Future Frame Prediction with Generative Adversarial Networks

Magnus Wallgren

Master's thesis
2019:E16

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

Contents

Abstract

This report is about using generative adversarial networks with predictive coding networks for future frame prediction. Model selection choices for the components of the network are explored by training different models and testing their performance on next frame prediction in digital video from driving scenarios. Benefits and issues of using adversarial loss for future frame prediction as well as different choices for the model are discussed.

1 Introduction

Making short-term predictions about the future states of the world has many applications in control and machine intelligence. Autonomous vehicles need to predict where other road users will be to drive in a safe and efficient manner. Robots interacting with humans anticipating how the person will react to a certain action can optimize their behaviour with respect to predicted outcomes. Also weather forecasting, music generation, natural language processing and many other applications are problems best constructed as sequence prediction problems.

This report is concerned with future frame prediction in digital video and the use of a state of the art sequence prediction architecture PredNet [Lotter et al., 2015, Lotter et al., 2016], implemented in a modern artificial neural network framework that is inspired by the Predictive Coding theory in neuroscience [Friston and Kiebel, 2009]. Predictive Coding theory is one of many theories that describes human perception and action control. Predictive coding is a plausible theory in the sense that the brain has the necessary components to do all the computations needed for learning and inference, which makes it a good source of inspiration for machine intelligence using artificial neural networks. This work will look at improving on the PredNet architecture by using adversarial loss. Then a comparison of the performance on future frame prediction in video sequences is done between the network with and without adversarial loss and with different choice of parameters.

In the report there will first be a review of previous work related to Predictive Coding, future frame prediction using artificial neural networks and generative adversarial networks. A theoretical description of the future frame prediction problem will given. Then there will be descriptions of the suggested changes and additions to the model, which are tested and compared against to each other. Finally there will be a discussion followed by a conclusion.

2 Previous Work

Recently there have been a wide range of neural network techniques applied to the problem of future frame prediction. In [Srivastava et al., 2015] long short-term memory (LSTM) units [Hochreiter and Schmidhuber, 1997] was used together with unsupervised learning to predict sequences. [Mathieu et al., 2015] used generative adversarial networks (GAN) in a deep convolutional network to generate next frame predictions. [Xue et al., 2016] described a probabilistic motion sampling model based on variational auto-encoders [Kingma and Welling, 2013] and trained a generative encoder-decoder network to generate a distribution of next frame predictions based on this model. [Kalchbrenner et al., 2016] designed a probabilistic video model that encodes time, space and color structure in a dependency chain to estimate a joint distribution of future frames in a generative network. [Byeon et al., 2017] used a pyramidical convolutional LSTM structure that accumulates activation by propagating the convolutions for each space dimension and in the forward time dimension to the pixel to predict for each pixel. [Liang et al., 2017] separated the frame prediction problem into predicting flow and then predicting motion given flow in two joint networks trained with adversarial loss. [Villegas et al., 2017a] also separated motion and frame prediction but processed motion in different time scales by taking difference images between non-sequent frames. Instead of directly predicting pixels, [Villegas et al., 2017b] made a network that learned to estimate and predict human pose and then to generate an image based on the predicted pose.

[Lotter et al., 2015, Lotter et al., 2016] took inspiration from Predictive Coding [Friston and Kiebel, 2009] to design a neural network called PredNet for next frame prediction. Their network uses convolutional LSTMs as recurrent state units and the difference between prediction and current frame as error units. This is done in a hierarchical structure where errors are propagated up in the hierarchy to higher level states and the recurrent states are propagated down to lower level states. [Zhong et al., 2018] used PredNet along with a Multi-layer Perceptron (MLP) and trained the networks jointly to control a robot, optimizing predicted action, using PredNet as generator.

3 Method

3.1 Future Frame Prediction Problem

Future frame prediction is the problem to estimate the next frame, or a sequence of next frames, given past frames. Here the focus is the problem of finding the next frame. In general the next frame x_{t+1} follows an unknown distribution P that depends on all previous frames

$$P(x_{t+1}|x_t, x_{t-1}, ..., x_1), \tag{1}$$

where $x_1, x_2, ..., x_t$ is a sequence of observed frames. In this setting the video sequence data are natural video sequences, and as such it is only possible to measure one of the real future frames x_{t+1}. The task is to find a function that generates the prediction $\hat{x}_{t+1}$ that maximizes the probability given previous frames as

$$\hat{x}_{t+1} = G_\theta(x_t, r_t) = \max_{x_{t+1}} P(x_{t+1}|x_t, x_{t-1}, ..., x_1), \tag{2}$$

where G_θ is a generator, here implemented as a neural network with weights θ trained to generate predictions given the current frame x_t and the current network state r_t as input. The generator will sometimes be written as G without the subscript of the weights, when the weights are not immediately relevant. The generator G only sees one frame at a time, but can save a representation of previous frames $x_1, ..., x_{t-1}$ as r_t, which endows the network with the ability to encode motion.

The learning problem for the network G_θ is to vary θ to find the G_θ that minimizes some distance measure between the real data x_{t+1} and the generated samples $\hat{x}_{t+1}$. This distance between the predictions and the real next frames, denoted as $L(\hat{x}_{t+1}, x_{t+1})$, is the loss function one wishes to minimize. Some different loss functions such as adversarial loss and L_1 distance will be explored here. The loss can also measure the distance between network states calculated from real and predicted frames as $L(\hat{x}_{t+1}, x_{t+1}, \hat{r}_{t+1}, r_{t+1})$ or more generally the distance between a function of the predicted and real frame $L(u(\hat{x}_{t+1}), u(x_{t+1}))$ for transfer learning, where an example of u could be a utility function of outcomes of actions done based on evidence $\hat{x}_{t+1}, x_{t+1}$. The loss function is optimized by means of some optimization algorithm based on gradient descent. The loss function should be chosen so that the distribution converges to P as the loss function is minimized.

To predict arbitrary future frames τ steps into the futue, the predictions $\hat{x}_{t+1}$ are passed to the network as real frames to iteratively generate one frame at a time. Given a sequence $x_{1,...,t}$ frames $x_{t+\tau+1}$ can, generally, be generated as

$$\hat{x}_{t+\tau+1} = G(\hat{x}_{t+\tau}, r_{t+\tau}), \quad \text{where } \hat{x}_{t+\tau} = G(\hat{x}_{t+\tau-1}, r_{t+\tau-1}) \quad \text{if } \tau > 1, \tag{3}$$

$$\hat{x}_{t+1} = G(x_t, r_t) \quad \text{if } \tau = 1. \tag{4}$$

In this work the focus is on $\tau = 1$.

3.2 Artificial Neural Networks

Artificial Neural Networks are systems that use a model of a neuron in a network to learn to approximate a function from some data. The learning algorithms are referred to as training and is done by means of some gradient descent optimization method, minimizing a loss function with respect to the network node weights. An introduction to neural networks can be found in the deep learning book [Goodfellow et al., 2016]. The reader is assumed to be familiar with neural networks, but most of the report can be understood without that background knowledge.

