Appendix

In the appendix, we first provide the details of the uniform encoding strategy. We then give a detailed proof of the Theorem 2. In Section C, we illustrate the searched architectures with figures. Supplementary experimental details and more ablation studies are provided in Section D and E respectively.

A Details of the Uniform Encoding Strategy

In practice, the encoding strategies of different architecture spaces are required to be unified for implementing the cross-domain predictor. Taking NAS-Bench-101, NAS-Bench-201, and DARTS as examples, although these three search spaces all use cells to make up architectures, the representation methods of architectures in different search spaces vary greatly. In addition, the original encoding strategies for architectures in different search spaces are also quite different, so the progressive subspace adaptation cannot be carried out directly.

To solve this issue, we propose a uniform encoding strategy that consists of four main steps. These steps unify the encoding from four aspects, including the number of input, operation location, mapping of operations, and cell types. After these four main steps, all the cells in NAS-Bench-201 and DARTS can be converted to the formulation of NAS-Bench-101. And then, we will introduce them in detail next.

The number of inputs: We firstly separate the cell having two inputs in DARTS into two cells with a single input of each. An example is given based on Fig. 4 (c), and we will introduce how to separate it into two cells in detail. Supposing that the first cell contains 'In_1', then the operations only connected to node 'In_2' need to be deleted in the first cell. A similar process will be carried out in the second cell. This step unifies the number of 'input' nodes in different cells and ensures that there is only one input in each cell. At the same time, this step can reduce the number of operations in cells of DARTS and make it more similar to the smaller cells in NAS-Bench-101 and NAS-Bench-201.

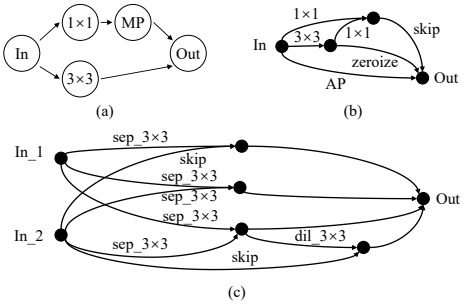


Figure 4: Examples in three search spaces. The subfigure (a): a cell in NAS-Bench-101, (b): a cell in NAS-Bench-201, (c): a cell in DARTS. 'In' denotes the input node, and 'Out' denotes the output node. Different operations are represented by 1×1, 3×3, *etc.* 1×1 and 3×3 denote for 1×1 and 3×3 convolution operations respectively. MP and AP are short for max pooling and average pooling.

Operation location: Another obvious difference is the operation location. According to Fig. 4, the operations are represented by vertices in NAS-Bench-101, while are represented by edges in the remaining two search spaces. In addition, the skip connection and the zeroize operation will not appear in the form of NAS-Bench-101 because they are both implicitly represented by the edges, *i.e.*, the connections. This step is to convert the cells to the formulation of NAS-Bench-101 because it can be more intuitively converted into adjacency matrix M and operation feature O_0 which are the inputs of CDP. Because the conversion methods of NAS-Bench-201 and DARTS are almost the same, we only introduce how to convert the cell in NAS-Bench-201 to the form of NAS-Bench-101. The vertices in cells of NAS-Bench-201 are densely connected since the vertices in the back may receive the connection from all the vertices in the front. Then we specify an order for these six edges. After that, the template adjacency matrix M_t, which should be the same in all cells, can be obtained by regarding the edges as vertices. Finally, the adjacency matrix is pruned according to skip connection and zeroize operation.

Mapping of operations: One intractable issue is how to encode the feature of operations caused by the various types of operations. We hope to map the operation types in the target architecture space and the source architecture space as much as possible and as accurately as possible, and try to manually divide similar operations into one category based on their characteristics. Firstly, there are two types of pooling operations, namely max pooling and average pooling. We make their feature

encoding the same because of their similar functions. Secondly, there are separable convolutions and dilated separable convolutions with kernel sizes of 3 and 5 in DARTS. Fortunately, 3×3 convolutions are utilized in both NAS-Bench-101 and NAS-Bench-201. According to the above principle, the convolutions with the same kernel size are classified into one category. However, we cannot find any suitable operation corresponding to the 5×5 convolutions in the source architecture space. One solution is to map the 5×5 convolution to the 1×1 convolution which is also isolated. But their roles in the neural network are too far apart, so we do not choose this solution. Finally, we define these operations as *unknown* type, and the ablation study provided in the manuscript proves the benefits of doing so.

