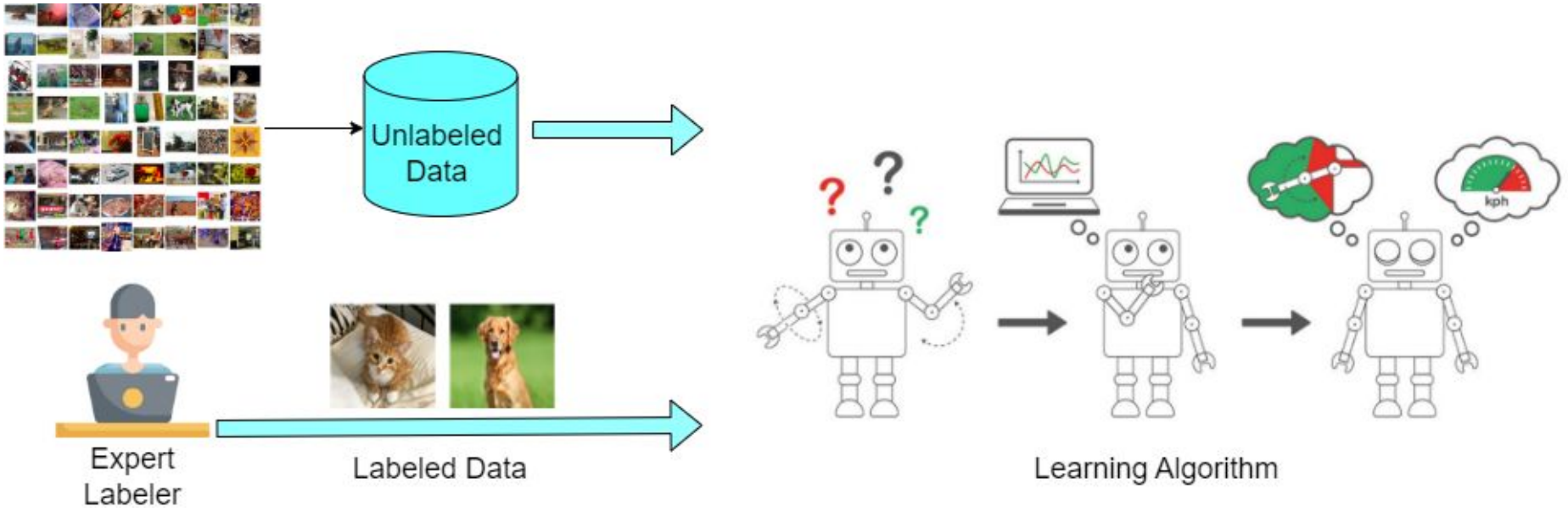
Semi-Supervised Learning

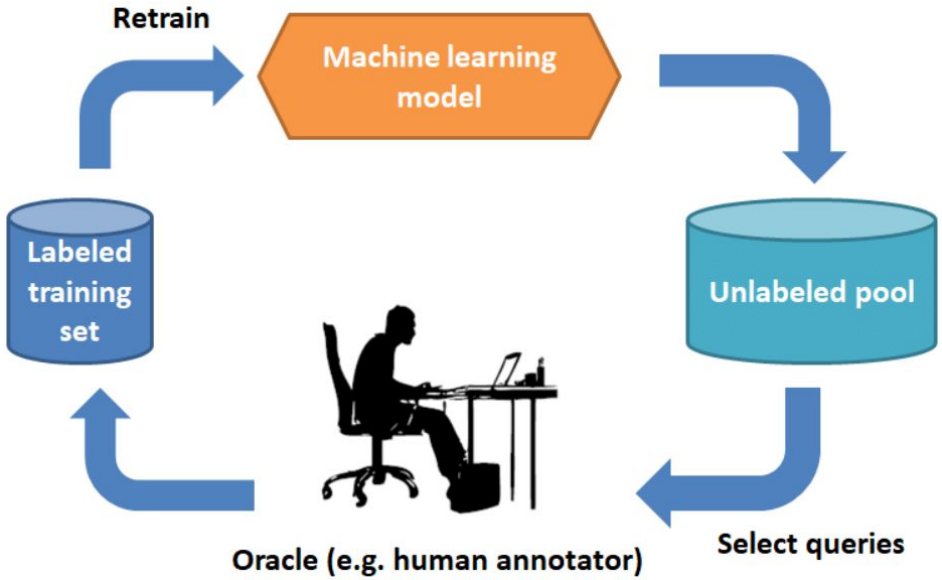


1 - What's Semi-Supervised Learning

Giải pháp khi giải quyết bài toán bị giới hạn labeled data (ít data, label-unlabel data,...)

- **Pre-training + fine-tuning**
- **Semi-supervised learning**
- **Pre-training + dataset auto-generation**
- **Active learning**

Active Learning =>



1 - What's Semi-Supervised Learning

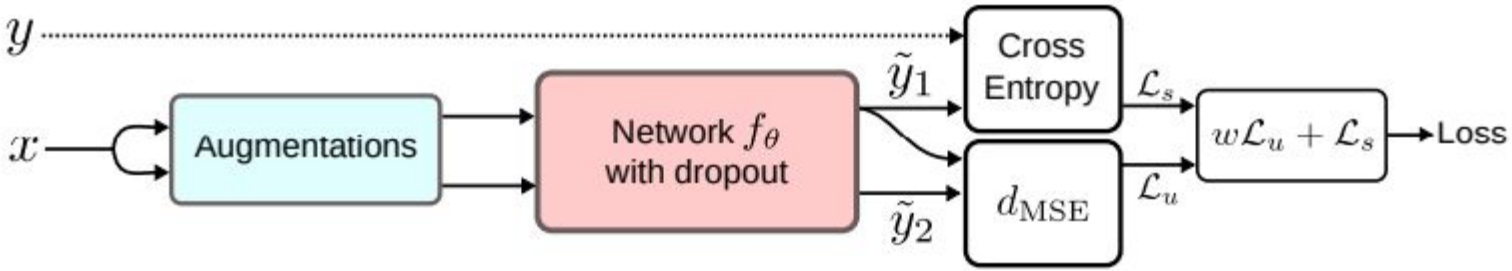
Semi-Supervised Learning cho phép học trên 2 loại dữ liệu labeled và unlabeled

Một số giả thiết được đặt ra trong bài toán semi-supervised learning

- **Smoothness**: Nếu 2 data gần với nhau trong vùng **high-density** của feature space, thì nhãn của chúng rất gần với nhau
- **Cluster**: Feature space bao gồm cả vùng dense và sparse. Với vùng Dense sẽ nhóm các điểm dữ liệu về cùng một cụm => Các điểm dữ liệu trong một cụm được kỳ vọng sẽ cùng nhãn.
- **Low-Density Separation**: Đường phân tách giữa các lớp (boundary) thường nằm ở vùng sparse nếu không boundary sẽ tách vùng Dense thành 2 cụm nhỏ ứng với 2 lớp.

