УДК 519.246.8+004.032.26+311.2
DOI: 10.18469/ikt.2024.22.3.03

ПРЕДСКАЗАНИЕ ВРЕМЕНИ РАБОТЫ ВАРИАЦИОННЫХ АВТОКОДИРОВЩИКОВ НА ОСНОВЕ МОДЕЛЕЙ ВРЕМЕННЫХ РЯДОВ СЕМЕЙСТВА ARIMA

Березкин А.А., Ченский А.А., Киричек Р.В.
Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, Санкт-Петербург, РФ
E-mail: berezkin.aa@sut.ru, chenskii.aa@sut.ru, kirichek@sut.ru

В настоящее время нейронные сети вида «вариационный автокодировщик» применяются в разных сферах, в том числе для сжатия изображений. В задачах FPV-управления беспилотными летательными аппаратами они находят применение в составе нейросетевых кодеков видеопотока. Нейросетевые кодеки позволяют обеспечить большую степень сжатия кадров и передавать FPV-видеопотоки по каналам с низкой пропускной способностью. Время ра-боты вариационных автокодировщиков непостоянно и зависит от оборудования и степени его загрузки. В настоя-щей работе время работы вариационных автокодировщиков рассматривается как временной ряд. Исследуются предсказательные модели времени выполнения вариационных автокодировщиков как на этапе кодирования видеопотока, так и на этапе декодирования. В результате показана непригодность использования ARIMA-моделей для предсказания времени работы выполнения автокодировщиков. Разработаны предсказательные модели для времени работы кодировщиков и декодировщиков.

Ключевые слова: *латентное пространство признаков, вариационный автокодировщик, нейронные сети, ARIMA, нейросетевой кодек, предсказание, статистическая модель*

Введение

Нейросетевой автокодировщик – это вид нейронных сетей, кодирующий исходные данные x_i в некоторый тензор внутреннего латентного пространства признаков (ЛПП) zi и декодирующий данный тензор в восстановленные данные $\widehat{x_i}$. Назначение автокодировщиков заключается в более компактном представлении входа при минимизации ошибки между исходными данными xi и восстановленными данными $\widehat{x_i}$. Вариационный автокодировщик (рисунок 1) – это разновидность автокодировщиков, преобразующий исходные данные x_i в некоторое распределение P(z) [1; 2]. Тензор z_i получается в результате извлечения выборки из распределения P(z).

Пусть $p(x)$ – распределение исходных данных x, $p(x|z)$ – распределение исходных данных x при известном тензоре ЛПП z, а $q(z|x)$ – распределение тензора ЛПП z при известных исходных данных x. Распределение $p(x|z)$ зависит от набора весов вариационного автокодировщика $\theta \in \Omega$, где Ω – множество всех возможных наборов весов вариационного автокодировщика [2]. Тогда $p_\theta(x \mid z)$ – распределение исходных данных x при известном тензоре ЛПП z и наборе весов θ. Тогда выполняется соотношение:

$$P(x) = \int P(x \mid z;\theta) P(z) dz \text{ [1].}$$

Автокодировщики состоят из двух частей: кодировщика $C(x)$ и декодировщика $DCD(z)$. Кодировщик преобразует исходные данные x в тензор ЛПП z, либо в случае вариационных автокодировщиков, в распределение тензоров ЛПП. Декодировщик преобразует тензор ЛПП z в восстановленные данные x'.

Исходные данные x в зависимости от разновидности автокодировщика $z = \mu + \sigma \odot \varepsilon$, где μ – среднее значение x, σ – среднеквадратическое отклонение x и $\varepsilon \in N(0,1)$. x могут представлять собой: изображения [3], аудиоданные [4], видео [5], таблицы [6] и графы [7; 8].

Одним из вариантов практического применений вариационных автокодировщиков являются нейросетевые кодеки [9; 10]. Они состоят из нейросетевых кодеров, кодирующих кадры входного видеопотока, и нейросетевых декодеров, восстанавливающих данные кадры на приемной стороне. Нейросетевые кодеки используются для сжатия кадров видеопотока, передачи сжатого представления кадров видеопотока по каналам с низкой пропускной способностью и их последующего восстановления. В состав нейросетевых кодеков помимо вариационных автокодировщиков также входят алгоритмы интерполяций для изменения разрешений изображений, алгоритмы квантования для изменения типа данных тензоров ЛПП, алгоритмы сжатия тензоров ЛПП [9].

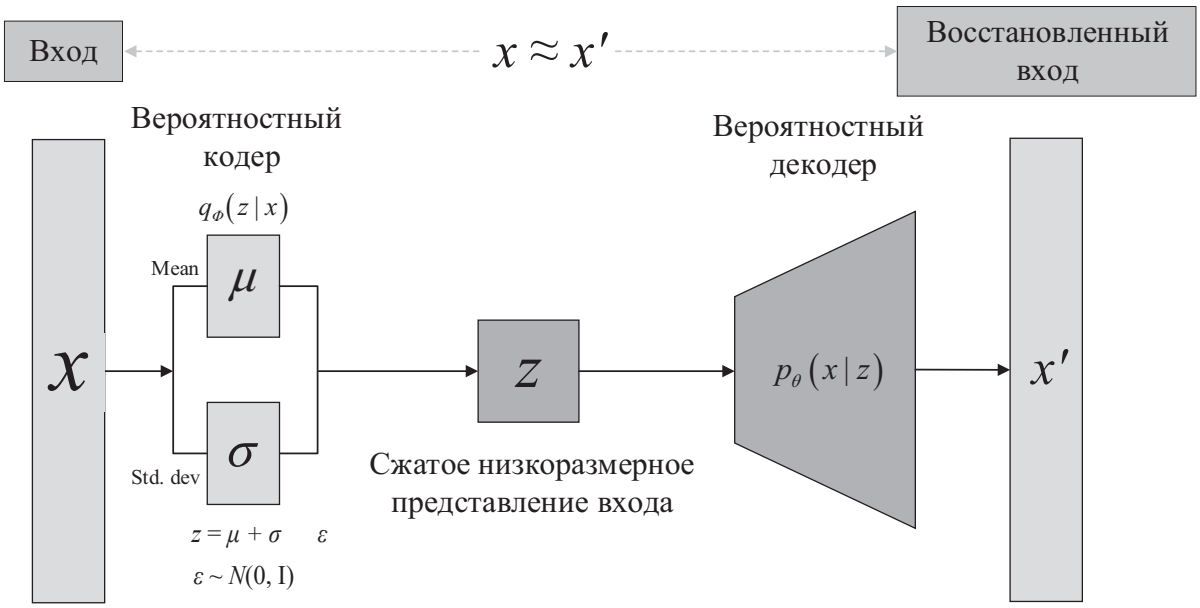

Рисунок 1. Нейросетевой вариационной автокодировщик

Одной из областей применения нейросетевых кодеков является управление беспилотными воздушными судами (БВС) от первого лица (FPV-управление). Беспилотная авиационная система (БАС) включает станцию внешнего пилота (СВП) и беспилотное воздушное судно (БВС). СВП используется внешним пилотом (ВП) для управления БВС. Информационный обмен БАС осуществляется по каналам информационного обмена и содержит две линии связи: линию связи с землей и линию связи с бортом. Линия связи с землей предназначена для передачи информации от БВС к СВП. По ней передаются видеопоток и данные телеметрии. Линия связи с бортом предназначена для передачи команд управления с СВП на БВС.

Видеопоток и данные с БВС принимаются на СВП и отображаются ВП. На их основе ВП формирует команды управления и передает их на БВС. Для успешного выполнения полетных заданий команды управления должны соответствовать полетной обстановке. Соответствие команд управления полетной обстановке зависит от ряда факторов таких, как:

– сетевые задержки передачи данных по линии связи с землей;

– сетевые задержки передачи данных по линии связи с бортом;

– качество исходных кадров видеопотока;

– качество восстановленных кадров видеопотока;

– интенсивность видеопотока на стороне ВП;

– корректность данных с БВС.

Одной из задач обеспечения соответствия команд управления полетной обстановкой и, таким образом, качества FPV-управления является обеспечение достаточной интенсивности видеопотока на СВП. Слишком низкая интенсивность видеопотока приводит к уменьшению возможности воспринимать и, как следствие, реагировать на внештатные ситуации, связанные с неполадками БВС и непредвиденными изменениями полетной обстановки.

Интенсивность видеопотока характеризуется количеством кадров в секунду (Frame Per Second, FPS). Нейросетевой кодер размещается на БВС, а нейросетевой декодер – на стороне СВП. Соответственно, кодирование с передачей по сети и декодирование выполняются параллельно. Интенсивность видеопотока определяется максимальной задержкой:

$$FPS = \frac{1000}{\max\left(t_К + t_{сеть}; t_Д\right)} = \\ = \min\left(\frac{1000}{t_К + t_{сеть}}; \frac{1000}{t_Д}\right)\left(\frac{к}{с}\right), \quad (1)$$

где $t_К$ – время работы нейросетевого кодера в мс;

$t_{сеть}$ – сетевая задержка в миллисекундах;

$t_Д$ – время работы нейросетевого декодера в миллисекундах.

