BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONICS ENGINEERING

ROBUST KEYWORD SPOTTING IN NOISY ENVIRONMENTS BASED ON DEEP LEARNING

BY

FATİH MERCAN

MASTER OF SCIENCE THESIS

ANKARA - 2025

BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONICS ENGINEERING

ROBUST KEYWORD SPOTTING IN NOISY ENVIRONMENTS BASED ON DEEP LEARNING

BY

FATİH MERCAN

MASTER OF SCIENCE THESIS

ADVISOR

PROF. DR. HAMİT ERDEM

ANKARA – 2025

BAŞKENT UNIVERSITY

INSTITUTE OF SCIENCE AND ENGINEERING

This study, which was prepared by Fatih MERCAN, for the program of Master of Science in Electrical and Electronics Engineering has been approved in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Electrical and Electronics Engineering Department Department by the following committee.

Date of Thesis Defense: 02 / 05 / 2025

Thesis Title: Robust Keyword Spotting In Noisy Environments Based On Deep Learning

Examining Committee Members	**Signature**
Assoc. Prof. Dr. Derya YILMAZ, Gazi University	……..
Prof. Dr. Hamit ERDEM, Başkent University	……..
Assoc. Prof. Dr. Selda GÜNEY, Başkent University	……..

APPROVAL

Prof. Dr. Dilek ÇÖKELİLER SERDAROĞLU

Director, Institute of Science and Engineering

Date: … / … / ……..

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLER ENSTİTÜSÜ

YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: … / … / 20…

Öğrencinin Adı, Soyadı : Fatih MERCAN

Öğrencinin Numarası :

Anabilim Dalı : Elektrik-Elektronik Mühendisliği Ana Bilim Dalı

Programı : Elektrik-Elektronik Mühendisliği Tezli Yüksek Lisans

Programı

Danışmanın Unvanı/Adı, Soyadı : Prof. Dr. Hamit ERDEM

Tez Başlığı : Robust Keyword Spotting In Noisy Environments Based On Deep Learning

Yukarıda başlığı belirtilen Yüksek Lisans/Doktora tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 52 sayfalık kısmına ilişkin, 16 / 06 / 2025 tarihinde şahsım/tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 10'dur. Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:…….

ONAY

Tarih: … / … / 20…

Öğrenci Danışmanı:Prof. Dr. Hamit ERDEM

……..

This work is dedicated to my Mother Mukaddes KOPARICI, my sisters Mervenur and Betül MERCAN and my dear friend who I Dennis barking who couldn't see my wizard rank go up.

Fatih Mercan Ankara-2025

ACKNOWLEDGEMENTS

First and foremost, I am deeply grateful to my thesis advisor, Prof. Dr. Hamit ERDEM, whose expert guidance and endless patience have been indispensable in bringing this work to fruition. I also extend my thanks to my thesis jury for their invaluable insights and constructive feedback.

My heartfelt gratitude goes to my mother, Mukaddes KOPARICI, for her unwavering dedication and support—countless nights she watched over me as I toiled, ensuring I did not neglect my health and doing everything in her power to help me succeed.

I would like to thank my sisters, Mervenur and Betül MERCAN, for simply being there and encouraging me to keep going. To Paul Gocht, whose unrelenting friendship helped me gather myself over these past months—without you, I am not sure I would have made it through.

I am also indebted to Avery Ptettitt, who enabled me to embark on this program, and to Raina Elizabeth Samuel, for her visit and for often acting as my first reader. Marine Grossman deserves special thanks: without her constant support through my first semester and beyond, I do not know where I would be today.

Finally, I dedicate this work to my dear friend Dennis Barking, who was not here to see its completion but whose steadfast belief in my pursuit of knowledge never wavered. I have now attained the rank of Archwizard—Dennis, I hope you are watching over me.

Truly, thank you all.

ÖZET

Fatih MERCAN

DERIN ÖĞRENMEYE DAYALI GÜRÜLTÜLÜ ORTAMLARDA SAĞLAM ANAHTAR KELİME TESPİTİ

Başkent Üniversitesi

Fen Bilimleri Enstitüsü

Elektrik ve Elektronik Mühendisliği Anabilim Dalı

2025

Bu tez çalışmasında, olumsuz gürültü koşullarında anahtar kelime algılama (KWS) performansını artırmak amacıyla, tamamlayıcı akustik bilgilerin dönüştürücü tabanlı bir meta-sınıflandırıcı çerçevesi aracılığıyla bütünleştirildiği özgün bir sistem önerilmektedir. Bu doğrultuda, Berg ve diğerleri tarafından geliştirilen Keyword Transformer ailesinin bir türevi olan KWT-1 modeli bilgi damıtımı (knowledge distillation) uygulanmaksızın yeniden tasarlanmış ve Google Speech Commands v2 veri kümesi üzerinde 12 sınıflı bir sınıflandırma görevi için eğitilmiştir. Modelin dayanıklılığını artırmak amacıyla, ilgili literatürde önerilen kapsamlı veri artırma stratejileri uygulanmıştır. Sisteme tamamlayıcı akustik özellikler kazandırmak amacıyla iki ek modül geliştirilmiştir: bir gürültü türü sınıflandırıcısı ve bir sinyal-gürültü oranı (SNR) tahmin modeli. Gürültü türü sınıflandırıcısı, Abdoli ve diğerlerinin yöntemi temel alınarak tasarlanmış tek boyutlu bir evrişimli sinir ağı mimarisi kullanılarak geliştirilmiş ve UrbanSound8K veri kümesi üzerinde on farklı çevresel gürültü sınıfını tanıyacak şekilde eğitilmiştir. SNR tahmin modeli ise, kum saati (hourglass) tarzında özgün bir evrişimsel sinir ağı mimarisi benimseyerek sürekli SNR tahmini gerçekleştirmektedir. Bu modelin eğitimi sırasında, Google Speech Commands v2 veri kümesinden elde edilen temiz konuşma örnekleri, UrbanSound8K veri kümesinden seçilen gürültü sinyalleriyle 0 ila 20 dB arasında rastgele belirlenen SNR seviyelerinde karıştırılarak gerçekçi akustik ortamlar simüle edilmiştir. Anahtar kelime tahmini, gürültü türü sınıflandırması ve SNR tahmini olmak üzere üç farklı modülden elde edilen çıktılar, karar düzeyinde dönüştürücü tabanlı bir meta-sınıflandırıcı kullanılarak bütünleştirilmiştir. Bu yapı içerisinde her bir model çıktısı ayrı bir token olarak ele alınmış, ortak bir gömme (embedding) uzayına projekte edilmiş ve bir dönüştürücü kodlayıcı (transformer encoder) bloğu aracılığıyla işlenmiştir. Bu tasarım, anlamsal, çevresel ve akustik faktörler arasındaki

karmaşık ilişkileri etkin bir şekilde modellemeyi amaçlamaktadır. Önerilen birleştirme modeli, anahtar kelime sınıflandırma doğruluğu bakımından temel KWT-1 modelinin performansını aşamamakla birlikte, kum saati tarzında tasarlanan SNR tahmin ağı sayesinde mevcut sinir ağı tabanlı yaklaşımlara kıyasla daha başarılı sonuçlar elde edilmiştir. Modelin performansı, anahtar kelime ve gürültü türü tespiti için sınıflandırma doğruluğu; SNR tahmini için ise ortalama mutlak hata (MAE) metriği kullanılarak değerlendirilmiştir.

ANAHTAR KELİMELER: Anahtar Kelime Tespiti, Dönüştürücü Tabanlı Anahtar Kelime Algılama Modeli, Dönüştürücü Tabanlı Meta-Sınıflandırıcı, Gürültü Türü Sınıflandırması, Sinyal-Gürültü Oranı Tahmini

ABSTRACT

Fatih MERCAN

ROBUST KEYWORD SPOTTING IN NOISY ENVIRONMENTS BASED ON DEEP LEARNING

Başkent University

Institute of Science

Department of Electrical and Electronics Engineering

2025

The present thesis introduces a novel keyword spotting (KWS) system aimed at enhancing performance under adverse noisy conditions by integrating supplementary acoustic information through a transformer-based meta-classifier framework. To accomplish this, KWT-1—a variant of the Keyword Transformer family introduced by Berg et al.—is reimplemented as the base KWS component. This model is applied without the use of knowledge distillation and is trained on the Google Speech Commands v2 dataset for a 12-label classification task. Extensive data augmentation strategies are employed in alignment with the original study to ensure robust model performance. To extract complementary acoustic features, two additional modules are integrated: a noise type classifier and a signal-to-noise ratio (SNR) prediction model. The noise classifier is implemented as a one-dimensional convolutional neural network informed by the methodology of Abdoli et al. and trained on the UrbanSound8K dataset to recognize ten distinct environmental noise classes. The SNR prediction model adopts a novel hourglass-style convolutional architecture to perform continuous SNR regression. During its training, clean speech samples from the Google Speech Commands v2 dataset are mixed with noise from UrbanSound8K at random SNR levels ranging from 0 to 20 dB, simulating realistic acoustic environments. The outputs from the three branches—keyword prediction, noise type, and estimated SNR—are fused at the decision level using a transformer-based meta-classifier. In this configuration, each model output is treated as a discrete token, projected into a shared embedding space, and processed by a transformer encoder block. This design is intended to capture the complex interdependencies between semantic, environmental, and acoustic factors. Although the proposed fusion model did not surpass the performance of the standalone KWT-1 baseline in terms of keyword classification accuracy, the work contributes to the academic literature

by introducing an hourglass-style CNN for SNR level estimation that outperforms existing neural network-based approaches. Evaluation is conducted using classification accuracy for keyword and noise type detection tasks and mean absolute error (MAE) for the SNR regression task.

KEYWORDS: Keyword Detection, Converter Based Keyword Detection Model, Converter Based Meta-Classifier, Noise Type Classification, Signal-to-Noise Ratio Estimation

PREFACE

If you've picked up this thesis and wondered what the author endured in writing it, you wouldn't believe half the stories I could tell. I faced heartbreak and loss the likes of which I'd never before experienced—and never hope to again. If you, too, are wrestling with personal challenges while tackling your own work, know this: it is often impossible to sit down and write. So pause, breathe, and remember that perfection is not the goal—completion is.

I am proud to have finished this work despite everything I went through. There were countless moments when I wanted to quit and times I believed I could not carry on. But each time, my friends and family lifted me back to my feet. I hope your support system is as strong as mine, and that you, too, find the strength to reach the finish line.

TABLE OF CONTENTS

LIST OF TABLES

Page

LİST OF FİGURES

Page

LIST OF SYMBOLS AND ABBREVIATIONS

A	Amplitude scaling factor in Gammatone filter impulse response
AGC	Automatic Gain Control
ASR	Automatic Speech Recognition
CNN	Convolutional Neural Network
Conv1D	One-dimensional convolution layer
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
ERB	Equivalent Rectangular Bandwidth (scale for Gammatone filters)
FFN	Feed-Forward Network sublayer in Transformer
Gammatone filter	Biologically inspired cochlear filter bank
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit (training hardware)
GSCv2	Google Speech Commands v2 dataset
GRU	Gated Recurrent Unit
HG CNN	Hourglass-style convolutional neural network for SNR estimation
HMM	Hidden Markov Model
KWT	Keyword Transformer family
KWT-1	Smallest KWT variant (12 layers, 1 head, 64-dim)
k	Order of Gammatone filter
α	Label-smoothing factor
LVCSR	Large Vocabulary Continuous Speech Recognition
LSTM	Long Short-Term Memory network
MAE	Mean Absolute Error
F	Number of Mel-frequency filter bank channels (40)
MFCC	Mel-Frequency Cepstral Coefficients
MLP	Multi-Layer Perceptron
MMSE	Minimum-Mean-Square-Error estimator
MSLE	Mean Squared Logarithmic Error loss
MSE	Mean Squared Error
NLP	Natural Language Processing
PCEN	Per-Channel Energy Normalization
p	Patch size
PCEN	Per-Channel Energy Normalization
Q, K, V	Query, Key, Value matrices in self-attention
RMS	Root Mean Square
RNN	Recurrent Neural Network
SNR	Signal-to-Noise Ratio
SpecAugment	Spectrogram augmentation (time/frequency masking)
STFT	Short-Time Fourier Transform
TPU	Tensor Processing Unit

VAD	Voice Activity Detection
ViT	Vision Transformer
VSR	Visual Speech Recognition
WER	Word Error Rate
b	Bias vector in linear projections
d	Embedding dimension (64)
$\mathbb{1}_{mask}$	Binary mask indicator in SpecAugment
h	Number of attention heads (e.g. 4 in meta-classifier)
K	Total number of classes (12) in label smoothing
M	MFCC feature matrix
N_{mask}	Number of time/frequency masks in SpecAugment
W	Weight matrix in linear projections
x_i	Input token i (class token or patch)
y_i	True class index for label smoothing
π	Mathematical constant pi (≈ 3.14159), used in the cosine-decay learning-rate schedule
ϵ	Small constant for numerical stability in Discrete Cosine Transform

1. INTRODUCTION

Speaking to a machine and having it perform a task was once the realm of science fiction. However, with the paradigm shift in human-machine interactions, this concept has become a reality. Today, many end users are familiar with and actively use voice assistants such as Apple's Siri, Microsoft's Cortana, and Google's Assistant, engaging with their personal devices effortlessly [1]. These voice assistants offer an "always listening" feature, increasing the demand for a less computationally expensive alternative to Automatic Speech Recognition (ASR), a technology that continuously converts spoken language into text using machine learning and signal processing techniques[2]. Since these assistants rely on a wake word or phrase—such as "Ok Google," "Siri," or "Hey Cortana"—keyword spotting (KWS) has become the standard method for wake-word detection [1]

Beyond voice assistants, KWS is widely used in various Natural Language Processing (NLP) tasks, including audio indexing, call monitoring, speech data mining, and voice command recognition[3].

Since KWS systems are often designed for embedded deployment in small devices such as smart speakers, wearables, and home automation systems, research has largely focused on improving computational efficiency. Over the past three decades, numerous approaches have been proposed. Early KWS approaches relied on Large Vocabulary Continuous Speech Recognition (LVCSR) systems, which decoded full speech sequences before searching for specific keywords. While this method allowed flexible keyword detection, it was computationally expensive and introduced latency since it required a full transcript of the speech before searching making it impractical to use for real-world scenarios[2]. To combat these issues, Hidden Markov Models (HMMs) have been proposed [4]. These models used Gaussian Mixture Models (GMMs) to represent acoustic features, and the keywords were specifically modeled while all other speech was categorized as filler. While this approach has lowered the latency issues with the LVCSR approach it still depended on computationally demanding Viterbi decoding.[2]

Today, with the advancements in the field, the paradigm for KWS has shifted towards machine learning techniques, particularly deep neural networks (DNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid tree neural networks[5]. These approaches allow for easier customization of posterior handling based on the available computational power [2]. They also provide improvements in accuracy [5]

Keyword Spotting (KWS) can be performed using different modalities, including audio speech recognition, video recognition, or an audio-visual hybrid approach [2]. While most research focuses on audio-based KWS, incorporating visual cues—when the application scenario permits—can enhance robustness, particularly in noisy environments, since visual signals remain unaffected by acoustic noise. A particular challenge in Visual Speech Recognition (VSR) is the presence of homophenes, where different sounds can appear identical on a person's lips (e.g., "pat," "mat," and "bat") [6]. This limitation makes Audio-Visual Speech Recognition (AVSR) valuable, as its bimodal approach helps overcome the shortcomings of unimodal methods.

The pipeline for a modern KWS system starts with speech data acquisition, followed by preprocessing, usually in the form of feature extraction, a machine-learning-based acoustic model, posterior handling, and decision-making [2]. The feature extraction used for modeling acoustic features of speech data most commonly uses Mel-scale related features such as Mel-frequency cepstral coefficients(MFCCs) and log-Mel spectral coefficients, and a widely standard practice to normalize either one of these Mel-scale related features[2].

After feature extraction, the acoustic model is responsible for keyword detection. Modern KWS approaches leverage deep learning techniques, enabling more accurate and efficient keyword detection. CNNs have been widely adopted due to their ability to capture spatial patterns in spectrograms, while RNNs, such as long short-term memory (LSTM) networks and gated recurrent units (GRUs), process sequential dependencies in speech. Hybrid models that combine CNNs for feature extraction and RNNs for temporal modelling have demonstrated improved performance over traditional methods[7]. These models aren't without their shortcomings. CNNs rely on local receptive fields, which may not fully capture long-range dependencies in speech sequences, while RNNs suffer from sequential processing bottlenecks, making them computationally expensive for real-time applications. To overcome these challenges, transformer-based architectures have been introduced in KWS systems.