2 - Consistency Regularization

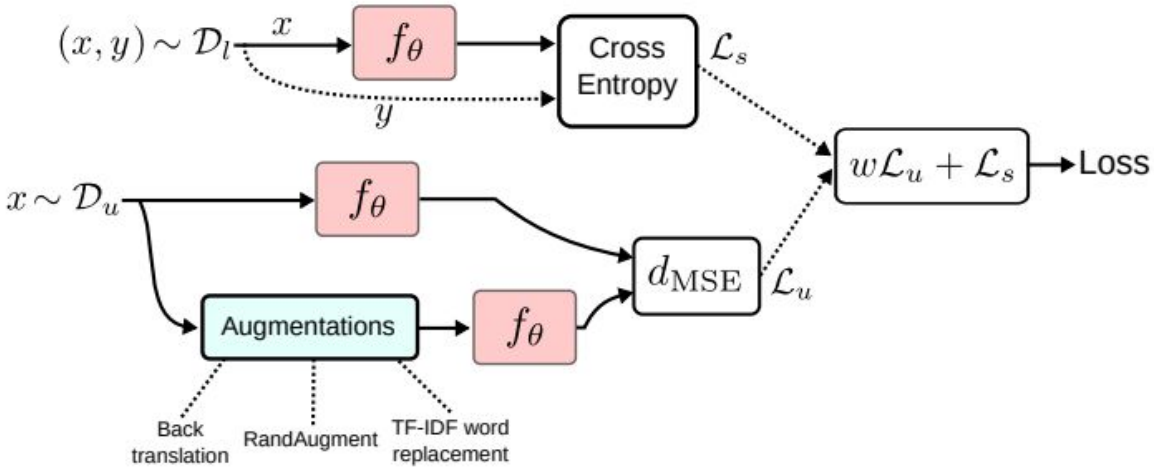
Pi model



MSE giữa 2 output được tính toán cho unsupervised loss

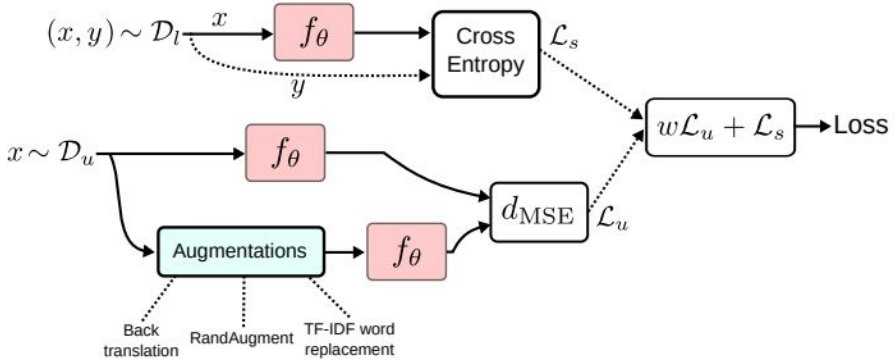
2 - Consistency Regularization

Unsupervised Data Augmentation



2 - Consistency Regularization

Unsupervised Data Augmentation

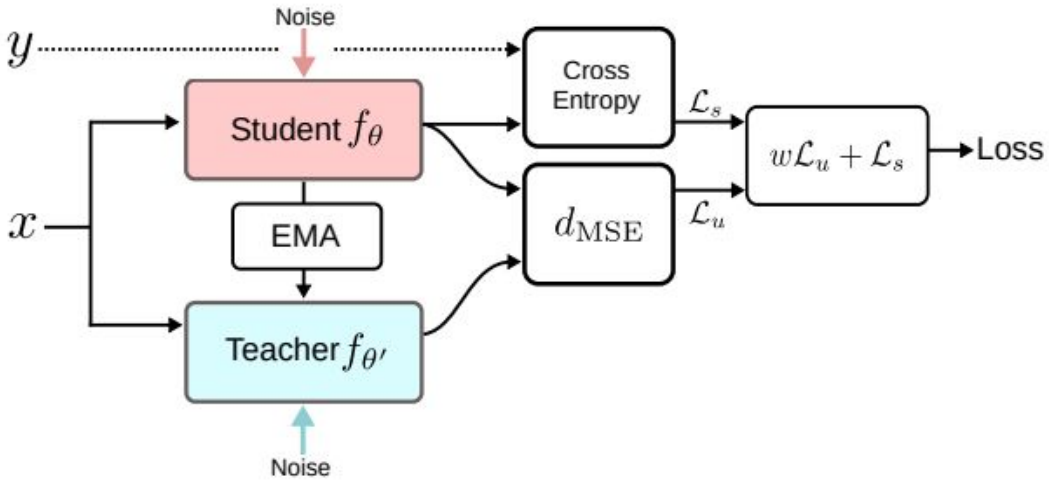



```
train_transform_ps_strong = transforms.Compose([
    transforms.Resize((config['image_res'], config['image_res']), interpolation=InterpolationMode.BICUBIC),
    transforms.RandomHorizontalFlip(),
    RandomAugment(2, 7, isPIL=True, augs=['Identity', 'AutoContrast', 'Equalize', 'Brightness', 'Sharpness',
                                          'ShearX', 'ShearY', 'TranslateX', 'TranslateY', 'Rotate']),

    transforms.ToTensor(),
    normalize,
    RandomErasing(probability=config['erasing_p'], mean=[0.0, 0.0, 0.0])
])
```

