MEML: a deep data augmentation method by mean extrapolation in middle layers

DONGCHEN LIU*, LUNZHANG*, XIANSEN JIANG2, CAIXIA SU3, YUFENG FAN4, AND YONGFENG CAO#

1Guizhou Normal University, Guiyang City, Guizhou Province, China
2Zunyi Sports School, China
3Guizhou Normal University, Guiyang City, Guizhou Province, China
Corresponding author: yongfeng cao (e-mail: cyfcs@whu.edu.cn).
DongChen Liu and lun zhang contributed equally to this work

This work is supported by The Science and Technology Foundation of Gui Zhou Province([2018]1114)

ABSTRACT Data augmentation, generating new data that are similar but not same with the original data by making a series of transformations to the original data, is one of the mainstream methods to alleviate the problem of insufficient data. Instead of augmenting input data, this paper proposes a method for augmenting features in the middle layers of deep models, called MEML (Mean Extrapolation in Middle Layers). It gets the features outputted by any middle layer of deep models and then creates new features by extrapolating some randomly selected features with their corresponding class mean. After that, it replaces these selected features with the new ones, and then let the new composed output continue to propagate forward. Experiments on two classic deep neural network models and three image datasets show that our MEML method can significantly improve the model classification accuracy and outperform the state-of-the-art feature space augmentation methods such as dropout and K Nearest Neighbors extrapolation in most experiments. Interestingly, when coupled with some input space augmentation methods, e.g., rotation and horizontal flip, MEML could further improve the performance of deep models, implying that the input space augmentation methods and MEML could complement each other.

INDEX TERMS data augmentation; extrapolation; feature space;

I. INTRODUCTION

In recent years, deep learning has made great breakthroughs in many fields (e.g. images, speech, text). The premise of these progress is a large amount of data which can help models to optimize towards the optimal solution, thereby improve the performance of the models. However, in many fields or tasks, it is often difficult to obtain sufficient data due to privacy protection and the high cost of labeling. Data augmentation, generating new data that are similar but not same with the original data by making a series of transformations to the original data, is one of the mainstream methods to alleviate the problem of insufficient data. Many researches have used data augmentation to improve model performance. For example, Simard et al. [1] used Elastic distortion method to increase the sample number of MNIST dataset. Horizontal reflection and color transformation were used by Krizhevsky et al. [2] to reduce model overfitting when training the renowned AlexNet model. In addition, GAN-based data augmentation methods [3]–[6] and noise injection [7] are also often used by researchers. Most of these above methods do augmentation in the input space and are not universal. For example, horizontal and vertical reflection are effective methods for CIFAR-10 dataset, but not suitable for MINIST dataset. Excessive color transformation may cause the data to lose its original semantic information, and it is not suitable for tasks with color as the basic feature. How to find the suitable transformations for a dataset requires professional participation, providing a challenge for developing generalizable augmentation methods. In order to solve this problem, researchers from Google Brain proposed AutoAugment [8] and RandAugment [9] to automatically search for suitable data augmentation strategies.

Yoshua et al. [10] and Ozair & Bengio [11] claimed that the higher-level samples fill more uniformly the space they...
occupy and the high-density manifolds tend to unfold when represented at higher levels, which means that extrapolating between high-level samples is easier to stay in the manifold. So, instead of doing augmentation in the input space, in this paper, we present a method for augmenting features in the middle layers of deep models, called MEML (Mean Extrapolation in Middle Layers). We increase the amount of data by extrapolating the features outputted by middle layers of deep neural networks (DNNs), thereby improving the performance of DNNs.

Our MEML method offers several advantages:

1) Our method can be applied on any layer of DNNs as a separate module, which greatly increase the flexibility of our method (see Section III).
2) The augmented samples are generated in the process of forward propagation by simple calculations, and don’t need extra disk space to store them (see Section III).
3) Our method can be combined with input space augmentation methods to further improve the model performance (see Section IV.D).

We evaluated our method on three public image datasets: CIFAR-10, CIFAR-100, Tiny-ImageNet-200 and two classic image classification models: VGG16 [12], RESNET18 [13], demonstrating that it can significantly improve the classification accuracy of deep models and is suitable for different datasets and different models. We also compared it with dropout [14] and K Nearest Neighbors (KNN) extrapolation [15], which are the most relevant to our method.

