텐서플로와 머신러닝으로 시작하는 자연어처리

Lecture 1
2019. 06. 01

강사 소개

전창욱

- 2018 국어 정보 경진 대회 금상
- 텐서플로와 머신러닝으로 시작하는 자연어 처리 집필
- Korquad 최대 6등 기록
- 현) 네이블 커뮤니케이션즈 시니어 개발자

조중현

- 중앙대학교 수학과
- 텐서플로와 머신러닝으로 시작하는 자연어 처리 집필
- AI blog *“reniew’s blog”* 운영
- 현) 뤼이드 AI Researcher

텐서플로와 머신러닝으로 시작하는 자연어 처리

로지스틱 회귀부터 트랜스포머 챗봇까지

전창욱, 최태균, 조중현 지음

DS 데이터 사이언스 시리즈 _030

위키북스

01 들어가며

02 자연어 처리 개발 준비

텐서플로와 머신러닝으로 시작하는 자연어 처리

로지스틱 회귀부터 트랜스포머 챗봇까지

전창욱, 최태균, 조중현 지음

DS 데이터 사이언스 시리즈 _030

위키북스

03

자연어 처리 개요

텐서플로와 머신러닝으로 시작하는 자연어 처리

로지스틱 회귀부터 트랜스포머 챗봇까지

전창욱, 최태균, 조중현 지음

DS 데이터 사이언스 시리즈 _030

위키북스

텐서플로와 머신러닝으로 시작하는 자연어 처리

로지스틱 회귀부터 트랜스포머 챗봇까지

전창욱, 최태균, 조중현 지음

DS 데이터 사이언스 시리즈 _030

위키북스

06 챗봇 만들기

부록

텐서플로와 머신러닝으로 시작하는 자연어 처리

로지스틱 회귀부터 트랜스포머 챗봇까지

DS 데이터 사이언스 시리즈 _030

전창욱, 최태균, 조중현 지음

위키북스

What do we learn?

What do we learn?

- Basic TensorFlow
- Word Representation
- Text Classification
- Text Similarity
- Text Generation / Machine Translation

What do we learn?

- Basic TensorFlow – 1주차
- Word Representation – 2주차
- Text Classification – 3주차
- Text Similarity – 4 주차
- Text Generation / Machine Translation – 5 - 8 주차

What do we learn?

- Basic TensorFlow
  - Traditional TensorFlow implementation
  - Estimator vs Session
  - tf.data vs Placeholder & feed_dict
  - TensorFlow modules

What do we learn?

- Word Representation
  - Data format
    - Tabular data
    - Image data
    - Text data
  - One - hot
  - Bag of Words
  - Neural Text Representation
    - Word2Vec
    - GloVe
    - FastText
    - ELMo

What do we learn?

- Text Classification
  - What is Text Classification?
  - Why do we need Text Classification?
  - Text Classification Dataset
  - How do we evaluate Text Classification Model?
  - How does Text Classification work?
  - Classification Papers
    - Convolutional neural networks for sentence classification
    - Character-level Convolutional Networks for Text Classification

What do we learn?

- Text Similarity
  - What is Text Similarity?
  - Why do we need Text Similarity?
  - Similarity 측정
  - Measuring Similarity by Classification
  - Natural Language Inference
  - Natural Language Inference Dataset
  - Similarity Algorithm
    - Ma-LSTM(Paper)
    - XG-Boost

What do we learn?

- Text Generation / Machine Translation
  - What is Text Generation
  - What is Neural Machine Translation
  - Papers
    - Sequence to Sequence
    - Neural Machine Translation by Jointly Learning to Align and Translate
    - Attention is All You Need
  - How do we evaluate?
  - Neural Machine Translation Dataset
  - What is Attention Mechanism
  - Why do we need Attention
  - Various Attention Mechanism
  - Another methods to enrich NMT performance
  - Transformer

Why do we learn?

Why do we learn?

ex) 커리어 전환, 실무에서의 자연어 처리 활용 ….

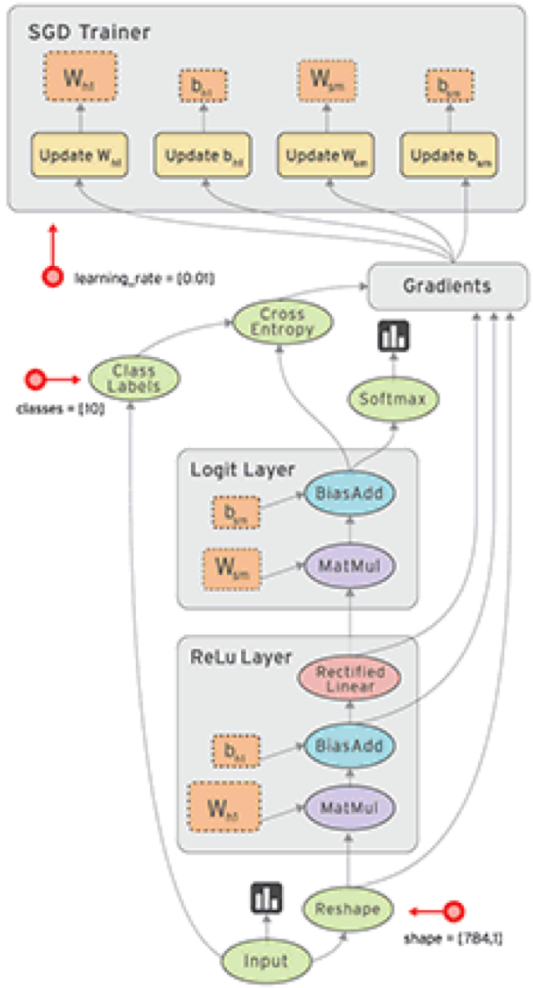
Basic TensorFlow

Basic TensorFlow

1. Traditional TensorFlow implementation
2. Estimator vs Session
3. tf.data vs Placeholder & feed_dict
4. TensorFlow modules

TensorFlow

Graphs and Sessions



TensorFlow

TensorFlow는 (1) 연산(Operation)을 정의하는 것과

(2) 정의한 연산을 실행하는 부분이 분리되어 있다.