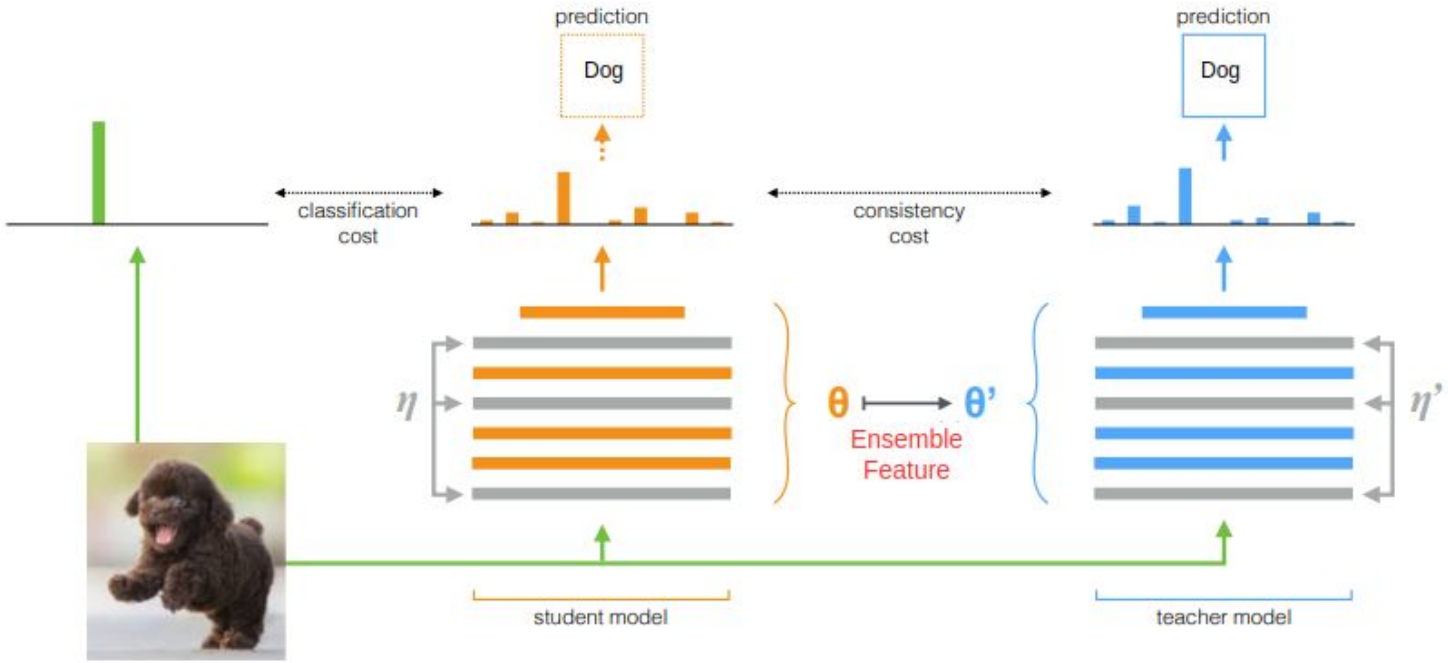
2 - Consistency Regularization

Mean Teacher



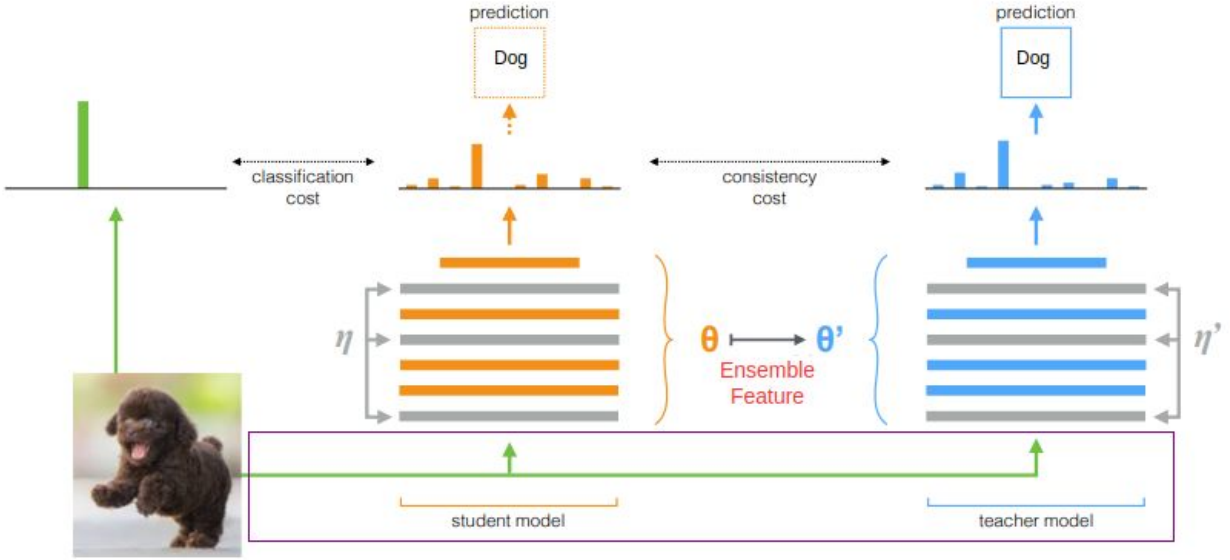
2 - Consistency Regularization

Mean Teacher



