

Гибкие (anytime) алгоритмы поиска решений для интеллектуальных систем поддержки принятия решений реального времени (на примере задачи классификации образов)

А.П. Еремеев ^{1✉}, С.М. Башкова ¹

¹ Национальный исследовательский университет «Московский энергетический институт», г. Москва, 111250, Россия

Ссылка для цитирования

Еремеев А.П., Башкова С.М. Гибкие (anytime) алгоритмы поиска решений для интеллектуальных систем поддержки принятия решений реального времени (на примере задачи классификации образов) // Программные продукты и системы. 2025. Т. 38. № 2. С. 181–187. doi: 10.15827/0236-235X.150.181-187

Информация о статье

Группа специальностей ВАК: 1.2.1, 2.3.3

Поступила в редакцию: 15.11.2024

После доработки: 26.12.2024

Принята к публикации: 14.01.2025

Аннотация. Актуальной задачей при разработке методов и инструментальных программных средств конструирования современных и перспективных интеллектуальных систем поддержки принятия решений реального времени, функционирующих в условиях достаточно жестких временных ограничений, является разработка так называемых гибких (anytime) алгоритмов поиска решений. Данные алгоритмы способны с некоторого момента времени выдавать приемлемое решение, постепенно улучшая его вплоть до получения оптимального решения, при соответствующем увеличении компьютерных ресурсов (как правило, времени). Целями данной работы являются исследование и разработка гибких алгоритмов поиска решений для применения их в интеллектуальных системах поддержки принятия решений реального времени при достаточно жестких ограничениях на время реагирования при возникновении проблемных (аварийных, нештатных и т.п.) ситуаций. Рассматриваются алгоритмы на основе нейросетевого подхода на примере решения задачи классификации образов (изображений). Дается сравнительный анализ нейронных сетей, использующих так называемый метод раннего выхода для решения такой задачи, и предлагается оригинальная anytime-модификация нейросети, позволяющая получить более раннее решение по классификации образов (изображений), чем при применении классического подхода, что актуально для систем реального времени. Описываются программная реализация, выполненная с применением фреймворков Tensorflow и Keras языка программирования Python, и результаты компьютерного моделирования, подтверждающие перспективность предложенного подхода. Исследования и разработки выполняются в рамках создания базовых инструментальных средств построения интеллектуальных систем поддержки принятия решений реального времени для помощи оперативно-диспетчерскому персоналу (лицам, принимающим решения) при управлении, мониторинге и диагностике сложных технических и организационных систем, а также при распознавании и классификации проблемных ситуаций.

Ключевые слова: искусственный интеллект, гибкий (anytime) алгоритм, нейронная сеть, интеллектуальная система, реальное время, распознавание образов, поддержка принятия решений

Благодарности. Работа выполнена при финансовой поддержке РФФИ, проект № 24-11-00285, <https://rscf.ru/project/24-11-00285/>

Введение. В процессе разработки инструментальных средств для построения интеллектуальных систем поддержки принятия решений реального времени (ИСППР РВ), функционирующих при достаточно жестких временных ограничениях, специалистам приходится решать важные задачи минимизации задержек и максимизации точности алгоритма поиска решения. В качестве одного из возможных решений этой проблемы предлагается использовать так называемые гибкие (anytime) алгоритмы. Анализ и применение anytime-алгоритмов рассматриваются на примере задач компьютерного зрения реального времени, а именно задачи классификации образов (изображений). Данная работа является продолжением исследований авторов по anytime-алгоритмам для

интеллектуальных систем реального времени, начатых в [1, 2].

Anytime-алгоритмы – это алгоритмы, способные выдавать промежуточные результаты, продолжая уточнять полученное решение при увеличении временных и вычислительных ресурсов вплоть до получения оптимального решения [2]. Они позволяют находить быстрое решение за малое время и повысить его точность по мере работы алгоритма, что может быть весьма важным в системах реального времени. Следует отметить, что в современных научных публикациях данным алгоритмам, несмотря на их актуальность, практически не уделяется внимание. Основная задача проектирования нейросетевых anytime-алгоритмов – повысить точность промежуточных прогнозов

без ущерба для конечного выхода. С этой целью в работе дается сравнительный анализ различных иерархических архитектур нейронных сетей, особенно архитектуры с ранними выходами с использованием взвешенной функции потерь.

