Deep CTR Prediction in Display Advertising

Junxuan Chen

ABSTRACT

Click through rate (CTR) prediction of image ads is the core task of online display advertising systems, and logistic regression (LR) has been frequently applied as the prediction model. However, LR model lacks the ability of extracting complex and intrinsic nonlinear features from handcrafted high-dimensional image features, which limits its effectiveness. To solve this issue, in this paper, we introduce a novel deep neural network (DNN) based model that directly predicts the CTR of an image ad based on raw image pixels and other basic features in one step. The DNN model employs convolution layers to automatically extract representative visual features from images, and nonlinear CTR features are then learned from visual features and other contextual features by using fully-connected layers. Empirical evaluations on a real world dataset with over 50 million records demonstrate the effectiveness and efficiency of this method.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

Keywords

DNN, CNN, Click through rate, Image Ads, Display Advertising

1. INTRODUCTION

Online display advertising generates a significant amount of revenue by showing image ads on various web pages. More and more advertisers prefer displaying image ads (Figure 1) because they are more attractive and comprehensible compared with textual ads. This has led to a huge demand on approaches that are able to choose the most proper image ad to show for a particular user when he or she is visiting a web page.

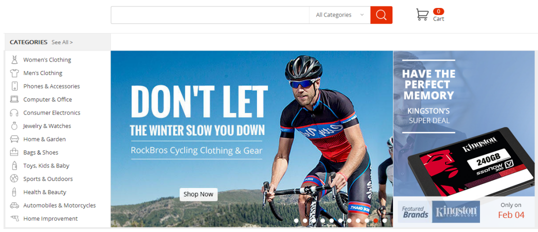

Figure 1: Display ads on an e-commerce web page.

In most online advertising system, predicting a user response for a display advertisement, especially click through rate (CTR), is the core task of ads allocation. Typically an ads system predicts and ranks the CTR of available ads based on contextual information, and then shows the top ads to the users. In general, prediction models are learned from past click data based on machine learning techniques.

Features that are used to represent an ad are extremely important in a machine learning model. In recent years, to make the CTR prediction model more accurate, many researchers use millions of features to describe a user's response record (we call it an ad impression). Typically, an image ad impression has basic features and visual features. The basic features are information about users, products and ad positions in a web page, etc. Visual features describe the visual appearance of an image ad at different levels. For example, color and texture are low level features, while face and other contextual objects are high level features. Low level and high level features both have the power to influence the CTR of an image ad (Figure 2).

Among different machine learning models that have been applied to predict ads CTR using the above features, Logistic regression (LR) is the mostly well-known and widely-used one due to its simplicity and effectiveness. Also, LR is easy to be parallelized on a distributed computing system thus it is not challenging to make it work on billions of samples. Being able to handle big data efficiently is necessary for a typical advertising system especially when the prediction model needs to be updated frequently to deal with new ads. However, LR is a linear model which is inferior in extracting complex and effective nonlinear features from handcrafted feature pools. Though one can mitigate this issue by computing the second-order conjunctions of the

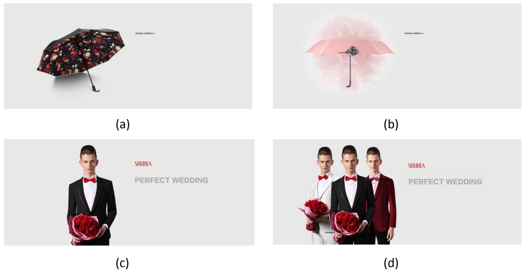

Figure 2: CTRs of image ads (a) and (b) are 1.27% and 0.83%. (b) suffers from low contrast between product and background obviously. CTRs of (c) and (d) are 2.40% and 2.23%. We find too many subjects in an men's clothing ad brings negative effect. (the number of impressions of each ad is sufficiently high to make the CTR has statistical meaning).

features, it still can not extract higher-order nonlinear representative features and may cause feature explosive if we continue increasing the conjunction order.

