Deep reference prior

CIFAR10

Semi-supervised learning

Setting:

Data
- **use 250 labels, i.e., 25 per class**

Optimization
- **nesterov SGD, lr=K * 0.03, weight_decay = 5e-4 / K (not apply to BN and prior)**
- **cosine schedule without warmup (follow repo code for comparison)**
- **mix-precision training**

Hyper-params (baseline)
- **alpha=0.05, gamma=0.5, tau=1/3, ema_decay=0.999**

Augmentation
- **Labeled: Horizontal flip + crop + normalize**
- **Unlabeled weak: Horizontal flip + crop + normalize**
- **Unlabeled strong: Horizontal flip + crop + RandAugment(n=2,m=10) + normalize**
- **Test: normalize**

Evaluation:

acc: deep prior accuracy(ensemble on probability not prediction)
ema acc: deep prior accuracy with ema params

All acc is computed on the full test set

Results:

Default hyper-params: alpha=0.05, gamma=0.5, tau=1/3, ema_decay=0.999, epoch=50

1.Test train-able prior (uniform vs allow backprop):

	acc	ema acc	loss	H_y	H_yw	ce_loss
Trainable prior	**0.809**	**0.809**	**0.111**	**0.141**	**0.105**	**0.0002**
Uniform prior	**0.792**	**0.824**	**0.132**	**0.455**	**0.139**	**0.0013**
	0.809	**0.834**	**0.132**	**0.455**	**0.139**	**0.0012**
UniformPrior avg	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**

Uniform prior (prior is not train-able) outperforms -> no need to weight particles

2.Test H_yw computation (n_order=2 vs n_order=1)
We are currently computing H_yw as if n_order is 1, but theoretically they should be same.

3.Test H_y computation

	acc	ema acc	loss	H_y	H_yw	ce_loss
default	**0.785**	**0.819**	**0.129**	**0.385**	**0.134**	**0.0006**
use weak_aug	**0.794**	**0.822**	**0.147**	**0.163**	**0.138**	**0.0012**

Try to use only weak augment to see the benefit from strong augment.

4.Test mix_up
Mixup alpha=beta=0.75 -> run for ~20 epochs, hurts performance

5.Test bn_momentum

Increase from 0.001 to 0.1 to see the difference. (Not improve, but not hurt too much)

	acc	ema acc	loss	H_y	H_yw	ce_loss
default(bn=0.001)	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**
bn=0.01	**0.803**	**0.821**	**0.131**	**0.452**	**0.138**	**0.0011**
bn=0.1	**0.812**	**0.813**	**0.141**	**0.433**	**0.146**	**0.0001**

Increasing BN slightly hurts, but did not observ the “very large effect on accuracy” mentioned in the paper

6.Test tricks

Ablation study of each trick

6.1 Gradient stopping

	acc	ema acc	loss	H_y	H_yw	ce_loss
default	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**
No grad stop	**0.277**	**0.332**	**0.007**	**0.636**	**0.354**	**0.0028**

When using backprop on weak augment, progress has stagnated since epoch 2.
But the gap is too larg?

6.2 Jensen's inequality

	acc	ema acc	loss	H_y	H_yw	ce_loss
default	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**
No jensen	**0.10**	**0.10**	**NaN**			

At epoch 7, everything is still fine and ema acc ~0.45. It corrupted later when H_yw becoomes zero.

6.3 Label threshold

	acc	ema acc	loss	H_y	H_yw	ce_loss	mask_ratio
default(0.95)	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**	**0.894**
thresh = 0.99	**0.813**	**0.832**	**0.108**	**0.473**	**0.119**	**0.0013**	**0.864**
thresh = 0.9	**0.809**	**0.823**	**0.142**	**0.449**	**0.147**	**0.0013**	**0.946**
thresh = 0	**0.775**	**0.815**	**0.203**	**0.360**	**0.202**	**0.0018**	**1.0**

High threshold helps, but the difference is small.

6.4 EMA decay

	acc	ema acc	loss	H_y	H_yw	ce_loss
default(0.999)	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**

decay = 0.99	**0.797**	**0.819**	**0.130**	**0.454**	**0.137**	**0.0013**
decay = 0.9	**0.819**	**0.815**	**0.131**	**0.460**	**0.139**	**0.0010**

Slow update makes it more stable.

