

Master's Thesis

Professorship for Audio Information Processing
Technische Universität München
Univ.-Prof. Dr.-Ing. Bernhard U. Seeber

MP3 Compression as a Means to Improve Robustness against Adversarial Noise Targeting Attention-based End-to-End Speech Recognition

B.Eng. Iustina Andronic

Advisor: Dipl.-Ing.(Univ.) Ludwig Kürzinger
Supervisor: Univ.-Prof. Dr.-Ing. Bernhard U. Seeber

Started on: 14.10.2019
Submitted on: 14.04.2020

Abstract

Adversarial Examples represent an imminent security threat to any Machine Learning system. The present thesis addresses this issue by proposing MP3-compression as a potential measure to reduce the susceptibility of Automatic Speech Recognition (ASR) systems to be mislead by Audio Adversarial Examples (AAEs). In essence, we used the Fast Gradient Sign Method (FGSM) to generate untargeted AAEs in the form of adversarial noise added to original speech samples. We used a feature inversion procedure to convert the adversarial examples from the feature into the audio domain. Different from prior work, we targeted an end-to-end, fully neural ASR system (namely ESPnet) featuring a hybrid decoder enhanced with both Connectionist Temporal Classification (CTC) and Attention mechanisms. We found that MP3 compression applied to adversarial examples indeed reduces the recognition errors when compared to raw, uncompressed adversarial inputs. This result was validated by experiments with four ASR models trained on four types of audio data (uncompressed .wav format, as well as MP3 formats at three compression bitrates - 128, 64 and 24 kbps). Additionally, when we decoded compressed adversarial examples originating from a different audio format than the training data, in a train-test mismatch scenario, we observed a further alleviation in the error rates. In a parallel series of decoding experiments, we found that MP3 compression applied to speech inputs augmented with *non-adversarial noise* triggers an opposite behaviour of the ASR systems, in which more transcription errors are achieved than for uncompressed noise-augmented inputs. This finding consolidates the previous ones by suggesting that MP3 encoding is effective in diminishing only the adversarial noise. Finally, a statistical test performed on the estimated Signal-to-Noise Ratio (SNR) of adversarial inputs confirmed that MP3-compressed adversarial samples had higher SNRs (hence less adversarial noise) than uncompressed adversarial inputs.

Statement of Autorship

I, Iustina Andronic, hereby declare that I am the sole author of this Thesis, which is a requirement for completing the *Master of Science in Neuroengineering* at the Technical University of Munich. I have not used any sources other than those listed in the bibliography and identified as references. I further declare that this Thesis was not submitted to another examination board and was not published elsewhere.

Munich, 14.04.2020

Acknowledgements

This thesis represents the last milestone for completing the Master's Degree in Neuroengineering, and I am grateful to all the people that have turned this challenging journey into a highly valuable and ultimately enjoyable experience.

In the first place, I am utterly grateful to Univ.-Prof. Dr.-Ing. Bernhard U. Seeber for his overall supervision, technical expertise and elevated academic standards, which have prompted me to deliver meaningful scientific output. At the same time, I am thankful for his support on the personal level, in the form of honest career advice and concrete help with my scholarship extension.

I would also like to express my gratitude to Dipl.-Ing. (Univ.) Ludwig Kürzinger, for his constant guidance throughout the entire six months, from topic proposal to manuscript proofreading and refinement, for his genuine and inspiring enthusiasm towards the topic, his readiness to answer my questions, his organized and result-oriented mindset that have thoroughly shaped the way in which I approached this project.

I am sincerely indebted to Ricardo, for his openness, patience and goodwill to clarify the code and ESPnet framework since day one, for the thought-provoking discussions we had in the lab and his always-present good spirits.

I would next like to express profound appreciation to my MSNE colleagues from batch '17-'18, especially Tanya, Viktorija, Sai Lam and Francisco, for their support and friendship, their contagious passion for science, their colourful personalities and the culturally-enriching exchange we went through.

My deepest thanks go to my fiancée, Teodor, for his loving support and morale-boosting advice, for teaching me a more rational approach of life, and in what concerns this Thesis, for his great help with Latex editing.

I also feel extremely grateful to my parents, Emanuel and Daniela, to my sister Andra-Dumitrița, to my extended family and all my friends, for their ongoing encouragement to overcome tough moments and for their trust in my abilities and decisions.

Last but not least, I thank DAAD (Deutscher Akademischer Austauschdienst) for providing the financial security to complete this Master's Degree without additional worries.

Finally, I am entirely thankful to God for blessing me with such wonderful people and opportunities, and for endowing me with the strength to accomplish a remarkable journey, full of learning and self-discovery.

In the honour of my beloved (maternal) grandmother.

Table of Contents

List of Abbreviations

AAE(s)	=	Audio Adversarial Example(s)
ASR	=	Automatic Speech Recognition
CER	=	Character Error Rate
CTC	=	Connectionist Temporal Classification
DFT	=	Discrete Fourier Transform
DNN	=	Deep Neural Network
EOS	=	End of Sequence
ESPnet	=	End-to-End Speech Processing Toolkit (Watanabe et al., 2018)
FGSM	=	Fast Gradient Sign Method
GMM	=	Gaussian Mixture Model
HMM	=	Hidden Markov Model
LM	=	Language Model
LSTM	=	Long Short-Term Memory
MFCC(s)	=	Mel Frequency Cepstral Coefficient(s)
ML	=	Machine Learning
MP3	=	MPEG-1/2 Audio Layer III
RNN	=	Recurrent Neural Network
SNR	=	Signal-to-Noise Ratio
STFT	=	Short-Time Fourier Transform
WER	=	Word Error Rate

List of Figures

List of Tables

1

Introduction

1.1 Motivation and Problem Statement

In our increasingly digitized world, vocal assistants have easily found their place in many users' lives. They are deployed in a wide variety of applications, such as smart-home systems, conversational devices, humanoid companionship robots, shopping assistants, call-center answering machines, voice-based search algorithms, videos' subtitle captioning, real-time dictation and translation software, speech transcription devices for the deaf, patient monitoring systems - to name just a few. Regardless of their extensive deployment, there is one feature that all speech recognition systems have in common: they can seemingly understand any vocal command as if they were another human being. What made them highly appealing and popular is their ease of operation and interactivity. These systems have instantiated a new and engaging way of relating with technology, making it possible for humans to directly communicate, issue commands and even chat with computers by means of vocal interaction, which comes very natural to the users. We are thus slowly transitioning from the epoch of everything being *at the touch of a button* or *one click away*, to everything being *a vocal command away*.

Yet as is the case with any digital technology of major impact and usability, vocal assistants are prone to imminent security threats. In particular, Automatic Speech Recognition (ASR) systems can be *hacked* into recognizing hidden voice commands delivered on purpose by a malicious external agent. In technical vocabulary, these are termed *audio adversarial examples* - they represent voice commands seemingly innocuous to a human listener, but which carry along a hidden message that can be decoded accordingly by the ASR system, and even passed on to a vocally-triggered execution system.

For a better exemplification, let us consider the scenario illustrated in Fig. 1.1 and suppose that a user wants to shop on Amazon by means of vocal commands with a smart-home assistant, while the radio/TV is turned on. We also assume that an adversarial tune which includes a *hidden* deceptive command (e.g., *Activate backdoor* or *Transfer X amount of money to the account Y*) embedded in the original music is played over the radio/TV. Depending on the amount of environmental noise and other nearby sound sources, this command could be picked up and executed by a smart-home assistant without the user taking notice. This is nothing but an extreme example of an *over-the-air, targeted* adversarial attack, in which the attacker wants the system to recognize a *specific* adversarial and malicious phrase.

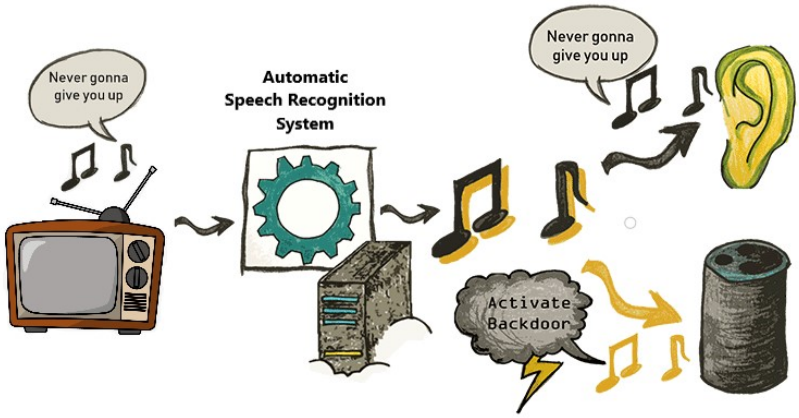


Figure 1.1: An empirical adversarial attack against a commercial ASR system (adapted from https://adversarial-attacks.net/)

Luckily, such vocal attacks have not been documented outside the research community so far. The mere harmful potential of adversarial attacks is enough to drive major research efforts into extensively studying their nature and characteristics, and to devise appropriate detection and defense methods, with the goal of improving the robustness of ASR systems.

1.2 Contribution

This thesis addresses the above-stated security threat by proposing MP3-compression as a potential measure to reduce ASR systems' susceptibility to be mislead by adversarial examples. We eventually aim for *more robust ASR systems* against adversarial noise.

In essence, we generated *untargeted* adversarial examples in the form of *adversarial noise* added to original speech samples. We used a feature inversion procedure to convert the adversarial examples from the feature to the audio domain. Different from prior work, we targeted an end-to-end, fully neural ASR system, namely ESPnet, featuring a *hybrid* decoder enhanced with both Connectionist Temporal Classification (CTC) and Attention mechanisms. Notably, this work did not focus on over-the-air, real-time attacks, but rather on direct attacks, in which adversarial samples were digitally presented at the system's input.

We found that MP3 compression applied to adversarial examples indeed reduces the recognition errors when compared to raw, uncompressed adversarial inputs. This result was validated by experiments with four ASR models trained on four types of audio data (uncompressed *.wav* format, as well as MP3 formats at three compression bitrates). Additionally, when we decoded compressed adversarial examples originating from a different audio format than the training data, in a train-test mismatch setting, we observed a further alleviation in the error rates. In a parallel set of decoding experiments, we found that MP3 compression applied to speech inputs augmented with *non-adversarial noise* triggers an opposite behaviour of the ASR systems, in which more transcription

errors are achieved than for uncompressed noise-augmented inputs. This finding consolidates the previous ones by suggesting that MP3 encoding is effective in diminishing only the adversarial noise. Finally, a statistical test performed on the estimated SNR of adversarial inputs confirmed that MP3-compressed adversarial samples had higher SNRs (hence less adversarial noise) than uncompressed adversarial inputs.

Thesis structure

The thesis is organized in six main chapters. The first (present one) has staged the context of this research, while the second overviews literature works that created Audio Adversarial Examples (AAEs) and proposed several defense tactics. The third chapter dives into the technicalities of end-to-end speech recognition systems, the Fast Gradient Sign Method (FGSM) for creating adversarial noise, and MP3 compression. The fourth chapter describes the complete experimental framework, the fifth presents and discusses the results, while the last one draws several conclusions, outlines this work's limitations, as well as future research directions.

2

Overview of Audio Adversarial Examples (AAE)

Deep Neural Networks (DNNs) have become nowadays the method of choice when it comes to any pattern recognition task. In the field of ASR in particular, they are highly appealing due to their ability to model the acoustics and linguistics of any kind of language and to perform speaker-independent and noise-robust speech recognition. Many present-day marketed devices (e.g., Google Home, Amazon Echo) and their accompanying voice assistant software (e.g., Cortana, Siri, Alexa) function by means of a deep network architecture. However, their success also comes at a cost - a great number of network parameters, sometimes on the order of millions, that need to be optimized. This is not that much of a problem with the availability of powerful computers, but it opens up a generous playground for possible attackers that purposefully target the vulnerabilities of these systems. In response to these security concerns, the research field is very active into staging a wide range of attack scenarios and researching appropriate defense schemes.

Adversarial examples constitute one of the security threats that is intensely researched, being also the object-of-study of this thesis. In a general sense, adversarial examples represent inputs that have been specifically designed by an adversary to cause a Machine Learning (ML) algorithm to produce a misclassification (Qin et al., 2019). Initial work on adversarial examples focused mainly on the domain of image classification. Perhaps the most famous example is the one from Goodfellow et al. (2015), who demonstrated that an adversarial image can be classified by the neural network as a *gibbon*, although to the human eye, it clearly looks like a *panda*.

Adversarial examples are usually created by adding a carefully crafted noise to an original input. However, the exact method differs from domain to domain due to their particularities. For example, in ASR, adversarial examples are harder to construct because speech is a temporal signal and is processed in a sequential manner, unlike images, which are static and exhibit spatial instead of temporal dependencies. Furthermore, visual adversarial examples tend to be more imperceptible to a human observer than Audio Adversarial Examples (AAEs). This in turn makes visual adversarial examples more effective over-the-air, i.e., in a physical, real-world environment (Qin et al., 2019). In contrast, the physical characteristics of the environment seem to interfere to a greater extent with adversarial examples in the audio domain, making the creation of robust, over-the-air AAEs a more complicated task, in which room characteristics and

background noises need to be accounted for.

Yet regardless of the target domain, the threat model and attack scenario can take different forms, depending on the adversary's background, knowledge, and scope (Hu et al., 2019). Two types of attacks distinguish themselves based on the adversary's knowledge of the target ASR system:

- The *white-box attack* assumes that the adversary has full knowledge of the target neural network model, including the model type, model architecture, and values of all the parameters and training weights. This gives him the privilege to compute gradients through the model in order to generate the adversarial examples.
- The *black-box attack* assumes that the adversary has no access to the internal characteristics of the target neural networks, and acts as a normal user who can only observe the output given by the model. In this more realistic scenario, adversarial examples are created just by monitoring the system's output after iteratively and systematically modifying the input.

Besides, the adversary's purpose and intentions render two further types of attacks:

- The *non-targeted attack* aims to cause the neural networks to predict *any arbitrary incorrect transcription* for the adversarial example.
- The *targeted attack* aims to induce the neural networks to assign *a specific, desired transcription* for the adversarial example. Such an attack is dangerous since it could force an ASR system to execute the specified malicious commands.

All in all, adversarial examples are a property of any deeper neural network. Their mere existence indicates that there are *blind spots* in the input space of deep learning models, and this is believed to be a consequence of the models being too linear, rather than nonlinear, as Goodfellow et al. (2014) postulate. The misleading capability of adversarial examples further implies that the models are *unsmooth*, since a tiny perturbation in input space can trigger a drastic change in the output space (Sun et al., 2019).

2.1 Prior Work on AAE Generation

Although the topic has come into focus roughly five-six years ago, there is already an impressive corpus of literature that tackles the creation of AAEs. Most of the works have set about to pursue one of the two major goals, sometimes succeeding in both: that their AAEs work over-the-air, and that they are less obviously perceptible. This section conveys a brief overview of the most outstanding papers that generated AAEs.

Carlini et al. (2016) were among the first to tackle the security issues of voice interfaces, and introduced the so-called *hidden voice commands*, demonstrating that targeted attacks against archetypal ASR systems, solely based on Hidden Markov Models (HMM), are feasible. They used inverse feature extraction to create *obfuscated* adversarial audio samples, which sounded like highly distorted speech hidden within noise and were

hardly understandable by a human. However, these first attempts had an unnatural robotic tone that could still bring suspicions to the thoughtful listeners. Interestingly, their examples also represent one of the first documented successful attacks in an over-the-air setting, targeting the Google Assistant on a mobile phone. Nevertheless, in their approach, they had to completely synthesize new adversarial audio, while most of the modern AAEs are created starting from a legitimate, non-adversarial input audio.

With a different approach, Iter et al. (2017) used the WaveNet pre-trained model to construct adversarial mel-frequency cepstral coefficient (MFCC) features, which they subsequently inverted back to the audio domain to get the adversarial audio. Their adversarial perturbations were imperceptible, but did not work over-the-air.

Zhang et al. (2017) proposed the ingenious DolphinAttacks, showing that it was even possible to completely hide a transcription by leveraging nonlinearities of microphones to modulate the baseband audio signal with ultrasound higher than 20 kHz. However, the modulation was highly microphone-dependent, so their attack was restricted to work only on the microphone for which it was optimized.

While previous works mostly targeted short adversarial phrases, Carlini and Wagner (2018) constructed adversarial perturbations for designated longer sentences. Similar to previous adversarial attacks on image classifiers, Carlini's novel attack was achieved with a gradient-descent-based minimization, but the loss function was replaced with the novel CTC-loss, which is optimal for time sequences. On the downside, the constraint for minimizing the added adversarial noise did not consider the limits and sensitivities of human auditory perception. Hence, although the introduced perturbations were quiet, they could still be noticed by a human.

