ELASTIC: Improving CNNs with Dynamic Scaling Policies

Huiyu Wang[1]* Aniruddha Kembhavi[2] Ali Farhadi[2,3,4] Alan Yuille[1] Mohammad Rastegari[2,4]
[1]Johns Hopkins University [2]PRIOR @ Allen Institute for AI
[3]University of Washington [4]Xnor.ai
huiyu@jhu.edu {anik,mohammadr}@allenai.org ali@cs.uw.edu alan.l.yuille@gmail.com

Abstract

Scale variation has been a challenge from traditional to modern approaches in computer vision. Most solutions to scale issues have a similar theme: a set of intuitive and manually designed policies that are generic and fixed (e.g. SIFT or feature pyramid). We argue that the scaling policy should be learned from data. In this paper, we introduce ELASTIC*, a simple, efficient and yet very effective approach to learn a dynamic scale policy from data. We formulate the scaling policy as a non-linear function inside the network's structure that (a) is learned from data, (b) is instance specific, (c) does not add extra computation, and (d) can be applied on any network architecture. We applied* ELASTIC *to several state-of-the-art network architectures and showed consistent improvement without extra (sometimes even lower) computation on ImageNet classification, MSCOCO multi-label classification, and PASCAL VOC semantic segmentation. Our results show major improvement for images with scale challenges. Our code is available here: https://github.com/allenai/elastic*

1. Introduction

Scale variation has been one of the main challenges in computer vision. There is a rich literature on different approaches to encoding scale variations in computer vision algorithms [20]. In feature engineering, there have been manually prescribed solutions that offer scale robustness. For example, the idea of searching for scale first and then extracting features based on a known scale used in SIFT or the idea of using feature pyramids are examples of these prescribed solutions. Some of these ideas have also been migrated to feature learning using deep learning in modern recognition solutions.

The majority of the solutions in old-school and even modern approaches to encode scale are manually designed and fixed solutions. For example, most state-of-the-art image classification networks [16, 31, 10, 14, 38, 42] use the feature pyramid policy where a network looks at the larger resolution first and then goes to smaller ones as it proceeds through the layers. Despite the fact that this common practice seems to be a natural and intuitive choice, we argue that this scale policy is not necessarily the best one for all possible scale variations in images. We claim that an ideal scale policy should (1) be learned from the data; (2) be instance specific; (3) not add extra computational burden; and (4) be

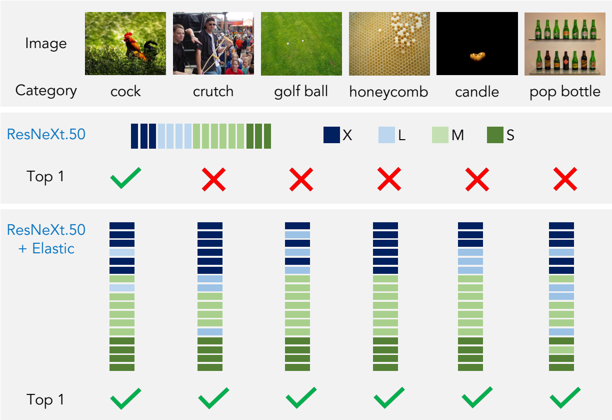

Figure 1: **Dynamic scale policy.** Scaling policies in CNNs are typically integrated into the network architecture manually in a pyramidal fashion. The color bar in this figure (second row) shows the scales at different blocks of the ResNext50 architecture. The early layers receive eXtra-large resolutions and in the following layers resolutions decrease as Large, Medium, and Small. We argue that scaling policies in CNNs should be instance-specific. Our Elastic model (the third row) allows different scaling policies for different input images and it learns from the training data how to pick the best policy. For scale challenging images e.g. images with lots of small(or diverse scale) objects, it is crucial that network can adapt its scale policy based on the input. As it can be seen in this figure, Elastic gives a better prediction for these scale challenging images. (See section 4.1.1 for more details)

[*] Work done while an intern at AI2.

applicable to any network architecture.

For example, instead of looking at the scales according to the feature pyramid policy if we process the images in Figure 1 based on a learned and instance specific policy we see an improved performance. In images with scale challenges like the golf ball image in Figure 1 the learned scale policy might differ dramatically from a pyramid policy, resulting in correct classification of that instance. The learned policy for this instance starts from looking at the image from a large scale (dark blue color), and then goes immediately to a smaller scale, and then goes back to a large scale followed by a small scale and so on.

In this paper, we introduce ELASTIC, an approach to learn instance-specific and not-necessarily-pyramidal scale policies with no extra(or lower) computational cost. Our solution is simple, efficient, and very effective on a wide range of network architectures for image classification and segmentation. Our Elastic model can be applied on any CNN architectures simply by adding downsamplings and upsamplings in parallel branches at each layer and let the network learn from data a scaling policy in which inputs being processed at different resolutions in each layer. We named our model ELASTIC because each layer in the network is flexible in terms of choosing the best scale by a soft policy.

