Correct Normalization Matters: Understanding the Effect of Normalization On Deep Neural Network Models For Click-Through Rate Prediction

Zhiqiang Wang, Qingyun She, PengTao Zhang, Junlin Zhang
Sina Weibo Corp
Beijing, China
{zhiqiang36,qingyun,pengtao1,junlin6}@staff.weibo.com

ABSTRACT

Normalization has become one of the most fundamental components in many deep neural networks for machine learning tasks while deep neural network has also been widely used in CTR estimation field. Among most of the proposed deep neural network models, few model utilize normalization approaches. Though some works such as Deep & Cross Network (DCN) and Neural Factorization Machine (NFM) use Batch Normalization in MLP part of the structure, there isn't work to thoroughly explore the effect of the normalization on the DNN ranking systems. In this paper, we conduct a systematic study on the effect of widely used normalization schemas by applying the various normalization approaches to both feature embedding and MLP part in DNN model. Extensive experiments are conduct on three real-world datasets and the experiment results demonstrate that the correct normalization significantly enhances model's performance. We also propose a new and effective normalization approaches based on LayerNorm named variance only LayerNorm(VO-LN) in this work. A normalization enhanced DNN model named NormDNN is also proposed based on the above-mentioned observation. As for the reason why normalization works for DNN models in CTR estimation, we find that the variance of normalization plays the main role and give an explanation in this work.

1 INTRODUCTION

Normalization has become one of the most fundamental components in many deep neural networks for machine learning tasks, especially in Computer Vision (CV) and Natural Language Processing (NLP). However, very different kinds of normalization are used in CV and NLP. For example, Batch Normalization (BatchNorm or BN) [8] is widely adopted in CV, but it leads to significant performance degradation when naively used in NLP. Instead, Layer Normalization (LayerNorm or LN) [1] is the standard normalization method utilized in NLP.

On the other side, deep neural network has also been widely used in CTR estimation field [2–7, 9–11, 17, 19–21]. Some deep learning based models have been introduced and achieved success such as wide & deep [3], DeepFM[4] and xDeepFM[9] .Most DNN ranking models use the feature embedding to represent information and shallow MLP layers to model high-order interactions in an implicit way. These two commonly used components play important roles in current state-of-the-art ranking systems.

Taking the both sides into consideration, we care about the following questions: What's the effect of various normalization methods on deep neural network models for CTR estimation? Is there one method outperforming other normalization approaches just like LayerNorm in NLP or BatchNorm in CV? What's the reason if some normalization works?

Among most of the proposed deep neural network models, few of them utilize normalization approaches. Though some works such as DCN[15] and Neural Factorization Machine (NFM) [5] use Batch-Norm in MLP part of the structure, there isn't work to thoroughly explore the effect of the normalization on the DNN ranking systems. In this paper, we conduct a systematic study on the effect of widely used normalization schemas by applying the various normalization approaches to both feature embedding and MLP part in DNN model. Experimental results show the correct normalization helps the training of DNN models and boosts the model performance with a large margin. We also simplify the LayerNorm and propose a new and effective normalization method in this work. A normalization enhanced DNN model named NormDNN is also proposed based on the above-mentioned observation. Further more, we find the variance of normalization mainly contributes to this positive effect. To the best of our knowledge, this is the first work to verify the importance of normalization on DNN ranking system through systematic study.

The contributions of our work are summarized as follows:

(1) In this work, we propose a new normalization approach based on LayerNorm: variance-only LayerNorm(VO-LN). The experimental results show that the proposed normalization method has comparable performance with layer normalization and significantly enhance DNN model's performance.

(2) We apply various normalization approaches to the feature embedding part and the MLP part of DNN model, including commonly used normalization and our proposed approach. Extensive experiments are conduct on three real-world datasets and the experiment results demonstrate that the correct normalization or normalization combination significantly enhances model's performance. As far as we know, this is the first work to apply normalization to feature embedding and prove its effectiveness.

(3) We propose NormDNN model in this paper which is a normalization enhanced DNN adopting the following normalization strategy: variance-only LayerNorm or LayerNorm for numerical feature, BatchNorm for categorical feature and variance-only LayerNorm for MLP. NormDNN achieves significantly better performance than complex model such as xDeepFM. NormDNN is more applicable in many industry applications because of its better performance and high

computation efficiency compared with many state-of-the-art complex neural network models.

(4) To prove the universal validity of normalization for neural network ranking model, we also apply several normalization approaches to DeepFM and xDeepFM model. The experiments results imply that the correct normalization also boosts these model's performances with a large margin.

