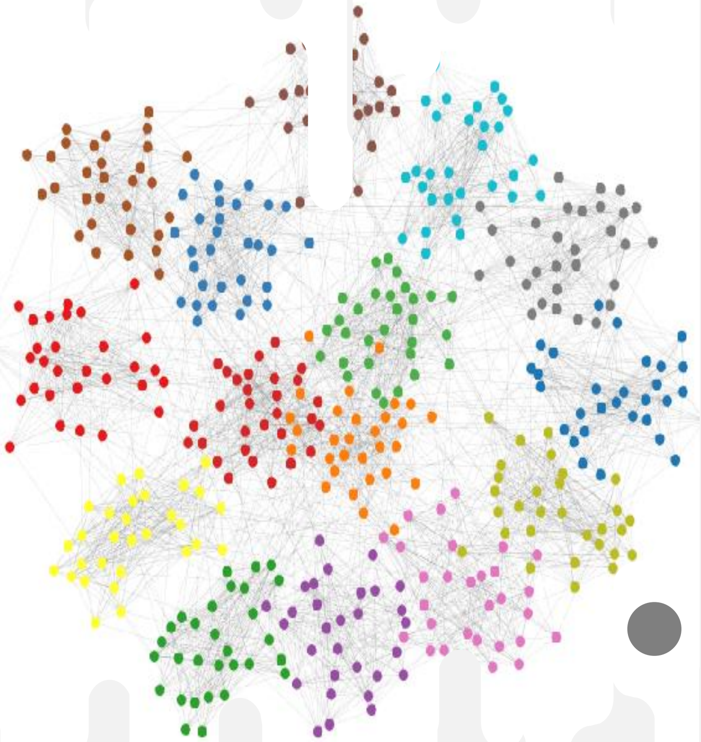
Model Compression Homework

繳交期限：2025 / 05 / 27（二）23:59

TA email: lobsterlab.cs.nthu@gmail.com



大綱

We are going to cover …

01 作業介紹

02 提供的 **code**

03 繳交內容

04 配分標準

1

作業介紹

- DEEP COMPRESSION
- Python library you need

DEEP COMPRESSION

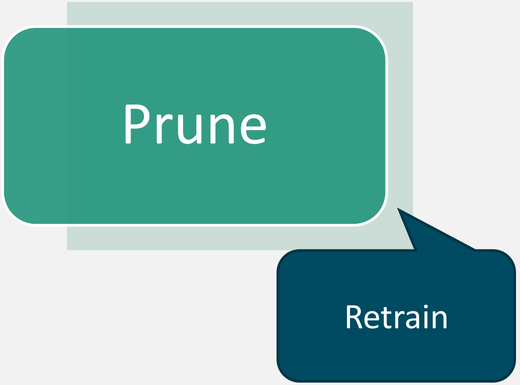
- 架構：

Prune

https://arxiv.org/pdf/1510.00149.pdf

DEEP COMPRESSION

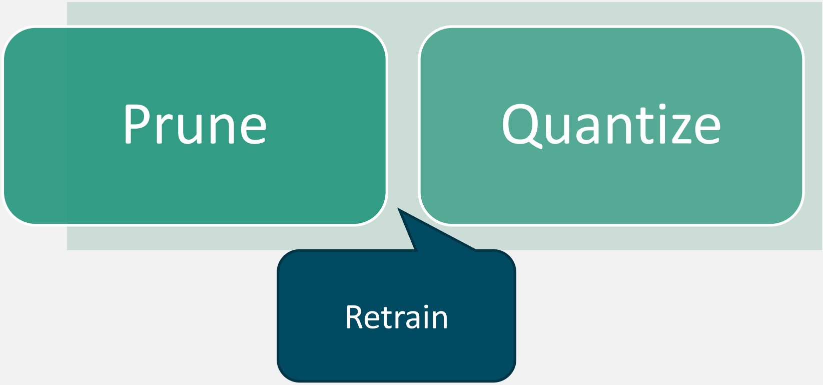
- 架構：



https://arxiv.org/pdf/1510.00149.pdf

DEEP COMPRESSION

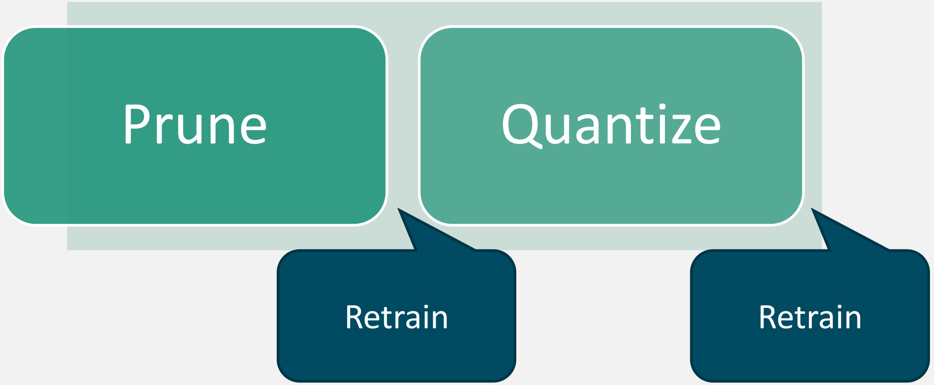
- 架構：



https://arxiv.org/pdf/1510.00149.pdf

DEEP COMPRESSION

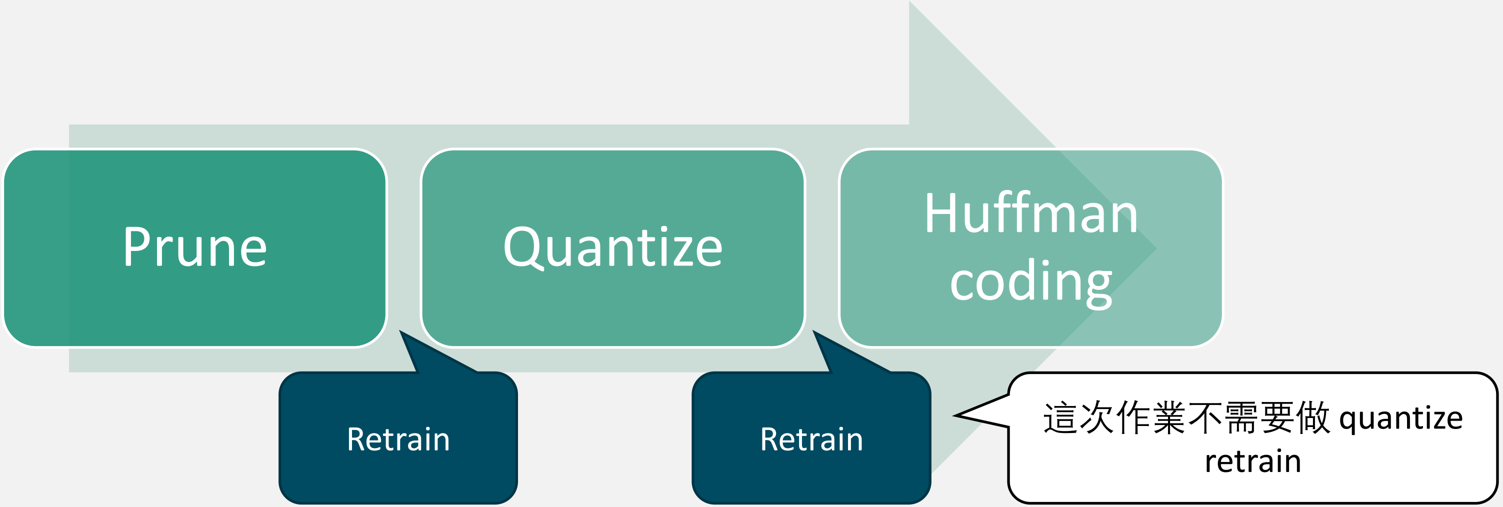
- 架構：



https://arxiv.org/pdf/1510.00149.pdf

DEEP COMPRESSION

- 架構：



Prune

Retrain

Quantize

Retrain

Huffman coding

這次作業不需要做 quantize retrain

https://arxiv.org/pdf/1510.00149.pdf

Python library

- pytorch
  - 深度學習套件
  - 記得載 GPU 版

v0.22.0

Python library

- torchvision
  - 下載常用測試 data 套件
  - 本次資料使用 cifar10
  - 也可用來下載mnist、cifar100

v1.6.1

Python library

- scikit-learn (sklearn)
  - 機器學習套件
  - 本次要使用 cluster 的演算法 : Kmeans
  - (在 quantize 那一步)

scikits learn

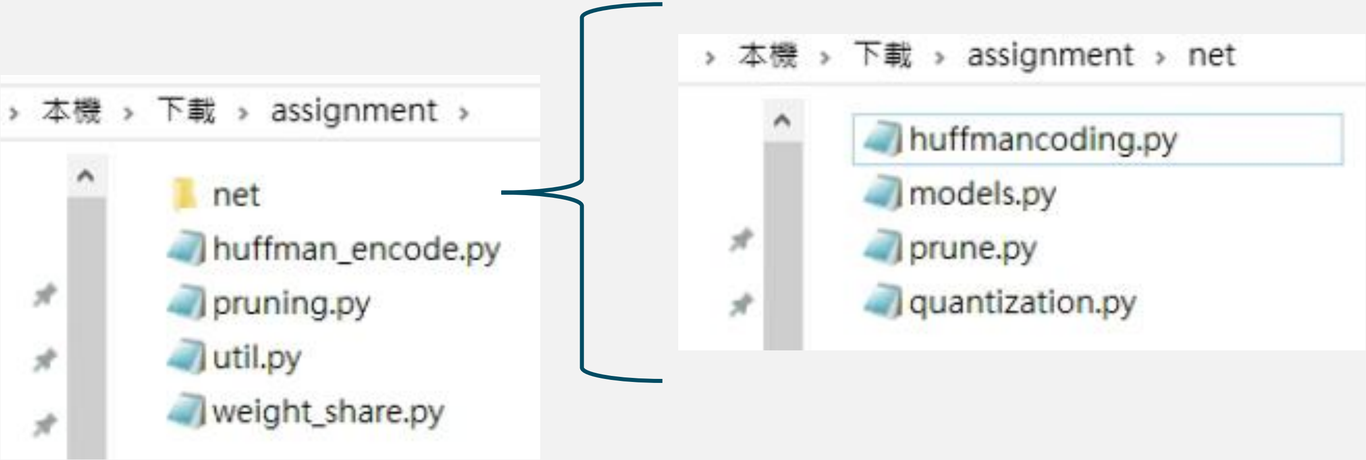
machine learning in Python

2

提供的 code

- 模型
- **Prune** 部分
- Quantize 部分
- Huffman 部分

Assignment



› 本機 › 下載 › assignment ›

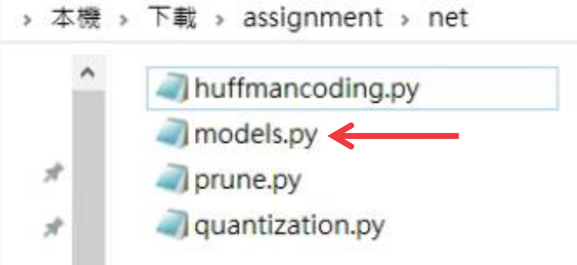
- net
- huffman_encode.py
- pruning.py
- util.py
- weight_share.py