In a parallel line of work, there were successful attempts to hide transcripts within more complex types of audio signals, like music (Yuan et al., 2018; Yakura and Sakuma, 2019; Schönherr et al., 2018) or birdsong recordings (Schönherr et al., 2018). Yuan et al. (2018) described CommanderSong, which was able to hide transcripts within music and was effective over-the-air. However, their approach was only shown to be successful in music, and did not include noise reduction based on human perception, nor was it independent from speakers and recording devices, as the attack parameters had to be specially adjusted for these components.

Schönherr et al. (2018) were the first to develop imperceptible attacks by leveraging human psychoacoustics. Specifically, they applied a *psychoacoustic hiding* model that manipulates the acoustic signal below the thresholds of human perception. They attacked the Kaldi ASR system, which is partially based on neural networks but also uses some traditional components such as a HMMs instead of an RNN for final classification. In a subsequent publication (Schönherr et al., 2019), they enhanced their implementation and produced *generic* adversarial examples, which remained *robust* in over-the-air attacks. To this end, they used room impulse responses (RIRs) to simulate arbitrary room characteristics and included them in the optimization process, thus obtaining adversarial examples that are transferable across a wide range of rooms.

In a concurrent work, Qin et al. (2019) successfully constructed imperceptible AAEs (verified by a human study) based on the psychoacoustic principle of auditory masking, while retaining 100% targeted success rate on arbitrary full-sentence targets. Simul-

taneously, they also make progress towards developing robust adversarial examples by constructing perturbations which remain effective even after applying realistic, simulated environmental distortions. The difference to Schönherr et al. (2019) is that they target a fully end-to-end ASR system.

2.2 Prior Work on AAE Defenses

Defending against adversarial examples has been primarily investigated in the image domain. According to the review of Hu et al. (2019), the existing defense strategies for image adversarial examples can be grossly grouped in two categories: (1) *proactive*, aiming to enhance the robustness during the training procedure of the recognition models themselves, and (2) *reactive*, aiming to detect the existence of adversarial examples after the neural networks are trained.

The former approach usually includes network training with adversarial examples, as well as network distillation, which considers the output of a pre-trained network as soft labels, thereby using them as targets to train a second network (Papernot et al., 2016). The latter approach makes use of input transformations (e.g., JPEG compression, cropping, resizing), with the goal to recover the genuine transcription. In parallel, it explores adversarial detection methods, which are meant to declare a given input as adversarial or benign. The defensive measures on images inspired several approaches to combat the adversarial examples in the sound domain, some of which are reviewed below.

Yang et al. (2018) applied some primitive signal processing transformations on audio data to disrupt the adversarial perturbations without decreasing their quality too much. Specifically, they tested local smoothing (i.e., replacing each audio sample with the average value over a surrounding window), down-sampling, and quantization as easy-to-implement and fast-to-operate input transformations. The outcome was that the pre-processing defense was more effective for shorter-length adversarial examples.

Rajaratnam et al. (2018) likewise explored the effectiveness of audio pre-processing methods such as band-pass filtering, compression (both MP3 and AAC - Advanced Audio Coding), but also ventured to complex speech coding algorithms (Speex, Opus) so as to mitigate the audio adversarial attacks. They experimented with a keyword-spotting system, optimized for recognizing simple, isolated words like *yes, no, up, down* etc., and showed that an *ensemble* strategy composed of both speech coding and other form of pre-processing (e.g., compression) makes for a stronger defense against AAEs, compared to isolated pre-processing defenses.

The authors of CommanderSong (Yuan et al., 2018) put forward two methods for *detecting* an adversarial input. The first is based on their observation that noise from the background or a speaker decreased the success rate of adversarial examples while having little impact on the identification of the original, non-adversarial audio commands. Therefore, they suggested adding some *turbulent* noise to the input audio and checking if the perturbed input yields a different recognition result than the original one, which would mean that the respective input can be classified as an adversarial example. This

approach, however, could not be applied to over-the-air adversarial examples, for which the background noise has already been modelled during their generation. Their second proposal considered down-sampling: if the ASR system outputs different results for a certain input and its down-sampled version, then the input has a high probability to be an adversarial example. Experiments meant to evaluate the effectiveness of this second approach showed it was successful for both directly-presented, as well as over-the-air AAEs.

Sun et al. (2019) proposed the training of general noise-robust acoustic models with two novel methods: adversarial data augmentation and adversarial regularization (also called adversarial training). In adversarial data augmentation, they create an adversarial version of each mini-batch of data and update the network's parameters with the original mini-batch augmented with its adversarial version. In contrast, adversarial regularization means that the loss function optimizing the network is *per se* amended by the loss on adversarial examples, in addition to the loss on the original data. In other words, adversarial examples are explicitly introduced into the loss function, which makes the model more robust to minor deviations from the original training data. They validated their scheme with experiments on Chinese speech datasets and observed that the proposed strategies improved the robustness of an end-to-end ASR model to datasets augmented with Musan artificial noise, compared to the model's performance for the baseline (clean) dataset. However, they did not test if these approaches were robust against the adversarial noise itself.

As shown in the previous section, Schönherr et al. (2018) created their AAEs with a psychoacoustic model that introduced the adversarial noise below the calculated frequency masking thresholds, which is the *sweet spot* for the hidden transcriptions. According to the authors, the exact same principle could be used conversely, so as to defend a system against hidden input. As such, they hypothesize that a perceptual based audio encoder like MP3 could remove exactly those inaudible ranges in the audio, where the adversarial perturbation lies. MP3 is supposed to clean up the audio files before they reach the speech recognition system and ideally, the system should be trained on MP3 data beforehand, so that it can learn its particularities. This will ensure that recognition performance will not suffer when decoding non-adversarial input that is also MP3-compressed. The authors speculate that this strategy will not circumvent blind spots of DNNs completely, but will force the attacker to move their secret messages into the human hearing range so as to still *trick* the system. But this could be less dangerous since audible AAEs could easily raise the users' suspicions, making them aware of the uncovered transmitted message. Notably, the authors included the MP3 scenario in their *future work* section, but to date they did not publish any experimental work to verify it.

Consequently, we borrow this idea and exploit it in the present thesis. Our research question could be formulated as follows: to what extent can MP3 compression aid in removing the adversarial noise and thus recover the benign character of the input (which should transcribe to the original, non-adversarial phrase)?

3

Technical Background

This chapter introduces the major technical concepts that form the backbone of the present thesis: the hybrid End-to-End Automatic Speech Recognition paradigm, along with two of its main architectures - Connectionist Temporal Classification (CTC) and Attention, the Fast Gradient Sign Method (FGSM) for creating adversarial examples, and the MP3 compression technique.

3.1 End-to-End ASR

Automatic Speech Recognition can generally be regarded as a sequence-to-sequence problem: the system has to learn how to generate an output sequence of words Y, called transcription, from an input sequence of acoustic features X extracted from speech. ASR models are intrinsically probabilistic, aiming to compute the posterior distribution $p(Y|X)$, equivalent to the most likely word sequence Y, given the speech feature sequence X.

The earliest-emerged ASR systems are quite complicated in achieving this task, because they consist of various separate modules such as acoustic, lexicon, and language models. More modern systems are based on a hybrid architecture composed of a Hidden Markov Model (HMM) and a Deep Neural Network (DNN), which still requires a multi-stage training procedure to make the components work together (Chorowski et al., 2015).

Yet this approach presents several inconveniences, as pointed out by Watanabe et al. (2017). Firstly, step-wise refinement is needed to build accurate modules. For instance, in creating the acoustic models from scratch, one would first need to estimate the HMMs and Gaussian Mixture Models (GMMs), in order to obtain the state-dependent HMM structure and phonetic alignments, before the DNN is trained. Secondly, linguistic information is quite problematic, since it is taken from handcrafted pronunciation dictionaries, which are defined by linguistic rules and are thus prone to human error, which a fully data-driven system could avoid. Thirdly, the hybrid HMM-DNN ASR systems often use *conditional independence assumptions* (especially Markov assumptions) between output tokens, while this is not necessarily true for real-world data, where pronunciation of a particular phoneme is heavily influenced by the phonetic context (e.g., the phoneme *e* is pronounced differently in the words *lead* and *red*). The conditional independence assumptions can thus lead to high recognition errors, so they should better be avoided. Another problem is that inference (or decoding) is a complex process, which has to be

performed by integrating all modules. Finite state transducers were meant to achieve this, but the implementation of well-optimized transducers is quite complicated. Finally, all the consisting modules are optimized separately with different objective functions, which can result in optimization incoherence. This happens because separate optimization is oblivious to the desired joint-functionality of all the modules.

End-to-end ASR, on the contrary, aims to simplify this module-based approach into a single-network architecture within a deep learning framework, which directly estimates $p(Y|X)$. In an end-to-end model, the multiple modules are merged in one deep network for joint training. This network realizes the actual mapping of acoustic signals into output label sequences without the carefully-designed intermediate states used previously in HMM-based models. The joint training achieves globally optimal results as it uses an objective function that is highly relevant to the final evaluation criteria. Another appealing feature of end-to-end models is that they use soft alignments, allowing each audio frame to be mapped to all possible output tokens with a certain probability distribution, which does not require a forced, explicit correspondence (Wang et al., 2019).

A typical end-to-end speech recognition model would include the following parts: an encoder, an aligner and a decoder. In short, the encoder maps the input speech feature sequence into a hidden representation, the alignment block realizes the alignments between input and output sequences, while the decoder takes in the encoder's hidden representation and decodes it to the final transcription. The encoder and decoder networks are typically implemented with the same architecture, often made up of Recurrent Neural Networks (RNNs).

There are two major types of end-to-end architectures for ASR: (1) Connectionist Temporal Classification and (2) Attention-based Encoder-Decoder systems, both of which are thoroughly presented in the following sub-sections.

3.1.1 Connectionist Temporal Classification (CTC)

CTC is a method to get around not knowing the alignment between the input and the output sequences, hence it is termed *alignment-free*. In essence, it uses Markov assumptions to efficiently solve sequential problems by dynamic programming, which computes all possible hard alignments by calculating different paths, then it achieves soft alignments by aggregating the hard alignments. CTC assumes that output labels are independent of each other when enumerating hard alignments. In the following, a detailed description of CTC functionality is presented, inspired from the excellent visual material of Hannun (2017) and the overview of Wang et al. (2019).

The *Alignment* concept

To get the probability of an output given an input, CTC works by summing over the probability of all possible alignments between the two sequences. But before diving deeper into the technicalities of CTC, the concept of *alignment* has to be clarified.

An **alignment** represents the mapping between the output tokens (e.g., characters, sub-word units or words) and the input acoustic frames. In other words, it tells us

which output token corresponds to each acoustic frame. Alignment is thus an important concept that has to be tackled by any type of ASR system.

When creating the alignments, it is useful to consider that the input can have frames of silence, with no corresponding output. Furthermore, some words can comprise sequentially repeated letters (e.g., *hel**l**o*), so we would like to have the same output token for several consecutive input frames, without collapsing them. To solve these issues, CTC introduces a *new token* to the set of allowed outputs - *the blank token (ϵ)*, which does not correspond to any acoustic information in the input and will simply be ignored in the output.

The alignments allowed by CTC have a few important properties. The valid alignments are those that have the same length as the input sequence and map to the final output sequence Y after merging identical subsequent tokens and discarding the ϵ tokens, as shown in Fig. 3.1. Furthermore, the allowed alignments between the input and output sequences are *monotonic*. This means that future frames of the output sequence cannot align to earlier frames of the input sequence. A third property is that the alignment of X to Y is *many-to-one*: one or more input elements can align to a single output element but not vice-versa. This in turn implies the last property, that the length of Y cannot be greater than the length of X.

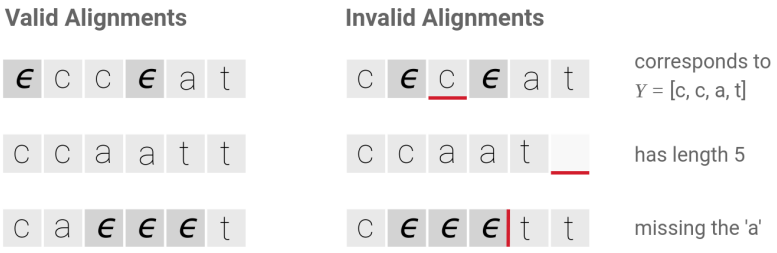


Figure 3.1: Valid vs invalid CTC alignments (Hannun, 2017)

CTC training

Concerning the CTC training approach, it can be summarized as follows: for a given input, the model is trained to maximize the probability it assigns to the right output. For this, the conditional probability $p(Y|X)$ needs to be efficiently computed and must also be differentiable, allowing the use of gradient backpropagation. Thus, the CTC-objective function optimized during training for a single (X, Y) pair is:

$$p(Y|X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t\left(a_t|X\right) \tag{3.1}$$

In plain words, the CTC conditional probability $p(Y|X)$ *marginalizes* (sums) over the set of valid alignments $A \in \mathcal{A}_{X,Y}$ (taken from the set of all possible alignments between X and Y), and computes the probability for a single alignment step-by-step. Models trained with CTC loss function typically use a recurrent neural network (RNN)

to estimate the per time-step probabilities, $p_t(a_t|X)$, i.e., the probability of each output token at each time step given the input sequence. RNNs usually work well because they can model the temporal dependencies and thus account for context in the input. The computation steps of the CTC loss function are depicted in fig 3.2.

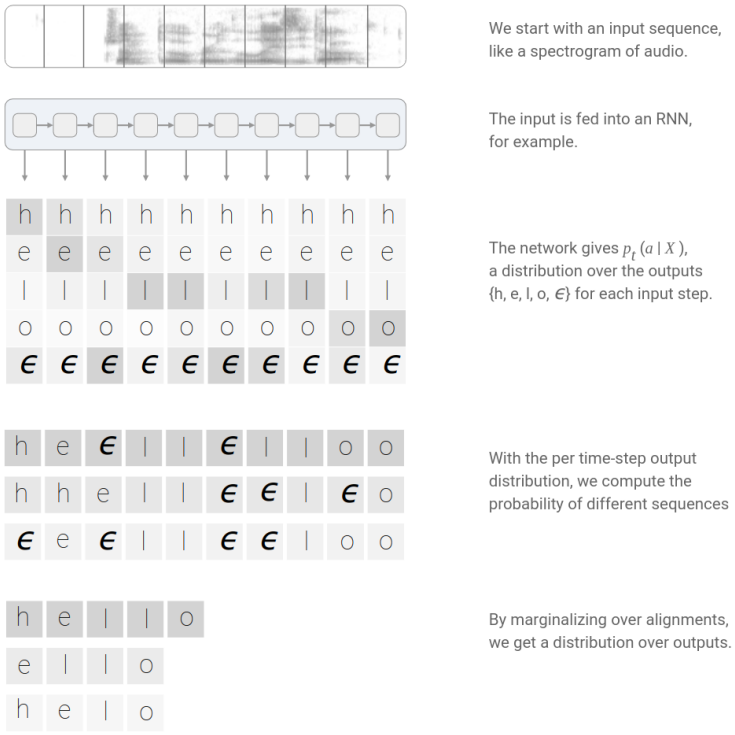


Figure 3.2: Computational steps of CTC-loss (objective function) (Hannun, 2017)

However, the CTC-loss is computationally expensive. A direct approach could be to compute the score for each alignment and sum them all up on-the-fly. But this can prove too slow for a vast number of possible alignments. Luckily though, the loss can be computed much faster by means of a dynamic programming algorithm, using the following trick: if two alignments collapse to the same output at the same step, they can be merged.

From eq. 3.1, it is clear that the CTC loss function is differentiable with respect to the per time-step output probabilities $p_t(a_t|X)$, since it is made up of their summation and multiplication. Thus, the gradient of the loss function with respect to the (unnormalized) output probabilities can be analytically computed, and from there, the backpropagation through the RNN's weights is achieved as usual.

CTC decoding

After training the model, it will be used for decoding new, unseen input. This means finding the most likely Y^* given an X, formalized by:

$$Y^* = \arg\max_{Y} \prod_{t=1}^{T} p_t(a_t \mid X) \tag{3.2}$$

A regular *beam search* algorithm is used for this purpose. It computes a new set of hypotheses at each input step by extending each hypothesis from the previous set with all possible output characters, and keeps only the top candidates, with the highest probabilities. In CTC decoding, the standard beam search is modified so that it can handle multiple alignments mapping to the same output. Thus, instead of keeping a list of alignments in the beam, only the output prefixes will be stored after collapsing sequentially repeated tokens and removing the ϵ characters. At each step of the search, the scores for a given prefix are accumulated based on all the alignments which map to it. This is illustrated in Fig. 3.3.

