MEMPUTE V4

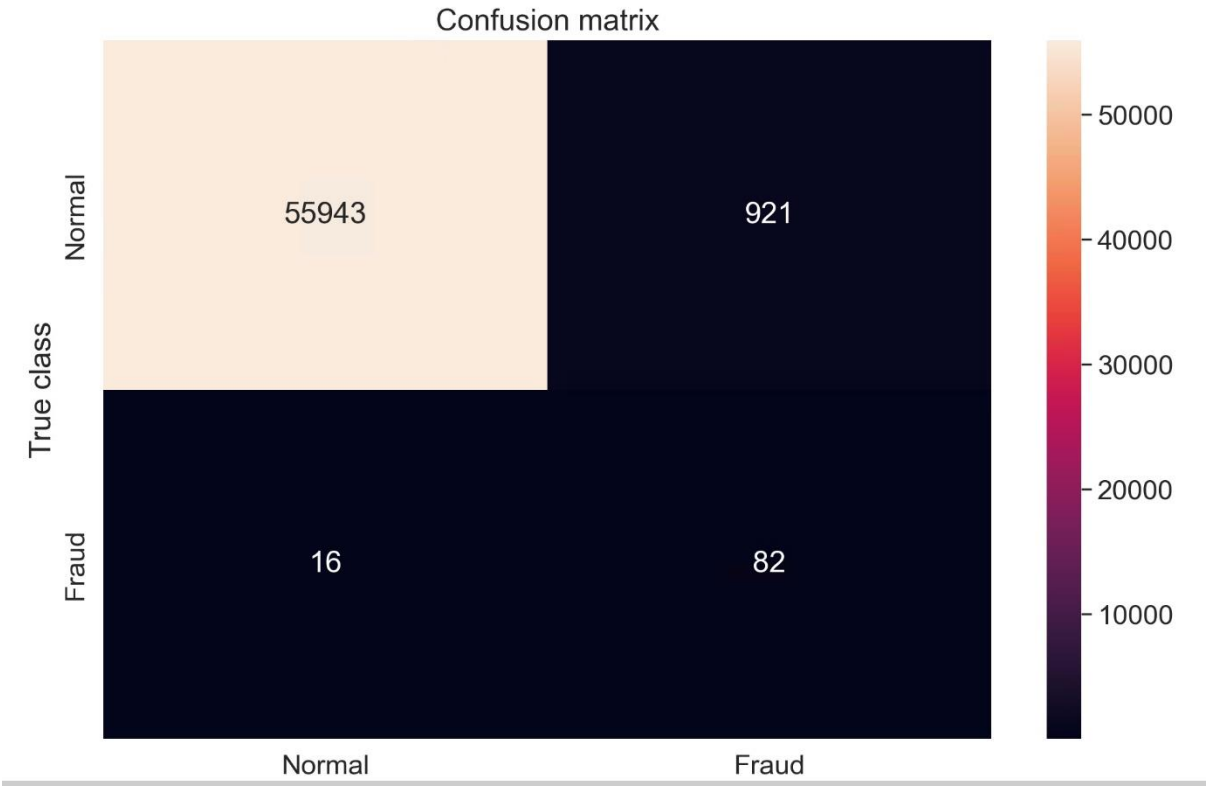
- mempute time series model -

mempute는 트랜스포머를 개선한 시계열 신경망 모델으로서 트랜스포머 계열 GPT대비 우수한 패턴 탐지 능력과 정확하고 빠른 학습 속도, 적은 메모리 사용량으로 GPT 모델의 대안으로서 현재 대량 코퍼스 학습에 따른 전력문제, gpu 수급문제를 원천적으로 해결할 수 있는 모델이다.

LLM Foundation Model

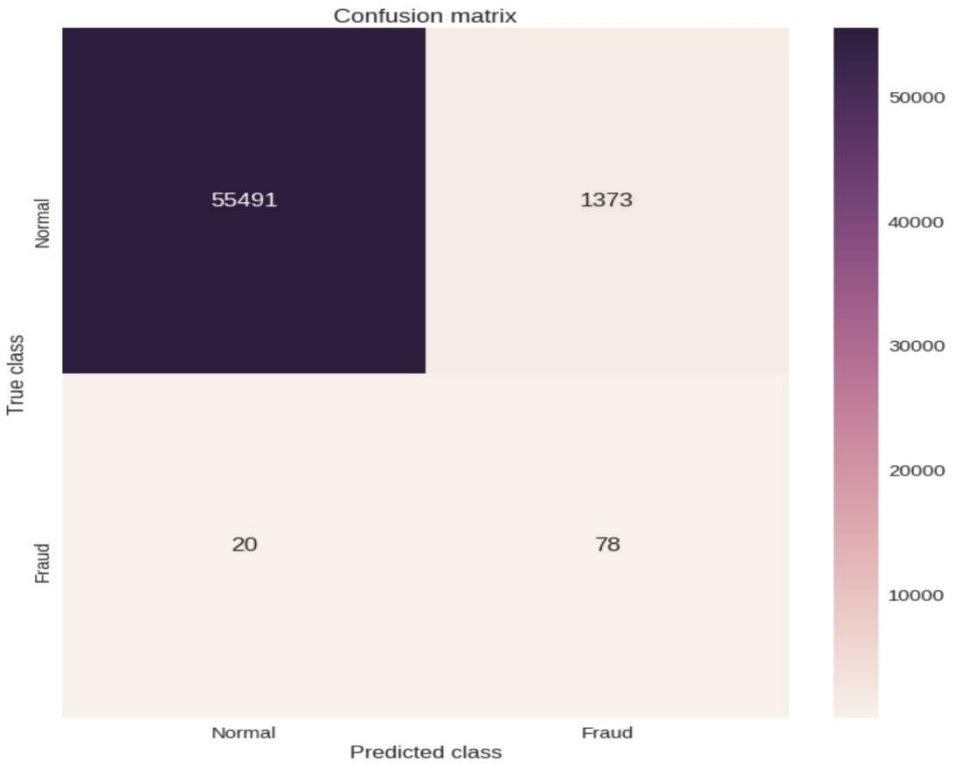
- KorQuAD 말뭉치를 512개 토큰 시퀀스로 사전학습 시킨 결과 GPT SLM의 경우 에포크 800번에 손실오차 0.6596을 나타낸다.
- Mempute의 경우 에포크 300번에 손실오차 0.3819에 도달하여 빠른 학습 속도를 나타낸다.
- 2023년 신문 기사 말뭉치를 4K 토큰 시퀀스로 사전학습 시킨 결과 GPT 는 1개 layer 구성으로도 H100 80G 메모리를 초과하여 1개 H100으로 학습이 불가능 하다
- Mempute SLM은 신문 기사 데이터를 4K 토큰 시퀀스, 20개 레이어 구성으로 로 학습 시킬 경우 activation 메모리가 66G 소요되어 23억 파라미터를 H100한대로 사전학습이 가능하다.
- Mempute 모델은 토큰 시퀀스 길이 증가에 따라 메모리가 2차 지수적이 아닌 선형적으로 증가하므로 멀티 gpu 서버 한대로서 LLM으로 구현이 가능하다.
- 38억 매개변수와 4k토큰 시퀀스를 지원하는 파이3미니와 같은 SLM도 사전학습에 H100 100대가 필요한 것에 반해 본 파운데이션 모델은 가속기 단 몇대로 학습이 가능하고 학습 파라미터와 토큰 시퀀스 길이의 증가에 비례하여 그 차이는 지수적으로 커져 현재의 대용량 LLM 사전학습에 따른 비용문제를 근본적으로 해결한다.

이상탐지 적용결과




```
normal-normal : 55943 normal-proud : 921 proud:normal : 16 proud:proud : 82
nan cnt : 0
accuracy: 0.9835504371335276 precision: 0.9997140763755612 recall:
0.9838034608874089 f1 score: 0.9916949058172084
reverse precision: 0.8367346930237402 recall: 0.08175473578447112 f1 score:
0.1489554787600424
```

원본 모델



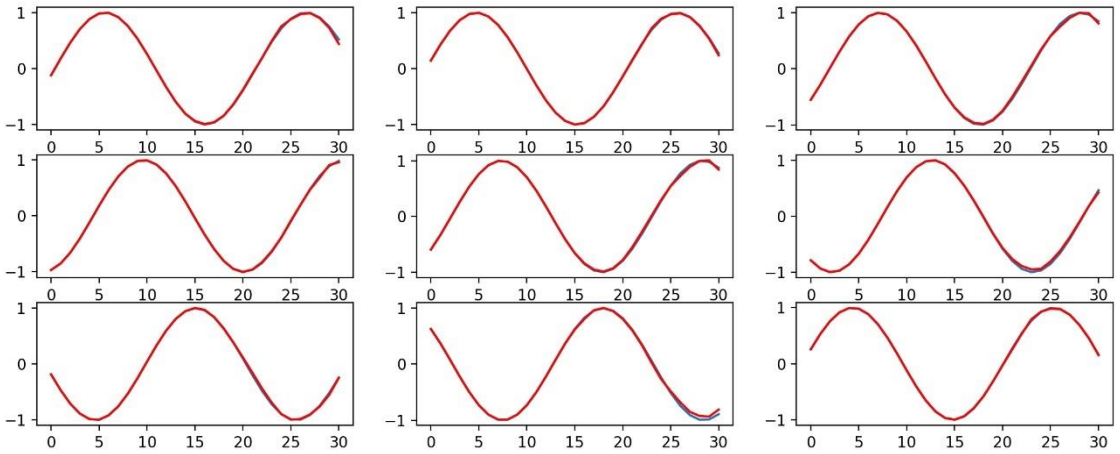
출처

https://github.com/KerasKorea/KEKOxTutorial/blob/master/20_Keras%EC%9D%98%20Autoencoder%EB%A5%BC%20%ED%99%9C%EC%9A%A9%ED%95%B4%20%EC%8B%A0%EC%9A%A9%EC%B9%B4%EB%93%9C%20%EC%9D%B4%EC%83%81%20%EA%B1%B0%EB%9E%98%20%ED%83%90%EC%A7%80%ED%95%98%EA%B8%B0.md

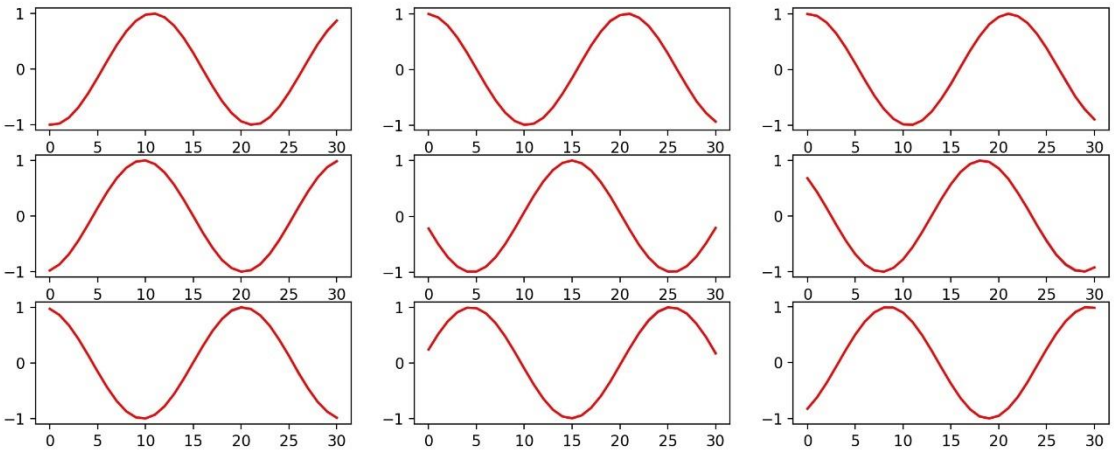
Sign curve predication

- 테스트 데이터의 전체 길이의 반을 입력으로 주고 나머지 반을 auto regression으로 예측
- GPT 와 비교하여 그래프 육안 식별 과 평균제곱 오차 모두 Mempute가 정확
- 이와 같이 규칙성이 명확한 케이스는 정확한 학습 및 추론이 되야 하나 학습 오차를 보면 트랜스포머 계열 모델은 오차가 계속 수렴하지 않고 진동하며 정확도가 떨어지고 테스트 셋 실험 결과 그래프 역시 육안으로도 오차가 확인된다. 이에 반해 Mempute 모델은 오차가 계속 수렴하며 정밀한 추론 결과를 보여준다.

GPT



Mempute

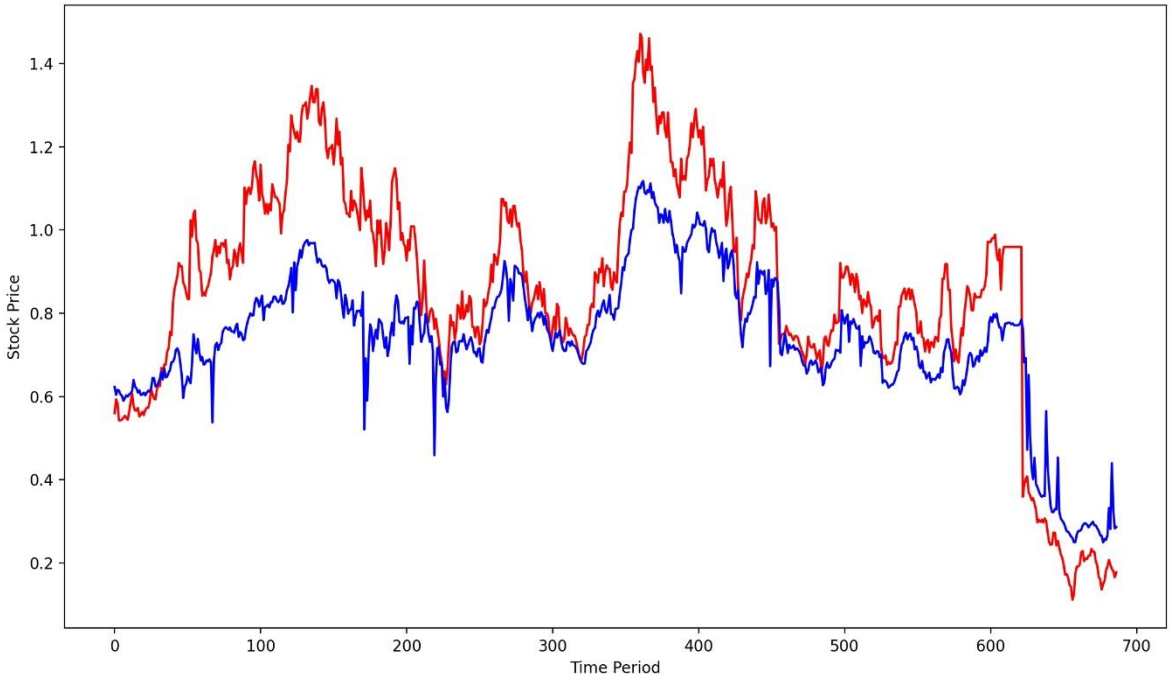

1
0
−1
0 5 10 15 20 25 30


lev: 2 epoch: 48 i: 5120 train mean mloss: 0.000000 loss: 0.017039
epoch: 48 i: 0 train mloss: 0.000000 loss: 0.017235
lev: 2 epoch: 49 i: 5120 train mean mloss: 0.000000 loss: 0.017010
epoch: 49 i: 0 train mloss: 0.000000 loss: 0.016702
lev: 2 epoch: 50 i: 5120 train mean mloss: 0.000000 loss: 0.017054
error: 1.679741
lev: 2 epoch: 50 train mean loss: 0.017054 error: 1.679741

