Implementation of Bootstrapped MAE and Analysis

Kai Zhang

Summary

This report shows the implementation of Bootstrapped MAE algorithm according to the coding test.After the implementation,I did the linear evaluation and whole network fine tuning on the training set and compared the validation accuracies results.Experiment analysis and other phenomenas observed throughout the whole code implementation are also included.

Firstly,I implemented the original MAE algorithm based on DeiT-Tiny network architecture. DeiT-Tiny network architecture has little difference with the ViT network architecture, for adding a distillation token and more data enhancement strategies.etc.That's not important,though,we just use it as the encoder architecture of MAE.After implementation,I pretrained the MAE on the cifar-10 datasets,and got the trained MAE encoder.With the trained encoder,I did the linear evaluation,and got the best accuracy rate as 75.15% before fine tuning.After fine tuning,the best accuracy rate could reach 73.47%(It should be higher than linear evaluation,but for the limit of epoch numbers and time,it's not the best).The implementation details and experiment results are showed in Part I.

Then,I implemented the Bootstrapped MAE algorithm based on the original MAE.That's not hard to implement:I changed the training target to the previous MAE encoder's output as epochs went by,and encapsulates the code details,making it very simple to call and use.After implementation,I did the same pretrained work,linear evaluation,whole network fine tuning as Part I.Of course,to compare the Bootstrapped MAE algorithm and figure out the best bootstrapped there were more experiments and analysis.More details are showed in Part II.The best results of all MAE-K models belong to MAE-n,which got the best accuracy rate as 75.37% in the linear evaluation and 73.44% (It should be higher than linear evaluation,but for the limit of epoch numbers and time,it's not the best)after the whole network fine tuning.

For Part III,I analyzed the experiments results with my own observation and understanding.Besides,I sorted out more implementation details and tricks during the experiment,and also pointed out the problems in my implementation process,and some possible errors in principle.Although the network worked,there might be errors in principle of the my implementation.

In the end,I must point out that my code implementation is based on open source code,and the Github links of the origin code and my code are listed here:

Github original code: https://github.com/liujiyuan13/MAE-code

My implementation code: https://github.com/zhangkai0425/MAE-K

Contents

1 Part I:MAE

1.1 DeiT-Tiny

The Deit-Tiny network is like the Vit network,just adding a distillation token and more data enhancement strategies.I used it as the architecture of MAE encoder.There are offical DeiT repository on Github,and my implementation was based on that: https://github.com/facebookresearch/deit

There are two models called *deittiny* in the Github repository,the first is *deit-tiny-patch16-224*,which is very similar to ViT.In fact,it just uses ViT model from *timm.models.vision_transformer*.The other is *deit-tiny-distilled-patch16-224*,which contains the distillation token and differs more from ViT model.I used the last one,because I first used the ViT to implement the MAE encoder,which was not in line with the task setting,so I believed that the last one was more different from the ViT.

The parameters of deit-tiny are as Table1.Note that my Deit-Tiny has some differences with the official one,the most import one is that the embed_dim is changed from 192 to 48.That's because 48 equals to 4*4*3,which is one patch's pixels number(with 3 channels),and that's easier for me to use it in the Bootstrapped MAE training strategy.On the other hand,I believe that if a patch has only 48 pixels in total,encoding it with more than 48 dims features is not helpful,as information entropy does not increase.(This is just a qualitative thought,the main purpose is to make the code simpler.)

Table 1 Parameters description.

Parameters	Value
img_size	32*32*3
patch_size	4*4
num_classes	10
embed_dim	48(different from the origin)
depth	12
num_heads	3
mlp_ratio	4
qkv_bias	True
norm_layer	partial(nn.LayerNorm, eps=1e-6)

1.2 MAE

The implementation of MAE contains a encode(deit-tiny) and a decoder. Deit-Tiny is introduced in the 1.1.Now let look at the decoder part:

Actually,my code is based on the public code in Github,the code have already implementd the MAE,while his encoder is the ViT model.I used his decoder with

only little modification.The decoder is also a transformer,after the transformer output,it uses a linear network to change the output's shape to the same shape of the input image,and then it can calculate the MSE loss with input and output.

The parameters of decoder are shown as Table2.

Table 2 Parameters description.

