Deploying Person Retrieval System on Edge Devices

Meet Rathi *B.Tech CSE* *SEAS, AU* AU2240106

Aditya Agrawal *B.Tech CSE* *SEAS, AU* AU2240153

Harsh Panchal *B.Tech CSE* *SEAS, AU* AU2240160

***Abstract*—The text-based person retrieval abilities of multi-modal CLIP models face deployment barriers when running them on reduced-resource edge devices like Nvidia Jetson Orin AGX due to performance time and ability constraints. The paper introduces a modified system for real-time person retrieval based on CLIP which uses the Person Retrieval System dataset. The system applies a ViT-B/32 Vision Transformer encoder together with LoRA fine-tuning and FP16 precision to optimize efficiency. The deployment using TensorRT and CUDA together with ONNX-based deployment and model pruning and quantization speeds up inference operations. The optimized model achieves performance tests including retrieval accuracy and inference speed and resource utilization which show excellent results suitable for edge-based applications within intelligent surveillance systems.**

***Index Terms*—CLIP model, Low-Rank Adaptation (LoRA), Nvidia Jetson Orin AGX, real-time person retrieval, edge computing, model optimization, inference acceleration, resource-constrained devices, TensorRT, CUDA, model quantization, deep learning.**

I. Introduction

Image retrieval, a pivotal component of computer vision, underpins applications such as real-time identity verification, intelligent surveillance, and content-based image search. Using the CLIP retrieval model with Nvidia Jetson Orin AGX edge systems brings multiple challenges because these edge devices experience computational complexity in addition to limited power usage and high inference speed and constrained memory availability. This work describes an optimized version of CLIP-based person retrieval specifically designed for edge systems through the implementation of LoRA as well as model pruning and quantization and FP16 precision for improved performance without sacrificing accuracy. The system utilizes DistilBERT and ViT-B/32 components inside CLIP for optimally processing 54,522 image-caption pairs from the PRS dataset and generates text-based person search capabilities suitable for real-time IoT and surveillance systems.

The research produces three main results. The first contribution involves optimizing the lightweight CLIP framework using ViT-B/32 together with DistilBERT to perform efficient embedding alignment while maintaining high retrieval precision and lowering the computational load. The research combines LoRA parameter-efficient fine-tuning techniques with pruning and quantization and FP16/INT8 precision to optimize memory performance and speed and achieves better results on the PRS dataset than baseline CLIP implementations. Additionally it offers practical guidelines for developing person retrieval systems that run in real time on edge hardware by handling problems associated with TensorRT compatibility, dynamic resource scheduling and I/O latency optimization thus enabling the expansion and operational readiness of edge-based computer vision approaches.

Identify applicable funding agency here. If none, delete this.

II. Methodology

The development and deployment of a CLIP-based person retrieval system for edge devices with Nvidia Jetson Orin AGX as a specific focus is explained in this section. The approach for building this person retrieval system targets the PRS dataset according to six sequential steps which begin with preprocessing and end with inference. The methodology presents an explanation of theoretical elements and implementation reasons for each stage.

A. Preprocessing

The PRS dataset receives preprocessing treatment to achieve model compatibility and reduce training calculations. The following is used:

1) **Dataset Loading:** The PRS dataset consisting of 54,522 image-caption pairs is loaded from JSON format. The normalization process turns both images and captions into standardized formats which removes the necessity of executing additional preprocessing methods. The person retrieval task benefits from data handling optimization when this method is used.
2) **Data Storage:** The processed data finds its storage as NumPy arrays which contain image file paths and tokenized captions together with unique identifiers. The chosen data system format improves loading speed and uses minimal memory resources on appliances operating at device edges.

B. Dataset Preparation

To develop ICFG-PEDES we built an extensive preparation process for an image-caption pair collection designed for person re-identification tasks. Our data preparation process involved converting the JSON data to CSV format to manage data efficiently with standardized image path normalization

through prefix removal and path validity checks resulting in logged invalid path records for data quality maintenance. The image size was standardized to 224x224 pixels while normalization was applied to fulfill model requirements. A prompt of "A photo of a person" was used to improve caption context before performing tokenization using transformer-based tokenization methods and fixing all tokens to a uniform length. The 80-20 training-validation split method was used to create separate sets from the data which enabled model development alongside performance assessment thus establishing a strong data foundation.