Transformers, initially designed for natural language processing, have gained attention in speech processing due to their ability to capture global contextual clues through self-attention mechanisms. Unlike RNNs, transformers process entire sequences of data in parallel, which enables significant computational efficiency. One such transformer-based model is the Keyword Transformer (KWT), which leverages multi-head self-attention to learn relationships between different time frames in a speech signal. The KWT model family (KWT-1, KWT-2, KWT-3) provides different configurations optimized for various

computational constraints, offering improved performance over CNN/RNN-based models in terms of both accuracy and efficiency[2] [5]

Since the applications of the KWS in the real world often face the challenge of noisy environments, the research often tests the performance of the proposed approaches under different signal-to-noise ratios. Background noise and reverberations are detrimental to the KWS performance, increasing false rejections and acceptances. Under low SNR conditions, the speech components of audio data can become indistinguishable from the noise; to address this issue, research proposed frontend methods such as automatic gain control (AGC)[8] and later a derivative of AGC per-channel energy normalization (PCEN), and back-end methods such as multi-style training and adversarial training. DNN feature enhancements have also been studied in the forms of Enhancement mask estimation, Noise estimation, clean speech estimation, and Filter parameter estimation[2]

In this thesis, a novel keyword spotting (KWS) system is proposed to improve performance under challenging noisy conditions by integrating supplementary acoustic information through a transformer-based meta-classifier. The meta-classifier employs a late decision-level fusion strategy, combining the outputs of three independently trained models: a keyword classification model, a noise type classifier, and a signal-to-noise ratio (SNR) regression model. Specifically, the keyword spotting component is based on KWT-1, a variant of the Keyword Transformer family, trained on the Google Speech Commands v2 dataset. The environmental noise classifier is implemented as a convolutional neural network trained to distinguish between ten noise categories using the UrbanSound8K dataset. The SNR regression model, built using an hourglass-style convolutional architecture, is trained on synthetic mixtures of speech and noise to estimate the SNR of each utterance.

These three outputs—keyword prediction, noise type, and SNR estimation—are encoded and fused within a transformer-based meta-classifier, which learns to capture interdependencies across modalities to improve final keyword predictions. Although the overall system did not outperform the baseline KWT-1 in terms of keyword classification accuracy, it contributes to the literature by introducing a novel SNR estimation model that demonstrates superior performance compared to existing neural network-based approaches.

2. METHODS

2.1. Datasets

2.1.1. Google speech commands V2

This work relies on two complementary datasets: the Google Speech Commands v2 corpus for spoken keywords and UrbanSound8K for environmental noise. The Google Speech Commands v2 (GSCv2) dataset is a widely used benchmark for keyword spotting, containing about _105,000_ one-second utterances of 35 distinct short words (commands) recorded by thousands of speakers[9]. These include a core set of 10 common commands (e.g., “Yes”, “No”, “Up”, “Down”, “Left”, “Right”, “On”, “Off”, “Stop”, “Go”), digits zero through nine, and several other short words to provide a diverse phonetic coverage[9]. All audio clips are sampled at 16 kHz and trimmed to one second. The purpose of GSCv2 is to facilitate the training of lightweight speech recognition models on a limited vocabulary[10]. It provides a standard evaluation for keyword detection systems in clean conditions, which we leverage as the base for our speech input. We will use this dataset to train and evaluate the keyword recognition component of our system.

2.1.2. UrbanSound8K

The UrbanSound8K dataset complements the speech data by providing a rich collection of tagged environmental noises. UrbanSound8K contains 8,732 labeled sound excerpts of urban environmental sounds, each up to 4 seconds long[11]. These clips are drawn from 10 noise categories common in city environments: _air conditioner hum, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren,_ and _street music_[11]. The audio excerpts were obtained from real field recordings (via Freesound[12]) and thus exhibit authentic background noise characteristics[11]. They vary in sampling rate and format (often originally 44.1 kHz [11], so for our purposes, we resample and convert them to a unified format (e.g. 16 kHz mono) during preprocessing. UrbanSound8K is an environmental noise modeling resource: its categories encompass a range of impulsive noises (e.g., gunshots, car horns), continuous mechanical noise (air conditioners, engines), human and music noises, etc., which makes it highly relevant for training models to recognize and characterize background noise. In this thesis, we use

UrbanSound8K both to simulate noisy conditions for speech commands and to train a dedicated noise classifier. By pairing the Speech Commands with concurrent UrbanSound noise samples, we can create a realistic dataset of spoken keywords embedded in various ambient noises. The UrbanSound8K classes serve as labels for different noise environments, allowing a model to learn distinctions like “traffic noise” vs. “children playing,” which provides context to the speech model about the type of interference present.

2.2. Keyword Detection Model

2.2.1 Background and motivation

Keyword spotting (KWS) refers to the task of detecting pre-defined spoken commands in continuous audio streams and is a critical component in a wide array of low-power and real-time applications such as voice-controlled assistants, smart home systems, and embedded devices. These systems often operate under stringent computational constraints and are expected to remain constantly active, which necessitates models that achieve a balance between efficiency and robustness[5].

Historically, early keyword spotting systems relied on Large Vocabulary Continuous Speech Recognition (LVCSR) pipelines, which transcribed the entire utterance before matching it against keywords. Although accurate, these systems were computationally expensive and incurred substantial latency. In response, Hidden Markov Model (HMM)-based systems, often using Gaussian Mixture Models (GMMs) to represent acoustic distributions, were introduced as more efficient alternatives. However, these systems exhibited limited generalization, particularly in noisy or spontaneous speech conditions [5].

The advent of deep learning ushered in a new generation of KWS architectures, with convolutional neural networks (CNNs) and recurrent neural networks (RNNs) emerging as dominant paradigms. CNNs excel at capturing local Spectro-temporal patterns, while RNNs—such as LSTMs and GRUs—are adept at modeling long-term dependencies through recurrence. Yet, both architectures present fundamental limitations. CNNs are constrained by their localized receptive fields, impeding their ability to model long-range dependencies, while RNNs are inherently sequential, limiting parallelizability and increasing inference time [13].

To address these limitations, Vaswani et al. (2017) introduced the Transformer architecture, which employs a self-attention mechanism to capture global dependencies

without the need for recurrence or convolution. The core operation is scaled dot-product attention, mathematically formulated as shown in Equation 2.1:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where Q, K, and V denote the query, key, and value matrices consecutively, and d_k is the dimensionality of the key vectors. This formulation enables the model to compute contextual relationships between all input positions simultaneously, offering superior modeling of long-range dependencies with high parallel efficiency[13]

Self-attention allows each element in a sequence to attend to all other elements, facilitating the modeling of temporal or spatial dependencies in a fully dynamic and content-based manner. Unlike convolution or recurrence—which impose fixed receptive fields or sequential dependencies—attention mechanisms compute adaptive weightings between all positions in parallel, enabling the model to prioritize context based on learned relevance rather than structural proximity.

The Transformer architecture is composed of stacked encoder (and optionally decoder) layers, each containing a multi-head self-attention mechanism followed by a position-wise feed-forward network. These are equipped with residual connections and layer normalization, and the entire model is trained end-to-end. The design eliminates the need for recurrence or convolution, allowing for fully parallelized training and inference, which significantly accelerates computation and enables greater scalability across diverse sequence modelling tasks.

Building on the Transformer's success in NLP, Dosovitskiy et al. proposed the Vision Transformer (ViT), which demonstrated that transformers could outperform convolutional networks on image classification tasks by directly applying self-attention over flattened image patches with minimal architectural modification[14]. ViT's success showed that inductive biases such as locality and translation equivariance, which are implicit in CNNs, could be replaced by learned relationships across tokenized inputs—provided that sufficient training data is available.

Informed by the ViT framework, Berg et al. introduced the Keyword Transformer (KWT)—a fully self-attentional architecture designed specifically for keyword spotting tasks. In KWT, audio signals are transformed into Mel-frequency cepstral coefficients (MFCCs) and divided into patches along the temporal axis. These patches, representing individual or grouped MFCC frames, are flattened and projected into an embedding space, where a learnable class token is prepended. After the addition of positional encodings, the

sequence is passed through a series of Transformer encoder blocks. The output corresponding to the class token is used for classification [5].

KWT offers two major advantages over CNN/RNN-based KWS systems. First, it enables global modeling of temporal context, allowing the network to dynamically weigh the importance of different time steps. Second, it supports fully parallel computation at both training and inference time, enabling faster execution and reduced latency—properties essential for deployment in real-time environments.

In this thesis, the KWT-1 configuration, the smallest and most efficient variant proposed by Berg et al., is selected as the base model. Its simplicity, compact parameter count, and proven effectiveness on the Google Speech Commands dataset make it an ideal foundation for further extension. In the chapters that follow, this model is augmented with two auxiliary components—a signal-to-noise ratio (SNR) prediction module and a noise-type classification module—with the goal of improving keyword spotting accuracy under adverse acoustic conditions.

2.2.2 Model overview

The architecture implemented in this study is based on the KWT-1 configuration from the Keyword Transformer model family proposed by Berg et al., representing the smallest and most efficient variant within the series. The KWT-1 model preserves the essential characteristics of a Transformer-based encoder while maintaining a minimal parameter footprint suitable for low-latency applications. It contains 12 Transformer encoder layers, each with a single attention head, and operates with a model dimension of 64, yielding a total parameter count of approximately 607K—substantially lower than typical CNN-RNN hybrid models used in keyword spotting [5].

2.2.2.1. Input representation and preprocessing

The input to the model consists of raw speech signals sampled at 16 kHz and truncated or zero-padded to a fixed length of one second. Each waveform is transformed into Mel-frequency cepstral coefficients (MFCCs) using the following pipeline: a short-time Fourier transform (STFT) is computed with a window size of 30 ms and a hop length of 10 ms; the resulting spectrum is passed through a Mel-scale filter bank, followed by logarithmic compression and a discrete cosine transform (DCT). This results in a 2D MFCC feature map with 40 coefficients per frame and 98 frames per second. The MFCC representation is defined as Equation 2.2:

$$\mathrm{MFCC}(x) = \mathrm{DCT}(\log(M \cdot |\mathrm{STFT}(x)|^2 + \epsilon)) \quad (2.2)$$

where M is the Mel filter bank matrix and ϵ is a small constant for numerical stability.

Unlike the original KWT-1, which used a patch size of 10 frames, the implementation in this study adopts a patch size of 1, treating each individual MFCC frame as a separate token. Each 40×1 patch is flattened into a 40-dimensional vector and linearly projected into a 64-dimensional embedding space, resulting in 98 input tokens per example.

2.2.2.2 Class token and positional embedding

Following ViT [14], we first project each flattened MFCC patch into a D-dimensional embedding and prepend a learnable class token $c \in R^D$. If we have N patch embeddings $z_1, \ldots, z_N$, we form the extended sequence represented as Equation 2.3:

$$Z = [c; z_1; \ldots; z_N] \in R^{(N+1)\times D} \quad (2.3)$$

Simultaneously, we learn a positional-embedding matrix represented as Equation 2.4:

$$P = [p_0; p_1; \ldots; p_N] \in R^{(N+1)\times D} \quad (2.4)$$

where p_i encodes the position of the ith token (with p_0 for the class token). We then add these two to obtain the Transformer input calculated as shown in the Equation 2.5:

$$X = Z + P \quad (2.5)$$

2.2.2.3. Transformer Encoder Stack

The resulting sequence of 99 tokens (1 class token + 98 patch tokens) is passed through a stack of 12 Transformer encoder layers. Each layer consists of two sublayers as shown in Equation 2.7 and 2.8:

a) Multi-head self-attention (MHSA):

$$\mathrm{MHSA}(X) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)W^O \quad (2.6)$$

where each head computes as shown in Equation 2.7:

$$\mathrm{head}_i = \mathrm{Attention}\left(XW_i^Q, XW_i^K, XW_i^V\right) \quad (2.7)$$

b) Feed-forward network (FFN):

$$\mathrm{FFN}(x) = W_2 \cdot \mathrm{GELU}(W_1 x + b_1) + b_2 \quad (2.8)$$

Residual connections and post-normalization are applied after both the attention and feed-forward blocks, matching the configuration reported by [5]. As illustrated in Figure 2.1, each transformer encoder block employs a post-norm residual design in which each sub-

layer's output is normalized immediately after its residual addition. First, the input sequence of embedded MFCC patches is processed by a multi-head self-attention module; its output is added back to the original input via a residual connection and then passed through a Layer Normalization layer. The normalized features are then fed into a two-layer, position-wise feed-forward network (MLP) with a non-linear activation; the MLP output is again summed with its input through a residual link and followed by a second Layer Normalization. This post-norm configuration promotes stable gradient flow and faster convergence when stacking deep encoder blocks, while allowing the model to capture both long-range temporal dependencies and rich non-linear feature transformations. Figure 2.2 shows the entire model along with the input pipeline.

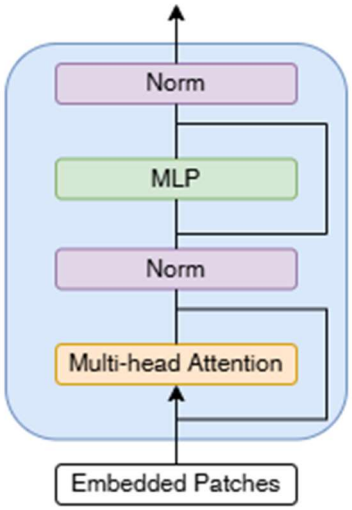

Figure 2.1. Post-norrm transformer

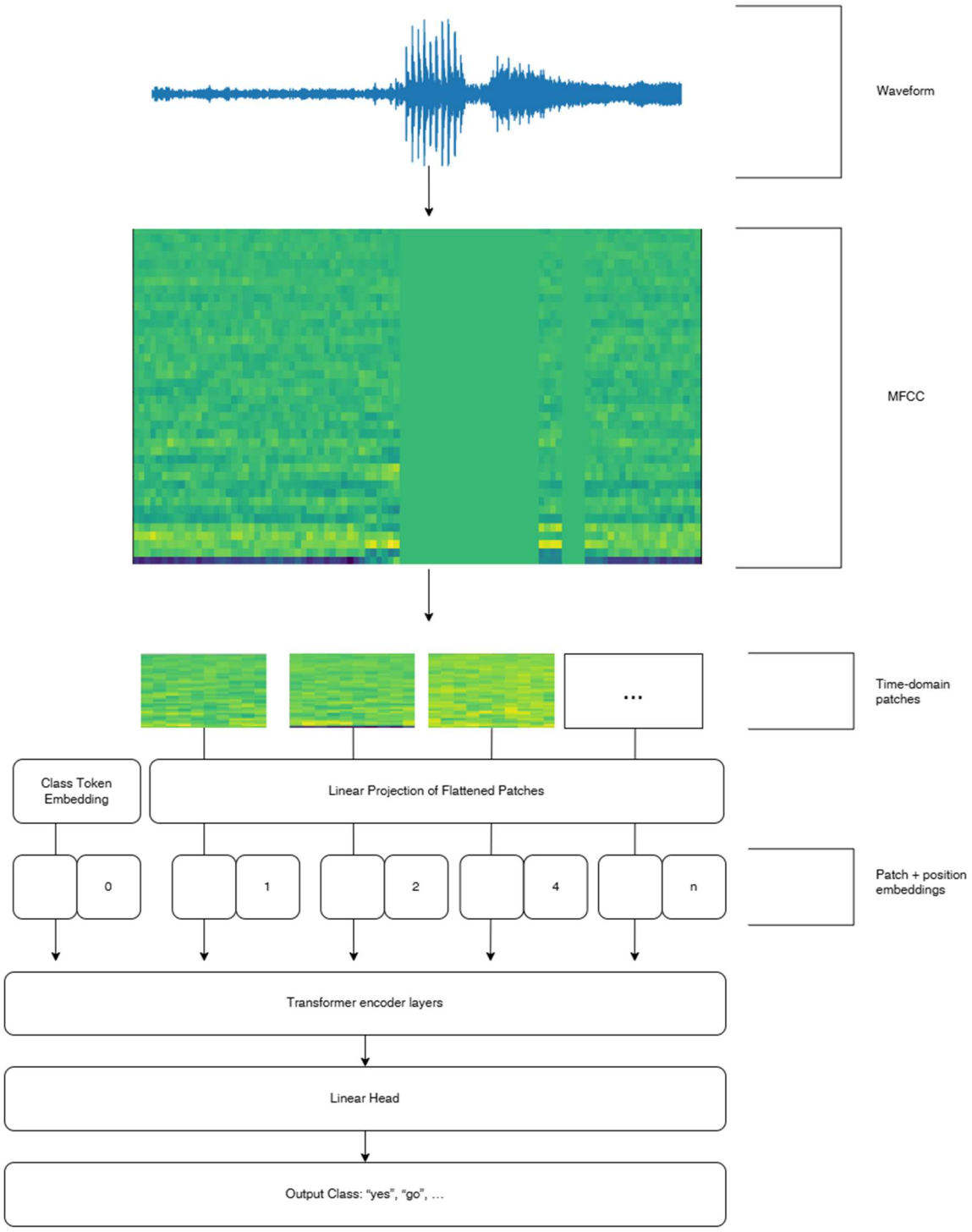

Figure 2.2. KWT-1 model architecture with input illustration

Table 2.1. KWT-1 Implementation Parameters

Parameter	**Value**
Layers (depth)	12
Model dimension	64
Feed-forward dimension	256
Attention heads	1
Patch size	1 (single MFCC frame)
Tokens per sequence	99 (1 class + 98 MFCC frames)

Note that although Berg et al. report a total of approximately 607,000 parameters for the KWT-1 configuration, our reimplementation contains 603,276 parameters, reflecting minor implementation and serialization differences.

