Analysis of Artificial Intelligence and Visual Analytics Techniques: the Case of t-SNE

YvXiang Dan
Sun Yat-sen University
danyx@mail2.sysu.edu.cn

Abstract

How to quickly and effectively mine valuable information from massive data to better guide decision-making is an important goal of big data analysis. Visual analytics is an important big data analytics method, which takes advantage of human visual perception, uses visual charts and graphs to intuitively present the laws embedded in complex data, and supports human-centred interactive data analysis. This article will take the t-SNE visualisation of classification models based on the CIFAR-10 dataset as an example to explore the visualisation techniques for artificial intelligence.

1 Introduction

With the rapid development of computer hardware and big data processing technology, the bottleneck of intelligent analysis of massive data has changed from 'how to quickly process massive data' to 'how to quickly and effectively mine valuable information from massive data'.[Luo Yuyu, 2024] Visualisation and visual analytics are based on human visual perception, combining data analysis and human-computer interaction technologies, and using visual diagrams to deconstruct the knowledge and laws embedded in complex data.

In the field of AI deep learning, model training is a necessary part. And in the training process, in order to more directly and effectively monitor the model's parameters, evaluation indicators and other information, we often need to visualise the training to help people better understand and adjust the training model.

In this paper, we complete the deep learning model training and testing process implemented on the CIFAR-10 training set, and use simple visualisation techniques such as TSNE[van der Maaten and Hinton, 2008] to demonstrate the quality of the model more intuitively.

2 Visualisation Technology

Visualisation technology can present complex data in the form of visual charts and graphs, using the characteristics of human visual perception to help users deconstruct the information contained in complex data. A good visualisation result has the characteristics of simplifying the complexity and making it clear at a glance, which can deconstruct the complex information contained in the data, efficiently convey the knowledge of the data, draw the readers' attention to the important characteristics of the data and empower the readers with the ability to see the different aspects of the data.

In a nutshell, visualisation refers to selecting appropriate data attributes from a given dataset, performing necessary data transformation operations, choosing appropriate visual coding methods, and finally rendering the visualisation results in a drawing.

In deep learning, taking the multi-classification task on CIFAR-10 as an example, the main contents stored in the trained model include model parameters such as weights and biases; model architecture such as layer types and layer configurations; and important parameters such as loss functions.[Hajibabaee *et al.*, 2021]

Obviously, the essence of a model is an ordered collection of data. Therefore, it is feasible to apply visualisation techniques to such models to assist researchers in monitoring the quality of model training.

3 Setup

3.1 Basic Properties of the CIFAR-10

CIFAR-10 is a training set of RGB images of real objects, each image in the collection has a size of 32x32 and there are 10 image categories with 6000 images in each category, of which there are 5000 training images and 1000 test images.[Krizhevsky and Hinton, 2009]

It is worth noting that most of the images in CIFAR-10 contain a considerable amount of noise, and there are large differences in the percentage of objects and features in the images under the same category.

3.2 Import and loading of CIFAR-10

In this post, for CIFAR-10, the keras.datasets method in TensorFlow will be used for loading it.In the model training process mentioned later in this article, we do not change the ratio of training and test images in CIFAR-10 in training, nor do we make any changes to the images in the dataset before putting them into training.

Parameters	Value
optimizer	adam
loss	sparse categorical crossentropy
metrics	sparse categorical accuracy
kwargs	default
target tensors	default

Table 1: Parameters in the compile

Parameters	Value
batch size	32 (default)
epochs	10
validation split	0.2

Table 2: Parameters in the fit

4 Basic Training of the Model

I used keras.Sequential() in TensorFlow to build the model and the base training of the model will be based on a four-layer neural network. The first layer of the network has two convolutional layers (16 3*3 convolutional kernels) and a pooling layer (2*2 convolutional kernels with maximum pooling method and ReLU activation function). The second layer has two convolutional layers (32 3*3 convolutional kernels) and a pooling layer (maximum pooling method 2*2 convolutional kernels, activation function ReLU). The third layer is an implicit layer and the activation function is the ReLU function. The fourth output layer is and the activation function is softmax.

4.1 Configuration of Learning Process

The learning process is configured using the modeling method compile in Sequential() with the following parameters (Table 1) in the configuration.

Role of Parameters: Set the object of the optimizer to adam. The loss function of the model is set to a sparse cross-entropy loss function. The calculation of the accuracy is set to the sparse classification accuracy function.

