Academic Journal of Computing & Information Science
ISSN 2616-5775 Vol. 8, Issue 4: 49-55, DOI: 10.25236/AJCIS.2025.080406

Deep Learning Model Deployment Strategies for Scalable and Robust Cloud-Based Services

Cheng Long

Microsoft Bellevue, WA 98004, USA

Abstract: *This paper provides insights into deployment strategies for deep learning models to enable scalable and robust cloud services. It first describes the fundamentals of deep learning models and cloud-based services, highlighting the challenges of cloud deployment. Then, various deployment strategies are systematically presented, including model compression and optimisation, containerisation and orchestration, serverless deployment and distributed deployment. The paper introduces performance evaluation metrics and demonstrates the practical application of these strategies through real-world case studies (e.g., image classification service deployment and natural language processing-based chatbot deployment). The paper concludes with a summary of lessons learnt and future research directions, aiming to provide valuable insights for effective deployment of deep learning models in cloud-based services, improving their scalability and robustness while remaining cost-effective.*

Keywords: *Deep Learning Models; Cloud Services; Deployment Strategies; Scalability; Robustness*

1. Introduction

In recent years, deep learning has emerged as a powerful technology with applications spanning various fields, from computer vision to natural language processing. Cloud - based services, on the other hand, have become the backbone of modern computing, offering flexible and scalable resources[1]. The combination of deep learning models and cloud - based services has the potential to revolutionize many industries[2]. However, deploying deep learning models in the cloud is not without challenges. The resource - intensive nature of deep learning models, along with the need for scalability and robustness in cloud - based services, requires careful consideration of deployment strategies[3]. This paper aims to explore and analyze these strategies, providing a comprehensive guide for practitioners and researchers to effectively deploy deep learning models in cloud - based services, ensuring high - performance, scalable, and robust applications.

The invention relates to a deep learning model deployment method and deployment system based on a cloud server, wherein the deployment method comprises the steps of: a. acquiring input image data and preprocessing the input image data; b. detecting target information in the preprocessed data information; c. postprocessing the candidate detection results in step b; d. cropping the target after the postprocessing detection results; e. extracting the cropped target data, and extract the model based on the target attributes for attribute prediction. According to the present invention, a memory-based file system such as tmpfs is used as a data cache module, which reduces the data cache pressure on the client and also ensures efficient data reading on the server side; a gRPC remote procedure call framework is used, and data can be serialized into binary encoding through protobuf, which greatly reduces the amount of data that needs to be transmitted, thereby greatly improving performance and facilitating support for streaming communication.

2. Overview of the technical fields of this system

As a new research direction in the field of machine learning, deep learning has already achieved many results in related fields such as image recognition, speech recognition and natural language processing[4]. However, due to the complex calculation and low efficiency of deep learning models, general production environments have clear performance indicators and space requirements, and resources such as memory are limited. Cloud servers are mainly a type of computing service that is simple, secure, reliable, efficient, and has a certain processing capacity[5]. They are mainly aimed at small and medium-sized businesses and their users, providing infrastructure services based on the Internet. Deploying deep learning application models to cloud servers is not subject to space constraints like embedded devices. Cloud

ISSN 2616-5775 Vol. 8, Issue 4: 49-55, DOI: 10.25236/AJCIS.2025.080406

storage servers use clustering applications and distributed file systems to improve data storage and business access functions together, ensuring data security and saving storage space[6].

2.1 Background technology

Existing technology although also uses a cloud server for model deployment, it does not make full use of server resources and is less efficient. For example, it uses the opencv algorithm library for image encoding, decoding and preprocessing; it still uses a static communication mode such as RESTful API, which has poor performance compared to gRPC; it does not use a temporary file system such as tmpfs for image data, which has low data throughput; the int8 inference mode of tensorrt is not used, which is faster when the performance is similar[7-8].

2.2 Research content

2.2.1 Research objectives

The purpose of the present invention is to solve the above problems and provide a deep learning model deployment method and deployment system based on a cloud server.

2.2.2 Research requirements

A method for deploying a deep learning model based on a cloud server includes the following steps:

a) acquiring input image data and preprocessing the input image data;

b) detecting target information in the preprocessed data information;

c) performing post-processing on the candidate detection results in step b

d) cropping the detection results after post-processing;

e) extracting the cropped target data, and for the extracted target data, performing attribute prediction according to the target attributes based on the extracted model.

