Оригинальная статья / Original article

УДК 004.415.25

https://doi.org/10.21869/2223-1560-2025-29-3-86-98

CC BY 4.0

Применение глубокого обучения сверточной нейронной сети для классификации жестов из набора данных Sign Language MNIST

М.В. Бобырь [1] ✉, А.А. Асеев [1]

[1] Юго-Западный государственный университет
ул. 50 лет Октября, д. 94, г. Курск 305040, Российская Федерация

✉ e-mail: fregat_mn@rambler.ru

Резюме

Цель исследования. *Задача распознавания жестов в системах компьютерного зрения имеет важное значение для разработки доступных интерфейсов взаимодействия человека с компьютером, в том числе и для людей с ограниченными возможностями. Традиционные методы, например использование ручного выделения признаков (HOG, SIFT) в сочетании с классификаторами типа SVM, обладают ограниченной точностью и чувстви-тельны к изменениям освещения, фона и позы руки. Целью данной работы является построение и обучение сверточной нейронной сети (CNN) для эффективной классификации жестов на основе набора данных Sign Language MNIST. В рамках исследования решались задачи предобработки данных, проектирования архитектуры модели, её обучения и оценки качества распознавания на тестовом наборе.*

Методы. *Использовались библиотеки TensorFlow и Keras для реализации CNN. Модель включает сверточ-ные слои для извлечения локальных признаков, слой Flatten для векторизации, полносвязные слои с функ-цией активации ReLU и выходной слой с Softmax. Обучение проводилось с использованием оптимизатора Adam и функции потерь sparse_categorical_crossentropy на 27 455 изображениях, тестирование — на 7 172 примерах.*

Результаты. *Предложенная модель достигла точности 89,14 % на тестовом наборе данных после 18 эпох обучения, что превосходит результаты традиционных методов (HOG + SVM – 70,1 %) и простых нейронных сетей (78,4 %).*

Заключение. *Применение сверточных нейронных сетей для классификации жестов является эффективным подходом, обеспечивающим высокую точность и устойчивость к вариациям входных данных, что делает его перспективным для задач компьютерного зрения и разработки систем жестового взаимодействия.*

Ключевые слова: *нейронная сеть; сверточная нейронная сеть; полносвязный слой; функция активации; функция потерь; Sign Language MNIST; CPU; GPU.*

Конфликт интересов: *Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

© Бобырь М.В., Асеев А.А., 2025

Для цитирования: Бобырь М.В., Асеев А.А. Применение глубокого обучения сверточной нейронной сети для классификации жестов из набора данных Sign Language MNIST // Известия Юго-Западного государственного университета. 2025; 29(3): 86-98. https://doi.org/10.21869/-2223-1560-2025-29-3-86-98.

Поступила в редакцию 14.05.2025 *Подписана в печать 23.06.2025* *Опубликована 30.09.2025*

Applying deep learning convolutional neural network to classify gestures from MNIST Sign Language dataset

Maksim V. Bobyr [1] ✉, Artem A. Aseev [1]

[1] Southwest State University
50 Let Oktyabrya str. 94, Kursk 305040, Russian Federation

✉ e-mail: fregat_mn@rambler.ru

Abstract

Relevance. *Gesture recognition in computer vision systems is important for the development of accessible human-computer interaction interfaces, including for people with disabilities. Traditional methods, such as manual feature extraction (HOG, SIFT) in combination with SVM classifiers, have limited accuracy and are sensitive to changes in lighting, background, and hand pose.*

Purpose of research. *The aim of this work is to build and train a convolutional neural network (CNN) for efficient gesture classification based on the Sign Language MNIST dataset. The study addressed the problems of data preprocessing, model architecture design, training, and recognition quality assessment on the test set.*

Methods. *TensorFlow and Keras libraries were used to implement the CNN. The model includes convolutional layers for local feature extraction, a Flatten layer for vectorization, fully connected layers with a ReLU activation function, and an output layer with Softmax. The training was performed using the Adam optimizer and the sparse_categorical_crossentropy loss function on 27,455 images, and testing was performed on 7,172 examples.*

Results. *The proposed model achieved 89.14% accuracy on the test dataset after 18 training epochs, which outperforms traditional methods (HOG + SVM - 70.1%) and simple neural networks (78.4%).*

Conclusion. *The use of convolutional neural networks for gesture classification is an effective approach that provides high accuracy and is robust to variations in input data, making it promising for computer vision and gesture interaction systems.*

Keywords: *neural network; convolutional neural network; fully connected layer; activation function; loss function; Sign Language MNIST; CPU; GPU.*

Conflict of interest. *The Authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.*

For citation: Bobyr M. V., Aseev A. A. Applying deep learning convolutional neural network to classify gestures from MNIST Sign Language dataset. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the South-west State University*. 2025; 29(3): 86-98 (In Russ.). https:// doi.org/10.21869/2223-1560-2025-29-3-86-98.