Our experimental evaluations show improvements in image classification on ImageNet[29], multi-label classification on MSCOCO[19], and semantic segmentation on PASCAL VOC for ResNeXt[35], SE-ResNeXt[12], DenseNet[14], and Deep Layer Aggregation (DLA)[38] architectures. Furthermore, our results show major improvements (about 4%) on images with scale challenges (lots of small objects or large variation across scales within the same image) and lower improvements for images without scale challenges. Our qualitative analysis shows that images with similar scaling policies (over the layers of the network) are sharing similar complexity pattern in terms of scales of the objects appearing in the image.

2. Related Work

The idea behind Elastic is conceptually simple and there are several approaches in the literature using similar concepts. Therefore, we study all the categories of related CNN models and clarify the differences and similarities to our model. There are several approaches to fusing information at different visual resolutions. The majority of them are classified into four categories (depicted in Figure 2(b-e)).

Image pyramid: An input image is passed through a model multiple times at different resolutions and predictions are made independently at all levels. The final output is computed as an ensemble of outputs from all resolutions. This approach has been a common practice in [5, 6, 30].

Loss pyramid: This method enforces multiple loss functions at different resolutions. [33] uses this approach to improve the utilization of computing resources inside the network. SSD [21] and MS-CNN [2] also use losses at multiple layers of the feature hierarchy.

Filter pyramid: Each layer is divided into multiple branches with different filter sizes (typically referred to as the split-transform-merge architecture). The variation in filter sizes results in capturing different scales but with additional parameters and operations. The inception family of networks [33, 34, 32] use this approach. To further reduce the complexity of the filter pyramid [25, 36, 37] use dilated convolutions to cover a larger receptive field with the same number of FLOPs. In addition, [4] used 2 CNNs to deal with high and low frequencies, and [40] proposed to adaptively choose from 2 CNNs with different capacity.

Feature pyramid: This is the most common approach to incorporate multiple scales in a CNN architecture. Features from different resolutions are fused in a network by either concatenation or summation. Fully convolutional networks [23] add up the scores from multiple scales to compute the final class score. Hypercolumns [8] use earlier layers in the network to capture low-level information and describe a pixel in a vector. Several other approaches (HyperNet [15], ParseNet [22], and ION [1]) concatenate the outputs from multiple layers to compute the final output. Several recent methods including SharpMask [27] and U-Net [28] for segmentation, Stacked Hourglass networks [26] for keypoint estimation and Recombinator networks [11] for face detection, have used skip connections to incorporate low-level feature maps on multiple resolutions and semantic levels. [13] extends DenseNet[14] to fuse features across different resolution blocks. Feature pyramid networks (FPNs) [18] are designed to normalize resolution and equalize semantics across the levels of a pyramidal feature resolution hierarchy through top-down and lateral connections. Likewise, DLA [38] proposes an iterative and hierarchical deep aggregation that fuses features from different resolutions.

Elastic resembles models from the Filter pyramid family as well as the Feature pyramid family, in that it introduces parallel branches of computation (a la Filter pyramid) and also fuses information from different scales (a la Feature pyramid). The major difference to the feature pyramid models is that in Elastic every layer in the network considers information at multiple scales uniquely whereas in feature pyramid the information for higher or lower resolution is injected from the other layers. Elastic provides an exponential number of scaling paths across the layers and yet keeps the computational complexity the same (or even lower) as the base model. The major difference to the filter pyramid is that the number of FLOPs to cover a higher receptive field in Elastic is proportionally lower, due to the downsampling whereas in the filter pyramid the FLOPs is higher or the same as the original convolution.

3. Model

In this section, we elaborate the structure of our proposed Elastic and illustrate standard CNN architectures being augmented with our Elastic. We also contrast our model with other multi-scale approaches.

3.1. Scale policy in CNN blocks

Formally, a layer in a CNN can be expressed as

$$\mathcal{F}(x) = \sigma\Big(\sum_{i=1}^{q} \mathcal{T}_i(x)\Big) \tag{1}$$

where q is the number of branches to be aggregated, $\mathcal{T}_i(x)$ can be an arbitrary function (normally it is a combination of convolution, batch normalization and activation), and σ are nonlinearities. A few $\mathcal{F}(x)$ are stacked into a stage to process information in one spatial resolution. Stages with decreasing spatial resolutions are stacked to integrate a pyramid scale policy in the network architecture. A network example of 3 stages with 2 layers in each stage is

$$\mathcal{N} = \mathcal{F}_{32} \circ \mathcal{F}_{31} \circ \mathcal{D}_{r_2} \circ \mathcal{F}_{22} \circ \mathcal{F}_{21} \circ \mathcal{D}_{r_1} \circ \mathcal{F}_{12} \circ \mathcal{F}_{11} \tag{2}$$

where $\mathcal{D}_{r_i}$ indicates the resolution decrease by ratio $r_i > 1$ after a few layers. $\mathcal{D}_{r_i}$ can be simply implemented by increasing the stride in the convolution right after. For example, ResNeXt[35] stacks bottleneck layers in each resolution and use convolution with stride 2 to reduce spatial resolution. This leads to a fixed scaling policy that enforces a linear relationship between number of layers and the effective receptive field of those layers. Parameters of $\mathcal{T}_i(x)$ and the elements in input tenors x are all of the tangible ingredients in a CNN that define computational capacity of the model. Under a fixed computational capacity measured by FLOPs, to improve the accuracy of such a model, we can either increase number of parameters in $\mathcal{T}_i(x)$ and decrease the resolution of x or increase the resolution of x and decrease number of parameters in $\mathcal{T}_i(x)$. By adjusting the input resolutions at each layer and number of parameters, we can define a scaling policy across the network. We argue that finding the optimal scaling policy (a trade-off between the resolution and number of parameters in each layer) is not trivial. There are several model designs toward increasing the accuracy and manually injecting variations of feature pyramid but most of them are at the cost of higher FLOPs and more parameters in the network. In the next section, we explain our solution that can learn an optimal scaling policy and maintain or reduce number of parameters and FLOPs while improving the accuracy.

