Self-supervised Video Prediction

Lab Vision Systems: Learning Computer Vision on GPUs

Schmidt, Heiko
Akter, Mst Mahfuja

Universität Bonn
heiko@uni-bonn.de, Matrikelnummer: 2754786
s6msakte@cs.uni-bonn.de, Matrikelnummer: 3214647

Abstract. To solve self-supervised learning problems including video prediction by using Deep Learning algorithms is a non trivial task for many years. Video prediction problem requires analyzing the video frames, temporally and spatially, and constructing a model to predict how the environment changes over the time. Fully convolutional neural networks prevent the modeling of location-dependent patterns because of its spatially invariant behaviour. In this paper we describe a model to predict future frames from a sequence of frames by using different convolutional modules. In our experiment we compare two models which both take a single seed frame and produce a single output frame. Our proposed method have two major drawbacks, blurriness of predicted frames and copying of the seed frames.

1 Introduction

A wide applications like scene understanding and prediction of future image in computer vision have triggered a lot of research in recent times. Visual recognition systems performing image classification, detection of objects and predict the future movement of objects have been popular and challenging tasks on this area of research. Significant developments in the area of neural networks have fueled the improved performance of future image prediction which is a sub-domain of visual recognition systems.

The Convolutional neural network(CNN) consists of a series of convolutional layers where the neurons are connected from one layer to the next layer. The main applications of these networks is image recognition, object detection, document analysis, historic and environmental collection, etc. For this project we are using fully convolutional neural network, location-depend Convolution neural networks and convolutional gated recurrent networks unit (convGRU). ConvGRU model is a recursive modelule which consider the past sequence of input and store respective information.

This project aims to predict three future frames from three seed frames from a video. To find a good model we used some regularization techniques, brute force search for the hyper parameters and simplified methods of known methods. For

this implementation, we have used **pytorch** library and **pytorch_lightning** to display the performance of model.

The rest of the paper is organized as follows: Section 2 describes the Related Work, Section 3 describes the details of different model we used in our project, Section 4 describes our model architecture and used methods, Section 5 describes the outcome of our experiments and the last section 6 describes the concluding remarks.

2 Related Work

Our proposed model is inspired from the Video Ladder Network (VLN) [CNH+16] and [20119]. VLN is a fully convolutional neural encoder-decoder network that is augmented by both recurrent and feed forward connections in all layers. These connections form a lateral recurrent residual block at each layer, where the feed forward connection works as a skip connection and the recurrent connection represents the residual connection.

On [20119], the model NimbRoNet2 have used pretrained ResNet18, each convBlock consists of two convolutional layers followed by batch-norm and ReLU activations. Instead of a convolutional layer they have used a location-dependent convolution in the last layer.

The fully convolutional neural network is spatially invariant which prevents to model location-dependent features. To overcome this problem we have introduced a combination of Location-depend convolution [AFB18] and ConvGRU [TAS17] layer in our model. This location dependency convolutional layer can encode location features and learn location dependent biases. The convGRU [TAS17] model extracts moving objects from a video. This model learns spatio-temporal features from intermediate visual representation, hence this part of model has a greatest impact on the future frame prediction in our model.

3 Literature

3.1 Video Ladder Network

Video Ladder Network (VLN) [CNH+16] is a fully convolutional neural network to generate future video frames efficiently. VLN is an encoder-decoder network where the decoder exploits temporal summaries generated from all layers of the encoder. The encoder consists of dilated convolutional layers, whereas the decoder uses normal convolutions. Both encoder and decoder use Batch-Bormalization (BN) and leaky-ReLU activation function followed by a convolutional layer. This way, the model provides fast inference and achieves outstanding results, while having a simple network structure.

3.2 Location Dependency Convolutional Neural Network

To predict video frame, location dependency on deep convolution network [AFB18] introduces new location-biased convolutional layers to overcome the spatial invariant behavior of CNN. This model encodes location features of the input in

separate channels and convolutional layers with learnable location-dependent biases. The effectiveness of location-dependent bias is evaluated on different architectures including VLN 3.1 model. This model significantly outperformed spatially invariant models in video prediction.

