

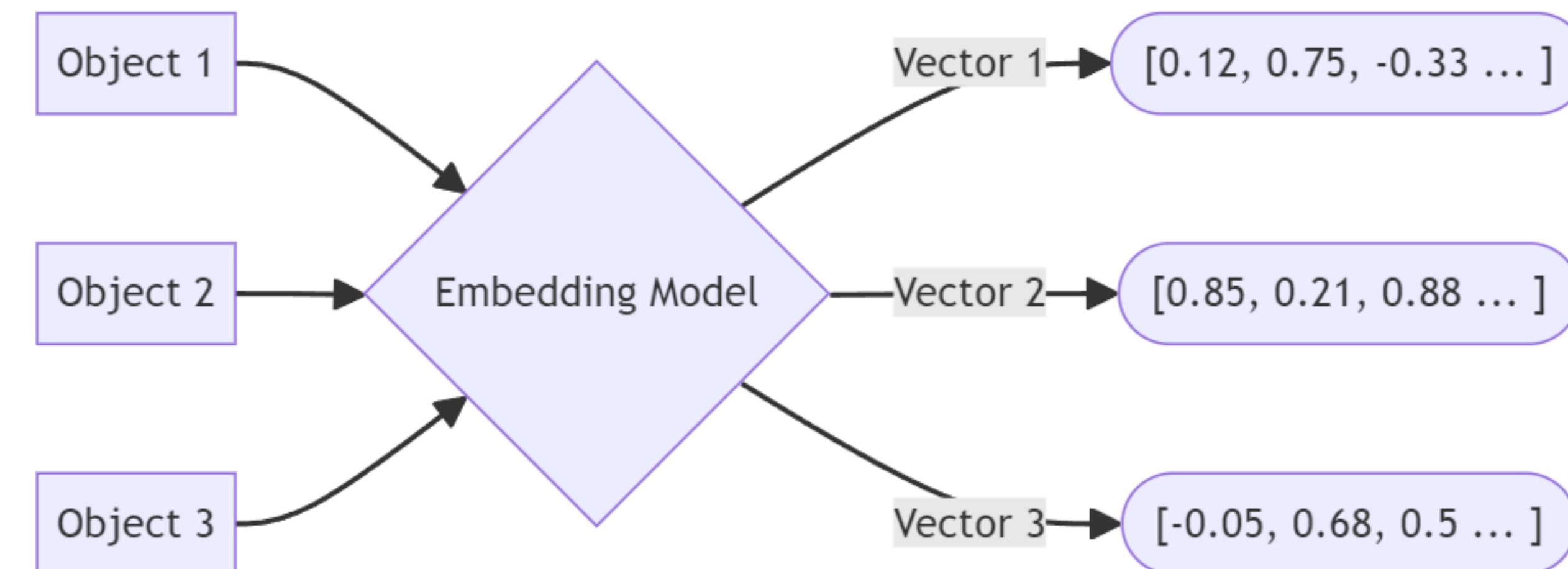
# Обучение представлений

День 4

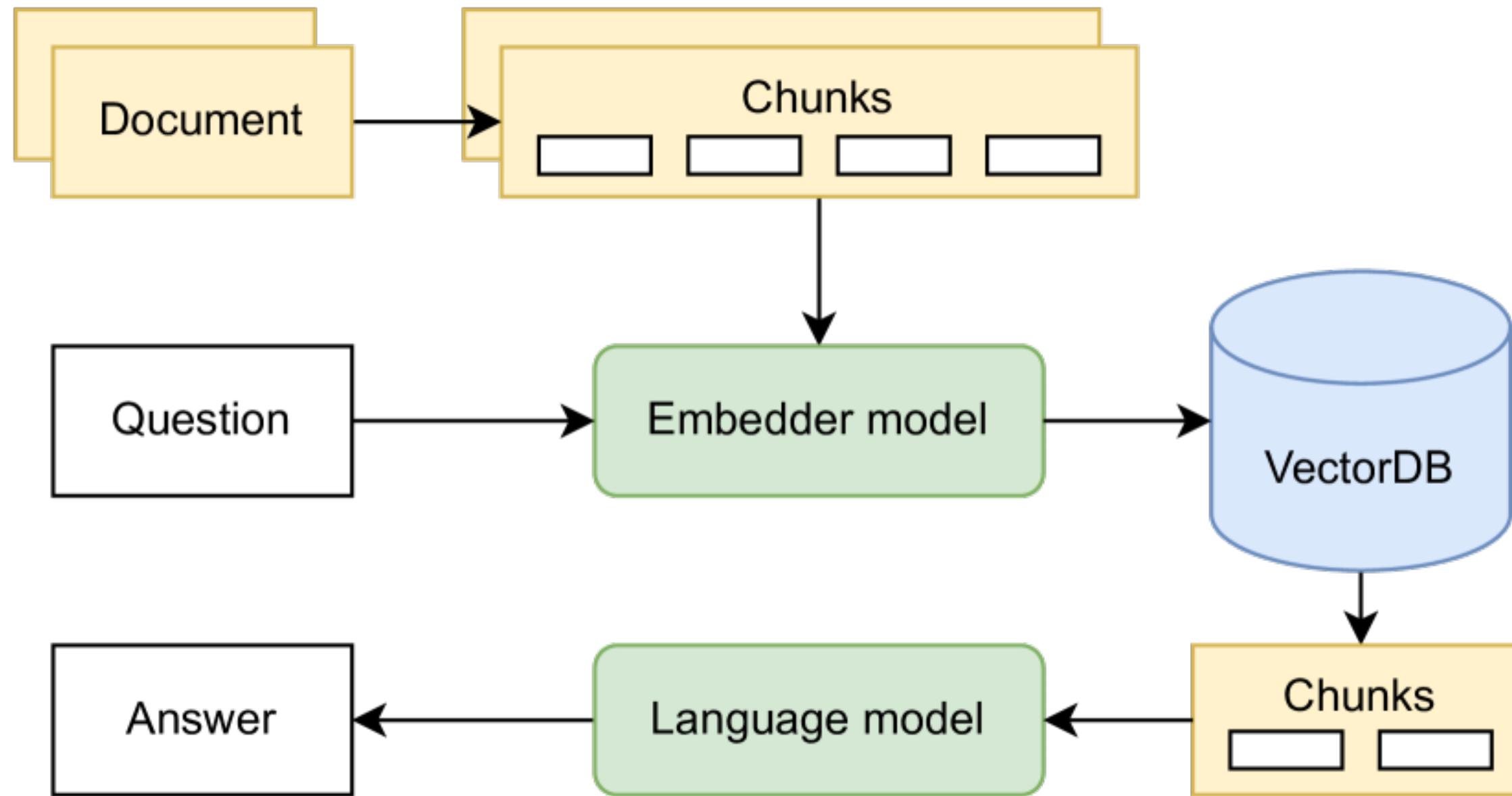
# Определение

Эмбеддинг — векторное представление произвольного объекта.

- **Поточечные** — точка временного ряда.
- **Посегментные** — временной ряд.



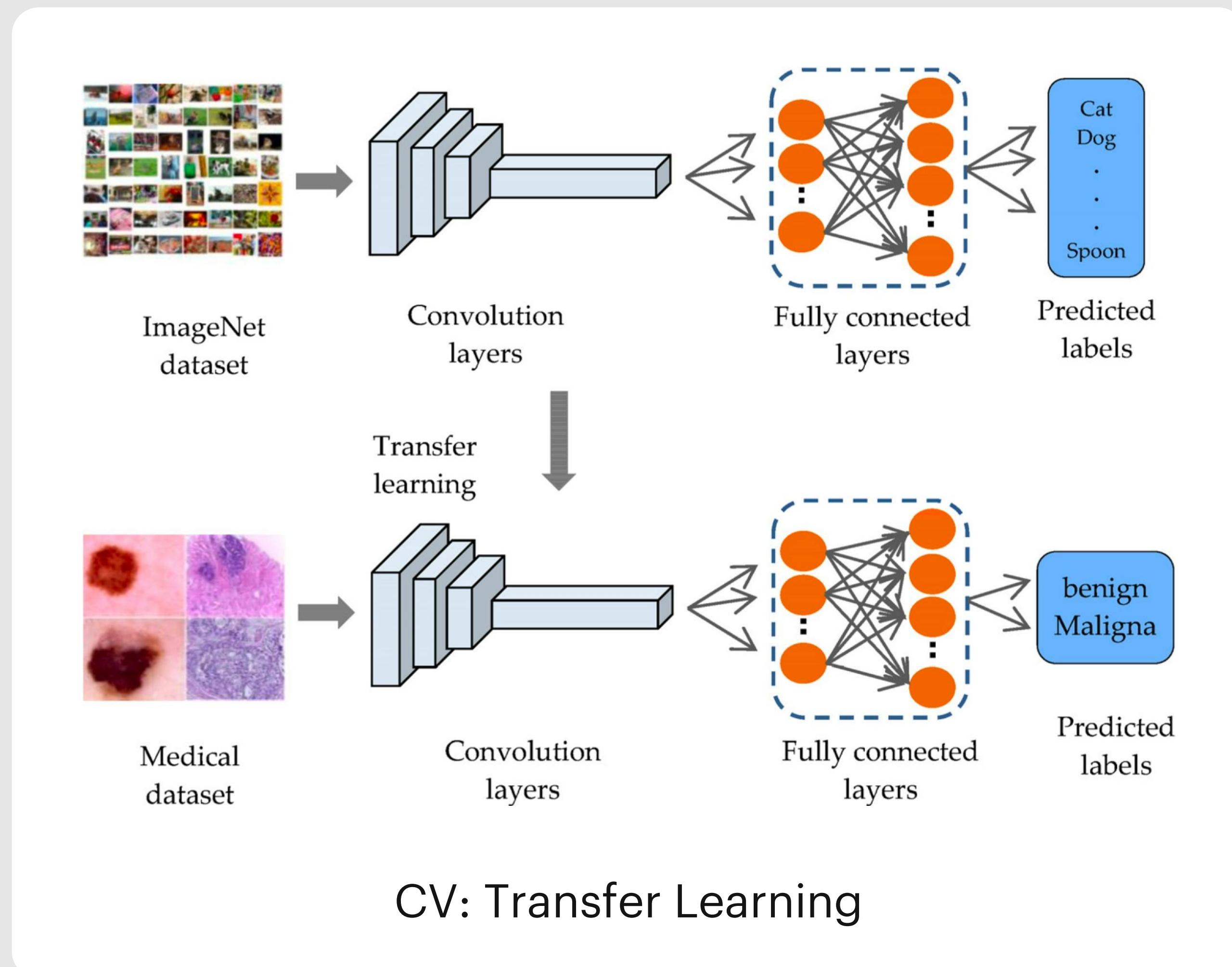
# Зачем нужны эмбеддинги?



- Представляем тексты в виде векторов.
- Умеем искать семантически близкие тексты.

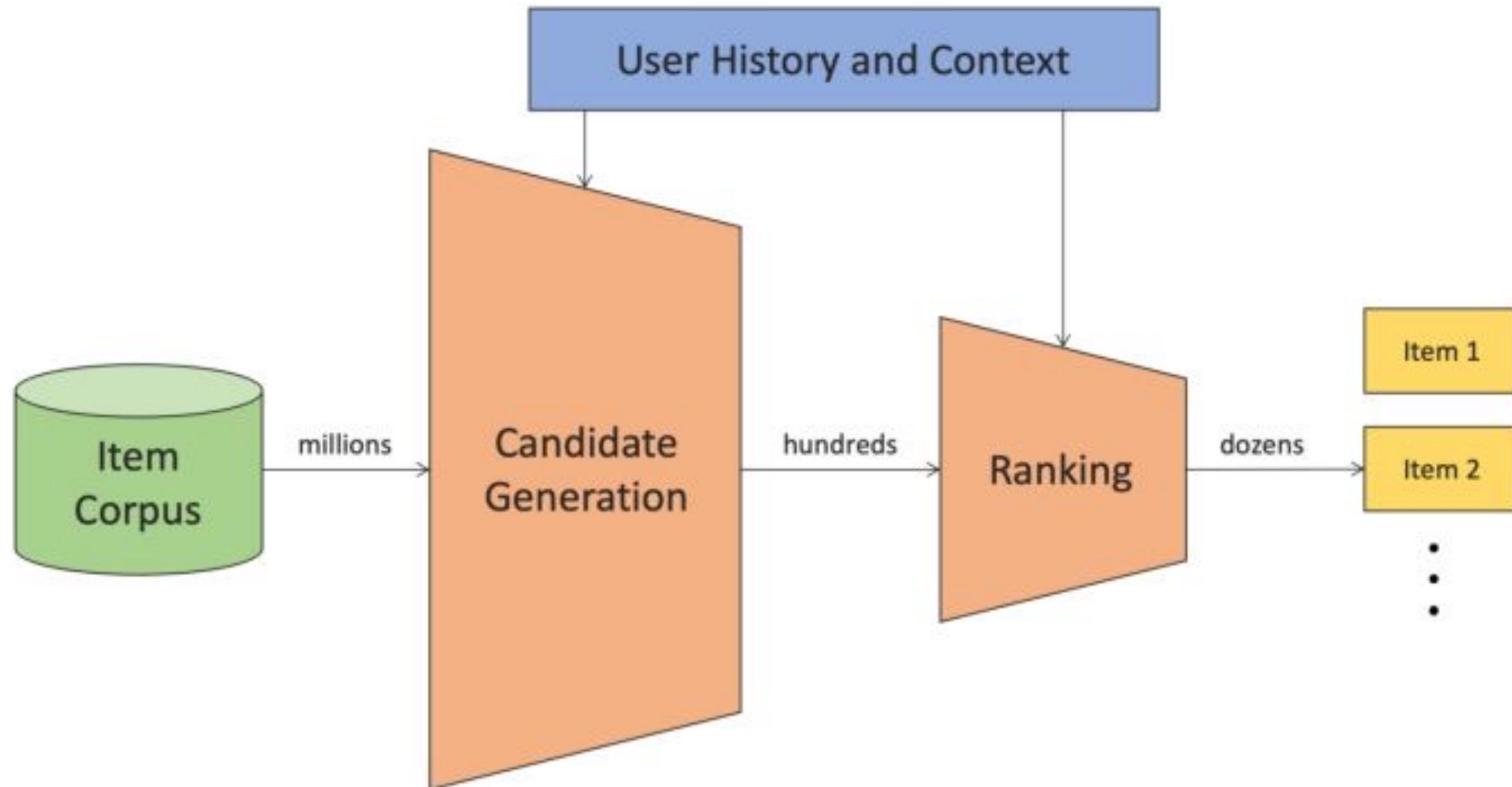
NLP: RAG

# Зачем нужны эмбеддинги?



- Учим эмбеддинги на большом корпусе картинок.
- Используем готовые эмбеддинги для других задач.

# Зачем нужны эмбеддинги?



- Учим эмбеддинги для пользователей и айтемов.
- Часто мультимодальные (тексты/картинки/графы).
- Используем:
  - для кандидатогенерации,
  - ранжирования.

Recsys: какие айтемы нравятся пользователю?

# Зачем нужны эмбеддинги?

Forecasting

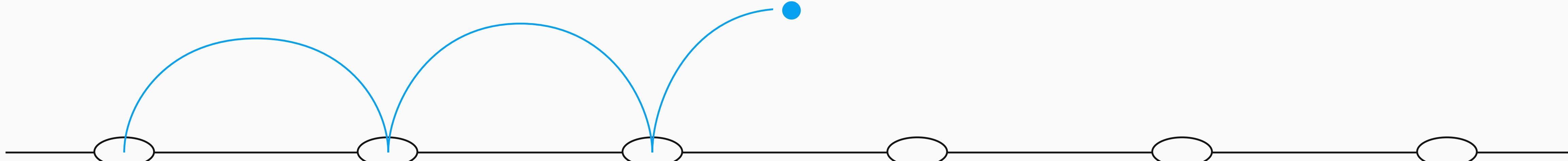
Anomaly detection

Meta Learning

Classification

Clustering

Time Series: везде нужны признаки, а **какие**?

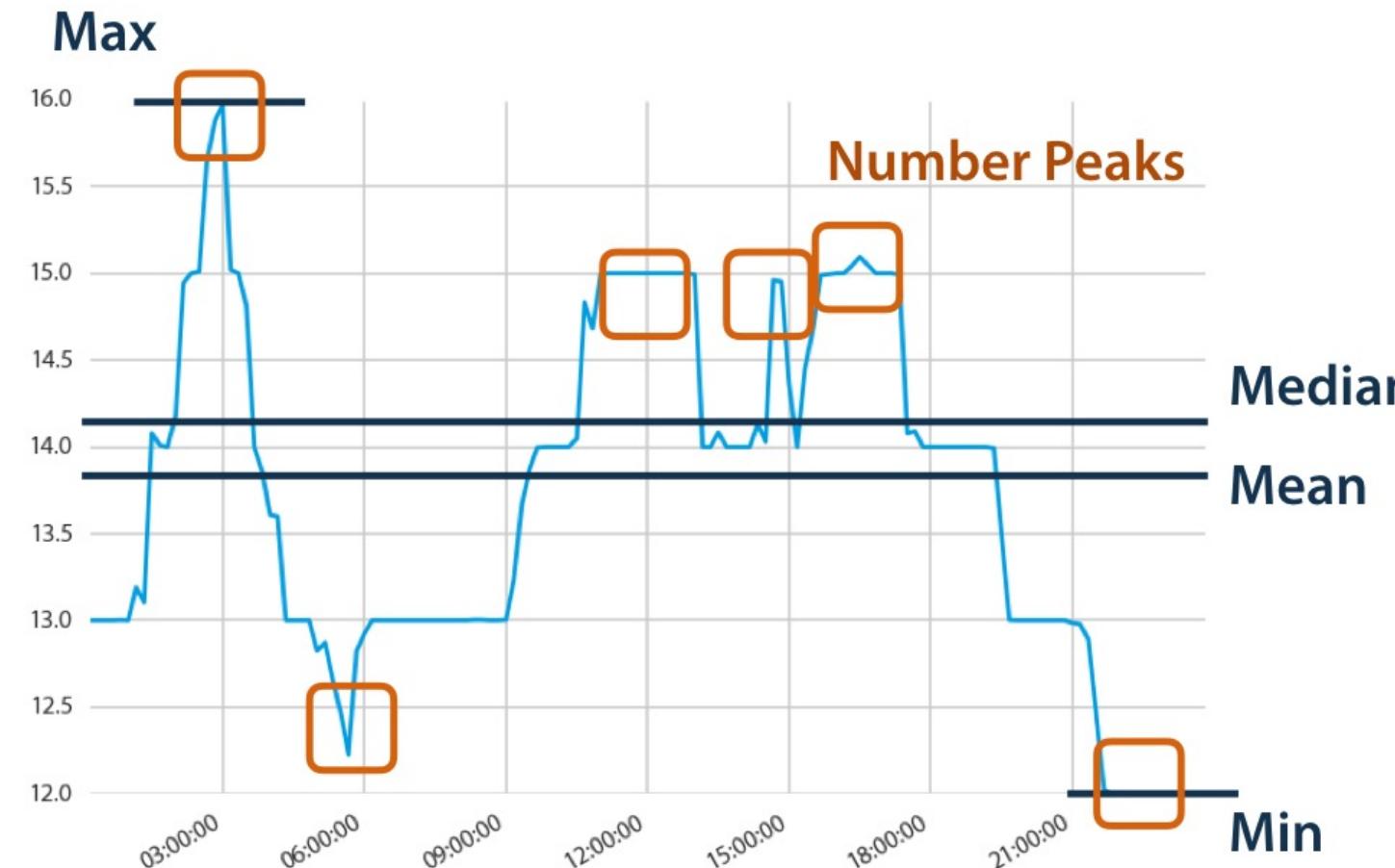


# Основные подходы

- **Классические признаки:** извлекаем набор статистических характеристик из ряда.
- **Методы классификации и кластеризации (skip)** — отдельные походы, не всегда можно в чистом виде получить признаки.
- **DL-подходы:** обучаем self-supervised-модель.

# Классические признаки

# Идея



- Посчитаем некоторый набор характеристик ряда.  
Например:
- **простейшие статистики:** min, max, std и другие;
- **структурные характеристики:** trend, seasonality и другие;
- **предсказуемость:** stability, lumpiness, Shannon entropy;
- ...

Фактически просто применяем набор параметрических функций от ряда:

$$\{f_i(\theta) : P \rightarrow R\}_{i=1}^n.$$

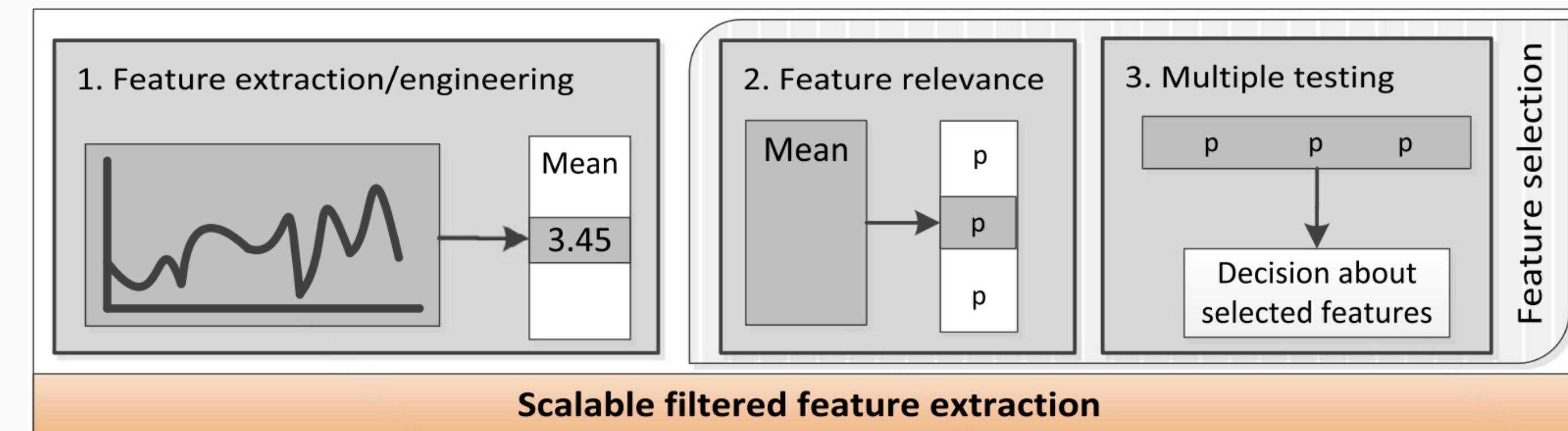
# Популярные библиотеки



# Классические признаки. TSFresh + TSFel

# TSFresh

- 63 метода выделения фичей → 794 фичи.
- Автоматический отбор признаков по статтестам.
- Поддерживает параллелизацию:
  - Multiprocessing/Dask/Spark.