3.2. The ELASTIC structure

In order to learn image features at different scales, we propose to add down-samplings and up-samplings in parallel branches at each layer and let the network make decision on adjusting its process toward various resolutions at each layer. Networks can learn this policy from training data. We add down-samplings and up-samplings in parallel branches at each layer and divide all the parameters across these branches as follows:

$$\mathcal{F}(x) = \sigma\Big(\sum_{i=1}^{q} \mathcal{U}_{r_i}(\mathcal{T}_i(\mathcal{D}_{r_i}(x)))\Big) \tag{3}$$

$$\mathcal{N} = \mathcal{F}_{32} \circ \mathcal{F}_{31} \circ \mathcal{F}_{22} \circ \mathcal{F}_{21} \circ \mathcal{F}_{12} \circ \mathcal{F}_{11} \tag{4}$$

where $\mathcal{D}_{r_i}(x)$ and $\mathcal{U}_{r_i}(x)$ are respectively downsampling and upsampling functions which change spatial resolutions of features in a layer. Unlike in equation 2, a few $\mathcal{F}$ are applied sequentially without downsampling the main stream, and $\mathcal{N}(x)$ has exactly the same resolution as original x.

Note that the learned scaling policy in this formulation will be instance-specific i.e. for different image instances, the network may activate branches in different resolutions at each layer. In section 4 we show that this instance-specific scaling policy improves prediction on images with scale challenges e.g. images consist of lots of small objects or highly diverse object sizes.

Conceptually, we propose a new structure where information is always kept at a high spatial resolution, and each layer or branch processes information at a lower or equal resolution. In this way we decouple feature processing resolution ($\mathcal{T}_i$ processes information at different resolutions) from feature storage resolution (the main stream resolution of the network). This encourages the model to process different scales separately at different branches in a layer and thus capture cross-scale information. More interestingly, since we apply Elastic to almost all blocks, the dynamic combination of multiple scaling options at each layer leads to exponentially many different scaling paths. They interpolate between the largest and the smallest possible scale and collectively capture various scales. In fact, this intuition is aligned with our experiments, where we have observed different categories of images adopt different scaling paths (see section 4.1.1). For example, categories with clean and uniform background images mostly choose the low-resolution paths across the network and categories with complex and cluttered objects and background mostly choose the high-resolution paths across the network.

The computational cost of our Elastic model is equal to or lower than the base model, because at each layer the maximum resolution is the original resolution of the input tensor. Low resolution branches reduce the computation and give us extra room for adding more layers to match the computation of the original model.

This simple add-on of downsamplings and upsamplings (Elastic) can be applied to any CNN layers $\mathcal{T}_i(x)$ in any architecture to improve accuracy of a model. Our applications are introduced in the next section.

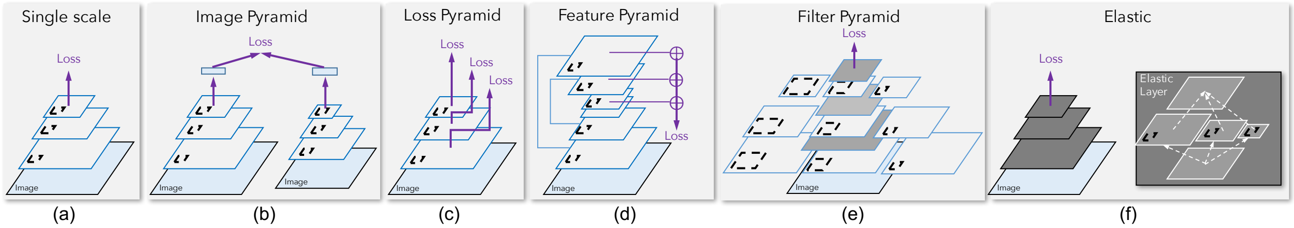

Figure 2: **Multi-scaling model structures.** This figure illustrates different approaches to multi-scaling in CNN models and our Elastic model. The solid-line rectangles show the input size and the dashed-line rectangles shows the filter size.