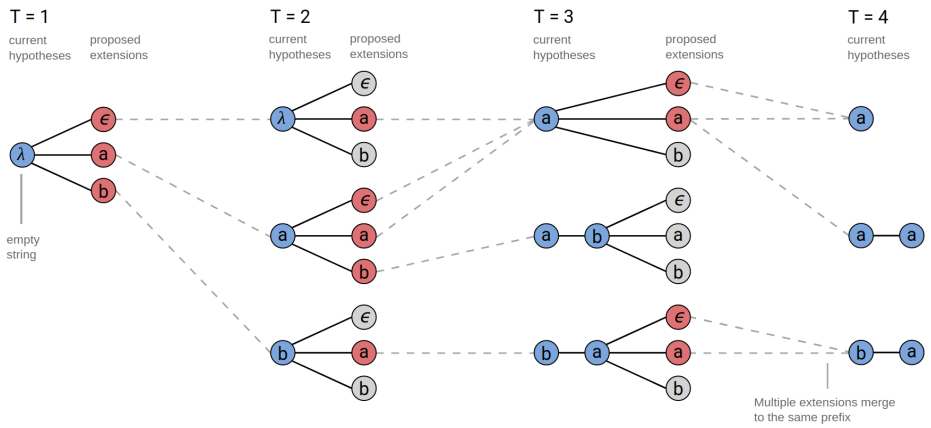


Figure 3.3: The CTC beam search algorithm with an output alphabet {ϵ, a, b} and a beam size of three (Hannun, 2017)

Although CTC is a powerful algorithm that overcomes the data alignment problem, it still has some shortcomings. One of them is the assumption that every output is conditionally independent of other outputs given the input, which is actually not valid for the sequence-to-sequence speech recognition problem, where phonetic context influences both previous and future utterances. This is also the reason why CTC cannot learn a Language Model (LM) over the output implicitly, in contrast to models which are conditionally dependent. A separate LM could be included in the decoding stage to boost the accuracy, however this would violate the end-to-end principle, considering that the LM needs to be separately trained. As such, the speech recognition network trained by CTC objective function should only be treated as an acoustic model. Finally, for scenarios where the output sequence is longer, CTC is powerless, because it can only map input sequences to output sequences that are shorter than it.

3.1.2 Attention-based Encoder-Decoder Models

Another possibility for end-to-end encoder-decoder systems is to enhance the decoder with an Attention mechanism. It would be interesting to first trace back the roots of the Attention concept.

The following concrete examples reveal a common approach in the way humans solve sequential tasks. When translating some text, we intuitively focus on the word we are presently translating, but at the same time we consider the context and other references of the current word in the original sentence. When we take notes or transcribe some audio recording, we listen carefully to the segment we are actively writing down, while trying to reference it to what was just being said some seconds/minutes ago. Similarly, if we are asked to describe the room we are in, we will *in turn* focus on different objects surrounding us. In a general sense, at every time step of the sequential task, we attend only to *some part* of the input information.

Neural networks can achieve the same behavior using an *Attention mechanism*, i.e., they can be trained to focus on a subset of the information they are given as input. For example, a decoder RNN can attend over the output of another RNN, representing the encoder (Olah and Carter, 2016). At every time step, it will focus on different positions in the encoder's hidden state sequence $\boldsymbol{h}$.

The issue with a conventional sequence-to-sequence model is that it has to boil down the entire input into a single, fixed-size hidden vector, which will only be fed *once* to the decoder network. Attention overcomes this by allowing the encoding RNN, which processes the input, to pass along information from each time step, while the decoding RNN generates the output by focusing on *certain* parts of the *entire* output sequence of the encoder, deemed relevant by the Attention mechanism. Thus, the Attention-based decoder takes in all the encoder's hidden states *after* the *entire input* was processed and creates the relevant context vector at each decoding time step. The step-by-step process is formalized below.

The encoder network is usually a Bidirectional Long Short Term Memory (BLSTM) RNN with a pyramidal structure, i.e., the outputs are sub-sampled so as to reduce the length of the final hidden vector length. Since the input speech signals can be hundreds of frames long, this sub-sampling is required to reduce the computational complexity (Chan et al., 2015). But conceptually, the encoder converts the input feature sequence X into a frame-wise hidden vector $\boldsymbol{h} = [h_1, h_2, \ldots, h_T]$:

$$\boldsymbol{h} = Encoder(X) \tag{3.3}$$

At each time step i, the Attention mechanism generates a *context vector* c_i as a weighted representation of *all* the encoder's hidden states. The context vector also carries information from the previous hidden state of the decoder, q_{i-1}:

$$c_i = Attention(\boldsymbol{h}, q_{i-1}) \tag{3.4}$$

By all means, Attention must also be differentiable, so that the network will *learn* where to focus. Thus, the following *trick* is implemented: the network will be allowed to focus everywhere, just to different extents.

There are different types of Attention, e.g., content-based Attention, location-aware Attention, hybrid Attention (Wang et al., 2019). The *location-aware Attention* is employed in this thesis and explained below, with notations borrowed from Watanabe et al. (2017).

Analogous to an alignment model in conventional ASR, the location-aware Attention attends the encoder's hidden values $\boldsymbol{h}$ in a recurrent manner (eq. 3.5), by carrying over information from the previous decoder state vector q_{i-1} and its previous Attention weights $\alpha_{i-1,t}$, in order to calculate the Attention weights $\alpha_{i,t}$ for the current i-th decoding step. Specifically, at each decoding timestep i, the *Attention* function will compute the scalar energy $e_{i,t}$, for each time step t of the encoder. The scalar energy is then converted into a probability distribution over the encoder's times steps using a softmax function, rendering the Attention weights $\alpha_{i,t}$ (eq. 3.7). The weights are then used to create the context vector c_i by linearly blending the encoder's hidden states from all its time steps t (eq. 3.8).

$$e_{i,t} = f(q_{i-1}, h_t, \alpha_{i-1,t}) = \boldsymbol{g}^\top \tanh(\mathrm{Lin}(q_{i-1}) + \mathrm{Lin}(h_t) + \mathrm{Lin}(f_t)) \tag{3.5}$$

$$\{f_t\}_{t=1}^{T} = \boldsymbol{K} * \alpha_{i-1,t} \tag{3.6}$$

$$\alpha_{i,t} = \mathrm{Softmax}(\{e_{i,t}\}_{t=1}^{T}) = \frac{\exp(e_{i,t})}{\sum_t \exp(e_{i,t})} \tag{3.7}$$

$$c_i = \sum_{t=1}^{T} \alpha_{i,t} h_t \tag{3.8}$$

In the above equations, $\boldsymbol{g}$ is a learnable vector that reduces the activations of the inner linear layers of the Attention network to a single scalar, tanh is the hyperbolic tangent activation function, Lin denotes linear layers with learnable matrix parameters, $*$ denotes the convolution operator and $\boldsymbol{K}$ the learnable convolution kernel (Kürzinger et al., 2019).

Finally, at every output step o_i, the RNN decoder produces a probability distribution over the next output label conditioned on all the previous outputs $o_{<i}$ (eq. 3.9). The distribution for the next output label o_i is a function of the current decoder state q_i, which in turn is a function of the previous state q_{i-1}, the previously emitted output label o_{i-1} and current context c_i (eq. 3.10). The recurrent decoding process runs until the special *End-of-Sequence (EOS)* output label is reached. The basic architecture of an encoder-decoder network with Attention is represented in Fig. 3.4.

$$P(o_i|X, o_{<i}) = \mathrm{Softmax}(\mathrm{Lin}(q_i)) \tag{3.9}$$

$$q_i = \mathrm{RNN}(q_{i-1}, o_{i-1}, c_i) \tag{3.10}$$

One downside of the temporal Attention mechanism in speech recognition is that it is too flexible, allowing extremely *non-monotonic* alignments (Watanabe et al., 2017). This may be well desired for applications like machine translation, where the input and output word orders are different (e.g., two adjacent words in the output translated sequence may correspond to locations far away in the original sequence, while two words that are

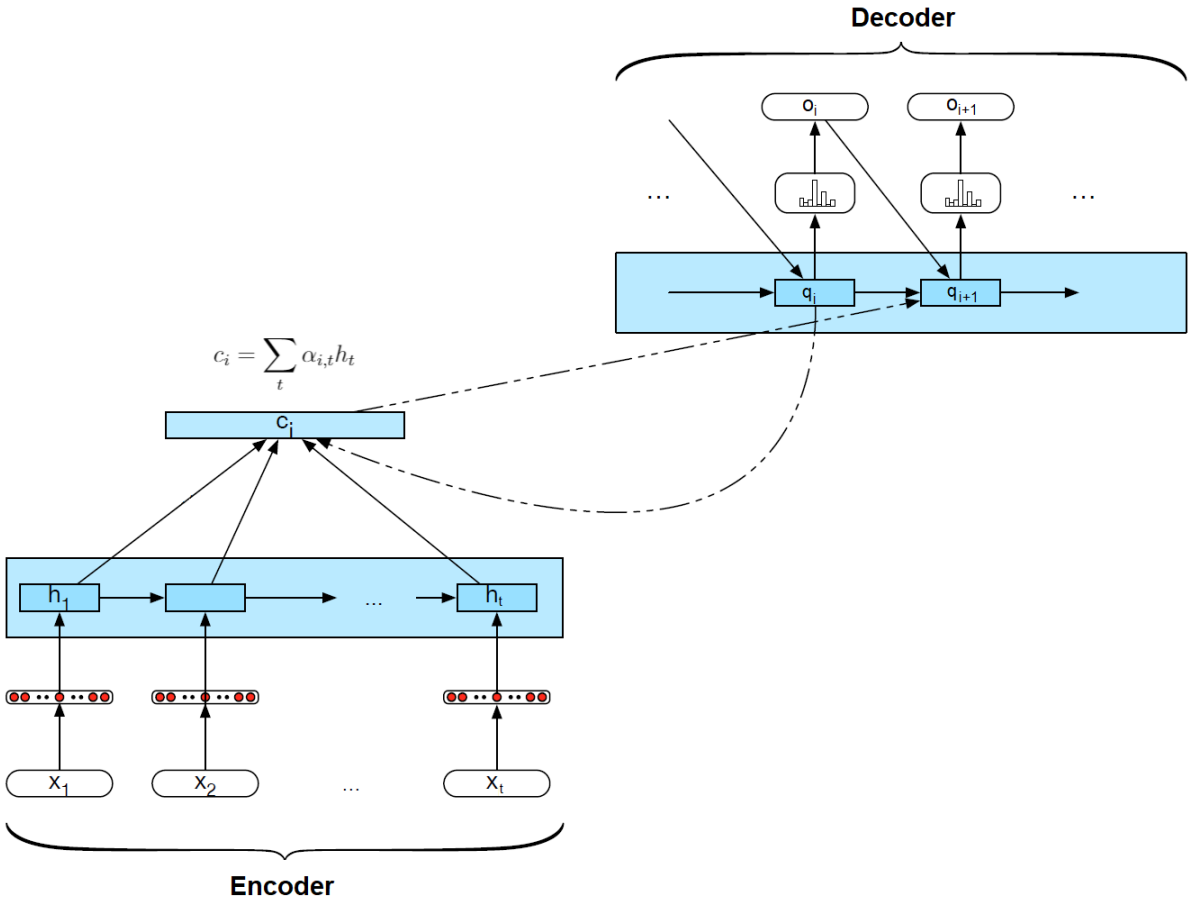


Figure 3.4: Encoder-decoder network with Attention. Computing the value for the decoder's current state q_{i+1} is based on the previous hidden state q_i, the previous output o_i, and the current context vector c_{i+1}. The context vector is derived from the Attention computation based on comparing the previous hidden state q_i to all of the encoder hidden states. Figure adapted from *Jurafsky and Martin (2013)*

far apart in the translated sequence may correspond to adjacent parts of the original sequence). However, in speech recognition, the feature inputs and corresponding letter outputs generally proceed in the same order. In particular, if one output label appears many times in different locations in the transcription, they should be matched to different parts of the speech input sequence. Nevertheless, using the Attention mechanism, they are likely to attend to the same part of the input speech. A viable solution for this would be to use CTC as a regularizer in a hybrid CTC-Attention architecture, as illustrated in the upcoming Section (3.1.3).

To sum up, Attention can be regarded as an auto-regressive neural network that can generate sequences. The next output character is computed as a function of all the input audio data, the previous decoder state and the previous output character decision. Notably, Attention-based models do away with conditional independence assumptions.

3.1.3 Hybrid CTC-Attention Models

Hybrid CTC-Attention models for speech recognition combine the advantages of both paradigms in both model training and inference. On the one hand, the Attention mechanism is data-driven and directly outputs a character sequence, while CTC imposes the necessary monotonic constraints on the alignments (Watanabe et al., 2017). These two concepts can be mathematically merged in a weighted fashion within the *multi-objective training function*:

$$Loss_{hybrid} = \alpha \log p_{CTC}(Y|X) + (1-\alpha) \log p_{Attention}(Y|X), \quad \alpha \in [0,1] \tag{3.11}$$

The parameter α controls how much of the CTC-objective is allowed. Networks trained with $\alpha = 0$ and $\alpha = 1$ consist of an Attention-only and CTC-only architecture, respectively, whereas networks trained with $\alpha \in (0,1)$ are designated as hybrid models.

In the inference (decoding) stage, the most probable letter sequence hypothesis Y^* is found via the *joint one-pass decoding beam search*. This is nothing but an extended version of the regular beam search (Fig. 3.3 from Section 3.1.1) that iteratively builds up a list of letter sequences, i.e., partial hypotheses Y ordered by their probability, until the EOS token is detected. The joint beam search has the particularity of combining both CTC and Attention posterior probabilities and optionally embed an LM (eq. 3.13). The LM can account for *out-of-vocabulary* word sequences, i.e., word pairs occurrences that are not present in the training set. The LM weight in decoding is parametrized by β. Notably, the λ parameter controls the amount of CTC in decoding, and can take a different value from the α used above for model training.

$$Y^* = \arg\max_{Y} p_{hybrid}(Y|X) \tag{3.12}$$

$$p_{Hybrid}(Y|X) = \lambda \log p_{CTC}(Y|X) + (1-\lambda) \log p_{Attention}(Y|X) + \beta p_{LM}(Y|X) \tag{3.13}$$

The effectiveness of the hybrid CTC-Attention end-to-end architecture was demonstrated, among others, by (Watanabe et al., 2017) on various ASR tasks (English, spontaneous Japanese and Mandarin Chinese), and by Kürzinger et al. (2019) on the English TEDLIUM-2 corpus. The recognition performance is comparable to the state-of-the-art conventional DNN-HMM systems, leading Watanabe and colleagues to develop the widely-adopted speech recognition framework called ESPnet (Watanabe et al., 2018). This toolkit was also used in the present thesis to implement the experiments described in Chapter 5.

3.2 Fast Gradient Sign Method (FGSM)

The threat model assumed in this thesis aims to create *untargeted* AAEs by means of the Fast Gradient Sign Method (FGSM), in a *white-box* scenario, where the network model and its parameters are fully known. This is in fact one of the most simplistic methods documented in the literature, initially developed for the image domain by Goodfellow et al. (2015), and subsequently adapted to the audio domain. It should be pointed

out that creating adversarial examples is just an intermediate step, and the focus of the thesis is not to develop highly optimized and powerful adversarial examples, but to explore ways to improve ASR models' robustness to general adversarial noise. This section formalizes the definition of AAEs and the FGSM method utilized in creating them.

Firstly, we must ascertain that a neural network is a parameterized function, $f(x, \theta)$, where x is the input (multi-dimensional feature vector) and θ represents the model parameters, which are iteratively optimized during the training stage. The trained model $f(x, \theta)$, with optimal, fixed-parameters, will then be used in the inference stage to predict the output sequence y corresponding to the input x. In this context, for an adversarial input $\hat{x}$, the following relations hold:

$$\hat{x} = x + \delta \tag{3.14}$$

$$\|\delta\| \ll \|x\| \tag{3.15}$$

$$y \neq f(\hat{x}, \theta) \tag{3.16}$$

δ is called the adversarial perturbation (noise) and is added to the original input x in order to trigger the network to output a wrong transcription, different from the ground truth y. Ideally, this perturbation should be much lower in amplitude than the original input x, so as to keep it unnoticeable to a human listener, but at the same time just high enough to push the system into misclassification. In this thesis, δ is computed via a gradient-based approach, namely the FGSM method, proposed by Goodfellow et al. (2015):

$$\delta_{\text{FGSM}} = \epsilon \operatorname{sgn}(\nabla_x J(x, y, \theta)) \tag{3.17}$$

$J(x, y, \theta)$ is the so-called loss or objective function, which serves as the cost measure between the network's output and the true label y for the input x. The model optimization procedure aims to minimize this cost, with methods such as Stochastic Gradient Descent (SGD), which computes the gradients of the loss with regard to the network's parameters θ. Cross Entropy is often used as loss function for classification tasks.

According to the definition of adversarial examples, the goal is to generate a new example $\hat{x}(\approx x)$ which *increases* the value of the cost function. Since the input has to be changed to become adversarial, this time *the derivative of the loss with respect to the input* x must be calculated, with θ being the freezed parameters of the already trained network. The sign of the gradient and the additional tunable constant ϵ are used to control the amount of noise and thus satisfy the constraint of eq. 3.15. In other words, by climbing up against the direction of gradient, FGSM finds the adversarial noise that will maximize the cost function, as illustrated in Fig. 3.5.

