

# Extracting meaningful relationships between commonly-traded commodities

PB3 - Final Results

Yifei Liu - yliu523  
Kwong Chun Wu - kwu764  
Kam leong Ao - kao323  
Yong Yu - yyu482  
Mark Chen - zche677

# Project Recap: Proposal

- GNN
- Crypto
- Black Box

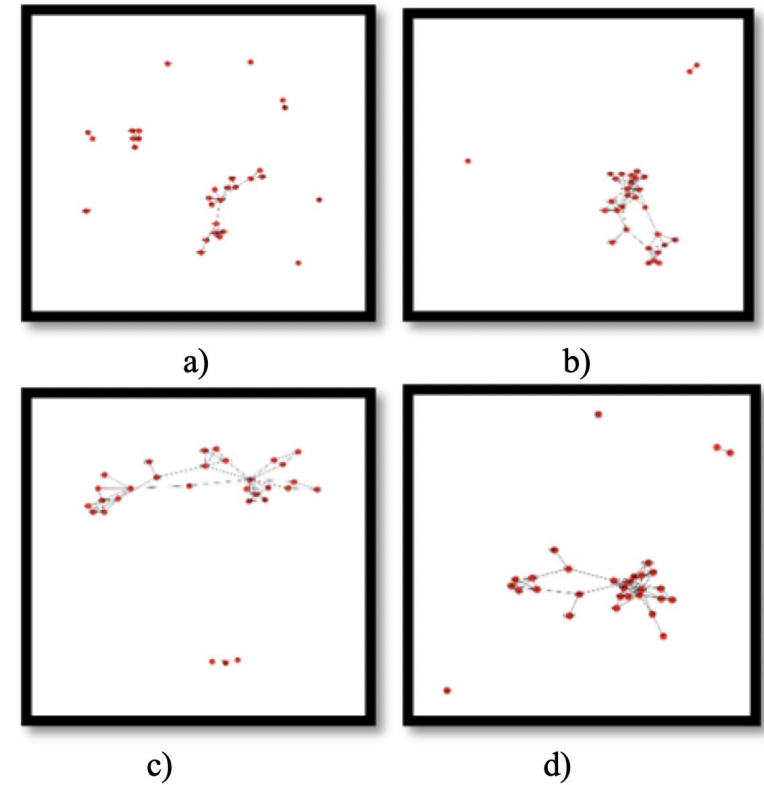


Fig. 1: Different graph structures created by different correlation coefficients [1]

# Project Recap: Proof of Concept

- ~~GNN~~ ► GAT - Graph Attention Network
- ~~Crypto~~ ► Traded Commodities
- ~~Black Box~~ ► Breakdown GAT
- Issues with GAT training

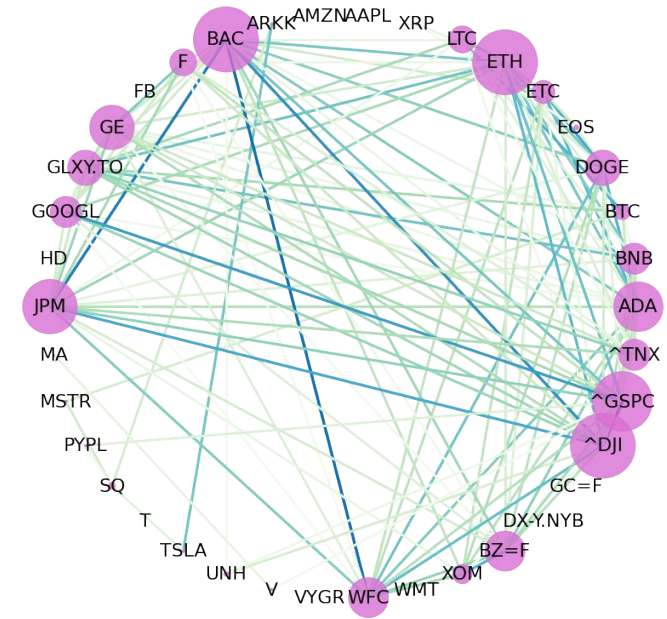


Fig. 2: Correlation matrix visualization - all commodities, Kendall coefficient  
[Generated by Mark]

# Project Recap: Methodology Improvements

- Literature survey-Inspired
- Normalize commodity prices
- 5-Week sliding window GAT
- Additional LSTM component

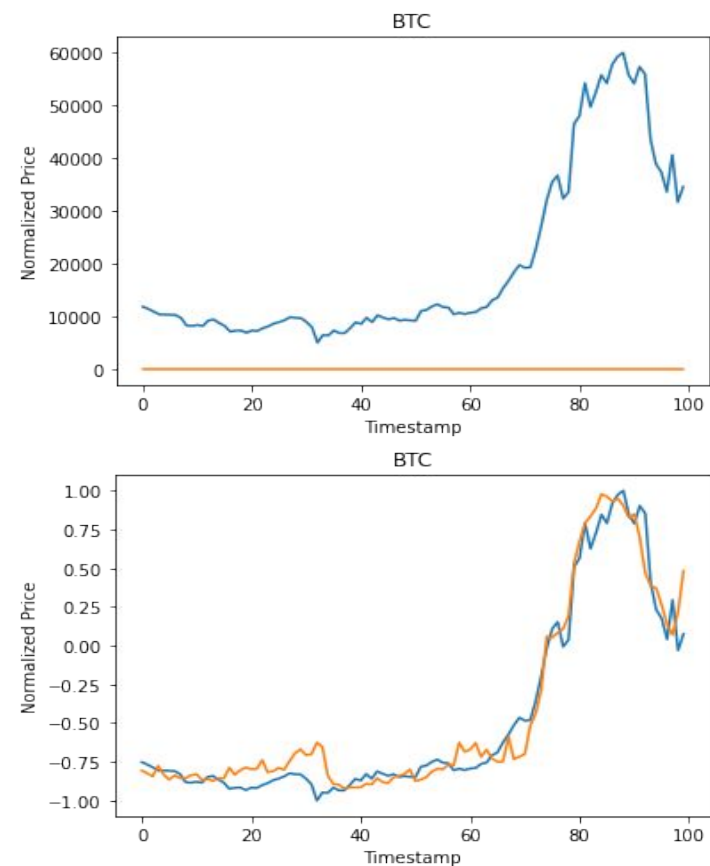


Fig. 3: Performance of model before & after literature survey inspired changes.  
[Generated by Mark]

# Project Recap: Final Results

- Results of final tuning
- Address feedback:
- Baseline & Evaluation
- Hyperparameter tuning
- Risks & Further work

Before we get into the details:

- Formally re-define problem
- High-level overview of model

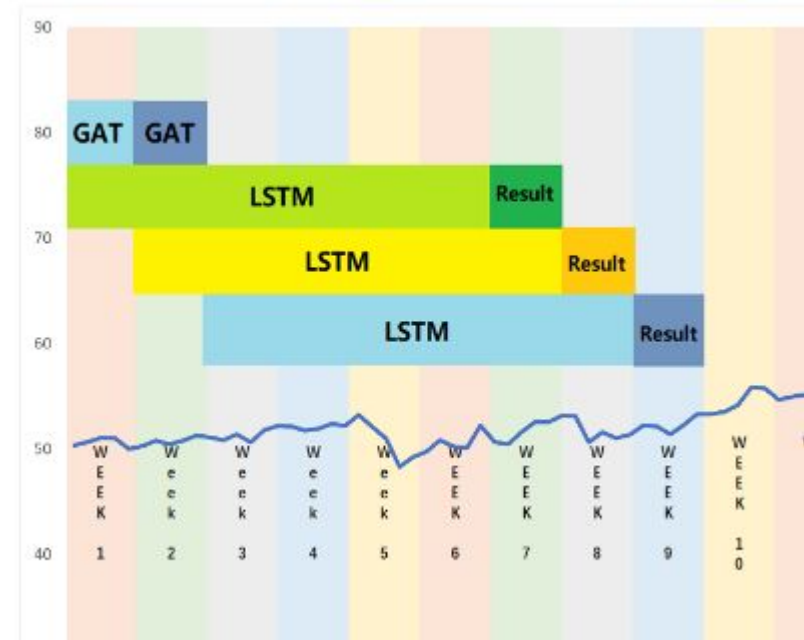


Fig. 4: Illustration of final model behaviour  
[Created by Yong]

# Problem Definition

- **Motivation:** To gain a deeper understanding of GAT applications in commodity-trading.
- **Problem:** Vanilla GAT implementation is not time-aware out of the box.
- **Objective:** Find the best way to customize GAT so that it can thrive when working in time-series context of commodity trading.
- **Visual Overview:**

Date	Open	High	Low	Close
7/1/2019	46.042	46.62	45.256	45.434
7/2/2019	45.778	45.83	44.444	44.91
7/3/2019	47.878	48.314	46.902	46.98
7/5/2019	46.914	47.09	46.16	46.62
7/8/2019	46.248	46.45	45.732	46.068
7/9/2019	45.794	46.2	45.456	46.012
7/10/2019	46.83	47.788	46.628	47.784
7/11/2019	47.628	48.3	47.16	47.72
7/12/2019	47.95	49.076	47.942	49.016
7/15/2019	49.6	50.884	48.972	50.7

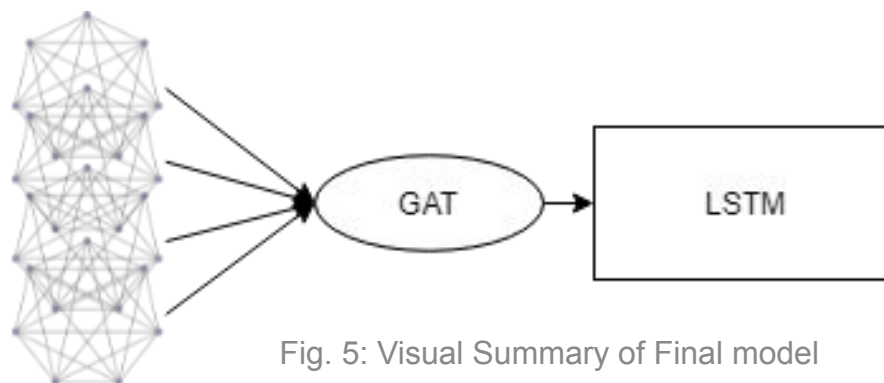
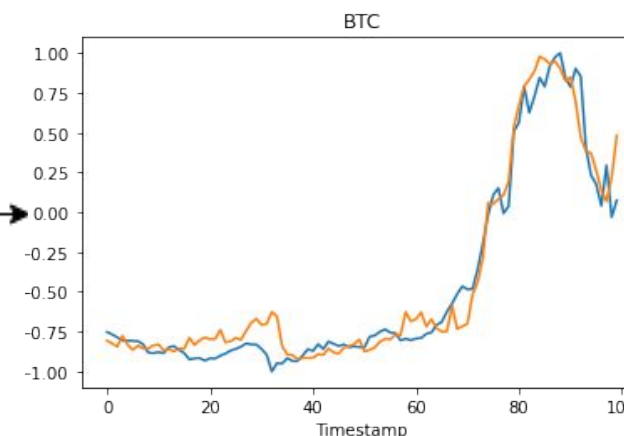


Fig. 5: Visual Summary of Final model  
[Created by Yifei]



# Experimental design - Data

Data used in our experiment.

Types	<ul style="list-style-type: none"><li>→ <b>22</b> stocks</li><li>→ <b>9</b> cryptocurrencies</li><li>→ <b>6</b> indexes + commodities</li></ul>
Length	<b>2 years</b> — 1st July, 2019 to 30th June, 2021
Interval	<b>Weekly (Average)</b>
Features	<b>4</b> — Prices of Open, High, Low & Close
Normalization	<b>Standard scaler</b>
Train / Test split	<b>80%</b> for train & <b>20%</b> for test
Package implementation	Pytorch & Pytorch Geometric 1.9.1

# Training Process

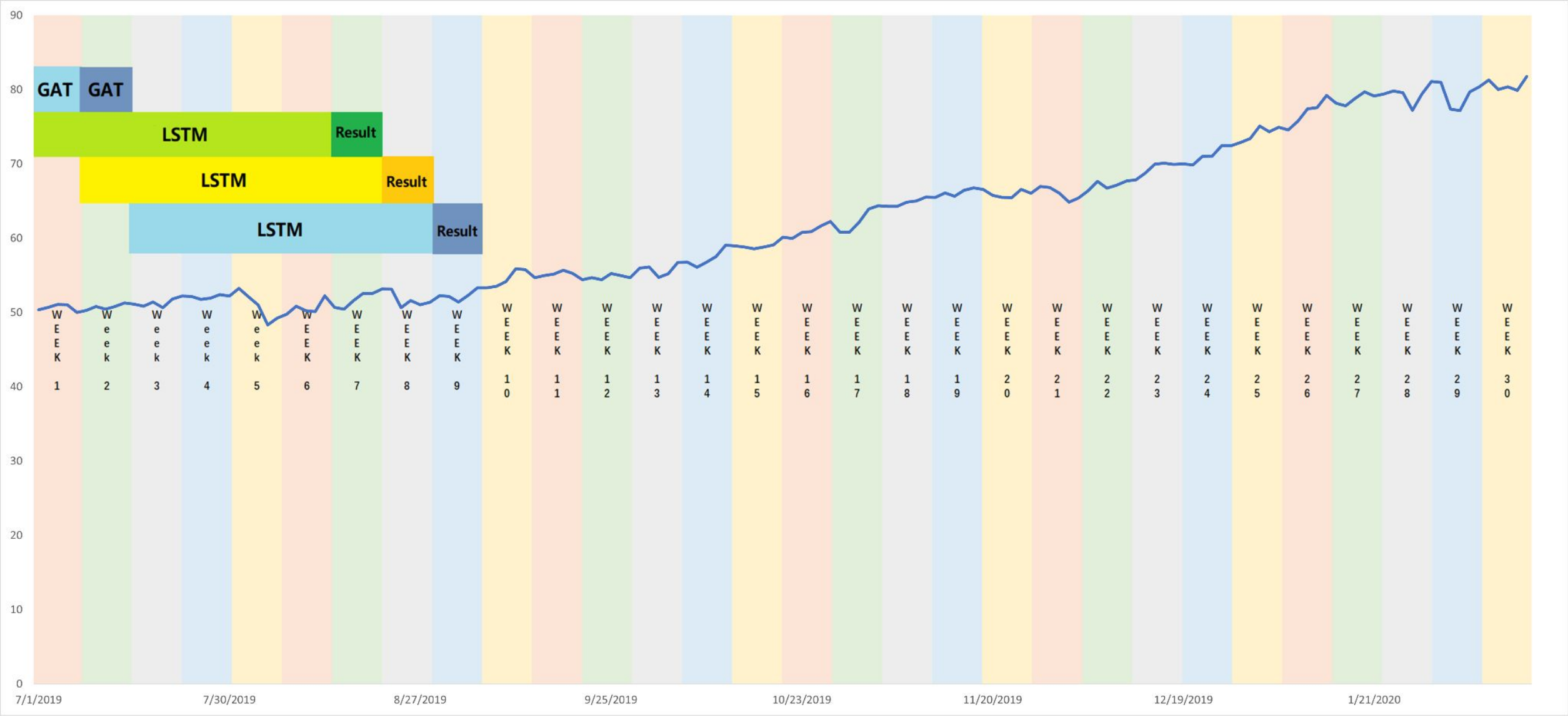


Fig. 6: Illustration of final model behaviour  
[Created by Yong]



# Experimental design: GAT+LSTM model

- Five hyperparameter settings in our proposed model:

Parameters	Default Setting
<b>N</b> : Using previous N weeks to predict the next week	N = 5
<b>F</b> : Feature inputted into GAT	F = Close, Open
<b>O</b> : Output feature size before the GAT's last layer	O = 8
<b>H</b> : Attention mechanism number (heads)	H = 2
<b>L</b> : Number of hidden feature size in LSTM	L = 32

- Training Phase: Adam Optimizer and MSE loss function
  - Learning rate = 0.001
  - Epochs = 500

# Experimental design: Baseline model

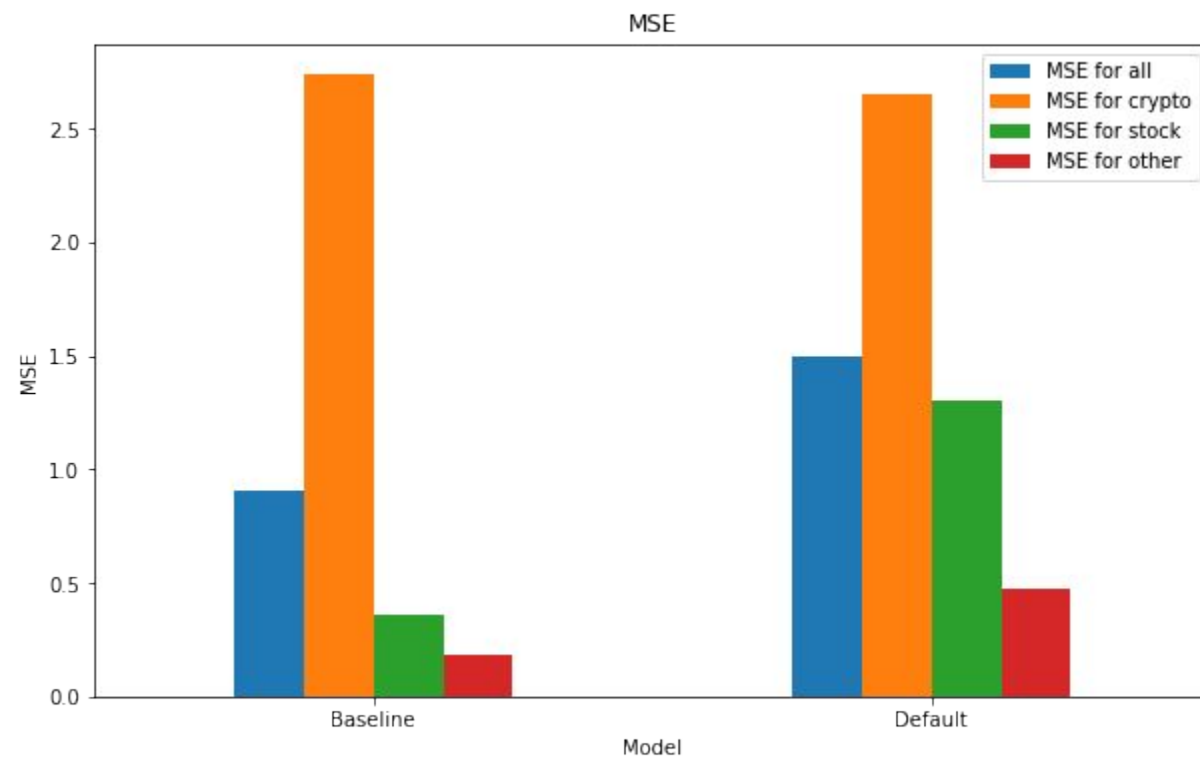
- Three hyperparameters in the baseline model (Pure LSTM):

Parameters	Baseline Setting
<b>N</b> : Using previous N weeks to predict the next week	N = 5
<b>F</b> : Feature inputted into LSTM	F = Close
<b>L</b> : Number of hidden feature size in LSTM	L = 32

- Training Phase: Adam Optimizer and MSE loss function
  - Learning rate = 0.001
  - Epochs = 200
- To compare the accuracy and MSE value with proposed model as the goal

# Result: Baseline vs Default

MSE



# Result: Is MSE enough?

In trading models, taking MSE to measure accuracy is **not** enough. **DIRECTION** of prediction can't be ignored.

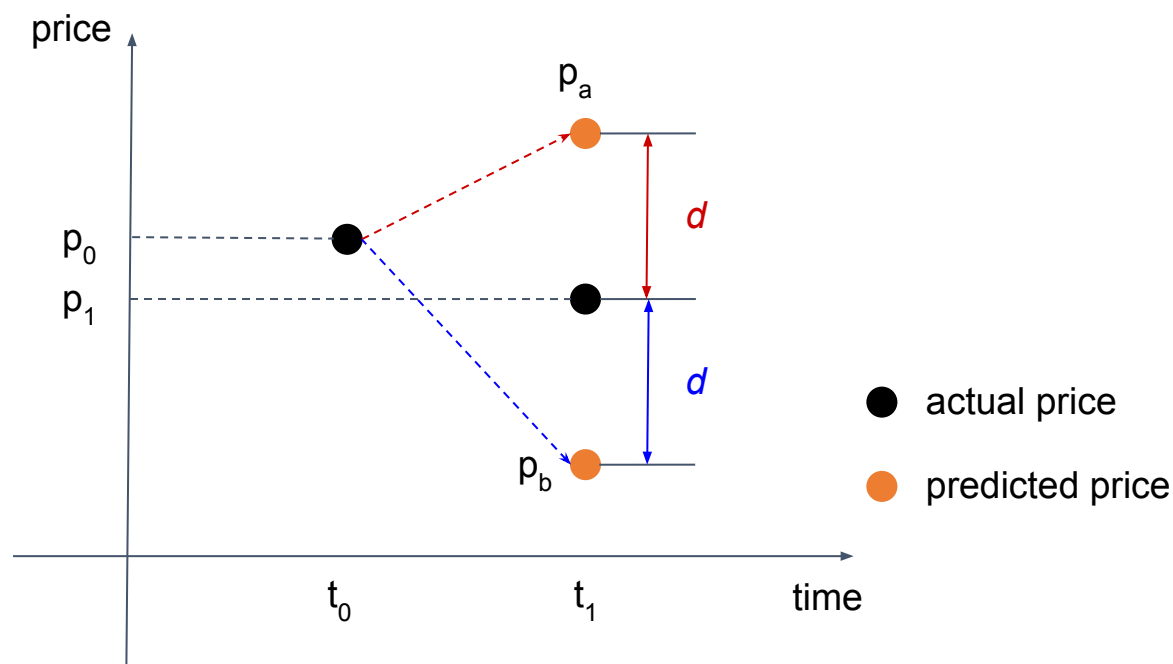


Fig. 7: Baseline vs GAT-LSTM Accuracy Comparison  
[Generated by Henry]

The spot price will drop from  $p_0$  to  $p_1$ , where  $p_a$  and  $p_b$  are two possible predictions at  $t_1$  by the model. We would buy if a rise is predicted.

Both  $p_a$  and  $p_b$  gives the same MSE.

1. Take action based on  $p_a$ , we will buy.  
Result = loss.
2. Take action based on  $p_b$ , no buy.  
Result = no loss.

