Can We Trust Strong Teachers? Auditing Knowledge Distillation for Binary Code Similarity Detection

Minh Khang Van
k.van@campus.lmu.de
LMU Munich
Munich, Germany

Yunru Wang
yunru.wang@ifi.lmu.de
LMU Munich
Munich, Germany

Johannes Kinder
johannes.kinder@lmu.de
LMU Munich
Munich, Germany

Abstract

Large binary models are costly to use for Binary Code Similarity Detection (BCSD) at scale, motivating their distillation into compact encoders. Existing BCSD distillation work focuses mainly on teacher reliability, while student-side factors remain underexplored. We conduct a controlled study on a large-scale cross-project benchmark, fixing the student architecture and task objective while comparing three distillation objectives with the retrieval-specialized CLAP and the best-performing objective with the generative decompiler Nova. Our results show that direct task training produces a strong student that nearly matches zero-shot CLAP, while distillation adds only marginal gains. We therefore study two student-side factors that shape how teacher supervision is received: its interaction with task training and the input visible to the student. Under the same loss coefficient, different distillation objectives produce markedly different gradient interactions with InfoNCE, yet these statistics do not yield a simple rule for loss balancing. Increasing the student context from 64 to 1,024 tokens consistently improves embedding distillation and narrows the gap to CLAP. These results show that reliable BCSD distillation depends not only on teacher quality, but also on the student's optimization and input constraints.

CCS Concepts

• **Security and privacy → Software reverse engineering**.

Keywords

binary code similarity detection, knowledge distillation, binary representation learning

ACM Reference Format:
Minh Khang Van, Yunru Wang, and Johannes Kinder. 2026. Can We Trust Strong Teachers? Auditing Knowledge Distillation for Binary Code Similarity Detection. In *Proceedings of the 1st International Workshop on Reliable and trustworthy Automated Software Engineering (RASE '26), October 12–16, 2026, Munich, Germany*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3820756.3844917

1 Introduction

Binary code similarity detection (BCSD) supports automated vulnerability localization, patch analysis, and code-clone detection when source code or reliable symbols are unavailable, making it an important component of software supply-chain security [2, 5, 6, 24]. This task is challenging because the same function can have very different binary representations across compilers, optimization levels, architectures, and obfuscation settings, while unrelated functions may still exhibit similar instruction and control-flow patterns [13]. Accordingly, learning-based methods have increasingly moved beyond hand-crafted features toward representations learned from binaries. Reliable similarity detection, however, still depends on capturing semantics that remain stable across different settings [17].

Obtaining such representations at scale creates a practical trade-off. Searching millions of functions requires compact encoders that can process and compare functions efficiently. Their limited capacity and short input windows, however, may prevent them from modeling semantic evidence distributed across long functions. Recent high-capacity models for binary analysis can capture such evidence from longer contexts [4, 9, 16, 18, 25], but are too expensive to apply to every function in a large retrieval corpus. This motivates using them as offline teachers for compact models.

Recent work has explored using knowledge distillation as a general add-on to existing specialized BCSD models. BinSKD [28] uses CLAP [21], a specialized binary representation model with strong retrieval performance, as the teacher. It keeps each student's original architecture, input processing, and task loss, and adds CLAP's pairwise similarity relations to four heterogeneous students: HermesSim [8], jTrans [22], SAFE [14], and BinaryAI [27]. BinSKD focuses on teacher-signal quality, using a query-level gate to retain supervision only when CLAP outperforms the student. This improves all four students, but the gains vary substantially. At a pool size of 8,192, HermesSim improves from 0.655 to 0.750 Recall@1 and slightly surpasses CLAP at 0.744, whereas BinaryAI rises only from approximately 0.12 to 0.23 and remains far behind. Without selection, direct relational distillation instead reduces HermesSim's MRR drastically from 0.716 to 0.069. These student-side differences in input representation, architecture, task objective, and original performance make the reported gains difficult to compare, while their individual effects remain unexplored in BinSKD. These findings raise a broader trustworthiness question: when the student and its training pipeline are held fixed, how much reliable value does teacher supervision add beyond direct task training, and what limits that value?

To answer this question, rather than refining teacher signals, we study how student-side factors shape their effectiveness. We conduct a controlled study on the large-scale cross-project BCSD benchmark [13], covering multiple compilers and optimization levels. We fix the student architecture to a compact BERT-style encoder, pretrain it with masked language modeling (MLM), and train it for retrieval with InfoNCE [20], following a common design in recent

RASE '26, Munich, Germany

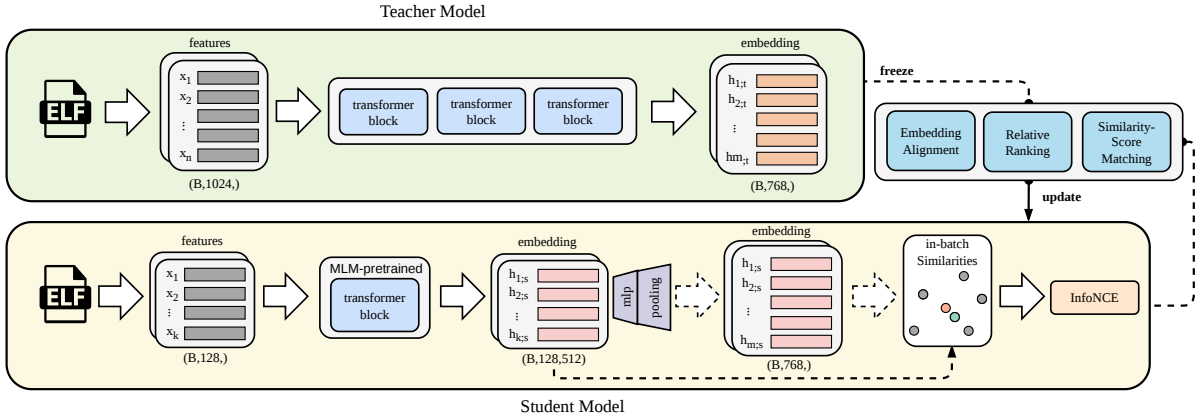

Figure 1: Overview of the controlled distillation framework. For a batch of binary functions, a frozen teacher supervises an MLM-pretrained compact student using InfoNCE and one of three distillation objectives.

