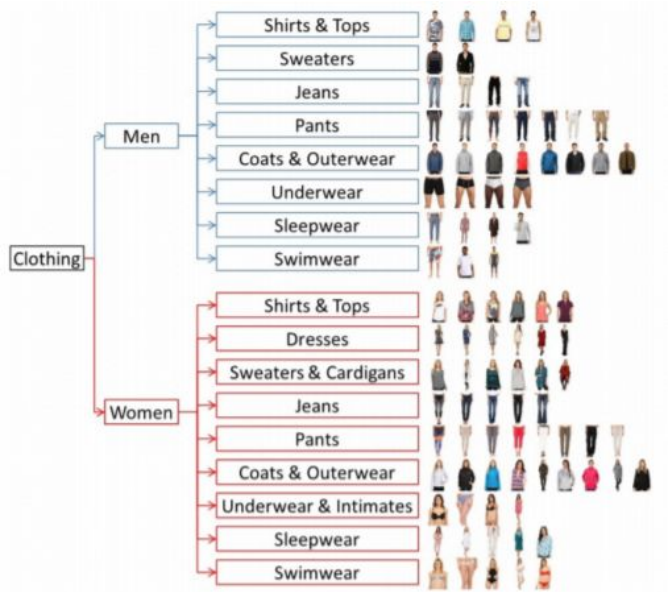
CV project with Naver 200K dataset

TEAM 13 : PIRL

Lee Jaewon Lee Jaejin Sung Jinyoung

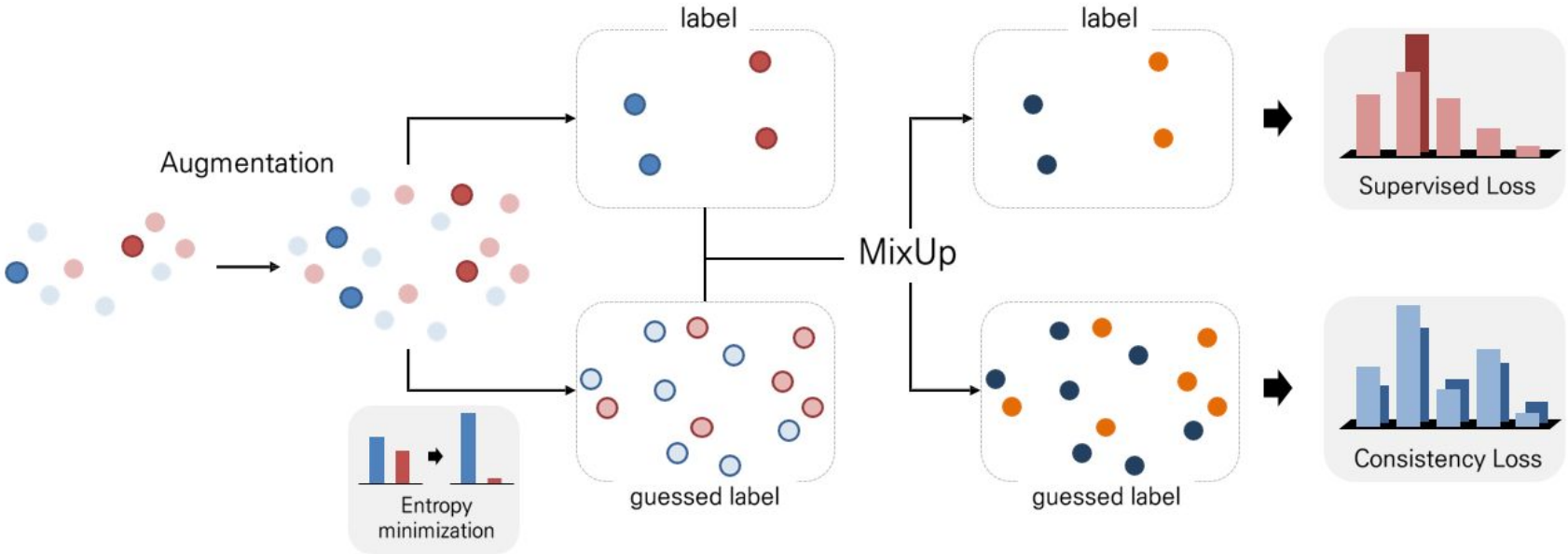
Introduction - Dataset

- KAIST-NAVER product 200K Class 265



Introduction - Base model

- Entropy Minimization + Consistency Regularization + Mix Up
  = MixMatch



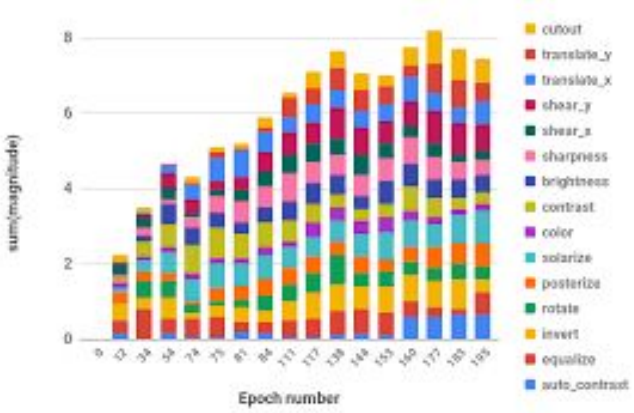
How to maximize “MixMatch”’s performance?

Our Methods

1. RandAugmentation
2. Pseudo-labeling
3. Reinforced Mix-up
4. Consistency Regularization

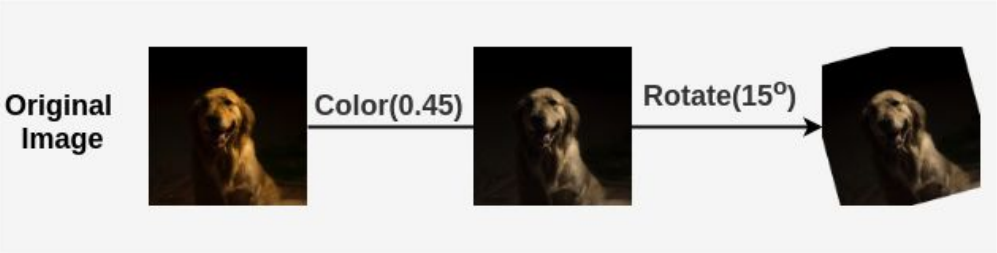
1. RandAugmentation

- Previous augmentation techniques use learned augmentation policies which has big parameter space.
- RandAugment use uniform probability between techniques and set of magnitude parameters.
- But training magnitude parameters follows a similar schedule during training and postulate that a single global distortion M may suffice for parameterizing all transformations.