› 本機 › 下載 › assignment › net

- huffmancoding.py
- models.py
- prune.py
- quantization.py

模型

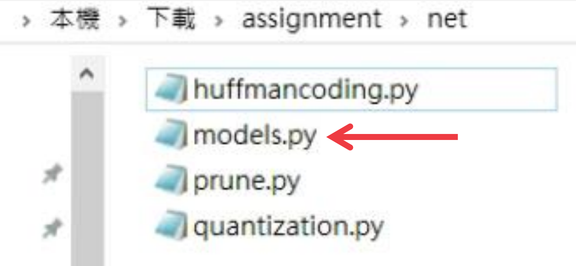
- AlexNet



› 本機 › 下載 › assignment › net
huffmancoding.py
models.py
prune.py
quantization.py

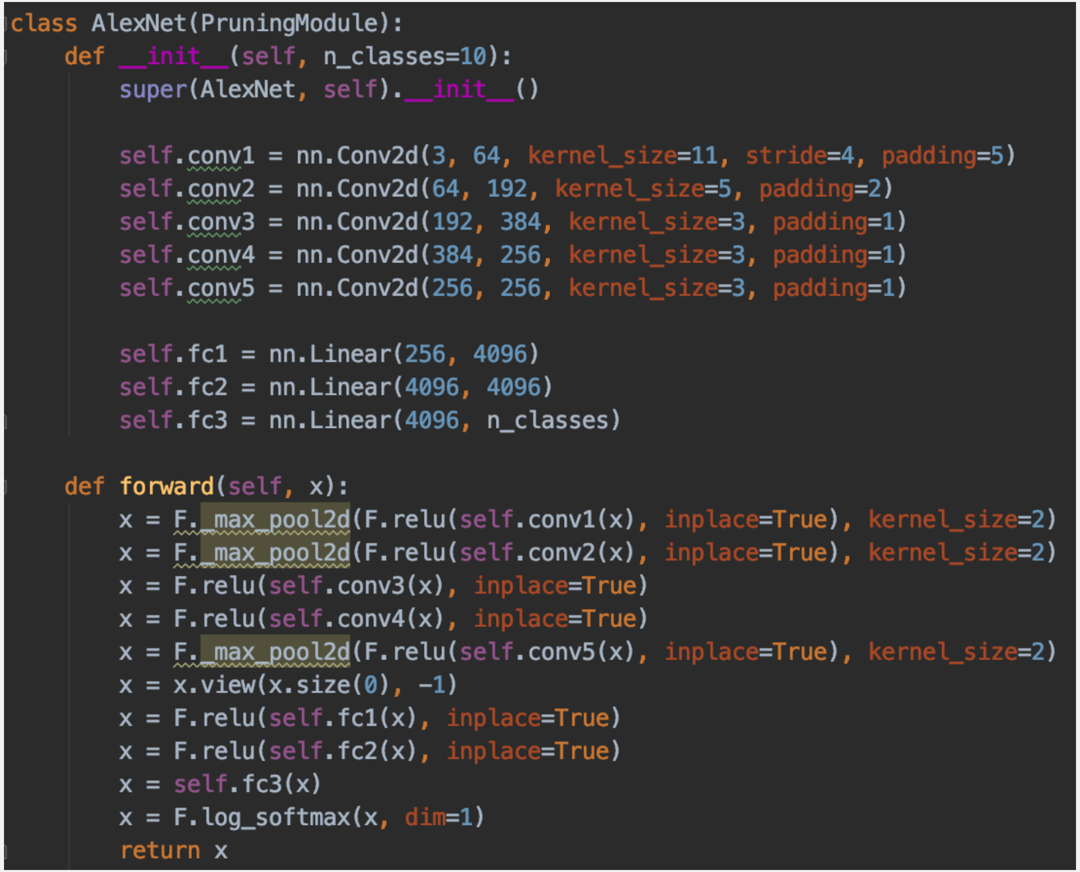
模型

- AlexNet



› 本機 › 下載 › assignment › net

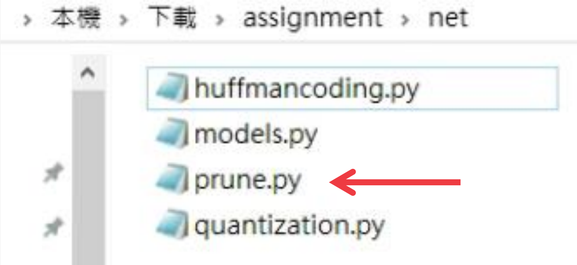
huffmancoding.py
models.py
prune.py
quantization.py



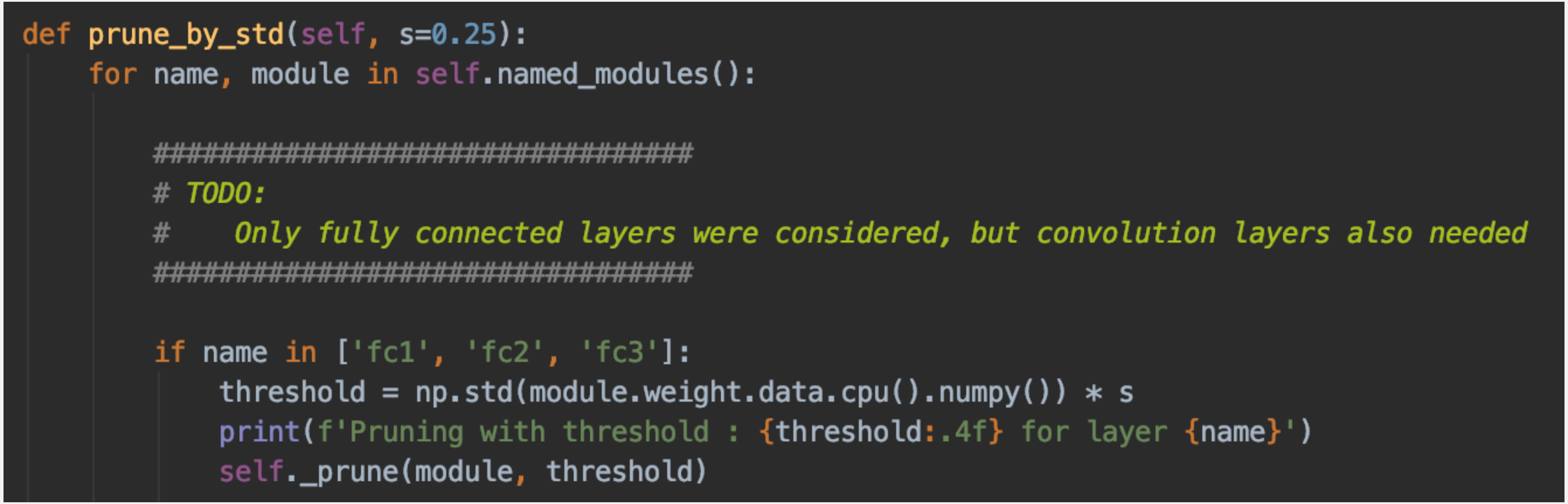
```
class AlexNet(PruningModule):
    def __init__(self, n_classes=10):
        super(AlexNet, self).__init__()

        self.conv1 = nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=5)
        self.conv2 = nn.Conv2d(64, 192, kernel_size=5, padding=2)
        self.conv3 = nn.Conv2d(192, 384, kernel_size=3, padding=1)
        self.conv4 = nn.Conv2d(384, 256, kernel_size=3, padding=1)
        self.conv5 = nn.Conv2d(256, 256, kernel_size=3, padding=1)

        self.fc1 = nn.Linear(256, 4096)
        self.fc2 = nn.Linear(4096, 4096)
        self.fc3 = nn.Linear(4096, n_classes)

    def forward(self, x):
        x = F._max_pool2d(F.relu(self.conv1(x), inplace=True), kernel_size=2)
        x = F._max_pool2d(F.relu(self.conv2(x), inplace=True), kernel_size=2)
        x = F.relu(self.conv3(x), inplace=True)
        x = F.relu(self.conv4(x), inplace=True)
        x = F._max_pool2d(F.relu(self.conv5(x), inplace=True), kernel_size=2)
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x), inplace=True)
        x = F.relu(self.fc2(x), inplace=True)
        x = self.fc3(x)
        x = F.log_softmax(x, dim=1)
        return x
```

Prune

- prune.py
- 只提供 prune by std 的 fully connected 部分（壓縮率不可預期）
- 可選擇實作 prune by percentile 與否（壓縮率可預期）

