# October Progress Update

## Video-Audio-Text Predictive Coding

Aman Bhargava, Sept ‘22

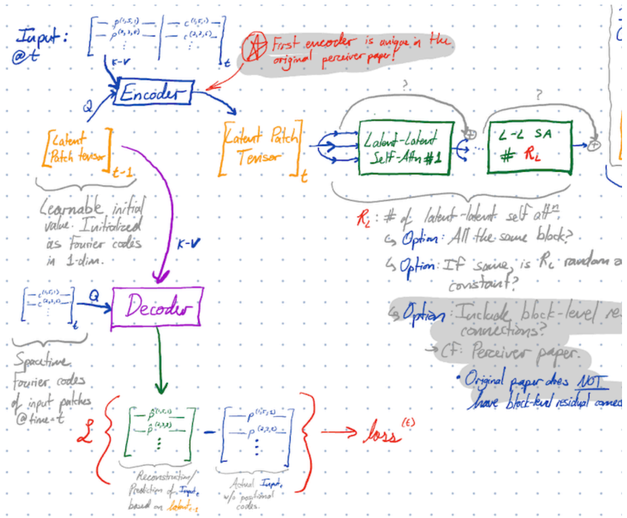
# I: Overview

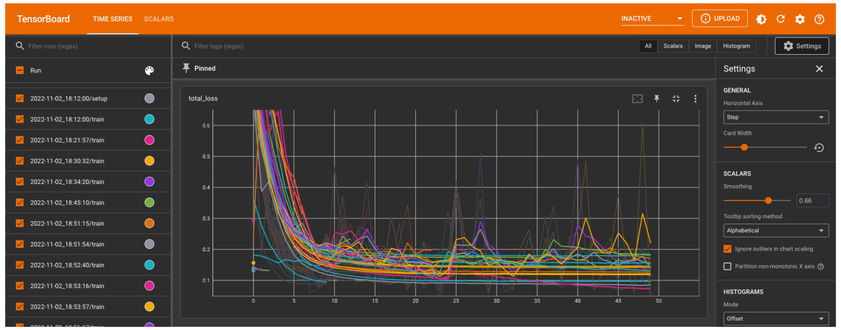
# Project background

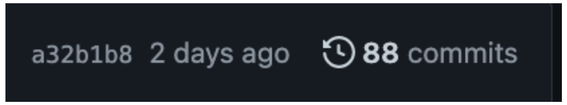
- **Current goal**: Multimodal predictive coding machine. Can we build a system that can **understand** dynamics of audio-video-narration via self-supervision?
- **Downstream questions/applications**:
  - *Few-shot object and action recognition.*
  - *Generating language tokens to aid in audio-video-narration prediction.*
  - *Example: model that can share environmental information with other agents via learned/induced language.*

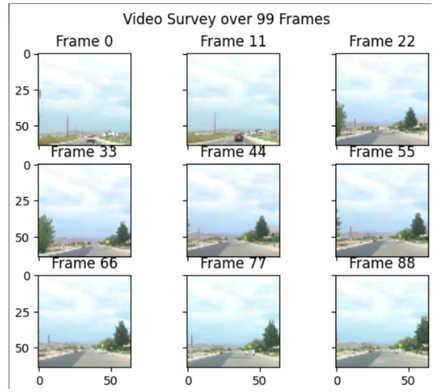
# Progress Summary

- Designed & implemented **dynamical perceiver autoencoder** for video.
- Implemented **training loop** for predictive coding.
- Tested functionality via **overfitting** tests, small overnight tests.
- Added **profiling & training dashboard**: revealed input bottleneck.



# Help Needed

## Guiding questions for the update

- What **performance metric**(s)/downstream task(s) are we actually shooting for?
  - *Video autoencoding/predictive coding isn't common — what do we compare to?*
  - *Relevant for design decisions in training, validation, architectures* — e.g., present vs. future window sizing.
- How/when to incorporate **text + audio**? When is video "working" well enough?
- Addressing long training time: add convolutions? Reduce problem complexity?

