24^{th} *Computer Vision Winter Workshop*
Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)
Stift Vorau, Austria, February 6–8, 2019
DOI: 10.3217/978-3-85125-652-9-03

A Spatiotemporal Generative Adversarial Network to Generate Human Action Videos

Stefan Ainetter, Axel Pinz
Graz Univerisity of Technology
stefan.ainetter@student.tugraz.at,
axel.pinz@tugraz.at

Abstract. *We propose a method to generate high resolution human action videos, by extending a 2D generator network to the spatiotemporal domain. Our generative model consists of a fully convolutional 3D generator, combined with a domain specific video classifier. By using activation maximization in the spatiotemporal domain, we are able to generate action videos at a spatial resolution of* $227 \times 227px$*. Our model, evaluated on the UCF-101 dataset, achieves a state-of-the-art Inception Score of 23.44. Additionally, we improve the accuracy of a video classifier using our videos for data augmentation, which proves high quality and variance of our generated videos.*

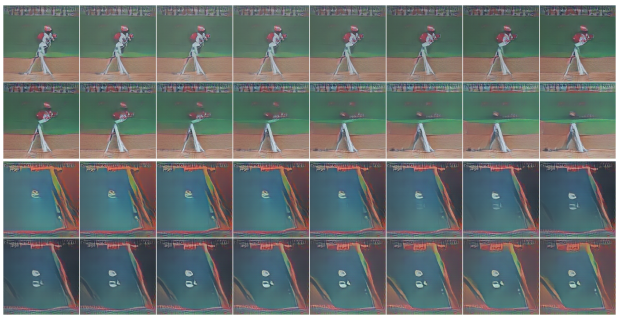

Figure 1. Visualization of generated videos using our $STGAN$. Each video consists of 16 consecutive frames. Results are shown for the UCF-101 classes **Baseball Pitch** (top) and **Billiard** (bottom).

1. Introduction

Introduced in 2014, Generative Adversarial Networks (GANs) [13] have been continuously researched due to their ability to learn perceptual representations of images in an unsupervised manner. Especially in computer vision, GANs can be beneficial for a variety of tasks like clustering, classification, and sample generation. Recent research on GANs for image generation has led to methods which can synthesize images up to a spatial resolution of 2 megapixels [4, 42]. However, video generation lags behind, with current resolutions of, e.g. 64×64 pixels [29], so that further research is needed to extend the use of GANs towards the generation of realistic looking videos.

This work addresses the generation of videos in the domain of action recognition. Training a model to generate videos of human actions will provide further insight into this domain, and therefore might improve the performance and design of current action recognition classifiers. For this purpose, we persent a spatiotemporal GAN ($STGAN$), which can generate videos in the domain of action recognition at a spatial resolution of $227 \times 227px$. Figure 1 shows two examples of our generated video clips with a duration of 16 frames.

We quantitatively evaluate our results using the Inception Score (IS) [32]. In comparison to other approaches for action video generation, we establish a new state-of-the-art both in terms of IS and spatial resolution. Furthermore, we demonstrate that it is possible to fool the Inception Score metric by generating samples that are optimized to achieve a high score, as claimed by [32]. We show how to use IS to obtain valid results, by strictly separating the video generation process and the evaluation. In addition, we use our generated videos for data augmentation, by training an action recognition classifier with both, real and generated data. This increases the accuracy of the classifier and indicates an overall high image quality and variance in our generated videos.

2. Related Work

Generative models for image generation. The idea of GANs was first proposed in [13], outperforming other generative models like autoencoders [20, 40] and Boltzmann machines [23, 31] according to image quality. Since then, GANs have been a popular field of research, and are used in different domains like image in-painting [17], image-to-image translation [24, 43] and super resolution [22]. GAN training using the loss function proposed in [13] is considered as unstable. Therefore, Dosovitskiy and Brox [7] combined the GAN loss with additional losses in image and feature space to make the training more stable. Their loss function proved to be stable during training and generated realistic looking images. Several attempts [25, 11, 26] used this loss function in combination with activation maximization [45, 9, 46, 33] to generate class specific images.

Another popular approach is the Wasserstein-GAN [2], where the loss function is modeled by the Earth-Mover or Wasserstein-1 distance. Several GANs [28, 14, 32, 3, 1] adapted this loss function to improve quality, stability and variation of the generated images.

