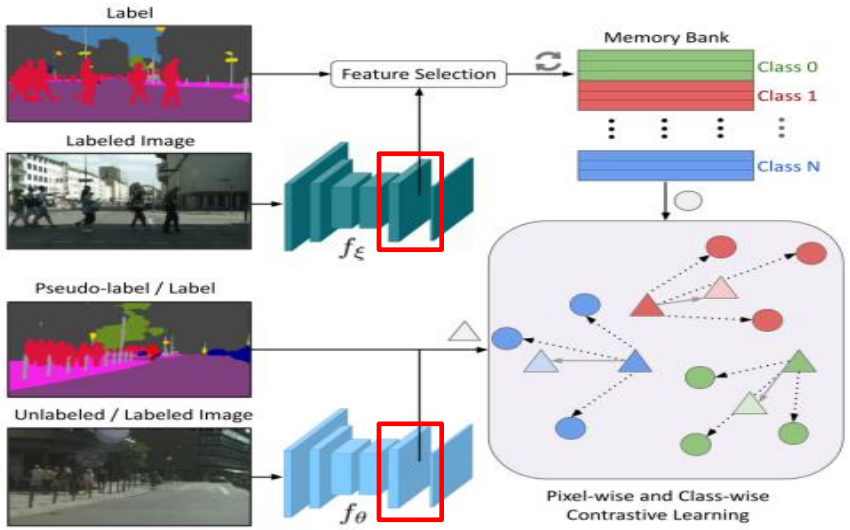
Semi-Supervised Segmentation with Memory-Bank

Semi-Supervised Semantic Segmentation with Pixel-Level Contrastive Learning from a Class-wise Memory Bank

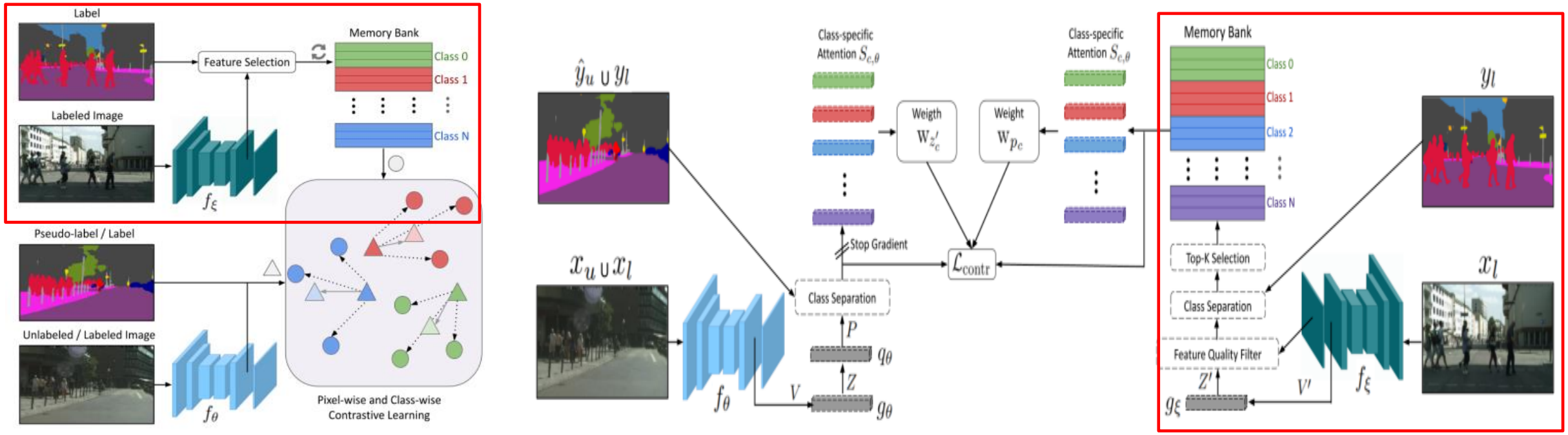
Inigo Alonso[1] Alberto Sabater[1] David Ferstl[2] Luis Montesano[1,] Ana C. Murillo[1]
[1]RoPeRT group, at DIIS - I3A, Universidad de Zaragoza, Spain
[2]Magic Leap, Zürich, Switzerland
[3]Bitbrain, Zaragoza, Spain
{inigo, asabater, montesano, acm}@unizar.es, dferstl@magicleap.com

