Continuous Adaptation for Interactive Object Segmentation by Learning from Corrections

Theodora Kontogianni[*+]
RWTH Aachen
kontogianni@vision.rwth-aachen.de

Michael Gygli[*]
Google Research
gyglim@google.com

Jasper Uijlings
Google Research
jrru@google.com

Vittorio Ferrari
Google Research
vittoferrari@google.com

Abstract

In interactive object segmentation a user collaborates with a computer vision model to segment an object. Recent works rely on convolutional neural networks to predict the segmentation, taking the image and the corrections made by the user as input. By training on large datasets they offer strong performance, but they keep model parameters fixed at test time. Instead, we treat user corrections as training examples to update our model on-the-fly to the data at hand. This enables it to successfully adapt to the appearance of a particular test image, to distributions shifts in the whole test set, and even to large domain changes, where the imaging modality changes between training and testing. We extensively evaluate our method on 8 diverse datasets and improve over a fixed model on all of them. Our method shows the most dramatic improvements when training and testing domains differ, where it produces segmentation masks of the desired quality from 60-70% less user input. Furthermore we achieve state-of-the-art on four standard interactive segmentation datasets: PASCAL VOC12, GrabCut, DAVIS16 and Berkeley.

1. Introduction

In interactive object segmentation a human collaborates with a Computer Vision model to segment an object of interest [12, 46, 52, 11]. The process iteratively alternates between the user providing corrections on the current segmentation and the model refining the segmentation based on these corrections. The objective of the model is to infer the right segmentation from as few corrections as possible (typically point clicks [8, 15] or strokes [46, 24] on mislabeled pixels). This enables fast and accurate object segmentation, which is indispensable for both image editing [2] and collecting ground-truth segmentation masks at scale [11].

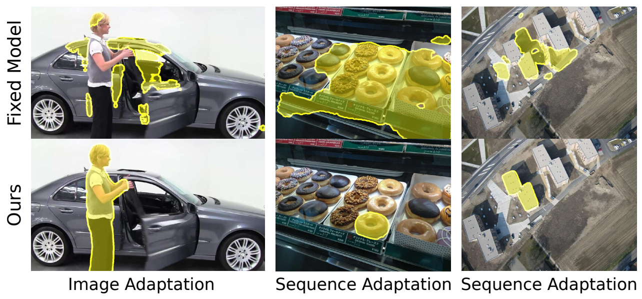


Figure 1: **Example segmentations for a fixed model (top) and our adaptive methods (bottom).** A fixed model performs poorly when foreground and background share similar appearance (left), when it is used to segment new object classes absent in the training set (center, donut class), or when the model is tested on a different image domain than it is trained on (right). By using corrections to adapt the model parameters to a specific test image, or to the test image sequence, our method substantially improves its predictions. The input is four corrections in all cases shown.

State-of-the-art methods train a convolutional neural network (CNNs) which takes the image and the user corrections as input and predicts a foreground/background segmentation [52, 33, 10, 35, 31, 11, 28]. At test time, the model parameters are fixed and corrections are used as additional input to guide the model predictions. But in fact, user corrections directly specify the ground-truth labelling of the corrected pixels. In this paper we capitalize on this observation. We treat user corrections as training examples to adapt our model on-the-fly to the data at hand. We do this in two ways: (1) in *single image adaptation* we iteratively adapt model parameters to a specific image; (2) in *image sequence adaptation* we adapt model parameters to a

[*]Equal contribution.
[+]Work done while interning at Google.

sequence of images. Each of these leads to distinct advantages over using a fixed model.

During *single image adaptation* our model learns the specific appearance of the current object instance and the surrounding background. This allows it to adapt even to subtle differences between foreground and background for that specific example. This is necessary when the image has objects which have similar color as the background (Fig. 1a), has blurry object boundaries, or has low contrast. Importantly, image adaptation enables user corrections to have non-local effects. For example, when marking something to be background, this information will be propagated to distant regions in the image with the same appearance. Finally, a fixed model can sometimes ignore the user corrections and overrule them in its next prediction. We avoid this undesired behavior by updating the model parameters until its predictions respect the user corrections.

During *image sequence adaptation* we continuously adapt the model to a new sequence of images, thus optimizing its parameters for their distribution, which might be different than the training one. We can also adapt to a new object class distribution. An important case is adaptation to a set of new classes unseen during training. Another is specialization to a single class, useful for collecting many examples in high-precision domains, such as segmentations of *pedestrians* for self-driving car applications. Fig. 1b shows an example of specializing to the single, unseen class *donut*. Finally, our idea of image sequence adaptation also enables to handle large domain changes, where the imaging modality changes between training and testing. We demonstrate this by training on consumer photos while testing on medical and aerial images (Fig. 1c). Naturally, single image adaptation and image sequence adaptation can be used jointly, leading to a method that combines their advantages.

In summary: we treat user corrections as training examples to adapt our segmentation model on-the-fly to the data at hand. We evaluate our method on 8 diverse datasets. On all it shows reductions in corrections necessary to achieve a target quality, compared to a fixed model. With single image adaptation we reduce the number of clicks by 20% on unseen classes of COCO [34]. With image sequence adaptation we bridge large domain changes: when trained on consumer photos (PASCAL VOC [20]) and tested on aerial imagery (Rooftop [49]) we reduce the number of clicks by 60%. With the combination we achieve state-of-the-art on four standard interactive segmentation datasets: PASCAL VOC12 [20], GrabCut [46], DAVIS [40] and Berkeley [37].

