Benchmarking Transformer-Based Transcription on Embedded GPUs for Space Applications

by

Marika Schubert

B.S. Electrical Engineering and Applied Mathematics, University of Colorado Boulder, 2019

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Electrical and Computer Engineering

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Marika Schubert

It was defended on

March 31, 2021

and approved by

Amro El-Jaroudi, PhD, Associate Professor, Electrical and Computer Engineering

Jingtong Hu, PhD, Associate Professor, Electrical and Computer Engineering

Thesis Advisor: Alan D. George, PhD, Department Chair, Electrical and Computer Engineering

Benchmarking Transformer-Based Transcription on Embedded GPUs for Space Applications

Marika Schubert, M.S.

University of Pittsburgh, 2021

Speech transcription is a necessary tool for backend applications commonly found in voice assistants. Transcription is typically performed using cloud-based servers or custom hardware, but those resources are not always amenable to space environments due to size, weight, power, and cost constraints. Therefore, it is important to determine the performance of and optimal conditions for running transcription on hardware that is feasible for deployment in a space application. This research investigates and evaluates the performance of the wav2vec2 speech transcription engine, the current state-of-the-art model for this domain with and without optimizations. The target hardware, the NVIDIA Xavier NX Jetson embedded GPU, was chosen for its modern GPU architecture and small form factor. In addition to examining the input scaling behavior, we evaluate the hyperparameters of the clustered attention optimization, and average power and energy for inference relative to the operating power mode of the device. The clustered attention model outperformed the improved-clustered model for large input sizes, but the wav2vec2 model without clustering performed better for small input sizes. The clustered model energy per inference (13.90 J) was less than energy per inference of the improved-cluster model (15.03 J) and the vanilla softmax model (15.85 J). All models meet real-time speech processing requirements necessary to perform onboard inference entirely on a space system.

Table of Contents

List of Tables

List of Figures

Preface

This work was supported by SHREC industry and agency members and by the IUCRC Program of the National Science Foundation under Grant No. CNS-1738783.

1.0 Introduction

As human space exploration pushes towards lunar orbit with the planned Gateway outpost [14], there will be limited connectivity with traditional terrestrial resources. Current operations on the International Space Station rely heavily on ground personnel to assist the orbiting crew with procedures and vehicle operations. For lunar missions, the need for a conversational interface capable of assisting astronauts, similar to a chatbot, will become critical for activities where it is difficult to consult a screen or manual. This chat interface will serve in place of humans in mission control for tasks such as assisting in procedures, helping to locate objects, and relaying information about the vehicle with the crew. Chatbots, like the Amazon Alexa voice assistant and supporting services [1], rely on speech transcription as the enabling technology for their backend, text-based natural language processing (NLP) applications. The vast majority of similar applications are tailored to consumer electronics, which are ill-suited for remote, extreme environments such as those found in space. An example conceptional flow for a practical audio system for Gateway is shown in Fig. 1.

Transcription for space applications remains a vital but undetermined piece of such a conversational system. Transcription as a whole typically involves preprocessing to reduce noise and isolate speech. This is followed by feature vector calculation and frequency analysis used as input to a NLP or machine-learning-based model. These models translate feature vectors to an intermediate language representation. Frequently, the representation used is phonemes, which are the distinct sounds that compose human speech. This intermediate representation is then provided to a language model which translates the input sequence to human-readable text. A common basis for a language model is connectionist temporal classification (CTC). These models are able to generate an output sequence where the input and output sequences may not be aligned, or where the input has potentially redundant characters or symbols [7]. Depending on the framework, there may be additional compensation for pronunciation of different words. An outline of a representative framework can be seen in Fig. 2.

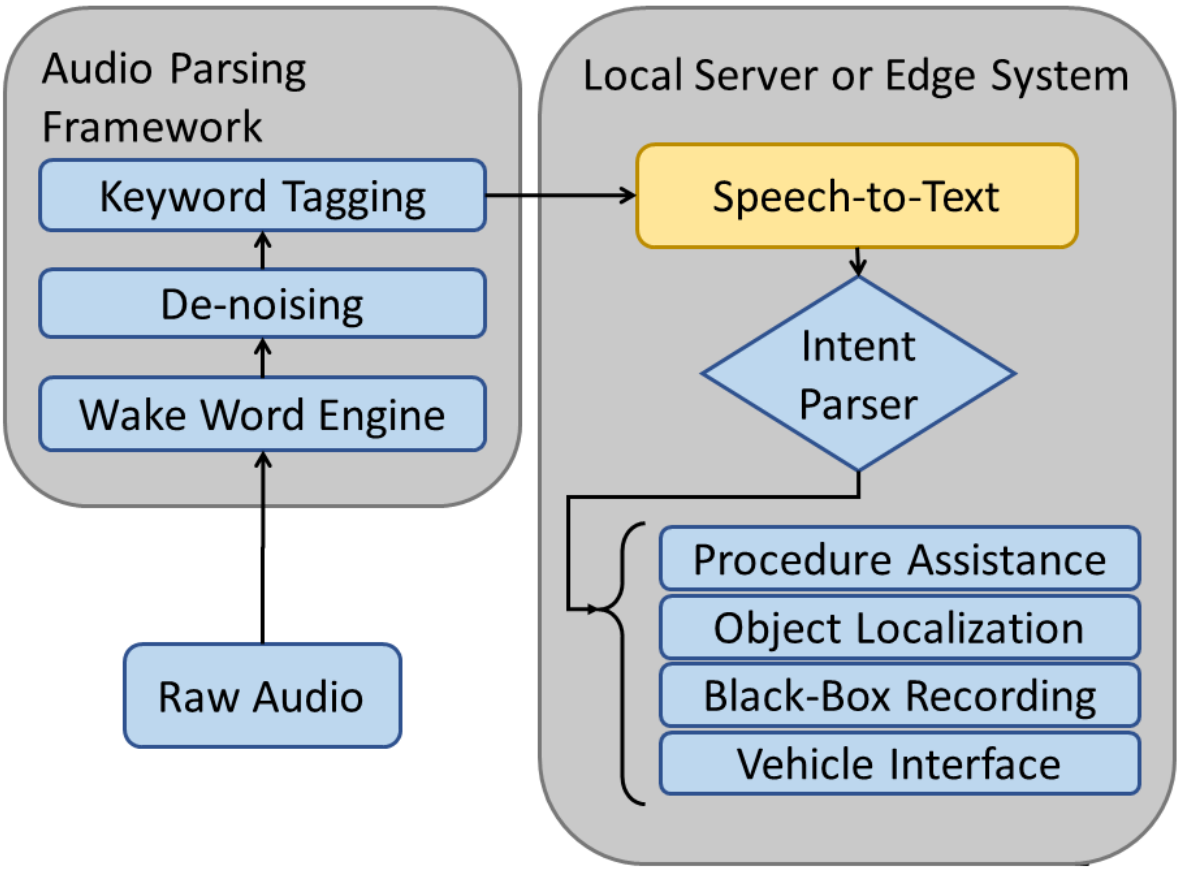

Figure 1: Conceptual layout of space-based audio processing framework.

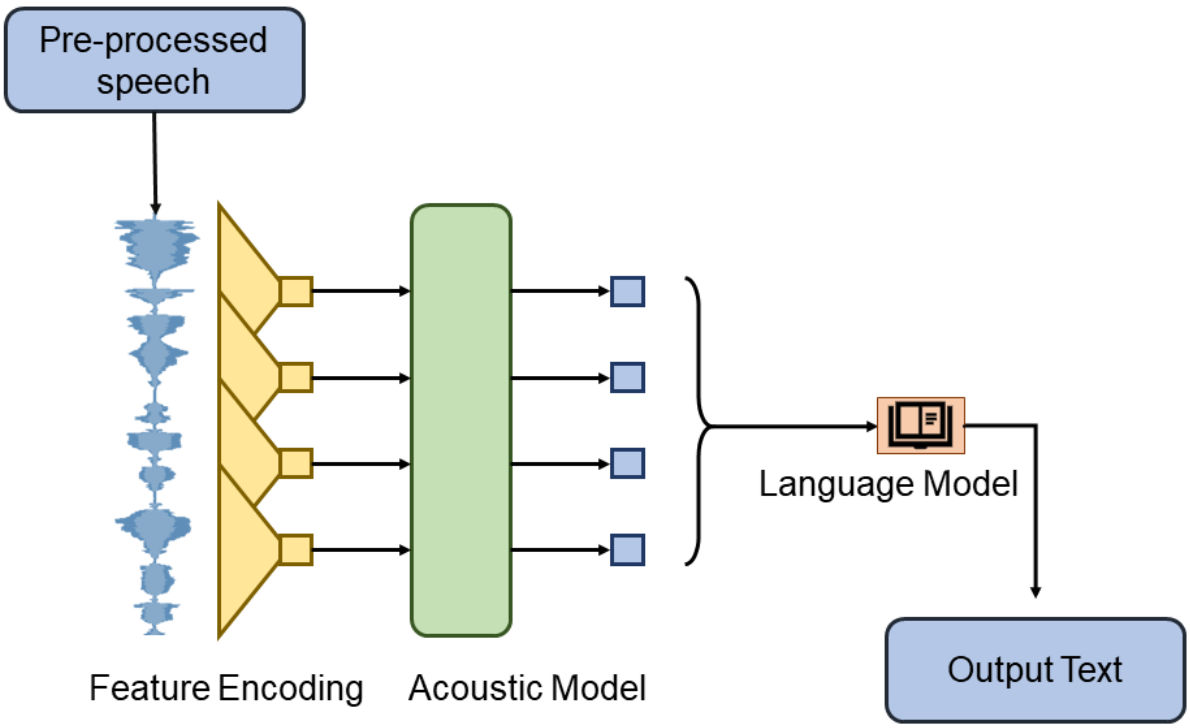

Figure 2: High-level diagram of transcription frameworks

Most systems capable of transcription do so with the aid of cloud servers, including those provided by Google [6], IBM [24], and Amazon [10]. Communication to a lunar (moon-based) spacecraft from Earth, for instance, would incur a minimum of 2.25-second round-trip time. This minimum latency figure would be complicated by other factors such as loss of signal (LOS), where the link for such a communication could be missing due to spacecraft position or satellite availability [23]. This high latency and complexity of communication would reduce usefulness of the voice system and impact crew productivity when real-time feedback was necessary. This issue would be further magnified if this system were deployed in even more remote environments such as Mars.

