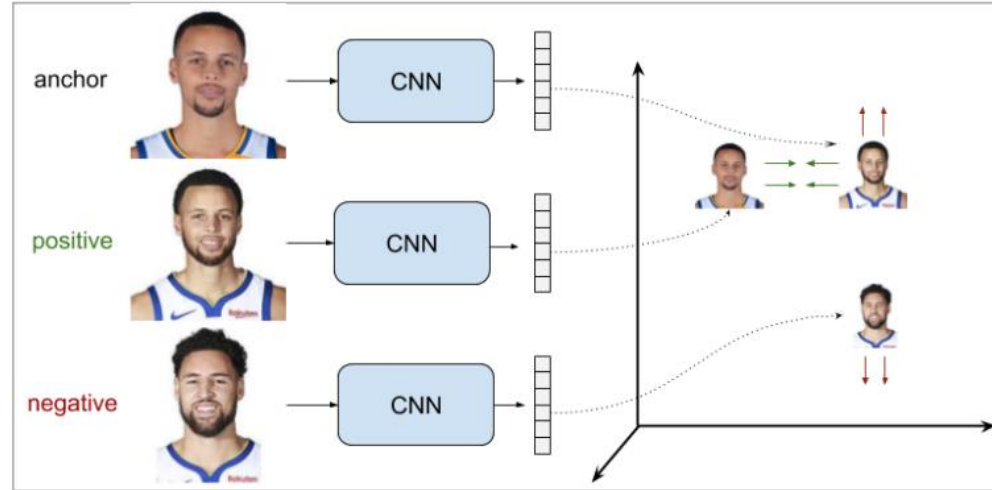


This week

- *Transformer based ReID*
 - *Patch-wise triplet loss*
- *Relation between CLS token & Triplet loss*

Back ground

➤ Triplet loss



- 1) Extract features from **three input data points(anchor, positive, negative)** and get an embedded representation for each of them
- 2) Define a **metric function** to measure the similarity between those representations, for instance Euclidian distance
- 3) Train the extractors to produce **similar representations for anchor and positive**, and **distant representations for anchor and negative**

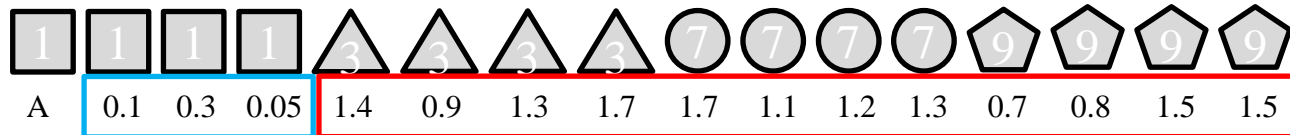
Back ground

➤ Triplet hard mining

- Ex) Batch size 16



- Compute distance matrix for each pair → 16 x 16
- Anchor : Current sample
- Positive : The most dissimilar sample among which have same ID with anchor
- Negative : The most similar sample among which have different ID with anchor






$$L(r_a, r_p, r_n) = \max(0, m + d(r_a, r_p) - d(r_a, r_n))$$

r_a, r_p, r_n : sample representations

d : distance function

Back ground

➤ Triplet hard mining

Triplet			
	A	P	N
Distance		0.3	0.7

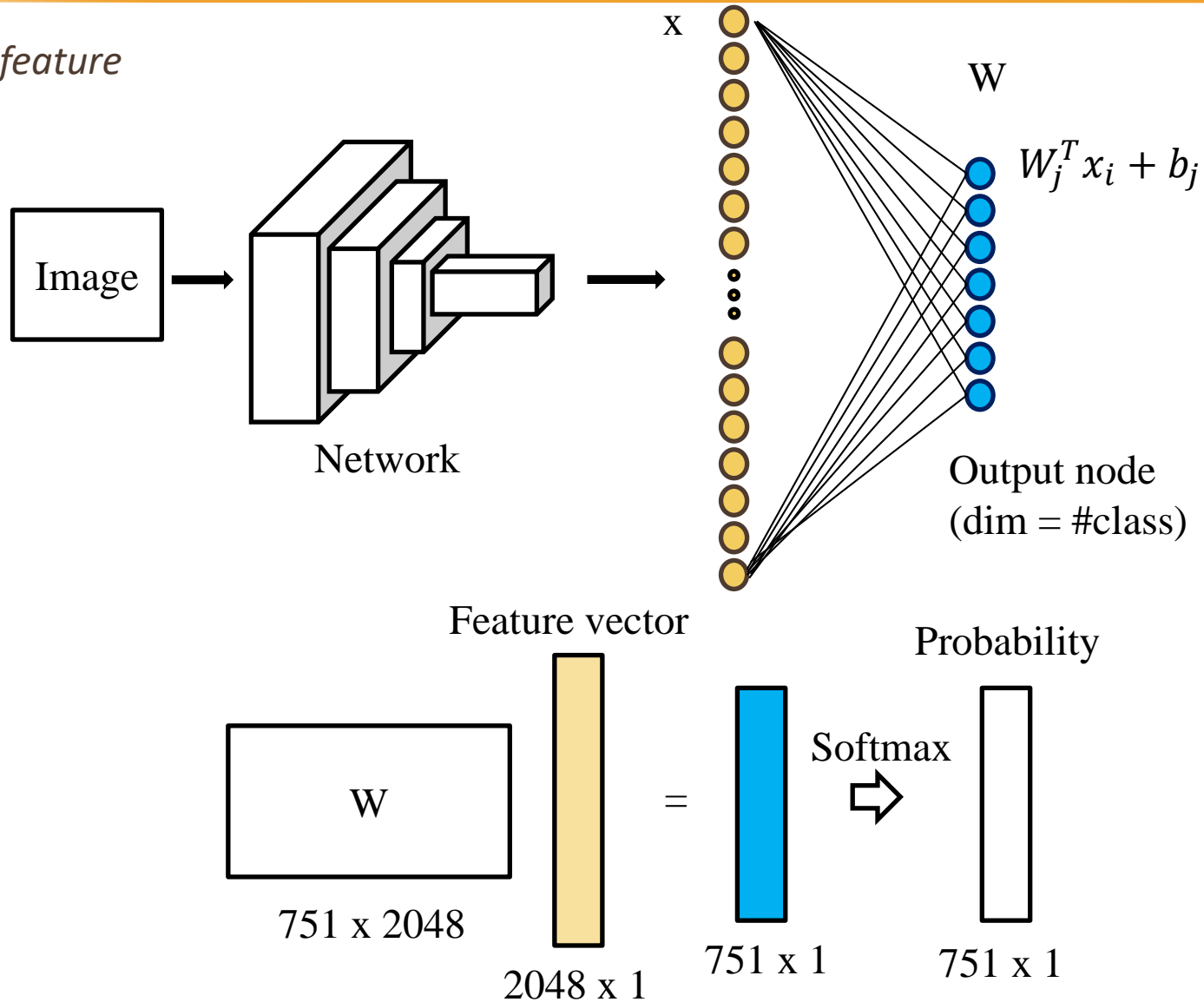
$$L(r_a, r_p, r_n) = \max(0, m + d(r_a, r_p) - d(r_a, r_n))$$

