Paying Attention to Video Generation

Rishika Bhagwatkar rishika.vnit@gmail.com
Khurshed Fitter khurshedpf@gmail.com
Saketh Bachu saketh7000@gmail.com
Akshay Kulkarni akshayk.vnit@gmail.com
Shital Chiddarwar s.chiddarwar@gmail.com
IvLabs, Mechanical Engineering Department, Visvesvaraya National Institute of Technology, India

Abstract

Video generation is a challenging research topic which has been tackled by a variety of methods including Generative Adversarial Networks (GANs), Variational Autoencoders (VAE), optical flow and autoregressive models. However, most of the existing works model the task as image manipulation and learn pixel-level transforms. In contrast, we propose a latent vector manipulation approach using sequential models, particularly the Generative Pre-trained Transformer (GPT). Further, we propose a novel Attention-based Discretized Autoencoder (ADAE) which learns a finite-sized codebook that serves as a basis for latent space representations of frames, to be modelled by the sequential model. To tackle the reduced resolution or the diversity bottleneck caused by the finite codebook, we propose attention-based soft-alignment instead of a hard distance-based choice for sampling the latent vectors. We extensively evaluate the proposed approach on the BAIR Robot Pushing, Sky Time-lapse and Dinosaur Game datasets and compare with state-of-the-art (SOTA) approaches. Upon experimentation, we find that our model suffers mode collapse owing to a single vector latent space learned by the ADAE. The cause for this mode collapse is traced back to the peaky attention scores resulting from the codebook (Keys and Values) and the encoder's output (Query). Through our findings, we highlight the importance of reliable latent space frame representations for successful sequential modelling.

Keywords: Video Generation, GPT, Transformer, Attention, Discretized Autoencoder

1. Introduction

Machine Learning (ML) paradigms and model architectures are designed and optimized, keeping in mind the task that is intended to be solved or tackled (Alom et al., 2019; Liu et al., 2017). This, in turn introduces passive yet influential notions for domain-specific conventions and methodologies for modelling ML systems. While some notions like the direct dependence of a model's performance and the amount of relevant training data (Alom et al., 2019; SUG, 2018) hold true for most cases, other factors like model architectures and evaluation metrics differ drastically across tasks. A prime example of this is the stark difference between models that deal with images and those that deal with temporal data like speech and text. We propose an attempt to harness the best of both worlds for video generation tasks.

Generating videos from initial frame(s), on the face of it, seems to be an image manipulation task which could be tackled satisfactorily using models based on or employing generative paradigms, mainly Generative Adversarial Networks (GANs) (Zhang et al., 2019; Gur et al., 2020; Clark et al., 2019; Luc et al., 2020). Such techniques usually focus on learning pixel-level transforms (Li et al., 2017; Parmar et al., 2018; Vondrick and Torralba, 2017; Zhang et al., 2019) and try to generate succeeding frames using learnable motion or pixel flow characteristics (Ohnishi et al., 2017; Patraucean et al., 2016). However, an alternative stance could be to formulate video generation as a sequence modelling task (Mittal et al., 2017; Dandi et al., 2019a; He et al., 2018) on a finite set of latent vectors. The recent advancements in Natural Language Processing (NLP) (Brown et al., 2020; Devlin et al., 2019; Radford et al., 2018, 2019; Vaswani et al., 2017) reflect the success of sequence modelling on latent space representations of language tokens.

Most of the current literature on video generation uses the former approach. However, we choose the latter as our initial stance to leverage the advancements in sequential modelling. Further, most video generation approaches use pixel-level predictions to model and generate images as a series of pixels (van den Oord et al., 2016; Chen et al., 2017). We focus on latent vector manipulation using sequential models, namely the Generative Pre-trained Transformer (GPT) (Brown et al., 2020; Radford et al., 2018, 2019). Sequence models perform well at learning the plausible neighbours or successors of a given set of language tokens. We propose to extend the idea such that the model learns to predict the latent space representation of a frame, by conditioning over a set of already available (predicted or provided) latent vectors of previous frames.

We propose a novel Attention-based Discretized Autoencoder (ADAE) which learns a discretized finite set of vectors called the codebook. The codebook is then used to generate frame embeddings, analogous to word embeddings (Almeida and Xexéo, 2019; Mikolov et al., 2013) in NLP. We use a discretized Autoencoder (AE) to ensure a finite latent basis size, analogous to vocabularies in NLP. We intend to validate our approach on the BAIR Robot Pushing (Ebert et al., 2017), Sky Time-lapse (Xiong et al., 2018) and Dinosaur Game datasets and compare the performance with state-of-the-art (SOTA) approaches.
The main contributions of our work are:

1. A novel ADAE to represent an entire frame as an encoded vector referred to as a frame embedding.
2. Utilizing the sequential modelling prowess of the GPT to sequentially model frame embeddings.