Without a remote server, the next source of assistance may come from a edge accelerator such as a graphics processing unit (GPU). Currently, there is a general interest in flying GPUs both for their traditional rendering and display use cases, but also for allowing machine learning in remote environments [4]. For reasons of constrained size, weight, power, and cost (SWaP-C), it is valuable to assess the feasibility of running transcription on an embedded GPU.

The speech-embedding framework benchmarked in this research was wav2vec2, which is a state-of-the-art (SOTA) transformer-based model that translates sounds to a latent speech representation using convolutional neural network feature encoding followed by a transformer acoustic model [26, 3]. Additionally, two optimizations known as fast-transformers and attention clustering are added to reduce runtime [9, 28].

This research benchmarks the runtime of the wav2vec2 model on a representative embedded GPU for space applications. The key contributions of this paper are the comparisons of number of clusters in wav2vec2 optimizations, insights into power and energy consumption during inference, and the effect of varying GPU power limits on the runtime for this class of transcription model.

2.0 Related Work

This section describes the current SOTA of speech transcription, both with transformers and more general recurrent structures. It additionally explores some of the models used for complex language tasks and how those models inform the choice of wav2vec2 for speech-to-text computation on an edge device. Moreover, it will introduce the issue of transformer scaling, as well as optimizations chosen to overcome this issue.

While the subject of speech processing for space applications is a niche area, transcription as a whole has a large body of supporting research driven by the push to integrate voice control into consumer electronics and software. Transcription, also described as speech-to-text, is a necessary function for converting audio signals for use in NLP backends designed to accept text as an input. Many of the most prominent transcription applications have, therefore, been created by companies like Amazon, Facebook, and Google, as well as large research labs. Some systems, like IBM's Watson [24] and NVIDIA's emerging Jarvis framework [19] are available via API. Other transcription apps are in precompiled formats for specific hardware, like Google's on-device speech recognition for Pixel phones [8]. Open-source code for transcription, however, is beneficial for the development of a local, offline application for space missions.

Transcription is considered a sequence-to-sequence learning task. The goal of sequence-to-sequence tasks is to convert a sequential signal (e.g. audio) to a sequential output of a different domain (e.g. a string of text). There are a variety of machine learning classes that are able to perform this type of task. The models used typically require memory, recurrence, or inclusion of output states.

One of the early demonstrations of end-to-end automatic speech recognition was found in Deep Speech 2 [2]. Deep Speech 2 performed transcription using several layers of recurrent neural networks (RNNs) and convolutional layers with CTC loss for training. This widely adopted loss metric allows for the translation of sequences that do not have strict labeling alignment between their input and output. Additionally, Amodei et. al. demonstrated a model that could be deployed on GPU servers, but highlighted the viability of a fully

trained machine-learning model for speech transcription over one that required individually designed components as previously accomplished by models like Kaldi [22]. Deep Speech 2 was originally considered by the authors of this paper, but initial results demonstrated that it was too large and slow for deployment on embedded platforms.

Wav2vec is another transcription model developed with the fairseq, which is a tool developed by Facebook for a variety of NLP text generation tasks including transcription [25, 20]. Wav2vec achieved a higher accuracy than Deep Speech 2 with significantly less training data. Unlike Deep Speech 2, wav2vec was composed of two layers of cascaded convolutional neural networks. The first network is a designed to perform feature embedding from audio. This network could be pre-trained independently of the second network on unlabeled data, meaning that it was able to achieve better accuracy with the same input data. This feature encoding reduces the dimension of the input data and provides feature vectors that better represent the underlying language. The second network converts this to an interpretable context which was used to predict the next character or phoneme that would occur in audio. Compared to DeepSpeech 2, this model was able to achieve a slightly better word error rate (WER) (3.1 for Deep Speech 2 and 2.18 for a fine-tuned wav2vec2 model).

A sequence-to-sequence structure that has gained interest in recent years is known as a transformer [26]. These models consist of an encoder/decoder structure connected by an attention mechanism. This structure requires positional encoding to inform the model of the relative position of an input within a sequence, making it inefficient for memory-bandwidth-constrained hardware. A conceptual diagram of a transformer is shown in Fig 3. A mathematical description is detailed in Sec. 3.

A transformer based model frequently cited for natural language understanding (NLU) tasks is the Bidirectional Encoder Representations from Transformers (BERT) [5]. Devlin et al. demonstrated the accuracy of their model on a wide variety of NLU tasks including sentence prediction, understanding, and sentence segmentation in a variety of languages, but not translation. The utility of BERT for NLU tasks is also well demonstrated in its adoption by other researchers [11, 12]. However, it was not desirable for this research as it did not include a method of encoding audio to feature vectors or a model trained for this purpose.

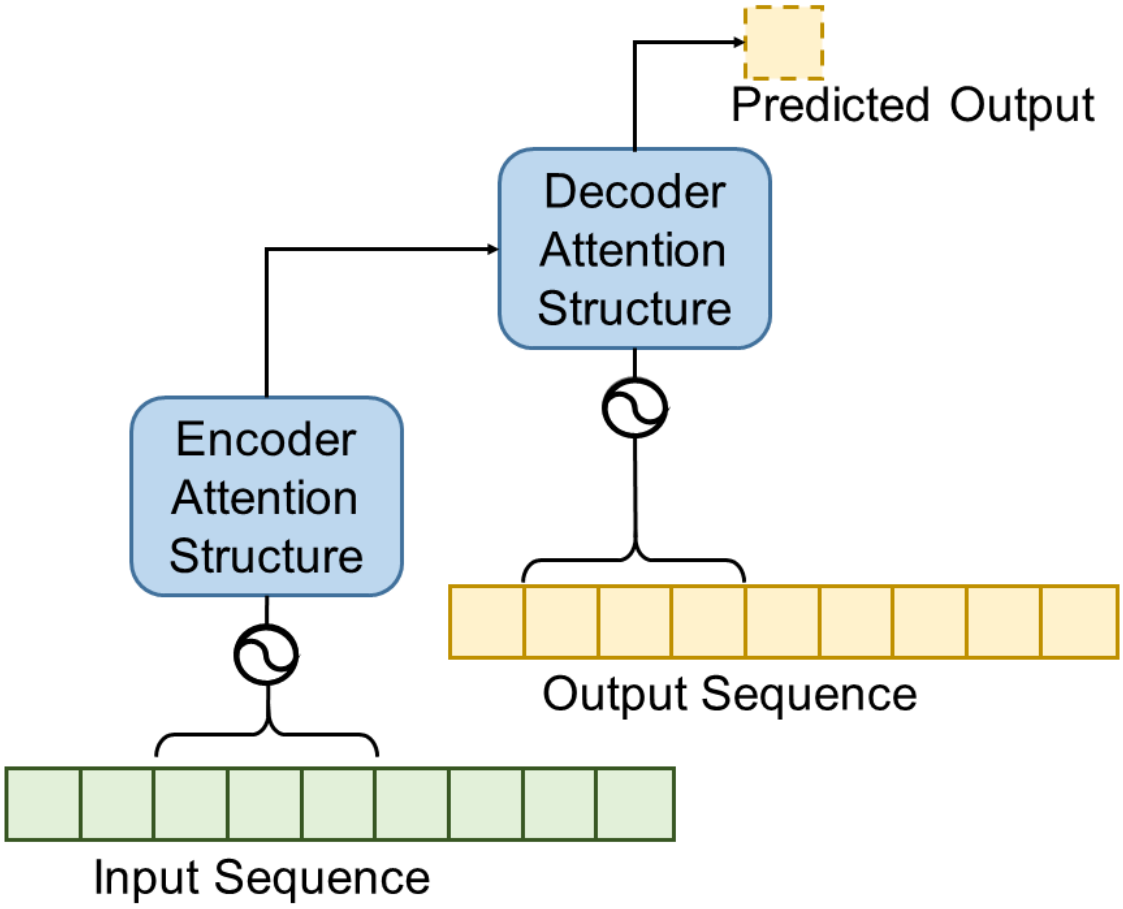

Figure 3: Conceptual diagram of a transformer.

