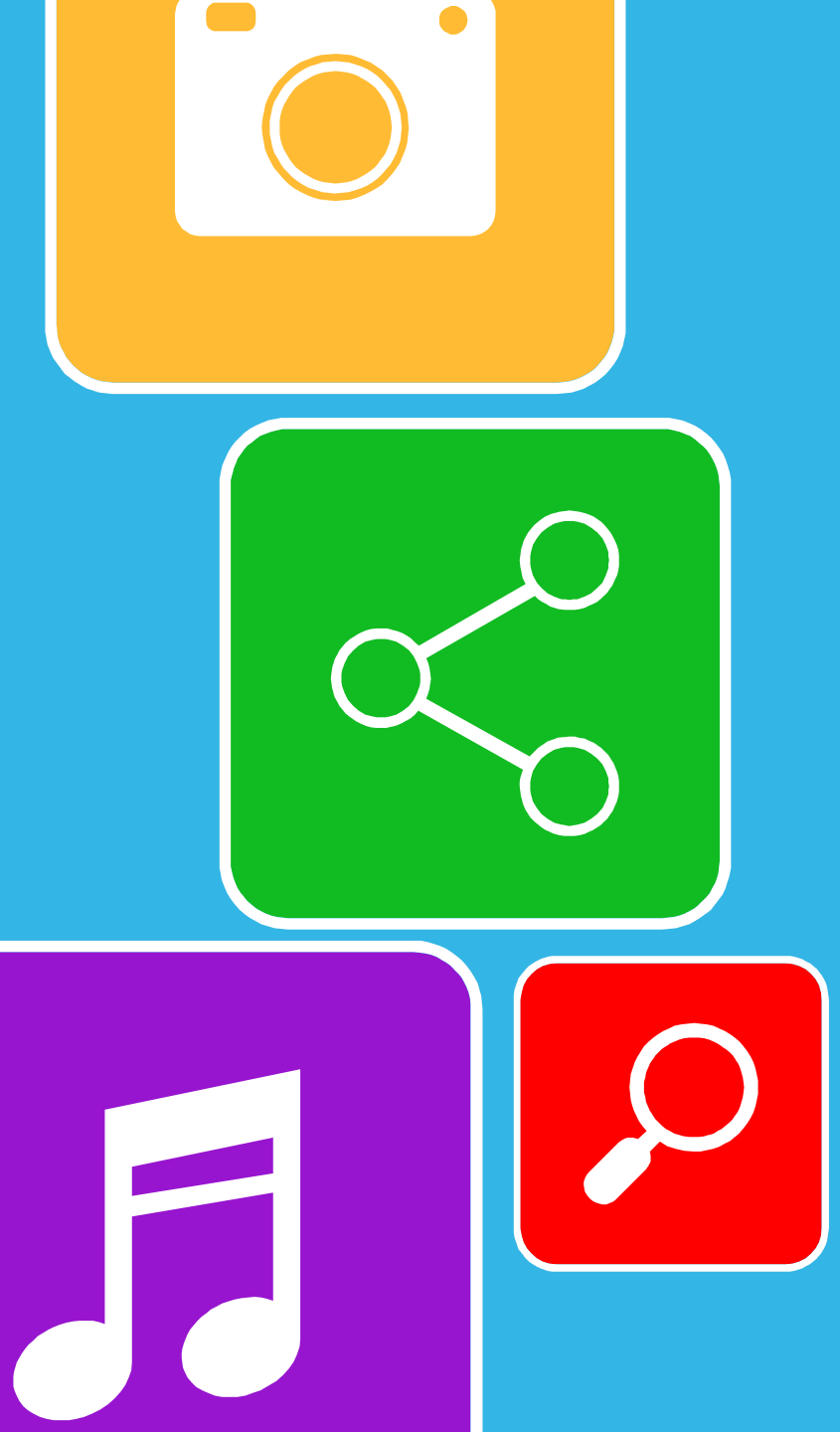


TensorFlow Tutorial2

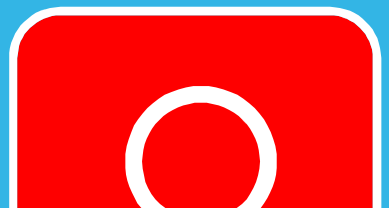
2016.12.23

Yunjey Choi, MS Student

DAVIAN Lab (led by Prof. Jaegul Choo)

Korea University



Contents

- Review
- FFNN with Embedding
- Convolutional Neural Network
- Recurrent Neural Network

01 Review

Review: Random Normal

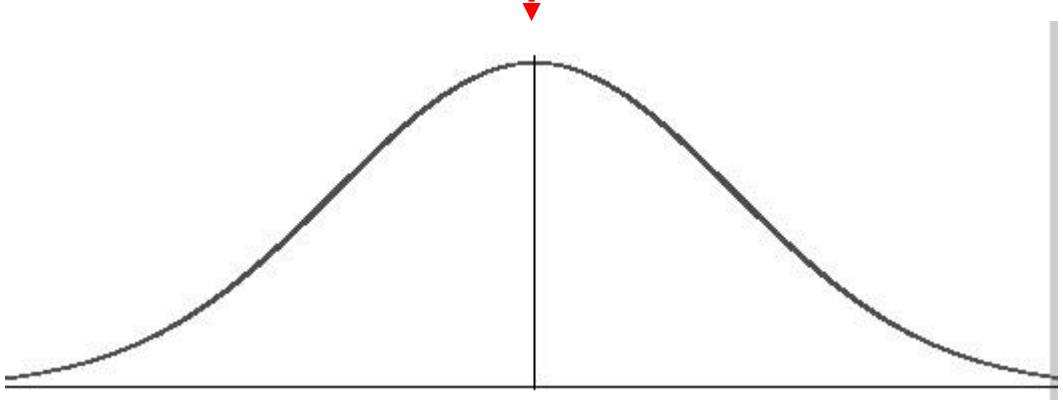
```
normal = tf.random_normal([2, 3], stddev=0.1)
print normal.eval()
```

```
[[ 0.04854904 -0.06484753  0.01029386]
 [-0.03241342  0.09126773  0.15588009]]
```

Review: Random Normal

```
normal = tf.random_normal([2, 3], stddev=0.1)
print normal.eval()
```

```
[[ 0.04854904 -0.06484753  0.01029386]
 [-0.03241342  0.09126773  0.15588009]]
```



Review: Random Normal

```
normal = tf.random_normal([2, 3], stddev=0.1)
print normal.eval()
```

```
[[ 0.04854904 -0.06484753  0.01029386]
 [-0.03241342  0.09126773  0.15588009]]
```

matrix of shape (2, 3)

Review: Variables

변수를 생성하는 첫 번째 방법 (또 다른 방법은 뒤에서 설명)

```
# variable will be initialized with normal distribution
var = tf.Variable(tf.random_normal([2, 3], stddev=0.1), name='my_var')
print var.op.name
tf.initialize_all_variables().run()
print var.eval()
```

```
my_var
[[-0.09178112  0.14630298 -0.00147568]
 [ 0.0576584  -0.01467518 -0.02173685]]
```

Review: Variables

```
# variable will be initialized with normal distribution
var = tf.Variable(tf.random_normal([2, 3], stddev=0.1), name='my_var')  변수를 생성.
print var.op.name
tf.initialize_all_variables().run()
print var.eval()
```

```
my_var
[[-0.09178112  0.14630298 -0.00147568]
 [ 0.0576584  -0.01467518 -0.02173685]]
```

Review: Variables

```
# variable will be initialized with normal distribution
var = tf.Variable(tf.random_normal([2, 3], stddev=0.1), name='my_var')
print var.op.name
tf.initialize_all_variables().run()
print var.eval()
```

변수의 값을 보려면 항상 초기화하는 연산을 먼저 실행해야한다.

```
my_var
[[-0.09178112  0.14630298 -0.00147568]
 [ 0.0576584  -0.01467518 -0.02173685]]
```

Review: Variables

변수를 생성.

```
# variable will be initialized with normal distribution
var = tf.Variable(tf.random_normal([2, 3], stddev=0.1), name='my_var')
print var.op.name
tf.initialize_all_variables().run()
print var.eval()
```

변수의 값을 보려면 항상 초기화하는 연산을 먼저 실행해야한다.

```
my_var
[[-0.09178112  0.14630298 -0.00147568]
 [ 0.0576584  -0.01467518 -0.02173685]]
```

Review: Variables

Variable will be initialized with random normal distribution

```
# variable will be initialized with normal distribution
var = tf.Variable(tf.random_normal([2, 3], stddev=0.1), name='my_var')
print var.op.name
tf.initialize_all_variables().run()
print var.eval()
```

```
my_var
[[-0.09178112  0.14630298 -0.00147568]
 [ 0.0576584  -0.01467518 -0.02173685]]
```

Review: Variables

```
# variable will be initialized with normal distribution
var = tf.Variable(tf.random_normal([2, 3], stddev=0.1), name='my_var')
print var.op.name
tf.initialize_all_variables().run()
print var.eval()
```

variable name

```
my_var
[[-0.09178112  0.14630298 -0.00147568]
 [ 0.0576584  -0.01467518 -0.02173685]]
```

Review: Variables

```
# variable will be initialized with normal distribution
var = tf.Variable(tf.random_normal([2, 3], stddev=0.1), name='my_var')
print var.op.name
tf.initialize_all_variables().run()
print var.eval()
```

Initialize variable

```
my_var
[[-0.09178112  0.14630298 -0.00147568]
 [ 0.0576584  -0.01467518 -0.02173685]]
```

Review: Variables

변수를 생성하는 두 번째 방법

```
v = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.01))
tf.initialize_all_variables().run()
print v.eval()
```

```
[[-0.00565953  0.00716706 -0.02132921]
 [-0.00839901  0.00784649 -0.00300771]]
```

Review: Variables

Variable name

```
v = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.01))
tf.initialize_all_variables().run()
print v.eval()
```

```
[[-0.00565953  0.00716706 -0.02132921]
 [-0.00839901  0.00784649 -0.00300771]]
```

Review: Variables

create a matrix of shape (2, 3)

```
v = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.01))
tf.initialize_all_variables().run()
print v.eval()
```

```
[[-0.00565953  0.00716706 -0.02132921]
 [-0.00839901  0.00784649 -0.00300771]]
```

Review: Variables

변수를 생성할 때 초기화하는 함수

```
v = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.01))
tf.initialize_all_variables().run()
print v.eval()
```

```
[[-0.00565953  0.00716706 -0.02132921]
 [-0.00839901  0.00784649 -0.00300771]]
```

Review: Variable Scope

variable scope: 변수 이름을 효율적으로 관리할 수 있게 해준다

```
with tf.variable_scope('layer1'):
    w = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.1))
    print w.op.name

with tf.variable_scope('layer2'):
    w = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.1))
    print w.op.name
```

```
layer1/v
layer2/v
```

Review: Variable Scope

```
with tf.variable_scope('layer1'):
    w = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.1))
    print w.op.name

with tf.variable_scope('layer2'):
    w = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.1))
    print w.op.name
```

```
layer1/v
layer2/v
```

변수 이름 앞에 layer1/ 이 추가 됨

Review: Variable Scope

```
with tf.variable_scope('layer1'):
    w = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.1))
    print w.op.name

with tf.variable_scope('layer2'):
    w = tf.get_variable('v', shape=[2, 3], initializer=tf.random_normal_initializer(stddev=0.1))
    print w.op.name
```

```
layer1/v
layer2/v
```

변수 이름 앞에 layer2/ 이 추가 됨

Review: Reuse Variables

Reuse Variables: 이미 생성한 변수를 불러오고 싶을 때 사용

```
with tf.variable_scope('layer1', reuse=True):
    w = tf.get_variable('v')   # Unlike above, we don't need to specify shape and initializer
    print w.op.name

# or

with tf.variable_scope('layer1') as scope:
    scope.reuse_variables()
    w = tf.get_variable('v')
    print w.op.name
```

```
layer1/v
layer1/v
```

Review: Reuse Variables

첫 번째 방법

```
with tf.variable_scope('layer1', reuse=True):
    w = tf.get_variable('v')   # Unlike above, we don't need to specify shape and initializer
    print w.op.name

# or

with tf.variable_scope('layer1') as scope:
    scope.reuse_variables()
    w = tf.get_variable('v')
    print w.op.name
```

```
layer1/v
layer1/v
```