r_a, r_p, r_n : sample representations
 d : distance function



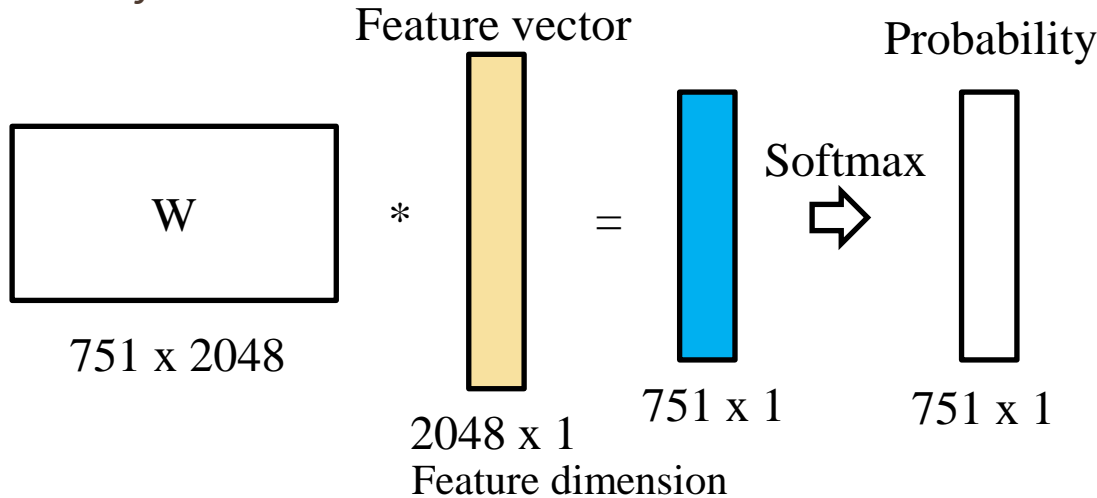
Background

➤ Class feature



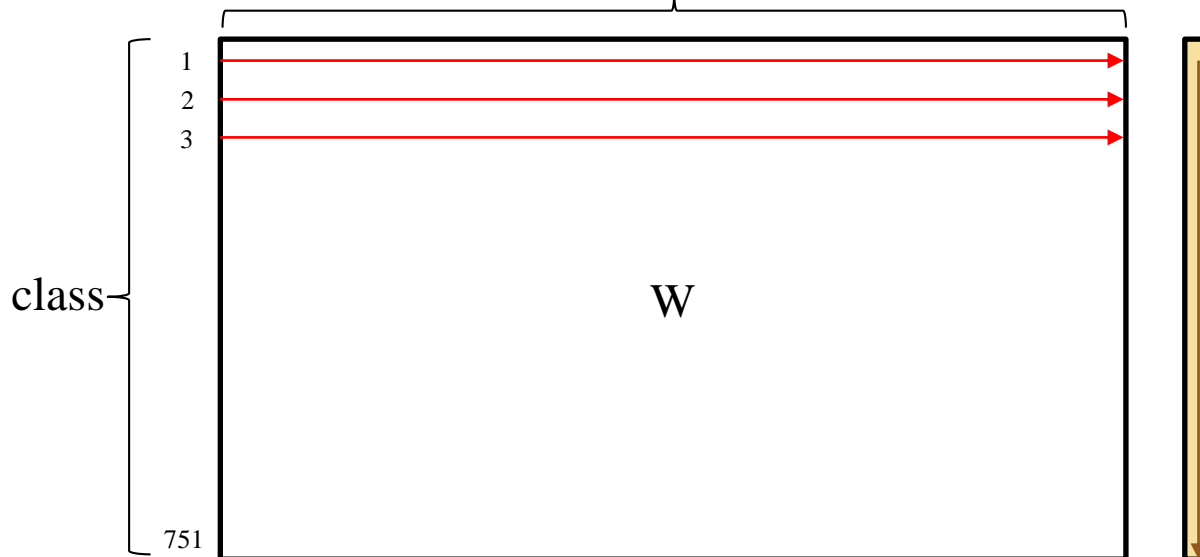
Background

➤ Class feature



$$W_j^T x_i = \|W_j\| \|x_i\| \cos \theta_j$$

j : class index
 i : sample index



*Note

Training이 거듭될수록, Weight vector의 각 row는 해당 class를 대표하는 feature vector의 값에 가깝게 update됨

Back ground

➤ Drawback of Triplet loss

- Negative feature vector의 어떤 feature가 Anchor와 차이가 나는지 알 수 없음



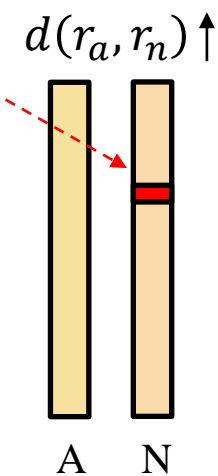
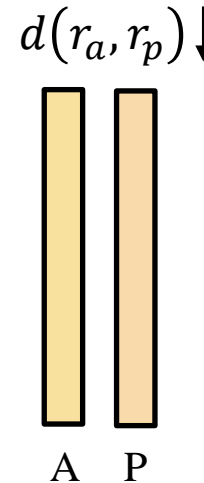
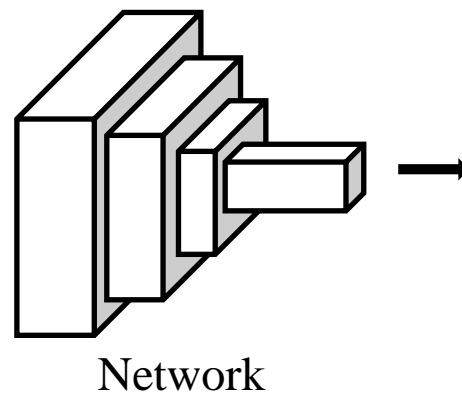
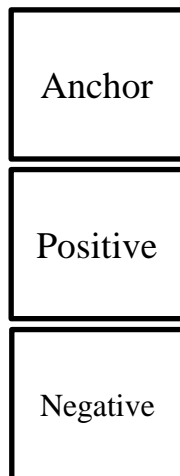
A

P

N

$$L(r_a, r_p, r_n) = \max(0, m + d(r_a, r_p) - d(r_a, r_n))$$

r_a, r_p, r_n : sample representations
 d : distance function

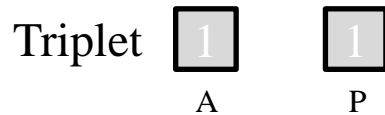
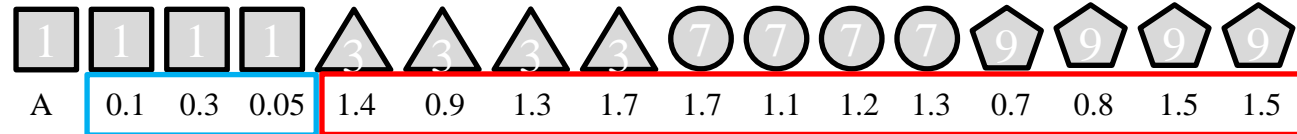