BCSD research [10, 22]. We also measure the practical efficiency gain of this compact student for offline corpus encoding. With the student and task training fixed, we use CLAP [21], a retrieval-specialized model, as the primary teacher and evaluate all three distillation objectives. To examine teacher choice, we apply the best-performing CLAP distillation objective to Nova [9], a generative decompiler, under the same setup. We then study two student-side factors beyond BinSKD: how distillation interacts with task training in the student's parameter space, and how the student's input coverage affects what can be transferred from the teacher.

Our controlled study leads to three main findings:

- **Direct task training accounts for most student performance.** Pure distillation transfers useful BCSD knowledge but remains substantially weaker than direct InfoNCE training. Under joint training, distillation provides only small, metric-dependent gains, and differences in teacher performance are not consistently inherited by the student.
- **Distillation objectives interact differently with direct task training.** Under the same loss weight, embedding alignment, ranking, and similarity matching produce substantially different gradient directions and scales relative to InfoNCE. These results show that the same teacher can impose very different optimization pressure on the student's direct task objective depending on how its knowledge is transferred.
- **Student context strongly affects knowledge transfer.** Increasing the student context from 64 to 1,024 tokens consistently improves embedding distillation and reduces the nDCG@1024 gap to CLAP from 0.1143 to 0.0270. Thus, input coverage explains a substantial part of the observed teacher-student gap.

To the best of our knowledge, this is the first controlled empirical study of student-side factors in BCSD distillation. By fixing the student pipeline, we study how direct task training, loss interaction, and student input context shape knowledge transfer. Our results show that teacher quality alone is insufficient: effective distillation also depends on how teacher supervision interacts with student training and how much relevant input the student can observe. Our code and experimental artifacts are available at [1].

2 Knowledge Distillation

To systematically investigate how teacher selection and supervision signals influence knowledge transfer during model compression, we design a controlled distillation framework. Our framework isolates two factors that determine distillation effectiveness: the source of transferred knowledge (teacher model) and the form of supervision signal (distillation objective). As illustrated in Figure 1, we freeze the teacher parameters and optimize a compact student encoder using task-specific contrastive learning combined with one of three distillation objectives: **Embedding Alignment**, **Relative Ranking**, and **Similarity-Score Matching**. This design allows us to separate the effects of teacher choice and distillation objective on student performance.

2.1 Teacher Model

In our controlled setting, we use two teachers with different relationships to the downstream task. CLAP [21] is our primary teacher and Nova [9] serves as a complementary case. CLAP is a task-specific binary representation model trained directly for BCSD, and therefore represents a task-aligned source of supervision. Nova, in contrast, is a decoder-only model trained for binary decompilation rather than retrieval. We derive its function-level representation by mean-pooling the final hidden states. Nova serves as a contrasting teacher whose representations encode binary semantics through assembly-to-source generation but are less directly aligned with the BCSD objective. Using both teachers allows us to test whether our observations about student-side conditions depend on the teacher being specialized for the downstream task.

2.2 Student Model

To support scalable retrieval over large BCSD corpora, we distill teacher knowledge into a compact BERT-style student encoder [1] with lower computational cost and memory use. The student is

[1] https://github.com/withblues/BinAI

designed to isolate the effect of knowledge distillation by keeping the architecture relatively straightforward. BERT-style encoders are widely used for binary representation learning: jTrans [22] extends BERT with jump-aware modeling and a jump target prediction objective, while PalmTree [10] augments BERT-style pretraining to capture structural and data-flow relations between instructions. We initialize the student with masked language modeling (MLM) before BCSD-specific fine-tuning. By keeping the student architecture compact and standard, we can study teacher supervision without confounding it with additional architectural components.

Beyond reducing the number of model parameters, we restrict the student's input length to control the cost of encoding and indexing large function corpora. For downstream BCSD training and retrieval, we restrict the student input to 128 tokens, whereas the teachers process up to 1,024 tokens, reducing the student-visible context by 87.5%. We choose this relatively small downstream context window to represent a resource-constrained deployment setting rather than to match the capacity of the teachers. Consequently, during downstream training the student may observe only part of the code on which a teacher representation depends. We study the effect of this input coverage separately in RQ3 (Section 3.5).

2.3 Task-Aligned Student Training

To isolate the contribution of teacher supervision from task-aligned fine-tuning, we establish the student baseline trained exclusively with the BCSD objective InfoNCE, which directly shapes the embedding space for similarity retrieval by increasing similarity between positive function pairs while separating in-batch negative candidates.

For a batch of N query-positive pairs ($2N$ representations in total), the model computes pairwise cosine similarities across all function representations. For a query function i and its corresponding positive example $p(i)$, the InfoNCE objective is formulated as:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{2N}\sum_{i=1}^{2N}\log\frac{\exp(S^s_{i,p(i)}/\tau)}{\sum_{j\neq i}^{2N}\exp(S^s_{ij}/\tau)}. \tag{1}$$

where S^s_{ij} denotes the cosine similarity between student function embeddings, τ is the temperature parameter, and N represents the number of query-positive pairs in the batch. Positive pairs are constructed from binary functions compiled from the same source-level function within the same binary identity, while the remaining functions in the mini-batch serve as implicit negative candidates after filtering potential false negatives.

2.4 Knowledge Distillation Objectives

We compare three distillation objectives that are all applicable to retrieval tasks but impose different constraints on the student. Embedding alignment asks the student to reproduce the teacher representation, similarity matching preserves metric relationships between functions, and ranking distillation retains only their relative order. This range allows us to examine whether a compact student benefits more from reproducing the teacher's representation directly or from preserving less-restrictive relational information. A useful distillation target should therefore transfer information that matters for BCSD, that the compact student can reproduce, and that does not work against the InfoNCE training objective.