# Result: MSE vs Accuracy

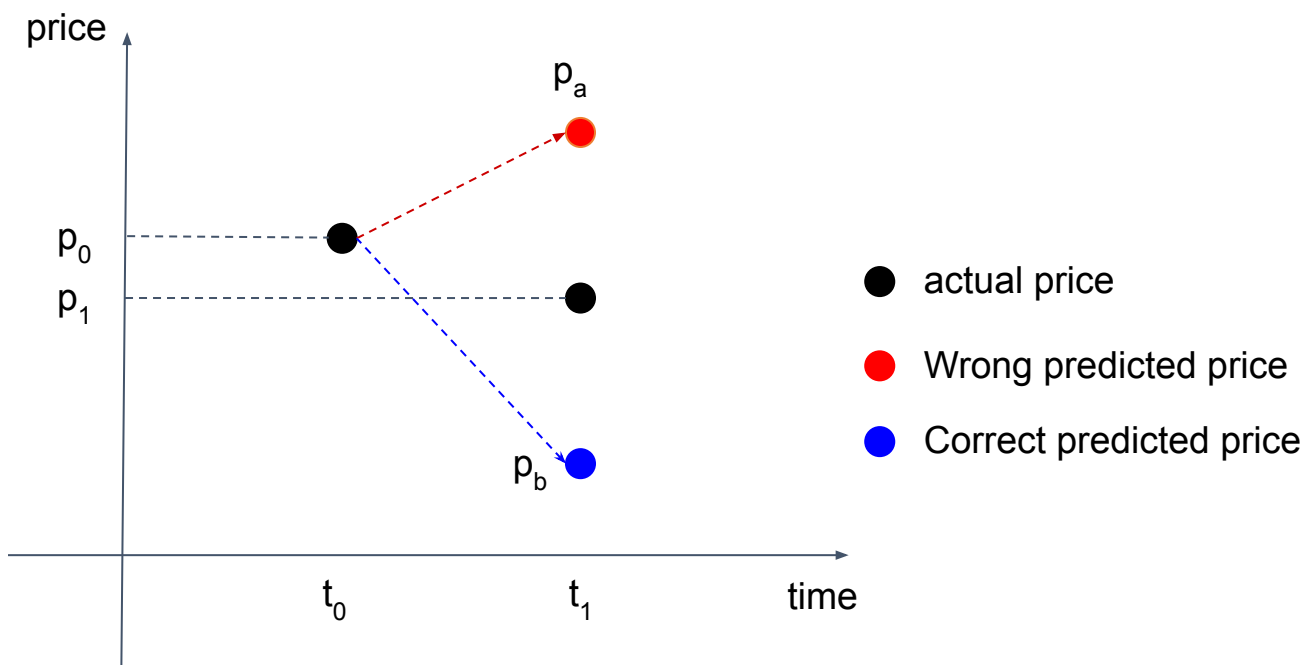
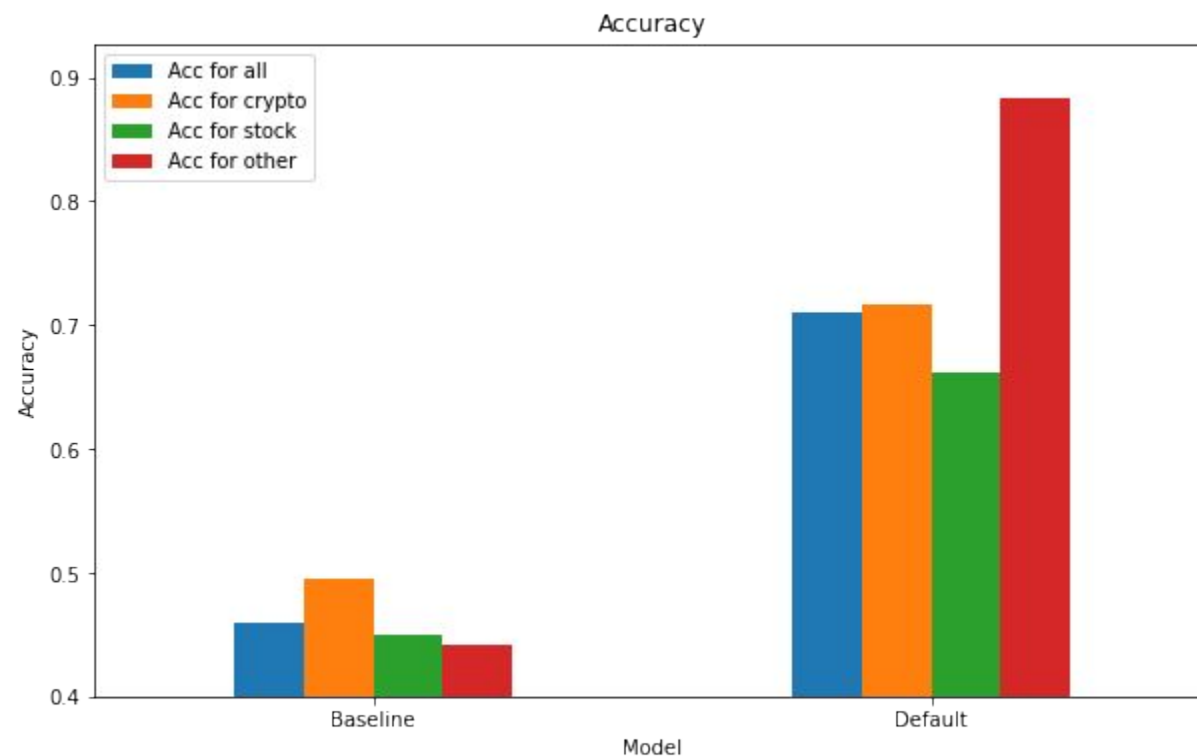


Fig. 8: Baseline vs GAT-LSTM Accuracy Comparison  
[Generated by Henry]

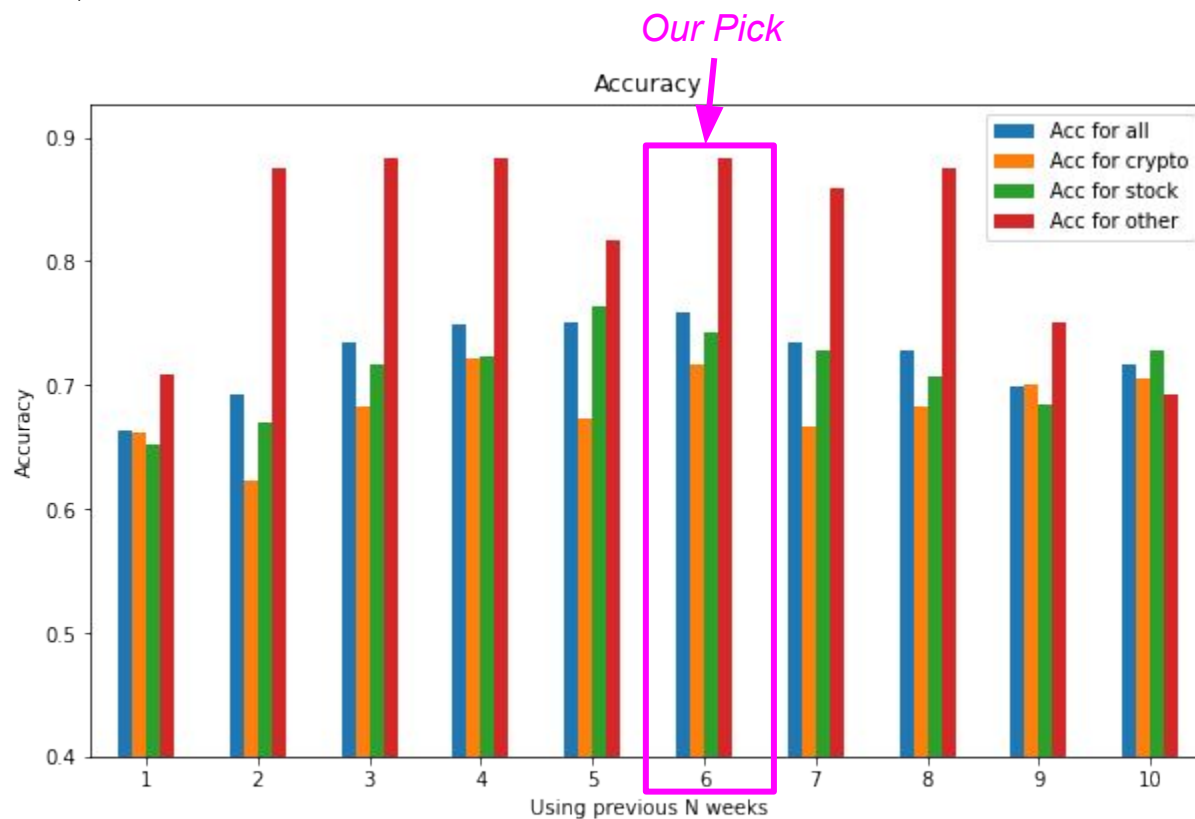
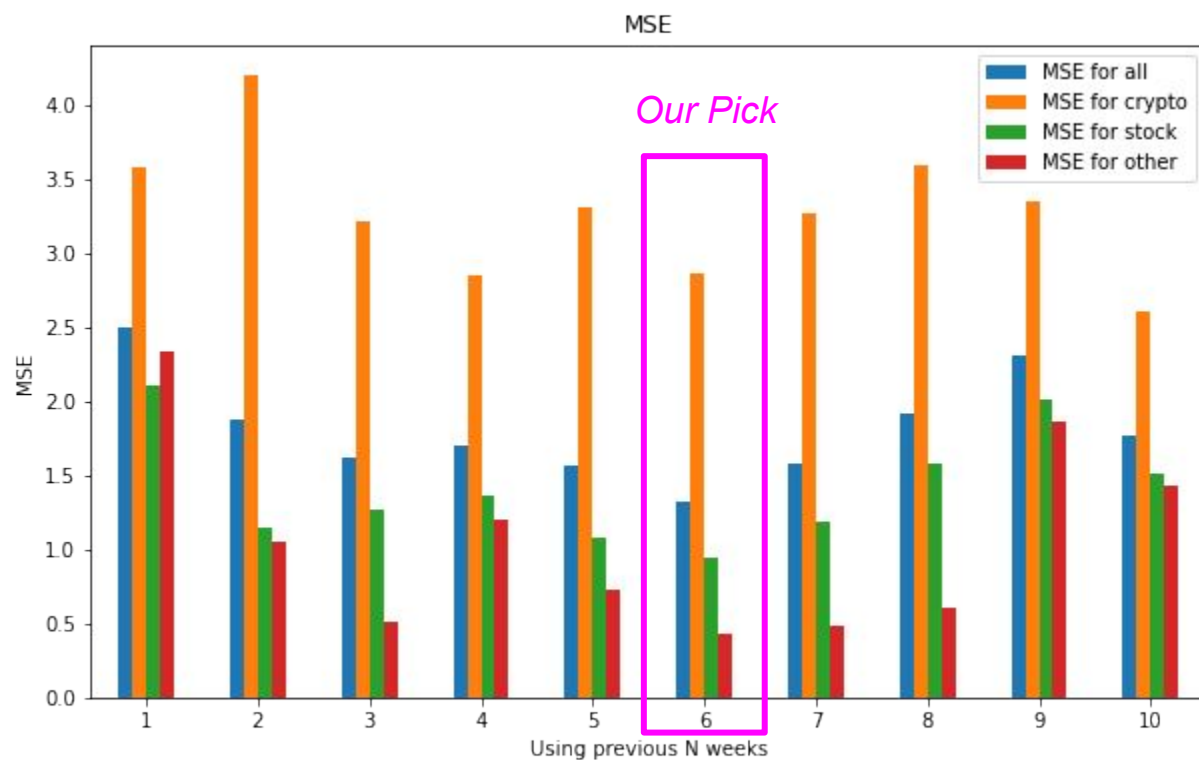
# Result: Baseline vs Default

Accuracy



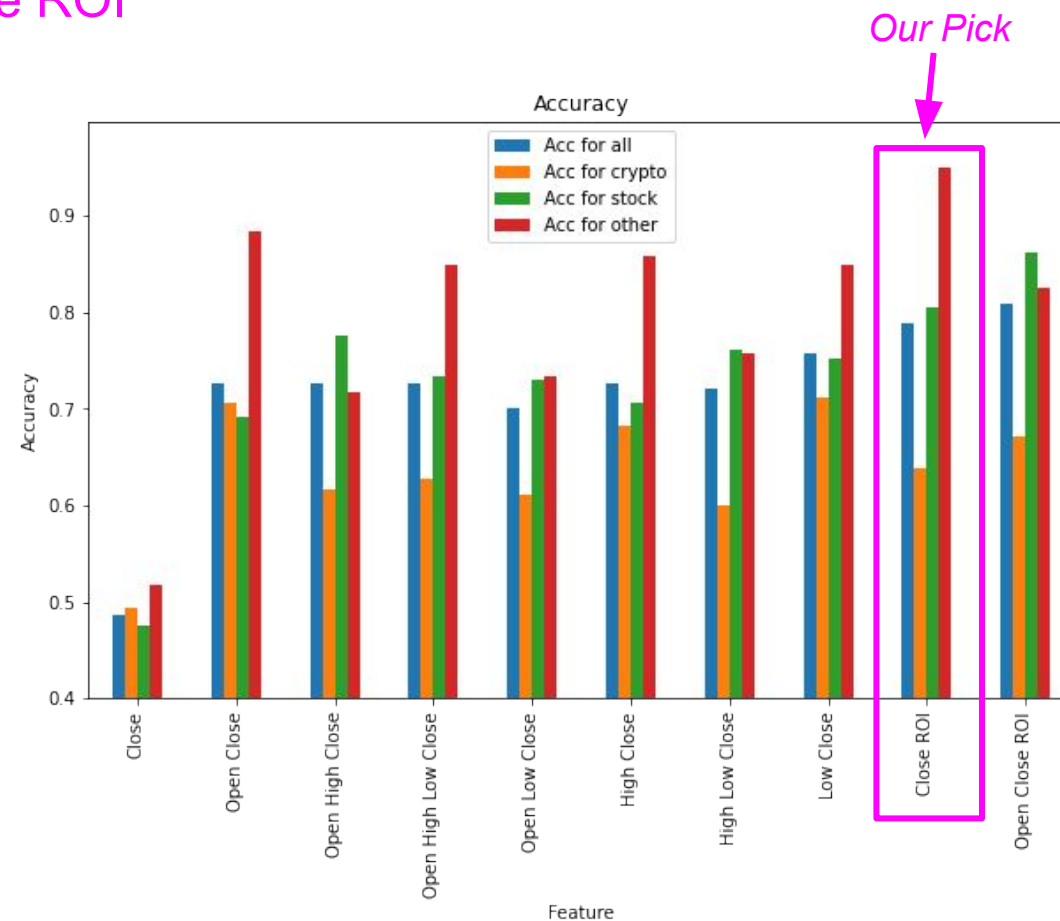
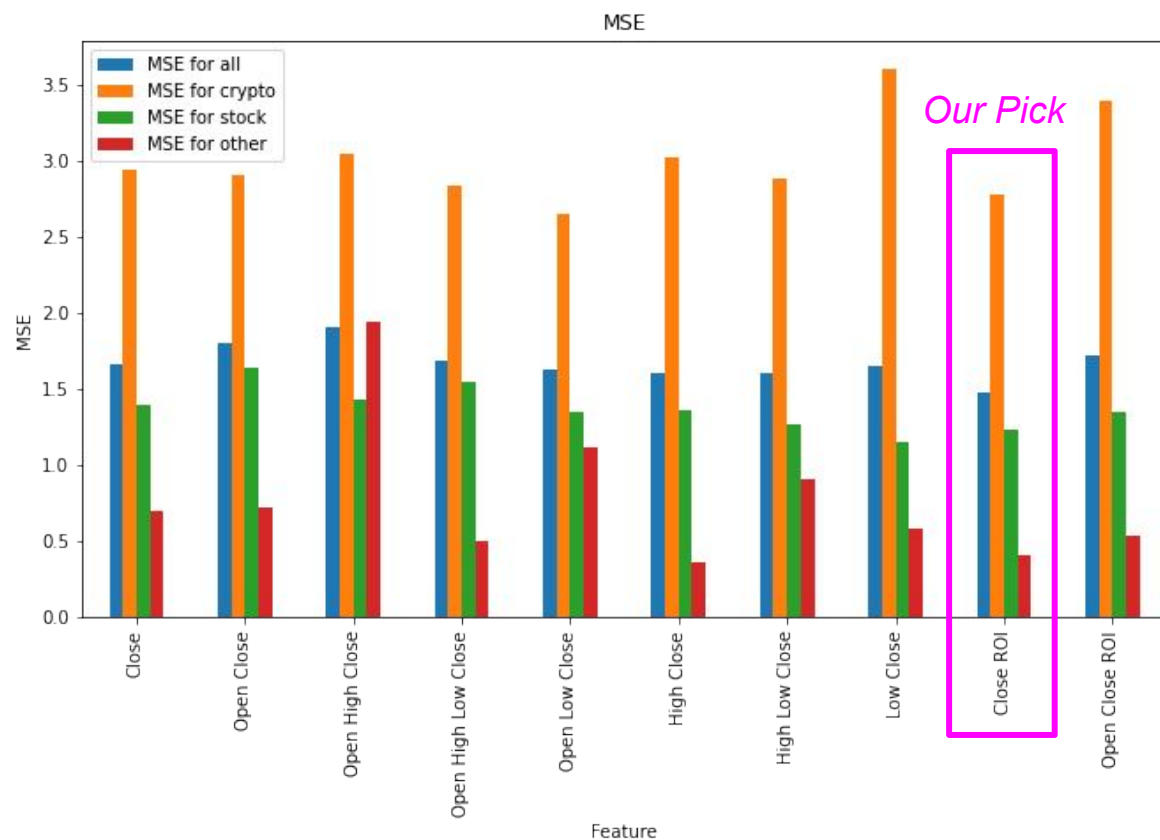
# Hyperparameter Optimization: N

**N:** Using previous N weeks to predict the next week, **Best N = 6**



# Hyperparameter Optimization: F

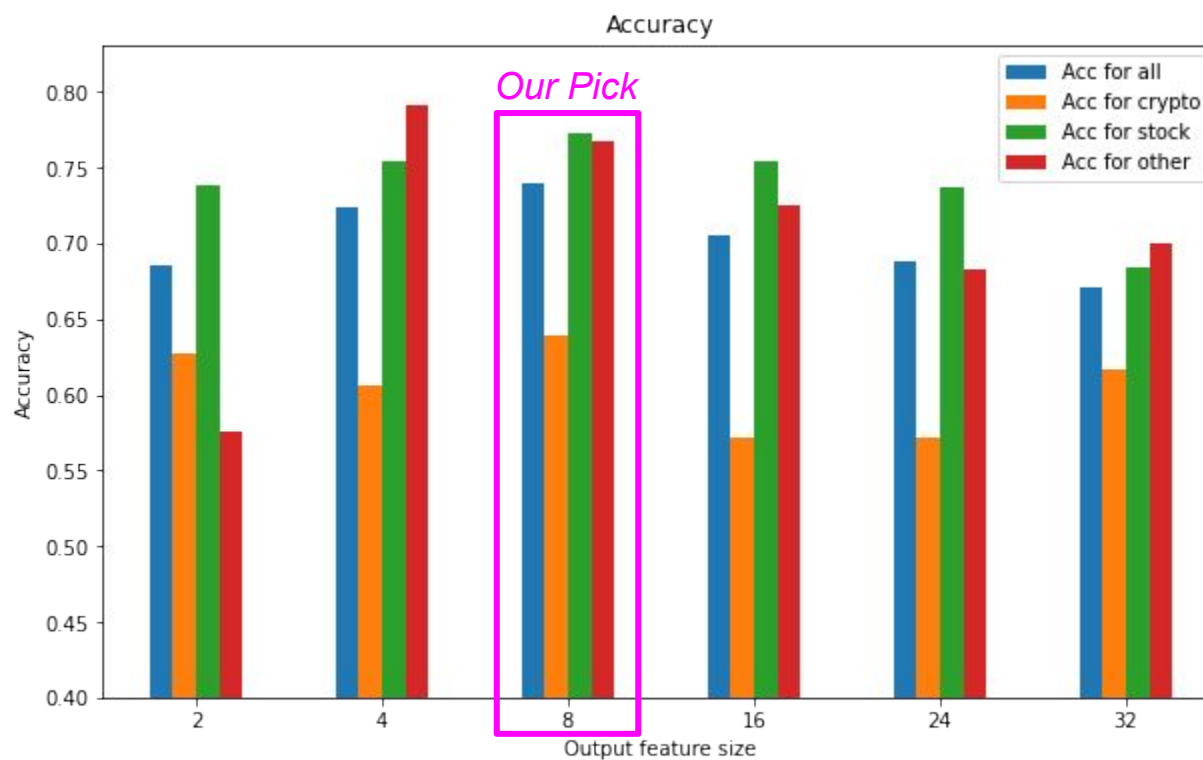
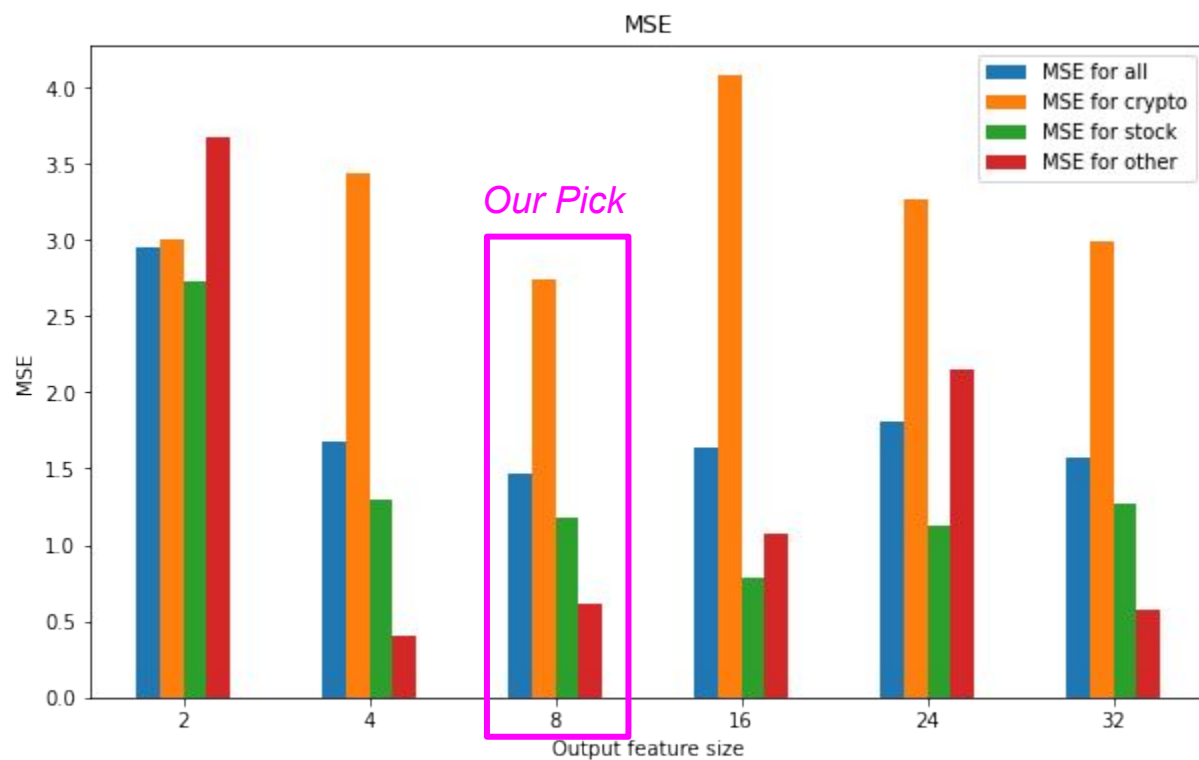
**F:** Feature(s) inputted into GAT, **Best F = Close ROI**





