CPU USAGE PREDICTION WITH LTSM & GRU

ATHANASIOS TASIS [1093503] &
KWNSTANTINOS ALEXOPOULOS

PREPROCESSING 1 – ANALYZING THE DATA

	Unnamed: 0	timestamp	nodeid	node_cpu_usage	node_memory_usage
18868390	18868390	19740000	c962e6dfccd9eff5d3c9c0b0ec4840e060198c06f90735...	0.684157	0.758245
18868391	18868391	19740000	d8917e24e4ece24cefe7cec8b58e74f3d44f4482f6174d...	0.682535	0.767011
18868392	18868392	19740000	517dfc566f6d75645db5e04292d3a49bedb914532cf41d...	0.720583	0.829290
18868393	18868393	19740000	e203c6870f647a4ff900c8e906a54f891936f937c90bd2...	0.810112	0.795873
18868394	18868394	19740000	bd804d03b6f749f7a8d378fa885463b81a789730f9008f...	0.670382	0.723119
18868395	18868395	19740000	e25cee26e05f7b761b3b6a799387076204b81a89456584...	0.704788	0.742582
18868396	18868396	19740000	bab298a07cc548715b66f7be7f235a67ba8938cf5158e8...	0.697225	0.799916
18868397	18868397	19740000	6e59abf993ed62be45fe199ee4620f16ad85b824ad58de...	0.568611	0.728240
18868398	18868398	19740000	7eb8c2676ff1ae3acee1b70b0380becfb9f4b591b8a7e8...	0.619951	0.749109
18868399	18868399	19740000	f3064e4578a663535970809f08202def86d967cee28ff5...	0.652030	0.752251

Αναλύσαμε το head του αρχείου csv.

2 στήλες που μας ενδιαφέρουν: cpu usage & memory usage

Διαγράψαμε τελείως τα rows με NaN τιμές.

Υπολογίσαμε minimum & maximum τιμές για κάθε στήλη.

Μέσω της συνάρτησης calculate_volatility βρήκαμε ποια nodes είχαν ακραίες διακυμάνσεις σε σύντομο χρονικό διάστημα (threshold 0.75 σε < 3 ts).

Βρήκαμε τα nodes με συνεχόμενο high cpu usage (>0.9 για παραπάνω από το 50% των ts) και αντίστοιχα για low usage (<0.1) και super low usage (<0.05).

Ταξινομήσαμε τα δεδομένα μας πρώτα βάση nodeid και στη συνέχεια βάση timestamp

Βρήκαμε ποια nodes έχουν χαμηλό volatility (<0.1 για πάνω από 50% ts) και παρατηρήσαμε ότι σε όλα τα nodes το memory usage παρουσιάζει πολύ μικρές αλλαγές.

PREPROCESSING 2 - DOWNSIZING

Για να έχουμε καλύτερη προσπέλαση του αρχείου χρησιμοποιήσαμε γνωστές μεθόδους για να μειώσουμε το μέγεθος του αρχείου μας.

1) κάθε uniqe nodeid από ένα string 64 χαρακτήρων το μετατρέψαμε σε integer (1 - 1400)

2) Περιορίσαμε την δεκαδική ακρίβεια του cpu & memory usage στα 4 δεκαδικά από 16 που ήταν αρχικά.

3) Το timestamp το μετατρέψαμε σε μικρότερους αριθμούς αντί για unix (1, 2, 3, 4 κτλ)

4) Αφαιρέσαμε τις NaN τιμές.

5) Καθώς το memory usage μένει κυρίως σταθερό αποφασίσαμε με τον υπεύθυνο της εργασίας να μην ασχοληθούμε καθώς δεν έχει και πολύ νόημα να το προβλέψουμε.

	A	B	C	D	E	F
1	timestamp,nodeid,node_cpu_usage,node_memory_usage					
2	0,1,0.5639,0.7686					
3	1,1,0.5372,0.7688					
4	2,1,0.5497,0.7686					
5	3,1,0.5513,0.7687					
6	4,1,0.5887,0.7687					
7	5,1,0.6925,0.7688					
8	6,1,0.7217,0.7687					
9	7,1,0.6646,0.7687					
10	8,1,0.5804,0.7685					
11	9,1,0.5609,0.7686					
12	10,1,0.6112,0.7687					
13	11,1,0.6157,0.7689					
14	12,1,0.6115,0.7687					
15	13,1,0.6307,0.7689					
16	14,1,0.6485,0.7688					
17	15,1,0.6299,0.7693					
18	16,1,0.5987,0.769					
19	17,1,0.5878,0.7689					
20	18,1,0.5808,0.7691					

TRAINING THE MODEL

01

Αποφασίσαμε να κάνουμε ένα 80 – 20 split μεταξύ training & testing set.

