Final Report
Zihao Liu

- **Abstract**

This final report summarizes all the work I have done in this project. The sequence of this report follows the requirements of the exercise part 1. Due to the limited calculation power of my laptop, I chose to use Google colab (GPU) to complete all the model writing and running. Also, due to the time limitation of this project, I did not follow the original paper's model hyper-parameters of SSSD4 and CSDI models but reduce the number of iteration and diffusion steps (the model is also converged). Both SSSD4 and CSDI models have been converted into TensorFlow framework and can run, be saved and loaded successfully. Models' training and testing are both successfully tested with the unit tests, regression test and integration tests. The training and testing datasets used are the "Mujoco" datasets provided by the paper's author Alcaraz. The required financial data has been input to train and test.

- **Literature review**

The target paper we focus on in this report is Alcaraz and Strodthoff (2022). This research shows the better missing data imputation performance of Structured State Space Diffusion (SSSD) Model than other state-of-the-art probabilistic imputation methods with multiple data sets. All the testing is done in three missingness scenarios: random missing (Rm), missing not at random (Mnr) and blackout missing (Bm). The main benchmark used to compare is the Conditional score-based diffusion model (CSDI) (Tashiro, et al, 2021) which uses diffusion models as DiffWave-variants. Alcaraz and Strodthoff (2022) also use diffusion models (DiffWave framework) but they modify the internals of the DiffWave architecture using two structured state space sequence (S4) layers and the dimension of the input data has been changed to 3-D (diffusion only along the time dimension). As structured state space model draws on a linear state space transition equation, connecting a one-dimensional input sequence to a one-dimensional output sequence via a N-dimensional hidden state (Gu, A, 2021). This modification of S4 is considered to be able to better capture the long-term-dependencies in time series data than dilated convolutions in the benchmark structure.

After this work, the experimental algorithm is created as SSSDS4 model which is an autoregressive model. The authors also expand it to SSSDSA model with a SaShiMi architecture by combining S4 layers in a U-net-inspired configuration which is a non-autoregressive model. For the comparison, CSDI model is extended to CSDIS4 model as two benchmarks. Two diffusion conditions are considered that diffusion process is imputed to the portions of the sample (D1) and to the entire sample (D0). Based on the paper's results, both SSSDS4 and SSSDSA beat the two CSDI benchmarks under three averaged MAE, RMSE and CRPS criteria using multiple data sets (with multiple data structures). The reason given by authors is that only proposed two SSSD models can capture all essential signal features.

As our research focuses on the holiday missing financial data imputation, we may focus more on the "Bm" scenario (different target stocks are seen as channels). Also, we only provide the comparison between SSSDS4 and CSDI algorithms under TensorFlow framework due to the

limited time. More exploring can be done based on the transformed TensorFlow configurations. To answer specific financial data missing questions, we provide some potential methods like Gap trading and time-series trading. Dahlquist and Bauer (2012) introduce the details on Gap trading strategies from gap identification to gap closing. Returns can be realized based on the information overstock reflected by a market gap. Also, Mathuria and Bhakar (2014) provide a portfolio optimization based on the non-probabilistic Information Gap Decision Theory.

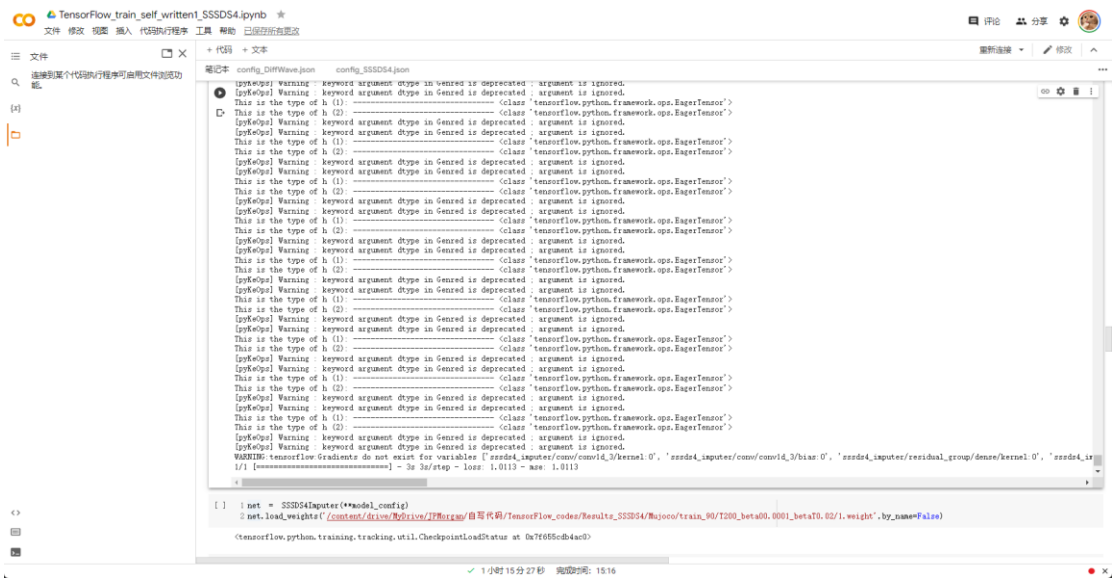
- **Part 1**
- **(a)**

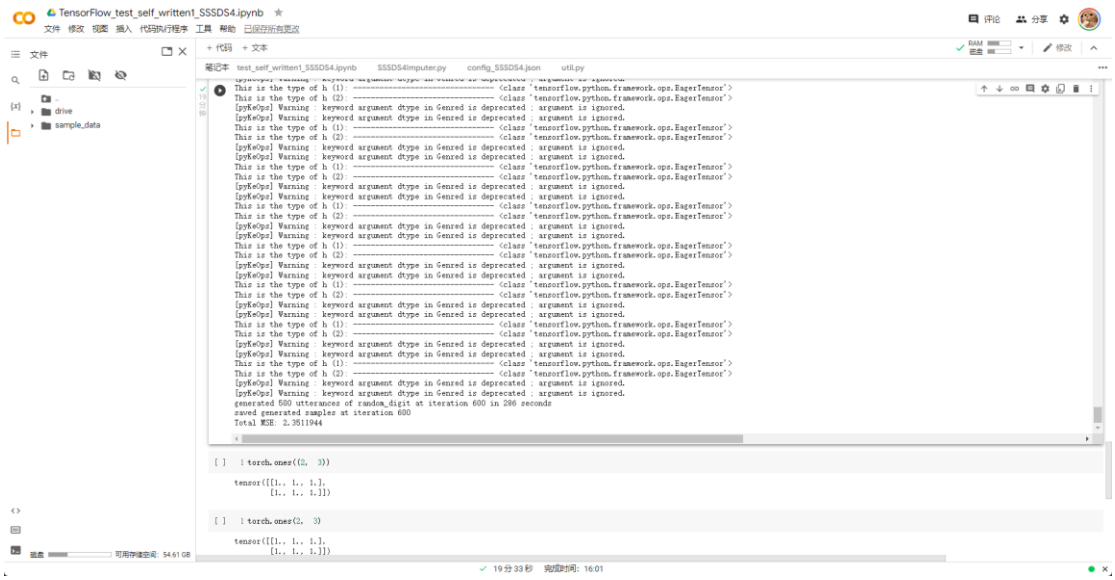
Due to the time limitation and the out-of-memory issue of this project, under the ensuring of the model's converging, we have reduced some hyper-parameters of the original model's configuration to save more time in both the training and testing. However, we can increase the size of these parameters later when we do the real predictions. These small-size experiments are used to show our models can successfully run.

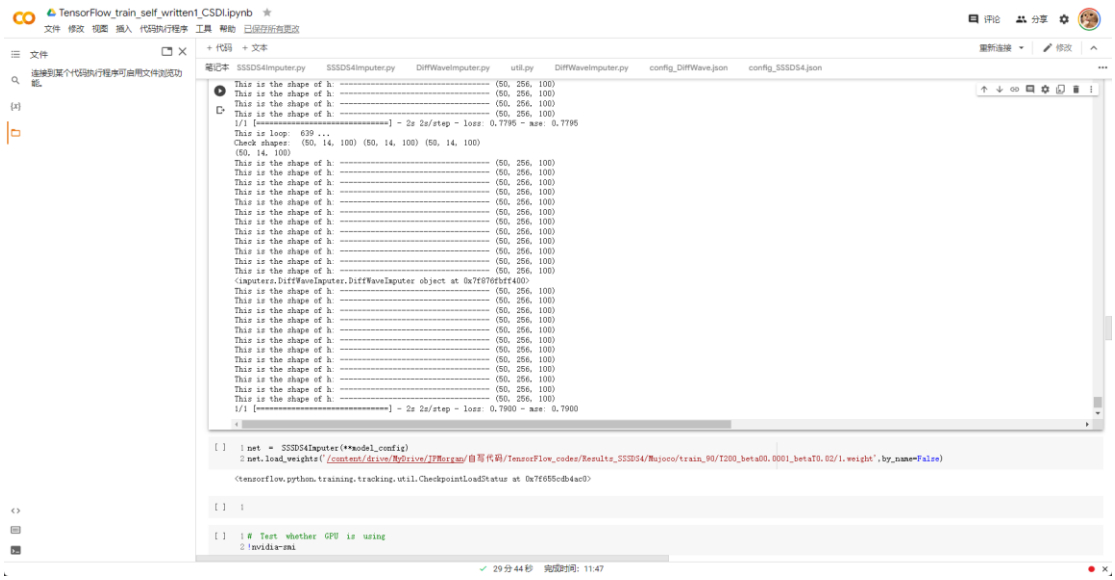
The reduced parameters include **in the training:** (1) the number of iteration ("n_iters") from 150000 to 600. (2) the k missing time steps for each feature across the sample length ("missing_k") from 200 to 90 (3) the number of residual layers ("num_res_layers") from 36 to 12; **in the testing:** (1) the number of diffusion steps ("T") from 200 to 10.

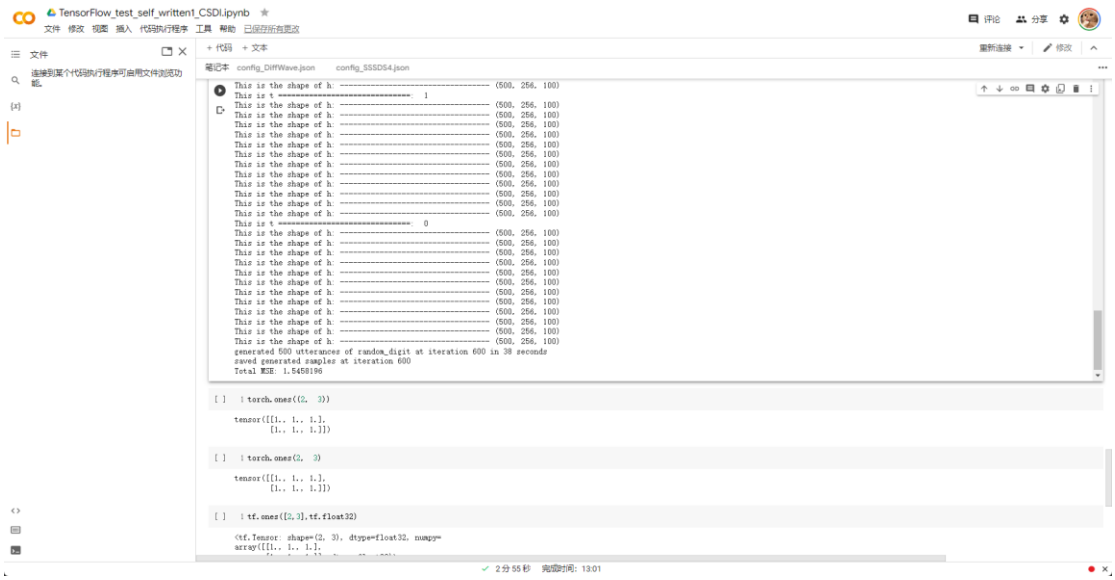
-The following experiments are under the whole sample diffusion ("only_generate_missing" = 0)