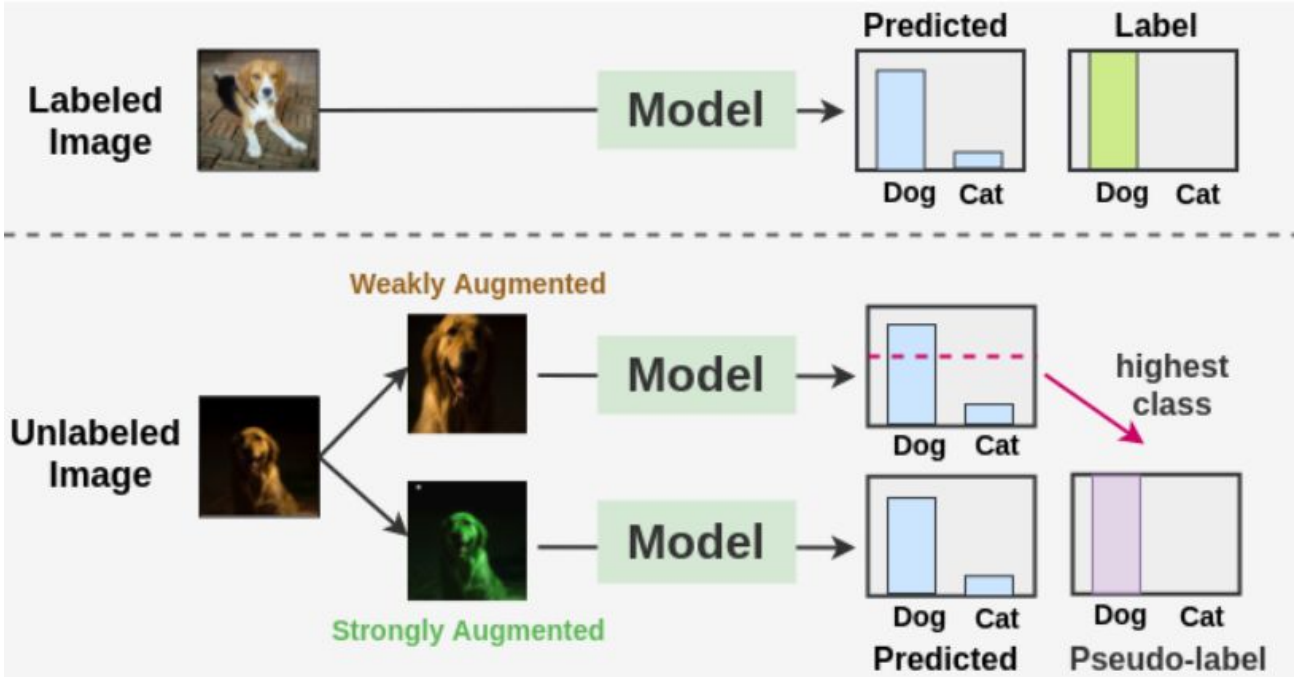
1. RandAugmentation

- RandAugmentMC(2,10)
- Two random augmentations
- Maximum strength of 10 was given



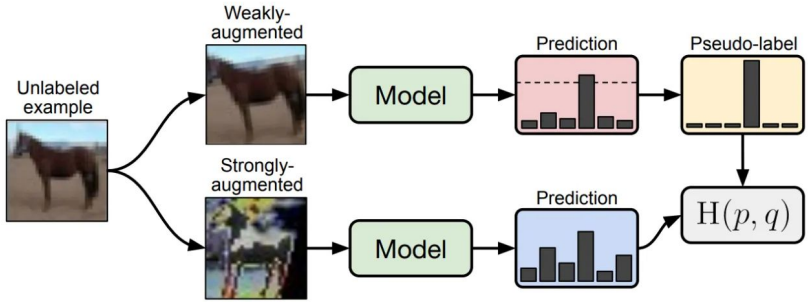

```python
class RandAugmentMC(object):
    def __init__(self, n, m):
        assert n >= 1
        assert 1 <= m <= 10
        self.n = n
        self.m = m
        self.augment_pool = fixmatch_augment_pool()

    def __call__(self, img):
        ops = random.choices(self.augment_pool, k=self.n)
        for op, max_v, bias in ops:
            v = np.random.randint(1, self.m)
            if random.random() < 0.5:
                img = op(img, v=v, max_v=max_v, bias=bias)
        img = CutoutAbs(img, 16)
        return img
```

1. Data Augmentation Pipeline



2. Pseudo-Labeling

- Instead of sharpening
- Make Pseudo label of weakly-augmented one
- Use this pseudo label as a label for strong augmented data.



3. Reinforced Mix Up

$$\begin{aligned} \hat{x} &= \lambda x_i + (1-\lambda)x_j, \\ \hat{y} &= \lambda y_i + (1-\lambda)y_j, \end{aligned}$$