2. Related Work

Interactive Object Segmentation. Traditional methods model interactive segmentation as an energy minimization problem on a graph defined over pixels [12, 46, 7, 24, 41]. User inputs are used to create an image-specific appearance model based on low-level features (*e.g.* color), which is then used to predict foreground and background probabilities. A pairwise smoothness term between neighboring pixels encourages regular segmentation outputs. Hence these classical methods are based on a weak appearance model which is specialized to one specific image.

Recent methods rely on Convolutional Neural Networks (CNNs) to interactively produce a segmentation mask [52, 33, 10, 35, 31, 15, 27, 28, 3]. These methods take the image and user corrections (transformed into a guidance map) as input and map them to foreground and background probabilities. This mapping is optimized over a training dataset and remains fixed at test time. Hence these models have a strong appearance model but it is not optimized for the image or dataset at hand.

Our method combines the advantages of traditional and recent approaches: We use a modern CNN to learn a strong initial appearance model from a training set. During segmentation of a new test image, we adapt the model to it. It thus learns an appearance model of foreground and background specifically for that image. Furthermore, we also continuously adapt the model to the new image and class distribution of the test set, which may be far from what the model is originally trained on.

Gradient Descent during Inference. Several methods iteratively minimize a loss during inference time. The concurrent work of [50] uses self-supervision to adapt the feature extractor of a multi-tasking model to the test distribution. Instead, we directly adapt the full model by minimizing the task loss. Others iteratively update the inputs of a model [23, 25, 28], *e.g.* for style transfer [23]. In the domain of interactive segmentation, [28] updates the guidance map which encodes the user corrections and is input to the model. Instead our method updates the model parameters, making it more general and allowing it to adapt to individual images as well as sequences.

In-domain Fine-Tuning. In other applications it is common practice to fine-tune on in-domain data when transferring a model to a new domain [13, 39, 51, 57]. For example, when supervision for the first frame of a test video is available [40, 51, 13], or after annotating a subset of an image dataset [39, 57]. In interactive segmentation the only existing attempt is [1], which performs polygon annotation. However, (1) it does not consider adapting to a particular image; (2) their process to fine-tune on a dataset involves 3 different models, so they do it only a few times per dataset; (3) they cannot directly train on user corrections, only on complete masks from previous images; (4) finally, they require a bounding box on the object as input.

Few-shot and Continual Learning. Our method automatically adapts to distribution shifts and domain changes. It performs domain adaptation from limited supervision, similar to few-shot learning [43, 22, 48, 42]. It also relates to continual learning [44, 21], except that the output label space of the classifier is fixed. As in other work, our method needs to balance preserving existing knowledge and adapting to the new data. This is often done by fine-tuning on new tasks while discouraging large changes in the network parameters, either by penalizing changes to important parameters [30, 54, 5, 6] or penalizing changing predictions of the model on old tasks [32, 47, 38]. Alternatively, some training data of the old task is kept and the model is trained on a mixture of the old and new task data [44, 9].

3. Method

We adopt a typical interactive object segmentation process [12, 52, 35, 31, 11, 28]: the model is given an image and makes an initial foreground/background prediction for every pixel. The prediction is then overlaid on the image and presented to the user, who is asked to make a correction. The user clicks on a single pixel to mark that it was incorrectly predicted to be foreground instead of background or vice versa. The model then updates the predicted segmentation based on all corrections received so far. This process iterates until the segmentation reaches a desired quality level.

We start by describing the model we build on (Sec. 3.1). Then, we describe our core contribution: treating user corrections as training examples at test-time to adapt our model on-the-fly (Sec. 3.2). Lastly, we describe how we simulate user corrections to train and test our method (Sec. 3.3).

3.1. Initial Segmentation Model

We use a standard interactive segmentation model [52] as the basis of our method. The model takes an RGB image and the user corrections as input and produces a segmentation mask. As in [11] we encode the position of user corrections by placing binary disks into a two-channel map with the same resolution as the image (one channel for foreground and one for background corrections). The user corrections are concatenated with the RGB image to form a 5-channel map $\mathbf{x}$ which is provided as input to the network.

As network architecture we use DeepLabV3+ [17], which has demonstrated excellent performance on semantic segmentation. It modifies a strong image classification network [18] for dense pixel-level prediction by incorporating multi-scale context via Atrous Spatial Pyramid Pooling [16] and a decoder to increase the output spatial resolution. However, we note that our method does not depend on a specific architecture and can be used with others as well.

For training the model we need a training dataset $\mathcal{D}$ with ground-truth object segmentations, as well as user corrections which we simulate (Sec. 3.3). We train the model using the cross-entropy loss over all pixels in an image:

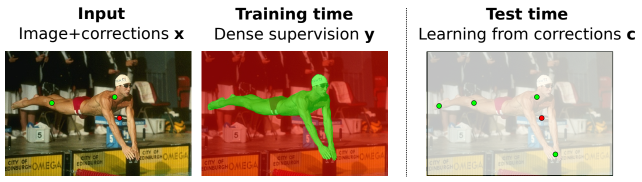


Figure 2: **Corrections as training examples.** For training the initial model, full supervision is available, allowing to compute a loss over all the pixels in the image. At test time, the user provides sparse supervision in the form of corrections. We use these to adapt the model parameters.

$$\mathcal{L}_{\text{CE}}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) = \frac{1}{|\mathbf{y}|}\{-\mathbf{y}\log \mathbf{f}(\mathbf{x};\boldsymbol{\theta}) - (1-\mathbf{y})\log(1-\mathbf{f}(\mathbf{x};\boldsymbol{\theta}))\} \tag{1}$$

where $\mathbf{x}$ is the 5-channel input defined above (image plus corrections), $\mathbf{y} \in \{0, 1\}$ are the pixel labels for the ground-truth object segmentations, and $\mathbf{f}(\mathbf{x};\boldsymbol{\theta})$ represents the mapping of the convolutional network parameterized by $\boldsymbol{\theta}$. $|\cdot|$ denotes the l_1 norm.