(5) As for the reason why normalization works for DNN models in CTR estimation, we find that the variance of normalization plays the main role and give an explanation in this paper.

The rest of this paper is organized as follows. Section 2 introduces some related works which are relevant with our work. We introduce our proposed models in detail in Section 3. The experimental results on three real world datasets are presented and discussed in Section 4. Section 5 concludes our work in this paper.

2 RELATED WORK

2.1 Normalization

Normalization techniques have been recognized as very effective components in deep learning. Many normalization approaches have been proposed with the three most popular ones being BatchNorm[8], LayerNorm [1] and GroupNorm[16]. Batch Normalization (Batch Norm or BN)[8] normalizes the features by the mean and variance computed within a mini-batch. This has been shown by many practices to ease optimization and enable very deep networks to converge. Another example is layer normalization (Layer Norm or LN)[1] which was proposed to ease optimization of recurrent neural networks. Statistics of layer normalization are not computed across the N samples in a mini-batch but are estimated in a layer-wise manner for each sample independently. It's an easy way to extend LayerNorm to GroupNorm (GN)[16], where the normalization is performed across a partition of the features/channels with different pre-defined groups. Normalization methods have shown success in accelerating the training of deep networks. In general, BatchNorm [8] and GroupNorm [16] are widely adopted in CV and LayerNorm [1] is the standard normalization scheme used in NLP.

Another line of research on normalization is to understand why BatchNorm helps training in CV and LayerNorm helps training in NLP. For example, the original explanation was that BatchNorm reduces the so-called "Internal Covariance Shift" [8]. However, this explanation was viewed as incorrect or incomplete and the study of [12] argued that the underlying reason that BatchNorm helps training is that it results in a smoother loss landscape. Shen etc [13] perform a systematic study of NLP transformer models to understand why BatchNorm has a poor performance and find that the statistics of NLP data across the batch dimension exhibit large fluctuations throughout training which results in instability. Xu etc [18] find that the derivatives of the mean and variance in Layer-Norm used in Transformer for NLP tasks are more important than forward normalization by re-centering and re-scaling backward gradients. Furthermore, they find that the parameters of LayerNorm, including the bias and gain, increase the risk of over-fitting and do not work in most cases.

2.2 Neural Network based CTR Models

Many deep learning based CTR models have been proposed in recent years and it is the key factor for most of these neural network based models to effectively model the feature interactions or feature importance.

Factorization-Machine Supported Neural Networks (FNN)[20] is a feed-forward neural network using FM to pre-train the embedding layer. Wide & Deep Learning[3] jointly trains wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems. However, expertise feature engineering is still needed on the input to the wide part of Wide & Deep model. To alleviate manual efforts in feature engineering, DeepFM[4] replaces the wide part of Wide & Deep model with FM and shares the feature embedding between the FM and deep component. FiBiNet[7] and FAT-DeepFFM[19] dynamically learn the importance of features via the Squeeze-Excitation network (SENET) mechanism based on different backbone networks.

While most DNN ranking models process high-order feature interactions in implicit way, some works explicitly introduce high-order feature interactions by sub-network. Deep & Cross Network (DCN)[15] efficiently captures feature interactions of bounded degrees in an explicit fashion. Similarly, eXtreme Deep Factorization Machine (xDeepFM) [9] also models the low-order and high-order feature interactions in an explicit way by proposing a novel Compressed Interaction Network (CIN) part. AutoInt[14] proposes a multi-head self-attentive neural network with residual connections to explicitly model the feature interactions in the low-dimensional space.

3 OUR WORK

In this section, we first describe the proposed variance-only Layer-Norm. We conduct extensive experiments to verify the effectiveness of normalization in section 4 and the details about how to apply the normalization on feature embedding and MLP will be introduced in this section. Finally the reason why normalization works is introduced.

3.1 Variance-Only LayerNorm

First, we briefly review the formulation of LayerNorm. Let $\mathbf{x} = (x_1, x_2, ..., x_H)$ denotes the vector representation of an input of size H to normalization layers. LayerNorm re-centers and re-scales input $\mathbf{x}$ as

$$\mathbf{h} = \mathbf{g} \odot N(\mathbf{x}) + \mathbf{b}, \quad N(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\delta},$$

$$\mu = \frac{1}{H} \sum_{i=1}^{H} x_i, \quad \delta = \sqrt{\frac{1}{H} \sum_{i=1}^{H} (x_i - \mu)^2} \tag{1}$$

where $\mathbf{h}$ is the output of a LayerNorm layer. $\odot$ is a dot production operation. μ and δ are the mean and standard deviation of input. Bias $\mathbf{b}$ and gain $\mathbf{g}$ are parameters with the same dimension H.