GANs have also been used in the context of super-resolution [34, 22] to create high resolution output from low resolution input. Denton et al. [5] proposed a framework that generates images in a coarse-to-fine fashion, combining a conditional GAN model with a Laplacian pyramid representation. Karras et al. [19] were able to generate images at a resolution of $1024 \times 1024px$ using GANs and a coarse-to-fine network structure. For this approach, both the generator and discriminator grow progressively during training.

Video Generation using GANs. Video recognition and classification received a lot of attention in recent past. Therefore, generating videos via generative models is also drawing much attention. TGAN [29] generates videos for action recognition using two generators, where one generator learns to generate images, and the other one models temporal coherence. It produces frames with the spatial resolution of $64 \times 64px$. Vondrick et al. [41] proposed a model to generate videos with a spatiotemporal convolutional architecture that separates foreground and background. Tulyakov et al. [39] proposed a video generator which decomposes motion and content of videos. They use an additional recurrent neural network combined with an image generator, to model a sequence of consecutive frames. Recently, TGANv2 [30] was introduced, which uses subsampling layers in the generator and several sub-discriminators for video generation. Fuchs [11] used a similar network architecture compared to our approach to generate videos of arbitrary length, with the goal to visualize deep driving models. Koltun and Chen [4], and Wang et al. [42] are able to synthesize videos with a spatial resolution of $2048 \times 1024px$, also using progressively growing network architectures. However, they use semantic layouts instead of noise as input, which simplifies the process of generating samples.

Evaluation methods for generative models. Directly comparing the images of two different generative models, in terms of image quality, is hard to achieve. Salimans et al. [32] proposed an evaluation metric called the Inception Score (IS). To achieve a high IS, images should contain meaningful objects and the GAN should generate varied images. This metric is widely used to evaluate the quality of generated images [14] and videos [36, 29, 39, 30]. Another way of GAN evaluation is to measure the variance of the generated data. This can be achieved using structural similarity [44], and was used as evaluation metric in [27, 19]. Another popular method is using the Fréchet Inception distance [15], which is based on the Fréchet distance [8].

3. Generative Adversarial Networks for Image Generation

Our spatiotemporal GAN is based on a 2D generative model provided by [7], which is used for image generation. This section briefly summarizes the important aspects about the spatial network architecture and the loss function for image generation. Note that we did not train the 2D generator on our own, but used a pre-trained version provided by [7], before we extended the architecture to generate videos instead of images.

3.1. Spatial Network Architecture

The 2D generative model consists of a generator and a discriminator, with an additional encoder network used for feature extraction. During training, the extracted features (which are provided by the encoder) serve as input code vector $\mathbf{z}$ for the generator and the discriminator. All code vectors $\mathbf{z}$ belong to a low-dimensional input space, which we denote as

latent space.

Generator. The generator network consists of fully-connected layers, followed by convolutional and up-convolutional layers to map a code vector $\mathbf{z}$ to an image. Upconvolutional layers make it possible to up-sample the respective layer input so that the code vector $\mathbf{z}$ with dimension 1×4096 maps to an image of $256 \times 256px$. Rectified Linear Units (ReLUs) are used as nonlinear activation functions in all layers.

Discriminator. The architecture of the discriminator network consists of five convolutional layers, followed by a pooling layer to extract features of the input images. These features are concatenated with a code vector $\mathbf{z}$, which is extracted from real images by the encoder network. Dropout [37] is performed on the resulting feature vector before it is further processed by two fully-connected layers and then classified as real or fake image.

Encoder network. The network architecture of the encoder is AlexNet [21] pre-trained on ImageNet. During training, the encoder network is used as auxiliary network to extract features from real and generated images. These features are then used for several purposes: 1) During training, the generator uses features from real images as input vector, rather than randomly generated ones. This helps to identify compact regions in the latent space which correspond to natural images. 2) The discriminator gets a copy of the same features, to prevent unbalanced training. 3) The extracted features of the encoder are used to calculate the loss function.

3.2. Loss Function for Image Generation

Dosovitskiy and Brox [7] proposed a generator loss function which minimizes distances in image and feature space, additionally to the adversarial loss. This boosts the invariance with respect to irrelevant transformations, and the sensitivity to local image statistics, according to [7]. The composition of these three losses leads to better results in image quality, because the additional feature loss reflects the perceptual similarity of images. The loss in image space helps to stabilize the training process, as adversarial training is known to be unstable and sensitive to hyperparameter selection [13].