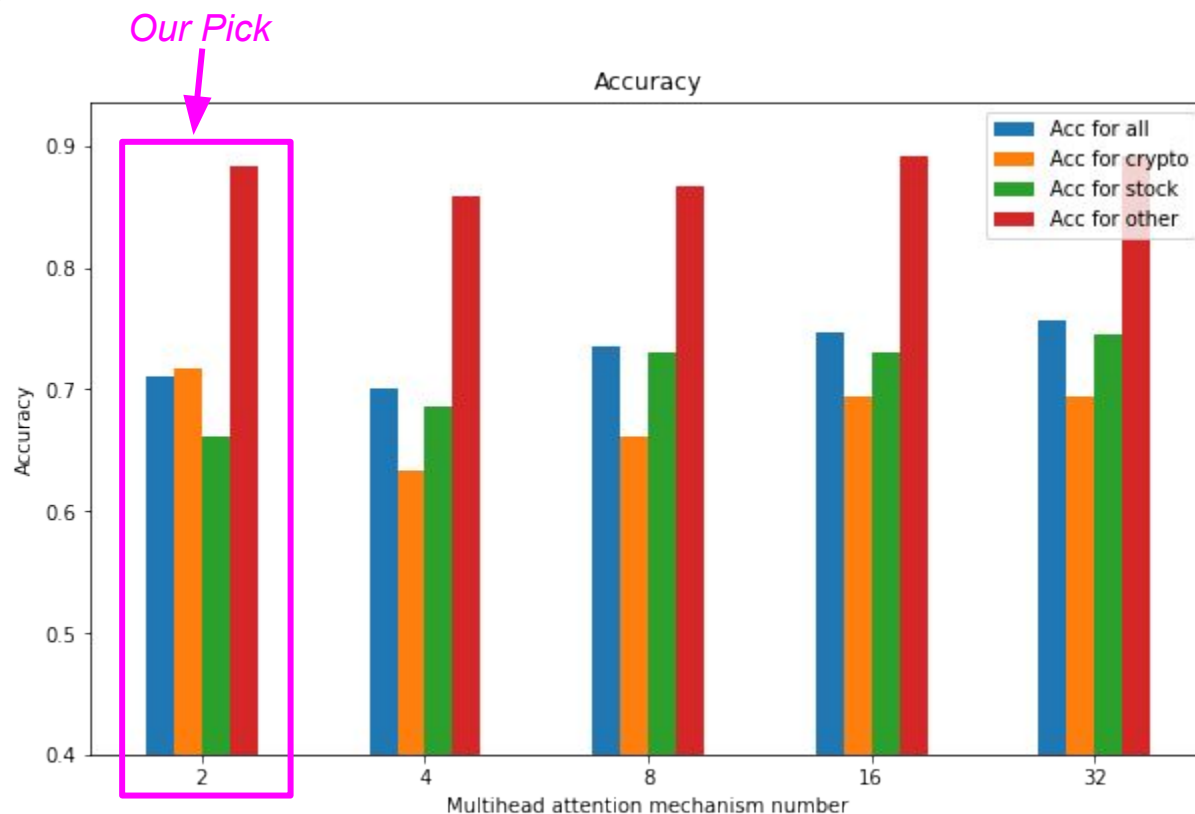
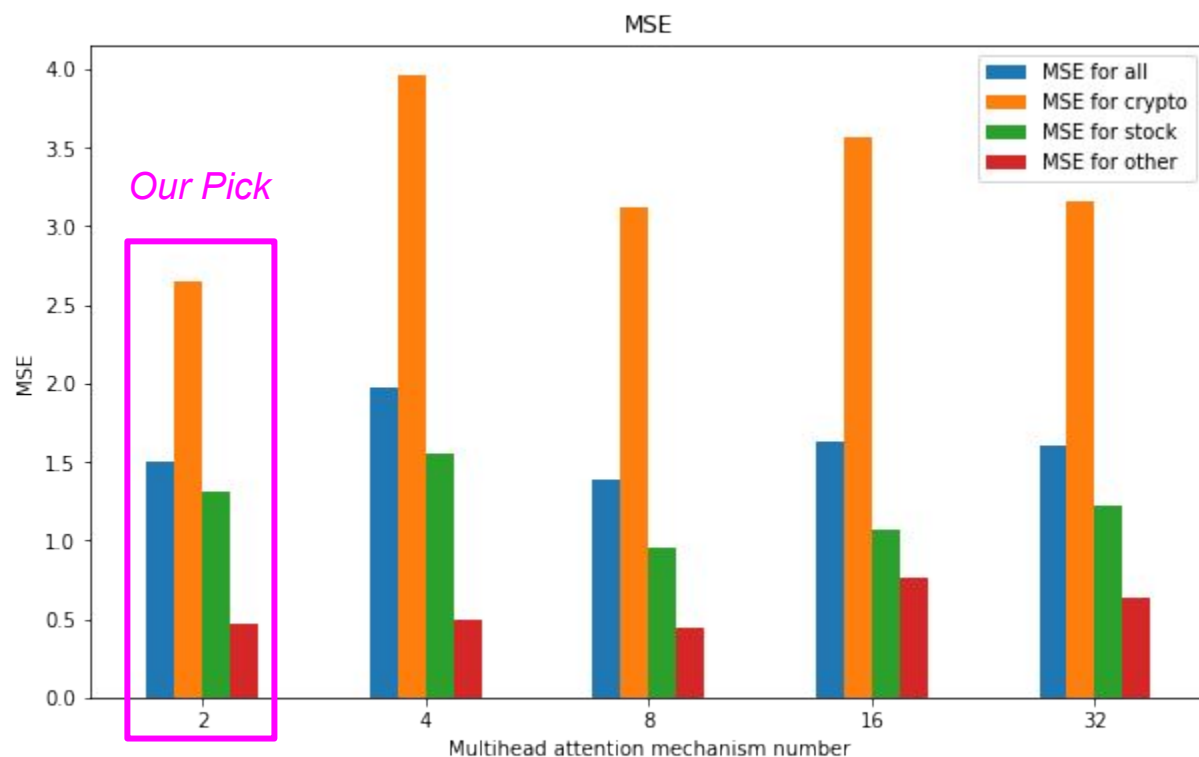
# Hyperparameter Optimization: O

**O**: Output feature size before the GAT's last layer, **Best O = 8**



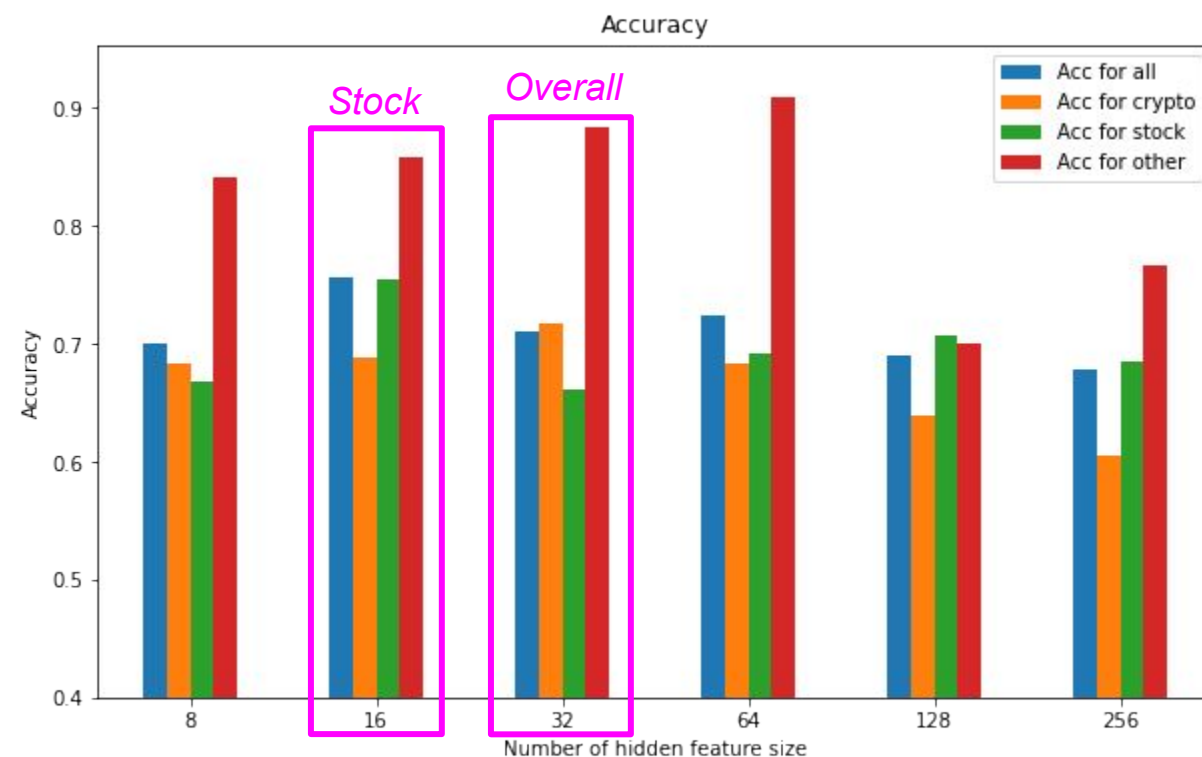
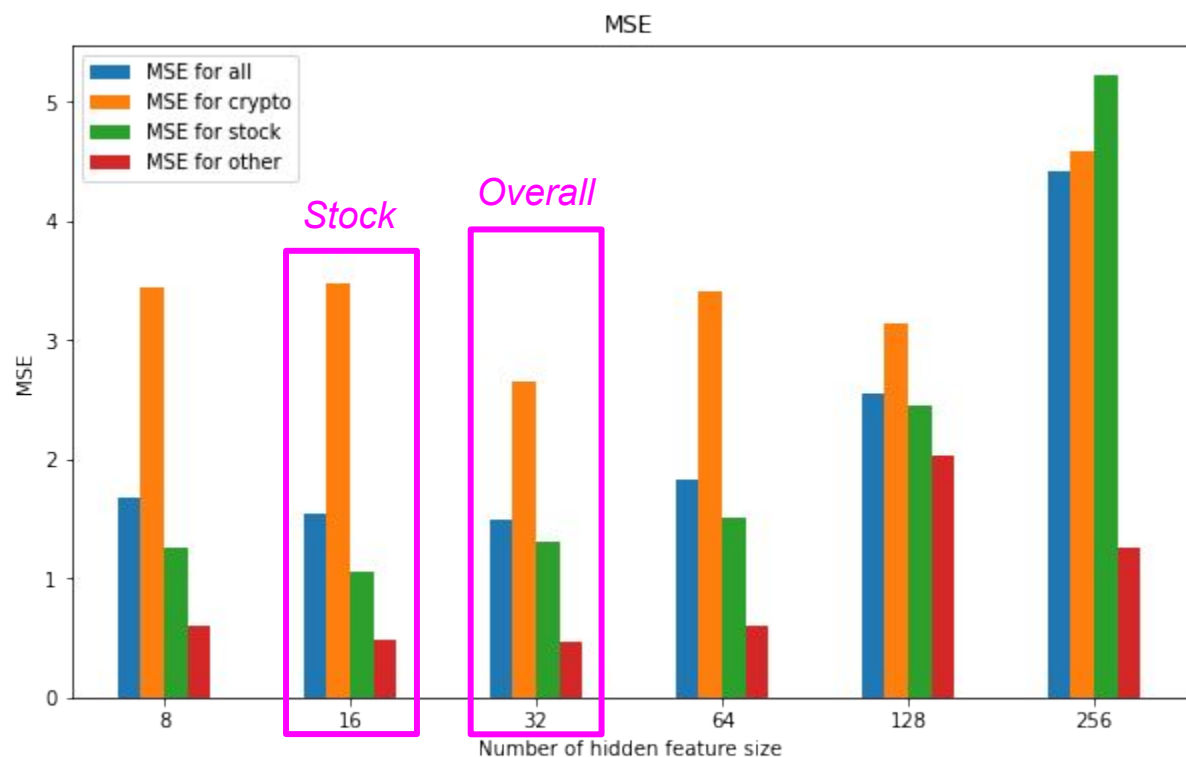
# Hyperparameter Optimization: H

**H:** Attention mechanism number (heads number), **Best H = 2**



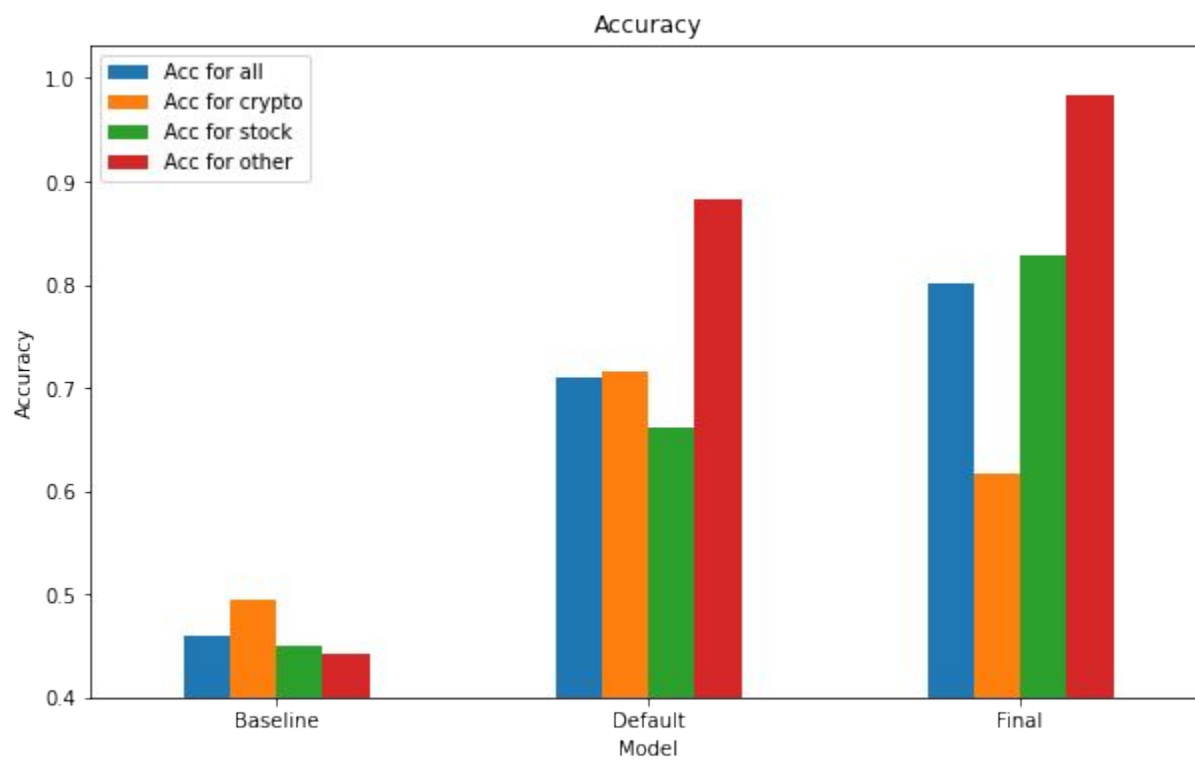
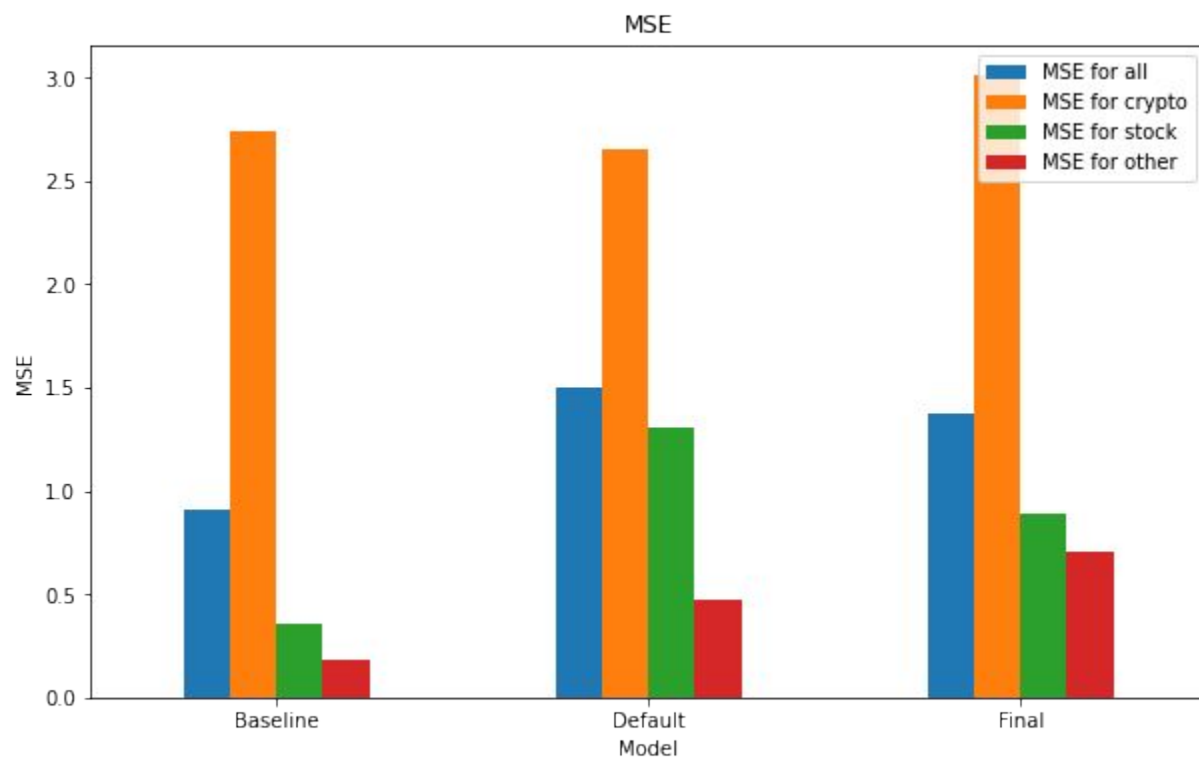
# Hyperparameter Optimization: L

