AutoLoss-Zero: Searching Loss Functions from Scratch for Generic Tasks

Hao Li[1*†], Tianwen Fu[2*†], Jifeng Dai[4,5], Hongsheng Li[1], Gao Huang[4], Xizhou Zhu[3‡]
[1]CUHK-SenseTime Joint Laboratory, The Chinese University of Hong Kong
[2]Department of Information Engineering, The Chinese University of Hong Kong
[3]SenseTime Research [4]Tsinghua University
[5]Qing Yuan Research Institute, Shanghai Jiao Tong University
haoli@link.cuhk.edu.hk, futianwen@ie.cuhk.edu.hk, daijifeng001@gmail.com
hsli@ee.cuhk.edu.hk, gaohuang@tsinghua.edu.cn, zhuwalter@sensetime.com

Abstract

Significant progress has been achieved in automating the design of various components in deep networks. However, the automatic design of loss functions for generic tasks with various evaluation metrics remains under-investigated. Previous works on handcrafting loss functions heavily rely on human expertise, which limits their extensibility. Meanwhile, searching for loss functions is nontrivial due to the vast search space. Existing efforts mainly tackle the issue by employing task-specific heuristics on specific tasks and particular metrics. Such work cannot be extended to other tasks without arduous human effort. In this paper, we propose AutoLoss-Zero, which is a general framework for searching loss functions from scratch for generic tasks. Specifically, we design an elementary search space composed only of primitive mathematical operators to accommodate the heterogeneous tasks and evaluation metrics. A variant of the evolutionary algorithm is employed to discover loss functions in the elementary search space. A loss-rejection protocol and a gradient-equivalence-check strategy are developed so as to improve the search efficiency, which are applicable to generic tasks. Extensive experiments on various computer vision tasks demonstrate that our searched loss functions are on par with or superior to existing loss functions, which generalize well to different datasets and networks. Code shall be released.

1. Introduction

Recent years have witnessed exciting progress in AutoML for deep learning [15, 35, 36, 41, 42, 68]. The automatic design of many components has been explored, ranging from architectures (*e.g.*, neural architectures [50] and normalization-activation operations [35]) to learning strategies (*e.g.*, data augmentation strategies [15], dropout patterns [42], and training hyper-parameters [16]). However, to automate the entire deep learning process, an essential component is under-investigated, namely, the automatic design of loss functions for generic tasks.

Loss functions are indispensable parts in deep network training. In various tasks, including semantic segmentation [7, 65], object detection [19, 51], instance segmentation [3, 23] and pose estimation [56], cross-entropy (CE) and L1/L2 losses are the default choices for categorization and regression, respectively. As the default loss functions are usually approximations for specific evaluation metrics, there usually exists a misalignment between the surrogate loss and the final evaluation metric. For example, for bounding box localization in object detection, L1 loss is widely used, while the IoU metric is the standard evaluation metric [63]. Similar discrepancy has also been observed in semantic segmentation [31], where some metrics measure the accuracy of the whole image, while others focus more on the segmentation boundaries. The misalignment between network training and evaluation results in sub-optimal solutions with degraded performance.

A multitude of handcrafted loss functions have been proposed for different evaluation metrics. Since most desired metrics are non-differentiable and cannot be directly used as training objectives, many existing works [4, 19, 29, 33, 44, 53, 61] design differentiable variants of the CE and L1/L2 losses by carefully analyzing specific evaluation metrics. Another series of works [2, 38, 43, 45, 52, 63, 66] handcraft clever surrogate losses based on the mathematical expressions of specific evaluation metrics. Although these handcrafted loss functions show improvement on their target metrics, they heavily rely on expertise and careful analysis for specific scenarios, which limits their extendibility.

In this paper, we aim to automate the design of loss functions for generic tasks. Although there are several pioneer works [30, 31, 37, 58] on loss function search, they are

*Equal contribution. †This work is done when Hao Li and Tianwen Fu are interns at SenseTime Research. ‡Corresponding author.

all limited to specific tasks and particular evaluation metrics, with task-specific heuristics, which cannot be applied to generic tasks. For example, [31] constructs the search space by parametrizing the evaluation metrics of semantic segmentation, which can hardly be applied to mAP metric in object detection; [37] proposes a rejection protocol for object detection, which is designed based on specific analysis of mAP metric properties by human expertise. Searching loss functions for generic tasks is much more challenging, because of the heterogeneity of various tasks and evaluation metrics. The search space should be composed of basic primitive operators so as to accommodate such heterogeneity, and the search algorithm should be efficient enough so as to find the best combination of basic primitives for the given task and evaluation metric. Meanwhile, no task-specific heuristics should be involved in the search.

This paper presents a general loss function search framework applicable to various evaluation metrics across different tasks, named AutoLoss-Zero. We build our search space only with primitive mathematical operators to enjoy the high diversity and expressiveness. A variant of the evolutionary algorithm is employed to discover the high-quality loss functions from scratch with minimal human expertise. Specifically, AutoLoss-Zero formulates loss functions as computational graphs composed only of primitive mathematical operators (see Table 1). The computation graphs are randomly built from scratch, and are evolved according to their performance on the target evaluation metrics. In the search algorithm, to improve the search efficiency, we propose a loss-rejection protocol that efficiently filters out the unpromising loss function candidates, which brings great speed-up to the search procedure. A gradient-equivalence-check strategy is developed to avoid duplicate evaluations of equivalent loss functions. The loss-rejection protocol and the gradient-equivalence-check strategy, with no task-specific or metric-specific design, are generally applicable to various tasks and metrics.