TensorFlow

TensorFlow는 (1) 연산(Operation)을 정의하는 것과 → **Graph**

(2) 정의한 연산을 실행하는 부분이 분리되어 있다. → **Session**

TensorFlow

TensorFlow는 (1) 연산(Operation)을 정의하는 것과 → **Graph**

(2) 정의한 연산을 실행하는 부분이 분리되어 있다. → **Session**

Not in eager execution!

TensorFlow

1. Define operation

```
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)

# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)
```

TensorFlow

1. Define operation
2. Execute operation by Session

```
# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow

```python
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)

# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow

모델 파라미터 정의

```
import tensorflow as tf
import numpy as np
```

```
# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)
```

```
# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow

Placeholder 정의

```
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)
```

```
# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)
```

```
# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow

Graph 그리기

```
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)

# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)
```

```
# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2
```

```
# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow

Loss, Optimizer, Prediction 정의

```
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)

# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2
```

```
# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)
```

```
# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow

Run Training by Session

```python
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)

# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)
```

```python
# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow

그 외 학습, 검증, 예측, 배포를 위해서 추가로 구현 필요.

```python
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)

# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

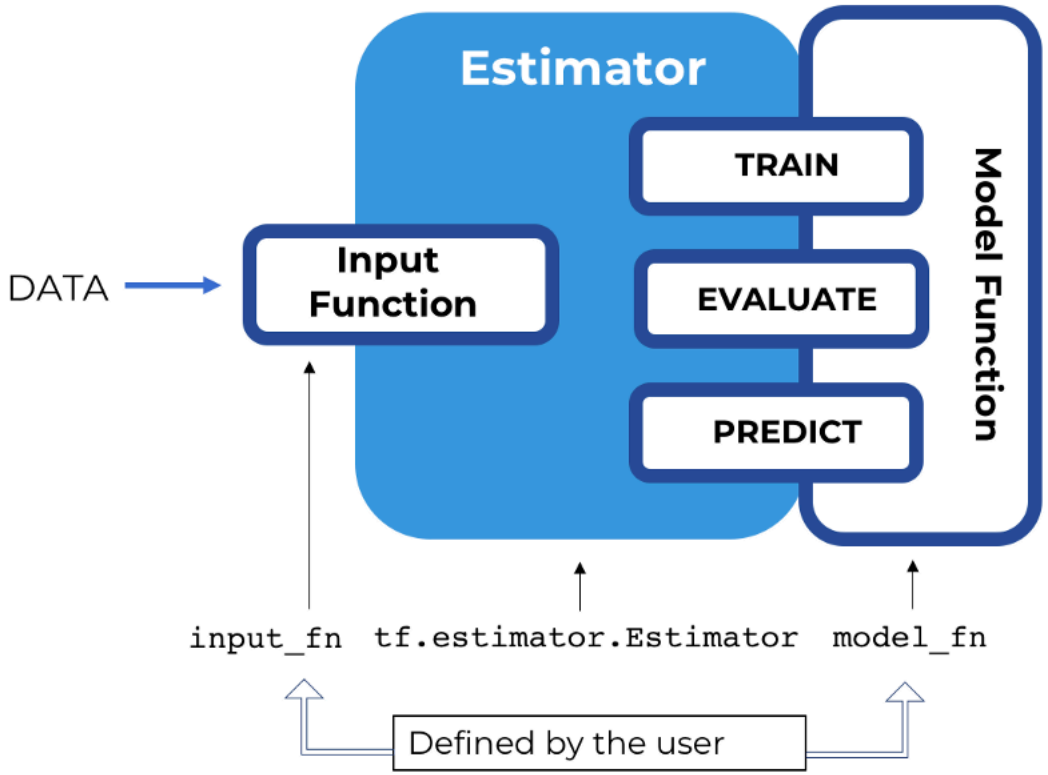
Estimator

Estimator

“TensorFlow’s high-level machine learning API (tf.estimator) makes it easy to configure, train, and evaluate a variety of machine learning models”

Estimator

"TensorFlow's high-level machine learning API (tf.estimator) makes it easy to configure, train, and evaluate a variety of machine learning models"



https://torres.ai

Estimator

Estimator 의 장점

1. High-Level API를 통해 간단하게 모델을 구현할 수 있다.
2. 개발, 연구자들 사이에서 모델을 간단하게 공유할 수 있다.
3. Model을 변경할 필요 없이 CPU, GPU, TPU에서 실행 할 수 있다.
4. 아래의 기능을 직접 구현할 필요없이 사용 할 수 있다.
    - Build Graph
    - Train / Evaluate / Predict
    - 배포(export)
    - Variable 초기화
    - Data 불러오기
    - Checkpoint 파일을 만들기
    - TensorBoard를 사용하기 위한 Summaries를 저장

