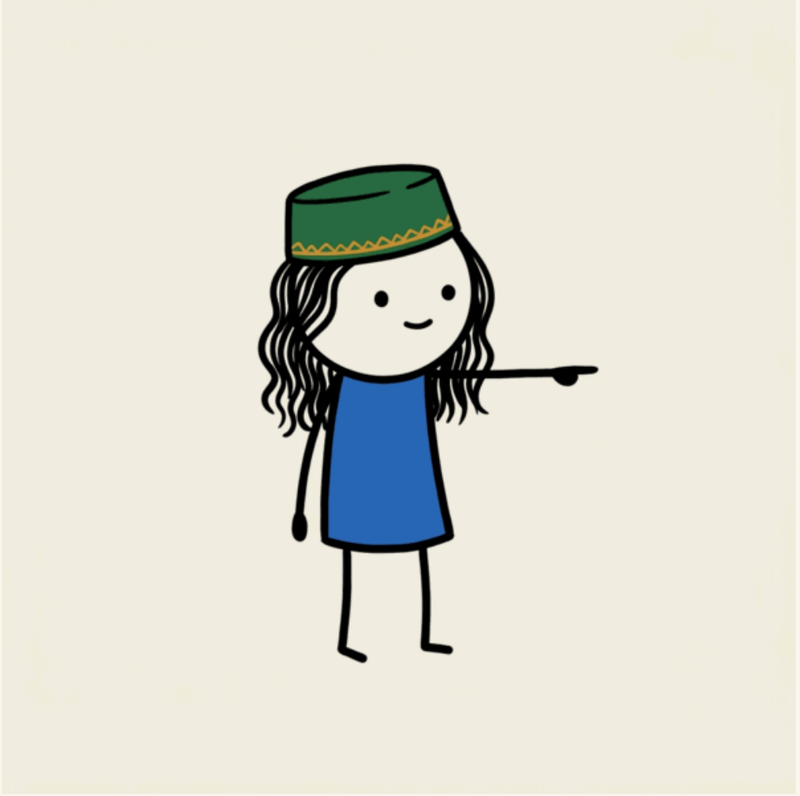
# What is the signal telling us?

Match the output contract and evaluation protocol to the operational decision.

From time-series tasks
to learned representations
and foundation models.

Lecture 2 · 90 minutes



# Serega has a clever forecaster. Is the pump healthy?

It predicts the next reading accurately.
Does that prove normal operation?

> Keep this question open until the final experiment.



# Five tasks. Five different outputs

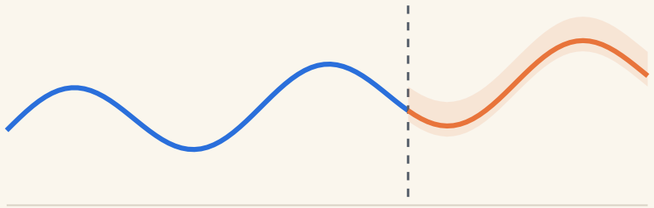
Choose the required output before choosing the model.

## Classification



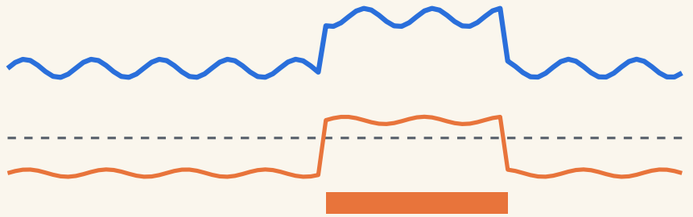

One observed window → one class

## Forecasting



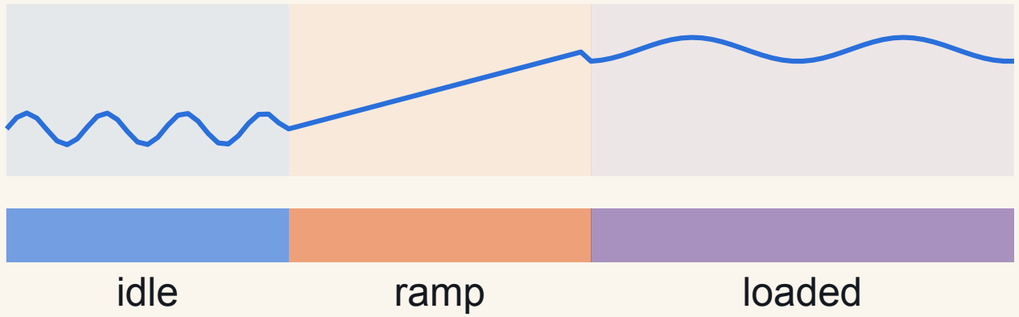
History → future values + interval

## Anomaly detection



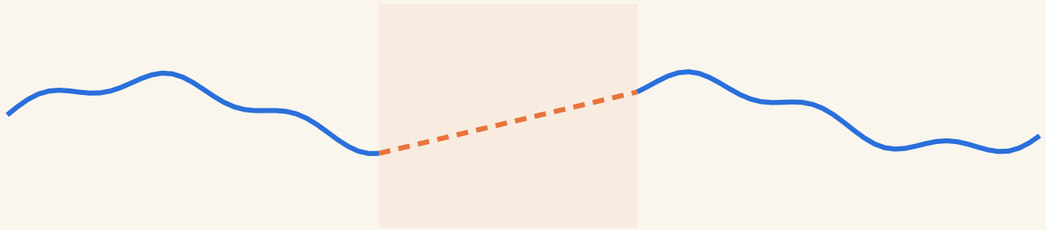
Score → threshold → alarms

## Segmentation




One state label for each time position

## Imputation



missing → filled

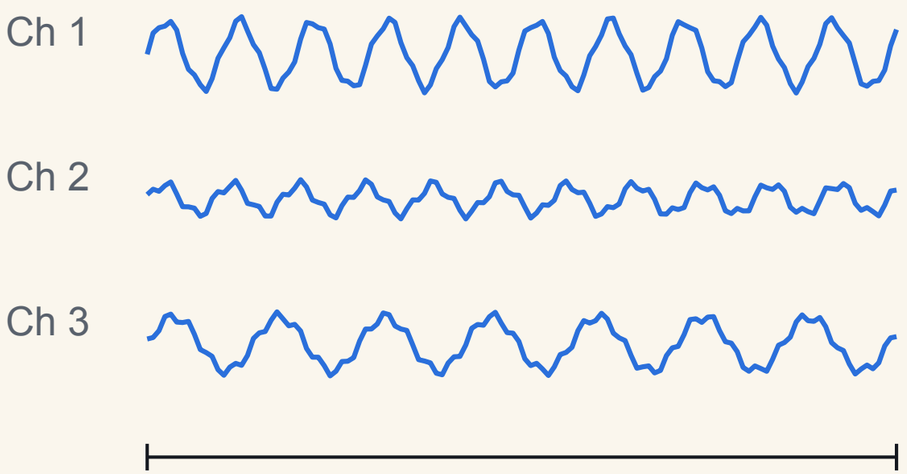
Observed samples stay unchanged

# Classification: a label for an observed window

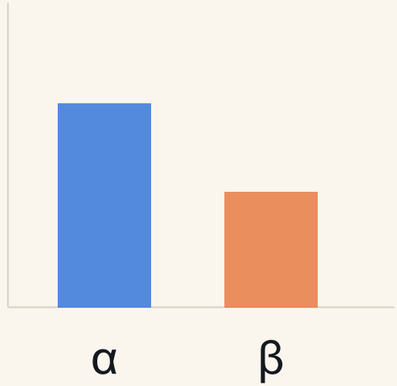
**One observed epoch → one class.** Bandpower + logistic regression; MiniROCKET features or learned EEGNet features.

## Fully observed EEG epoch

Illustrative motor-imagery example




## Feature baseline




Bandpower
+ logistic regression

## One output

Example target:

**left imagery**

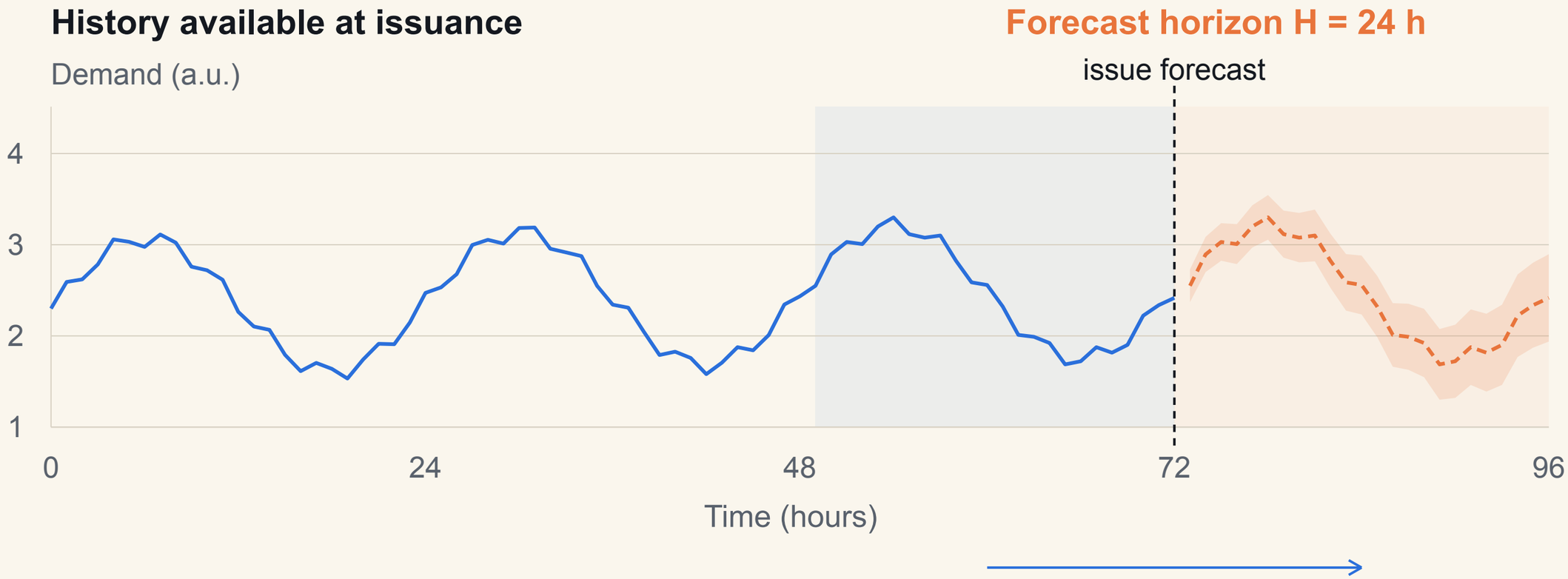
One class per epoch
after observation

## Other feature routes: MiniROCKET · EEGNet

Synthetic traces and illustrative feature values; no trained accuracy or prediction is claimed.

# Forecasting: predict values that have not arrived

**History → future horizon.** Compare every model at the same issuance time and forecast horizon.

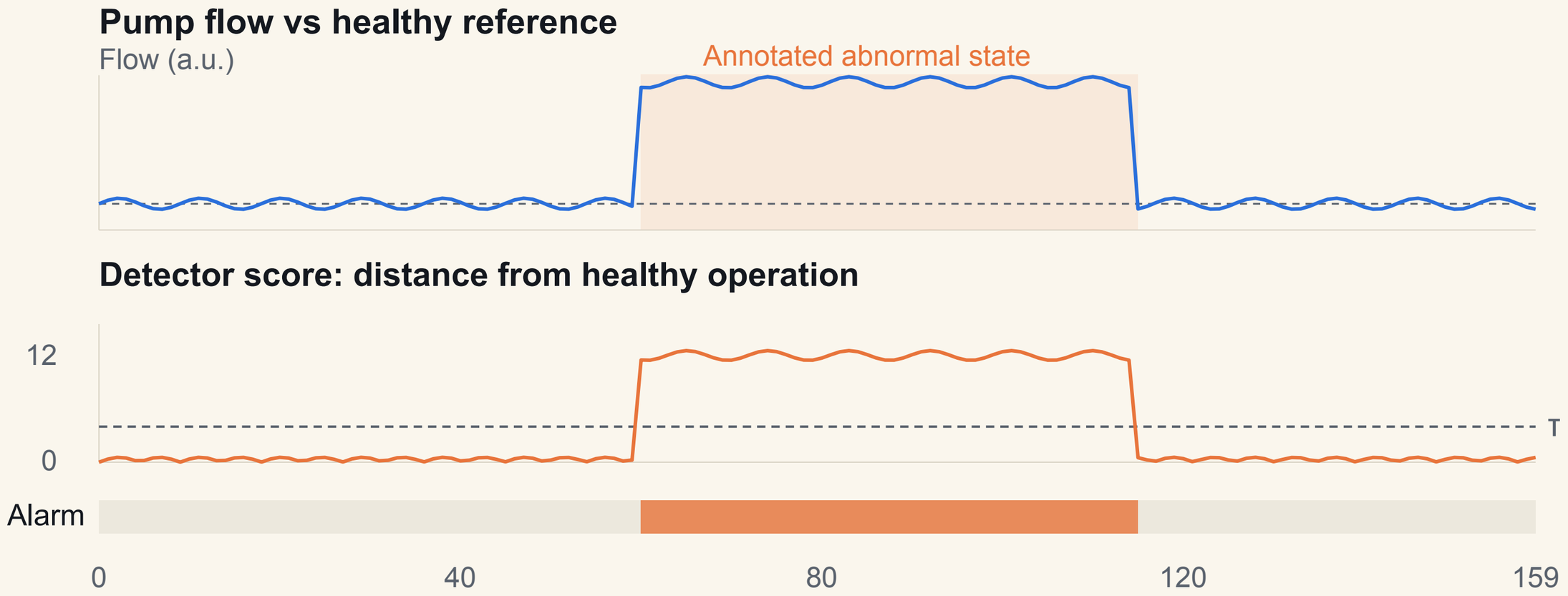



Seasonal naive: repeat the last matching daily positions.

Solid: observations · dashed: forecast · illustrative interval; actual future hidden.

# Anomaly detection: unusual relative to what?

**A score becomes a decision only after thresholding.** Define whether the label covers an onset or the entire abnormal state.