As we all know, LayerNorm has been widely used and proved to be effective in many NLP tasks. However, Xu etc.[18] point out that the parameters of LayerNorm, including the bias and gain, increase the risk of over-fitting and do not work in most cases. Their

experiments in four NLP datasets show that a simple version of LayerNorm without the bias and gain outperforms LayerNorm. Though their conclusion is draw mainly from the NLP tasks and they primarily consider normalization on Transformer and Transformer-XL networks. We wonder whether the same conclusion can be draw in DNN ranking models and design a similar simper version LayerNorm by removing bias and gain from LN as follows:

$$\mathbf{h} = N(\mathbf{x}), \quad N(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\delta},$$

$$\mu = \frac{1}{H}\sum_{i=1}^{H}x_i, \quad \delta = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(x_i - \mu)^2} \tag{2}$$

We call this version LayerNorm simple-LayerNorm (S-LN) just as the original paper [18] named. Our experimental results show that simple-LayerNorm has comparable performance with LayerNorm, which implies the bias and gain in LayerNorm bring neither good nor bad effect to DNN models in CTR estimation field. Our conclusion is slightly different from that in NLP field because their experimental results [18] show the advantages for simple-LayerNorm over the standard LayerNorm in several NLP tasks. We deem that may come from the difference of network structure and research field.

According to our empirical observations, we find re-centering the input x in simple-LayerNorm has little effect on the performance of DNN ranking model. So we propose variance-only LayerNorm(VO-LN) by further removing the mean from simple-LayerNorm as follows:

$$h = \frac{\mathbf{x}}{\delta}, \quad \mu = \frac{1}{H}\sum_{i=1}^{H}x_i, \quad \delta = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(x_i - \mu)^2} \tag{3}$$

Though the variance-only LayerNorm seems rather simple, our experimental results demonstrate it has comparable or even better performance in several CTR datasets than standard LayerNorm.

3.2 NormDNN

Most DNN ranking models use the feature embedding to represent information and shallow MLP layers to model high-order interactions in an implicit way. These two commonly used components play important roles in current state-of-the-art ranking systems. So we have three options to apply normalization: normalization only on feature embedding, normalization only on MLP part, normalization both on feature embedding and MLP part.

We also find different parts of DNN model need different normalization method and propose the following unified normalization combination strategy: variance-only LayerNorm or LayerNorm for numerical feature, BatchNorm for categorical feature and variance-only LayerNorm for MLP. We call this normalization enhanced DNN model with this unified normalization strategy "NormDNN" in this paper. NormDNN achieves significantly better performance than complex model such as xDeepFM. We will discuss this in Section 4.6.

3.2.1 Normalization on Feature Embedding. The input data of CTR tasks usually consists of sparse and dense features and the sparse features are mostly categorical type. Such features are

encoded as one-hot vectors which often lead to excessively high-dimensional feature spaces for large vocabularies. The common solution to this problem is to introduce the embedding layer.

An embedding layer is applied upon the raw feature input to compress it to a low dimensional, dense real-value vector. The result of embedding layer is a wide concatenated vector:

$$V_{emb} = concat(e_1, e_2, ..., e_i, ..., e_f) \tag{4}$$

where f denotes the number of fields, and $e_i \in \mathbb{R}^k$ denotes the embedding of one field. Although the feature lengths of instances can be various, their embedding are of the same length $f \times k$, where k is the dimension of field embedding.

As we all know, features in CTR tasks usually can be segregated into categorical features and numerical features. There are two widely used approaches to convert the numerical feature into embedding. The first one is to quantize each numerical feature into discrete buckets, and the feature is then represented by the bucket ID. We can map bucket ID to an embedding vector. The second method maps the feature field into an embedding vector as follows:

$$\mathbf{v_i} = \mathbf{e_i}\mathbf{x_i} \tag{5}$$

where $\mathbf{e}_i$ is an embedding vector for field i, and $\mathbf{x}_i$ is a scalar value. In our experiments, we adopt the second approach to convert numerical features into embedding.

