This CVPR 2021 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Compatibility-aware Heterogeneous Visual Search

Rahul Duggal* Hao Zhou Shuo Yang Yuanjun Xiong
Wei Xia† Zhuowen Tu Stefano Soatto

AWS/Amazon AI

rduggal7@gatech.edu {zhouho, shuoy, yuanjx, wxia, ztu, soattos}@amazon.com

Abstract

*We tackle the problem of visual search under resource constraints. Existing systems use the same embedding model to compute representations (embeddings) for the query and gallery images. Such systems inherently face a hard accuracy-efficiency trade-off: the embedding model needs to be large enough to ensure high accuracy, yet small enough to enable query-embedding computation on resource-constrained platforms. This trade-off could be mitigated if gallery embeddings are generated from a large model and query embeddings are extracted using a compact model. The key to building such a system is to ensure representation compatibility between the query and gallery models. In this paper, we address two forms of compatibility: One enforced by modifying the parameters of each model that computes the embeddings. The other by modifying the architectures that compute the embeddings, leading to compatibility-aware neural architecture search (*CMP-NAS*). We test* CMP-NAS *on challenging retrieval tasks for fashion images (DeepFashion2), and face images (IJB-C). Compared to ordinary (homogeneous) visual search using the largest embedding model (paragon),* CMP-NAS *achieves 80-fold and 23-fold cost reduction while maintaining accuracy within* 0.3% *and* 1.6% *of the paragon on DeepFashion2 and IJB-C respectively.*

1. Introduction

A visual search system in an "open universe" setting is often composed of a gallery model ϕ_g and a query model ϕ_q, both mapping an input image to a vector representation known as *embedding*. The gallery model ϕ_g is typically used to map a set of gallery images onto their embedding vectors, a process known as indexing, while the query model extracts embeddings from query images to perform search against the indexed gallery. Most existing visual search approaches [24, 26, 38, 2, 31] use the same model architecture for both ϕ_q and ϕ_g. We refer to this setup as *homogeneous visual search*. An approach that uses different model architectures for ϕ_q and ϕ_g is referred to as *heterogeneous visual search* (HVS).

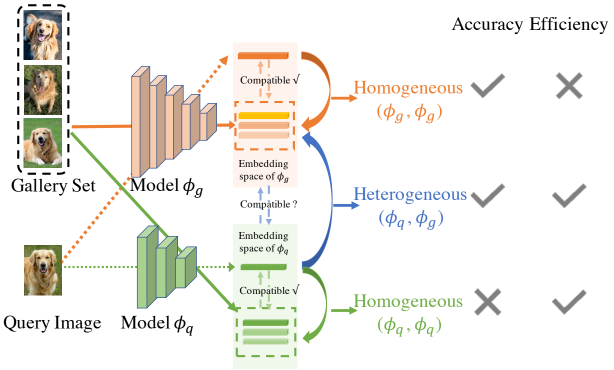


Figure 1: Homogeneous visual search uses the same embedding model, either large (orange) to meet performance specifications, or small (green) to meet cost constraints, forcing a dichotomy. Heterogeneous Visual Search (blue) uses a large model to compute embeddings for the gallery, and a small model for the query images. This allows high efficiency without sacrificing accuracy, provided that the green and orange embedding models are designed and trained to be *compatible*.

The use of the same $\phi_g = \phi_q$ trivially ensures that gallery and query images are mapped to the same vector space where the search is conducted. However, this engenders a hard accuracy-efficiency trade-off (Fig. 1)—choosing a large architecture ϕ_g for both query and gallery achieves high-accuracy at a loss of efficiency; choosing a small architecture ϕ_q improves efficiency to the detriment of accuracy, which is compounded since in practice, indexing only happens sporadically while querying is performed continuously. This leads to efficiency being driven mainly by the query model. HVS allows the use of a small model ϕ_q for querying, and a large model ϕ_g for indexing, partly

[*] Currently at the Georgia Institute of Technology. Work conducted during an internship with Amazon AI.

[†] Corresponding author

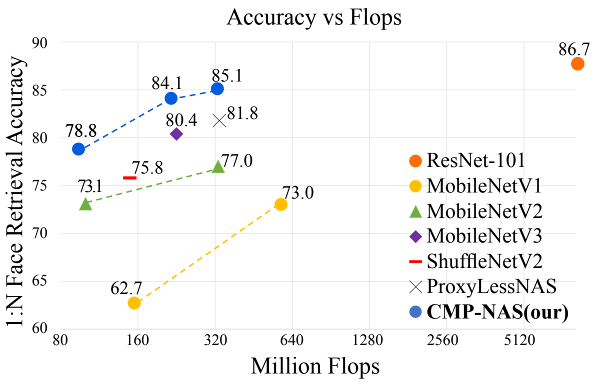


Figure 2: The trade-off between accuracy and efficiency for a heterogeneous system performing 1:N Face retrieval on DeepFashion2. We use a ResNet-101 as the gallery model and compare different architectures as query models. For MobileNetV1 and V2, we provide results with width $0.5\times$ and $1\times$.

mitigating the accuracy-complexity trade-off by enlarging the trade space. The challenge in HVS is to ensure that ϕ_g and ϕ_q live in the same metric (vector) space. This can be done for given architectures ϕ_g, ϕ_q, by training the weights so the resulting embeddings are metrically compatible [28]. However, one can also enlarge the trade space by including the architecture in the design of metrically compatible models. Typically, ϕ_g is chosen to match the best current state-of-the-art (paragon) while the designer can search among query architectures ϕ_q to maximize efficiency while ensuring that performance remains close to the paragon.