1. Random missing ("masking" = "rm")

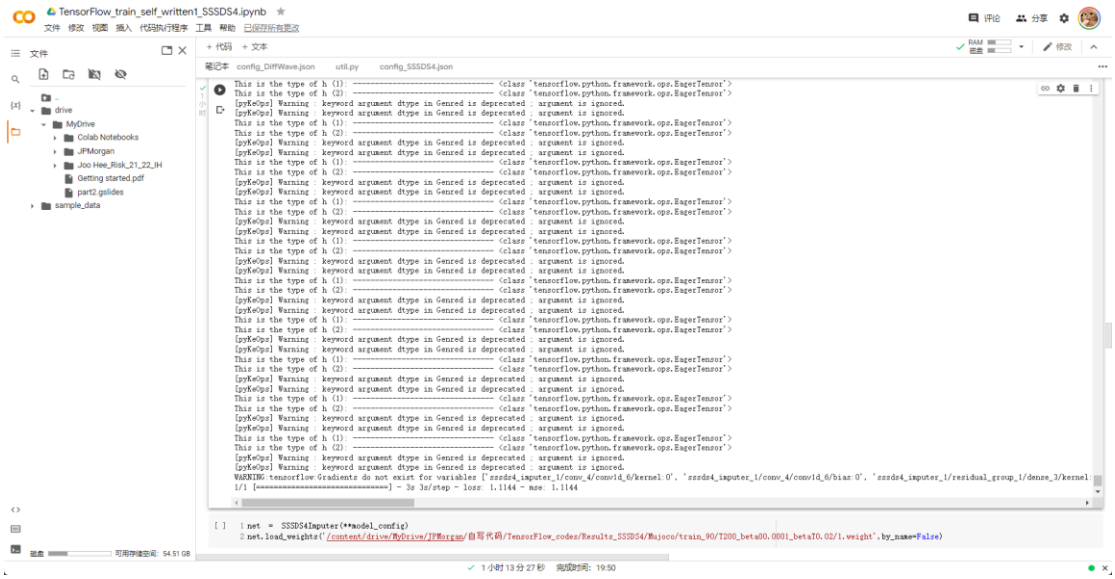


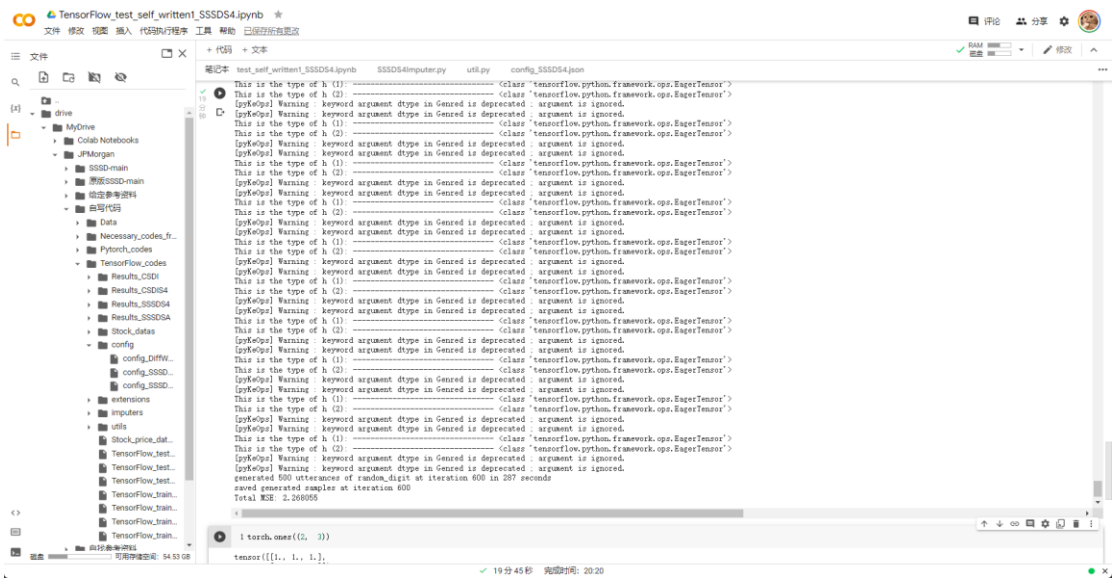

TensorFlow_test_self_written1_SSSDS4.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助 已保存所有更改
评论 分享
+ 代码 + 文本
修改
笔记本 test_self_written1_SSSDS4.ipynb SSSDS4Imputer.py config_SSSDS4.json util.py
文件
drive
sample_data
This is the type of h (1): <class 'tensorflow.python.framework.ops.EagerTensor'>
This is the type of h (2): <class 'tensorflow.python.framework.ops.EagerTensor'>
[pyKeOps] Warning : keyword argument dtype in Genred is deprecated : argument is ignored.
generated 500 utterances of random_digit at iteration 600 in 286 seconds
saved generated samples at iteration 600
Total MSE: 2.3511944
[ ] 1 torch.ones((2, 3))
tensor([[1., 1., 1.],
[1., 1., 1.]])
[ ] 1 torch.ones(2, 3)
tensor([[1., 1., 1.],
[1., 1., 1.]])
磁盘 可用存储空间：54.61 GB
19 分 33 秒 完成时间：16:01

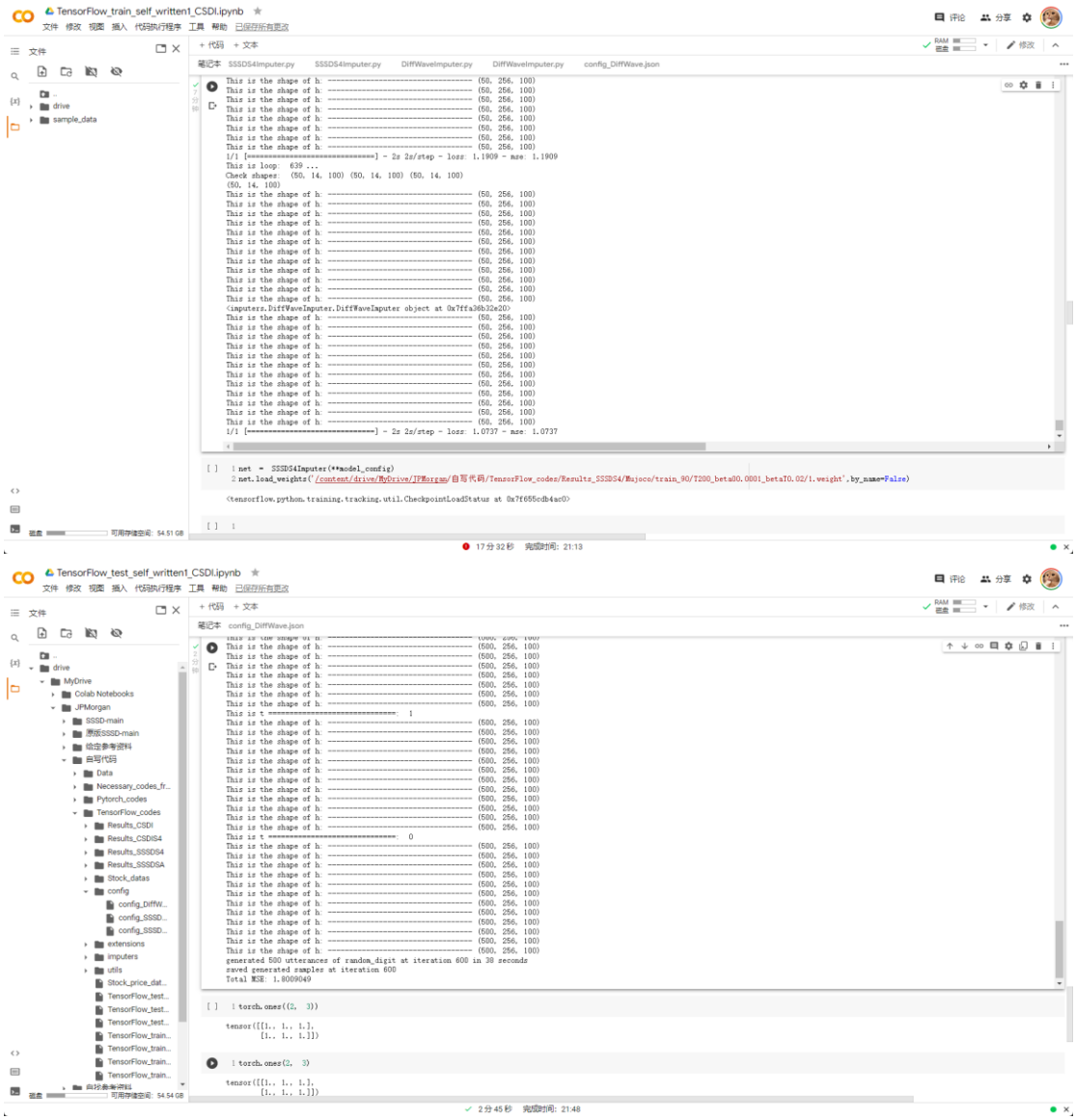

TensorFlow_train_self_written1_CSDI.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助 已保存所有更改
评论 分享
+ 代码 + 文本
重新连接
修改
笔记本 SSSDS4Imputer.py SSSDS4Imputer.py DiffWaveImputer.py util.py DiffWaveImputer.py config_DiffWave.json config_SSSDS4.json
文件
连接到某个代码执行程序可启用文件浏览功能。
This is the shape of h: ---------- (50, 256, 100)
1/1 [==============================] - 2s 2s/step - loss: 0.7795 - mae: 0.7795
This is loop: 639 ...
Check shapes: (50, 14, 100) (50, 14, 100) (50, 14, 100)
(50, 14, 100)
<imputers.DiffWaveImputer.DiffWaveImputer object at 0x7f876fbff400>
1/1 [==============================] - 2s 2s/step - loss: 0.7900 - mae: 0.7900
[ ] net = SSSDS4Imputer(**model_config)
net.load_weights('/content/drive/MyDrive/ZFMorgan/自写代码/TensorFlow_codes/Results_SSSDS4/Mujoco/train_90/T200_beta00.0001_betaT0.02/1.weight',by_name=False)
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f655cdb4ac0>
[ ] 1
[ ] 1 # Test whether GPU is using
2 !nvidia-smi
29 分 44 秒 完成时间：11:47