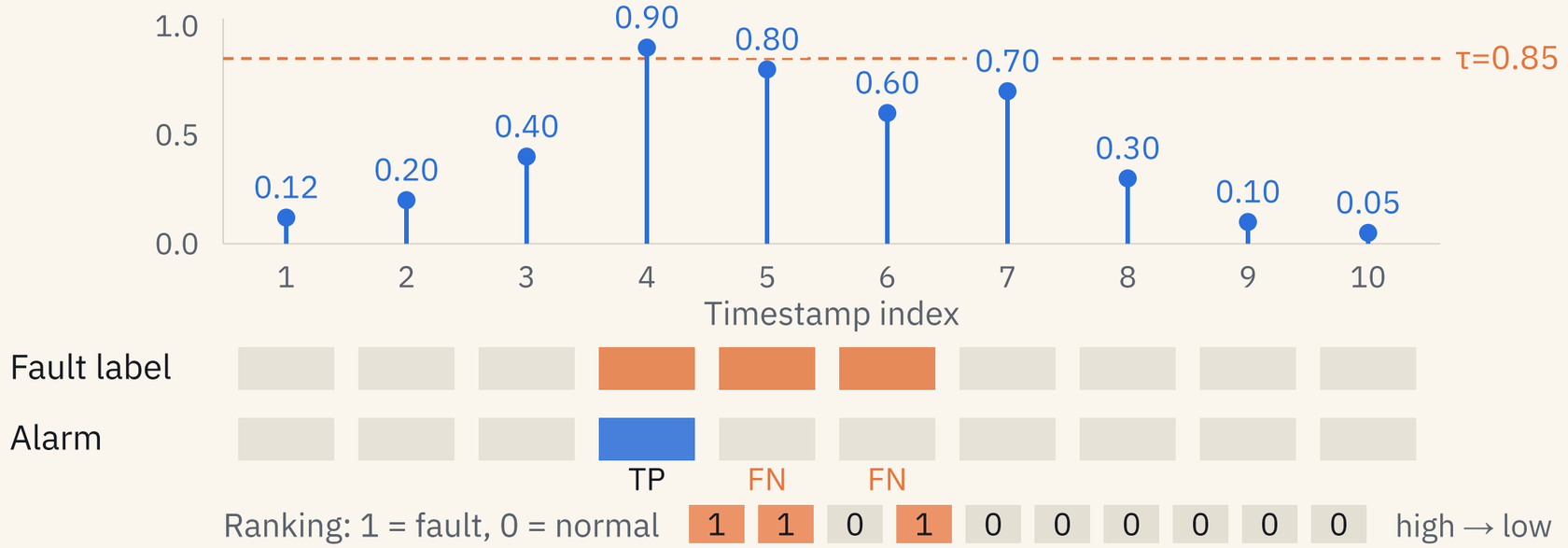


WORKED EXAMPLE

# Scores rank; thresholds decide

**One faulty point alarms; two are missed. Precision is high, but recall is low.**

CONSTRUCTED SCORES · arbitrary units, not fault probabilities




## Decision metrics

TP 1 · FP 0 · FN 2

Precision 1.000
TP / (TP + FP)

Recall 0.333
TP / (TP + FN)

F1 0.500
2TP / (2TP + FP + FN)

AP 0.917
(1 + 1 + 3/4) / 3

Normal alarm: 0.0%
0 / 7 normal timestamps

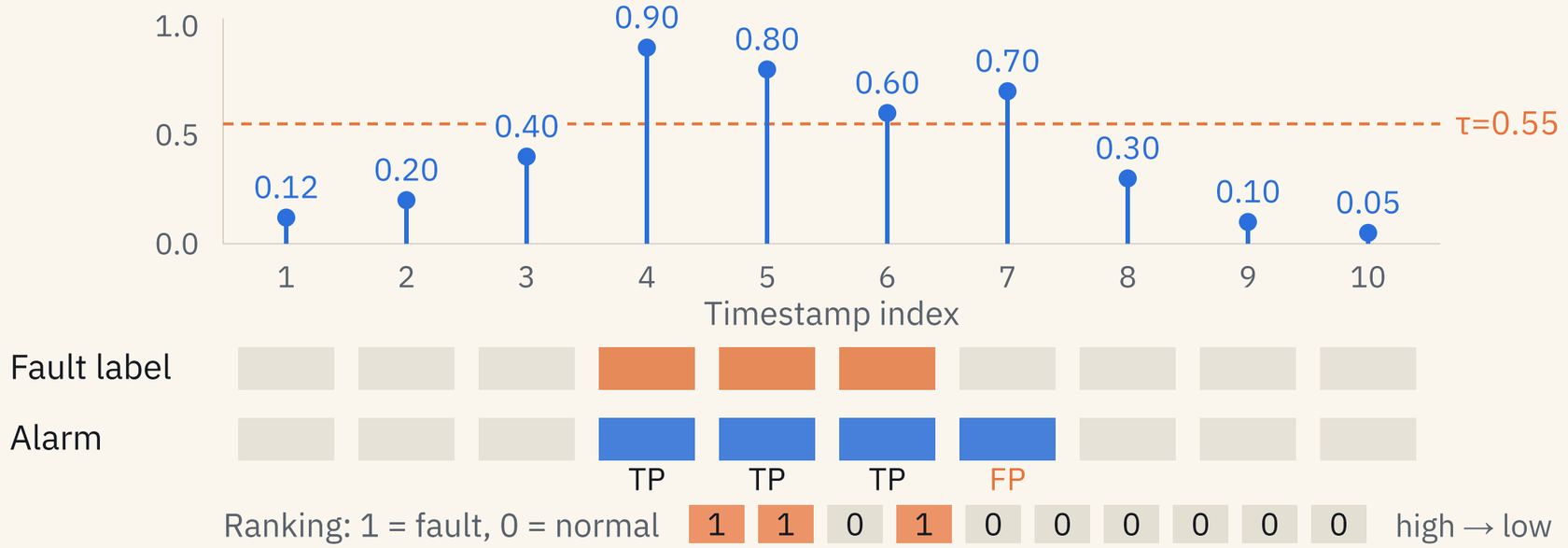
Keyframe 1/3 · Count one operating point — TP 1 · FP 0 · FN 2 · F1 0.500 · AP 0.917 · normal alarm 0.0%

WORKED EXAMPLE

# Scores rank; thresholds decide

**At 0.55, all faulty points alarm, together with one normal point. Count both the gain and the cost.**

CONSTRUCTED SCORES · arbitrary units, not fault probabilities




## Decision metrics

TP 3 · FP 1 · FN 0

Precision 0.750
TP / (TP + FP)

Recall 1.000
TP / (TP + FN)

F1 0.857
2TP / (2TP + FP + FN)

AP 0.917
(1 + 1 + 3/4) / 3

Normal alarm: 14.3%
1 / 7 normal timestamps

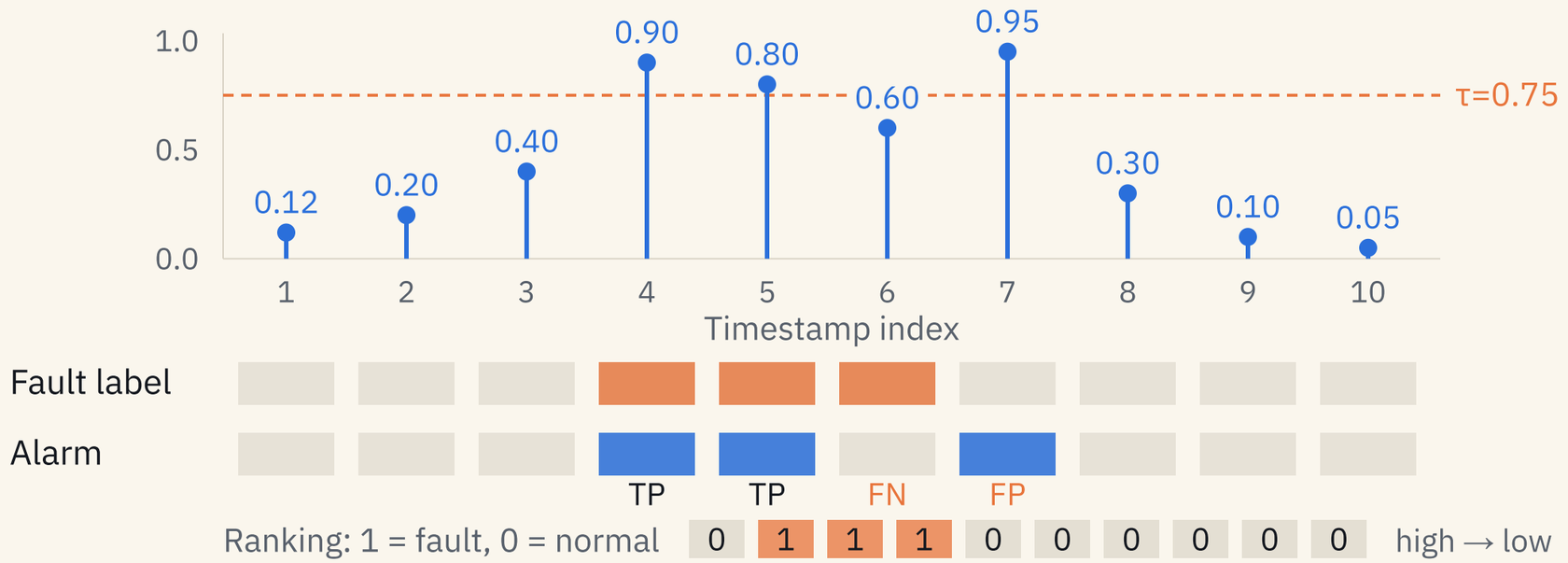
Keyframe 2/3 · Recall meets false alarms — TP 3 · FP 1 · FN 0 · F1 0.857 · AP 0.917 · normal alarm 14.3%

WORKED EXAMPLE

# Scores rank; thresholds decide

**Now a normal point has the highest score. The ranking changes, so AP changes too.**

CONSTRUCTED SCORES · arbitrary units, not fault probabilities




## Decision metrics

TP 2 · FP 1 · FN 1

Precision 0.667
TP / (TP + FP)

Recall 0.667
TP / (TP + FN)

F1 0.667
2TP / (2TP + FP + FN)

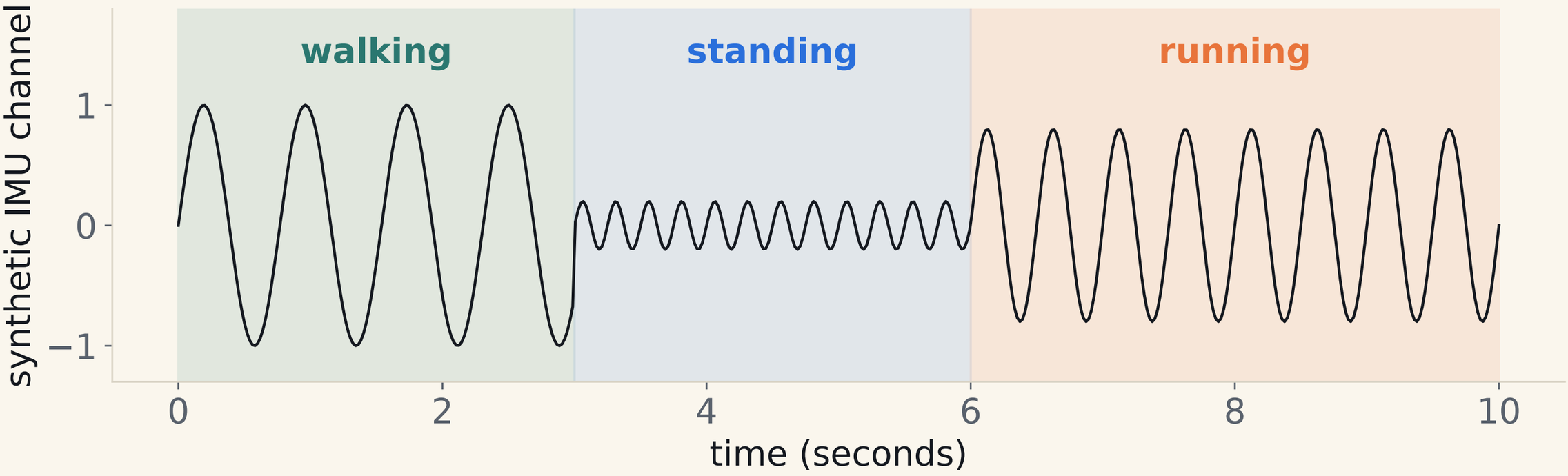
AP 0.639
(1/2 + 2/3 + 3/4) / 3

Normal alarm: 14.3%
1 / 7 normal timestamps

Keyframe 3/3 · Change the ranking — TP 2 · FP 1 · FN 1 · F1 0.667 · AP 0.639 · normal alarm 14.3%

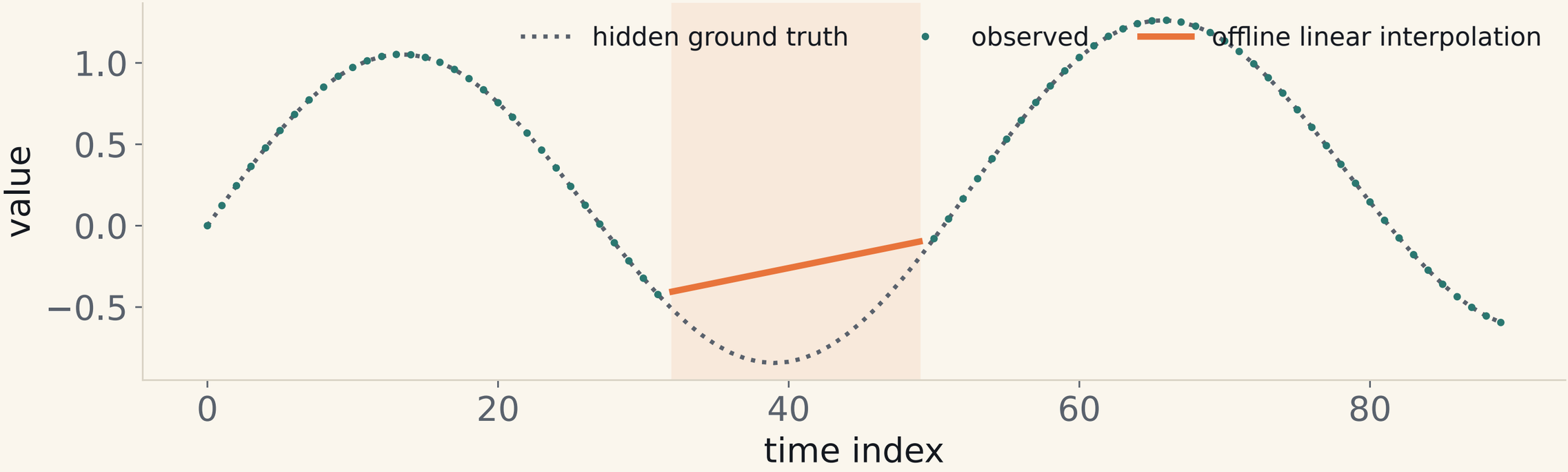
# Segmentation: label the timeline, or find its boundaries

**State labeling:** HMM or temporal CNN. **Change-point detection:** PELT / ruptures.



# Imputation: what information may cross the gap?

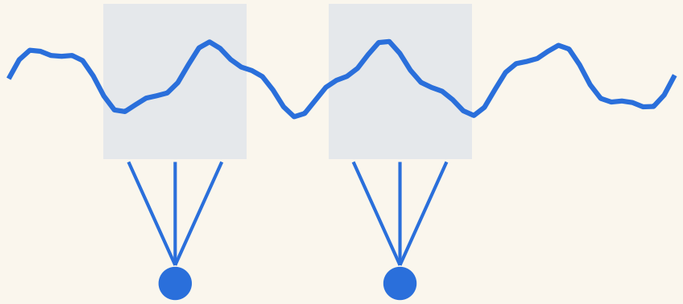
Linear interpolation → Kalman methods → SAITS. Offline filling may use right context; online filling may not.



# The architecture expresses an assumption

Read the models: TCN · EEGNet · DeepAR · PatchTST · Chronos-2.

## CNN / TCN

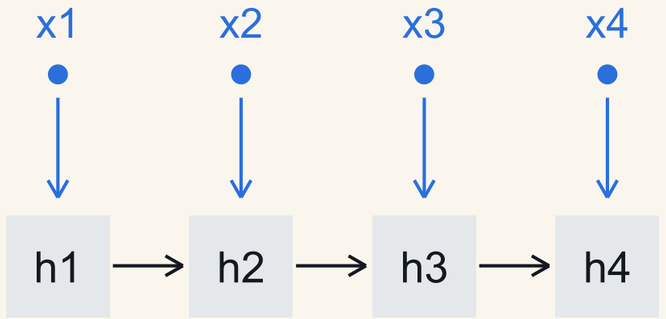


Shared local filter

TCN · causal dilated filters
EEGNet · temporal + spatial

Assumption: useful patterns repeat across time.

## RNN / LSTM

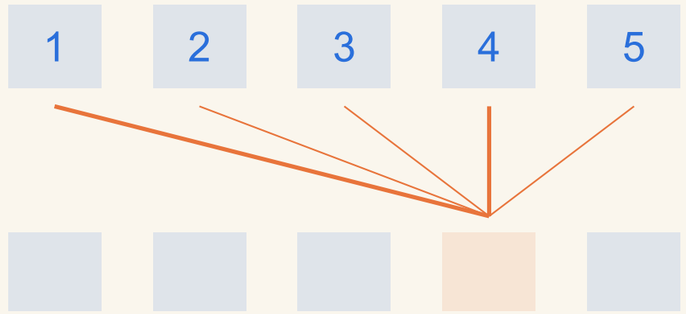



Update a compact state

DeepAR · autoregressive LSTM
Probabilistic forecast outputs

Assumption: a learned state can retain useful history.