**L**: Number of hidden feature size in LSTM, **Best L for stocks = 16**; **Best L for crypto and others = 32**



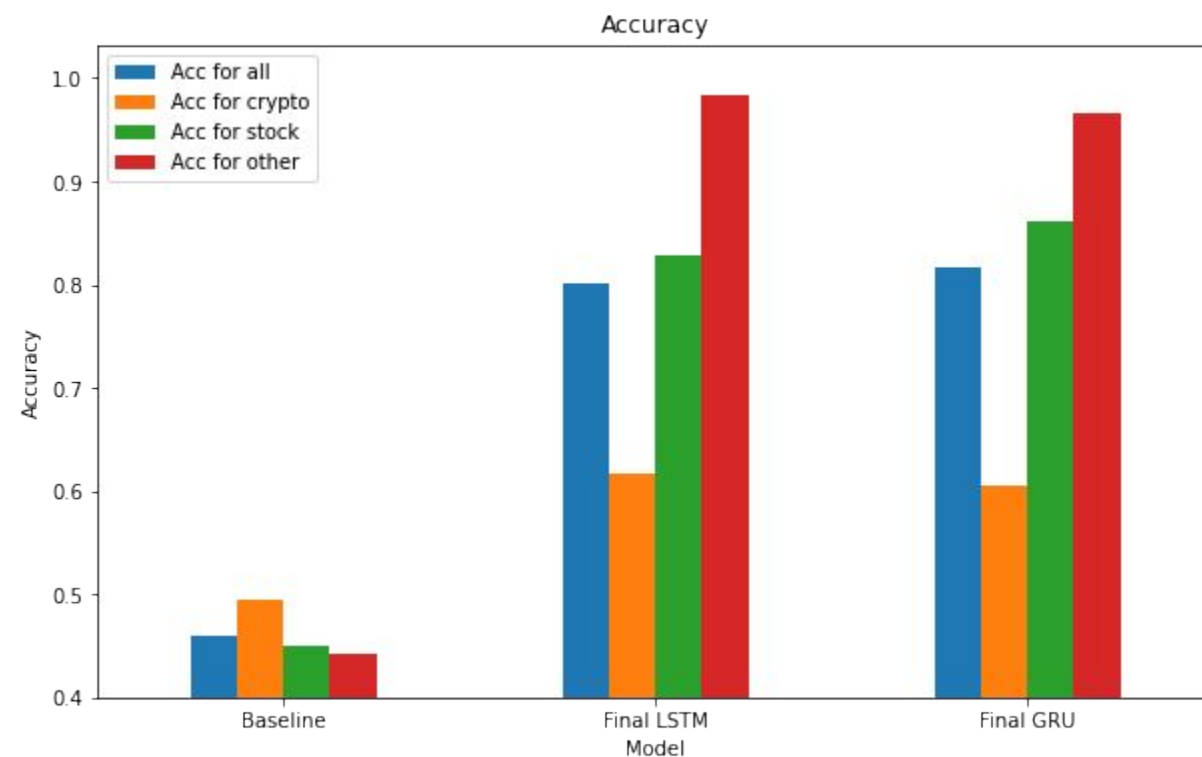
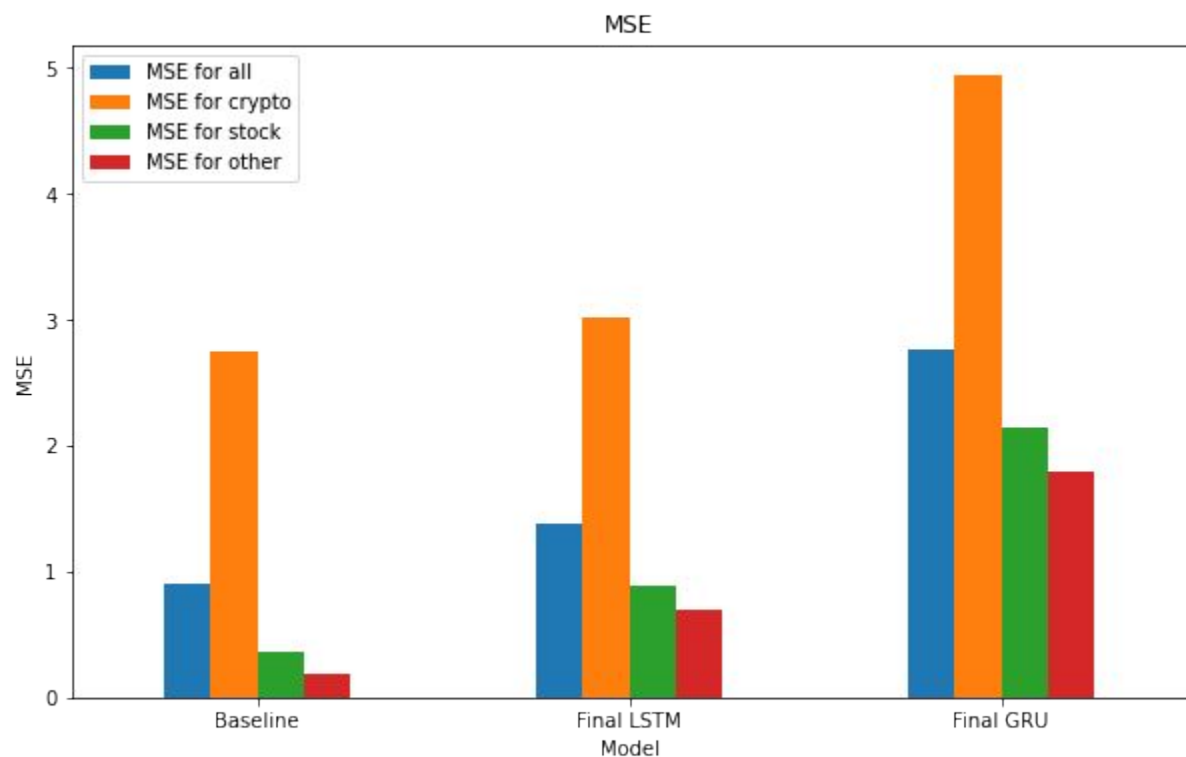
# Final Result: GAT&LSTM

$N = 6$ ,  $F = \text{Close ROI}$ ,  $O = 8$ ,  $H = 2$ ,  $L$  for stocks = 16; Best  $L$  for crypto and others = 32



# Final Result: GAT&GRU

$N = 6$ ,  $F = \text{Close ROI}$ ,  $O = 8$ ,  $H = 2$ ,  $L$  for stocks = 16; Best  $L$  for crypto and others = 32



# Result: Attention Coefficient

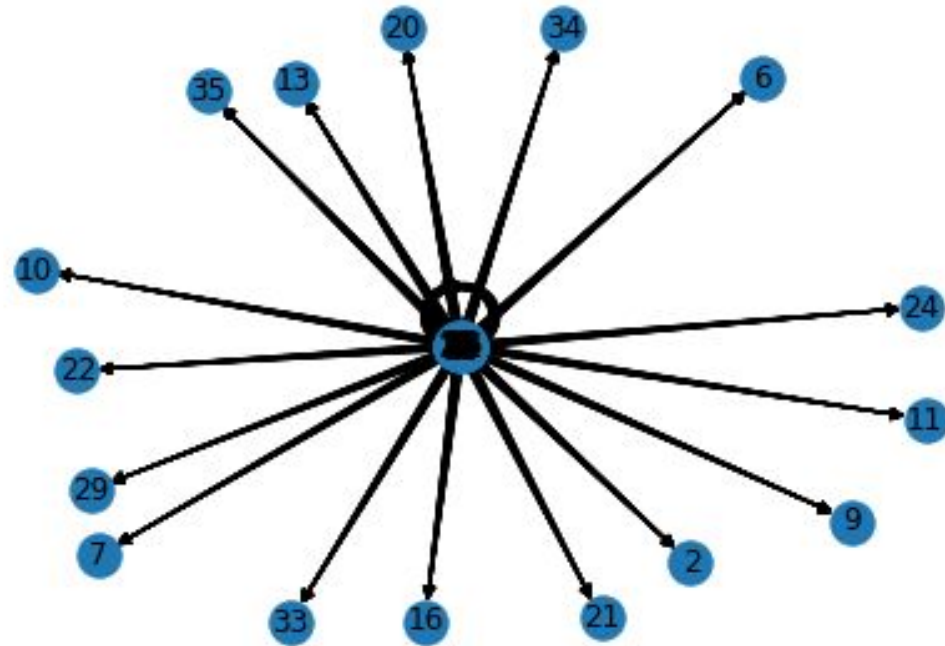


Fig. 9: Attention Coefficient Visualization  
[Generated by Yong]

AAPL (Week 80)  
Weight > 0.027

**BTC**  
**ETH**  
**AMZN**  
**GOOGL**  
**PYPL**  
**ARKK**  
**SQ**

# Risk 1: Interaction between Hyperparameters

- Five hyperparameter settings in our proposed model:

Parameters	Type
<b>N</b> : Using previous N weeks to predict the next week	General
<b>F</b> : Feature inputted into GAT	General
<b>O</b> : Output feature size before the GAT's last layer	GAT
<b>H</b> : Attention mechanism number (heads)	GAT
<b>L</b> : Number of hidden feature size in LSTM	LSTM

## Risk 2: Running time

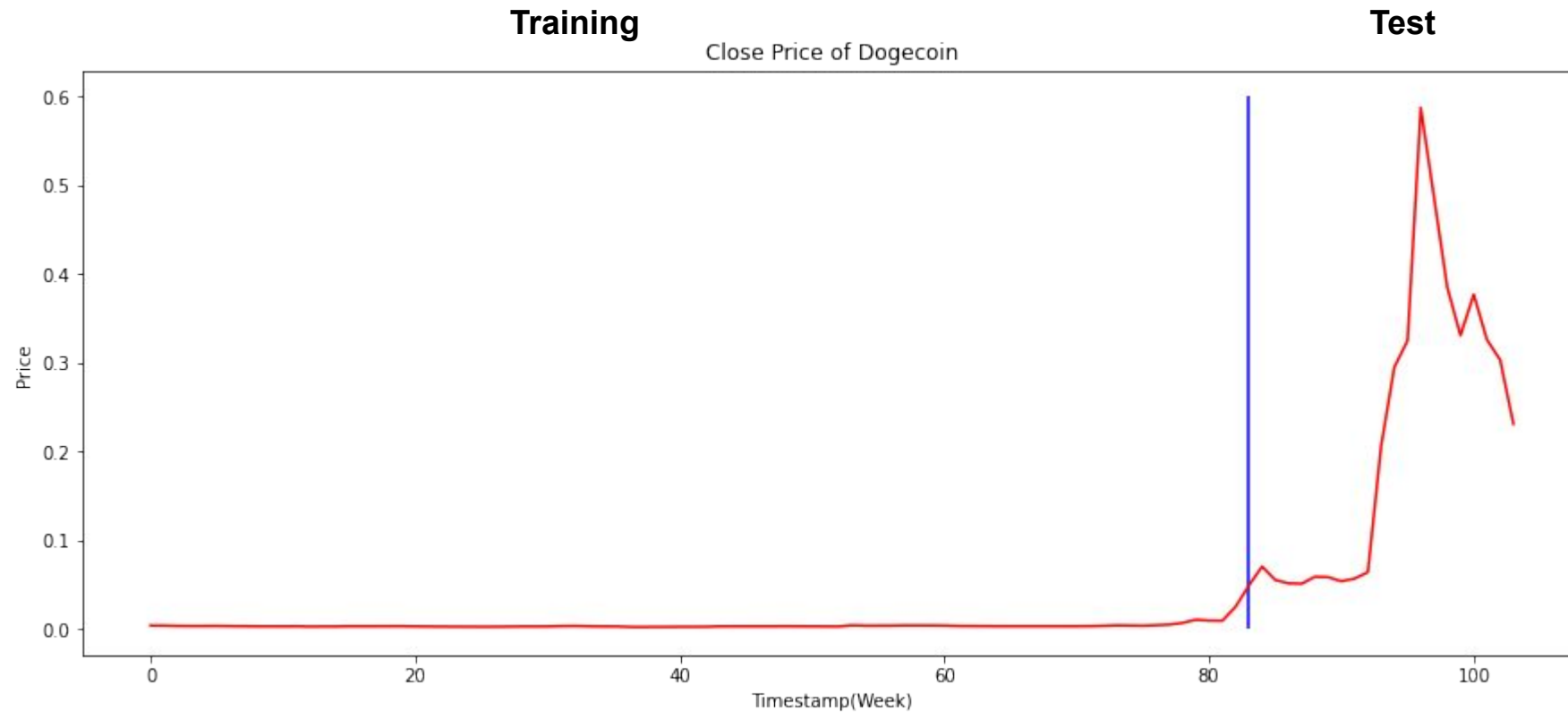
AMD Ryzen 9 5900X + Nvidia GeForce RTX 3080  
On Cuda

Model	Running time
Baseline - LSTM	Less than 1 minutes
GAT+LSTM	45 minutes

More than 100 runs conducted during hyperparameter tuning.



# Risk 3: Different variance between train and test



# Reflections

We learnt ...

Several neural network models which we can't learn from class

We are happy with ...

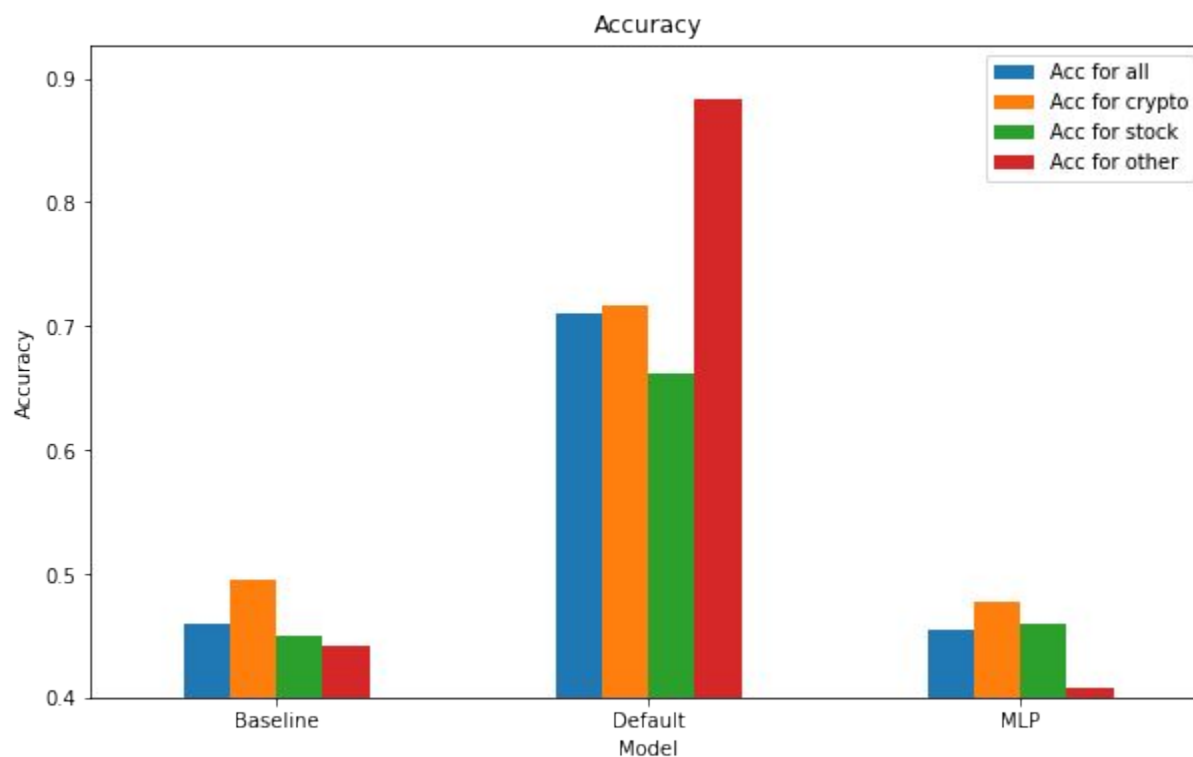
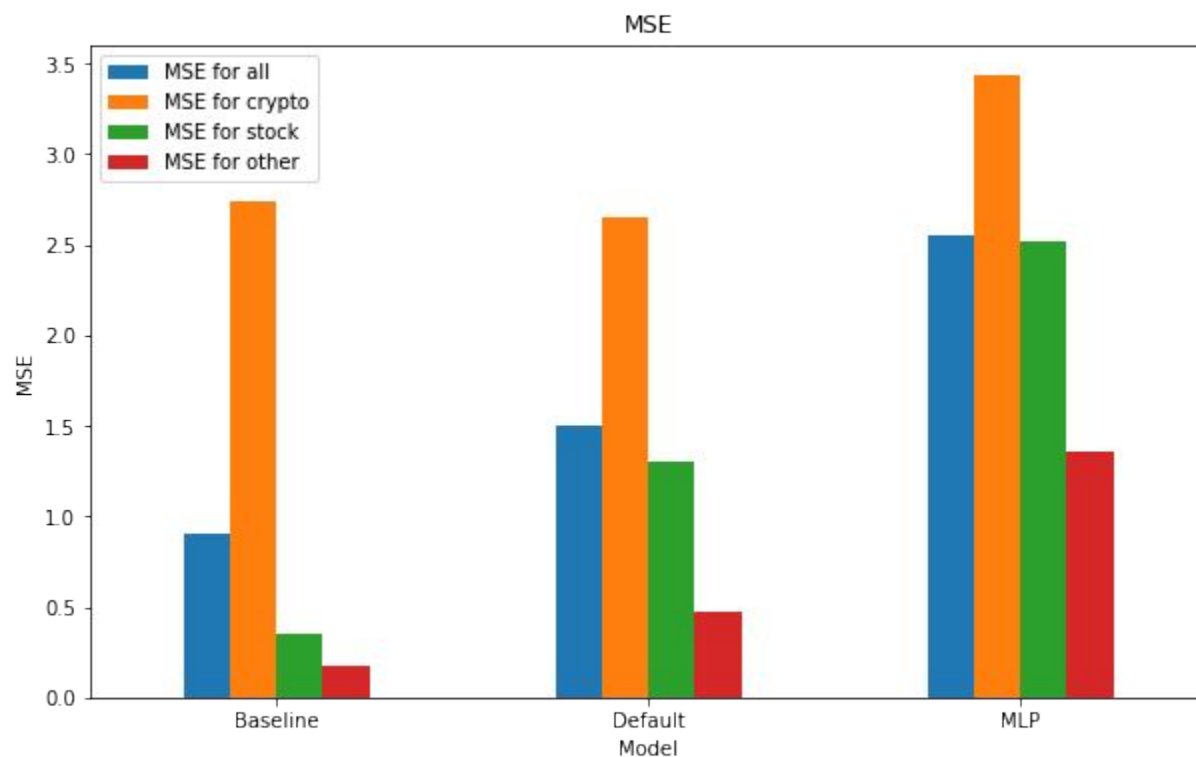
Successfully implemented GAT, LSTM and GRU with a decent accuracy.

We want to do better ...

Improve the model for cryptocurrencies.

# Reflections: Multi-Layer Perceptron (MLP)

MSE and Accuracy of MLP model with Baseline model and Default model.



# Future works

We wish to improve our results by...

1. Tune the best model hyperparameter
2. Utilise Multi Layer Perceptron
3. Separately build specific models for cryptocurrencies and stocks
4. Test our model with out-of-sample data

# Conclusion

- Started with a interest in Finance & new ML algorithms
- **Focus:** GAT - Time insensitive
- LSTM & GRU Modules - Enables Time Series Data
- Latent Transform - MLP

Our findings:

- GAT-LSTM Hybrid model - useful for **trend** prediction
- Baseline LSTM - **46%** Accuracy
- Default GAT-LSTM - **70%** Accuracy
- Tuned GAT-LSTM - **80%** Accuracy

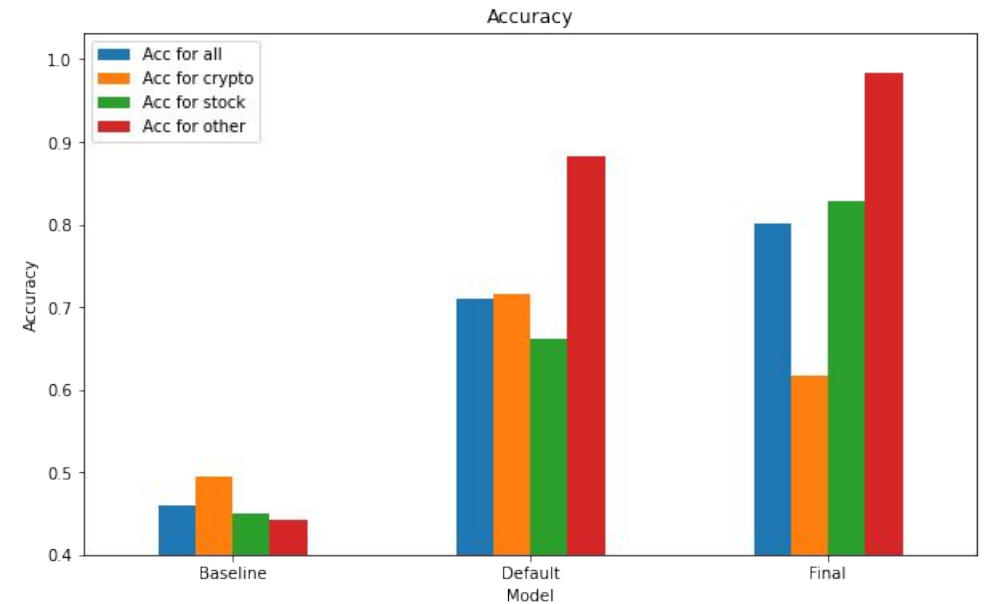


Fig. 10: Baseline vs GAT-LSTM Accuracy Comparison  
[Generated by Mark]

# Group member contribution

Name	Team	Main Task	Contribution
Henry Wu	Data Engineer	Data Retrieval + Hyperparameter Optimization	20%
Kam Leong Ao	Data Engineer	Data Retrieval + Data Engineering	20%
Mark Chen	Machine Learning	GAT + LSTM Components + Baseline Models	20%
Yong Yu	Machine Learning	GAT + GRU Components + Attention Coefficients	20%
Yifei Liu	Machine Learning	GAT + MLP Component + Overall project direction	20%

# Reference / GitHub

[1] Stock Market Prediction Using Ensemble of Graph Theory, Machine Learning and Deep Learning Models - Pratik Patil et al.

<https://doi.org/10.1145/3378936.3378972>

[2] An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition - Chenyang Si et al.

<https://arxiv.org/pdf/1902.09130.pdf>

(Concepts were learnt and applied from the above papers, all code was written from scratch by referencing PyTorch Documentation by our ML team)

Our Github:

- [https://github.com/ProfiterolePuff/DL\\_Stock\\_Prediction](https://github.com/ProfiterolePuff/DL_Stock_Prediction)

Thank you for your <sub>(graph)</sub> attention!

Questions / Comments?



# Hidden slides past this point - not used

Please ignore all slides past this point

# Reflections - kam

- what worked well and what didn't
- what could be improved
  - Reflection: Talk about issues encountered and how you approached them; include reflections on what worked well, what could be improved, what you learned while working on this project (what you would have done different, if you had chance to redo the whole project.)
- Things we went well:
  - Proactive & punctual attitude
  - Bold try on new technique (GAT)
- Things that we need to improve:
  - Take much care on lecturers' feedbacks

# High-level Project Summary

- big picture summary
- started thinking x, ran into y, ended up at z
- The presentation should be self-contained. Briefly introduce your project goal, i.e. include the research problem, objectives, and used methodology. Furthermore, include a clear experimental design and used datasets, new results (all results, highlighting the improvements you did from the previous presentation), main conclusions and reflection on the overall project and achievements.

