Scaling autoregressive video models

Block-Seminar on deep learning for bio-medical data analysis
Problem description (what and why)

**Task:** Video generation

- High degree stochasticity
- Sheer amount of data

**Figure 1.** Video generation

\[
\begin{align*}
t-n & \quad \ldots \quad t-1 \quad t \\
\end{align*}
\]
Problem description (what and why)

**Video 1.** Perspective video prediction (Weissenborn et al., 2019)
Overview and method (how)

- Video specific models
- Latent variable models
- Auto regressive
- Adversarial training

**Scaling autoregressive video models**

16 x 64 x 64
Presentation overview

Background
- Autoregressive video
- Image subscaling

Contributions
- 3D video processing
- Subscale
- Block-level self attention

Experiments
- Datasets
- Experiments
- Metrics
- Results
Given the previous values $x_1, \ldots, x_t$ the model outputs the predictive probability distribution $P(x_{t+1}|x_1, \ldots, x_t)$ for $x_{t+1}$.

Why for video?

+ Conceptually simple
+ Tractable likelihood
- Pixel generated one at a time

Figure 2. Autoregressive generation ([Deep Ar Models](https://example.com), 2019)
Easier to compactly encode long-range dependencies across the many pixels in the large images

Subscale but with an explicit upscaling mechanism

Figure 4.B Downscaling and upscaling process
Take away: video forecasting with autoregressive approach on subscaled video
Contributions

I - Spatio-temporal video processing

II - Sub scaled 3D blocks

III - Local self attention in video

Figure 5. The architecture of the autoregressive video model (Weissenborn et al., 2019)
Overall architecture simplified

Figure 6. The simplified architecture of the autoregressive video model
Video represented as 3D spatio-temporal volumes

\[ \mathbf{x} \in \mathbb{R}^{T \times H \times W \times N_c} \]

Spatial neighborhoods around positions are maintained allowing a smaller receptive field of the attention mechanism

Figure 7. Video processing as a 3D object
Contribution II -- sub scaled 3D blocks

Figure 8. Subscaled video pipeline

Francisco Rivera (4772543)
Contribution III -- 3D local self attention (1)

Figure 9.1 Block attention

Francisco Rivera (4772543)
Contribution III -- 3D local self attention (2)

Figure 9.2 Block attention

\[
\tilde{z} = \left[ \text{attention}_1(z); \cdots; \text{attention}_{n_a}(z) \right] W_p + z \\
\begin{align*}
z' &= \text{relu} (\text{layernorm} (\tilde{z}) T_1) T_2 + \tilde{z} \\
&= \text{relu} (\text{layernorm} (\tilde{z}) T_1) T_2 + \tilde{z}
\end{align*}
\]
Overall architecture simplified

Figure 6. The simplified architecture of the autoregressive video model

Varying block size on each of the 8 layers allows information exchange between blocks

s1=4,8,4
s2=4,4,8
s3=1,32,4
s4=1,4,32
...
**Take away:** video (3D) memory consumption motivated to use block wise attention on a subscaled video
Datasets employed

**BAIR robot pushing**

- 40K training videos and 256 test videos
- Robot arm pushing and grasping objects in a box

**Kinetics-600**

- 400K Youtube videos ranging over 600 action classes
- Center-crop and down-sample each frame to 64x64 with a width-3 Lanczos filter and anti-aliasing

---

*Figure 10. Arm example (Berkeley, 2018)*

*Figure 11. Kinetics example (Carreira, 2017)*
Experimental setup

**Spatio-temporal subscaling**
- Full temporal subscaling
  - $s = (4, 2, 2)$
  - $16\times64\times64 \rightarrow 16$ slices of $4\times32\times32$

**Spatial subscaling**
- Shorter videos with frames subscaled to $32\times32$
  - $s = (1, 2, 2)$
  - $4\times64\times64 \rightarrow 4$ slices of $4\times32\times32$

**Single frame**
- No subscaling
  - $s = (16, 1, 1)$
  - $16\times64\times64 \rightarrow 16$ slices of $1\times64\times64$

(L)
- Hidden size increase from $d=512$ to $d=2048$
  - Attention heads from $na=8$ to $na=16$
1. Compute the negative log likelihood in base e

2. Apply change of base for converting log base e to log base 2

3. Then divide by the number of pixels (e.g. 3072 pixels for a 32x32 rgb image)
Generalization of the Fréchet distance, which measures how the generative model captures data distribution from which the observed data was generated.

\[
d(P_R, P_G) = \min_{X,Y} \mathbb{E}|X - Y|^2 \rightarrow |\mu_R - \mu_G|^2 + \text{Tr}\left(\Sigma_R + \Sigma_G - 2(\Sigma_R\Sigma_G)^{1/2}\right)
\]

Sample from PR (real data distribution)

I3D network pre-trained on Kinetics

Sample from PG (generative distribution)

Responses of:
- the logits in final layer
- output of last pool layer

Fit multivariate gaussian

Figure 12. FVD score
Subscaling can have a slightly negative effect on bits/dim.

