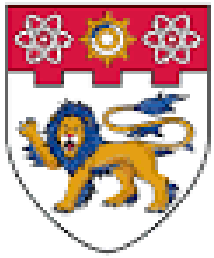


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Master in Science (Artificial Intelligence)

(MSAI)

AI-6103

Final Project

Name of Student:
Teo Lim Fong

Matriculation No: G2101964G

Tale of content

Contents

Introduction

Overfitting is a major problem in neural network that usually resulting good training losses and accuracy but poor testing or validation accuracy and losses. The cause of overfitting is mainly due to the either lack of training datasets or training and testing datasets are not in the same distribution or the model learned noise or random fluctuation in the training datasets. The common solutions for overfitting are mainly early stopping, data augmentation, regularization and dropout.

Data augmentation can solve overfitting by using different augmentation to the training datasets such as rotating, flipping, adjust the brightness, adding occlusion to avoid the network from memorizing the feature in the training datasets. Weight decay also known as L2 norm is another method that can prevent overfitting by adding square of its weight and small weight decay coefficients into the loss function to prevent the weight from getting large in the network. However, if the value of weight decay coefficients is too large, it will not learn well and might result in underfitting. Dropout is another solution for overfitting that random remove layers in the neural network.

In the next section, I will be introducing four other regularizations techniques. There are Stable Weight Decay, Jacobian Regularizations, CutMix augmentation and DropBlock.

The following experiments will be using the parameters listed in Annex A.

Methodology

Stable weight decay

Weight decay is normally coupled with learning rate scheduler in SGD. However, how you ever wonder why weight decay must be coupled with learning rate scheduler? In this paper **[1]**, the author will analysis the behavior of weight decay with different configuration and proposed a weight decay scheduler for deep learning which is also known as stable weight decay (SWD).

So, why weight decay is normally coupled with learning rate scheduler in SGD?

$$\theta_t = (1 - \lambda')\theta_{t-1} - \eta\frac{\partial L(\theta_{t-1})}{\partial \theta}, \qquad (1)$$

$$\theta_t = (1 - \eta_t\lambda)\theta_{t-1} - \eta_t\frac{\partial L(\theta_{t-1})}{\partial \theta}, \qquad (2)$$

In equation 1, constant learning rate is used with certain weigh decay coefficient λ'. In equation 2, learning rate scheduler is used together with weigh decay coefficient. Theoretically, equation 2 will be more robust because we understand that in order to optimize the network learning rate decay is needed. Therefore, the author defined a statement that is '*The weight decay is stable if the stationary point are stable during training*.' The author stated this statement because unstable weight decay can cause fluctuation in the training and does not coverage which is the main reason why weigh decay must be coupled with learning rate scheduler.

Normally, we avoid large batch size during training because it will not coverage well and will have a higher chance of stucking at the local minima instead of global. However, **[2]** figure out the linear relationship between learning rate and the number of batch improved the performance of the network. This means that if the number of batch increase by a factor of *k*, then the learning rate also must be increase by a factor of *k*. Even though large batch size improved the performance of the network, but it is too computationally expensive and according to the article, people tend to remain the number of epochs in training and decrease the number of iteration by *k*. With the large number of batches and smaller iteration mean that the network cannot coverage. This conclude that the batch-learning rate is not ideal. On the other hand, instead of the increasing the batch-learning rate, the author proposed the batch-weight decay incrementation because large weight decay coverage faster. This means that if the number of batch increase by a factor of *k*, then the weight decay also must be increase by a factor of *k*.

Next, we will talk about the instability of decoupled weight decay in adaptive gradient methods. The reasons why AdamW is unstable during stationary point is because the decoupled weight decay does not consist of effective learning rate. The explanation is listed below.

Equation 3 shown the formula for AdamW.

$$\theta_t = (1 - \eta\lambda)\theta_{t-1} - \eta v_t^{-\frac{1}{2}}\frac{\partial L(\theta_{t-1})}{\partial \theta}, \qquad (3)$$

In the formula, the author identified $\eta v_t^{-\frac{1}{2}}$ as effective learning rate and $v_t^{-\frac{1}{2}}$ refers to the element-wise power of the vector. The first part of the formula is called the decoupled weight decay, which

consists of the learning rate $\eta\lambda$ instead of the effective learning rate $\eta v_t^{-\frac{1}{2}}$. Thus, this is the reason why AdamW is unstable at stationary. However, the author solve the issue by proposing a non - element-wise scheduler by adding weight decay scheduler into decoupled weight decay. The new formula look as shown in equation 4.

$$\theta_t = (\mathbf{1} - \eta\bar{v}_t^{-p}\lambda)\theta_{t-1} - \eta v_t^{-\frac{1}{2}}\frac{\partial L(\theta_{t-1})}{\partial\theta}, \qquad (4)$$

According to the author, the proposed weight decay scheduler SWD demonstrate an impressive result using ADAMS. The initial paper experimented Cifar 10 using vanilla net, with step size learning rate scheduler and ADAMS.