Parameters	Value
decoder_dim	512
decoder_heads	8
decoder_dim_heads	8
depth	8

1.3 Pretrain

In the pretrain part,I use the cifar-10 dataset without any other datasets or pretrained models.The cifar-10 dataset contains 50000 images for training,and 10000 images for validation.I use the batchsize as 512,and training for most 200 epochs.Every 10 epochs,I test the model on the validation dataset,logging the accuracy and save the model as checkpoint file.After that,I plot the training loss curve as Fig1.(The codes contains tensorboard,but I didn't use that because I trained for many times that it easy to confuse the logging files.So I just export the command line output to log.txt and visualize the result myself.)

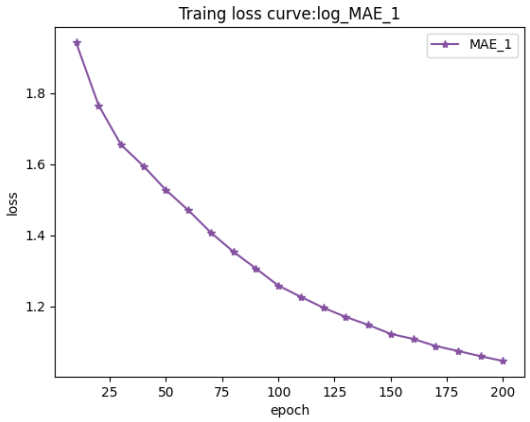

Figure 1: Training loss curve:MAE1

1.4 Linear Evaluation

To do the linear evaluation,I also only do a little modification on the Github code.In a nutshell,the linear evaluation part uses the output of pretrained MAE model's encoder output.So I just need to fix the MAE model encoder,and train the

last linear network.Note that there is a difference between my implementation and the task setting:the task setting says I should use 8*8 = 64 tokens as the input of the linear network,but I use 64 + 2 = 66 tokens as the input.That's because most transformer paper says they only use the cls token as input,if I only use the 64 tokens of image patches,cls token will not be used,besides the distillation token.I'm not sure about that,so I use 66.But on the other hand,even if the cls and distillation tokens should not be used,the linear network will automatically drop the two tokens during the linear probing process.Because if this two tokens are not useful,the linear network((fully connected layer) will adjust the corresponding weight to zero,as epoch goes by.I set the max epoachs as 900,all the parameters are in Table3.

Table 3 Parameters description(Linear Evaluation).

Parameters	Value
batch_size	512
epochs	900
base_lr	5e-2 (will change with the epoch and batch_size)
weight_decay	0
momentum	0.9
optimizer	LARS

I also plot the accuracy and epochs shown in Fig2.

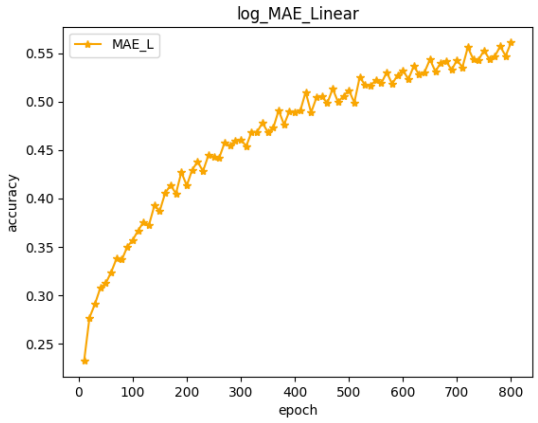

Figure 2: Linear Evaluation:MAE1

1.5 Fine tune

In the fine tune part,I don't need to fix the MAE encoder,but to release it and to train it together with the fc linear network.There,if I want to fine tune the network,I should not use a large learning rate.So I use a learning rate lower than the linear evaluation process.The accuracy and epochs are in Fig3.I set the max epoachs as 900,all the parameters are in Table4.