Review: Reuse Variables

```
with tf.variable_scope('layer1', reuse=True):
    w = tf.get_variable('v')   # Unlike above, we don't need to specify shape and initializer
    print w.op.name

# or

with tf.variable_scope('layer1') as scope:
    scope.reuse_variables()
    w = tf.get_variable('v')
    print w.op.name
```

두 번째 방법

```
layer1/v
layer1/v
```

Review: Reuse Variables

생성할 때와는 달리 shape과 initializer없이 name만 기입하면 된다

```
with tf.variable_scope('layer1', reuse=True):
    w = tf.get_variable('v')   # Unlike above, we don't need to specify shape and initializer
    print w.op.name

# or

with tf.variable_scope('layer1') as scope:
    scope.reuse_variables()
    w = tf.get_variable('v')
    print w.op.name
```

```
layer1/v
layer1/v
```

Review: Place Holder

TensorFlow 자료형	Neural Network
Variable	Parameter 혹은 Weight
Placeholder	Input 과 Output

Placeholder는 input과 output을 담을 그릇 역할을 해준다

```
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])
```

Review: Place Holder

TensorFlow 자료형	Neural Network
Variable	Parameter 혹은 Weight
Placeholder	Input 과 Output

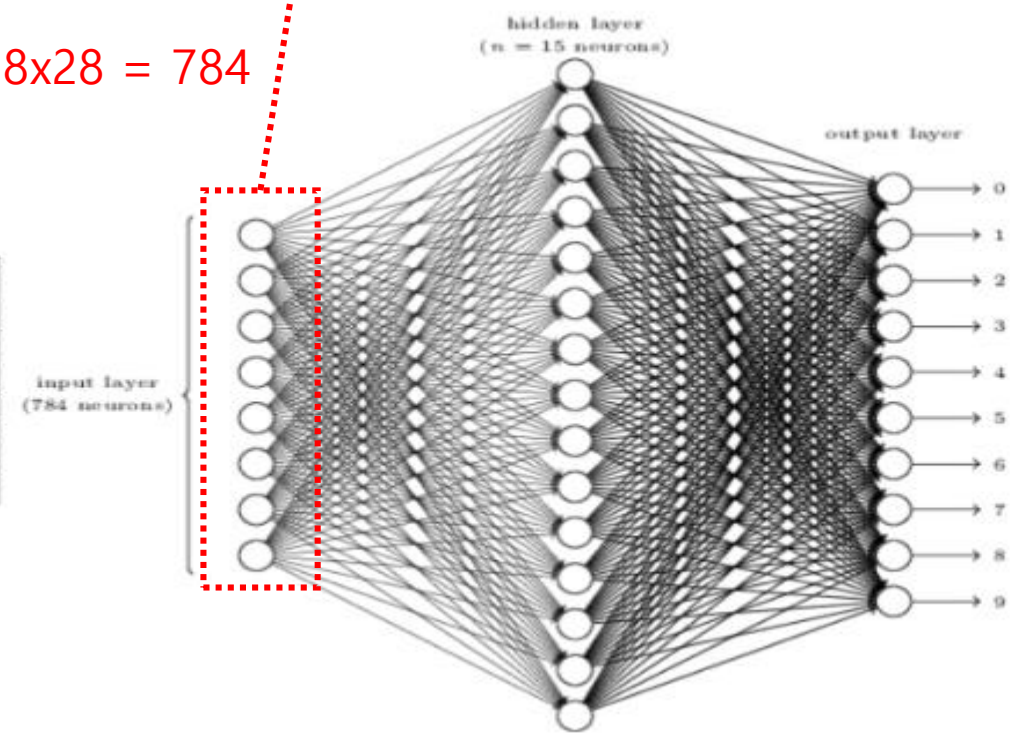
None은 variable batch size를 위해서 사용

```
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])
```

Review: Place Holder

```
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])
```

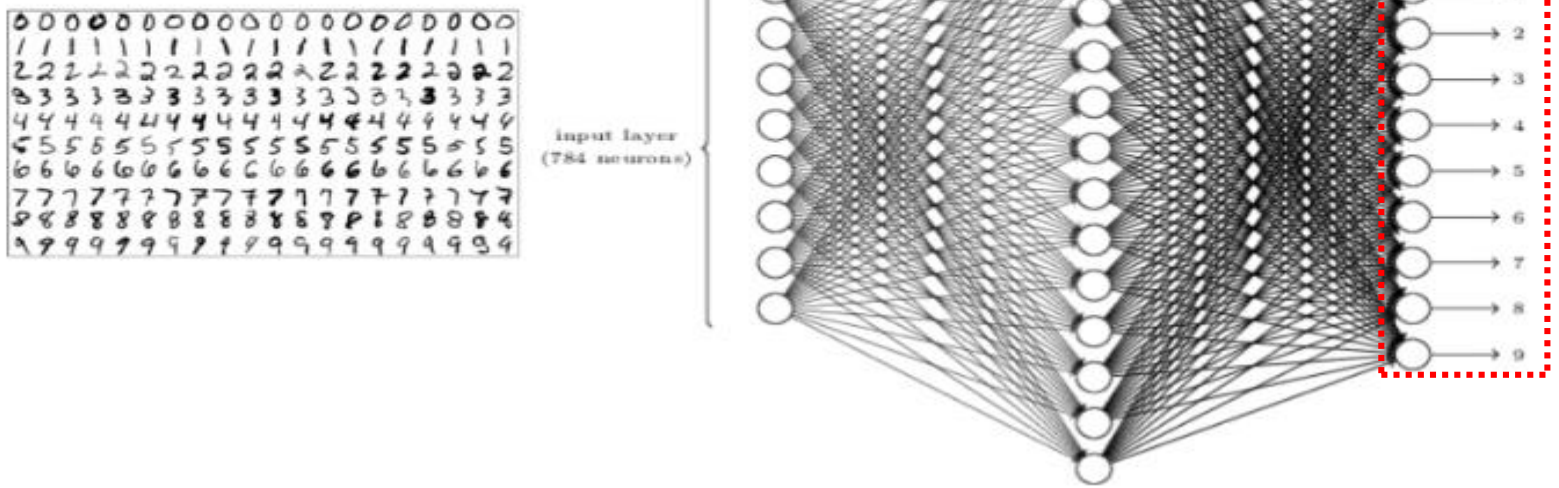
Dimension of input = 28x28 = 784



Review: Place Holder

```
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])
```

Dimension of output =10



02 FFNN with Embedding

Titanic dataset

PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450
6	0	3	Moran, Mr. James	male	28	0	0	330877
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736

Target Class

1 if that person survives and 0... if they do not :(

PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450
6	0	3	Moran, Mr. James	male	28	0	0	330877
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736

Categorical Features

Passenger class

Ticket number

PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450
6	0	3	Moran, Mr. James	male	28	0	0	330877
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736

Categorical Features

Continuous Features

SibSp: Number of Siblings/Spouses Aboard

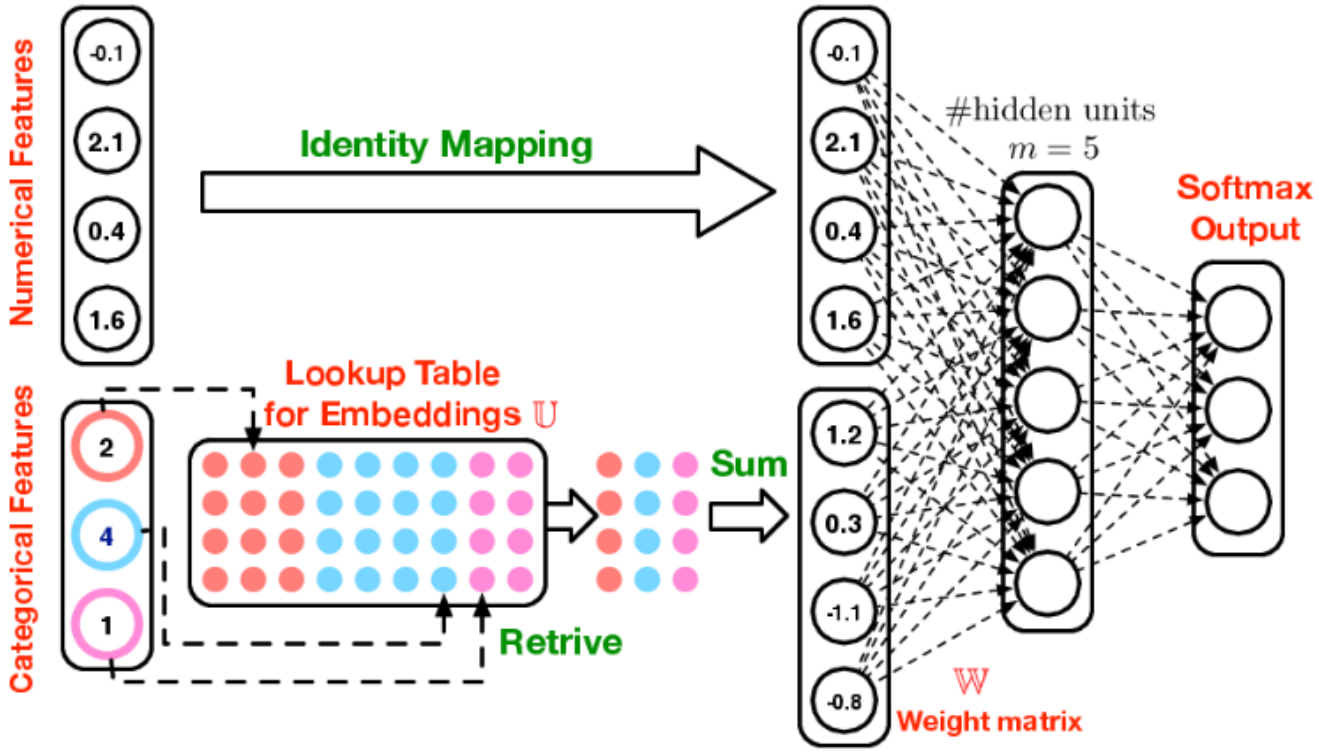
Continuous Features

Parch: Number of Parents/Children Aboard

PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450
6	0	3	Moran, Mr. James	male	28	0	0	330877
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736

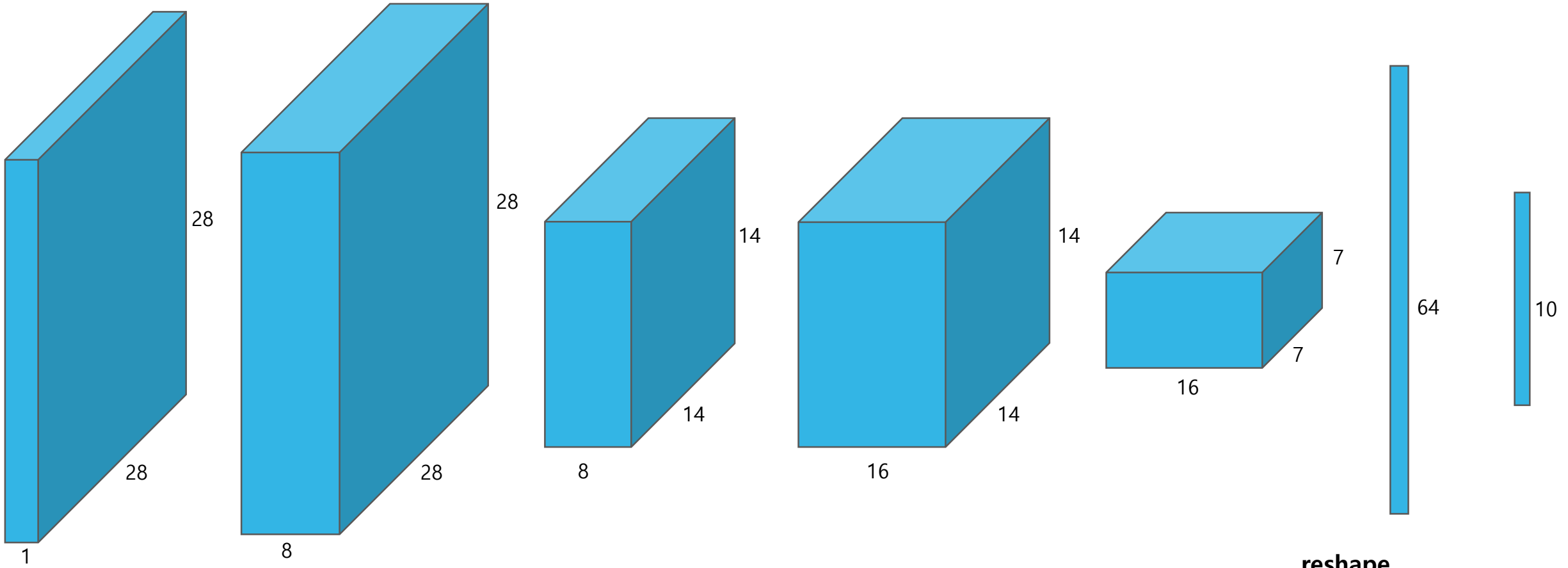
Neural Network with Categorical Features

