Learning Segmentation from Object Color

Stephan Brehm, Philipp Harzig, Moritz Einfalt, Rainer Lienhart
University of Augsburg
Augsburg, Germany
{stephan.brehm, philipp.harzig, moritz.einfalt, rainer.lienhart}@informatik.uni-augsburg.de

Abstract

We propose Color Shift GAN (CSGAN), a method that allows learning to segment an object class without the need for pixel-wise annotations. We exploit a single textual annotation of the basic object color per image to learn the semantics of an object class. By using only a textual basic color annotation of each object, we are able to drastically reduce labeling efforts. We created a dataset of 29,910 images of cars and annotated the basic color of their body works. Our model reaches 61.4% IoU on our test data. CSGAN trained with additional 128 pixel-wise annotations reaches 62.0%. By adding 45,150 unlabeled images to the training of CSGAN we are able to increase IoU to 65.0% without using a single pixel-wise annotation. This verifies that our weak objective is sufficient for learning segmentation.

1 Introduction

Training supervised deep segmentation models requires lots of data that is annotated at the pixel level. Creating such annotations is very time-consuming. We propose Color Shift GAN (CSGAN), a method that can be used to weakly supervise a deep segmentation model by exploiting only a single textual annotation of the basic color of each object instance. By this, we are able to massively reduce labeling efforts as no pixel annotations are needed. We created a dataset of cars by downloading image search results from various online image search services. Removing wrong search results and annotating object color in 29.910 images took a single person only about 6 days (8h per day). That is more than 10 annotated images per minute.

Color Shift GAN (CSGAN) is a DNN architecture for translating the color of objects of a given object class in images. We define *color translation* as the task of changing the color of objects of a given class in an image to a specific target color. Our intuition is that a deep model, that is able to translate the color of an object, at some point internally needs to infer the location of the object at the pixel level. CSGAN is designed to expose this information. Our CSGAN approach models an image-to-image translation task that changes object colors from any source color to any given target object color. By learning with an adversarial objective, we do not need pairwise data that shows the same object in the same scene with different colors. We propose to separate color translation into the prediction of locations and a separate estimate of the change in color. Because of that, we are able to distill the location information. Our main contributions are:

- We propose a generative model that is in major parts a discriminative model, which we train for segmentation by using only textual information on the color of object instances.
- We present *cars30k*, a dataset consisting of $29,910$ images of cars that are annotated with their respective manufacturer, the car model and the basic color of the car's body works. The *cars30k* dataset is publicly available[1].

2 Related Work

Color based segmentation of images has been studied for decades [2]. However, most algorithms lack the semantic component of our proposed system. Others [15, 18] use additional semantic labels to weakly supervise segmentation models. Mirza et al.[13] introduced Conditional GANs, that allow to use class annotations to explicitly control the output of generative models learned with an adversarial loss. We condition our model on object colors. This allows to explicitly control the target object color. We are able to learn our model without pre-training on any larger database of class annotated images like Imagenet[3] or MSCOCO[10]. Unlike others[17], we also do not learn from pixel-wise masks but use image level color information instead. At test time, the learned model behaves just like any other fully convolutional segmentation network.

[1] http://bit.ly/cars30k

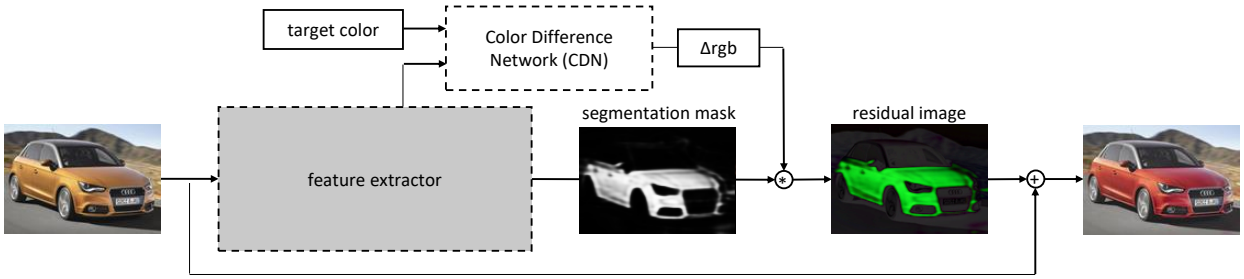

Figure 1. The proposed generator architecture. We extract features by a stack of 10 residual blocks. The CDN (top) uses the features and a given target color to estimate the color change from source color (left, yellow) to target color (right, red). We fuse the estimated intensities and the color vector to create a mask of pixel-wise color changes. Adding these changes to the input image yields a new image with different object color (right). $+/*$ mean pixel-wise addition and multiplication.

