Predicting the Future using Multi-stage Generative Adversarial Networks

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Predicting the future from a sequence of video frames is one of the most sought after yet challenging task in the field of computer vision and machine learning. Although there have been efforts for tracking using motion trajectories and flow features, the problem of generating unseen frames has not been studied extensively. In this paper, we aim to deal with this through the use of simple convolutional models inside a multi-stage Generative Adversarial Networks (GAN) framework. The proposed method uses two stages of GANs to generate crisp and clear set of future frames. Although GANs have been used in the past for predicting the future, none of the works consider relation between subsequent frames in the temporal dimension. Our main contribution lies in formulating two objective functions using a simple yet subtle idea of using a very popular technique of template matching - the Normalized Cross Correlation (NCC) and the Pairwise Contrastive Divergence (PCD). This method, when coupled with the traditional L2 loss, provides superior performance on three real world video datasets *viz.* Sports-1M, UCF-101 and the KITTI.

1 Introduction

Video frame prediction has always been one of the fundamental problems in computer vision as it caters to a wide range of applications including self-driving cars, surveilance, robotics and inpainting. However, the challenge lies in the fact that, real-world scenes tend to be complex, and predicting the future events requires modelling of complicated internal representations of the ongoing events. Recently, the work of [10] modeled this problem in the framework of Generative Adversarial Networks (GAN). Generative models, as introduced by Goodfellow *et. al.*, [4] try to generate images from random noise by simultaneously training a generator (G) and a discriminator network (D) in a process similar to a zero-sum game. Mathieu *et. al.* [10] shows the effectiveness of this adversarial training in the domain of frame prediction using a combination of two objective functions (along with the basic adversarial loss) employed on a multi-scale generator network. This idea stems from the fact that the original $L2$-loss tends to produce blurry frames. This was overcame by the use of Gradient Difference Loss (GDL), which showed significant improvement over the past approaches when compared using similarity and sharpness measures. However, this approach, although producing satisfying results for the first few predicted frames, tends to generate blurry results for predictions far away ($\sim$6) in the future.

In this paper, we aim to get over this tendency of producing blurry predictions by taking into account the relation between consecutive frames in the temporal dimension also. We propose two objective functions: (a) **Normalized Cross-Correlation Loss (NCCL)** and (b) **Pairwise Contrastive Divergence Loss (PCDL)** for effectively capturing inter-frame relationships in the GAN framework. NCCL maximizes the cross-correlation between neighbourhood patches from consecutive frames

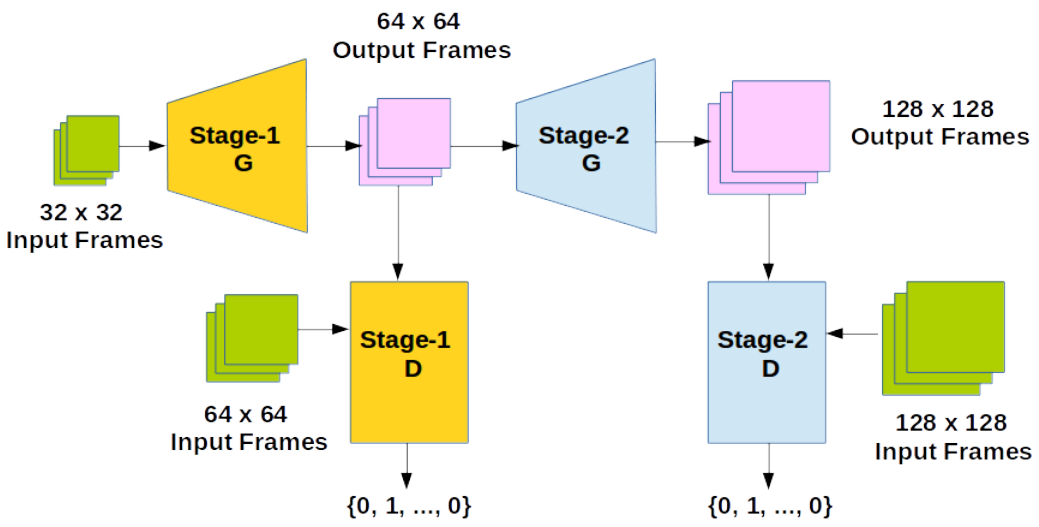

Figure 1: The proposed multi-stage GAN framework. The stage-1 generator network outputs a low-resolution version of predicted frames which are then fed through the stage-2 generator. Discriminators at both the stages predict 0 or 1 per predicted frame to denote its origin: synthetic or original.

whereas, CDL applies a penalty when subsequent generated frames are predicted wrongly by the discriminator network (D), thereby separating them far apart in the feature space.