2.4.1 Embedding-Alignment Distillation. Embedding-alignment distillation aligns student representations with teacher outputs by minimizing the discrepancy between corresponding function embeddings [7]. For two vectors u, v, let $\text{sim}(u,v) = u^\top v/(\|u\|\|v\|)$ denote their cosine similarity. Since BCSD retrieval relies on angular similarity, we optimize the cosine distance between the teacher and student representations of the same binary function, $\mathcal{L}_{\text{emb}} = 1 - \text{sim}(z_t, z_s)$, where z_t and z_s denote the teacher and student embeddings of the same binary function. Because the teacher model is frozen, the loss acts solely as a directional regression target, directly encouraging the student embedding z_s to align with the fixed reference vector z_t without requiring negative sampling.

Teacher and student embeddings may have different dimensionalities. To enable alignment, we introduce a trainable projection head that maps the student representation into the teacher embedding space before computing the loss. The projection head is optimized jointly with the student encoder.

2.4.2 Similarity-based Distillation. Rather than matching individual embeddings, similarity-based distillation transfers the structure of the teacher embedding space [19]. This is particularly relevant for BCSD, where retrieval depends on functional similarity rather than exact vector coordinates. For a batch of N functions we form the pairwise cosine similarity matrix $S \in \mathbb{R}^{N\times N}$ with $S_{ij} = \text{sim}(z_i, z_j)$. Because both similarity-based objectives operate on these scalar similarities rather than on the embeddings themselves, they decouple the student architecture from the teacher and allow different embedding dimensionalities without an auxiliary projection head. We consider two objectives:

Relative Ranking. Ranking distillation preserves the ordering induced by the teacher without requiring the student to reproduce its embeddings or exact similarity values. For each query function i in a batch of size B, we take the k in-batch candidates the teacher scores highest, re-indexed so that $S^t_{i1} \geq \cdots \geq S^t_{ik}$, and form all $\binom{k}{2}$ pairs (j, m) with $j < m$, so that the teacher always prefers j over m. The student is trained to reproduce each of these preferences:

$$\mathcal{L}_{\text{rank}} = -\frac{1}{B\binom{k}{2}}\sum_{i=1}^{B}\sum_{j<m}\underbrace{\frac{S^t_{ij}-S^t_{im}}{\tau}}_{\text{teacher margin}}\log\sigma\left(\frac{S^s_{ij}-S^s_{im}}{\tau}\right), \tag{2}$$

where $\sigma(x) = (1+e^{-x})^{-1}$ is the logistic sigmoid and τ the distillation temperature. The loss penalizes disagreements with the teacher's ordering, while weighting each pair by the teacher margin. Since this margin is non-negative by construction, confidently separated pairs receive more weight than near-ties.

Similarity-score matching. Similarity matching transfers the continuous relational distance structure of the teacher's representation space. The student directly matches the pairwise cosine similarities produced by the teacher:

$$\mathcal{L}_{\text{sim}} = \frac{1}{N^2}\sum_{i,j}\left(S^s_{ij} - S^t_{ij}\right)^2 \tag{3}$$

where S^s and S^t denote the student and teacher pairwise similarity matrices, respectively. Unlike ranking distillation, it preserves the numerical differences between teacher similarity scores.

2.5 Joint Training

To evaluate whether teacher supervision provides complementary information beyond task-aligned contrastive learning, we jointly optimize the student using $\mathcal{L}_{\text{joint}} = \lambda_{\text{InfoNCE}}\mathcal{L}_{\text{InfoNCE}} + \lambda_{\text{distill}}\mathcal{L}_{\text{distill}}$, where $\mathcal{L}_{\text{distill}} \in \{\mathcal{L}_{\text{emb}}, \mathcal{L}_{\text{sim}}, \mathcal{L}_{\text{rank}}\}$ is the selected teacher supervision objective. In all experiments we set $\lambda_{\text{InfoNCE}} = \lambda_{\text{distill}} = 1$, allowing us to compare distillation objectives under the same joint-training setup without additional loss-weight tuning. This lets us measure whether teacher knowledge complements or interferes with task-aligned representation learning.

3 Evaluation

Our evaluation first quantifies the practical benefit of a compact student for large-scale BCSD, and then examines how effectively knowledge from pretrained binary models can be transferred to it. We study the roles of teacher supervision, distillation objectives, and student context length through the following research questions:

RQ1: To what extent does teacher supervision improve a compact student beyond direct BCSD training (§3.3)?
RQ2: How do distillation objectives differ in their interaction with task training (§3.4)?
RQ3: How does the student's context window affect knowledge transfer from the teacher (§3.5)?

We first describe the experimental setup and quantify the practical efficiency gains that motivate the use of a compact student, followed by the experiments and findings for each research question.

3.1 Experimental Setup

To ensure fair comparisons, all student models were implemented using PyTorch and HuggingFace Transformers and trained on a single NVIDIA H100 GPU with identical configurations.

Dataset. We utilize the Cisco Talos BCSD dataset [13], which contains binary functions compiled from multiple C and C++ projects under different compilation configurations. Although the original dataset contains multiple architectures, we focus on the x64 subset in this work. We follow the benchmark's cross-project split, keeping training, validation, and test projects disjoint to prevent project-level leakage and evaluate generalization to unseen projects. The resulting dataset contains approximately 3.60m training binary functions, 55.3k validation binary functions, and 514k test binary functions. CLAP's pretraining data are documented, but Nova's are not fully specified, so overlap with our evaluation projects cannot be ruled out.

Model Configuration and Training. For CLAP, we use the released preprocessing pipeline and tokenizer. Nova releases its tokenizer but not preprocessing code, so we reconstruct its input format from the provided processed examples. Nova readout representations are obtained by mean pooling the final hidden states.

The student uses disassembly text with direct jump targets rebased to instruction-position labels, as also done by CLAP and Nova, while other normalization details differ slightly across models. We train a WordPiece tokenizer with NFKC normalization and whitespace pre-tokenization, resulting in a vocabulary of 33,555 tokens. The vocabulary was constructed once from the complete x64 corpus and may therefore reflect lexical information from held-out projects, although vocabulary construction is unsupervised and uses neither retrieval labels nor function correspondences. We keep a separate student tokenizer to keep its input pipeline fixed across teachers and avoid coupling it to any teacher-specific tokenizer.