## Transformer




Relate visible patches

PatchTST · patches per channel
Chronos-2 · time/group attention

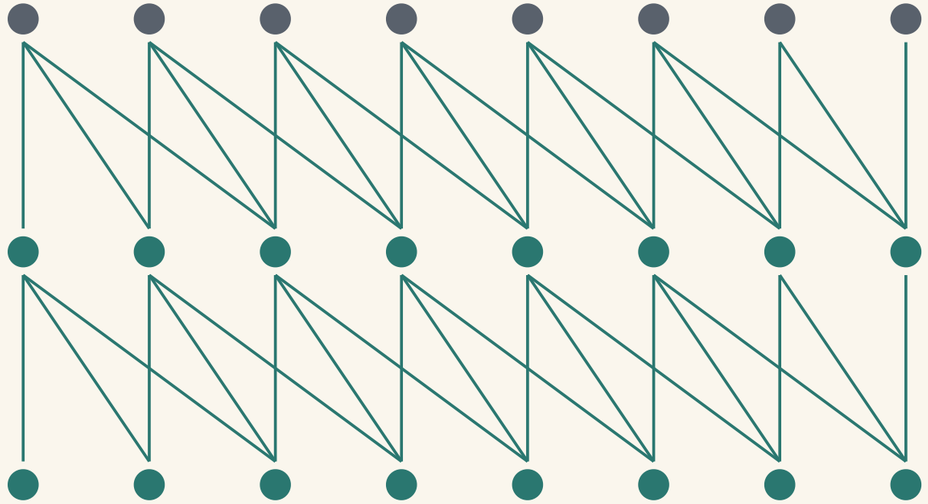
Assumption: selected context positions are worth relating.

Representative models and architectural choices; this is not a performance ranking.

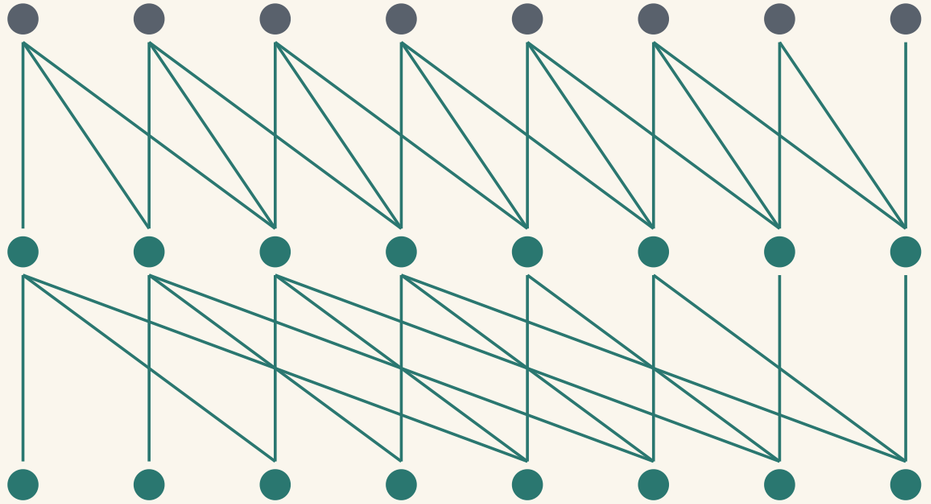
# A causal TCN grows its field of view

Dilation skips input positions while retaining a local kernel. Future values must stay outside the receptive field.

## CAUSAL CONVOLUTION




## DILATION EXPANDS THE RECEPTIVE FIELD

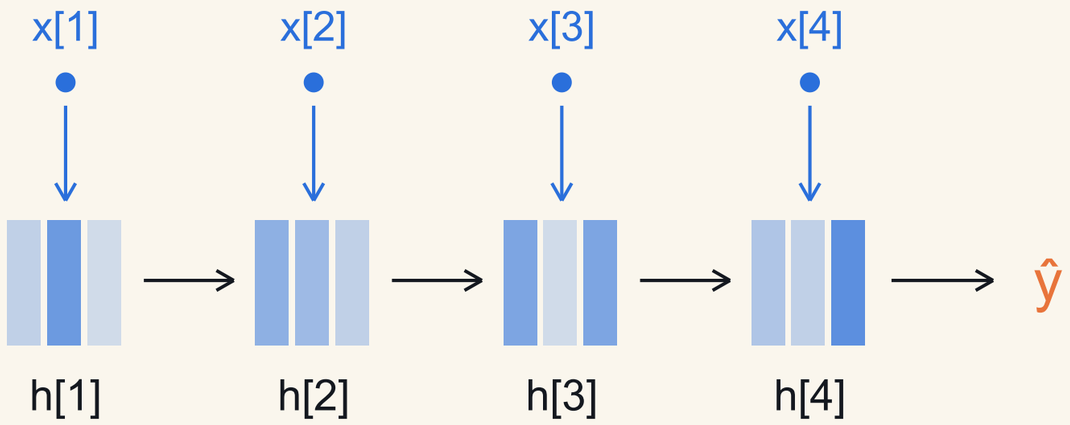



Every edge respects the information available at prediction time.

# Memory and attention answer different questions

Both can learn long dependencies; they store and access the available history differently.

## Recurrent memory

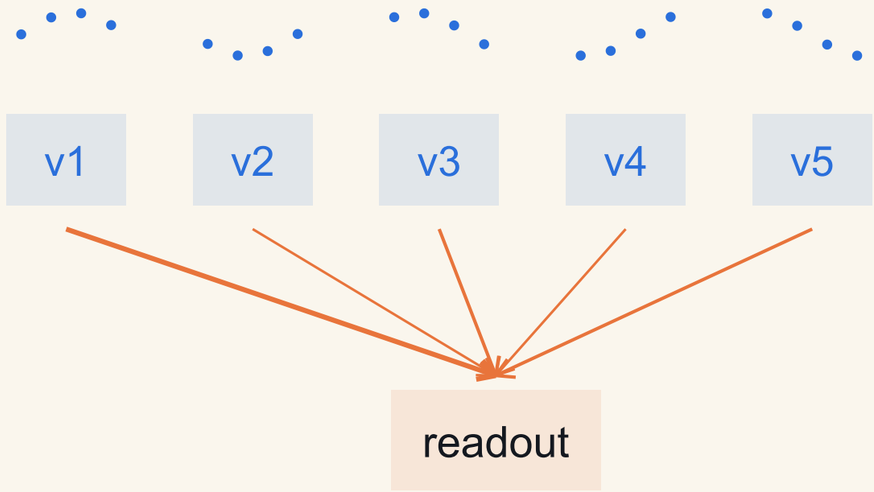



Each observation updates the same state.

Earlier information must survive in h[t].

## Attention over visible patches

A query reads the retained context




$y = \Sigma\, a_j v_j$ · positions encode order

Access depends on the visibility mask.

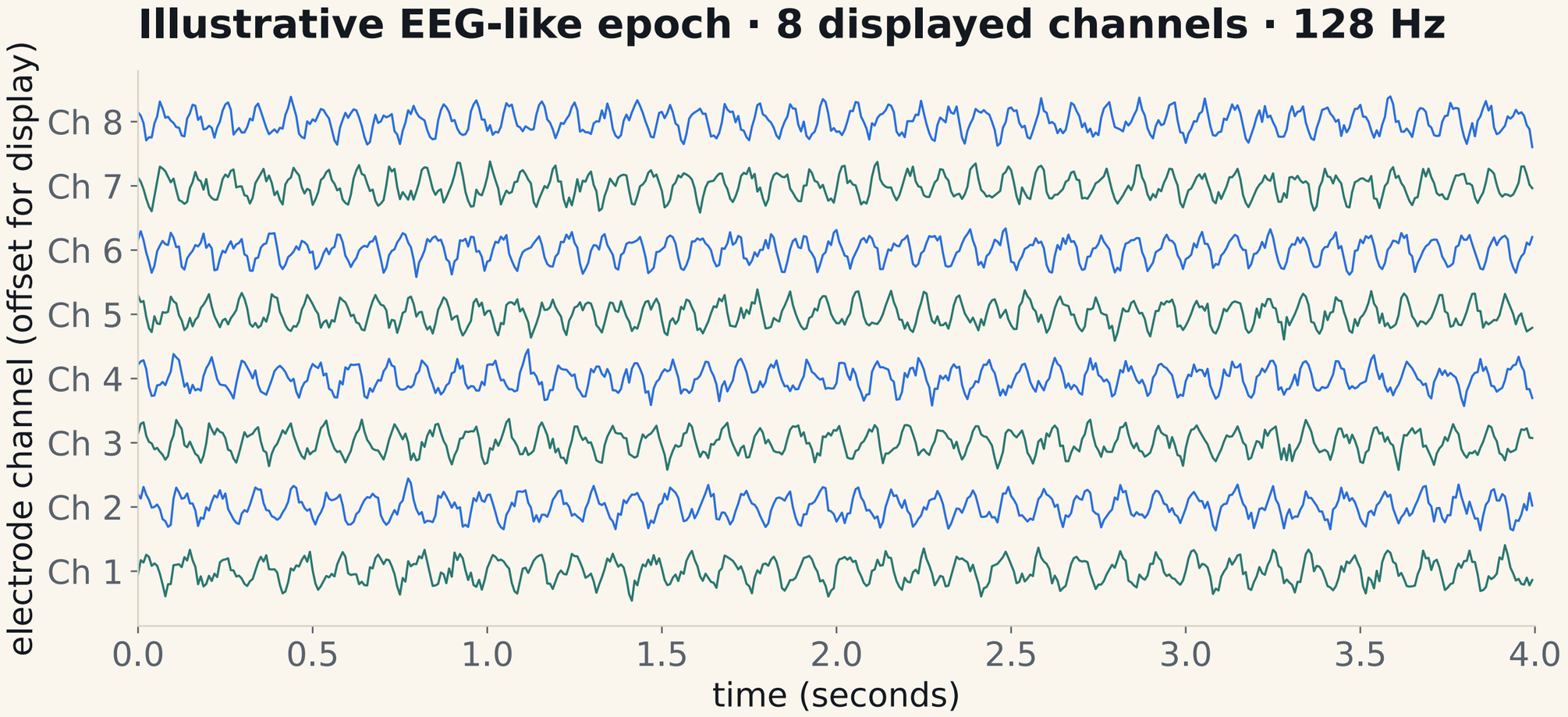
Example: DeepAR / LSTM

Schematic routing, not learned attention weights or a causal explanation.

# EEG: time is only one of the axes

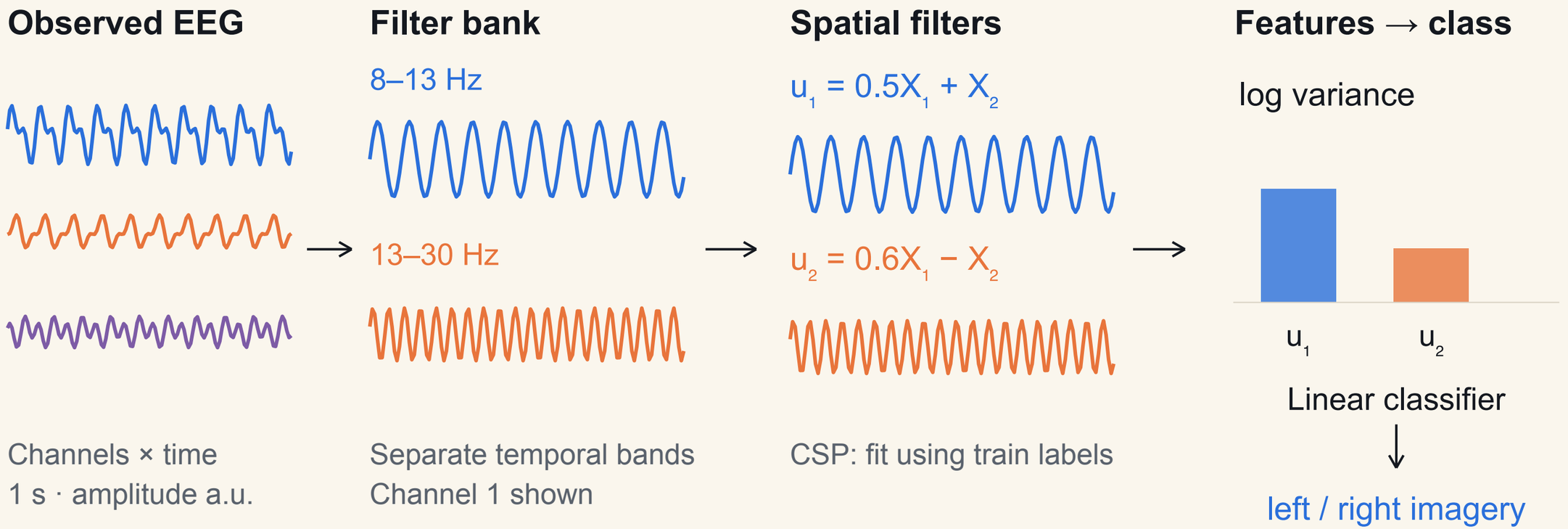
Read a neural time-series model by following its axes and operations.

Same axis-reading skill, different target: **one movement label for an observed EEG epoch**.



# A classical EEG pipeline already has useful structure

EEGNet makes a related sequence of temporal and spatial operations learnable end to end.