# TSFresh (Признаки)

## Базовые

- [absolute\\_maximum](#)
- [count\\_above \(x, t\)](#)
- [has\\_duplicate \(x\)](#)
- [length \(x\)](#)

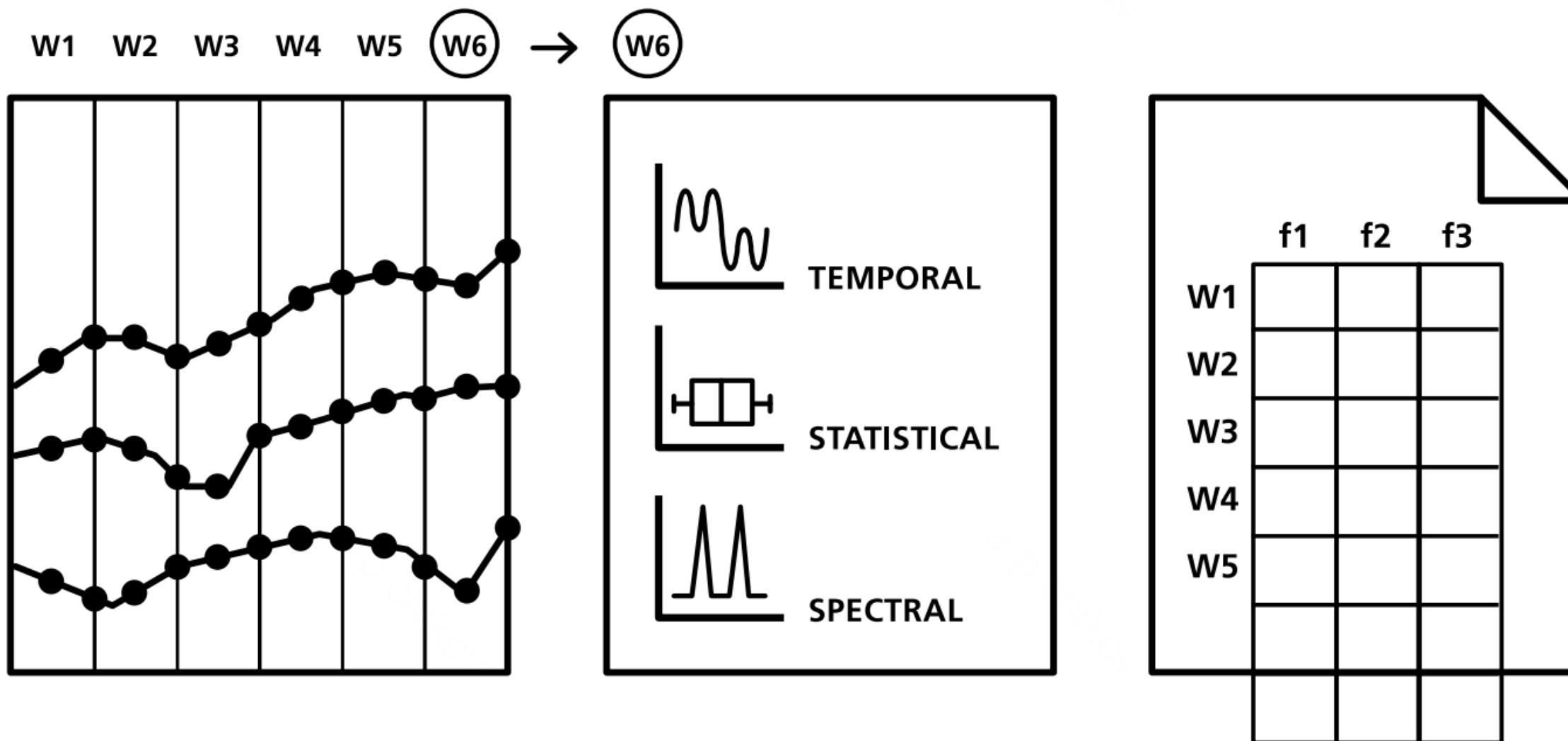
## Интересные

- [agg\\_autocorrelation \(x, param\)](#)
- [agg\\_linear\\_trend \(x, param\)](#)
- [ar\\_coefficient \(x, param\)](#)
- [augmented\\_dickey\\_fuller \(x, param\)](#)
- [C3 \(x, lag\)](#)
- [cid\\_ce \(x, normalize\)](#)
- [lempel\\_ziv\\_complexity \(x, bins\)](#)
- [symmetry\\_looking \(x, param\)](#)

60 фичей из разны  
доменов:

- temporal,
- statistical,
- spectral,
- fractal.

Для каждой фичи  
оценели сложность  
на синтетике  $O(n^2)$ ,  
 $O(n \log n)$ ,  $O(n)$ ,  
 $O(\log n)$  and  $O(1)$ .



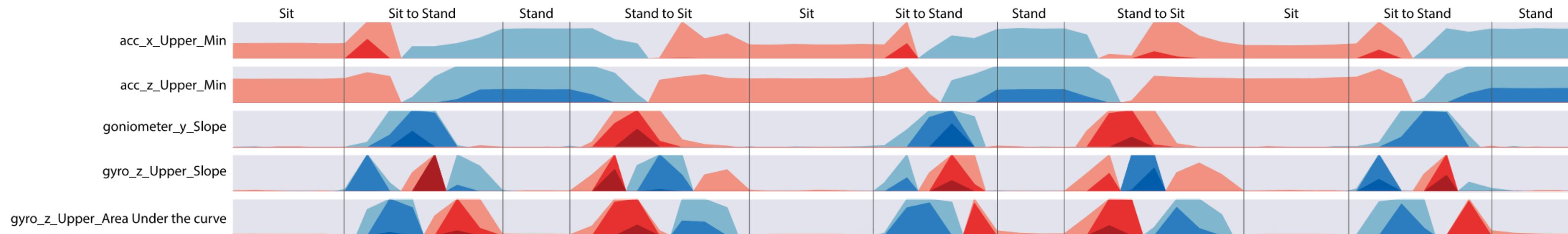
# TSFel (Human Activity Recognition)



Показали демо на задаче HAR.

- Есть данные с разных датчиков:
  - Accelerometer/Gyroscope/Goniometer.
- Нужно определить, чем занимается человек:
  - Stand/Sit/Stand-to-sit/Sit-to-stand.

- **acc\_x\_Upper\_Min** и **acc\_z\_Upper\_Min** хорошо разделяют sit vs stand.
- **goniometer\_y\_Slope** хорошо разделяет sit-to-stand vs stand-to-sit.



# Классические признаки. Catch 22

# Catch22

**Идея:** найдём минимальный по количеству и максимально разнообразный набор фичей, который хорошо работает на любой задаче классификации.

- Сделали фильтрацию 7658 → 22 фичи.
- Подбирали под 93 классификационных датасета.

Dataset	Train Size	Test Size	Length	No. of Classes	Type
AbnormalHeartbeat	303	303	3053	5	AUDIO
ACSF1	100	100	1460	10	DEVICE
Adiac	390	391	176	37	IMAGE
AllGestureWiimoteX	300	700	0	10	HAR
AllGestureWiimoteY	300	700	0	10	HAR
AllGestureWiimoteZ	300	700	0	10	HAR
ArrowHead	36	175	251	3	IMAGE
ArticularyWordRecognition	275	300	144	25	MOTION
AsphaltObstacles	390	391	0	4	MOTION
AsphaltObstaclesCoordinates	390	391	0	4	MOTION
AsphaltPavementType	1055	1056	0	3	MOTION
AsphaltPavementTypeCoordinates	1055	1056	0	3	MOTION
AsphaltRegularity	751	751	0	2	MOTION
AsphaltRegularityCoordinates	751	751	0	2	MOTION
AtrialFibrillation	15	15	640	3	ECG

[Ссылка](#)

# Catch22 (Фичи)

- **basic statistics** of time-series values (e. g., location, spread, Gaussianity, outlier properties)
- **linear correlations** (e. g., autocorrelation, power spectral features)
- **stationarity** (e. g., StatAv, sliding window measures, prediction errors)
- **entropy** (e. g., auto-mutual information, Approximate Entropy, Lempel-Ziv complexity),
- **physical nonlinear time-series analysis** (e. g., correlation dimension, Lyapunov exponent estimates, surrogate data analysis)
- **linear and nonlinear model parameters**, fits, and predictive power (e. g., from autoregressive moving average (ARMA), Holt-Winters...)
- ....

Много разных параметрических функций от рядов, отражающих разные характеристики

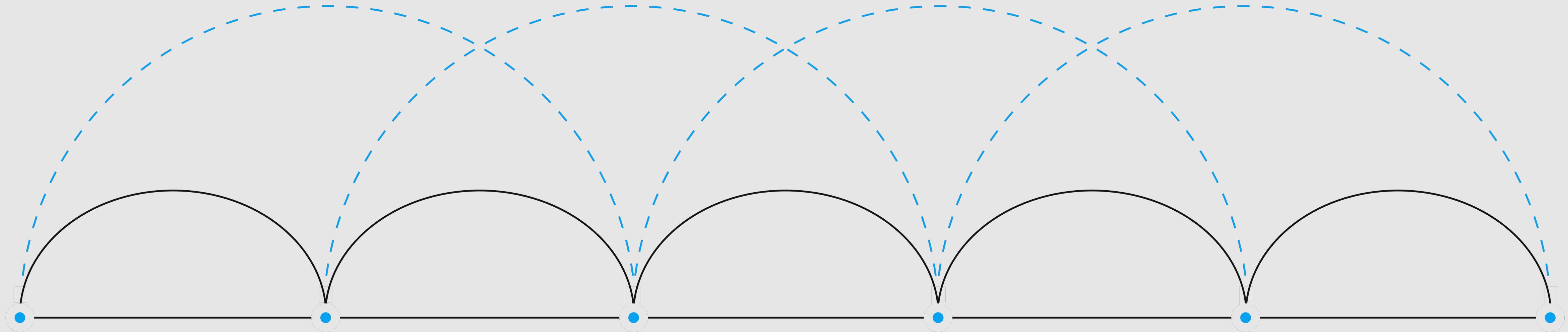


# Catch22 (Dummy Prefiltering)

- Удаляем фичи, чувствительные к масштабу.
  - Удаляем фичи, которые выплываются больше 80% пропусков.
- 
- Ряды в датасетах z-score normalized.
  - Если в задаче важные mean и var, это может сильно скинуть качество (**catch24**).

# Catch22 (Performance Evaluation)

Будем искать фичи, которые хорошо работают **индивидуально** → оцениваем качество классификатора на 1 фиче.

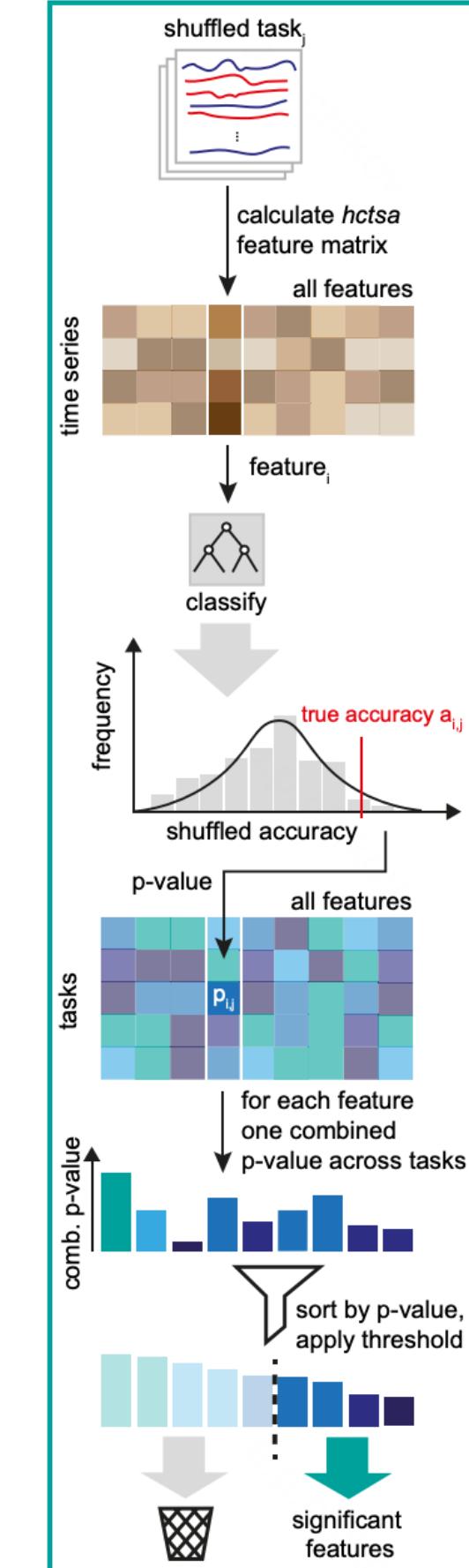


# Catch22 (Performance Evaluation)



Удалим фичи, которые **перформят как рандом**.

- Строим распределение качества случайного классификатора (нормальное).
- Тестируем гипотезу `accuracy_random < accuracy_algo`.
- Делаем коррекцию на множественное тестирование гипотез (Holm-Bonferroni).
- Отбираем значимые фичи.

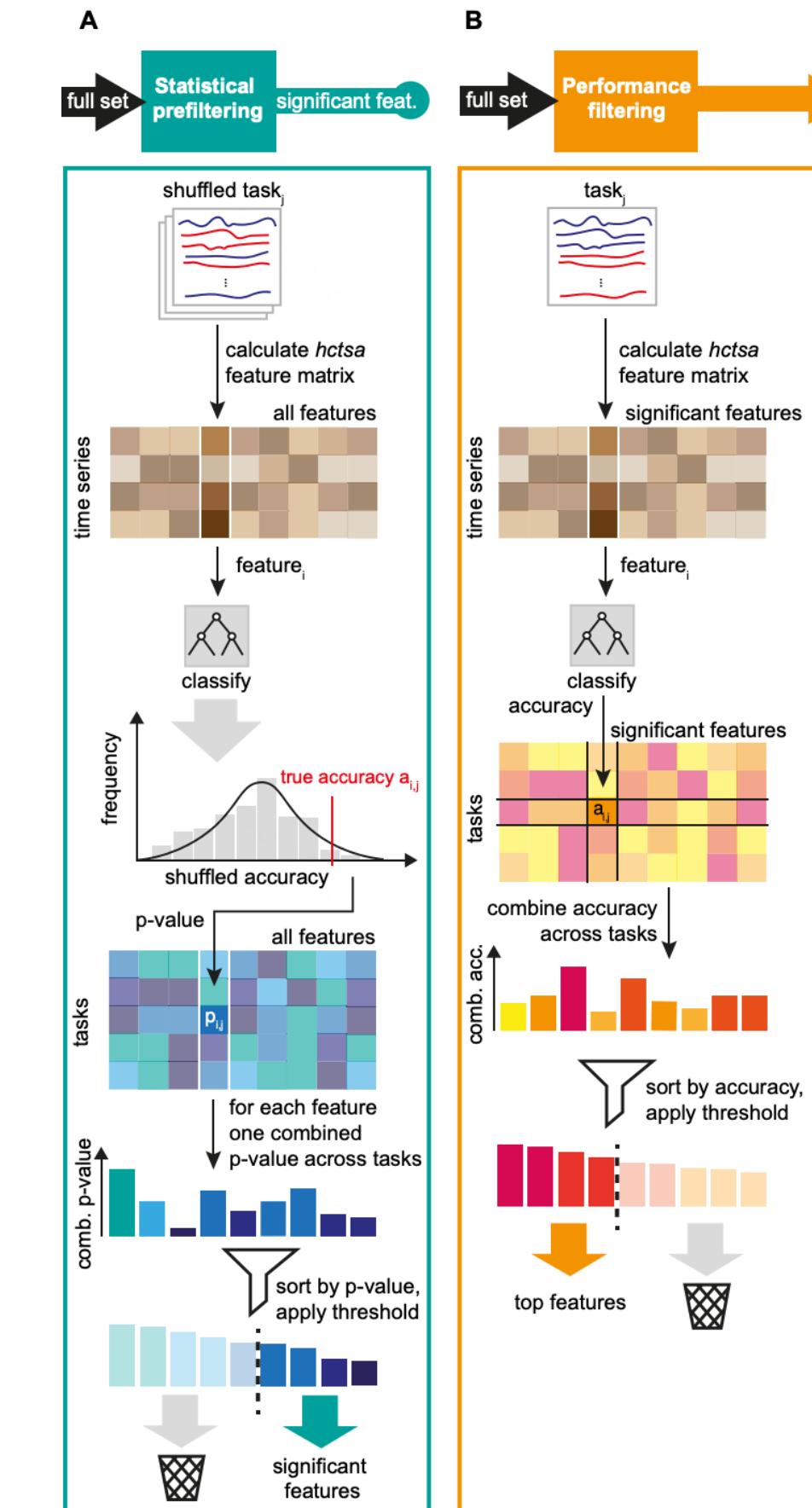


# Catch22 (Performance Evaluation)

Берём топ-N лучших фичей.

- Считаем скоры для каждой задачи:  
 $\text{score} = \text{mean class-balanced classification accuracy}$ .
- Нормализация на среднее accuracy по таску.
- Performance фичи = среднее нормализованное accuracy.

$$a_{i,j}^n = \frac{a_{i,j}}{\bar{a}_j}. \quad a_i^{n,c} = \frac{1}{M} \sum_{j=1}^M a_{i,j}^n$$

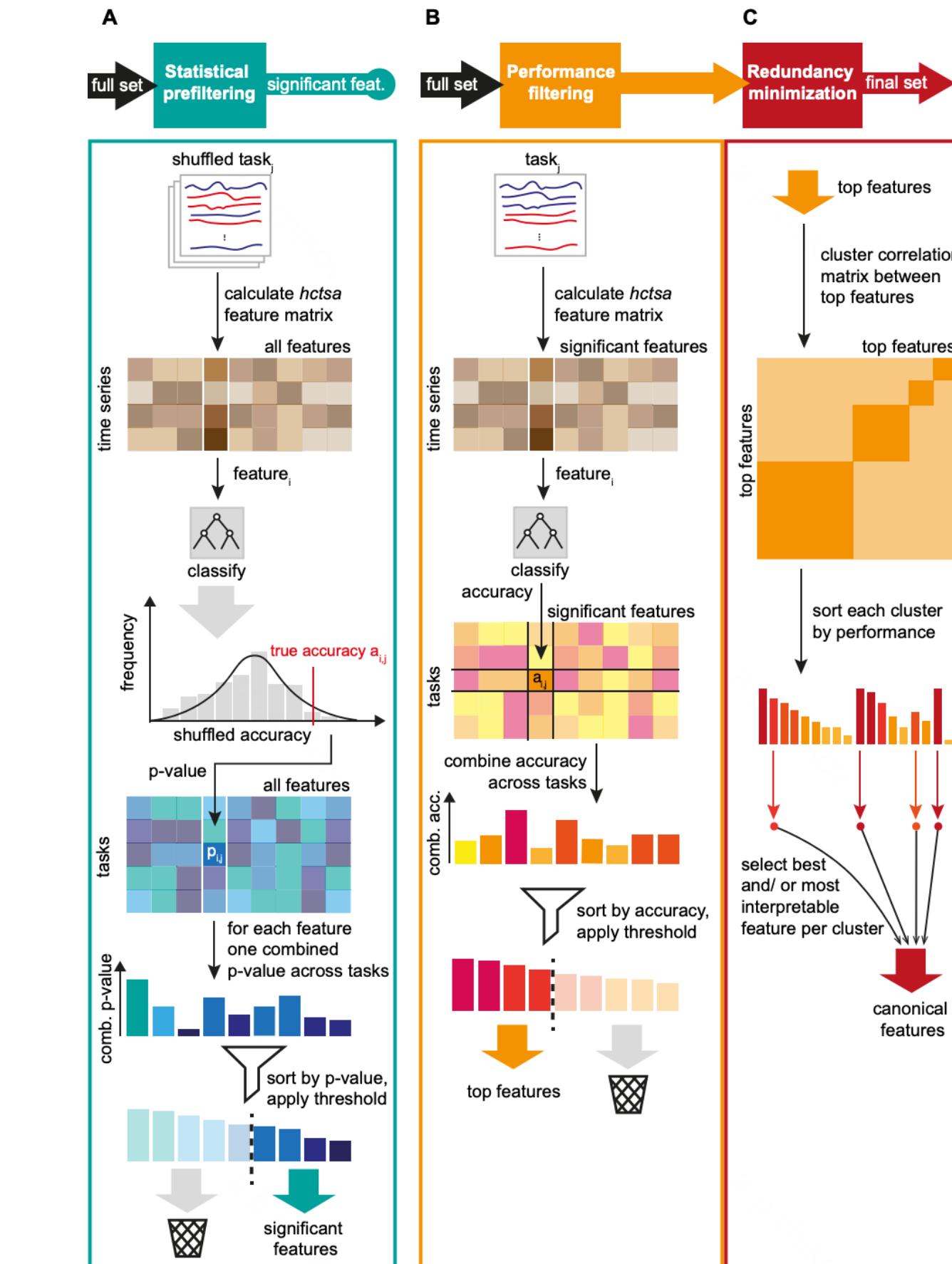


# Catch22 (RedundancyMinimization)

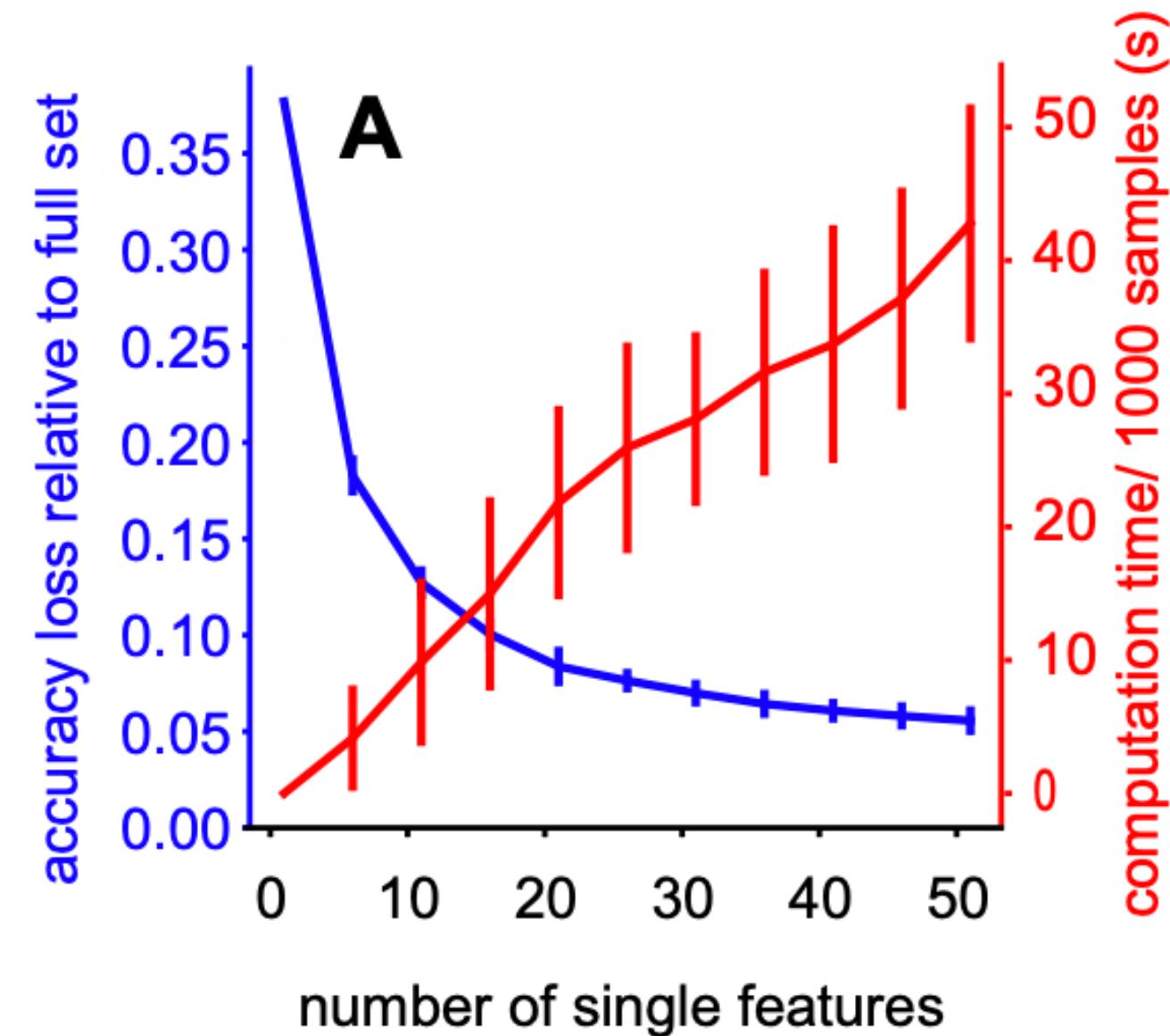


Отбираем максимально разнообразные фичи.