2 - Consistency Regularization

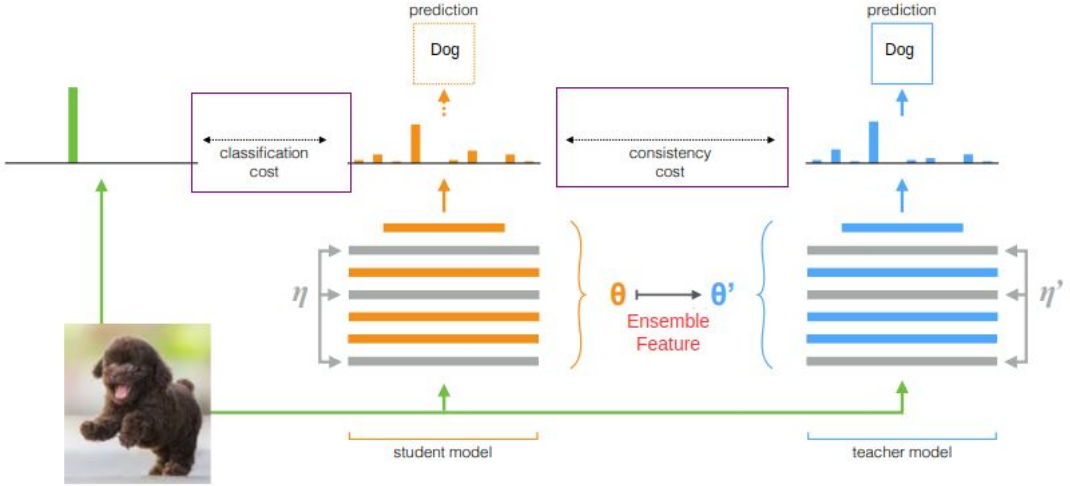
Mean Teacher




```
# forward pass
student_pred = student(images)
teacher_pred= teacher(images)
```