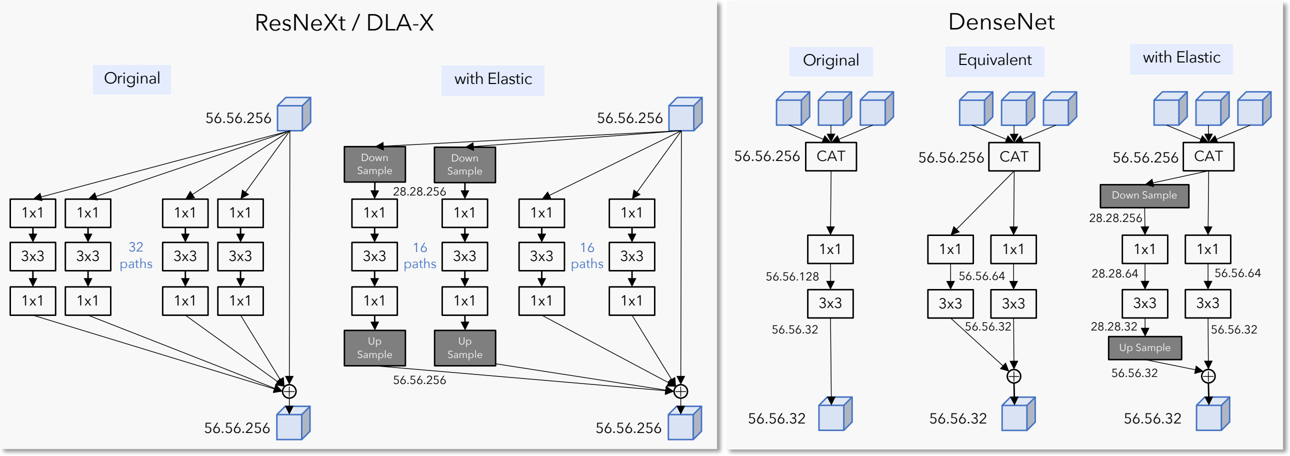

Figure 3: **Left:** ResNeXt bottleneck vs. Elastic bottleneck. **Right:** DenseNet block vs. its equivalent form vs. Elastic block. Elastic blocks spend half of the paths processing downsampled inputs in a low resolution, then the processed features are upsampled and added back to features with the original resolution. Elastic blocks have the same number of parameters and less FLOPs than original blocks

3.3. Augmenting models with Elastic

Now, we show how to apply Elastic on different network architecture. To showcase the power of Elastic, we apply Elastic on some state-of-the-art network architectures: ResNeXt[35], Deep Layer Aggregation (DLA)[38], and DenseNet[14]. A natural way of applying Elastic on current classification models is to augment bottleneck layers with multiple branches. This makes our modification on ResNeXt and DLA almost identical. At each layer we apply downsampling and bilinear upsampling to a portion of branches, as shown in Figure 3-left. In DenseNet we compile an equivalent version by parallelizing a single branch into two branches and then apply downsampling and upsampling on some of the branches, as shown in Figure 3-right. Note that applying Elastic reduces FLOPs in each layer. To match the original FLOPs we increase number of layers in the network while dividing similar number of FLOPs across resolutions.

Relation to other multi-scaling approaches As discussed in section 2, most of current multi-scaling approaches can be categorized into four different categories (1) *image pyramid*, (2) *loss pyramid* (3) *filter pyramid*, and (4) *feature pyramid*. Figure 2(b-e) demonstrates the structure of these categories. All of these models can improve the accuracy usually under a higher computational budget. Elastic (Figure 2) guarantees no extra computational cost while achieving better accuracy. Filter pyramid is the most similar model to Elastic. The major difference to the filter pyramid is that the number of FLOPs to cover a higher receptive field in Elastic is proportionally lower due to the downsampling whereas in the filter pyramid the FLOPs is higher or the same as the original convolution depending of filter size or dilation parameters. Table 1 compares the FLOPs and number of parameters between Elastic and feature/filter pyramid for a single convolutional operation. Note that the FLOPs and parameters in Elastic is always (under any branching q and scaling ratio r) lower or equal to the original model whereas in filter/feature pyramid this is higher or equal. Feature pyramid methods are usually applied on top of an existing classification model, by concatenating features from different resolutions. It is capable of merging features from different scales in the backbone model and shows improvements on various tasks, but it does not intrinsically change the scaling policy. Our Elastic structure can be viewed as a feature pyramid inside a layer,

Multi-Scaling Method	FLOPs	Parameters
Single Scale	n^2ck^2	ck^2
Feature Pyramid (concat)	$n^2(qc)k^2$	$(qc)k^2$
Feature Pyramid (add)	n^2ck^2	ck^2
Filter Pyramid (standard)	$\sum_{i=1}^{q} \frac{n^2c(kr_i)^2}{b_i}$	$\sum_{i=1}^{q} \frac{c(kr_i)^2}{b_i}$
Filter Pyramid (dilated)	n^2ck^2	ck^2
Elastic	$\sum_{i=1}^{q} \frac{(\frac{n}{r_i})^2ck^2}{b_i}$	ck^2

Table 1: **Computation in multi-scaling models.** This table compares the FLOPs and number of parameters between Elastic and feature/filter pyramid for a single convolutional operation, where the input tensor is $n \times n \times c$ and the filter size is $k \times k$. q denotes the number of branches in the layer, where $\sum_1^q \frac{1}{b_i} = 1$ and $b_i > 1$ and $r_i > 1$ denote the branching and scaling ratio respectively. Note that the FLOPs and parameters in Elastic is always (under any branching q and scaling ratio r) lower than or equal to the original model whereas in feature/filter pyramid is higher or equal.

which is able to model different scaling policies. Spatial pyramid pooling or Atrous(dilated) spatial pyramid shares the same limitation as feature pyramid methods.