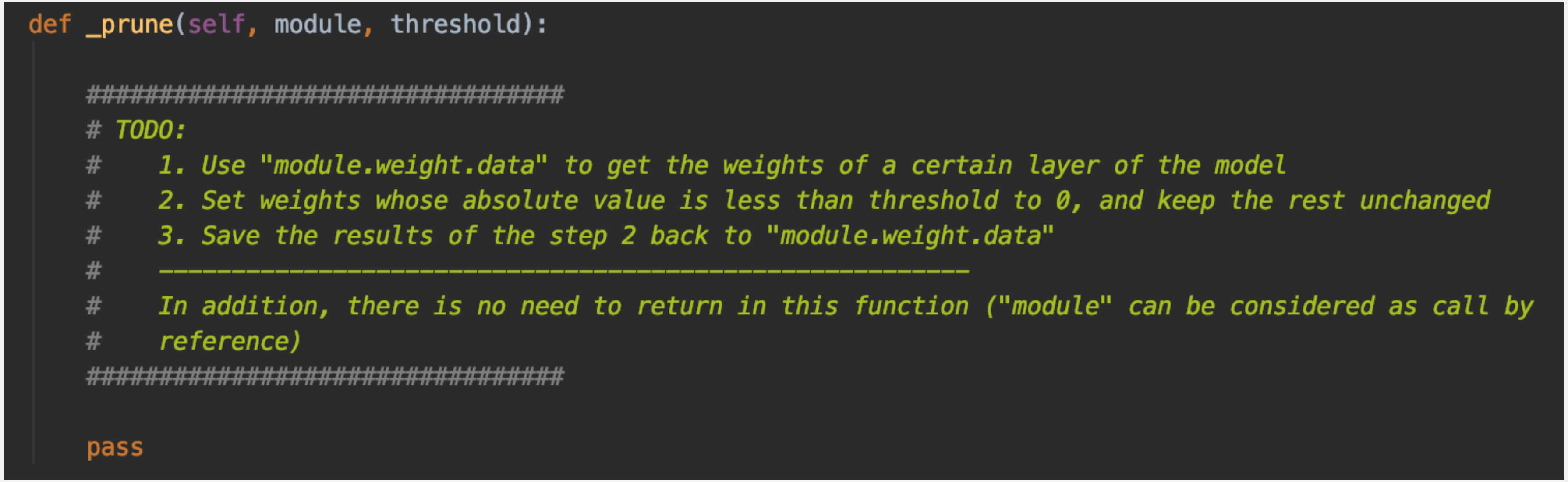


prune.py



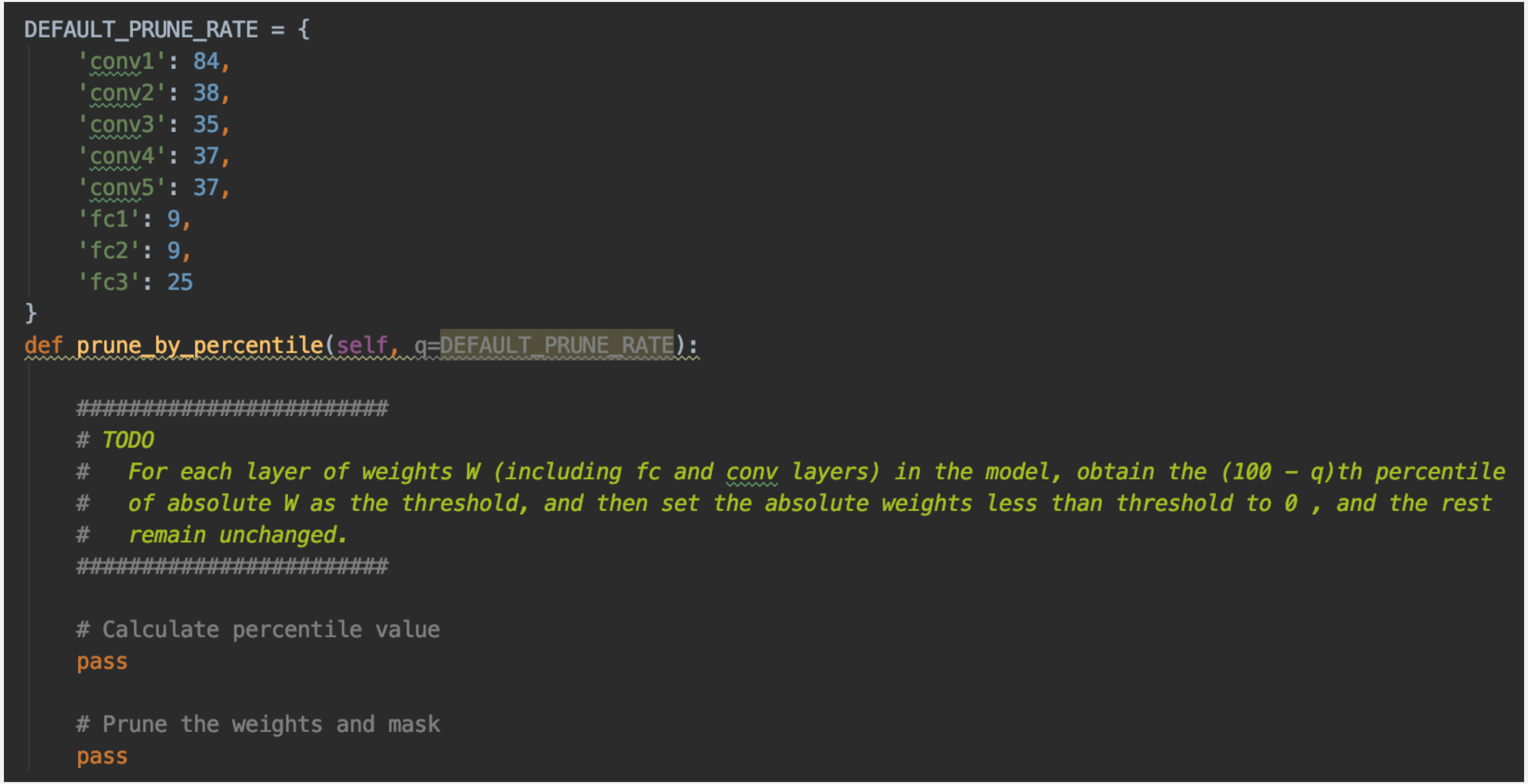
```
def prune_by_std(self, s=0.25):
    for name, module in self.named_modules():

        #################################
        # TODO:
        #    Only fully connected layers were considered, but convolution layers also needed
        #################################

        if name in ['fc1', 'fc2', 'fc3']:
            threshold = np.std(module.weight.data.cpu().numpy()) * s
            print(f'Pruning with threshold : {threshold:.4f} for layer {name}')
            self._prune(module, threshold)
```

prune.py



```
def _prune(self, module, threshold):

    ##################################
    # TODO:
    #    1. Use "module.weight.data" to get the weights of a certain layer of the model
    #    2. Set weights whose absolute value is less than threshold to 0, and keep the rest unchanged
    #    3. Save the results of the step 2 back to "module.weight.data"
    #    ---------------------------------------------------------
    #    In addition, there is no need to return in this function ("module" can be considered as call by
    #    reference)
    ##################################

    pass
```

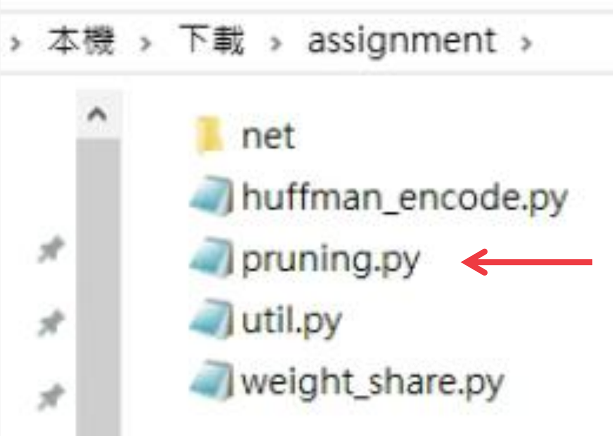
prune.py



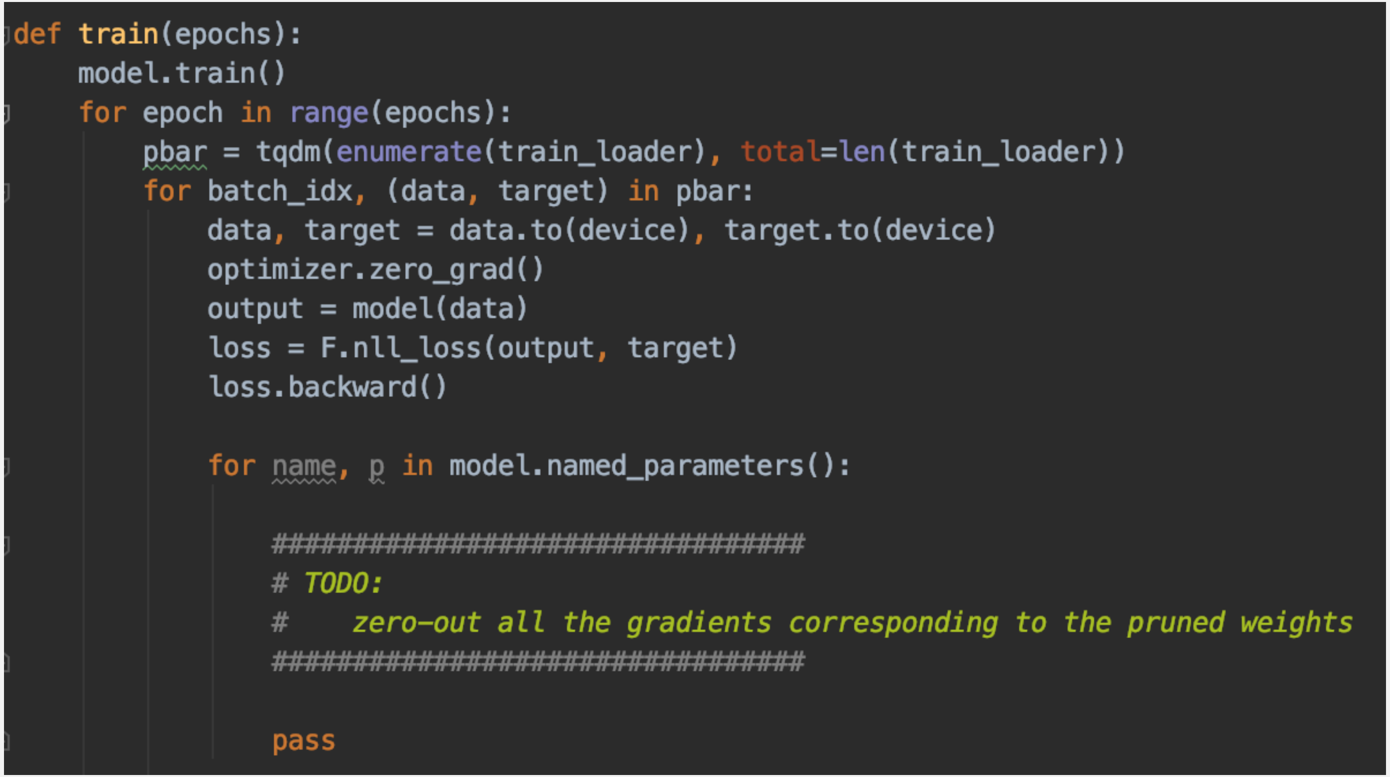
```
DEFAULT_PRUNE_RATE = {
    'conv1': 84,
    'conv2': 38,
    'conv3': 35,
    'conv4': 37,
    'conv5': 37,
    'fc1': 9,
    'fc2': 9,
    'fc3': 25
}
def prune_by_percentile(self, q=DEFAULT_PRUNE_RATE):

    ########################
    # TODO
    #   For each layer of weights W (including fc and conv layers) in the model, obtain the (100 - q)th percentile
    #   of absolute W as the threshold, and then set the absolute weights less than threshold to 0 , and the rest
    #   remain unchanged.
    ########################

    # Calculate percentile value
    pass

    # Prune the weights and mask
    pass
```

