AV-NAS: Audio-Visual Multi-Level Semantic Neural Architecture Search for Video Hashing

Yong Chen
School of Computer Science, BUPT
Beijing, China
alphawolf.chen@gmail.com

Yuxiang Zhou
School of Computer Science, BUPT
Beijing, China
yuxiang.zhou@bupt.edu.cn

Hailiang Dong
School of Computer Science, BUPT
Beijing, China
hailiang.dong@bupt.edu.cn

Rui Liu
School of Computer Science and Engineering, Beihang University
Beijing, China
lr@buaa.edu.cn

Zhouchen Lin*
School of Intelligence Science and Technology, Peking University
Beijing, China
zlin@pku.edu.cn

Dell Zhang*
Institute of Artificial Intelligence, China Telecom
Shanghai, China
dell.z@ieee.org

Abstract

Existing video hashing techniques for large-scale video retrieval often overlook inherent audio signals, which can potentially compromise retrieval performance. Incorporating both visual and audio signals, however, complicates neural architecture design, rendering the manual crafting of joint audio-visual neural network models challenging. To address this issue, we propose AV-NAS, a method that leverages data-driven Neural Architecture Search (NAS) within a tailored audio-visual network space to automatically discover the optimal video hashing network. Our approach offers: (1) a versatile multi-level semantic architecture based on audio-visual signals, defining a mixed search space encompassing diverse network modules such as MLP, CNN, Transformer, and Mamba, as well as operations like Add, Hadamard, SiLU, LayerNorm, and Skip; (2) a differentiable relaxation of the combinatorial search problem, converting it into a unified differentiable optimization problem which we tackle through our "coarse search-pruning-finetuning" strategy. Our experiments on large-scale video datasets show that AV-NAS can discover architectures distinct from expert designs and lead to substantial performance improvements over current state-of-the-art methods including the recently emerged AVHash.

CCS Concepts

• **Information systems → Combination, fusion and federated search**; **Top-k retrieval in databases**; **Video search**.

Keywords

Video Retrieval, Learning to Hash, Neural Architecture Search.

*Zhouchen Lin and Dell Zhang are the corresponding authors. This work is supported by National Key R&D Program of China (2022ZD0160300) and the NSF China (No. 62276004, 62372054). Besides, Zhouchen Lin is now also working at State Key Lab of General AI & Institute for AI, Peking University, and Pazhou Lab, Guangzhou, China.

SIGIR '25, July 13–18, 2025, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3729899

ACM Reference Format:
Yong Chen, Yuxiang Zhou, Hailiang Dong, Rui Liu, Zhouchen Lin, and Dell Zhang. 2025. AV-NAS: Audio-Visual Multi-Level Semantic Neural Architecture Search for Video Hashing. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3729899

1 Introduction

Video, a powerful medium combining visuals and audios, creates immersive experiences and plays an increasingly vital role beyond text and images. Consider a scenario with 1 billion videos, each represented by a 1,000-dimension real-valued vector. The storage requirement would reach 8TB (1,000 dims×8 bytes/dim×1B). Additionally, semantic computations at this scale become highly inefficient due to floating-point operations. Video hashing—encoding videos into compact binary vectors—reduces storage to just 8GB (64 bits/vector×1B), easily fitting in a standard 16GB memory system. Binary operations enable hardware-accelerated XOR comparisons, allowing sublinear-time pairwise searches. By using binary codes as video fingerprints, retrieval achieves near O(1) complexity, enabling ultra-fast similarity searches across massive databases.

Existing video hashing methods, spanning from unsupervised (e.g., BTH [27], DKPH [25], ConMH [47]) to supervised (e.g., SRH [18], DSVH [3], AVH [46]), exhibit two notable limitations: firstly, they largely neglect audio signals beyond spatio-temporal keyframes (with AVHash [59] being a very recent exception) in video semantic modeling; secondly, hash functions often rely on expert-designed neural networks (such as MLP, CNN, LSTM, Transformer), based on varying interpretations of video keyframes. However, we posit that: (1) both visual and audio signals are indispensable for comprehending video semantics; (2) expert-crafted networks may not yield optimal performance, and thus, Neural Architecture Search (NAS) methods, like DARTS and other approaches [29, 36, 37], should be employed for data-driven discovery of optimal networks in video hashing. These insights underpin our research endeavor.

For the inherent visual & audio signals in videos, we propose a multi-level semantic network structure, depicted in Fig. 1, specifically tailored to process spatio-temporal keyframes and temporal sound wave encodings. This structure encompasses fusion, semantic transformation, and final hash encoding stages, culminating in

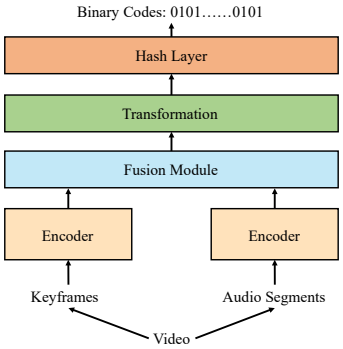


Figure 1: The abstract framework of AV-NAS.

compact binary codes. Each constituent module within our framework undergoes rigorous analysis and design, constructing a comprehensive combinatorial search space (Fig. 2) populated by diverse network components. Leveraging discrete-continuous mixed network architecture search, we relax discrete variables into continuous ones, employing a unified coarse search strategy to prune unnecessary components and identify optimal network structures. These are then fine-tuned on the training dataset. Dubbed AV-NAS, our method stands out on two public video datasets, surpassing state-of-the-art video hashing techniques. Notably, our search strategy consistently identifies optimal network structures across both datasets, revealing both expert-designed Transformer-based patterns and a unique "FFN+CNN" combination. This underscores the robustness and versatility of AV-NAS in video semantic retrieval.

To summarize, our work's primary contributions are as follows:

- Our research first trailblazes the integration of NAS to forge an innovative audio-visual multi-level semantic architecture, skillfully amalgamating MLPs, CNNs, Transformers, and the cutting-edge Mamba structure for video hashing.
- We seamlessly blend the Transformer with the Mamba framework, creating a versatile and unified paradigm. Additionally, we introduce a groundbreaking differentiable "coarse search-pruning-finetuning" strategy, enabling automated, efficient, and robust discovery of optimal network architectures.
- Extensive experiments, including ablation studies, highlight AV-NAS's superior performance in video retrieval, outshining SOTA methods. Its consistent results across datasets contrast sharply with classic NAS algorithms. Notably, AV-NAS beats expert-designed Transformers and Mamba architectures, significantly boosting mAP scores, paving the way for transformative advancements in large-scale video data management and mining.

2 Related Work

2.1 Video Hashing

Video hashing [35, 38, 50, 51, 55] aims to produce semantic binary vectors for videos, drastically cutting storage needs and boosting retrieval speeds in information retrieval contexts.

Early methods relied on pre-trained CNNs to extract features from video keyframes, averaged them for overall representations, and then used image hashing models (e.g., LSH [10], ITQ [16], SDH [34], COSDISH [24], SCDH [8], CSQ [56], LTH [7], and MDSH [44]) to create binary vectors as hash codes. Deviating from this, DSVH [3] introduced the use of a 3DCNN for end-to-end spatial-temporal feature extraction and binary code generation. MLP-based methods like DVH [28], MCMSH [20], and EUVH [12] employed various techniques for comprehensive video representations. LSTM-based approaches, such as SRH [18], AVH [46], SSTH [58], and NPH [26], modeled video structures and captured spatio-temporal content. Transformer-based models, including ConMH [47], BTH [27], and CHAIN [49], utilized encoder-decoder architectures or contrastive learning frameworks for self-supervised binary code learning, with CHAIN also incorporating collaborative tasks for enhanced spatio-temporal modeling.

Remarks. *Current video hashing techniques primarily rely on keyframe sequences for semantics, largely ignoring audio signals (except the very recent AVHash [59]), and employ a variety of network architectures based on different research insights, potentially not yielding the most optimal solution.*

In contrast, our method leverages videos' intrinsic audio-visual signals, constructs a multi-level search space (Fig. 2) tailored for audio-visual features, and employs NAS to discover the optimal network architecture for video hashing.

2.2 Neural Architecture Search

Neural Architecture Search (NAS) [11, 13] is a technique aimed at automating the design process of artificial neural networks (ANNs). It is widely used to create neural networks that are either comparable or superior to those manually designed by experts [60, 61]. NAS finds applications in various domains, including CNNs for computer vision tasks [36, 54, 57, 61], RNNs in natural language processing [4, 48], and even automatic speech recognition [14, 40].

In the field of computer vision, several NAS approaches stand out as exemplary, such as FP-DARTS [45], AutoFormer [6], GLiT [2], Point-NAS [52], and ViTAS [39].