# II: Progress

*Model & Training*
*Preliminary Tests*
*Software Hurdles*

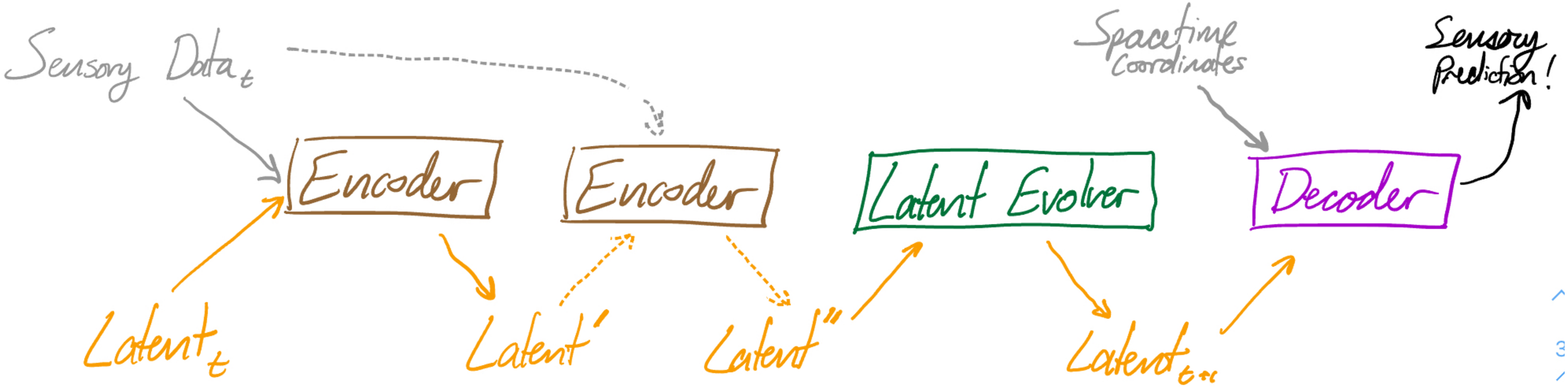
# Model Architecture

## Overview

① Encoder : Sensory Input × Latent → Latent

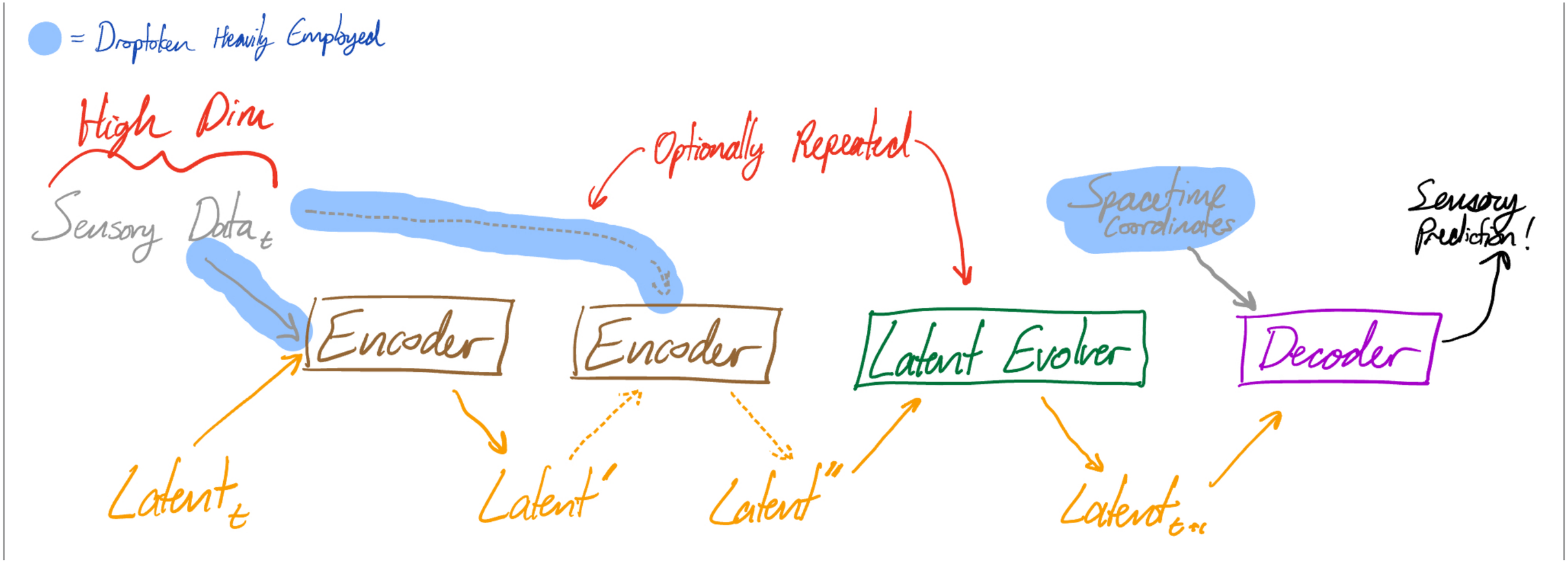
② Latent Evolver : Latent × Latent → Latent

③ Decoder : Spacetime Coords × Latent → Sensory Prediction @ spacetime coordinates