2. Related work

2.1. Video generation

Videos can be generated from text, initial frames, complete videos and even pixel transformations on existing frames (Li et al., 2017; Vondrick and Torralba, 2017; Parmar et al., 2018; Cai et al., 2018; Girdhar et al., 2019). Video generation started off as a deterministic modelling task and later shifted towards approaches involving GANs (Goodfellow et al., 2014; Zhang et al., 2019; Gur et al., 2020; Mathieu et al., 2016; Vondrick et al., 2016; Saito et al., 2017; Tulyakov et al., 2017; Acharya et al., 2018; Saito and Saito, 2018; Clark et al., 2019; Luc et al., 2020), Variational Autoencoders (VAEs) (Dandi et al., 2019b; Gur et al., 2020; van den Oord et al., 2018; Razavi et al., 2019; Denton and Fergus, 2018; Denton and Birodkar, 2017; Lee et al., 2018; Hsieh et al., 2018), optical flow(Ohnishi et al., 2017;

Patraucean et al., 2016) and autoregressive models(Ranzato et al., 2016; Srivastava et al., 2016; Shi et al., 2015; Kalchbrenner et al., 2016; Weissenborn et al., 2020; Ho et al., 2019). Using only VAEs does not provide the best of results, and using GANs usually makes the models computationally expensive and difficult to train.

The current state of the art approaches like TRIVD-GAN-FP (Luc et al., 2020) and autoregressive models like PixelSnail (Chen et al., 2017; van den Oord et al., 2016) deliver great quality frames but come at a heavy price of training and deployment overheads and do not seem to perform well on resolutions beyond 64×64 pixels. A similar issue arises with the attention and subscaling-based Video Transformer (Weissenborn et al., 2020; Menick and Kalchbrenner, 2018).

The latest work, Latent Video Transformer (Rakhimov et al., 2020) generates videos by generating latent space representations of frames, by coupling a Vector Quantized Variational Autoencoder (VQ-VAE) (van den Oord et al., 2018) and an entire Transformer (Vaswani et al., 2017). We use only the decoder blocks of the transformer, stacked on top of each other to form a GPT-based sequential model along with our ADAE. The choice of decoder blocks of the Transformer, over the encoder blocks, is due to the presence of masked self-attention which forces the model to condition on the previously available (provided or generated) frame embeddings. While masking off random samples (BERT-based paradigm (Devlin et al., 2019)) may enrich the robustness of the generated embeddings, we prefer the GPT-based approach to learn sequential modelling.

2.2. Latent space representation

The robustness and effectiveness of most NLP models depend heavily on the robustness and contextual capacity of the token embeddings used (Almeida and Xexéo, 2019). Word embeddings are the most commonly used latent space or embedded representations of tokens. The GPT and BERT-based approaches are prime examples of successful NLP models banking on robust embeddings. Other approaches like ELMo (Peters et al., 2018), further decompose words into characters and hence do not need to worry about a fixed vocabulary space as opposed to models like GPT and BERT which use a fixed size embedding space and outliers are usually assigned a special token, resulting in a single referencing index for each token.

Latent space representations for images are usually continuous d-dimensional vector spaces (d is the bottleneck or encoding dimension) and are learned using autoencoder-based models. The VQ-VAE (van den Oord et al., 2018) and VQ-VAE-2 (Razavi et al., 2019) models, learn a discretized, finite subset of the d-dimensional latent space called the codebook, the size of which is fixed a priori. However, the discrete vectors in the codebook, contain mainly positional information about the latent space representation of an image.

We propose a novel Attention-based Discretized Autoencoder (ADAE) that discretizes the continuous latent space $\mathbb{R}^d$ into a finite codebook. However, unlike the VQ-VAE models (van den Oord et al., 2018; Razavi et al., 2019), the codebook in our model forms a basis that is used to encode an entire frame into a single vector using an attention-based soft-aligned linear combination (Vaswani et al., 2017; Bahdanau et al., 2016) of the codebook vectors.