3.3 Generative Model

The predictive coding network of [Lotter et al., 2015, Lotter et al., 2016] is used as generator network. The network has layers arranged in a top-down hierarchal fashion, where higher layers in the hierarchy carry more abstract representations of the frames. [Lotter et al., 2015, Lotter et al., 2016] use the name levels for what is usually called layers in artificial neural networks. Here the name layer will be used. The model can be seen in figure 1. In each layer there is a convolutional prediction unit that makes a prediction $\hat{x}_t^l$ of the frame x_t^l in layer l in the hierarchy. The prediction error is passed to a convolutional LSTM unit that updates its states r_t^l, which is passed to the prediction unit. The frame x_t^0 is the input image x_t and the states in higher layers are calculated

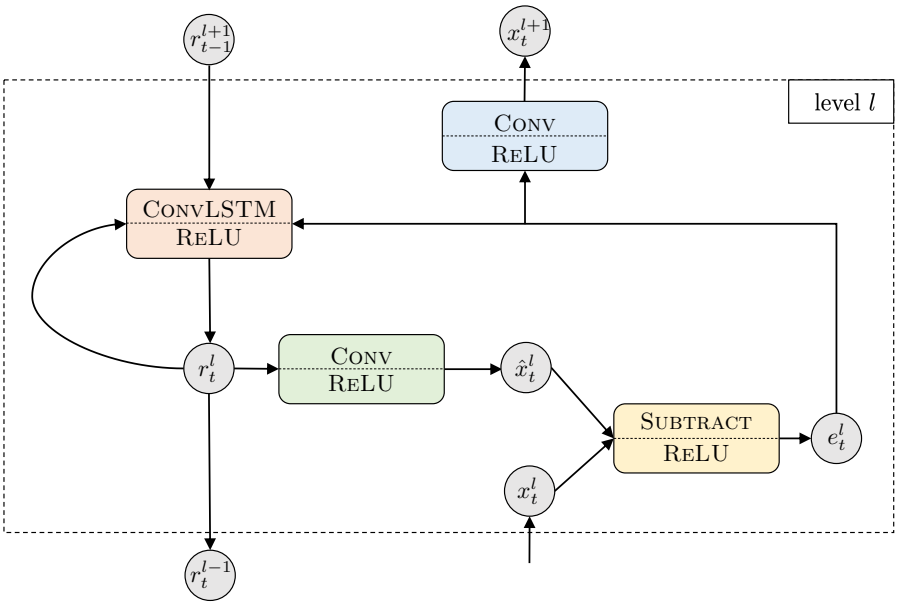

Figure 1: This figure shows the states and model elements in layer l of the PredNet model. These layers are stacked in a hierarchal structure, where x_0^t is the input image and $\hat{x}_0^t$ is the predicted image. At higher layers x_l^t is a more abstract spatial representation of the image dynamics. r_l^t are recurrent representations of the state of the model, represented as the recurrent state of the ConvLSTM unit. e_l^t is the prediction error produced by subtracting the input and the prediction and separating it into two population, one for negative errors and one for positive errors. The exact update equations are listed in (5).

in a convolutional layer that takes the error e_t^{l-1} of the lower layer as input. The update rules are the same as in [Lotter et al., 2016] and are listed in (5).

$$x_t^l = \begin{cases} x_t, & \text{if } l < 0 \\ \text{MaxPool}(\text{ReLU}(\text{Conv}(e_t^{l-1}))), & l > 0 \end{cases} \tag{5a}$$

$$\hat{x}_t^l = \text{ReLU}(\text{Conv}(r_t^l)) \tag{5b}$$

$$e_t^l = [\text{ReLU}(x_t^l - \hat{x}_t^l), \text{ReLU}(\hat{x}_t^l - x_t^l)] \tag{5c}$$

$$r_t^l = \text{ConvLSTM}(e_{t-1}^l, r_{t-1}^l, \text{Upsample}(r_t^{l+1})) \tag{5d}$$

3.4 GAN - Generative Adversarial Network

To train the network the loss function must first be defined. Training a generative model using mean squared error or mean absolute error as loss leads to blurry looking results [Lotter et al., 2016]. A way to combat this is to use adversial loss in a generative adversarial model, GAN [Goodfellow et al., 2014]. The idea behind GAN is to use two adversely trained neural network models, one that generates fake samples, called generator G, and one that distinguishes fake samples from real samples, called critic D. The goal is to train G to generate samples close to the real samples and using the ability of D to distinguish between the fake and real samples in a loss function of the form

$$L_D = L(D(x), D(\hat{x})), \tag{6}$$

instead of using per-pixel mean squared or mean absolute error. The critic D is incorporated into the generator loss function in such a way that minimizing the critic loss makes G try to generate samples that D cannot distinguish from real samples. At the same time the critic loss has the adverse objective, to distinguish real samples from fake samples, in the critic loss.

3.4.1 Related work on GAN for future frame prediction

While GAN is general in that it can be applied to any type of generative model, their typical application is to capture distributions of image sets with no input or only class label inputs. Future frame prediction is closely related to the class of image-to-image translation problem, in which [Isola et al., 2017] took an approach that used GAN conditioned on input images for a wide variety of these problems. While they make the statement that all image-to-image translation problems can be treated the same way, they did not apply it to future frame prediction. In other previous work GANs have been used for future frame prediction. [Liang et al., 2017] and [Jang et al., 2018] used GAN loss to predict motion and optical flow. For the work presented in this report GAN will be used directly as loss on the image frames (6), rather than constructing a loss on motion. This is similar to [Mathieu et al., 2015] who used GAN conditioned on a sequence of input images on different image scales for future frame prediction. Just like in [Isola et al., 2017] the conditioning will be from input image to output image and the GAN loss will be combined with L_1 loss, but instead of using the normal GAN [Goodfellow et al., 2014], the improved version of Wasserstein GAN [Arjovsky et al., 2017, Gulrajani et al., 2017] is the GAN used here. This is because it does not have problems with mode collapse and the generator trains well even when the critic gets ahead in training, thus making training more reliable than ordinary GAN.

3.4.2 Adversarial model

The adversarial model is described in more detail here, with its main components critic loss, generator loss and the training algorithm. The critic loss is as follows.

$$L_D(x_{t+1}, \hat{x}_{t+1}) = \mathbb{E}\left(D(x_{t+1})\right) - \mathbb{E}\left(D(\hat{x}_{t+1})\right) + \lambda_{gp}\,\mathbb{E}\left(||\nabla_{\tilde{x}_{t+1}} D(\tilde{x}_{t+1})||_2 - 1\right), \tag{7}$$

where $x_{t+1} \sim \mathbb{P}_R$ are real samples, $\hat{x}_{t+1} \sim \mathbb{P}_G$ are generated samples and $\tilde{x}_{t+1} \in \mathbb{P}_{\tilde{x}_{t+1}}$ are uniformly sampled on the straight line between the drawn samples x_{t+1} and $\hat{x}_{t+1}$. D is the critic network and λ_{gp} is a loss weight. This is the unconditioned version of the critic loss. The first two terms are the critic losses on real and fake samples. The last term is a penalty on the gradient. The gradient penalty checks the gradient of the critic at a point uniformly sampled between the generated sample and the real sample and penalizes it as it diverges from 1. This is done because the critic has to be 1-Lipschitz in order to be able to use the Kantorovich-Rubinstein duality to obtain the form of 1-Wasserstein distance in (7), which is satisfied when the gradients are less than 1 everywhere [Arjovsky et al., 2017]. Penalizing any divergence from 1, instead of just penalizing the gradients being less than 1, has the added benefit of preventing the vanishing gradient problem.

