International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Lip-to-speech Recognition using Deep Learning

Mr. Venkatesh Kumar M
Department of CSE
ACS College Of Engineering
Bengaluru, India
veenkat@gmail.com

Nisha P
Department of CSE
ACS College Of Engineering
Bengaluru, India
nishup1610@gmail.com

Puneeth V
Department of CSE
ACS College Of Engineering
Bengaluru, India pune
ethvishwamurthy13@gmail.com

Monish Gowda M
Department of CSE
ACS College Of Engineering
Bengaluru, India
monishgowda27@gmail.com

Muhammed Nabeel Musthafa
Department of CSE
ACS College Of Engineering
Bengaluru, India
nabeelmusthafa15@gmail.com

***Abstract* — You can often figure out what someone's saying just by watching their lips move, especially if it's noisy or hard to hear. We're checking out how to create speech that sounds real just from watching someone's lips. In the real world, folks use context clues and what they know about the speaker to get what they're saying. So, instead of just letting a computer figure it out on its own, we're trying a more personal way. We want to figure out how to match lip movements to speech for actual people who might use all sorts of words and sentences. To help with this, we made a big set of data that's like real life and let everyone use it. We tried a few ways to test our method – numbers, what users thought, and how people rated it. We saw that our method is easier to understand and makes more sense than other ways to turn lip movements into speech.**

I. INTRODUCTION

Usually, people can figure out what someone's saying by just watching their lips. This is super helpful when it's quiet or you can't hear very well. What we're trying to do is make speech just from watching someone's lips move. Basically, we want to make spoken words just from how your lips move, which is really hard. We know that how you read lips changes depending on who's talking, how they talk, and what's going on around you. So, any good lip-to-speech thing needs to think about all that. Ideally, we want to make something that can turn lip movements into speech, even if we've never seen that person's lips before

For folks who can't hear well, and for older people, lipreading is really important. They often watch how people's mouths form words, with or without sound. For lip movements to be useful, they need to be easy to see and consistent. How your face moves, especially your lips and the muscles around them, helps a lot in understanding what someone means. But the tricky thing is that lip movements aren't always clear. Different sounds can look alike on your lips, so it's hard to know exactly what's being said. Things that help you see lip movements better can make speech recognition better too, so lip-based talking works better.

Since smartphones have good cameras now, we can use video to get more info. Lip reading is when you guess what someone's saying by watching their lips. But getting speech right from just lip movements is hard. One of the big problems is that some sounds look the same on your lips, even when they sound different. For example, the way you make the p sound in park looks almost like the b in bark and the m in mark. That makes it hard to tell words apart just by looking, which shows how tricky it is to turn lip movements into speech.The adoption of such a decentralized and intelligent access control model aligns with the vision of future-proof healthcare systems: systems that prioritize patient privacy, system resilience, and operational efficiency. By promoting transparency, reducing administrative overhead, and minimizing security risks, this project aims to redefine trust in healthcare data management, ensuring that sensitive medical information remains secure, accessible, and patient-controlled even as healthcare systems evolve in an increasingly digital world. Furthermore, the project's hybrid blockchain-cloud architecture addresses traditional scalability and storage cost issues, setting a new standard for secure, scalable healthcare IT solutions.

II. HARDWARE REQUIREMENTS

What you need for building the app differs from what users need. To develop it well, a machine must handle coding, testing, and running tools without lag. Storage should be 500 GB or larger - either HDD or SSD - to fit software kits, code bases, example data, and saved work. Running several programs at once works better when memory hits 8 GB or above. Efficiency during builds improves noticeably with that much RAM available. Fast chips, think 2.5 GHz or more, keep up when building code, running models, or working live. On top of that, a separate graphics card like an NVIDIA GTX 1650 - or better - helps power through visuals and smart algorithms, particularly while managing videos or AI math.