We produce the initial parameters $\boldsymbol{\theta}^*$ of the segmentation model by minimizing $\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}} \mathcal{L}_{\text{CE}}(\mathbf{x}_i, \mathbf{y}_i; \boldsymbol{\theta})$ over the training set using stochastic gradient descent.

3.2. Learning from Corrections at Test-Time

In previous interactive object segmentation works all model parameters are fixed at test time [52, 10, 35, 31, 11, 28]. Corrections are only used as inputs to guide the predictions. Instead, we treat corrections as ground-truth labels to adapt the model at test time. We achieve this by minimizing the cross-entropy loss over the corrected pixels:

$$\mathcal{L}_{\text{GCE}}(\mathbf{x}, \mathbf{c}; \boldsymbol{\theta}) = \frac{\mathbf{1}[\mathbf{c}{\neq}{-1}]^T}{|\mathbf{1}[\mathbf{c}{\neq}{-1}]|}\Big\{-\mathbf{c}\log \mathbf{f}(\mathbf{x};\boldsymbol{\theta}) - (1-\mathbf{c})\log(1-\mathbf{f}(\mathbf{x};\boldsymbol{\theta}))\Big\}, \tag{2}$$

where $\mathbf{1}$ in an indicator function and $\mathbf{c}$ is a vector of values $\{1, 0, -1\}$, indicating what pixels were corrected to what label. Pixels that were corrected to be positive are set to 1 and negative pixels to 0. The remaining ones are set to -1, so that they are ignored in the loss. As there are very few corrections available at test time, this loss is computed over a sparse set of pixels. This is in contrast to the initial training which had supervision at every pixel (Sec. 3.1). Hence, special care needs to be taken to make this form of supervision useful in practice, as we discuss next. We illustrate the contrast between the two forms of supervision in Fig. 2.

Regularization. We propose two forms of regularization. First, we regularize the model by treating the initial mask prediction $\mathbf{p}$ as ground-truth and making it a target in the cross-entropy loss, *i.e.* $\mathcal{L}_{\mathrm{CE}}(\mathbf{x}, \mathbf{p}; \boldsymbol{\theta})$. Having this loss prevents the model from focusing only on the user corrections while forgetting the initially good predictions on pixels for which no corrections were given.

Second, we regularize the network to prevent it from forgetting patterns learned on the initial training set, such as object shape priors. Specifically, we add a cost for changing important network parameters [30, 54, 5]:

$$\mathcal{L}_{\mathrm{F}}(\boldsymbol{\theta}) = \boldsymbol{\Omega}^T \left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right)^{\odot 2}, \tag{3}$$

where $\boldsymbol{\theta}^*$ are the initial model parameters, $\boldsymbol{\theta}$ are the updated parameters and $\boldsymbol{\Omega}$ is the importance of each parameter. $(\cdot)^{\odot 2}$ is the element-wise square (Hadamard square). Intuitively, this loss penalizes changing the network parameters away from their initial values, where the penalty is higher for important parameters. We compute $\boldsymbol{\Omega}$ using Memory-Aware Synapses (MAS) [5], which estimates importance based on how much changes to the parameters affect the prediction of the model.

Combined loss. Our full method uses a linear combination of the above losses:

$$\begin{aligned}\mathcal{L}_{\mathrm{ADAPT}}(\mathbf{x}, \mathbf{p}, \mathbf{c}; \boldsymbol{\theta}) =& \lambda \mathcal{L}_{\mathrm{GCE}}(\mathbf{x}, \mathbf{c}; \boldsymbol{\theta}) \\ &+ (1-\lambda)\mathcal{L}_{\mathrm{GCE}}(\mathbf{x}, \mathbf{p}; \boldsymbol{\theta}) \\ &+ \gamma \mathcal{L}_{\mathrm{F}}(\boldsymbol{\theta}),\end{aligned} \tag{4}$$

where λ balances the importance of the user corrections *vs.* the predicted mask, and γ defines the strength of parameter regularization. Next, we introduce *single image adaptation* and *image sequence adaptation*, which both minimize Eq. (4). Their difference lies in how the model parameters $\boldsymbol{\theta}$ are updated: individually for each image or over a sequence.

3.2.1 Adapting to a single image

We adapt the segmentation model to a particular image by training on the click corrections. We start from the segmentation model with parameters $\boldsymbol{\theta}^*$ fit to the initial training set (Sec. 3.1). Then we update them by running several gradient descent steps to minimize (4) every time the user makes a correction (Algo. 1). We choose the learning rate and the number of update steps such that the updated model respects the user corrections. This effectively turns corrections into constraints. This process results in a segmentation mask $\mathbf{p}$, predicted using the updated parameters $\boldsymbol{\theta}$.