By minimizing (7) the critic will be trained to minimize $D(x_{t+1})$, the critic value on real samples, and to maximize $D(\hat{x}_{t+1})$, the critic value on fake samples. The critic value can be interpreted as scores for the samples based on how real they look. To further help the critic, it is conditioned on the generator input frame x_t by passing the real/fake sample x_{t+1} or $\hat{x}_{t+1}$ with x_t. If the critic would not be conditioned on input frames it could at best distinguish realistically looking frames, while a conditioned critic can potentially also take states of object in the input frame into account. Writing this in the loss function it becomes

$$L_D(x_{t+1}, \hat{x}_{t+1}|x_t) = \mathbb{E}\left(D(x_{t+1}, x_t)\right) - \mathbb{E}\left(D(\hat{x}_{t+1}, x_t)\right) + \lambda_{gp}\,\mathbb{E}(||\nabla_{\tilde{x}_{t+1}} D(\tilde{x}_{t+1}, x_t)||_2 - 1). \tag{8}$$

A graph visualization of the critic training model is shown in figure 2.

The generator loss function is the following combination of the critic loss on generated samples and L_1 distance,

$$L_G = \mathbb{E}\left(D(\hat{x}_{t+1})\right) + \lambda_{L1}\,\mathbb{E}\left(||\hat{x}_{t+1} - x_{t+1}||_1\right). \tag{9}$$

and the corresponding conditioned loss is

$$L_G = \mathbb{E}\left(D(\hat{x}_{t+1}, x_t)\right) + \lambda_{L1}\,\mathbb{E}\left(||\hat{x}_{t+1} - x_{t+1}||_1\right). \tag{10}$$

Minimizing this loss trains the generator to have a negative critic value on the generated samples, while maintaining low absolute pixel error. The generator training minimizes critic loss on generated samples while the critic training is trying to minimize the negative critic score on generated samples, that is to maximize the critic score on generated samples. These adverse objectives of the critic and generator are trained alternatingly to make the critic better at distinguishing generated samples from real samples and to make the generator better at generating fake samples that can fool the critic.

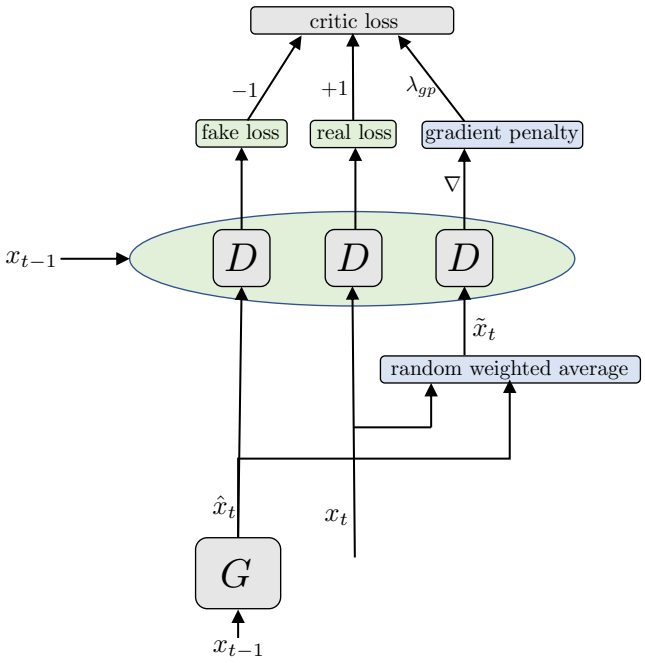

Figure 2: This figure shows a simplified version of the critic computational graph. The input frame x_{t-1} is passed through the generator G to generate a prediction $\hat{x}_t$. $\tilde{x}_t$ is uniformly sampled on the line between the prediction $\hat{x}_t$ and the real frame x_t. x_t, $\hat{x}_t$ and $\tilde{x}_t$ are passed to the critic along with x_{t-1} to get the wasserstein loss and the gradient penalty. The losses are summed as in (8) to get the total critic loss. The loss is applied to every time slice of the input and optimized with back propagation to get the critic weight updates. To be more precise, x_{t-1} is not actually passed to the generator, but rather G generates a prediction based on its internal states and updates its internal states using x_t after generating a prediction. See listing 1.

The L_1 loss term is added as in [Isola et al., 2017], who found that a combination of conditioned adversarial loss and L_1 loss is best for a variety of image-to-image translation tasks. [Lotter et al., 2016] did not use adversarial loss but they had best results using L_1 loss, producing less blurry results than L_2 loss. A graph visualization of the generator training model is shown in figure 3.

3.5 Training

To train the combined model both losses (7) and (9) are minimized. This is done by alternating between updating the generator G_θ weights θ and the critic D_w weights w using Adam steps [Kingma and Ba, 2014]. The training is done in minibatches of size M, with N_D minibatches for every batch. N_D is called the training ratio and defines how many times the critic weights are updated for each generator update. The complete training algorithm is listed in listing 1, with a list of parameters in table 1. First a batch of $M \cdot N_D$ real samples are drawn and split into N_D minibatches. The critic weights are updated using these minibatches and then another minibatch of the same size is made from a combination of samples in each of the minibatches, which is then used to update the generator. The procedure is similar to that in [Gulrajani et al., 2017] but without the sampling and with the conditional critic.

The samples are sequences that the generator and critic process frame by frame. For each time step the generator is first asked to predict the current frame using its internal states, and then it receives the real current frame which it uses to update its internal states. This way the generator can only use the past frame to make predictions. The critic receives the current fake or real frame along with the previous real frame. This way the prediction for the first frame in the sequence is based off of nothing, and this has to be taken into account when updating the weights. This is

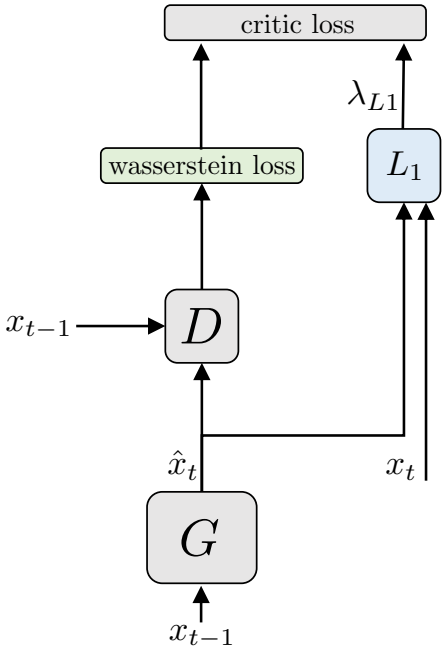

Figure 3: This figure shows the generator computational graph. The generator G takes the input frame x_{t-1} and makes a prediction $\hat{x}_t$. The crtic D is evaluated on $\hat{x}_t$ with x_{t-1} to get the wasserstein loss and the L_1 distance between x_t and $\hat{x}_t$ is calculated to get the L_1 loss. L_1 loss and wasserstein loss are summed to get the total generator loss. This computation is applied to every time slice of the input and optimized using back propagation to update the generator weights. To be more precise, x_{t-1} is not actually passed to the generator, but rather G generates a prediction based on its internal states and updates its internal states using x_t after generating a prediction. See listing 1.