Интенсивность видеопотока не является постоянной величиной. Нейросетевой кодек потребляет большое количество вычислительных ресурсов и время выполнения нейросетевого кодера и декодера сильно зависят от имеющихся вычислительных ресурсов [11]. Следовательно, интенсивность видеопотока зависит от сети передачи, оборудования для размещения нейросетевого кодера, оборудования для размещения нейросете-

вого декодера, загруженности оборудования для размещения нейросетевого кодера, загруженности оборудования для размещения нейросетевого декодера. Тем не менее, вычислительные ресурсы используются для выполнения нескольких потоков. Как было показано в [11], время выполнения нейросетевых кодера и декодера изменяется.

В исследовании [12] представлены методы обеспечения адаптивного контроля интенсивности кадров видеопотока: Roerich [13; 14], SKAB (Skoltech Anomaly Benchmark) [15], контрольные карты Шухарта [16]. Данные методы используются для контроля процесса изменения FPS и обнаружения периодов хорошего и плохого состояния процесса FPS, а также изменения такого процесса (рисунок 2). При ухудшении процесса предполагается задействование компенсаторных механизмов предсказания кадров [12].

В [11] показано, что для корректной работы компенсаторных механизмов необходимо обеспечить возможность предсказания и оценки FPS после предсказания кадров и сравнения с исходными данными. Для этого разработана регрессионная модель предсказания времени выполнения предсказания кадров, и на ее основе составлена математическая модель интенсивности видеопотока. Тем не менее, модель, представленная в [11], плохо работает в граничных условиях, когда:

$$t_{\phi C} + t_{DCD} + t_I + T \approx \frac{0{,}0000378 \times width \times height}{k_p},$$

где t_{CD} – время кодирования (мс);

t_{DCD} – время декодирования (мс);

t_I – время интерполяции (мс);

$T_з$ – дополнительная задержка до поступления кадра из канала связи (мс);

width – ширина кадра;

height – высота кадра;

k_p – коэффициент, показывающий, во сколько раз быстрее нейронная сеть выполняется на видеокарте Nvidia A100 80 Гб, чем на используемой видеокарте.

Так, согласно модели, с $Tз = 0$ при увеличении количества предсказываемых кадров разрешения 1280×720 должно наблюдаться медленное и стабильное уменьшение FPS, в то время как эксперименты показали флуктуации FPS.

Это вызвано тем, что в предлагаемой модели фигурируют только средние значения времени выполнения отдельных операций, таких как кодирование, декодирование, предсказание, без учета их изменения во времени. Сама регрессионная модель, на основе которой составлена математическая модель, имеет ограниченную область применения вследствие гетероскедастичности (дисперсия времени предсказания увеличивается при увеличении количества предсказываемых пикселей).

Для устранения недостатков указанных моделей время кодирования и декодирования в настоящем исследовании рассматриваются и анализируются как временные ряды; составляются модели времени кодирования и декодирования с учетом возможной автокорреляции и вероятностной компоненты.

Нейросетевой кодек

В настоящей работе используется нейросетевой кодек, представленный в [9; 10]. Он состоит из нейросетевого кодера (рисунок 3) и нейросетевого декодера (рисунок 4) [9].

Принцип работы нейросетевого кодера следующий. На его вход поступает кадр видеопотока *f*.