Проводится сравнительный анализ нейронных сетей, использующих метод раннего выхода для решения задач классификации, и предлагается оригинальная anytime-модификация нейросети с ранними выходами для задачи классификации образов. Алгоритмы подобного типа могут быть использованы в ИСППР РВ при решении задач компьютерного зрения для генерации быстрых прогнозов и их последующего уточнения.

Сети раннего выхода

Архитектура нейронной сети на основе методов раннего выхода (*Early-Exit Network, EEN*) [3–5], помимо основной нейронной сети глубокого обучения, включает в себя также внутренние классификаторы (*Internal Classifier, IC*) на промежуточных уровнях, что позволяет выдавать прогнозы от более раннего IC, если они уже достигли требуемой точности. Методы раннего выхода в задаче классификации изображений используют тот факт, что некоторые образцы данных классифицируются проще, чем другие, а значит, могут быть обработаны только частью сети, что сокращает вычислительные затраты. Было установлено, что этот метод не только сокращает объем вычислений, но и может повысить точность класси-

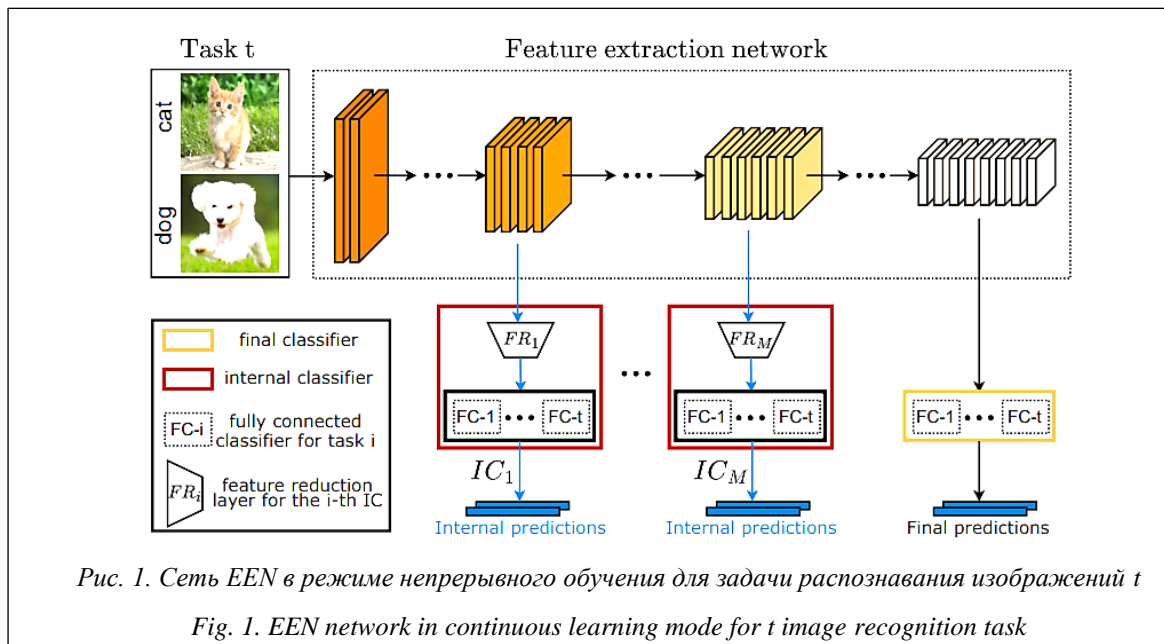
фикации, предотвращая чрезмерное обучение сети на простых образцах: явление, когда образцы, которые могут быть правильно классифицированы в ранних IC, распространяются дальше по сети и могут быть неправильно классифицированы в более поздних IC.