We apply normalization on feature embedding based on the feature field as follows:

$$N(V_{emb}) = concat(N(e_1), N(e_2), ..., N(e_i), ..., N(e_f)) \tag{6}$$

where N can be BatchNorm, LayerNorm, GroupNorm, Simple-LayerNorm or variance-only LayerNorm. The bias and gain are shared for features in same feature field if the normalization contains these parameters.

For the LayerNorm based normalization approaches (LayerNorm, Simple-LayerNorm and variance-only LayerNorm), we regard each feature's embedding as a layer to compute the mean and variance of normalization. As for the GroupNorm, the feature embedding is divided into several groups to compute mean and variance. BatchNorm computes the statistics within a mini-batch.

In real life applications, the CTR tasks usually contain both categorical features and numerical features. We find the different kinds of feature need corresponding normalization method and we will discuss this in detail in Section 4.2.

3.2.2 Normalization on MLP Part. As for the feed-forward layer in DNN model, the normalization on MLP is just as usual method does. That is to say, BatchNorm's mean and variance are computed within a mini-batch and LayerNorm based normalizations's statistics are estimated in a layer-wise manner. As for the GroupNorm, we can divide the neural units contained in MLP into several groups and the statistics are estimated in group-wise manner.

Notice that we have two places to put normalization operation on the MLP: one place is before non-linear operation and another place is after non-linear operation. For clarity of the description, we use LayerNorm as an example. If we put normalization after
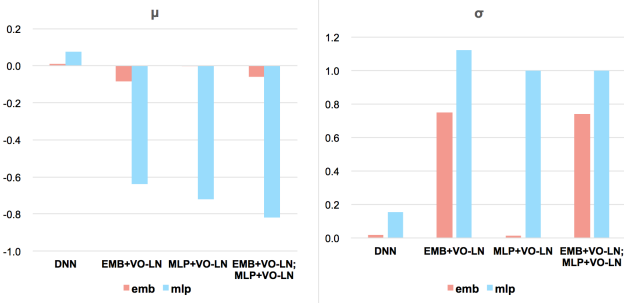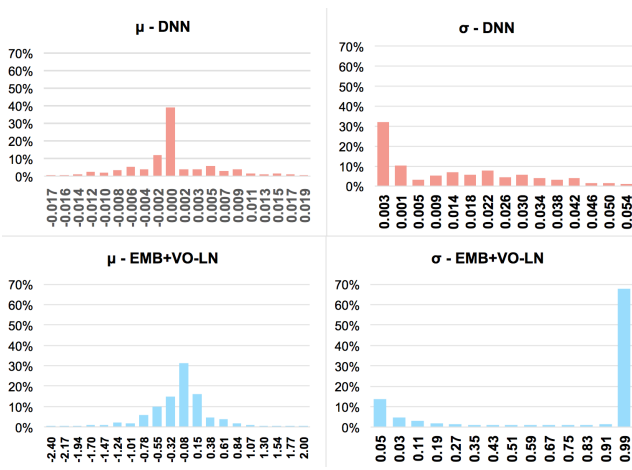
Figure 1: Average Statistics of Different Models

Figure 2: Statistics Changes after VO-LN on Feature Embedding

non-linear operation, we have:

$$LN(V_{hidden}) = LN(ReLU(W_i\mathbf{x})) \tag{7}$$

where $\mathbf{x} \in \mathbb{R}^t$ refers to the input of feed-forward layer, $W_i \in \mathbb{R}^{m \times t}$ are parameters for the layer, t and m respectively denotes the size of input layer and neural number of feed-forward layer.

If we put normalization before non-linear operation, we have:

$$LN(V_{hidden}) = ReLU(LN(W_i\mathbf{x})) \tag{8}$$

We find the performance of the normalization before non-linear consistently outperforms that of the normalization after non-linear operation. So all the normalization used in MLP part is put before non-linear operation in our paper.

3.3 Understanding Why Normalization Works

In this section, we discuss the reason why normalization works in DNN model. The related experimental results are presented in Section 4.2.

As mentioned in Section 3.1, we find three LayerNorm based models(LayerNorm, Simple-LayerNorm and variance-only Layer-Norm) have comparable performance on three real-life datasets and all the three normalization approaches work. From these observations, we can draw the following conclusion: Because the simple

LayerNorm just removes the bias and gain from LayerNorm, the similar model performance implies the bias and gain have no effect on the final model performance. Further more, the variance-only LayerNorm removes the mean from simple LayerNorm and it has comparable performance with LayerNorm and simple LayerNorm. That implies the mean in simple LayerNorm doesn't contribute to the final better performance. So we can draw the conclusion that it's variance in variance-only LayerNorm that helps boosting model's performance. The reason why LayerNorm and simple LayerNorm also work lies in that they contain variance.