4. Experiments

In this section, we present experiments on applying Elastic to current strong classification models. We evaluate their performances on ImageNet classification, and we show consistent improvements over current models. Furthermore, in order to show the generality of our approach, we transfer our pre-trained Elastic models to multi-label image classification and semantic segmentation. We use ResNeXt [35], DenseNet[14] and DLA [38] as our base models to be augmented with Elastic.

Implementation details. We use the official PyTorch ImageNet codebase with random crop augmentation but without color or lighting augmentation, and we report standard 224×224 single crop error on the validation set. We train our model with 8 workers (GPUs) and 32 samples per worker. Following DLA [38], all models are trained for 120 epochs with learning rate 0.1 and divided by 10 at epoch 30, 60, 90. We initialize our models using normal He initialization [9]. Stride-2 average poolings are adopted as our downsamplings unless otherwise notified since most of our downsamplings are 2× downsamplings, in which case bilinear downsampling is equivalent to average pooling. Also, Elastic add-on is applied to all blocks except stride-2 ones or high-level blocks operating at resolution 7.

4.1. ImageNet classification

We evaluate Elastic on ImageNet[29] 1000 way classification task (ILSVRC2012). The ILSVRC 2012 dataset contains 1.2 million training images and 50 thousand validation images. In this experiment, we show that our Elastic add-on consistently improves the accuracy of the state-of-the-art models without introducing extra computation or parameters. Table 2 compares the top-1 and top-5 error rates of all of the base models with the Elastic augmentation (indicated by '**+Elastic**') and shows the number of parameters and FLOPs used for a single inference. Besides DenseNet, ResNeXt, DLA, SE-ResNeXt50+Elastic is also reported. In all the tables "*" denotes our implementation of the model. It shows that our improvement is almost orthogonal to the channel calibration proposed in [12]. In addition, we include ResNeXt50x2+Elastic to show that our improvement does not come from more depth added to ResNeXt101. In Figure 4 we project the numbers in the Table 2 into two plots: accuracy vs. number of parameters (Figure 4-left) and accuracy vs. FLOPs (Figure 4-right). This plot shows that our Elastic model can reach to a higher accuracy without any extra (or with lower) computational cost.

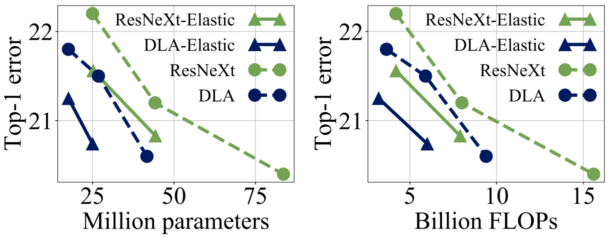

Figure 4: **Imagenet Accuracy vs. FLOPS and Parameters** This figure shows our Elastic model can achieve a lower error without any extra (or with lower) computational cost.

4.1.1 Scale policy analysis

To analyze the learned scale policy of our Elastic model, we define a simple score that shows at each block what was the resolution level (high or low) that the input tensor was processed. We formally define this scale policy score at each block by differences of mean activations in high-resolution and low-resolution branches.

$$S = \frac{1}{4HWC} \sum_{h=1}^{2H} \sum_{w=1}^{2W} \sum_{c=1}^{C} x_{hwc}^{high} - \frac{1}{HWC} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} x_{hwc}^{low} \quad (5)$$

where H, W, C are the height, width and number of channels in low resolution branches. x^{high} and x^{low} are the activations after 3×3 convolutions, fixed batch normalizations, and ReLU in high-resolution and low-resolution branches respectively. Figure 5 shows all of the categories in ImageNet validation sorted by the mean scale policy score S (average over all layers for all images within a category). As it can be seen, categories with more complex images appear to have a larger S i.e. they mostly go through high-resolution branches in each block and images with simpler

Model	# Params	FLOPs	Top-1	Top-5
DenseNet201*	20.0M	4.4B	22.25	6.26
DenseNet201+Elastic	19.5M	4.3B	**22.07**	**6.00**
ResNeXt50	25.0M	4.2B	22.2	-
ResNeXt50*	25.0M	4.2B	22.23	6.25
ResNeXt50+Elastic	25.2M	4.2B	**21.56**	**5.83**
SE-ResNeXt50*	27.6M	4.2B	21.87	5.93
SE-ResNeXt50+Elastic	27.8M	4.2B	**21.38**	**5.86**
ResNeXt101	44.2M	8.0B	21.2	5.6
ResNeXt101*	44.2M	8.0B	21.18	5.83
ResNeXt101+Elastic	44.3M	7.9B	**20.83**	**5.41**
ResNeXt50x2+Elastic	45.6M	7.9B	20.86	5.52
DLA-X60	17.6M	3.6B	21.8	-
DLA-X60*	17.6M	3.6B	21.92	6.03
DLA-X60+Elastic	17.6M	3.2B	**21.25**	**5.71**
DLA-X102	26.8M	6.0B	21.5	-
DLA-X102+Elastic	25.0M	6.0B	**20.71**	**5.38**

Table 2: **State-of-the-art model comparisons on ImageNet validation set.** Base models (DenseNet, ResNeXt, and DLA) are augmented by Elastic (indicated by '+Elastic'). * indicates our implementation of these models. Note that augmenting with Elastic always improves accuracy across the board.