What users need sets out what tools work best to run the app on their gadgets. Starting from Android 8.0, called Oreo, newer phones will handle it just fine. To get going, one has to have the Lip2Speech app ready on the device. Instead of sound, it uses quiet or unclear video clips showing mouth motions. From those, spoken words come through clearly in audio form. With everything lined up right, things keep working smoothly whether testing or daily use.

III. SOFTWARE REQUIREMENTS

1. OpenCV: OpenCV's a big free toolkit used for spotting things in pictures and videos, plus doing smart tasks like learning from data. Instead of just one way, it works with several coding languages - think Python, C++, or Java. This tool handles visuals, whether stills or moving clips, to detect stuff like people's faces or handwritten notes. Pairing it with other tools boosts what you can pull off; take Numpy, built for

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

fast math crunching - it teams up smoothly, expanding what's possible without extra hassle.
2. Makepg: Makepg helps put together software so it works with Pacman. This tool runs tasks automatically when creating packages. Besides grabbing code from online, it checks if files are legit. Also verifies what's needed before setting up options for compiling. While adjusting how things get built, it handles turning source into working programs. Instead of manual steps, it sets up a mini system to place compiled parts. Custom tweaks go in at this stage without extra effort. Afterward, it adds details about the package being made. Finally, everything gets packed neatly into an installable format.

3. azel: Bazel's a tool you can use without paying, helping automate how software gets built and checked. This system isn't basic - it handles complex tasks well when making programs. It works smoothly on big jobs, thanks to tools made for scaling up workloads. You can create app bundles ready to roll out, whether targeting Android or Apple devices. Here's what it offers: Support multiple languages. Support multiple platforms. Support multiple repositories. Support high-level build language. Fast and reliable.

4. Docker Containers: Docker offers tools you can use to run apps in isolated spaces known as containers. These rely on your operating system's built-in features instead of full machine emulation. You're able to access basic functions at no cost, while extra perks require payment. Behind the scenes, a program named Docker Engine manages where these containers live and how they operate. A container works like a neat little box holding your program plus everything it needs, letting it run smoothly no matter where you move it - whether on a laptop or a big server. Each setup stays consistent because nothing gets left behind when shifting around. A Docker container image holds software in a small, independent bundle you can run right away - it's got the code along with what it needs to work, like tools, libraries, and config files, while using minimal resources because it shares the host OS kernel instead of running a full machine.

5. Flutter Framework: Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.

6. Flask: Flask's a small web tool made with Python. Being called a microframework means it skips needing specific tools or extra software. No built-in system handles databases, checks forms, or includes parts others usually pack in by default. Still, you can plug in extras to boost your app - works just like native pieces. Extensions fit right in without hassle.

7. TensorFlow: TensorFlow's an open platform built for machine learning from start to finish. Researchers use its wide collection of adaptable tools, while devs rely on shared code and support from the crowd. Instead of just one job, it handles many - especially teaching and running deep neural nets. Its network helps people explore new ideas plus launch real-world apps without hassle.

IV.SYSTEM DESIGN

Systems design means setting up how a system's parts - like structure, pieces, connections, and information - work together to meet goals. It's like using big-picture thinking when building new tech stuff. Companies offering these services figure out what kind of setup fits best by mixing computers, programs, and networks. Instead of just picking tools at random, they guide customers toward smart choices based on needs. After planning comes building: putting it all together, testing, then rolling it out properly.

SYSTEM ARCHITECTURE

The way a system is set up shows how it's organized and works. It also lets you look at it in different ways. On the other hand, a description of this setup gives you a clearer, more detailed idea of the actual system, so you can understand how it's built and how it acts overall. A use case diagram is a simple way to see how someone uses a system. It shows what the user does and how they work with different parts of the system. This kind of diagram also helps you figure out who all the users of a system are and what they do. Use case diagrams usually go along with other diagrams to give you a fuller view of what the system does