The rest of the paper is organized as follows: section 2 describes the multi-stage generative adversarial arhitecture used, the sections 3 - 5 introduce the different loss funtions employed: the adversarial loss (AL), $L2$-loss (L2) and most importantly NCCL and CDL. We show the results of our experiments on Sports-1M [8], UCF-101 [13] and KITTI [3] and compare them with state-of-the-art techniques in section 6. Finally, we conclude our paper highlighting the key points and future direction of research in section 7.

2 Multi-stage Generative Adversarial Model

Generative Adversarial Networks (GAN) [4] are composed of two networks: (a) the Generator (G) and (b) the Discriminator (D). The generator G tries to generate realistic images by learning to model the true data distribution p_{data} and thereby trying to make the task of differentiating between original and generated images by the discriminator difficult. The discriminator D, in the other hand, is optimized to distinguish between the synthetic and the real images. In essence, this procedure of alternate learning is similar to the process of two player min-max games. Overall, the GANs try to minimize the following objective function

$$\min_{G} \max_{D} v(D, G) = \mathbb{E}_{x \sim p_{data}}[log(D(x))] + \mathbb{E}_{z \sim p_z}[log(1 - D(G(z)))] \tag{1}$$

where, x is a real image from the true distribution p_{data} and z is vector sampled from the distibution p_z, usually uniform or Gaussian. The adversarial loss emplyed in this paper is slightly different from equation 1, as, the input to our network is a sequence of frames of a video, instead of a noise vector z.

As convolutions account only for short-range relationships, pooling layers are used to garner information from wider range. But, this process generates low resolution images. To overcome this, Mathieu *et. al.* [10] uses a multi-scale generator network , equivalent to the reconstruction process of a Laplacian pyramid [**?**], coupled with discriminator networks to produce high-quality output frames of size 32×32. There are two shortcomings of this approach:

a. Generating image output at higher dimensions *viz.* (128×128) or (256×256), requires multiple use of some statc upsampling operator applied on the output of the generators. In our proposed model, this upsampling is handled by the generator models implicitly through the use of consecutive unpooling operations, thereby generating predicted frames at much higher resolution in lesser number of scales.

b. As the generator network parameters are not learned with respect to any objective function which captures the temporal relationship effectively, the output becomes blurry after ~ 4 frames.

To overcome the first issue, we propose a multi-stage (2-stage in this paper) generative adversarial network.

2.1 Stage-1

Generating the output frame(s) directly often produces blurry outcomes. Instead, we simplify the process by first generating crude, low-resolution version of the frame(s) to be predicted. The stage-1 generator (G_1) consists of a series of convolutional layers coupled with unpooling layers [15] which upsample the frames. We used ReLU non-linearity in all but the last layer, in which case, tanh was used following the scheme of [**?**]. The inputs to G_1 are m number of consecutive frames of dimension $W_0 \times H_0$, whereas the outputs are n predicted frames of size $W_1 \times H_1$, where, $W_1 = W_0 \times 2$ and $H_1 = H_0 \times 2$. These outputs, stacked with the upsampled version of the original input frames, produce the input of dimension $(m+n) \times W_1 \times H_1$ to the stage-1 discriminator (D_1). D_1 applies a chain of convolutional layers followed by multiple fully-connected layers to finally produce an output vector of dimension $(m+n)$, consisting of 0's and 1's.

One of the key difference of our proposed GAN framework is that, the discriminator network produces decision output for multiple frames, instead of a single $0/1$ outcome. This is exploited by one of the proposed objective functions, the CDL, which is described in later sections.