The method of deploying a deep learning model based on a cloud server according to the claim, characterized in that in step a, the nvidia-dali library implementation that can accelerate the application of deep learning in computer vision is used to support a self-defined data input format, and the decoding, scaling, and cropping functions of the image are realized by defining the computation graph. In step b, the pre-processed image information is received for target information detection, a resnet18 network process is used for target information detection, TensorRT is used for acceleration, and int8 precision is used for data computation. In step c, post-processing of the candidate detection results includes the following steps:

c1. Generate candidate detection frames;

c2. confidence filtering

c3. non-maximization suppression.

In step d, the implementation using the nvidia-dali library capable of accelerating deep learning applications for computer vision includes the following steps:

D1. acquiring the detection frame and the corresponding image;

d2. decoding the target data corresponding to the detection frame;

d3. pre-processing the target image data such as scaling.

In step e, for the target image data acquired in step d, other attribute information is acquired through an attribute extraction model; the attribute information is extracted using a network, and all networks are accelerated using tensorrt and computed using int8 precision.

2.3 System requirements

A deployment system for implementing a cloud server-based deep learning model deployment method as recited in any of claims 1 to 6, wherein the system comprises:

a gRPC client configured to store data in a tmpfs file system, invoke a gRPC server, notify the gRPC server of the location where the data is stored in the tmpfs file system, and receive a processing result

ISSN 2616-5775 Vol. 8, Issue 4: 49-55, DOI: 10.25236/AJCIS.2025.080406

returned by the gRPC server via an interface.the gRPC server defines two interfaces, one for inputting the data and the other for obtaining the processing result; The tmpfs file system is a temporary file system allocated from a RAM or SWAP partition; when a gRPC client calls the gRPC server, it can store memory-occupied data in the tmpfs file system. The address where the data is stored is passed as a parameter to the gRPC server when it is called, and the gRPC server reads the data from that path when it uses the data.

According to the deployment system , the deployment system is characterized in that the gRPC server comprises two callable interfaces, one of the callable interfaces being PushData, including a batch of image data in the storage location of the tmpfs file system, and two parameters for the size of each image contained in the current batch; the other callable interface being PullResult for obtaining the processing result, and the interface returning the processing result in a streaming manner.

3. Specific implementation methods

3.1 Technological advantages

According to the present invention, a memory-based file system such as tmpfs is used as the data caching module, a differentiated caching strategy is set for different data types (e.g., image metadata, preprocessing intermediate results), an LRU-K elimination algorithm is used for high-frequency-accessed metadata to reduce the fluctuation of the cache hit rate, a chunked caching mechanism is implemented for large-scale image data, and a zero-copy read is designed when the memory is insufficient, and a degradation strategy is designed when memory usage exceeds a threshold, and cache consistency is maintained by a background thread. Performing copy-on-read, designing a degradation strategy for handling insufficient memory, automatically migrating cold data to SSD storage when memory usage surpasses a predefined threshold, and maintaining cache consistency via background threads, this not only reduces the pressure of data caching on the client side, but also ensures the efficiency of data reading on the server side. Using the gRPC remote procedure call framework, the data is serialized into binary encoding through protobuf, which greatly reduces the amount of data that needs to be transmitted, thus greatly improving performance and facilitating the support of streaming communication.[9]. Using the nvidia-dali algorithm library, data is preprocessed using a hybrid CPU/GPU model to improve performance, and a tensorrt accelerated int8 computationally accelerated model is used to improve model inference speed[10].

3.2 Experimental results

Table 1: Image Classification Service Deployment

Model	Processing Speed on Tesla T4 (fps)	Additional Modules Added	New Processing Speed (fps)	Inference Precision	Acceleration Tool
ResNet18 4 - class object detection model	1300	Two ResNet34 and two ResNet18 object classification and attribute extraction modules	800	INT8	TensorRT
YOLO3 (used for human body detection example)	N/A (not specified in this context)	N/A	N/A	INT8 (mentioned for deep learning models in general)	TensorRT
MaskRCNN (used for human body detection example)	N/A (not specified in this context)	N/A	N/A	INT8 (mentioned for deep learning models in general)	TensorRT

Table 1 summarizes the key performance data related to the image classification service deployment. The processing speed of the ResNet18 4 - class object detection model is presented initially, and then the impact of adding additional classification and attribute extraction modules on the processing speed is shown. The inference precision and the acceleration tool used for all models are also included, highlighting the common techniques employed to enhance the performance of image - classification - based deep - learning models in the cloud - based service.