2 - Consistency Regularization

Mean Teacher



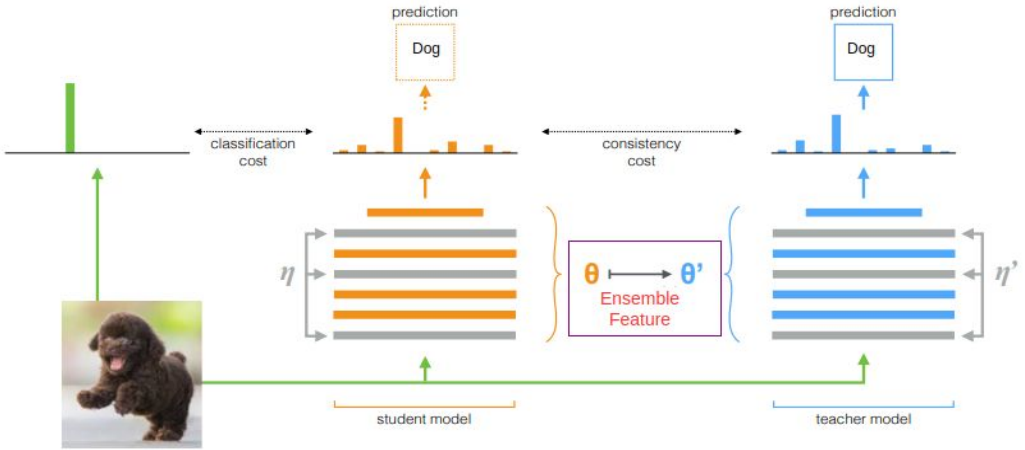

```
student_class, student_consistency = student_pred, student_pred

student_class_loss = class_criterion(student_class, labels) # CrossEntropy
consistency_loss = consistency_criterion(student_consistency, teacher_pred) # MSE

loss = student_class_loss + consistency_loss
```

2 - Consistency Regularization

Mean Teacher



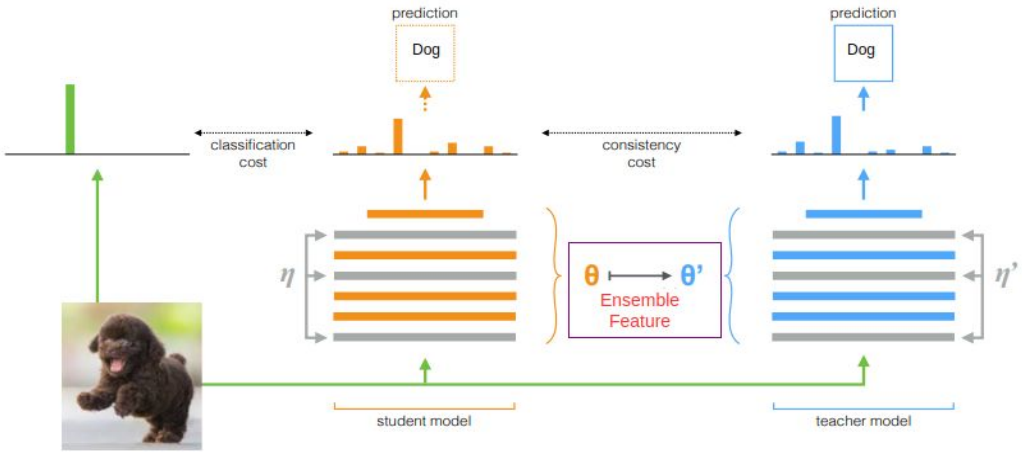
$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$$