3. Approach

3.1. Attention-based discretized autoencoder

In typical reconstruction AEs, the encoder $E(.;\theta)$ learns to map an input $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ to a vector belonging to a continuous latent space $\mathbb{R}^d$, i.e. $E(\mathbf{x};\theta) = \mathbf{z} \in \mathbb{R}^d$ where d is the bottleneck dimension. The decoder jointly learns a mapping from $\mathbf{z} \in \mathbb{R}^d$ to $D(\mathbf{z};\phi) = \mathbf{x}' \in \mathbb{R}^{H\times W\times C}$. The loss criterion is simply the Mean Squared Error (MSE) loss between the output of the decoder and the input.

$$\mathcal{L}(\mathbf{x}, \mathbf{x}'; \theta, \phi) = \|\, \mathbf{x} - \mathbf{x}' \,\|_2^2 \tag{1}$$

Such AEs can not be directly paired with language models due to the latter's requirement of finite vocabulary sizes. We discretize the latent space of the AE into a finite set $\xi = \{\mathbf{e}_i\}\ \forall\ i \in [1, N]$, forming a codebook of N vectors with each vector $\mathbf{e}_i \in \mathbb{R}^d$. The latent vector $\mathbf{z}$ is now a linear combination of the codebook vectors.

$$\mathbf{z} = \sum_{i\in I} a_i \mathbf{e}_i \tag{2}$$

Where, I is the set of top-k indices, ranked in decreasing order of normalized attention scores $a_i = A(E(\mathbf{x};\theta), \mathbf{e}_i)$ as per the attention metric $A(.,.)$. This introduces a soft-alignment metric as opposed to a choice based on strict minimal distance. If we replace the linear combination by

$$\mathbf{z} = \mathbf{e}_j \text{ such that } \|\, E(\mathbf{x};\theta) - \mathbf{e}_j \,\|_2^2 < \|\, E(\mathbf{x};\theta) - \mathbf{e}_i \,\|_2^2 \ \ \forall\ i \in [1, N] - \{j\} \tag{3}$$

then similar inputs may produce identical outputs leading to a diversity bottleneck. We tackle this by introducing an attention-based soft-alignment (Bahdanau et al., 2016; Vaswani et al., 2017). We formulate our loss function as

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}; \theta, \phi, \mathbf{e}_i) = \|\, \mathbf{y} - \hat{\mathbf{y}} \,\|_2^2 + \|\, \mathrm{sg}[E(\mathbf{x};\theta)] - \mathbf{e}_i \,\|_2^2 + \beta \,\|\, E(\mathbf{x};\theta) - \mathrm{sg}[\mathbf{e}_i] \,\|_2^2 \ \ \forall\ i \in I \tag{4}$$

The loss function is similar to that proposed by Razavi et al. (2019) and the first term is simply the MSE loss between the label and decoded output. The stop-gradient operation is denoted by sg[.] and β is a hyperparameter which incorporates the relative reluctance of changing the codebook corresponding to the encoder's output.
The fundamental difference between our Attention-based Discretized Autoencoder (ADAE) (shown in Fig. 1a) and the VQ-VAE (van den Oord et al., 2018; Razavi et al., 2019) models, is that the latter focuses on learning a grid-like latent space representation for an input, with the latent vectors being sampled from a discrete codebook and the reconstruction is done pixel by pixel. Our model on the other hand, learns to encode an entire frame as a linear combination of discrete vectors from a learnable codebook. The reconstruction is not autoregressive (as opposed to the works by van den Oord et al. (2018); Razavi et al. (2019)) and the decoder learns to model the distribution $p(\mathbf{x} \mid \mathbf{z})$.

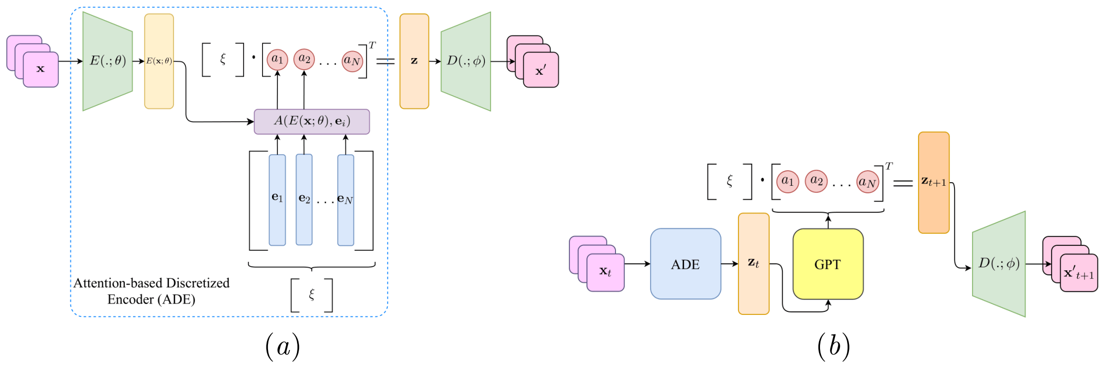


Figure 1: **(a)** A block diagram of our ADAE model. The attention metric $A(.,.)$ calculates attention scores between the encoder's output and codebook vectors. The normalized scores are used as coefficients of the corresponding codebook vectors to get the frame embedding ($\mathbf{z}$). The frame embedding is then passed to the decoder for reconstruction. **(b)** A block diagram of our approach. The GPT is given t frame embeddings ($\mathbf{z}_{\{1,2,\ldots t\}}$) and it predicts t probability distributions, each corresponding to the model's prediction for the succeeding frame embedding, for each frame in the input set. **Note:** In both the figures, we have shown only 1 RGB frame, with top-k filtering disabled.