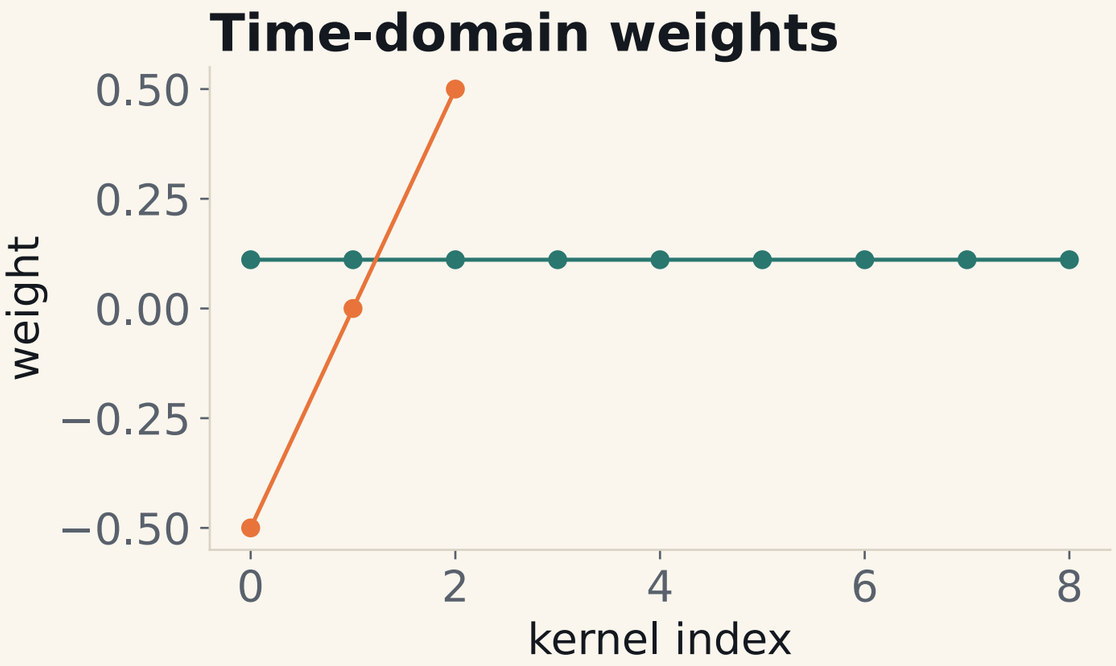


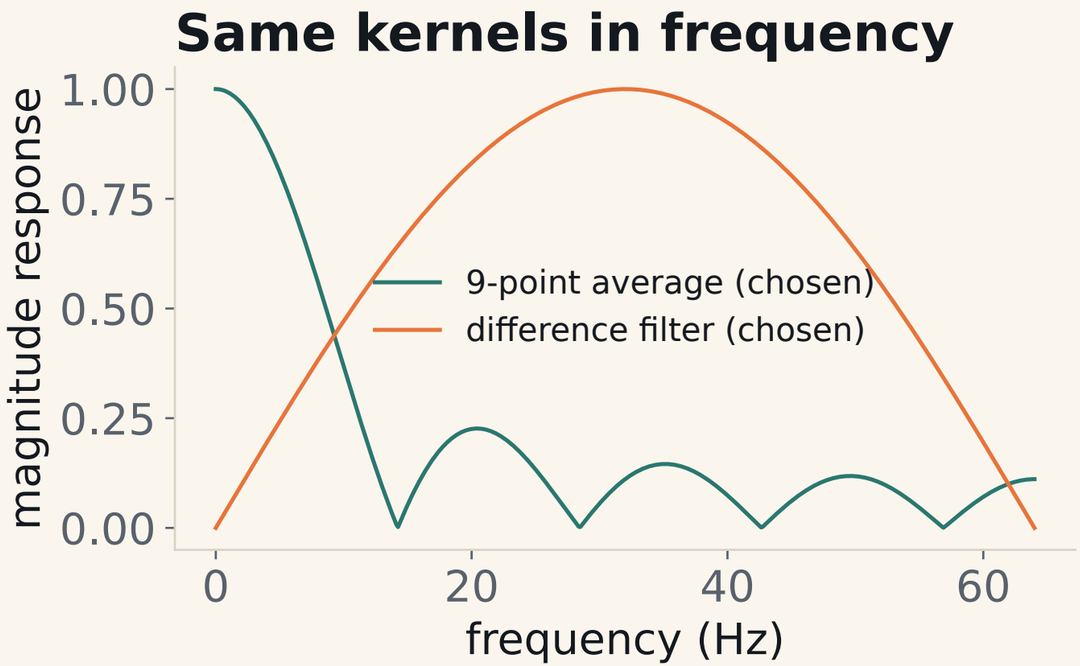

CSP favors high variance in one class and low variance in the other; it is not PCA.

Constructed sources and hand-set mixing; fit CSP and the classifier inside each training fold.

# A temporal kernel has a frequency response

These kernels are chosen by hand. EEGNet learns temporal weights; we can inspect them with the same Fourier tools.



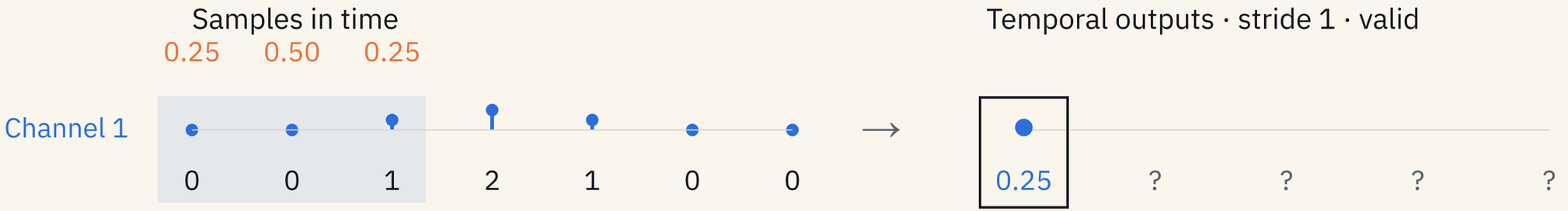



# One convolution, computed by hand

**Only these three samples contribute to this output. The chosen weights are fixed for this example.**

Pump channels → EEG electrodes: same filtering arithmetic

ILLUSTRATIVE values and hand-chosen weights; no trained EEG result




$$0.25 \times 0.0 + 0.50 \times 0.0 + 0.25 \times 1.0 = 0.25$$

Shared temporal kernel: [0.25, 0.50, 0.25]

Next: repeat along each electrode, then learn spatial combinations.

Output position 0 · input samples 0–2

Keyframe 1/3 · Multiply and sum — Channel 1 temporal output 0: 0.25 · input samples 0–2 · shared kernel [0.25, 0.50, 0.25]

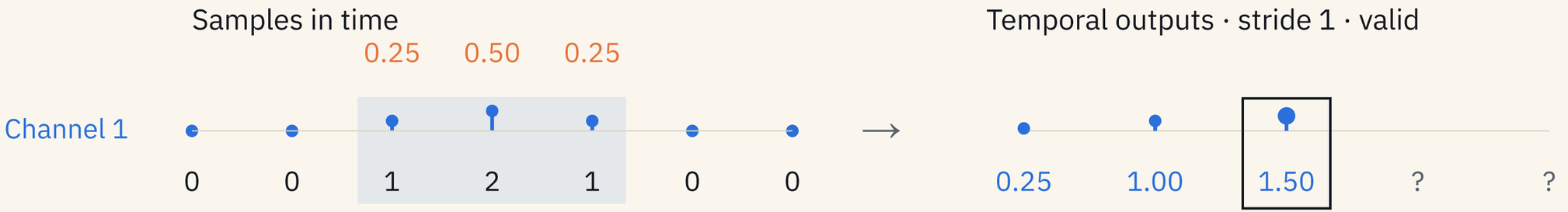
WORKED EXAMPLE

# One convolution, computed by hand

## 0.25 × 1 + 0.50 × 2 + 0.25 × 1 = 1.50. Which axis did we mix?

Pump channels → EEG electrodes: same filtering arithmetic

ILLUSTRATIVE values and hand-chosen weights; no trained EEG result




0.25 × 1.0 + 0.50 × 2.0 + 0.25 × 1.0 = 1.50

Shared temporal kernel: [0.25, 0.50, 0.25]

Next: repeat along each electrode, then learn spatial combinations.

Output position 2 · input samples 2–4

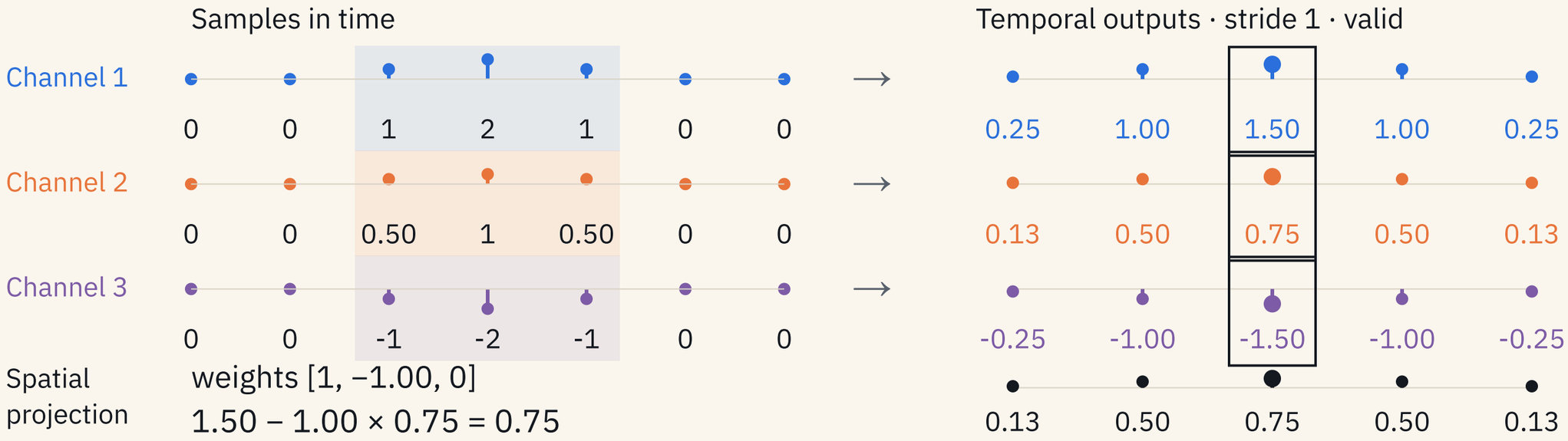
Keyframe 2/3 · At the peak — Channel 1 temporal output 2: 1.50 · input samples 2–4 · shared kernel [0.25, 0.50, 0.25]

# One convolution, computed by hand

**Now mix electrodes at the same time position. Temporal filtering and spatial mixing are different operations.**

Pump channels → EEG electrodes: same filtering arithmetic

ILLUSTRATIVE values and hand-chosen weights; no trained EEG result

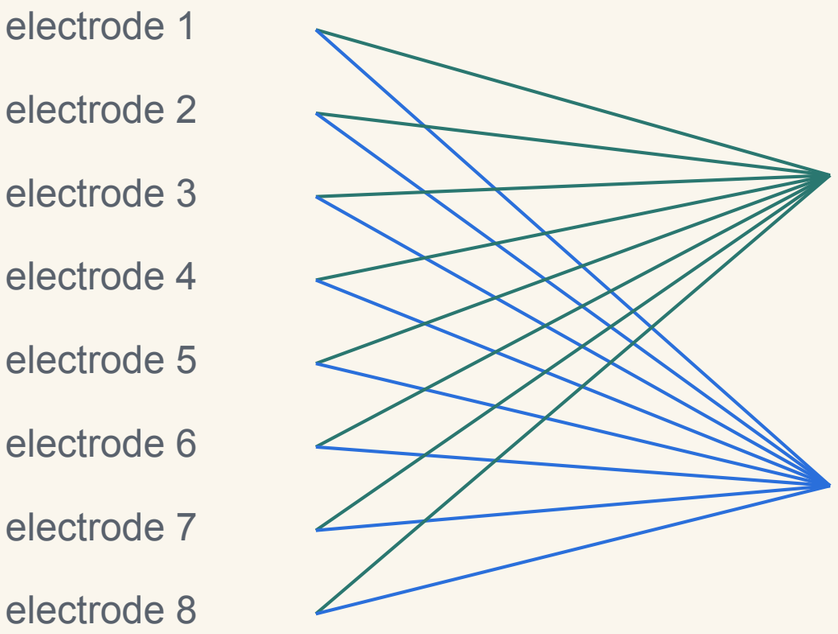



Keyframe 3/3 · Spatial projection — Temporal output 2: 1.50 · spatial weights [1, −1.00, 0] · spatial output 0.75

# Temporal first. Spatial second.

Each temporal feature gets its own spatial projections. **Depthwise** means we do not mix different temporal filters here.

## ONE TEMPORAL FILTER → TWO SPATIAL PROJECTIONS




**spatial filter A**
weighted channel sum

**spatial filter B**
different channel sum

8 temporal filters

× 2 spatial filters each

× 64 electrodes

**= 1,024 weights**

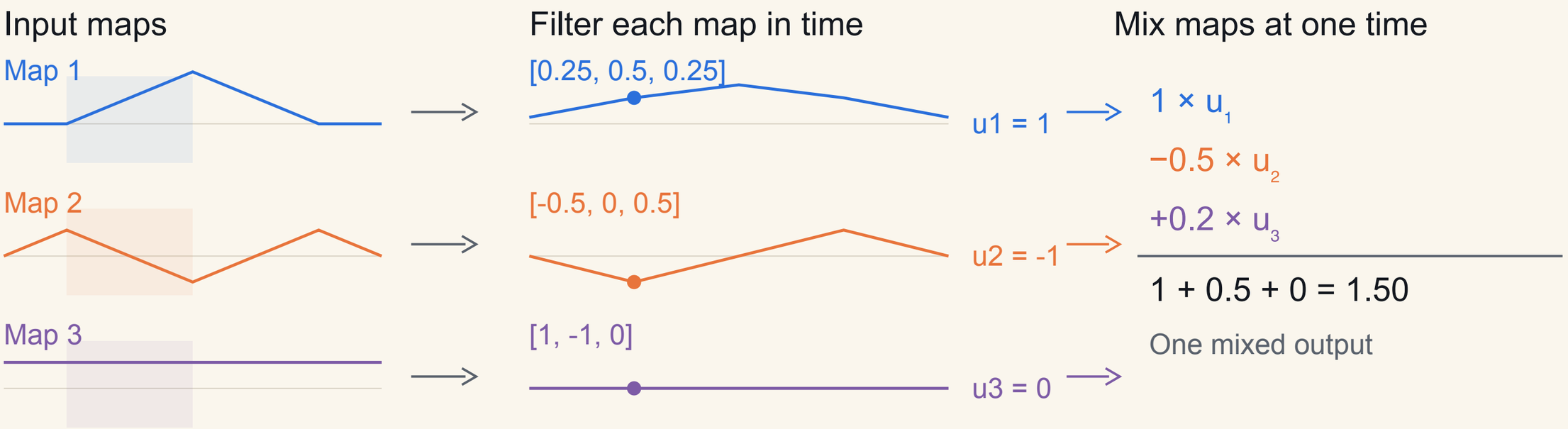
vs 8,192 for full convolution

A projection mixes measured channels; it does not identify a brain source by itself.

# Separable convolution: refine, then mix

**Focus:** Distinguish per-map temporal refinement from pointwise feature-map mixing.

A small numerical example: 3 of 16 feature maps




Real block: 16 × 16 temporal weights + 16 × 16 mixing weights = 512

Full 16-tap convolution would use 16 × 16 × 16 = 4,096 weights. Example weights above are illustrative.

# Follow one EEG epoch through the network

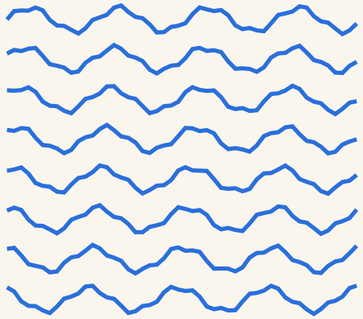
**Focus:** Follow which EEGNet operations change maps, electrodes and time.

EEGNet-8,2 · 64 electrodes · 512 samples at 128 Hz = 4 s · 2 classes

Axes: maps × electrodes × time. Feature values below are schematic; batch axis omitted.

### 1 Observe

1 × 64 × 512

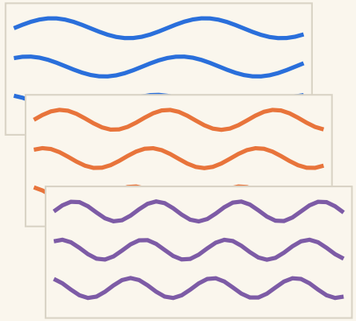


64 electrode rows

512 time samples

### 2 Temporal filters

8 × 64 × 512



8 learned filters

Shared over 64 rows

Kernel: 1 × 64; same

### 3 Spatial + pool

16 × 1 × 128



8 × 2 spatial maps

64 electrodes → 1

Pool 4: 512 → 128

### 4 Separable + pool

16 × 1 × 16



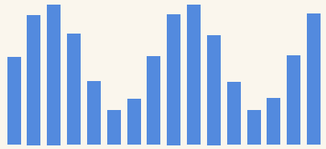
16 maps × 16 bins

Pool 8: 128 → 16

Electrode axis stays 1

### 5 Flatten + classify

256 → 2



256 features

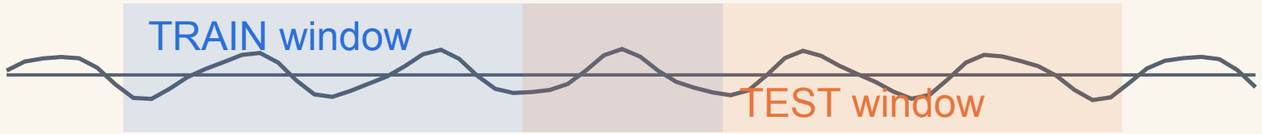
zL zR

Softmax → left / right

# An EEG score is only as honest as its split

Choose the unit that will be new at deployment

Leakage risk — Two windows of the same trial share samples




Same trial

New trial — Person A — Trial 1 | Trial 2 | Trial 3 — Known person

New person — Person A | Person B | Person C — Unseen person

Within each fold: fit scaler / CSP / model on TRAIN → apply frozen to TEST

Keep every overlapping window from a trial together. Session holdout answers a third deployment question.