2.2.2.4. Output and Classification

After the final encoder layer, only the class token is retained. This vector is passed through a dense layer with SoftMax activation to produce posterior probabilities over the 12 output classes (10 target keywords, `_silence_,` `_unknown_`). The classification decision is made solely based on the representation encoded in this token.

This finer temporal resolution improves the model's capacity to capture transient phonetic events across speech segments. While increasing the number of attention tokens introduces greater computational demand, the lightweight design of the KWT-1 architecture ensures that runtime efficiency remains within deployable bounds.

2.2.3 Implementation details and divergences

The implementation of the Keyword Transformer (KWT-1) in this study strictly follows the core architectural specifications provided by[5] while introducing targeted modifications motivated by the objectives of reproducibility and experimental clarity. This section outlines the key aspects of the implementation, including dataset handling, augmentation policy, training strategy, and deliberate divergences from the original setup.

2.2.3.1. Dataset configuration and label mapping

The model was trained on the Google Speech Commands dataset (version 2), comprising approximately 105,000 one-second utterances sampled at 16 kHz. The

classification task was constrained to 12 categories: ten target keywords (“yes,” “no,” “up,” “down,” “left,” “right,” “on,” “off,” “stop” and “go”) and two auxiliary classes: `_silence_` for silent segments and `_unknown_` for all other labels. Following Warden (2018) [9], the dataset was partitioned into training, validation, and test sets in an 80: 10 : 10 ratio. We used the provided `validation_list.txt` and `testing_list.txt` files to construct these splits: utterances listed in `validation_list.txt` and `testing_list.txt` were assigned to the validation and test sets, respectively, and all remaining utterances formed the training set. Although equivalent partitions could have been derived programmatically from file names, adherence to the pre-defined lists ensures consistency and reproducibility with established benchmarking protocols.

To construct the silence class, one-second segments were randomly extracted from the long recordings in the dataset’s _background noise_ folder and included as silent examples. The _unknown_ class comprised all utterances whose labels were not among the ten target keywords; these non-target examples were aggregated and randomly sampled to yield a class distribution comparable to the target keywords.

2.3.3.2. Spectrogram extraction and augmentation

Each raw waveform was processed into Mel-frequency cepstral coefficients (MFCCs) using a short-time Fourier transform (STFT) with a window size of 480 samples (30 ms) and a hop length of 160 samples (10 ms). The resulting MFCC matrices contain 40 coefficients across 98 time steps, consistent with the input format expected by the revised KWT-1 implementation.

To enhance robustness and promote generalization, several augmentation techniques were applied:

- Time shifting: Random shifts of up to ±100 ms were applied to the waveform.
- Speed perturbation: Signals were resampled with factors between 0.85× and 1.15×.
- Background mixing: Noise from the dataset’s `_background_noise_` folder was randomly added at low volume.
- SpecAugment: Time and frequency masking were applied to the spectrogram following the procedure described in the original paper.

These augmentations were implemented using a combination of NumPy, Librosa, and TensorFlow’s data pipeline, with performance-optimized use of `tf.py_function` and `AUTOTUNE`.

2.3.3.3. Model architecture and training setup

The architecture was implemented in TensorFlow 2.18 using the Keras Functional API. All custom layers—such as the class token insertion layer, position embedding layer, and the residual normalization blocks—were registered using the `@tf.keras.utils.register_keras_serializable` decorator to ensure compatibility with model saving and loading mechanisms.

The model was compiled with a label smoothing loss function calculated with the formula given in Equation 2.9:

$$L_{smoothed} = CE\left((1-\alpha)\cdot y + \frac{\alpha}{K}, \hat{y}\right) \tag{2.9}$$

where $\alpha = 0.1$ is the smoothing factor and $K = 12$ is the number of classes. This loss was implemented using a custom `sparse_ce_label_smooth` function.

Optimization was performed using the AdamW optimizer with the following hyperparameters:

- Base learning rate: 1×10^{-3}
- Weight decay: 0.1
- Warmup steps: ≈ 10 epochs
- Total training steps: 23,000
- Learning rate schedule: warmup followed by cosine decay

Cosasine decay is calculated as shown in equation 2.10:

$$\eta(t) = \begin{cases} \eta_0 \dfrac{t}{t_{warmup}} & , \text{if } t \leq t_{warmup} \\ \eta_0 \dfrac{1}{2}\left(1 + \cos(\pi \dfrac{t - t_{warmup}}{T - t_{warmup}}\right) & , \text{if } t > t_{warmup} \end{cases} \tag{2.10}$$

The training was executed on a TPU v3 back-end in Google Colab using the `tf.distribute.TPUStrategy`.

Deliberate Divergences from the Original KWT Implementation

While the implementation adheres closely to the original KWT-1 design, a key divergence was introduced: knowledge distillation was omitted. The original study proposed a distillation setup wherein the class token received attention not only from patch tokens but also from a distillation token trained under the supervision of a teacher model. In the present implementation, this mechanism was excluded to focus solely on the intrinsic performance

of the base model, as the downstream goal was to later enhance KWT-1 via auxiliary acoustic inputs rather than additional supervision.

In addition, no pretrained weights were used, and training was conducted from scratch using only the official Google Speech Commands dataset and its associated noise samples.

2.2.4. Preprocessing and augmentation pipeline

The effectiveness of a keyword spotting (KWS) model under real-world acoustic variability is closely tied to the diversity and quality of its input representation. This section details the preprocessing and data augmentation steps used to prepare the input to the Keyword Transformer (KWT-1) model, all of which are aligned with the methodological practices reported by Berg et al.[5]

2.2.4.1. Audio standardization and duration normalization

All input utterances were sampled at 16 kHz and standardized to a fixed length of 1 second (16,000 samples). Audio files shorter than 1 second were zero-padded; those longer were truncated. This step ensures consistent tensor shapes, a prerequisite for batch processing on TPUs.

As illustrated in Figure 2.3, the raw waveform exhibits the complete one-second utterance at a 16 kHz sampling rate.”

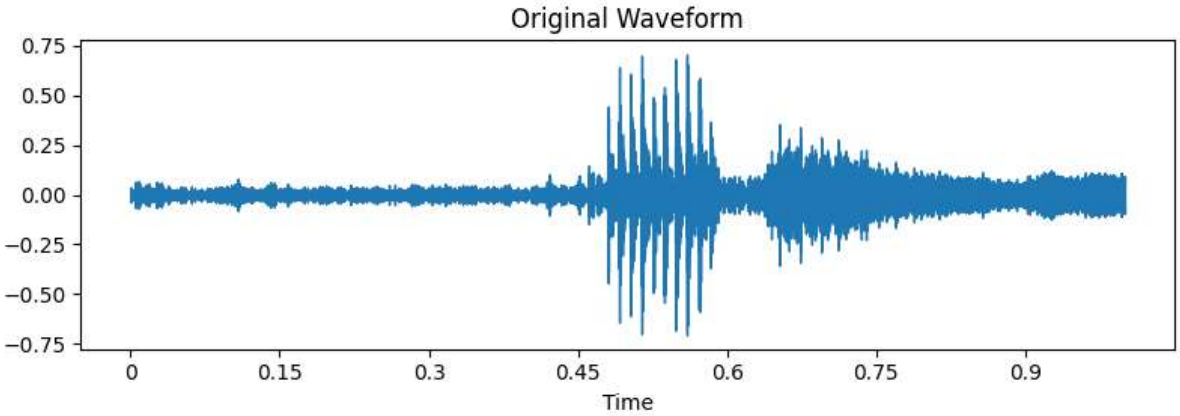

Figure 2.3. Example raw wave form

2.2.4.2. MFCC extraction

Each raw waveform was transformed into Mel-frequency cepstral coefficients (MFCCs) using a short-time Fourier transform (STFT) with a window size of 480 samples

(30 ms) and a hop length of 160 samples (10 ms). The resulting spectrogram was passed through a Mel-scale filter bank, followed by logarithmic compression and a discrete cosine transform (DCT), yielding a feature matrix of shape 40×98 (cepstral coefficients × time frames). This representation aligns with the input expected by the KWT-1 model configured for a patch size of 1 (i.e., per-frame tokenization). MFFC was calculated with the Equation 2.2. Figure 2.4 depicts the MFCC representation prior to any augmentation.

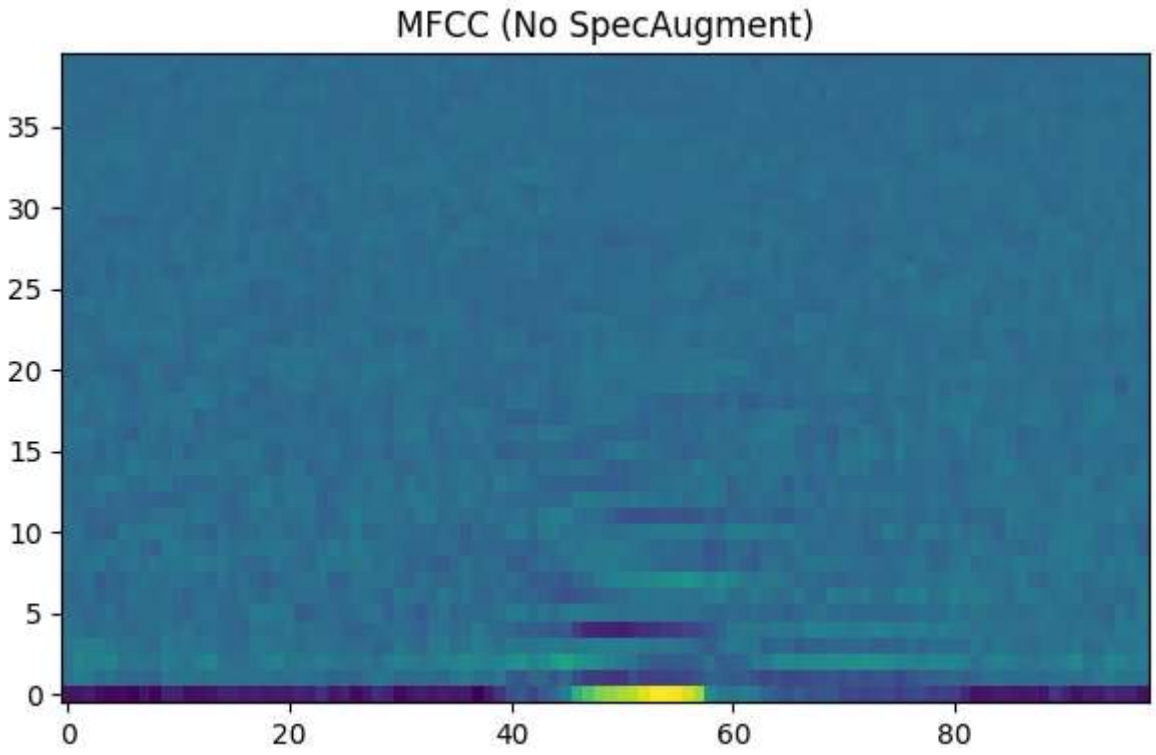

Figure 2.4 Visual MFCC representation

2.2.4.3. Waveform-Level Augmentation

To improve generalization and simulate real-world acoustic variability, several waveform-level augmentation techniques were applied during training:

1. Random Time Shifting:

The raw waveform $x(t)$ is shifted by a random integer $\tau \in [-\Delta, \Delta]$samples, where $\Delta = 1600$ corresponds to ±100 ms at 16 kHz. The time-shifted signal $x_{shift}(t)$ is given by the function shown in the Equation 2.12:

$$x_{shift}(t) = \begin{cases} x(t+\tau) & , \text{if } 0 \leq t+\tau < T \\ 0 & , \text{otherwise} \end{cases} \quad (2.12)$$

where T is the total number of samples (16,000).

As seen in Figure 2.5, the utterance is displaced by a randomly sampled shift within ±100 ms."

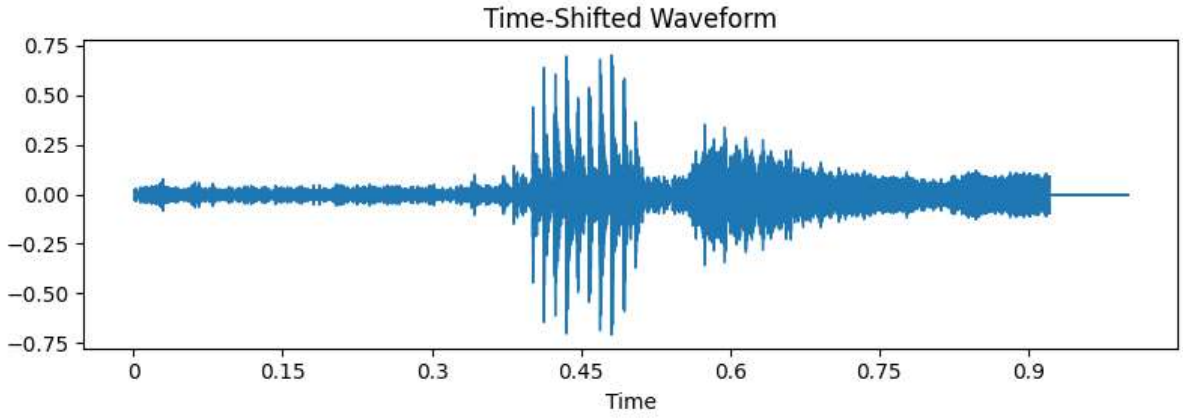

Figure 2.5. Wave form with time shifting applied

2. Speed Perturbation:

To simulate variations in speaking rate and pitch, the signal is resampled by a factor $r \in [0.85,1.15]$. The perturbed waveform $x_{spd}(t)$ is expressed via the function given in Equation 2.13:

$$x_{spd}(t) = x(rt) \tag{2.13}$$

where interpolation is used to compute non-integer indices. The signal is subsequently resampled or padded to maintain a consistent duration of 1 second.

3. Background Noise Injection:

Given a background noise sample $n(t)$ and a clean waveform $x(t)$, additive noise is applied as shown in the equation 2.14:

$$x_{noisy}(t) = x(t) + \lambda \cdot n(t) \tag{2.14}$$

where $\lambda \in [0.0,0.1]$ is a randomly sampled noise gain factor. If $n(t)$ is shorter than $x(t)$, it is zero-padded; if longer, a random segment of length T is selected.

Figure 2.6 shows the effect of mixing a background-noise segment into the original signal.

These augmentations were implemented using `librosa` and NumPy, and were integrated into the TensorFlow data pipeline via `tf.py_function` to maintain efficient execution within a TPU environment.