Description	Symbol	Default Value
Number of training iterations	N	31250 (400 epochs)
Training ratio	N_D	5
Minibatch size	M	4
Sequence length	T	10
Gradient penalty weight	λ_{gp}	10
L_1 loss weight	λ_{L1}	100
Learning rate	α	0.0001
Adams momentum weight	β_1	0.0
Adams momentum weight	β_2	0.9
Time loss weights	λ_{t_D}, $\lambda_{t_{L1}}$	See equation 11 and 12

Table 1: This table shows a list of parameters that are used in the training algorithm, listed in listing 1, with their default values.

done by weighting the loss for each time step with time loss weights

$$\lambda_{t_D} = \begin{cases} 0, & \text{if } t = 0, \\ 1, & \text{if } t > 0, \end{cases} \tag{11}$$

$$\lambda_{t_{L1}} = \begin{cases} 0, & \text{if } t = 0, \\ 1/(T-1), & \text{if } t > 0, \end{cases} \tag{12}$$

where λ_{t_D} and $\lambda_{t_{L1}}$ are time loss weights for critic loss and L_1 loss respectively.

3.6 Critic model

A visualization of the critic model is shown in figure 4. The critic uses a structure with stacked convolutional layers followed by densely connected layers. The benefit of using convolutions is that they use weight sharing, requiring less trainable parameters while still being able to capture information over an area in the image. Because of the weight sharing the convolutional layers are also translationally invariant, which is to some extent an expected property in a next frame prediction model, while the densely connected layers are not. The total receptive field of the convolutional layers can be calculated by tracing the filter sizes and strides of the convolution filters backward. For the parameters used in the experiments, shown in table 2, the total receptive field is 70x70 pixels.

layer type	filter size	stride
Conv2D	3x3x16	1
Conv2D	3x3x32	1
Conv2D	3x3x32	2
Flatten		
Dense	128	
Dense	1	

Table 2: This table shows the filter sizes and stride of the convolutional layers in the critic and the number of ouput nodes in the densely connected layers. For densely connected layers the filter size is the number of nodes in the layer. A filter size of AxBxC represents a stack of C filters each covering an area of AxB pixels. The number of input channels to each layer is the stack size C of the previous layer.

3.6.1 Training Ratio

Training ratio N_D is a hyperparameter in the WGAN model that defines how many critic weight updates should be done for each generator weight update. [Gulrajani et al., 2017] use sampling, generating new samples for each critic update. However sampling is not well suited for future frame prediction because the problem has a one-to-one mapping between input and output in the data. Potentially there could be variations in possible future frames, due to noise and the random nature

Algorithm 1 Training algorithm for GAN model. The required values are hyperparameters and are listed in table 1.

Require: number of training iterations N, training ratio N_D, minibatch size M, sequence length T, gradient penalty weight λ_{gp}, L_1 loss weight λ_{L1}, learning rate α, Adams momentum weights β_1 and β_2

Require: time loss weights $\lambda_{t_D}, \lambda_{t_{L1}}$

Require: initial generator weights θ and critic weights w

for $n = 1 : N$ **do**

Draw batch $\{x_t^{n_D,m}\}_{t=0,\dots,T}^{n_D=1,\dots,N_D;m=1,\dots,M}$, where $x_0^{n_D,m}$ is an empty frame

Draw a random number $\epsilon \sim U[0, 1]$.

for $n_D = 1 : N_D$ **do**

for $m = 1 : M$ **do**

$r \leftarrow 0$

for $t = 1 : T$ **do**

$x, x_{\text{prev}} \leftarrow x_t^{n_D,m}, x_{t-1}^{n_D,m}$

$\hat{x}, r \leftarrow G_\theta(x, r)$

$\tilde{x} \leftarrow \epsilon\hat{x} + (1 - \epsilon)x$

$L_D^{(m,t)} \leftarrow D_w(\hat{x}, x_{\text{prev}}) - D_w(x, x_{\text{prev}}) + \lambda_{gp}(||\nabla_{\tilde{x}} D(\tilde{x}, x_{\text{prev}})||_2 - 1)$

end for

end for

$w \leftarrow \text{ADAM}(\nabla_w \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=1}^{T} \lambda_{t_D} L_D^{(m,t)}, w, \alpha, \beta_1, \beta_2)$

end for

Set batch $\{x_t^m\}_{t=0,\dots,T}^{m=1,\dots,M} = \text{Repartion}(\{x_t^{n_D,m}\}_{t=0,\dots,T}^{n_D=1,\dots,N_D;m=1,\dots,M})$

for $m = 1 : M$ **do**

$r \leftarrow 0$

for $t = 1 : T$ **do**

$x, x_{\text{prev}} \leftarrow x_t^m, x_{t-1}^m$

$\hat{x}, r \leftarrow G_\theta(x, r)$

$L_G^{(m,t)} \leftarrow -\lambda_{t_D} D_w(\hat{x}, x_{\text{prev}}) + \lambda_{t_{L1}} \lambda_{L1} ||\hat{x} - x||_1$

end for

end for

$\theta \leftarrow \text{ADAM}(\nabla_\theta \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=1}^{T} L_G^{(m,t)}, \alpha, \beta_1, \beta_2)$

end for

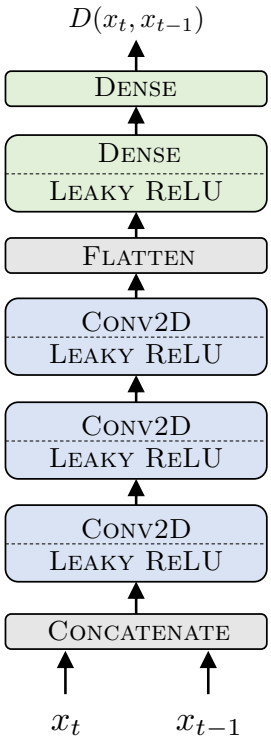

Figure 4: This is a visualization of the layers in the critic model. Predictions $\hat{x}_t$ or real samples x_t are concatenated with the previous frame x_{t-1} and then passed through convolutional layers with Leaky ReLU as activation function. This is followed by flattening the samples to the shape $[n_{\text{batch}}, T, n_{\text{pixels}} \cdot n_{\text{channels}}]$, a densely connected layer with Leaky ReLU activation and another densely connected layer. Details about filter parameters can be seen in table 2.

	Category			
Set	City	Residential	Road	Total
Training	820	2790	556	4166
Validation	11	0	0	11
Testing	16	48	30	94

Table 3: This table shows the dataset split for the Kitti dataset when using sequences of length 10.

of the observed dynamics, but it is still only possible to observe one of the possible future frames. Sampling could possibly model the differences in scene dynamics, but with only one possible output frame to calculate the loss on, it could easily lead to the model just ignoring the latent input [Isola et al., 2017].

To still be able to vary the number of times the critic is updated per generator update, the batches are divided into minibatches, with the number of minibatches being N_D. Then each critic update uses one of the minibatches and the generator update uses a mixed partition of the minibatches of the same size as a minibatch, as described in algorithm 1. The relation between the number of times the critic is updated per generator update is not what matters here, but rather the amount of data used to update the critic and generator. The critic could almost equivalently be updated with one batch and N_D times larger learning rate because sampling is not used. Either way the reason for having the training ratio is to give the critic an advantage in training.