Prune

- pruning.py
- Initial train + prune + prune retrain 的部分



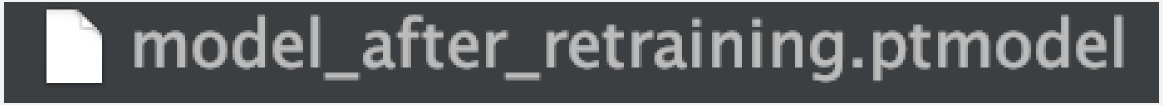
pruning.py



```
def train(epochs):
    model.train()
    for epoch in range(epochs):
        pbar = tqdm(enumerate(train_loader), total=len(train_loader))
        for batch_idx, (data, target) in pbar:
            data, target = data.to(device), target.to(device)
            optimizer.zero_grad()
            output = model(data)
            loss = F.nll_loss(output, target)
            loss.backward()

            for name, p in model.named_parameters():

                #################################
                # TODO:
                #    zero-out all the gradients corresponding to the pruned weights
                #################################

                pass
```

Run : pruning.py

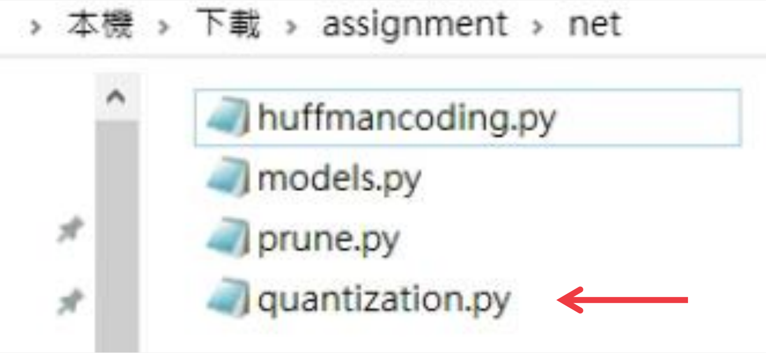
- 生出 :



model_after_retraining.ptmodel

Quantize

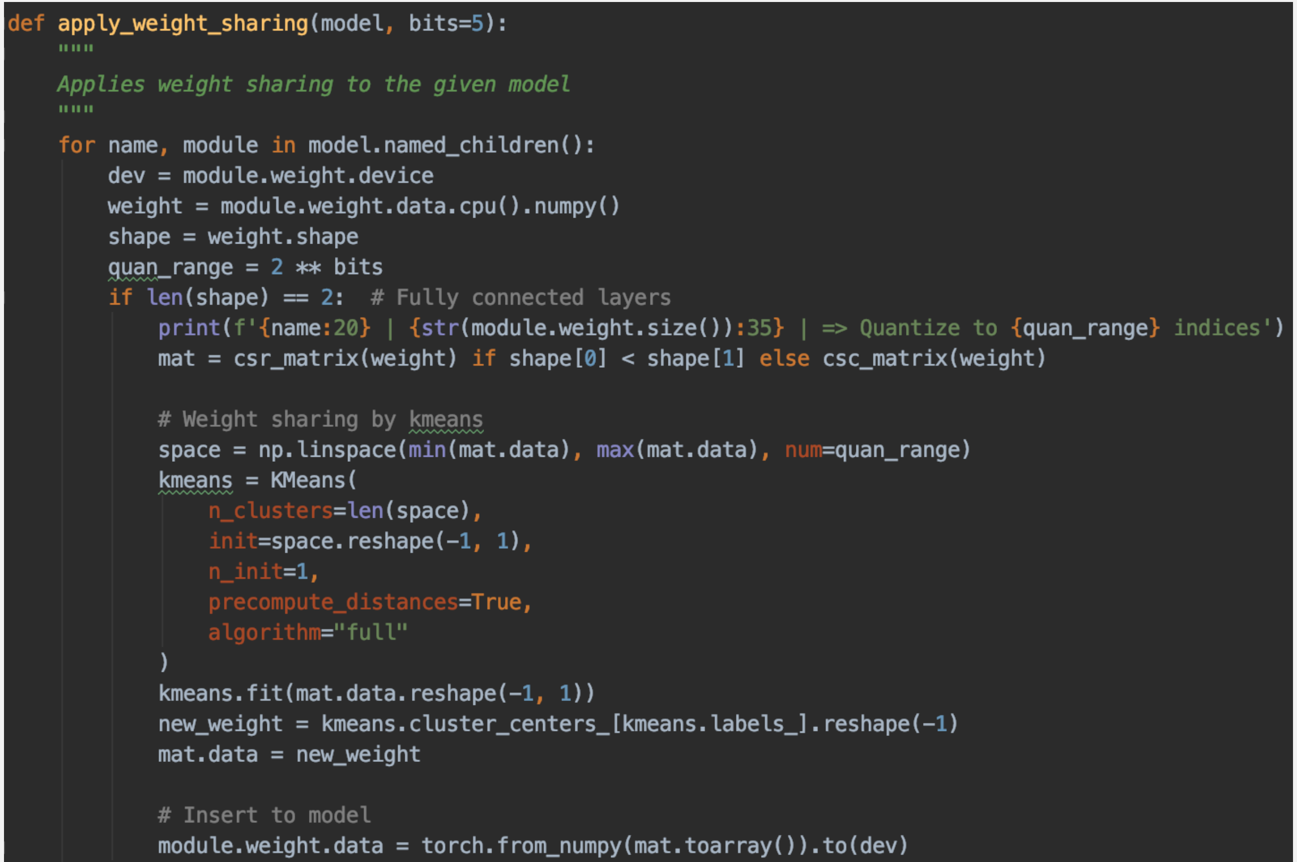
- quantization.py
- 只提供 fully connected 部分



› 本機 › 下載 › assignment › net

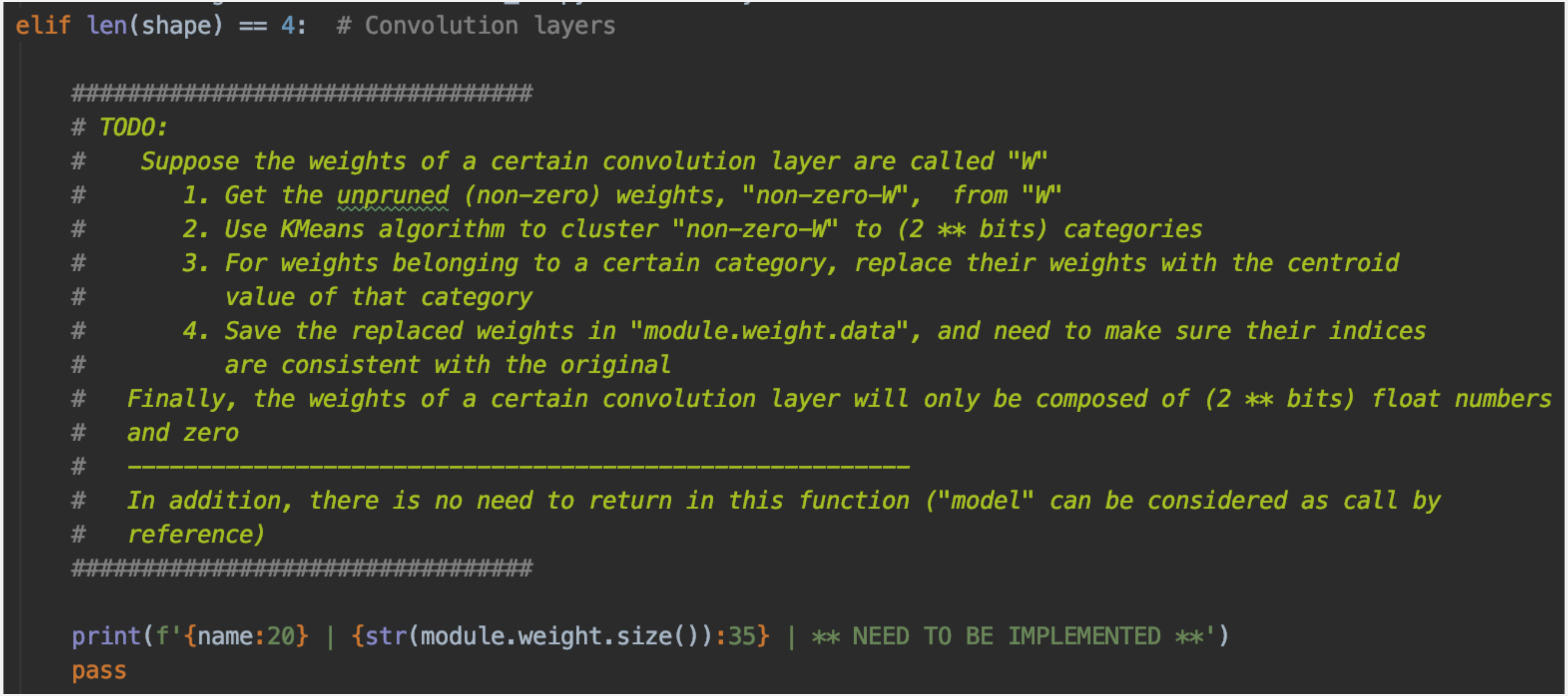
huffmancoding.py
models.py
prune.py
quantization.py

quantization.py



```python
def apply_weight_sharing(model, bits=5):
    """
    Applies weight sharing to the given model
    """
    for name, module in model.named_children():
        dev = module.weight.device
        weight = module.weight.data.cpu().numpy()
        shape = weight.shape
        quan_range = 2 ** bits
        if len(shape) == 2:  # Fully connected layers
            print(f'{name:20} | {str(module.weight.size()):35} | => Quantize to {quan_range} indices')
            mat = csr_matrix(weight) if shape[0] < shape[1] else csc_matrix(weight)

            # Weight sharing by kmeans
            space = np.linspace(min(mat.data), max(mat.data), num=quan_range)
            kmeans = KMeans(
                n_clusters=len(space),
                init=space.reshape(-1, 1),
                n_init=1,
                precompute_distances=True,
                algorithm="full"
            )
            kmeans.fit(mat.data.reshape(-1, 1))
            new_weight = kmeans.cluster_centers_[kmeans.labels_].reshape(-1)
            mat.data = new_weight

            # Insert to model
            module.weight.data = torch.from_numpy(mat.toarray()).to(dev)
```