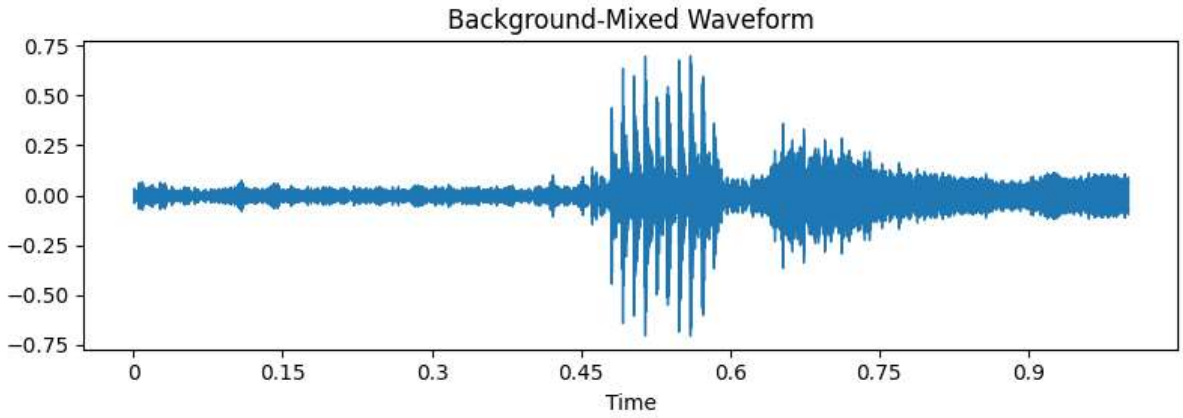

Figure 2.6. Wave form with background noise mixed

2.2.4.2. MFCC-Level augmentation: SpecAugment

SpecAugment[15]was applied directly to the MFCC matrices to simulate both temporal and spectral dropouts, encouraging the model to rely on global context. Let $M \in R^{F\times T}$ be the MFCC matrix for one example ($F = 40$ Mel bins, $T = 98$ frames). We performed:

1. Time Masking
   - Number of masks: $N_{time} = 2$
   - For each mask, sample calculate expression given as Equation 2.15:

$$w_t \sim \text{Uniform}(0, T_{max}), \quad T_{max} = 25, \qquad t_0 \sim \text{Uniform}(0, T - w_t) \tag{2.15}$$

then zero out the block shown in Equation 2.16:

$$M[:, t_0 : t_0 + w_t] = 0 \tag{2.16}$$

2. Frequency Masking
   - Number of masks: $N_{freq} = 2$
   - For each mask, sample calculate expression given as Equation 2.17

$$w_f \sim \text{Uniform}(0, F_{max}), \quad F_{max} = 7, \qquad f_0 \sim \text{Uniform}(0, F - w_f) \tag{2.17}$$

then zero out the block shown in Equation 2.18:

$$M[f_0 : f_0 + w_f, :] = 0 \tag{2.18}$$

Equivalently, letting $1_{ms} \in \{0,1\}^{F\times T}$ indicate all masked regions, the augmented MFCC is canbe expressed as shown in Equation 2.19:

$$\widehat{M} = M \odot (1 - 1_{ms}) \tag{2.19}$$

Each mask is resampled per training example on every epoch via our `tf.data` pipeline, ensuring maximal diversity. Park et al. report up to a 15 % relative WER reduction on LibriSpeech, demonstrating the regularization strength of SpecAugment. Figure 2.7 illustrates the zeroed regions produced by SpecAugment on the MFCC matrix.

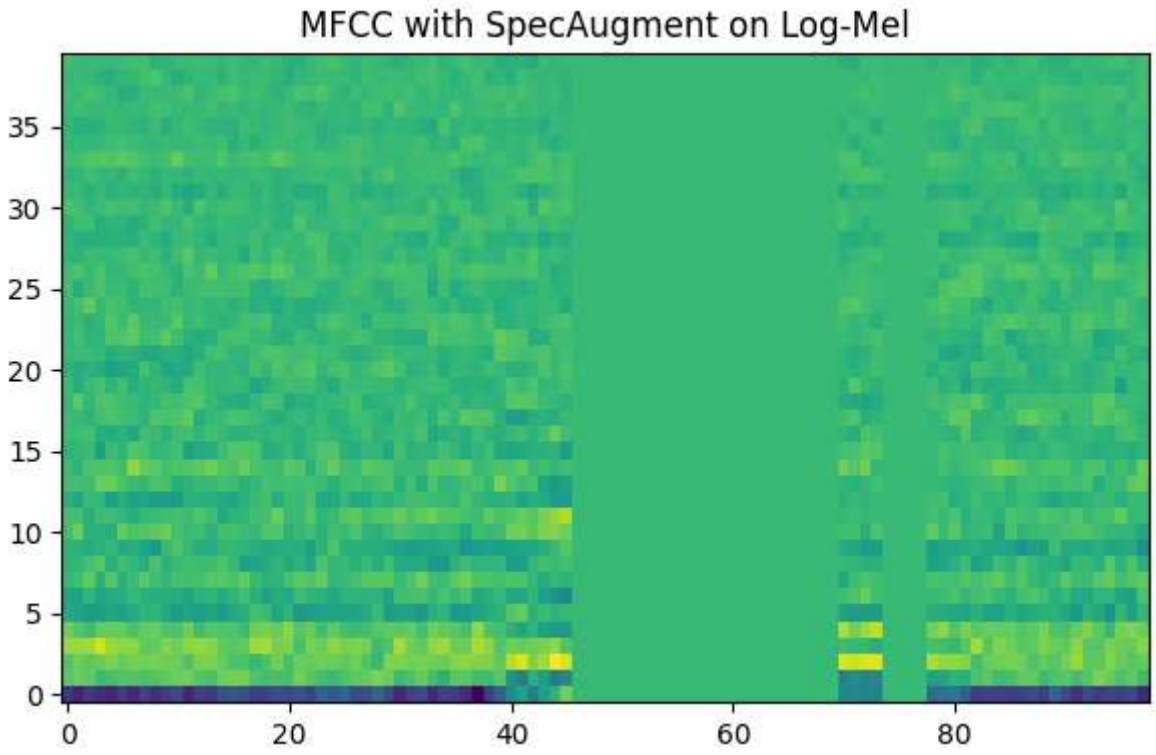

Figure 2.7. MFFC after applying SpecAugment

2.2.4.3. Dataset pipeline optimization

To maximize data throughput and training efficiency, the dataset was processed using TensorFlow's `tf.data` API with the following optimizations:

- Parallel Mapping: All transformations were executed using `num_parallel_calls=tf.data.AUTOTUNE` to leverage available CPU threads.
- Caching and Prefetching: Datasets were optionally cached and prefetched to minimize I/O bottlenecks.
- Batching: Spectrograms were batched using `drop_remainder=True` to ensure uniform tensor dimensions compatible with TPU execution.

Table 2.2. Data augmentation values and methods

Step	**Value / Method**
Audio duration	1.0 sec (16,000 samples)
STFT window size	480 samples (30 ms)
STFT hop length	160 samples (10 ms)
MFCC coefficients	40
Time frames	98
Background noise volume	0.0 to 0.1
Speed perturbation range	0.85× – 1.15×
Time shift range	±1600 samples (±100 ms)
SpecAugment time mask	2 masks, up to 25 frames

These preprocessing and augmentation steps form the foundation of the KWT-1 model's input pipeline and were instrumental in enabling the model to perform robustly in low-SNR and acoustically variable environments, as further explored in subsequent sections.

To quantify the contribution of each augmentation technique, we applied augmentations incrementally in the following order: random time shifting, speed perturbation, background-noise mixing, and finally SpecAugment. At each stage, we observed a stepwise improvement in validation accuracy compared to the previous configuration. Moreover, when SpecAugment masks and background-noise parameters were sampled only once at the start of training (rather than resampled each epoch), the training loss plateaued prematurely and validation performance degraded significantly. These findings demonstrate that resampling augmentation parameters on every epoch is essential for the model to adapt to diverse inputs and to learn effectively.

2.2.5. Training configuration and evaluation

The KWT-1 model was trained using a supervised learning setup designed to closely replicate the experimental conditions described in [5]. This section outlines the optimizer settings, learning rate schedule, loss function, and evaluation methodology used to establish the baseline performance of the model under clean and mildly noisy conditions.

2.2.5.1. Loss function and label smoothing

The classification task was framed as a 12-class single-label prediction problem. To enhance generalization and prevent overconfidence in the SoftMax output, label smoothing was applied. Specifically, ground truth labels were converted into smoothed one-hot vectors as following the expression given in the Equation 2.20:

$$\widetilde{y_i} = \{1 - \alpha + \frac{\alpha}{K}, \text{if i} = y \ \frac{\alpha}{K}, , \{\text{otherwise } \#(2.20)$$

where $\alpha = 0.1$ is the smoothing factor, $K = 12$ is the number of classes, and y is the true class index. This smoothed label was used with categorical cross-entropy loss to compute gradients during backpropagation.

2.2.5.2. Optimization strategy

Training was performed using the AdamW optimizer, which decouples weight decay from gradient-based updates and has demonstrated improved performance for transformer architectures. The key hyperparameters were as follows:

- Base learning rate:$\eta_0 = 1 \times 10^{-3}$
- Weight decay: 0.1
- Beta values: $\beta_1 = 0.9, \quad \beta_2 = 0.999$
- Gradient clipping: not applied

To stabilize training and improve convergence, a warmup phase followed by cosine decay was used to schedule the learning rate as we shown in equation 2.10 where t is the global training step, $t_{warmup} \approx 1800$, and $T = 23000$ is the total number of training steps.

2.2.5.3. Training infrastructure

Model training was conducted on a Google Colab TPU v3 using the `tf.distribute.TPUStrategy()` framework. A batch size of 512 was selected to fully utilize TPU memory while enabling stable gradient estimates.

On the Google Colab TPU v3-8 platform, each training epoch required on average 79 s of wall-clock time; thus, over 140 epochs the total training duration amounted to approximately three hours.

- Batch size: 512
- Number of training steps: 23,000
- Epochs: $\left\lceil \frac{\text{Training Steps}}{\text{Training Data size/Batch Size}} \right\rceil \approx 140$

- Hardware: TPU v3-8 (Colab)

Intermediate validation was conducted after each epoch using a held-out validation set specified by `validation_list.txt`. The final model was selected based on best validation accuracy.

2.2.5.4. Performance evaluation

After training, the model was evaluated on the standard test set specified by `testing_list.txt`. Performance metrics were computed as follows:

- Accuracy: The primary evaluation metric, defined as the percentage of correctly predicted labels.
- Loss: Cross-entropy loss on the test set.
- Confusion Matrix (optional): To analyze class-wise prediction patterns.

The test accuracy obtained was consistent with the original results reported for KWT-1 on the same task (approx. 90.17%), confirming the fidelity of the reimplementation and the adequacy of the training setup.

Figure 2.8 shows the progression of training and validation loss over 140 epochs. The training loss decreases steadily from approximately 1.90 to 0.94, while the validation loss declines from about 1.65 to 0.78, indicating that the model converges without dramatic overfitting. The gap between the two curves narrows after epoch 60, suggesting that the regularization and augmentation strategies maintain generalization throughout training.

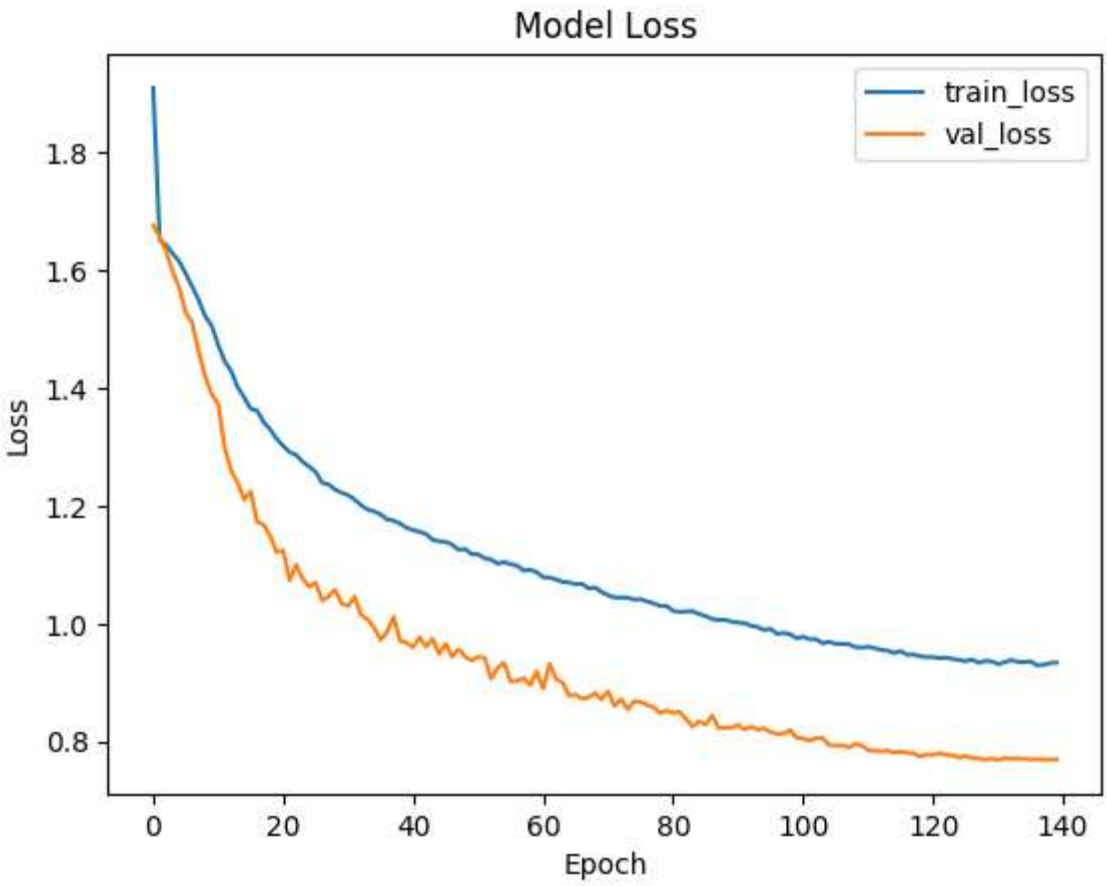

Figure 2.8. KWT-1 implantation training and validation losses

Figure 2.9 presents the corresponding accuracy curves. Validation accuracy rises sharply during the first 40 epochs—reaching roughly 0.85—and then gradually plateaus at around 0.90 by the 140th epoch. The training accuracy follows a similar trajectory but remains slightly lower than validation, which reflects the model's stable performance on unseen data.

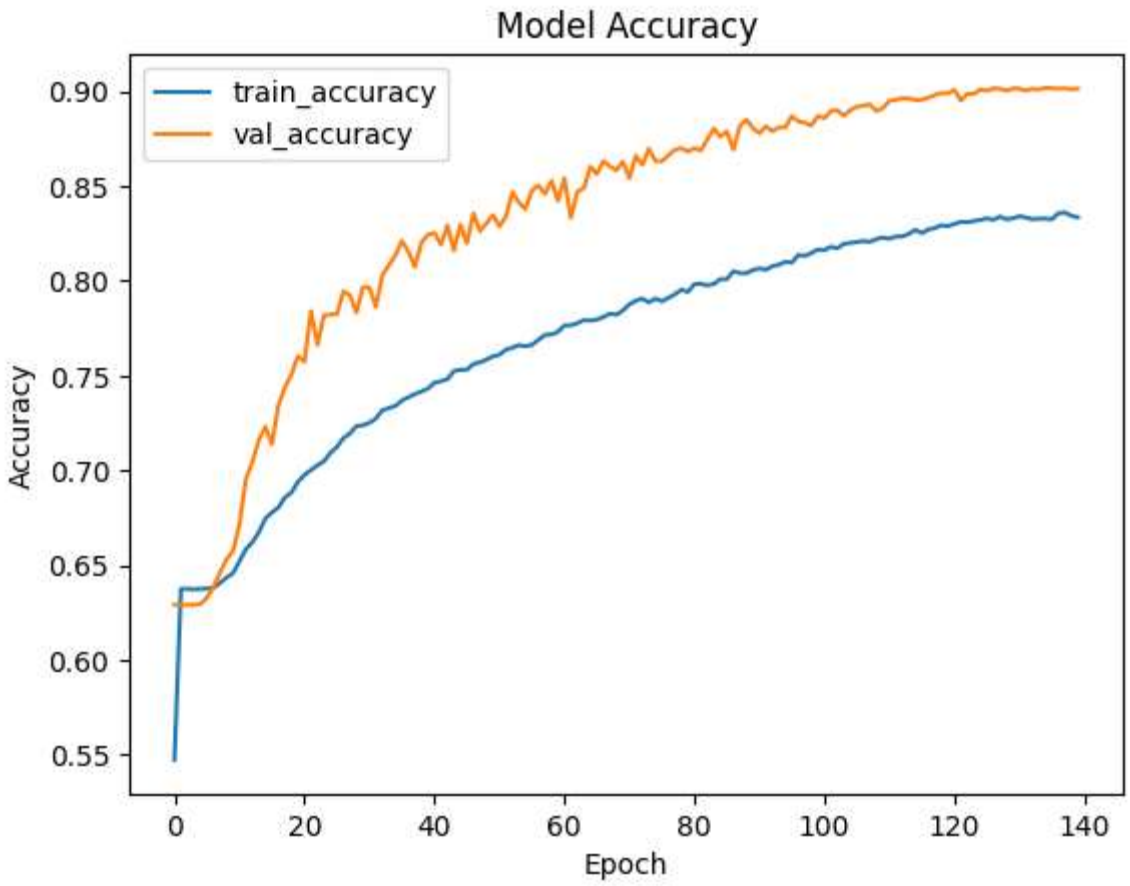

Figure 2.9. KWT-1 implantation training and validation accuracies

Table 2.3. KWT-1 training parameters summary

Parameter	**Value**
Optimizer	AdamW
Learning rate schedule	Warmup + Cosine Decay
Weight decay	0.1
Label smoothing factor	0.1
Total steps	23,000
Batch size	512
Training device	TPU v3
Final test accuracy	~90.17%

The resulting trained model serves as the baseline for all subsequent experiments involving noisy inference and multimodal fusion. Its reproducibility and clean-condition

performance provide a controlled foundation for evaluating the added utility of SNR and noise-type predictions.