Cell types: There are two types of cells in DARTS, namely normal cell and reduction cell. In the end, one architecture in DARTS will be represented by totally 4 cells multiple by 2 cells which are separated from the first step. The weighted sum of these four cell scores is used as the performance of the architecture. We use the weighted sum score of these four cells as the performance value of the architecture. Considering the contribution of different types of cells to the entire architecture, four neural predictors are trained with different datasets split from the whole training datasets of NAS-Bench-101 and NAS-Bench-201. The size of the training dataset is proportional to the proportion of the cell type predicted by the neural predictor in the architecture.

After the above four main steps, all the cells in NAS-Bench-201 and DARTS can be converted to the form of NAS-Bench-101. The last step is to pad zeros to ensure the sizes of all the adjacency matrices are the same. Following the conventions in [57], the positions of the zero-padding are in the second to last row and second to last column. In addition, we use one-hot encoding to encode the operations, namely *input*, *output*, *convolution* 1×1, *convolution* 3×3, *pooling*, and *zeroize*. And the *unknown* operation is encoded with zeros.

B Proofs

B.1 Proof of Lemma 1.

Lemma 1. *Let $\epsilon_T(h)$ and $\epsilon_S(h)$ be the expected error on target and source domain and $\mathcal{H}$ be a hypothesis space, for $h \in \mathcal{H}$:*

$$\epsilon_T(h) \leq \epsilon_S(h) + d_{\mathcal{H}}(\widetilde{\mathcal{D}}_S, \widetilde{\mathcal{D}}_T) + \lambda, \tag{14}$$

where $\lambda = \epsilon_T(h^*) + \epsilon_S(h^*)$ is the combined error of ideal hypothesis $h^* = \arg\min_{h \in \mathcal{H}}(\epsilon_T(h) + \epsilon_S(h))$ on both domains.

Proof. We first introduce the $\mathcal{A}$-distance which can measure the difference between probability distributions. The formulation is shown as follows:

$$d_{\mathcal{A}}(\mathcal{D}, \mathcal{D}') = 2 \sup_{A \in \mathcal{A}} |\Pr_{\mathcal{D}}[A] - \Pr_{\mathcal{D}'}[A]|. \tag{15}$$

We follow the conventions in domain adaptation to assume a binary classification scenario, where $\mathcal{Y} = \{0, 1\}$. For a binary hypothesis space $\mathcal{H}$, we use $d_{\mathcal{H}}$ to indicate the $\mathcal{A}$-distance.

Let $\mathcal{Z}_h = \{\mathbf{z} \in \mathcal{Z} : h(\mathbf{z}) = 1\}$ represents the set of latent representations in the latent space $\mathcal{Z}$ that are classified as category 1 by h, $h^* = \arg\min_{h \in \mathcal{H}}(\epsilon_T(h) + \epsilon_S(h))$, and we have the following inequalities:

$$\epsilon_T(h) \leq \epsilon_T(h^*) + \Pr_{\mathcal{D}_T}[\mathcal{Z}_h \Delta \mathcal{Z}_{h^*}], \tag{16}$$

$$\Pr_{\mathcal{D}_S}[\mathcal{Z}_h \Delta \mathcal{Z}_{h^*}] \leq \epsilon_S(h) + \epsilon_S(h^*), \tag{17}$$

where Δ is the XOR operation. The reason why inequality (16) holds is as follows. The first term after the inequality sign $\epsilon_T(h^*)$ is the error rate of h^*, including the error when the hypotheses of h, h^* are in agreement, and the second term is the error probability when the hypotheses of h, h^* are in disagreement. Thus, the expected error on the target domain is less than the sum of these two terms. The reason why inequality (17) holds as follows. The first term $\Pr_{\mathcal{D}_S}[\mathcal{Z}_h \Delta \mathcal{Z}_{h^*}]$ denotes the hypotheses of h, h^* are in disagreement, *i.e.*, $h(\mathbf{z}) \oplus h^*(\mathbf{z}) = 1, \oplus : XOR$. At this time, only one of the hypotheses is wrong, so it is less than the sum of the errors of h and h^*.

