Journal homepage: https://itce.vn.ua/en

Information Technologies and Computer Engineering **Vol. 23, No. 1. 2026**

Article's History: Received: 12.11.2025 Revised: 13.02.2026
Accepted: 26.03.2026 Published: 20.04.2026

ISSN 1999-9941; e-ISSN 2078-6387
UDC 004.934.2:004.42
DOI: 10.31649/vitce/1.2026.110

Improving the efficiency of Whisper-based audio stream processing with CTranslate2 and FFMpeg tools

Vladyslav Radin*
Postraduate Student
National University "Kyiv Aviation Institute"
03058, 1 Liubomyr Huzar Ave., Kyiv, Ukraine
https://orcid.org/0009-0009-1101-6888

Myroslav Riabyi
PhD in Technical Sciences, Associate Professor
National University "Kyiv Aviation Institute"
03058, 1 Liubomyr Huzar Ave., Kyiv, Ukraine
https://orcid.org/0000-0002-9651-9135

Abstract. The relevance of the study lies in the need to increase the performance and scalability of automatic speech recognition systems on devices with limited resources, which determines the goal of the work – to optimise Whisper by integrating CTranslate2 to accelerate calculations and FFmpeg for unified preparation of audio data. Experimental studies were conducted using the Whisper Turbo model on a graphics processor unit with support for the Compute Unified Device Architecture (CUDA) platform. The basic pipeline in the Python programming language, the optimised inference execution mechanism via CTranslate2 and the configuration with hybrid quantisation in the int8_float16 format were compared. The efficiency was evaluated using the indicators of prediction (inference) execution time, video memory use, and automatic speech recognition accuracy (Word Error Rate). Experimental results showed that the basic Whisper Turbo configuration provided the highest recognition accuracy (Word Error Rate = 0), but was characterised by high inference latency (8.5 s per audio file) and significant video memory consumption (4.9 GB). CTranslate2 integration reduced the processing time to 4.9 s (1.7 × speedup) and reduced Video Random Access Memory usage to 1.8 GB (-63%) without loss of quality. Further application of hybrid quantisation int8_float16 provided a reduction of inference time to 3.8 s and a reduction of memory consumption to 1 GB, which corresponds to an overall speedup of about 2.2× and an almost fivefold (4.9×) reduction in Video Random Access Memory requirements compared to the standard implementation, with unchanged Word Error Rate = 0. The obtained results confirmed the effectiveness of the combination of CTranslate2 and hybrid quantisation for building high-performance real-time Automatic Speech Recognition systems without compromising accuracy. The conclusions confirmed the practical suitability of the proposed configuration for multi-user services and edge scenarios without compromising speed and accuracy. The results of the study can be used by developers of automatic speech recognition systems to optimise models on memory-limited GPUs, and by companies providing streaming audio and multi-user services

Keywords: quantisation; automatic speech recognition; operator fusion; video memory; resource efficiency

Introduction

The rapid growth of audio data volumes and the spread of real-time applications have led to increased performance requirements for automatic speech recognition systems. A typical speech-to-text pipeline includes signal preprocessing, neural network inference, and text decoding, with the main computational load falling on transformer architectures with attention mechanisms. Whisper-class models demonstrate high accuracy, but are characterised by significant latency and resource consumption, which limits the scalability. Specialised tools are used to improve efficiency,

Suggested Citation:
Radin, V., & Riabyi, M. (2026). Improving the efficiency of Whisper-based audio stream processing with CTranslate2 and FFMpeg tools. *Information Technologies and Computer Engineering*, 23(1), 110-124. doi: 10.31649/vitce/1.2026.110

*Corresponding author

in particular FFMpeg for fast processing of audio streams and CTranslate2 for optimised transformer inference. At the same time, the problems of quadratic complexity of self- and cross-attention and additional costs caused by beam search remain relevant. In this context, optimisation methods, in particular operator fusion and quantisation, play a critical role, which allow reducing latency and memory consumption with minimal impact on recognition quality.

In the modern scientific discourse, research on automatic speech recognition is shifting towards combining deep neural architectures with practical aspects of performance and deployment on limited computing platforms. Thus, in the work of O. Nakhod (2025), an approach to automatic Ukrainian speech recognition based on deep learning methods is presented, where the main emphasis is on adapting modern neural network models to the specifics of the national language corpus. The author demonstrated that the use of transformer architectures allows achieving high accuracy rates even with a limited amount of training data, while emphasising the increased computational requirements of such models, which actualises the problem of the further optimisation for practical application. A more generalised overview of modern approaches is given in the study by Ye. Mrozek (2024), which systematises classical and neural network methods for solving speech recognition problems, including hidden Markov models, deep recurrent networks and transformers. The author showed that the transition to end-to-end architectures simplified the construction of Automatic Speech Recognition (ASR) systems, but at the same time led to a sharp increase in the complexity of inference and memory consumption.

The practical aspect of the performance of the Whisper series models is considered in detail in the work of Y. Cao (2025), where an empirical assessment of the performance of various Whisper configurations on the Raspberry Pi platform was carried out. The author demonstrated that, despite the high quality of transcription, the basic implementations of the models are too resource-intensive for edge devices, and achieving acceptable time characteristics is possible only by optimising inference and reducing the bit depth of calculations. The results obtained confirmed the critical role of such approaches as quantisation and hardware-oriented acceleration libraries for the practical deployment of ASR systems in environments with limited memory and computing power.

In the publication of M. Maurya *et al*. (2025), the authors conducted a comprehensive analysis of architectural approaches to building speech recognition systems, including the use of hybrid models based on neural networks, transformers, and statistical methods. The researchers examined in detail the challenges associated with noise robustness, speaker variability, and processing latency, and also emphasised the critical role of optimising computational resources for implementing ASR in mobile and edge scenarios. The results confirmed that the effectiveness of the system is determined not only by decoding accuracy, but also by the balance between performance, scalability, and hardware limitations.

Another direction of Whisper development is presented in the work of I. Thorbecke *et al*. (2024), which considers the use of knowledge distillation to build fast streaming ASR models based on Whisper as a teacher model. The authors proposed the fast streaming transducer, which inherited the acoustic and linguistic properties of Whisper, but has lower computational complexity. Empirical results showed that distilled models provide lower latency and better real-time performance while maintaining an acceptable Word Error Rate (WER). An example of an applied use of speech recognition systems is given in the work by X. Wu *et al*. (2023), which describes a system for assessing the quality of English pronunciation based on continuous speech recognition in a multi-terminal environment. The authors' results demonstrated typical challenges for distributed ASR systems: the need for fast processing of streams from different clients, limited end-device resources, and requirements for stability of operation.

N.T. Hung *et al*. (2025) presented Effwhis, an optimised approach to streaming speech recognition based on Whisper. The authors demonstrated that modifications to buffering and resource management can reduce inference latency while maintaining recognition accuracy at the standard model level. This work highlighted the practical feasibility of optimising Whisper for streaming applications where low latency and efficient memory usage are important. At the same time, F. Chettiar *et al*. (2025) investigated deep neural architectures for multilingual video translation with the integration of speech recognition and synthesis. The authors demonstrated that combining ASR and TTS within a single model reduces errors in transfer between languages and increases semantic consistency of results. Special attention was paid to optimising processing latency and ensuring consistent quality when working with different language pairs. This has demonstrated a trend towards building complex multimodal systems, in which speech recognition modules function as components of broader adaptive platforms.

In the work of A. Trabelsi *et al*. (2024), an assessment of the impact of noise reduction filters on the quality of open ASR models, including Whisper, was proposed. The study showed that pre-processing of audio to reduce noise does not always correlate with an increase in transcription accuracy; in some cases, excessive aggressiveness of noise reduction algorithms can reduce WER. In turn, J. El Bahri *et al.* (2025) compared the performance of Whisper (Base and Large) with the Google Speech-to-Text V2 service in terms of energy efficiency and computational costs. The study confirmed that optimised Whisper models using quantisation and accelerated libraries demonstrate a better balance between resource consumption and accuracy, especially on devices with limited computing capabilities.