To address these problems, other methods such as factorization machine, decision tree, neural network are widely used. Though these methods can extract non-linear features, they only deal with basic features and handcrafted visual features, which are inferior in describing images. In this paper, we propose a deep neural network (DNN) to directly predict the CTR of an image ad from raw pixels and other basic features. Our DNN model contains convolution layers to extract representative visual features and then fully-connected layers that can learn the complex and effective nonlinear features among basic and visual features. The main contributions of this work can be summarized as follows:

1. To the best of our knowledge, it is the first paper that proposes an end to end deep model that directly predicts the CTR of image ads in one step. Experiments show that our deep net model outperforms state-of-the-art methods in a real world dataset with more than 50 million user response records.

2. Efficient methods are introduced to tackle the challenge of high-dimensionality and huge data amount in the model training stage. The proposed methods reduce the training time significantly and make it feasible to train on a normal PC with GPUs even with large-scale real world training data.

The paper is organized as follows. Section 2 introduces the related work, followed by an overview of our scheme in Section 3. In Section 4, we describe the proposed DNN model in detail, and we show the challenges in the training stage as well as our solutions in Section 5. Section 6 presents the experimental results, and then Section 7 is the conclusion.

2. RELATED WORK

We consider display advertising CTR prediction and deep neural network are two mostly related areas to our work.

2.1 Display Advertising CTR prediction

Since the display advertising has taken a large share of online advertising market, many works addressing the CTR prediction problem have been published. In [19, 3], authors handcraft many features from raw data and use logistic regression (LR) to predict the click-through rate. [4] also uses LR to deal with the CTR problem and scales it to billions of samples and millions of parameters on a distributed learning system. In [18], a Hierarchical Importance-aware Factorization Machine (HIFM) is introduced, which provides a generic latent factor framework that incorporates importance weights and hierarchical learning. In [6], boosted decision trees have been used to build a prediction model. In [10], a model which combines decision trees with logistic regression has been proposed, and outperforms either of the above two models. [2] combines artificial neural networks with decision trees and also brings an improvement. All of these methods are very effective when deal with ads without images. However, when it comes to the images ads, they can only use pre-extracted image features, which is less flexible to take account of the unique properties of different datasets.

Therefore, the image features in display advertisement have received more and more attention. In [1, 5], the impact of visual appearance on user's response in online display advertising is considered at first time. They extract over 30 handcrafted features from ad images and build a CTR prediction model using image and basic features. [16] is the most related work in literature with us, in which convolution neural network (CNN) is used to extract image features from ads. However, there are two important differences between their method and ours. First, they do not consider basic features when extracting image features using CNN. Second, when predicting the CTR they use logistic regression which lacks the ability in exploring the complex relations between image and basic features. Most of the information in their image features is redundant such as product category which is included in basic features. As a result, their model only achieves limited improvements when combining both kinds of features. Worse still, when the dataset contains many categories of products, it can hardly converge when training. Our model uses an end to end model to predict the CTR of image ads using basic features and raw images in one step, in which image features can be seen as supplementary to the basic features.

2.2 Deep Neural Network

In recent years, deep neural network has achieved big breakthroughs in many fields. In computer vision filed, convolution neural network (CNN) [14] is one of the most efficient tools to extract effective image features from raw image pixels. In speech recognition, deep belief network (DBN) [11] is used and much better performance is obtained comparing with Gaussian mixture models. DNN has also been used in CTR prediction in recent public competitions[1,2]. In these two competitions, only basic features are available. A four-layer DNN which only uses basic features achieves better or comparable performance than LR with feature conjunction, factorization machines, decision trees, etc. Comparing with this DNN, our model can extract more powerful features by taking consideration of the visual features in image ads.

[1]https://www.kaggle.com/c/avazu-ctr-prediction
[2]https://www.kaggle.com/c/criteo-display-ad-challenge

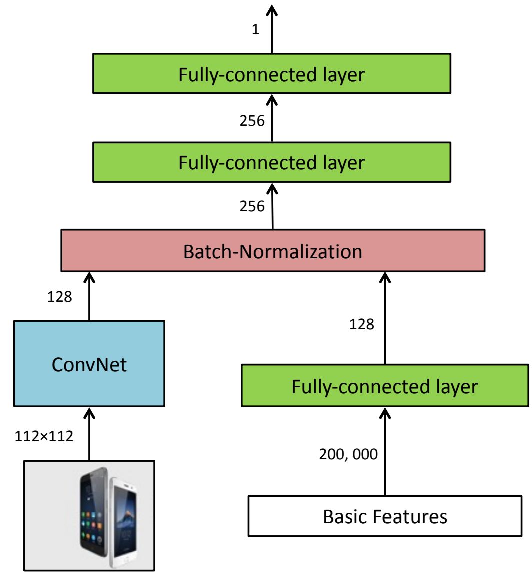

Figure 3: The overall architecture of the network. The output of each fully-connected layer is then pass through a ReLU nonlinear activation function.