7.Test hyper-params

Ablation study of each hyper-param

7.1 Alpha

	acc	ema acc	loss	H_y	H_yw	ce_loss
default(0.05)	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**
alpha=0.1	**0.810**	**0.835**	**0.095**	**0.546**	**0.138**	**0.0012**
alpha=0.2	**0.715**	**0.740**	**-0.266**	**1.608**	**0.083**	**0.0019**
alpha=0.5	**0.142**	**0.155**	**-1.261**	**2.280**	**0.017**	**0.0019**

7.2 Tau

	acc	ema acc	loss	H_y	H_yw	ce_loss
default(1/3)	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**
tau = 0.1	**0.697**	**0.749**	**0.138**	**0.398**	**0.153**	**0.0033**
tau=0.5	**0.806**	**0.836**	**0.137**	**0.486**	**0.127**	**0.0011**

7.3 n_order

	acc	ema acc	loss	H_y	H_yw	ce_loss
default(2)	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**
n_order = 1	**0.813**	**0.826**	**0.093**	**0.462**	**0.105**	**0.0004**
n_order = 4	**0.816**	**0.820**	**0.102**	**0.292**	**0.105**	**0.0004**

7.4 weight_decay

	acc	ema acc	loss	H_y	H_yw	ce_loss
default(1e-4)	0.801	0.829	0.132	0.455	0.139	0.0012
wd = 5e-5	0.806	0.818	0.089	0.442	0.101	0.0001
wd = 5e-4	0.816	0.820	0.102	0.292	0.105	0.0004

8 Augmentation

8.1 num_ops

	acc	ema acc	loss	H_y	H_yw	ce_loss	mask_ratio
default(2)	0.801	0.829	0.132	0.455	0.139	0.0012	
num_ops=4	0.852	0.862	0.189	0.455	0.189	0.0011	0.924
num_ops=6	0.872	0.896	0.242	0.484	0.238	0.0009	0.909
num_ops=8	0.846	0.884	0.027	1.652	0.105	0.0019	0.303

n=2,m=10: left=strong, right=weak

n=4,m=10: left=strong, right=weak

n=6,m=10: left=strong, right=weak

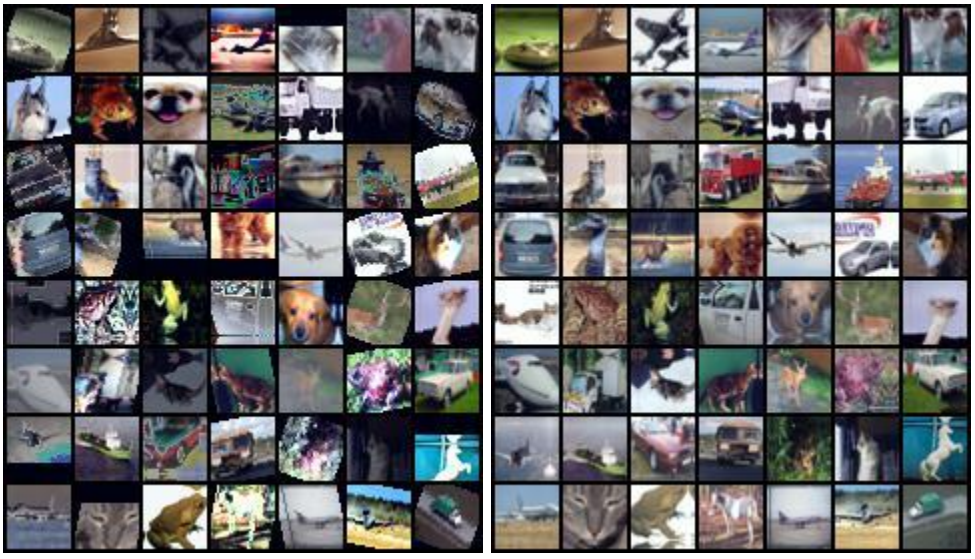
Observation: Images tolerate different levels of augmentation before distrotion

8.2 magnitude

	acc	ema acc	loss	H_y	H_yw	ce_loss	mask_ratio
default(10)	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**	
magnitude=5	**0.687**	**0.745**	**0.110**	**0.538**	**0.123**	**0.0022**	**0.911**
magnitude=20	**0.884**	**0.896**	**0.227**	**0.457**	**0.224**	**0.0009**	**0.927**

magnitude=30	**0.879**	**0.896**	**0.379**	**0.506**	**0.361**	**0.0009**	**0.908**

n=2,m=10: left=strong, right=weak



n=2,m=20: left=strong, right=weak

n=2,m=30: left=strong, right=weak

Observation: Similar as num_ops, magnitude should not exceed a certain threshold

8.3 n_ops & magnitude

n_ops sampled uniformly from [n1,n2], magnitude samled from [m1, m2]
default = ([2,2], [10,10])

	acc	ema acc	loss	H_y	H_yw	ce_loss	mask_ratio
default	**0.801**	**0.829**	**0.132**	**0.455**	**0.139**	**0.0012**	
n=4, m=15	**0.879**	**0.896**	**0.308**	**0.490**	**0.297**	**0.0013**	**0.907**
[2,4], [10,20]	**0.868**	**0.880**	**0.276**	**0.487**	**0.269**	**0.0016**	**0.912**
[2,5], [15,25]	**0.861**	**0.886**	**0.443**	**0.521**	**0.419**	**0.0015**	**0.899**