In this work, we pursue compatibility by optimizing both the model parameters (weights) as well as the model architecture. We show that (1) weight inheritance [19] and (2) backward-compatible training (BCT) [28] can achieve compatibility through weight optimization. Among these, the latter is more general in that it works with *arbitrary* embedding functions ϕ_g and ϕ_q. We expand beyond BCT to neural architecture search (NAS) [43, 4, 8, 30] with our proposed compatibility-aware NAS (CMP-NAS) strategy that searches for a query model ϕ_q that is maximally efficient while being compatible with ϕ_g. We hypothesize that CMP-NAS can simultaneously find the architecture of query model and its weights that achieve efficiency similar to that of the smallest (query) model, and accuracy close to that of the paragon (gallery model). Indeed the results in Fig. 2 shows that CMP-NAS outperforms all of the state-of-the-art off-the-shelf architectures designed for mobile platforms with resource constraints. Compared with paragon (state-of-the-art high-compute homogeneous visual search) methods, HVS reduce query model flops by 23× with only 1.6% in loss of search accuracy for the task of face retrieval.

Our *contributions* can be summarized as follows: 1) we demonstrate that an HVS system allows to better trade off accuracy and complexity, by optimizing over both model parameters and architecture. 2) We propose a novel CMP-NAS method combining weight-based compatibility with a novel reward function to achieve compatibility-aware architecture search for HVS. 3) We show that our CMP-NAS can reduce model complexity many-fold with only a marginal drop in accuracy. For instance, we achieve 23× reduction in flops with only 1.6% drop in retrieval accuracy on face retrieval using standard benchmarks.

2. Related Work

Visual search: Most prior visual search systems construct embedding vectors either by aggregating hand-crafted features [37, 23, 29, 1, 40, 42], or through feature maps extracted from a convolutional neural network [26, 41, 38, 32, 26, 2, 14, 31, 25]. The latter, being more prevalent in recent times, differ from us in that they follow the homogeneous visual search setting and suffer from a hard accuracy and efficiency trade-off. Recently, [3] discusses the asymmetric testing task which is similar to our heterogeneous setting. However, their method is unable to ensure that the heterogeneous accuracy supersedes the homogeneous one (compatibility rule in Sec. 3.1.1). Such a system is not practically useful since the homogeneous deployment achieves both a higher accuracy and a higher efficiency.

Cross-model compatibility: The broad goal of this area is to ensure embeddings generated by different models are compatible. Some recent works ensure cross-model compatibility by learning transformation functions from the query embedding space to the gallery one [33, 5, 13]. Different from these works, our approach directly optimizes the query model such that its metric space aligns with that of the gallery. This leads to more flexibility in designing the query model and allows us to introduce architecture search in the metric space alignment process. Our idea of model compatibility, as metric space alignment, is similar to the one in backward-compatible training (BCT) [28]. However, [28] only considers compatibility through model weights, whereas, we generalize this concept to the model architecture. Additionally, [28] targets for compatibility between an updated model and its previous (less powerful) version, the application scenarios of which are different from this work.

Architecture Optimization: Recent progress demonstrates the advantages of automated architecture design over manual design through techniques such as neural architecture search (NAS) [43, 30, 8, 4]. Most existing NAS algorithms however, search for architectures that achieve the best accuracy when used independently. In contrast, our task necessitates a deployment scenario with two models: one for processing the query images and another for processing the gallery. Recently, [18, 15] propose to use a large teacher model to guide the architecture search process for a smaller student which is essentially knowledge distillation in architecture space. However, our experiments show that knowledge distillation cannot guarantee compatibility and thus these methods may not succeed in optimizing the architecture in that aspect. To the best of our knowledge, CMP-NAS is the first to consider the notion of compatibility during architecture optimization.

3. Methodology

We use ϕ to denote an embedding model in a visual search system and κ to denote the classifier that is used to train ϕ. We further assume ϕ is determined by its architecture a and weights w. For our visual search system, a gallery model is first trained on a training set $\mathcal{T}$ and then used to map each image x in the gallery set $\mathcal{G}$ onto an embedding vector $\phi_g(x) \in \mathbb{R}^K$. Note this mapping process only uses the embedding portion ϕ_g. During test time, we use the query model (trained previously on $\mathcal{T}$) to map the query image x' onto an embedding vector $\phi_q(x') \in \mathbb{R}^K$. The closest match is then obtained through a nearest neighbor search in the embedding space. Typically, visual search accuracy is measured through some metric, such as top-10 accuracy, which we denote by $M(\phi_q, \phi_g; \mathcal{Q}, \mathcal{G})$. This is calculated by processing query image set $\mathcal{Q}$ with ϕ_q and processing the gallery set $\mathcal{G}$ with ϕ_g. For simplification, we omit the image sets and adopt the notation $M(\phi_q, \phi_g)$ to denote our accuracy metric.

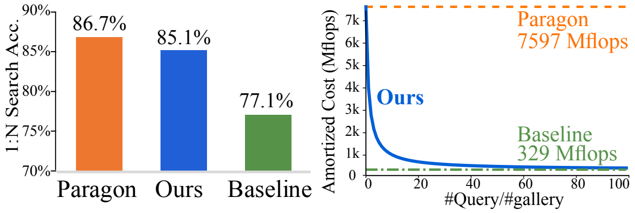
3.1. Homogeneous *vs*. heterogeneous visual search

Assuming ϕ_q and ϕ_g are different models and ϕ_q is smaller than ϕ_g, we define two kinds of visual search:

- **Homogeneous visual search** uses the same embedding model to process the gallery and query images, and is denoted by (ϕ_q, ϕ_q) or (ϕ_g, ϕ_g).
- **Heterogeneous visual search** uses different embedding models to process the query and gallery images, respectively, and is denoted by (ϕ_q, ϕ_g).