Background

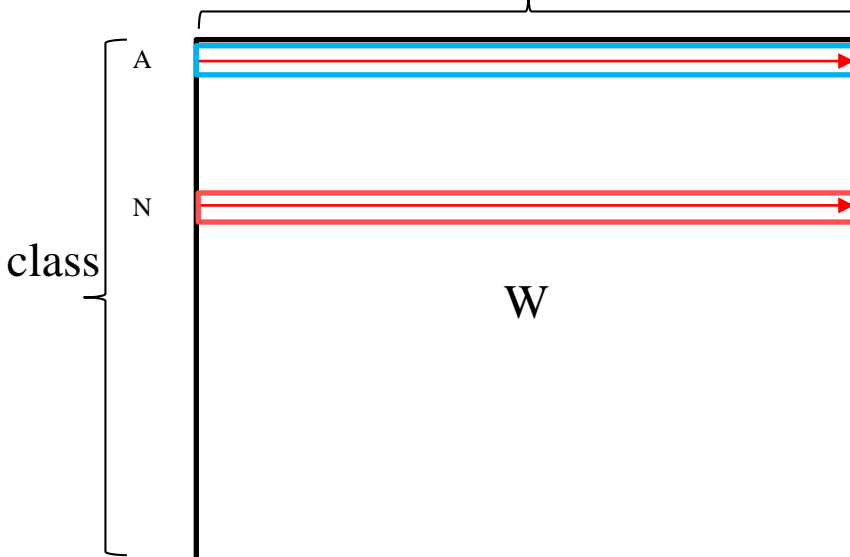
[1]Y. Lv, Y. Gu, and L. Xinggao, "The Dilemma of TriHard Loss and an Element-Weighted TriHard Loss for Person Re-Identification," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

➤ Using Class feature in Triplet loss

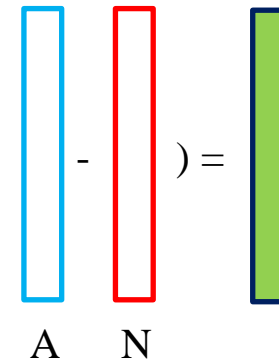
- Feature vector 간 distance를 구할 때 각 dimension에 weight를 다르게 부여



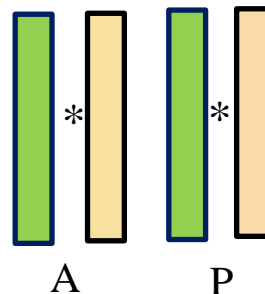
Feature dimension



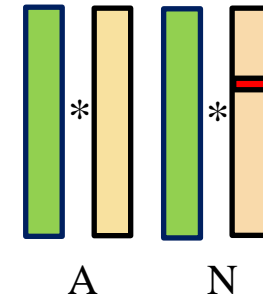
$$\text{Weight} = \text{Abs}(\text{A} - \text{N}) =$$



$d(r_a, r_p)$



$d(r_a, r_n)$



Element-wise weighted A, P, N

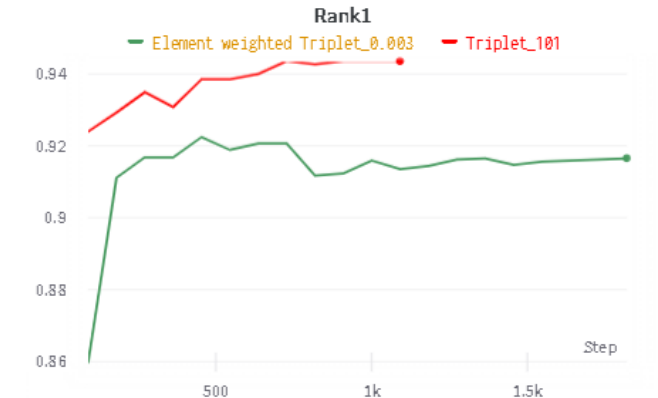
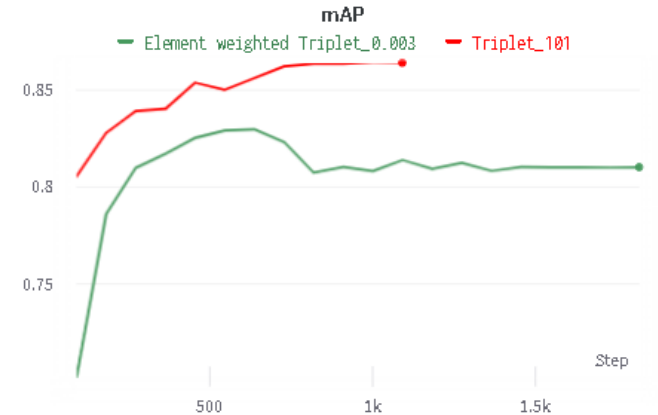
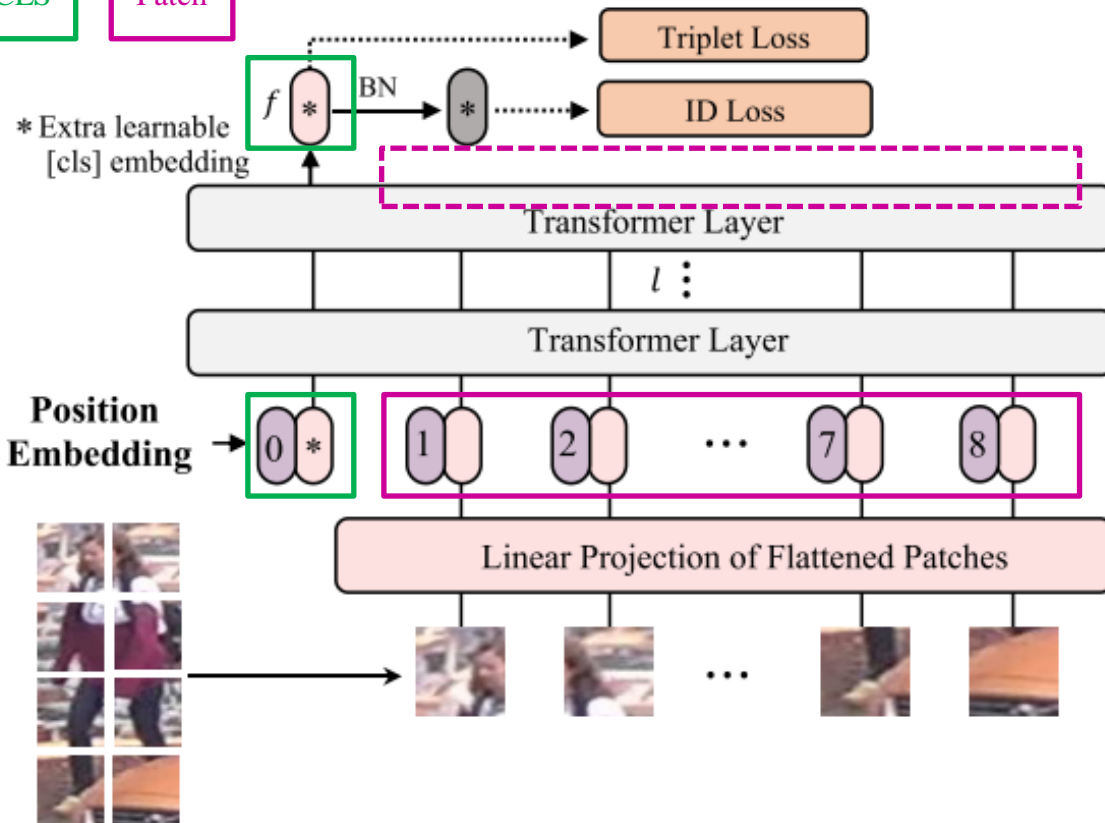
Transformer based Re-ID

➤ CLS token 을 이용해서 Triplet loss와 ID loss 계산

- CLS token feature 에도 Element-wise weighted Triplet loss 를 적용할 수 있지 않을까?

