A Survey of Autoregressive Models for Image and Video Generation

Saqib Azim
UC San Diego
sazim@ucsd.edu

Mehul Arora
UC San Diego
mearora@ucsd.edu

Narayanan Elavathur Ranganatha
UC San Diego
nelavathurranganatha@ucsd.edu

Mahesh Kumar
UC San Diego
ar223@ucsd.edu

Abstract

This survey paper offers a comprehensive overview of recent advances in autoregressive (AR) models for image and video generation. It discusses state-of-the-art AR models like PixelCNN [34], PixelRNN [34], Gated PixelCNN [26], and PixelSNAIL [5], emphasizing their unique architectures and contributions. The main challenge in AR models, handling long-range dependencies effectively, is addressed through various approaches, such as gated activations, self-attention mechanisms, and residual blocks. The paper presents Locally Masked Convolution [15] and Autoregressive Diffusion Models [1, 13] as examples of order-agnostic approaches, improving upon traditional autoregressive models. Transformer-based networks [9, 23, 43] are explored for autoregressive image generation, showcasing superior performance in image quality and synthesis tasks. Quantization-based models [44] enhance image diversity and quality through feature quantization and variational regularization. The paper then discusses Autoregressive modeling in pixel space [17, 25, 40] and latent space [24, 28, 38, 41] for video generation. The paper concludes by discussing the strengths, limitations, and future research directions in autoregressive models for image and video generation, providing valuable insights for researchers and practitioners.

1. Introduction

Generative models are crucial in unsupervised learning as they allow for the generation of realistic and diverse samples from complex data distributions. These models aim to learn the empirical distribution of training data and generate new images or samples that resemble the dataset. However, estimating the distribution of natural images, which are high-dimensional and unstructured, presents significant challenges. The task involves striking a balance between building complex and expressive models while ensuring tractability and scalability.

Deep generative models for images can be categorized into three main types: Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Autoregressive (AR) Models. VAEs approximate a latent space and generate diverse images by sampling from the learned latent space. GANs, on the other hand, produce sharp, high-resolution images, but there is no guarantee that they learn the entire data distribution effectively, potentially leading to incomplete coverage of the distribution.

AR models have emerged as a promising approach for capturing the conditional distribution of high-dimensional and structured data, such as images and videos. Unlike GANs, AR models guarantee capturing the entire data distribution, resulting in a diverse set of generated samples. Neural AR models offer tractable likelihood computation, ease of training, and have shown superiority over latent variable models. However, AR models face limitations in handling high-resolution images due to growing memory and computation requirements.

In recent years, the focus on autoregressive models has expanded to include video generation. Autoregressive video generation aims to generate realistic and coherent video sequences by predicting each frame based on previously generated frames. This task is particularly challenging as it requires not only accurate image generation but also the modeling of the physical dynamics and temporal dependencies within the video. Autoregressive video generation models have been developed to address these challenges, exploring strategies such as atomic-level modeling of pixels across time and space or autoregressive modeling in a downsampled latent space.

2. Autoregressive Generative Models

An Autoregressive (AR) model is usually used in the context of time-series modeling where it poses density estimation as a sequence modeling task in order to model the conditional distribution over the next element conditioned on all previous elements. AR models can also be used to generate spatial data (e.g., images, videos) by imposing some temporal ordering on the input spatial data. An effective approach used by autoregressive models is to treat an $n \times n$ image x as a sequence of random variable pixels denoted by $x_1, x_2, \ldots, x_{n^2}$ and represent the joint distribu-

tion of image pixels $P_{\text{data}}(x)$ as the product of conditional probabilities of all (or a subset) of previous pixels as shown in below equation 1.

$$P_{\text{data}}(x) = P(x_1, x_2, \ldots, x_{n^2}) = \prod_{i=1}^{n^2} P(x_i | x_1, \ldots, x_{i-1})$$

This factorization converts the joint modeling problem into a sequence problem. Every pixel depends on all the pixels above and to the left of it, and not on any pixel not yet predicted. To calculate these conditional likelihoods $P(x_i|x_1, \ldots, x_{i-1})$ in a tractable and scalable way, one can use universal approximators like deep neural networks. There have been a number of efforts in the past years to use deep AR models to sequentially predict image pixels.

Initial contributions to tractably compute the joint distribution as a product of conditionals were provided in fully visible neural networks [2] and Neural Autoregressive Distribution Estimator (NADE) [32]. Autoregressive models have also been pivotal in the recent success and rapid development of Large Language Models (LLM) following the creation of Transformers [37]. Autoregressive models for images started taking off most notably with PixelRNN and PixelCNN [34] probabilistic generative models.

The generation of images and videos poses unique challenges due to their intricate spatial dependencies and rich semantic content. Autoregressive models address these challenges by factorizing the joint distribution of pixels using conditional probabilities. Unlike other generative models like Variational Autoencoders (VAEs) or Generative Adversarial Networks (GANs), autoregressive models explicitly model the dependencies between pixels and generate samples sequentially, one pixel at a time. These networks have the advantage of returning explicit probability densities, unlike alternatives such as GANs, making it straightforward to apply in domains such as compression and probabilistic planning and exploration.