Neural network learns a numerical embedding for each category of a categorical feature, based on which we can visualize all categories in the embedding space and extract knowledge of similarity between categories



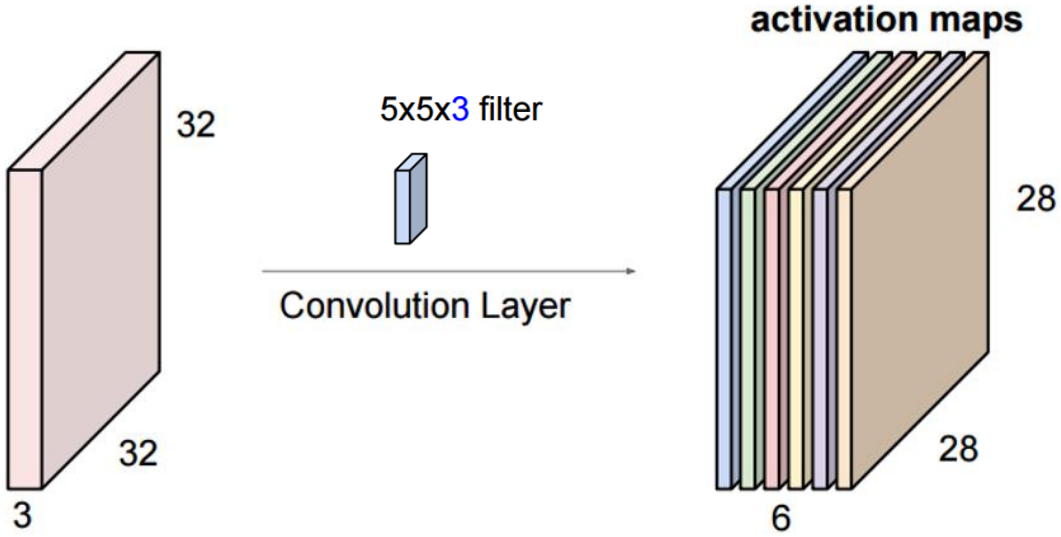
03 Convolutional Neural Network

Convolutional Neural Network



TensorFlow Implementation

Convolution Layer

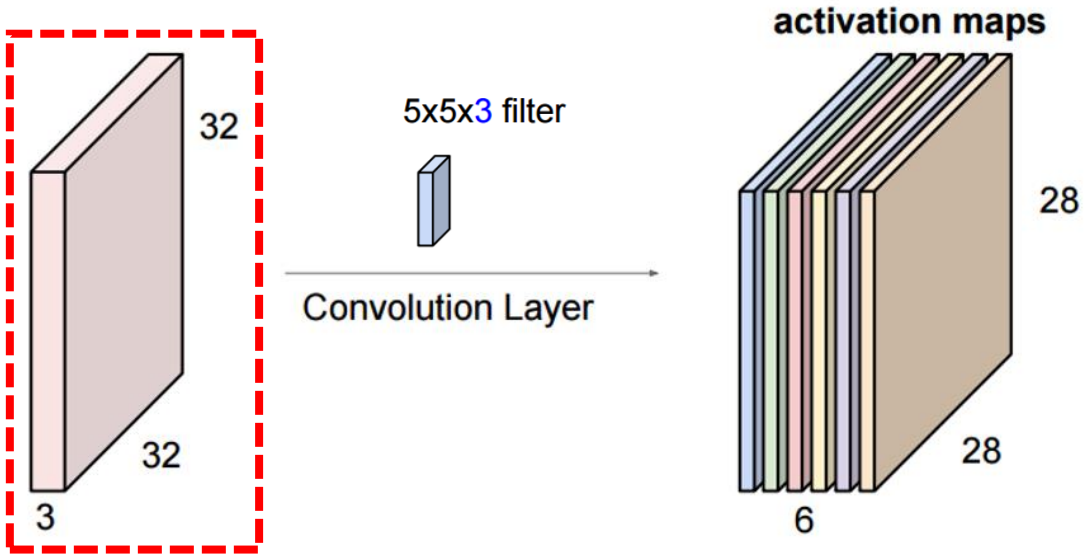



```
weight = tf.Variable(tf.truncated_normal(shape=[5, 5, 3, 6]))
bias = tf.Variable(tf.zeros(shape=[32]))

conv = tf.nn.conv2d(input_image, weight, [1, 1, 1, 1], 'VALID') + bias
```

TensorFlow Implementation

Convolution Layer

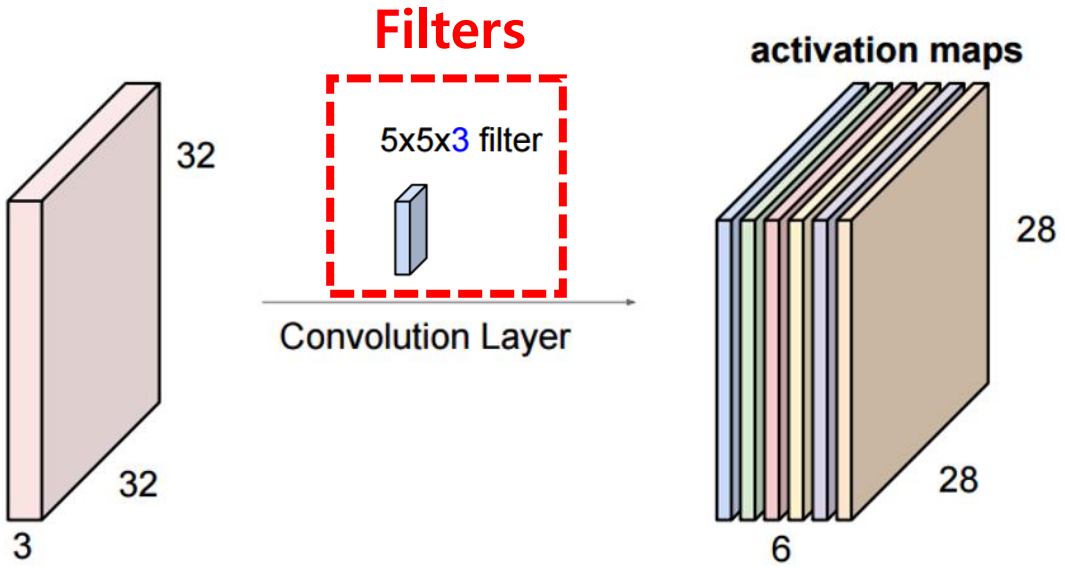



```
weight = tf.Variable(tf.truncated_normal(shape=[5, 5, 3, 6]))

conv = tf.nn.conv2d(input_image, weight, [1, 1, 1, 1], 'VALID')
```

TensorFlow Implementation

Convolution Layer

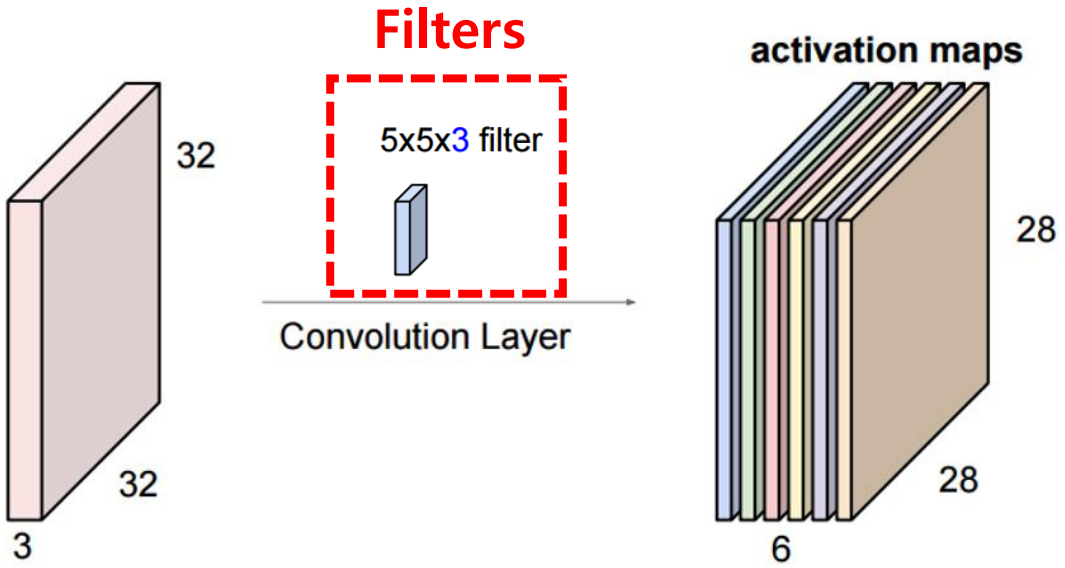



```
weight = tf.Variable(tf.truncated_normal(shape=[5, 5, 3, 6]))

conv = tf.nn.conv2d(input_image, weight, [1, 1, 1, 1], 'VALID')
```

TensorFlow Implementation

Convolution Layer

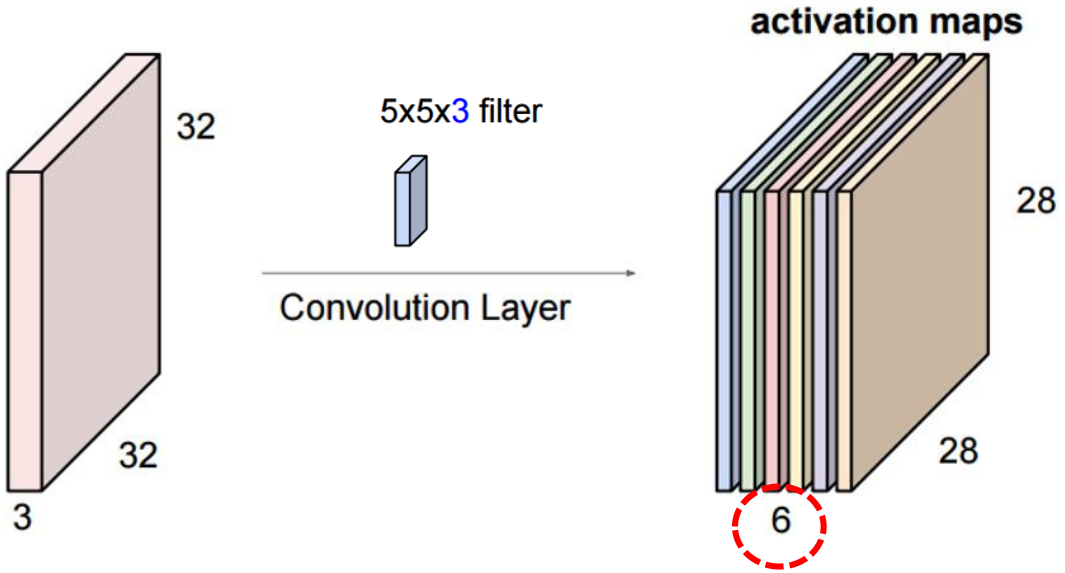



```
weight = tf.Variable(tf.truncated_normal(shape=[5, 5, 3, 6]))

conv = tf.nn.conv2d(input_image, weight, [1, 1, 1, 1], 'VALID')
```