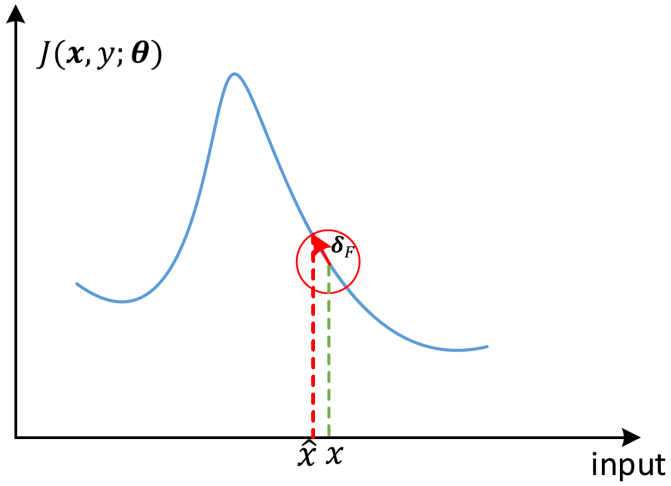


Figure 3.5: finds the worst perturbation δ that maximizes the cost function J, by computing the gradient of the loss with regard to the input x. (Source: Sun et al. (2019))

FGSM generates AAEs starting from any audio input in a supervised way, since it requires the true label y and maximizes the cost with regard to it. Also, it is to be noted that x does *not* represent the *audio* input, but the features that were extracted on a frame-by-frame basis from the audio. Consequently, this method will create *adversarial features* $\hat{x}$ by adding the adversarial noise δ in a point-wise manner to the original, non-adversarial features. To obtain the adversarial audio waveform, one needs to either backpropagate the gradients of the loss through the feature extraction stage, or alternatively, to reverse the adversarial features in the audio domain with the help of inverse operations. Since the former approach is more computationally demanding, we choose the later, further described in Section 4.3.

3.3 MP3 Compression in ASR Context

Audio compression generally aims to reduce the storage space requirements for audio data, which is tremendously useful for a wide palette of applications, such as Internet audio streaming, digital audio broadcasting, digital television and portable audio devices. For this, low-bitrate audio coding is a key requirement, and there are many popular audio formats (e.g., MP3, MPEG-4 AAC, Vorbis OGG, Dolby AC-3) that implement it by exploiting the characteristics of the human sound perception. Correspondingly, they produce a minimal digital representation of the input audio, which is perceptually very similar to the original when decoded, thus achieving high compression rates (Sirum and Sanches, 2004).

MP3 in particular refers to Layer 3 of the MPEG/audio algorithm, which was the first international standard for a high-fidelity digital-audio compression algorithm adopted by ISO in 1992 (Brandenburg, 1999). It uses a lossy, perceptual audio coding scheme based on a psychoacoustic model of human hearing, which considers hearing thresholds, as well as temporal and spectral masking phenomena. Since the standard itself is defined as

open, its specifications are available to anyone for free. But like for any basic perceptual audio coding system, the implementations of MP3 encoders must consist of three basic blocks: a filter bank block, a perceptual model block and a quantization/coding block. In a nutshell, their functionality can be summarized as follows: the analysis filter bank first decomposes the audio data into its spectral components. The perceptual model, following rules from psychoacoustics, estimates the *just noticeable noise levels* for each sub-band of the filter bank. Finally, in the quantizantion stage, the individual spectral components are quantized and coded with the aim of keeping the quantization noise below the level given by the perceptual model so as to meet the bitrate requirements (Sirum and Sanches, 2004).

Despite its widespread popularity and usage, MP3 has its own shortcomings. The encoding method is lossy by nature and introduces audible distortions, which can easily be identified using spectral analysis. The two main factors that cause signal degradation are the low-pass filtering effect and the introduction of unnatural spectral gaps, which are spectrum bins with a very low energy (Borsky et al., 2017). Moreover, being a lossy codec means that when an MP3 file is decompressed, it is not possible to retrieve all information lost during the compression, so the original signal cannot be fully recovered.

In speech recognition, the MP3's shortcomings become relevant because ASR models require massive amounts of data, and MP3 datasets are often preferred due to the low disk space they consume. Considering that the acoustic processing front-end of most ASR systems heavily relies on spectral features, the MP3's spectral distortions raised concerns over their impact on speech recognition performance. It has been shown that training traditional HMM-GMM systems on MP3 data at very low bitrates ($\leq$ 24 kbps) significantly impairs the recognition performance (Sirum and Sanches, 2004; Nouza et al., 2013), because the MFCC features used by these systems are substantially affected by compression. Some compensation measures like additive noise to fill the spectral gaps have also been proposed (Borsky et al., 2017). In comparison, hybrid HMM-DNN systems show higher robustness when dealing with the unnatural spectral sparseness of low-bitrate compressed signals (Seps et al., 2014). As for end-to-end ASR systems, they are usually trained on huge corpora of MP3 data like TED-LIUM (Hernandez et al., 2018) or Mozilla CommonVoice (Ardila et al., 2019) and still achieve state-of-the-art results. What possibly makes them so robust to MP3 compression is that most of the modern ASR corpora of MP3 format are of very high quality, being compressed at high bitrates, with negligible spectral artifacts. Moreover, end-to-end systems are solely based on neural networks, which are more versatile in dealing with any type of data, provided the training set is big enough. Consequently, they are also able to learn the intrinsic patterns of MP3 audio.

Another issue to consider when using compressed datasets is to match the training and testing conditions. This means that it would be most appropriate to train the ASR system with data of the same format as the one used for decoding, otherwise the performance of both HMM and DNN-based ASR systems can drop, as shown by Borsky et al. (2017).

With regard to our research question on adversarial examples, it has been hypothesized that MP3 can actually turn into a robust countermeasure, potentially discarding the adversarial noise that is placed in the frequency bins inaudible to humans. *The effectiveness of MP3 compression to strip the adversarial examples off their malicious character is precisely the main point of investigation in this thesis.*

4

Experimental Setup

This section thoroughly describes the practical steps that have been performed in the quest to validate the research hypothesis, which assumes MP3 compression's suitability as a measure to decrease the effectiveness of adversarial noise, and hence to enhance ASR systems' robustness.

The general experimental workflow is portrayed in the following schematic:

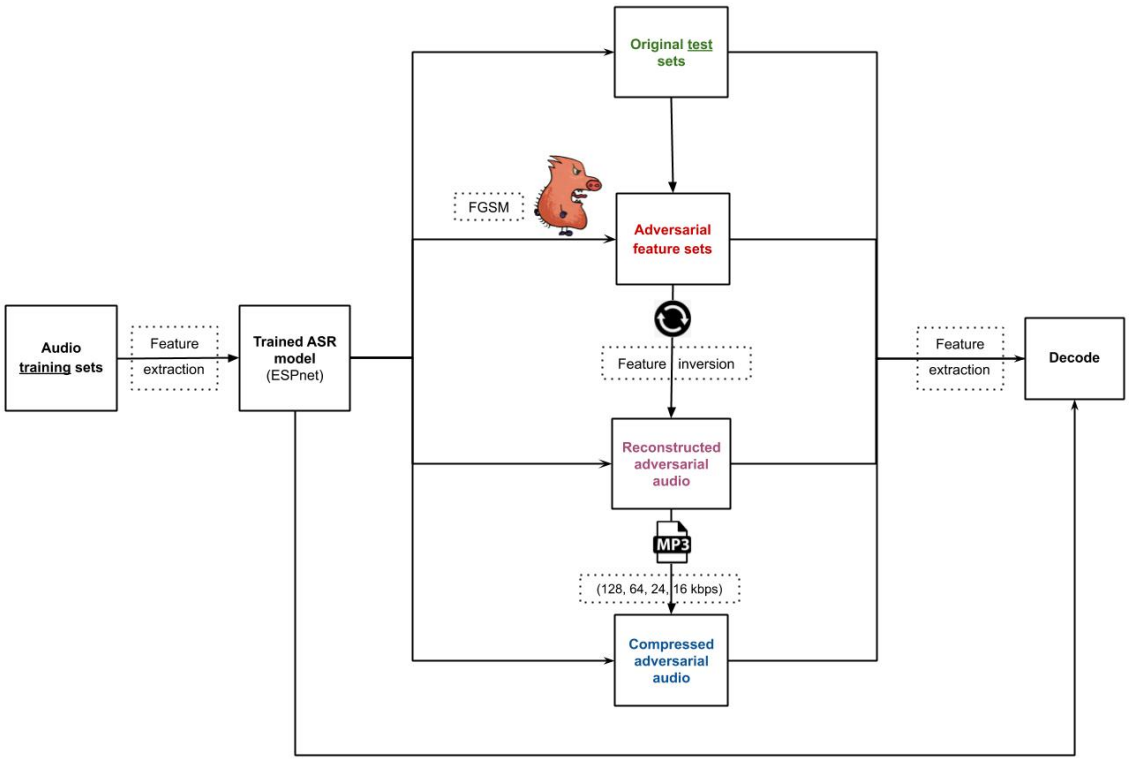


Figure 4.1: Thesis experimental workflow: various ESPnet ASR models are trained with the Voxforge audio training data and subsequently employed to decode four different test sets: the original, non-adversarial VoxForge test set, followed by an adversarial feature set created from the original one with the Fast Gradient Sign Method (FGSM), a test set of adversarial reconstructed audio, and finally test sets of MP3-compressed adversarial audio at various bitrates.

The upcoming sections elaborate on each step in the block diagram.

4.1 The VoxForge Speech Corpus as Audio Input

All the performed experiments are based on the VoxForge dataset[1], which is a relatively small open-source speech database consisting of short, simple sentences uttered by various volunteer contributors in 17 languages. Because the audio setup and recording microphone vary from user to user, the quality of the recordings is not always top-notch, but is overall acceptable. Clogged voices or noisy recordings are not uncommon, however this introduces more variety, which is desired in recognition tasks so that the models learn to generalize well over a variety of inputs.

The English share of the dataset was used in the present thesis. Audio files are originally in *.wav* format and have the following specifications: single channel (mono), 16-bit signed integer PCM (Pulse Code Modulation) values, sampling rate of 16 kHz, bitrate of 256 kbps. The entire set of English audio recordings amounts to ≈130.1 hours collected from 1,234 speakers with various English accents, making it a relatively small speech corpus, compared to others used in the speech recognition community. However, its choice in our experiments was motivated by the fact that it was the only open-source dataset recorded and released in an uncompressed format, which allowed for further compression and thus, for the exploration of our research question.

The original VoxForge dataset is split in three partitions, corresponding to the data for training (80%), validation (10%) and testing (10%) the ASR models.

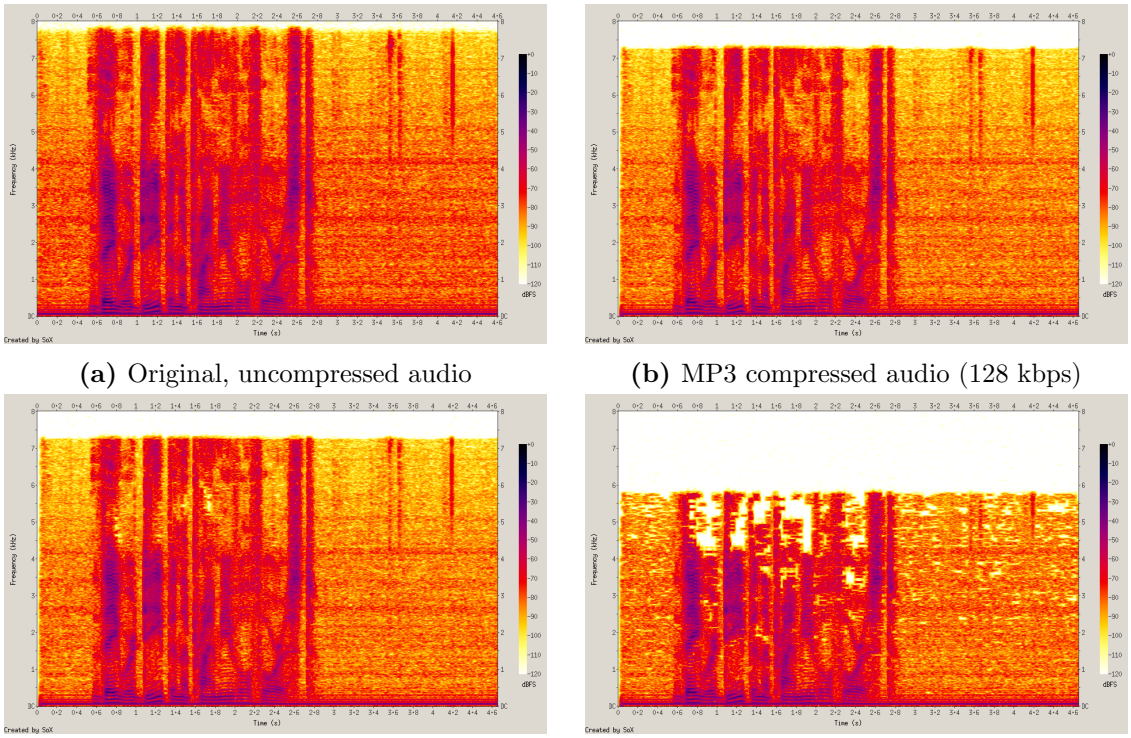
4.1.1 MP3 Compression using *Lame* encoder

Essentially, MP3 compression was applied to the original VoxForge dataset splits at three bitrates (128, 64 and 24 kbps), thus creating three compressed versions of the original dataset. *Lame*[2] was the chosen MP3 encoder and command line tool for batch compression. A volume downscale factor of 0.7 was set so as to reduce the volume of the input audio to 70% of its original amplitude and thus avoid clipping effects during compression, which can cause additional unwanted spectral artifacts. Subsequently, the compressed audio files were converted back to *.wav* format at 256 kbps, as this is the native input format for the employed ASR system (described later in Section 4.2). Notably, this de-compression does not bring further audio quality impairment to the already compressed MP3 files.

To visually explore the effects of MP3 compression, we computed the spectrograms for an audio sample that transcribes to the phrase *They were deep in the primeval forest*, as well as for its MP3-compressed variants at the three bitrates (Fig. 4.2). An open-source command line tool called *sox*[3] was used for creating the spectrograms. The plain visual inspection reveals the two main spectral effects introduced by MP3 compression:

[1] VoxForge dataset (downloaded on 15.11.2019) - http://www.voxforge.org/home/downloads
[2] LAME Ain't an MP3 Encoder - https://lame.sourceforge.io/about.php
[3] sox (Sound eXchange, the Swiss Army knife of audio manipulation) - http://sox.sourceforge.net

(1) high-frequency band limitation and (2) the unnatural spectral valleys, which are regions of very low energy in the mid-frequencies range. It is clearly visible that with minimizing the compression bitrate, i.e., applying more aggressive compression, both these effects become highly prominent.



(a) Original, uncompressed audio

(b) MP3 compressed audio (128 kbps)

(c) MP3 compressed audio (64 kbps)

(d) MP3 compressed audio (24 kbps)

Figure 4.2: Spectral effects of MP3-compression on the speech signal (x and y represent the time and frequency axis, respectively, while the color represents the intensity of the spectral magnitude).

4.1.2 Feature Extraction

Like in any machine learning task, representative features need to be extracted from raw data in order to create a dense representation of the input content. Learning the core information, ideally without any noise, is crucial to train systems that will make correct inferences on new, unseen data. For the ASR task in this thesis, mel filterbank features (abbreviated *mel fbanks*) were extracted by Librosa toolbox (McFee et al., 2015), with the exact steps being illustrated in Fig. 4.3 and the extraction parameters summarized in Table 4.1.

Initially, a sliding window of ∼32 ms traverses the entire input signal, with an overlap of 10 ms, in order to capture the dynamics between frames and hence, the speech phonetic context. The standard *Hann* window (Oppenheim, 1999) is then point-wisely multiplied with each extracted audio frame. The rationale for using this window is that its amplitude gradually drops off near the edges of a frame, thus avoiding an abrupt

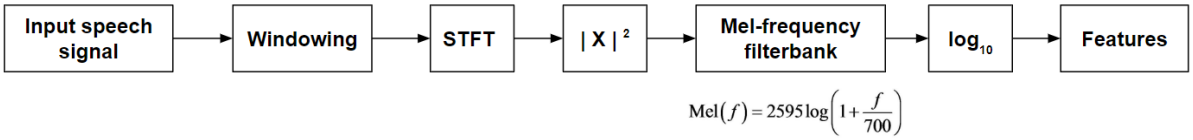


Figure 4.3: Computation pipeline of log mel filterbank features (mel fbanks)

Table 4.1: Feature extraction parameters

Parameter	Value
Sampling frequency	16 kHz
Window type	Hann
Window size	512 samples
Window overlap	160 samples
# of FFT points	512 samples
# of mel freq. bins	80
Minimum mel-scale frequency	80 Hz
Maximum mel-scale frequency	7800 Hz

cutoff at the start and end of each frame, which would cause the occurrence of side frequency lobes, i.e., noise, in the spectral domain. The Discrete Fourier Transform (DFT) is thereafter applied on each windowed audio frame via the Short-Time Fourier Transform (STFT) algorithm, generating the spectro-temporal representation of the signal, i.e., the so-called spectrogram. The absolute values of the spectrogram are then squared to generate the power spectrum, which is subsequently filtered by the mel-scale filterbank.

