CAP6610 Machine Learning Project Report

Michael Francis Pérez
University of Florida
Computer and Information Science and Engineering Department
432 Newell Drive Gainesville FL 32611
`michaelperez012@ufl.edu`

Abstract

The paper "Generating Video with Scene Dynamics" (1), which proposes *VideoGAN* for video recognition and generation, was reviewed and implemented. The GAN was first trained for video generation on UCF-101. The generated videos are not photo-realistic, but they do contain realistic motions. Then, the discriminator in the GAN was fine-tuned for video classification in two experiments, for a binary and a multi-label task. The discriminator performed well on the binary classification task. The multi-label classification experiment was conducted but not evaluated due to a data loading error.

1 Introduction

Video generation is an important deep learning task which has applications in simulations and forecasting. (1) Similarly, video recognition (for example, action detection) has several applications in medicine, sociology, crime detection, and human-computer interaction. These tasks are interesting because the generative adversarial network (*GAN*) framework can learn each of these tasks jointly during training. In addition, although the *GAN* can generate images and voices with uncanny realism, the *GAN*'s fidelity for videos not as impressive. Video generation is a challenging task; a lot of work is left to be done before realistic videos can be generated in an unconstrained setting. (2)

2 Problem Statement

A new framework for estimating generative models via an adversarial process was developed in 2014, the *GAN* (3). Two models, a generative model G that captures the data distribution and a discriminative model D that estimates the probability that a sample $\mathbf{x}$ came from the training data rather than G, are trained jointly. G takes as input a noise vector z, sampled from a normal distribution $p_{noise}(z)$, and up-samples it into an image. D outputs a scalar probability that an input image $\mathbf{x}$ is from the real data distribution. D and G play a two-player mini-max game with value function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(\mathbf{x})] + \mathbb{E}_{z \sim p_{noise}(z)}[\log(1 - D(G(\mathbf{z}))] \tag{1}$$

The framework is termed "adversarial" because the generative model competes against an adversary, the discriminative model, that learns to distinguish between samples from the real and generated distributions. Competition causes both models to improve until the generated samples are indistinguishable from samples from the real data distribution, and the generator learns to fool the discriminator. The quality of generated images is assessed by fitting a Gaussian Parzen window to the images and then computing the log-likelihood on the test set. The MNIST (4) dataset contains images of 70,000 28×28 handwritten digits, while CIFAR-10 (5) consists of 60,000 32×32 color images from 10 classes. The log-likelihood estimates on MNIST and CIFAR-10 show competitive results to existing generative models, suggesting the *GAN* framework's viability.

During training, Equation 1 is modified to train G to maximize $D(G(\boldsymbol{z}))$ instead of it being trained to minimize $\log(1 - D(G(\boldsymbol{z}))$. This objective results in the same fixed point dynamics of D and G but provides stronger gradients early in training.

In practice during training of GANs we optimize θ_p rather than p_g. D and G are defined as multilayer perceptrons, which can be concerning because this introduces multiple critical points in the parameter space. However, despite their lack of theoretical guarantees, the stellar performance of multilayer perceptrons suggests they are a decent choice.

To capture motion, videos can naturally be decomposed into a spatial component, which has information about the objects and scenes in a video, and a temporal component, which describes the movement of objects and the camera. (6) A two-stream CNN architecture (6) was used to establish a new state-of-the-art method for action recognition by combining a spatial and temporal recognition stream by late fusion (7). The spatial stream operates on single frames from the input video, while the temporal stream operates on multi-frame optical flow. An optical flow is a set of displacement vector fields between two consecutive frames. The temporal stream input is formed by stacking the optical flow displacement fields between a sequence of consecutive frames. Instead of explicitly calculating optical flow before training, *VideoGAN* (1) learns motion features during training.

Three-dimensional (3D) CNNs and spatio-temporal convolutions were used to achieve a new state-of-the-art in video object recognition and scene classification, entitled Convolution 3D feature (*C3D*) (8). The authors (8) argue that only 3D convolutions preserve the temporal information of the input signals, because 2D convolution collapses the temporal information. The Net A very deep CNN architecture from (9) was adapted by replacing all 2D convolution and pooling operations with their 3D counterparts. Filter kernels of size $3 \times 3 \times 3$ that operate over space and time are used; 16-frame clips are used as input to the network. The UCF-101 dataset (10) contains 13,320 video clips labeled into 101 action classes. The primary evaluation is performed using UCF-101; the authors achieve an 11% improvement over the two-stream approach (6), which can be attributed to *C3D* modeling temporal signals better.