Received 14.05.2025 *Accepted 23.06.2025* *Published 30.09.2025*

***

Введение

Сверточные нейронные сети (CNN) представляют собой мощный инструмент глубокого обучения, широко применяемый для задач компьютерного зрения, включая классификацию изображений. Впервые предложенные Яном Лекуном

в 1988 году [1], CNN изначально требовали значительных вычислительных ресурсов, что ограничивало их использование до появления графических процессоров (GPU). С 2012 года, после победы AlexNet в конкурсе ImageNet [2], сверточные сети начали доминировать в задачах распознавания образов, вытесняя классические методы машинного обучения [3, 4].

Классификация жестов, таких как в наборе данных Sign Language MNIST, является одной из актуальных задач компьютерного зрения, находящей применение в системах управления устройствами, интерфейсах для людей с ограниченными возможностями и образовательных платформах. Однако традиционные подходы к решению этой задачи, такие как методы на основе ручного выделения признаков (например, гистограммы ориентированных градиентов (HOG) или SIFT [5]), демонстрируют ограниченную точность и чувствительность к вариациям освещения и фона. Например, исследование показало, что классификаторы, использующие HOG в сочетании с алгоритмом SVM, достигают точности лишь около 70% на аналогичных наборах данных жестов, что значительно ниже современных CNN [6].

Другой альтернативой являются простые полносвязные нейронные сети, которые, несмотря на свою универсальность, требуют большого количества параметров и склонны к переобучению на задачах с изображениями. В работе Ли и др. было продемонстрировано, что полносвязная сеть с двумя скрытыми слоями достигает точности около 78% на наборе Sign Language MNIST, но требует значительно больше времени на обучение и плохо обобщает на новых данных [7]. Кроме того, такие сети не способны эффективно извлекать локальные признаки, что делает их менее подходящими для обработки изображений по сравнению с CNN, которые используют сверточные слои для автоматического выделения иерархических признаков [8].

В данной статье предлагается подход, основанный на применении сверточной нейронной сети, реализованной с использованием библиотек TensorFlow и Keras, для классификации жестов из набора данных Sign Language MNIST. Этот набор включает 34 627 изображений жестов, представляющих буквы американского жестового языка (ASL), и разделен на 27 455 обучающих и 7 172 тестовых примера. Предложенная модель использует сверточные слои для извлечения признаков и полносвязные слои для классификации, что позволяет достичь высокой точности (89.1% на тестовом наборе) за 18 эпох обучения. Такой подход превосходит традиционные методы и простые нейронные сети по точности, скорости обучения и устойчивости к вариациям данных, обеспечивая эффективное решение задачи классификации жестов.

Материалы и методы

В качестве набора данных использован Sign Language MNIST, который

включает 34 627 изображений жестов рук, представляющих буквы алфавита американского жестового языка (ASL). Из них 27 455 изображений используются для обучения, а 7 172 – для тестирования. Все изображения имеют размер 28x28 пикселей в оттенках серого и принадлежат одной из 24 категорий, показанных на рис. 1.