2023.06.27

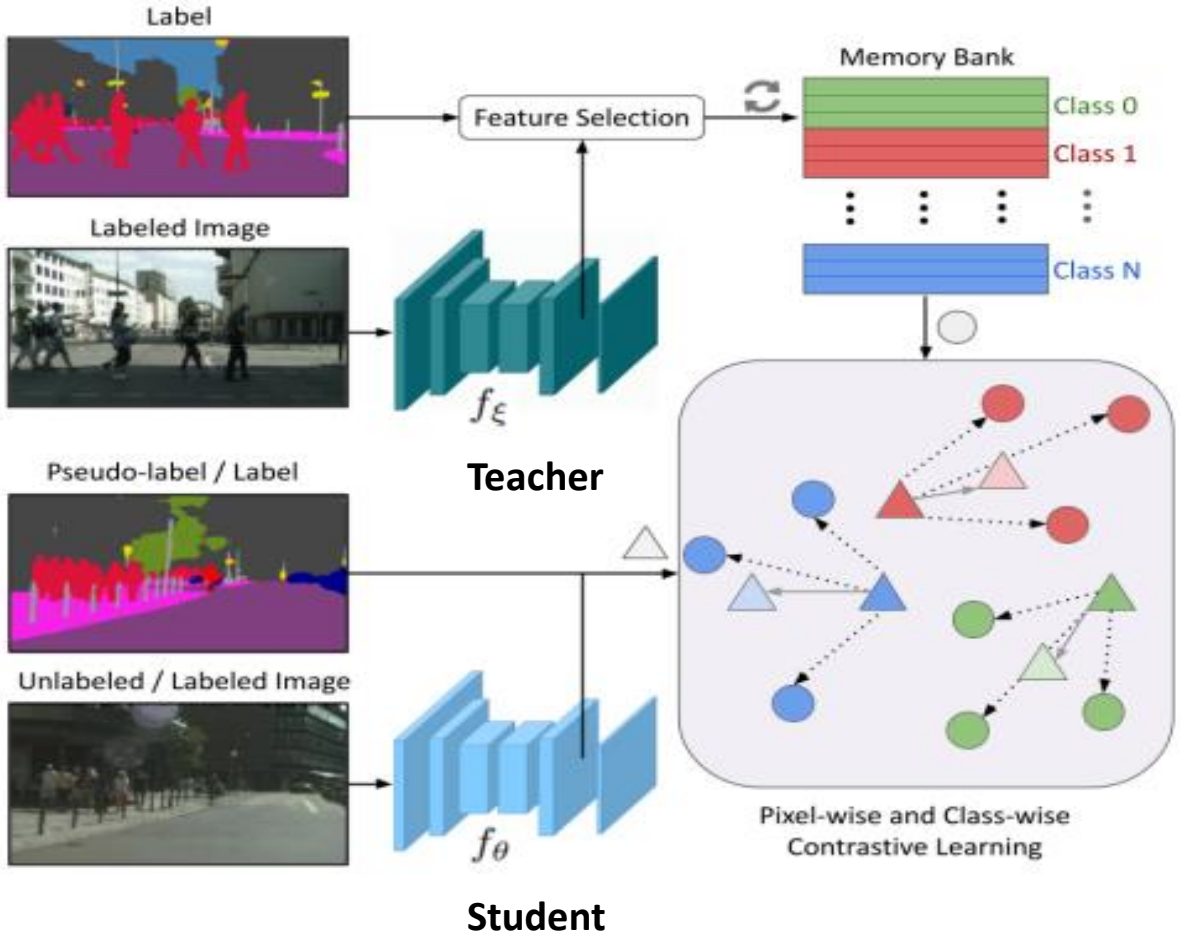


- DeepLabV3 기반
- ASPP layer를 통과한 high-feature와 low-feature를 concatenate한 후 Decoding 단계를 거친 feature를 contrastive learning/memory bank update에 사용