- Иерархическая кластеризация на корреляции векторов перформанса.
- Из кластера по фиче — лучший перформанс среди фичей в кластере + руками.



# Catch22 (Эксперименты)

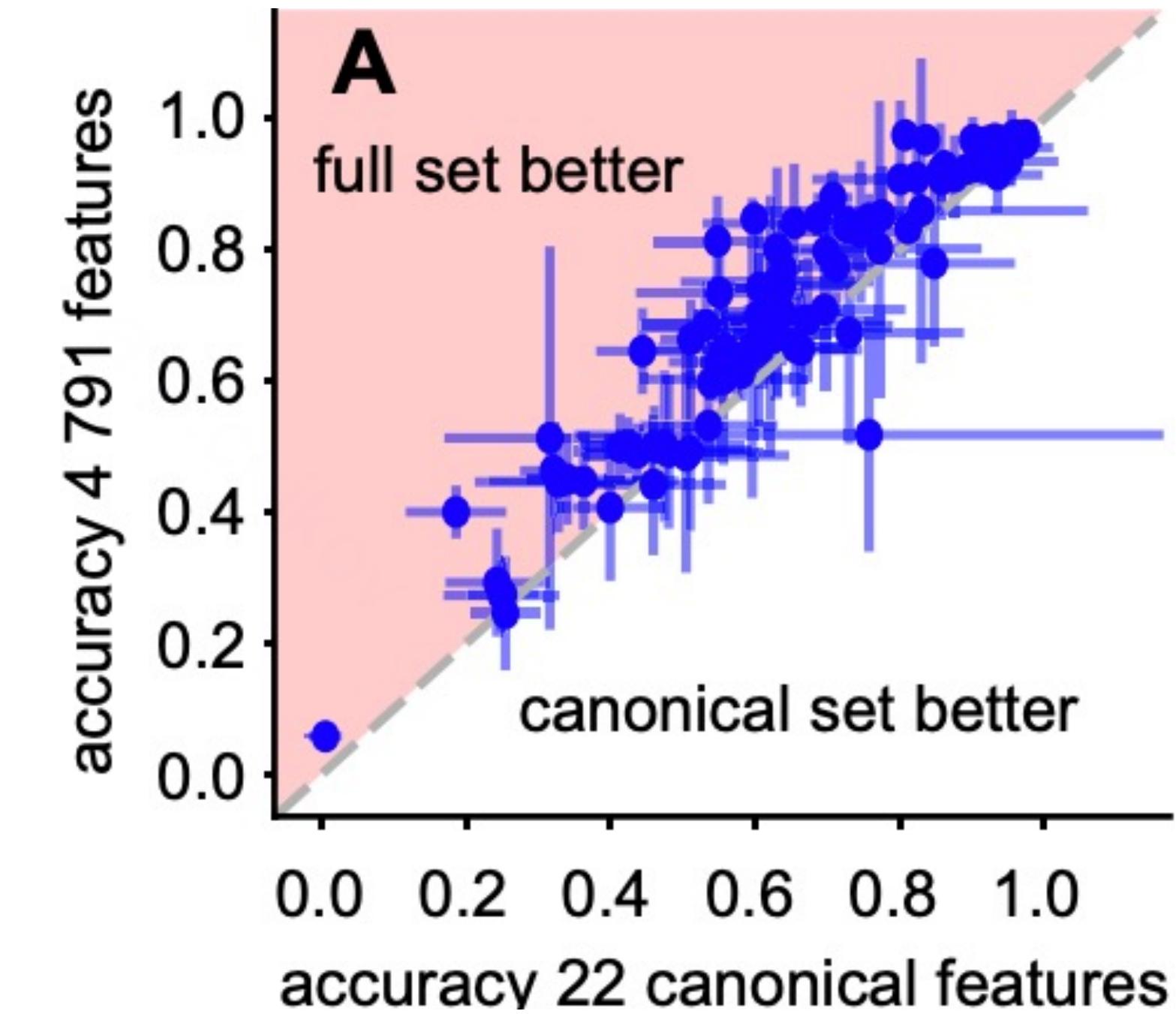


- Синяя — относительная потеря точности.
- Красная — время подсчёта фичей.
- Усы — N от 100 до 1000.

- При 20 фичах (кластерах) точность практически не проседает (5%).
- Большая дисперсия по компьютеру, но не по качеству от N.

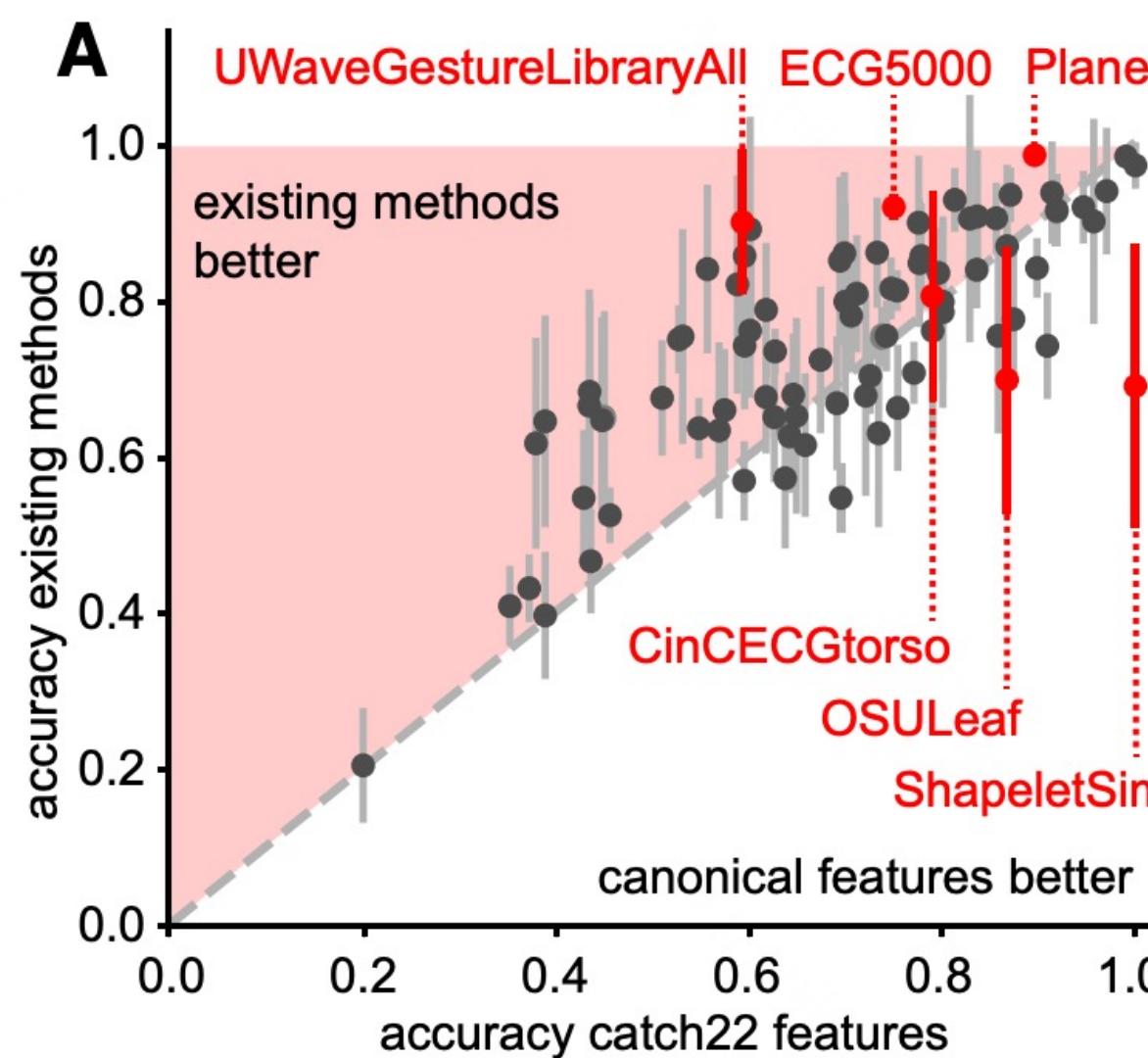
# Catch22 (Эксперименты)

- В среднем работает не сильно хуже:
  - catch22 ~ 72%;
  - 4791 features ~ 77%.
- На конкретных задачах даже обыгрывает полный набор или такое же качество.

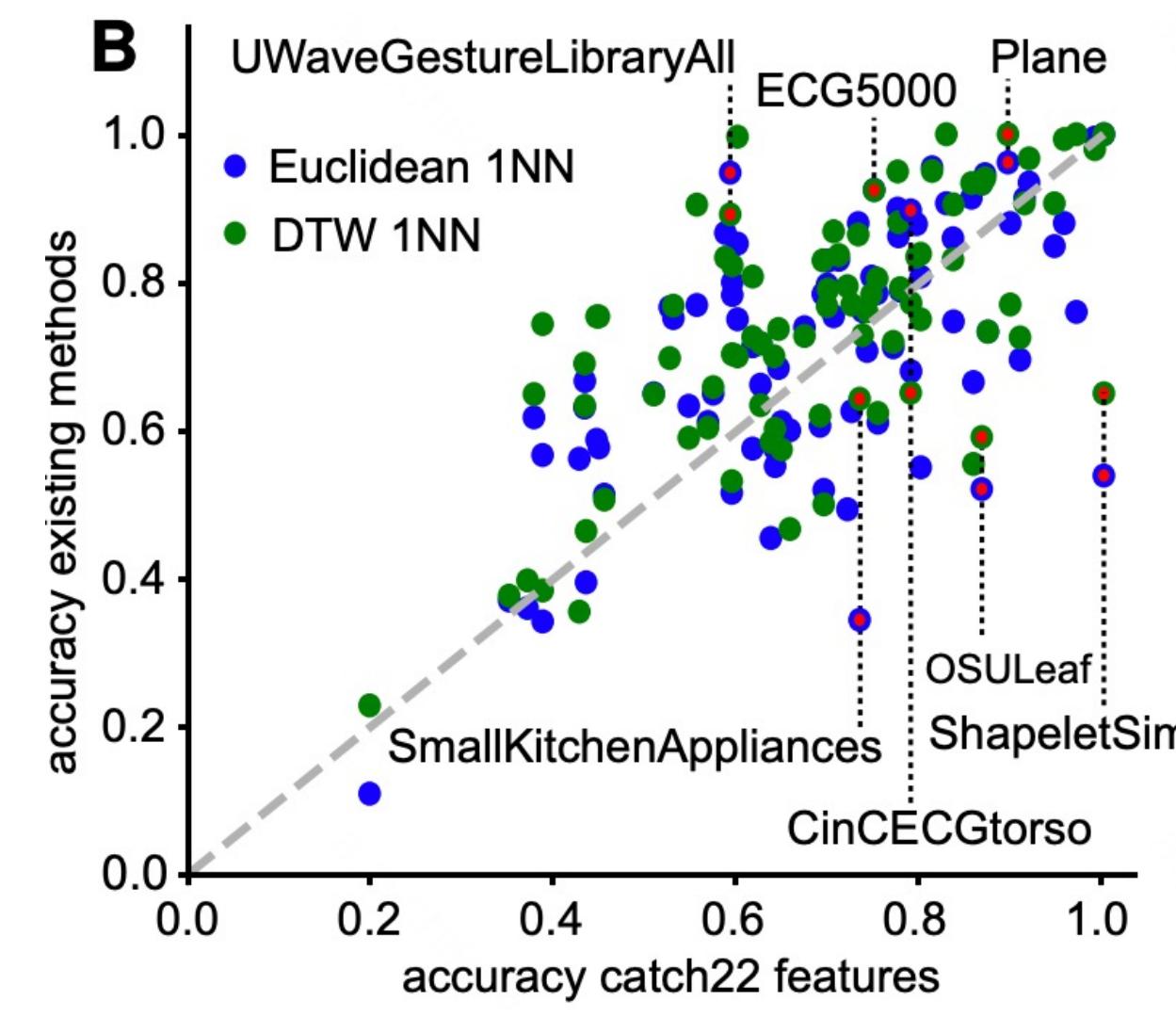


Сравнение качества  
для разных наборов фичей

# Catch22 (Эксперименты)



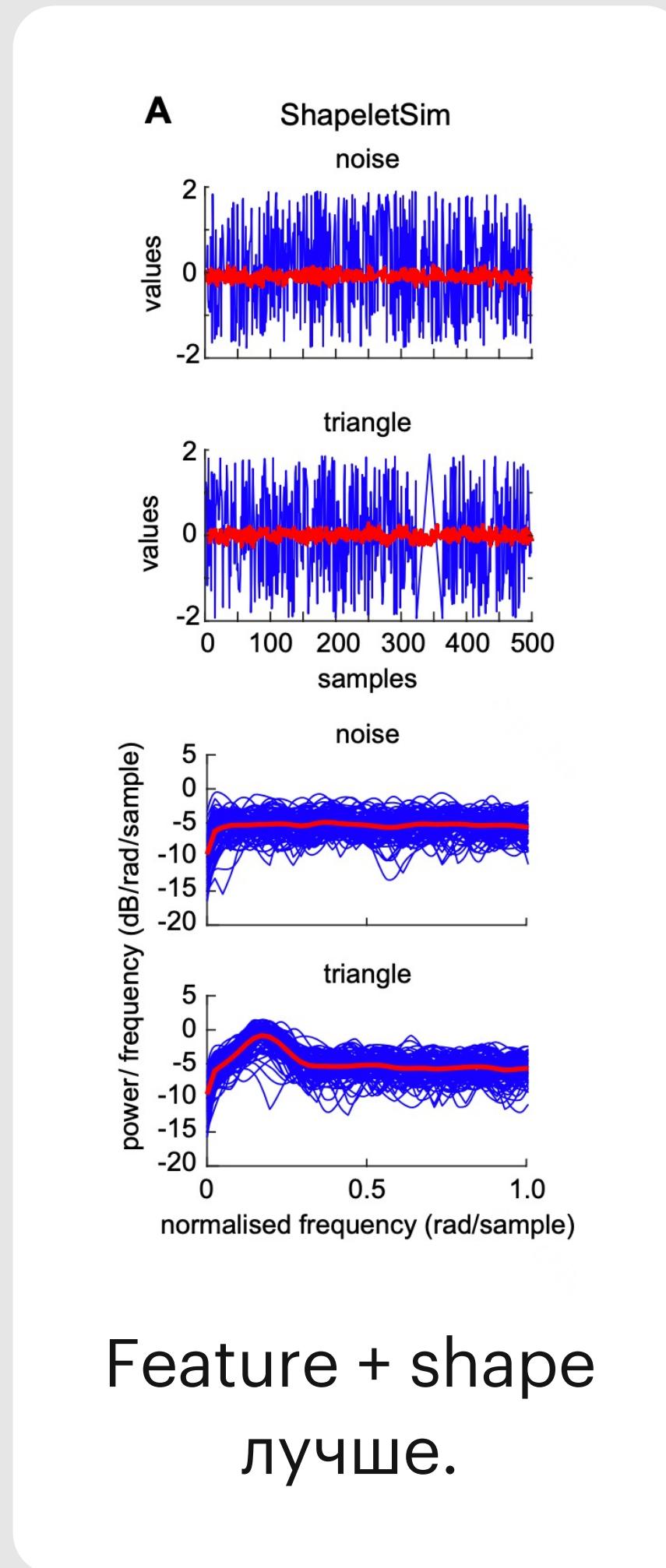
Сравнение  
с классификационными  
подходами



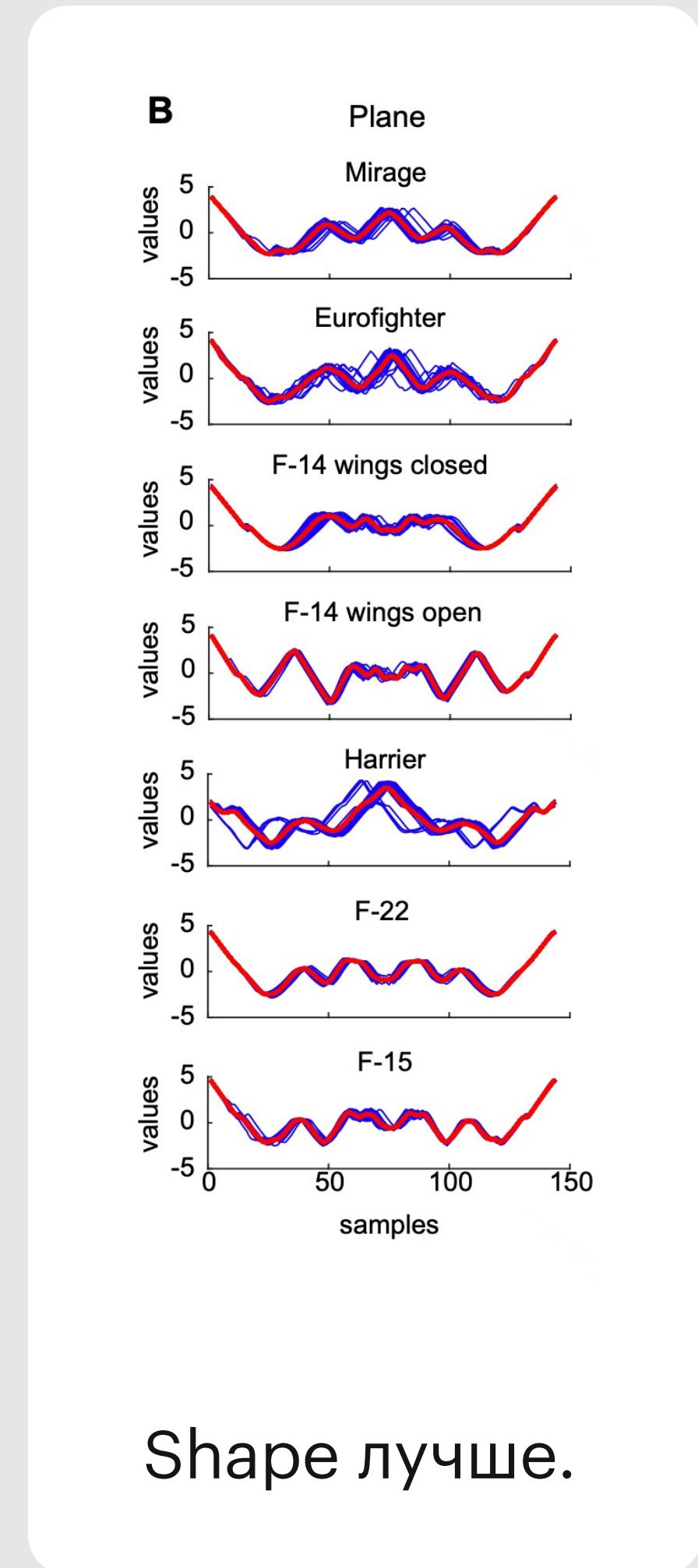
Сравнение с KNN

- Тоже неплохо,  
но где-то заметно хуже.
- Для части датасетов  
важны «форма рядов»  
или наличие паттернов.

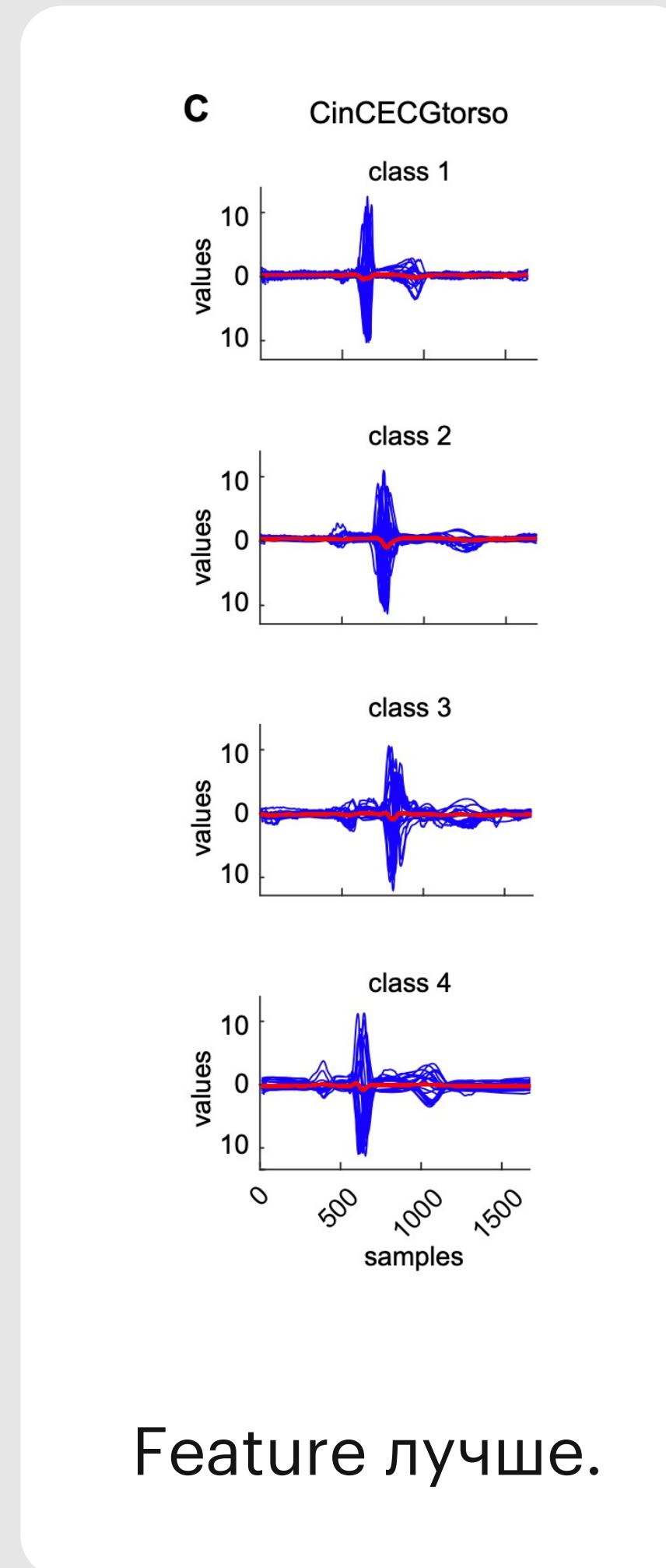
# Catch22 (Эксперименты)



Feature + shape  
лучше.



