







Compress and Compare: Interactively Evaluating Efficiency and Behavior Across ML Model Compression Experiments

Angie Boggust*[†] , Venkatesh Sivaraman*[†] , Yannick Assogba , Donghao Ren ,
Dominik Moritz , and Fred Hohman 

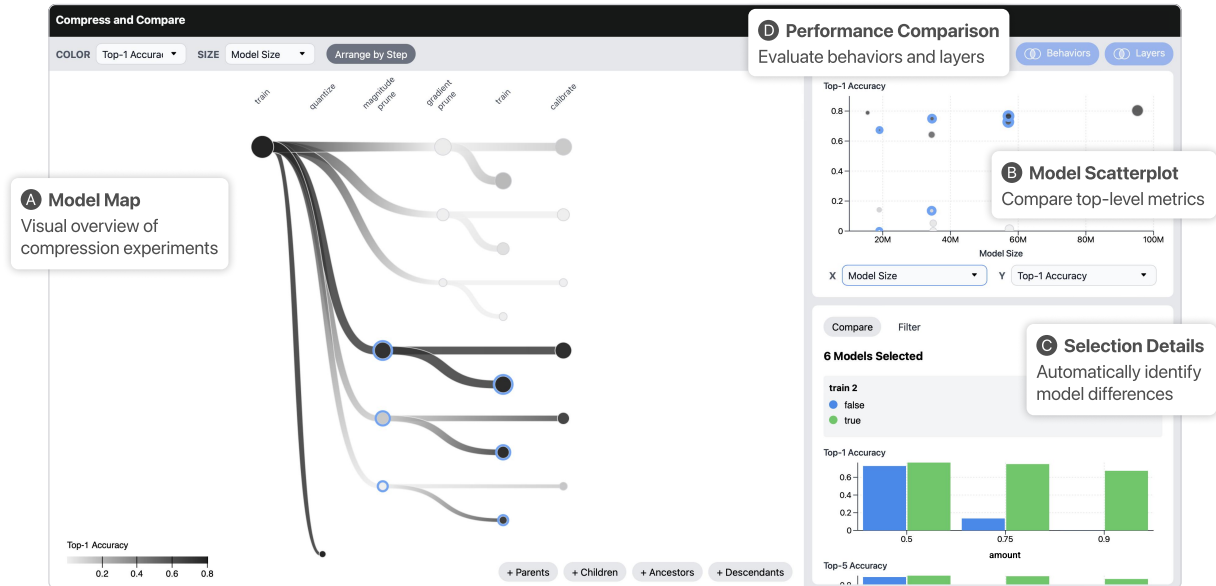


Fig. 1: COMPRESS AND COMPARE helps ML practitioners analyze and compare compression experiments. The (A) **Model Map** displays an overview of the compressed models and the experimental operations used to create them. Users can compare models’ top-level metrics in the (B) **Model Scatterplot** and their operational differences in the (C) **Selection Details** view. To compare models’ behaviors and internal layer characteristics, users can visit the (D) **Performance Comparison** views shown in Fig. 5.

Abstract—To deploy machine learning models on-device, practitioners use compression algorithms to shrink and speed up models while maintaining their high-quality output. A critical aspect of compression in practice is model comparison, including tracking many compression experiments, identifying subtle changes in model behavior, and negotiating complex accuracy-efficiency trade-offs. However, existing compression tools poorly support comparison, leading to tedious and, sometimes, incomplete analyses spread across disjoint tools. To support real-world comparative workflows, we develop an interactive visual system called COMPRESS AND COMPARE. Within a single interface, COMPRESS AND COMPARE surfaces promising compression strategies by visualizing provenance relationships between compressed models and reveals compression-induced behavior changes by comparing models’ predictions, weights, and activations. We demonstrate how COMPRESS AND COMPARE supports common compression analysis tasks through two case studies, debugging failed compression on generative language models and identifying compression artifacts in image classification models. We further evaluate COMPRESS AND COMPARE in a user study with eight compression experts, illustrating its potential to provide structure to compression workflows, help practitioners build intuition about compression, and encourage thorough analysis of compression’s effect on model behavior. Through these evaluations, we identify compression-specific challenges that future visual analytics tools should consider and COMPRESS AND COMPARE visualizations that may generalize to broader model comparison tasks.

Index Terms—Efficient machine learning, model compression, visual analytics, model comparison

1 INTRODUCTION

- * Authors contributed equally.
- † Work done at Apple.
- Angie Boggust is with the Massachusetts Institute of Technology. E-mail: aboggust@csail.mit.edu
- Venkatesh Sivaraman is with Carnegie Mellon University. E-mail: venkats@cmu.edu
- Yannick Assogba, Donghao Ren, Dominik Moritz, and Fred Hohman are with Apple. E-mail: {yassogba,donghao,domoritz,fredhohman}@apple.com

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxx

Machine learning (ML) models have dramatically increased in scale over the past several years, with published models rising from 1 billion parameters in 2018 to over 100 billion parameters as of 2024 [17, 59]. This trend has produced models with exciting emergent capabilities that have enabled new user experiences, like real-time translation [44] and code generation [13]. However, this scale also incurs greater technical, financial, and environmental costs to integrate these models into everyday use [4]. As a result, *model compression* has emerged as an essential family of techniques to make large models viable for practical usecases, particularly in domains where models must run on end-user devices to lower latency or access private user data [24].

ML practitioners apply compression with the intent to maintain the accuracy of a large model while reducing the space required to store it and the time required to perform inference. However, which compression technique or combination of techniques will achieve this balance remains task- and model-specific [24]. The ML literature has proposed various compression techniques for different model architectures and user priorities, such as low space consumption, low latency, or fast execution on optimized hardware [11, 12, 16, 66]. Nevertheless, identifying the right compression strategy can require anywhere from a few to several dozen experiments [24], taking time that is often not accounted for in accuracy-focused model development timelines. It can also be challenging to communicate experimental results within and across teams, particularly those with varying ML expertise. Even when these efforts are successful, compression can alter model behavior in subtle and unexpected ways, creating new errors or biased outputs [27] that are hard to capture with a single metric.

Although compression is increasingly used in research and industry domains, there has been little work using visualization to make compression techniques more interpretable and comprehensible. Initial work on compression visualization has focused on specific compression techniques, like neural network pruning [32, 53] or profiling a single model’s power and performance characteristics on specific hardware [25]. While these methods begin to demonstrate the value of visual tools for compression tasks, ML practitioners often need to take a broader approach to experimentation. As they mix and match multiple techniques in different orderings, the number of experiments and models they produce quickly expands, creating visualization challenges that are not well-supported by available tools for either interactive compression or model comparison [19, 40, 41].

In this work, we explore how to address model compression challenges using interactive visualization. We first identify four model compression challenges by synthesizing prior qualitative findings on how ML practitioners use compression [24] with insights from the ML literature. In response to these challenges, we introduce COMPRESS AND COMPARE, an interactive visualization system for comparing the performance and behavior of a suite of compressed models. Through an overview visualization called the **Model Map**, our system helps users track their compression experiments and how they relate to one another. Users can select subsets of models to automatically visualize differences in their accuracy, efficiency, and provenance. To deeply inspect a smaller set of models, the system provides detailed comparisons of instance-level behaviors and internal activations. Through case studies and user studies, we demonstrate how COMPRESS AND COMPARE can lead to insights throughout model compression and how it expands the design space of interactive ML development tools to account for challenges made salient by compression. We contribute:

- **Four identified compression challenges** ML practitioners face when developing and selecting model compression strategies.
- **COMPRESS AND COMPARE**, an interactive visualization system enabling comparative analysis over many compressed models.
- **Case studies on two common compression tasks** demonstrating how COMPRESS AND COMPARE can help debug failed compression experiments and identify compression-induced bias.
- **A user study with eight compression practitioners** illustrating how COMPRESS AND COMPARE help users build intuition by providing structure to their compression workflows.

2 BACKGROUND AND RELATED WORK

2.1 Techniques for Model Compression

Model compression encompasses various techniques that reduce the storage space, memory, power, or time required to run an ML model while preserving its original behavior as much as possible [7, 8, 10, 38, 57]. Compression is increasingly essential for running ML models, both in resource-constrained settings such as mobile devices and for extremely large models (e.g., generative language models).

