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

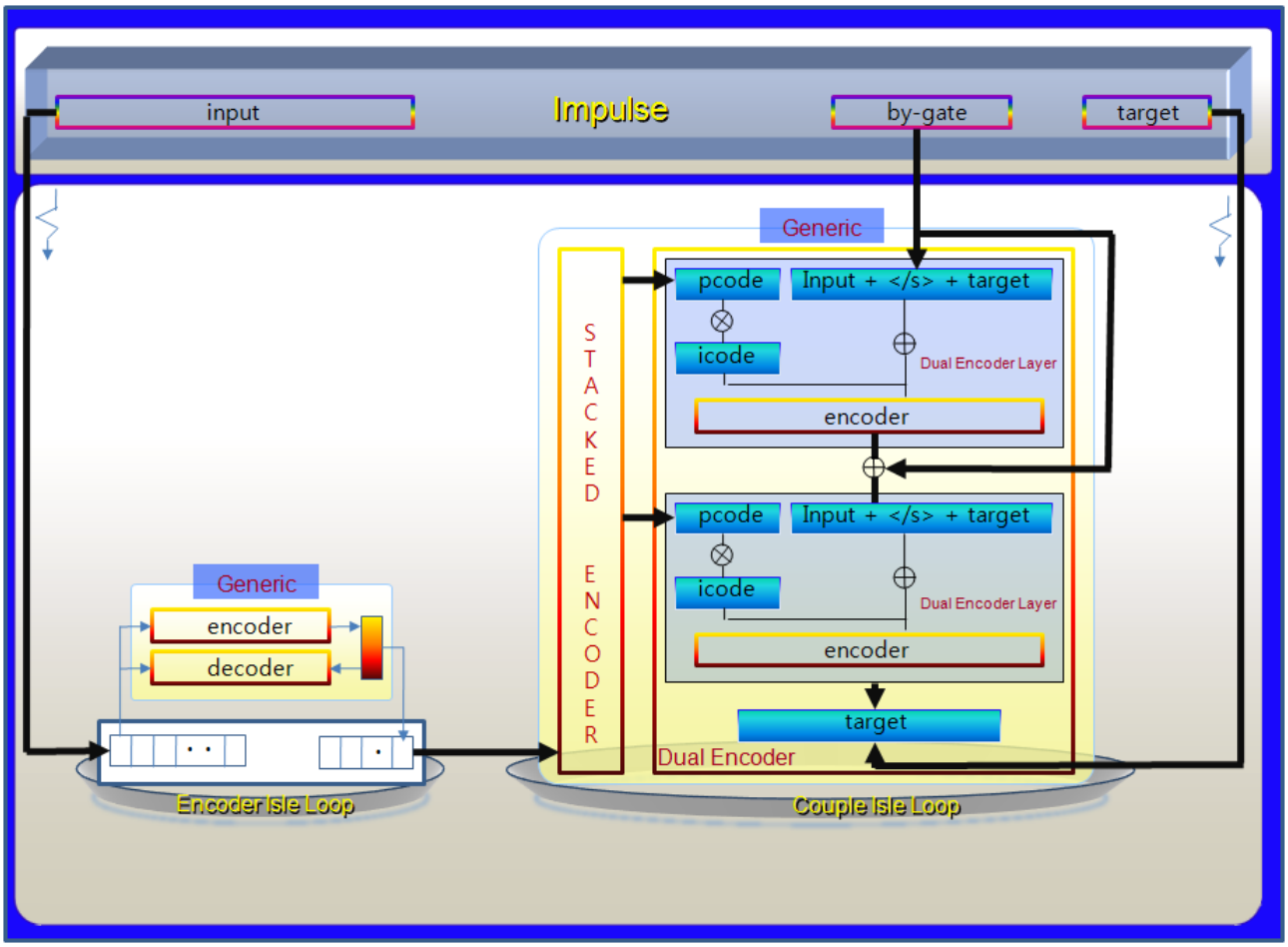
원본 데이터의 복원이나 합성에 관한 것이다. Link 오퍼레이션은 다이나젠 루프의 출력을 연결하여 Projection (투사)을 수행하고 데이터 동기화를 수행하는 impulse와 함께 단위 신경망인 제너릭을 연결하여 거대 신경망 시스템의 구성을 지원한다.

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.



