

# HNet 2.0 User Manual

Latest update: 2019-01-19

Heejo You

1. Brief description of HNet 2.0 .....	4
A. Install and Run .....	4
B. Workspace and Load .....	4
C. Process setup .....	5
D. Pattern setup .....	5
E. Learning setup .....	6
F. Test setup .....	6
G. Train .....	7
2. Details of each feature.....	7
A. Process .....	7
i. Make new process.....	7
ii. Shortcut tabs.....	7
iii. Custom tab.....	7
iv. Process graph.....	8
v. Multi processes and reuse.....	8
B. Pattern .....	9
C. Learning .....	9
i. Make new learning.....	9
ii. Matching.....	9
iii. Learning flow and multi matching .....	9
D. Test .....	10
i. Name and mini batch.....	10
ii. Get tensor.....	10
iii. Attached pattern column.....	10
iv. Matching.....	10
E. Train .....	10
i. Train information.....	11
ii. Weight.....	11
F. Result file .....	11
3. Advanced use.....	11

A.	Prerequisite files and modify script .....	11
i.	Prerequisite.pickle.....	11
ii.	Prerequisite.txt.....	11
B.	Function adding .....	12
C.	Shortcut add .....	13
4.	Appendix.....	14
A.	Function description .....	14
i.	Placeholder.....	14
ii.	Forward.....	14
iii.	RNN.....	15
iv.	Activation Calc.....	17
v.	Reshape.....	18
vi.	Fundamental math.....	19
vii.	Loss calc.....	20
viii.	Optimizer.....	22
ix.	Test Calc.....	22
x.	Create.....	24
xi.	Other .....	24

# 1. Brief description of HNet 2.0

## A. Install and Run

HNet 2.0은 Python 3.6.4 버전에서 제작 및 테스트되었으며, 2.x에서의 작동여부는 확인되지 않았습니다. 또한 HNet 2.0은 Tensorflow에 기반한 GPU 가속을 지원하므로, GPU 가속이 필요하시다면 CUDA 등의 설치를 통해 사전에 환경을 구성하여 주십시오.

프로그램의 시작에 앞서 다음 명령어를 terminal(command line)에 입력하여 modules의 버전업을 완료해 주십시오.

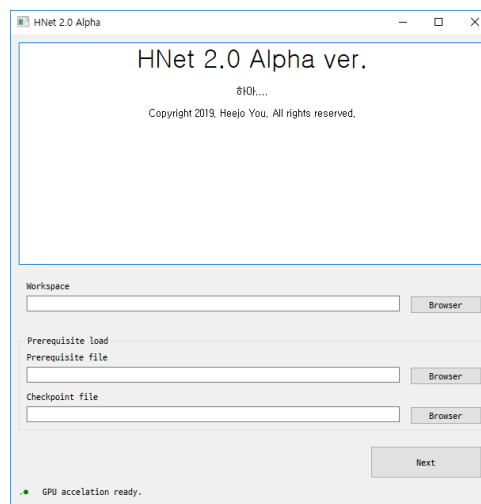
```
pip install -r requirements.txt
```

HNet 2.0 구동을 위한 환경설정이 완료되었다면 다음의 명령어를 terminal에 입력하여 HNet 2.0을 실행해 주십시오.

```
Python hnet2_gui.py
```

HNet 2.0은 Main 화면을 포함하여 총 5단계의 설정 이후 모델의 학습 및 테스트가 진행됩니다.

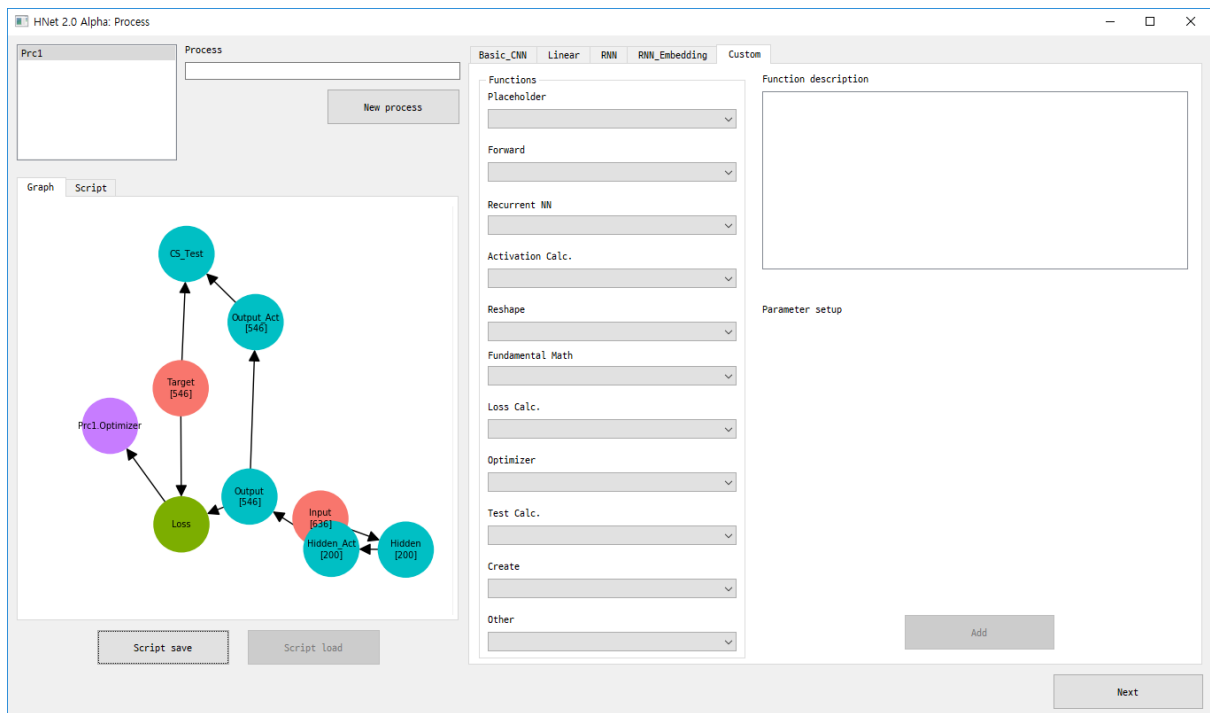
## B. Workspace and Load



workspace는 모델의 모든 정보가 저장되는 폴더(디렉터리)입니다. HNet 2.0은 workspace에 모델의 구조정보, 학습된 weight 정보, 그리고 테스트 정보가 기록되게 됩니다.

Prerequisite load는 이전에 구성한 모델을 다시 불러오는 기능입니다. 이중 Prerequisite file은 모델의 설정 정보에 관한 파일입니다. '.pickle'과 '.txt'로 이뤄진 파일을 불러올 수 있으며, 이 파일은 5단계의 세팅이 모두 끝났을 때 workspace에 생성됩니다. 만약 prerequisite file을 사용하여 모델을 불러올 경우 나머지 단계를 모두 건너뛰고 바로 train과정으로 들어갑니다. 한편 checkpoint file 이전에 학습했던 모델의 weight에 관한 파일입니다. 즉 prerequisite file로 불러온 모델 내의 weight들을 이전에 학습했던 상태로 변경해줍니다. 만약 prerequisite file만 사용한다면 처음부터 학습을 시작합니다.

### C. Process setup



Process는 HNet 2.0에서 하나의 activation flow를 나타냅니다. 사용자는 우상단의 shortcut 탭들을 이용해서 미리 정해놓은 구조의 process를 불러오거나, custom tab을 활용해서 자유롭게 process를 구성할 수 있습니다. Process의 기본적인 구성은 다음과 같습니다. 1) 시작값과 목표값을 입력해주는 placeholder (그래프의 red). 2) 연산 과정을 표현하는 tensor (그래프의 blue green). 3) output tensor와 target placeholder을 비교하여, 모델이 예측이 틀린 정도를 표현해주는 loss (graph의 green). 4) loss를 줄이기 위해 각 weight를 수정하는 optimizer(그래프의 purple).

Custom tap은 TensorFlow나 딥러닝에서 주로 사용하는 function들을 process 구성에 사용할 수 있도록 지원합니다. 각 function은 크게 11종류로 분류되어있습니다. function을 선택하면 우상단에 해당 function에 대해 간략한 description이 표시됩니다. 사용자는 각 function의 세부 파라미터를 결정하고 process에 추가할 수 있습니다.