where $\lambda \in [0, 1]$ is a random number

Image

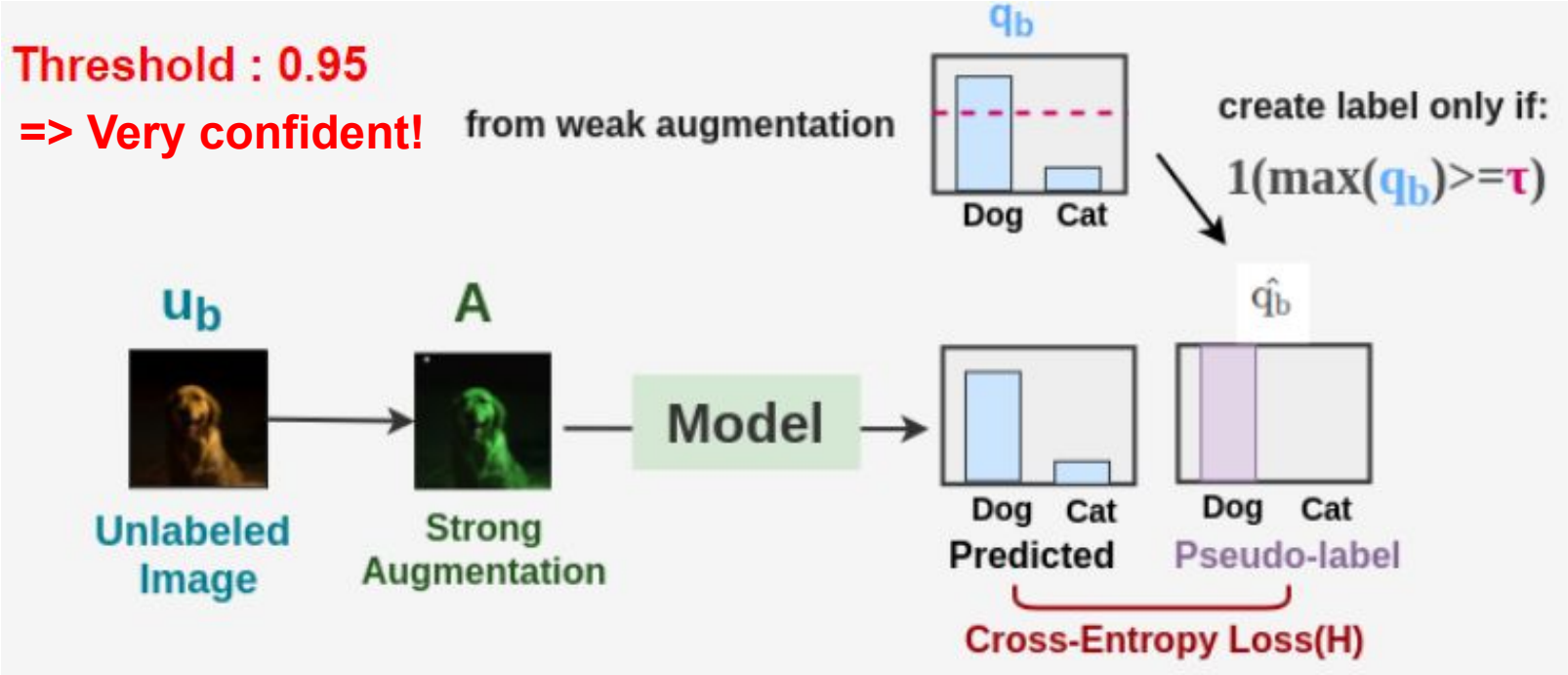
Label

Actual Label

Pseudo Label
from weak-augmented

?

4. Consistency Regularization



4. Consistency Regularization

$$Loss = L_x + lambda_u * L_u$$

$$L_x = crossEntropy(pred_x, label_x)$$

$$L_u = crossEntropy(pred_u, label_u) * mask$$

threshold masking

Experiment

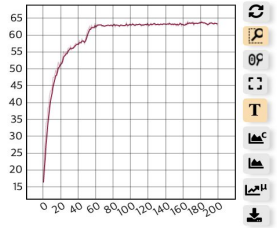
- Architecture
  - Resnet18 / EfficientNet

- Training
  - random augmentation(crop, brightness, color, rotate, shear, ...)
  - smaller lambda_u for train beginning efficiency

Results

train/loss/acc

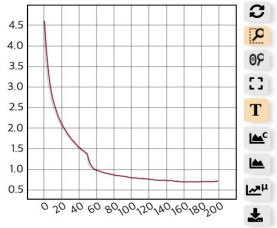
Min: 1.62e+1 Max: 6.41e+1



step

train/loss/avg

Min: 6.67e-1 Max: 4.60e+0

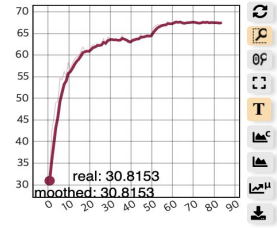


step

+ randaugment
+ strongAug to labeled

train/loss/acc

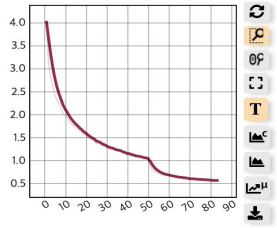
Min: 3.08e+1 Max: 6.78e+1



step

train/loss/avg

Min: 5.51e-1 Max: 4.03e+0



step

+ EfficientNet

	Best top1 acc	Best top5 acc
Baseline(Res18)	56.42	78.98
RandAugment(Res18)	63.56	83.32
Reinforced mix-up(Res18)	63.44	83.69
Pseudo-label&Mask(Res18)	64.69	83.52
RandAugment With no Mix-up(Efficient)	68.19	85.2
RandAugment With reinforced Mix-up(Efficient)	69.62	86.84

Conclusion

- Dividing it into strongly augmented one and weakly augmented one through Randaugment, and achieved model's performance improvement.

- Pseudo-labeling of unlabeled data was done in a situation where the threshold was higher than 0.95, so the only highly confident data was used.

- Subsequent improvements will be made by fine-tuning the hyper parameters and increasing the precision of the loss calculation.