We illustrate the accuracy-efficiency trade-off faced by visual search systems in Fig. 3. A homogeneous system with a larger embedding model (*e.g*. ResNet-101 [9], denoted as paragon) achieves a higher accuracy due to better embeddings (orange bar in Fig. 3(a)) but also consumes more flops during query time (orange line in Fig. 3(b)). On the other hand, a smaller embedding model (*e.g*. MobileNetV2 [27], denoted as baseline) in the homogeneous setting achieves the opposite end of the trade-off (green bar and line in Fig. 3(a),(b)). Our heterogeneous system (blue bar and line in Fig. 3(a),(b)) achieves accuracy within 1.6% of the paragon and efficiency of the baseline.

When computing the cost of a visual search method, one has to take into account both the cost of indexing, which happens sporadically, and the cost of querying, which occurs continuously. While large, the indexing cost is amortized through the lifetime of the system. To capture both, in Fig. 3(b) we report the amortized cost of embedding the query and gallery images, as a function of the ratio of queries to gallery images processed. In most practical systems, the number of queries exceeds the number of indexed images by orders of magnitude, so the relevant cost is the asymptote, but we report the entire curve for completeness. The initial condition for that curve is the cost of the paragon. Our goal is to design a system that has a cost approaching the asymptote (b), with performance approaching the paragon (a).



(a) Accuracy on IJB-C.　(b) Amortized cost analysis

Figure 3: Accuracy-efficiency trade-off for visual search. In (a) we compare the 1:N face retrieval accuracy (TPIR@FPIR=10^{-1}) on IJB-C. We denote the homogeneous system with ResNet-101 and MobileNetV2 as the paragon and baseline respectively. In (b) we observe that, as the size of the query set increases, the complexity of our heterogeneous system converges to that of the baseline.

3.1.1 Notion of Compatibility

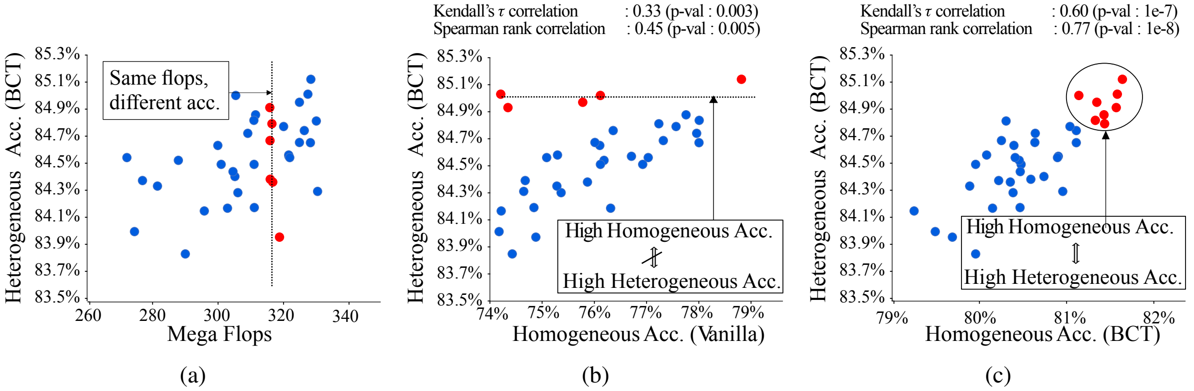
A key requirement of a heterogeneous system is that the query and gallery models should be compatible. We define this notion through the **compatibility rule** which states that:

> A *smaller* model ϕ_q is compatible with a *larger* model ϕ_g if it satisfies the inequality $M(\phi_q, \phi_g) > M(\phi_q, \phi_q)$.

We note that satisfying this rule is a necessary condition for heterogeneous search. A heterogeneous system violating this condition, *i.e*. $M(\phi_q, \phi_g) < M(\phi_q, \phi_q)$, is not practically useful since the homogeneous system $M(\phi_q, \phi_q)$ achieves both higher efficiency and accuracy. Additionally, a practically useful heterogeneous system should also satisfy $M(\phi_q, \phi_g) \approx M(\phi_g, \phi_g)$. In subsequent sections, we study how to achieve both these goals through weight and architecture compatibility.

3.2. Compatibility for Heterogeneous Models

In this section, we discuss different ways to obtain compatible query and gallery models ϕ_q, ϕ_g that satisfy the compatibility rule. While a general treatment may optimize ϕ_q and ϕ_g jointly, in this paper, we consider the simpler case when ϕ_g is fixed to a standard large model (ResNet-101) while we optimize the query model ϕ_q. For the subsequent discussion, we assume the gallery model ϕ_g has an architecture a_g and is parameterized by weights w_g. Corresponding



(a)

(b)

(c)

Figure 4: NAS Motivation. We randomly sample 40 architectures from the ShuffleNet search space of [8] and train them from scratch. Observe that (a) Architectures with same flops (shown with red circles) can have different heterogeneous accuracies proving that architecture has a measurable impact on compatibility. (b) Architectures (shown in red) achieving the highest heterogeneous accuracy with BCT training are not the ones achieving the highest homogeneous accuracy with vanilla training. This means that traditional NAS (which optimizes for homogeneous accuracy while using vanilla training) may fail to find the most compatible models. (c) When trained with BCT, the architectures achieving the highest heterogeneous accuracy also achieve the highest homogeneous accuracy. This means simply equipping traditional NAS with BCT will aid the search for compatible architectures.

quantities for the query model are ϕ_q with architecture a_q and parameterized by weights w_q. To train the query and gallery models ϕ_q, ϕ_g we use the classification-based training [28, 39] with the query and gallery classifiers denoted by κ_q and κ_g respectively. In what follows, we discuss two levels of compatibility—weight level and architecture level.