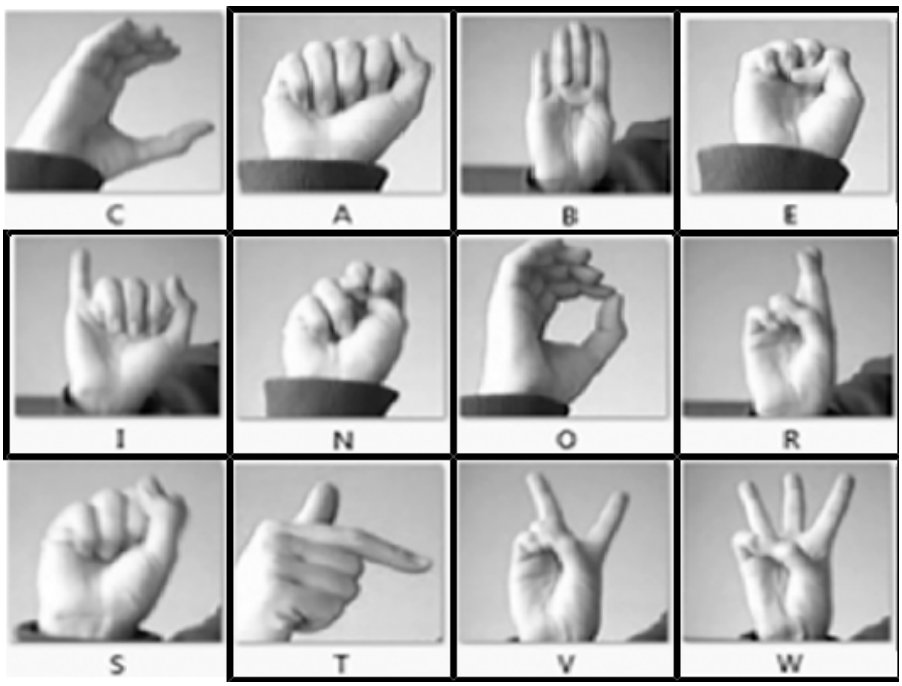

Рис. 1. Категории жестов из датасета Sign Language MNIST

Fig. 1. Gesture Categories from the MNIST Sign Language Dataset

На рис. 1 приведена часть (12) категорий символов, каждый из которых соответствует своему классу: 2 – C; 0 – A; 1 – B; 4 – E; 8 – I; 13 – N; 14 – O; 17 – R; 18 – S; 19 – T; 21 – V; 22 – W.

Набор данных, частично состоящий из перечисленных категорий жестов, загружается автоматически с использованием TensorFlow [9]:

```
Python:
sign_language_mnist =
tf.keras_datasets_sign_language_mnist
(train_images, train_labels),
(test_images, test_labels) =
sign_language_mnist_load_data()
```

Этап предобработки данных представлен процессом нормализации. Под нормализацией подразумевается процесс приведения входных значений к единому масштабу, что ускоряет сходимость градиентного спуска и стабилизирует процесс обучения [10]. В нейросетях нормализация особенно важна, так как разные диапазоны значений входных данных могут привести к нестабильности весов и увеличению времени обучения [11].

Нормализация применяется к изображениям, текстовым данным и числовым признакам. В случае изображений, как в Sign Language MNIST, пиксельные значения находятся в диапазоне

[0, 255], и их нормализуют в диапазон [0, 1] или [-1, 1].

В коде нормализация описывается следующим образом:

```
Python:
train_images = train_images / 255.0
test_images = test_images / 255.0
```

Такой метод основан на мин-макс нормализации, которая определяется выражением:

$$X_{flat} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad (1)$$

где X – исходное изображение пикселя (от 0 до 255); $X_{\min}$ – минимальное значение в исходном диапазоне (0); $X_{\max}$ – максимальное значение в исходном диапазоне (255).

Это означает, что все значения приводятся к диапазону [0, 1], что помогает модели быстрее обучаться и снижает вероятность возникновения нестабильных весов.

Архитектура нейронной сети включает в себя следующие слои:

`Flatten()`: Преобразует двумерное изображение размерностью 28*28 пикселей в одномерный вектор длиной 784 элементов. Преобразование Flatten слоя описывается следующей формулой:

$$X_{\text{norm}}=R\text{i}, i = 1...784. \quad (2)$$

`Dense(512, activation = 'relu')`: Полносвязный слой с 512 нейронами и функцией активации ReLU. Функция обработки слоя определяется следующим выражением:

$$b_1 = ReLU(W_1X_{flat} + b_1), \quad (3)$$

где X – входной вектор (X_{flat} после Flatten); W – матрица весов слоя; b – вектор смещения; ReLU – функция активации; h – выходной вектор после слоя.

Использование функции активации ReLU (Rectified Linear Unit) в скрытых слоях позволяет нейронной сети обучаться быстрее и избегать проблемы затухающих градиентов. Функция активации ReLU представлена в следующем виде:

$$ReLU(x) = \max(0, x), \quad (4)$$

где x – входное значение в нейроне; ReLU(x) – выход после активации, при этом отрицательные значения заменяются на 0.

`Dense(256, activation = 'relu')`: Полносвязный слой с 256 нейронами и функцией активации ReLU. Функция обработки слоя с использованием ReLU описывается следующей формулой:

$$h_2 = ReLU(W_2X_{flat} + b_2), \quad (5)$$

`Dense(10, activation = 'softmax')`: Выходной слой с 10 нейронами и функцией активации softmax. Функция обработки слоя с использованием Softmax описывается следующей формулой:

$$y = Softmax(W_3h_2 + b_3), \quad (6)$$

где y – выходной вектор после слоя; softmax – функция активации; h_2 – выходной вектор после предыдущего слоя.

