Privacy-preserving Online AutoML for Domain-Specific Face Detection Supplementary Material

Chenqian Yan[1†*] Yuge Zhang[1†] Quanlu Zhang[1] Yaming Yang[1]
Xinyang Jiang[1] Yuqing Yang[1] Baoyuan Wang[2]
Microsoft Research[1] Xiaobing.ai[2]
im.cqyan@gmail.com,{yugzhan,quzha,yayaming,xinyangjiang,yuqyang}@microsoft.com,wangbaoyuan@xiaobing.ai

A. Pseudo-code for HyperFD

We summarize the flow of the full algorithm as follows.

Algorithm 1 HyperFD

Input: Search space of configurations $\mathcal{C}$. Offline prepared datasets $\mathcal{D}_{\text{offline}}$. A sequence of online datasets $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$. Detector training and evaluation pipeline $\text{AP}(c, d)$. Budget for each task B. Batches to update for each task N_{iters}. Iterations of updates of transformation module N_{trans}. Learning rates η_z and η_θ.
Output: Best configurations $\{c_1^*, c_2^*, \ldots, c_N^*\}$.

▷ *Warm-up*
$S_{\text{offline}} \leftarrow \{(c, d, \text{AP}(c, d)) \mid c \in \mathcal{C}, d \in \mathcal{D}_{\text{offline}}\}$.
$\theta \leftarrow \arg\min_\theta \mathcal{L}_{\text{total}}(F; S_{\text{offline}})$.
$S_{\text{past}} \leftarrow \{\}$.
$\theta^{(0)} \leftarrow \theta$.

for $t = 1, 2, \ldots, N$ **do**
  ▷ *Inference*
  Get top-B configs $C_t = \{c_1, c_2, \ldots, c_B\}$ with F.
  $c_t^* \leftarrow \arg\max_i \text{AP}(C_{t,i}, d_t)$. ▷ *Costly step*

  ▷ *Training*
  $S \leftarrow \{(c, d_t, \text{AP}(c, d_t) \mid c \in C_t\}$
  for $i = 1, 2, \ldots, N_{\text{iters}}$ **do**
    ▷ *Training of transformation module*
    for $k = 1, 2, \ldots, N_{\text{trans}}$ **do**
      for $u = 1, 2, \ldots, t-1$ **do**
        $\mathbf{Z}^{(u)} \leftarrow \mathbf{Z}^{(u)} - \eta_z \nabla_{\mathbf{Z}^{(u)}} \mathcal{L}_{\text{trans}}(\mathbf{Z}^{(u)}; \mathcal{D}_{\text{offline}})$.
      end for
    end for
    ▷ *Training of performance ranker*
    $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}_{\text{total}}(F; S \bigcup S_{\text{past}})$.
  end for

  $\theta^{(i)} \leftarrow \theta$.
  $S_{\text{past}} \leftarrow S_{\text{past}} \bigcup \{(c, \phi_t, \text{AP}(c, d_t) \mid c \in C_t\}$.
end for

B. Experiment setups

B.1. Detector training

We use RetinaFace [9] with MobileNet-V2 [30] (channels $\times 0.5$ variant). The channels of FPN and context modules are set to 80. The backbone is firstly pre-trained on ImageNet [7], and then the full detector is further pre-trained on WIDER-Face [36] to boost the performance of the model, especially on small datasets like PASCAL [11]. For training, we set batch size to 32, image size to 640×640 (after crop), and uses LeakyReLU as activation functions. We perform a validation per 10 epochs of training, and we adopt a "Reduce LR on Plateau" policy that decays the learning rate by 10 when metric on validation set stagnates in the 5 past evaluations. The maximum epochs of training is 1000, but we stop the training when the validation performance no longer increases for 8 times. For evaluation, we follow [9, 22, 35] to rescale the shorter side of images to 720 pixels, ignore bounding boxes smaller than 36 pixels, and perform Non-maximum Suppression (NMS) on overlap predictions with a 0.4 IoU threshold. We select the best model in history and evaluate it on test dataset. We use Average-Precision at IoU 0.5 (AP@50) as our evaluation metric, with the evaluation scheme implemented in MMDetection [5].

B.2. Detection datasets

We gather 12 public datasets for our evaluation, which includes AFLW [24], Anime [27], FaceMask [16], FDDB [17], FDDB-360 [13], MAFA [15], Pascal VOC [11], UFDD [25], UMDAA-02 [23], WIDER-Face [36], WIDER-360 [14], WIKI [29]. The datasets span multiple categories, including faces with masks, anime

*Work done as an intern at MSRA. † Equal contribution.