Adapting the initial model to the current test image brings two core advantages. First, it learns about the specific appearance of the object and background in the current image. Hence corrections have a larger impact and can also improve the segmentation of distant image regions which have similar appearance. The model can also adapt to low-level photometric properties of this image, such as overall

Algorithm 1 Image adaptation algorithm.

```
1:  function IMAGEADAPTATION(input $\mathbf{x}$, labels $\mathbf{y}$, target iou $\mathcal{J}^t$, initial parameters $\boldsymbol{\theta}^*$, learning rate $\lambda$, number of steps $k$)
2:      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^*$                         ▷ Initialize adaptation model
3:      $\mathbf{c} \leftarrow -\mathbf{1}$                          ▷ Start with no corrections
4:      for $i \leftarrow 0..20$ do                  ▷ Iterate predicting and correcting
5:          $\mathbf{p} \leftarrow \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$                               ▷ Predict mask
6:          $\mathcal{J} \leftarrow \mathrm{IOU}(\mathbf{p}, \mathbf{y})$                        ▷ Compute the IOU
7:          if $\mathcal{J} \geq \mathcal{J}^t$ then           ▷ Stop if mask has required IOU
8:              return $(\mathbf{p}, |\mathbf{1}[\mathbf{c} \neq -1]|, \mathcal{J})$
9:          end if
10:         $\mathbf{c} \leftarrow \mathbf{c} \cup$ GETCORRECTION($\mathbf{x}, \mathbf{p}, \mathbf{c}$)        ▷ User input
11:         $\mathbf{x} \leftarrow$ UPDATEGUIDANCE($\mathbf{x}, \mathbf{c}$)
12:         for step $\leftarrow 1..k$ do          ▷ Update model parameters
13:             $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \mathcal{L}_{\mathrm{ADAPT}}(\mathbf{x}, \mathbf{p}, \mathbf{c}; \boldsymbol{\theta})$
14:         end for
15:     end for
16:     return $(\mathbf{p}, |\mathbf{1}[\mathbf{c} \neq -1]|, \mathcal{J})$
17: end function
```

illumination, blur, and noise, which results in better segmentation in general. Second, our adaptation step makes the corrections effectively hard constraints, so the model will preserve the corrected labeling in later iterations too.

This adaptation is done for each image separately, and the updated $\boldsymbol{\theta}$ is discarded once an object is segmented.

3.2.2 Adapting to an image sequence

Here we describe how to continuously adapt the segmentation model to a sequence of test images using an online algorithm. Again, we start from the model parameters $\boldsymbol{\theta}^*$ fit to the initial training set (sec. 3.1). When the first test image arrives, we perform interactive segmentation using these initial parameters. Then, after segmenting each image $I_t = (\mathbf{x}_t, \mathbf{c}_t)$, the model parameters are updated to $\boldsymbol{\theta}_{t+1}$ by doing a single gradient descent step to minimize Eq. (4) for that image. The updated model parameters are used to segment the next image I_{t+1}.

If all corrections were used equally in the loss and guidance, the model could eventually degrade to simply using the guidance information as a prediction, without relying on image appearance. We avoid this by subsampling the clicks given to the network as guidance, but using all clicks to compute the adaptation loss (Eq. (4)). This forces the network to rely on appearance for propagating the corrections to the rest of the image, where the loss is sparsely evaluated at the pixel locations which were corrected.

Through the method described above our model adapts to the whole test image sequence, but does so gradually, as images arrive in the sequence. As a consequence the process is fast, does not require storing a growing number of images, and can be used in a online setting. In this fashion it can adapt to changing appearance properties, adapt to unseen classes, and specialize to one particular class. It can even adapt to radically different image domains as we demonstrate in Sec. 4.3.

3.2.3 Combined adaptation

The two types of adaptation described above can easily be combined. For a test image I_t, we segment the object using the image adaptation method of Sec. 3.2.1 (Algo. 1). After segmenting a test image, we gather all corrections provided for that image and apply a image sequence adaptation step to update the model parameters $\boldsymbol{\theta}_t$ to $\boldsymbol{\theta}_{t+1}$ (Sec. 3.2.2). At the next image, the image adaptation process will thus start from parameters $\boldsymbol{\theta}_{t+1}$ better suited for the test sequence. This combination allows to leverage the distinct advantages of the two types of adaptation.

3.3. Simulating user corrections

To train and test our method we rely on simulated user corrections, as is common practice [52, 33, 10, 35, 31, 28].

Test-time corrections. When interactively segmenting an object, the user clicks on a mistake in the model prediction. To simulate this we follow [52, 10, 35], which assumes that the user clicks on the largest error region. We obtain this error region by comparing the model predictions with the ground-truth and select its center pixel.

Train-time corrections. Ideally one wants to train with the same user model that is used at test-time. To make this computationally feasible, we train the model in two stages [35].

First, we sample corrections using ground-truth object segmentations [10, 28, 31, 33, 52]. Positive user corrections are sampled uniformly at random on the object. Negative user corrections are sampled according to three strategies: (1) uniformly at random from pixels around the object, (2) uniformly at random on other objects, and (3) uniformly around the object. We use these corrections to train the model until convergence.

Then, we continue training by iteratively sampling corrections [35]. For each image we keep a set of user corrections $\mathbf{c}$. Given $\mathbf{c}$ we predict a segmentation mask, simulate the next user correction (as done at test time), and add it to $\mathbf{c}$. Based on this additional correction, we predict a new segmentation mask and minimize the loss (Eq. (1)). Initially, $\mathbf{c}$ corresponds to the corrections simulated in the first stage, and over time more user corrections are added. As we want the model to work well even with few user corrections, we thus periodically reset $\mathbf{c}$ to the initial clicks.

4. Experiments

We extensively evaluate our single image adaptation and image sequence adaptation methods on several standard datasets as well as on aerial and medical images. These correspond to increasingly challenging adaptation scenarios.

	Berkeley [37]	**YouTube-VOS** [53]	**COCO** [34]		
			seen	unseen	unseen 6k
clicks@q%	90%	85%	85%	85%	85%
Initial model	5.4	8.8	10.0	11.9	13.2
IA	4.9	7.4	9.1	10.7	10.6
SA	5.3	6.9	9.7	10.6	10.0
IA+SA	4.9	-	9.1	9.9	9.3

Table 1: **Adapting to distribution shifts.** Both, single image adaptation (IA) and image sequence adaptation (SA) improve over the initial model with fixed weights when the test distribution differs from training.