02

Δοκιμάσαμε διάφορα loopback windows όπως 30 , 50 , 60 και 20.

03

Δοκιμάσαμε διάφορα epoch από 30 έως 100.

04

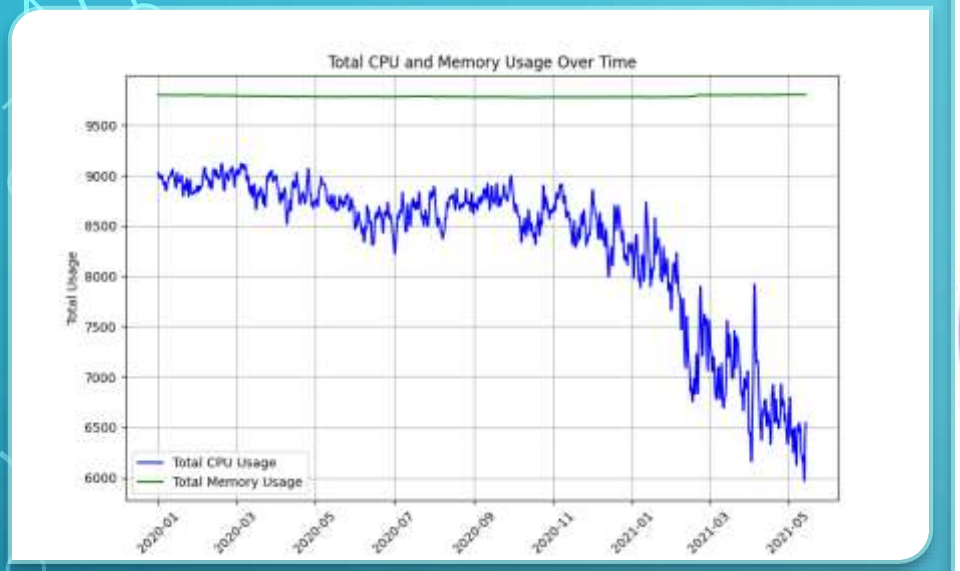
Ξεκινήσαμε προβλέποντας ένα μόνο node και όταν, βάση των μετρικών μας, ήμασταν ευχαριστημένοι συνεχίσαμε τρέχοντας το μοντέλο μας για batches από nodes (10,50,100,200).

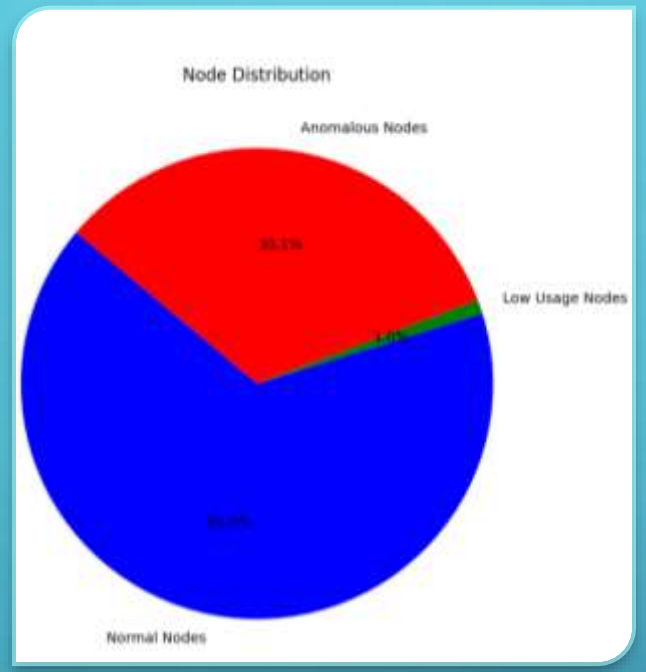
ΜΕΤΡΙΚΕΣ ΛΑΘΟΥΣ & ΠΡΟΒΛΗΜΑΤΑ ΠΟΥ ΑΝΤΙΜΕΤΩΠΙΣΑΜΕ

Για το evaluation του κάθε μοντέλου σε κάθε τεστ που κάναμε, υπολογίζαμε κάθε φορα το MSE, MAE, MAPE. Μετά από σύντομη έρευνα συμφωνήσαμε ότι το MAPE θέλουμε να είναι σίγουρα <20% και ιδανικά <10%. Επιπροσθέτως, για να έχουμε μια καλύτερη εικόνα, κάναμε plot το actual vs predicted node usage για να δούμε και visually πόσο καλά λειτουργεί το εκάστοτε μοντέλο.