3.2. Generative pre-trained transformer

A GPT (Radford et al., 2018, 2019; Brown et al., 2020) is a large, generative, attention-based sequential model, based on the Transformer model (Vaswani et al., 2017). However, the model utilizes only the decoder blocks of the Transformer model and employs masked self-attention to learn sequential generation of tokens.

We propose to interpose a GPT-based model between the encoder and decoder parts of our proposed ADAE (as shown in Fig. 1b) to learn sequential modelling over the latent vectors. The output of such models is usually a probability distribution vector of length equal to the number of elements in the vocabulary. The indices of top-k probability scores from this output distribution are used to form the set I. This set contains indices of the attention scores to be used as coefficients of the codebook vectors to form a linear combination as per Eq 2. The result of this linear combination (frame embedding) is then passed to the ADAE's decoder for reconstruction.

4. Experiments

4.1. Datasets

- **BAIR Robot Pushing Dataset:** This dataset (Ebert et al., 2017) contains about 59,000 videos ($\approx$ 1.5 million video frames) of robots interacting with objects mainly via pushing motions. It is divided into a training set (57,000 videos), an unseen test set and a seen test set (1,250 videos each).

- **Sky Time-lapse Dataset:** This dataset (Xiong et al., 2018) contains about 38,000 videos of 32 frames each. The videos are time lapse shots of the sky with clouds and stars moving across the frames and are sourced mainly from YouTube.
- **Dinosaur Game Dataset:** We have curated over 5 hours of video, containing clips of Google Chrome's "Dinosaur Game". The frames can be binarized into black and white frames, hence providing us with a comfortably learnable dataset to test prototypes.

We resize all the frames to $64 \times 64 \times 3$ pixels for a fair comparison with previous works, although we shall further experiment using higher resolutions to evaluate the scalability. Further, we introduce a downsampling parameter (m_d) to downsample or skip frames from videos, in cases where consecutive frames are too similar. This helps ensure that the model does not end up learning an approximate identity map for some cases.

4.2. Teacher forcing

Teacher forcing ratio (r_{tf}) is the probability of using the ground truth as an input to a sequential model, instead of its own prediction at a given time stamp. Usually, it is set to 1 while training and 0 while testing. However, we set it to 1 for a few initial frames (up to one fourth of video length) and then set it as a hyperparameter which determines the trade-off between ease of convergence while training and robustness while sampling.

4.3. Model architectures

Attention-based Discretized Autoencoder: We propose an ADAE which discretizes the continuous latent space $\mathbb{R}^d$ into a fixed size codebook. This makes it almost intuitive to use its codebook as a basis for the vocabulary or embedding space of a language model, like the GPT employed by us.

We intend to implement broadly two classes of autoencoder architectures as a part of our ablation study. The encoders are based on the architectures employed in VQ-VAE (van den Oord et al., 2018; Razavi et al., 2019) and ResNet (He et al., 2015) along with their respective symmetrical (mirror image) inversions as the decoder networks.

Further we propose varying the codebook size N, from about 1,000 to 65,000 (in steps of $\approx$ 8,000) and encoding/bottleneck dimension d, from around 200 to more than 2,000 (in steps of $\approx$ 250). We intend to compare amongst and choose from a few attention metrics, namely, dot product attention (Bahdanau et al., 2016), energy based attention and Key-Query-Value-based attention (Vaswani et al., 2017) and shall be tuning the value of k for choosing the top-k attention scores, for each metric.

Generative Pre-trained Transformer: The GPT model forms the sequential kernel of our model and is interposed between the encoder and the decoder of our ADAE. It takes in attended frame embeddings ($\mathbf{z}_1, \mathbf{z}_2, \dots \mathbf{z}_{t-1} \in \mathbb{R}^d$) and outputs the attention scores to be used as coefficients for the linear combinations of the codebook vectors, to get the next frame embedding ($\mathbf{z}_t \in \mathbb{R}^d$) as per Eq 2.

We shall vary the depth of the model from about 10 to 40 stacked, modified Transformer (Vaswani et al., 2017) decoder blocks, similar to the architectures proposed by Radford et al.

(2019). Further, we will be using learnable positional embeddings as opposed to sinusoidal ones used by Vaswani et al. (2017).