There is one drawback with this method and that is that the generator can only be trained on $1/N_D$ part of the training dataset, which can be a big problem depending on how much data is available. The effect of using the learning rate suggested in [Gulrajani et al., 2017] $N_D = 5$ is compared to using $N_D = 1$ in the experiments. It would be interesting to try other learning rates but that will be left to future work, and instead the effect of using scheduled optimization (that is having many critic updates for each generator update) as opposed to alternating optimization.

4 Experiments

4.1 Datasets

In the experiments the Kitti dataset [Geiger et al., 2013] is used for training and the Caltech Pedestrians datasets [Dollár et al., 2009] is used for evaluation. Kitti contains 6 hours of recordings from driving in Germany with a camera on the car roof. The recordings are from diverse driving scenarios in urban, suburban and rural environments. This dataset is split into a training set, validation set and testing set. The number of samples in each set is listed in table 3. The results on this testing set can be compared to the results on Caltech Pedestrians to evaluate the generalization performance when testing on a new dataset. Caltech Pedestrians contains 10 hours of recording, with a camera mounted in the dashcam location, driving around Los Angeles. All video sequences are split into sample sequences with 10 frames in each sequence and frame rate 10 Hz. Each frame is an image of size 128x160 pixels in RGB format.

4.2 Tests

The generator model is trained on the Kitti dataset to generate next frame predictions, with five different loss settings. The main setting is that of using a combination of conditional adversarial loss and L_1 loss as in (10) with $\lambda_{L1} = 100$ as suggested in [Isola et al., 2017]. A baseline model with only L_1 loss is trained as a comparison. To see the effect of conditioning in the critic a model with no conditioning is trained with loss as in (9). A model with only adversarial loss is also trained to be able to see the effects of the L_1 term.

4.3 Evaluation Method

The best way to evaluate generative models is not always clear. In this setting there is a target frame to compare against using some metric. The metrics could be Structural Similarity Index Measure (SSIM), mean average error, mean squared error or some other per pixel metrics. These metrics have the benefit that they reward per pixel accuracy. Future frame prediction is a probabilistic

problem and the error metric should measure the difference in distribution between the generator and the real distribution, but because there is only one target in the data for each frame, any error metric can at best take into account the mean of the distribution. Another drawback of these metrics is that they reward blurry predictions because blurry images have a lower average error for misaligned images.

Another metric that can be used is to let humans evaluate if the predictions look real. Blurry images would be punished by this and humans can to some extent imagine if a frame can realistically be the frame following the previous frame. Exact alignment and high per pixel accuracy is more difficult for a human to critique. Another problem with human evaluation is the binary nature of the answer to the question of whether the samples look real or not, maybe resulting in that all fake samples are evaluated as fake. One approach to human evaluation can be found in [Villegas et al., 2017b] where human evaluation was used to compare between samples generated by different models, giving them a way to compare the quality of the different methods.

Evaluation is also application specific. If the goal is to use the generative model in a sensor function in an autonomous vehicle, then per pixel accuracy may be more important than how real the sample looks to a human, because it represents geometrically accurate information. On the other hand if the goal is to generate fake videos for social media, then human evaluation is maybe more appropriate. In this article per pixel metrics will be used, but the reader should keep the issues with these metrics in mind.

To see the performance of the critic it has to be looked at in terms of its performance during training, as that is when it is used, and it also has to be looked at in the light of how the generator is performing, because the generator trains better when the critic can provide it with a meaningful loss. The critic loss, the generator loss and the classification accuracy are important metrics, and the development of these metrics during training can be useful in determining if the generator is learning something from the critic. An ideal critic trains the generator to be an ideal generator, if the design of the generator allows it, which means that it can generate samples indistinguishable from real samples. Because the samples become indistinguishable, the critic accuracy goes to 0.5. At the same time, if a critic does not learn anything and always makes the same distinction between samples, then the critic accuracy would also be 0.5, but the critic performance would be very bad.

In normal GANs the loss is the classification accuracy of a discriminator, making it easy to interpret. For WGAN there is no way to directly measure the classification accuracy of the critic. One might be tempted to count the sign of the Wasserstein loss, counting negative loss for a real sample and positive loss for a fake sample as a correct classification, but this is not a correct interpretation. It is true that minimizing the critic loss encourages the critic to give a low wasserstein value for real samples and a high wasserstein value for fake samples, but the critic value can have an arbitrary translation built into it. In the critic loss function (8) this translation cancels because

$$\mathbb{E}\left(D(x_{t+1}, x_t) + C\right) - \mathbb{E}\left(D(\hat{x}_{t+1}, x_t) + C\right) = \mathbb{E}\left(D(x_{t+1}, x_t)\right) - \mathbb{E}\left(D(\hat{x}_{t+1}, x_t)\right). \tag{13}$$

Because of this cancellation the sum $\mathbb{E}\left(D(x_{t+1}, x_t)\right) - \mathbb{E}\left(D(\hat{x}_{t+1}, x_t)\right)$ can be interpreted as the performance of the critic. When this value is zero, it must be that wasserstein loss is same for real and fake samples, implying that the critic makes no distinction between the samples. Likewise, when this value is negative, the critic succesfully makes some distinction between real and fake samples. The value should not be positive but could become by chance. In practice the full critic loss, including the gradient penalty term, can be used equivalently, because the gradient penalty should be small compared to the wasserstein loss. If the gradient penalty term grows large, there are problems with either exploding or vanishing gradients.

The critic is a binary classifier with an unknown classification threshold. A useful way to analyze binary classifiers with unknown thresholds is a Receiver Operating Characteristics (ROC) curve. In a ROC curve the true positive rate (tpr) is plotted against the false positive rate (fpr) for different classification thresholds. True positive rate is the rate of correctly classified real samples and false positive rate is the rate of incorrectly classified fake samples.

5 Results

In this section the performance of the different models will be presented in terms of the evaluation metrics on the testing dataset. Additionaly the performance of the models during training, on the validation split of the training dataset, is presented.

epoch	mean absolute error	critic wasserstein loss
25	0.03181	-6.5055
50	0.02600	-7.8402
75	0.02416	-7.6461
100	0.02276	-7.7055
125	0.02311	-5.9373
150	0.02213	-6.3951
175	0.02100	-7.9216
200	0.02043	-7.4577
225	0.02021	-7.0891
250	0.01945	-7.8119
275	0.01974	-7.3401
300	0.01877	-7.4514
325	0.01856	-3.4960
350	0.01799	-6.5167
375	0.01811	-6.9934
400	0.01757	-7.9941

Table 4: This table shows the validation loss after every 25th epoch in terms of mean absolute error and wasserstein loss for the critic.

5.1 Supplementary Material

Supplementary material of full resolution animated samples and code for the project can be found at https://bitbucket.org/MagnusWallgren/exjobb/src/master/.

5.2 Training

The purpose of GAN is to be used during training. Here different parameters in the model are looked at in terms of training performance. The parameters are number of epochs in training, training ratio N_D and loss function. The parameter space is large in terms of how much time it would take to train enough models to explore the whole parameter space. Instead a standard model is used and each parameter is varied one-by-one to explore how the training behavior changes with different parameter changes.