Memory-Bank Update



- 기존 방식은 고정된 Feature Dictionary 이용



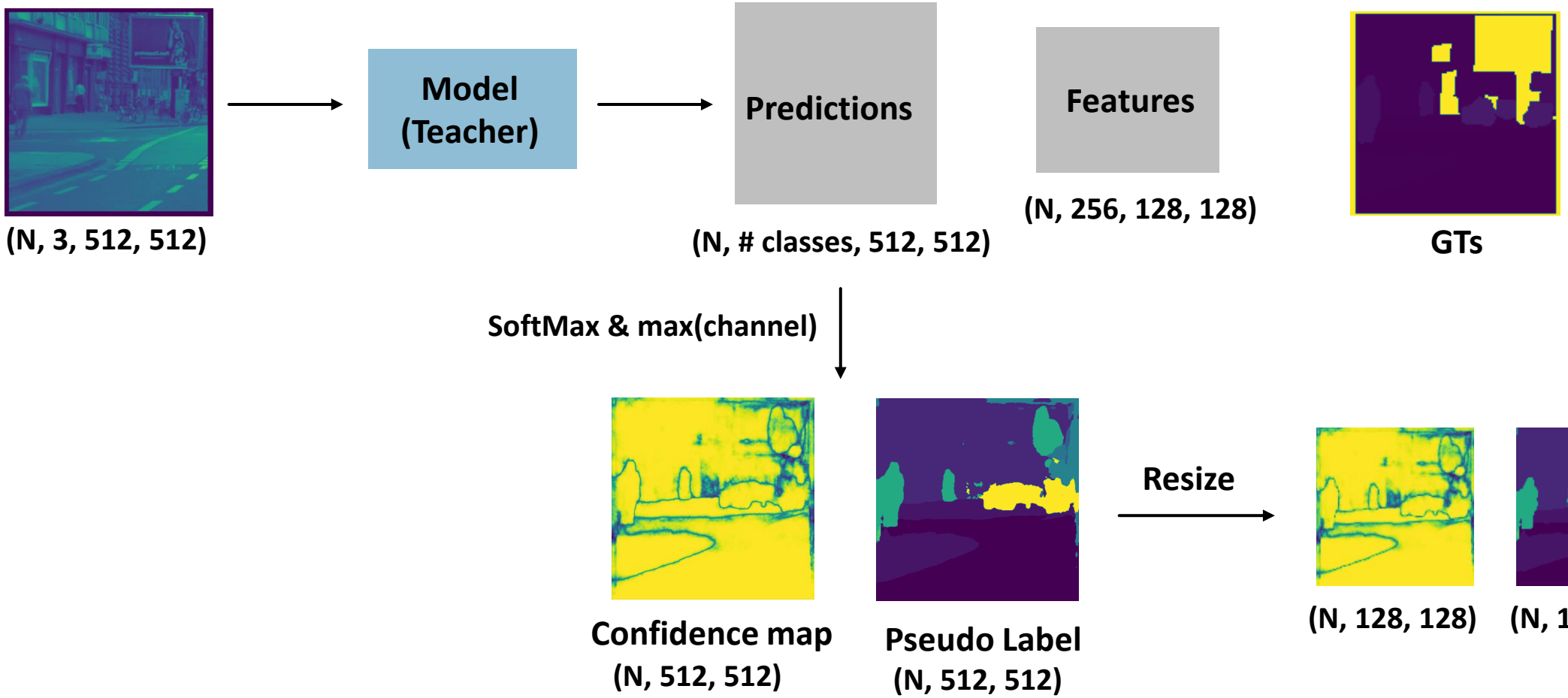
* Teacher의 파라미터는 EMA에 따라 update

$$\xi = \tau\xi + (1-\tau)\theta.$$

- Student 네트워크는 Teacher 학습에 사용된 labeled sample도 입력으로 받아 feature 출력 $(x_l \,\&x_u)$
- **Student의 feature 출력들$(x_l \,\&x_u)$ 과 Memory 간의 코사인 유사도 최대화 (class 별로)**

Step 1. Get Pseudo-Labels

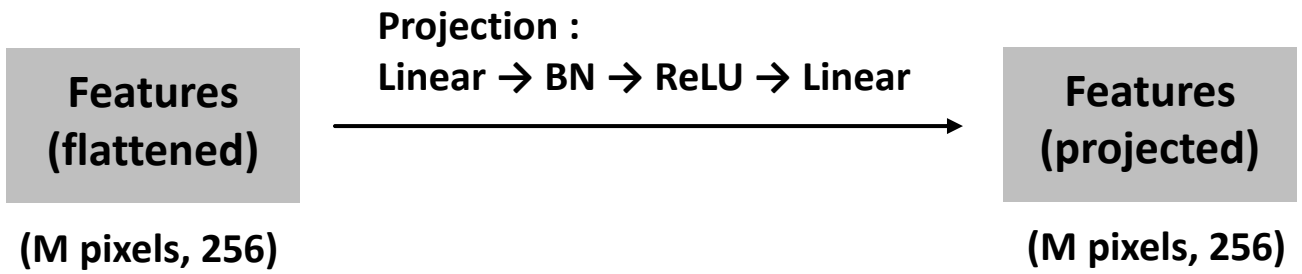
- Source image(GT O)를 Teacher 네트워크에 입력으로 넣어 Pseudo label 생성
- Teacher 네트워크는 EMA 방식으로만 update