Spatio-temporal subscaling shows lesser deformation on occluded objects.
<table>
<thead>
<tr>
<th>Models</th>
<th>Bits/dim</th>
<th>FVD</th>
<th>FVD (Avg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Frame</td>
<td>1.40</td>
<td>243±6</td>
<td>413±11</td>
</tr>
<tr>
<td>Spatial Sub.</td>
<td>1.47</td>
<td>263±6</td>
<td>450±15</td>
</tr>
<tr>
<td>Spatiotemp. Sub.</td>
<td>1.49</td>
<td>195±7</td>
<td>375±11</td>
</tr>
<tr>
<td>Single frame (L)</td>
<td>1.14</td>
<td>207±8</td>
<td>353±13</td>
</tr>
<tr>
<td>Spatiotemp. Sub. (L)</td>
<td>1.19</td>
<td>170±5</td>
<td>316±12</td>
</tr>
</tbody>
</table>

![Figure 14. Kinetics prediction (Weissenborn et al., 2019)](image-url)
Limitations

Not good results on the **full Kinetics** dataset -- cooking videos is an exception

- Failure modes on:
  - Freezing movement
  - Object distortions to continuations that completely break after a few frames

*Video 2*. Full Kinetics video prediction (Weissenborn et al., 2019)
Final remarks

- Employ a **3D sub block sampling** to videos so that **block-local self attention** is feasible

- **Expensive** to generate a pixel at a time

- First attempt at modeling **real-world videos** of an unprecedented complexity


Previous approaches and context

- Deterministic Approaches: For video prediction
- Generative Latent variables: Video models
- Optical flow: For video prediction
- Explicit motion model

---

Figure 15. Complete Background
Figure 1: **Left:** A visualization of the PixelCNN that maps a neighborhood of pixels to prediction for the next pixel. To generate pixel $x_i$ the model can only condition on the previously generated pixels $x_1, \ldots, x_{i-1}$. **Middle:** an example matrix that is used to mask the 5x5 filters to make sure the model cannot read pixels below (or strictly to the right) of the current pixel to make its predictions. **Right:** Top: PixelCNNs have a *blind spot* in the receptive field that can not be used to make predictions. Bottom: Two convolutional stacks (blue and purple) allow to capture the whole receptive field.

**Figure 16.** Masked convolution explained ([DeepMind](https://deepmind.com), 2016)
The illustrated transformer -- paths

Figure 17.A The illustrated transformer (Alammar, 2018)
The illustrated transformer -- qkv

Multiplying $x_1$ by the $WQ$ weight matrix produces $q_1$, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

**Figure 17.B** The illustrated transformer (Alammar, 2018)
The illustrated transformer -- z calculation

**Figure 17.C** The illustrated transformer (Alammar, 2018)
The illustrated transformer -- overview

1) This is our input sentence* 
2) We embed each word* 
3) Split into 8 heads. We multiply X or R with weight matrices 
4) Calculate attention using the resulting Q/K/V matrices 
5) Concatenate the resulting Z matrices, then multiply with weight matrix $W^o$ to produce the output of the layer

* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

---

**Figure 17.D** The illustrated transformer ([Alammar](#), 2018)
Separable attention crucially reduces the asymptotic memory cost from $O((HWT)^2)$ to $\max [O(H^2WT), O(HW^2T), O(HWT^2)]$ while still allowing the result of the module to contain features at each location accumulated from all other features at any spatio-temporal location.

**Figure 18.** Separable attention (Clark, 2019)
Generating long sequences with sparse transformers

Full self-attention for autoregressive models defines $S_i = \{j : j \leq i\}$, allowing every element to attend to all previous positions and its own position.

Factorized self-attention instead has $p$ separate attention heads, where the $m$th head defines a subset of the indices $A_i^{(m)} \subset \{j : j \leq i\}$ and lets $S_i = A_i^{(m)}$. We are chiefly interested in efficient choices for the subset $A$, where $|A_i^{(m)}| \propto \sqrt{n}$.

Figure 19. Sparse transformers (Child, 2019)
Figure 20. Block attention (Parmar, 2018)
Batch normalization normalizes the input features across the batch dimension.

The key feature of layer normalization is that it normalizes the inputs across the features.

**Figure 21.** Batch normalization  (Keitakurita, 2018)
Due to masking, no direct connection exist between this two pixels. → Creating a direct connection with a layer that extends both on width/height/2 frames is expensive → **No apparent repercussion if s > 1**
Lanczos filter: Filter that maps each sample of the given signal to a translated and scaled copy of the Lanczos kernel. The sum of these translated and scaled kernels is then evaluated at the desired points.

Lanczos kernel: which is a sinc function windowed by the central lobe of a second, longer, sinc function.

\[
L(x) = \begin{cases} 
\text{sinc}(x) \cdot \text{sinc}(x/a) & \text{if } -a < x < a, \\
0 & \text{otherwise.}
\end{cases}
\]

Figure 23. Lanczos filter (Wikipedia, 2020)
Figure 1: **Left: Multi-scale prior** The flow model uses a multi-scale architecture using several levels of stochastic variables. **Right: Autoregressive latent-dynamic prior** The input at each timestep $x_t$ is encoded into multiple levels of stochastic variables $(z_t^{(1)}, \ldots, z_t^{(L)})$. We model those levels through a sequential process $\prod_t \prod_l p(z_t^{(l)} | z_{<t}^{(l)}, z_t^{(>l)})$.

Generative Model that uses normalizing flows with markovian dynamics on the latent state of the system

Figure 24. VideoFlow architecture (Kumar, 2019)
Stochastic variational video prediction (SV2P) method that predicts a different possible future for each sample of its latent variables.

Figure 25. SV2P (Babaeizadeh, 2018)
Figure 3. The Inflated Inception-V1 architecture (left) and its detailed inception submodule (right). The strides of convolution and pooling operators are 1 where not specified, and batch normalization layers, ReLu’s and the softmax at the end are not shown. The theoretical sizes of receptive field sizes for a few layers in the network are provided in the format “time,x,y” – the units are frames and pixels. The predictions are obtained convolutionally in time and averaged.

Figure 26. Two Stream Inflated 3D Convolution  (Carreira, 2018)
“Employ a 3D local self attention mechanism on a sub block sampling fashion to predict unseen video frames”.

Figure 27. The architecture of the autoregressive video Model