Let $\lambda = \epsilon_T(h^*) + \epsilon_S(h^*)$, and we can have:

$$\begin{aligned}
\epsilon_T(h) &\leq \epsilon_T(h^*) + \Pr_{\mathcal{D}_T}[\mathcal{Z}_h \Delta \mathcal{Z}_{h^*}] \\
&\leq \epsilon_T(h^*) + \Pr_{\mathcal{D}_S}[\mathcal{Z}_h \Delta \mathcal{Z}_{h^*}] + |\Pr_{\mathcal{D}_S}[\mathcal{Z}_h \Delta \mathcal{Z}_{h^*}] - \Pr_{\mathcal{D}_T}[\mathcal{Z}_h \Delta \mathcal{Z}_{h^*}]| \\
&\leq \epsilon_T(h^*) + \Pr_{\mathcal{D}_S}[\mathcal{Z}_h \Delta \mathcal{Z}_{h^*}] + d_{\mathcal{H}}(\widetilde{\mathcal{D}}_S, \widetilde{\mathcal{D}}_T) \\
&\leq \epsilon_T(h^*) + \epsilon_S(h) + \epsilon_S(h^*) + d_{\mathcal{H}}(\widetilde{\mathcal{D}}_S, \widetilde{\mathcal{D}}_T) \\
&\leq \epsilon_S(h) + d_{\mathcal{H}}(\widetilde{\mathcal{D}}_S, \widetilde{\mathcal{D}}_T) + \lambda.
\end{aligned}$$

□

B.2 Proof of Theorem 2

Theorem 2. *Let d' be the VC-dimension of $\mathcal{H}'$, m be the size of $\widetilde{\mathcal{U}}_{S,valid}$, m' be the size of unlabeled samples $\widetilde{\mathcal{U}}_S$ and $\widetilde{\mathcal{U}}_T$. With probability of $1-\delta$, for $h \in \mathcal{H}'$:*

$$\begin{aligned}
\epsilon_T(h) \leq \hat{\epsilon}_{S,valid}(h) + 2d_k(\widetilde{\mathcal{D}}_S, \widetilde{\mathcal{D}}_T) + \frac{2(d' \log m - \log \delta)}{3m} + \sqrt{\frac{2(d' \log m - \log \delta)}{m}} \\
+4\sqrt{\frac{d' \log(2m') + \log(\frac{4}{\delta})}{m'}} + 2 + \lambda.
\end{aligned} \tag{18}$$

Proof. Firstly, we found an upper bound represented by empirical error on source validation dataset $\hat{\epsilon}_{S,valid}(h)$ for the expected error on source domain $\epsilon_S(h)$ in Equation (14). Following the proof in [32], let $\kappa_i = \epsilon_S(h) - \mathcal{L}(h(\mathbf{z}_i), y_i)$ for $h \in \mathcal{H}$ and $\mathbf{z}_i \in \widetilde{\mathcal{U}}_{S,valid}$, and m is the size of $\widetilde{\mathcal{U}}_{S,valid}$. We have:

$$\epsilon_S(h) - \hat{\epsilon}_{S,valid}(h) \leq \frac{1}{m} \sum_{i=1}^{m} \kappa_i(h). \tag{19}$$

As $0 \leq \epsilon_S(h) \leq 1$ and $0 \leq \mathcal{L}(h(\mathbf{z}_i), y_i) \leq 1$, we can derive the boundary $-1 \leq \kappa_i \leq 1$ and $\mathbb{E}[\kappa_i(h)^2] \leq 1$, $|\kappa_i| \leq 1$. By using Bernstein inequality, we have:

$$\Pr\left(\frac{1}{m} \sum_{i=1}^{m} \kappa_i(h) > \xi\right) \leq \exp\left(-\frac{\xi^2 m}{2(1 + \frac{\xi}{3})}\right). \tag{20}$$

Considering union bound over all $h \in \mathcal{H}$, Equation (21) can be derived from Equation (20) with VC-dimension d' of $\mathcal{H}$:

$$\Pr\left(\cup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \kappa_i(h) > \xi\right) \leq m^{d'} \exp\left(-\frac{\xi^2 m}{2(1 + \frac{\xi}{3})}\right). \tag{21}$$

Let $\delta = m^{d'} \exp\left(-\frac{\xi^2 m}{2(1+\frac{\xi}{3})}\right)$ and solve the Equation (21) for ξ:

$$\xi = \frac{d' \log m - \log \delta}{3m} \pm \sqrt{\left(\frac{d' \log m - \log \delta}{3m}\right)^2 + \frac{2(d' \log m - \log \delta)}{m}}. \tag{22}$$

Because $\xi \geq 0$ and $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, we can simplify the Equation (22) as:

$$\xi \leq \frac{2(d' \log m - \log \delta)}{3m} + \sqrt{\frac{2(d' \log m - \log \delta)}{m}}. \tag{23}$$

