Phenaki

A Preprint

1 Encoder Decoder C-ViViT

C-ViViT is a modifaction of ViViT that is itself is Video ViT. It consists of Encoder and Decoder. As input encoder gets $t_x + 1$ frames, each of size $h_x \times w_x \times c_x$. Then, first frame is treated independetly and split in patches $w_p \times h_p \times c_p$ and other frames are grouped as $w_p \times h_p \times c_p \times t_p$. This separation allows the first frame to be processed like an image (enabling text-to-image conditioning) as it uses only spatial attention on the first frame. Then patches are flattened and projected into d_z dimensional space. Thus

$$t_z = \frac{t_x}{t_p} \qquad w_z = \frac{w_x}{w_p} \qquad h_z = \frac{h_x}{h_p} \tag{1}$$

Resulting shape: $(t_z + 1) \times w_z \times h_z \times d_z$. Then it is followed by Spatial transformer: For each frame divide it into a grid of patches → flatten them into tokens → apply self-attention within that frame only; and Causal Temporal Transformer: For each spatial location (e.g., patch at top-left) track that patch across multiple frames. Coming to decoder, it is simply the same as Encoder, but upside down. The main idea that the decoder is given past frames as context (frozen tokens) and it must predict new tokens autoregressively.

2 Quantization

$$L_{VQ} = ||sg(z) - e||_2^2 + \beta||z - sg(e)||_2^2$$

where $sg(z) \equiv x$, $\frac{d}{dx}sg(x) \equiv 0$ - stop gradient operator. β - commitment loss weight, and e is a codebook vector from codebook E. he index to the codebook vector closest to z is found by $i = \text{argmin}_j||z - E_j||_2^2$

As for training:

$$L = L_{VQ} + 0.1L_{adv} + 0.1L_{ip} + 1L_{vp} + 1L_2$$

L_{VQ} is defined previously, L_{adv} is Adversarial Loss(Standard GAN loss using a discriminator D), L_{ip} is Image Perceptual Loss which is Feature-wise L2 loss using a pretrained CNN (e.g. VGG) over individual frames, L_{vp} is Video Perceptual Loss, basically same idea but applied using 3D CNN eg. I3D

3 T2V generation with bidirectional transformers

1. Training Strategy

Goal: Efficient text-to-video generation using a masked bidirectional transformer (MaskGIT-style) with significantly fewer sampling steps than autoregressive models.

- Input videos are encoded into discrete tokens $a \in \mathbb{N}^{(t_z+1)\times w_z \times h_z}$ using a pretrained C-ViViT encoder, flattened into a sequence of N tokens.

- At each training step i, a random mask ratio $\gamma_i \in [0, 1]$ is sampled.
- $\lceil \gamma_i \cdot N \rceil$ tokens are replaced with [MASK].
- The model minimizes cross-entropy loss over masked tokens, conditioned on:
  - Encoded text embeddings p,
  - Unmasked video tokens $\bar{a}_M$.
- Objective (MVTM loss):

$$L_{\text{mask}} = -\sum_{i=1}^{N} m_i \log p(a_i | \bar{a}_M, p)$$

  where $m_i = 1$ if a_i is masked.
- Classifier-free guidance: text condition is randomly dropped with 10% probability during training.
- To support both video and image data:
  - If only a single frame: masking is applied to first $w_z \cdot h_z$ tokens.
  - If full video: masking is applied to all tokens.

2. Inference Procedure

- All video tokens are initialized as [MASK].
- At each sampling step i, the model:
  - Predicts all unknown tokens in parallel.
  - Keeps a fraction β_i of confident predictions.
  - Re-masks remaining tokens and repeats.
- Sampling typically converges in 12–48 steps.
- Guidance scale λ controls strength of text-video alignment.

3. Long Video Extrapolation

- After initial video generation:
  - Encode last K frames using C-ViViT.
  - Initialize masked transformer with their tokens.
  - Generate the next segment of video tokens.
- Text prompt can be updated to guide visual transitions and storytelling.

Algorithm 1 Phenaki Video Generation with Masked Transformer

Require: Text prompt p, sampling steps T, masking schedule $\{\gamma_i\}_{i=1}^{T}$, sampling schedule $\{\beta_i\}_{i=1}^{T}$
Ensure: Generated video tokens a

```
Initialize all video tokens a_0 ← [MASK]
for i = 1 to T do
    Predict masked tokens â_i ~ p(a_i | ā_M, p)
    Select top β_i fraction of most confident predictions
    Replace selected masked tokens in a_{i-1} with predicted values to get a_i
    Re-mask remaining tokens in a_i
    Decode final video tokens a_T using VQ decoder to obtain video frames return a_T
```