PROJECT PORTFOLIO

Jongha Kim
Department of Electronic Engineering
Inha University
tomato949@naver.com
https://github.com/hytric

CONTENTS

1 AUDIO TO AUDIO UNIT TRANSLATION

1.1 BASE PAPERS

[1] Kim, M., Choi, J., Kim, D., & Ro, Y. M. (2023). Textless Unit-to-Unit Training for Many-to-Many Multilingual Speech-to-Speech Translation. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 32, 3934-3946. Kim et al. (2023)

[2] Choi, J., Park, S. J., Kim, M., & Ro, Y. M. (2024). AV2AV: Direct Audio-Visual Speech to Audio-Visual Speech Translation with Unified Audio-Visual Speech Representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (pp. 27315–27327). Choi et al. (2024)

1.2 MOTIVATION

Current audio-visual translation methods typically involve several sequential steps:

- **STT (Speech-to-Text)**: Converts original audio-visual speech into text transcription.
- **Machine Translation (MT)**: Translates the transcribed text into the target language.
- **TTS (Text-to-Speech) and Video Generation**: Synthesizes speech audio and generates corresponding video (e.g., lip movements) from the translated text.

To address the limitations of these multi-step pipelines, prior research has proposed an "End-to-End" approach. This approach directly translates input audio-visual speech into the target language's audio-visual speech, thereby eliminating intermediate text and speech processing steps.

However, these papers only focused on and tested translations for SVO (Subject-Verb-Object) languages, such as English, French, Spanish, and Chinese. They excluded SOV (Subject-Object-Verb) languages like Korean and Japanese.

The overall system architecture is illustrated in Figure 1. Please refer to this figure to understand the complete pipeline.

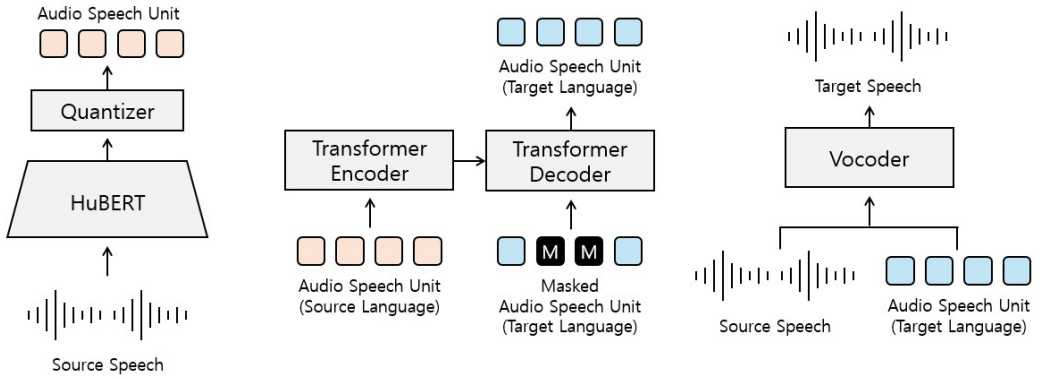

Figure 1: Figure. 1: Architectural diagram of the proposed Audio-to-Audio Speech Unit Translation system. It illustrates the three main stages: (left) extracting discrete audio speech units from source speech using HuBERT and a Quantizer, (middle) translating source language units to target language units via a Transformer Encoder-Decoder, and (right) synthesizing the target speech from the generated units and source speech using a Vocoder.

1.3 CONTRIBUTION

- **Code Reproduction**: We reconstruct the model architecture, training parameters, and preprocessing steps by referencing the original papers. Through actual training experiments, we optimize the implementation for better performance.

- **Language Expansion**: We propose methods to expand the translation capabilities to SOV (Subject-Object-Verb) languages like Korean and Japanese, which have different grammatical structures from SVO languages.

1.4 Proposed Method

1.4.1 Dataset

- Multilingual AIhub data: AIhub Dataset
- LibriSpeech: LibriSpeech Dataset

1.4.2 Framework

- Fairseq and speech-resynthesis
- PyTorch

1.4.3 Hardware Specification

- Nvidia A6000 48GB GPU

1.4.4 Model Specifications

Model	**Audio input**	**HuBERT**	**Transformer**	**Vocoder**
Specifications	• Segment Size: 8960 • Code Hop Size: 320 • Sampling Rate: 16000 Hz • FFT Size: 1024 • Hop Size: 256 • Window Size: 1024 • Frequency Range: 0-8000 Hz • Number of Mel Bands: 80	URL	URL	URL

1.4.5 HuBERT

The detailed structure of the HuBERT model is shown in Figure 2. Please refer to this figure to understand the internal architecture.