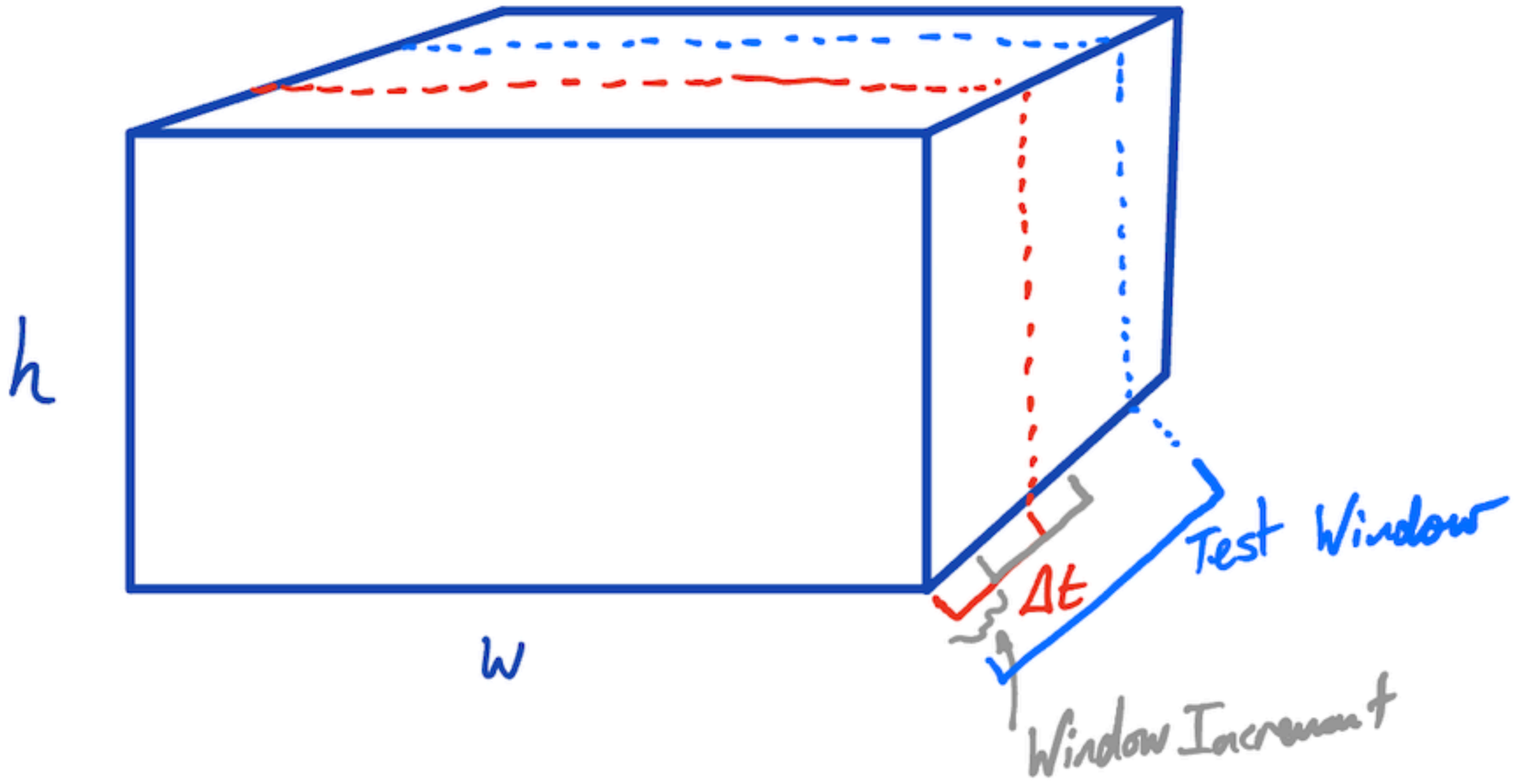
# Model Architecture

## Overview



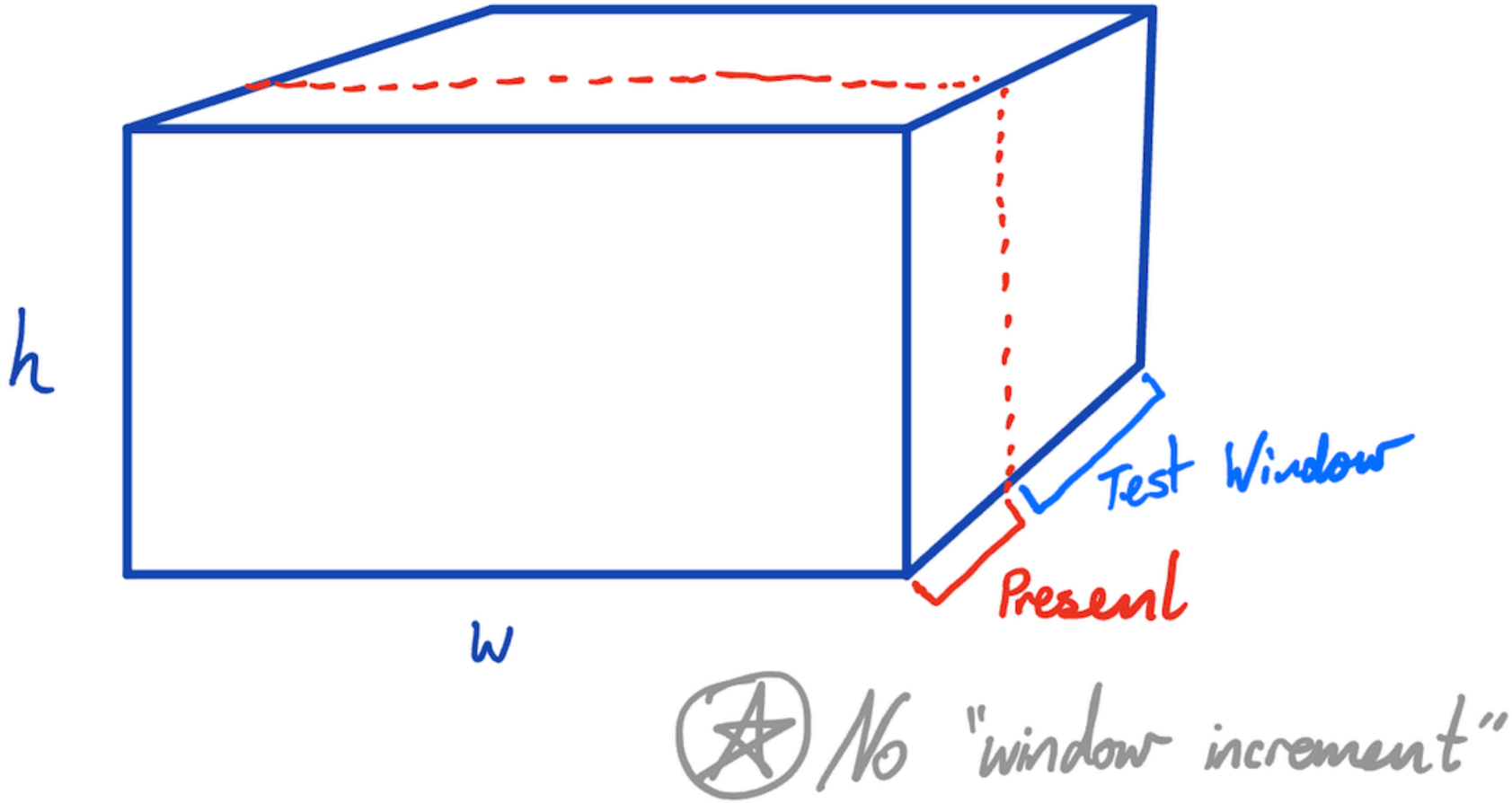
# Model Architecture

## Training Scheme I



# Model Architecture

## Training Scheme II



# Model Architecture

## Primary Optimization (Perceiver)

TRANSFORMER STACK KEY PRINCIPLE

- $Q_{in} \in \mathbb{R}^{N \times C}$ // Queries
- $K_{in} \in \mathbb{R}^{M \times D}$ // Keys

$$\Rightarrow \quad Q = Q_{in} W_Q \in \mathbb{R}^{N \times D_{head}}$$
$$K = K_{in} W_K \in \mathbb{R}^{M \times D_{head}}$$
$$V = K_{in} W_V \in \mathbb{R}^{M \times D_{out}}$$

$$Attn(Q_{in}, K_{in}) = Softmax(QK^T)V$$

$QK^T$: $N \times M$

$Softmax(QK^T)V$: $N \times D_{out}$

$$\Rightarrow O(MN) \text{ complexity}$$

Videos are high-dim.
We need either **low N** or **low M** to process.

*Row-wise tokenization: **M, N** are the number of tokens in the queries/keys respectively.

# Model Architecture

## Primary Optimization (Us vs. Perceiver)

### Transformer Stack Key Principle

- $Q_{in} \in \mathbb{R}^{N \times C}$ // Queries
- $K_{in} \in \mathbb{R}^{M \times D}$ // Keys

$$\Rightarrow \quad Q = Q_{in} W_Q \in \mathbb{R}^{N \times D_{head}}$$
$$K = K_{in} W_K \in \mathbb{R}^{M \times D_{head}}$$
$$V = K_{in} W_V \in \mathbb{R}^{M \times D_{out}}$$

$$Attn(Q_{in}, K_{in}) = Softmax(QK^T)V$$

$QK^T$: $N \times M$

Result: $N \times D_{out}$

$$\Rightarrow O(MN) \text{ complexity}$$

Videos are high-dim.
We need either **low N** or **low M** to process.

**Perceiver**: Use small latent **N**, do processing on that.
**Us**: Video still has massive **M** — must optimize further.

*Row-wise tokenization: **M, N** are the number of tokens in the queries/keys respectively.

# Model Architecture

## Additional Optimizations — Reducing M

- **Droptoken**: We can drastically reduce **M** (num input tokens) by randomly dropping a portion of them.
- **Iterative re-exposure**: We can re-expose the model to the input to offset the information cost of drop token (i.e., reselect dropped tokens per exposure).
- **Predict randomly sampled future events**: Instead of reconstructing every future input token, only query with a random selection of spacetime codes.
  - The model can only minimize the expected value of prediction loss across random future tokens, hence minimizing loss across all future tokens.
  - Decoder directly decodes future patches — no iteration required.

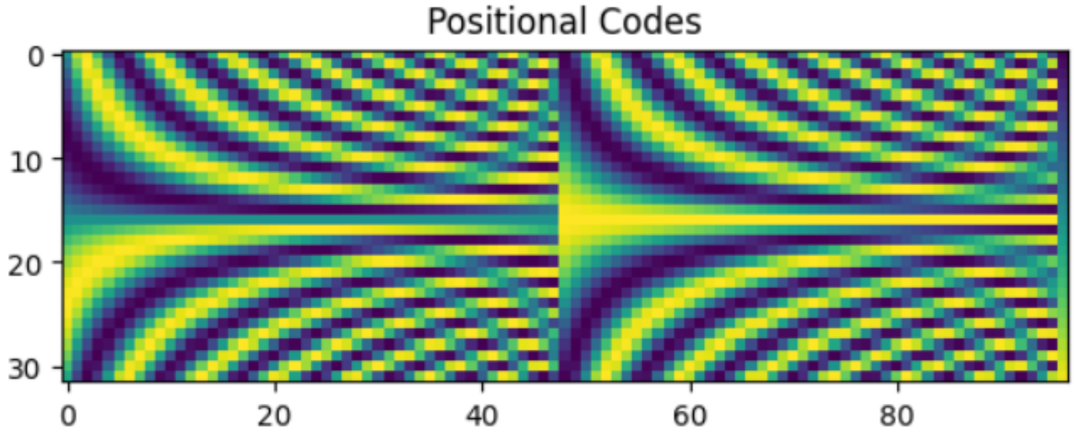
# Model Architecture

## Specifics: Encoder & Decoder

- **Transformer block**: {MHA -> FNN} with layer norm + residuals around each sub-block.
- **Encoder/Decoder**: Sequence of transformer blocks.
  - **Encoder**: Query = latent, value = input bytes. Token size is set by query, residuals carry forward latent state values from before.
  - **Decoder**: Query = spacetime codes, value = latent. Token dimensionality is **expanded** to match source patches in one of the TF block's FFN (usually the final one due to cost).

# Model Architecture

## Specifics: Spacetime Codes




From Stanley '07, Vaswani '17, ...