We validate our framework on various computer vision tasks, including semantic segmentation, object detection, instance segmentation, and pose estimation. Extensive experiments on large-scale datasets such as COCO [34], Pascal VOC [17] and Cityscapes [13] show that the searched losses are on par with or superior to existing handcrafted and specifically searched loss functions. Ablation studies show that our searched loss functions can effectively generalize to different networks and datasets. Our main contributions can be summarized as follows:

- AutoLoss-Zero is a general AutoML framework to search loss functions from scratch for generic tasks with minimal human expertise. The effectiveness is demonstrated on a variety of computer vision tasks.
- A novel loss-rejection protocol is developed to filter out the unpromising loss functions efficiently. A gradient-equivalence-check strategy is also developed to avoid duplicate evaluations. These techniques bring great improvement to the search efficiency, and are designed with special focus to enable generalization to all tasks and metrics without extra effort.
- The searched loss functions by themselves are contributions, because they are transferable across different models and datasets with competitive performance.

2. Related Work

Hand-crafted loss functions for prevalent evaluation metrics have been studied by numerous works. A large fraction of previous works develop loss function variants based on the standard cross-entropy loss and L1/L2 loss. For categorization, [29, 33, 53, 61] mitigate the imbalance of samples by incorporating different sample weights. [4, 44] propose to up-weight the losses at boundary pixels to deliver more accurate boundaries. For regression, Smooth-L1 loss [19] is proposed for improved stability and convergence. Another line of research [2, 38, 43, 45, 52, 63, 66, 67] deals with the misalignment between loss functions and various evaluation metrics by handcrafting differentiable extensions or surrogates of metrics as loss functions, including segmentation IoU [2, 45], F1 score [38], bounding box IoU [52, 63, 66, 67], and Average Precision [43].

Although these handcrafted losses are successful under different scenarios, they heavily rely on careful design and expertise for analyzing the property of specific metrics. In contrast, we propose an automated loss design framework that is generally suitable for different tasks and metrics.

Direct optimization for non-differentiable evaluation metrics has also been studied. For structural SVMs [57], [26, 46, 64] propose non-gradient methods to directly optimize ideal metrics. [22, 39, 55] apply loss-augmented inference to derive the gradients from the expectation of metrics. However, the computational complexity is high, which requires specifically designed efficient algorithms for different metrics. Policy gradients [1, 47, 48, 54, 60] are also adopted to directly optimize non-differentiable metrics. However, these methods suffer from: 1) complicated action space, which requires task specific approximations [48]; 2) high variance of gradient estimation and objective instability [59]. Recently, [5, 40] adopt error-driven learning for object detection, which is limited to specific scenarios.

Although these methods mitigate the mis-alignment issue between training objectives and evaluation metrics, they require specific analysis and designs for the target metrics.

AutoML for generic tasks has long been pursued in machine learning research [25]. Recent works include automated search for neural architecture (NAS) [36, 41, 68], normalization-activation operations [35], dropout patterns [42], data augmentation [15], and training hyper-parameters [16]. Most of the existing works aim to specialize an architecture built upon expert-designed operators [36, 41, 68], or search for specific hyper-parameters in a fixed formula [15, 16, 42].

Our work shares a similar philosophy to AutoML-Zero [49] and EvoNorm [35], which employ evolutionary algorithms to search for ML algorithms or normalization-activation operations from only primitive mathematical operations. However, for loss functions, the search space design is quite different and there are unique properties that can be leveraged for efficient search. We introduce 1) an effective search space for loss functions with specific initialization and mutation operations; and 2) a loss-rejection protocol and a gradient-equivalence-check strategy to improve the search algorithm efficiency.

Loss function search has raised the interest of researchers in recent years. All the pioneer works [30, 31, 37, 58] are limited to specific tasks and metrics, with task-specific heuristics. Specifically, [30, 58] search for optimal losses for face recognition. The searched loss functions are optimal combinations of existing handcrafted variants of the cross-entropy loss. As the resulting objective is essentially an integration of existing loss functions, it cannot solve the mis-alignment between cross-entropy losses and many target metrics well. Recently, [31] proposes to search loss functions for semantic segmentation by substituting the logical operations in metrics with parameterized functions. However, such parameterization cannot be easily extended for generic metrics, such as mAP in object detection, where the matching and ranking are difficult to be parameterized.

A closely related work is [37], which searches loss functions for object detection. Similar to our method, [37] also formulates loss functions as the combination of primitive operators. However, [37] initializes the search from well-performed handcrafted loss functions specific for object detection, and separately searches for one loss branch with the other loss branch fixed as initialization. Moreover, [37] designs their loss-rejection protocol specifically for object detection, and cannot be applied to other tasks. In contrast, our method can simultaneously search for multiple loss branches from random initialization without starting from any human-designed loss functions. Our method has no specialized design for specific tasks or metrics, and consequently is applicable to generic tasks.

3. Method

Given a task (*e.g.*, semantic segmentation and object detection) and a corresponding evaluation metric (*e.g.*, mIoU and mAP), AutoLoss-Zero aims to automatically search a proper loss function from scratch for training a neural network. A general search space is proposed, in which each loss function is represented as a computational graph. The graph takes the network predictions and ground truths as inputs, and transforms them into a final loss value. With minimal human expertise, only primitive mathematical operations (see Table 1) are used as the intermediate computational nodes to accommodate the high diversity among different tasks and metrics. An efficient evolutionary algorithm is employed to search the loss function for the given task and metric. To enable the evolution, effective random initialization and mutation operations are defined. A novel loss-rejection protocol and gradient-equivalence-check strategy are also proposed to improve the search efficiency, which are applicable to generic tasks. Different form [31, 37], our method is designed with special focus on generality, so that no task-specific heuristics are employed.

Element-wise Operator	**Expression**	**Arity**
Add	$x + y$	2
Mul	$x \times y$	2
Neg	$-x$	1
Abs	$\|x\|$	1
Inv	$1/(x+\epsilon)$	1
Log	$\text{sign}(x) \cdot \log(\|x\| + \epsilon)$	1
Exp	e^x	1
Tanh	$\tanh(x)$	1
Square	x^2	1
Sqrt	$\text{sign}(x) \cdot \sqrt{\|x\| + \epsilon}$	1
†Aggregation Operator	**Expression**	**Arity**
Mean_{nhw}	$\frac{1}{NHW} \sum_{nhw} x_{nchw}$	1
Mean_{c}	$\frac{1}{C} \sum_{c} x_{nchw}$	1
$\text{Max-Pooling}_{3\times3}$	$\text{Max-Pooling}_{3\times3}(x)$	1
$\text{Min-Pooling}_{3\times3}$	$\text{Min-Pooling}_{3\times3}(x)$	1

Table 1. Primitive operator set $\mathcal{H}$. x and y are of the same shape of (N, C, H, W), which are the input tensors of the operators. $\epsilon = 10^{-12}$ is a small positive number for avoiding infinite values or gradients. This primitive operator set is shared in all of our experiments. † Each aggregation operator is a mapping that replaces the elements of the input tensor with the aggregated values. Both the stride and padding of Max/Min-Pooling are set as 1. Thus, all of the operators preserve the shape of the input tensor(s).