For my experiment, I will be incorporating Stable Weight Decay into Resnet-18 (training section) and experimenting with SGDS and AdamS. The 'S' refers to weight decay scheduler.

	Stable weight decay	
	SGDS	AdamS
Training loss	0.0006	0.0010
Testing loss	0.2218	0.2187
Training accuracy	99.99	100
Testing accuracy	95.05	94.77

Table 1: Comparing different optimizers using weight decay scheduler

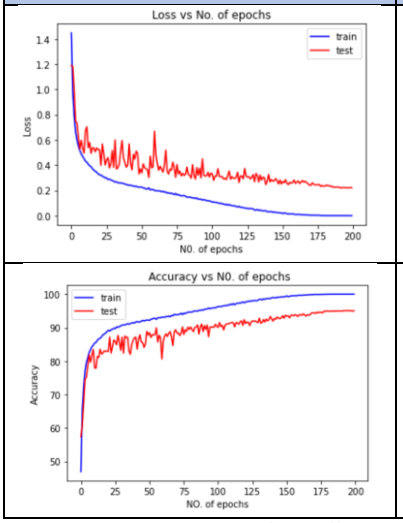
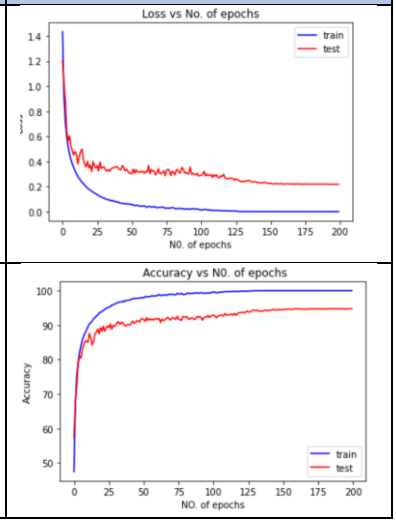
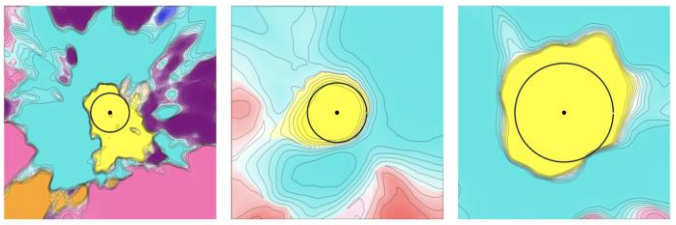
SGDS	AdamS
	

Table 2: Diagram of the different optimizers using weight decay scheduler

Jacobian regularization

Overfitting can cause instability to the network and the common solution to that is to add weight decay to reduce the overfitting in the training datasets. However, it is still remained ambiguous whether by adding weight decay stabilize the network and maximize the classification margins. Therefore, the author implemented a novel Jacobian Regularization **[3]** that can enhance the stated issue.

In this paper, the author experimented a cross section decision boundary using (a) without regularization, (b) with weigh decay and (c) with Jacobian Regularization. The colors represent the different classes predicted and we can see that without regularization, the decision boundary is quite messy. With weight decay, the decision boundary is quite smooth but it does not have maximize classification margin.



(a) Without regularization (b) With L^2 regularization (c) With Jacobian regularization

Jacobian regularization is proposed to enhance the classification margins by using input-output Jacobian matrix. The aim of this Jacobian regularization is to reduce the losses in the training by minimizing the Frobenius Norm with stochastic gradient descent (SGD) and remain stable to enhance the classification margins.

$$\widetilde{z}_c = f_c(\boldsymbol{x}+\boldsymbol{\epsilon}) = f_c(\boldsymbol{x}) + \sum_{i=1}^{I} \epsilon_i \cdot \frac{\partial f_c}{\partial x_i}(\boldsymbol{x}) + O(\epsilon^2) = z_c + \sum_{i=1}^{I} J_{c;i}(\boldsymbol{x}) \cdot \epsilon_i + O(\epsilon^2) \qquad (5)$$

