**Tech Promotion Work Summary**

# Employee Instructions

1. **Partner with your manager to define a next-level goal** (e.g., project, research, presentation, etc.) that allows you to demonstrate one or more next-level capabilities, development areas, and/or Leadership Principles. Next-level criteria can be found in your [Role Guideline](https://ivy-help-center.talent.a2z.com/article/article-1568200618517-Gtmgg1nif) in the “Moving to…” a level section. If your guideline is missing this section, your manager will need to clarify which expectations and technical skills at the next level they want you to demonstrate.
2. **Document *Planned Work* in Part I**. The purpose of documenting a planned goal is to get clear on what is expected and how you and your manager foresee that it gives you the opportunity to demonstrate the next level. If you get a new manager, this can inform them of previous agreements and inform them about what you are working on (relative to the next level).
3. **When you complete the work, fill out Part II*.*** This is a best practice. If you change teams or get a new manager, this gives the new manager a documented record of the work you have done to demonstrate the next level. It is also a record of the impact you had and how your work demonstrates our [Leadership Principles](https://inside.amazon.com/en/About/corevalues_EN/Pages/LeadershipPrinciples.aspx).
4. **Provide at least two work samples that best demonstrate the work** **delivered** (e.g., scope, impact, expertise, technical skill, etc.). Work samples vary by job role. Examples:
   1. Technical work samples - Demonstrate technical and/or scientific expertise. These include scientific research papers, models, algorithm design documents, technical specifications, working mockups/prototypes, code samples, code review commentary (both given and received), architecture/design documents, patent submissions, etc.
   2. Standard work samples - Demonstrate other skills. These include PR/FAQs, narratives, Product Roadmaps, OP1 documents, MBR/QBR materials, functional specifications, UX designs, presentation decks, broadcast videos, project proposals/ROI case studies, wikis, resource planning documents, etc.
5. **After your manager has documented their feedback in Part III, discuss whether it would be helpful to gather additional peer and/or customer feedback**. This is highly recommended, but not required. Your manager gathers the feedback and decides whether some comments are appropriate to add to **Part III**. Work Summary feedback providers can be any level, however higher levels offer better next-level perspectives.

# Manager Instructions

1. **Partner with your employee on a one or more goals that give them the opportunity to demonstrate the next level**. Review the next-level criteria defined in the employee’s [Role Guideline](https://ivy-help-center.talent.a2z.com/article/article-1568200618517-Gtmgg1nif). Design projects or other goals that deliver value for the team’s customer, technology, and business domain. Goals can also give them the opportunity to demonstrate Leadership Principles or resolve a gap identified when hiring or in a previous attempt to promote**.**
2. **If the employee is getting a Planned or Completed Work Review** – Make sure the employee’s summary is clear and well written.
3. **When the work is completed:**
   1. Evaluate the quality of the employee’s work and the results they delivered against the next-level capabilities (and Tech bar) expressed in the employee’s Role Guideline
   2. Document your feedback in **Part III**
   3. Make sure their work samples demonstrate key skills required at the next level
   4. Discuss whether there are peers or customers the employee thinks would be good to get feedback from about the specific effort in the work summary.