2.2 Stage-2

The second stage network closely resembles the stage-1 architecture, having difference only in the input and output dimensions. The input to the stage-2 generator (G_2) is formed by stacking the predicted frames and the upsampled inputs of G_1, thereby having dimension of $(m+n) \times W_1 \times H_1$. The output of G_2 are n predicted high-resolution frames of size $W_2 \times H_2$, where, $W_2 = W_0 \times 4$ and $H_2 = H_0 \times 4$. The stage-2 discriminator (D_2), works in similar fashion as D_1, producing output vector of length $(m+n)$.

Effectively, the multi-stage model can be represented by the following recursive equations:

$$Y_k = \begin{cases} G_k(Y_{k-1}, X_{k-1}), for \quad k > 2 \\ G_k(X_{k-1}), for \quad k = 1 \end{cases} \tag{2}$$

where, Y_k is the set of predicted frames, X_k are the input frames at the kth stage of the generator network G_k.

2.3 Training the multi-stage GAN

The training procedure of the multi-stage GAN model follows that of the original generative adversarial networks with minor differences. The training of the discriminator and the generator is described as follows:

Training of the discriminator Considering the input to the discriminator (D)as X (series of m frames) and the target output to be Y (series of n frames), D is trained to distinguish between synthetic and original inputs by classifying (X, Y) into class 1 and $(X, G(X))$ into class 0. Hence, for each of the stages k, we train D with target $\vec{1}$ for (X, Y) and target $\vec{0}$ for $(X, G(X))$. The loss function for training D can be described as follows:

$$\mathcal{L}_{adv}^{D} = \sum_{k=1}^{N_{stages}} L_{bce}(D_k(X_k, Y_k), \vec{1}) + L_{bce}(D_k(X_k, G_k(X_k)), \vec{0}) \tag{3}$$

where, L_{bce}, the binary cross-entropy loss can be defined as:

$$L_{bce}(Y, Y') = -\sum_{i=1}^{|Y|} Y'^{i} log(Y^i) + (1 - Y'^{i}) log(1 - Y^i), Y^i \in \{0, 1\}, Y'^{i} \in [0, 1] \tag{4}$$

Training of the generator We perform an optimization step on the generator network (G), keeping the weights of D fixed, by feeding a set of consecutive frames X sampled from the training data with target Y (set of ground-truth output frames) and minimize the following the adversarial loss:

$$\mathcal{L}^G_{adv} = \sum_{k=1}^{N_{stages}} L_{bce}(D_k(X_k, G_k(X_k)), \vec{1}) \tag{5}$$

By minimizing the above two losses (eqn. 3, 5), G tries to make the discriminator believe that, the source of the generated frames is the input data space itself. Although this alternate optimization of D and G is theoritically correct, in practical purposes, this produces an unstable system where G can produce samples that consecutively move far away from the original input space and in consequence D distinguishes them easily. To overcome this unstability inherent in the GAN principle, and to make much clear and crisp predicted frames at high resolution, we add to more objective functions (a) Normalized Cross Correlation Loss (NCCL) and (b)Pairwise Contrastive Divergence Loss (PCDL) to the original adversarial loss (refer to eqn. 3,5).

3 Normalized Cross-Correlation Loss

Video data is different from traditional image data in the fact that, it offers a far richer space of data distribution by adding the temporal dimension along with the spatial one. Convolutional Neural Networks (CNN) can only capture short-range relationships, a small part of the vast available information, from the input video data, that too in the spatial domain. Although this can be somewhat alleviated by the use of 3D convolutions [7], this increases the number of learnable parameters by a large scale. Normalized cross-correlation has been used since long time in the field of video analytics [1, 12, 9] to model the space-time relationships present in the data.

Normalized cross correlation (NCC) measures the similarity of two image patches as a function of the displacement of one relative to the other. This can be mathematically defined as

$$NCC(f,g) = \sum_{x,y} \frac{(f(x,y) - \mu_f)(g(x,y) - \mu_g)}{\sigma_f \sigma_g} \tag{6}$$

where, $f(x,y)$ is a subimage, $g(x,y)$ is the template to be matched, μ_f, μ_g denotes the mean of the subimage and the template respectively and σ_f, σ_g denotes the standard deviation of f and g respectively.