# Previous feedback address - yifei

- P2: Risks considered
- P2: Loss function described
- 
- Big picture of model described
- Experimental setup - data split - param settings - baselines - eval metrics
- Overview of experiments
- Baseline solutions comparisons added
- Risks considered
- References cited correctly

# Questions from Henry -- after intro...?

1. Experimental design:
  - a. GAT not time aware
  - b. combine GAT and LSTM? (tune parameters) (methodology present)
  - c. Baseline = LSTM? (Yes)
2. Results:
  - a. Compare result from i) GAT+LSTM vs ii) LSTM only, how much we improved, what is accuracy.
3. Extension: GRU and others? (did we use GRU in LSTM?)
4. Risk
5. Reflection
6. Conclusion

# Experimental design - DE

- data split
- Explicitly state the experimental design and evaluation, e.g., training/validation/test split. You should provide details on model performance measures, how the parameters and hyper-parameters are tuned, what computational resources are used, the training time and the convergence of your model if applicable. Results should be presented in clean, processed format that can be digested easily. Avoid snapshots of terminal outputs, large tables. The listener, whether is the teacher or the a fellow student, has limited capacity to extract relevant information in short period of time. Remove information on the slides that you do not plan to comment or explain, but key information relevant for understanding the methodology/results has to be there. Abbreviations have to be introduced, and used only if saves you space and avoid repetitions.
- 
- 
- The package of Pytorch 1.9.1 version with CPU compute platform is implemented
- 37 datasets including 22 stocks, 9 Cryptocurrencies, and 6 other commodities
- Daily datasets are obtained between the period of 1st July, 2019 to 30th June, 2021
- Period data completeness as our main consideration
- Date, Open, High, Low, Close are the five features of our project
- Those datasets split 80/20 to train and evaluate the model effectiveness
-

# Experimental design - DE

- Baseline model: Only LSTM model implementation
- Deploy the same dataset and the same LSTM model setting , but epochs = 200
- Compared with our proposed hybrid model to examine the effectiveness (MSE and Accuracy)

Parameter Settings	Baseline Model
Input Dimension	1 (talk about input feature)
Hidden Dimension	32
Number of Layers	2
Input Dimension	1
Epoch	200
Learning Rate	0.001

# Experimental design - mark / yong

- param settings
- mlp/gru/lstm/gat
- diff lstm/gru for each class
-



# Experimental design - mark/yong

- evaluation metrics

# why and how it works - yong

- high level recap of all experimental design
- justify choices
- how each step plays into next and compliment each other
- show entire workflow
-

# results - yong/mark

- show proof that it works
- something interesting about attention coefficients
- compare against last week!
- Again, it should be clear what you have done and what already was done. In this sense, please be explicit on what results are taken from the literature (including online resources) if you wish to compare your method against, and what are your results as a result of novel and/or fine-tuned analysis. It is important that you report (and motivate) any changes since the proposal talk. We expect you to comment on the given feedback in the last presentation

# Long Short-Term Memory

## Cell state

Transfers relative information through the whole sequence

## Forget gate

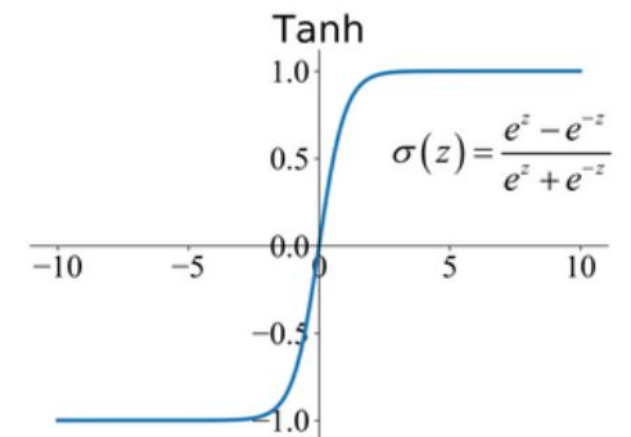
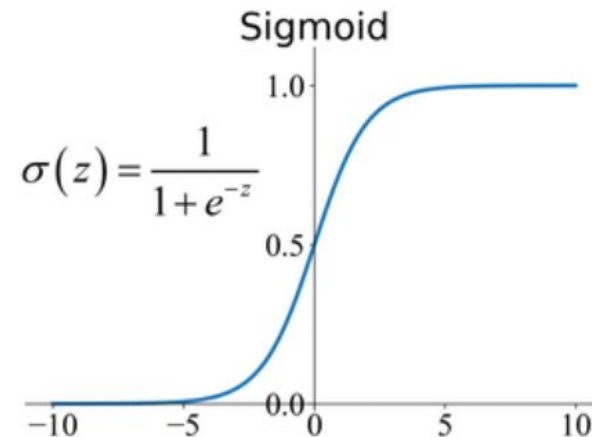
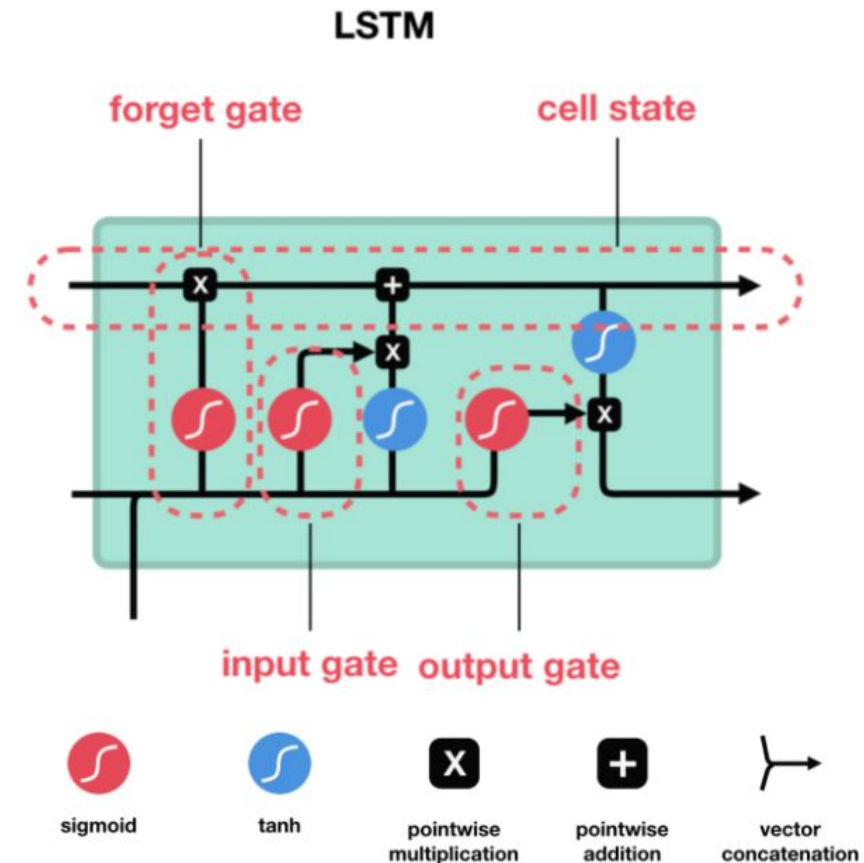
Decides what information should be kept or throw away

## Input gate

Add current sequence information to the cell state

## Output gate

Decides what the next hidden state



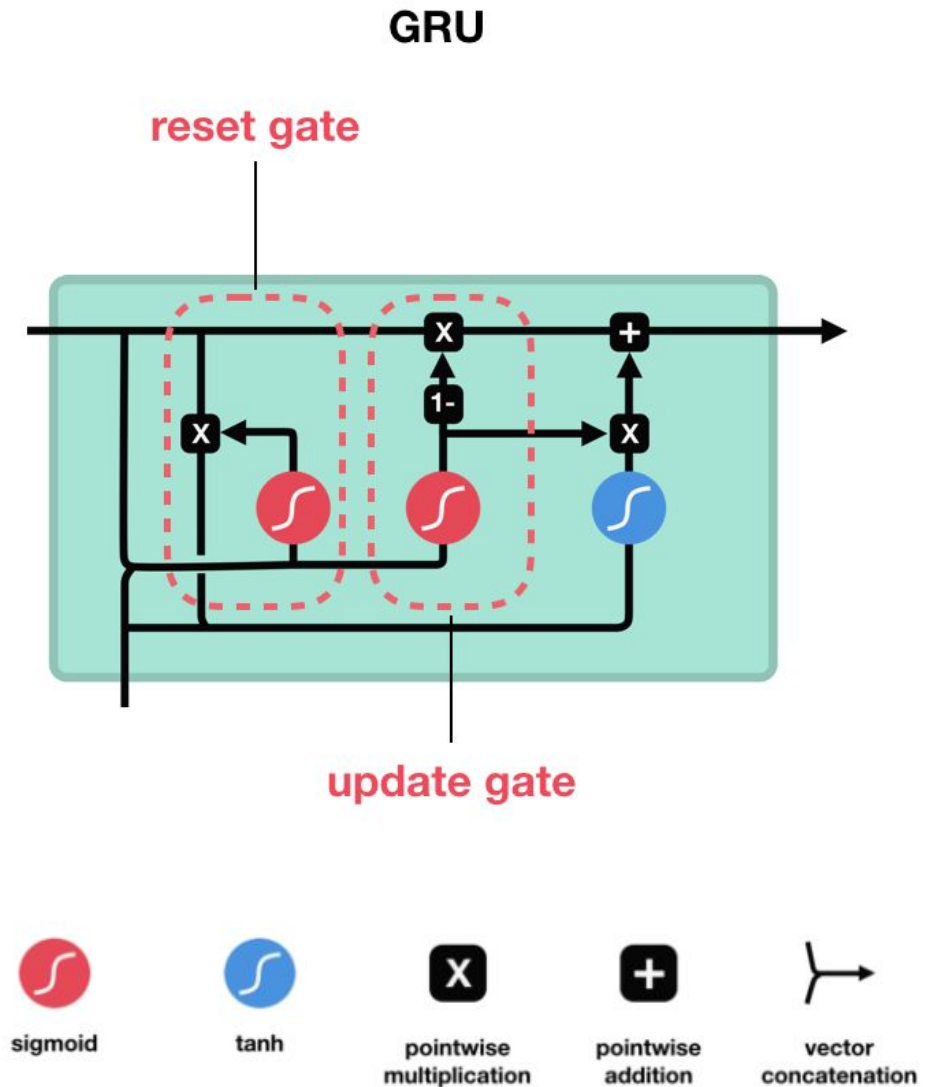
# Gated Recurrent Units

## Reset gate

Decide how much past information to forget

## Update gate

Decides what information need to throw away and what new information need to add



# Full Model 1: Preprocessing

Replace missing values with previous available value

Turn daily transactions into weekly

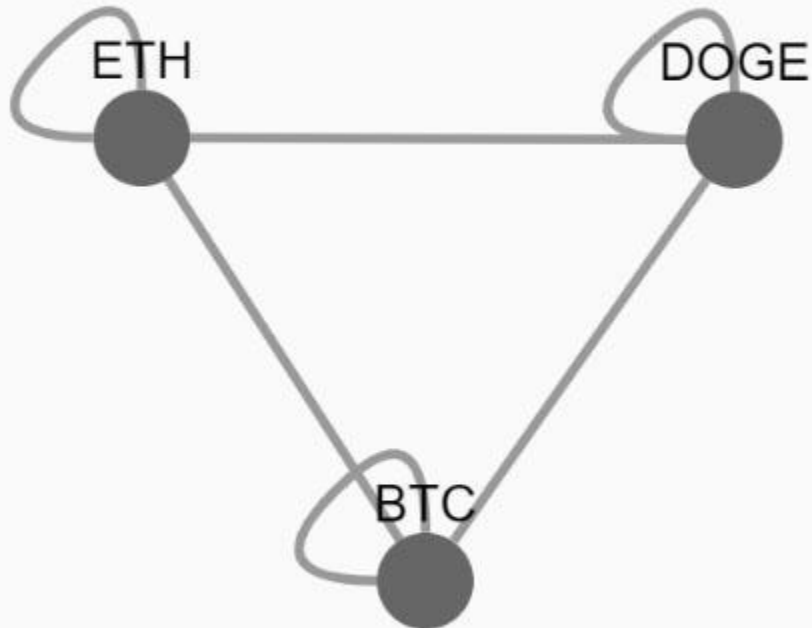
drop all transactions which are not Monday

Normalization on all features

MinMaxScaler to normalize all features into range  $(-1, 1)$

# Full Model 2: Input Graph

For each timestamp (e.g. W1, W2, etc.)



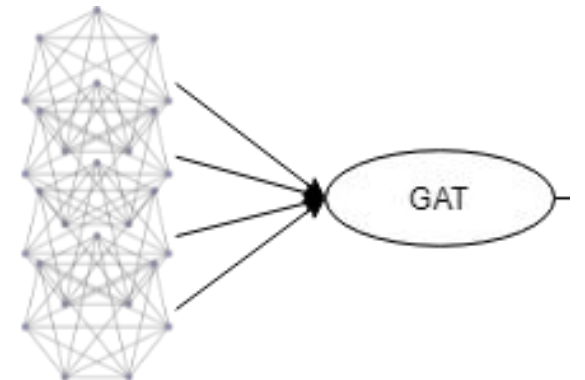
Feature vector before input:

BTC: Close, Open, High, Low, etc.

ETH: Close, Open, High, Low, etc.

DOGE: Close, Open, High, Low, etc.

Date	Open	High	Low	Close
7/1/2019	46.042	46.62	45.256	45.434
7/2/2019	45.778	45.83	44.444	44.91
7/3/2019	47.878	48.314	46.902	46.98
7/5/2019	46.914	47.09	46.16	46.62
7/8/2019	46.248	46.45	45.732	46.068
7/9/2019	45.794	46.2	45.456	46.012
7/10/2019	46.83	47.788	46.628	47.784
7/11/2019	47.628	48.3	47.16	47.72
7/12/2019	47.95	49.076	47.942	49.016
7/15/2019	49.6	50.884	48.972	50.7



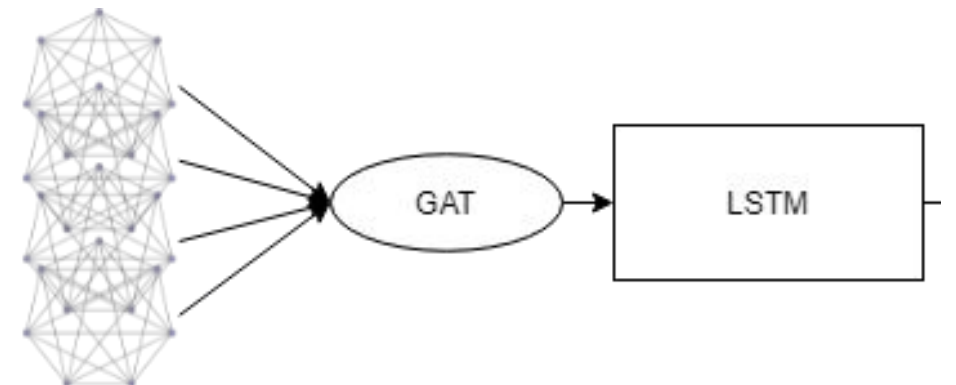
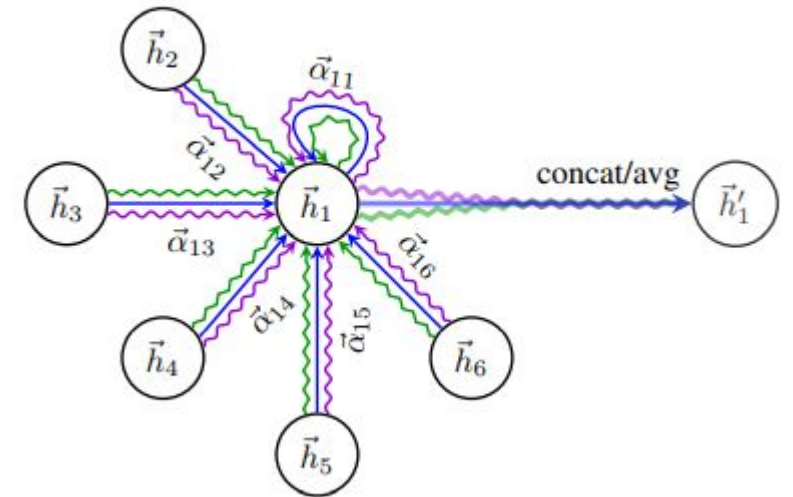
# Full Model 3: GAT

Universal GAT model takes the graph input

Extracting relationship between crypto/stocks/commodities

Embed feature vector of nodes through attention mechanism

Pass the output of GAT to LSTM





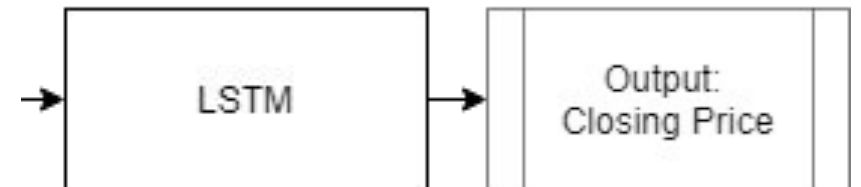
# Full Model 4: LSTM and Final Output

Dealing with sequences (Time Series)

Uses every five weeks' embedded features as training data

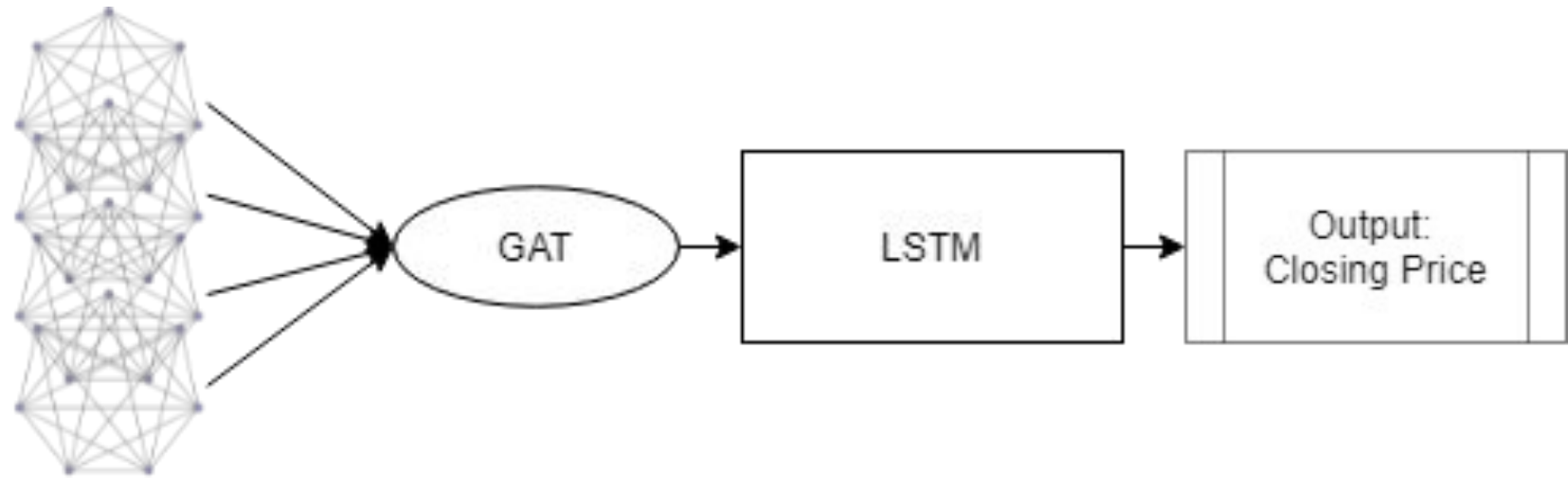
Predicts the next week's close price

Output Prediction

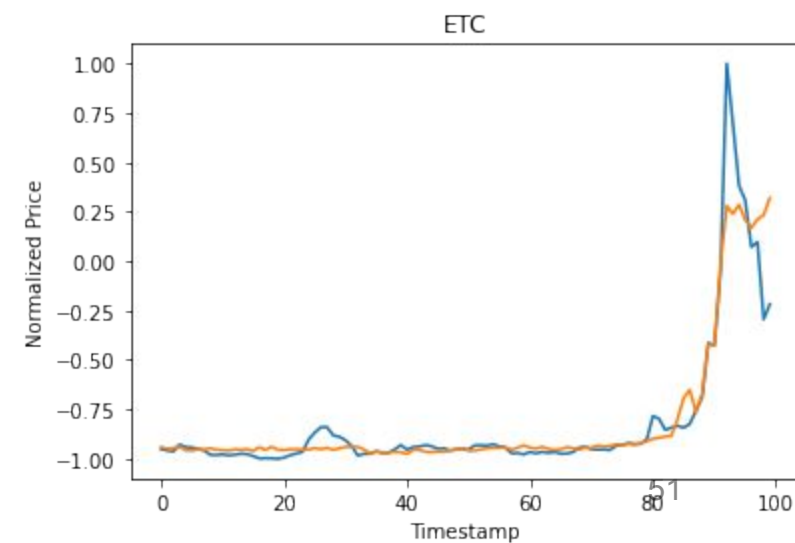
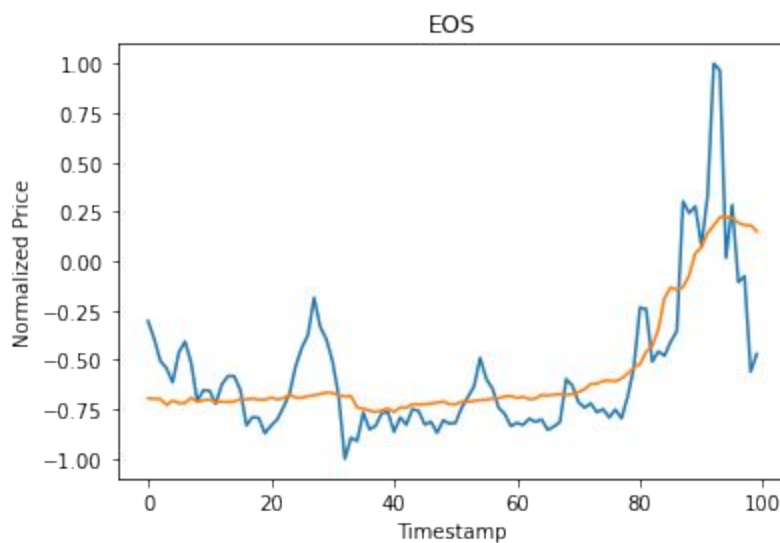
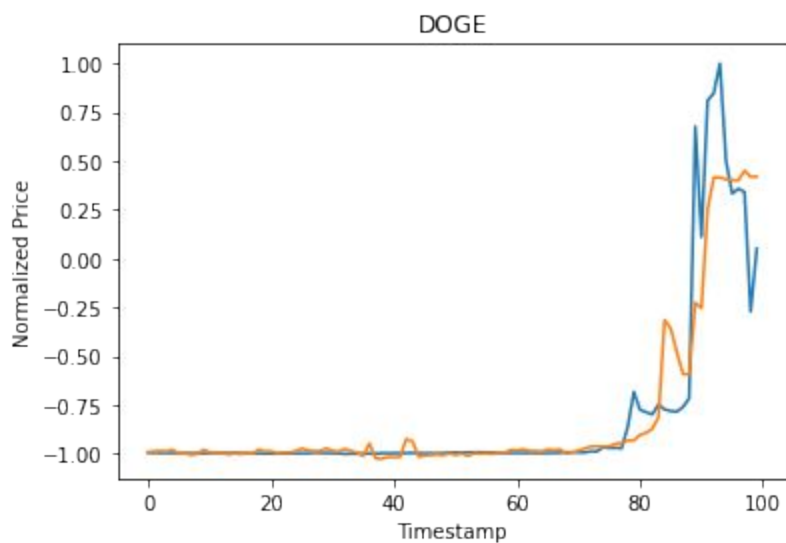
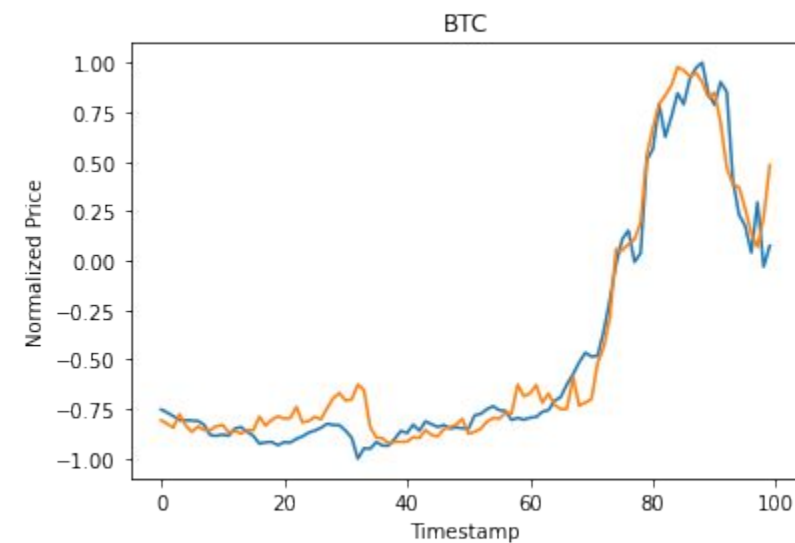
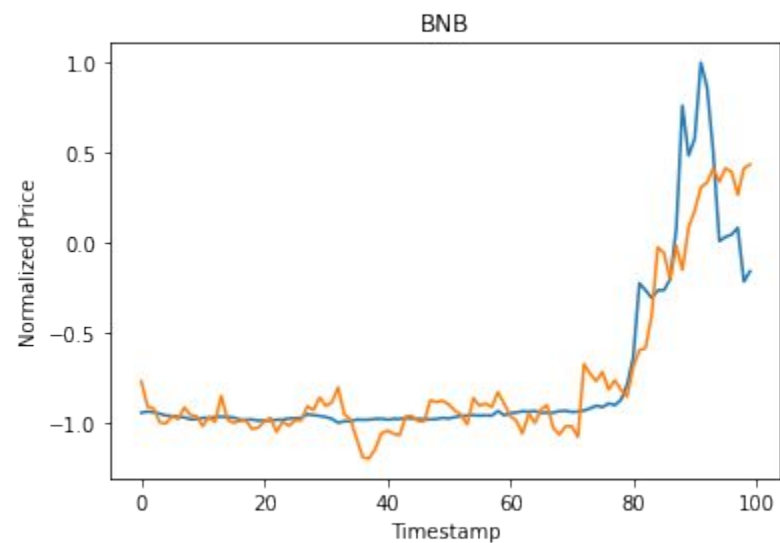
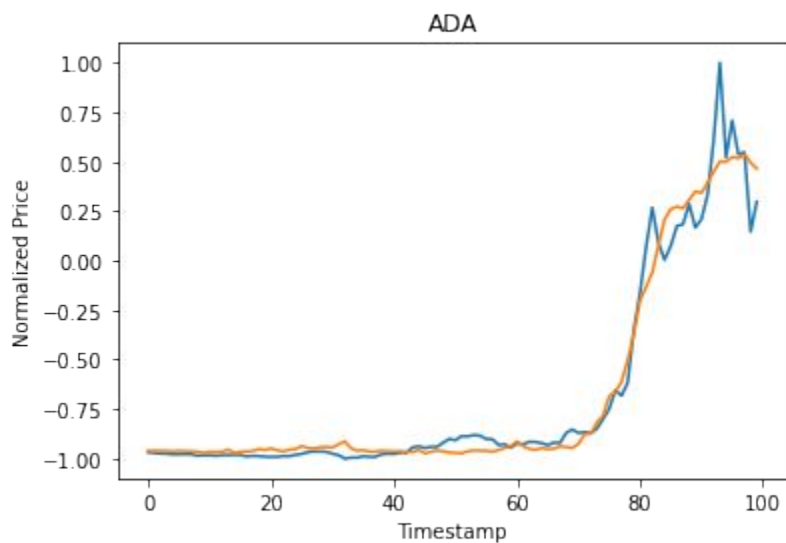