VideoGAN (1) is the first work to extensively investigate *GAN*'s for video. The authors design a one-stream architecture and a two-stream architecture for the generator G. The one-stream architecture uses spatio-temporal convolutions (8) to provide spatial and temporal invariance, and fractionally strided convolutions (11) to up-sample efficiently. This architecture is a variant of *DCGAN* (12) that is extended in time. The two-stream generator architecture models a static background and moving foreground according this expression:

$$G_2(z) = m(z) \odot f(z) + (1 - m(z)) \odot b(z), \tag{2}$$

where $0 \leq m(z) \leq 1$ is a mask that selects either the foreground $f(z)$ or the background $b(z)$ at each pixel and time step, and $\odot$ is element-wise multiplication. $f(z)$ is the same network as the one-stream architecture, $b(z)$ is similar to the generator in *DCGAN* (12), and $m(z)$ shares weights with $f(z)$ except for the last layer, which has one output channel. The generator outputs 64×64 pixel videos up to 32 frames long ($\sim$1 second). The discriminator network is a five-layer spatio-temporal CNN with kernels of size $4 \times 4 \times 4$. The discriminator architecture uses strided convolutions instead of fractionally strided convolutions in order to down-sample the image and the last layer outputs a binary classification (real or fake).

3 Algorithm

GANs are usually trained using mini-batch stochastic gradient descent (Algorithm 1). In this algorithm k is a hyperparameter for the number of training steps to apply to D at each iteration.

(3) shows that the optimal discriminator for any given generator G is

$$D^*_G(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}, \tag{3}$$

```
for number of training iterations do
    Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(1)}\}$ from noise prior $p_g(\boldsymbol{z})$
    Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(1)}\}$ from data generating distribution $p_{data}(\boldsymbol{x})$
    Update discriminator $D$ by ascending its stochastic gradient:
    $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(\boldsymbol{x}^{(i)}) + \log(1 - D(G(\boldsymbol{z}^{(i)})))]$
    for k steps do
        Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(1)}\}$ from noise prior $p_g(\boldsymbol{z})$
        Update generator $G$ by descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(\boldsymbol{z}^{(i)})))]$
    end
end
```

Algorithm 1: Mini-batch Stochastic Gradient Descent

D's training objective can be understood as maximizing the log-likelihood for the conditional probability of $P(Y = y|x)$, where Y indicates whether $\boldsymbol{x}$ came from p_{data} $(y = 1)$ or p_g $(y = 0)$.

The objective in equation 1 can be rewritten:

$$C(G) = \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{z \sim p_{noise}(z)}[\log(1 - D_G^*(G(\boldsymbol{z}))] \quad (4)$$

$= \mathbb{E}_{x \sim p_{data}(x)}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{z \sim p_{noise}(z)}[\log(1 - D_G^*(\boldsymbol{x})]$

$= \mathbb{E}_{x \sim p_{data}(x)}[\log \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}] + \mathbb{E}_{z \sim p_{noise}(z)}[\log \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}]$

(3) shows that the global minimum of $C(G)$ is attained if and only if $p_g = p_{data}$. For an ideal discriminator (3) the objective $C(G)$ is equivalent to the Jensen-Shannon divergence between the distributions p_g and p_{data}.

This algorithm is not guaranteed to converge, but empirically it works well. (3) proves that p_g converges to p_{data} if D and G have enough capacity and if D is allowed to reach its optimum given G at each training step of Algorithm 1.

In *VideoGAN*, the discriminator and generator are trained via stochastic gradient descent. The *ADAM* optimizer (13) is used with a fixed learning rate of 0.0002 and momentum of 0.5. The latent code $z \in \mathbb{R}^{100}$ is sampled from a normal distribution, and a batch size of 64 is used. After every layer in the generator other than the output layer, there is a batch normalization layer (14) and then a ReLU activation function. In the discriminator, batch normalization is used with leaky ReLU (15). This training procedure took several days to train on one GPU.

In one experiment, the authors (1) evaluated *VideoGAN*'s performance classifying actions on UCF-101. 5,000 hours of unlabeled Flickr videos (16) are pre-processed, through stabilization of camera motion using SIFT and RANSAC, and normalization. The model is trained on the large unlabeled dataset; then, the discriminator is fine-tuned on a relatively small set of labeled videos. The discriminator's output function is modified to be a K-way softmax classifier instead of a binary classifier, and dropout (17) is used as the second-to-last layer to reduce overfitting. This fine-tuning procedure yields a classification accuracy of 52.1% on UCF-101.

4 Experiments

Three experiments were conducted to evaluate *VideoGAN*'s performance in video generation and classification. The video generation performance was qualitatively evaluated after training the GAN on a dataset of golf scenes (1). Generation results are presented in Section 4.1.1. Then, the pre-trained GAN is fine-tuned on two subsets of UCF-101 videos, a binary classification task and a multi-label classification task. Classification accuracy for these experiments is reported in Sections 4.2.1 and 4.3.1, respectively.