The composite loss function $\mathcal{L}$ according to [7] is defined as

$$\mathcal{L} = \lambda_{img}\mathcal{L}_{img} + \lambda_{feat}\mathcal{L}_{feat} + \lambda_{adv}\mathcal{L}_{adv}, \quad (1)$$

with the Euclidean loss in image space $\mathcal{L}_{img}$, the Euclidean loss in feature space $\mathcal{L}_{feat}$, and the adversarial loss $\mathcal{L}_{adv}$. All three parts are weighted with a specific parameter λ.

The loss in image space $\mathcal{L}_{img}$ is written as

$$\mathcal{L}_{img} = \sum_{h,w} \|G(\mathbf{z}) - \mathbf{x}\|_2^2, \quad (2)$$

where $G(\mathbf{z})$ and $\mathbf{x}$ are the generated and real images, respectively, with the dimensionality $[H \times W \times 3]$, where H and W define the spatial resolution of the RGB image. The indices h, w are therefore the spatial indices of the images.

The feature loss $\mathcal{L}_{feat}$ is defined as

$$\mathcal{L}_{feat} = \sum_{h,w} \|E(G(\mathbf{z})) - E(\mathbf{x})\|_2^2, \quad (3)$$

using the encoder network E to extract features form real and generated images. The adversarial loss $\mathcal{L}_{adv}$ is defined as

$$\mathcal{L}_{adv} = -\log D(G(\mathbf{z})), \quad (4)$$

where $D(\cdot)$ defines the discriminator function.

Furthermore, it is also necessary to optimize the discriminator, to successfully classify real and fake images. The generator and discriminator are trained concurrently, using

$$\mathcal{L}_{discr} = -[\log(D(\mathbf{x})) + \log(1 - D(G(\mathbf{z})))], \quad (5)$$

as the loss function for the discriminator.

4. Spatiotemporal Network Structure and Parameter Transfer

We extended the network structures described in Subsection 3.1 to generate videos instead of images. The main idea is to convert the filters from spatial to spatiotemporal kernels using the approach proposed in [10]. Then, the parameters of the pre-trained networks provided by [7] are transferred to the spatiotemporal domain. The new spatiotemporal weights $\mathbf{w}_{3D}$ are initialized layerwise, by following these steps:

1. The temporal weights for each layer are defined as $\mathbf{w}_{2D}$ with the dimensions $[W \times H \times C]$, where W and H define the spatial filter size, and C defines the number of channels. In the first step, the dimensions of the spatiotemporal weights $\mathbf{w}_{3D}$ are defined as $[W \times H \times T' \times C]$, where T' describes the temporal filter depth.

Layer	Kernel	Stride	Output Size
G_fc8	/	/	16, 4096
G_fc7	/	/	16, 4096
G_fc6	/	/	16, 4096
reshape layer	/	/	4x4x16, 256
G_upconv5	4x4x3	2x2x1	8x8x16, 256
G_conv5	3x3x3	1x1x1	8x8x16, 512
G_upconv4	4x4x3	2x2x1	16x16x16, 256
G_conv4	3x3x3	1x1x1	16x16x16, 256
G_upconv3	4x4x3	2x2x1	32x32x16, 128
G_conv3	3x3x3	1x1x1	32x32x16, 128
G_upconv2	4x4x3	2x2x1	64x64x16, 64
G_upconv1	4x4x3	2x2x1	128x128x16, 32
G_upconv0	4x4x3	2x2x1	256x256x16, 3

Table 1. Network structure of 3D generator. The spatiotemporal dimensions for kernel, padding and stride are shown in the order [width x height x time]. The output size is written as [width x height x time, # channels]. We applied [1 x 1 x 1] padding at all convolutional and upconvolutional layers. For video generation, the temporal duration was set to $T = 16$ frames.

2. All spatial weights $\mathbf{w}_{2D}$ are then copied to each temporal position t of the weights $\hat{\mathbf{w}}_{3D}$. This step can be written as

$$\hat{\mathbf{w}}_{3D}(t) = \mathbf{w}_{2D}, \forall t \in [1, T']. \tag{6}$$

3. The last step is to divide the spatiotemporal weights $\hat{\mathbf{w}}_{3D}$ by the temporal filter depth T':

$$\mathbf{w}_{3D} = \frac{\hat{\mathbf{w}}_{3D}}{T'}. \tag{7}$$

This step is used to average the weights across time, and therefore serves as temporal smoothing.

We decided to use a temporal filter depth of $T' = 3$ for all convolutional and upconvolutional layers. All biases and weights from the fully-connected layers are directly copied from the 2D to the 3D architecture.