3. OVERVIEW

As aforementioned, in this paper, each record of user's behavior on an ad is called an impression. denoted by x. Each impression has an image u with a resolution of around 120×200. Besides the image, the basic feature vector is denoted by $v \in \mathbf{R}^d$ such as the user's gender, product's category, ad position in the web page, and usually d can be very large, say, from a few thousand to many million. Our goal is to predict the probability that a user clicks on an image ad given these features. We will still use logistic regression to map our predicted CTR value $\hat{y}$ to 0 to 1, thus the CTR prediction problem can be written as:

$$\hat{y} = \frac{1}{1+e^{-z}} \tag{1}$$

$$z = f(x) \tag{2}$$

where $f(.)$ is what we are going to learn from training data, that is, the embedding function that maps an impression to a real value z. Suppose we have N impressions $\mathbf{X} = [x_1, x_2...x_N]$ and each with a label $y_i \in \{0, 1\}$ depends on the user's feedback, 0 means *not clicked* while 1 means *clicked*. Then the learning problem is defined as minimizing a Logarithmic Loss (Logloss):

$$L(\mathbf{W}) = -\frac{1}{N}\sum_i (y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)) + \lambda ||\mathbf{W}||^2 \tag{3}$$

where $\mathbf{W}$ is the parameters of the embedding function $f(.)$ and λ is a regularization parameter that controls the model complexity.

In this model, what we need to learn is the embedding function $f(.)$. Conventional methods extract handcrafted visual features from raw image u and concatenate them with basic features v, then learn linear or nonlinear transformations to obtain the embedding function. In this paper we learn this function directly from raw pixels of an image ad and the basic features using one integrated deep neural network.

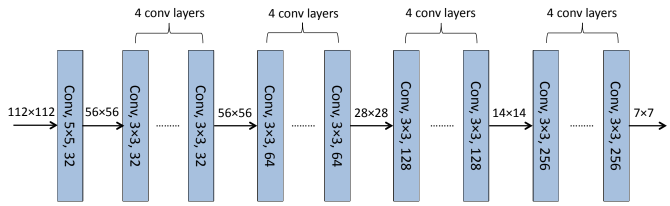

Figure 4: The architecture of the 17-layer *Convnet* in our model.

4. NETWORK ARCHITECTURE

Considering basic features and raw images come from two different domains, we cannot simply concatenate them together directly in the network. Training two separate networks is also inferior since it cannot take into account the correlations between the two features. As a result, our network adopts two different sub-networks to deal with basic features and raw images, respectively, and then uses multiple fully-connected layers to capture their correlations.

As illustrated in Figure 3, a deep neural network called DeepCTR is designed which contains three parts. One part, *Convnet*, takes raw image u as input and follows with a convolution network. The output of the *Convnet* is a feature vector of the raw image. The second part which is called *Basicnet*, takes basic features v as input and follows a fully-connected layer to reduce the dimensionality. Subsequently, outputs of *Convnet* and *Basicnet* are concatenated into one vector and fed to two fully-connected layers. The output of the last fully-connected layer is a real value z. This part is called *Combnet*. On the top of the whole network, Logloss is computed as described in Section 3.

The design of *Convnet* is inspired by the network in [8, 20], as shown in Figure 4. The network consists of 17 convolution layers. The first convolution layer uses 5×5 convolution kernels. Following first layer, there are four groups and each has four layers with 3×3 kernels. We do not build a very deep network such as more than 50 layers in consideration of the trade off between performance and training time. We pre-train the *Convnet* on the images in training dataset with category labels. We use two fully-connected layers with 1024 hidden nodes (we call them *fc18* and *fc19*), a fully-connected layer with 96-way outputs (we call it *fc20*) and a softmax after the *Convnet* in pre-training period. Since our unique images set is smaller (to be detailed in Section 6) than ImageNet [7], we use half number of outputs in each group comparing with [8]. After pre-training, a 128-way fully-connected layer is connected behind the last convolution layer. Then we train the whole DeepCTR using Logloss from end to end.