C. Model Architecture

The developed CLIP model architecture proved its value by adapting to both experimental research requirements and practical deployment conditions. As our initial method we combined a Vision Transformer (ViT) as the image encoder for visual embedding compactness and a transformer-based text encoder for tokenized caption embedding production. When conducting research we analyzed multiple model configurations through the combination of ResNet50 for image encoding and DistilBERT for text encoding for experimental purposes. The project employed linear transformations coupled with normalization in projection layers but we also conducted experiments with additional projection heads that used activation functions and dropout to boost flexibility in research settings.

We utilized contrastive loss for image-text alignment because it produces high similarity values between matched pairs together with low similarities between non-matched pairs. Contrastive loss combines the total losses between image-to-text cross-entropy and text-to-image cross-entropy calculations. A batch containing N image-text pairs has normalizations for image embeddings z_i and text embeddings z_t. We compute the similarity matrix through cosine similarity operations $s_{ij} = \frac{z_i \cdot z_t}{\|z_i\| \|z_t\|}$ that use a temperature parameter τ. The contrastive loss consists of two parts:

The training objective contains half coefficient-weighted sum of image-to-text and text-to-image cross-entropy values, where:

$$\mathcal{L}_{\text{image-to-text}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(s_{ii}/\tau)}{\sum_{j=1}^{N} \exp(s_{ij}/\tau)} \tag{1}$$

$$\mathcal{L}_{\text{text-to-image}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(s_{ii}/\tau)}{\sum_{j=1}^{N} \exp(s_{ji}/\tau)} \tag{2}$$

The contrastive loss invokes an average between two cross-entropy losses that pair image embeddings to text embeddings followed by text embeddings to image embeddings. The cosine similarity of embeddings operates through a temperature-based scale that enables the flexibility of distribution behavior when performing retrieval operations in different contexts.

D. Model Configuration

The model required specific modifications to reach its maximum efficiency and obtain optimal processing parameters which would meet various application requirements. Memory management required moderate batch training at first followed by sizing down the batch size during inference to improve edge performance. AdamW served as the optimization method because we tested both uniform and non-uniform learning rates among the model components with weight decay added as an overfitting control mechanism. The training duration remained short since it reached maximum one epoch to optimize extractable pre-training information. During GPU unavailability the system operated with CPU backups at the bottom priority yet GPU received priority status for operations. FP16 mixed precision training operated as our primary computational method because it reduced resource utilization while accelerating computation across diverse resource settings.

E. Training

The training process of CLIP model involved ICFG-PEDES dataset alongside contrastive loss optimization for person re-identification which enabled image and text embedding alignment. Image-text pairs were processed in batches so we adjusted weight parameters according to loss value calculations for improved training efficiency through mixed precision. Our tests involved proof-of-concept learning rate setups between model components but we conducted other experiments with a uniformed rate. Progress monitoring involved examining validation performance yet we decided to preserve the model through state dictionary which used either the final model or the model with minimum validation loss depending on application requirements. The method's correct implementation enabled the model to develop proficient capabilities for locating particular images from text demands.

F. Inference

A text query to image retrieval inference process was developed for research evaluation along with real-time utilization. We derived image embeddings from validation data in advance then stored them for efficient query execution. The system processed text input by tokenization followed by embedding computation and cosine similarity with images which produced matching results in a limited number of output items. Research evaluation metrics included Recall@K (K=1, 5, 10) together with Mean Average Precision (mAP) as we displayed retrieved images in grids for qualitative research assessments. The system focused both on precision for accurate retrieval and practical usability needs for deployment requirements.

III. Model Optimization and Deployment

An implementation of optimization methods and deployment protocols enabled efficient operation for the CLIP-based person retrieval system which ran on Nvidia Jetson Orin AGX edge hardware. This part details optimization approaches using LoRA and model pruning alongside an explanation of the deployment method which starts with exporting the model in ONNX format then converts it to a TensorRT engine before running inference on the edge device. The implementation strategy achieves real-time performance for hardware systems by utilizing these deployment steps.