# Transfer the filtering idea back to pump sensors

## A filtered tensor has 2 maps × 4 sensors × T. Which operation makes one sensor mixture per map while keeping maps separate?

1. Apply another temporal kernel along T, independently on each sensor.
2. Use a depthwise spatial kernel spanning all four sensors, separately per map.
3. Use unrestricted convolution mixing maps and sensors together.

Which axis collapses, and which axis must stay separate?

Pause here. The next keyframe contains the explanation.

# Transfer the filtering idea back to pump sensors

## A filtered tensor has 2 maps × 4 sensors × T. Which operation makes one sensor mixture per map while keeping maps separate?

1. Apply another temporal kernel along T, independently on each sensor.
2. **Use a depthwise spatial kernel spanning all four sensors, separately per map. ✓**
3. Use unrestricted convolution mixing maps and sensors together.

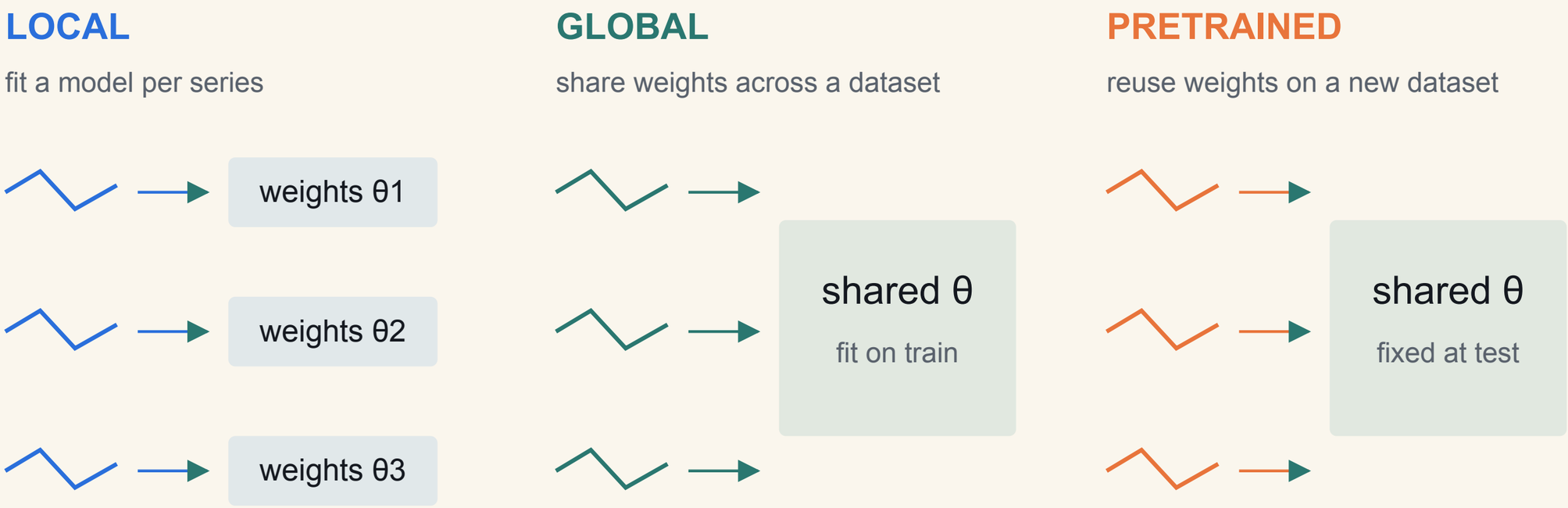
Which axis collapses, and which axis must stay separate?

> **Reasoning**
>
> Collapse the sensor axis 4 → 1 and preserve 2 maps: output 2 × 1 × T for depth multiplier 1. This transfers EEGNet's spatial constraint to a pump example; it does not transfer trained EEG weights or establish which architecture is best.

# What changes when we pretrain once?

Explain pretrained forecasting and evaluate a complete detector under a fixed alarm budget.

Local models fit each series. Global models share weights. A pretrained forecaster reuses weights on a new task.




In-context learning changes the input context. Fine-tuning changes the weights.

# Chronos-2: a modern forecasting model we can read

`autogluon/chronos-2-small` · 6 blocks · 8 heads · hidden size 512 · 13 quantiles

Lab input: 8 sensors × 128 observations → forecast the next 16 observations

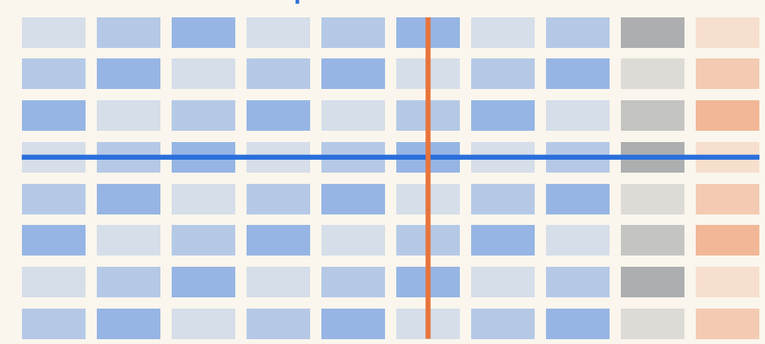
## 1 Scale each sensor



Standardize + asinh

8 non-overlapping
patches of 16 / sensor

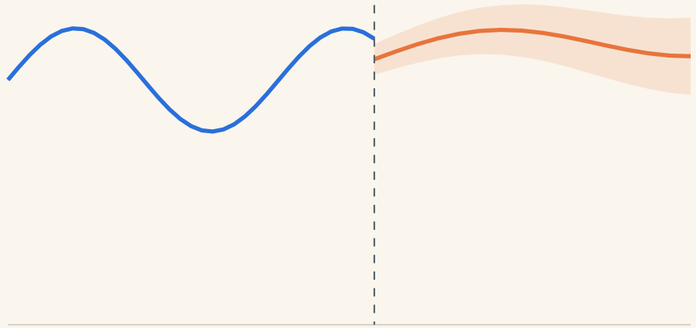
## 2 Embed patches; alternate attention




Group attention: across 8 sensors

6 blocks · each token has 512 values

## 3 Quantiles → original units




13 quantiles × 16 steps
for each of 8 targets

Forecast traces are schematic.

# A forecasting task is data plus a visibility contract

Values supplied to the encoder at forecast issuance

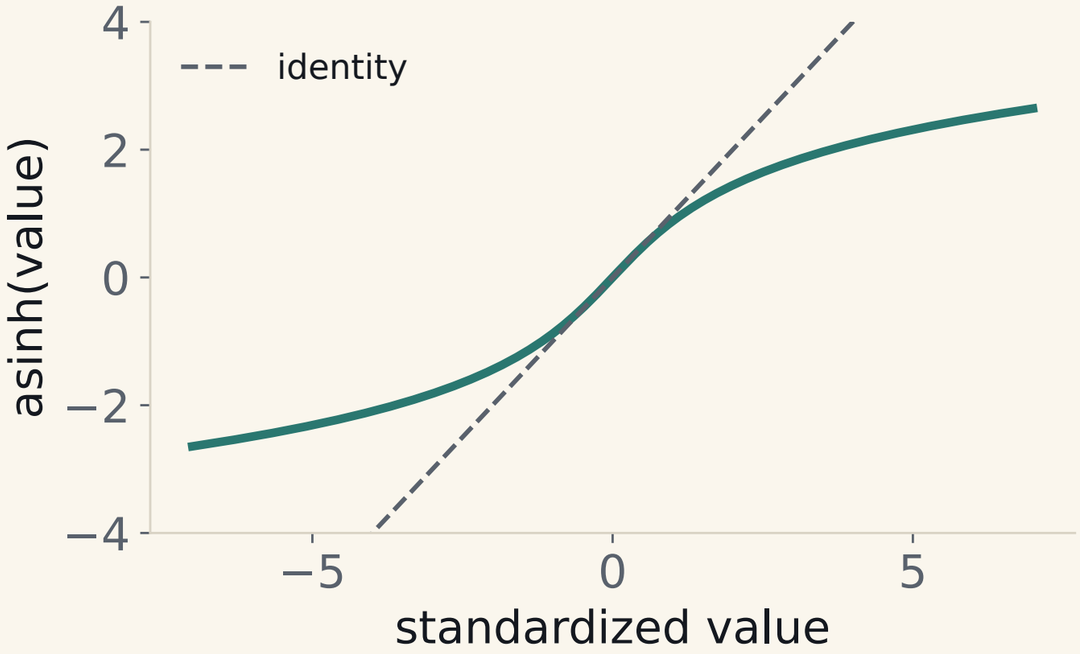
| | t−3 | t−2 | t−1 | t | t+1 | t+2 | t+3 |
|---|---|---|---|---|---|---|---|
| Pressure · target | 98<br>mask 1 | 100<br>mask 1 | ?<br>mask 0 | 101<br>mask 1 | ?<br>mask 0 | ?<br>mask 0 | ?<br>mask 0 |
| Flow · target | 1.0<br>mask 1 | 0.9<br>mask 1 | 1.1<br>mask 1 | 1.0<br>mask 1 | ?<br>mask 0 | ?<br>mask 0 | ?<br>mask 0 |
| Vibration · past only | 0.2<br>mask 1 | 0.3<br>mask 1 | 0.2<br>mask 1 | 0.3<br>mask 1 | ?<br>mask 0 | ?<br>mask 0 | ?<br>mask 0 |
| Valve plan · known future | 0<br>mask 1 | 0<br>mask 1 | 0<br>mask 1 | 1<br>mask 1 | 1<br>mask 1 | 0<br>mask 1 | 0<br>mask 1 |

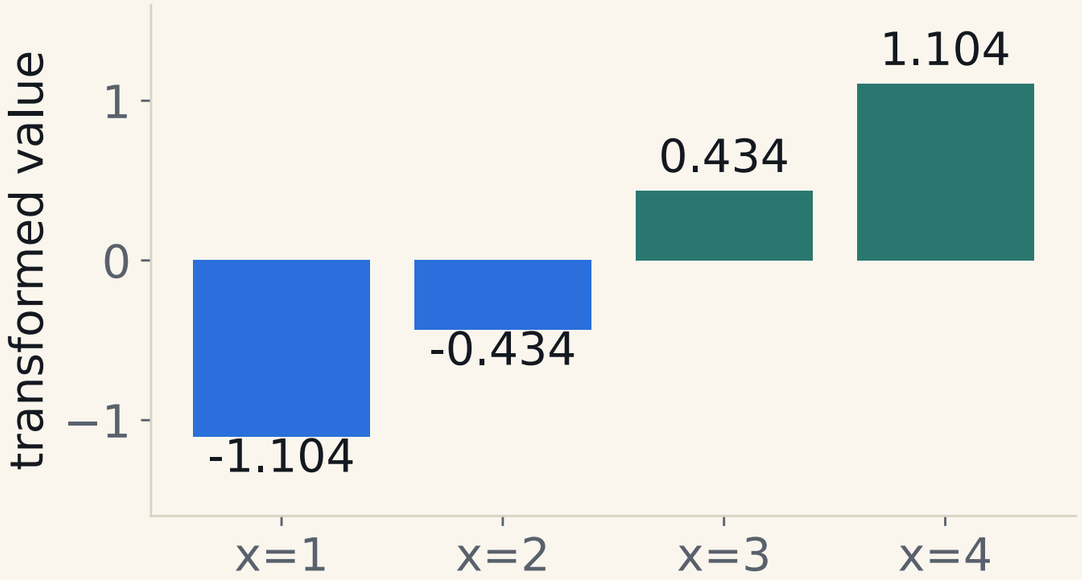
Future target values stay hidden. A measured zero has mask 1; an unavailable value has mask 0.

Illustrative covariates; the SKAB lab supplies 8 targets and no future measured sensor values.

# Scaling keeps units from dominating the network

$$z = \operatorname{asinh}((x - \mu) / \sigma) \quad \rightarrow \quad \hat{x} = \mu + \sigma \sinh(\hat{z})$$



# A missing value and a measured zero are different

The schedule is hypothetical. These are input-construction numbers, **not a measured forecast**.

**ILLUSTRATIVE TOY · P = 2 · T = 4 · H = 4 · d = 8**

| | past 1 | past 2 | past 3 | past 4 | future 1 | future 2 | future 3 | future 4 |
|---|---|---|---|---|---|---|---|---|
| pressure target | 1 | 2 | 3 | 4 | ? | ? | ? | ? |
| flow target | 4 | 3 | 2 | 1 | ? | ? | ? | ? |
| known schedule | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

? → value 0 + observed mask 0

observed zero → value 0 + mask 1

**Two context patches + [REG] + two future patches → conceptual shape 3 × 5 × 8**

# A patch becomes a vector, without value quantization

**Focus:** Explain how values, time and an observation mask become one patch representation.

One 16-point patch · example historical mean = 100, standard deviation = 2

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Raw pressure | 98 | 100 | 102 | 104 | ? | 101 | 99 | 100 | 102 | 98 | 100 | 101 | 103 | 100 | 99 | 102 |
| Time feature | -128 /8192 | -127 /8192 | -126 /8192 | -125 /8192 | -124 /8192 | -123 /8192 | -122 /8192 | -121 /8192 | -120 /8192 | -119 /8192 | -118 /8192 | -117 /8192 | -116 /8192 | -115 /8192 | -114 /8192 | -113 /8192 |
| asinh((x−100)/2) | -0.88 | 0.00 | 0.88 | 1.44 | 0.00 | 0.48 | -0.48 | 0.00 | 0.88 | -0.88 | 0.00 | 0.48 | 1.19 | 0.00 | -0.48 | 0.88 |
| Observed mask | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

98 → −1 → asinh(−1) = −0.881; the missing entry becomes value 0 with mask 0.

16 time + 16 values + 16 masks = 48 numbers ⟶ Residual MLP ⟶

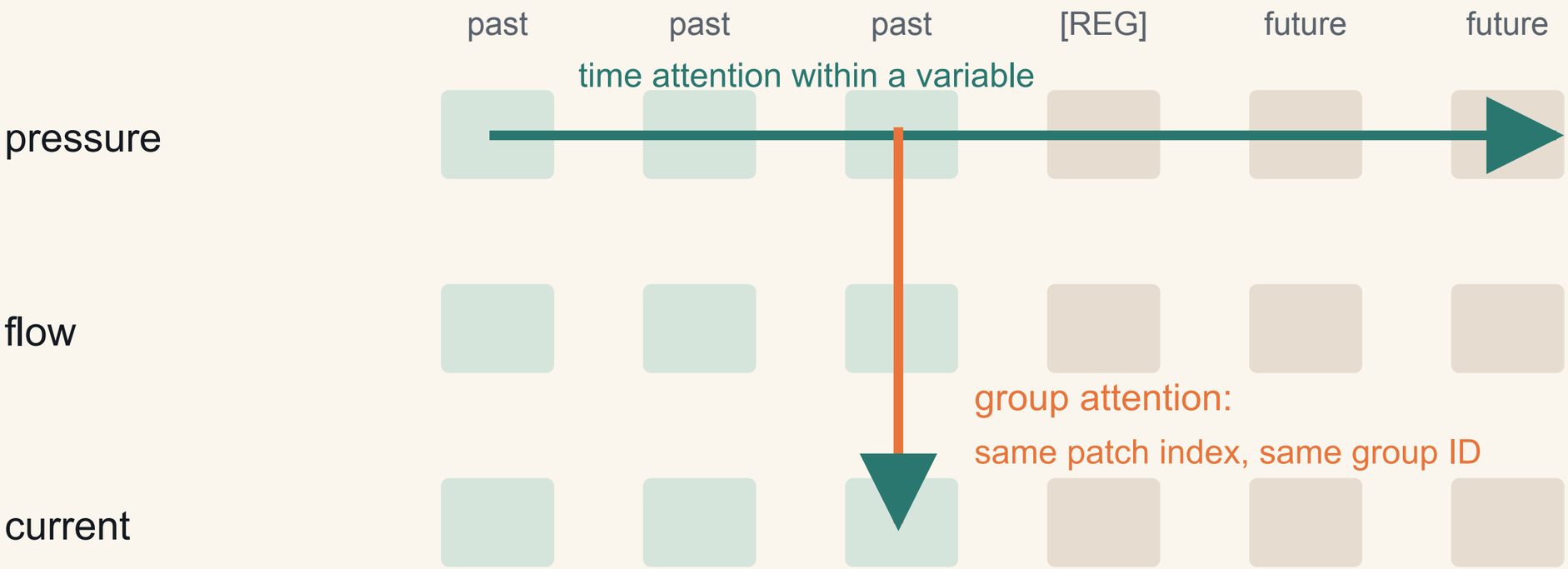


512 continuous features

# Time attention reads within one variable

Chronos-2 uses an **encoder**. Its time attention is bidirectional across the supplied context and future placeholders.