Ένα από τα προβλήματα που αντιμετωπίσαμε είναι ότι τρέχοντας το μοντέλο μας σε ένα συγκεκριμένο node μπορεί να παίρναμε πολύ καλά αποτελέσματα, αλλά τρέχοντας το ίδιο ακριβώς μοντέλο σε κάποιο άλλο node το MAPE να ήταν >25%.

ΠΡΟΒΛΗΜΑΤΑ & ΛΥΣΕΙΣ





- Ξεκινήσαμε λοιπόν να αναλύουμε τα διαγράμματα που είχαμε στα χέρια μας ώστε να βρούμε ένα τρόπο να παίρνουμε καλά αποτελέσματα για ένα πιο ευρύ φάσμα από nodes.
- Θα πρέπει να είμαστε επίσης προσεκτικοί ώστε να μην κάνουμε overtrain το μοντέλο μας για να πετύχουμε αυτό το πιο ευρύ φάσμα.

```
Minimum values:
 timestamp                                                          0
nodeid          00048fcb278bd600fa0f41878de098d0971438740bdc0e...
node_cpu_usage                                              0.000606
node_memory_usage                                           0.002783
dtype: object

Maximum values:
 timestamp                                                   43170000
nodeid          fff43b80e176fadb97b76263b32b3abd895474e841eed8...
node_cpu_usage                                              0.977531
node_memory_usage                                           0.981635
dtype: object
Number of unique node IDs: 13118
```

```
Peak CPU Usage Time: [32700000]

Nodes with Constant Low Usage: 129

Nodes with Constant High Usage: 69

Nodes with Sudden Spikes: 1766

Total Nodes in Dataset: 13116
```

EVALUATION OF THE MODEL(LSTM)

```
Test Loss: 0.005045074038207531
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0040
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
Test MSE: 0.0050
Test MAE: 0.0421
Test MAPE: 12.14%
```

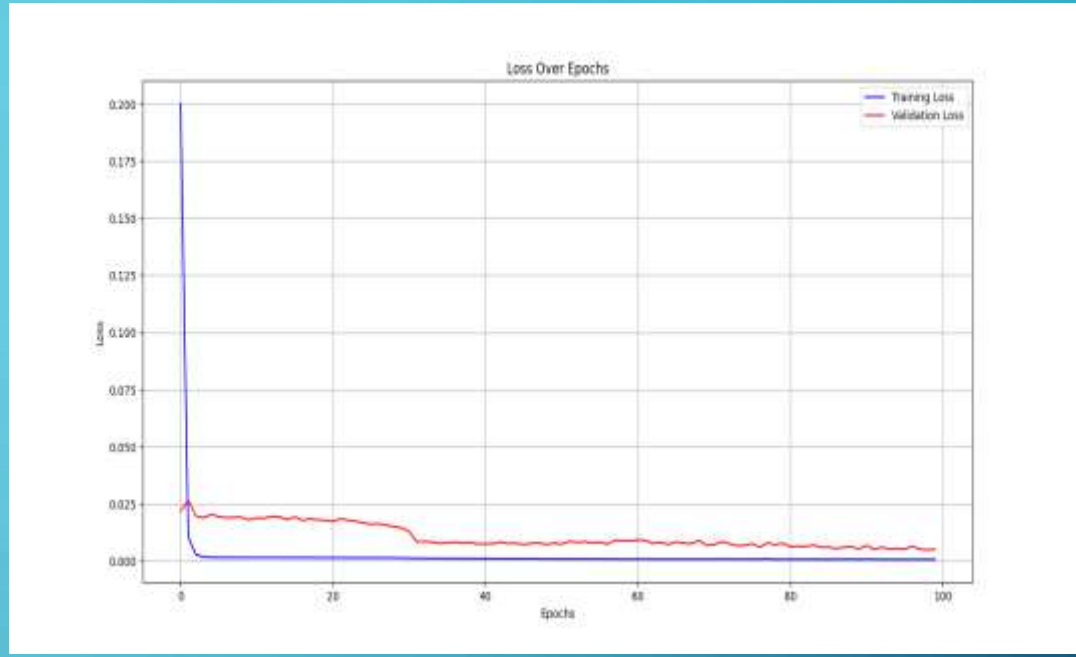
METRICS FOR NODE 5

```
Test Loss: 0.005686640739440918
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0045
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
Test MSE: 0.0057
Test MAE: 0.0496
Test MAPE: 13.81%
```

