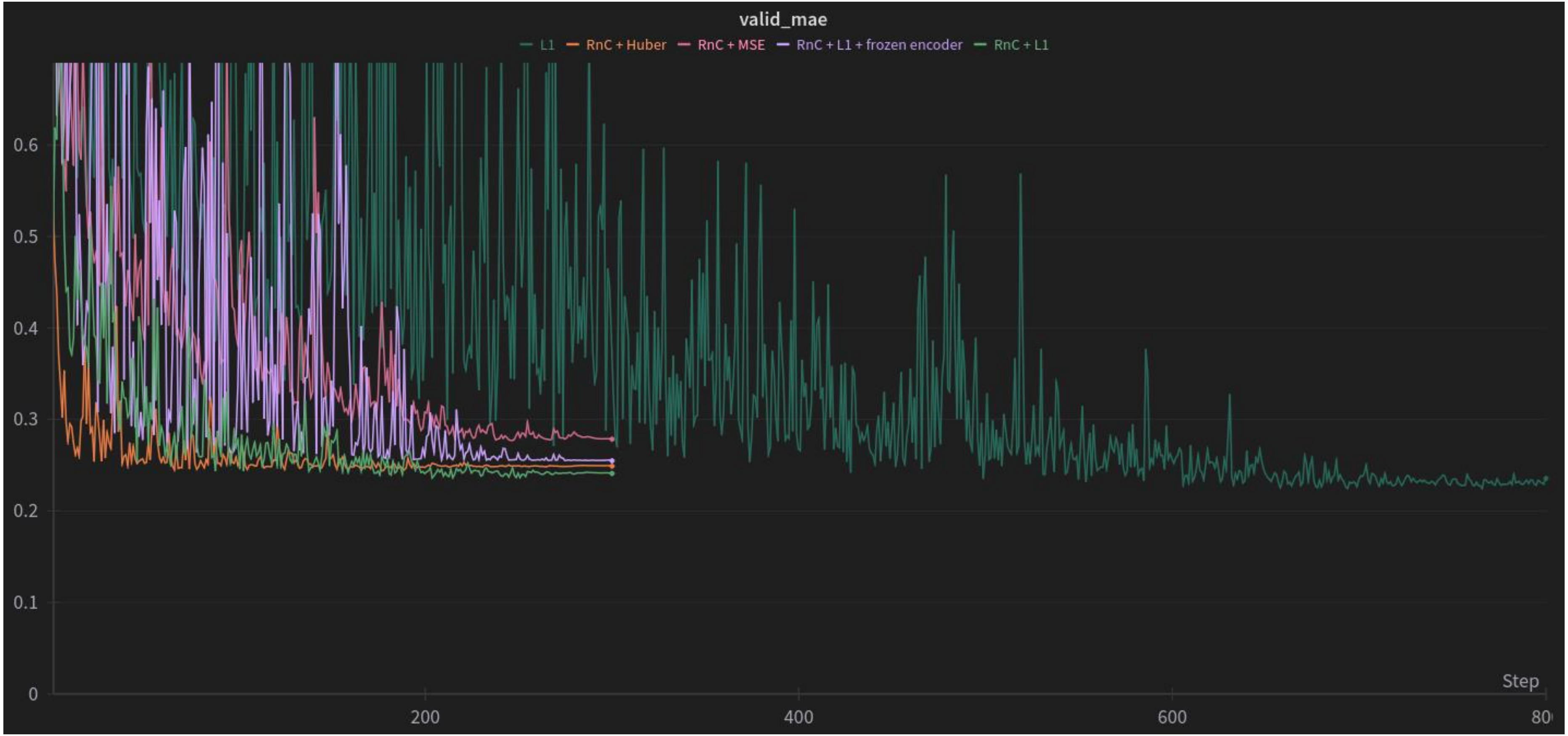
5th trial, 2x iterations on RnC	800 Iterations on RnC and normal-L1, then 300 for L1/L2/Huber					
Experiments	Test MAE	Test RMSE	Test MSE	Val MAE	Val RMSE	Val MSE
L1 (UPDATEDx2)	0.247	0.326	0.106	**0.224**	0.325	0.106
RnC(L1) + freeze	0.219	**0.297**	**0.088**	0.255	0.359	0.129
RnC(L1)	**0.212**	0.314	0.099	0.235	0.345	0.119
RnC(L2)	0.266	0.342	0.117	0.276	0.366	0.134
RnC(Huber)	0.242	0.326	0.106	0.245	**0.317**	**0.101**

Notice, all RnC based methods **beat normal-L1**, and the difference is **significant**.