Estimator

1. Pre-made Estimator
2. Custom Estimator

Estimator

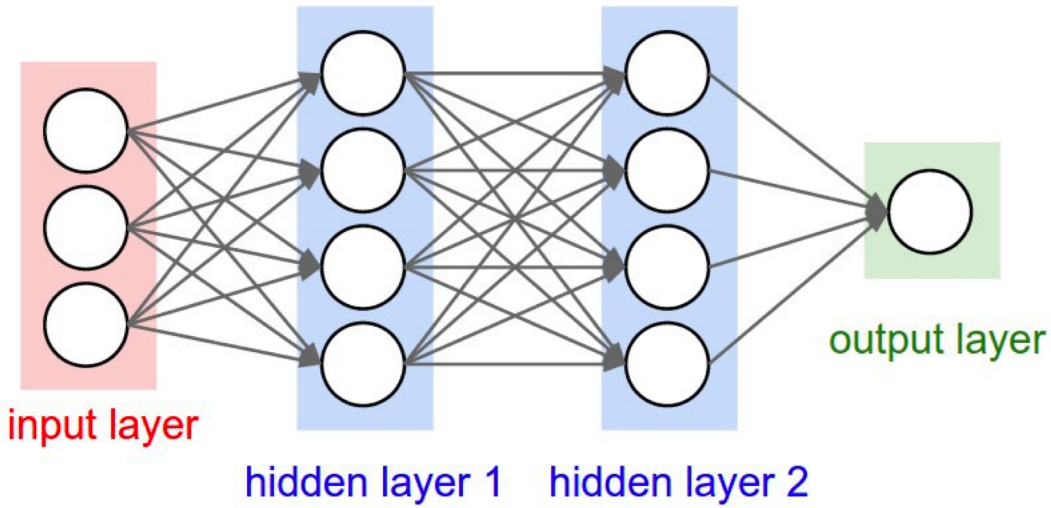
Pre-made Estimator

```
classifier = tf.estimator.DNNClassifier(feature_columns=feature_columns,
                                        hidden_units = [256, 32],
                                        optimizer = tf.train.AdamOptimizer(learning_rate),
                                        n_classes = 10,
                                        dropout = 0.1,
                                        model_dir="./dnn_classifier/")
```

Estimator

Pre-made Estimator

```
classifier = tf.estimator.DNNClassifier(feature_columns=feature_columns,
                                        hidden_units = [256, 32],
                                        optimizer = tf.train.AdamOptimizer(learning_rate),
                                        n_classes = 10,
                                        dropout = 0.1,
                                        model_dir="./dnn_classifier/")
```



https://hackernoon.com/

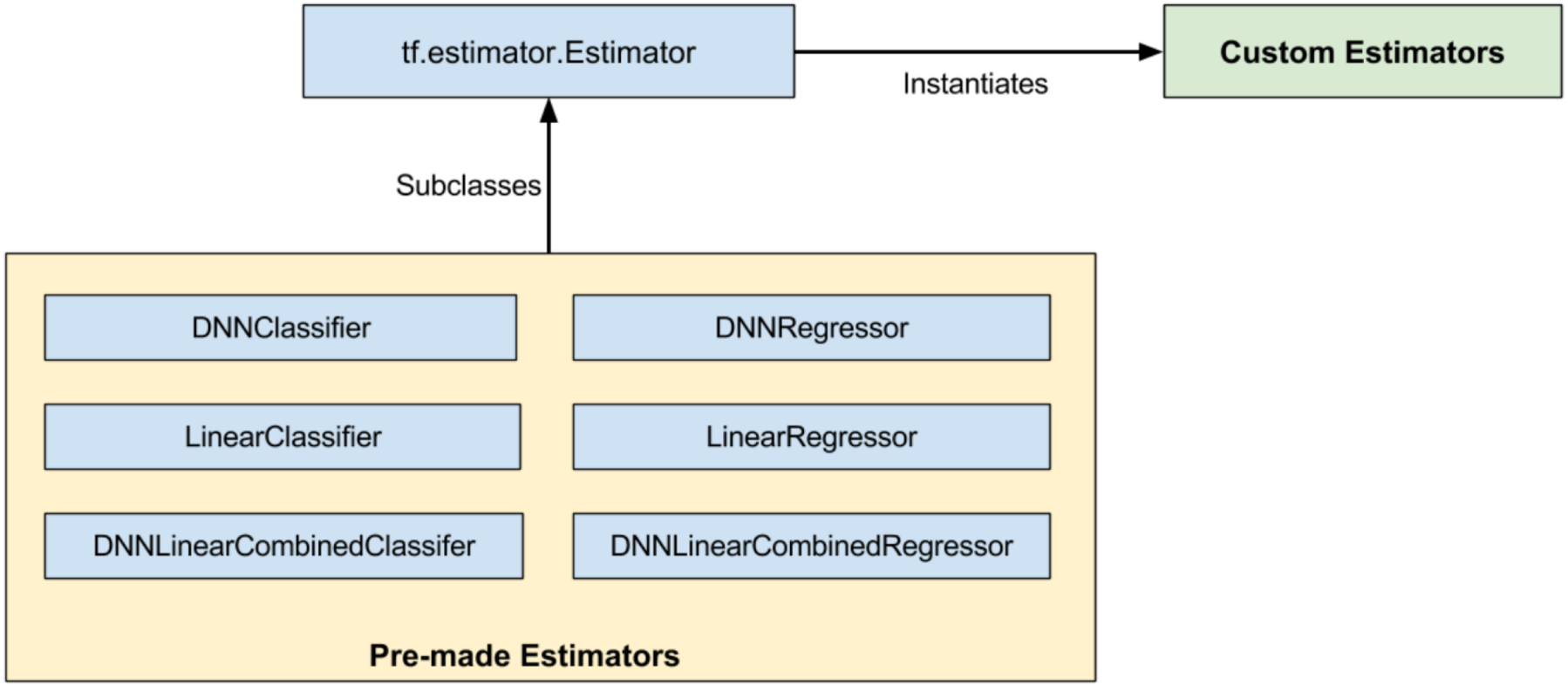
Estimator

Pre-made Estimator

```
classifier = tf.estimator.DNNClassifier(feature_columns=feature_columns,
                                        hidden_units = [256, 32],
                                        optimizer = tf.train.AdamOptimizer(learning_rate),
                                        n_classes = 10,
                                        dropout = 0.1,
                                        model_dir="./dnn_classifier/")


# 학습
classifier.train(input_fn = train_input_fn)

# 검증
classifier.evaluate(input_fn = eval_input_fn)

# 예측
prediction = classifier.predict(input_fn = train_input_fn)
```

Estimator

Custom Estimator



https://www.tensorflow.com

Estimator

Custom Estimator

1. Define Input function
2. Define Model function
3. Instantiate Estimator

Estimator

Custom Estimator (define input function)

```
def train_input_fn(features, labels, batch_size):
    ...

    return features, labels
```