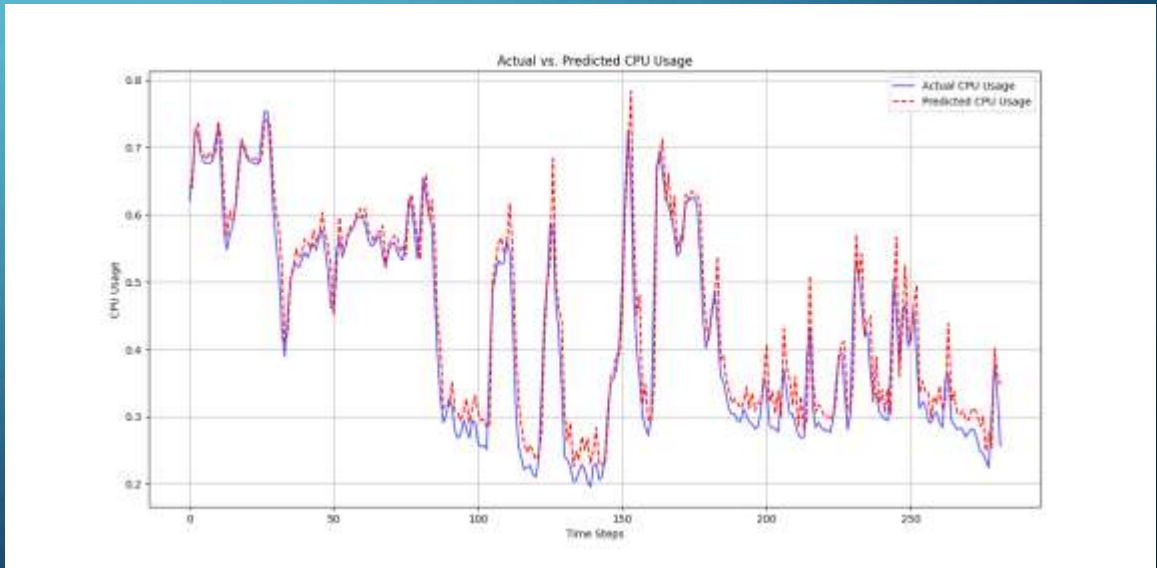
WITH LOOKBACK 20

```
Test Loss: 0.008484912104904652
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0067
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Test MSE: 0.0085
Test MAE: 0.0554
Test MAPE: 17.25%
```

WITH LOOKBACK 60



LOSS VS EPOCH



LOOKBACK: 30, EPOCH: 50

OPTIMISING THE MODEL USING TRANSFER LEARNING

- **Fine-tuning** των τελευταίων layers ώστε να προσαρμοστούν στα νέα δεδομένα
- Χρήση **freezing layers** για διατήρηση χρήσιμων weights από το pre-trained model
- **Regularization techniques** (Dropout, L2 penalty) για αποφυγή **overfitting**
- Με αυτήν την προσέγγιση, επιτυγχάνουμε **καλύτερο accuracy** με λιγότερα **training epochs**, διατηρώντας **efficiency & performance**!

```
Training general model on nodes 1 to 100 with horizon 15...
Epoch 1/5
1703/1703 ━━━━━━━━━━━━━━━━━━━━ 33s 16ms/step - loss: 0.0322 - val_loss: 0.0051
Epoch 2/5
1703/1703 ━━━━━━━━━━━━━━━━━━━━ 33s 13ms/step - loss: 0.0045 - val_loss: 0.0054
Epoch 3/5
1703/1703 ━━━━━━━━━━━━━━━━━━━━ 22s 13ms/step - loss: 0.0044 - val_loss: 0.0048
Epoch 4/5
1703/1703 ━━━━━━━━━━━━━━━━━━━━ 41s 13ms/step - loss: 0.0044 - val_loss: 0.0049
Epoch 5/5
1703/1703 ━━━━━━━━━━━━━━━━━━━━ 42s 13ms/step - loss: 0.0043 - val_loss: 0.0049
General Model Test Loss: 0.0049
852/852 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step
General Model Aggregated Test MAPE: 8.81%
9/9 ━━━━━━━━━━━━━━━━━━━━ 2s 119ms/step
```

```
Node 113: Loss = 0.0023, MAPE = 5.51%
Node 114: Loss = 0.0010, MAPE = 3.15%
Node 115: Loss = 0.0066, MAPE = 11.76%
Node 116: Loss = 0.0061, MAPE = 10.11%
Node 117: Loss = 0.0168, MAPE = 15.12%
```

