CIS680 Final Project: Data augmentation with generative adversarial networks

Nikolaos Kolotouros
nkolot@seas.upenn.edu

Karl Pertsch
pertsch@seas.upenn.edu

Oleh Rybkin
oleh@seas.upenn.edu

Abstract

In this work we propose a method for generating synthetic scenes that can be used for dataset augmentation. We employ a generative adversarial network (GAN) for creating semantic segmentation masks that are passed to a second network which generates realistic scenes. The components of our architecture are independently pre-trained and the full pipeline is finetuned in an end-to-end fashion. We also demonstrate the end-to-end training from scratch. A mutual information loss formulation further enforces the generation of varied scenes. This work is a step towards using GANs to incorporate unsupervised data in supervised problem settings.

1. Introduction

Despite the recent advances in the field of deep learning, a problem that persists is the need for vast amounts of labeled data. As a consequence, huge efforts have been made to artificially increase the dataset size by applying augmentation methods [13, 18, 6]. However, these methods are often hand-constructed, e.g. cropping or flipping the image, and thus limited in their ability to increase the diversity of the training data. Improvements in generative adversarial networks (GANs) [5] open the opportunity to go beyond previous simple augmentation schemes and generate labeled data from scratch, making use of the increased variability present in unlabeled data.

GANs have proven to be an effective tool for generating new samples that are similar to a given data distribution, either only from random noise [16, 12, 4] or conditioned on an input [14, 19, 11, 17, 21]. Recent methods have explored these generative capabilities of GANs for data augmentation: in [17] the authors use a GAN to refine synthetic images generated from the 3D model of a human eye; in [19] a similar procedure is applied to generate realistic images of barcode-like markers. The results have shown that augmentation with newly generated data yields considerable improvement in accuracy, making use of the GAN's ability to process great amounts of unlabeled real data.

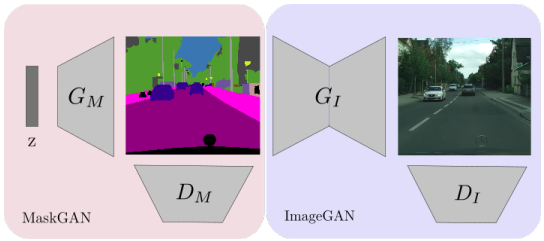

Figure 1: Our proposed scene generation model. MaskGAN creates a synthetic segmentation mask that is used as input for the image generation network ImageGAN. To generate a new scene, the latent representation z is sampled from a normal distribution and passed through the two-network pipeline.

In this work we propose a method that can generate pairs of realistic images and semantic segmentation masks, using a fully CNN-based pipeline (see figure 1). While approaches like [17, 19] rely on refining synthetic images rendered from a separate process, our method is able to generate new images by directly sampling from a random noise distribution. Additionally, after pre-training the generators for segmentation masks and images separately, our pipeline can be fine-tuned in an end-to-end fashion using an unsupervised training procedure.

Our proposed architecture consists of two models: a generative network, *MaskGAN*, that creates semantic segmentation masks using random noise input and a second GAN, *ImageGAN*, that processes the generated masks and outputs a corresponding scene image. While ImageGAN can be any conditional scene generator network, we employ an architecture similar to the one proposed by Isola et al. [11].

In order to validate our approach we train our pipeline on the Cityscapes dataset [2] and show the successful generation of varied semantic segmentation masks. We also demonstrate the end-to-end training of our full pipeline both for finetuning and from scratch.

Our work makes three main contributions:

- demonstrate the creation of realistic segmentation

masks from random noise using GANs

- improve the variability of generated masks by enforcing a mutual information loss between the noise input and the generated masks
- introduce network architectures for the end-to-end training of a segmentation augmentation pipeline

The remainder of this paper first gives a brief introduction into the background of our work in Section 2, then introduces our proposed architecture in Section 3, explains the conducted experiments in Section 4 and summarizes our findings in Section 5.