3.1. Search Space

The search spaces of most AutoML approaches [15, 35, 36, 41, 42, 68] are specially designed for particular purposes and not suitable for loss functions. In loss function search, [31] proposes a loss function search space specifically for semantic segmentation, which cannot be extended to generic tasks. In [30, 58], the search space is simply the combination of existing loss functions, which cannot form new loss functions. The search space of [37] is also of primitives, which is most similar to ours. However, the primitives of [37] is a constrained set for the specific task of object detection. In this subsection, we design a general search space for loss functions applicable for generic tasks and evaluation metrics.

AutoLoss-Zero seeks to search the proper loss function for training the networks that maximizes the given evaluation metric ξ. The loss function $L(\hat{y}, y; \mathcal{N}_\omega)$ is defined on the network prediction $\hat{y}$ and its ground-truth training target

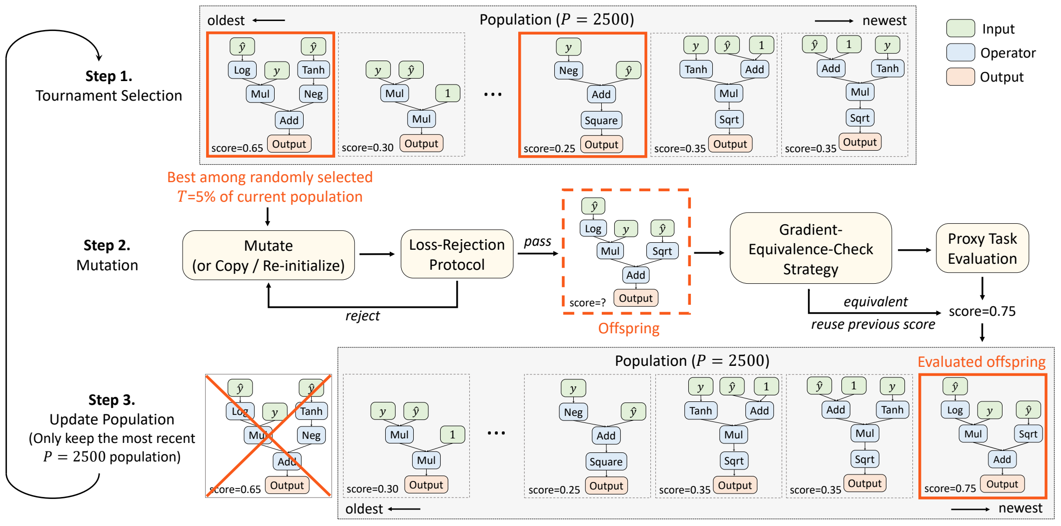


Figure 1. Overview of the search pipeline.

y, where $\mathcal{N}_\omega$ is a network parameterized with ω. The search target can be formulated as a nested optimization,

$$\begin{aligned} \max_L \quad & f(L;\xi) = \xi\left(\mathcal{N}_{\omega^*(L)};\mathcal{S}_{\text{eval}}\right), \\ \text{s.t.} \quad & \omega^*(L) = \arg\min_\omega \; \mathbb{E}_{(\hat{y},y)\in\mathcal{S}_{\text{train}}}\left[L(\hat{y},y;\mathcal{N}_\omega)\right], \end{aligned} \tag{1}$$

where $f(L;\xi)$ is the evaluation score of the loss function L under the given metric ξ, and $\omega^*(L)$ is the network parameters trained with L. $\mathbb{E}[\cdot]$ is the mathematical expectation. $\mathcal{S}_{\text{train}}$ and $\mathcal{S}_{\text{eval}}$ are the training and evaluation sets used in the search process, respectively. The network prediction $\hat{y}$ and its training target y share the same shape of (N,C,H,W). For each tensor, we use N,C,H,W to refer to the size of its batch, channel, width and height, respectively[1].

Eq. (1) provides a general optimization formula, but it cannot be trivially optimized by naïve search methods in an affordable time. We design the search space and algorithm to efficiently optimize Eq. (1) without loss of generality.

Loss Function Representation. The loss function L is represented as a computational graph $\mathcal{G}$. The computational graph is a rooted tree, where the leaf nodes are inputs (*i.e.*, network predictions and training targets), and the root is the output. The intermediate computational nodes are selected from a set of primitive mathematical operations (see Table 1), which transform the inputs into the final loss value.

The input tensors of the computational graph are sampled with replacement from $\{y,\hat{y},1\}$, where the additional constant 1 is included to improve the flexibility of the search space. The output tensor o has the same shape of (N,C,H,W) as the inputs, which is further aggregated to form the final loss value as

$$L(\hat{y},y) = \frac{1}{NHW}\sum\nolimits_{nchw} o_{nchw}. \tag{2}$$

Here, we do not normalize among the channel dimension, following the common practice of the cross-entropy loss.

[1]For the predictions and training targets without spatial dimensions, we set $H=1$ and $W=1$ without loss of generality.

As some tasks may have multiple loss branches (*e.g.*, the classification and regression branches in object detection), we represent the loss of each branch as an individual computational graph, and sum their loss values together as the final loss. For a loss with M branches, given the predictions $\{\hat{y}^1,\hat{y}^2,\ldots,\hat{y}^M\}$ and their ground-truth training targets $\{y^1,y^2,\ldots,y^M\}$ of each loss branch, the final loss function is represented as $L(\hat{y},y)=\sum_{i=1}^M L_i(\hat{y}^i,y^i)$.