3 Color Shift GAN

Segmentation is the task of labeling each pixel according to the class it belongs to. In a binary setting, that is separating pixels that belong to a specific *object* class from *background* pixels. Our intuition is that translating the basic color of an object cannot be done without prior segmenting the image into *object* and *no object*. However, learning to translate object colors in a classic supervised setting would require us to have identical target images for all training examples in which only the color of the object of interest is altered. Gathering such a dataset of real images seems impractical or even impossible. One way to avoid the need for pairwise data is adversarial learning. Due to their formulation, generative adversarial networks (GANs) allow to learn color translation in a weakly supervised manner, in which the generator learns to transfer object color from an input color to a target color without the necessity of pairwise data.

If we consider a model that converts objects in images from one color to another, this model, at some point, needs to identify the location of the object at the pixel level. Otherwise, it could change color in other regions, too. However, in deep models, this information about the location of objects is encoded somewhere in the deep features. By design, CSGAN explicitly predicts object location via a color intensity map. With uniformly colored objects, color translation can be separated into the location at which the color should be changed as well as the actual change in color. We propose a model that explicitly separates these predictions of location and color. At test time we simply strip the color prediction part of the learned model and retrieve the object locations as segmentation masks.

3.1 Generator Architecture

The input of our generator is an image depicting a colored object. The first three convolutional layers perform downsampling with a combined stride of $s = 4$. They are followed by a stack of ten residual blocks[6]. The resulting features are then used as input to a mask prediction head. This mask is then combined with a color change vector that is estimated in the Color Difference Network (CDN). The CDN is a small convolutional network which combines image features with a target color prior.

3.1.1 Color Difference Network

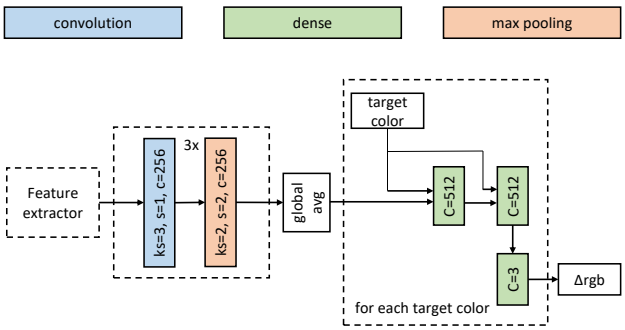

Figure 2. In the CDN, we combine image features with a given target color prior to estimate the color change from source color to target color. Multiple inputs means channel-wise concatenation.

In Figure 2, we depict the **Color Difference Network** (CDN). Its task is to estimate the desired change in object color Δrgb. Estimating Δrgb correctly involves considering the original object color as well as the target color and connecting them in RGB-space. The CDN embeds features obtained by the backbone feature extractor into a vector via a small encoder network consisting of three convolution layers each followed by max pooling with stride $s = 2$.

We reduce the remaining spatial information to a single vector of features by global average pooling. We add a target color prior by stacking a one-hot encoded color vector. The resulting vector is a representation of the source and target color. During training, we use this representation to estimate Δrgb for any possible target color via three dense layers.

3.1.2 Mask Prediction Head

To obtain semantic masks that match the spatial dimensions of the input image, we upsample features from the feature extractor by two consecutive deconvolution layers with stride $s = 2$. We use a single convolution layer to predict a single channel output based on this features. We call this arrangement of layers the *mask prediction head*. We obtain segmentation masks by thresholding the output of the *mask prediction head*. Figure 1 shows the mask prediction head in parallel to the CDN. The output of the mask prediction head represents the intensity of the color change for each pixel of the input image. We multiply it's output with the output of the CDN element-wise to obtain a residual image in RGB-space. Pixel-wise addition of the original image and the residual image yields a new image in which only the color of the object of interest is altered. Note, that the mask prediction head has no knowledge of the target color which is only fed to the CDN. This is why we neither need the CDN nor any color label at test time. To obtain multiple result images with different object colors during training, we only calculate the output of the mask prediction head once, and multiply it with different color vectors obtained from the CDN. This allows us to create images in all possible target colors with only a single forward pass through the feature extractor and the mask prediction head. Outputs x of the last layer in the mask prediction head are activated by a custom activation function. The output y of this function is calculated as:

$$y = \begin{cases} |x| & -1 \leq x \leq 1 \\ 1 + (|x| - 1) \cdot 0.1 & \text{else} \end{cases} \quad (1)$$