3. Autoregressive Image Generation

3.1. Order Dependent Autoregressors

PixelRNN [34], one of the pioneer works introduced in 2016, is composed of 2-D LSTM layers which capture the information from previously generated pixels to model the current pixel distribution. RNNs have been shown to be extremely efficient in handling sequence problems. The network scans the image one row and one pixel at a time in each row and predicts conditional distribution $P(x_i|x_1, \ldots, x_{i-1})$ of the current pixel over 256 possible intensity values. Each pixel x_i is in turn jointly determined by 3 color channels (RGB) which are modeled successively and conditioned on the other color channels as well as on all previously generated pixels i.e., B conditioned on (R, G), and G conditioned on R. Thus, the conditional distribution of pixel x_i given previously predicted pixels can be expressed as follows:

$$P(x_i|x_{<i}) = P(x_i^R|x_{<i})P(x_i^G|x_{<i}, x_i^R)P(x_i^B|x_{<i}, x_i^R, x_i^G)$$

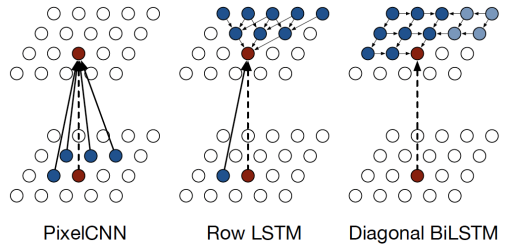

Figure 1. Visualization of the input-to-state and state-to-state mappings for the three proposed architectures. (source: [34])

PixelRNN has two variants based on LSTM layer types - Row LSTM (convolution applied along each row) and Diagonal BiLSTM (convolution applied along image diagonals). The Row LSTM is a unidirectional layer that processes images row by row, computing features for each row using 1-D convolutions. The layer captures a triangular context (receptive field) above each pixel, shown in Figure 1 (middle), and hence is unable to capture entire context. Diagonal BiLSTM scans the image diagonally, starting from one corner and moving towards the opposite corner computing the LSTM state along each diagonal step, and therefore captures the entire context as shown in Figure 1 (right). To facilitate this process, the input map is skewed to enable diagonal convolutions which are then used to compute the input-to-state and state-to-state components of the Diagonal BiLSTM. The network also incorporates residual (or skip) connections [11] from one LSTM layer to the next to improve the convergence speed and propagate signals more directly through the network.

In PixelCNN [34], every conditional distribution $P(x_i|x_1, \ldots, x_{i-1})$ is modeled by a CNN that takes as input an image and outputs a 256-way probability distribution for each (sub)-pixel (R,G,B) in a sequential manner. To ensure CNN only uses information from pixels above and to the left of the current pixel, the convolution filters are masked as shown in Figure 3 by zeroing out the filter weights connecting to future pixels not yet predicted. This is achieved by splitting the feature maps at every layer of the network into three channels (R,G,B) and using either Mask A or B depending on the layer as shown in Figure 3 (right).

PixelCNN architecture is described in Figure 2 and is trained by maximizing a tractable log-likelihood function of the training data. Inference in PixelCNN is sequential

PixelCNN	Row LSTM	Diagonal BiLSTM
7×7 conv mask A		
Multiple residual blocks: (see fig 5)		
Conv 3×3 mask B	Row LSTM i-s: 3×1 mask B s-s: 3×1 no mask	Diagonal BiLSTM i-s: 1×1 mask B s-s: 1×2 no mask
ReLU followed by 1×1 conv, mask B (2 layers)		
256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)		

Figure 2. PixelCNN and PixelRNN Architecture. i-s and s-s stands for input-state and state-state convolutions (source: [34])

Figure 3. PixelCNN uses masked convolution layers to ensure that (sub)-pixel i only uses past (sub)-pixels. For input layer (Mask A), "B" sub-pixel is connected to "G" and "R" of input pixel i and all previous pixels (li), "G" sub-pixel is connected to "R" and all previous pixels (li), and "R" is connected to only previous pixels (li). For convolution layers other than the first one (Mask B), any sub-pixel output from a convolution layer has already been masked from its corresponding input sub-pixel. (source: [34])

i.e., generated pixel-by-pixel. First, an all-zero image is passed to the model to predict the distribution of the first (top-left) pixel which is modeled independently of other pixels. Given the distribution, we sample to get a realization for the first pixel and update our image. Each subsequent pixel distribution is generated in sequence conditioned on all previously sampled pixels. This process is repeated until the entire image is generated. The sampling process is relatively slow compared with other generative models such as VAE or GANs, where all pixels are generated in one go. However, recent advances use cached values to reduce the sampling time.

PixelRNNs generally outperform PixelCNNs, but the latter is faster to train due to the parallelizability of convolutions, which is advantageous for large image sizes. One possible reason for this advantage is that LSTM's recurrent connections enable each layer to access the entire neighborhood of previous pixels, whereas the available neighborhood region for PixelCNN grows linearly with the depth of the convolutional stack. However, this limitation can be mitigated by using more layers, as described in Gated PixelCNN. During training and evaluation, the distributions over pixel values are computed in parallel, while image generation during inference is sequential.