# PART I: PLANNED WORK

(Part I required for a [Planned Work Review](https://ivy-help-center.talent.a2z.com/article/article-1588633382719-aPKCqTQob).)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Employee Information** | | | | |
| **Employee Name:** | Kai Zhen | **Current Job Title:** | Applied Scientist II |
| **Manager Name:** | Hieu Duy Nguyen | **Proposed Job Title:** | Applied Scientist III |
| **Steam Member:** | Dave Limp | **Current Business Title:** | Applied Scientist |  |
| **Steam Direct:** | Tom Taylor | **Proposed Business Title:** | Sr. Applied Scientist |  |  |
|  | | | | |
| **Goal (e.g., Project, narrative, presentation, DEMONSTRATE A LEADERSHIP PRINCIPLE)** | | | | |
| *Use your Role Guideline to define a single goal that allows you to demonstrate the type of work expected at the next level, our Leadership Principles, or other area to be considered for a promotion. Summarize in one or two paragraphs.* | | | | |
| The landscape of Automatic Speech Recognition (ASR) has undergone a significant shift towards Generative AI, redefining its computational paradigm. Core ASR neural components now boast over 1 billion parameters (as seen in Let’s Chat) and are projected to reach 7 billion and beyond in the next generation of speech-to-speech architectures. This evolution promises Alexa users a more natural and expressive conversational experience through enhanced speech responses. However, it also entails substantial costs in terms of both training and inference time. Ensuring cost parity, alongside maintaining acceptable user-perceived latency, remains crucial to sustain the business viability of these ASR services (Kingpin Goal xxxxx).  This project aims to develop One-to-Few-Shot Compression for Alexa ASR: a versatile, post-training or fine-tuning staged, neural efficiency tool to employ 50% structured sparsification and sub-4-bit weight quantization to Large ASR models at no WER degradation. The primary objective is to reduce runtime latency and inference cost by accelerating core Alexa ASR components on Nvidia’s Ampere-based GPU instances, which serves as a pivotal factor in attaining “latency- and cost-neutral operating points”. The specific goals encompass (a) adapting the existing compression-aware training methodology to the finetuning stage with the state-of-the-art compression level to reduce training costs; (b) developing and merging highly reusable, production-ready code in Phasa for Conformer, PyRama for Transformer, and future Speech-to-speech multi-modal LLMs; (c) collaborating with both internal and external hardware teams to facilitate the end-to-end accuracy and latency testing regarding sparsification using TensorRT-9 SDK; (d) establishing alignment with the Language Modeling team to showcase or productize sparsification in their BERT-based Rescorer for the en-US locale. Measuring the goals: (a) incurring minimal training cost with a target duration of <2 days of training/finetuning for model sparsification; (b) implementing the sparsity pattern that matches what the hardware actually supports to realize >10% latency reduction in P50 user perceived latency (UPL); (c) minimizing the WER degradation, ideally 2% or less relative degradation from most test sets. To execute these goals effectively: (a) initiating mini model releases with the release owner to showcase on-par WER when sparsity is enabled; (b) hosting bi-weekly sync meetings with key figures from hardware and ASR science teams to address potential blockers and facilitate collaboration; (c) providing regular updates to L6+ stakeholders via email or LR highlights to gather guidance and support for next steps. To keep the goal realistic, I should (a) make the recipe easy to use with minimal intervention from the release owner; (b) make the recipe adaptable to facilitate incremental learning. To exhibit the qualities of the next level, I need to (a) demonstrate strong ownership of designing the sparsification algorithm and leading technology implementation while coordinating cross-team efforts; (b) deliver a robust, high-quality, and end-to-end neural efficiency solution on time with tangible production impact; (c) earn trust and bring influence to multiple teams regarding their current and future production roadmap. | | | | |
|  | | | | |
| **Associated Next-Level Criteria and/or Leadership Principles** | | | | |
| *Cut and paste the next-level criteria or Leadership Principles that this work intends to demonstrate. A single goal can demonstrate readiness in one or more areas. If the work is to demonstrate a different requirement for promotion, document it here.* | | | | |
| **Associated Next Level Criteria:**   * You lead the design, implementation, and successful delivery of large, critical, or difficult scientific solutions involving a significant amount of work. These efforts can be new solutions or a refactor of existing solutions. You heavily influence the design and write a significant portion of the “critical-path” code. * You demonstrate influence over 1-2 teams, either via a collaborative efforts or by increasing their scientific knowledge, usage of advanced or specialized techniques, or engineering best practices. * You influence your team’s science and business strategy by making insightful contributions to team priorities and approach. You take the lead in identifying and solving ambiguous problems, science deficiencies, or areas where your team’s solutions bottlenecks the innovation of other teams. You make solutions simpler. * Your solutions, code, designs, and scientific artifacts (e.g., papers) set a great example to others. You work very efficiently and routinely deliver the right things.   **Leadership Principles:**   * **Deliver Results:** Leaders focus on the key inputs for their business and deliver them with the right quality and in a timely fashion. Despite setbacks, they rise to the occasion and never settle. * **Have Backbone; Disagree and Commit:** Leaders are obligated to respectfully challenge decisions when they disagree, even when doing so is uncomfortable or exhausting. Leaders have conviction and are tenacious. They do not compromise for the sake of social cohesion. Once a decision is determined, they commit wholly. * **Think Big**: Thinking small is a self-fulfilling prophecy. Leaders create and communicate a bold direction that inspires results. They think differently and look around corners for ways to serve customers. | | | | |
|  | | | | |

# PART II: Results

(Complete this section after you finish the work. Parts I, II, and III required for [Completed Work Reviews](https://ivy-help-center.talent.a2z.com/article/article-1588633382719-aPKCqTQob).)