Within the domain of natural language processing, notable NAS methods include DDNSA [4], NAS-BERT [53], NAS-HRL [5], HAT [43], and LLaMA-NAS [33].

In the realm of automatic speech recognition, LightSpeech [30] and DARTS-CONFORMER [37] are among the representative NAS approaches.

Remarks. *NAS has been extensively explored in diverse fields, including image classification, object detection, recommendation systems, knowledge tracing, and speech recognition. However, there remains ample untapped potential in NAS for learning to hash videos, presenting exciting prospects for further research.*

Currently, NAS primarily focuses on searching for various structures (e.g., CNNs, RNNs, Transformers) based on single-modal signals. Relatively limited effort has been dedicated to multi-level semantic modeling of multi-modal signals and comprehensive searches for hybrid architectures integrating MLPs, CNNs, Transformers [42], and Mamba [17].

3 Preliminaries

Videos encompass not only spatio-temporal keyframes but also temporal audio signals. This paper harnesses these inherent audio-visual cues, integrates them into a unified representation, and then

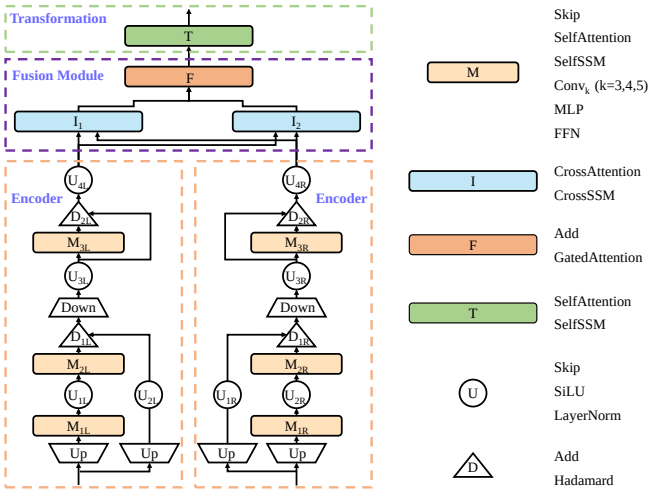


Figure 2: The AV-NAS search space includes 6 operation types: M for sequence modeling, I for cross-modal exchange, F for fusion, T for transformation, U for unary operations, and D for binary operations.

applies a transformation block to generate the final binary vector via a hash layer (as shown in Fig. 1).

Specifically, given a set of V videos denoted as $\mathcal{S} = \{\mathbf{v}_i\}_{i=1}^{V}$, where $\mathbf{v}_i$ represents the i-th video in the dataset $\mathcal{S}$, the proposed AV-NAS method aims to *search and learn* a hash function that maps each video to a compact binary vector.

Each video $\mathbf{v}_i$ is characterized by m time-series keyframes $\{\mathbf{I}_j\}_{j=1}^{m}$ and corresponding audio segments $\{\mathbf{A}_j\}_{j=1}^{m}$. We extact ViT [32] features $\{\mathbf{f}_j^{\mathrm{i}}\}_{j=1}^{m}$ for the keyframes and AST [15] features $\{\mathbf{f}_j^{\mathrm{a}}\}_{j=1}^{m}$ for the audio segments, where i and a denote the image and audio modalities, respectively. These features are processed within the branches of the AV-NAS framework (Fig. 1), undergoing separate encoding stages to produce image and audio embeddings. A Fusion module integrates these embeddings, followed by a transformation module for a unified representation. Finally, a hash layer generates a binary-like embedding for the video, which is binarized into the final hash code $\mathbf{b}_i \in \{-1, +1\}^q$ using the $\mathrm{Sign}(\cdot)$[1] function, where q marks the code length.

4 The Proposed Approach: AV-NAS

4.1 Network Components

The AV-NAS framework comprises several key components: "Input", "Encoder", "Fusion Module", "Transformation", and "Hash Layer", as illustrated in Fig. 1, which are described in detail below.

Input. Based on $\{\mathbf{f}_j^{\mathrm{i}}\}_{j=1}^{m} \in \mathbb{R}^{m\times 768}$ and $\{\mathbf{f}_j^{\mathrm{a}}\}_{j=1}^{m} \in \mathbb{R}^{m\times 768}$, we organize them as:

$$\mathbf{Z}_{\mathrm{i}} = [\mathbf{f}_1^{\mathrm{i}}, \mathbf{f}_2^{\mathrm{i}}, \cdots, \mathbf{f}_m^{\mathrm{i}}]^T, \tag{1}$$

$$\mathbf{Z}_{\mathrm{a}} = [\mathbf{f}_1^{\mathrm{a}}, \mathbf{f}_2^{\mathrm{a}}, \cdots, \mathbf{f}_m^{\mathrm{a}}]^T, \tag{2}$$

which correspond to the inputs of keyframes and audio segments, respectively.

[1] If $x \geq 0$, then $\mathrm{Sign}(x) = +1$; else, $\mathrm{Sign}(x) = -1$.

Encoder. AV-NAS covers two modality-specific encoders, each tailored to capture the distinct characteristics inherent to its respective modality. To endow the encoder with the capability to generalize to prevalent encoding models, including the Transformer [42], Mamba [17], CNNs and MLPs, we have devised a specialized structure, which is delineated in the encoder part of Fig. 2.

For an input $\mathbf{X} \in \mathbb{R}^{m\times d}$ of a certain modality, the forward pass can be expressed as follows:

$$\mathrm{E}_1(\mathbf{X}) = \mathrm{U}_3 \circ \mathrm{Down}\Big(\mathrm{D}_1(\mathrm{M}_2 \circ \mathrm{U}_1 \circ \mathrm{M}_1 \circ \mathrm{Up}(\mathbf{X}), \mathrm{U}_2 \circ \mathrm{Up}(\mathbf{X}))\Big), \tag{3}$$

$$\mathrm{E}_2(\mathbf{X}) = \mathrm{U}_4 \circ \mathrm{D}_2(\mathrm{M}_3(\mathbf{X}), \mathbf{X}), \tag{4}$$

$$\mathrm{Encoder}(\mathbf{X}) = \mathrm{E}_2 \circ \mathrm{E}_1(\mathbf{X}). \tag{5}$$

Here, $\mathrm{Down}(\mathbf{X}) = \mathrm{FC}_d(\mathbf{X})$ denotes a fully connected layer outputting d dimensions, while $\mathrm{Up}(\mathbf{X}) = \mathrm{FC}_{e\cdot d}(\mathbf{X})$ represents another fully connected layer outputting $e \cdot d$ dimensions, where e is a hyperparameter (defaulting to 2) controlling the dimensional expansion factor. The notations $\mathbf{U}_{*L}$ or $\mathbf{U}_{*R}$ indicate the left and right branches, respectively.

This Up-Down structure is devised to augment the encoder's generalizability, particularly accommodating architectures like Mamba [17] that exhibit dimension expansion followed by reduction. The symbol $\circ$ signifies the composition of two operations.

The overall architecture of the Encoder can be conceptualized as a two-component structure, echoing the design of the Transformer encoder. Within this framework, **M**, **U**, and **D** denote three distinct cell types, each serving a specific role in the network and offering a range of operations for selection. Our goal is to identify and implement the most optimal operation for each cell.

Note that the proposed model architecture, combined with the diverse set of operations available within each cell, confers exceptional flexibility upon the encoder. To demonstrate this flexibility, we show how our model can be configured to replicate several well-established network structures.

For example, the Transformer encoder [42] can be represented within our search space by selecting specific operations for each cell, as formulated below:

$$\begin{aligned}\mathrm{E}_1(\mathbf{X}) = \mathrm{LN} \circ \mathrm{Down}\Big(&\mathrm{MLP} \circ \mathrm{Skip} \circ \mathrm{SelfAttention} \circ \mathrm{Up}(\mathbf{X}) \\ &+ \mathrm{Skip} \circ \mathrm{Up}(\mathbf{X})\Big),\end{aligned} \tag{6}$$

$$\mathrm{E}_2(\mathbf{X}) = \mathrm{LN}\Big(\mathrm{FFN}(\mathbf{X}) + \mathbf{X}\Big), \tag{7}$$

$$\mathrm{Transformer}(\mathbf{X}) = \mathrm{E}_2 \circ \mathrm{E}_1(\mathbf{X}), \tag{8}$$

where E_1 denotes the attention module, and E_2 represents the feed-forward network (FFN) module. Note that "LN" signifies the LayerNorm operator. Unlike traditional Transformers, our model incorporates additional Up-Down operations on both the input and the overall output. These modifications, originally designed for compatibility with the Mamba structure, have subsequently shown empirical improvements in model performance (see Table 3).