4.2 Parameters for Model Training

Model training is configured using the fit method in Sequential() with the following configuration parameters (Table 2) :

Role of Parameters: The number of objects in a single round of model training was set to 32, the number of training rounds was set to 10, and the images used for validation were 20% of the total number of images.

4.3 Results for Model Training

Based on the function algorithm chosen in the previous section, the loss function and accuracy of the model obtained from training can be calculated for both the training and test sets. With the help of the matplotlib.pyplot library, we can write this data in 2D coordinates and present it as a graph.

We can see that the basic indicators of the model are as follows in Figure 1:

In the image, the horizontal coordinate indicates the number of training rounds and the vertical coordinate indicates the loss function value or accuracy of the model. It is not difficult to find that for the training set, as the number of rounds increases, the loss function of the model becomes steadily smaller and the accuracy steadily increases; while for the validation set, the loss function of the model becomes smaller and then larger, and the accuracy reaches a certain value and then stops growing.

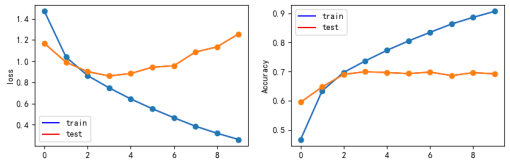

Figure 1: Loss function and accuracy for both the training and test sets

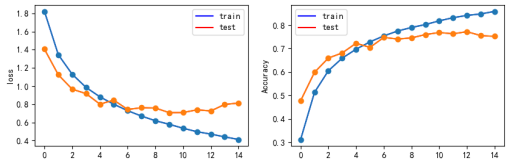

Figure 2: Loss function and accuracy of the model after optimizing the neural network

5 Analysis of Training Results and Improvement of the Training Process

The accuracy of the model on the validation set rises to about 70% and then no longer rises steadily with the number of training rounds, but instead shows a small drop in accuracy in one round. This is due to "overfitting", i.e., too many training rounds of the model causing it to perform well on samples that have already been encountered and poorly on samples that have not been seen. Therefore, it is necessary to try to improve the model training process.

5.1 Deepening and Thickening Neural Networks

I tried to thicken and deepen the neural network for training. The two convolutional layers of the first layer of the network are changed from 16 3*3 convolutional kernels to 32. The two convolutional layers of the second layer are changed from 32 to 64 3*3 convolutional kernels. A convolutional layer containing 128 3*3 convolutional kernels and a pooling layer identical to the first two layers are introduced for the third layer. The original third is merged into the fourth layer.

The training was performed again with batch size of 32, 15 iterations and test set ratio of 0.2, and the following results were obtained in Figure 2.

Apparently, the model's accuracy on the test set reaches its highest value during the 11th round of training, a value that is about 5-7% larger than the highest value for the model in the previous passages.

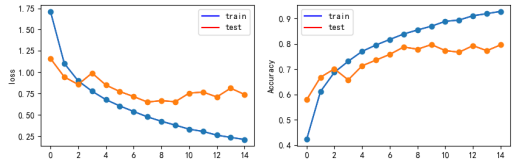

Figure 3: Loss function and model accuracy after adding batch normalization layers

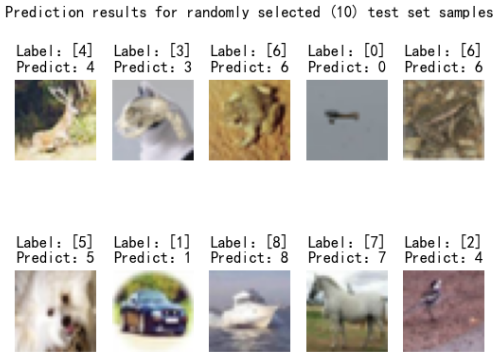

Figure 4: Visualization based on random image judgment

5.2 Adding Batch Normalization Layers

Batch normalization is a commonly used technique to speed up the training process of neural networks and improve the stability of the model. It reduces the problem of internal covariate shift by normalizing the inputs of each layer, making the distribution of inputs in each layer more stable.

After adding the batch normalization layer after a few convolutional layers, the training is performed again with the same parameters and the results are shown in Figure 3:

With the addition of the batch normalization layer, the peak accuracy of the model on the test set is again about 5% higher compared to the model in Figure 2.

6 Visualization of Training Results

In terms of accuracy alone, the training quality of the model was improved by about 14.2% by optimizing the neural network, performing batch normalization and other methods. However, this does not give an intuitive sense of the gap between them, and so the visualization of the model is important.