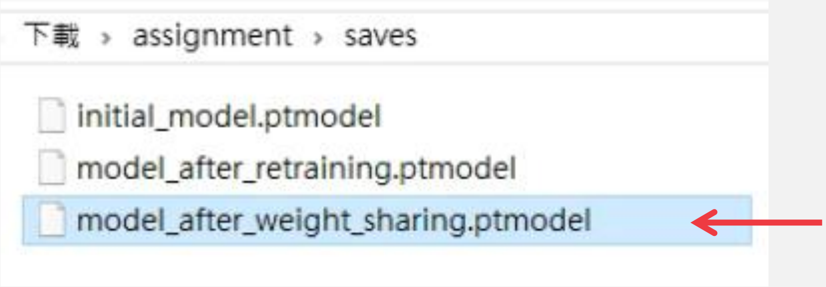
quantization.py



```python
elif len(shape) == 4:  # Convolution layers

    #################################
    # TODO:
    #    Suppose the weights of a certain convolution layer are called "W"
    #       1. Get the unpruned (non-zero) weights, "non-zero-W",  from "W"
    #       2. Use KMeans algorithm to cluster "non-zero-W" to (2 ** bits) categories
    #       3. For weights belonging to a certain category, replace their weights with the centroid
    #          value of that category
    #       4. Save the replaced weights in "module.weight.data", and need to make sure their indices
    #          are consistent with the original
    #   Finally, the weights of a certain convolution layer will only be composed of (2 ** bits) float numbers
    #   and zero
    #   --------------------------------------------------------
    #   In addition, there is no need to return in this function ("model" can be considered as call by
    #   reference)
    #################################

    print(f'{name:20} | {str(module.weight.size()):35} | ** NEED TO BE IMPLEMENTED **')
    pass
```

Run : weight_share.py

- 生出：

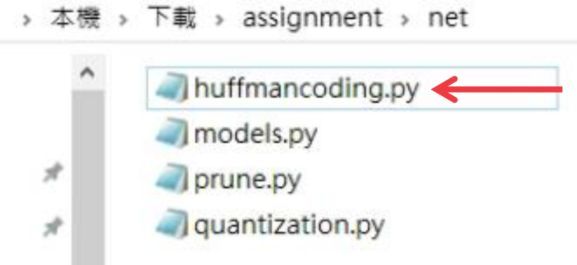


下載 › assignment › saves

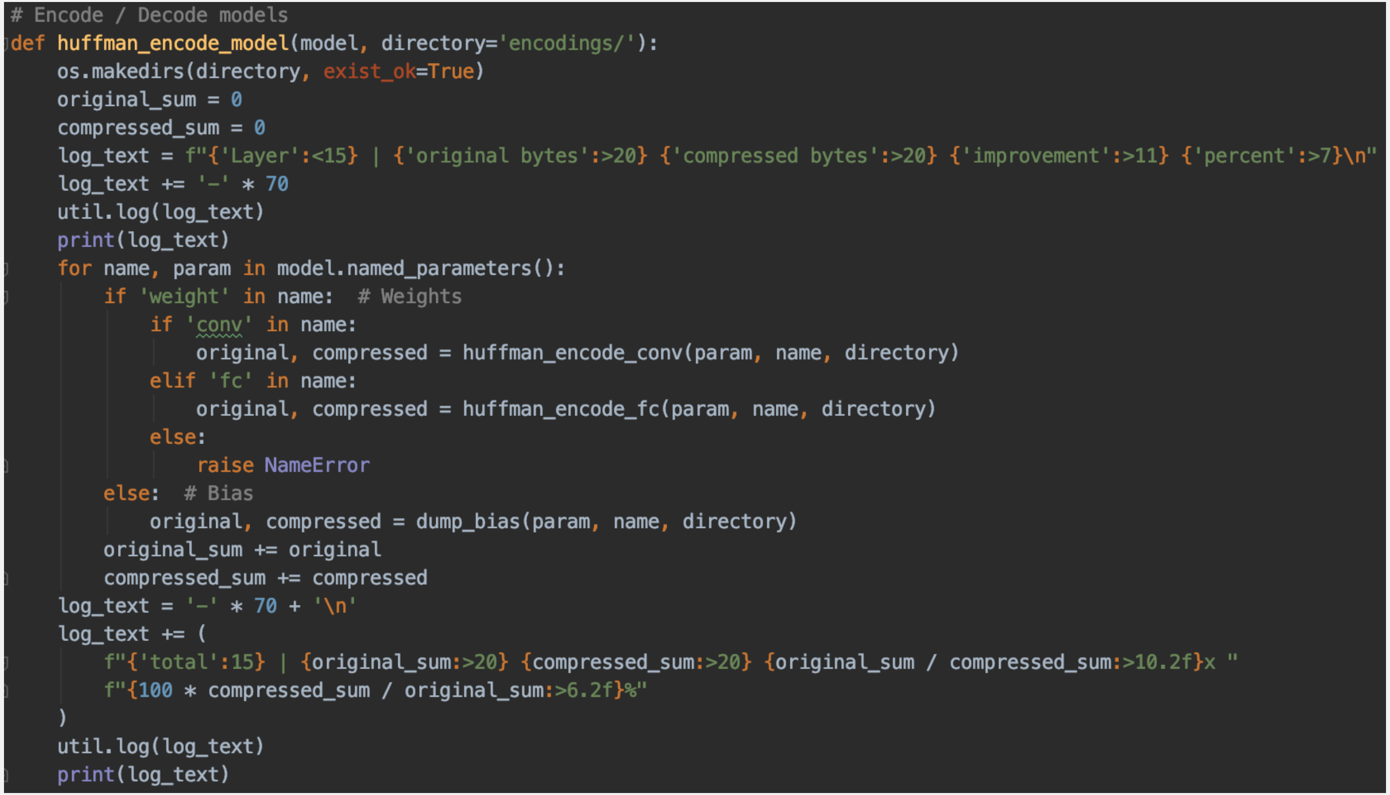
initial_model.ptmodel
model_after_retraining.ptmodel
model_after_weight_sharing.ptmodel

Huffman coding

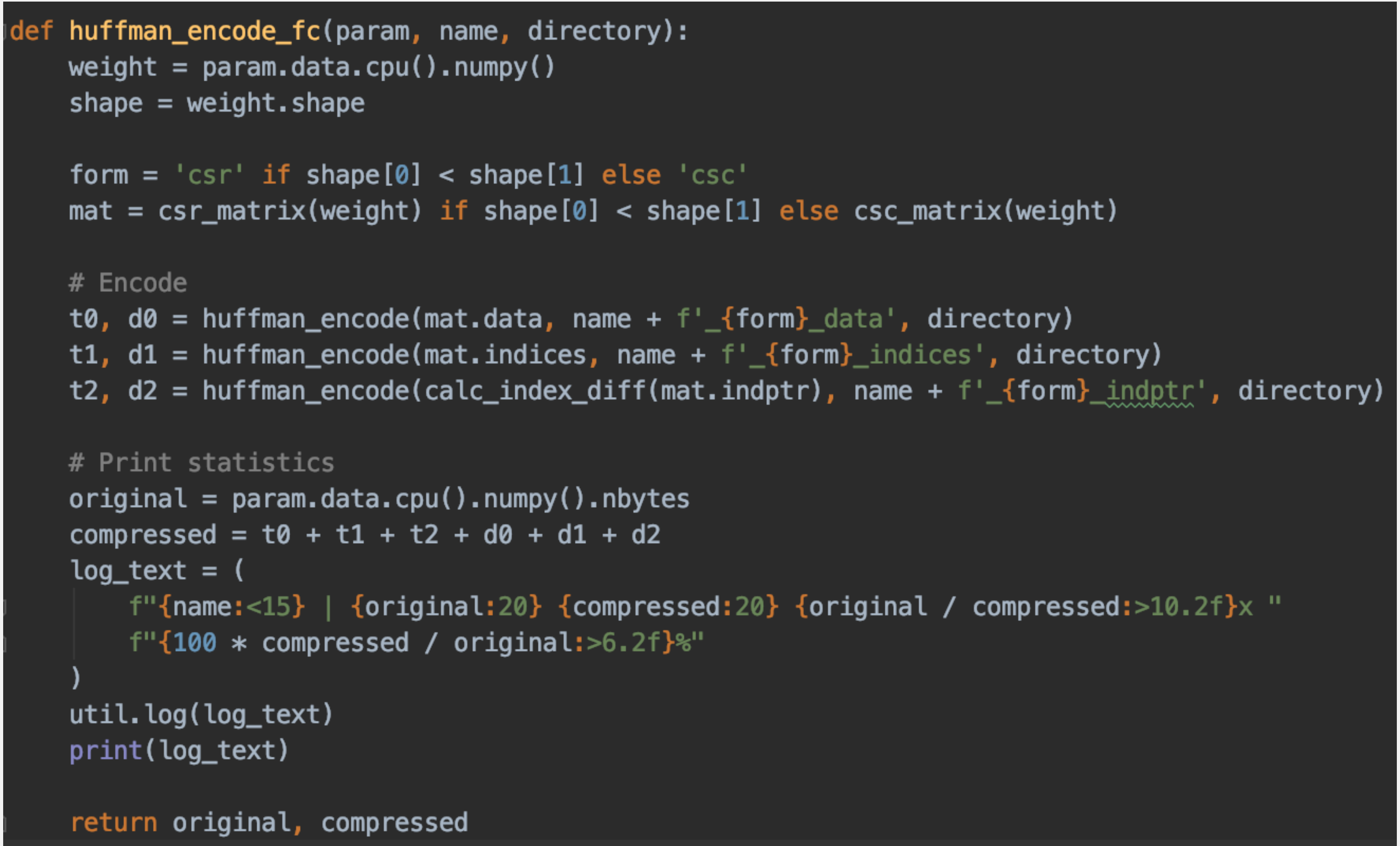
- huffmancoding.py
- 分別有 encode、decode 部分
- 且都只提供 fully connected 部分