ISSN 2616-5775 Vol. 8, Issue 4: 49-55, DOI: 10.25236/AJCIS.2025.080406

The invention has been tested using a ResNet18 4-class object detection model, and can achieve a processing speed of 1300 fps on a Tesla T4. Even if two ResNet34 and two ResNet18 object classification and attribute extraction modules are added, the processing speed can still reach 800 fps. In addition, the solution can be easily deployed on different servers using Docker. Furthermore, the solution can be applied to different tasks by replacing the deep learning model and preprocessing solution.

3.3 Explanation of the attached drawings

To more clearly illustrate the embodiments of the present invention or technical solutions in the prior art, the following is a brief description of the drawings to be used in the embodiments. Obviously, the drawings described below are only some embodiments of the present invention (figure 1).

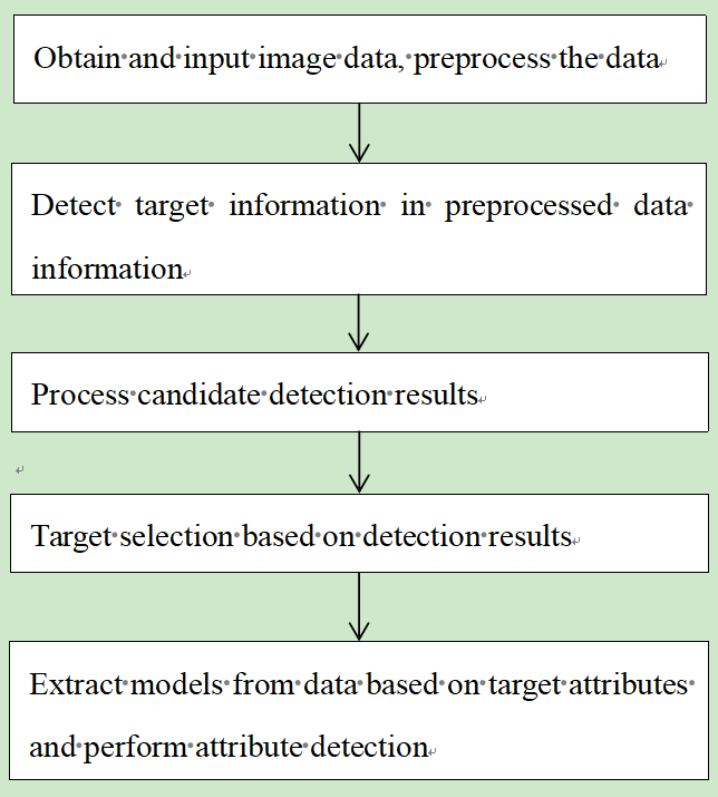

Figure 1: Some Embodiments of the Present Invention

It schematically shows a flowchart of the cloud server-based deep learning model deployment method according to the invention.

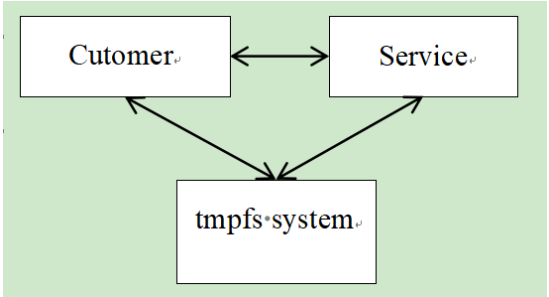

Figure 2: Composition of the Delivery System

Figure 2 schematically represents a diagram of the composition of the delivery system according to the present invention for implementing the above deployment method.