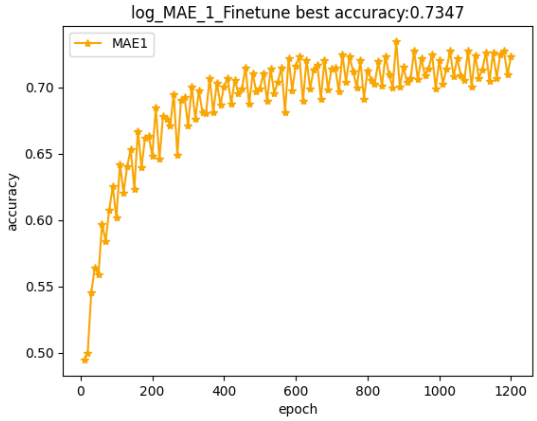

Figure 3: Finetune:MAE1

Table 4 Parameters description(Finetune).

Parameters	Value
batch_size	512
epochs	1200
base_lr	2e-2 (will change with the epoch and batch_size)
weight_decay	5e-2
momentum	0.9-0.99
optimizer	AdamW

2 Part II:Bootstrapped MAE

2.1 Bootstrapped MAE

The most import part of this report and the task is the implementation of Bootstrapped MAE.Although it's not hard to implement,I need to talk about the details during the implementation.

First,I modify the original encoding dimension.As I said before in Part I,I changed the encoding dims of DeiT-Tiny from 192 to 48,making it easier to compute the reconstruct loss between the previous MAE encoder output and the current MAE decoder output.As the decoder output is 4*4*3(Actually in the code,the decoder first outputs a 512 dims feature,and then a fc net module transforms it linearly to 4*4*3),now we can treat the previous encoder output as a image.Therefore,computing the MSE loss becomes really really easy.

Second,as the task requires,I use the DeiT-Tiny network as the encoder,so I have to consider the cls token and dist token.When I encode the tokens,I add the cls token and dist token,all of them go through the tranformer and attention module together.However,after the tranformer and attention module.I'm not sure whether it is rational,but it works.

To make it more clear,let me retell the operation agagin:When the MAE encode works,it encodes the unmasked patches,which is 64*0.25 = 16 in our case.However, considering the cls token and dist token,now I add this two tokens,so we get 16 + 2 = 18 tokens,then,I put the masked tokens(without encoding,only positional embedding and random initial values) together with the 18 tokens,now 48+18 = 66 tokens in total into the decoder network.The output is 66 decoded tokens,but we only choose the masked 48 tokens to compute the reconstruction loss with original image.And,when I need to implement the Bootstrapped MAE,I have to compute the MSE loss of the previous MAE encoder output(let's call it MAE-(K-1)).So,for the MAE-(K-1),I encoder the 2 + 64 = 66 tokens,they go through the transformer block and attention module,blabla.After that,I only save the 64 tokens as the reconstruction target,their dims are 64*48,the same as the input image.

In the end,to follow the task settings,I need to train the MAE-K from the weights of MAE-(K-1).I realize this by keeping a single model during the whole training process.That is to say,the model always will be there during the 200 epochs,but the MAE-(K-1) model will be saved as a temporary model when training the MAE-K model.

The words maybe still not clear enough,especially written in English.So,I would be very glad to talk about more code details if there can be a meeting online in the next few days!And here,let's we look at the results of the Bootstrapped MAE.

2.2 Pretrain

There is no need to repeat the pretrain process,as it is the same as MAE in part I.In my code,the max bootstrap times K of MAE-K is a parameter that can be changed.Therefore,I can do many experiments just by changing that parameter.With 8 GPUs from the Lab server(my current lab),I did lots of experiments in these days,and got many different results.For the limit of the report,I can only post

some results. Mostly,they are similar,although there is a general fluctuation with different learning rate or optimizer.I choose K as 2,4,10,20,50,100,200,here are the results during the pretrain process.

The results are in Fig4.From the results we can see that the Bootstrapped MAE training loss is smaller than the MAE,that mays the training will converge faster.However,whether smaller training loss is useful or not depends on the linear evaluation and finetuning accuracy results.