VALUES: $\begin{bmatrix} \sin(f_k \pi x_d) \\ \cos(f_k \pi x_d) \end{bmatrix}$

- $f_k$ = $k^{th}$ freq band
  - ↳ Uniform $\in [1, 1/2]$
  - ↳ $1/2 \equiv$ Nyquist frequency ($1$ = max frequency you can resolve)
- $x_d$ : Input token idx
  - ↳ $x_d \in [-1, 1]$ on each dim, $d$ = idx # $\in [M]$

↳ Final positional encoding $\equiv$

$$\left\{ \begin{bmatrix} \sin(f_k \pi x_d) \\ \cos(f_k \pi x_d) \end{bmatrix}_{k=1}^{K}, f_k \in [1, 1/2] \right\} \in \mathbb{R}^{d(2K+1)}$$

$x_d$

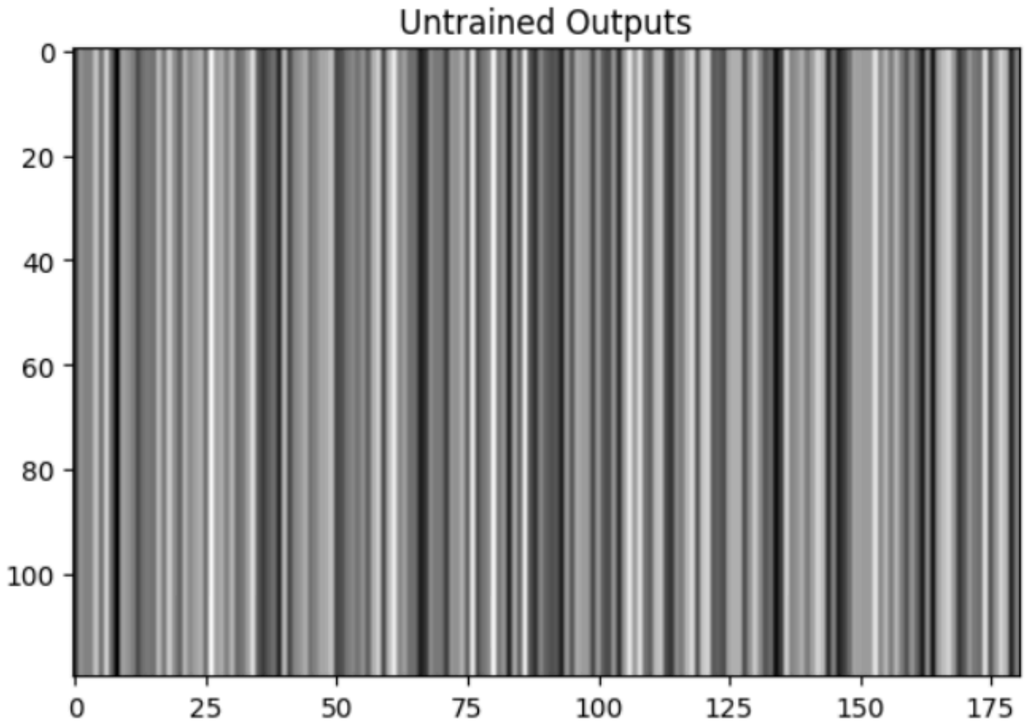
# Model Architecture

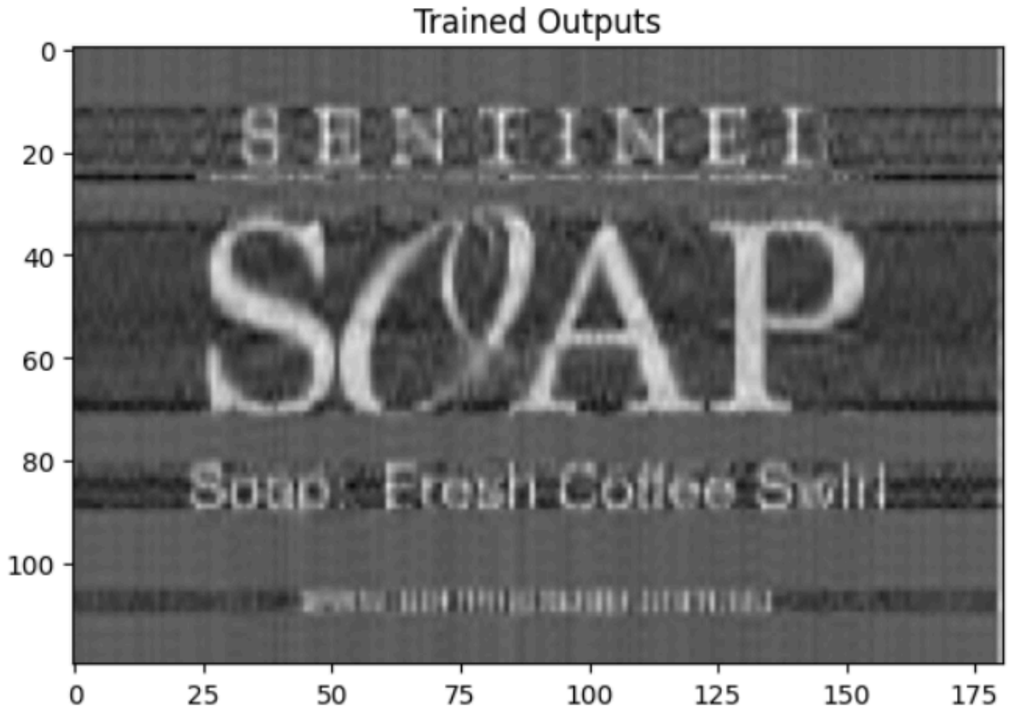
## Misc. Design Decisions/Considerations — Cf. 08_Perceiver_AE.ipynb

- Token expansion in decoder prevents residual connection.
- Latent evolver can either have identical or distinct sub-blocks.
- Initial latent state value is now learnable.
- Temporality is only introduced in training loop — the architecture does not demand it.

# Preliminary Overfitting Tests

## No positional codes, no patch generation.

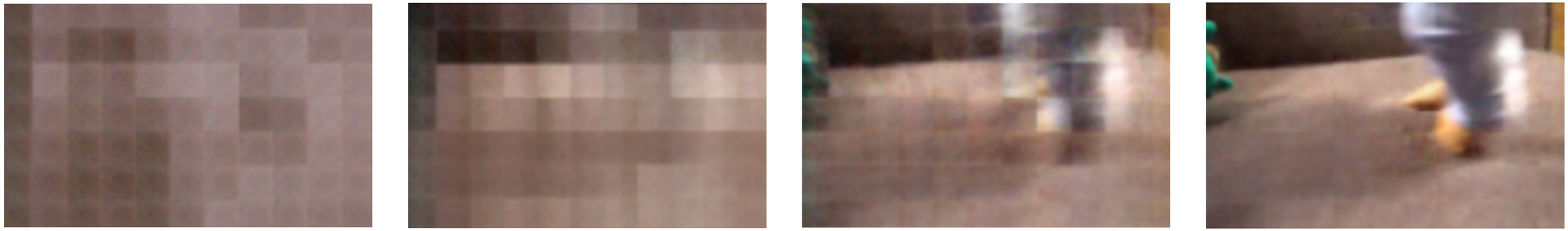



*Barcode pattern == all output tokens are identical!