Part II. (Novel/Extra) Trying the loss on a Graph Regression task,

First, **results** and then the **approach**. **[PLOTS]**

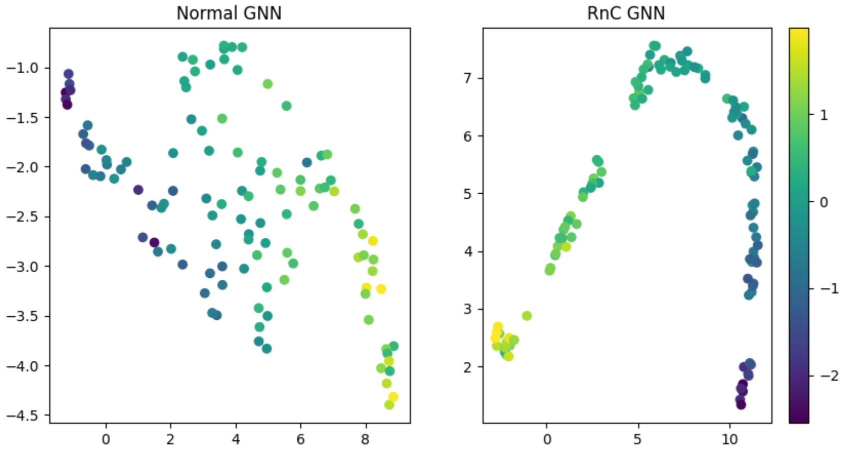



valid_mae
L1
RnC + Huber
RnC + MSE
RnC + L1 + frozen encoder
RnC + L1
0.6
0.5
0.4
0.3
0.2
0.1
0
Step
200
400
600
80

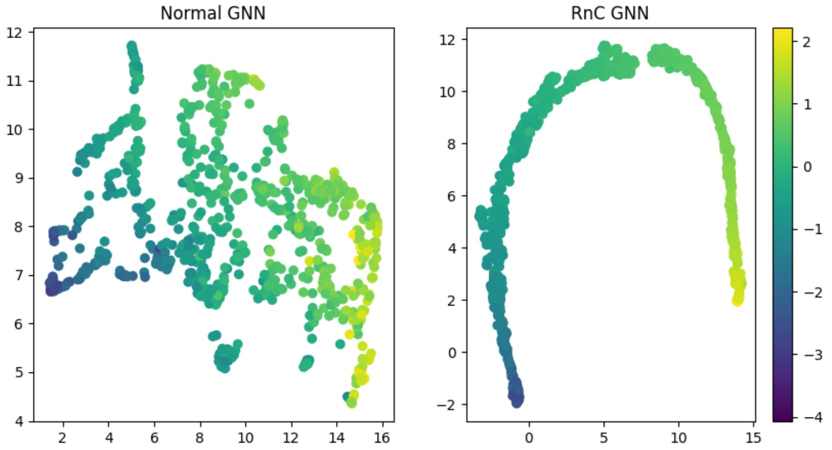
Part II. **(Novel/Extra)** Trying the loss on a Graph Regression task,

First, **results** and then the **approach**. **[REP-SPACE]**

Test



Train



Part II. **(Novel/Extra)** Trying the loss on a Graph Regression task,

Approach / Interesting findings,

A bigger MLP, **doesn't** work!

```
'''
EXTRA -
args.out_channels),
nn.ReLU(),
nn.Linear(args.out_channels, 512),
nn.ReLU(),
nn.Linear(512,
'''
```