Most compression techniques fall into one of three classes: (1) *quantization and palettization* reduce the space required to store each

individual parameter, (2) *pruning* removes parameters while optionally adjusting the others to compensate, and (3) *factorization and distillation* find a different set of parameters that mimic the behavior of the original model [8]. The most straightforward instantiations of these techniques are quantization (converting high-precision formats like 32-bit floats to lower-precision formats like 8-bit integers) and unstructured magnitude pruning (zero-ing out weights with the smallest absolute values). These foundational techniques form the starting point for many real-world compression strategies because they perform well in practice, are easy to understand, and are straightforward to compute [24, 37]. Additional routines often employed to tune the resulting compressed model include *fine-tuning* (training the model on a data subset) and *calibration* (adjusting model parameters to compensate for compression).

In cases where off-the-shelf techniques are insufficient, task-specific techniques have been developed to achieve better efficiency trade-offs [14, 15, 22, 31, 65, 70]. These methods vary by which aspects of model efficiency they target, how computationally expensive applying the compression is, and whether or not they depend on additional training or calibration data. For example, some methods utilize random data samples to decide which parameters can most easily be pruned or restore intermediate activations [2, 12, 39], while others are data-agnostic [20, 49] and can optionally be followed by a retraining step. Individual weights can be modified independently [12, 49], or compression can be performed in a structured manner at the level of neurons or layers [55]. These algorithmic choices give rise to a large space of possible compression strategies, each of which has different overall performance characteristics in terms of space consumption, inference time, and accuracy preservation. Helping ML practitioners navigate this space is a key design opportunity addressed in our work (see Sec. 3).

2.2 Pitfalls in Evaluating Compressed Models

While compression techniques are designed to improve model efficiency while preserving accuracy, they have been known to substantially alter model behavior even while maintaining similar top-level metrics. For example, Hooker et al. [27] find that pruning image classification models has negligible effects on overall accuracy but disproportionately impacts the accuracy of rare subgroups. Similarly, Liebenwein et al. [34] find that pruning image models often results in poor generalization on distribution-shifted inputs.

Since these behavior changes do not always impact top-level metrics, they can be hard to identify in advance without conducting bespoke analyses dedicated to finding them. For example, a prior interview study with ML practitioners [24] described a situation where an object detection model produced jittered outputs after quantization, a phenomenon that was not uncovered until the model was tested on-device in a demo setting. While tools for model comparison are applicable to this task, compression poses additional analytical challenges, such as comparing more than two models at once and understanding how models are derived from one another. Our work aims to address these challenges by providing practitioners with visual tools to inspect the behavior of several compressed models during development.

2.3 Visualization for Model Understanding and Comparison

Visualization has been essential in building generalizable knowledge about ML architectures [23, 60, 67] and helping practitioners make sense of specific models [1, 28, 50, 56, 62, 63]. Many tools use comparison to make insights about model behavior and internals more meaningful, e.g., by jointly visualizing embedding spaces to distinguish meaningful data clusters from spurious ones [5, 54]. Other comparative approaches extend analysis subtasks to multiple models, including comparing model errors [41], instance-level outputs [19, 64], or internal representations [40]. Tools have been developed to help practitioners select from a wide array of models using comparisons of top-level metrics [52, 68]. In our work, we use comparative visualization to help users identify promising strategies and filter out unviable experiments.

Generating and evaluating *compressed* ML models is a more nascent area in VIS4ML—most prior work for this task is limited to either specific compression techniques or profiling efficiency metrics without considering behavior. For example, *CNNPruner* [32] uses a Taylor

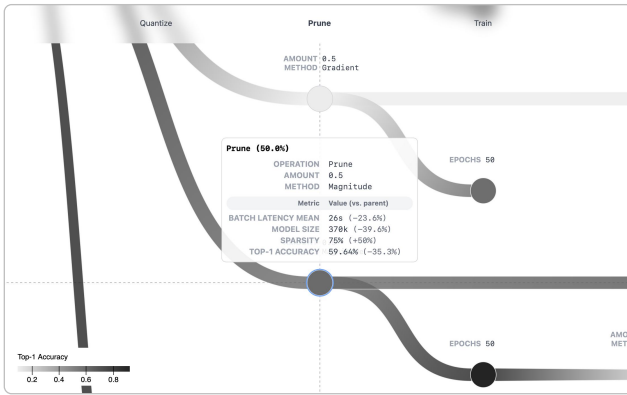


Fig. 2: Hovering over a **Model Map** model displays a tooltip containing the models’ top-level metrics, including latency, size, sparsity, accuracy, and compression operation. Here, the selected model has been 50% pruned, improving its latency and size but reducing its accuracy.

expansion criterion to prune filters in convolutional networks, while ViNNPruner [53] supports a wider variety of architectures but uses an interactive pruning scheme that is difficult to scale to large models. Meanwhile, Talaria [25] helps users estimate the effects of compression on the efficiency of any model, but it requires accuracy to be evaluated separately. Unlike these prior works, our system aims to support more general, iterative compression workflows that may involve dozens of models with different sets of techniques applied.

Overall, these prior systems have focused on either model compression or model comparison alone without taking into account how the two tasks are often intertwined during a model development pipeline [24]. Evaluating the potential of interactive tools to help at the intersection of these two challenges is the primary focus of our work.

3 DESIGN CHALLENGES FOR COMPRESSION

To identify key compression challenges and motivate the design of interactive tools for compression, we synthesized insights from ML and HCI literature. Recently, Hohman et al. [24] explored ML practitioners’ needs and perspectives on model compression via an interview study with 30 compression experts. This work describes how compression experts experiment with different compression techniques to satisfy efficiency and accuracy constraints within multidisciplinary teams. While their focus was providing a broad overview of compression experts’ tacit knowledge, we distill specific findings from their study that are relevant to the design of compression tools and supplement them with recent insights from ML literature to synthesize key challenges for our system to address.

C1. Identifying the optimal compression strategy is time-consuming and task-specific. A one-size-fits-all strategy for model compression does not exist. Even compression experts do not know how to achieve the best balance of efficiency and accuracy a priori and often experiment with multiple compression algorithms for each new task and model [24, 25]. However, existing compression tools focus on exploring the results of a single compression experiment [43, 52, 61] instead of assessing the design space of all possible experiments. As a result, compression experts in our evaluation study (Sec. 6) had complex and time-consuming comparison workflows, such as flipping between the same tool loaded with different models and maintaining large spreadsheets of model results. Compression tools should support practitioners in comparing compressed models based on their performance, efficiency, behavior, and provenance to identify promising strategies.

C2. Compression requires human trade-offs across multiple metrics. Practitioners often discuss model compression as an effort to meet resource targets on memory, time, and accuracy [24]. These budgets are often negotiated, set, and adjusted based on how models would affect the user experience. For example, a model that could run overnight

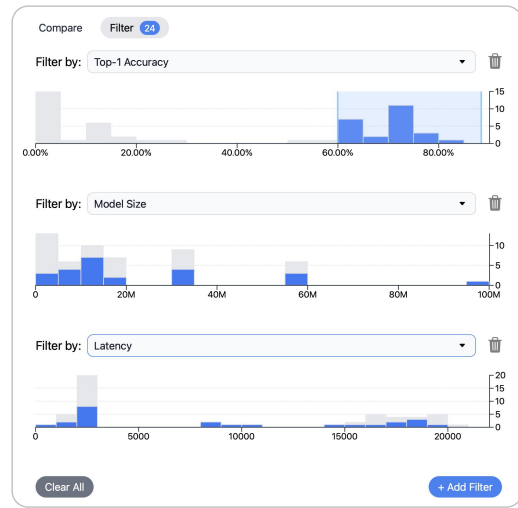


Fig. 3: The **Filter** view allows users to define hard budgets using metrics, like accuracy, model size, or latency. The user can add several filters, see the distribution of values for the selected metrics, and brush on the histograms to disable models with metric values outside of the range.

while a device is charging would be granted more memory and time since it would be unlikely to disrupt the user. Similar decisions take place around accuracy; a model that errs more on sensitive or critical subgroups would not be deployed, while one that makes reasonable or recoverable errors could be viable [24]. As a result, practitioners need tools to quickly assess model variants and make trade-offs between their metrics to help them satisfy these multifaceted constraints.

