ISSN 2616-5775 Vol. 8, Issue 8: 67-71, DOI: 10.25236/AJCIS.2025.080810

The Development of Simple Speech Recognition Program and Problem Solving

Sibo Song[a,#], Zexin Yin[b,#], Haoze Liu[c,#], Peiyan Han[d,#], Siying Song[e,#], Zitai Wang[f,#]

Beijing 21st Century School, Beijing, China
[a]jacksong090121@gmail.com, [b]yzx15610189050@163.com, [c]2427519020@qq.com, [d]18514778980@163.com, [e]17310397095@163.com, [f]2909566327@qq.com
[#]Co-first author

Abstract: *To meet the edge-side application requirements of speech recognition in the human-computer interaction scenario of AI refrigerators, this paper designs a concise deep learning architecture for speech recognition tasks. This architecture consists of 4 convolutional layers, 2 pooling layers, and 2 fully connected layers, featuring a small parameter count, suitability for edge-side computing, and low power consumption. It can effectively address the limited hardware processing capabilities of devices such as AI refrigerators. Verification results on the target dataset show that the speech recognition accuracy of this architecture reaches, which can meet the practical requirements of speech command recognition in the human-computer interaction of AI refrigerators.*

Keywords: *AI Refrigerator, Speech Recognition, Lightweight Convolutional Neural Network, Lightweight CNN, Edge Computing*

1. Introduction

Artificial intelligence has greatly improved how we interact with technology. Voice recognition, which feels more natural, has replaced traditional keyboards and mice as the main way to control devices [1].Smart homes exemplify this trend—they offer convenience, enhance user experience, and smoothly incorporate technology into everyday life.

For example, smart refrigerators show how AI technology works in real life. Modern refrigerators focus on AI, sensors, and scanning technologies. Dai (2024) improved recognition accuracy in tough situations like far-away shots, small objects, and occlusion by upgrading the Feature Pyramid Network (FPN)[2]. Rokhva et al. (2024) used a lightweight MobileNetV2 model to perform quick, real-time food recognition, making it a practical solution for devices with limited processing power[3]. In sensing tech, Banoth and Murthy (2024) used weight, odor, or light sensors to monitor stored food in real time[4]. Various AI models, such as convolutional neural networks (CNNs), have also been used for identifying food items, as shown by Lubura et al. (2022), who improved accuracy in distinguishing similar foods through image recognition[5].

Speech recognition has made big progress recently. Since around 2018, large-scale deep learning models began dominating this area. For instance, Wav2Vec 2.0 used pre-training on huge amounts of unlabeled speech data to reach performance on par with, or better than, traditional supervised methods[6]. The 2022 OpenAI Whisper model demonstrated outstanding versatility across multiple languages and tasks[7]. These developments show that speech recognition is moving toward bigger, more accurate, and more widely applicable systems, often with hundreds of millions or even billions of parameters for open-domain tasks.

However, as household devices, smart refrigerators are subject to multiple limitations, such as hardware computing power, memory, and power consumption. For instance, many large transformer architectures operate non-streaming or require full-sequence input, resulting in unacceptable delays for real-time command-and-control interactions expected in smart appliances[8]. Even if remote inference is used, the energy needed for transmitting audio to the cloud, processing, and receiving results is nontrivial, and reliance on continuous connectivity raises both privacy concerns and maintenance burden[9]. If the above-mentioned ultra-large-scale speech recognition model is directly adopted, it will lead to low operational efficiency and even make deployment impossible.

To address the challenges of limited computation, memory, and power in speech recognition for AI-enabled refrigerators, this system adopts a lightweight model based on a Convolutional Neural Network (CNN).This approach helps manage these constraints through several key strategies. First, CNNs rely on local receptive fields and weight-sharing mechanisms, which drastically cut down the number of model parameters and reduce the computational and storage demands. This means the model can run smoothly on the limited hardware of refrigerators. The design of CNNs, with their sparse connections and layered structure, also makes the computation more efficient and conserves memory[10]. Besides, by stacking convolutional and pooling layers, the data's dimensions are reduced, which lowers the processing load and resource use.