In the domain of video frame(s) prediction, we incorporate the NCC by first extracting small non-overlapping square patches of size $h \times h$ ($1 < h \leq 4$), denoted by a 3-tuple $P_t\{x, y, h\}$, where, x and y are the co-ordinates of the top-left pixel of a particular patch, from the predicted frame at time t and then calculating the cross-correlation score with the patch extracted from the ground truth frame at time $(t-1)$, represented by $\hat{P}_{t-1}\{x-2, y-2, h+2\}$.

In simpler terms, we aim to find the cross-correlation score between a small portion of the current predicted frame and the local neighborhood of that in the previous ground-truth frame. We assume that, the motion present in the full frame can be subdivided into smaller parts and can be effectively approximated by looking into small local neighborhoods in the temporal dimension. This stems from the fact that, unless the video contains significant jitter or unexpected random events like scene change, the motion features remain smooth over time.

The step-by-step process for finding the cross-correlation score by matching local patches of predicted and ground truth frames is described in algorithm 1.

The idea of calculating the NCC score can be effectively modeled into an objective function for the generator network G, where, it tries to maximize the score over a batch of inputs. In essence, this objective funtion tries to model the temporal data distribution by smoothing the local motion features generated by the convolutional model. This loss function, $\mathcal{L}_{NCC}$ can be defined as

$$\mathcal{L}_{NCC} = \sum_{batch=1}^{N} Score_{NCC} \tag{7}$$

where, $batch$ denotes a minibatch of input frames and $Score_{NCC}$, obtained using algorithm 1, is the average normalized cross-correlation score per batch. The generator tries to maximize $\mathcal{L}_{NCC}$ along with the adversarial loss defined in section 2.

```
**Input:** Ground-truth frames ($GT$), Predicted frames ($PRED$)
**Output:** Cross-correlation score ($Score_{NCC}$)
**Variables:**
$h$ = height and width of an image patch
$t$ = current time;
**Initialize:** $Score_{NCC} = 0$;
for $t = 1$ upto $T$ do
    for $i = 0$ upto $H$, $i \leftarrow i + h$ do
        for $j = 0$ upto $H$, $j \leftarrow j + h$ do
            $P_t \leftarrow extract\_patch(PRED_t, i, j, h)$;
            \\ Extracts a patch from the predicted frame at time $t$ of dimension $h \times h$ starting from the top-left pixel index $(i, j)$;
            $\hat{P}_{t-1} \leftarrow extract\_patch(GT_{t-1}, i-2, j-2, h+2)$;
            \\ Extracts a patch from the ground-truth frame at time $(t-1)$ of dimension $(h+2) \times (h+2)$ starting from the top-left pixel index $(i-2, j-2)$;
            $\mu_{P_t} \leftarrow avg(P_t)$;
            $\mu_{\hat{P}_{t-1}} \leftarrow avg(\hat{P}_{t-1})$;
            $\sigma_{P_t} \leftarrow standard\_deviation(P_t)$;
            $\sigma_{\hat{P}_{t-1}} \leftarrow standard\_deviation(\hat{P}_{t-1})$;
            $Score_{NCC} \leftarrow Score_{NCC} + Absolute\big(\sum_{x,y} \frac{(P_t(x,y)-\mu_{P_t})(\hat{P}_{t-1}(x,y)-\mu_{\hat{P}_{t-1}})}{\sigma_{P_t}\sigma_{P\hat{}_{t-1}}}\big)$;
        end
    end
end
$Score_{NCC} \leftarrow avg(Score_{NCC})$;
```

Algorithm 1: Calculation of the normalized cross-correlation score for finding similarity between a set of predicted frame(s) and a set of ground-truth frame(s).

4 Pairwise Contrastive Divergence Loss

As discussed in 3, the proposed method tries to capture motion features that vary slowly over time. The NCCL aims to achieve this using local similarity measures. To complement this in a global scale, we use the idea of pairwise contrastive divergence over the input frames. The idea of exploiting this temporal coherence for learning motion features has been studied in the recent past [5, 6, 11].

