A STEPPED SAMPLING METHOD FOR VIDEO DETECTION USING LSTM

Anonymous authors
Paper under double-blind review

ABSTRACT

Artificial neural networks that simulate human achieves great successes. From the perspective of simulating human memory method, we propose a stepped sampler based on the repeated input. We repeatedly inputted data to the LSTM model stepwise in a batch. The stepped sampler is used to strengthen the ability of fusing the temporal information in LSTM. We tested the stepped sampler on the LSTM built-in in PyTorch. Compared with the traditional sampler of PyTorch, such as sequential sampler, batch sampler, the training loss of the proposed stepped sampler converges faster in the training of the model, and the training loss after convergence is more stable. Meanwhile, it can maintain a higher test accuracy. We quantified the algorithm of the stepped sampler. We assume that, the artificial neural networks have human-like characteristics, and human learning method could be used for machine learning. Our code will be available online soon.

1 INTRODUCTION

The emergence of convolutional neural networks (CNN) (LeCun et al., 1989) has improved the self-learning ability of artificial neural networks. Recurrent Neural Network (RNN) (Mikolov et al., 2010) is used to process the temporal information data. RNN takes the output of the previous time period as the input of the next time period, effectively using the temporal information of the input sequence.

RNN sometimes may have the problem of gradient disappearance or gradient explosion. Hochreiter et al. (Hochreiter & Schmidhuber, 1997) proposed LSTM. LSTM adds gates to RNN, thus it can effectively avoid the problem of gradient disappearance or explosion. These gates include the forgetting gates, the input gates, and the output gates. The forgetting gate seems to be the most important among them. LSTM may simulate the memory process of human brain. Human brain selectively forgets some information for learning better.

Consider that one of the principles of neural networks may be learned from biological neural networks, for those artificial neural networks with the memory effects, such as LSTM, learning from the memory method of human, which is the repeated input and timely review, we study the effect of this method with repeated input on LSTM detection results, without considering changing the LSTM network structure.

In this study, we learn the effect of the proposed input method on neural networks with memory, such as LSTM. Specifically, it is to repeatedly input training data by simulating the repeated input and timely review method of the human memory, in a stepped sampling way.

2 RELATED WORK

Gibbs sampling is one of the earlier data sampling algorithms, which is proposed by Geman et al. (Geman & Geman, 1984) in 1984. Gibbs sampling is to make the probability of the data sample approximately equal to the required probability distribution via iterations. Gibbs sampling randomly selects data from an initial input sequence, and iterates according to the specified conditional probabilities, which are related to the required probability distribution of the final sampling data. After iterations, Gibbs sampling generates data which is consistent with the required probability

distribution. Hu et al. (Hu et al., 2018) used neural networks to generate a sampler, which transfer the initial data distribution to the target distribution. The method can generate the sampling data at the same time of training. This method works with the un-normalized probability density function. Wang et al. (Wang et al., 2018) used Generative Adversarial Nets (GAN) (Goodfellow et al., 2014) to generate the negative samples. The approach is the first to combine GAN with the negative sampling method, which improves the training effect of the streaming recommend system. Chu et al. (Chu et al., 2019) proposed a novel sampler that can sample both the positive and the negative data from the input data sequences, so as to let the classifier utilize the Regions of Interests and the background of the data. The sampler is used in the few-shot image classifier, which uses the reinforcement learning method. The reinforcement learning algorithm (Kaelbling et al., 1996) needs to continuously select the regions of interests from the images, subsequently to recognize the content of the Regions of Interests. Sampling these Regions of Interests can improve the efficiency of reinforcement learning, for the reason of the reduction of the input samples.

3 Materials and Methods

This paper is from the perspective of data input, rather than the neural network structure, and study the impact of the memory effect on the temporal sequence neural networks (such as LSTM). The process simulates the memory process of human brain, repeats the input data in a stepped way. The mode we used was the wheel tactic (Smith, 1994) when we recited words, by establishing a novel data sampler in the LSTM model training.

3.1 Dataset

The dataset in the experiment is UCF101 (Soomro et al., 2012), which is a human action recognition video dataset. The name of each folder indicates the annotation of the video.