Функция активации Softmax необходима для преобразования выходных значений в вероятностное распределение по категориям [12]. Эта функция представлена в следующем виде:

$$\text{Softmax}(z_i) = \frac{e^{zi}}{\sum_{i=1}^{10} e^{zi}}, \quad (7)$$

где z_i – входные значения; e^{zi} – экспоненциальное преобразование.

Рассмотрим структуру сверточной нейронной сети. На рис. 2 представлена архитектура модели, отображающая последовательность слоев, процесс обучения и их взаимосвязь.

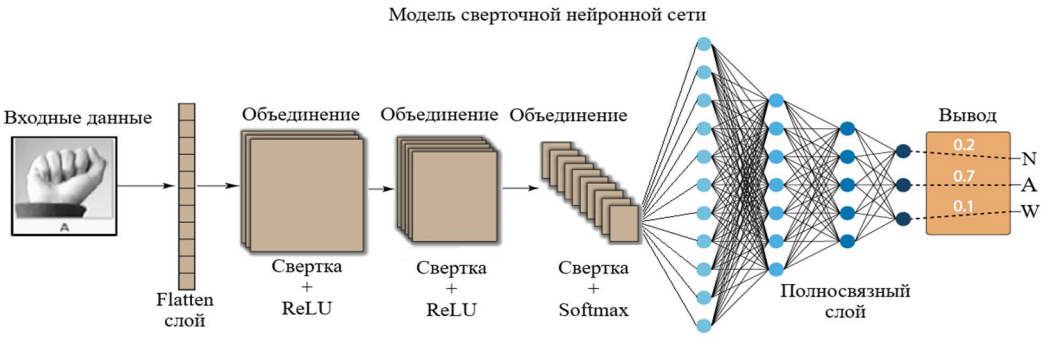

Рис. 2. Модель сверточной нейронной сети, обученной на наборе данных Sign Language MNIST

Fig. 2. Convolutional Neural Network Model Trained on the Sign Language MNIST Dataset

Flatten слой преобразует многомерный тензор в одномерный вектор, что необходимо при передаче данных от сверточных слоев к полносвязным. Например, если входной тензор имеет размерность 28×28×64. Этот слой просто «разворачивает» его в последовательность чисел длиной 50176. Это необходимо, потому что большинство классических классификационных алгоритмов, таких как полносвязные нейронные сети, требуют одномерного представления данных [13].

Далее происходит обработка свёрточным слоем. Это, пожалуй, самый важный элемент в сверточных нейросетях, поскольку именно он отвечает за извлечение признаков. В отличие от полносвязных слоев, которые анализируют входные данные в их целостности, сверточный слой действует локально, используя фильтры (или ядра) для сканирования изображения небольшими участками. Эти фильтры могут быть разных размеров, но чаще всего встречаются 3×3 или 5×5. Они работают по принципу скользящего окна: проходя по изображению, фильтры «реагируют» на определенные паттерны – углы, границы, текстуры. Это позволяет нейросети автоматически выделять ключевые особенности объектов, такие как контуры, формы или даже более сложные структуры на более поздних уровнях.

Процесс работы сверточного слоя можно представить как просмотр фотографии через небольшое окно, в котором постепенно фиксируются наиболее важные детали. Однако в отличие от простого визуального осмотра, здесь на каждом шаге фильтр выполняет математическую операцию свертки – перемножает значения пикселей на свои весовые коэффициенты и суммирует их. Итогом становится карта признаков – своего рода переработанное изображение, где уже выделены важные элементы. Чем глубже слой, тем более сложные и абстрактные признаки он может выявлять: первые слои, как правило, замечают лишь простые границы и формы, а более поздние

способны определять силуэты жестов, отделять руки от фона или даже различать пальцы на фоне ладоней.

Когда все признаки выделены, следующим этапом обычно идет полносвязный слой. Это классический элемент нейросетей, который соединяет каждый нейрон предыдущего слоя со всеми нейронами следующего. В отличие от сверточных слоев, которые обрабатывают данные локально, полносвязные слои рассматривают входной вектор целиком, что делает их очень мощным инструментом для классификации. По сути, они выполняют линейное преобразование входных данных с помощью весов и смещений, а затем применяют к результату нелинейную функцию активации, например, ReLU или Softmax.