Despite numerous optimisations of Whisper for streaming recognition and model quantisation, previous studies were mostly limited to the analysis of individual hardware and software improvements and did not conduct a comprehensive comparison of the effect of operator fusion, quantisation, and audio stream preparation on speed, Video

Random Access Memory (VRAM) consumption, and transcription accuracy simultaneously. The aim of the work was to improve the Whisper speech recognition system using the CTranslate2 and FFmpeg tools. To achieve this goal, the following tasks were formulated: to develop and experimentally verify the basic configuration of the speech recognition system based on Whisper Turbo; to create an optimised version of Whisper using CTranslate2 without quantisation, and then compare its performance with the basic version; to implement the Whisper version using CTranslate2 together with quantisation changes (int8_float16 format).

Materials and Methods

The study was experimental and applied in nature and was conducted in late 2025 in a controlled laboratory environment using a fixed hardware and software configuration. To ensure the reproducibility and comparability of the experimental configurations, three key software components were used: Whisper for audio-to-text transformation, CTranslate2 for optimising the inference process of the transformer model, and FFmpeg for audio data preparation. Below is a generalised description of each tool and its role in the study (Table 1).

Table 1. Characteristics and advantages of the tools used

Tool	Main function	Features	Reason for selection in the study
Whisper (Turbo)	Automatic speech recognition (Speech-to-Text, STT)	Transformer encoder-decoder architecture, autoregressive decoder, large-v3 Turbo	High recognition accuracy, support for long audio sequences, reliability in tasks of generating text from audio
CTranslate2	Optimisation of transformer model inference	Tensor operator fusion, management of internal memory buffers, SIMD and GPU support, weight quantisation	Accelerates inference, reduces video memory consumption, allows the use of hybrid weight formats without loss of accuracy
FFmpeg	Preparation and processing of audio files	Decoding, normalisation, resampling, format conversion, multithreading, hardware acceleration	Provides a unified audio format and parameters, stabilises input data for objective comparison of different Whisper configurations

Source: compiled by the authors

The table shows that the efficiency of the speech recognition system is achieved through the interaction of three components: Whisper provides accurate audio-to-text conversion, FFmpeg stabilises the input data and reduces signal variability, and CTranslate2 optimises the computational pipeline and allows the use of quantisation to accelerate inference and reduce hardware costs; the combined use of these tools creates a scalable, productive and resource-efficient platform for processing audio streams in various scenarios, including edge computing and multi-user services. Audio data preparation in all experimental configurations was carried out using FFmpeg for decoding, normalisation and bringing the signal to unified parameters. The main differences between the configurations were the use of different inference backends: standard Whisper Python-pipeline or optimised CTranslate2 with hybrid quantisation. Figure 1 illustrates the step-by-step process of processing an audio stream in an ASR system.

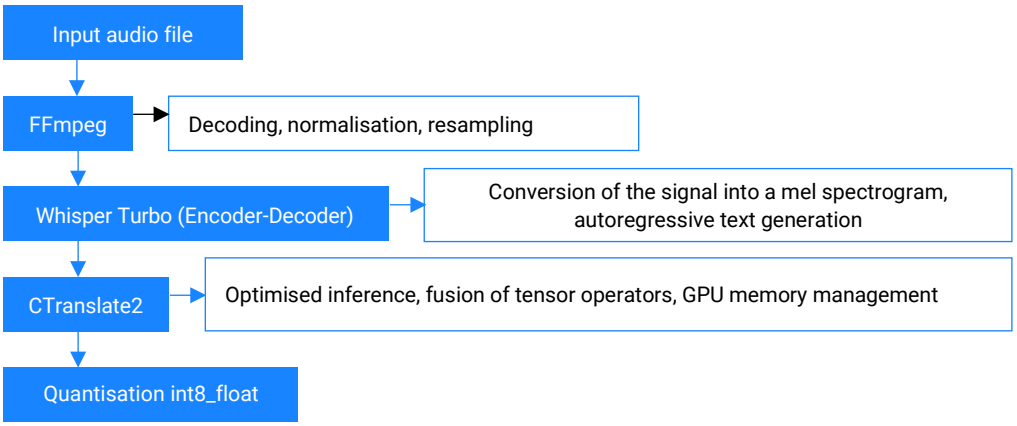

Figure 1. Schematic of the audio stream processing using Whisper, CTranslate2 and FFmpeg, taking into account quantisation

Source: compiled by the authors

The audio data was prepared using FFmpeg through decoding, normalisation and resampling to a unified set of parameters; the processing was carried out by the Whisper Turbo model with an encoder-decoder transformer architecture, CTranslate2 was used to optimise the inference, and the hybrid quantisation int8_float16 reduced the bit depth of the model weights, accelerating the calculations without losing accuracy. For experimental evaluation, one control

audio file with a duration of 2 min 41 s with high recording quality and minimal background noise was used. The total number of words in the speech signal was 165, which provided a sufficient sequence length to activate both the encoder and decoder mechanisms of the transformer. All audio data was converted to a single format and sampling rate using FFmpeg, after which it was fed directly into the ASR pipeline. Formally, the inference process was described by representing the signal in the form of a feature matrix (1):

$$X \in R^{T \times d}, \quad (1)$$

and subsequent self-attention operations with projections Q (queries), K (keys), V (values) and the scaled dot-product attention mechanism, which allowed analytically relating the time complexity to the parameter T and the dimensionality of the representations, where T corresponds to the number of time steps (frames) obtained in the process of spectral analysis; d is the dimensionality of the feature vector at each time step. It is the parameter T that determines the scale of calculations in subsequent self-attention layers and affects the overall computational complexity. This approach allowed minimising the variability associated with different parameters of the input signal and focusing exclusively on the impact of computational optimisations.

The experimental study was structured as a step-by-step comparison of three configurations of the automatic speech recognition system: a basic implementation of Whisper Turbo in a standard Python environment with hardware acceleration via Compute Unified Device Architecture (CUDA), a modified version of Whisper Turbo with integration of the CTranslate2 library, and an extended configuration of Whisper Turbo with CTranslate2 and additional quantisation of model parameters. This sequence allowed evaluating the contribution of each level of optimisation, starting from eliminating the overhead of the high-level pipeline and ending with reducing the bit depth of calculations. For each configuration, key performance indicators were recorded, namely the total inference time of one audio file, peak GPU video memory consumption, and the value of the WER recognition quality indicator. The processing duration was defined as the difference between the start and end timestamps of the transcription function call, which covers the full audio signal processing cycle, including file decoding, acoustic feature generation, passing through the encoder and decoder of the transformer architecture, and generating the final text transcription. This measurement approach provided an integral assessment of the real system latency in a practical usage scenario.

Below is a code fragment that illustrates the basic configuration of using Whisper Turbo in a standard Python-pipeline with hardware acceleration via CUDA and measuring the inference time:

```
import time
import whisper
#using "turbo" model on graphic card
model = whisper.load_model("turbo", 'cuda:0')
start = time.time()
result = model.transcribe("audio.mp3")
#result text
print(result["text"])
end = time.time()
#displaying processing time
print(end - start)
```

The audio stream processing pipeline included a full inference cycle: loading a pre-trained model, decoding the audio file, building internal features, passing through the encoder and decoder of the transformer architecture, and generating a text transcription. To measure performance, the inference time was used, recorded between the start and end of the model.transcribe() function call. All experiments were performed on a local computing platform with an Intel Core i7 EVO 13700H processor, 16 GB of RAM, and an NVIDIA GeForce RTX 4050 Studio graphics adapter with 6 GB of video memory. This configuration represents a typical modern workstation or a high-performance laptop. It allowed extrapolating the results to practical scenarios for deploying ASR systems outside of data centres.

Results

Basic performance indicators of the standard Whisper implementation

The experimental results demonstrated the performance of the standard implementation of the Whisper model in the Turbo configuration without additional optimisations. The model was used in a standard Python pipeline with GPU acceleration via CUDA, which is typical for practical speech recognition systems. Experimental results showed that processing one audio file in the basic Whisper Turbo configuration took 8.5 s, which is a relatively high figure for scenarios focused on processing streaming or quasi-real-time audio data. At the same time, the recognition quality remained at its maximum: the Word Error Rate value was equal to 0, which indicates correct reproduction of the speech signal without errors. These results confirm the effectiveness of the base Whisper Turbo configuration for use in real-world speech recognition scenarios (Fig. 2).

