

Network Sparsification, Pruning, Quantization

MLAI Lab.

2020. 07. 23

많은 것을 배웠지만?!



어떻게 만들어야 되는지 전혀 감이 잡히지 않는다 !

실습 목표

가중치 가지치기 (Weight Pruning)의 원리와 코드를 **쉽게** 이해하자 !

강의 내용 중에서..

Making Deep Neural Networks More Efficient

Reducing the number of parameters of given networks (weights / neurons)

- **Static:** **Pruning** sparsity-inducing regularizations and priors (Bayesian)
- **Dynamic:** Partial network evaluation

Reducing the memory and computation requirement per weight.

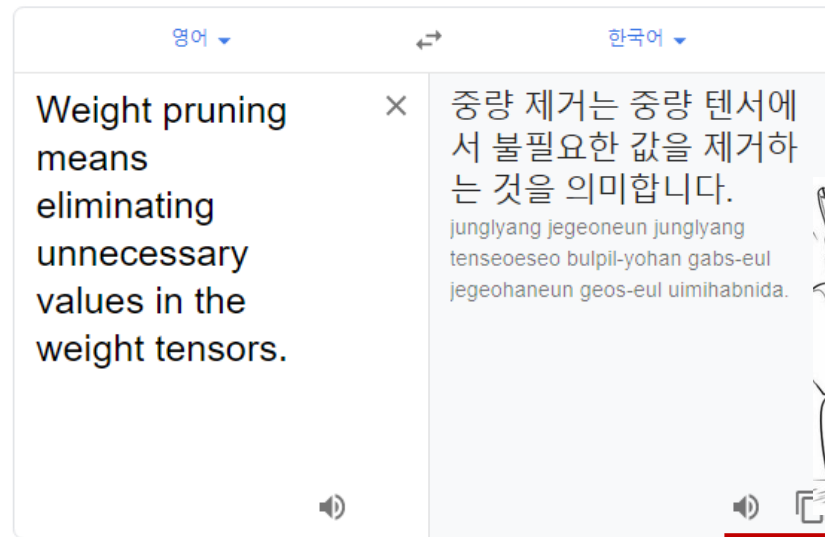
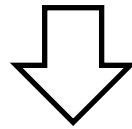
- Quantization (Bit compression)

Transferring knowledge to smaller networks – Knowledge distillation

Designing inherently efficient architectures – MobileNetv2, ShuffleNet

가중치 가지치기의 정의

“**Weight pruning means** eliminating unnecessary values in the **weight** tensors.”



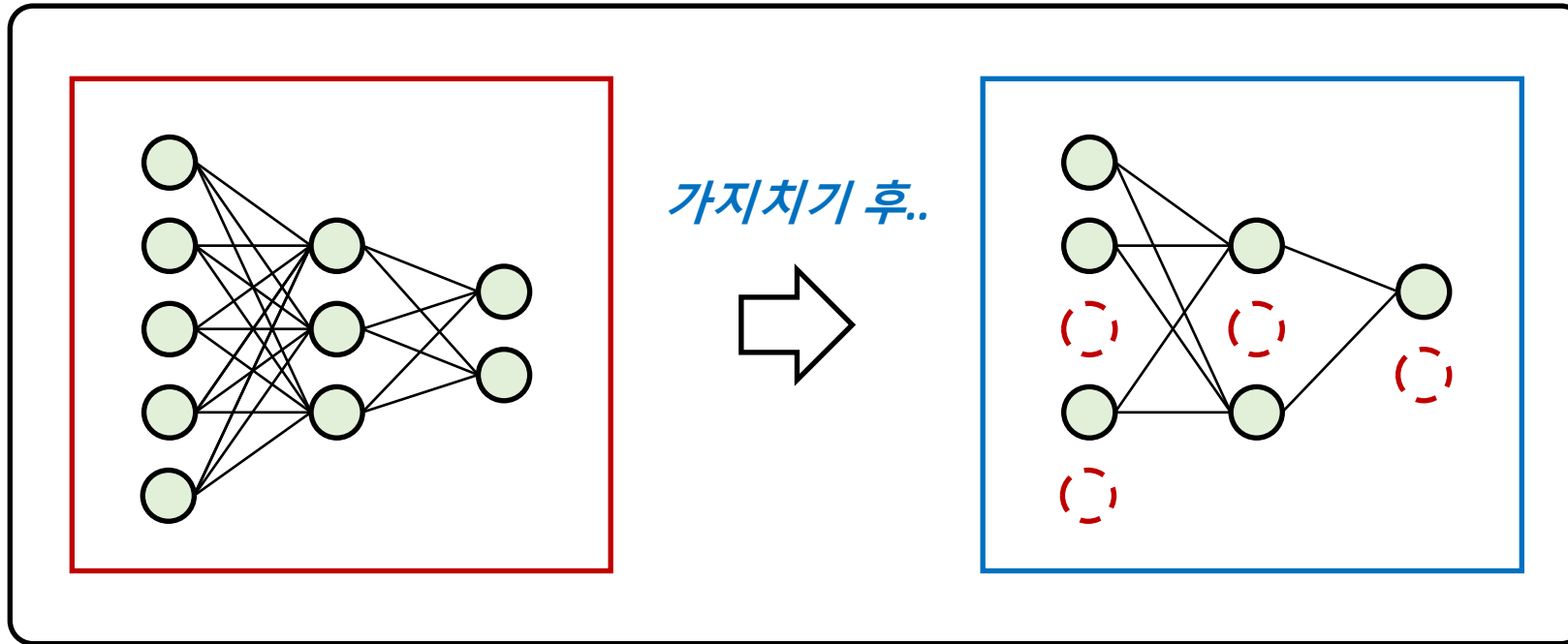
Google 번역에서 열기



사용자 의견

신고!

그림으로 이해해보자면?