9. Final Run

Epoch=200
Group v1(default): alpha=0.05, gamma=0.5, tau=1/3, ema_decay=0.999,
Group v2: alpha=0.05, gamma=0.5, tau=1/3, ema_decay=0.999, nops=[2,5], magni=[15,25]
Group v3: alpha=0.05, gamma=0.5, tau=1/3, ema_decay=0.999, nops=[3,5], magni=[15,25]
Group v4: alpha=0.05, gamma=0.5, tau=2/3, ema_decay=0.999, nops=[3,5], magni=[15,25]

Result

	acc	ema acc	loss	H_y	H_yw	ce_loss	mask_ratio

v1	**0.866**	**0.883**	**0.054**	**0.379**	**0.067**	**0.0003**	**0.985**
v2	**0.928**	**0.940**	**0.287**	**0.437**	**0.277**	**0.0004**	**0.958**
v3	**0.846**	**0.913**	**0.016**	**1.745**	**0.101**	**0.0008**	**0.297**
v4	**0.846**	**0.899**	**-0.068**	**1.973**	**0.061**	**0.0008**	**0.227**

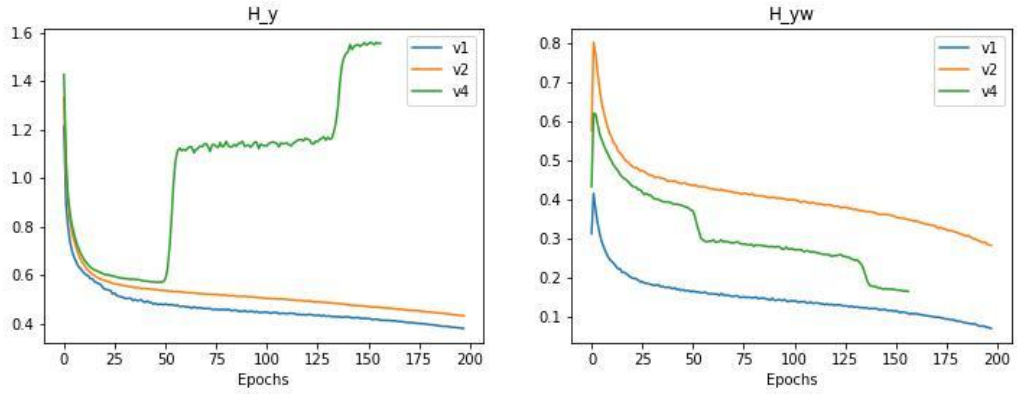
Observations:

- V2 gives the best ema val acc of 0.939 (0.056 improvement compared to v1), telling us stronger augmentation helps
- V3 reached a max ema val acc of 0.915 at epoch=113, but then began to drop rapidly.
- V4 copies V3 but increases Tau to ⅔, i.e., giving strong augment less weight, but particles still corrupted halfway

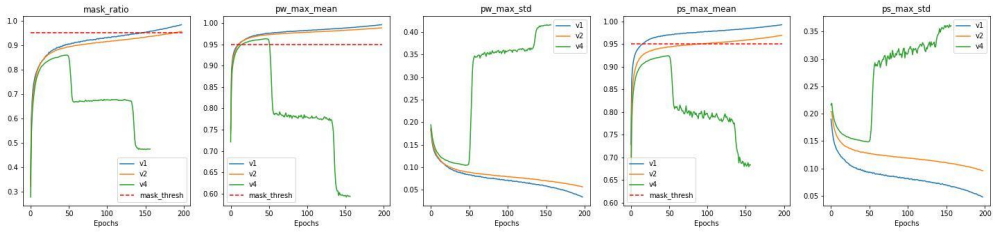
Analysis

V1 - V2 - V4

- H_y grows up



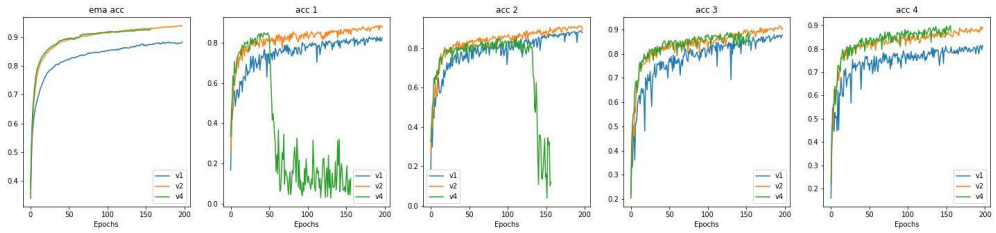
  - Particles are more diverse
- Mask_ratio drops