3.3 Convolutional Gated Recurrent Unit

The convolutional gated recurrent unit (ConvGRU) [TAS17] that encodes the spatio-temporal evolution of objects in the input video sequence. This model extract moving objects from the input image sequences by learning patterns during the forward pass. The ConvGRU unit replaces the linear layer at the gates with convolutional layers

ConvGRU model learn from a small number of input video sequences. It takes a video frame as input, assigns each pixel an object or background label based on the learned spatio-temporal features as well as the "visual memory" specific to the video. The visual memory is implemented with convolutional gated recurrent units, which allows to propagate spatial information over time. This method have been very successful in the future image prediction tasks and proved to produce good results for object localization.

4 Methods

4.1 Our Model

Like the model NimbRoNet2 [20119], we have used the pretrained ResNet18 shown in figure 1 in the right part of model.

The middle part of the model is made of ConvGRU blocks which contain a convolutional layer and three ConvGRU layer. This blocks contain recursive connections which makes the model be able to use old information. The last two convGRU blocks contain location-dependent convolution instead of a convolutional layer. Each convGRU block processes the input and produces an output. Later all outputs get concatenated.

Left part of the model is sequential conv block which contains a ReLU activation, a Batch-Norm, then a convolutional layer, next a 2×2 pixel shuffle and at last again a convolutional layer.
Figure 1 shows the model architecture. To keep the model simple, residual connections of Residual Network are not depicted.

4.2 Dataset (UFC101)

In our project we used UCF101 - Action Recognition data set. Since our implementation is in pytorch we used torchvision.datasets.UCF101. This dataset creates a big collection of small video clips of fixed length, specified by the parameters "frames_per_clip" and "step_between_clips". The last parameter controls how many steps are between each clip. There is a parameter which specifies if the data set is in mode train or test. The significant difference between both is that the test data set can contain clips of different resolutions.

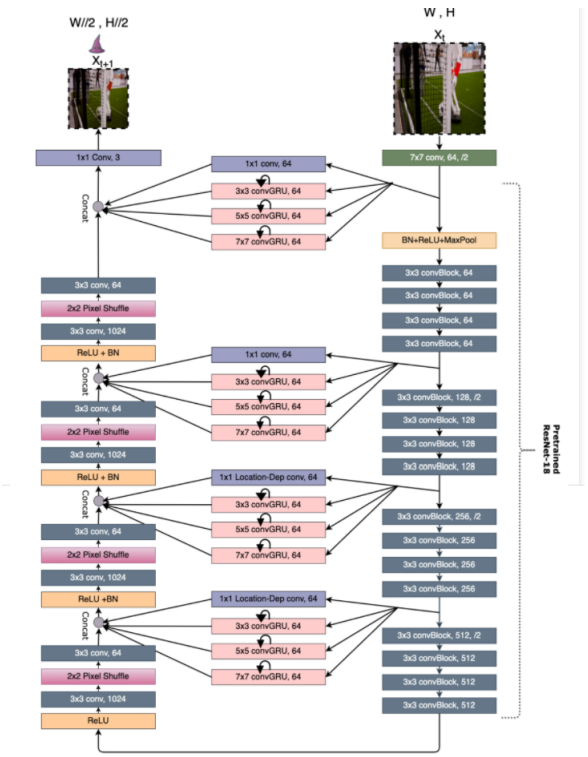


Fig. 1: Self-supervised Video Prediction model. Right: Input image is passed into convBlock. Middle: The outcome from ConvBlock is passed into ConvGRU block. Right: The outcome from convGRU block is concatenated with two CNN layer and feed into forward. [* The model is taken from vision lab systems project assignment]

4.3 Data Augmentations

For the video prediction task we used the same standard data augmentations as for the Resnet18. These can be found in the original paper [HZRS15]. For our data set we have slight different parameters for training, validation and test.