In equation (5) , the formula shown the stability model prediction using input-output Jacobian. When the values of Jacobian is huge, the prediction will become unstable. To address this issue, we need to minimize the magnitude of Jacobian matrix by minimizing the square of the Frobenius norm. One good thing about Jacobian regularization is that, it can work well with other regularization such as weigh decay L_2 norm. In supervised learning, supervised loss function is used to optimize with other regularization to minimize the bare losses. The bare loss function can be defined as

$$\mathcal{L}_{\text{bare}}\left(\{\boldsymbol{x}^\alpha, \boldsymbol{y}^\alpha\}_{\alpha\in\mathcal{B}}; \boldsymbol{\theta}\right) = \frac{1}{|\mathcal{B}|} \sum_{\alpha\in\mathcal{B}} \mathcal{L}_{\text{super}}\left[\boldsymbol{f}(\boldsymbol{x}^\alpha); \boldsymbol{y}^\alpha\right] + \mathcal{R}(\boldsymbol{\theta}) \qquad (6)$$

However, in order to find the joint loss, we need to incorporate Jacobian regularizer into the above equation by adding the Frobenius norm into mini-batch $\mathcal{B}$ with λ_{JR} parameter. λ_{JR} is a Jacobian regularizer hyperparameter that are able to finetune to get the optimal joint loss. Accordingly to the author, with the correct λ_{JR} parameter, the model will be more robust and large classification margins.

$$\mathcal{L}^{\mathcal{B}}_{\text{joint}}\left(\boldsymbol{\theta}\right) = \mathcal{L}_{\text{bare}}(\{\boldsymbol{x}^\alpha, \boldsymbol{y}^\alpha\}_{\alpha\in\mathcal{B}}; \boldsymbol{\theta}) + \frac{\lambda_{\text{JR}}}{2}\left[\frac{1}{|\mathcal{B}|}\sum_{\alpha\in\mathcal{B}} ||J(\boldsymbol{x}^\alpha)||^2_{\text{F}}\right] \qquad (7)$$

For my experiment, I will be incorporating Jacobian Regularization into Resnet-18 (training section) and experimenting with λ_{JR} = 0.1 with weight decay and λ_{JR} = 0.1 with no weight decay.

	Jacobian Regularization	
	λ_{JR} no weight decay	λ_{JR} with weight decay
Training loss	0.0103	0.0366
Testing loss	0.6095	0.268
Training accuracy	100	99.99
Testing accuracy	80.25	91.54

Table 3: Comparing Jacobian Regularization with and without weight decay

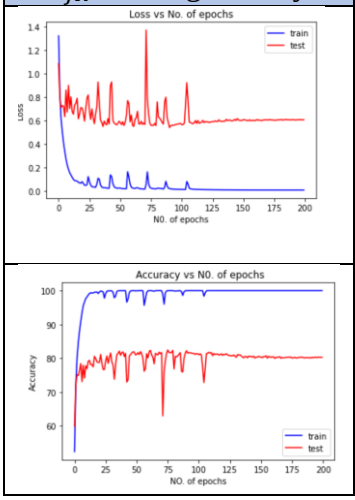
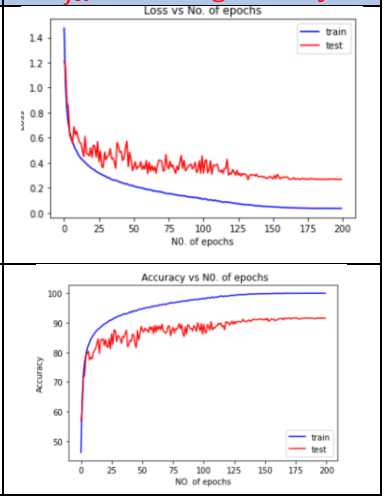
λ_{JR} no weight decay	λ_{JR} with weight decay
	

Table 4: Diagram of the Jacobian Regularization with and without weight decay

CutMix Augmentation

Data augmentation is an important feature that manipulate the orientation of the training datasets to avoid overfitting. The common data augmentation are random cropping, horizontal flipping, changing of brightness and colors. These type of data augmentations cannot solve the occlusion issue in dataset. Therefore, cutout augmentation is introduced by adding rectangle at random location in the datasets to let the network learn the feature of that particular object. However, by adding a block of black pixels into the image can also mean that some of the information is loss which is inefficient during training. Therefore, the author proposed CutMix Augmentation **[4]** that simply replaced the block of black pixels with another random image from training dataset. Below shows an example of different augmentation. From the left (original, Mixup, Cutout, CutMix).