II. RELATED WORK
A. INPUT SPACE AUGMENTATION

Most of the early data augmentation methods do augmentation in the input space, the difference of them lies in the way the augmented samples are generated. Geometric transformations, such as shifting, rotation, scaling, are the most basic and widely used ways. LeCun et al. [16] performed a series of affine transformations to training samples to improve the performance of LeNet-5 model, which is one of the most early and well-known convolutional neural networks. Krizhevsky et al. [2] used horizontal reflection and color transformation to reduce the overfitting of AlexNet model. Simonyan et al. [12] also used geometric transformations to generate new data when training the classic VGGNET model for the 2014 Large Scale Visual Recognition Challenge (ILSVRC).

Zhong et al. [17] proposed random erasing for data augmentation. Random erasing generates a new sample by randomly covering an n × m patch of an image and masking it with either 0 s, 255 s, or random values. Our MEML method generates new samples by adding a disturbance rather than discarding part of the information. Inoue et al. [18] mixed images by calculating the average of pixels of two images, which can be regarded as interpolation in the input space. We use mean extrapolation, which is very similar to interpolation, to generate new samples, but we extrapolate in the feature space.

Shijie et al. [19] explored the impact of various input space augmentation methods on image classification tasks with DNN models. They found that some appropriate combination methods are slightly more effective than the individuals. In this paper, we combine MEML (feature space augmentation method) and input space augmentation methods, and prove that this combined method is more effective than a single method, which extends the conclusion of [19].

B. FEATURE SPACE AUGMENTATION

With the development of deep neural networks, researchers have proposed a series of feature space augmentation methods. Among them, interpolation [20], extrapolation [15], dropout [14], and feature combination [22] are the most relevant to our method.

Chawla et al. [20] first used interpolation in the feature space to solve the problem of data imbalance. After that, many researches ([15], [21]) further researched and developed the interpolation method. DeVries et al. [15] used a auto-encoder model to project input data into feature space and then applied three data augmentation methods (KNN interpolation, KNN extrapolation, and adding Gaussian noise) to get the augmented samples. Among these three methods, the KNN extrapolation worked best. Wong et al. [21] evaluated the effect of two data augmentation methods, elastic deformations (input space augmentation) and SMOTE [20] (feature space augmentation), in improving the performance of classifiers. They found that the input space augmentation works better than feature space augmentation, when suitable input space transformations are used.

Dropout [14] lets some neurons stop working with a certain probability when data stream propagates forward. During this process, the outputs of these neurons are changed, which is equivalent to generating new data. So, dropout [14] can be regarded as a feature space augmentation method. Chu et al. [22] presented a novel approach to address the long-tailed problem. They combined the class-generic features in ample classes with class-specific features in tail classes to increase the number of tail classes samples. Their augmentation method can be applied on any middle layer of models and generate augmented samples online.

Our MEML method could be seen as an improvement of the KNN extrapolation method (when K=1). We try to generate better new samples by using more statistical information of a mini-batch. At the same time, inspired by [22] and [14], we design a module to implement the augmentation process and insert the module into deep models. Specifically, it is to complete the augmentation in the process of forward propagation instead of separating the augmentation process from the training process like [15].

III. METHOD

The proposed data augmentation method, MEML, is based on mini-batch gradient descent algorithm of training deep neural networks.
The MEML method could be applied on any middle layer of DNNs. As shown in Fig. 1, a MEML module is inserted after the t-th layer of the DNN to generate augmented samples. During the model training process, when each mini-batch samples flow through the module, it will do following: firstly, it gets the feature vectors outputted by the previous layer and calculates the class mean for each class; secondly, it randomly selects some feature vectors according to the augmentation ratio $\beta \in [0,1]$; thirdly, it generates new feature vectors by extrapolating these selected feature vectors with their corresponding class means; fourthly, it uses the newly generated feature vectors to replace their corresponding old ones, and then sends the updated feature vectors to the next layer to continue forward propagation. The pseudo code of MEML method is illustrated in Algorithm 1.