3.2 Video Temporal Information

We consider that, video temporal information is the correlations of the objects between frames, or within a frame. We also consider that, this kind of correlation could be analogized to the correlation of human knowledge, and machine learning may be similar with human learning. From the point, we derive the equations of video temporal information from Bayes' theorem, and mutual information of information theory.

3.2.1 The Temporal Information Between the Frames

Bayes' theorem (Bayes & Hume, 1763), i.e. a conditional probability, can be described as:

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} \tag{1}$$

where $P(A|B)$ is the probability of event A occurring, under the condition that event B occurs. $P(A \bigcap B)$ is the probability of event A and event B occurring at the same time. $P(B)$ is the probability of event B occurring. $P(B) \neq 0$.

We consider that, Bayes' theorem reflects a temporal correlation, that is, event B occurs first, and then event A occurs. The video frame could be analogized to a kind of Venn diagram, and the objects in the frame could be analogized to the events in the Venn diagram. Thus, we apply this event temporal correlation to the temporal information between frames.

We use the area ratio of object A in the video frame R(A) to replace the occurrence probability of the event. The area ratio is defined as:

$$R(A) = \frac{area\ of\ object\ A}{area\ of\ frame} \tag{2}$$

Refer to Bayes' theorem, according to the sequential relations of the temporal information between frames, i.e., the state of the object in the next frame is derived from the state of the object in the previous frame, the temporal information between the frames of the object A is:

$$T_A(nf|pf) = \frac{R(A_{pf} \bigcap A_{nf})}{R(A_{pf})} \tag{3}$$

where "nf" represents the next frame, "pf" represents the previous frame, $T_A(nf|pf)$ denotes the temporal information between the frames of the object A, $R(A_{pf} \bigcap A_{nf})$ represents the overlapping area ratio of the previous frame and the next frame of the object A. $R(A_{pf})$ represents the area ratio of the object A in the previous frame.

Similarly, the temporal information between the frames of the object B is:

$$T_B(nf|pf) = \frac{R(B_{pf} \bigcap B_{nf})}{R(B_{pf})} \tag{4}$$

Then, the temporal information between the frames can be described as:

$$T_{bf} = T_A(nf|pf) + T_B(nf|pf) + \cdots = \frac{R(A_{pf} \bigcap A_{nf})}{R(A_{pf})} + \frac{R(B_{pf} \bigcap B_{nf})}{R(B_{pf})} + \cdots \tag{5}$$

where T_{bf} denotes the temporal information between the frames.

3.2.2 The Temporal Information Within a Frame

The temporal information within a frame reflects the correlations of objects in the same image (frame). The correlations within the same image can be represented by mutual information (Shannon, 1948) in information theory. The discrete equation of mutual information is

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log(\frac{p(x,y)}{p(x)p(y)}) \tag{6}$$

where $p(x, y)$ is the joint probability function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability functions of X and Y.

Referring to the discrete equation of mutual information, we propose the temporal information within a frame as: when there is an overlapping area between object A and object B in the frame,

$$T(A \bigcap B) = \sum_{B \in F} \sum_{A \in F} R(A \bigcap B) log(\frac{R(A \bigcap B)}{R(A)R(B)}) \tag{7}$$

where $T(A \bigcap B)$ represents the intra-frame temporal information when object A and object B have overlapping areas, F represents all objects in the frame, $R(A \bigcap B)$ represents the overlapping area ratio of object A and object B in the frame, $R(A)$ and $R(B)$ respectively represent the area ratio of the object A and the object B in the frame.

Correlated objects in a frame generally have overlapping areas. For example, blue sky and white clouds, grass and pets, roads and cars, etc. For the correlation of objects in a frame without overlapping areas, since most of the correlated objects in a frame may have overlapping areas, the mathematical representation of the correlation of objects in a frame without overlapping areas can be weakened by a certain form.

We realize this form by taking the logarithm. Then, its mathematical expression is

$$T(A \bigcup B) = \sum_{B \in F} \sum_{A \in F} log\left(R(A \bigcup B) log(\frac{R(A \bigcup B)}{R(A)R(B)})\right) \tag{8}$$

where $T(A \bigcup B)$ represents the temporal information in a frame with no overlapping area between object A and object B, F represents the all objects in the frame, and $R(A \bigcup B)$ represents the total area ratio of object A and object B in a frame. $R(A)$ represents the area ratio of the object A in the frame, and $R(B)$ represents the area ratio of the object B in the frame.