Primitive Operators. Table 1 summarizes the primitive operator set $\mathcal{H}$ used in our search space, including element-wise operators and aggregation operators that enable information exchange across spatial and channel dimensions. Each aggregation operator is a mapping that replaces the elements of the input tensor with the aggregated values. All the primitive operators preserve the shape of the input tensors in order to ensure the validity of computations.

3.2. Search Algorithm

Inspired by the recent applications of AutoML [35, 49], a variant of evolutionary algorithm is employed for searching loss functions. In existing works of loss function search [30, 31, 37, 58], variants of reinforcement learning or evolution algorithm are also adopted. However, the search methods in [30, 31, 37, 58] are designed for searching in specific tasks and particular metrics, with task-specific heuristics, which can hardly be applied to generic tasks. Here, AutoLoss-Zero searches loss functions for generic tasks from random initialization with minimal human expertise. The proposed method has no specialized design for specific tasks or metrics, which is widely applicable to generic tasks.

Figure 1 illustrates the search pipeline of AutoLoss-Zero. At initialization, K loss functions ($K=20$ by default) are randomly generated to form the initial population. Each evolution picks T ratio of population ($T=5\%$ by default) at random, and selects the one with the highest evaluation score as the parent, *i.e.*, tournament selection [20]. The

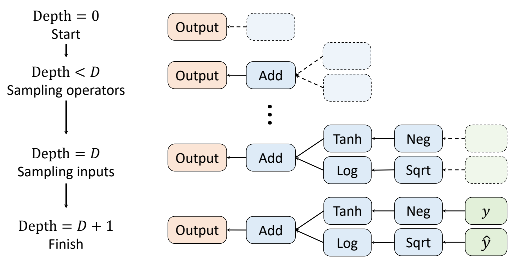


Figure 2. Random initialization of loss functions.

parent is used to produce offspring through well-designed mutation operations. Following [35, 49], only the most recent P loss functions ($P = 2500$ by default) are maintained.

As the search space is very sparse with a large number of unpromising loss functions, a novel loss-rejection protocol is developed to efficiently filter out loss functions that are not negatively correlated with the given evaluation metric. During the search, the initialization / mutation process of an individual loss function would be repeated until the resulting loss function can pass the loss-rejection protocol.

In order to further improve the search efficiency, a gradient-equivalence-check strategy is developed to avoid re-evaluating mathematically equivalent loss functions. Similar to [35, 49], lightweight proxy tasks are employed to reduce the computational cost of evaluating loss functions, which will be discussed at the end of this subsection.

Random Initialization of Loss Functions. To ensure generality, heuristic initialization as in [37] shall not be employed. Instead, our computational graph of each initial loss function is randomly generated. Figure 2 illustrates the process of loss function generation. Starting from a graph with root (*i.e.*, the output node) only, each node would randomly sample one or two operators from the primitive operator set $\mathcal{H}$ (see Table 1), and append to the graph as its child node(s). The root has one child. For each computational node, the number of child nodes is decided by its operator arity.

When a computational node reaches the target depth D ($D = 3$ by default), it randomly selects input tensor(s) as its child node(s). The input tensors would be the leaf nodes of the computational graph. Each randomly generated computational graph has a depth of $D + 1$, with D computational nodes on each path from the root to a leaf node.

Mutation. The mutation process is inspired by [49], but the candidate mutation operations are specially designed for our search space. Figure 3 illustrates the candidate mutation operations, which are defined as:

- *Insertion.* An operator randomly sampled from $\mathcal{H}$ is inserted between a randomly selected non-root node and its parent. If the operator has an arity of 2, it would randomly select an input as the additional child.
- *Deletion.* An intermediate computational node is randomly selected and removed. For the removed node, one of its child nodes is randomly picked to become the new child of its parent.

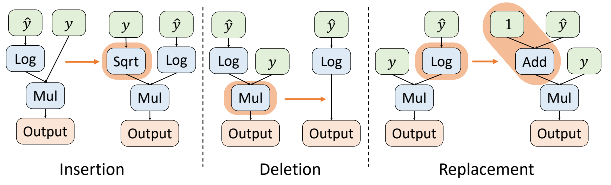


Figure 3. Candidate mutation operations.

- *Replacement.* An operator is randomly sampled from $\mathcal{H}$ to replace a randomly selected non-root node. If the non-root node has children more than the operator arity, a random subset of the child nodes with the same number as the arity are kept as the children. Otherwise, it would randomly select inputs as the additional children.

To produce the offspring, the given computational graph is processed by the three sequential steps:

1. Directly copy the graph with a probability of 10%.
2. If copying is not performed, randomly re-initialize the full computational graph with a probability of 50%.
3. If re-initialization is not performed, sequentially perform two mutation operations uniformly sampled from $\{Insertion, Deletion, Replacement\}$.

Loss-Rejection Protocol. Our search space is highly flexible, in which only primitive operations are used to construct the loss functions. Similar to [35, 37, 49], such flexibility leads to a large and sparse search space. Most loss function candidates result in network performance that is not better than random guessing. In loss function search, to improve the search efficiency, [37] designs a loss-rejection protocol to filter out unpromising loss functions before training the networks. However, it is specifically designed for object detection, which cannot be directly applied to generic tasks. Here, we propose a novel loss-rejection protocol that is generally applicable to various tasks and metrics.