Unlike [15], our synthetic feature vectors are generated in the process of forward propagation rather than in advance and will be discarded when the back propagation ends, so extra disk space are not needed to store them.

### A. MEAN EXTRAPOLATION

The formula of mean extrapolation is as follows:

$$x' = x + r \lambda (x - M_x)$$  \hspace{1cm} (1)

where $x$ refers to the feature vector of a input sample, $M_x$ refers to the mean of feature vectors that have the same label with $x$, $x'$ is the synthetic feature vector, scalar $\lambda \in (0, \infty)$ controls the degree of extrapolation, scalar $r \sim Bernoulli(\beta)$ determines whether to do augmentation, $\beta \in [0,1]$ is the augmentation ratio. The geometric meaning of mean extrapolation is shown in Fig. 2.

The extrapolation degree parameter $\lambda$ plays a crucial role in our MEML method. If $\lambda$ is set too small, the synthetic feature vectors will be too similar to the original feature vectors, causing it difficult to bring new information to improve model performance. On the other hand, if $\lambda$ is set too big, the synthetic feature vectors will be easily far away from the true data distribution, which will cause a negative effect to the model.

The output feature vectors of MEML module could also be consider as the input feature vectors add a disturbance $\Delta x$. In this sense, the (1) is transformed into (2):

$$x' = x + \Delta x = x + r \lambda (x - M_x)$$  \hspace{1cm} (2)
can be expressed by neurons stop working with a certain probability. This process changes the outputs of DNN layers by making controls the extrapolation degree.

**METHODS**

### B. DIFFERENCE AND CONNECTIONS WITH OTHER METHODS

1) Dropout

Dropout changes the outputs of DNN layers by making neurons stop working with a certain probability. This process can be expressed by

\[ x' = x + \Delta x = x - r \odot x \quad (3) \]

where \( \odot \) is Hadamard product, \( x \) and \( x' \) refer to the output of DNN layers before and after doing dropout, respectively, \( r \) is a random vector with each element \( r_i \in \{0, 1\} \) sampled from Bernoulli Distribution.

By comparing (2) and (3), it could be seen that MEML and dropout are essentially in the same line of adding disturbance to the outputs of DNN layers. The difference lies in the disturbance \( \Delta x \). Dropout makes neurons stop working with a certain probability, so the element \( (\Delta x)_i \) of \( \Delta x \) only has two states: 0 or \( -x_i \); However, MEML generates \( \Delta x \) with \( r \lambda (x - M_x) \), so the element \( (\Delta x)_i \) has two states: 0 or \( \lambda (x_i - (M_x)_i) \). MEML exploits batch based statistics information \( M_x \) and a hyper-parameter \( \lambda \) to create a more meaningful disturbance \( \Delta x \), thereby generating a better feature vector.

2) KNN extrapolation

MEML is very similar with the KNN extrapolation method proposed by DeVries et al. [15]. When \( K=1 \), the KNN extrapolation formula could be expressed as

\[ x' = x + \Delta x = x + r \lambda (x - x_{1NN}) \quad (4) \]

where \( x_{1NN} \) is the feature vector that is the nearest to the feature vector \( x \) and has the same class label with it in a mini-batch. As shown in Fig. 3, the only difference between 1NN extrapolation and MEML is using \( x_{1NN} \) or \( M_x \) to generate disturbance.

Experimental results show that compared with 1NN extrapolation, MEML is easier to generate good synthetic feature vectors. In our opinion, there are two possible reasons for it: (1) compared with a single feature vector \( x_{1NN} \), the mean of feature vectors, \( M_x \), contains more statistical information, so \( M_x \) is more representative of the class it belong to, and so the extrapolation direction from \( M_x \) to \( x \) is more likely to be meaningful than from \( x_{1NN} \) to \( x \); (2) since the mean is mostly in the center of all feature vectors in the same class, the difference between a single feature vector and \( M_x \) does not change much. So, it is easier to find an extrapolation degree that is suitable for most situations.

**IV. EXPERIMENTS AND ANALYSIS**

### A. COMMON SETTINGS

The proposed method was tested on three public image datasets: CIFAR-10, CIFAR-100, and Tiny-ImageNet-200. VGG16 and RESNET18 were used here because they are classic models for image classification and their depths are suitable for our task and hardware facilities.