5.2.1 Increasing Training Time

Increasing the training time can lead to increased performance because the weights are updated for more iterations, but it can also lead to decreased performance because of overtraining. Overtraining means that the model is trained to have a low loss on training data but does not generalize well to other data. To investigate the robustness against overtraining two models with conditional WGAN and L_1 loss and $N_D = 5$ were trained for 250 and 400 epochs respectively. Critic and generator validation losses are shown in figure 5 and listed in table 4 for the model trained for 400 epochs. Table 5 lists the evaluation metrics on the test dataset for both models.

The validation mean absolute error decreases for the whole training duration for the 400 epoch model, which suggests that there is no overtraining in the generator. The performance on the test set is better for the model trained for 250 epochs, which does not agree well with the consistent decrease in validation loss for the model trained for 400 epochs. This may be due to variation in weight initialization leading to the 250 epoch model finding a better minimum than the 400 epoch model. Seeing that variations such as weight initialization makes a bigger difference than increasing training time from 250 to 400 epochs, it can safely be said that the increased performance from training more than 250 epochs is not large. There is a trade-off between performane and training time and training the model for 250 epochs already takes 28 hours on two GPUs. How long the training time should be depends on the application. For models that need to be fine-tuned, going over 250 epochs may be important but otherwise it does not seem useful. Even training two different models and choosing the one with lower test error could be a better approach than training for more than 250 epochs. Even better could be to train an ensemble network, which is a weighted sum of models that are trained jointly.

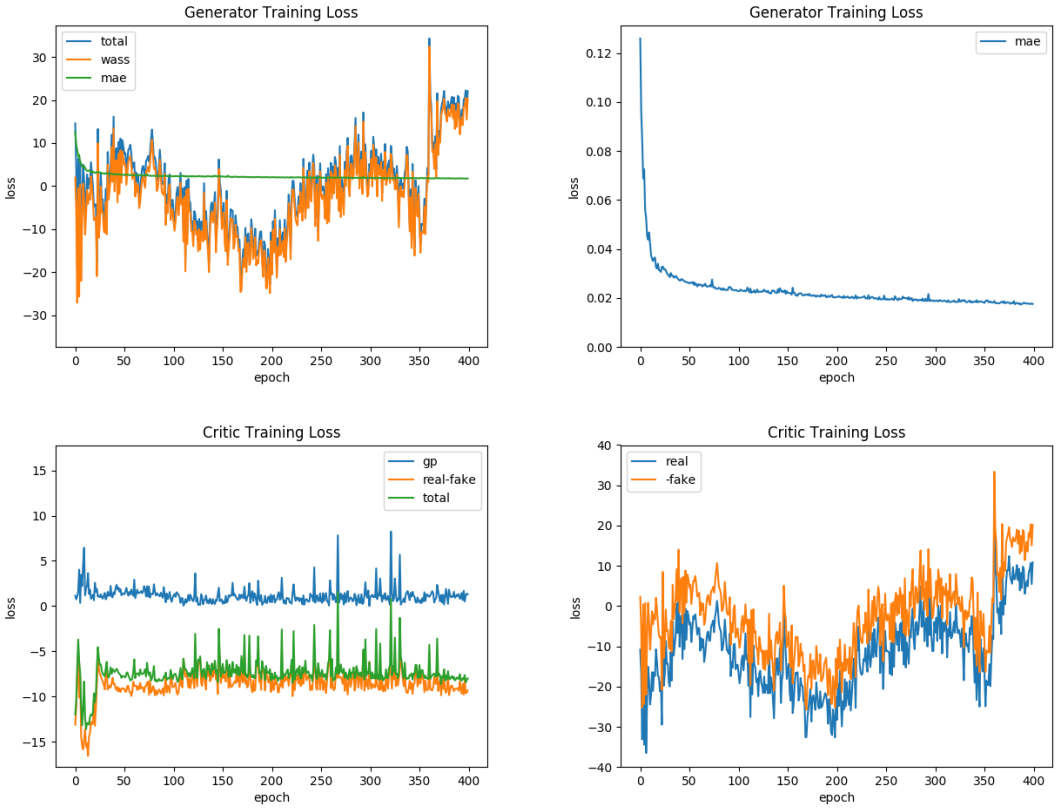

Figure 5: This figure shows the generator and critic validation loss during training, at the end of each epoch for 400 epochs, as means over batches of validation samples. The generator losses (top) are given as the terms in the generator loss in (10). In the top left figure `wass` is the Wasserstein loss on generated samples and `mae` is L_1 loss, including the loss weight $\lambda_{L1} = 100$. In the top right figure the L_1 loss is shown without the scaling weight λ_{L1}. The critic loss (bottom left and bottom right) is given as the terms in the critic loss in (8), including the loss weights. `real-fake` is the difference between the Wasserstein loss on fake and real samples $D(x) - D(\hat{x})$, `gp` is the gradient penalty and `total` is the total critic loss. `real` and `fake` are critic values on real and fake samples, $D(x)$ and $-D(\hat{x})$, respectively.

Loss	Parameters		Pedestrians			Kitti		
	epochs	N_D	SSIM	MSE	MAE	SSIM	MSE	MAE
CWGAN + L_1	250	5	0.9028	0.003939	0.03311	0.8580	0.004431	0.03497
CWGAN + L_1	400	5	0.8788	0.004828	0.03709	0.8188	0.005753	0.04054
L_1	250	5	0.9257	0.002954	0.02607	0.8946	0.003302	0.02816
CWGAN	250	5	0.7535	0.01136	0.06011	0.6664	0.01492	0.06937
WGAN + L_1	250	1	0.8861	0.004394	0.03409	0.7844	0.006875	0.04411
CWGAN + L_1	250	1	0.9018	0.003879	0.03388	0.8510	0.004745	0.03688
Previous Frame			0.8581	0.006969	0.03474	0.6965	0.01435	0.05912

Table 5: This table shows performance of generators trained with different losses as means over about 1000 samples. CWGAN is short for Conditional WGAN and WGAN is WGAN without conditioning. SSIM is Structural Similarity Index Measure, MSE is Mean Square Error and MAE is Mean Absolute Error. All models use training ratio $N_D = 5$ except for the model in the bottom row and all models are trained for 250 epochs except for the model in the second row. The model with only L_1 loss performs best on all metrics and the model with only WGAN loss performs worst.

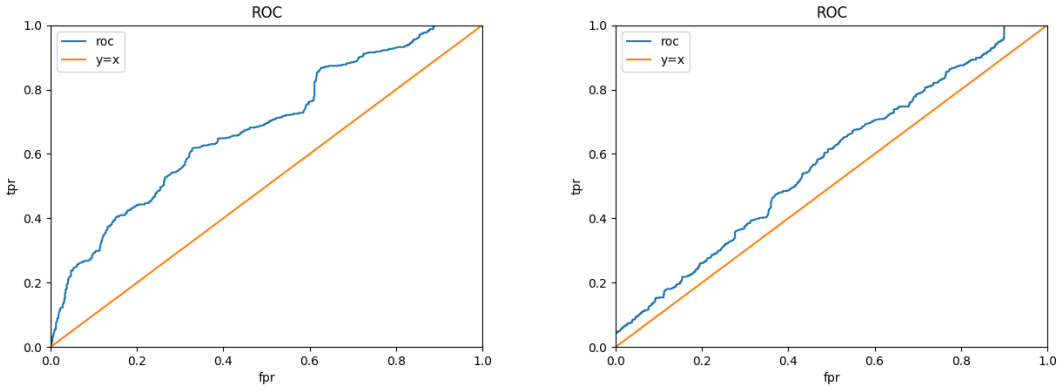

Figure 6: This figure shows ROC curves for two different critics. The true positive rates and false positive rates are evaluated on about 1000 samples at the end of training. The left plot shows ROC for a critic trained without conditining and the right plot shows ROC for a critic trained with conditioning. The critic without conditioning is clearly better than the critic with conditioning, with higher tpr and lower fpr for most thresholds. Both models use training ratio 1 and are trained for 250 epochs.