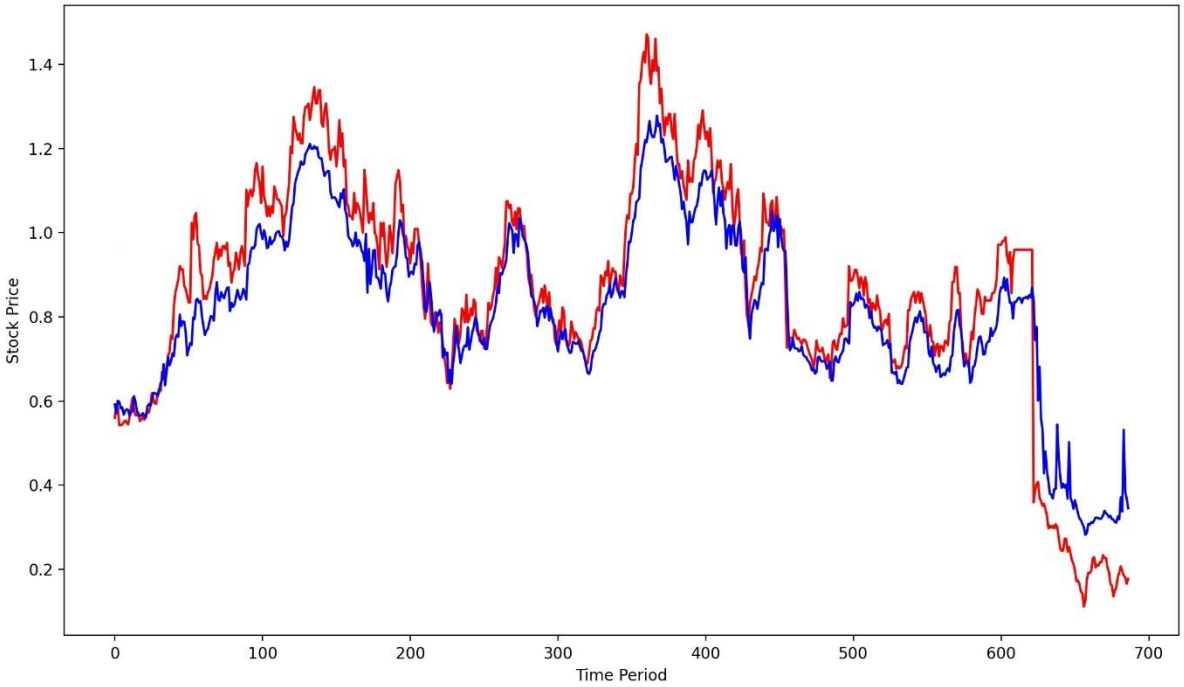
주가 예측

- 과거 추세 데이터로 학습 및 예측하는 케이스에 있어 일반적인 경우와 달리 입력 데이터에 목표 주가에 관한 정보와 미래 값을 포함하지 않고 학습하여 다음 날 종가를 예측한다.
- GPT 의 경우보다 목표치에 더 근접한 추론 결과를 나타낸다.

GPT



Mempute



깃허브: [mempute/v4 (github.com)]

MEMPUTE V3

Neuromorphic Hybrid Machine Learning

- Neuronet -

뉴로모픽 하이브리드 학습 머신은 인간의 사고 과정을 모방한 알고리즘과 심층 신경망을 결합하여 기존 신경망의 부족한 성능을 한층 강화한 학습 및 추론 엔진이다.

NeuroNet을 이상탐지에 적용한 테스트 결과

- **2만건 데이터 셈프링, 비정상 34건 나머지 정상 데이터**
- **변별력을 위해 비 정상데이터를 기준으로 점수 산정**

신경망 오토인코더 테스트 결과

f1 score: 0.10485436893203885 recall: 0.056133056133056136 precision: 0.7941176470588235

normal-normal : 19512 normal-proud : 454 proud:normal : 7 proud:proud : 27

NeuroNet테스트 결과

f1 score: 0.12958963282937366 recall: 0.06993006993006994 precision: 0.8823529411764706

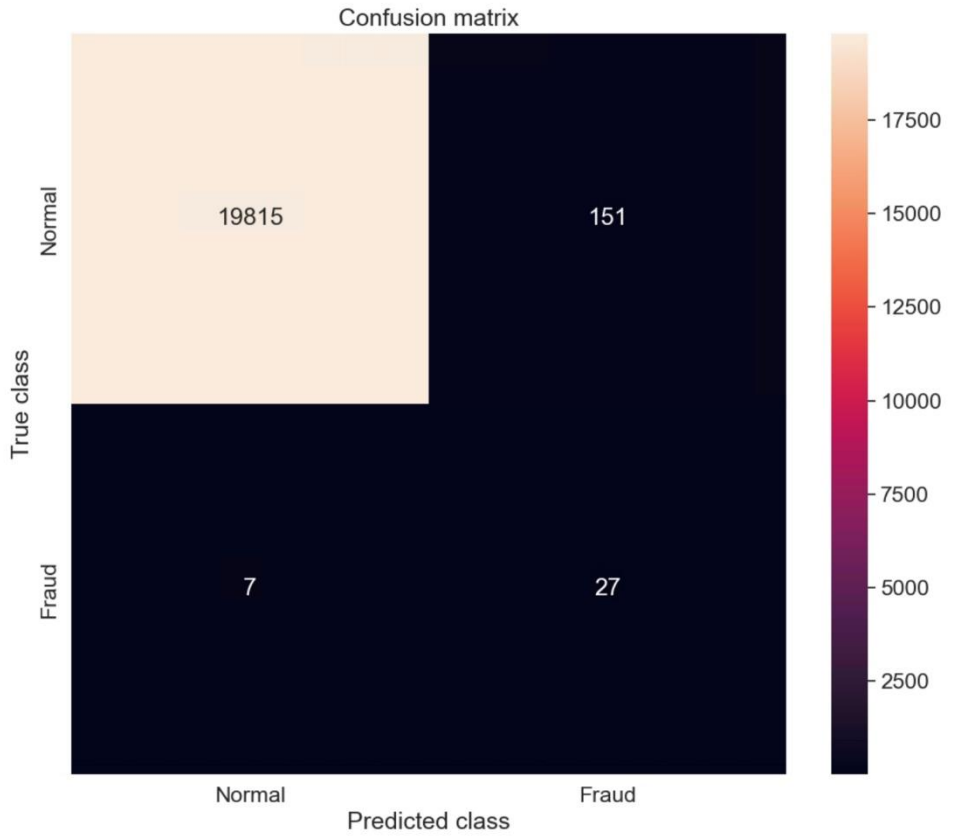
normal-normal : 19567 normal-proud : 399 proud:normal : 4 proud:proud : 30

f1 score: 0.7540983606557378 recall: 0.8518518518518519 precision: 0.6764705882352942

normal-normal : 19962 normal-proud : 4 proud:normal : 11 proud:proud : 23

f1 score: 0.3529411414413293 recall: 0.22689075611185652 precision: 0.7941176447231834

normal-normal : 19874 normal-proud : 92 proud:normal : 7 proud:proud : 27



신경망 오토 인코더 방식과 비교하여 뉴로넷은 하이퍼파라미터 조정에 따라 다양한 결과가 산출되고 모든 케이스에서 기존 방식을 크게 웃도는 예측 성능을 나타내며 향후 챗봇과 같은 다양한 시계열 분야에 적용할 경우 상당한 개선 결과가 기대된다.

Bayesian Pattern Probability Inference Engine

- Pattern Database Green Technology -

Pattern Database는 빈도수에 기반하여 패턴을 발견하고 베이지안 확률 추론을 수행하는 베이지안 패턴 확률 추론 머신이다. 학습과정은 주어진 데이터에 대하여 에포크 한번에 완결된다.
Pattern Database는 메모노드라는 분산 퓨전 관계형 데이터베이스위에서 구성된다. 모든 발견 패턴은 베이터베이스에 저장되고 추론에 사용된다. 메모노드 데이터베이스에 관한 내용은 www.memonode.com을 참고하고 언급된 모든 예제들은 깃허브https://github.com/mempute 에 있으며 **Pattern Database**에 관한 예제는 https://github.com/mempute/anormal 위치에 mempute_pattern_chain_database_example.ipynb 이 있다.

Pattern Database는 데이터에서 빈도수에 의해 규칙을 발견하고 이를 패턴화한다. 패턴들은 서로 경쟁하고 가장 높은 강도의 것이 **invoke**되어 추론에 사용된다.
Pattern Database는 목표값을 타겟팅 하기위해 목표값과의 오차로부터 역전파로 파라미터 오차를 조정하는 연역방식이 아니다. 데이터로 부터 라벨없이 빈도수에 의해 데이터를 가장 잘 설명하는 패턴을 발견하게 되며 이 과정은 귀납적이다. 목표값과의 연결 역시 가장 연결 강도가 높은 연결을 스스로 발견하고 사전 학습하므로써 추론시에 사후 예측한다. 이 모든 과정은 귀납식이다.
Pattern Database는 시간대 혹은 이벤트 등의 범주 단위 정도의 연관성이면 된다. 우리는 많은 경우에 입력에 잘 정합되는 목표값을 알지 못하며 이럴때 **Pattern Database**는 주어진 입력에 가장 가능성 높은 목표를 사후확률로 예측해 낸다. 연역방식인 신경망과 달리 귀납방식으로 추론함으로써 과적합의 위험없이 정확한 추론이 가능하다.
또한 멀티 소스대 타겟을 연결 학습하여 소스중 타겟에 정상성이 가장 높은 입력 요소를 추론하여 발견해 낸다.

이러한 기능성으로 **Pattern Database**는 예측, 분류, 타겟 라벨의 자동생성, 데이터 정상성 판별, 오류 데이터 필터링 등 빅데이터 처리의 모든 광범위한 영역에서 신경망뿐 아니라 이제까지 발명되어진 어떠한 데이터 처리 기술보다 다변적 기능을 강력하고도 범용적으로 수행할 수 있다.
또한 신경망과의 하이브리드 학습을 수행하여 신경망을 더 광범위한 영역에서 부스팅하여 기존에

신경망으로 해결하지 못했던 난제들을 해결할 수 있다.

타겟 라벨 생성은 높은 레벨의 패턴 추상화로 이루어 진다. 신경망 이나 기타 기술은 이러한 기능이 없으며 신경망은 잠재코드의 차원을 작게 줄이면 정보는 붕괴되어 의미없어 진다. 언어 모델에서 발전된 트랜스포머는 압축을 수행하지 않으며 컨볼루션 망은 압축을 위해 최대값만을 선택함으로써 많은 정보가 손실된다. 깃허브에 이상데이터 판별 예제에서 **Pattern Database**로 생성한 타겟 라벨으로 신경망과 하이브리드 학습을 진행한 결과를 보면 높은 수준의 제한된 라벨은 학습되어 판별 효과가 있었으나 더 넓은 범위의 낮은 레벨의 라벨은 목표값으로 학습되지 못하였다. 신경망은 매우 제한된 높은 정합성을 갖은 타겟 범주의 학습만 가능할 뿐이다. 언어의 경우 **one hot**으로 분류되어야 할 어휘 사이즈가 **10000**개도 쉽게 넘을 수 있고 신경망의 언어 모델은 이를 달성하나 언어라는 것은 문법도 있고 의미적으로도 우리는 정형된 패턴으로 사용하기에 가능한 것이다. 개나 고양이 분류에서 보듯 이러한 분류들도 또한 정상성이 매우 높은 데이터이다.

다만 이문서의 두번째 파트인 **mempute** 신경망 버전의 강력한 시계열망인 제너릭 망으로는 생성된 라벨로 하이브리드 학습을 진행할수록 오차가 줄어들어 학습이 됨을 확인할 수 있다. 그러나 규칙이 희박한 자연데이터로부터 넓은 범위의 이산 라벨을 수렴시키는 것은 신경망에게는 매우 버거운 일로써 학습시간이 오래 걸린다. 사전학습은 인코더를 생성하는 과정인데 **Pattern Database**의 추상화된 라벨은 단일 혹은 몇 개 차원 정도로 입력에 대한 분류 태그를 제공하여 이를 타겟으로 인코더 학습시킬 경우 더 높은 사전학습 결과를 얻을 수 있을 것이다.

Pattern Database는 비지도 학습, 타겟 연결 학습, 사전학습, 전이학습, 증강학습 및 보상에 따른 패턴 강도를 조정으로 강화학습등 다양한 학습을 수행 할 수 있다.

Pattern Database는 학습의 결과로 발견된 패턴을 데이버테이스에 저장하고 계속적인 추가학습으로 인해 패턴이 데이터베이스에 누적됨에 따라 점점 더 정확하고 오류에 강한 추론이 가능하다. 이는 통계표본이 클수록 모집단과 유사해져 예측 결과가 정확해 지는 것과 같은 이유이다

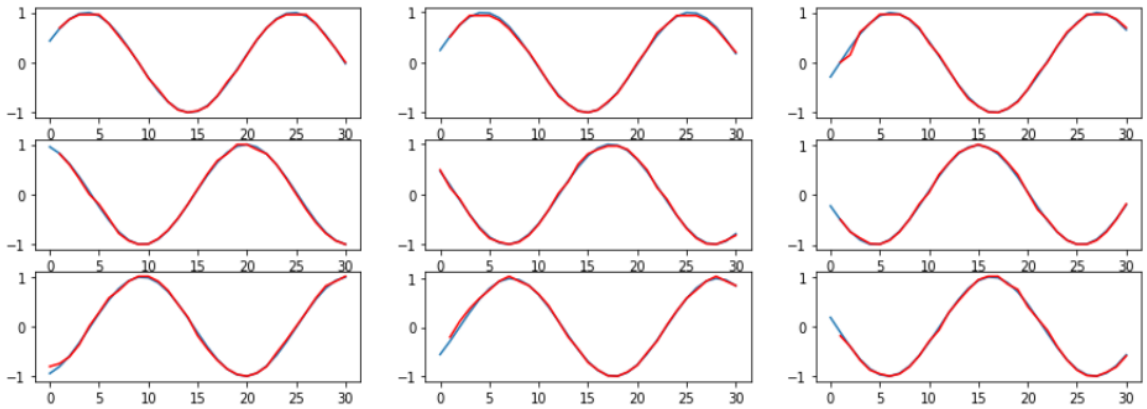
Pattern Database는 고성능의 대용량 분산 관계형 데이터베이스위에서 구동되어 대량의 패턴 학습 및 추론이 가능하며 분산 분할 학습과 패턴 병합을 하여 대량의 패턴 처리를 빠른 시간에 수행할 수 있다.