# Shape лучше.



# Feature лучше.

- Для части датасетов важны «форма рядов» или наличие паттернов.

# Классические признаки. Kats + Meta Learning



# Kats

- Использовали 40 фичей для Meta Learning.
- 6–20x ускорили подбор модели гиперпараметров.

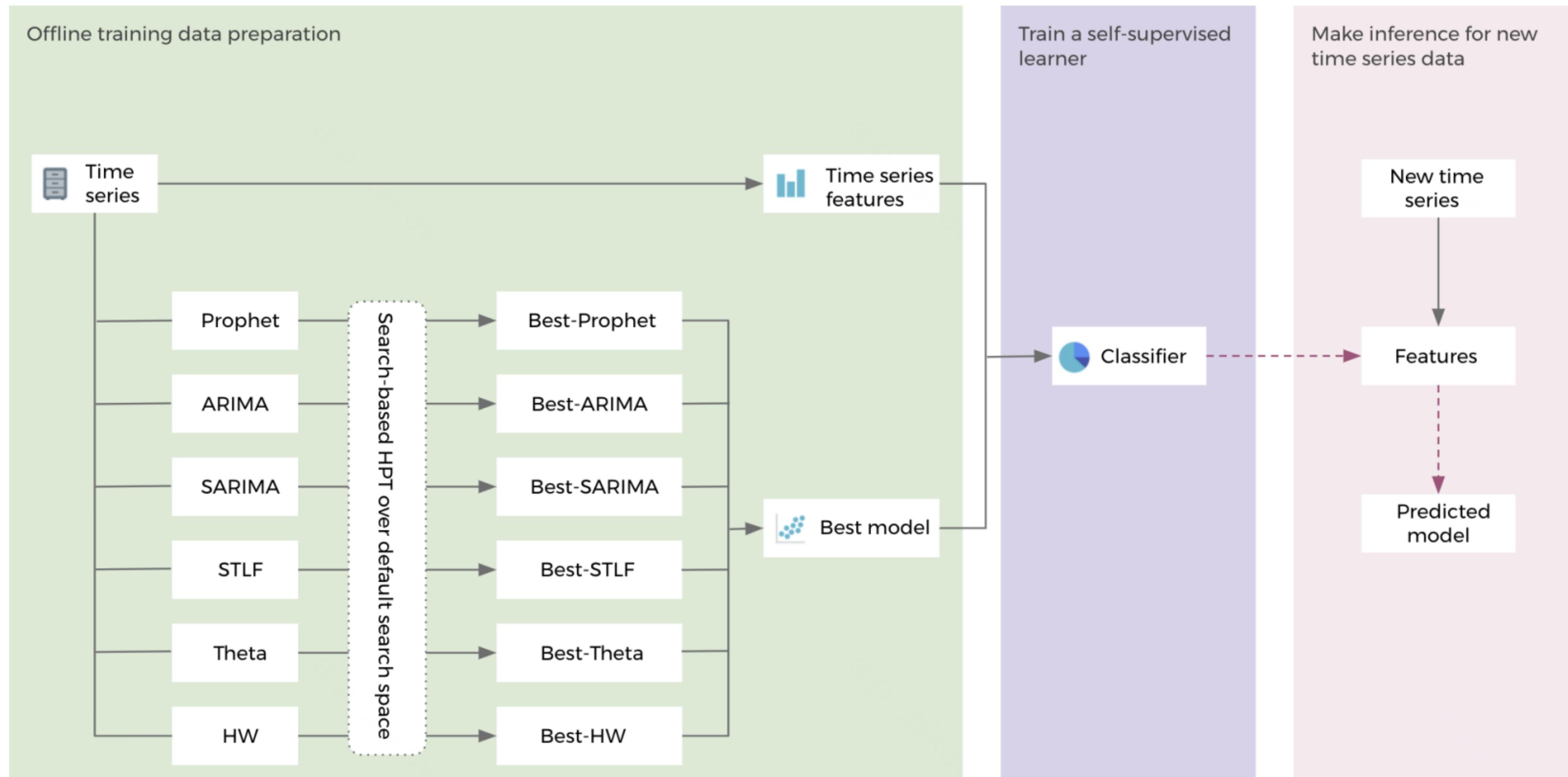
# Kats (Признаки)



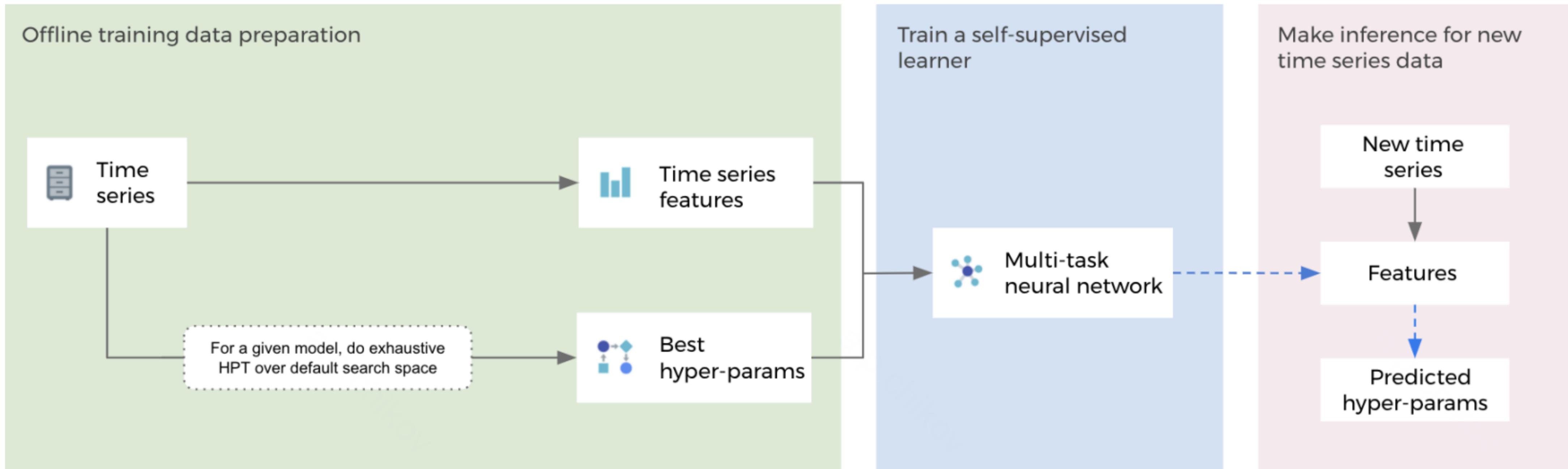
## Примеры признаков

- **Spectral entropy:** Shannon entropy of spectral density function.
- **Lumpiness:** variance of variances within non-overlapping windows.
- **Stability:** variance of means within non-overlapping.
- **Trend/Seasonal strength:** Trend strength is the variance explained by STL trend term, and seasonal strength is the variance explained by STL seasonality terms.
- **Holt-Winter's Parameters:** estimates the smoothing parameter for the level-alpha, trend-beta of HW's linear trend, and additive seasonal trend-gamma.

# Kats (Подбор модели)

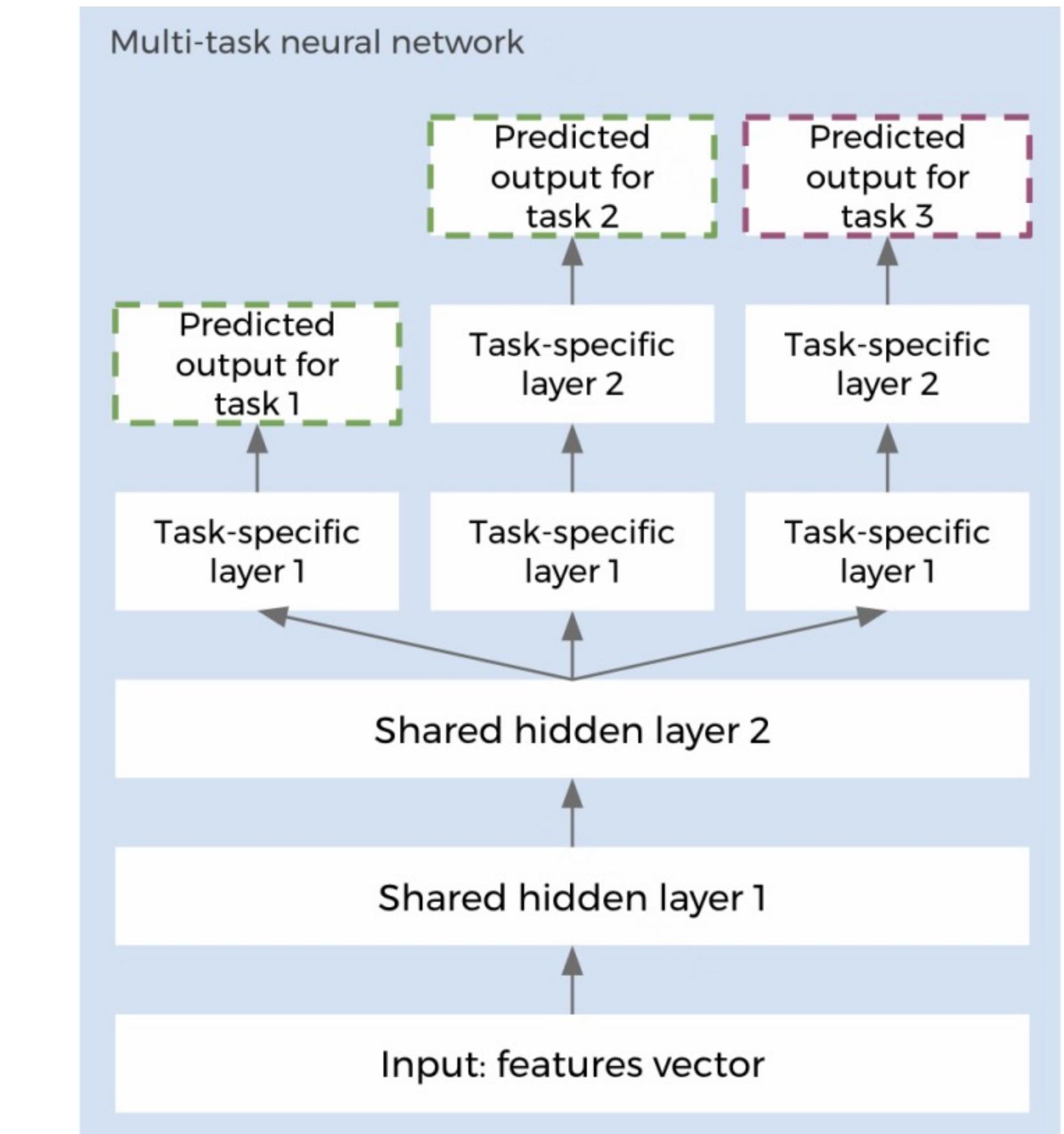


# Kats (Подбор гиперпараметров)



# Kats (Подбор гиперпараметров)

- Отдельная сетка на каждую модель
- Отдельная голова на каждый гиперпараметр
- Parameter Sharing





# Kats: датасет (набор данных)

## Данные

- Facebook\* — 2,852 ряда из разных доменов.
- M3-Competition — 1,428 месячных ряда из разных доменов.

## Модели

STLF, Theta, Prophet, ARIMA, SARIMA, and Holt-Winter's

## Результирующий датасет:

- **best\_model** — лучшая модель;
- **best\_hpt** — лучшие гиперпараметры

(По 20 случайных сэмплов на модель)

Модель	Гиперпараметры
Prophet	seasonality_mode ~ Categorical["additive", "multiplicative"] seasonality_prior_scale ~ LogUniform[0.01, 10] changepoint_prior_scale ~ LogUniform[0.001, 0.5] changepoint_range ~ Uniform[0.85, 0.95] holidays_prior_scale ~ LogUniform[0.01, 10]
SARIMA	p, d ~ IntUniform[1, 6] q ~ IntUniform[1, 2] P ~ IntUniform[0, 2] D, Q ~ IntUniform[0, 1] s ~ Categorical[0, 7, 30]
Holt-Winters	trend, seasonal ~ Categorical["add", "mul", "null"] damped_trend, use_boxcox ~ Categorical[False, True] seasonal_periods ~ Categorical[7, 30, null]
SMA	window ~ IntUniform[1, 10] seasonality ~ Categorical[1, 7, 30]
Linear	l1_ratio ~ Uniform[0, 1] alpha ~ LogUniform[0.00001, 1000] lags ~ IntUniform[15, 57] use_log, use_std ~ Categorical[False, True] linear_trend, date_flags ~ Categorical[False, True]

# Kats: оценка качества

- Качество работы подобранных моделей (не важны ошибки классификации)

$$SMAPE = \frac{1}{n} \times \sum \frac{|forecast value - actual value|}{(|actual value| + |forecast value|)/2}$$

Учитываем различия в масштабе.

- Качество классификации

$$BalancedAccuracy = \frac{1}{c} \left( \frac{TP_1}{TP_1+FN_1} + \dots + \frac{TP_c}{TP_c+FN_c} \right)$$

Учитываем дисбаланс классов.



# Kats (Эксперименты)

Сравним одиночные модели и оптимальную модель (SSL).

	SSL	Prophet	ARIMA	STLF	SARIMA	HW	Theta
Forecasts MAPE (train)	<b>0.0871</b>	0.1154	0.1683	0.1139	0.1253	0.1178	0.1441
MAPE Change % (train)	<b>Baseline</b>	-32.4811%	-93.1784%	-30.7371%	-43.8146%	-35.1966%	-65.4825%
Forecasts MAPE (test)	<b>0.1078</b>	0.1155	0.1620	0.1142	0.1250	0.1191	0.1437
MAPE Change % (test)	<b>Baseline</b>	-7.1096%	-50.2112%	-5.9272%	-15.9224%	-10.4991%	-33.2728%

Table 1: Performance comparison on Facebook Infrastructure data

Для получения хорошего качества достаточно 1000 рядов (на датасете Facebook\*).

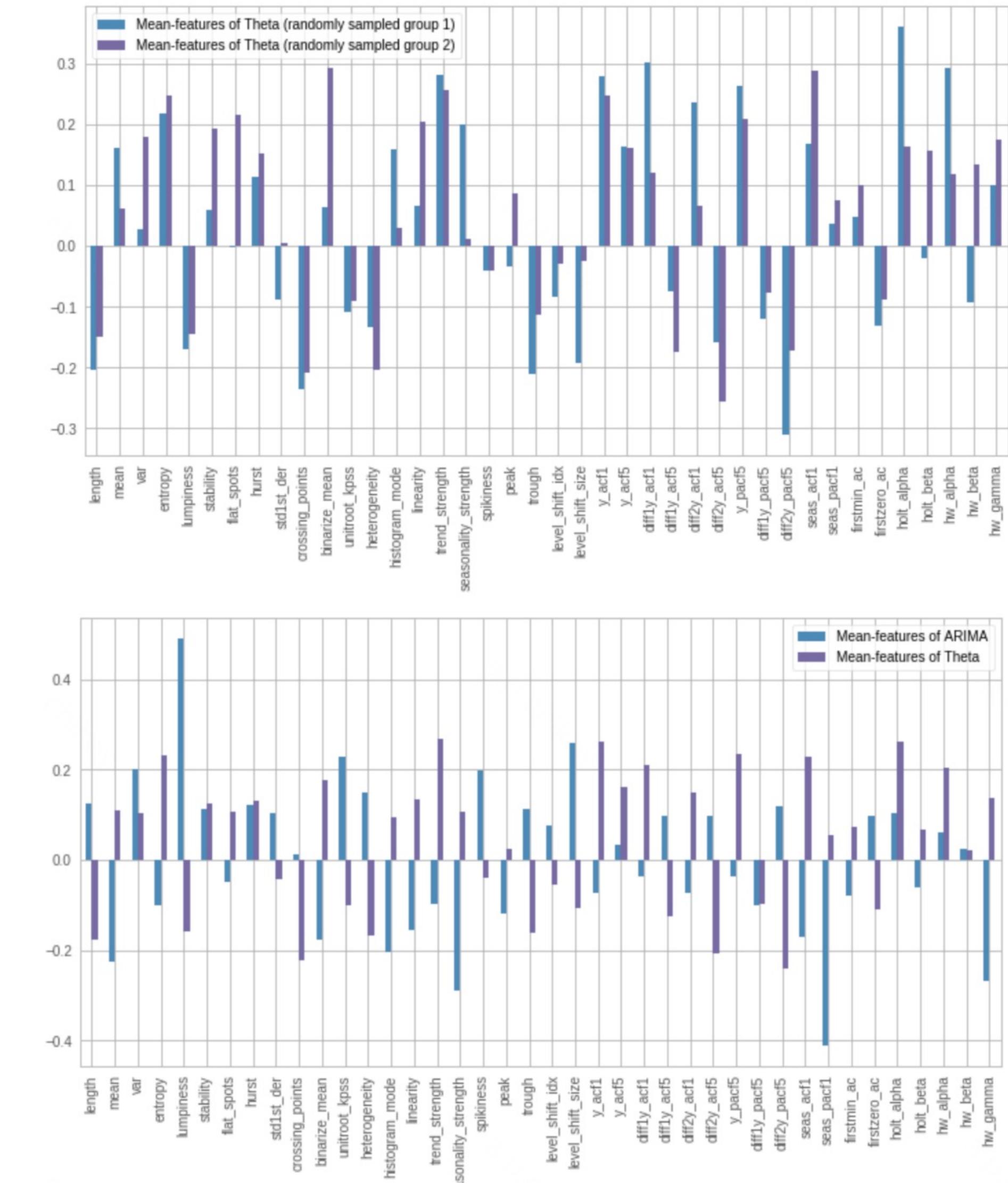
# Kats (Эксперименты)

Похожие фичи → похожие модели.

## ПРИМЕР

Для разной длины рядов нужны  
разные модели:

- короткие — простые модели;
- длинные — сложные модели  
(больше инфы для обучения).



# Kats (Эксперименты)

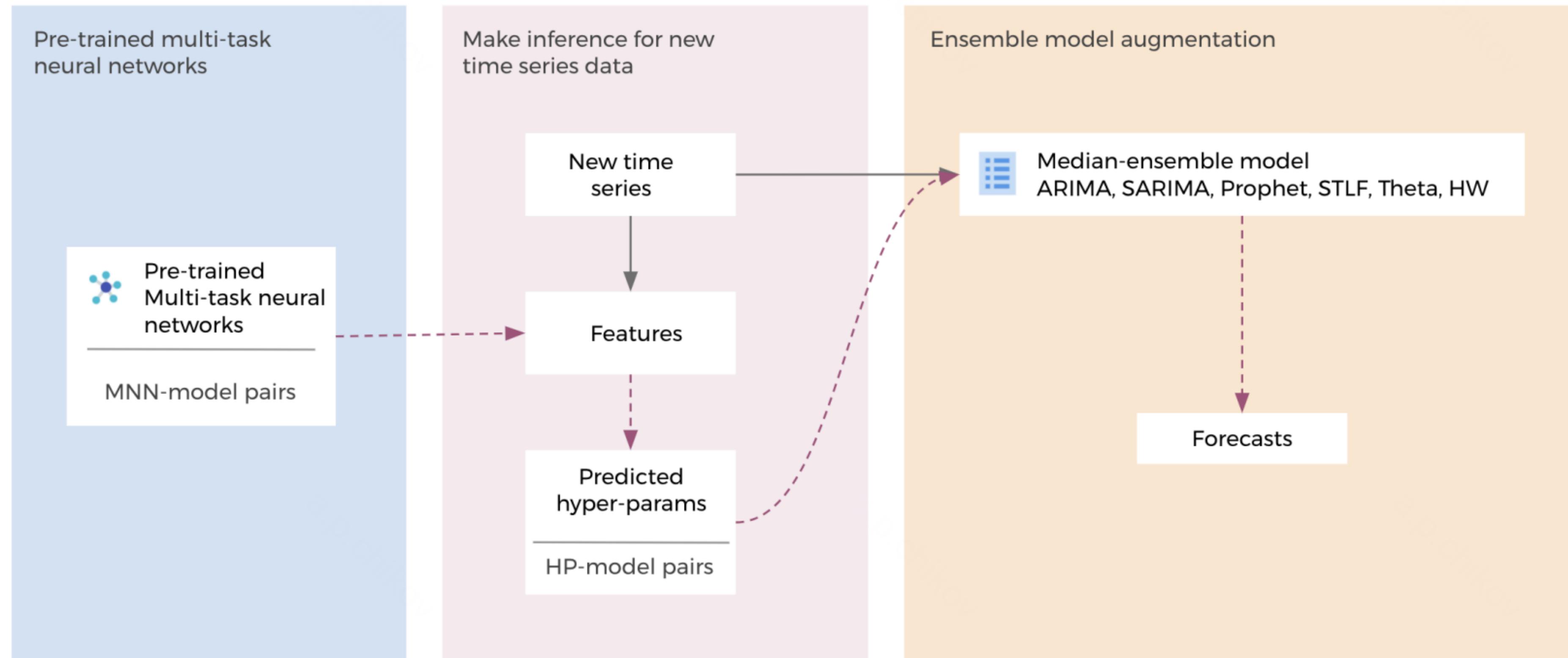
	T + y/2	T + y	T + 1.5 y	T + 2 y
T	<b>27.00%</b>	34.50%	33.50%	42.50%
T + y/2	NA	<b>22.50%</b>	27.50%	40.50%
T + y	NA	NA	<b>17.00%</b>	37.00%
T + 1.5 y	NA	NA	NA	<b>17.00%</b>

**Table 2: Consistency of model selection**

Процент рядов, для которых  
поменялась оптимальная модель

- Модель меняется со временем.
- Длинные ряды более устойчивы к модели.
- Практика: даже на коротких горизонтах есть эффект смены модели.