CLS

Patch



Transformer based Re-ID

- Elemented weighted triplet loss로 학습이 잘 되지 않은 이유에 대한 고찰
 - ViT의 특성 상 CLS token에 image의 feature가 직접적으로 반영되지 않은 것은 아닌가
 - 즉, CLS token은 patch의 image feature를 나타내는 embedding space와 구분된 space에 projection될 수도 있지않은가?
 - Image 전체의 feature를 나타내기 위해 patch feature와 같은 수의 dimension을 사용
 - CLS token 그자체가 learnable parameter
 - CLS token에 Image의 fine feature가 반영될수 있는가?

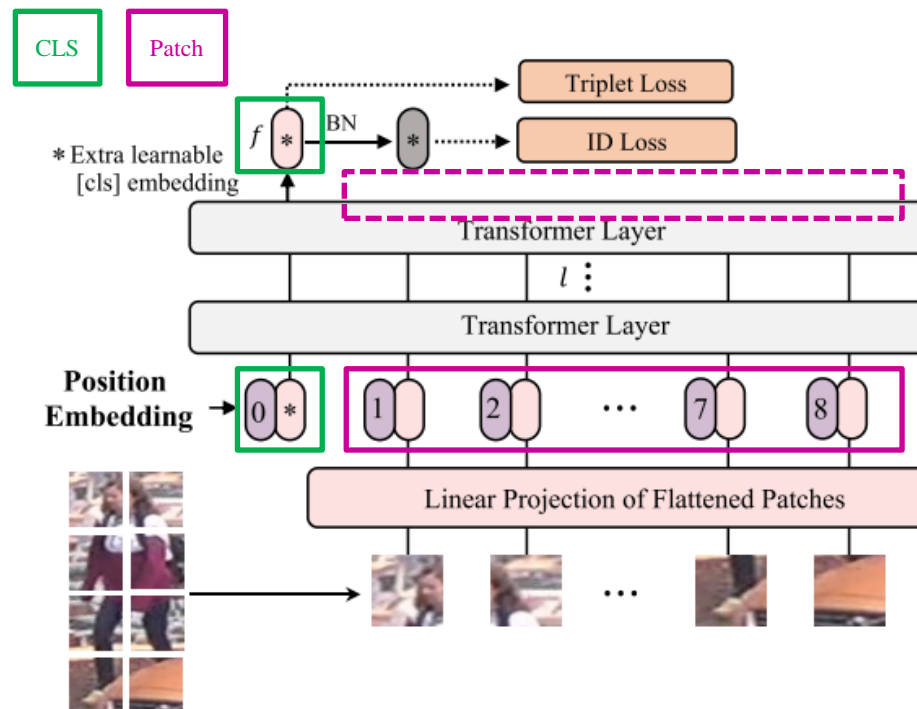


Image size $256 * 128$

Patch size $16 * 16$

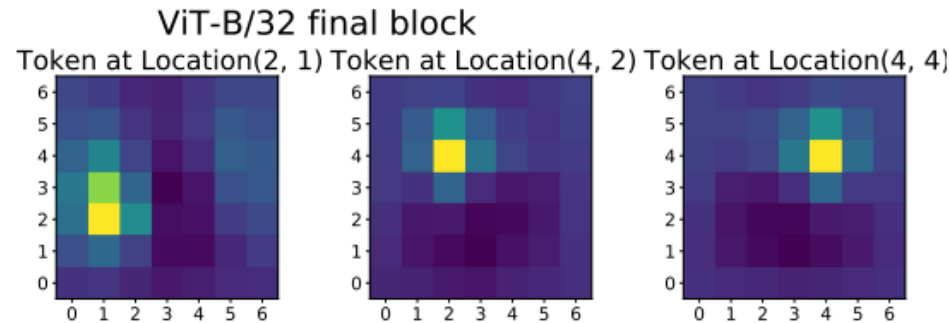
Patch Dimension = $16 * 16 * 3 = 768$

Patch $16 * 8 + \text{CLS } 1 = 129$

Transformer based Re-ID

➤ Patch – wise weighted triplet loss

- Patch token 들로 부터 image 의 feature를 뽑고, 이를 이용해서 triplet loss를 구성
- Global average pooling(Patch feature들의 평균) ? $128 * 768 \rightarrow 768$
- 각 Patch token은 상응하는 input patch와 가장 유사도가 높다는점을 이용



- 같은 position에 있는 A,P,N의 patch token들에 대해 patch wise(A-N,A-P)로 euclidean distance를 구한뒤, distance가 큰 patch만을 weighted global average pooling (Elemented weighted triplet loss의 Idea와 동일)



A

P

N

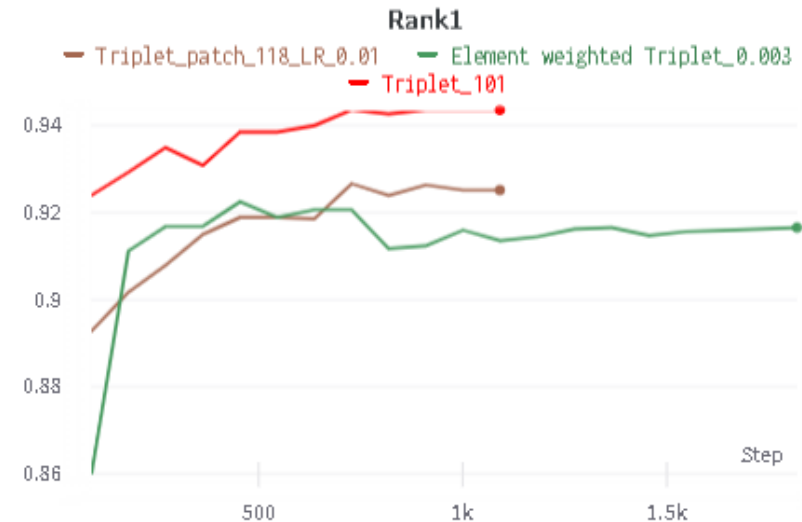
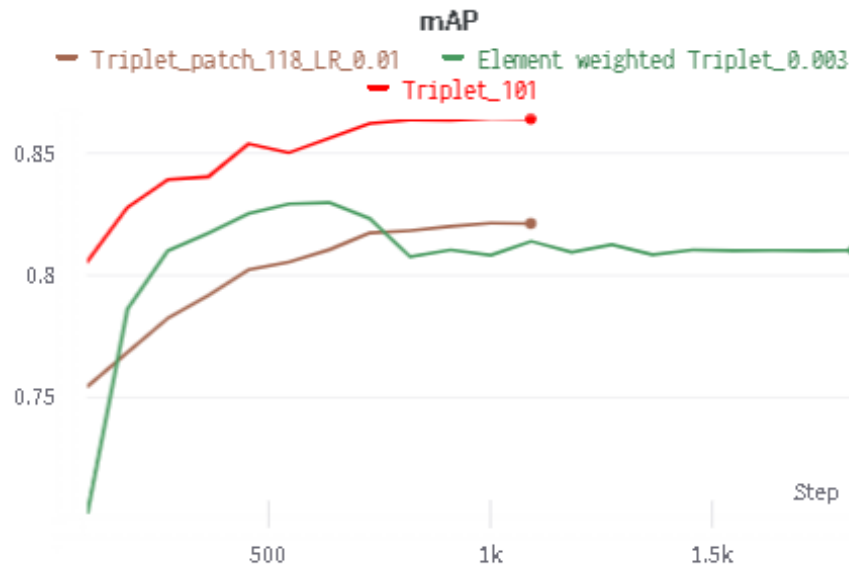
기본 idea :