After calling the model.transcribe ("audio.mp3") method, a full cycle of processing the audio file was performed, which included signal decoding, pre-processing, formation of internal acoustic features and generation of text transcription. Recording timestamps at the beginning and end of the inference execution allowed determining the total duration of audio stream processing, which is used as a baseline for further comparison with optimised system configurations. At the same time, an analysis of hardware resource usage showed that the peak video memory consumption reached 4.9 GB VRAM, which is a significant indicator even for the optimised Turbo model (Fig. 3).

```
(.venv) PS C:\Users\vladr\Documents\GITProjects\Whisper\defaultWhisper\pythonProje
ct> python main.py
 Мені тринадцятий минало, Я пас ягнята за селом. Чи то так сонечко сіяло, Чи так м
ені чогось було, Мені так любо-любо стало, Не наче в Бога. Уже покликали до паю, А
 я собі у бур'яні Молюся Богу, і не знаю, Чого маленькому мені Тоді так приязно мо
лилось, Чого так весело було. Господнє небо і село, Ягня, здається, веселилось, І
сонце гріло, не пекло, Та недовго сонце гріло, Недовго молилось, запекло, Зачервон
іло, І рай запалило. Мов прокинувся, дивлюся, Село почорніло, Боже, небо голубіє,
І те померніло. Поглянув я на ягнята, Не мої ягнята, Обернувся я на хати, Нема в м
ене хати. Не дав мені Бог нічого, І хлинули сльози, Тяжкі сльози. А дівчина при са
мій дорозі, Недалеко коло мене, Плоскінь вибирала. Та й почула, що я плачу. Прийшл
а, привітала, Утирала мої сльози, І поцілувала. Не наче сонце засіяло, Не наче все
 на світі стало моє, Лани, гаї, сади, І ми, жартуючи, погнали, Чужі ягнята до води
. Бридня! А й досі, як згадаю, То серце плаче та болить. Чому Господь не дав дожит
ь Малого віку у тім раю? Умер би, орючи на ниві, Нічого б у світі не знав, Не був
би в світі юродивим, Людей і Бога не прокляв.
8.509320497512817
```

Figure 2. The result of processing the standard Whisper configuration with the Turbo model
Source: compiled by the authors

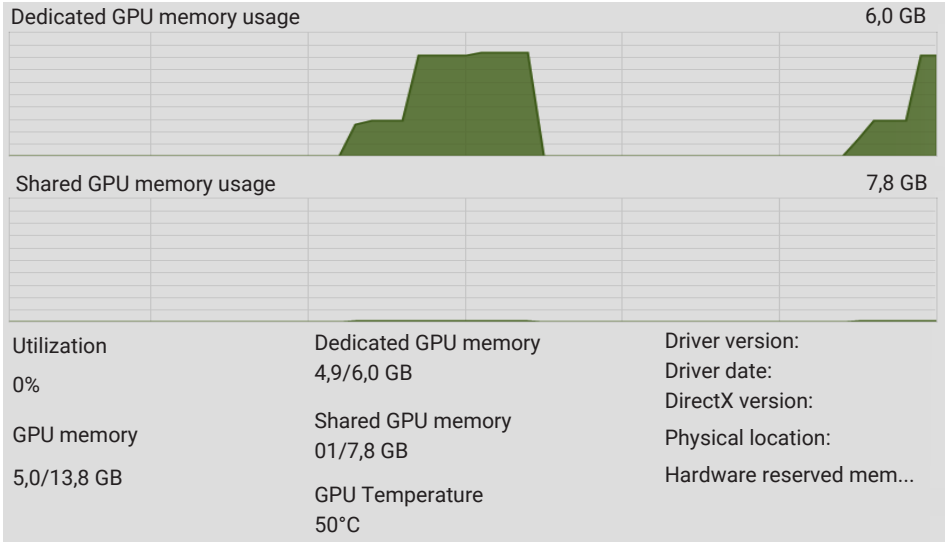

Figure 3. Resource consumption of the standard Whisper configuration with Turbo model
Source: compiled by the authors

The use of almost 5 GB of video memory limits the possibility of implementing such a configuration on common hardware, in particular on laptops or workstations with graphics adapters with 4-6 GB of VRAM. In addition, high resource consumption makes it difficult to simultaneously process multiple audio streams on a single GPU. From a scaling perspective, this means that supporting numerous parallel audio sessions requires either a significant increase in the number of GPUs, or the use of optimisation methods aimed at reducing the model size and computational costs without degrading accuracy.

Optimisation of the Whisper computational pipeline using CTranslate2

The main direction of optimisation of the studied automatic speech recognition system was the use of the CTranslate2 library, which is focused on high-performance inference of transformer models. The key advantage of this approach is the direct management of memory and internal buffers, which allowed minimising the number of calls to high-level abstractions typical in the standard Python-pipeline. As a result, the overhead of memory management was reduced, which had a positive effect on latency and overall performance of the system. To increase efficiency, CTranslate2 implements the fusion of adjacent tensor operations, combining these operations into a single call to the GPU computing core. This approach provided more efficient data caching and significantly reduced communication costs between individual computation stages, which is critical when processing large layers of the Whisper transformer architecture. The processing was carried out on the basis of the mathematical model described by formula (1), which determines the matrix representation of the mel-spectrogram and further transformations in the encoder. In each self-attention layer of the encoder, the feature matrix X is linearly projected into the spaces of queries, keys, and values:

$$Attn(X) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2)$$

where $Q = XW_Q$, $K = XW_K$, $V = XW_V$, W_Q, W_K, $W_V \in R^{d \times d_k}$ – parameter matrices, in which Q is the query matrix, which encodes what information each position of the sequence "looks" for in other positions. Each row of Q corresponds to one time step and contains a feature vector, which determines the relevance of other frames for this position; K is the key matrix, which encodes what information each position can provide. During the multiplication QK^T, the similarity measure between queries and keys is calculated, i.e., the weight of the mutual influence between time steps is determined; V is a matrix of values that contains the actual information representation of each position. After normalisation of the attention coefficients through the function, the weights are applied to V, forming a new aggregated representation of the features; d_k is the dimension of the key space.

The quadratic dependence on T, which arises as a result of the multiplication , determines the computational complexity of one self-attention layer of the order If there are L_e (the number of sequentially applied self-attention + feed-forward blocks in model encoder) such layers in the encoder, the total computational cost of the encoder part can be approximated as (3):

$$C_{enc}(T) \approx L_e \times C_{self}(T, d_k), \quad (3)$$

which explains the significant contribution of the encoder to the overall latency, especially for long audio fragments. The Whisper decoder operates in autoregressive mode.

At each step t, a partial sequence of tokens of length t is formed, which interacts with the encoder output through the cross-attention mechanism (4):

$$Attn_t = softmax\left(\frac{Q_t K_e^T}{\sqrt{d_k}}\right)V_e, \quad (4)$$

where $Q_t = H_t W_Q^{(d)}, K_e = H_e W_K^{(d)}, V_e = H_e W_V^{(d)}$ – trained weight matrices (attention-head parameters d); H_e is the encoder output, and H_t is the decoder representation at step t.

For one decoding step, the cross-attention complexity C_{cross} is estimated as (5):

$$C_{cross}(T, t) = O(Ttd_k), \quad (5)$$

which reflects the dependence on both the length of the input signal and the current length of the generated sequence.

For autoregressive sequence generation of length N (length of the generated (target) sequence), i.e. the number of tokens that the model autoregressively produces at the output) with Ld (number of decoder layers in the transformer architecture), the basic cost (without optimisations) is approximated by (6):

$$C_{dec}(T, N) \approx \sum_{t=1}^{N} L_d \times C_{cross}(T, t) = L_d \times O\left(Td_k \frac{N(N+1)}{2}\right). \quad (6)$$

Applying beam search with bea m size b linearly scales these costs, since the calculations are performed in parallel for several hypotheses (7):

$$C_{dec}^{beam}(T, N, b) \approx b \times C_{dec}(T, N). \quad (7)$$

Thus, the total computational cost of the standard Whisper implementation in this implementation can be represented as the sum of the encoder and decoder costs:

$$C_{base}(T, N, b) \approx C_{enc}(T) + C_{dec}^{beam}(T, N, b). \quad (8)$$

In CTranslate2, the implementation involves replacing the sequence of individual tensor operations, such as matrix calculations, normalisations, and activations, with the use of a dedicated computational kernel. This means that instead of executing a set of operators in stages (9):

$$O = \{o_1, o_2, \ldots, o_k\}, \quad (9)$$

each of which is accompanied by separate accesses to global memory, to a single fused operation (10):

$$\tilde{o} = f(o_1, o_2, \ldots, o_k), \quad (10)$$

which is executed within a single GPU kernel launch. Each operator o_i has its own computational cost c_b and memory access cost m_i.