2. Background

Generative Adversarial Networks (GANs) were first introduced by Goodfellow et al. [5] and have since found widespread application in various fields ranging such as Computer Vision and Reinforcement Learning [11, 8]. GANs formulate the problem of generating an image from a latent data distribution p_{data} as a minimax game between two adversarial networks: a generator $G(\boldsymbol{z})$ that generates an image from a noise distribution $\boldsymbol{z}$, and a discriminator $D(\boldsymbol{y})$ that predicts whether its input $\boldsymbol{y}$ is sampled from the true data distribution p_{data} or the distribution $p_{\boldsymbol{z}}$ that generates from noise. The objective V can be formalized as:

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{\boldsymbol{y}\sim p_{data}(\boldsymbol{y})}\big[\log\big(D(\boldsymbol{y})\big)\big] \\ &+ \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\big[\log\big(1-D(G(\boldsymbol{z}))\big)\big]\end{aligned} \tag{1}$$

It was proven, that this problem has a single optimum when the generator distribution $p_{\boldsymbol{z}}$ matches the training data distribution p_{data} [5]. The concept of GANs has been extended to networks that are conditioned on an input $\boldsymbol{x}$, i.e. Conditional GANs. The discriminator $D(\boldsymbol{x},\boldsymbol{y})$ decides whether a *pair* of image and conditioning input is drawn from the data distribution or generated by the generator $G(\boldsymbol{x},\boldsymbol{z})$ that draws from the noise distribution and is conditioned on the same input $\boldsymbol{x}$. In [11] Conditional GANs are used to perform image to image translation, e.g. between semantic segmentation masks and street scenes. While recent results using GANs are impressive [20], the networks struggle to model multi-modal data distributions, especially in the unconditional case, i.e. the generator learns to produce samples from a small fraction of the real data distribution with high quality, resulting in low discriminator loss but also low output variability. A way to overcome this is to force the generator to use the noise that is given as input. In the InfoGAN architecture [1] this is done by predicting the latent input variables of the generator in the discriminator and penalizing both networks with a loss on this prediction. The GAN objective in Equation (1) is augmented with a term that measures the mutual information between the latent input of the generator $\boldsymbol{c}$ and its output $G(\boldsymbol{c},\boldsymbol{z})$. The mutual information $I(\boldsymbol{c},G(\boldsymbol{c},\boldsymbol{z}))$ is defined by:

$$I(\boldsymbol{c},G(\boldsymbol{c},\boldsymbol{z})) = H(\boldsymbol{c}) - H(\boldsymbol{c}|G(\boldsymbol{c},\boldsymbol{z})) \tag{2}$$

Here $H(\cdot)$ denotes the entropy. While (2) is hard to compute exactly as it requires access to the posterior $P(\boldsymbol{c}|\boldsymbol{y})$, it can be approximated by its lower bound $Q(\boldsymbol{c}|\boldsymbol{y})$ that can be predicted with an arbitrary regressor, e.g. a neural network. Following [1], the mutual information loss $L_I(G,Q)$ can be defined as follows:

$$L_I(G,Q) = \mathbb{E}_{\boldsymbol{c}\sim P(\boldsymbol{c}),\boldsymbol{y}\sim G(\boldsymbol{c},\boldsymbol{z})}\big[\log Q(\boldsymbol{c}|\boldsymbol{y})\big] + H(\boldsymbol{c}) \tag{3}$$

The GAN objective is then extended weighting the mutual information loss with a hyperparameter λ:

$$\min_{G,Q}\max_D V_{\text{InfoGAN}}(D,G,Q) = V(D,G) - \lambda L_I(G,Q) \tag{4}$$

3. Method

Our architecture divides the task of generating a pair of semantic segmentation mask and scene image into two sub-tasks (see Figure 1). A first network, *MaskGAN*, predicts the segmentation mask from a sample of the noise distribution $\boldsymbol{z}$, outputting a RGB image of the segmentation mask. The second stage, *ImageGAN*, uses the image to image translation of [11] to generate a scene image conditioned on the mask input. Both are trained separately at first and then finetuned in an end-to-end fashion. The following paragraphs describe the individual stages of our model in more detail. If not noted otherwise, all layers are followed by a batch normalization layer [10] and a ReLU nonlinearity [13].

3.1. Segmentation Mask Generation

The first network, *MaskGAN*, consists of an encoder $G_M(\boldsymbol{c},\boldsymbol{z})$ that takes latent variable $\boldsymbol{c}$ and noise $\boldsymbol{z}$ as input and outputs a semantic segmentation mask $\boldsymbol{y}\in\mathbb{R}^{n_i^2\times 3}$ in form of an RGB image with resolution n_i. The latent variable $\boldsymbol{c}$ is modeled as a categorical variable $c_i\in\{0,1\}^{n_c}$, i.e. a one-hot vector of dimension n_c. The noise input $\boldsymbol{z}\in\mathcal{N}(0,1)^{n_z}$ is sampled from a zero-mean normal distribution with unit standard deviation. Both inputs are concatenated and passed through two consecutive fully-connected network layers. The resulting embedding is reshaped and processed by a stack of deconvolutional layers to generate the output of dimension n_i. The final output is produced with a hyperbolic tangent nonlinearity to bound the output within the RGB color scale. As an alternative to the RGB regression task presented here we experimented with pixel-wise classification to retrieve a segmentation mask, but found training to be very slow in comparison.