HORIZON: 15

```
GPU is available. Training on GPU.
Training general model on all nodes data...
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `inp
  super().__init__(**kwargs)
898/898 ━━━━━━━━━━━━━━━━━━━━ 15s 12ms/step - loss: 0.0249 - val_loss: 0.0018
Epoch 2/5
898/898 ━━━━━━━━━━━━━━━━━━━━ 15s 8ms/step - loss: 9.4824e-04 - val_loss: 0.0012
Epoch 3/5
898/898 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 5.7552e-04 - val_loss: 8.3536e-04
Epoch 4/5
898/898 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 5.1141e-04 - val_loss: 0.0014
Epoch 5/5
898/898 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step - loss: 4.9524e-04 - val_loss: 0.0022
General Model Test Loss: 0.0022
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `inp
  super().__init__(**kwargs)
9/9 ━━━━━━━━━━━━━━━━━━━━ 2s 115ms/step
Node 100: Loss = 0.0042, MAPE = 6.81%
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `inp
  super().__init__(**kwargs)
```

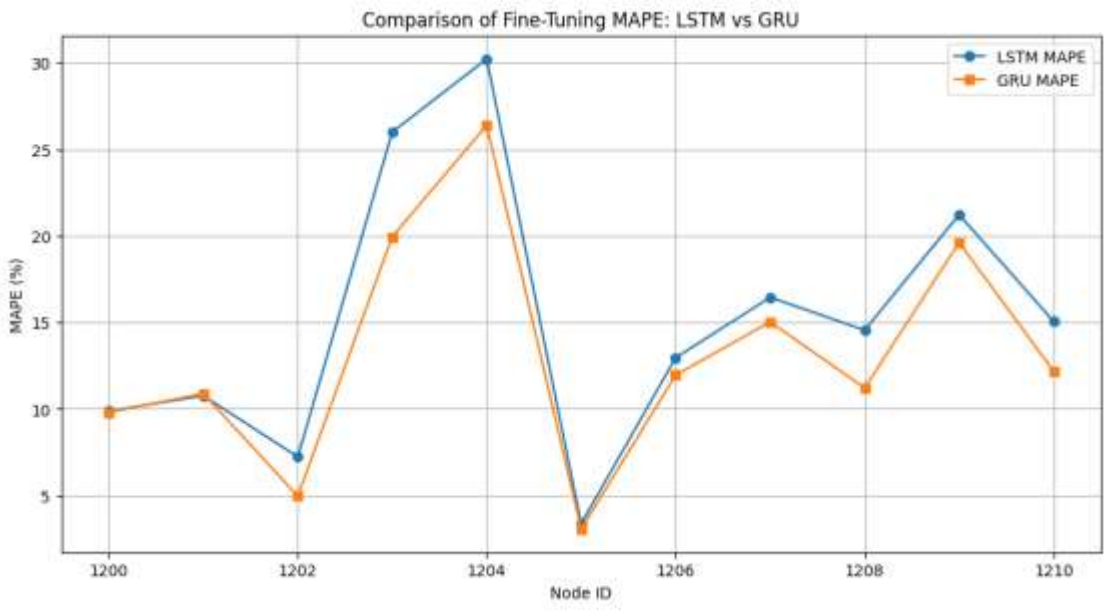
FOR NODES 100 TO 150

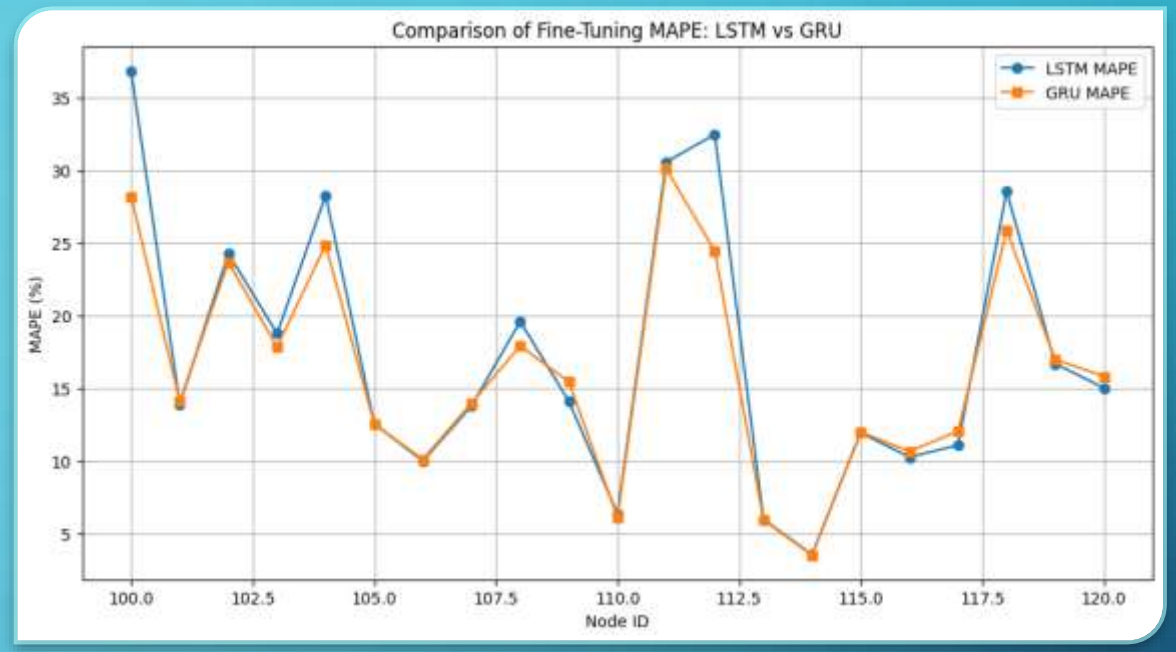
```
Sample 2: Actual CPU = [0.6491 0.6491 0.7436 0.7436 0.7594], Predicted CPU = [0.6774951  0.68240947 0.6842388  0.6850252  0.6849842 ]
Sample 3: Actual CPU = [0.6491 0.7436 0.7436 0.7594 0.7594], Predicted CPU = [0.66536534 0.6730889  0.67620295 0.6787466  0.6778753 ]
Sample 4: Actual CPU = [0.7436 0.7436 0.7594 0.7594 0.7415], Predicted CPU = [0.67102855 0.6762182  0.6786662  0.6797725  0.679096  ]
Sample 5: Actual CPU = [0.7436 0.7594 0.7594 0.7415 0.7415], Predicted CPU = [0.7473038  0.7270413  0.73128694 0.71736765 0.71734416]
Node 114: Loss = 0.0007, MAPE = 2.66%
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 87ms/step

Node 115 Fine-tuning Results (first 5 samples):
Sample 1: Actual CPU = [0.6313 0.6022 0.6022 0.5988 0.5988], Predicted CPU = [0.6741196  0.6640689  0.65644157 0.6559844  0.6500915 ]
