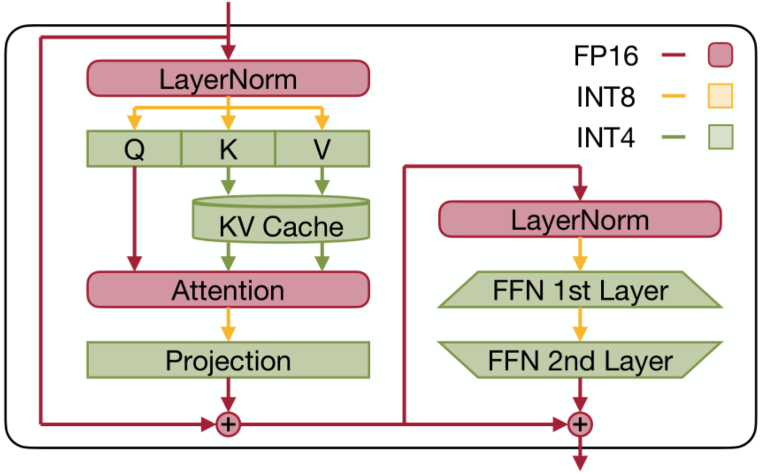
On-Device AI 실습: Pruning for LLM

Dongkun Shin
Intelligent Embedded Systems Lab.
Sungkyunkwan University

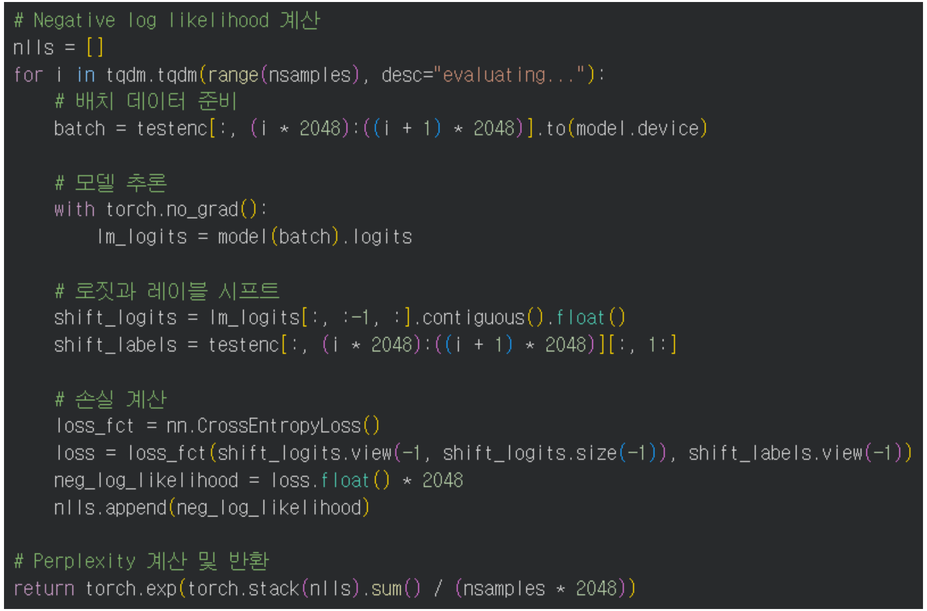
Post-Training Pruning for LLMs

- **LLM Pruning 시 pruning 대상은 Multi-head attention의 projection layer (FC) 와 FFN의 FC layer들임.**