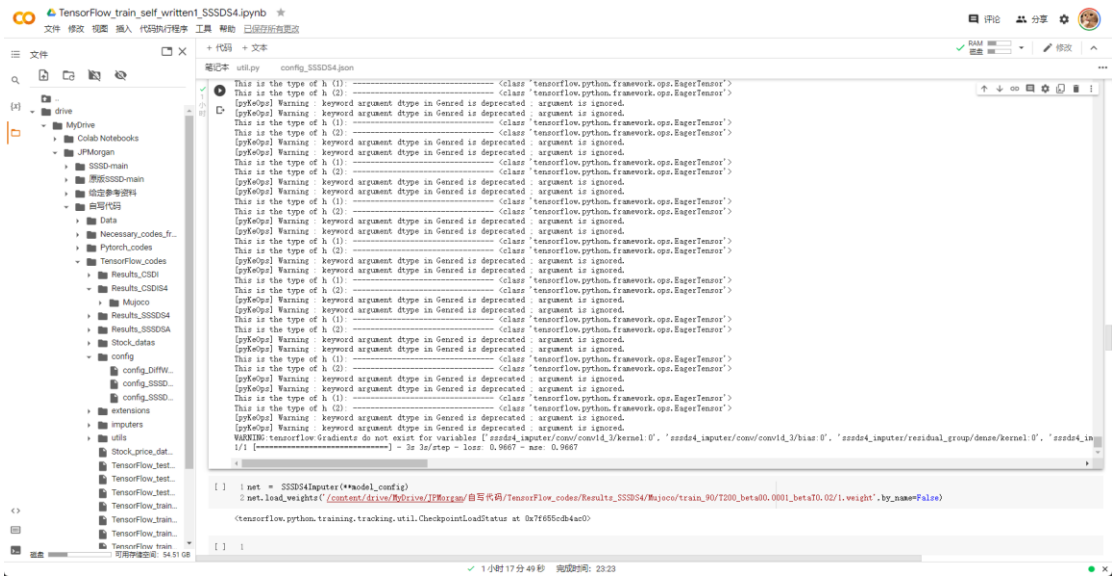
2. Missing not at random (“masking” = ”mnr”)

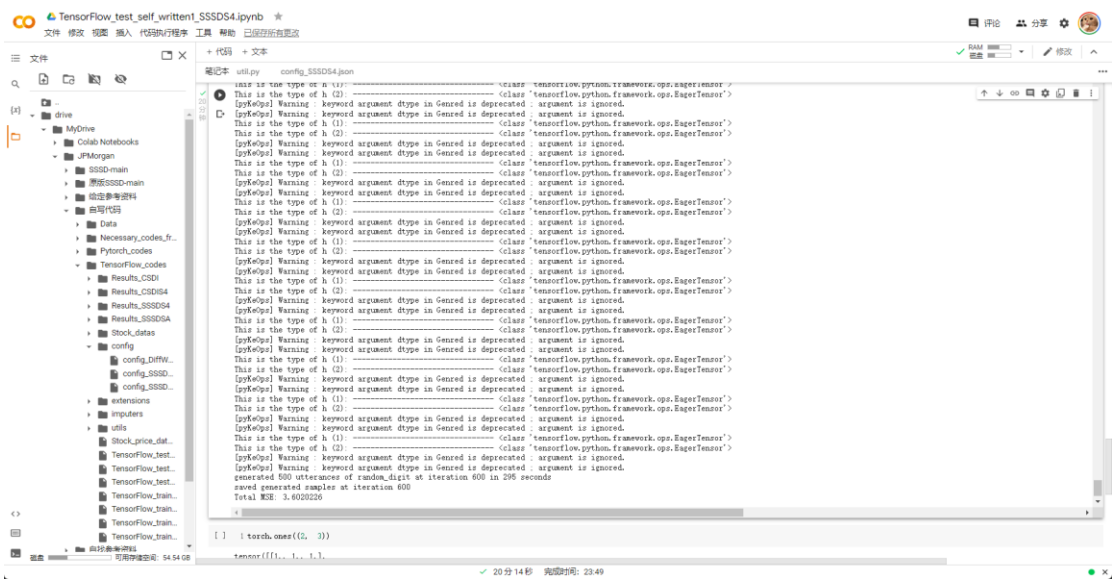


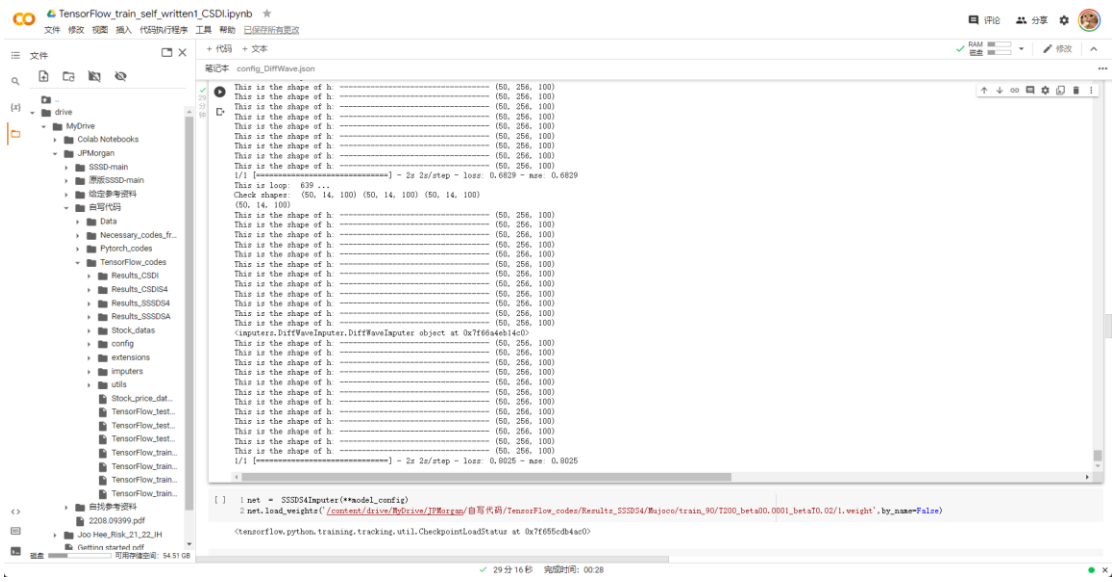


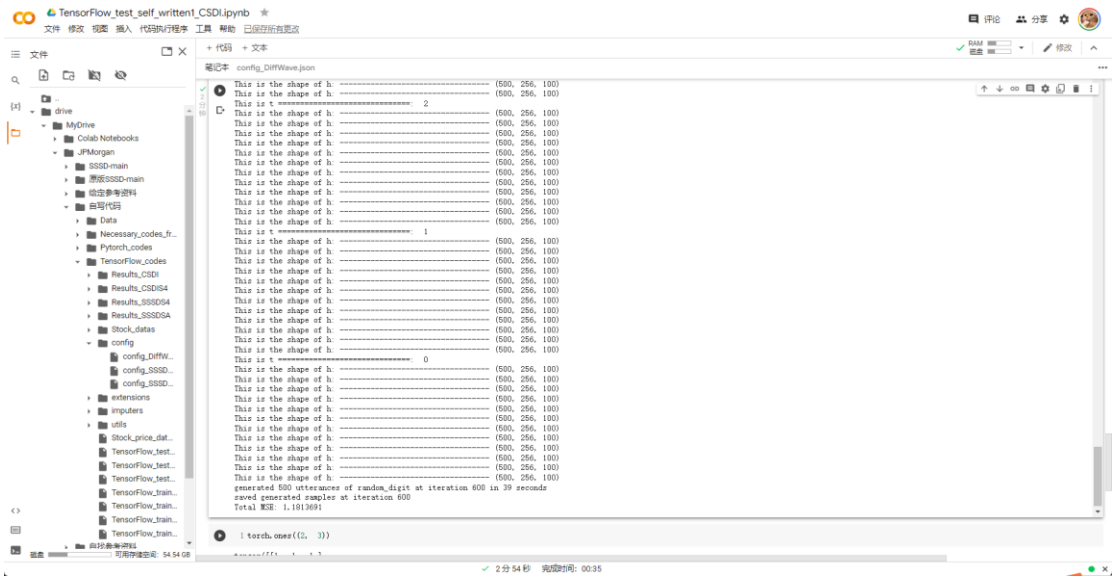


3. Blackout missing (“masking” = ”bm”)


TensorFlow_train_self_written1_SSSDS4.ipynb
net = SSSDS4Imputer(**model_config)
net.load_weights('/content/drive/MyDrive/JPMorgan/自写代码/TensorFlow_codes/Results_SSSDS4/Mujoco/train_90/T200_beta00.0001_betaT0.02/1.weight',by_name=False)
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f655cdb4ac0>
1 小时 17 分 49 秒 完成时间: 23:23

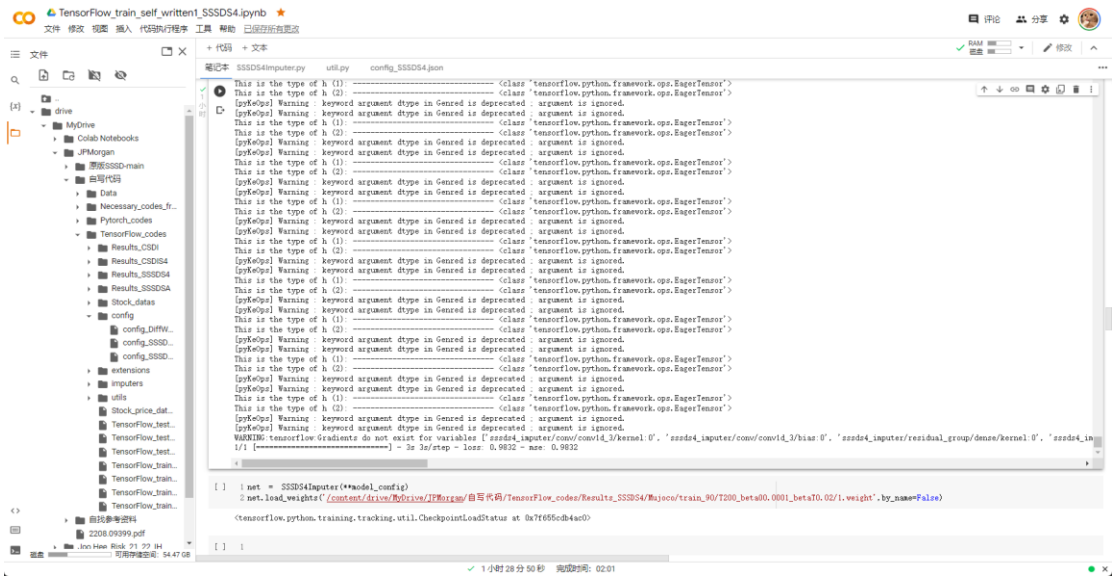

TensorFlow_test_self_written1_SSSDS4.ipynb
generated 500 utterances of random_digit at iteration 600 in 295 seconds
saved generated samples at iteration 600
Total MSE: 3.6020226
torch.ones((2, 3))
tensor([[1., 1., 1.],
20 分 14 秒 完成时间: 23:49

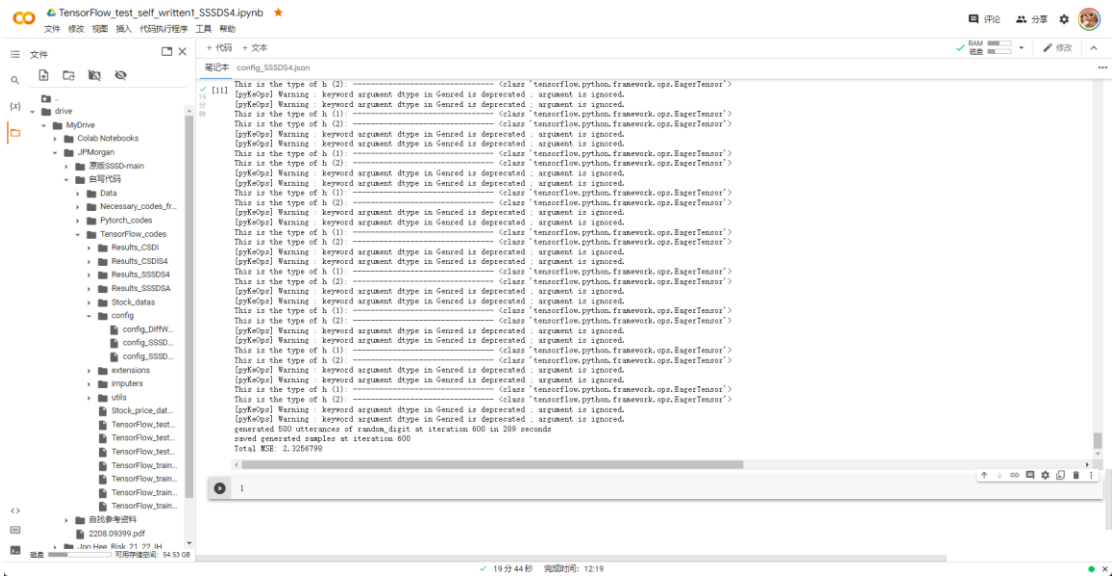

TensorFlow_train_self_written1_CSDI.ipynb
This is loop: 639 ...
Check shapes: (50, 14, 100) (50, 14, 100) (50, 14, 100)
(50, 14, 100)
net = SSSDS4Imputer(**model_config)
net.load_weights('/content/drive/MyDrive/JPMorgan/自写代码/TensorFlow_codes/Results_SSSDS4/Mujoco/train_90/T200_beta00.0001_betaT0.02/1.weight',by_name=False)
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f655cdb4ac0>
29 分 16 秒 完成时间: 00:28



