Towards sharper video predictions beyond traditional loss functions

Akilesh B

Introduction

- Learning to predict future images from a video sequence involves the construction of an internal representation that models the image evolution accurately, and therefore, its content and dynamics.

- In this work, we address the problem of frame prediction as many vision applications could benefit from the knowledge of the next frames of videos.

Problems?

- Difference with image reconstruction is that the ability of a model to predict future frames requires to build accurate, non trivial internal representations, even in the absence of other constraints (such as sparsity).
- However, lack of sharpness (or losing high-frequency components) can be observed in future frame predictions.
- The cause of this problem is multi-fold in nature, like: bad choice of loss function, architecture constraints etc.

Related work

- Ranzato et al. [1] defined a recurrent network architecture inspired from language modeling, predicting the frames in a discrete space of patch clusters.

- Srivastava et al. [2] adapted a LSTM model (add_ref) to future frame prediction. However the *l*2 loss function inherently produces blurry results as it comes from the assumption that the data is drawn from a Gaussian distribution and works poorly with multimodal distributions.

Methodology

$Y = \{Y^1, Y^2, Y^3 \dots Y^n\}$ be a sequence of frames to predict from input frames $X = \{X^1, X^2, X^3 \dots X^m\}$ in a video sequence.

Traditionally, it's done by minimizing the distance for instance l_p

$$\mathcal{L}_p(X, Y) = \ell_p(G(X), Y) = \|G(X) - Y\|_p^p$$

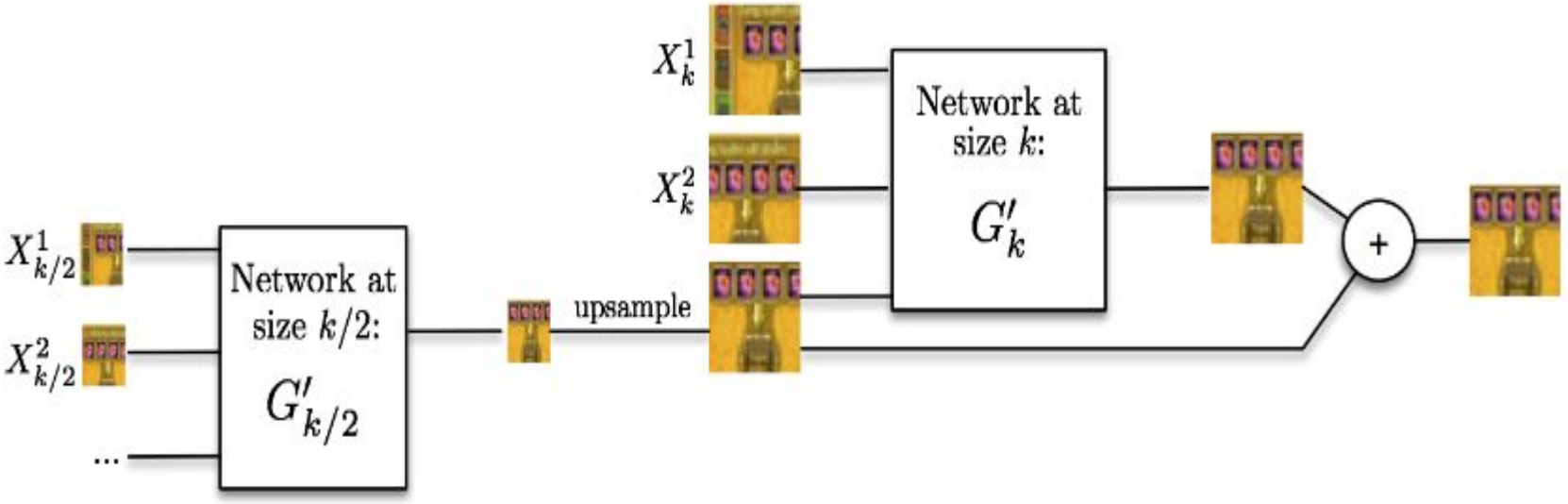
Problems with above approach?