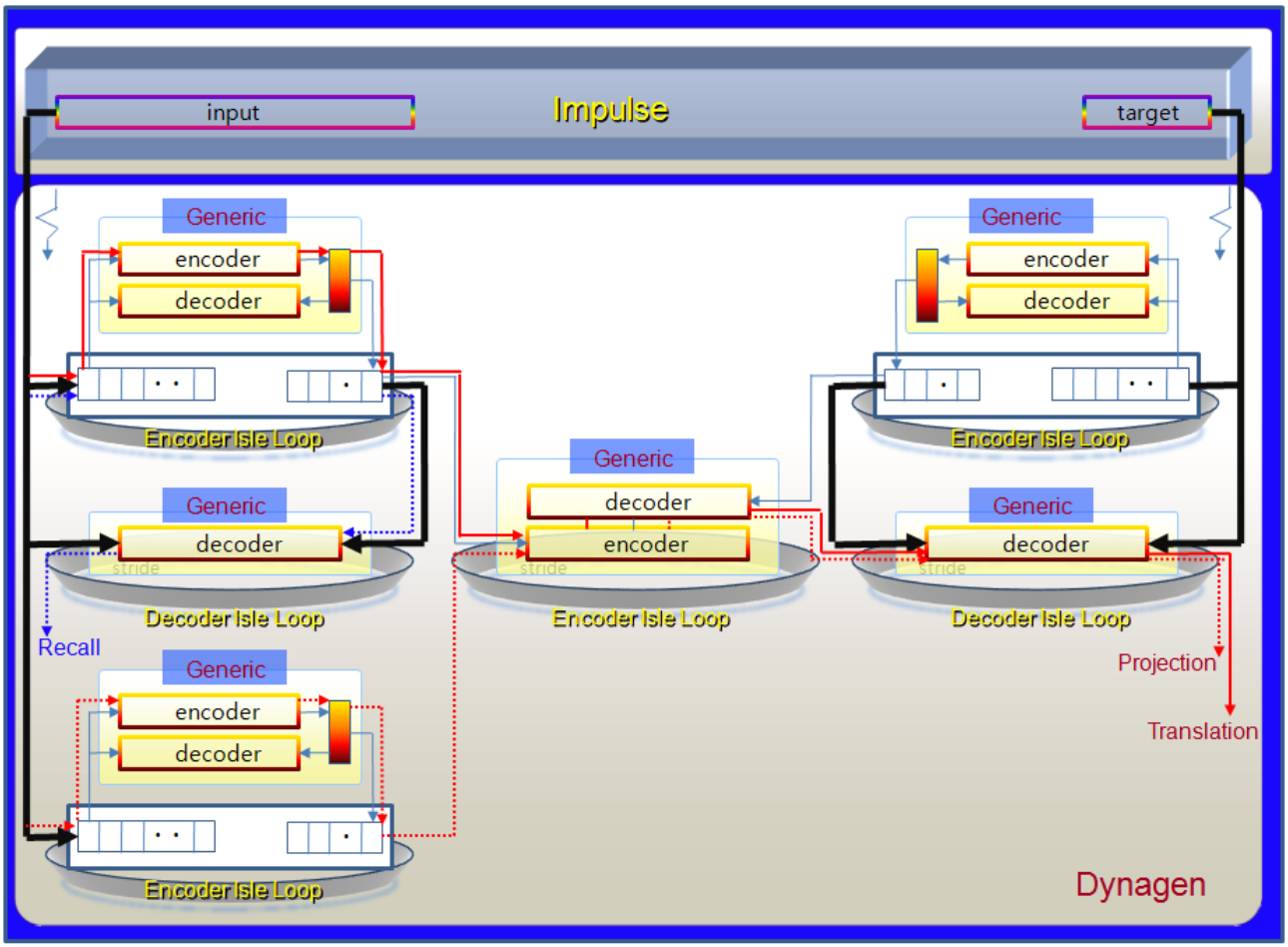
Dynagen Classification Operation & Generic Auto Regression

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

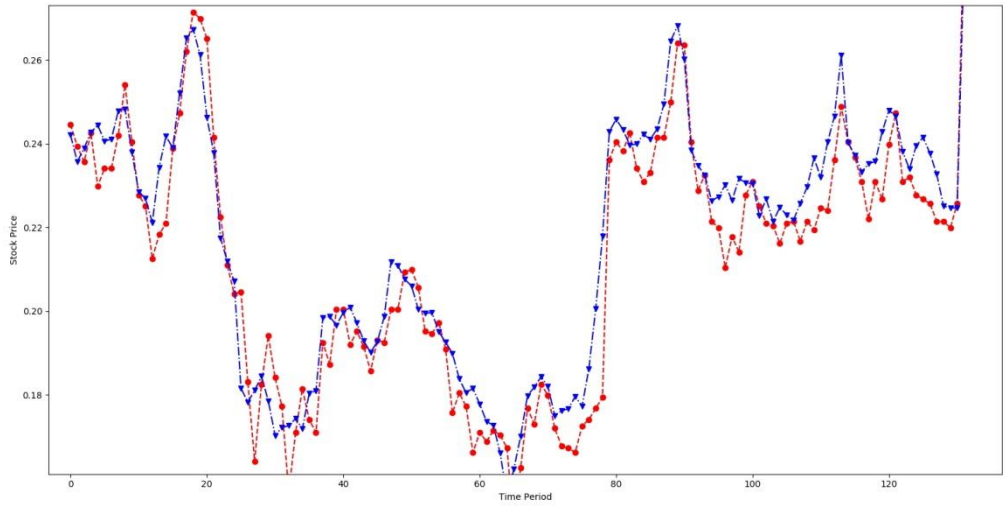
앞쪽의 정보가 소실되는 문제을 안고 있다. 이를 개선하는 방법으로 요즘 시계열 처리의 대세인 Transfomer계열이 있는데 셀프 어텐션 기법으로 인하여 기울기 소실 문제를 극복하고 어느 위치에서나 어텐션 할 수 있음을 장점으로 내세우지만 계산량이 시퀀스 길이에 비례하여 exponential하게 증가하여 시퀀스 길이에 심각한 제약이 있다. xlnet같은 경우는 이전 문맥 코드를 두기는 하나 매우 shallow하다. 따라서 여전히 long sequence 처리 문제에서 자유롭지 못하다. 또한 여러 실험을 해본 결과 패턴 생성이 RNN계열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