C3. Top-level metrics can obscure important differences between compressed models. Although compression can achieve comparable performance to the original model, it can alter the model’s behavior by introducing new errors or biases. In some cases, these differences are random or imperceptible, but in others, they can be problematic, such as reducing the quality of model predictions [24], changing model explanations [34], or increasing bias [27]. Unfortunately, behavioral changes do not always correspond to a change in evaluation metrics, so humans must be involved in the evaluation process to catch dangerous behaviors. While various visualization techniques compare the behaviors of *pairs* of models [5, 40, 54], there remain opportunities to help practitioners compare *many* compressed models’ behavioral changes.

C4. Compression can have unintended, hard-to-debug effects on model internals. Practitioners often have a set of heuristics they expect compression algorithms to follow, such as not compressing early layers and compressing layers proportional to their number of parameters. However, when compression algorithms do not follow these expectations, it can be difficult for practitioners to determine why [24]. For example, a network’s outputs may change significantly because a layer ceased to produce meaningful output or because its activations had a different distribution that caused downstream layers’ outputs to change as well. To ensure compression only impacts the desired portions of the model, practitioners often go layer-by-layer to find errors and bottlenecks. Particularly for deep networks with hundreds of layers and billions of intermediate outputs, parameter-wise model comparison becomes a challenging task and an opportunity for visualization tooling.

4 DESIGN OF COMPRESS AND COMPARE

We developed an interactive interface called COMPRESS AND COMPARE to address the four compression design challenges (Sec. 3). The tool consists of two main views: the **Compression Overview** supports high-level comparison and model selection from large-scale compression experiments (C1 and C2), and the **Performance Comparison** view enables fine-grained inspection of model behaviors and internals for a small number of candidate models (C3 and C4).

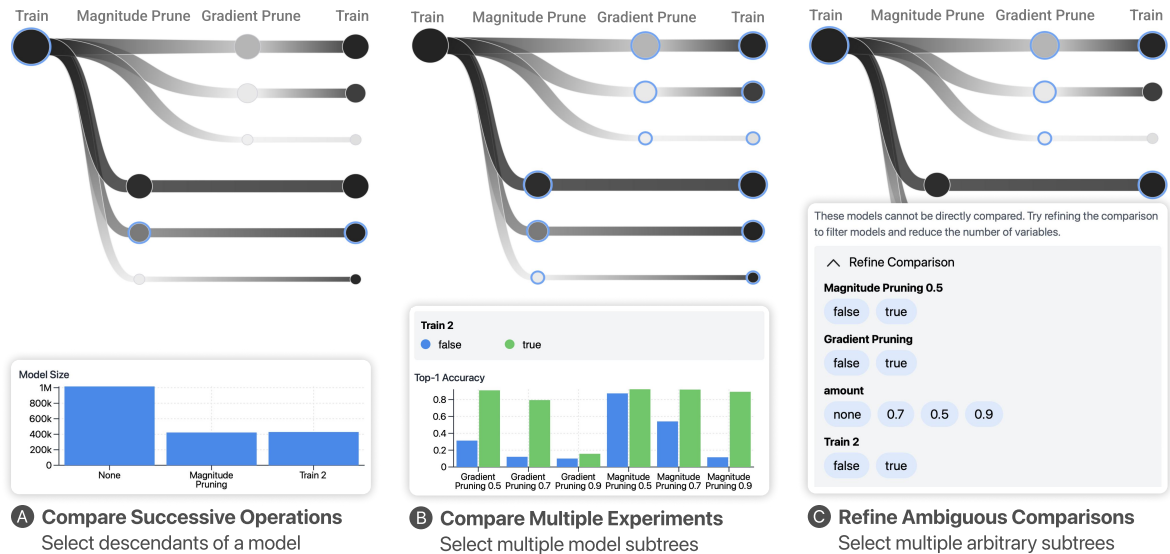


Fig. 4: The **Selection Details** view automatically generates meaningful comparisons between selected **Model Map** models, such as comparing successive operations by selecting the descendants of a model (A) or comparing multiple compression algorithms by selecting multiple subtrees (B). If a direct comparison does not exist, the interface prompts the user to refine the comparison to the attributes they are interested in visualizing (C).

4.1 Compression Overview

In COMPRESS AND COMPARE, experiments are represented as a set of trees, where each node is a model and edge is an *operation* performed on a parent model to produce a child model. This structural choice helps address one of the main visualization challenges of tracking compression experiments (C1): simultaneously depicting the variation in metrics across models along with dependencies in how the models were generated. The **Model Map** (Fig. 1A) addresses this using a node-link tree diagram, where nodes are positioned using a custom algorithm that vertically aligns nodes based on either the operations used to produce them or their step in the compression experiment. Models are rendered as circles whose color and size encode performance properties, commonly accuracy and model size. To emphasize the sequential nature of compression experimentation, the color and width of the edges smoothly interpolate between the parent and child nodes. Hovering over a model displays a tooltip with the model’s top-level metrics (Fig. 2), such as latency, size, sparsity, accuracy, and the operation that created this new model from its parent.

While the **Model Map**’s layout prioritizes understanding model dependencies, the **Model Scatterplot** (Fig. 1B) visualizes model metrics along the spatial axes, helping direct the user to viable models and find natural model groupings. For example, the classic Pareto curve often used by compression experts [24] can easily be recreated by setting model size or latency on the *x*-axis and accuracy on the *y*-axis. Node color and size are consistent between the **Model Scatterplot** and **Model Map**, and the two visualizations are connected via brushing and linking [3]. The **Filter** view also depicts model metrics through customizable histograms that can be brushed to filter the **Model Map** and **Model Scatterplot**. When the user specifies a filter, models that do not meet the filter criteria become semitransparent and unselectable. This allows users to narrow down the set of viable models by expressing their project’s space and performance budgets (C2).

When one or more models are selected, the user can view information about the models’ metrics in the **Selection Details** view (Fig. 1C). While displaying metrics for multiple models in a table would be straightforward, it would obscure the dependency structure between the models, making it harder to reason about the effects of different compression operations. Therefore, we develop a technique to automatically create a grouped bar chart from a subset of the model dependency tree. Our algorithm traverses the tree recursively to identify a minimum-cost set of “variables” that compactly explain the selected models’ differences. Variables include operation parameter values, presence or

absence of an operation, and type of operation applied. The algorithm attempts to fit multiple alternative variable types at each stage of the tree traversal and chooses the assignment that results in the shortest and simplest set of variables (e.g., a variable for an operation’s parameters is considered simpler than a variable for operation type). This ensures that similar operations are mapped to each other. For example, the models resulting from *Prune* → *Quantize* and *Prune* → *Calibrate* → *Quantize* can be explained by *Calibrate = true* or *false*.

If the models can be represented using two variables or fewer, we generate a bar chart by mapping the *x*-axis and color encodings to the two variables. If more variables are required, the algorithm attempts to iteratively simplify the variable set by identifying conditional dependencies and cumulative relationships between pairs of variables. As shown in Fig. 4A and B, this simplification allows us to visualize several successive operations as a single *x*-axis encoding or combine multiple variables that are related to the same operation. When the list of variables needed to describe the selection cannot be simplified to two encodings, a **Refine Comparison** view allows the user to generate bar charts for subsets of the selection that *can* be compared (Fig. 4C).

4.2 Performance Comparison

While the **Compression Overview** visualizes trends across large sets of models, the **Performance Comparison** view enables a deeper comparison of a smaller group of models’ **Behaviors** and **Layers**. We use a combination of juxtaposition and explicit encodings [18] to enable comparisons of multiple models at a time. Comparison is performed with respect to a base model, which is user-customizable but defaults to the selected model closest to the tree’s root node.

The **Behaviors** tab (Fig. 5, left) presents the results of evaluating each selected model on a validation dataset. Each model is a column in a table, where rows represent either class-level comparisons or individual instances. When configuring their data, users can define *comparison metrics* that operate on the model outputs, enabling comparisons like differences in top-1 predictions or the KL divergence of the softmax probabilities. These comparison metrics are summarized in the table headers and are depicted as sparkline bar charts in each row. Notably, the interface supports both absolute per-model values and relative values compared to the base model. Users can sort and filter by absolute or relative metrics for any model, allowing them to quickly identify classes or instances impacted the most by compression (C3).

The final and lowest-level component of the interface is the **Layers** tab (Fig. 5, right), which exposes the internals of the selected models.

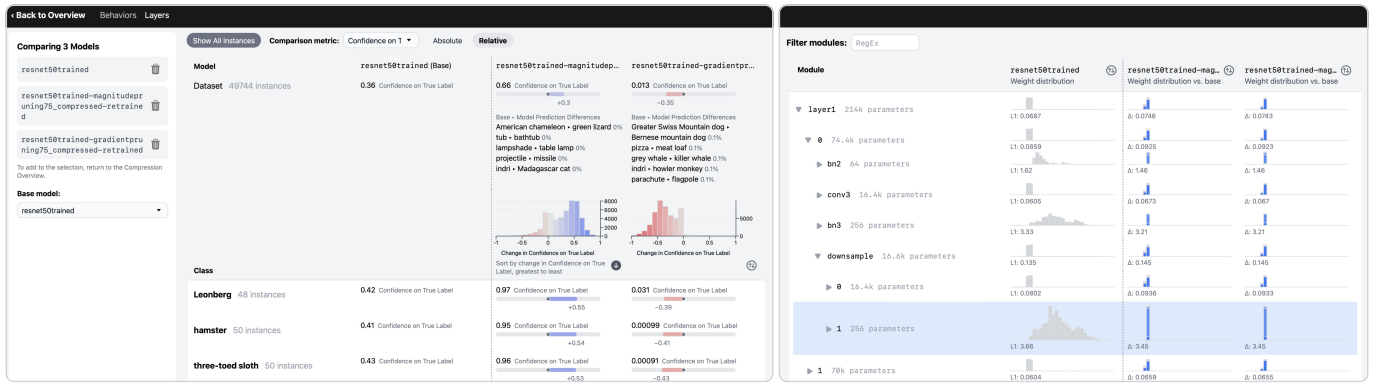


Fig. 5: The **Performance Comparison** view provides an in-depth comparison of two or more models. The **Behaviors** tab (left) displays differences between models’ predictions, distributions of comparison metrics, and a breakdown of the selected comparison metric at the class or instance level. Meanwhile, the **Layers** tab (right) compares the sparsity, weights, and activations across layers in the models using a file tree structure.

Like the **Behaviors** tab, this view comprises a table where each column contains information about a model relative to the base; however, here, each row represents a module in the nested hierarchy of modules that makes up each network. Within this structure, the user can choose from visualizing the proportion of zero weights in each module, the distribution of the weight values, and the distribution of the activations (intermediate outputs) on a random data sample. Weight values and activations are depicted as stacked histograms so that the height of the bars forms the overall value distribution while the color indicates the degree of change relative to the base. This highlights parameters and models that have changed more than others, which can reveal bugs such as over-pruned layers or outlier activations (C4).

4.3 Setup and Implementation Details

We designed COMPRESS AND COMPARE for a highly customizable user workflow. To begin visualizing models, users write a simple Python script that invokes the COMPRESS AND COMPARE backend server and provides information about the models. Users specify models as a JSON object that details the operations used to produce each model and their performance across a set of user-defined metrics. Users can easily integrate specification creation into their existing model training procedures by updating the JSON file each time they train and evaluate a new model or evaluate against a new metric. To access information about the model’s behaviors and layers, users write Python callbacks to retrieve instance-level outputs and layer activations, both of which can be either pre-computed or evaluated in real-time. This flexibility allows COMPRESS AND COMPARE to support any Python-based ML toolchain and accommodate very large models. Additionally, the COMPRESS AND COMPARE Python package provides helper functions to accelerate setup with common frameworks such as PyTorch and HuggingFace. The model servers for the use cases in this paper require around 200 lines of code, mostly consisting of boilerplate code that would already have been written in the course of experimentation.

The COMPRESS AND COMPARE frontend, implemented in SvelteKit¹, is static and can be hosted publicly. Visualizations are developed using D3.js² and LayerCake³. Code is available at: <https://github.com/apple/ml-compress-and-compare>.

5 CASE STUDIES OF COMMON COMPRESSION TASKS

We illustrate how COMPRESS AND COMPARE supports real-world compression workflows via two case studies.

5.1 Repairing Models Broken By Compression

A previously accurate model can “break” when compression is applied too heavily or broadly, resulting in low performance and nonsensical

outputs. However, it can be challenging for users to determine which components of a model are causing its performance to degrade after compression (Sec. 6.2.2). We demonstrate how COMPRESS AND COMPARE can help practitioners identify and resolve breakages (C4) in the context of a generative language model for question answering.

We use an off-the-shelf T5-Large model [46] that achieves an F1 score of 90.5% on the Stanford Question Answering dataset [47] (Fig. 6A). The model’s original performance is competitive with humans’, but since the model is large (775 million parameters), we’d like to compress it to improve its speed and space utilization. Following common compression workflows from our participants (Sec. 6.2) and the literature (Sec. 2), we apply magnitude pruning across all of the model’s parameters. However, this causes steep performance drops even at low levels of compression (e.g., 4% F1 after pruning only 10% of parameters). Looking at the top changes in predicted answers in the **Behaviors** view (Fig. 6B), we see that magnitude pruning has broken the model’s generation. The 10% pruned model repeats words from the context paragraph (e.g., “Super Bowl LII LII LII . . .”), and the 30% pruned model’s output is meaningless (“a a . . .”).

COMPRESS AND COMPARE can help us understand why magnitude pruning has negatively affected the model. There are many possible reasons this compression strategy could have failed—the model may have low compressibility, essential weights may have been inadvertently pruned, or magnitude pruning may not be well-suited to this task. Since it is challenging to determine the cause using performance alone, we use the **Layers** view to inspect parts of the model that have been pruned. Sorting the models’ layers by how much their weights have changed, we see that the most changed layers are all normalization layers (Fig. 6C). Normalization layers ensure a consistent activation distribution throughout the model, so over-pruning them can lead to unexpected behavior. However, since the model has relatively few normalization weights, we would not necessarily expect magnitude pruning to have pruned them so aggressively. To test if pruning the normalization layers caused the performance drop, we design a follow-up experiment that restores the normalization layers in the pruned models to match the original model, effectively unpruning them. This leads to a full recovery in F1 for the 10% and 30% pruned models, indicating that pruning the normalization layers was a substantial issue in our original compression experiment.

We can also use COMPRESS AND COMPARE to understand if the repaired models can be pruned any further. To do so, we browse model activations in the **Layers** view for the original model, the 30% pruned model with restored normalization layers (the fixed model), and the 50% pruned model with restored normalization layers (a broken model). We observe that the outputs of the self-attention module have changed significantly during pruning, even in the working model, which may signify that the model is robust to changes in these modules. By pruning additional parameters from the attention modules, we can reduce the model size by roughly 30% while achieving 83% F1 score.

¹<https://kit.svelte.dev>

²<https://d3js.org>

³<https://layercake.graphics>

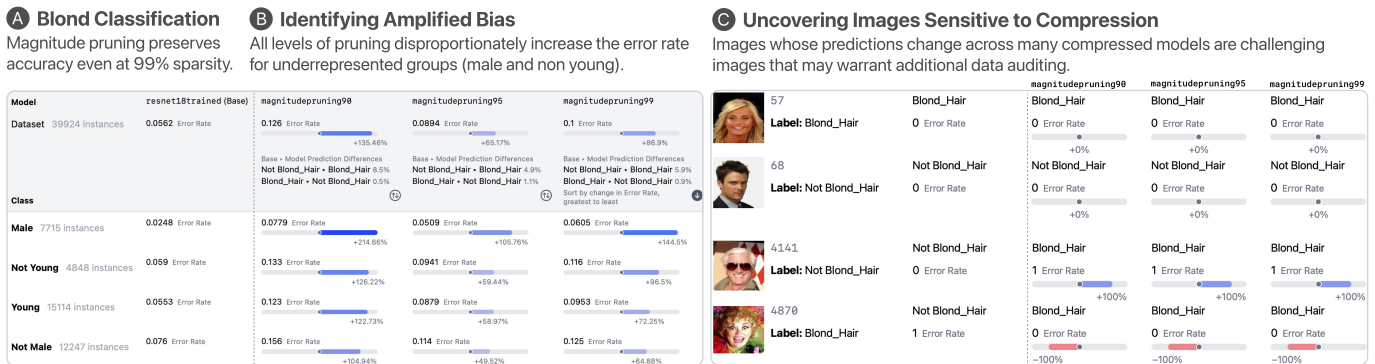


Fig. 7: COMPRESS AND COMPARE can help identify compression induced bias and perform data auditing. The **Behaviors** view reveals that compressing image classification models (A) disproportionately impacts rare classes (B) by forgetting hard-to-classify images (C).

COMPRESS AND COMPARE to complete two tasks:

T1. First, we showed participants a ResNet50 image classification model [9, 21, 45] and 19 of its compressed variants, including 8-bit quantization, global gradient and magnitude pruning at various sparsity levels, and post-pruning fine-tuning and calibration. We asked participants to identify successful experiments, explain why some experiments failed, and suggest future experiments.

T2. Next, we updated the interface to display all 52 models, including the 20 models from **T1**, a MobileNet V2 model [51] with global magnitude pruning, iteratively pruned and fine-tuned models, and models with combined compression techniques. We asked participants to evaluate the latency, size, accuracy, and behavior of these models and select the model(s) they felt confident deploying.

To analyze the results of the study, two authors reviewed the video-recorded sessions. They performed open coding to identify participants’ insights during the tasks and their broader perspectives on compression and COMPRESS AND COMPARE. Using the resulting 357 codes, they conducted an iterative affinity diagramming and thematic analysis process to identify progressively higher-level themes. This process generated 17 final themes that we use to structure **Sec. 6.2**.

6.2 Study Results

Overall, participants reported that COMPRESS AND COMPARE provided a structured compression workflow, enabling them to perform analyses that would have been challenging and time-consuming with existing tools, build intuition about compression techniques that inspire new compression experiments, and identify subtle but problematic model behaviors introduced during compression.

6.2.1 Unifying Disjoint Compression Workflows

Our user study participants reported needing multiple tools to execute their current compression tasks, leading to tedious back-and-forth analysis across tools that made it challenging to understand the overall impact of their compression experiments. While each participant was knowledgeable about compression (average 3.5 on a self-reported 1–5 scale of compression experience) and used compression regularly (5/8 use compression in every project), there was not a standard set of tools used by all participants. This inconsistency was often due to a lack of compression-specific tooling that supported the breadth of participants’ tasks, including training and compressing models (P1–P8), evaluating their performance and behavior (P4, P6, P7), and ensuring they meet specified efficiency budgets (P1–P8). As a result, participants often created custom tools, such as Jupyter Notebook charts for performance analyses (P1, P6, P8) and spreadsheets for model tracking (P3, P5), or they repurposed existing general model analysis tools to understand the performance of a compressed model (P1–P4). When compression-specific tools existed, participants found them invaluable to their analysis (P3, P4). However, these tools were often still specific

to a particular aspect of compression analysis (e.g., hardware performance [25]). As a result, participants found it challenging to perform comprehensive analysis and lamented that existing tools did not support many of their most critical tasks, like budgeting (P2), layer-wise analysis (P1), and compression-specific comparison (P2, P3, P4).

Unlike participants’ existing workflows, with COMPRESS AND COMPARE, critical compression tasks, like metric analysis and budgeting (**C2**), experiment comparison (**C1**), and layer-wise behavioral inspection (**C3** and **C4**) are all in one place:

“[In] a lot of my typical workflows [...] you have to have 10 tabs in parallel in your browser and switch between them. I find that [COMPRESS AND COMPARE] is really bringing all the different aspects into a single view.” — P2

For instance, an approach used by 4/8 participants was to identify candidate models that fit their performance budget via the **Filter**, hone into the one or two best-performing models using the **Model Scatterplot** metrics, and search for patterns in the best models’ compression recipes using the **Model Map**. Through this process, participants quickly identified “deployable” models that were small enough to fit on device while still maintaining task performance. P1, P5, P6, P7, and P8’s **Compression Overview** analysis quickly converged from 52 models to a single 8-bit quantized ResNet50 model that reduced latency and size while nearly maintaining the uncompressed model’s accuracy. Having a comprehensive overview of compressed model variants gave participants confidence to pitch this model to their teammates and use it to design new experiments that could result in even greater efficiency.

The unified compression interface also sparked discussion about how COMPRESS AND COMPARE could integrate into collaborative compression settings. Participants regularly collaborate with team members to complete their compression tasks, such as sending compressed models to QA specialists for targeted evaluation (P4, P5), negotiating resource budgets with product managers (P6, P8), and mentoring model developers on compression methods (P2). However, it can be challenging to collaborate on a compressed model across various tools and with collaborators with varying skill sets. In participants’ current workflows, experimental results are distributed across tools, so participants were excited to use COMPRESS AND COMPARE as a centralized communication tool. P6 and P8 were interested in presenting their experiments to budget managers using COMPRESS AND COMPARE to advocate for budget increases by interactively demonstrating how best-case model performance improves as the budget relaxes. Participants also expressed interest in using COMPRESS AND COMPARE to collaboratively compress models, such as by flagging potential compression-induced issues in the **Behaviors** view for review by their QA teammates (P4, P5) and setting up experiments in the **Model Map** that demonstrate compression pitfalls to less experienced engineers (P2). Whereas existing compression workflows tended to become ad hoc when experimenting with many different algorithms, techniques, and pipeline structures, participants were excited for COMPRESS AND COMPARE to provide structure to the collaborative search for an efficient and accurate model.

6.2.2 Building Intuition about Model Compression

Beyond simply selecting a desired compressed model, COMPRESS AND COMPARE’s visual and interactive components helped participants build intuition for how compression algorithms impact model performance and generate hypotheses about ways to improve future experiments. Participants found the combination of the **Model Map** and the **Selection Details** view to be an intuitive way to understand and reason about the space of compression experiments. Viewing the columns in the **Model Map** helped users understand the set of compression algorithms that had been applied (P3, P4, P7) and identify patterns in how the best-performing models were generated (P2, P7). To dig into a particular pattern, participants would often run visual “experiments” by selecting a group of models within a region of the **Model Map** (e.g., all the magnitude pruned models) and comparing their metrics in the **Selection Details** view (P2, P4, P6, P8). These in-depth explorations influenced participants’ intuitions about how compression techniques affected their models more broadly. For instance, by comparing the accuracy and efficiency of two quantized models, P4 identified that quantization preserved performance much better for a large ResNet50 (25.6M parameters) than for a smaller MobileNet V2 (3.5M parameters). While P4 regularly uses quantization, this discrepancy in performance caused them to reflect that the success of quantization “*is definitely dependent on the base model; if you want something to work for quantization, you have to start at the right place.*” Finding the best compression technique is challenging (C1), so building intuition for the types of models that benefit from quantization can help P4 design more effective compression recipes moving forward, such as those that only apply quantization to large models.

Building on their high-level understanding of the experimental space, participants used the **Performance Comparison** views to develop a deeper intuition about compression’s impact on models’ internal representations. Using the **Layers** view, participants debugged subtle problems in compression experiments that led to poor model performance. For example, P8 recognized that a particular model’s “*batch norms had been absolutely flattened*” by quantization. Batch normalization layers can have a substantial impact on downstream performance because they set the output value ranges at each layer, so this finding led P8 to suggest “*freezing all the batch norms*” during quantization as a way to maintain model performance. By building intuition through their **Layers** exploration, participants ideated a range of subsequent experiments. These next steps included combining current compression techniques in new ways (e.g., combining magnitude and gradient pruning) and expanding the space of operations, such as by tailoring compression to specific layers of the network. For instance, in their **Layers** analysis, P5 and P7 noticed earlier network layers had fewer parameters yet were pruned at the same rate as later layers. They hypothesized that, since later layers have more parameters, they have more redundancy and could withstand greater compression rates, so they designed an experiment that pruned layers as a function of their number of parameters. With COMPRESS AND COMPARE, participants deepened their understanding of how model parameters respond to compression techniques (C4) and used their insights to generate new experiments that could lead to more efficient and accurate models.

6.2.3 Encouraging Comprehensive Compression Analysis

By interactively integrating behavioral analysis with traditional metric-based compression analysis, COMPRESS AND COMPARE extended participants’ existing behavioral analysis workflows and motivated them to consider the broader impacts of compression. Evaluating compressed model behavior on held out data was a key aspect of some participants’ workflows (P5–P8) because it helped them identify subtle but important behavioral changes (C3):

“When you compress a model you care about its quality on rare classes. The biggest risk when you compress your model is all of a sudden it becomes [problematic].” — P8

Participants used the **Behaviors** view to analyze model behavior across an entire dataset to ensure that compression had not induced biases

or spurious correlations. The ability to sort by relative change in correctness helped them identify classes that experienced the most errors and inspect individual instances that were misclassified. This procedure revealed that many compressed models’ mistakes were acceptable, such as mistakes on multi-object images (e.g., coffee pot in a stove image) or related classes (e.g., Great Dane misclassified as another dog breed). However, it also helped them uncover subtle patterns and identify potential compression-induced biases. For instance, P5 sorted the **Behaviors** view by decreasing change in correctness and observed that stove images had lost 22% accuracy, whereas overall the model only experienced a few percent decrease. Inspecting the stove images whose classifications had changed revealed a spurious correlation between stoves and microwaves. P5 worried that the compressed model could be relying on the presence of one to classify the other. By viewing model behavior over an entire dataset, participants, like P5, were able to identify concerning patterns in the model’s behavior, hypothesize reasons for the problem (e.g., a disproportionate amount of training images contain both objects), and develop a plan to address them (e.g., flagging these examples for QA team review).

While participants primarily analyze correctness in their current workflows, having access to additional plug-and-play metrics in the **Behaviors** view spurred new analysis processes. During P1’s behavioral analysis, they discovered that magnitude pruning resulted in a larger KL divergence in output probabilities than quantization. While they do not use KL divergence in their standard analysis pipeline, having access to it in COMPRESS AND COMPARE helped them distinguish between otherwise similar compressed models and select the one that best reflected the original model’s outputs.

“I don’t look at KL divergence very frequently, but KL divergence is zero for [the quantized] model and non-zero for [the pruned] model. There was only a 6% accuracy regression [for the pruned model], so it’s surprising. I’m impressed by the KL divergence [of the quantized model] being extremely low. It’s the incumbent solution.” — P1

Similarly, P2 and P3 uncovered that magnitude pruning resulted in higher model confidence than quantization. With the knowledge that these differences in model outputs existed, participants were able to generate hypotheses for their existence (e.g., confidence increases may be a result of overfitting with fewer parameters (P2)) and strategies to account for them (e.g., setting a different prediction threshold based on the compression algorithm (P3)). Overall, COMPRESS AND COMPARE extended participants’ compression workflows to integrate behavioral analyses with their standard metric-based budgeting procedures. As a result, participants seamlessly switched between the two, iteratively selecting a candidate compressed model and interrogating its behavior to identify biases and hypothesize new compression experiments.

7 DISCUSSION

We present COMPRESS AND COMPARE, an interactive visualization system for tracking and comparing compression experiments. Based on challenges experienced by real-world users, we design COMPRESS AND COMPARE to support critical and unsupported compression analysis tasks, including managing interconnected compression experiments, interrogating the impact of compression on model behaviors, and ideating promising future compression experiments. Through case studies on generative language and image classification models, we demonstrate how our system helps users repair issues with compression and identify compression-induced bias. Moreover, our user study with compression experts illustrates how COMPRESS AND COMPARE shifts users’ compression workflows from disjoint analysis across tools toward a single analysis platform that facilitates collaborative decision-making about model selection and exploration. Here, we discuss the implications of our results for future ML development and evaluation tools, as well as the current limitations of our work and possible solutions.

7.1 Designing Compression-Aware ML Workflows

Throughout the design and analysis of COMPRESS AND COMPARE, we encountered compression-specific challenges. While compression

analysis could be considered a special case of general ML evaluation, our user study participants struggled to extend traditional model evaluation workflows to their compression-specific tasks. Our work suggests ways future tools can improve the overall ML development process by integrating compression considerations:

Bridging data- and model-centric evaluations. Most existing ML development tools either focus on data-centric evaluations of model accuracy and behavior [5, 6, 50, 54] or architecture-specific evaluations of model internal layer characteristics [25, 42]. In contrast, we found that linking data-centric (i.e., **Behaviors**) and architecture-specific visualizations (i.e., **Layers**) helped users interpret the functional characteristics of model components, similar to systems like DeepCompare [40] and ActiVis [28]. For instance, in our case studies and user studies, participants used the connection between the **Layers** and **Behaviors** views to identify that compressing batch normalization layers directly worsened the quality of the model’s outputs. By identifying a functional relationship between the model’s architecture and its behavior, users were then able to ideate new, better-performing experiments (e.g., removing compression on normalization layers). By developing joint visualizations of evaluation data and model architectures, compression tools can help users connect aspects of a model’s design to changes in its behavior.

Negotiating trade-offs between model quality metrics. Compression practitioners often trade off between model size, latency, and accuracy. While existing ML pipelines have addressed similar issues between accuracy and fairness [58] and accuracy across tasks [35], interactive model selection tools have not explicitly explored helping practitioners make these trade-offs, e.g., by identifying Pareto frontiers. Further, our user study participants indicated that efficiency and accuracy budgets are often collaborative and malleable targets, as opposed to hard quantitative thresholds. By visualizing experimental results and metric trade-offs, compression tools can help practitioners communicate their constraints and advocate for budget changes when needed.

Tracking model provenance during iterative development. Unlike model architecture and hyperparameter search, where experiments are often simple Cartesian products of several variables, compression experimentation is more readily modeled as a tree of “recipes” where nodes represent models and edges represent operations. Practitioners often start with a single model or a few related models that are known to perform well and apply varying compression recipes to them, creating the branching structure depicted in the **Model Map**. This process results in many interconnected models, and it can be challenging to keep track of the operations that created a particular model and its relationship to other models. User study participants found visualizing models in this tree structure helped them build intuition for aspects of experiments that worked well and how future experiments may behave. Future compression tools may consider a similar tree visualization or new ways to communicate model provenance to practitioners.

Comparing complex differences across many models. Existing tools (including those for compression) tend to focus on profiling and evaluating a single model [1, 6, 25, 53]. In contrast, our study underscores the value of directly supporting comparison [18, 54]. By visually juxtaposing metrics, predictions, and internals from several models, COMPRESS AND COMPARE reduces the cognitive load required to determine which differences are most actionable. Our system enables workflows that rapidly transition between *comparative relationships*, ranging from black-box comparisons of top-level metrics to comparisons of individual layer activations. Future tools can prioritize comparison and look to practitioners’ existing comparison strategies to understand when linking multiple comparative relationships leads to productive insights.

These compression-specific challenges provide a basis for the design of future compression-vis tools, but they could also suggest ways to improve general ML analysis workflows. For instance, our user study participants found the **Model Map** tree structure to be highly intuitive, and some suggested applying it to other stages of model development (P4, P18, P15), such as creating a timeline-based **Model Map** that organized model results based on when a user ran each experiment. Moreover, extending the layer-wise activation comparison in the **Layers** view could support complex hyperparameter search workflows.

Additionally, features demonstrated in COMPRESS AND COMPARE may be worth integrating into general-purpose ML tools, helping those tools cover a broader range of existing workflows and encouraging practitioners to focus on efficiency earlier in the development process.

7.2 Limitations and Future Work

We designed COMPRESS AND COMPARE to support practitioners current compression workflows; however, as model compression becomes a more established and standardized discipline, it is possible that many of the iterative workflows we observed in this study will be superseded by automated approaches. However, interactive visualization has potential benefits for compression work even if automated approaches eventually become standard. First, even though effective AutoML strategies exist, these techniques often still require data scientists to invest considerable time to distill model behavior into a single objective function [29]. Second, COMPRESS AND COMPARE seeks not only to help practitioners produce the best compressed model, but also to help them build intuition about compression techniques and how they affect model characteristics. Just as prior work visualizing ML model internals and behaviors [30, 48, 69] has allowed people to understand how these models work, a better collective understanding of compression can make these techniques more accessible and inspire new approaches.

Our user study and prior formative research primarily interviewed compression experts and ML engineers who were already well-versed in model compression. While this allowed us to get the most relevant feedback on how to design compression-specific systems, our system and takeaways do not consider the needs of novice users. It is likely COMPRESS AND COMPARE could be extended to support ML practitioners less accustomed to compression, such as by providing suggestions on what techniques to try or incorporating a graphical interface to run compression experiments. Such adaptations would empower non-experts to apply compression, but they could also support compression experts in running on-the-fly experiments based on their insights from COMPRESS AND COMPARE, such as removing compression from a specific layer or making a slight change to the sparsity value.

Further, several participants suggested extensions to COMPRESS AND COMPARE that could help it better match the specific needs of their teams, including supporting very large datasets and models as well as displaying custom efficiency metrics. Although the tool currently supports running model computations remotely or in advance, it does require the models being compared to be loaded simultaneously in memory so that each layer can be compared one-at-a-time. More efficient comparison techniques that do not require jointly loading several models may help COMPRESS AND COMPARE scale more easily.

Finally, a key requirement of COMPRESS AND COMPARE is its integration between the visualization system’s and a user’s model development code. However, like many other prototype tools for ML development, the capabilities that make COMPRESS AND COMPARE a versatile tool can also necessitate considerable set-up work, particularly for custom model architectures. User study participants noted that the potential difficulty of importing models into the tool could be a critical hurdle to accepting it into their workflow. Future work is needed to design code-level interfaces that link COMPRESS AND COMPARE into the tools practitioners already use to develop models.

8 CONCLUSION

Making ML models smaller, faster, and more energy-efficient can enable exciting new user experiences and broaden access to existing ones. Our work forms part of a nascent body of literature on facilitating the process of compressing models, which can help make these use cases practical as models become increasingly large and powerful. Through the design and evaluation of COMPRESS AND COMPARE, we aimed to understand and address challenges in ML model development made salient by compression, including the need for extensive iteration, human-centered trade-offs between user experience metrics, and subtle changes in model behavior. Future compression-focused data visualization research can continue to make creating efficient ML models easier for a wider range of developers and teams, helping them make the experiences they envision a reality.

ACKNOWLEDGMENTS

We thank our colleagues at Apple for their guidance on this research and our case study participants for generously sharing their time and knowledge with us. We especially thank Guillaume Seguin and Xiaoyi Zhang for lending their expertise on compression and encouraging us to pursue this line of research.

REFERENCES

- [1] A. Bäuerle, Á. A. Cabrera, F. Hohman, M. Maher, D. Koski, X. Suau, T. Barik, and D. Moritz. Symphony: Composing Interactive Interfaces for Machine Learning. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 210:1–210:14. ACM, New York, 2022. doi: 10.1145/3491102.3502102 2, 9
- [2] C. Baykal, L. Liebenwein, I. Gilitschenski, D. Feldman, and D. Rus. SiPPing neural networks: Sensitivity-informed provable pruning of neural networks. *arXiv*, 2019. doi: 10.48550/arXiv.1910.05422 2
- [3] R. A. Becker and W. S. Cleveland. Brushing Scatterplots. *Technometrics*, 29(2):127–142, 1987. doi: 10.2307/1269768 4
- [4] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Conference on Fairness, Accountability, and Transparency (FAccT)*, pp. 610–623. ACM, 2021. doi: 10.1145/3442188.3445922 1
- [5] A. Boggust, B. Carter, and A. Satyanarayan. Embedding Comparator: Visualizing Differences in Global Structure and Local Neighborhoods via Small Multiples. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pp. 746–766. ACM, New York, 2022. doi: 10.1145/3490099.3511122 2, 3, 9
- [6] Á. A. Cabrera, E. Fu, D. Bertucci, K. Holstein, A. Talwalkar, J. I. Hong, and A. Perer. Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 419:1–419:14. ACM, New York, 2023. doi: 10.1145/3544548.3581268 9
- [7] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018. doi: 10.1109/MSP.2017.2765695 2
- [8] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 53:5113–5155, 2020. doi: 10.1007/s10462-020-09816-7 2
- [9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. CVF/IEEE, 2009. doi: 10.1109/CVPR.2009.5206848 7
- [10] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. *Proceedings of the IEEE*, 108(4):485–532, 2020. doi: 10.1109/JPROC.2020.2976475 2
- [11] J. Frankle and M. Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. doi: 10.48550/arXiv.1803.03635 2
- [12] E. Frantar and D. Alistarh. SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot. In *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 202, pp. 10323–10337. PMLR, 2023. doi: 10.48550/arXiv.2301.00774 2
- [13] N. Friedman. Introducing github copilot: your ai pair programmer. <https://github.blog/news-insights/product-news/introducing-github-copilot-ai-pair-programmer/>, February 2022. Accessed: 2024-08-01. 1
- [14] T. Gale, E. Elsen, and S. Hooker. The State of Sparsity in Deep Neural Networks. *arXiv*, 2019. doi: 10.48550/arXiv.1902.09574 2
- [15] N. Gamba, K. Kudrolli, A. Dhoot, and A. Pedram. Campfire: Compressible, Regularization-Free, Structured Sparse Training for Hardware Accelerators. *arXiv*, 2020. doi: 10.48550/arXiv.2001.03253 2
- [16] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference. In G. K. Thiruvathukal, Y. Lu, J. Kim, Y. Chen, and B. Chen, eds., *Low-Power Computer Vision: Improving the Efficiency of Artificial Intelligence*, pp. 291–326. CRC Press, Boca Raton, FL, 2022. doi: 10.1201/9781003162810-13 2
- [17] C. Giattino, E. Mathieu, V. Samborska, J. Broden, and M. Roser. Artificial intelligence. *Our World in Data*, 2023. <https://ourworldindata.org/artificial-intelligence.1>
- [18] M. Gleicher. Considerations for Visualizing Comparison. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):413–423, 2018. doi: 10.1109/TVCG.2017.2744199 4, 9
- [19] M. Gleicher, A. Barve, X. Yu, and F. Heimerl. Boxer: Interactive Comparison of Classifier Results. *Computer Graphics Forum*, 39(3):181–193, 2020. doi: 10.1111/CGF.13972 2
- [20] S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. doi: 10.48550/arXiv.1510.00149 2
- [21] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE Computer Society, Piscataway, NJ, 2016. doi: 10.1109/CVPR.2016.90 6, 7
- [22] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 2234–2240. IJCAI, 2018. doi: 10.24963/IJCAI.2018/309 2
- [23] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2674–2693, 2018. doi: 10.1109/TVCG.2018.2843369 2
- [24] F. Hohman, M. B. Kery, D. Ren, and D. Moritz. Model Compression in Practice: Lessons Learned from Practitioners Creating On-device Machine Learning Experiences. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 645:1–645:18. ACM, New York, NY, 2024. doi: 10.1145/3613904.3642109 1, 2, 3, 4
- [25] F. Hohman, C. Wang, J. Lee, J. Görtler, D. Moritz, J. P. Bigham, Z. Ren, C. Foret, Q. Shan, and X. Zhang. Talaria: Interactively Optimizing Machine Learning Models for Efficient Inference. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 648:1–648:19. ACM, New York, NY, 2024. doi: 10.1145/3613904.3642628 2, 3, 7, 9
- [26] S. Hooker, A. Courville, G. Clark, Y. Dauphin, and A. Frome. What Do Compressed Deep Neural Networks Forget? *arXiv*, 2019. doi: 10.48550/arXiv.1911.05248 6
- [27] S. Hooker, N. Moorsos, G. Clark, S. Bengio, and E. Denton. Characterising Bias in Compressed Models. *arXiv*, 2020. doi: 10.48550/arXiv.2010.03058 2, 3, 6
- [28] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):88–97, 2018. doi: 10.1109/TVCG.2017.2744718 2, 9
- [29] S. K. Karmaker (“Santu”), M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni. AutoML to Date and Beyond: Challenges and Opportunities. *ACM Computing Surveys*, 54(8):175:1–175:36, 2022. doi: 10.1145/3470918 9
- [30] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and Understanding Recurrent Networks. *arXiv*, 2015. doi: 10.48550/arXiv.1506.02078 9
- [31] N. Lee, T. Ajanthan, and P. H. S. Torr. SNIP: single-shot network pruning based on connection sensitivity. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. doi: 10.48550/arXiv.1810.02340 2
- [32] G. Li, J. Wang, H. Shen, K. Chen, G. Shan, and Z. Lu. CNNPruner: Pruning convolutional neural networks with visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1364–1373, 2021. doi: 10.1109/TVCG.2020.3030461 2
- [33] Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. Gonzalez. Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers. In *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 119, pp. 5958–5968. PMLR, 2020. doi: 10.48550/arXiv.2002.11794 6
- [34] L. Liebenwein, C. Baykal, B. Carter, D. Gifford, and D. Rus. Lost in Pruning: The Effects of Pruning Neural Networks beyond Test Accuracy. *Proceedings of Machine Learning and Systems*, 3:93–138, 2021. doi: 10.48550/arXiv.2103.03014 2, 3, 6
- [35] X. Lin, H. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong. Pareto Multi-Task Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:12037–12047, 2019. doi: 10.48550/arXiv.1912.12854 9
- [36] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep Learning Face Attributes in the Wild. In *Proceedings of the International Conference on Computer*