4.1 Video Generation

A preprocessed dataset of one-second long videos of golf scenes was downloaded from `http://data.csail.mit.edu/videogan/golf.tar.bz2`. The camera motion in this dataset is already stabilized: frames are warped based on SIFT keypoints and the estimated homography between frames using RANSAC. Videos are 32 frames long (~ 1 second), have a frame rate of 25 frames per second, and have a resolution of $64x64$ pixels. In this dataset the videos are already converted into an image of vertically concatenated frames to decrease the computational cost of loading data.

VideoGAN was first implemented in Torch7 (`https://github.com/cvondrick/videogan`). A GitHub repository containing a PyTorch implementation of VideoGAN was cloned and heavily modified (`https://github.com/yhjo09/videogan-pytorch`). The model was trained from scratch using the same architecture and most of the same hyperparameters detailed in the *VideoGAN* paper (1). The Adam optimizer (13) is used with a fixed learning rate of 0.0002 and momentum of 0.5. Training was conducted using the *HiPerGator* supercomputer at the University of Florida. The GAN is trained for video generation for 10 epochs and with a batch size of 20. In the *VideoGAN* paper the model is trained for 100 epochs with a batch size of 100. Training took about five days using one NVIDIA A100 GPU.

4.1.1 Results and Analysis

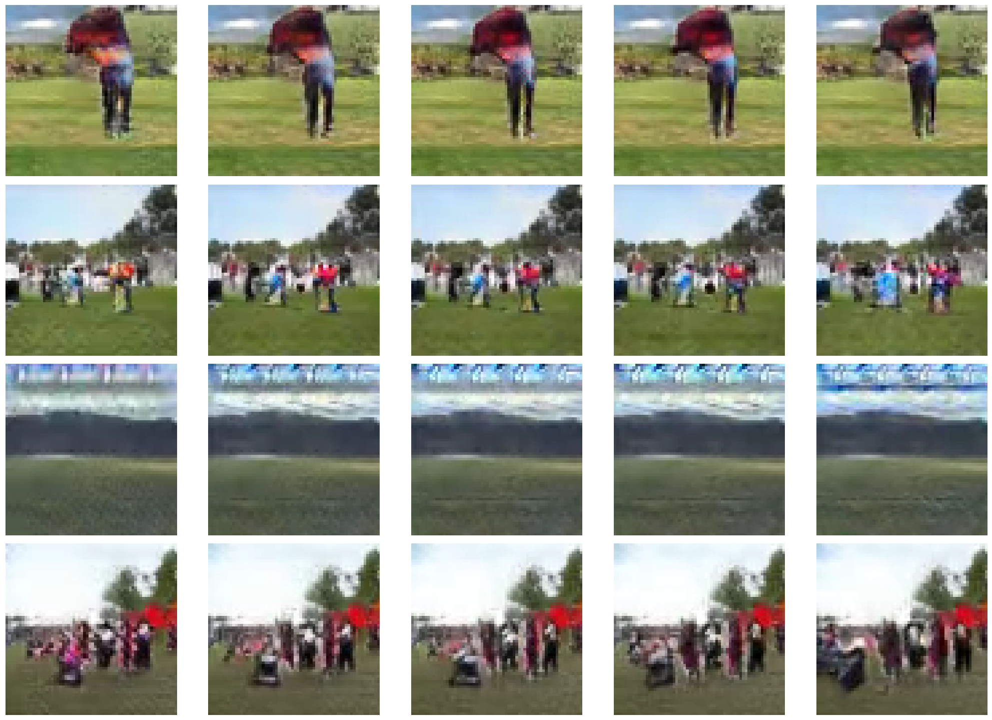

Figure 1: Golf Course Generated Videos. From left to right, frames 1, 8, 16, 24, and 32, from 4 generated videos

Figure 1 depicts five frames from four different generated videos. The generated videos are fairly sharp and the motion patterns correspond to a golf scene. Most generated scenes depict people walking on grass. The videos are low resolution; objects and people look like blobs. Nevertheless, short motion is generated and modeled. The scenes appear to have a static background and moving foreground.

4.2 Binary Video Classification

After training the GAN for video generation, the discriminator was fine-tuned for video classification. The fine-tuning procedure described in the paper was used. I first investigated whether the discriminator can learn a binary classification task, differentiating between YoYo and Kayaking videos. The UCF-101 dataset was downloaded and partitioned by creating folders to organize videos by their action class. This was necessary to use PyTorch's built-in data loading class because it assumes this directory structure.

The state dictionaries of the discriminator and its optimizer were loaded. During unsupervised pretraining, the discriminator was trained to output a scalar $[0, 1]$ representing the probability that a training sample came from the training data instead of the generator. The last layer was changed to output two nodes instead of one, to represent the two action classes. The discriminator was then fine-tuned on a training set of 249 YoYo and Kayaking videos for 24 epochs with a batch size of 10, with labels.