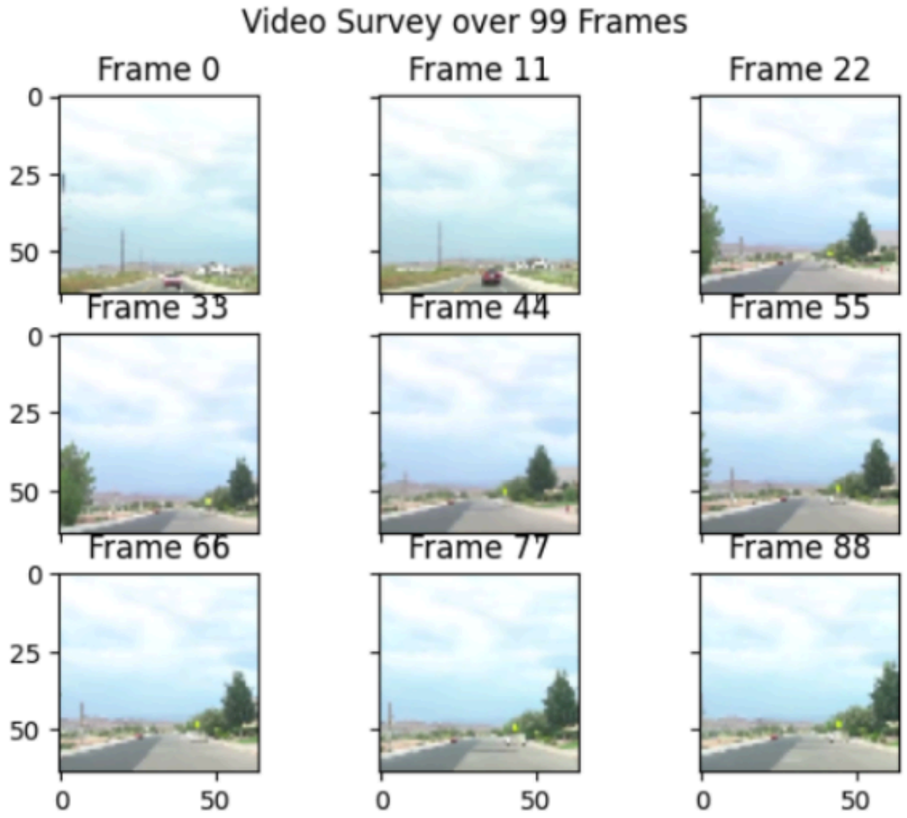
# Preliminary Overfitting Tests

**With positional codes, proper patch generation.**

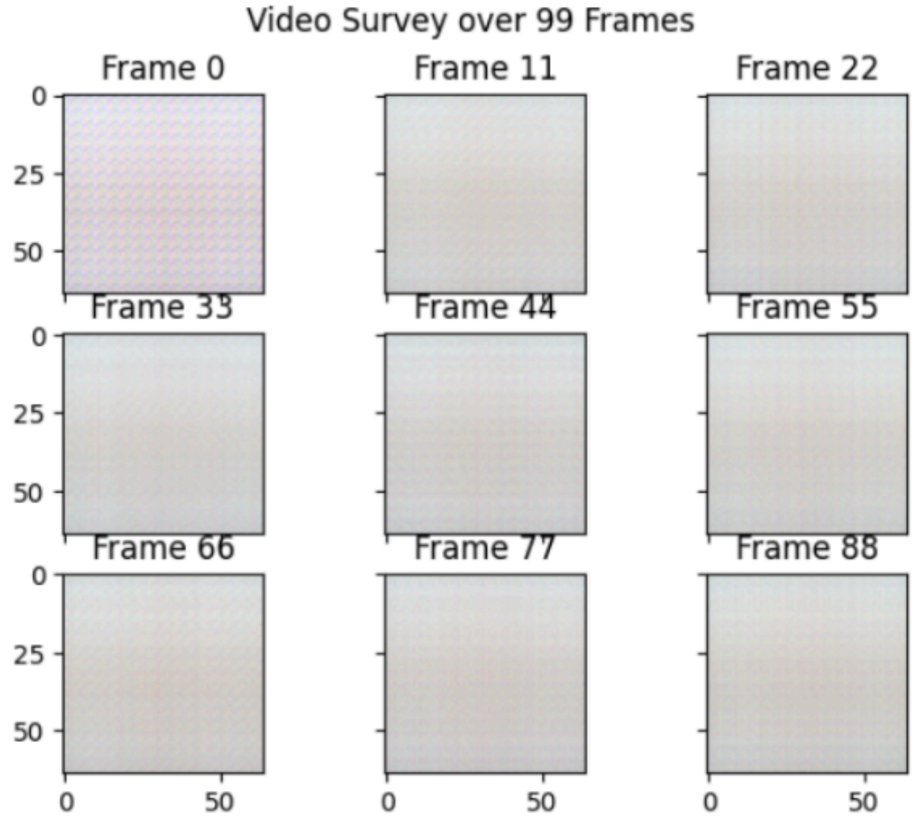


# 3-Hour Training 1

## Training Strategy I, Default Parameters




Ground Truth




Reconstruction

# Training Parameters

## Ever Expanding List

```
Train model M1 (perceiver AE).

optional arguments:
  -h, --help            show this help message and exit
  --output-folder OUTPUT_FOLDER
                        Folder to keep checkpoints, loss plots,
                        records of arguments, etc. The folder
                        will be created if it doesn't exist. To
                        make a folder with the current date of
                        format `2022-10-26_12:30:33`, add
                        `{now}` to the end of the string.
                        Default=../training/debug/{now}.
  --data-folder DATA_FOLDER
                        Folder containing the `.mp4` video files
                        to use for the experiment.
  --num-frames NUM_FRAMES
                        Number of frames to gather from each
                        video for the dataset. Default=100.
  --frame-size FRAME_SIZE
                        `height,width` of each video frame.
```

# Training Parameters

## Ever Expanding List

```
--patch-hwd PATCH_HWD
                      `patch_height,patch_width,patch_duration
                      `. Default=16,16,3
--batch-size BATCH_SIZE
                      Size of the batch (number of video
                      tensors per training batch). Default=10
--num-prefetch NUM_PREFETCH
                      Number of dataset batches that are pre-
                      fetched. Default=4.
--k-mu-space K_MU_SPACE
                      `k,mu` for spatial Fourier codes.
                      Default=15,20.
--k-mu-time K_MU_TIME
                      `k,mu` for temporal Fourier codes.
                      Default=64,200
--overfit OVERFIT     Take the first `n` videos from `mp4list`
                      and overfit the model on those.
                      Default=-1 (i.e. use the full dataset)
--cpu-only            Include this flag to force training to
                      use only CPU.
```

# Training Parameters

## Ever Expanding List

```
--one-gpu             Include this flag to force training to
                      use only one GPU.
--ckpt-period CKPT_PERIOD
                      Number of iterations separating each
                      checkpoint. Default=50
--num-iters NUM_ITERS
                      Number of total iterations. Each
                      iteration is one training step on a
                      batch of `--batch-size` videos each with
                      `--num-frames`. Default=200
--lr LR               Primary optimizer learning rate.
                      Default=0.001
--alpha ALPHA         Weighting between present timewindow
                      prediction vs. far future prediction
                      errors. 1 -> only present, 0 -> only
                      future. Default=0.7
--blind-iters BLIND_ITERS
                      Number of exposures at the beginning of
                      procesing a video sequence that is NOT
                      counted towards loss.
```

# Training Parameters

## Ever Expanding List

```
--present PRESENT     Number of frames in the `present` time
                      window. Default=5.
--future FUTURE       Number of frames in the `future` time
                      window. Default=30.
--future-selection-probability FUTURE_SELECTION_PROBABILITY
                      Probability of a token from the future
                      time window being selected. Default=0.1
--window-inc WINDOW_INC
                      How much the `present` and `future` time
                      windows are incremented each iteration
                      in FRAMES. Default=5.
--restore-from RESTORE_FROM
                      Path to an experiment directory. We will
                      look at the `checkpoints` subdirectory
                      and start from the most recent one.
--latent-dims LATENT_DIMS
                      Dimensions of the latent
                      state/predictive code tensor in the
                      model. Comma separated
                      `num_tokens,token_dim`. Default=100,700
```