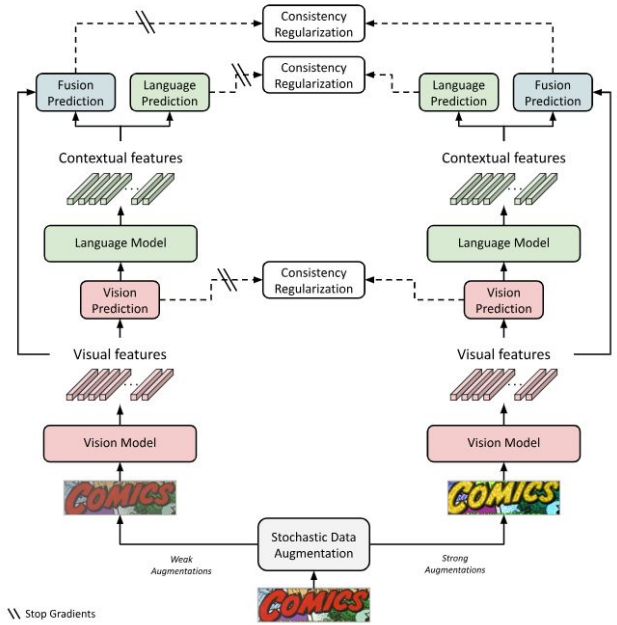
```
def update_teacher_params(student, teacher, alpha, global_step):
    # Use the true average until the exponential average is more correct
    alpha = min(1 - 1 / (global_step + 1), alpha)
    for ema_param, param in zip(teacher.parameters(), student.parameters()):
        ema_param.data.mul_(alpha).add_(1 - alpha, param.data)
```

```
# backward
optimizer.zero_grad()
loss.backward()
optimizer.step()
global_step += 1
update_teacher_params(student, teacher, 0.995, global_step)
```