The student is a 6-layer BERT encoder with hidden size 512, intermediate size 2,048, and 8 attention heads (36.9M parameters). We pretrain it from scratch with MLM on functions from the downstream training projects for six epochs using dynamic masking ($p = 0.15$), sequences of up to 1,024 tokens, AdamW with learning rate 5×10^{-5}, and an effective batch size of 128. Validation projects are used for checkpoint selection, and all student parameter updates use only functions from the downstream training projects.

For downstream BCSD training and distillation, we use AdamW with learning rate 1×10^{-5}, linear warmup followed by linear decay, six epochs, and a batch size of 128. The maximum input length is 128 tokens by default. For InfoNCE, each batch contains 128 query-positive pairs, giving 256 representations. Each representation has one explicit positive and at most 254 in-batch negatives after excluding itself; same-identity and identical-token collisions are masked. Function representations are obtained by attention-mask-aware mean pooling over the final hidden states. For embedding alignment, a trainable linear layer projects the 512-dimensional student representation to the teacher dimension before L2 normalization.

Retrieval-Pool Construction. The retrieval corpus contains all 514,071 x64 test functions, with no per-query candidate sampling. Two functions are considered a match if they share the same tuple (`project`, `binary_name`, `function_name`), corresponding to different compiled instances of the same source-level function. Functions with at least one other matching instance are used as queries, with the query itself removed from the candidate pool; unmatched functions remain as negative candidates. All compiler families, versions, and optimization levels in the test split are retained.

Compared with BinSKD's sampled pools of up to 8,192 candidates with one positive target per query, our evaluation uses the full 514k-function corpus and allows multiple relevant instances per query, providing a more realistic large-scale retrieval setting.

Metrics. All embeddings are L2-normalized and candidates are ranked by cosine similarity. Because each query may have multiple ground-truth matches, we focus on Recall@k and nDCG@k at larger cutoffs to evaluate retrieval coverage and ranking quality. Because the candidate pool contains 514k functions, we use $k \in \{512, 1024\}$ for high-recall candidate generation; even these cutoffs correspond to only about 0.10% and 0.20% of the full pool. We additionally report MRR to characterize the rank of the first ground-truth match. Recall@1 is less informative in our multi-positive setting, where a correct top-ranked match contributes only a fraction of the query's total ground-truth matches.

Model Variants. All student variants start from the same MLM-pretrained BERT encoder, and both teachers remain frozen. We name them `[type]_[objective]_[T]`, where `[type]` is `distil` (distillation only) or `joint` (with InfoNCE), `[objective]` is `embd`, `sim`, or `rank` (§2.4), and `T` is the supervising teacher. `Student Baseline` denotes the task-only model trained with InfoNCE alone.

Table 1: Offline corpus-encoding efficiency on an H100 GPU. Time and throughput report the mean with standard deviation as a subscript; memory is the maximum observed peak.

Model	Corpus Time (s)	Throughput (functions/s)	Peak GPU Memory (MiB)
CLAP	$4224.36_{.87}$	$121.69_{.03}$	15,499
Student Baseline	$\mathbf{125.06}_{1.19}$	$\mathbf{4110.87}_{38.64}$	**1,083**

For CLAP, we evaluate all three objectives, while for Nova, we use the best-performing CLAP distillation objective.

3.2 Student Efficiency

Before evaluating knowledge transfer, we first verify the practical motivation for distilling into a compact student. Large-scale BCSD may require indexing repositories with millions of functions and repeatedly updating the index as new binaries or firmware arrive, making even offline corpus encoding computationally and memory intensive. We compare CLAP with the default 128-token student on the same 514,071-function test corpus using one NVIDIA H100 GPU and a batch size of 64; each measurement is repeated five times.

As shown in Table 1, `Student Baseline` reduces corpus encoding time from 4,224.36 s to 125.06 s, corresponding to a 33.78× increase in throughput, while reducing peak GPU memory from 15,499 MiB to 1,083 MiB (14.31×). These savings motivate compact encoders for large-scale BCSD. Distillation provides a natural way to transfer prior knowledge from large pretrained models to such a student; the following RQs examine how effectively this knowledge can be transferred.

3.3 RQ1: How Much Does Teacher Supervision Contribute Beyond Direct BCSD Training?

In RQ1, we ask how much teacher supervision adds beyond direct BCSD training. We evaluate all three distillation objectives with CLAP under joint training, test embedding and similarity distillation without InfoNCE, and apply the best CLAP objective to Nova to assess the effect of teacher strength.

Effect of training objectives. We separate the contribution of teacher supervision from that of direct BCSD training, comparing the same student under MLM pretraining only, pure distillation, InfoNCE only, and joint training.

Table 2 shows that pure distillation transfers useful BCSD information from CLAP. `distil_sim_CLAP` raises nDCG@1024 from 0.4790 at the `Student Baseline` checkpoint to 0.6310, and `distil_embd_CLAP` also improves over this checkpoint. However, the randomly initialized InfoNCE model `Student Baseline (no MLM)` already reaches 0.6941, outperforming both pure-distillation variants. MLM pretraining in `Student Baseline` provides only a smaller additional gain. Direct task training is therefore the main source of student retrieval performance.

Once InfoNCE is present, the effect of teacher supervision depends on the distillation target. `joint_sim_CLAP` provides only a small improvement in nDCG@1024, from 0.7010 to 0.7060. `joint_rank_CLAP` remains close to the baseline, while `joint_embd_CLAP` degrades performance. Thus, teacher supervision provides a useful signal in isolation, but adds little beyond direct task training and can be harmful depending on the distillation objective.

Table 2: Ablation of task-aligned training and distillation with CLAP as the teacher. Joint variants combine InfoNCE with a distillation objective; pure-distillation variants exclude InfoNCE. All variants are single training runs.