Inspired by the fact that minimizing the proper loss functions should correspond to maximizing the given evaluation metric, we develop an efficient loss-rejection protocol for generic tasks. Given B random samples ($B = 5$ by default) from the training set $\mathcal{S}_{\text{train}}$ and a randomly initialized network $\mathcal{N}_{\omega_0}$, we record the network predictions and the corresponding training targets as $\{(\hat{y}_b, y_b)\}_{b=1}^{B}$. To efficiently estimate the correlation between the given evaluation metric ξ and a candidate loss function L, a correlation score $g(L;\xi)$ is calculated as

$$g(L;\xi) = \frac{1}{B}\sum_{b=1}^{B} \xi\left(\hat{y}_b^*(L), y_b\right) - \xi\left(\hat{y}_b, y_b\right), \quad \text{s.t.} \quad \hat{y}_b^*(L) = \arg\min_{\hat{y}_b} L(\hat{y}_b, y_b), \tag{3}$$

where $\hat{y}_b^*(L)$ is the predictions optimized with loss L. A large $g(L;\xi)$ indicates that minimizing the loss L corresponds to maximizing the evaluation metric ξ. Otherwise, if $g(L;\xi)$ is less than a threshold η, the loss function L is regarded as unpromising, which should be rejected.

Here, to speed up the rejection process, the loss function optimization is directly applied to the network prediction

$\hat{y}_b$, instead of the network parameters ω. Since the network computation is omitted, the rejection process is very efficient. With a single GPU, the proposed loss-rejection protocol can reach a throughput of 500$\sim$1000 loss functions per minute. In search, the initialization / mutation process of an individual loss function would be repeated until the resulting loss function can pass the loss-rejection protocol.

Gradient-Equivalence-Check Strategy. To avoid re-evaluating mathematically equivalent loss functions, a gradient-equivalence-check strategy is developed. For each loss function L, we compute its gradient norms w.r.t. the network predictions used in the loss-rejection protocol as $\{\|\partial L/\partial \hat{y}_b\|_2\}_{b=1}^{B}$. If for all of the B samples, two loss functions have the same gradient norms within two significant digits, they are considered equivalent, and the previous evaluation metric score would be reused.

Proxy Task. The evaluation of loss functions requires network training, which costs the most time in the search. Similar to AutoML works [35, 49], to accelerate the search, lightweight proxy tasks for network training are employed in the loss function evaluation. Specifically, fewer training iterations, smaller models and down-sampled images are adopted (see Section 4 and Appendix A). We further improve the efficiency by stopping the network training with invalid loss values (*i.e.*, NaN and Inf values).

4. Experiments

Implementation Details. For the evolutionary algorithm, the population is initialized with $K = 20$ randomly generated loss functions, and is restricted to most recent $P = 2500$ losses. The ratio of tournament selection [20] is set as $T = 5\%$ of current population. During random initialization and mutations, the sampling probabilities for all the operators in Table 1 are the same. The initial depth of computational graphs is $D = 3$. For the loss-rejection protocol and the gradient-equivalence-check strategy, $B = 5$ samples are randomly selected from $\mathcal{S}_{train}$. The search and re-training experiments are conducted on 4 NVIDIA V100 GPUs. The proxy tasks are designed such that 300 evaluations can be conducted in 48 hours. More details are in Appendix A.

4.1. Semantic Segmentation

Settings. Semantic segmentation concerns categorizing each pixel in an image into a specific class. PASCAL VOC 2012 [17] with extra annotations [21] is utilized for our experiments. The target evaluation metrics include Mean IoU (mIoU), Frequency Weighted IoU (FWIoU), Global Accuracy (gAcc), Mean Accuracy (mAcc), Boundary IoU (BIoU) [28] and Boundary F1 Score (BF1) [14]. The first four metrics measure the overall segmentation accuracy, and the other two metrics evaluate the boundary accuracy.

During search, we use DeepLabv3+ [8] with ResNet-50 [24] as the network. Following [31], we simplify the proxy task by down-sampling the input images to the resolution of 128$\times$128, and reducing the training schedule to 3 epochs (1/10 of the normal training schedule). After the search procedure, we re-train the segmentation networks with ResNet-101 [24] as the backbone for 30 epochs. The input image resolution is 512$\times$512. The re-training setting is the same as [8], except that the searched loss function is utilized. More details are in Appendix A.1.

Results. Table 2 compares our searched losses with the widely used cross-entropy loss, other metric-specific handcrafted loss functions, and the surrogate losses searched by Auto Seg-loss (ASL) [31], CSE-Autoloss (CSE) [37] and AutoML-Zero (AML) [49]. Note that ASL is restricted to a specific designed search space for semantic segmentation, which cannot be simply extended to handle generic metrics; CSE designs initialization and rejection protocols specifically for object detection, and we make our best effort to implement it on semantic segmentation. The results show that our searched losses outperform the manually designed losses consistently, and on par with or better than the searched losses by ASL on all target metrics. CSE fails to find loss functions better than CE regardless of initialization. AML is designed for searching machine learning algorithms with over 10^{10} evaluations, and could not find any well-performing loss functions with the same time and computational resource as ours. Appendix B presents the formulas of the discovered loss functions, which indicate that the intermediate aggregations (*e.g.*, Max-Pooling$_{3\times3}$ and Mean$_{nhw}$) between non-linear operations may have potential benefits for metrics such as mAcc, BIoU, and BF1.

Generalization of the searched functions. To verify the generalization ability of the searched losses, we conduct re-training experiments on different datasets and networks using the CE loss and the losses originally searched for DeepLabv3+ [8] with ResNet50 [24] on PASCAL VOC [17]. Due to limited computational resource, we only compare on mIoU and BF1 metrics. Table 3 summarizes the results on PASCAL VOC and Cityscapes [13], using DeepLabv3+ / PSPNet [65] with ResNet-50 / ResNet-101 as the networks. The results show that the searched loss functions generalize well between different datasets, and can be applied to various semantic segmentation networks.

4.2. Object Detection

Settings. Object detection is the task of detecting the bounding boxes and categories of instances belonging to certain classes. To evaluate our algorithm, we conduct experiments on the widely used COCO dataset [34]. The target evaluation metric is Mean Average Precision (mAP).