To understand how the variance influences the model performance, we analyze the change of statistics both in embedding layer and MLP after applying variance-only LayerNorm on Cretio dataset (Figure. 1). From Figure 1,we can see that the average mean and variance of feature embedding and MLP are very small positive number if we don't apply normalization on any part of DNN model. If we use variance-only LayerNorm only on feature embedding, the variance of feature embedding greatly increases and that change pushes bit value of many feature embedding to a much larger negative number (Figure 1 and Figure 2). Through the network connections, these statistics changes of feature embedding are transferred to the MLP layer and the corresponding statistics of MLP show the similar trend even though we didn't apply any normalization on it (Figure 1). If we just apply variance-only LayerNorm on MLP of DNN model, we see the similar changes that the average variance of MLP neurons greatly increases and that also pushes output of many neurons to negative number (Figure 3). If we utilize variance-only LayerNorm both on feature embedding and MLP, we observe the similar trend.

As for the influence of variance on MLP, we can see that large fraction of neuron outputs is pushed into negative number from small positive number after applying variance-only LayerNorm (Figure 3). That means these neuron responses were removed because the following non-linear function is ReLU. We deem this avoid many noises in MLP responses and accelerate the training of the model because of the introduction of the variance.

As for the influence of variance on feature embedding, we can analyze the effect of normalization from another viewpoint. As we all know, the features in CTR tasks are very sparse and there are a large amount of low frequency features. This will lead to the under-fitting of these long tail feature's embedding because there is less training data for them. The under-fitting embedding may contain noise which brings difficulty for feed-forward layer to capture complex feature interactions. We think the correct normalization on feature embedding can alleviate this situation.

We can derive the derivative of $\mathcal{L}$ with respect to input x_i after inserting variance-only LayerNorm as follows. Assume the derivative of $\mathcal{L}$ with respect to h is given, ie. $\frac{\partial \mathcal{L}}{\partial h}$ is known. Then the derivative with respect to input x_i can be write as:

$$\frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial \mathcal{L}}{\partial h} * \frac{\partial h}{\partial x_i} \tag{9}$$

$$\frac{\partial h}{\partial x_i} = \frac{1}{\delta} - \frac{x_i(x_i - \mu)}{\delta^3 \cdot H} \tag{10}$$

where μ and δ are the mean and standard deviation of input. H is the size of input.
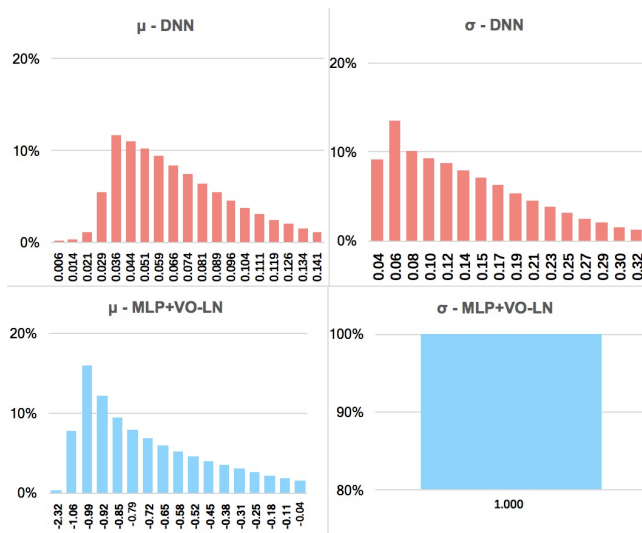
Figure 3: Statistics Changes after VO-LN on MLP Part

We can see that the second sub-item of formula (10) is nearly zero and can be ignored during the beginning phrase of the model training because parameter initiation approach usually sets the initial value of parameters to be very little random number near zero. So it's the first sub-item that mainly influences the derivative of h respect to input x_i. From figure 1, we know that the variance before normalization is rather small positive number. That is to say, the derivative will be made much larger because of the introduction of variance-only LayerNorm. This means the loss will become much more sensitive to the little change of input x_i because of the existence of variance-only LayerNorm. The variance brings faster convergence for low frequency feature embedding and alleviates the under-fitting of these feature embeddings.