Пример сети EEN для задачи классификации образов (изображений) приведен на рисунке 1 [5]. Это классическая нейронная сеть с M внутренними классификаторами IC, подключенными на промежуточных уровнях сети. Она классифицирует входные данные постепенно на каждом этапе внутренней классификации. Во время вывода, если достоверность прогнозов IC превышает пороговое значение, сеть EEN возвращает прогноз и пропускает остальные вычисления. Классификатор IC состоит из слоя сокращения признаков (*Feature Reduction, FR*), который уменьшает размер входных данных, и полносвязного слоя-классификатора (*Final Classifier, FC*), генерирующего прогнозы. Как видно из рисунка 1, на разных этапах работы сети подключаются классификаторы IC, состоящие из слоя классификаторов FR и многоуровневого полносвязного классификатора класса FC.

Целевая функция при полном обучении сети EEN с подобной архитектурой – взвешенная сумма потерь для каждого классификатора [5], которая задается формулой

$$\arg \min_{\theta} \sum_{i=1}^{M+1} w_i * L_i(C_i(E_i(X_i; \theta_i)), Y_i),$$

где C_i – i -й классификатор, обученный в ходе решения задачи t , $i = 1, \dots, M$ – внутренние



классификаторы, $M + 1$ – конечный классификатор сети; L_i – функция потерь i -го классификатора; w_i – вес, присвоенный потере i -го классификатора; E_i – вектор признаков, полученный в результате вывода части сети с параметрами θ_i ; X_i – входные данные; Y_i – фактические значения.

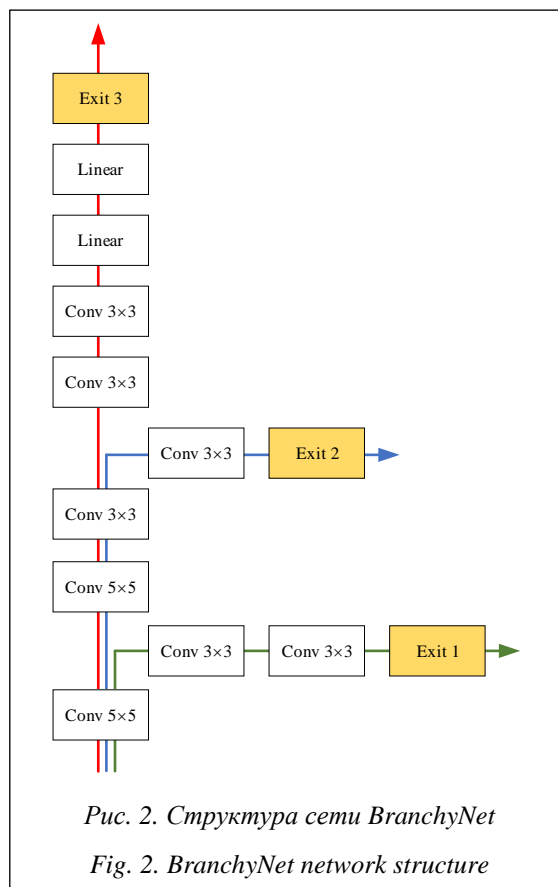
Обучение такой сети может быть как полным, со всеми классификаторами и обновлением всех параметров θ , так и частичным, подразумевающим только обучение внутренних классификаторов, добавленных к предварительно обученной основе. Для этого замораживаются веса основной сети, следовательно, ее прогнозы остаются прежними, а обновление весов происходит только во внутренних классификаторах. Такое обучение происходит быстро, однако имеет недостаток: так как стандартное обучение направлено только на повышение точности окончательного прогнозирования, в результате применения внутренних классификаторов поверх предварительно обученных слоев получаются относительно слабые промежуточные прогнозы.

Сеть BranchyNet

В архитектуре нейронной сети BranchyNet к основной ветви исходной нейросети добавляются боковые ветви для обеспечения раннего выхода на определенных тестовых образцах [6]. Принцип работы такой сети основан на том, что информация, полученная на более ранних этапах работы сетью глубокого обучения, позволяет корректно классифицировать большую часть совокупности данных. Используя эти выборки с прогнозированием на более ранних этапах и таким образом избегая послышной обработки для всех уровней, сеть Branchy-Net значительно сокращает время выполнения и затраты аппаратных ресурсов для классификации большинства образцов. На рисунке 2 [6] показано, как BranchyNet модифицирует стандартную нейросетевую архитектуру AlexNet, добавляя две ветви с соответствующими точками выхода. Эти ветви раннего выхода позволяют выборкам, которые могут быть точно классифицированы на ранних стадиях развития сети, завершать работу на этих этапах.