Estimator

Custom Estimator (define model function)

```
def model_fn(features, labels, mode, params):
    """

    Operations

    """

    return tf.estimator.EstimatorSpec( ... )
```

Estimator

Custom Estimator (define model function)

```
def model_fn(features, labels, mode, params):
    """

    Operations

    """

    return tf.estimator.EstimatorSpec( ... )
```

- features: 모델의 입력값들의 dictionary
- labels: 모델의 라벨
- mode: 현재 모델 함수가 실행된 상태(train/evaluate/predict)
- params: (Optional) 모델에 필요한 추가적인 하이퍼 파라미터

Estimator

Custom Estimator (define model function)

```
def model_fn(features, labels, mode, params):
    """

    Operations

    """

    return tf.estimator.EstimatorSpec(mode=mode,
                                      predictions=predictions,
                                      loss=loss,
                                      eval_metric_ops = metrics,
                                      train_op=train_op)
```

Estimator

Custom Estimator (instantiate Estimator)

```
estimator = tf.estimator.Estimator(model_fn = model_fn,
                                   params = {'hidden_dims': 256,
                                             'dropout_rate': 0.5},
                                   model_dir = './custom_model/')
```

Estimator

Custom Estimator (instantiate Estimator)

```
estimator = tf.estimator.Estimator(model_fn = model_fn,
                                   params = {'hidden_dims': 256,
                                             'dropout_rate': 0.5},
                                   model_dir = './custom_model/')
#train
estimator.train(train_input_fn)

#evaluate
estimator.evaluate(eval_input_fn)

#predict
estimator.predict(pred_input_fn)
```

tf.data

tf.data

TensorFlow Dataset API

```
tf.data.Dataset()
```

tf.data

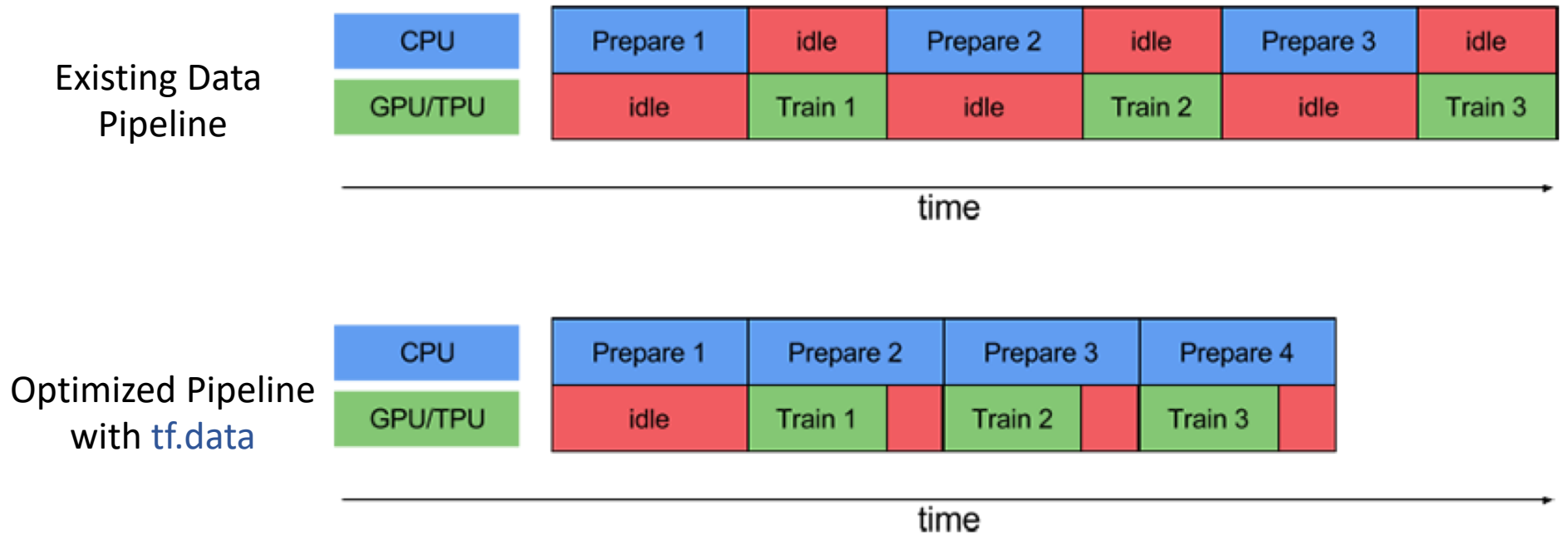
Advantage of Dataset API

- 기존의 placeholder & feed_dict 방법보다 높은 성능 (속도 측면)
- 간단한 batch, epoch, shuffle, map 구현
- tf.estimator 와 같이 사용하기에 적합

tf.data

Advantage of Dataset API

- **기존의 placeholder & feed_dict 방법보다 높은 성능 (속도 측면)**
- 간단한 batch, epoch, shuffle, map 구현
- tf.estimator 와 같이 사용하기에 적합



tf.data

Advantage of Dataset API

- 기존의 placeholder & feed_dict 방법보다 높은 성능 (속도 측면)
- **간단한 batch, epoch, shuffle, map 구현**
- tf.estimator 와 같이 사용하기에 적합

```
dataset = tf.data.Dataset.from_tensor_slices(data)

# Batch
dataset = dataset.batch(batch_size)

# Repeat for Epochs
dataset = dataset.repeat(epoch)

# Shuffle
dataset = datset.shuffle(buffer_size)

# Mapping
datset = datset.map(mapping_fn)
```

tf.data

Advantage of Dataset API

- 기존의 placeholder & feed_dict 방법보다 높은 성능 (속도 측면)
- 간단한 batch, epoch, shuffle, map 구현
- **tf.estimator 와 같이 사용하기에 적합**

```
#train
estimator.train(train_input_fn)
```

