2024 27th International Conference on Computer and Information Technology (ICCIT)
20-22 December 2024, Cox's Bazar, Bangladesh

2024 27th International Conference on Computer and Information Technology (ICCIT) | 979-8-3315-1909-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCIT64611.2024.11022352

CTS-Synthesizer: Handwritten Character to Speech Conversion using CNN and Google Text-to-Speech Synthesizer

Showrajit Saha, Nursadul Mamun
Robust Speech Processing Laboratory (RSPL), Department of Electronics and Telecommunication Engineering
Chittagong University of Engineering and Technology, Chittagong
22mete101@student.cuet.ac.bd, nursad.mamun@cuet.ac.bd

***Abstract*—In the ever-evolving field of artificial intelligence and computer vision, a Handwritten Character-to-speech (CTS) Synthesizer has emerged as an integral component within human-like interactions. Although research has been studied to recognize either digit or character recognition, this paper presents a machine learning-based method for real-time converting handwritten alphanumeric (refer to both English characters and digits) words to speech. The approach utilizes a convolutional neural network (CNN) for precise handwritten character recognition and integrates the Google Text-to-Speech (TTS) synthesizer for converting recognized characters into speech. The proposed model is trained and tested using a combination of two well-known handwritten datasets (one for characters and one for digits). The CNN architecture is employed to effectively extract features and classify characters and digits from pre-processed images. Once the text is recognized, it is then processed by Google Text-to-Speech to generate natural-sounding speech. The results demonstrate that the proposed model achieves high accuracy in predicting 36 English handwritten characters and digits, with an overall accuracy of 99.22%. The proposed system can be applied in various domains and made a valuable contribution to the field of Natural Language Processing (NLP), offering significant potential applications in assistive assistive technology for visually impaired individuals and automated data entry systems and beyond.**

***Index Terms*—Handwritten character recognition, Convolutional Neural Networks (CNN), MNIST dataset, Text-to-Speech (TTS), Google Text-to-Speech (gTTS), Character to Speech (CTS), Natural Language Processing (NLP).**

I. Introduction

In the digital era, handwritten character detection and voice transformation are among the most intriguing and challenging research areas within deep learning. The diversity of individual handwriting styles makes manual transcription both error-prone and time-consuming. Consequently, machine learning approaches have emerged as a reliable solution for identifying and interpreting handwritten characters and digits [1].

Numerous character recognition techniques have been developed to enhance image classification accuracy for handwritten text [2]–[8]. For instance, Younis and colleagues, in 2017, combined Optical Character Recognition (OCR) with face recognition, achieving a 98.47% accuracy rate in character recognition [7]. Similarly, in 2019, Vijayalaxmi and his team utilized Convolutional Neural Networks (CNNs) to accurately classify handwritten digits [2]. However, their approach did not address converting recognized digits into speech. Mahmud and colleagues further advanced the use of CNNs for recognizing both handwritten English characters and digits, demonstrating impressive accuracy using the MNIST dataset. Following this, Akash and his team sought to optimize time and space efficiency by leveraging the properties of words as a whole rather than individual letters [4]. Their approach integrated image processing, text-to-speech (TTS) synthesis, OCR, and natural language processing (NLP) to extract relevant information. However, the complexity of this method could hinder its real-time applicability, especially in tasks requiring rapid processing.

In 2020, Rajbongshi and colleagues explored the use of CNNs in combination with Google TTS for Bangla OCR and speech conversion on a Raspberry Pi platform. This process employed Tesseract OCR for text extraction and the Google TTS engine for speech synthesis [6]. Despite their innovative approach, the accuracy of handwritten character recognition was suboptimal, and real-time performance posed significant challenges, particularly for applications demanding quick processing and response times.

The primary goal of these studies was to develop a pattern recognition technique capable of accurately reading handwritten letters and digits. The integration of CNNs with TTS systems, as demonstrated in these works, holds significant promise for enhancing the efficiency and accuracy of character recognition models. Such models have the potential to be applied in real-world scenarios, improving the robustness and effectiveness of character recognition and voice transformation applications.

This study proposes a machine learning-based approach, CTS-Synthesizer, for handwritten alphanumeric word recognition. The method enhances the performance of identifying English digits and characters, both individually and collectively, by utilizing CNNs with optimized hyperparameters. Additionally, the recognized digits or characters are converted to speech corresponding to the input.