- Convolutions only account for short-range dependencies, limited by the size of their kernels. The approach used in this paper is to combine multiple scales linearly as in the reconstruction process of a Laplacian pyramid [3].
- Using an l2 loss, and to a lesser extent l1 , produces blurry predictions.
- If the probability distribution for an output pixel has two equally likely modes v1 and v2, the value v_{avg} = (v1 +v2) / 2 minimizes the l_2 loss over the data, even if v_{avg} has very low probability.

Approach

- Use Adversarial training for video prediction.
- The discriminative model *D* takes a sequence of frames, and is trained to predict the probability that the last frames of the sequence are generated by *G*.
- The discriminative model *D* is a multi-scale convolutional network with a single scalar output.

Multi-scale architecture



Training D

Let (X, Y) be a sample from the dataset. Note that X (respectively Y) is a sequence of m (respectively n) frames.

We train D to classify the input (X, Y) into class 1 and the input (X, G(X)) into class 0.

More precisely, for each scale k, we perform one SGD iteration of D_k while keeping the weights of G fixed.

$$\mathcal{L}_{adv}^{D}(X,Y) = \sum_{k=1}^{N_{\text{scales}}} L_{bce}(D_k(X_k, Y_k), 1) + L_{bce}(D_k(X_k, G_k(X)), 0)$$

Training G

Let (X, Y) be a different data sample. While keeping the weights of D fixed, we perform one SGD step on G to minimize the adversarial loss.

$$\mathcal{L}_{adv}^{G}(X,Y) = \sum_{k=1}^{N_{\text{scales}}} L_{bce}(D_k(X_k, G_k(X_k)), 1)$$

Algorithm

Algorithm 1: Training adversarial networks for next frame generation

Set the learning rates ρ_D and ρ_G, and weights $\lambda_{adv}, \lambda_{\ell_p}$.

while *not converged* **do**

Update the discriminator D**:**

Get M data samples $(X, Y) = (X^{(1)}, Y^{(1)}), \ldots, (X^{(M)}, Y^{(M)})$

$$W_D = W_D - \rho_D \sum_{i=1}^{M} \frac{\partial \mathcal{L}_{adv}^{D}(X^{(i)}, Y^{(i)})}{\partial W_D}$$

Update the generator G**:**

Get M new data samples $(X, Y) = (X^{(1)}, Y^{(1)}), \ldots, (X^{(M)}, Y^{(M)})$

$$W_G = W_G - \rho_G \sum_{i=1}^{M} \left(\lambda_{adv} \frac{\partial \mathcal{L}_{adv}^{G}(X^{(i)}, Y^{(i)})}{\partial W_G} + \lambda_{\ell_p} \frac{\partial \mathcal{L}_{\ell_p}(X^{(i)}, Y^{(i)})}{\partial W_G} \right)$$

Combining losses

$$\mathcal{L}(X,Y) = \lambda_{adv}\mathcal{L}^{G}_{adv}(X,Y) + \lambda_{\ell_p}\mathcal{L}_p(X,Y) + \lambda_{gdl}\mathcal{L}_{gdl}(X,Y)$$

Where,

$$\mathcal{L}_{gdl}(X,Y) = L_{gdl}(\hat{Y},Y) = \sum_{i,j} \Big| \big||Y_{i,j} - Y_{i-1,j}| - |\hat{Y}_{i,j} - \hat{Y}_{i-1,j}|\big|^{\alpha} + \big||Y_{i,j-1} - Y_{i,j}| - |\hat{Y}_{i,j-1} - \hat{Y}_{i,j}|\big|^{\alpha}$$

Experiments

Used UCF101 dataset. Code in Tensorflow.

We use 4 input frames to predict one future frame.

We train our network by randomly selecting temporal sequences of patches of 32 x 32 pixels after making sure they show enough movement (quantified by the l_2 difference between the frames). This is done primarily to keep the memory usage low.