3.2.1 Weight-level compatibility

Given the gallery model ϕ_g and its classifier κ_g, weight-level compatibility aims to learn the weights w_q of query model ϕ_q such that the compatibility rule is satisfied. To this end, the optimal query weights w^*_q, and its corresponding classifier κ^*_q can be learned by minimizing a composite loss over the training set $\mathcal{T}$.

$$w^*_q, \kappa^*_q = \underset{w_q, \kappa_q}{\operatorname{argmin}} \{\lambda_1 \mathcal{L}_1(w_q, \kappa_q; \mathcal{T}) + \lambda_2 \mathcal{L}_2(w_q, \kappa_q, w_g, \kappa_g; \mathcal{T})\}, \quad (1)$$

where $\mathcal{L}_1$ is a classification loss such as the Cosine margin [36], Norm-Softmax [35] and $\mathcal{L}_2$ is the additional term which promotes compatibility. We consider four training methods which can be described using Eq.1 as follows:

1. Vanilla training: Considers $\lambda_2 = 0$.
2. Knowledge Distillation [11]: $\mathcal{L}_2$ is the temperature smoothed cross-entropy loss between the logits of query and gallery model.
3. Fine-tuning: Initializes w_q using w_g and κ_q using κ_g and considers $\lambda_2 = 0$.
4. Backward-compatible training (BCT) [28]: Uses $\mathcal{L}_2 = \mathcal{L}_1(w_q, \kappa_g; \mathcal{T})$. This ensures that the query embedding model learns a representation that is compatible with the gallery classifier.

We compare these methods in Tab. 4, and find that only the last two succeed in ensuring compatibility. Among these two, fine-tuning is more restrictive since it makes a stronger assumption about the query architecture—it requires the query model to have a similar network structure, kernel size, layer configuration as the gallery model. In contrast, BCT poses no such restriction and can be used to train any query architecture. Thus we use [28] as our default method to ensure weight-level compatibility. Recently [3] proposed to learn the weights of a query model by minimizing the L_2 distance between query and gallery embeddings, however, both [3] and [28] observe that the resulting query model does not satisfy the compatibility rule.

3.2.2 Architecture-level compatibility

Given the gallery model ϕ_g and classifier κ_g, the problem of architecture-level compatibility aims to search for an architecture a_q for the query model ϕ_q that is most compatible with a fixed gallery model. The need of architecture level compatibility is motivated by two questions:

Q1 How much does architecture impact compatibility?

Q2 Can traditional NAS find compatible architectures?

To answer these questions, we randomly sample 40 architectures from the ShuffleNet search space [8], with each having roughly 300 Million flops.

A1 We train these architectures with BCT ($\lambda_1 = 1, \lambda_2 = 1$ in Eq. 1) and plot the heterogeneous accuracy *vs.* flops in Fig. 4(a). There are two observations: (1) heterogeneous accuracy is not correlated with flops and (2) architectures with similar flops can achieve different accuracy, which indicates architecture indeed has a measurable impact on accuracy.

A2 We plot the homogeneous accuracy of models with vanilla training (target of traditional NAS) *vs*. heterogeneous accuracy of the same models trained with BCT (our target) in Fig. 4(b). We observe that: (1) The correlation between the two accuracy is low and (2) The architectures with the highest heterogeneous accuracy are not those with highest homogeneous accuracy. This indicates traditional NAS may not be successful in searching for compatible architectures.

We further investigate the correlation between homogeneous (with BCT) and heterogeneous accuracy (with BCT) in Fig. 4(c) and discover that the correlation of these two accuracies is much higher than that in Fig. 4(b). This offers a key insight that equipping traditional NAS with BCT may help in searching compatible architectures.

Architecture optimization with CMP-NAS Based on the intuition developed previously, we develop CMP-NAS using the following notation. Denote by $\phi_q(a_q, w_q)$ a candidate query embedding model with architecture a_q and weights w_q. We further denote κ_q as its corresponding classifier. With CMP-NAS, we solve a two-step optimization problem where the first step amounts to learning the best set of weights— w_q^* (for the embedding model ϕ_q) and κ_q^* (for the common classifier)–by minimizing a classification loss $\mathcal{L}$ over the training set $\mathcal{T}$ as below

$$\begin{aligned} w_q^*, \kappa_q^* = \underset{w_q, \kappa_q}{\operatorname{argmin}} \{ & \lambda_1 \mathcal{L}(\phi_q(a_q, w_q), \kappa_q; \mathcal{T}) \\ & + \lambda_2 \mathcal{L}(\phi_q(a_q, w_q), \kappa_g; \mathcal{T}) \}, \end{aligned} \quad (2)$$

where $\mathcal{L}$ can be any classification loss such as Cosine margin [36], Norm-Softmax [35]. Similar to BCT, the second term $\mathcal{L}(\phi_q(a_q, w_q), \kappa_g; \mathcal{T})$ ensures that the candidate query embedding model $\phi_q(a_q, w_q^*)$ learns a representation that is compatible with the gallery classifier.

Using weights w_q^* and κ_q^* from above, the second step amounts to finding the best query architecture a_q^* in a search space Ω, by maximizing a reward $\mathcal{R}$ evaluated on the validation set as below

$$a_q^* = \underset{a_q \in \Omega}{\operatorname{argmax}} \, \mathcal{R} \left(\phi_q(a_q, w_q^*), \kappa_q^* \right) . \quad (3)$$

We consider three candidate rewards presented in Tab 1. Similar to traditional NAS, homogeneous accuracy $M(\phi_q(a_q, w_q), \phi_q(a_q, w_q))$ is our baseline reward $\mathcal{R}_1$. Recall that however, we are interested in searching for the architecture which achieves the best heterogeneous accuracy. With this aim, we design rewards $\mathcal{R}_2$ and $\mathcal{R}_3$ which include the heterogeneous accuracy in their formulation.

Reward	Formulation
$\mathcal{R}_1$	$M(\phi_q(a_q, w_q), \phi_q(a_q, w_q)$
$\mathcal{R}_2$	$M(\phi_q(a_q, w_q), \phi_g)$
$\mathcal{R}_3$	$M(\phi_q(a_q, w_q), \phi_q(a_q, w_q)) \times M(\phi_q(a_q, w_q), \phi_g)$

Table 1: Different rewards considered with CMP-NAS. The rewards $\mathcal{R}_1$, $\mathcal{R}_2$ prioritize either the symmetric or asymmetric accuracy while ignoring the other. $\mathcal{R}_3$ prioritizes *both* accuracies and consistently outperforms other rewards.