For the training data we first apply color jittering(related to [KSH12]), after that with probability 0.5 randomly flip the image horizontally , then apply a random resized crop of the image(scale: $90\% - 100\%$) and at last apply the normalization for standard image networks.

The validation data augmentations differs from the training data augmentations by using a normal resizing while the test data augmentations also uses a normal resizing and doesn't apply the random horizontal flip.

4.4 Early Stopping

Early stopping is form of regularization which is used to avoid overfitting while training the model. In our model we used early stopping mainly to prevent the model from fast convergences into bad local optima. These local optima doesn't necessarily have to correspond to overfitting. This will be explained in more detail in the next chapter.

4.5 Cyclic learning rates

The method of cyclic learning rate [Smi15] is a technique which can solve the problem of finding a adequate learning rate for the network. We were inspired by this technique and made a very simple version of it.

For our model we have small list of learning rates starting with a small one. The list is to see in table 1. The reasoning for the exact values of the learnrates and the choice of differences come purely from the experience of running over 50 models.

4.6 Weight decay

Weight decay is a regularization technique which prevents overfitting. Weight decay in our model had a slightly different effect. We had a lot of models where the learning curves converged but neither improves nor deteriorates. The main issue here is that the training is stuck. Weight decay overcomes this because it flattens the error surface and allows the training slowly to go on and find better minimums.

4.7 Hyperparameters

Normally in literature, researcher uses techniques as k-fold cross validation, grid-search, random-search, random-optimization, etc. to find good hyperparameters for their model. In our work we couldn't use any of these techniques because our time window to train models were short after we validated the correctness of our implementation. We have used frames_per _clip = 24 and frames_between _clips = 60 in our model. All other hyperparameters are depicted in the table 1

Batch size	32	**Image size**	128×128
Optimizer	Adam	**Weight Decay**	$1e^{-3}, 1e^{-3}, 1e^{-3}, 5e^{-3}$
Early stopping	pat:35 delta:0.05	**Learning rate**	$1e^{-6}, 1e^{-6}, 1e^{-7}, 1e^{-5}, 1e^{-6}$
Momentum	0.2	**loss functions**	MSE, L1, DSSIM
Gradient clipping	3.0	**test loss**	2.18026

Table 1: Hyperparameter list which we have used in our model

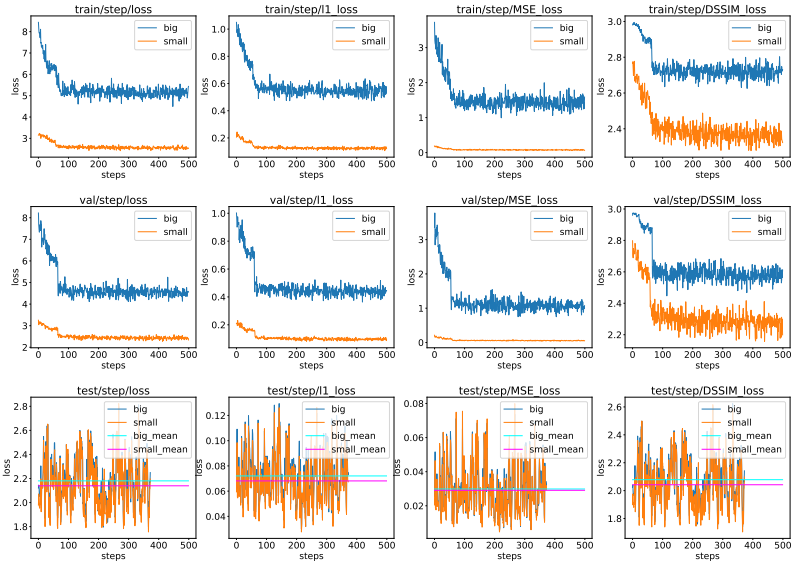
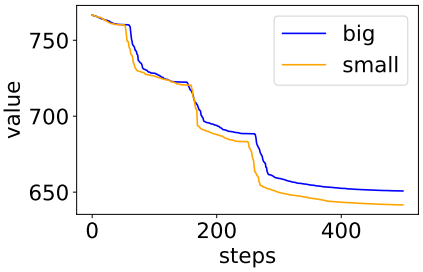