Step 2. Feature Quality Filter

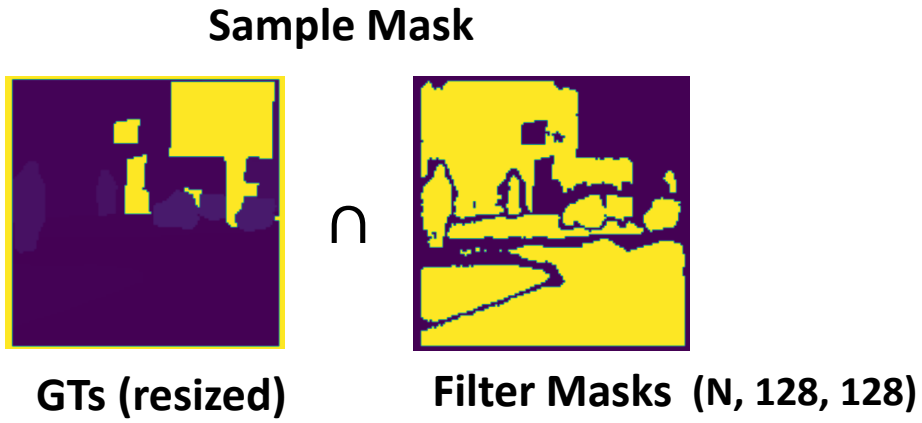
1. GT 기준으로 정확하게 예측되었으며, confidence score가 0.95 초과인 pixel들에 해당하는 feature만 추출

2. 추출된 feature를 projection



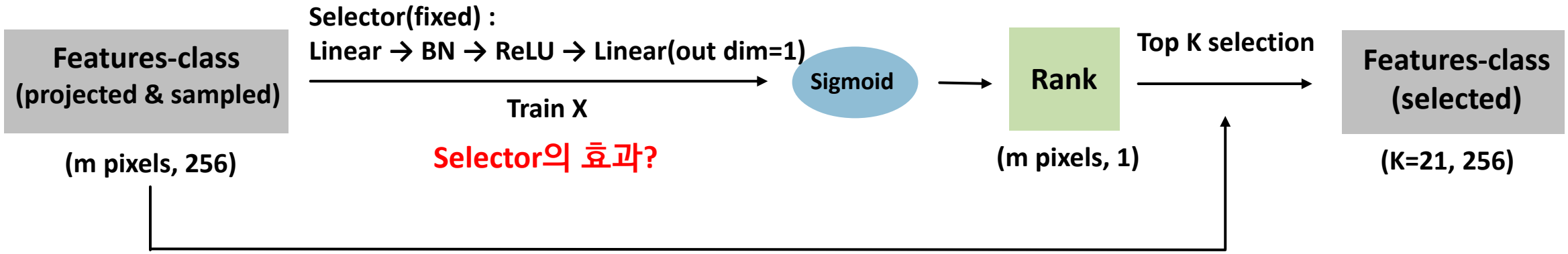
Step 3. Update Memory-Bank

1. GT mask에서 각 class별로 Step2-1 과정에 해당하는 영역들만 고려



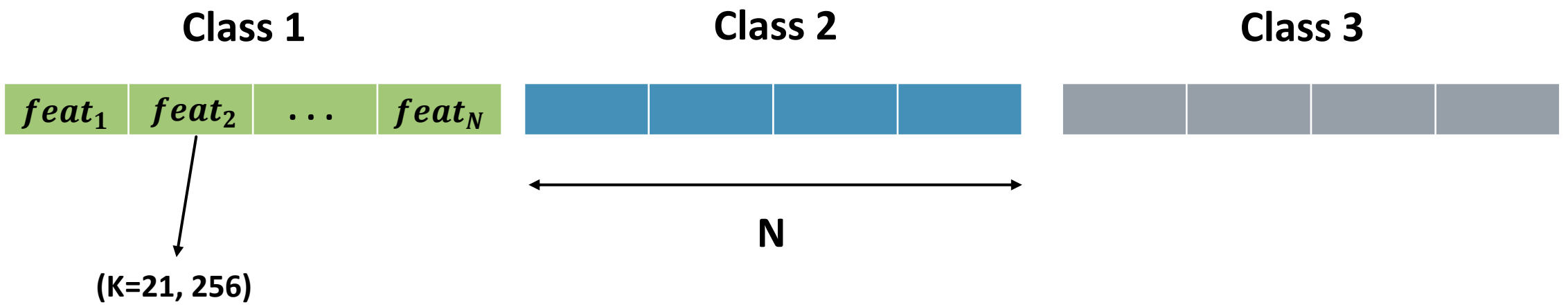
2. 각 class별로 Sample Mask에 해당하는 영역의 features(projected)만 추출 – assume m pixels