By assumming that, motion features vary slowly over time, we describe $\hat{Y}_t$ and $\hat{Y}_{t-1}$ as a temporal pair, where, $\hat{Y}_i$ and $\hat{Y}_{t-1}$ are the predicted frames at time t and $(t-1)$ respectively, if the outputs of the discriminator network D for both these frames are 1. With this notation, we model the slowness principle of the motion features using an objective function as

$$\begin{aligned}\mathcal{L}_{PCDL} &= \sum_{i=0}^{T} D_\delta(\hat{Y}_i, \hat{Y}_{i+1}, p_i) \\ &= \sum_{i=0}^{T} p_i d(\hat{Y}_i, \hat{Y}_{i+1}) + (1-p_i) max(0, \delta - d(\hat{Y}_i, \hat{Y}_{i+1}))\end{aligned} \tag{8}$$

where, T is the time-duration of the frames predicted, p_i is the output decision ($p_i \in \{0, 1\}$) of the discriminator, $d(x, y)$ is a distance measure ($L2$ in this paper) and δ is the margin.

Equation 8, in simpler terms, tries to minimize the distance between frames that have been predicted correctly and encourages the distance in the negative case, upto a margin δ.

PCDL can be extended upto higher order versions, taking into account triplets or n number of predicted frames instead of the general pairwise case. For the sake of simplicity and to keep the training time low, we did not exploit versions above $n = 2$, although this may improve the overall result.

5 Combined Loss

Finally, we combine the objective functions described in sections 2 - 8 along with the general $L2$-loss with different weights as

$$\begin{aligned}\mathcal{L}_{Combined} =& \lambda_{adv}\mathcal{L}^{G}_{adv}(X,Y) + \lambda_{L2}\mathcal{L}_{L2}(X,Y)\\ &+ \lambda_{NCCL}\mathcal{L}_{NCCL}(X,Y) + \lambda_{PCDL}\mathcal{L}_{PCDL}(X,Y)\end{aligned} \tag{9}$$

For the sake of simplicity, all the weights *viz.* $\lambda_{L2}, \lambda_{NCCL}$ and λ_{PCDL} have been set as 1, while $\lambda_a dv$ equals 0.05 for all the experimental studies.

6 Experiments

Experimental studies of our video frame(s) prediction model have been carried out on video clips from Sports-1m, UCF-101 [13] and KITTI [3]. The input-output configuration used for training the system is as follows: **input**: 4 frames and **output**: 6 frames. We compare our results to recent state-of-the-art methods by computing two popular metrics: (a) **Peak Signal to Noise Ratio (PSNR)** and (b) **Structural Similarity Index Measure (SSIM)** [14].

6.1 Datasets

Sports-1M A large collection of sports videos collected from YouTube spread over 487 classes. The main reason for chosing this dataset is the amount of movement in the frames. Being a collection of sports videos, this has sufficient amount of motion present in most of the frames, making it an efficient dataset for training the prediction model.

UCF-101 This dataset contains 13320 annotated videos belonging to 101 classes having 180 frames/video on average. The frames in this video do not contain as much movement as the Sports-1m and hence this is used only for testing purpose.

KITTI This consists of high-resolution video data from different road conditions. We have taken raw data from two categories: (a) city and (b) road.

6.2 Architecture of the network

Table 1: Network architecture. G and D represents the generator and discriminator networks respectively. U denotes an unpooling operation which upsamples an input of dimension $h \times h$ into $(h \times 2) \times (h \times 2)$.

Network	Stage-1 (G)	Stage-2 (G)	Stage-1 (D)	Stage-2 (D)
Number of feature maps	64, 128, 256U, 128, 64	64, 128, 256, 512U, 256, 128, 64	64, 128, 256	128, 256, 512, 256, 128
Kernel sizes	5, 3, 3, 3, 5	5, 5, 5, 5, 5, 5	3, 5, 5	7, 5, 5, 5, 5
Fully connected	N/A	N/A	1024, 512	1024, 512

The architecture used for the generator (G) and discriminator (D) networks for experimental studies is shown in table 1. All of the convolutional layers except the last one in both stages of G is followed by relu non-linearity. The last layer is tied with tanh activation function. In both the stages of G, we use unpooling layers to upsample the image into higher resolution in magnitude of 2 in both dimensions (height and width). The learning rate is set to 0.003 for G, which is gradually decreased to 0.0004 over time. The discriminator (D) uses ReLU non-linearities and is trained with a learning rate of 0.03. We use minibatches of 8 clips for training the overall network.

6.3 Evaluation metric

Assesment of the quality of the predicted frames is done by two methods: (a) Peak Signal to Noise Ratio (PSNR) and (b) Structural Similarity Index Measure (SSIM).