LLM Architecture

Perplexity



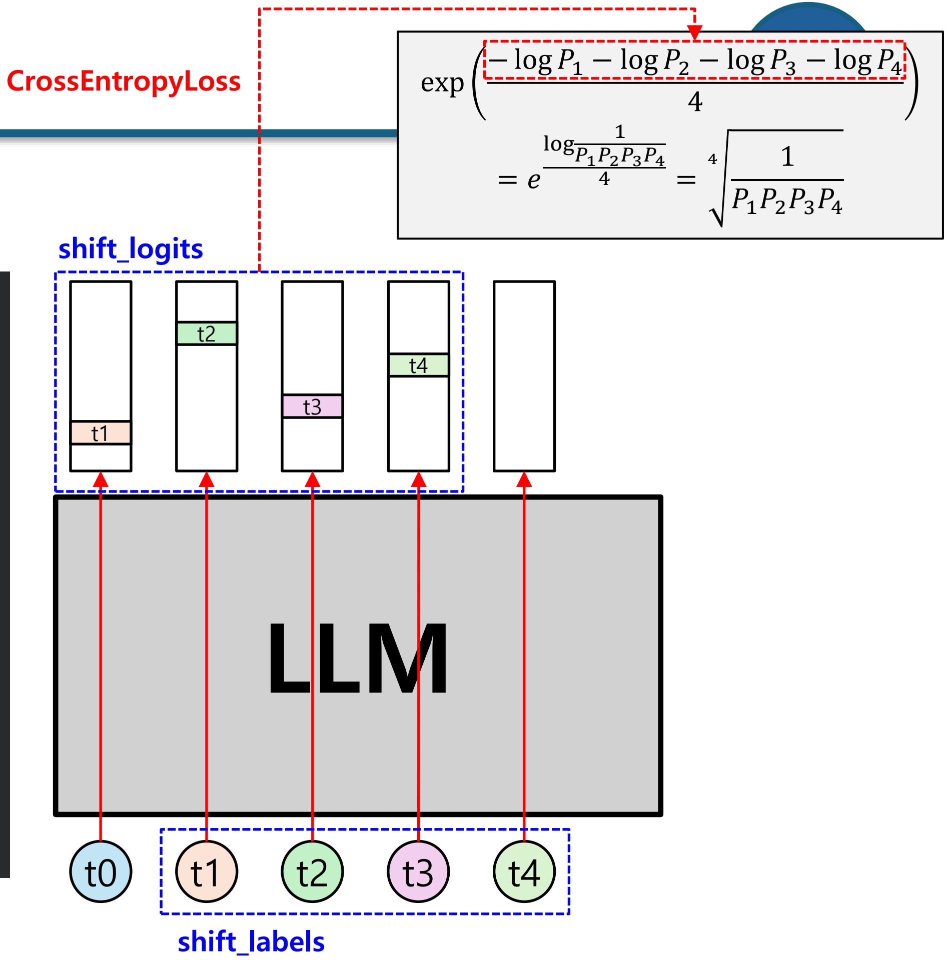
```
# Negative log likelihood 계산
nlls = []
for i in tqdm.tqdm(range(nsamples), desc="evaluating..."):
    # 배치 데이터 준비
    batch = testenc[:, (i * 2048):((i + 1) * 2048)].to(model.device)

    # 모델 추론
    with torch.no_grad():
        lm_logits = model(batch).logits

    # 로짓과 레이블 시프트
    shift_logits = lm_logits[:, :-1, :].contiguous().float()
    shift_labels = testenc[:, (i * 2048):((i + 1) * 2048)][:, 1:]

    # 손실 계산
    loss_fct = nn.CrossEntropyLoss()
    loss = loss_fct(shift_logits.view(-1, shift_logits.size(-1)), shift_labels.view(-1))
    neg_log_likelihood = loss.float() * 2048
    nlls.append(neg_log_likelihood)

# Perplexity 계산 및 반환
return torch.exp(torch.stack(nlls).sum() / (nsamples * 2048))
```