Adaptation scenarios. We first consider *distribution shift*, where the training and test image sets come from the same general domain, consumer photos, but differ in their image and object statistics (Sec. 4.1). This includes differences in image complexity, object size distribution, and when the test set contains object classes absent during training. Then, we consider a *class specialization* scenario, where a sequence of objects of a single class has to be segmented (Sec. 4.2). Finally we test how our method handles large *domain changes* where the imaging modality changes between training and testing. We demonstrate this by going from consumer photos to aerial and medical images (Sec. 4.3).

Initial segmentation model. All experiments start from the model of Sec. 3.1, trained on PASCAL VOC12 [20] augmented with SBD [26] (10582 images with 24125 segmented instances of 20 object classes). This initial model already achieves state-of-the-art results on the PASCAL VOC12 validation set, simply by increasing the encoder resolution compared to [35] (Tab. 4). This shows that models with fixed parameters perform well when the train and test probability distributions match. When testing our adaptation method this model serves as our point of reference.

Evaluation metrics. We evaluate with two standard metrics [52, 33, 10, 35, 31, 11, 28]: (1) **IoU@k**, the average intersection-over-union between the ground-truth and predicted segmentation masks, given k corrections per image, and (2) **clicks@q%**, the average number of corrections needed to reach an IoU of $q\%$ on every image (thresholded at 20 clicks).

Hyperparameter selection. We optimize the hyperparameters for both single image adaptation and image sequence adaptation on a subset of the ADE20k dataset [55, 56]. Hence, the hyperparameters are optimized for adapting from PASCAL VOC12 to ADE20k, which is distinct from the distribution shifts and domain changes we evaluate on.

4.1. Adapting to distribution shift

We test how well we can adapt the initial model trained on PASCAL VOC12 to other consumer photos datasets.

Datasets. We test on: (1) *Berkeley [37]*, 100 images with a single foreground object. (2) *YouTube-VOS [53]*, a large video object segmentation dataset. We use the test set of the 2019 challenge, where we take the first frame with ground truth (1169 objects). (3) *COCO [34]*, a large segmentation dataset with 80 object classes. 20 of those overlap with the ones in the PASCAL VOC12 dataset and are thus *seen* during training. The other 60 are *unseen*. We sample 10 objects per class from the validation set and separately report results for seen (200 objects) and unseen classes (600 objects) as in [52, 36]. We also study how image sequence adaptation behaves on longer sequences of 100 objects for each unseen class (named *COCO unseen 6k*).

Results. We report our results in Tab. 1 and in Fig. 3. Both types of adaptation improve performance on all the tested datasets. On the first few user corrections the *single image adaptation* (IA) model performs similarly to the initial model as it is initialized with the same parameters. But as more corrections are provided, it uses these more effectively to adapt its appearance model to a specific image. Thus, it performs particularly well in the high-click regime, which is especially useful for objects that are challenging to segment (*e.g.* due to low illumination, Fig. 4), or when very accurate masks are desired.

During *image sequence adaptation* (SA), the model adapts to the test image distribution and thus learns to produce good segmentation masks given just a few clicks (Fig. 3a). As a result, SA outperforms the initial model on all datasets with distribution shifts (Tab. 1). By adapting from images to the video frames of YouTube-VOS, SA reduces the clicks needed to reach 85% IoU by more than 20%. Importantly, we find that our method adapts fast, making a real difference after just a few images, and then it keeps on improving even as the test sequence becomes thousands of images long (Fig. 3b). This translates to a large improvement given a fixed budget of 4 clicks per object: on the COCO unseen 6k split it achieves 69% IoU compared to the 57% of the initial model without adaptation (Fig. 3a).

Generally, the curves for image sequence adaptation grow faster in the low click regime than the single image adaptation ones, but then exhibit stronger diminishing returns in the higher click regime (Fig. 3a). Hence, combining the two compounds their advantages leading to a method that considerably improves over the initial model on the full range of number of corrections and sequence lengths (Fig. 3a). Compared to the fixed model, our combined method significantly reduces the number of clicks needed to reach the target accuracy on all datasets: from a 9% reduction on Berkeley and COCO seen, to a 30% reduction on COCO unseen 6k.

	clicks @ 85% IoU			
	Donut	**Bench**	**Umbrella**	**Bed**
Initial model	11.6	15.1	13.1	6.8
IA	9.2	14.1	11.9	5.5
SA	7.1	14.0	11.1	5.5
IA+SA	6.5	13.3	10.2	5.0

Table 2: **Class specialization.** Adapting the model to a specific class outperforms the fixed initial model on all tested classes. Naturally, gains are larger for image sequence adaptation, as it can adapt to the class over time.

Method	**DRIONS-DB [14]** **Medical Dataset** clicks@90% IoU	**Rooftop [49]** **Aerial Dataset** clicks@80% IoU
Initial model	13.3	8.9
IA	11.4	6.3
SA	3.6	3.6
IA+SA	3.1	3.6

Table 3: **Domain change results**. Both types of adaptation (IA and SA) outperform the initial model on both datasets. SA requires only 3.6 clicks in both datasets, an impressive reduction of 70% and 60%, respectively.