Dataset	# Images			# Faces	# Faces
	Train	Val	Test	per image	w. landm.
AFLW [24]	12477	2079	6239	1.5	79%
Anime [27]	3703	617	1851	1.2	None
FaceMask [16]	2554	425	1278	1.7	None
FDDB [17]	1693	282	846	1.8	None
FDDB-360 [13]	4867	811	2433	1.6	None
MAFA [15]	23261	2585	4927	1.0	None
Pascal VOC [11]	511	85	255	1.9	None
UFDD [25]	1781	296	892	3.7	None
UMDAA-02 [23]	16934	1881	4708	1.0	None
WIDER-Face [36]	9665	1600	4832	12.2	80%
WIDER-360 [14]	38321	6394	19145	7.3	None
WIKI [29]	20941	3490	10471	1.2	None

Table 1. Statistical information of all datasets.

faces, faces from fisheye cameras, selfies from cellphones and etc. A preview of the datasets is shown in Figure 1. Apart from cleanup for illegal bounding boxes and corrupted images (particularly on WIKI, WIDER-Face, and UMDAA-02), the quality of annotations on AFLW and WIKI is particularly low, as they are designed for other vision tasks (*e.g.* facial landmarks and face recognition). Thus, we only consider the original ground truth bounding boxes as a reference and use RetinaFace [9] to ensure the quality of weak labels with bipartite matching towards old labels. Specifically, we replace the original ground truth labels with the weak labels if IoU threshold > 0.4 by bipartite matching. For unmatched labels, we mark the confidence > 0.95 as positive. For hard cases that detectors have little confidence, we manually checked each of them. We also unify the format of facial landmarks to 5 points (*i.e.* eyes, noses, mouths), preserving the landmark annotations in AFLW (downsize the 19 landmarks) and WIDER-Face. Finally, we split each dataset into three splits: train, val and test, in the ratio of 6:1:3. This split is fixed and we will use it in all our following experiments. The overall statistics of all processed datasets are provided in Table 1.

B.3. Search space

The hyper-parameter search space (HPO space) is shown in Table 2.

The neural architecture search space (NAS space) is shown in Table 3. The space is essentially a channels $\times 0.5$ variant of ProxylessNAS search space [3]. For detection purposes, we extract feature maps from the end of stage 3, 5 and 7, with down-sampling by 8, 16 and 32 respectively. To benefit from pre-training, we use parameter-remapping [12], where we train the largest network in the search space first, and map the weights to the target network with a set of rules.

Optimizer	{ SGD, Adam }
Learning rate	$\{3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-2}\}$
Min crop ratio	$\{0.3, 0.55\}$
IoU threshold	$\{0.4, 0.5, 0.6\}$
Location loss weight	$\{2, 4, 8\}$
Neg-pos samples ratio	$\{2, 7\}$

Table 2. HPO space contains 216 combinations of hyper-parameters. For IoU threshold, we show the positive matching threshold in the table, while the negative threshold $=$ pos$/2+0.05$.

Stage	Depth	Expand ratio	Kernel size	Width	Stride
1	1	1	3	8	1
2	1–3	$\{4, 6\}$	$\{3, 5, 7\}$	12	2
3	1–3	$\{4, 6\}$	$\{3, 5, 7\}$	16	2
4	2–4	$\{4, 6\}$	$\{3, 5, 7\}$	32	2
5	3–4	$\{4, 6\}$	$\{3, 5, 7\}$	48	1
6	2–4	$\{4, 6\}$	$\{3, 5, 7\}$	80	2
7	1	$\{4, 6\}$	$\{3, 5, 7\}$	160	1

Table 3. Search space for neural architectures.

Task	Model	Layer
Classification	EfficientNet [32]	Last hidden layer
Classification	IBN-Net [26]	Last hidden layer
Classification	Places365 [18]	Last hidden layer
Detection	DETR [4]	Transformer encoder
Detection	RetinaNet [21]	FPN
Detection	CenterFace [35]	Before heads
Face recognition	ArcFace [8] †	Final layer
Face emotion	FerPlus [1]	Last hidden layer
Face age	VGG16-Age [20]	Final layer
Face gender	VGG16-Gender [20]	Last hidden layer
Label statistics ‡	–	–

Table 4. List of pre-trained models used to generated features for augmentation. †: For ArcFace, we have two variants which feed the full image and the dominant face crop respectively. ‡: Areas of bounding boxes on the image, described by `scipy.describe`.

B.4. Dataset augmentation

Dataset augmentation intends to create a large number of diverse datasets. To this end, we first extract features for each image using pre-trained models with diverse usages collected from model zoos (*e.g.*, ONNX model zoo[1]). We use intermediate layers of features so that fine-grained distribution information is not lost. Table 4 shows a list of the used pre-trained models.

For each set of generated features, we run multiple combinations of clustering algorithms and configurations to get a diverse series of clustering results. The list is described in Table 5. The implementations are with scikit-learn, and we throw away too small sub-datasets generated (less than 800 images) and too large subsets (less than 800 images are not

[1] https://github.com/onnx/models

Figure 1. Preview of 12 datasets. We show 3 images per dataset.