При проектировании сети BranchyNet необходимо учитывать ряд факторов:

- расположение точек ответвления;
- структуру ответвления (весовые уровни, полносвязные слои и т.д.), а также его размер и глубину;
- классификатор в точке выхода из ответвления;



– критерии выхода из ответвления и связанные с этим затраты на проверку в соответствии с критериями;

– обучение классификаторов в точках выхода из всех ветвей.

В общем виде понятие ветвей может применяться рекурсивно, то есть сама ветвь может иметь ответвления, что приводит к древовидной структуре сети. Помимо задач классификации, эта архитектура может также применяться к другим задачам, таким как сегментация изображений и обнаружение объектов.

Для задачи классификации в качестве цели оптимизации обычно используется функция перекрестных потерь энтропии с функцией softmax. В BranchyNet она используется следующим образом. Пусть y – вектор меток однозначной истинности, x – входная выборка, а C – набор всех возможных меток. Целевую функцию можно записать в виде

$$L(\hat{y}; y; \theta) = -\frac{1}{|C|} \sum_{c \in C} y_c \log \hat{y}_c,$$

где

$$\hat{y} = \text{soft max}(z) = \frac{\exp(z)}{\sum_{c \in C} \exp(z_c)},$$

$$z = f_{\text{exit}_n}(x; \theta),$$

где f_{exit_n} – выходные данные n -й выходной ветви; а θ представляет параметры слоев от точки входа до точки выхода [6].

Целью проектирования каждой выходной ветви является минимизация этой функции потерь. Чтобы обучить еще и всю сеть BranchyNet, формируется совместная оптимизационная задача в виде взвешенной суммы функций потерь каждой выходной ветви:

$$L_{branchnet}(\hat{y}, y; \theta) = \sum_{n=1}^N w_n L(\hat{y}_{exit_n}, y; \theta),$$

где N – общее количество точек выхода; w_n – значение каждой из потерь.

После обучения разветвленную сеть можно использовать для быстрого вывода, классифицируя выборки на более ранних этапах в сети на основе алгоритма, представленного на рисунке 3 [6]. Если классификатор в точке выхода из ветви имеет высокую степень уверенности в правильности маркировки тестовой выборки x , то выборка завершается и возвращает прогнозируемую метку на ранней стадии без дальнейших вычислений, выполняемых вышестоящими ветвями в сети. В качестве показателя