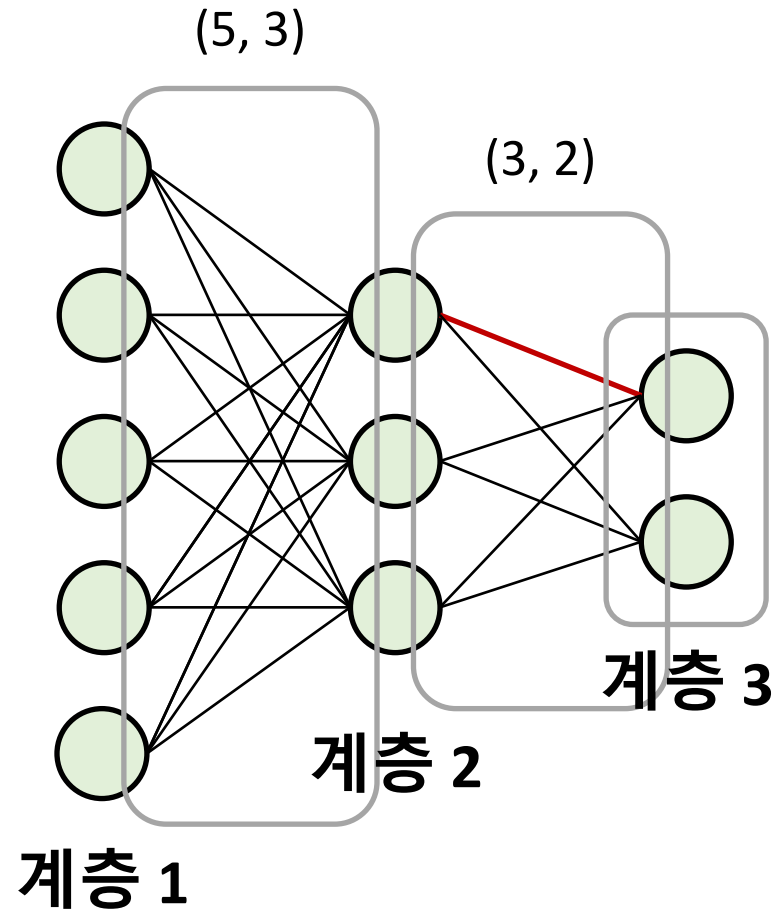


가지치기의 **기준**은?

금일 실습할 가지치기 기준

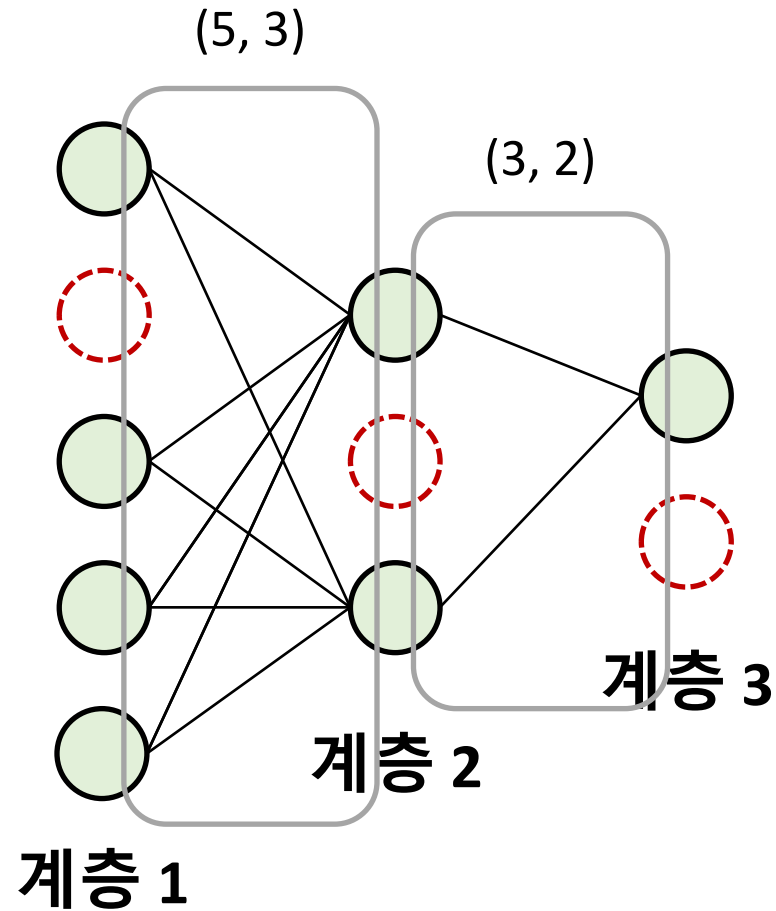
1. Weights의 **절댓값**을 기준으로 가지치기
2. Weights의 **$L2Norm$** 을 기준으로 가지치기 → 금일의 구현 목표 !

절댓값을 기준으로 가지치기



“**Weights의 절대값을 기준으로 정렬**하여
작은 순서대로 가지치기”

