MEMPUTE

- AI 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 list 리턴

'

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	Generic	
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	프레임웍크 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 vocabulary 갯수(선형데이터 이면 0) ⑤ 출력 vocabulary 갯수(선형데이터 이면 0) ⑥ 워드 임베딩 차원값(선형데이터 이면 0) ⑦ 활성함수 코드값 ⑧ 학습률 ⑨ 네임 스코프 이름

출력매개변수		
반환값	Generic *	망 오브젝트
method	train predict accuracy messureAccucracy	학습 추론 정확도 측정 그래프 정촥도 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 그래프 정촥도 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 vocabulary 갯수(선형데이터 이면 0) ⑤ 출력 vocabulary 갯수(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 vocabulary 갯수(선형데이터 이면 0) ⑤ 출력 vocabulary 갯수(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 그래프 정착도 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();
```

```cpp
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

```c
#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 %f\n", *(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;
```