TensorFlow Implementation

Convolution Layer



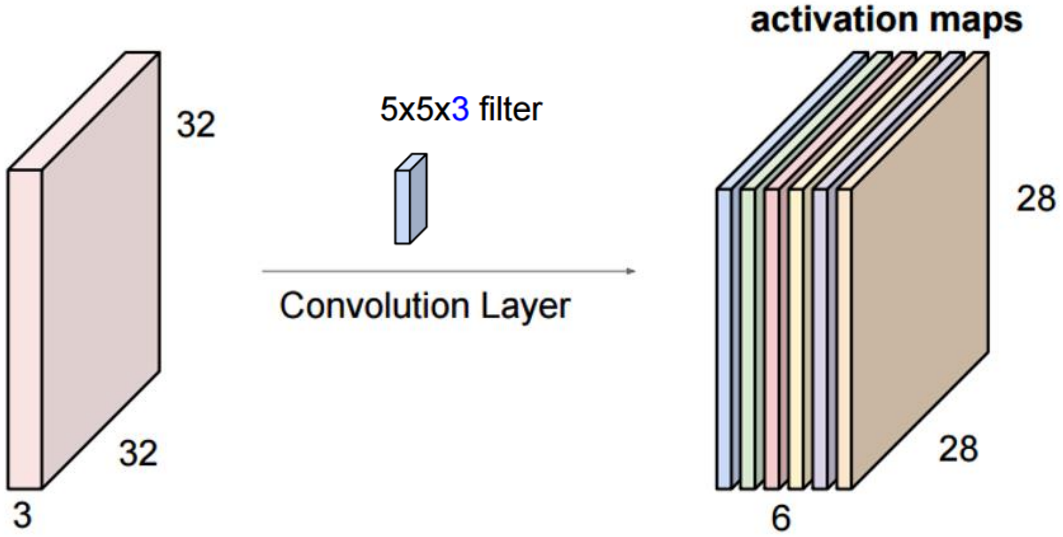
사용할 Filter 개수

```
weight = tf.Variable(tf.truncated_normal(shape=[5, 5, 3, 6]))

conv = tf.nn.conv2d(input_image, weight, [1, 1, 1, 1], 'VALID')
```

TensorFlow Implementation

Convolution Layer



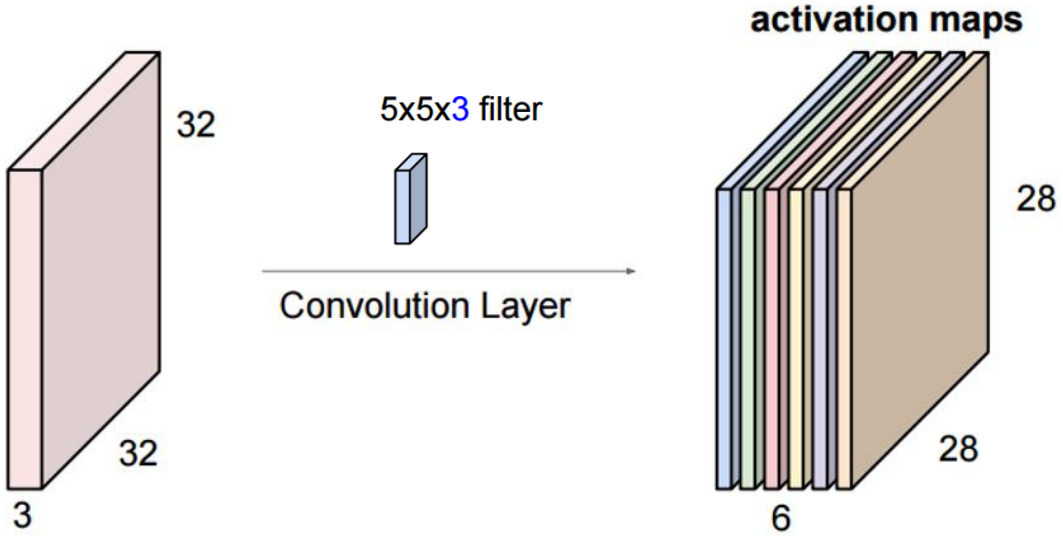
stride size (batch, width, height, channel)

```
weight = tf.Variable(tf.truncated_normal(shape=[5, 5, 3, 6]))

conv = tf.nn.conv2d(input_image, weight, [1, 1, 1, 1], 'VALID')
```

TensorFlow Implementation

Convolution Layer



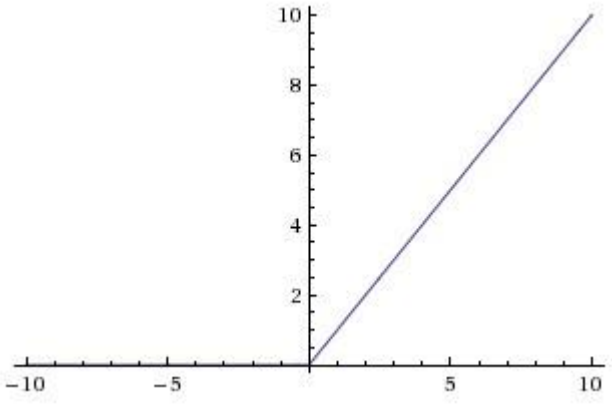
VALID: padding x
SAME: padding o

```
weight = tf.Variable(tf.truncated_normal(shape=[5, 5, 3, 6]))

conv = tf.nn.conv2d(input_image, weight, [1, 1, 1, 1], 'VALID')
```

TensorFlow Implementation

Rectified Linear Unit (ReLU)



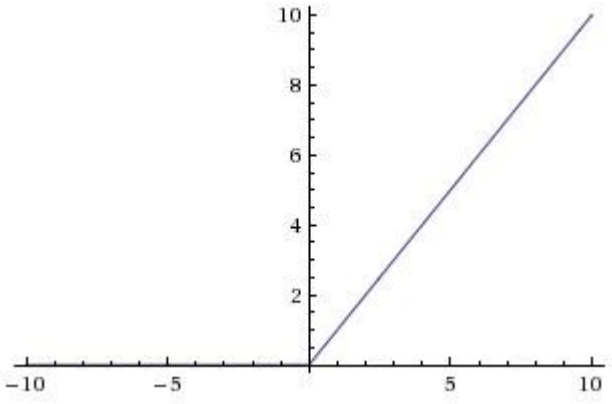
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

```
relu = tf.nn.relu(conv)
```

TensorFlow Implementation

Rectified Linear Unit (ReLU)



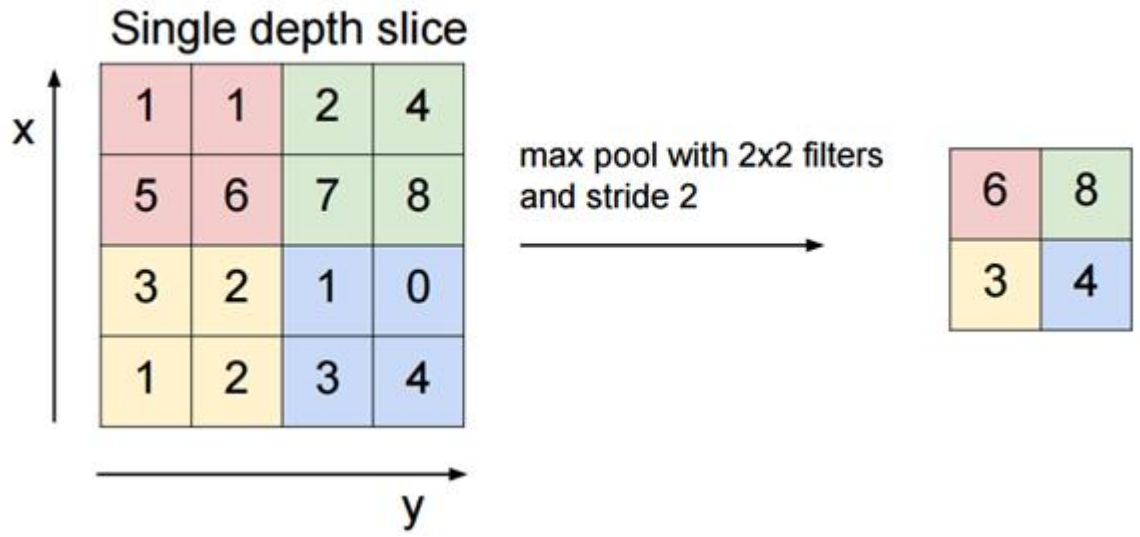
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

```
relu = tf.nn.relu(conv)
```

TensorFlow Implementation

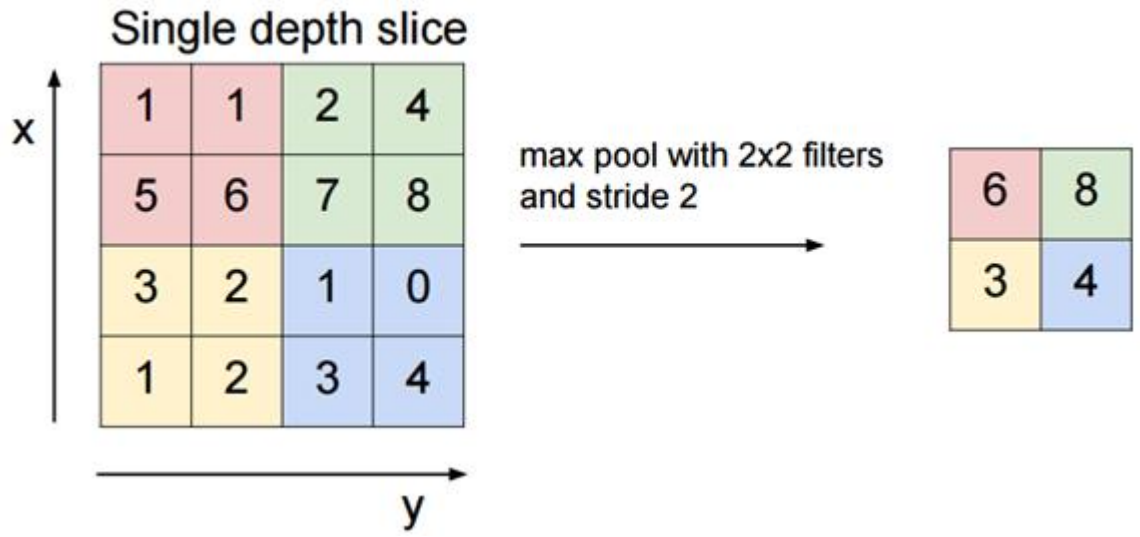
Max Pooling




```
pool = tf.nn.max_pool(relu, [1, 2, 2, 1], [1, 2, 2, 1], 'VALID')
```

TensorFlow Implementation

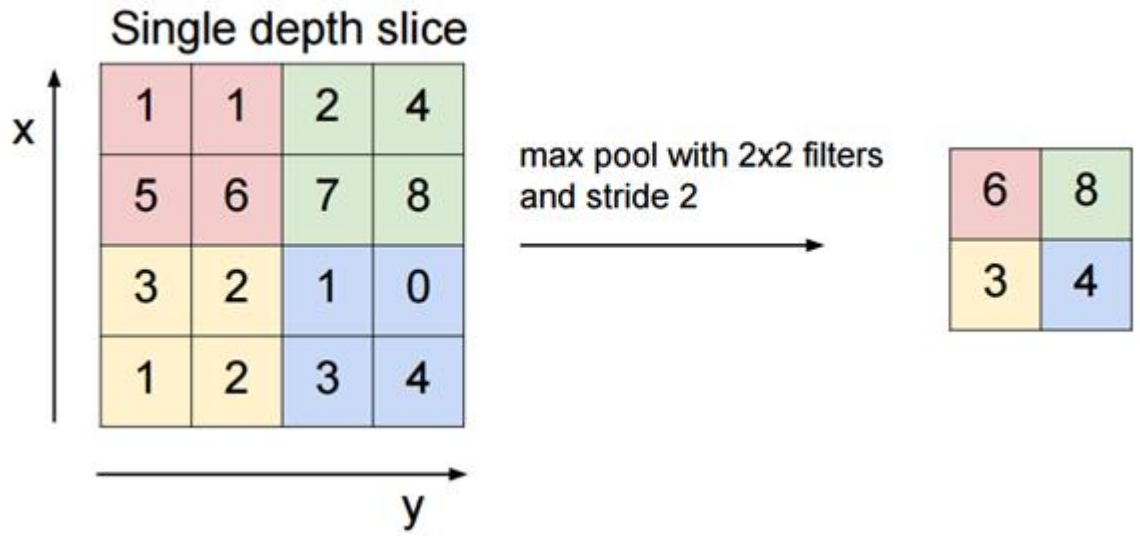
Max Pooling




```
pool = tf.nn.max_pool(relu, [1, 2, 2, 1], [1, 2, 2, 1], 'VALID')
```