Вместе вышеперечисленные слои образуют фундамент большинства современных моделей обработки изображений. Свёрточные слои вычленяют признаки, Flatten помогает передать их в линейное пространство, а полносвязные слои принимают финальное решение.

В процессе обучения модели применялся оптимизатор Adam и функция потерь sparse_categorical_crossentropy.

Adam – один из самых популярных оптимизаторов в глубоких нейросетевых моделях. Он совмещает в себе два алгоритма: Momentum – использует экспоненциальное сглаживание градиента, чтобы ускорять движение в нужном направлении и избегать колебаний. RMSprop – адаптивно изменяет шаг обучения для каждого параметра, уменьшая его, если градиенты имеют большую дисперсию [14]. Adam решает проблемы стандартного градиентного спуска, который используется для обновления весов и минимизации потерь. Веса в классической реализации градиентного спуска обновляются по формуле

$$\omega = \omega - \alpha \cdot \nabla L \quad (8)$$

где ω – текущие параметры модели; α – скорость обучения; ∇L – градиент функции потерь L.

Однако стандартный градиентный спуск зачастую работает неэффективно из-за резких изменений градиентов. Adam решает проблему следующим образом: Подсчетом первого момента (экспоненциальное сглаживание градиента) Adam отслеживает среднее значение градиентов mt. Процесс описывается следующей формулой

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot \nabla L_t, \quad (9)$$

где β_1 – гиперпараметр (обычно 0.9), который контролирует влияние предыдущего значения. Это позволяет "запоминать" прошлые градиенты и производить процесс обучения более плавно.

Подсчетом второго момента (среднеквадратичное сглаживание) Adam отслеживает среднее значение градиентов ut. Процесс описывается следующей формулой:

$$u_t = \beta_2 u_{t-1} + (1 - \beta_2) (\nabla L_t) \cdot (\nabla L_t), \quad (10)$$

где β_1 – гиперпараметр (обычно 0.999), который контролирует влияние предыдущего значения. Это позволяет "запоминать" прошлые градиенты и обучать более плавно.

Так как m_t и u_t в начале обучения стремятся к нулю, используется коррекция смещения:

$$m_t = \frac{m_t}{1-\beta_1^t}, \quad u_t = \frac{u_t}{1-\beta_2^t}. \qquad (11)$$

Исходя из всех модификаций классического градиентного спуска Adam, веса обновляются по следующей формуле:

$$\omega = \omega - \frac{\alpha}{\sqrt{u_t + \epsilon}} \cdot m_t, \qquad (12)$$

где ϵ (обычно 10-8) используется для предотвращения деления на ноль.

Таким образом, Adam оптимизирует обучение для каждого параметра, что делает его эффективнее, чем стандартный градиентный спуск [15].

Следующим шагом обучения является функция потерь

`sparse_categorical_crossentropy`,

которая измеряет, насколько предсказания модели отличаются от реальных меток классов. Данная функция используется, когда задача обучения классификационная и метки классов представлены целыми числами [16]. Учитывая перечисленные факторы, функция вычисляется по следующей формуле:

$$L = -\sum_{i-1}^{c} y_i \log(u_i), \qquad (13)$$

где c – количество классов; y_i – метка истинности (0 или 1); u_i – предсказанная вероятность класса i.

Получается, что в случае, если модель предсказала распределение [0.1, 0.7, 0.2], как на рис. 2, то функция потерь будет:

$$L = -\log(0{,}7) = 0{,}36. \qquad (14)$$

Обучение проводилось в течение 18 эпох и описано в коде следующим образом:

```
Python:
model.compile(optimizer = 'adam',
loss = "sparse_categorical_crossentropy",
metrics=['accuracy'])
model.fit(train_images, train_labels,
epochs = 18)
```

Динамика обучения модели относительно обучающих и тестовых данных показана на графике рис. 3.