5. SPEED UP TRAINING

Typically a deep neural network has millions of parameters which make it difficult to train. With the development of GPUs, one can train a deep CNN with 1 million images in two days on a single machine. However, it is still time infeasible for our network since we have more than 50 million samples. Moreover, the dimensionality of basic features is nearly 200,000 which leads to much more parameters in our network than a normal deep neural network. Directly training our network on a single machine may take hundreds of days to converge according to a rough estimation. Even using multi-machine can hardly resolve the training problem.

To make it feasible to train our DeepCTR model on a single machine, we adopt two techniques: using sparse fully-connected layer and a new data sampling scheme.

5.1 Sparse Fully-Connected Layer

In CTR prediction, the basic feature of an ad impression includes user information like gender, age, purchasing power, and ad information like ad ID, ad category, ad zone, etc. This information is usually encoded by one-hot encoding which makes the feature dimension very large. For example, it is nearly 200,000 in our dataset. Consequently, the first fully-connected layer using the basic feature as input has around 60 million parameters, which is similar to the number of parameters in AlexNet [14]. However, the basic feature is extremely sparse due to the one-hot encoding. Using sparse matrix in first fully-connected layer can largely reduce the computing complexity and GPU memory usage.

In our model, we use compressed sparse row (CSR) format to represent a batch of basic features V. When computing network forward, sparse matrix multiplication can be used. When backward pass, we only need to update the weights that link to small numbers of nonzero dimensions.

5.2 Data Sampling

Another crucial issue in CTR model training is data amount. The training data in CTR prediction usually has lots of noises. To train a robust CTR prediction model, we need millions of ad samples to reduce the effect of those noises. Though the sparse fully-connected layer can largely reduce the forward-backward time in *Basicnet*, training the *Convnet* on such a large dataset still requires too much time if we adopt an ordinary training strategy. In this paper, we propose a much faster training method based on an intrinsic property of the image ads click records, that is, many impressions share a same image. Though the total size of the dataset is very large, the number of unique images is relatively smaller. Since a good many of basic features can be processed quickly by sparse fully-connected layer, in this paper we employ a data sampling method that groups basic features of a same image together to tackle this problem, which is detailed as follows.

Suppose the unique images set in our dataset is $\mathbf{U}$, the set of impressions related to an image u are $\mathbf{X}_u$ and basic features are $\mathbf{V}_u$. At each iteration, suppose the training batch size is n, we sample n images U from $\mathbf{U}$. Together with each image $u \in U$, we sample k basic features V_u from $\mathbf{V}_u$ with replacement. Thus we have n images and kn basic features in each batch. After *Convnet*, we have n image features. For each feature vector $conv_u$ we copy it k times to have C_u and send them forward to *Combnet* along with V_u. In backforward time, the gradient of each image feature vector can be computed as:

$$\nabla(conv_u) = \frac{1}{k} \sum_{c \in C_u} \nabla(c) \tag{4}$$

The training method is summarized in Alg. 1 and Alg. 2.

Algorithm 1 Training a DeepCTR network

Input: : Network Net with parameter $\mathbf{W}$, unique images set $\mathbf{U}$, basic features set $\mathbf{V}$, labels $\mathbf{Y}$, batch size n, basic feature sample number k.

Output: : Network for CTR prediction, Net

Initialize Net.

Compute the sample probability $p(u)$ of each image u,

$$p(u) = \frac{\#\mathbf{V}_u}{\sum_{u' \in \mathbf{U}} \#\mathbf{V}_{u'}} \tag{5}$$

repeat

  Sample n images U according to $p(u)$.

  For each u in U, sample k basic features V_u from $\mathbf{V}_u$ with labels Y_u uniformly with replacement.

  $forward\_backward(Net, U, V, Y)$.

until Net converges

Algorithm 2 $forward\_backward$

Input: : Network Net with parameters $\mathbf{W}$ which contains a *Convnet*, *Basicnet* and *Combnet*, image samples U, basic features V, labels Y, basic feature sample number k.

Compute the feature vector $conv_u$ of each image u: $conv = net\_foward(Convnet, U)$

Copy each feature vector k times so we have C.