4.4. Training and evaluation metrics

As a part of our ablation experiments, we shall train our ADAE and GPT models jointly as well as separately. When training separately, we shall train the ADAE as a reconstruction autoencoder ($\mathbf{x} = \mathbf{y}$) first and then fine-tune it while training the GPT model. While training jointly, we would train both the networks from scratch. However, while training jointly, the labels would be the succeeding frame as opposed to the same frame in the disjoint case.

Finally, we intend to evaluate our model on BAIR Robot Pushing and Sky Time-lapse datasets and compare it with other models as shown in Tables 1 and 2.

Table 1: Frechet Video Distance (FVD) (Unterthiner et al., 2018) scores of various models, referred from the work by Rakhimov et al. (2020).

Method	FVD (↓)
SVP-FP (Denton and Fergus, 2018)	315.5
CDNA (Ebert et al., 2017)	296.5
LVT (Rakhimov et al., 2020)	125.8 ± 2.9
SV2P (Denton and Fergus, 2018)	262.5
SAVP (Lee et al., 2018)	116.4
DVD-GAN-FP (Clark et al., 2019)	109.8
TriVD-GAN-FP (Luc et al., 2020)	103.3
Video Transformer (Weissenborn et al., 2020)	94 ± 2
Ours	407.4

Table 2: Peak Signal to Noise Ratio (PSNR) and Strucutral Similarity Index (SSIM) (Zhou Wang et al., 2004) scores of various models, referred from the work by Zhang et al. (2020).

Method	PSNR (↑)	SSIM (↑)
MoCoGAN (Tulyakov et al., 2017)	23.867	0.849
MDGAN (Intrator et al., 2018)	23.042	0.822
DTVNet (Zhang et al., 2020)	29.917	0.916
Ours	2.445	0.192

Note: The FVD score has been reported on the BAIR dataset whereas the PSNR and SSIM scores have been reported on the Sky Time-lapse dataset (following previous works).

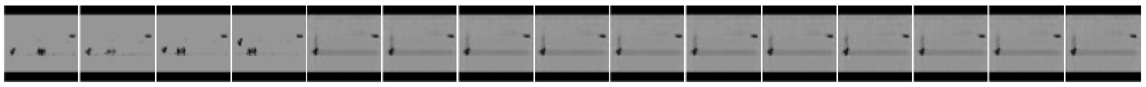
5. Discussion

We consider potential issues that could arise in the proposed approach. One of them could be mode-collapse where our model generates sequences of almost indistinguishable (as per human standards) frames. In such a case, we would increase the downsampling rate in order to ensure enough diversity between consecutive frames. On the other hand, if our model is unable to converge, then we may consider reducing the downsampling rate in an attempt to make variations more subtle.

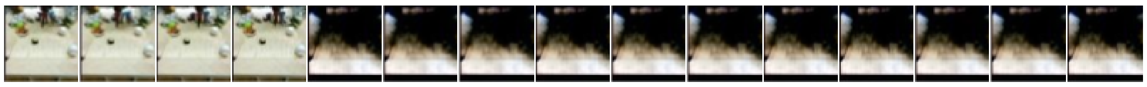
Another plausible issue could be unwanted blurry regions or artifacts appearing after reconstruction, owing to the fact that the latent space representation focuses more on the entire frame rather than local groups of pixels. We could tackle this by dividing frames into grids and learning multiple codebooks.

6. Results

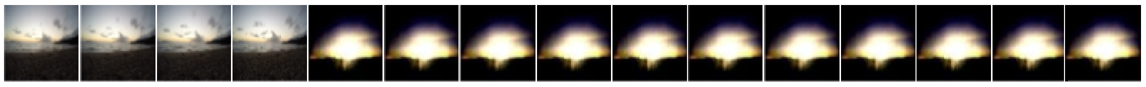
We trained the models on sequences of 15 frames each. We performed ablations for all the possible values of hyperparameters within the ranges stated above. Additionally, we made suitable modifications that we have documented and justified in the next section. We performed experiments using a single Nvidia 1660 Ti GPU.



(a) Dinosaur Game Dataset



(b) BAIR Robot Pushing Dataset



(c) Sky Time-lapse Dataset

Figure 2: We sampled 15 frame videos from the model after providing 4 initial frames as input. However, the model suffers mode collapse in all cases.