Our CMP-NAS formulation in Eq. 2 and rewards in Tab. 1 is general and can work with any NAS method. For demonstration, we test our idea with the single path one shot NAS [8] and consists of the following two components:

Search Space: Similar to popular weight sharing methods [4, 8, 17], the search space of our query model consists of a shufflenet-based super-network. The super-network consists of 20 sequentially stacked choice blocks. Each choice block can select one of four operations: $k \times k$ convolutional blocks ($k \in 3, 5, 7$) inspired by ShuffleNetV2 [21] and a 3×3 Xception [6] inspired convolutional block. Additionally, each choice block can also select from 10 different channel choices $0.2 - 2.0\times$. During training we use the loss formulation in Eq. 2 to train this super-network whereby, in each batch a new architecture is sampled uniformly [8] and only the weights corresponding to it are updated.

Search Strategy: To search for the most compatible architecture, CMP-NAS uses evolutionary search [8] fitted with the different rewards outlined in Tab. 1. The search is fast because each architecture inherits the weights from the super-network. In the end, we obtain the five best architectures and re-train them from scratch with BCT.

4. Experiments

We evaluate the efficacy of our heterogeneous system on two tasks: face retrieval, as it is one of the "open-universe" problems with the largest publicly available datasets; and fashion retrieval which necessitates an open-set treatment due to the constant evolution of fashion items. We use face retrieval as the main benchmark for our ablation studies.

4.1. Datasets, Metrics and Gallery Model

Face Retrieval: We use the IMDB-Face dataset [34] to train the embedding model for the face retrieval task. The IMDB-Face dataset contains over 1.7M images of about 59k celebrities. If not otherwise specified, we use 95% of the data as training set to train our embedding model and use the remaining 5% as a validation dataset to compute the rewards for architecture search. For testing, we use the widely used IJB-C face recognition benchmark dataset [22].

The performance is evaluated using the true positive identification rate at a false positive identification rate of 10^{-1} (TPIR@FPIR=10^{-1}). Throughout the evaluation, we use a ResNet-101 as the fixed gallery model ϕ_g.

Fashion Retrieval: We evaluate the proposed method on Commercial-Consumer Clothes Retrieval task on DeepFashion2 dataset [7]. It contains 337K commercial-consumer clothes pairs in the training set, from which 90% of the data is used for training the embedding and the rest 10% is used for computing the rewards in architecture search. We report the test accuracy using Top-10 retrieval accuracy on the original validation set, which contains 10,844 consumer images with 12,377 query items, and 21,309 commercial images with 36,961 items in the gallery. ResNet-101 is used as the fixed gallery model ϕ_g.

4.2. Implementation Details

Our query and gallery models take a 112×112 image as input and output an embedding vector of 128 dimensions.

Face retrieval: We use mis-alignment and color distortion for data augmentation [28]. Following recent state-of-the-art [36], we train our gallery ResNet-101 model using the cosine margin loss [36] with margin set to 0.4. We use the SGD optimizer with weight decay 5×10^{-4}. The initial learning rate is set to 0.1 which decreases to 0.01, 0.001 and 0.0001 after 8, 12 and 14 epochs. Our gallery model is trained for 16 epochs with a batch size 320. We train the query models for 32 epochs with a cosine learning rate decay schedule [20]. The initial learning rate is set to 1.3 all query models except MobileNetV1(1×) which uses 0.1.

Fashion retrieval: The original fashion retrieval task with DeepFashion2 [7] requires to first detect and then retrieve fashion items. Since we only tackle the retrieval task, we construct our retrieval-only dataset by using ground truth bounding box annotations to extract the fashion items. To train the gallery model, we follow [39] in using normalized cross entropy loss with temperature 0.5. The gallery model is trained for 40 epochs using an initial learning rate of 3.0 with cosine decay. The weight decay is set to 10^{-4}. Our query models are trained with BCT for 80 epochs using an initial learning rate of 10 with cosine decay.

Runtime: On a system containing 8 Tesla V100 GPUs, the entire pipeline for the face (and fashion) retrieval takes roughly 100 (45) hours. This includes roughly 8 (8) hours to train the gallery model, 32 (14) hours to train the query super-network, 48 (20) hours for evolutionary search and 2 (2) hours to train the final query architecture.

For additional implementation details specific to CMP-NAS, please refer to the supplementary material.

4.3. Baseline and paragon for visual search

Since gallery features can be pre-computed and there is no computational constraint on the gallery side, we fix the gallery model to a ResNet-101. In terms of visual search accuracy, the paragon is achieved by the (ResNet-101, ResNet-101) system on both the face and fashion retrieval tasks. On the other hand, we use (MobileNetV2, MobileNetV2) and (MobileNetV3, MobileNetV3) as the baseline for face and fashion retrieval respectively, since they achieve the highest accuracy among the MobileNet family.

	Gallery Model	Query Model	Acc. (%)	Query Flops (M)
Paragon	ResNet-101	ResNet-101	86.7	7597
Proposed	ResNet-101	CMP-NAS	85.1	327
Baseline	MobileNetV2	MobileNetV2	77.1	329

Table 2: Comparison with baseline and paragon for 1:N face retrieval on IJB-C. Accuracy is reported as TPIR(%)@FPIR=10^{-1}. All the models except the paragon are trained with BCT loss using ResNet-101 as the "teacher".