The paper is organized as follows: Sec. II details the implementation of our proposed CTS-Synthesizer. Sec. III describes the experimental setup and evaluation results, followed by the conclusion in Sec. IV.

979-8-3315-1909-4/24/$31.00 ©2024 IEEE

II. Proposed Methodology

The overall block diagram of the proposed system is shown in Fig. 1. The process begins with preprocessing the input image of the handwritten symbol to eliminate any blank spaces. Next, OCR is applied to identify individual characters in the handwritten content, converting them into a digital format understandable by a machine. A series of Convolutional Neural Network (CNN) layers are then used to accurately classify the input symbols. Finally, the recognized characters are converted to speech using Google Text-to-Speech synthesis.

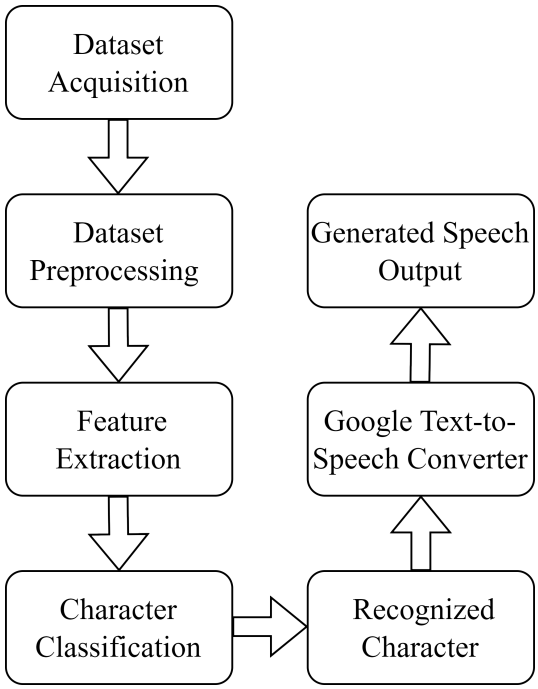

Fig. 1. Basic block diagram of the proposed CTS-Synthesizer.

A. Dataset Acquisition

The proposed model begins by collecting and converting the dataset into a machine-readable format to enable efficient processing and analysis. In this study, we utilized two primary datasets: the A to Z Handwritten Alphabets dataset [9], comprising 372,450 samples of alphabetic characters, and the MNIST dataset [8], which includes 70,000 handwritten digits ranging from 0 to 9. The A to Z dataset offers a comprehensive collection of alphabetic characters, capturing a wide array of handwriting styles and variations. Meanwhile, the MNIST dataset serves as a widely recognized benchmark for digit recognition in machine learning.

To enhance the robustness and accuracy of our model, we merged these two datasets, creating a diverse and extensive collection of handwritten characters and digits. Fig. 2 illustrates this combined dataset, highlighting the variety and breadth of samples integrated from both sources. This amalgamation enables our model to effectively learn from a rich set of inputs, thereby improving its ability to recognize and classify handwritten text with greater accuracy.

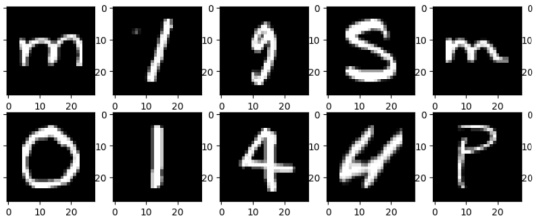

Fig. 2. Example of the combined handwritten dataset, showcasing samples from both the A to Z Handwritten Alphabets dataset and the MNIST dataset.

B. Data Prepossessing

The datasets are preprocessed to enable effective character interpretation by the machine. Each image is resized to a consistent scale, and formatted into a 4-dimensional structure suitable for the model. The resized dataset is then shuffled to randomize the order of samples. The two datasets are combined to create a unified dataset containing both digits and letters, which is subsequently fed into the Convolutional Neural Network (CNN). The overall dataset is split into training and testing sets, with 80% allocated for training and 20% reserved for testing. Finally, a normalization technique is applied to the training images, scaling the pixel values to a range of 0 to 1.