Therefore, with a probability of at least $1-\delta$, for $h \in \mathcal{H}$:

$$\begin{aligned}
\epsilon_S(h) - \hat{\epsilon}_{S,valid}(h) &\leq \frac{2(d' \log m - \log \delta)}{3m} + \sqrt{\frac{2(d' \log m - \log \delta)}{m}} \\
\epsilon_S(h) &\leq \hat{\epsilon}_{S,valid}(h) + \frac{2(d' \log m - \log \delta)}{3m} + \sqrt{\frac{2(d' \log m - \log \delta)}{m}}.
\end{aligned} \tag{24}$$

Furthermore, by enlarging the expected domain distance to its upper bound with the empirical domain distance measured by finite samples according to [3], we have:

$$\begin{aligned}\epsilon_T(h) &\leq \hat{\epsilon}_{S,valid}(h) + \frac{2(d' \log m - \log \delta)}{3m} + \sqrt{\frac{2(d' \log m - \log \delta)}{m}} + d_{\mathcal{H}}(\widetilde{\mathcal{D}}_S, \widetilde{\mathcal{D}}_T) + \lambda \\ &\leq \hat{\epsilon}_{S,valid}(h) + \frac{2(d' \log m - \log \delta)}{3m} + \sqrt{\frac{2(d' \log m - \log \delta)}{m}} \\ &\quad + \hat{d}_{\mathcal{H}}(\widetilde{\mathcal{U}}_S, \widetilde{\mathcal{U}}_T) + 4\sqrt{\frac{d' \log(2m') + \log(\frac{4}{\delta})}{m'}} + \lambda. \end{aligned} \tag{25}$$

Secondly, we found an upper bound represented by MMD for the distance $\hat{d}_{\mathcal{H}}(\widetilde{\mathcal{U}}_S, \widetilde{\mathcal{U}}_T)$ in Equation (25). Following the assumption in [35], we also choose Parzen window classifier [47] as h, and the empirical distance can be bounded by:

$$\begin{aligned}\hat{d}_{\mathcal{H}}(\widetilde{\mathcal{U}}_S, \widetilde{\mathcal{U}}_T) &\leq 2\left(1 - \inf_{h \in \mathcal{H}} \sum_{i=1}^{m'} \frac{L[h(\mathbf{z}_i^s) = 1] + L[h(\mathbf{z}_i^t) = -1]}{m'}\right) \\ &= 2 + 2d_k(\widetilde{\mathcal{D}}_S, \widetilde{\mathcal{D}}_T), \end{aligned} \tag{26}$$

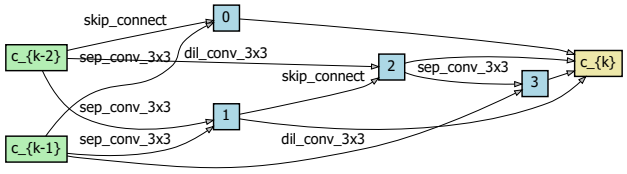
where $L(\cdot)$ is the loss function of Parzen window classifier.

Finally, by applying the bound between $\hat{d}_{\mathcal{H}}(\widetilde{\mathcal{U}}_S, \widetilde{\mathcal{U}}_T)$ and $d_k(\widetilde{\mathcal{D}}_S, \widetilde{\mathcal{D}}_T)$ on Equation (25), we can have Equation (18).

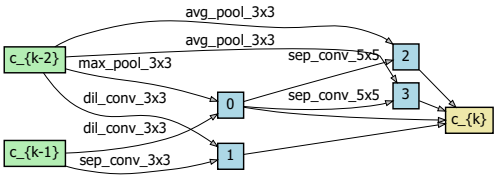
□

C Searched Architectures

We use four figures to display the searched architectures. Specifically, the searched normal cell and reduction cell on ImageNet are shown in Fig. 5. While the searched cells on CIFAR-10 are shown in Fig. 6.



(a)

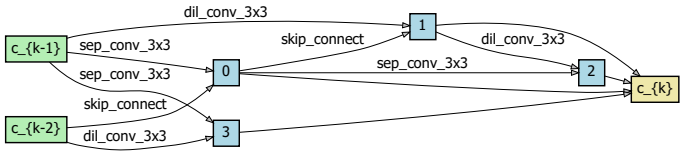


(b)

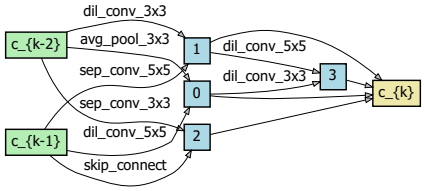
Figure 5: Searched normal cell (top) and reduction cell (bottom) on ImageNet.