Besides, Mamba [17], known for its efficiency in modeling long-range dependencies, could also be represented within our search space, as formulated below:

$$\begin{aligned}\mathrm{E}_1(\mathbf{X}) = \mathrm{Skip} \circ \mathrm{Down}\Big(&(\mathrm{SSM} \circ \mathrm{SiLU} \circ \mathrm{Conv}_k \circ \mathrm{Up}(\mathbf{X})) \\ &\otimes (\mathrm{SiLU} \circ \mathrm{Up}(\mathbf{X}))\Big),\end{aligned} \tag{9}$$

$$\mathrm{E}_2(\mathbf{X}) = \mathrm{LN}\Big(\mathrm{Skip}(\mathbf{X}) + \mathbf{X}\Big), \tag{10}$$

$$\mathrm{Mamba}(\mathbf{X}) = \mathrm{E}_2 \circ \mathrm{E}_1(\mathbf{X}). \tag{11}$$

Here, E_1 forms the core of Mamba, incorporating the SSM, SiLU, Conv, and Up/Down-operations. E_2 includes layer normalization (LN) and skip connections, mimicking the original Mamba's direct connections. Specifically, LN scales and shifts the data without altering its distribution, and the skip connection with the original input creates a residual structure, preserving main characteristics while enabling subtle adjustments. Thus, although not a strict direct connection, E_2 serves a similar function.

After unimodal feature encoding, we obtain feature sequences for both keyframes and audio segments:

$$\mathbf{E}_\mathrm{i} = \mathrm{Encoder}_{img}(\mathbf{Z}_\mathrm{i}), \tag{12}$$

$$\mathbf{E}_\mathrm{a} = \mathrm{Encoder}_{aud}(\mathbf{Z}_\mathrm{a}), \tag{13}$$

where $\mathbf{E}_\mathrm{i}$ and $\mathbf{E}_\mathrm{a}$ represent the encoded feature sequences for image and audio, respectively.

Fusion. To fully exploit the multimodal data's potential, capturing and fusing semantic interactions between audio and visual modalities are crucial. Therefore, we have designed a streamlined search space for the Fusion module:

$$\mathbf{F}_o = \mathrm{F}\Big(\mathrm{I}_1(\mathbf{E}_\mathrm{i}, \mathbf{E}_\mathrm{a}), \mathrm{I}_2(\mathbf{E}_\mathrm{a}, \mathbf{E}_\mathrm{i})\Big), \tag{14}$$

where the Fusion module comprises 2 types of operations: **I** and **F**. **I** facilitates the exchange of information between different modalities, capturing cross-modal dependencies and relationships to enhance the comprehension of multimodal inputs. Operation **F** combines the interacted features from various modalities, synthesizing the information into a unified representation.

As shown in Fig. 3, given two groups of input features, $\mathbf{X} \in \mathbb{R}^{m\times d}$ and $\mathbf{Y} \in \mathbb{R}^{m\times d}$, originating from distinct modalities, their sequential features are initially processed through LayerNorm and linear layers, respectively. Subsequently, these features are forwarded to the Conv1d and SiLU modules, which can be described as:

$$\mathbf{c}_x = \mathrm{Conv1d}_x(\mathrm{Linear}_x(\mathrm{LN}(\mathbf{X}))), \tag{15}$$

$$\mathbf{c}_x^{\mathrm{SiLU}} = \mathrm{SiLU}(\mathrm{Linear}_x(\mathrm{LN}(\mathbf{X}))), \tag{16}$$

$$\mathbf{c}_x^{\mathrm{Res}} = \mathrm{Linear}_x(\mathrm{LN}(\mathbf{X})), \tag{17}$$

$$\mathbf{c}_y = \mathrm{Conv1d}_y(\mathrm{Linear}_y(\mathrm{LN}(\mathbf{Y}))). \tag{18}$$

In what follows, $\mathbf{c}_x$ and $\mathbf{c}_y$ are simultaneously passed to CrossSSM modules, which are illustrated in Fig. 3&4 and formulated as:

$$\text{Transformation}: \begin{cases} \mathbf{B}_y = \mathrm{Linear}_\mathbf{B}(\mathbf{c}_y), & (19)\\ \mathbf{C}_y = \mathrm{Linear}_\mathbf{C}(\mathbf{c}_y), & (20)\\ \mathbf{\Delta}_y = \mathrm{Linear}_\mathbf{\Delta}(\mathbf{c}_y), & (21) \end{cases}$$

$$\text{Discretization}: \begin{cases} \overline{\mathbf{A}}_{1y} = \exp(\mathbf{\Delta}_y \mathbf{A}_{1y}), & (22)\\ \overline{\mathbf{A}}_{2y} = \exp(\mathbf{\Delta}_y \mathbf{A}_{2y}), & (23)\\ \overline{\mathbf{B}}_y = \mathbf{\Delta}_y \mathbf{B}_y, & (24) \end{cases}$$

$$\text{Cross Fusion}: \begin{cases} \mathbf{h}_x^t = \overline{\mathbf{A}}_{1y}\mathbf{h}_x^{t-1} + \overline{\mathbf{B}}_y \mathbf{c}_x^t, \quad \mathbf{h}_x^0 = 0, & (25)\\ \mathbf{z}^t = \mathbf{C}_y \mathbf{h}_x^t, & (26)\\ \mathbf{h}_x'^t = \overline{\mathbf{A}}_{2y}\mathbf{h}_x'^{t-1} + \overline{\mathbf{B}}_y \mathbf{c}_x'^t, \quad \mathbf{h}_x'^0 = 0, & (27)\\ \mathbf{z}'^t = \mathbf{C}_y \mathbf{h}_x'^t. & (28) \end{cases}$$

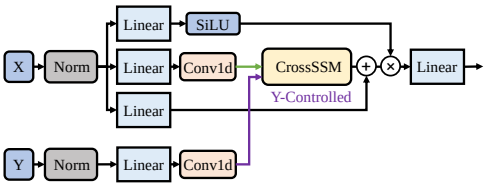


Figure 3: The CrossMamba block.

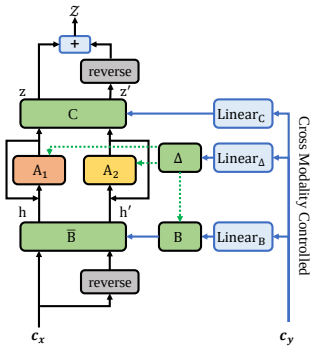


Figure 4: The CrossSSM module in CrossMamba (Fig. 3).

$$\text{Concatenation}: \begin{cases} \mathbf{z} = [\mathbf{z}^1, \mathbf{z}^2, \cdots, \mathbf{z}^m], & (29)\\ \mathbf{z}' = [\mathbf{z}'^1, \mathbf{z}'^2, \cdots, \mathbf{z}'^m], & (30) \end{cases}$$

where $\mathbf{c}_x^t$ represents the input at time step t and $\mathbf{z}$ denotes the selective scan output. To achieve bi-directional scanning, we reverse $\mathbf{c}_x$ to $\mathbf{c}_x'$, and then obtain the reversed sequence $\mathbf{z}'$.

Then we can get the output of CrossMamba:

$$\mathcal{Z} = \mathbf{z} + \mathrm{reverse}(\mathbf{z}'), \tag{31}$$

$$\mathbf{Z} = \mathrm{Linear}^F((\mathcal{Z} + \mathbf{c}_x^{\mathrm{Res}}) \otimes \mathbf{SiLU}(\mathbf{c}_x^{\mathrm{SiLU}})). \tag{32}$$

Thus, there comes $\mathbf{Z} = \mathrm{CrossMamba}(\mathbf{X}, \mathbf{Y})$, where the CrossSSM module corresponds to Eqs. (19-31), i.e., Fig. 4.

Here, drawing inspiration from the **CrossAttention** mechanism in Transformer [42], we introduce **CrossSSM**, a novel cross-fusion mechanism for multi-modalities, built upon Mamba [17]. Notably, Mamba's inherent SSM module is also termed **SelfSSM**, paralleling the relationship between **SelfAttention** and **CrossAttention** in Transformer [42].

Transformation. After fusing the two modality features, we introduce a Transformation module to derive the ultimate representation of the video. This transformation is:

$$\mathbf{h} = \mathrm{T}(\mathbf{F}_o), \tag{33}$$

where $\mathbf{h}$ represents the final sequence representation, and T signifies the transformation operation. Within our search space, we offer 2 alternatives for T: SelfAttention and SelfSSM.