#### D. Pattern setup

HNet 2.0 Alpha: Pattern

Pattern file

Name

Add

VOISer

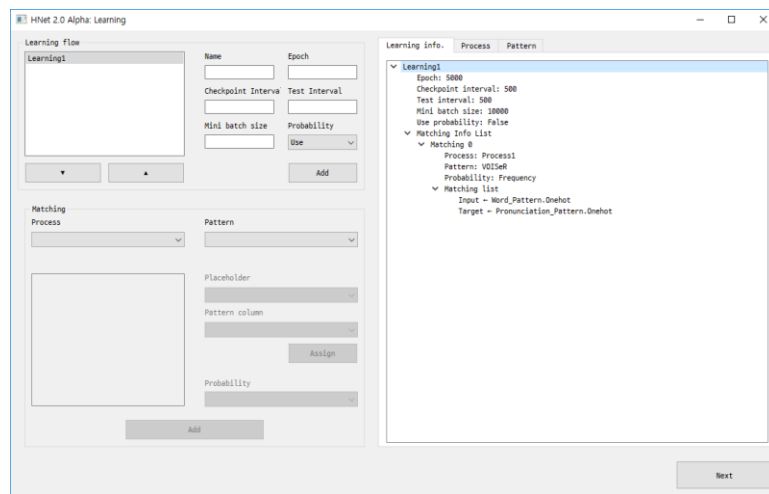
	Frequency	Pronunciation	Word	rd_Pattern_Oneh	rd_Pattern_Inde	clation_Pattern
0	0.9132075471698113	e	a	[0. 0. 0. 0. 0.	[26 52 52 52 52	[0. 0. 0. 0. 0.
1	0.5882075471698113	Eɹŋ	Aaron	[0. 0. 0. 0. 0.	52 52 52 52 52.	[0. 0. 0. 0. 0.
2	0.4311320754716981	@əʊ	aback	[1. 0. 0. 0. 0.	[ 0. 36 43 48 39	[0. 0. 0. 0. 0.
3	0.4443396226415094	əb@əs	abacus	[0. 0. 0. 0. 0.	52 52 52 52 52.	[0. 0. 1. 0. 0.
4	0.5382075471698113	@ənt@n	abandon	[0. 0. 0. 0. 0.	[26 27 26 28 36	[0. 0. 0. 1. 0.
5	0.5533918867924528	@ənt@nd	abandoned	[0. 0. 0. 0. 0.	52 52 52 52 52.	[0. 0. 0. 0. 0.
6	0.4759439622642515	@ənt@ŋ	abandoning	[0. 0. 0. 0. 0.	[26 27 26 39 29	[0. 0. 0. 1. 0.
7	0.46462641509434	@ənt@nmənt	abandonment	[0. 0. 0. 0. 0.	40 39 52 52 52.	[0. 0. 0. 0. 0.
8	0.310377358498566	@bes	abase	[0. 0. 0. 0. 0.	[26 27 26 39 29	[0. 0. 0. 1. 0.
9	0.2491811320754717	@esmənt	abasement	[0. 0. 0. 0. 0.	40 39 38 38 39.	[0. 0. 0. 0. 0.
10	0.15	@əʃ	abash	[0. 0. 0. 0. 0.	[26 27 26 44 38	[0. 0. 0. 1. 0.
11	0.415566873735849	@æt	abate	[0. 0. 0. 0. 0.	52 52 52 52 52.	[0. 0. 0. 0. 0.
12	0.38443396226415094	@et@d	abated	[0. 0. 0. 0. 0.	[26 27 26 45 38	[0. 0. 0. 1. 0.
13	0.31509433962264155	@əbs	abbess	[0. 0. 0. 0. 0.	52 52 52 52 52.	[0. 0. 0. 0. 0.
14	0.514622641509434	@bi	abbey	[0. 0. 0. 0. 0.	[26 27 27 39 44	[0. 0. 0. 0. 0.
15	0.4528301886792453	@əbt	abbot	[0. 0. 0. 0. 0.	[26 27 27 39 50	[0. 0. 0. 0. 0.
<				[0. 0. 0. 0. 0.	52 52 52 52 52.	[0. 0. 0. 0. 0.
>				[0. 0. 0. 0. 0.	52 52 52 52 52.	[0. 0. 0. 0. 0.]

The preview only displays up to the 100th pattern.

Next

HNet 2.0은 pickle된 Pandas의 dataframe을 패턴으로 입력받습니다. 패턴 파일과 해당 패턴의 모델 내에서의 name을 설정하고 입력하면 하단에서 입력된 패턴의 간략한 정보를 확인할 수 있습니다.

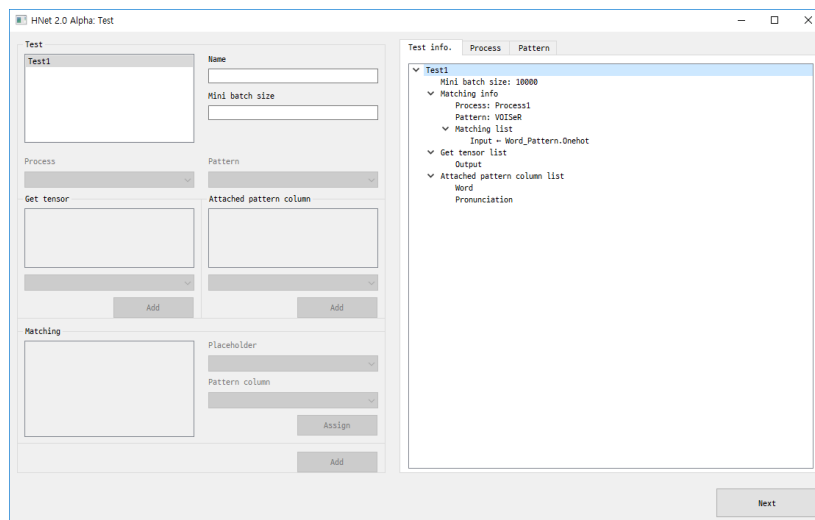
### E. Learning setup



process가 학습의 구체적인 방법을, pattern이 무엇을 학습할 것인지에 대한 설정이라면, learning은 이 둘을 매칭하고, 얼마나 학습할 것인지를 설정합니다. 사용자는 learning에서 몇 epoch 동안 학습이 이뤄지는지, 또한 학습 중간에 중간저장과 테스트를 얼마나 자주할 것인지를 결정하게 됩니다. 또한 process에서 입력된 어떤 pattern column이 해당 placeholder에 입력될 것인지도 learning에서 matching하게 됩니다.

입력된 learning information은 우측 패널에서 확인이 가능하며, 같은 위치의 탭을 이용하여 process와 pattern을 다시 확인할 수 있습니다.

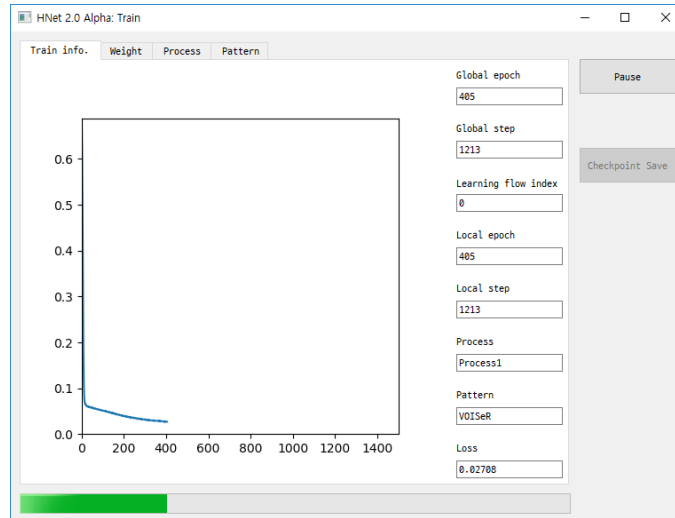
### F. Test setup



Test는 learning의 도중 혹은 종료시점에 모델로부터 어떤 정보를 추출할 것인지를 결정합니다. 필요에 따라서 learning과 동일하거나 다른 process를 사용할 수 있으며, process내의 대부분의 tensor를 추출할 수 있습니다.

입력된 test information은 Learning과 동일하게 우측 패널에서 확인이 가능하며, 같은 위치의 탭을 이용하여 process와 pattern을 다시 확인할 수 있습니다.

## G. Train



전체 설정이 종료되면 train 화면이 제시되고 학습을 진행하게 됩니다. 학습 중 HNet 2.0은 현재의 학습 진행 상황, 간단한 loss의 변동을 그래프로 제시하며, 사용자는 학습 상황을 지속적으로 확인할 수 있습니다. 그리고 상단의 탭을 이용하여 모델의 현재의 weight들의 형태를 확인할 수 있습니다.

## 2. Details of each feature

### A. Process

#### i. Make new process

좌상단의 textbox에 작성할 process의 이름을 입력한 후 add 버튼을 누르면 tensor가 없는 빈 process가 생성됩니다. Backend로 작동하는 TensorFlow의 설정상 추가된 process와 tensor는 삭제할 수 없으므로 process 변경이 필요하다면 프로그램을 종료한 후 재시작해야 합니다.

#### ii. Shortcut tabs

Shortcut tabs은 모델 구성에 필요한 몇 가지 변수를 입력하여 사전에 구성되어 있는 process를 구성하는 기능입니다. Process가 생성되면 상단의 탭들이 활성화됩니다. 필요한 process 구조를 선택하여 각 parameter를 입력하고 'Add' button을 누르면 지정된 process가 구축됩니다.