In all experiments, the classification model begins with an empty VGG16 or RESNET18 model. The results obtained from normal model training method without data augmentation is called baseline. For ease of analysis, in each experiment, only one middle layer of the above two models is selected and after which, a MEML module is inserted for data augmentation. The chosen layers are the second, tenth, thirteenth convolution layers of the VGG16 model and the first, thirteenth, seventeenth convolution layers of the RESNET18 model. The Cross-entropy function is chosen as the loss function, and the learning rate is set to 0.001. The architecture of the two models and the hyper-parameters used in our experiments are reported in Appendix.

For alleviating the bias of performance (classification accuracy) caused by random factors, the average metrics of three independent runs will be employed.

### B. COMPARISON ON DIFFERENT LAYERS

This section’s experiments are set up to explore the effectiveness of MEML on different middle layers and the sensitivity of MEML to extrapolation degree parameter \( \lambda \) on the basis of a fixed augmentation ratio of 0.4. The results are shown in Fig. 4 (VGG16 model) and Fig. 5 (RESNET18 model), respectively.

According to Fig. 4 and Fig. 5, the performance is very different when MEML is applied on different layers. Specifically, for VGG16 model, performance is better on the higher...
layers (tenth and thirteenth layers) of the model than on the lower layer (second layer) in almost all extrapolation degree. For RESNET18 model, the same rule could also be found on CIFAR-10 and Tiny-ImageNet-200 datasets, except on CIFAR-100 dataset. This is related to the characteristics of features extracted from different layers. Zeiler et al. [23] claimed that the neural network extracts features layer by layer in the process of forward propagation, and the features extracted from higher layers are more abstract and complex than these from lower layers. In some sense, extrapolating between high-level abstract features of higher layers is easier to generate real useful features than extrapolating between low-level features of lower layers. In addition, on every dataset, the best performance improvement is always got by applying MEML on the middle and later layers of the model (the tenth layer of VGG16 model and the thirteenth layer of RESNET18 model). All these findings suggest that when inserting a MEML module for data augmentation, the middle and later layers of DNN models should have the priority to be considered.

Another worth noting thing is the sensitivity of MEML to extrapolation degree. It can be seen from Fig. 4 and Fig. 5 that this sensitivity varies for different inserting layers, different models and different datasets. But there is a rough rule could be found that the performance of MEML first increases and then goes down with the increasing of the extrapolation degree, suggesting to try the extrapolation degree parameter from small to large, when applying our MEML method.

C. COMPARISON WITH DROPOUT AND 1NN EXTRAPOLATION

This section’s experiments are set up to compare MEML with dropout [14] and the KNN extrapolation [15]. For the sake of fairness, parameter K are set to 1.

Specifically, for 1NN extrapolation and MEML, we run them with different extrapolation degrees and a fixed augmentation ratio of 0.4. For dropout, we run it with a fixed extrapolation degree of 1 and different augmentation ratio (it is called dropout rate here).

The accuracy-extrapolation degree/dropout rate curves are shown in Fig. 6-Fig. 11. Table 1 and Table 2 show the highest accuracy got by these three methods on different middle layers of classification model.

1) VGG16 model

Roughly speaking, for all three datasets, MEML gets higher accuracy than 1NN extrapolation and dropout, whatever the extrapolation degree value is when applied on the tenth layer, and in most of the extrapolation degree value when applied on the thirteenth layer, but not always so when they are applied on the second layer.

When the extrapolation degree is very small, the effect of MEML and the 1NN extrapolation is nearly the same. But as the extrapolation degree increases, our MEML method outperforms the 1NN extrapolation, especially on the tenth and thirteenth layers. The reason is that our MEML method is easier to generate good augmented samples than 1NN extrapolation as explained in Section III.B
MEML is worse than Dropout on the second layer for CIFAR-10 and CIFAR-100 datasets, but is obviously better than Dropout on the tenth and thirteenth layers for all three datasets.

2) RESNET18 model

For RENSRT18 model, the situation becomes a little different, MEML no longer has an absolute advantage. It can be seen from Fig. 9 that the best performance of CIFAR-10 dataset is got by dropout on the thirteenth layer.