Hash Layer. To obtain the final hash codes for the input videos, we append a hash layer, specifically a fully connected (FC) network equipped with the $\mathrm{Tanh}(\cdot)$ activation function, atop the Transformation module. Formally, given a video $\mathbf{v}_i$, we transform its output $\mathbf{h}$ from the Transformation module into a q-dimensional binary-like real-valued vector $\mathbf{f}_i$ via:

$$\mathbf{f}_i = \mathrm{Tanh}(\mathrm{FC}(\mathbf{h})) \in \mathbb{R}^q, \tag{34}$$

which is then forwarded to the Sign(·) function for the final binary code $\mathbf{b}_i$, i.e.,

$$\mathbf{b}_i = \text{Sign}(\mathbf{f}_i) \in \{-1, +1\}^q. \tag{35}$$

Remarks. *Please note that, as illustrated in Fig. 2, six distinct operation types (***M***, ***U***, ***D***, ***I***, ***F***, ***T***) are depicted, each encompassing a set of varying operators. This configuration results in an extensive potential search space, covering over* 7.83×10^{10} *combinations (calculated by multiplying the operator counts across all components). For detailed descriptions of these operations, please refer to Table 1.*

4.2 Objective Function

Our network is optimized using a contrastive learning approach, specifically the **InfoNCE loss** [21]. This loss function operates on triplets of samples: an anchor (the reference sample), a positive sample (from the same class as the anchor), and n_0 negative samples (from different classes). The core principle of InfoNCE is to enhance the model's discriminatory capability between similar and dissimilar samples by minimizing the negative log-likelihood function, i.e.,

$$\mathcal{L}(\mathbf{a}, \mathbf{p}, \mathbf{n}) = -\log \frac{e^{\mathbf{sim}(\mathbf{a},\mathbf{p})/\tau}}{e^{\mathbf{sim}(\mathbf{a},\mathbf{p})/\tau} + \sum_{i=1}^{n_0} e^{\mathbf{sim}(\mathbf{a},\mathbf{n}_i)/\tau}}, \tag{36}$$

where $\mathbf{a}$, $\mathbf{p}$, and $\mathbf{n} = \{\mathbf{n}_i\}_{i=1}^{n_0}$ represent the embeddings of the anchor, positive, and n_0 negative samples, respectively. The function $\mathbf{sim}(\cdot,\cdot)$ measures the similarity between two embeddings (e.g., cosine similarity), while τ is a hyper-parameter controlling the sensitivity of Eq. (36) to difficult negative samples.

In our implementation, we apply this loss function to video features. To be specific, let $\mathbf{a}_f, \mathbf{p}_f, \{\mathbf{n}_{f,j}\}_{j=1}^{n_0}$ denote the binary-like real-valued features extracted from the anchor, positive, and negative videos, respectively. The video-specific loss is then formulated as follows:

$$\mathcal{L}_V = \mathcal{L}(\mathbf{a}_f, \mathbf{p}_f, \{\mathbf{n}_{f,j}\}_{j=1}^{n_0}), \tag{37}$$

which encourages the network to learn representations that cluster videos of the same class closer together, while simultaneously pushing videos of different classes further apart.

4.3 Search Strategy

Denote by $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ the training and the validation loss, respectively. The objective of NAS is to identify an optimal architecture α^* that minimizes the validation loss $\mathcal{L}_{val}(\mathbf{W}^*, \alpha^*)$, where $\mathbf{W}^*$ represents the weights associated with α^* by minimizing the training loss, i.e., $\mathbf{W}^* = \arg\min_{\mathbf{W}} \mathcal{L}_{train}(\mathbf{W}, \alpha^*)$. This constitutes a bilevel optimization problem, with α as the upper-level variable and $\mathbf{W}$ as the lower-level variable:

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{\text{val}}(\mathbf{W}^*(\alpha), \alpha), \\ \text{s.t.} \quad & \mathbf{W}^*(\alpha) = \arg\min_{\mathbf{W}} \mathcal{L}_{train}(\mathbf{W}, \alpha). \end{aligned} \tag{38}$$

Directly optimizing the bilevel problem poses significant challenges due to its nested nature and the vast search space of possible architectures, making exhaustive search methods computationally intractable. To address these challenges, DARTS [29] relaxes the search space to be continuous and proposes a two-stage optimization approach for an approximate solution. Although promising, this method suffers from high memory consumption and an unstable process. Different from DARTS, we propose a simplified neural architecture search method that simultaneously optimizes weight parameters $\mathbf{W}$ and architecture parameters α.

We optimize a predefined structure where each component (e.g., **M**, **U**, **D**) corresponds to a cell. Each cell $\mathcal{F}$ contains multiple operations (e.g., **FFN**, **Conv**, **SSM**), and our goal is to select the optimal operation for each cell. To enable gradient-based optimization, we relax the discrete operation selection into a continuous domain. Following the approach of DARTS, we represent the mixed operation for each search space as a softmax distribution over all possible operations, i.e.,

$$\bar{f}(\mathbf{X}) = \sum_{f \in \mathcal{F}} \frac{\exp(\alpha_f)}{\sum_{f' \in \mathcal{F}} \exp(\alpha_{f'})} f(\mathbf{X}), \tag{39}$$

where α_f denotes the learnable weight associated with each primitive operation $f \in \mathcal{F}$.

During the architecture selection phase, we determine the optimal operation f^* for each search space by selecting the operation with the highest weight.

By leveraging continuous relaxation, we can jointly learn the architecture α and the weights $\mathbf{W}$ through gradient descent. Our method encompasses a three-stage optimization process: coarse search, pruning, and fine-tuning. The initial state is denoted as $\mathcal{N}(\mathbf{W}, \mathcal{A}(\alpha))$, where $\mathbf{W}$ represents the weight parameters and $\mathcal{A}(\alpha)$ represents the architecture parameterized by α. The training process is described as follows:

1. Coarse Search Stage: initial state: $\mathcal{N}(\mathbf{W}, \mathcal{A}(\alpha))$.

a) Update $\mathbf{W}$ and α using mini-batch gradient descent on the training set:

$$\min_{\mathbf{W},\alpha} \quad \mathcal{L}_{train}(\mathbf{W}, \alpha); \tag{40}$$

b) After each epoch, we evaluate the model performance on the validation set and save the best-performing $\mathbf{W}$ and α:

$$\mathbf{W}^*, \alpha^* = \arg\max_{\mathbf{W},\alpha} \text{mAP}_{val}(\mathbf{W}, \alpha); \tag{41}$$

Final state after Coarse Search Stage: $\mathcal{N}(W^*, \mathcal{A}(\alpha^*))$.

2. Pruning Stage: initial state: $\mathcal{N}(W^*, \mathcal{A}(\alpha^*))$.

a) Apply softmax to the architecture parameters for each mixed operation:

$$\beta_f = \frac{\exp(\alpha_f^*)}{\sum_{f' \in \mathcal{F}} \exp(\alpha_{f'}^*)}; \tag{42}$$

b) Select the operation with the highest softmax probability:

$$f^* = \arg\max_{f \in \mathcal{F}} \beta_f; \tag{43}$$

Final state after Pruning Stage: $\mathcal{N}(W^*, \mathcal{A}^*)$, $\mathcal{A}^*$ represents the deterministic architecture where each search space contains only the selected operation f^*.

3. Fine-tuning Stage: initial state: $\mathcal{N}(\mathbf{W}^*, \mathcal{A}^*)$.

a) Train the pruned network on the training set:

$$\min_{\mathbf{W}} \quad \mathcal{L}_{train}(\mathbf{W}, \mathcal{A}^*); \tag{44}$$

b) Evaluate the model performance on the validation set and save the best-performing weights:

$$\mathbf{W}^{**} = \arg\max_{\mathbf{W}} \text{mAP}_{val}(\mathbf{W}, \mathcal{A}^*); \tag{45}$$

Table 1: Summary of AV-NAS operations. Note that "———" indicates absence of hyperparameters.