Evaluation metrics

$$\text{PSNR}(Y, \hat{Y}) = 10 \log_{10} \frac{\max_{\hat{Y}}^2}{\frac{1}{N} \sum_{i=0}^{N} (Y_i - \hat{Y}_i)^2}$$

$$\text{Sharp. diff.}(Y, \hat{Y}) = 10 \log_{10} \frac{\max_{\hat{Y}}^2}{\frac{1}{N} \left(\sum_i \sum_j |(\nabla_i Y + \nabla_j Y) - (\nabla_i \hat{Y} + \nabla_j \hat{Y})| \right)}$$

where $\nabla_i Y = |Y_{i,j} - Y_{i-1,j}|$ and $\nabla_j Y = |Y_{i,j} - Y_{i,j-1}|$.

Deblurring approach 1

The authors in [add_ref] propose a simple yet effective l_0 regularized prior based on intensity and gradient for text image deblurring.

The proposed image prior is motivated by observing distinct properties of text images.

We first explore the usage of this deblurring algorithm by passing the frames predicted by the above approach through this deblur module.

Deblurring approach 2

- Explore the usage of Unnatural L_0 Sparse Representation for Natural Image Deblurring as proposed in [add_ref].
- This is specialized for natural image deblurring, so gave good results when we pass the predicted frame through it.

Results

With deblur approach 1, very bad! (overly specialized for text de-blurring)

input_3

gen_0

gt_0

Deblurred
(approach 2)

input_3

gen_0

gt_0

Deblurred
(approach 2)

input_3

gen_0

gt_0

Deblurred
(approach 2)

Quantitative results

Method	PSNR	Sharpness
Our approach (before deblurring)	23.7	15.8
State-of-the-art	31.5	25.4

The above results are on 378 test videos from UCF101 (same test set used by SoTA)

Drawbacks with above approach?

Although the results given by natural image deblurring approach 2 is good, it's a two stage approach (predict frame and then deblur). So it's not end-to-end trainable.

Perceptual loss? - propose the use of features extracted from a pretrained VGG network instead of low-level pixel-wise error measures.

Specifically use a loss function based on the euclidean distance between feature maps extracted from the VGG19 network. (eg. in Super Resolution GAN paper).

Multiple discriminators?

Inspired by GMAN paper (link), we used two discriminators D1, D2. The generator trains using feedback aggregated over multiple discriminators (in our case it’s the mean).

D1 detects whether it's real or fake and D2 detects the blurriness (the input to D2 is ground truth frame and blurred ground truth frame).

We did a Gaussian blurring with 5x5 kernel and variance 5. D1, D2 are trained simultaneously and their losses are averaged before passing it to the G.

Quantitative results

The values of PSNR and sharpness are 23.0528, 15.3786 respectively after 145000 steps. The best values reported in paper are 31.5, 25.4 respectively.

Method	PSNR	Sharpness
Ours	23.0528	15.3786
SoTA	31.5	25.4

What's up next?

In previous approach, D2 doesn't have the generated image passed as input.

Pre-train D2 to specialize on blur detection and then perform adversarial training.

We are using 10000 images of CIFAR 10 blurred using one of the three blurring techniques: a) Gaussian blur b) Median blur c) Bilateral filtering, for training D2.

Issues?

We are facing issues with initializing weights of D2 in TensorFlow with pre-trained CIFAR 10 model mentioned above.

Initializing the weights for the entire network in TensorFlow is straight-forward (Ref) but in our case, we want to do it only for D2.

Other possible improvements

Train GAN better? Follow the hacks suggested by Soumith (link).

Some of them include: Use leaky ReLU instead of ReLU to avoid sparse gradients, Use SGD for discriminator and Adam for generator.

Label Smoothing, i.e. if you have two target labels: Real=1 and Fake=0, then for each incoming sample, if it is real, then replace the label with a random number between 0.7 and 1.2, and if it is a fake sample, replace it with 0.0 and 0.3 (for instance).

References

[1]: M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos.

[2] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms.

[3] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks.