We sincerely appreciate the reviewers' insightful comments and constructive feedback.

Question Reviewer A: One of the design goals is to store the learned models. However, the design of Wild Turkey shows no significant differences compared with Bourbon.

Answer Reviewer A: In terms of storing the learned index model, two primary strategies exist: embedding the model directly within the SSTable file or maintaining separate model files corresponding to each SSTable. To ensure that model training and storage operations do not negatively impact the tail latency during compaction evaluations, we chose to store learned index models separately from their associated SSTables, as discussed in Section 4.3.

This design aligns closely with Bourbon(OSDI'20)'s approach, where model files are also managed independently. However, Wild Turkey distinguishes itself by employing multi-layer learned models, unlike Bourbon's single-layer models. Specifically, in Wild Turkey's LAC, the size of SSTables progressively increases at higher levels. Despite this growth in SSTable sizes, the corresponding model file sizes remain constant, due to the hierarchical structure of the multi-layer model: instead of expanding the model file size, the model complexity is managed through additional hierarchical layers.

Question Reviewer B: Comment on how your findings are generalized for different workloads. Explain why your model has a smaller size than Bourbon. Answer Reviewer B: Our findings apply to different workloads due to the LSM-tree's compaction process and leveled structure. Higher levels typically have more compact data distributions which is a direct result of compaction. Regardless of the dataset or workload, our findings remain universally applicable. Based on this, we recommend that when using learned index in LSM-trees, we should consider the data distribution across different levels and adopt level- or file-specific strategies.

Our model has a smaller size than Bourbon due to our dynamic error bound adjustment. Both our approach and Bourbon use Piecewise Linear Regression, where the algorithm segments the index based on a user-defined minimum error bound. In short, a smaller error bound results in more segments and more index models (Figure 6). We analyzed how, under the same error bound, different data characteristics affect segmentation. Since SSTables at different LSM-tree levels exhibit varying data distributions (Figure 7(a)), the number of segments differs significantly (Figure 7(b)). To balance lookup performance and model size, we employ Reinforcement Learning to adjust error bounds based on data characteristics.

The real-world datasets we used are the same as the ones used in the paper Benchmarking Learned Indexes (VLDB'20) and are commonly used for index performance evaluation. [27] states that the OSM dataset has highly random data distribution, while the FB dataset, aside from a few outliers, is largely linear and dense. We consider these two datasets as extremes, with Book and Wiki falling between them. We emphasize that in traditional indexing, Wisckey suffers from performance degradation when SSTable size increases, while a learned index behaves differently. Among the datasets we tested, OSM is the most sensitive to SSTable size and exhibits the most negative impact (Figure 5(c)). This is because OSM keys are highly disordered (Section 5.3), making it harder for a learned index to generalize, leading to more additional layers in the model and increased lookup costs. This dataset effectively demonstrates the limitations of learned indexes with difficult data distributions. Even with a larger SSTable size, the lookup complexity remains stable. As seen in Figure 5(a), FindFiles latency decreases while SearchIB+DB increases which neutralizes the negative impact to the overall read latency. In contrast, for FB (Figure 5(b)), which has a simpler key distribution, the impact is smaller because fewer segments are needed. We use 10M dataset for breakdown analysis following Bourbon's approach. In key-value separation, a 10M dataset allows the LSM-tree to build up to higher levels while making the read breakdown more observable.

Question1 Reviewer C: Why does read latency decrease as SSTable size increases? This is counterintuitive and should be explained. Answer1 Reviewer C: In traditional systems, increasing the SSTable size typically results in higher search latency due to rising binary search costs and larger index overheads, as demonstrated in our motivation (Figure 5). However, our proposed Wild Turkey addresses these issues with two key optimizations: Learning-Aware Compaction (LAC) and Wild-Learning. Specifically, LAC optimizes the SSTable sizes for each level, leveraging larger SSTables at higher levels to exploit learned index benefits for improved read performance, while maintaining smaller SSTables at lower levels to reduce write amplification. Wild-Learning employs Reinforcement Learning to dynamically determine an optimal error bound tailored specifically to the evolving data distribution within each SSTable. This adaptively reduces internal search latency by ensuring highly efficient lookups that dynamically reflect data characteristics.

As shown in Figure 16, the observed decrease in read latency with increasing SSTable size arises primarily from two factors. First, the time spent locating the relevant SSTable ("FindFiles") decreases by 36.4% compared to Bourbon. Second, Wild Turkey significantly enhances internal SSTable search efficiency through the Wild Learning mechanism, which dynamically adapts to the underlying data distribution.

Additionally, Wild Turkey enhances search efficiency beyond what is achieved by Bourbon. While Bourbon relies on a single-layer learned index model paired with binary search, Wild Turkey employs a multi-layer learned index. This hierarchical approach significantly reduces the internal search latency within SSTables, particularly beneficial for larger SSTables containing many segments. Consequently, Wild Turkey effectively offsets—and even reverses—the typical latency penalties associated with increasing SSTable sizes.

Question2 Reviewer C: Given that they are read-only, how are the real-world datasets in [27] evaluated on write workloads? Answer2 Reviewer C: To do our experiments, we obtain the datasets and load them into Wild Turkey. This approach is also utilized by other studies, such as Bourbon (OSDI'20) and TreeLine (VLDB'22), both using Amazon Reviews and OSM datasets in their evaluations.

Question3 Reviewer C: Would longer-running experiments with larger datasets provide more representative/informative results? Answer3 Reviewer C: Our chosen dataset size (64M keys) aligns with or exceeds dataset sizes commonly adopted in comparable studies such as TreeLine (VLDB'22), Bourbon (OSDI'20), and LeaderKV (ICDE'24), demonstrating its adequacy for producing representative and meaningful experimental results. Furthermore, our research explicitly focuses on application-level optimizations at the system architecture level, aiming to maximize the benefits derived from integrating learned indexes within LSM-based key-value stores. Hardware-specific factors, such as performance impacts arising from advanced storage devices like NVMe SSDs, fall beyond the scope of our current work. This intentional separation distinguishes our approach from research that specifically addresses hardware-level optimizations.