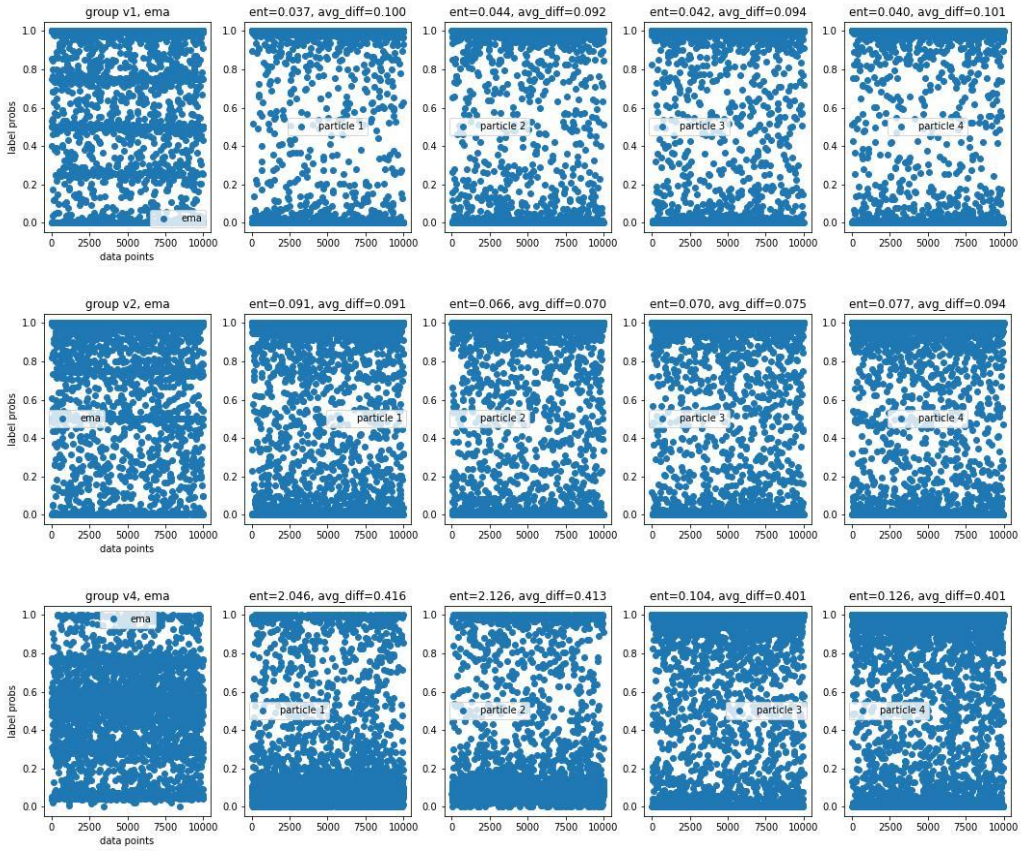

```
p_w = prob_weak_augment    # shape: (bz,n_label,K)
p_s = prob_strong_augment  # shape: (bz,n_label,K)
pw_max = p_w.max(dim=1)[0]
ps_max = p_s.max(dim=1)[0]
pw_max_mean = pw_max.mean() # fig 2
```

```
pw_max_std = pw_max.mean()  # fig 3
ps_max_mean = ps_max.mean() # fig 4
ps_max_std = ps_max.mean()  # fig 5
```

    - Particles are less confident (in terms of weak augment)
- Particles corrupt



    - Corrupted particles brought down ema acc (compared to v2, but its log is missing, will fill it later), but it remains high as long as some are still valid




```
# outs is softmax probs, shape: (n_val, n_label, K)
label_preds = outs[range(len(outs)), labels, :]
label_ema_preds = ema_outs[range(len(ema_outs)), labels]
mean_diff = np.abs((label_preds[:, i]) - label_ema_preds).mean()
```

    - Corrupted particles start to make random predictions

Explan & Todo:

Explain:

- Stronger augmentation captures more variation (“nuance”) to improve performance
- Too strong augment produces unrealistic images, noise finally disrupts the training process

Todo:

- Use particle corruption as indication of augmentation being too strong?