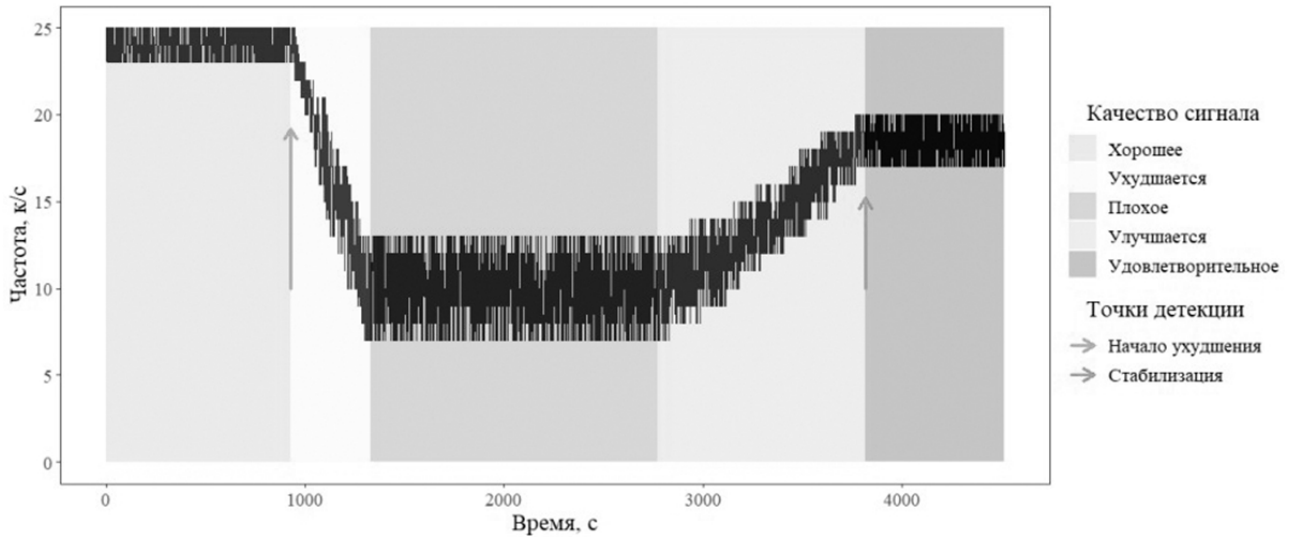

Рисунок 2. Контроль процесса изменения FPS

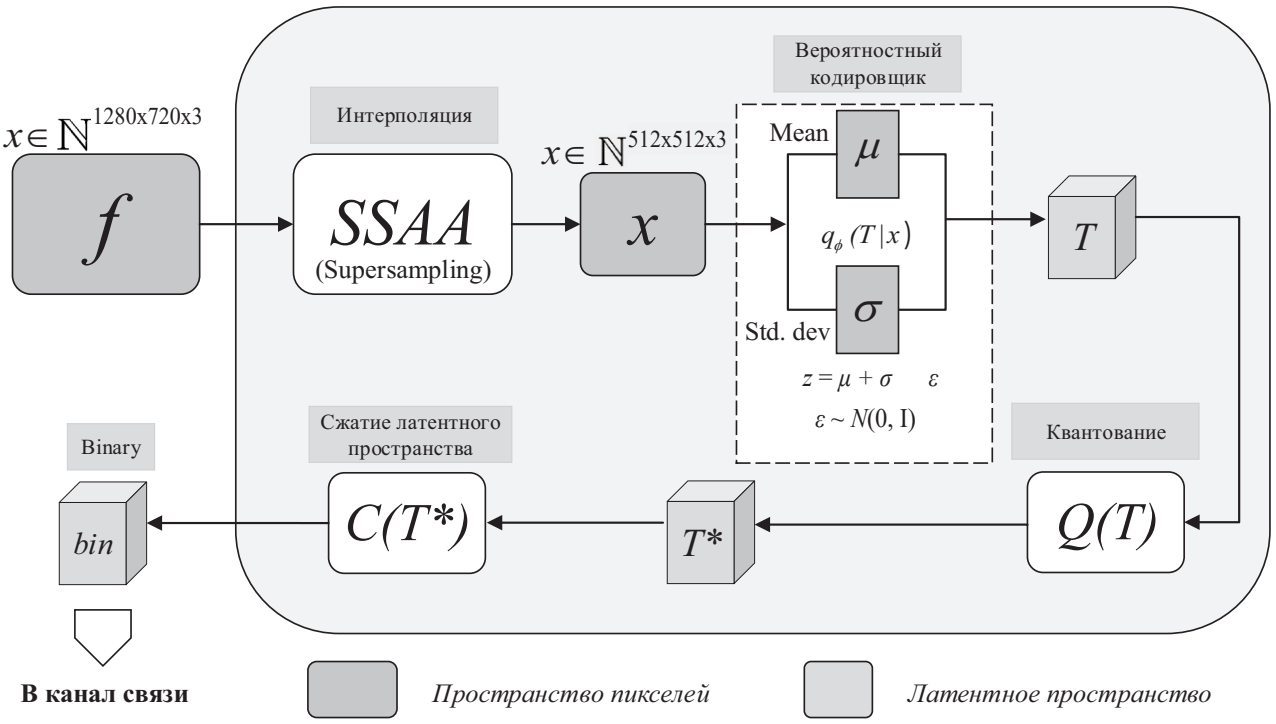

Рисунок 3. Нейросетевой кодер

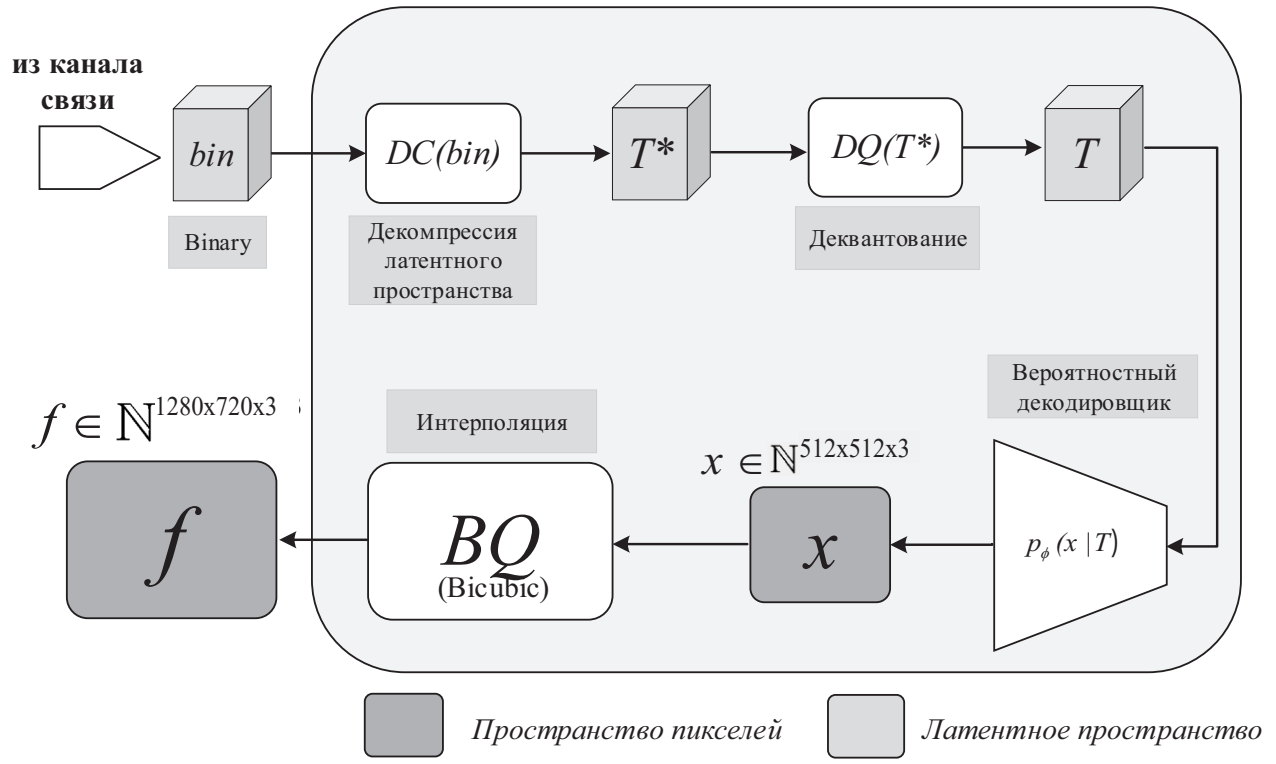

Рисунок 4. Нейросетевой декодер

Далее он интерполируется методом суперсэмплинга (Super Sampling Anti-Aliasing, SSAA) до разрешения 512×512. Интерполированный кадр x кодируется вероятностным кодировщиком в тензор ЛПП T и квантуется $T^* = Q(T)$. Квантованный тензор ЛПП T^* сжимается в выходную бинарную последовательность $bin = C(T^*)$. Полученная бинарная последовательность bin передается по линии связи с землей.

Принцип работы нейросетевого декодера аналогичен кодеру, где процесс преобразований ЛПП происходит в обратном порядке: по линии связи

с землей поступает бинарная последовательность *bin*. Производится ее декомпрессия $T^* = DC(bin)$. Квантованный тензор ЛПП T^* деквантуется $T = DQ(T^*)$. Тензор ЛПП декодируется. Затем полученный интерполируемый кадр x вновь интерполируется методом бикубической интерполяции (BQ) до исходного разрешения (HD). В результате на выходе нейросетевого декодера получается кадр видеопотока f, который отправляется на устройство отображения ВП.

Для рассмотренного нейросетевого кодека:

$$t_{\text{К}} = t_I + t_{CD} + t_Q + t_C \left(\text{мс}\right), \tag{2}$$

где t_I – время интерполяции на кодере (уменьшение разрешения до 512×512);

t_{CD} – время кодирования;

t_Q – время квантования;

t_C – время сжатия в мс.

$$t_{\text{Д}} = t_{DC} + t_{DQ} + t_{DCD} + t_{I2} \left(\text{мс}\right), \tag{3}$$

где t_{DC} – время декомпрессии;

t_{DQ} – время деквантования;

t_{DCD} – время декодирования;

t_{I2} – время интерполяции на декодере (увеличение разрешения до исходного) в мс.

Таким образом, исходя из (2) и (3), для данного нейросетевого кодека выражение (1) может быть записано как:

$$\begin{gathered} FPS = \frac{1000}{\max\left(t_{\text{К}} + t_{\text{сеть}}; t_{\text{Д}}\right)} = \\ = \min\left(\frac{1000}{t_{\text{К}} + t_{\text{сеть}}}; \frac{1000}{t_{\text{Д}}}\right)\left(\frac{k}{c}\right), \\ FPS = \min\left(\frac{1000}{t_{\text{К}} + t_{\text{сеть}}}; \frac{1000}{t_{\text{Д}}}\right) = \\ \text{in}\left(\frac{1000}{t_I + t_{CD} + t_Q + t_C + t_{\text{сеть}}}; \frac{1000}{t_{DC} + t_{DQ} + t_{DCD} + t_{I2}}\right). \end{gathered} \tag{4}$$

Согласно [11],

$$t_{DCD} \gg t_{DC} \gg t_{I2}\,. \tag{5}$$

При этом, согласно [10], для VQ и KL моделей вариационных автокодировщиков время кодирования сопоставимо со временем декодирования (от $t_{CD} \approx \frac{t_{DCD}}{2}$ для KL-f32 до $t_{CD} \approx 3 \times t_{DCD}$ для VQ-f4). Обозначим сопоставимость времени выполнения операций через ~:

$$t_{DCD} \sim t_{CD}\,. \tag{6}$$

Пусть тогда:

$$t_Q \sim t_{DQ},\ t_C \sim t_{DC},\ t_I \sim t_{I2}\,. \tag{7}$$

На основе (6) и (7), обобщим формулу (5):

$$t_{DCD} \sim t_{CD} \gg t_{DC} \sim t_C \gg t_{I2} \sim t_I\,. \tag{8}$$

Используемые алгоритмы квантования [17] не задействуют нейронные сети и потребляют меньшее количество ресурсов, чем вариационные автокодировщики. Таким образом, примем также:

$$t_{DCD} \sim t_{CD} \gg t_{DQ} \sim t_Q\,. \tag{9}$$

На основе (4), (8) и (9) получим грубую оценку интенсивности видеопотока:

$$\begin{gathered} FPS = \min\left(\frac{1000}{t_{\text{К}} + t_{\text{сеть}}}; \frac{1000}{t_{\text{Д}}}\right) \approx \\ \approx \min\left(\frac{1000}{t_{CD} + t_{\text{сеть}}}; \frac{1000}{t_{DC}}\right)\left(\frac{k}{c}\right)\,. \end{gathered} \tag{10}$$

Из формулы (10) видно, что для приблизительной оценки интенсивности видеопотока необходимо знать t_{CD}, t_{DC} на заданном оборудовании и $t_{сеть}$ для сети, которая используется для линии связи с землей. $t_{сеть}$ не может быть получена на нейросетевом кодеке непосредственно и требует отдельной реализации в связке с протоколом FPV-управления. Таким образом, в системе адаптивной интенсивности видеопотока на стороне нейросетевого кодека важным становится требование к получению, анализу и прогнозированию параметров t_{CD} и t_{DC}.

Целью настоящей работы является исследование и моделирование времени кодирования t_{CD} и декодирования t_{DCD} как временных рядов.