Pattern Database는 신경망의 강력한 전처리기로서도 사용될 수 있다. 입력데이터에서 타겟을 가장 잘 설명하는 요소를 정확하게 분리하내는 과정은 사람의 판단이나 개입이 필요없어 데이터 학습의 모든 과정을 자동화시킬 수 있다. 이와 같은 기능성으로 **Pattern Database**는 다양한 분야에서 범용적으로 사용될 수 있을뿐만 아니라 이상 탐지나 개인화 추천 시스템등 일반적으로 기존 기계학습 방법으로 효과가 적거나 결과가 애매한 분야에 탁월한 성능을 기대 할 수 있고 추론 결

과에 대한 확률 해석 설명을 제공할 수 있다.

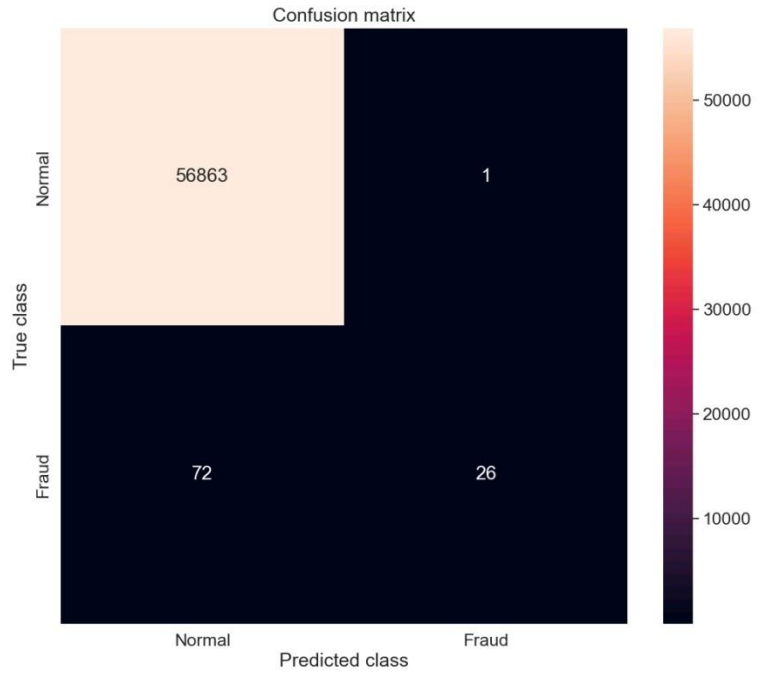
간단한 Pattern Database적용 예

사인 곡선 예측

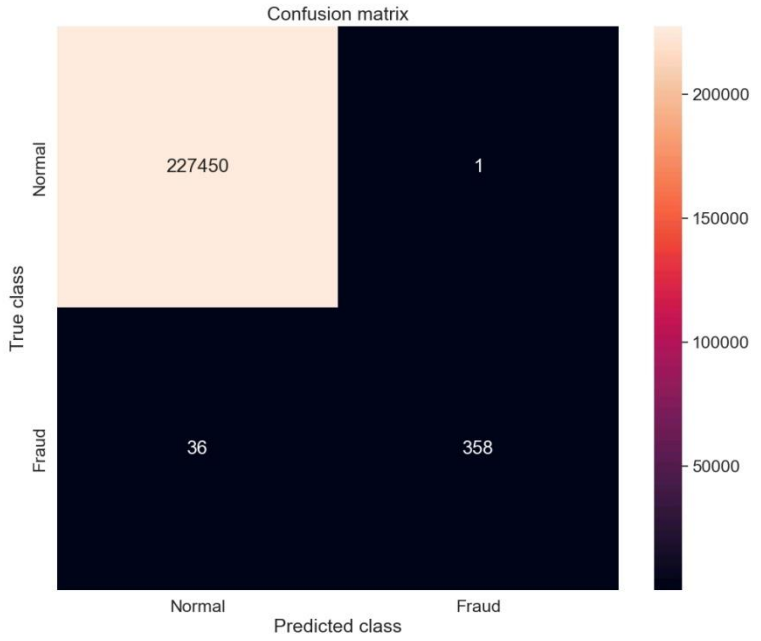


아마존 주가 예측

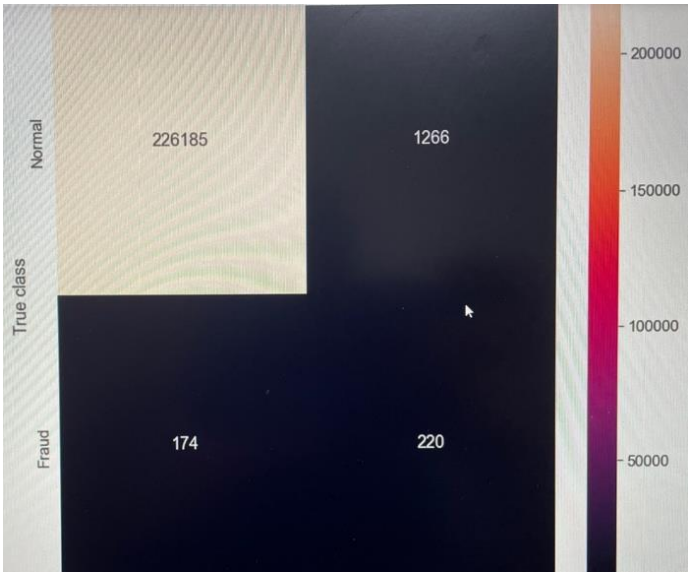
이상 탐지 데이터 예측



이상 탐지 학습 데이터 예측



신경망 학습 데이터 예측



Anormal detection(이상 탐지)는 비정상 데이터가 적어 신경망으로는 지도 학습을 하지 못하고 오토인코더로 학습하는데 테스트 데이터 예측도 정확도가 낮을뿐 아니라 학습데이터를 예측한 결과도 마찬가지이다. 신경망의 목적이 일반화에 있어 당연한 결과인데 이상 탐지의 경우 정상과 비정상의 데이터 비율이 비대칭이어서 학습 데이터가 축적되어도 최총 정확도가 향상될 수 없다.
학습 데이터를 예측한다는 것은 학습데이터에서 가능도 높은 패턴을 발견해 내는 일반화 과정뿐만 아니라 이러한 고 가능도 패턴을 데이터베이스에 직접적으로 계속 누적함으로써 경험치를 높혀 정확도가 지속적으로 향상되는 것을 의미한다.
위 이상탐지의 예를 살펴보면 차원 축소 압축이 2만분의 1로 된 것으로 압축은 일반화의 한 과정이고 학습데이터 예측이 이 정도 높은 일반화로도 높은 정확도로 나온 결과이고 테스트 데이터 예측도 같은 압축으로 나온 결과로서 예측되지 못한 비정상 케이스는 학습 데이터에는 없는 패턴이다. 또한 패턴 데이터베이스와 신경망을 하이브리드 형태로 증강 학습법을 수행한다면 state of the art를 달성 할 수 있을 것이다.
다음은 이상 탐지 예제의 패턴 추론과정의 확률 계산을 출력한 것으로서 이상인데 정상으로 판별하여 추론이 틀린 경우를 설명한다. prior_v는 목표값이고 0은 정상, 1은 이상 임을 나타내고 이상으로 예측해야하나 posterior(사후확률)을 보면 정상 출력이 -10.810163, 이상 출력이 -18.120785 으로서 정상 목표값의 확률이 더 높아 정상으로 판별한 것을 나타낸다. 여기서 pid는 목표값의 아이디, prior st는 사전확률, likely는 event(사건)하에서 발생되는 prior확률인 가능도이다.

event-prior probable result: pid(4249817) prior_v(0.000000) posterior(-10.810163) likely(-10.809829) prior(-0.000334) prior st(2993) prior ast(2994)

event-prior probable result: pid(4249833) prior_v(1.000000) posterior(-18.120785) likely(-10.116419) prior(-8.004366) prior st(1) prior ast(2994)

NEURONET

- MEMPUTE NEUROMORPHIC HYBRID MACHINE LEARNING -

- 메뉴얼 -

뉴로넷의 API 는 파이썬과 C 가 1:1 매칭되며 본 문서는 파이썬을 기준으로 한다.

파이썬 클라이언트 API

neuronet

- 주어진 이름으로 신경망을 생성한다.

neurogate

- Pattern network database 에 패턴망을 생성한다.

close_net

- 신경망 및 패턴망 패쇄.

loadnet

- 뉴로넷 로드.

update_param

- 파라미터를 변경한다.

xtrain

- 뉴로넷 학습.

connect_train

- 패턴 데이터베이스에서 타겟 패턴 연결 학습, 타겟 패턴 연결이 필요없으면 생략.

xpredict

- 뉴로넷 추론.

neuronet(param, gid, name)

- param: 파라미터 딕셔너리
- gid: GPU index
- name: 뉴로넷 이름
- return: 뉴로넷 핸들

neurogate(net, stmt, name, learning_rate, x, y, init)

- net: neuronet()으로 생성된 뉴로넷 핸들
- stmt: pattern net database 핸들

- name: 패턴 망 이름
- learning_rate: 패턴 생성 학습률
- x: 입력값
- y: 목표값
- init: 패턴망 생성이면 1, 로드이면 0

xtrain(net, x, xother, tunning, epoch, relay = 0, decord = 0)
- net: neuronet()으로 생성된 뉴로넷 핸들
- x: 오토인코딩 학습할 학습 데이터
- xother: 디코더 학습할 데이터, x 와 동일하면 None
- tunning: 레벨 학습이면 0, 전체 튜닝 학습이면 1
- epoch: 에포크
- relay: 이전 학습 레벨을 계속하여 학습할려면 1, 아니면 0
- decord: 1 이면 디코더 학습, 0 이면 인코더 학습
- return: next level sequence

connect_train(net, x, y)
- net: neuronet()으로 생성된 뉴로넷 핸들
- x: xtrain() 리턴된 다음 레벨 시퀀스
- y: 패턴망에서 연결 학습될 타겟 데이터

xpredict(net, x, decord = 0)
- net: neuronet()으로 생성된 뉴로넷 핸들
- x: 디코딩 추론할 데이터
- decord: 1 이면 디코더 예측, 0 이면 패턴 네트워크 예측
- return: 추론 결과

update_param (net, param)
- net: neuronet()으로 생성된 뉴로넷 핸들
- param: 변경할 파라미터 딕셔너리

ex) 뉴로넷 생성

```
    import mempute as mp
    from morphic import *

    train_params = dict(
                input_size=inp_size,
                hidden_sz=16,
                learn_rate=1e-4,
                drop_rate=0.1,
                signid_mse = mse,
                wgt_save = 0,
                layer_norm = False,
                n_epair = 3, #encode depth 개수, 1 부터 시작
                residual = 1,
                on_schedule = False,
                dtype = mp.tfloat,
                levelhold = 0.7,
                tunehold = 0.98,
                seed = 0,
                decay = 0.0001,
                decode_active = 1,
                regression = 0,
                dec_lev_learn = 1,
                decode_infeat = dec_infeat_other,
                size_embed = dec_infeat_other + 1 if embedding else 0, #+1 은
    패딩 추가
                batch_size=batch_size)

train_params = default_param(**train_params)
param = param2array(**train_params)
 net = mp.neuronet(param, gid, model_name)
 mp.neurogate(net, stmt, model_name, sensor, x_train, y_train, 1)
 mp.close_net(net)
```

ex) 뉴로넷 로드

```
net, param = mp.loadnet(gid, model_name)
mp.neurogate(net, stmt, model_name, sensor, x_train, y_train, 0)
```

ex) 뉴로넷 학습

```
reent = np.expand_dims(x_train, -1)
for i in range(level):
    reent = mp.xtrain(net, reent, xtrain_other, tuning, epoch, relay)

mp.connect_train(net, reent, y_train)#나머지 상위 및 연결 학습
```

MEMPUTE V2

mempute deep learning framework & time series neural network

딥러닝 프레임워크 및 시계열 신경망

mempute는 기계 학습 신경망 프레임워크로서 c++와 파이썬을 지원하고 일반 선형대수 라이브러리에 대하여 그래프를 생성하여 자동미분 역전파 기능을 수행한다. 또한 데이터 수평/수직 분할 알고리즘을 내부적으로 수행하여 병렬 학습 수행 및 분산 수행을 지원한다. 또한 API가 c++와 파이썬이 1 : 1 매칭되어 c++ 환경에서도 쉬운 모델링이 가능하다.
Mempute는 프레임 워크에 시계열 패턴 과 이미지 인식을 수행 할 수 있는 **Dynagen** 과 **Generic** 신경망이 포함되어있다. **Generic**은 시계열 신경망이고 **Dynagen**은 **Generic**망을 이미지 인식으로 확장시킨다.

Impulse는 **dynagen**망을 사용하여 학습 및 추론할때 함께 사용되는데 **dynagen** 망은 학습이 단계적, 계층적으로 수행되므로 입력과 타겟 데이터의 동기화 및 **flow control**을 수행하고 동기식, 비동기식 데이터 입력, 학습, 추론 인터페이스를 지원한다.

dynagen은 일반화된 제너릭망으로 런타임에 동적으로 연결하여 대부분의 목적하는 신경망을 별도의 모델링 없이 바로 수행시킨다. 계층적으로 여러개의 **isle loop**로 구성되며 루프는 하위 망으로서 제너릭망을 로컬 또는 원격에 **invoke**하여 독립적으로 수행시키고 인 아웃을 파이프 시스템으로서 연결하여 여러개의 제너릭 망을 분산 병렬 수행시킨다.

루프의 종류로는 **encoder, decoder, couple isle loop**가 있고 인코더는 입력을 차원 축소하여 압축하고 디코더는 압축된 잠재코드인 인코더의 출력을 입력으로 하고 인코더의 입력을 타겟으로 디코딩 학습 수행한다. 커플 루프는 인코더 루프의 출력을 입력과 타겟으로 하여 연결 학습을 수행한다.