#	Algorithm	Configuration
1	K-Means	MaxIter=2000, k-means++ init
2	K-Means	MaxIter=2000, random init
3	PCA + K-Means	#components=20
4	PCA + t-SNE + K-Means	#components=2
5	DBSCAN	eps=10, metric=l2
6	DBSCAN	eps=10, metric=cosine
7	Agglomerative	
8	Agglomerative	affinity=mahattan
9	Birch [37]	

Table 5. List of clustering algorithms and configurations to group the images based on features. The "number of clusters" is iterated in 2, 3, 5 and 7 for each configuration. The interpretation of configurations corresponds to parameters in scikit-learn.

Hidden units	64
Learning rate (η_z and η_θ)	0.0001
Optimizer	Adam
Batch size	4
N_{iters} (batches per task)	50
N_{trans} (transf. iterations)	20
λ_{sim} (triplet loss weight)	0.03
α (triplet loss margin)	0.5
λ_{reg} (SI weight)	10000
$\lvert S_{\text{offline}}\rvert : \lvert S_{\text{past}}\rvert : \lvert S\rvert$	5 : 1 : 1
Exploration-exploitation ratio	0.5

Table 6. Hyper-parameters to train the performance ranker. See Alg. 1 and §3 for explanation of notations.

selected). We respect the original train-validation-test split, and also remove subsets where the split becomes too deviated from balance. Finally, we have got 1418 sub-datasets generated from the original WIDER-Face dataset.

B.5. Performance ranker

Important hyper-parameters for performance ranker is shown in Table 6. Notably, we make the samples from S_{offline} to be more likely to be sampled, because there are significantly more offline samples than online samples.

For configuration encoder used in NAS, we use GIN [34] with 2 layers, dropout rate 0.2 and a learn-able epsilon. We add two virtual nodes to aggregate information from all nodes in each layer, similar to [31].

To accelerate the evaluation and save the huge computational cost to train and evaluate configurations repeatedly, we build a benchmark, *i.e.*, a performance lookup table, which consists of the validation and test AP of a specific configuration trained on a specific dataset. For each dataset, we randomly sample 200 distinct configurations from HPO and NAS space respectively. Afterwards, the ranker is inferenced on the 200 and the best is selected from them. This practice follows many recent NAS works [2, 19, 28, 33]. Notably, for larger space, the random sampling step can be easily replaced with an active learning approach (*e.g.*, bayesian optimization), but since the newly sampled configuration is likely to be a unseen one, we would not be able to use a benchmark to accelerate this process.

C. More experiment results

The performance of HyperFD on each dataset is shown in Table 7. Since the datasets can appear in random order during our evaluation, this table shows an average case of how much the dataset can benefit from others. HyperFD outcompetes baselines on most of the datasets. We also note that different datasets have different difficulties, causing performance gains to diverse. For example, we can easily get 1.1% AP improvement on WIDER-360, but for WIKI, the baseline is very close to perfect and the room for improvement is very narrow. The standard deviations of multiple runs with different random seeds are also shown in those tables.

D. Analysis of performance benchmark

As we have collected a large number of triplets (configurations, datasets and performances), we share some of the observations on this performance benchmark. We hope

Method	AFLW	ANIME	FaceMask	FDDB	FDDB-360	MAFA	Pascal VOC	UFDD	UMDAA-02	WIDER-360	WIKI	Average
Random search	99.15±0.12	97.58±0.18	94.32±0.43	97.42±0.22	97.00±0.18	92.98±0.48	97.36±0.41	78.50±0.59	99.63±0.04	67.23±0.97	99.67±0.08	92.80±9.90
Best on WIDER	99.00±0.00	97.53±0.00	94.02±0.00	97.11±0.00	**97.22±0.00**	93.13±0.00	97.12±0.00	78.27±0.00	99.60±0.00	**69.53±0.00**	99.63±0.00	92.92±9.36
Tr-AutoML	99.12±0.12	97.58±0.16	**94.51±0.35**	97.34±0.29	96.91±0.23	**93.28±0.35**	97.20±0.60	78.59±0.46	99.60±0.06	66.84±1.36	99.67±0.06	92.79±9.96
HyperSTAR	99.09±0.19	97.60±0.16	94.32±0.46	97.40±0.25	96.97±0.25	92.86±0.40	97.21±0.32	78.76±0.42	99.63±0.04	66.86±1.23	99.66±0.06	92.76±9.94
SCoT	99.17±0.11	97.62±0.20	94.25±0.55	97.38±0.16	96.91±0.23	92.80±0.46	97.39±0.36	78.49±0.57	99.62±0.04	67.40±0.72	99.69±0.04	92.79±9.86
HyperFD (ResNet)	99.21±0.09	**97.69±0.14**	94.47±0.32	97.53±0.12	96.93±0.21	92.68±0.54	97.53±0.39	78.63±0.41	99.63±0.03	66.59±0.79	**99.71±0.04**	92.78±10.05
HyperFD (stats)	99.21±0.06	97.61±0.16	94.36±0.36	97.54±0.12	96.91±0.15	92.83±0.34	97.56±0.12	**78.77±0.48**	99.62±0.04	66.79±1.07	99.70±0.04	92.81±9.98
HyperFD (MSE)	99.17±0.08	97.43±0.20	94.43±0.48	97.42±0.24	96.96±0.14	93.08±0.46	97.34±0.26	78.47±0.58	99.62±0.05	67.16±0.80	99.69±0.02	92.80±9.91
HyperFD	**99.25±0.01**	97.57±0.11	94.39±0.28	**97.62±0.02**	96.93±0.22	92.82±0.34	**97.63±0.05**	78.62±0.32	**99.64±0.01**	68.31±0.49	99.71±0.01	**92.95±9.66**