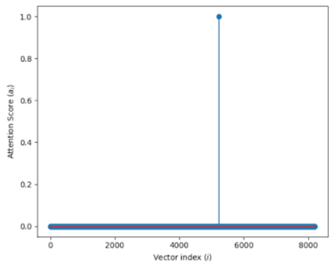
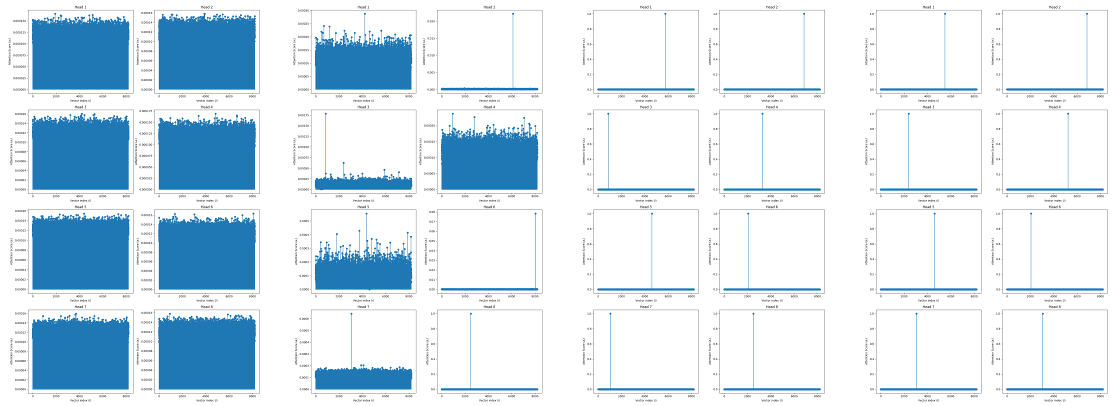


Figure 3: The output of the GPT is a normalized attention score distribution used to generate the next frame embedding by taking a linear combination of the codebook vectors. However, our model collapses to a unit magnitude delta distribution as the output.



(a) 1 epoch (b) 20 epochs (c) 40 epochs (d) 200 epochs

Figure 4: During training, we visualized the attention score distributions over the codebook vectors for each head at regular intervals. All the ablations started off with uniform attention distributions but quickly collapsed to peaky attention score distributions.

7. Findings and modifications

We carried out the ablation studies as proposed in the above sections. We have reported the findings for each sub-part below.

7.1. Joint training

The proposed solution was to train the entire model, including the GPT, from scratch. However, for any combination of the hyperparameters, we were unable to prevent mode collapse. The attention scores $a_i(\mathrm{s})$ that are the normalized output of the GPT (as described in Eq 2), were all 0, except at one value of i, i.e. the distribution was a unit magnitude delta distribution as shown in Fig 3. This resulted in a hard-choice selection of a particular codebook vector and the model updated that vector, to minimize the loss, as opposed to learning all the codebook vectors. This resulted in a random "choice" based learning for a single codebook vector. Hence each output prediction was the same frame, indicating that the model had collapsed.

Upon probing previous layers of the model, we found that the cause for a peaky distribution at the output of the GPT was a similar attention score distribution used to calculate the vector $\mathbf{z}$ from the vector $E(\mathbf{x})$. Hence, it was not the GPT but the ADAE that was restricting the latent space to a single vector instead of linear combinations of the entire codebook.

We attempted to mitigate the above-mentioned problem by including a scaling factor that scales down the attention scores before calculating their $Softmax$, analogous to variable temperature attention. Regardless of the scale, the attention scores continued forming peaky distributions resulting in mode collapse. Additionally, changing the teacher forcing ratio (r_{tf}) and downsampling parameter (m_d) yielded no significant improvement. Even after changing the dimensionality of the model, which includes the Key, Query and Value dimensions, we were unable to avoid mode collapse. The model failed to overfit even on small datasets. To probe the high bias choices made by the model, we employed a disjoint, more focused approach of training each component individually and then fine-tuning the entire system.

7.2. Disjoint training

The GPT manipulates frame embeddings generated by the ADAE, which emphasizes the pivotal role of the ADAE. Thus, we aimed to train the ADAE first. The trained ADAE can then be used as a reliable model to generate latent representations of frames, i.e. frame embeddings. We performed ablation studies that involved changes to the codebook, followed by the encoder and the loss function, and finally, the initialization and attention metric.

We tested the reliability of the GPT model by training it on vector sequences, for both the cases of embeddings - learnable and frozen (pre-trained). The GPT performed at par with the findings reported by Radford et al. (2019), validating its proper functioning.

7.2.1. Codebook

We experimented with various codebook sizes in the proposed range of values. However, negligible deterioration or improvement was observed in these ablations. This concurs with the reasoning that the peaky attention distribution picks one codebook vector and the training process directs the model to learn a single latent frame representation. This latent vector is learnt to minimize the loss function, specifically the reconstruction term, resulting in an output frame that is very close to the average of all the frames in the dataset.