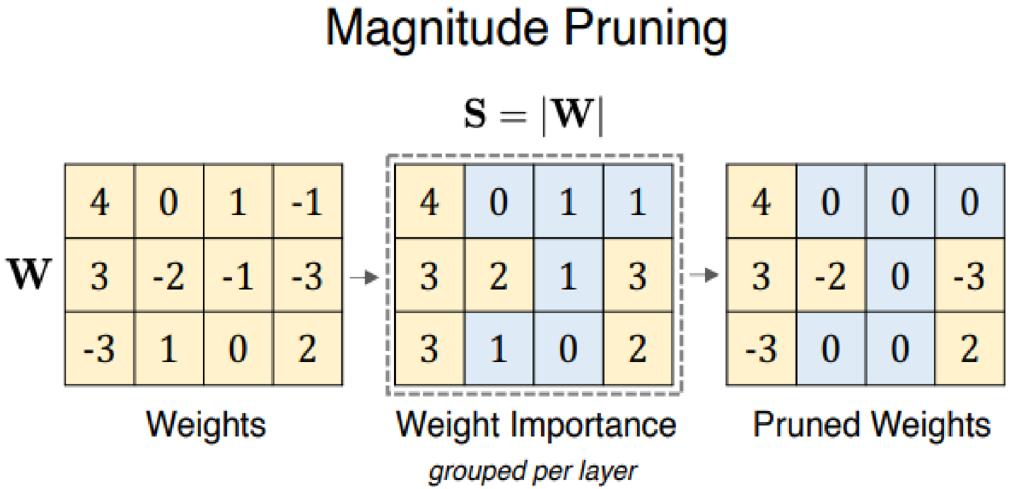


$$\exp\left(\frac{-\log P_1 - \log P_2 - \log P_3 - \log P_4}{4}\right)$$

$$= e^{\frac{\log\frac{1}{P_1 P_2 P_3 P_4}}{4}} = \sqrt[4]{\frac{1}{P_1 P_2 P_3 P_4}}$$

[실습 1] Magnitude-based Pruning 구현

- Weight의 magnitude 만을 이용하여 pruning 수행



Magnitude Pruning

$\mathbf{S} = |\mathbf{W}|$

W

4	0	1	-1
3	-2	-1	-3
-3	1	0	2

Weights

4	0	1	1
3	2	1	3
3	1	0	2

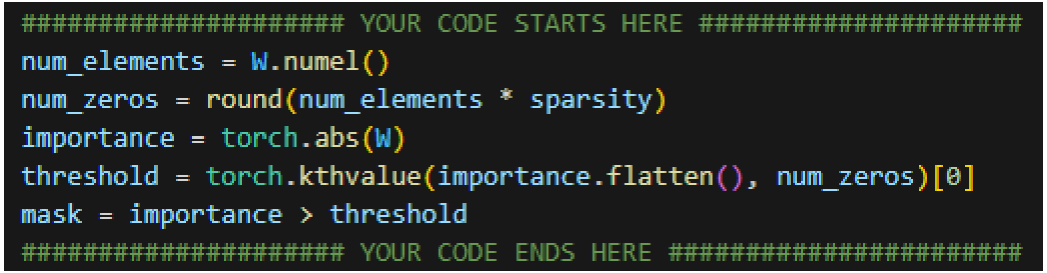
Weight Importance

grouped per layer

4	0	0	0
3	-2	0	-3
-3	0	0	2

Pruned Weights

[실습 1] Magnitude-based Pruning

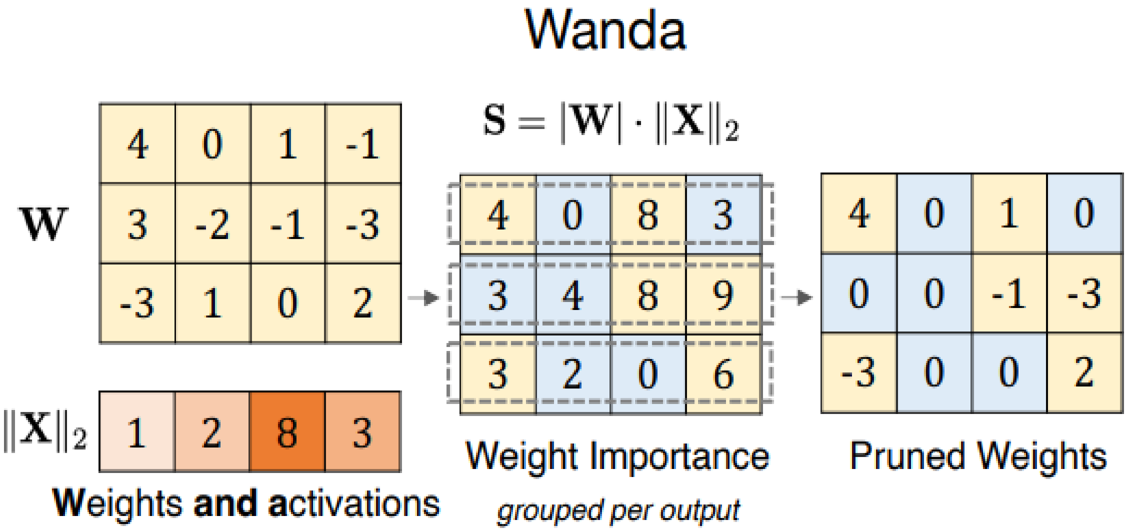


```
##################### YOUR CODE STARTS HERE #####################
num_elements = W.numel()
num_zeros = round(num_elements * sparsity)
importance = torch.abs(W)
threshold = torch.kthvalue(importance.flatten(), num_zeros)[0]
mask = importance > threshold
##################### YOUR CODE ENDS HERE #######################
```