(a) AP of searched configuration on HPO space. The higher the better.

Method	AFLW	ANIME	FaceMask	FDDB	FDDB-360	MAFA	Pascal VOC	UFDD	UMDAA-02	WIDER-360	WIKI	Average
Random search	99.07±0.07	97.27±0.19	93.95±0.27	96.85±0.26	96.70±0.18	93.91±0.29	95.58±0.50	76.89±0.60	99.72±0.03	66.17±0.43	99.51±0.06	92.33±10.24
Best on WIDER	99.14±0.00	97.37±0.17	**94.36±0.09**	96.87±0.08	**96.95±0.02**	93.65±0.09	95.87±0.00	77.39±0.02	99.70±0.01	**66.75±0.13**	**99.59±0.00**	92.51±10.07
Tr-AutoML	99.03±0.13	97.38±0.21	93.89±0.42	96.71±0.29	96.80±0.11	93.95±0.45	95.15±0.76	76.67±0.66	99.71±0.04	66.35±0.30	99.45±0.11	92.28±10.22
HyperSTAR	99.06±0.09	97.31±0.18	93.99±0.17	96.85±0.25	96.71±0.12	93.84±0.34	95.65±0.54	76.84±0.63	99.71±0.03	66.02±0.39	99.50±0.04	92.32±10.29
SCoT	99.07±0.06	97.26±0.16	93.93±0.26	96.80±0.27	96.71±0.22	93.83±0.33	95.53±0.52	76.79±0.62	99.73±0.02	66.18±0.35	99.50±0.06	92.30±10.25
HyperFD (ResNet)	99.13±0.04	97.35±0.13	94.07±0.28	96.90±0.14	96.78±0.12	93.91±0.33	95.79±0.38	77.39±0.24	99.73±0.04	66.54±0.21	99.57±0.03	92.47±10.11
HyperFD (stats)	99.14±0.05	97.33±0.18	93.94±0.15	96.93±0.20	96.75±0.15	94.03±0.30	95.73±0.47	77.41±0.35	99.73±0.03	66.63±0.19	99.57±0.03	92.47±10.09
HyperFD (MSE)	99.07±0.07	97.21±0.19	93.79±0.19	**97.10±0.30**	96.67±0.11	93.99±0.22	95.38±0.59	77.13±0.36	**99.73±0.03**	66.18±0.34	99.57±0.03	92.35±10.21
HyperFD	**99.16±0.05**	**97.40±0.07**	93.89±0.25	97.05±0.15	96.90±0.11	**94.13±0.05**	**95.92±0.12**	**77.61±0.07**	99.72±0.03	66.64±0.05	99.58±0.02	**92.55±10.08**

(b) AP of searched configuration on NAS space. The higher the better.

Method	AFLW	ANIME	FaceMask	FDDB	FDDB-360	MAFA	Pascal VOC	UFDD	UMDAA-02	WIDER-360	WIKI	Average
Random search	20.09±16.21	20.09±16.22	20.09±16.22	20.09±16.22	20.09±16.22	20.09±16.21	20.09±16.22	20.09±16.22	20.09±16.22	20.10±16.21	20.09±16.22	20.09±16.21
Best on WIDER	41.40±0.00	19.91±0.00	31.94±0.00	47.22±0.00	**2.78±0.00**	12.68±0.00	30.09±0.00	25.46±0.00	30.09±0.00	**0.48±0.00**	34.26±0.00	25.12±0.00
Tr-AutoML	23.90±17.57	19.87±17.47	**13.35±11.52**	24.32±19.19	27.39±20.26	**10.58±9.94**	26.03±22.64	17.81±12.23	31.48±22.02	29.49±24.19	19.76±15.84	22.18±17.53
HyperSTAR	26.49±22.28	17.64±14.24	20.14±16.01	21.34±18.66	21.60±21.23	23.26±14.44	27.18±14.03	**12.75±10.95**	20.76±15.37	27.76±21.04	23.10±15.07	22.00±16.67
SCoT	16.56±15.89	18.03±15.29	23.77±22.20	24.58±14.01	26.90±19.23	26.34±15.43	19.10±15.88	20.86±16.35	21.94±15.76	16.24±11.10	14.84±11.29	20.83±15.68
HyperFD (ResNet)	10.75±12.58	**10.09±9.67**	14.23±10.39	11.74±9.95	26.41±19.02	30.91±19.29	12.72±14.73	15.23±12.01	20.22±12.78	30.78±15.75	**8.41±10.46**	17.41±13.33
HyperFD (Statistics)	10.37±9.27	16.00±14.12	18.08±13.88	10.42±9.89	27.22±15.80	24.53±12.56	11.81±5.32	13.26±11.94	21.67±16.47	28.55±17.73	11.62±10.50	17.59±12.50
HyperFD (MSE)	16.60±11.28	35.40±23.94	17.56±15.89	20.19±18.65	22.65±14.64	17.15±14.33	21.37±11.39	18.48±17.16	25.04±18.04	20.18±12.14	15.56±7.39	20.93±14.99
HyperFD	**4.48±0.75**	18.20±7.70	16.19±8.62	**3.83±1.83**	25.75±19.55	25.36±12.19	**8.32±2.23**	15.35±6.09	**12.94±5.15**	4.66±4.09	9.63±4.37	**13.16±6.60**