RoBERTa is one particularly notable BERT-based model [12] for this use case. This model employed the base BERT training and examined optimizations of hyperparameters and more extensive training to achieve SOTA accuracies on several of the benchmarks that BERT had originally demonstrated. RoBERTa achieved this partly by drastically expanding its training data to include five publicly available corpora. RoBERTa additionally rebuilt the BERT base model using the fairseq repository.

The successor to wav2vec, Wav2vec2, traded wav2vec's second convolutional network for a transformer layer, the overall structure for which can be seen in Fig. 4. Additionally, wav2vec2 incorporated some of the training methods demonstrated with BERT, like selective input masking to allow the model to better generalize [5]. This allowed it to achieve SOTA word error rate (WER) on the LibriSpeech corpus using fine-tuning with a small amount of unlabeled data [3, 21]. This model demonstrates that a semi-supervised learning technique that is well suited for applications where there are a relatively small amount of labeled data for training. This will likely be the case for space applications as much of the spoken language will be jargon and acronyms unique to this domain. There are example models for wav2vec2 available as a starting point, which could be fine-tuned on more powerful terrestrial hardware for mission-specific terminology and speakers.

Wav2vec2 still suffers from the scaling issue inherent to transformers, specifically that computation time scales on the order of $\mathcal{O}(N^2)$ where N is the dimension of the input sequence. It is, however, possible to implement transformers with a slight alteration of the underlying equations to reduce the computation time prediction scaling to a factor of $\mathcal{O}(N)$ through an approximation method demonstrated in [9]. One can further optimize transformers using clustered attention [28]. Clustering partitions the input sequences and calculates the centroid of this data to use in place of the entire cluster, introducing a small but bounded error. Vyas et al. augment their clustering algorithm by additionally considering the attention keys that have the highest weights, referring to this algorithm as improved-clustering. This method is intended to overcome scenarios where there are too few clusters for the given input size or where the error introduced by clustering is too high. For the purposes of their experiment, the authors ran the RoBERTa translation model [12] using their modified processes and saw only marginally decreased accuracy for many benchmarks using improved-

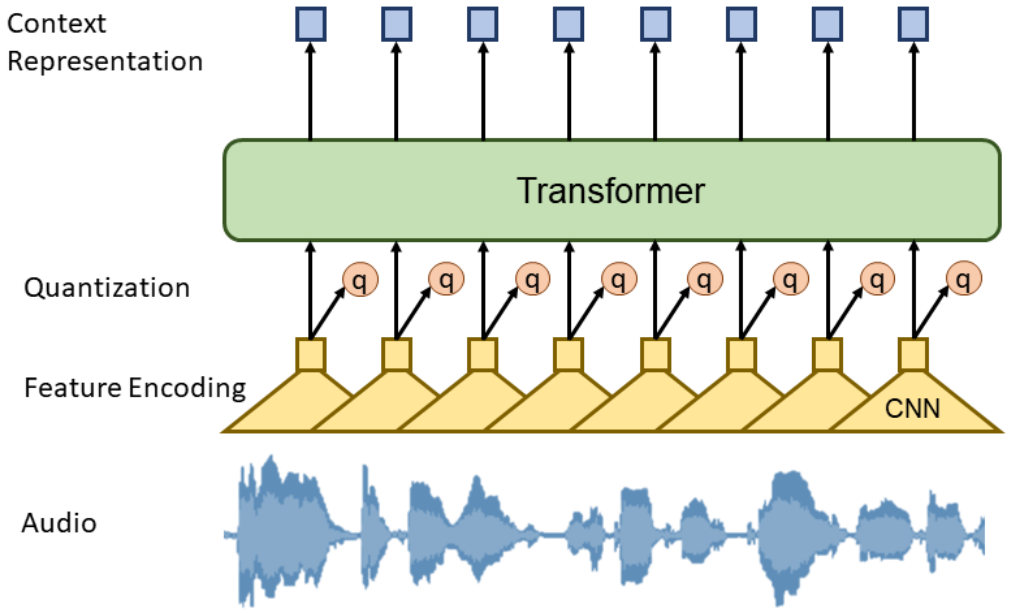

Figure 4: Conceptual diagram of wav2vec2 model.

clustering, but more complicated tasks saw more significant losses. Additionally, for short sequence lengths, the full model performed inferences faster that the clustered model.

While the usefulness of these optimized were proved in GPU hardware, there is no guarantee that those improvements will scale well across applications and varying hardware. The performance of the model presented by He et al. [8] is entirely dependent on the computing architecture provided. In [13], Narang et al. examine various optimizations for transformers as well as several fundamental model variations on Google's tensor processing units (TPUs). In their research, most of the modifications studied were ineffectual on the TPU hardware, despite their supposed benefit from their original publications. The main conclusions of the research are that models may not show the same level of improvements across codebases or applications, meaning that were too dependent on a particular framework or transformer implementation. For practical purposes, this means that a transformer likely needs to be optimized for a particular codebase and hardware to perform well, but conclusions may not transfer when one of these underlying states change. For this reason, it is necessary to test optimizations in the desired application to understand their contextual use.

3.0 Background

This section will describe the optimizations present in the wav2vec2 approximation used in this research. The derivation of transformer models, as well as the exact derivations for these optimizations can be found in the referenced supporting literature [9, 26, 28], but a highlight of the relevant mathematical changes are presented. Details about the target device architecture are also provided.

3.1 Transformers

For a set of general scaled dot-product attention unit, define a set of queries (inputs) as Q, a matrix of keys K, and a matrix of values V. Let d_k be the dimension of the keys. From a high level, this describes a mapping from keys to an output. The overall equation describing a transformer is seen in Equ. 3-1 [26]. Multi-headed attention, like implemented in wav2vec2, is a concatenation of several of these units.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3-1}$$

Note that attention is composed of matrix multiplications and a softmax operation. While GPUs are well equipped for highly parallel operations such as matrix multiplication, softmax is more difficult to implement well on a GPU. Softmax maps a given input vector (or matrix here) to the range $(0, 1)$ using exponentials. For a vector $w = w_1, w_2...w_J$, the softmax σ is shown in Eq 3-2. Note that this is is primarily a normalization function.

$$\sigma(w)_i = \frac{e^{w_i}}{\Sigma_{j=1}^{J} e^{w_j}} \tag{3-2}$$

A typical transformer operation cost scales relative to input size N, the dimension of the queries d_Q, and the dimension of the values d_V in Equ. 3-3.

$$\mathcal{O}(N^2\ max(d_Q, d_V)) \tag{3-3}$$

3.2 Fast-Transformers

The fast, linear transformers discussed in Sec. 2 are a reformulation of the original transformer that has an runtime that scales linearly with respect to input sequence length [9]. This is done through a reformulation which allows the transformer to precalculate some values to reduce the cost of an input query. This creates a larger burden on the memory, but can reduce computation time overall. In [9], to linearize the transformer equation, the authors applied two techniques. The first was to linearize the calculation of the transformer itself without regard to the softmax function (abstracting it to a row-wise calculation $\phi(\cdot)$). Through their derivation, they proved the conclusion in Eq 3-4.

$$\left(\phi(Q)\phi(K)^T\right)V = \phi(Q)\left(\phi(K)^T V\right) \tag{3-4}$$

With this reformulation, one could pre-calculate $\left(\phi(K)^T V\right)$, allowing for the values to be reused across different queries. This assumption does require that $\phi(\cdot)$ can be calculated in this way.

The authors replaced the exponential calculation in the softmax kernel with a second-degree polynomial approximation. This process reduces the complexity to $\mathcal{O}(Nd_Q^2 d_V)$. This is preferable to Equ. 3-3 when $N > d_Q^2$, which is true when the input sequence is very large. When this is not true, the authors implement a exponential linear unit as the feature vector calculation reducing the complexity to $\mathcal{O}(Nd_Q 2d_V)$. Conceptually, this change in calculation procedure is shown in Fig 5.

With wav2vec2, the CNNs reduce the dimensionality of the input vectors, but the result is still large. Additionally, the transformer model has eight attention heads in the base model which means any speedup in the transformer calculation should be seen in the final performance. This optimization is a strong candidate for optimizing wav2vec2.

3.3 Clustered Attention

Clustered attention is another method for reducing runtime that is employed by this research [28]. This method involves calculating an intermediate matrix reference to the *centroid matrix* which contains the centroids of non-overlapping clusters S. The calculation of this matrix is show in Equ. 3-5 where query vectors Q_j are divided into clusters of size C.

$$Q_j^c = \frac{\Sigma_{i=1}^{N} S_{ij} Q_i}{\Sigma_{i=1}^{N} S_{ij}} \tag{3-5}$$

Q_j^C is used in place of the original query matrix Q, reducing the dimension of the queries by a factor equal to the cluster size C. Complexity is then reduced to the expression in f. 3-6.

$$\mathcal{O}(NC\max(d_Q, d_V)) \tag{3-6}$$

Note that this is most useful where $C << N$, which should always be true. This is beneficial for an architecture like wav2vec2 which has large input sequences, so most potential cluster sizes should provide measurable speedup.