3. 2에서 추출된 feature를 활용하여 Rank 계산

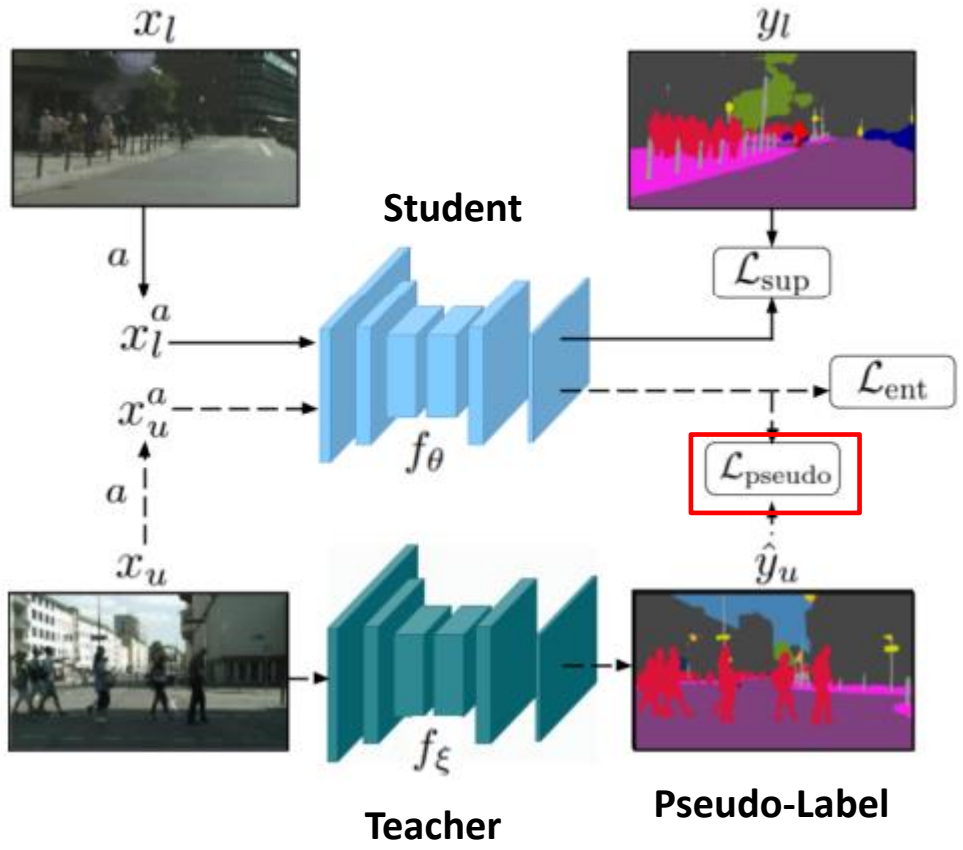


4. 3에서 최종적으로 선별된 class feature를 Memory에 저장 (선입선출 방식으로 유지)

- Feature Memory는 각 class 별로 N개의 선별된 feature들 저장
- N은 이미지 데이터 개수 및 batch size에 비례



Learning From Pseudo-Label



- Teacher 네트워크는 Student 학습 시 기본적으로 fix됨
- **단, 매 epoch마다 EMA를 통해 update됨**
- 이에 따라 생성되는 pseudo label 역시 매 epoch마다 다름

$$\xi = \tau\xi + (1-\tau)\theta.$$

EMA (gradient descent X)

- **기존 방식은 pseudo label이 고정된 상태**

Step 1. Get Pseudo-Labels

- GT가 없는 target 이미지를 Teacher 네트워크에 입력으로 넣어 pseudo label 생성
- **Pseudo label은 매 iteration마다 EMA를 통해 update되는 Teacher네트워크를 통해 생성됨 (not fixed)**

(N, 3, 512, 512)

Model (Teacher)

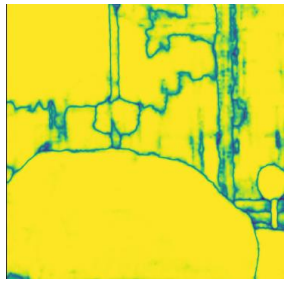
Predictions

(N, # classes, 512, 512)