tf.data

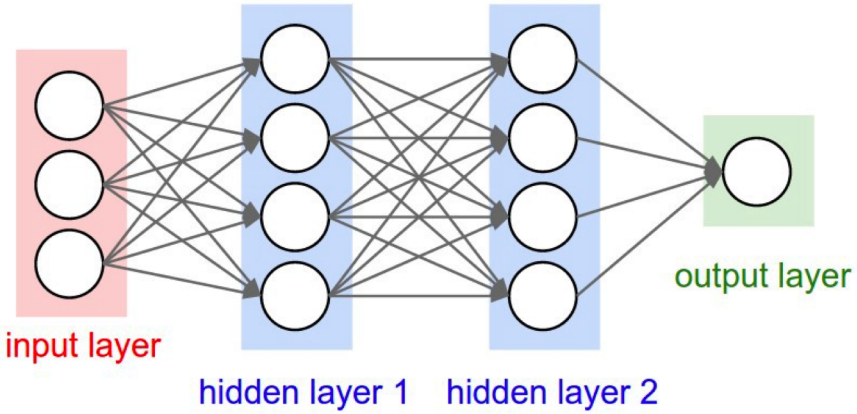
Advantage of Dataset API

- 기존의 placeholder & feed_dict 방법보다 높은 성능 (속도 측면)
- 간단한 batch, epoch, shuffle, map 구현
- **tf.estimator 와 같이 사용하기에 적합**

```
def train_input_fn():
    dataset = tf.data.Dataset.from_tensor_slices(data)
    dataset = dataset.batch(batch_size)
    datset = dataset.repeat(epoch)
    datset = dataset.shuffle(buffer_size)
    datset = datset.map(mapping_fn)

    return datset
```

TensorFlow Modules

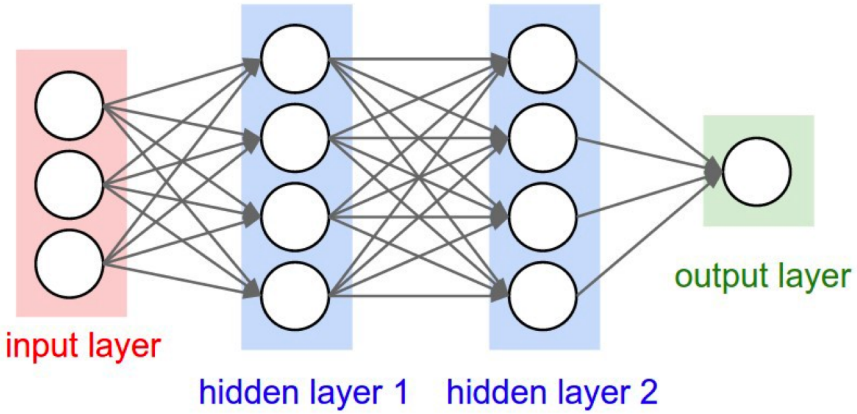
TensorFlow Modules



https://hackernoon.com/

```python
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)

# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow Modules



https://hackernoon.com/

```python
import tensorflow as tf
import numpy as np
```

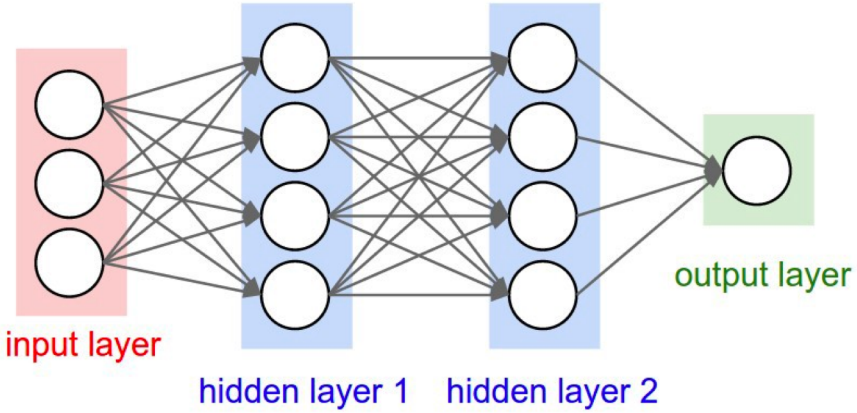
```python
# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)
```

```python
# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)
```

```python
# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2
```

```python
# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

TensorFlow Modules



https://hackernoon.com/

```
out = tf.keras.layers.Dense(units = hidden_units, activation = tf.nn.relu)(x)
logits = tf.keras.layers.Dense(units = num_class)(out)
```

TensorFlow Modules

Module: tf.nn

Module: tf.layers

Module: tf.keras.layers

TensorFlow Modules

```
# tf.layers modeule
tf.layers.conv1d(inputs = input_data, filters = conv_dim, kernel_size = kernel_size)

# tf.nn module
tf.nn.conv1d(value = input_data, filters = conv_filter, stride = 1, padding = 'VALID')

# tf.keras.layers module
tf.keras.layers.Conv1D(filters = conv_dim, kernel_size = kernel_size)(input_data)
```

TensorFlow Modules

```
# tf.layers modeule
tf.layers.conv1d(inputs = input_data, filters = conv_dim, kernel_size = kernel_size)

# tf.nn module
tf.nn.conv1d(value = input_data, filters = conv_filter, stride = 1, padding = 'VALID')

# tf.keras.layers module
tf.keras.layers.Conv1D(filters = conv_dim, kernel_size = kernel_size)(input_data)
```

TensorFlow 2.0 에서는 대부분 tf.kears.layers 모듈로 통합

TensorFlow Modules

```
# tf.layers modeule
tf.layers.conv1d(inputs = input_data, filters = conv_dim, kernel_size = kernel_size)

# tf.nn module
tf.nn.conv1d(value = input_data, filters = conv_filter, stride = 1, padding = 'VALID')

# tf.keras.layers module
tf.keras.layers.Conv1D(filters = conv_dim, kernel_size = kernel_size)(input_data)
```