6.1 A Visualization Method for Randomly Sampling Test Sets

After training, 10 images are randomly selected from the 10,000 images in the test set, and the model is allowed to judge them again, and both the judgment results and the image label values are printed on the images. One can visualize the quality of the model from these results, as in Figure 4:

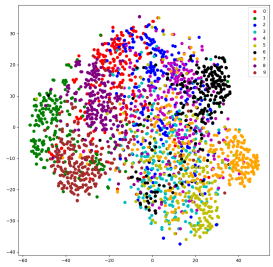

Figure 5: Visualization of the model shown in Figure 1

Such a visualization is certainly straightforward and simple, yet it is like a continuation of model training that is deceiving itself.

6.2 TSNE Visualization Technology

TSNE, also known as t-SNE, full name is t-distributed Stochastic Neighbor Embedding. It is a dimensionality reduction technique used to represent a high-dimensional dataset in a two-dimensional or three-dimensional low-dimensional space and retain the local characteristics of the dataset (e.g., the relative distances between the two pieces of data) so as to achieve the visualisation of the high-dimensional dataset.[Laurens and Hinton, 2008]

t-SNE can be regarded as one of the most effective data reduction and visualisation methods at present. When we want to classify a high-dimensional dataset, but it is not clear whether this dataset has good separability, we can project the data into 2-dimensional or 3-dimensional space and observe it through t-SNE: if it has separability in the low-dimensional space, the data is separable; if it is indivisible in the low-dimensional space, it is probably If it is not separable in low dimensional space, it may be because the dataset itself is not separable, or the data in the dataset is not suitable for projection into low dimensional space.

Therefore, we can use the TSNE technique to load random images and trained models in the training set, and for the high-dimensional vectors obtained from the model judgement, find out the pairwise similarity between neighbouring points in the high-dimensional space, and map each point in the high-dimensional space to the low-dimensional mapping based on the pairwise similarity of the points in the high-dimensional space. This is to reduce the high-dimensional data to a two-dimensional image that can be easily recognised by humans.

For the models in Fig. 1, Fig. 2 and Fig. 3, we randomly select 2500 unduplicated validation images from 10000 validation images in CIFAR-10 for judgement and dimensionality reduction respectively, and the results are shown in Fig. 5, Fig. 6 and Fig. 7:

After visualization, it is easy to see that the higher quality model in terms of accuracy corresponds to a more focused and neat image. Such a result corresponds to the higher dimensional space is the more distinct features between points, which is in line with our goal of training multi-classification models.

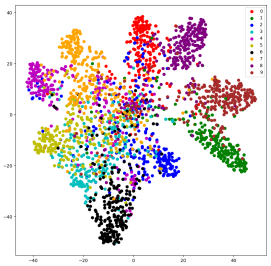

Figure 6: Visualization of the model shown in Figure 2

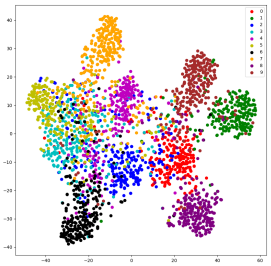

Figure 7: Visualization of the model shown in Figure 3

7 Summaries

As can be seen from the simple experiments in this post, visualisation techniques (e.g., t-SNE) play an important role in AI deep learning by helping us to understand the behaviour of the model, evaluate the model performance, debug and optimise the model, and provide intuitive tools for education and dissemination.

In the future, with the improvement of algorithmic efficiency, the development of interactive tools, and the combination with other technologies (e.g., interpretable research, VR/AR, etc.), we expect that visualisation technologies will play a greater role in deep learning, and promote the transparency, interpretability, and popularity of AI.

References

[Hajibabaee *et al.*, 2021] Parisa Hajibabaee, Farhad Pourkamali-Anaraki, and Mohammad Amin Hariri-Ardebili. An empirical evaluation of the t-sne algorithm for data visualization in structural engineering. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1674–1680, 2021.

[Krizhevsky and Hinton, 2009] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4), 2009.

[Laurens and Hinton, 2008] Van Der Maaten Laurens and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2605):2579–2605, 2008.

[Luo Yuyu, 2024] Xie Yupeng Li Guoliang Luo Yuyu, Qin Xuedi. Overview of intelligent data visual analytics. *Journal of Software*, 35(1):356, 01 2024.

[van der Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.