#### iii. Custom tab

만약 shortcut에 있는 process를 이용하지 않고 직접 구성하고자 한다면, custom tab을 이용할 수 있습니다. Custom tab은 Process가 생성되면 가운데의 우측의 function에 관련된 기능이 활성화됩니다. 사용자는 필요한 placeholder, tensor, loss, optimizer를 추가하여 process를 형성할 수 있습니다.

### Placeholder

Placeholder는 사용자가 지정한 데이터를 입력받는 영역으로서, 정확히는 tensor의 한 특수한 종류입니다. 사용자는 차후에 어떤 데이터를 입력 받을지 고려하여 placeholder의 크기를 결정하고 추가해야 합니다. 예를 들어 학습에 사용될 입력 패턴과 출력 패턴이 각각 size 200, 100의 vector라면, size 200과 100의 placeholder로 설정하여 process에 추가해야 합니다. 복수의 placeholder가 하나의 process에 추가될 수 있습니다.

Process에 placeholder를 추가하기 위해서는 'Placeholder' combobox에서 선택 후에, 세부 parameter를 설정하고 파라미터 영역의 하단에 있는 add 버튼을 누르면 됩니다. 각 parameter의 구체적인 사항은 4.A.i에서 확인하십시오.

## Tensor

Tensor의 정확한 정의는 어떤 연산이 실행된 결과물입니다. 이미 존재하는 placeholder나 tensor에 어떤 function을 사용하여 연산을 실시하면 그 결과물로서 tensor가 생성됩니다. 따라서, 대부분의 tensor는 단독으로 형성될 수 없고 최초의 tensor는 placeholder에 어떤 function을 적용하여 생성하게 됩니다.

비록 대부분의 function은 단일한 tensor를 입력받아 단일한 새 tensor를 생성하지만, 항상 그런 것은 아닙니다. 'concat'과 같이 입력에 필요한 tensor가 여러 개 일 수 있으며, 'split'과 같이 단일한 값을 받아 복수의 tensor를 생성하는 경우도 있습니다.

HNet 2.0은 기본적으로 4.A의 목록에 있는 function을 지원하며, 사용자는 필요에 따라 직접 function을 추가할 수 있습니다. Function 추가에 대한 구체적인 사항은 3.B를 참조하십시오.

## Loss

Loss는 어떤 두 tensor의 vector를 비교하여 그 차이를 나타내는 값입니다. HNet 2.0의 모든 학습은 loss를 감소시키는 방향으로 진행됩니다. Loss를 계산하는데 이용되는 두 tensor는 같은 shape(같은 size)여야만 합니다. 또한 HNet 2.0에서 이 값은 tensor로서 이용되지 않으며, optimizer를 제외한 다른 function에 이용될 수 없습니다. HNet 2.0은 기본적으로 4가지의 loss function을 지원하며, 각 loss function의 사용에 관해서는 4.A.vii를 참조하십시오.

## Optimizer

HNet 2.0에서 Optimizer는 현재까지 process에 추가된 모든 loss를 감소시키도록 유도하는 연산자입니다. 모델의 학습을 위해서 process에 optimizer가 반드시 필요하며, optimizer가 없는 process는 오직 test에만 이용될 수 있습니다. 또한 optimizer는 하나의 process에 하나만 추가될 수 있고, optimizer가 추가된 이후엔 loss를 추가할 수 없습니다. 따라서 필수는 아니지만 optimizer는 process 구성의 마지막에 추가할 것을 권장합니다. HNet 2.0에서 지원하는 optimizer에 대한 구체적인 사항은 4.A.viii에서 확인하십시오.

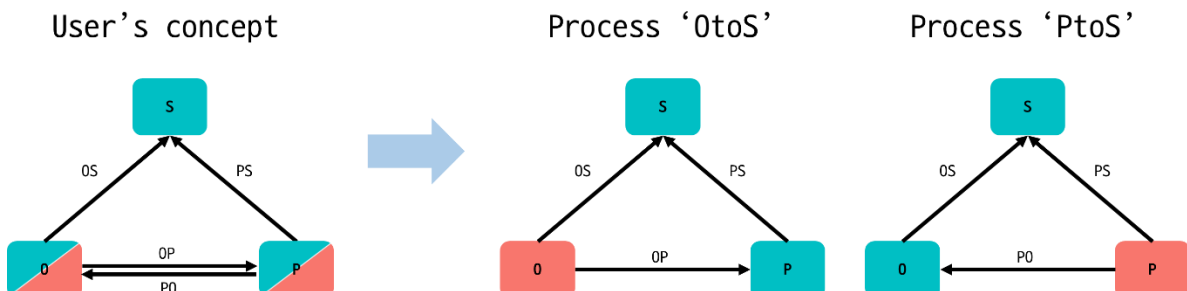
### iv. Process graph

Process graph는 화면의 좌하단에 위치합니다. Process의 각 구성요소들이 어떻게 연결되어 있는가를 네트워크로 나타내며, 각 node의 색은 구성요소가 어떤 범주에 들어가는지를 표현합니다. 1) placeholder: red. 2) tensor: blue. 3) loss: green. 4) optimizer: purple.

또한 placeholder와 tensor node에는 해당 node의 size를 표시해주는 숫자가 적혀있으므로, tensor를 추가할 때 이를 참고할 수 있습니다.

### v. Multi processes and reuse

HNet 2.0은 multi processes를 지원합니다. 각 process는 기본적으로 서로 독립적이며, 1회의 학습 또한 단일한 process 단위로 이뤄집니다. 하지만 사용자 구상에 따라서 하나의 모델이 서로 다른 process로 여러 번 학습이 필요할 경우가 있습니다. 다음의 figure는 간략화된 하나의 예입니다.



컨셉 모델은 S가 output이지만, 0가 입력인 경우와 P가 입력인 경우가 있으며, 어느 하나가 선택되면 나머지 하나는 hidden으로 기능하는 구조입니다. 만약 process '0toS'와 'PtoS'가 완전히 독립적이라면, 하나의 프로세스



를 구성하여 학습하고 종료 후 다른 하나를 구성하여 학습시키면 됩니다. 하지만 만약 사용자가 두 process의 weights 'OS'와 'PS'를 동일하게 할 필요가 있다면, 두 process를 모두 설정하고 tensor를 입력할 때 reuse를 사용할 수 있습니다. 일부 'dense'나 'conv1d'와 같은 function은 parameter로서 reuse를 갖고 있습니다. 사용자는 reuse를 사용함으로써 여러 process들을 서로 연관성을 갖도록 할 수 있고, 커다란 모델의 부분으로 다룰 수 있습니다.

## B. Pattern

HNet 2.0은 pandas의 dataframe을 pattern으로 사용합니다. 사용자별로 필요로 하는 pattern의 형태가 다를 수 있기 때문에, HNet 2.0은 자체적으로 pattern을 구성하는 툴을 지원하지는 않습니다. 하지만 패턴 구성에 있어서 몇 가지 규칙이 있습니다. 첫째로 placeholder에 입력되는 column은 numpy array로 구성되어야 합니다. 사용자의 필요에 따라서 모델의 학습에 사용되지 않는 정보 또한 column으로 존재할 수 있습니다. 하지만 해당 정보는 HNet 2.0의 학습에 사용될 수 없습니다. 둘째로 column내의 패턴은 같은 타입의 동일한 shape이어야 합니다. HNet 2.0에서 하나의 column은 동일한 placeholder에 들어가기 때문에, column내 값들 간에 일관성이 유지되지 않을 경우 에러가 발생합니다. 셋째로 probability로 사용될 column은 0과 1사이의 scalar 값이어야 합니다. Probability에 대해서는 learning에서 세부적으로 언급하겠습니다. 넷째로, 하나의 process에서 사용할 columns은 반드시 하나의 pattern안에 모두 들어있어야 합니다. 복수의 process가 동일한 pattern을 사용하는 것은 가능하지만, 하나의 process가 복수의 pattern을 사용하는 것은 허용되지 않습니다.

## C. Learning

### i. Make new learning

좌상단에서 learning에 대한 6개의 parameter를 설정하고 add 버튼을 누르면 새로운 learning이 추가됩니다. 6개 파라미터의 의미는 다음과 같습니다. 1) name: 설정하는 learning의 식별자. 이 값은 사용자의 편의를 위해 사용되며 학습에 영향을 주지 않습니다. 2) epoch: 해당 learning을 얼마나 학습할 것인지 결정합니다. 1 epoch은 입력되는 패턴 전체가 한 번 학습되었음을 의미합니다. 3) checkpoint interval: 얼마나 자주 모델의 weight를 중간저장 할지 결정합니다. 4) test interval: 얼마나 자주 지정된 test를 수행할지 결정합니다. 5) mini batch size: 한 번 학습이 시도될 때 얼마나 많은 패턴이 학습될지 결정합니다. Mini batch size가 클수록 학습 속도가 빠르지만, 사용자의 메모리 환경보다 너무 큰 mini batch size를 사용할 경우 out of memory 문제를 발생시킬 수 있습니다. 6) probability: 입력되는 패턴의 학습가능성을 적용할 것인지 결정합니다. Use로 설정할 경우 매 epoch마다 각 항목의 probability 값과의 비교를 시행하여 해당 패턴의 학습 여부를 판단하게 됩니다. No use의 경우 매 epoch에서 모든 패턴이 학습됩니다.

