Deep Nets with Subsampling Layers Unwittingly Discard Useful Activations at Test-Time

Chiao-An Yang[1], Ziwei Liu[2], and Raymond A. Yeh[1]

[1] Department of Computer Science, Purdue University
[2] S-Lab, Nanyang Technological University

Abstract. Subsampling layers play a crucial role in deep nets by discarding a portion of an activation map to reduce its spatial dimensions. This encourages the deep net to learn higher-level representations. Contrary to this motivation, we hypothesize that the discarded activations are useful and can be incorporated on the fly to improve models' prediction. To validate our hypothesis, we propose a search and aggregate method to find useful activation maps to be used at test-time. We applied our approach to the task of image classification and semantic segmentation. Extensive experiments over nine different architectures on ImageNet, CityScapes, and ADE20K show that our method consistently improves model test-time performance. Additionally, it complements existing test-time augmentation techniques to provide further performance gains.

1 Introduction

In computer vision, deep nets are commonly trained with the assumption that data samples are drawn independently and identically from an unknown distribution [42]. Following this assumption, it is intuitive that the same model, *i.e.*, same forward pass, should be applied to all samples during both the training and testing time. However, when the assumption is not met then *changing the test time procedure* may lead to better performance. For example, test-time augmentation (TTA) leverages additional prior information, *i.e.*, knowing the suitable augmentations, over the data distribution to improve model performance. For vision models, TTA methods apply random augmentations, *e.g.*, random crops, flips, and rotation, to the test image and perform majority voting to make a final prediction [17, 46, 54, 55, 70]. With the success of TTA, a natural question arises: are there other choices for modifying the test-time procedure?

In this work, we present an orthogonal approach to improving the model performance at test-time. Instead of imposing additional knowledge through data augmentation, we re-examine the ones that are *built into the deep net architecture*. We focus on the knowledge built into subsampling and pooling layers. Our observation is that models with subsampling layers do not utilize activations to their fullest, as some activations are discarded. A question arises, can the discharged activations benefit the model? The main challenges are: (a) identifying which of the discarded activations are useful; and (b) how to incorporate these activations into a test-time procedure.

We formulate an activations search space for a given pre-trained deep net. Each state in this space corresponds to an activations map that can be extracted by choosing different selection indices (s) in the subsampling layers. Given a computation budget, we conduct a greedy search for the set of most promising activation maps based on a confidence criterion. We then aggregate these activation maps using a weighted average to make a final prediction.

We conduct extensive experiments on ImageNet [25] over nine different pre-trained networks, including ConvNets and Vision Transformers, to validate the efficacy of the proposed approach. We also evaluate our method on the task of semantic segmentation using Cityscapes [11] and ADE20K [77] with different segmentation networks, including FCN [36], DeepLab [7, 8], and SegFormer [66]. Overall, we find that our test-time procedure improves both classification and segmentation performance. Additionally, our approach achieves *additional gains* when it is used in conjunction with existing TTA methods, *i.e.*, our approach complements TTA.

Our contributions are as follows:

- We identified that deep nets with subsampling layers discard activations that could be useful for prediction.
- We propose a framework to search over the discarded activations with a learned criterion and aggregate useful ones, via an attention aggregation module, to improve model performance at test-time.
- Extensive experiments on various deep nets demonstrate the effectiveness of the proposed test-time procedure on both image classification and semantic segmentation tasks.

2 Related work

We briefly discuss related work in test-time augmentation, adaptation, ensemble models, and pooling layers.

Test-time augmentation. As our approach can be viewed as a form of Test-Time Augmentation (TTA) but over the set of selection indices, hence we briefly review TTA. At a high level, TTA aims to increase a model's performance at test-time by **at the cost of computation increases.** This is beneficial for tasks where the risk of making a mistake significantly outweighs the computation *e.g.*, medical-related tasks. We note that the computation increase may be very significant. Let's consider image classification on ImageNet, TTA applied to their model includes 144 crops per-image [55], *i.e.*, **an increase 144× in compute.** More recent TTA method [52], further increases the computation to 150×.

The most common form of TTA is to augment the input image and combine the output from each augmentation to make a final prediction for improving task performance. Common augmentations include random cropping [4, 54, 55], flipping [17, 46], or combinations of the augmentations [41]. See Fig. 1 for a comparison of standard, test-time augmentation, and our test-time procedure.

Recent works [10, 16, 23, 31, 38, 41, 52] try to learn the distribution of data augmentation for test-time impacts. Given a set of TTA augmentation, a

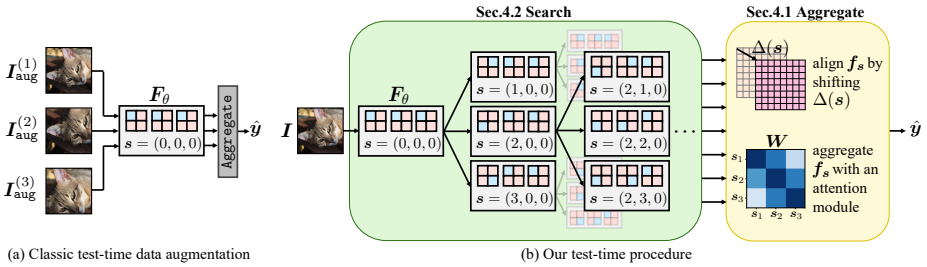

Fig. 1: Comparisons on test-time procedures. (a) In classic test-time augmentation, the output $\hat{\boldsymbol{y}}$ is aggregated from different augmented images $\boldsymbol{I}_{\texttt{aug}}$ feeding into the same model $\boldsymbol{F}_\theta$ with default selection indices $\boldsymbol{s} = (0, 0, 0)$. **(b)** In our procedure, $\hat{\boldsymbol{y}}$ is aggregated over one single image $\boldsymbol{I}$ feeding into $\boldsymbol{F}_\theta$ but activations are extracted over a set of selection indices $\boldsymbol{s}$. We apply a searching algorithm to search for the top-$B_{\texttt{ours}}$ selection indices $\boldsymbol{s}$ based on a scoring function (Sec 4.2). We then aggregate (Sec 4.1) the resulting feature set $\mathcal{F} = \{\boldsymbol{f}_{\boldsymbol{s}}\}$ by first aligning each feature according to $\boldsymbol{s}$ and then merging them using an attention aggregation module.

common approach is to average the resulting logits to make a final prediction. GPS [38] learns to pick the top-k augmentation transforms based on an iterative greedy search. Shanmugam et al. [52] learn a weighted function, deciding which augmentation transform is more important than the other. Chun et al. [10] also studied how to effectively combine/aggregate augmentations' output into a single prediction via the Entropy Weight Method (EWM) [3, 32], *i.e.*, the final prediction is a weighted combination based on entropy. DiffTPT [14] leverages diffusion models to generate high-quality augmented data. TeSLA [57] introduced flipped cross-entropy loss to adapt the pre-trained model during test time. Kim et al. [23] learn to select designated data transformation for each image by introducing an auxiliary module on estimating losses. Unfortunately, we are not able to reproduce their results [23] due to its heavy computation requirement.