The discriminator $D(\boldsymbol{y})$ of the MaskGAN network first processes the input RGB segmentation mask with a stack of convolutional layers, that decreases resolution while increasing the number of channels. The final layer again replaces batch normalization and ReLU with an alternative non-linearity, a sigmoid function, to bound the output $D(\boldsymbol{y}) \in [0,1]^{n_d \times n_d}$. The discriminator outputs confidence values for n_d^2 patches of the image. On a second branch the latent posterior distribution $Q(\boldsymbol{c}|\boldsymbol{y})$ is predicted for the InfoGAN loss. To enforce variability on a macro-scale, the segmentation mask is first downsampled to a resolution $n_{i'} \ll n_i$ and then passed to a single convolutional layer followed by a softmax output in order to predict a probability distribution.

In accordance to Equation (1), the discriminator GAN loss $L_{D_{\text{GAN}}}$ is defined by:

$$L_{D_{\text{GAN}}} = -\big[\log(D(\boldsymbol{y})) + \log\big(1 - D(G(\boldsymbol{c}, \boldsymbol{z}))\big)\big] \quad (5)$$

Following [9], the generator GAN loss $L_{G_{\text{GAN}}}$ can be slightly reformulated for more stable training resulting in:

$$L_{G_{\text{GAN}}} = -\log\big(D(G(\boldsymbol{c}, \boldsymbol{z}))\big) \quad (6)$$

In addition it proved useful to add a L_1 loss to the generator objective in order to speed-up convergence to meaningful masks:

$$L_{L_1} = \big|\boldsymbol{y} - G(\boldsymbol{c}, \boldsymbol{z})\big| \quad (7)$$

Finally the mutual information loss between $\boldsymbol{c}$ and $Q(\boldsymbol{c}|\boldsymbol{y})$ can be computed by:

$$L_{MI} = -\big[\log\big(Q(\boldsymbol{c}|\boldsymbol{y})\big) \cdot \boldsymbol{c} - \log\big(\boldsymbol{c}\big) \cdot P(\boldsymbol{c})\big] \quad (8)$$

When adding all loss components with their respective weighing factors $\lambda_i > 0$ we obtain our final loss formulations for the MaskGAN network:

$$\begin{aligned} L_D &= \lambda_{\text{GAN}} \cdot L_{D_{\text{GAN}}} + \lambda_{\text{MI}} \cdot L_{\text{MI}} \\ L_G &= \lambda_{\text{GAN}} \cdot L_{G_{\text{GAN}}} + \lambda_{\text{MI}} \cdot L_{\text{MI}} + \lambda_{L_1} \cdot L_{L_1} \end{aligned} \quad (9)$$

3.2. Boundary Mask Generation

In a second set of MaskGAN experiments, we aimed at generating boundary masks along with the segmentation masks, in order to resolve ambiguities in overlapping objects and enable the generation of augmentation data for object detection methods. We therefore introduce a second branch in our network, that processes intermediate feature maps in the generator with a separate set of upconvolutional layers and outputs a boundary mask $\boldsymbol{y}_b \in [0,1]^{n_i \times n_i}$. This mask is then concatenated with the output of the semantic mask prediction branch and jointly refined using an hourglass architecture [15]. It consists of an encoder and a decoder part that share features through skip connections and both make heavy use of residual blocks [7]. Besides adding more weights to the network, the joint refinement encourages consistency between the the masks and the encoder-decoder structure enables the usage of global image features.

The loss formulations from the previous section still apply, as the boundary mask is simply treated as an additional channel in the target image. We supervise the training with the full loss formulation before and after the refinement. The boundary masks for the target semantic segmentation masks are computed by convolving a Laplacian Filter with the corresponding instance masks. We experimented with using the same procedure to initialize the boundary prediction by convolving the regressed semantic segmentation mask with a Laplacian Filter. However, the noise in the segmentation mask propagated to the resulting boundary mask, effectively hampering training.

We employ the hourglass refinement at full mask resolution n_i. Yet, we tried to operate on downsampled images and upsample in the end in order to reduce the number of parameters that need to be learned. However, the training did not progress in this setting, probably the discriminator quickly learns the artifacts introduced by the upsampling and can therefore perfectly distinguish between real and generated images, while the generator has no means to compensate for the artifacts.