입력된 learning의 세부 정보는 우측의 패널에서 확인할 수 있습니다. 그리고 만약 입력된 learning이 잘못되었을 경우 좌측의 listbox에서 해당 learning을 double click하면 삭제할 수 있습니다.

### ii. Matching

모든 learning은 반드시 matching이 설정되어야 합니다. 해당 learning에서 이용할 process와 pattern를 하나씩 선택하면 하단의 matching 패널이 활성화됩니다. placeholder와 pattern column을 각각 선택하고 assign 버튼을 누르면 해당 정보가 입력되며, 사용자는 모든 placeholder에 특정 column을 할당해야 합니다. 그리고 만약 learning이 probability를 사용한다면, 사용할 probability column 또한 설정하십시오. Add 버튼을 누르면 learning에 설정한 matching이 할당됩니다.

### iii. Learning flow and multi matching

#### Learning flow

HNet 2.0은 learning flow를 지원합니다. 예를 들어 사용자가 성인이 되었을 때 second language를 배우는 late bilingual에 대한 모델을 구성하고자 한다고 생각해 봅시다. 이를 위해서는 환경의 변화가 고려되어야 합니다. 초기의 모델은 오직 mother language에 관한 정보만을 학습합니다. 하지만 일정 epoch이 지나면 모델은 mother

language와 second language에 관한 정보를 모두 학습하는 형태가 됩니다. 사용자는 복수의 learning을 설정할 수 있으며, 이들의 순서를 설정함으로써 환경이 변화하였을 경우의 모델을 구성할 수 있습니다. 좌측 learning 패널의 '▲'와 '▼' 버튼을 통해 learning의 순서를 변경할 수 있습니다.

#### Multi matching

Learning flow와는 별개로, HNet 2.0은 Multi matching을 지원합니다. A에서 언급한 multi process에 대한 figure를 다시 생각해봅시다. 이 두 process는 어느 쪽이 우선적으로 학습될 수 없으며, 하나의 learning 안에서 두 process에 대한 학습 모두 이루어져야 합니다. 이를 위해 사용자는 하나의 learning 안에 복수의 matching을 설정할 수 있습니다. Multi matching이 설정되었을 때, matching의 학습 순서는 mini batch 단위로 random이 되고, 모든 mini batch의 학습이 1 epoch이 됩니다. 예를 들어, matching 1에 대한 mini batch인 batch1-1, batch1-2, batch1-3과 matching 2에 대한 mini batch인 batch2-1, batch2-2가 있다면, 이들 5개의 미니배치의 학습 순서는 random이고, 5개 mini batch가 전부 학습되었을 때 epoch 1 증가합니다.

#### D. Test

learning에서 설정된 test interval마다 설정된 test가 실시되고 결과가 기록됩니다. Learning과는 다르게 test는 matching까지 포함해서 하나의 test가 됩니다. 사용자는 여러 개의 test를 설정할 수 있으며, 여러 test를 설정했을 경우, 매 test interval마다 모든 test가 실시됩니다.

##### i. Name and mini batch

Test에서 name은 테스트 결과에 영향을 주지 않으나 차후 생성될 결과파일의 식별자가 됩니다. Mini batch size는 한 번에 얼마나 많은 패턴이 테스트되는가를 결정합니다. mini batch size가 클수록 테스트 시간이 감소하지만, 너무 큰 mini batch size를 사용할 경우 out of memory 문제를 발생시킬 수 있습니다. Mini batch size는 테스트 결과에는 영향을 주지 않습니다.

Name과 mini batch를 입력한 후 test에서 이용할 process와 pattern를 하나씩 선택하면 하단의 matching 패널들이 활성화됩니다.

##### ii. Get tensor

테스트 중 활성화 값을 추출할 tensor들을 선택합니다. Loss와 optimizer를 제외한 모든 tensor가 선택 가능하며, 적어도 1개 이상 tensor가 선택되어야 합니다.

##### iii. Attached pattern column

Pattern 중에서 테스트 결과 파일에 추가적으로 부착될 column들을 선택합니다. Pattern 내의 모든 column이 선택 가능합니다.

##### iv. Matching

모든 test는 반드시 matching이 설정되어야 합니다. placeholder와 pattern column을 각각 선택하고 assign 버튼을 누르면 해당 정보가 입력되며, 사용자는 'Get tensor'에서 선택한 tensor값을 추출하기 위해 필요한 모든 placeholder에 특정 column을 할당해야 합니다. 이에 따라 일부 placeholder는 값이 할당될 필요가 없으며, 불필요한 할당이 있을 경우 메시지로 알린 후 제외됩니다.

#### E. Train

모든 설정이 종료되면 train 화면으로 넘어가며, 사용자가 start 버튼을 누르면 학습이 시작됩니다. 학습의 진행상황은 화면 하단에 있는 progress bar에서 확인할 수 있습니다. 중간에 일시정지가 필요하다면 pause 버튼(start 버튼과 동일함)을 누르십시오. 현재 epoch이 종료됨과 동시에 일시정지 합니다. 일시정지시 해당 시점의 checkpoint를 추가로 저장할 수 있습니다.

#### i. Train information

Train information 탭은 progress bar를 제외한 나머지 학습상황을 표시합니다. 화면에 표시되는 정보는 다음과 같습니다. 1) global epoch: 모든 learning flow에 대하여 얼마나 많은 epoch이 학습되었는가를 표시합니다. 2) global step: 모든 learning flow에 대하여 얼마나 많은 횟수의 학습이 진행되었는가를 표시합니다. 3) learning flow index: 전체 learning flow내에서, 현재 진행중인 learning이 몇 번째 learning인지 표시합니다. 시작값은 0입니다. 4) local epoch: 현재 learning에 대하여 얼마나 많은 epoch이 학습되었는가를 표시합니다. 5) local step: 현재 learning에 대하여 얼마나 많은 횟수의 학습이 진행되었는가를 표시합니다. 6) process: 현재 step이 사용한 process를 표시합니다. 7) pattern: 현재 step이 사용한 pattern을 표시합니다. 8) loss: 현재 step이 도출한 loss 값을 표시합니다. 9) graph: 해당 학습에서 출력된 loss합의 변화 흐름을 표시합니다. 하나의 epoch내의 여러 step의 학습이 실시되었을 경우 loss는 해당 epoch내의 모든 step에서 산출된 loss의 평균값입니다.

#### ii. Weight

Weight 탭은 현재 학습중인 모델이 사용하고 있는 weight들을 이미지화 하여 표시합니다. Convolution filter와 같은 3D (convolution 1d)나 4D (convolution 2d) weight의 경우 index slide를 통해 원하는 필터를 선택하여 볼 수 있습니다. Weight 이미지의 가로는 입력받는 tensor의 차원을, 세로는 출력되는 tensor의 차원을 나타냅니다.

#### F. Result file

학습 중간, 그리고 학습 종료시마다 HNet 2.0은 test의 결과 파일을 저장합니다. 저장 위치는 지정된 workspace 폴더 내의 'Result' 폴더이며, 파일명은

'GE\_{global\_epoch}.LI\_{learning\_flow\_index}.LE\_{local\_epoch}.pickle' ({ }는 해당 변수)

입니다. 파일은 pickled dict 형식으로 되어 있으며, python의 pickle library를 통해 내부를 확인할 수 있습니다.

Dict의 각 item 중 'global\_Epoch', 'global\_Step', 'learning\_Index', 'local\_Epoch' 'local\_Step'은 해당 테스트가 실시된 시점의 학습 진행 상황에 대한 정보를 나타냅니다. 그리고 이를 제외한 'Result'라는 하부 dict 내에 실제 test 결과가 들어있습니다. 'Result' dict은 실시된 각 test 명을 key로 보유합니다. 이들 각 item은 해당 test 결과가 기록된 pandas dataframe으로 되어 있으며, dataframe의 column은 test 설정시의 'get tensor'와 'attached pattern column'으로 구성되어 있습니다. 만약 test 설정시에 'get tensor'와 'attached pattern column' 사이에 겹치는 이름이 있었다면, 각각 '.Result'와 '.Pattern'의 태그가 column name에 추가로 붙습니다.

### 3. Advanced use

#### A. Prerequisite files and modify script

Test의 설정까지 종료되면, workspace폴더 밑에 model 폴더가 생기고, 'Prerequisite.pickle'과 'Prerequisite.txt'의 2개의 파일이 생성됩니다.