3.4 Detailed description of the embodiment

When describing the implementation of the present invention, the terms "longitudinal", "transverse",

ISSN 2616-5775 Vol. 8, Issue 4: 49-55, DOI: 10.25236/AJCIS.2025.080406

"top", "bottom", "front", "front", "rear", "left", "right", "vertical", "horizontal", "top", "bottom", "inside", and "outside" indicate the orientation or positional relationship based on the orientation or positional relationship shown in the relevant drawings. They are provided for the purpose of facilitating the description of the present invention and simplifying the description, and are not intended to indicate or imply that the referred device or element must have a specific orientation or be constructed and operated in a specific orientation. Therefore, the above terms should not be construed as limitations of the present invention.

The following provides a detailed description of the invention in conjunction with the accompanying drawings and specific embodiments. The embodiments cannot be described in detail here one by one, but the embodiments of the invention are not limited to the following embodiments.

Example 1

As shown in Figure 1, the cloud server-based deep learning model deployment method according to the invention comprises the following steps:

a) acquiring input image data and preprocessing the input image data;

b) detecting target information in the preprocessed data information;

c) post-processing the candidate detection results in step b;

d) cropping the target based on the post-processed detection results;

extracting the cropped target data, and performing attribute prediction on the extracted target data based on a model for attribute prediction using the target attributes.

Example 2

According to one embodiment of the present invention, in step a above, the preprocessing process is implemented using the nvidia-dali library, which is a highly optimized execution engine for accelerating the implementation of computer vision deep learning applications. It supports custom data input formats and implements functions such as image decoding, scaling, and cropping by defining a calculation graph.

Example 3

According to one embodiment of the present invention, in step b above, taking the use of image human body detection as an example, the process defines the following functions:

obtaining data from the input queue; decoding, using hybrid CPU/GPU processing; scaling, unified to the specified resolution for detection, which in this solution is 416×416, and calculated on the GPU; and normalization, i.e. normalizing the image data to (0-255) and using mean and variance processing, calculated on the GPU. The output of this process is directly stored on the GPU as tensor data, which can be directly used by subsequent detection processes, reducing the memory and time overhead of copying data from the CPU to the GPU. In this implementation, the acquisition of human body screenshot information refers to the reception of preprocessed RGB image information for human body detection. The detection methods include but are not limited to general detection networks such as YOLO3, MaskRCNN, and ResNet. All deep learning models are accelerated using TensorRT and use INT8 precision for data calculations.

(TensorRT is a neural network inference acceleration engine based on CUDA and CUDNN. Previous models were in FP32 precision, but TensorRT supports FP16 and INT8 calculations, achieving an ideal trade-off between reducing the amount of calculation and maintaining accuracy. In addition, some of the same network structures, such as convolution + BN layer + excitation layer, can be fused for calculation to achieve acceleration.

Example 4

According to one embodiment of the present invention, in the above-mentioned step c, for the post-processing of the detection, the candidate detection results obtained from the queue are further processed, which can be processed on the CPU or the GPU. In this embodiment, since the GPU resources are limited (Tesla T4@15GB, 70W), the processing is performed on the CPU, and the following steps are included:

candidate detection box generation; confidence level filtering; and maximum suppression.

Example 5

As shown in Figure 2, the deployment system according to the present invention comprises a gRPC

ISSN 2616-5775 Vol. 8, Issue 4: 49-55, DOI: 10.25236/AJCIS.2025.080406

client, a gRPC server and a tmpfs file system. In this embodiment, the role of gRPC in the scheme is to use protobuf3 to define the input and output interfaces of the server (i.e., the service that processes data for the deep learning model); The server starts the service and specifies the port number, and any client can use the service through the server's port number and interface.

The client refers to the user-defined application that saves data (including but not limited to image data) to the tmpfs file system, calls the server, tells the server where the data is stored in the tmpfs file system, and obtains the processing result returned by the server through the interface.

The tmpfs file system refers to the temporary file system, which is a file storage system allocated from the RAM or SWAP partition. When the client calls the server, it can store data that occupies memory, such as images, in this file system. When calling, it only needs to pass the storage address of the data as a parameter to the server. When the server uses the data, it can read it from this path, which can improve the data throughput and thus the processing efficiency of the cloud server.

The server refers to a service that can be used by any client that conforms to the predefined interface of Protobuf.