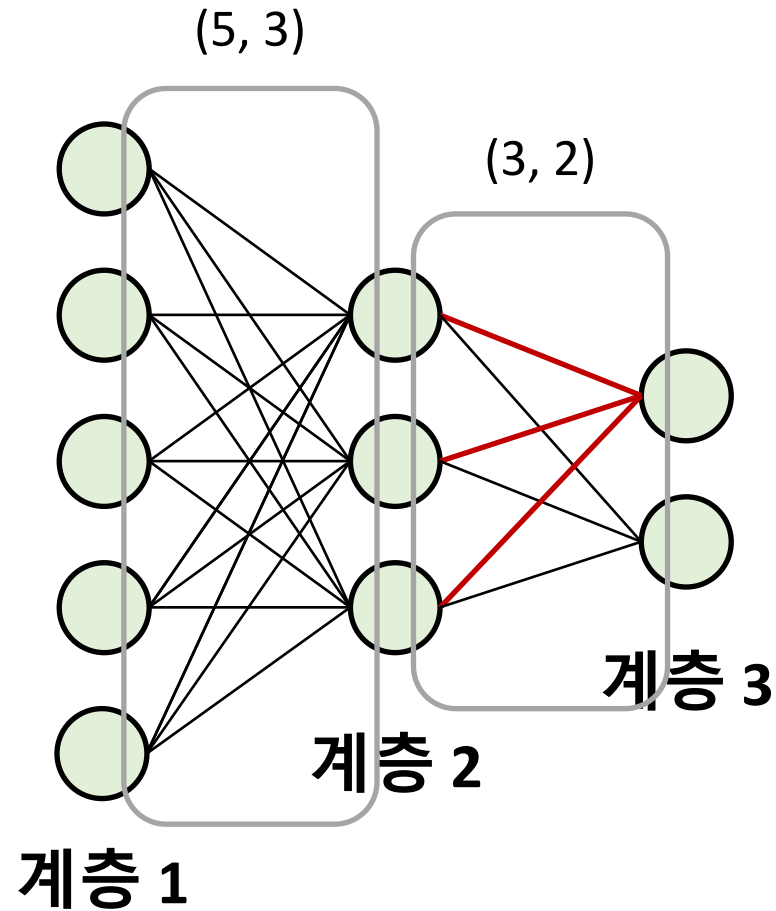
절댓값을 기준으로 가지치기



(3, 2) weights의 가지치기를 수행한다고 가정하면,

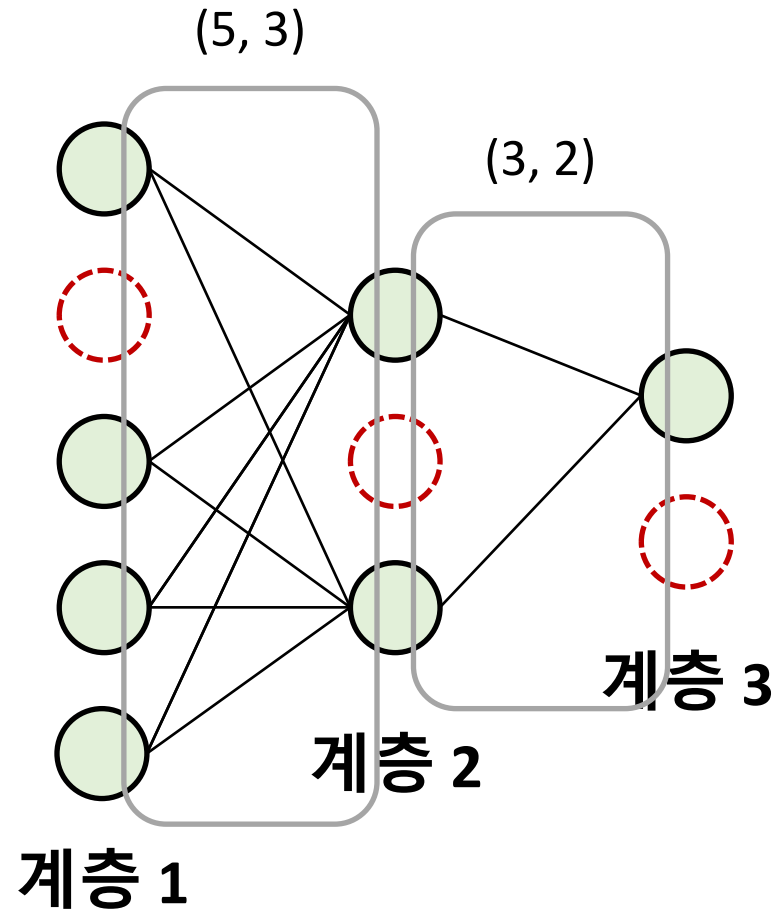
$$\begin{bmatrix} 0.5 & 0.21 \\ -0.1 & 0.12 \\ -0.4 & 0.35 \end{bmatrix} \xrightarrow{abs(\cdot)} \begin{bmatrix} 0.5 & \mathbf{0.21} \\ \mathbf{0.1} & \mathbf{0.12} \\ 0.4 & \mathbf{0.35} \end{bmatrix}$$

L2Norm을 기준으로 가지치기



“한 개의 뉴런에 대한 Weights에 대하여
L2Norm을 기준으로 **정렬**하여 가지치기”

L2Norm을 기준으로 가지치기



(3, 2) weights의 가지치기를 수행한다고 가정하면,

$$\begin{bmatrix} 0.5 & 0.21 \\ -0.1 & 0.12 \\ -0.4 & 0.35 \end{bmatrix} \xrightarrow{L2Norm(\cdot)} \begin{bmatrix} 0.648 & 0.421 \end{bmatrix}$$

$$\begin{aligned} L2Norm(\cdot) &= \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \\ &= \sqrt{(0.5)^2 + (-0.1)^2 + (-0.4)^2} \\ &= 0.648 \end{aligned}$$

실습 과정

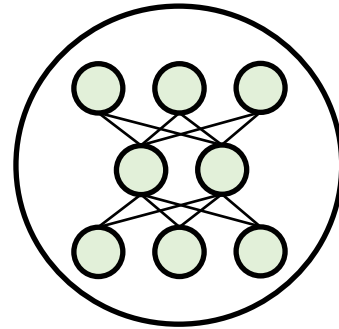
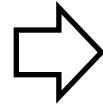
1. 실습 서버 접속하기
2. 환경설정하기
3. 실습 코드 설명
4. 실습 진행
5. 실습 환경 지우기 (중요)

실습 과정

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

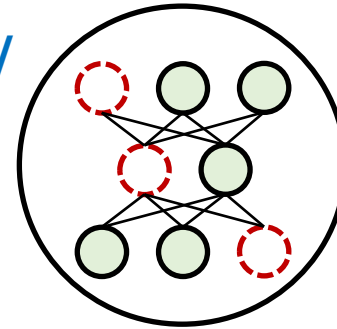
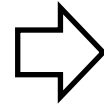
데이터 셋