Sample 2: Actual CPU = [0.6022 0.6022 0.5988 0.5988 0.6097], Predicted CPU = [0.6326568  0.6320858  0.6296644  0.63374805 0.62965095]
Sample 3: Actual CPU = [0.6022 0.5988 0.5988 0.6097 0.6097], Predicted CPU = [0.6147655  0.61666334 0.616255   0.62063056 0.61831504]
Sample 4: Actual CPU = [0.5988 0.5988 0.6097 0.6097 0.523 ], Predicted CPU = [0.61631334 0.6167518  0.6158442  0.61895555 0.61741924]
Sample 5: Actual CPU = [0.5988 0.6097 0.6097 0.523  0.523 ], Predicted CPU = [0.6095706  0.611251   0.61084205 0.6145587  0.6135759 ]
Node 115: Loss = 0.0051, MAPE = 10.26%
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 95ms/step

Node 116 Fine-tuning Results (first 5 samples):
Sample 1: Actual CPU = [0.5809 0.6145 0.5931 0.6587 0.7298], Predicted CPU = [0.5323482 0.5524973 0.5680435 0.5787367 0.5878862]
Sample 2: Actual CPU = [0.6145 0.5931 0.6587 0.7298 0.7262], Predicted CPU = [0.6093154 0.6116762 0.6179086 0.6189516 0.625381 ]
Sample 3: Actual CPU = [0.5931 0.6587 0.7298 0.7262 0.6699], Predicted CPU = [0.6203229  0.62300605 0.62772    0.63066256 0.6348195 ]
Sample 4: Actual CPU = [0.6587 0.7298 0.7262 0.6699 0.6186], Predicted CPU = [0.6057322  0.6117529  0.6183412  0.6235816  0.62720096]
Sample 5: Actual CPU = [0.7298 0.7262 0.6699 0.6186 0.5601], Predicted CPU = [0.684455   0.6732094  0.6715471  0.666751   0.66753304]
Node 116: Loss = 0.0037, MAPE = 7.52%
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 86ms/step

Node 117 Fine-tuning Results (first 5 samples):
Sample 1: Actual CPU = [0.6326 0.6372 0.5975 0.5834 0.6246], Predicted CPU = [0.6136998  0.6140496  0.61293    0.61100644 0.60523826]
Sample 2: Actual CPU = [0.6372 0.5975 0.5834 0.6246 0.6983], Predicted CPU = [0.6421521  0.6362693  0.6317197  0.62625265 0.61900437]
```