3.3. Scene Image Generation

For the scene image generation part we used a Conditional GAN with an architecture similar to [11] and [20]. Specifically, our first approach was to replicate the architecture of [11]. We conditioned our image generation network on the semantic segmentation masks provided in the Cityscapes dataset [3]. However, the results were not always satisfying. We observed that in some frames, the network could not distinguish between instances of the same class that overlapped, and also most images produced by the network had noisy artifacts.

One of the main reasons why Pix2Pix [11] fails to produce realistic images is because it uses a simple loss function for the generator that consists of the GAN loss coupled with an L_1 loss that tries to match the generated image with the ground truth image. This combined loss function focuses on low-level features of the image and fails to capture useful information that can be extracted from higher-level features. So, in an approach similar to [20], we tried to enforce the generator to generate images that match the ground truth image features at multiple hierarchical levels. The generator loss function that we used in training is

$$L_G = L_{GAN} + L_{feat} \quad (10)$$

where

$$L_{feat} = \sum_{i=1}^{N_d} \frac{1}{N_i} ||D^{(i)}(s, G(s)) - D^{(i)}(s, I_{gt}(s)||_1 \quad (11)$$

and L_{GAN} is the usual generator loss in GANs. In the above equation, N_d is the number of layers in the discriminator, N_i the number of neurons in the i-th layer and $D^{(i)}$ the values at the i-th layer of the discriminator. Essentially, L_{feat} uses the features learned by the discriminator at all layers of the network to force the discriminator to try and match these features between the produced and ground truth images ($G(s)$ and $I_{gt}(s)$ respectively). Indeed, using this loss function we observed a slight improvement in the quality of the output images compared to the simpler approach adopted in pix2pix.

To further improve the quality of our output images we also attempted to use the ground truth boundary masks, i.e. condition ImageGAN on both semantic segmentation scenes and object boundaries. In the simpler architecture that uses only semantic segmentation masks we observed that when a mask contained overlapping instances of objects in the same class, the results tended to be blurry, so the boundaries would help the network to identify these objects as separate instances. Indeed, this technique improved the overall quality of the output images, but as noted previously, we had to remove it from our final architecture because we were not able to succeed in generating realistic image boundaries using MaskGAN.

3.4. End-to-end Training

For the end-to-end training we propose two different approaches. In a first experiment we *finetune* the two pretrained stages. We add supervise the pipeline with a GAN loss on the generated output scene images. In this setting there is no need for mask annotations on the training dataset. However, we have the option to add an intermediate supervision in form of a second GAN loss on the mask level, that prevents the network from deviating from real masks in the intermediate representation. In this setting we also add the mutual information loss on the mask stage, to further encourage variability in the generated scenes.

We additionally propose an architecture that enables the end-to-end training of the whole pipeline *from scratch*. The main challenge here is to incorporate the L_1 loss on the final image stage, as it has proven vital for convergence during the individual training of the image generation network. However, we cannot enforce L_1 consistency between the real target image that is fed to the discriminator in the scene generation stage and the image predicted from the generated mask, as both scenes are possibly very different. Instead we do two forward passes through the ImageGAN generator and use the discriminator on both outputs, enforcing L_1 consistency only between the real scene image and the one generated from the real mask. All losses are then added in a weighted sum and optimized jointly in a single backprop pass.

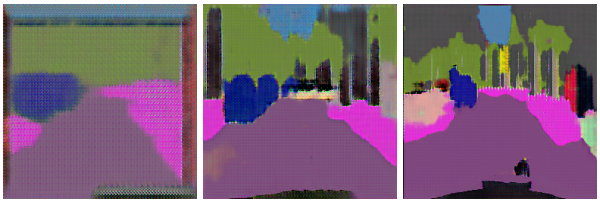

Figure 2: Output of MaskGAN at different stages of training. **left to right**: after 12 200, 18 200 and 59 300 iterations. The noise level is reduced while the detail in the masks increases.

4. Experiments

For the experiments we followed the structure of our architecture: we first trained MaskGAN and ImageGAN separately and then combined them in an end-to-end training. All experiments were carried out on a Nvidia Tesla V100, our implementation in Tensorflow can be found in the attached code. Throughout all experiments we use an image/mask resolution of 256.