학습

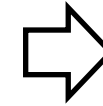


심층 학습 모델

가지치기

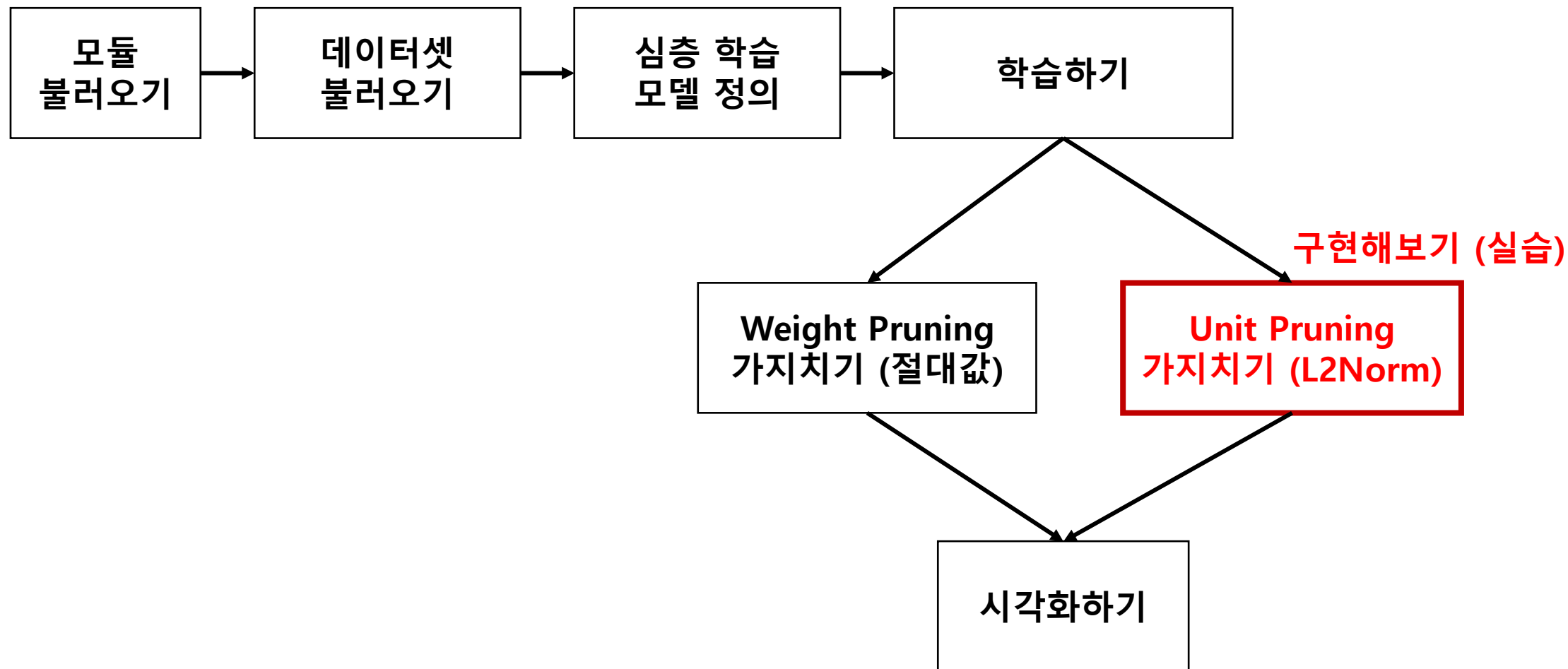


경량화 된 심층 학습 모델

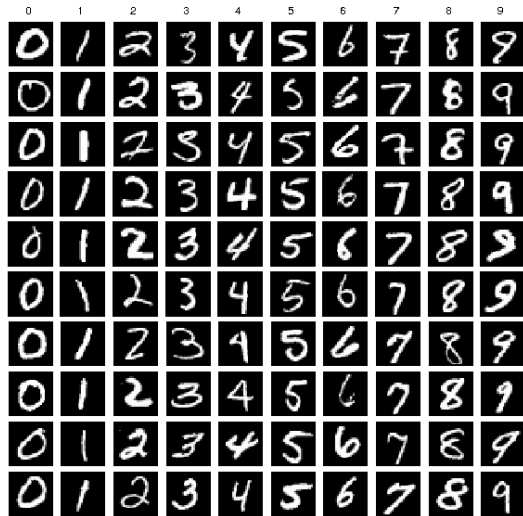


“시각화 하기”

실습 코드 설명



데이터셋



MNIST



Fashion-MNIST

MNIST: 0부터 9까지 총 10개의 분류가능한 클래스

Fashion-MNIST: T-shirt, Dress 등 총 10개의 분류가능한 클래스

심층 학습 모델 & 학습

5개의 Fully-connected layer와 ReLU 함수로 구성.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1000)	785000
dense_1 (Dense)	(None, 1000)	1001000
dense_2 (Dense)	(None, 500)	500500
dense_3 (Dense)	(None, 200)	100200
dense_4 (Dense)	(None, 10)	2010
Total params: 2,388,710		
Trainable params: 2,388,710		
Non-trainable params: 0		

마지막 단에 Softmax 함수를 통해 클래스 예측 확률을 0~1 사이로 정규화
Cross Entropy Loss 함수를 이용하여 Loss 학습 & 최소화

실습 서버 접속하기

- MobaXterm을 이용해주세요.



- 다음 명령어를 이용해서 각 자리에 할당 된 IP에 접속해주세요.

```
$ ssh -X -p 2222 com[LastDigit(s)]@[IP]
```

기본적으로 5~26 번 PC까지는 계정 번호와 IP주소 끝자리가 일치합니다.

ex) com05@143.248.159.5 / com0502@143.248.159.5

ex) com26@143.248.159.26 / com0502@143.248.159.26

그러나, 예외가 있습니다.