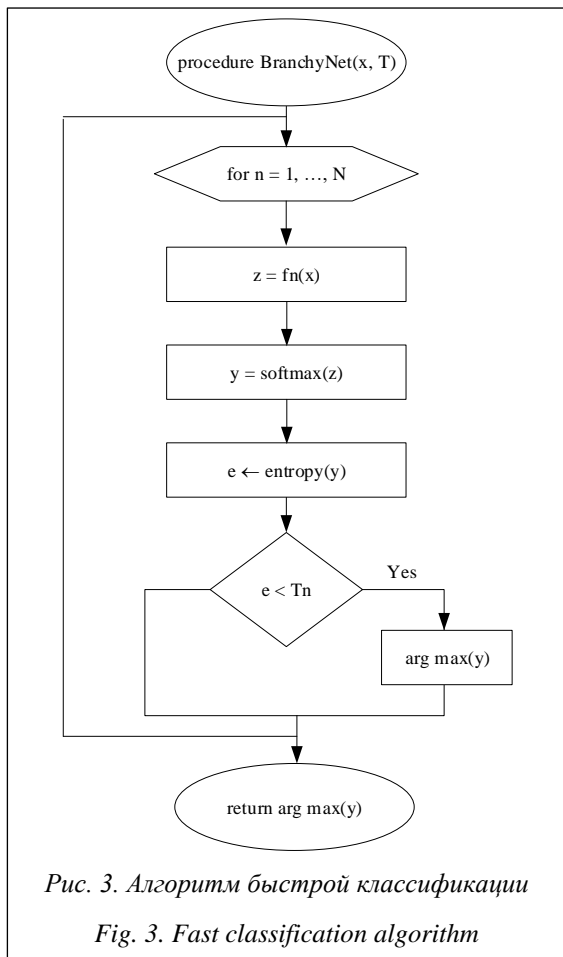


Рис. 3. Алгоритм быстрой классификации

Fig. 3. Fast classification algorithm

уверенности классификатора в выборке используется энтропия:

$$entropy(y) = \sum_{c \in C} y_c \log y_c,$$

где y – вектор, содержащий вычисленные вероятности для всех возможных меток классов, а C – набор всех возможных меток. Чтобы выполнить быстрый вывод в сети BranchyNet, используется данный алгоритм.

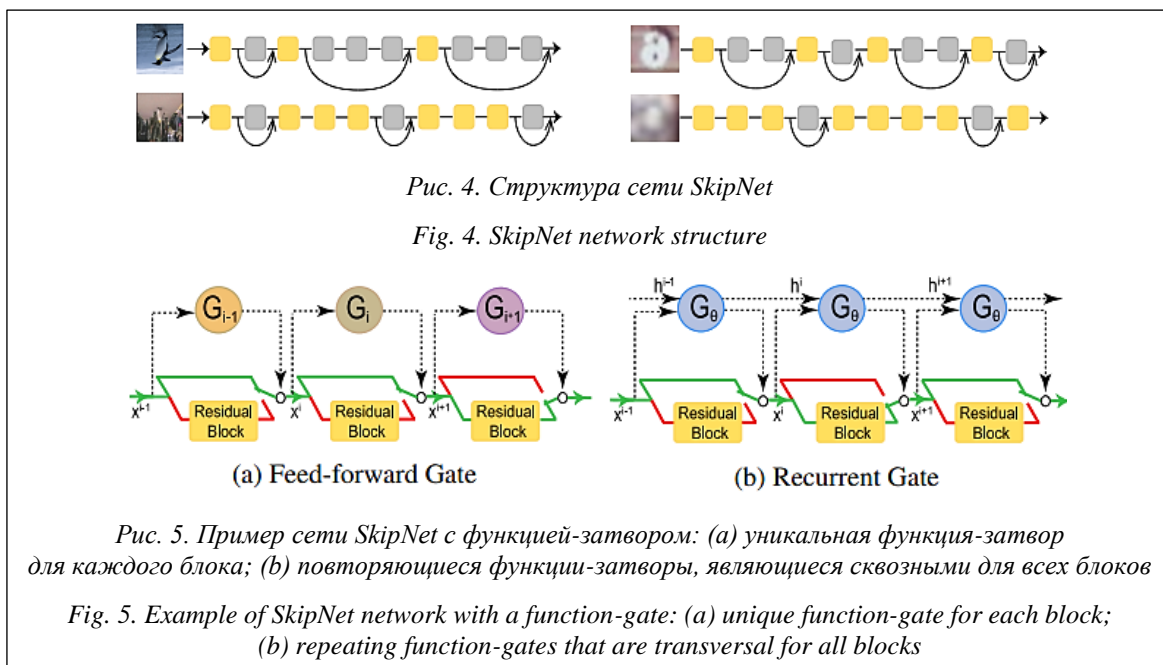
Для выполнения алгоритма требуется вектор T , n -й элемент которого ($n = 1, 2, \dots, N$) является пороговым значением, используемым для определения того, должен ли вход x завершаться в n -й точке выхода. Алгоритм начинается с самой низкой точки выхода и повторяется до самой высокой и конечной точки выхода сети. Для каждой точки выхода входная выборка подается через соответствующую ветвь. Затем вычисляются функция softmax и энтропия выходных данных и проверяется, находится ли энтропия ниже порогового значения точки выхода T_n . Если энтропия меньше T_n , то возвращается метка класса с максимальной оценкой (вероятностью). В противном случае выборка продолжается до следующей точки выхода. Если образец достигает последней точки выхода, то всегда возвращается метка с максимальной оценкой.