Model	nDCG @512	Recall @512	nDCG @1024	Recall @1024
Baselines				
Student Baseline (no InfoNCE)	0.4674	0.5205	0.4790	0.5544
Student Baseline (no MLM)	0.6826	0.8388	0.6941	0.8731
Student Baseline	0.6895	0.8562	0.7010	0.8905
Pure Distillation				
distil_embd_CLAP	0.5915	0.7057	0.6043	0.7437
distil_sim_CLAP	0.6173	0.7516	0.6310	0.7924
Joint Training				
joint_embd_CLAP	0.6745	0.8273	0.6866	0.8631
joint_rank_CLAP	0.6879	0.8499	0.6996	0.8850
joint_sim_CLAP	**0.6949**	**0.8608**	**0.7060**	**0.8936**

Table 3: Zero-shot teachers and jointly trained students (RQ1). Student results report the mean with population standard deviation as subscript over seeds 42, 7, and 99. Teachers are evaluated once zero-shot.

Model	**MRR**	**nDCG @512**	**R @512**	**nDCG @1024**	**R @1024**
Teacher (zero-shot)					
CLAP	0.6416	**0.7104**	**0.8667**	**0.7186**	**0.8917**
Nova	**0.6686**	0.4821	0.5232	0.4946	0.5590
Task-aligned student					
Student Baseline	$0.6544_{.0295}$	$0.6906_{.0008}$	$0.8574_{.0011}$	$0.7021_{.0007}$	$0.8914_{.0009}$
joint_sim_CLAP	$0.6298_{.0314}$	$0.6932_{.0012}$	$\mathbf{0.8597}_{.0008}$	$0.7044_{.0012}$	$\mathbf{0.8929}_{.0005}$
joint_sim_Nova	$\mathbf{0.6619}_{.0141}$	$\mathbf{0.6937}_{.0007}$	$0.8538_{.0008}$	$\mathbf{0.7047}_{.0006}$	$0.8867_{.0012}$

Effect of teacher choice. As shown in Table 3, CLAP and Nova exhibit different zero-shot retrieval profiles. CLAP performs substantially better at larger cutoffs, reaching 0.7186 nDCG@1024 and 0.8917 Recall@1024, compared with 0.4946 and 0.5590 for Nova. In contrast, Nova achieves higher MRR (0.6686 versus 0.6416), indicating better top-ranked retrieval, while CLAP better recovers the broader set of matching instances. The two teachers therefore have different strengths, and their relative quality depends on the retrieval metric.

Despite their different zero-shot profiles, the jointly trained students perform similarly. At nDCG@1024, CLAP exceeds Nova by 0.2240 as a teacher (0.7186 vs. 0.4946), yet after distillation `joint_sim_CLAP` and `joint_sim_Nova` reach 0.7044 and 0.7047, respectively, both only slightly above the `Student Baseline` of 0.7021. Their 0.0003 difference is smaller than the observed seed variation. More broadly, `joint_sim_Nova` is numerically best in MRR and nDCG, while `joint_sim_CLAP` is best in Recall. Thus, teacher strength is not consistently inherited by the student, and distillation adds only small gains beyond direct InfoNCE training.

Answer to RQ1. Pure distillation transfers useful signal but remains weaker than direct InfoNCE training. Under joint training, teacher strengths are not consistently inherited, with CLAP students

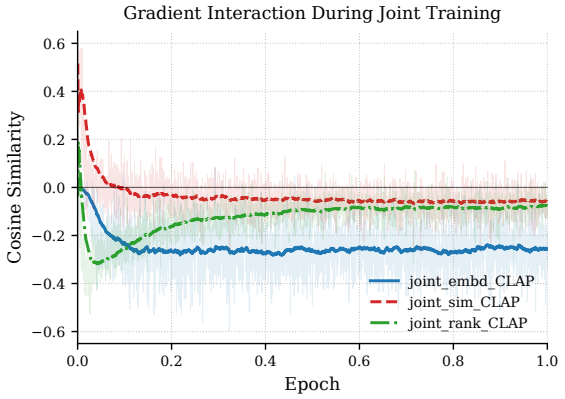

Figure 2: Gradient cosine similarity between InfoNCE and distillation objectives during the first epoch.

Table 4: Gradient interaction during Epoch 1 and retrieval performance after Epochs 1 and 6. c_t denotes gradient cosine similarity and $r_t = \|g_{\text{KD}}\|/\|g_{\text{task}}\|$.

Model	Mean c_t	Mean r_t	nDCG@1024 Epoch 1	nDCG@1024 Epoch 6
joint_sim_CLAP	−0.0403	0.0834	0.6964	0.7060
joint_rank_CLAP	−0.1226	1.9823	0.6915	0.6996
joint_embd_CLAP	−0.2514	0.4720	0.6886	0.6866

favoring Recall and Nova students favoring MRR and nDCG. Overall, distillation provides only small, metric-specific gains beyond direct task training.

3.4 RQ2: How Do Distillation Objectives Interact with Joint Training?

§3.3 shows that direct InfoNCE training accounts for most of the student's performance, while the joint distillation objectives produce different outcomes. We therefore examine how each distillation objective interacts with InfoNCE under the same fixed loss coefficient using gradient analysis.

Gradient Analysis. At each training step t, we compute gradients of the InfoNCE and distillation losses with respect to the final student encoder layer, $g_{\text{task}} = \nabla_\theta \mathcal{L}_{\text{InfoNCE}}$ and $g_{\text{KD}} = \nabla_\theta \mathcal{L}_{\text{KD}}$. We measure their cosine similarity $c_t = g_{\text{task}}^\top g_{\text{KD}} / (\|g_{\text{task}}\| \|g_{\text{KD}}\|)$ and norm ratio $r_t = \|g_{\text{KD}}\|/\|g_{\text{task}}\|$. Here, c_t captures directional alignment, with positive values indicating agreement, negative values indicating conflict, and values near zero indicating near-orthogonality; r_t measures relative gradient magnitude.

We record both statistics during the first epoch and evaluate retrieval after Epochs 1 and 6 to relate early gradient interaction to training outcomes. Figure 2 shows c_t over training, and Table 4 summarizes the gradient statistics and nDCG@1024.