Further, tuning the hyperparameters influencing the attention mechanism also yielded no meaningful improvement. The codebook vectors are used as Keys and Values, to compute attention scores with the Queries, that are the encoder's output vectors $E(\mathbf{x}, \theta)$. Thus, we introduce three linear transforms, to transform the Key, Query and Value inputs to their respective dimensions. Despite the Key, Query and Value dimensions being tunable hyperparameters, we were unable to alter and stabilize the attention distribution to include multiple codebook vectors instead of "choosing" only one.

Initially, the attention distributions are uniform, but rapidly collapse to a peaky distribution as displayed in Fig 4. In rare cases when the attention distribution would start off with multiple noticeable peaks, it would quickly collapse to one of them within a few epochs. Even with small learning rates and gradient clipping, we were unable to prevent the model from collapsing. This resulted in the obvious implication of getting the same output frame for all the inputs, hence confirming the root cause of the mode collapse problem described above.

7.2.2. Encoder

Initially, we experimented with the encoder and decoder in parallel as we proposed using the mirror image of the encoder as the decoder. Later, we tried asymmetrical decoders by introducing differences with regards to depth and combinations of layers.

Since both the proposed encoder types were based on the ResNet-50, we performed the following three classes of ablations:

1. Training a randomly initialized ResNet-50 encoder along with the codebook and the decoder from scratch.

2. Fine-tuning a pre-trained ResNet-50 encoder while learning the codebook and the decoder from scratch.

3. Freezing the pre-trained ResNet-50 encoder for a few epochs and then fine-tuning it along with the codebook and the decoder.

We used PyTorch's (Paszke et al., 2019) inbuilt ImageNet pre-trained (Deng et al., 2009) ResNet-50 with a top-1 accuracy of 76.130% and a top-5 accuracy of 92.862%. Further, we used a linear layer to map the output of the convolutional layers to the required embedding dimension in accordance with the Query dimension.

However, none of the ablations were able to solve the problem of peaky attention score distributions, regardless of the embedding dimension. Owing to the collapse in attention scores, the encoder learns weights that maximise similarity with the chosen codebook vector, hence generating very similar encodings $E(\mathbf{x})$ for different inputs. As stated above, changes to the subscaling factor yielded no improvement in the results and the outputs kept collapsing to a single frame, regardless of the input.

7.2.3. Loss Function

Our experiments with loss functions initially used the proposed loss function as the baseline, while varying values of β. Further, we experimented with l_1 and Huber (Smooth-l_1 Norm) losses in place of the squared error proposed in Eq 4 and even in cases when we considered only the reconstruction term. In Eq 4, the second and third terms focus on aligning the codebook vectors with the encoder's output and vice-versa. However, upon realising that the mode collapse problem resulted in the same output frame for each input frame due to peaky attention, we removed the second and third terms in the loss function to emphasize more on achieving better reconstruction. Also, this mode collapse implied that the use of top-k choices became trivial. Further, we attempted to improve the latent space representation of a frame $\mathbf{z}$ by modifying the proposed loss function as follows,

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}; \theta, \phi, \mathbf{z}) = \|\, \mathbf{y} - \hat{\mathbf{y}} \,\|_2^2 + \|\, \mathrm{sg}[E(\mathbf{x}; \theta)] - \mathbf{z} \,\|_2^2 + \beta \,\|\, E(\mathbf{x}; \theta) - \mathrm{sg}[\mathbf{z}] \,\|_2^2 \quad \forall\, i \in I \tag{5}$$

where $\mathbf{z}$ is the linear combination of the codebook vectors, weighted using attention scores as stated in Eq 2. The intuition supporting the choice was to focus more on learning reliable frame embeddings while ensuring that they contain adequate information to enable the decoder to reconstruct the frames. As opposed to the loss function proposed in Eq 4 which focuses on increasing similarity between the encoder's output and the codebook vectors, this loss function emphasises more on improving the frame embedding, which in turn improves the representational capacity of the codebook vectors. However, none of the above attempts resulted in noticeable improvements.

7.2.4. Initialization and Attention Metric

Upon comparing different weight initializations like Xavier (Glorot and Bengio, 2010) and sampling from normal distributions, we concluded that the model performed better (with respect to minimizing the objective function) when using a unit normal initialization for all the weights. Also, using weight decay and employing AdamW optimizer (Loshchilov and Hutter, 2019) with 0.01 weight decay and 0.001 learning rate resulted in the best optimization. Further, regardless of the normalisation and preprocessing employed, the mode collapse persisted for all three datasets, proving the data agnostic nature of the problem.

As ablations for the attention metrics, we implemented dot product attention (Bahdanau et al., 2016), energy-based attention and Key-Query-Value-based attention (Vaswani et al., 2017). Also, in all the paradigms, we scaled the attention scores using a scaling factor with values ranging from the square root of the dimensionality (bottleneck) of the model to its inverse, as proposed by (Vaswani et al., 2017). Regardless of the metric and scale that were chosen, the problem of peaky attention score distributions persisted.