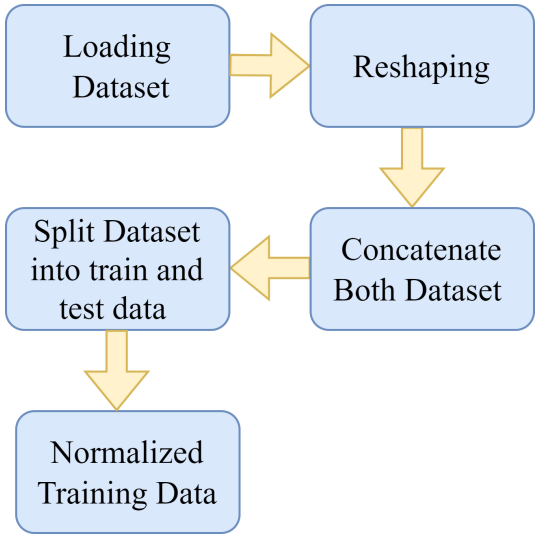

Fig. 3. A comprehensive flowchart illustrating the steps involved in preparing the dataset for model training and evaluation.

C. Network Architecture

Convolution involves multiplying two linear signals to extract features from an input. In CNNs, this process is facilitated by the convolutional and pooling layers. The convolutional layer applies a set of filters, or kernels, to the input image, where each filter is a 2D array of weights. These filters help in identifying key features such as edges and shapes by multiplying them with the pixel values of the image. Multiple convolutional layers are often stacked to capture increasingly complex features, with the pooling layer helping to reduce the

size of the feature maps and the number of parameters, thereby decreasing computational requirements and processing time. Pooling layers play a crucial role by shrinking the dimensions of the feature maps, reducing the computational cost, and making the model more robust to variations in the input image. After feature extraction through convolution and pooling layers, the CNN architecture uses fully connected layers to map the features into the final output. Fully connected layers can sometimes lead to overfitting, where the model learns to memorize the training data rather than generalize from it. Regularization techniques such as dropout mitigate this issue by randomly removing some input nodes, which helps prevent overfitting and encourages generalizable learning.

In a modified CNN architecture designed for recognizing handwritten alphanumeric, the input images are resized to 28×28 pixels in grayscale. The first convolutional layer uses 32 filters with a 5×5 window size and a ReLU activation function to capture various features from the input. This layer learns 32 different feature maps, enabling the extraction of edges, corners, and shapes. The architecture is replicated in subsequent convolutional layers to deepen the model, allowing it to learn more intricate features. Batch normalization layers are applied to speed up training and stabilize the model, followed by a MaxPooling2D layer with a 2×2 window size to reduce data dimensionality and enhance robustness against small input variations.

After the MaxPooling2D layer, a 50% dropout layer is introduced to reduce overfitting by randomly disabling some neurons during training. This is followed by batch normalization to further stabilize and accelerate the learning process. The flattened layer then transforms the 2D feature maps into a 1D vector for the fully connected layers. The first dense layer, with 256 neurons, performs matrix multiplication to learn complex feature relationships. The final output is generated by a dense layer with 36 nodes and softmax activation for multi-class classification. For the MNIST dataset, this final dense layer has 10 nodes, while for recognizing handwritten uppercase letters, it has 26 nodes. A diagram of this modified CNN model is illustrated in Fig. 6.

D. Google Text-To-Speech Synthesizer

The Google Text-to-Speech Synthesizer [10] is a powerful tool that leverages advanced AI algorithms to convert written text into natural-sounding speech. This technology is highly effective in providing auditory output for various applications, particularly for enhancing accessibility. In this paper, we demonstrate how the Google Text-to-Speech Synthesizer can be seamlessly integrated to capture handwritten content processed in earlier stages and convert it into audio output. This capability is especially beneficial for visually impaired users, as it enables them to access and interact with handwritten information through auditory means, significantly improving their ability to engage with digital content.

III. Experimental Results and Analysis

This section presents and analyzes the experimental results obtained from the proposed CTS-Synthesizer across various datasets. The performance of the CTS-Synthesizer is evaluated in terms of accuracy, efficiency, and robustness, providing a comprehensive assessment of its capabilities. Detailed comparisons with baseline models are also included to highlight the improvements introduced by the CTS-Synthesizer. Additionally, the results are discussed in the context of different dataset characteristics, showcasing how the synthesizer adapts and performs under varying conditions.