4 EXPERIMENTAL RESULTS

In this section, we empirically evaluate the effect of various normalization approaches on deep neural networks on three real-world datasets and conduct detailed studies to answer the following research questions:

- **RQ1** What's the effect of various normalization approaches applied only on feature embedding part of DNN model?
- **RQ2** What's the effect of various normalizations approaches applied only on MLP part of DNN model?
- **RQ3** What's the effect of various normalization approaches applied both on feature embedding part and the MLP part of DNN model? Does our proposed variance-only LayerNorm work?
- **RQ4** Does categorical feature or numerical feature need specific normalization?
- **RQ5** Is there a best normalization combination for DNN model?
- **RQ6** Can we draw the similar conclusion about the effect of normalization in other state-of-the-art models such as DeepFM or xDeepFM?

Table 1: Statistics of the evaluation datasets

Datasets	#Instances	#fields	#features
Criteo	45M	39(26 Cat;13 Num)	30M
Malware	8.92M	82(all Cat)	9.89M
Avazu	40.43M	23(all Cat)	0.64M

In the following, we will first describe the experimental settings, followed by answering the above research questions.

4.1 Experiment Setup

4.1.1 Datasets. The following three data sets are used in our experiments:

(1) **Criteo**[1] **Dataset:** As a very famous public real world display ad dataset with each ad display information and corresponding user click feedback, Criteo data set is widely used in many CTR model evaluation. There are 26 anonymous categorical fields and 13 continuous feature fields in Criteo data set.

(2) **Malware** [2] **Dataset:** Malware is a dataset from Kaggle competitions published in the Microsoft Malware prediction. The goal of this competition is to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine. All the 82 feature fields are categorical.

(3) **Avazu**[3] **Dataset:** The Avazu dataset consists of several days of ad click- through data which is ordered chronologically. For each click data, there are 24 fields which indicate elements of a single ad impression.

We randomly split instances by 8 : 1 : 1 for training , validation and test while Table 1 lists the statistics of the evaluation datasets. For these datasets, a small improvement in prediction accuracy is regarded as practically significant because it will bring a large increase in a company's revenue if the company has a very large user base.

4.1.2 Evaluation Metric. AUC (Area Under ROC) is used in our experiments as the evaluation metrics. This metric is very popular for binary classification tasks. AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. It is insensitive to the classification threshold and the positive ratio. AUC's upper bound is 1 and larger value indicates a better performance.

4.1.3 Models for Comparisons. We mainly use DNN model as the baseline to evaluation the effect of various normalization methods because it's a commonly used component in many current neural network models. DeepFM and xDeepFM are also regarded as another two baselines to further verify the effectiveness of these approaches. Among these baseline models, DNN and DeepFM implicitly capture high order interactions while xDeepFM models high order interactions in explicit way.

---

[1]Criteo http://labs.criteo.com/downloads/download-terabyte-click-logs/
[2]Malware https://www.kaggle.com/c/microsoft-malware-prediction
[3]Avazu http://www.kaggle.com/c/avazu-ctr-prediction

Table 2: Overall performance (AUC) of DNN models with various normalization approaches on feature embedding on three datasets

	Criteo	Avazu	Malware
DNN	0.8054	0.7820	0.7263
+BN	0.8066	0.7847	0.7364
+GN2	0.8096	0.7835	0.7330
+LN	0.8093	0.7848	0.7341
+S-LN	0.8093	0.7843	0.7343
+VO-LN	0.8094	0.7839	0.7358

Table 3: Overall performance (AUC) of DNN models with various normalization approaches on MLP part on three datasets

	Criteo	Avazu	Malware
DNN	0.8054	0.7820	0.7263
+BN	0.8071	0.7836	0.7388
+GN2	0.8073	0.7836	0.7388
+LN	0.8071	0.7851	0.7378
+S-LN	0.8070	0.7847	0.7376
+VO-LN	0.8075	0.7849	0.7373

4.1.4 Implementation Details. We implement all the models with Tensorflow in our experiments. For optimization method, we use the Adam with a mini-batch size of 1000 and a learning rate is set to 0.0001. Focusing on normalization approaches in our paper, we make the dimension of field embedding for all models to be a fixed value of 10. For models with DNN part, the depth of hidden layers is set to 3, the number of neurons per layer is 400, all activation function are ReLU. We conduct our experiments with 2 Tesla $K40$ GPUs.