# Kats (Эксперименты)



Подберём гиперпараметры для ансамбля.

# Kats (Эксперименты)

		Method	Avg-MAPE	% Avg-MAPE Change	Median-MAPE	% Median-MAPE Change	# Fails	Runtime
Ensemble Model	1	Ensemble + HP	<b>0.111</b>	<b>-54.502%</b>	<b>0.096</b>	<b>-38.407%</b>	0	6*20=120
	2	Ensemble + Random-HP	0.154	-36.629%	0.130	-16.517%	0	6*1=6
	3	Ensemble + SSL-HPT	0.139	-42.917%	0.116	-25.379%	0	6*1=6
Random Model	4	Random model + HP	0.128	-47.372%	0.104	-32.976%	0	1*20=20
	5	Random model + Random-HP	0.243	<b>Baseline</b>	0.155	<b>Baseline</b>	12	1*1=1
	6	Random model + SSL-HPT	0.178	-26.889%	0.128	-17.252%	3	1*1=1
SSL-MS	7	SSL-MS + HP	<b>0.113</b>	<b>-53.639%</b>	<b>0.090</b>	<b>-42.174%</b>	0	1*20=20
	8	SSL-MS + Random-HP	0.278	14.536%	0.162	4.020%	24	1*1=1
	9	SSL-MS + SSL-HPT	0.168	-30.787%	0.124	-20.401%	5	1*1=1

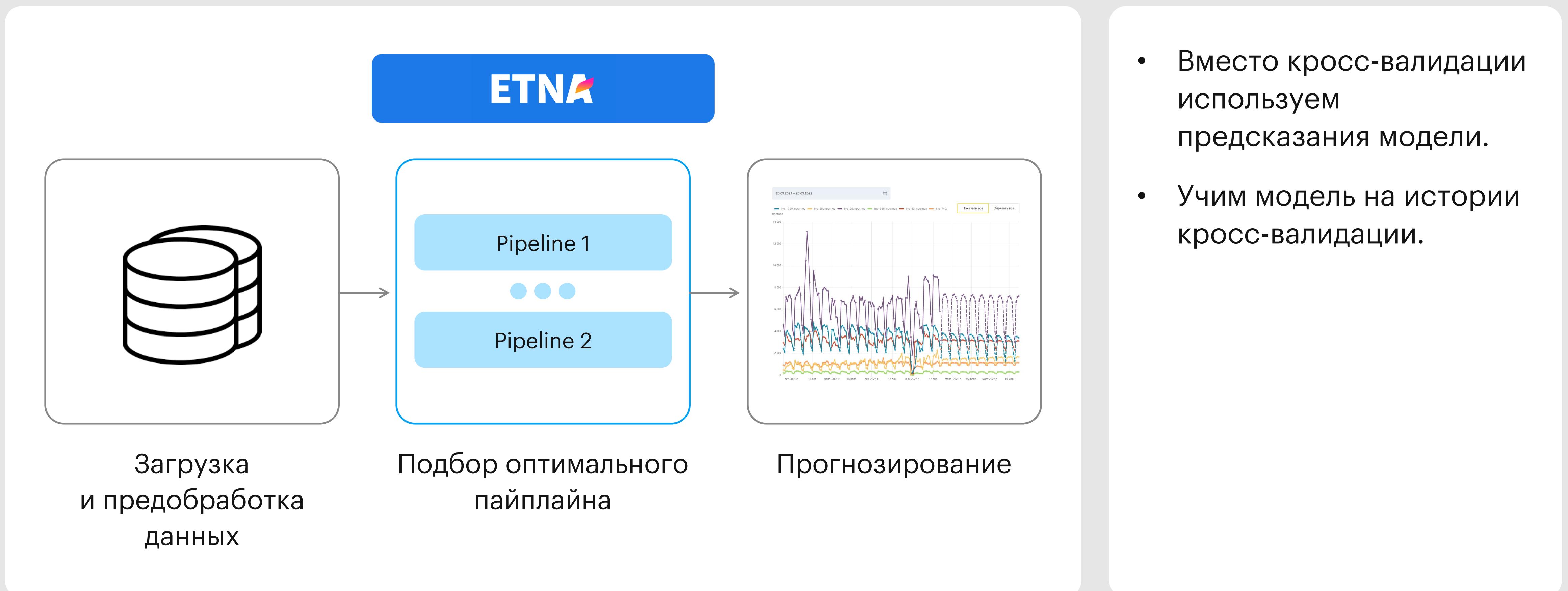
Table 3: Overall comparison on Facebook Infrastructure data

Сравним

- ансамбль;
- случайную модель;
- модель, подобранную через Meta Learning.

Meta Learning + ансамблирование  
выигрывают  
(вот это поворот).

# Где можно применить



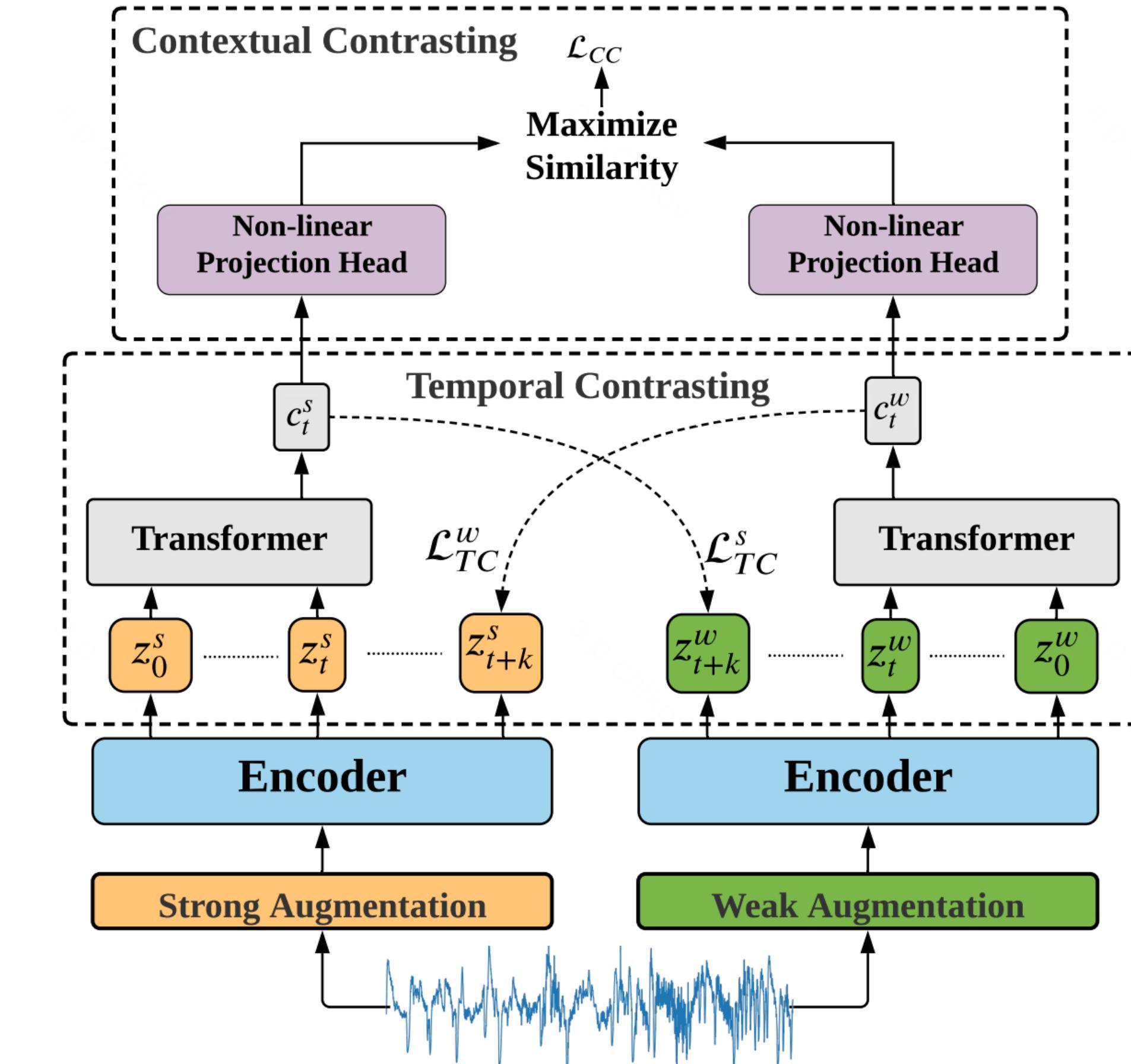
# DL-подходы

# DL-подходы. TS-TCC

# TS-TCC



- Weak and Strong Augmentations.
- Temporal Contrasting: сближаем одинаковые таймстемпы в рамках ряда.
- Contextual Contrasting: сближаем одинаковые ряды.

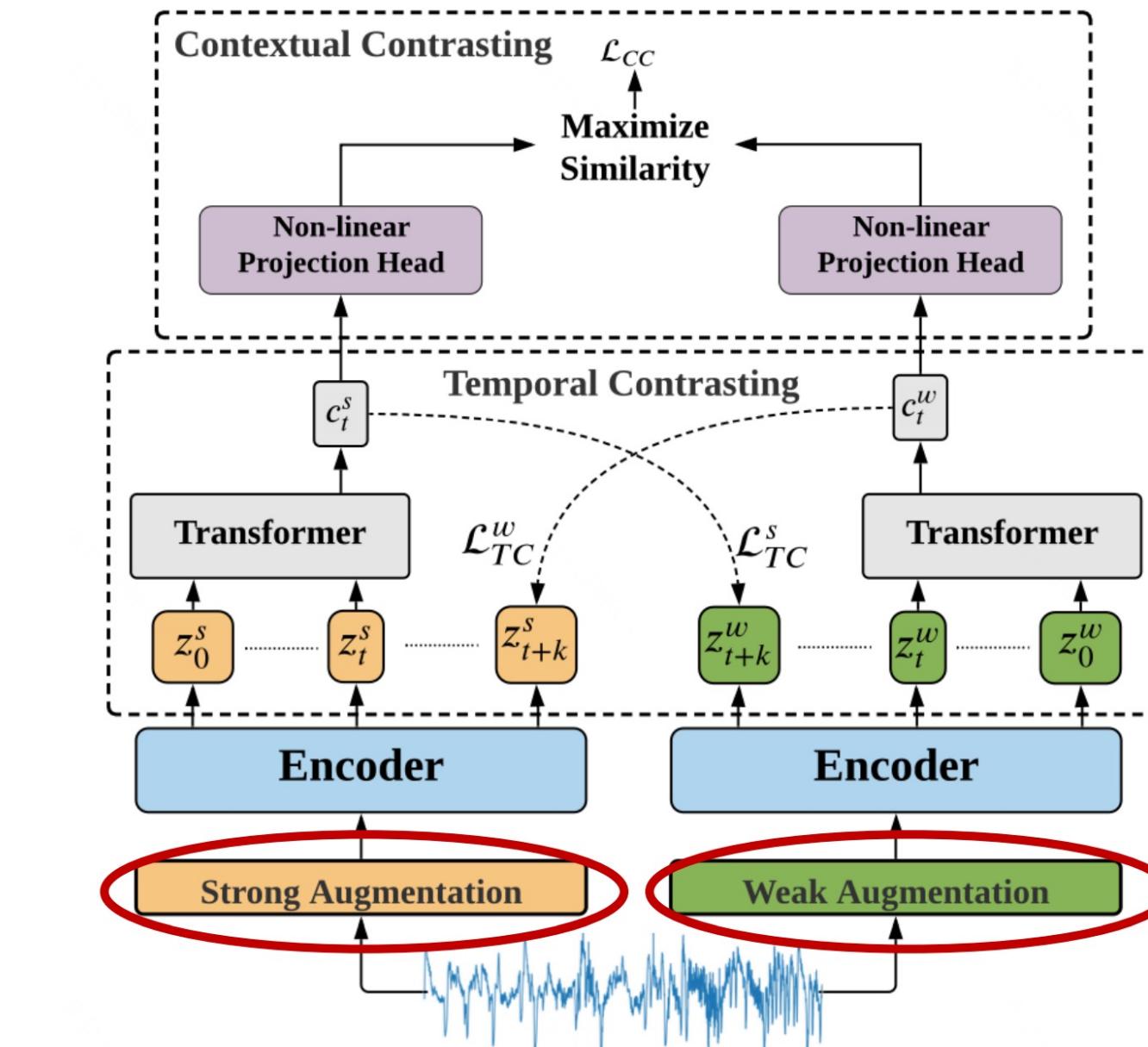


# TS-TCC (Augmentation)

**Идея:** разные аугментации дают более  
робастное представление.

- **Weak** —  $x * N(2, \sigma_1)$  (jitter-and-scale).
- **Strong**
  - Разделим на M подрядов.
  - Перемешаем куски.
  - Для каждого куска  $x+N(0, \sigma_2)$ .

jitter\_scale\_ratio: float = 1.1,  
max\_seg: int = 4,  
jitter\_ratio: float = 0.8,



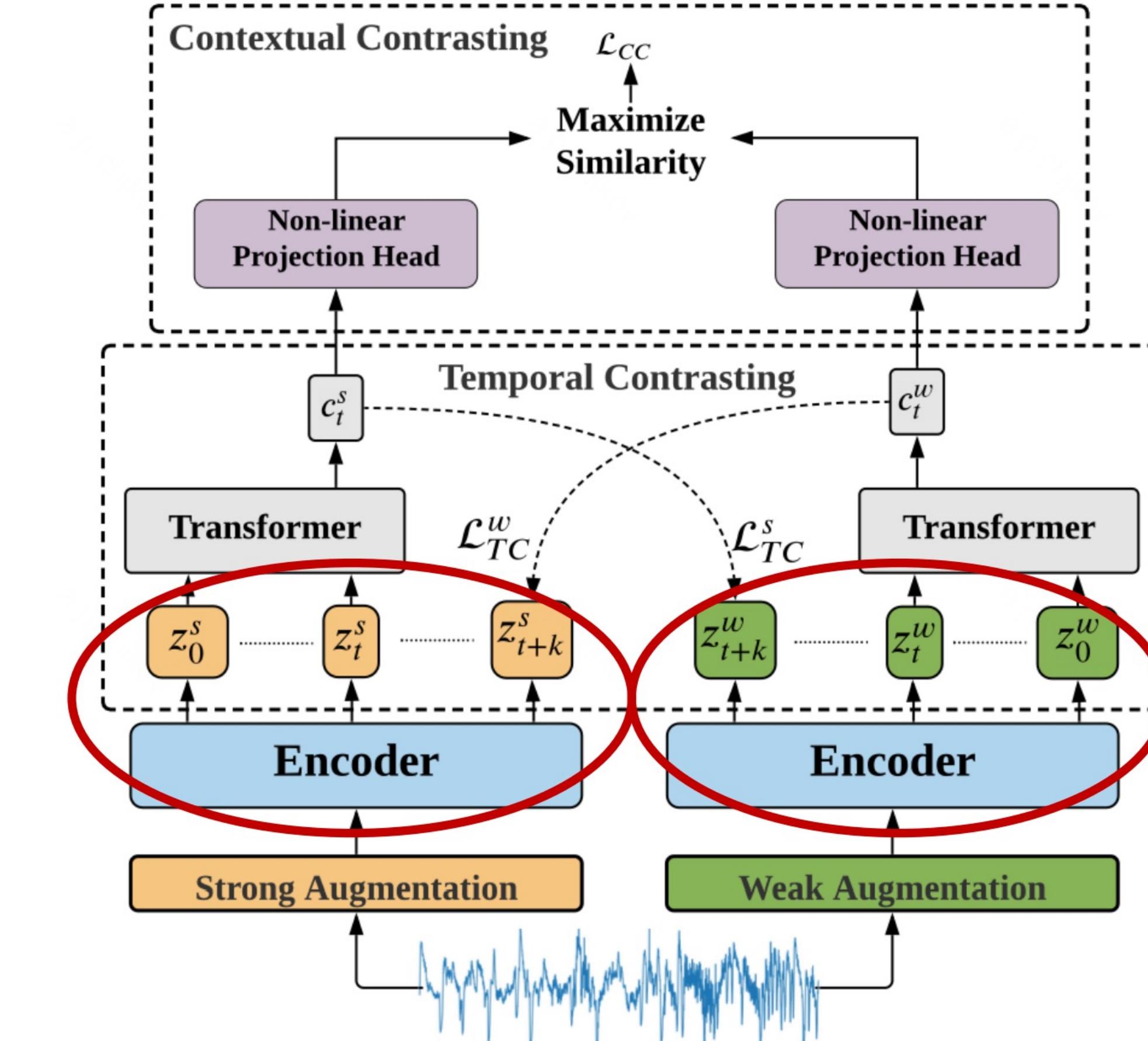
- Нужно подбирать под данные:
- max\_seg — чем длиннее, тем больше;
  - jitter — чем больше масштаб, тем больше.

# TS-TCC (Encoder)

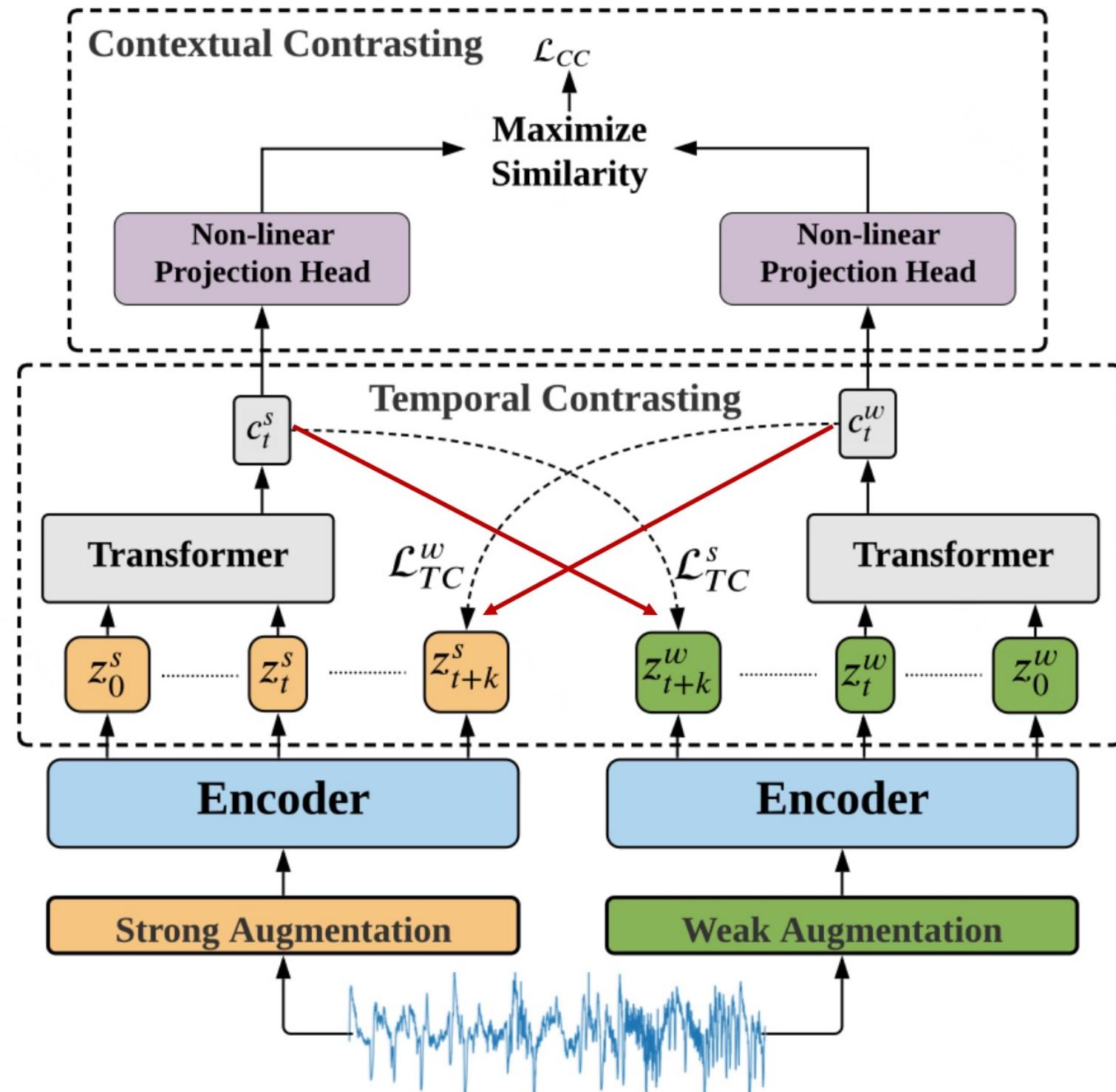
- Encoder — 3 свёрточных слоя (не меняет размерность по времени).
- Transformer — авторегрессионно получает контекстный вектор  $c_t = f(z_{<t})$ .

output\_dims: int = 32  
 kernel\_size: int = 7,  
 dropout: float = 0.35,

tc\_hidden\_dim: int = 32,  
 heads: int = 1,  
 depth: int = 4,  
 n\_seq\_steps: int = 0



# TS-TCC (Архитектура)



Предсказываем будущее  $Z[t:t+K]$  по вектору контекста  $C[0:t]$  для другой аугментации.