2. System design and optimization

2.1 Devising

A typical convolutional neural network includes an input layer, multiple convolutional layers, pooling layers, fully connected layers, and an output layer. Convolutional and pooling layers are usually arranged in an alternating pattern—each convolutional layer is followed by a pooling layer, and then the next convolutional layer. The key feature of convolutional layers is that each neuron in the output only connects to a small, local area of the input. The neuron calculates a weighted sum of this local region using the convolution kernel, adds a bias, and then passes the result through a nonlinear activation function. This process, called convolution, is how CNNs extract features. The input layer acts as the entry point, accepting raw data—such as images—and normalizing it to prepare for further processing. (Figure 1)

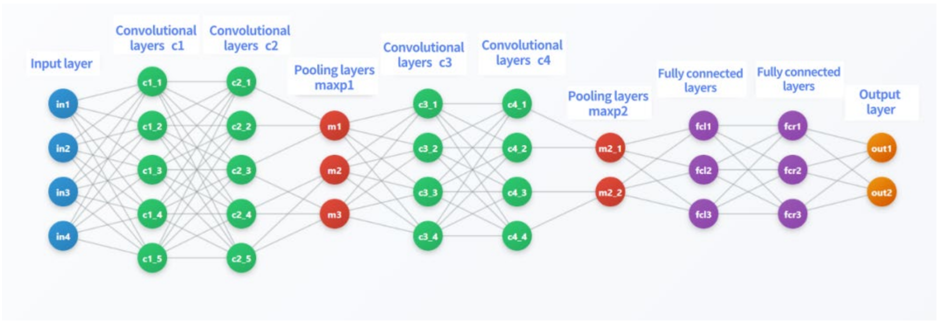

Figure 1 Schematic diagram of system architecture

A convolutional neural network consists of an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer.

2.1.1 Input layer

The input layer serves as the entry point of the network, responsible for receiving and normalising raw input image data to prepare it for subsequent feature processing.

2.1.2 Convolutional layer

The convolutional layer consists of multiple feature maps; each composed of numerous neurons. Each neuron in the layer is connected to a local region of the feature map from the previous layer through a convolution kernel. Its fundamental principle involves using a convolution kernel (such as a 3×3 kernel) to extract local details within an image. The first couple of convolutional layers focus on simple features like edges and corners, while the later layers combine these basics into more complicated patterns.

2.1.3 Maxpooling

After certain convolutional layers, we add pooling layers with a stride of two. These layers help to shrink the size of the feature maps step-by-step. This not only makes the model faster and less heavy on calculations but also helps it be better at recognizing objects even if they shift or change a little in the image, making it more reliable.

2.1.4 Fully Connected Layers

They represent the fully connected layers (FC) located at the front or on the left and right sides, respectively. In the CNN structure, after passing through multiple convolutional layers and sampling layers, there is one or more fully connected layers connected. Each neuron in the fully connected layer is fully connected to all neurons in the previous layer. The fully connected layer can integrate local information with category discrimination from the convolutional layers or sampling layers.

2.1.5 Output Layer

The output layer is typically a fully connected layer, with the number of neurons determined by the number of task categories. Together with an activation function, it ultimately outputs the predicted probability distribution for each category.

2.2 Optimization

Regularization in neural networks involves limiting the complexity of the model or adding constraints to enhance its generalization ability. Based on the paper of the relationship between convolutional neural network and parameters [10], we attempted to notice that dropout is used to avoid overfitting during training. Even if the output values of hidden layer neurons become 0 with a probability of 0.5, this technique renders some hidden layer nodes ineffective. These nodes do not participate in the forward propagation process of CNN, nor do they participate in the backward propagation process. For each sample input to the network, due to the randomness of the dropout technique, the corresponding network structure is different, but all these structures share weights. Since a neuron cannot rely on other specific neurons to exist, this technique reduces the complexity of mutual adaptation among neurons, enabling neurons to learn better features.