CutMix combine the idea of Mixup and Cutout, where the occlusion in replaced by another object in the training datasets. This method can also learn the partial feature of both objects which enhance the localization ability. Therefore, the solution to combining random two images are shown below.

In order to combine two training images together, the author define a combination operation shown in equation 8. The generated training sample can be represented as $\tilde{x}$ and $\tilde{y}$.

$$\tilde{x} = \mathbf{M} \odot x_A + (\mathbf{1} - \mathbf{M}) \odot x_B$$
$$\tilde{y} = \lambda y_A + (1 - \lambda) y_B, \qquad (8)$$

M represent binary mask and $\odot$ denotes element-wise multiplication. λ is the ratio between the two points in beta distribution $\beta(\alpha, \alpha)$. According to the author, the ideal α is 1. x_A is the portion in the image that is removed and x_B is the portion in another image that is cropped and pasted into the location of x_A. In every iteration, a new ($\tilde{x}$, $\tilde{y}$) image will be randomly generated in the mini-batch. Therefore, the network will learn different aspect and configuration of the object.

For my experiment, I will be incorporating Cutmix Augmentation into Resnet-18 (training section) and experimenting with α = 1 and α = 0.5. The other hyperparameters can be found in Annex A.

	Cutmix Augmentation	
	α = 0.5	α = 1
Training loss	0.3638	0.6408
Testing loss	0.1604	0.1605
Training accuracy	89.53	82.95
Testing accuracy	95.82	95.59

Table 5: Comparing with different alpha value in Cutmix augmentation

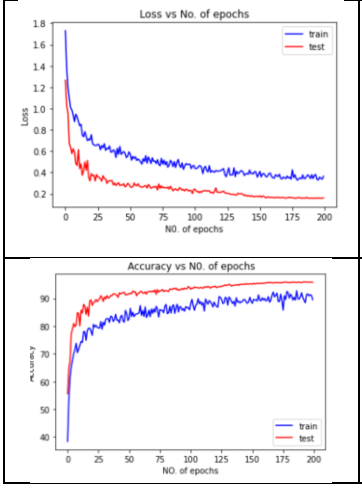
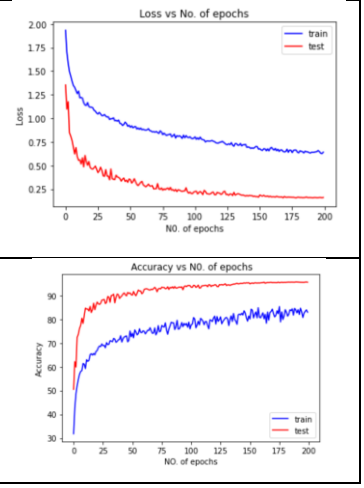
α = 0.5	α = 1
	

Table 6: Diagram of different alpha value in Cutmix augmentation

DropBlock

Dropout is one of the traditional regularizations that is added into layers, specifically the fully connected layers, in neural network. In dropout, there is a hyper parameter that determine the probability of the layers to be dropout and it is normally used during training. However, dropout has some cons. According to the author, even with applying dropout into convolutional layers, the activation units still contained some correlated information which will be circulate around the network. Thus, the author proposed DropBlock **[5]** where surrounding unit of the feature map is removed together.

The difference between DropBlock and Dropout is that DropBlock removed the entire surrounding regions from the feature map in a layer instead of random removing a unit from the layers. DropBlock has two parameters, the block size and the γ. Block size is the number of blocks to be removed and γ refer to the number of activation units to be removed. In order to find the value of γ, we need to input the probability, *keep_prob,* of the layer to be removed. The formula of γ can be defined as

$$\gamma = \frac{1 - keep\_prob}{block\_size^2} \frac{feat\_size^2}{(feat\_size - block\_size + 1)^2} \tag{9}$$

In Dropout, the parameter, *keep_prob,* is constant throughout the training. However, in DropBlock, constant *keep_prob* is not good in training. Therefore, the author proposed a linear scheduled DropBlock which will gradually decrease from 1 to value of *keep_prob*. The result show improvement in image classification datasets such as ImageNet.

For my experiment, I will be incorporating DropBlock into first and second layers in ResNet-18 with parameters of **a) (0.9 keep_prob and 7 blocks)** and **b) (0.7 keep_prob with 7 blocks**). The other hyperparameters will be stated in Annex A.