1번 PC: com01@143.248.159.27 / com0102@143.248.159.27

2번 PC: com02@143.248.159.28 / com0202@143.248.159.28

4번 PC: com04@143.248.159.3 / com0402@143.248.159.3

환경 설정하기

다음 명령어로 환경설정 해주세요.

```
$ source ~/.bashrc  
$ conda activate pruning_simple
```

코드 다운로드

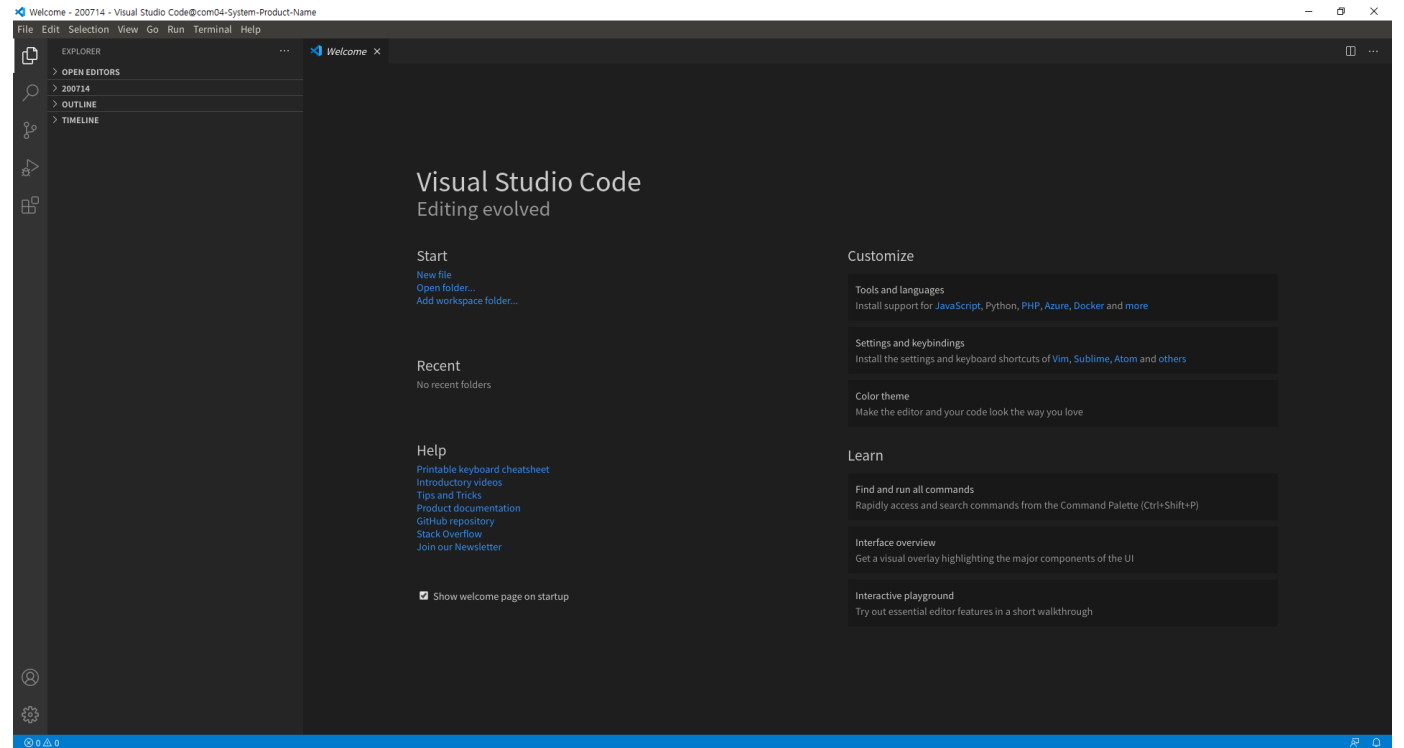
다음 명령어로 코드 다운로드를 해주세요.

```
$ git clone https://github.com/h19920918/samsung_ds_summer_pruning.git  
$ cd samsung_ds_summer_pruning
```