Since the above equation uses a logarithmic operator, when $R(A \bigcup B) log(\frac{R(A \bigcup B)}{R(A)R(B)}) < 1$, $log\left(R(A \bigcup B) log(\frac{R(A \bigcup B)}{R(A)R(B)})\right) < 0$; when $R(A \bigcup B) log(\frac{R(A \bigcup B)}{R(A)R(B)}) > 1$, $log\left(R(A \bigcup B) log(\frac{R(A \bigcup B)}{R(A)R(B)})\right) > 0$. This kind of positive and negative value realizes the reduction of the correlation of non-overlapping objects in a frame. Moreover, such negative values are very small. We consider that, the logarithmic operator is suitable for the correlation of non-overlapping objects in a frame.

Then, the temporal information within the frame can be the sum of the overlapping areas and the non-overlapping areas. It can be expressed in mathematical expression as:

$$\begin{aligned} T_{wf} &= T(A \bigcap B) + T(A \bigcup B) \\ &= \sum_{B \in F} \sum_{A \in F} R(A \bigcap B) log(\frac{R(A \bigcap B)}{R(A)R(B)}) + \sum_{B \in F} \sum_{A \in F} log\left(R(A \bigcup B) log(\frac{R(A \bigcup B)}{R(A)R(B)})\right) \end{aligned} \tag{9}$$

where T_{wf} denotes the temporal information within a frame.

3.2.3 The Equation of Video Temporal Information

The video temporal information is the sum of the one between the frames and the one within a frame. Then, the temporal information of the frame is:

$$\begin{aligned} T &= T_{bf} + T_{wf} \\ &= \frac{R(A_{pf} \bigcap A_{nf})}{R(A_{pf})} + \frac{R(B_{pf} \bigcap B_{nf})}{R(B_{pf})} + \cdots \\ &+ \sum_{B \in F} \sum_{A \in F} R(A \bigcap B) log(\frac{R(A \bigcap B)}{R(A)R(B)})\Big|_{A \bigcap B} \\ &+ \sum_{B \in F} \sum_{A \in F} log\left(R(A \bigcup B) log(\frac{R(A \bigcup B)}{R(A)R(B)})\right)\Big|_{A \bigcup B} \end{aligned} \tag{10}$$

3.3 LSTM

The LSTM architecture we used in this paper is to start with a CNN backbone. The CNN backbone has four convolutional layers. The dimension of convolution kernels of each convolutional layer is 32, 64, 128, 256. The size of each convolution kernel is $5 \times 5, 3 \times 3, 3 \times 3, 3 \times 3$, the stride of the convolution kernel is 2, and the padding is 0. Each convolutional layer is followed by a batch normalization (BN) (Ioffe & Szegedy, 2015) layer and a ReLU layer. The last part of the CNN model is 3 fully connected (FC) layers, which use dropout function. The dimensions of the 3 FC layers are 1024, 768 and 512 respectively. The LSTM used in the paper is the model existed in PyTorch. The input dimension of LSTM is 512, the hidden layer dimension is 512, and the number of hidden layers is 3. The next is two fully connected (FC) layers followed by dropout function. The dimension of the FC layers is 256. The dropout rate of the CNN backbone and LSTM are both 0.3.

3.4 Commonly Used Samplers in PyTorch

Common sampler in PyTorch (Paszke et al., 2019) include random sampler, weighted random sampler, batch sampler, etc. Batch sampling is commonly used.

3.5 The Stepped Sampler

Analogous to human learning, an important point is repetition. And learning data by sampling should be a kind of repetition, since the data in each batch can be designed to be entered repeatedly. We suppose that, repetition is so important, not only for human beings, but also for computers. To make computers better use the repetition, analogizing the way how we recite words, we propose a stepped repetition input method, which is in the form of the stepped sampler.