Features

(N, 256, 128, 128)

SoftMax & max(channel)

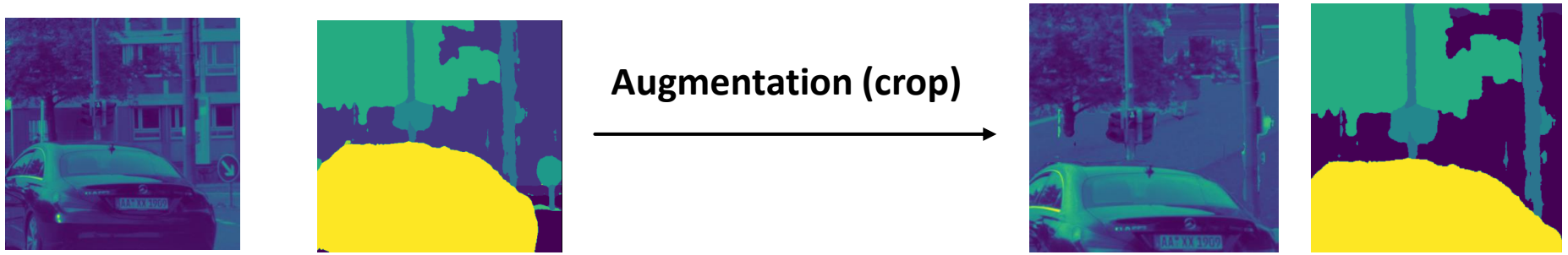


Confidence map
(N, 512, 512)

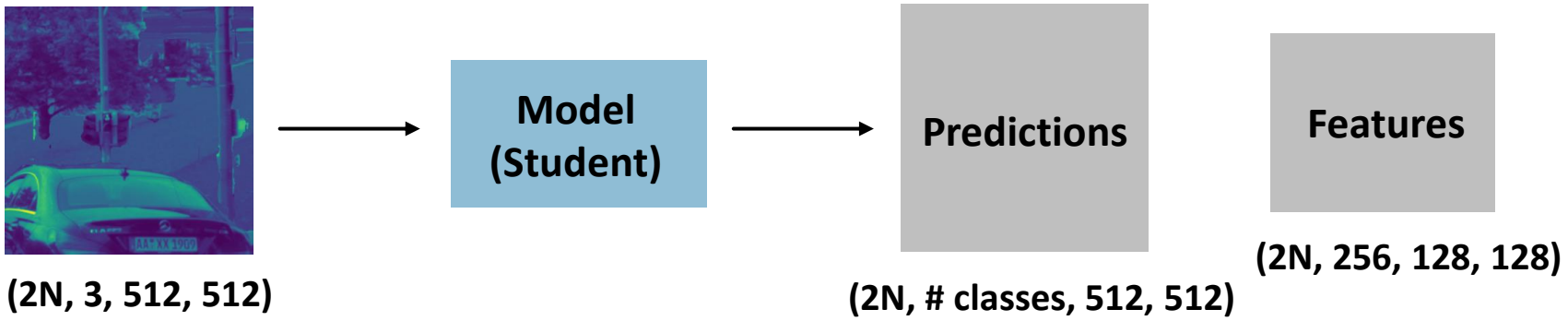
Pseudo Label
(N, 512, 512)