Fig. 2: Loss curves of total loss, L1-loss, MSE-loss and DSSIM loss for both models; top row is train loss; middle row is validation loss; bottom row is test loss, also contains mean line



Weight plot: sum of the norm of each parameter from each model

Steps	MSE	avg_MSE
660/660	0.0702	0.0006
1980/1320	0.0681	0.0006
3960/2640	0.0653	0.0006
7267/5296	0.0647	0.0006

Table 2: Depicts the MSE loss between each parameter of both models(left); and the average of the MSE(right); steps: "small"(left),"big"(right)

5 Results

5.1 Experiment setup and results

Given the 3 seed frames s_1, s_2, s_3 we decided to use a simple strategy to create the prediction frames p_1, p_2, p_3. Our strategy: s_i produces p_i, so that every prediction frame only uses a single input frame for information.

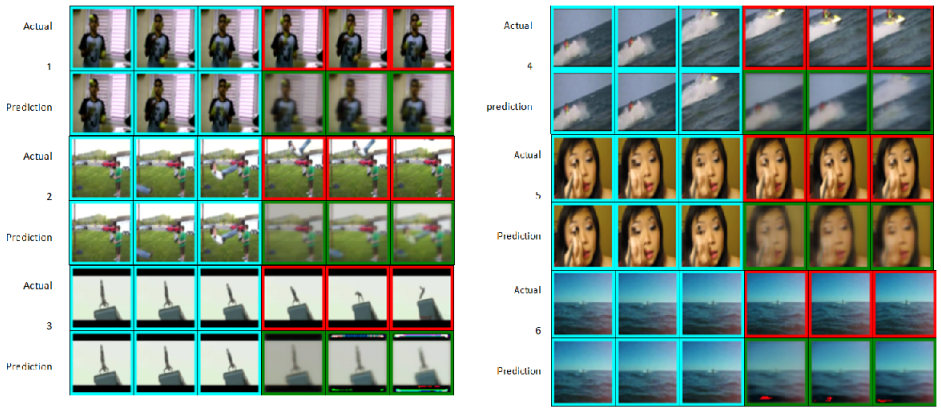


Fig. 3: Video prediction of both model tested with dataset UFC101 4.2

For our experiment we trained two models with similar hyperparameters except the scaling of the loss functions. The "small" model uses $[1.0, 0.7, 0.3]$ as scaling for DSSIM-loss, MSE-loss and L1-loss where the "big" model factors all scales by 10. The main purpose of this two model experiment was to compare and see which scaling shows better results in learning the video prediction model.

We trained our networks until the squared sum of model's parameters converged. This can be seen in Figure 2. Experience showed that 4 phases of training are sufficient. The first 3 phases use early stopping and the $4th$ phase is 2 additional epochs of training. Only weight decay is changed in this phases.

Internally an early stopping phase trains each learn rate once and stops when the difference between the current validation loss and the best seen value so far exceed the delta value for more than patience times.

In figure 2 we see that both models are very similar in their learning progress and the kind of images they produce. This is strongly supported by Table 2. Especially, the average of MSE is 0.0006 meaning that only a very small set of parameters have significantly different weights.

Additionally we can see that "small" is achieving smaller losses in training and validation but this effect is more likely to be caused by the scaling rather than learning a better model. We can see the results image on the test data set which indicates that both models perform similar and there are only very small differences.