4.1. Segmentation Mask Generation

For the training of the segmentation mask generator, we choose the dimension of the noise input $n_z = 100$ and the one of the latent variable $n_c = 20$. We train the network until the visual quality of the masks does not improve anymore, resulting in a total of approximately 60 000 iterations with batch size 10. The architecture of the generator can be found in the appendix in Tab. 1, and for the discriminator in Tab. 3.

Figure 2 shows samples from the training process of MaskGAN. It can be seen, that over time, the noise, that is dominant in the leftmost mask is reduced, while the detail, especially in the background increases. This shows that the discriminator gets more sensitive towards details in the image and the generator adapts its outputs accordingly.

We examine the influence of the different inputs on the output of the MaskGAN in Figure 3. It can be clearly seen, that the mutual information loss L_{MI} forces the generator to use the latent input variable in the generation process. When changing its value, the appearance of the scene changes substantially. It's only because the variable $\boldsymbol{c}$ is thereby encoded in the mask that the discriminator has a chance to infer its value. In contrast, the network seems to largely ignore the noise input vector $\boldsymbol{z}$. We examine similar mode collapses when training completely without the mutual information loss, i.e. the generator ends up produc-

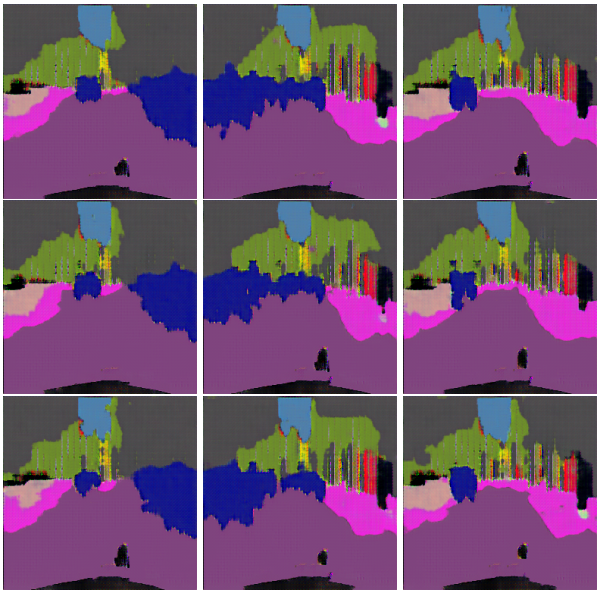

Figure 3: Variation in MaskGAN outputs using InfoGAN loss. **left to right**: variations in the latent variable $\boldsymbol{c}$, **top to bottom**: variations in the input noise distribution $\boldsymbol{z}$. Changing the latent variable has strong influence on the output of MaskGAN while changing the input noise has nearly no effect.

ing only very little variation in its output. This proves, that the introduction of the mutual information loss is vital for generating variability in the MaskGANs output. Additionally, the mutual information loss converged very fast in all experiments we conducted, adding the variability virtually free of charge.

Note that with the used representation of $\boldsymbol{c}$ as a one-hot encoded vector, the number of distinct scenes the generator can create is limited to the dimension of the latent variable n_c. However, the latent representation can be easily extended to include non-binary discrete variables or even continuous distributions [1], giving rise to a further increased variability in the output. While the computation of the mutual information changes mathematically, the concept remains the same.

In Figure 4 we can see the the evolution of the discriminator and generator losses for MaskGAN. We observe that in general the training procedure is not very stable, but this is generally expected when training GANs, because the form of the loss function changes constantly during training. As a general trend however we can observe that when the generator loss increases, the discriminator loss decreases and vice versa. Another factor that contributes to the large variations of the loss during training is the use of a relatively small batch size, because of memory limitations.

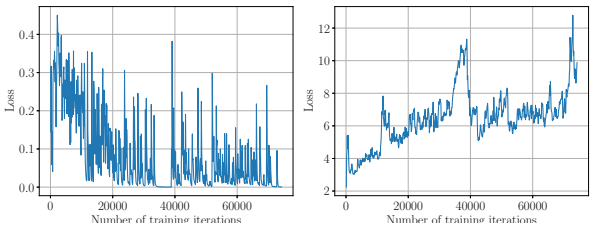

Figure 4: Training losses over number of iterations for MaskGAN. **left to right**: discriminator loss, generator loss

4.2. Boundary Mask Generation

For the generation of instance boundary maps along with semantic segmentation masks we employ the architecture detailed in Section 3.2. Our results are summarized in Figure 5. On the left we show examples of groundtruth masks with corresponding boundary maps, generated by convolving the semantic segmentation with a Laplacian filter. The figure then depicts the predictions of the generator in the intermediate stage, before the refinement. Note that we employ GAN supervision already at that stage, so the network could be able to learn meaningful masks and boundaries already here. However, the results we get are very noisy and of low quality. The images on the right show the output of the refinement stage. It can be seen, that the network learns to predict meaningful boundaries, that correspond to the predicted masks. However, the quality of the mask itself is low: it features a lot of noise and in large parts the regressed colors deviate far from the actual data distribution. Also note, that the generated scene after refinement deviates severely from the one passed to the hourglass. This suggests that the refinement step largely ignores the input and rather memorizes the segmentation mask and its corresponding boundaries.