We use Faster R-CNN [51] with ResNet-50 [24] and FPN [32] as the detection network. There are 4 loss branches, *i.e.*, the classification and regression branches for the RPN [51] sub-network and Fast R-CNN [19] sub-

Loss Function		FWIoU	gAcc	mAcc	BIoU	mIoU	BF1
Cross Entropy		91.3	**95.2**	87.3	70.6	78.7	65.3
WCE [53]		85.6	91.1	**92.6**	61.8	69.6	37.6
DiceLoss [38]		91.3	95.1	87.5	69.9	77.8	64.4
Lovàsz [2]		**91.8**	**95.4**	88.6	72.5	79.7	66.7
DPCE [4]		**91.8**	**95.5**	87.8	71.9	79.8	66.5
SSIM [44]		**91.7**	**95.4**	87.9	71.5	79.3	66.4
FWIoU	ASL [31]	**91.9**	**95.4**	89.2	75.1	80.0	65.7
	Ours	**91.7**	**95.2**	87.7	72.9	78.7	64.6
gAcc	ASL [31]	**91.8**	**95.5**	89.0	74.1	79.7	64.4
	Ours	**91.7**	**95.3**	88.7	73.6	79.4	64.8
mAcc	ASL [31]	85.9	91.3	**92.7**	72.9	69.8	35.6
	Ours	89.2	93.7	**92.6**	73.7	75.3	44.1
BIoU	ASL [31]	69.9	62.6	81.3	**79.2**	49.0	39.0
	Ours	69.5	80.5	67.1	**79.3**	50.0	34.4
mIoU	CSE [37]	91.4	95.2	87.0	72.6	78.1	64.1
	CSE-RandInit	89.6	93.9	83.1	64.6	71.9	56.5
	AML [49]	59.5	64.4	4.9	1.3	4.0	0.4
	ASL [31]	**92.1**	**95.7**	88.2	73.4	**81.0**	68.9
	Ours	**92.1**	**95.7**	89.1	74.1	**80.7**	66.0
BF1	CSE [37]	91.8	95.4	88.5	73.7	79.4	65.1
	CSE-RandInit	69.3	75.6	9.0	3.0	5.3	1.0
	AML [49]	0.5	2.6	4.7	1.7	0.8	1.1
	ASL [31]	1.0	2.7	6.5	7.4	1.9	74.8
	Ours	4.2	9.1	11.9	26.1	7.3	**76.7**

Table 2. Semantic segmentation results of DeepLabv3+ [8] with ResNet-101 [24] on PASCAL VOC [17]. Results of the target metric(s) for each loss function are underlined, and the highest results within a tolerance of 0.5 are in **bold**. AutoML-Zero (AML), CSE-Autoloss with CE initialization (CSE) and random initialization (CSE-RandInit) are re-implemented according to their papers.

Dataset		Cityscapes		VOC			
Network		R101-DLv3+		R50-DLv3+		R101-PSP	
Loss Function		mIoU	BF1	mIoU	BF1	mIoU	BF1
Cross Entropy		80.0	62.2	76.2	61.8	77.9	64.7
mIoU	ASL [31]	**80.7**	66.5	**78.4**	66.9	**78.9**	65.7
	Ours	**80.4**	63.8	**78.0**	62.8	**78.5**	64.9
BF1	ASL [31]	6.7	**78.0**	1.4	70.8	1.6	71.8
	Ours	16.0	**77.5**	10.4	**79.2**	11.5	**76.4**

Table 3. Generalization of the searched loss functions for semantic segmentation among different datasets and networks. The losses are originally searched for DeepLabv3+ [8] with ResNet-50 [24] on PASCAL VOC [17]. "R50" and "R101" are the abbreviations of ResNet-50 and ResNet-101, respectively. "DLv3+" and "PSP" denote the DeepLabv3+ and PSPNet [65], respectively.

network. We search for loss functions of the 4 branches simultaneously from scratch. Following [52], we use the intersection, union and enclosing areas between the predicted and ground-truth boxes as the regression loss inputs.

During the search, we train the network with 1/4 of the COCO data for 1 epoch as the proxy task. We further simplify the network by only using the last three feature levels of FPN, and reducing the channels of the detection head by half. After the search procedure, we re-train the detection network with the searched loss functions. The re-training hyper-parameters are the same as the default settings of MMDetection [6]. More details are in Appendix A.2.

Loss Function				mAP
Cls_{RPN}	Reg_{RPN}	Cls_{RCNN}	Reg_{RCNN}	
CE	L1	CE	L1	37.3
CE	L1	CE	IoULoss [63]	37.9
CE	L1	CE	GIoULoss [52]	37.6
CE	L1	CSE-Auto-A [37]		38.5
CE	L1	CSE-RandInit		0.0
CE	L1	Ours		38.0
Ours				38.1

Table 4. Object detection results of ResNet-50 [24] on COCO [34]. Cls and Reg are the classification and regression branches, respectively, where the subscripts RPN and RCNN denote the RPN [51] sub-network and Fast R-CNN [19] sub-network, respectively. CSE-RandInit denotes our implementation of CSE-Autoloss [37] that initializes from random loss function instead of CEI [37] and GIoU [52] losses.

Dataset	COCO	VOC
Network	ResNet-101	ResNet-50
Loss Function	mAP	mAP
CE + L1 + CE + IoULoss [63]	39.7	80.4
Ours	39.9	80.6

Table 5. Generalization of the searched losses for object detection among different datasets and networks. The loss is originally searched for ResNet-50 [24] on COCO [34].

Results. Table 4 compares our searched loss functions with the handcrafted loss functions and the searched function by [37]. The effectiveness of our method is verified for searching on only the 2 branches of the Fast R-CNN [19] sub-network, and on all of the 4 branches. Results show that our searched losses are on par with the existing handcrafted and searched loss functions. Note that [37] designs the search space and strategies specifically for object detection, and fails to find any reasonable loss functions without hand-crafted initialization, while AutoLoss-Zero is a general framework that searches for loss functions from scratch; [31] constructs the search space by parameterizing the evaluation metrics, which can hardly be applied to mAP due to the complicated matching and ranking processes. The formulas of the discovered loss functions are presented in Appendix B. The searched loss function for bounding box regression shares a similar expression with the GIoULoss [52], confirming the effectiveness of the handcrafted loss function.