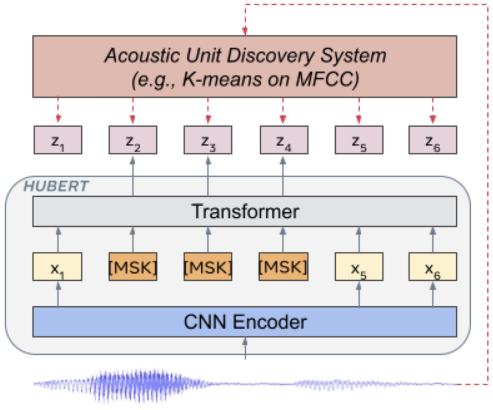

Figure 2: Hubert model structure

1. **K-means Clustering on MFCC Features:** We perform K-means clustering on the extracted MFCC (Mel-frequency cepstral coefficients) feature vectors.
2. **Using as "Pseudo-Labels" or "Target Units":** These clustered results are then used as "pseudo-labels" or "target units." This helps us find repeated sound patterns (like basic

speech sounds or phonemes) within the speech data. We then turn these patterns into distinct, countable units. The initial HuBERT model uses these units as its "correct answers."

3. **Masking and Prediction:** Next, we apply masking. The model then learns to predict the K-means cluster ID for the masked parts of the speech, using the surrounding unmasked speech information.
4. **Loss Function:** We train the model using a classification loss, like Cross-Entropy loss.
5. **Feature Extraction for HuBERT (mHuBERT):** Speech features are taken from the 11th layer of the HuBERT (or mHuBERT) model.
6. **Quantization to Speech Units:** These features go through K-means clustering and are turned into 1,000 distinct speech units, which act like a fixed vocabulary size.
7. **"Ours" Model:** Our specific model ("Ours") uses 500 units because, based on our experience, this number gives the best distribution of these units.

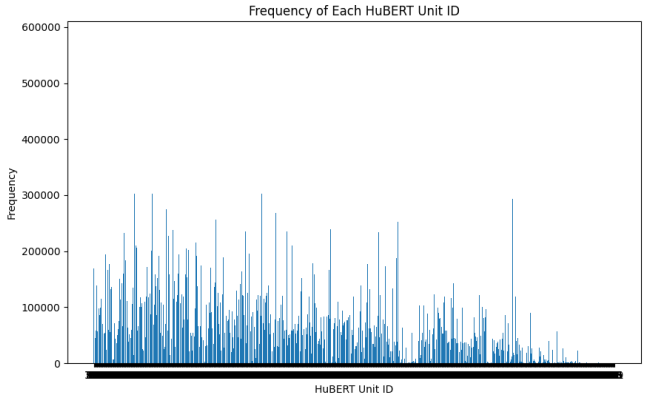

Figure 3: English Hubart output, 500 unit distribution

1.4.6 VANILLA TRANSFORMER

The **Transformer model** can translate between multiple languages, based on unit-to-unit translation methodology [15] Lee et al. (2023).

- **Language Identification**: To know which language to translate from, the model uses **special language tags** (like language tokens) at the beginning of a sentence, such as `<en>` for English or `<kr>` for Korean. For instance, in English, you'd input `<en> unit1, unit2, ....`
- **Translation Direction**: To specify the target language (e.g., if you want to translate into English, you'd input `<en>`), you provide this token to the *decoder* part of the model. This tells the model to start translating into that specific language.
- **Learning Method**: The model learns by using a **masking technique**. Masking involves hiding certain words in a sentence and having the model predict what those hidden words should be. This helps the model understand the context and learn to translate more accurately.

The Transformer architecture used in our system is illustrated in Figure 4. Please refer to this figure for architectural details.

1.4.7 VOCODER

The **Vocoder** is like a voice maker. Its main job is to turn voice "units" into actual sound waves.

- **Basic Structure**: It mainly follows the design of **HiFi-GAN**, which is based on voice units. This means it creates raw sound waves (waveforms) from translated AV voice units.

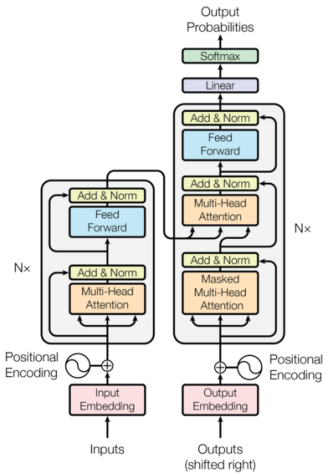

Figure 4: Vanilla Transformer structure

- **Speaker Voice Extraction**: It takes the sound of an existing voice (as a Mel spectrogram) and makes a single **speaker embedding**, called a **d-vector** [97]. This helps it copy the speaker's voice quality.
- **Speaker Info Added**: This extracted d-vector is attached (*concatenated*) to each feature vector of the voice units.

$$
\begin{aligned}
[\text{ Voice Unit 1 Feature }] &\rightarrow [\text{ Voice Unit 1 Feature } | \text{ d-vector }] \\
[\text{ Voice Unit 2 Feature }] &\rightarrow [\text{ Voice Unit 2 Feature } | \text{ d-vector }] \\
[\text{ Voice Unit 3 Feature }] &\rightarrow [\text{ Voice Unit 3 Feature } | \text{ d-vector }] \\
&\dots \\
[\text{ Voice Unit N Feature }] &\rightarrow [\text{ Voice Unit N Feature } | \text{ d-vector }]
\end{aligned}
$$

- **Sound Wave Creation**: These features, now combined with speaker info, are sent to the standard **HiFi-GAN** [92] to finally create the sound waves.