4.2. Adapting to a specific class

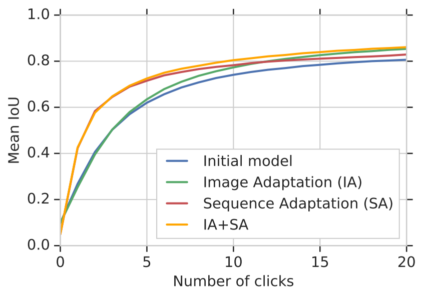
When a user segments objects of a single class at test-time, image sequence adaptation naturally specializes its appearance model to that class. We evaluate this phenomenon on 4 COCO classes. We form 4 test image sequences, each focusing on a single class, containing objects of varied appearance. The classes are selected based on how image sequence adaptation compared to the initial model in Sec. 4.1: (1) donut (2540 objects), the class that improved most, (2) bench (3500), which shows average improvement, (3) umbrella (3979), which does not improve significantly, and (4) bed (1450), one of the few classes where performance degrades slightly.

Results. Tab. 2, Fig. 3c present results. The class specialization brought by our image sequence adaptation (SA) leads to good masks from very few clicks. For example, on the donut class it reduces clicks@85% by 39% compared to the initial model (Tab. 2). Given just 2 clicks, it reaches 66% IoU for that class, compared to 25% IoU for the initial model (Fig. 3c). The results for the other classes follow a similar pattern, showing that image sequence adaptation learns an effective appearance model for a single class.

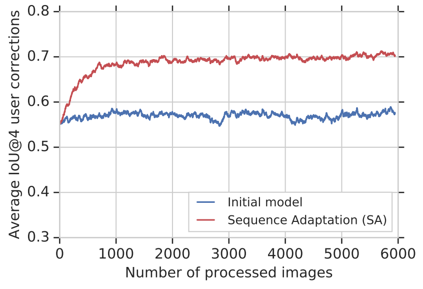
4.3. Adapting to domain changes

We now test our method's capabilities of adapting to domain changes by evaluating on aerial and medical imagery.

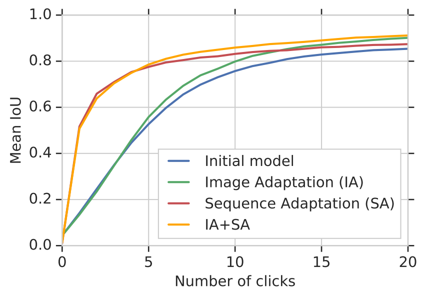
Datasets. We explore two test datasets: (1) *Rooftop Aerial [49]*, a dataset of 65 aerial images with segmented rooftops and (2) *DRIONS-DB [14]*, a dataset of 110 reti-



(a) Average IoU@k for varying k on the COCO unseen 6k split. Both form of adaptation lead to significant improvement compared to the initial model.



(b) IoU@4 clicks as a function of the number of images processed. Image sequence adaptation quickly improves over the initial model with fixed weights.



(c) IoU@k for varying k when specializing to donuts. Image sequence adaptation offers large improvements by learning an appearance model specifically for that class.

Figure 3: **Results for adapting to distribution shifts (a,b), and to a specific class (c).**

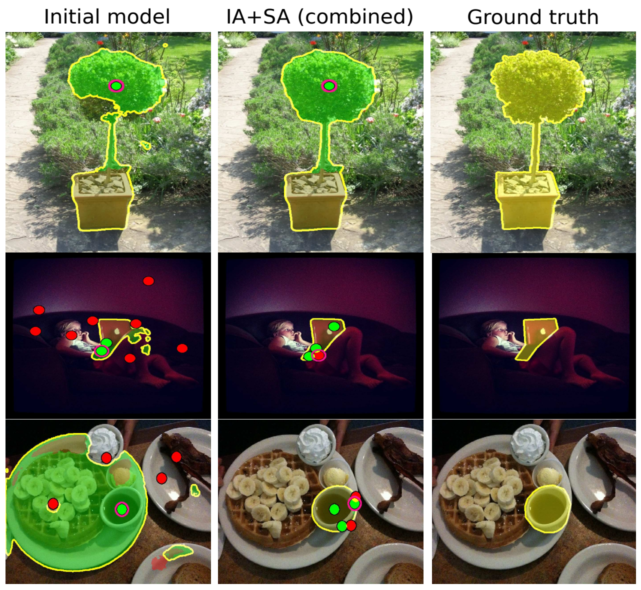


Figure 4: **Qualitative results** of the initial model and our combined adaptation. Red circles are negative clicks and green ones are positive. The red area shows the pixels that turned to background and the green the ones turned to foreground with the latest clicks. The latest click, which lead to this update is highlighted by surrounding it with a pink circle. Our method produces accurate masks with fewer clicks.

nal images with a segmentation of the optic disc of the eye fundus. We use the masks of the first expert. Importantly, the initial model is trained on PASCAL VOC12 and adaptation is tuned for ADE20k, both consumer photos datasets. Hence, we explore truly large domain changes here.

Results. Both our forms of adaptation significantly improve over the initial model (Tab. 3, Fig. 5). Single image adaptation can only adapt to a limited extent, as it independently adapts to individual images starting from the initial model every time. Nonetheless, it offers a significant improvement, reducing the number of clicks needed to reach the desired IoU by 14%-29%. Image sequence adaptation (SA) shows extremely strong performance, as its adaptation effects accumulate over the duration of the test sequence. It reduces the needed user input by 60% for the Rooftop Aerial dataset and by over 70% for DRIONS-DB. This improvement is even larger when combining the two types of adaptation (Tab. 3). Importantly, we find that our method adapts fast: on DRIONS-DB clicks@90% drops quickly and converges to just 2 corrections, as the length of the test sequence increases (Fig. 5a). In contrast, the initial model performs poorly on both datasets. On the Rooftop Aerial dataset, it needs even more clicks than there are points in the ground truth polygons (8.9 *vs.* 5.1). This shows that even a state-of-the-art model like our initial model fails to generalize to truly different domains and highlights the importance of adaptation.