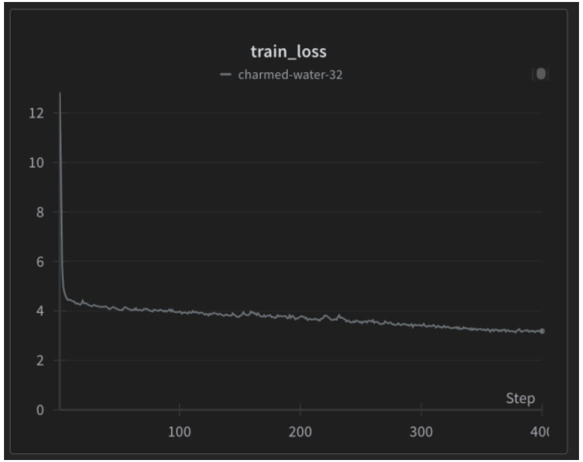
```
self.mlp = nn.Sequential(nn.Linear(2*args.out_channels, 1))
```

First and Zeroth trials!	Results on ESOL (1128 molecules, 902 train, 113 val, 113 train), results ater / => small MLP (1)					
Experiments	Test MAE	Test RMSE	Test MSE	Val MAE	Val RMSE	Val MSE
L1	**0.207 / 0.224**	**0.279 / 0.289**	**0.078 / 0.083**	**0.206 / 0.227**	**0.294 / 0.336**	**0.086 / 0.113**
RnC(L1) + freeze	0.278 / 0.311	0.366 / 0.408	0.134 / 0.166	0.247 / 0.251	0.335 / 0.349	0.112 / 0.122
RnC(L1)	0.335 / 0.248	0.437 / 0.332	0.191 / 0.110	0.304 / 0.260	0.423 / 0.412	0.177 / 0.170
RnC(L2)	0.267 / 0.294	0.361 / 0.363	0.13 / 0.131	0.255 / 0.286	0.359 / 0.396	0.129 / 0.157

Part II. (Novel/Extra) Trying the loss on a Graph Regression task,

Approach / Interesting findings,

Bigger LR for RnC.



Second trial with more RnC	Results on ESOL (1128 molecules, 902 train, 113 val, 113 train), results ater / => small MLP (2)					
Experiments	Test MAE	Test RMSE	Test MSE	Val MAE	Val RMSE	Val MSE
L1 (same)	**0.224**	**0.289**	**0.083**	0.227	0.336	0.113
RnC(L1) + freeze	0.311	0.408	0.166	0.251	0.349	0.122
RnC(L1)	0.248	0.317	0.1	**0.215**	0.314	0.098
RnC(L2)	0.25	0.328	0.108	**0.215**	**0.292**	**0.086**

Part II. (Novel/Extra) Trying the loss on a Graph Regression task,

Approach / Interesting findings,

Expose normal-L1 and RnC to **same information**.

6th trial, less batch_size RnC	Batch Size = 128					
Experiments	Test MAE	Test RMSE	Test MSE	Val MAE	Val RMSE	Val MSE
L1 (UPDATED)	0.258	**0.34**	**0.116**	0.252	0.36	0.13
RnC(L1) + freeze	**0.253**	0.368	0.136	0.233	0.338	0.114
RnC(L1)	0.265	0.387	0.15	0.221	0.32	0.103
RnC(L2)	0.268	0.375	0.141	**0.219**	**0.299**	**0.089**
RnC(Huber)	0.288	0.402	0.162	0.227	0.301	0.091

Part II. (Novel/Extra) Trying the loss on a Graph Regression task,

Further experiments?

1. Test till what epoch does RnC performance really increase? Because, loss was decreasing.
2. Try to see what hyperparams work for a bigger MLP?
3. More regression datasets?

Thank you! Questions?

[link to our public repository with all the code]

[link to google sheet with experimental results on the AgeDB dataset]

[link to google sheet with a (few) more experiments for the ESOL dataset]

[link to the w&b report with all runs on the AgeDB dataset]

[link to the w&b with all runs on the ESOL dataset]