Then for the basic version of the implementation, the result will be (11):

$$C_{layer}^{base} = \sum_{i=1}^{k} c_i \,, M_{layer}^{base} = \sum_{i=1}^{k} m_i. \quad (11)$$

where c_i is the computational cost; m_i is the memory access cost for each operator, then after fusing these costs are reduced to (12):

$$C_{layer}^{opt} \approx \alpha \sum_{i=1}^{k} c_i, \; M_{layer}^{opt} = \beta \sum_{i=1}^{k} m_i, \quad (12)$$

where the coefficients $\alpha, \beta \in (0.1)$ reflect the reduction in computational and memory costs. This is due to operator optimisation and improved caching.

The overall efficiency gain of the model can be represented through the speedup coefficient (13):

$$S_{fuse} = \frac{C_{base}(T,N,b)}{C_{opt}(T,N,b)} \approx \frac{C_{enc}^{base} + C_{dec}^{base}}{\alpha C_{enc}^{base} + \alpha C_{dec}^{base}} = \frac{1}{\alpha}. \quad (13)$$

Thus, the given formalism demonstrates that the reduction of the coefficient $\alpha\backslash alphaa$ directly transforms into a reduction of the inference time, while the reduction of $\beta\backslash betaa$ reduces the load on the memory subsystem. This theoretically justifies the experimentally recorded acceleration of Whisper after the integration of CTranslate2 and explains the mechanisms of reducing the latency of audio-stream processing without losing the recognition accuracy. The results of Whisper inference using CTranslate2 for the Turbo model are shown in Figure 4.

```
 \pythonProject1\main.py
  Мені тринадцятий минало, Я пас ягнята за селом. Чи то так сонечко
   сіяло, Чи так мені чогось було. Мені так любо-любо стало, Не наче в
   Бога. Уже покликали до паю, А я собі у бур'яні Молюся Богу. І не
   знаю, чого,  Маленькому мені тоді Так приязно молилось. Чого так
   весело було? Господнє небо І село, Ягня, здається, веселилось, І
   сонце гріло, Не пекло. Та недовго сонце гріло, Недовго молилось,
   Запекло, зачервоніло, І рай запалило. Мов прокинувся,  Село
   почорніло, Боже, небо голубеє, І те померніло. Поглянув я на ягнята,
    Не мої ягнята, Обернувся я на хати, Нема в мене хати. Не дав мені
   Бог нічого, І хлинули сльози, Тяжкі сльози. А дівчина при самій
   дорозі, Недалеко коло мене,  Лоскинь вибирала. Та й почула, Що я
   плачу. Прийшла, привітала, Утирала мої сльози І поцілувала. Неначе
   сонце засіяло, Неначе все на світі стало моє, Лани, гаї, сади,
   Жартуючи погнали Чужі ягнята До води Бридня. А й досі, як згадаю, То
    серце плаче Та болить. Чому Господь Не дав дожить Малого віку У тім
    раю? Умер би Орючи на ниві, Нічого б у світі не знав, Не був би в
   світі  Юродивим Людей І Бога Не прокляв.
4.984297275543213
```

Figure 4. Result of Whisper+CTranslate2 processing with the Turbo model
Source: compiled by the authors

For additional optimisation of the load distribution between CPU and GPU, as well as for efficient encoding and decoding of the audio signal, FFmpeg was used instead of Whisper's standard audio module. As a result of adaptation of Whisper ASR to CTranslate2, the full processing cycle of the original audio file was reduced to 4.9 s. At the same time, the maximum recognition accuracy (WER = 0) was maintained, and the hardware resource consumption was significantly reduced: 1.8 GB VRAM compared to 4.9 GB in the standard configuration (Fig. 5).

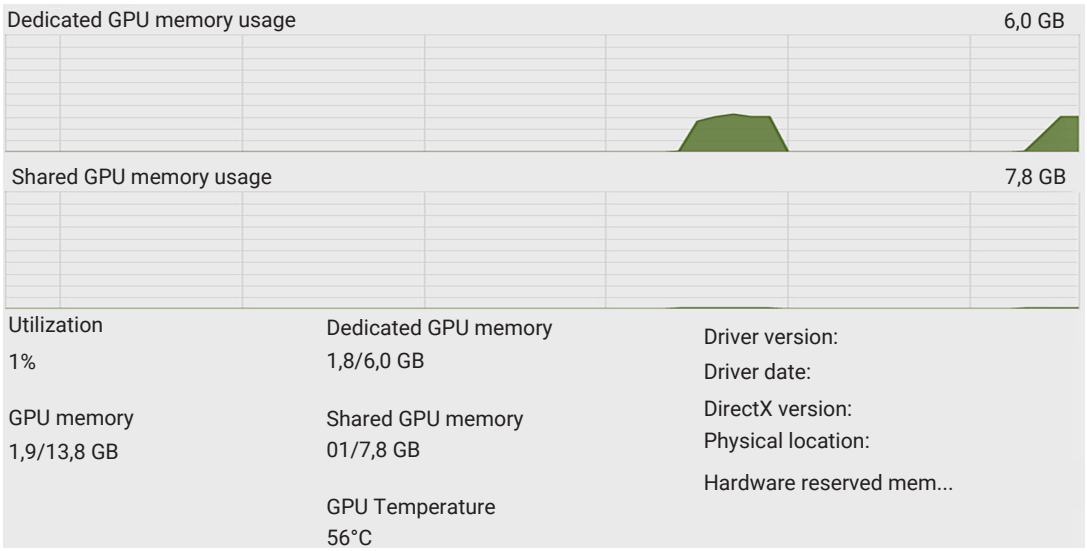

Figure 5. Video card resource consumption Whisper+CTranslate2 with the Turbo model
Source: compiled by the authors

The results obtained confirm the effectiveness of using CTranslate2 as a tool for optimising transformer models for automatic speech recognition tasks, especially in the context of reducing latency and hardware resource requirements without losing quality. Reducing the processing time from 8.5 to 4.9 s provides an almost 1.7-fold system speedup without degrading recognition quality. In addition, reducing video memory consumption from 4.9 GB to 1.8 GB allows the model to be used on a variety of hardware platforms, including more affordable GPUs. This is important for situations where multiple audio streams need to be processed in parallel, for example, when monitoring different speech sources or creating multi-user speech recognition systems.

Another advantage of CTranslate2 is the ability to quantise the model. In this case, quantisation means reducing the bit depth in which the model weights are represented, starting from the original format. This approach is based on the assumption that the computational performance of the model does not usually require high precision

(e.g., FP32) during inference. Thus, reducing the precision can lead to a small loss in recognition accuracy. Formally, the quantisation process can be represented as a mapping of values from the continuous space of real numbers to a discrete integer space with limited bit width. This mapping is described by relation (14):

$$Q_q \div R \to Z, x_q = Q_q(x), x \approx s \times x_q, \quad (14)$$

where $Q_q \div R \to Z$ denotes the quantisation operator (Q_q), which maps values from the space of real numbers (R) into the discrete (integer) space (Z); x – the initial real value of the parameter or activation; x_q – its quantised representation; s is the scale factor that restores the approximate value in the original number space. The scale factor plays a key role in reducing the approximation error and determines the trade-off between representation accuracy and computational efficiency.

From the point of view of hardware implementation, the reduction in bit depth directly affects the cost of performing basic linear algebraic operations, primarily matrix multiplications, which dominate in transformer architectures. If denoting the computational cost of an operation in FP32 format as c_{fp32}, and in quantised format as c_q, then for large matrix operations the following approximation (15) is valid:

$$c_q \approx \gamma c_{fp32}, \gamma \in (0.1), \quad (15)$$

where the coefficient γ reflects the relative reduction in computational costs due to the use of smaller bit widths and specialised vector instructions of the processor or GPU. Similarly, reducing the bit widths of weights and intermediate tensors leads to a proportional reduction in memory requirements, which can be expressed as (16):

$$m_q \approx \delta m_{fp32}, \delta \in (0.1), \quad (16)$$

where m_{fp32} and m_q are the memory sizes required to store parameters in FP32 and quantised formats, respectively. Reducing the coefficient $\delta \backslash delta\delta$ is especially important for scenarios where models are deployed on devices with limited VRAM. Combining the effects of quantisation with the optimisations achieved through the fusion of operations in CTranslate2 allows formulating a generalised model of the computational complexity of the optimised system. If the basic complexity of the standard Whisper implementation is denoted as $C_{base}(T, N, b)$ (formula (8)), then for the quantised configuration (17):

$$C_{opt}^{quant}(T, N, b) \approx \gamma S_{fuse}^{-1} C_{base}(T, N, b), \quad (17)$$

where $S_{fuse}^{-1} = \alpha$ corresponds to the effect of fusing operations, and γ is an additional reduction in computational cost due to quantisation.