Gated PixelCNN [26, 35] combines the strengths of PixelRNN and PixelCNN by using masked convolutions with LSTM gates, achieving performances comparable to PixelRNN on CIFAR-10 and ImageNet datasets while training as fast as PixelCNN. Feed-forward neural networks with gates have been explored in previous works, such as highway networks [30], grid LSTM [16] and neural GPUs [47], and have generally proved beneficial to performance. Taking inspiration from it, the authors replace PixelCNN's ReLU activations between the masked convolutions with the following gated activation unit (hence named Gated PixelCNN), similar to LSTM gates in PixelRNN, which helps it to model more complex interactions.

$$y = \tanh(W_{k,f} * x) \odot \text{sigmoid}(W_{k,g} * x) \qquad (1)$$

PixelCNNs have a blind spot in their receptive field that cannot be used to make predictions. This paper removes that blind spot by combining two convolutional network stacks: a horizontal stack conditioning on the current row so far and vertical stack conditioning on all rows above whose outputs are combined after each layer. Every layer in the horizontal stack takes as input the output of the previous layer as well as that of the vertical stack. If we had connected the output of the horizontal stack into the vertical stack, it would be able to use information about pixels that are below or to the right of the current pixel which would break the conditional distribution.

It also introduces a conditional variant of the Gated PixelCNN (Conditional PixelCNN) that models the complex conditional distributions of images given a latent vector embedding. Gated PixelCNN outperforms the PixelCNN, and performs comparably to PixelRNN.

PixelCNN++ [27], introduced by OpenAI, essentially replaces PixelCNN with substantial improvements. It contains a number of modifications (listed below) to the original PixelCNN [34] that simplify its network structure and improves performance.

- Instead of using a 256-way softmax for the conditional distribution, it uses a discretized logistic mixture likelihood that improves training speed and allows for better handling of sub-pixel values.

- PixelCNN models the generative process based on individual color channels. The authors claim the complexity arising from separate modeling of subpixels is unnecessary and simplify the modeling of dependencies between color channels by conditioning output joint distribution over all 3 channels of a predicted pixel.

- PixelCNN employs convolutions with a small receptive field, which captures local dependencies but may not adequately model long-range structures. To alleviate this limitation and enhance the generated image quality, it incorporates downsampling using stride 2 convolutions, thus reducing computational costs compared to dilated convolutions while preserving multi-resolution processing.
- This method enhances the information flow and model structure by incorporating additional shortcut connections between specific layers, resembling the architecture of VAE and U-net models. In addition, dropout regularization is employed to prevent overfitting, leading to improved training and the generation quality.

PixelCNN++ is evaluated on the CIFAR-10 dataset and produces better results as compared to PixelRNN, PixelCNN, and Gated PixelCNN in terms of log-likelihood.

PixelSNAIL [5] incorporated the ideas of attention mechanisms into the architecture of PixelCNN++ and made several other changes to its structure.

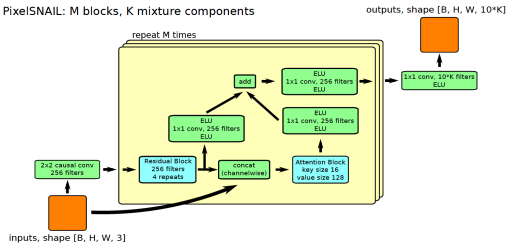

Figure 4. PixelSNAIL Architecture

The main challenge in previous methods is the ability to model long-range dependencies effectively. PixelSNAIL addresses this challenge by combining causal convolutions with self-attention mechanisms to estimate the probability density. Causal convolutions provide high-bandwidth access to earlier parts of the sequence, while self-attention allows infinite access to information far away in the sequence. By interleaving these two, PixelSNAIL achieves high-bandwidth access without constraints on the amount of information it can effectively use.

Its architecture builds off the architecture of PixelCNNs and consists of two building blocks as shown in Figure 4, namely residual blocks and attention blocks. Residual blocks consist of multiple 2D convolutions each with residual connections and gated activations. They are masked to ensure current pixel can only access pixels to the left and above it. Attention blocks perform key-value lookups by projecting the input to lower dimensions and utilizing softmax-attention with masking to ensure causality.

PixelSNAIL outperforms previous autoregressive models such as PixelRNN and PixelCNN [34], PixelCNN++ [27], and Image Transformers [23] in terms of negative log-likelihood on CIFAR-10 and ImageNet 32×32 datasets, which suggests that both causal convolutions and attention block are essential components of this architecture. One drawback is that due to the sequential sampling of each pixel, the sampling speed is comparable to previous autoregressive models.