Results. The three distillation objectives produce substantially different gradient interactions despite using the same loss coefficient. `joint_sim_CLAP` is only weakly opposed to InfoNCE on average ($c_t = -0.0403$) and has a small relative gradient ($r_t = 0.0834$). `joint_rank_CLAP` shows stronger directional conflict ($c_t = -0.1226$) and the largest relative magnitude ($r_t = 1.9823$). `joint_embd_CLAP` is the most opposed to InfoNCE ($c_t = -0.2514$), although its relative magnitude is smaller ($r_t = 0.4720$). Thus, using the same coefficient does not give the three distillation losses the same effective influence during optimization.

The retrieval results initially suggest a simple pattern: stronger directional conflict with InfoNCE corresponds to lower nDCG@1024. At Epoch 1, `joint_sim_CLAP`, `joint_rank_CLAP`, and `joint_embd_CLAP` achieve 0.6964, 0.6915, and 0.6886, respectively, following the same ordering as their gradient cosine similarity. The same ordering remains after six epochs at 0.7060, 0.6996, and 0.6866. Notably, `joint_embd_CLAP` slightly degrades after the first epoch, whereas `joint_sim_CLAP` and `joint_rank_CLAP` continue to improve.

However, cosine similarity alone does not capture the full gradient interaction. `joint_rank_CLAP` has a much larger gradient than `joint_embd_CLAP`, yet it still achieves better retrieval. Thus, the apparent relation between gradient conflict and retrieval quality is not sufficient to explain the differences between objectives. This observation is important for gradient-based loss balancing. Although gradient signals are commonly used to adjust the weights of multiple training objectives [11], our results show that simple measures of gradient alignment or magnitude do not provide a complete rule for balancing task and distillation losses. Gradient-aware distillation therefore requires careful design rather than mechanically favoring objectives that conflict less with the student's main task loss.

Answer to RQ2. The same loss coefficient produces substantially different gradient interactions across distillation objectives, and simple gradient statistics alone do not provide a reliable rule for loss balancing.

3.5 RQ3: How Does Student Context Length Affect Distillation?

`CLAP` processes up to 1,024 tokens, whereas the student uses 128 by default. Distillation therefore compresses not only model capacity but also the amount of input available to the student. To isolate the latter effect, we train the distillation-only `distil_embd_CLAP` student with maximum sequence lengths of 64, 128, 256, 512, and 1,024 tokens, while keeping the student architecture, teacher, distillation objective, training data, and test corpus fixed. We use `distil_embd_CLAP` because it is the weakest pure-distillation objective at the default 128-token setting, providing a stringent test of whether additional student context can recover lost transfer.

As shown in Table 5, increasing context length monotonically improves broad retrieval quality. Relative to the default 128-token setting, the 1,024-token student gains 0.0873 nDCG@1024 and 0.1094 Recall@1024. The nDCG@1024 gap to `CLAP` shrinks from 0.1143 at 128 tokens to 0.0270 at 1,024 tokens. The same monotonic trend is observed across the other reported metrics.

Answer to RQ3. With matched context length, the compact student approaches `CLAP` using distillation alone, without direct BCSD supervision. This suggests that, when task labels are unavailable, preserving student context is important for effective knowledge transfer. Context compression explains a substantial part of the

Table 5: Evaluation of retrieval performance across model variants, where suffix numbers denote context size $L \in \{64, 128, 256, 512, 1024\}$. Metrics are reported for nDCG@K and Recall@K at cutoffs $K \in \{512, 1024\}$. All distilled models are based on CLAP as teacher.

Model	nDCG @512	Recall @512	nDCG @1024	Recall @1024
CLAP	**0.7104**	**0.8667**	**0.7186**	**0.8917**
distil_embd_64	0.5012	0.5874	0.5145	0.6266
distil_embd_128	0.5915	0.7057	0.6043	0.7437
distil_embd_256	0.6424	0.7694	0.6535	0.8029
distil_embd_512	0.6709	0.8060	0.6815	0.8381
distil_embd_1024	0.6814	0.8219	0.6916	0.8531

teacher-student gap, while the remaining difference may reflect model capacity and other factors not isolated here.

4 Limitations and Future Work

Student Capacity. We evaluate a single student architecture and parameter budget. The observed benefit of distillation, as well as the effect of context length, may therefore differ for smaller or larger students. Future work should vary student capacity while controlling context length to separate these two forms of compression.

Teacher and Distillation Scope. We evaluate all three distillation objectives only with CLAP, while Nova is tested only with similarity matching under joint training. The relative behavior of different distillation objectives may therefore differ across teachers. Future work should evaluate broader teacher-objective combinations, compare different pooling or representation-extraction strategies for teachers, and study additional distillation targets, including soft-target KL matching over temperature-scaled teacher similarities.

Pretraining-Data Overlap. Our downstream training, validation, and test projects are disjoint. CLAP's pretraining corpus is based on public datasets with documented sources. Nova, however, is initialized from DeepSeek-Coder, pretrained on assembly from The Stack and AnghaBench, and later fine-tuned on BinaryCorp-3M. Since a complete project list is not available for these pretraining data, overlap with our validation or test projects cannot be excluded. This could affect both Nova's zero-shot results and the supervision used for distillation.

Experimental Scope. The gradient analysis covers only the first epoch, so the observed interactions may not persist throughout training. Most ablation and diagnostic experiments use a single seed, making small differences between configurations uncertain. MRR is not reported for every experiment, which limits comparisons of first-match ranking behavior across ablations. Finally, results are aggregated across compiler families and optimization levels and may therefore conceal configuration-specific effects.

5 Related Work

Learning-based BCSD. Learning-based BCSD has largely been developed around task-specific models. Early sequence and graph models, including Asm2Vec [3], SAFE [14], and Gemini [26], replace hand-crafted similarity rules with learned functions. Later systems such as PalmTree [10], jTrans [22], and HermesSim [8] improve these representations through assembly pretraining or data flow and control flow structure. Their efficiency makes them suitable for large candidate pools, but their performance remains tied to the chosen binary representation and preprocessing pipeline. Our student follows this established line: a compact Transformer encoder pretrained with MLM and fine-tuned for retrieval with InfoNCE.