Thus, the total performance gain is formed multiplicatively due to two independent optimisation mechanisms: structural optimisation of the computational graph and reduction of numerical accuracy. The resulting formalisation explains the experimentally observed reduction in inference time and reduction in video memory consumption without degrading the WER indicator. It also confirms the feasibility of using quantisation in combination with CTranslate2 as an effective tool for adapting Whisper to real-time scenarios and limited hardware resources. Figure 6 shows the VRAM usage profile after the Whisper model transitions to a less resource-intensive numeric format for representing parameters and activations. The reduction in GPU memory usage compared to the standard full-bit (FP32/FP16) implementation is visually recorded, which indicates the effectiveness of the applied quantisation or low-bit data representation.

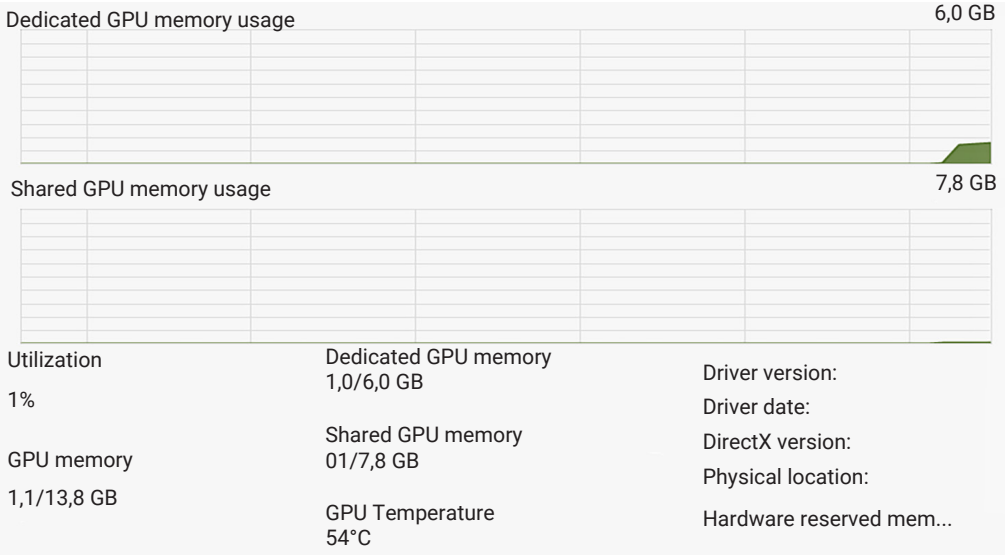

Figure 6. Resource consumption of the Whisper+CTranslate2 video card with the Turbo model and int8_float16 quantisation

Source: compiled by the authors

The reduction in VRAM is directly related to the reduction in the size of the model parameters, intermediate activation tensors and service buffers. The transition, for example, from FP32 (32 bits) to INT8 (8 bits) or FP16 (16 bits) theoretically reduces memory costs by a factor of 4 or 2 for each tensor, respectively. This not only reduces static memory consumption, but also significantly reduces dynamic costs associated with data caching and transfer between levels of the memory hierarchy (L2 cache, shared memory, global memory GPU). From the point of view of system architecture, this leads to a reduction in memory bandwidth pressure, which is a critical bottleneck in transformer models, where operations such as GEMM and attention dominate. Thus, not only the fact of reducing VRAM consumption is illustrated, but also a fundamental change in the calculation profile: the model moves from memory-bound mode closer to compute-bound, which is desirable from the point of view of optimising performance. At the same time, Figure 7 presents the result of the Whisper automatic speech recognition system in the Turbo configuration after integrating CTranslate2 and applying hybrid quantisation int8_float16. The result demonstrates correct text transcription of the input audio signal, which indicates the preservation of semantic and lexical recognition accuracy under conditions of a significant reduction in computational and memory costs. The actual processing time of the audio file in this configuration was 3.8 s, which is the lowest value among all the options studied. At the same time, the Word Error Rate (WER) indicator remained equal to 0, i.e., quantisation and optimisation did not lead to degradation of the recognition quality.

```
\pythonProject1\main.py
 Мені тринадцятий минало. Я пас ягнята за селом. Чи то так сонечко
  сіяло? Чи так мені чогось було? Мені так любо-любо стало, не наче в
  Бога. Уже покликали до паю, а я собі у бур'яні молюся Богу. І не
  знаю, чого.  Маленькому мені тоді так приязно молилось. Чого так
  весело було? Господнє небо і село. Ягня, здається, веселилось. І
  сонце гріло, не пекло. Та недовго сонце гріло, недовго молилось.
  Запекло, зачервоніло і рай запалило. Мов прокинувся, дивлюсь.  Село
  почорніло. Боже, небо голубеє і те померніло. Поглянув я на ягнята.
  Не мої ягнята. Обернувся я на хати. Нема в мене хати. Не дав мені
  Бог нічого. І хлинули сльози. Тяжкі сльози. А дівчина при самій
  дорозі недалеко коло мене  Лоскінь вибирала. Та й почула, що я
  плачу. Прийшла. Привітала. Утирала мої сльози. І поцілувала. Не наче
   сонце засіяло. Не наче все на світі стало моє. Лани, гаї, сади.
  Жартуючи погнали чужі ягнята до води. Бридня. А й досі, як згадаю,
  то серце плаче та болить. Чому Господь не дав дожить малого віку у
  тім раю? Умер би, орючи на ниві. Нічого б у світі не знав. Не був би
   в світі  Юродивим. Людей і Бога не прокляв.
3.8336899280548096
```

Figure 7. Result of Whisper+CTranslate2 processing with Turbo model and int8_float16 quantisation
Source: compiled by the authors

The achieved effect is due to the use of the hybrid numerical format int8_float16, in which the weights of the neural network are stored in the 8-bit int8 format, and the intermediate activations and calculations are performed in the 16-bit float16 format. This approach reduces the amount of memory for storing the model and the number of accesses to global memory, while maintaining sufficient dynamic range and numerical stability for multilayer transformer calculations. Figure 8 demonstrated the practical suitability of the optimised Whisper configuration for scenarios focused on real-time or multithreaded audio data processing, as well as for use on edge devices with limited hardware resources. Reduced VRAM consumption allows the system to scale without additional costs for high-performance GPUs, which makes this implementation effective for streaming ASR services, multilingual platforms, and information space monitoring systems.

Comprehensive comparison of audio stream processing configurations

In the experiment, the standard Whisper provided the maximum recognition accuracy (WER = 0), but was characterised by a high processing time for one audio file (8.5 s) and significant VRAM consumption (4.9 GB). The results showed that with such a configuration, the implementation is resource-intensive and limits the system's scalability on hardware with limited video memory. The visualisation of the resource consumption profile demonstrates a stable GPU load level with peak values during encoder and decoder calculations, which indicates significant costs for processing large tensor operations. Optimisation using CTranslate2 allowed significantly reducing both audio processing time and memory consumption. Thanks to operator fusion and optimised kernels, and more efficient data caching, the transcription time was almost halved to 4.9 s, while VRAM usage decreased to 1.8 GB. At the internal

architecture level, this was achieved by consolidating tensor operations, which reduced the number of memory accesses and the overhead of communication between the GPU core and RAM, which was confirmed by graphical profiling.

Additional quantisation of the model to the int8_float16 format demonstrated a further increase in efficiency. The weights of the neural network were converted to an 8-bit integer format, while the intermediate calculations remained in the float16 format, which provided the optimal balance between accuracy and performance. As a result, the processing time was reduced to 3.8 s, and the amount of VRAM consumed was reduced to 1 GB. This configuration also supported a zero WER, indicating that there was no loss of accuracy when applying resource-saving optimisations. To summarise the results, Table 2 was formed, which displays the main performance and resource usage metrics for each configuration. The data demonstrate a clear correlation between the level of optimisation and the reduction in hardware costs, as well as a clear increase in performance while maintaining maximum recognition accuracy.