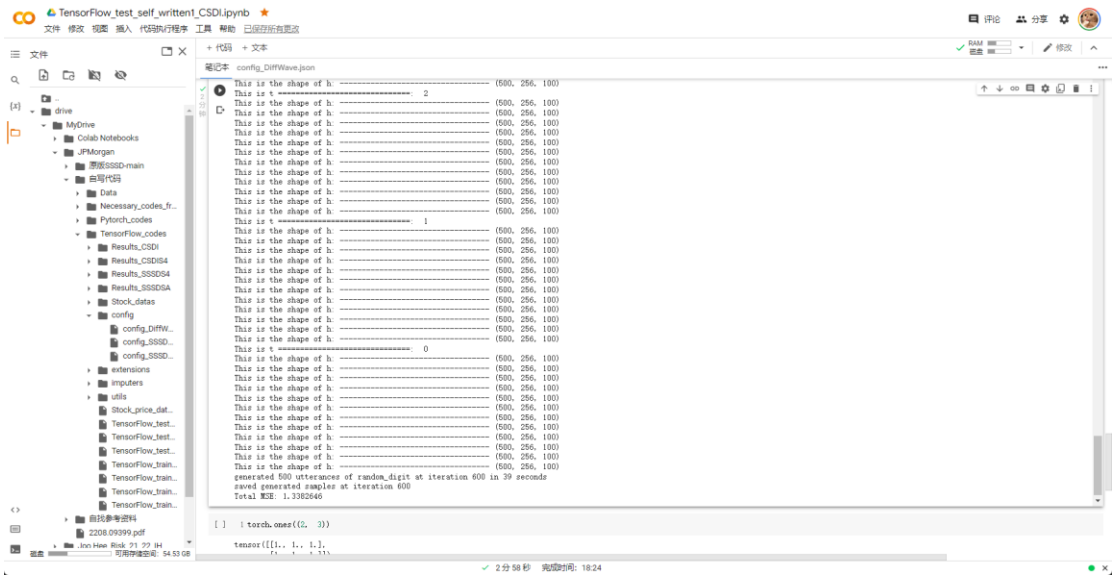
-The following experiments are under the only missing portions of the signal diffusion ("only_generate_missing" = 1)

1. Random missing (“masking” = ”rm”)

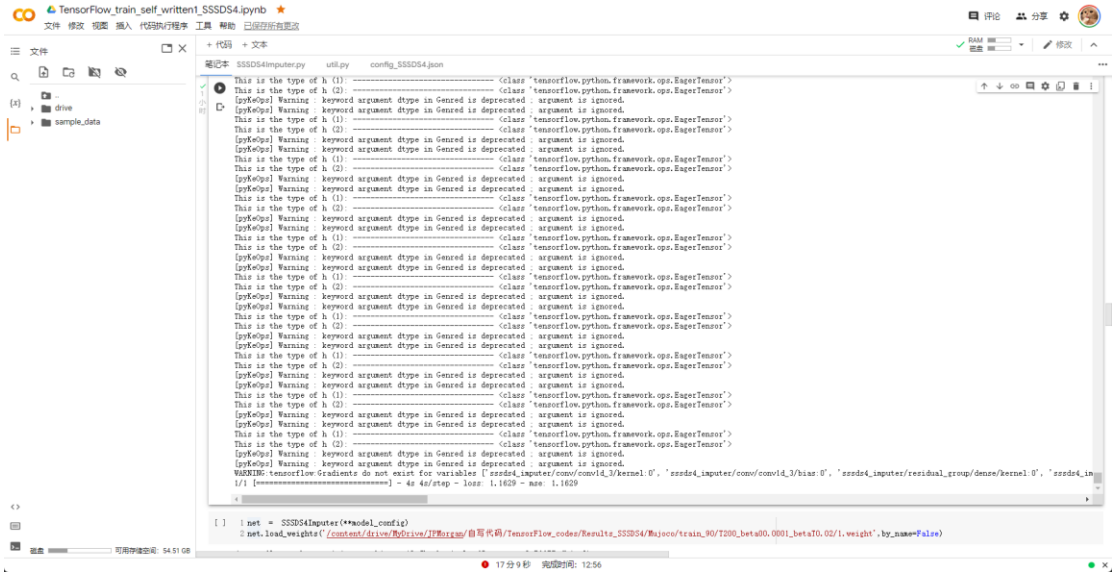


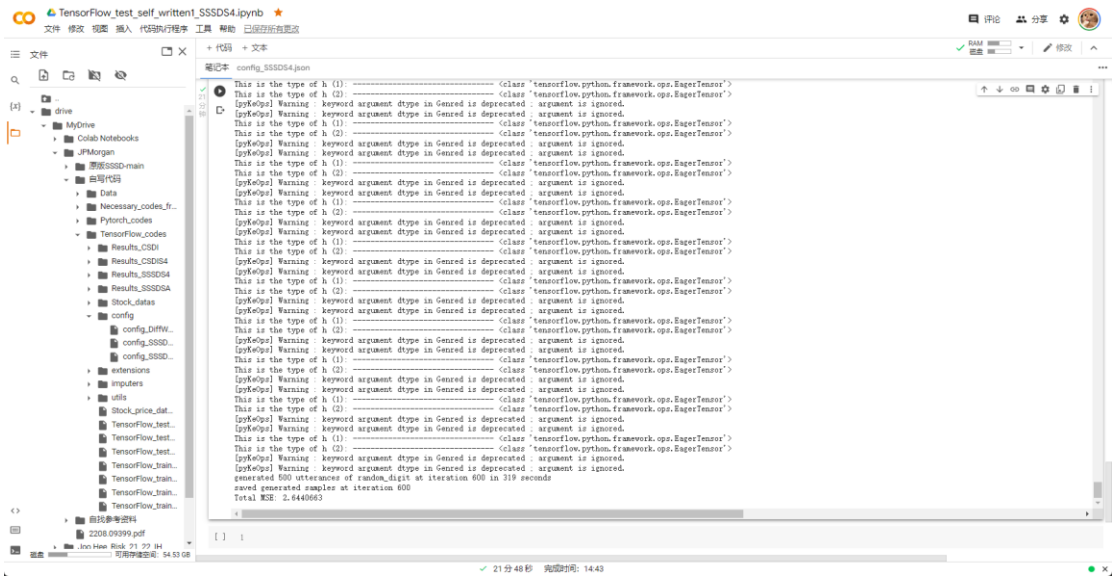

TensorFlow_test_self_written1_SSSDS4.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助
评论 分享
+ 代码 + 文本
笔记本 config_SSSDS4.json
drive
MyDrive
Colab Notebooks
JPMorgan
原版SSSD-main
自写代码
Data
Necessary_codes_fr...
Pytorch_codes
TensorFlow_codes
Results_CSDI
Results_CSDIS4
Results_SSSDS4
Results_SSSDSA
Stock_datas
config
config_DiffW...
config_SSSD...
config_SSSD...
extensions
imputers
utils
Stock_price_dat...
TensorFlow_test...
TensorFlow_test...
TensorFlow_test...
TensorFlow_train...
TensorFlow_train...
TensorFlow_train...
TensorFlow_train...
自找参考资料
2208.09399.pdf
可用存储空间: 54.53 GB
[11]
[pyKeOps] Warning : keyword argument dtype in Genred is deprecated : argument is ignored.
This is the type of h (2): ---------- <class 'tensorflow.python.framework.ops.EagerTensor'>
This is the type of h (1): ---------- <class 'tensorflow.python.framework.ops.EagerTensor'>
generated 500 utterances of random_digit at iteration 600 in 289 seconds
saved generated samples at iteration 600
Total MSE: 2.3256790
19 分 44 秒 完成时间: 12:19

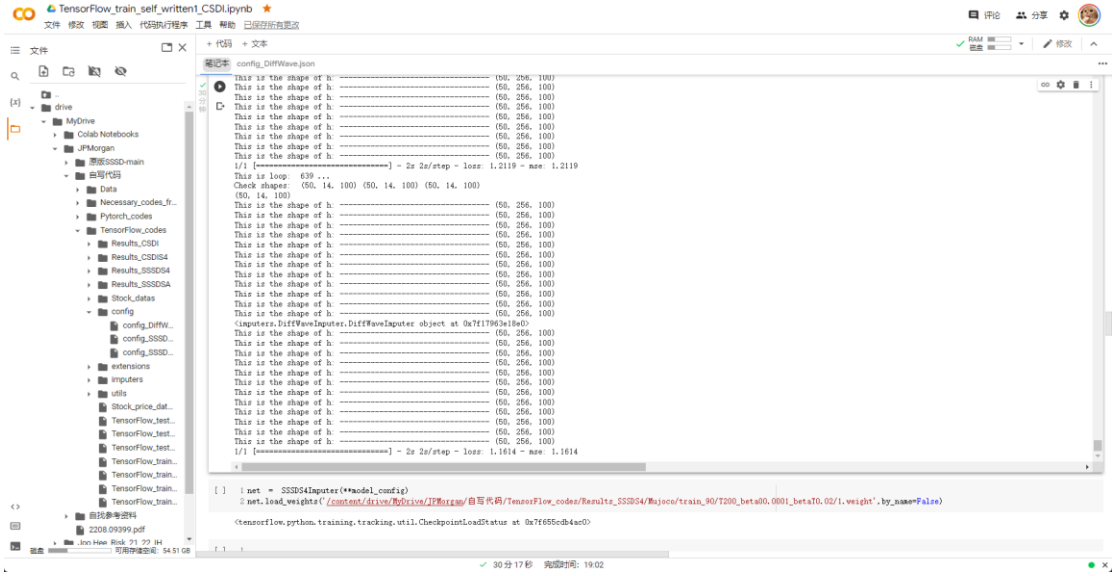

TensorFlow_train_self_written1_CSDI.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助 已保存所有更改
评论 分享
+ 代码 + 文本
重新连接
笔记本 config_DiffWave.json
连接到某个代码执行程序可启用文件浏览功能。
This is the shape of h: ---------- (50, 256, 100)
This is loop: 639 ...
Check shapes: (50, 14, 100) (50, 14, 100) (50, 14, 100)
(50, 14, 100)
<imputerz.DiffWaveImputer.DiffWaveImputer object at 0x7fc0f6819580>
1/1 [==============================] - 2s 2s/step - loss: 0.7568 - mse: 0.7568
net = SSSDS4Imputer(**model_config)
net.load_weights('/content/drive/MyDrive/JPMorgan/自写代码/TensorFlow_codes/Results_SSSDS4/Mujoco/train_90/T200_beta00.0001_betaT0.02/1.weight',by_name=False)
30 分 32 秒 完成时间: 17:34