The mel filterbank (Fig. 4.4) consists of a set of triangular band-pass filters which mimic the frequency selectivity of the basilar membrane in the human ear, thus being psychoacoustically-motivated. The triangular filter bands are increasingly wider at the higher frequencies, reflecting that human hearing is less sensitive in that range. The mel scale amplitudes are calculated with the formula shown in Fig. 4.3 above.

After the mel fbank is applied to each time frame in the spectrogram, the energies in each triangular band are summed up and logarithmized, in order to account for the non-linear loudness perception in humans. Finally, for each acoustic frame, the feature vector consists of 80 values, corresponding to each bin in the log mel fbank.

Notably, log mel fbanks are quite simplistic features, and some other common pre-processing steps such as signal pre-emphasis, dithering, DC-offset removal, mean and variance normalization were deliberately avoided, in order to make the features easier to invert back into the audio domain, as described below in Section 4.3. In ASR, it is also common-practice to append pitch-related features to the mel fbank feature vector. However, decoding trials with and without pitch features did not render a significant difference in recognition accuracy, so we decided not to include them in our feature vectors, thus also avoiding additional complexity in the feature inversion process.

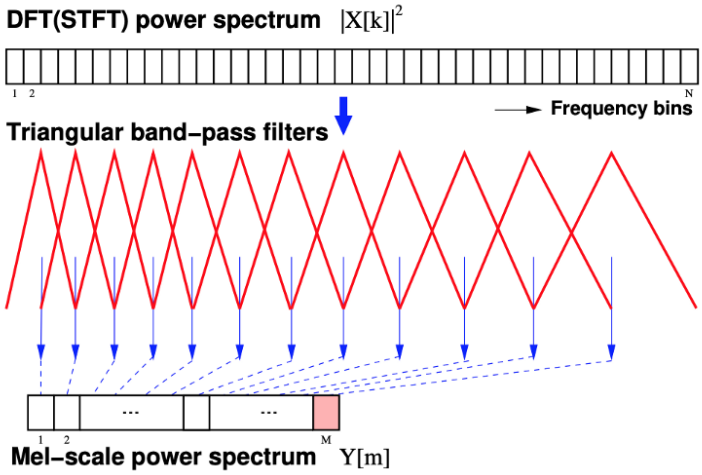


Figure 4.4: Mel filterbank illustration
(source: lecture slides in *Automatic Speech Recognition*, Bell (2018))

A further distinction should be made - the mel fbank features have been favoured against the other popular ASR features called Mel Frequency Cepstral Coefficients (MFCC) for the following reasons. Firstly, MFCC computation requires to further process the mel fbanks and decorrelate them via the Discrete Cosine Transform (DCT), which produces the cepstrum. The dimension of the final feature vector would then be further diminished by selecting only the first N desired cepstral coefficients. These extra steps were initially motivated by certain limitations of the old-fashioned HMM-GMM recognition systems, namely that they were making independence assumptions on their inputs, thus being unable to fit correlated data. The decorrelated MFCCs were thus the ideal input of the traditional HMM-GMM systems.

Neural networks, on the other hand, do not make any independence assumptions, so they can smoothly handle correlated input. In fact, it has been shown that neural-based ASR systems work better in conjunction with correlated fbank features than with MFCCs (Abdel-rahman, 2014). Some preliminary test decodings we performed on both MFCCs and mel fbank feature inputs have confirmed that our ASR systems performed worse, in terms of ∼2-3% higher error rates, when MFCCs were given as input. In light of these results, mel fbanks represented the most reasonable choice for the feature vector.

By and large, mel fbank features were fed as input to the ASR systems in both the training and decoding stages.

4.2 The Targeted ASR System

An end-to-end, fully neural model with hybrid CTC and Attention mechanism is employed for performing the speech recognition experiments on the VoxForge dataset. The CTC, Attention, as well as the hybrid CTC-Attention concept have been explained in detail in the *Technical Background* chapter (Section 3.1.3).

In short, Attention is an effective methodology to perform sequence-to-sequence training, but it suffers from some limitations when it comes to monotonic alignment. This is why CTC loss is used in parallel to aid the Attention mechanism in both training and decoding steps. The multi-objective learning framework unifies the Attention-loss and CTC-loss with a linear interpolation weight (the α parameter from eq. 3.11 in Section 3.1.3), which is set to 0.3 in our experiments, allowing Attention to dominate over CTC.

The two main blocks of our end-to-end network are the shared encoder and the joint CTC-Attention decoder (Fig. 4.5). The model parameters are listed altogether in Table 4.2.

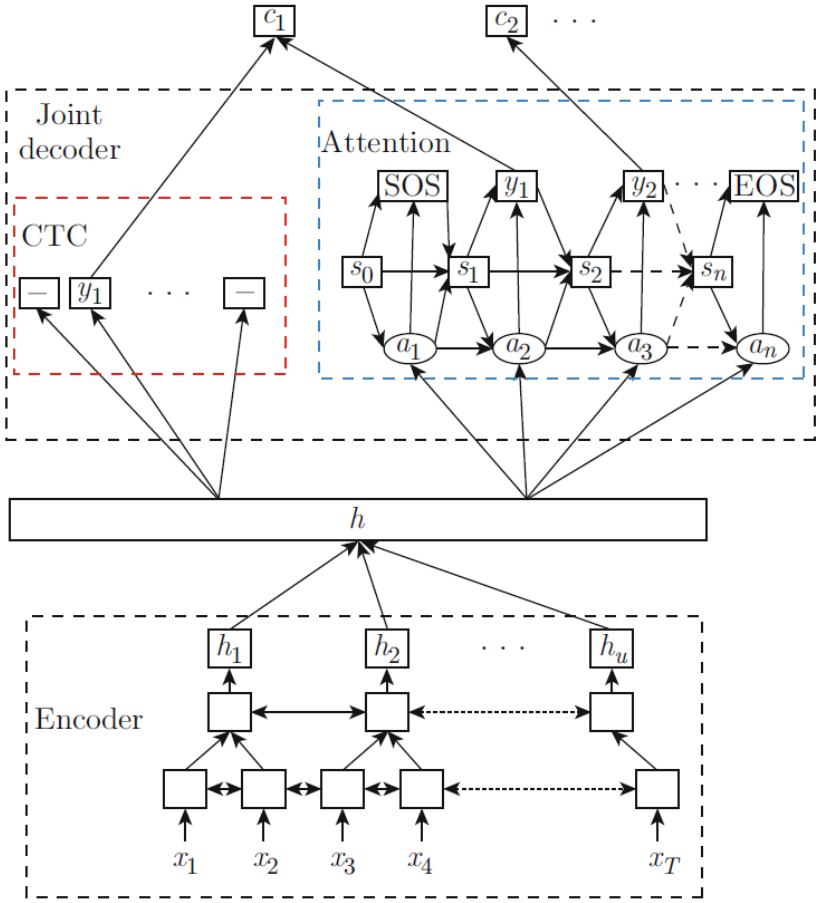


Figure 4.5: Block diagram of the hybrid, end-to-end ESPnet model with joint CTC-Attention decoder (Zhu and Cheng, 2020)

The encoder block consists of initial convolutional layers from the VGGnet architecture (Simonyan and Zisserman, 2014), followed by stacked layers of bidirectional Long-Short Term Memory cells with projections (BLSTMp) (Chan et al., 2015). The bidirectionality implies that a particular BLSTM cell receives inputs from both the previous, as well as the subsequent LSTM cells, thus modelling both the past and future context of the current step. The projections are implemented in a pyramidal structure, such that the sequence output lengths of the second and third pyramidal BLSTM layers are reduced by a factor of 2. Considering the input feature sequence is usually hundreds or thousands of frames long, this layer output sub-sampling is effective in achieving faster convergence.

The joint decoder consists of a single layer with 300 LSTM cells, where a location-based Attention mechanism is present, in conjunction with CTC. The CTC weight in decoding (λ from eq. 3.13) was also set to 0.3, as in the training stage. A Language Model was not available for the VoxForge corpus, so decoding was performed without it, which also kept the computation time shorter.

The AdaDelta algorithm was used as training optimizer. Many trials of network training were run for different hyperparameter sets, until the optimal one was settled upon (the one depicted in Table 4.2). Each training session took $\approx$ 7-12 hours on a NVIDIA Titan Xp graphics card[4], depending on the model's complexity. The ASR system operates on a character-level, outputting sequences of characters, taken from a pre-defined dictionary. Model training and decoding were implemented with the *ESPnet end-to-end speech processing toolkit* (Watanabe et al., 2018), which is in turn based on the *Kaldi ASR toolkit* (Povey et al., 2011) for the data preparation stages.

Eventually, from the entire pool of trained networks, there were only four major ESPnet models chosen for performing the decoding experiments of interest. These four models were trained with the *same* configuration from Table 4.2, but with different input data, listed in Table 4.3. These are training sets of the English VoxForge corpus in four different audio formats: (#1) the original (uncompressed) *.wav* format and (#2 - #4) three MP3 formats compressed at the bitrates of 128, 64 and 24 kbps. When a particular ESPnet model is mentioned in Chapter (5) (*Evaluation*), it is important to keep in mind the audio format of the data it was trained with (uncompressed or MP3 format).

Each of these models is thereafter used to make predictions (inferences) on four different test sets, listed in Table 4.4. A point to note is that all test sets decoded by a particular model originate from the same audio format (uncompressed or MP3) as the one used for training the respective model. The different test sets were represented with different blocks in the experimental workflow scheme (Fig. 4.1). Each test set consisted of $\sim$8000 utterances and the decoding process lasted for about 40 minutes per test set, with the decoding configuration presented in Table 4.2.

[4]Full specifications of NVIDIA Titan Xp graphics card are available at this link

Table 4.2: Optimal ESPnet configuration for the ASR VoxForge English task

Category	Parameter	Value
Encoder related	Encoder type	VGG-BLSTM
	# of encoder layers	4
	# of encoder units/layer	320
	# of encoder projections/layer	320
	Subsampling (skip every n-th frame from input)	1_2_2_1_1
Decoder related	# of decoder layers	1
	# of encoder units/layer	300
	Decoding beam size	10
	CTC-weight (λ)	0.3
	Batch size	25
	Language Model	Not used
	Scheduled sampling probability	0.5
Attention related	Attention type	Location-aware
	Attention dimension	300
	# of Attention convolutional channels	10
	# of Attention filters	100
Training related	Multi-task learning weight (α)	0.3
	Optimizer name	AdaDelta
	# of training epochs	25
	Batch size	35

Table 4.3: Overview of the main four ESPnet models

Model index	Train data
#1	VoxForge uncompressed (original)
#2	VoxForge 128 kbps - MP3
#3	VoxForge 64 kbps - MP3
#4	VoxForge 24 kbps - MP3

Table 4.4: Test datasets to be decoded by *each* trained ESPnet model

Test data for ***each*** **trained ESPnet model**
(1) original test set (non-adversarial)
(2) adversarial features created from (1)
(3) reconstructed adversarial audio created from (2)
(4) compressed adversarial audio created from (3)

4.3 Adversarial Audio Generation

The four ESPnet networks are subsequently integrated with the FGSM algorithm to generate an adversarial instance for each utterance of the four original, non-adversarial test sets (in the four audio formats mentioned above). FGSM, described in more detail in Section 3.2, creates customized, input-dependent adversarial noise, based on the one-step back-propagation from eq. 3.17. Since only the sign of the gradient is considered, the adversarial noise added point-wise to the original speech input is effectively reduced to $\pm\epsilon$. As a reminder, ϵ is the parameter that controls the level of adversarial noise and it was set to 0.3 in our experiments.

It should be pointed out that FGSM is not optimized for psychoacoustic imperceptibility, so there is *no guarantee* that the adversarial noise created with this method could not be eventually perceived by a human. However, despite being audible, the FGSM adversarial noise is not obfuscated, unlike the noise created by Carlini et al. (2016). Their obfuscated adversarial examples sound like heavily distorted speech, but which can still be deciphered by a very attentive listener (audio samples are available at this link[5]). In contrast, the FGSM noise sounds more like short bursts or cracks intertwined with the original voice, which fortunately does not give the slightest hint with regard to the embedded hidden transcription.

Because the networks take acoustic feature vectors as input, FGSM originally creates adversarial examples in the *feature domain*. Yet in order to evaluate our research hypothesis, we needed adversarial examples in the *audio domain*, so we proceeded to *invert* the adversarial features and thus artificially reconstruct the adversarial audio. The exact inversion steps are illustrated in Fig. 4.6.

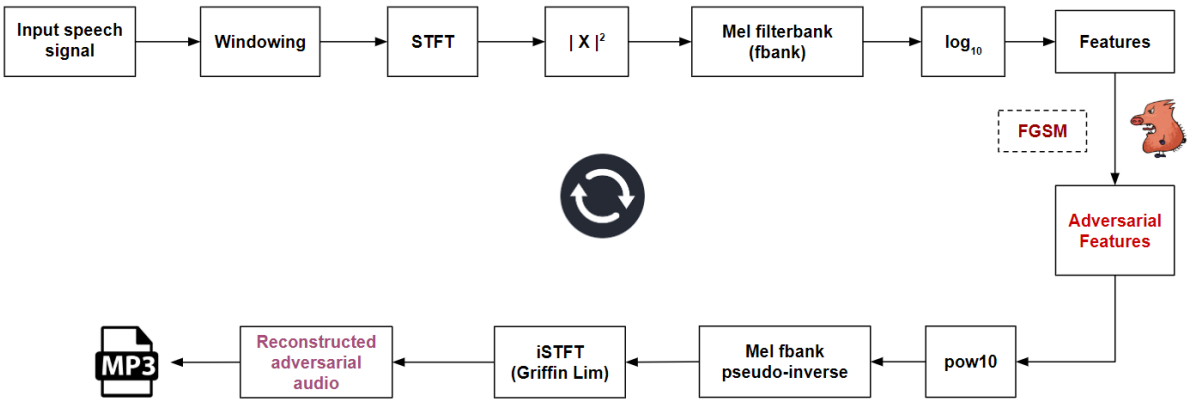


Figure 4.6: Pipeline of adversarial audio reconstruction by means of feature inversion

Practically, every step from the forward feature extraction pipeline is reversed. The power of 10 and pseudo-inverse are applied to reverse the effects of the logarithm and mel-filterbank, respectively. The inverse STFT (iSTFT) was implemented with the Griffin Lim algorithm (Perraudin et al., 2013), which takes in the STFT magnitude matrix

[5]Obfuscated adversarial audio by Carlini et al. (2016) - https://www.hiddenvoicecommands.com/black-box

and randomly initializes the phase estimates. It will then reconstruct a real-valued time-domain signal by alternating forward- and inverse-STFT operations. Analogous to feature extraction, all feature inversion steps were implemented with functions from Librosa toolbox (McFee et al., 2015) in Python.

It is important to point out that mel fbank features are a lossy representation of the original audio input, because they lack the phase information of the spectrum, which in turn renders a highly accurate audio reconstruction simply *not* feasible. Indeed, listening tests revealed that it is relatively easy for a human to distinguish the reconstruction artefacts when comparing pairs of original (non-adversarial) audio samples with their reconstructed versions by means of this method.

Nevertheless, we were mainly interested in the impact of the reconstruction process on the speech recognition accuracy, from the perspective of the ASR system. Thus, we performed the following sanity check: we used the ESPnet model #1 (trained on uncompressed audio files) to decode both the original (uncompressed, non-adversarial) test set and its reconstructed counterpart. We observed just a mild 1.3% absolute increase in the Character Error Rate (CER) for the reconstructed set, which implies that the relevant acoustic features are preserved even after audio reconstruction. This validation experiment overall confirms that speech recognition accuracy for non-adversarial reconstructed inputs is not considerably impaired by the implemented reconstruction method.

Although still imperfect and flawed by audible artefacts, the adversarial audio reconstruction was strictly necessary for being able to proceed with the next experimental step, which consists in applying MP3 compression on the adversarial inputs.

The whole adversarial creation process (including the FGSM and the audio reconstruction step) lasted for ∼20 hours for each test set, using the same NVIDIA Titan Xp GPU. This was a lengthy procedure because the input utterances were processed one at a time, and backpropagating the gradients in the FGSM step posed the strongest computational load. The reconstructed adversarial audio sets were then compressed with MP3 and provided again for the inference stage to the corresponding ASR systems.

Several samples of reconstructed adversarial audio, along with the original audio files and their MP3 versions can be retrieved from this GitHub repository[6].