Table 2. Comparison of Whisper configurations by key metrics

Configuration	Processing time, s	VRAM, GB	WER	Acceleration, ×	VRAM reduction, ×
Standard Whisper	8.5	4.9	0	1	1
Whisper + CTranslate2	4.9	1.8	0	1.7	2.7
Whisper + CTranslate2 + quantisation	3.8	1	0	2.2	4.9

Source: compiled by the authors

Analysis of the table demonstrates a clear correlation between the level of optimisation of the Whisper system and its performance, while also reflecting the effectiveness of resource-saving strategies. Comparison of configurations shows that the use of CTranslate2 provides a significant reduction in processing time and VRAM consumption without loss of accuracy, which indicates optimisation of calculations and reduction of overhead for memory access. Additional quantisation in the int8_float16 format not only continued the trend of reducing hardware requirements, but also demonstrated that it is possible to achieve significant processing acceleration while maintaining a zero WER, that is, without compromising recognition quality.

Synthesising these data, it is worth concluding that optimisation through CTranslate2 and quantisation does not simply reduce resource consumption, but changes the architectural efficiency of calculations: the combination of tensor operations and the reduction of the amount of processed data allows achieving acceleration by more than two times while simultaneously reducing video memory consumption by almost five times. This highlights the strategic importance of combined approaches, where core-level optimisation is combined with numerical downscaling techniques, providing a high-performance and resource-efficient environment for automatic speech recognition. At the same time, this configuration opens up opportunities for scaling on devices with limited computing power and VRAM, making the system more versatile and suitable for real-world implementation in edge computing and audio streaming environments. Figure 8 presents a graphical summary of the relationship between processing time and VRAM consumption for the three configurations.

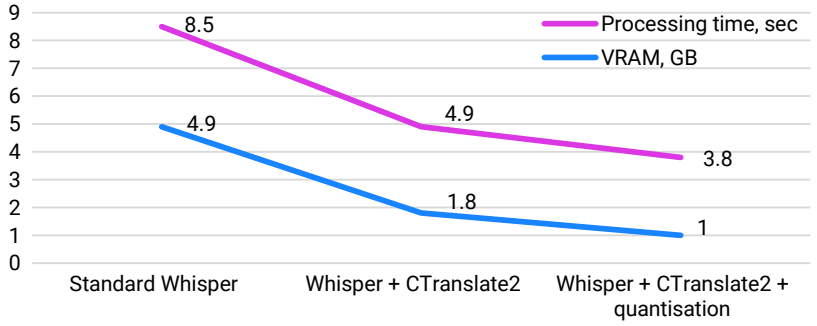

Figure 8. Dependence between audio processing time and VRAM consumption for three Whisper configurations
Source: compiled by the authors

The graph clearly demonstrates that the use of CTranslate2 and quantisation provides two key advantages: significant acceleration of transcription and a significant reduction in hardware resource requirements, while the recognition accuracy is not reduced. This approach allows for more flexible scaling of the system, ensuring its operation on limited hardware platforms, and supporting parallel processing of multiple audio streams, which is critical for multi-user and edge scenarios. Thus, a comprehensive comparison of the configurations confirmed the effectiveness of the step-by-step optimisations of Whisper, and also provided quantitative and visual arguments in favour of integrating CTranslate2 and quantisation for high-performance automatic speech recognition tasks.

Discussion

The obtained results confirm the trend towards intensive development of Whisper-type model optimisation approaches for real-time scenarios and resource-constrained environments. In particular, the recorded reduction in inference time and memory consumption without loss of accuracy is consistent with the current direction of research focused on increasing the computational efficiency of ASR systems. Thus, the empirical data obtained within the framework of the current study not only demonstrate the practical feasibility of the applied optimisations, but also confirm that the adaptation of Whisper to edge- and near-real-time scenarios is a technically justified and methodologically sound strategy for the development of modern automatic speech recognition systems. In order to objectively interpret the obtained results, the findings should be correlated with the work of other authors who studied the issue of increasing the efficiency of speech systems.

Thus, in the work of Y. Moslem *et al*. (2025), the SpeechT system is described, which used mentoring session techniques to increase the accuracy of speech translation. The authors showed that the performance of the system depended on the quality of data preparation and the training structure. Compared with the present results, the use of CTranslate2 and quantisation allowed reducing the processing time and peak VRAM consumption without losing accuracy, which was not achieved in the work by Y. Moslem *et al*. In turn, J. Ala-Rantala (2025) presented a system for generating visual content based on voice commands with low latency, demonstrating the advantages of optimised pipelines and minimising latency. Compared with the conducted study, these results partially aligned with the current processing time indicator, however, the models used were generative and did not take into account the specifics of ASR.

Methods of selecting and filtering audio data for effective further training of ASR models, which increased accuracy in highly specialised domains, were applied by P. Rangappa *et al*. (2025). The authors focused on data quality for fine-tuning, while the conducted study focused on optimising inference and reducing the load on the hardware. Therefore, although the accuracy (WER) indicators were similar, the processing time and resource consumption were significantly different. At the same time, A. Znotins *et al*. (2025) presented an open LATE toolkit for Latvian and Latgalian, which allowed for high-accuracy transcription of low-resource languages. The study showed a significant impact of high-quality corpus preparation on transcription accuracy. Compared with the present study, the authors' WER results were similar for well-prepared audio data, but the use of CTranslate2 and quantisation provided a significant reduction in processing time and peak VRAM consumption.

In the thesis of S. Kim (2024), a full-stack approach for efficient inference of deep models was presented, which included hardware optimisation, improved memory management, and parallelisation of calculations. The author proved that a comprehensive combination of hardware and software optimisations allows achieving a significant reduction in processing time and power consumption without loss of accuracy. Compared with the present study, the application of CTranslate2 for Whisper Turbo provided a similar reduction in inference time and VRAM, but the absolute processing times were higher, which is explained by the less optimised hardware configuration in the test environment and different audio characteristics. At the same time, the V-APA system, which uses voice commands to automate business processes, is presented in the study of M.H. Hwang *et al*. (2026). The authors noted the critical dependence of the system performance on the speed and accuracy of ASR, especially in real-time scenarios. The results obtained partially confirmed these conclusions: the integration of Whisper with CTranslate2 and quantisation reduced the inference time and peak VRAM consumption, which could potentially improve the efficiency of systems like V-APA. However, direct correlation of recognition accuracy (WER) was limited due to different language contexts and the specificity of the audio data.

A. Menshawy & M. Fahmy (2025) reviewed strategies and patterns for designing large language models in an enterprise environment. The authors emphasised the importance of optimising models to ensure a balance between accuracy, speed, and resource costs. The results obtained directly correlate with this approach: the use of CTranslate2 and quantisation allowed increasing the performance of Whisper without losing transcription quality. At the same time, in the enterprise context described by A. Menshawy & M. Fahmy, the focus was on integration into large systems and scalability, while the current study was local. Transformer optimisation techniques for low-latency inference, including the use of hardware acceleration, changes in layer architecture, and code optimisations, were explored in the work by A. Kasoju & T. Vishwakarma (2025). The authors showed that properly selected optimisations can significantly reduce processing time without losing accuracy. Their results correlate with current observations of reduced inference time using CTranslate2 and quantisation, which confirmed the effectiveness of hardware-software optimisations. L. Zhang *et al*. (2025) proposed LoRA-INT8 Whisper, a framework for Cantonese language recognition on edge devices with low resource costs. The authors demonstrated that quantisation to INT8 combined with LoRA layer dimensionality reduction allows for real-time processing under limited resources, with minimal loss of accuracy. The current study also noted a reduction in inference time and peak VRAM consumption during quantisation, but the WER in the experiments remained zero. These differences can be explained by differences in audio file types.

A. Orhon *et al*. (2025) presented WhisperKit, a real-time solution on user devices using billion-scale transformers. The authors focused on efficient memory organisation and parallel batch processing to achieve low latency while maintaining high accuracy. They showed that architectural optimisations and hardware integration critically affect performance. At the same time, the obtained results of

current study for WER were somewhat inconsistent: the current audio file was shorter, with different noise characteristics, which led to zero recognition error rates, although the inference performance (processing time and VRAM) was consistent. N. Wang *et al*. (2022) proposed a comprehensive approach to deep transformer compression, including pruning, quantisation, and knowledge distillation. The researchers showed that the combination of these methods allows reducing the model size by more than 70%, while maintaining close to the original accuracy on standard FFMpeg tasks. Compared to the conducted study, the application of Whisper Turbo quantisation also reduced the model size and VRAM, increasing the inference speed.