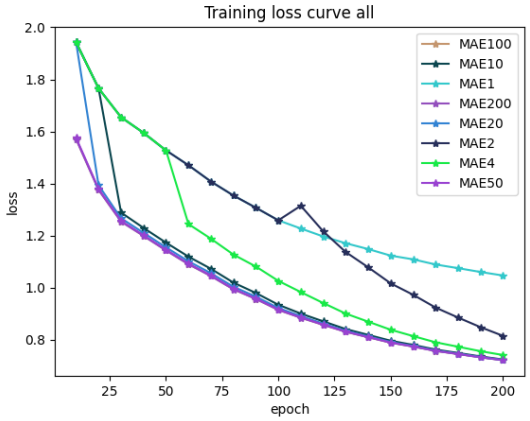

Figure 4: Training loss curve:MAE1-MAE200

2.3 Linear Evaluation

After pretraining,I do the linear evaluation(linear probing) in 900 epochs.The training parameters are the same as Table3.The results(along with MAE-1) are shown in Fig5 - Fig6.

Actually,the results are quite similar to each other.Although there exists difference,it is very tiny.Not only are they close in accuracy, but their behavior in training is also close to each other.I check the code for many times,and I really load different MAE-K models!But they perform really similarly.The best accuracy is from MAE-2,which achieves 75.37%.While the original MAE also reach 75.15%,they are quite similar.Besides,I did many experiments with different training parameters,this result is only one of them.Limited to the length of the report,I choose one of them,which final accuracy is better than others.

Therefore,from my experiments results,actually I didn't find much gain of the new algorithm.There are many probable reasons and I will discuss about that in the Part III.Now let's look at the results of whole network fine tune.

2.4 Fine tune

I use the same parameters as Table4 in Part I to do the whole network fine-tune.Here,I use the AdamW optimizer,the same as the MAE paper.And I choose the smaller learning rate,as it is the finetune stage.For other parameters,I use the same with the original Github code.

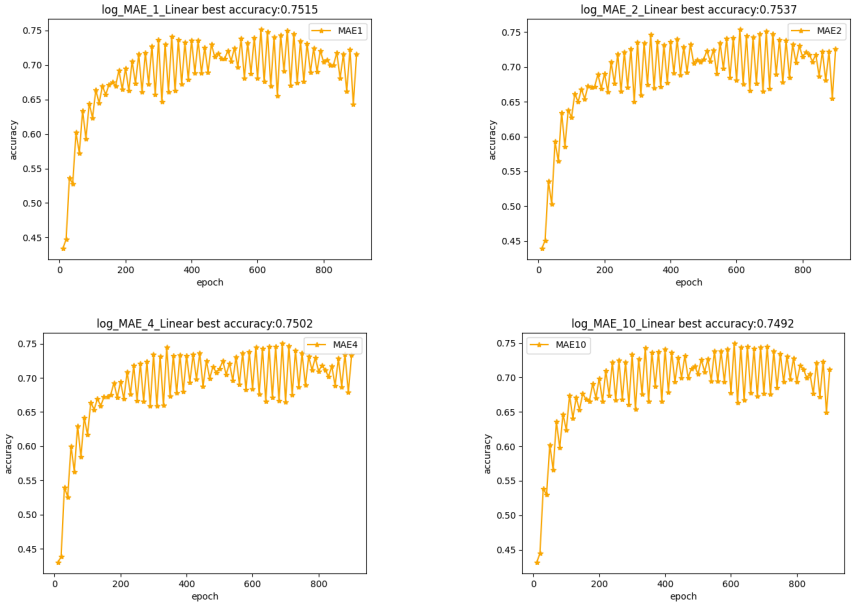

Figure 5: Linear Evaluation Results of MAE-[1 2 4 10]