2.2.6 Analytical validation and reproducibility

To assess the fidelity of the Keyword Transformer (KWT-1) implementation presented in this thesis, a detailed comparison was made against the methodological specifications and performance benchmarks established in the original study by Berg et al.[5]. Particular emphasis was placed on verifying reproducibility, validating architectural correctness, and isolating the effects of controlled divergences.

2.2.6.1. Performance alignment with baseline results

The final test accuracy achieved by the implemented KWT-1 model was 90.17 %, evaluated on the official Google Speech Commands v2 test set. This performance is substantially lower than the 97.72 % baseline reported by Berg et al. for the non-distilled KWT-1 configuration[5]. Our reimplementation comprises approximately 603 K parameters—about 4 K fewer than the 607 K reported in the original study—and omits the distillation token; these differences likely contributed to the observed reduction in accuracy.

2.2.6.2. Controlled divergences and justifications

The only intentional divergence from the original implementation pertains to the exclusion of the knowledge distillation mechanism. While Berg et al. proposed a distillation setup wherein the class token received attention from a supervised distillation token derived from a teacher model, this component was omitted in the present study. Notably, the empirical results reported in the original paper indicate that the performance gain obtained through distillation was marginal—on the order of 0.1% absolute accuracy—which was not deemed sufficient to justify the added architectural complexity. Furthermore, the primary focus of this thesis is to explore the utility of auxiliary acoustic representations, specifically noise-type classification and signal-to-noise ratio (SNR) estimation, in enhancing keyword spotting robustness. To preserve interpretability and isolate the effects of these auxiliary modules, the base KWT-1 model was evaluated in its non-distilled form.

Importantly, all other aspects of the model configuration—including the number of layers (12), model dimension (64), number of attention heads (1), feed-forward expansion

(256), and patching strategy—were replicated with high fidelity. The learning rate schedule, label smoothing factor, batch size, and augmentation policies were also aligned with the original study.

2.2.6.3. Hardware and environment considerations

While the original implementation was benchmarked on GPUs, this study utilized Google Colab's TPU v3 platform. Despite this infrastructure shift, no discrepancies were observed in terms of training stability, convergence speed, or final model accuracy. All training was conducted from scratch, without access to pretrained weights or external data sources.

In addition, the model and all associated layers were implemented in TensorFlow 2.18 with explicit serialization support via Keras' custom layer registration. The entire training pipeline—comprising data preprocessing, augmentation, model training, and evaluation—was developed to ensure compatibility with Colab environments and reproducibility under standard open-source tooling.

2.2.6.4. Reproducibility assurance

To ensure experimental reproducibility, all random operations in the dataset pipeline were controlled through consistent seeding. The training script, model configuration, and checkpoints were saved and version-controlled. The trained model was exported in `.keras` format and backed up for reuse in downstream modules involving noise classification, SNR prediction, and fusion analysis.

Table 2.4. Reproducibility summary

Aspect	**Methodological Alignment**
Dataset partitioning	validation_list.txt, testing_list.txt
Model architecture	Identical to KWT-1
Distillation	Not used (controlled exclusion)
Learning rate schedule	Matched (warmup + cosine decay)
Label smoothing	Matched (0.1)
Training environment	TPU v3 (vs. GPU in original study)

Codebase Independence and Reimplementation Justification

The original implementation of the Keyword Transformer provided by Berg et al. is publicly available via the ARM-software GitHub repository [16]However, this codebase depends on legacy versions of TensorFlow and other supporting libraries that are now deprecated and incompatible with current machine learning infrastructure, such as TensorFlow ≥2.12 and Google Colab's TPU environment. As a result, a complete reimplementation of the KWT-1 model and its associated training pipeline was undertaken in this study. This reimplementation was executed from first principles using the TensorFlow 2.18 Keras API, and all components—including custom Transformer layers, data preprocessing routines, augmentation logic, and training utilities—were rewritten to ensure compatibility, reproducibility, and extensibility. This approach not only aligned the codebase with contemporary practices but also allowed for fine-grained control over architectural experiments and integration with auxiliary models developed later in the thesis.

In conclusion, the results obtained validate the reimplementation of KWT-1 as functionally equivalent to its original design, and confirm the suitability of the model as a stable baseline for further experimentation. The next chapters of this thesis will examine whether performance under noisy conditions can be further enhanced through the integration of complementary acoustic information—specifically, environmental noise type and signal-to-noise ratio estimates.

2.3. Noise Type Classification

2.3.1 Motivation and background

The ability of keyword spotting systems to perform reliably in real-world acoustic environments depends not only on their recognition capacity but also on their adaptability to background conditions. In this context, a noise type classification module was developed to infer the acoustic environment based on the characteristics of the background sound. This module operates as an independent classifier trained to recognize environmental noise categories, enabling the system to incorporate contextual acoustic information at inference time.

The model design draws structural inspiration from the end-to-end architecture proposed by Abdoli et al. (2019), where audio classification is performed directly on raw waveforms using a stack of one-dimensional convolutional layers [17]. Unlike earlier systems that rely on spectrograms or other hand-engineered features, this approach attempts

to model the entire transformation from waveform to classification decision within the network itself, enabling more direct optimization and eliminating lossy preprocessing stages.

2.3.1.1 Acoustic characteristics of environmental noise

Environmental noise types exhibit considerable variability in both spectral and temporal domains, making them a challenging classification target. The UrbanSound8K dataset includes ten distinct classes that range from impulsive, short-duration sounds (e.g., _gun_shot_, _car_horn_) to continuous background sounds (e.g., _air_conditioner_, _engine_idling_). These variations directly affect the temporal envelope and frequency content of the waveform, which in turn influences how early and deep layers of the model respond during feature extraction.

Transient events such as _siren_, _dog_bark_, or _gun_shot_ are characterized by sudden spectral onsets, sharp energy peaks, and narrow temporal footprints. In contrast, categories such as _street_music_ or _children_playing_ often contain overlapping sound sources with broader frequency distributions and less predictable structure. Additionally, certain noise types—particularly _drilling_ and _jackhammer_—may exhibit periodic rhythmic features, further complicating their separation from each other.

These factors necessitate a classification model that can capture both local acoustic events and broader temporal dynamics. A raw waveform input preserves this structure without transformation losses, allowing the model to learn optimal filters for different noise morphologies.

2.3.2 Dataset and preprocessing

The classifier was trained and evaluated on the UrbanSound8K dataset, which includes 8,732 labeled audio clips grouped into 10 urban noise categories. The dataset was first unpacked and restructured into a format compatible with the model's expectations. Each audio file was resampled to 16 kHz, converted to mono, and segmented into 1-second frames (16,000 samples) with 50% temporal overlap, using a sliding window. Each frame was paired with its corresponding `classID` (0–9) and `clipID`.

All ten official folds provided in UrbanSound8K were used to extract frames in parallel. This ensured that the dataset captured the full diversity of acoustic events and recording conditions. However, the experiment did not follow a fold-based cross-validation protocol. Instead, the entire dataset was treated as a unified pool, and frames were randomly split into training and validation sets using stratified sampling. This choice was made to

support efficient training of the model as a standalone noise classifier, rather than evaluating generalization across specific recording sessions or folds.

Frames shorter than one second were zero-padded. The resulting dataset was a 2D NumPy array where each row contained a waveform frame, the associated noise class label, and the encoded clip identifier. Labels were then one-hot encoded for use with the SoftMax classification objective.

2.3.3. Model architecture

The noise type classifier is implemented as a one-dimensional convolutional neural network derived from the general structure of EnvNet[18]and the biologically inspired extension by Abdoli et al. (2019) [17]The model accepts raw waveform input of shape (16000, 1) and outputs a ten-dimensional SoftMax vector corresponding to the noise class probabilities.

Unlike the model in [17], which uses trainable filters throughout, the present implementation introduces a non-trainable Gammatone filter bank as the first layer. These filters approximate the impulse response of the human cochlea by distributing center frequencies on an Equivalent Rectangular Bandwidth (ERB) scale. - The overall model architecture, comprising convolutional, pooling, and fully connected layers, is illustrated in Figure 2.10.

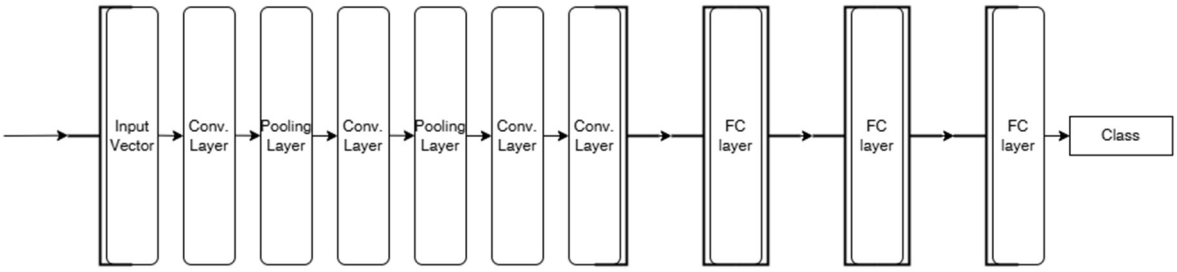

Figure 2.10. Noise type detection model architecture

2.3.3.1 Gammatone filter banks: Theory and application

The impulse response of a Gammatone filter is defined as g(t) given in 2.22:

$$g(t) = at^{n-1}e^{-2\pi bt}\cos\cos(2\pi f_c t + \phi), \quad t > 0 \tag{2.22}$$

where a is an amplitude scaling factor, n is the filter order, b is the bandwidth based on ERB, and f_c is the centre frequency. The ERB scale is given calculation of Equation 2.23:

$$\mathrm{ERB}(f) = 24.7 + 0.108f \tag{2.23}$$

This formulation allows frequency resolution to be distributed non-linearly in a biologically realistic way. Filters were initialized across the ERB scale starting at 100 Hz, with 64 filters in total (one per output channel in the first Conv1D layer). The Gammatone filters preserve both phase and fine temporal structure, unlike Fourier-based features such as MFCCs. Fixing this layer during training reduces learnable parameters and imposes an inductive prior reflecting early auditory processing.

2.3.3.2 Temporal resolution and stride strategy

The model's convolutional layers are designed to gradually reduce temporal resolution while increasing abstraction. The first layer uses a large kernel (512 samples, ~32 ms) and stride 1 to preserve timing fidelity. This is followed by a series of convolutional and pooling operations with increasing stride to capture longer-range dependencies.

Intermediate layers (kernel sizes 32, 16, 8; strides 2) progressively expand the receptive field while preserving fine-grained onset patterns. Max pooling (pool size 8) compresses the temporal dimension, reducing computation and improving generalization. The final convolutional layer outputs feature maps that summarize hundreds of milliseconds of context—crucial for classes such as _siren_ and _jackhammer_.

2.3.3.3 Model capacity and regularization strategy

The model includes two dense layers with 128 and 64 units, respectively, each followed by dropout (rate = 0.25). These sizes were chosen to balance representational capacity with overfitting control. Larger configurations showed diminished generalization in early experiments. Dropout helps prevent co-adaptation and encourages redundancy in learned features.

The use of a fixed Gammatone layer also serves as architectural regularization, freeing capacity for deeper layers to specialize in discriminating between similar noise types. This design supports better convergence and a more interpretable feature hierarchy.

2.3.4 Training configuration

The model was compiled using the Adam optimizer with a learning rate of 0.001. Although Abdoli et al. [17]originally employed the Adadelta optimizer, the present study utilized Adam to achieve faster convergence and improved training stability. The loss function was mean squared logarithmic error (MSLE). Two callbacks were applied:

EarlyStopping (patience = 10) and ReduceLROnPlateau (patience = 5, factor = 0.5). Training proceeded for up to 100 epochs with a batch size of 100.

Table 2.5. SNR prediction model training parameters

Parameter	**Value**
Optimizer	Adam
Learning rate	0.001
Loss function	Mean Squared Log Error
Batch size	100
Epochs (max)	100
Early stopping	Patience = 10
Reduce LR on Plateau	Patience = 5, factor = 0.5
Input shape	(16000, 1)
Dropout rate	0.25

2.3.4.1 Training dynamics and learning behavior

Figure 2.12 shows the training and validation accuracy across epochs. During the first ten epochs, the model achieves rapid gains—from approximately 35 % to 80 % on the training set and from 50 % to 75 % on the validation set—indicating effective initial representation learning. Between epochs 15 and 20, both curves begin to plateau, at which point the ReduceLROnPlateau callback reduces the learning rate, producing a modest recovery in validation accuracy and culminating in final accuracies of approximately 97.5 % (training) and 89.3 % (validation) by epoch 70. As depicted in Figure 2.12, the parallel trajectories of the two curves throughout training confirm stable convergence without overfitting.

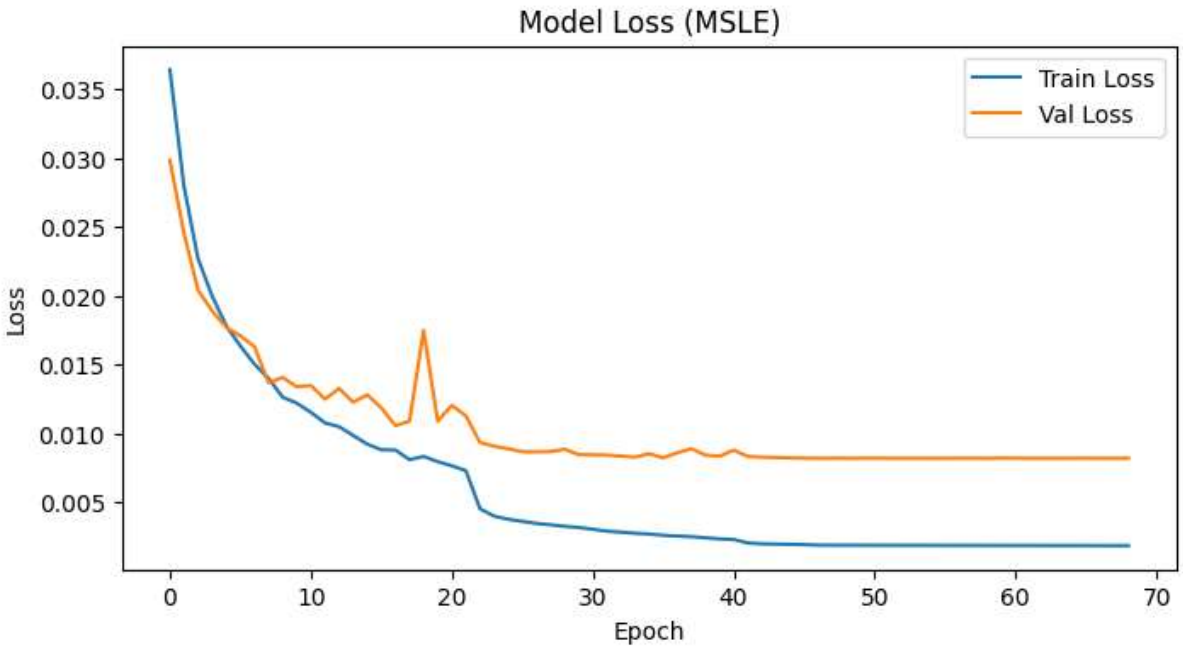

Figure 2.11. Noise Type Detection model training and validation losses

Figure 2.11 further illustrates that both training and validation loss decrease sharply during the early epochs, falling below 0.02 by epoch 10. A small spike in validation loss appears around epoch 20—coincident with the learning-rate reduction—after which both loss curves converge to approximately 0.002 (training) and 0.008 (validation). The absence of sustained oscillations or divergence in the loss curves, coupled with the small gap between training and validation metrics, validates the effectiveness of the chosen optimizer, dynamic learning-rate schedule, batch size, and regularization strategies.