The complete vocoder architecture is shown in Figure 5. Please refer to this figure to understand how voice units are converted to speech.

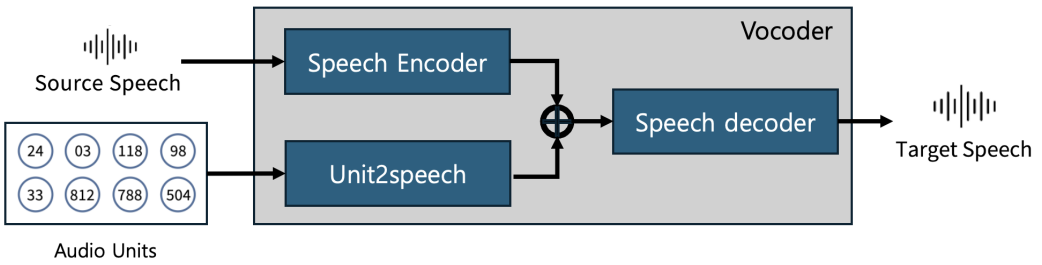

Figure 5: Vocoder model structure, Unit2speech is Hifi-GAN

Duration Predictor a module that guesses how long each voice unit should last.

- This is used to make sure data like text, video, and audio stay in sync.
- **Structure**: It's like the duration predictor in TTS (Text-to-Speech) models. It has two 1D convolution layers and one classifier.
- **Our Approach**: To copy the voice quality (timbre) of the input audio, our vocoder takes the original audio as input. To keep these two parts in sync, we added the Duration Predictor, which was suggested in the original Av2av paper.

- However, it didn't actually make the performance better. In fact, it caused the speech to sound stretched out.
- So, we did not apply it.

1.4.8 SVO (SUBJECT-VERB-OBJECT) AND SOV (SUBJECT-OBJECT-VERB) CONVERSION

Our approach to handling different language structures is shown in Figure 6. Please refer to this figure to understand the conversion process.

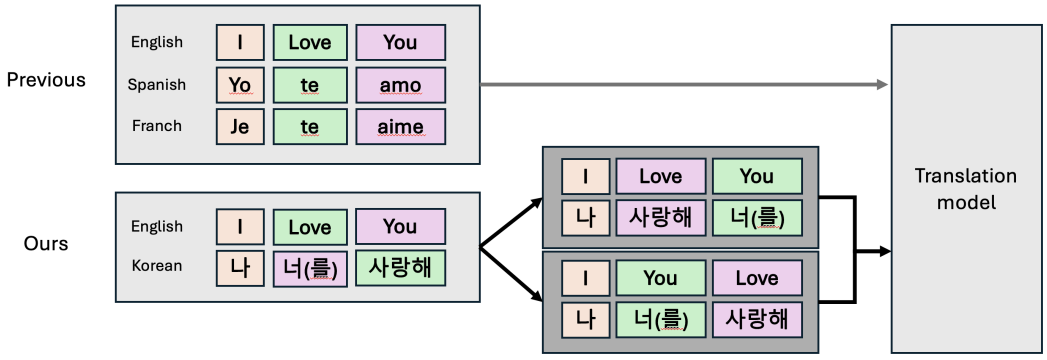

Figure 6: Our approach, SVO and SOV Conversion

- When we trained our model with just Korean-English pair datasets, without changing the data format, we faced a problem: the word sounds became muffled or unclear.
- This happened because the model struggled with matching words between the two languages during translation.
- By using mutual conversion (like SVO , SOV), the Transformer model focused less on finding words with the same meaning and more on matching sounds in the same time frame. This made the learning task simpler for the model.

Our Approach

So, we used an **LLM (Large Language Model)**, specifically **Llama 3.1 8B Instruct** [5] Touvron et al. (2023), to change the word order. We did this using the voice transcripts we already had in our dataset.

After that, we used the **"Google Text-to-Speech" model** to turn these new texts into audio. We then used this new audio as our training (TR) dataset. The vocoder used in this process is based on HiFi-GAN [10] Kong et al. (2020).

The dataset pairing structure is illustrated in Figure 7. Please refer to this figure to understand how the training pairs are constructed.