**ALTERNATING ATTENTION ON A VARIABLE × PATCH GRID**




Bidirectional encoder attention is safe here: unknown future targets are never supplied.

# Group attention reads across related variables

At one patch index, series with the same group ID exchange information. Different group IDs block that exchange.

## GROUP IDs CHANGE WHICH SERIES CAN SHARE INFORMATION

### independent series

series 1 group 1

series 2 group 2

series 3 group 3

No natural ordering of variables.

### one multivariate task

series 1 group 1

series 2 group 1

series 3 group 1

No natural ordering of variables.

### two separate systems

series 1 group 1

series 2 group 1

series 3 group 2

No natural ordering of variables.

# The head predicts quantiles for the whole horizon

## Direct multi-step forecast

Each future patch embedding produces values at several quantile levels.

$$H \times \text{targets} \times \text{quantiles}$$

Multiple future patches come from one forward pass.

## Read the uncertainty correctly

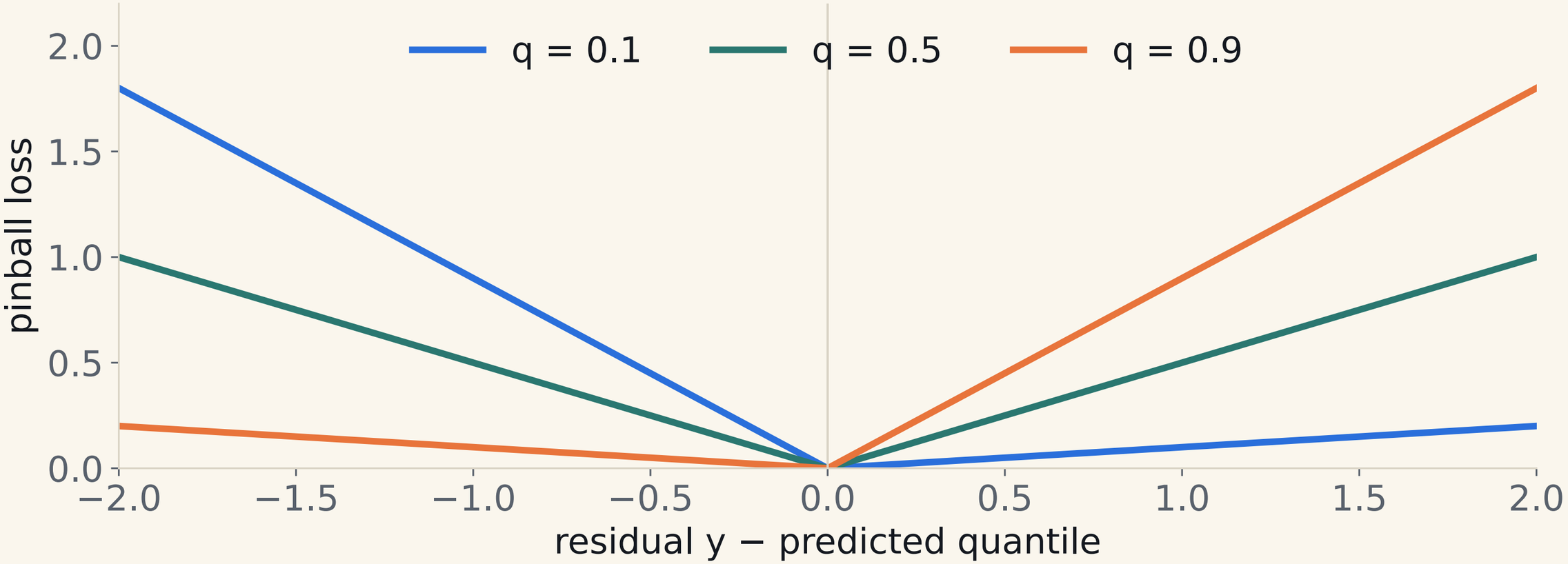
$q = 0.5$ is the predicted median.

$[q = 0.1, q = 0.9]$ is a nominal 80% prediction interval.

> A set of marginal quantiles is not a sampled joint future trajectory.

# Quantile loss makes over- and underprediction cost different

At q = 0.9: underpredict by 1 → loss 0.9; overpredict by 1 → loss 0.1.



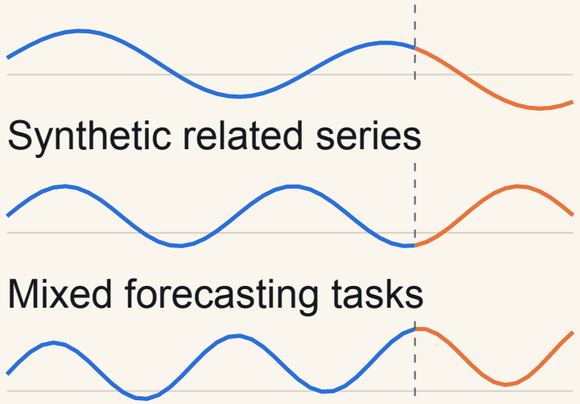
# Pretraining teaches transferable forecasting structure

## PRETRAINING · update forecasting weights

Real-world series

Synthetic related series

Mixed forecasting tasks



Blue: model input

Orange: withheld future labels

**Shared forecaster fθ**

Predict future quantiles

↓

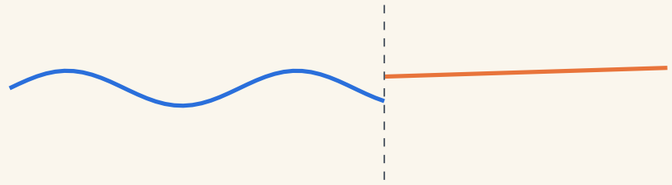
Quantile loss vs true future

$$\theta \leftarrow \theta - \eta \nabla_\theta L_{quantile}$$

Targets are labels for the loss;
they are not future input values.

## LAB · freeze forecasting weights

New pump context



Same θ · new context and masks

Detector wrapper still needs:

normal reference + score mapping

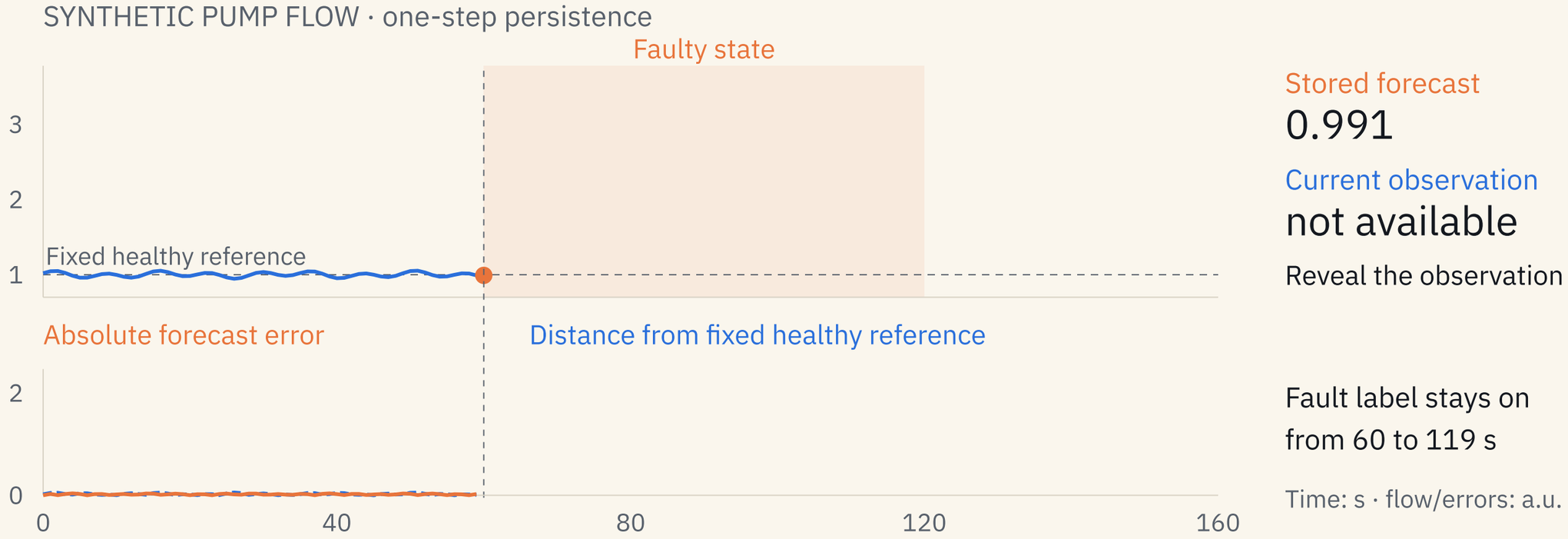
validation threshold selection

Zero-shot forecasting does not mean calibration-free detection. Dataset overlap must be audited, not assumed absent.

WORKED EXAMPLE

# A forecast can become comfortable with a fault

**At t = 60, the forecast is already stored. Predict its error before revealing the new flow measurement.**

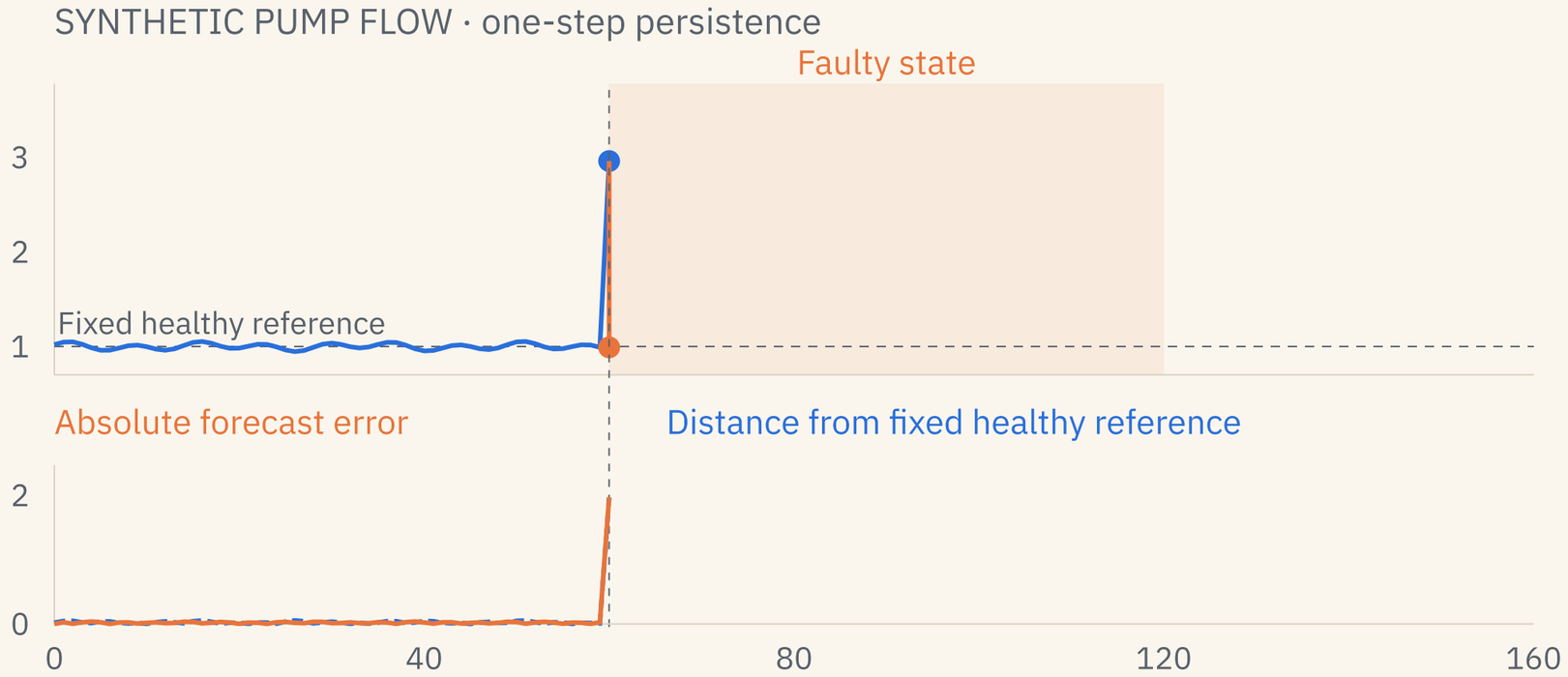



Keyframe 1/3 · Issue the forecast — t = 60 s · forecast 0.991 · current observation not yet available · synthetic one-step persistence

WORKED EXAMPLE

# A forecast can become comfortable with a fault

**The first shifted measurement arrives: the residual jumps. The pump has entered a sustained abnormal state.**




Stored forecast

0.991

Current observation

2.961

Absolute error: 1.970

Fault label stays on from 60 to 119 s

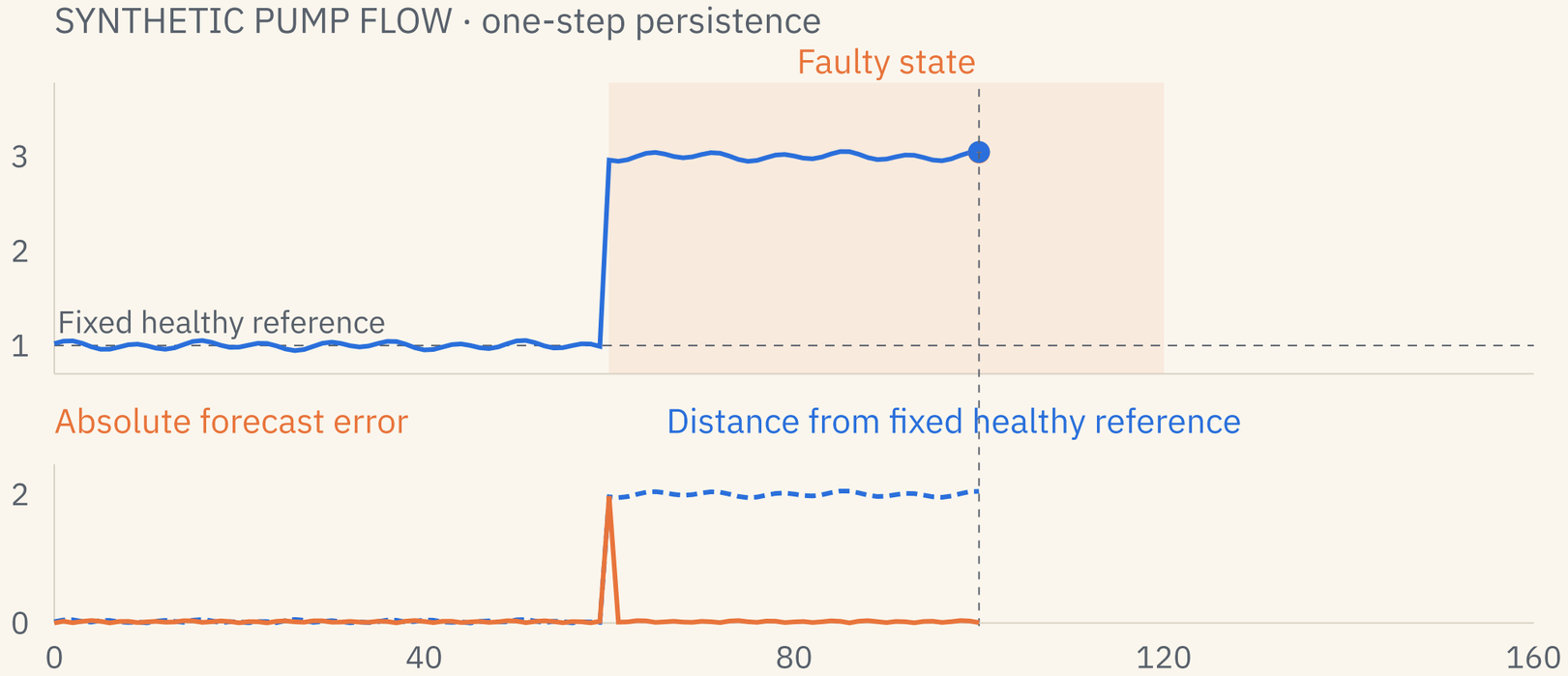
Time: s · flow/errors: a.u.