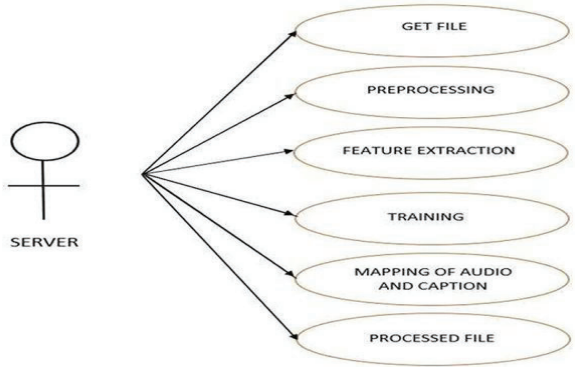

Fig 1.1 Use Case Diagram – Server

The way a system is set up shows how it's organized and what it does. It lets you look at it from different angles. An architecture description gives you a better idea of how the thing actually works, so you can get the gist of how it's built and how it acts. A use case diagram is a plain picture of how someone uses a system. It shows what they do and how they mess with stuff in the system. It also comes in handy for figuring out who uses the system and what they do with it. They're usually used with other diagrams for a full view of what the system can do. The app lets you drop in a video file. When it's in, the video heads to the server to get worked on. The server pulls out the main stuff from the video, fires up the trained model, and spits out a better version with sound and words. When it's all done, the souped-up video gets saved and sent back to you, and you can watch the finished video with sound and captions.
A sequence diagram shows how different parts act and talk to each other in order. It points out the classes and things that are part of something and shows the messages they send back and forth to get something done. Sequence diagrams usually go with use cases and show up in the system's brain. People call them event diagrams or event scenarios. When you pick a video and upload it with the app, the file goes to the server to get processed. The server grabs the part of the video with the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

lip movements and other visual stuff, then works with a trained deep learning model to make the text and sound. The server lines up the lip movements with the speech by using its sequence to sequence framework. After everything's done, the improved video with sound and captions is saved and sent back to you through the app.

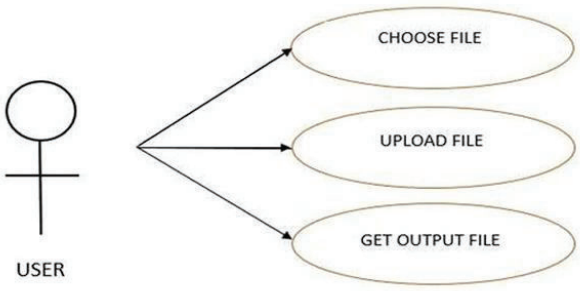

Fig 1.2 Use Case Diagram – Website

METHODOLOGY FOLLOWED

Methodology is just the way a group organizes its stuff—systems, parts, data, all that—to get things done right. It's like using a systems-thinking approach when you're making new stuff. This kind of service is really helpful if your company makes computer systems that mix hardware, software, and communication stuff. Methodologists can assist you in choosing the best hardware and software for a project, then they can design, set up, and launch the whole system for you.

System architecture is all about the different parts of a system and how they're made to work together. Think of it as one big team. To talk about these architectures, experts came up with special languages called Architectural Description Languages (ADLs). When planning a system's architecture, you figure out the main parts and how they relate. First, you decide what kind of system it is. Then, you create an architecture that handles how those parts talk to and work with each other. By the end of this planning stage – it's like making a construction model – you have a structural plan of the product. This plan shows how the system will be built and gives a summary of what it will do.

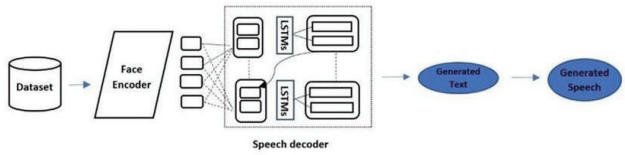

Fig 1.3 Methodology

V. IMPLEMENTATION

This tech turns silent lip movements in videos into text and audio you can understand. It works by taking the video and chopping it up into individual pictures at 30 pictures per second. This breakdown is the first step for analyzing the video and teaching the system how to translate lip movements.

First, each video frame gets processed. We use Dlib's tech to pinpoint 68 facial landmarks, which helps us find the face and mouth. To make our model work better with different speakers , we mess with the data a bit – cropping, tweaking brightness, flipping, and rotating the images. This adds some mix to the data. After prepping the data, we jump into the visual recognition part.