The structure of the proposed stepped sampler is illustrated in Figure 1. It is established on the batch sampler. The stepped sampler divides a batch into some sub-batches. There is a duplication between the previous sub-batch and the next sub-batch. Like human memory, this sampler adopts the principle of adjacent repetition (Crowder, 1968), namely, the back of the previous sub-batch is the same with the front of the next sub-batch.

We regard artificial neural networks as biological neural networks. The previous research is to add memory units to deep learning networks, such as RNN, LSTM, etc. In this paper, we try to study the human learning method on the machine system.

The structure of the stepped sampler shows that, the input data of different batches is partly duplicated. The repeated input seems to increase the redundancy, but the experimental results show that, with our experimental environment, this method can accelerate the convergence of LSTM model. There is a stride between the previous sub-batch and the next sub-batch. The stride size n can be set manually. Section 4 describes the comparative experiments on the sampler with different stride size.

3.6 THE ALGORITHM OF THE STEPPED SAMPLER

The stepped sampler is designed on the basis of batch sampler. The algorithm is designed to implement stepped sampling within each batch via the batch sampler. The workflow of the algorithm is as follows: the data first goes through the sequential sampler of PyTorch, then, they are processed to batches by the batch sampler. Finally, the data in each batch are divided into sub-batches with the same strides by the stepped sampler.

As shown in Figure 1, assuming that the iteration number of the stepped sampler in a batch is d, it can be concluded from the figure:

$$L = m + n \times d \tag{11}$$

It can be deduced that, the stepped sampler iteration number per batch, d is:

$$d = \frac{L - m}{n} \tag{12}$$

Especially, if the step stride n was 1 illustrated in Figure 1, it can be seen from Equation 11, the stepped sampler iteration number in a batch, d is:

$$d = L - m \tag{13}$$

Equation 12 and 13 are used as the iteration number within a batch in Algorithm 1. The number of batches is offered by the framework, and the number of epochs is set manually.

The algorithm of the proposed sampler is shown in Algorithm 1. The overall idea is to implement the stepped sampler in each batch after the sequential sampler and batch sampler of PyTorch. Line 12 of Algorithm 1 is that, after each previous stepped sub-batchs output, the starting coordinate is moved by n (step stride) data from the starting position of the previous sub-batch.

4 RESULTS

4.1 EXPERIMENT SETUP

The system used in the experiment was a workstation with 32 GB CPU RAM and a NVIDIA GeForce 1080ti GPU. The processor was Intel i7 8700, the operating system was Ubuntu 16.04 64 bits. PyTorch version used in the experiment was 1.0.1, Python version was 3.6, Numpy version was 1.20.4, Sklearn version was 0.20.4, and Matplotlib, Pandas, tqdm were implemented as the software environment.

4.2 TRAINING

The detection accuracy evaluation and cross-entropy loss were used for the training of the models. The accuracy evaluation in the experiment used the accuracy score tool in the Sklearn package of Python. The cross-entropy loss used the function of PyTorch. The accuracy and loss were graphically depicted in Figure 3 and 4. The accuracy and loss were computed every epoch. The dataset of UCF101 was split into training set and test set by a ratio of 3:1. After training, an overall accuracy and loss were computed by the test set, to evaluate the performance of the models. The epoch was set to about 150.

For the parameter optimization, we used Adam as the optimization algorithm. We experimented different batch sizes and step sizes. Which is shown in Figure 1 that, change the size of L, m and the step stride n for the experiment.

Our experiment is trained from scratch. The learning rate was set to 0.0001. The momentum was set to 0.01. The operator of batch normalization (BN) and ReLU activation were used after each

convolutional layer in the CNN backbone. The CNN backbone is not shown in Figure 1. The data transformation was applied to enhance the network. The input frames were transformed into 256×342 pixels.