Step 2. Consistency Regularization

1. Step1에의 unlabeled images와 생성된 pseudo label들에 대한 augmentation 진행

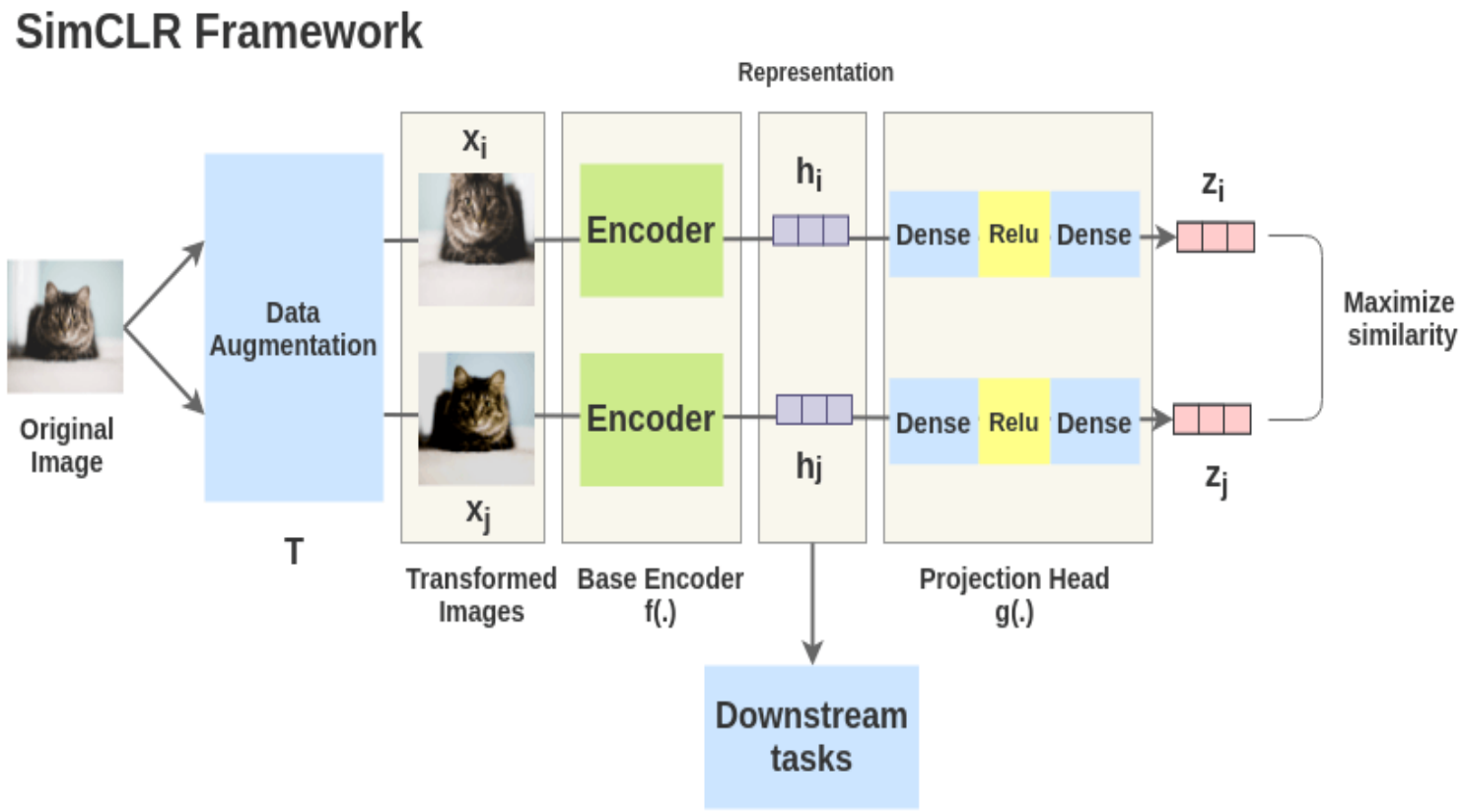


2. 1의 augmented unlabeled images를 student 모델에 입력으로 넣어 **pseudo label prediction** 출력



3. **Step1의 pseudo label과 Step2의 pseudo label prediction과의 Cross-Entropy 계산**

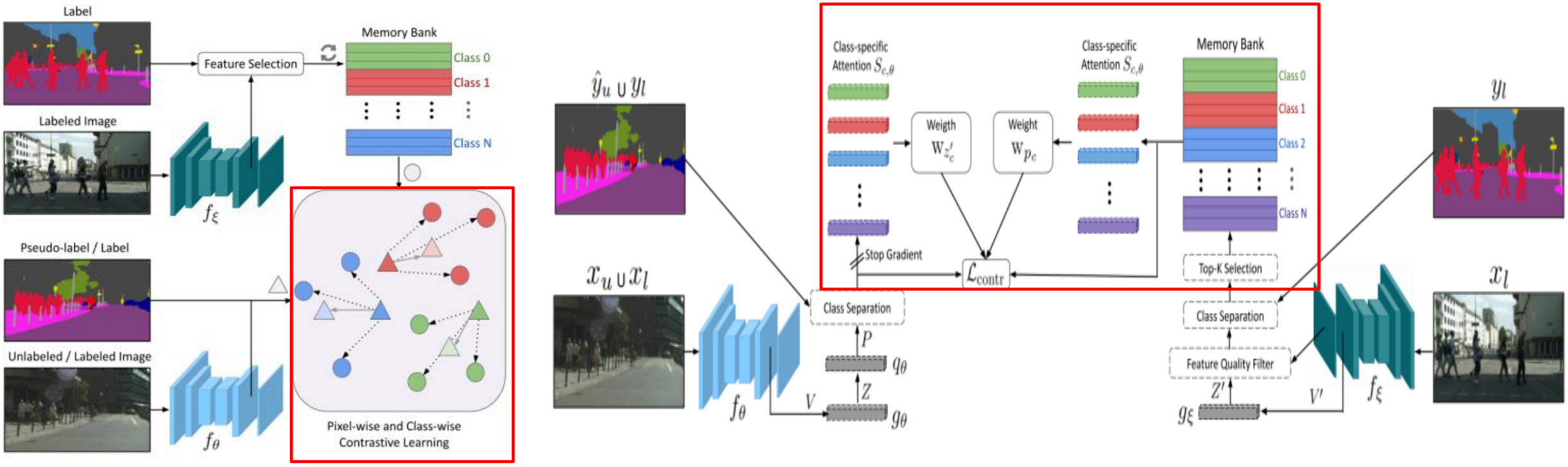
Consistency Regularization in Self-Supervised Learning