Keyframe 2/3 · Observe the fault — t = 60 s · forecast 0.991 · observed 2.961 · residual 1.970 · synthetic one-step persistence

WORKED EXAMPLE

# A forecast can become comfortable with a fault

**The forecast error is small inside the fault. Distance from the fixed healthy reference remains large.**




Stored forecast

3.041

Current observation

3.047

Absolute error: 0.005

Fault label stays on from 60 to 119 s

Time: s · flow/errors: a.u.

Keyframe 3/3 · Predictable fault — t = 100 s · forecast 3.041 · observed 3.047 · residual 0.005 · synthetic one-step persistence

# A detector is a chain of choices

Compare complete detection systems, then inspect which stage explains their differences.

## The comparison unit is the whole detection pipeline

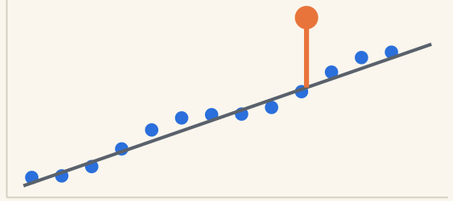
### Representation
raw + chosen history



[level, difference, spread]

Preserve the useful change.

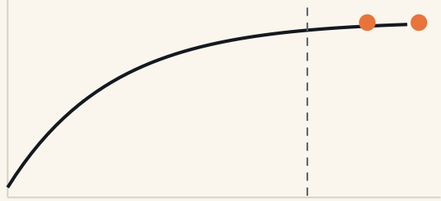
### → Raw detector
distance or residual



PCA Q / IF / forecast error

Useful ranking, arbitrary units.

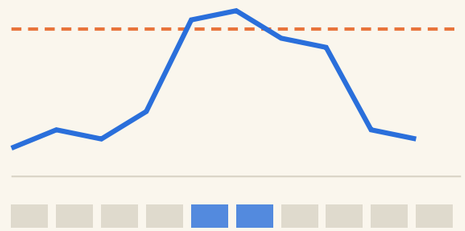
### → Calibration
healthy reference



Map scores; inspect tail ties.

Percentile ≠ fault probability.

### → Alarm policy
cutoff + time budget



Thresholded scores become actual alarms.

Forecast weights may stay fixed while score mapping and the alarm cutoff are fitted.

Keep transformations, calibration and policy in the comparison; report their assumptions.

# The lab question: detect the entire abnormal state

**Which pipeline detects the full abnormal state under a 1% normal alarm-time budget?**

## Fit

Normal prefixes only.

Fit scaler and classical detector parameters.

## Healthy calibration

Separate normal prefixes.

Map raw scores to empirical percentiles.

## Labeled validation

Select a threshold under a normal alarm-time budget.

Freeze the choice.

## Test

Evaluate once.

State detection first; onset delay second.

# Macro and pooled answer different questions

In this lab, AP and F1 give every run one vote; normal alarm time gives every normal timestamp one vote.

## Two runs ask two averaging questions

Constructed counts; the same rules are used in the SKAB result table.

| Run | Normal timestamps · one false alarm in each run | State F1 |
|---|---|---|
| A |  1/10 = 10% normal alarm time | 0.750<br>TP 3, FP 1, FN 1 |
| B |  1/100 = 1% normal alarm time | 0.615<br>TP 4, FP 1, FN 4 |

### Equal weight per run

Macro F1 = (0.750 + 0.615) / 2 = 0.683

Macro AP also averages the AP of each run.

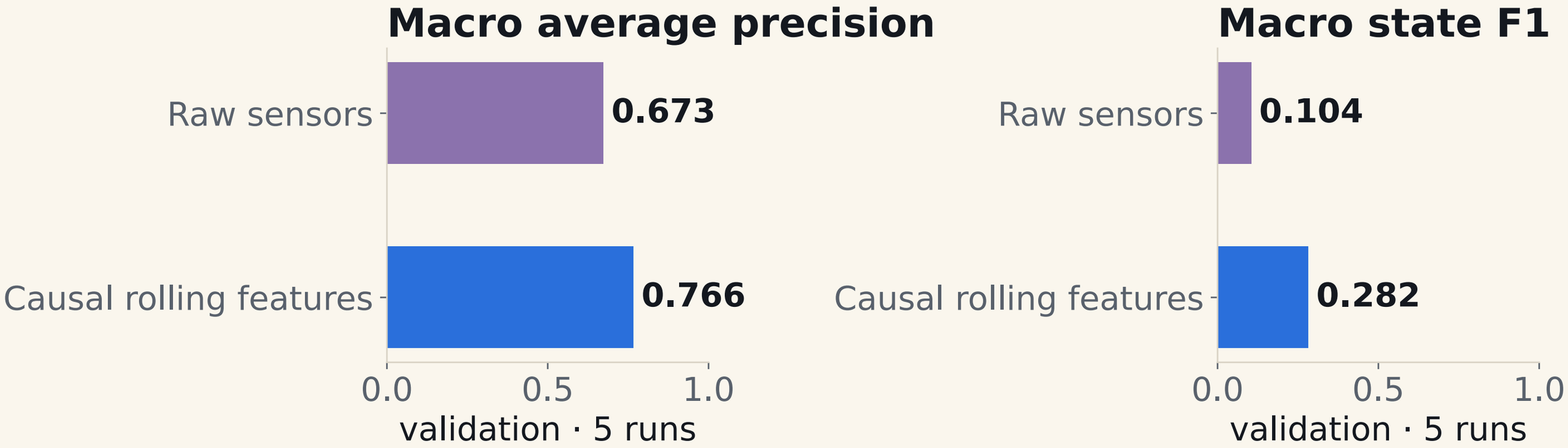
### Weight by normal exposure

Pooled normal alarm = (1 + 1) / (10 + 100)

= 1.82%, rather than the run mean of 5.5%.

# Rolling features help this model on validation

Same Isolation Forest: 200 trees. Raw sensors → 48 causal sensor, difference and rolling features. **Validation only.**



# Compare the complete validation pipelines

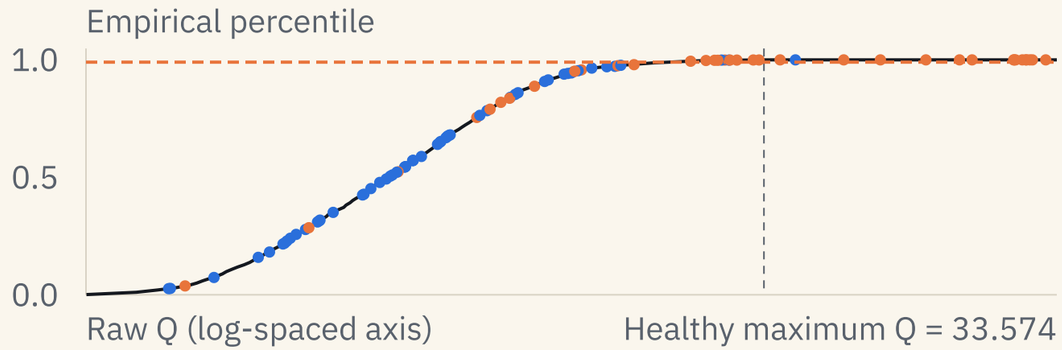
| Complete detection pipeline | Macro AP | Macro F1 | Normal alarm time |
|---|---|---|---|
| Normal distance | 0.768 | 0.000 | 0.000% |
| PCA Q | 0.855 | 0.000 | 0.000% |
| IF: raw sensors | 0.673 | 0.104 | 0.935% |
| IF: rolling features | 0.766 | 0.282 | 0.999% |
| Mean fusion | 0.877 | 0.522 | 0.999% |
| Max fusion | 0.775 | 0.000 | 0.000% |
| Persistence residual | 0.435 | 0.039 | 0.967% |
| Chronos residual | 0.442 | 0.091 | 0.967% |
| Chronos interval | 0.362 | 0.001 | 0.902% |

Validation only · AP/F1: mean over 5 runs · alarm time: pooled normal timestamps.
AP: calibrated ranking · F1: selected cutoff · normal-time budget: 1%.

# Useful ranking can fail at the chosen cutoff

**PCA Q is a distance from the healthy subspace. Healthy calibration converts it to an empirical percentile.**

RAW PCA Q → EMPIRICAL CALIBRATION




ONE CUTOFF · FIVE VALIDATION RUNS

Cutoff: 0.989791

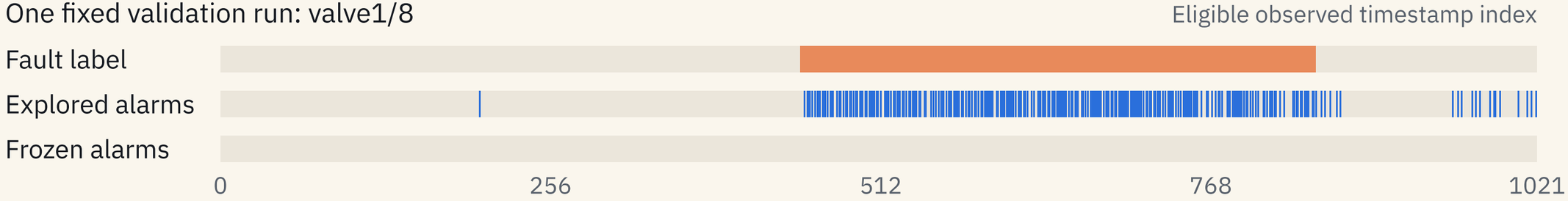
Normal alarm time: 6.639%

206 / 3103 normal timestamps · budget ≤ 1%

State F1: 0.737 Ranking AP: 0.855

This explored cutoff violates the alarm budget

One fixed validation run: valve1/8

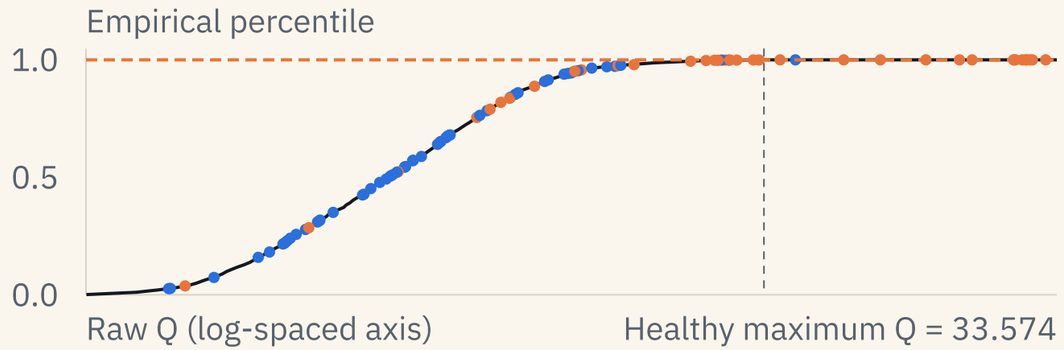



Keyframe 1/3 · Map a raw score — Cutoff 0.989791 · pooled normal alarm 6.639% · macro state F1 0.737 · real validation only

# Useful ranking can fail at the chosen cutoff

**At the top available ECDF rank, 41 of 3,103 normal validation timestamps alarm: 1.321%, above the 1% budget.**

RAW PCA Q → EMPIRICAL CALIBRATION




ONE CUTOFF · FIVE VALIDATION RUNS

Cutoff: 0.999556

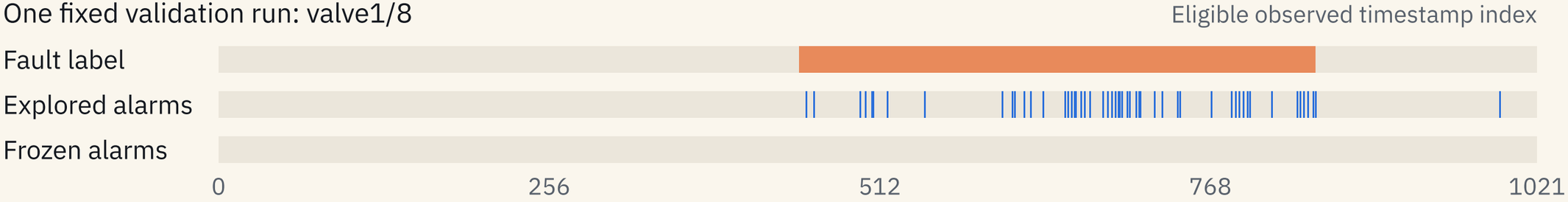
Normal alarm time: 1.321%

41 / 3103 normal timestamps · budget ≤ 1%

State F1: 0.525 Ranking AP: 0.855

This explored cutoff violates the alarm budget

One fixed validation run: valve1/8



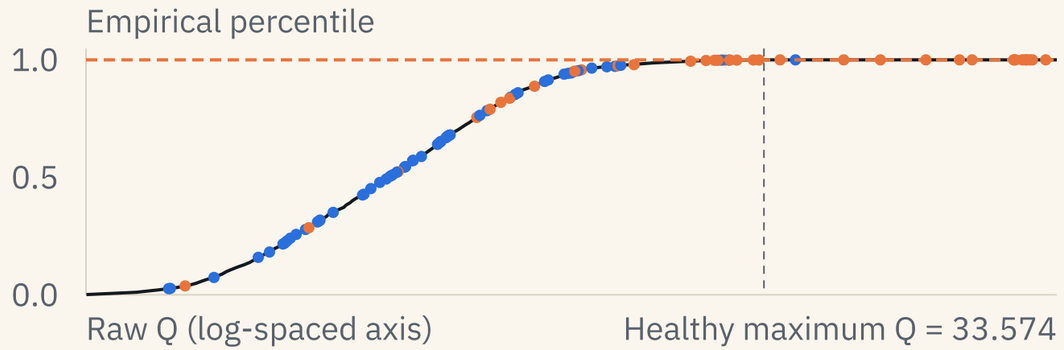

Keyframe 2/3 · Enter the top bin — Cutoff 0.999556 · pooled normal alarm 1.321% · macro state F1 0.525 · real validation only