HORIZON: 5

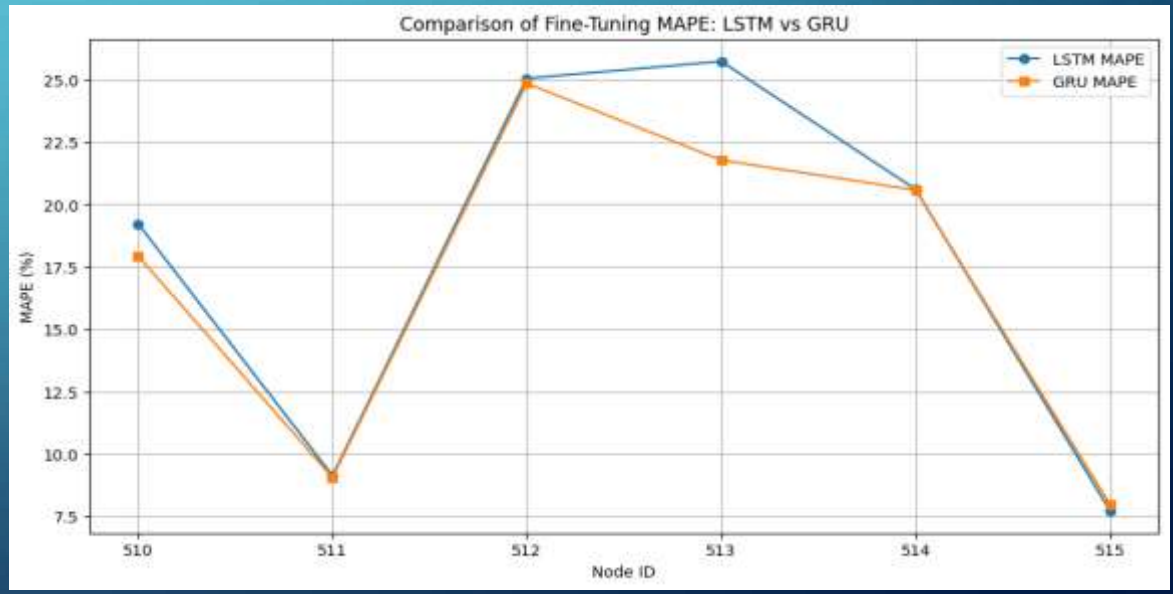
ΔΗΜΙΟΥΡΓΙΑ GRU ΜΟΝΤΕΛΟΥ

Δουλέψαμε πάνω στο ίδιο, ήδη preprocessed, αρχείο. Και με το κατάλληλο tweaking, «παίξαμε» με τις παραμέτρους ώστε να δούμε πώς θα κάνουμε την καλύτερη δυνατή πρόβλεψη.





LONG HORIZON(15) COMPARISON



SHORT HORIZON(5) COMPARISON