Сеть SkipNet

Сеть SkipNet, изображенная на рисунке 4 [7], представляет собой модификацию остаточных нейронных сетей (ResNet), которая динамически выбирает, какие слои сверточной нейронной сети следует пропустить при выводе. SkipNet учится пропускать слои в зависимости от входных данных.

Задача динамического пропуска рассматривается как последовательное принятие решений, в котором выходные данные предыдущих уровней используются для принятия решения о том, следует ли обходить последующий уровень. Основной задачей динамического пропуска является пропуск наибольшего возможного количества слоев при сохранении точности всей сети.

SkipNet представляет собой сверточную нейросеть, в которой отдельные слои выборочно включаются или исключаются для конкретного входного сигнала. Выбор слоев для каждого входного сигнала осуществляется с помощью достаточно простых функций-затворов, которые размещаются между слоями (рис. 5 [7]). Функции-затворы G^i преобразуют



входные данные предыдущего уровня или группы уровней в двоичное решение о выполнении или пропуске последующего слоя или группы слоев.

Более точно, пусть x^i – вход, $F^i(x^i)$ – выход i -го слоя или группы слоев сети, тогда выход слоя-затвора будет определяться следующим образом [7]:

$$x^{i+1} = G^i(x^i)F^i(x^i) + (1 - G^i(x^i))x^i,$$

где $G^i(x^i) \in \{0, 1\}$ – функция-затвор для слоя i .

Реализация anytime-нейросети для задачи распознавания изображений

С помощью библиотек Keras (<https://keras.io/api>; <https://pymodelchecking.readthedocs.io/en/latest>), TensorFlow (<https://www.tensorflow.org>) и языка программирования Python реализована и обучена сверточная anytime-нейросеть на примере датасета CIFAR-10 (<https://www.cs.toronto.edu/~kriz/cifar.html>). Набор данных CIFAR-10 состоит из 60 000 цветных изображений 32×32 в 10 классах, по 6 000 изображений в каждом классе. Имеются 50 000 обучающих изображений и 10 000 тестовых.

Реализованная сеть типа Early-Exit ResNet (<http://www.swsys.ru/uploaded/image/2025-2/11.jpg>) основана на модели остаточной нейронной сети ResNet с добавлением двух промежуточных выходов. Сеть состоит из начального сверточного слоя (Conv2D), шести остаточных блоков, каждый из которых включает два блока, состоящих из сверточного слоя, слоя пакетной нормализации (BatchNormalization) и актива-

ции ReLU и завершающего блока из слоя выбора среднего значения (GlobalAveragePooling2D) и полносвязного слоя (Dense). Для предотвращения переобучения в сверточных слоях сети использована L2-регуляризация. Такая конструкция способствует решению проблемы затухающих градиентов в глубоких слоях сети и повышает точность классификации.

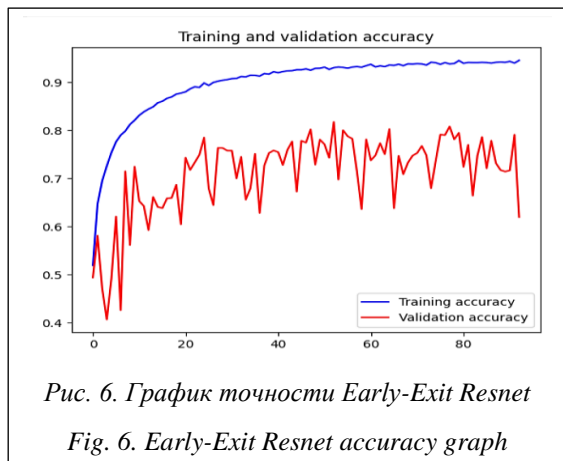
Для компьютерной реализации сети Early-Exit ResNet были выбраны следующие параметры: optimizer – Adam; loss – категориальная кросс-энтропия (categorical_crossentropy); loss_weights – первый выход: 0.3, второй выход: 0.3, главный выход: 1.0 (значение потерь, которое будет минимизировано моделью, представляет собой взвешенную сумму всех отдельных потерь с коэффициентами, определенными этим параметром); metrics – метрика accuracy; epoch – заданное количество 100, регулируется функцией EarlyStopping с параметром остановка в 40 эпох, чтобы наглядно посмотреть, где начнется переобучение, и сохранить веса модели с лучшим результатом с помощью метода ModelCheckpoint.