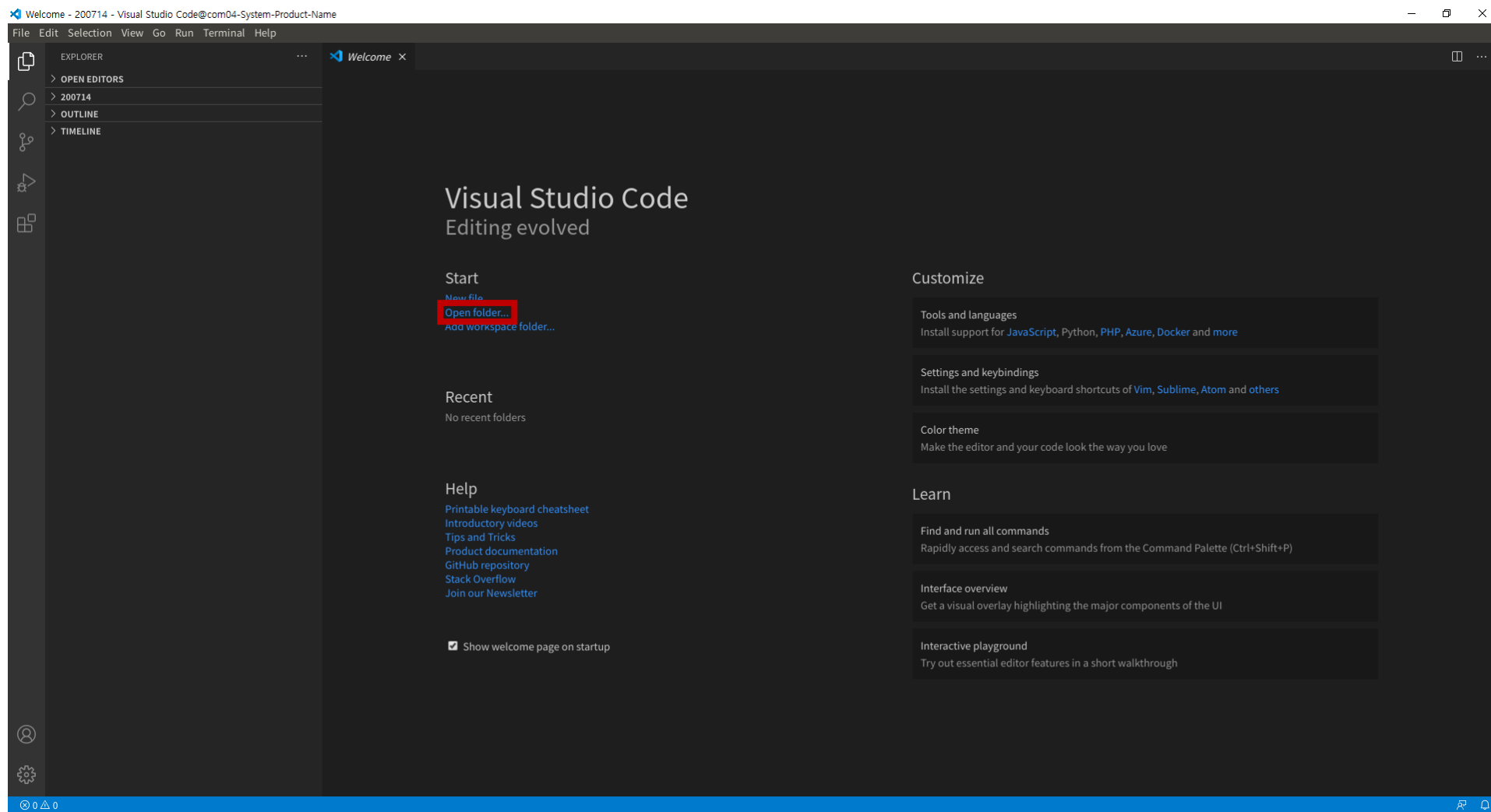
비주얼 스튜디오 실행

다음 명령어로 비주얼 스튜디오를 실행해주세요.

```
$ code
```



코드 폴더 불러오기



모듈 불러오기

```
1 # 필요한 라이브러리 불러오기
2 import math
3 import matplotlib
4 import matplotlib.pyplot as plt
5 import matplotlib.axes as axes
6 import numpy as np
7 import os
8 # os.environ["CUDA_VISIBLE_DEVICES"]="0"
9 # os.environ["CUDA_VISIBLE_DEVICES"]="1"
10
11 import pandas as pd
12 import seaborn as sns
13 import tempfile
14 import tensorboard
15 import tensorflow as tf
16 import timeit
17 import zipfile
18
19 from IPython.core.pylabtools import figsize
20 from numpy import linalg as LA
21 from tensorflow.keras.models import load_model
22 from tensorflow_model_optimization.sparsity import keras as sparsity
23
24 # Tensorflow 버전 확인
25 print(tf.__version__)
26
27 plt.style.use('fivethirtyeight')
28 sns.set_context('notebook')
29 pd.set_option('display.max_rows', 30)
30 np.random.seed(1337)
31 %config InlineBackend.figure_format = 'retina'
32 %load_ext tensorboard
```

조교 가이드에 따라 0 또는 1로 설정해주세요.

데이터셋 불러오기

```
1 # MNIST와 Fashion-MNIST 데이터셋 다운로드 / 불러오기 함수
2 def load_dataset(dataset='mnist'):
3     """
4     Loads and preprocesses the data for this task.
5     Args:
6         dataset: the name of the dataset to be used for this classification task.
7                 (mnist | fmnist)
8     Returns:
9         x_train: Features for training data
10        x_test: Features for test data
11        y_train: Labels for training data
12        y_test: Labels for test data
13        num_classes: Number of classes for the dataset
14    """
```

MNIST와 Fashion-MNIST를 로드합니다.

```
56 # MNIST 데이터셋 정의
57 mnist_x_train, mnist_x_test, mnist_y_train, mnist_y_test, num_classes, input_shape = load_dataset(dataset='mnist')
58
59 # Fashion-MNIST 데이터셋 정의
60 fmnist_x_train, fmnist_x_test, fmnist_y_train, fmnist_y_test, num_classes, input_shape = load_dataset(dataset='fmnist')
```

심층 학습 모델 정의

```
4 # 심층 학습 모델 빌드 함수
5 def build_model_arch(input_shape, num_classes, sparsity=0.0):
6     """
7     Builds the model architecture
8     Args:
9         input_shape: The tuple describing the input shape
10        num_classes: how many classes the data labels belong to
11        sparsity: For compressing already sparse models, how much sparsity was used
12    Returns:
13        model: an un-compiled TF.Keras model with 4 hidden
14        dense layers with shapes [1000, 1000, 500, 200]
15    """
16
17    model = tf.keras.Sequential()
18
19    # 5 Fully-connected layer with ReLU function 정의
20    # 한개의 클래스의 Probability를 최대화 하기위해서 마지막 단계 Softmax function 사용
21    model.add(l.Dense(int(1000-(1000*sparsity)), activation='relu',
22                      input_shape=input_shape),
23              l.Dense(int(1000-(1000*sparsity)), activation='relu'))
24    model.add(l.Dense(int(500-(500*sparsity)), activation='relu'))
25    model.add(l.Dense(int(200-(200*sparsity)), activation='relu'))
26    model.add(l.Dense(num_classes, activation='softmax'))
27
28    return model
```

네 개의 Fully Connected Layer와 마지막 Softmax layer로 구성된 모델을 빌드합니다.

```
30 # MNIST와 Fashion-MNIST 모델 빌드 (Sparsity == 0.0)
31 mnist_model_base = build_model_arch(input_shape, num_classes)
32 fmnist_model_base = build_model_arch(input_shape, num_classes)
```