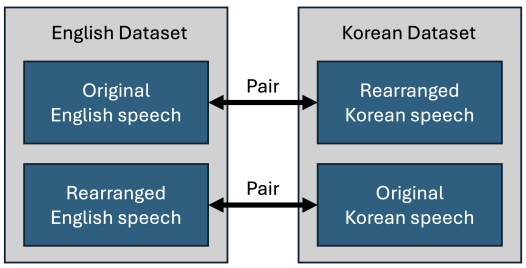

Figure 7: Audio Dataset Learning Pairs

Original Speech Script	Rearranged Speech Script
In particular, he thinks visual images work in symbolic function like language.	In particular, he visual images language like symbolic function work.
"특히 그는 시각 이미지가 언어와 같은 기호적 기능을 한다고 본다."	"특히 그는 생각한다 시각 이미지가 일한다고 기호적 기능 언어처럼".

Table 1: Comparison of Original and Rearranged Speech Scripts

Model	BLEU	COMET
Textless NLP	66.9	0.1338
AV2AV	60.1	0.1587
Our Model	42.8	0.1009

Table 2: Model Performance Comparison

1.5 METRIC

- **Korean Language Characteristics and Current Evaluation Method Issues**: The **Korean language is very different from English**. Korean words are often made of smaller meaning units (morphemes), and word forms can change a lot based on grammatical markers like particles or endings.
- **Lack of Similar Models for Comparison**: There are **no other English-Korean S2ST (Speech-to-Speech Translation) models** currently available. Because of this, we don't have a direct comparison for our model.
- **Conclusion on Scoring**: Due to these two conditions, the **existing scoring methods are not suitable for Korean**. Therefore, we measured the score only using the **generated English speech**.

1.6 CONCLUSION

- The output speech isn't perfect, but it's understandable.
- The translated speech doesn't have a steady pitch, and there's a lot of background noise.
- It should work better if trained with more diverse Korean voices.

1.7 FUTURE WORKS

- **Need for a Model Reflecting Korean Characteristics**: Due to the nature of the Korean language, the same word can sound different based on its position, or its form can change significantly due to particles and endings.
- **Verb Position Challenge**: The position of the verb is the point of greatest variation. A method is needed to easily incorporate verbs that are far apart in two different word orders due to sentence structure.
- **Additional Metric Provision**: For Korean, evaluation is primarily done using phoneme-based metrics. Therefore, we need to provide additional metrics that reflect this.

1.8 PAPER REVIEW

- paper review: **HuBERT** https://hytric.github.io/paperreview/HuBERT/
- paper review: **UTUT** https://hytric.github.io/paperreview/UTUT/
- paper review: **AV2AV** https://hytric.github.io/paperreview/AV2AV/
- paper review: **HiFi-GAN** https://hytric.github.io/paperreview/HiFi-GAN/

2 Scene Graph to Video Generation

2.1 Base Papers

[3] Dhariwal, P., & Nichol, A. (2022). Diffusion-Based Scene Graph to Image Generation with Masked Contrastive Pre-Training. arXiv preprint arXiv:2201.00308. Dhariwal & Nichol (2022)

[4] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10684–10695. Rombach et al. (2022)

2.2 Motivation

Scene Graph (SG) is a method to represent objects within a visual scene, such as an image or video, and the relationships between those objects, in a structured graph format.

An example of Scene Graph representation is shown in Figure 8. Please refer to this figure to understand the concept.

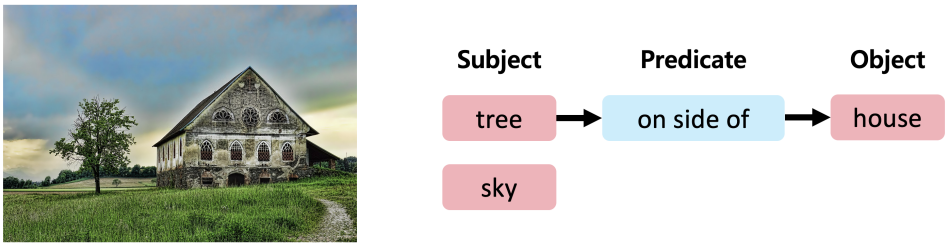

Figure 8: Scene Graph example

It describes how objects within an image are related to each other, similar to how humans understand an image. Currently, **Image Generation (IG)** models that utilize scene graphs already exist [14] Yang et al. (2022), and several related research papers have been published. However, this SG idea has not yet been properly applied to **Video Generation (VG)**.

We are focusing on this research gap and aim to apply scene graphs to video generation. To date, there has been research on generating general images using scene graphs, but there is no existing research specifically targeting general videos.

2.3 Contribution

- If we can create video in a way that reflects how humans think, it's like gaining the ability to directly direct scenes.
- Because this feature is a very important advantage for long video generation, we additionally propose a module that generates long videos.

The comparison between previous approaches and our method is illustrated in Figure 9. Please refer to this figure to understand the architectural differences.