4.2 Effectiveness of Normalization on Feature Embedding (RQ1)

To verify the effectiveness of various normalization approaches on DNN models, comparison experiments are conduced on three evaluation datasets. We add different kinds of normalization either on the embedding layer or hidden layer of a standard DNN model which has 3 MLP layers with 400 neurons per layer. As for GourpNorm, we find GN with 2 groups perform best compared with other group number setting. So we only report the experimental results with this setting (GN2). We find normalization on feature embedding helps boosting DNN model's performance. The Table 2 shows the experimental results. From Table 2, we have the following observations:

(1) If we add normalization on feature embedding, all normalization methods help model's training on three datasets, including BatchNorm, GroupNorm ,LayerNorm, Simple LN and our proposed variance-only LN.
(2) As for some widely used normalization, BatchNorm performs best on Malware dataset while worst on Criteo dataset. That implies the performance of BatchNorm depends on specific datasets and GroupNorm shows the similar trend. LayerNorm keeps relatively high performance on all three datasets.
(3) As for LayerNorm based normalizations, LayerNorm,Simple-LN and variance-only LN show comparable performance on all three datasets.

4.3 Effectiveness of Normalization on MLP Part (RQ2)

We also conduct experiments to apply the different kinds of normalization only on MLP part of DNN model. The overall performances of DNN model with different normalizations on three evaluation

Table 4: Overall performance (AUC) of DNN models with various normalization approaches on both embedding part and MLP part on three datasets

EMB	MLP	Criteo	Avazu	Malware
w/o	w/o	0.8054	0.7820	0.7263
+BN	+BN	0.8068	0.7845	0.7393
+BN	+LN	0.8075	0.7863	0.7393
+BN	+VO-LN	0.8077	**0.7869**	**0.7402**
+LN	+BN	0.8094	0.7838	0.7387
+LN	+LN	0.8096	0.7852	0.7372
+LN	+VO-LN	**0.8098**	0.7857	0.7372
+VO-LN	+BN	0.8092	0.7823	0.7394
+VO-LN	+LN	0.8092	0.7841	0.7376
+VO-LN	+VO-LN	0.8097	0.7850	0.7383

datasets are show in the Table 3. From the experimental results, we can see that:

(1) Various normalization approaches show comparable performance on both Criteo and Malware datasets. BatchNorm and GroupNorm slightly underperform LayerNorm based approaches on Avazu dataset.
(2) Compared with normalization only on feature embedding, normalization only on MLP part performs better on Malware dataset and worse on Criteo dataset on the whole. That may imply that selection between the normalization on embedding and MLP part depends on specific task.

4.4 Normalization Combination on Both Feature Embedding and MLP (RQ3)

As discussed in Section 4.2, we can apply the normalization both on the feature embedding part and MLP part of DNN model. Extensive experiments have been conducted and we find the following three normalizations perform better when we combine various normalizations in different part of DNN model: BatchNorm, LayerNorm and Variance-only LN. So we present experimental results of 9 combinations in Table 4.

From the results in Table 4, we can see that:

Table 5: Overall performance (AUC) of DNN models with various normalization approaches for numerical and categorical features on Criteo dataset

EMB		MLP	
Num	Cat		
w/o	w/o	w/o	0.8054
+BN	+BN	+VO-LN	0.8077
+BN	+LN	+VO-LN	0.8068
+BN	+VO-LN	+VO-LN	0.8066
+LN	+LN	+VO-LN	0.8097
+LN	+BN	+VO-LN	**0.8105**
+VO-LN	+BN	+VO-LN	**0.8107**
+VO-LN	+VO-LN	+VO-LN	0.8098

(1) If we choose the correct normalization combination, the DNN model performs better than any model which only uses normalization in one part of DNN model, either feature embedding or MLP part. That means they are complementary and it's better to use them both under real-life applications.
(2) Compared with a standard DNN model, the performances of DNN model with normalizations outperform baseline with a large margin when correct normalization combination are selected.
(3) If we adopt BatchNorm in normalization combination, the conclusion that its performance depends on dataset still holds.
(4) Choosing variance-only LayerNorm in MLP part, we usually have relatively higher performance models, no matter which normalization is used in feature embedding part. It tells us that we'd better use VO-LN as normalization in MLP part when we try to combine the normalizations.

4.5 Normalization for Numerical and Categorical Feature (RQ4)

From the experimental results shown in Table 1, we observe that the performance degrades if we adopt BatchNorm instead of Layer-Norm based approaches on feature embedding on Criteo dataset. Considering only the Criteo dataset contains both categorical and numerical feature, we assume that the performance difference is related to the numerical or categorical feature. So we design some normalization combination experiments to testify this assumption. As discussed in Section 4.4, we fix the normalization used in MLP to be variance-only LayerNorm and apply different normalization for numerical and categorical feature. The experimental results can be seen in Table 5.