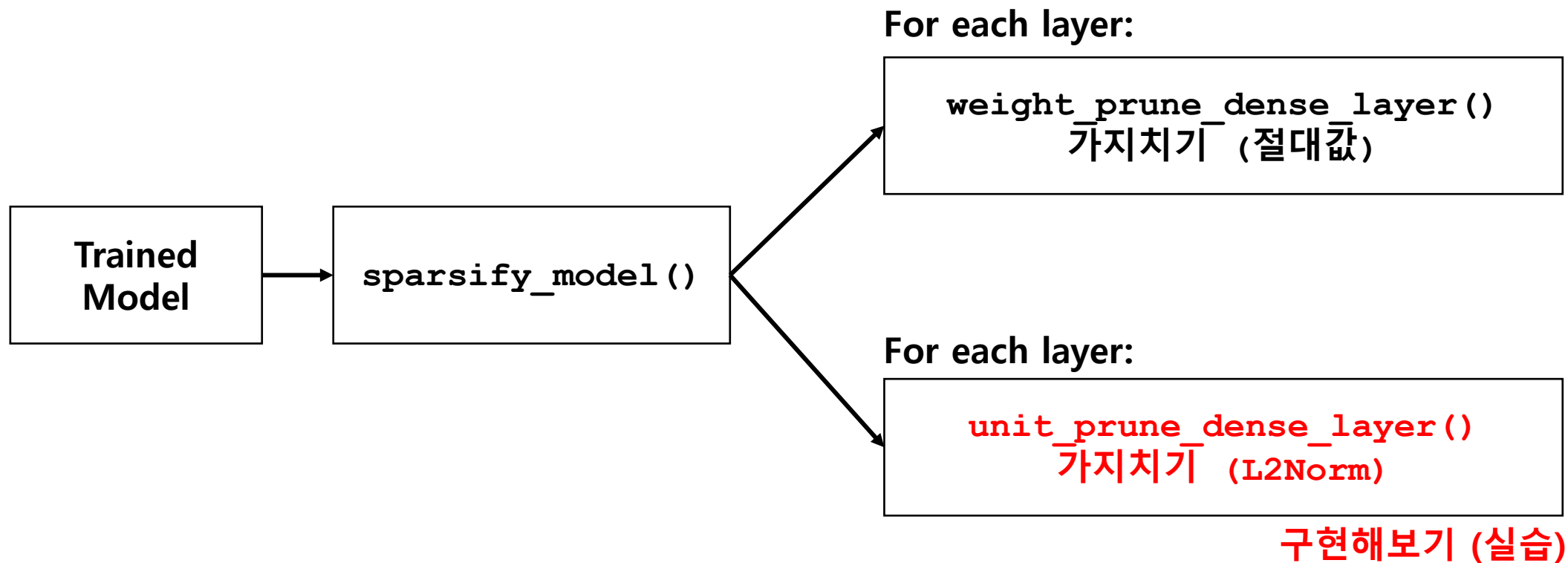

학습하기 (Non-Sparse)

```
1 # 심층 학습 모델 학습 함수
2 def make_nosparse_model(model, x_train, y_train, batch_size, epochs, x_test, y_test):
3     """
4     Training our original model, pre-pruning
5     Args:
6         model: Uncompiled Keras model
7         x_train: Features for training data
8         y_train: Labels for training data
9         batch_size: Batch size for training
10        epochs: Number of epochs for training
11        x_test: Features for test data
12        y_test: Labels for test data
13    Returns:
14        model: compiled model
15        score: List of both final test loss and final test accuracy
16    """
```

Optimizer: Adam Optimizer
Loss Function: Cross Entropy
Metric: Accuracy
Batch Size: 128
Epochs: 10

```
24     model.compile(
25         optimizer='adam',
26         loss=tf.keras.losses.categorical_crossentropy,
27         metrics=['accuracy'])
28
29     # 심층 학습 모델 학습
30     model.fit(x_train, y_train,
31             batch_size=batch_size,
32             epochs=epochs,
33             verbose=1,
34             callbacks=callbacks,
35             validation_data=(x_test, y_test))
36
37     # 심층 학습 모델 평가
38     score = model.evaluate(x_test, y_test, verbose=0)
39     print('Test loss:', score[0])
40     print('Test accuracy:', score[1])
41
42     return model, score
```

가지치기



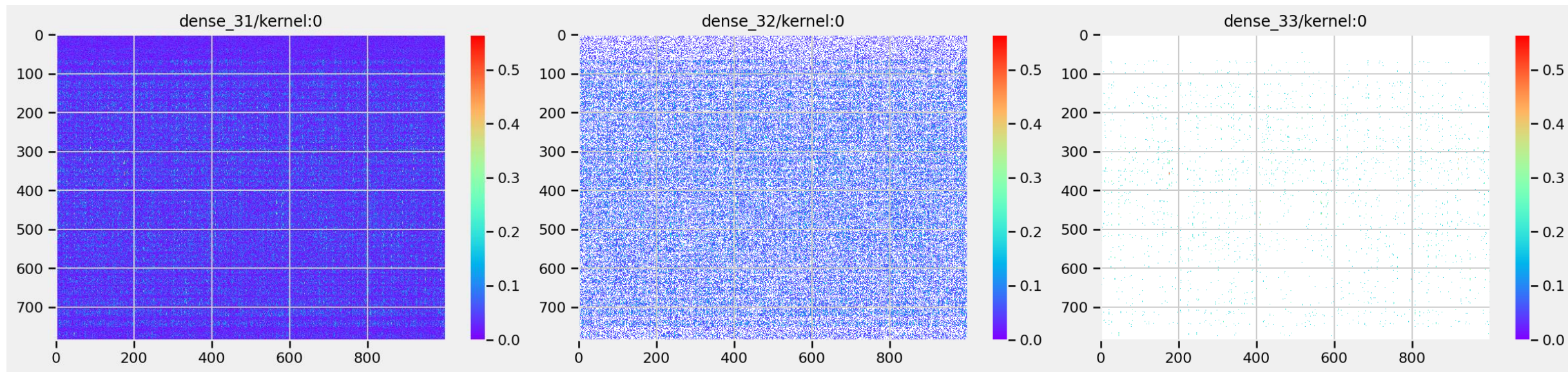
가지치기 (절대값)

```
1 # Pruning 함수 (Weight)
2 def weight_prune_dense_layer(k_weights, b_weights, k_sparsity):
3     """
4     Takes in matrices of kernel and bias weights (for a dense
5     layer) and returns the unit-pruned versions of each
6     Args:
7         k_weights: 2D matrix of the
8         b_weights: 1D matrix of the biases of a dense layer
9         k_sparsity: percentage of weights to set to 0
10    Returns:
11        kernel_weights: sparse matrix with same shape as the original
12        kernel weight matrix
13        bias_weights: sparse array with same shape as the original
14        bias array
15    """
```

```
17 # Kernel weights 복사
18 kernel_weights = np.copy(k_weights)
19
20 # Weight의 절대값을 기준으로 인덱스 정렬 (2D matrix)
21 ind = np.unravel_index(
22     np.argsort(
23         np.abs(kernel_weights),
24         axis=None),
25     kernel_weights.shape)
26
27 # Pruning 하는 Weights 갯수 정의
28 cutoff = int(len(ind[0])*k_sparsity)
29
30 # 정렬된 인덱스를 기준으로 Weight pruning 수행 (0)
31 sparse_cutoff_inds = (ind[0][0:cutoff], ind[1][0:cutoff])
32 kernel_weights[sparse_cutoff_inds] = 0.
33
34 # Kernel bias에 대하여 동일한 작업 수행
35 bias_weights = np.copy(b_weights)
36
37 ind = np.unravel_index(
38     np.argsort(
39         np.abs(bias_weights),
40         axis=None),
41     bias_weights.shape)
42
43 cutoff = int(len(ind[0])*k_sparsity)
44
45 sparse_cutoff_inds = (ind[0][0:cutoff])
46 bias_weights[sparse_cutoff_inds] = 0.
47
48 return kernel_weights, bias_weights
```