A. Model Optimization with LoRA

To fine-tune CLIP model for person retrieval on the Nvidia Jetson Orin AGX, Low-Rank Adaptation (LoRA) was used. In LoRA, we decompose weight update into low rank matrices and save trainable parameters via this decomposition. The model is configured with rank of 16 and trained on the dataset of PRS dataset which improves the retrieval accuracy with lesser memory used. This is a good approach for resource constrained edge devices: trade expressiveness of model for performance. By applying lightweight fine-tuning, LoRA ensures that its robust deployment is suitable for real applications like intelligent surveillance with resource allocation.

B. Model Pruning

Additional optimization of CLIP model occurred through structured pruning which reduced network weight redundancy and decreased the model's memory footprint and computational requirements. Weight pruning used magnitude thresholds to make weights with minor values zero. Accurate calibration methods were applied during this procedure to protect the efficiency of real-time person retrieval in the condensed model. The pruned model served as the basis for deployment steps because it resulted in lower system latency and resource requirements on the Jetson Orin AGX platform.

C. Exporting ONNX

The optimized CLIP model received ONNX export processing for achieving interoperability across multiple inference platforms, enabling efficient deployment on the Jetson Orin AGX. Using `torch_onnx_export` from PyTorch, the PyTorch JIT-scripted model `converted_model.pt` was converted into the ONNX file `vit_onnx`. The input image shape and size were defined by the dimensions (1, 3, 224, 224), as the CLIP model processes single RGB images resized to 224 x 224 pixels. Dynamic axes defined for the batch dimension ensure the model handles dynamic runtime workload data, which is vital for real-time applications with varying patterns of incoming data.

The ONNX export utilized opset version 12 and supported multiple operators through it to provide compatibility with TensorRT and various operator modes. Constant folding was implemented in the model to optimize the graph structure by performing static operation computations ahead of time which reduced execution overhead during the inference operation. The ONNX export process entailed thorough validation checks to confirm the ONNX model remained faithful to the source PyTorch model through the utilization of ONNX checker for recognizing graph inconsistencies. Prior to running inference on the Jetson Orin AGX the preventive resource-constrained device error protocols were essential for this step. The ONNX model created from CLIP acts as a portable framework-independent representation which secures simple integration with Nvidia's TensorRT framework for optimal deployment on edge systems.

D. Converting to TensorRT Engine

The ONNX model became a high-performance TensorRT engine (vit.engine) through the Nvidia TensorRT library for effective Nvidia Jetson Orin AGX inference. The hardware-specific optimizing techniques in TensorRT provide layer fusion alongside precision calibration and automatic kernel tuning which make it optimized for Nvidia systems. Wedetermined a 256 MB memory pool restriction during ONNX conversion because edge devices have limited resources. A TensorRT builder used an ONNX parser to construct a network which contained optimization parameters allowing variable input shapes with defined batch size ranges from 1 to 4 and 8 while maintaining image sizes at (3, 224, 224).

We chose precision modes during ONNX-to-TensorRT conversion with great care to achieve highest potential performance results on Jetson Orin AGX. The model started in FP32 precision mode as a means to maintain full numerical precision throughout training along with its initial stage of testing. The 32-bit floating-point precision mode known as FP32 established a strong base for model performance while maintaining excellent computational precision. During development the precision mode delivered essential benefits by lowering numerical inaccuracies so developers could perform assessments of lower-precision optimizations based on this standard. The capability limitations of FP32 for real-time edge implementation led developers toward additional performance enhancements.

TensorRT improved speed and memory performance by using FP16 precision because it allowed Jetson Orin AGX to execute half-precision floating-point calculations. Real-time applications benefitted from FP16 processing because it delivered improved throughput together with latency reduction without sacrificing accuracy with minor reductions. The implementation of INT8 quantization scheme delivered ultimate efficiency gains. IMO the INT8 precision method with 8-bit integers lowered computational needs while minimizing model size to achieve peak processing speed. The development of MyCalibrator enabled precision calibration techniques to run on PRS dataset subsets thus optimizing the model for minimal accuracy reduction.