It is to be noted that our method differs from the aforementioned works in several ways. First, all TTA methods modify the inputs on the image space using various data transformations. On the other hand, we use the discarded activations in the feature space of the subsampling layers. Second, unlike these TTA methods that learn general domain knowledge, our searching and aggregation procedure adapts to *each image* and generates an instance-based augmented feature. Finally, our learned aggregation can be generalized to different test-time budgets *without* the need for retraining. In contrast, existing works [38, 52] need to retrain their aggregation layer if they change the test-time budget.

Ensemble methods. Classical ensemble methods such as stacking [63] and bagging [6] aggregate predictions from multiple models to create an improved prediction. Recent works [21, 43, 61, 74] focus on how to efficiently learn a set of diverse models. We note that ensemble models require training *multiple models* and running each of them at test time. On the contrary, our approach only requires *a single* pre-trained model (with subsampling layers) to make a prediction. In other words, our approach is orthogonal to the ensemble methods.

If the ensembled deep nets contain subsampling layers, then we can apply our approach on top of it.

Subsampling and pooling layers. Subsampling layers can be traced back to the origin of convolutional neural networks [15, 27] where striding is used to reduce the spatial dimension and increase the networks' receptive field. Another choice to reduce the spatial dimension is pooling layers, *e.g.*, Average Pooling [26], Max-Pooling [47, 69], or other generalizations [48, 51, 53, 72]. To preserve the lost information from downsampling layers, anti-aliased CNN [73] inserts a `max-blur-pool` layer to the pooling layer. In this work, we show that deep nets with subsampling layers can improve their test-time performance by using discarded activations. Additional relevant details and notation of subsampling layers are reviewed in Sec. 3.

3 Preliminaries

We provide a brief review of the subsampling layer and how it is used in deep nets. For readability, we describe these ideas on 1D "images", in practice, they are generalized to 2D activation maps with multiple channels.

Subsampling layers. In its simplest form, a subsampling operation $\texttt{Sub}_R : \mathbb{R}^N \mapsto \mathbb{R}^{\lfloor N/R \rfloor}$ with a subsampling rate $R \in \mathbb{N}$ reduces an input image $\boldsymbol{I}$'s spatial size from N to $\lfloor N/R \rfloor$ following:

$$\texttt{Sub}_R(\boldsymbol{I}; s)[n] = \boldsymbol{I}[Rn + s] \quad \forall n \in \mathbb{Z}, \quad (1)$$

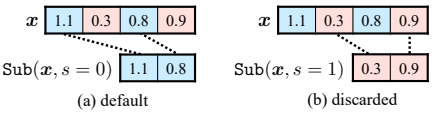

Fig. 2: Subsampling by two.

where $s \in \{0, \ldots, R-1\}$ denotes a selection index. In most deep learning frameworks [1, 5, 45], a subsampling layer defaults to $s = 0$, *i.e.*, when subsampling by a factor of two ($R = 2$) the activations on the even indices (s=0) are kept and the odd indices ($s = 1$) activations are discarded; as illustrated in Fig. 2. In this work, we show that using the discarded activation, *i.e.*, $s \neq 0$, at test-time can lead to performance gains.

Deep nets with subsampling layers. Many deep nets in computer vision contain subsampling layers to reduce spatial dimensions. We take a very generic view toward subsampling layers. Consider a convolution layer with stride two, it can be viewed as a convolution layer with stride *one* followed by a subsampling by a factor of two. Similarly, any pooling operation can be decomposed into a "sliding" operation followed by subsampling. For example, max pooling is equivalent to a sliding max filter followed by a subsampling layer. With this perspective, many notable deep net architectures contain subsampling layers, including early works such as LeNet [27], the popular ResNet [18], MobileNet [19, 20, 50], and the recent vision transformers (ViTs) [13, 33, 34].

Classification formulation. An image classification model with K classes is trained over a dataset $\mathcal{D} = \{(\boldsymbol{I}, \boldsymbol{y})\}$ by minimizing the negative log-likelihood:

$$\mathcal{L}(\mathcal{D}) = -\sum_{(\boldsymbol{I},\boldsymbol{y})\in\mathcal{D}} \sum_{k=1}^{K} \boldsymbol{y}[k] \log(\hat{\boldsymbol{y}}[k](\boldsymbol{I})), \quad (2)$$

where, $\boldsymbol{y}$ denotes the class label in one-hot representation, and $\hat{\boldsymbol{y}}_c(\boldsymbol{I})$ denotes the predicted probability of class c. The deep net making the prediction can be defined as a composition of a feature extractor F_θ and a classifier C_ϕ, *i.e.*,

$$\hat{\boldsymbol{y}}(\boldsymbol{I}) = C_\phi \circ F_\theta(\boldsymbol{I}, \boldsymbol{s}), \text{with } \boldsymbol{s} = \boldsymbol{0}. \tag{3}$$

Consider a feature extractor F_θ consisting of L subsampling layers, each with a subsampling factor $R^{(l)}$. We introduce a selection vector $\boldsymbol{s} \in \mathcal{S}$ to denote indices of the activations, *i.e.*, a tuple of s in Eq. (1) for each layer. The set $\mathcal{S} = \prod_{l=1}^{L}\{0, \ldots, R^{(l)}\}$ denotes the selection index for all the possible activations that can be extracted from a given image. The default forward pass corresponds to using $\boldsymbol{s} = (0, 0, 0, ...) = \boldsymbol{0}$ and all other $\mathcal{S}\backslash\{\boldsymbol{0}\}$ activations are disgarded.

4 Our Approach

Given a trained deep net with subsampling layers, we develop a test-time procedure that improves model performance. The approach is motivated by the observation that subsampling layers discard activations, *e.g.*, a subsampling factor of two on a 2D feature map leads to losing $\frac{3}{4}$ of the spatial activations. As these discarded activations contain information about the input image, we believe they can be incorporated at test-time to improve model performance.

The idea is to keep a subset of the discarded activations and aggregate them into an improved prediction. To do so, we need to answer the following:

(a) **How to better aggregate the activations?** Using all test-time transformations is not always a good idea [38, 52]. Similarly, simply averaging all discarded activations leads to degraded performance.

(b) **Which of the discarded activations to retain?** Naively keeping all the discarded activations leads to an exponential growth in the number of feature maps which is impractical.