1. 차이가 큰 patch일수록 해당 class를 대표할 수 있는 patch일 것
2. ID loss과 triplet loss의 분리

Transformer based Re-ID

➤ Patch – wise weighted triplet loss

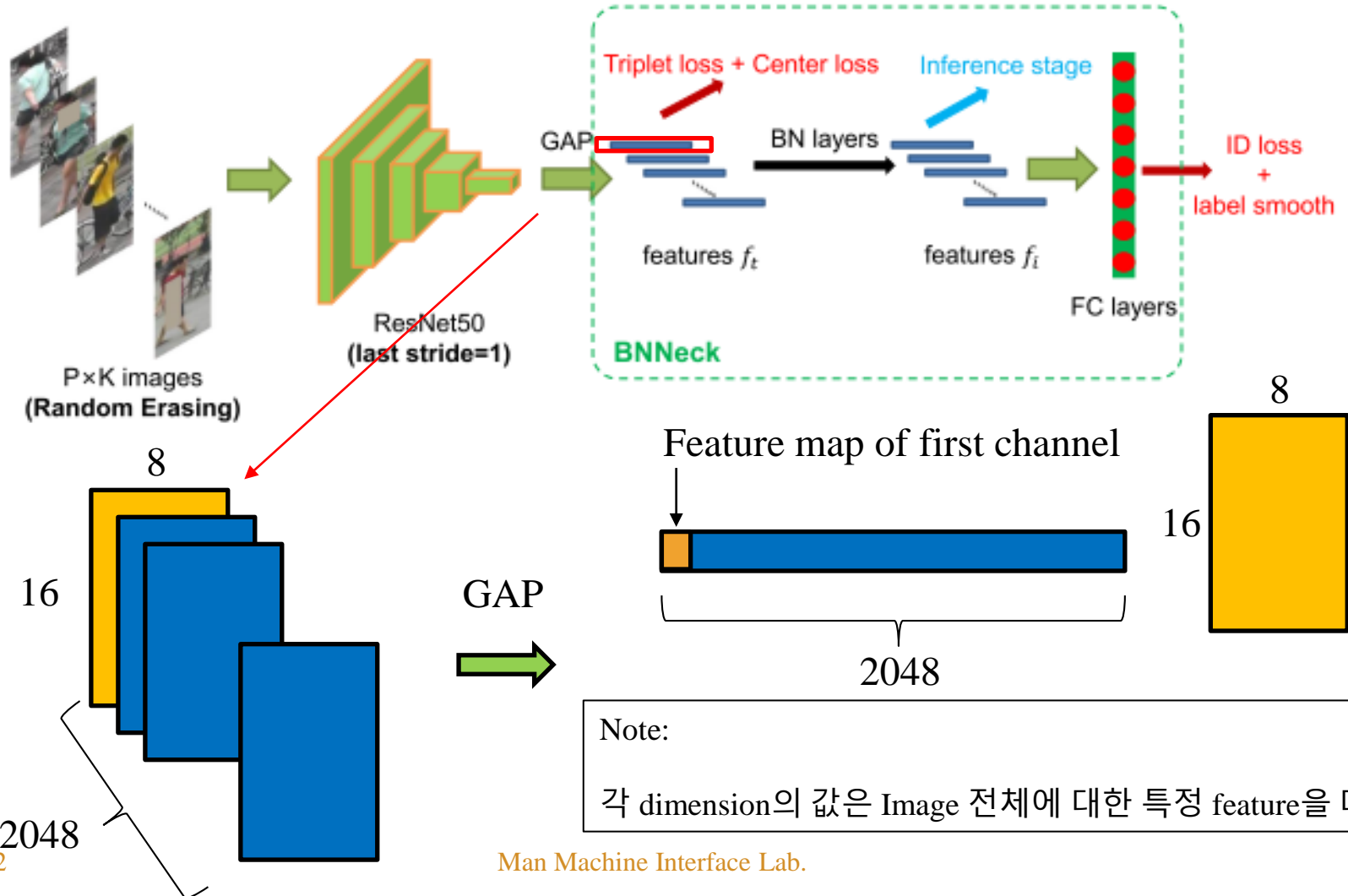
- 같은 position에 있는 A,P,N의 patch token들에 대해 patch wise(A-N,A-P)로 distance를 구한뒤, distance간 차이가 큰 일정 %만큼의 patch만을 weighted global average pooling (Elemented weighted triplet loss의 Idea와 동일)



- 추가적인 image의 feature를 사용해도 CLS token만을 사용할 때보다 성능이 오르지 않는다면, 그 이유는 무엇일까..?

CNN features

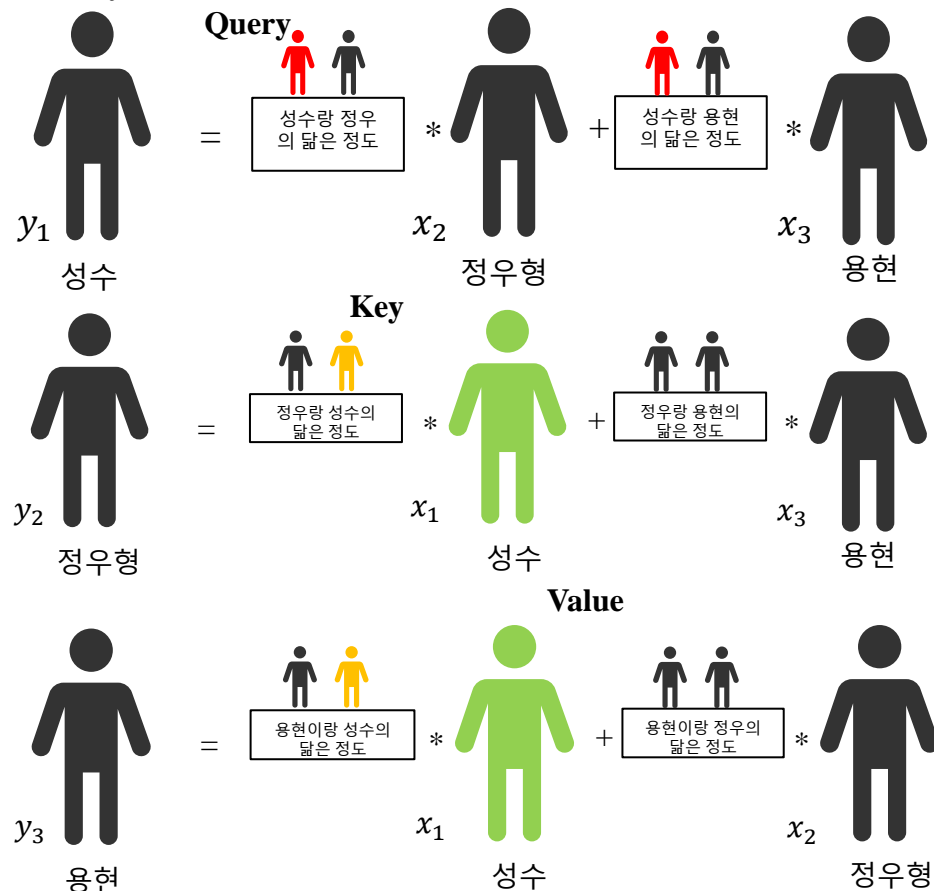
➤ CNN based Re-ID model



ViT CLS Token

➤ Based on Self-Attention

- ViT는 self-Attention mechanism을 통해 patch간의 상관 관계를 학습하는 모델
- Every input vector x_i is used in **three different ways** in the self attention operation