5.3 Testing

5.3.1 Weighting of Critic

In the figure it can be seen that the L_1 term in the generator loss is outweighed by the critic loss, even though the L_1 term is weighted by $\lambda_{L1} = 100$. This is expected to happen because, even with the loss weight, the L_1 loss is small. The maximum possible distance between two pixels is 1 with pixels value in the range $[0, 1]$ or 100 if including the loss weight. On the other hand the critic loss can be arbitrarily large, because of the canceled translation and in the figure it varies between -30 and 5. The L_1 loss decreases with increasing epoch, indicating that the generator learns to minimize the pixel distance well.

In the critic loss the total loss varies in the range $[-14, -4]$ for the first 25 epochs and then stabilizes to the range $[-8, -5]$ for the rest of the training. The important thing to note here is that the total loss is always negative. This means that the critic outperforms the generator, in that it is able to distinguish between real and fake samples better than guessing. From the 25th epoch the total critic loss only has small variations and can neither be said to be increasing or decreasing. This could be because the critic and generator have converged and the generator is unable to learn anything more from the critic, because of some limit in the design or training. This does not seem to be the case because the generator L_1 loss keeps decreasing. Another possibility is that the critic and generator are learning at the same rate and that neither of them have converged, but the critic stays ahead of the generator. The gradient penalty in the critic is small, indicating that there is no problem with vanishing or exploding gradients.

5.3.2 Conditioning the Critic

To see the effect of conditioning the critic on the previous frame two models are trained, one where the critic is conditioned on the input frame using the loss function in (8), and one where the critic is not conditioned on the input frame using the loss function in (7). The ROC curve of the critics are shown in figure 6 and the area under the curves are listed in table 6.

The area under the ROC curve is larger for the critic without conditioning and from the shape of the ROC curves it is apparent that the critic without conditioning consistently has higher true positive rate and lower false positive rate for different classification thresholds. The critic performance should always be looked at in relation to the generator performance because a better generator makes distinguishing between samples more difficult for the critic. The performance of the generators is better for the conditioned model in terms of per pixel error as can be seen in table 5. The question then is if the better critic performance of the model without conditioning has any benefit that is not measured in per pixel error. Looking at the the samples in the supplementary material it seems that the critic without conditioning produces samples that are sharper but contain

Loss	N_D	ROC curve area	
		Pedestrians	Kitti
CWGAN + L_1	5	0.6684	0.7462
CWGAN	5	0.6052	0.6402
WGAN + L_1	1	0.6808	0.6772
CWGAN + L_1	1	0.5740	0.5885

Table 6: This table lists the area under the Receiver Operating Characteristics (ROC) curve for some different critics. Here CWGAN stands for Conditional WGAN and WGAN is WGAN without conditioning. N_D is the training ratio. ROC curve plots can be seen in figures 6, 7 and 8

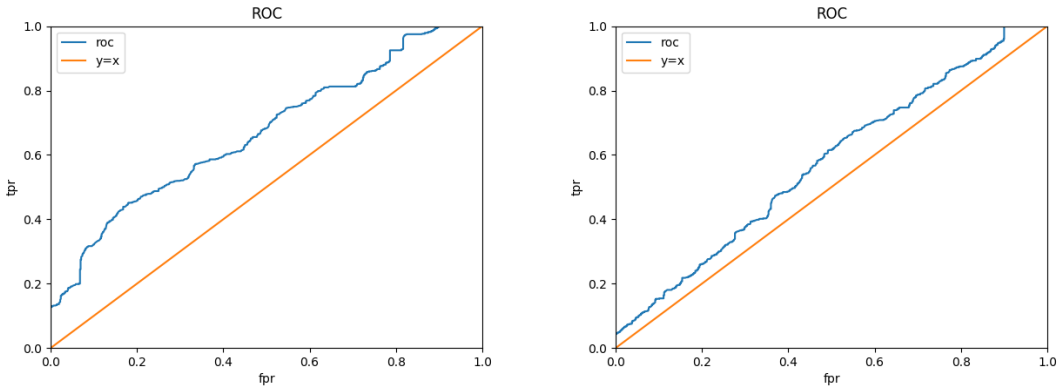

Figure 7: This figure shows ROC curves for two different critics. The true positive rates and false positive rates are evaluated on about 1000 samples at the end of training. The left plot shows ROC for a critic trained with training ratio $N_D = 5$ and the right plot shows ROC for a critic trained with $N_D = 1$. Both critics are conditional.

some visual artifacts, such as flickering colors in parts with large amounts of motion.

5.3.3 Training Ratio

To see the effect of using different training ratios N_D two models are trained, one with $N_D = 1$ and one with $N_D = 5$. Their ROC curves are shown in figure 7 and the corresponding area under the curves are listed in table 6. The performance of the generators for each of the models are listed in 5. The generator performance of the model trained with $N_D = 5$ is consistently better than the model with $N_D = 1$. The critic trained with training ratio $N_D = 5$ has better critic performance as can be seen in the ROC curve. The critic being better for higher training ratio is what is expected, since a higher training ratio is meant to give the critic an advantage in training. There is no clear difference in the visual quality of the output sequences in the supplementary material.

5.3.4 Combining GAN and L_1 Loss

GAN loss and L_1 loss can be combined arbitrarily or used individually, leaving three different combinations, only GAN loss, only L_1 loss or GAN and L_1 loss combined. Here one each of these models are trained, with the GAN models being trained with training ratio 5 and using conditioning. The combined model weights the L_1 loss term with $\lambda_{L1} = 100$ as in (8). This may seem like a large loss weight, but the L_1 loss is small and the critic can be scaled in training to suppress the L_1 loss. This weight is the same as suggested in [Isola et al., 2017], in which they use the classic GAN architecture, while here it is used with WGAN.