Operation Type	Functionality	Specific Operations	Formula/Description	Hyperparameters
M	Sequence Modeling	FFN	$\mathbf{Z} = \mathrm{FC}_d \circ \mathrm{Drop}_p \circ \mathrm{ReLU} \circ \mathrm{FC}_{4d}(\mathbf{X})$	Dropout: $p = 0.1$
		MLP	$\mathbf{Z} = \mathrm{Drop}_p \circ \mathrm{FC}_d(\mathbf{X})$	Dropout: $p = 0.1$
		Conv_k	$\mathbf{Z} = \mathrm{Conv1d}_k(\mathbf{X})$	Kernel Size: $k \in \{3, 4, 5\}$
		SelfAttention	$\mathbf{Z} = \mathrm{MHA}(\mathbf{X}, \mathbf{X}, \mathbf{X})$ [42]	Head Num: 12
		SelfSSM	Detailed in Mamba [17]	———
U	Unary Transformation	SiLU	$\mathbf{Z} = \mathbf{X} \cdot \mathrm{Sigmoid}(\mathbf{X})$	———
		LayerNorm ("LN" for short)	Standard LayerNorm	———
		Skip	$\mathbf{Z} = \mathbf{X}$	———
D	Binary Transformation	Addition (+)	$\mathbf{Z} = \mathbf{X} + \mathbf{Y}$	———
		Hadamard Product ($\otimes$)	$\mathbf{Z} = \mathbf{X} \otimes \mathbf{Y}$	———
I	Cross-Modal Interaction	CrossAttention	$\mathbf{Z} = \mathrm{MHA}(\mathbf{X}, \mathbf{Y}, \mathbf{Y})$ [42]	Head Num: 12
		CrossSSM	Detailed in Fig. 3	———
F	Modal Fusion	Add	$\mathbf{Z} = \mathbf{X} + \mathbf{Y}$	———
		GatedAttention	$\mathbf{Z} = \rho \cdot \mathbf{X} + (1 - \rho) \cdot \mathbf{Y}$	Gate weight ρ: Learnable
T	Feature Transformation	SelfAttention	$\mathbf{Z} = \mathrm{MHA}(\mathbf{X}, \mathbf{X}, \mathbf{X})$ [42]	Head Num: 12
		SelfSSM	Detailed in Mamba [17]	———

Final state: $\mathcal{N}(\mathbf{W}^{**}, \mathcal{A}^{*})$.

After the "Coarse Search-Pruning-Finetuning" pipeline, the automatically discovered architecture and fine-tuned parameters jointly serve as a hashing function for video encodings.

5 Experiments

5.1 Datasets and Settings

Datasets. We evaluate AV-NAS using two large public video datasets: ActivityNet [22] and FCVID [23].

ActivityNet encompasses 14,948 YouTube videos spanning 200 classes, with durations varying from a few minutes to 30 minutes. For our experiments, we allocated 10,023 videos for training, 2,512 for validation, and 2,413 for testing, ensuring no overlap. Similarly, FCVID comprises 91,016 videos across 239 classes, averaging 167 seconds each. We randomly divided these into subsets: 45,508 for training, 22,754 for validation, and another 22,754 for testing, maintaining non-overlapping sets.

Baselines. We compared our AV-NAS method with several SOTA video hashing approaches, including SSTH [58], BTH [27], DKPH [25], MCMSH [20], ConMH [47], SRH [18], AVH [46], DSVH [3], and AVHash [59]. The first five competitors are unsupervised methods, while the last four are supervised ones. Note that the architectures of all these baseline methods are expert-designed based on CNN (3DCNN), MLP, LSTM, and Transformer architectures.

Besides, we also conducted comparative experiments by selecting NAS algorithms based on random selection, gradient optimization (DARTS [29], ProxylessNAS [1]), genetic algorithms (SPOS [19]), and reinforcement learning (ENAS [31]), all tailored to the search space we designed (Fig. 2), against our proposed search method.

Metrics. We evaluated video retrieval performance using three widely-adopted metrics: mean Average Precision at the top-K retrieved results (i.e., mAP@K), Precision at the top-K retrieved results (i.e., P@K), and Recall at the top-K retrieved results (i.e., R@K) [9, 18, 46, 47]. Additionally, we plotted P-R (Precision vs. Recall) curves to make comprehensive comparisons between the video retrieval behaviors of different methods.

Image/Audio Processing. Similar to SSTH [58], BTH [27], and ConMH [47], we uniformly select 25 frames for each video in the chosen datasets. Each image is resized to 336×336×3 and segmented into 14×14 patches. These images are then processed using the visual branch of CLIP [32], which features a depth of 24 layers, 16 multi-head attentions, and a hidden size of 1024, yielding high-level image representations within a 1024-dimensional feature space. These features are subsequently transformed into 768-dimensional vectors through a fully-connected (FC) layer.

To ensure temporal alignment, each audio track is uniformly segmented into 25 parts, matching the 25-frame video sampling rate. These audio segments are processed using the Audio Spectrogram Transformer (AST) [15], a 12-layer Transformer architecture featuring 12 multi-head attention mechanisms and 768-dimensional embeddings. Leveraging large-scale audio pretraining, AST extracts discriminative 768-D audio representations that capture high-level acoustic patterns.

Code Project. Following pre-processing, video inputs are processed through the AV-NAS architecture (Fig. 1) to produce compact binary hash codes. Our PyTorch implementation leverages NVIDIA Tesla V100 GPUs (32GB memory) for efficient training and inference. To facilitate reproducibility, AV-NAS is accessible at https://github.com/iFamilyi/AV-NAS, complete with datasets and parameter settings.

5.2 Results and Analyses

Automatically Discovered Architectures. Using the same AV-NAS search space, we experimented on both ActivityNet and FCVID, uncovering optimal network structures named *Arch-1* and *Arch-2*, as shown in Figs. 5a and 5b.

Table 2: Performance comparisons with SOTA methods w.r.t. mAP, time efficiency, and model size.

Line No.	Method/ (mAP,Time, Size)/ Dataset	ActivityNet							FCVID						
1		mAP (K=100)		Time (64bits)				#Params (64bits)	mAP (K=100)		Time (64bits)				#Params (64bits)
2		32bits	64bits	SuperNet /epoch	Search /time	Train /epoch	Infer /12.5k		32bits	64bits	SuperNet /epoch	Search /time	Train /epoch	Infer /12.5k	
3	MCMSH	0.1488	0.2849	0	0	1m31s	18s	1.2M	0.3442	0.3723	0	0	4m59s	59s	1.2M
4	BTH	0.1845	0.2721	0	0	26s	12s	2M	0.2987	0.3593	0	0	2m8s	2m18s	2M
5	DKPH	0.1528	0.2283	0	0	39s	17s	4M	0.3256	0.3731	0	0	54s	58s	4M
6	SSTH	0.1727	0.2395	0	0	4m42s	29s	5M	0.2275	0.3037	0	0	20m24s	5m38s	5M
7	ConMH	0.2974	0.3312	0	0	8s	13s	11.3M	0.3899	0.4232	0	0	30s	49s	11.3M
8	DSVH	0.2056	0.2979	0	0	2m33s	1m44s	63M	0.2616	0.3683	0	0	11m	6m19s	63M
9	SRH	0.3205	0.4266	0	0	8s	19s	4.5M	0.3699	0.4783	0	0	25s	38s	4.5M
10	AVH	0.3576	0.4722	0	0	10s	12s	22M	0.3318	0.4818	0	0	32s	37s	22M
11	AVHash	0.8530	0.8676	0	0	25s	7s	28.4M	0.9208	0.9213	0	0	1m47s	33s	28.4M
12	DARTS	0.8630	0.8860	2m17s	0	50s	10s	38.6M	0.9180	0.9199	10m	0	3m	1m16s	67.1M
13	SPOS	**0.8785**	0.8982	1m20s	6h	2m10s	19s	30.4M	0.9165	0.9198	6m20s	40h	5m40s	1m25s	32.1M
14	ProxylessNAS	0.8571	0.8690	2m	0	1m10s	22s	60.4M	0.8470	0.8638	3m	0	2m10s	1m15s	56.5M
15	ENAS	0.8545	0.8696	5m	30m	2m	16s	38.7M	0.9094	0.9193	7m	1h	10m	1m13s	36.3M
16	Random 1	0.8411	0.8562	0	0	28s	12s	23.9M	0.9143	0.9201	0	0	2m53s	53s	23.9M
17	Random 2	0.8649	0.8880	0	0	25s	9s	23.3M	0.9204	0.9264	0	0	1m49s	51s	23.3M
18	Random 3	0.7447	0.8071	0	0	32s	13s	35.7M	0.8908	0.8984	0	0	2m18s	57s	35.7M
19	Random 4	0.8420	0.8554	0	0	42s	12s	33.3M	0.9106	0.9181	0	0	3m	55s	33.3M
20	Random 5	0.8348	0.8573	0	0	22s	9s	21.9M	0.9063	0.9080	0	0	1m36s	41s	21.9M
21	Arch-1/2 -Visual	0.8650	0.8795	0	0	35s	9s	23.7M	0.9192	0.9231	0	0	1m44s	42s	21.4M
22	Arch-1/2 -Audio	0.2379	0.2541	0	0	34s	9s	22.7M	0.4350	0.4387	0	0	1m38s	42s	22.7M
23	AV-NAS (Arch-1)	0.8750	**0.9010**	1m6s	0	50s	11s	42.7M	0.9215	0.9253	4m54s	0	2m58s	53s	42.7M
24	AV-NAS (Arch-2)	0.8724	0.8950	1m6s	0	50s	11s	40.5M	**0.9230**	**0.9321**	4m54s	0	2m58s	53s	40.5M

Upon scrutinizing the architectures in Fig. 5, several insights emerge: (1) Both architectures (Fig. 5a&5b) share consistent Transformation and Fusion modules, along with a uniform encoder framework; while there are also some dataset-specific variations in the encoder internals. Specifically, Arch-2 incorporates additional σ activation functions and Layer Normalization (N) in the Visual Encoder, absent in Arch-1. Similarly, differences in the implementation of multiplication ($\otimes$) and addition (+) within the ResNet structure distinguish the Audio Encoders of Arch-1 and Arch-2. (2) The optimal network structures discovered by automated search share some components with expert designs, such as using the Transformer for spatio-temporal keyframe modeling. However, they also deviate, notably with the automated search opting for a novel "FFN+CNN" pattern for temporal audio modeling, contrary to experts' typical preference for Transformer or Mamba structures.