3.4 Improved Clustering

Improved clustering is an additional computation step on top of clustering designed to reduce the error created by this approximation. It does this by by creating a distribution based on the k keys with the highest attention values. To conform with the paradigm of softmax (all outputs summing to 1), the results have to be normalized adding additional complexity. For the sake of clarity, subsequent calculations use the definition in Equ. 3-7. Note that Q^C is defined in Equ. 3-5

$$A^C = \text{softmax}\left(\frac{Q^C K^T}{\sqrt{d_K}}\right) \tag{3-7}$$

To make this calculation, Vyas et al. introduce a sparse matrix T where $T_{ij} = 1$ where the i-th key is the top k keys and $Tij = 0$ otherwise. They use this matrix to calculate weights $\hat{m}_j$ as shown in Equ. 3-8..

$$\hat{m}_j = \Sigma_{i=1}^{N} \left[T_{ji} A_{ji}^{C} \right] \tag{3-8}$$

These $\hat{m}_j$ are then used to weight an alternate normalization function shown in Equ. 3-9. Note that this calculation only occurs for values where $T_{ij} = 1$.

$$M_{il} = \frac{\hat{m}_j e^{Q_i K_l^T}}{\Sigma_{r=1}^{R} T_{rj} e^{Q_i K_r^T}} \tag{3-9}$$

These weights are then substituted into the output of the softmax layer as shown in Equ. 3-10. This operation in shown in the diagram in Fig. 5.

$$\hat{A}_{il} = \begin{cases} M_{il} & \text{if } T_{jl} = 1 \\ A_{jl}^{C} & \text{else} \end{cases} \tag{3-10}$$

The matrix $\hat{A}$ is similar to the A^C, except where $T_{ij} = 1$. At these points, the value is replaced by the normalized attention for the top k keys. Note that $\hat{A}$ still needs to be multiplied by V to compute the full attention. This mechanism overall adds $\mathcal{O}(Nk \max(d_k, d_v))$ asymptotic complexity, and so is expected to run slower, but is potentially more accurate than the centroid method on its own. A graphical summary of the different types of attention used in this paper are shown in Fig. 5.

3.5 Embedded GPUs

NVIDIA, known for its high-performance consumer- and server-grade GPUs, has an additional line of system-on-module (SoM) GPU platforms. These heterogeneous architectures combine an ARM CPU with an NVIDIA GPU on the same die and packaged in an embedded form factor. The entire system can be constrained to meet the requirements of battery-powered environments. Compared to their server-grade counterparts, these boards attain only a fraction of the memory bandwidth (∼50GB/s vs ∼900GB/s) but also operate

at significantly lower power (∼15W vs ∼300W) [16, 17]. For these reasons, embedded GPUs would be able to provide acceleration for sufficiently small or optimized models onboard spacecraft.

While embedded GPUs exhibit desirable SWaP-C properties, they are also less capable than their consumer- or server-grade counterparts. Most transcription models are trained on high-performance hardware, and therefore there is little data on performance for edge devices. Embedded GPUs have a shared memory path between the CPU and GPU, which limits their performance for memory-bound applications. The most important characteristics of embedded GPUs for the purpose of this research is the shared path to memory and the feasibility of deployment in space applications due to desirable SWaP-C.

This research targeted the NVIDIA Jetson Xavier NX. This system has 6 Carmel ARM-based cores (AArch64 architecture) with an NVIDIA Volta GPU and 8GB of LPDDR4x memory. The GPU portion contains 384 CUDA cores, 48 Tensor Cores, and two Deep Learning Accelerators (DLAs). Tensor Cores are designed to accelerate tensor operations, specifically matrix multiplication [17]. DLAs are a structure that accelerate other deep-learning operations like convolution [15].

The Xavier NX GPU supports five standard power modes. The configurations of these power modes are summarized in Table 1. There are two operational power budgets: 10W and 15W. Within each power budget, the main difference between power modes is number of available CPU cores and their operating frequency. This should not have an effect on the GPU operation as the function in question should be taking place exclusively on the GPU. For all modes, it is assumed that if fewer CPU cores are active, a smaller percentage of the power budget is used for the CPU, and the GPU may run at a higher power. All modes were considered as the CPU idle power was assumed to impact the maximum GPU power.

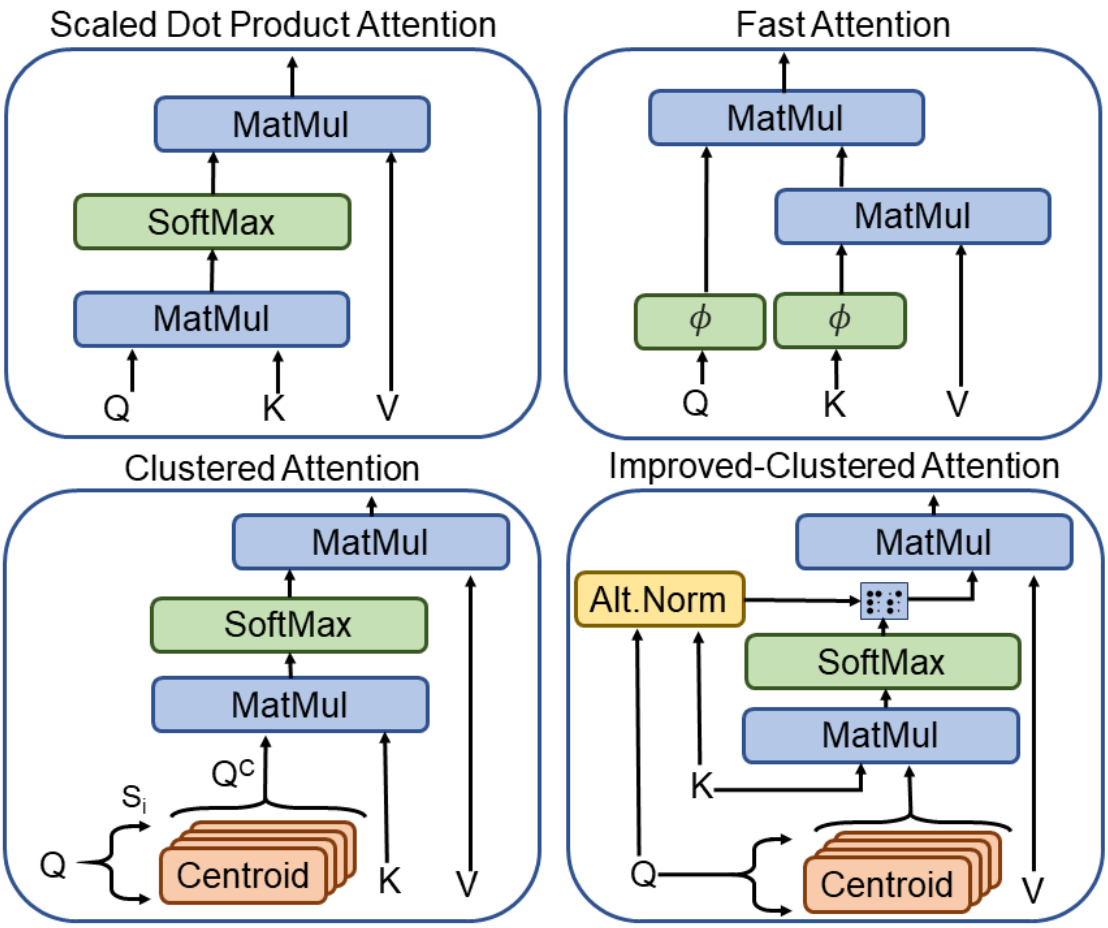

Figure 5: Types of attention.

Table 1: Summary of Xavier NX Power Modes

Mode ID	Power Budget (W)	Online CPU Count	CPU Max Frequency (MHz)	GPU Max Frequency (MHz)	Memory Max Frequency (MHz)
0	15	2	1900	1100	1600
1	15	4	1400	1100	1600
2	15	6	1400	1100	1600
3	10	2	1500	800	1600
4	10	4	1200	800	1600

4.0 Approach

This research examined an approximation of wav2vec2 based on a fork from the fairseq repository [9, 20, 28]. This fork augments wav2vec with both linear transformers and input clustering. This was done on the NVIDIA Jetson Xavier NX GPU.

4.1 Experiments

In this section, details of the experiments performed for this research. The first experiment was intented to collect data on the accuracy of the models under different hyperparameters. The next experiment tested if the model execution time was approximately linear relative to the input sequence length. The last experiment explored the effect of GPU mode on power and energy consumption of the Xavier GPU.

The original wav2vec2's SOTA accuracy has been demonstrated in literature [3] and so was considered out of scope for this study. The optimizations here do introduce small errors into the model, but it is assumed that these errors can be compensated for in model tuning.