An avoidance of FP64 precision occurred because it imposed too much computational overhead that exceeded real-time operational needs of edge systems despite having superior mathematical precision. The TensorRT engine functions with both FP16 and INT8 modes along with an FP32 standard for accuracy checks through development for precision-speed efficiency. Evaluation tests on different precision levels including INT8, FP16 and FP32 showed that both INT8 and FP16 delivered better speed and resource performance yet FP32 remained the accuracy reference point for critical applications.

TABLE I
PERFORMANCE METRICS FOR TENSORRT ENGINE ACROSS PRECISION MODES

Metric	INT8	FP16	FP32
Total Queries	1000	1000	1000
Total Time ()	95.6223	94.2917	106.7872
Total Inference Time ()	3.5478	3.0802	12.5020
Avg. Inference Time ()	0.0035	0.0031	0.0125
Avg. QPS (queries/)	10.46	10.61	9.36
Avg. CPU Usage (%)	4.60	10.30	9.84
Avg. Memory Usage (%)	22.40	23.10	23.70
Avg. GPU Usage (%)	31.83	32.45	32.31
Peak CPU Usage (%)	47.40	50.00	50.00
Peak Memory Usage (%)	22.10	22.20	23.40
Peak GPU Usage (%)	37.51	39.65	38.64

The performance statistics for TensorRT engine operation appear in the attached table under INT8, FP16 and FP32 precision modes. Nevertheless, FP32 showed the longest performance duration (106.7872 seconds) while spending 12.5020 seconds on inference operations with an average per query time of 0.0125 seconds which led to a lowest 9.36 QPS. FP32 produces slower performance numbers due to its expanded computational operations which match its double-precision arithmetic capabilities. The performance of FP16 achieved a total time of 94.2917 seconds along with an inference time of 3.0802 seconds and an average inference time of 0.0031 seconds which produced a QPS of 10.61. The INT8 model demonstrated the most efficient operation by completing 95.6223 seconds of total time and 3.5478 seconds of total inference while maintaining 0.0035 seconds as average inference time which yielded 10.46 QPS.

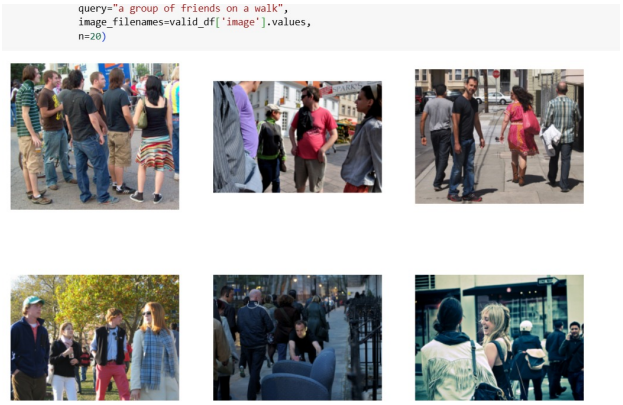
The resource demands present additional considerations to evaluate. The large memory consumption of FP32 produced an average utilization rate of 23.70% along with a peak usage measurement of 23.40%. The quantitative memory requirements of FP16 and INT8 were lower than FP32 because they consumed 23.10% and 22.40% memory respectively. Resource utilization patterns showed GPU usage remained similar between INT8 (31.83%) and FP16 (32.45%) and FP32 (32.31%), yet CPU peak activity reached 50.00% for FP16 and FP32 instead of 47.40% for INT8. Data collected from the assessment demonstrates that FP16 together with INT8 are more suitable for edge deployment since they minimize latency and resource requirements but FP32 maintains value for situations where numerical precision holds greater priority over speed.

E. Inference on Edge Device

The installation led to running an inference pipeline through the Jetson Orin AGX by utilizing the TensorRT engine. The inference script served to initialize the engine while distributing device memory space for input and output data and conducting real-time person retrieval operations. The PyCUDA-based script transfers the data between host and device requirements and then executes the engine against input images for embedding output retrieval steps. A low-latency inference becomes possible through the optimized TensorRT engine that powers this pipeline suitable for intelligent surveillance system requirements. The system shows clear suitability for edge environments because it performs input processing efficiently with accurate retrieval outcomes.