2.4 Proposed Method

Dataset:

- Action Genome

Framework:

- Pytorch

Hardware spec

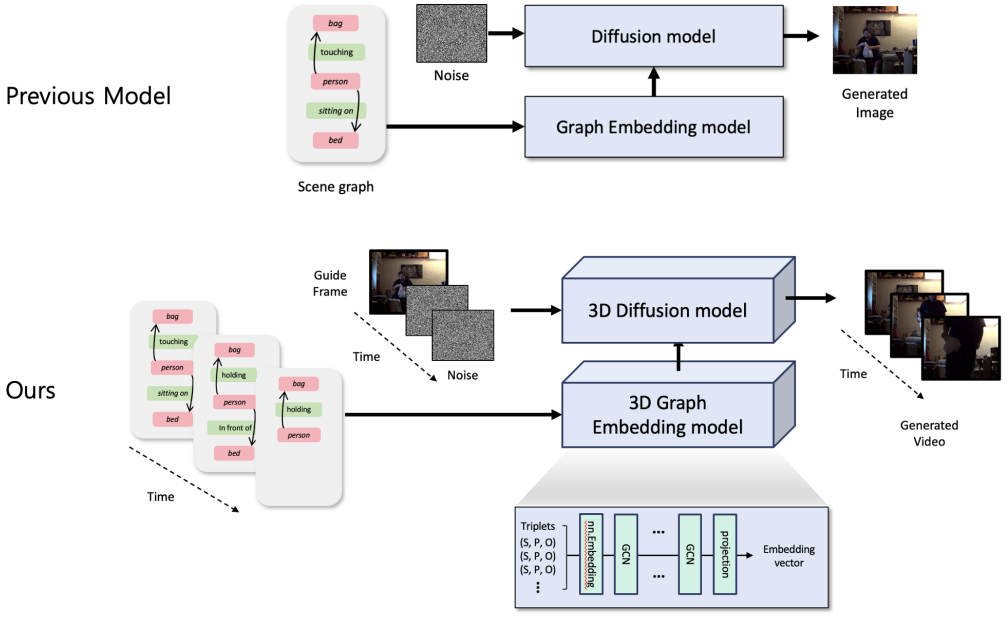

Figure 9: Previous (top) and Ours (bottom): SG2video Model structure

- Nvidia A6000 48G gpu

2.4.1 SG EMBEDDING

- Utilizing SG Embedding Model Structure Proposed in Existing Image Generation Models: Relational Graph Convolutional Network (R-GCN) [6] Schlichtkrull et al. (2018)
- R-GCN is a method that learns directed graphs using a CNN.
- In the graph convolution net, Triplets: (Subject, Predicate, object) are separated and learned.
- In an intermediate step, triplets connected to one word are combined through an Object-wise Pooling layer.

The SG embedding model architecture is shown in Figure 10. Please refer to this figure to understand the embedding process.

Finally, we use SGClip: Learns Image Encoder (Clip model) and Graph Encoder with contrastive learning [7] van den Oord et al. (2018).

2.4.2 DIFFUSION MODEL

The Stable Diffusion model structure is illustrated in Figure 11. Please refer to this figure to understand the diffusion process.

- We condition LDMs either via concatenation or by a more general cross-attention mechanism [8] Ho et al. (2020).
- Plays a decisive role in increasing computational efficiency by using the Latent Space concept from VAE.
- U-Net's backbone, an encoder-decoder structure that has shown good performance in image segmentation, etc., is suitable for effectively learning and transferring features at various scales.
- A Generator that can more flexibly condition the Diffusion Model by utilizing the Cross-Attention mechanism.
- For video generation, expands the time axis, extends the structure to 3D.