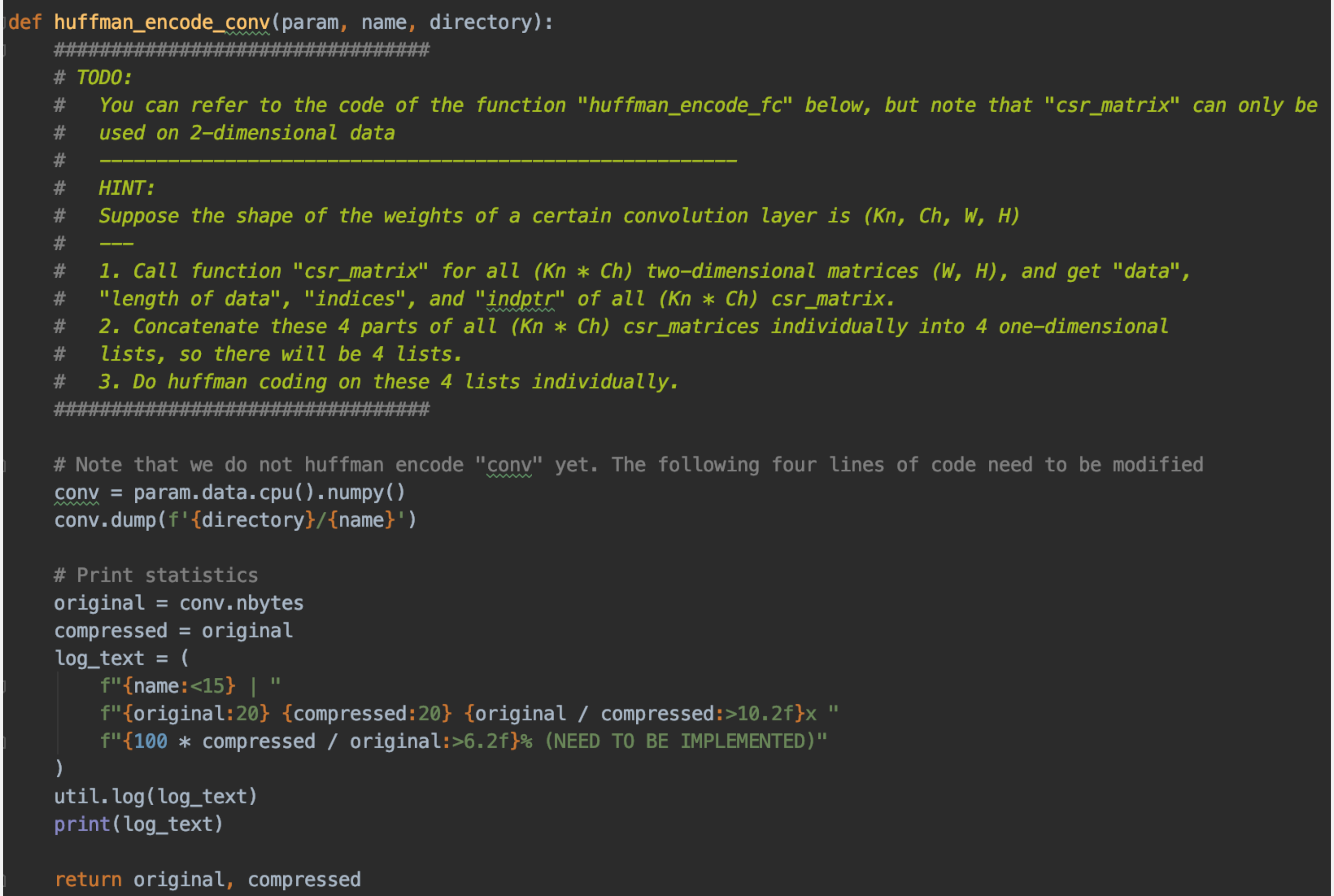
Huffman encode



```python
# Encode / Decode models
def huffman_encode_model(model, directory='encodings/'):
    os.makedirs(directory, exist_ok=True)
    original_sum = 0
    compressed_sum = 0
    log_text = f"{'Layer':<15} | {'original bytes':>20} {'compressed bytes':>20} {'improvement':>11} {'percent':>7}\n"
    log_text += '-' * 70
    util.log(log_text)
    print(log_text)
    for name, param in model.named_parameters():
        if 'weight' in name:  # Weights
            if 'conv' in name:
                original, compressed = huffman_encode_conv(param, name, directory)
            elif 'fc' in name:
                original, compressed = huffman_encode_fc(param, name, directory)
            else:
                raise NameError
        else:  # Bias
            original, compressed = dump_bias(param, name, directory)
        original_sum += original
        compressed_sum += compressed
    log_text = '-' * 70 + '\n'
    log_text += (
        f"{'total':15} | {original_sum:>20} {compressed_sum:>20} {original_sum / compressed_sum:>10.2f}x "
        f"{100 * compressed_sum / original_sum:>6.2f}%"
    )
    util.log(log_text)
    print(log_text)
```

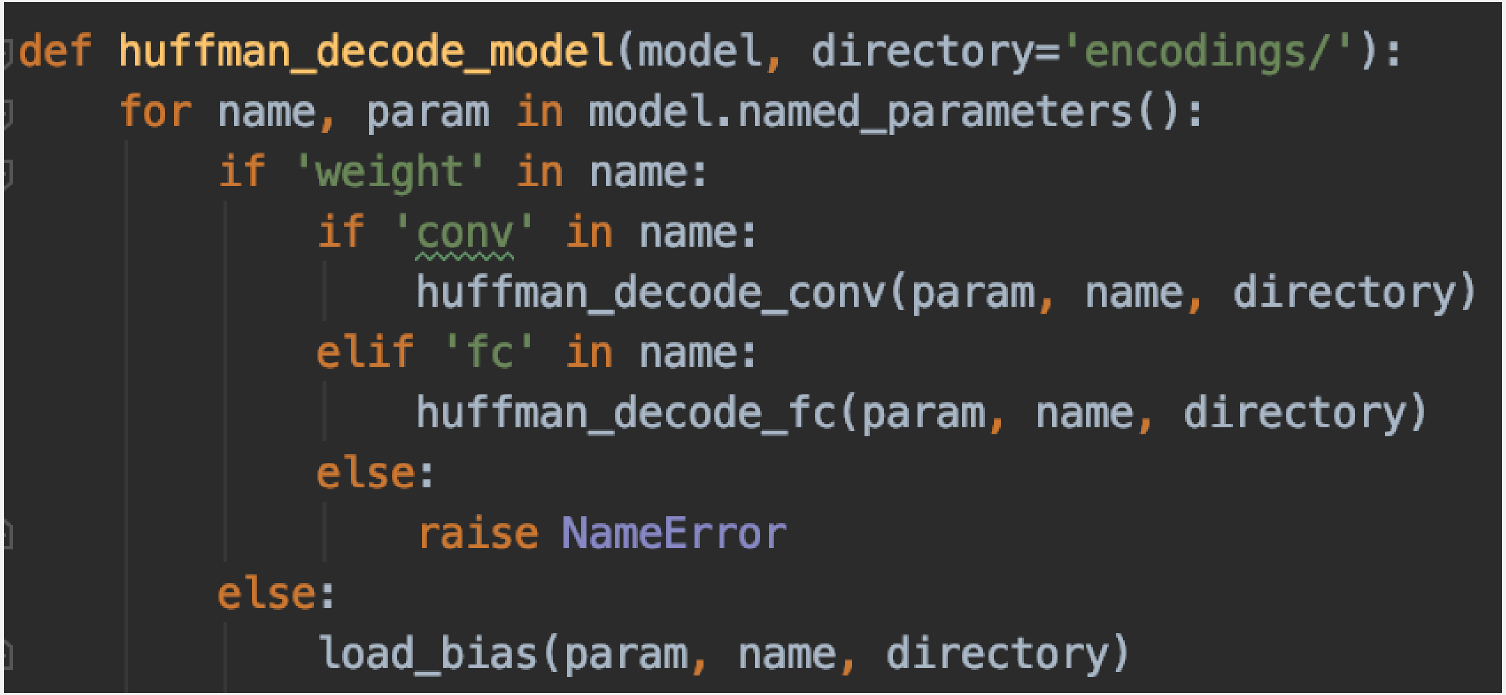


```python
def huffman_encode_fc(param, name, directory):
    weight = param.data.cpu().numpy()
    shape = weight.shape

    form = 'csr' if shape[0] < shape[1] else 'csc'
    mat = csr_matrix(weight) if shape[0] < shape[1] else csc_matrix(weight)

    # Encode
    t0, d0 = huffman_encode(mat.data, name + f'_{form}_data', directory)
    t1, d1 = huffman_encode(mat.indices, name + f'_{form}_indices', directory)
    t2, d2 = huffman_encode(calc_index_diff(mat.indptr), name + f'_{form}_indptr', directory)

    # Print statistics
    original = param.data.cpu().numpy().nbytes
    compressed = t0 + t1 + t2 + d0 + d1 + d2
    log_text = (
        f"{name:<15} | {original:20} {compressed:20} {original / compressed:>10.2f}x "
        f"{100 * compressed / original:>6.2f}%"
    )
    util.log(log_text)
    print(log_text)

    return original, compressed
```