TensorFlow Implementation

Max Pooling

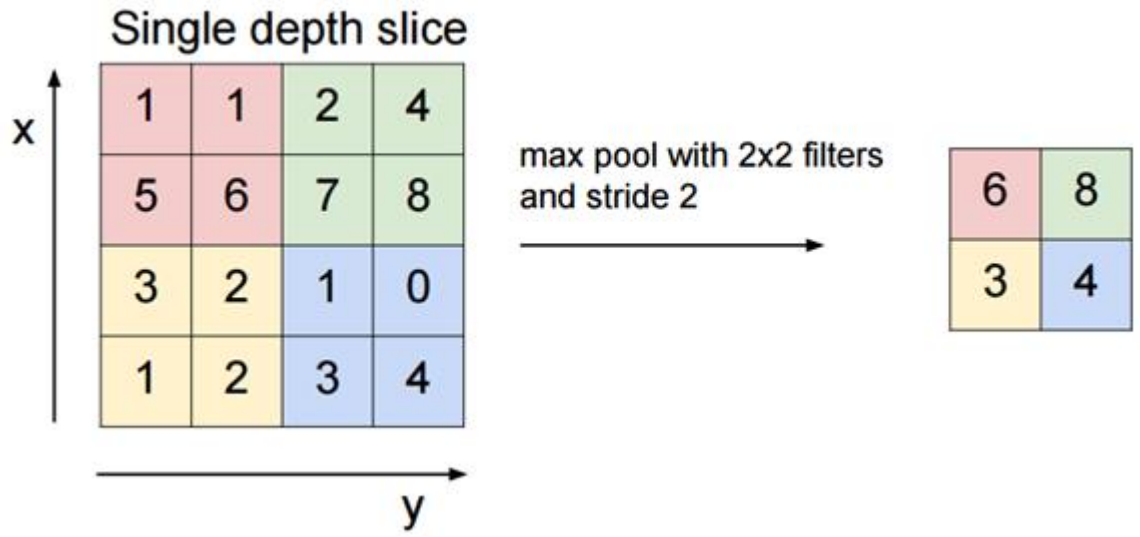


filter size (batch, width, height, channel)

```
pool = tf.nn.max_pool(relu, [1, 2, 2, 1], [1, 2, 2, 1], 'VALID')
```

TensorFlow Implementation

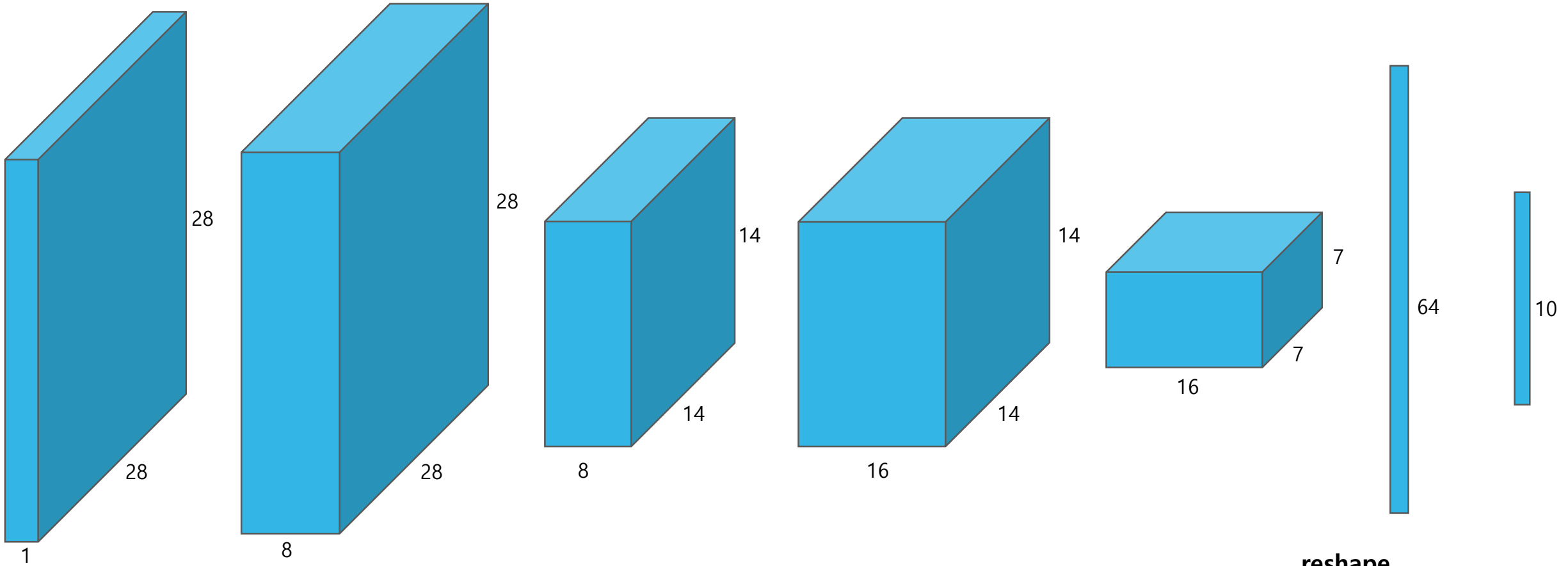
Max Pooling



stride size (batch, width, height, channel)

```
pool = tf.nn.max_pool(relu, [1, 2, 2, 1], [1, 2, 2, 1], 'VALID')
```