A 3D CNN looks at how the lips move over time. Video clips are made the same length to keep things consistent. These feature cubes show the small details of how the lips change as someone speaks. For the sound part, SpeechNet creates a spectrogram feature cube by pulling out Mel frequency energy stuff from the audio data. These features show the acoustic patterns of speech. Then, the system picks up on these patterns and starts to match lip movements with the correct speech sounds. This lets the system make links between what the lips are doing and what's being said.

VI. RESULTS

The Lip-to-Speech and Text project is all about making a system that turns silent lip movements from videos into text and speech you can actually get. It uses a smart deep learning model to look at lip movements and then makes the matching text and sound. By reading these lip movements, the system can make spoken output, even if the original sound is gone.

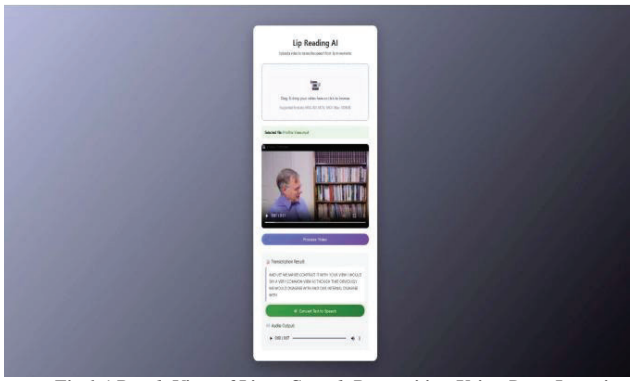

Fig 1.4 Result View of Lip to Speech Recognition Using Deep Learning

We're trying to make current tech better by adding more words, making the speech clearer, and getting the sound to be more natural, we're hoping it will make speech that's easier to understand and sounds just like the person's real voice.

We also want to put this system on a phone app. People could upload videos with bad or no sound and get back a version with synced speech and subtitles. Basically, we want to create lip-to-speech and text tech that's dependable, spot-on, flexible, and simple to use.

Earlier approaches like predictive coding, manual feature design, or basic networks often generate robotic or lifeless speech. This work instead uses advanced sequence-to-sequence models, adding convolutional layers, memory units, and stronger cleanup steps before processing. These upgrades help capture irregular mouth movements, timing quirks, and subtle hints earlier systems tended to overlook.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