Despite slight variations in the branch encoder details, the overall spirit of the structures remains consistent, as also evidenced by the comparable mAP values and model sizes in Table 2 (Lines 23-24).

Performance Comparisons with SOTA methods. We conducted exhaustive experiments on public datasets for existing video hashing methods, summarizing the results, including those of our AV-NAS, in Table 2 (Lines 3-11, 23-24). Notably, AV-NAS discovered tailored network structures, Arch-1&Arch-2, for each dataset. Across all datasets, our evaluations revealed their superiority.

Upon reviewing Table 2, it is clear that AV-NAS outperforms others in mAP scores, demonstrating its effectiveness for large-scale video retrieval (p-value < 0.05 vs. AVHash). Both Arch-1 (optimal for ActivityNet) and Arch-2 (optimal for FCVID) achieved peak mAP values. When cross-tested, they maintained high retrieval metrics with minimal deviations due to their nearly identical structures with minor encoder differences. Therefore, Arch-1 and Arch-2 will be our default configurations moving forward.

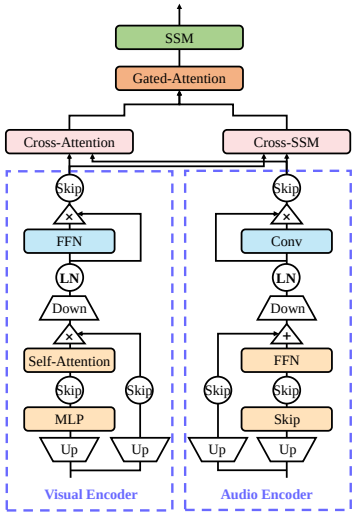
Moreover, we plotted P@K, R@K, and P-R curves in Fig. 6, showing AV-NAS's significant upward trend across all metrics. This underscores our method's superiority in retrieval accuracy and recall rate, providing a clear advantage in the comprehensive P-R curve.

5.3 Ablation Studies

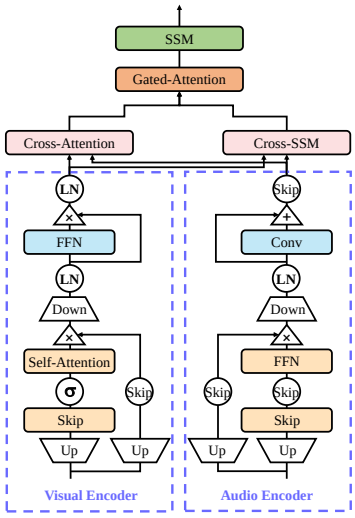
Expert-Designed vs. Automatically-Discovered. Intrigued by the possibility that automatically discovered network architectures could outperform expert designs in video retrieval, we developed 3 video hashing networks. Two were based on the pure Transformer architecture (e.g., Transformer [42] and its variant Transformer+ with additional Up and Down operators, akin to the visual branch in Fig.5b), and one was rooted in the pure Mamba architecture [17]. The experimental outcomes are concisely presented in Table 3.

Our analysis shows that video hashing methods based on Transformer and Mamba architectures exhibit comparable high mAP scores across datasets. However, AV-NAS, an automatically discovered architecture, significantly outperforms them. This highlights the efficacy of a data-driven NAS approach, where the architecture discovered through automated search demonstrates a refined ability to selectively utilize pertinent techniques for diverse multi-modal data representation.

Specifically, AV-NAS chooses an "FFN+CNN" combination for modeling temporal audio signals, achieving a more precise video semantic representation. Audio signals exhibit strong linearity and local temporal characteristics, making this combination effective. FFN integrates global features like pitch, tone, and emotion, while



(a) Arch-1: ActivityNet.



(b) Arch-2: FCVID.

Figure 5: Figs. 5a and 5b depict the architectures discovered on ActivityNet (*Arch-1*) and FCVID (*Arch-2*), respectively.

CNN captures local temporal features such as frequency changes and pitch shifts. This enables CNN to refine local variations based on FFN's global representation, reducing noise interference and enhancing robustness. In contrast, Transformer/Mamba, while skilled at modeling global spatiotemporal relationships, may overlook subtle local audio changes and accumulate noise in longer sequences.

Single-Signal vs. Multi-Signal. AV-NAS is specifically engineered to leverage combined audio-visual signals for semantic hash coding of videos. This raises the inquiry: what is the impact on video retrieval performance when relying solely on visual or audio signals? To address this, we developed video hashing techniques tailored for single-modal data within the AV-NAS framework across diverse datasets. These methods entail removing one modality branch and substituting the "Fusion Module" with a "Self-Fusion Module" (e.g., SelfAttention or SelfSSM), while preserving the remainder of the structure. The results are summarized in Table 2 (Lines 21-22).

Table 3: The comparison between the pure Transformer or Mamba based architectures and those found by AV-NAS.

Method/mAP/ Dataset	ActivityNet (K=100)		FCVID (K=100)	
	32 bits	64 bits	32 bits	64 bits
Transformer	0.8379	0.8569	0.9135	0.9150
Transformer+	0.8434	0.8653	0.9150	0.9170
Mamba	0.8300	0.8741	0.9102	0.9240
AV-NAS (Ours)	**0.8750**	**0.9010**	**0.9230**	**0.9321**

A clear pattern emerges: across datasets, video semantic hashing based solely on keyframes or audio falls short of methods integrating both. This highlights the significance of multi-modal semantic signals in video understanding. While visual signals dominate in video semantic representation, audio also contributes positively, reinforcing the complementary nature of these modalities in video analysis. These findings align with those reported in AVHash [59].

Different NAS Methods. As shown in Table 2 (Lines 12-20, 23-24), we gathered comprehensive data on supernet pre-training, structure search, fine-tuning times, and corresponding parameters. Notably, "Random 1-5" represents randomly assembled networks, while DARTS/ProxylessNAS use gradient optimization, SPOS employs a genetic algorithm, and ENAS leverages reinforcement learning. Our proposed AV-NAS adopts a relaxed holistic gradient optimization akin to DARTS but with a one-level approach.

Firstly, excluding random selection (which requires no pre-training or search), AV-NAS significantly outperforms DARTS, ProxylessNAS, ENAS, and SPOS in time efficiency. SPOS and ENAS, in particular, require substantial additional time for structure search based on pre-trained supernet weights.

Secondly, AV-NAS consistently achieves the highest mAP scores on both datasets, with consistent structures across datasets, unlike other methods like DARTS/ProxylessNAS (varying widely with 38.6M/60.4M and 67.1M/56.5M parameters on ActivityNet and FCVID) and SPOS/ENAS (smaller size variations but significant structural differences).

Thirdly, randomly searched structures exhibit significant mAP variability, with performance ranging from good ("Random 2") to poor ("Random 3").

In summary, our optimization strategy of relaxation, one-level holistic optimization, pruning, and fine-tuning emerges as a viable, effective, and robust NAS approach, validating its efficacy.

5.4 t-SNE Visualization

Given FCVID's extensive diversity across 200 categories, simultaneously comparing all videos poses a significant challenge. To tackle this, we curated **FCVID-small** by randomly selecting 10 categories, with 50 videos per category, serving as a benchmark for evaluating encoding capabilities.