A. Classification accuracy for individual dataset

Initially, the MNIST dataset [8] was utilized for the experiment. With 15 epochs of training, the proposed CNN model achieved a test accuracy of 99.26% with a 50% dropout layer, surpassing the performance of other methods. This indicates the model's effectiveness in recognizing digits while maintaining robustness against overfitting. A comparison table, shown in Table I, highlights the training accuracy, test accuracy, and validation accuracy for digit recognition across different dropout layer configurations in the MNIST dataset.

TABLE I
Experimental Results for MNIST dataset

Dropout Layer	Training Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
None	99.66	98.90	98.79
0.1	99.64	98.77	99.00
0.25	99.61	99.04	99.02
0.50	**99.61**	**99.23**	**99.26**
0.60	99.62	99.03	99.13

Subsequently, the A-Z Handwritten Alphabet dataset [9] was employed to train the proposed CNN model independently, without merging it with the MNIST dataset. The model achieved a testing accuracy of approximately 99.54% for the recognition of capital English letters and digits. This high accuracy demonstrates the model's capability to accurately distinguish handwritten characters across a broader range of classes. In the case of applying a dropout layer specifically for the A-Z Handwritten Alphabet dataset, a corresponding table, Table. II, presents a detailed comparison of the training accuracy, test accuracy, and validation accuracy for CNN-based character recognition. This comparison provides insight into how different dropout configurations impact the model's performance in recognizing both digits and letters.

B. Classification accuracy for combined datasets

The A-Z Handwritten Alphabet dataset was combined with the MNIST dataset to further train the proposed CNN architecture and evaluate its performance on a more comprehensive dataset. The testing accuracy remained consistent with the results obtained in previous steps, indicating the robustness of the model. The training process involved 18 epochs, with the combined dataset being split into 80% for training and 20% for testing, achieving a test accuracy of 99.22%, shown

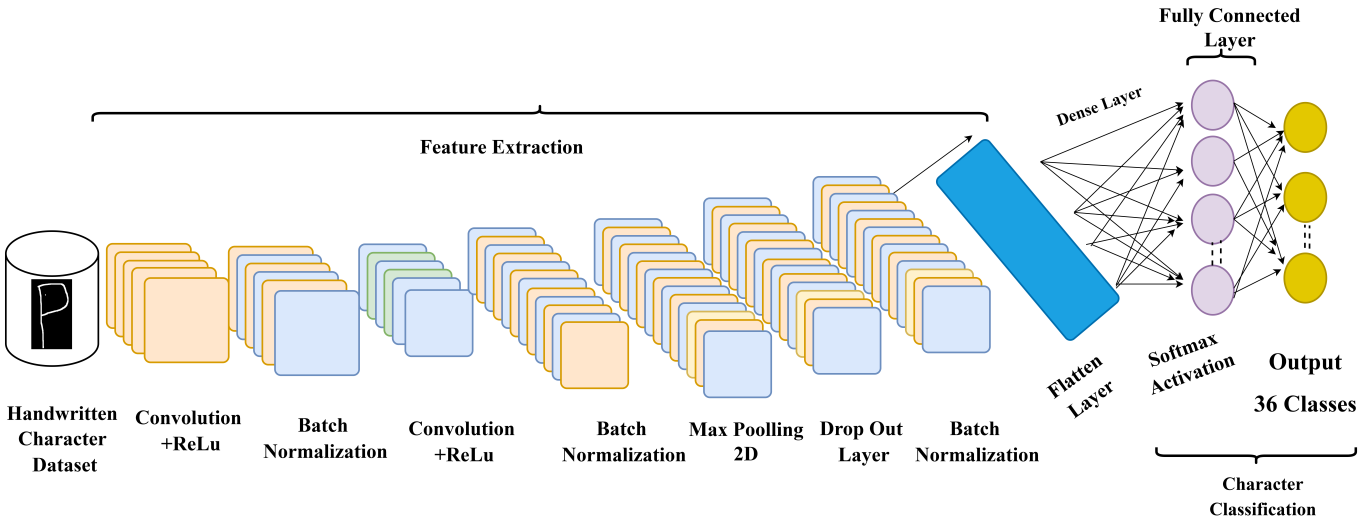

Fig. 4. Diagram illustrating the proposed architecture, showcasing the layers and processes involved in feature extraction and classification.