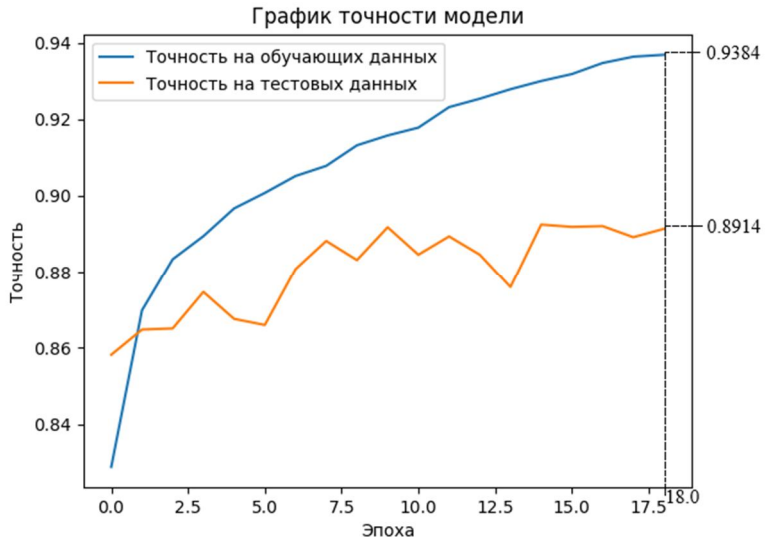

Рис. 3. График точности модели по эпохам относительно обучающих и тестовых данных

Fig. 3. Plot of model accuracy over epochs against training and test data

По графику понятно, что за 18 эпох обучения нейросеть достигла достаточно высокой точности как при проверке на обучающих данных (93.84%), так и при проверке на тестовых данных (89.14%).

Результаты моделирования работы обученной модели распознавания жестов представлены в разделе «Результаты и их обсуждение».

Результаты и их обсуждение

После обучения и тестирования нейросетевой модели были проведены точечные проверки качества распознавания жестов. Результаты проверок показаны на рис. 4.

Предложенная сверточная нейронная сеть (CNN), реализованная с использованием библиотек TensorFlow и Keras, достигла точности 89.14% на тестовом наборе данных Sign Language MNIST после 18 эпох обучения. Для оценки эффективности предложенного подхода было проведено сравнение с альтернативными методами, результаты которого представлены в табл. 1.

Таблица 1. Сравнение эффективности методов нейронных сетей

Table 1. Comparison of the effectiveness of neural network methods

Нейронная сеть / Neural network	Количество эпох обучения / Number of training epochs	Точность, % / Accuracy, %
TensorFlow и Keras	18	89.14
HOG и SVM	18	70.1
Сеть с двумя скрытыми слоями	18	78.4

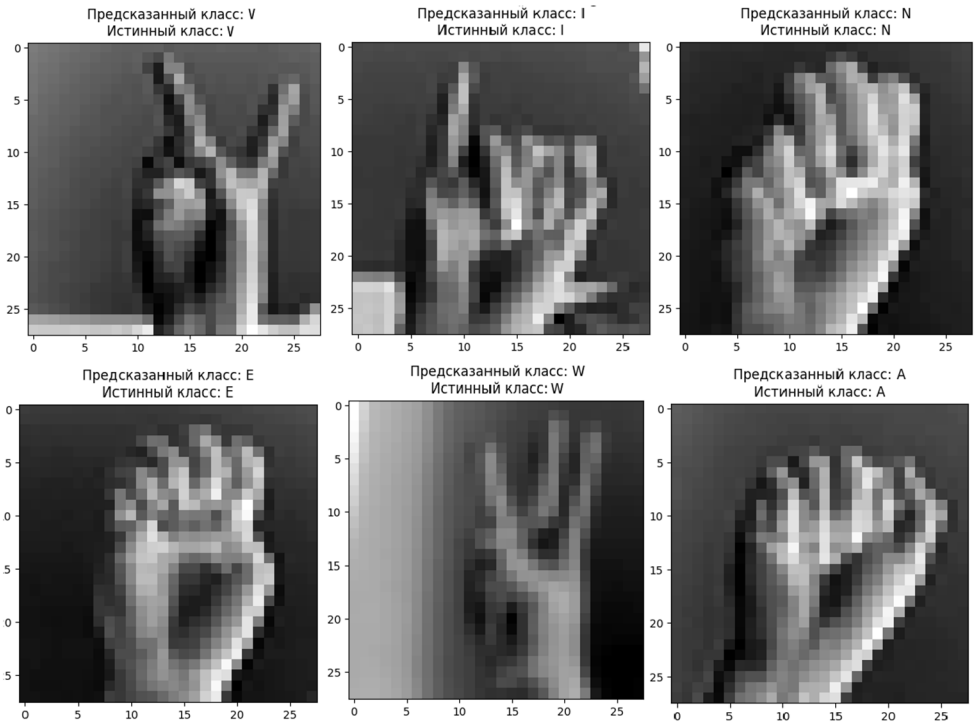

Рис. 4. Точечные проверки качества распознавания жестов, где приведены следующие классы (буквы): **V** (21), **I** (8), **N** (13), **E** (4), **W** (22), **A** (0)