To address (a), in Sec. 4.1, we describe how we learn an aggregation function based on the attention mechanism. To address (b), in Sec. 4.2, we discuss how the learned attention values can be used as a search criterion for finding useful discarded activations. An overview of the proposed aggregate and search framework is visually illustrated in Fig. 1(b).

4.1 Aggregating selected activations for prediction

Given a set of indices $\hat{\mathcal{S}}$, an image feature $\boldsymbol{f}_s \in \mathbb{R}^d$ is extracted for each $\boldsymbol{s} \in \hat{\mathcal{S}}$ to form a feature set

$$\mathcal{F} = \{\boldsymbol{f}_{\boldsymbol{s}} \mid \boldsymbol{s} \in \hat{\mathcal{S}}\}, \text{where } \boldsymbol{f}_{\boldsymbol{s}} = F_\theta(\boldsymbol{I}; \boldsymbol{s}) \tag{4}$$

and $F_\theta(\boldsymbol{I}; \boldsymbol{s})$ denotes the deep net's backbone. The size of indices set $|\hat{\mathcal{S}}|$ is equal to a user-specified test-time budget $B_{\texttt{Ours}}$ which corresponds to the number of forward passes needed at test time. These features are combined via an

aggregation function $A : \mathbb{R}^{d \times B_{\texttt{ours}}} \mapsto \mathbb{R}^d$ and passed to the pre-trained classifier C_ϕ to make a prediction $\hat{\boldsymbol{y}} = C_\phi\left(A(\mathcal{F})\right)$.

This aggregation function can be learned, *e.g.*, following the setup of Shanmugam et al. [52] or it can be learning-free, *e.g.*, following the classic TTA where the aggregation A is simply the average of all $\boldsymbol{f_s}$, *i.e.*, $A(\mathcal{F}) = \frac{1}{B_{\texttt{ours}}} \sum_{\boldsymbol{s} \in \hat{\mathcal{S}}} \boldsymbol{f_s}$. We now discuss our proposed learned aggregation function based on attention, and a learning-free aggregation based on entropy.

Learning an aggregation function. We propose a learnable multi-head attention module to be the aggregation function for two reasons: **(a)** The importance of one feature $\boldsymbol{f_s}$ is relative to other features. **(b)** An attention module is a set operator [28] that can take an input of variable size for the feature set $\mathcal{F}$. In other words, the aggregation function can be trained on one fixed budget and evaluated at *any arbitrary testing budget*. Contrarily, the learned aggregation function proposed by Shanmugam et al. [52] is retrained for each test-time budget.

For each $\boldsymbol{f_s}$, we learn its query $\boldsymbol{q_s}$, key $\boldsymbol{k_s}$, and value vector $\boldsymbol{v_s}$ using a fully connected layer. The attention matrix $\boldsymbol{W}$ is obtained by computing the inner product and normalizing through a softmax among the queries and keys for all $(\boldsymbol{s}, \boldsymbol{s}')$ pair, *i.e.*,

$$W_{\boldsymbol{s}\boldsymbol{s}'} = \exp(\boldsymbol{q_s}^\intercal \boldsymbol{k_{s'}}) / \left(\sum_{\boldsymbol{s}'' \in \hat{\mathcal{S}}} \exp(\boldsymbol{q_s}^\intercal \boldsymbol{k_{s''}}) \right). \tag{5}$$

The attention module's output is then used to learn an offset from the average features, *i.e.*,

$$A_{\texttt{learned}}(\mathcal{F}) = \frac{1}{B_{\texttt{ours}}} \sum_{\boldsymbol{s} \in \hat{\mathcal{S}}} \left(\boldsymbol{f_s} + \texttt{MLP} \sum_{\boldsymbol{s}' \in \hat{\mathcal{S}}} W_{\boldsymbol{s}\boldsymbol{s}'} \boldsymbol{v_{s'}} \right), \tag{6}$$

to obtain the aggregated feature over the set $\hat{\mathcal{S}}$. Here, $\texttt{MLP}$ denotes an MLP model.

Learning-free aggregation. Inspired by Chun et al. [10], we use the entropy of the logits to quantify the confidence of the model in its prediction and weight each activation accordingly. Lower entropy indicates the more confident the model is for a prediction and thus it should receive a higher weight. We propose to weight each $\boldsymbol{f_s}$

$$w_{\boldsymbol{s}} = \frac{1}{Z_{\boldsymbol{s}}} \left(1 - \frac{H(C_\phi(\boldsymbol{f_s}))}{\log K} \right), \tag{7}$$

where $H(\cdot)$ is the entropy function, K is the number of classes, and $Z_{\boldsymbol{s}}$ is the normalization term such that $\sum_{\boldsymbol{s}} w_{\boldsymbol{s}} = 1$. Overall, our training-free aggregation is as follows,

$$A_{\texttt{entropy}}(\mathcal{F}) = \sum_{\boldsymbol{s} \in \hat{\mathcal{S}}} w_{\boldsymbol{s}} \boldsymbol{f_s}. \tag{8}$$

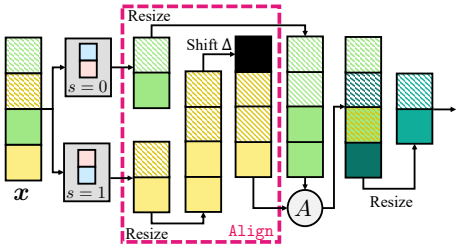

Fig. 3: Illustration of $\texttt{Align}_s$ with a single subsampling layer.

Spatial alignment ($\texttt{Align}_s$). To properly aggregate feature maps $\boldsymbol{f_s}$ from different states $\boldsymbol{s}$, we find it more beneficial to consider their shifting in feature space before aggregation to avoid spatial mismatch and adjust Eq. (4) to

$$\mathcal{F} = \{\boldsymbol{f_s} \mid \boldsymbol{s} \in \hat{\mathcal{S}}\}, \text{where } \boldsymbol{f_s} = \texttt{Align}_{\boldsymbol{s}}(F_\theta(\boldsymbol{I}; \boldsymbol{s})). \tag{9}$$

We perform an alignment based on the relative shift with respect to the input resolution. $\texttt{Align}_s$ first resizes the activation maps to the input resolution, then shifts perform a shift Δ relative to the default activation $F_\theta(\boldsymbol{I}; \boldsymbol{0})$. In Fig. 3, we illustrate this alignment using an input of length 4 with a single subsampling by 2. The green activations are extracted with $\boldsymbol{s} = 0$ and the yellow activations ($\boldsymbol{s} = 1$) are aligned by shifting with $\Delta = 1$. For a specific $\boldsymbol{s}$, we can compute the shift Δ as follows:

$$\begin{aligned} \Delta = & \quad \textstyle\sum_{l=1}^{L} s_l \left(\prod_{l'=1}^{l} R^{(l')} \right) \\ & = s_1 + s_2 R^{(1)} + s_3 R^{(1)} R^{(2)} + \cdots, \end{aligned} \tag{10}$$

where $\boldsymbol{s} = (s_1, s_2, \ldots, s_L)$ denotes the selection indices and $R^{(l)}$ denotes the subsampling rate at the l^{th} layer.