- Vision (ICCV), pp. 3730–3738. IEEE Computer Society, Washington, D.C., 2015. doi: 10.1109/ICCV.2015.425 6
- [37] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the Value of Network Pruning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. doi: 10.48550/arXiv.1810.05270 2
- [38] G. Menghani. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. *ACM Computing Surveys*, 55(12):259:1–259:37, 2023. doi: 10.1145/3578938 2
- [39] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance Estimation for Neural Network Pruning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11264–11272. CVF/IEEE, 2019. doi: 10.1109/CVPR.2019.01152 2
- [40] S. Murugesan, S. Malik, F. Du, E. Koh, and T. M. Lai. DeepCompare: visual and interactive comparison of deep learning model performance. *IEEE Computer Graphics and Applications*, 39(5):47–59, 2019. doi: 10.1109/MCG.2019.2919033 2, 3, 9
- [41] S. Narkar, Y. Zhang, Q. V. Liao, D. Wang, and J. D. Weisz. Model LineUpper: Supporting interactive model comparison at multiple levels for automl. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pp. 170–174. ACM, New York, NY, 2021. doi: 10.1145/3397481.3450658 2
- [42] C. Olah, A. Mordvintsev, and L. Schubert. Feature Visualization. *Distill*, 2017. <https://distill.pub/2017/feature-visualization>. doi: 10.23915/distill.00007 9
- [43] J. P. Ono, S. Castelo, R. Lopez, E. Bertini, J. Freire, and C. Silva. Pipeline-Profiler: A Visual Analytics Tool for the Exploration of AutoML Pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):390–400, 2021. doi: 10.1109/TVCG.2020.3030361 3
- [44] OpenAI. Hello gpt-4o. <https://openai.com/index/hello-gpt-4o/>, May 2024. Accessed: 2024-08-01. 1
- [45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:8024–8035, 2019. doi: 10.48550/arXiv.1912.01703 7
- [46] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:140:1–140:67, 2020. doi: 10.5555/3455716.3455856 5
- [47] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2383–2392. ACL, Stroudsburg, PA, 2016. doi: 10.18653/V1/D16-1264 5
- [48] E. Reif, A. Yuan, M. Wattenberg, F. B. Viegas, A. Coenen, A. Pearce, and B. Kim. Visualizing and Measuring the Geometry of BERT. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:8592–8600, 2019. doi: 10.48550/arXiv.1906.02715 9
- [49] A. Renda, J. Frankle, and M. Carbin. Comparing Rewinding and Fine-tuning in Neural Network Pruning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. doi: 10.48550/arXiv.2003.02389 2
- [50] S. Robertson, Z. J. Wang, D. Moritz, M. B. Kery, and F. Hohman. Angler: Helping Machine Translation Practitioners Prioritize Model Improvements. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 832:1–832:20. ACM, New York, 2023. doi: 10.1145/3544548.3580790 2, 9
- [51] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520. CVF/IEEE, 2018. doi: 10.1109/CVPR.2018.00474 7
- [52] S. Schelter, J.-H. Böse, J. Kirschnick, T. Klein, and S. Seufert. Automatically Tracking Metadata and Provenance of Machine Learning Experiments. In *Workshop on ML Systems at Advances in Neural Information Processing Systems (NIPS)*, 2017. 2, 3
- [53] U. Schlegel, S. Schiegg, and D. A. Keim. ViNNPruner: Visual Interactive Pruning for Deep Learning. In *Machine Learning Methods in Visualisation for Big Data (MLVis@EuroVis)*, pp. 13–17. Eurographics Association, Eindhoven, The Netherlands, 2022. doi: 10.2312/MLVIS.20221070 2, 3, 9
- [54] V. Sivaraman, Y. Wu, and A. Perer. Emblaze: Illuminating Machine Learning Representations through Interactive Comparison of Embedding Spaces. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pp. 418–432. ACM, New York, 2022. doi: 10.1145/3490099.3511137 2, 3, 9
- [55] X. Suau, L. Zappella, and N. Apostoloff. Filter Distillation for Network Compression. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pp. 3129–3138. IEEE, Piscataway, NJ, 2020. doi: 10.1109/WACV45572.2020.9093546 2
- [56] H. Suresh, D. Shammugam, T. Chen, A. G. Bryan, A. D’Amour, J. Gutttag, and A. Satyanarayan. Kaleidoscope: Semantically-grounded, context-specific ML model evaluation. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 775:1–775:13. ACM, New York, 2023. doi: 10.1145/3544548.3581482 2
- [57] M. Treviso, J.-U. Lee, T. Ji, B. van Aken, Q. Cao, M. R. Ciosici, M. Hassid, K. Heafield, S. Hooker, C. Raffel, P. H. Martins, A. F. T. Martins, J. Z. Forde, P. Milder, E. Simpson, N. Slonim, J. Dodge, E. Strubell, N. Balasubramanian, L. Derczynski, I. Gurevych, and R. Schwartz. Efficient Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 11:826–860, 2023. doi: 10.1162/tacl_a_00577 2
- [58] A. Valdivia, J. Sánchez-Monedero, and J. Casillas. How fair can we go in machine learning? Assessing the boundaries of accuracy and fairness. *International Journal of Intelligent Systems*, 36(4):1619–1643, 2021. doi: 10.1002/INT.22354 9
- [59] P. Villalobos, J. Sevilla, T. Besiroglu, L. Heim, A. Ho, and M. Hobbahn. Machine Learning Model Sizes and the Parameter Gap. *arXiv*, 2022. doi: 10.48550/arXiv.2207.02852 1
- [60] J. Wang, S. Liu, and W. Zhang. Visual Analytics for Machine Learning: A Data Perspective Survey. *IEEE Transactions on Visualization and Computer Graphics*, 2024. To appear. doi: 10.1109/TVCG.2024.3357065 2
- [61] D. K. I. Weidele, J. D. Weisz, E. Oduor, M. Muller, J. Andres, A. Gray, and D. Wang. AutoAIViz: Opening the Blackbox of Automated Artificial Intelligence with Conditional Parallel Coordinates. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, p. 308–312. ACM, New York, NY, 2020. doi: 10.1145/3377325.3377538 3
- [62] M. M. Welsh, D. Koski, M. Sarabia, N. Sivakumar, I. Arawjo, A. Joshi, M. Doumbouya, X. Suau, L. Zappella, and N. Apostoloff. Data and Network Inspection Kit, 2023. 2
- [63] T. Wu, M. T. Ribeiro, J. Heer, and D. S. Weld. Errudite: Scalable, Reproducible, and Testable Error Analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 747–763. ACL, Stroudsburg, PA, 2019. doi: 10.18653/V1/P19-1073 2
- [64] X. Xuan, X. Zhang, O.-H. Kwon, and K.-L. Ma. VAC-CNN: A Visual Analytics System for Comparative Studies of Deep Convolutional Neural Networks. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2326–2337, 2022. doi: 10.1109/TVCG.2022.3165347 2
- [65] R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, X. Han, M. Gao, C. Lin, and L. S. Davis. NISP: Pruning Networks Using Neuron Importance Score Propagation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9194–9203. CVF/IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00958 2
- [66] S. Yu, T. Chen, J. Shen, H. Yuan, J. Tan, S. Yang, J. Liu, and Z. Wang. Unified Visual Transformer Compression. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. doi: 10.48550/arXiv.2203.08243 2
- [67] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. A survey of visual analytics techniques for machine learning. *Computational Visual Media*, 7(1):3–36, 2021. doi: 10.1007/S41095-020-0191-7 2
- [68] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, et al. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Engineering Bulletin*, 41(4):39–45, 2018. doi: 10.1145/3399579.3399867 2
- [69] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, vol. 8689, pp. 818–833. Springer, New York, NY, 2014. doi: 10.1007/978-3-319-10590-1_53 9
- [70] M. Zhu and S. Gupta. To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. *arXiv*, 2017. doi: 10.48550/arXiv.1710.01878 2, 6