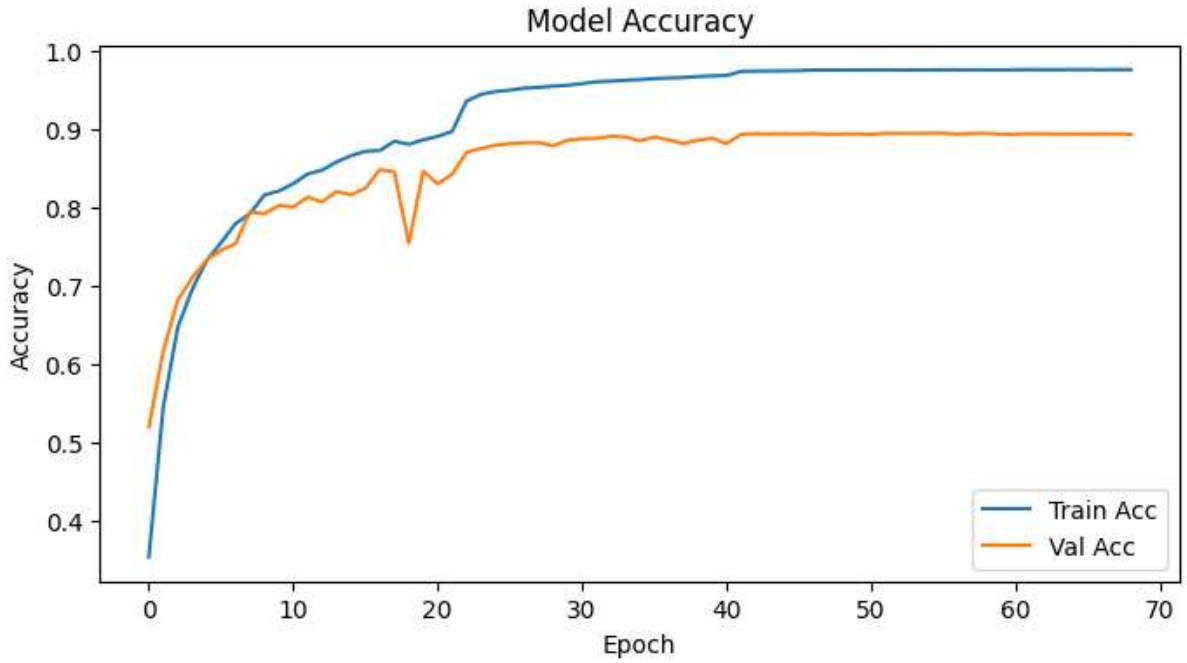

Figure 2.12. Noise Type Detection model training and validation accuracies

2.3.5 Evaluation procedure

The model was evaluated at the clip level. For each `clipID`, frame-level SoftMax predictions were summed, and the class with the highest aggregated probability was selected. This aggregation preserves the semantic identity of clips across frames and reduces variance from outlier frames.

Metrics included:

- Frame-level validation accuracy and loss
- Clip-level accuracy
- Confusion matrix
- Classification report (precision, recall, F1-score)

2.3.5.1 Confusion matrix and per-class insights

_car_horn_, which demonstrated minimal confusion with other categories. The most significant confusion occurred between _dog_bark_ and _children_playing_, likely due to overlapping acoustic characteristics such as sudden onsets and wide spectral coverage. Additional overlap was observed between _children_playing_ and _street_music_, attributed to the presence of musical and vocal elements. Moderate confusion was also noted between _drilling_ and _jackhammer_, reflecting similarities in their periodic rhythmic structures, with the clip-level performance closely matching that reported by Abdoli et al.

[17](approximately 90% in the present study versus 89% in the original work). The small improvement in accuracy observed may be attributable to the use of a learning rate scheduling strategy (ReduceLROnPlateau), which was not explicitly described in the original study. The detailed confusion matrix, depicted in Figure 2.13, highlights both the classifier's robust performance and the primary areas of confusion between acoustically similar classes.

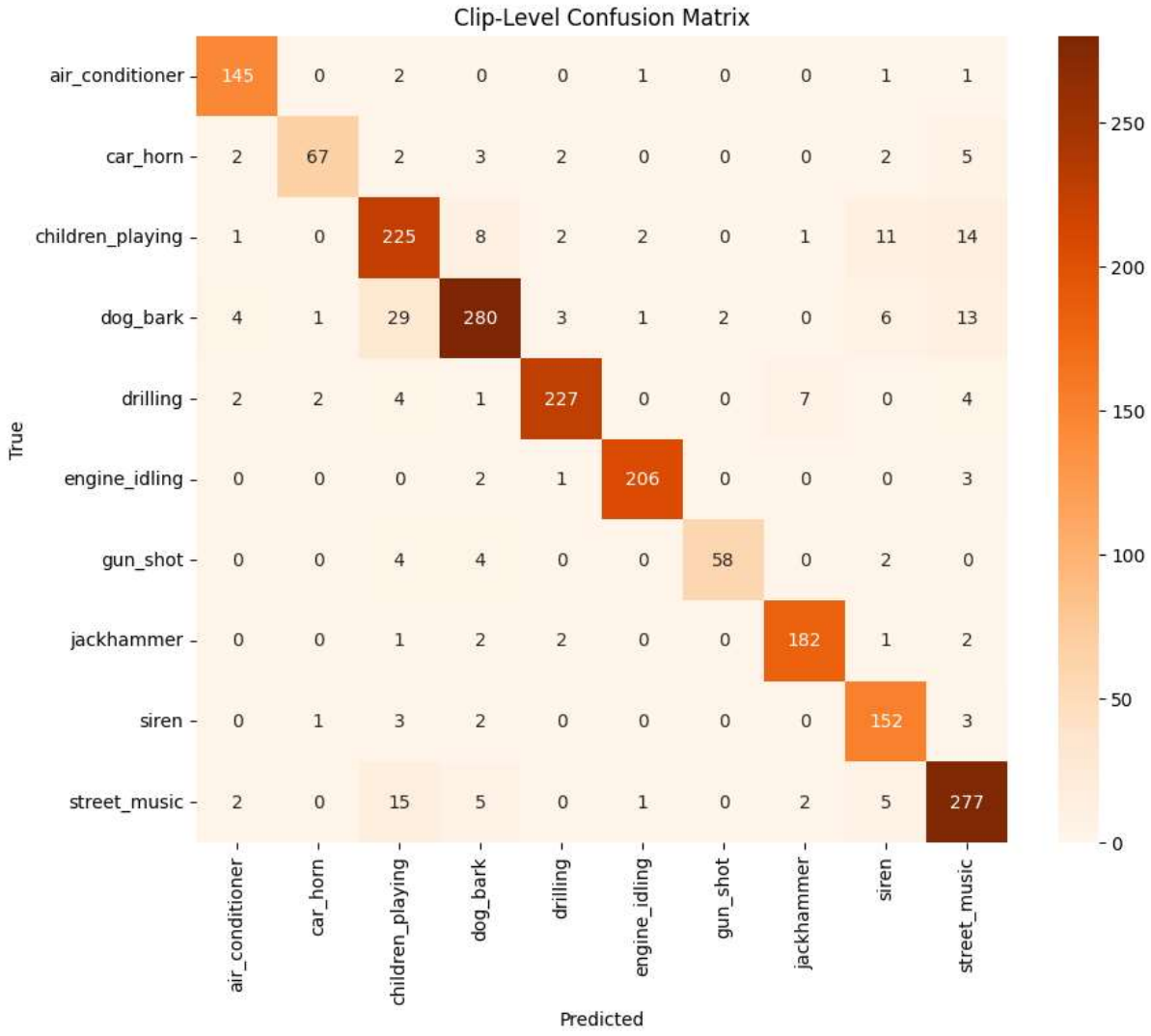

Figure 2.13. Noise Type Detection Model clip level confusion matrix

2.3.6 Observations

The Gammatone-initialized, end-to-end 1D CNN architecture achieved high clip-level accuracy across diverse environmental noise classes. The model was compact, interpretable, and robust to overfitting.

2.3.6.1 Implementation details and reproducibility

The model was implemented using TensorFlow 2.18 in Google Colab with reproducible data preparation. All experiments were conducted on a Colab-provided NVIDIA A100 GPU instance. Each training epoch required approximately 7 seconds, and early stopping typically occurred after 70 epochs, resulting in a total training time of roughly 8 minutes. Waveform loading and resampling were performed using Librosa, and parallel frame extraction was done with `ThreadPoolExecutor`. To ensure reproducibility, random seeds were set for dataset splitting and model initialization. The final model was serialized in `.keras` format, including all custom components such as the Gammatone-initialized convolutional layers.

2.4 Signal-to-Noise Ratio Estimation

2.4.1 Motivation and background

Robust keyword spotting in real-world acoustic conditions requires not only detecting the spoken content but also adapting to the quality of the received signal. One critical metric for assessing this quality is the signal-to-noise ratio (SNR), which measures the relative energy of speech compared to background noise.

Classical SNR estimation techniques—such as the decision-directed MMSE spectral estimator of Ephraim & Malah [19] and the minimum-statistics noise tracking method of [20]—rely on statistical assumptions (e.g. stationary Gaussian noise) and often require a VAD to segment speech and noise. These methods typically break down when noise is non-stationary or when SNR falls below 0 dB.

More recently, deep-learning approaches have shown superior robustness by learning a direct mapping from noisy speech to SNR. For example, Li et al.[21] employed LSTM networks on frame-level acoustic features and achieved significant error reductions over classical estimators 3. However, most prior work operates on spectrogram inputs or uses heavy sequential models, which incur extra preprocessing cost and latency.

In this work, we propose to estimate SNR directly from raw 1 D waveforms via a lightweight, hourglass-style convolutional neural network. The resulting SNR value is used as an auxiliary feature in our decision-level fusion for keyword spotting (Section 3.4).

2.4.2 Dataset construction

To supervise SNR regression, we dynamically mix clean speech and noise on-the-fly:

- Clean speech: utterances from Google Speech Commands.
- Noise: clips from UrbanSound8K.
- Duration: every sample is normalized to 1 second (16 000 samples at 16 kHz).

Given clean waveform $x(t)$ and noise $n(t)$, both of length $T = 16000$, their RMS energies are calculated as Equation 2.24:

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{T}\sum_{t=1}^{T} x(t)^2}, \quad \mathrm{RMS}(n) = \sqrt{\frac{1}{T}\sum_{t=1}^{T} n(t)^2} \tag{2.24}$$

To mix at a target SNR (dB), we compute the gain via Equation 2.25:

$$\lambda = \frac{\mathrm{RMS}(x)}{\mathrm{RMS}(n)} \cdot 10^{-\frac{\mathrm{SNR_{dB}}}{20}} \tag{2.25}$$

and form the new signal shown in Equation 2.26

$$\hat{x}(t) = x(t) + \lambda \cdot n(t) \tag{2.26}$$

We uniformly sample $\mathrm{SNR_{dB}} \in [0,20]\mathrm{dB}$ for each mixture, re-generating new mixtures every epoch. This "infinite augmentation" strategy matches best practices in speech enhancement 4 and prevents overfitting to particular noise clips.

2.4.3 Model architecture

Our SNR estimator is built as a 1D hourglass CNN: the input waveform is first down-sampled through successive Conv1D + MaxPooling blocks, then up-sampled back to original resolution via UpSampling1D and symmetric Conv1D layers with skip connections, ending in a regression head.

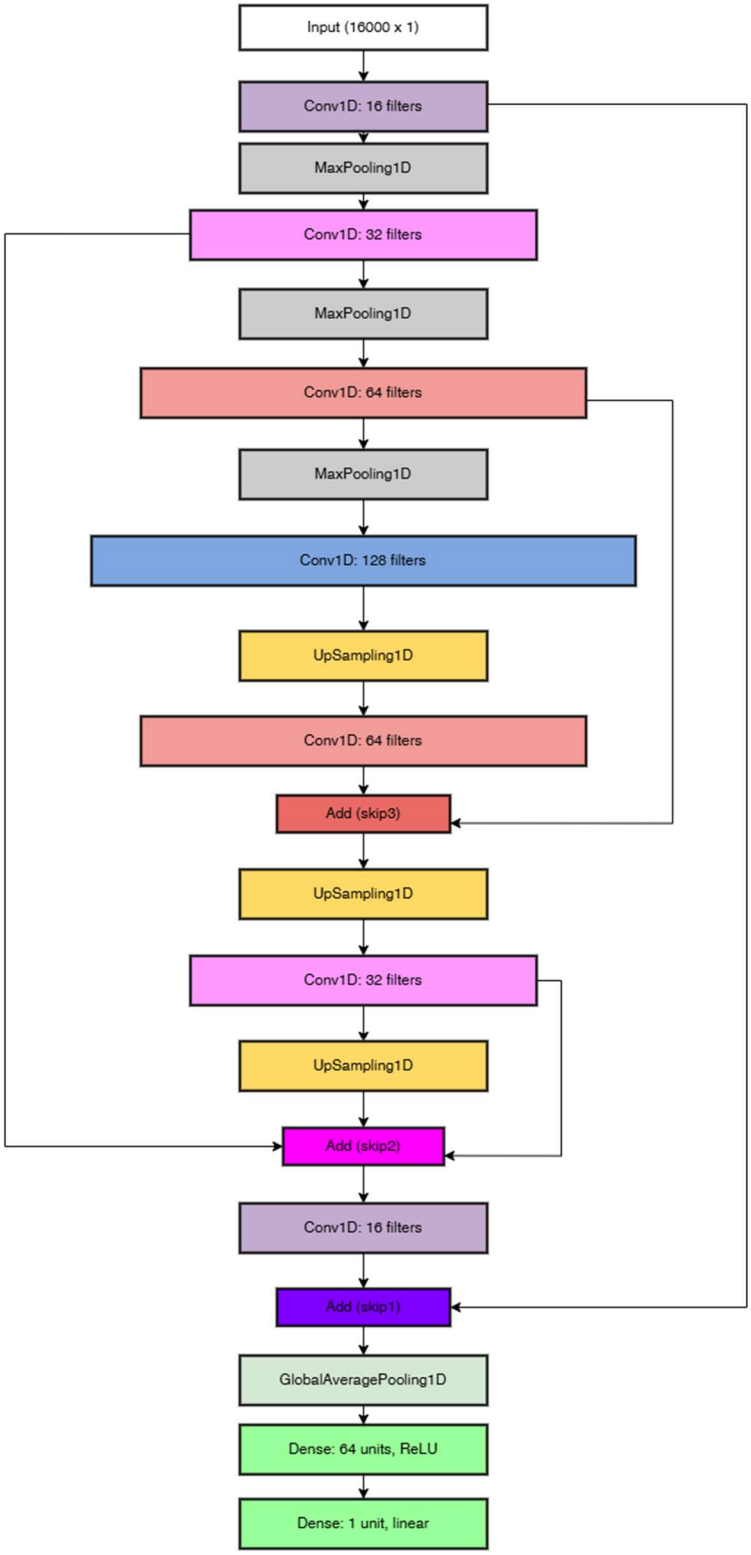

Figure 2.14. Hourglass architecture of the SNR prediction model

This design captures both global energy trends (via pooling) and local waveform details (via skips), without the overhead of spectrogram computation.

2.4.4 Training configuration

We train for 56 epochs on batches of 32 raw-waveform inputs, optimizing by using the formula in the Equation 2.27:

$$\mathrm{ME} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}(\widehat{s_{\mathrm{i}}} - \mathrm{s_i})^2 \tag{2.27}$$

with Adam (lr = 1×10^{-3}). We monitor MAE calculated via the Equation 2.28:

$$\mathrm{ME} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}|\widehat{s_{\mathrm{i}}} - \mathrm{s_i}| \tag{2.28}$$

for early stopping, though only MSE drives training.