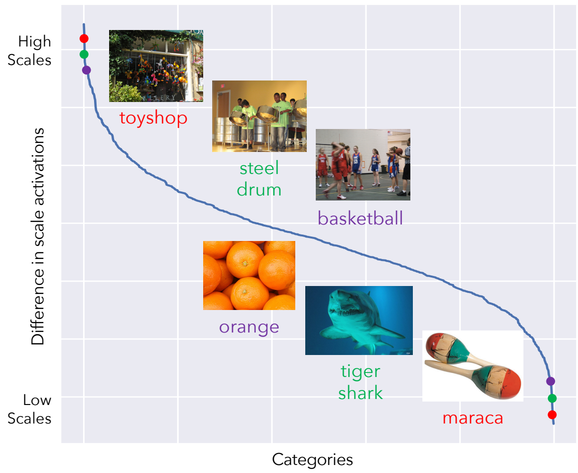

Figure 5: **Scale policy for complex vs. simple image categories**. This figure shows the overall block scale policy score on the entire ImageNet categories. It shows that categories with complex image patterns mostly go through the high-resolution branches in the network and categories with simpler image pattern go through the low-resolution branches.

patterns appear to have smaller S which means they mostly go through the low-resolution branches in each block.

To analyze the impact of the scale policy on the accuracy of the Elastic, we represent each image (in the ImageNet validation set) by a 17-dimensional vector such that the values of the 17 elements are the scale policy score S for the 17 Elastic blocks in a ResNeXt50+Elastic model. Then we apply tsne[24] on all these vectors to get a two-dimensional visualization. In figure 6-(left) we draw all the images in the tsne coordinates. It can be seen that images are clustered based on their complexity pattern. In figure 6-(middle) for all of the images we show the 17 scale policy scores S in 17 blocks. As it can be seen most of the images go through the high-resolution branches on the early layers and low-resolution branches at the later layers but some images break this pattern. For examples, images pointed by the green circle are activating high-resolution branches in the 13^{th} block of the network. These images usually contain a complex pattern that the network needs to extract features in high-resolution to classify correctly. Images pointed by the purple circle are activating low-resolution branches at early layers, the 4^{th} block of the network. These images usually contain a simple pattern that the network can classify at low-resolution early on. In Figure 6-(right) we show the density of all validation images in the tsne space in the bottom row, and in the top row, we show the density of images that are correctly classified by our Elastic model and miss-classified by the base ResNeXt model. This comparison shows that most of the images that Elastic can improve predictions on are the ones with more challenging scale properties. Some of them are pointed out by the yellow circle.

4.2. MS COCO multi-label classification

To further investigate the generality of our model, we finetune our ImageNet pre-trained model and evaluate on MS COCO multi-label classification task. The MSCOCO images are far more complicated in that there exist multiple objects from different categories and scales in each image.

Implementation details. All models that we report are finetuned from ImageNet pre-trained model for 36 epochs with learning rate starting at 0.001 and being divided by 10 at epoch 24, 30. We train on 4 workers and 24 images per worker with SGD and weight decay of 0.0005. We train our models with binary cross entropy (BCE) loss, which is usually used as a baseline for domain-specific works that explicitly model spatial or semantic relations. We use the same data augmentations as our ImageNet training, and adopt standard multi-label testing on images resized to 224×224.

Evaluation metrics. Following the literature of multi-label classification[41, 7, 39, 17], results are evaluated using macro/micro evaluations. After training the models with BCE loss, labels with greater than 0.5 probability are considered positive. Then, macro and micro F1-scores are calculated to measure overall performance and the average of per-class performances respectively.

Results. Table 3 shows that elastic consistently improves per-class F1 and overall F1. In the case of DLA, Elastic augmentation even reduces the FLOPs and number of parameters by a large margin.

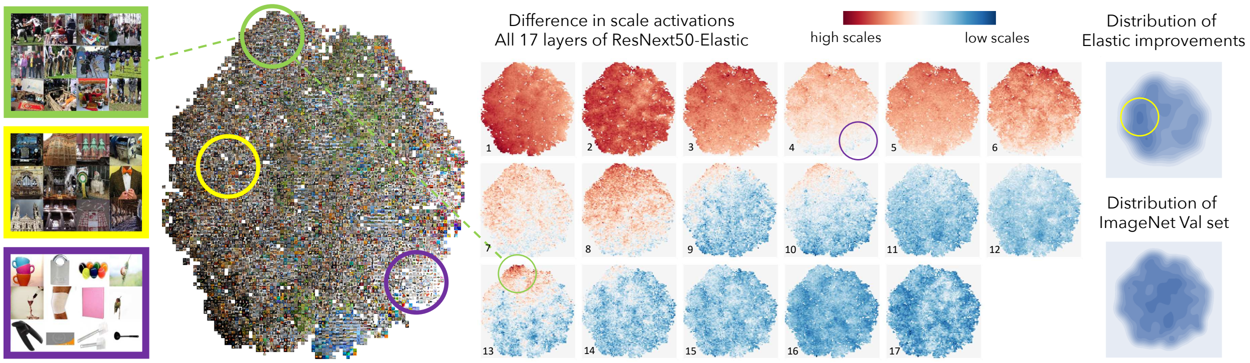

Figure 6: **Scale policy analysis.** This figure shows the impact of the scale policy on the accuracy of our Elastic model. (left) shows all the ImageNet validation set clustered using tsne by their scale policy pattern in the ResNeXt50+Elastic as discussed in section 4.1.1. (middle) shows the the scale policy score of all the images at 17 blocks of the network. Most of the images use high-resolution features at early layers and low-resolution features at later layers but some images break this pattern. Images pointed in the green circle use high-resolution features in the 13th block. Images pointed in the purple circle use low-resolution features in the 4th block. These images usually contain a simpler pattern. (right)-bottom shows the density of images in the tsne space and (right)-top shows the density of the images that got correctly classified by Elastic model but miss-classified by the base ResNeXt model. This shows that Elastic can improve prediction when images are challenging in terms of their scale information. Some samples are pointed by the yellow circle. Best viewed in color.