3.2. Order-Agnostic Autoregressive Models

Order-Agnostic Autoregressive Models generate variables in a random order $\sigma \in S_D$ where S_D is the permutation of all integers $1, 2 \ldots D$. The log-likelihood for such a model is given by:

$$\log p(x) \geq \mathbb{E}_{\sigma \sim \mathcal{U}(S_D)} \sum_{t=1}^{D} \log p(x_{\sigma(t)} | x_{<\sigma(t)}) \qquad (2)$$

This can then treated as a latent variable model and therefore the log-likelihood can be written as:

$$\log p(x) = \log \mathbb{E}_{\sigma \sim \mathcal{U}(S_D)} p(x|\sigma) \geq \mathbb{E}_{\sigma \sim \mathcal{U}(S_D)} \log p(x|\sigma) \qquad (3)$$

A expected lower-bound can be derived for the log-likelihood which is then used to train the models as:

$$\log p(x) \geq \mathbb{E}_{t \sim \mathcal{U}(1,\ldots,D)} [D \cdot \mathcal{L}_t]$$

$$\mathcal{L}_t = \left[\frac{1}{D-t+1} \mathbb{E}_{\sigma \sim \mathcal{U}(S_D)} \sum_{k \in \sigma(\geq t)} \log p(x_k | x_{\sigma(\leq t)}) \right]$$

3.2.1 Locally Masked Convolutions

LMConv [15] attempts to approximate the joint pixel distribution by minimizing KL-divergence $D_{KL}(P_{\text{data}} || P_\theta)$ or the log-likelihood of the samples. The PixelCNN family uses masked convolutions to control information flow by setting certain weights of convolution filters to 0. One of the drawbacks of PixelCNN is it can only model a single distribution of the joint density by scanning in a specific raster ordering. For tasks such as image completion, these models are unable to use much of the observed context. LMConv generates data in arbitrary order by applying arbitrary masks to the weights at each image location using locally masked convolutions (modification to the standard convolution), providing flexibility similar to NADE [32] and parallelizability similar to MADE and PixelCNN. This allows control over the generation order and parallel computation of conditionals for evaluating likelihood. The network transforms input images into tensors of log-probabilities that define conditional distributions. The paper also explores different image generation orders, such as raster scan, S-curve order, and Hilbert space-filling curve order, and shows that LMConv provides flexibility in modeling image generation orders and improves the likelihood

estimates over methods like PixelCNN++ by averaging joint distributions obtained from multiple orderings during inference.

3.2.2 Autoregressive Diffusion Models (ARDM) [13]

This paper proposes a model that combines Order-Agnostic Autoregressive models [33] and absorbing discrete diffusion [1]. We outline the autoregressive approach used and do not mention the diffusion methodology used. This model masks variables at the input and predicts those at the output to simulate the sampling of different σ as stated above. Let us say we have $x \in \mathcal{X} = 1, 2, \ldots, K^D$ representing the discrete variables with K classes. In our case, this would be the pixels or patches of the image. Let us sat we have a neural network $f : \mathcal{X} \rightarrow \mathbb{R}^{D \times K}$ which outputs probability vectors. For a given permutation array σ, the elmentwise comparison $\mathbf{m} = \sigma < t$ which produces a boolean mask which is used to mask out a set of inputs and train.

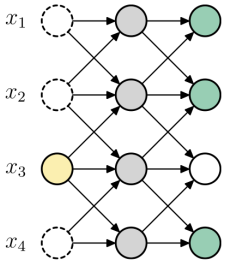

Figure 5. A single training step of ARDM [13]

The model predicts the distributions over multiple variables($x_{\sigma(t+k)}$) while conditioning over $x_{\sigma(<t)}$ at the same time. This allows the model to be parallelized while estimating the image.

3.3. Transformer Based Networks

Most of the previously described models process the image data sequentially which loses the global context. To attend to the global context along with the local context, further research has been conducted. With the great success of transformer-based models in language modeling, transformer-based architectures were also tested in the field of computer vision, especially generative modeling. To this end, Niki et al. proposed Image Transformer [23] for autoregressive image generation. The Image transformer architecture takes inspiration from the Transformer model architecture, which was originally designed for sequence-to-sequence tasks in natural language processing.

Image Transformer models the problem of image generation as a sequence modeling problem where it learns the joint distribution of pixels. Each pixel is predicted based on the previously predicted pixels following the autoregressive strategy. One of the main components of the image transformer is the self-attention layer which allows it to capture both global and local context as done in the language modeling process. However, the authors propose to use self-attention to attend to local neighborhoods which increases the size of the receptive field when compared to a neural network, and the model can now also cater to images of higher resolution. It uses negative log-likelihood metric to train the model on generative tasks like image completion and super-resolution. The model performance opens up a new domain of transformer-based autoregressive models and serves as proof that this architecture can attain state-of-the-art performance.

Building on the transformer-based architecture, Esser et al proposed ImageBART [9]. Their main motivation behind proposing ImageBART [9] is that most of the autoregressive models model images as 1 dimensional (1D) vector with looking at only the previously generated pixels while generating the current pixel and thus ignore a global context. ImageBART [9] introduces a novel coarse-to-fine strategy to tackle this issue. ImageBart takes inspiration from BART [20] language model and incorporates bi-directional learning into multimodal data consisting of images and text.