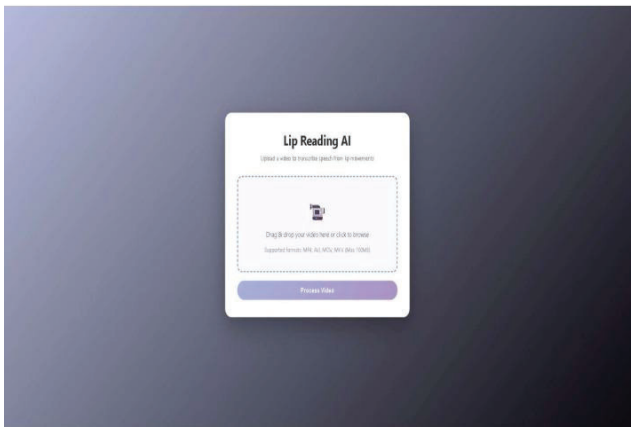

Fig 1.5 Landing Page of Lip to Speech Recognition Using Deep Learning

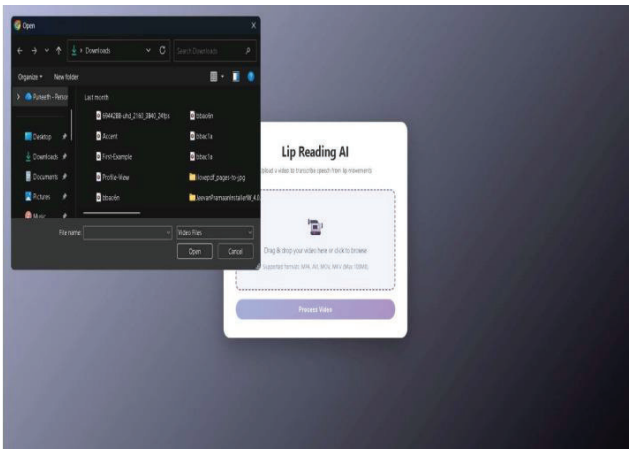

Fig 1.6 Uploading a Video File View in the Website Application

We also want to put this system on a phone app. People could upload videos with bad or no sound and get back a version with synced speech and subtitles. Basically, we want to create lip-to-speech and text tech that's dependable, spot-on, flexible, and simple to use.

VII. CONCLUSION

The "Lip to Speech and Text Tech" effort tackles a tricky slice of visual speech rebuilding clear spoken words using only quiet mouth motions. I often wonder how folks pull this off naturally in noisy spots or dead silence, but teaching machines to copy it needs heavy-duty models, way more data, along with smart feature grabbing that won't fail when things get chaotic. This project made us tackle each issue step by step, then find ways to go from live video straight to sound matching the speaker's intent. My numbers show it wasn't about hitting some perfect score but narrowing the tough gap between movement and message - crafting a tool that turns mouth movements into words folks believe. People who can't hear well or have trouble speaking get an actual chance to keep up with loved ones, coworkers - so they don't end up left behind when talk gets going.It pushes folks to focus on how they move their mouth and speak, so over time they start talking more clearly - without realizing it's happening right away. Lipreading helps folks join in easier, so fewer end up left out when hearing's an issue - especially when they'd otherwise sit quiet at the edges. Instead of missing cues, seeing words gives a clearer shot at keeping up without feeling sidelined by noise or fast talk.

The research keeps highlighting similar issues - models struggle with words that sound alike, tiny data sets, jerky head motions, plus the fact videos lack audio cues. Earlier approaches like predictive coding, manual feature design, or basic networks often generate robotic or lifeless speech. This work instead uses advanced sequence-to-sequence models, adding convolutional layers, memory units, and stronger cleanup steps before processing.

The other aims or functions of Lip to Speech are: Creating a system that adjusts to various voices without starting over each time. Handling bigger vocabularies so the system doesn't freeze when someone uses less predictable words. Making fewer mistakes when light varies, positions shift - also cutting glitches from shaky footage. Creating talk that flows better, more like actual humans speak, without sounding robotic.

REFERENCES

[1] riantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. The conversation: Deep audiovisual speech enhancement. arXiv preprint arXiv:1804.04121, 2018.

[2] riantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. Deep lip reading: a comparison of models and an online application. In INTERSPEECH, 2018.

[3] riantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. Lrs3-ted: a large-scale dataset for visual speech recognition. arXiv preprint arXiv:1809.00496, 2018.

[4] assan Akbari, Himani Arora, Liangliang Cao, and Nima Mesgarani. Lip2audspec: Speech reconstruction from silent lip movements video. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2516–2520, 2017.

[5] oon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3444–3453. IEEE, 2017.

[6] on Son Chung and Andrew Zisserman. Lip reading in the wild. In Asian Conference on Computer Vision, pages 87– 103. Springer, 2016.

[7] artin Cooke, Jon Barker, Stuart Cunningham, and Xu Shao. An audio-visual corpus for speech perception and automatic speech recognition. The Journal of the Acoustical Society of America, 120(5):2421–2424, 2006.

[8] Da id A Ebert and Paul S Heckerling. Communication with deaf patients: knowledge, beliefs, and practices of physicians. Jama, 273(3):227–229, 1995.

[9] Ariel hrat, Tavi Halperin, and Shmuel Peleg. Improved speech reconstruction from silent video. 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pages 455–462, 2017.

[10] Ariel Ephrat and Shmuel Peleg. Vid2speech: speech reconstruction from silent video. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5095– 5099. IEEE, 2017