Обучение сети было остановлено на 53-й эпохе (рис. 6) со следующими значениями точности (в процентах):

76,5 – точность первого раннего выхода сети на обучающей выборке;

92 – точность второго раннего выхода сети на обучающей выборке;

93 – точность главного выхода сети на обучающей выборке;



65,6 – точность первого раннего выхода сети на тестовой выборке;

81,5 – точность второго раннего выхода сети на тестовой выборке;

81,7 – точность главного выхода сети на тестовой выборке.

Данная реализация сети раннего выхода Early-Exit ResNet на основе anytime-алгоритма показала достаточно высокие значения точности на тестовой выборке и, что важно, высокую точность прогнозов ранних выходов. Сеть также оказалась устойчивой к переобучению,

что существенно экономит время поиска решения и обосновывает ее применение в ИС РВ типа ИСППР РВ.

Заключение

В работе были рассмотрены anytime-модификации сверточных нейронных сетей для задачи распознавания образов. Описаны структуры сверточных сетей с ранними выходами, ветвящейся структурой и пропуском слоев. Приведены формулы функций потерь для обучения anytime-архитектур. Anytime-нейросеть Early-Exit ResNet разработана и обучена на множестве CIFAR-10 с помощью фреймворков Keras и TensorFlow языка программирования Python. Предложенная сеть показывает хорошую точность на промежуточных прогнозах без ущерба для итогового прогноза.

Данные исследования и разработки выполняются в плане реализации базовых инструментальных (математических и программных) средств для конструирования ИСППР РВ и ИС РВ в целом для классификации, диагностики и мониторинга сложных технологических и организационных объектов и процессов.

Список литературы

1. Еремеев А.П., Митрофанов Д.А. Гибкий алгоритм моделирования иерархических рассуждений для систем реального времени. В кн.: Интеллектуальные системы. Коллективная монография. Вып. 5. М.: Физматлит, 2011. С. 85–110.
2. Еремеев А.П., Башкова С.М. Реализация гибких (anytime) алгоритмов поиска решения для интеллектуальных систем реального времени // Тр. Междунар. науч.-технич. конгресса «ИС&ИТ-2024». 2024. Т. 1. С. 236–243.
3. Rahmath P.H., Srivastava V., Chaurasia K., Pacheco R.G., Couto R.S. Early-exit deep neural network – a comprehensive survey. ACM Computing Surveys, 2024, vol. 57, no. 3, art. 75.
4. Zniber A., Karrakchou O., Ghogho M. Dynamic early exiting predictive coding neural networks. ArXiv, 2023, art. 2309.02022. URL: <https://arxiv.org/abs/2309.02022> (дата обращения: 10.10.2024).
5. Szatkowski F., Yang F., Twardowski B., Trzcinski T. Accelerated inference and reduced forgetting: The dual benefits of early-exit networks in continual learning. Proc. ICLR, 2024, pp. 1–16.
6. Teerapittayanon S., McDanel B., Kung H.T. BranchyNet: Fast inference via early exiting from deep neural networks. Proc. ICPR, 2016, pp. 2464–2469. doi: 10.1109/ICPR.2016.7900006.
7. Wang X., Yu F., Dou Z., Darrel T., Gonzales J.E. SkipNet: Learning dynamic routing in convolutional networks. In: LNIP. Proc. ECCV, 2018, vol. 11217, pp. 409–424. doi: 10.1007/978-3-030-01261-8_25.