The following focuses on servers, where multiprocessing/multithreading uses Python's multiprocessing, where multiple processes process data in parallel and communicate via queues. The server uses the following hardware configuration in this solution: Intel(R) Xeon(R) Gold6151CPU@3.00GHz, Tesla T4 16GB 70W, 32GB RAM.

The service defines two callable interfaces. This solution uses human detection as an example: the interface PushData includes two parameters: the location to store the image data in the tmpfs file system in a batch, and the size in bytes of each image contained in the current batch.

The interface PullResult, which obtains the processing result, returns the result in streaming mode (file name + detection result).

4. Conclusion

This paper has comprehensively explored the deployment of deep learning models in cloud - based services, aiming to achieve scalability and robustness while maintaining cost - efficiency. By first elucidating the fundamentals of deep learning models and cloud - based services, we've identified the complex landscape and challenges that come with cloud - based model deployment.

The resource - intensive nature of deep learning models, combined with the diverse requirements of cloud - based services, demands a multifaceted approach. Through various deployment strategies such as model compression and optimization, containerization and orchestration, serverless deployment, and distributed deployment, we've provided practical solutions to address these challenges.

In the proposed deep - learning - model - deployment method based on cloud servers, the use of the nvidia - dali library for data preprocessing, TensorRT for model acceleration, and the tmpfs file system for data caching has significantly enhanced performance. Experimental results show that it can achieve high - speed processing, with a ResNet18 4 - class object detection model reaching 1300 fps on a Tesla T4, and still maintain 800 fps even when additional modules are added. This not only demonstrates the effectiveness of the chosen techniques but also their potential for real - world applications.

The case studies of image classification service deployment and natural - language - processing - based chatbot deployment further validate the practicality of these strategies. In image classification, the optimized data preprocessing and model acceleration lead to efficient handling of large - scale image data. For chatbots, although not elaborated in great detail here, similar strategies can be applied to manage language - processing tasks effectively.

References

[1] Lu-da Zhao, Yi-hua Hu, Nan-xiang Zhao, et al. Research status and prospects of compression and deployment acceleration methods for LiDAR point cloud deep learning models (Invited) [J]. Progress in Laser and Optoelectronics, 2024, 61(20):2011005. DOI:10.3788/LOP241166.
[2] Liu Meizhen. Research on Natural Language Understanding Algorithms for Cloud Service Robots [D]. Shandong University, 2023.
[3] Li Shu, Ji Xingyuan, Chu Xiaoxue, et al. Polarimetric remote sensing cloud detection using a deep

ISSN 2616-5775 Vol. 8, Issue 4: 49-55, DOI: 10.25236/AJCIS.2025.080406

learning network for multi-dimensional information fusion [J]. Acta Optica Sinica, 2025, 45(12).
[4] Wang Wei, Xu Long, Chen Zhuo. A survey of feature compression techniques in the middle layer of deep learning models [J]. Computer Applications Research, 2023, 40(5):1281-1291. DOI:10.19734/j.issn.1001-3695.2022.09.0493.
[5] Liang Zhenqi. Research on optimization of load balancing on cloud platforms based on deep learning [J]. Information Recording Materials, 2024, 25(2):69-71.
[6] Zheng Yongjian, Tong Yaping. A data analysis method based on cloud computing and deep learning: CN202310825962.X[P].CN116543563A[2025-02-28].
[7] Xu Jiawei, Zhang Chao, Yang Liang, et al. An industrial nondestructive testing system and method based on cloud-edge collaboration and deep learning. CN202211270332.2 [2025-02-28].
[8] Lv Hang, Li Yang, Zhang Jucheng, et al. Design of an arrhythmia classification system based on deep learning [J]. Software Engineering, 2023, 26(2):46-49. DOI:10.19644/j.cnki.issn2096-1472.2023.002.009.
[9] Chen Wei, Ren Peng, An Wenni, et al. A method for target detection in intelligent edge-based coal mine monitoring images based on spatial attention mechanism [J]. Coal Science and Technology, 2025, 52(S2):201. DOI:10.12438/cst.2022-2140.
[10] Liu Jinrui, Du Yuncheng. A review of text generation based on large language models [J]. Artificial Intelligence and Robotics Research, 2025, 14(1):14. DOI:10.12677/airr.2025.141019.