S.M. Ebrahimipour *et al.* (2025) focused on latency-oriented pruning and quantisation of self-trained transformers for edge devices. The authors showed that the combination of these methods allows for a significant reduction in inference time and memory consumption without a critical decrease in accuracy. Quantisation in the study also reduced processing time by 20-30% and reduced VRAM. The publication of S. Khadse (2025) analysed the prospects for using small language models and resource-saving artificial intelligence technologies for edge deployment. The author emphasised that model compression, computational optimisation, and adaptation to local devices critically affect performance, which confirms the validity of the experiments with CTranslate2 and quantisation. At the same time, the author noted that excessive compression can reduce recognition quality, which is consistent with current observations: optimal quantisation parameters allowed balancing performance and WER, while a more aggressive reduction in model accuracy led to a significant deterioration in recognition.

In the article by V. Potocnik *et al*. (2024), it was shown that optimising the inference of foundation models on a multicore Reduced Instruction Set Computer, version Five (RISC-V) platform with numerous small cores allows achieving high performance at low energy costs. The authors used specialised distributed inference algorithms and optimised libraries to manage calculations on microcores. Compared with the conducted study, the results confirmed the effectiveness of inference optimisation to reduce processing time and reduce energy consumption. However, in the current experiment, the absolute values of inference time on GPU were higher, which is explained by the difference in hardware architectures: RISC-V with numerous small cores provides parallel processing, while these tests were performed on a standard GPU platform with a smaller number of threads for simultaneous calculation.

In turn, M.M. Kalhoro & M. Masab (2025) investigated lightweight online ASR methods, emphasising the importance of increasing the attention of the model for real-time recognition accuracy. The authors showed that even a small improvement in attention mechanisms can reduce WER and improve recognition stability on complex audio, in particular in noisy conditions. At the same time, R. Vergallo *et al*. (2025) performed a large-scale evaluation of quantisation to reduce the energy footprint of deep learning models. The researchers showed that 8-16-bit quantisation schemes significantly reduce energy consumption without significant loss of accuracy on supervised datasets. The results obtained are partially correlated: a decrease in VRAM and inference time was recorded for Whisper Turbo quantisation. This discrepancy is explained by differences in the test cases, since the experiments included real audio files with different noise backgrounds and accents, which complicated the recognition accuracy.

C. Wu *et al.* (2025) showed that quantisation, pruning, and specialised inference engines are critical for practical implementation of real-time ASR. These findings correlate with the current results: using CTranslate2 together with int8/FP16 quantisation significantly reduced inference time and memory consumption without a proportional increase in computational complexity. In turn, C. Feng *et al*. (2025) showed that when quantising to 8 bits, a significant reduction in computational costs and memory size can be achieved without a significant loss of accuracy, while with a more aggressive reduction in bit depth (4/3 bits), the recognition error increases markedly. Compared to the conducted study, where the use of hybrid quantisation int8_float16 and optimisation via CTranslate2 allowed reducing the inference time and the consumption of video memory while maintaining a zero WER, the authors' work confirmed the key trend: 8-bit quantisation is an effective compromise between speed and accuracy. Thus, the comparison showed that the results of the conducted study generally correlate with the conclusions of other authors on the effectiveness of inference optimisation and quantisation to reduce processing time and resources. At the same time, discrepancies in recognition accuracy are explained by different testing conditions, the nature of the audio and hardware platforms, which emphasises the need for a comprehensive approach when evaluating the performance of ASR models in different usage scenarios.

Conclusions

This paper investigated the effectiveness of optimizing the Whisper speech recognition system through the integration of CTranslate2 and hybrid quantisation to reduce inference time and lower video memory requirements on GPUs with limited resources. The experiments showed that the basic Whisper Turbo configuration with the standard Python-pipeline and CUDA provided the maximum quality of speech recognition (WER = 0), but was characterised by a significant inference delay (8.5 s per audio file) and high video memory consumption (4.9 GB). Such indicators significantly limited the scalability of the system and its practical use on GPUs with limited VRAM, especially in real-time scenarios and parallel processing of audio streams. The integration of CTranslate2 allowed optimising the computational pipeline through operator fusion and more efficient memory management. In this configuration, the inference time was reduced to 4.9 s, and the VRAM consumption was reduced to 1.8 GB, which corresponds to approximately 1.7-fold acceleration and a reduction in memory costs by

almost 63% compared to the basic implementation. Importantly, these optimisations did not degrade the recognition quality, as the WER remained zero. Further application of hybrid quantisation int8_float16 provided additional efficiency gains: processing time was reduced to 3.8 s, and video memory use was reduced to 1 GB. Taken together, this resulted in an overall speedup of about 2.2× and an almost fivefold reduction in VRAM requirements (4.9×) compared to the standard Whisper configuration, while maintaining WER = 0. Analysis of architectural features confirmed that the main contribution to latency is the quadratic complexity of self-attention in the encoder and the autoregressive decoder with cross-attention, further complicated by beam search. The optimised CTranslate2 backend reduced the overhead of memory access and the number of intermediate tensor operations, while quantisation reduced the amount of data transferred and the computational cost of matrix multiplications.

The study was limited to using a single high-quality audio file, assessing accuracy only by the WER indicator, and testing on a single hardware platform, which does not allow fully generalising the results to different recording conditions, types of errors, and classes of computing devices. In future studies, it is important to expand the audio dataset, in particular by adding noisy recordings, which will allow assessing the robustness of the models to noise influences. In addition, it is necessary to compare the results obtained using different computational libraries and quantisation schemes to identify the most effective approaches. It is also necessary to assess the impact of optimisations on more complex tasks, such as semantic classification and tone analysis, where even minor errors in transcription can significantly affect the accuracy of message interpretation.

Acknowledgements
None.

Funding
The study was not funded.

Conflict of Interest
None.

References

[1] Ala-Rantala, J. (2025). *Low-latency voice-guided visual content generation using generative AI models*. (Master's thesis, Tampere University, Tampere, Finland).

[2] Cao, Y. (2025). Performance evaluation of whisper-series speech transcription models on raspberry Pi. In *Proceedings of the tenth ACM/IEEE symposium on edge computing* (article number 59). New York: ACM. doi: 10.1145/3769102.3774244.

[3] Chettiar, F.F., Lahrani, H., & Rathor, K. (2025). Multilingual video translation and speech synthesis: A deep learning approach for seamless language adaptation. In *Proceedings of the international conference on interdisciplinary approaches in technology and management for social innovation* (pp. 1-6). Gwalior: IEEE. doi: 10.1109/IATMSI64286.2025.10985230.

[4] Ebrahimipour, S.M., Mozafari, S.H., Clark, J.J., Gross, W.J., & Meyer, B.H. (2025). Latency-aware pruning and quantization of self-supervised speech transformers for edge devices. *ACM Transactions on Embedded Computing Systems*. doi: 10.1145/3746638.

[5] El Bahri, J., Kouissi, M., & Begdouri, M.A. (2025). Sustainable speech recognition: Energy, carbon, and performance comparison of whisper (base and large) and google speech-to-text V2 (Chirp/USM). In H. Gibet Tani, M. Kouissi, M. Ben Ahmed, B.A. Abdelhakim & L. Elaachak (Eds.), *Energy-efficient algorithms and systems in computing: Optimizing performance and sustainability through advanced computational methods* (pp. 213-226). Cham: Springer. doi: 10.1007/978-3-032-04114-2_14.

[6] Feng, C., Lin, Y., Zhuo, S., Su, C., Ramakrishnan, R.K., Yuan, Z., & Zhang, X. (2025). Edge-ASR: Towards low-bit quantization of automatic speech recognition models. *ArXiv*. doi: 10.48550/arXiv.2507.07877.

[7] Hung, N.T., Phuc, V.H., Dung, N.T., Duc, L.X., Nhu, M.T., & Van, P.T. (2025). Effwhis: A proposed efficient approach for speech-to-text streaming whisper. In *Proceedings of the 7th international conference on knowledge and system engineering* (pp. 1-6). Da Lat: IEEE. doi: 10.1109/KSE68178.2025.11309493.

[8] Hwang, M.H., Shin, J., & Bang, J. (2026). V-APA: A voice-driven agentic process automation system. *Computer Speech & Language*, 99, article number 101938. doi: 10.1016/j.csl.2026.101938.