When tested on CIFAR-100 dataset, all three methods are worse than the baseline on the first and seventeenth layers, and MEML is worse than 1NN extrapolation in the seventeenth layer. But MEML achieves the best result for the CIFAR-100 dataset on the thirteenth layer.

According to Fig. 11, MEML achieves the best result for Tiny-ImageNet-200 dataset on the thirteenth layer and outperforms the 1NN extrapolation method on all three layers.

Overall, in a total of six experiments (two models coupled with three datasets), MEML achieves the highest accuracy in five of them, dropout achieves once. The best results are all achieved on the tenth layer of the VGG16 model or the thirteenth layer of the RESNET18 model.

D. COMPARISON WITH INPUT SPACE AUGMENTATION METHODS

This section’s experiments are set up to compare MEML with input space augmentation methods.
Four different methods are conducted on the CIFAR-100 dataset here, including baseline (without dropout and data augmentation), input space augmentation (random rotation and horizontal flip), feature space augmentation (MEML), and input space augmentation plus MEML. For input space augmentation, the angle range of rotation is set to (-15°, 15°), the probability of horizontal flip is 0.5. The input space augmentation plus MEML method is tested with different extrapolation degrees and a fixed augmentation ratio of 0.4 as we do in Section IV.B.

The results (only the highest accuracy of each method) are shown in Table 3 and Table 4.

It could be found that the input space augmentation methods are better than MEML on both two models, which verifies the conclusion of [21]. This may be due to the unexplainable nature of neural networks. When augmenting input data, using human prior knowledge, we can judge whether these operations change the semantic information of the data. However, when augmenting features, it is difficult to determine due to the uninterpretability of neural networks.

Interestingly, the input space augmentation plus MEML method further improves the accuracy on the tenth, thirteenth layers of VGG16 model and all three layers of RESNET18 model. This implies that the input space augmentation methods and MEML could complement each other, which greatly increases the usability of our method.
TABLE 1. The highest accuracy of different methods on VGG16 model

| dataset       | layer   | dropout            | 1NNEX             | MEML               | baseline |
|---------------|---------|--------------------|-------------------|--------------------|----------|
| CIFAR-10      | second  | 0.8798 ± 0.0111    | 0.8786 ± 0.0014   | 0.8783 ± 0.0006    | 0.877 ± 0.0018 |
|               | Tenth   | 0.8811 ± 0.0018    | 0.8832 ± 0.0014   | 0.8881 ± 0.0012    |          |
|               | Thirteenth | 0.878 ± 0.0019    | 0.880 ± 0.0006    | 0.8807 ± 0.001     |          |
| CIFAR-100     | second  | 0.5887 ± 0.0068    | 0.5793 ± 0.0015   | 0.5769 ± 0.0015    | 0.5809 ± 0.0026 |
|               | tenth   | 0.5887 ± 0.0039    | 0.6008 ± 0.0029   | 0.6137 ± 0.0012    |          |
|               | thirteenth | 0.5865 ± 0.0022  | 0.603 ± 0.0026    | 0.6083 ± 0.0031    |          |
| Tiny-ImageNet-200 | second  | 0.4754 ± 0.0042    | 0.4753 ± 0.0038   | 0.4747 ± 0.0024    | 0.4753 ± 0.002 |
|               | tenth   | 0.4877 ± 0.0012    | 0.4822 ± 0.0021   | 0.4927 ± 0.0081    |          |
|               | thirteenth | 0.4724 ± 0.003    | 0.4787 ± 0.0021   | 0.4905 ± 0.0019    |          |