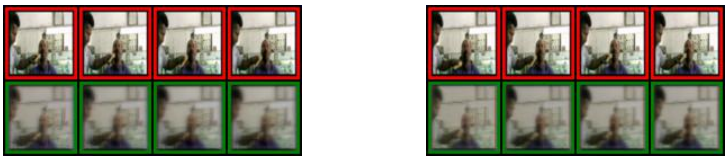
Figure 3 shows a set of samples where each sample compares our predictions with the truth. Frames are bordered with different colors to distinguish them. Seed frames have a cyan border, the correct predictions(labels) have a red border and the predicted frames from our models have a green border.

Sequence 1, 2 and 3 are the outcome of "small" model and Sequence 4, 5 and 6 are from "big" model. As we can see there is not visual difference in both of model.

Sequence 1 and 5 look like a good predictions because movements are repetitive and no fast movements among frames. In sequence 2 and 4 we see that the model is copying the 1 frame into 4, the 2 frame into 5, and 3 frame into 6. Hence prediction is far away from truth frames. In sequence 3 and 6, the model is not only copying the seed frames but also artifact is seen in predicted images. Moreover, our prediction images are little blurry than our actual frames.

5.2 Difficulties of resulting images

Blurriness of images:



(a) "big" model: single frame at $660, 1320, 2640, 5296$ steps while training

(b) "small" model: single frame at $660, 1980, 3960, 7276$ steps while training

Fig. 4: Blurriness of frames while training and test in both model

The first problem we encountered in the video prediction task is that our predicted frames are blurry and not sharp as the actual truth frames. This result can be seen in the Figures 4a and 4b which show snapshots of an direct comparison of the truth frames and our predictions during training.Also the Figures show that training the model until the squared sum of weights converges, doesn't improve the sharpness of the predicted images.

On the other hand, In Figure 3 in sample 5 we can can see that the $2nd$ and $3rd$ predictions are sharper on testdata.The root cause is that the hidden states of the convGRU already contain information about the video and can be used to produce sharper images. So feeding information to the hidden states can improve sharpness while training longer doesn't.

A possible experiment to analyse this effect is to repeat the input feeding multiple times for each prediction frame before actually computing the loss.

Copying of seed frames

The second problem we encountered in the video prediction task is that our models learned to copy the seed frames instead of actually predict future movement. This effect can be seen in 3.

Our ideas to overcome this problem is based on the assumption that the copying functionality comes from the convergence into bad minima. To learn the identity function on the input can be considered a good reduction of the loss for the model but it miss to learn the actual functionality.

We used early stopping and weight decay to overcome this problem. Both methods are known to help against a convergence into minima which overfits the training data. Unfortunately our results show that these regularization techniques had no effect. They helped to avoid a fast early convergence.

Additionally we tried to find the root cause of the copying problem by changing parts of our implementation, trying out different hyper parameters and use all 3 input frames to predict the next frames. None of these options solved the problem.

6 Conclusion

Two challenges in our future frame prediction are blurriness of images and the copying of the seed frames. In our experiment with two models we have shown that scaling the loss functions has a very little impact on what functionality the models will learn. Also our resulting prediction shows that the network actually could produce real pictures but they are blurry. This problem might be resolved by repeatedly input the seed frames so that the hidden states of the convGRU contain better information.

The other problem is that our models didn't predict future movement but copied the given input frames. The methods early stopping, learning until the weights of the model converges and weight decay couldn't resolve this problem. Training the model for longer might resolve this. Unfortunately we can't give intuition how to solve this problem by our approach.

Video prediction task is indeed a challenging problem for which one must choose appropriate architecture and philosophy to achieve good results. Especially the processing of the seed information plays a crucial role.

References

20119. Robocup 2019: Robot world cup xxiii. *Lecture Notes in Computer Science*, 2019.

AFB18. Niloofar Azizi, Hafez Farazi, and Sven Behnke. Location dependency in video prediction, 2018.

CNH+16. Francesco Cricri, Xingyang Ni, Mikko Honkala, Emre Aksu, and Moncef Gabbouj. Video ladder networks, 2016.

HZRS15. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

KSH12. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Smi15. Leslie N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015.

TAS17. Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning video object segmentation with visual memory, 2017.