4.3 EXPERIMENT RESULTS

Figure 2 are the visualized results.We tested the sampler of batch size 25. Figure 3 presents the experimental results. Each subfigure shows the training loss and test accuracy of the models. The difference of the models is only the sampler, for comparing the results only caused by the sampler. Figure 3(a) is the model of traditional sampler, which is the sequential sampler and the batch sampler in PyTorch, and the other in Figure 3 are the models of the proposed stepped sampler. Figure 3 (b), (c), (d), (e), (f) are only different from the step stride for comparing. From Figure 3, we consider that step stride 2 (batch size 25, step size 20, in Figure 3(c)) is the optimal. The training loss in Figure 3(a) has many jitters, even when the epoch is more than 110, while the training loss in the other subfigures are much smoother, and can converge earlier than the traditional batch sampler model (Figure 3(a)). Nonetheless, the test accuracy score of the traditional batch sampler model (Figure 3(a)) is slightly higher. The test accuracy of Figure 3(a) can be 0.656, while the test accuracy of the model with stride 2 stepped sampler (Figure 3(c)) can be 0.603. The test accuracy of the models in Figure 3 is shown in Table 1.

From Figure 3, the following could be concluded: **a)** In the model training, LSTM with the stepped sampler converges faster than LSTM with the traditional sampler, and the convergence effect is better, i.e., there is no large jitter after the drop. **b)** When the batch size and the step size are fixed, the smaller the step stride was, the worse the detection effect became. Similarly, the larger the step stride was, the worse the detection effect became either. If the batch size and the step size are fixed, the detection effect seems to be a normal distribution of the step stride. **c)** However, LSTM with the traditional sampler whose batch size is 25 has a higher test accuracy on the test set. Although this value is not much higher than the optimal stepped sampler model (Figure 3(c)).

From Table 1 we may conclude that, for the same batch size, the test accuracy of the LSTM with stepped sampler rises faster than the traditional sampler LSTM. This can also be seen in Figure 3. Figure 4 is the illustration of the traditional LSTM and the stepped sampler LSTM, when the batch sizes are all set to 15. The test accuracy score of Figure 4(c) is higher than Figure 4(a), which denotes that, the stepped sampler LSTM may have higher test accuracy than the traditional sampler LSTM, when the batch size is 15, step size is 10, step stride is 5 of the stepped sampler LSTM, and the traditional sampler LSTM is with batch size 15. We can see that, there is a large jitter when the epoch is about 100 in Figure 4(a). Figure 4(b) and Figure 4(c) have no large jitter after the epoch is about 60. The training loss of Figure 4(c) drops faster than Figure 4(a). The test accuracy of the three models is shown in Table 2. From Figure 4 we can see that, the training loss of the stepped sampler model still converges faster than the traditional sampler model.

In our experiments, most LSTM models with the stepped sampler have a more stable convergence of training loss, compared with the traditional LSTM models based on the same batch size. The stepped sampler LSTM may have a higher test accuracy than the traditional sampler LSTM, in the same batch size, which can reach the value of 0.639 (Table 2).Our test uses the shuffle operation, to make the test results more objective. Since the test data is shuffled, there should be less temporal information between the data, the test results may depend on the sampler and model only, which could reflect the detection effect well.

5 DISCUSSION

The experiment is to study the detection effect of the proposed sampler, which simulates one of the human brain memory law, repeating the input, to use the temporal information (we think it is the correlation in/between frames) of videos.

As the data are partly repeated inputted, it may be equivalent to the timely knowledge review of human brain, which strengthens the memory of the LSTM network, and reduces the information forgetting. LSTM can selectively memorize the temporal information, which is human-like.