Модели временных рядов

В настоящей работе рассматривается семейство моделей временных рядов ARIMA [18–21]. Семейство включает такие модели, как AR (Auroregressive – авторегрессионная модель), MA (Moving Average – модель скользящей средней), ARMA (Autoregressive Moving Average – авторегрессионная модель скользящей средней), ARIMA (Autoregressive Integrated Moving Average Model –интегрированная модель авторегрессии скользящего среднего). Существуют сезонные модели SAR (Seasonal Autoregressive – сезонная авторегрессионная модель), SMA (Seasonal Moving Average – сезонная модель скользящей средней), SARMA (Seasonal Autoregressive Moving Average – сезонная авторегрессионная модель скользящей средней), SARIMA (Seasonal Autoregressive Integrated Moving Average Model – сезонная интегрированная модель авторегрессии-скользящего среднего). Эти модели подразумевают наличие тренда и используются в основном в эконометрике. По этой причине сезонные модели SAR, SMA, SARMA и SARIMA не рассматривается в настоящей работе.

Модель AR(p) имеет параметр порядка p. Пусть $wt \sim wn\left(0,\sigma_w^2\right)$ – процесс белого шума. Процесс белого шума представляет собой последовательность случайных некоррелированных значений со средним значением 0 и среднеквадратическим отклонением, равным σ_w^2 [20]. Тогда случайный процесс y(t) является AR, когда выполняется соотношение [19]:

$$y(t)-\mu=\varphi_1\left(y\left(t-1\right)-\mu\right)+\varphi_2\left(y\left(t-2\right)-\mu\right)+...+$$
$$+\varphi_p\left(y\left(t-p\right)-\mu\right)+wt_t \quad ,$$

где μ – среднее значение случайного процесса $y(t)$;
$\varphi_1,\varphi_2,...,\varphi_p$ – коэффициенты AR-модели.

Модель MA(q) имеет параметр порядка q. Случайный процесс $y(t)$ является MA, когда выполняется соотношение [19]:

$$y(t)=\mu+wt_t+\theta_1 wt_{t-1}+...+\theta_q wt_{t-q},$$

где $\theta_1,...,\theta_q$ – коэффициенты MA-модели.

Дисперсия случайного процесса MA равна:

$$D\left[y(t)\right]=\sigma_w^2\times\left(1+\theta_1^2+...+\theta_q^2\right).$$

Модель ARMA(p, q) является комбинацией AR и MA-моделей. Она содержит два параметра: p – порядок авторегрессии и q – порядок скользящего среднего. Случайный процесс $y(t)$ является ARMA, когда выполняется соотношение [19]:

$$y(t)-\mu=\varphi_1\left(y\left(t-1\right)-\mu\right)+$$
$$+\varphi_2\left(y\left(t-2\right)-\mu\right)+...+\varphi_p\left(y\left(t-p\right)-\mu\right)+ \quad (11)$$
$$+wt_t+\theta_1 wt_{t-1}+...+\theta_q wt_{t-q} \quad .$$

Модели AR, MA и ARMA предназначены для стационарных рядов. Пусть есть случайный процесс $y(t)$, заданный на множестве T. Случайный процесс $y(t)$ строго стационарен, если при $\forall n,t_1,t_1,...,t_n,\tau$ распределения случайных величин $\left\{y\left(t_1\right),y\left(t_2\right),...,y\left(t_n\right)\right\}$ и $\left\{y\left(t_1+\tau\right),y\left(t_2+\tau\right),...,y\left(t_n+\tau\right)\right\}$ одинаковы.

Ряды не стационарны, когда в них присутствует тренд и сезонность. В случае сезонности необходимо использовать сезонные модели SAR, SMA, SARMA, SARIMA. Пусть есть некоторый временной ряд $\hat{y}(t)$, в котором присутствует тренд. Тогда для применения моделей AR, MA, ARMA необходимо провести детрендирование ряда путем вычитания его тренда. Тем не менее, в ряде случаев детрендирование ряда не приводит к появлению стационарного ряда. Другим способом преобразования ряда в стационарный является дифференцирование ряда [19]:

$$\Delta^k\hat{y}(t)=\hat{y}(t)-C_k^1\hat{y}(t-1)+C_k^2\hat{y}(t-2)-...+$$
$$+\left(-1\right)^k\hat{y}(t-k) \quad ,$$

где k – порядок интеграции временного ряда.

Модель ARIMA(p, k, q) основана на дифференцировании временного ряда. Она содержит параметры p, k и q. Случайный процесс $\hat{y}(t)$ является ARIMA, когда выполняется соотношение (11) при $y\left(t\right)=\Delta^k\hat{y}(t)$.

Модели AR и MA являются частным случаем модели ARMA с $q=0$, либо $p=0$ соответственно. Модель ARMA в свою очередь является частным случаем модели ARIMA с $k=0$.

Методика исследования

Целью настоящего исследования является анализ и моделирование времени кодирования t_{CD} и декодирования t_{DCD} как временных рядов. Для достижения цели необходимо выполнить ряд задач:

1. *Составление четырех временных рядов:* тренировочного и проверочного отдельно для времени кодирования t_{CD} и времени декодирования t_{DCD}. Для составления временных рядов будем использовать авторский набор данных из 1000 кадров видеопотока, полученных в рамках проектно-образовательного интенсива «Архипелаг 2024» [22] (рисунок 5). Кадры видеопотока поступают на нейросетевой кодер, после чего фиксируется время кодирования. Полученные на выходе бинарные последовательности *bin* поступают на нейросетевой декодер, после чего фиксируется время декодирования кадра видеопотока.

В составе нейросетевого кодека (рисунки 4 и 5) используется вариационный автокодировщик KL-f4. Для корректного измерения времени кодирования и декодирования вариационным

Рисунок 5. Примеры кадров авторского набора данных

автокодировщиком используется функция *torch.cuda.syncronize()* из библиотеки PyTorch [23] для языка программирования Python. Исполнение вариационного автокодировщика проводится на видеокарте Nvidia A100 80 Гб [24]. Полученные бинарные последовательности длиной 1000 преобразуются в формат *numpy.ndarray*.

2. *Идентификация моделей ARIMA.* Идентификация модели ARIMA(*p*, *k*, *q*) для некоторого временного ряда $\hat{y}(t)$ заключается в нахождении значений параметров *p*, *k* и *q*.

На первом этапе необходимо получить значение параметра *k*. Для этого необходимо найти такое значение *k*, при котором временной ряд $\Delta^k \hat{y}(t)$ является стационарным. Если временной ряд $\hat{y}(t)$ уже является стационарным, то *k = 0* и дифференцирование ряда не требуется. В результате будет получен стационарный временной ряд $y(t) = \Delta^k \hat{y}(t)$.

Проверка временного ряда на стационарность проводится расширенным тестом Дики-Фулера (Augmented Dickey-Fuller Test, ADF) [25] с уровнем значимости 5%. Тест основан на понятии единичного корня. Если первые разности временного ряда $\hat{y}(t)$ образуют стационарный ряд, то временный ряд $\hat{y}(t)$ имеет единичный корень. Когда у временного ряда $\hat{y}(t)$ есть единичный корень, он не является стационарным.

Представим временной ряд $\hat{y}(t)$ в виде [21]:

$$\hat{y}(t) = C_t + \beta \hat{y}(t-1) + \sum_{i=1}^{p-1} \phi_i \hat{y}(t-i) + wt_t ,$$

где C_t – значения некоторой детерминированной функции, и найдем оценку $\hat{\beta}$ значения коэффициента β методом наименьших квадратов. Тогда получим ADF-статистику:

$$ADF_{\text{ñò àò}} = \frac{\hat{\beta} - 1}{\sigma_{\hat{\beta}}},$$

где $\sigma_{\hat{\beta}}$ – среднеквадратическое отклонение $\hat{\beta}$.

Гипотеза расширенного теста Дики-Фулера H0: $\beta = 1$. Это значит, что временной ряд $\hat{y}(t)$ имеет единичный корень и не является стационарным. Гипотеза расширенного теста Дики-Фулера H1: $\beta < 1$. В этом случае временной ряд $\hat{y}(t)$ не имеет единичного корня и является стационарным.

После нахождения значения параметра *k* модели ARIMA необходимо найти также значения параметров *p* и *q*. Модель ARIMA для временного ряда $\hat{y}(t)$ сводится к модели ARMA для временного ряда *y*(*t*). ARIMA сводится к AR для ряда *y*(*t*) при *q* = 0 и к MA при *p* = 0. Данные случаи сравнительно легко идентифицировать при построении графиков зависимостей значений функции автокорреляции (Autocorrelation Function, ACF) и функции частной автокорреляции (Partial Autocorrelation Function, PACF) от лага (корреллограмм).

Функция автокорреляции ACF показывает, насколько коррелируют значения временного ряда $\hat{y}(t)$ с некоторым лагом *l* [19]:

$$r(l) = \text{corr}[\hat{y}(t), \hat{y}(t+l)] = \frac{\text{cov}\left[\hat{y}(t), \hat{y}(t+l)\right]}{\sigma\left(\hat{y}(t)\right) \times \sigma\left(\hat{y}(t+l)\right)} = \frac{\text{cov}\left[\hat{y}(t), \hat{y}(t+l)\right]}{D\left(\hat{y}(t)\right)},$$

где corr – функция корреляции;

cov – функция ковариации;

D – дисперсия;

σ – среднеквадратическое отклонение.