4.1.1 Linearity

The untrained small wav2vec model was used to demonstrate the ceiling for runtimes as model pruning may be training-data dependent. To test the behavior of the model without computing accuracy, the input sequence length (32k-450k samples), model type (no clustering, clustered, and improved-clustered), number of clusters (50, 100, and 150), and GPU mode were varied. For all clustered models, the "conditional-full" option was used to prevent errors with input sizes that are too small by introducing zero padding. The codebase leveraged for this model did not allow for batch size increases, but these are not necessary for this particular application as it targets real-time inference. All calculations were performed with single-precision floating-point.

For each size, model, cluster number, and power mode, the runtime was averaged over ten model runs. GPU cache priming runs were not counted in the final results as it is assumed that the GPU cache is either used frequently enough to limit this operation, or infrequently enough that it could be primed prior to use. As a note, cache priming was required when the model was updated, and took between five and ten seconds. Therefore, priming would need to be planned for in practical settings.

4.1.2 Clustering

Using the same data set as the one collected for the previous section, but specifically examines the effect of clustering and improved-clustering on the model. This subset of data looks at three input sizes: 170kB, 920kB, and 1800kB across cluster sizes in GPU mode 2.

4.1.3 Power

Power numbers were collected on a subset of these runs using system calls during 100 evaluations with three sizes of data (small, medium, large) in each GPU mode with 50 clusters where applicable. Power measurements for the Xavier device are collected by using direct system calls to the I2C power monitor, which returns values at mW-precision at a 1-second sampling frequency. These measurements are of the CPU and GPU combined power rail [17]. Unlike other platforms in the same GPU family (like the Xavier AGX), the CPU and GPU power can not be measured independently.

4.2 Evaluation Platform

The software environment was reproduced from the Google Colab script developed by the authors of [9] in an embedded Ubuntu 18.04 installation on the NVIDIA Xavier NX. For this application, Python v3.6.12, PyTorch v1.7.0 for AArch64 (specifically compiled for Jetson platforms) [18], fast-transformers v0.3.0, sounddevice v0.4.1 and the branch of fairseq by Apoorv [27] were used.

Note that audtorch is additionally required to load the model, and v0.6.4 was used. For AArch64 architectures like this one, it was required that llvmlite be compiled from source to support the audtorch dependency numba. The audtorch package additionally requires that the sox package (v14.4.2.) is installed.

The NVIDIA Jetson Xavier NX is run in all of its default power modes, which have power budgets of 10W and 15W. These modes additionally have varying numbers of active CPU cores. All tests were run with single-precision floating-point values. With the Volta architecture, that means that the Tensor Cores were unused.

5.0 Results

This section contains data collected on the power and energy consumption, inference time, and effect number of clusters on execution time. These results verify that this model is feasible for space applications from an energy and latency perspective. Additionally, the effect of attention clustering type and size is discussed to inform practical implementations. All error bars presented represent a 95% confidence interval.

5.1 GPU Power Modes and Execution Time

The effect of the different GPU power modes is shown in Fig. 6. For this evaluation, the number of clusters was set to 50 for the clustered models, and the times are shown for the 170kB, 920kB, and 1800kB data sizes. Note that there are error bars representing the 95% confidence interval, but they are small. The first three power modes have similar execution times, regardless of the input size. However, we also see that when the power budget is reduced to 10W, the execution time of the model does increase, indicating that the GPU is being throttled to meet the power constraint. Note that, in all cases, there are small increases in the runtime as additional cores are brought online. These cores consume power, reducing power in the budget available for the GPU.

5.2 GPU Power and Energy

The average power of the varying GPU modes is shown in Fig. 7. Note that the error bars represent a 95% confidence interval. This graph shows that the power modes do strictly dictate the power of the device. An interesting feature is that the improved-clustered model does not require the full 15W to run in the higher power modes, which means it may not be optimized well for this device. Smaller models are still able to use the full device power.

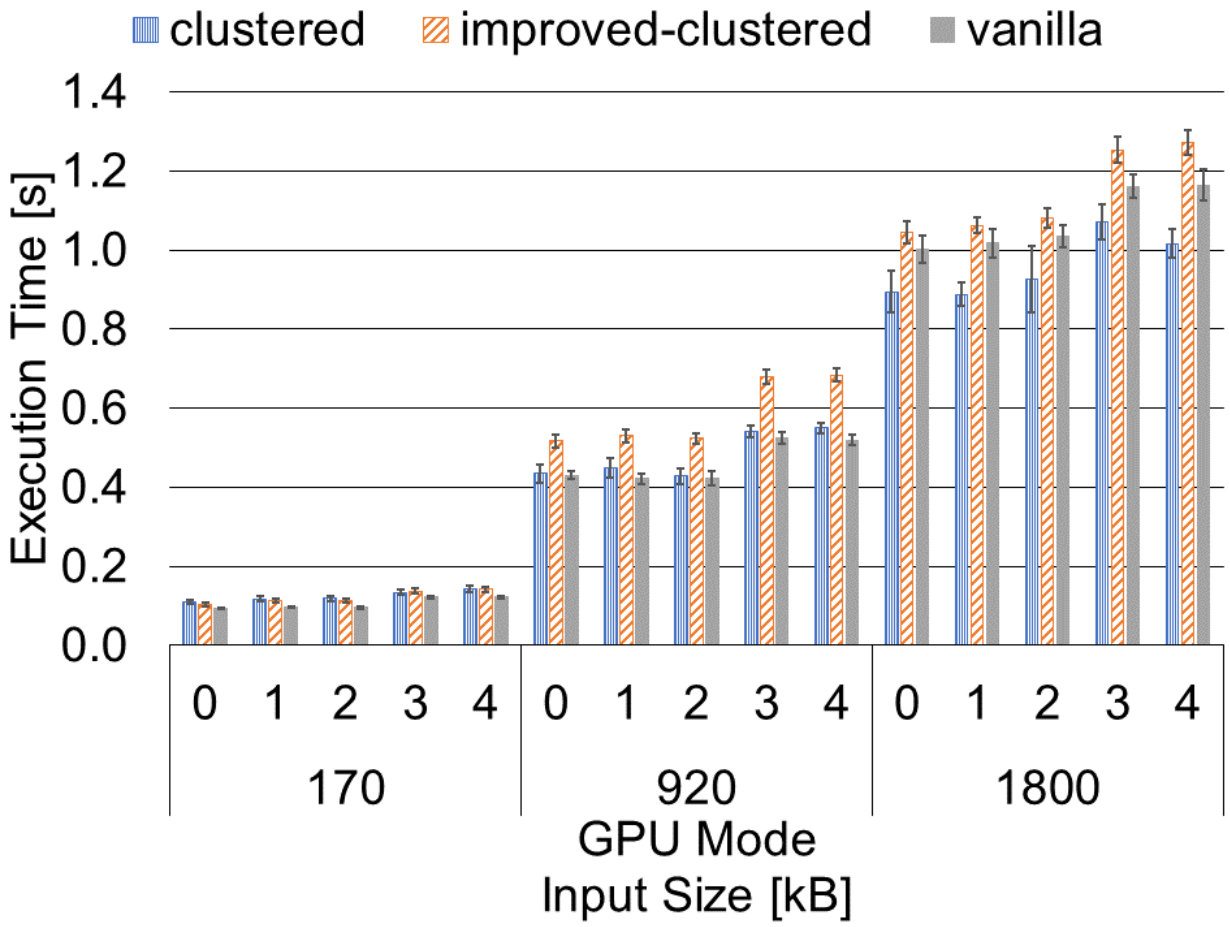

Figure 6: Effect of GPU power modes on execution time for different data sizes.