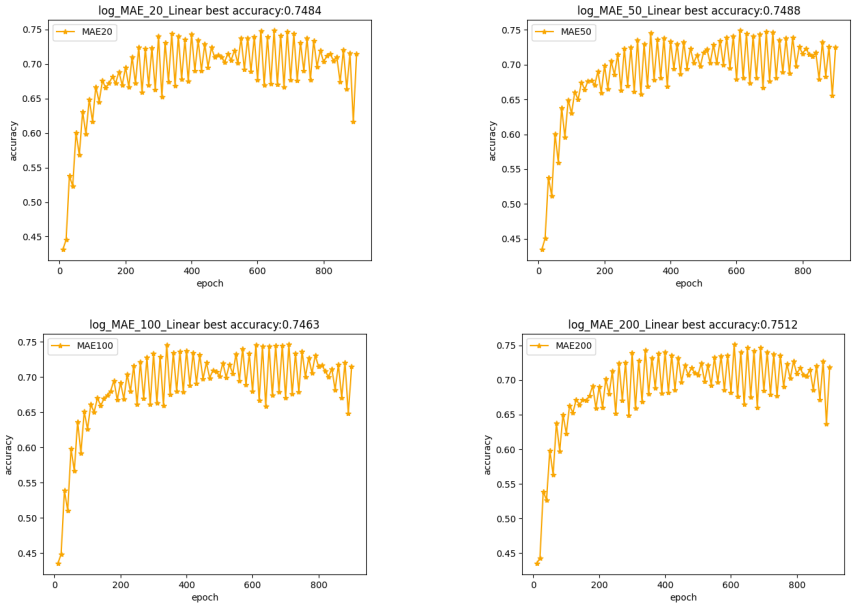

Figure 6: Linear Evaluation Results of MAE-[20 50 100 200]

The finetuned results are shown in Fig7 - Fig8.

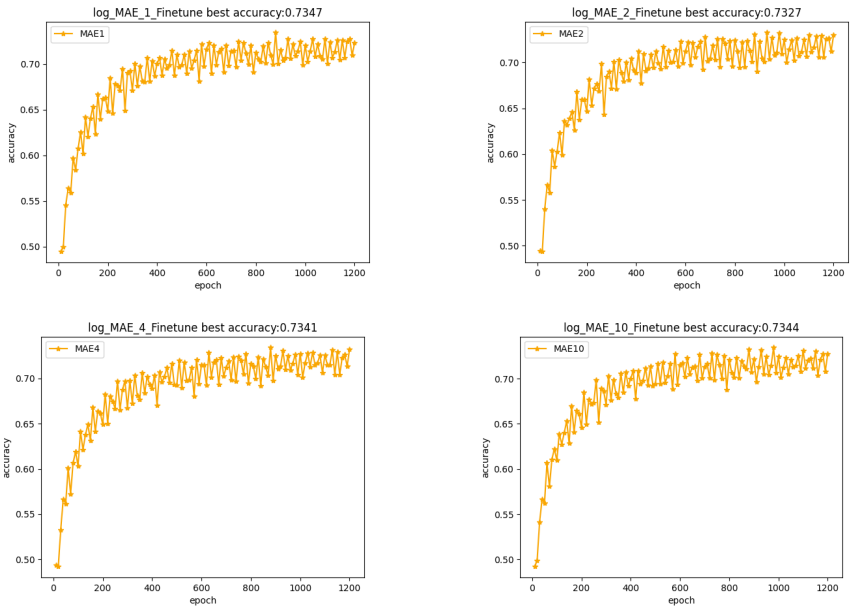

Figure 7: Finetune Results of MAE-[1 2 4 10]

3 Part III:Analysis

3.1 Comparison

I will talk about the comparison of MAE and Bootstrapped MAE basing on my experiment results.

As I observe in my experiment results,the Bootstrapped MAE algorithm is not better than the MAE on cifar-10.They perform similarly most of the time.There are many probable reasons:

- First,the max epoch in pretraining process is set to 200,and as I observe,the training loss is still decreasing,not reaches the lowest point.This is probably my learning rate or other parameters are not good enough,but I don't have much time to do my comparing experiments in the pretraining process.(Because I must do lots of linear evaluation and finetune experiments,and I have to do my other homework in the daytime.)So,based on my settings,the max epoch is not big enough.

- Second,I think semantic details is not enough in the cifar-10 datasets,because the images is not big enough,only 32*32.With so small and blurred images,even traditional methods like super pixel cannot get the good semantic segmentation.So,I think that even if Bootstrapped MAE algorithm works,it may not be useful in the cifar-10 datasets.Besides,I'm not sure whether the task conclusion is right.Calculating the reconstruct pixel MSE loss doesn't certainly mean that the MAE will lose the semantic details,because the encoder and decoder