변환 후

- 층을 단순하게 쌓을 경우엔 `tf.keras.Sequential` 이 적합합니다. (복잡한 모델인 경우 맞춤형 층과 모델이나 함수형 API를 참고하세요.)
- 모델이 변수와 규제 손실을 관리합니다.
- `tf.layers` 에서 `tf.keras.layers` 로 바로 매핑되기 때문에 일대일로 변환됩니다.

https://www.tensorflow.org/alpha/guide/migration_guide

TensorFlow Modules

```
# tf.layers modeule
tf.layers.conv1d(inputs = input_data, filters = conv_dim, kernel_size = kernel_size)

# tf.nn module
tf.nn.conv1d(value = input_data, filters = conv_filter, stride = 1, padding = 'VALID')

# tf.keras.layers module
tf.keras.layers.Conv1D(filters = conv_dim, kernel_size = kernel_size)(input_data)
```

변환 후

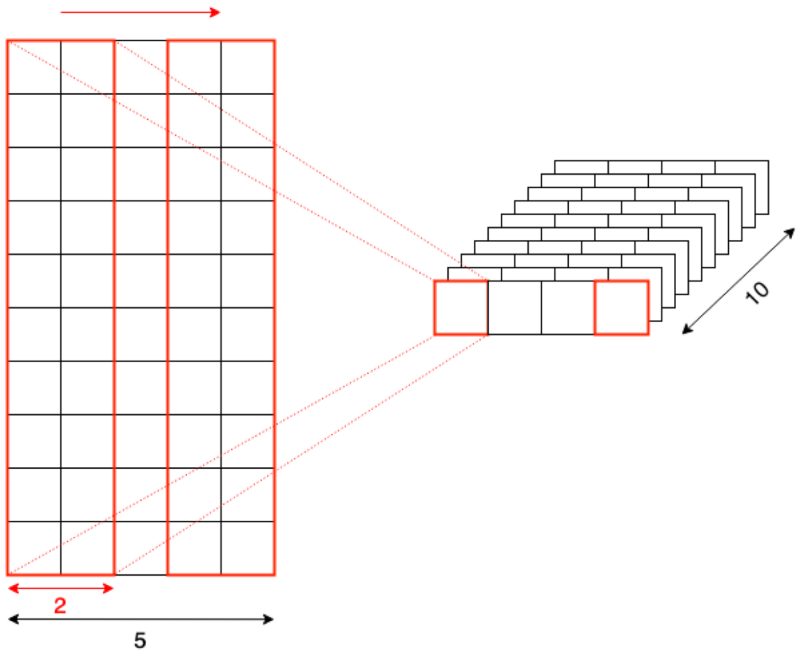
- 층을 단순하게 쌓을 경우엔 `tf.keras.Sequential` 이 적합합니다. (복잡한 모델인 경우 맞춤형 층과 모델이나 함수형 API를 참고하세요.)
- 모델이 변수와 규제 손실을 관리합니다.
- `tf.layers` 에서 `tf.keras.layers` 로 바로 매핑되기 때문에 일대일로 변환됩니다.

https://www.tensorflow.org/alpha/guide/migration_guide

TensorFlow Modules

1. tf.keras.layers.Conv1D

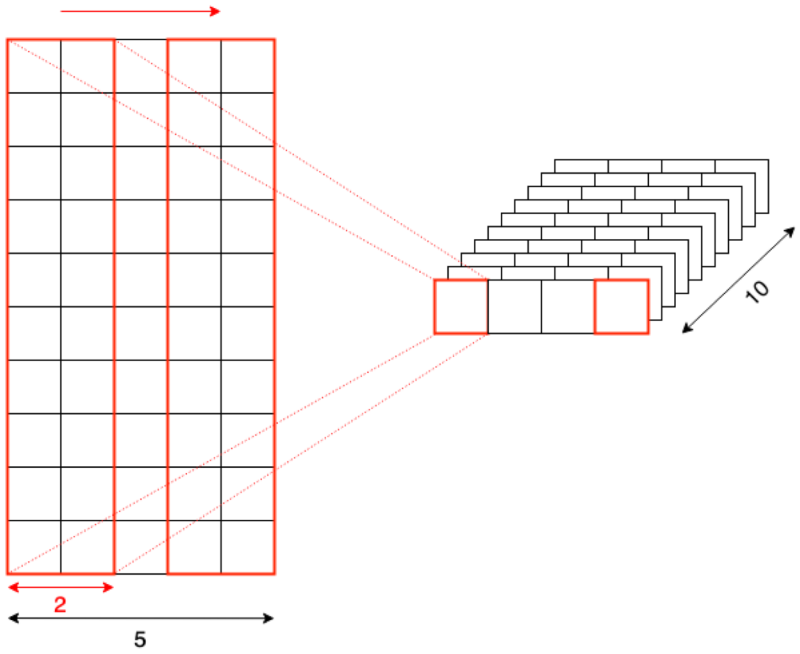
```
tf.keras.layers.Conv1D(filters = 10,
                       kernel_size = 2,
                       strides = 1,
                       padding = 'valid')
```