TABLE 2. The highest accuracy of different methods on RESNET18 model

| dataset       | layer   | dropout            | 1NNEX             | MEML               | baseline |
|---------------|---------|--------------------|-------------------|--------------------|----------|
| CIFAR-10      | First   | 0.8644 ± 0.0016    | 0.8662 ± 0.0011   | 0.8643 ± 0.0015    | 0.8635 ± 0.001 |
|               | Thirteenth | 0.8786 ± 0.0017  | 0.8665 ± 0.0024   | 0.875 ± 0.001     |          |
|               | Seventeenth | 0.8665 ± 0.0018  | 0.8685 ± 0.0012   | 0.8711 ± 0.0003    |          |
| CIFAR-100     | First   | 0.607 ± 0.0027     | 0.6082 ± 0.0024   | 0.607 ± 0.001     | 0.6058 ± 0.0036 |
|               | Thirteenth | 0.6123 ± 0.0028  | 0.603 ± 0.0055    | 0.6331 ± 0.0022    |          |
|               | Seventeenth | 0.6039 ± 0.0049  | 0.5994 ± 0.0014   | 0.5889 ± 0.0104    |          |
| Tiny-ImageNet-200 | First   | 0.4947 ± 0.003    | 0.4847 ± 0.0032   | 0.4884 ± 0.0012    | 0.4821 ± 0.0006 |
|               | Thirteenth | 0.4987 ± 0.0035  | 0.4803 ± 0.0012   | 0.5001 ± 0.0022    |          |
|               | Seventeenth | 0.4768 ± 0.0009  | 0.4841 ± 0.0012   | 0.4968 ± 0.0023    |          |

TABLE 3. The highest accuracy of VGG16 model on CIFAR-100 dataset

| layer               | baseline | input layer augmentation | MEML | input layer augmentation +MEML |
|---------------------|----------|--------------------------|------|-------------------------------|
| second layer        | 0.5809 ± 0.0026 | -                         | 0.5769 ± 0.00015 | 0.6263 ± 0.0013         |
| tenth layer         | -        | -                        | 0.6286 ± 0.0017  | 0.6374 ± 0.0008         |
| thirteenth layer    | 0.6083 ± 0.0031 | -                         | 0.6083 ± 0.0031  | 0.646 ± 0.0019          |

TABLE 4. The highest accuracy of RESNET18 model on CIFAR-100 dataset

| layer               | baseline | input layer augmentation | MEML | input layer augmentation +MEML |
|---------------------|----------|--------------------------|------|-------------------------------|
| first layer         | 0.607 ± 0.001   | -                        | 0.607 ± 0.001   | 0.6464 ± 0.0026         |
| thirteenth layer    | 0.6331 ± 0.0022 | -                        | 0.6331 ± 0.0022 | 0.6428 ± 0.0023         |
| seventeenth layer   | 0.5889 ± 0.0104 | -                        | 0.5889 ± 0.0104 | 0.6505 ± 0.0003         |

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a new feature space data augmentation method for deep neural networks, called MEML. We insert a MEML module in DNNs to extrapolate features. In theory and practice, the MEML module could be easily inserted after any layer of DNNs just like those classic
modules, such as convolution, pooling, dropout, etc.

The performance of MEML was tested on three image datasets and two classic classification DNN models. The results show that our MEML method works better on the higher DNN layers (in our experiments, it is the tenth, thirteenth layers of the VGG16 model and the thirteenth, seventeenth layers of the RESNET18 model) than on the lower DNN layers, implying that the higher-level abstract features extracted by higher layers are more suitable for feature space augmentation.

As a feature space augmentation method, MEML outperforms the state-of-the-art feature space augmentation methods, such as dropout [14] and KNN extrapolation [15], in most experiments in this paper. Interestingly, when coupled with some input space augmentation methods (rotation and horizontal flip), MEML could further improve the performance of DNN models, implying that the input space augmentation methods and MEML could complement each other.

Due to the limitation of computing resources and for ease of analysis, this paper only considers the situation that only one MEML module is inserted. It is conceivable that multiple MEML modules would work better than one.

The extrapolation degree is a crucial parameter to MEML. However, how to find an optimal value is yet unanswered, which deserves further research. In addition, although MEML has been verified on image datasets and achieved good results, its cross-domain performance needs to be verified on other types of datasets (such as text, speech, etc.), which will be one of our future works.