(c) Rank (normalized) of searched configuration on HPO space. The lower the better.

Method	AFLW	ANIME	FaceMask	FDDB	FDDB-360	MAFA	Pascal VOC	UFDD	UMDAA-02	WIDER-360	WIKI	Average
Random search	20.10±16.21	20.10±16.21	20.10±16.21	20.10±16.21	20.10±16.21	20.10±16.21	20.10±16.21	20.10±16.21	20.10±16.21	20.10±16.20	20.10±16.20	20.10±16.21
Best on WIDER	6.00±0.00	11.04±7.09	**2.81±1.27**	15.14±3.53	**2.62±0.96**	34.91±6.15	7.54±0.00	4.51±0.08	30.50±4.69	**2.64±2.24**	**3.01±0.30**	10.97±2.39
Tr-AutoML	30.46±29.48	16.94±17.09	26.73±26.91	27.82±21.15	8.54±7.09	21.39±24.41	32.22±27.95	30.82±22.48	34.46±21.25	13.47±9.68	35.34±29.03	25.29±21.50
HyperSTAR	23.70±21.27	17.20±14.12	15.05±10.76	19.67±15.01	18.15±11.33	24.55±18.37	18.02±17.67	21.91±17.90	26.15±14.62	25.78±14.72	21.83±12.33	21.09±15.28
SCoT	20.08±13.80	20.12±12.34	20.52±16.06	23.53±16.45	21.12±18.48	25.43±17.77	21.21±18.01	22.69±18.24	15.60±8.85	19.02±13.15	25.03±19.44	21.30±15.69
HyperFD (ResNet)	9.03±6.82	11.44±6.42	14.34±10.72	13.97±6.63	11.75±8.00	21.61±16.19	13.10±10.08	5.94±5.18	17.31±15.18	7.26±6.16	5.90±6.08	11.97±8.86
HyperFD (Statistics)	8.03±6.10	13.45±15.37	17.42±9.92	14.20±10.38	14.82±11.45	15.10±14.21	15.85±12.39	6.96±4.78	**15.43±14.29**	4.82±4.82	6.52±6.74	12.06±10.04
HyperFD (MSE)	22.07±19.74	26.07±19.50	29.64±15.64	8.93±7.67	21.07±11.73	15.08±10.85	26.63±21.48	12.42±8.73	15.79±12.35	18.69±12.56	5.37±6.05	18.34±13.30
HyperFD	**5.04±3.32**	**6.73±1.73**	23.15±12.00	**7.51±6.50**	4.86±4.19	**5.42±0.69**	**6.83±1.74**	**3.03±0.54**	16.70±11.03	2.91±0.49	3.46±4.03	**7.78±4.21**

(d) Rank (normalized) of searched configuration on NAS space. The lower the better.

Table 7. Performance of HyperFD per dataset, along with standard deviation.

those insights will inspire future research work of transferable AutoML.

Performance distribution and sensitivity to search space. We check the detection performance (AP@50), as shown in Figure 2. The hardest dataset is "WIDER-360", on which the best AP is less than 70%. UMDAA-02 turns out to be the easiest dataset of all, on which the majority of configurations are above 99.2%. The overall performance of HPO space is generally higher than NAS space, but MAFA and UMDAA-02 are two exceptions, where NAS is more useful than HPO.

The datasets that are most sensitive to hyper-parameter tuning are WIDER-360 and UFDD, where the gap between the best hyper-parameter and the worst differ by as much as 10%. while performances on UMDAA-02, WIKI and AFLW are very close, with less than 2% min-max-difference. For neural architecture search, the best and worst are closer in general, indicating that the final performance is less sensitive to changes in architectures alone.

To combine the advantages from both HPO and NAS space, one approach is do a joint search of hyper-parameters and neural architectures. However, this poses new challenges, both to search space design and search algorithms. There have been a few recent works that are jointly optimizing hyper-parameters and architectures [6, 10], but this problem remains challenging and open, even outside the context of transferable AutoML.