[실습 2, 3] Wanda Pruning 구현

- Calibration을 통한 feature의 값을 sampling

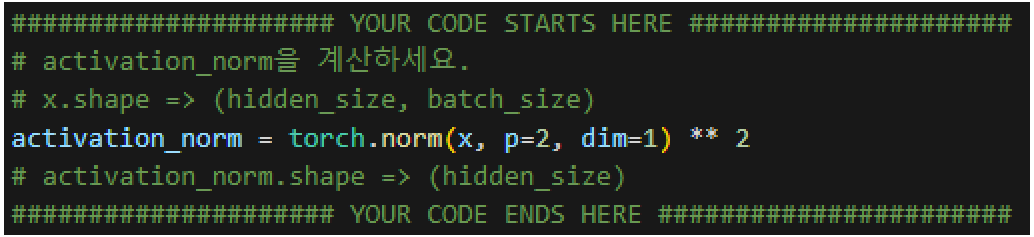
$$\|\mathbf{X}\|_2 = \sqrt{\sum_i \mathbf{X}_i^2}$$



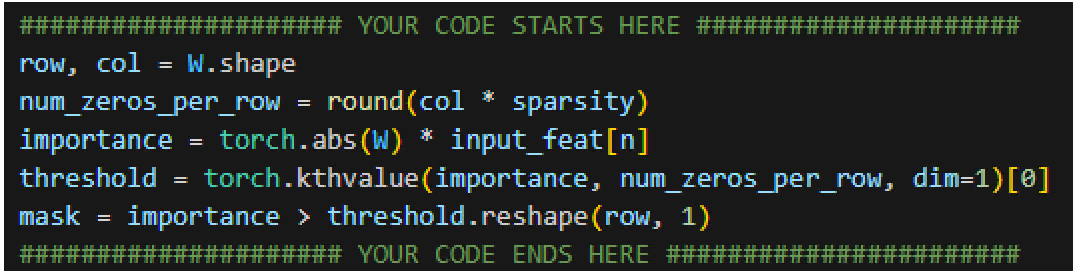
$$\mathbf{W} = \begin{bmatrix} 4 & 0 & 1 & -1 \\ 3 & -2 & -1 & -3 \\ -3 & 1 & 0 & 2 \end{bmatrix} \quad \|\mathbf{X}\|_2 = \begin{bmatrix} 1 & 2 & 8 & 3 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 4 & 0 & 8 & 3 \\ 3 & 4 & 8 & 9 \\ 3 & 2 & 0 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 0 & 1 & 0 \\ 0 & 0 & -1 & -3 \\ -3 & 0 & 0 & 2 \end{bmatrix}$$

[실습 2, 3] Answer



```
###################### YOUR CODE STARTS HERE ######################
# activation_norm을 계산하세요.
# x.shape => (hidden_size, batch_size)
activation_norm = torch.norm(x, p=2, dim=1) ** 2
# activation_norm.shape => (hidden_size)
###################### YOUR CODE ENDS HERE ########################
```



```
###################### YOUR CODE STARTS HERE ######################
row, col = W.shape
num_zeros_per_row = round(col * sparsity)
importance = torch.abs(W) * input_feat[n]
threshold = torch.kthvalue(importance, num_zeros_per_row, dim=1)[0]
mask = importance > threshold.reshape(row, 1)
###################### YOUR CODE ENDS HERE ########################
```