- Dictionary를 사용하진 않지만 feature matching을 활용하고 있음
- **동일한 이미지에 대해 서로 다른 augmentation**을 적용 후, feature space에서 코사인 유사도 최대화

Matching Loss

Contrastive Learning from a Class-wise Memory Bank



Unlabeled images의 feature를 Memory-Bank에 저장된 labeled images의 feature들에 align

- 현재 방식은 평균 벡터간의 코사인 유사도에 기반하고 있음

Step 1. Get Pseudo-Labels (Augmented & Resized)

1. Learning from Pseudo-Label의 Step2.1에서 생성된 augmented pseudo label을 feature size에 맞게 resize

(2N, 128, 128)

Augmented & resized pseudo labels

2. Learning from Pseudo-Label의 Step2.2에서 생성된 feature에서

Step 1. Get Pseudo-Labels (from augmented unlabeled images)

1. From Step2.2 of Learning from Pseudo Label (feature size에 맞게 resize)

Unlabeled (augmented)

(2N, 3, 512, 512)

→ **Model (Student)** →

Predictions

(2N, # classes, 512, 512)

Features

(2N, 256, 128, 128)

Pseudo Label (augmented)

(2N, 3, 512, 512)

2. 1의 pseudo label(resized)에 해당하는 영역의 feature만 추출 – assume K pixels
3. 2에서 추출된 feature들을 projection (Linear → BN → ReLU → Linear)

Features

(N, 256, 128, 128)

(2N, 3, 128, 128)

→

Features (selected)

(K pixels, 256)

Projection →

Features (projected)

(K pixels, 256)

Step 2. Contrastive Learning with Memory-Bank

1. 특정 class에 속하는 pseudo label 영역의 픽셀들에 해당하는 feature만 추출 (class-wise split)

2. Memory-Bank에서도 특정 class에 해당하는 저장된 feature 추출 (N, 256) Z'_c * N=Top K selection에 의해 선별된 pixel들

3. 1, 2의 feature들을 normalize한 후 **M x N similarity matrix**계산 $\mathcal{C}(p_c, z'_c) = \frac{\langle p_c, z'_c \rangle}{\|p_c\|_2 \cdot \|z'_c\|_2},$

4. 3의 결과로부터 [0, 2]의 값을 가지는 Distance Matrix 계산 $\mathcal{D}(p_c, z'_c) = \mathrm{w}_{p_c}\mathrm{w}_{z'_c}(1 - \mathcal{C}(p_c, z'_c)),$

   Rescale weights

   - Distance가 0에 가까울수록 유사도가 높은 벡터들

* Memory의 feature는 선별된 소수의 픽셀들에 대한 것이기 때문에 pair-wise하게 계산됨에도 memory issue X

Step 3. Get per-feature weights for Contrastive-Learning

- Step2의 Contrastive Learning에 도움이 되는 feature 벡터들에 가중치를 부여하여 weighted Distance 계산

1. Step2의 1, 2에서 추출된 feature들을 selector에 의해 projection (Linear → BN → ReLU → Linear)

P_c **(M pixels, 256)** —**Selector & sigmoid**→ W_{P_c} **(M pixels, 1)**

Z'_c **(N, 256)** —**Selector & sigmoid**→ $W_{Z'_C}$ **(N, 1)**

*** Selector & sigmoid : Class-specific attention module**

2. 1에서 계산된 weigh들을 Step2에서 계산된 Distance에 곱함