|  |
| --- |
| **Quality of Work/Challenge** |
| *Summarize how your work demonstrated the* ***challenge*** *expected at the next level. Where possible, give examples.* |
| **Ambiguity:**  While the scientific definition of model compression is clear, the project is still considered ambiguous due to multiple factors, especially concerning the release process of the Alexa ASR's actual model. Regarding when to enable model compression, there are multiple options such as during training (gradually enforcing weights to be sparse or with lower precision while optimizing on the training data) and post-training (simply converting weights to the compressed format). While in-training approaches have proven to be robust in terms of accuracy, they are usually resource-consuming. For large ASR modeling, this can raise concerns due to the considerable computation cost, where each model training can take weeks on 8 or more p4 instances. Moreover, the current Large ASR model release procedure involves several stages of training/finetuning, making enabling compression less feasible. Any single change in the upstream training stages necessitates revalidating the compression algorithm throughout the process. In contrast, post-training approaches are favored as they compress the model in one shot, typically at the very end of all those training and fine-tuning stages. However, since they often aren't conditioned on all training data or only a fraction of fine-tuning data for calibration, the performance of post-training compressed models may not consistently hold across all test sets. This inconsistency could potentially become a significant obstacle when releasing the feature.  Concerning how to enable model compression, one can resort to wrapper-based methods (adding quantizer/dequantizer stubs to be invoked when the layer is called) or in-place weight modifiers (modifying weights from all compressible layers before the checkpoint is generated for model evaluation). Wrapper-based methods are well supported in PyTorch, as shown in Low-Rank Adaptation of Large Language Models (LoRA), and can be invoked for every training step. However, adding too many wrappers to each layer leads to recipe conflicts and nullification of certain features. For example, adding a quantizer wrapper may override the LoRA mechanism enabled by the wrapper, if not considered carefully beforehand. Conversely, in-place weight modifier compresses weights within tensors without the addition of wrappers. Therefore, this approach avoids recipe conflicts or clashes between different techniques. However, designing an in-place weight modifier requires careful consideration to ensure it does not impede training speed and can accommodate multiple compression types cohesively.  Akin to model evaluators, the weight modifier can be visualized as a callback during which the model training is temporarily paused. To ensure a manageable training speed, we might limit compression to no more than a few shots during training or fine-tuning stages. In that case, adding extra regularizers to the loss function may not be the most effective approach, and it’s not technically clear how to impose gradual compression to allow the compression-induced loss to be recovered. Regarding the harmonization of multiple compressors, it remains unclear whether quantization should precede sparsification or vice versa. It's also uncertain if the in-range WER observed from a single compressor remains consistent when another is added on top.  **Scope of Influence:**  To successfully productize our One-to-Few-Shot Compression in Large-ASR, it's imperative to establish consensus not only among ASR teams for modeling but also within engineering teams for hardware benchmarking. I prioritize close collaboration with these teams to integrate our innovations into their production models, aiming for alignment and agreement toward model releases.  For example, let's consider the case of Large Rescorer Sparsification [1]. Over two quarters, I led weekly meetings involving Jari Kolehmainen (Sr. AS from the Seattle LM team), Yile Gu (Sr. AS from the Seattle LM team), and others to synchronize our neural efficiency tool's design, select appropriate models for testing, and more. Our combined efforts were recognized in the Runtime Modeling LR as a highlight in both 2023 Q2 [2] and 2023 Q4 [3]. This recognition notably followed the deployment of our 2:4 sparsification to ContextBERT for the Large ASR v2.0 release.  