The generator performance in evaluation is listed in table 5. The ROC curves of the critics are shown in figure 8 and the area under the curves are listed in table 6. The generator performance is best for only L_1 loss and worst for only WGAN loss, notably it is worse than using the previous frame as prediction. For the models that use GAN, the model that uses the combined loss has a higher true positive rate and the GAN only model has lower false positive rate for their respective optimal threshold. The area under the ROC curve is larger for the combined model. From this

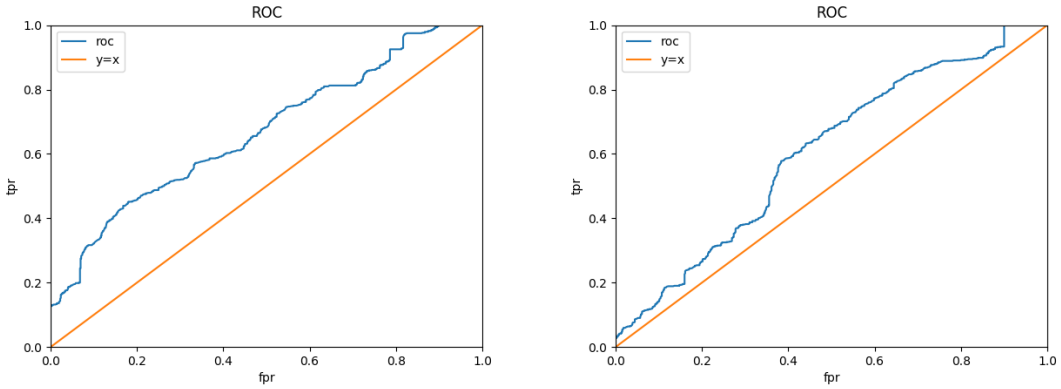

Figure 8: This figure shows ROC curves for two different critics. The true positive rates and false positive rates are evaluated on about 1000 samples at the end of training. The left plot shows ROC for a critic in a model trained with a combination of GAN and L_1 loss and the right plot shows ROC for a critic in a model trained with only GAN loss. Both critics are conditional and use training ratio $N_D = 5$.

it is not immediately clear if the critic is better for the combined model, but from taking into account the poor generator performance for the GAN-only model it is expected that the GAN-only critic should have a better performance. Therefore it seems that the combined critic has better performance than the GAN-only critic, which is notable because the only differences between the models lie in the generator loss function and not in the critic.

5.4 Output Sequences

Figure 9 shows sequences of generated samples with the real sequence. Table 5 shows the performance in evaluation for the different models.

6 Discussion

6.1 Model selection

The generator model selection that has been explored in this report is the choice between L_1 loss, WGAN or a combination of the two as loss function. The critic model has the options between using conditioning or not and what training ratio to use. For the generator it is clear that adding WGAN to the loss makes the generator worse in terms of per pixel error. Other than that it is not apparent if the addition of WGAN loss has the supposed benefit of generating less blurry samples. It looks like it does have this effect on the output sequences for the model without conditioning, but without any objective method of testing it, this conclusion cannot be drawn. It can be concluded that using WGAN loss only, without adding L_1 loss, results in low performance in terms of per pixel error and output sequences that do not make correct predictions, even having worse performance than using last frame as prediction.

The two best performing critics are the critic with conditional WGAN loss and training ratio $N_D = 5$ and the critic without conditioning and training ratio $N_D = 1$. With higher training ratio it is expected that the critic performs better, and comparing the conditioned critics with $N_D = 1$ and $N_D = 5$ shows that this is the case. Whether this results in better generator performance cannot be answered without having some way of evaluating the performance other than per pixel error. Seeing that $N_D = 5$ worked better than $N_D = 1$ for the conditioned model and that not using conditioning led to better critic performance for the critics with training ratio $N_D = 1$, it would be interesting to see if $N_D = 5$ without conditioning is the best parameter choices for the critic.

Other than performance in terms of output error, the critics are not equally efficient in computation. It is desirable to be able to reduce the long training time and in this aspect the critic model selection matters. Conditioning the critic means that the input data is twice as large and therefore twice as much data has to uploaded with each batch, which could have a negative impact

Conditional WGAN + L_1 loss with training ratio 5

Conditional WGAN + L_1 loss with training ratio 5, 400 epochs

Unconditional WGAN + L_1 loss with training ratio 5

Conditional WGAN loss with training ratio 5

L_1

Conditional WGAN + L_1 loss with training ratio 1

Figure 9: This figure shows generated sequences (bottom) along with real sequences (top). The images have been downsampled. More examples with full resolution, animated examples and side by side comparisons can be seen at https://bitbucket.org/MagnusWallgren/exjobb/src/master/

on efficiency [Chollet et al., 2015]. With this in mind it might be better to not use conditioning when efficiency is important. Using L_1 loss only for the generator has a much higher impact on computational performance, because the critic does not have to be trained at all.

6.2 Limitations in Data

One problem that was presented in section 3.1 is that the future frame prediction problem is the problem of estimating a probability density of possible future frames given past frame, but with natural video sequences only one future frame from the distribution is available for each sequence of past frames, and as such the generator can learn the mean of the distribution at best. A way to combat this could be to use some way of artificially generating frames. This could be by generating the entire sequences artificially or by using some data augmentation technique. It would be important that the generated data actually captures the distribution if the model should be able to be transferred to a problem on real data, but the problem to begin with is that the distribution is unknown.

7 Conclusion

This report has explored using Wassersteing GAN with predective coding networks for future frame prediction in digital videos. The best model, in terms of per pixel error, using adversarial loss is the conditioned WGAN model with training ratio 5, but this model does not perform better than just using L_1 loss. Other than per pixel error, it is difficult to evaluate the quality of generated samples without adding subjective judgement, which makes it unclear if there are other benefits of using adversarial loss.

References

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

[Byeon et al., 2017] Byeon, W., Wang, Q., Srivastava, R. K., and Koumoutsakos, P. (2017). Fully context-aware video prediction. *CoRR*, abs/1710.08518.

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras. https://keras.io.

[Dollár et al., 2009] Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2009). Pedestrian detection: A benchmark. In *CVPR*.

[Friston and Kiebel, 2009] Friston, K. and Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 364(1521):1211–1221.

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

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[Gulrajani et al., 2017] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *CoRR*, abs/1704.00028.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976. IEEE.

[Jang et al., 2018] Jang, Y., Kim, G., and Song, Y. (2018). Video prediction with appearance and motion conditions.

[Kalchbrenner et al., 2016] Kalchbrenner, N., Oord, A. v. d., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2016). Video pixel networks. *arXiv preprint arXiv:1610.00527*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Liang et al., 2017] Liang, X., Lee, L., Dai, W., and Xing, E. P. (2017). Dual motion GAN for future-flow embedded video prediction. *CoRR*, abs/1708.00284.

[Lotter et al., 2015] Lotter, W., Kreiman, G., and Cox, D. D. (2015). Unsupervised learning of visual structure using predictive generative networks. *CoRR*, abs/1511.06380.

[Lotter et al., 2016] Lotter, W., Kreiman, G., and Cox, D. D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. *CoRR*, abs/1605.08104.

[Mathieu et al., 2015] Mathieu, M., Couprie, C., and LeCun, Y. (2015). Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440.

[Srivastava et al., 2015] Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852.

[Villegas et al., 2017a] Villegas, R., Yang, J., Hong, S., Lin, X., and Lee, H. (2017a). Decomposing motion and content for natural video sequence prediction. *CoRR*, abs/1706.08033.

[Villegas et al., 2017b] Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., and Lee, H. (2017b). Learning to generate long-term future via hierarchical prediction.

[Xue et al., 2016] Xue, T., Wu, J., Bouman, K. L., and Freeman, W. T. (2016). Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. *CoRR*, abs/1607.02586.

[Zhong et al., 2018] Zhong, J., Ogata, T., and Cangelosi, A. (2018). Encoding longer-term contextual multi-modal information in a predictive coding model. *CoRR*, abs/1804.06774.

Master's Theses in Mathematical Sciences 2019:E16
ISSN 1404-6342

LUTFMA-3378-2019

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/