Model	F1-PerClass	F1-Overall
ResNet101*	69.98	74.58
DenseNet201*	69.95	74.50
DenseNet201+Elastic	**70.40**	**74.99**
DLA-X60*	70.79	75.41
DLA-X60+Elastic	**71.35**	**75.77**
ResNeXt50*	70.12	74.52
ResNeXt50+Elastic	**71.08**	**75.37**
ResNeXt101*	70.95	75.21
ResNeXt101+Elastic	**71.83**	**75.93**

Table 3: **MSCOCO multi-class classification.** This table shows the generality of our Elastic model by finetuning pre-trained ImageNet models on MSCOCO multi-class images with binary cross entropy loss. Elastic improves F1 scores all across the board.

Scale challenging images. We claimed that Elastic is very effective on scale challenging images. Now, we empirically show that a large portion of the accuracy improvement of our Elastic model is rooted in a better scale policy learning. We follow MSCOCO official split of *small*, *medium*, and *large* objects. Per-class and overall F1, on small, medium and large objects, are computed. Since we don't have per-scale predictions, false positives are shared and re-defined as cases where none of small, medium, large object appears, but the model predicts positive. Results in Table 4 show that ResNeXt50 + Elastic provides the largest gains on small objects. Elastic allows large objects to be dynamically captured by low resolution paths, so filters in high resolution branches do not waste capacity dealing with parts of large objects. Elastic blocks also merge various scales and feed scale-invariant features into the next block, so it shares computation in all higher blocks, and thus allows more capacity for small objects, at high resolution. This proves our hypothesis that Elastic understands scale challenging images better through scale policy learning.

Scale stress test. Besides standard testing where images are resized to 224 × 224, we also perform a stress test on the validation set. MSCOCO images' resolutions are ~640 × 480. Given a DLA-X60 model trained with 224 × 224 images, we also test it with images from different resolutions: 96 × 96, 448 × 448, 896 × 896 and change the last average pooling layer accordingly. Figure 7 shows that Elastic does not only perform well on trained scale, but also shows greater improvement on higher resolution images at test time. In addition, we do not observe an accuracy drop on 96 × 96 test, though the total computation assigned to low level is reduced in DLA-X60+Elastic.

4.3. PASCAL VOC semantic segmentation

To show the strength of our Elastic model on a pixel level classification task, we report experiments on PASCAL VOC semantic segmentation. ResNeXt models use weight decay 5e-4 instead of 1e-4 in ResNet. All models are trained for 50 epochs and we report mean intersection-over-union (IOU) on the val set. Other implementation details follow [3], with MG(1, 2, 4), ASPP(6, 12, 18), image pooling, OS=16, batch

Model	Sm-C	Md-C	Lg-C	Sm-O	Md-O	Lg-O
ResNeXt50	45.57	61.99	65.88	58.51	68.51	77.53
+Elastic	46.67	63.05	66.46	59.47	69.47	78.03
Relative	2.43%	1.72%	0.88%	1.64%	1.40%	0.65%

Table 4: F1 scores on small, medium, and large objects respectively. C means per-class F1 and O means overall F1. ResNeXt50 + Elastic improves the most on small objects.

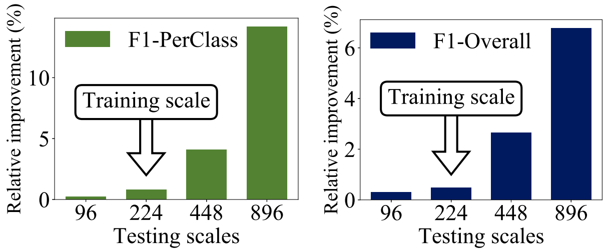

Figure 7: **Scale stress test** on MSCOCO multi-label classification. This bar chart shows the relative F1 improvement of DLA-x60 being augmented Elastic over different image resolutions. Although both models are trained on 224×224 images, Elastic shows larger improvement when tested on high-resolution images.

Model	Original	Elastic
ResNeXt50*	75.29	**77.70**
ResNeXt101*	77.47	**78.51**
DLA-X60*	69.96	**73.59**

Table 5: **PASCAL VOC semantic segmentation**. This table compares the accuracy of semantic image segmentation (mIOU%) using Elastic models vs. the original model. Elastic models outperform original models by a large margin. This supports that Elastic learns a scale policy that allows processing high-level semantic information and low-level boundary information together.