Moreover, to achieve neural efficiency via hardware acceleration for our ASR services, I keep a close collaboration with hardware, engineering, and service teams to gather latency metrics aligned with production settings. I work closely with Buddha N (Sr. SDE in NeMoRT) and Sonal Pareek (Sr. SDE in NeMoRT) to benchmark sparse BERT and Conformer models on TensorRT-9, a widely accessible SDK from Nvidia, validating the latency benefits from sparsification and bf16 [4].  Last but not least, I consistently publish and present our recent findings externally, as showcased in ICASSP [5], and internally, within our neural efficiency workshop [6] and ASR PIT group meetings [7]. These collective efforts significantly enhance the visibility of our research and development endeavors, contributing to a more expansive production impact.  **Scientific and Technical Complexity:**  Alexa ASR’s One-to-Few-Shot Compression mechanism needs to support both N:M sparsification and sub-4-bit quantization in Phasa and PyRama for our current and future Large ASR and Speech-to-Speech components to stay business viable. Comparing with the baseline Large ASR model, our approach aims at an 8X or greater compression (from a 32-bit version to an 8-bit and 2:4 sparse version) at no WER degradation. Innovating and productizing a mechanism as such entails tackling complex problems from both scientific and technical perspectives.  To begin with, we must pinpoint the primary business challenge in implementing model compression for Large ASR models: one of the most evident challenges is controlling costs. Given that reusing our previously established in-training compression approaches is not recommended, we need a One-to-Few-Shot Compression mechanism for finetuning or post-training that maintains comparable WER. Scientifically, it poses a challenge to gradually compress the model within a few finetuning epochs without necessitating the modification to the loss function.  In addition, productizing such neural efficiency innovation in Large ASR model release is technically intricate. The complex in particular stems from the multi-staged model release procedure involving several teams’ endeavor. For instance, by the time the sparsification is scheduled for release, the development of the BERT-based Rescorer featured fundamentally different training and fine-tuning stages compared to earlier phases. To address the accuracy gap for contact names, a separate Context Encoder (CE) was introduced into the pre-trained RescoreBERT, transforming it into ContextBERT. The training of CE depends on the availability of contextual information from the first-pass neural biasing. Consequently, apart from enabling sparsification for RescoreBERT during fine-tuning, we also need to apply sparsification to ContextBERT, which is fine-tuned using the newly trained slot embeddings, to ensure 0.5% or less WER degradation on average among over 40 test sets.  Moreover, when designing our One-to-Few-Shot Compression mechanism, I need to make the mechanism versatile, as the hardware evolves, such that it can extend its support for other compressors easily, such as block sparsity, 8-bit quantization, 4-bit quantization and sub-4-bit quantization, etc. In that regard, it is essential to separate the compression scheduler from the specific implementation of the compressor. Achieving this requires designing and implementing: 1) the compression scheduler, 2) the specific compression utilities, and 3) the compression configuration alongside all other hyperparameters in a consolidated yet well-maintained approach. Such a solution hasn't been present in PyRama yet, and pioneering such an effort within a short timeline poses a significant challenge.  **Impact:**  In this project, we developed One-to-Few-Shot Compression, a versatile platform-agnostic (for both PyRama and Phasa) neural efficiency paradigm. When instantiating it as sparsity-aware finetuning (SAF), we can compress Alexa's acoustic and language models by up to 50% without hindering WER on external test sets (such as AWS\_Transcribe\_eval\_set), general internal test sets (such as enus\_letschat\_alpha), and particular personalization test sets (such as contact\_names\_250w). Our compression mechanism does not impede training time and can lead to runtime cost and latency reduction which is crucial to achieve cost parity and improve accuracy-latency tradeoffs. Deployed successfully in Lets-Chat v2.0, SAF reduced latency by 17.8% on p50 data setting and ~40.0% on p99 in the BERT-based Rescorer on TensorRT-9 while maintaining the WER across over 40 Large ASR test sets [3]. SAF's application extended to compressing the 1B param Conformer, showcasing <0.5% relative degradation on both Alexa\_prize\_onlygoldens and enus\_letschat\_alpha with up to 10% latency reduction even with batch size of 1 [8]. Merged in Phasa/PyRama Core through rigorous validation and successful production intake, SAF solidified its role as a versatile and efficient tool within Alexa's neural efficiency methods.  