# Training Parameters

## Ever Expanding List

```
--nheads NHEADS
--keydim KEYDIM
--mhadropout MHADROPOUT
--n-enc-blocks N_ENC_BLOCKS
                      Number of encoder blocks in the model.
                      Default=3.
--p-droptoken P_DROPTOKEN
                      Expected portion of input tokens
                      retained on each exposure of the latent
                      state. Default=0.5.
--no-re-droptoken     Include this flag if you do NOT want the
                      dropped tokens to be resampled on each
                      exposure.
--n-latent-blocks N_LATENT_BLOCKS
                      Number of transformer blocks in the
                      latent module.
--distinct-latent     Include this flag to make each latent
                      block distinct (non-identical)
--n-dec-blocks N_DEC_BLOCKS
                      Number of transformer blocks in the
```

# Software Hurdles

## Resolved

- Video un-patching/un-flattening is hard.
- **Modularization**: custom layers for encoder, decoder, latent evolver — swappable + independent.
- **Checkpointing**: Caching + retrieving model parameters is smooth, data preparation parameters too!
- **Dashboard**: loss, tensor statistics, etc. are viewable in real time.
- **Profiling**: GPU + host statistics on idle time, IO, memory utilization/ fragmentation, etc. available via dashboard.

TensorBoard TIME SERIES SCALARS INACTIVE UPLOAD

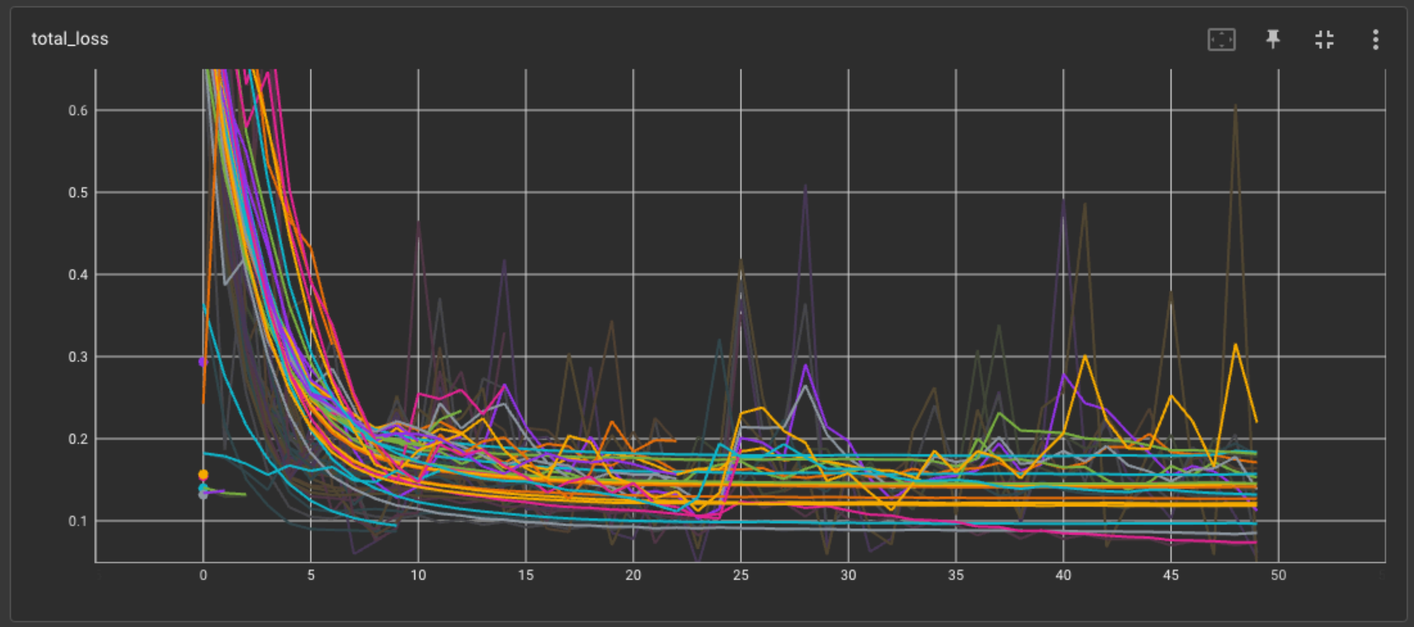
Filter runs (regex)

Filter tags (regex) All Scalars Image Histogram Settings

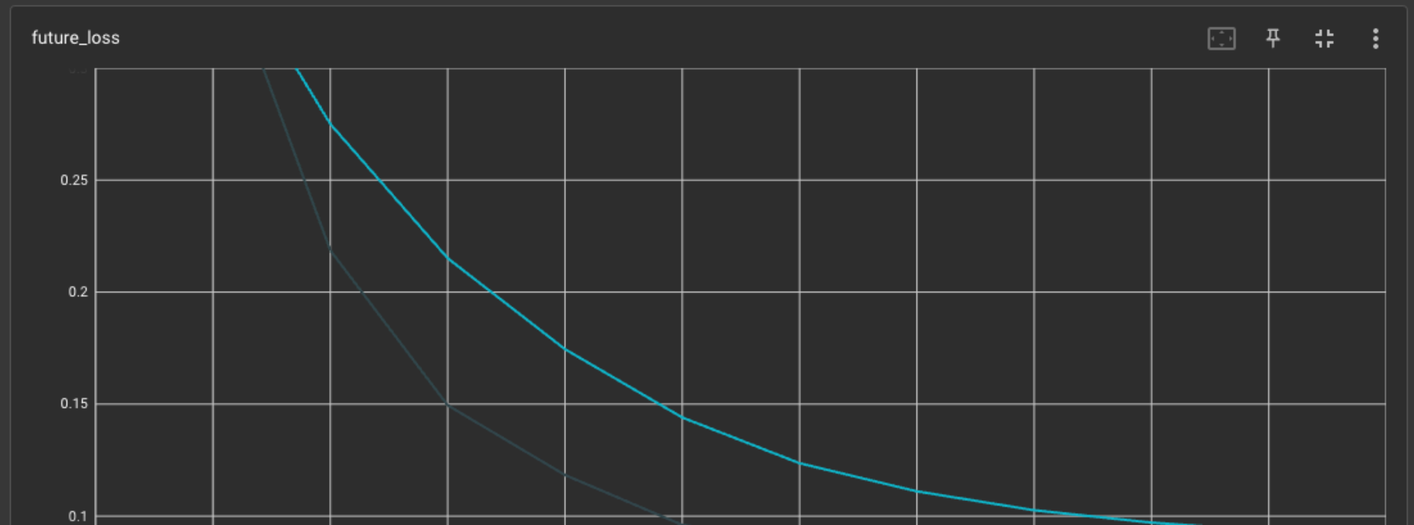
Run

2022-11-02_18:12:00/setup
2022-11-02_18:12:00/train
2022-11-02_18:21:57/train
2022-11-02_18:30:32/train
2022-11-02_18:34:20/train
2022-11-02_18:45:10/train
2022-11-02_18:51:15/train
2022-11-02_18:51:54/train
2022-11-02_18:52:40/train
2022-11-02_18:53:16/train
2022-11-02_18:53:57/train
2022-11-02_18:56:17/train
2022-11-02_18:57:25/train
2022-11-02_18:57:53/train
2022-11-02_18:58:51/train
2022-11-02_19:00:54/train
2022-11-02_19:01:45/train
2022-11-02_19:03:05/train
2022-11-02_19:04:18/train
2022-11-02_19:07:08/train
2022-11-02_19:08:28/train
2022-11-02_19:11:44/train