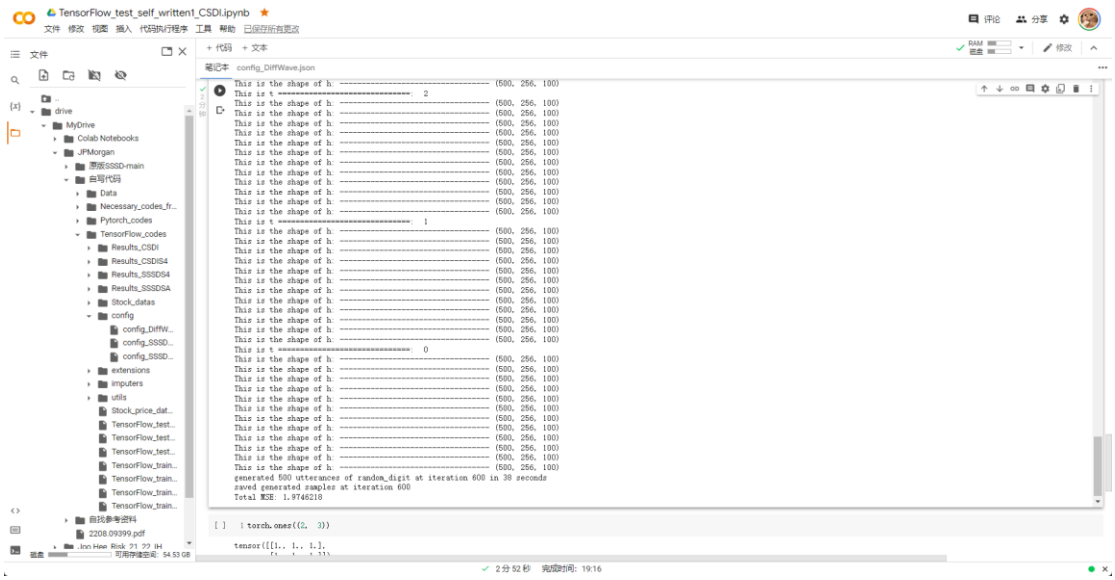

TensorFlow_test_self_written1_CSDI.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助 已保存所有更改
评论 分享
+ 代码 + 文本
笔记本 config_DiffWave.json
This is the shape of h: ---------- (500, 256, 100)
This is t ---------- 2
This is t ---------- 1
This is t ---------- 0
generated 500 utterances of random_digit at iteration 600 in 39 seconds
saved generated samples at iteration 600
Total MSE: 1.3382646
torch.ones((2, 3))
tensor([[1., 1., 1.],
2 分 58 秒 完成时间: 18:24

2. Missing not at random (“masking” = ”mnr”)

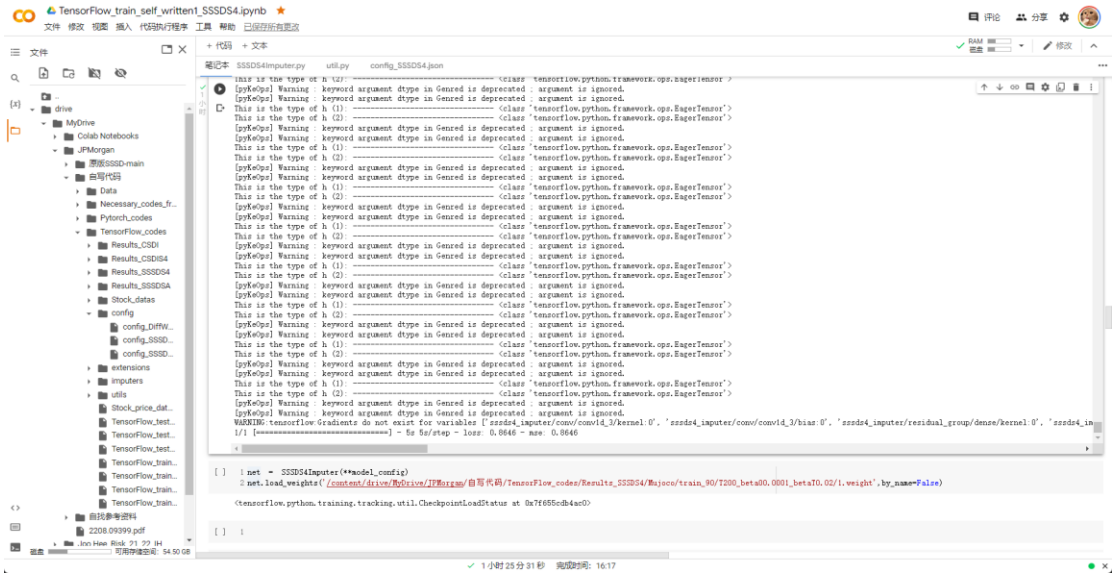


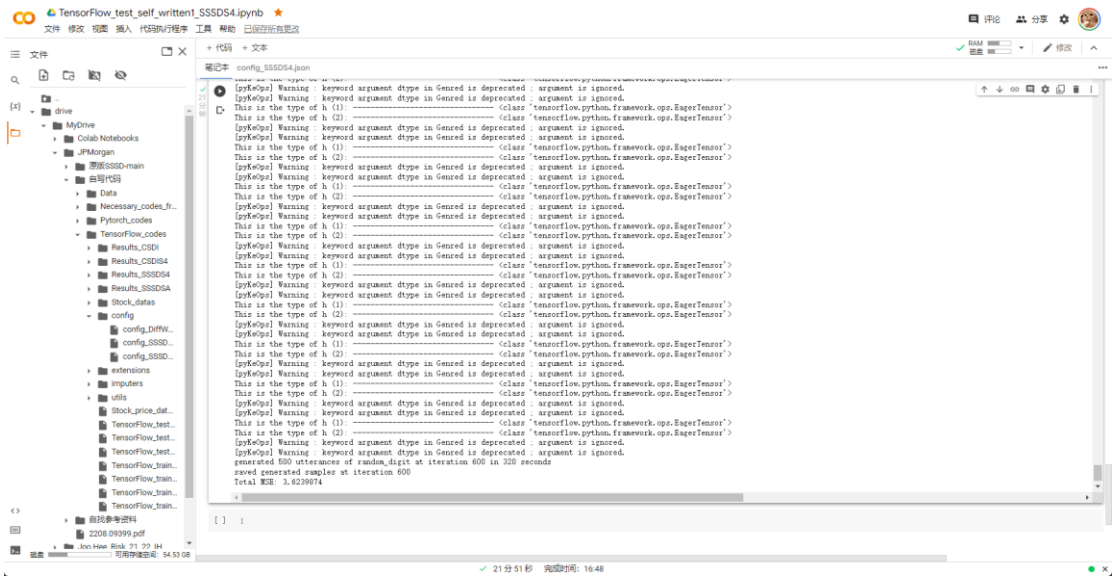


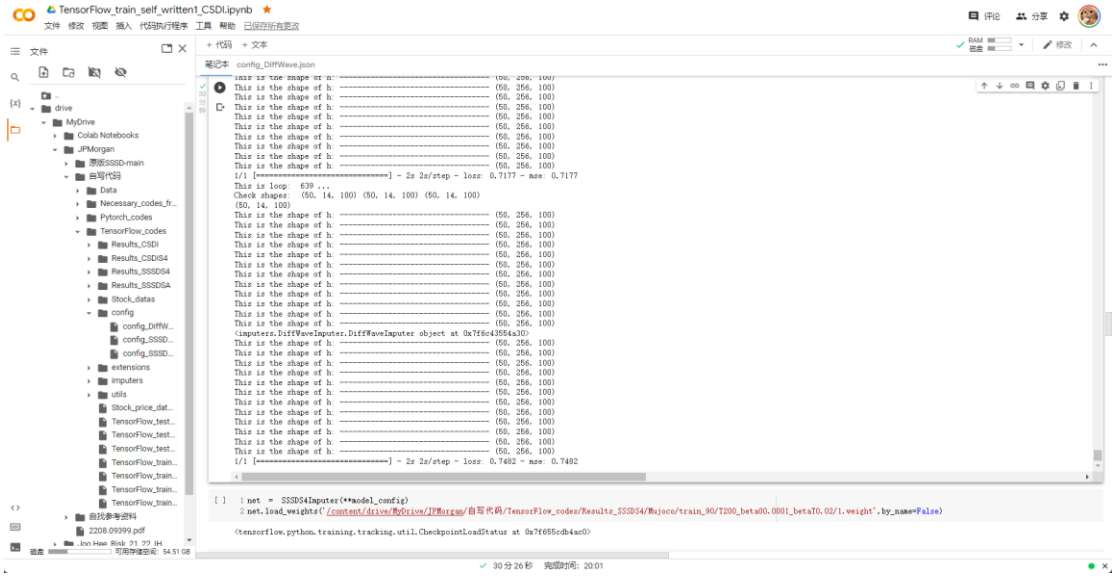


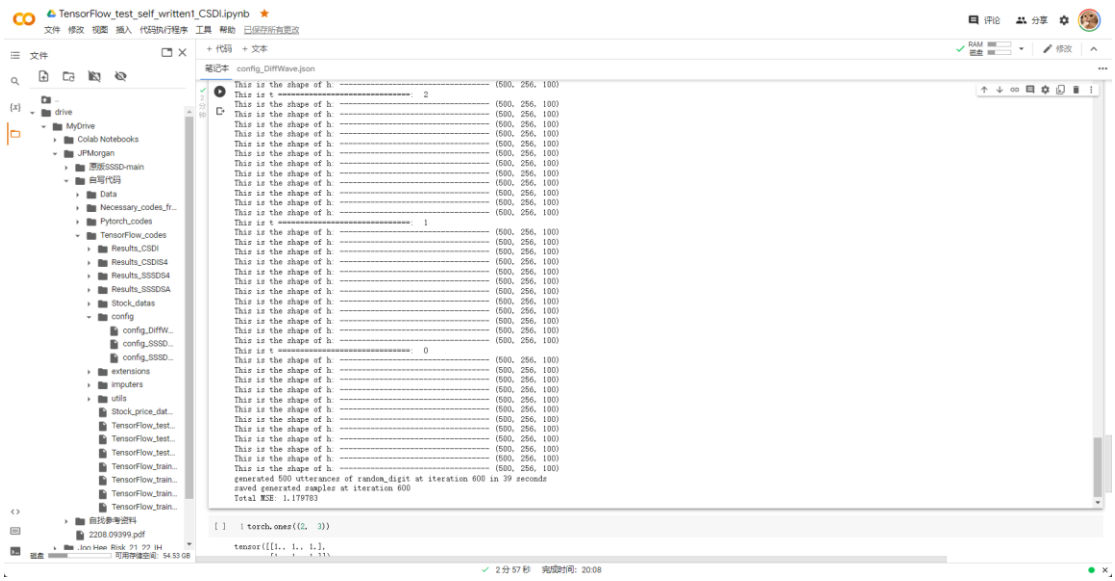


3. Blackout missing (“masking” = ”bm”)




TensorFlow_test_self_written1_SSSDS4.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助 已保存所有更改
+ 代码 + 文本
笔记本 config_SSSDS4.json
[pyKeOps] Warning : keyword argument dtype in Genred is deprecated : argument is ignored.
This is the type of h (1): <class 'tensorflow.python.framework.ops.EagerTensor'>
This is the type of h (2): <class 'tensorflow.python.framework.ops.EagerTensor'>
generated 500 utterances of random_digit at iteration 600 in 320 seconds
saved generated samples at iteration 600
Total MSE: 3.6239074
21 分 51 秒 完成时间: 16:48