4.2 Searching for useful activations

We have described how we aggregated a set of features $\hat{\mathcal{S}}$. In this section, we will describe how to find this set of indices where the corresponding activations would benefit the model performance from the discarded set $\mathcal{S}\backslash\{0\}$. As its size $|\mathcal{S}|$ grows exponentially with the number of subsampling layers, naively iterating for all activations would be expensive. To address this, we propose a greedy search to

Algorithm 1 `Search` for activations (Top-down)

Inputs: $\boldsymbol{I}$, $\mathcal{S}$, `Criterion`, Budget $B_{\texttt{Ours}}$
$Q \leftarrow$ Init an empty priority queue
$E \leftarrow$ Init a dictionary of empty set $\forall \boldsymbol{s} \in \mathcal{S}$
$\hat{\mathcal{S}} = \{\}$ # Keeps track of returned states
Q.insert($(\boldsymbol{0}, 0)$)
while $|\hat{\mathcal{S}}| \leq B_{\texttt{Ours}}$ **do**
  $\boldsymbol{s} = Q$.pop()
  $l = \min(\{1, \ldots, L\} \backslash E[\boldsymbol{s}])$ # *Top-down*
  $E[\boldsymbol{s}]$.add(l) # *Keeps track of expanded* l
  for all $\boldsymbol{s}' \in$ Neighbors($\boldsymbol{s}, l$) **do**
    # Q sorted by `Criterion`
    Q.insert($(\boldsymbol{s}', \texttt{Criterion}(\boldsymbol{s}', \boldsymbol{I}))$)
    $\hat{\mathcal{S}}$.add($\boldsymbol{s}'$)
Return: $\hat{\mathcal{S}}$

gradually grow a set of selected activations within a given compute budget $B_{\texttt{Ours}}$. The high-level idea of the search algorithm is summarized in Alg. 1 and visualized in Fig. 4.

Search procedure. We formulate the task of finding a set of promising activation maps $\hat{\mathcal{S}}$ as a search problem over the state space $\mathcal{S} = \prod_{l=1}^{L}\{0, \dots, R^{(l)} - 1\}$ where each state $\boldsymbol{s} = (s_1, s_2, \dots, s_L)$ corresponds to a tuple of selection index for each of the subsampling layer l in the pre-trained deep net. The `Search`, in Alg. 1, returns all the states it has visited within a computation budget $B_{\texttt{ours}}$.

To implement this search, we utilize a priority queue Q (lowest values are popped first) to determine which of the following states is more promising to visit next. The priority queue is sorted based on the values computed from a $\texttt{Criterion}(\boldsymbol{f_s})$ function. In theory, this criterion can be any function that maps a feature map $\boldsymbol{f_s}$ to a real number. Later in this section, we describe our proposed learned and learning-free criteria.

For each visited state $\boldsymbol{s}$, we then add its "neighboring states" for a subsampling layer l. We define the neighbors of state $\boldsymbol{s} = \mathbf{0}$ at the 0^{th} layer is the set $\cup_{i\in(1,2,\dots R^{(0)})}\{(i, 0, 0, \dots)\}$. The expanded layer l is selected in a top-down fashion. Finally, we use a dictionary E to keep track of the expanded layers for each state to avoid redundancies.

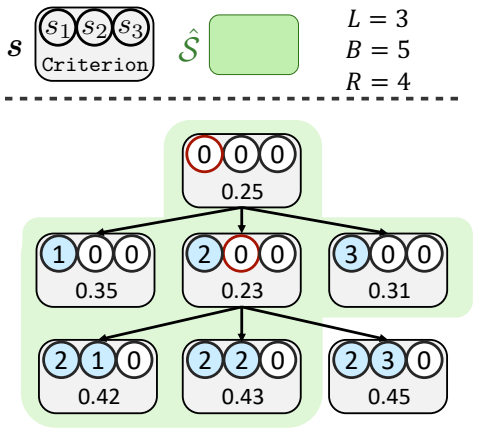

Fig. 4: Search for activations. From the initial state $(0, 0, 0)$, we add its 3 neighbors ($l = 1$) in a top-down fashion. Next, state $(2, 0, 0)$ has the lowest criterion, hence we further add its 3 neighbors ($l = 2$). Finally, the lowest-$B_{\texttt{ours}}$ states are returned in $\hat{\mathcal{S}}$. We keep track of the expanded l in a dictionary E.

Learned criterion. We prioritize expanding the node with the highest attention score from the attention $\boldsymbol{W}$ in Eq. (5) since the correspondent feature contributes the most in the final aggregated feature. Thus, we choose

$$\texttt{Criterion}_{learned}(\boldsymbol{f_s}) = (\sum_{\boldsymbol{s'}\in\hat{\mathcal{S}}} W_{\boldsymbol{s}\boldsymbol{s'}})^{-1}. \tag{11}$$

Learning-free criterion. As in the aggregation, we found that entropy H is a suitable choice for finding useful features, *i.e.*, the search should emphasize on high-confidence regions of the feature space, therefore we choose

$$\texttt{Criterion}_{entropy}(\boldsymbol{f_s}) = \sum_{k=1}^{K} \hat{\boldsymbol{y}}_{\boldsymbol{s}}[k] \log \hat{\boldsymbol{y}}_{\boldsymbol{s}}[k] = H(\hat{\boldsymbol{y}}_{\boldsymbol{s}}), \tag{12}$$

where $\hat{\boldsymbol{y}}_{\boldsymbol{s}} = C_\phi(\boldsymbol{f_s})$ corresponds to the predicted probability from the classifier.

Table 1: Comparison to TTA methods on ImageNet with expanded [52] TTA policies. We evaluate on ImageNet under $B_{\texttt{total}} \in \{30, 100, 150\}$ with various model architectures. For each $B_{\texttt{total}}$, we report the top-1 Acc. of baseline TTA methods with and without our learned approach. Whichever is the better one is bolded. The results of our learned procedure are highlighted.