2.4.5 Evaluation and results

On a held-out test set (Speech Commands utterances mixed with unseen UrbanSound8K noises at 0–20 dB):

- Test MSE: 2.0287
- Test MAE: 0.8257 dB

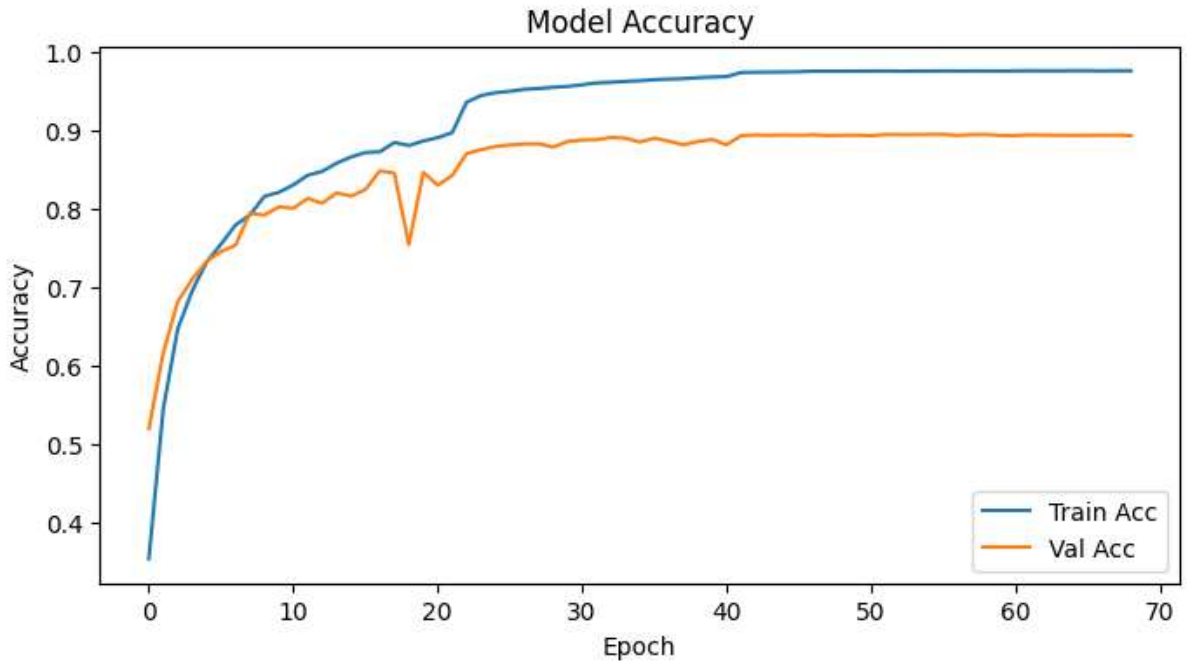

Figure 2.15. SNR prediction model training and validation accuracies

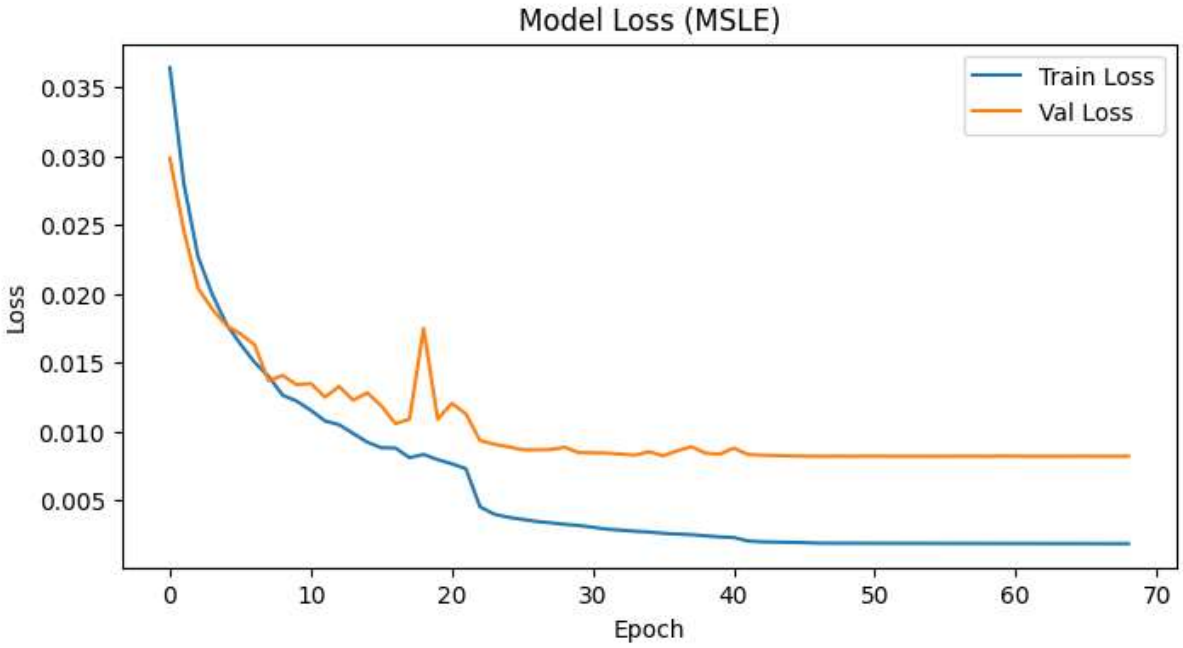

Figure 2.16. SNR prediction model training and validation losses

Compared to classical estimators—Decision-Directed MMSE [19] (MSE > 10 at low SNR) and Minimum-Statistics [20](MAE > 3 dB)—our CNN achieves < 1 dB MAE across the entire 0–20 dB range. Against the LSTM model of Li et al.[21], we match their ~0.8 dB MAE but with lower inference latency (no spectrogram or recurrence).

These results confirm that a raw-waveform hourglass CNN can deliver state-of-the-art SNR estimates in real time, enabling more robust downstream keyword spotting under noisy conditions.

2.5 Decision-Level Fusion via Transformer-Based Meta-Classifier

2.5.1 Motivation and background

Although the KWT-1 model demonstrates high performance under clean conditions, its accuracy deteriorates significantly in noisy environments. To address this limitation, a meta-classifier was designed to integrate complementary acoustic information—namely, noise type and signal-to-noise ratio (SNR)—into the keyword classification process. Rather than relying on early or intermediate feature fusion, the proposed approach adopts a decision-level fusion strategy. In this framework, the outputs of three independently trained models are combined using a lightweight transformer encoder to generate the final keyword prediction.

Each model is responsible for capturing a distinct aspect of the acoustic scene:

- The KWT-1 model predicts the spoken keyword based on MFCC representations.
- The EnvNet model identifies the noise type from raw waveforms.
- The hourglass CNN estimates the SNR level of the input signal.

These outputs are treated as tokens, projected into a shared embedding space, and jointly processed by a transformer encoder. This modular design allows each model to operate independently while enabling the fusion layer to learn cross-modal interactions at the prediction level.

2.5.2 Model architecture and tokenization

The meta-classifier receives a fused input vector consisting of three components are expressed as 2.29:

$$\begin{aligned} z_{KWS} &\in R^{12} \quad \text{: softmax output from the KWT-1 model} \\ z_{noise} &\in R^{10} \quad \text{: softmax output from the noise classifier} \\ &\in R \quad \text{: scalar SNR value predicted by the hourglass CNN} \end{aligned} \tag{2.29}$$

Each of these is independently projected into a common d-dimensional embedding space, where d=64 The projection functions are defined in Equation 2.30:

$$\begin{aligned} t_{KWS} &= \phi_{KWS}(z_{KWS}) = W_{KWS}z_{KWS} + b_{KWS} \in R^d \\ t_{KWS} &= \phi_{KWS}(z_{KWS}) = W_{KWS}z_{KWS} + b_{KWS} \in R^d \\ t_{SNR} &= \phi_{SNR}(s) = W_{SNR}s + b_{SNR} \in R^d \end{aligned} \tag{2.30}$$

The resulting token embeddings are concatenated to form a token sequence is concatenated as shown in Equation 2.31:

$$\mathbf{T} = \begin{bmatrix} t_{KWS} \\ t_{noise} \\ t_{snr} \end{bmatrix} \in R^d \tag{2.31}$$

A learnable positional embedding matrix $\mathbf{P} \in R^{3\times d}$ is added element-wise to encode positional information to compute Equation 2.32:

$$\mathbf{X_0} = \mathbf{T} + \mathbf{P} \tag{2.32}$$

This embedded sequence $\mathbf{X_0}$ serves as input to a transformer encoder block comprising multi-head self-attention (4 heads), a feed-forward subnetwork (hidden dimension = 128), dropout, residual connections, and layer normalization. The output is globally averaged and passed through a final SoftMax classifier to produce the keyword prediction.

Figure 2.17 depicts the overall Transformer-based meta-classifier architecture. The model first ingests three distinct inputs— (1) the 12-dimensional keyword SoftMax from KWT-1, (2) the 10-dimensional noise-type SoftMax, and (3) the scalar SNR estimate—and projects each through a dedicated linear layer into a shared 64-dimensional embedding space. These three token embeddings are then concatenated into a sequence and augmented with learnable positional embeddings to encode their modality order. The resulting sequence is processed by a two-layer Transformer encoder, each layer comprising four attention heads, a 128-dimensional feed-forward subnetwork, residual connections, dropout, and layer normalization. Finally, the pooled [CLS] token output is fed through a SoftMax classification head to produce the final 12-way keyword prediction. This design enables the meta-classifier to dynamically attend across heterogeneous acoustic and noise features, yielding more robust keyword recognition under varying environmental conditions.

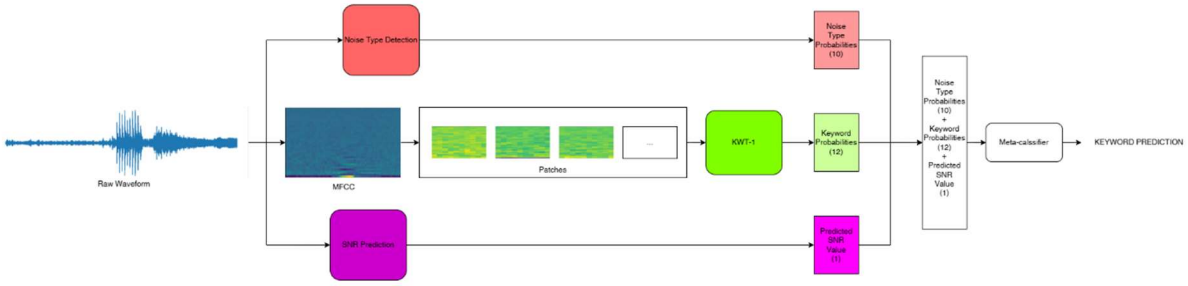

Figure 2.17. Meta-classifier illustrating the input pipeline

The structural layout of the fusion module is illustrated in Figure 2.18, showing how the three tokenized outputs are embedded, combined with positional encodings, and passed through the transformer block described mathematically in Equations

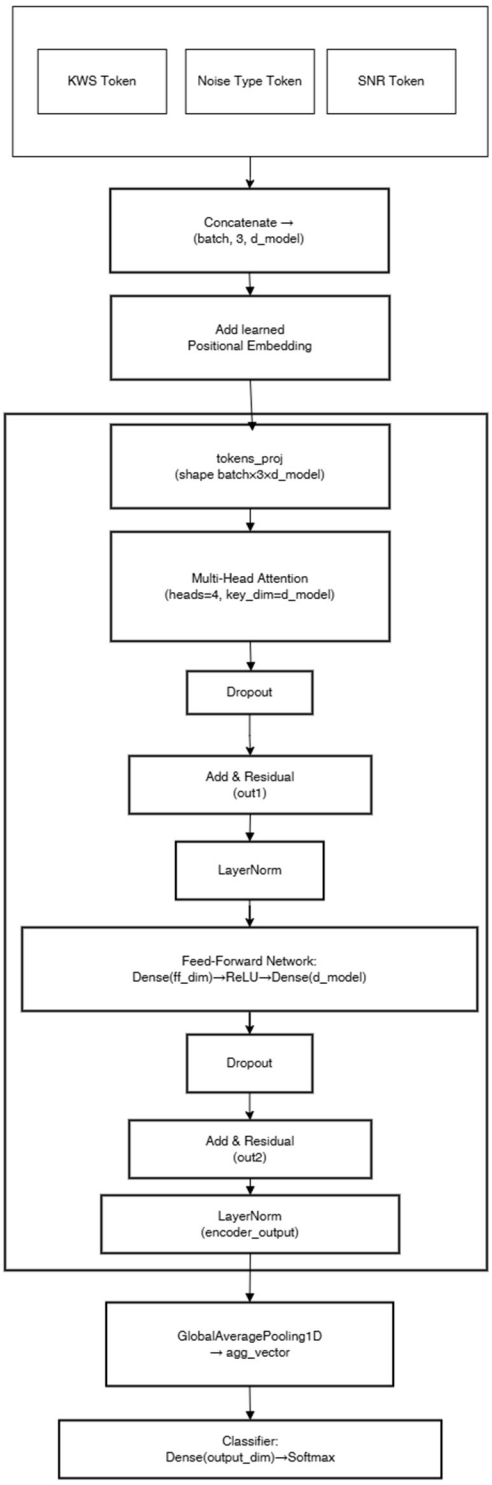

Figure 2.18. Meta -classifier transformer architecture

2.5.3 Dataset construction

To train the meta-classifier, a dataset of fused output vectors was synthesized from the Google Speech Commands v2 dataset. Each utterance was normalized to 1 second and sampled at 16 kHz. Background noise samples were drawn from the UrbanSound8K dataset,

and the clean utterance x(t) was mixed with noise n(t) at a randomly sampled SNR level between 0 and 20 dB as expressed in Equation 2.33:

$$\hat{x}(t) = x(t) + \lambda \cdot n(t), \quad \lambda = \frac{RMS(x)}{RMS(n)} \cdot 10^{-SNR_{dB}/20} \tag{2.33}$$

The resulting noisy waveform was passed through each of the pre-trained models:

- KWT-1 model: predicts a 12-class SoftMax vector from MFCC features.
- EnvNet: predicts a 10-class noise distribution from raw waveform.
- Hourglass CNN: outputs a scalar SNR estimate from waveform input.

The outputs were concatenated into a 23-dimensional feature vector and paired with the original ground-truth keyword label. To improve generalization, each training sample was augmented three times with different noise samples and SNR values. Official `validation_list.txt` and `testing_list.txt` partitions were preserved for validation and evaluation.

2.5.4 Training configuration

The meta-classifier was trained using the Adam optimizer (learning rate = 0.001) and sparse categorical cross-entropy loss. Training was conducted for up to 200 epochs with early stopping (patience = 15) and learning rate reduction on plateau (patience = 5, factor = 0.5). A batch size of 128 was used. Training and evaluation were performed using a Google Colab TPU v3 back-end. All data pipelines were optimized with the TensorFlow `tf.data` API, and all augmentation, feature extraction, and inference steps were cached for efficient throughput.

As depicted in Figure 2.19, the meta-classifier's training and validation accuracy both converge above 85 % within the first 50 epochs, exhibiting only a slight generalization gap thereafter. The loss curves in Figure 2.20 mirror this behavior, showing a smooth and sustained decrease over all 200 epochs, which confirms that the fusion tokens are being effectively learned with minimal overfitting.

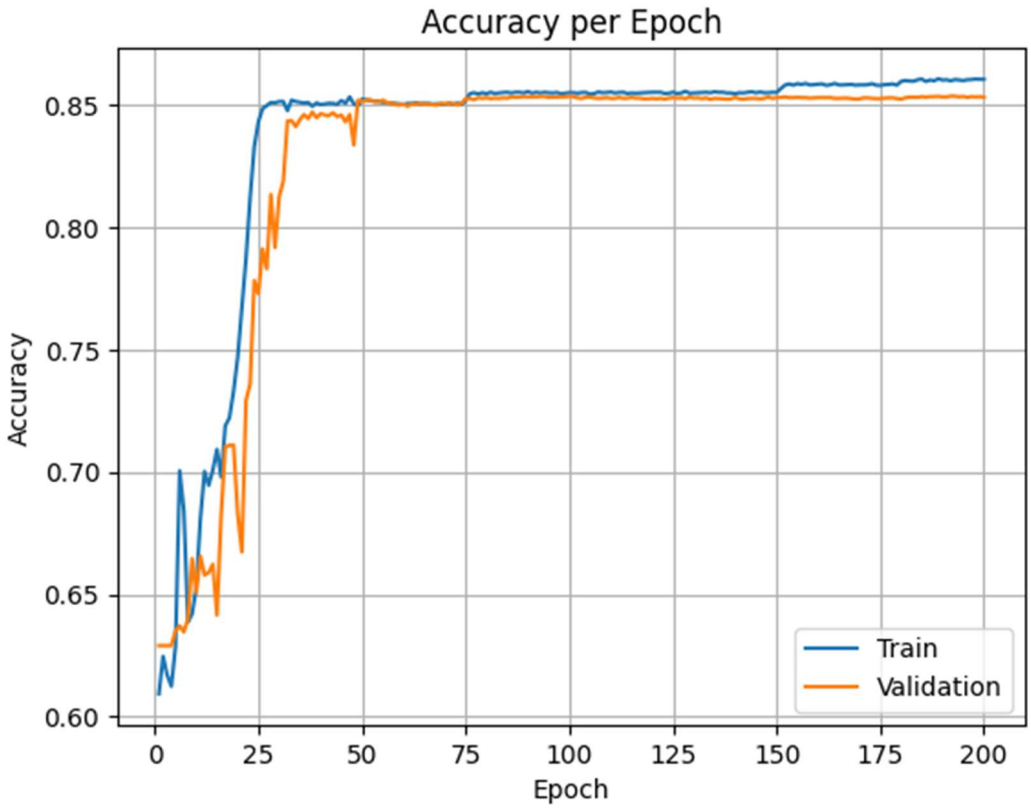

Figure 2.19. Meta classifier training and validation accuracy

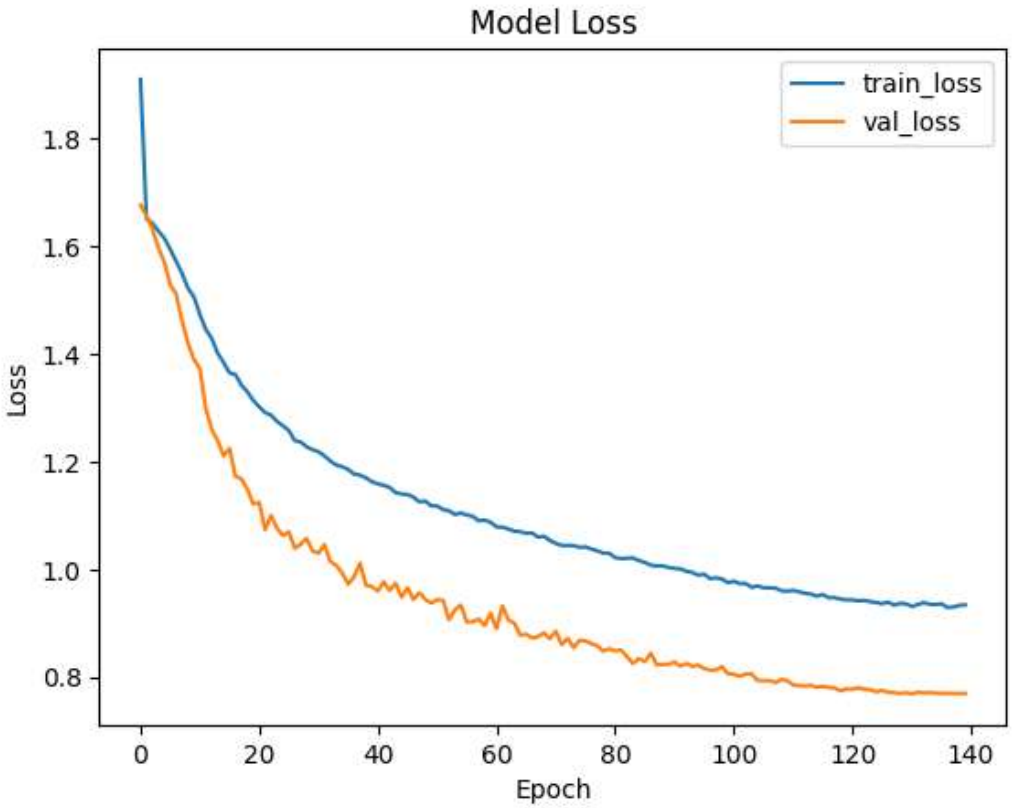

Figure 2.20. Meta classifier training and validation losses

2.5.5 Evaluation and results

The meta-classifier was evaluated on both clean and noisy test sets. Under clean conditions:

- KWT-1 baseline accuracy: 93.57%, test loss: 0.2964
- Meta-classifier accuracy: 93.14%, test loss: 0.2327
- Under noisy conditions (SNR ∈ [0, 20] dB):
- KWT-1 baseline accuracy: 84.93%, test loss: 0.5409
- Meta-classifier accuracy: 84.99%, test loss: 0.4985

Although the meta-classifier produced lower cross-entropy loss in both conditions, the observed gain in classification accuracy was negligible. Since accuracy was the primary evaluation metric, the model cannot be considered a performance improvement over the baseline.

2.5.6 Observations

The transformer-based fusion model provides a modular, interpretable framework for incorporating auxiliary acoustic information into keyword recognition. Its decision-level design allows each branch model to be trained independently and updated without architectural coupling. This property is particularly desirable in real-world systems, where deployment constraints often preclude end-to-end retraining.

Nevertheless, the fusion strategy did not yield a meaningful increase in classification accuracy. The slight improvement in prediction calibration—as indicated by lower test loss—suggests that the transformer meta-classifier can learn to weigh complementary signals. However, this does not translate into improved decision-making when the target metric is accuracy. Future research may explore deeper transformer encoders, alternative token encoding schemes (e.g., discretized SNR bins), or contrastive training objectives to more effectively leverage the multi-source input.

3. CONCLUSION

In conclusion, this work developed a transformer-based keyword spotting system with explicit noise-awareness and provided a thorough evaluation of its components. The proposed architecture is modular, consisting of a high-performance keyword detector augmented by auxiliary noise analysis models and a fusion mechanism. Our approach introduces several technical contributions to the field of keyword spotting and robust speech processing. First, we applied the Keyword Transformer (KWT-1) model to the keyword spotting task and confirmed its effectiveness on the Google Speech Commands benchmark, achieving accuracy on par with the latest reported results. This demonstrates the benefit of a self-attention-based architecture in KWS, as opposed to traditional convolutional or recurrent models. Second, we incorporated a novel SNR estimation from raw audio using an hourglass CNN. To our knowledge, frameworks for KWS have not previously included a dedicated neural network estimating the SNR of the input; our system thus pioneers the integration of this capability. The SNR model learns directly from the waveform, exemplifying end-to-end feature learning for noise characterization. This design choice is in line with recent research trends that leverage supervised learning for SNR estimation to improve over classical approaches. Third, we designed a decision-level integration strategy using a transformer encoder that fuses the outputs of the keyword model, noise type classifier, and SNR estimator. By using a transformer for late fusion, our system can, in principle, learn complex attention mappings between the content and context representations. This fusion module represents a flexible integration of heterogeneous information: the linguistic content (keyword class probabilities) and the environmental context (noise class and level). The thesis thus contributes a comprehensive exploration of how additional context (noise type and SNR) might be combined with a base KWS model in a modular fashion.

The experimental findings, however, revealed a gap between the intended advantages of the fusion approach and the actual outcomes. Notably, we did not observe an improvement in keyword spotting accuracy when using the transformer-based fusion, compared to the standalone KWT-1 model without context. In other words, the added noise type and SNR information _did not translate into better classification performance_ on the test data. This is an important negative result: it suggests that the straightforward decision-level fusion strategy may not effectively harness the provided context, at least not with the training

regime and data we used. Several factors could explain this outcome. One factor is the late integration itself – by the time the transformer sees the information, the KWS model's internal representation might already be too certain (or too confused) about its prediction, leaving limited opportunity for adjustment. Another factor is the independent training of modules; since the KWS, noise classifier, and SNR estimator were trained separately, their outputs were not optimized jointly for the end task, which could lead to a suboptimal fusion (e.g., inconsistent feature scaling or misaligned representations). It is also worth noting that the noise conditions and types were somewhat limited (focused on a few urban sound classes and SNR ranges). The context provided might not have been rich or varied enough to aid the KWS model in general – for instance, KWT-1 might already handle those specific noise conditions well, whereas more unusual or complex noise scenarios (not present in training) could be where context would help. In summary, while the system architecture is conceptually sound and each module performed well in isolation, the combined system did not surpass the baseline. This highlights a key limitation and suggests that more sophisticated methods or training strategies are necessary to realize the full benefit of integrating environmental context into keyword spotting. Nonetheless, the research presented in this thesis establishes a foundation for noise-aware KWS and offers valuable insights. The modular design and findings can inform future efforts, indicating which aspects of the system need refinement. In particular, the lack of accuracy gain through late fusion underscores the importance of when and how the integration of context is done, pointing toward potential improvements as discussed next.

3.1. Future Work

Building on the lessons learned, several research directions emerge to enhance the noise-aware KWS framework:

Joint training of modules: A promising direction is to train the KWS model and the auxiliary context models in a multi-task or end-to-end manner instead of training them independently. By sharing gradients and learning a unified objective, the system could discover representations that better link the presence of certain noise types or SNR levels with adjustments in the keyword decision boundary. For example, the noise classifier and SNR estimator could be treated as additional output heads of the keyword transformer, enabling the encoder to simultaneously learn to recognize keywords _and_ to characterize noise. Recent studies have shown that incorporating a noise classification task into a speech model's training can improve its noise robustness[22]. Joint optimization might therefore

allow the KWT-1 to implicitly adjust its internal features in accordance with the background conditions, yielding a more seamless integration of content and context.

Deeper integration of context into early model stages: Our current fusion occurs at the decision level, after the KWS model has already produced its output. Future work should investigate integrating noise context at earlier stages of the feature extraction or inference process. One idea is to feed the estimated noise characteristics as _inputs or conditioning signals_ to the KWS network. For instance, an embedding representing the noise type (or a continuous noise feature vector) could be concatenated with the acoustic features (e.g. mel-spectrogram or intermediate embeddings) before or during the transformer encoding. Another approach is to insert attentional fusion layers that allow the KWS encoder to attend to noise features alongside speech features. This concept is analogous to techniques in audio-visual speech recognition where visual context is fused within the encoder, yielding improved performance[23]. By fusing context earlier, the model's hidden layers can co-adapt to both the speech and noise characteristics, potentially learning more robust representations. Exploring such _early-fusion_ or _mid-fusion_ strategies (as opposed to late-fusion) is likely to be fruitful, given that prior work has found them to enrich the combined representation and reduce over-reliance on the primary modality[23]

Advanced fusion methods: Even if the current transformer encoder for fusion did not improve accuracy, more sophisticated fusion architectures could be investigated. One potential direction is to use cross-modal attention or gating mechanisms that dynamically weight the contribution of noise context. For example, a gated fusion network might learn to _suppress_ the noise classifier's influence when the SNR is high (clean audio) and amplify it when the audio is very noisy. Alternatively, a _stacked_ fusion approach could be tried, where the outputs of the KWS model are iteratively refined by conditioning on context (allowing multiple back-and-forth interactions between the speech and noise representations). The transformer-based fusion module could also be expanded with additional layers or two-stream attention (one stream focusing on keyword features, another on noise features, with interactions)[23]. Another idea is to leverage knowledge distillation or auxiliary loss: for instance, train the fusion network to mimic the KWS output on clean data and improve on noisy data, thus explicitly guiding it to become equal or better than the baseline. In summary, developing more powerful fusion techniques – perhaps inspired by multimodal learning research – is an open avenue that could overcome the limitations of the simple late fusion used in this work.

Diverse noise and acoustic conditions: Finally, to ensure the system generalizes well, it should be evaluated and trained on a more diverse set of noise environments and real-world conditions. Future work should incorporate a broader noise dataset covering many different ambient sounds, beyond the few urban noise classes used here. This could include noises from household settings, nature, industrial environments, overlapping speech, music, etc. For example, leveraging large sound event collections like FSD50K (which contains over 50k clips across 200 sound classes) could provide a wealth of varied background audio[24] By exposing the model to a wide range of noise types and SNR variations, we can better assess its robustness and push its capabilities. Additionally, introducing time-varying noise (where the noise type or level changes within an utterance) and room acoustics effects (reverberation) would move the evaluation closer to real-world scenarios. A more exhaustive noise training regimen, possibly combined with the above suggestions (joint learning and advanced fusion), may unlock the performance gains that were not realized in the present study. The system could then be truly noise-adaptive, maintaining high accuracy across a spectrum of challenging acoustic settings.

In summary, while the current work did not observe accuracy improvements from the late fusion approach, it opens several pathways for future exploration. By pursuing joint training, earlier context integration, smarter fusion mechanisms, and testing on more varied noises, future research can build upon this modular architecture to achieve a more _noise-robust keyword spotting_ system. Each of these directions addresses a limitation identified in our results, and together they aim to fully capitalize on the rich information that environmental context can offer to speech recognition models. With these refinements, we anticipate that the goal of improving KWS performance through noise-awareness can be realized in subsequent studies.

REFERENCES

[1] Y. Huang, 'Research on the Development of Voice Assistants in the Era of Artificial Intelligence', *SHS Web Conf.*, vol. 155, p. 03019, 2023, doi: 10.1051/shsconf/202315503019.

[2] I. Lopez-Espejo, Z.-H. Tan, J. H. L. Hansen, and J. Jensen, 'Deep Spoken Keyword Spotting: An Overview', *IEEE Access*, vol. 10, pp. 4169–4199, 2022, doi: 10.1109/ACCESS.2021.3139508.

[3] Y. Zhuang, X. Chang, Y. Qian, and K. Yu, 'Unrestricted Vocabulary Keyword Spotting Using LSTM-CTC', in *Interspeech 2016*, ISCA, Sep. 2016, pp. 938–942. doi: 10.21437/Interspeech.2016-753.

[4] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, 'Continuous hidden Markov modeling for speaker-independent word spotting', in *International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, UK: IEEE, 1989, pp. 627–630. doi: 10.1109/ICASSP.1989.266505.

[5] A. Berg, M. O'Connor, and M. T. Cruz, 'Keyword Transformer: A Self-Attention Model for Keyword Spotting', in *Interspeech 2021*, Aug. 2021, pp. 4249–4253. doi: 10.21437/Interspeech.2021-1286.

[6] K. R. Prajwal, L. Momeni, T. Afouras, and A. Zisserman, 'Visual Keyword Spotting with Attention', Oct. 29, 2021, *arXiv*: arXiv:2110.15957. Accessed: Sep. 28, 2023. [Online]. Available: http://arxiv.org/abs/2110.15957

[7] A. Mehrish, N. Majumder, R. Bhardwaj, R. Mihalcea, and S. Poria, 'A Review of Deep Learning Techniques for Speech Processing', May 30, 2023, *arXiv*: arXiv:2305.00359. doi: 10.48550/arXiv.2305.00359.

[8] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, 'Trainable Frontend For Robust and Far-Field Keyword Spotting', Jul. 19, 2016, *arXiv*: arXiv:1607.05666. doi: 10.48550/arXiv.1607.05666.

[9] P. Warden, 'Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition', Apr. 09, 2018, *arXiv*: arXiv:1804.03209. doi: 10.48550/arXiv.1804.03209.

[10] S. Quintas, I. Ferrané, and T. Pellegrini, 'Enhancing Synthetic Training Data for Speech Commands: From ASR-Based Filtering to Domain Adaptation in SSL Latent Space', Sep. 19, 2024, *arXiv*: arXiv:2409.12745. doi: 10.48550/arXiv.2409.12745.

[11] J. Salamon, C. Jacoby, and J. P. Bello, 'A Dataset and Taxonomy for Urban Sound Research', in *Proceedings of the 22nd ACM international conference on Multimedia*, Orlando Florida USA: ACM, Nov. 2014, pp. 1041–1044. doi: 10.1145/2647868.2655045.

[12] 'Freesound'. Accessed: May 04, 2025. [Online]. Available: https://freesound.org/

[13] A. Vaswani *et al.*, 'Attention Is All You Need', Aug. 02, 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[14] A. Dosovitskiy *et al.*, 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale', Jun. 03, 2021, *arXiv*: arXiv:2010.11929. doi: 10.48550/arXiv.2010.11929.

[15] D. S. Park *et al.*, 'SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition', in *Interspeech 2019*, Sep. 2019, pp. 2613–2617. doi: 10.21437/Interspeech.2019-2680.

[16] *Keyword Transformer*. (Oct. 04, 2021). [Online]. Available: https://github.com/ARM-software/keyword-transformer

[17] S. Abdoli, P. Cardinal, and A. Lameiras Koerich, ‘End-to-end environmental sound classification using a 1D convolutional neural network’, *Expert Systems with Applications*, vol. 136, pp. 252–263, Dec. 2019, doi: 10.1016/j.eswa.2019.06.040.

[18] Y. Tokozume and T. Harada, ‘Learning environmental sounds with end-to-end convolutional neural network’, in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA: IEEE, Mar. 2017, pp. 2721–2725. doi: 10.1109/ICASSP.2017.7952651.

[19] Y. Ephraim and D. Malah, ‘Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator’, *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 6, pp. 1109–1121, Dec. 1984, doi: 10.1109/TASSP.1984.1164453.

[20] R. Martin, ‘Noise power spectral density estimation based on optimal smoothing and minimum statistics’, *IEEE Trans. Speech Audio Process.*, vol. 9, no. 5, pp. 504–512, Jul. 2001, doi: 10.1109/89.928915.

[21] H. Li, D. Wang, X. Zhang, and G. Gao, ‘Frame-Level Signal-to-Noise Ratio Estimation Using Deep Learning’, in *Interspeech 2020*, ISCA, Oct. 2020, pp. 4626–4630. doi: 10.21437/Interspeech.2020-2475.

[22] M. Ryu, J.-W. Kim, M. Oh, S. Lee, and H. Park, ‘Noise-Agnostic Multitask Whisper Training for Reducing False Alarm Errors in Call-for-Help Detection’, Jan. 20, 2025, *arXiv*: arXiv:2501.11631. doi: 10.48550/arXiv.2501.11631.

[23] L. Wei, J. Zhang, J. Hou, and L. Dai, ‘Attentive Fusion Enhanced Audio-Visual Encoding for Transformer Based Robust Speech Recognition’, Aug. 06, 2020, *arXiv*: arXiv:2008.02686. doi: 10.48550/arXiv.2008.02686.

[24] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, ‘FSD50K: An Open Dataset of Human-Labeled Sound Events’, Apr. 23, 2022, *arXiv*: arXiv:2010.00475. doi: 10.48550/arXiv.2010.00475.