Recent BCSD work increasingly relies on higher-capacity models. CLAP [21] aligns assembly with natural language, while Nova [9] trains a generative binary model through decompilation; other approaches translate binaries into pseudocode or summaries before applying code LLMs to similarity search [12, 23]. Their richer semantics and longer contexts come at high retrieval cost, motivating distillation into compact BCSD encoders.

Knowledge Distillation for BCSD. Knowledge distillation transfers information from a higher-capacity teacher to a smaller student. For representation models, it can align teacher and student embeddings or match relations among examples instead of embedding coordinates [15, 19]. Relational matching better accommodates different teacher-student architectures, but depends on useful teacher similarities or rankings. BinSKD is the closest work to ours [28]. It uses CLAP to supervise four existing binary function-matching models through pairwise similarity relations. Because CLAP is not more reliable than the student on every query, BinSKD applies distillation selectively and focuses training on teacher-student ranking discrepancies. Its results show both sides of this approach: selective transfer improves the specialized students, whereas direct relational distillation can severely reduce performance. Our work does not propose another filtering or weighting mechanism. Instead, we study distillation from the student's perspective. Using a fixed compact student, we compare different forms of teacher supervision, analyze how distillation gradients interact with task training, and examine how student input coverage constrains transfer. These analyses show that effective distillation depends not only on teacher reliability, but also on how teacher supervision interacts with the student's task optimization and the information available to the student.

6 Conclusion

We presented a controlled study of knowledge distillation for BCSD, examining how teacher choice, distillation objective, and student context affect transfer to a compact encoder. The student achieves 33.78× higher corpus-encoding throughput and 14.31× lower peak GPU memory than CLAP. Direct InfoNCE training accounts for most student performance, while distillation adds only limited benefit and teacher strengths are not consistently inherited by their students. To understand this limited transfer, we study two student-side factors: how teacher supervision interacts with student's task optimization and how much input context is available to the student. Different distillation objectives produce markedly different gradient interactions with InfoNCE under the same loss weight, while increasing student context consistently improves embedding distillation and narrows the gap to CLAP. Overall, effective distillation depends not only on teacher quality, but also on how supervision interacts with student training and how much relevant input the student can observe.

Acknowledgments

ChatGPT was used only for grammar correction and refinement of wording. This work was funded in part by the Deutsche Forschungsgemeinschaft (DFG), 378803395 (ConVeY).

Data Availability Statement

The dataset used in this study is the publicly available Cisco Talos BCSD benchmark [13]. Our source code, including preprocessing scripts, model configurations, and evaluation scripts, is publicly available at https://github.com/withblues/BinAI. Model weights and additional experimental data are archived on Zenodo at https://doi.org/10.5281/zenodo.22249637. The CLAP [21] and Nova [9] teacher models were obtained from their respective public releases.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. doi:10.18653/V1/N19-1423

[2] Steven H. H. Ding, Benjamin C. M. Fung, and Philippe Charland. 2016. Kam1n0: MapReduce-based Assembly Clone Search for Reverse Engineering. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 461–470. doi:10.1145/2939672.2939719

[3] Steven H. H. Ding, Benjamin C. M. Fung, and Philippe Charland. 2019. Asm2Vec: Boosting Static Representation Robustness for Binary Clone Search against Code Obfuscation and Compiler Optimization. In *IEEE Symp. Security and Privacy (SP)*. IEEE, 472–489. doi:10.1109/SP.2019.00003

[4] Luke Dramko, Claire Le Goues, and Edward J. Schwartz. 2026. Idioms: A Simple and Effective Framework for Turbo-Charging Local Neural Decompilation with Well-Defined Types. In *Network and Distributed System Security Symposium (NDSS)*. The Internet Society. doi:10.14722/ndss.2026.240795

[5] Sebastian Eschweiler, Khaled Yakdan, and Elmar Gerhards-Padilla. 2016. discovRE: Efficient Cross-Architecture Identification of Bugs in Binary Code. In *Network and Distributed System Security Symp. (NDSS)*. The Internet Society. doi:10.14722/ndss.2016.23185

[6] Qian Feng, Rundong Zhou, Chengcheng Xu, Yao Cheng, Brian Testa, and Heng Yin. 2016. Scalable Graph-based Bug Search for Firmware Images. In *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 480–491. doi:10.1145/2976749.2978370

[7] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge Distillation: A Survey. *Int. J. Comput. Vis.* 129, 6 (2021), 1789–1819. doi:10.1007/S11263-021-01453-Z

[8] Haojie He, Xingwei Lin, Ziang Weng, Ruijie Zhao, Shuitao Gan, Libo Chen, Yuede Ji, Jiashui Wang, and Zhi Xue. 2024. Code is not Natural Language: Unlock the Power of Semantics-Oriented Graph Representation for Binary Code Similarity Detection. In *Security Symp. (USENIX)*, Davide Balzarotti and Wenyuan Xu (Eds.). USENIX Association.

[9] Nan Jiang, Chengxiao Wang, Kevin Liu, Xiangzhe Xu, Lin Tan, Xiangyu Zhang, and Petr Babkin. 2025. Nova: Generative Language Models for Assembly Code with Hierarchical Attention and Contrastive Learning. In *Int. Conf. Learning Representations (ICLR)*. OpenReview.net.

[10] Xuezixiang Li, Yu Qu, and Heng Yin. 2021. PalmTree: Learning an Assembly Language Model for Instruction Embedding. In *ACM SIGSAC Conf. Computer and Communications Security (CCS)*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, 3236–3251. doi:10.1145/3460120.3484587

[11] Xingyu Lin, Harjatin Singh Baweja, George Kantor, and David Held. 2019. Adaptive Auxiliary Task Weighting for Reinforcement Learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 4773–4784.