Note that our compression-aware finetuning paradigm is not only applicable to sparsification but also 8-bit, sub-8-bit and even sub-4-bit quantization with minimal effort. To further reduce the model size, approximating the bit-depth of 4, we prototyped 8-bit (compatible with SmoothQuant.) BERT along with 2:4 sparsification (50% sparsity) and observed consistent accuracies from all the en-US Large Model Evaluation Config in Dory-BlueShift-Speech. The combination of 8-bit (8-bit weights only or SmoothQuant with both weights and activations in 8-bit) quantization and 2:4 sparsification led to 61.9% throughput improvement and 38.3% latency reduction.  Our One-to-Few-Shot compression achieves state-of-the-art performance in LLM quantization, demonstrating that a 3-bit 1B ContextBERT maintains equivalent WER across more than 40 test sets [9]. Consequently, this advancement holds the potential to extend its benefits to even larger Speech-to-Speech multi-modal LLMs, potentially enabling the hosting of 7B-30B parameter Llama models on a single GPU, thereby offering significant cost savings.  **Execution:**  The evolution of Large ASR modeling is advancing rapidly. Bridging the gap between innovative prototyping and actual product integration presents significant challenges. To effectively execute our research and development plan, I need to oversee algorithmic design, implementation, training, and testing. Additionally, it is crucial to dedicate substantial effort to coordinate with SDEs from hardware acceleration teams, applied scientists from the language modeling team, and TPMs and leaders from the LRs. Both perspectives are essential for aligning to identify potential blockers and synchronize release schedules.  To foster early consensus on the viability of compressing large ASR models by 50% without compromising WER, I initiated the endeavor on sparsity-aware fine-tuning as early as Q2-2023. For frugality, I began experimenting with only one P3 instance, although the actual model was built across multiple P4 instances. This approach saved tens of thousands for Alexa. It significantly accelerated the development cycle, enabling us to identify algorithmic flaws and make corresponding improvements swiftly. Consequently, we could regularly synchronize with Rescorer model owners, providing them with promising, up-to-date experimental results. This execution strategy effectively garnered trust from collaborating teams. Our initial progress on Rescorer sparsification was highlighted in the Runtime Modeling LR in Q2-2023 [2].  Furthermore, I've been conducting weekly syncs to facilitate cross-team coordination regarding model artifacts and data readiness [10]. This coordination is critical to adapt our production-ready experimental plan for the model release. I collaborated closely with Jari Kolehmainen (Sr. AS from Seattle LM team) and Yile Gu (Sr. AS from Seattle LM team) to finalize the candidate experimentation plan. Thanks to our consistent coordination and robust algorithmic design, we successfully adapted our SAF mechanism to ContextBERT, which can be pruned in just a few shots within 4-5 hours of finetuning with no WER degradation [1].  Lastly, I authored succinct documentation for the Hardware-Acceleration LR, including the one summarizing the current state of sparsification for the Let’s Chat model release [11]. I also remained engaged with Rolando Jimenez (Principal TPM), Denis Filimonov (Principal AS), and other leaders, both offline and during meetings, actively seeking their advice based on the latest results. These collaborative efforts contributed significantly to the successful rollout of sparsification in the Let’s Chat v2.0 release.  **Knowledge:**  With over 5 years of experience in model and speech/audio embedding compression, I have specialized in designing end-to-end waveform coding solutions. These solutions effectively reduce the sample rate and bit depth of acoustic feature maps, achieving a compression ratio of 10x or more in a lossless manner. In the realm of sparsification, in particular, I’ve worked on policy-based unstructured pruning for RNN-T and channel-wise sparsification for Conformer, prior to accelerating Large ASR models. My extensive production experience encompasses hands-on involvement in Alexa ASR modeling pipelines. This includes specific tasks like data preparation, model design, implementation, training, packaging, evaluation, and various testing methodologies. My relevant knowledge base enables me to innovate in proposing novel model compression mechanisms and streamline the product intake procedure.  **General qualifications:**  I began my journey with AHS-ASR in May 2020 as an applied scientist intern, already having published a paper on ASR model sparsification during this internship. Subsequently, I transitioned to a full-time role with the AHS ASR group at Amazon PIT, where I led several impactful neural efficiency projects for both on-device and cloud ASR, significantly influencing production and research initiatives. I earned a Ph.D. in computer science and cognitive science from Indiana University Bloomington, where I spearheaded multiple speech and audio neural waveform coding projects, pioneering a new research area. The results of my work were published in prestigious signal processing and speech processing conferences and journals, including Interspeech, ICASSP, IEEE Signal Processing Letters (SPL), and IEEE T-ASLP. In recognition of my research contributions, the Cognitive Science Program at IU honored me with the Outstanding Research Award in 2021. Additionally, I hold six US patents as an inventor, showcasing my contributions to innovative technological solutions. Before joining Amazon, I contributed to BERT-based multi-modal recommendation systems as a machine learning relevance intern at LinkedIn from 2018 to 2019. My educational background includes an M.S. in Computer Science from Tsinghua University in 2015 and a B.S. in Software Engineering from Xidian University in 2012. During my time at Xidian University, I received the Honorable Mention award for the International Mathematical Contest in Modeling (MCM) and was a three-time recipient of the National Scholarship. |

|  |
| --- |
| **Impact/Measure of Success** |
| *Summarize in one or two paragraphs the impact this effort had on customers, your team/organization/business area, partners, and others. What data (quantitative1 metrics or qualitative measures) can you provide?* |
| In this project, we proposed and implemented One-to-Few-Shot Compression supporting both arbitrary bit-depth post-training quantization (PTQ) and 2:4 sparsity-aware finetuning (SAF). Instantiated as the Alexa-specific policy-based sparsity-aware finetuning (SAF) paradigm, we can compress the Alexa acoustic model (1B+ param Conformer) and language model (1B+ param BERT) by up to 50% without compromising the WER via Pryon decoding.  In 2023 Q4, we've successfully productized SAF by incorporating N:M sparsity in BERT-based 1B param Rescorer for the Lets-Chat v2.0 release [3]. Among the 20 layers, we pruned 50% of the weights to zero across 14 layers using N:M SAF. This resulted in a 17.8% relative reduction in latency on TensorRT-9, utilizing <7x16> inputs as the p50 production setting [4]. In terms of accuracy, the sparse Rescorer maintained consistent WER (<1.5% relative degradation) from over 40 test sets in end-to-end benchmarking. As a results, the sparse ContextBERT retained a Word Error Rate Reduction (WERR) of >20% on contact-name test set, achieved from Gazetteer modeling and first-pass Neural Biasing, while simultaneously achieving significant reduction for the memory footprint and latency [1][3]. In addition to BERT-based language model, SAF's application in compressing the 1B param Conformer, the acoustic model, was showcased: with over 90% kernels in the reduced-frame-rate encoder of the Conformer pruned by 50%, the Let’s Chat GPU package only exhibits <0.5% relative degradation on both Alexa\_prize\_onlygoldens and enus\_letschat\_alpha [8], yet with up to 10% latency reduction measured via NeMoRT-Bench [4][8].  Additionally, to further compress Large ASR models, we proved that 8-bit quantization and 2:4 sparsification can be jointly conducted on ContextBERT without degrading WER, yet approximating 4-bit weight quantization level [9]. I coordinated with Nvidia and requested the hardware benchmark: Compared to non-sparse BF16 baseline, the sparse 8-bit Transformer yields 61.9% throughput improvement and 38.3% latency reduction, both measured on the p50 data with the batch size of 7 and sequence length of 16. With NeMoRT’s custom acceleration kernel as an ONNX plug-in, 2:4 sparsity yields ~40% latency reduction. Along with 8-bit quantization, one can host Llama up to 30B parameters on a single GPU.  