2 - Consistency Regularization

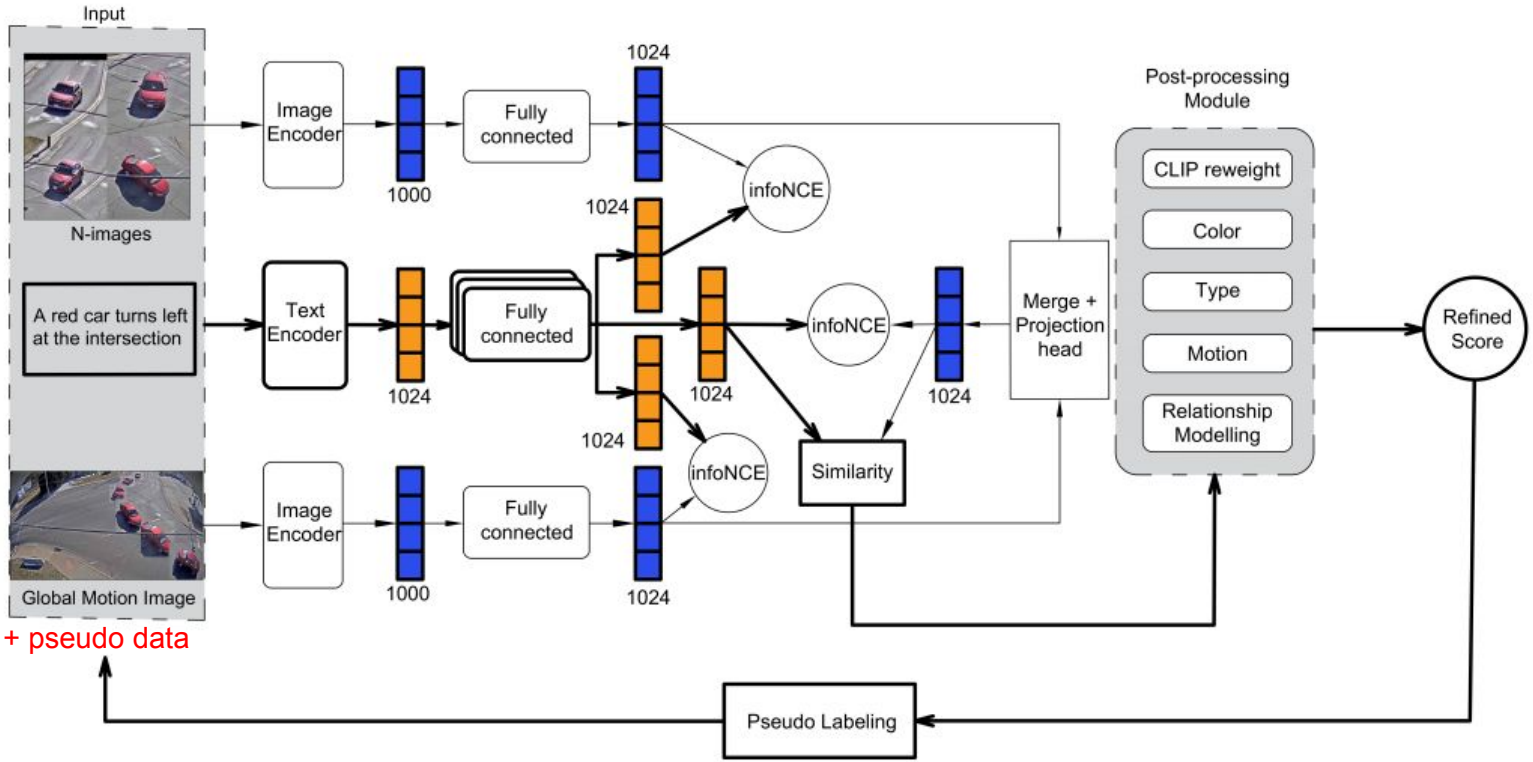
Mean Teacher





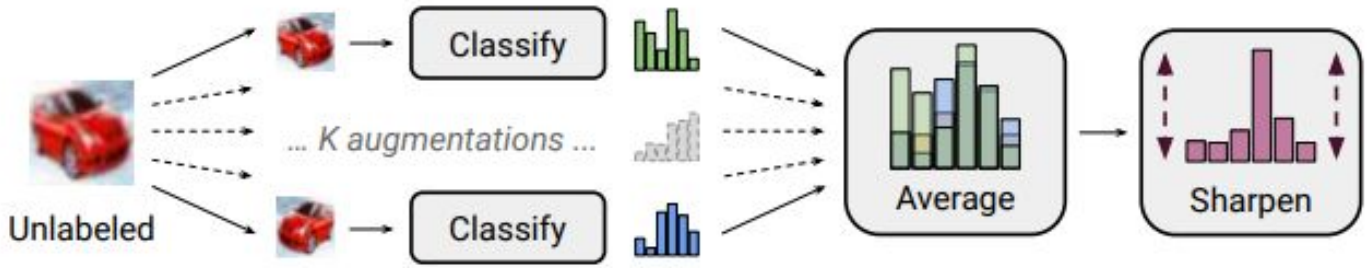
Pipeline Text Recognition

3 - Pseudo Labeling



3 - Pseudo Labeling

Pseudo Labeling with Consistency Regularization - MixMatch



Data Augmentation được áp dụng vào unlabeled data với K lần. Mỗi lần augment đều được fed qua backbone. Sau đó tính trung bình trên K predictions.

3 - Pseudo Labeling

Pseudo Labeling with Consistency Regularization - MixMatch

Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (resp. unlabeled with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \text{Augment}(x_b)$ // *Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \text{Augment}(u_b)$ // *Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \text{p}_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$ // *Compute average predictions across all augmentations of* u_b
  $q_b = \text{Sharpen}(\bar{q}_b, T)$ // *Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ // *Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ // *Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ // *Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ // *Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ // *Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$