Функция частной автокорреляции PACF показывает автокорреляцию между разделенными *l* лагами временного ряда при устранении неявного влияния значений временного ряда с промежуточными лагами.

Пусть дана матрица коэффициентов корреляции [19]:

$$R = \begin{bmatrix} r_{00} & r_{01} & \cdots & r_{0l} \\ r_{10} & r_{11} & \cdots & r_{1l} \\ \vdots & \vdots & \ddots & \vdots \\ r_{l0} & r_{l1} & \cdots & r_{ll} \end{bmatrix},$$

где $\forall i \in \{0,1,...,k\}; \forall j \in \{0,1,...,k\} : \begin{cases} r_{ij} = r(s) \\ s = |i - j| \end{cases}$.

Пусть R_{ij} – алгебраическое дополнение к *rij*. Тогда коэффициент частичной корреляции равен:

$$r_{\text{част}}(l) = -\frac{R_{0l}}{\sqrt{R_{00} \times R_{kk}}}.$$

Пусть *y*(*t*) является временным рядом модели AR. Тогда на корреллограммах ACF будет постепенно убывать, а PACF станет приблизительно равна нулю после лага *l* = *p*.

Пусть *y*(*t*) является временным рядом модели MA. Тогда на корреллограммах ACF станет приблизительно равна нулю после лага *l* = *q*, а PACF будет постепенно убывать.

В случае же, если на корреллограммах и ACF, и PACF постепенно убывают, $p \neq 0$, $q \neq 0$, а *y*(*t*) является временным рядом модели ARMA. ACF и PACF не являются информативными для идентификации ARMA-модели [21]. Одним из методов нахождения значений параметров *p* и *q* в данном случае является так называемый сеточный метод, заключающийся в переборе различных комбинаций значений *p* и *q* и сравнении полученных моделей.

3. *Обучение моделей.* Обучение ARIMA-моделей проводится с помощью метода максимального правдоподобия, реализованного в методе *fit* класса *statmodels.tsa.arima.ARIMA* библиотеки *statsmodels* [26] языка программирования Python.

4. *Верификация моделей.* Пусть есть две модели: ARIMAcd для кодирования и ARIMAdcd для декодирования и два временных ряда: *cdi* для t_{CD} и *dcdi* для *tDCD* со средними значениями $\overline{t}_{CD}$ и $\overline{t}_{DCD}$. Предсказываемые значения с помощью ARIMA-моделей: $\widehat{cd}_i$ и $\widehat{dcd}_i$. Тогда находится среднеквадратическая ошибка предсказания MSE (Mean Squared Error) [27] для предсказания ARIMA-моделями и для предположения среднего значения:

$$MSE_{cd\_arima} = \frac{1}{DTST}\sum_{i=1}^{DTST}\left(\widehat{cd}_i - cd_i\right)^2,$$

$$MSE_{cd\_mean} = \frac{1}{DTST}\sum_{i=1}^{DTST}\left(\overline{t}_{CD} - cd_i\right)^2,$$

$$MSE_{dcd\_arima} = \frac{1}{DTST}\sum_{i=1}^{DTST}\left(\widehat{dcd}_i - dcd_i\right)^2,$$

$$MSE_{dcd\_mean} = \frac{1}{DTST}\sum_{i=1}^{DTST}\left(\overline{t}_{DCD} - dcd_i\right)^2,$$

где $DTST$ – длина набора данных;

MSE_{cd_arima} – ошибка предсказания моделью $ARIMA_{cd}$;

MSE_{cd_mean} – ошибка предсказания средним временем кодирования $\overline{t}_{CD}$;

MSE_{dcd_arima} – ошибка предсказания моделью *ARIMAdcd*;

MSE_{dcd_mean} – ошибка предсказания средним временем декодирования $\overline{t}_{DCD}$.

ARIMAcd получает лучшие результаты, чем при использовании $\overline{t}_{CD}$, когда выполняется неравенство:

$$MSE_{cd\_arima} < MSE_{cd\_mean}.$$

ARIMAdcd получает лучшие результаты, чем при использовании $\overline{t}_{DCD}$, когда выполняется неравенство:

$$MSE_{dcd\_arima} < MSE_{dcd\_mean}.$$

При выполнении обоих неравенств составленные в рамках исследования модели считаются верифицированными.

Все полученные на этапе 2 и обученные на этапе 3 модели верифицируются. Для каждой получаются MSE на тестовом наборе и критерий Акайкэ (Akaike Information Criterion, AIC) на тренировочном наборе [21]:

$$AIC = \frac{-2}{T}\ln(L) + \frac{2}{T}PRM,$$

где L – максимальное значение функции правдоподобия;

T – размер выборки;

PRM – количество параметров модели.

Модели среднего с белым шумом соответствует модель ARIMA(0, 0, 0).

Временные ряды кодирования и декодирования

Кадры видеопотока авторского набора поступают на вход нейросетевого кодека. В результате вычисляется время кодирования (рисунок 6 – тренировочный временной ряд; рисунок 7 – тестовый временной ряд) и время декодирования (рисунок 8 – тренировочный временной ряд и рисунок 9 – тестовый временной ряд).