Furthermore, our method, operating solely as a post-training quantizer, showcased on-par WER even when the ContextBERT is in 3-bit, which offers more than 8X compression from the 32-bit baseline model [9]. Achieving these outcomes holds significant value in attaining cost parity and enhancing accuracy-latency tradeoffs. Our approach also significantly reduces training costs without necessitating training-from-scratch. It efficiently delivers the expected quantized/sparsified ASR components from their pre-trained versions in just one to a few shots or less than 1 day's effort. This design of One-to-Few-Shot Compression aligns perfectly with the development process of Let’s Chat Alexa ASR, minimizing production intake risks. The implementation of our method underwent rigorous scrutiny by SDEs for both PyRama and Phasa before merging into their respective Core folders in the mainline. Our neural efficiency tool features adaptability to various Alexa ASR model components, both presently and in the future, for reducing latency and inference cost. |
|  |

|  |
| --- |
| **Leadership Skill & InFluence (Section required for promotions into L6+ only)** |
| Please describe how this work gave you the opportunity to demonstrate any other leadership skills (e.g., drive a best practice, influence a needed change, build consensus on an approach)? How did you apply judgment (e.g., design choices, trade-offs, priorities) or make decisions with long-term effects? |
| In this project, we pioneered One-to-Few-Shot Compression for Alexa ASR and productized Sparsity-Aware Finetuning (SAF) to compress Alexa's models by up to 50% while maintaining WER and boosting runtime efficiency, crucial for cost parity. Implemented in Lets-Chat v2.0, it cut latency by 17.8% (p50) and ~40.0% (p99) in BERT-based Rescorer on TensorRT-9, preserving accuracy. It compressed the 1B param Conformer with <0.5% degradation on test sets and up to 10% latency reduction even at batch size 1. Combined with INT8 quantization, SAF achieved 61.9% throughput improvement and 38.3% latency reduction, enabling hosting of large models like Llama on a single GPU. Merged into Phasa/PyRama Core after validation, our One-to-Few-Shot Compression solidified its role in optimizing neural efficiency for present and future Alexa's systems.  **Deliver Results:**  While sparsification stands as a well-established model compression technique in the literature, delivering it on the right hardware to bring actual benefits to our customers’ experience remains a complex endeavor. To begin with, implementing Conformer and BERT in two separate code bases, namely Phasa and PyRama respectively, necessitates the development of a sparsification-aware fine-tuning (SAF) mechanism for both platforms, demanding a substantial amount of effort. Additionally, both Conformer and BERT are with multiple stages of training in production setting: Enabling SAF on the right stage with right version necessitates tremendous efforts on coordination.  To deliver results, I consistently engaged with the release owners for both Conformer and BERT to build consensus early on that SAF does not impact accuracy. Take BERT as an example, we first proved that SAF yielded little-to-no accuracy degradation on 170M BERT back in Q2/2023. Because of that, the Seattle-LM team (Yile Gu, Senior Applied Scientist and Jari Kolehmainen, Senior Applied Scientist) was inclined to support us on applying SAF to 1B BERT in Q3/2023, and then ContextBERT with Gazetteer finetuning in Q4/2024 with end-to-end Dory-BlueShift decoding on over 40 test sets for accuracy verification. Moreover, I proactively collaborated with Buddha N (Senior SDE from NeMoRT) and Sonal Pareek (Senior SDE from NeMoRT) on hardware testing for latency metrics for several rounds, meticulously documenting quantitative evidence of latency reduction and throughput improvement through sparsification.  From initiating early consensus-building to conducting rounds of model-level testing and culminating in comprehensive end-to-end testing, we successfully delivered results: Alexa’s first sparse Transformer that maintains in-range accuracy across over 40 test sets and achieves approximately 20% latency reduction. Furthermore, we established a robust and reusable SAF paradigm that naturally applies to most ASR/NU neural components in the PyRama mainline.  **Have Backbone; Disagree and Commit:**  We encountered significant challenges during the rollout of sparsification for the large ASR model release. One major issue stemmed from the instability of the TensorRT-9 software development kit (SDK), causing a delay in the general access (GA) SDK's release by over a quarter, from August 28th to December 7th. Consequently, our sparse model, initially scheduled for release in November, wouldn't yield any latency benefits until after the SDK's deployment. This led to setbacks in obtaining p4 instances and necessitated a potential delay to our release plan. As the technology owner, I authored documentation detailing the current state of sparsification [xxx], emphasizing its technological readiness. All necessary codes have been merged into the mainline, and enabling sparsity for BERT requires minimal effort, just half a day of fine-tuning. Additionally, I highlighted previously demonstrated evidence regarding accuracy and latency improvements. I communicated extensively via email and discussions with various stakeholders involved in hardware acceleration and large ASR modeling within the LR, including Denis Filimonov (Principal Applied Scientist at Alexa-ASR), Ariya Rastrow (Senior Principal Applied Scientist at Alexa-ASR), Ehry MacRostie (Senior Manager at Alexa Speech), and Rolando Jimenez (Principal TPM at Alexa Speech), among others.  Despite the challenges, I remained steadfast and committed to the project. With the support of these stakeholders, we collectively decided to proceed with the original release plan. Eventually, we successfully delivered sparse BERT in v2.0 on time.  **Think Big:**  I make bold moves w. caution.  Think corners. Never be conveniently naïve. Work with hot-fixing, stuff. |

|  |  |  |
| --- | --- | --- |
| **WORK SAMPLES** | | |
|  | **Work Samples Name and Link** |
|  | [Large Rescorer Sparsification](https://wiki.labcollab.net/confluence/display/Doppler/Large+Rescorer+Sparsification) | Wiki for Large Rescorer Sparsification | |
|  | [Runtime Modeling LR Highlights](https://quip-amazon.com/vf3FA6wsCsi9/Runtime-Modeling-Bi-Weekly-Update-05022023#temp:C:CDX8c5b2e1e4cfa44b5bde4a7e26) | Highlights from WS-3: Large Rescorer on matching accuracies from 2:4 sparsified RescoreBERT | |
|  | [Runtime Modeling LR Highlights](https://quip-amazon.com/kpGlAM6UVxgy/Runtime-and-Multi-Task-Bi-Weekly-Update-11142023) | Highlights on the deployment of Sparsity Aware Finetuning to ContextBERT in Let’s Chat v2.0 model release | |
|  | [Sparse Model Benchmarking on TensorRT-9](https://quip-amazon.com/FiygAkLbAM6u/TRT-9-testing-on-sparsity) | Latency metrics from sparse Large ASR models on TensorRT-9 | |
|  | [Sparsity via Compressed Sensing for ASR](https://assets.amazon.science/d1/3e/1301067541d0ac20abbcfe0d93d5/sparsification-via-compressed-sensing-for-automatic-speech-recognition.pdf) | IEEE ICASSP publication on ASR sparsification | |
|  | [Presentation for Neural Efficiency Workshop](https://broadcast.amazon.com/videos/879139) | Our presentation on quantization, sparsification and overview | |
|  | [Presentation for AHS Group Meetings](https://broadcast.amazon.com/videos/787930) | Our presentation on recent progress for 2:4 sparse RescoreBERT | |
|  | [Large Conformer Inference Optimization](https://wiki.labcollab.net/confluence/display/Doppler/Large+Conformer+Inference+Optimization) | Wiki for Large Conformer Sparsification | |
|  | [Sparse and Sub-4-Bit Quantization](https://wiki.labcollab.net/confluence/pages/viewpage.action?pageId=2435575677) | Wiki for Sub-4-Bit Level Quantization for Large ASR | |
|  | [Meeting Notes with LM team and NeMoRT](https://quip-amazon.com/M0aaAYJzbGQk/Sparsebf16-ContextRescoreBERT-for-1114-Launch) | Meeting notes with LM team and NeMoRT for bf16+sparse ContextBERT | |
|  | [Sparsification: Status-Quo](https://quip-amazon.com/epZaApKIelZo/Sparsification-the-status-quo-and-questions-you-may-have) | The summary document on sparsification for HWA LR regarding Let’s Chat model release | |

**--SUMMARY SHOULD NOT EXCEED 3 PAGES--**

# PART III: Evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| **Manager FeedbACK** | | | |
| *Date:* | | | |
|  | | | |
|  | | | |
| **Additional FeedbACK** | | | |
| **feedback provider Name:** |  | **Feedback provider business title:** |  |
|  |  |  |  |
| **Feedback date:** |  | **Feedback provider level:** |  |
| **Feedback type:** | Select | **Relationship to candidate:** | *(e.g., customer, peer, stakeholder, teammate)* |
|  | | | |
| **Additional FeedbACK** | | | |
| **feedback provider Name:** |  | **Feedback provider business title:** |  |
| **Feedback date:** |  | **Feedback provider level:** |  |
| **Feedback type:** | Select | **Relationship to candidate:** | *(e.g., customer, peer, stakeholder, teammate)* |
|  | | | |
|  | | | |