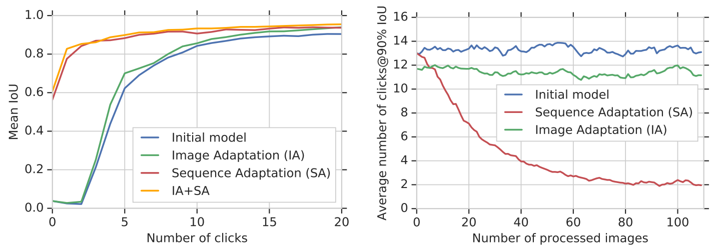
To summarize: We show that our method can bridge large domain changes spanning varied datasets and sequence lengths. With a simple update scheme of doing a single gradient descent step per image, our image sequence adaptation successfully addresses a major shortcoming of neural networks, for the case of interactive segmentation: Their poor generalization to changing distributions [45, 4].

4.4. Comparison to State-of-the-Art

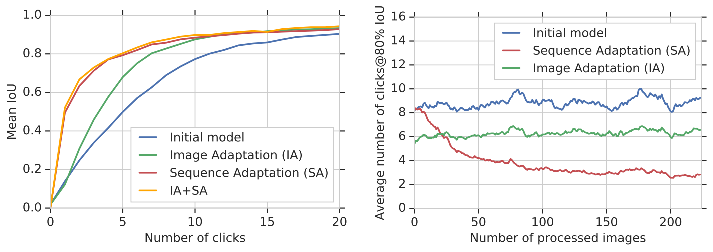
We compare our best method, the combination of single image adaptation and image sequence adaptation (IA+SA), against state-of-the-art methods on standard datasets: Berkeley and COCO, introduced in Sec. 4.1, and the following:

Datasets. (1) *GrabCut [46]*, 49 images with segmentation masks. (2) *DAVIS16 [40]*, 50 high-resolution videos out of which we sample 10% of the frames uniformly at random as in [31, 28, 36, 52] and (3) *PASCAL VOC12 validation*, with 1449 images.

Results. Tab. 4 shows results. Our adaptation method (IA+SA) improves over the initial model on all datasets, and



(a) DRIONS-DB dataset.



(b) Rooftop Aerial dataset.

Figure 5: **Results for adapting to domain change.** For each dataset we show the average IoU@k for varying k on the left and IoU@4 clicks as a function of the number of images processed on the right. Image adaptation provides a consistent improvement over the test sequences. Instead, image sequence adaptation adapts its appearance model to the new domain over time (5a and 5b right). This quickly leads to a largely better IoU for any number of corrections (5a, 5b left).

	VOC12 [20]	**GrabCut** [46]	**Berkeley** [37]	**DAVIS** [40]	**COCO** [34]		
	validation			10% of frames	seen	unseen	average
Method / clicks@q%	85% IoU	90% IoU	90% IoU	85% IoU	85% IoU	85% IoU	85% IoU
iFCN w/ GraphCut [52]	6.88	6.04	8.65	-	8.31	7.82	7.94
TSLFN [27]	4.58	3.76	6.49	-	-	-	9.62
ITIS [35]	3.80	5.60	-	-	-	-	-
CAG [36]	3.62	3.58	5.60	-	5.40	6.10	5.93
BRS [28]	-	3.60	5.08	5.58	-	-	-
Latent Diversity [31]	-	4.79	-	5.95	-	-	7.86
Initial model	3.44	3.29	5.36	6.44	10.03	11.93	11.51
IA+SA combined	3.18	3.07	4.94	5.57	9.14	9.87	9.69

Table 4: **State-of-the-art comparison.** We compare our approach against other interactive segmentation methods in standard datasets. By using corrections to adapt the model parameters, we set a new state-of-the-art on 3 of them (see text).

outperforms or matches state-of-the-art on 4 of the 5 (PASCAL VOC12, GrabCut, DAVIS16 and Berkeley). It brings improvements even when the initial model already offers strong performance and needs less than 4 clicks on average (PASCAL VOC12, GrabCut). The (small) improvement on PASCAL VOC12 further shows that our method helps even when the training and testing distributions match exactly.

On DAVIS16 our method shows strong improvement over the initial model and we match the state-of-the-art method [28], which operates at a higher output resolution. We note that the standard evaluation protocol favors adaptive methods, as the same objects appear repeatedly in the test sequence.

On COCO all adaptation methods substantially improve over the initial model, especially on the unseen classes, where adaptation is particularly useful. Interestingly, the impact of the adaptation is greater on COCO than on the other datasets. COCO is more challenging due to its many small objects and high appearance diversity. While some existing works report even better clicks@q% than us, these cannot be directly compared, since the 10 test objects on which previous works report results are not known. In addition, some previous works [31] predict at the full input image resolution, whereas our backbone network predicts at 1/4 of it. If we ignore objects smaller than 80×80 pixels as in [11], our IA+SA method improves from 9.9 to 5.6 clicks@85% on the unseen classes of COCO. Importantly, our adaptation methods could be applied on top of any network architecture.

4.5. Ablation Study

We ablate the benefit of treating corrections as training examples (on COCO unseen 6k). For this, we selectively remove them from the loss (Eq. (4)). For single image adaptation, this leads to a parameter update that makes the model more confident in its current prediction, but this does not improve the segmentation masks. Instead, training on corrections improves clicks@85% from 13.2 to 10.6. For image sequence adaptation, switching off the corrections corresponds to treating the predicted mask as groundtruth and updating the model with it. This approach implicitly contains corrections in the mask and thus improves clicks@85% from 13.2 for the fixed initial model to 11.9. Explicitly using correction in the loss offers an additional gain of almost 2 clicks, down to 10. This shows that treating user corrections as training examples is key to our method: They are necessary for single image adaptation and highly beneficial for image sequence adaptation.