The temporal stride is kept dense throughout all network layers, to ensure that the full duration of the sequence is preserved through the whole network. Table 1 shows a detailed overview of our 3D generator architecture after parameter transfer.

4.1. Generating Videos using Activation Maximization

To generate videos for a specific action, we use our spatiotemporal generator with Activation Maximization (AM) [9, 46, 33] to maximize the probability that a generated video belongs to a specific class. For this purpose, we use a task specific condition network Φ. This condition network is a pre-trained classifier, and delivers output probabilities for each class in the dataset. The goal of this technique is to find a specific code vector that maximizes the activation of a neuron h. Formally, the objective function can be written as

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}} (\Phi_h(G(\mathbf{z})) - \lambda|\mathbf{z}|), \tag{8}$$

with the parameter λ weighing the regularization term. To perform activation maximization in combination with a generative model, we build on the PPGN framework [25], which is an extension of [26]. The experimental setup using our generator in combination with a condition network is shown in Figure 2. In the forward path, our 3D generator produces videos given a random code vector $\mathbf{z}$. These generated videos serve as input for the condition network, which calculates the softmax probability that the video belongs to a specific UCF-101 class. As we want to maximize the activation for a specific output neuron h, we can back-propagate the gradient $\frac{\partial \log \Phi_h(G(\mathbf{z}))}{\partial \mathbf{z}}$ through the condition network and the 3D generator, to update the code vector $\mathbf{z}$. Note that Gaussian noise $\mathcal{N}(0,\ 10^{-17})$ is added to the gradient before it is applied, to boost the variance of the generated videos.
We perform up to 100 iterations of AM, where the learning rate for the update function starts at 1.0 and linearly decays to 0.1. Other AM specific parameters are taken from [25]. Figure 3 shows the iterative process of AM, transforming random code vectors to a specific action video.
The generator serves as a learned natural image prior, which makes it possible to synthesize interpretable videos. Without using the generator as prior, it would not be possible to generate realistic looking samples, because the set of all possible outputs is too vast.

4.2. Condition Networks

A condition network is a trained classifier, which is used to perform activation maximization. Therefore, the generator and the condition network should

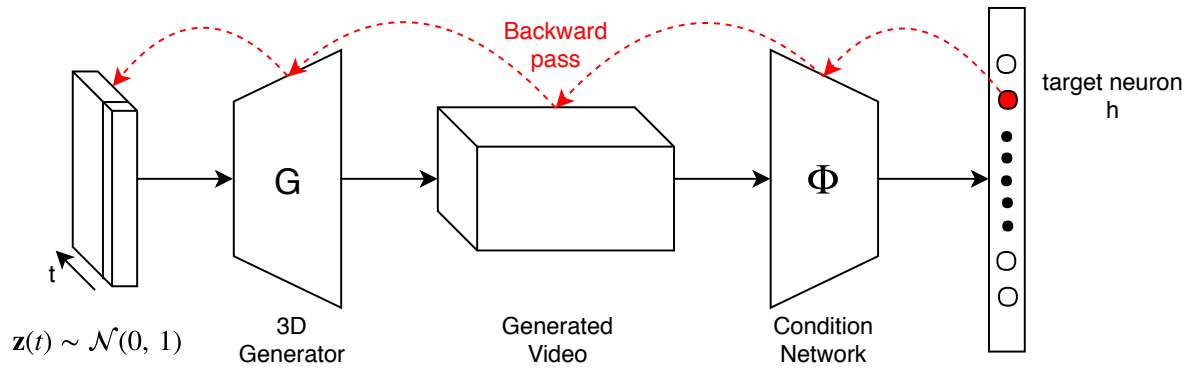

Figure 2. Video generation using activation maximization. The 3D generator G uses a random code vector $\mathbf{z}(t)$, which consists of multiple 1×4096 latent vectors, where each vector corresponds to one frame of the generate video. This video serves as input for a pre-trained condition network Φ. The goal is to maximize the activation for a specific target neuron h, and to back-propagate the gradient through both networks to adjust the code vector $\mathbf{z}(t)$.