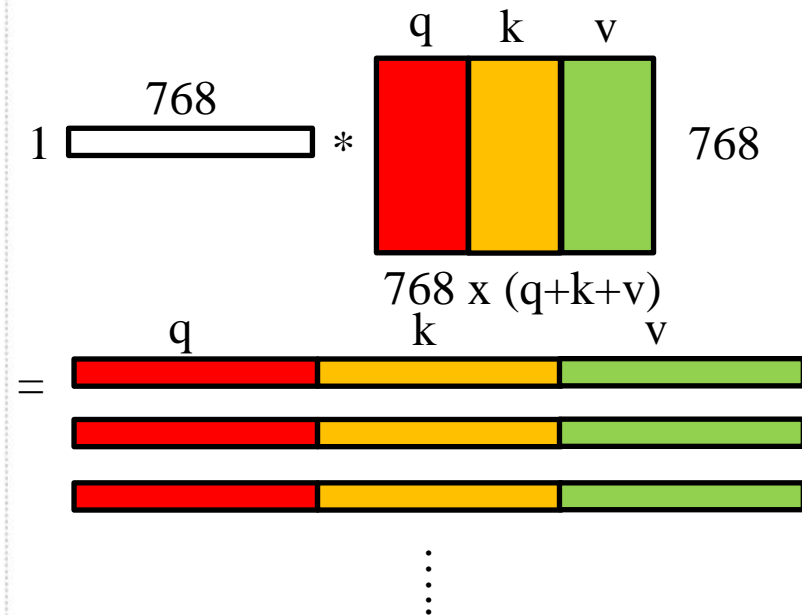
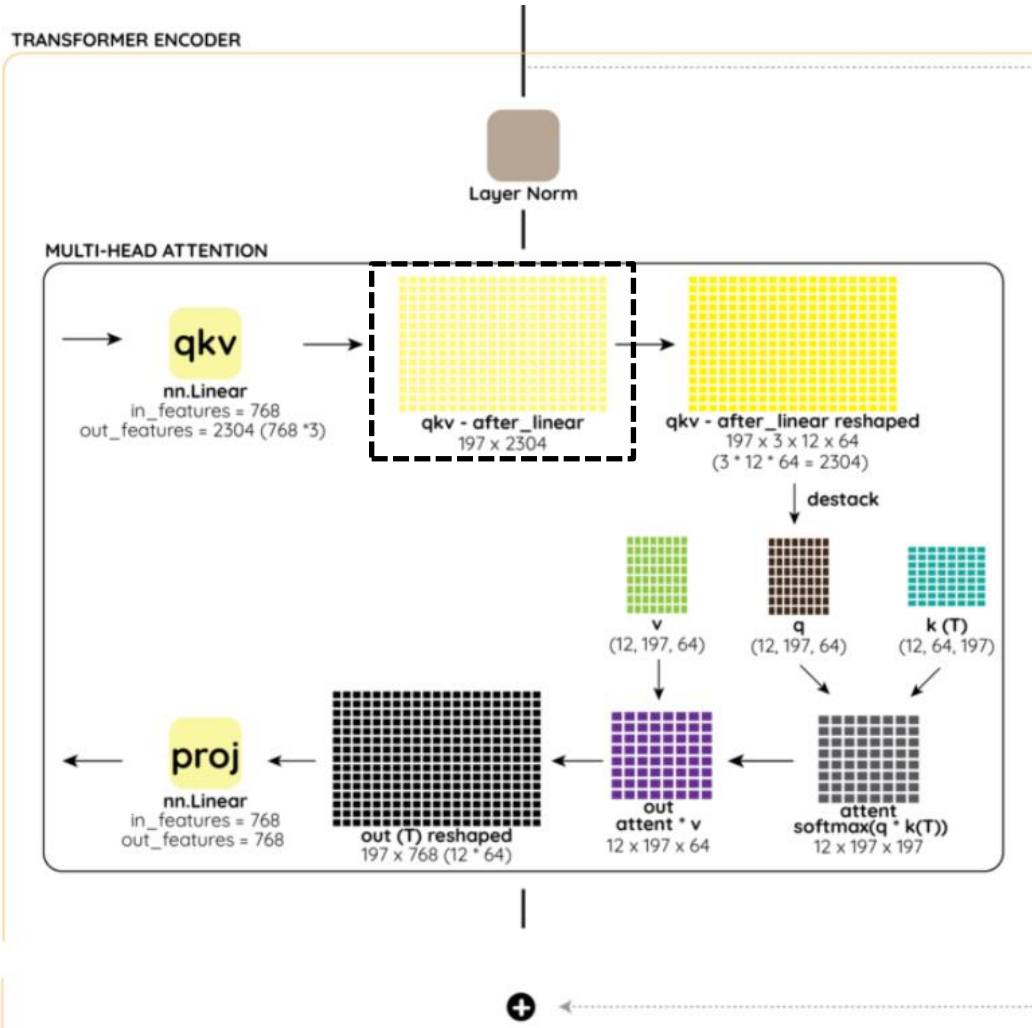


ViT CLS Token

➤ Query, Key, Value

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

* q,k,v dimension= 768

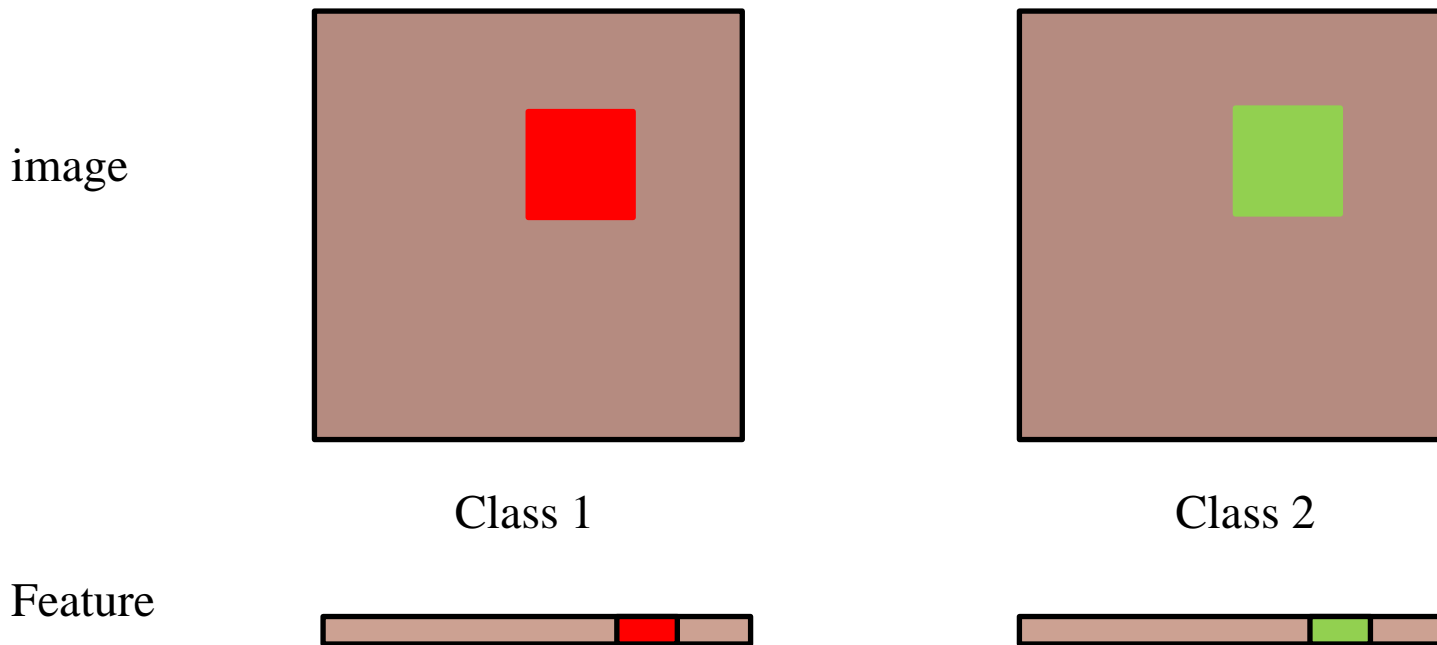


ViT CLS Token

➤ Based on Self-Attention

- ViT는 self-Attention mechanism을 통해 patch간의 **상관 관계**를 학습하는 모델
- 학습과정에서 model 은 Image의 class 를 나타내는데 도움이 되는 patch를 선택
- 선택된 Patch들의 weighted sum으로 CLS token을 생성