An explanation for the observed results is, that the very noisy non-refined boundaries make it easy for the discriminator to distinguish the generated pairs from the originals, thereby hampering the learning process of both masks and boundaries, which in turn makes it even easier to differentiate. On the other hand the hourglass can predict high quality boundaries, but fails to learn good masks. This can be because the discriminator concentrates on the boundaries as the easier feature to distinguish in the non-refined layer. As both discriminators share weights, there is less focus on masks in both stages. This behavior was also observed when we tried to only regress boundaries for classes with multiple instance annotations, i.e. humans and cars.

As a result of the unsuccessful experiments we decided to concentrate on semantic masks. Yet, the regression of meaningful boundary masks remains an important problem, as it is essential for using GANs for augmentation of object

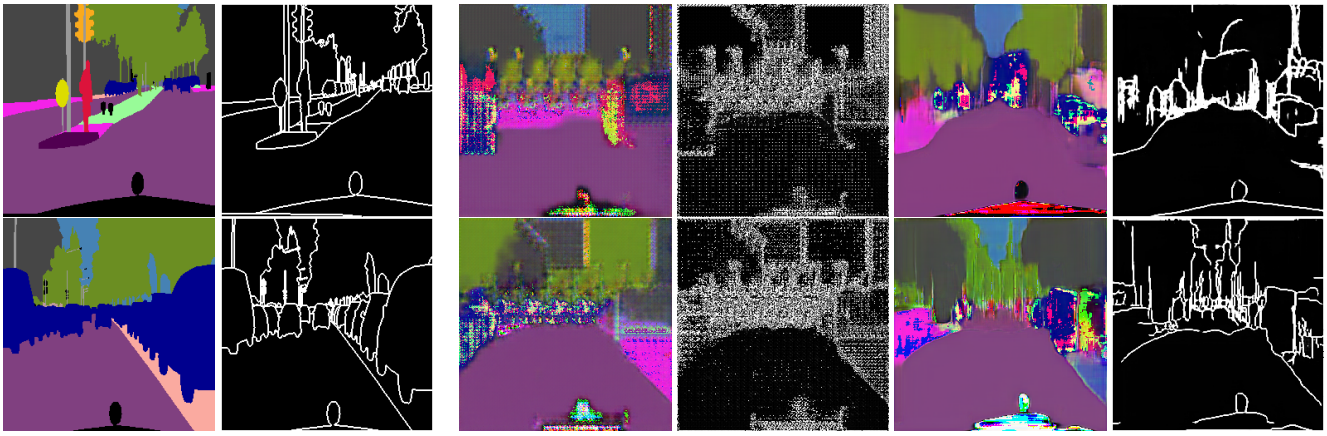

Figure 5: Boundary mask generation with hourglass refinement. **left to right**: groundtruth mask and boundaries, non-refined mask and boundary predictions, refined mask and boundary predictions. While mask and boundaries are of bad quality before the hourglass refinement, the network learns to predict reasonable boundaries, but fails to generate corresponding masks of high quality.

related tasks. A line of research worth investigation in the future would be to have a separate discriminator that optimizes just masks and another one that takes pairs of masks and boundaries as input, but whis loss is *just propagated through the boundary generation layers.* This architecture would ensure, that the discriminator cannot simply ignore the masks because the boundaries are easier to distinguish. Also a curriculum learning scheme, in which masks are trained first and then boundaries are added seems promising, as the latter can benefit from a solid mask estimate.

4.3. Scene Image Generation

In Figure 6 we can see some example outputs of the scene generator network ImageGAN. The network was trained for 160 epochs on the Cityscapes dataset, with a batch size of 10 and using the feature matching loss. The results of the network are very good, if we also take into account the difficulty of the task, i.e. scene generation from semantic segmentation masks. The architecture of the generator is in the Tab. 2, and the discriminator architecture is the same as for mask discriminator.