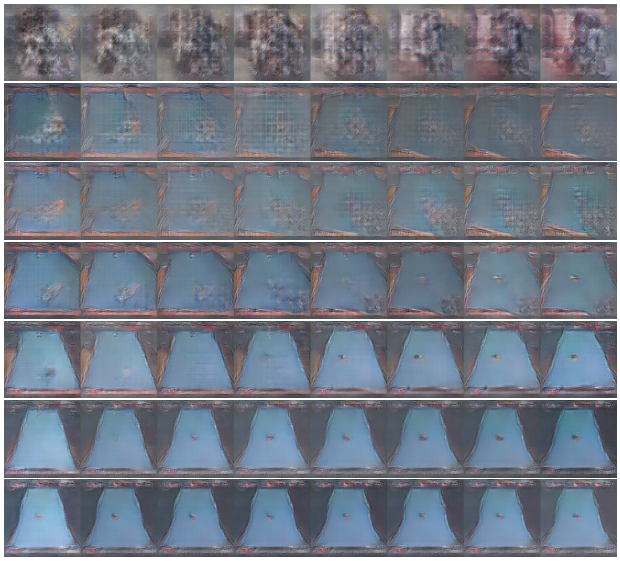

Figure 3. Visualization of activation maximization for class **Billiard**. Top to bottom shows the process of activation maximization applied to 8 frames at different iteration steps. The sequence of images changes from random patterns at top to billiard at bottom. The videos were generated with our spatiotemporal GAN and the LRCN condition network.

operate in a similar domain, using similar datasets for training. As we want to generate videos for action recognition we were looking for classifier networks trained on datasets like UCF-101 [35] and Sports-1M [18]. Therefore, we used LRCN [6] and C3D [38] as condition networks in our experiments. Note that the spatial resolution of our generated videos directly depends on the spatial input dimension of the condition network. Although our 3D generator would be able to generate output videos with $256 \times 256px$, our generated videos after AM are restricted to $227 \times 227px$, which is the spatial resolution for the input of LRCN (which we used in our main experiments).

5. Evaluation

The evaluation of our $STGAN$ is divided into three different parts:

1. Qualitative evaluation of the generated videos with analysis of findings and potential failure cases.
2. Quantitative evaluation using the Inception Score, and comparison to state-of-the-art approaches.
3. Using our generated videos for data augmentation, to increase the performance of an action recognition classifier.

5.1. Qualitative Evaluation

We performed AM for each class in the UCF-101 dataset, where the input for our generator is a sequence of random code vectors, each of them drawn from a Gaussian distribution $\mathcal{N}(0,\ 1)$. We decided to sample videos with a length of 16 frames in all main experiments, to be consistent in our evaluations. Figure 4 shows generated image sequences for several classes, using our $STGAN$.
General findings, according to the quality of generated samples are:
1) One can see that our $STGAN$ does have problems to generate faces correctly, shown in the example for the class **Apply Lipstick** (Figure 4 top). This is because our $STGAN$ is not able to learn the correct number of face parts, like eyes and nose. Similar

problems occur for all classes which focus on human faces. The reason for this problem could be the usage of max-pooling at some stage of the convolutional neural network, which makes the representation invariant to small translations of the input. This means, that the network only focuses on a feature being absent or present, but not on how many times it occurs [12].

2) Videos generated for the class **Billiard** (Figure 4 middle) show results with high image quality. This means, that the quality for each individual frame is high, and the whole video shows a reasonable sequence of consecutive frames which represent a specific action. According to temporal coherence, it seems that the combination of weight transfer to 3D and a condition network pre-trained on video action recognition are enough to generate meaningful videos. Our $STGAN$, which was never trained on video data, seems to inherit the information of temporal coherence from the condition network (which was trained on action videos) during AM.

3) Because we use AM, the generated videos indicate what is most important for the condition network to differentiate between different classes. For the class **Clean and Jerk** (Figure 4 bottom), the network clearly focuses on the weights. It seems that the human body is not important for classifying this action, and therefore the frames contain no human body. This insight could possibly be used to further understand how convolutional neural networks work, and could therefore help to improve the quality of action recognition classifiers.

5.2. Quantitative Evaluation

We quantitatively evaluated our $STGAN$ using the Inception Score (IS). We compare our results to other state-of-the-art GANs, which also operate in the domain of video generation for action recognition.

Evaluation procedure. In order to compute the IS, we generated 10000 videos, which were randomly sampled from a uniform distribution over all UCF-101 classes. Afterwards, we used the C3D action recognition classifier provided by [16] to calculate the IS. The resolution and the number of frames for a C3D input video are $112 \times 112px$ and 16, respectively. Therefore, we resized our generated samples to match the spatial dimension. Additionally, we calculate the IS for the whole UCF-101 dataset. This serves as a key score indicating the IS for real videos. Table 2 shows the IS of our approaches compared to other state-of-the-art GANs for video generation. Note that the evaluation procedure of all approaches in Table 2 is the same, which enables us to directly compare the methods with each other. The IS of the other approaches is provided by the authors in their papers.