TTA	Ours	ResNet18			ResNet50			MobileNetV2			InceptionV3		
		30	100	150	30	100	150	30	100	150	30	100	150
GPS [38]	✗	70.51	70.51	70.51	76.50	76.50	76.50	72.24	72.24	72.24	71.48	71.48	71.48
	✓	**70.74**	**70.74**	**70.69**	**76.74**	**76.84**	**76.87**	**72.37**	**72.61**	**72.58**	**71.86**	**72.05**	**72.02**
ClassTTA [52]	✗	69.09	68.23	66.40	75.40	74.88	73.56	70.58	69.97	67.81	70.80	70.39	70.34
	✓	**70.37**	**70.36**	**70.37**	**76.58**	**76.61**	**76.65**	**71.44**	**71.68**	**71.63**	**71.93**	**71.99**	**72.00**
AugTTA [52]	✗	70.55	70.66	70.28	76.54	76.59	76.47	72.33	72.42	72.46	71.65	71.88	71.98
	✓	**70.75**	**70.79**	**70.74**	**76.76**	**76.84**	**76.89**	**72.41**	**72.62**	**72.58**	**72.09**	**72.24**	**72.24**

Extension to semantic segmentation. To extend the classification formulation to semantic segmentation, we view semantic segmentation as a per-pixel classification problem. For example, the entropy will then be computed for each pixel location. To aggregate the selected activations, we compute a per-pixel weight map for each activation instead of a scalar weight for the entire activation map.

5 Experiments

To validate the proposed test-time procedure, we conduct experiments on two computer vision tasks: image classification, and semantic segmentation. In the following, we provide the experiment setup and implementation details before discussing the results.

5.1 Image classification

Experiment setup. Recent TTA methods on image classification focus on learning TTA. Hence, we focus on comparison using the learned aggregation and criterion from our approach. The results of our learning-free version are included in the appendix.

We evaluate our learned methods on two image classification datasets, namely ImageNet [25] and Flowers102 [44]. We provide experiments on additional datasets [24] in the appendix. For ImageNet, we use the pre-trained weights released by Pytorch. We randomly selected 20,000 and 5,000 images from `train` of ImageNet to be used for training and validation of the aggregation function. We report results on the `val` set of ImageNet. For Flowers102, we fine-tune the pre-trained ImageNet weights from Pytorch on Flowers102 using the official training/validation/testing split for Flowers102. For both ImageNet and Flowers102, each image is first resized to 256px and cropped to 224px.

Please note, this experiment setup differs from Shanmugam et al. [52] as their setup is non-conventional. Shanmugam et al. [52] resplit ImageNet's `val`

Table 2: Comparison to TTA methods on Flowers102 with expanded [52] TTA policies. We evaluate on Flowers102 under $B_{\texttt{total}} \in \{30, 100, 150\}$ with various model architectures. For each $B_{\texttt{total}}$, we report the top-1 Acc. of baseline TTA methods with and without our learned approach. Whichever is the better one is bolded. The results of our learned procedure are highlighted.

TTA	Ours	ResNet18			ResNet50			MobileNetV2			InceptionV3		
		30	100	150	30	100	150	30	100	150	30	100	150
GPS [38]	✗	**89.04**	89.04	89.04	**91.12**	**91.12**	91.12	89.85	89.85	89.85	87.69	87.69	87.69
	✓	88.93	**89.20**	**89.19**	91.05	91.02	**91.17**	**89.90**	**90.10**	**90.05**	**87.95**	**87.93**	**87.79**
ClassTTA [52]	✗	87.97	87.81	86.39	90.84	90.19	90.06	89.41	88.00	85.54	87.10	87.41	84.81
	✓	**88.97**	**89.15**	**89.06**	**91.04**	**90.97**	**91.02**	**89.49**	**89.58**	**89.67**	**87.43**	**87.43**	**87.46**
AugTTA [52]	✗	88.73	88.86	88.55	90.91	90.84	90.81	**90.10**	90.00	89.82	87.35	87.43	87.43
	✓	**89.02**	**89.55**	**89.55**	**91.14**	**90.93**	**90.97**	90.08	**90.13**	**90.11**	**87.62**	**87.62**	**87.79**

set into train/validation/test sets. In other words, *their models are trained on a subset of ImageNet's original* `val` *set*. They also reported non-conventional "multiple runs". For each "run", they evaluate *the same trained model* over different test subsets splits, where each split is sampled (with replacement) from their ImageNet's test split[3]. Instead of following the unconventional setup, we conduct experiments using their released code on standard `val` split for each of the datasets. For multiple runs, each experiment is repeated 5 times with different random seeds, where the *training split* is resampled, and the aggregation module is reinitialization. Overall, we observe the standard deviations for our method to be less than 0.05%.

Implementation details. Following Shanmugam et al. [52], we choose ResNet18, ResNet50 [18], MobileNetV2 [50], and InceptionV3 [56] to test the generalizability of our procedure on multiple backbones. Please refer to the supplementary for our definition of F_θ and C_ϕ and more results on additional backbones, *e.g.* ResNext50 [67], ShuffleNetV2 [39], Swin [34], and SwinV2 [33].

To train our aggregation module A, we fix the budget $B_{\texttt{ours}} = 30$ during training. To encourage the model to expand unseen nodes rather than focusing on seen ones, during training, we randomly sampled the nodes with probability inverse-correlated to its `Criterion`. We use the AdamW optimizer [37] and the cosine-annealing scheduler. The initial learning rate on ImageNet and Flowers102 is set to $1e^{-6}$.The training batch sizes are 32 for all datasets.

Baselines. For baselines, we select several state-of-the-art TTA methods, namely GPS [38], AugTTA [52], and ClassTTA [52]. For a fair comparison, each method is evaluated under the same test-time budget $B_{\texttt{total}}$, *i.e.*, the number of total forward passes required to generate the final outputs.

Both baseline TTA methods and our procedure required their test-time budget. We denote the former one as $B_{\texttt{tta}}$ and the latter one as $B_{\texttt{ours}}$. Since the baseline TTA methods and ours are independent of each other, the overall budget $B_{\texttt{total}}$ is the product of $B_{\texttt{tta}}$ and $B_{\texttt{ours}}$. For example, when comparing under the budget

[3]Please see their code `github.com/divyashan/test-time-augmentation`) at `utils/evaluate.py` : `L58`.

Table 3: Ablation on searching method. We report the top-1 Acc. (%) and latency (img/s) on ImageNet with different choices of search space under $B_{\texttt{ours}} = 30$.

Alg. 1: line 7	Acc.↑	Latency↓
$\{1, \cdots, L\}$	79.64	53.46
$\{1, \cdots, L-1\}$	79.52	25.06
$\{2, \cdots, L\}$	**80.06**	25.84
$\{2, \cdots, L-1\}$	<u>79.88</u>	21.21

Table 4: Ablation on aggregation. We report the top-1 Acc. (%) and latency (img/s) on ImageNet with different choices of $A(\mathcal{F})$ under $B_{\texttt{ours}} = 30$.

$A(\mathcal{F})$	Acc.↑	Latency↓
`Avg`	79.38	19.55
`Entropy`	79.44	19.56
Ours w/o `Align`	<u>79.52</u>	20.09
Ours (w/ `Align`)	**79.88**	21.21

Table 5: Ablation on searching criterion. We report the top-1 Acc. (%) and latecny (img/sec) on ImageNet with different choices of $\texttt{Criterion}(\boldsymbol{f_s})$ under $B_{\texttt{ours}} = 30$.