[12] Zhijie Liu, Qiyi Tang, Sen Nie, Shi Wu, Liang Feng Zhang, and Yutian Tang. 2025. KEENHash: Hashing Programs into Function-Aware Embeddings for Large-Scale Binary Code Similarity Analysis. *Proc. ACM Softw. Eng.* 2, ISSTA (2025), 801–824. doi:10.1145/3728911

[13] Andrea Marcelli, Mariano Graziano, Xabier Ugarte-Pedrero, Yanick Fratantonio, Mohamad Mansouri, and Davide Balzarotti. 2022. How Machine Learning Is Solving the Binary Function Similarity Problem. In *Security Symp. (USENIX)*, Kevin R. B. Butler and Kurt Thomas (Eds.). USENIX Association, 2099–2116.

[14] Luca Massarelli, Giuseppe Antonio Di Luna, Fabio Petroni, Roberto Baldoni, and Leonardo Querzoni. 2019. SAFE: Self-Attentive Function Embeddings for Binary Similarity. In *Proc. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA) (Lecture Notes in Computer Science, Vol. 11543)*, Roberto Perdisci, Clémentine Maurice, Giorgio Giacinto, and Magnus Almgren (Eds.). Springer, 309–329. doi:10.1007/978-3-030-22038-9_15

[15] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. 2019. Relational Knowledge Distillation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 3967–3976. doi:10.1109/CVPR.2019.00409

[16] Xiuwei Shang, Shaoyin Cheng, Guoqiang Chen, Yanming Zhang, Li Hu, Xiao Yu, Gangyang Li, Weiming Zhang, and Nenghai Yu. 2024. How Far Have We Gone in Binary Code Understanding Using Large Language Models. In *IEEE Int. Conf. Software Maintenance and Evolution (ICSME)*. IEEE, 1–12. doi:10.1109/ICSME58944.2024.00012

[17] Jingyi Shi, Yufeng Chen, Yang Xiao, Yuekang Li, Zhengzi Xu, Sihao Qiu, Chi Zhang, Keyu Qi, Yeting Li, Xingchu Chen, Yanyan Zou, Yang Liu, and Wei Huo. 2025. A Large Scale Study of AI-based Binary Function Similarity Detection Techniques for Security Researchers and Practitioners. In *Proc. IEEE/ACM Int. Conf. Automated Software Engineering (ASE)*. 1070–1082. doi:10.1109/ASE63991.2025.00093

[18] Hanzhuo Tan, Qi Luo, Jing Li, and Yuqun Zhang. 2024. LLM4Decompile: Decompiling Binary Code with Large Language Models. In *Proc. of the 2024 Conf. Empirical Methods in Natural Language Processing, EMNLP*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 3473–3487. doi:10.18653/V1/2024.EMNLP-MAIN.203

[19] Frederick Tung and Greg Mori. 2019. Similarity-Preserving Knowledge Distillation. In *IEEE Int. Conf. Computer Vision (ICCV)*. IEEE, 1365–1374. doi:10.1109/ICCV.2019.00145

[20] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018). doi:10.48550/arXiv.1807.03748

[21] Hao Wang, Zeyu Gao, Chao Zhang, Zihan Sha, Mingyang Sun, Yuchen Zhou, Wenyu Zhu, Wenju Sun, Han Qiu, and Xi Xiao. 2024. CLAP: Learning Transferable Binary Code Representations with Natural Language Supervision. In *Proc. ACM SIGSOFT Int. Symp. Software Testing and Analysis (ISSTA)*, Maria Christakis and Michael Pradel (Eds.). ACM, 503–515. doi:10.1145/3650212.3652145

[22] Hao Wang, Wenjie Qu, Gilad Katz, Wenyu Zhu, Zeyu Gao, Han Qiu, Jianwei Zhuge, and Chao Zhang. 2022. jTrans: jump-aware transformer for binary code similarity detection. In *ACM SIGSOFT Int Symp. Software Testing and Analysis (ISSTA)*, Sukyoung Ryu and Yannis Smaragdakis (Eds.). ACM, 1–13. doi:10.1145/3533767.3534367

[23] Zixiang Xian, Chenhui Cui, Rubing Huang, Chunrong Fang, and Zhenyu Chen. 2024. zsLLMCode: An Effective Approach for Functional Code Embedding via LLM with Zero-Shot Learning. *CoRR* abs/2409.14644 (2024). doi:10.48550/ARXIV.2409.14644

[24] Yang Xiao, Bihuan Chen, Chendong Yu, Zhengzi Xu, Zimu Yuan, Feng Li, Binghong Liu, Yang Liu, Wei Huo, Wei Zou, and Wenchang Shi. 2020. MVP: Detecting Vulnerabilities using Patch-Enhanced Vulnerability Signatures. In *Security Symp. (USENIX)*, Srdjan Capkun and Franziska Roesner (Eds.). USENIX Association, 1165–1182.

[25] Danning Xie, Zhuo Zhang, Nan Jiang, Xiangzhe Xu, Lin Tan, and Xiangyu Zhang. 2024. ReSym: Harnessing LLMs to Recover Variable and Data Structure Symbols from Stripped Binaries. In *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie (Eds.). ACM, 4554–4568. doi:10.1145/3658644.3670340

[26] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. 2017. Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection. In *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 363–376. doi:10.1145/3133956.3134018

[27] Zeping Yu, Rui Cao, Qiyi Tang, Sen Nie, Junzhou Huang, and Shi Wu. 2020. Order Matters: Semantic-Aware Neural Networks for Binary Code Similarity Detection. In *Proc. AAAI Conf. Artificial Intelligence (AAAI)*. AAAI Press, 1145–1152. doi:10.1609/AAAI.V34I01.5466

[28] Shize Zhou, Peiyu Liu, Lirong Fu, Tong Ye, and Wenhai Wang. 2026. Selective Knowledge Distillation: Fusing LLM Semantic Strengths with DNN Efficiency for Binary Code Similarity Detection. In *Proc. Association for Computational Linguistics (ACL)*, Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (Eds.). Association for Computational Linguistics, 26002–26014. doi:10.18653/V1/2026.ACL-LONG.1193