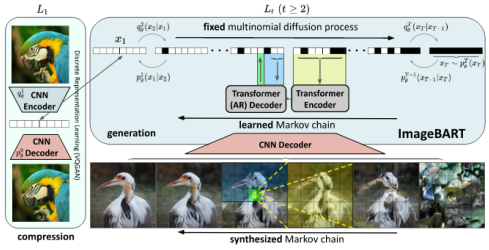

Figure 6. ImageBART Overview Architecture (source: [9])

ImageBART uses a multinomial diffusion process and learns to invert it via Markov chain into a compact and discrete image representation space. They use a coarse-to-fine strategy where each autoregressive model attains global context from the previous representation in the hierarchy. This hierarchical approach also solves the problem of generating a complete image consistent with the partial image being provided, which is a task that was failed by most of the autoregressive models before. Esser et al [9] try their approach on different image generative tasks like conditional/unconditional image generation, image completion, conditional inpainting, etc. One important thing to note here is that for smaller datasets like [18] ImageBART overfits, thus requiring large datasets to train.

Pathways Autoregressive Text-to-Image Model (Parti)

[43] is another transformer-based encoder-decoder architecture that treats text-to-image generation as a sequence-to-sequence modeling problem, analogous to machine translation allowing it to benefit from advances in large language models (LLMs). It supports content-rich image synthesis involving complex text compositions and world knowledge. It is a 2-stage model composed of an image tokenizer and an autoregressive model as highlighted in Figure 7. The first stage involves training a powerful image tokenizer - ViT-VQGAN [42] that encodes an input image into a sequence of discrete visual tokens during the training phase and takes advantage of its ability to reconstruct such image token sequences as high-quality and visually diverse images. The target outputs are sequences of image tokens instead of text tokens in another language. The second stage trains an autoregressive sequence-to-sequence encoder-decoder model that generates visual image tokens from text tokens. These visual tokens are passed through the decoder to an image detokenizer that generates an image during inference. Parti's components – encoder, decoder,

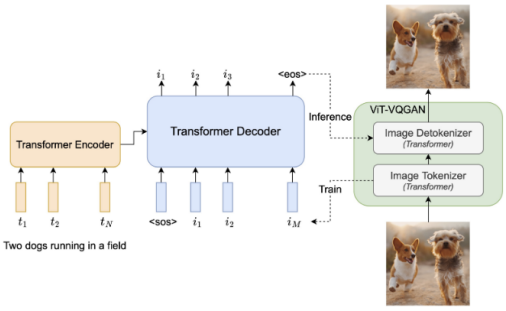

Figure 7. Parti Overview Architecture (source: [43])

and image tokenizer – are based on standard transformers [37]. This paper demonstrates that AR models can achieve state-of-the-art performance and also shows that scaling the model size consistently improves model performance and capability to generate high-fidelity photo-realistic images.

3.4. Quantization Based Models

Auto-regressive Image Synthesis with Integrated Quantization [44] proposes a novel idea of integrating feature quantization with variational regularizer into the autoregressive part of the generative modeling. The variational regularizer regularizes the feature distributions by penalizing the latent spaces of the feature distribution if the inter-domain variation is higher. The publication also provides a novel way of dealing with the distribution uncertainty in the autoregressive module by proposing a Gumbel sampling strategy. The architecture of integrated quantization VAE (IQ-VAE) can be described in three main parts -

- The image (X) and condition (C) are encoded into the latent space using an Autoencoder concurrently.
- This distribution of X and C is then modeled by an autoregressive transformer
- Then the model samples diverse sequence distributions which are then inversely quantized and then merged with the encoded condition.

The approach was tested on multiple datasets which contained different variety of images like DeepFashion [21], CelebA-HQ [22], ADE20k [45], etc. They evaluate their proposed approach on tasks of semantic-to-image, edge-to-image and keypoint-to-image generation.

Table 1. Performance of different models on CIFAR10 [18] Image generation task. The values have been curated from different publications and only reported for those models whose values were publically available

Models	bits/dimension
ARDM [13] (Upscale 4)	**2.64**
Image Transformer [23]	2.89
PixelCNN [34]	3.14
PixelRNN [34]	3
Gated PixelCNN [26]	3.03
PixelCNN++ [27]	2.92
PixelSNAIL [5]	2.88

4. Autoregressive Video Generation

Autoregressive video generation has been an active area of research, aiming to generate realistic and coherent video sequences by predicting each frame based on previously generated frames. Learning an accurate video prediction model can be very difficult as this requires both high-fidelity image generation and also modeling the physical dynamics of the video content. This is an extremely difficult learning task and requires a lot of computing resources. There are three general strategies that can be employed for autoregressive video generation.

1. modeling at an atomic pixel level across time and space.
2. modeling each frame based on the previous frame
3. modeling done in a downsampled latent space.

The latter works better as natural images contain a lot of redundancies, hence video compression formats like JPEG [39] and video compression approaches like MPEG [19] work so seamlessly. These redundancies can be removed by mapping the images to a downsampled latent space resulting in image denoising. Furthermore, autoregressively modeling the latent space rather than the pixels provides improved sampling speed and compute requirements due to the reduced dimensionality.