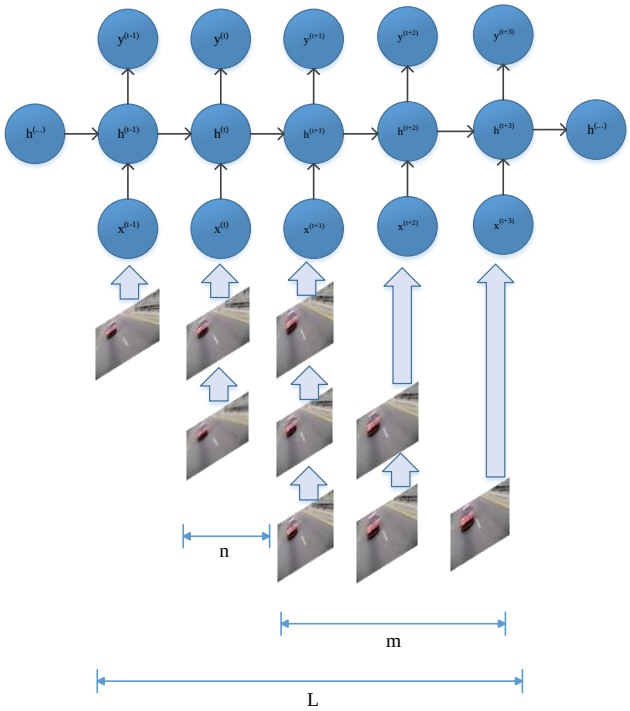

Figure 1: The illustration of the proposed stepped sampler. The upper half of the figure is the LSTM network, the lower half describes the workflow of the stepped sampler. The step size m is the size of the sub-batch, which is set to 3 in the figure, the step stride n is set to 1, the batch size L is set to 5 for the illustration. We operated the stepped sampling of the input data within each batch, since we want to strengthen the repetition. The car images are from Youtube-Objects dataset (Brox & Malik, 2010).

Algorithm 1: The stepped sampler

Input: Dataset, batch size L, step size m, step stride n
Output: Stepped sub-batch of the dataset
Initialize the dataset by Sequential sampler of PyTorch;
for $Batch = 1, 2, \cdots, |len(Batchsampler)|$ **do** `// use batch sampler to traverse all data`
  Initialize empty set step_batch[];
  for $idx = 1, 2, \cdots, L$ **do** `// traverse the elements in a batch of the batch sampler`
    output the idx-th item batch[idx] into step_batch[];
    $idx+ = 1$;
    if *len(step_batch[])* == *m* **then** `// when the size of step_batch reaches m`
      return step_batch[]; `// output the sub-batch`
      Reset step_batch[] to empty set;
      $idx = idx - m + n$; `// move the coordinate to the next sub-batch by stride n`
    end
  end
end

Figure 2: The video detection results of UCF101, which uses the proposed stepped sampler. Some frames of the videos are detected wrong, such as the frame of Line 1, Column 2, the groudtruth is BlowingCandles.

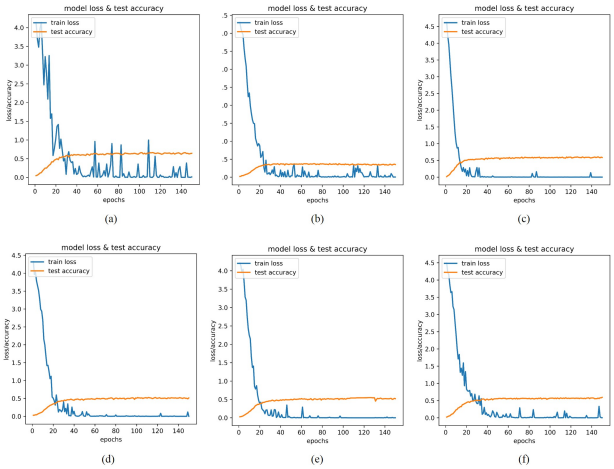

Figure 3: The training loss and test accuracy of the traditional sampler LSTM model and the stepped sampler LSTM model. The blue curves denote the training loss, and the yellow curves denote the test accuracy. The above six models are all with batch size 25. Subfigure (a) is a traditional LSTM model, which uses the batch sampler in PyTorch. Subfigure (b) is with the proposed stepped sampler, whose batch size is 25, step size is 20, step stride is 1. Subfigure (c), (d), (e), (f) are with the stepped sampler, whose batch size is 25, step size is 20, step stride is 2, 3, 4, 5, respectively. We want to study the better step stride when the batch size and step size are fixed.

Table 1: The test accuracy of the six models in Figure 3. BatchSampler denotes Figure 3(a), stride 1 stepped denotes Figure 3(b), stride 2 stepped denotes Figure 3(c), etc.

model	epoch 10	epoch 50	epoch 100	epoch 120	epoch 150
BatchSampler	0.289	0.621	0.650	0.642	0.635
stride 1 stepped	0.117	0.369	0.358	0.360	0.352
stride 2 stepped	0.307	0.566	0.577	0.602	0.587
stride 3 stepped	0.133	0.471	0.506	0.505	0.514
stride 4 stepped	0.170	0.502	0.506	0.546	0.521
stride 5 stepped	0.232	0.553	0.570	0.568	0.593