Convolutional Neural Network

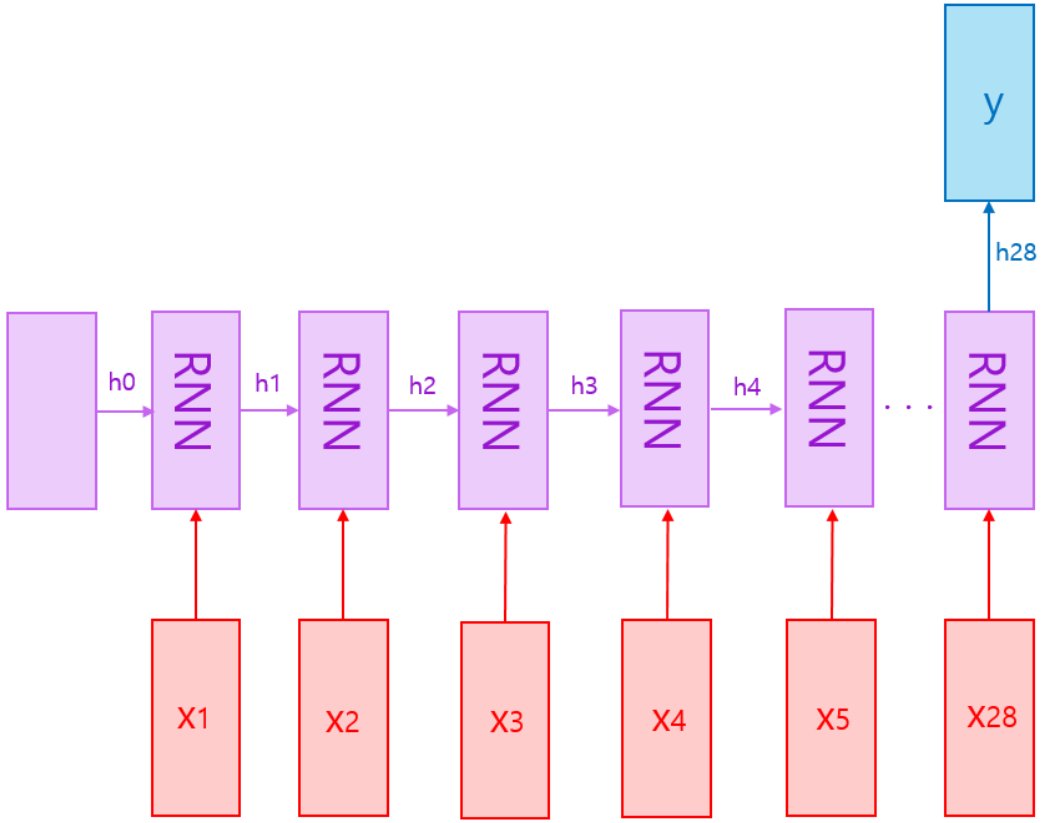


Convolutional Neural Network

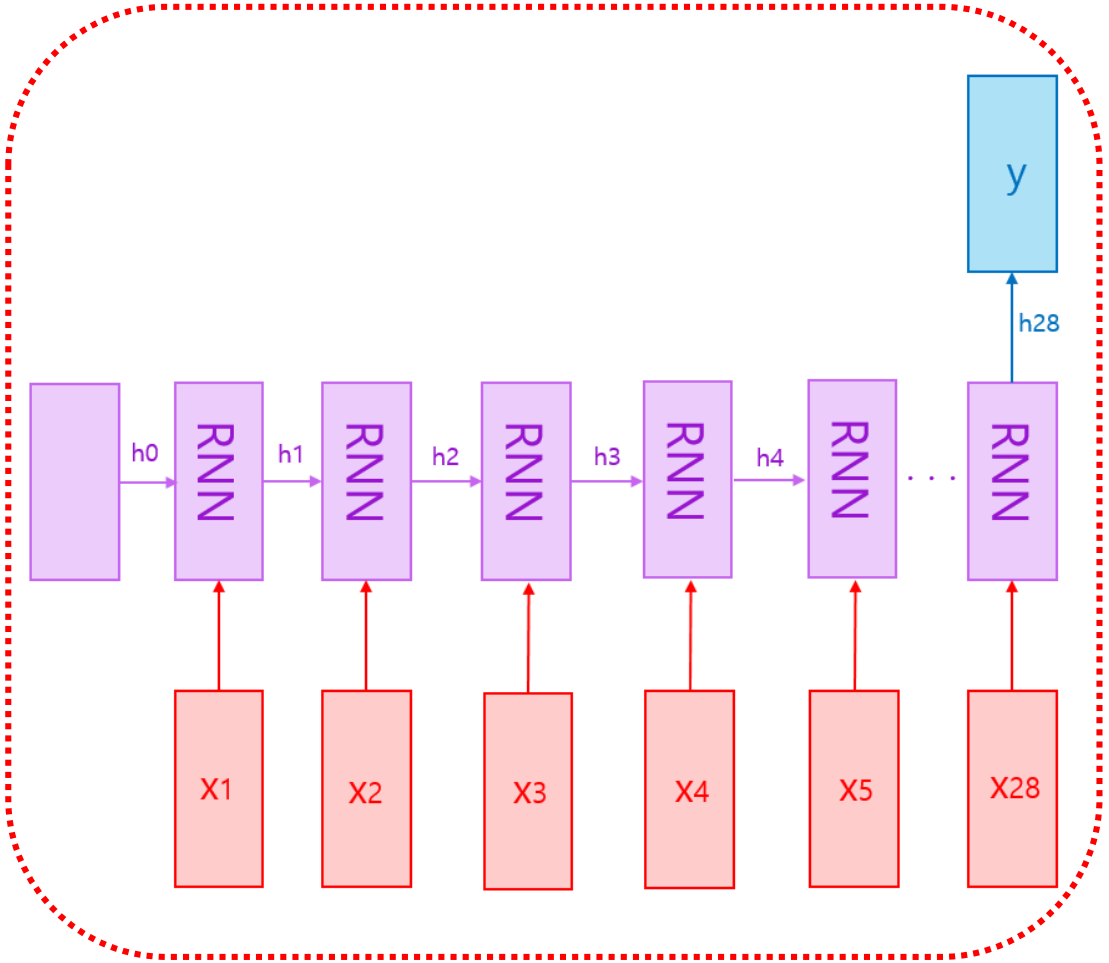
실습

04 Recurrent Neural Network

Recurrent Neural Network

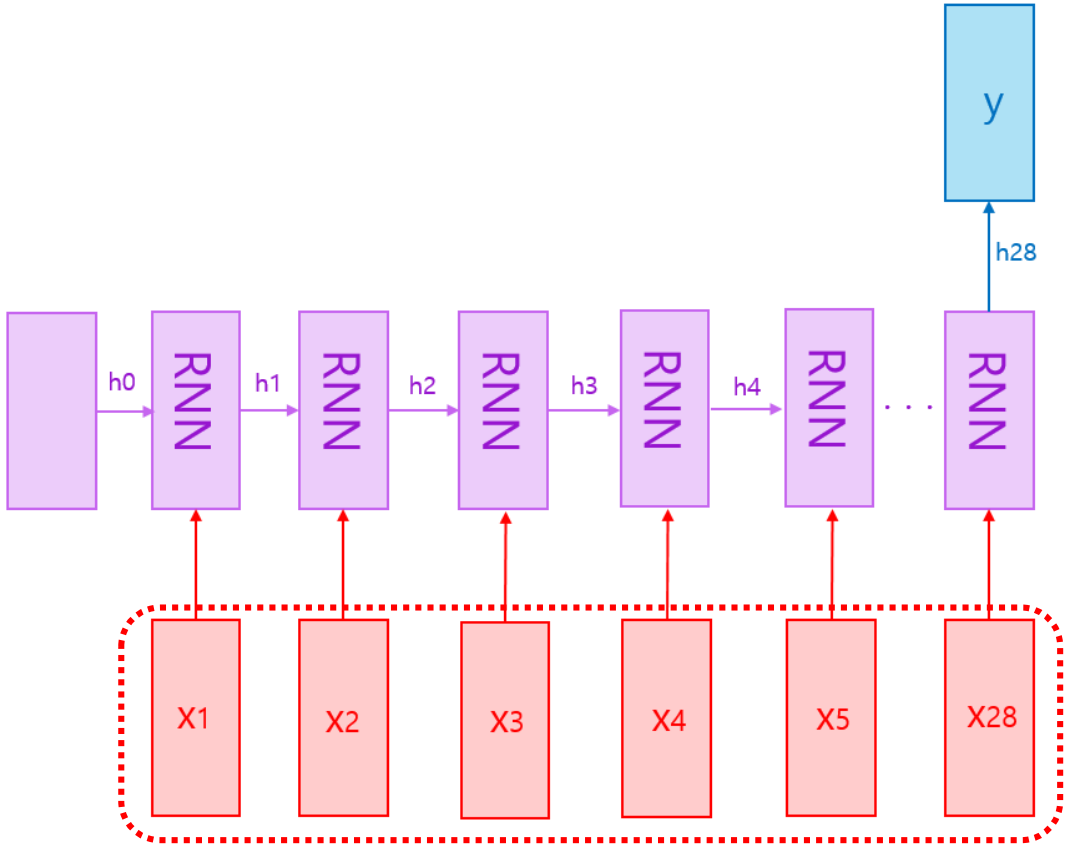


Recurrent Neural Network



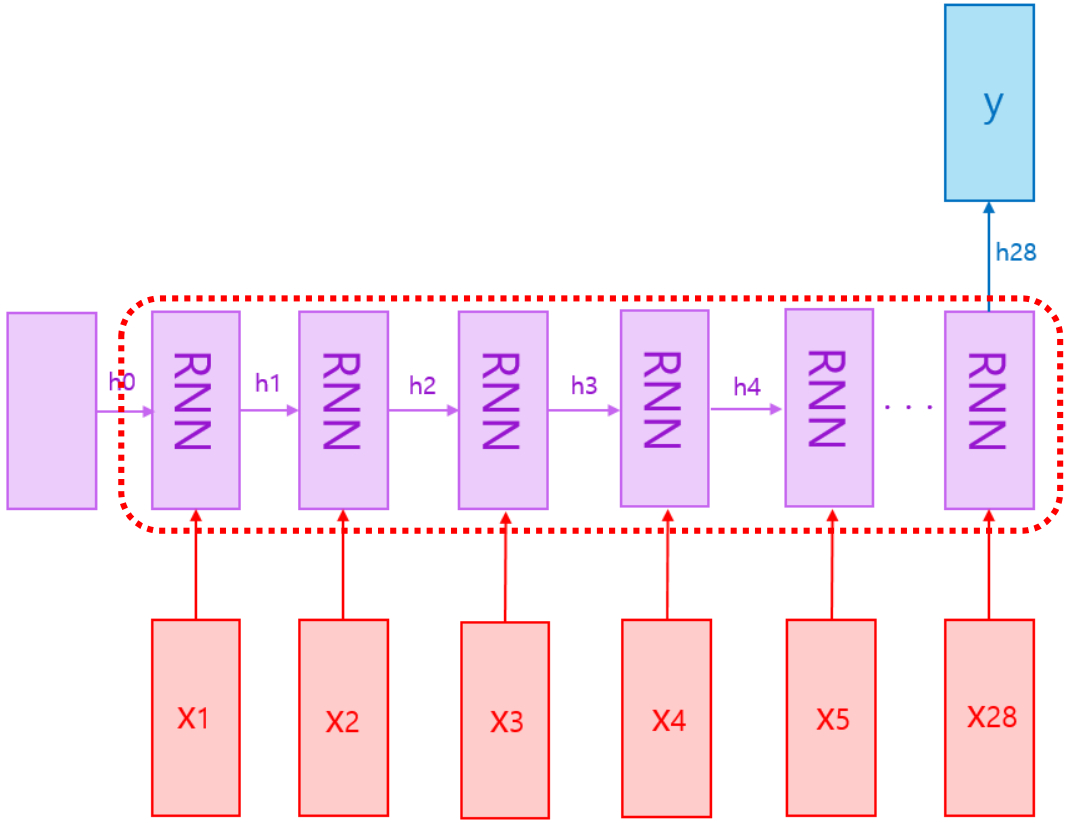
Recurrent Neural Network (many to one)

Recurrent Neural Network



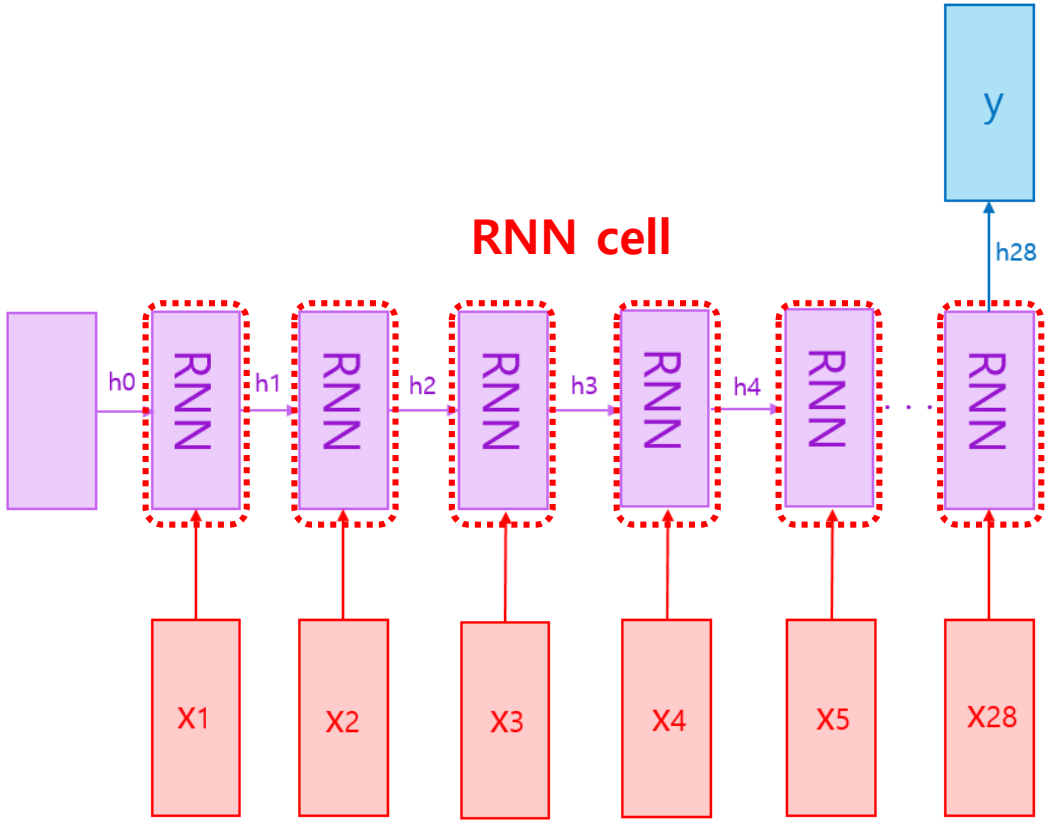
sequence of input vectors (many)