Fig. 4. Point tests of gesture recognition quality, where the following classes (letters) are given: **V** (21), **I** (8), **N** (13), **E** (4), **W** (22), **A** (0)

Метод, основанный на гистограммах ориентированных градиентов (HOG) в сочетании с классификатором SVM, показал точность 70.1% на аналогичных наборах данных жестов [17], что значительно ниже предложенной CNN с использованием TensorFlow и Keras. Полносвязная нейронная сеть с двумя скрытыми слоями, рассмотренная в [18], достигла точности около 78.4% на наборе Sign Language MNIST, но требовала большего времени на обучение и демонстрировала склонность к переобучению из-за отсутствия сверточных слоев, которые эффективно извлекают иерархические признаки.

Выводы

Сравнение подтверждает, что предложенная CNN превосходит традиционные методы и простые нейронные сети по точности, скорости сходимости и устойчивости к вариациям данных. Высокая точность (89.14%) и стабильные результаты на тестовом наборе демонстрируют эффективность использования сверточных слоев для автоматического выделения признаков и полносвязных слоев для классификации в задачах распознавания жестов.

Полученные в ходе исследования результаты могут быть применены в системах управления и робототехнике, что согласуется с исследованиями нечетких систем управления [19, 20].

В перспективе возможно дальнейшее улучшение точности классификации за счет модернизации алгоритмов, используемых в Adam. Это позволит избежать эффекта переобучения и оптимальнее использовать ресурсы, задействованные для этого процесса.

Список литературы

1. Gradient-based learning applied to document recognition / Y. LeCun, L. Bottou, Y. Bengio, P. Haffner // Proceedings of the IEEE. 1998. № 86(11). P. 2278–2324.

2. Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks // Advances in Neural Information Processing Systems. 2012. Vol. 25. P. 1097–1105.

3. Воронцов К. В. Машинное обучение и анализ данных // Труды международной научной конференции "Нейроинформатика". М.: МФТИ, 2020. 452 с.

4. Петров И. В., Смирнов А. А. Применение сверточных нейронных сетей для классификации изображений в задачах компьютерного зрения // Искусственный интеллект и принятие решений. 2021. № 2. С. 45-58.

5. Китенко А. М. Метод поиска и разметки артефактов на изображениях с использованием алгоритмов детекции и сегментации // Системы анализа и обработки данных. 2021. № 4(84). С. 7-18.

6. Robust Hand Gesture Recognition Using HOG-9ULBP Features and SVM Model / J. Li, C. Li, J. Han, et al. // Electronics. 2022. Vol. 11(7). P. 988.

7. Козлов С. В., Иванова Е. П. Сравнительный анализ архитектур глубоких нейронных сетей для распознавания образов // Программные продукты и системы. 2022. № 3. С. 28-36.

8. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // International Conference on Learning Representations (ICLR). 2015. arXiv:1409.1556.

9. Kumar R., Patel S., Sharma M. Enhancing Sign Language Detection through MediaPipe and Convolutional Neural Networks // arXiv preprint. 2024. arXiv:2406.03729v1.

10. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // Proceedings of the 32nd International Conference on Machine Learning (ICML). 2015. P. 448-456.

11. Семенов Д. А., Кузнецов М. И. Оптимизация процесса обучения сверточных нейронных сетей с использованием адаптивных алгоритмов // Информационные технологии. 2023. Т. 29, № 4. С. 195-203.

12. Nair V., Hinton G. E. Rectified Linear Units Improve Restricted Boltzmann Machines // Proceedings of the 27th International Conference on Machine Learning (ICML). 2010. P. 807-814.

13. Deep Residual Learning for Image Recognition / K. He, X. Zhang, S. Ren, J. Sun // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 770-778.

14. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization // International Conference on Learning Representations (ICLR). 2015. arXiv:1412.6980.

15. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov // Journal of Machine Learning Research. 2014. Vol. 15, № 1. P. 1929-1958.

16. Going deeper with convolutions / C. Szegedy, W. Liu, Y. Jia, et al. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015. P. 1-9.

17. Фаворская М. Н., Пахирка А. И. Построение карт глубины при обнаружении презентационных атак в системах распознавания лиц // Информационные и математические технологии в науке и управлении. 2022. № 3(27). С. 40-48.

18. Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation / N. C. Camgoz, O. Koller, S. Hadfield, R. Bowden // IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020. P. 10023-10033.