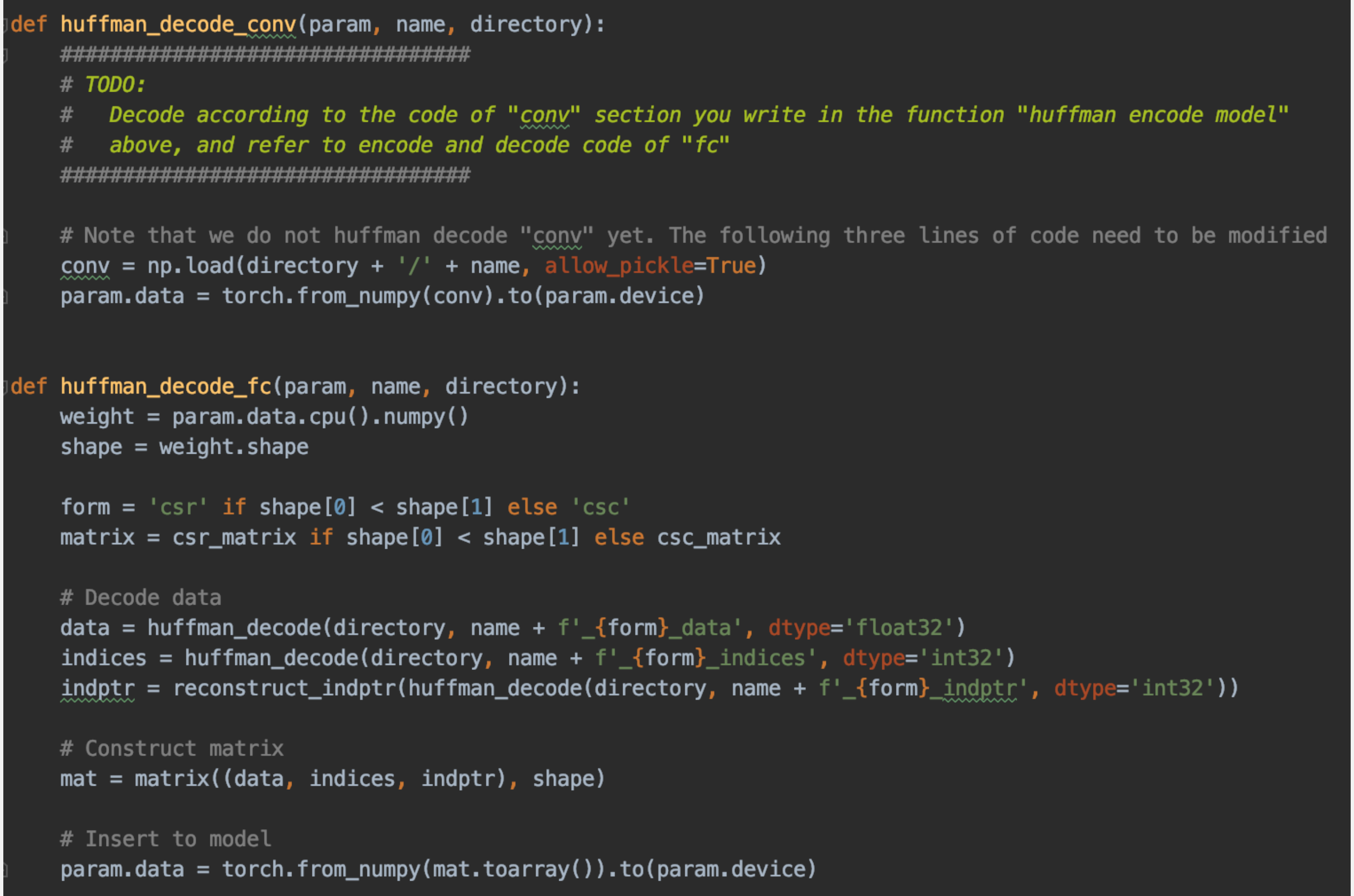


```python
def huffman_encode_conv(param, name, directory):
    #################################
    # TODO:
    #   You can refer to the code of the function "huffman_encode_fc" below, but note that "csr_matrix" can only be
    #   used on 2-dimensional data
    #   --------------------------------------------------------
    #   HINT:
    #   Suppose the shape of the weights of a certain convolution layer is (Kn, Ch, W, H)
    #   ---
    #   1. Call function "csr_matrix" for all (Kn * Ch) two-dimensional matrices (W, H), and get "data",
    #   "length of data", "indices", and "indptr" of all (Kn * Ch) csr_matrix.
    #   2. Concatenate these 4 parts of all (Kn * Ch) csr_matrices individually into 4 one-dimensional
    #   lists, so there will be 4 lists.
    #   3. Do huffman coding on these 4 lists individually.
    #################################

    # Note that we do not huffman encode "conv" yet. The following four lines of code need to be modified
    conv = param.data.cpu().numpy()
    conv.dump(f'{directory}/{name}')

    # Print statistics
    original = conv.nbytes
    compressed = original
    log_text = (
        f"{name:<15} | "
        f"{original:20} {compressed:20} {original / compressed:>10.2f}x "
        f"{100 * compressed / original:>6.2f}% (NEED TO BE IMPLEMENTED)"
    )
    util.log(log_text)
    print(log_text)

    return original, compressed
```

Huffman decode



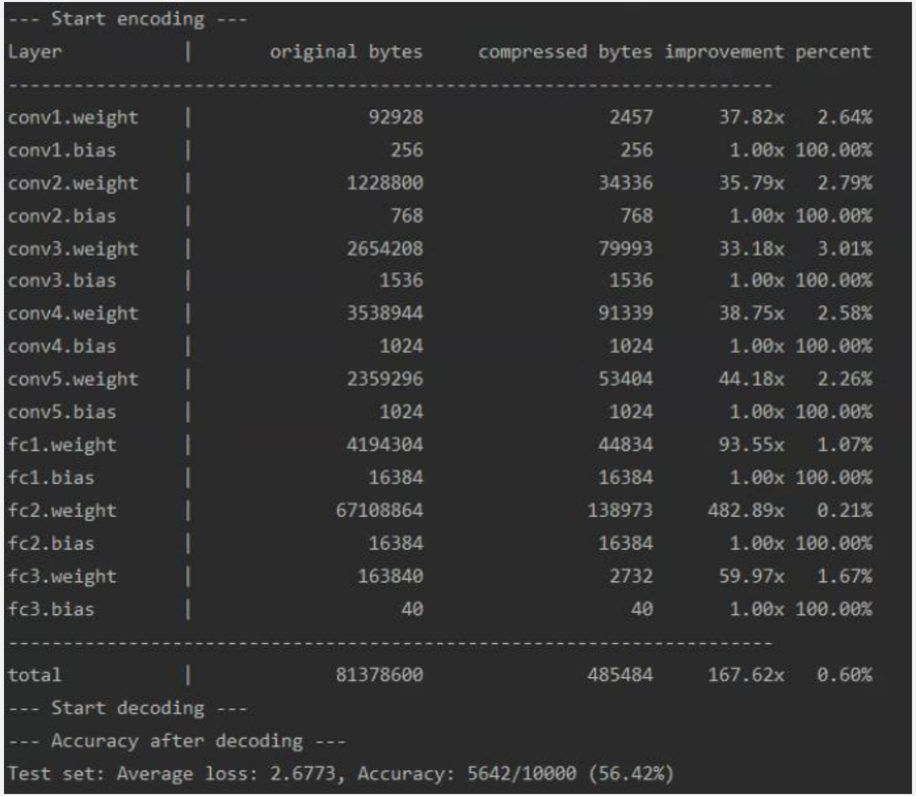
```python
def huffman_decode_model(model, directory='encodings/'):
    for name, param in model.named_parameters():
        if 'weight' in name:
            if 'conv' in name:
                huffman_decode_conv(param, name, directory)
            elif 'fc' in name:
                huffman_decode_fc(param, name, directory)
            else:
                raise NameError
        else:
            load_bias(param, name, directory)
```



```
def huffman_decode_conv(param, name, directory):
    #################################
    # TODO:
    #   Decode according to the code of "conv" section you write in the function "huffman encode model"
    #   above, and refer to encode and decode code of "fc"
    #################################

    # Note that we do not huffman decode "conv" yet. The following three lines of code need to be modified
    conv = np.load(directory + '/' + name, allow_pickle=True)
    param.data = torch.from_numpy(conv).to(param.device)


def huffman_decode_fc(param, name, directory):
    weight = param.data.cpu().numpy()
    shape = weight.shape

    form = 'csr' if shape[0] < shape[1] else 'csc'
    matrix = csr_matrix if shape[0] < shape[1] else csc_matrix

    # Decode data
    data = huffman_decode(directory, name + f'_{form}_data', dtype='float32')
    indices = huffman_decode(directory, name + f'_{form}_indices', dtype='int32')
    indptr = reconstruct_indptr(huffman_decode(directory, name + f'_{form}_indptr', dtype='int32'))

    # Construct matrix
    mat = matrix((data, indices, indptr), shape)

    # Insert to model
    param.data = torch.from_numpy(mat.toarray()).to(param.device)
```

Run : huffman_coding.py

- 觀察結果 :
  - 壓縮率
  - accuracy



```
--- Start encoding ---
Layer            |        original bytes     compressed bytes improvement percent
----------------------------------------------------------------------
conv1.weight     |                92928                 2457      37.82x   2.64%
conv1.bias       |                  256                  256       1.00x 100.00%
conv2.weight     |              1228800                34336      35.79x   2.79%
conv2.bias       |                  768                  768       1.00x 100.00%
conv3.weight     |              2654208                79993      33.18x   3.01%
conv3.bias       |                 1536                 1536       1.00x 100.00%
conv4.weight     |              3538944                91339      38.75x   2.58%
conv4.bias       |                 1024                 1024       1.00x 100.00%
conv5.weight     |              2359296                53404      44.18x   2.26%
conv5.bias       |                 1024                 1024       1.00x 100.00%
fc1.weight       |              4194304                44834      93.55x   1.07%
fc1.bias         |                16384                16384       1.00x 100.00%
fc2.weight       |             67108864               138973     482.89x   0.21%
fc2.bias         |                16384                16384       1.00x 100.00%
fc3.weight       |               163840                 2732      59.97x   1.67%
fc3.bias         |                   40                   40       1.00x 100.00%
----------------------------------------------------------------------
total            |             81378600               485484     167.62x   0.60%
--- Start decoding ---
--- Accuracy after decoding ---
Test set: Average loss: 2.6773, Accuracy: 5642/10000 (56.42%)
```

3

繳交內容

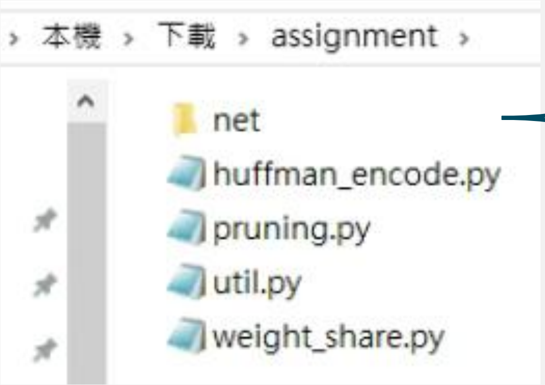
TODO

- 有 TODO 的 py 檔：
  - prune.py
  - pruning.py
  - quantization.py
  - huffmancoding.py