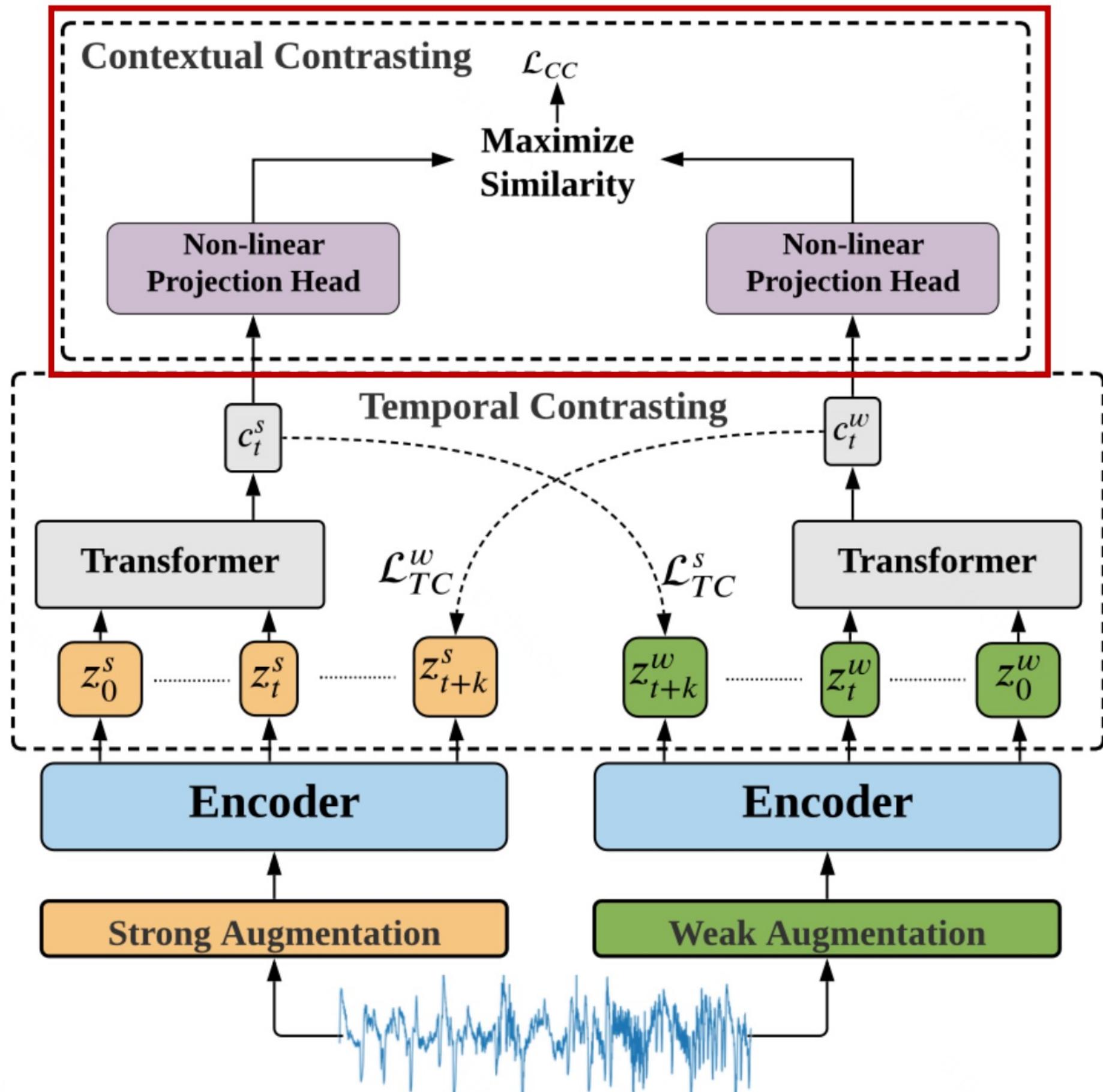
$$\text{Temporal Contrasting } s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

$$\mathcal{L}_{TC}^s = -\frac{1}{K} \sum_{k=1}^K \log \frac{\exp((\mathcal{W}_k(c_t^s))^T z_{t+k}^w)}{\sum_{n \in \mathcal{N}_{t,k}} \exp((\mathcal{W}_k(c_t^s))^T z_n^w)}$$

$$\mathcal{L}_{TC}^w = -\frac{1}{K} \sum_{k=1}^K \log \frac{\exp((\mathcal{W}_k(c_t^w))^T z_{t+k}^s)}{\sum_{n \in \mathcal{N}_{t,k}} \exp((\mathcal{W}_k(c_t^w))^T z_n^s)}$$

**timesteps: int = 7**

# TS-TCC (Архитектура)



Сближаем представления  
для соответствующих векторов  
контекста ( $c_t^w$  и  $c_t^s$  — позитивы,  
остальные — негативы).

$$\mathcal{L}_{CC} = - \sum_{i=1}^N \log \frac{\exp \left( \text{sim} \left( \mathbf{c}_t^i, \mathbf{c}_t^{i+} \right) / \tau \right)}{\sum_{m=1}^{2N} \mathbb{1}_{[m \neq i]} \exp \left( \text{sim} \left( \mathbf{c}_t^i, \mathbf{c}_t^m \right) / \tau \right)}$$

Смешиваем лосы:

$$\mathcal{L} = \lambda_1 \cdot (\mathcal{L}_{TC}^s + \mathcal{L}_{TC}^w) + \lambda_2 \cdot \mathcal{L}_{CC},$$

# TS-TCC (Эксперименты)

## Линейный слой поверх предобученного энкодера

	HAR		Sleep-EDF		Epilepsy	
Baseline	ACC	MF1	ACC	MF1	ACC	MF1
Random Initialization	57.89±5.13	55.45±5.49	35.61±6.96	23.80±7.96	90.26±1.77	81.12±4.22
Supervised	90.14±2.49	90.31±2.24	<b>83.41±1.44</b>	<b>74.78±0.86</b>	96.66±0.24	94.52±0.43
SSL-ECG [P. Sarkar, 2020]	65.34±1.63	63.75±1.37	74.58±0.60	65.44±0.97	93.72±0.45	89.15±0.93
CPC [Oord <i>et al.</i> , 2018]	83.85±1.51	83.27±1.66	82.82±1.68	73.94±1.75	96.61±0.43	94.44±0.69
SimCLR [Chen <i>et al.</i> , 2020]	80.97±2.46	80.19±2.64	78.91±3.11	68.60±2.71	96.05±0.34	93.53±0.63
TS-TCC ( <i>ours</i> )	<b>90.37±0.34</b>	<b>90.38±0.39</b>	83.00±0.71	73.57±0.74	<b>97.23±0.10</b>	<b>95.54±0.08</b>

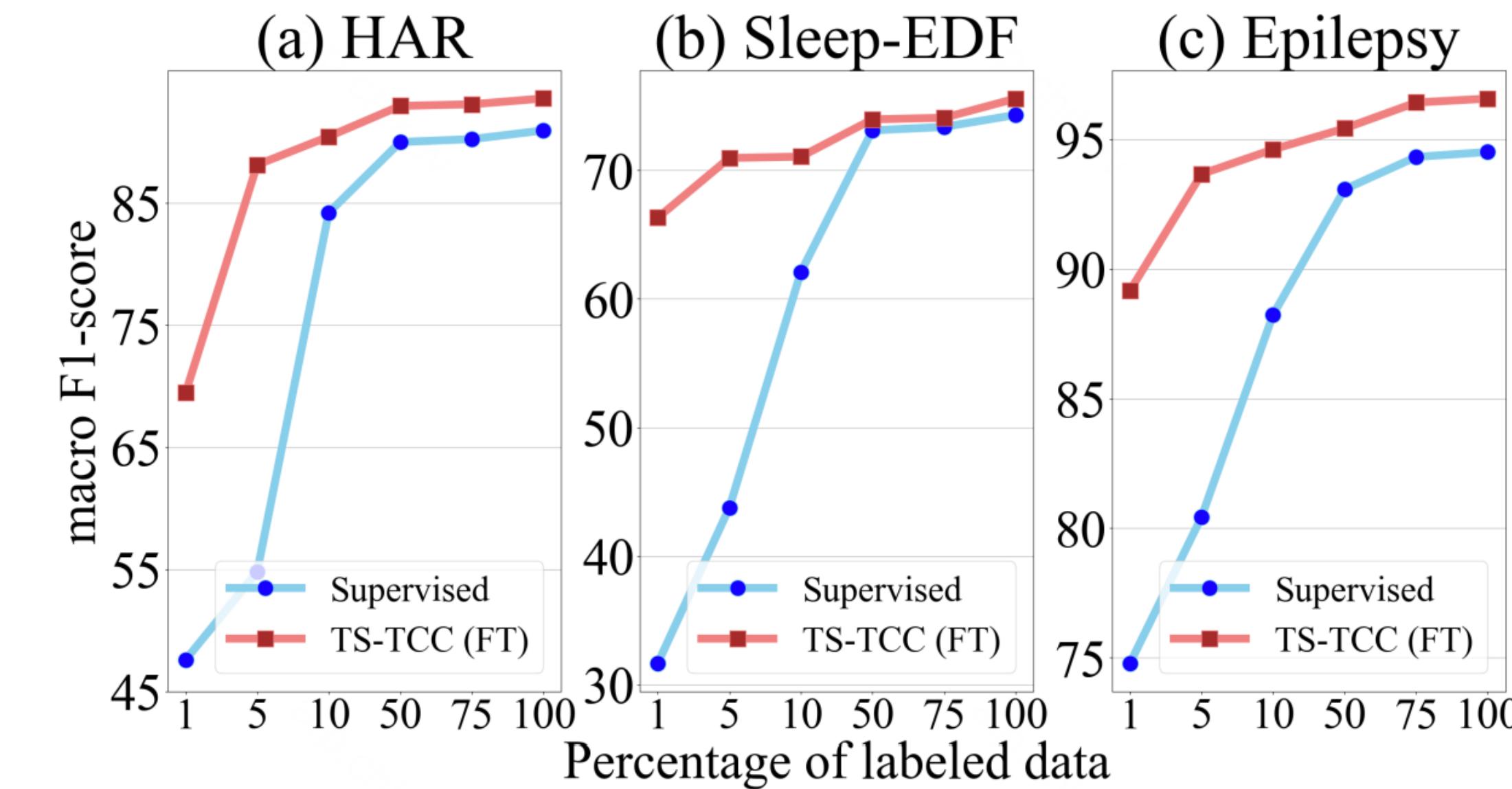
### Задача классификации

- Human activity recognition (HAR)
- Sleep stage classification
- Epileptic seizure prediction

# TS-TCC (Эксперименты)

## Semi-supervised

- Supervised
- Self-supervised + Fine Tuning



## Transfer learning (Zero Shot)

- Supervised
- Fine-tuning

	A→B	A→C	A→D	B→A	B→C	B→D	C→A	C→B	C→D	D→A	D→B	D→C	AVG
Supervised	34.38	44.94	34.57	<b>52.93</b>	63.67	<b>99.82</b>	<b>52.93</b>	84.02	83.54	<b>53.15</b>	99.56	62.43	63.83
TS-TCC (FT)	<b>43.15</b>	<b>51.50</b>	<b>42.74</b>	47.98	<b>70.38</b>	99.30	38.89	<b>98.31</b>	<b>99.38</b>	51.91	<b>99.96</b>	<b>70.31</b>	<b>67.82</b>

# TS-TCC (Ablation)

Component	HAR		Sleep-EDF		Epilepsy	
	ACC	MF1	ACC	MF1	ACC	MF1
TC only	82.76±1.50	82.17±1.64	80.55±0.39	70.99±0.86	94.39±1.19	90.93±1.41
TC + X-Aug	87.86±1.33	87.91±1.09	81.58±1.70	71.88±1.71	95.56±0.24	92.57±0.29
TS-TCC (TC + X-Aug + CC)	<b>90.37±0.34</b>	<b>90.38±0.39</b>	<b>83.00±0.71</b>	<b>73.57±0.74</b>	<b>97.23±0.10</b>	<b>95.54±0.08</b>
TS-TCC (Weak only)	76.55±3.59	75.14±4.66	80.90±1.87	72.51±1.74	97.18±0.17	95.47±0.31
TS-TCC (Strong only)	60.23±3.31	56.15±4.14	78.55±2.94	68.05±1.87	97.14±0.23	95.39±0.29

TC only — предсказываем своё будущее в ТС.

TC + X-Aug — всё без СС.

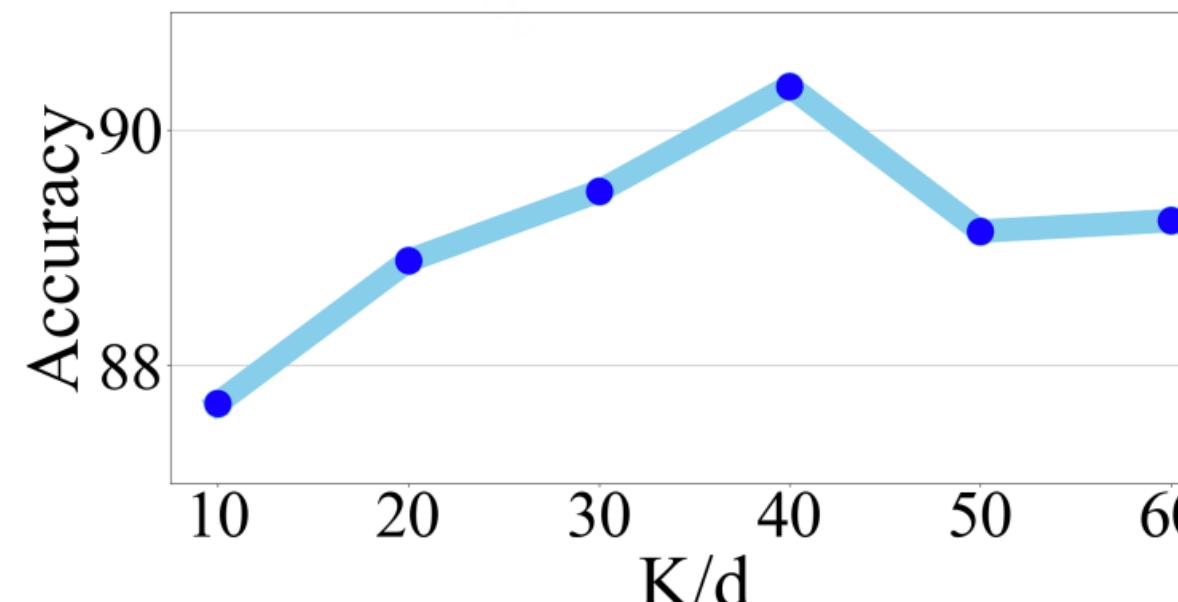
- Week + Strong — это важно.
- ТС — это важно.
- СС — это важно.

# TS-TCC (Гиперпараметры)

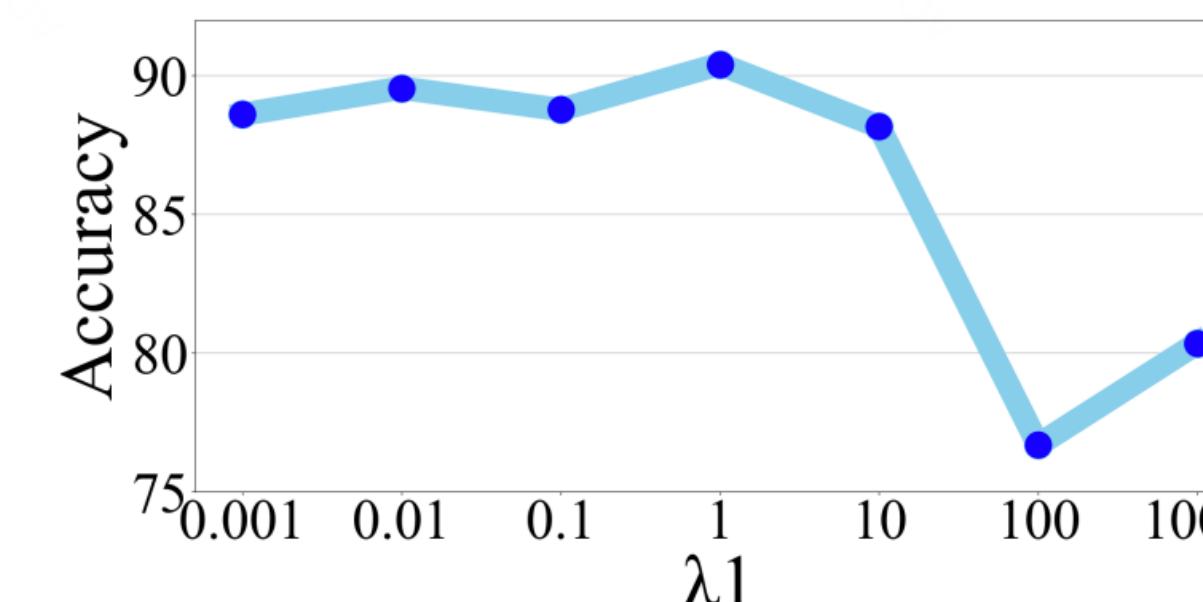
→  $K$  — горизонт прогноза в Temporal Contrasting.

Чем больше  $K$ , тем лучше, оптимально  $K = 40\% \text{len(ts)}$ .

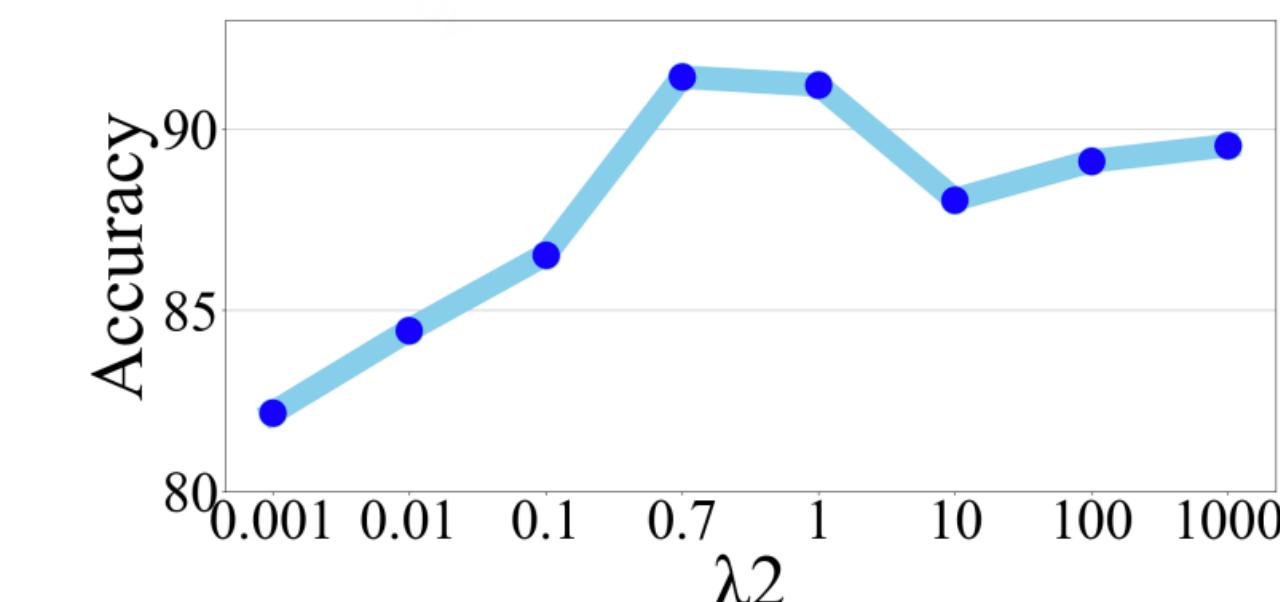
→  $\lambda_1, \lambda_2$  — веса лосов.  
 $\lambda_1$  and  $\lambda_2 = 1$  и  $0.7$ .



(a)



(b)



(c)



# TS-TCC (Гиперпараметры)

## Энкодер

output\_dims: int = 32,  
kernel\_size: int = 7,  
dropout: float = 0.35,

## Аугментации

jitter\_scale\_ratio: float = 1.1,  
max\_seg: int = 4,  
jitter\_ratio: float = 0.8,

## Трансформер

tc\_hidden\_dim: int = 32,  
n\_seq\_steps: int = 0  
heads: int = 1,  
depth: int = 4,

## Лосы

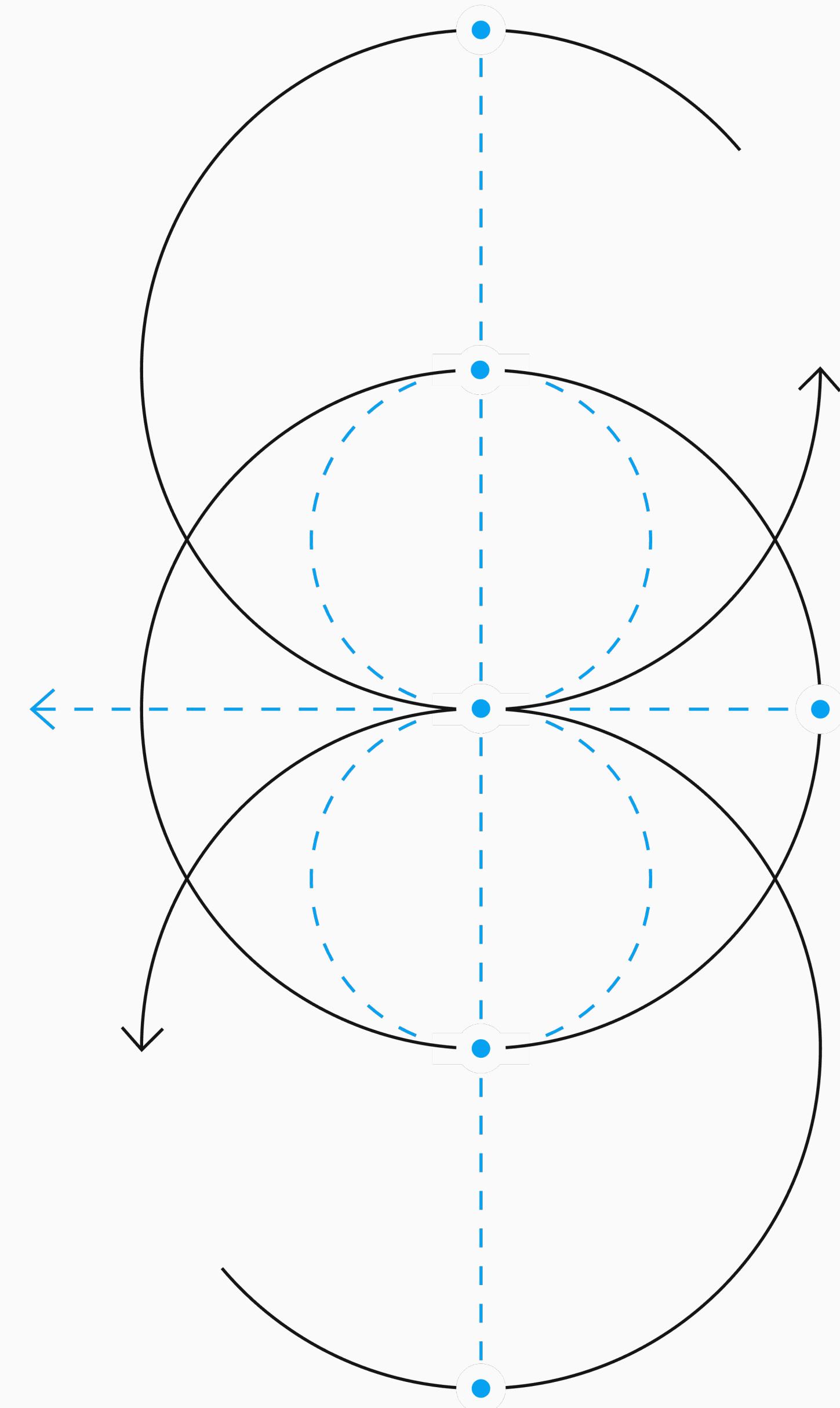
timesteps: int = 7,  
temperature: float = 0.2,  
lambda1: float = 1,  
lambda2: float = 0.7,