TensorFlow Modules

1. tf.keras.layers.Conv1D

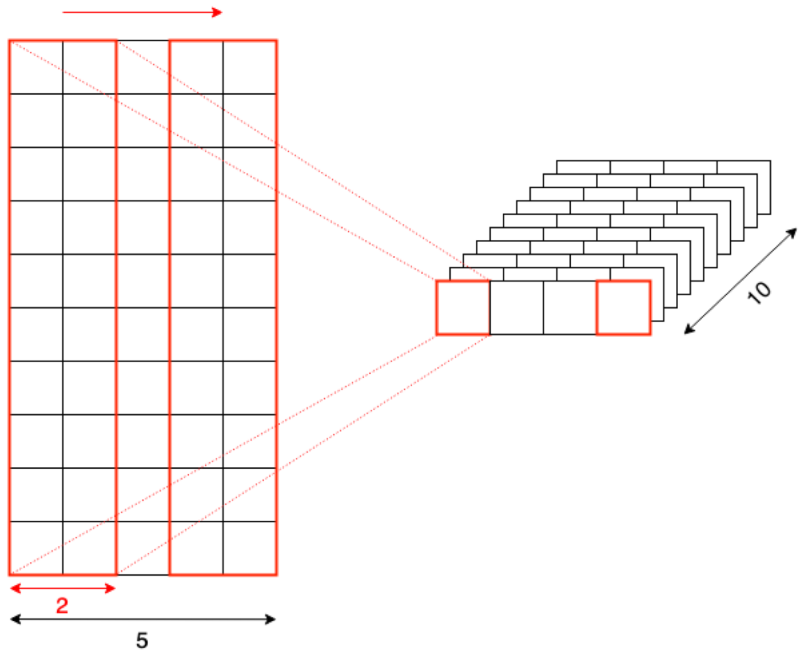
```
tf.keras.layers.Conv1D(filters = 10,
                       kernel_size = 2)
```



TensorFlow Modules

1. tf.keras.layers.Conv1D

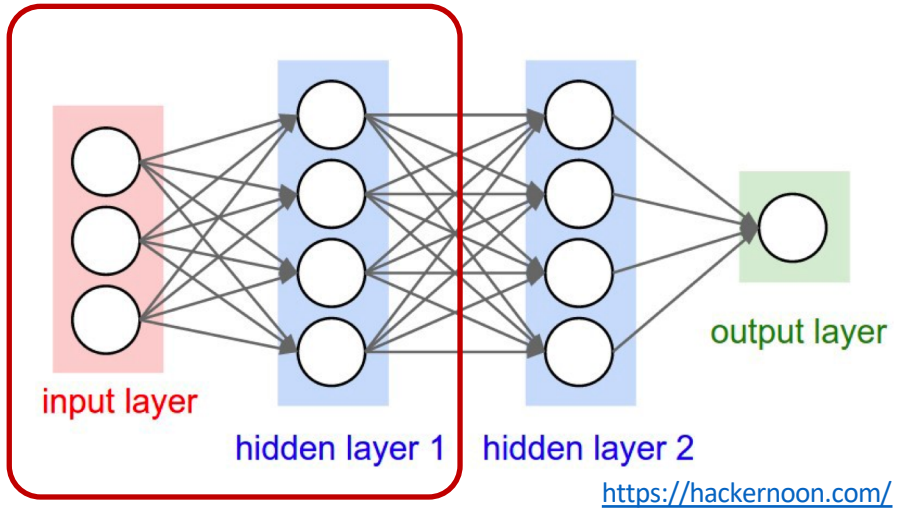
```
conv_out = tf.keras.layers.Conv1D(filters = 10, kernel_size = 2)(inputs)
```



TensorFlow Modules

2. tf.keras.layers.Dense

```
tf.keras.layers.Dense(units = 4, activation = None)
```

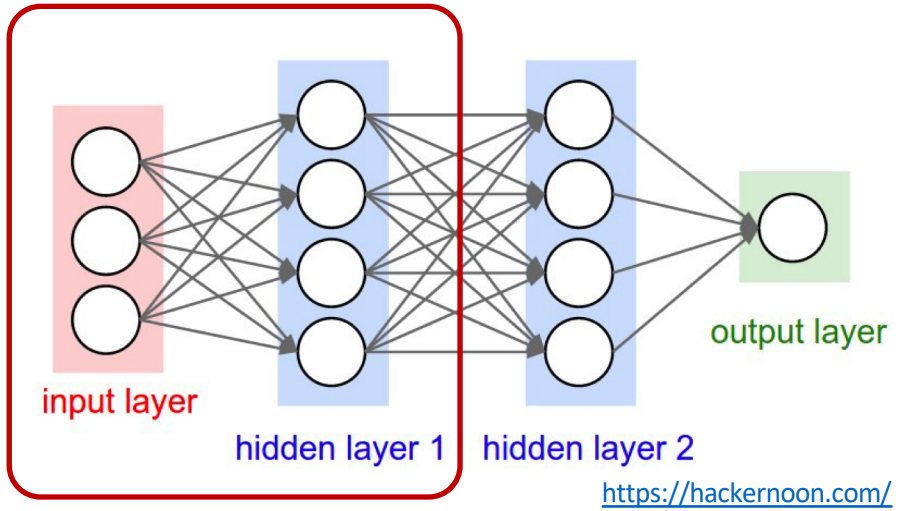


https://hackernoon.com/

TensorFlow Modules

2. tf.keras.layers.Dense

```
tf.keras.layers.Dense(units = 4, activation = tf.nn.sigmoid)(inputs)
```