Our custom activation function forces segmentation masks to be positive and slows learning when activations exceed a value of 1.0. Keeping values positive has two effects. First, we require intensities to be positive. Second, the resulting activations represent positive pixel-wise scaling factors for the color change vector Δrgb estimated by the CDN. This way, we explicitly assign the task of predicting the direction in color space to the CDN, i.e., the CDN also needs to predict the correct sign of each individual component of the color change vector Δrgb. Note, that other activation functions are also suitable here. However, performance of our models consistently degraded when using sigmoid or ReLU functions instead of our customized activation function. We attribute this to the fact that the gradient of our activation function equals one for all input values between $[-1, 1]$ and the fact that we allow some gradients to flow for input values outside of this range. In contrast, ReLU stops gradient flow for negative feature values and the sigmoid activation function produces fading gradients when close to its limits.

s	8	8	8	8	4	4	4	4	4	2	2	2
$it(\times 10k)$	0	6	12	18	24	30	36	42	48	54	60	66

Table 1. The discriminator growing schedule. Scale s is the ratio between the spatial extent of the input image and the spatial extent of the deep features. We start with images down-sampled by a factor of 8 and increase image resolution over time. New blocks are added every 60,000 iterations (it). The number of filters that are learned in each convolution layer is $s \times 64$.

3.2 Discriminator Architecture

We use a deep markovian discriminator (PatchGAN[7]), which consists of stacked residual blocks[6], i.e., we calculate the error of the discriminator as the average error on an output map instead of the error of a single output neuron. We grow residual blocks progressively as shown in [8] by increasing the depth of the discriminator as training progresses. This allows growing the learned object model from low frequency coarse contents in images in the beginning to fine grained high frequency details towards the end of learning. During the growing phase, we slowly fade in new residual blocks and increase the image resolution. The growing schedule is given in Table 1. We add a new block every 60,000 iterations and slowly fade it in over a period of 30,000 iterations. Note that unlike [8] we do not grow the generator because it largely operates on a constant resolution at a quarter of the original image size anyways.

3.3 Experimental Framework

Optimization We use the Adam Optimizer [9] with $\beta_1 = 0.5$ and $\beta_2 = 0.9$ and learning rate $\eta = 0.0001$. We trained and tested all models on a single NVIDIA Tesla V100. Training time for 2 million iterations is ≈ 9 days when using images scaled to 128 pixels on the long side. We use a standard GAN objective as proposed by Goodfellow et al.[5] with additional discriminator regularization as proposed by Roth et al.[16]. Each training step consists of generator update and discriminator update. The generator network is trained with a batch size of 1. The discriminator sees a single real image and all differently colored images that were created by the generator. Note that our discriminator has an output map for each color instead of just a single output neuron.

Figure 3. [Best viewed in color] Predicted segmentation masks and the resulting color augmented objects. On the left is the source image. The CSGAN column shows the semantic masks predicted by our generator. The VOC column shows the masks predicted by the additional output layer of the combined (w/s) variant. The columns labeled with colors, show the color translated results using the color vector predicted by the CDN. Note, that although not all colors are translated correctly, the masks predicted by CSGAN are very clear and precise.

Color Supervision We use the information about the basic color of our objects to supervise the discriminator, i.e., we define separate output neurons for each color. Thus, we ask multiple questions: "Does the image show a real red object or a fake red object?", "Does the image show a real blue object or a fake blue object?"... and so on. Color supervision requires us to provide learning feedback only for the relevant output neurons, i.e., we only propagate errors for the output map that represents the current color label. That is because the question "does this image show a real blue object or a fake blue object?" does not make any sense if we look at a red object. Note, that the generator network is always conditioned on the target color by feeding the one-hot encoded target color to the CDN.

Dataset To the best of our knowledge, there is no public dataset with annotated object color. Thus, we cannot compare our model on any public benchmark data. We do, however, train a variant of CSGAN using our weak objective and additional strong supervision. We report intersection over union scores of this combined model on the Pascal VOC2012 segmentation validation set in section 4. We created a dataset of 29,910 images of cars from the internet. All images were annotated with one of 18 different car manufacturers and 67 specific models as well as the dominant color of their bodywork. We distinguish 11 different colors. All images show at least one car very prominently. During training we re-balance our dataset such that all colors are equally likely. For testing we labeled the colored regions of the body work of cars in 67 images. We make the *cars30k* dataset publicly available in order to support development in fields in which cars and/or the colors of objects are important.