Validation-test correlation. For each dataset, we show the correlation between rankings on validation set versus rankings based on test set, *i.e.*, whether the better configuration found with validation dataset still performs better on a unseen test dataset. The results are shown in Figure 3. If the correlation is low, it means that a model that performs better on validation set does not necessarily performs better on the test dataset. This could be caused by the gap between distributions of validation and test set, and indistinguishability between configurations. Datasets suffering from such problem is ANIME and FaceMask, and NAS space is generally worse than HPO space. However, most of the numbers (especially on HPO space) are still higher than 0.8, indicat-

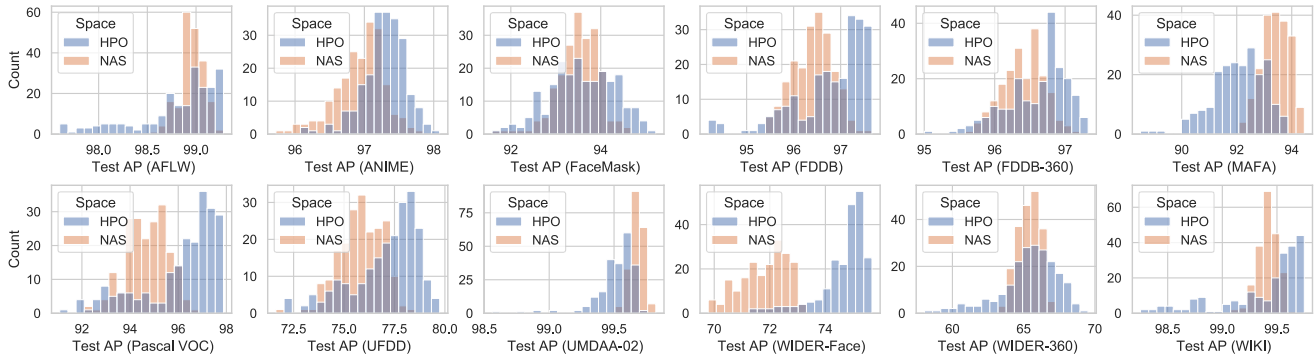


Figure 2. AP distribution on 12 datasets. The bars count for how many configurations we have sampled lie in a specific AP range.

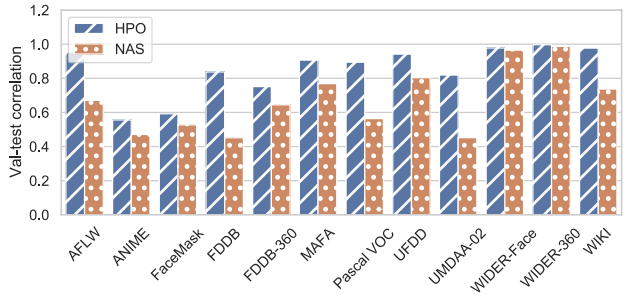


Figure 3. Correlations of configuration rankings on validation dataset and test dataset.

ing that the trained model is still likely to perform well on a real-world unseen dataset.

Configuration ranking correlation. We examine the correlation between the ranking of configurations on different datasets, and check whether different datasets have similar favors to some configurations. The results are shown in Figure 4. There are two interesting findings. (i) The heatmap on HPO and NAS space have a very different outlook, which means there is no transferability without a well-defined search space. For example, AFLW and WIKI have a high correlation on HPO space, but low on NAS. We hypothesis that the common characteristics that enables the transfer on HPO space does not apply well to architecture search. (ii) The overall correlations are higher for NAS space, indicating that datasets have more similar preferences for architectures. The correlations between WIDER-Face and WIDER-360 are especially high, which explains why "Best on WIDER" outperforms all other methods in Table 7.