	DropBlock	
	(a)	(b)
Training loss	0.0577	0.0383
Testing loss	0.2373	0.2373
Training accuracy	98.22	98.80
Testing accuracy	93.22	93.59

Table 7: Comparing with different configuration

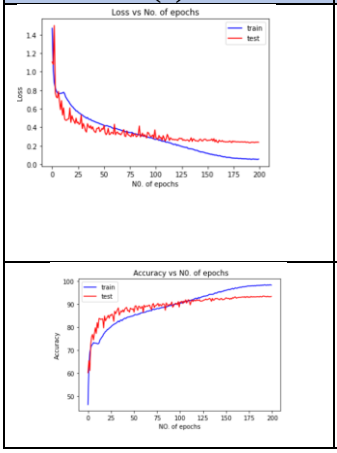
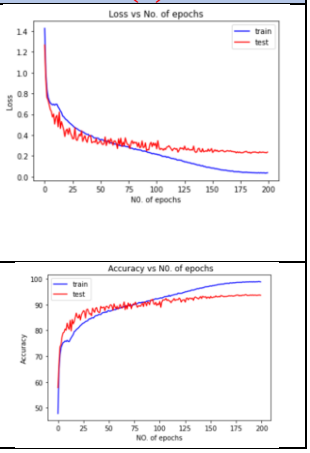
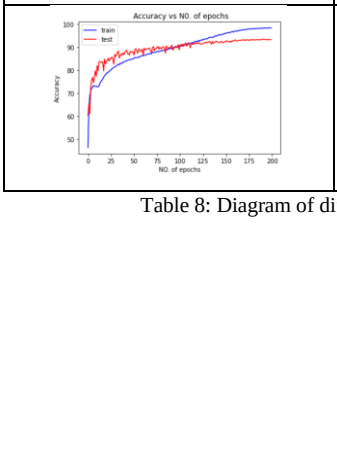
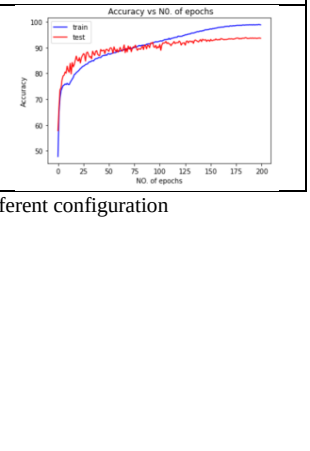
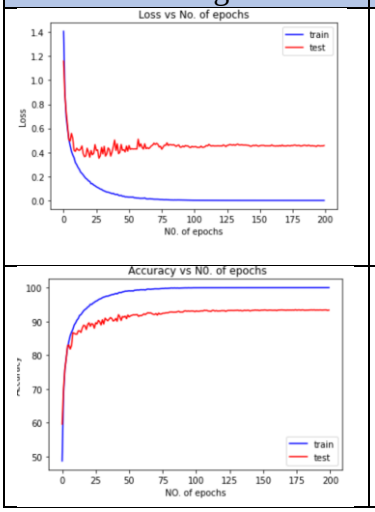
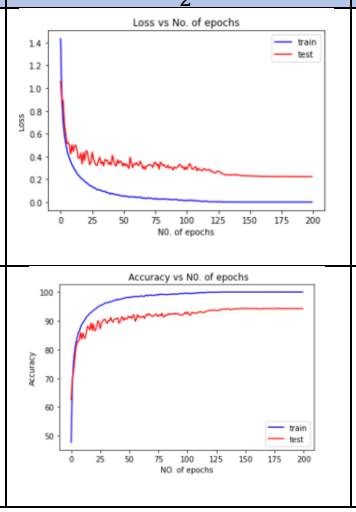
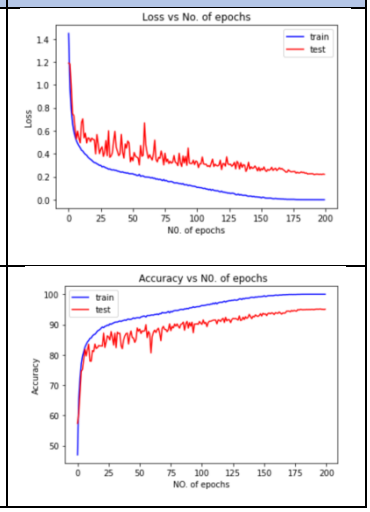
(a)	(b)
	