# Overview of entire workflow

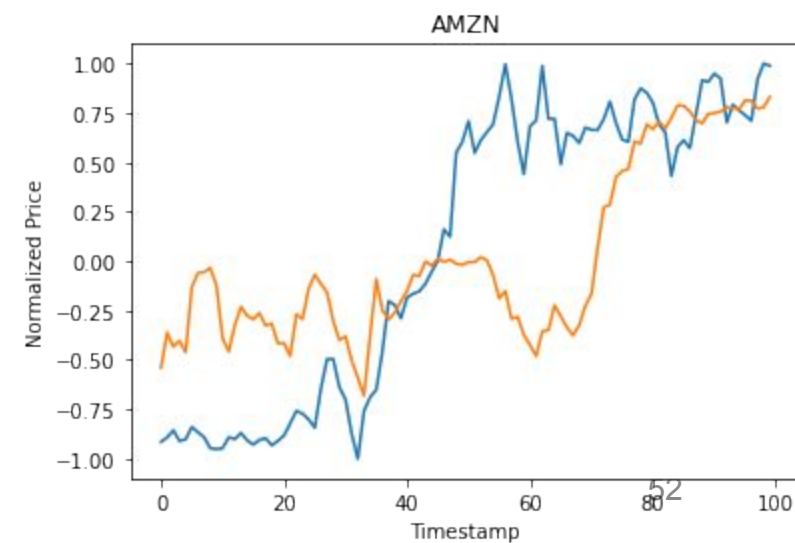
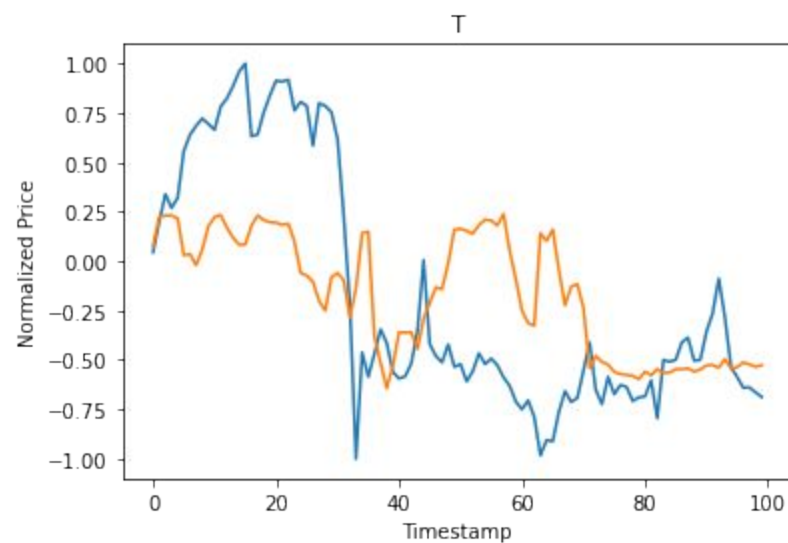
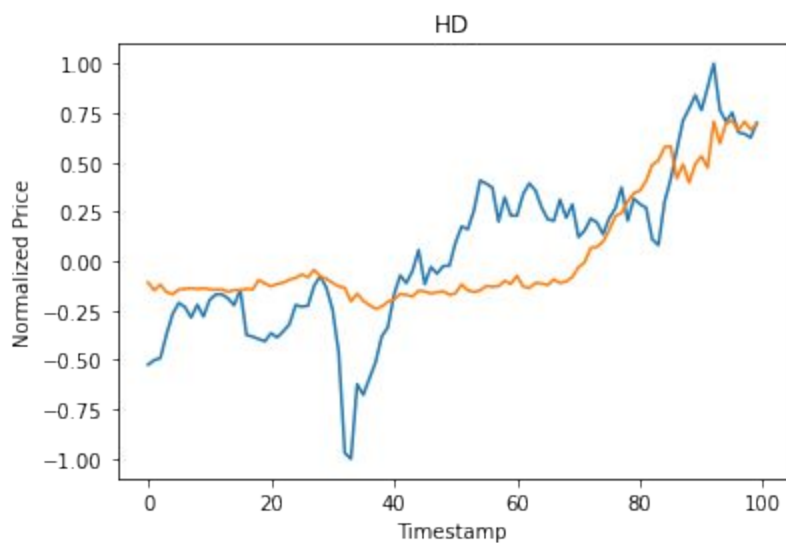
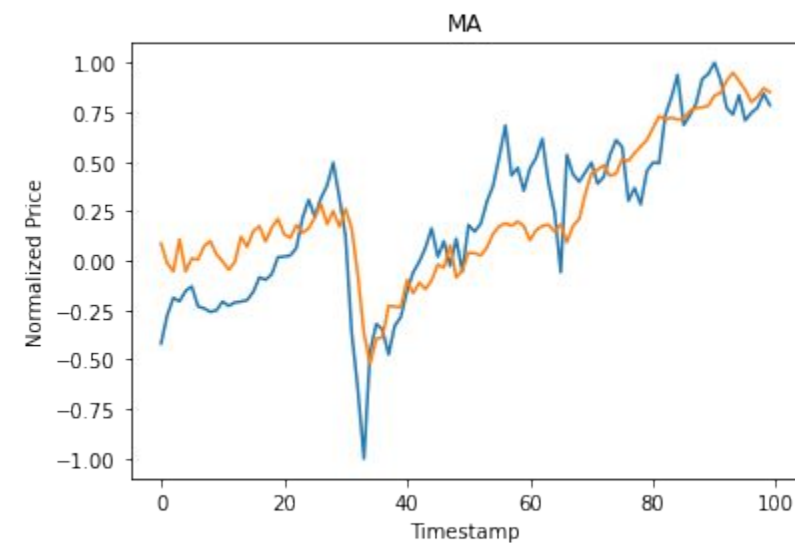
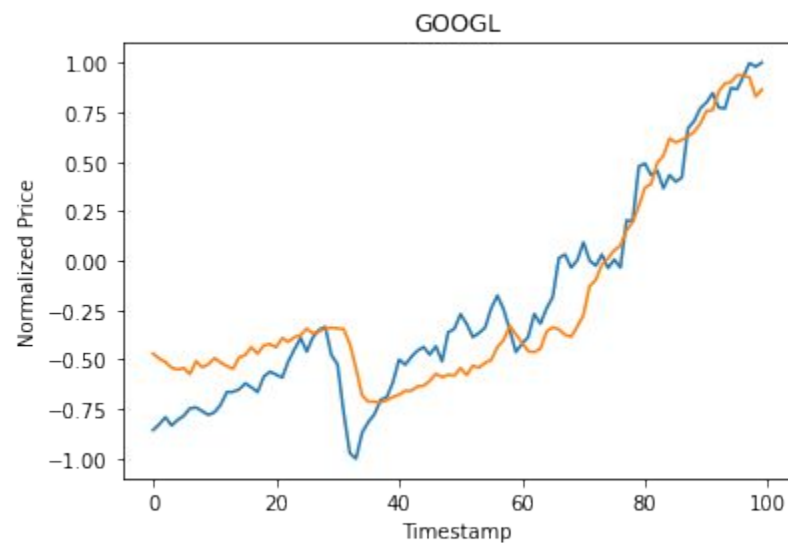
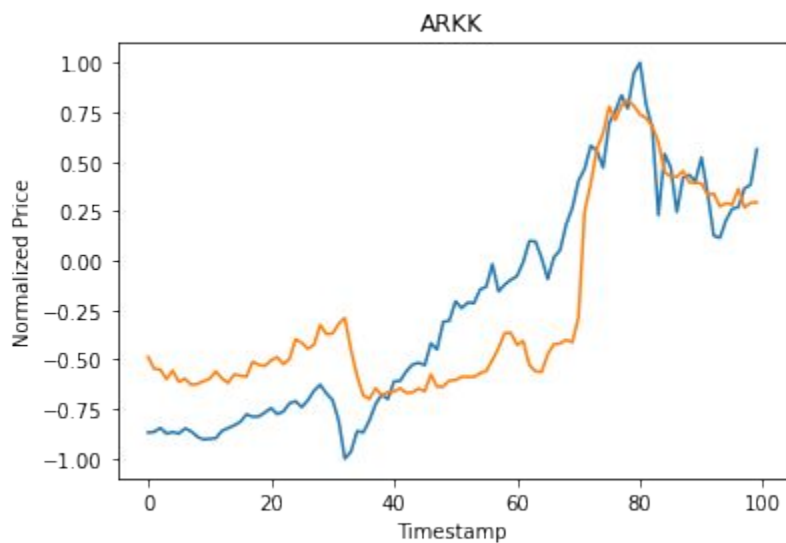
Date	Open	High	Low	Close
7/1/2019	46.042	46.62	45.256	45.434
7/2/2019	45.778	45.83	44.444	44.91
7/3/2019	47.878	48.314	46.902	46.98
7/5/2019	46.914	47.09	46.16	46.62
7/8/2019	46.248	46.45	45.732	46.068
7/9/2019	45.794	46.2	45.456	46.012
7/10/2019	46.83	47.788	46.628	47.784
7/11/2019	47.628	48.3	47.16	47.72
7/12/2019	47.95	49.076	47.942	49.016
7/15/2019	49.6	50.884	48.972	50.7



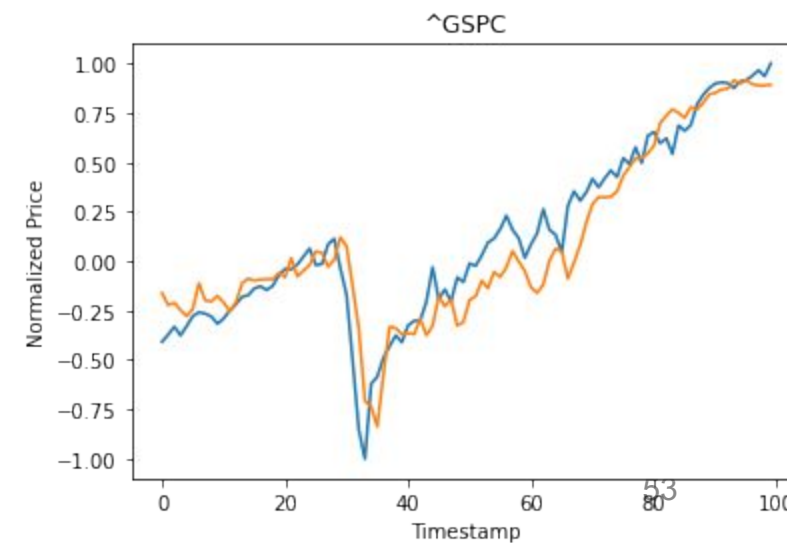
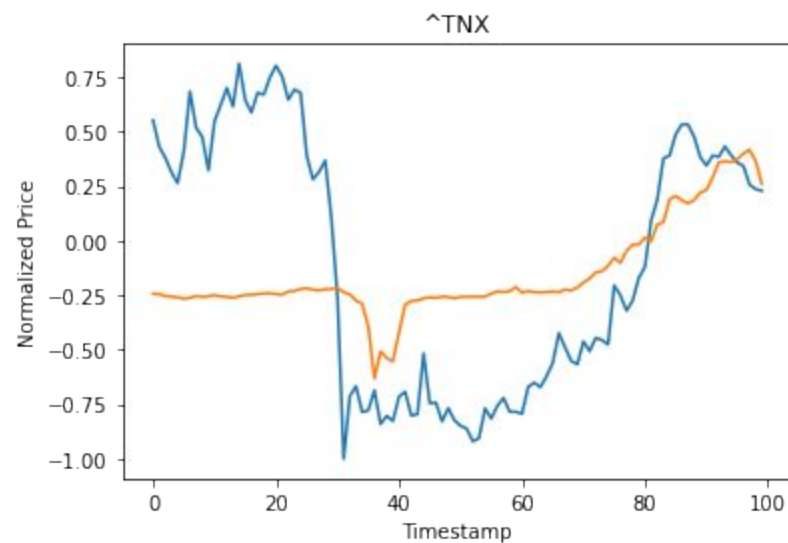
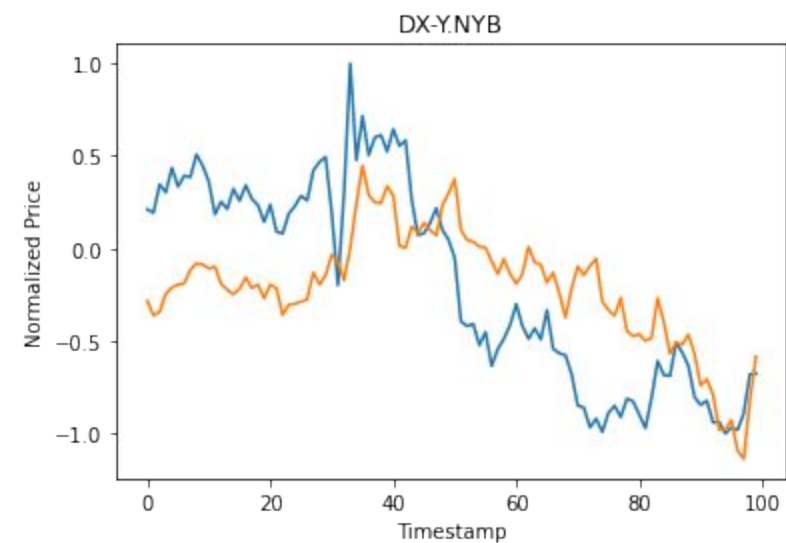
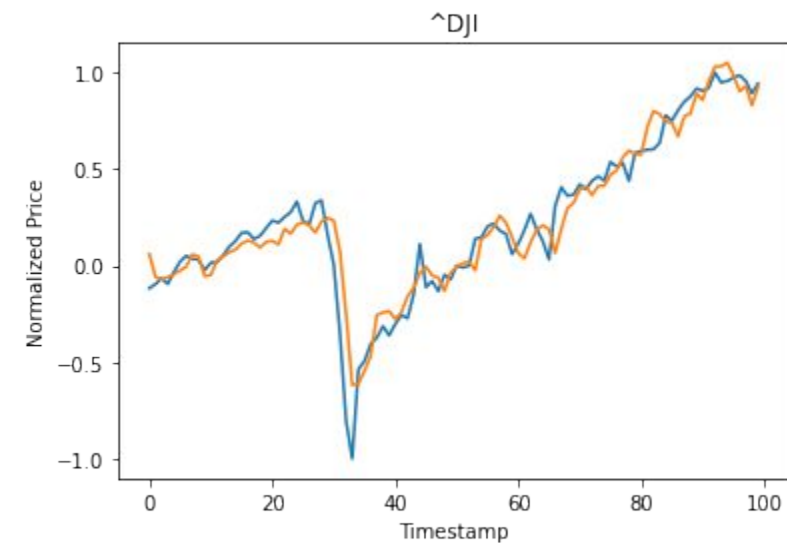
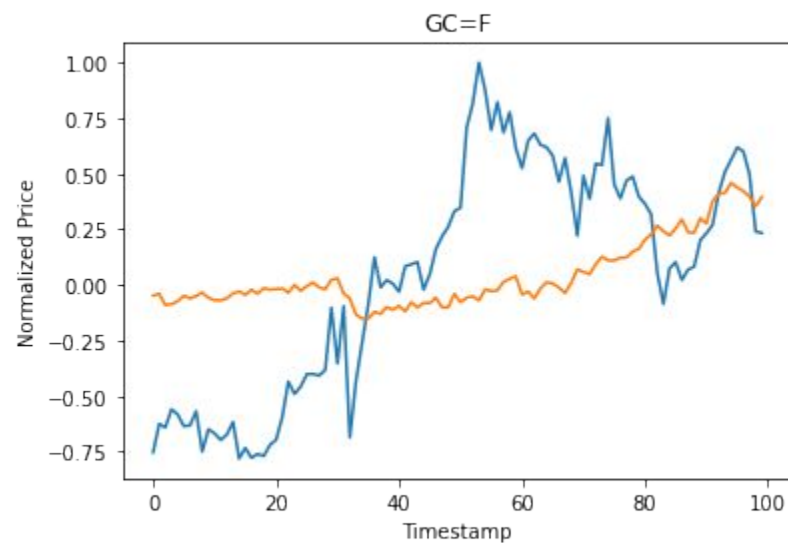
# Result: Crypto



# Result: Stock



# Result: Commodity and Index



# Conclusions

- Changes and refinements in this stage
  - Since the GAT is not time-aware, additional component, e.g. LSTM, enable GAT to work with time-series data
  - normalise the price changes
- Low performance on predicting fluctuate commodities as one of our model disadvantages

# Next Steps

## 1. Enhance the model :

- Develop LSTM/GRU models for every asset class for a more precise result.
- Transform the features to latent space and observe effect on model performance - MLP.

## 2. Result :

- Observe the relations of attention coefficient of every graph to understand the relationship between the nodes.
- Compare the performance of GAT model with traditional models.

# Timeline Assessment

- Very close to our schedule :)
- What's coming...

Week 9 & 10	Finalize our experiment in coming two weeks <ul style="list-style-type: none"><li>• Add LSTM models</li><li>• Performance comparisons</li><li>• Final troubleshootings</li></ul>
Week 11 & 12	Final presentation + Final report



# Reference / GitHub

- Graph Attention Networks - Petar Veličković et al.  
<https://arxiv.org/abs/1710.10903>
- An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition - Chenyang Si et al.  
<https://arxiv.org/pdf/1902.09130.pdf>

Our Github:

- [https://github.com/FLMAPO/DL\\_Stock\\_Prediction](https://github.com/FLMAPO/DL_Stock_Prediction)

Thank you for your <sub>(graph)</sub> attention!

Questions / Comments?