4.1. Autoregressive modeling in Pixel Space

Video Pixel Network (VPN) [17] is a probabalistic video model and one of the first models that can generate videos

by estimating the joint distribution of pixel values. VPN is designed to capture the time, space, and color structure of video tensors and encode it as a four-dimensional dependency chain. It uses a factorization approach that allows it to model stochastic transitions locally and globally without introducing independence assumptions. The architecture of the VPN consists of resolution-preserving CNN encoders and PixelCNN decoders. The encoders preserve the spatial resolution of input frames, while the decoders capture space and color dependencies using masked convolutions. The VPN also incorporates a convolutional LSTM to model temporal dependencies. This is primarily what makes it a Autoregressive model. The VPN is benchmarked on the Moving MNIST and Robotic Pushing datasets. It generates videos that closely match the ground truth and can generalize to the motion of novel objects. The paper compares the VPN with a baseline model that lacks spatial and color dependencies, demonstrating the importance of these dependencies in avoiding systematic artifacts in generated videos. The VPN models each conditional factor as a discrete multinomial distribution which allows for arbitrarily multimodal predictions.

The model from [25] builds on the PixelCNN architecture by parallelising the processing to improve speed. They also benchmark their model on the Action-Conditioned Video Generation task and beat the VPN architecture. They extend their architecture for video generation by adding 2 previous frames as input which act as the context c and maximizing the conditional likelihood of the next frame based on this context. The conditional likelihood is as follows:

$$p(x_{1:T}^{1:G}|c;\theta) = \prod_{g=1}^{G} p(x_{1:T}^{(g),c}|x_{1:T}^{(1:g-1)};c;\theta) \tag{4}$$

where the assumption is that G groups contain T pixels.

Video Transformer [40] scales then recent advances in autoregressive neural architectures to modern hardware accelerators. Videos are modelled as a 3D volume without making distinctions between the spatial and temporal dimensions. The distribution $p(x)$ is modelled over videos x :

$$p(x) = \prod_{i=0}^{N_p-1} \prod_{k=0}^{N_c-1} p(x_{\pi(i)})^k|x_{<\pi(i)}), x_{\pi(i)})^{<k}) \tag{5}$$

Where T is time, H is height, W is the width and N_c is the number of channels $N_p = T \cdot H \cdot W$ is the number of pixels and π is the ordering which is given by raster scan ordering, combined with subscale. The following ideas are introduced which help in scaling the models:

- Block-local Self-attention: The adjacency matrix A ϵ $R^{N_p x N_p}$ in standard transformers interconnects every element. This leads to quadratic growth and is infeasible for videos. Hence, video is divided into smaller non-overlapping 3D blocks and attention is applied individually to these blocks. This approach can be efficiently implemented in TPUs, which allows increasing model scale with minimal loss in expressive power. The proposed Video Transformer comprises multiple stacked self-attention layers which divide the data into smaller blocks of size (t,h,w) and apply self-attention. Operating on these 3D sub-volumes leads to lack of information exchange between the blocks. This can be addressed by changing the block sizes between layers. The resulting outputs do not have observable artifacts.

- Spatiotemporal Subsampling: A subscaling factor $\mathbf{s} = (s_t, s_h, s_w)$ is defined which divides the video volume into $(s_h \cdot s_w \ cdots_t)$ slices each of resolution $(T/s_t, H/s_h, W/s_w)$. The slices are generated one at a time which reduces pixels in memory to N_p/s and allows scaling the model by s. The current slices are generated conditioned on the pixels of previous slices. The first step is passing through a subscale encoder where the video undergoes masking and embedding. Subsequently, a 3D padded convolution is applied with stride $\mathbf{s}$ to obtain the desired resolution. Then the positional embedding for each axis and embedding of the current slice is added to the convolution output which is further transformed by a linear projection and passed to the L block-local self-attention layers. The resulting output is passed to the subscale decoder. The decoder is similar to the encoder, the current slice is embedded and a 3x3x3 masked convolution is applied. Then the positional embeddings are added for all the dimensions and this is passed through the L block-local self-attention layers, with masking. The final pixel intensities are predicted by MLP with a single hidden layer and are conditioned on this output. The video slice loss is defined as the negative log-likelihood.

4.2. Autoregressive Frame Prediction

These models predict the next frame based on a previous frame or set of previous frames as input. [14] frames this task as generating a sequence of T video frames, denoted as $V' = \{f_t^{'}\}_{t=1}^{T}$ given an image $f_0 \epsilon R^{HxWx3}$ as the starter frame. This is a hard problem as no temporal information is provided by the input. Being autoregressive0 it can generate videos of arbitrary lengths and smooth trajectories. However, this approach suffers from two main disadvantages. First, as the generation process goes on, noises and undesirable artifacts accumulate, and as a result, the generation quality suffers over time. Second, the nature of autoregressive generation leads to discrepant behaviors during the training and testing phases. To counter these two main ideas are utilized:

- Complementary mask mechanism is a key component of the generator network. It separates static pixels and variable pixels such that the model learns to reuse some pixels from the previous time step. In this way, the quality degradation problem in the autoregressive generation process is greatly suppressed. The mask used for static and variable pixels sums to one enforcing the complementary relationship while constructing a frame.