다이나젠 망은 오퍼레이션으로서 **Classification, Translation, Recall, Link**를 지원하고 **long term sequence** 시계열 데이터를 루프를 적층하여 auto encoding 방식으로 압축 추상화를 수행한다.
다이나젠의 **Classification**은 분류 문제를 학습하고 **Translation**는 **seq-to-seq** 연결학습하며 **Recall**은

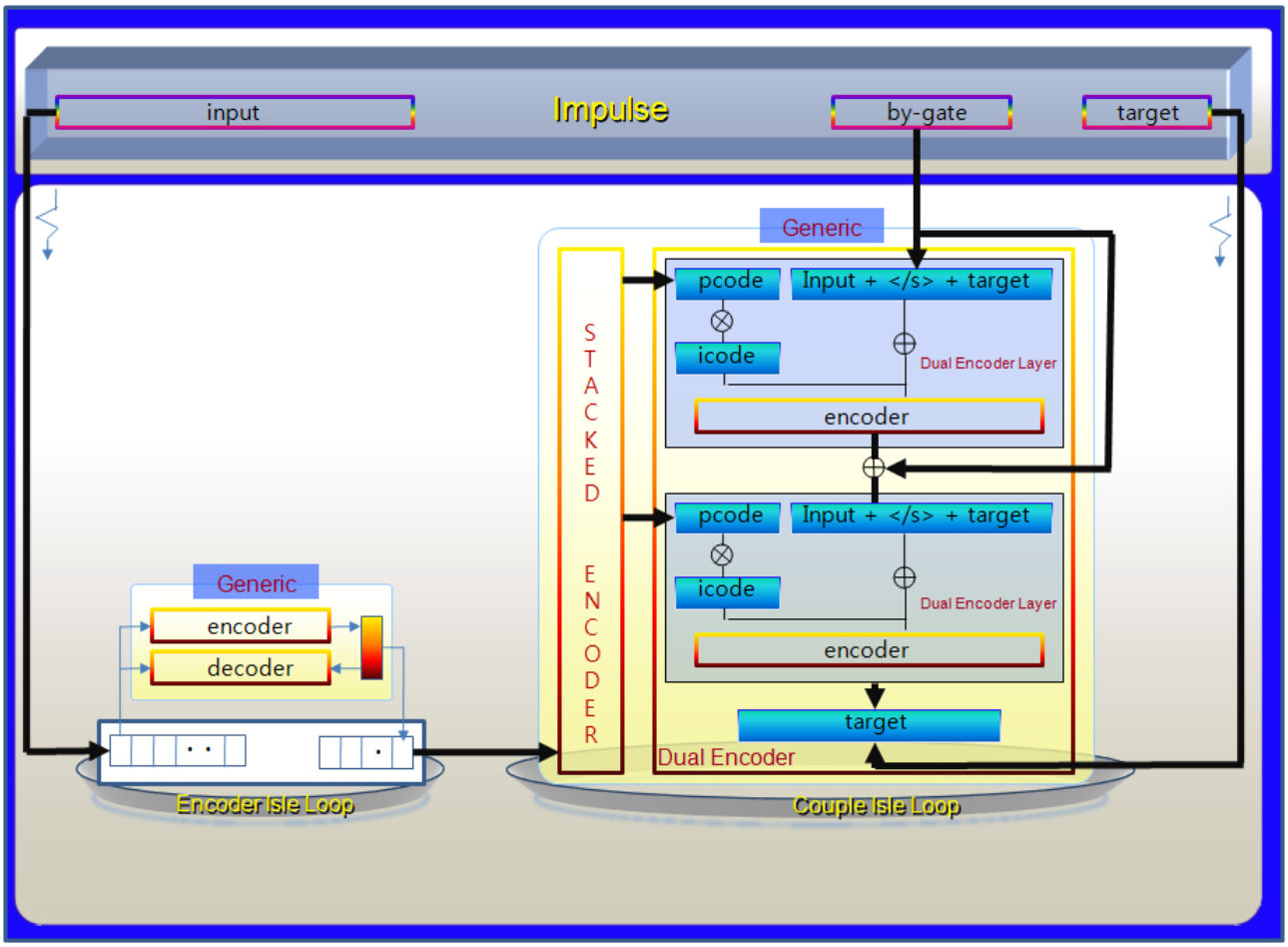
원본 데이터의 복원이나 합성에 관한 것이다. Link 오퍼레이션은 다이나젠 루프의 출력을 연결하여 Projection (투사)을 수행하고 데이터 동기화를 수행하는 impulse와 함께 단위 신경망인 제너릭을 연결하여 거대 신경망 시스템의 구성을 지원한다.

제너릭망은 신경망의 하위 오퍼레이션을 오토인코더를 구성하여 일반화, 추상화 시키고 패턴을 탐지하고 생성하는 인코딩 기능은 매우 강력하여 세부적 사항의 encapsulation을 지지한다. Impulse나 Dynagen과 상관없이 독립적으로 수행할 수 있으며 자체적으로 적층 인코더와 디코더를 가지고 있고 분류, seq-to-seq연결 학습, 복원, 듀얼 인코더로 auto regression기능을 수행한다.

LTC(Long Term Comprehension)

LTC의 학습 구성 요소는 이전 코드 (pcode), 입력, 타겟 (목표값)이고 pcode는 단위 문맥 정보로서 long term sequence를 제너릭 망 또는 다이나 젠 망에서 차원 축소 압축하여 생성되고 양방향 참조 학습되어 독해력 테스트과 같은 down stream task를 수행 할 수 있게 한다.
입력은 down stream task에서 query로서 수행되고 순방향 참조 학습되며 이때 pcode는 key로서 수행된다. pcode가 없다면 입력에 문맥 정보를 포함 시킬 수 있다.

pre-training은 이전 문장을 입력으로 다음 문장을 예측하는 것으로 수행되어 별도의 라벨 데이터가 필요 없으며 추가적인 전처리가 거의 필요 없어 학습 데이터 확보에 어려움이 없으며 적은 노력으로 대량 학습 시킬 수 있다.
pre-training 과정에서 입력은 이전코드로 이동하여 이전코드 시퀀스 사이즈만큼 누적하여 적재되고 다음 입력과 함께 학습된다. 이와 같은 처리로 LTC는 일관된 문맥 정보를 유지하여 모든 down stream task에서 보다 높은 차원의 자연어 이해 수준을 제공한다.



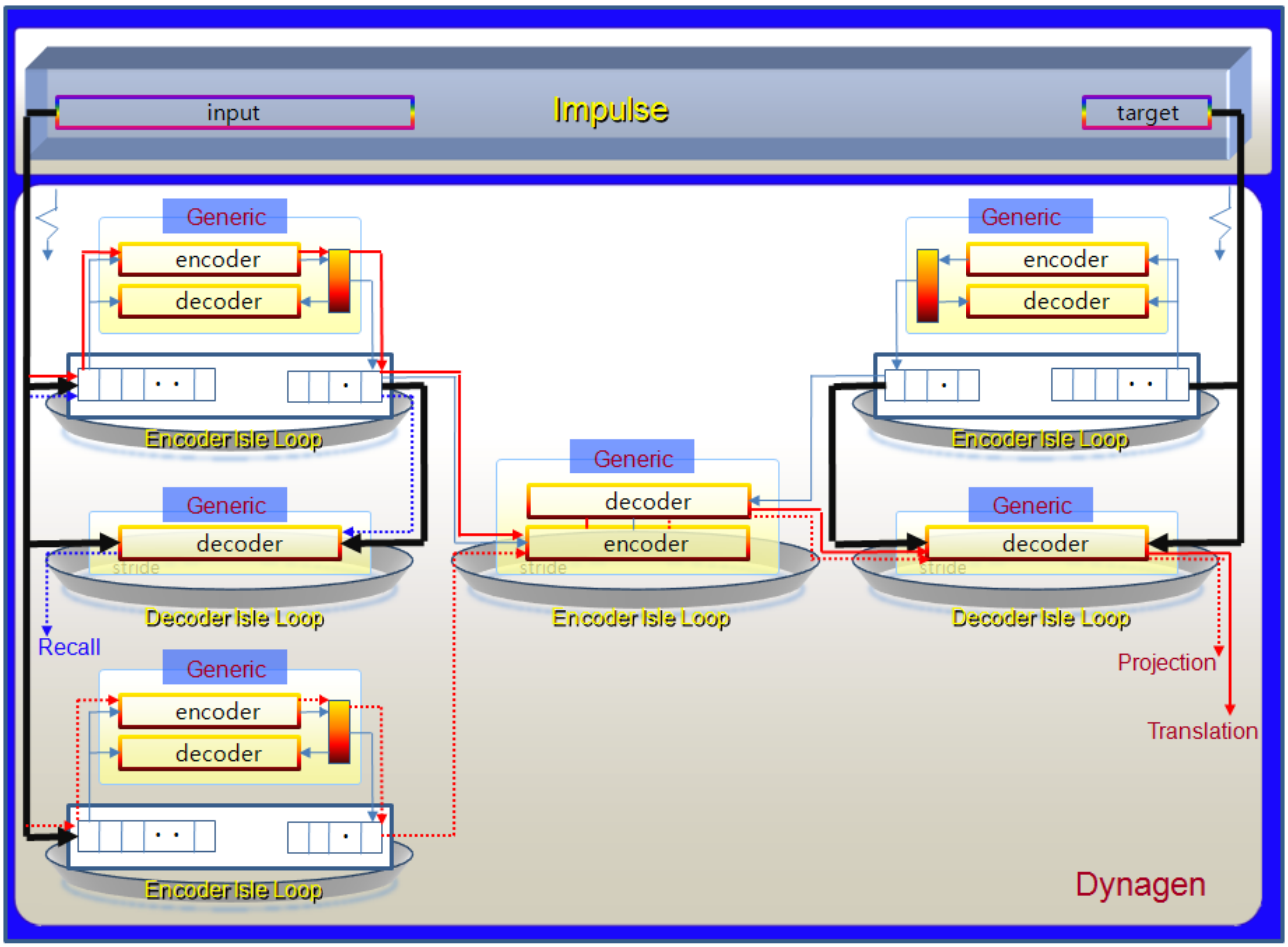
Dynagen Classification Operation & Generic Auto Regression

위 그림은 LTC에서 Dynagen Classification Operation 과 Generic Dual Encoder를 사용하여 seq-to-seq prediction을 수행하는 것을 나타낸다.

Dynagen Classification Operation은 타겟 측이 코드 압축이 없는 asymmetric 학습으로서 주로 분류에 사용되는 것으로 이에 Generic Dual Encoder를 Dynagen couple isle loop에 적재하여 연결한다.

다이나젠의 인코더 루프는 적층하여 구성될 수 있고 제너릭의 인코더와 디코더를 사용하여 최 하위에서 특징 패턴을 오토 인코딩 방식으로 추출한다.

추출된 특징 패턴 스트림은 커플 루프의 제너릭 적층 인코더에서 좀더 고수준의 추상화를 수행할 수 있다. 이 과정은 생략될 수 도 있으며 이렇게 압축 추상화된 코드는 제너릭 듀얼 인코더의 pcode에 적재되어 by-gate에 입력과 타겟 쌍이 적재된 것 과 함께 다시 한번 인코딩 되어 auto regression방식으로 타겟 스트림을 예측한다. 또한 듀얼 인코더 레이어 단위로 residual하여 많이 반복하면 할수록 더 강력한 예측 성능을 나타낸다.



위 그림은 다이나젠의 나머지 오퍼레이션을 나타낸다. 오퍼레이션의 세부 수행은 모두 제너릭의 인코더와 디코더를 사용하여 수행되는데 이렇게 일반화를 할 수 있는 것은 제너릭 인코더의 강력한 패턴 추출 및 생성 기능 때문에 가능하다.

Translation operation은 입력과 타겟 양측에서 **symmetric** 하게 각각 동시에 계층단위로 패턴 압축 수행되고 최종 압축 패턴이 커플 **isle**에서 연결 학습 하고 추론 과정에서는 빨간색 실선으로 표시된 경로들 따라 추론된다.

Projection은 **learning pair**와는 상관없는 다른 **isle loop**간에 **Link operation**에 의하여 연결되어 미리 학습되지 않은 것에도 예측을 수행하여 다양한 신경망 구성을 가능하게 한다.

특징

Generic 망은 시계열 데이터의 학습 및 추론을 수행한다. 일반적인 시계열망인 **RNN**과 이의 기울기 소실 문제를 개선한 **LSTM**은 여전히 **long sequence**인 경우 계속 곱이 행해지면서 뒤로 가면서

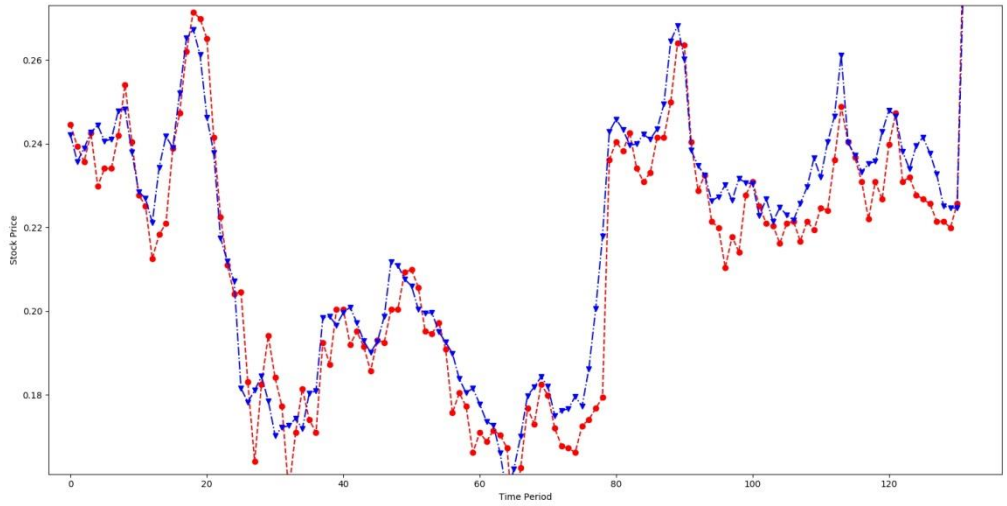
앞쪽의 정보가 소실되는 문제을 안고 있다. 이를 개선하는 방법으로 요즘 시계열 처리의 대세인 Transfomer계열이 있는데 셀프 어텐션 기법으로 인하여 기울기 소실 문제를 극복하고 어느 위치에서나 어텐션 할 수 있음을 장점으로 내세우지만 계산량이 시퀀스 길이에 비례하여 exponential하게 증가하여 시퀀스 길이에 심각한 제약이 있다. xlnet같은 경우는 이전 문맥 코드를 두기는 하나 매우 shallow하다. 따라서 여전히 long sequence 처리 문제에서 자유롭지 못하다. 또한 여러 실험을 해본 결과 패턴 생성이 RNN계열 만큼 강력하지 못해 정답 시퀀스를 정확히 맞추지 못하는 한계가 있다.