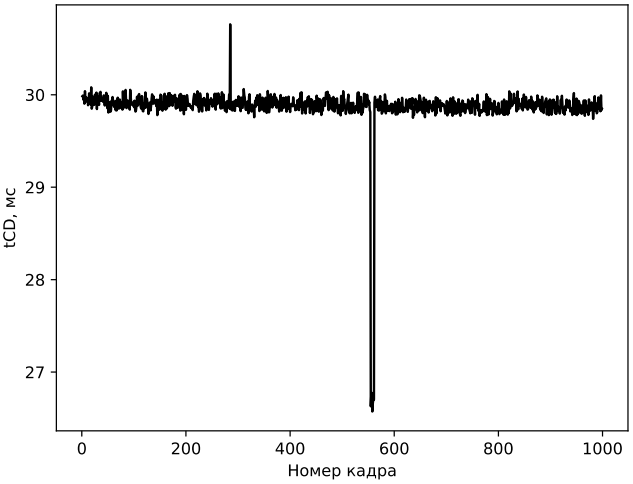

Рисунок 6. Тренировочный временной ряд времени кодирования

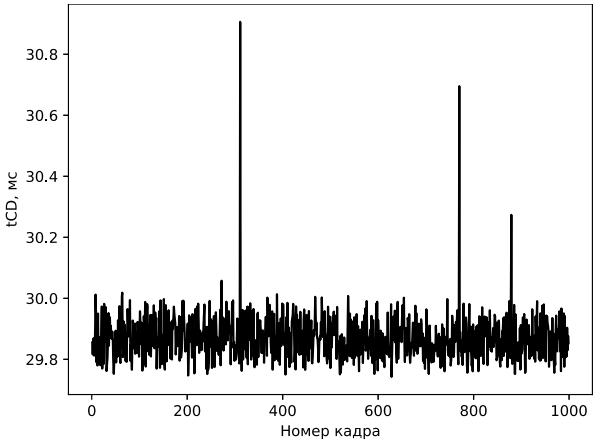

Рисунок 7. Тестовый временной ряд времени кодирования

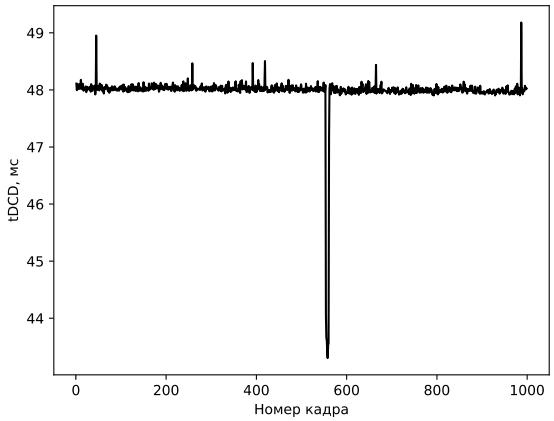

Рисунок 8. Тренировочный временной ряд времени декодирования

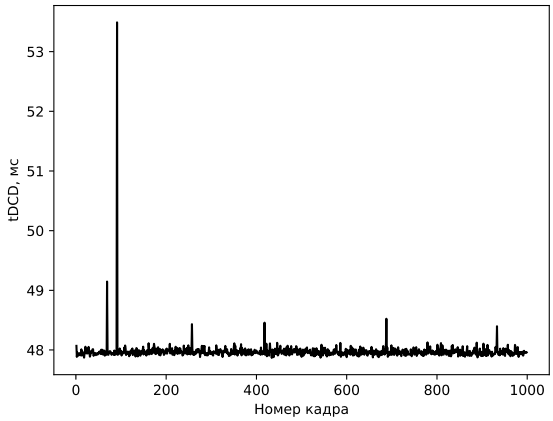

Рисунок 9. Тестовый временной ряд времени декодирования

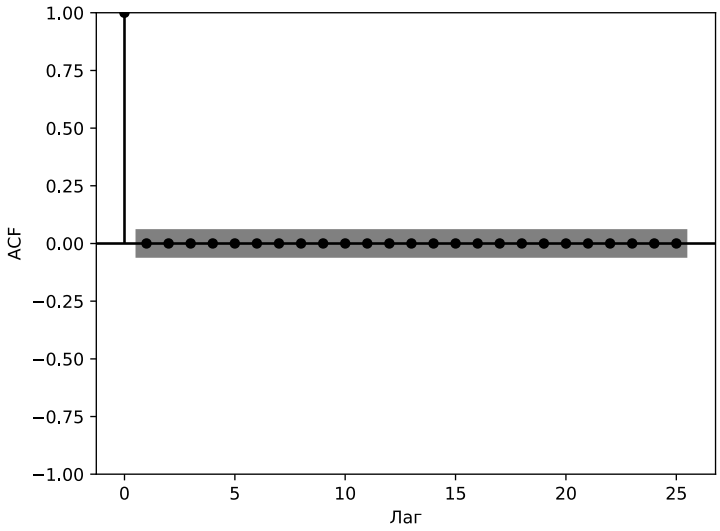

Рисунок 10. Зависимость ACF от лага для временного ряда кодирования

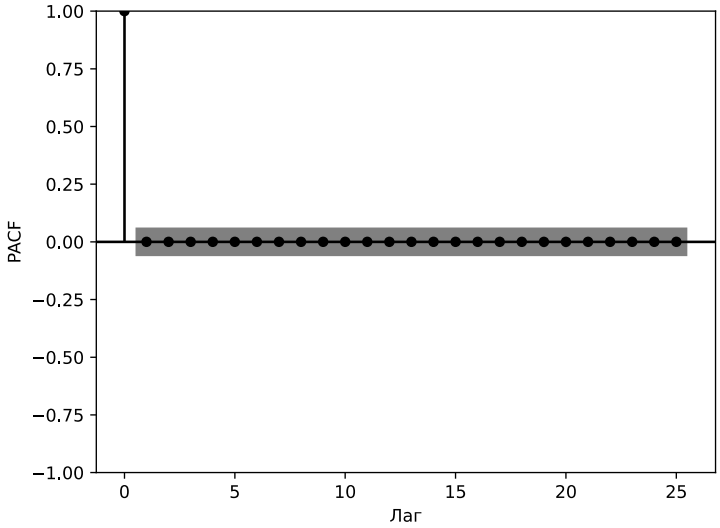

Рисунок 11. Зависимость PACF от лага для временного ряда кодирования

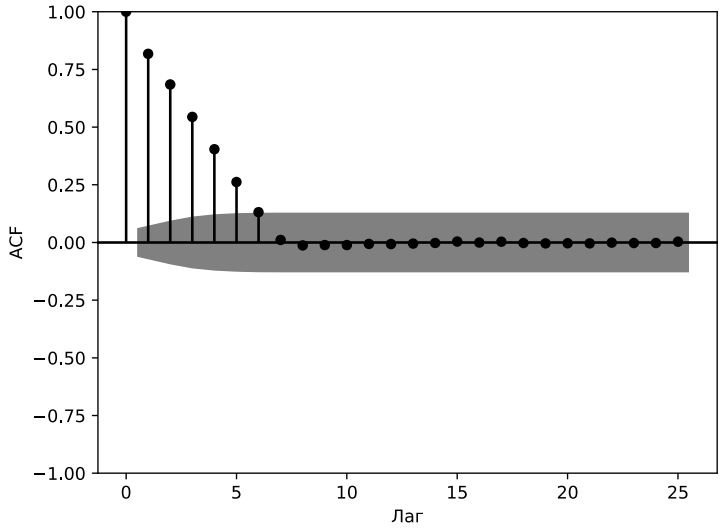

Рисунок 12. Зависимость ACF от лага для временного ряда декодирования

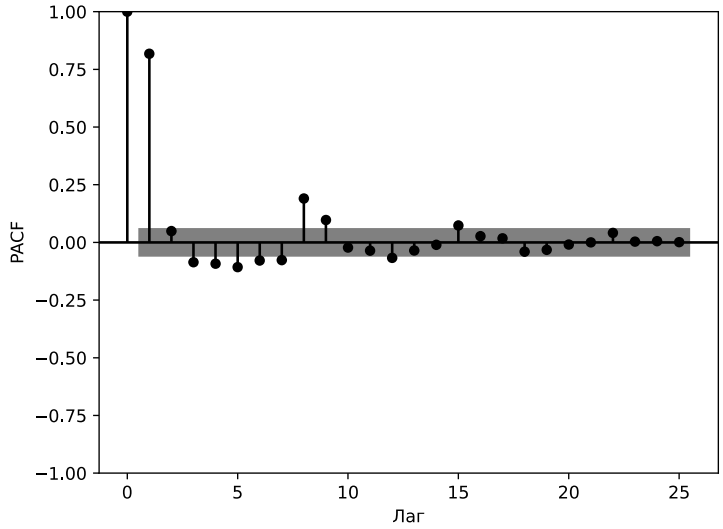

Рисунок 13. Зависимость PACF от лага для временного ряда декодирования

Идентификация моделей

Расширенный тест Дики-Фулера показывает, что все 4 временных ряда не имеют единичного корня и являются стационарными. Таким образом, k = 0.

В рамках идентификации моделей строятся корреллограммы для тестового временного ряда кодирования (рисунки 10, 11) и тестового временного ряда декодирования (рисунки 12, 13).

По построенным корреллограмам видно, что для времени кодирования лучшей является модель среднего, а для времени декодирования – MA(7). Тем не менее, данная оценка является приблизительной. В целях получения точного результата применимости ARIMA-моделей для предсказания времени кодирования и декодирования используем сеточный метод (grid method).

Обучение и верификация моделей

В результате обучения моделей получены лучшие модели (10 штук) для кодирования (таблица 1) и декодирования (таблица 2).

Таблица 1. Десять лучших моделей для предсказания времени кодирования

Место	Модель	MSE	AIC
1	ARIMA(0, 0, 0)	0,00557	269,189
2	ARIMA(0, 0, 5)	0,00805	-759,629
3	ARIMA(0, 0, 1)	0,00807	-305,097
4	ARIMA(6, 0, 1)	0,00840	-816,533
5	ARIMA(0, 0, 2)	0,00842	-530,118
6	ARIMA(2, 0, 0)	0,00844	-790,110
7	ARIMA(6, 0, 0)	0,00844	-815,352
8	ARIMA(2, 0, 2)	0,00845	-816,507
9	ARIMA(7, 0, 0)	0,00845	-836,789
10	ARIMA(1, 0, 3)	0,00846	-797,980

С параметрами $p = k = q = 0$ модели ARIMA вырождаются в модель среднего с белым шумом

$\widehat{cd}_i$ = 29,871 + wt_t(мс), σ_w^2 = 0,00764 (мс^2) для кодирования (рисунок 14) и $\widehat{dcd}_i$ = 47,982 + wt_t(мс), σ_w^2 = 0,00764 (мс^2) для декодирования (рисунок 15).

Наилучшими невырожденными моделями ARIMA являются MA(5), но они существенно уступают моделям среднего с белым шумом.

Использование AIC для выбора модели в данном случае неэффективно: необходимо использование тренировочного набора.

Таким образом, показано, что использование ARIMA-моделей для предсказания времени выполнения кодирования и декодирования неэффективно. Рекомендуется использовать модель белого шума со средним.