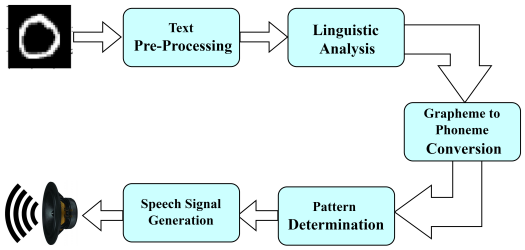

Fig. 5. Overview of a Text-to-speech Synthesizer.

TABLE II
EXPERIMENTAL RESULTS OF TRAINING CNN MODEL WITH A-Z HANDWRITTEN ALPHABET DATASET

Dropout Layer	Training Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
None	99.68	99.23	99.23
0.25	99.65	99.43	99.43
0.50	**99.61**	**99.54**	**99.54**
0.60	99.53	99.56	99.56

in Table III. With a 50% dropout rate, the model attained 99.41% training accuracy, demonstrating an effective reduction in overfitting.

TABLE III
EXPERIMENTAL RESULTS OF TRAINING MODEL WITH COMBINED DATASET

Dropout Layer	Training Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
None	99.68	98.83	98.83
0.25	99.63	99.08	99.08
0.40	99.53	99.09	99.09
0.50	**99.41**	**99.22**	**99.22**
0.60	99.44	99.09	99.09

Table IV presents a comparative analysis of various networks utilized for digit and character classification. The table highlights the performance metrics across different models, emphasizing that while most existing networks are designed for recognizing only one type of character, the proposed CTS-Synthesizer demonstrates superior performance. Notably, the CTS-Synthesizer outperforms these existing networks in terms of accuracy, generalization, and robustness, making it a more effective solution for both digit and character classification tasks.

TABLE IV
COMPARE ACCURACY WITH DIFFERENT CNN ARCHITECTURES FOR HANDWRITTEN LETTER AND DIGITS RECOGNITION PART.

Author	Work	Approach	Test Accuracy (%)
Mahmud at all., 2021 [1]	Digit and Character Recognition	CNN	98.94
Younis at all., 2017 [7]	Optical Character Recognition	CNN	98.46
Dutt at all., 2017 [9]	Digit Recognition	CNN	98.70
Maghari at all, 2017 [11]	Digit Recognition	DNN	98.08
Ranjan at all., 2019 [12]	Character Recognition	CNN	98.85
Proposed CTS-Synthesizer	**Handwritten Digit and Character Recognition**	**CNN**	**99.22**

After recognizing the characters and digits, the Google Text-to-Speech Synthesizer (gTTS) was used to convert the recognized characters into speech, generating clear and accurate audio output from the recognized text [10]. This integration of gTTS not only enhances the functionality of the model by providing audible feedback but also improves accessibility for users, making the system more versatile and user-friendly. The

combination of handwritten digit and character recognition with speech synthesis represents a significant advancement in the usability of handwritten data.

C. Confusion matrix

The confusion matrix provides a comprehensive view of the model's performance by detailing the number of correct and incorrect classifications, as shown in Fig. 6 for test predictions after data augmentation on the MNIST digit dataset. It highlights true and misclassified predictions for each class, offering insights into the model's strengths and weaknesses. Misclassification analysis revealed recurring errors, such as confusion between visually similar characters like 'o' and '0' or 'l' and '1,' due to shape ambiguity. In the MNIST dataset, digits like '3' and '8' or '4' and '9' were often misclassified because of overlapping features in certain handwriting styles. Similarly, ambiguous characters like 'O' and 'Q' or 'I' and 'L' showed higher error rates in the A-Z dataset. Poor-quality samples, including noise or incomplete strokes, further impacted accuracy, particularly for less distinct characters, while imbalanced datasets contributed to the misclassification of rarely occurring characters like 'Z.' Architectural limitations of the CNN and potential labeling errors also introduced noise into the learning process. Addressing these challenges through data augmentation, noise reduction, balancing datasets, and fine-tuning the CNN architecture can significantly improve accuracy and reduce misclassification rates across both datasets.