# DL-подходы. TS2Vec

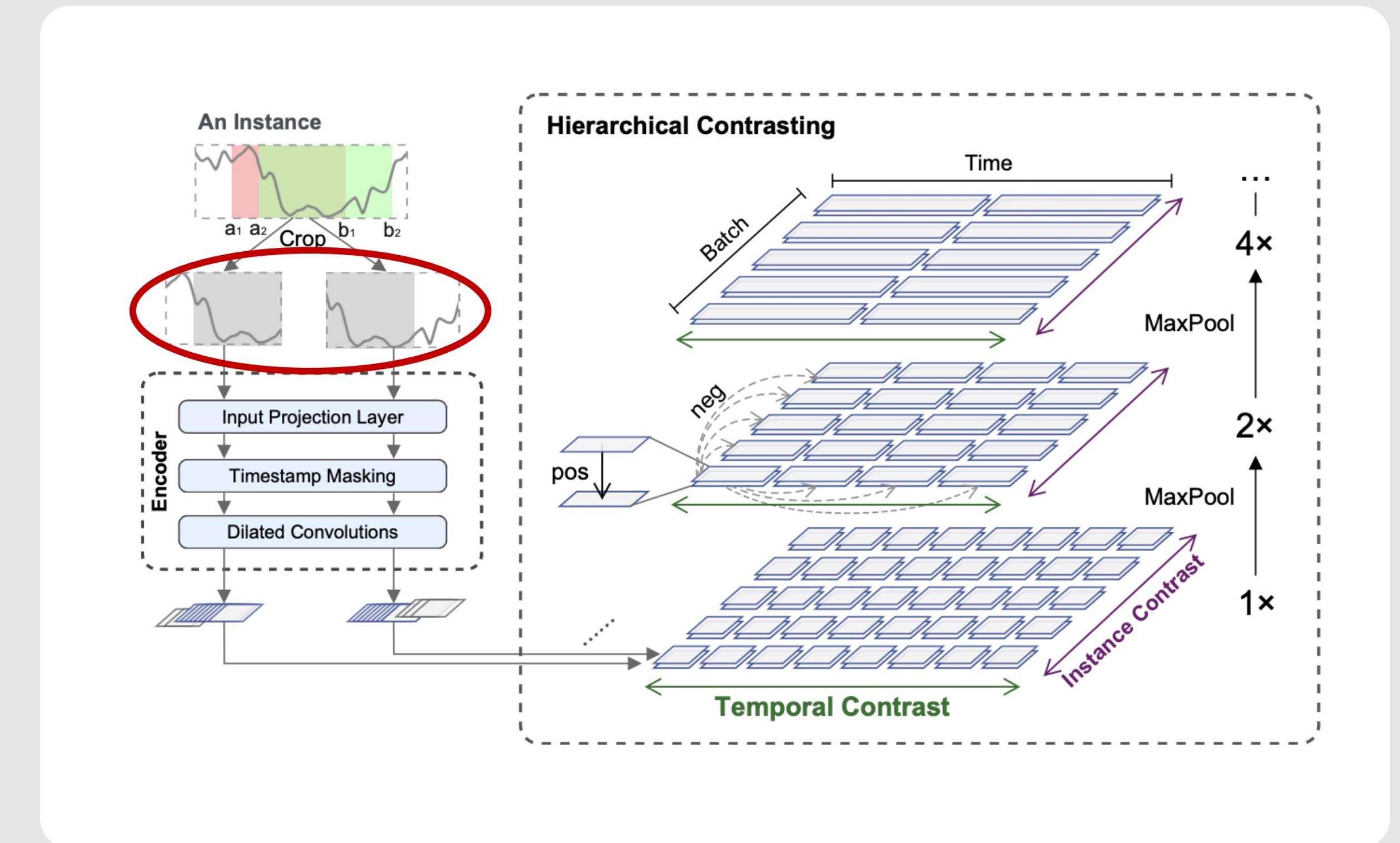
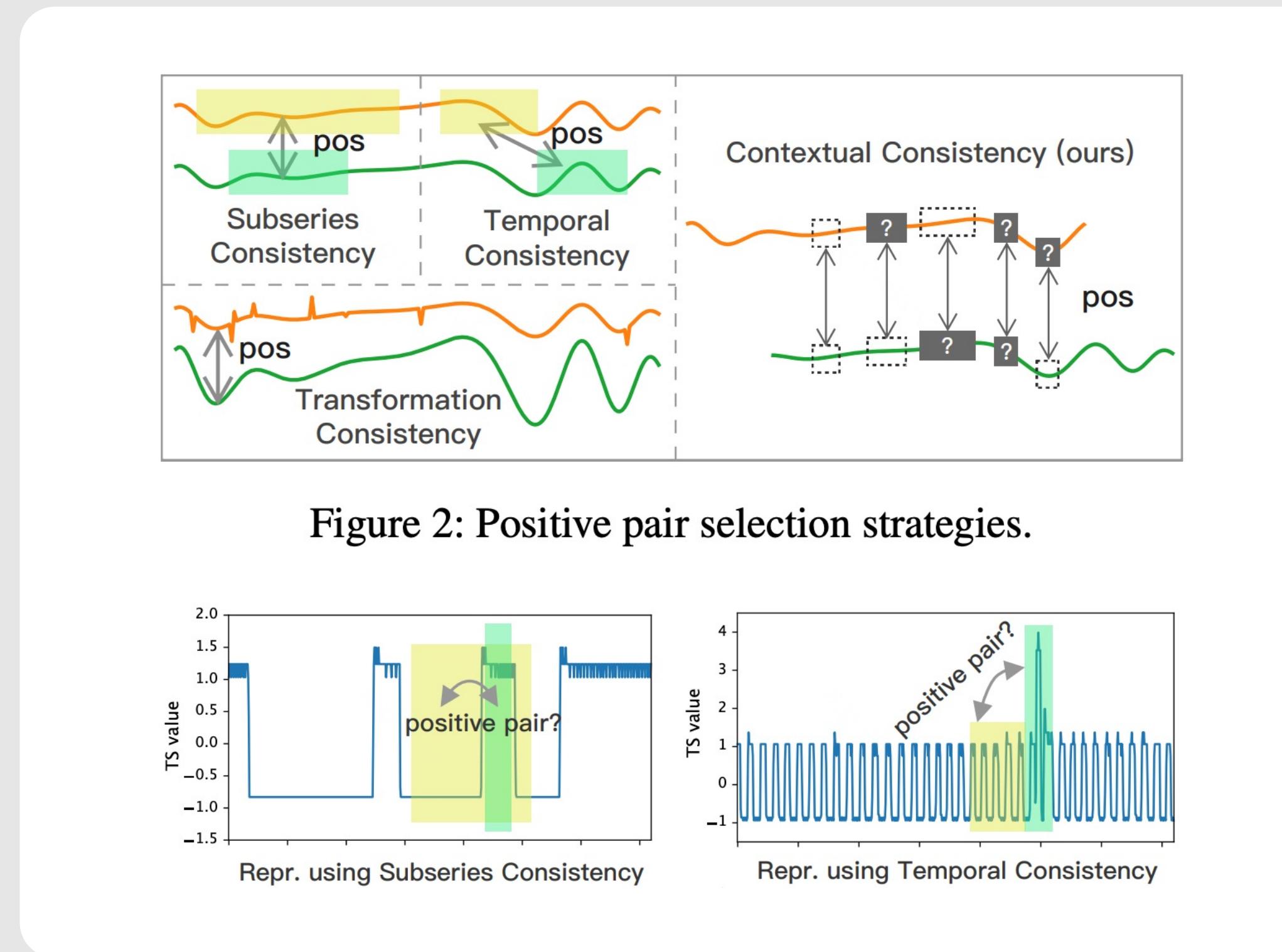
# TS2Vec



- Contrastive Learning
  - Hierarchical Contrasting (temporal + contextual) позволяет ловить паттерны на разных частотах.
  - Contextual Consistency — более правильные позитивы в контексте временных рядов.
- Переход от поточечных эмбеддингов к посегментным через агрегацию.
- Sota:
  - Classification,
  - Forecasting,
  - Anomaly Detection.



# TS2Vec (Contextual Consistency)



**Позитивы** = тот же самый ряд, но **маскированный**.

# TS2Vec (Contextual Consistency)

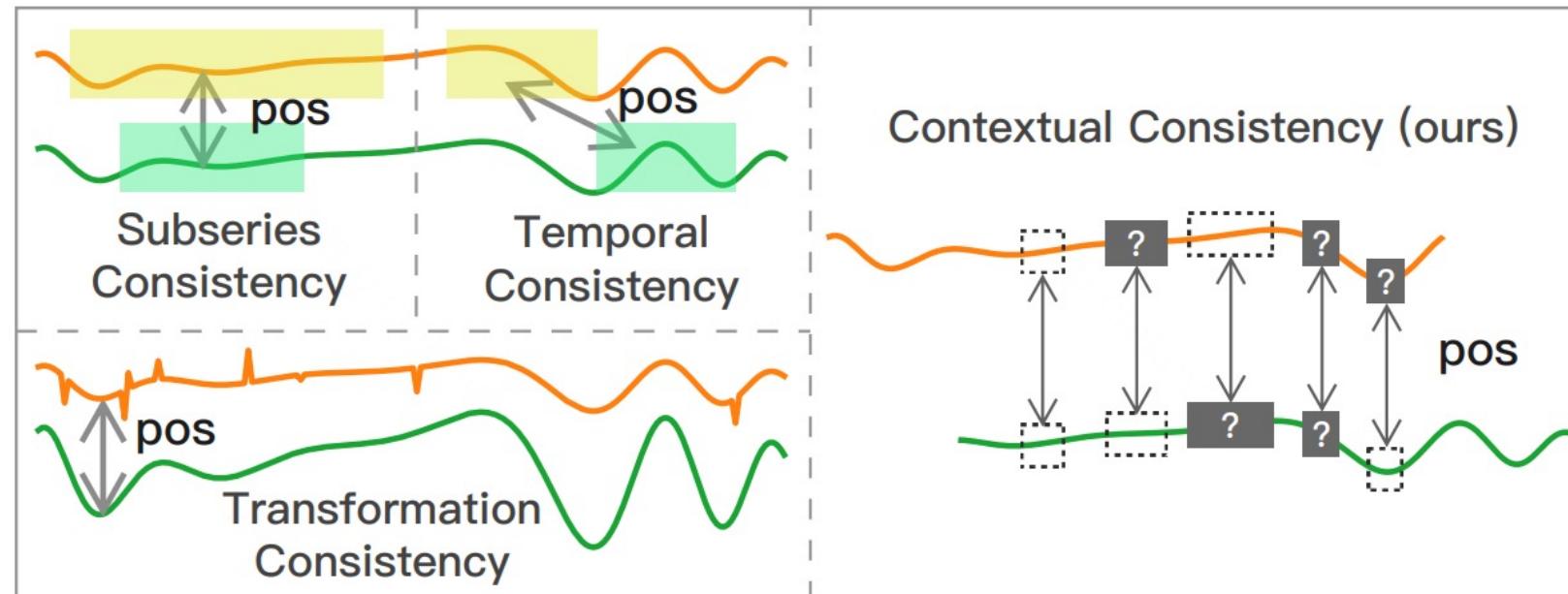
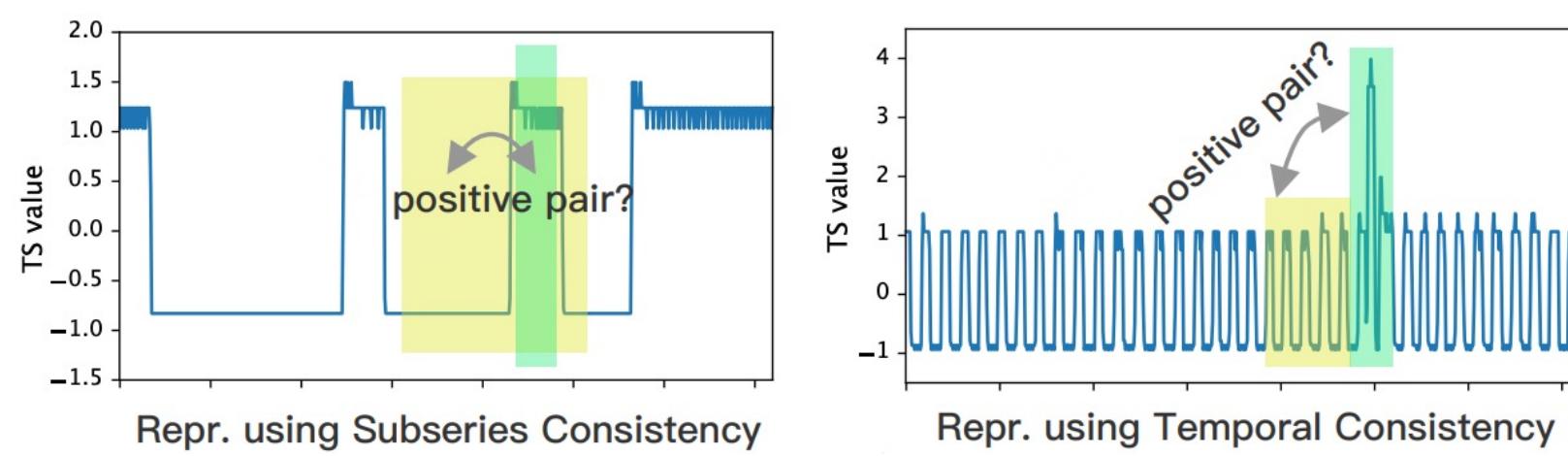


Figure 2: Positive pair selection strategies.

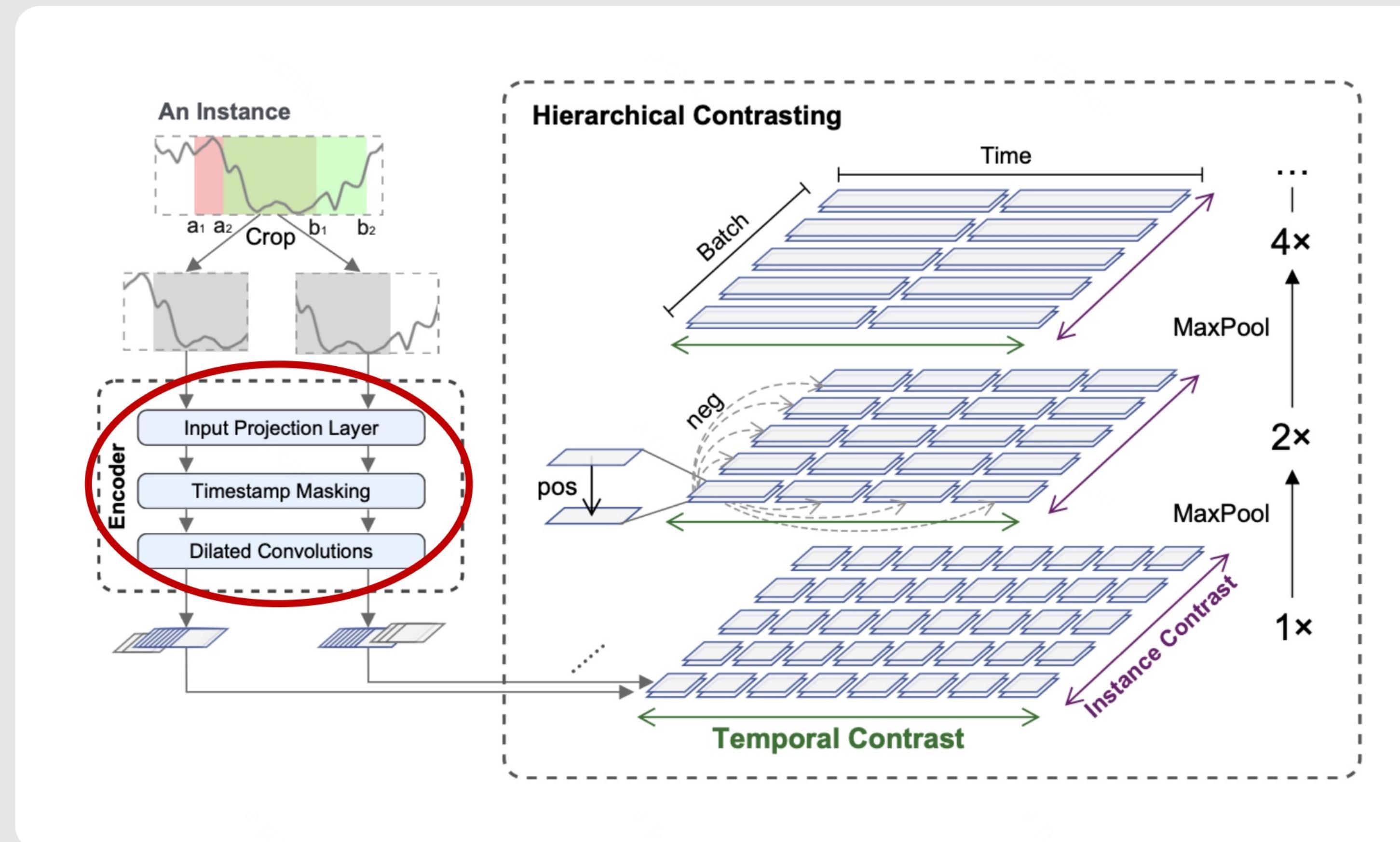


## Альтернативы

1. Subseries — свой подряд.
2. Transformation — деформация самого себя.
3. Temporal — соседние подряды.

**Проблема:** часто во временных рядах такие пары **не позитивы**.

# TS2Vec (Энкодер)



- Маскируем спроектированные элементы.
- Dilated Convolutions, чтобы был большой receptive field.

# TS2Vec (Hierarchical Contrasting)

Algorithm 1: Calculating the hierarchical contrastive loss

---

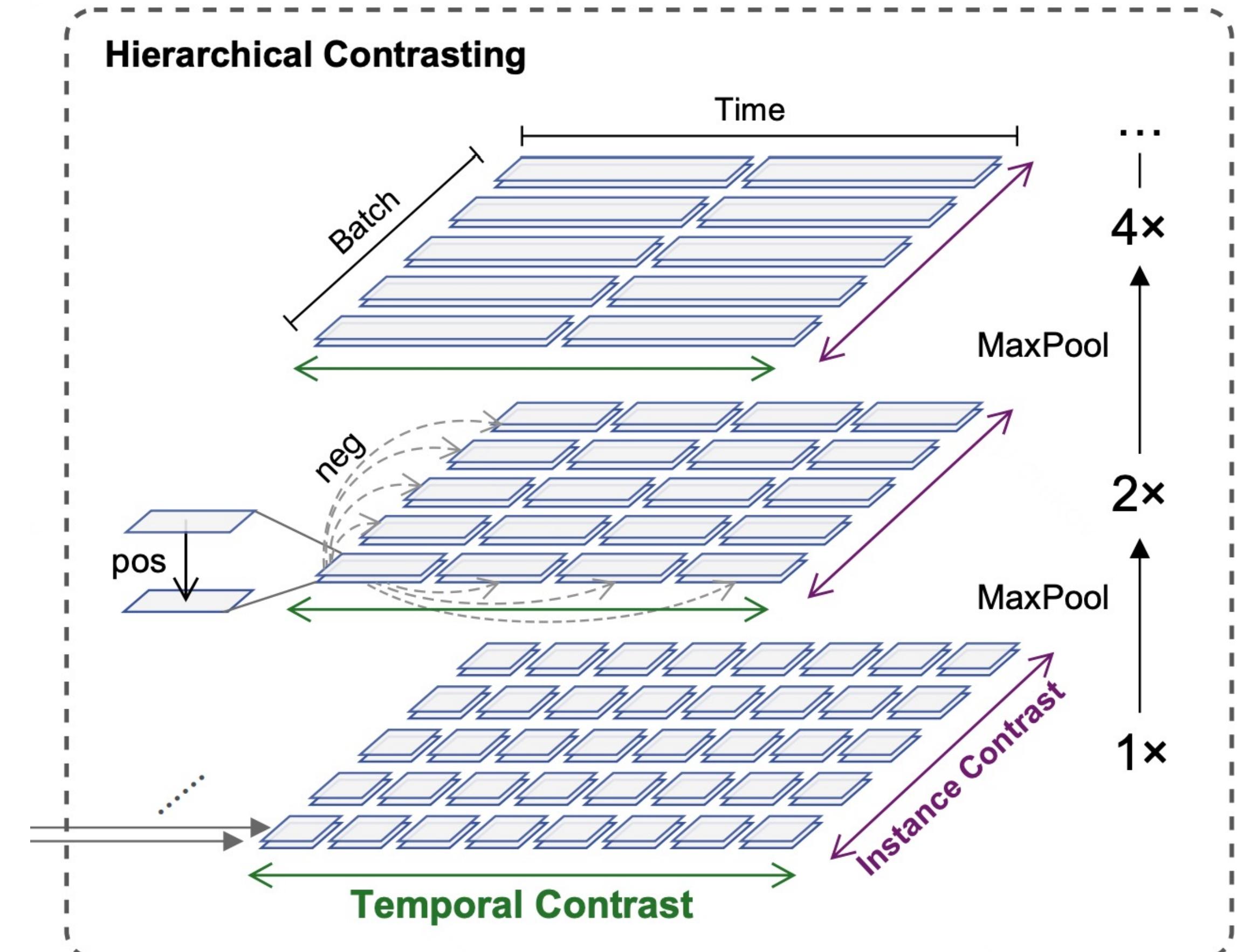
```

1: procedure HIERLOSS( $r, r'$ )
2:    $\mathcal{L}_{hier} \leftarrow \mathcal{L}_{dual}(r, r');$ 
3:    $d \leftarrow 1;$ 
4:   while time_length( $r$ ) > 1 do
5:     // The maxpool1d operates along the time axis.
6:      $r \leftarrow \text{maxpool1d}(r, \text{kernel\_size} = 2);$ 
7:      $r' \leftarrow \text{maxpool1d}(r', \text{kernel\_size} = 2);$ 
8:      $\mathcal{L}_{hier} \leftarrow \mathcal{L}_{hier} + \mathcal{L}_{dual}(r, r');$ 
9:      $d \leftarrow d + 1;$ 
10:    end while
11:     $\mathcal{L}_{hier} \leftarrow \mathcal{L}_{hier}/d;$ 
12:    return  $\mathcal{L}_{hier}$ 
13: end procedure