Data Augmentation and Preprocessing We apply different augmentation techniques that target color values, scale and location. Augmenting color values, in our setting has to be done very carefully. A strong change in the color of an object would counteract our proposed training procedure, because the discriminator relies on getting the correct color label. Thus, we only change image colors slightly by randomizing image brightness, image contrast and color saturation. We normalize RGB values to a range between -1 and 1. The default image size used during training is 128 pixels on the longer side of the image. We allow a random deviation from that size of at max 30% in size. All images are resized such that the original aspect ratio is preserved. Augmentation in location is achieved by randomly flipping images from left to right at a rate of 0.5. We do not pad the images.

3.4 Additional Details

We use Leaky ReLU[12] activation functions and spectral normalization[14] in all layers except for the output layers. All residual blocks are pre-activated. We do not use any feature normalization method. Many feature normalization techniques scale or shift feature values depending on the values or statistics of other features. Due to these dependen-

cies on other features keeping color information is particularly hard in feature normalized networks. We use reflection padding to keep spatial dimensions of the activations constant across all layers that operate on the same scale. In our case, many objects are located somewhere in the center of the images. Knowing image boundaries would make it easy to identify the center of the image and probably result in a location dependent bias of the network output. Reflection padding makes finding the boundaries of an image harder for the network.

Figure 4. Images from the MSCOCO dataset. We expand masks from CSGAN to a convex hull in order to make our object model more compatible to the MSCOCO object model.

4 Results

We conducted experiments on our dataset that consists of images of cars. We show results in Table 2. The basic CSGAN *w* is purely weakly supervised with color annotations from the *cars30k* dataset. We compare to multiple variants of CSGAN that are learned with additional pixel-wise masks from different subsets of the Pascal VOC2012 segmentation set[4]. We show that a variant of CSGAN that is learned using only weak supervision and additional unlabeled images is superior to the variant that is trained with masks from the VOC2012 data.

4.1 Can we do better with additional data?

We conduct two types of experiments to test the effects of additional data on the performance of CSGAN. First, we add strong supervision using annotations from the Pascal VOC2012 segmentation training set. Second, we train a weakly-supervised CSGAN with additional unlabeled data.

The object model of our weakly-supervised CSGAN differs from the object model defined by the VOC2012 segmentation data. Due to the formulation of our model, we are only able to segment object parts that are colored uniformly. For cars, that is usually the body works. The object model for the class *car* in the VOC2012 segmentation data is fundamentally different. Parts like windows and wheels that are colored differently from the body works also belong to the object. This means, we cannot use the same parameters to learn our weakly supervised object model and the VOC2012 segmentation model. We use an additional output layer in parallel to our mask prediction head. The conflict between our object model and the VOC2012 object model is also observable in the scores that we report in Table 2. A weakly supervised CSGAN that is strongly supervised with additional 128 masks of cars (*w/s* cars only) achieves only 60.4% IoU on our test data while a model that is supervised with all VOC2012 classes (*w/s* all cls w/o cars) except for the *car* class reaches 61.9%. We learn the VOC2012 segmentation data with a softmax activated cross-entropy target. This way, both our mask prediction head and the VOC2012 head utilize the same features, but predict different outputs. During training, in each step, we feed a single image from the VOC2012 dataset as well as a single image from our color labeled dataset. We report intersection over union scores (IoU) on the subset of the VOC2012 segmentation dataset that shows cars as well as the IoU on our dataset wherever applicable. We compare multiple models (*w/s*) trained with our weakly supervised target and additional labels from different subsets of the VOC2012 segmentation data. Here, *s* means that we used pixel-wise masks from the Pascal VOC2012 data for training. *w* means that we used our weakly supervised objective. *w/s* means that we used both weak and strong supervision. *cars only* means that we only used images that show cars from the VOC2012 training set. *All cls* means that we used masks of all 20 object classes annotated in the VOC2012 data. *all cls w/o cars* means that we used all VOC2012 masks except the car masks. Note, that some images that show cars, also show other annotated objects. In that case we did not remove the images but simply erased the car annotations. We also trained a purely supervised model (*s*) using all of the 20 object classes annotated in the VOC2012 segmentation set. Using additional pixel-wise annotations did not improve our results which strongly reinforces our statement, that color is enough to learn the semantics of certain objects.