PSNR Given a ground-truth frame Y and a predicted frame $\hat{Y}$ of dimension $n \times n$, PSNR can be formulated as

$$PSNR(Y, \hat{Y}) = 10log_{10} \frac{max_{\hat{Y}}^2}{\frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(Y(i,j) - \hat{Y}(i,j))^2} \tag{10}$$

where, $max_{\hat{Y}}^2$ denotes the maximum possible pixel value of the image.

SSIM Calculated over various windows of an image, the SSIM between two windows x and y can be calculated as

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{11}$$

where, μ_x, μ_y are the mean of x, y, σ_x^2, σ_y^2 are the variance of x, y, σ_{xy} is the covariance of x and y and C_1, C_2 are two constants.

As the frames in videos are composed of foreground and background, and in most cases the background is static (not the case in the KITTI dataset, as it has videos taken from camera mounted on a moving car), we extract random sequences of 32×32 patches from the frames where there is significant motion. Calculation of motion presence is done by the use of optical flow method of *Brox et. al.* [2].

6.4 Comparison

We compare the results by testing on videos from UCF-101 using model trained on the Sports-1M dataset in table 2. Superiority of our method over the most recent work [10] can be clearly detrmined from the comparison. We followed similar choice of test set videos as in [10] to make a fair comparison. One of the impressive facts in our model is that, it can produce acceptably good predictions even in the 6th frame, which is a significant result, considering the compared model uses separate models for achieving this feat.

Table 2: Comparison of different methods for the UCF-101 dataset. (*) indicates models fine tuned on patches of size 64×64 [10]. (-) denotes unavailability of data. GDL stands for Gradient Difference Loss [10].

	1st frame prediction scores		2nd frame prediction scores		6th frame prediction scores	
Methods	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
L2	27.6	0.86	22.5	0.81	-	-
L1	28.7	0.88	23.8	0.83	-	-
GDL L1	29.4	0.90	24.9	0.84	-	-
GDL L1*	29.9	0.90	26.4	0.87	-	-
Adv + GDL fine-tuned*	32.0	0.92	28.9	0.89	-	-
Adv + NCCL	33.6	0.93	29.8	0.90	20.2	0.65
Adv + NCCL + PCDL	0	0	0	0	0	0
Adv + NCCL + PCDL + L2	0	0	0	0	0	0

We also trained our model on the KITTI dataset and report the findings in table 3. As this dataset consists of road data captured using a camera mounted on a moving car, it has good motion presence in most of the frames. In spite of this, we followed the method of calculating strong motion areas as in the case of Sports-1M to ensure proper training of the models.

Finally, we show the prediction results obtained on both the UCF-101 and KITTI in figure (INSERT FIG).

7 Conclusion

In this paper, we modified the Generative Adversarial Networks (GAN) framework with the use of unpooling operations and introduced two objective functions based on the normalized cross-correlation and the contrastive divergence estimate, in the domain of video frame(s) prediction.

Table 3: Experimental results on KITTI datset.

	1st frame prediction scores		2nd frame prediction scores		6th frame prediction scores	
Methods	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Adv + NCCL	0	0	0	0	0	0
Adv + NCCL + PCDL	0	0	0	0	0	0
Adv + NCCL + PCDL + L2	0	0	0	0	0	0

Studies show clear improvement of the proposed methods over the recent best methods. These objective functions can be used with more complex networks involving 3D convolutions and recurrent neural networks. In the future, we aim to learn weights for the cross-correlation such that it focuses adaptively on areas involving varying amount of motion.

References

[1] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. In *Aerospace/Defense Sensing, Simulation, and Controls*, pages 95–102. International Society for Optics and Photonics, 2001.

[2] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2011.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[4] 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.

[5] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4086–4093, 2015.

[6] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

[7] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[8] 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.

[9] J. Luo and E. E. Konofagou. A fast normalized cross-correlation calculation method for motion estimation. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 57(6):1347–1357, 2010.

[10] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[11] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM, 2009.

[12] A. Nakhmani and A. Tannenbaum. A new distance measure based on generalized image normalized cross-correlation for robust video tracking and image recognition. *Pattern recognition letters*, 34(3):315–321, 2013.

[13] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[15] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.