After fine-tuning, the discriminator's classification performance is evaluated by testing it on a test set of 20 YoYo and Kayaking videos. The accuracy is printed afterwards and the state dictionaries are saved.

4.2.1 Results and Analysis

This experiment was performed 10 different times and the accuracy each run was between 0.9 to 1.0. These results show that after fine-tuning, the discriminator is a useful video classifier. As the number of classes increase, the prediction accuracy should decrease. Although the $VideoGAN$ paper does not perform this experiment, this high accuracy for a binary classification task makes sense in the context of the reported results. A 52.1% accuracy was reported for a multi-label classification task in the $VideoGAN$ paper. A dropout layer was not added because the classification results did not show signs of overfitting.

4.3 Multi-label Video Classification

I then investigated whether the discriminator can learn a multi-label classification task and classify the 101 action classes of UCF-101. The full UCF-101 dataset was downloaded and partitioned.

The state dictionaries of the discriminator and its optimizer were loaded The last layer is changed to output 101 nodes instead of one to represent the 101 action classes. The discriminator is then fine-tuned on a training set of $9,537$ UCF-101 videos for 24 epochs with a batch size of 10, with labels.

After fine-tuning, the discriminator's classification performance is evaluated by testing it on a test set of $3,783$ UCF-101 videos. The accuracy is printed afterwards and the state dictionaries are saved.

4.3.1 Results and Analysis

My goal was to reproduce the result from the *VideoGAN* paper (52.1% accuracy). The data was preprocessed and the fine-tuning algorithm was implemented. When fine-tuning the model I am experiencing an error related to the dimensions of the UCF-101 videos. This prevented me from evaluating the classification performance on the test set.

5 Conclusion

My results agree with the generation and classification and results from the *VideoGAN* paper (1). The generated videos are slightly less realistic than the frames presented in the paper. This can be attributed to training for fewer iterations and with a smaller batch size. The classification results achieved for the binary task are stellar.

The static background and moving foreground assumption in *VideoGAN* is a limitation because the model cannot generate videos with a moving background or static foreground. Later works (*Temporal GAN*) (18) removed this assumption and improved the generation results.

There are several open research questions related to this framework. Quantitatively evaluating the fidelity of generated videos is difficult. *VideoGAN* did so manually by having workers on Amazon Mechanical Turk compare the realism of different videos. Subsequent video generation methods (18) extended the Inception score metric commonly used to evaluate the fidelity of generated images, to videos. To calculate this metric a 3D CNN called *C3D* (8) that is pretrained on UCF-101 classifies videos in the dataset. Then the marginal distribution is compared with the conditional label distribution using the Kullback-Leibler divergence formula. This metric encourages distinct yet varied generations. A drawback of this metric is that it is limited by features that the 3D CNN detects.

Realistic, unconstrained video generation is a difficult problem which has not been solved yet. State-of-the-art methods (1) (2) generate videos that are longer duration, higher resolution, and more convincing than *VideoGAN*'s videos, yet they are not photo-realistic. Different approaches for video completion and prediction have produced more realistic videos, but these methods are constrained by the input frames, which is a slightly different task.

The classification process performed in *VideoGAN* is semi-supervised. The GAN is first trained unsupervised for video generation, and then the discriminator is fine-tuned with labels for classification. Unsupervised video classification is a difficult yet important problem which has not been solved yet.

References

[1] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *CoRR*, vol. abs/1609.02612, 2016. [Online]. Available: http://arxiv.org/abs/1609.02612

[2] A. Clark, J. Donahue, and K. Simonyan, "Efficient video generation on complex datasets," *CoRR*, vol. abs/1907.06571, 2019. [Online]. Available: http://arxiv.org/abs/1907.06571

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, O. Sherjil, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[4] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[5] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," no. 0, 2009.

[6] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *CoRR*, vol. abs/1406.2199, 2014. [Online]. Available: http://arxiv.org/abs/1406.2199

[7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[8] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3D: generic features for video analysis," *CoRR*, vol. abs/1412.0767, 2014. [Online]. Available: http://arxiv.org/abs/1412.0767

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[10] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012. [Online]. Available: http://arxiv.org/abs/1212.0402

[11] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," pp. 2528–2535, 2010.

[12] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.06434

[13] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations*, 12 2014.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[15] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” *CoRR*, vol. abs/1505.00853, 2015. [Online]. Available: http://arxiv.org/abs/1505.00853

[16] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L. Li, “The new data and new challenges in multimedia research,” *CoRR*, vol. abs/1503.01817, 2015. [Online]. Available: http://arxiv.org/abs/1503.01817

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[18] M. Saito and E. Matsumoto, “Temporal generative adversarial nets,” *CoRR*, vol. abs/1611.06624, 2016. [Online]. Available: http://arxiv.org/abs/1611.06624