In CSGAN, by design, the training of the generator does not need any color annotations. Color labels are only needed to train the discriminator. We train another CSGAN using all images from the *cars30k* dataset and 45,150 additional unlabeled images of cars from the internet. This model is learned using only the weak objective of CSGAN. It achieves 65.0% IoU on our test data which is considerably higher than all other versions of CSGAN.

4.2 Discussion

CSGAN can segment all types of objects that commonly exhibit one dominant color but occur in multiple different colors. Examples are vehicles like bicycles, cars, airplanes, boats and buses as well as animals like horses, dogs, sheep and many others. Crafted objects like handbags, umbrellas, doors and laptops are also often uniformly colored. We tested CSGAN not only with images of cars, but also with images of alpacas and images of handbags. In both cases, CSGAN produced reasonable results even though we

	w	*w* + unlabeled data	*w/s* cars only	*w/s* all cls	*w/s* all cls w/o cars	*s* all cls
#voc img	0	0	128	1,464	1,415	1,464
#car img	29,120	29,120	29,120	29,120	29,120	0
#unlabeled img	0	45,150	0	0	0	0
IoU(cars30k)	0.614	**0.650**	0.604	0.620	0.619	-
IoU(cars VOC)	-	-	**0.438**	0.409	-	0.437

Table 2. Results of our experiments of CSGAN trained with weak supervision *w*) and strong supervision (*s*). *w/s* describes the variants of CSGAN that are trained using weak and strong supervision. Rows one to three give the number of images from the respective datasets that were used for training.

trained with datasets consisting of less than 500 images.

In Figure 3 we show qualitative results. The results of the color translation look very realistic. The masks estimated by CSGAN are very detailed with good alignment at the borders of objects. Note, that this is due to the assumption that objects of interest are uniformly colored to a large extent. This is both an advantage as well as a disadvantage of CSGAN. The advantage is that when it can be used the resulting masks are very accurate. The disadvantage is that not all types of real objects exhibit a dominant color. Others, such as tires, only exist in one color. In those cases, the proposed color translation is not possible. Many objects can also be divided into parts that a CSGAN can learn to segment. Examples include but are not limited to houses (facade, roof tiles) and humans (shirt, trousers, shoes, skin). Segmentations of individual parts of objects would also provide richer information than segmentations of the whole object. Note that, scores achieved on the Pascal VOC2012 segmentation validation set are relatively low in comparison to other methods on this dataset. We attribute this to the very low image resolution that we used. In comparison, common methods[1, 11] use images scaled to 500 pixels or more on the longer side. CSGAN is learned using the assumption that images show a single foreground object. Thus, by default, it does not learn to segment multiple objects in images as long as they do not share a common color, but instead focuses on the largest object. We add an additional object detection step before segmenting in order to circumvent this restriction of CSGAN. Figure 4 shows results on images with multiple instances from the MSCOCO dataset[10].

5 Conclusion

We proposed Color Shift GAN, a framework that allows to train a segmentation model without the use of pixel-wise annotations. CSGAN can be learned with image level textual color annotations. Thus, we are able to drastically reduce labeling efforts in comparison to the efforts needed to create pixel-wise annotations. Due to the use of the proposed Color Difference Networks we are able to separate object location from object color. The adversarial setting allows us to train the generative part of CSGAN without using any color labels at all. Thus, we are able to increase dataset size easily at nearly zero cost. The learned segmentations are very detailed especially at the borders of objects.

References

[1] L.-C. Chen et al. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[2] H. Cheng et al. Color image segmentation: advances and prospects. *Pattern Recognition*, 2001.

[3] J. Deng et al. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR*, 2009.

[4] M. Everingham et al. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[5] I. Goodfellow et al. Generative adversarial nets. In *NIPS*, 2014.

[6] K. He et al. Deep residual learning for image recognition. In *IEEE CVPR*, 2016.

[7] P. Isola et al. Image-to-image translation with conditional adversarial networks. In *IEEE CVPR*, 2017.

[8] T. Karras et al. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[10] T.-Y. Lin et al. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[11] J. Long et al. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, 2015.

[12] A. L. Maas et al. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

[13] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[14] T. Miyato et al. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[15] D. Pathak et al. Constrained convolutional neural networks for weakly supervised segmentation. In *IEEE ICCV*, 2015.

[16] K. Roth et al. Stabilizing training of generative adversarial networks through regularization. In *NIPS*. 2017.

[17] R. R. Shetty et al. Adversarial scene editing: Automatic object removal from weak supervision. In *NIPS*, 2018.

[18] Y. Wei et al. Stc: A simple to complex framework for weakly-supervised semantic segmentation. *PAMI*, 2017.