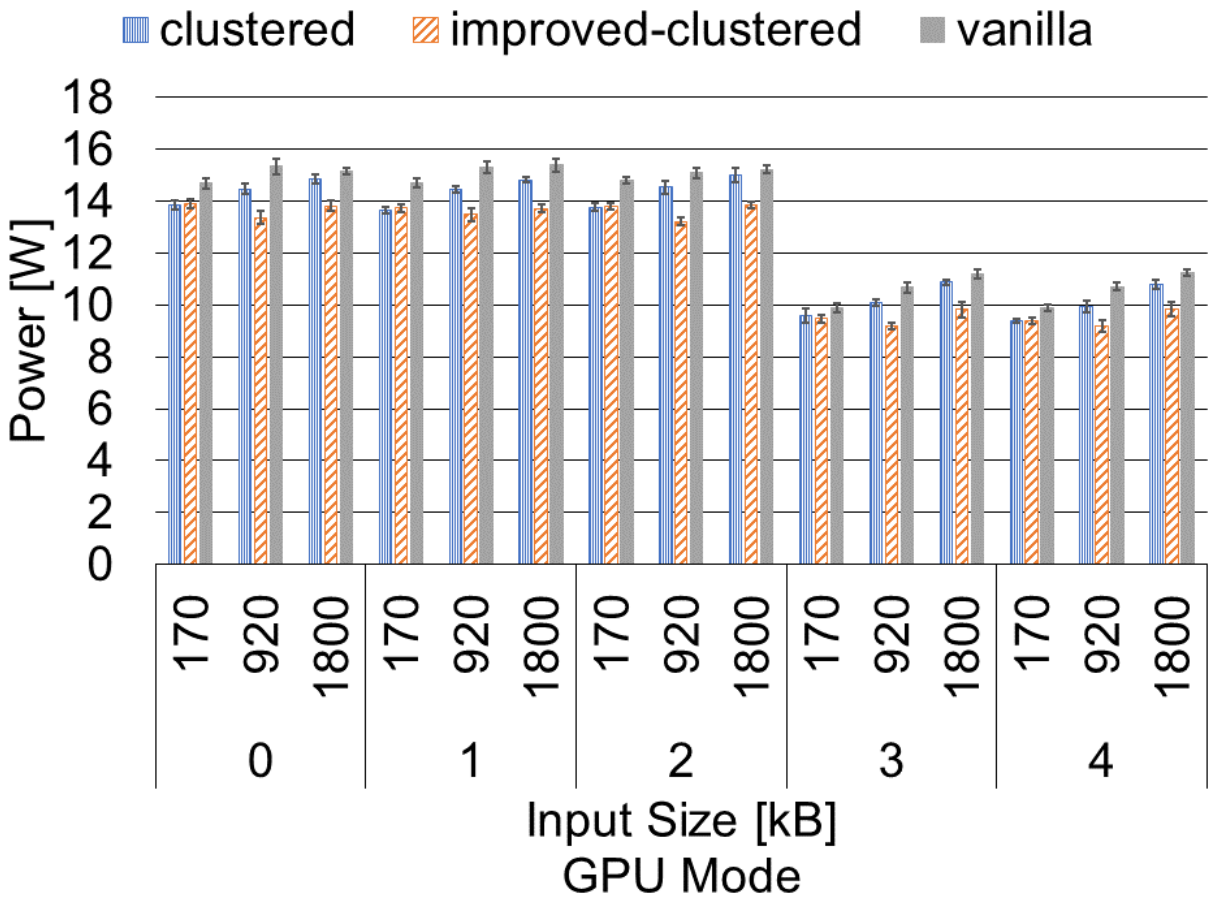

Figure 7: Effect of GPU power modes on power consumption.

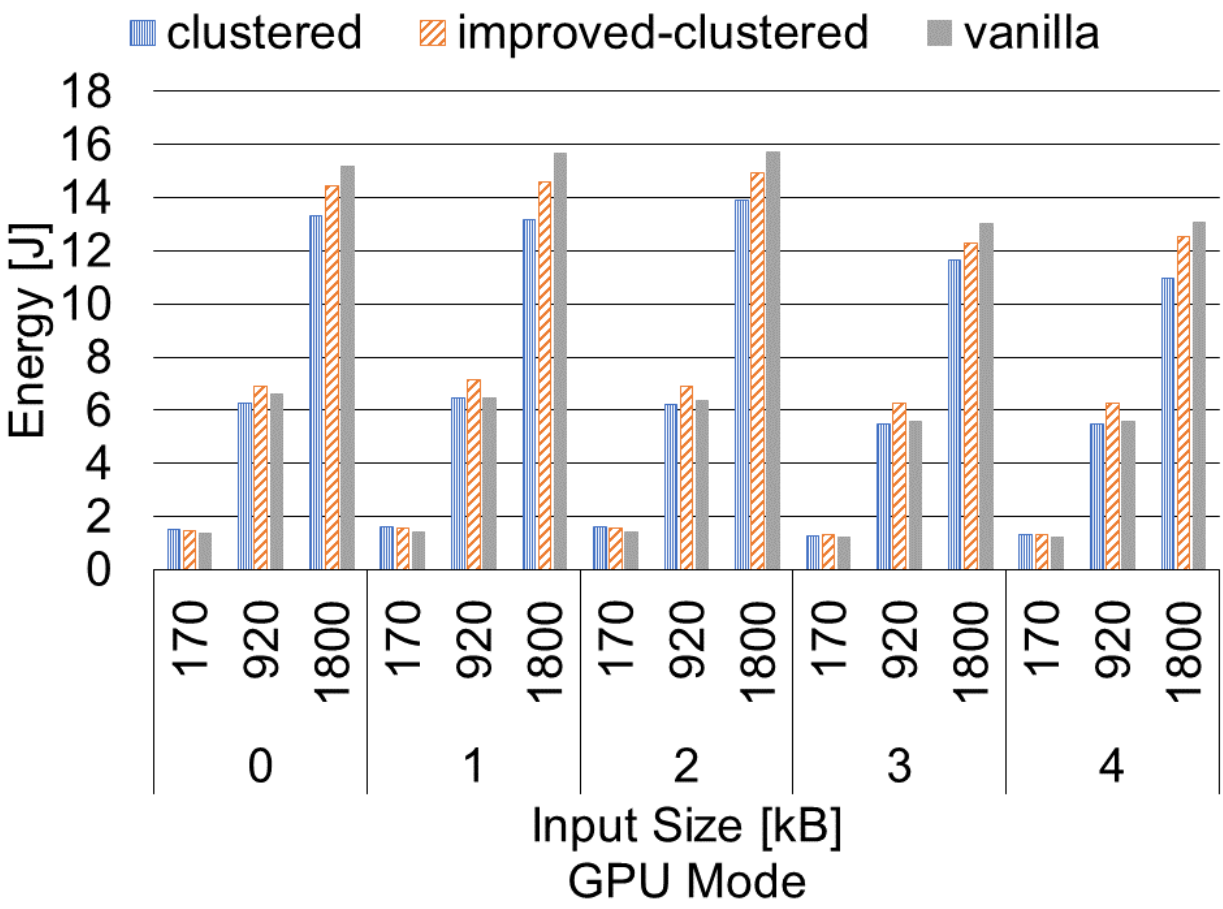

Figure 8: Effect of GPU power modes on energy per inference.

Energy is shown in Fig. 8. These results were calculated using the average execution times for each inference and the average power consumption during inference for three different sizes of data. In terms of total energy per inference, the vanilla softmax model has a higher energy cost than the other two models. The clustered model had the lowest inference energy, specifically in the 10W power modes. Between the 10W modes, the energy is similar for all models except the clustered model.

It is also worth noting that the smaller model require less energy per inference, but they are also processing much less data. This can be overcome by normalizing energy consumed by the size of the data.

The energy used per kB of input data is shown in Fig. 9. This chart only shows power modes 0 and 3, with this scaling, there is little discernible difference between the first three modes or the last two. While the energy efficiency of small and large data sizes is poor, there does appear to be a decrease for medium sized data inputs. This phenomenon will be explored in the discussion Sec. 6.

5.3 Linear Scaling

Due to system complexity, it was prudent to confirm that runtime linearly scales with input size for this model, so the results are reported for the vanilla softmax, clustered, and improved-clustered attention models with varied input sequence lengths. These results are shown in Fig. 10. Note that sequence length is listed in time of input audio sample (assuming a 16kHz sampling rate). Again, the error bars represent a 95% confidence interval. Within expected input ranges, which were determined by the input size bounds in the LibriSpeech dev-clean dataset, the execution time is linear with coefficients of determination exceeding 0.99. For improved-clustering, small input sequences have similar execution times as the other models, but it becomes significantly slower once the input length reaches 10 seconds. This is likely due to the increased overhead for calculating which of the k input keys will be provided to the model.