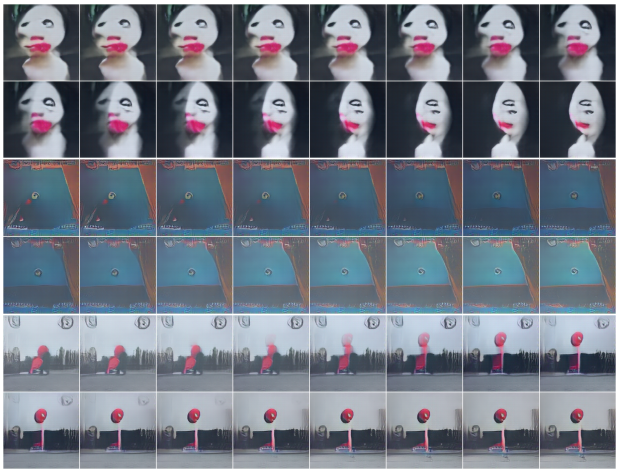

Figure 4. Visualization of generated videos using $STGAN$ in combination with LRCN as condition network. Each video consists of 16 consecutive frames. Results are shown for the classes **Apply Lipstick**, **Billiard** and **Clean and Jerk** (top to bottom).

Method	Inception Score
MoCoGAN [39]	12.42
Conditional TGAN [29]	15.83
TGANv2 ($bs = 64$) [30][1]	22.70
TGANv2 ($bs = 256$) [30][1]	24.34
STGAN	**23.44**
STGAN (C3D)	66.33
UCF101 dataset	70.07

Table 2. IS for different models trained on UCF-101. State-of-the-art approaches (MoCoGAN, TGAN and TGANv2) are compared to our $STGAN$. Also, the IS for the UCF-101 dataset is stated. Our IS, achieved by using different networks for conditioning and classifying, is written in **boldface**. Please note that we outperformed all published competitors [39, 29] by a large margin[1].

Our STGAN achieves state-of-the-art Inception Scores. Using LRCN as condition network and the C3D network for evaluation, we achieve an IS of 23.44, which outperforms several other approaches and is comparable to TGANv2 [30][1]. It is also worth mentioning that our $STGAN$ gener-

[1]During the preparation of our paper, we note the appearance of [30] on arXiv. This is, to our knowledge, the only work presenting results comparable to ours.

Figure 5. Visualization of generated videos using $STGAN$ and C3D as condition network. Each video consists of 16 consecutive frames. Results are shown for the classes **Baseball Pitch** (top) and **Clean and Jerk** (bottom). Although these generated samples achieve a high IS (see Table 2), the videos to not contain any real action scenarios. Therefore, we label them as adversarial examples, which fooled the classifier used for the IS calculation.

ates videos with the highest spatial resolution, compared to the approaches in Table 2. Note that the high IS of TGANv2 comes with high computational cost. Our $STGAN$ runs on a single 12Gb GPU, whereas TGANv2 needs four and 16 GPUs (12Gb) to run their model with a batch size of 64 and 256, respectively (corresponds to the IS of TGANv2 ($bs = 64$) and TGANv2 ($bs = 256$) in Table 2). Therefore, our model has a clear advantage in terms of computational cost and memory consumption.

Fooling the Inception Score is possible. If the same classifier is used as condition network for AM **and** to calculate the IS, the IS can be fooled, because the generated samples are in fact generated to have a high probability with a certain condition network. We proved this by using the C3D network for sample generation and calculation of the IS (corresponds to $STGAN(C3D)$ in Table 2). The IS for this setup is 66.33, which would indicate samples that are comparable to real data, in terms of image quality and variance. However, Figure 5 shows examples using $STGAN(C3D)$ for video generation. By analyzing these samples, one can see that the videos do not contain any real objects, indicating that these are adversarial examples. Salimans et al. [32] stated that directly optimizing the IS would lead to adversarial examples, but did not provide results to prove this claim. Our experiments prove this claim, and show that it is easily possible to fool the IS.

5.3. Supervised Learning using Generated Videos

Additional to the previous evaluations, we used our generated videos for data augmentation, to train an action recognition classifier. We used a C3D implementation [16] as classifier, and the UCF-101 dataset for training and evaluation.