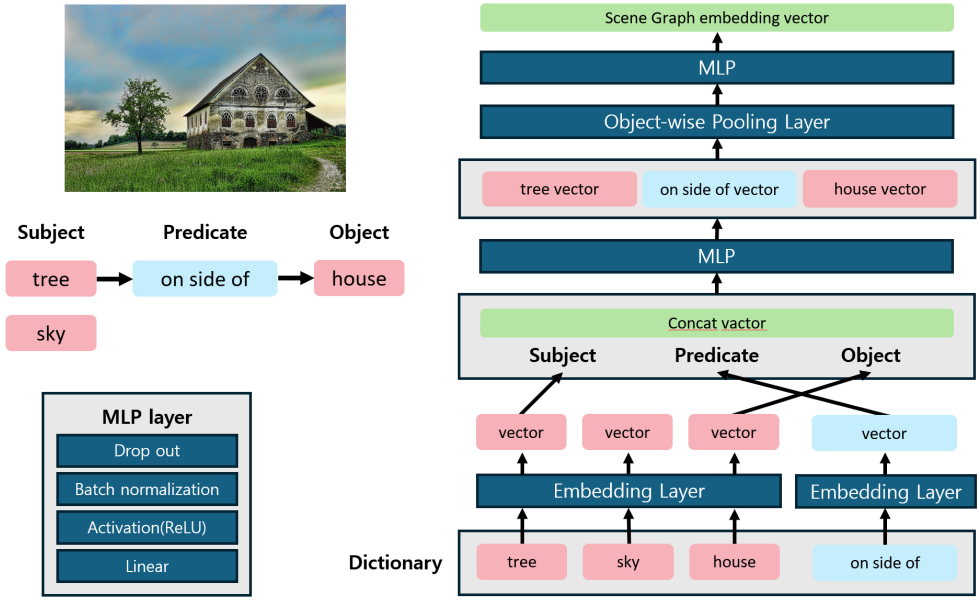

Figure 10: SG Embedding Model structure

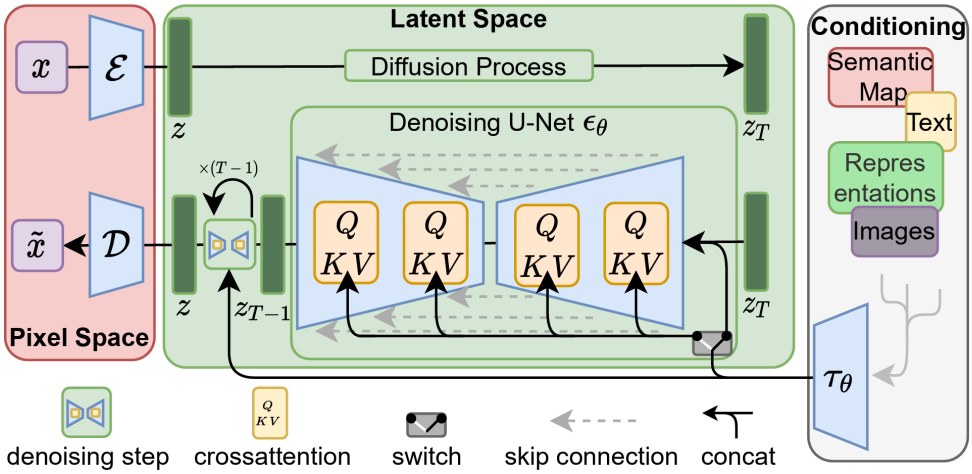

Figure 11: Stable Diffusion Model structure

2.4.3 Long Video Generation

- As input to the model, existing diffusion models put a noise frame into every frame. However, for autoregressive video generation, it is applied to the very first frame.

2.5 Conclusion

- The results were not as good as we had hoped.
- All features are implemented in the code.
- There aren't enough datasets that pair Scene Graphs with videos.
- The Scene Graphs didn't always match up well with the video content (this is called poor alignment).

2.6 FUTURE WORKS

- Instead of relying on standard Scene Graph datasets, we plan to extract "task graphs" directly from video content (e.g., from instructional videos like HowTo100M [13] Miech et al. (2019), by analyzing captions or key steps). These video-mined task graphs will condition our diffusion model for video generation. A key challenge is effectively capturing and representing their temporal sequences.
- Leveraging Richer Knowledge Graphs (like ConceptNet): Current Scene Graph-video datasets are often low quality. To address this, we propose using broader knowledge bases like ConceptNet [11] Speer et al. (2017) (or similar KGs built from video metadata) to create more robust graph embeddings. These enriched embeddings, potentially using models like ALBERT [12] Lan et al. (2019) with a Knowledge Graph Encoder, will provide stronger, more meaningful conditions for training our video generation models, moving beyond simpler visual scene graphs.

3 Predicting Water Quality: Embedded Device Development and Sensor Data Analysis

3.1 Motivation

Our goal is to analyze water quality data to predict pollution levels, enabling proactive water quality management. We are particularly focused on predicting water quality in extreme conditions, such as those found in wastewater treatment plants. Here's how we plan to achieve this and the benefits:

- **Cost Reduction with Soft Sensors:** Instead of relying on expensive physical water quality measurement equipment, we will use "soft sensors" (virtual sensors powered by AI predictions). This approach significantly reduces costs.
- **Enhanced Process Control:** By measuring water quality indicators at intermediate stages where physical sensor installation is difficult, we can achieve more detailed and precise control over chemical dosing and overall process adjustments.

The system overview is shown in Figure 12. Please refer to this figure to understand the complete monitoring system.