➤ Ex) CNN

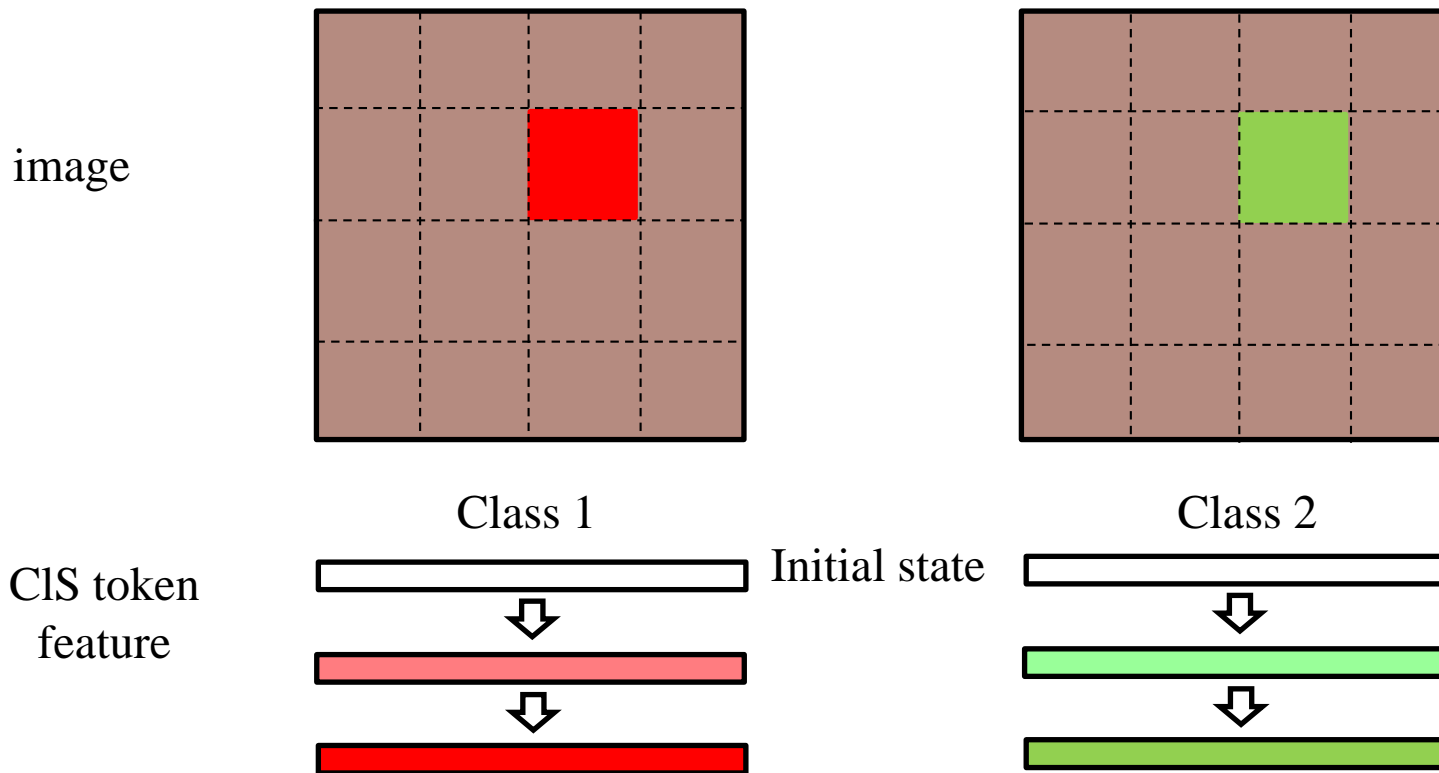


ViT CLS Token

➤ Based on Self-Attention

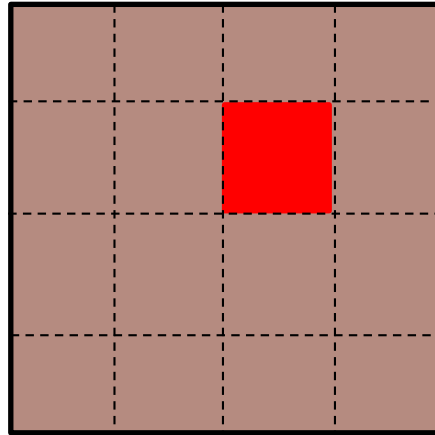
- ViT는 self-Attention mechanism을 통해 patch간의 상관 관계를 학습하는 모델
- 학습과정에서 model 은 Image의 class 를 나타내는데 도움이 되는 patch를 선택
- 선택된 Patch들의 weighted sum으로 CLS token을 생성 (Note : CLS token은 learnable parameter)

➤ Ex) ViT



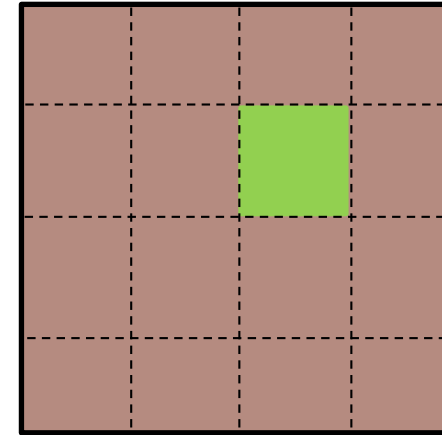
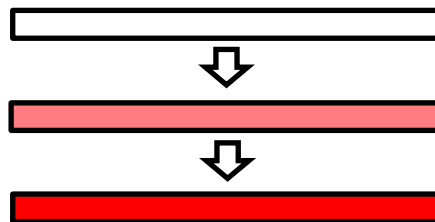
ViT CLS Token

image



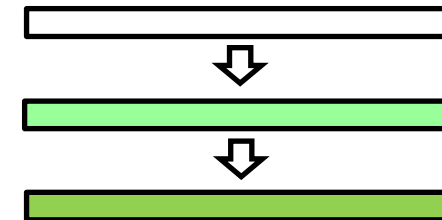
Class 1

CLS token
feature



Class 2

Initial



- 1. Element – weighted Triplet loss로 학습이 안된 이유?
 - CLS token과 상관도가 높은 patch(class를 대표할 수 있는)들의 weighted sum이 다를것이기 때문
- 2. Additional feature가 triplet 학습에 도움이 되지 않은 이유?
 - CLS token 그 자체로 이미 image의 class를 판별하기 유용한 patch feature의 집합체
 - Model 학습과정에서 선별한 patch token과 , 내가 선별한 patch들의 기준의 차이에 의해 CLS token의 학습에 방해가 되었을 것

Transformer based Re-ID

- Elemented weighted triplet loss로 학습이 잘 되지 않은 이유에 대한 고찰
 - ViT의 특성 상 CLS token에 image의 feature가 직접적으로 반영되지 않은 것은 아닌가
→ Patch들의 Weighted summation형식으로 반영되어 있음
 - 즉, CLS token은 patch의 image feature를 나타내는 embedding space와 구분된 space에 projection될 수도 있지않은가?
→ CLS token의 feature dimension은 각 patch token들이 projection되는 embedding space와 동일
 - CLS token에 Image의 fine feature가 반영될수 있는가?