5.3.1 Dataset Preprocessing

Our $STGAN$ enables us to generate videos for all 101 classes of the UCF-101 dataset. However, not all generated samples have the same image quality, due to the variance in the generated data. Therefore, we analyzed the results of our Inception Score calculation, and chose samples for data augmentation that contributed to a high score. Furthermore, we limited the number of samples per class to 100, to prevent unbalancing the training data too much. Figure 6 shows a comparison of the UCF-101 train split 1 and our generated samples.

5.3.2 Training Schedule

We used the pre-trained C3D model, which was trained on the Sports-1M dataset as initialization for our network parameters. The pre-trained model is available online[2]. Other training parameters were taken from the original code [16]. We fine-tuned the classifier on UCF-101, using our generated videos, the UCF-101 train split 1, and a combination of both. For each experiment, the training data were randomly shuffled at the beginning of training. After 200 iterations of training, using a batch size of 10, we evaluated the performance of the classifier by calculating the random clip accuracy on the UCF-101 test split 1. This helped to determine the optimal point to stop training and avoided overfitting.

5.3.3 Classification Results

Table 3 provides the accuracy of our trained classifiers. If we use our generated samples without any real data for training, we achieve an accuracy of 15.6%. This is significantly lower than the accuracy for train split 1, which is 50.6% in our experiments. However, this experiment indicates that our generated samples provide meaningful information about the UCF-101 dataset. Using both, the real and generated data for training the classifier, we achieve an accuracy of 52.2%. This improvement shows that there is some additional information in the augmented data, although the generated data do not have

[2]https://github.com/hx173149/C3D-tensorflow, last visited: Dec. 2018

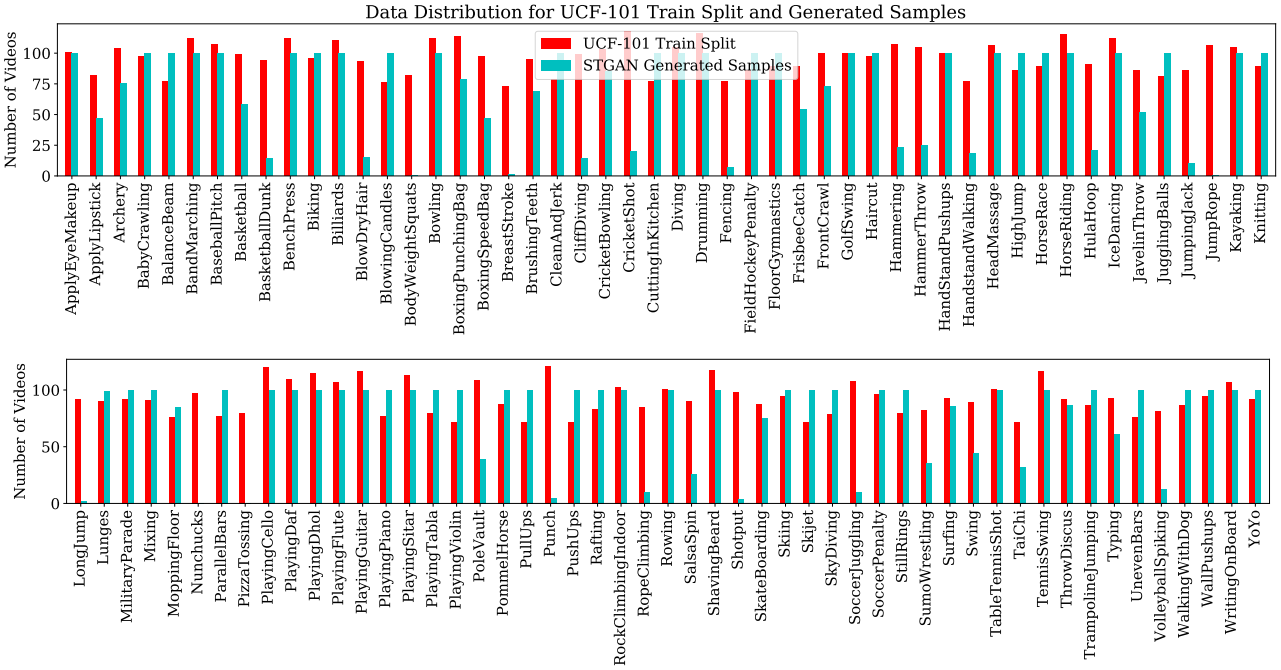

Figure 6. Comparison of data distribution between UCF-101 train split 1 and our generated samples. The generated samples for data augmentation were chosen according to the IS, and the number of samples per class is restricted to at most 100 samples. The data distribution shows that our $STGAN$ is not able to produce high quality samples for certain classes e.g. JumpRope, Nunchucks, Pizza Tossing.