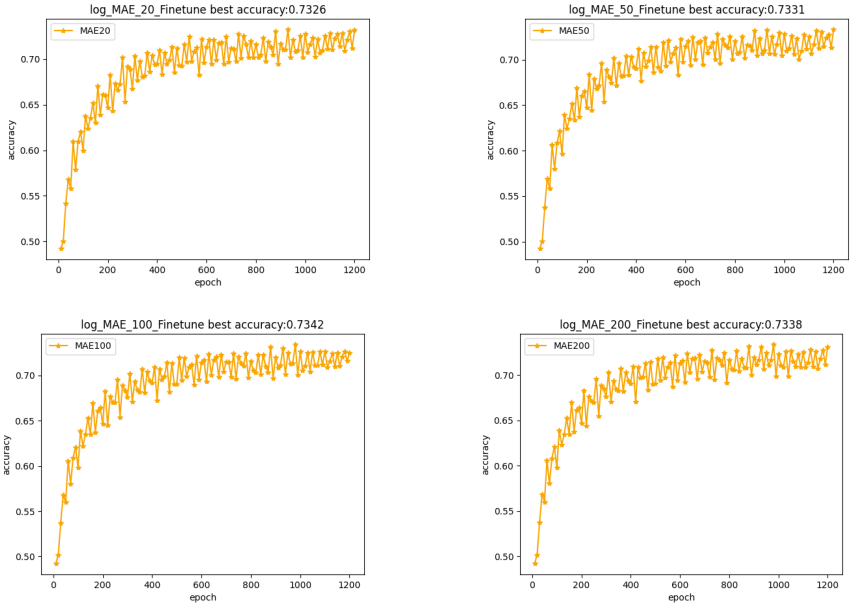

Figure 8: Finetune Results of MAE-[20 50 100 200]

is not symmetric.I think there is no strict proof that reconstructing pixel MSE with a unsymmetric decoder will lose the semantic details definitely.It's just a conjecture,in some sense.

- Another,actually,in the linear evaluation and finetune part,the accuracy still not get the best.Now the best is about 75%,that's because I tried about 3 learning rate and did the linear evaluation on all MAE-K models,and the last one reached about 75%.In the previous two cases,best accuracy is about 50% and 60%.There is still many learning rates to try,though.That is to say,the pretrained models still not fully performance.And if tring more learning rate and reach the full ability of the MAE-K models,I maybe find more differences between them.

3.2 Details

During my experiments,I find that some parameters matter much.For the learning rate,I try 3 different one,and I fine that if the learning rate is small,its best accuracy is about 60%,training for 1000 epochs.If the learning rate is too large,it will perform better in the beginning,but will quickly go down as epoch goes by.If I choose the middle one,I can get the best accuracy more than 70% for about 900 epochs.Learning rate matters in my experiments,and matters much more than the different MAE-K models.

Also,I first implemented the MAE with ViT as the original Github code.I did some experiments of linear evaluation and finetuning.After asking about the DeiT-Tiny,I changed the encoder from ViT to Deit-Tiny.What I found is that Deit-Tiny performed better than the ViT encoder,the accuracy is higher about 20 percent.

3.3 Other Ideas

There are many ideas during my implementation and experiments.Actually,I think the Bootstrapped MAE algorithm is not new.Although it is not a symmetrical training methods,it is still very similar to the communication equalization algorithm.(Decision directed mode in equalizer)That is to say,the algorithm use its 'own' output as the training target in the next training time,and change its own parameters.They are very similar,so I think although I didn't learn much transformer network in the past,most of the ideas and methods are the same.

I also think about the new methods of Bootstrapped MAE algorithm.One of them is that I can train the model using the bootstrapped strategy from the beginning.That is to say,I can use the output of the previous model encoder as reconstruct target all the time.I think it a good idea,but during my code implementation,I find that it is just a special case of the algorithm:I just need to change the K to 200 and I can realize the idea.So I tried that.

In addition,if there is more time,I'd like to try this method on more datasets,like image-net.However,considering the training time will be too long,and I really have to do other homework projects recently as the end of the term approaches,I cannot try that idea recently.

Actually,when I begin to train MAE-K,I think the results will be very different,maybe the middle of the MAE-K is the best.And I even think about the reason before the experiments results:That is,if the k is too small,like MAE-1,it will lose the semantic details as the task setting.If the k is too large,like MAE-200,it will not learn the deep feature of the image before it bootstraps to the next stage.However,the result breaks my initial idea.

There are other tiny ideas and thoughts during the experiments,but I cannot write all of them due to the limit of time.I would be very happy to talk about if there is a meeting with you before the summer camp!