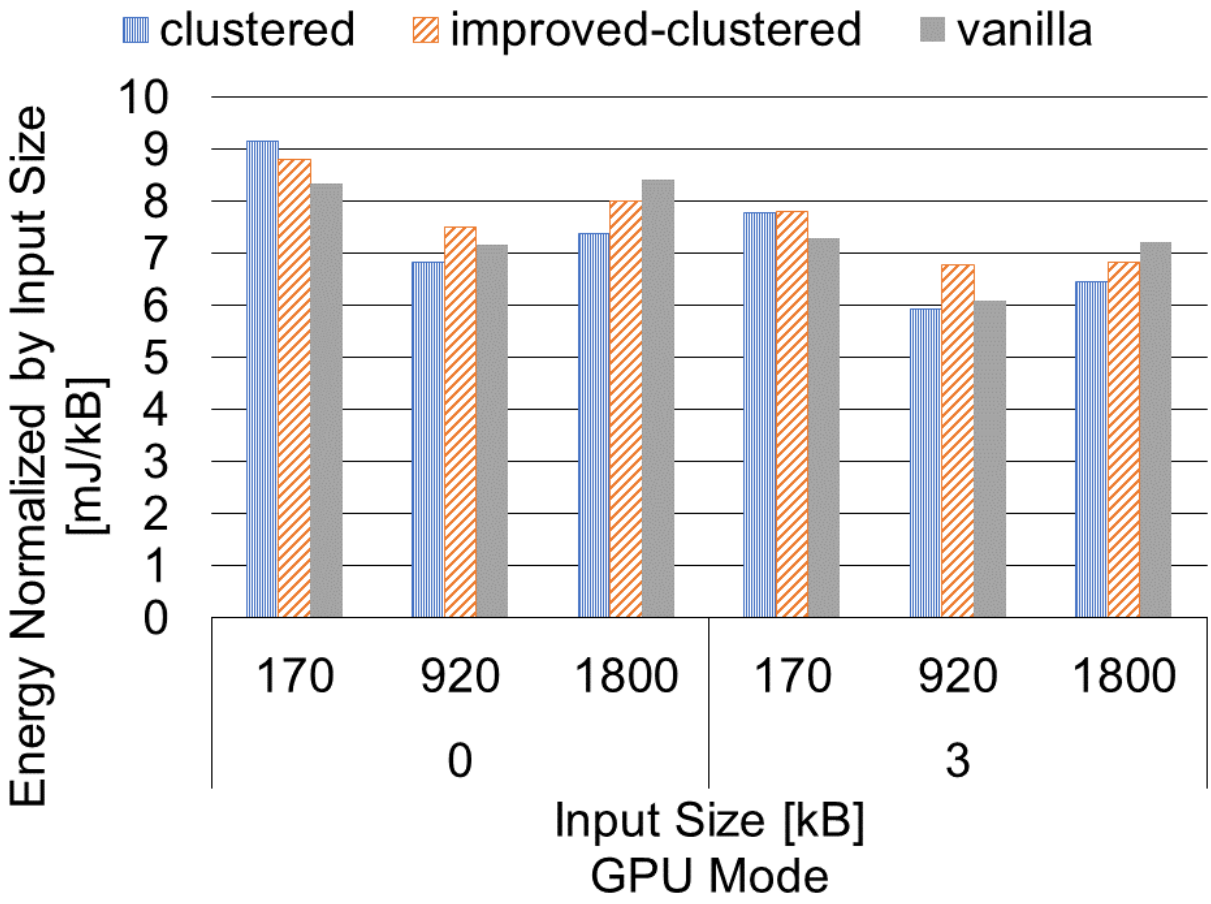

Figure 9: Energy used per kB of input data.

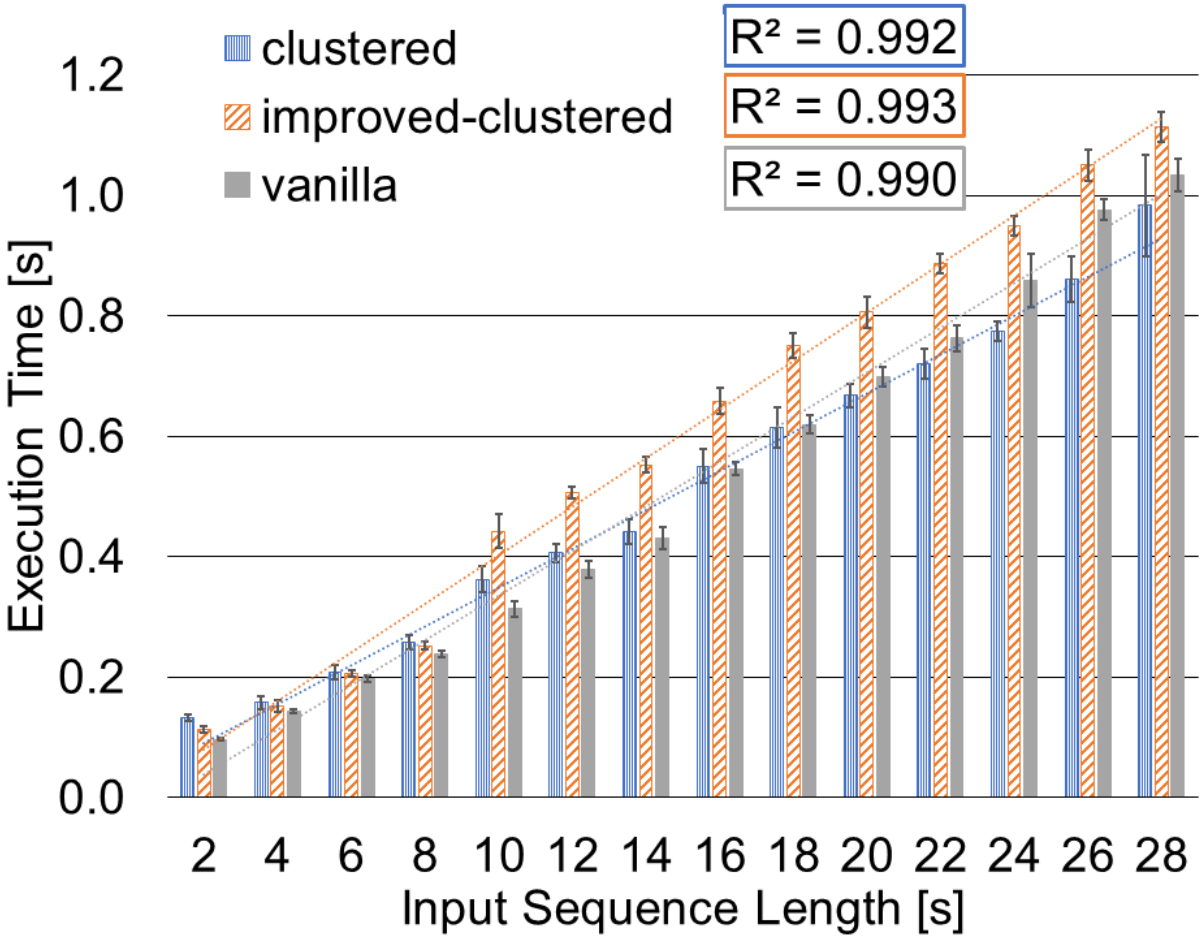

Figure 10: Execution time compared to input sequence length.

5.4 Cluster Number

The effect of number of clusters on relative performance is shown in Fig. 11. Note that more clusters results in smaller clusters. This data was collected in the default GPU power mode (2) with a small, medium, and large sample. For this hardware and model, these optimizations only improve performance for very large audio samples. This makes sense as clustering was intended for models with large input vectors. However, for even modest audio samples, the vanilla softmax model is faster.

It is also worth noting that the improved-clustering model is much slower than the base clustered model due to the additional computation of the relevant keys. This trend is also consistent with the performance metrics provided by Katharopoulos et al. on RoBERTa execution with regards to input size [9].

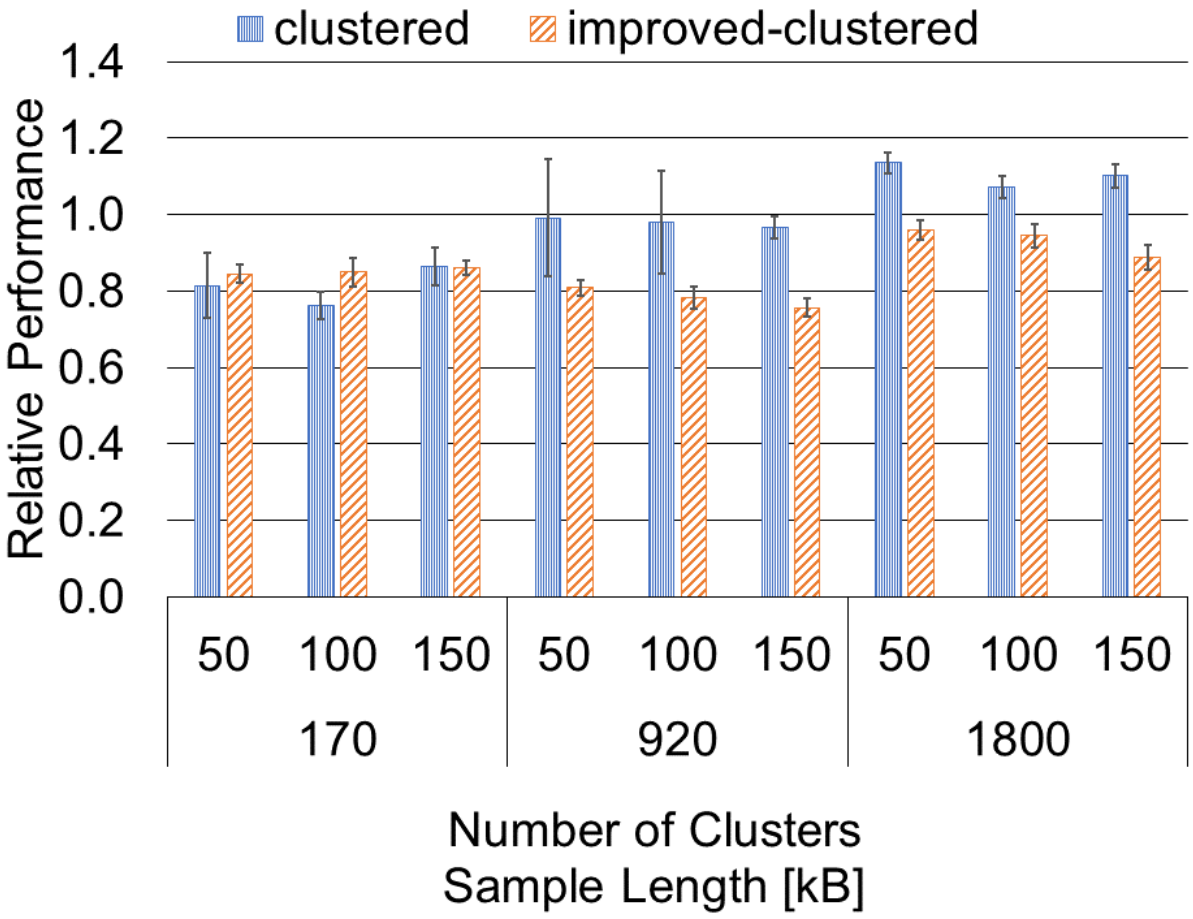

Figure 11: Effect of number of clusters on relative performance (GPU power mode 2).