Recurrent Neural Network

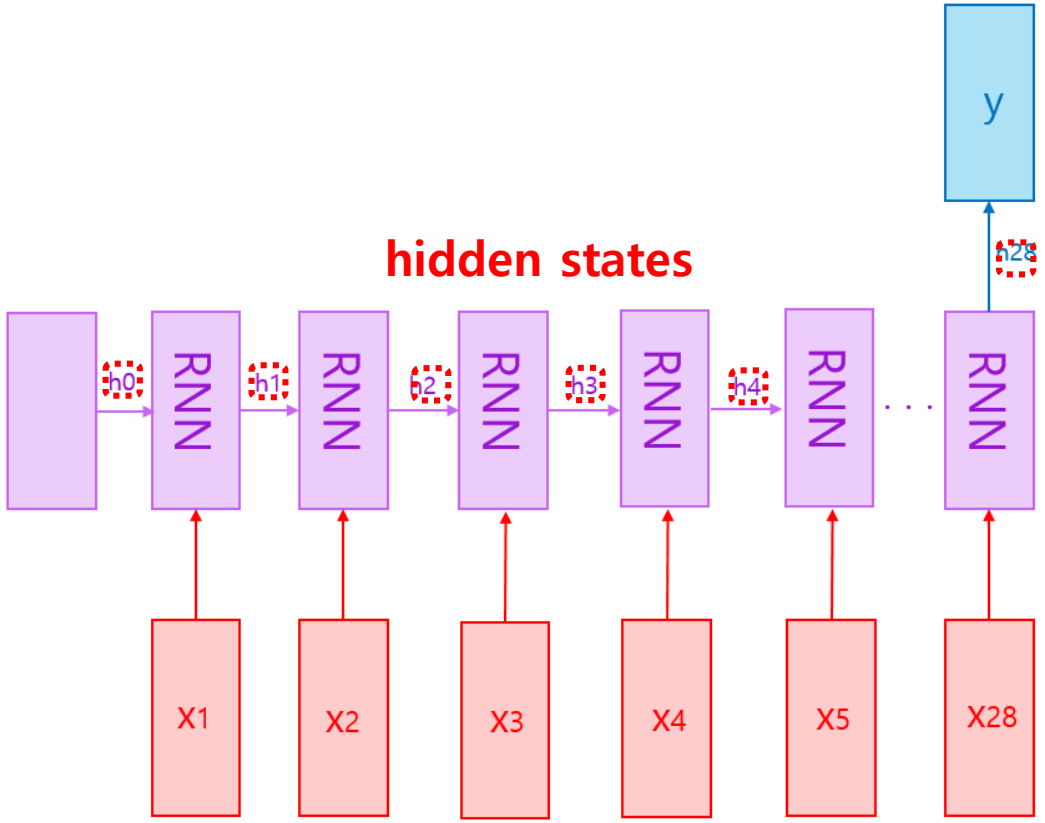


RNN cell and hidden states

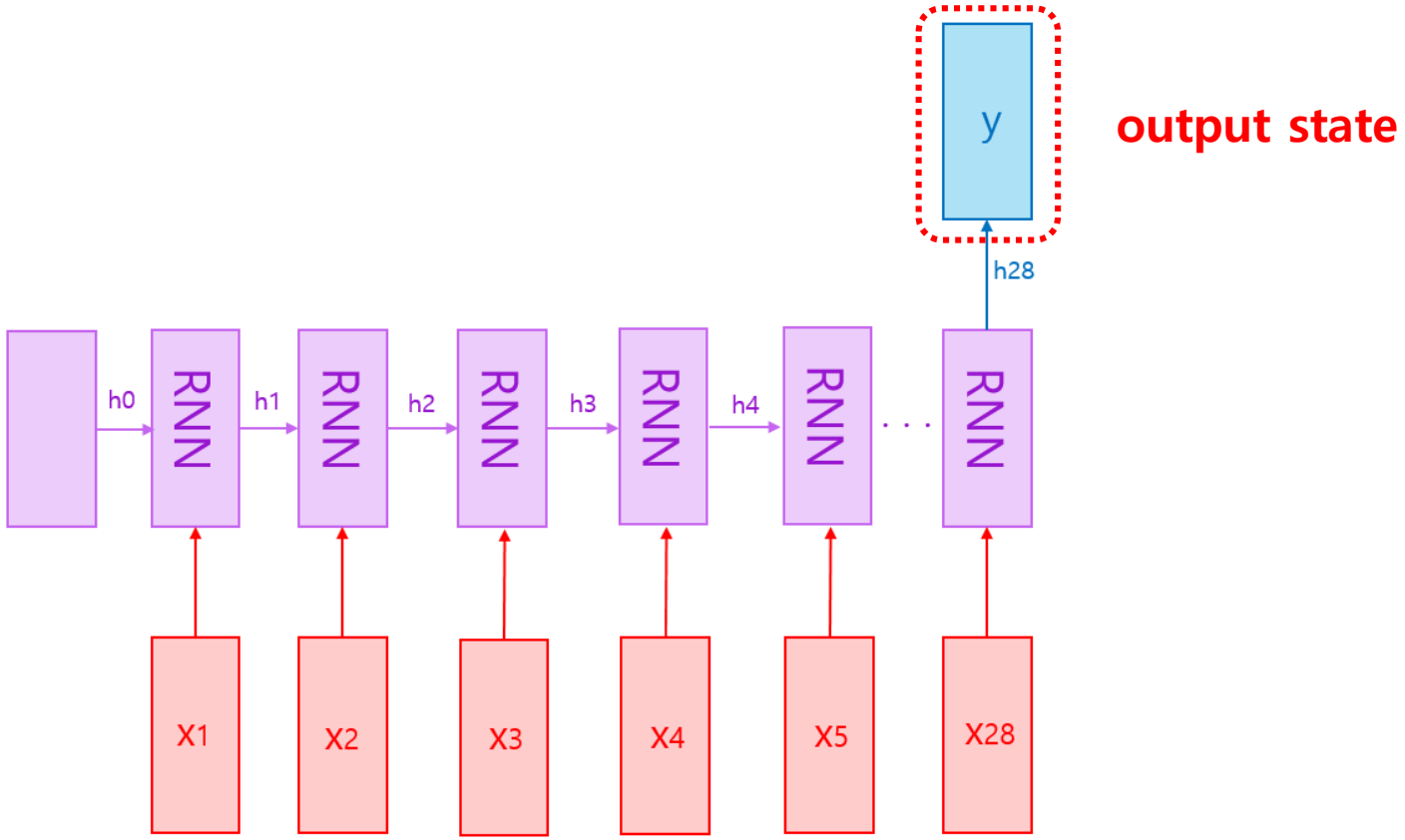
Recurrent Neural Network



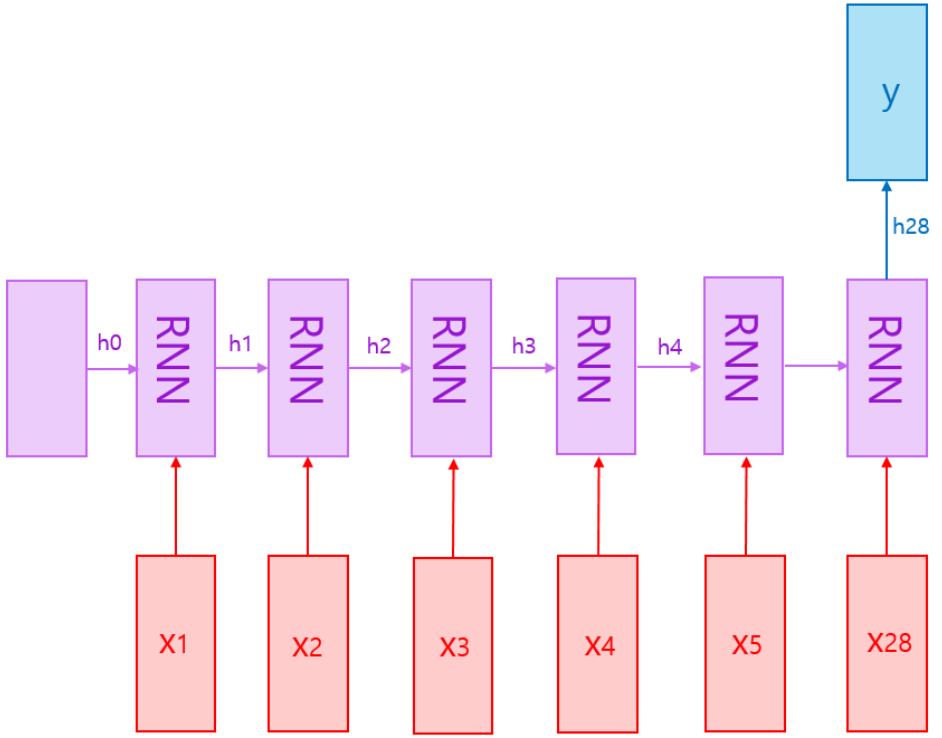
Recurrent Neural Network



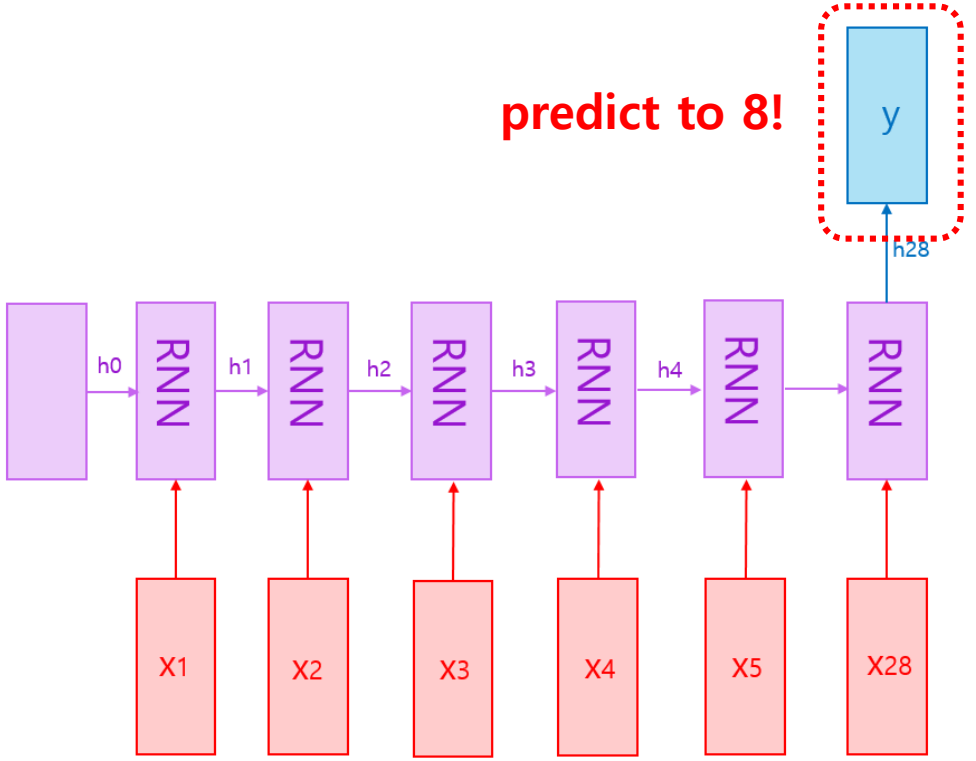
Recurrent Neural Network



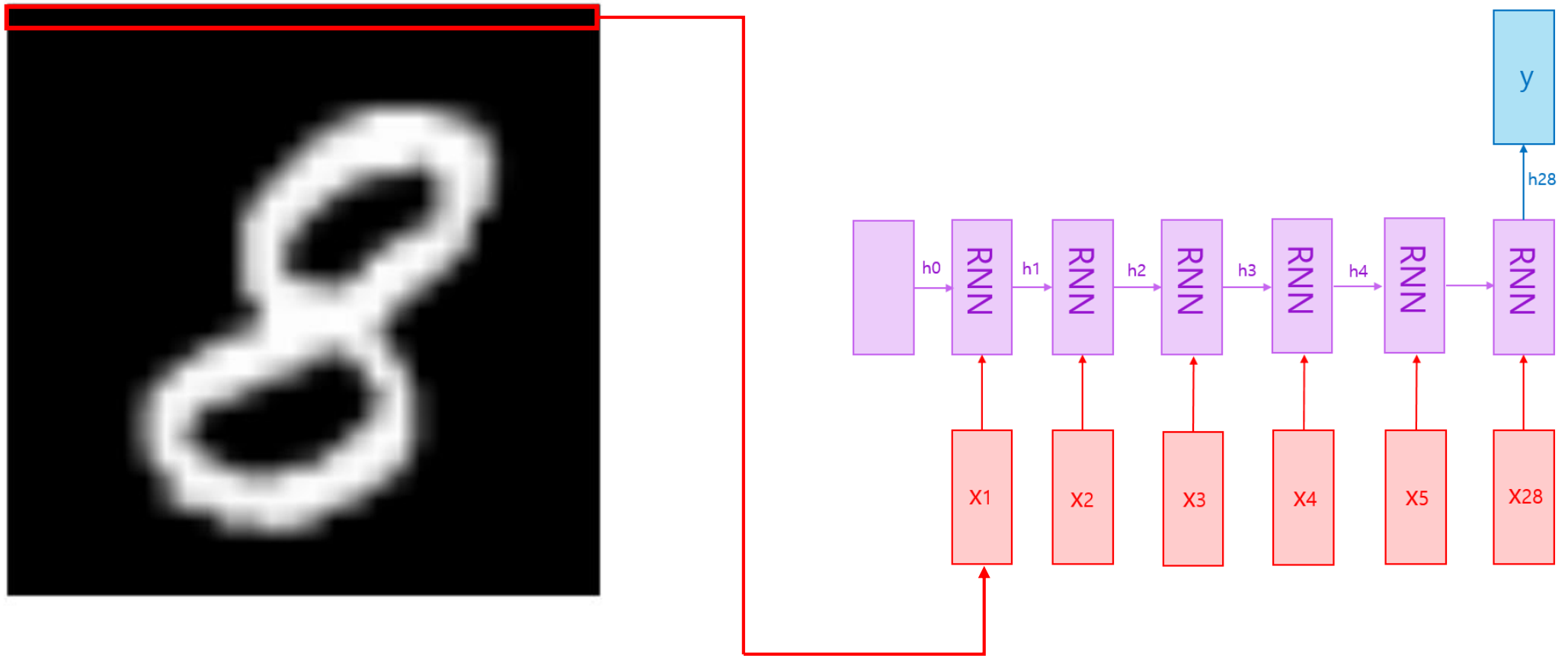