시각화하기

```
1 # 모델 파라미터 Visualization
2 def visualize_model_weights(sparse_model):
3     """
4     Visualize the weights of the layers of the sparse model.
5     For weights with values of 0, they will be represented by the color white
6     Args:
7         sparse_model: a TF.Keras model
8     """
```



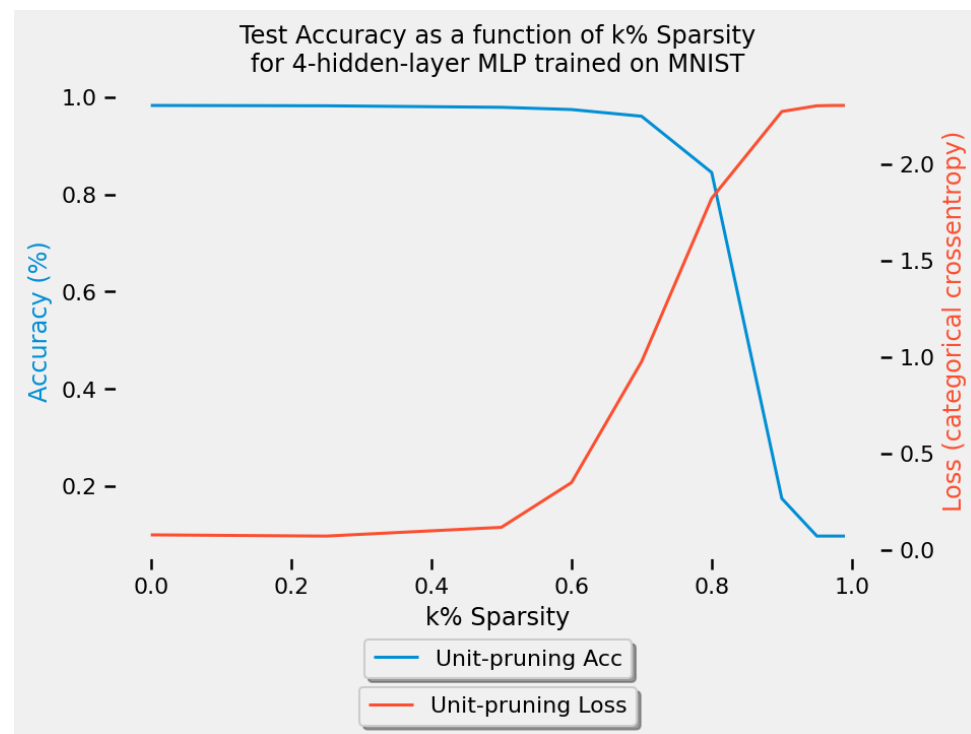
Sparsity = 0.0

Sparsity = 0.5

Sparsity = 0.99

시각화하기

```
1 # MNIST 데이터셋 성능 Visualization
2 fig = plt.figure()
3 ax1 = fig.add_subplot(1, 1, 1)
4 plt.grid(b=None)
5 ax2 = ax1.twinx()
6 plt.grid(b=None)
7 plt.title('Test Accuracy as a function of k% Sparsity\nfor 4-hidden-layer MLP trained on MNIST')
8 # ax1.plot(sparsity_summary['k_sparsity'].values,
9 #         sparsity_summary['mnist_acc_weight'].values,
10 #         '#008fd5', linestyle=':', label='Weight-pruning Acc')
11 ax1.plot(sparsity_summary['k_sparsity'].values,
12         sparsity_summary['mnist_acc_unit'].values,
13         '#008fd5', linestyle='-', label='Unit-pruning Acc')
14 # ax2.plot(sparsity_summary['k_sparsity'].values,
15 #         sparsity_summary['mnist_loss_weight'].values,
16 #         '#fc4f30', linestyle=':', label='Weight-pruning Loss')
17 ax2.plot(sparsity_summary['k_sparsity'].values,
18         sparsity_summary['mnist_loss_unit'].values,
19         '#fc4f30', linestyle='-', label='Unit-pruning Loss')
20
21 ax1.set_ylabel('Accuracy (%)', color='#008fd5')
22 ax2.set_ylabel('Loss (categorical crossentropy)', color='#fc4f30')
23 ax1.set_xlabel('k% Sparsity')
24 ax1.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), shadow=True, ncol=2);
25 ax2.legend(loc='upper center', bbox_to_anchor=(0.5, -0.25), shadow=True, ncol=2);
26 plt.savefig('images/MNIST_sparsity_comparisons.png')
```



가지치기 (L2Norm)

무엇이든 물어보세요!

```
1 # Pruning 함수 (Unit)
2 def unit_prune_dense_layer(k_weights, b_weights, k_sparsity):
3     """
4     Takes in matrices of kernel and bias weights (for a dense
5     layer) and returns the unit-pruned versions of each
6     Args:
7         k_weights: 2D matrix of the
8         b_weights: 1D matrix of the biases of a dense layer
9         k_sparsity: percentage of weights to set to 0
10    Returns:
11        kernel_weights: sparse matrix with same shape as the original
12        kernel weight matrix
13        bias_weights: sparse array with same shape as the original
14        bias array
15    """
16
17    # Kernel weights 복사
18
19    # Column-wise L2 Norms 계산 후 인덱스 정렬
20
21    # Pruning 하는 Weights 갯수 정의
22
23    # 정렬된 인덱스를 기준으로 Weight pruning 수행 (0)
24
25    # Kernel bias에 대하여 Weight를 기준으로 Pruning 된 인덱스에 대하여 0 적용
26
27    return kernel_weights, bias_weights
```

실습 환경 지우기

다음 명령어로 실습 환경을 삭제해주세요.

```
$ conda deactivate  
$ conda env remove -n pruning_simple
```

실습 과제

1. Adam Optimizer 대신 SGD Optimizer 사용해보기
2. 기존 모델에서 두 개 이상의 Fully-connected Layer를 더 쌓아보기
3. L2Norm 기준 가지치기 함수 완성
4. CNN 기반 심층 학습 모델 정의 후 학습&가지치기 수행하기 (심화)