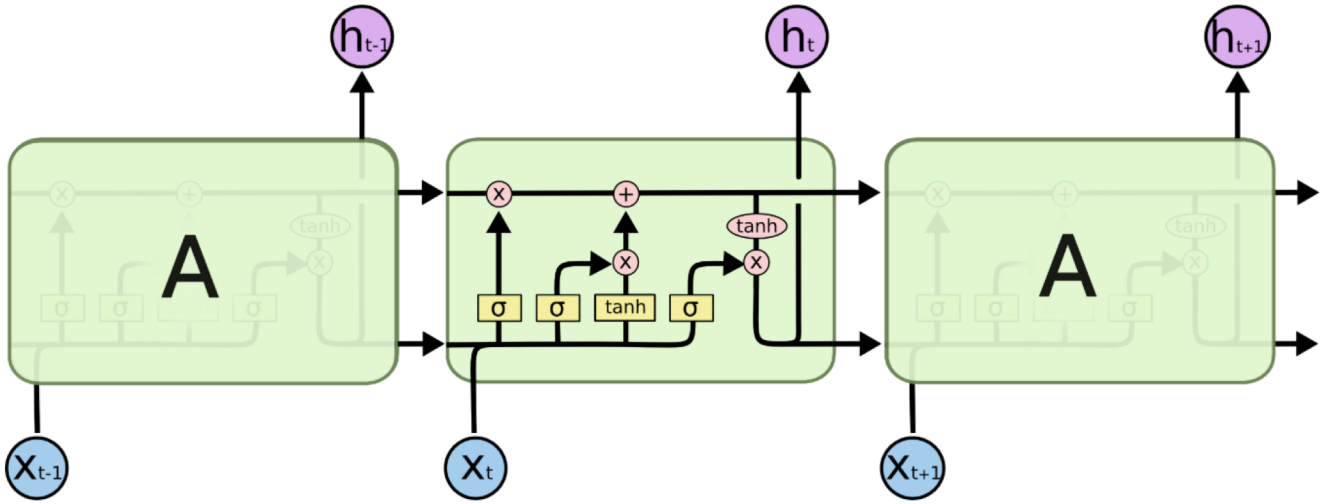


https://hackernoon.com/

TensorFlow Modules

3. tf.keras.layers.LSTM

```
tf.keras.layers.LSTM(units = lstm_dim)
```



http://colah.github.io/

TensorFlow Modules

More Arguments

```
tf.keras.layers.Conv1D(filters,
                       kernel_size,
                       strides=1,
                       padding='valid',
                       data_format='channels_last',
                       dilation_rate=1,
                       activation=None,
                       use_bias=True,
                       kernel_initializer='glorot_uniform',
                       bias_initializer='zeros',
                       kernel_regularizer=None,
                       bias_regularizer=None,
                       activity_regularizer=None,
                       kernel_constraint=None,
                       bias_constraint=None,
                       **kwargs)
```

```
tf.keras.layers.Dense(units,
                      activation=None,
                      use_bias=True,
                      kernel_initializer='glorot_uniform',
                      bias_initializer='zeros',
                      kernel_regularizer=None,
                      bias_regularizer=None,
                      activity_regularizer=None,
                      kernel_constraint=None,
                      bias_constraint=None,
                      **kwargs)
```

```
tf.keras.layers.LSTM(units,
                     activation='tanh',
                     recurrent_activation='hard_sigmoid',
                     use_bias=True,
                     kernel_initializer='glorot_uniform',
                     recurrent_initializer='orthogonal',
                     bias_initializer='zeros',
                     unit_forget_bias=True,
                     kernel_regularizer=None,
                     recurrent_regularizer=None,
                     bias_regularizer=None,
                     activity_regularizer=None,
                     kernel_constraint=None,
                     recurrent_constraint=None,
                     bias_constraint=None,
                     dropout=0.0,
                     recurrent_dropout=0.0,
                     implementation=1,
                     return_sequences=False,
                     return_state=False,
                     go_backwards=False,
                     stateful=False,
                     unroll=False,
                     **kwargs)
```

Traditional
=> Our style implementation

```
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)

# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

Traditional
=> Our style implementation

```
def model_fn(mode, features, labels):

    hidden = tf.keras.layers.Dense(hidden_dimension)(features['inputs'])
    logits = tf.keras.layers.Dense(num_classes)(hidden)

    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))
    optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)
    prediction = tf.nn.softmax(logits, axis = -1)

    return tf.estimator.EstimatorSpec(mode, predictions = prediction, loss = loss, train_op = optimizer)

estimator = tf.estimator.Estimator(model_fn)
estimator.train(train_input_fn)
```

Traditional
=> Our style implementation

Using TensorFlow Module

```
import tensorflow as tf
import numpy as np
```

```
# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)
```

```
# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)
```

```
# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2
```

```
# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

Traditional
=> Our style implementation

Using TensorFlow Module

```
def model_fn(mode, features, labels):

    hidden = tf.keras.layers.Dense(hidden_dimension)(features['inputs'])
    logits = tf.keras.layers.Dense(num_classes)(hidden)

    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))
    optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)
    prediction = tf.nn.softmax(logits, axis = -1)

    return tf.estimator.EstimatorSpec(mode, predictions = prediction, loss = loss, train_op = optimizer)

estimator = tf.estimator.Estimator(model_fn)
estimator.train(train_input_fn)
```

Traditional
=> Our style implementation

Using Estimator, tf.data

```
import tensorflow as tf
import numpy as np

# 모델 파라미터 정의
W_1 = tf.get_variable('W1', shape = [input_shape, hidden_dimension], dtype = tf.float32)
b_1 = tf.get_variable('b1', shape = [hidden_dimension], dtype = tf.float32)

W_2 = tf.get_variable('W2', shape = [hidden_dimension, num_classes], dtype = tf.float32)
b_2 = tf.get_variable('b2', shape = [num_classes], dtype = tf.float32)
```

```
# Placeholder 정의
x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)
```

```
# Graph 그리기
hiddens = tf.nn.relu(tf.matmul(x, W_1) + b_1)
logits = tf.matmul(hiddens, W_2) + b_2

# loss 함수 정의
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))

# optimizer 정의
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

# prediction 정의
prediction = tf.nn.softmax(logits, axis = -1)

# Run by Session
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
```

```
for epoch in range(total_epoch):
    epoch_loss = 0
    for input_data, label_data in zip(train_inputs, train_labels):
        _, loss_ = sess.run([optimizer, loss], feed_dict = {x: input_data, y: label_data})
        epoch_loss += loss_
```

Traditional
=> Our style implementation

Using Estimator, tf.data

```
def model_fn(mode, features, labels):
    hidden = tf.keras.layers.Dense(hidden_dimension)(features['inputs'])
    logits = tf.keras.layers.Dense(num_classes)(hidden)

    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = logits, labels = y))
    optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)
    prediction = tf.nn.softmax(logits, axis = -1)

    return tf.estimator.EstimatorSpec(mode, predictions = prediction, loss = loss, train_op = optimizer)

estimator = tf.estimator.Estimator(model_fn)
estimator.train(train_input_fn)
```

Implementation