- 請根據 TODO 的提示內容完成程式

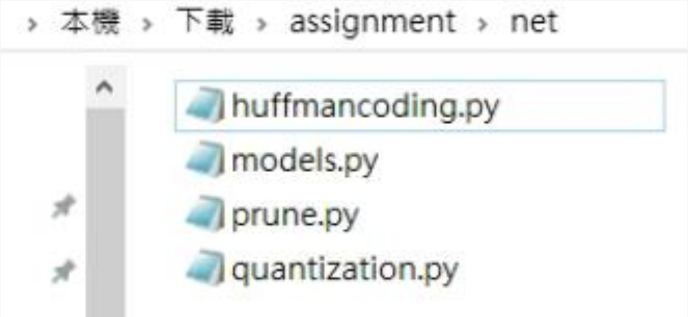
繳交規格

- 原本拿到的



› 本機 › 下載 › assignment ›

net
huffman_encode.py
pruning.py
util.py
weight_share.py



› 本機 › 下載 › assignment › net

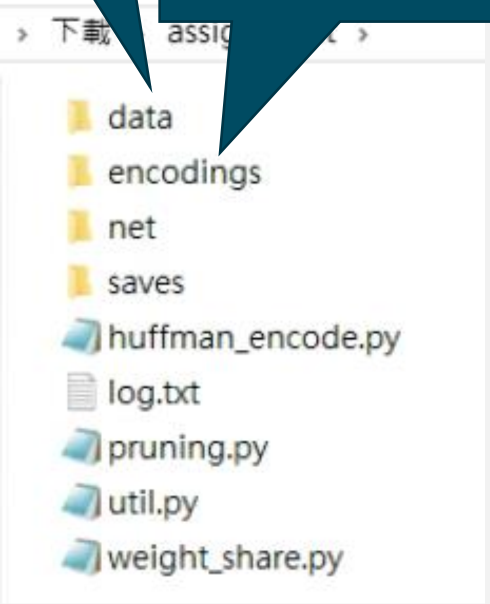
huffmancoding.py
models.py
prune.py
quantization.py

繳交規格

- 跑完程式後，應該長這樣

torchvision 下載的 data

huffmancoding 所生成



› 下載 › assig ›

data

encodings

net

saves

huffman_encode.py

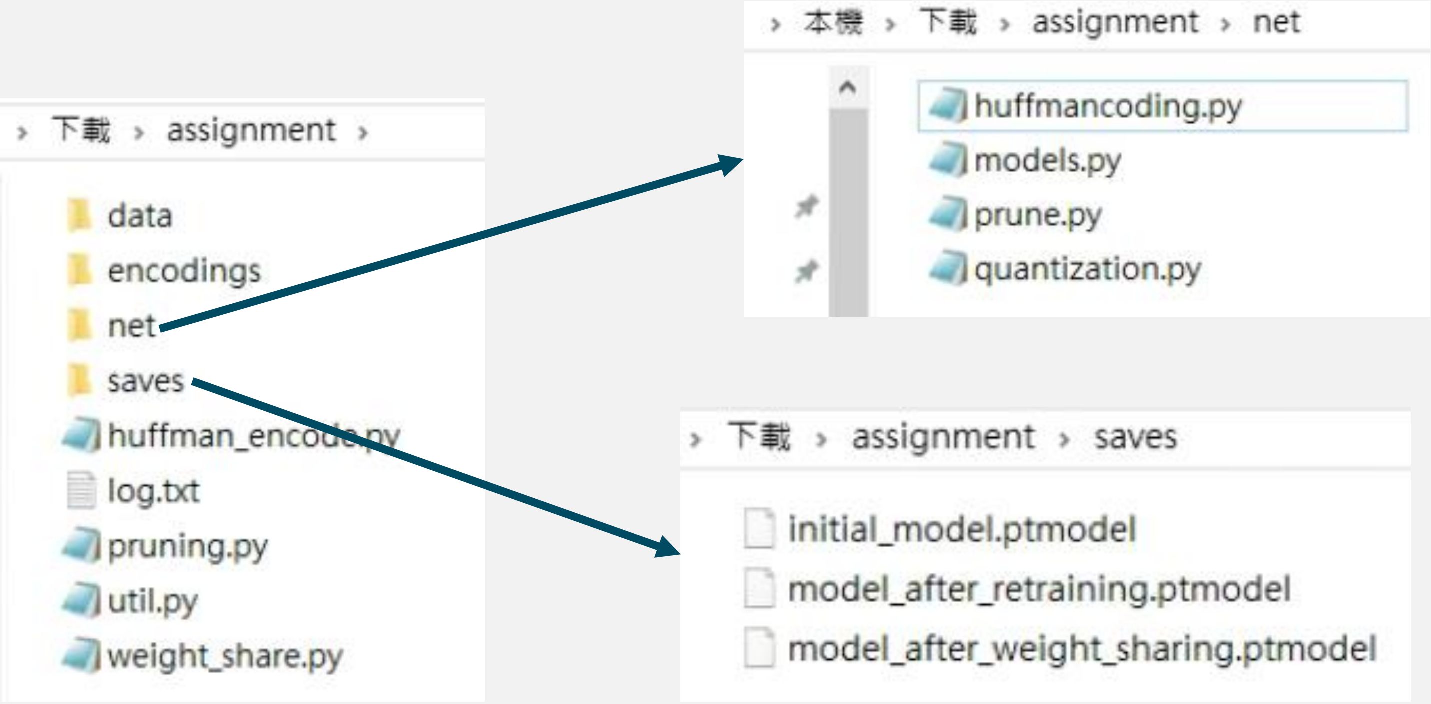
log.txt

pruning.py

util.py

weight_share.py

繳交規格



› 下載 › assignment ›

data
encodings
net
saves
huffman_encode.py
log.txt
pruning.py
util.py
weight_share.py

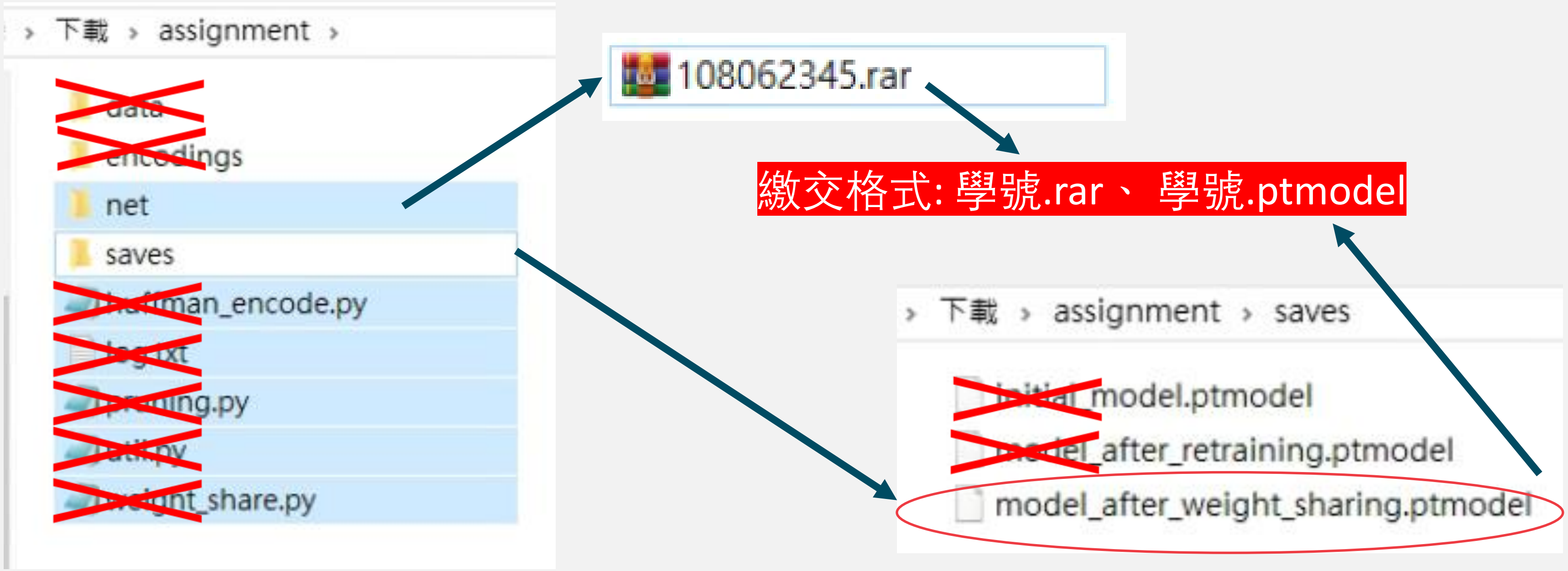
› 本機 › 下載 › assignment › net

huffmancoding.py
models.py
prune.py
quantization.py

› 下載 › assignment › saves

initial_model.ptmodel
model_after_retraining.ptmodel
model_after_weight_sharing.ptmodel

繳交規格



› 下載 › assignment ›

- ~~data~~
- ~~encodings~~
- net
- saves
- ~~huffman_encode.py~~
- ~~log.txt~~
- ~~pruning.py~~
- ~~util.py~~
- ~~weight_share.py~~

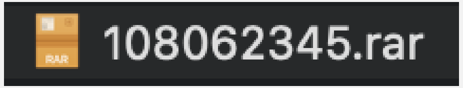
108062345.rar

繳交格式: 學號.rar、 學號.ptmodel

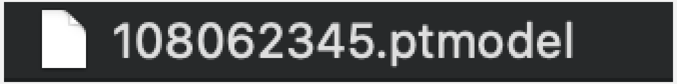
› 下載 › assignment › saves

- ~~initial_model.ptmodel~~
- ~~model_after_retraining.ptmodel~~
- model_after_weight_sharing.ptmodel

繳交規格



108062345.rar



108062345.ptmodel

最終繳交這兩項

4

配分標準

配分

- (70%) Prune model : 完成 prune.py、pruning.py 中 # TODO

- (10%) Quantize model : 完成 quantization.py 中 # TODO

- (10%) Huffman coding : 完成 huffmancoding.py 中 # TODO

- (10%) 壓縮率排名 : accuracy > 58%，最終模型 compression rate 越高越好 ! 排名前1/3得10分、中1/3得5分、後1/3得0分，會用同學繳回的模型計算，排名改完會公布於 eeclass

Thank you

鼓勵討論，嚴禁抄襲