# Hidden slides past this point - not used

Please ignore all slides past this point

# Next Steps

## 1. Data :

- Screen the data on hand and explore some more meaningful/ higher **representativeness** for our experiment

## 2. Model :

- Develop three LSTM models for three different types of commodities (e.g.,stocks, indexes, and cryptocurrencies)

## 3. Result :

- Compare the performance of GAT model with traditional models
- Observe the changes of attention coefficient of every graph to understand the relationship between the nodes

# Next Steps

- Henry Kam
- experiments we can run to add explainability to the model.
  - Compare with traditional models?
- different normalization
- start this after monday kaiqi meeting
  - calculate attention coefficient of every graph and observe its changes
  - for each stock/ index/ commodity, we have different LSTM, so we will apply LSTM on each asset class in the later experiment.
- experiments with highly correlated stocks

## • Data Improvements / Dataset

- variable selection(?) ← select some useful
- actual high low xx > transform to latent features

# Old slides beyond this point

- Left these in because may be useful to copy in some for this week
-

# Methodology

## Goal

Predict return ratio

$$R = \frac{V_f - V_i}{V_i}$$

## Dataset

Three dataset: Cryptocurrency, stock, commodity (last two years)

Week as an interval - Stock not trade in weekend

Total 112 weeks, 80 weeks for training, 16 weeks for validation, and 16 weeks for testing

Every 7 weeks as a data interval, 6 weeks as training data, and the 7th week as the prediction target

$V_f$  = final value,

$V_i$  = initial value

# Graph Structure

## Features

Normalize open, high, low, close price, and volume  
Classification - Cryptocurrency, stock, commodity

## Edges

All vectors are connected

## Label

Weekly average return ratio

```
8 class CryptoDataset(Dataset):
9     def __init__(self, root, transform=None, pre_transform=None):
10         super(CryptoDataset, self).__init__(root, transform, pre_transform)
11
12     @property
13     def raw_file_names(self):
14         crypto_path = glob.glob("./" + self.root + "/*.csv")
15         crypto_filenames = []
16         for crypto_file in crypto_path:
17             name = os.path.basename(crypto_file)
18             crypto_filenames.append(name)
19         return crypto_filenames
20
21     @property
22     def processed_file_names(self):
23         data_rows = pd.read_csv(self.raw_paths[0]).shape[0]
24         return [f"data_{i}.pt" for i in range(data_rows)]
25
26     def download(self):
27         pass
28
29     def process(self):
30         file_no = 0
31         data_rows = pd.read_csv(self.raw_paths[0]).shape[0]
32         for i in range(data_rows):
33             data_time = 0
34             all_node_feats=[]
35             label_y=[]
36             for raw_path in self.raw_paths:
37                 # load data from 'raw_path'
38                 data_file = pd.read_csv(raw_path)
39                 if (data_time == 0):
40                     date_time=data_file.iloc[0]["Date"]
41                     daily_data = data_file.loc[data_file["Date"] == date_time]
42                     if (daily_data.shape[0]>0):
43                         node_feats = self._get_node_features(daily_data)
44                         all_node_feats.append(node_feats)
45                         label_y.append(self._buy_or_not(data_file,date_time))
46                     edge_index = self.generateEdge(len(all_node_feats))
47                     all_node_feats = torch.tensor(all_node_feats, dtype=torch.float)
48                     label_y = torch.tensor(label_y, dtype=torch.long)
49                     mask = 'train_mask' if i <= data_rows * 0.8 else 'val_mask' if i <= data_rows * 0.9 else 'test_mask'
50                     data = Data(x=all_node_feats, edge_index=edge_index, y=label_y, mask=mask)
51
52                     if self.pre_filter is not None and not self.pre_filter(data):
53                         continue
54
55                     if self.pre_transform is not None:
56                         data = self.pre_transform(data)
57
58                     torch.save(data, os.path.join(self.processed_dir, 'data_{:02d}.pt'.format(file_no)))
59                     file_no += 1
60
61     def len(self):
62         return len(self.processed_file_names)
63
64     def get(self, idx):
65         data = torch.load(os.path.join(self.processed_dir, 'data_{:02d}.pt'.format(idx)))
66         return data
67
68     def _get_node_features(self, daily_data):
69         all_node_feats = []
70         all_node_feats.append(daily_data["Open"].item())
71         all_node_feats.append(daily_data["High"].item())
72         all_node_feats.append(daily_data["Low"].item())
73         all_node_feats.append(daily_data["Close"].item())
74         all_node_feats.append(daily_data["Volume"].item())
75         return all_node_feats
76
77     def _get_return_ratio(self, daily_data):
78         return_ratio = []
79         data = np.array(daily_data["Close"])
80         for i in range(len(data)):
81             if i == 0:
82                 return_ratio.append(0)
83             else:
84                 return_ratio.append((data[i]-data[i-1])/data[i-1])
85         return return_ratio
86
87     def _buy_or_not(self, data_file, date_time):
88         buyOrNot=0
89         data = np.array(data_file["Close"])
90         data_row = data_file.loc[data_file["Date"] == date_time]
91         if data_row.shape[0]>0:
92             index = data_file.loc[data_file["Date"] == date_time].index[0]
93             if (index < data_file.shape[0]-1 and data[index+1]>data[index]):
94                 buyOrNot=1
95         return buyOrNot
96
97     def generateEdge(self, n):
98         edges = []
99         for i in range(n):
100             for j in range(n):
101                 edges.append([i, j])
102         return torch.tensor(np.transpose(edges), dtype=torch.long)
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



# Evaluation

- Rank the ground-truth top-K weekly return ratio of all stocks.
- Predicted weekly return ratio rank top-K
- Mean Reciprocal Rank

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$

# Timeline Assessment

- Slightly behind scheduled timeline.
- Stage 1 and 2 completed, skeleton code fully functional up until training function, still need to smooth out a few things before training phase.
- ML Team branching off with different approaches to solve the above problem.

```
<ipython-input-8-6f2f4d81e4f1> in train(train_dataset)
    50     print(out)
    51     loss = F.nll_loss(out, data.y)
→   52     loss.backward()
    53     optimizer.step()
    54

C:\Anaconda\lib\site-packages\torch\tensor.py in backward(self, gradient, retain_graph, create_graph, inputs)
    253         create_graph=create_graph,
    254         inputs=inputs)
→   255         torch.autograd.backward(self, gradient, retain_graph, create_graph, inputs=inputs)
    256
    257     def register_hook(self, hook):

C:\Anaconda\lib\site-packages\torch\autograd\__init__.py in backward(tensors, grad_tensors, retain_graph, create_graph, grad_variables, inputs)
    145         retain_graph = create_graph
    146
→   147         Variable._execution_engine.run_backward(
    148             tensors, grad_tensors, retain_graph, create_graph, inputs,
    149             allow_unreachable=True, accumulate_grad=True) # allow_unreachable flag

RuntimeError: CUDA error: CUBLAS_STATUS_ALLOC_FAILED when calling `cublasCreate(handle)`
```