Method	# FLOPs	Top-1 error
Original (no Elastic)	3.6B	21.92
Bilinear w/ AP	**3.2B**	21.25
Nearest w/ AP	3.2B	21.49
Trained Dilated Filter w/ AP	3.6B	**21.20**
Trained Dilated Filter	3.6B	21.60
Trained Filter	3.2B	21.52

Table 6: **Ablation study of up(down)sampling methods.** In this table, we show the accuracy of ImageNet classification using Elastic by different choices of up(down)sampling methods. **w/ AP** indicates average pooling. Our experiment shows Elastic with bilinear up(down)sampling is the best choice with reduced FLOPs.

High-Res	Low-Res	FLOPs	Top-1 error
100%	0%	3.6B	21.92
50%	50%	3.2B	21.25
75%	25%	3.4B	21.35
25%	75%	2.9B	21.44

Table 7: **Ablation study of high(low) resolution branching rates**. In this table, we evaluate different branching rate across high and low-resolutions at each block. We observe that the best trade-off is when we equally divide the branches into high and low-resolutions. Independent of the ratio, all variations of branching are better than the base model.

size of 16, for both training and validation, without bells and whistles. Our ResNet101 reproduces the mIOU of 77.21% reported in [3]. Our DLA models use the original iterative deep aggregation as a decoder and are trained with the same scheduling as [3]. In Table. 5, Elastic shows a large margin of improvement. This verifies that Elastic finds the scale policy that allows processing high-level semantic information and low-level boundary information together, which is critical in the task of semantic segmentation.

4.4. Ablation study

In this section, we study the effect of different elements in Elastic models. We chose DLA-X60 as our baseline and applied Elastic to perform the ablation experiments.

Upsampling/Downsampling methods. We carried our experiments with bilinear up(down)sampling on DLA-X60+Elastic. In Table 6 we show the accuracy of ImageNet classification using Elastic by different choices of up(down)sampling methods: Bilinear, Nearest, Trained filters and Trained Dilated filters with and without average pooling (indicated by **w/ AP**). Our experiment shows Elastic with the bilinear up(down)sampling is the best choice.

High/low-resolution branching rate. We sweep over different choices of dividing parallel branches in the blocks into the high and low-resolutions. In table 7 we compare the variations of the percentage of branches allocated to high and low-resolutions at each block. This experiment shows that the best trade-off is when we equally divide the branches into high and low-resolutions. Interestingly, all of the branching options are outperforming the vanilla model (without Elastic). This shows that our Elastic model is quite robust to this parameter.

5. Conclusion

We proposed Elastic, a model that captures scale variations in images by learning the scale policy from data. Our Elastic model is simple, efficient and very effective. Our model can easily be applied to any CNN architectures and improve accuracy while maintaining the same computation (or lower) as the original model. We applied Elastic to several state-of-the-art network architectures and showed consistent improvement on ImageNet classification, MSCOCO multi-class classification, and PASCAL VOC semantic segmentation. Our results show major improvement for images with scale challenges e.g. images consist of several small objects or objects with large scale variations.

Acknowledgments

We thank Zhishuai Zhang, Siyuan Qiao, and Chenxi Liu for their insightful comments and suggestions. This work is in part supported by ONR N00014-12-1-0883, NSF IIS-165205, NSF IIS-1637479, NSF IIS-1703166, Sloan Fellowship, NVIDIA Artificial Intelligence Lab, and Allen Institute for artificial intelligence.

References

[1] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2874–2883, 2016. 2

[2] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer, 2016. 2

[3] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 7, 8

[4] T. Chen, L. Lin, W. Zuo, X. Luo, and L. Zhang. Learning a wavelet-like auto-encoder to accelerate deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 2

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 2

[7] W. Ge, S. Yang, and Y. Yu. Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1277–1286, 2018. 6

[8] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015. 2

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 5

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[11] S. Honari, J. Yosinski, P. Vincent, and C. Pal. Recombinator networks: Learning coarse-to-fine feature aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5743–5752, 2016. 2

[12] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 7, 2017. 2, 5

[13] G. Huang, S. Liu, L. van der Maaten, and K. Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. *group*, 3(12):11, 2017. 2

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017. 1, 2, 4, 5

[15] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 845–853, 2016. 2

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[17] Y. Li, Y. Song, and J. Luo. Improving pairwise ranking for multi-label image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6

[18] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. 2

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[20] T. Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994. 1

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[22] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 2

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2

[24] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 6

[25] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. *arXiv preprint arXiv:1803.06815*, 2018. 2

[26] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016. 2

[27] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016. 2

[28] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 2, 5

[30] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[32] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017. 2

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 2

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2

[35] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017. 2, 3, 4, 5

[36] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016. 2

[37] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. *CVPR*, 2017. 2

[38] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2403–2412, 2018. 1, 2, 4, 5

[39] J. Zhang, Q. Wu, C. Shen, J. Zhang, and J. Lu. Multi-label image classification with regional latent semantic dependencies. *IEEE Transactions on Multimedia*, 2018. 6

[40] H.-Y. Zhou, B.-B. Gao, and J. Wu. Adaptive feeding: Achieving fast and accurate detections by adaptively combining object detectors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3505–3513, 2017. 2

[41] F. Zhu, H. Li, W. Ouyang, N. Yu, and X. Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. *arXiv preprint arXiv:1702.05891*, 2017. 6

[42] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 1