$\texttt{Criterion}(\boldsymbol{f_s})$	Acc.↑	Latency↓
Random	79.80	13.53
Δ	79.72	12.65
$H(\hat{\boldsymbol{y}})$ (Eq. (12))	<u>79.86</u>	21.45
Ours (Eq. (11))	**79.88**	21.21

$B_{\texttt{total}} = 150$ in Tab. 1, $(B_{\texttt{tta}}, B_{\texttt{ours}}) = (150, 1)$ is selected for the experiments w/o Ours, while $(B_{\texttt{tta}}, B_{\texttt{ours}}) = (10, 15)$ is selected for the experiments w/ Ours. Please refer to the supplementary for more detailed configuration.

We follow the `expanded` TTA policy settings [52] and build a TTA pool containing various data transformations, such as `Flip`, `Colorization`, or `Blur`. The original `expanded` policy [52] fixes $B_{\texttt{tta}}$ at 128. We increase the possible range of $B_{\texttt{tta}}$ for `expanded` policy to cover up to 1000. Please refer to supplementary for the detailed policy.

Comparision to TTA. We report the comparison between the baseline TTA methods with and without ours in Tab. 1 and 2 on ImageNet and Flowers102 respectively.

We report the performance under $B_{\texttt{total}} = 30$, 100, and 150 in Tab. 1. We observe that the baseline TTA methods do not scale well when increasing $B_{\texttt{total}}$. The performance of GPS remains the same since the number of budget is fixed in their design; the gains on AugTTA are often minuscule or slightly negative; while ClassTTA usually suffers from using more budget due to its difficulty in converging. Our approach instead achieves consistent gain when using higher $B_{\texttt{total}}$. Moreover, ours outperforms its counterpart in all cases. On average, we improve the top-1 Acc. of TTA baselines by 0.87%. Specifically, we improve GPS on average by 0.32%, ClassTTA by 2.01%, and AugTTA by 0.19%.

In Tab. 2, we report the performance under the same settings on Flowers102. The baseline TTA methods do not benefit much from using more $B_{\texttt{total}}$. Our approach outperforms its counterpart in most cases while remaining competitive in the rest. For example, we achieved the top-1 Acc. over all baselines by 0.57%.

Ablation study. We conduct ablation studies using ResNet-18 as the base architecture and report the performance of our method in our validation split of ImageNet without any TTA. We use $B_{\texttt{ours}}$=30 in these studies since the difference between different choices can be minuscule when the budget is small and the searching process is short.

In Tab. 3, we test other choices for searching the space $\mathcal{S}$ in Alg. 1: line 8. Specifically, we consider limiting the search space $\{1, \cdots, L\}$ by considering fewer layers. We ablate by removing the expansion on the first layer ($l = 1$). We hypothesize that the layer has a small offset and therefore should have little

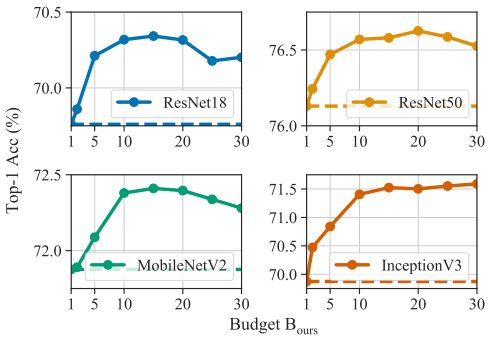

Fig. 5: Acc. vs. budget. We observe an initial gain in Acc. when increasing the budget $B_{\texttt{ours}}$. The improvement plateaus when $B_{\texttt{ours}}$ reaches about 15.

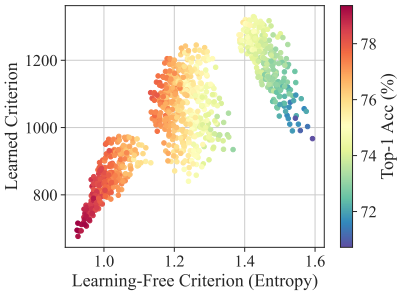

Fig. 6: $\texttt{Criterion}_{\texttt{learned}}(\boldsymbol{f}_{\boldsymbol{s}})$ **vs.** $H(\hat{\boldsymbol{y}}_{\boldsymbol{s}})$ **vs. Accuracy.** Each point represents an experiment in which one sole $\boldsymbol{s} \in \hat{\mathcal{S}}$ is used for prediction and its associated top-1 Acc. over its learned criterion and entropy.

impact on final outputs. On the other hand, we can omit the expansion on the last layer ($l = L$) because the large alignment necessary from the aggregation deteriorates the quality of the activations. Limiting the search space also greatly decreases the latency of our approach due to faster spatial alignment.

Overall, we observe that it is beneficial to limit the search space for better performance and faster speed. Overall, we find out that top-down search while omitting the last layer has the best performance in image classification. We choose to omit both the first and last layer as our default option because it balances between good performance and fast latency.

In Tab. 4, we consider other aggregation methods besides our proposed one in Eq. (6). The baseline is to simply average all activations. We report the performance of training-free aggregation by entropy-weighting in Eq. (8). Finally, we ablate the `Align` in Eq. (6) to show the importance of aligning feature maps before aggregation. Our proposed learned aggregation yields the best performance. Additionally, without aligning feature maps, the performance drops significantly by 0.36% in top-1 accuracy.

In Tab. 5, we test other choices for `Criterion`, determining which $\boldsymbol{s}'$ to expand first in Alg. 1. The baseline is to expand purely by random. One can suggest choosing $\boldsymbol{s}'$ based on the offsets $\boldsymbol{\Delta}$ (see Eq. (10)). We show that expanding the $\boldsymbol{s}'$ with the lowest entropy (most confident prediction) is competitive when training is not feasible. Using learned attention $\boldsymbol{W}$ from the aggregation module as the criterion yields the best performance.

Analysis on budget $B_{\texttt{ours}}$. In Fig. 5, we study the effect of the budget $B_{\texttt{ours}}$ to our procedure on the selected backbones. We report the top-1 Acc. on our ImageNet testing split w/o any TTA methods. Our method steadily gains improvements on its own when we increase the budget $B_{\texttt{ours}}$ from 1 to roughly 10. Performance gains are mostly saturated when the budget reaches 20, *i.e.*, more budgets yield limited gains.

Table 6: Cityscapes and ADE20K results. We consider various encoder backbones and decoders w/ and w/o our procedure under different $B_{ours} \in \{4, 10\}$ and TTA (horizontal flip). We report the mIoU score on both Cityscapes and ADE20K semantic segmentation. The results of our non-learned procedure are highlighted. Our procedure makes improvements on all combinations of architectures in both datasets. We also show that ours can improve the results of horizontal flip, proofing the complementary between our approach and standard TTA.

Dataset	HFlip	ResNet50-FCN			ResNet50-DeepLab			MobileNet-FCN			MobileNet-DeepLab			MiT-SegFormer		
		✗	4	10	✗	4	10	✗	4	10	✗	4	10	✗	4	10
CityScape	✗	72.35	72.50	**72.56**	79.60	79.72	**79.73**	71.19	71.27	**71.83**	75.32	75.58	**75.60**	76.54	77.01	**77.05**
	✓	72.71	72.88	**72.97**	80.08	**80.15**	80.09	71.69	72.03	**72.10**	75.56	75.75	**75.76**	76.84	77.11	**77.12**
ADE20K	✗	35.94	36.12	**36.20**	42.72	42.77	**42.81**	19.55	**19.57**	19.56	33.92	34.01	**34.09**	37.41	**37.68**	37.67
	✓	36.31	36.46	**36.50**	43.02	**43.07**	43.06	19.72	19.72	**19.73**	34.18	34.20	**34.28**	38.03	**38.33**	38.28

We provide additional results on various pre-trained backbones [62] in the appendix, *e.g.* MobileNetV3 [19] Multi-scale ViT (MViTv2) [29, 29], DenseNet [22], VGG [54], RepVGG [12], DeiT [58], CoaT [68], ConvNeXTV2 [35, 64], XCiT [2], VOLO [71], PvTV2 [59], PvTV2 [60], Efficientformer [30], and ShuffleNet [75].

Analysis of the proposed criteria. We present a visualization in Fig. 6 demonstrating that our learned attention $\boldsymbol{W}$ in Eq. (6) is a good selection criterion. For the set of selection indices $\boldsymbol{s} \in \mathcal{S}$ over ImageNet (validation), we plot out the accuracy of its associated top-1 Acc. verses its learned criterion in Eq. (11) and the learning-free criterion based on entropy Eq. (12). That is, each point represents an experiment in which one sole $\boldsymbol{s} \in \hat{\mathcal{S}}$ is used for prediction and its associated top-1 Acc. Empirically, we observe that indices $\boldsymbol{s}$ with lower $\texttt{Criterion}(\boldsymbol{f_s})$ tend to have better accuracy, hence it is reasonable to prioritize them when selecting $\hat{\mathcal{S}}$.

5.2 Semantic segmentation

Experiment setup. As semantic segmentation has not been studied in learned TTA baselines [38, 52], we consider the non-learned TTA setting for this section. We conduct experiments using the non-learned version of our approach on CityScapes and ADE20K datasets [11, 76, 77]. We test on several semantic segmentation backbones and decoders, including ResNets [18], MobileNets [20], FCN [36], DeepLab [7, 8], and MiT+SegFormer [66].

Implementation details. Our implementation is based on MMSeg (OpenMMLab Segmentation) [40]. It provides pre-trained weights and benchmarks on Cityscapes [11] and ADE20k [76, 77]. As for the searching strategy, different from the observation in image classification, we do not find it beneficial to limit the search space of Alg. 1. All search spaces remain unchanged as $\{1, 2, \cdots, L\}$. For evaluation metrics, we report the mIoU score on both datasets.

Results. In Tab. 6, we report the results on Cityscapes and ADE20K semantic segmentation. We test with various combinations of encoder backbones and decoders for semantic segmentation to show that our method generalizes

over architectures, including both conv. backbones (MobileNet, ResNet-50) and transformer (MiT) backbones.

We observe that our approach improves the performance over baseline models on all combinations of encoder backbones and decoders. The largest performance gains are 0.51 on Cityscapes mIoU with MiT + SegFormer ($B_{\texttt{ours}} = 10$) and 0.27 on ADE20K mIoU with MiT + SegFormer ($B_{\texttt{ours}} = 4$). These experiments show that our method can be extended to semantic segmentation with consistent improvements.

Comparison to TTA. In Tab. 6, we also report the comparison to HorizotalFlip, which is a popular choice for TTA on semantic segmentation. Our test-time procedure further improves the performance when TTA is used. We observe a 0.30 gain in mIoU on MiT+Segformer ($B_{\texttt{ours}} = 4$). The results show that our approach is orthogonal to TTA.

We provide additional results on various segmentation backbones in the appendix, *e.g.* MobileNetV3+LRASPP [19], UNet [49], Swin [34], UperNet [65], and Twins [9].

5.3 Discussion & limitation

On both image classification and semantic segmentation, our proposed test-time procedure consistently demonstrates improvements over the baselines on a wide range of deep net architectures. We further point out that the achieved gains are considered significant in image classification and segmentation. However, the improved performance does come at a cost.

The limitation common among all test-time procedures is the increase in test-time computation. Multiple forward passes are needed to make a final prediction, and our method is no different. The increase in test-time computation means that the approach is not suitable for applications with real-time or low-power constraints. On the flip side, our method will be desirable for tasks where the accuracy is substantially more important than the compute time, *e.g.*, medical-related tasks. For these tasks, we believe our method is an appealing option to further improve performance.

6 Conclusion

We propose a test-time procedure that leverages the activation maps originally discarded by the subsampling layers. By solving a search problem to identify useful activations and then aggregating them together, via a learned aggregation module and criterion, the model can make more accurate predictions at test-time. On image classification and semantic segmentation tasks, we show that our approach is effective over nine different architectures. Additionally, it complements existing test-time augmentation approaches. These results suggest that our approach is a compelling method in addition to the existing testing TTA framework, especially for tasks that do not have real-time constraints.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: OSDI (2016)
2. Ali, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., et al.: Xcit: Cross-covariance image transformers. In: Proc. NeurIPS (2021)
3. Amiri, V., Rezaei, M., Sohrabi, N.: Groundwater quality assessment using entropy weighted water quality index (EWQI) in Lenjanat, Iran. Environmental Earth Sciences (2014)
4. Bahat, Y., Shakhnarovich, G.: Classification confidence estimation with test-time data-augmentation. arXiv e-prints pp. arXiv–2006 (2020)
5. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q.: JAX: composable transformations of Python+NumPy programs (2018), URL http://github.com/google/jax
6. Breiman, L.: Bagging predictors. Machine learning (1996)
7. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
8. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proc. ECCV (2018)
9. Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., Shen, C.: Twins: Revisiting spatial attention design in vision transformers. arXiv preprint arXiv:2104.13840 (2021)
10. Chun, S., Lee, J.Y., Kim, J.: Cyclic test time augmentation with entropy weight method. In: Proc. UAI (2022)
11. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes dataset for semantic urban scene understanding. In: Proc. CVPR (2016)
12. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: RepVGG: Making VGG-style ConvNets great again. In: Proc. CVPR (2021)
13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: Proc. ICLR (2021)
14. Feng, C.M., Yu, K., Liu, Y., Khan, S., Zuo, W.: Diverse data augmentation with diffusions for effective test-time prompt tuning. In: Proc. ICCV (2023)
15. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics (1980)
16. Gaillochet, M., Desrosiers, C., Lombaert, H.: Taal: Test-time augmentation for active learning in medical image segmentation. In: MICCAI Workshop on Data Augmentation, Labelling, and Imperfections (2022)

17. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proc. ICCV (2017)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR (2016)
19. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for MobileNetV3. In: Proc. ICCV (2019)
20. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
21. Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J.E., Weinberger, K.Q.: Snapshot ensembles: Train 1, get m for free. In: Proc. ICLR (2017)
22. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proc. CVPR (2017)
23. Kim, I., Kim, Y., Kim, S.: Learning loss for test-time augmentation. In: Proc. NeurIPS (2020)
24. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proc. NeurIPS (2012)
26. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Handwritten digit recognition with a back-propagation network. In: Proc. NeurIPS (1989)
27. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Shape, contour and grouping in computer vision (1999)
28. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: Proc. ICML (2019)
29. Li, Y., Wu, C.Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., Feichtenhofer, C.: MViTv2: Improved multiscale vision transformers for classification and detection. In: Proc. CVPR (2022)
30. Li, Y., Yuan, G., Wen, Y., Hu, J., Evangelidis, G., Tulyakov, S., Wang, Y., Ren, J.: Efficientformer: Vision transformers at mobilenet speed. In: Proc. NeurIPS (2022)
31. Li, Z., Kamnitsas, K., Dou, Q., Qin, C., Glocker, B.: Joint optimization of class-specific training-and test-time data augmentation in segmentation. IEEE Transactions on Medical Imaging (2023)
32. Liu, L., Zhou, J., An, X., Zhang, Y., Yang, L.: Using fuzzy theory and information entropy for water quality assessment in Three Gorges region, China. Expert Systems with Applications (2010)
33. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al.: Swin transformer v2: Scaling up capacity and resolution. In: Proc. CVPR (2022)

34. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proc. ICCV (2021)
35. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proc. CVPR (2022)
36. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proc. CVPR (2015)
37. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Proc. ICLR (2019)
38. Lyzhov, A., Molchanova, Y., Ashukha, A., Molchanov, D., Vetrov, D.: Greedy policy search: A simple baseline for learnable test-time augmentation. In: Proc. UAI (2020)
39. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proc. ECCV (2018)
40. MMSegmentation Contributors: MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. https://github.com/open-mmlab/mmsegmentation (2020)
41. Moshkov, N., Mathe, B., Kertesz-Farkas, A., Hollandi, R., Horvath, P.: Test-time augmentation for deep learning-based cell segmentation on microscopy images. Scientific reports (2020)
42. Murphy, K.P.: Probabilistic machine learning: an introduction. MIT press (2022)
43. Nam, G., Yoon, J., Lee, Y., Lee, J.: Diversity matters when learning from ensembles. In: Proc. NeurIPS (2021)
44. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: Indian conference on computer vision, graphics & image processing (2008)
45. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Proc. NeurIPS (2019)
46. Pavllo, D., Feichtenhofer, C., Grangier, D., Auli, M.: 3D human pose estimation in video with temporal convolutions and semi-supervised training. In: Proc. CVPR (2019)
47. Ranzato, M., Huang, F.J., Boureau, Y.L., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: Proc. CVPR (2007)
48. Rojas-Gomez, R.A., Lim, T.Y., Schwing, A., Do, M., Yeh, R.A.: Learnable polyphase sampling for shift invariant and equivariant convolutional networks. In: Proc. NeurIPS (2022)
49. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Proc. MICCAI (2015)
50. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proc. CVPR (2018)

51. Sermanet, P., Chintala, S., LeCun, Y.: Convolutional neural networks applied to house numbers digit classification. In: Proc. ICPR (2012)
52. Shanmugam, D., Blalock, D., Balakrishnan, G., Guttag, J.: Better aggregation in test-time augmentation. In: Proc. ICCV (2021)
53. Shi, Z., Ye, Y., Wu, Y.: Rank-based pooling for deep convolutional neural networks. Neural Networks (2016)
54. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proc. ICLR (2015)
55. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proc. CVPR (2015)
56. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proc. CVPR (2016)
57. Tomar, D., Vray, G., Bozorgtabar, B., Thiran, J.P.: Tesla: Test-time self-learning with automatic adversarial augmentation. In: Proc. CVPR (2023)
58. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: Proc. ICML (2021)
59. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proc. ICCV (2021)
60. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pvt v2: Improved baselines with pyramid vision transformer. Computational Visual Media (2022)
61. Wen, Y., Tran, D., Ba, J.: BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In: Proc. ICLR (2020)
62. Wightman, R.: Pytorch image models. https://github.com/rwightman/pytorch-image-models (2019), https://doi.org/10.5281/zenodo.4414861
63. Wolpert, D.H.: Stacked generalization. Neural networks (1992)
64. Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I.S., Xie, S.: Convnext v2: Co-designing and scaling convnets with masked autoencoders. arXiv preprint arXiv:2301.00808 (2023)
65. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: Proc. ECCV (2018)
66. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: SegFormer: Simple and efficient design for semantic segmentation with transformers. In: Proc. NeurIPS (2021)
67. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proc. CVPR (2017)
68. Xu, W., Xu, Y., Chang, T., Tu, Z.: Co-scale conv-attentional image transformers. In: Proc. ICCV (2021)
69. Yamaguchi, K., Sakamoto, K., Akabane, T., Fujimoto, Y.: A neural network for speaker-independent isolated word recognition. In: ICSLP (1990)
70. Yang, X., Zeng, Z., Teo, S.G., Wang, L., Chandrasekhar, V., Hoi, S.: Deep learning for practical image recognition: Case study on Kaggle competitions. In: ACM SIGKDD (2018)

71. Yuan, L., Hou, Q., Jiang, Z., Feng, J., Yan, S.: Volo: Vision outlooker for visual recognition. PAMI (2022)
72. Zeiler, M., Fergus, R.: Stochastic pooling for regularization of deep convolutional neural networks. In: Proc. ICLR (2013)
73. Zhang, R.: Making convolutional networks shift-invariant again. In: Proc. ICML (2019)
74. Zhang, S., Liu, M., Yan, J.: The diversified ensemble neural network. In: Proc. NeurIPS (2020)
75. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proc. CVPR (2018)
76. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ADE20K dataset. In: Proc. CVPR (2017)
77. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ADE20K dataset. IJCV (2019)