WORKED EXAMPLE

# Useful ranking can fail at the chosen cutoff

**The frozen feasible cutoff lies just above the top bin. No validation observations alarm, so state F1 is zero.**

RAW PCA Q → EMPIRICAL CALIBRATION




ONE CUTOFF · FIVE VALIDATION RUNS

Cutoff: Above top bin

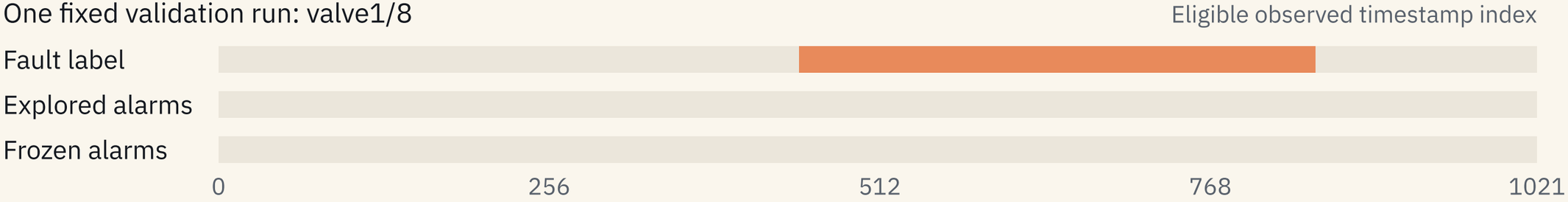
Normal alarm time: 0.000%

0 / 3103 normal timestamps · budget ≤ 1%

State F1: 0.000 Ranking AP: 0.855

Frozen official policy: quiet; F1 = 0

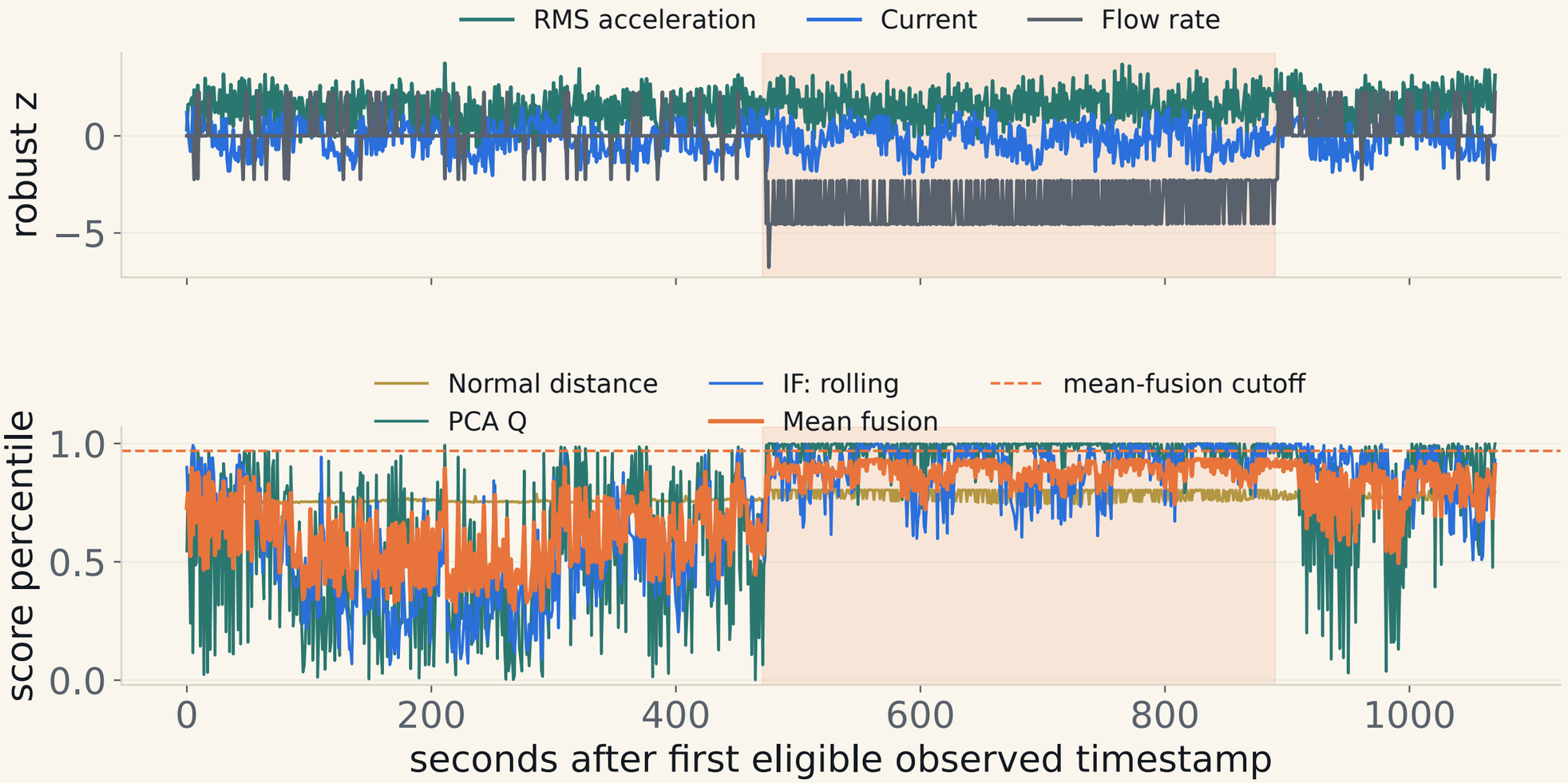
One fixed validation run: valve1/8




Keyframe 3/3 · Respect the budget — Cutoff above top bin · pooled normal alarm 0.000% · macro state F1 0.000 · real validation only

# Averaging helps here; max inherits the saturated tail

Fixed validation run: **valve1/8**. Shading = abnormal state. Mean fusion averages three calibrated scores.



# Compare complete anomaly-detection pipelines

| Pipeline | Macro AP | Macro state F1 | Normal alarm time |
|---|---|---|---|
| Mean fusion | 0.877 | 0.522 | 0.999% |
| Persistence residual | 0.435 | 0.039 | 0.967% |
| Chronos residual | 0.442 | 0.091 | 0.967% |
| Chronos interval | 0.362 | 0.001 | 0.902% |

8 targets · context 128 · horizon/stride 16 · cross-learning across runs disabled.
Local CPU: 7.82s for 816 forecast tasks; model loading excluded.

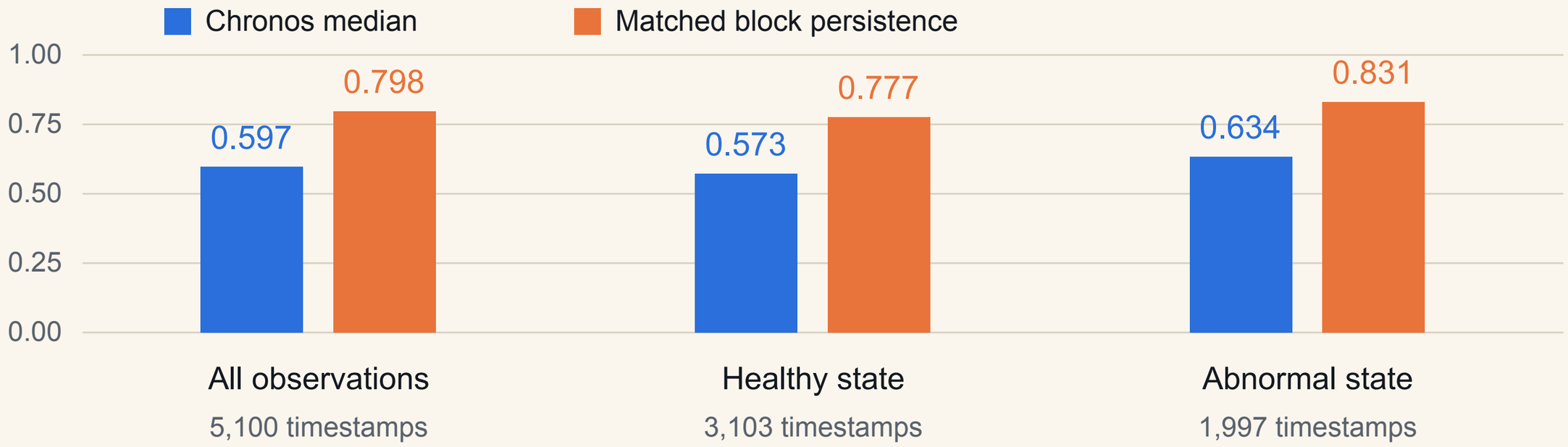
> These are selected detector pipelines. Forecast accuracy is measured separately on the next slides.

# Better forecasts do not guarantee better state detection

Chronos reduces point-forecast error on these validation runs; its selected residual detector still misses most faulty timestamps.

Same 5 validation runs · 8 target channels · same 16-step forecast origins

Normalized MAE ↓ · equal weight for each run × channel

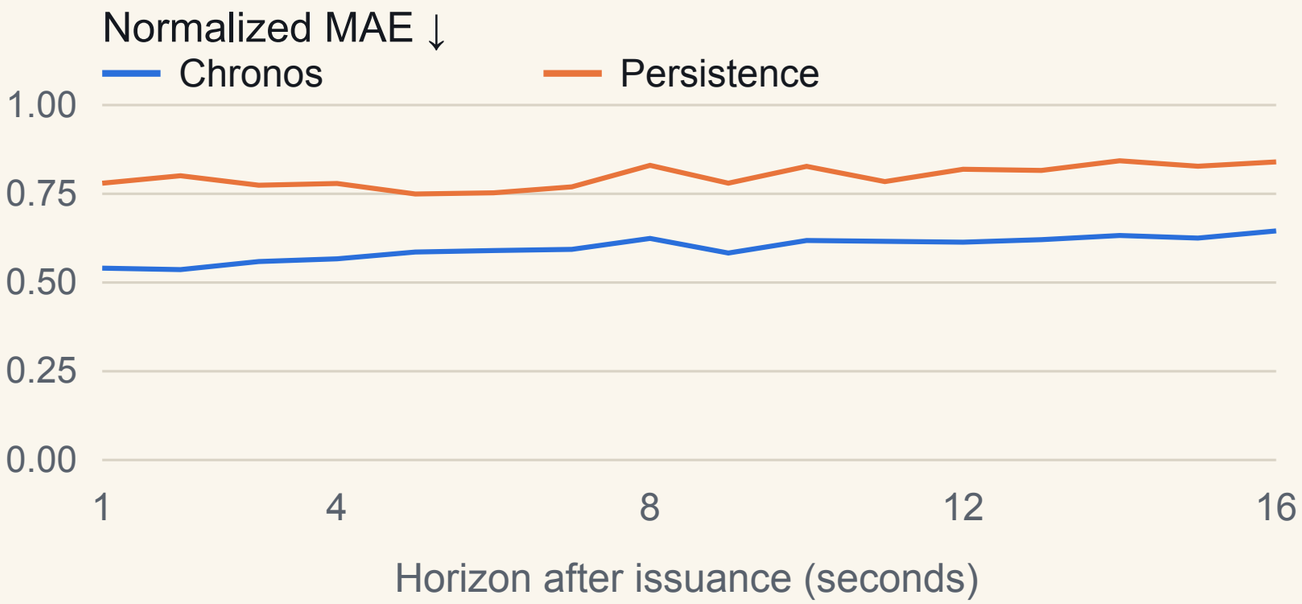



Overall error reduction: 25.2%

# Read the horizon and check the interval

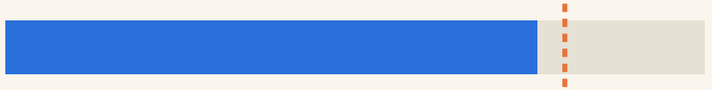
The nominal 80% Chronos interval covers about 76% here. Read coverage together with width.

## Compare forecast errors by horizon; measure interval coverage




### Chronos [q10, q90]

Nominal interval: 80%



Observed coverage: 76.0%

Mean width / fit scale: 1.836

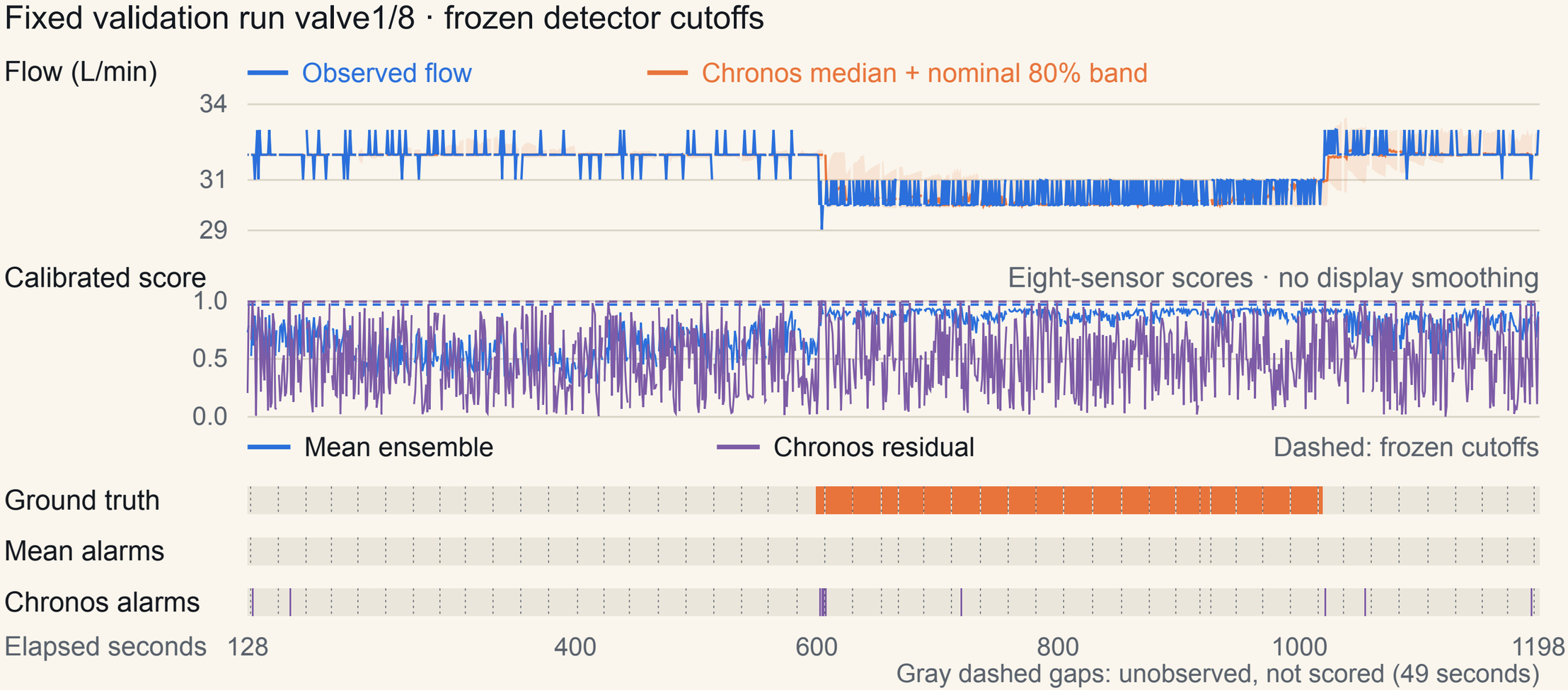
Persistence supplies a point, not a prediction interval.

Coverage and width are paired. A wide interval can cover more without being more informative.

Validation only · 21 / 40,800 median-outside-band cases; raw quantiles retained.

# Read the score, then read the actual alarm

A high score is not an alarm. On this fixed run, the mean ensemble issues no alerts at its frozen cutoff.




Mean: 0 / 400 abnormal timestamps detected. Chronos: 5 / 400, plus 5 false-positive timestamps.

# Serega can read the forecast. You define the question.

Specify the target.
Follow the axes and the masks.
Compare with an honest baseline.

> In the lab: explain one success and one failure before changing the model.