```
Traceback (most recent call last):
  File "price_embedding.py", line 13, in <module>
    from ge import Struc2Vec
ImportError: cannot import name 'Struc2Vec' from 'ge' (/Users/yliu523/opt/anaconda3/lib/python3.8/site-packages/ge/__init__.py)
(base) yliu523@Firstling-Air code % pip install ge
Requirement already satisfied: ge in /Users/yliu523/opt/anaconda3/lib/python3.8/site-packages (0.0.2)
(base) yliu523@Firstling-Air code % pip install struc2vec
ERROR: Could not find a version that satisfies the requirement struc2vec (from versions: none)
ERROR: No matching distribution found for struc2vec
(base) yliu523@Firstling-Air code % python price_embedding.py
Traceback (most recent call last):
  File "price_embedding.py", line 49, in <module>
    [struc2vec_embedding((f, Dim, PI)) for f in os.listdir(vg_dir)]
  File "price_embedding.py", line 49, in <listcomp>
    [struc2vec_embedding((f, Dim, PI)) for f in os.listdir(vg_dir)]
  File "price_embedding.py", line 36, in struc2vec_embedding
    model = Struc2Vec(G, walk_length=10, num_walks=80, workers=40, verbose=40, stay_prob=0.3, opt1_reduce_len=True, opt2_reduce_sim_calc=True, opt3_num_layers=N, temp_path='./temp_struc2vec_%s/' % PI, reuse=False) #init model
NameError: name 'Struc2Vec' is not defined
(base) yliu523@Firstling-Air code %
```

# Reference / GitHub

- Stanford GraphNN Slides - <http://web.stanford.edu/class/cs224w/slides/03-nodeemb.pdf>
- GAT Infographic 1&2:  
<https://www.youtube.com/watch?v=A-yKQamf2Fc>  
<https://www.youtube.com/watch?v=uF53xsT7mjc>
- Valeria Cortez, 2017. Visualising stocks correlations with Networkx
- LSTM & GRU -  
<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

Our Github:

- <https://github.com/YongYu123/760-PB3>

Thank you for your <sub>(graph)</sub> attention!

Questions / Comments?

# Comments from last presentation

1. Choosing the right graph network for the graph structure -Taken!
1. Diversify your data - Taken!
1. Investigate the robustness of the graph structure to adversarial attack.

# GAT Intro

Breakdown into phases of GAT workings

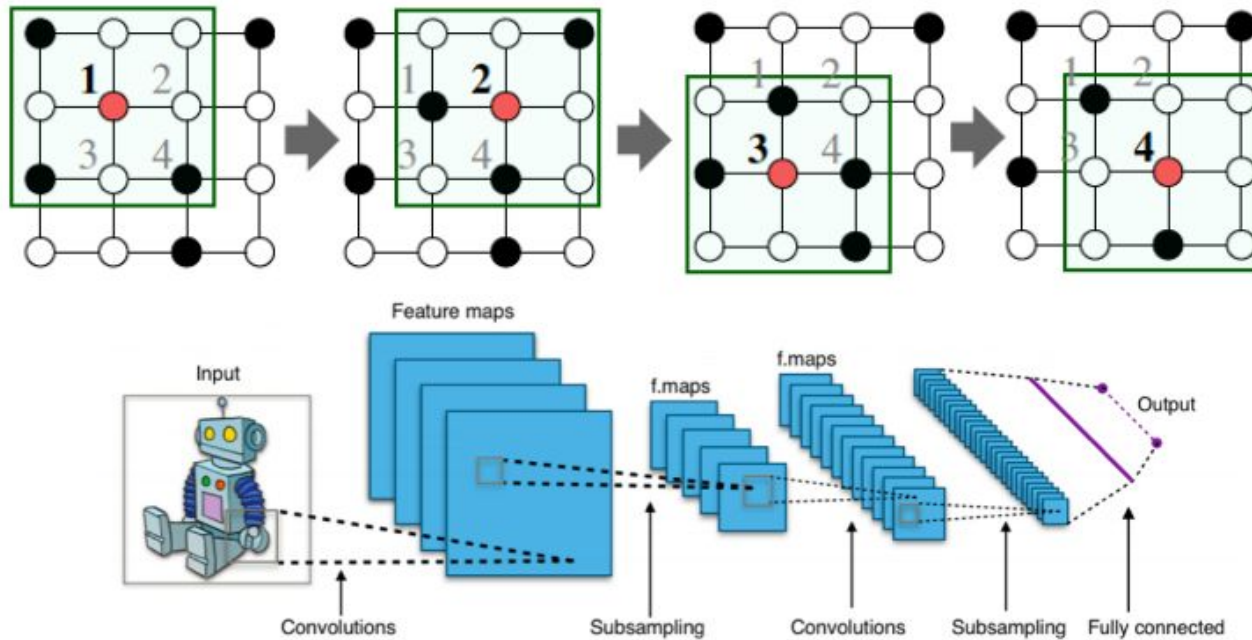
Yifei

address feedback 1 from last week

1. Pay attention on choosing the right graph network for the graph structure you will use (to support weighted graphs).

# GNN Intro - Graph Convolutional Network

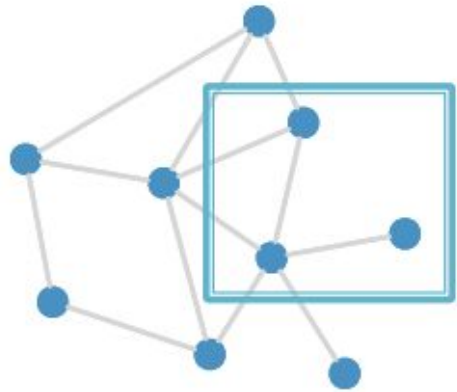
## CNN on an image:



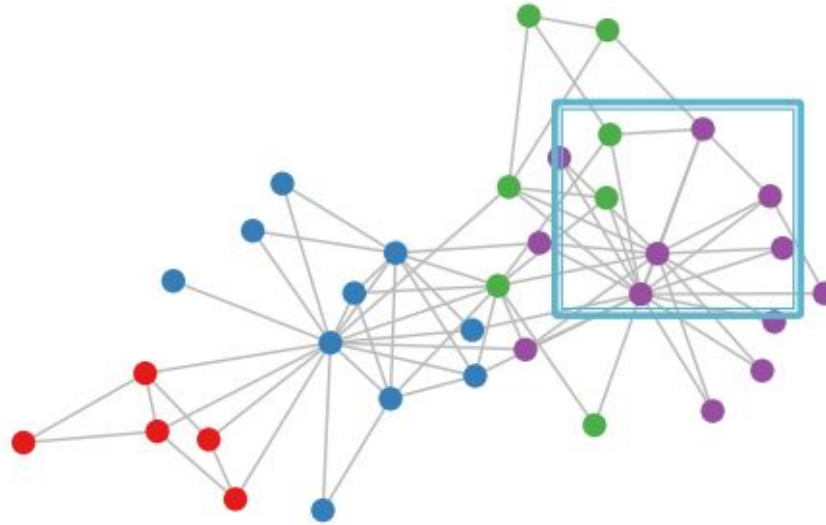
Goal is to generalize convolutions beyond simple lattices  
Leverage node features/attributes (e.g., text, images)

# GNN Intro - GCN

**But our graphs look like this:**



or this:

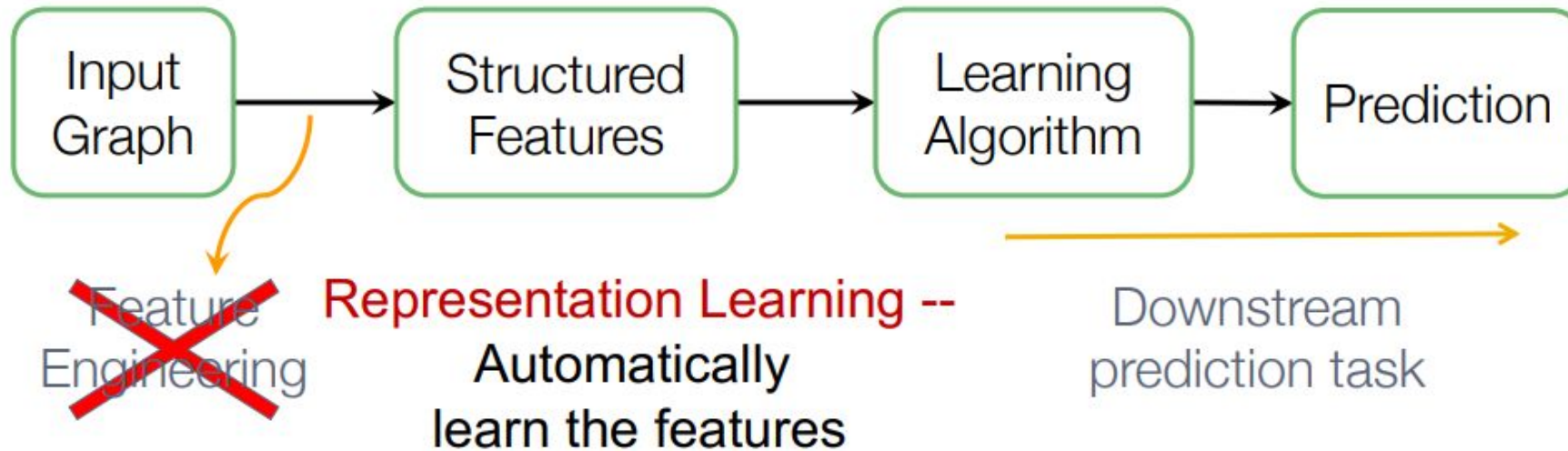


- There is no fixed notion of locality or sliding window on the graph
- Graph is permutation invariant



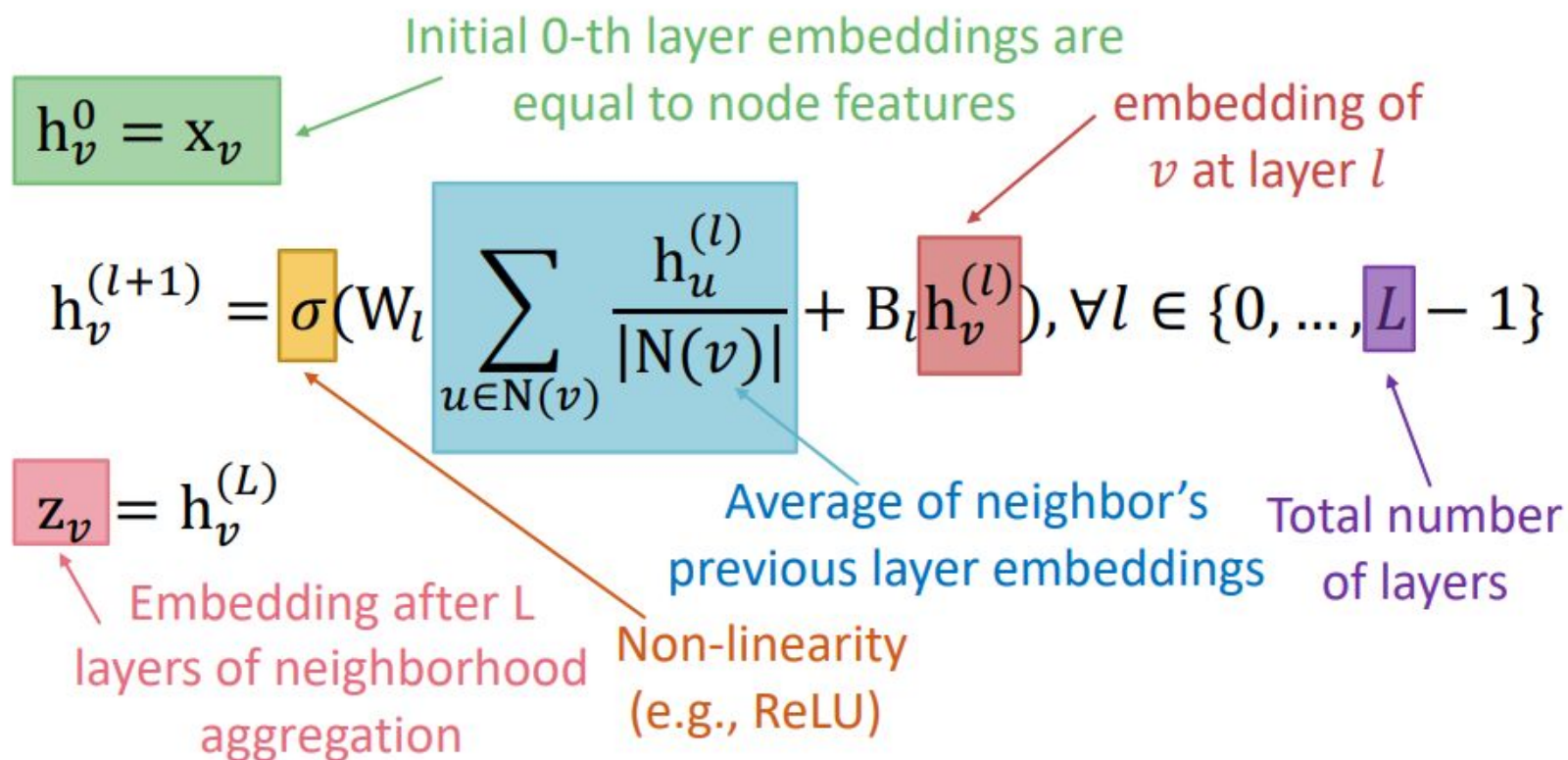
# GAT Intro

**Graph Representation Learning alleviates the need to do feature engineering **every single time**.**



# GAT Intro

- **Basic approach:** Average neighbor messages and apply a neural network



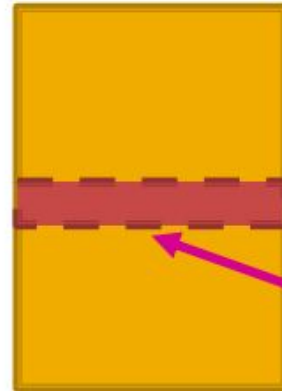
# GAT Intro

- **Many aggregations can be performed efficiently by (sparse) matrix operations**

- Let  $H^{(l)} = [h_1^{(l)} \dots h_{|V|}^{(l)}]^T$
- Then:  $\sum_{u \in N_v} h_u^{(l)} = A_{v,:} H^{(l)}$
- Let  $D$  be diagonal matrix where  $D_{v,v} = \text{Deg}(v) = |N(v)|$ 
  - The inverse of  $D$ :  $D^{-1}$  is also diagonal:  
 $D_{v,v}^{-1} = 1/|N(v)|$
- **Therefore,**

$$\boxed{\sum_{u \in N(v)} \frac{h_u^{(l-1)}}{|N(v)|}} \longrightarrow H^{(l+1)} = D^{-1} A H^{(l)}$$

Matrix of hidden embeddings  $H^{k-1}$



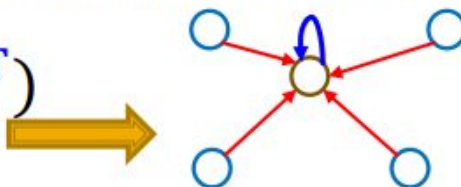
$h_i^{k-1}$

# GAT Intro

- Re-writing update function in matrix form:

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W_l^T + H^{(l)}B_l^T)$$

where  $\tilde{A} = D^{-1}A$



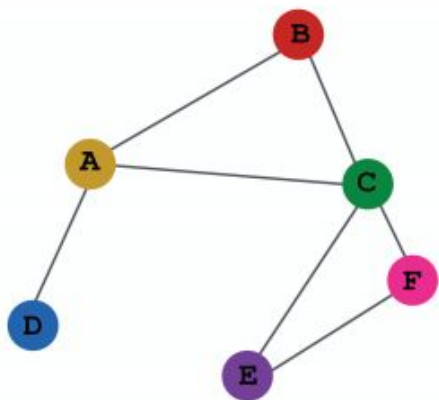
$H^{(l)} = [h_1^{(l)} \dots h_{|V|}^{(l)}]^T$

- Red: neighborhood aggregation
- Blue: self transformation

- In practice, this implies that efficient sparse matrix multiplication can be used ( $\tilde{A}$  is sparse)
- **Note:** not all GNNs can be expressed in matrix form, when aggregation function is complex



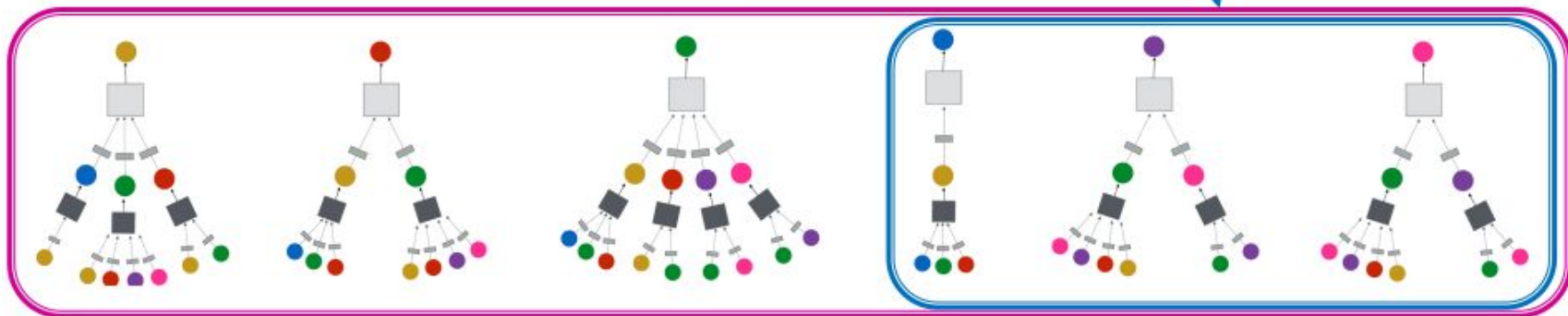
# GAT Intro



INPUT GRAPH

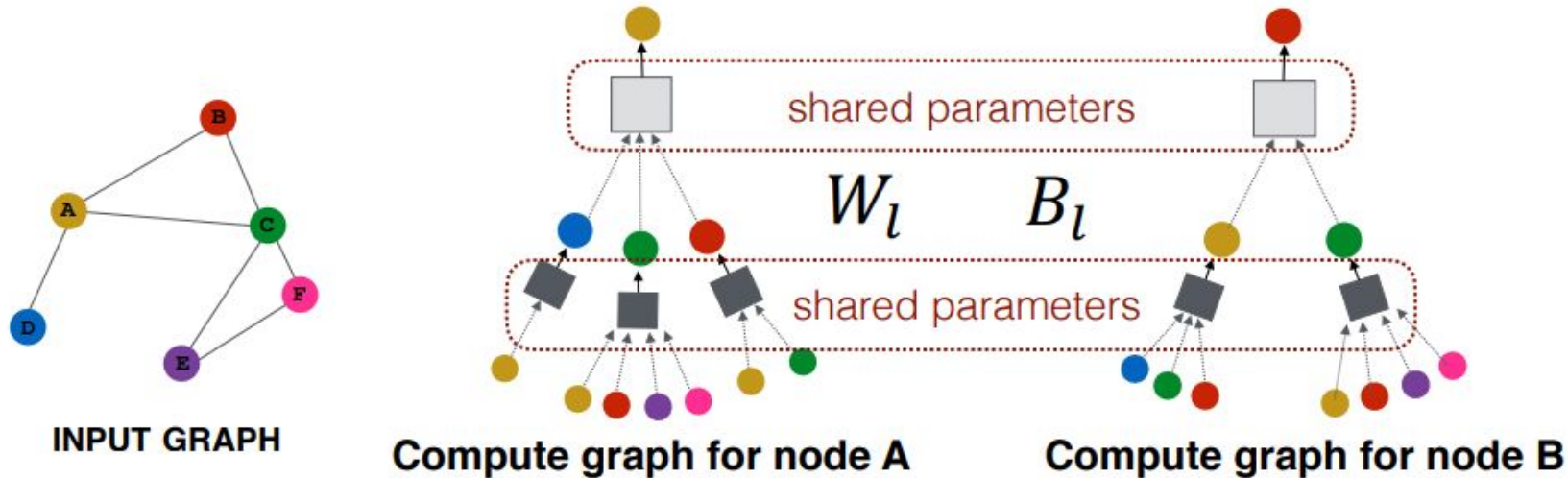
**(4) Generate embeddings  
for nodes as needed**

**Even for nodes we never  
trained on!**

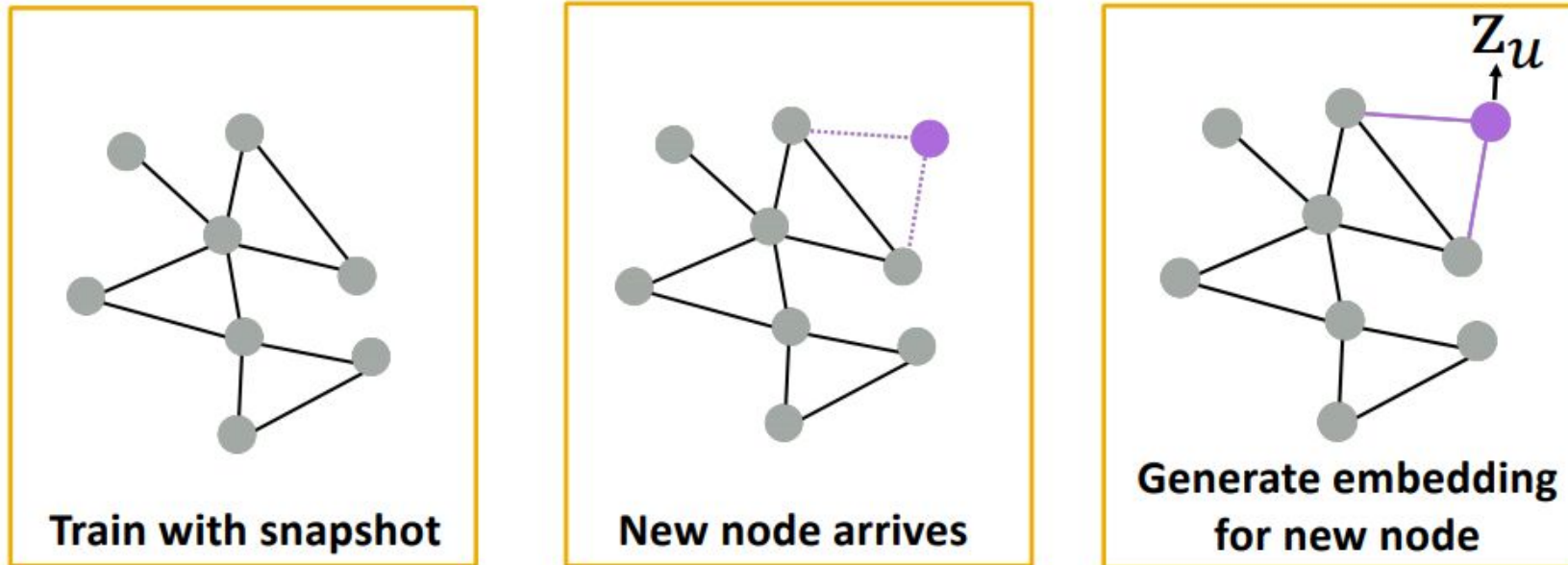


# GAT Intro

- The same aggregation parameters are shared for all nodes:
  - The number of model parameters is sublinear in  $|V|$  and we can **generalize to unseen nodes!**



# GAT Intro



- Many application settings constantly encounter previously unseen nodes:
  - E.g., Reddit, YouTube, Google Scholar
- Need to generate new embeddings “on the fly”

# GAT Intro

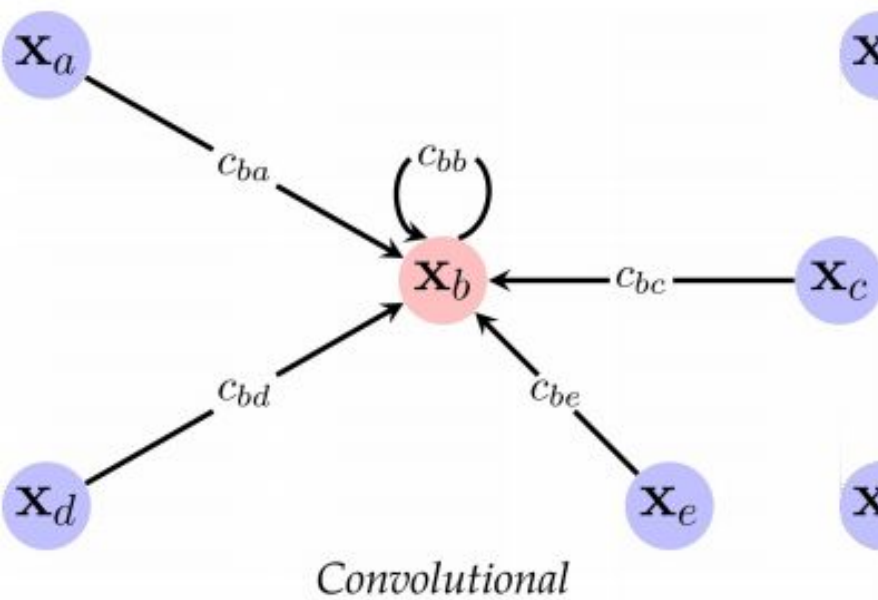
## ■ (3) Graph Attention Networks

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \underbrace{\alpha_{vu}}_{\text{Attention weights}} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right)$$

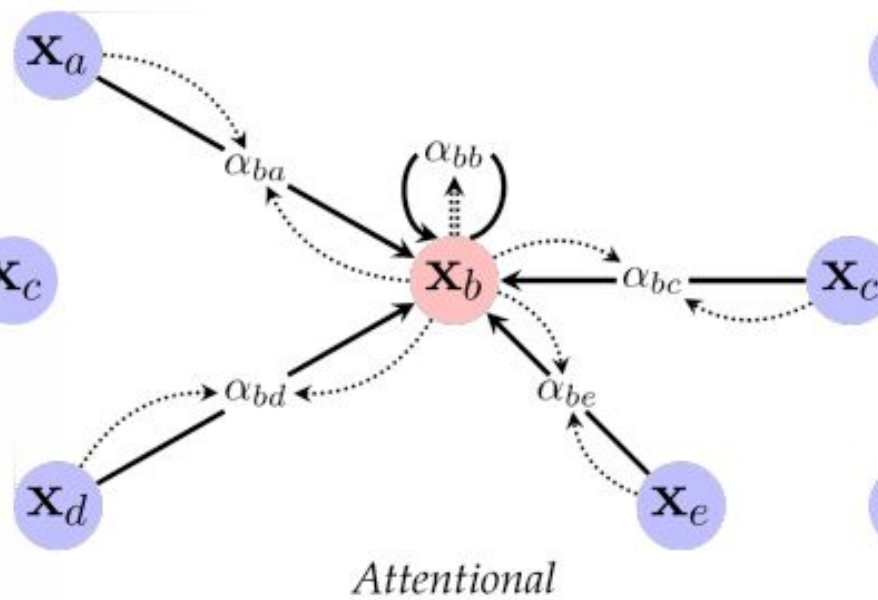
## ■ In GCN / GraphSAGE

- $\alpha_{vu} = \frac{1}{|N(v)|}$  is the **weighting factor (importance)** of node  $u$ 's message to node  $v$
- $\Rightarrow \alpha_{vu}$  is defined **explicitly** based on the structural properties of the graph (node degree)
- $\Rightarrow$  All neighbors  $u \in N(v)$  are equally important to node  $v$

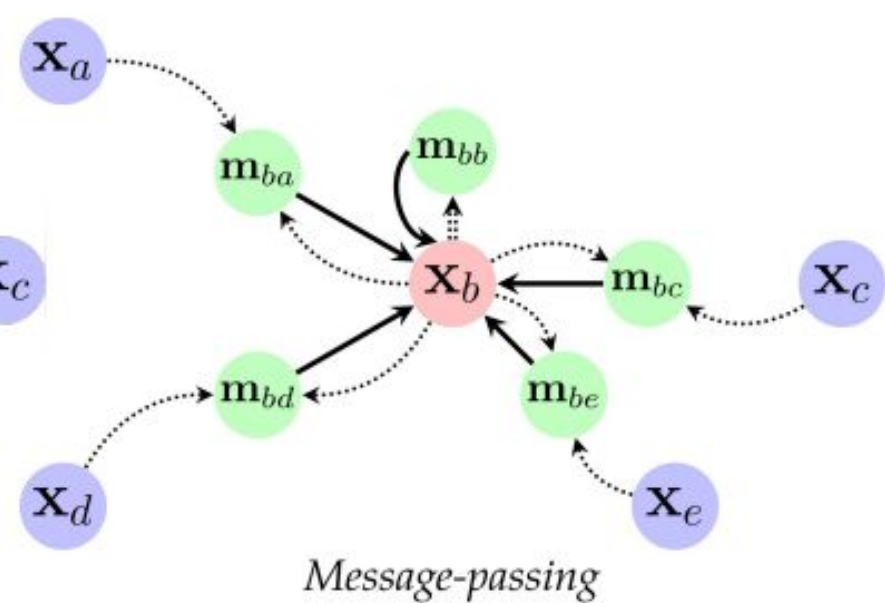




$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \psi(\mathbf{x}_j) \right)$$



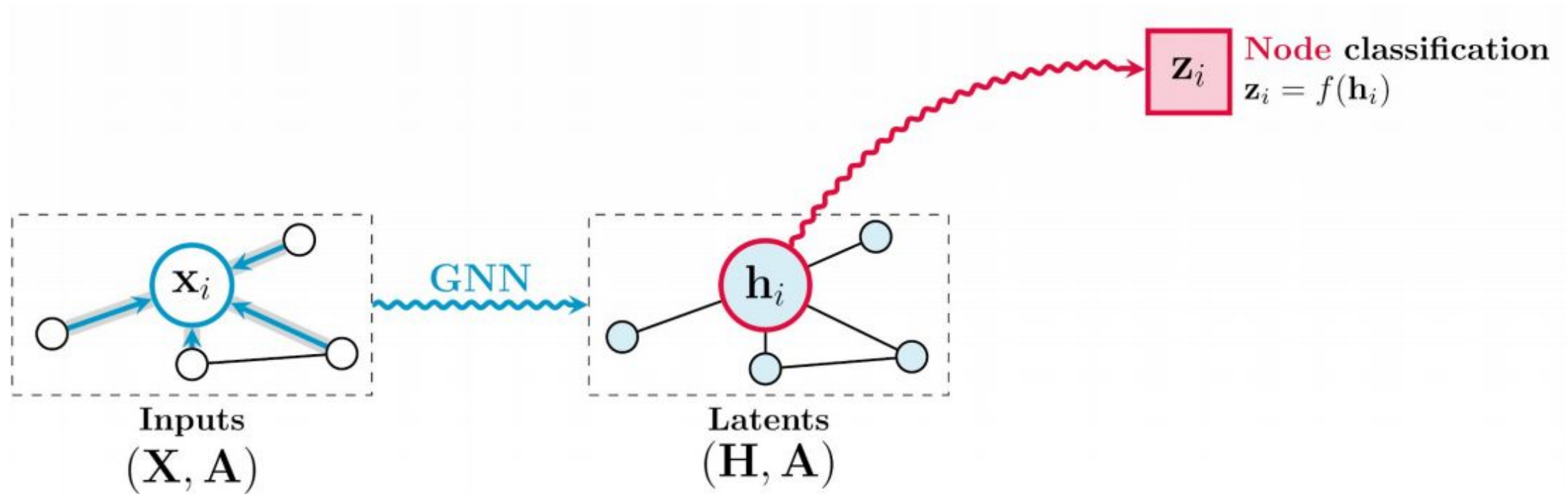
$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j) \psi(\mathbf{x}_j) \right)$$



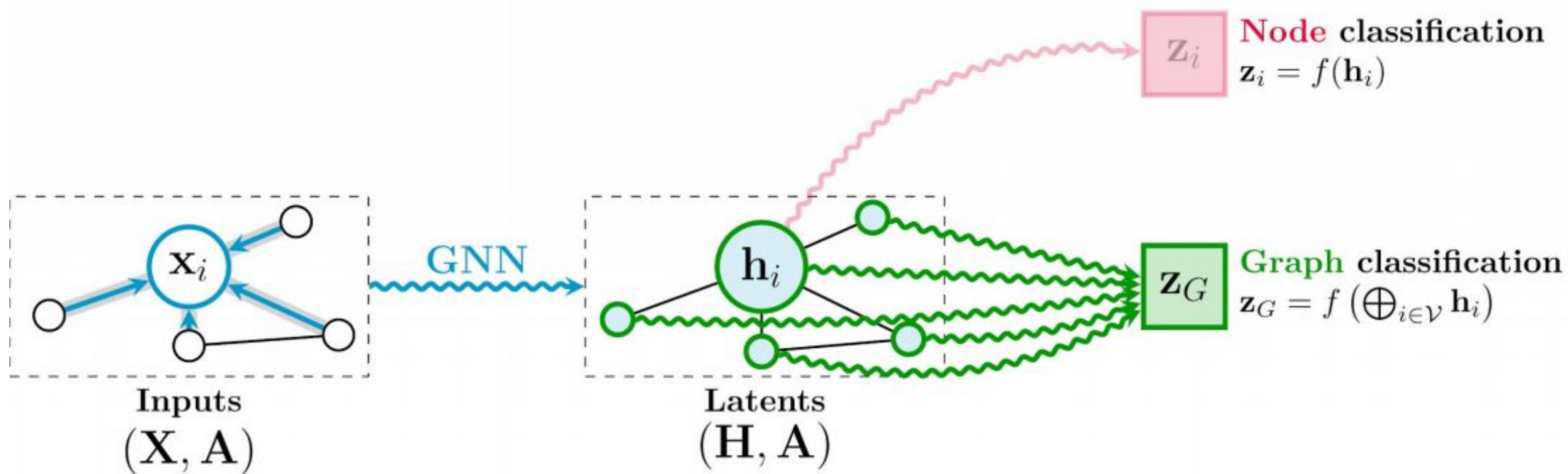
$$\mathbf{h}_i = \phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j) \right)$$



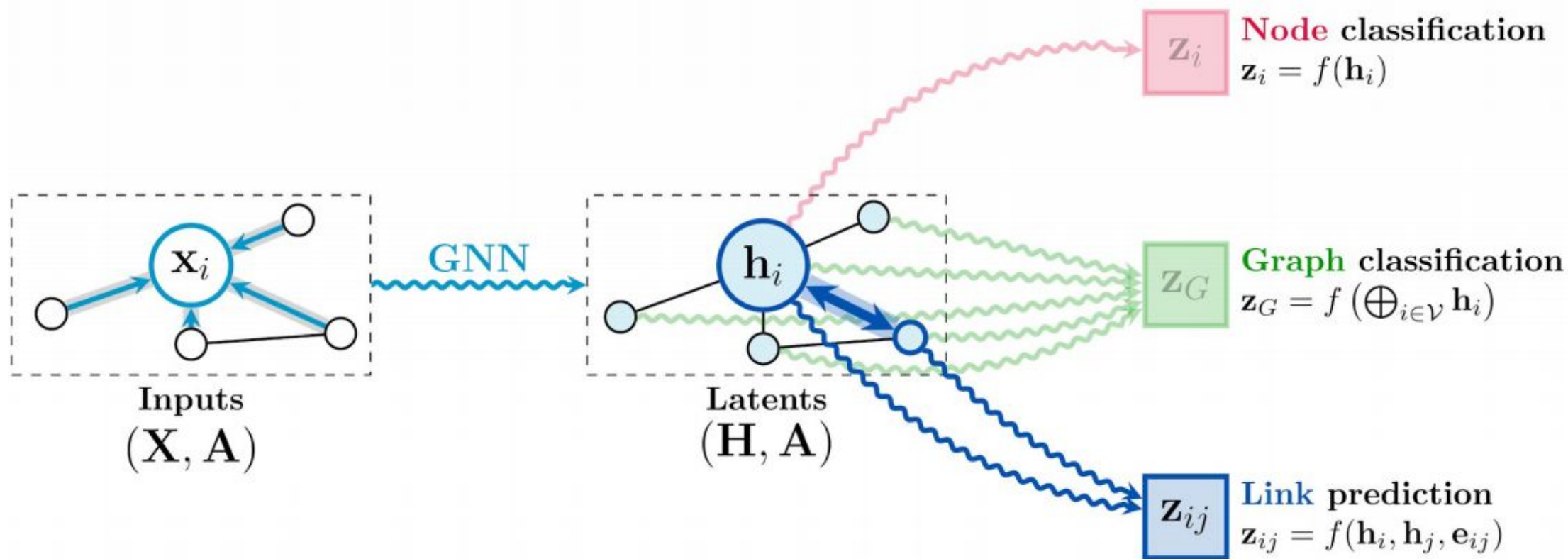
# GAT Intro



# GAT Intro



# GAT Intro

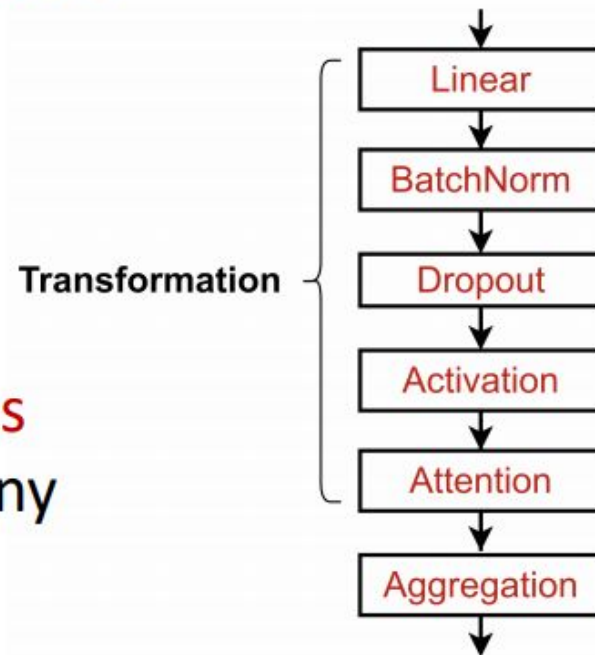


# GAT Intro - Further Motivation

- In practice, these classic GNN layers are a great starting point

- We can often get better performance by considering a general GNN layer design
- Concretely, we can include modern deep learning modules that proved to be useful in many domains

A suggested GNN Layer



# Data

Diversified our data into 3 groups

1. Cryptocurrency  
BTC/ ETH/ BNB/ XRP/ ... Total 9 coins
1. Stocks  
Tech/ Payment/ Financial/ Firms buying coins/ Others ...  
Total 24 stocks
1. Index and commodities prices  
S&P 500/ Dow Jones/ USD Index/ Gold Price/ Oil Price/ 10 yr  
Treasury Yield