19. Исследование устройства нечеткого цифрового фильтра для робота-манипулятора / М.В. Бобырь, Н.А. Милостная, В.А. Булатников, М.Ю. Лунева // Известия Юго-Западного государственного университета. 2020. Т. 24, №1. С. 115-129. https:// doi.org/10.21869/2223-1560-2020-24-1-115-129

20. Бобырь М. В., Нассер А. А., Абдулджаббар М. А. Исследование свойств мягкого алгоритма нечетко-логического вывода // Известия Юго-Западного государственного университета. 2016. № 1. С. 31-49.

References

1. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998; (86): 2278-2324.

2. Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*. 2012; 25: 1097-1105.

3. Vorontsov K. V. Machine learning and data analysis. In: *Trudy mezhdunarodnoi nauchnoi konferentsii "Neiroinformatika" = Proceedings of the international scientific conference "Neuroinformatics"*. Moscow; 2020. 452 p. (In Russ.).

4. Petrov I. V., Smirnov A. A. Application of convolutional neural networks for image classification in computer vision problems. *Iskusstvennyi intellekt i prinyatie reshenii = Artificial Intelligence and Decision Making.* 2021; (2): 45-58. (In Russ.).

5. Kitenko A. M. Method for searching and marking artifacts in images using detection and segmentation algorithms. *Sistemy analiza i obrabotki dannykh = Data analysis and processing systems.* 2021; (4): 7-18. (In Russ.).

6. Li J., Li C., Han J., et al. Robust Hand Gesture Recognition Using HOG-9ULBP Features and SVM Model. *Electronics*. 2022; 11 (7): 988.

7. Kozlov S. V., Ivanova E. P. Comparative analysis of deep neural network architectures for pattern recognition. *Programmnye produkty i sistemy = Software products and systems*. 2022: (3): 28-36. (In Russ.).

8. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*. 2015; arXiv:1409.1556.

9. Kumar R., Patel S., Sharma M. Enhancing Sign Language Detection through MediaPipe and Convolutional Neural Networks. arXiv preprint, 2024, arXiv:2406.03729v1.

10. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015. P. 448-456.

11. Semenov D. A., Kuznetsov M. I. Optimization of the Training Process of Convolutional Neural Networks Using Adaptive Algorithms. *Informatsionnye tekhnologii = Information Technologies.* 2023; 29(4): 195-203. (In Russ.).

12. Nair V., Hinton G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010. P. 807-814.

13. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. P. 770-778.

14. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015, arXiv:1412.6980.

15. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014; 15(1): 1929-1958.

16. Szegedy C., Liu W., Jia Y., et al. Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015. P. 1-9.

17. Favorskaya M. N., Pakhirka A. I. Construction of depth maps for detection of presentation attacks in face recognition systems. *Informatsionnye i matematicheskie tekhnologii v nauke i upravlenii = Information and mathematical technologies in science and management*. 2022; (3): 40-48. (In Russ.).

18. Camgoz N. C., Koller O., Hadfield S., Bowden R. Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. P. 10023-10033.

19. Bobyr M. V., Milostnaya N. A., Bulatnikov V. A, Luneva M. Yu. Fuzzy Digital Filter Device Study for the Robot Manipulator. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2020; 24(1): 115-129 (In Russ.). https://doi.org/10.21869/2223-1560-2020-24-1-115-129

20 Bobyr. M. V., Nasser A. A., Abduljabbar M. A. Study of the properties of a soft algorithm for fuzzy-logical inference. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2016; (1): 31-49. (In Russ.).

Информация об авторах / Information about the Authors

Бобырь Максим Владимирович, доктор технических наук, профессор кафедры программной инженерии, Юго-Западный государственный университет, г. Курск, Российская Федерация, e-mail: fregat_mn@rambler.ru, ORCID: http://orcid.org/0000-0002-5400-6817, Researcher ID: G-2604-2013

Maxim V. Bobyr, Dr. of Sci. (Engineering), Professor of the Software Engineering Department, Southwest State University, Kursk, Russian Federation, e-mail: fregat_mn@rambler.ru, ORCID: http://orcid.org/0000-0002-5400-6817, Researcher ID: G-2604-2013

Асеев Артем Андреевич, аспирант кафедры программной инженерии, Юго-Западный государственный университет, г. Курск, Российская Федерация, e-mail: aseeff.artem@yandex.ru, ORCID: http://orcid.org/ 0009-0007-8271-7660

Artem A. Aseev, Post-Graduate Student of the Software Engineering Department, Southwest State University, Kursk, Russian Federation, e-mail: aseeff.artem@yandex.ru, ORCID: http://orcid.org/ 0009-0007-8271-7660