#### i. Prerequisite.pickle

'Prerequisite.pickle'은 workspace를 제외하고 모든 설정 정보를 보유하고 있습니다. 사용자는 차후에 이 파일을 로드 함으로써 동일한 모델의 학습을 진행하거나, 혹은 checkpoint를 같이 로드함으로써 이전에 중단됐던 학습을 이어서 진행할 수 있습니다. Python의 pickle library를 통해 내부 수정이 가능하지만, 권장하지 않습니다.

#### ii. Prerequisite.txt

'Prerequisite.txt' 동일한 설정 정보 파일입니다. 다만, pickle과는 달리 pattern 정보는 포함되어 있지 않으며, txt파일로 되어 있어 text editor를 통해 열어볼 수 있습니다. 내부는 HNet 2.0을 조작하는 파이썬 코드의

일부로 구성되어 있으며, 사용자는 필요에 따라 파일을 수정하여 다른 형태의 모델을 구성하거나 일부 파라미터를 변경하는 것이 가능합니다. 이 파일 또한 '.pickle'과 동일하게 로드될 수 있습니다. 하지만 파일을 수정하여 사용할 때 몇 가지 고려해야만 합니다. 1) pattern의 path를 설정하십시오. '#Pattern' comment 이하의 코드는 pattern을 입력 받는 부분입니다. 파라미터에 패턴 파일이 지정되어 있지 않으므로 사용자는 'pattern\_Path=' 파라미터에 패턴의 경로를 직접 추가해야 합니다. 해당 설정을 처음 작성할 때 사용하였던 패턴의 경로를 입력하십시오. 2) process를 수정하면 checkpoint를 사용할 수 없게 될 수도 있습니다. Process의 수정은 모델이 사용하는 weight의 일부 혹은 전체의 shape을 변경하게 될 가능성이 있고, 이 경우 checkpoint는 호환되지 않습니다. 3) 로드되고 train 화면이 제시되면 학습을 시작하기 전에 process, pattern, learning, 그리고 test 탭을 확인하십시오. 구조와 사용하는 패턴이 논리적으로 올바르지 않다면 대부분의 경우 로드 중 오류가 발생하고 넘어가지 않습니다. 하지만 일부 경우를 대비하여 직접 확인할 필요가 있습니다.

## B. Function adding

HNet 2.0은 대부분의 사용자가 주로 사용하는 function들을 기본적으로 포함하고 있습니다. 하지만 사용자에게 따라 자주 사용하는 function이 다르거나, 직접 customized function을 만들어서 사용하는 경우가 있습니다. 이를 위해 사용자는 필요에 따라 직접 Process 화면에서 사용 가능한 function을 추가할 수 있습니다.

Process 화면의 모든 function들은 'HNet2/GUI/Process/Process\_Func.py'의 'func\_Dict'에 dict 형태로 기록되어 있습니다. 신규 function을 추가하고자 하는 사용자는 function의 설정을 HNet 2.0의 규칙에 맞게 입력해야 합니다. 아래는 HNet에서 설정되어 있는 'Layer\_dense'의 코드입니다.

```
func_Dict["Layer_Dense"] = {
    "Func_Type": Func_Type.Forward,
    "Tensor_Type": Tensor_Type.Tensor,
    "Parameter": OrderedDict([
        ("name", Parameter_Type.Name),
        ("input_Tensor_Names", Parameter_Type.Tensor),
        ("tensor_Func", tf.layers.dense),
        ("units", Parameter_Type.Positive_int),
        ("use_bias", Parameter_Type.Bool),
        ("reuse_Tensor_Name", Parameter_Type.Process_and_Tensor_with_None)
    ]),
    "Required": ["name", "input_Tensor_Names", "tensor_Func", "units"],
    "Description": ""
}
```

코드를 살펴보면 각 function에 대해 총 5가지 정보가 입력되어야 함을 알 수 있습니다. 1) Func\_Type: 이는 function이 어느 카테고리에 포함되는지를 나타냅니다. 사용할 수 있는 function 카테고리에 관해서는 'HNet2/GUI/GUI\_Enum.py'에 있는 'Func\_Type' class를 참조하십시오. 2) Tensor\_Type: 이는 function이 어떤 tensor를 생성하는지 나타냅니다. Tensor type에 관해서는 'HNet2/HNet\_Enum.py'에 있는 'Tensor\_Type' class를 참조하십시오. 3) Parameter: 이는 function이 실제 처리될 때 어떤 parameter들을 입력받을지 설정합니다. tensor type 및 function에 따라 필요한 parameter가 다르므로, 이를 명확히 정의해야 합니다. Tensor type 별로 필수적인 parameter에 관해서 확인하기 위해서는 'HNet2/Manager/Process.py'를 참조하십시오. 또한 'name'은 str, units는 int와 같이 parameter 별로 서로 다른 type의 변수를 요구하므로, 이를 parameter type으로 미리 설정해야 합니다. Parameter type에 관해서는 'HNet2/GUI/GUI\_Enum.py'에 있는 'Parameter\_Type' class를 참조하십시오. 4) Required: parameter 중에서 반드시 입력되어야 하는 parameter를 설정합니다. Required로 설정된 파라미터는 할당되지 않으면 process 내에 입력되지 않도록 되어 있습니다. 5) Description: function을 선택하였을 때, process 창의 우상단에 나타나는 function에 대한 설명을 나타냅니다. 이 값은 function의 기능에 영향을 주지 않습니다.

만약 tensorflow에 포함되어있는 built-in function 외에 사용자가 직접 function을 작성 및 추가하고 싶은 경우, 'Process\_Func.py' 파일 내의 'Embedding' function과 'HNet2/Compatible\_Functions' 내에 있는 각 파일들

을 예제로서 참고하십시오.

추가적으로, HNet 2.0은 batch size의 변화를 고려하지 않고 있습니다. 만약 function이 batch size를 다루는 경우엔 HNet 2.0에서는 사용하지 않을 것을 권장합니다. 이 기능은 python과 tensorflow 사용에 익숙한 사용자를 대상으로 한 기능입니다.

### C. Shortcut add

'HNet2/GUI/Process/Shortcut/' 경로 안에 존재하는 '.py' 파일들은 shortcut을 구성하는 파일들입니다. 사용자는 이 경로에 새로운 script를 추가함으로써, shortcut을 추가할 수 있습니다. Shortcut script는 3가지 정보가 입력되어야 합니다. 1) description: 이는 shortcut의 간략한 설명을 담고 있습니다. 이 값은 shortcut의 수행에 영향을 주지 않습니다. 2) parameter\_Dict: 이는 어떤 정보를 GUI 환경 내에서 변수로 입력받을 지 결정하는 변수들이 들어있습니다. Dict 내의 key들은 이후 스크립트에서 동일한 key를 사용해 변수로 이용될 수 있습니다. 3) Run\_Shortcut: HNet 2.0의 process 입력을 처리하는 script가 들어있는 python method입니다. 사용자는 두 가지 방법으로 Run\_Shortcut에 사용할 수 있는 script를 얻을 수 있습니다. 하나는 사용자가 process를 구성하였을 때, process의 graph와 같은 위치의 탭에 있는 script save 기능을 이용하는 것입니다. Script save 버튼을 누르면 Workspace 폴더 밑의 model 폴더에 'Process\_Script.txt'가 생성되고, 이를 사용할 수 있습니다. 다른 하나는 모든 설정이 종료되었을 때 생성되는 'Prerequisite.txt'에서 process 부분을 사용할 수 있습니다. 이 두 파일에서 얻는 스크립트를 Run\_Shortcut에 넣고, 고정된 값을 parameter\_Dict에서 구성한 파라미터로 변경함으로써 shortcut을 구성할 수 있습니다. 구체적인 예시는 'HNet2/GUI/Process/Shortcut/' 경로 안에 있는 파일들을 참고하십시오.

## 4. Appendix

### A. Function description

설명에서 단일 문자 'B'는 batch를 의미합니다.

#### i. Placeholder

Placeholder function은 process 내에 placeholder를 추가합니다. Placeholder는 name, dtype, shape의 세 parameter를 입력받으며, name, dtype은 필수입니다.

- 1) Name: process 내에서의 구분자로서 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) Dtype: placeholder의 타입을 결정합니다. float32, int32, bool의 3가지 중 하나를 입력받을 수 있습니다.
- 3) shape: placeholder의 형태를 결정합니다. 이 parameter에는 단일한 숫자나 혹은 ','로 구분된 숫자의 목록을 넣을 수 있습니다. 사용자가 차후에 넣으려고 하는 패턴의 형태와 일치하도록 넣으시기 바랍니다. 아무 값도 할당되지 않을 경우 scalar가 입력될 것으로 판단합니다. \*부정형 타입은 현재로서는 자체적으로 처리되는 batch size 외엔 사용자의 설정으로 지원하지 않습니다.

#### ii. Forward

Forward는 주로 weight를 이용하여 tensor의 계산이 일어나는 function들의 모음입니다.