Recurrent Neural Network



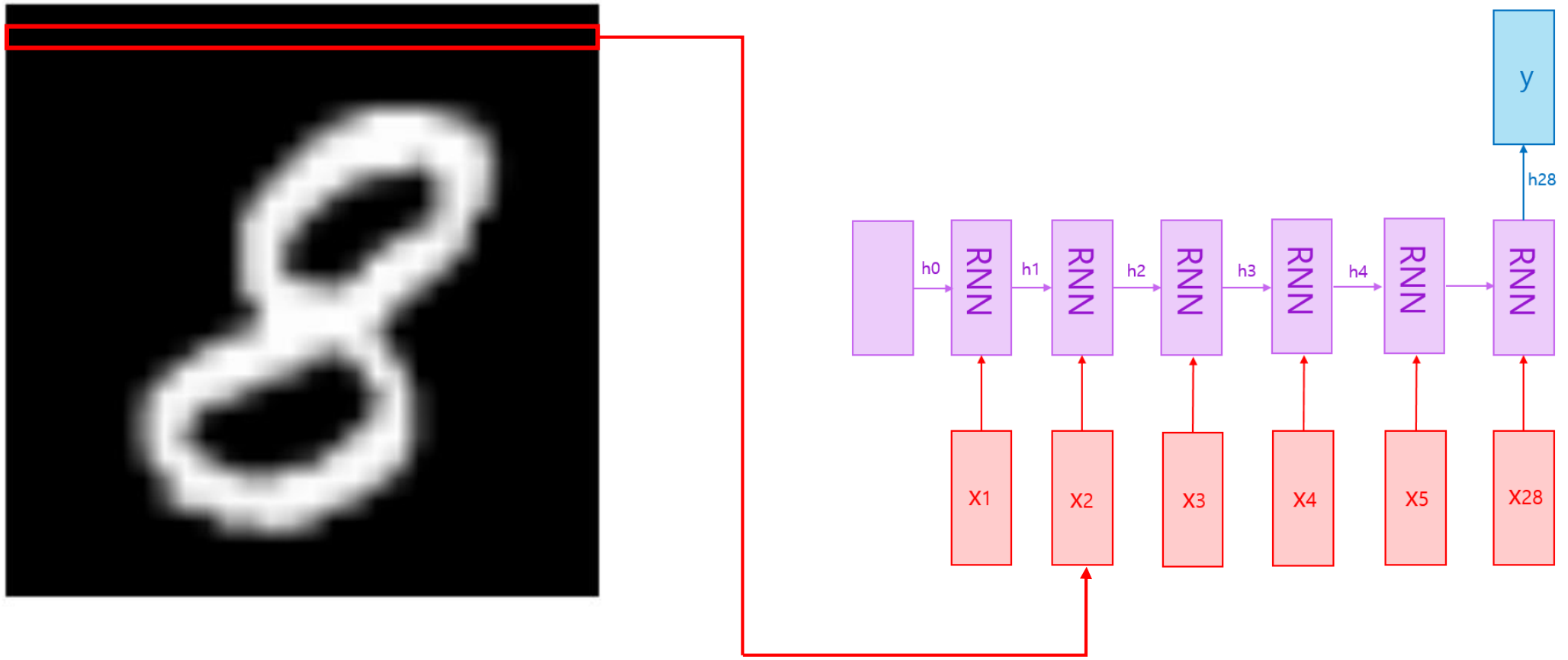
Recurrent Neural Network



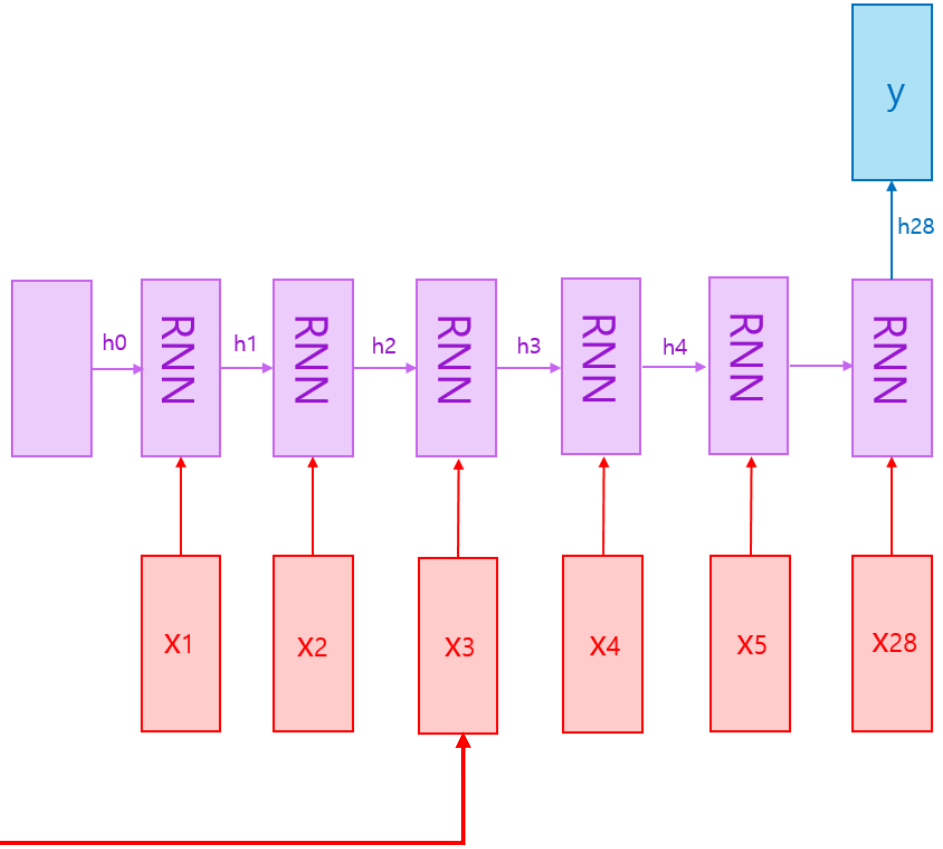
Recurrent Neural Network



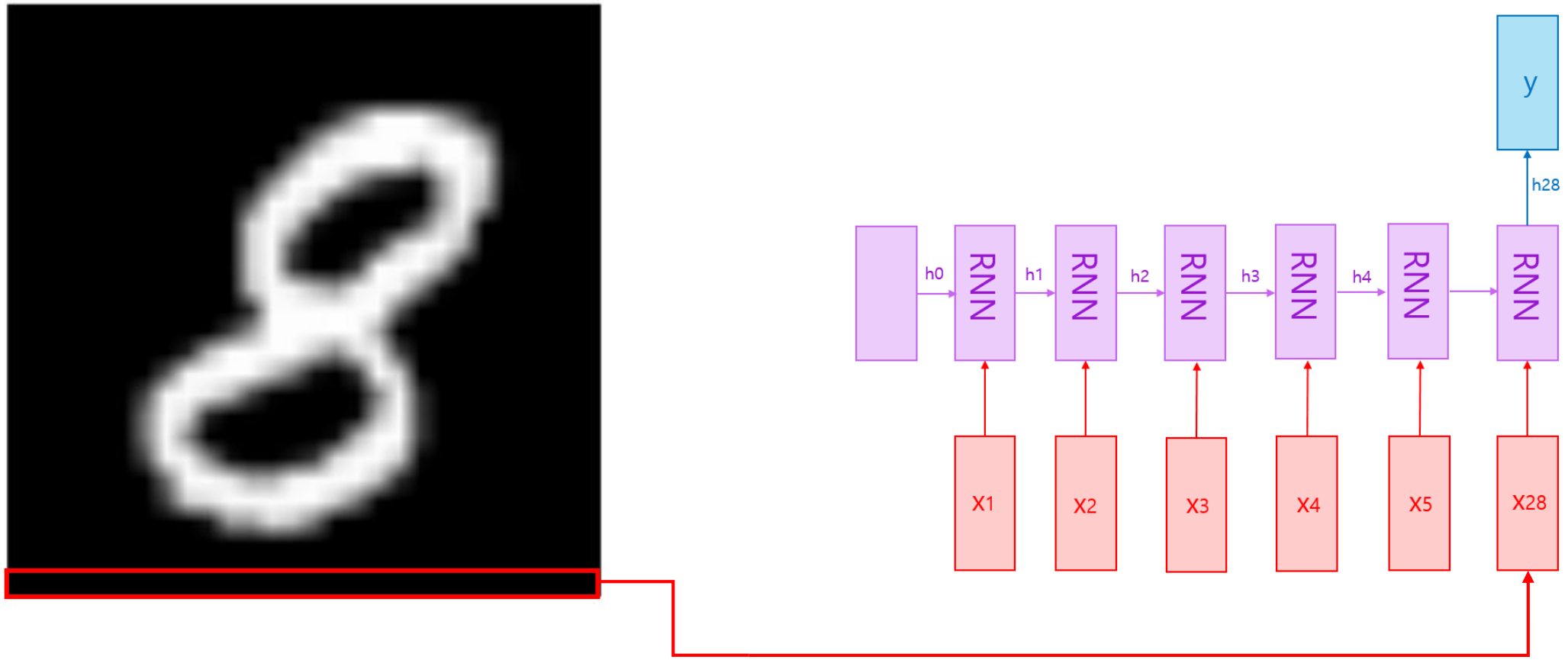
Recurrent Neural Network



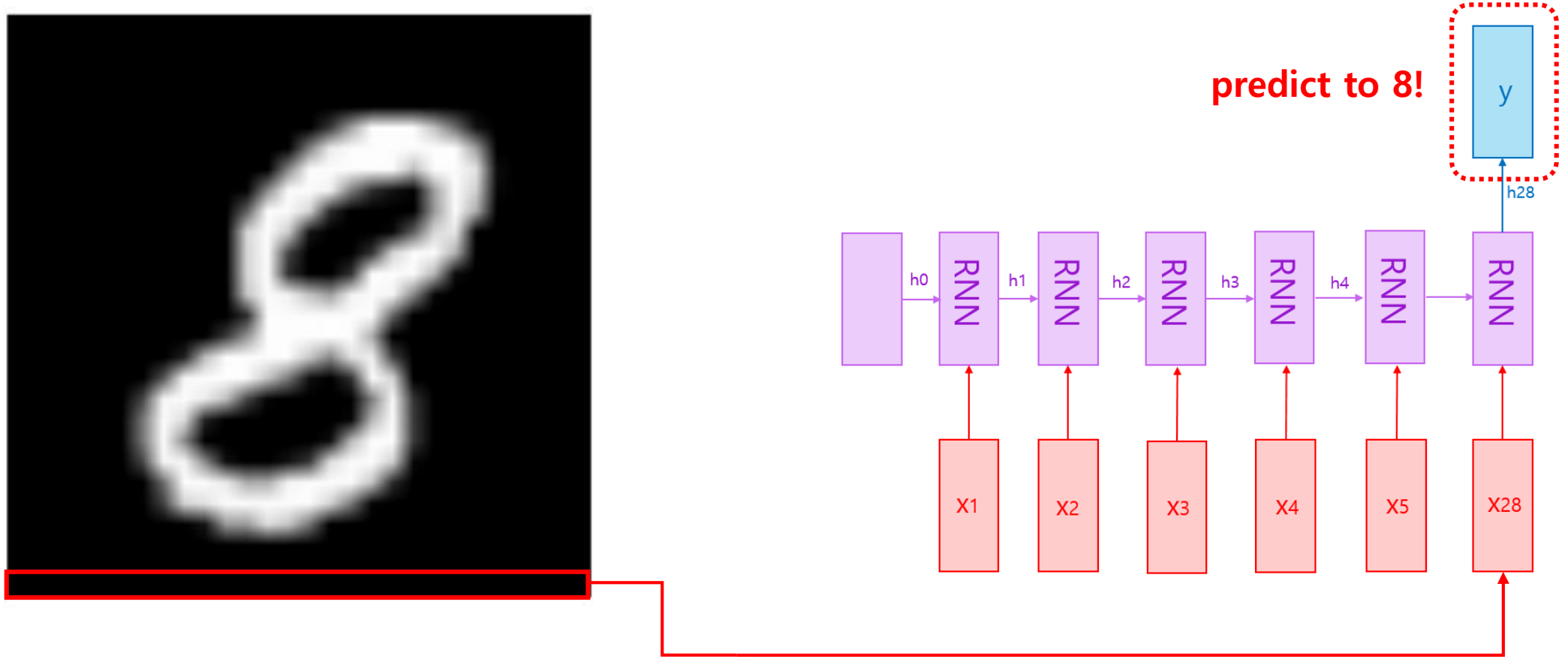
Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network

실습