Training Procedure	Accuracy
Generated Samples Only	15,6%
Train Split 1	50,6%
Train Split 1 + Generated Samples	**52,2%**

Table 3. Accuracy of C3D action recognition classifier. The accuracy was calculated for the UCF-101 test split 1, where we randomly extracted 16 consecutive frames from each clip for testing. Training with generated samples leads to a lower score, compared to training with real data. However, training with both, real and generated data leads to an increase of accuracy, which indicates that our generated samples contain valuable additional information for action recognition.

the same image quality as the real ones. Note that the goal of this experiment was to show the practical benefit of our $STGAN$, rather than competing for a state-of-the-art accuracy for UCF-101. This means, that another network architecture, and a more careful selection of hyperparameters, combined with an optimized training schedule, would certainly lead to a higher overall accuracy for all experiments.

6. Conclusion

We have presented a method to generate human action videos for 101 action classes at a spatial resolution of $227 \times 227px$, by extending a generative model proposed for image generation to the spatiotemporal domain. A condition network performs activation maximization, where our generator is used as image prior. In terms of image quality measured by the Inception Score, our approach outperforms all published state-of-the-art methods by a large margin, and also improves with respect to spatial resolution of the generated videos.

Our evaluations also demonstrate a major weakness of the Inception Score metric: We were able to generate adversarial examples, which contained no meaningful objects, but achieved an exceptionally high Inception Score. This means that (a) adversarial examples deserve in-depth research in the future, and (b) Inception Score should be reconsidered.

Action videos generated by our approach can readily be used for data augmentation, in addition to real training data. This leads to an increased accuracy of the classifier, which underlines the high quality and variance of our videos, and highlights direct benefits towards improved recognition.

References

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial net-

works. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 214–223, 2017. 2

[3] D. Berthelot, T. Schumm, and L. Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 2

[4] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1529, 2017. 1, 2

[5] 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 (NIPS)*, pages 1486–1494, 2015. 2

[6] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015. 5

[7] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 658–666, 2016. 2, 3

[8] D. Dowson and B. Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982. 2

[9] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical report, University of Montreal, 2009. 2, 4

[10] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3468–3476, 2016. 3

[11] H. Fuchs. Visualizing and Understanding Deep Driving Models. Masters Thesis at Institute of Electrical Measurement and Measurement Signal Processing, Graz University of Technology, 2017. 2

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 6

[13] 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 (NIPS)*, pages 2672–2680, 2014. 1, 2, 3

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5767–5777, 2017. 2

[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6626–6637, 2017. 2

[16] HouXin. C3D-tensorflow. `https://github.com/hx173149/C3D-tensorflow`, 2018. last visited: Sep 2018. 6, 7

[17] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and Locally Consistent Image Completion. *ACM Transactions on Graphics (TOG)*, 36(4):107:1–107:14, 2017. 2

[18] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014. 5

[19] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 2

[21] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 3

[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017. 2

[23] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 609–616, 2009. 2

[24] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 700–708, 2017. 2

[25] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3510–3520, 2017. 2, 4

[26] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3387–3395, 2016. 2, 4

[27] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2642–2651, 2017. 2

[28] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016. 2

[29] M. Saito, E. Matsumoto, and S. Saito. Temporal generative adversarial nets with singular value clipping. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2849–2858, 2017. 1, 2, 6

[30] M. Saito and S. Saito. TGANv2: Efficient training of large models for video generation with multiple subsampling layers. *arXiv preprint arXiv:1811.09245*, 2018. 2, 6

[31] R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 693–700, 2010. 2

[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2234–2242, 2016. 1, 2, 7

[33] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *International Conference on Learning Representations (ICLR)*, 2014. 2, 4

[34] C. K. Soenderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised map inference for image super-resolution. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[35] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. Technical report, Center for Research in Computer Vision (CRCV), November 2012. 5

[36] C. Spampinato, S. Palazzo, P. D'Oro, F. Murabito, D. Giordano, and M. Shah. Vos-gan: Adversarial learning of visual-temporal dynamics for unsupervised dense prediction in videos. *arXiv preprint arXiv:1803.09092*, 2018. 2

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014. 3

[38] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015. 5

[39] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6

[40] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103, 2008. 2

[41] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, pages 613–621, 2016. 2

[42] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 1, 2

[43] 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. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[44] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, volume 2, pages 1398–1402, 2003. 2

[45] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015. 2

[46] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833, 2014. 2, 4