	Gallery Model	Query Model	Acc. (%)	Query Flops (M)
Paragon	ResNet-101	ResNet-101	65.2	7597
Proposed	ResNet-101	CMP-NAS	64.9	211
Baseline	MobileNetV3	MobileNetV3	62.7	226

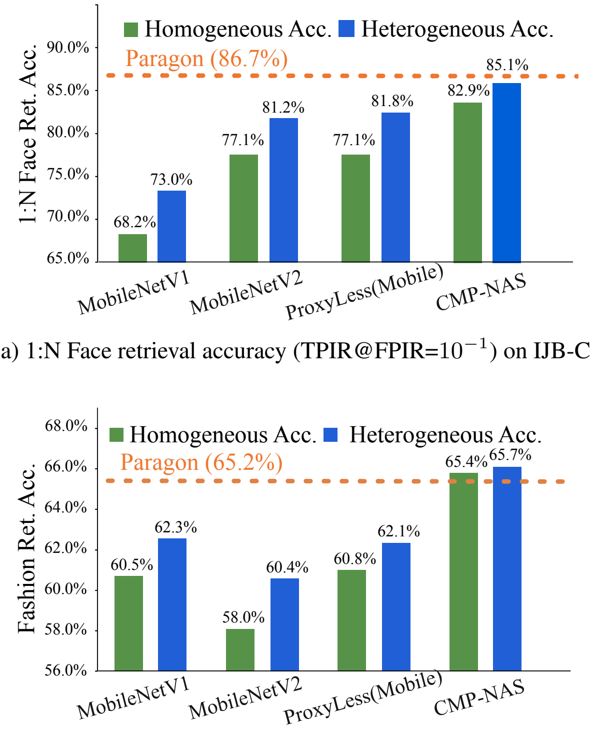
Table 3: Comparison with baseline and paragon for fashion retrieval on DeepFashion2. Accuracy is reported as Top-10 retrieval accuracy. All the models except the paragon are trained with BCT loss using ResNet-101 as the "teacher".

Tab. 2 shows almost a 10% gap in accuracy between the paragon and baseline for face retrieval. On the other end, the baseline consumes 23× fewer query flops than the paragon. This establishes the goal of our heterogeneous system: To achieve accuracy similar to the paragon while consuming query flops similar to the baseline. Indeed, the middle rows of Tab. 2 shows this goal is achieved by our proposed heterogeneous system (ResNet-101, CMP-NAS) which consumes similar query flops as the baseline with only 1.6% accuracy drop compared to the paragon. Tab. 3 reveals a similar observation on DeepFashion2.

4.4. Dissecting the performance of CMP-NAS

In this subsection, we break down the accuracy achieved by our heterogeneous system in terms of the improvement due to (1) weights and (2) architecture compatibility.

Improvement due to weight-compatibility: To observe the improvement due to weight compatibility, Fig. 5 (a),(b) shows the homogeneous and heterogeneous accuracy obtained by three state-of-the-art query models with *static* architectures in the 300 Million flops range trained with BCT (Eq. 1). We observe that heterogeneous system outperforms homogeneous system on average by 3.95% and 1.45% for face and fashion retrieval respectively. This indicates that considering weight-compatibility alone is beneficial.



(a) 1:N Face retrieval accuracy (TPIR@FPIR=10^{-1}) on IJB-C.

(b) Fashion retrieval accuracy (top-10) on DeepFashion2.

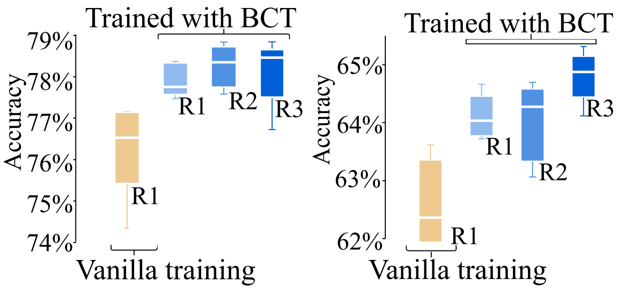
Figure 5: Evaluating the heterogeneous and homogeneous search accuracy for face and fashion retrieval tasks using different query models. CMP-NAS outperforms other baselines and achieves accuracy close to the paragon.

Improvement due to architecture compatibility: To see the additional benefits due to architecture compatibility, in Fig. 5 we compare the heterogeneous accuracy achieved by the query models obtained via CMP-NAS and trained with BCT. On IJB-C and DeepFashion2 datasets, CMP-NAS outperforms the second-best model ProxyLess(Mobile) by 3.3%, 3.6% in terms of heterogeneous accuracy. This shows architecture compatibility can improve accuracy by a large margin. Additionally, we also observe gains of 5.8% and 4.8% in terms of homogeneous accuracy.

Comparing different methods for weight-compatibility: In Sec. 3.2, we discussed four ways to achieve weight-compatibility; (1) vanilla training, (2) knowledge distillation [11], (3) fine-tuning, and (4) BCT [28]. To quantitatively compare these methods, we obtain the query network by pruning the gallery ResNet-101 model using magnitude pruning [16] and channel pruning [10]. To obtain the (pruned) query model, we prune 90% of filters from the first two layers in each residual block of the gallery network. The query model obtained is trained with each of the four methods and Tab. 4 shows both the homogeneous and heterogeneous search accuracy. We observe only fine-tuning and BCT can achieve weight-compatibility wherein accuracy of the heterogeneous system supersedes that of the homogeneous system. Training from scratch and knowledge distillation on the other hand, cannot ensure compatibility and obtain 0.0% accuracy for heterogeneous search. Among fine-tuning and BCT, we prefer BCT to ensure weight compatibility for two reasons: (1) fine-tuning is restrictive: it poses a strong requirement on the query architecture *e.g.* query model is obtained by pruning the gallery model and (2) the model trained with BCT performs better.