##### Layer dense

Layer dense는 입력된 1차원 vector와 weight를 행렬곱하여 지정된 크기의 1차원 vector tensor를 추가합니다. 만약 입력되는 tensor가 n차원일 경우 마지막 n 차원의 값들이 vector로서 처리되어 출력 또한 마지막 차원이 지정된 크기인 n차원 tensor가 됩니다 (e.g. [B,3,5,7] -dense-> [B,3,5,10]). Parameter 목록은 다음과 같습니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) units: 출력되는 vector의 크기를 결정합니다. 이 값은 필수입니다.
- 4) use\_bias: 연산에 항상 1이 입력되는 bias를 추가할 것인지 결정합니다.
- 5) reuse\_Tensor\_Name: 다른 layer dense에서 weight를 가져옵니다. 이 경우 해당 layer와 weight를 공유합니다.

##### Conv1d

입력된 3d 데이터에 filter를 적용하여 다른 3d 데이터로 convolution합니다.

- 1) name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Names: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B, step, channel]형태의 3D 여야만 합니다. 이 값은 필수입니다.
- 3) filters: 연산에 얼마나 많은 filter를 사용할지 결정합니다. 이 값은 출력되는 tensor의 channel 숫자와 동일합니다. 이 값은 필수입니다.
- 4) kernel\_size: 연산에 사용되는 filter의 크기를 결정합니다. 이 값은 필수입니다.
- 5) strides: convolution시의 filter 이동의 크기를 결정합니다.
- 6) padding: padding을 통해 convolution 후의 step 크기가 입력된 tensor와 동일하게 할 것인지 결정합니다.

7) use\_bias: 연산에 항상 1이 입력되는 bias를 추가할 것인지 결정합니다.

5) reuse\_Tensor\_Name: 다른 conv1d에서 weight를 가져옵니다. 이 경우 해당 layer와 weight를 공유합니다.

### Conv2d

입력된 4d [B, height, width, channel] 데이터에 filter를 적용하여 다른 4d 데이터로 convolution합니다.

1) name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Names: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B, height, width, channel]형태의 4D 여야만 합니다. 이 값은 필수입니다.

3) filters: 연산에 얼마나 많은 filter를 사용할지 결정합니다. 이 값은 출력되는 tensor의 channel 숫자와 동일합니다. 이 값은 필수입니다.

4) kernel\_size: 연산에 사용되는 filter의 크기를 결정합니다. 이 값은 필수입니다.

5) strides: convolution시의 filter 이동의 크기를 결정합니다.

6) padding: padding을 통해 convolution 후의 step 크기가 입력된 tensor와 동일하게 할 것인지 결정합니다.

7) use\_bias: 연산에 항상 1이 입력되는 bias를 추가할 것인지 결정합니다.

5) reuse\_Tensor\_Name: 다른 conv2d에서 weight를 가져옵니다. 이 경우 해당 layer와 weight를 공유합니다.

### Conv2d\_Transpose

입력된 4d [B, height, width, channel] 데이터에 filter를 적용하여 다른 3d 데이터로 transposed convolution을 실시합니다.

1) name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Names: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B, height, width, channel]형태의 3D 여야만 합니다. 이 값은 필수입니다.

3) filters: 연산에 얼마나 많은 filter를 사용할지 결정합니다. 이 값은 출력되는 tensor의 channel 숫자와 동일합니다. 이 값은 필수입니다.

4) kernel\_size: 연산에 사용되는 filter의 크기를 결정합니다. 이 값은 필수입니다.

5) strides: convolution시의 filter 이동의 크기를 결정합니다.

6) padding: padding을 통해 convolution 후의 step 크기가 입력된 tensor와 동일하게 할 것인지 결정합니다.

7) use\_bias: 연산에 항상 1이 입력되는 bias를 추가할 것인지 결정합니다.

5) reuse\_Tensor\_Name: 다른 conv2d transpose에서 weight를 가져옵니다. 이 경우 해당 layer와 weight를 공유합니다.

### iii. RNN

Recurrent neural network(RNN)은 시간 정보를 고려한 tensor의 계산이 일어나는 function들의 모음입니다.

#### RNN LSTM

LSTM cell을 사용한 RNN 연산을 실시합니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B,

time, vector]의 형태의 3D여야만 합니다. 이 값은 필수입니다.

3) state\_Reset: 만약 True라면 학습의 시작시 초기 state가 이전 학습의 마지막 state가 됩니다. False일 경우 모든 값이 0인 state가 사용됩니다. Test에는 이전학습의 마지막 state가 적용되지 않습니다 (항상 0 state).

4) num\_units: 출력되는 vector의 크기를 결정합니다. 이 값은 필수입니다.

5) reuse\_Tensor\_Name: 다른 RNN\_LSTM에서 weight를 가져옵니다. 이 경우 해당 layer와 weight를 공유합니다.

#### RNN GRU

GRU cell을 사용한 RNN 연산을 실시합니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B, time, vector]의 형태의 3D여야만 합니다. 이 값은 필수입니다.

3) state\_Reset: 만약 True라면 학습의 시작시 초기 state가 이전 학습의 마지막 state가 됩니다. False일 경우 모든 값이 0인 state가 사용됩니다. Test에는 이전학습의 마지막 state가 적용되지 않습니다 (항상 0 state).

4) num\_units: 출력되는 vector의 크기를 결정합니다. 이 값은 필수입니다.

5) reuse\_Tensor\_Name: 다른 RNN\_GRU에서 weight를 가져옵니다. 이 경우 해당 layer와 weight를 공유합니다.

#### RNN Basic

Basic RNN cell을 사용한 RNN 연산을 실시합니다 (back propagation through time).

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B, time, vector]의 형태의 3D여야만 합니다. 이 값은 필수입니다.

3) state\_Reset: 만약 True라면 학습의 시작시 초기 state가 이전 학습의 마지막 state가 됩니다. False일 경우 모든 값이 0인 state가 사용됩니다. Test에는 이전학습의 마지막 state가 적용되지 않습니다 (항상 0 state).

4) num\_units: 출력되는 vector의 크기를 결정합니다. 이 값은 필수입니다.

5) reuse\_Tensor\_Name: 다른 RNN\_Basic에서 weight를 가져옵니다. 이 경우 해당 layer와 weight를 공유합니다.

#### RNN Feedback

Basic cell과 projection 모두를 state로 받는 RNN 연산을 실시합니다. (Elman + Jordan network). 이 cell은 hidden과 projection 모두를 출력합니다. 출력되는 vector의 크기는 num\_hidden\_units + num\_projection\_units 입니다. hidden vector 쪽은 tanh가 적용되고, projection vector는 non scaled logit 상태로 출력됩니다. 출력되는 tensor를 split으로 분할할 것을 권장합니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B, time, vector]의 형태의 3D여야만 합니다. 이 값은 필수입니다.

3) state\_Reset: 만약 True라면 학습의 시작시 초기 state가 이전 학습의 마지막 state가 됩니다. False일 경우 모든 값이 0인 state가 사용됩니다. Test에는 이전학습의 마지막 state가 적용되지 않습니다 (항상 0 state).

4) num\_hidden\_units: 출력되는 vector 중 hidden의 크기를 결정합니다. 이 값은 필수입니다.

5) num\_projection\_units: 출력되는 vector 중 projection의 크기를 결정합니다. 이 값은 필수입니다.



6) projection\_state\_activation: RNN 연산 내에서 projection에 적용할 activation function을 결정합니다. 이 값은 필수입니다. \*이 값은 출력되는 projection vector에는 적용되지 않습니다.

7) reuse\_Tensor\_Name: 다른 RNN\_Feedback에서 weight를 가져옵니다. 이 경우 해당 layer와 weight를 공유합니다.

#### iv. Activation Calc.

Tensor를 특정 범위의 값으로 normalized된 활성화 값으로 변환하는 function들의 모음입니다.

##### Sigmoid

Tensor에  $y = 1/(1 + \exp(-x))$ 의 식을 적용하여 0과 1의 범위 내의 값으로 변환합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

##### Softmax

Tensor에  $y = \exp(x)/\sum(\exp(\text{logits}))$ 를 적용하여 vector를 전체 합 1의 확률로 변환합니다. logits은 입력되는 전체 vector입니다. 2차원 이상일 경우 마지막 차원에 적용됩니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

##### Tanh

Tensor에 hyperbolic tangent(tanh)를 적용하여 -1에서 1사이의 값으로 전환합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

##### ReLU

Tensor에  $y = \max(x, 0)$ 을 적용하여 모든 0 이하의 값을 0으로 전환합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

##### Leaky\_ReLU

Tensor에  $y = \max(x, 0) + \min(0, \alpha * x)$ 을 적용합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) alpha: 함수에 사용될 인자 alpha를 설정합니다. 설정되지 않을 경우 0.2가 할당됩니다.

##### Softplus

Tensor에  $y = \log(\exp(x) + 1)$ 을 적용합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### v. Reshape

값의 변동 없이 tensor의 모양을 변환하는 function들의 모음입니다.