In Figure 7 we can see the discriminator and generator loss functions at different times during training. Compared to the results in Figure 4 we can see that the training is more stable than the training of MaskGAN, and the results are also of better quality. This is expected up to a certain point, because it is easier to train efficiently a Conditional GAN compared to an Unconditional which tries to generate new samples from a distribution using only random noise as input.

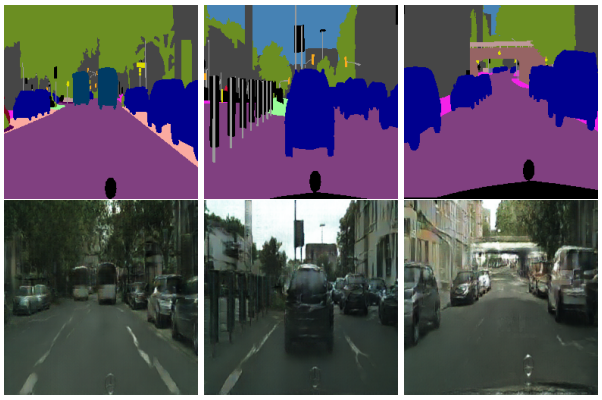

Figure 6: Scene generation using ImageGAN. **top:** input semantic segmentation masks. **bottom:** generated scenes using the input masks

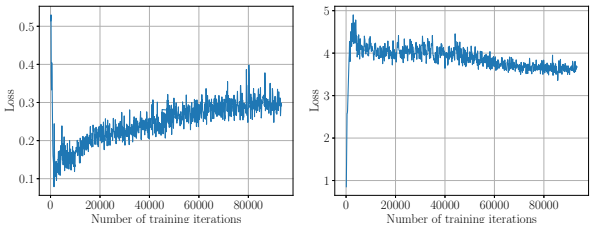

Figure 7: Training losses over number of iterations for ImageGAN. **left**: discriminator loss, **right**: generator loss

4.4. End-to-end Training

In a first experiment we finetuned the pre-trained network stages in an end-to-end fashion. We tried two different setups: supervising only the final scene image prediction or

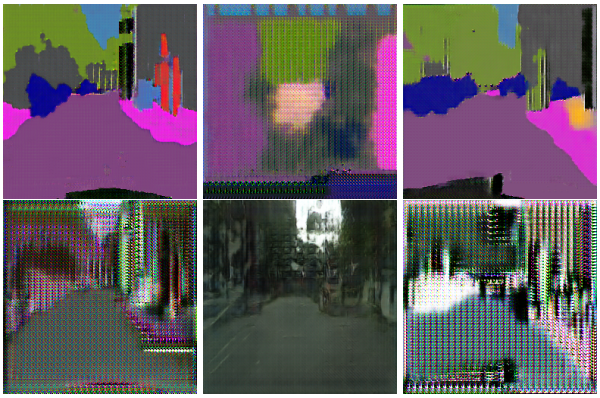

Figure 8: End-to-end finetuning. **top**: generated segmentation masks, **bottom**: generated scene images, **left to right**: without finetuning, finetuned without mask-level GAN loss, finetuned with mask-level GAN loss.

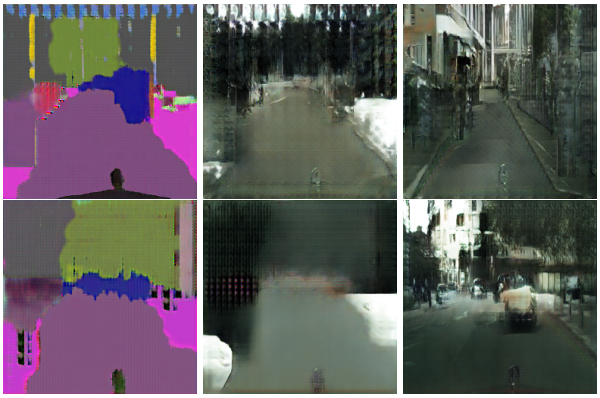

Figure 9: End-to-end training from scratch. **left to right**: predicted segmentation mask, image generated from predicted segmentation, image generated from (different) groundtruth segmentation.

adding an intermediate GAN loss on the masks. The results of both training runs are depicted in Figure 8.

It can be seen, that without finetuning, the scene image generator amplifies the noise in the mask predictions, because it was never exposed to noisy masks before. When training without intermediate supervision, the ImageGAN stage learns to reduce the noise by largely ignoring the mask input and hallucinating street scenes. On the other hand, with intermediate supervision, the quality of the masks is not degraded, but the noise in the output is also not decreased. It can be concluded, that both pre-trained networks are stuck in their local optima and the finetuning process is not able to move both to a joint optimum, that ensures high quality in both outputs.