```

---

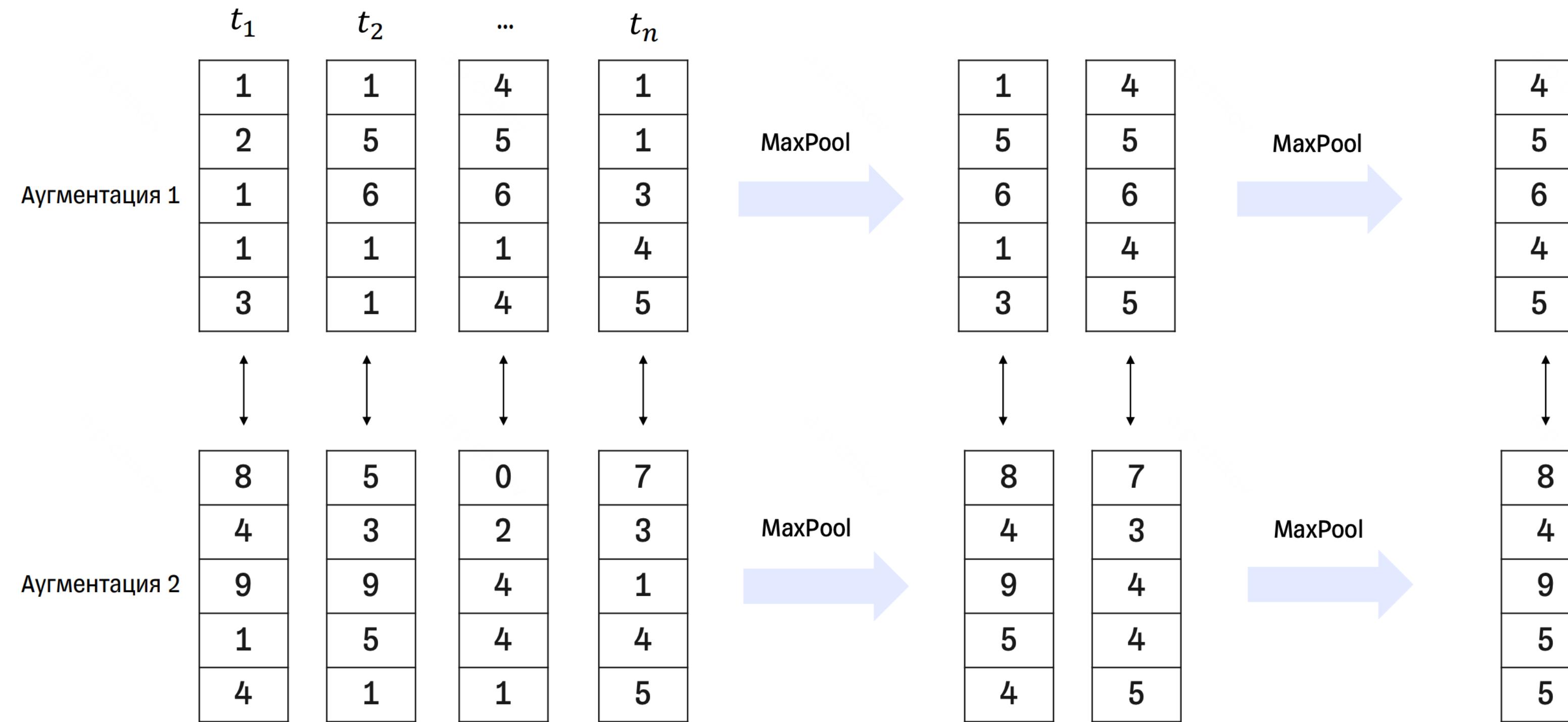
$$\mathcal{L}_{dual} = \frac{1}{NT} \sum_i \sum_t \left( \ell_{temp}^{(i,t)} + \ell_{inst}^{(i,t)} \right)$$



# TS2Vec (Hierarchical Contrasting)

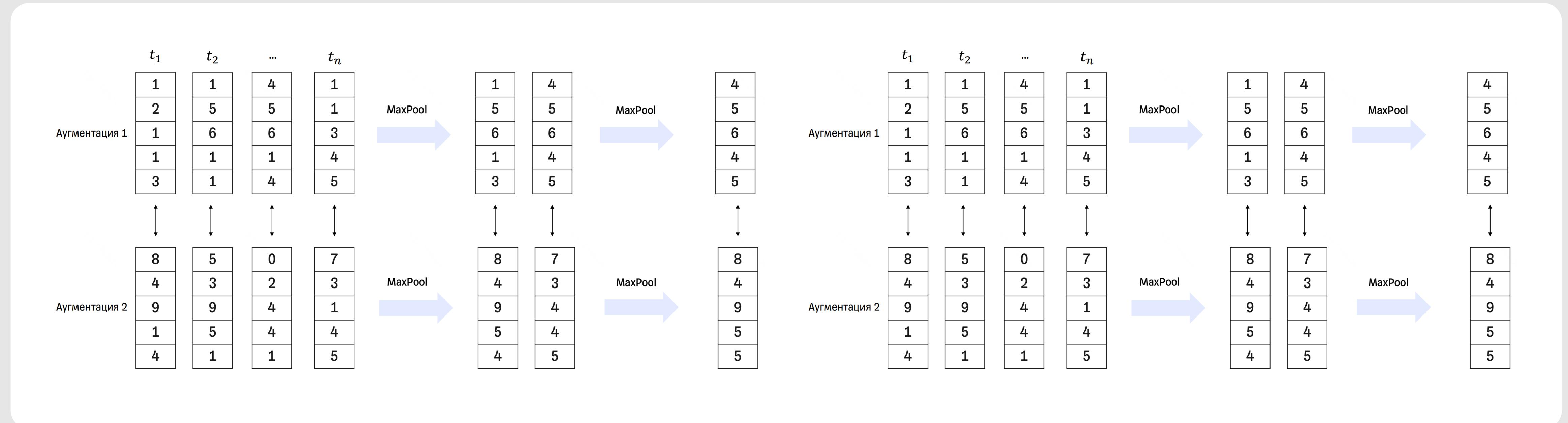


Temporal Contrasting: сближаем представления для одинаковых таймстемпов.

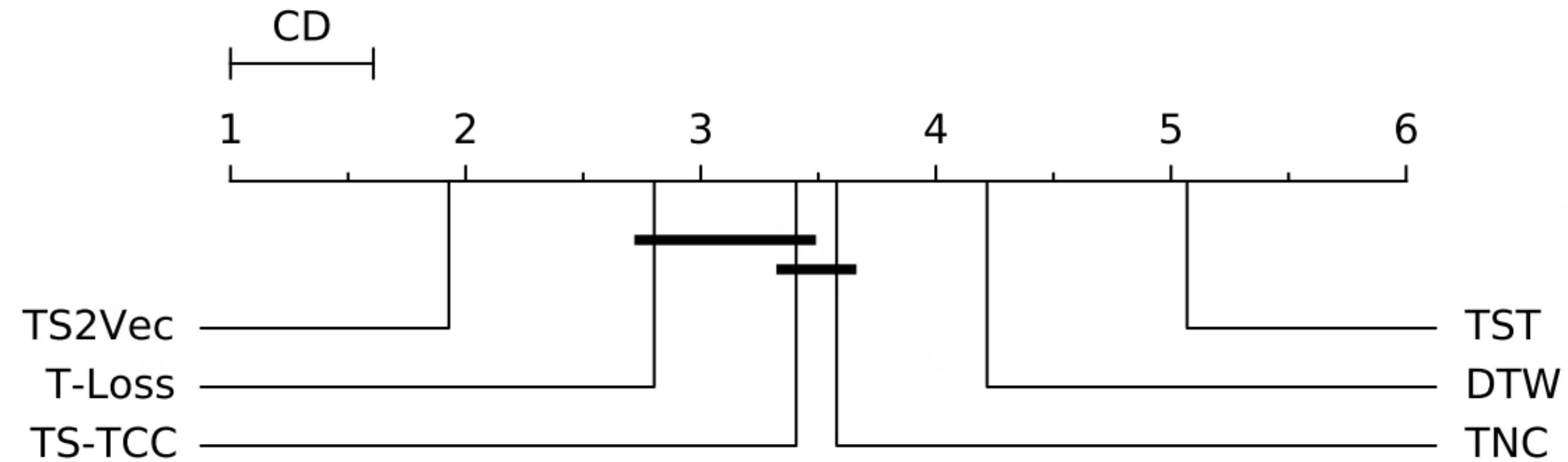


# TS2Vec (Hierarchical Contrasting)

**Instance Contrasting:** сближаем представления для одинаковых рядов.



# TS2Vec (Эксперименты)



Учим линейный классификатор поверх unsupervised-методов.

## Классификация

125 UCR datasets			
Method	Avg. Acc.	Avg. Rank	Training Time (hours)
DTW	0.727	4.33	—
TNC	0.761	3.52	228.4
TST	0.641	5.23	17.1
TS-TCC	0.757	3.38	1.1
T-Loss	0.806	2.73	38.0
TS2Vec	<b>0.830 (+2.4%)</b>	<b>1.82</b>	<b>0.9</b>

Быстро обучается на GPU.

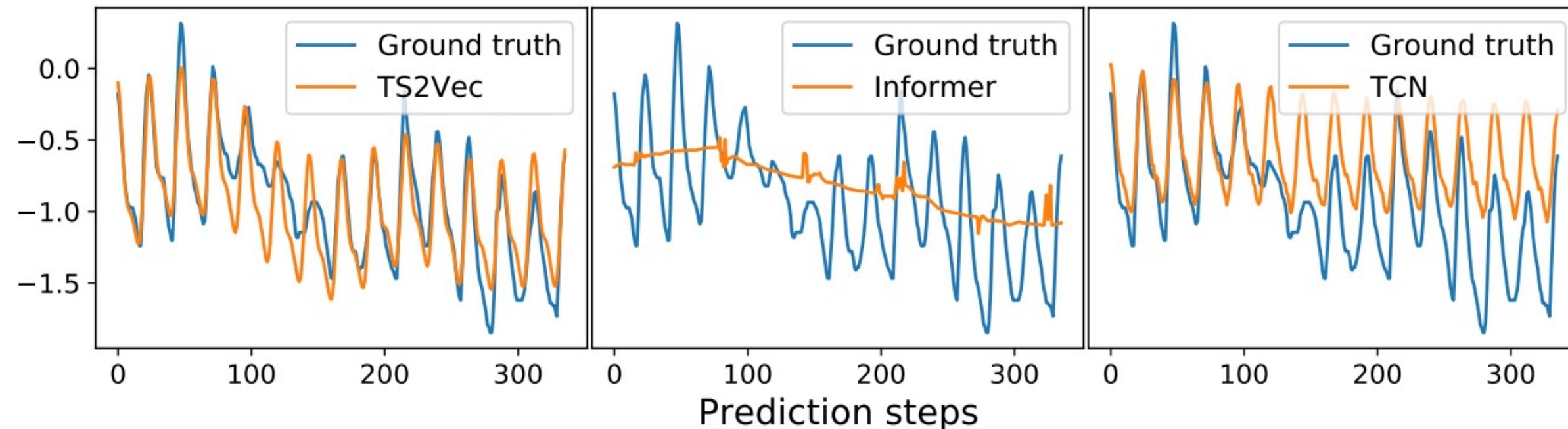
# TS2Vec (Эксперименты)

Dataset	H	TS2Vec	Informer	LogTrans	N-BEATS	TCN	LSTnet
ETTh <sub>1</sub>	24	<b>0.039</b>	0.098	0.103	0.094	0.075	0.108
	48	<b>0.062</b>	0.158	0.167	0.210	0.227	0.175
	168	<b>0.134</b>	0.183	0.207	0.232	0.316	0.396
	336	<b>0.154</b>	0.222	0.230	0.232	0.306	0.468
	720	<b>0.163</b>	0.269	0.273	0.322	0.390	0.659
ETTh <sub>2</sub>	24	<b>0.090</b>	0.093	0.102	0.198	0.103	3.554
	48	<b>0.124</b>	0.155	0.169	0.234	0.142	3.190
	168	<b>0.208</b>	0.232	0.246	0.331	0.227	2.800
	336	<b>0.213</b>	0.263	0.267	0.431	0.296	2.753
	720	<b>0.214</b>	0.277	0.303	0.437	0.325	2.878
ETTm <sub>1</sub>	24	<b>0.015</b>	0.030	0.065	0.054	0.041	0.090
	48	<b>0.027</b>	0.069	0.078	0.190	0.101	0.179
	96	<b>0.044</b>	0.194	0.199	0.183	0.142	0.272
	288	<b>0.103</b>	0.401	0.411	0.186	0.318	0.462
	672	<b>0.156</b>	0.512	0.598	0.197	0.397	0.639
Electric.	24	0.260	<b>0.251</b>	0.528	0.427	0.263	0.281
	48	<b>0.319</b>	0.346	0.409	0.551	0.373	0.381
	168	<b>0.427</b>	0.544	0.959	0.893	0.609	0.599
	336	<b>0.565</b>	0.713	1.079	1.035	0.855	0.823
	720	<b>0.861</b>	1.182	1.001	1.548	1.263	1.278
Avg.		<b>0.209</b>	0.310	0.370	0.399	0.338	1.099

Table 2: Univariate time series forecasting results on MSE.

Forecasting

Учим линейную модель  
поверх unsupervised-методов.  
Для предсказания используется эмбеддинг  
последней точки.



Лучшие модели

# TS2Vec (Эксперименты)

## Как выделять аномалии

- Для точки получаем 2 эмбеддинга:
  - $r_t^u$  — не маскируем  $t$ ;
  - $r_t^m$  — маскируем  $t$ .
- Считаем скор аномальности:
 
$$\alpha_t = \|r_t^u - r_t^m\|_1 \cdot \frac{\alpha_t - \bar{\alpha}_t}{\bar{\alpha}_t}$$
- Нормируем на среднее в окне.
- Считаем аномалией всё за порогом:
 
$$\alpha_t^{adj} > \mu + \beta\sigma$$

Anomaly Detection

	Yahoo			KPI		
	F <sub>1</sub>	Prec.	Rec.	F <sub>1</sub>	Prec.	Rec.
SPOT	0.338	0.269	0.454	0.217	0.786	0.126
DSPOT	0.316	0.241	0.458	0.521	0.623	0.447
DONUT	0.026	0.013	0.825	0.347	0.371	0.326
SR	0.563	0.451	0.747	0.622	0.647	0.598
TS2Vec	<b>0.745</b>	0.729	0.762	<b>0.677</b>	0.929	0.533

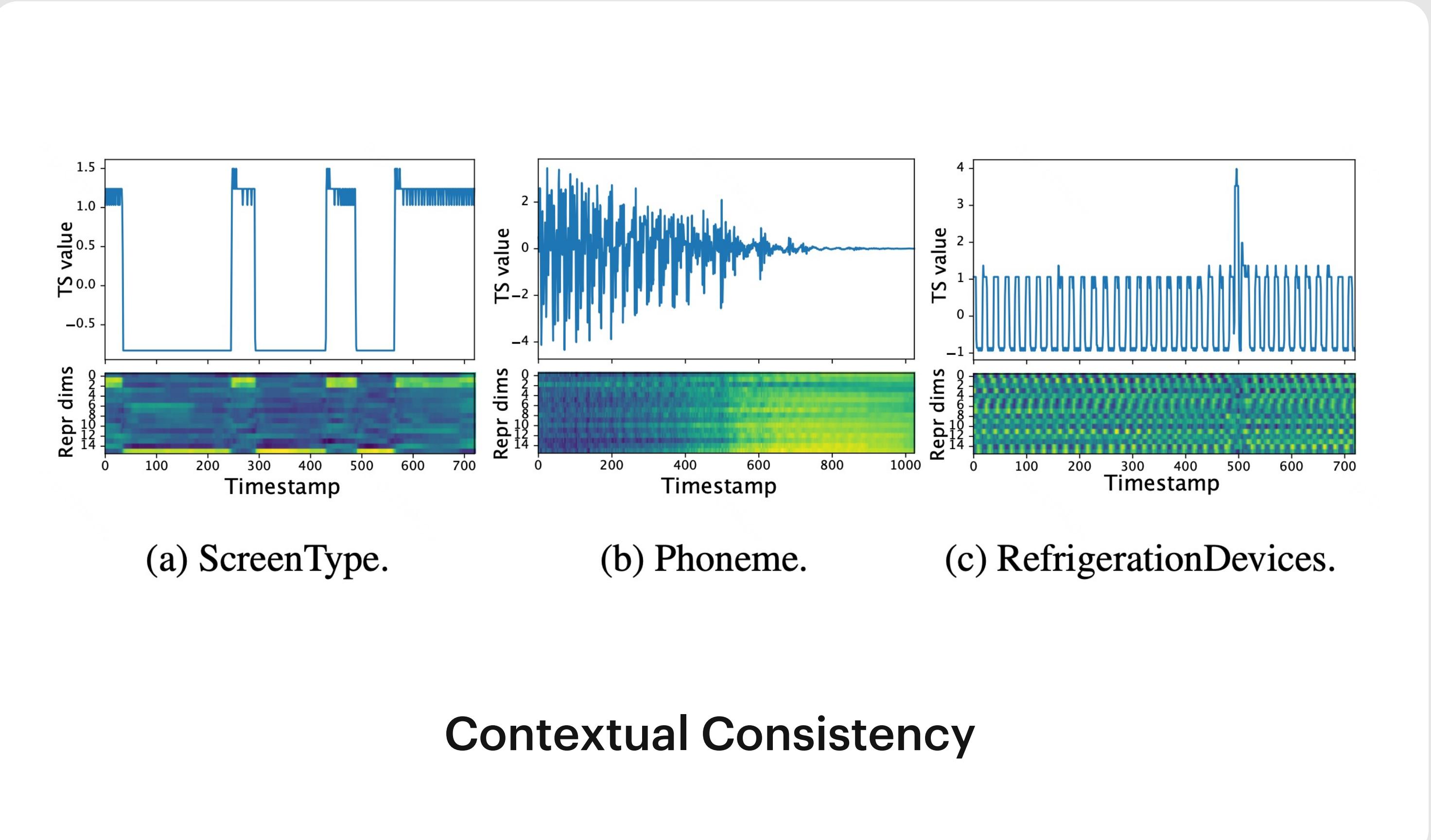
<i>Cold-start:</i>						
	F <sub>1</sub>	Prec.	Rec.	F <sub>1</sub>	Prec.	Rec.
FFT	0.291	0.202	0.517	0.538	0.478	0.615
Twitter-AD	0.245	0.166	0.462	0.330	0.411	0.276
Luminol	0.388	0.254	0.818	0.417	0.306	0.650
SR	0.529	0.404	0.765	0.666	0.637	0.697
TS2Vec <sup>†</sup>	<b>0.726</b>	0.692	0.763	<b>0.676</b>	0.907	0.540

Метрики для точечных аномалий **online**

# TS2Vec (Ablation)



	Avg. Accuracy
<b>TS2Vec</b>	<b>0.829</b>
w/o Temporal Contrast	0.819 (-1.0%)
w/o Instance Contrast	0.824 (-0.5%)
w/o Hierarchical Contrast	0.812 (-1.7%)
w/o Random Cropping	0.808 (-2.1%)
w/o Timestamp Masking	0.820 (-0.9%)
w/o Input Projection Layer	0.817 (-1.2%)
<i>Positive Pair Selection</i>	
Contextual Consistency	
→ Temporal Consistency	0.807 (-2.2%)
→ Subseries Consistency	0.780 (-4.9%)
<i>Augmentations</i>	
+ Jitter	0.814 (-1.5%)
+ Scaling	0.814 (-1.5%)
+ Permutation	0.796 (-3.3%)
<i>Backbone Architectures</i>	
Dilated CNN	
→ LSTM	0.779 (-5.0%)
→ Transformer	0.647 (-18.2%)



# TS-TCC (Гиперпараметры)



## Генерация кропов

max\_train\_length: Optional[int] = None,  
temporal\_unit: int = 0,

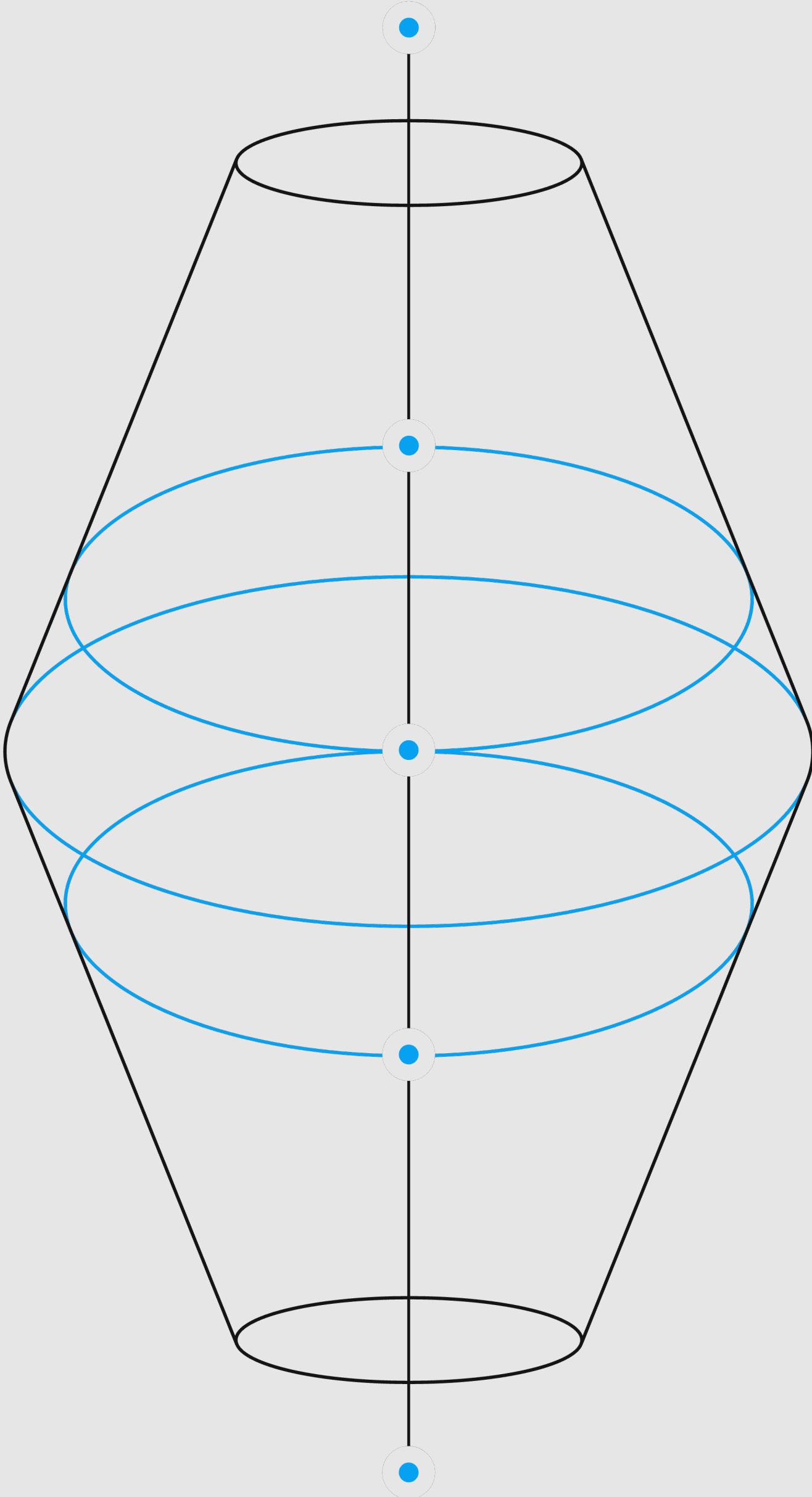
## Энкодер

output\_dims: int = 320,  
hidden\_dims: int = 64,  
depth: int = 10,

## Инференс (контекст)

mask: Literal["binomial", "continuous", "all\_true", "all\_false", "mask\_last"] = "all\_true",  
sliding\_length: Optional[int] = None,  
sliding\_padding: int = 0,

# Заключение



# Про что поговорили

- Зачем нужны векторные представления (эмбеддинги)
- Классические методы
- DL-методы