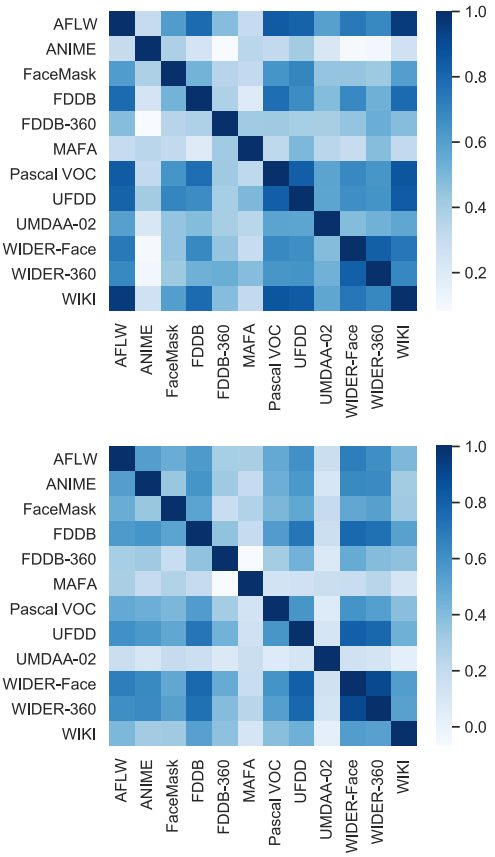
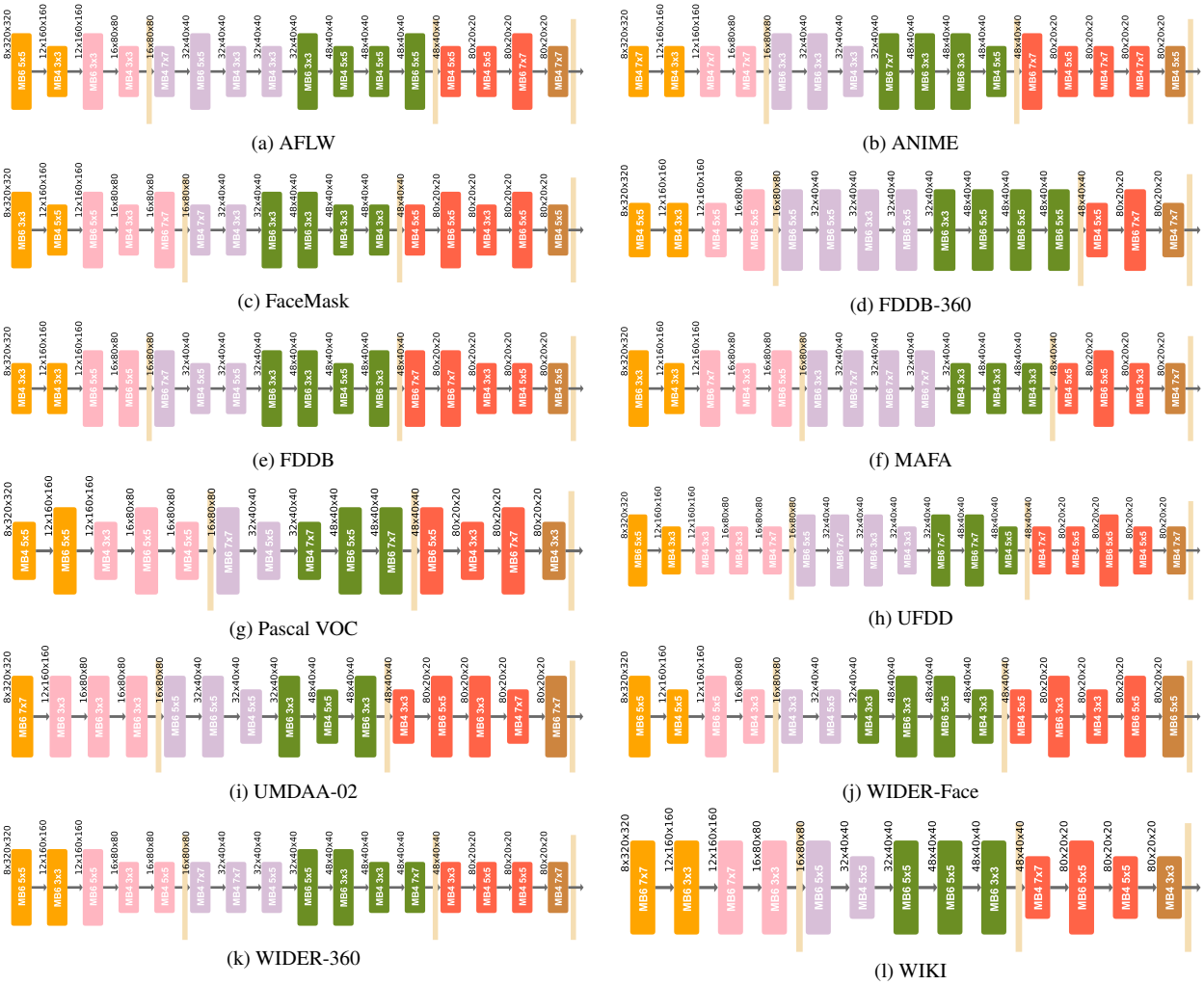


Figure 4. Mutual pearson correlation between hyperparameters on different datasets. The darker color indicates two datasets share more similar preferences for hyper-parameters / architectures. **(top)**: HPO space. **(bottom)**: NAS space.

Best hyper-parameter/architecture visualization. We show the best hyper-parameter found on our search space in Table 8, and the best architecture found in Figure 5. As illustrated, we observe no clear similarities among the best hyper-parameters or architectures, suggesting the need of tailored hparam/arch for each dataset.



(a) AFLW (b) ANIME (c) FaceMask (d) FDDB-360 (e) FDDB (f) MAFA (g) Pascal VOC (h) UFDD (i) UMDAA-02 (j) WIDER-Face (k) WIDER-360 (l) WIKI

Figure 5. Best neural architecture (backbone) on each dataset.