#### Concat

입력받은 tensor들을 지정된 차원을 기준으로 결합합니다. 만약 2D 이상일 경우 지정된 차원을 제외한 나머지 차원의 크기는 동일해야 합니다 (e.g. [B, 5, 32], [B, 5, 17] -> [B, 5, 49]).

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) axis: 결합할 차원을 지정합니다. 이 값은 필수입니다.

#### Split

입력받은 tensor를 지정된 차원 기준으로 분할합니다. num\_or\_size\_splits에 할당된 숫자의 개수만큼의 tensor가 생성됩니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) num\_or\_size\_splits: 분할된 각 tensor의 크기를 결정합니다. 전체의 숫자의 합은 분할되기 전의 차원의 크기와 동일해야 합니다. 이 값은 필수입니다.
- 3) axis: 분할할 차원을 지정합니다. 이 값은 필수입니다.

#### Expand Dimds

입력받은 tensor에 지정된 차원을 삽입합니다. (e.g. [B,3,5,7] -expand dim axis2-> [B,3,1,5,7])

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) axis: 삽입될 차원을 지정합니다. 이 값은 필수입니다.

#### Tile

각 차원별로 지정된 횟수만큼 반복/복사된 tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) multiples: 각 차원별 반복 횟수를 지정합니다. 이 값의 길이는 입력되는 tensor의 차원수와 일치해야 합니다. 이 값은 필수입니다.

#### Reshape

입력받은 tensor의 shape을 재정의 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

3) shape: 변형할 shape을 지정합니다. 전체 값의 합계는 입력받는 tensor의 전체 차원 크기의 합과 동일해야 합니다. 이 값은 필수입니다.

#### Flatten

입력받은 tensor의 모든 차원을 제거하고 1차원 vector의 batch로 변환합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Squeeze

입력받은 tensor에서 지정된 차원들을 제거합니다. 지정된 차원은 크기가 1이어야 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) axis: 제거할 차원들을 지정합니다. 지정된 차원들은 크기가 1이어야만 합니다. 이 값은 필수입니다.

#### vi. Fundamental math

기본적인 산술연산에 관련된 function들의 모음입니다.

#### Add\_n

입력받는 tensor들을 더합니다. 모든 tensor는 동일한 shape이어야 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Add

입력받는 두 tensor를 더합니다. 두 tensor는 동일한 shape이어야 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Subtract

첫 번째 tensor에서 두 번째 tensor를 뺍니다. 두 tensor는 동일한 shape이어야 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Multiply

입력받은 두 tensor를 elementwise-multiply합니다. 두 tensor는 동일한 shape이어야 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Divide

첫 번째 tensor에서 두 번째 tensor를 elementwise-divide합니다. 두 tensor는 동일한 shape이어야 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor들로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Mean

입력받은 tensor의 지정된 axis들을 기준으로 평균값을 구합니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

3) axis: 평균을 구할 기준이 되는 axis를 설정합니다.

4) keepdims: 평균을 구한 뒤에 tensor의 차원을 유지할 것인지 결정합니다. False일 경우 '입력된 axis 개수 - 1' 차원이 됩니다.

#### Sum

입력받은 tensor의 지정된 axis들을 기준으로 합계를 구합니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

3) axis: 합계를 구할 기준이 되는 axis를 설정합니다.

4) keepdims: 합계를 구한 뒤에 tensor의 차원을 유지할 것인지 결정합니다. False일 경우 '입력된 axis 개수 - 1' 차원이 됩니다.

#### Max

입력받은 tensor의 지정된 axis들을 기준으로 최대값을 구합니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

3) axis: 최대값을 구할 기준이 되는 axis를 설정합니다.

4) keepdims: 최대값을 구한 뒤에 tensor의 차원을 유지할 것인지 결정합니다. False일 경우 '입력된 axis 개수 - 1' 차원이 됩니다.

#### Min

입력받은 tensor의 지정된 axis들을 기준으로 최소값을 구합니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 진행할 것인지를 결정합니다. 이 값은 필수입니다.

3) axis: 최소값을 구할 기준이 되는 axis를 설정합니다.

4) keepdims: 최소값을 구한 뒤에 tensor의 차원을 유지할 것인지 결정합니다. False일 경우 '입력된 axis 개수 - 1' 차원이 됩니다.

#### vii. Loss calc.

두 tensor의 비교를 통해 학습에 사용되는 loss를 생성하는데 사용되는 function의 모음입니다.

#### Absolute\_Difference\_Loss

label과 prediction의 차의 절대값의 합입니다 (L1 loss). Label과 prediction은 동일한 shape이어야 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) label\_Tensor\_Name: 학습의 목표로 하는 관측값입니다. 이 값은 필수입니다.
- 3) prediction\_Tensor\_Name: 모델이 산출한 예측값입니다. 이 값은 필수입니다.
- 4) weights: 해당 loss에 가중치를 부여합니다. 설정하지 않으면 1입니다.

#### Mean\_Squared\_Error\_Loss

label과 prediction의 차의 제곱의 합입니다 (L2 loss). Label과 prediction은 동일한 shape이어야 합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) label\_Tensor\_Name: 학습의 목표로 하는 관측값입니다. 이 값은 필수입니다.
- 3) prediction\_Tensor\_Name: 모델이 산출한 예측값입니다. 이 값은 필수입니다.
- 4) weights: 해당 loss에 가중치를 부여합니다. 설정하지 않으면 1입니다.

#### Sigmoid\_Loss

Label과 logit의 Sigmoid cross entropy loss를 계산합니다. Label과 logit은 동일한 shape이어야 합니다. \* Logit은 sigmoid를 적용하기 전의 unscaled value입니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) label\_Tensor\_Name: 학습의 목표로 하는 관측값입니다. 이 값은 필수입니다.
- 3) logit\_Tensor\_Name: 모델이 산출한 unscaled logit입니다. 이 값은 필수입니다.
- 4) weights: 해당 loss에 가중치를 부여합니다. 설정하지 않으면 1입니다.

#### Softmax\_Loss

Label과 logit의 Softmax cross entropy loss를 계산합니다. Label과 logit은 동일한 shape이어야 합니다. \* Logit은 softmax를 적용하기 전의 unscaled value입니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) label\_Tensor\_Name: 학습의 목표로 하는 관측값입니다. 이 값은 필수입니다.
- 3) logit\_Tensor\_Name: 모델이 산출한 unscaled logit입니다. 이 값은 필수입니다.
- 4) weights: 해당 loss에 가중치를 부여합니다. 설정하지 않으면 1입니다.

#### Sparse\_Softmax\_Loss

Label과 logit의 Softmax cross entropy loss를 계산합니다. Label은 logit보다 한 차원 적으며, logit의 마지막 차원에 대한 index중 하나가 들어있는 int32 dtype이어야 합니다 (e.g. if logit shape= [32, 50], label shape= [32] and each value is from 0 to 49). \*Logit은 softmax를 적용하기 전의 unscaled logit입니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) label\_Tensor\_Name: 학습의 목표로 하는 관측값입니다. 이 값은 필수입니다.
- 3) logit\_Tensor\_Name: 모델이 산출한 unscaled logit입니다. 이 값은 필수입니다.
- 4) weights: 해당 loss에 가중치를 부여합니다. 설정하지 않으면 1입니다.

## viii. Optimizer

Optimizer는 process 내에 있는 모든 loss들을 더하고, 할당된 loss를 감소시켜 모델을 학습하도록 하는 연산자입니다. Optimizer는 고정된 이름을 갖고 있으며 process당 하나만 생성될 수 있습니다.

### ADAM\_Optimizer

ADAM 알고리즘을 사용한 optimizer를 생성합니다.

- 1) initial\_learning\_rate: 초기 학습률을 결정합니다. 설정하지 않으면 0.001이 할당됩니다.
- 2) beta1: beta1을 설정합니다. 설정하지 않으면 0.9가 할당됩니다.
- 3) beta2: beta2를 설정합니다. 설정하지 않으면 0.999가 할당됩니다.
- 4) epsilon: epsilon을 설정합니다. 설정하지 않으면  $1e-8$ 이 설정됩니다.
- 5) decay\_method: learning rate의 decay 방법을 설정합니다. 설정하지 않으면 no decay입니다.
- 6) decay\_steps: decay\_method가 exponential 방식일 때 decay 기준 중 하나입니다. 몇 번의 학습을 시도했을 때 decay\_rate에 도달할지 결정합니다. 설정하지 않으면 1000이며, Exponential 방식이 아닐 경우 이 값은 무시됩니다.
- 7) decay\_rate: decay\_method가 exponential 방식일 때 decay 기준 중 하나입니다. Decay\_steps에 도달했을 때, 어느 정도의 learning rate가 감소할지 결정합니다. 설정하지 않으면 0.5이며, Exponential 방식이 아닐 경우 이 값은 무시됩니다.
- 8) warmup\_steps: 학습이 warmup\_steps에 도달할 때까지 학습률은 점차 증가하며, warmup\_steps에서 initial\_learning\_rate가 됩니다. 그 이후엔 점차 감소합니다.
- 9) applied\_variable: 학습 중 업데이트가 진행되는 weight의 목록입니다. 제외된 값은 해당 process에서 업데이트가 진행되지 않고 고정됩니다.