APPENDIX

As we discussed before, we insert a MEML module in the basic VGG16 and RESNET18 models to do data augmentation. The specific insertion locations are shown in the Fig. 12 and Fig. 13. Hyper-parameters used for training the VGG16 and RESNET18 models in our experiments are summarized in Table 5 and Table 6.

| dataset       | method   | optimizer | \( \eta \) | \( b \) | \#epoch |
|---------------|----------|-----------|------------|--------|---------|
| CIFAR-10      | dropout  | sgd       | 0.001      | 64     | 200     |
|               | 1NNEX    | sgd       | 0.001      | 64     | 200     |
|               | MEML     | sgd       | 0.001      | 64     | 200     |
| CIFAR-100     | dropout  | sgd       | 0.001      | 256    | 200     |
|               | 1NNEX    | sgd       | 0.001      | 256    | 200     |
|               | MEML     | sgd       | 0.001      | 256    | 200     |
| Tiny-ImageNet-200 | dropout  | sgd       | 0.001    | 256    | 200     |
|               | 1NNEX    | sgd       | 0.001    | 256    | 200     |
|               | MEML     | sgd       | 0.001    | 256    | 200     |

| dataset       | method   | optimizer | \( \eta \) | \( b \) | \#epoch |
|---------------|----------|-----------|------------|--------|---------|
| CIFAR-10      | dropout  | adam      | 0.001      | 64     | 200     |
|               | 1NNEX    | adam      | 0.001      | 64     | 200     |
|               | MEML     | adam      | 0.001      | 64     | 200     |
| CIFAR-100     | dropout  | adam      | 0.001      | 256    | 200     |
|               | 1NNEX    | adam      | 0.001      | 256    | 200     |
|               | MEML     | adam      | 0.001      | 256    | 200     |
| Tiny-ImageNet-200 | dropout  | sgd       | 0.001    | 256    | 200     |
|               | 1NNEX    | sgd       | 0.001    | 256    | 200     |
|               | MEML     | sgd       | 0.001    | 256    | 200     |
REFERENCES

[1] P. Y. Simard, D. Steinkraus, J. C. Platt et al., “Best practices for convolutional neural networks applied to visual document analysis.” in Icdar, vol. 3, no. 2003, 2003.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in neural information processing systems, vol. 25, pp. 1097–1105, 2012.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” Advances in neural information processing systems, vol. 27, 2014.

[4] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv preprint arXiv:1511.06434, 2015.

[5] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” arXiv preprint arXiv:1710.10196, 2017.

[6] X. Wang, A. Shrivastava, and A. Gupta, “A-fast-rcnn: Hard positive generation via adversary for object detection,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2606–2615.

[7] F. J. Moreno-Barea, F. Strazzera, J. M. Jerez, D. Urda, and L. Franco, “Forward noise adjustment scheme for data augmentation,” in 2018 IEEE symposium series on computational intelligence (SSCI). IEEE, 2018, pp. 728–734.

[8] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation policies from data,” arXiv preprint arXiv:1805.09501, 2018.

[9] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “Rand augment: Practical automated data augmentation with a reduced search space,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 702–703.

[10] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai, “Better mixing via deep representations,” in International conference on machine learning, PMLR, 2013, pp. 552–560.

[11] S. Ozair and Y. Bengio, “Deep directed generative autoencoders,” arXiv preprint arXiv:1410.0630, 2014.

[12] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” The journal of machine learning research, vol. 15, no. 1, pp. 1929–1958, 2014.

[15] T. DeVries and G. W. Taylor, “Dataset augmentation in feature space,” arXiv preprint arXiv:1702.05538, 2017.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random erasing data augmentation,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, 2020, pp. 13 001–13 008.

[18] H. Inoue, “Data augmentation by pairing samples for images classification,” arXiv preprint arXiv:1801.02929, 2018.

[19] J. Shijie, W. Ping, J. Peiyi, and H. Siping, “Research on data augmentation for image classification based on convolution neural networks,” in 2017 Chinese automation congress (CAC). IEEE, 2017, pp. 4165–4170.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” Journal of artificial intelligence research, vol. 16, pp. 321–357, 2002.

[21] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, “Understanding data augmentation for classification: when to warp?” in 2016 international conference on digital image computing: techniques and applications (DICTA). IEEE, 2016, pp. 1–6.

[22] P. Chu, X. Bian, S. Liu, and H. Ling, “Feature space augmentation for long-tailed data,” in Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16. Springer, 2020, pp. 694–710.

[23] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in European conference on computer vision. Springer, 2014, pp. 818–833.