8. Conclusion

In this work, we propose a latent vector manipulation approach to video generation. Specifically, we introduce a novel Attention-based Discretized Autoencoder (ADAE) to learn a finite set of vectors. This set acts as the basis for the frame embeddings, which are sequentially modelled using a GPT.

However, through systematic experimentation, we conclude that our proposed model suffers from mode collapse, regardless of the hyperparameter choices and training paradigm. Upon probation, we concluded that the root cause for mode collapse is the peaky attention score distribution for the codebook vectors, with regards to the encoder's output. Hence, the ADAE collapses to a singular latent space, forcing the GPT to collapse. We validated the functionality of the GPT by using it to model over vector sequences and achieved performance at par with the original work. Further, modifications to the loss function and even the decoder architecture yielded no improvements. Through our exhaustive study, we establish that our approach requires reliable latent frame representations as a foundational prerequisite.

References

Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Towards high resolution video generation with progressive growing of sliced wasserstein gans, 2018.

Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey, 2019.

Md. Zahangir Alom, Tarek Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Nasrin, Mahmudul Hasan, Brian Essen, Abdul Awwal, and Vijayan Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8:292, 03 2019. doi: 10.3390/electronics8030292.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. Deep video generation, prediction and completion of human action sequences. *Lecture Notes in Computer Science*, page 374–390, 2018. ISSN 1611-3349. doi: 10.1007/978-3-030-01216-8_23. URL http://dx.doi.org/10.1007/978-3-030-01216-8_23.

Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model, 2017.

Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets, 2019.

Yatin Dandi, Aniket Das, Soumye Singhal, Vinay P. Namboodiri, and Piyush Rai. Jointly trained image and video generation using residual vectors, 2019a.

Yatin Dandi, Aniket Das, Soumye Singhal, Vinay P. Namboodiri, and Piyush Rai. Jointly trained image and video generation using residual vectors, 2019b.

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. doi: 10.1109/CVPR.2009.5206848.

Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. *arXiv*, 2017.

Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. *arXiv*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

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

Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv*, 2014.

Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical patch vae-gan: Generating diverse videos from a single sample, 2020.

Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control, 2018.

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

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2019.

Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. *arXiv*, 2018.

Yotam Intrator, Gilad Katz, and Asaf Shabtai. Mdgan: Boosting anomaly detection using multi-discriminator generative adversarial networks, 2018.

Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *arXiv*, 2016.

Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction, 2018.

Yitong Li, Martin Renqiang Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text, 2017.

Shixia Liu, Xiting Wang, Mengchen Liu, and Jun Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1):48 – 56, 2017. ISSN 2468-502X. doi: https://doi.org/10.1016/j.visinf.2017.01.006. URL http://www.sciencedirect.com/science/article/pii/S2468502X17300086.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

Pauline Luc, Aidan Clark, Sander Dieleman, Diego de Las Casas, Yotam Doron, Albin Cassirer, and Karen Simonyan. Transformation-based adversarial video prediction on large-scale data, 2020.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv*, 2016.

Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling, 2018.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

Gaurav Mittal, Tanya Marwah, and Vineeth N. Balasubramanian. Sync-draw. *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*, 2017. doi: 10.1145/3123266.3123309. URL http://dx.doi.org/10.1145/3123266.3123309.

Katsunori Ohnishi, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Hierarchical video generation from orthogonal information: Optical flow and texture. *arXiv*, 2017.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Viorica Patraucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. *arXiv*, 2016.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer, 2020.

MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv*, 2016.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019.

Masaki Saito and Shunta Saito. Tganv2: Efficient training of large models for video generation with multiple subsampling layers. *arXiv*, 11 2018.

Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping, 2017.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv*, 2015.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv*, 2016.

Hyontai SUG. Performance of machine learning algorithms and diversity in data. *MATEC Web of Conferences*, 210:04019, 01 2018. doi: 10.1051/matecconf/201821004019.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation, 2017.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *CoRR*, abs/1812.01717, 2018. URL http://arxiv.org/abs/1812.01717.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016.

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

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

C. Vondrick and A. Torralba. Generating the future with adversarial transformers. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2992–3000, 2017.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics, 2016.

Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. *arXiv*, 2020.

Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks, 2018.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363. PMLR, 2019. URL http://proceedings.mlr.press/v97/zhang19d.html.

Jiangning Zhang, Chao Xu, Liang Liu, Mengmeng Wang, Xia Wu, Yong Liu, and Yunliang Jiang. Dtvnet: Dynamic time-lapse video generation via single still image, 2020.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004.