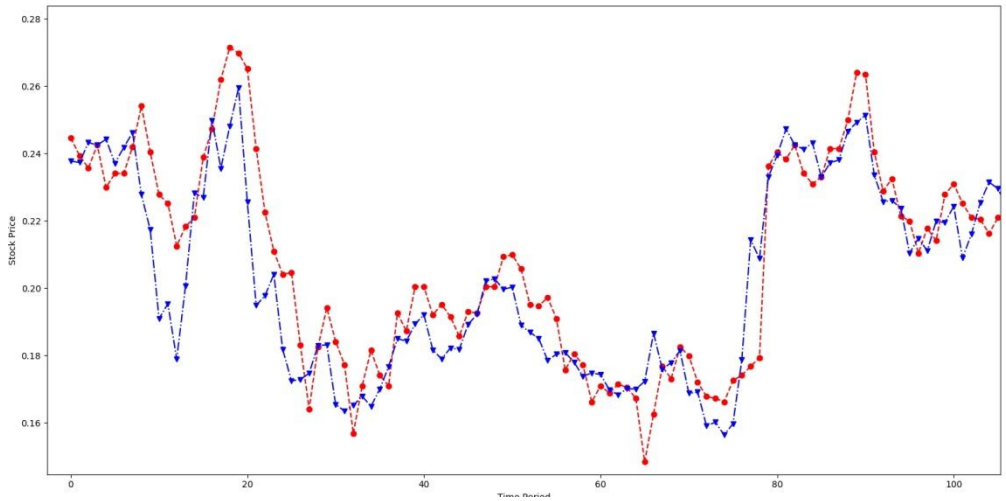
부가하여 이미지 인식 분야 성능을 나타내는 컨볼루션 망이 있으나 특징을 추출하기 위하여 컨볼루션 연산을 수행하는 과정에서 연산 자체와 풀링 레이어에 의하여 정보의 파괴가 심하고 생성 모델을 수행 할 수 없는 것이 단점이다.
이러한 점을 극복하고자 트랜스포머 모델을 이미지 인식에 적용하는 시도가 있으나 결과는 지켜봐야 할 일이다.

이러한 점들에 비춰봤을 때 제너릭 망은 시퀀스 길이에 제약을 받지 않고 강력하게 패턴을 탐지하고 생성한다는 것은 커다란 장점이며 이러한 결과로 보다 더 정확한 시계열 예측을 지원한다. 또한 다이나젠 망과 함께 사용하여 이미지 인식 분야에도 적용하여 정보의 손실 없이 특징을 파악하고 이미지 구성 요소의 관계를 이해하는 인식 결과에 따라서 다양한 어플리케이션과 언어 모델에서와 마찬가지로 Generative Model을 적용 하는 것이 기대된다.
Impulse 와 Dynagen망은 Generic으로 구성된 단위 신경망을 손쉽게 동적으로 연결 확장시켜 대단위 신경망 시스템 구성을 지원하고 후에 Auto ML의 상위 개념으로서 스스로 학습하고 확장되는 베이스를 지지한다.

주가 예측

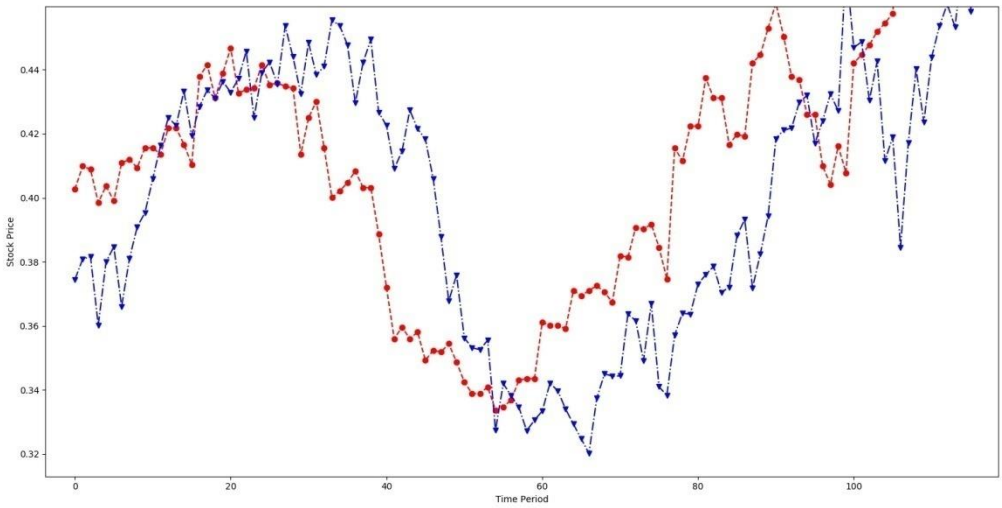
다음은 강력한 패턴 탐지의 근거로서 주가 예측 그래프를 보여준다.
학습은 Dynagen과 Generic망을 함께 사용하여 입력 값을 pcode로 하였다.





위 그림에서 파란색 선은 예측을 빨간색 선은 실제 주가를 나타낸다.

두 개 모두 학습된 데이터를 예측한 것으로서 LSTM이나 기타 시계열 망은 학습한 데이터 조차 위와 같이 예측이 선행되는 모습을 보이지 못하고 예측한 날짜만큼 Lagging된다. 더욱 유의할 점은 위 그림은 목표 주가로 학습하지 않았다는 것이다. 타 4개 내지는 8개 종목의 시가, 종가, 거래량 등을 각 종목별로 입력으로 하고 목표 종목의 6일 앞의 주가를 타겟값으로 하여 예측 한 것이다. 타 시계열 망으로는 6일 앞 예측이면 6일치가 lagging되어 전혀 예측 성능을 보이지 못하고 더군다나 여러 개의 타 종목 주가를 입력으로 한다면 그 상관 패턴 분석은 힘들어 진다.



위 그림은 학습되지 않은 테스트 데이터를 동일한 방법으로 타 종목 주가를 입력으로 하여 목표 종목 주가를 6일치 앞을 예측한 것으로서 초기 대략 40일 정도는 그 이전 까지의 데이터를 학습한 결과로 실제 주가의 패턴을 거의 예측해낸 것을 보여준다. 그 이후는 학습 데이터와 점점 멀어지므로 예측의 정확도는 줄어들 수 밖에 없다.
여기에 주가 관련 각종 지수나 뉴스들을 본 신경망으로 텍스트 마이닝하여 입력 한 다면 더욱 더 정확한 예측 성능을 나타낼 것이 기대된다.

챗봇

다음은 대략 8200개 정도의 대화문을 학습하고 3500여개 대화문을 테스트한 결과의 일부 이고 [Source]가 입력 값, [Truth]가 타겟 값, [Translated]가 예측 값을 나타낸다.

========== train batch ==========

[Source] 너무 많이 먹어서 소화시켜야 하는데 움직이기가 싫어 </s>

[Truth] 소화제 챙겨드세요 </s>

[Translated] 소화제 챙겨드세요 </s>

[Source] 전세비 올려달래 어떡하지 </s>

[Truth] 사정을 잘 설명해보세요 </s>

[Translated] 사정을 잘 설명해보세요 </s>

[Source] 독박 육아 짜증나 </s>

[Truth] 배우자와 대화를 나눠보세요 </s>

[Translated] 배우자와 대화를 나눠보세요 </s>

[Source] 오늘 라면 먹고 싶어 </s>

[Truth] 맛나게 드세요 </s>

[Translated] 맛나게 드세요 </s>

[Source] 약 한달 전 헤어진 그에게 </s>

[Truth] 연락할 생각 하지마세요 </s>

[Translated] 연락할 생각 하지마세요 </s>

epoch: 0 step: 17, train(A): 0.972027, test(B): 0.023392, B-A: -0.948635

ep #: 0 step #: 17

========== test batch ==========

[Source] 연애상담해줬더니 나만 바보됐어 </s>

[Truth] 다음부터는 해주지 마세요 </s>

[Translated] 몸에 어떤 성분이 부족한지 알아보세요 </s>

[Source] 이 사람 없으면 못 살 거 같아 </s>

[Truth] 시간이 지나면 잘살고 있는 당신을 보게 될 거예요 </s>

[Translated] 거리를 두세요 </s>

[Source] 썸 타는 사이인데 카톡하다가 마무리 어떻게 해 </s>

[Truth] 잘자요 내일도 보고싶어요 라고 하는 건 어떨까요 </s>

[Translated] 달라지는 게 없다면 만나지 않는 게 더 나을 수도 있어요 </s>

[Source] 좋아하는 사람한테 적극적인 여자 별로야 </s>

[Truth] 적극적이면 오히려 더 좋을 것 같아요 </s>

[Translated] 저한테 하나씩 싶네요 </s>

[Source] 사랑하는건지 나도 모르겠어 </s>

[Truth] 없어도 살 수 있는지 생각해보세요 </s>

[Translated] 확신이 없나봐요 </s>

학습 데이터 셋을 추론한 결과는 정확도가 대략 80%~100%를 나타내고 테스트 셋은
이 정도 소량의 학습 데이터로는 정확도를 논할 수 없으나 아래 테스트 셋 결과의 마지막을 보면 입력문 "사랑하는건지 나도 모르겠어**" 에 대한 정답이 "** 없어도 살 수 있는지 생각해보세요 **" 인데 예측 값 "**확신이 없나봐요 **" 를 보면 문맥에 맞는 응답을 한 것을 볼 수 있다. 이때의 입력값은 학습 데이터에는 없었던 것이다. 이는 일반화가 달성되어 적은 데이터로 학습한 결과를 가지고도 테스트 데이터 셋 에서도 패턴이 추론될 수 있는 한 최대로 추론됨을 의미한다 하겠다.**
두 가지 예제 모두 인코더로만 구성 된 것으로서 과적합의 여지가 없으며 적은 학습 데이터로도 정확도와 일반화가 모두 만족된 결과를 나타낸다.

트랜스포머 류의 시계열 모델을 학습시켜 본 결과 하이퍼 파라미터를 다르게 하여 여러번 해봐도 이 정도의 데이터로는 학습 셋 조차 거의 대부분 정답 시퀀스를 맞춰 낼 수 없어 정확도를 계산하는 것이 무의미 하였다. 데이터가 많아도 정확도가 올라갈 것이라고 기대되는 학습의 징후는 보여지지 않았다.

Mempute framework 및 위 예제 코드는 깃허브 https://github.com/mempute

- Deep Learning Framework Library -

Mempute Class Interface

이름	Tracer	
설명	세션 관리, 데이터 라이프 사이클 관리, thread safe	
함수	Run	학습 실행
	saveWeight	가중치 저장
	loadWeigth	가중치 로드
	namescope	네임 스페이스 정의
	directx	플럭스 포워드 기능만 수행 설정/리셋
	trainvar	Train variable list리턴

,

이름	Flux	
설명	텐서	
‘함수	dot	행렬 곱
	mul	
	plus	
	minus	
	div	
	matmul	배치 행렬 곱
	split	
	unstack	
	reshape	
	expand_dims	
	squeeze	
	transpose	
	softmax	
	squaredDifference	
	softmaxCrossEntropy	오차함수
	sum	
	mean	
	meanSqureError	오차함수
	tanh	활성함수

	relu	활성함수
	sigmoid	활성함수
	prelu	활성함수
	sqrt	
	log	
	embedding_lookup	
	one_hot	
	slice	
	argmax	
	equal	
	feedf	학습 데이터 입력
	feedt	학습 데이터 타입변환 입력
	copyf	데이터 복사
	copyt	데이터 타입변환 복사
	dstrw	배열 형태 데이터 write
	shape	
	begin_p	읽기모드 데이터 시작 포인트
	begin_wp	쓰기모드 데이터 시작 포인트
	end_p	데이터 종료 포인트
	at_d	플럭스 원소값 리턴
	printo	플럭스 내용 출력
	printg	플럭스 기울기 출력
	arange	순차값 생성
	fill	임의값 일괄 설정
	expofill	지수 순열 값 생성
	expand_elen	지정 차원 현행 값 확장
	randn	정규 분포 생성
	not_equal	
	layer_dense	
	layer_normal	레이어 정규화

이름	Initializer	
설명	가중치 값 초기화 함수	
‘함수	xavier	
	he	

	one	
	zero	

이름	AdamOptmizier	
설명	옵티마이저	
‘함수	minimize	

이름	GradientDescentOptimizer	
설명	옵티마이저	
‘함수	minimize	

이름	Coaxial	
설명	시계열 신경망	
‘함수	train	학습
	predict	평가

이름	Stratus	
설명	시계열 신경망	
‘함수	train	학습
	predict	평가

Mempute Static Function Interface

이름	BoostMemput	
설명	프레임워크 run	
헤더파일	memput.h	
	형	설명
입력매개변수		
출력매개변수		
반환값		

이름	trace	
설명	Tracer 객체 생성	
헤더파일	memput.h	
	형	설명
입력매개변수	①sytet ②bytet *	① 1: 디버깅 즉시 실행 모도 ② 네임스페이스
출력매개변수		
반환값	Tracer *	

이름	flux	
설명	Flux 객체 생성	
헤더파일	memput.h	
	형	설명
입력매개변수	①Tracer * ② initializer_list<intt> ③ubytet ④ubytet ⑤vinitfp ⑥ bytet *	① Tracer ② shape info ③ 데이터 타입 ④ Flux type ⑤ Initializer ⑥ 네밍스페이스
출력매개변수		
반환값	Flux *	플럭스 객체

이름	concat	
설명	플럭스 병합	
헤더파일	memput.h	
	형	설명
입력매개변수	① vector<Flux *> or initializer_list<Flux *> ② intt	① 병합할 플럭스 리스트 ② 병합 축
출력매개변수		
반환값	Flux *	병합 플럭스

이름	stack	
설명	플럭스 적층	
헤더파일	memput.h	
	형	설명
입력매개변수	① vector<Flux *> or initializer_list<Flux *> ② intt	① 적층할 플럭스 리스트 ② 적층 축
출력매개변수		
반환값	Flux *	적층 플럭스

이름	generic	
설명	범용 신경망 생성	
헤더파일	memput.h	
	형	설명
입력매개변수	**① Flux*** **② Flux*** **③ intt** **④ intt** **⑤ intt** **⑥ intt** **⑦ sytet** **⑧ flott** **⑨ Bytet ***	① 입력값 플럭스 ② 목표값 플럭스 ③ 잠재코드 차원값 ④ 입력 vocabulary갯수(선형데이터 이면 0) ⑤ 출력 vocabulary갯수(선형데이터 이면 0) ⑥ 워드 임베딩 차원값(선형데이터 이면 0) ⑦ 활성함수 코드값 ⑧ 학습률 ⑨ 네임 스코프 이름

출력매개변수		
반환값	Generic *	망 오브잭트
method	train predict accuracy messureAccucracy	학습 추론 정확도 측정 그라프 정착도 측정

이름	**impulse**	
설명	**Dynagen 묶음, 학습, 추론**	
헤더파일		
	형	**설명**
입력매개변수	**① Bytet ***	① 네임 스코프 이름
출력매개변수		
반환값	Dynagen*	망 오브잭트
method	train predict accuracy messureAccucracy	학습 추론 정확도 측정 그라프 정착도 측정

이름	**Dynagen**	
설명	**대용량 범용 신경망 생성**	
헤더파일	memput.h	
	형	**설명**
입력매개변수	**① Flux*** **② Flux*** **③ intt** **④ intt** **⑤ intt** **⑥ intt** **⑦ sytet** **⑧ flott** **⑩ Bytet ***	① 입력값 플럭스 ② 목표값 플럭스 ③ 잠재코드 차원값 ④ 입력 vocabulary갯수(선형데이터 이면 0) ⑤ 출력 vocabulary갯수(선형데이터 이면 0) ⑥ 워드 임베딩 차원값(선형데이터 이면 0) ⑦ 활성함수 코드값 ⑧ 학습률 ⑩ 네임 스코프 이름
출력매개변수		
반환값	Dynagen*	망 오브젝트