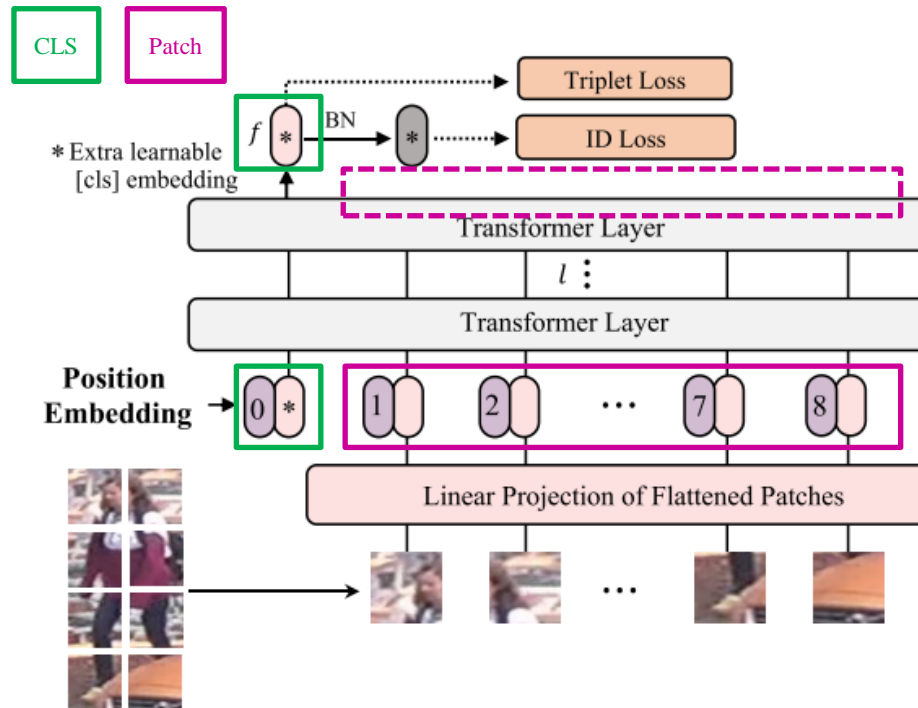


Image size $256 * 128$

Patch size $16 * 16$

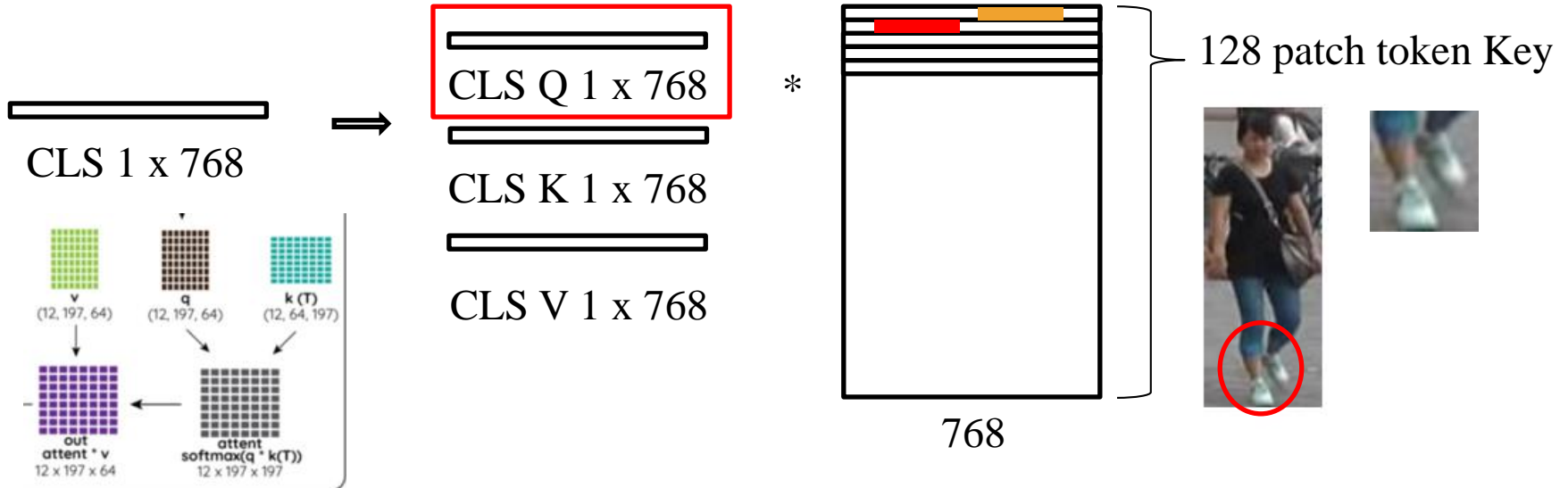
Patch Dimension = $16 * 16 * 3 = 768$

Patch $16 * 8 + \text{CLS } 1 = 129$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Weighted Triplet loss for CLS token

- 그렇다면 CLS token에 weighted triplet loss를 어떻게 적용해야할까?



- CLS query token이 각 patch token(Key)와 곱해질 때 patch의 어느 dimension에 의해 상관 관계가 dominant 하게 정해졌는가?
 - 즉, CLS token이 Patch의 어느 부분에 집중하여 class판별에 유용하다고 판단하였는지를 보자
- 1. CLS 의 Query vector와 patch 의 Key vector간 차이가 작은 부분을 dominant 한 feature로 본 후,
 - CLS Patch와의 상관관계(Q*K값)와 dominant한 feature를 이용하여 image 전체를 나타내는 feature vector를 생성 (GAP 또는 patch wise distance에 기반한 feature가 아니라)
 - Anchor sample과 Negative sample 각각에 대해 feature vector를 뽑은 후, 이를 이용하여 Triplet loss를 구하기
- 2. CLS token과의 내적값이 높은 patch의 Key vector 들을 Anchor, Negative sample에 대해 각각 얻은 후 이들 간의 distance를 이용하여 triplet loss 를 구성

Weighted Triplet loss for CLS token

➤ 정리

- “ViT를 이용해서 metric learning을 하고자 할때, CLS token만을 이용한 triplet learning이 과연 최선인가?”
- ViT의 특성을 고려하여, 학습과정에서 얻을 수 있는 재료들을 잘 이용함으로써 Fine grained feature가 필요한 Re-ID에 도움이 될 수 있는 triplet learning을 고안하는 것이 목표
 - CNN의 GAP feature와 비교했을때, CLS Token 속 정보는 그와 성질이 다름
 - CLS Token속의 함축된 정보를 풀어내어 detail들이 학습에 직접 반영될 수 있도록 유도