Gallery model	Query model	Train Scratch	Finetune	BCT	KD
Magnitude prune	Magnitude prune	84.4	84.9	86.4	86.8
ResNet-101	Magnitude prune	0.0	86.5	87.2	0.0
Channel prune	Channel prune	84.2	85.2	86.5	87.0
ResNet-101	Channel prune	0.0	86.3	87.4	0.0

Table 4: Comparing techniques for achieving weight-level compatibility on the 1:N face retrieval task. The query model ϕ_q is obtain by pruning 90% of filters in the first two layers of each residual block of the gallery model. We see that for both pruning methods, training the query model with BCT loss leads to the highest heterogeneous accuracy.



(a) 1:N search on IJB-C.

(b) Top-10 on DeepFashion2.

Figure 6: Ablating on training strategies (vanilla, BCT) and rewards ($\mathcal{R}_1 - \mathcal{R}_3$) for CMP-NAS These plots show the heterogeneous accuracy of the best 5 models (under 100 Mflops) discovered by each method and trained from scratch with BCT. Observe that the ingredients of CMP-NAS *i.e.* BCT training + reward $\mathcal{R}_3$ perform the best.

Comparing CMP-NAS with baseline NAS[8]: To measure the gains relative to baseline NAS, in Fig. 6, we present a barplot of the heterogeneous accuracy achieved by the best 5 architectures obtained by vanilla NAS (yellow bar) and CMP-NAS (blue bars) when trained from scratch using BCT. The baseline considers the vanilla loss ($\lambda_2 = 0$ in Eq. 1) to train the super-network and searches using reward $\mathcal{R}_1$ while CMP-NAS uses BCT to train the super-network and can search using rewards $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$. On both datasets, CMP-NAS outperforms the baseline by $2 - 2.5\%$.

Comparing reward choices for CMP-NAS: In Fig. 6, the performance of different reward choices (of Tab. 1) are shown by blue plots. As expected, the baseline reward ($\mathcal{R}_1$) performs worst since its target (homogeneous accuracy) is misaligned with our target (heterogeneous accuracy). The

Gallery model	Query model	Query MFlops	Fashion retrieval top-10 search	Face retrieval 1:N search
ResNet-101	ResNet-101	7597	65.1	86.7
ResNet-101	MobileNetV1(1x)	579	62.3	73.0
	MobileNetV2(1x)	329	60.4	77.0
	ProxyLess(mobile)	332	62.1	81.8
	CMP-NAS-a(Face)	**327**	65.4	**85.1**
	CMP-NAS-a(Fashion)	**314**	**65.7**	84.2
ResNet-101	MobileNetV3	226	63.0	80.4
	CMP-NAS-b(Face)	**216**	64.4	**84.1**
	CMP-NAS-b(Fashion)	**211**	**64.9**	81.5
ResNet-101	MobileNetV1(0.5x)	155	60.3	62.7
	ShuffleNetV2(1x)	149	63.3	75.8
	ShuffleNetV1(1x,g=1)	148	62.6	76.0
	MobileNetV2(0.5x)	100	62.0	73.1
	CMP-NAS-c(Face)	**94**	62.4	**78.8**
	CMP-NAS-c(Fashion)	**93**	**64.8**	77.8

Table 5: Evaluating architectures searched with CMP-NAS for fashion retrieval (denoted as fashion) and face retrieval (denoted as face) tasks. We search models for three different complexity tiers: 100, 230 and 330 Mflops and use the best architecture to report the results. The searched models outperform other architectures by 3 $\sim$ 5% on both the tasks.

second reward ($\mathcal{R}_2$) is much better since it directly optimizes the target while the composite reward ($\mathcal{R}_3$) works best with especially large gains observed on DeepFashion2.

4.5. Generalization performance of CMP-NAS

In this section, we investigate the performance of CMP-NAS under different compute constraints, application scenarios and tasks. Inspired by state-of-the-art architectures for mobile deployment we select three computational tiers: 330 million flops (similar to MobileNetV2), 230 Mflops (similar to MobileNetV3), and 100 Mflops (similar to ShuffleNetV2). For each computational tier, we implement a heterogeneous system using the models searched by CMP-NAS. These models are denoted by CMP-NAS-a (330 Mflops), CMP-NAS-b (230 Mflops), and CMP-NAS-c (100 Mflops). Additionally, we append "(Face)"/"(Fashion)" to the model name, *e.g*. "CMP-NAS-a(Face)"/"CMP-NAS-a(Fashion)", to denote the architecture searched on the face or fashion datasets respectively.

CMP-NAS for different resource constraints: We compare the performance of architectures searched by CMP-NAS for each computational tier in Tab. 5. For each task, we look at the model searched on the same task *e.g*. for face retrieval we look at CMP-NAS-a(Face) *etc*. On both datasets the models searched by CMP-NAS consistently outperform other state-of-the-art baselines. For 330 Mflops tier, CMP-NAS-a outperforms the second best (ProxyLess(Mobile) [4]) by 3.6% and by 3.1% on the fashion and face retrieval tasks respectively. Similarly, CMP-NAS-b outperforms the second best network MobileNetV3 [12] by 1.9% and 3.7% on the corresponding tasks. Finally, for the 100M tier, CMP-NAS-c achieves 1.5% and 3.0% improvement over the second best network ShuffleNetV2(1×) while consuming 33% fewer flops. These results establish the generalization ability of the CMP-NAS for different computation constraints.

Gallery model	Query model	Query MFlops	Homogeneous accuracy	Heterogeneous accuracy
ResNet-101	ResNet-101	7597	85.4	-
ResNet-101	ProxyLess(mobile)	332	75.5	80.3
	CMP-NAS-a(Face)	**327**	**81.6**	**84.5**
ResNet-101	MobileNetV3	226	74.3	79.9
	CMP-NAS-b(Face)	**216**	**79.0**	**82.8**
ResNet-101	ShuffleNetV2(1x)	149	66.8	74.8
	CMP-NAS-c(Face)	**94**	**71.5**	**78.3**

Table 6: Evaluating the models CMP-NAS-a,b,c(Face) on the 1:1 face verification task using IJB-C. Accuracy metric is TAR@FAR=10^{-4}. The searched models outperform the baselines indicating they can generalize across tasks.