For FCVID-small, we obtained 64-bit video representation vectors using four top-performing methods and visualized them in a 2D semantic space via t-SNE [41] (Fig. 7). The visualization reveals that AV-NAS clusters videos more distinctly than competitive

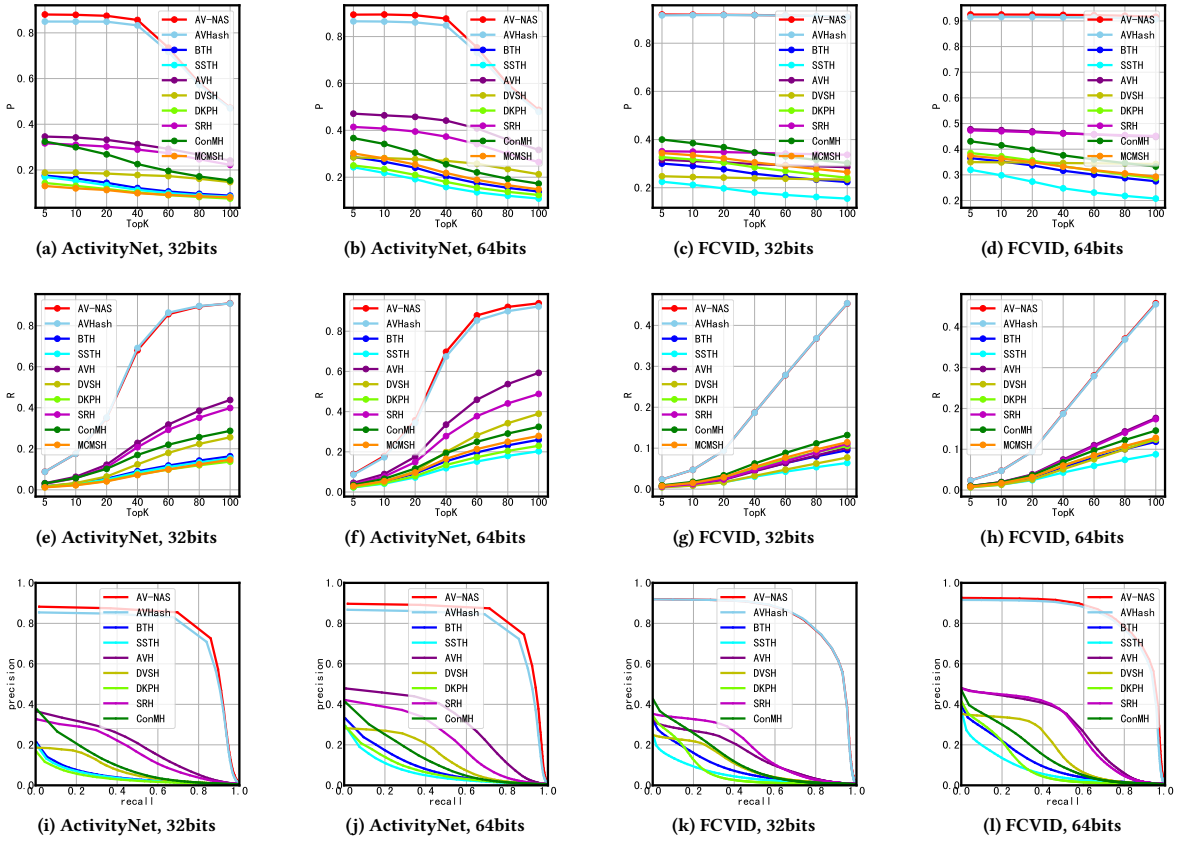


Figure 6: Performance comparisons w.r.t. P@K, R@K, and P-R curve on ActivityNet and FCVID datasets (32 bits and 64 bits).

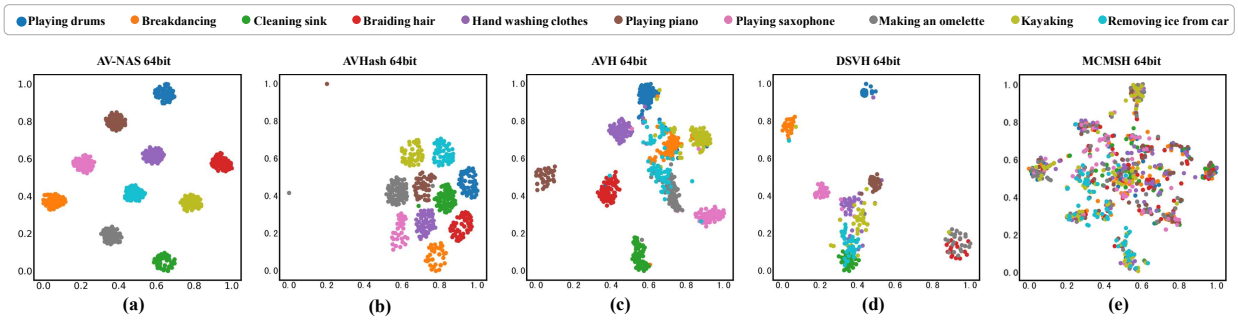


Figure 7: t-SNE visualizations of AV-NAS and four most competitive methods on FCVID-small.

supervised methods (AVHash/AVH/DSVH) and the unsupervised approach (MCMSH). This underscores AV-NAS's superior representation learning capabilities, also evident in ActivityNet.

6 Conclusions

This paper introduces a search space for multi-level semantic hybrid networks integrating audio-visual signals. It proposes a "coarse search-prune-finetune" framework to identify the optimal video hashing network, addressing a critical gap in NAS research for multi-modal video hashing. Experimental results show that AV-NAS not only identifies network structures similar to expert designs, such as Transformers and CrossAttention for spatio-temporal keyframe characterization, but also uncovers novel architectures. These include the "FFN+CNN" pattern for temporal audio modeling, a novel CrossSSM mechanism for cross-modal fusion, and an optimal transformation expression based on Mamba for unified features. Notably, the optimal neural network discovered by AV-NAS outperforms high-performing pure Transformer or Mamba architectures in retrieval metrics on large-scale video datasets.

References

[1] Han Cai, Ligeng Zhu, and Song Han. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *ICLR*.

[2] Boyu Chen, Peixia Li, Chuming Li, Baopu Li, Lei Bai, Chen Lin, Ming Sun, Junjie Yan, and Wanli Ouyang. 2021. GLiT: Neural Architecture Search for Global and Local Image Transformer. In *ICCV*. 12–21.

[3] Hanqing Chen, Chunyan Hu, Feifei Lee, Chaowei Lin, Wei Yao, Lu Chen, and Qiu Chen. 2021. A Supervised Video Hashing Method Based on a Deep 3D Convolutional Neural Network for Large-Scale Video Retrieval. *Sensors* 21, 9 (2021), 3094.

[4] Kuan-Chun Chen, Cheng-Te Li, and Kuo-Jung Lee. 2023. DDNAS: Discretized Differentiable Neural Architecture Search for Text Classification. *ACM Trans. Intell. Syst. Technol.* 14, 5 (2023), 88:1–88:22.

[5] Liang Chen, Xueming Yan, Zilong Wang, and Han Huang. 2023. Neural Architecture Search with Heterogeneous Representation Learning for Zero-Shot Multi-Label Text Classification. In *IJCNN*. 1–8.

[6] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. 2021. AutoFormer: Searching Transformers for Visual Recognition. In *ICCV*. 12250–12260.

[7] Yong Chen, Yuqing Hou, Shu Leng, Qing Zhang, Zhouchen Lin, and Dell Zhang. 2021. Long-Tail Hashing. In *SIGIR*. 1328–1338.

[8] Yong Chen, Zhibao Tian, Hui Zhang, Jun Wang, and Dell Zhang. 2020. Strongly Constrained Discrete Hashing. *IEEE Trans. Image Process.* 29 (2020), 3596–3611.

[9] Yong Chen, Hui Zhang, Zhibao Tian, Jun Wang, Dell Zhang, and Xuelong Li. 2022. Enhanced Discrete Multi-Modal Hashing: More Constraints Yet Less Time to Learn. *IEEE Trans. Knowl. Data Eng.* 34, 3 (2022), 1177–1190.

[10] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*. 253–262.

[11] Xuanyi Dong, David Jacob Kedziora, Katarzyna Musial, and Bogdan Gabrys. 2024. Automated Deep Learning: Neural Architecture Search Is Not the End. *Found. Trends Mach. Learn.* 17, 5 (2024), 767–920.

[12] Jingru Duan, Yanbin Hao, Bin Zhu, Lechao Cheng, Pengyuan Zhou, and Xiang Wang. 2024. Efficient Unsupervised Video Hashing With Contextual Modeling and Structural Controlling. *IEEE Trans. Multim.* 26 (2024), 7438–7450.

[13] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. *J. Mach. Learn. Res.* 20 (2019), 55:1–55:21.

[14] Yingying Gao, Shilei Zhang, Zihao Cui, Chao Deng, and Junlan Feng. 2023. Cascaded Multi-task Adaptive Learning Based on Neural Architecture Search. In *InterSpeech*. 246–250.

[15] Yuan Gong, Yu-An Chung, and James Glass. 2021. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778* (2021).