TensorFlow_train_self_written1_CSDI.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助 已保存所有更改
+ 代码 + 文本
笔记本 config_DiffWave.json
This is the shape of h: (50, 256, 100)
1/1 [==============================] - 2s 2s/step - loss: 0.7177 - mse: 0.7177
This is loop: 639 ...
Check shapes: (50, 14, 100) (50, 14, 100) (50, 14, 100)
(50, 14, 100)
<imputers.DiffWaveImputer.DiffWaveImputer object at 0x7f6c43654a30>
1/1 [==============================] - 2s 2s/step - loss: 0.7482 - mse: 0.7482
net = SSSDS4Imputer(**model_config)
net.load_weights('/content/drive/MyDrive/JPMorgan/自写代码/TensorFlow_codes/Results_SSSDS4/Mujoco/train_90/T200_beta00.0001_betaT0.02/1.weight',by_name=False)
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f655cdb4ac0>
30 分 26 秒 完成时间: 20:01


TensorFlow_test_self_written1_CSDI.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助 已保存所有更改
+ 代码 + 文本
笔记本 config_DiffWave.json
This is the shape of h: (500, 256, 100)
This is t ==========: 2
This is t ==========: 1
This is t ==========: 0
generated 500 utterances of random_digit at iteration 600 in 39 seconds
saved generated samples at iteration 600
Total MSE: 1.179783
torch.ones((2, 3))
tensor([[1., 1., 1.],
2 分 57 秒 完成时间: 20:08

Summary:

Table 1. MSE values in training and testing sets

	only_generate_missing = 0		only_generate_missing = 1	
	SSSDS4	**CSDI**	**SSSDS4**	**CSDI**
Rm (training)	1.0113	0.7900	0.9832	0.7568
Mnr (training)	1.1144	1.0737	1.1629	1.1614
Bm (training)	0.9667	0.8025	0.8646	0.7482
Rm (testing)	2.3512	1.5458	2.3257	1.3383
Mnr (testing)	2.2681	1.8009	2.6441	1.9746
Bm (testing)	3.6020	1.1814	3.6240	1.1798

Using the given "Mujoco" training and testing datasets, in the experiments with these small-scale hyper-parameters, CSDI realized both better training and testing results than the SSSDS4 under the mean squared error (MSE) criterion.

The potential reason for this result may be the very small-scale hyper-parameters. Under very small number of iterations and small number of diffusion steps, the advantages of SSSDS4 cannot be realized. To really replicate the original paper's results, we need to use much larger hyper-parameters. This work can be done later with much more time. Until now, two models can run, be saved, and loaded successfully.

- **(b)**

We have downloaded all the required stock price data from Yahoo Finance through the yfinance API. The specific stocks included in the current Dow Jones 30, EuroStoxx 50 and Hang Seng are determined by researching these indexes with the public information (e.g. http://www.aastocks.com/, https://finance.yahoo.com/, https://markets.businessinsider.com/). The list of the determined included stocks is the following (also shown in the Stock_price_data.ipynb) and each stock's name is matched to the Yahoo's ticker (in the brackets):

Table 2. Downloaded index data from Yahoo Finance

Dow Jones 30	EuroStoxx 50	Hang Seng
American Express (axp)	AB InBev (BUD)	CKH HOLDINGS (0001.HK)
Amgen (amgn)	adidas (ADS.DE)	CLP HOLDINGS (00002.HK)
Apple (aapl)	Adyen B.V. Parts Sociales (ADYEN.AS)	HK & CHINA GAS (00003.HK)
Boeing (ba)	Ahold Delhaize (AD.AS)	HSBC HOLDINGS (00005.HK)
Caterpillar (cat)	Air Liquide (AI.PA)	POWER ASSETS (00006.HK)
Chevron (cvx)	Airbus (AIR.PA)	HANG SENG BANK (00011.HK)
Cisco Systems (csco)	Allianz (ALV.DE)	HENDERSON LAND (00012.HK)
Coca-Cola (ko)	ASML NV (ASML)	SHK PPT (00016.HK)
Dow, Inc. (dow)	AXA (CS.PA)	NEW WORLD DEV (00017.HK)
Goldman Sachs (gs)	BASF (BAS.DE)	GALAXY ENT (00027.HK)
Home Depot (hd)	Bayer (BAYZF)	MTR CORPORATION (00066.HK)

Honeywell International (hon)	BBVA (BBVA)	HANG LUNG PPT (00101.HK)
Intel (intc)	BMW (BMW.DE)	GEELY AUTO (00175.HK)
International Business Machines (ibm)	BNP Paribas (BNP.PA)	ALI HEALTH (00241.HK)
Johnson & Johnson (jnj)	CRH (CRH)	CITIC (00267.HK)
JPMorgan Chase (jpm)	Danone (BN.PA)	WH GROUP (00288.HK)
McDonald's (mcd)	Deutsche Börse (DB1.DE)	CHINA RES BEER (00291.HK)
3M (mmm)	Deutsche Post (DPW.DE)	OOIL (00316.HK)
Merck (mrk)	Deutsche Telekom (DTE.DE)	TINGYI (00322.HK)
Microsoft (msft)	Enel (ENL.DE)	SINOPEC CORP (00386.HK)
Nike (nke)	Eni (E)	HKEX (00388.HK)
Procter & Gamble (pg)	EssilorLuxottica (EL.PA)	TECHTRONIC IND (00669.HK)
Salesforce.com (crm)	Flutter Entertainment (PDYPY)	CHINA OVERSEAS (00688.HK)
Travelers (trv)	Iberdrola (IBDRY)	TENCENT (00700.HK)
UnitedHealth Group (unh)	Inditex (IDEXY)	CHINA UNICOM (00762.HK)
Verizon (vz)	Infineon (IFX.DE)	LINK REIT (00823.HK)
Visa (v)	ING Group (ING)	PETROCHINA (00857.HK)
Walgreens Boots Alliance (wba)	Intesa Sanpaolo (ISP.MI)	XINYI GLASS (00868.HK)
Walmart (wmt)	Kering (KER.PA)	ZHONGSHENG HLDG (00881.HK)
Walt Disney (dis)	Linde (LIN)	CNOOC (00883.HK)
	L'Oréal (OR.PA)	CCB (00939.HK)
	LVMH Moet Hennessy Louis Vuitton (LVMUY)	CHINA MOBILE (00941.HK)
	Mercedes-Benz Group (MBG.DE)	LONGFOR GROUP (00960.HK)
	Münchener Rückversicherungs-Gesellschaft (MUV2.MI)	XINYI SOLAR (00968.HK)
	Nokia (NOK)	SMIC (00981.HK)
	Nordea Bank Abp Registered Shs (NDA-FI.HE)	LENOVO GROUP (00992.HK)
	Pernod Ricard (PRNDY)	CKI HOLDINGS (01038.HK)
	Prosus (PRX.AS)	HENGAN INT'L (01044.HK)
	SAFRAN (SEJ1.SG)	CHINA SHENHUA (01088.HK)
	Sanofi (SNW.SG)	CSPC PHARMA (01093.HK)
	Santander (SAN)	CHINA RES LAND (01109.HK)
	SAP (SAP)	CK ASSET (01113.HK)
	Schneider Electric (SU.PA)	SINO BIOPHARM (01177.HK)
	Siemens (SIE.DE)	CHINA RES MIXC (01209.HK)
	Stellantis (STLA)	BYD COMPANY (01211.HK)
	TotalEnergies (TTFNF)	AIA (01299.HK)
	VINCI (DG.PA)	CHINAHONGQIAO (01378.HK)
	Volkswagen (VOW3.DE)	ICBC (01398.HK)
	Vonovia (VNA.DE)	XIAOMI-W (01810.HK)
		BUD APAC (01876.HK)
		SANDS CHINA LTD (01928.HK)

		CHOW TAI FOOK (01929.HK)
		WHARF REIC (01997.HK)
		COUNTRY GARDEN (02007.HK)
		ANTA SPORTS (02020.HK)
		WUXI BIO (02269.HK)
		SHENZHOU INTL (02313.HK)
		PING AN (02318.HK)
		MENGNIU DAIRY (02319.HK)
		LI NING (02331.HK)
		SUNNY OPTICAL (02382.HK)
		BOC HONG KONG (02388.HK)
		CHINA LIFE (02628.HK)
		ENN ENERGY (02688.HK)
		MEITUAN-W (03690.HK)
		HANSOH PHARMA (03692.HK)
		CM BANK (03968.HK)
		BANK OF CHINA (03988.HK)
		CG SERVICES (06098.HK)
		HAIER SMARTHOME (06690.HK)
		HAIDILAO (06862.HK)
		JD-SW (09618.HK)
		NONGFU SPRING (09633.HK)
		BIDU-SW (09888.HK)
		BABA-SW (09988.HK)
		NTES-S (09999.HK)

Next, we have removed the stocks without a 10-year history. The whole study time period is from 2013-01-23 to 2023-01-20, daily basis. If the stock has no price data on 2013-01-23 then we confirm that this stock does not have sufficient historical data and we remove it. After this cleaning work, there are 29 stocks kept in Dow Jones 30, 17 stocks kept in EuroStoxx 50 and 58 stocks kept in Hang Seng. The ticker list of these kept stocks is the following:

Table 3. Kept stock tickers in three indexes

Dow Jones 30	EuroStoxx 50	Hang Seng
['axp', 'amgn', 'aapl', 'ba', 'cat', 'cvx', 'csco', 'ko', 'gs', 'hd', 'hon', 'intc', 'ibm', 'jnj', 'jpm', 'mcd', 'mmm','mrk', 'msft', 'nke', 'pg', 'crm', 'trv', 'unh', 'vz', 'v', 'wba','wmt', 'dis']	['BUD', 'ASML', 'BAYZF', 'BBVA', 'CRH', 'E','IBDRY', 'IDEXY', 'ING', 'LIN', 'LVMUY', 'NOK', 'PRNDY', 'SAN', 'SAP','STLA', 'TTFNF']	'0001.HK', '0002.HK', '0003.HK', '0005.HK','0006.HK', '0011.HK', '0012.HK', '0016.HK', '0017.HK', '0027.HK', '0066.HK', '0101.HK', '0175.HK', '0241.HK', '0267.HK', '0291.HK','0316.HK', '0322.HK', '0386.HK', '0388.HK', '0669.HK', '0688.HK','0700.HK', '0762.HK', '0823.HK', '0857.HK', '0868.HK', '0881.HK','0883.HK', '0939.HK', '0941.HK', '0960.HK', '0981.HK', '0992.HK','1038.HK', '1044.HK', '1088.HK', '1093.HK', '1109.HK', '1177.HK', '1211.HK', '1299.HK', '1378.HK', '1398.HK', '1928.HK',

		'1929.HK','2007.HK', '2020.HK', '2313.HK', '2318.HK', '2319.HK', '2331.HK','2382.HK', '2388.HK', '2628.HK', '2688.HK', '3968.HK', '3988.HK'

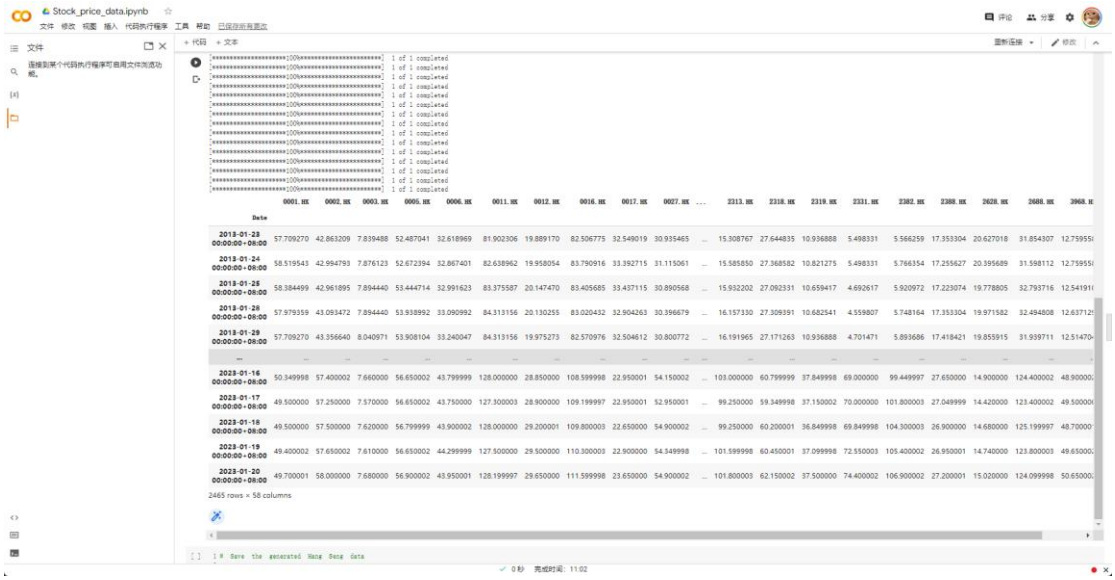
More details can be found in “Stock_price_data.ipynb”.