D Supplementary Experimental Details

This section is a supplement to the experimental details in the manuscript.



(a)



(b)

Figure 6: Searched normal cell (top) and reduction cell (bottom) on CIFAR-10.

Settings of GCN: GCN is set up to fully utilize GPU memory and computing power. Specifically, the training batch size is set as 1K and the learning rate is 2e-3. In addition, because the neural architectures in multiple spaces are complex, we set the number of hidden layers as 5. The other parameters remain the same as suggested in [55].

Training settings of the searched architectures: Based on the convention, we also follow the positive training methods suggested in [2] to improve the classification accuracy on ImageNet. Specifically, the total training epoch is set as 350. The cosine learning rate with the decay 4e-5 is used as the optimizer. Regularization methods such as labeling smoothing [51] and AutoAugment [9] are also adopted. In addition, the squeeze-and-excitation module [24] is attached after each cell. Furthermore, the training batch size on ImageNet is set as 512, and the learning rate is 0.2. As for the training details on CIFAR-10, we follow the conventions of the previous works [33, 58].

E More Ablation Studies

This section is a supplement to the ablation study of the manuscript.

The numbers of categories: We first conduct an ablation study on the numbers of categories C in LMMD, and the results are shown in Fig. 7. As discussed in Subsection 3.2, when C is equal to 1, LMMD is mathematically equivalent to MMD. Because of the generalizability of MMD, not surprisingly, the predictor has been improved. Then we increase C to display the effect of subspace adaptation. As can be seen from Fig. 7, LMMD works better than MMD when C is set as 2. However, when C is increased to 3, the prediction performance does not improve but exacerbates. Further increasing C to 4 results in continued deterioration of performance. This phenomenon may be caused by the incorrect pseudo-labels discussed in Subsection 3.2.

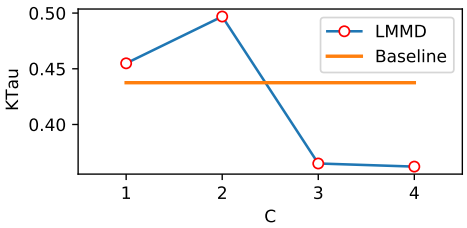


Figure 7: The performance of the predictor under different numbers of categories C in LMMD, where C denotes the number of subspaces. The baseline denotes the KTau value without LMMD.

The hyper-parameter K of Equation (9): We decide the maximum number of subspaces K by an ablation study. Table 6 shows increasing K from 2 to 3, the performance is improved. But increasing K to 4, the performance is worse. This is because when K increases, the probability that the predicted

Table 6: Ablation study on K.

	$K=2$	$K=3$	$K=4$
Cost (GPU minutes)	55	72	85
KTau	0.4921	**0.5306**	0.5172

pseudo-label is wrong will also increase. In addition, when K increases, the search cost will also increase. So we set K to 3.

The impact of kernel ϕ: We have tried three different ϕ, *i.e.*, Rational Quadratic Kernel (RQK), Laplace kernel, and Gaussian kernel. As can be seen from Table 7, the Gaussian kernel is significantly better than the other two in terms of KTau. And this is why we use it.

Table 7: The impact on kernel ϕ.

	RQK	Laplace	Gaussian
KTau	0.4135	0.4698	**0.5306**

Other improvements: As shown in Table 8, we divided the improvements into two parts. The first part is the baseline where no improvements are applied. The second part includes three tricks. The first trick is to normalize the performance values in NAS-Bench-101 and NAS-Bench-201 separately. A great improvement is obtained in terms of KTau value. On this basis, we continue to fine-tune the penalty parameter θ in Equation (4) of the manuscript. In the end, $\theta = \frac{2}{1+exp\{-10\frac{e}{E}\}} - 1$ after fine-tuning, and a smaller improvement can be observed. On the basis of these two tricks, we use the *unknown* operation to represent 5×5 convolution which is explored in the uniform encoding strategy. As can be seen from Table 8, this trick also brings positive effects. Therefore, we used these three tricks throughout the experiments introduced in the manuscript.

Table 8: Ablation study on shallow DARTS dataset. Δ denotes the difference from the previous improvement.

Improvement	KTau	Δ
Baseline	0.3403	—
Separate normalization	0.4079	+ 0.0676
Fine-tuning θ	0.4107	+ 0.0028
Unknown operation	0.4375	+ 0.0268