### GD\_Optimizer

gradient descent 알고리즘을 사용한 optimizer를 생성합니다.

- 1) initial\_learning\_rate: 초기 학습률을 결정합니다. 설정하지 않으면 0.001이 할당됩니다.
- 2) decay\_method: learning rate의 decay 방법을 설정합니다. 설정하지 않으면 no decay입니다.
- 3) decay\_steps: decay\_method가 exponential 방식일 때 decay 기준 중 하나입니다. 몇 번의 학습을 시도했을 때 decay\_rate에 도달할지 결정합니다. 설정하지 않으면 1000이며, Exponential 방식이 아닐 경우 이 값은 무시됩니다.
- 4) decay\_rate: decay\_method가 exponential 방식일 때 decay 기준 중 하나입니다. Decay\_steps에 도달했을 때, 어느 정도의 learning rate가 감소할지 결정합니다. 설정하지 않으면 0.5이며, Exponential 방식이 아닐 경우 이 값은 무시됩니다.
- 5) warmup\_steps: 학습이 warmup\_steps에 도달할 때까지 학습률은 점차 증가하며, warmup\_steps에서 initial\_learning\_rate가 됩니다. 그 이후엔 점차 감소합니다.
- 6) applied\_variable: 학습 중 업데이트가 진행되는 weight의 목록입니다. 제외된 값은 해당 process에서 업데이트가 진행되지 않고 고정됩니다.

## ix. Test Calc.

분석에 주로 사용되는 loss값들을 계산하는 function들의 모음입니다. 위의 loss calc. 카테고리와는 다르게

test시 get tensor에 할당될 수 있고, 각 패턴별로 계산됩니다.

#### Mean\_Squared\_error

입력받은 두 tensor의 mean squared error tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Cross\_Entropy

입력받은 두 tensor의 cross entropy tensor를 생성합니다. 첫 번째 tensor가 label, 두 번째 tensor가 prediction으로 할당됩니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Cosine\_Similarity

입력받은 두 tensor의 cosine similarity tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Euclidean\_Distance

입력받은 두 tensor의 Euclidean distance tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Multi\_Test

입력받은 두 tensor의 mean squared error, cross entropy, cosine similarity, Euclidean distance tensor들을 각각 생성합니다. Cross entropy 계산의 경우 첫 번째 tensor가 label, 두 번째 tensor가 prediction으로 할당됩니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor들로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Arg\_Max

입력받은 tensor의 마지막 차원 기준으로 가장 높은 값을 갖는 index를 표시해주는 tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### Semantic\_Stress

입력받은 tensor의 semantic stress를 생성합니다. Semantic stress의 구체적 정의에 관해선 다음 논문을 참고하십시오:

Plaut, D. C. (1997). Structure and function in the lexical system: Insights from distributed models of word reading and lexical decision. *Language and cognitive processes*, 12(5-6), 765-806.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

#### x. Create

다른 tensor를 입력받지 않고, 주어진 shape과 함수에 따라서 자동적으로 tensor를 생성하는 Create function의 모음입니다.

##### Ones

모든 값이 1로 구성된 tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) shape: 생성되는 tensor의 shape을 설정합니다. 이 값은 필수입니다.

##### Zeors

모든 값이 0으로 구성된 tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) shape: 생성되는 tensor의 shape을 설정합니다. 이 값은 필수입니다.

##### Random\_Normal

Normal distribution에 따라 tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) shape: 생성되는 tensor의 shape을 설정합니다. 이 값은 필수입니다.
- 3) mean: Gaussian noise의 mean을 설정합니다. 설정하지 않을 경우 0이 할당됩니다.
- 4) stddev: Gaussian noise의 standard deviation을 설정합니다. 설정하지 않을 경우 1이 할당됩니다.

##### Random\_Uniform

uniform distribution에 따라 tensor를 생성합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) shape: 생성되는 tensor의 shape을 설정합니다. 이 값은 필수입니다.
- 3) minval: 생성되는 tensor의 최소값을 설정합니다. 이 값은 필수입니다.
- 4) maxval: 생성되는 tensor의 최대값을 설정합니다. 이 값은 필수입니다.

#### xi. Other

위의 10가지 카테고리에 포함되지 않는 기타 function들을 모음입니다.

##### Clip

입력받은 tensor의 주어진 범위 밖의 값을 깎아냅니다. min보다 낮은 값은 min으로, max보다 높은 값은 max로 고정됩니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.



3) clip\_value\_min: min값을 결정합니다. 이 값은 필수입니다.

4) clip\_value\_max: max값을 결정합니다. 이 값은 필수입니다.

#### Dropout

입력받은 tensor의 각 cell을 지정된 확률 rate로 0으로 만듭니다. 동시에, tensor에 1/rate를 곱합니다. Test에서는 이 function이 적용되지 않습니다 (입력된 tensor를 그대로 출력)

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

3) rate: dropout이 적용될 확률을 설정합니다. 설정하지 않을 경우 0.5가 할당됩니다.

#### Indexing

지정된 축의 지정된 인덱스만 추출합니다. Shape [B,3,5,7]인 tensor에 axis 2, index 3일 경우, 입력받은 텐서의[:, :, 3, :]인 shape [B,3,7]의 tensor가 생성됩니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

3) axis: 어느 축을 기준으로 추출할 것인지 결정합니다. 이 값은 필수입니다.

3) index: 설정된 축의 어느 인덱스를 추출할 것인지 결정합니다. 이 값은 필수입니다.

#### Embedding

입력받는 n차원 int tensor에 대해 n+1차원 float tensor를 생성합니다. 단일한 모델 내에서 같은 int값은 항상 같은 float vector가 됩니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 반드시 int tensor여야 합니다. 이 값은 필수입니다.

3) id\_size: 입력받는 int tensor의 값의 범위를 설정합니다. 이 값은 필수입니다.

4) embedding\_size: 생성되는 float vector의 크기를 설정합니다. 이 값은 필수입니다.

5) reuse\_Tensor\_Name: 다른 embedding에서 embedding variable을 가져옵니다. 이 경우 두 embedding은 같은 int값에 대해 같은 float vector를 생성합니다.

#### Noise\_Normal

입력받는 tensor에 Gaussian noise를 추가합니다.

1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.

2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.

3) mean: Gaussian noise의 mean을 설정합니다. 설정하지 않을 경우 0이 할당됩니다.

4) stddev: Gaussian noise의 standard deviation을 설정합니다. 설정하지 않을 경우 1이 할당됩니다.

#### Noise\_Uniform

입력받는 tensor에 uniform noise를 추가합니다.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) minval: noise의 최소값을 설정합니다. 이 값은 필수입니다.
- 4) maxval: noise의 최대값을 설정합니다. 이 값은 필수입니다.

#### Batch\_Normalization

입력받는 tensor의 마지막차원에 batch normalization을 실시한 tensor를 생성합니다. Batch normalization에 대한 보다 자세한 사항은 tensorflow의 'tf.layers.batch\_normalization'에 관한 문서와 다음 논문을 참조하십시오:

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

- 1) Name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Name: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 이 값은 필수입니다.
- 3) momentum: momentum을 설정합니다. 설정하지 않을 경우 0.99가 할당됩니다.
- 4) epsilon: epsilon을 설정합니다. 설정하지 않을 경우 0.001이 할당됩니다.
- 5) reuse\_Tensor\_Name: 다른 batch normalization에서 weight를 가져옵니다. 이 경우 두 batch normalization은 weight를 공유합니다.

#### Max\_Pooling1d

입력된 3d tensor에 max pooling을 실시합니다.

- 1) name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Names: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B, step, channel]형태의 3D 여야만 합니다. 이 값은 필수입니다.
- 3) pool\_size: pooling window의 크기를 결정합니다. 이 값은 필수입니다.
- 5) strides: pooling window 이동의 크기를 결정합니다. 이 값은 필수입니다.
- 6) padding: padding을 통해 pooling 후 tensor의 shape이 입력된 tensor와 동일하게 할 것인지 결정합니다.

#### Max\_Pooling2d

입력된 4d tensor에 max pooling을 실시합니다.

- 1) name: process 내에서의 구분자로 기능합니다. 이 값은 학습에 영향을 주지 않습니다.
- 2) input\_Tensor\_Names: 어느 tensor로부터 계산을 진행할 것인지를 결정합니다. 입력되는 tensor의 shape은 [B, height, width, channel]형태의 4D 여야만 합니다. 이 값은 필수입니다.
- 3) pool\_size: pooling window의 크기를 결정합니다. 이 값은 필수입니다.
- 5) strides: pooling window 이동의 크기를 결정합니다. 이 값은 필수입니다.
- 6) padding: padding을 통해 pooling 후 tensor의 shape이 입력된 tensor와 동일하게 할 것인지 결정합니다.