6.0 Discussion

This section will analyze the data presented in 5. Here, the power and energy of the system will be discussed, as well as the effect of clustering on these metrics. The linearity of this system will also be discussed under various optimizations.

6.1 Execution Time

From a practical perspective, the model executes fast enough to be viable in an onboard processing system. The largest input, roughly 28 seconds of audio, can be processed in approximately 1.2 seconds. This sample is significantly longer than a standard query to a chatbot, which is usually on the order of several seconds. Even input sample lengths up to ten seconds execute in under half a second, which is significantly less than the floor for latency of lunar round-trip communication with a ground server (2.25 seconds) [23]. While it is assumed there is still unknown latency associated with other backend systems such as intent parsing or database searches, this transcription engine is sufficiently quick for other expensive computations to take place and still be advantageous over an earth-based system.

As shown in the results, execution time scaled roughly linearly to the input size. While this isn't strictly a requirement for a space-based platform, this does inform implementation choices. There is no meaningful benefit from the perspective of execution time to restricting the input length of the vector, meaning that the choice to split an input vector would likely be for reasons related to power.

In this experiment, it was found that the vanilla softmax model was the best performing model up to 18 second queries (where both vanilla and clustered models executed in about 0.7s). Realistically, queries to a chatbot are likely a single sentence, and therefore only a few seconds long. Additionally, it is assumed that the system would be on only when needed, similar to current ISS systems that employ push to talk (PTT) or voice operated switches (VOX). Therefore, it is recommended that the vast majority of queries should be run on

a vanilla model. If this system were to be left operating continuously for black-box audio recording, longer audio segments would be produced, and it may be advantageous to use a clustered model.

Overall, for short input audio segments (up to and including 8 seconds), improved-clustered models executed faster than the clustered model. After this point, the improved-clustered model executed slower than the clustered model by a wide margin. This behavior confusing as the improved-clustered method is asymptotically always slower than the clustered model. However, it may be that the improved-clustered model performs better than the clustered model at this level because of how the cache is primed during these optimizations.

6.2 Power and Energy

From the energy perspective, the softmax vanilla model operated at roughly a joule more than improved-clustered model, and two joules more than the clustered model when running the largest model tolerated by memory capacity. This means that for large input sizes, the clustered model is more desirable under energy constraints. However, when a model is run on a small audio sample (170kB), the softmax vanilla model becomes the more desirable model by margins of 0.8-1.1 J for 15W modes and 0.3-0.5 J for 10W modes.

Looking at energy normalized to the amount of data processed, both 170kB and 1800kB input sizes perform worse than the 920kB size. This is likely a result of how memory is accessed within this model. For small input sizes we observe memory-bound behavior, as expected. Particularly for a high-latency device like a GPU, a small input results in under-utilization of hardware compute resources, resulting in reduced overall efficiency. For large input sizes, we see compute-bound behavior since there is more data than hardware can consume concurrently. The size in-between represents a point where the system is neither memory nor compute bound and is thus more efficient overall. Practically, there is limited control over the length input data since it is variable depending on use case. Determining the precise region for this behavior is considered out of scope for this research since input sizes are constrained by system users and not set by the algorithm. However, it is still worth

noting that particularly large input would sequences benefit from being split up. This type of splitting would be relevant in cases where the speech-to-text engine is used to create long transcripts and audio may be cached before processing.

6.3 Effect of Clustering

The clustering optimizations examined by this paper did not provide speedup unless the input sequence was larger than practical for this application. For medium data sizes, clustering is approximately as good as the softmax vanilla model with 50 and 100 clusters, which indicates that it is the point at which the clustering algorithm becomes advantageous over a softmax vanilla model. Improved-clustering, however, did not provide speedup in any of the configurations for this platform. The inability to transfer transformer optimizations to new algorithms and hardware is not an unreasonable find, and is consistent with other preliminary research [13].

Despite the short-comings of clustering as a whole, the base fast-transformers optimization does provide a fast enough implementation to warrant performing this calculation onboard the spacecraft instead of sending it to a ground station for processing. Clustering may be used if there are tight power margins at the cost of additional computation time. Improved-clustering is not recommended.

7.0 Conclusions

In this research, it is demonstrated that a base wav2vec2 model run on an embedded GPU can achieve real-time inference (less than 1 s for inputs $2 - 28$ s in length). This is a critical find for adapting transcription for space applications as it demonstrates the feasibility of adapting an open-source model intended for high-performance hardware to an embedded scale. Additionally, it was shown that the clustered model was a favorable optimization compared to a vanilla and improved-clustered model in terms of energy, but not runtime. The best energy consumption was derived by running the device in its 10W modes (3 and 4) over its 15W power modes (0, 1, and 2) by several joules per inference. It is also advised that these models be run with a small number of clusters (50 performed best in these tests). For most expected inputs, the best runtime is achieved with the vanilla model over the clustered model at the cost of optimal energy efficiency. Overall, it was found that wav2vec2 is amenable to the specific architecture of embedded GPUs in terms of speed and power consumption without optimization.

8.0 Future Work

Due to software constraints, authors were unable to leverage the Tensor Cores or DLAs on the Xavier NX. Future work includes modifying the software models used to accept automatic mixed precision. This would allow for utilization of the Tensor Cores and improve execution time.

Bibliography

[1] Amazon Alexa. Alexa Voice Service v20160207 — Alexa Voice Service, 2016.

[2] Dariom Amodei. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. *International Conference on Machine Learning*, 48:173–182, 2016.

[3] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *arXiv*, 2020.

[4] Fredrik C Bruhn, Nandinbaatar Tsog, Fabian Kunkel, Oskar Flordal, and Ian Troxel. Enabling radiation tolerant heterogeneous GPU-based onboard data processing in space. *CEAS Space Journal*, 12(4):551–564, 2020.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4171–4186, 10 2018.

[6] Google Cloud. Speech-to-Text: Automatic Speech Recognition.

[7] Alex Graves, Santiago Fernández, and Faustino Gomez. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the International Conference on Machine Learning, ICML 2006*, pages 369–376, 2006.

[8] Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo-yiin Chang, Kanishka Rao, and Alexander Gruenstein. Streaming End-to-end Speech Recognition For Mobile Devices. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2019-May:6381–6385, 11 2018.

[9] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. *Proceedings of Machine Learning Research*, 119:5156–5165, 11 2020.

[10] Joshua Y. Kim, Chunfeng Liu, Rafael A. Calvo, Kathryn McCabe, Silas C.R. Taylor, Björn W. Schuller, and Kaihang Wu. A comparison of online automatic speech recognition systems and the nonverbal responses to unintelligible speech, 4 2019.

[11] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *International Conference on Learning Representations*, 2020.

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv*, 2019.

[13] Sharan Narang, Hyung Won, Chung Yi Tay, William Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, Yanqi Zhou, Wei Li, Nan Ding, Jake Marcus, Adam Roberts, and Colin Raffel. Do Transformer Modifications Transfer Across Implementations and Applications? Technical report, ArXiv, 2 2021.

[14] NASA. NASA's Lunar Exploration Program Overview. Technical Report September, National Aeronautics and Space Administration, 2020.

[15] NVIDIA. Hardware Architectural Specification — NVDLA Documentation.

[16] NVIDIA. NVIDIA Jetson Linux Developer Guide : Clock Frequency and Power Management.

[17] NVIDIA. Whitepaper: NVIDIA Telsa V100 GPU Architecture: The World's Most Advanced Data Center GPU. Technical Report August, NVIDIA, 2017.

[18] NVIDIA. PyTorch for Jetson, 2019.

[19] NVIDIA. NVIDIA Jarvis, 2021.

[20] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[21] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *ICASSP, IEEE International*

Conference on Acoustics, Speech and Signal Processing - Proceedings, volume 2015-Augus, pages 5206–5210, 2015.

[22] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 12 2011.

[23] Marc Sanchez Net. *Support of Latency-sensitive Space Exploration Applications in Future Space Communication Systems*. PhD thesis, Massachusetts Institute of Technology, 2017.

[24] George Saon, Gakuto Kurata, Tom Sercu Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall Appen. English Conversational Telephone Speech Recognition by Humans and Machines. *ISCA*, pages 132–136, 2017.

[25] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, volume 2019-September, pages 3465–3469. International Speech Communication Association, 2019.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. *Advances in Neural Information Processing Systems*, 30:5999–6009, 2017.

[27] Apoorv Vyas. apoorv2904/fairseq: Facebook AI Research Sequence-to-Sequence Toolkit written in Python.

[28] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast Transformers with Clustered Attention. *Advances in Neural Information Processing Systems*, 33:21665–21674, 2020.