- Scheduled Sampling deals with the discrepancy between training and testing. If only ground truth next-frames are used during training, the model lacks the ability to deal with generated frames. Scheduled sampling is performed in the later epochs of training where the model randomly switches its input between the ground truth current frame f_t and the synthesized current frame $f_t^{`}$ produced by the previous time step. In this way, we can increase the model's exposure to its own generations, and bridge the behavioral gap between training and testing phases of generations.

The architecture comprises a generator that has two streams - ResNet-based (for mask) and a U-Net based (provides difference map between previous frame and next frame). The global discriminator distinguishes between ground-truth next frame and generated next frame. The local discriminator distinguishes between variable pixels. PatchGAN was used for both.

4.3. Latent Video Prediction

These are models that divide video generation into two sub-tasks - First, training an image-generator and second, training an autoregressive model for video prediction.

4.3.1 Predicting Video with VQVAE [38]

This paper uses hierarchical latent representations and combines them with autoregressive models (PixelCNNs) to predict videos. They use coarse-to-fine modeling and conditioning on latent representations to manage the complexity of video prediction while capturing temporal dependencies.

4.3.2 Transformer-Based Architectures

Latent Video Transformer (LVT) [24] utilizes a latent space and transformers to capture long-term dependencies across frames and generate high-quality videos. The transformer architecture enables efficient modeling of spatial and temporal dependencies, while the VAE framework provides a structured latent space for improved video synthesis. Firstly, the frames are encoded into a discrete latent space via a Vector Quantised VAE (VQ-VAE) [36]. Sliced Vector Quantization [46] is used to prevent index collapse (a common problem in VQ-VAEs). New latent frames are then generated in an autoregressive manner using a transformer. These latent frames are decoded back to pixel space via the VQ-VAE decoder. LVT achieves impressive results in generating diverse and realistic video sequences.

VideoGPT [41] also uses the VQ-VAE encoder-decoder and the autoregressive GPT model. VQ-VAE is responsible for learning a downsampled discrete representation of input video using 3D convolutions and axial self-attention. This encoding captures the spatial and temporal redundancies in the video frames. The discrete latents are then fed into the GPT-like model, which autoregressively predicts the next latent frame based on the previous ones using spatiotemporal position encodings. Finally, the decoded latents are upsampled to generate videos at the original resolution. The choice of using likelihood-based models for video generation is motivated by their success in modeling discrete data and their well-established training recipes. Likelihood-based models also offer advantages in terms of optimization and evaluation compared to adversarial models.

HARP [28] extends the above work to scale up the autoregressive video prediction to much higher resolutions and high-fidelity videos. HARP can do the above with minimal modifications to the existing training recipes. It replaces the VQ-VAE with a VQ-GAN [10] that proves to be effective for high-resolution image generation. Further, it uses a causal transformer for autoregressive modeling in the discrete latent space. Another advantage of HARP is that it can leverage image generators trained on a vast set of natural images (such as ImageNet [6] dataset) to train a high-resolution video prediction model on complex and large-scale datasets like Kinetics-600. It incorporates techniques like top-k sampling and data augmentation to further enhance video prediction quality. The paper also highlights some common shortcomings such as modeling complex inter-object interactions and sometimes getting stuck in local minima with degenerate video predictions.

Overall, VideoGPT [41], LVT [24] and HARP [28] provide a simple yet effective approach to video generation using a combination of VQ-VAE and Transformers, and it serves as a reference for minimalistic implementations of transformer-based video generation models.

Table 2. FVD Scores for models on the BAIR Robot Pushing dataset

Models	FVD (↓)
Video Transformer [40]	94
HARP [28]	99.3
Video GPT [41]	103.3
LVT [24]	125.8

5. Datasets and Metrics

5.1. Image Generation

Most approaches evaluate their methods on a subset of MNIST [7], CIFAR-10 [18], ImageNet 32 × 32 [6], and high-resolution CelebA-HQ [22] dataset. The metrics commonly used to compare image generation models are bits per dimension (BPD) or average negative log-likelihood (NLL) in equation 6, Fréchet Inception Distance (FID) [12], and Inception score over the image dimensions, which represent the measure of sample quality.

$$\text{NLL}(x) = -\log P(x_1, \ldots, x_{n^2}) = -\sum_{i=1}^{n^2} \log P(x_i | x_{<i}) \tag{6}$$