Таблица 2. Десять лучших моделей для предсказания времени декодирования

Место	Модель	MSE	AIC
1	ARIMA(0, 0, 0)	0,0351	894,898
2	ARIMA(0, 0, 5)	0,0597	-374,685
3	ARIMA(1, 0, 0)	0,0607	-439,497
4	ARIMA(2, 0, 3)	0,0610	-485,959
5	ARIMA(7, 0, 0)	0,0617	-477,467
6	ARIMA(2, 0, 2)	0,0620	-470,770
7	ARIMA(1, 0, 4)	0,0621	-469,063
8	ARIMA(6, 0, 1)	0,0622	-472,562
9	ARIMA(6, 0, 0)	0,0623	-473,891
10	ARIMA(4, 0, 1)	0,0624	-469,207

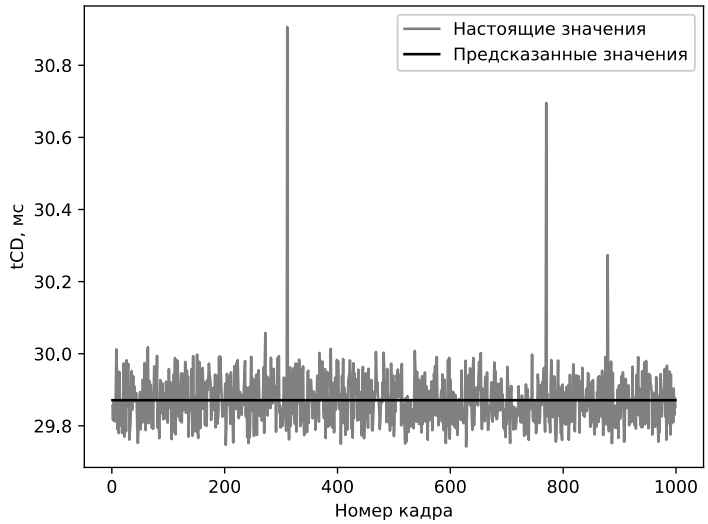

Рисунок 14. Предсказываемые и реальные значения времени кодирования

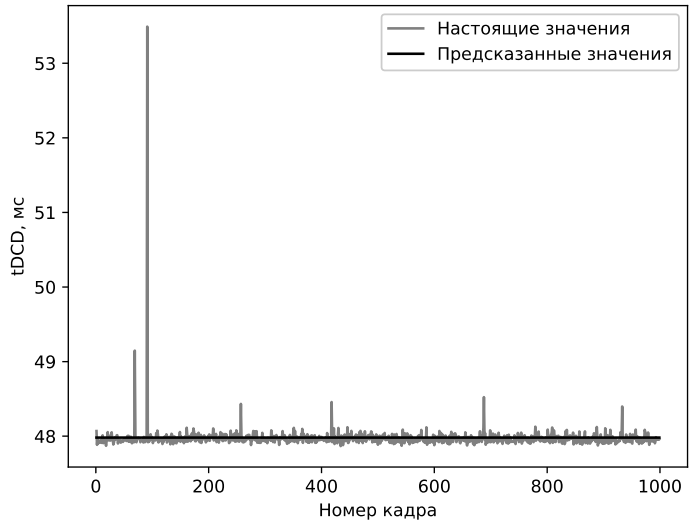

Рисунок 15. Предсказываемые и реальные значения времени декодирования

Заключение

Исследована возможность прогнозирования времени кодирования и декодирования вариационными автокодировщиками с помощью моделей временных рядов семейства ARIMA. Показано, что модели семейства ARIMA неэффективно использовать для предсказания времени выполнения кодирования и декодирования. Рекомендуется использовать модель среднего с белым шумом.

Лучшей моделью, собственно, семейства ARIMA для предсказания времени кодирования и декодирования является модель MA(5), однако, она существенно уступает модели среднего с белым шумом. При использовании модели MA(5) для этапа кодирования MSE = 0,00805 и для этапа декодирования MSE = 0,0597, в то время, как при использовании модели среднего с белым шумом MSE = 0,00557 и 0,0351, соответственно.

Литература

1. Kingma D.P., Welling M. Auto-encoding variational bayes. URL: https://arxiv.org/pdf/1312.6114 (дата обращения: 28.10.2024).
2. Doersch C. Tutorial on variational autoencoders. URL: https://www.researchgate.net/publication/304163568_Tutorial_on_Variational_Autoencoders (дата обращения: 05.11.2024).
3. Razavi A., Van den Oord A., Vinyals O. Generating diverse high-fidelity images with VQ-VAE-2 // Advances in Neural Information Processing Systems. 2019. URL: https://arxiv.org/pdf/1906.00446 (дата обращения: 05.11.2024).
4. ACVAE-VC: Non-parallel voice conversion with auxiliary classifier variational autoencoder / H. Kameoka [et al.] // IEEE/ACM Transactions on Audio, Speech, and Language Processing. 2019. Vol. 27, no. 9. P. 1432–1443.
5. CV-VAE: A compatible video VAE for latent generative video models / S. Zhao [et al.]. URL: https://www.researchgate.net/publication/381006549_CV-VAE_A_Compatible_Video_VAE_for_Latent_Generative_Video_Models (дата обращения: 02.11.2024).
6. Tab-VAE: A novel VAE for generating synthetic tabular data / S.M. Tazwar [et al.] // 13th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2024). Rome, 2024. P. 17–26.
7. A graph VAE and graph transformer approach to generating molecular graphs / J. Mitton [et al.]. URL: https://arxiv.org/pdf/2104.04345 (дата обращения: 28.10.2024).
8. D-VAE: A variational autoencoder for directed acyclic graphs / M. Zhang [et al.] // Advances in Neural Information Processing Systems. 2019. URL: https://www.academia.edu/74897067/D_VAE_A_Variational_Autoencoder_for_Directed_Acyclic_Graphs (дата обращения: 20.10.2024).
9. Исследование конфигураций нейросетевых кодеков для адаптивной системы сжатия кадров FPV-видеопотока при управлении беспилотными системами. Часть I. Методика / А.А. Березкин [и др.] // Электросвязь. 2024. № 9. С. 28–37.
10. Исследование конфигураций нейросетевых кодеков для адаптивной системы сжатия кадров FPV-видеопотока при управлении беспилотными системами. Часть II. Эксперимент / А.А. Березкин [и др.] // Электросвязь. 2024. № 10. С. 59–69.
11. Березкин А.А., Ченский А.А., Киричек Р.В. Исследование границ интенсивности видеопотока при FPV-управлении БПЛА в режиме предсказания кадров. Часть I: модели и методы // Вестник СибГУТИ. 2024. Т. 18, № 3. С. 115–139.
12. Березкин А.А., Паршин А.А., Лазарев А.А. Адаптивный контроль интенсивности видеопотока при передаче FPV-трафика беспилотных систем // 79-я научно-техническая конференция СПб НТО РЭС им. А.С. Попова, посвященная Дню радио: материалы региональной конференции. СПб.: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова, 2024. С. 158–161.
13. Hushchyn M., Ustyuzhanin A. Generalization of change-point detection in time series data based on direct density ratio estimation // Journal of Computational Science. 2021. Vol. 53. P. 101385. DOI: 10.1016/j.jocs.2021.101385
14. Hushchyn M., Arzymatov K., Derkach D. Online neural networks for change-point detection. URL: https://www.researchgate.net/publication/344485923_Online_Neural_Networks_for_Change-Point_Detection (дата обращения: 28.10.2024).
15. Katser I.D., Kozitsin V.O. Skoltech Anomaly Benchmark (SKAB) // Kaggle. 2020. URL: https://www.kaggle.com/dsv/1693952 (дата обращения: 04.11.2024).
16. Shewhart W.A., Walter A. Economic Control of Manufactured Product. Medison: D. Van Nostrand Company, 1931. 501 p.
17. Исследование методов квантования латентного пространства вариационного автокодировщика для кадров FPV видеопотока. Часть I / А.А. Березкин [и др.] // Электросвязь. 2024.

№ 6. С. 10–16.
18. Time Series Analysis: Forecasting and Control / G.E.P. Box [et al.]. 5th Ed. New Jersey: John Wiley & Sons, 2015. 720 p.
19. Бардасов С.А. Эконометрика: учебное пособие. 2-е издание, переработанное и дополненное // Тюмень: Издательство Тюменского государственного университета, 2010. 263 с.
20. Shumway R.H., Stoffer D.S. Time Series Analysis and Its Applications: With R Examples. 2th Ed. Springer, 2005. 575 p.
21. Tsay R.S. Analysis of Financial Time Series. 3th Ed. New Jersey: John Wiley & Sons, 2005. 677 p.
22. Архипелаг 2024. URL: https://xn--2035-43davo0a5a6bk9d.xn--p1ai/ (дата обращения: 07.11.2024).
23. PyTorch. URL: https://pytorch.org/ (дата обращения: 01.11.2024).
24. Choquette J., Gandhi W. Nvidia A100 GPU: Performance & innovation for GPU computing // 2020 IEEE Hot Chips 32 Symposium (HCS). URL: https://www.sci-hub.ru/10.1109/hcs49909.2020.9220622 (дата обращения: 15.10.2024).
25. Mushtaq R. Augmented dickey fuller test // SSRN Electronic Journal. 2011. DOI: http://dx.doi.org/10.2139/ssrn.1911068 25
26. Statsmodels. URL: https://www.statsmodels.org/stable/index.html (дата обращения: 27.10.2024).
27. A survey of forecast error measures / M.V. Shcherbakov [et al.] // World Applied Sciences Journal. 2013. Vol. 24, no. 24. P. 171–176.