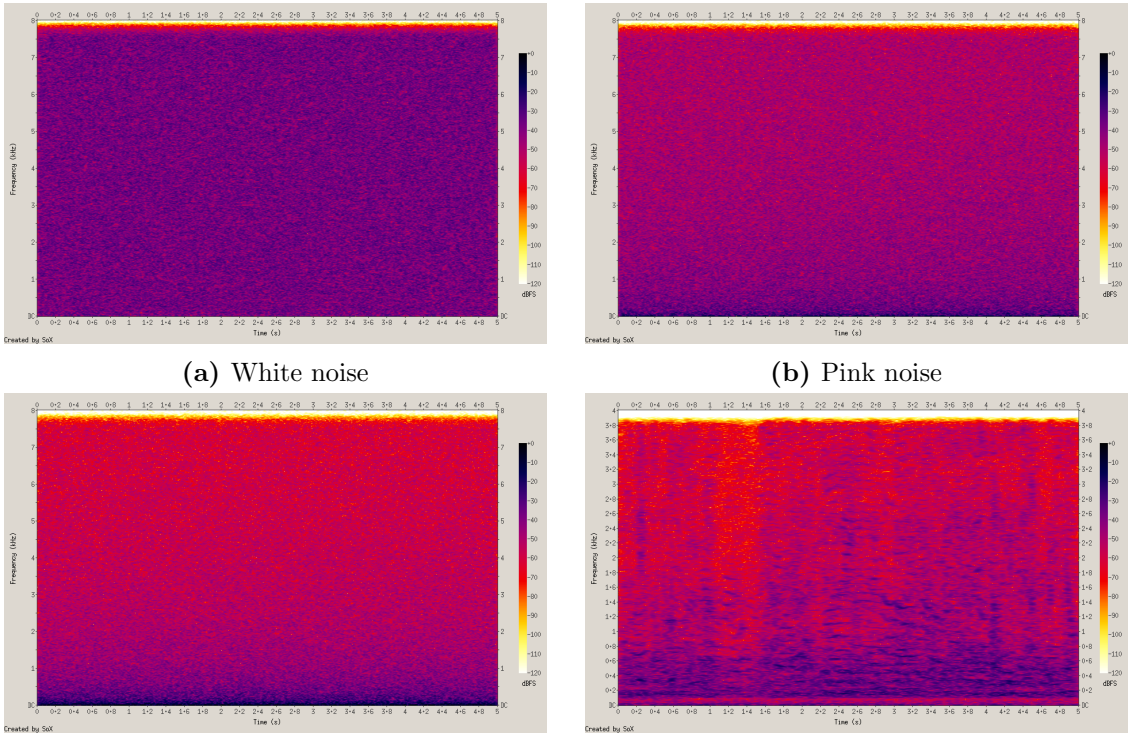
4.4 Data Augmentation with non-Adversarial Noise

It must be acknowledged that adversarial noise is not the only type of noise that can curb the performance of ASR systems. Environmental noise from background sound sources or electrical noise which is intrinsic to the recording equipment have been the primary types of noises explored in the ASR context. Therefore, it would be adequate to assess the effects of MP3 compression on audio that is corrupted by *non-adversarial noise* as well, so as to have a reference for the behaviour of the ASR system when

[6]https://github.com/iustinaabc/Masters-Thesis-MP3-compression-effects-on-adversarial-inputs-in-ASR

exposed to common types of noise. This would offer a wider perspective in which we can then interpret the behaviour of the same system when exposed to adversarial noise.

For this complementary analysis, we augmented the four original test sets with four types of non-adversarial noise, namely white, pink, brown and babble noise, boasting diverse spectral characteristics (Fig. 4.7). The white and pink noise are commonly occurring in electrical equipment, brown noise is a deep sound-wave dominated by low frequency content, while babble noise is a type of ambient sound consisting of simultaneous talking voices coming from different spatial directions, which creates the *babble* impression to the listener. From all these noises, the babble noise is the one with the spectral characteristics most close to human speech, and is thus expected to induce the highest impairment of speech recognition accuracy.



(a) White noise **(b)** Pink noise

(c) Brown noise **(d)** Babble noise

Figure 4.7: Spectrograms of the four non-adversarial noises used in data augmentation (created with *sox* command line tool). x and y represent the time and frequency axis, respectively, while the color represents the intensity of the spectral magnitude

Aiming for a comprehensive analysis, we experimented with six different Signal-to-Noise (SNR) ratios, namely 30, 10, 5, 0, -5 and -10 dB, on each noise. The positive SNR denotes that the overall level of the signal, in our case the speech utterance, is greater than the level of the noise, while the vice-versa holds for the negative SNR. The noise-augmented samples and their MP3-compressed versions are then fed as input to the decoding stage of the corresponding ASR system, i.e., the one that was trained on audio data of the same format as the test data. Mind that all the four ASR systems were

trained on the original, noise-free and non-adversarial data. Consequently, decoding noisy inputs is expected to cause more transcription errors than for clean data.

The general workflow of the noise augmentation experiments is depicted in Fig. 4.8. Augmenting the original VoxForge test sets with four types of non-adversarial noise at six SNRs will render 24 noise-augmented test sets for each ESPnet model, totalling 96 datasets for all the four models. The audio format of the newly noise-augmented datasets corresponds to the format of the original test sets of the respective models. All these datasets will then be compressed with MP3 at 24 kbps, rendering a second pool of 96 noise-augmented test sets in MP3 format, which will also be decoded by the corresponding ESPnet models.

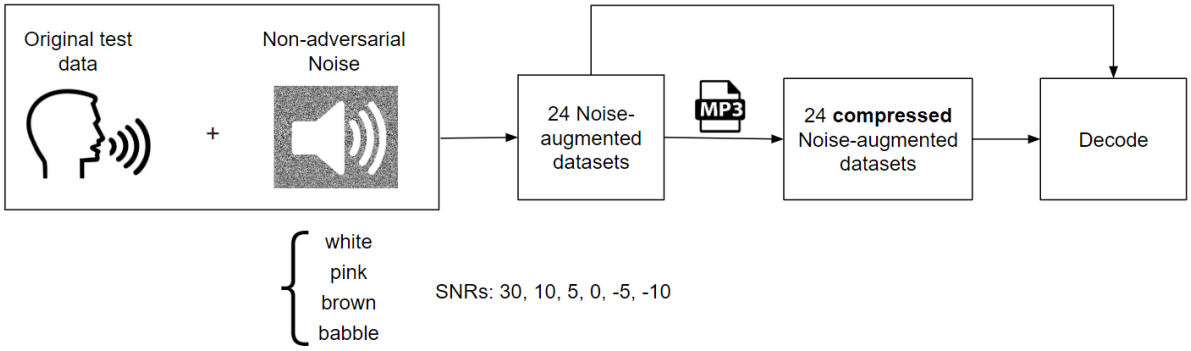


Figure 4.8: Experimental workflow for noise-augmented inputs decoded by a *single* ESPnet model

5

Evaluation

ASR systems' performance is commonly quantified with the *Levenshtein distance* metric (Navarro, 2001). It measures the difference between two text sequences by counting the minimum number of operations required to transform one string into the other. Practically, this distance is calculated between the ground truth transcription and the ASR system's output transcription for every input utterance, then averaged across all utterances and reported as Character or Word Error Rate (CER and WER, respectively). WER and CER scores are computed in a dynamic programming fashion by summing up the number of substituted (S), inserted (I) and deleted (D) units (characters or words), referenced to the total number of units in the ground truth transcription (N):

$$\text{WER/CER} = \frac{S + I + D}{N} \tag{5.1}$$

When testing ASR systems under highly unfavourable conditions, such as very noisy inputs, the output sequence predicted by the ASR system can have many errors and even be longer than the original, in which case CER and WER can exceed 100%.

The following sections report the CER and WER scores for the experiments we undertook by decoding different test sets with the already trained ESPnet models, described in Section 4.2. The test datasets used in the inference stage of *each* ESPnet model were listed in Table 4.4.

5.1 Baseline Results for Original, non-Adversarial Input

Table 5.1 conveys the results of the most basic experiment, that of decoding the original, non-adversarial input utterances with the four ESPnet models. In the following, we will refer to them as *baseline* results. For each model, the input test data was of the *same* audio format as the training data, thus complying with the *train-test format matching* prerequisite.

Table 5.1: Baseline ASR results for original, non-adversarial test data

ESPnet model	Test data format	CER [%]	WER [%]
#1	uncompressed (raw)	17.8	41.4
#2	128 kbps MP3	18.8	43.5
#3	64 kbps MP3	18.6	43
#4	24 kbps MP3	20.2	45.5

The first striking observation is that the WER score is more than double of the CER score for all the four cases, implying that the ASR systems transcribed *many* words with relatively *few* intra-word spelling errors (on the character level), as reflected by the transcription examples from Table 5.2. This is in line with the fact that Attention is dominant in our decoding setup, as it was previously demonstrated that Attention is capable of spelling words quite accurately (hence the low CER), but fails more readily at handling word sequences correctly (high WER) (Kürzinger et al., 2019).

Table 5.2: Example transcriptions decoded by *ESPnet model #1*. REF denotes the reference or ground truth, while HYP is the model hypothesis or prediction. The capital letters denote incorrectly transcribed words, while * symbolizes letters and words *inserted* by the model, but nonexistent in the reference transcription.

REF: beyond DISPUTE corry HUTCHINSON had married MABEL HOLMES HYP: beyond DISPEPED corry HARGENSON had married MABLE HOMES
REF: the VERY thing ernest * AGREED HYP: the FERRY thing ernest A GREET
REF: A CONTROVERSIAL FOUR HUNDRED and fifty *** MILLION dollar project HYP: THE CONTRIVERSAL FULL HNDER and fifty MEN IN dollar project

Since VoxForge is quite an old dataset, it has decreased in popularity within the research community, which made it relatively difficult to find other documented results and compare our systems' performance. To the best of our knowledge, the only place where results for the VoxForge dataset are still published and frequently updated is the ESPnet GitHub repository, since ESPnet is under continuous development and tested on most of the available speech corpora. The ESPnet developers recently reported that an ESPnet system trained with highly similar configuration as ours (VGG-BLSTMp architecture with the same number of network layers and units/layer as our models) achieved 11.6% CER and 42.2% WER on the *Italian share* of the VoxForge dataset[1]. However, because they used a different language, it is difficult to directly relate this result to ours. The only difference was that they used a beam size of 20 in decoding, while ours was 10. However, experimenting with a beam size of 20 did not bring much improvement (only 0.1% reduction in CER and WER scores) on our *English* VoxForge

[1]Kamo Naoyuki. VoxForge results, February 2020.
https://github.com/kamo-naoyuki/espnet/tree/186bddf5cfcea4cf04fccf04e620730804fbafa6/egs2/voxforge/asr1

test dataset. Consequently, in subsequent experiments, we preferred to use the beam size of 10 for faster decoding.

Another notable trend in Table 5.1 is the *slightly* higher percent of errors for MP3 compressed test data, in comparison to the errors for the raw, uncompressed audio (the original VoxForge format). This can be safely justified by the spectral information loss that occurs when applying more aggressive MP3 compression, i.e., at smaller bitrates. However, based on both CER and WER, the reduction in recognition performance to compressed data is not very substantial. The *poorest* result of 20.2% CER was produced for MP3 input at 24 kbps, being only 2.4% above the lowest CER of 17.8%, for uncompressed input.

5.2 Results for Adversarial Input

The results for decoding various *adversarial* test sets with the four trained ESPnet systems are presented in Table 5.3.

Table 5.3: CER results for decoding adversarial input (marked as [2], [3] and [4]). Note: each line in the table represents decoding results for various test sets that originate from the same audio format as the training data, reported in the very first column. The last column indicates a relative CER score difference, calculated as $\frac{[2]-[4]}{[2]} \cdot 100(\%)$. The baseline CER results from Table 5.1, hereby marked by [1], are depicted again for comparison purposes. The term *Adversarial* is abbreviated as *Adv.*

ESPnet model (source format of train & test inputs in [1], [2], [3], [4])	**Input test data**				**Relative CER difference (%) between [2] and [4]**
	[1] Original audio	**[2] Adv. features**	**[3] Reconstructed Adv. audio**	**[4] Compressed Adv. audio (24 kbps)**	
#1 (uncompressed)	17.8	70.5	62.2	57.4	-18.58
#2 (128 kbps-MP3)	18.8	72.3	64	58.4	-19.23
#3 (64 kbps-MP3)	18.6	71.8	63.1	56.5	**-21.31**
#4 (24 kbps-MP3)	20.2	69	60.5	55.3	-19.86

One can first notice that adversarial inputs in the feature domain (column [2] in Table 5.3) achieve the desired effect of increasing the error rate (by an absolute mean value of +52.05% over all models) compared to the baseline errors (column [1]), thus validating FGSM method as effective in creating adversarial features from input data of any audio format (uncompressed, as well as MP3-compressed).

The error scores for *reconstructed* adversarial audio created from adversarial features (as described in Section 4.3) are listed in column [3] of Table 5.3. It is easily noticeable that these error rates are also higher than the baseline (column [1]), but interestingly, lower than the CER scores for the adversarial features themselves (column [2]). This

suggests that our reconstruction method makes the adversarial audio less powerful in misleading the model, which on the other hand is beneficial for the system's robustness to adversarial noise.

When we further compress the adversarial audio with MP3 at the bitrate of 24 kbps, we observe an additional decline in the Character Error Rates (column [4] in Table 5.3), thus indicating that MP3 compression might be favourable in reducing the attack effectiveness of the adversarial input. However, these CER values are still much higher than the baseline (column [1]), by a mean absolute difference of +38.05%, across all ESPnet models.

The overall *relative* decrease in CER between the crude adversarial feature inputs (column [2]), considered most powerful based on the highest error rates, and the *compressed* adversarial audio (column [4]) is shown in the last column of Table 5.3. The strongest reduction effect (-21.31%) can be observed in the case of adversarial compressed input (24 kbps) originating from 64 kbps compressed data.

Explicit Error Patterns to Adversarial Input

It would be interesting to take a more in-depth look at the error patterns caused by the adversarial inputs. We thus enclose within Table 5.4 the detailed errors in terms of percentage of Inserted, Deleted and Substituted units (i.e., words or characters), which emerged when decoding with ESPnet model #1.

Table 5.4: Explicit error patterns for various test inputs decoded by *ESPnet model #1*. These results were obtained by branching out the numbers of the first row of previous Table (5.3). The first reported values represent the percentage of *character* Substitutions, Deletions or Insertions (% CER), while the value in parenthesis is analogous on the *word level* (% WER). The last row represents the summation of the corresponding values from the first three rows and is equivalent to the first row of Table 5.3.

	[1] Original audio	**[2] Adv. features**	**[3] Reconstructed Adv. audio**	**[4] Compressed Adv. Audio (24 kbps)**
% CER (WER) Substitutions	8.6 (33.6)	31.7 (77.2)	29.4 (75.5)	28.8 (73.8)
% CER (WER) Insertions	3.8 (4.4)	30.6 (24.1)	24.4 (20.8)	19.1 (16.3)
% CER (WER) Deletions	5.3 (3.4)	8.3 (3.8)	8.7 (3.8)	9.7 (4.4)
Total % CER (WER)	17.8 (41.4)	70.5 (105.1)	62.2 (100.1)	57.4 (94.5)

In view of this table, a couple of remarks can be made:

- As seen previously in Table 5.3, for adversarial inputs (columns [2]-[4] in Table 5.4), all the error values are higher than for the original, non-adversarial input (column [1]).

- *On the character level* (corresponding to the first values reported in the table), one can ascertain that most prevalent are the Substitutions, followed by Insertions, and ultimately by Deletion errors. The only case that deviates from this trend is the original, non-adversarial input (column [1]), for which slightly more Deletions were achieved than Insertions. On average, the three adversarial inputs yielded 30% character Substitutions, 24.7% character Insertions and 9% character Deletions.
- *On the word level* (corresponding to the value in parenthesis), we can observe a similar trend, but the amount of Deletions, Substitutions and Insertions seems more disproportionate. As such, across the three adversarial inputs (columns [2]-[4]), there were on average 75.5% word Substitutions, 20.4% word Insertions and only 4% word Deletions.

From these error patterns, we can overall conclude that adversarial inputs are most detrimental for Substitutions and Insertions at the word level, since they have the highest number of occurrences. Adversarial inputs impact the Deletions quota as well, but to a smaller extent (when compared to the non-adversarial input in column [1]) and mostly on the character level, with almost no effect on the word level.

Taken altogether, the results presented in this section show that MP3 compression diminishes to some degree the severity of the untargeted adversarial attack, reflected by the smaller error rates in column [4] when compared to columns [2] and [3] in Tables 5.3 and 5.4. Yet even after adversarial audio inputs were compressed, the models were not completely able to recover the correct transcriptions, since the error values in column [4] are still higher than the baseline (column [1]), regardless of the ESPnet model employed or the source format of the data. We could thus conclude that MP3 compression manages to only *partially* reduce the error rates to adversarial inputs.