이름	stratus	
설명	시계열 신경망 생성	
헤더파일	memput.h	
	형	설명
입력매개변수	① Flux* ② Flux* ③ intt ④ intt ⑤ intt ⑥ intt ⑦ sytet ⑧ flott ⑪ Bytet *	① 입력값 플럭스 ② 목표값 플럭스 ③ 잠재코드 차원값 ④ 입력 vocabulary갯수(선형데이터 이면 0) ⑤ 출력 vocabulary갯수(선형데이터 이면 0) ⑥ 워드 임베딩 차원값(선형데이터 이면 0) ⑦ 활성함수 코드값 ⑧ 학습률 ⑪ 네임 스코프 이름
출력매개변수		
반환값	Stratus *	망 오브젝트
method	train predict accuracy messureAccucracy	학습 추론 정확도 측정 그라프 정확도 측정

Mempute Global Definition

```
TON     : 전치 안함
TOA     : 선행 플럭스 전치
TOB     :: 후행 플럭스 전치
TOT     : 양측 전치

ACTF_TANH       tanh
ACTF_RELU       relu
ACTF_SIGM       sigmoid
ACTF2_PRELU     prelu
```

Sample Code

```
#include  "memput.h"
Tracer *tcr = trace(1);
```

dot

```
        a = flux(tcr, { 2, 3 }, tfloat, variable);
        a->arange(2 * 3)->printo();
        b = flux(tcr, { 3,2 }, tfloat, variable);
        b->arange(3 * 2)->printo();
        c = a->dot(b, { {1}, {0} }, 0);
        c->printo();

        a = flux(tcr, { 3,4 }, tfloat, variable);
        a->arange(3 * 4)->printo();
        b = flux(tcr, { 3,2 }, tfloat, variable);
        b->arange(3 * 2)->printo();
        c = a->dot(b, { {0}, {0} }, 0);
        c->printo();

        a = flux(tcr, { 2,3 }, tfloat, variable);
        a->arange(2 * 3)->printo();
        b = flux(tcr, { 3,2 }, tfloat, variable);
        b->arange(3 * 2)->printo();
        c = a->dot(b, { {0}, {1} }, 0);
        c->printo();

        a = flux(tcr, { 2,3 }, tfloat, variable);
        a->arange(2 * 3)->printo();
        b = flux(tcr, { 3,4,2 }, tfloat, variable);
        b->arange(3 * 4 * 2)->printo();
        c = a->dot(b, { {1}, {0} }, 0);
        c->printo();
        //dot_bw_check(a, b, c);

        a = flux(tcr, { 2,3,4 }, tfloat, variable);
        a->arange(2 * 3 * 4)->printo();
        b = flux(tcr, { 3,2 }, tfloat, variable);
        b->arange(3 * 2)->printo();
        c = a->dot(b, { {1}, {0} }, 0);
        c->printo();

        a = flux(tcr, { 2,3 }, tfloat, variable);
        a->arange(2 * 3)->printo();
```

```
b = flux(tcr, { 4,3,2 }, tfloat, variable);
b->arange(4 * 3 * 2)->printo();
c = a->dot(b, { {1}, {1} }, 0);
c->printo();

a = flux(tcr, { 4,3,2 }, tfloat, variable);
a->arange(4 * 3 * 2)->printo();
b = flux(tcr, { 3,2,3 }, tfloat, variable);
b->arange(3 * 2 * 3)->printo();
c = a->dot(b, { {2}, {1} }, 0);
c->printo();

a = flux(tcr, { 2,3,4,3 }, tfloat, variable);
a->arange(2 * 3 * 4 * 3)->printo();
b = flux(tcr, { 3,2 }, tfloat, variable);
b->arange(3 * 2)->printo();
c = a->dot(b, { {1}, {0} }, 0);
c->printo();

a = flux(tcr, { 4,3,2 }, tfloat, variable);
a->arange(4 * 3 * 2)->printo();
b = flux(tcr, { 2,3 }, tfloat, variable);
b->arange(2 * 3)->printo();
c = a->dot(b, { {2}, {0} }, 0);
c->printo();

a = flux(tcr, { 4,3,2 }, tfloat, variable);
a->arange(4 * 3 * 2)->printo();
b = flux(tcr, { 3,2,3 }, tfloat, variable);
b->arange(3 * 2 * 3)->printo();
c = a->dot(b, { {1, 2}, {0, 1} }, 0);
c->printo();//(0,1,2,3,4,5) * (0,3,6,9,12,15)

a = flux(tcr, { 4,3,2 }, tfloat, variable);
a->arange(4 * 3 * 2)->printo();
b = flux(tcr, { 3,2,3 }, tfloat, variable);
b->arange(3 * 2 * 3)->printo();
c = a->dot(b, { {1, 2}, {1, 0} }, 0);
c->printo();//(0,1,2,3,4,5) * (0,6,12,3,9,15)

a = flux(tcr, { 4,3,2 }, tfloat, variable);
a->arange(4 * 3 * 2)->printo();
b = flux(tcr, { 3,2,2 }, tfloat, variable);
b->arange(3 * 2 * 2)->printo();
c = a->dot(b, { {1, 2}, {0, 2} }, 0);
c->printo();//(0,1,2,3,4,5) * (0,1,4,5,8,9)
```

```
a = flux(tcr, { 4,2,6 }, tfloat, variable);
a->arange(4 * 2 * 6)->printo();
b = flux(tcr, { 3,4,3 }, tfloat, variable);
b->arange(3 * 4 * 3)->printo();
c = a->dot(b, { {1, 2}, {0, 1} }, 0);
c->printo();

a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = flux(tcr, { 2,3,4,2 }, tfloat, variable);
b->arange(2 * 3 * 4 * 2)->printo();
c = a->dot(b, { {1,3}, {1,3} }, 0);
c->printo();

a = flux(tcr, { 2,3,4,2,4 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2 * 4)->printo();
b = flux(tcr, { 2,3,4,2,4 }, tfloat, variable);
b->arange(2 * 3 * 4 * 2 * 4)->printo();
c = a->dot(b, { {1,3}, {1,3} }, 0);
c->printo();

a = flux(tcr, { 2,3,4,2,1 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2 * 1)->printo();
b = flux(tcr, { 2,3,4,2,1 }, tfloat, variable);
b->arange(2 * 3 * 4 * 2 * 1)->printo();
c = a->dot(b, { {2,3}, {2,3} }, 0);
c->printo();

a = flux(tcr, { 2,3,4,2,1 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2 * 1)->printo();
b = flux(tcr, { 2,3,4,2,1 }, tfloat, variable);
b->arange(2 * 3 * 4 * 2 * 1)->printo();
c = a->dot(b, { {1,4}, {1,4} }, 0);
c->printo();

a = flux(tcr, { 4,3,2 }, tfloat, variable);
a->arange(4 * 3 * 2)->printo();
b = flux(tcr, { 6,2 }, tfloat, variable);
b->arange(6 * 2)->printo();
c = a->dot(b, { {1, 2}, {0} }, 0);
c->printo();//(0,1,2,3,4,5) * (0,2,4,6,8,10)

a = flux(tcr, { 2,1,3,4 }, tfloat, variable);
a->arange(2*1*3*4)->printo();
b = flux(tcr, { 3,2 }, tfloat, variable);
b->arange(3*2)->printo();
c = a->dot(b, { {2}, {0} }, 0);
```

```
c->printo();
```

mul

```
a = flux(tcr, { 3, 2 }, tfloat, variable);
a->arange(3*2)->printo();
b = flux(tcr, { 3,2 }, tfloat, variable);
b->arange(3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 2, 3, 2 }, tfloat, variable);
a->arange(2*3 * 2)->printo();
b = flux(tcr, { 3,2 }, tfloat, variable);
b->arange(3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 3, 2 }, tfloat, variable);
a->arange(3 * 2)->printo();
b = flux(tcr, { 2,3,2 }, tfloat, variable);
b->arange(2*3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 2,1, 3, 2 }, tfloat, variable);
a->arange(2 * 3 * 2)->printo();
b = flux(tcr, { 3,3,2 }, tfloat, variable);
b->arange(3*3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 2,1, 3, 2 }, tfloat, variable);
a->arange(2 * 3 * 2)->printo();
b = flux(tcr, { 2,3,3,2 }, tfloat, variable);
b->arange(2 * 3 * 3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 2,1,1, 3, 2 }, tfloat, variable);
a->arange(2 * 3 * 2)->printo();
b = flux(tcr, { 2,3,2,3,2 }, tfloat, variable);
b->arange(2 * 3 * 2* 3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 2,1,3,2 }, tfloat, variable);
a->arange(2 * 3*2)->printo();
b = flux(tcr, { 2,3,3,2 }, tfloat, variable);
```

```
b->arange(2 * 3 * 3 * 2)->printo();
c = a->div(b);
c->printo();

a = flux(tcr, { 2,1,3,2 }, tfloat, variable);
a->arange(2 * 3 * 2)->printo();
b = flux(tcr, { 3,3,2 }, tfloat, variable);
b->arange(3 * 3 * 2)->printo();
c = a->plus(b);
c->printo();

a = flux(tcr, { 2,1,1,1, 3, 2 }, tfloat, variable);
a->arange(2 * 3 * 2)->printo();
b = flux(tcr, { 2,3,2,2,3,2 }, tfloat, variable);
b->arange(2 * 3 * 2 *2* 3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 2,1,1, 3, 2 }, tfloat, variable);
a->arange(2 * 3 * 2)->printo();
b = flux(tcr, { 3,3,2 }, tfloat, variable);
b->arange(3*3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 2,4,1,2,1 }, tfloat, variable);
a->arange(2 * 4 * 2)->printo();
b = flux(tcr, { 3,2,1 }, tfloat, variable);
b->arange(3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 2,4,1,2,1,1 }, tfloat, variable);
a->arange(2 * 4 * 2)->printo();
b = flux(tcr, { 3,2,1,1 }, tfloat, variable);
b->arange(3 * 2)->printo();
c = a->mul(b);
c->printo();


a = flux(tcr, { 2,4,1,2,1,1,2 }, tfloat, variable);
a->arange(2 * 4 * 2*2)->printo();
b = flux(tcr, { 3,2,1,1,2 }, tfloat, variable);
b->arange(3 * 2*2)->printo();
c = a->mul(b);
c->printo();
```

```
a = flux(tcr, { 2,1,2,1,1,2 }, tfloat, variable);
a->arange(2*2 * 2)->printo();
b = flux(tcr, { 3,2,1,1,2 }, tfloat, variable);
b->arange(3 * 2*2)->printo();
c = a->mul(b);
c->printo();


a = flux(tcr, { 2,1,2,2,1,2 }, tfloat, variable);
a->arange(2 * 2 * 2 *2)->printo();
b = flux(tcr, {   3,2,1,3,2 }, tfloat, variable);
b->arange(3 * 2 * 3*2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 1,1, 3, 2 }, tfloat, variable);
a->arange(3 * 2)->printo();
b = flux(tcr, { 2,3,3,2 }, tfloat, variable);
b->arange(2 * 3 * 3 * 2)->printo();
c = a->mul(b);
c->printo();

a = flux(tcr, { 1,2, 3, 1 }, tfloat, variable);
a->arange(3 * 2)->printo();
b = flux(tcr, { 2,2,3,1 }, tfloat, variable);
b->arange(2 * 2 * 3 * 1)->printo();
c = a->mul(b);
c->printo();
```

transpose

```
a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->transpose({ 0,1,3,2 });
b->printo();
trs_bw_check(a, b);
b = a->transpose({ 3,1,0,2 });
b->printo();

a = flux(tcr, { 2,3,1,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->transpose({ 3,1,0,4,2 });
b->printo();
```

stack, concat

```
a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->unstack(0);
printo(b);
c = stack(b, 0);
c->printo();

b = a->split(2, 0);
printo(b);
c = concat(b, 0);
c->printo();

a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->unstack(1);
printo(b);
c = stack(b, 1);
c->printo();

b = a->split(3, 1);
printo(b);
c = concat(b, 1);
c->printo();

a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->unstack(2);
printo(b);
c = stack(b, 2);
c->printo();

b = a->split(4, 2);
printo(b);
c = concat(b, 2);
c->printo();

a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->unstack(3);
printo(b);
c = stack(b, 3);
c->printo();

b = a->split(2, 3);
printo(b);
```

```
c = concat(b, 3);
c->printo();

a = flux(tcr, { 2,3,1 }, tfloat, variable);
a->arange(2 * 3)->printo();
b = a->unstack(0);
printo(b);
c = stack(b, 0);
c->printo();

b = a->split(2, 0);
printo(b);
c = concat(b, 0);
c->printo();

a = flux(tcr, { 2,3,1 }, tfloat, variable);
a->arange(2 * 3)->printo();
b = a->unstack(2);
printo(b);
c = stack(b, 2);
c->printo();

b = a->split(1, 2);
printo(b);
c = concat(b, 2);
c->printo();


a = flux(tcr, { 3,2,2 }, tfloat, variable);
a->arange(3*2*2)->printo();
vector<Flux *> l;
l.push_back(a);
a = flux(tcr, { 3,1,2 }, tfloat, variable);
a->arange(3 * 1 * 2)->printo();
l.push_back(a);
a = flux(tcr, { 3,3,2 }, tfloat, variable);
a->arange(3 * 3 * 2)->printo();
l.push_back(a);
c = concat(&l, 1);
c->printo();

l.clear();
a = flux(tcr, { 1,6 }, tfloat, variable);
a->arange(6)->printo();
l.push_back(a);
a = flux(tcr, { 6,6 }, tfloat, variable);
a->arange(6*6)->printo();
```

```
l.push_back(a);
c = concat(&l, 0);
c->printo();

l.clear();
a = flux(tcr, { 6,1 }, tfloat, variable);
a->arange(6)->printo();
l.push_back(a);
a = flux(tcr, { 6,6 }, tfloat, variable);
a->arange(6 * 6)->printo();
l.push_back(a);
c = concat(&l, 1);
c->printo();
```