For citation

Ereemeev, A.P., Bashkova, S.M. (2025) ‘Anytime algorithms for intelligent real-time decision support systems (in terms of a pattern classification task)’, *Software & Systems*, 38(2), pp. 181–187 (in Russ.). doi: 10.15827/0236-235X.150.181-187

Article info

Received: 15.11.2024

After revision: 26.12.2024

Accepted: 14.01.2025

Abstract. A relevant task for developing methods and software tools for designing modern and prospective intellectual systems of real-time decision support operating under fairly strict time constraints is the development of flexible (anytime) algorithms for decision search. Such algorithms are capable of producing an acceptable solution from a certain point in time, gradually improve it until the optimal solution is obtained with a corresponding increase in computer resources (usually time). The aim of this work is to study and develop flexible decision search algorithms to apply them in real-time intelligent decision support systems under strict restrictions on the response time when problematic (emergency, abnormal, etc.) situations arise. The authors consider algorithms based on the neural network approach on the example of solving the problem of pattern (image) classification. They give a comparative analysis of neural networks using the early output method to solve such problem. They propose an original anytime modification of the neural network that allows obtaining an earlier solution for pattern (image) classification compared to the classical approach. This is relevant for real-time systems. The authors describe the software implementation based on the Tensorflow and Keras frameworks of the Python programming language, as well as the results of computer modeling, which confirm that the proposed approach is promising. The authors carry out research and development in terms of creating basic tools for building intelligent real-time decision support systems to help operational and dispatch personnel (decision makers) in controlling, monitoring and diagnosing complex technical and organizational systems, as well as in recognizing and classifying problem situations.

Keywords: artificial intelligence, anytime algorithm, neural network, intelligent system, real time, pattern recognition, decision support

Acknowledgements. The work was financially supported by RSF, project no. 24-11-00285, <https://rscf.ru/project/24-11-00285/>

References

1. Ereemeev, A.P., Mitrofanov, D.A. (2011) 'Flexible algorithm for modeling hierarchical reasoning for real-time systems', in *Intelligent Systems. Collective Monograph*, (5), pp. 85–110 (in Russ.).
2. Ereemeev, A.P., Bashkova, S.M. (2024) 'Implementation of flexible (anytime) algorithms for finding solutions for intelligent real-time systems', *Proc. Int. Sci. and Tech. Congress "IS&IT'24"*, 1, pp. 236–243 (in Russ.).
3. Rahmath, P.H., Srivastava, V., Chaurasia, K., Pacheco, R.G., Couto, R.S. (2024) 'Early-exit deep neural network – a comprehensive survey', *ACM Computing Surveys*, 57(3), art. 75.
4. Zniber, A., Karrakchou, O., Ghogho, M. (2023) 'Dynamic early exiting predictive coding neural networks', *ArXiv*, art. 2309.02022, available at: <https://arxiv.org/abs/2309.02022> (accessed October 10, 2024).
5. Szatkowski, F., Yang, F., Twardowski, B., Trzciński, T. (2024) 'Accelerated inference and reduced forgetting: The dual benefits of early-exit networks in continual learning', *Proc. ICLR*, pp. 1–16.
6. Teerapittayanon, S., McDanel, B., Kung, H.T. (2016) 'BranchyNet: Fast inference via early exiting from deep neural networks', *Proc. ICPR*, pp. 2464–2469. doi: 10.1109/ICPR.2016.7900006.
7. Wang, X., Yu, F., Dou, Z., Darrel, T., Gonzales, J.E. (2018) 'SkipNet: Learning dynamic routing in convolutional networks', in *LNIP. Proc. ECCV*, 11217, pp. 409–424. doi: 10.1007/978-3-030-01261-8_25.

Авторы

Еремеев Александр Павлович¹, д.т.н.,
профессор, eremeev@appmat.ru
Башкова София Михайловна¹,
инженер, BashkovaSM@mpei.ru

Authors

Aleksandr P. Ereemeev¹, Dr.Sci. (Engineering),
Professor, eremeev@appmat.ru
Sofiya M. Bashkova¹,
Engineer, BashkovaSM@mpei.ru

¹ Национальный исследовательский университет
«Московский энергетический институт»,
г. Москва, 111250, Россия

¹ National Research University "MPEI",
Moscow, 111250,
Russian Federation