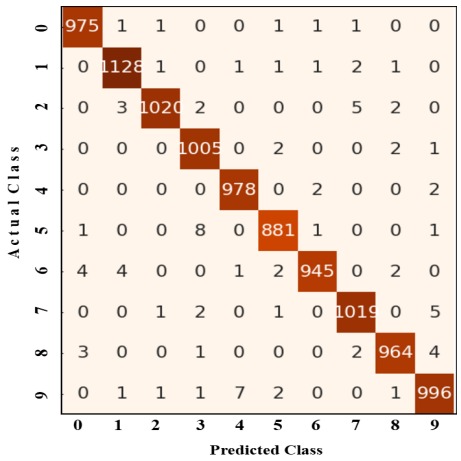

Fig. 6. Confusion matrix of MNIST digit classification.

CONCLUSION

The primary objective of this work is to enhance the accuracy of classifying handwritten alphanumerics using a CNN architecture. The recognized characters are then converted into speech using Google's Text-to-Speech Synthesizer, extending the application to various real-world scenarios. By integrating CNNs with Google's Text-to-Speech Synthesizer, the proposed method offers an effective solution for Handwritten English characters and digit recognition and speech conversion. The CNN architecture achieved an impressive 99.41% training accuracy and 99.22% testing accuracy, demonstrating effective mitigation of overfitting and strong generalizability to unseen handwritten characters and digits. This research is particularly beneficial for visually impaired or illiterate individuals, enabling them to listen to the content of handwritten numbers and characters. Additionally, it serves anyone who prefers auditory information over reading visual content. In summary, the tasks of handwritten English character and digit recognition, coupled with speech conversion via CNN and Google Text-to-Speech, are complex and time-intensive, requiring the integration of OCR, NLP, TTS, and image processing technologies.

REFERENCES

[1] A. Tanvin, a. Rahman, Sazi *et al.*, "Handwritten english character and digit recognition," *International Conference on Electronics, Communications and Information Technology*, pp. 1–4, 2021.

[2] V. Athila and A. S. Chandran, "Comparative analysis of algorithms used in handwritten digit recognition," *International Research Journal of Engineering and Technology*, vol. 8, no. 6, 2021.

[3] B. Vidhale, G. Khekare, C. Dhule, P. Chandankhede, A. Titarmare, and M. Tayade, "Multilingual text & handwritten digit recognition and conversion of regional languages into universal language using neural networks," pp. 1–5, 2021.

[4] A. Anand, R. Rastogi, A. Chadichal, G. S. Surana, and N. R. Latha, "Handwritten text recognition and conversion to speech." pp. 1–5, 2023.

[5] S. Akshay, J. Chandana, and G. Namita, "Handwritten english character recognition and speech synthesis to aid text-to-speech applications," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 8, no. 5, pp. 2249–8958, 2019.

[6] A. Rajbongshi, M. I. Islam, A. A. Biswas, M. M. Rahman, A. Majumder, and M. E. Islam, "Bangla optical character recognition and text-to-speech conversion using raspberry pi," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 6, 2020.

[7] K. S. Younis and A. A. Alkhateeb, "A new implementation of deep neural networks for optical character recognition and face recognition," *Proceedings of the new trends in information technology*, pp. 157–162, 2017.

[8] B. Gope, S. Pande, N. Karale, S. Dharmale, and P. Umekar, "Handwritten digits identification using mnist database via machine learning models," vol. 1022, no. 1, pp. 1–11, 2021.

[9] A. Dutt and A. Dutt, "Handwritten digit recognition using deep learning," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 6, no. 7, pp. 990–997, 2017.

[10] S. Janokar, S. Ratnaparkhi, M. Rathi, and A. Rathod, "Text-to-speech and speech-to-text converter—voice assistant," pp. 653–664, 2023.

[11] M. M. A. Ghosh and A. Y. Maghari, "A comparative study on handwriting digit recognition using neural networks," pp. 77–81, 2017.

[12] R. Jana and S. Bhattacharyya, "Character recognition from handwritten image using convolutional neural networks," pp. 23–30, 2019.