unstuck, split

```
a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->unstack(0);
printo(b);
split_bw_check(a, b);
c = stack(b, 0);
//c->printo();

b = a->split(2, 0);
printo(b);
split_bw_check(a, b);
c = concat(b, 0);
//c->printo();


a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->unstack(1);
printo(b);
split_bw_check(a, b);
c = stack(b, 1);
//c->printo();

b = a->split(3, 1);
printo(b);
split_bw_check(a, b);
c = concat(b, 1);
//c->printo();

a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->unstack(2);
```

```
printo(b);
split_bw_check(a, b);
c = stack(b, 2);
c->printo();

b = a->split(4, 2);
printo(b);
split_bw_check(a, b);
c = concat(b, 2);
c->printo();

a = flux(tcr, { 2,3,4,2 }, tfloat, variable);
a->arange(2 * 3 * 4 * 2)->printo();
b = a->unstack(3);
printo(b);
split_bw_check(a, b);
c = stack(b, 3);
c->printo();

b = a->split(2, 3);
printo(b);
split_bw_check(a, b);
c = concat(b, 3);
//c->printo();

a = flux(tcr, { 2,3,1 }, tfloat, variable);
a->arange(2 * 3)->printo();
b = a->unstack(0);
printo(b);
split_bw_check(a, b);
c = stack(b, 0);
//c->printo();

b = a->split(2, 0);
printo(b);
split_bw_check(a, b);
c = concat(b, 0);
//c->printo();

a = flux(tcr, { 2,3,1 }, tfloat, variable);
a->arange(2 * 3)->printo();
b = a->unstack(2);
printo(b);
split_bw_check(a, b);
c = stack(b, 2);
//c->printo();
```

```
b = a->split(1, 2);
printo(b);
split_bw_check(a, b);
c = concat(b, 2);
//c->printo();

a = flux(tcr, { 3,2,2 }, tfloat, variable);
a->arange(3 * 2 * 2)->printo();
vector<Flux *> l;
l.push_back(a);
a = flux(tcr, { 3,1,2 }, tfloat, variable);
a->arange(3 * 1 * 2)->printo();
l.push_back(a);
a = flux(tcr, { 3,3,2 }, tfloat, variable);
a->arange(3 * 3 * 2)->printo();
l.push_back(a);
c = concat(&l, 1);

l.clear();
a = flux(tcr, { 1,6 }, tfloat, variable);
a->arange(6)->printo();
l.push_back(a);
a = flux(tcr, { 6,6 }, tfloat, variable);
a->arange(6*6)->printo();
l.push_back(a);
c = concat(&l, 0);

l.clear();
a = flux(tcr, { 6,1 }, tfloat, variable);
a->arange(6)->printo();
l.push_back(a);
a = flux(tcr, { 6,6 }, tfloat, variable);
a->arange(6 * 6)->printo();
l.push_back(a);
c = concat(&l, 1);
```

slice

```
a = flux(tcr, { 4,3,2 }, tint, variable);
a->arange(-1);

b = a->slice({ {0,1}, {0, 2} });
b->printo();

b = a->slice({ {0,-1,2}, {0,-1, 2} });
b->printo();

b = a->slice({ {1,-2}, {1,-2} });
b->printo();

b = a->slice({ {1,-1}, {1,-1} });
b->printo();

a->slice({ {1}, {-1} })->printo();

a->slice({ {}, {-2}, {-2} })->printo();

a->slice({ {1}, {1}, {1} })->printo();

a->slice({ {}, {-2}, {-3} })->printo();

a->slice({ {}, {-2}, {3} })->printo();
```

one_hot, argmax, equal, not_equal, expand_dims, squeeze

```
Flux *a, *b, *c;

a = flux(tcr, "[[0, 2 ],₩
       [3, -1]]");
a->printo();
printf("----------------0₩n");
b = a->one_hot(4, 5.5, 0, 0);
b->printo();

a = flux(tcr, "[[0, 2 ],₩
       [3, 1]]");
a->printo();
printf("----------------0₩n");
b = a->one_hot(4, 5.0, 0, 0);
b->printo();
printf("----------------1₩n");
b = a->one_hot(3, 5.0, 0, 1);
b->printo();
printf("----------------2₩n");
b = a->one_hot(3, 5.0, 0, 2);
b->printo();
printf("----------------- -1₩n");
b = a->one_hot(3, 5.0, 0, -1);
b->printo();

printf("----------------₩n");
a = flux(tcr, "[[[0, 2 ], [3, 1]],₩
       [[0, 2 ], [3, 1]]]");
a->printo();
printf("----------------0₩n");
b = a->one_hot(4, 5.0, 0, 0);
b->printo();
printf("----------------1₩n");
b = a->one_hot(3, 5.0, 0, 1);
b->printo();
printf("----------------2₩n");
b = a->one_hot(3, 5.0, 0, 2);
b->printo();
printf("----------------3₩n");
b = a->one_hot(3, 5.0, 0, 3);
b->printo();

a = flux(tcr, "[[[0.1, 0.3, 0.5],₩
       [0.3, 0.5, 0.1]],₩
       [[0.5, 0.1, 0.3],₩
```

```
        [0.1, 0.3, 0.5]],₩
        [[0.3, 0.5, 0.1],₩
        [0.5, 0.1, 0.3]]]");
a->printo();
b = a->argmax(0);
b->printo();
b = a->argmax(1);
b->printo();
b = a->argmax(2);
b->printo();
printf("----------------₩n");
a = flux(tcr, "[ [[[0.1, 0.3, 0.5],₩
        [0.3, 0.5, 0.1]],₩
        [[0.5, 0.1, 0.3],₩
        [0.1, 0.3, 0.5]],₩
        [[0.3, 0.5, 0.1],₩
        [0.5, 0.1, 0.3]]],₩
        [[[0.1, 0.3, 0.5],₩
        [0.3, 0.5, 0.1]],₩
        [[0.5, 0.1, 0.3],₩
        [0.1, 0.3, 0.5]],₩
        [[0.3, 0.5, 0.1],₩
        [0.5, 0.1, 0.3]]] ]");
a->printo();
printf("----------------0₩n");
b = a->argmax(0);
b->printo();
printf("----------------1₩n");
b = a->argmax(1);
b->printo();
printf("----------------2₩n");
b = a->argmax(2);
b->printo();
printf("----------------3₩n");
b = a->argmax(3);
b->printo();

printf("-----------------------------------₩n");
a = flux(tcr, "[[[0.1, 0.3, 0.5],₩
        [0.3, 0.5, 0.1]],₩
        [[0.5, 0.1, 0.3],₩
        [0.1, 0.3, 0.5]],₩
        [[0.3, 0.5, 0.1],₩
        [0.5, 0.1, 0.3]]]");

b = flux(tcr, "[[[0.1, 0.2, 0.5],₩
        [0.3, 0.6, 0.1]],₩
```

```
        [[0.5, 0.1, 0.3],₩
        [0.1, 0.3, 0.5]],₩
        [[0.2, 0.5, 0.1],₩
        [0.5, 0.1, 0.7]]]");

c = a->equal(b);
c->printo();

c = a->not_equal(b);
c->printo();

c = a->equal(0.5);
c->printo();

a = flux(tcr, { 4,3,2 }, tfloat, trainable);
a->shape();
a->arange(-1);
a->printo();

b = a->expand_dims(1);
b->shape();
b->printo();

c = b->squeeze();
c->shape();
c->printo();

c = b->squeeze(1);
c->shape();
c->printo();
```

matmul

```
a = flux(tcr, { 3,2 }, tfloat, variable);
a->arange(-1);
a->printo();
b = flux(tcr, { 2,3 }, tfloat, variable);
b->arange(-1);
b->printo();
c = a->matmul(b);
c->printo();

a = flux(tcr, {4,3,2 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 4,2,3 }, tfloat, variable);
b->arange(-1);
c = a->matmul(b);
c->printo();


a = flux(tcr, { 4,2,3 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 4,2,3 }, tfloat, variable);
b->arange(-1);
c = a->matmul(b, 1);
c->printo();

a = flux(tcr, { 4,3,2 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 4,3,2 }, tfloat, variable);
b->arange(-1);
c = a->matmul(b, TOB);
c->printo();

a = flux(tcr, { 4,2,3 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 4,3,2 }, tfloat, variable);
b->arange(-1);
c = a->matmul(b, TOT);
c->printo();

a = flux(tcr, { 4,5,3 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 4,3,5 }, tfloat, variable);
b->arange(-1);
c = a->matmul(b);
```

```
c->printo();

a = flux(tcr, { 4,5,3 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 4,5,3 }, tfloat, variable);
b->arange(-1);
c = a->matmul(b, TOA);
c->printo();

a = flux(tcr, { 4,3,5 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 4,3,5 }, tfloat, variable);
b->arange(-1);
c = a->matmul(b, TOB);
c->printo();

a = flux(tcr, { 4,3,5 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 4,5,3 }, tfloat, variable);
b->arange(-1);
c = a->matmul(b, TOT);
c->printo();

a = flux(tcr, { 16,16,7 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 16,7,16 }, tfloat, trainable, Initializer::xavier);
b->arange(-1);
c = a->matmul(b);
c->printo();

a = flux(tcr, { 16,7,16 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 16,7,16 }, tfloat, trainable, Initializer::xavier);
b->arange(-1);
c = a->matmul(b, TOA);
c->printo();

a = flux(tcr, { 16,16,7 }, tfloat, variable);
a->arange(-1);
b = flux(tcr, { 16,16,7 }, tfloat, trainable, Initializer::xavier);
b->arange(-1);
c = a->matmul(b, TOB);
c->printo();


a = flux(tcr, { 16,7,16 }, tfloat, variable);
a->arange(-1);
```

```
b = flux(tcr, { 16,16,7 }, tfloat, trainable, Initializer::xavier);
b->arange(-1);
c = a->matmul(b, TOT);
c->printo();
```

graph exec

```
#define BATCH_SZ 16
#define X_TIME_SIZE 64
#define Y_TIME_SIZE X_TIME_SIZE
#define  FEATURE_SIZE 1
#define  HIDDEN_SIZE 32


	Tracer *tcr2 = trace(1);
	Flux *sample_x, *sample_y, *state;
	sample_x = flux(tcr2, { BATCH_SZ, X_TIME_SIZE, FEATURE_SIZE }, tfloat,
trainable);
	sample_y = flux(tcr2, { BATCH_SZ, X_TIME_SIZE, FEATURE_SIZE }, tfloat,
trainable);
	state = flux(tcr2, { BATCH_SZ, HIDDEN_SIZE }, tfloat, trainable);

	sample_x->randn(0, 0.5);
	sample_y->randn(0, 0.5);
	state->fill((floatt)0);
	tcr->sizeBatch(4);
	unit lap;
	float loss = 1000;
	for(intt i = 0;i < 100; i++) {
		rnn_input->feedf(sample_x);
		rnn_output->feedf(sample_y);
		init_state->feedf(state);
		lap = xucurrenttime();
		tcr->run({ op, total_loss });
		total_loss->printo();
		if(loss < *(floatt *)total_loss->begin_p()) {
			printf("!!! later big loss %fWn", *(floatt *)total_loss-
>begin_p());
		}
		loss = *(floatt *)total_loss->begin_p();
	}
	Flux *test_x = flux(tcr, { 1, X_TIME_SIZE, FEATURE_SIZE }, tfloat,
variable);
	test_x->randn(0, 0.5);
	for(intt i = 0;i < 3; i++) {
		rnn_input->feedf(test_x);
		rnn_output->feedf(test_x);
		init_state->feedf(state);
		lap = xucurrenttime();
		tcr->run({ total_loss, cy_pred });
		cy_pred->printo();
		total_loss->printo();
	}
```

```
delete tcr;
```

MEMPUTE V1

PATTERN NETWORK DATABASE

- 매뉴얼 -

패턴 네트워크 데이터베이스의 명령은 파이썬 MEMPUTE 크라이언트 API 와 서버측의 SQL 명령어의 결합으로 구성된다.

파이썬 클라이언트 API

driver

- Pattern network database 접속 드라이버를 로드한다.

connect

- Pattern network database 접속 세션을 개설한다.

statement

- Pattern network database sql 문장을 실행할 문장 객제를 생성한다..

direct

- 패턴 네트워크 데이터베이스 sql 문장을 실행한다.

mempute

- 패턴 네트워크 데이터베이스 sql 문장을 실행하고 결과로 파이썬 배열을 리턴받는다..

array

- 패턴 네트워크 데이터베이스와 데이터를 통신하기위한 배열을 생성한다.

inarray

- 서버측 배열에 클라이언트측 배열 데이터를 로드한다.

focus

- 배열을 포커싱하여 후행 작업의 대상이 되게 한다.

Ex)

```
drv = mp.driver(3)
con = mp.connect(drv, "loopback", "", "")
stmt = mp.statement(con)
mp.focus(stmt, "execute mempute('array', 'eval_input')")
mp.inarray(stmt, x_test, 1)
```

스키마 구성

Perception

- 학습 단위

퍼셉션 생성

- 퍼셉션 이름과 퍼셉션이 위치할 로케일 이름을 명시한다. 로케일 이름이 주어지지 않으면 기본 위치에 퍼셉션이 생성된다.

syntax

execute mempute('perception', 'perception_name locale locale_name positional')

execute : sql 실행 명령

mempute : 패턴 네트워크 실행 명령

perception : 퍼셉션 생성 명령

perception_name : 퍼셉션 이름

locale : 퍼셉션 생성 위치, 생략 하면 기본 위치에 생성

locale_name : 퍼셉션 생성 위치 이름

positional : [unfix|0|1|2]

- 시퀀스에 포지션 지정 여부 지시
- Unfix: 포시션 없음
- 0 : 포지션 없음
- 1 : 목표값만 포지션 설정
- 2 : 입력값과 목표값 모두 포지션 설정

ex)

mp.direct(stmt, "execute mempute('perception', 'anormal locale percep_loc')")

sequence

- 입력과 타겟 데이터의 길이 설정

syntax

execute mempute('sequene', input_size, target_size, 'sequence_reduce_idx', scale_figure)

sequence : 시퀀스 설정 명령

input_size : 입력값 길이 설정

taget_size : 목표값 길이 설정

sequence_reduce_idx : 시퀀스 아이템별 차원축소할 인덱스 나열

scale_figure : 양수이면 스케일 업 계수, 소수점 이하 지정 계수까지 취득, 음수이면 스케일 다운 계수, 소수점 이상 지정 계수까지 취득, 0 이면 스케일 조정 없