[9] Kalhoro, M.M., & Masab, M. (2025). Light-weight online real-time ASR: A bit more attention is needed. *Authorea Preprints*. doi: 10.22541/au.174914695.58777421/v1.

[10] Kasoju, A., & Vishwakarma, T. (2025). Optimizing transformer models for low-latency inference: Techniques, architectures, and code implementations. *International Journal of Science and Research*, 14, 857-866. doi: 10.21275/SR25409073105.

[11] Khadse, S. (2025). Small language models and efficient AI: The future of sustainable, accessible intelligence a comprehensive analysis of model compression, edge deployment, and resource-efficient AI systems. *SSRN*. doi: 10.2139/ssrn.5664971.

[12] Kim, S. (2024). *Full stack approach for efficient deep learning inference*. (Doctoral dissertation, University of California, Berkeley, USA).

[13] Maurya, M., Zaheer, M., Mohammad, N., Siddiqui, S., Khan, M., & Akram M. (2025). Speech recognition technologies: Design, challenges, and real-world applications. *International Journal of Innovative Research in Computer Science and Technology*, 13(3), 55-61. doi: 10.55524/ijircst.2025.13.3.9.

[14] Menshawy, A., & Fahmy, M. (2025). *LLMs in Enterprise: Design strategies, patterns, and best practices for large language model development*. Birmingham: Packt Publishing Ltd.

[15] Moslem, Y., Morán, J.J., Gonzalez-Gomez, M., Al Farouq, M.H., Abdou, F., & Deb, S. (2025). SpeechT: Findings of the first mentorship in speech translation. In *Proceedings of machine translation summit* 20th (Vol. 2, pp. 67-74). Geneva: European Association for Machine Translation.

[16] Mrozek, Ye. (2024). Analysis of modern approaches to speech recognition tasks. *Control Systems & Computers*, 4(308), 39-49. doi: 10.15407/csc.2024.04.039.

[17] Nakhod, O. (2025). Automatic recognition of Ukrainian speech based on deep learning. *Collection of Scientific Papers "ΛΟΓΟΣ"*, 24, 218-220. doi: 10.36074/logos-24.01.2025.043.

[18] Orhon, A., Okan, A., Durmus, B., Nagengast, Z., & Pacheco, E. (2025). WhisperKit: On-device real-time ASR with billion-scale transformers. *ArXiv*. doi: 10.48550/arXiv.2507.10860.

[19] Potocnik, V., Colagrande, L., Fischer, T., Bertaccini, L., Pagliari, D.J., Burrello, A., & Benini, L. (2024). Optimizing foundation model inference on a many-tiny-core open-source risc-v platform. *IEEE Transactions on Circuits and Systems for Artificial Intelligence*, 1(1), 37-52. doi: 10.1109/TCASAI.2024.3459412.

[20] Rangappa, P., *et al*. (2025). Speech data selection for efficient ASR fine-tuning using domain classifier and pseudo-label filtering. In *Proceedings of the IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 1-5). Hyderabad: IEEE. doi: 10.1109/ICASSP49660.2025.10888138.

[21] Thorbecke, I., Zuluaga-Gomez, J.P., Villatoro-Tello, E., Kumar, S., Rangappa, P., Burdisso, S., Motlicek, P., Pandia, K., & Ganapathiraju, A. (2024). Fast streaming transducer ASR prototyping via knowledge distillation with whisper. In *Findings of the Association for Computational Linguistics: EMNLP 2024* (pp. 16747-16762). Miami: Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.976.

[22] Trabelsi, A., Werey, L., Warichet, S., & Helbert, E. (2024). Is noise reduction improving open-source ASR transcription engines quality? In *Proceedings of the 16th international conference on agents and artificial intelligence* (pp. 1221-1228). Rome: Science and Technology Publications. doi: 10.5220/0012457100003636.

[23] Vergallo, R., Aprile, M., Cruz, L., Vadacca, R., & Mainetti, L. (2025). Large-scale evaluation of quantization for reducing the energy footprint of deep learning models. *SSRN*. doi: 10.2139/ssrn.5719661.

[24] Wang, N., Liu, C.C., Venkataramani, S., Sen, S., Chen, C.Y., El Maghraoui, K., Srinivasan, V., & Chang, L. (2022). Deep compression of pre-trained transformer models. In *Proceedings of the 36th international conference on neural information processing systems* (pp. 14140-14154). Ney York: Curran Associates.

[25] Wu, C., Pan, Y., Wu, H., & Ning, L. (2025). Integrating speech recognition into intelligent information systems: From statistical models to deep learning. *Informatics*, 12(4), article number 107. doi: 10.3390/informatics12040107.

[26] Wu, X., Zhang, Y., & Feng, B. (2023). English pronunciation quality evaluation system based on continuous speech recognition technology for multi-terminal. *Journal of Physics: Conference Series*, 2632, article number 012024. doi: 10.1088/1742-6596/2632/1/012024.

[27] Zhang, L., Wu, S., & Wang, Z. (2025). LoRA-INT8 whisper: A low-cost Cantonese speech recognition framework for edge devices. *Sensors*, 25(17), article number 5404. doi: 10.3390/s25175404.

[28] Znotins, A., Gosko, D., & Gruzitis, N. (2025). LATE: Open source toolkit for Latvian and latgalian speech transcription. In *Proceedings of the annual conference of the international speech communication association, INTERSPEECH* (pp. 306-307). Rotterdam: ISCA.

Підвищення ефективності обробки аудіопотоків на базі Whisper з інструментами CTranslate2 та FFMpeg

Владислав Радін

Аспірант
Національний університет «Київський авіаційний інститут»
03058, просп. Любомира Гузара, 1, м. Київ, Україна
https://orcid.org/0009-0009-1101-6888

Мирослав Рябий

Кандидат технічних наук, доцент
Національний університет «Київський авіаційний інститут»
03058, просп. Любомира Гузара, 1, м. Київ, Україна
https://orcid.org/0000-0002-9651-9135

Анотація. Актуальність дослідження полягає в необхідності підвищити продуктивність і масштабованість систем автоматичного розпізнавання мовлення на пристроях із обмеженими ресурсами, що обумовлює мету роботи – оптимізувати Whisper за допомогою інтеграції CTranslate2 для прискорення обчислень та FFmpeg для уніфікованої підготовки аудіоданих. Експериментальні дослідження проводилися з використанням моделі Whisper Turbo на графічному процесорі з підтримкою платформи обчислень Compute Unified Device Architecture. Порівнювалися базовий конвеєр на мові програмування Python, оптимізований механізм виконання інференсу через CTranslate2 та конфігурація з гібридною квантизацією у форматі int8_float16. Ефективність оцінювалася за показниками часу виконання передбачення (інференсу), використання відеопам'яті та точності автоматичного розпізнавання мовлення (Word Error Rate). Експериментальні результати показали, що базова конфігурація Whisper Turbo забезпечувала максимальну точність розпізнавання (Word Error Rate = 0), однак характеризувалася високою затримкою інференсу (8,5 с на аудіофайл) і значним споживанням відеопам'яті (4,9 ГБ). Інтеграція CTranslate2 скоротила час обробки до 4,9 с (прискорення 1,7×) та зменшила використання Video Random Access Memory до 1,8 ГБ (-63 %) без втрати якості. Подальше застосування гібридної квантизації int8_float16 забезпечило зниження часу інференсу до 3,8 с і скорочення споживання пам'яті до 1 ГБ, що відповідає загальному прискоренню близько 2,2× та майже п'ятикратному (4,9×) зменшенню вимог до Video Random Access Memory порівняно зі стандартною реалізацією, при незмінному Word Error Rate = 0. Отримані результати підтвердили ефективність поєднання CTranslate2 і гібридної квантизації для побудови високопродуктивних систем Automatic Speech Recognition реального часу без компромісу в точності. Висновки підтвердили практичну придатність запропонованої конфігурації для багатокористувацьких сервісів і edge-сценаріїв без компромісу між швидкістю та точністю. Результати дослідження можуть бути використані розробниками систем автоматичного розпізнавання мовлення для оптимізації моделей на графічних процесорах з обмеженим обсягом пам'яті, компаніями, що надають потокові аудіо- та багатокористувацькі сервіси

Ключові слова: квантизація; автоматичне розпізнавання мовлення; ф'юзування операторів; відеопам'ять; ресурсоефективність