Dataset	Best hyper-parameter
AFLW	crop:0.55_iou:0.5_locw:8.0_negp:2.0_lr:3e-04_sgd
ANIME	crop:0.55_iou:0.4_locw:2.0_negp:2.0_lr:1e-03_adam
FaceMask	crop:0.3_iou:0.4_locw:2.0_negp:7.0_lr:3e-04_adam
FDDB	crop:0.55_iou:0.5_locw:8.0_negp:2.0_lr:3e-04_sgd
FDDB-360	crop:0.55_iou:0.5_locw:8.0_negp:7.0_lr:1e-03_adam
MAFA	crop:0.55_iou:0.5_locw:2.0_negp:2.0_lr:3e-04_adam
Pascal VOC	crop:0.3_iou:0.4_locw:8.0_negp:2.0_lr:3e-04_adam
UFDD	crop:0.3_iou:0.4_locw:2.0_negp:2.0_lr:3e-03_sgd
UMDAA-02	crop:0.55_iou:0.5_locw:4.0_negp:7.0_lr:1e-03_sgd
WIDER-Face	crop:0.3_iou:0.6_locw:2.0_negp:7.0_lr:3e-03_sgd
WIDER-360	crop:0.55_iou:0.6_locw:2.0_negp:7.0_lr:3e-03_sgd‘
WIKI	crop:0.55_iou:0.4_locw:4.0_negp:2.0_lr:1e-03_sgd

Table 8. Best hyper-parameter searched on each dataset.

References

[1] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 279–283, 2016. 2

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559. PMLR, 2018. 3

[3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2018. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 2

[5] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu,

Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark, 2019. 1

[6] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using neural acquisition function. *arXiv e-prints*, pages arXiv–2006, 2020. 4

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[8] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 2

[9] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild, 2019. 1, 2

[10] Xuanyi Dong, Mingxing Tan, Adams Wei Yu, Daiyi Peng, Bogdan Gabrys, and Quoc V Le. Autohas: Differentiable hyper-parameter and architecture search. *arXiv e-prints*, pages arXiv–2006, 2020. 4

[11] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015. 1, 2

[12] Jiemin Fang, Yuzhu Sun, Kangjian Peng, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Fast neural network adaptation via parameter remapping and architecture search. In *International Conference on Learning Representations*, 2019. 2

[13] Jianglin Fu, Saeed Ranjbar Alvar, Ivan V. Bajic, and Rodney G. Vaughan. Fddb-360: Face detection in 360-degree fisheye images, 2019. 1, 2

[14] Jianglin Fu, Ivan V Bajić, and Rodney G Vaughan. Datasets for face and object detection in fisheye images. *Data in brief*, 27:104752, 2019. 1, 2

[15] Shiming Ge, Jia Li, Qiting Ye, and Zhao Luo. Detecting masked faces in the wild with lle-cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2

[16] Wobot Intelligence. Face mask detection dataset. https://www.kaggle.com/wobotintelligence/face-mask-detection-dataset, 2020. 1, 2

[17] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010. 1, 2

[18] Grigorios Kalliatakis. Keras-vgg16-places365. https://github.com/GKalliatakis/Keras-VGG16-places365, 2017. 2

[19] Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. Nas-bench-nlp: neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*, 2020. 3

[20] Sebastian Lapuschkin, Alexander Binder, Klaus-Robert Muller, and Wojciech Samek. Understanding and comparing deep neural networks for age and gender classification. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1629–1638, 2017. 2

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2

[22] Linzaer. Ultra-light-fast-generic-face-detector-1mb. https://github.com/Linzaer/Ultra-Light-Fast-Generic-Face-Detector-1MB, 2020. 1

[23] Upal Mahbub, Sayantan Sarkar, and Rama Chellappa. Partial face detection in the mobile domain, 2017. 1, 2

[24] Peter M. Roth Martin Koestinger, Paul Wohlhart and Horst Bischof. Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization. In *Proc. First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011. 1, 2

[25] Hajime Nada, Vishwanath A Sindagi, He Zhang, and Vishal M Patel. Pushing the limits of unconstrained face detection: a challenge dataset and baseline results. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–10. IEEE, 2018. 1, 2

[26] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 464–479, 2018. 2

[27] qhgz2013. anime-face-detector. https://github.com/qhgz2013/anime-face-detector, 2020. 1, 2

[28] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1882–1890, 2019. 3

[29] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, December 2015. 1, 2

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. 1

[31] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020. 3

[32] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 2

[33] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European Conference on Computer Vision*, pages 660–676. Springer, 2020. 3

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 3

[35] Yuanyuan Xu, Wan Yan, Haixin Sun, Genke Yang, and Jiliang Luo. Centerface: Joint face detection and alignment using face as point. In *arXiv:1911.03599*, 2019. 1, 2

[36] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[37] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996. 3