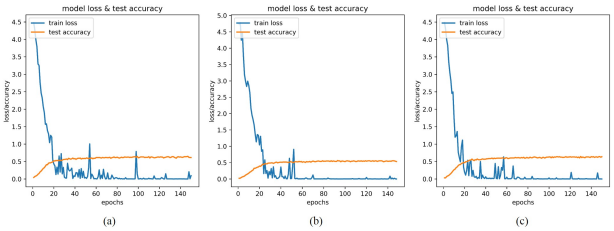

Figure 4: The training loss and the test accuracy of the traditional model and our stepped sampler model. The batch size of the three models are all 15. The blue curve denotes the training loss, and the yellow curve denotes the test accuracy. (a) is LSTM with the traditional sampler, batch sampler in PyTorch. (b) and (c) are LSTM with the proposed stepped sampler, with the stepped size 10, when the step stride of (b) is set to 3, and the step stride of (c) is set to 5. The training loss of (c) still converges faster than (a).

Table 2: The test accuracy of the three models in Figure 4. BatchSampler represents Figure 4(a), stride 3 stepped represents Figure 4(b), stride 5 stepped represents Figure 4(c).

model	epoch 10	epoch 50	epoch 100	epoch 120	epoch 150
BatchSampler	0.303	0.585	0.607	0.622	0.616
stride 3 stepped	0.178	0.517	0.532	0.544	0.535
stride 5 stepped	**0.282**	**0.593**	**0.631**	**0.639**	**0.637**

From Figure 3 we can see that, the repeating times of the stepped sampler is not the more the better, as shown in Figure 3(b), the stride is 1, and the convergence speed of the model is not improved much. In addition, the repeating times of the stepped sampler is not the less the better. As shown in Figure 3(f), the convergence speed of the model is even slower. The phenomenon is the same with human learning. Too much repetitive input and too little repetitive input would not improve the learning effect of human. The experiments seem to verify the similarity between machine learning and human learning. We assume that artificial neural networks have human-like characteristics.

Temporal information also seems to have human-like characteristics. Temporal information is the correlation of temporal sequences, analogous to human learning, it is the correlation of knowledge. In human learning, one of the important learning methods is to use the correlation of knowledge. Then, using the temporal information may also be one of the important learning methods of machine learning.

6 Conclusion

We refer to the human memory rules, and propose the stepped sampler, a repeating input method which uses the timely review approach. In our experiments, this method has a better promotion on the detection effect of LSTM. The experimental results show that, compared with the traditional sampler, the training loss of the stepped sampler converges faster, and is more stable after the convergence, i.e., there is no large jitter after convergence. The test accuracy of the model with the stepped sampler also reaches a high point faster and is more stable either. We analyzed the algorithm of stepped sampler and got several equations.

Another innovative point in the paper is that, we combine computer science with psychology, and use psychological methods to study artificial neural networks. The results show that, there may be a close relationship between biological neural network and artificial neural network, whatever in structure and even in principle. Artificial neural networks seem to have human-like characteristics, and human learning and machine learning seem to have similarities.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61773360.

REFERENCES

Thomas Bayes and David Hume. Bayes's theorem. In *Proceedings ofthe British Academy*, volume 113, pp. 91–109, 1763.

Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European conference on computer vision*, pp. 282–295. Springer, 2010.

Wen-Hsuan Chu, Yu-Jhe Li, Jing-Cheng Chang, and Yu-Chiang Frank Wang. Spot and learn: A maximum-entropy patch sampler for few-shot image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6251–6260, 2019.

Robert G Crowder. Intraserial repetition effects in immediate memory. *Journal of Verbal Learning and Verbal Behavior*, 7(2):446–451, 1968.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6): 721–741, 1984.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Tianyang Hu, Zixiang Chen, Hanxi Sun, Jincheng Bai, Mao Ye, and Guang Cheng. Stein neural sampler. *arXiv preprint arXiv:1810.03545*, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037, 2019.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Steven M Smith. Theoretical principles of context-dependent memory. *Theoretical aspects of memory*, 2:168–195, 1994.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2467–2475, 2018.