- **(c)**

Yes, we have found that in the original downloaded dataframes even we have set the same start date and end date. The rows of available time points from three indexes’ dataframes are different. There are 2517 rows of data for the stocks in Dow Jones 30.

There are 2517 rows of data for the stocks in EuroStoxx 50.

There are 2465 rows of data for the stocks in Hang Seng.



To find out the specific mismatching dates among these three indexes, we design a "Date mismatch check" which can be divided into three groups: (1) Dow 30 and Stoxx 50 (check again, Dow → Stoxx and Dow ← Stoxx) (2) Dow 30 and Hang Seng (check again, Dow → Hang Seng and Dow ← Hang Seng) (3) Stoxx 50 and Hang Seng (check again, Stoxx → Hang Seng and Stoxx ← Hang Seng). In each group, we check the mismatched dates with each other. If there are some dates in one index rather than in the other index, then we print it out and count the number of them. After the "Date mismatch check", we get the following results:

Table 4. Matched and mismatched dates among three indexes

Dow has rather than Stoxx	**Dow has rather than Hang Seng (112 in total)**	**Stoxx has rather than Hang Seng**
None	2013-02-11, 2013-02-12 2013-02-13, 2013-04-01 2013-04-04, 2013-05-01 2013-05-17, 2013-06-12 2013-07-01. 2013-09-20 2013-10-01, 2013-10-14 2013-12-26. 2014-01-31 2014-02-03, 2014-04-21 2014-05-01, 2014-05-06 2014-06-02, 2014-07-01 2014-09-09, 2014-10-01 2014-10-02, 2014-12-26 2015-02-19, 2015-02-20 2015-04-06, 2015-04-07 2015-05-01, 2015-07-01 2015-09-03, 2015-09-28 2015-10-01, 2015-10-21 2016-02-08, 2016-02-09	As we have confirmed that the Dates in Dow and Stoxx are same. So, the results here are same as the ones in the left cell.

	2016-02-10, 2016-03-28 2016-04-04, 2016-05-02 2016-06-09, 2016-07-01 2016-09-16, 2016-10-10 2016-12-27, 2017-01-30 2017-01-31, 2017-04-04 2017-04-17, 2017-05-01 2017-05-03, 2017-05-30 2017-10-02, 2017-10-05 2017-12-26, 2018-02-16 2018-04-02, 2018-04-05 2018-05-01, 2018-05-22 2018-06-18, 2018-07-02 2018-09-25, 2018-10-01 2018-10-17, 2018-12-26 2019-02-05, 2019-02-06 2019-02-07, 2019-04-05 2019-04-22, 2019-05-01 2019-05-13, 2019-06-07 2019-07-01, 2019-10-01 2019-10-07, 2019-12-26 2020-01-27, 2020-01-28 2020-04-13, 2020-04-30 2020-05-01, 2020-06-25 2020-07-01, 2020-10-01 2020-10-02, 2020-10-13 2020-10-26, 2021-02-12 2021-04-05, 2021-04-06 2021-05-19, 2021-06-14 2021-07-01, 2021-09-22 2021-10-01, 2021-10-13 2021-10-14, 2021-12-27 2022-02-01, 2022-02-02 2022-02-03, 2022-04-05 2022-04-18, 2022-05-02 2022-05-09, 2022-06-03 2022-07-01, 2022-09-12 2022-10-04, 2022-12-27	
Stoxx has rather than Dow	**Hang Seng has rather than Dow (60 in total)**	**Hang Seng has rather than Stoxx**
None	2013-02-18, 2013-05-27 2013-07-04, 2013-09-02 2013-11-28, 2014-01-20 2014-02-17, 2014-05-26	As we have confirmed that the Dates in Dow and Stoxx are same. So the results here are same as the ones in the left cell.

	2014-07-04, 2014-09-01 2014-11-27, 2015-01-19 2015-02-16, 2015-07-03 2015-09-07, 2015-11-26 2016-01-18, 2016-02-15 2016-05-30, 2016-07-04 2016-09-05, 2016-11-24 2017-01-16, 2017-02-20 2017-05-29, 2017-07-04 2017-09-04, 2017-11-23 2018-01-15, 2018-05-28 2018-07-04, 2018-09-03 2018-11-22, 2018-12-05 2019-01-21, 2019-02-18 2019-05-27, 2019-07-04 2019-09-02, 2019-11-28 2020-01-20, 2020-02-17 2020-05-25, 2020-07-03 2020-09-07, 2020-11-26 2021-01-18, 2021-05-31 2021-07-05, 2021-09-06 2021-11-25, 2021-12-24 2022-01-17, 2022-02-21 2022-05-30, 2022-06-20 2022-07-04, 2022-09-05 2022-11-24, 2023-01-16	

- **(d)**

Yes, if we would like to apply Alcaraz's model. The basic assumptions are following the assumptions of the diffusion model with conditional variants (the additional information we have, the imputation mask). These includes that (1) probability (with learnable parameters) of the diffusion should follow a normal distribution. Also, (2) as written in the paper that the probability's normal distribution is with a diagonal covariance matrix, it means that the diffusion values each other have no correlation (linear). (3) The diffusion process is a Markovian process which means that we can only get the value at time point (t) from the value at the time point (t-1) with no relation to the more previous values.

During the holidays with the missing data, when we input financial data into Alcaraz's model, it can be translated as that (1) the imputed stock prices' imputation probability should follow a normal distribution. (2) The generation of a stock's price at time point (t) has no correlation with the generation of the stock's price at other time points like (t-1), (t+1) and others. (3) There is no memory of the generated imputed data sequence in the holiday. It means that we can only generate the today's (t) imputed data based on the yesterday's (t-1) data. The yesterday's data can be also a generated one or the true historical data. In this case, we need

the assumption that "if the holidays were suddenly declared by law not holidays anymore, the behavior of the investors would remain the same on the day before the holiday". The reason is that the first generated imputed data is based on the real happened data one day before the holiday's starting date. So, if we would like to apply Alcaraz's model straightforwardly without a new training process then we need to keep this assumption.

However, we can also drop this assumption if we use the data after the holidays were declared not holidays by training the model again with the new data. By this way, the model will tune the parameters based on the investors' new behaviors. Then we can use this new trained model to do the forecast or impute the missing data. And we may need a large size of training data after this declaration to keep a relatively high forecast accuracy.

To test the assumption (1), we can change the assumed distribution in codes. In the original codes, "z = std_normal(audio.shape)" and "std_normal" is from "torch.normal(0, 1, size=size)". Here, we can change it to another distribution like an exponential distribution ("Tensor.exponential_"). After we do this, we input the financial testing set to calculate the MSE and compare this with the one with the normal distribution. To test the assumption (2), when we calculate the diffusion hyperparameters like "Alpha_bar" and "Beta_tilde", the original codes just use the "Alpha_bar" and "Beta_tilde" at time point (t-1) to infer them at time point (t): "Alpha_bar[t] *= Alpha_bar[t - 1]" and "Beta_tilde[t] *= (1 - Alpha_bar[t - 1]) / (1 - Alpha_bar[t])". We may create a new relation here to add these parameters at previous time points like time points (t-2) or (t-3). For instance, we design that "Alpha_bar[t] *= Alpha_bar[t - 1]* Alpha_bar[t - 2]". To test the assumption (3), if we have changed the relations in assumption (2) then this assumption has also been changed. To test this assumption straightforwardly, with the financial data, we can find some real cases that the holidays have been canceled. Based on the historical data, if there is one holiday has been canceled, we can use the historical data before this canceling to train the model then we apply the model to impute the missing data in the holidays. After the holiday has been canceled, we use the imputed data to compare with the real happened data in the former holiday and calculate MSE (in this case, the behavior of the investors remains the same). After that, we use the data after the holiday's canceling to train the model again then apply the model to impute the "holiday" (not anymore). Next, we calculate the MSE again and compare the two MSEs to see which one is smaller.

- **(e)**

No, SSSD algorithm performs better than CSDI only in Dow 30's stocks in most missing scenarios and CSDI performs better in Hang Seng's stocks. The approach is that using the same index's stocks as the training and testing sets to input into one algorithm (SSSDS4 or CSDI). To compare two algorithms, we use the Mean Squared Error (MSE) function as the objective function. As we have three indexes with three data missing scenarios ('Rm', 'Mnr' and 'Bm'), for each index, we test all the three missing scenarios to see in which case, one algorithm has a better performance than the other one.

In this process, we ignore the existing missing data but generate missing data by ourselves

with all ‘Rm’, ‘Mnr’ and ‘Bm’ conditions. By this way, we have the real data to calculate the MSE value and evaluate the model’s performance then compare two algorithms.

We repeat this process for 10 times (or more) with different random seeds to generate different training and testing set pairs to avoid any coincidence (the time sequence should be kept). For each process, we calculate the MSE value from the testing sets. Then after 10 times experiments, we calculate the average value of the generated 10 MSE values and the criterion is this average value. A smaller value shows a better performance. Also, the time series sequence should not be changed during this process.

Due to the limited time, we only have done one process as an instance. For all indexes, the first 80% historical data is used as the training set and the rest of latest 20% data is used as the testing set. More repeated processes should be done later for a more stable result. The machine learning framework used is the original Pytorch. For the hyper-parameters, only "n_iters": 150000 has been changed to 600 to save time. And “missing_k” has been adjusted to 90 in both SSSD and CSDI algorithms to compare them. The "in_channels" and "out_channels" have been adjusted based on the dimensions of the training and testing data sets. All hyper-parameters are consistent with all three indexes under all missing scenarios.

- `only_generate_missing == 1`

The mean MSE of all testing experiments using SSSDS4

	Dow 30	Stoxx 50	Hang Seng
Rm	38103.170	32070.158	11059.689
Mnr	38107.684	32086.475	11052.345
Bm	38118.434	32052.867	11060.863

The mean MSE of all testing experiments using CSDI

	Dow 30	Stoxx 50	Hang Seng
Rm	38113.066	32071.850	11045.801
Mnr	38117.660	32063.062	11050.241
Bm	38135.060	32057.389	11050.029

- `only_generate_missing == 0`

The mean MSE of all testing experiments using SSSDS4

	Dow 30	Stoxx 50	Hang Seng
Rm	38125.350	32063.809	11058.889
Mnr	38114.297	32116.291	11057.535
Bm	38096.740	32070.457	11067.503

The mean MSE of all testing experiments using CSDI

	Dow 30	Stoxx 50	Hang Seng
Rm	38137.990	32074.521	11057.562

Mnr	38126.920	32067.656	11064.073
Bm	38108.760	32062.998	11051.495

The MSE of both algorithms under every scenario is very small (lower than or around 1) but is very large in the testing set. So, we consider that both models may be overfitting with these financial data sets (small-size data sets).