5.3 Results for Adversarial Input in Train-Test Mismatch Setting

Next, we assessed a scenario of *train-test mismatch*, in which the test data comes from another type of audio format than the train data. This means that the model trained on *uncompressed* data has to decode adversarial input originating from *compressed* data, and vice versa for the model trained on compressed data. Table 5.5 reports the results for decoding adversarial audio in such a train-test mismatch scenario, with two of our ESPnet models, one trained on uncompressed data (#1) and the other trained on MP3-compressed data at 64 kbps (#3).

These results reveal that both ASR models, when presented with adversarial input originating from the opposite audio format than the one of the training data, achieve lower CER scores than in the train-test match case (represented by the asterisk values in columns [1] and [3] of Table 5.5). Additionally, if those adversarial inputs are subsequently compressed by MP3, a further reduction in the error rate occurs (columns

Table 5.5: CER results for decoding adversarial audio with two ESPnet models (#1 and #3) in a *train-test mismatch scenario*. The values with *(*)* are also present in Table 5.3 and denote the CER scores obtained for test data that originated from the *same* audio format as the training data (uncompressed for model #1, and 64 kbps MP3 for model #3), hence no mismatch, while all the other values are obtained for the train-test mismatch case. The bold values are the lowest CER scores obtained by the respective models (note that they both belong to the mismatch case).

Source format of test data	**Test sets decoded with ESPnet model #1**		**Test sets decoded with ESPnet model #3**	
	[1] Reconstructed adv. audio	**[2] 24 kbps MP3 adv. audio**	**[3] Reconstructed adv. audio**	**[4] 24 kbps MP3 adv. audio**
uncompressed	*62.2**	*57.4**	49.5	**47.5**
128 kbps-MP3	51.7	47.5	57.6	53.4
64 kbps-MP3	50.7	**46.5**	*63.1**	*56.5**
24 kbps-MP3	53.8	49.3	61	54.5

[2] and [4]). Specifically, the ESPnet model trained on ***un****compresssed* data (#1) decodes with least amount of errors (46.5%) compressed adversarial inputs originating from 64 kbps *MP3-compressed* data. Likewise, the ESPnet model trained on 64 kbps *MP3-compresssed* data (#3) decodes with least amount of errors (47.5%) compressed adversarial inputs originating from ***un****compressed* data.

Consequently, even though train-test mismatch is generally avoided in ASR tasks, in combination with MP3 compression applied to the input speech, it *appears* to cause a further decline in the error rates. However, these raw results can not tell us what is the individual impact of the train-test mismatch or MP3 compression alone. The analysis should be supplemented by decoding experiments in the train-test mismatch case performed on *non-adversarial input*, and then have the results subjected to statistical tests in order to find significant trends. Though imperative for making significant claims on the effects of MP3 compression, this is a laborious analysis that would have exceeded the available time-span of this thesis.

5.4 Results for Input Augmented with non-Adversarial Noise

In a final set of decoding trials, we experimented with speech inputs augmented with *non-adversarial noise*, which provide a comparison reference for the systems' behaviour to adversarial noise, as described in Section 4.4. These results are presented in Table 5.6.

Based on the CER values, one can overall observe that irrespective of the noise type, the lower the SNR (or the higher level of noise that is added to the input), the more error-prone the ASR system is. This trend is somewhat predictable, because more noise

Table 5.6: CER scores of *ESPnet model #1* for decoding test sets augmented with four types of non-adversarial noise (white, pink, brown and babble noise) and their MP3-compressed versions.

Test sets augmented with:	**SNR [dB]**					
	30	**10**	**5**	**0**	**-5**	**-10**
[1] white noise	19.1	32.7	41.9	53.7	66.2	78.2
& MP3 compressed (24kbps)	29.1	51.2	61.7	71.2	78.7	86
[2] pink noise	18.5	29.1	38.1	51.7	67.4	82.1
& MP3 compr. (24kbps)	26.9	42.5	53	66.4	79.8	89.9
[3] brown noise	17.9	19.7	21.9	26.1	34.1	47.8
& MP3 compr. (24kbps)	25.3	29	32.5	38	47.3	60.6
[4] babble noise	18.3	35.8	53.4	77.4	89	93.6
& MP3 compr. (24kbps)	25.8	48.2	66	83.6	93.1	95.4

is always detrimental to speech recognition. The most adverse effect seems to be in the case of white and babble noise (rows [1] and [4]), which also happen to have the richest spectral content, imminently interfering with the original speech.

Yet of utmost interest here is what happens when we apply the *same treatment* used for adversarial inputs to these novel speech inputs enhanced with non-adversarial noise, namely *MP3 compression*. These results are illustrated every second row in Table 5.6. We can remark that the errors are always higher for MP3-compressed inputs, than for non-compressed inputs, regardless of noise type or SNR. The *relative* difference between all pairs of CER values in Table 5.6 corresponding to uncompressed and compressed inputs augmented with noise are illustrated in Fig. 5.1

The main take from this figure is that the relative CER difference, although SNR-dependent, is always positive, reflecting the impact of MP3 compression in *amplifying* the error rates for speech samples augmented with non-adversarial noise. The graph shows that MP3 compression has less effect for negative SNRs, where the %CER relative increase is almost insignificant (close to 0), but a considerable effect at high and mid-range SNRs, where the CER difference fluctuates between 10% and 50%. Presumably, this can be accounted by the fact that high and mid-range SNRs correspond to small levels of added noise, and if these samples are subjected to MP3 compression, the encoding algorithm will remove not only the noise, but also relevant parts from the original speech signal, which determines the drop in recognition performance. Similar behaviour was observed for the other three ESPnet models (#2, #3 and #4) when presented with original and compressed noise-augmented inputs created from the corresponding audio format (see Appendix A.1).

Nevertheless, when comparing the results we obtained for adversarial and non-adversarial noise, we must keep in mind that *the same pattern* of white, pink, brown and babble noise was added point-wise to every input utterance, while adversarial noise was originally *customized* for each utterance, and was partly determined by the adversarial

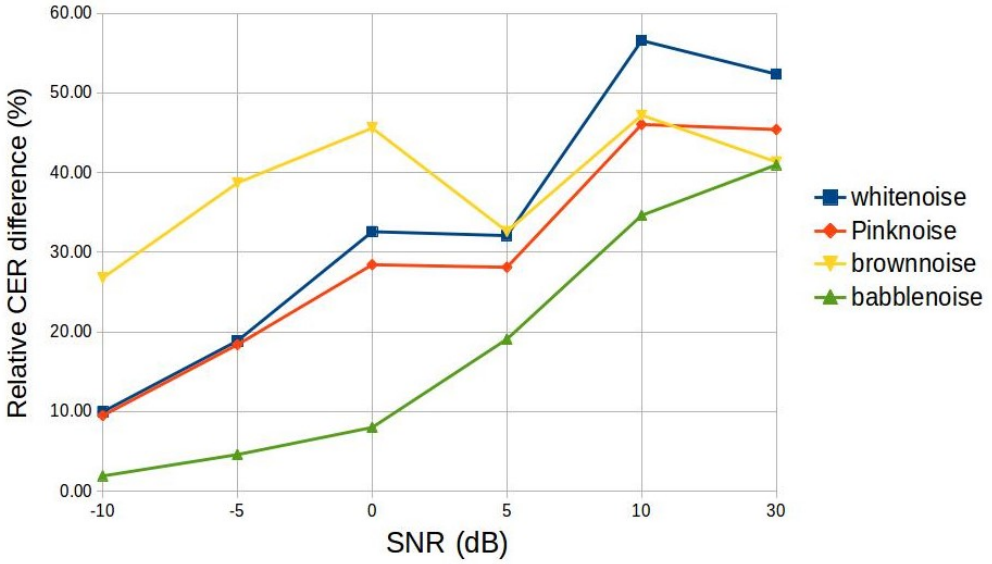


Figure 5.1: Relative CER difference (%) between uncompressed test sets augmented with non-adversarial noise and their MP3-compressed version (24 kbps) decoded by ESPnet model #1

audio reconstruction process. In rough lines, this means that non-adversarial noise is independent of the input utterance, while adversarial noise is utterance-dependent.

On the big picture, we can conclude from the error values that for non-adversarial noise, MP3 compression has the *inverse effect* compared to what was observed for adversarial noise. In other words, MP3 compression was capable to trigger a *reduction* in the amount of errors returned by the ASR systems to *adversarial input*, while *failing* to do so for *non-adversarial noise*. On the contrary, MP3 compression *increased* the amount of errors for inputs augmented with non-adversarial noise, especially at high and mid-range SNRs. This *could* entail that MP3 compression has the desired effect of *partially* reducing only adversarial perturbations, whereas deteriorating the non-adversarial noise.

5.5 Adversarial SNR Estimation

A comparison between adversarial and non-adversarial noise would not be complete unless we also estimate the amount of adversarial noise in terms of SNR. Ultimately, we would like to assess how does MP3 encoding impact the SNR of the adversarial samples. As it is relatively difficult to quantify the SNR in the original domain of the adversarial noise (the feature domain), we attempted to estimate the SNR of the *adversarial reconstructed utterances*, in both uncompressed and MP3 formats. We computed the SNR for each adversarial utterance as:

$$\mathrm{SNR}_{\mathrm{adv}} = 10 \log_{10} \frac{\text{signal power}}{\text{adversarial noise power}} \tag{5.2}$$

$$= 10 \log_{10} \frac{\text{power of the reconstructed speech (non-adversarial)}}{\text{power of the adversarial reconstructed noise}} \tag{5.3}$$

For both the uncompressed and MP3 adversarial utterances, we calculated the SNR with reference to the same signal: the *reconstructed version of the original speech utterance*, so as to introduce the same artefacts that were obtained when reconstructing the adversarial audio, and thus have a more accurate SNR estimation. As expected, we obtained different SNRs for each adversarial audio, because adversarial noise varies with the speech input. Moreover, the SNR values differed before and after MP3 compression for each adversarial sample. The SNRs were in the range [-5, +25] dB for the reconstructed adversarial utterances (uncompressed), and between [-5, +35] dB for their MP3 version, as the top histograms in Fig. 5.2 show.

It is interesting to observe, especially from the close-up plots (the normalized histograms (c) and (d) in Fig. 5.2), that *most* of the SNR values of reconstructed adversarial utterances (uncompressed) are negative, suggesting that the added adversarial noise has high levels and is therefore audible. However, after MP3 compression (Histograms (b) and (d)), most adversarial samples acquire positive SNRs, implying that the adversarial noise was diminished due to MP3 encoding.

To validate this, we applied the non-parametric, two-sided *Kolmogorov-Smirnov* (KS) statistical test (Berger and Zhou, 2014) and evaluated whether the bin counts from the normalized SNR histograms of adversarial samples before and after MP3 compression (plots (c) and (d)) come from the same underlying distribution. We obtained a *p-value* of 0.039, smaller than the critical α threshold of 0.05, legitimizing the claim that the observed difference between the underlying distributions of the two histograms is statistically significant.

When applying the *one-sided* version of the same test, we obtained a more significant result, corresponding to a *p-value* of 0.019. This leads to the approval of the alternative hypothesis which originally assumed that the empirical cumulative distribution function of the SNR values before MP3 compression is *smaller* than the empirical cumulative distribution function of the SNR values after MP3 compression.

Conclusively, this analysis confirms that MP3 compression has a significant effect in increasing the SNR values of adversarial samples, which essentially means that MP3 reduces the adversarial noise.

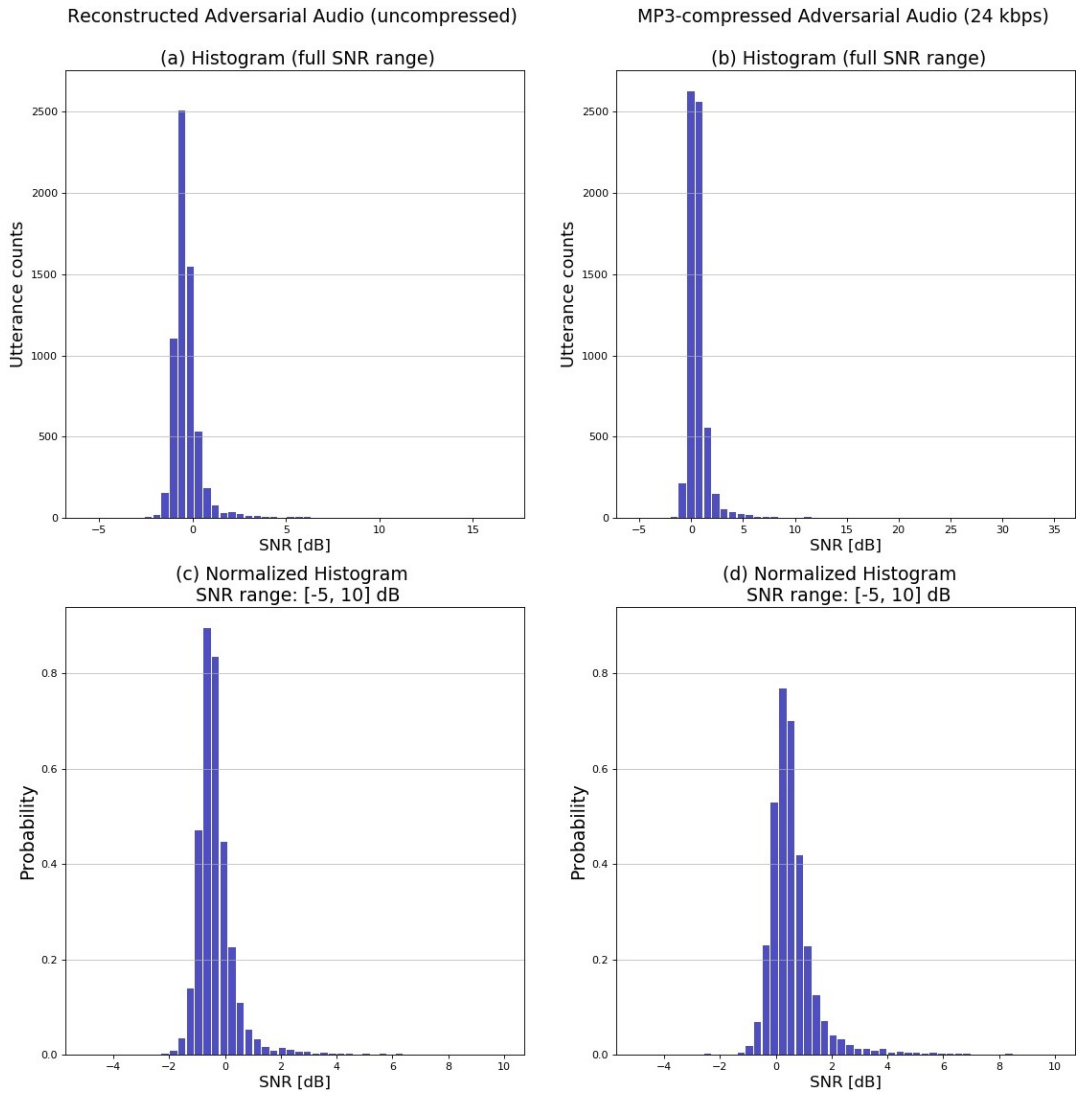


Figure 5.2: Histograms for SNR estimation of adversarial inputs: uncompressed (left plots) and MP3 at 24 kbps (right plots). (a) and (b) represent the original histograms on the full range of SNRs. (Note: because of the y axis scaling, the relatively few counts of extreme positive or negative SNR values are not visible in the histograms, despite being registered). For a better vizualization, (c) and (d) are normalized histograms on a narrower SNR range [-5, +10] dB. Number of histogram bins was set to 50 for all the plots.

6

Concluding Remarks

6.1 General Findings

Adversarial Examples are special input instances intended to induce misclassification in Machine Learning systems, thus exposing their security vulnerabilities. Aligned with contemporary efforts and attempts to overcome these issues, the main scope of this thesis was to investigate a defense strategy that could potentially improve the robustness of ASR systems to adversarial inputs. Specifically, we explored the potential of MP3 compression as a countermeasure to mitigate the effect of adversarial noise compromising speech inputs.

To this end, we trained several end-to-end ASR models with hybrid CTC-Attention architecture on different types of audio formats and engaged them in decoding several test datasets: original inputs, adversarial features, reconstructed adversarial audio, compressed adversarial audio, and finally, inputs augmented with non-adversarial noise. Moreover, the models were evaluated in two general scenarios: train-test match and mismatch, the latter referring to the case when the decoded test sets originated from a different audio format than the training data.

The adversarial feature inputs were built with the Fast Gradient Sign Method in an *untargeted* fashion, i.e., they were meant to be transcribed by the models to any incorrect phrase, deviating from the ground truth transcription. The adversarial features were then mapped back into the audio domain by using inverse feature extraction operations. The resulting adversarial audio (denoted as *reconstructed*) was thereafter MP3-compressed and presented to the models' input.