$loss = net\_forward(Basicnet$ and $Combnet, V, C)$.

$\nabla(C) = net\_backward(Combnet$ and $Basicnet, loss)$.

Compute $\nabla(conv_u)$ of each image u according to Eq. 4.

$net\_backward(Convnet, \nabla(conv))$.

Update network Net.

6. EXPERIMENT

In this section, a series of experiments are conducted to verify the superiority of our DeepCTR net.

6.1 Experimental Setting

6.1.1 Dataset

The experiment data comes from a commercial advertising platform (will expose it in the final version) in a random week of year 2015. We use the data from first six days as our training data and the data from last day as testing data. As described in Section 3, each impression consists of an ad x and a label y. An impression has an image u (Figure 2) and a basic feature vector v. The size of training data is 50 million while testing set is 9 million. We have 101,232 unique images in training data and 17,728 unique images in testing data. 3,090 images in testing set are never shown in training set. The basic feature v is one-hot encoded and has a dimensionality of 171,858. Following information is consisted by basic features:

1. Ad zone. The display zone of an ad on the web page. We have around 700 different ad zones in web pages.

2. Ad group. The ad group is a small set of ads. The ads in an ad group share almost same products but different ad images (in Figure 2, (a) and (b) belong to an ad group while (c) and (d) belong to another group). We have over 150,000 different ad groups in our dataset. Each ad group consists less than 20 different ads.

3. Ad target. The target groups of the ad. We have 10 targets in total.

4. Ad category. The category of the product in ads. We have 96 different categories, like clothing, food, household appliances.

5. User. The user information includes user's gender, age, purchasing power, etc.

6.1.2 Baselines

We use LR only with basic features as our first baseline. We call this method *lr basic* in following experiments. We also evaluate a two hidden layers DNN model only using basic features. The numbers of outputs of two hidden layers are 128 and 256 respectively. The model can be seen as our DeepCTR net without the *Convnet* part. This method is called *dnn basic*. We further replace the *Convnet* in our DeepCTR net with pre-extracted features, SIFT [15] with bag of words and the outputs of different layers of the pre-trained *Convnet*. We call these two methods *dnn sift* and *dnn layername* (for example *dnn conv17*).

6.1.3 Evaluation Metric

We use two popular metrics to evaluate the experiment result, Logloss and the area under receiver operator curve (AUC). Both of these metrics are relative to *lr basic*. Logloss can quantify the accuracy of the predicted click probability. AUC measures the ranking quality of the prediction.

6.1.4 Network Configuration

In our *Convnet*, a 112×112 random crop and horizontal mirror for the input image are used for data augmentation. Each group has four convolution layers followed by a batch normalization [12] and a ReLU [17] activation. The stride of the first convolution layer is 2 if the output size of a group halves. We initialize the layer weights as in [9]. When pre-training the *Convnet* on our image dataset with category labels, we use SGD with a mini-batch size of 128. The learning rate starts from 0.01 and is divided by 10 when test loss plateaus. The pre-trained model converges after around 120 epochs. The weight decay of the net is set as 0.0001 and momentum is 0.9.

After pre-training *Convnet*, we train our DeepCTR model from end to end. Other parts of our net use the same initialization method as *Convnet*. We choose the size of mini-batch n as 20, and $k = 500$. That is to say, we deal with 10,000 impressions per batch. We start with the learning rate 0.1, and divided it by 10 after 12, 20 and 28 epochs. The *Convnet* uses a smaller initial learning rate 0.001. The weight decay of the whole net is set as 0.00005. The *dnn basic*, *dnn sift* and *dnn layername* use the same learning strategy.

We implement our network on C++ Caffe toolbox [13] with some modifications like sparse fully-connected layer.