음, 생략이면 기본값으로 스케일 업 2

ex)

```
mp.direct(stmt, "execute mempute('sequence', 5, 3, 2)")
```

channel

- **데이터 구조 기술**

syntax

execute mempute('channel', division, 'attribute')

channel : 데이터 구조 설정 명령

division : 0 – 입력 구조 기술, 1 – 목표 구조 기술, 2 – 채널 기술 시작, 3 – 채널 기술 마감, 한번에 구성할 경우 2 와 3 생략 가능

attribute : { description_enum }

description_enum : description_enum | description

description : prefix figure type

prefix : scale | reduction | union | narrow

scale : l | L

l : scale up

L : scale down

reduction : r | R #차원 축소

union : union_notation scale_down_or_not #차원축소와 스케일 조정 함께 적용

union_notation : u | U

scale_down_or_not : L | #스케일 다운 혹은 지정하지 않으면 스케일 업

narrow : narrow_notation scale_down_or_not #채널별로 차원축소와 스케일 조정 함께 적용

narrow_notation: n | N

scale_down_or_not: L | # 스케일 다운 혹은 지정하지 않으면 스케일 업

figure : scale_digit . reduction_digit

scale_digit : #스케일 조정 계수

reduction_digit: #차원 축소 계수, 축소할 차원수의 역수

type : b | s | f | i | l | d #b(byte), s(short), f(float), i(32bit int), l(64bit int), d(double)

ex)

```
mp.direct(stmt, "execute mempute('channel', 1, '{u0.02d}')")
```

array

- **Pattern network database 와 데이터 입출력 인터페이스**

syntax

execute mempute('array', 'array_name locale locale_name division mode id_gen seq_save batch_size num_devide')

array : 배열 생성 및 참조 명령

array_name : 배열 이름

locale : 퍼셉션 생성 위치, 배열을 생성할 때 생략 하면 메모리 배열 생성, 참조 할때 로케일 이하 생략하면 기존에 배열 이름으로 등록된 배열 리턴

locale_name : 퍼셉션 생성 위치 이름

division : 1 – 입력배열, 0 – 출력배열, 2 – 입력 & 마인드 구성

mode : 1 – 바이너리, 0 – 문자형

id_gen : -1 – 시퀀스 아이디 엔진에서 생성하지 않고 입력에서 주어짐, 0 – 기본값으로 아이디 생성, 1 – 아이디 생성

seq_save : 0 – 시퀀스 저장 안함, 1 – 시퀀스 저장

size_batch : 일괄 실행 단위 설정, -1 - 32 양수 최대값, 0 – 5000,

num_devide : 1 - 컨볼루션 연산 윈도우 커널 스트라이드할때 하나의 원 이미지에서 윈도우 스트라이드 갯수만큼 설정하여 파생 시퀀스들을 하나로 묶는다, 0 – 안함

ex)

```
mp.array(stmt, "execute mempute('array', 'anormal_input locale array_loc 1 1 1 0 0 0')")
```

load

- **저장된 배열에서 메모리 로드**

syntax

execute mempute('load', 'array_name', start, end)

load : load command

array_name : array name

start : load 시작 시퀀스 아이디

end : load 끝 시퀀스 아이디, -1 이면 끝까지 로드

ex)

mp.direct(stmt, "execute mempute('load', 'anormal_target', 0, -1)")

cognit

- **학습 명령**

syntax

execute mempute('cognit', 'input_array', 'target_array')

ex)

mp.direct(stmt, "execute mempute('cognit', 'anormal_input', 'anormal_target')")

predict

- **추론 명령**

syntax

execute mempute('predict', 'input_array', 'output_array')

return value : 예측값 파이썬 배열

ex)

predictions = mp.mempute(stmt, "execute mempute('predict', 'eval_input', 'eval_output')")

oblivion

- **학습 완료 망각 명령**

syntax

execute mempute('oblivion', 0)

ex)

mp.direct(stmt, "execute mempute('oblivion', 0)")

phyper

- **Hyper parameter**
- **Sensitivity #민감도 설정**

ex)

mp.direct(stmt, "execute mempute('phyper', 'sensitivity 2.7')")

discriminate

- **판별 데이터를 생성한다.**

Syntax 1

execute mempute('discriminate', 'operate', 'parameter')

discriminate : 판별 명령

operate : 0 | 1 | 2 | 3 | 4

0 : 입력으로부터 추출되는 최종레벨 패턴 아이디 시퀀스를 출력하기위한 출력 시퀀스 사이즈 설정

Parameter : output_sequence_size

Ex) mp.mempute(stmt, execute mempute('discriminate', 0, 5)")

1 : revise inference 하기위해 패턴아이디 시퀀스를 입력받기위한 입력시퀀스 사이즈 설정

Parameter : input_sequence_size

Ex) mp.mempute(stmt, execute mempute('discriminate', 1, 5)")

2 : syntax 2 의 operate 1 의 강도순 연속 아이디 공간 사이즈 리턴

Ex) vocabulary_size = mp.mempute(stmt, "execute mempute('discriminate', 2, -1)")

3 : syntax 2 의 operate 1 의 수행과정에서 연속 아이디를 빌드하는 범위의 하한 레벨 설정, 이를 수행안하면 탑베렐만 대상으로 아이디 빌드

Ex) mp.mempute(stmt, "execute mempute('discriminate', 3, 8)")

4 : syntax 2 의 operate 103 의 입력의 특징 feature 를 추출하는데 있어서 parmeter 값이 1 이면 특징과 일치하는 입력 데이트를 그대로 출력 옵션, 0 이면 채널설정의 차원축소 옵션에 따라 차원 축소된 데이터를 출력

Ex) mp.mempute(stmt, "execute mempute('discriminate', 4, 1)")

syntax 2

execute mempute('discriminate', 'input_array', 'operate', 'parameter', 'output_array',

'target_array')

discriminate : 판별 명령

input_array : 입력 데이터 배열 이름

output_array : 출력 데이터 배열 이름

target_array : 목표 데이터 배열 이름

operate : 0 | 1 | 100 | 101 | 102 | 103

0 : 입력 데이터에 대한 사전 학습이 선행된 후에 시퀀스 강도 오더링, input_array 의 각 시퀀스별로 phyper test_margin 으로 설정된 레벨이상의 패턴들만 총합을 구하여 정렬

Parameter : 0 | 1 | 2 | 3

0 : 강도만 오더링 & descending

1 : 강도만 오더링 & ascending,

2 : 강도+탑레렙 넘버 오더링 & descending

3 : 강도+탑레렙 넘버 오더링 & ascending

Output_array format

[Sequence_id order_index strength_sum top_level_number]

Ex) mp.mempute(stmt, "execute mempute('discriminate', 'anormal_input', 0, 3, 'disc_output')")

1 : 패턴을 강도순으로 연속 아이디로 정의, input_array 의 각 시퀀스별로 탑레벨 패턴 혹은 phyper all_lev_build 가 설정되면 모든 레벨 패턴을 강도 기준으로 연속 아이디로 정의하고 각 패턴을 생성 아이디로 대체하여 output_array 에 출력한다.

Parameter: 0 | 1 | 2 | 3

0 : 로드 & 결과리턴

1 : 빌드 & 결과리턴

2 : 빌드 & 저장

3 : 빌드 & 저장 & 결과리턴

Ex)

mp.mempute(stmt, execute mempute('discriminate', 0, 5)")

mp.mempute(stmt, "execute mempute('discriminate', 'anormal_input', 1, 3, 'disc_output')")

100 : 외부 데이터에 대한 차원 축소 빌드 서비스

Parameter : 차원축소 사이즈

Ex)

mp.mempute(stmt, execute mempute('discriminate', 100, 20)")

101 : 외부 데이터에 대한 차원 축소 서비스

Parameter : 없음

Return value : eval_input 배열 데이터에 대한 차원 축소된 아이디를 desc_out 배열에 적재하여 리턴

Ex)

r = mp.mempute(stmt, "execute mempute('discriminate', 'eval_input', 101, 0, 'disc_output')")

102 : 외부 데이터에 대한 차원 축소 사이즈 리턴

Parameter : 없음

Return value : 외부 데이터에 대한 차원 축소 사이즈 리턴

Ex)

Vocabulary_size = mp.mempute(stmt, "execute mempute('discriminate', 102, -1)")

103 : 입력 데이터로부터 추출된 패턴들중에서 목표값 페턴을 가장 특징짓는 입력 feature 추출(타겟배열이 주어질때) 또는 사후 추론 결과를 가장 특징짓는 입력 feature 추출(타겟배열이 주어지지않을 때)

Parameter : margin #입력추출 패턴 탑 레벨에서 margin 레벨 개수만큼 하위 범위의 입력 패턴들을 대상으로 operation 수행

Return value : 입력 데이터에서 특징 추출이된 입력 feature 이외 나머지 원소들은 nan 값으로 재워진 입력배열과 동일 사이즈의 클라이언트 배열

Ex)

pickout = mp.mempute(stmt, "execute mempute('discriminate', 'anormal_input', 103, 1, 'anormal_output', 'anormal_target')")

associate - 입력 데이터의 마인드를 추출해 연상되는 타겟 장면을 리턴하는 연상 명령.

associate scene_id, scene_id2 - 두번째 장면이 첫번째 장면에 부합되는 총 마인

드 강도 합계 리턴

예) execute mempute('associate', 200, 201) - 선언된 타겟 게이트의 장면에서 200 번째와 201 번째의 부합 강도 리턴

associate scene_id
- 명시된 장면이 선언된 타겟 게이트의 장면에서 마인드 부합되는 총 강도 합계가 큰 순으로 장면 아이디를 result set 으로 반환.
- 이때 타겟 게이트는 mind 지지 명령으로 마인드 생성되있어야 한다.

예) execute mempute('associate', 200)

associate accord_string - 선언된 타겟 게이트에 대한 집합 연상 명령, 이때 타겟 게이트는 mind 지지 명령으로 마인드 생성되있어야 한다.

accord_string - accord/against inner/outer scene_id..

accord - scene_id 와 부합되는 장면 마인드
against - scene_id 에 반하는 장면 마인드
inner - 언급되는 장면들의 공통 마인드
outer - 언급되는 장면들의 union 마인드

예) execute mempute('associate', 'accord inner 200 300 against inner 201 301')
- 200, 300 장면의 공통 마인드에는 부합하고 201, 301 의 공통 마인드에는 반하는 장면들울 추출.

merge
- execute mempute('merge', suffix_1_or_0, 'merge_suffix', seperate_1_or_0, oblivion_1_or_0);
- 두개 인지망 학습 결과를 하나로 합치는 명령
- oblivion_1_or_0 - merge 후에 망각 루틴을 수행 할것인지 스텝 설정, 0 이면 망각 루틴 수행 안함

- seperate_1_or_0 - 1 이면 합쳐질 두개 인지망을 독립 인지망, 즉 워드 및 장면 일력이 별개로 입력.0 이면 워드 및 장면을 공유하고 학습만 서로 다른 범위를 수행한 결과를 merge
- suffix_1_or_0 가 1 이면 perception network 명령으로 합칠 인지망(호스트)을 지정한후 합쳐질 인지망게스트)을 merge_suffix 에 suffix 이름으로 설정, 호스트 인지망 이름에 suffix 가 추가된 인지망 이름의 게스트 인지망을 호스트 인지망에 merge

예)
execute mempute('percenption network', 'perception_name') - 합칠 호스트 인지망 설정.
execute mempute('merge', 1, 'guest_suffix_name', 0, 1/0) - 합쳐질 게스트 인지망 지시, merge 수행.

- 두개 독립적 인지망을 합치는 경우
예)

```
execute mempute('perception network', 'lang_pnet locale atom_db');

execute mempute('array', 'lang_array locale scene_db');

execute mempute('write', 1);

mempute load eng1_5000.txt kor1_5000.txt;

execute mempute('range', 0, 0);

execute mempute('percept');

execute mempute('perception network', 'lang_pnet2 locale atom_db');

execute mempute('array', 'lang_array2 locale scene_db');
```

execute mempute('write', 1);

mempute load eng5001_10000.txt kor5001_10000.txt;

execute mempute('range', 0, 0);

execute mempute('percept');

execute mempute('perception network', 'lang_pnet2');

execute mempute('array', 'lang_array2');

execute mempute('keep', 'lang_pnet2', 1);

execute mempute('perception network', 'lang_pnet');

execute mempute('array', 'lang_array');

execute mempute('merge', 0, 'lang_pnet2', 1, 1);

import - 원격의 인지망 학습 결과를 perception network 명령으로 포커스된 인지망으로 가져오는 명령

mempute import url [imp_percept_network_locale], [imp_a_percept_network_name], [imp_suffix_name]

url - 임포트할 원격지 설정 ip_address[:port_name] - port_name 을 생략하면 기본 포트로 접속

예)
execute mempute('import', 'loopback:45003')
- perception network 명령으로 설정된 호스트 인지망과 동일한 이름을 가진 원

격 인지망으로부터 학습 결과를 가져옴.

execute mempute('import', 'loopback:45003', 'imp_suffix_name')동
- perception network 명령으로 설정된 호스트 인지망과 동일한 이름과 imp_suffix_name 을 가진 원격 인지망으로부터 학습 결과를 가져옴.

execute mempute('import', 'loopback:45003', 'imp_percept_network_locale', 'imp_a_percept_network_name');
- 명시된 퍼셉션 이름과 로케일을 가진 원격 인지망으로 부터 학습 결과를 가져옴.

execute mempute('import', 'loopback:45003', 'imp_percept_network_locale', 'imp_a_percept_network_name', 'merge_suffix_name;');
- 명시된 퍼셉션 이름과 로케일, suffix 를 가진 원격 인지망으로 부터 학습 결과를 가져옴.

rebert - array 명령으로 설정된 포커스 게이트를 리셋 시킨다. 다른 array 명령을 수행하지 않고 리셋 시킬때 사용.

예) execute mempute('rebert')

reset - array 설정된 학습 포커스 리셋

reset step

step
learn - 학습 결과, 마인드 리셋
array - 배열, 마인드 리셋
all - 배열, 학습 결과, 마인드 리셋

예) execute mempute('reset', 'learn/array/all')

erase - array 설정된 학습 포커스 삭제

erase step

step
learn - 학습 결과, 마인드 삭제
array - 배열, 마인드 삭제
all - 배열, 학습 결과, 마인드 삭제

clear - 퍼셉션 단위 일괄 삭제 명령

execute mempute('clear', 'percept_name suffix_name percept_loc array_name array_loc step(learn/array/all) erase_api(erase/reset) rm_loc(erase/0)');

clear 명령 ex).

--execute mempute('clear', 'percept_name suffix_name percept_loc array_name array_loc step(learn/array/all) erase_api(erase/reset) rm_loc(erase/0)');

execute mempute('clear', 'atom_lotto 0 atom_db scene_lotto scene_db learn reset 0');

execute mempute('clear', 'atom_lotto 0 atom_db scene_lotto scene_db array reset 0');

execute mempute('clear', 'atom_lotto 0 atom_db scene_lotto scene_db all reset 0');

execute mempute('clear', 'atom_lotto 0 atom_db scene_lotto scene_db learn erase 0');

execute mempute('clear', 'atom_lotto 0 atom_db scene_lotto scene_db array erase 0');

-- 이 명령은 의미없다. scene_lotto 게이트만 선택되어 게이트 삭제는 위에서 됐으므로, all 은 퍼셉션을 삭제 하는 것인데 이럴려면 선택할때 scene_lotto 게이트 명을 주면 안된다.
execute mempute('clear', 'atom_lotto 0 atom_db scene_lotto scene_db all erase 0');

execute mempute('clear', 'atom_lotto 0 atom_db 0 0 learn erase 0');

execute mempute('clear', 'atom_lotto 0 atom_db 0 0 array erase 0');

execute mempute('clear', 'atom_lotto 0 atom_db 0 0 all erase 0');

-- 로케일 삭제는 루트 권한이 필요하므로 사용자가 루트 디비를 직접 로드한다.
load db root;

execute mempute('clear', 'atom_lotto 0 atom_db 0 0 array erase erase');
-- 위에서 all erase 로 퍼셉망의 모든 게이트를 삭제했으면 퍼셉망 이름 테이블에 해당 퍼셉망 정보가 없으므로 이럴경우네는 로케일 이름을 명시해야 한다.
execute mempute('clear', 'atom_lotto 0 atom_db 0 0 all erase erase');

학습 모니터링
Sqn> execute mempute('view', 'perception_name');