As a result we decided to train the pipeline end-to-end from scratch, so that both outputs get jointly optimized from the beginning. We employ the architecture detailed in Section 3.4. Results of the training at different stages are depicted in Figure 9. It can be seen, that both masks and scene images get optimized jointly, even though both do not reach the quality of the individually trained stages. The mask predictions are reasonable across training iterations. However, there is a clear trade-off between the performance of the scene image generator on generated masks (middle column) and real segmentations (right column). While in the first row we can see improved performance on the generated masks, the quality of the image generated from a real mask is lower than in the bottom row.

The joint optimization of both objectives makes the training more unstable and therefore needs more careful tuning of the relative weights of the loss components. While we were not able to optimize the training procedure due to time limitations, the end-to-end training from scratch seems to be the most promising approach for generating masks and scene images jointly and should therefore be explored in future research. A first focus can be to continually lower the weight of the loss on the image generated from the real mask, so that the network is forced to concentrate on producing high quality scenes from generated masks.

5. Conclusion

In this work we proposed an architecture for the generation of pairs of semantic segmentation masks and scene images from random noise. We demonstrated the successful generation of segmentation masks using a GAN network and further increased the variability in the output masks by enforcing a mutual information loss between the latent input variables and the generated segmentations. We also proposed ways to predict corresponding boundary masks to enable the usage of our method in the context of augmentation for object detection and showed first results. We further extended the pix2pix architecture of [11] by introducing intermediate feature losses and showed the generation of scene images from segmentation masks. Finally, we proposed multiple architectures to train our pipeline end-to-end, either by finetuning the individual stages or by training from scratch.

While this work could establish a foundation for the generation of augmentation data for semantic segmentation from noise, future work should concentrate on optimizing the training strategies for an end-to-end training of the full pipeline, in order to enable the joint generation of high quality segmentation masks and scene images. Additional attention should be given to the generation of boundary masks to generate augmentation data for instance-related tasks.

References

[1] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[4] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] 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.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[9] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*, 2016.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[11] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[12] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. 2017.

[13] 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.

[14] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[15] 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.

[16] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[17] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, abs/1612.07828, 2016.

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

[19] L. Sixt, B. Wild, and T. Landgraf. Rendergan: Generating realistic labeled data. *CoRR*, abs/1611.01331, 2016.

[20] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *arXiv preprint arXiv:1711.11585*, 2017.

[21] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. 2017.

A. Architectures

Input: 100 noise + 20 one-hot
1024 f.c. $4 \times 4 \times 1024$ fc
$4 \times 4 \times 512$ deconv. stride 2 $4 \times 4 \times 256$ deconv. stride 2 $4 \times 4 \times 128$ deconv. stride 2 $4 \times 4 \times 64$ deconv. stride 2 $4 \times 4 \times 32$ deconv. stride 2
$1 \times 1 \times 3$ conv.
Output: $256 \times 256 \times 3$ masks in RGB

Table 1: Mask Generator.

Input: $256 \times 256 \times 3$ masks in RGB
$4 \times 4 \times 64$ conv. $4 \times 4 \times 128$ conv. stride 2 $4 \times 4 \times 256$ conv. stride 2 $4 \times 4 \times 512$ conv. stride 2 $4 \times 4 \times 512$ conv. stride 2 $4 \times 4 \times 512$ conv. stride 2 $4 \times 4 \times 512$ conv. stride 2 $4 \times 4 \times 512$ conv. stride 2
$4 \times 4 \times 512$ deconv. stride 2 $4 \times 4 \times 512$ deconv. stride 2 $4 \times 4 \times 512$ deconv. stride 2 $4 \times 4 \times 512$ deconv. stride 2 $4 \times 4 \times 256$ deconv. stride 2 $4 \times 4 \times 128$ deconv. stride 2 $4 \times 4 \times 64$ deconv. stride 2 $4 \times 4 \times 3$ deconv. stride 2
Output: $256 \times 256 \times 3$ images in LAB

Table 2: Image Generator.

Input: $256 \times 256 \times 3$ images in RGB
$4 \times 4 \times 64$ conv. $4 \times 4 \times 128$ conv. stride 2 $4 \times 4 \times 256$ conv. stride 2 $4 \times 4 \times 512$ conv. $4 \times 4 \times 1$ conv.

Table 3: Discriminator.