Training phase:

$$\widetilde{h_\iota} = h_i \cdot r_i, r_i \sim Bernoulli(1-p) \tag{1}$$

Testing phase:

$$h_\iota^{\{test\}} = (1-p) \cdot h_i \tag{2}$$

Data Augmentation involves increasing sample diversity by transforming training data, reducing the model's sensitivity to noise in the training set, and is equivalent to adding regularization constraints in the data space.

3. Experiment

3.1 Dropout Rate

This setting controls how many neurons are randomly turned off during training. A higher dropout (like 0.9) offers stronger regularization, helping to prevent overfitting but might make training take longer. A lower dropout can lead to more overfitting. By trying different dropout values, we recorded how the accuracy on the test set changed to find the best balance.

3.2 Base Learning Rate

The learning rate determines how big each step is when updating the model's parameters. If it's too high, the training can become unstable or even diverge. Too low, and training takes forever or gets stuck in a suboptimal spot. We tested different starting learning rates, looked at the loss and accuracy results, and picked the best one that worked well with our decay plan.

3.3 Coping with data

3.3.1 Preprocessing and Standardization

Before the MFCCs are fed into the neural network, they undergo a critical preprocessing step visible in your code.

3.3.2 Architectural Interpretation as a "Pseudo-Image"

This is the most crucial conceptual step. The program treats the MFCC matrix as a single-channel image:

The time steps (x-axis) are treated as the width of the image.

The MFCC coefficients (y-axis) are treated as the height of the image.

The value of each MFCC coefficient is treated as the pixel intensity.

3.3.3 Convolutional Layers (add_conv)

These layers slide small filters (e.g., 3x3) across the MFCC "image." Then we met Pooling Layers (pooling=True): These layers (e.g., max-pooling) downsample the feature maps. Finally, combating Overfitting for Audio Data, Dropout (dropout_rate=0.9): This randomly drops neurons during training, preventing the network from becoming overly reliant on any single MFCC coefficient or feature and forcing it to learn robust, redundant patterns. This is essential for generalizing to new voices and recording conditions.

Limited Iterations & Early Stopping: As your experiments showed, training for too long leads to overfitting. The model starts to memorize the exact sounds in the training set rather than learning the general patterns. Monitoring performance on a validation set and stopping early is crucial.

4. Results

The impact of dropout rate on performance is mainly reflected in overfitting prevention. Experimental results (1,000 iterations) are shown in Table 1:

Table 1: Experimental results (1,000 iterations)

Drop_out_raate	Base_learning_rate	Literation_steps	Average_accuracy
0.9	0.0008	1000	66.8%
0.9	0.00085	1000	81.3%
0.9	0.000868	1000	72.3%
0.9	0.000875	1000	60.2%
0.9	0.0009	1000	76.6%%
0.9	0.001	1000	95.4%
0.9	0.0010941	1000	74.0%
0.9	0.00109	1000	77.3%
0.9	0.0015	1000	68%
0.9	0.002	1000	62.1%

From controlled-variable experiments, we found that a dropout rate around 0.00085 achieved the best performance, with average accuracy exceeding 80%.

However, when keeping these parameters fixed and increasing iterations to 1,500, accuracy changed again, indicating that the number of iterations is a key factor in determining training adequacy. We observed that after exceeding a certain optimal number of iterations, test performance declined, a typical symptom of overfitting.