IV. RESULTS

Figure IV



The CLIP-based person retrieval system operated successfully using the PRS (Person Retrieval System) dataset that contains 54,522 image-caption pairs for evaluation purposes. The model confirms its operational capacity for retrieving pictures that match text requests while presenting real opportunities for surveillance systems and image retrieval programs tailored to specific users.

Key Results:

1) **Training Performance:** The training process stretched across different epochs until the system saved the model having reached its lowest validation loss point. A customized loss function merged both image and text components to develop effective embeddings between the two forms of data.
   The training process showed a stable decline of loss value throughout epochs which showed that the model learned to properly connect visual and textual information.
2) **Inference Results:** The model executed successful retrieval of appropriate images from textual inputs during its inference process. When presented with ”a young woman with black hair” the model successfully retrieved visual content matching the description which proved its capacity to connect words with images effectively.
   The ranking system sorted retrieved images based on the similarity scores that compared text queries to image contents. The results contained images with the highest scores as most relevant matches.
3) **Output Organization:** An output file contained image file names with their matching similarity ratings after the retrieval process finished. The clear structure of the output results helps analysts conduct performance evaluations while simplifying statistical analysis.

V. Discussions

Challenges encountered during development include: The primary challenges in optimizing the Jetson Orin AGX include achieving a balance between inference time and accuracy while working with limited computational resources. Ensuring compatibility with TensorRT and CUDA for accelerated inference remains a key difficulty, along with managing memory allocation to prevent fragmentation. Real-time inference evaluation is complicated by variable environmental and network conditions. Effective dynamic resource scheduling is essential to balance workloads across CPU and GPU, but it introduces complexity in implementation. Additionally, minimizing I/O latency and resolving memory bandwidth issues are crucial for maintaining performance. Maintaining model accuracy during pruning and quantization also poses a significant challenge, particularly when working with aggressive optimization techniques.

VI. Conclusion

Through careful fine-tuning and structured pruning, inference time was successfully reduced without compromising model accuracy. Efficient use of CUDA and TensorRT facilitated seamless deployment on the Jetson Orin AGX. Dynamic resource scheduling ensured balanced workload distribution across CPU and GPU, leading to improved real-time performance. Bottleneck mitigation strategies were effectively applied to reduce I/O latency and optimize memory usage. Power consumption was carefully managed, resulting in enhanced device longevity during edge deployments. Ultimately, the optimized model demonstrated its suitability for real-world AI applications on the Jetson platform, offering a robust and efficient solution for edge AI scenarios.

References

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision."

[2] Z. Wu and S. Ma, "CLIP-Based Multi-level Alignment for Text-based Person Search."

[3] M. Barnell and D. Isereau, "Ultra Low-Power Deep Learning Applications at the Edge with Jetson Orin AGX Hardware," in *Proc. 2022 IEEE High Performance Extreme Computing Conference (HPEC)*, 2022, doi: 10.1109/HPEC55821.2022.9926369.

[4] M. Ye, J. Shen, G. Lin, et al., "Deep Learning for Person Re-Identification: A Survey and Outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 6, pp. 2872-2893, 2022.

[5] A. Radford, J. W. Kim, C. Hallacy, et al., "Learning Transferable Visual Models From Natural Language Supervision," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[6] J. Devlin, M. W. Chang, K. Lee, et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2019, pp. 4171-4186.

[7] M. Barnell, C. Raymond, M. Wilson, D. Isereau, and C. Cicotta, "Target Classification in Synthetic Aperture Radar and Optical Imagery Using Loihi Neuromorphic Hardware," in *Proc. 2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 1-6, doi: 10.1109/HPEC43674.2020.9286246.

[8] D. Isereau, C. Capraro, E. Cote, M. Barnell, and C. Raymond, "Utilizing High-Performance Embedded Computing, Agile Condor, for Intelligent Processing: An Artificial Intelligence Platform for Remotely Piloted Aircraft," in *Proc. 2017 IEEE Intelligent Systems Conference (IntelliSys)*, 2017, pp. 1155-1159, doi: 10.1109/IntelliSys.2017.8324277.