5.2. Video Generation

One of the most common datasets used for this task is the BAIR Robot Pushing [8] consisting of 40K training and 256 test videos. Some other datasets include the Moving MNIST [29] and the Kinetics-600 dataset [3] (now updated to 700). The most commonly used evaluation metric is Fréchet Video Distance (FVD) [31] which builds on the FID score [12]. FVD introduces a feature representation to capture the temporal coherence of the video content, in addition to each frame's quality. The metric uses an Inflated 3D ConvNet (I3D) [4] to learn pre-trained feature representation. The I3D model is applied to get a conditional label distribution for the video $p(y|x)$. Assuming the distribution for the model output is $p_{model} = \mathcal{N}(\mu_{model}, \Sigma_{model})$ and the distribution for the ground truth is $p_{GT} = \mathcal{N}(\mu_{GT}, \Sigma_{GT})$, we can calculate the Fréchet Distance as:

$$d(p_{model}, p_{GT}) = ||\mu_{model} - \mu_{world}||_2^2 + Tr(\Sigma_{model} + \Sigma_{GT} - 2(\Sigma_{model}\Sigma_{GT})^{\frac{1}{2}})$$

Multivariate Gaussian distribution is chosen because the Gaussian is the maximum entropy distribution for a given mean and covariance.

6. Insights and Critical Analysis

Autoregressive image generation models have been widely used due to their straightforward training approach. However, they also present a challenging optimization problem when it comes to maximizing the likelihood of data. To address this limitation, researchers have increasingly favored diffusion models, which offer a more manageable task. Instead of optimizing the entire likelihood of data, diffusion models sample and optimize a component of the likelihood. Order-agnostic autoregressive models are a step in this direction and should be explored more as they alleviate some difficulties associated with the optimization challenge and also allow for parallelized image generation leading to improved inference speeds.

For autoregressive video generation, current state-of-the-art models struggle to accurately model complex object interactions observed in the real-world, as exemplified by HARP. To overcome this limitation, incorporating physics priors can be a valuable approach. By disentangling the task of redundant image computations and modeling intricate interactions, physics priors help enhance the models' capability to capture realistic dynamics. Taking inspiration from Simultaneous Localization and Mapping (SLAM) systems, which implicitly model the physics of the environment and camera movement, can provide valuable insights for incorporating such priors into autoregressive video generation models.

By addressing the drawbacks and exploring these improvements in autoregressive image and video generation models, we can pave the way for more advanced and realistic generative models in computer vision.

References

[1] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023. 1, 5

[2] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 400–406, Cambridge, MA, USA, 1999. MIT Press. 2

[3] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset, 2022. 9

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset, 2018. 9

[5] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model, 2017. 1, 4, 6

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 8, 9

[7] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 9

[8] Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections, 2017. 9

[9] Patrick Esser, Robin Rombach, Andreas Blattmann, and Björn Ommer. Imagebart: Bidirectional context with multinomial diffusion for autoregressive image synthesis, 2021. 1, 5

[10] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021. 8

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. 9

[13] Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models, 2022. 1, 5, 6

[14] Jiahui Huang, Yew Ken Chia, Samson Yu, Kevin Yee, Dennis Küster, Eva G. Krumhuber, Dorien Herremans, and Gemma Roig. Single image video prediction with auto-regressive gans, May 2022. 7

[15] Ajay Jain, Pieter Abbeel, and Deepak Pathak. Locally masked convolution for autoregressive models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020. 1, 4

[16] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory, 2016. 3

[17] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks, 2016. 1, 6

[18] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 5, 6, 9

[19] Didier Le Gall. Mpeg: A video compression standard for multimedia applications. *Commun. ACM*, 34(4):46–58, apr 1991. 6

[20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019. 5

[21] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1096–1104, 2016. 6

[22] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild, 2015. 6, 9

[23] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer, 2018. 1, 4, 5, 6

[24] Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer. *CoRR*, abs/2006.10704, 2020. 1, 8

[25] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Dan Belov, and Nando de Freitas. Parallel multiscale autoregressive density estimation, 2017. 1, 7

[26] Scott E. Reed, Aaron van den Oord, Nal Kalchbrenner, Victor Bapst, Matthew M. Botvinick, and Nando de Freitas. Generating interpretable images with controllable structure, 2017. 1, 3, 6

[27] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017. 3, 4, 6

[28] Younggyo Seo, Kimin Lee, Fangchen Liu, Stephen James, and Pieter Abbeel. Harp: Autoregressive latent video prediction with high-fidelity image generator, 2022. 1, 8

[29] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms, 2016. 9

[30] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks, 2015. 3

[31] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new metric for video generation, 2019. 9

[32] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation, 2016. 2, 4

[33] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator, 2014. 5

[34] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016. 1, 2, 3, 4, 6

[35] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016. 3

[36] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. 8

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 2, 6

[38] Jacob Walker, Ali Razavi, and Aäron van den Oord. Predicting video with vqvae, 2021. 1, 8

[39] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992. 6

[40] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models, 2020. 1, 7, 8

[41] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers, 2021. 1, 8

[42] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan, 2022. 6

[43] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation, 2022. 1, 6

[44] Fangneng Zhan, Yingchen Yu, Rongliang Wu, Jiahui Zhang, Kaiwen Cui, Changgong Zhang, and Shijian Lu. Autoregressive image synthesis with integrated quantization, 2022. 1, 6

[45] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, 2017. 6

[46] Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables, 2018. 8

[47] Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms, 2016. 3