When we look at the experimental results and consider what we know about overfitting, we can clearly understand why these results happen. Overfitting happens when a model starts to memorize the noise, random quirks, and unique details in the training data instead of learning patterns that are useful for new data. This usually shows up as a clear gap between training and testing performance: the training loss keeps dropping, sometimes near zero, and training accuracy gets close to 80%. Meanwhile, test loss begins to rise, and test accuracy levels off or even declines after some initial improvement. The experimental data we have provides a good example of this issue and shows how to keep it in check. The experiments were run with a fixed dropout rate of 0.9 and 1000 training steps. They reveal how critical the choice of learning rate is—whether the model learns well or just memorizes data within a limited number of steps. - The "Perfect Balance" (Learning Rate = 0.001): The highest accuracy of 95.4% shows this is a very good setting. At this learning rate, the model moves smoothly through the loss environment, learning important features quickly enough to make good progress within 1000 steps, without becoming unstable. The high dropout rate helps prevent the model from overly depending on specific neurons, encouraging it to learn more strong features. - Too Low (like LR = 0.0008): When the learning rate is too small, such as 0.0008 which results in about 66.8% accuracy, the model learns very slowly. In just 1000 steps, it hasn't had enough time to pick up meaningful patterns, leading to underfitting. - Too High (for example, LR > 0.001): On the other hand, if the learning rate is too high, like 0.002 which gives 62.1% accuracy, the training becomes unstable. The updates are so big that they skip over the best solutions or bounce around, preventing the model from settling into a good state. This results in poor performance and ineffective learning. This data clearly shows why just running more training steps without carefully adjusting hyperparameters can do more harm than good. A wrong learning rate—either too high or too low—can cause slow, ineffective learning or unstable, diverging training. Even with a good learning rate, training for too long can lead to overfitting because the model starts to memorize noise once it has exhausted learning general patterns. Techniques such as early stopping, which stops training when validation performance worsens, and dropout, which randomly disables neurons to prevent them from co-adapting, are important tools. They help the model focus on learning general features and stop training before overfitting occurs. The high accuracy at a learning rate of 0.001 shows that the model reached a sweet spot, capturing general patterns before starting to memorize noise.

5. Conclusion

Hence, based on the characteristics of this target task of refrigerator speech interaction, take "adapt to the limited ability of edge side" as the purpose of this paper, several kinds of lightweight deep learning architecture models are proposed and verified. And the adopted lightweight deep learning architecture has an architecture composed of "4 convolutions + 2 pools + 2 full connected layer", characterized as small parameter, low energy consuming and solved the problem that traditional complicated and large-size parameter neural network speech models can not be operated normally and stable because of the limitation of hardware capabilities. On the test of the target dataset, the accuracy of speech recognition has already reached 92%, which can meet that AI refrigerators can understand voice instructions well, which proved that this designed lightweight deep learning structure has great effect on refrigerator scenes, etc. So it provides efficient and practical solution for domestic refrigerator industry. In existing model training tests in this paper, we have used a rather simple dataset with quite basic daily use scenarios, and also this field will expand furtherly in the future study. For one aspect, we collect and built the data set related to AI refrigerator strongly include the usage voice instructions. These data contain voice instructions of more common use scenarios for the AI refrigerator. For the other, we optimize and adjust the previous architecture by taking new datasets in mind, thus enhance speech recognition performance, meeting the requirement in more real scenario for refrigerators.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.
[2] Dai, X. (2024). Robust deep-learning-based refrigerator food recognition. Frontiers in Artificial Intelligence, 7.
[3] Rokhva, S., Teimourpour, B., & Soltani, A. H. (2024). Computer vision in the food industry: Accurate, real-time, and automatic food recognition with pretrained MobileNetV2. Food and Humanity, 3, 100378.
[4] Banoth, R. K., & Murthy, B. V. R. (2024). Soil Image Classification Using Transfer Learning Approach: MobileNetV2 with CNN. SN Computer Science, 5(1).
[5] Lubura, J., Pezo, L., Sandu, M. A., Voronova, V., Donsì, F., Šic Žlabur, J., ... & Voća, N. (2022). Food recognition and food waste estimation using convolutional neural network. Electronics, 11(22), 3746.
[6] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. Advances in neural information processing systems, 33, 12449-12460.
[7] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023, July). Robust speech recognition via large-scale weak supervision. In International conference on machine learning (pp. 28492-28518). PMLR.
[8] Graves, A., Jaitly, N., & Mohamed, A.-r. (2013). Hybrid speech recognition with deep bidirectional LSTM. In the 2013 IEEE International Conference on Acoustics, Speech, and Signal Processing (pp. 273–277).
[9] Wired. (2020). Worried About Privacy at Home? There's an AI for that. Wired. https://www. wired. com/story/edge-ai-appliances-privacy-at-home/
[10] Zhou Feiyan, Jin Linpeng, & Dong Jun. (2017). Review of Convolutional Neural Network Research. Journal of Computer Science, 40(6), 1229-1251.