6.2 Results and Discussion

In this section we compare the results of various methods and the influence of some network structures. First we compare the results of models with deep features in different levels. We plot the two metrics of *dnn conv13*, *dnn conv17*, *dnn fc18*, *dnn fc19*, and *dnn fc20* (Table 1). From the results, we find that *dnn conv17* and *dnn fc18* achieve best performance. Image features in these layers are of relatively high level but not highly group invariant [22]. Comparing with

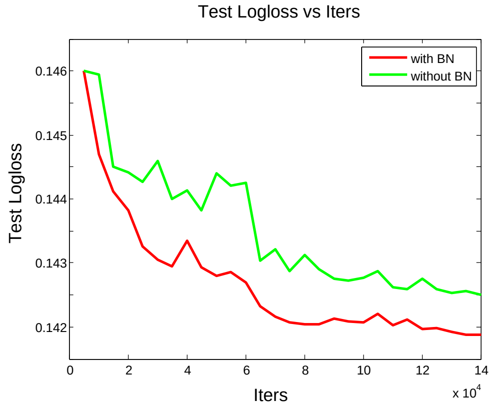

Figure 5: Test Logloss of the DeepCTR net with/without batch normalization in *Combnet*.

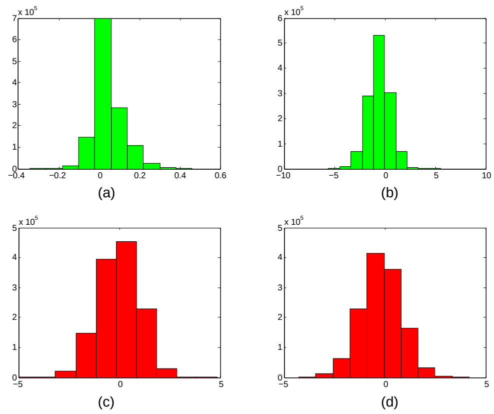

Figure 6: (a) and (b) are the histograms of outputs of *Basicnet* and *Convnet* without batch normalization while (c) and (d) with batch normalization.

following fully-connected layers, they have more discriminations in same category. Comparing with previous layers, they contain features in a sufficiently high-level which are superior in describing the objects in images. Consequently, we connect *conv17* layer in our DeepCTR model. We do not choose *fc18* because it needs higher computations. We have also tried to compare our DeepCTR with the approach in [16]. However the model in [16] does not converge on our dataset.

Comparison between baselines is shown in Table 1 too. From the result, it can be seen that a deep neural network and image features can both improve the CTR prediction accuracy. Comparing with handcrafted features, like SIFT, deep features have stronger power in describing the image, which leads to a significant improvement in the prediction accuracy. Our DeepCTR model goes one step further by using an end to end learning scheme. Ensemble of multiple deep networks usually brings better performance, so we train 3 DeepCTR models and average the their predictions, and it gives the best AUC and Logloss. Compared with *lr basic*, the AUC increase will bring us 1~2 percent CTR increase in the advertising system (according to online experiments), which will lead to over 1 million earnings growth per day for

Table 1: **relative AUC and Logloss. We omit *dnn* of methods using deep neural network with pre-extracted features.**

method	lr basic	basic	sift	conv13	conv17	fc18	fc19	fc20	DeepCTR	3 DeepCTRs
AUC(%)	-	0.45	0.54	1.25	1.32	1.31	1.11	0.97	1.62	**1.89**
Logloss(%)	-	-0.39	-0.45	-0.79	-0.86	-0.86	-0.74	-0.69	-1.11	**-1.30**

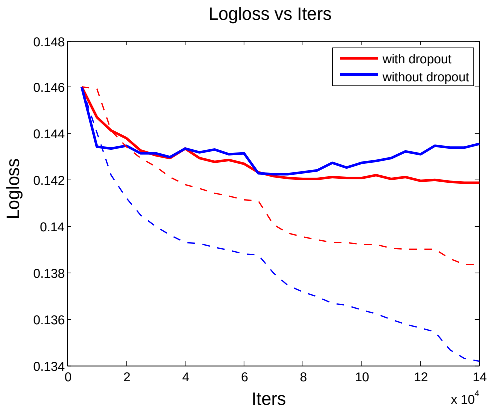

Figure 7: Logloss of the DeepCTR net with/without dropout in *Combnet*. Dashed lines denote training loss, and bold lines denote testing loss.

an 100 million ads business.

We further explore the influence of network structures in our DeepCTR model empirically. First we find that the batch normalization in the *Combnet* can speed up training and largely improve performance (Figure 5). To investigate the reason, we show the histogram (Figure 6) of the outputs of *Convnet* and *BaiscNet*. We can see from the histogram that two outputs have significant difference in scale and invariance. Simply concatenating these two different kinds of data stream makes the following fully-connected layer hard to converge.