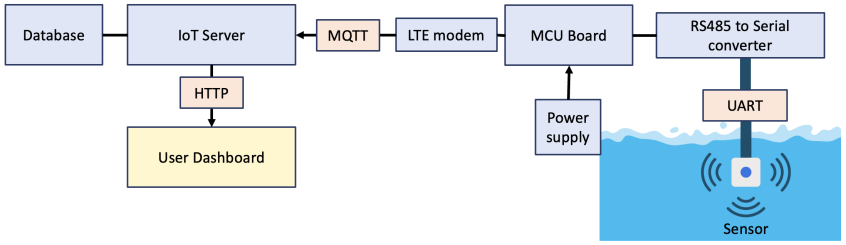

Figure 12: Water quality prediction system overview

3.2 Contribution

Our contributions include:

- Designed and implemented an IoT system for water quality monitoring with:
  - Hardware featuring multiple sensors and LTE connectivity
  - Firmware development using FreeRTOS
- Developed a prediction model that:
  - Combines trend analysis and noise modeling
  - Accurately forecasts water quality indicators using collected sensor data

3.3 Embedded System Design

We built an embedded device (using an MCU) to collect water quality data. The device's design included an LTE module, water quality sensors, internal serial communication components, and a power supply. We developed the firmware (the software for the device) using **FreeRTOS**. FreeRTOS is a real-time operating system (RTOS) kernel specifically designed for microcontrollers and small microprocessors. A key thing with embedded systems is they need to work all day, every day without errors. That's why we focused hard on writing good code for when things go wrong and for when the system needs to reboot. For reference, see Figure 13 for the RS485 converter used in our system.

Figure 13: RS485 to serial converter picture

3.4 Model Structure Design

Sensor data is time-series data, meaning the order of the information over time is very important. However, it often includes a lot of irregular fluctuations or 'noise,' which can make it difficult to analyze effectively. This means we need a way to handle these noisy parts. To address this, we developed an approach using an **ensemble method** with two separate models:

- A model to predict the main trends (the 'big picture' or 'larger flow'): For this model, predictions can be made using averaging and smoothing techniques, or by using standard time-series based models. This is because trends in nature are less impacted by volatile, short-lived values and more closely reflect the general direction over extended time.
- A model to predict the effects of random noise: For this part, we found that even a simple Multi-Layer Perceptron (MLP) – without using time-series information – showed excellent performance in predicting these random noise components. The reason is that noise tends to be more strongly influenced by present conditions rather than by the simple passage of time.

The ensemble model output is illustrated in Figure 14. Please refer to this figure to understand how trend prediction and noise modeling are combined.

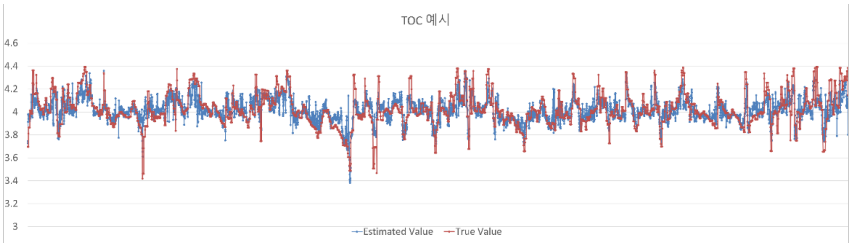

Figure 14: Ensemble model output combining trend prediction and noise modeling for water quality forecasting

We used **AutoML (Automated Machine Learning)** to automatically find the optimal hyperparameters (settings) for each model. This automation significantly reduced the time required to identify the best model configuration. Our approach draws inspiration from deep learning methodologies [16] Goodfellow et al. (2016).

The AutoML hyperparameter optimization results are shown in Figure 15. Please refer to this figure to understand the optimization process.

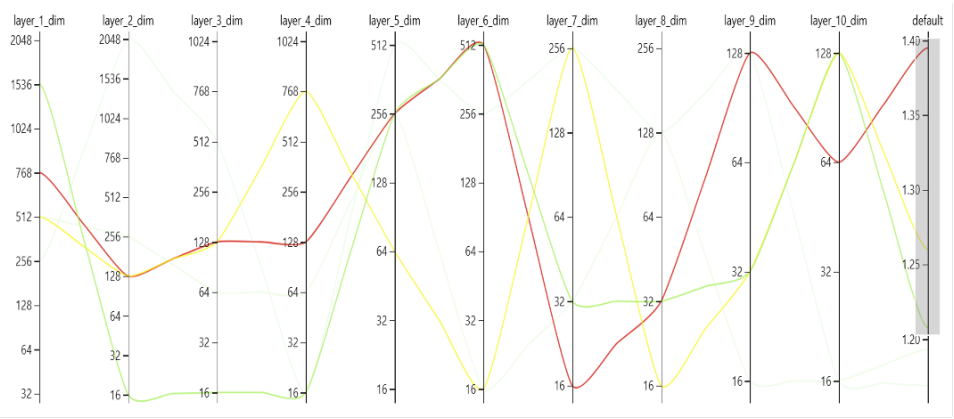

Figure 15: AutoML hyperparameter optimization results

3.3 Conclusion

- We successfully delivered a prototype (an early version of the product) to our client.
- The client is now using our predicted values to help manage their water quality.

3.3.1 Evaluation

Our prediction error (NMAE) improved significantly, going down from 15% to 8%.