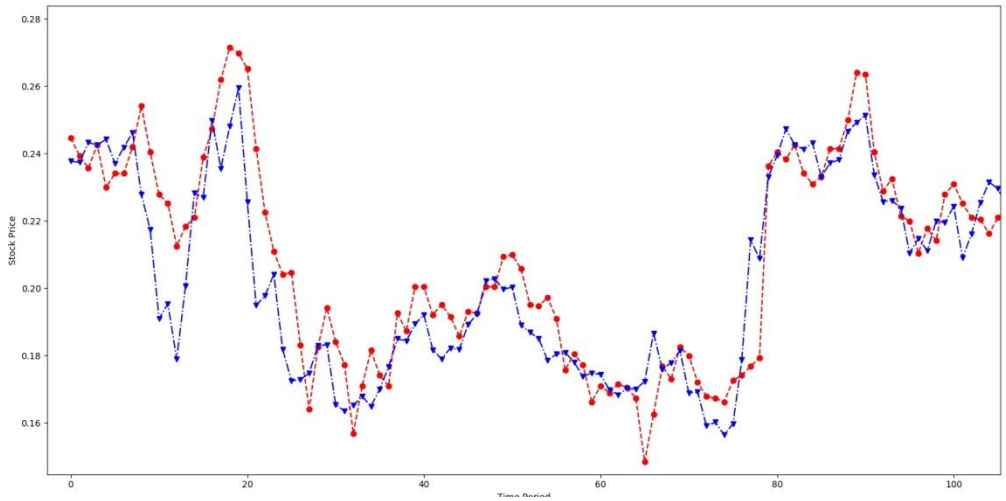
부가하여 이미지 인식 분야 성능을 나타내는 컨볼루션 망이 있으나 특징을 추출하기 위하여 컨볼루션 연산을 수행하는 과정에서 연산 자체와 풀링 레이어에 의하여 정보의 파괴가 심하고 생성 모델을 수행 할 수 없는 것이 단점이다.
이러한 점을 극복하고자 트랜스포머 모델을 이미지 인식에 적용하는 시도가 있으나 결과는 지켜봐야 할 일이다.

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

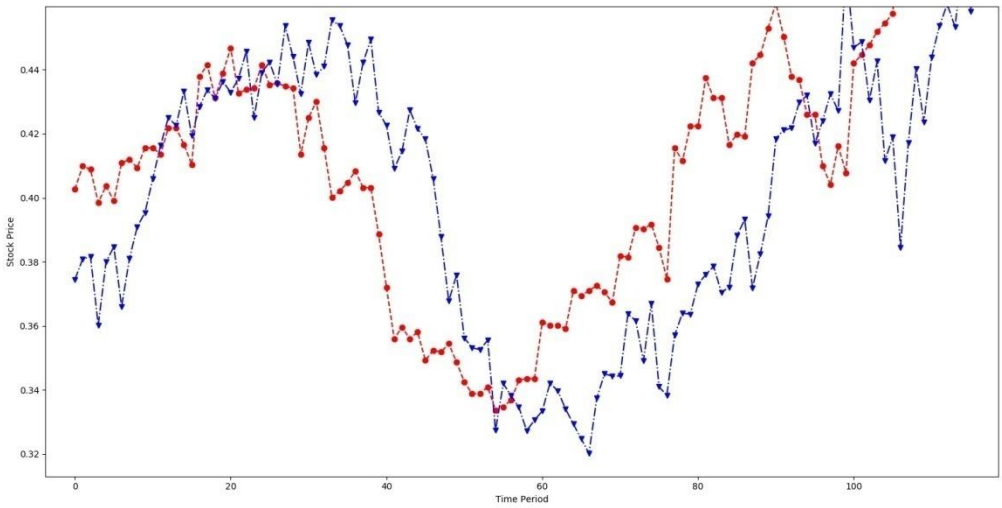
다음은 강력한 패턴 탐지의 근거로서 주가 예측 그래프를 보여준다.
학습은 Dynagen과 Generic망을 함께 사용하여 입력 값을 pcode로 하였다.





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;
```