From the results in Table 5, we can see that:

(1) For numerical features, performance of model with Layer-Norm or variance-only LayerNorm outperforms the model with BatchNorm. That implies we should utilize LayerNorm based approaches for numerical features.
(2) If we use LayerNorm based normalization for numerical feature and variance-only LayerNorm in MLP, we can see

Table 6: Overall performance (AUC) of DNN model with unified normalization combination(NormDNN) on three datasets

	Criteo	Avazu	Malware
DNN	0.8054	0.7820	0.7263
DeepFM	0.8056	0.7833	0.7295
xDeepFM	0.8063	0.7848	0.7322
NormDNN	**0.8107**	**0.7869**	**0.7402**

Table 7: Overall performance (AUC) of popular models with various normalization approaches on Criteo dataset

EMB	MLP	DNN	DeepFM	xDeepFM
w/o	w/o	0.8054	0.8056	0.8063
+LN	w/o	0.8093	0.8100	0.8100
+VO-LN	w/o	0.8094	0.8099	0.8100
w/o	+LN	0.8071	0.8073	0.8075
w/o	+VO-LN	0.8075	0.8076	0.8073
+LN	+LN	0.8096	0.8099	0.8100
+LN	+VO-LN	0.8098	0.8102	0.8103
+VO-LN	+VO-LN	0.8097	0.8101	0.8101

from Table 5 that we have best performance model with BatchNorm for categorical feature.

4.6 Performance of NormDNN (RQ5)

If we adopt the following unified normalization combination strategy in DNN ranking model: variance-only LayerNorm or Layer-Norm for numerical feature, BatchNorm for categorical feature and variance-only LayerNorm for MLP, we can gain the best performance model on all three datasets, which achieves significantly better performance than complex model such as xDeepFM. We call this normalization enhanced model with this unified normalization strategy "NormDNN" in this paper.

The experimental results in Table 6 prove this observation. It is easy to find that NormDNN is more applicable in many industry applications because of its better performance and high computation efficiency compared with many state-of-the-art complex neural network models.

4.7 Normalization on DeepFM and xDeepFM Models (RQ6)

In the following part of the paper, we study the impacts of normalization on two other popular deep neural network models, including DeepFM and xDeepFM. We design some normalization experiments to observe whether it also works for these two models. Notice that the input of FM component in DeepFM is the feature embedding before normalization. The performance of DeepFM degrades if FM component uses the same feature embedding after normalization as DNN component does.

The results in Table 7 show the impact of the various normalizations on model performance. It can be observed that:

(1) The performances of both models apparently increase when we add normalization into different parts of model. The experimental results tell us the normalization works for many current state-of-the-art models.

(2) If we select correct normalization combination for simple model such as DNN or DeepFM, the performances of the model with normalization outperform complex model without normalization such as xDeepFM. That means it's more practical to adding normalization on simple models in real-life applications.

5 CONCLUSION

In this paper, we firstly apply various normalization approaches to the feature embedding part and the MLP part of DNN model. Extensive experiments are conduct on three real-world datasets and the experiment results demonstrate that the correct normalization significantly enhances model's performance. We also simplify the LayerNorm and propose two new and effective normalization methods in this work. Further more, we find the variance of normalization mainly contributes to this positive effect.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 46–54.

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 7–10.

[4] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[5] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 355–364.

[6] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 1–9.

[7] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 169–177.

[8] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[9] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1754–1763.

[10] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, and et al. 2013. Ad Click Prediction: A View from the Trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. Association for Computing Machinery, New York, NY, USA, 1222–1230. https://doi.org/10.1145/2487575.2488200

[11] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[12] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. How does batch normalization help optimization?. In *Advances in Neural Information Processing Systems*. 2483–2493.

[13] Sheng Shen, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. 2020. Rethinking Batch Normalization in Transformers. *arXiv preprint arXiv:2003.07845* (2020).

[14] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[15] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. ACM, 12.

[16] Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 3–19.

[17] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[18] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. 2019. Understanding and Improving Layer Normalization. In *Advances in Neural Information Processing Systems*. 4383–4393.

[19] Junlin Zhang, Tongwen Huang, and Zhiqi Zhang. 2019. FAT-DeepFFM: Field Attentive Deep Field-aware Factorization Machine. *arXiv preprint arXiv:1905.06336* (2019).

[20] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *European conference on information retrieval*. Springer, 45–57.

[21] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.