A Technical Specifications

A.1 Hardware Requirements

A.1.1 Project 1: Audio to Audio Unit Translation

- **GPU**: NVIDIA A6000 48GB VRAM
- **CPU**: Intel Xeon
- **RAM**: 64GB DDR4

A.1.2 Project 2: Scene Graph to Video Generation

- **GPU**: NVIDIA A6000 48GB VRAM
- **CPU**: Intel Xeon
- **RAM**: 64GB DDR4

A.1.3 Project 3: Water Quality Prediction

- **Embedded Device**: STM32 MCU
- **Connectivity**: LTE Cat-1 module
- **Sensors**: pH, Conductivity, Temperature, Dissolved Oxygen (DO) sensors

A.2 Software Dependencies

A.2.1 Project 1: Audio to Audio Unit Translation

- **Framework**: PyTorch 1.12+, Fairseq, speech-resynthesis
- **Python Version**: 3.8+
- **Key Libraries**:
  - transformers
  - librosa
  - soundfile
  - numpy, scipy
  - matplotlib
- **CUDA**: 11.3 or higher

A.2.2 Project 2: Scene Graph to Video Generation

- **Framework**: PyTorch 1.11+
- **Python Version**: 3.8+
- **Key Libraries**:
  - pytorch
  - opencv-python
- **CUDA**: 11.6 or higher

A.2.3 Project 3: Water Quality Prediction

- **Embedded Firmware**: FreeRTOS
- **Development IDE**: STM32CubeIDE, VSCode
- **ML Framework**: scikit-learn, pandas, numpy, pytorch
- **AutoML**: NNI
- **Communication**: MQTT, UART protocols

A.3 Datasets Used

A.3.1 Project 1: Audio to Audio Unit Translation

- **Primary Dataset**:
  - AIhub Multilingual Dataset: Korean-English Speech Translation Dataset
  - LibriSpeech: English Speech Recognition Corpus
- **Size**: Each language has 1,000 hours of paired Korean-English speech
- **Format**: 16kHz WAV files with transcriptions

A.3.2 Project 2: Scene Graph to Video Generation

- **Primary Dataset**: Action Genome Dataset
- **Size**: 10K video clips with scene graph annotations
- **Resolution**: Various (480p)
- **Duration**: 3-30 seconds per clip

A.3.3 Project 3: Water Quality Prediction

- **Data Source**: Company's own internal IoT data
- **Parameters**: pH, Conductivity, Temperature, Dissolved Oxygen (DO), pH
- **Predict**: TOC, TN, TP
- **Frequency**: 5 minutes per sample
- **Duration**: 6 months of continuous monitoring

B Code Repository

B.1 Project 1: Audio Translation

- **Main Repository**: https://github.com/hytric/Ko_utut
- **Structure**:
  - `speech2unit/`: HuBERT-based audio encoding
  - `unit2unit/`: Transformer-based translation
  - `unit2speech/`: Vocoder for speech synthesis
- **Paper Reviews**:
  - HuBERT: https://hytric.github.io/paperreview/HuBERT/
  - UTUT: https://hytric.github.io/paperreview/UTUT/
  - AV2AV: https://hytric.github.io/paperreview/AV2AV/
  - HiFi-GAN: https://hytric.github.io/paperreview/HiFi-GAN/

C Contact Information

- **Author**: Jongha Kim
- **Institution**: Department of Electronic Engineering, Inha University
- **Email**: kimjongha674@gmail.com
- **GitHub**: https://github.com/hytric
- **Personal Blog**: https://hytric.github.io/
- **LinkedIn**: https://www.linkedin.com/in/

REFERENCES

Jeongsoo Choi, Se Jin Park, Minsu Kim, and Yong Man Ro. Av2av: Direct audio-visual speech to audio-visual speech translation with unified audio-visual speech representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27315–27327, 2024.

Prafulla Dhariwal and Alex Nichol. Diffusion-based scene graph to image generation with masked contrastive pre-training. *arXiv preprint arXiv:2201.00308*, 2022.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *arXiv preprint arXiv:2106.07447*, 2021.

Minsu Kim, Jeongsoo Choi, Dahun Kim, and Yong Man Ro. Textless unit-to-unit training for many-to-many multilingual speech-to-speech translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:3934–3946, 2023.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *arXiv preprint arXiv:2010.05646*, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Ann Lee, Hongyu Yang, Wei-Yen Ma, and Cheng Chen. Utut: Unsupervised task-aware unit-to-unit translation. *arXiv preprint arXiv:2308.01831*, 2023.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. *arXiv preprint arXiv:1906.03327*, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *International Semantic Web Conference*, pp. 202–218. Springer, 2018.

Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *arXiv preprint arXiv:1612.03975*, 2017.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Danning Yang, Shijing Zou, Yunze Wang, Chunhua Shen, Kai Yu, and Manmohan Chandraker. Sgdiff: A style guided diffusion model for fashion synthesis. *arXiv preprint arXiv:2211.11138*, 2022.