[16] Yunchao Gong and Svetlana Lazebnik. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*. 817–824.

[17] Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *CoRR* abs/2312.00752 (2023).

[18] Yun Gu, Chao Ma, and Jie Yang. 2016. Supervised Recurrent Hashing for Large Scale Video Retrieval. In *ACM Multimedia*. 272–276.

[19] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single Path One-Shot Neural Architecture Search with Uniform Sampling. In *ECCV*, Vol. 12361. 544–560.

[20] Yanbin Hao, Jingru Duan, Hao Zhang, Bin Zhu, Pengyuan Zhou, and Xiangnan He. 2022. Unsupervised Video Hashing with Multi-granularity Contextualization and Multi-structure Preservation. In *ACM Multimedia*. 3754–3763.

[21] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[22] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. ActivityNet: A large-scale video benchmark for human activity understanding. In *CVPR*. 961–970.

[23] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. 2018. Exploiting Feature and Class Relationships in Video Categorization with Regularized Deep Neural Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 2 (2018), 352–364.

[24] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou. 2016. Column Sampling Based Discrete Supervised Hashing. In *AAAI*. 1230–1236.

[25] Pandeng Li, Hongtao Xie, Jiannan Ge, Lei Zhang, Shaobo Min, and Yongdong Zhang. 2022. Dual-Stream Knowledge-Preserving Hashing for Unsupervised Video Retrieval. In *ECCV*, Vol. 13674. 181–197.

[26] Shuyan Li, Zhixiang Chen, Jiwen Lu, Xiu Li, and Jie Zhou. 2019. Neighborhood Preserving Hashing for Scalable Video Retrieval. In *ICCV*. 8211–8220.

[27] Shuyan Li, Xiu Li, Jiwen Lu, and Jie Zhou. 2021. Self-Supervised Video Hashing via Bidirectional Transformers. In *CVPR*. 13549–13558.

[28] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, and Jie Zhou. 2017. Deep Video Hashing. *IEEE Trans. Multim.* 19, 6 (2017), 1209–1219.

[29] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. In *ICLR*.

[30] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Jinzhu Li, Sheng Zhao, Enhong Chen, and Tie-Yan Liu. 2021. Lightspeech: Lightweight and Fast Text to Speech with Neural Architecture Search. In *ICASSP*. 5699–5703.

[31] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameter Sharing. In *ICML*, Vol. 80. 4092–4101.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, Vol. 139. 8748–8763.

[33] Anthony Sarah, Sharath Nittur Sridhar, Maciej Szankin, and Sairam Sundaresan. 2024. LLaMA-NAS: Efficient Neural Architecture Search for Large Language Models. *CoRR* abs/2405.18377 (2024).

[34] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised Discrete Hashing. In *CVPR*. 37–45.

[35] Xiaobo Shen, Yue Zhou, Yun-Hao Yuan, Xichen Yang, Long Lan, and Yuhui Zheng. 2023. Contrastive Transformer Hashing for Compact Video Representation. *IEEE Trans. Image Process.* 32 (2023), 5992–6003.

[36] Chunyin Sheng, Xiang Gao, Xiaopeng Hu, and Fan Wang. 2024. Differentiable Neural Architecture Search Based on Efficient Architecture for Lightweight Image Super-Resolution. In *MMM (Lecture Notes in Computer Science, Vol. 14556)*. 169–183.

[37] Xian Shi, Pan Zhou, Wei Chen, and Lei Xie. 2021. Darts-Conformer: Towards Efficient Gradient-Based Neural Architecture Search For End-to-End ASR. *CoRR* abs/2104.02868 (2021).

[38] Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Richang Hong. 2011. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM Multimedia*.

[39] Xiu Su, Shan You, Jiyang Xie, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, Xiao-Gang Wang, and Chang Xu. 2022. ViTAS: Vision Transformer Architecture Search. In *ECCV*, Vol. 13681. 139–157.

[40] Haiyang Sun, Zheng Lian, Bin Liu, Ying Li, Jianhua Tao, Licai Sun, Cong Cai, Meng Wang, and Yuan Cheng. 2023. EmotionNAS: Two-stream Neural Architecture Search for Speech Emotion Recognition. In *InterSpeech*. 3597–3601.

[41] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.

[43] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020. HAT: Hardware-Aware Transformers for Efficient Natural Language Processing. In *ACL*. 7675–7688.

[44] Liangdao Wang, Yan Pan, Cong Liu, Hanjiang Lai, Jian Yin, and Ye Liu. 2023. Deep Hashing with Minimal-Distance-Separated Hash Centers. In *CVPR*. 23455–23464.

[45] Wenna Wang, Xiuwei Zhang, Hengfei Cui, Hanlin Yin, and Yanning Zhang. 2023. FP-DARTS: Fast parallel differentiable neural architecture search for image classification. *Pattern Recognit.* 136 (2023), 109193:1–109193:11.

[46] Yingxin Wang, Xiushan Nie, Yang Shi, Xin Zhou, and Yilong Yin. 2021. Attention-Based Video Hashing for Large-Scale Video Retrieval. *IEEE Trans. Cogn. Dev. Syst.* 13, 3 (2021), 491–502.

[47] Yuting Wang, Jinpeng Wang, Bin Chen, Ziyun Zeng, and Shu-Tao Xia. 2023. Contrastive Masked Autoencoders for Self-Supervised Video Hashing. In *AAAI*. 2733–2741.

[48] Yujing Wang, Yaming Yang, Yiren Chen, Jing Bai, Ce Zhang, Guinan Su, Xiaoyu Kou, Yunhai Tong, Mao Yang, and Lidong Zhou. 2020. TextNAS: A Neural Architecture Search Space Tailored for Text Representation. In *AAAI*. 9242–9249.

[49] Rukai Wei, Yu Liu, Jingkuan Song, Heng Cui, Yanzhao Xie, and Ke Zhou. [n. d.]. CHAIN: Exploring Global-Local Spatio-Temporal Information for Improved Self-Supervised Video Hashing. In *ACM Multimedia*. 1677–1688.

[50] Gengshen Wu, Jungong Han, Yuchen Guo, Li Liu, Guiguang Ding, Qiang Ni, and Ling Shao. 2019. Unsupervised Deep Video Hashing via Balanced Code for Large-Scale Video Retrieval. *IEEE Transactions on Image Processing* 28, 4 (2019), 1993–2007.

[51] Ke Xia, Yuqing Ma, Xianglong Liu, Yadong Mu, and Li Liu. 2017. Temporal Binary Coding for Large-Scale Video Search. In *ACM Multimedia*. 333–341.

[52] Tao Xie, Haoming Zhang, Linqi Yang, Ke Wang, Kun Dai, Ruifeng Li, and Lijun Zhao. 2023. Point-NAS: A Novel Neural Architecture Search Framework for Point Cloud Analysis. *IEEE Trans. Image Process.* 32 (2023), 6526–6542.

[53] Jin Xu, Xu Tan, Renqian Luo, Kaitao Song, Jian Li, Tao Qin, and Tie-Yan Liu. 2021. NAS-BERT: Task-Agnostic and Adaptive-Size BERT Compression with Neural Architecture Search. In *KDD*. 1933–1943.

[54] Xun Yang, Shanshan Wang, Jian Dong, Jianfeng Dong, Meng Wang, and Tat-Seng Chua. 2022. Video Moment Retrieval With Cross-Modal Neural Architecture Search. *IEEE Trans. Image Process.* 31 (2022), 1204–1216.

[55] Guangnan Ye, Dong Liu, Jun Wang, and Shih-Fu Chang. 2013. Large-Scale Video Hashing via Structure Learning. (2013), 2272–2279.

[56] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis E. H. Tay, Zequn Jie, Wei Liu, and Jiashi Feng. 2020. Central Similarity Quantization for Efficient Image and Video Retrieval. In *CVPR*. 3080–3089.

[57] Lin Zhan, Jiayuan Fan, Peng Ye, and Jianjian Cao. 2023. A2S-NAS: Asymmetric Spectral-Spatial Neural Architecture Search for Hyperspectral Image Classification. In *ICASSP*. 1–5.
[58] Hanwang Zhang, Meng Wang, Richang Hong, and Tat-Seng Chua. 2016. Play and Rewind: Optimizing Binary Representations of Videos by Self-Supervised Temporal Hashing. In *ACM Multimedia*. 781–790.
[59] Yuxiang Zhou, Zhe Sun, Rui Liu, Yong Chen, and Dell Zhang. 2024. AVHash: Joint Audio-Visual Hashing for Video Retrieval. In *ACM Multimedia*. 2370–2378.
[60] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *ICLR*.
[61] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning Transferable Architectures for Scalable Image Recognition. In *CVPR*. 8697–8710.