Pinned




future_loss



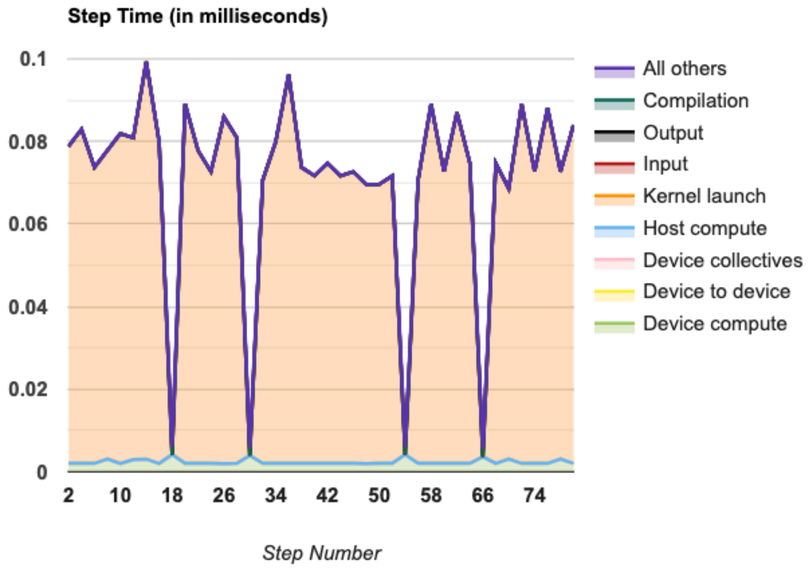

Settings

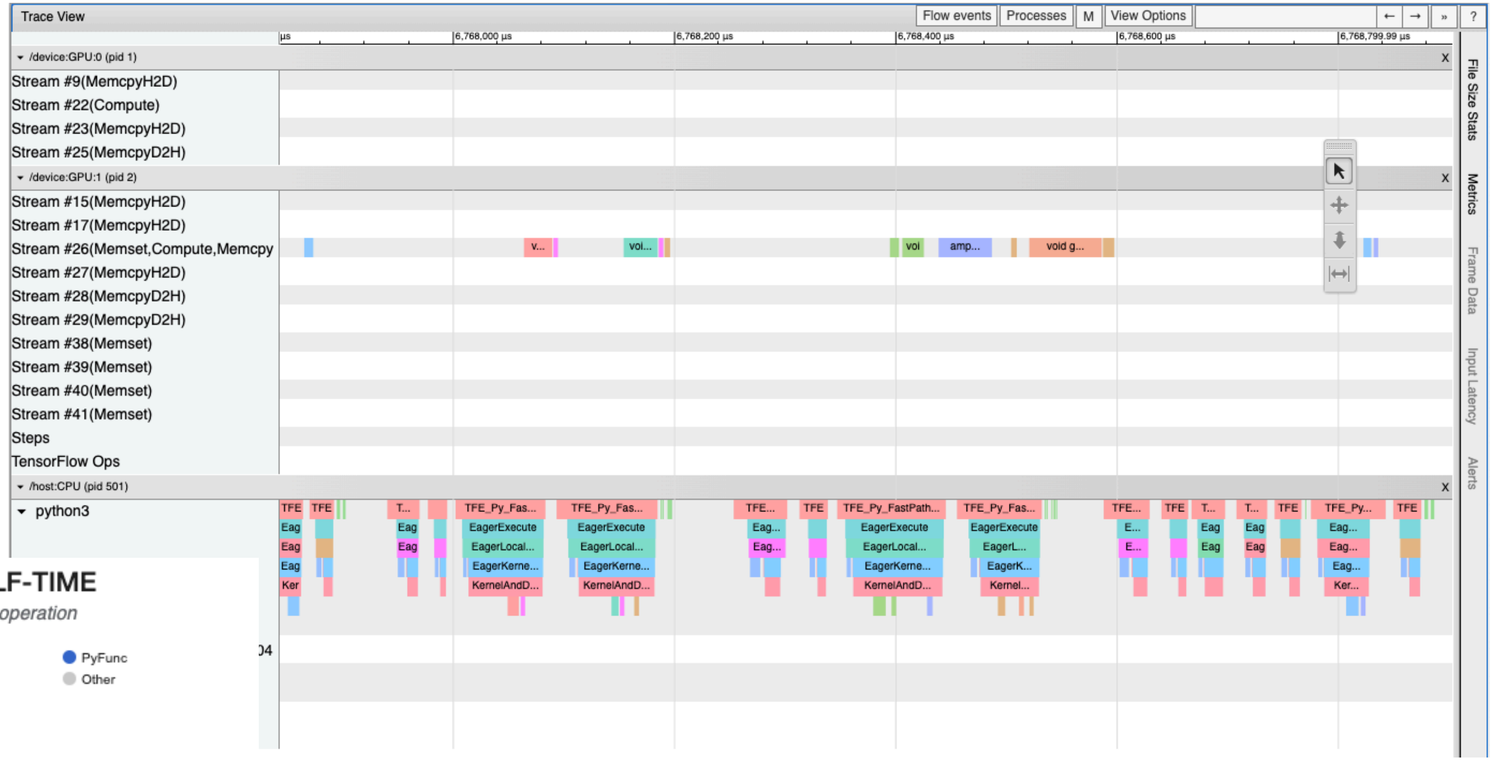
GENERAL
Horizontal Axis
Step
Card Width

SCALARS
Smoothing
0.66
Tooltip sorting method
Alphabetical
Ignore outliers in chart scaling
Partition non-monotonic X axis

HISTOGRAMS
Mode
Offset

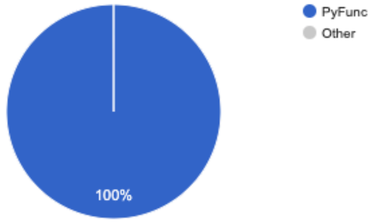
IMAGES
Brightness
Contrast
Show actual image size