CMP-NAS across different applications: For this experiment, we evaluate the architectures searched on the face dataset for the fashion retrieval task and vice versa. The results are shown in the Tab. 5. We observe that the architectures optimized for the face tasks CMP-NAS-a/b/c(Face) also outperform the baselines for the fashion retrieval task. Moreover, these models only lose $1-2\%$ accuracy compared to the best model searched on the fashion dataset. We make similar observations for CMP-NAS-a/b/c/(Fashion) evaluated on the face retrieval task. This shows that the architectures searched by CMP-NAS can generalize across application scenarios.

CMP-NAS for face verification: In table Tab. 6, we use the CMP-NAS-a/b/c(Face) models for another "open universe" problem: 1:1 face verification. The results show that the models searched by CMP-NAS outperform state-of-the art architectures by $3-5\%$ in the homogeneous and heterogeneous settings. Importantly, the compatibility rule is also achieved. This indicates the searched models can generalize across different tasks.

5. Discussion

We have presented a heterogeneous visual search system that achieves high accuracy with low computational cost. Key to building this system is ensuring the query and gallery models are compatible. We achieve this through joint weight and architecture compatibility optimization with CMP-NAS. There are, however, some limitations of our method: (1) Our method is limited to classification-based embedding training and does not directly work with metric learning based approaches; (2) We consider the simplified use-case for architecture optimization wherein the gallery model is fixed. A more general treatment of model compatibility may optimize both the gallery and query models. These limitations show that there is scope for improving our HVS system which can be tackled by future work.

References

[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] Artem Babenko and Victor Lempitsky. Aggregating deep convolutional features for image retrieval. *arXiv:1510.07493*, 2015.

[3] Mateusz Budnik and Yannis Avrithis. Asymmetric metric learning for knowledge transfer. *arXiv:2006.16331*, 2020.

[4] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

[5] Ken Chen, Yichao Wu, Haoyu Qin, Ding Liang, Xuebo Liu, and Junjie Yan. R3 adversarial network for cross model face recognition. In *CVPR*, 2019.

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[7] Yuying Ge, Ruimao Zhang, Xiaogang Wang, Xiaoou Tang, and Ping Luo. Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In *CVPR*, 2019.

[8] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[10] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.0253*, 2015.

[12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *ICCV*, 2019.

[13] Jie Hu, Rongrong Ji, Hong Liu, Shengchuan Zhang, Cheng Deng, and Qi Tian. Towards visual feature translation. In *CVPR*, 2019.

[14] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *ECCV*, 2016.

[15] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Blockwisely supervised neural architecture search with knowledge distillation. In *CVPR*, 2020.

[16] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2019.

[18] Yu Liu, Xuhui Jia, Mingxing Tan, Raviteja Vemulapalli, Yukun Zhu, Bradley Green, and Xiaogang Wang. Search to distill: Pearls are everywhere but not the eyes. In *CVPR*, 2020.

[19] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *ICLR*, 2019.

[20] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[21] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018.

[22] B. Maze, J. Adams, J. A. Duncan, N. Kalka, T. Miller, C. Otto, A. K. Jain, W. T. Niggel, J. Anderson, J. Cheney, and P. Grother. Iarpa janus benchmark - c: Face dataset and protocol. In *ICB*, 2018.

[23] Mira Park, Jesse S Jin, and Laurence S Wilson. Fast content-based image retrieval using quasi-gabor filter and reduction of image feature dimension. In *Proceedings fifth IEEE southwest symposium on image analysis and interpretation*, 2002.

[24] Adnan Qayyum, Syed Muhammad Anwar, Muhammad Awais, and Muhammad Majid. Medical image retrieval using deep convolutional neural network. *Neurocomputing*, 266:8–20, 2017.

[25] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *TPAMI*, 41(7):1655–1668, 2018.

[26] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *TMTA*, 4(3):251–258, 2016.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[28] Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. Towards backward-compatible representation learning. In *CVPR*, 2020.

[29] Christian Siagian and Laurent Itti. Rapid biologically-inspired scene classification using features shared with visual attention. *TPAMI*, 29(2):300–312, 2007.

[30] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *ICLR*, 2019.

[31] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv:1511.05879*, 2015.

[32] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[33] Chien-Yi Wang, Ya-Liang Chang, Shang-Ta Yang, Dong Chen, and Shang-Hong Lai. Unified representation learning for cross model compatibility. In *BMVC*, 2020.

[34] Fei Wang, Liren Chen, Cheng Li, Shiyao Huang, Yanjie Chen, Chen Qian, and Chen Change Loy. The devil of face recognition is in the noise. In *ECCV*, 2018.

[35] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *ACM Multimedia*, 2017.

[36] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, 2018.

[37] Christian Wengert, Matthijs Douze, and Hervé Jégou. Bag-of-colors for improved image search. In *ACM Multimedia*, 2011.

[38] Lingxi Xie, Richang Hong, Bo Zhang, and Qi Tian. Image classification and retrieval are one. In *ICMR*, 2015.

[39] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. In *BMVC*, 2019.

[40] Liang Zheng, Shengjin Wang, Ziqiong Liu, and Qi Tian. Packing and padding: Coupled multi-index for accurate image retrieval. In *CVPR*, 2014.

[41] Liang Zheng, Shengjin Wang, Lu Tian, Fei He, Ziqiong Liu, and Qi Tian. Query-adaptive late fusion for image search and person re-identification. In *CVPR*, 2015.

[42] Wengang Zhou, Yijuan Lu, Houqiang Li, and Qi Tian. Scalar quantization for large scale image search. In *ACM Multimedia*, 2012.

[43] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.