Generalization of the searched functions. We verify the generalization ability of the searched loss function in Table 5. The loss is originally searched on COCO [34] with ResNet-50 [24], and is used for training networks with different backbone (*i.e.*, ResNet-101) and on different dataset (*i.e.*, PASCAL VOC [17]). The results show that our searched loss functions can generalize well to different object detection networks and datasets.

4.3. Instance Segmentation

Settings. Instance segmentation is the task of detecting the segmentation masks and categories of instances. We also conduct experiments on COCO [34], except that the target metric is mAP with IoU defined on masks.

Mask R-CNN [23] with ResNet-50 [24] and FPN [32] is used as the network. We search for all the 5 loss branches simultaneously. The proxy task is the same as for object detection. We use the default hyper-parameters of MMDetection [6] for re-training. More details are in Appendix A.3.

Results. Table 6 (a) summarizes the results. The loss function searched from scratch by AutoLoss-Zero is on par with the existing manually designed loss functions. The discovered loss functions are presented in Appendix B.

4.4. Pose Estimation

Settings. Pose estimation is the task of localizing human keypoints. Experiments are conducted on COCO [34].

We use [62] with Resnet-50 [24] as the network. Following [12], person detection results provided by [62] are utilized. During search, we train the network for 4 epochs as the proxy task. We re-train the network after the search with the searched loss functions using the default training settings of MMPose [12]. More details are in Appendix A.4.

Results. Table 6 (b) compares our searched loss function with the widely used MSE loss. Starting from randomly initialized loss functions, our searched loss function is slightly better than the MSE loss, demonstrating the effectiveness of AutoLoss-Zero. Appendix B presents the formulas of the discovered loss functions. The searched function learns a regularization term to punish too large prediction values.

Loss Function	mAP
CE + L1 + CE + L1 + CE	34.6
CE + L1 + CE + IoULoss [63] + CE	34.4
CE + L1 + CE + GIoULoss [52] + CE	34.7
Ours	34.8

(a) Instance Segmentation

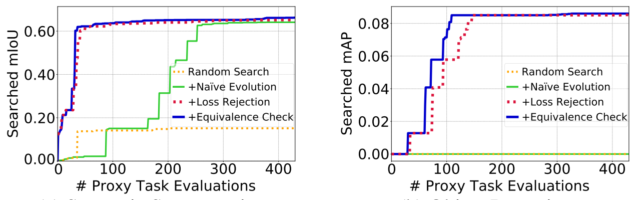
Loss Function	mAP
MSE	71.5
Ours	72.0

(b) Pose Estimation

Table 6. Instance segmentation and pose estimation results of ResNet-50 [24] on COCO [34]. In the first three rows of (a), the five losses correspond to the Cls_{RPN}, Reg_{RPN}, Cls_{RCNN}, Reg_{RCNN} and Mask branches, respectively. "MSE" in (b) denotes the mean square error loss used by [62].

4.5. Search Efficiency

We ablate the search efficiency of AutoLoss-Zero on semantic segmentation with the mIoU metric and object detection with the mAP metric. Figure 4 shows the search process, and Table 7 shows the re-training results. Due to the high sparsity of our search space and the restricted search cost (300 candidate loss function evaluations), no reasonable loss functions can be discovered with random search. Table 8 further presents the number of loss functions explored in 48 hours by AutoLoss-Zero. Over 10^6 loss functions can be explored, ensuring that AutoLoss-Zero can explore the huge and sparse search space within a reasonable time. More discussions are presented in Appendix C.



(a) Semantic Segmentation (b) Object Detection

Figure 4. Ablation study on search efficiency. Each curve presents the averaged scores over the top-5 losses in the current population.

Loss Function	mIoU
Random Search	2.2
Ours	80.7

(a) Semantic Segmentation

Loss Function	mAP
Random Search	0.0
Ours	38.1

(b) Object Detection

Table 7. Re-training results of random search and our algorithm.

	Speed-Up	# Explored Losses
Naïve Evolution	1×	∼300
+ Loss-Rejection Protocol	∼700×	$\sim 2.1\times 10^5$
+ Gradient-Equivalence-Check Strategy	∼1000×	$\sim 3.2\times 10^5$
+ †Stop Training for Invalid Loss Values	∼5000×	$\sim 1.5\times 10^6$

Table 8. Search speed of degenerated variants of AutoLoss-Zero on object detection. "# Explored Losses" demonstrates the number of losses that can be explored in 48 hours. † "Stop Training for Invalid Loss Values" means that the network training is stopped in the first 20 iterations due to invalid loss values (*i.e.*, NaN and Inf).

5. Conclusion

AutoLoss-Zero is a general framework for searching loss functions from scratch for generic tasks. The search space is composed only of basic primitive operators. A variant of evolutionary algorithm is employed for searching, where a loss-rejection protocol and a gradient-equivalence-check strategy are developed to improve the search efficiency. AutoLoss-Zero can discover loss functions that are on par with or superior to existing loss functions on various tasks with minimal human expertise.

Limitations. AutoLoss-Zero still requires certain times of evaluations on the proxy tasks, and the performance may degrade without enough search time. Future work may explore more efficient algorithms to reduce the search time.

Potential Negative Societal Impact. Our search on GPUs may consume lots of electricity and cause increased carbon emissions. Similar to the limitation of search time, this issue can be also alleviated with more efficient search algorithm.

Acknowledgments The work is supported by the National Key R&D Program of China (2020AAA0105200), Beijing Academy of Artificial Intelligence and the Institute for Guo Qiang of Tsinghua University, in part by the General Research Fund through the Research Grants Council of Hong Kong under Grants (Nos. 14204021, 14207319, 14203118, 14208619), in part by Research Impact Fund Grant No. R5001-18, and in part by CUHK Strategic Fund.