Based on the performances above, we can summary that applying diffusion to missing portions of the signal (only_generate_missing = 1), SSSDS4 model realizes a better performance than the CSDI one in all missing scenarios of Dow 30, "Rm" and "BM" missing scenarios of Stoxx 50 and none better cases in Hang Seng. Using the all-sample diffusion (only_generate_missing = 0), SSSDS4 model realizes a better performance than the CSDI one in all missing cases of Dow 30, "Rm" missing scenario of Stoxx 50, "Mnr" missing scenario of Hang Seng.

In the real world, for the financial stock data, we should focus more on the "Bm" case. In this case, the data at continuous time points are missing for all stocks in a specific market which is more like to be a "holiday period" for all stocks in an index. Also, the "all sample diffusion" is more suitable for the stock's data as there is much information in the non-missing proportion (non-holiday). Based on these two assumptions, we conclude that (1) SSSDS4 realizes a better performance than the CSDI for Dow 30's stocks (US market) (2) CSDI realizes a better performance for Stoxx 50's stocks (European market) and Hang Seng's stocks (Hong Kong market).

- **(f)**

We can use the idea from the Gap strategy trading (Dahlquist and Bauer, 2012, Mathuria and Bhakar, 2014). Due to the holiday happening, there should be market information overstock during the holiday period. This information should not be sufficiently reflected by the market in the trading day just one day before the holiday's starting date which means that the stock's information reflected by the market is not sufficient in the day before the holiday. During the holiday, the non-reflected information will be overstocked and will be reflected in the trading day just after the end of the holiday time. Normally, there should be a jump happens to reflect the overstocked information which can be either a UP jump or a Down jump (there are different definitions about Up or Down jump. E.g., the open price of the trading day after the holiday is higher than the highest price of the trading day before the holiday to be defined as a UP jump).

Figure 1. An instance of a UP jump

If a jump happens, the gap between the two prices can be uniformly (or other linear relations) allocated to the holiday days. For instance, the open price of the trading day after the holiday is 100 and the highest price of the trading day before the holiday is 90. There are 4 holiday days then we set the "real prices" during the holiday as 92, 94, 96, 98 respectively. By this way, we can calculate the loss function (e.g., MSE) and compare the imputed results with the allocated "real prices". We can also adjust this process by adjusting the definition of "jumps" like "a UP jump means that the next day's open price is higher than the previous day's close price".

Moreover, we can use a regression to fill up the holidays and the filled prices are set as the "real prices" (if no jump happens). We can use several trading days' price data before the holiday and several trading day's data after the holiday as the dependent variable Y. The time points (number of days) during this investment horizon (cover the holiday) are set as the independent variables X. Then we can run a regression (can be a simple linear one or Neural Network one) to fill up the missing data during the holiday and the predicted prices from this regression are used to fill this gap up. Then we see these filled data as the "real prices" and use them to compare with the values provided by the imputer algorithms.

- **(g)**

Yes, there are several methods to do this task. If we use the missing data imputation method as our model straightforwardly, we need to set up an assumption that "there are some macro-economic factors affecting all the UK, US and HK markets and the effects will spread among three markets". Under this assumption, we can build a time-series using the market data covering three markets' time zones. Let us say, we would like to forecast the price change of the trading hours on day (t) in HK. Then we should use the trading hours market data (can be the indexes like Dow 30 in US and Stoxx 50 in Europe) on day (t-1). Next, we can continue to forecast the price change of the HK market on day (t+1) based on the market data of US and Europe on day (t). By the way, we can build a time-series covering three markets with a time gap.

Figure 2. The time-series built by us

The reason for this time-series is that US, Europe, and Hong Kong markets are in different time zones. When the Hong Kong market is close (in day t), the European market will start and trade (in day t). Then the US market will start (in day t) after the European market is close (see, **Figure 3** for more time zone details). Next, the Hong Kong market will open in day (t+1).

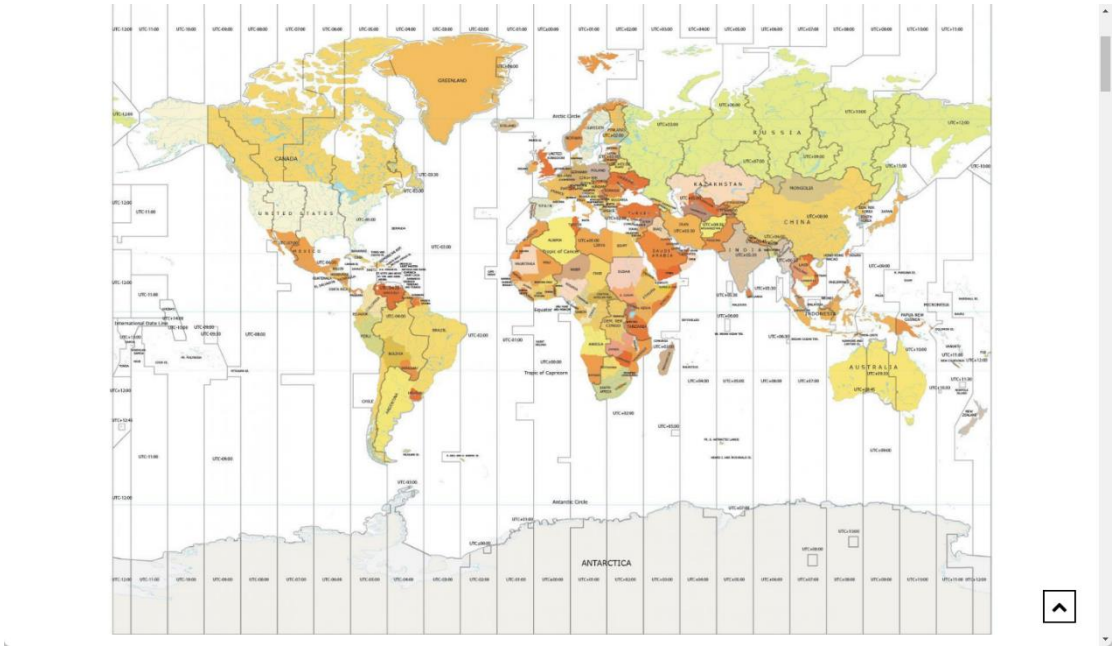


Figure 3. The world time zones

We can use this time gap among three markets especially the time gap between the US

market and the HK market as they are closely linked with time. We assume that there are some factors affecting all three markets so after the fluctuation happens in the US market in day (t). Then this fluctuation may be reflected by the Hong Kong market in day (t+1). To use the missing data imputation method, we firstly divide the whole data set into the training set and the testing set. The training set includes all the historical data covering three markets (see, **Figure 2**) and we use this set to train the network's configuration and parameters. In this process, we need to drop the HK market data (the stock price data which we would like to forecast) manually to create the missing parts. Because we have the real HK market data, we can calculate MSE during this process to train the algorithm. The scenario here should be "Bm" as the missing data is not randomly generated and to be forecasted. After this is done, we use the trained model to generate the forecasting data in the testing set. In the testing set, the missing data should be the HK market data on day (t+1). After the missing data has been imputed, we can calculate the price change which is our forecasted result.

However, we cannot use Alcaraz to impute all the missing data when we prepare our training set. It means that all the training set data should be real happened historical data rather than the imputed data. The reason is that we need to use the real data to train our algorithm to tune the correct parameters. The only part to be imputed is the missing part in the testing set. So, if there are holidays in the training set from the US or the European market causing missing data, we need to drop these samples as we cannot impute them.

There is another way to use the imputation method. As our purpose is to forecast the change of price rather than the whole prices during the trading hours. We can build another neural network (especially, RNN) to predict the "price changes" straightforwardly. So, the value of "price changes" of our target HK stock will be the dependent variable and the independent variables or features may be the "price changes" of the market indexes from the HK, US or European markets. Or we can use the indexes' component stocks' "price changes" as the features to train the algorithm. The layers used should include Long Short-Term Memory (LSTM) or RNN to "remember" the price changes from the previous trading days as the market may be not so effective. Or some ensemble algorithms like XGBoost or AdaBoost can be used to avoid overfitting issues by using pruning. In this case, we can use Alcaraz to impute all the missing data as we do not use Alcaraz to do the forecast but just generate sufficient samples.

- **Conclusion**

In this report, we have answered all questions in part 1. We show the results of our both transformed SSSDS4 and CSDI algorithms under TensorFlow framework. The relevant key points on algorithms' structure and innovative points have been introduced in the literature review section. After the required financial index data has been correctly collected and downloaded, we input them into both the original SSSDS4 and CSDI algorithms to train and test their performances. The results have been discussed. Also, we discuss multiple methods to deal with specific financial missing data questions with references.

- **Plan for the future research**

Due to the small-size hyper-parameters, the advantages of SSSD algorithm may not be

shown. We can increase the relevant parameters and the iteration numbers to train the algorithm again. Also, to work with the specific financial data missing, we may adjust the architecture of the algorithms and try to mitigate the overfitting. There are also other suggested architectures in part 2 which we can continue to try.

- **Written in the end**

Thank you very much for providing me with this opportunity to join this project. It is my pleasure and treasure to learn these advanced algorithms under two machine learning frameworks. During this period, I need to deal with lots of teaching assistant work and other research works in my university which slows down my efficiency. I have worked on this project with the squeezed time everyday and update the progress as soon as possible. Due to the limited time and calculation power, I can only complete the Part 1 with some reduced hyper-parameters. The good news is that technical questions have been overcome and all the transformed algorithms can run, be saved, and loaded successfully. So, we can obtain some useful results based on them. I really cherish this intern opportunity and believe I can contribute myself and learn more in this fantastic team with the friendly atmosphere. Later with more time, I can do more things and improve our obtained models and results. Thank you very much.

References:

1. Alcaraz, J.M.L. and Strodthoff, N., 2022. Diffusion-based time series imputation and forecasting with structured state space models. arXiv preprint arXiv:2208.09399.
2. Tashiro, Y., Song, J., Song, Y. and Ermon, S., 2021. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. Advances in Neural Information Processing Systems, 34, pp.24804-24816.
3. Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A. and Ré, C., 2021. Combining recurrent, convolutional, and continuous-time models with linear state space layers. Advances in neural information processing systems, 34, pp.572-585.
4. Dahlquist, J.R. and Bauer, R.J., 2012. Technical Analysis of Gaps: Identifying Profitable Gaps for Trading. FT Press.
5. Mathuria, P. and Bhakar, R., 2014. Info-gap approach to manage GenCo's trading portfolio with uncertain market returns. IEEE Transactions on Power Systems, 29(6), pp.2916-2925.
6. 阿斯達克財經網 (阿思達克財經網) aastocks.com - 免費即時股票及港股報價 HK free stock quote (no date) 阿斯達克財經網 (阿思達克財經網) AASTOCKS.com - 免費即時股票及港股報價 HK Free Stock Quote. Available at: http://www.aastocks.com/ (Accessed: January 30, 2023).
7. Yahoo Finance - Stock Market Live, quotes, Business & Finance News (no date) Yahoo! Finance. Yahoo! Available at: https://finance.yahoo.com/ (Accessed: January 30, 2023).
8. Markets insider: Stock market news, Realtime quotes and Charts (no date) Business Insider. Business Insider. Available at: https://markets.businessinsider.com/ (Accessed: January 30, 2023).