On the whole, four key findings emerged from our experiments:

1. In a train-test data *matching* scenario, both the adversarial audio reconstruction procedure, as well as MP3 compression of the reconstructed audio were beneficial in reducing the error rates, compared to the raw *adversarial feature* inputs. Nonetheless, the error reduction did not equal the baseline performance for non-adversarial input, which is a sign that the correct transcriptions were not completely recovered, even after applying MP3-compression to adversarial inputs.

2. In a train-test data *mismatch* scenario, it was found that compressing adversarial examples originating from a *particular* audio format and decoding them with models trained on *another* audio format, brings a further alleviation and reduction in the error rates.

3. When applied to speech inputs augmented with *non-adversarial* noise, MP3 compression triggered a *reverse effect* of the systems' response, mirrored by the recognition performance. Specifically, in the train-test match setting, presenting noise-augmented inputs in an MP3-compressed format led to more transcription errors than when those inputs were uncompressed. This finding backs up the first two by suggesting that MP3 compression would be beneficial only in mitigating adversarial noise, while deteriorating the results to non-adversarial noise.

4. Finally, we performed an assessment of the effect of MP3 compression on the estimated SNR values of adversarial samples. The analysis yielded a statistically significant result, supporting the bare observation that MP3 compression increased the SNR values of adversarial samples. Since SNR increase is directly associated with noise level reduction, we can infer that MP3 partially eliminates the adversarial noise.

6.2 Limitations

Inevitably, there are a number of limitations with our experimental setup. In the first place, neural networks are known to be data-hungry systems, and failing to provide sufficient data usually results in a high amount of errors. We believe this is the reason why our systems did not achieve state-of-the-art performance (<15% CER) for the baseline condition, i.e., for the original, uncompressed VoxForge dataset. We assume that the insufficient amount of training data (only ∼105 hours of speech), coupled with the lack of a Language Model in decoding, rendered the sub-optimal results. Unfortunately, we were constrained to use VoxForge corpus because we did not have the financial means to purchase a more extensive dataset in uncompressed format, like the *Wall Street Journal*.

Additionally, our reconstruction method for adversarial audio is quite rudimentary and introduces audible artefacts due to the random phase estimation in the process of STFT inversion. It is thus not unreasonable to think that the reconstruction process itself has to some degree corrupted the original adversarial noise computed in the feature domain. More accurate methods for audio reconstruction, or alternative ways to obtain the adversarial audio (for instance, by means of gradient backpropagation through the entire feature extraction stage) could also be considered in future developments, but at the cost of more computational complexity.

Ultimately, our research hypothesis claiming the MP3 compression's efficiency in mitigating adversarial attacks can be fully validated or declined only by subjecting the ASR performance results to an extensive statistical analysis. However, in order to draw significant conclusions, statistical tests require a high amount of data instances. In consequence, we would have needed more decoding experiments so as to apply statistical tests. Due to time constraints, this was unfortunately not achievable.

All things considered, MP3 compression is only a basic defensive measure against adversarial noise, if compared to others documented in the literature. It mainly aims at direct manipulation of the networks' input, while other defense paradigms aim at

modifying the ASR systems themselves, by means of special training strategies, e.g., adversarial training. If an attacker would know that MP3 is used as a defense measure, he could possibly use this knowledge to create adversarial examples to overcome the compression (for instance, by backpropagating the gradients through the compression stage).

Overall, the research community is very dynamic and has already embarked on the path to design more powerful adversarial attacks that bypass the current defensive measures, which in turn are expected to prompt the development of more efficient defenses.

6.3 Future Prospects

This work lays a foundation for future research and development of effective defenses against adversarial examples targeting ASR systems. One of the possible future directions would be to test MP3 compression against more powerful adversarial attacks, created with complex, e.g., for *targeted* and truly *inaudible* adversarial examples, created based on a psychoacoustic model and verified by standard listening tests. A new research question could arise: if perturbations are inaudible from the beginning, would MP3 algorithm still be effective in removing them?

The project could also be extended by eventually testing the same research hypothesis in a *black-box* attack scenario, where adversarial examples are created without knowledge of the ASR system, but by iteratively presenting different inputs and manipulating them solely based on the systems' response. Finally, future research should address the practicability of audio adversarial examples and explore defensive approaches against *over-the-air* adversarial attacks, which are optimized to work in the real world.

Bibliography

Mohamed Abdel-rahman. *Deep Neural Network acoustic models for ASR*. PhD thesis, University of Toronto, November 2014. URL `https://tspace.library.utoronto.ca/handle/1807/44123`.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

Peter Bell. Lecture slides in automatic speech recognition, 2018. URL `http://www.inf.ed.ac.uk/teaching/courses/asr/lectures-2020.html`.

Vance W. Berger and YanYan Zhou. Kolmogorov–Smirnov test: Overview. *Wiley statsref: Statistics reference online*, 2014.

Michal Borsky, Petr Mizera, Petr Pollak, and Jan Nouza. Dithering techniques in automatic recognition of speech corrupted by MP3 compression: Analysis, solutions and experiments. *Speech Communication*, 86(November):75–84, 2017.

Karlheinz Brandenburg. MP3 and AAC Explained. *Audio Engineering Society, 17th International Conference 2004 October 26–29*, pages 99–110, 1999.

Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018*, pages 1–7, 2018.

Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 513–530. USENIX Association, Aug 2016.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell, 2015. URL `http://arxiv.org/abs/1508.01211`.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *Advances in Neural Information Processing Systems*, 2015-Jan:577–585, 2015.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–11, 2015.

Ij Goodfellow, Jean Pouget-Abadie, and Mehdi Mirza. Generative Adversarial Networks (GANs)-Tutorial-PPT. *(NIPS) Neural Information Processing Systems*, pages 1–9, 2014.

Awni Hannun. Sequence modeling with ctc. *Distill*, 2017. https://distill.pub/2017/ctc.

François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Estève. Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation. In *International Conference on Speech and Computer*, pages 198–208. Springer, 2018.

Shengshan Hu, Xingcan Shang, Zhan Qin, Minghui Li, Qian Wang, and Cong Wang. Adversarial Examples for Automatic Speech Recognition: Attacks and Countermeasures. *IEEE Communications Magazine*, 57(10):120–126, 2019.

Dan Iter, Jade Huang, and Mike Jermann. Generating adversarial examples for speech recognition. *Stanford Technical Report*, 2017.

Daniel Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition.* Pearson Prentice Hall, 2013.

Ludwig Kürzinger, Tobias Watzel, Lujun Li, Robert Baumgartner, and Gerhard Rigoll. Exploring hybrid ctc/attention end-to-end speech recognition with gaussian processes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11658 LNAI:258–269, 2019.

Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

Jan Nouza, Petr Cerva, and Jan Silovsky. Adding controlled amount of noise to improve recognition of compressed and spectrally distorted speech. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 8046–8050, 2013.

Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 2016. URL `http://distill.pub/2016/augmented-rnns`.

Alan V Oppenheim. *Discrete-time signal processing.* Pearson Education India, 1999.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

Nathanaël Perraudin, Peter Balazs, and Peter L Søndergaard. A fast griffin-lim algorithm. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE, 2013.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell, and Colin Raffel. Imperceptible, Robust, and targeted adversarial examples for automatic speech recognition. *36th International Conference on Machine Learning, ICML 2019*, 2019-Jun: 9141–9150, 2019.

Krishan Rajaratnam, Basemah Alshemali, and Jugal Kalita. Speech coding and audio preprocessing for mitigating and detecting audio adversarial examples on automatic speech recognition. 07 2018.

Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *CoRR*, abs/1808.05665(February), 2018. URL `http://arxiv.org/abs/1808.05665`.

Lea Schönherr, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Imperio: Robust Over-the-Air Adversarial Examples for Automatic Speech Recognition Systems, 2019. URL `http://arxiv.org/abs/1908.01551`.

Ladislav Seps, Jiří Málek, Petr Červa, and Jan Nouza. Investigation of deep neural networks for robust recognition of nonlinearly distorted speech. 01 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Paulo Sirum and Ivandro Sanches. The Influence of Audio Compression on Speech Recognition Systems. In *SPECOM' 2004: 9th Conference Speech and Computer St. Petersburg, Russia*, 2004.

Sining Sun, Pengcheng Guo, Lei Xie, and Mei Yuh Hwang. Adversarial regularization for attention based end-to-end robust speech recognition. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 27(11):1826–1838, 2019.

Dong Wang, Xiaodong Wang, and Shaohe Lv. An overview of end-to-end automatic speech recognition. *Symmetry*, 11(8):1–26, 2019.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi. Hybrid CTC/Attention Architecture for End-to-End Speech Recognition. *IEEE Journal on Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. ESPNet: End-to-end speech processing toolkit. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018-Sep:2207–2211, 2018.

Hiromu Yakura and Jun Sakuma. Robust audio adversarial example for a physical attack. *IJCAI International Joint Conference on Artificial Intelligence*, 2019-Aug: 5334–5341, 2019.

Zhuolin Yang, Bo Li, Pin-Yu Chen, and Dawn Song. Towards mitigating audio adversarial perturbations, 2018. URL `https://openreview.net/forum?id=SyZ2nKJDz`.

Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiao Feng Wang, and Carl A. Gunter. CommanderSong: A systematic approach for practical adversarial voice recognition. *Proceedings of the 27th USENIX Security Symposium*, pages 49–64, 2018.

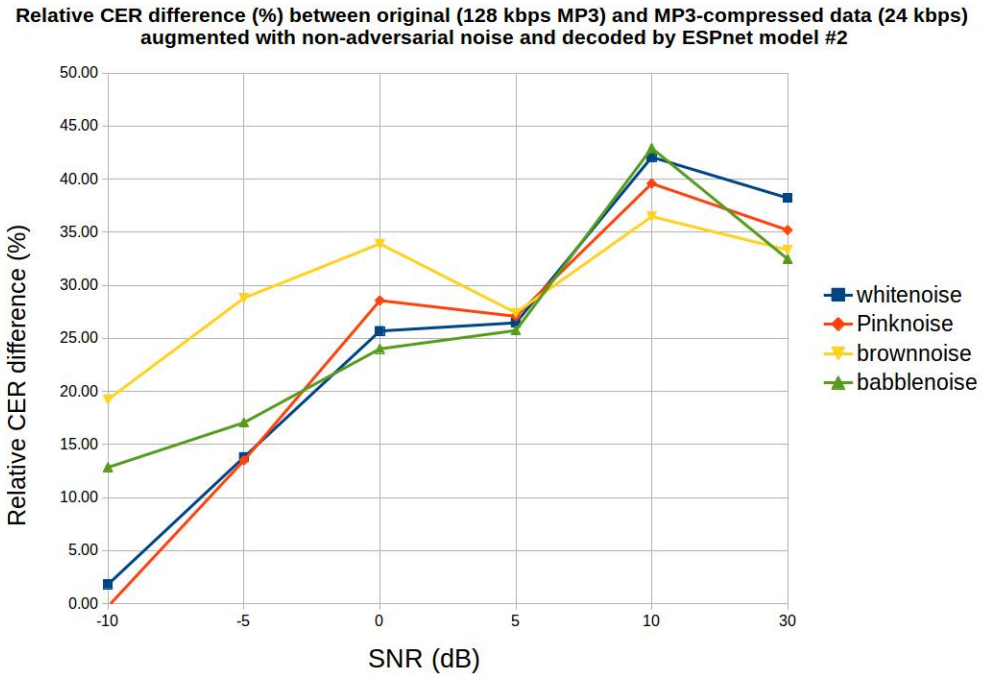
Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. DolphinAttack: Inaudible voice commands. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 103–117, 2017.

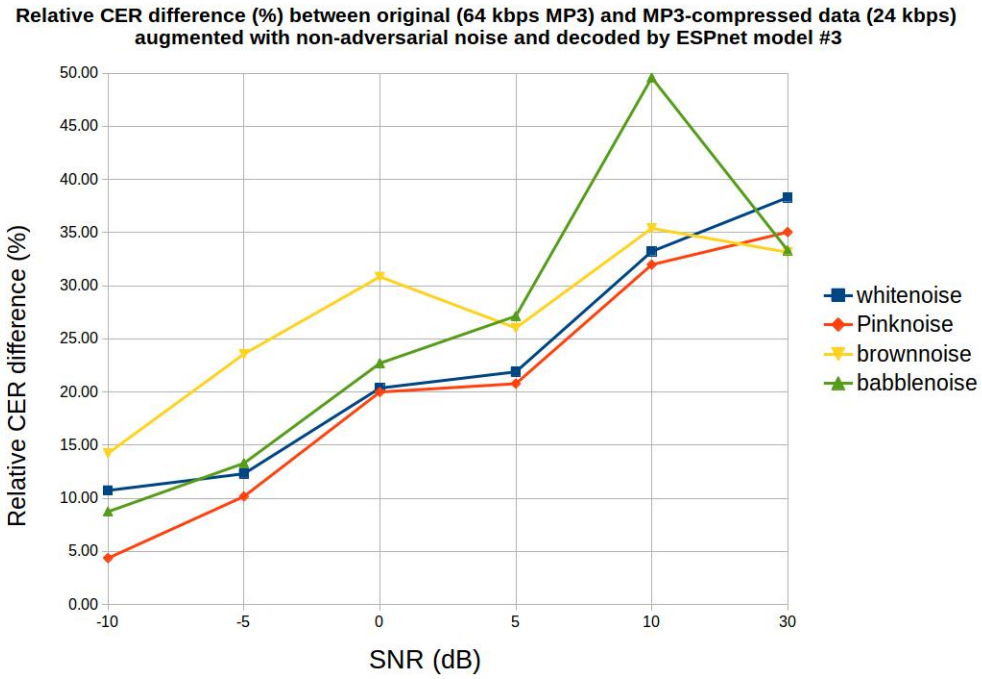
Tao Zhu and Chunling Cheng. Joint CTC-Attention End-to-End Speech Recognition with a Triangle Recurrent Neural Network Encoder. *Journal of Shanghai Jiaotong University (Science)*, 25(1):70–75, 2020.

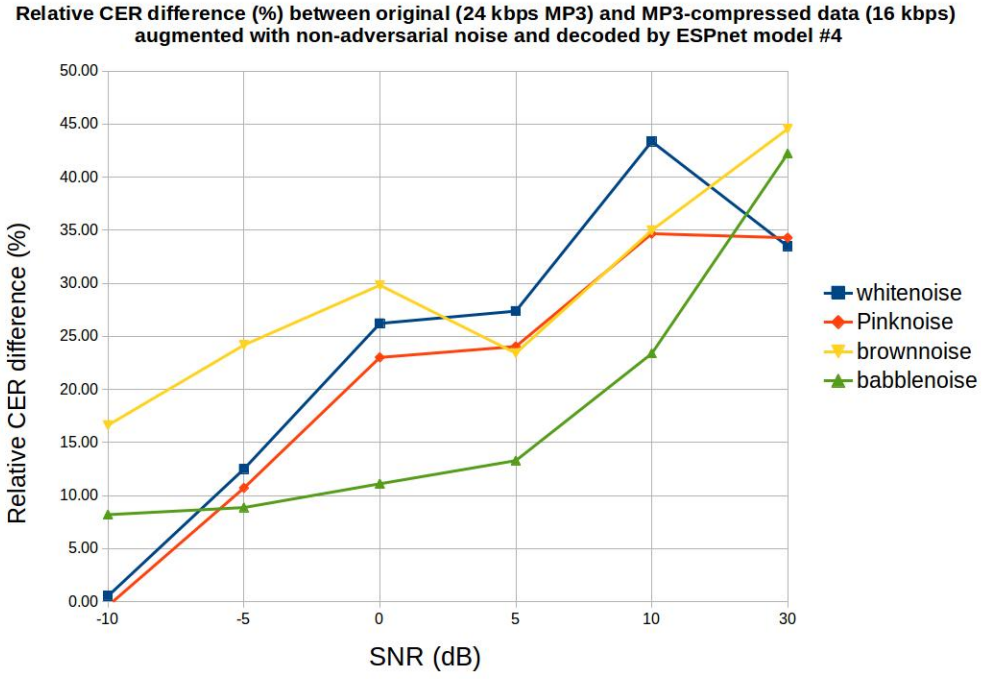
A

Supplementary Results

A.1 ASR Results for Input Augmented with non-Adversarial Noise and decoded by ESPnet models #2, #3 and #4




Relative CER difference (%) between original (64 kbps MP3) and MP3-compressed data (24 kbps) augmented with non-adversarial noise and decoded by ESPnet model #3
Relative CER difference (%)
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
-10
-5
0
5
10
30
SNR (dB)
whitenoise
Pinknoise
brownnoise
babblenoise


Relative CER difference (%) between original (24 kbps MP3) and MP3-compressed data (16 kbps) augmented with non-adversarial noise and decoded by ESPnet model #4
Relative CER difference (%)
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
-10
-5
0
5
10
30
SNR (dB)
whitenoise
Pinknoise
brownnoise
babblenoise