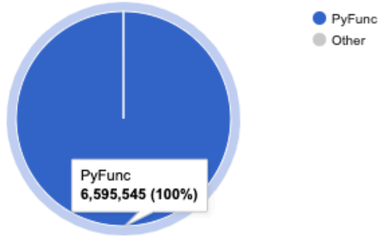

## ON DEVICE: TOTAL SELF-TIME (GROUPED BY TYPE)

*(in microseconds) of a TensorFlow operation type*



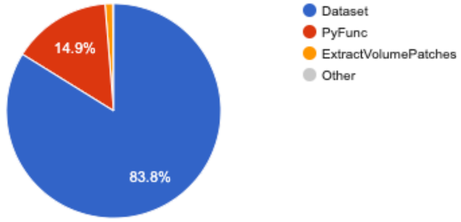

## ON DEVICE: TOTAL SELF-TIME

*(in microseconds) of a TensorFlow operation*



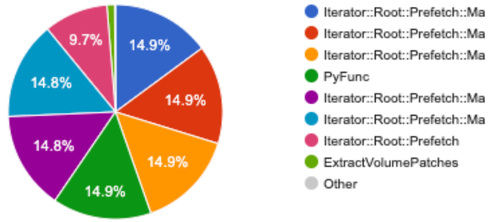

## ON HOST: TOTAL SELF-TIME (GROUPED BY TYPE)

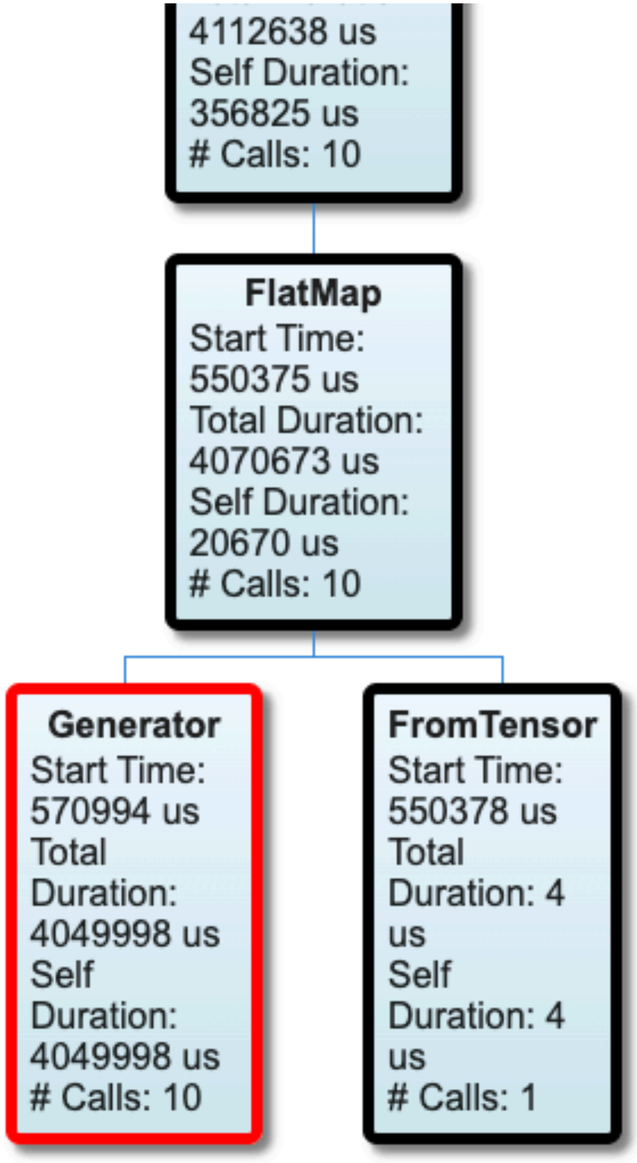
*(in microseconds) of a TensorFlow operation type*




## ON HOST: TOTAL SELF-TIME

*(in microseconds) of a TensorFlow operation*

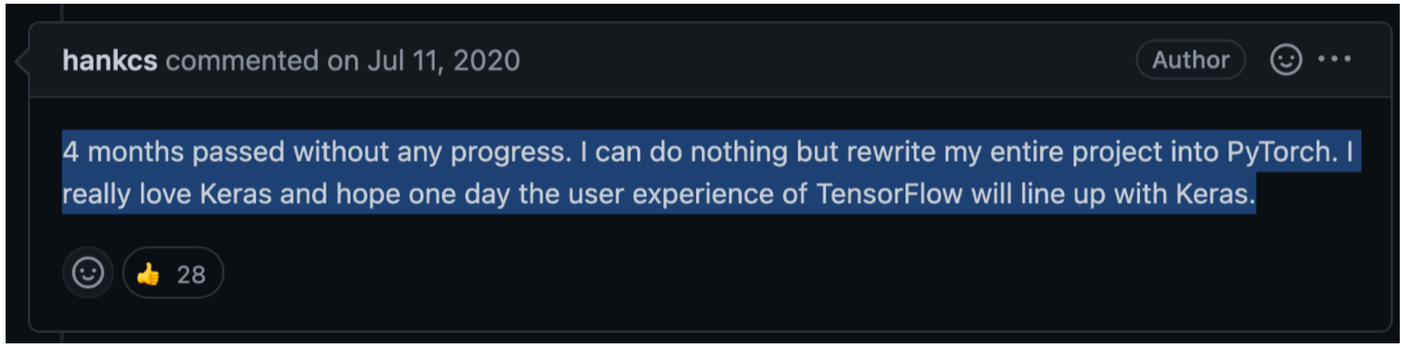





| Max Latency (us) | Bottleneck |
|---|---|
| 4,122,328 | Iterator Type: **Generator**<br>Long Name: Iterator::Root::Prefetch::MapAndBatch::ParallelMapV2::ParallelMapV2::ParallelMapV2::FlatMap[0]::Generator<br>Latency: 4,049,998 us |

## Suggestion

1. Check the locality of a host and input data. Ideally, they should be in the same cell (or very close, like the same region).
2. Parallelize reading from this dataset source. See here and here for more details.



# Software Hurdles

## Unresolved

- **Data loading**: Tensorflow dataset abstraction is too slow.
  - Memory leaks, sparse documentation, video is uncommon/expensive compression…
  - Next step: Build a parallelized data loader from scratch (`TFRecord` is uncompressed).
- **Hyperparameter search**: Training takes a very long time.
  - Transformers take a long time.
  - High cost to experiment with hyperparams — Bayesian hyperparam search? RandomSearch? Grid search? RL scheduler? Early stopping?
    - *Cost of fancy search vs. Cost of inefficient simple search.*

# III: Help Needed

Questions
“Street fighting math”
Initial thoughts

# Help Needed

## Guiding questions for the update

- What **performance metric**(s)/downstream task(s) are we actually shooting for?
  - *Video autoencoding/predictive coding isn't common — what do we compare to?*
  - *Relevant for design decisions in training, validation, architectures* — e.g., present vs. future window sizing, training scheme I vs. II.
- How/when to incorporate **text + audio**? When is video "working" well enough?
- Addressing **long training time**: add convolutions? Reduce problem complexity? Hyperparam search?

# Street Fighting Math

- [16px, 16px, 3frame] patch = 2304-dim token.
- 3 frames of 120x180 video = 77 tokens.
- 1 second of 120x180 video = 616 tokens (24fps).
- 10 minutes of 120x180 video = 369,600 tokens.
  - = 851,558,400 32-bit values = 3,406,233,600 bytes = ~3 GB

# Initial Thoughts

- **End goals/getting *a* result**:
  - Training scheme II is inviting.
  - Scaling down the window sizes approaches ViT-MAE task, but with an **important & interesting** novel addition. We can still use more audio + text!
  - If we can make it work with bigger windows, that's great — but it's interesting & novel either way (pending more focused lit review).
  - We just need to find a use for the predictive codes…
  - Helps with long training time, too — start simple, get more ambitious in problem scope.
- **Drawback**: training II may be harder to integrate with ESN's than training I.
  - Worth separating into {MMF} and {sequence investigations}? Eventually merging?

# Initial Thoughts

- **End goals: Usually a downstream standardized task** via transfer learning.
- Latent embeddings are *probably* going to be good IF the model is good.
  - *Probably* will contain high information for cross-modal tasks ({action classification from video, image classification, waveform classification} for VATT).
- Will require quite a bit more time to test, but seems to be the mainstream validation method.
- **Very cool question**: does predictive coding (as opposed to pure auto encoding) yield superior representations?