References

[1] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016. 2

[2] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018. 1, 2, 7

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: high quality object detection and instance segmentation. *TPAMI*, 2019. 1

[4] Francesco Caliva, Claudia Iriondo, Alejandro Morales Martinez, Sharmila Majumdar, and Valentina Pedoia. Distance map loss penalty term for semantic segmentation. In *MIDL*, 2019. 1, 2, 7

[5] Kean Chen, Weiyao Lin, John See, Ji Wang, Junni Zou, et al. Ap-loss for accurate one-stage object detection. *TPAMI*, 2020. 2

[6] 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. *arXiv preprint arXiv:1906.07155*, 2019. 7, 8, 11, 12, 13

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017. 1

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 6, 7, 11

[9] Cityscapes. https://www.cityscapes-dataset.com/license/. https://www.cityscapes-dataset.com/license/. 13

[10] Creative Commons. Creative commons attribution 4.0 international public license. https://creativecommons.org/licenses/by/4.0/legalcode. 13

[11] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 13

[12] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 8, 12, 13

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 6, 13

[14] Gabriela Csurka, Diane Larlus, Florent Perronnin, and France Meylan. What is a good evaluation measure for semantic segmentation? In *BMVC*, 2013. 6, 11

[15] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019. 1, 2, 3

[16] 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 preprint arXiv:2006.02049*, 2020. 1, 2, 3

[17] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015. 2, 6, 7, 11, 13

[18] Flickr, Inc. Flickr terms & conditions of use. https://www.flickr.com/help/terms, 2020. 13

[19] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 1, 2, 6, 7, 11

[20] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*. Elsevier, 1991. 4, 6, 11

[21] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 6, 11

[22] Tamir Hazan, Joseph Keshet, and David A McAllester. Direct loss minimization for structured prediction. In *NeurIPS*, 2010. 2

[23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 8, 12

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6, 7, 8, 11, 12

[25] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *arXiv preprint arXiv:1908.00709*, 2019. 2

[26] Thorsten Joachims. A support vector method for multivariate performance measures. In *ICML*, 2005. 2

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 12

[28] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 2009. 6, 11

[29] Buyu Li, Yu Liu, and Xiaogang Wang. Gradient harmonized single-stage detector. In *AAAI*, 2019. 1, 2

[30] Chuming Li, Xin Yuan, Chen Lin, Minghao Guo, Wei Wu, Junjie Yan, and Wanli Ouyang. Am-lfs: Automl for loss function search. In *ICCV*, 2019. 1, 3, 4

[31] Hao Li, Chenxin Tao, Xizhou Zhu, Xiaogang Wang, Gao Huang, and Jifeng Dai. Auto seg-loss: Searching metric surrogates for semantic segmentation. In *ICLR*, 2021. 1, 2, 3, 4, 6, 7, 11

[32] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6, 8, 11, 12

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1, 2

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In

ECCV, 2014. 2, 6, 7, 8, 11, 12, 13

[35] Hanxiao Liu, Andrew Brock, Karen Simonyan, and Quoc V Le. Evolving normalization-activation layers. In *NeurIPS*, 2020. 1, 2, 3, 4, 5, 6

[36] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2018. 1, 2, 3

[37] Peidong Liu, Gengwei Zhang, Bochao Wang, Hang Xu, Xiaodan Liang, Yong Jiang, and Zhenguo Li. Loss function discovery for object detection via convergence-simulation driven search. In *ICLR*, 2021. 1, 2, 3, 4, 5, 6, 7, 11, 12

[38] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016. 1, 2, 7

[39] Pritish Mohapatra, Michal Rolinek, CV Jawahar, Vladimir Kolmogorov, and M Pawan Kumar. Efficient optimization for rank-based loss functions. In *CVPR*, 2018. 2

[40] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. A ranking-based, balanced loss function unifying classification and localisation in object detection. In *NeurIPS*, 2020. 2

[41] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018. 1, 2, 3

[42] Hieu Pham and Quoc V Le. Autodropout: Learning dropout patterns to regularize deep networks. *arXiv preprint arXiv:2101.01761*, 2021. 1, 2, 3

[43] Qi Qian, Lei Chen, Hao Li, and Rong Jin. Dr loss: Improving object detection by distributional ranking. In *CVPR*, 2020. 1, 2

[44] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *CVPR*, 2019. 1, 2, 7

[45] Md Atiqur Rahman and Yang Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *ISVC*, 2016. 1, 2

[46] Mani Ranjbar, Tian Lan, Yang Wang, Steven N Robinovitch, Ze-Nian Li, and Greg Mori. Optimizing nondecomposable loss functions in structured prediction. *TPAMI*, 2012. 2

[47] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015. 2

[48] Yongming Rao, Dahua Lin, Jiwen Lu, and Jie Zhou. Learning globally optimized object detector via policy gradient. In *CVPR*, 2018. 2

[49] Esteban Real, Chen Liang, David So, and Quoc Le. Automl-zero: Evolving machine learning algorithms from scratch. In *ICML*, 2020. 3, 4, 5, 6, 7

[50] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2020. 1

[51] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 1, 6, 7, 11

[52] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019. 1, 2, 7, 8, 11

[53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1, 2, 7

[54] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*, 2015. 2

[55] Yang Song, Alexander Schwing, Raquel Urtasun, et al. Training deep neural networks via direct loss minimization. In *ICML*, 2016. 2

[56] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 1

[57] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, Yasemin Altun, and Yoram Singer. Large margin methods for structured and interdependent output variables. *JMLR*, 2005. 2

[58] Xiaobo Wang, Shuo Wang, Cheng Chi, Shifeng Zhang, and Tao Mei. Loss function search for face recognition. In *ICML*, 2020. 1, 3, 4

[59] Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. A study of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1808.08866*, 2018. 2

[60] Lijun Wu, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. Sequence prediction with unlabeled data by reward function learning. In *IJCAI*, 2017. 2

[61] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv:1605.06885*, 2016. 1, 2

[62] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, 2018. 8, 12

[63] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *ACM MM*, 2016. 1, 2, 7, 8

[64] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *SIGIR*, 2007. 2

[65] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1, 6, 7

[66] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *AAAI*, 2020. 1, 2

[67] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *arXiv preprint arXiv:2005.03572*, 2020. 2

[68] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 1, 2, 3