Дата поступления 11.11.2024
Дата принятия 13.12.2024

Березкин Александр Александрович, к.т.н., доцент кафедры программной инженерии и вычислительной техники (ПИВТ) СПбГУТ. 193232, Российская Федерация, г. Санкт-Петербург, пр. Большевиков, 22-1. Тел. +7 921 791-61-20. E-mail: berezkin.aa@sut.ru

Ченский Александр Александрович, магистрант кафедры ПИВТ Санкт-Петербургского государственного университет им. проф. М.А. Бонч-Бруевича (СПбГУТ). 193232, Российская Федерация, г. Санкт-Петербург, пр. Большевиков, 22-1. Тел. +7 911 091-35-57. E-mail: chenskii.aa@sut.ru

Киричек Руслан Валентинович, д.т.н., ректор, профессор кафедры ПИВТ СПбГУТ. 193232, Российская Федерация, г. Санкт-Петербург, пр. Большевиков, 22-1. Тел. +7 812 305-12-00. E-mail: kirichek@sut.ru

UDC 519.246.8+004.032.26+311.2
DOI: 10.18469/ikt.2024.22.3.03

PREDICTION OF VARIATIONAL AUTOENCODER EXECUTION TIME BASED ON ARIMA FAMILY SERIES MODELS

Berezkin A.A., Chenskiy A.A., Kirichek R.V.
Bonch-Bruevich Saint Petersburg University of Telecommunications, Saint Petersburg, Russian Federation
E-mail: berezkin.aa@sut.ru, chenskii.aa@sut.ru, kirichek@sut.ru

Currently, neural networks of the «variational autoencoder» type are used in various fields, including image compression. In FPV control of UAVs, they are used as a part of neural network video stream codecs. Neural network codecs allow for a high degree of frame compression and transmitting FPV video streams over low-bandwidth channels. Execution time of variational autoencoders is inconsistent and depends on the equipment and its load level. In this article, the execution time of variational autoencoders is considered as a time series. Predictive models of the execution time of variational autoencoders are studied both at the stage of video stream encoding and at the stage of decoding. As the result, it is shown that ARIMA models does may not be used to predict variational autoencoders operating time. Prediction models for encoder and decoder execution time are also provided.

Keywords: *latent space, variational autoencoder, neural networks, ARIMA, neural codec, prediction, statistical model*

Berezkin Alexander Alexandrovich, Bonch-Bruevich Saint Petersburg State University of Telecommunications, 22-1, Bolshevikov Avenue, Saint Petersburg, 193232, Russian Federation; Associate Professor of Program Engineering and Computer Science Department, PhD in Technical Science. Tel. +7 921 791-61-20. E-mail: berezkin.aa@sut.ru

Chenskiy Alexander Alexandrovich, Bonch-Bruevich Saint Petersburg State University of Telecommunications, 22-1, Bolshevikov Avenue, Saint Petersburg, 193232, Russian Federation; Master's Degree Student of Program Engineering and Computer Science Department. Tel. +7 911 091-35-57. E-mail: chenskii.aa@sut.ru

Kirichek Ruslan Valentinovich, Bonch-Bruevich Saint Petersburg State University of Telecommunications, 22-1, Bolshevikov Avenue, Saint Petersburg, 193232, Russian Federation; Rector, Professor of Program Engineering and Computer Science Department, Doctor of Technical Science. Tel. +7 812 305-12-00. E-mail: kirichek@sut.ru

References

1. Kingma D.P., Welling M. Auto-encoding variational bayes. URL: https://arxiv.org/pdf/1312.6114 (accessed: 28.10.2024).
2. Doersch C. Tutorial on variational autoencoders. URL: https://www.researchgate.net/publication/304163568_Tutorial_on_Variational_Autoencoders (accessed: 05.11.2024).
3. Razavi A., Van den Oord A., Vinyals O. Generating diverse high-fidelity images with VQ-VAE-2. *Advances in Neural Information Processing Systems*, 2019. URL: https://arxiv.org/pdf/1906.00446 (accessed: 05.11.2024).
4. Kameoka H. et al. ACVAE-VC: Non-parallel voice conversion with auxiliary classifier variational autoencoder. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019, vol. 27, no. 9, pp. 1432–1443.
5. Zhao S. et al. CV-VAE: A compatible video VAE for latent generative video models. URL: https://www.researchgate.net/publication/381006549_CV-VAE_A_Compatible_Video_VAE_for_Latent_Generative_Video_Models (accessed: 02.11.2024).
6. Tazwar S.M. et al. Tab-VAE: A novel VAE for generating synthetic tabular data. *13th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2024)*. Rome, 2024, pp. 17–26.
7. Mitton J. et al. A graph VAE and graph transformer approach to generating molecular graphs. URL: https://arxiv.org/pdf/2104.04345 (accessed: 28.10.2024).
8. Zhang M. et al. D-VAE: A variational autoencoder for directed acyclic graphs. *Advances in Neural Information Processing Systems*, 2019. URL: https://www.academia.edu/74897067/D_VAE_A_Variational_Autoencoder_for_Directed_Acyclic_Graphs (accessed: 20.10.2024).
9. Berezkin A.A. et al. Research of neural network codec configurations for adaptive FPV video stream frame compression system when controlling unmanned systems. Part I. Methodology. *Elektrosvjaz'*, 2024, no. 9, pp. 28–37. (In Russ.)
10. Berezkin A.A. et al. Research of neural network codec configurations for adaptive FPV video stream frame compression system when controlling unmanned systems. Part II. Experiment. *Elektrosvjaz'*, 2024, no. 10, pp. 59–69. (In Russ.)
11. Berezkin A.A., Chenskiy A.A., Kirichek R.V. Research of video stream intensity limits in UAV FPV control in frame prediction mode. Part I: models and methods. *Vestnik SibGUTI*, 2024, vol. 18, no. 3, pp. 115–139. (In Russ.)
12. Berezkin A.A., Parshin A.A., Lazarev A.A. Video stream intensity control in unmanned systems FPV traffic transmission. *79 nauchno-technicheskaja konferencia SPb NTO RES im. A.S. Popova, posvjashennaja Dnju radio: materialy regional'noj konferencii*. Saint Petersburg: Sankt-Peterburgskij gosudarstvennyj elektrotekhnicheskij universitet «LETI» im. V.I. Ul'yanova, 2024, pp. 158–161. (In Russ.)
13. Hushchyn M., Ustyuzhanin A. Generalization of change-point detection in time series data based

on direct density ratio estimation. *Journal of Computational Science*, 2021, vol. 53, pp. 101385. DOI: 10.1016/j.jocs.2021.101385
14. Hushchyn M., Arzymatov K., Derkach D. Online neural networks for change-point detection. URL: https://www.researchgate.net/publication/344485923_Online_Neural_Networks_for_Change-Point_Detection (accessed: 28.10.2024).
15. Katser I.D., Kozitsin V.O. Skoltech Anomaly Benchmark (SKAB). *Kaggle*, 2020. URL: https://www.kaggle.com/dsv/1693952 (accessed: 04.11.2024).
16. Shewhart W.A., Walter A. *Economic Control of Manufactured Product*. Medison: D. Van Nostrand Company, 1931. 501 p.
17. Berezkin A.A. et al. Research of variational autoencoder latent space quantization methods for FPV video stream frames. Part I. *Elektrosvjaz', 2024,* no. 6, pp. 10–16. (In Russ.)
18. Box G.E.P. et al. *Time Series Analysis: Forecasting and Control*. 5th Ed. New Jersey: John Wiley & Sons, 2015. 720 p.
19. Bardasov S.A. *Ekonometrics*: Textbook. 2nd Ed. Tjumen': Izdatel'stvo Tjumenskogo gosudarstvennogo universiteta, 2010, 263 p. (In Russ.)
20. Shumway R.H., Stoffer D.S. *Time Series Analysis and Its Applications: With R Examples*. 2nd Ed. Springer, 2005, 575 p.
21. Tsay R.S. *Analysis of Financial Time Series*. 3th Ed. New Jersey: John Wiley & Sons, 2005, 677 p.
22. Archipelago 2024. URL: https://xn--2035-43davo0a5a6bk9d.xn--p1ai/ (accessed: 07.11.2024).
23. PyTorch. URL: https://pytorch.org/ (accesed: 01.11.2024).
24. Choquette J., Gandhi W. Nvidia A100 GPU: Performance & innovation for GPU computing. *2020 IEEE Hot Chips 32 Symposium (HCS).* URL: https://www.sci-hub.ru/10.1109/hcs49909.2020.9220622 (accessed: 15.10.2024).
25. Mushtaq R. Augmented dickey fuller test. *SSRN Electronic Journal*, 2011. DOI: http://dx.doi.org/10.2139/ssrn.1911068
26. Statsmodels. URL: https://www.statsmodels.org/stable/index.html (accessed: 27.10.2024).
27. Shcherbakov M.V. et al. A survey of forecast error measures. *World Applied Sciences Journal*, 2013, vol. 24, no. 24, pp. 171–176.

Received 11.11.2024
Accepted 13.12.2024