	

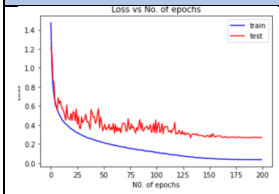
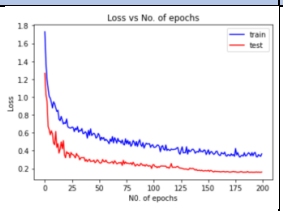
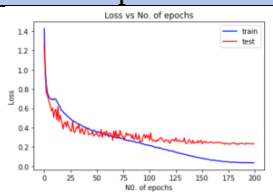
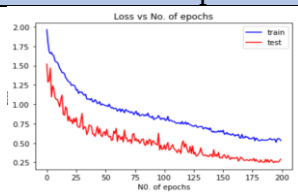
Table 8: Diagram of different configuration

Result

	Different Regularizers						
	No reg	L_2	SWD	Jacobian	CutMix	DropBlock	CutMix + DropBlock
Training loss	0.0002	0.0011	0.0006	0.0366	0.3638	0.0383	0.5316
Testing loss	0.4572	0.2242	0.2218	0.268	0.1604	0.2373	0.2897
Training accuracy	100	100	99.99	99.99	89.53	98.80	81.23
Testing accuracy	93.37	94.21	95.05	91.54	95.82	93.59	90.99

Table 9: Comparing with different regularizers

No reg	L_2	SWD
		

Jacobian	CutMix	DropBlock	CutMix + DropBlock
			

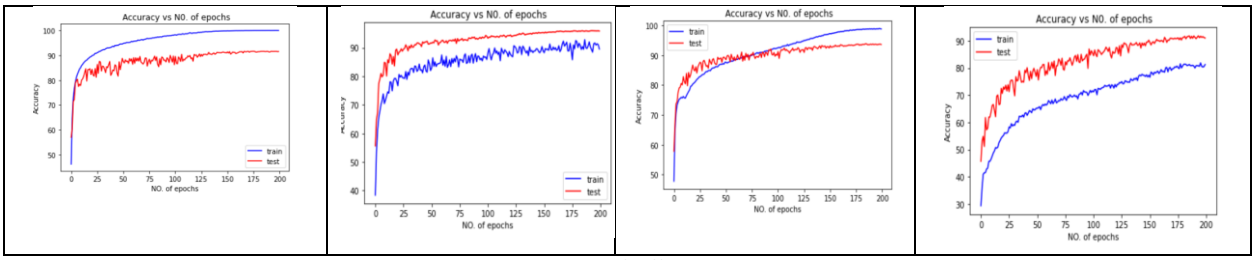


Table 10: Diagram of different regularizers

Conclusion

After experimenting different regularizations, cutmix seemed to be the best among the rest. Even though the training loss and accuracy is not the optimal, the testing loss and accuracy outperform the rest. The reason for that bad training accuracy and loss is understandable because the given datasets is much harder for the model to learn. It uses cutout to block several features of the object and paste with another partially cropped feature from another image. This is very challenging not only for neural network but also for us human. Unfortunately, the article recommends at least 300 epochs for the cutmix augmentation to be optimized. I did not train 300 epochs because I want a fair comparison with the rest. Ultimately, there is still other combination of regularization to be experiment such as cutout with dropblock, cutmix with SGDS.

Annex A

This is the default setting for training any regularization, otherwise stated.

Hyperparameters:

Datasets – Cifar 10

Number of epochs - 200

Learning rate – 0.01

Learning rate scheduler – Cosine Annealing scheduler

Optimizer – SGD with 0.9 momentum

Weight Decay – 0.0005

Reference

[1] Zeke Xie, Issei Sato, Masashi Sugiyama. UNDERSTANDING AND SCHEDULING WEIGHT DECA. 2021

[2] Zeke Xie, Issei Sato. A DIFFUSION THEORY FOR DEEP LEARNING DYNAM- ICS: STOCHASTIC GRADIENT DESCENT EXPONEN- TIALLY FAVORS FLAT MINIMA. 2021

[3] Judy Hoffman, Daniel A. Roberts. Robust Learning with Jacobian Regularization. 2019

[4] Sangdoo Yun, Dongyoon Han, Seong Joon Oh. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. 2019

[5] Golnaz Ghiasi, Tsung-Yi Lin, Quoc V. Le. DropBlock: A regularization method for convolutional networks. 2018