Dropout [21] is an efficient way to prevent over-fitting problem in deep neural network. Most deep convolution networks remove the dropout because batch normalization can regularize the models. However, in our DeepCTR model, we find it still suffers from over-fitting without dropout. We compare the loss curves of the model with/without dropout in the last two fully-connected layers. We can see that the model with dropout achieves lower testing Logloss, though we need more time to reach the lowest test loss.

We also evaluate the performance of the sparse fully-connected layer and our data sampling method. We plot computing time and memory overhead (Table 2) of the sparse fully-connected layer comparing with dense layer. Loss curves of training and testing are exactly the same since sparse fully-connected layer does not change any computing results in the net, so we do not plot them. From this table we can find dense layer requires much more computing time and memory than sparse one. Using sparse layer allows a lager batch size when training, which makes the net easier to converge.

Finally, we investigate whether the performance of our model descends using our data sampling method. We compare our method with a throughout shuffle of the dataset. Since training a DNN model with convolution layers on 50 million samples takes too much time, we only conduct experiments on *dnn conv17* model. We believe that our DeepCTR model performs similarly with *dnn conv17* model. From Table 3 we can see the performance of the model is not influenced by the data sampling method. At the same time, our method only costs 1/500 of the training time compared with the approach without data sampling. Using this data sampling method, training our DeepCTR model from end to end only takes around 12 hours to converge on a NVIDIA TESLA k20m GPU with 5 GB memory, which is acceptable for an online advertising system requiring daily update.

Table 2: forward-backward time and GPU memory overhead of first fully-connected layer with a batch size of 1,000.

	time (ms)	memory (MB)
sparse layer	6.67	397
dense layer	189.56	4667

Table 3: AUC and Logloss of *dnn conv17* model with our data sampling and a throughout shuffle.

	AUC(%)	Logloss(%)
data sampling	1.32	-0.86
throughout shuffle	1.31	-0.85

7. CONCLUSIONS

CTR prediction plays an important role in online display advertising business. Accurate prediction of the CTR of ads not only increases the revenue of web publishers, also improves the user experience. In this paper we propose an end to end integrated deep network to predict the CTR of image ads. It consists of *Convnet*, *Basicnet* and *Combnet*. *Convnet* is used to extract image features automatically while *Basicnet* is used to reduce the dimensionality of basic features. *Combnet* can learn complex and effective non-linear features from these two kinds of features. The usage of sparse fully-connected layer and data sampling techniques speeds up the training process significantly. We evaluate DeepCTR model on a 50 million real world dataset. The empirical result demonstrates the effectiveness and efficiency of our DeepCTR model.

8. REFERENCES

[1] J. Azimi, R. Zhang, Y. Zhou, V. Navalpakkam, J. Mao, and X. Fern. The impact of visual appearance on user response in online display advertising. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 457–458. ACM, 2012.

[2] A. I. Baqapuri and I. Trofimov. Using neural networks for click prediction of sponsored search. *arXiv preprint arXiv:1412.6601*, 2014.

[3] D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. In *Proceedings of the 17th international conference on World Wide Web*, pages 417–426. ACM, 2008.

[4] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):61, 2014.

[5] H. Cheng, R. v. Zwol, J. Azimi, E. Manavoglu, R. Zhang, Y. Zhou, and V. Navalpakkam. Multimedia features for click prediction of new ads in display advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 777–785. ACM, 2012.

[6] K. S. Dave and V. Varma. Learning the click-through rate for rare/new ads from similar ads. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 897–898. ACM, 2010.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[10] X. He, J. Pan, O. Jin, Xu, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1–9. ACM, 2014.

[11] G. Hinton, L. Deng, D. Yu, G. E. Dahl, Mohamed, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[15] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[16] K. Mo, B. Liu, L. Xiao, Y. Li, and J. Jiang. Image feature learning for cold start problem in display advertising. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3728–3734. AAAI Press, 2015.

[17] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[18] R. J. Oentaryo, E.-P. Lim, J.-W. Low, D. Lo, and M. Finegold. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 123–132. ACM, 2014.

[19] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530. ACM, 2007.

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

[21] 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*, 15(1):1929–1958, 2014.

[22] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.