4.6. Implementation Details

(1) Initial model: We use DeeplabV3+ [17] with Xception-65 [18], pre-trained on ImageNet [19] and PASCAL VOC12 [20], at resolution 513×513. Batch size is 2, atrous rates $\{12, 24, 36\}$. Encoder output stride is 8 and decoder stride is 4. We use SGD with momentum and initial learning rate 0.0002 with polynomial decay. We sample at most 5 foreground and 5 background corrections for stage one of training (Sec. 3.3). Corrections are encoded with disks of radius 3. *(2) Adaptation:* We adapt using Adam [29] with learning rate of 10^{-6} and batch size 1. For single image adaptation we do 10 SGD steps and regularize with $\lambda = 1$ and $\gamma = 1$. For image sequence adaptation we do a 1 SGD step and use $\lambda = 0.5$ and $\gamma = 2$. For the DRIONS-DB dataset we use a larger learning rate of 10^{-5}. We repeated experiments 10 times to test variance and found it to be minimal (≤ 0.01 standard deviation). Hence, we only report averages to improve readability.

5. Conclusion

We proposed an interactive object segmentation method that treats user corrections as training examples to update the model on-the-fly to the data at hand. We have shown experimentally that this enables successfully adapting to distributions shifts and even large domain changes. We have evaluated our method on 8 diverse datasets. On all datasets it shows reductions in the number of click corrections compared to a fixed model.

Acknowledgement. We thank Rodrigo Benenson, Jordi Pont-Tuset and Thomas Mensink for their inputs on this work.

References

[1] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN++. In *CVPR*, 2018.

[2] Adobe. Select a subject with just one click. https://helpx.adobe.com/photoshop/how-to/select-subject-one-click.html, 2018.

[3] Eirikur Agustsson, Jasper RR Uijlings, and Vittorio Ferrari. Interactive full image segmentation by considering all regions jointly. In *CVPR*, 2019.

[4] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *CVPR*, 2019.

[5] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

[6] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *CVPR*, 2019.

[7] Xue Bai and Guillermo Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV*, 2009.

[8] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the point: Semantic segmentation with point supervision. In *ECCV*, 2016.

[9] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *ICCV*, 2019.

[10] Arnaud Benard and Michael Gygli. Interactive video object segmentation in the wild. *arXiv*, 2017.

[11] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, 2019.

[12] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001.

[13] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *CVPR*, 2017.

[14] Enrique J Carmona, Mariano Rincón, Julián García-Feijoó, and José M Martínez-de-la Casa. Identification of the optic nerve head with genetic algorithms. *Artificial Intelligence in Medicine*, 2008.

[15] Ding-Jie Chen, Jui-Ting Chien, Hwann-Tzong Chen, and Long-Wen Chang. Tap and shoot segmentation. In *AAAI*, 2018.

[16] L-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A.L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. on PAMI*, 2018.

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

[18] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012.

[21] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv*, 2018.

[22] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[23] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[24] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, 2010.

[25] Michael Gygli, Mohammad Norouzi, and Anelia Angelova. Deep value networks learn to evaluate and iteratively refine structured outputs. In *ICML*, 2017.

[26] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[27] Yang Hu, Andrea Soltoggio, Russell Lock, and Steve Carter. A fully convolutional two-stream fusion network for interactive image segmentation. *Neural Networks*, 2019.

[28] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *CVPR*, 2019.

[29] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[30] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proc. Nat. Acad. Sci. USA*, 2017.

[31] Z. Li, Q. Chen, and V. Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018.

[32] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. on PAMI*, 2017.

[33] J.H. Liew, Y. Wei, W. Xiong, S-H. Ong, and J. Feng. Regional interactive image segmentation networks. In *ICCV*, 2017.

[34] 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 *ECCV*, 2014.

[35] S. Mahadevan, P. Voigtlaender, and B. Leibe. Iteratively trained interactive segmentation. In *BMVC*, 2018.

[36] Soumajit Majumder and Angela Yao. Content-aware multi-level guidance for interactive instance segmentation. In *CVPR*, 2019.

[37] Kevin McGuinness and Noel E Oconnor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 2010.

[38] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. In *ICCV Workshop*, 2019.

[39] D. P. Papadopoulos, Jasper R. R. Uijlings, F. Keller, and V. Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. In *CVPR*, 2016.

[40] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[41] Brian L Price, Bryan Morse, and Scott Cohen. Geodesic graph cut for interactive image segmentation. In *CVPR*, 2010.

[42] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *CVPR*, 2018.

[43] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016.

[44] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[45] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to CIFAR-10? *arXiv*, 2018.

[46] C. Rother, V. Kolmogorov, and A. Blake. GrabCut - Interactive Foreground Extraction using Iterated Graph Cut. *SIGGRAPH*, 23, 2004.

[47] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, 2017.

[48] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[49] Xiaolu Sun, C Mario Christoudias, and Pascal Fua. Free-shape polygonal object localization. In *ECCV*, 2014.

[50] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. Test-time training for out-of-distribution generalization. *arXiv*, 2019.

[51] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017.

[52] N. Xu, B. Price, S. Cohen, J. Yang, and T.S. Huang. Deep interactive object selection. In *CVPR*, 2016.

[53] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark. *arXiv*, 2018.

[54] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[55] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017.

[56] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20K dataset. *IJCV*, 2018.

[57] Zongwei Zhou, Jae Shin, Lei Zhang, Suryakanth Gurudu, Michael Gotway, and Jianming Liang. Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. In *CVPR*, 2017.