Automatic Transcription of Air Traffic Controller to Pilot Communication
Training Speech Recognition Models with the Open Source Toolkit CoquiSTT

M. May, M. Kleinert, H. Helmke
German Aerospace Center, Institute of Flight Guidance,
Lilienthalplatz 7, 38108 Braunschweig, Germany

Abstract
Despite all the advances in automation and digitalization the majority of communication between air traffic controllers and pilots is still implemented via analogue radio voice transmissions. If support systems also want to benefit from the verbal controller-pilot-communication, manual time-consuming inputs via mouse and keyboard are required. Automatic speech recognition (ASR) is a solution to minimize these manual inputs. Recently DLR, Idiap and Austro Control demonstrated that pre-filling of radar label entries supported by ASR already reaches a technology readiness level of six. The used ASR engine is based on Kaldi, which requires high expert knowledge of ASR for implementation and adaptation. Besides Kaldi a lot of open-source end-to-end ASR models like Whisper or wav2vec are available and are already pre-trained on large amounts of data of normal voice communication. These open source end-to-end models are often easier to adapt even for none speech recognition experts.
This paper presents the results, which the DLR achieved with the open-source CoquiSTT toolkit, which provides an already pre-trained English end-to-end model with 47,000 hours of regular English speech achieving a word error rate of 4.5% on the LibriSpeech clean test corpus. Using the model, however, on air traffic control voice communication results in word error rates of worse than 50%, even in lab environments. Training new models from scratch just on 10 hours of voice recordings from the target environment already makes word error rates below 10% possible. The best performance, however, is achieved, when the CoquiSTT pre-trained model is fine-tuned with air traffic control data from different areas. Word error rates below 5% were achieved, which enable, e.g., callsign recognition rates of better than 95%.

1. INTRODUCTION

Despite all the advances in automation and digitalization the majority of communication between air traffic controllers (ATCos) on the ground and pilots in the aircraft cockpit is still implemented via analogue radio voice transmissions. For this task ATCos are supported by various systems that provide information and make suggestions to improve the flow of air and ground traffic. These support systems often derive their information from surveillance data and flight plan information. The content of analogue voice transmissions, which directly influences the air traffic behavior is usually hidden from these systems.

In recent years research has shown that systems capable of recognizing and understanding air traffic control (ATC) voice transmissions provide benefits in multiple areas: Kerosene savings of up to 60 kg per flight are possible [1], workload of ATCo is reduced [2] or the time an ATCo just needs for maintaining the radar label content is reduced by a factor of 30 [3]. The core of these systems usually consists of two central components "Speech-To-Text" (STT) and "Text-to-Concept" (TTC). STT transforms the analogue voice transmission into a sequence of words that represent the spoken content of the transmission. TTC subsequently extracts defined concepts that clearly describe the semantics of the word sequence. Most of the time STT and TTC are implemented independent from each other, which allows the use of different models and approaches for both without affecting the performance of the other component.

The implementation of the STT component in the context of ATC-applications is often based on expert frameworks like Kaldi [4]. These frameworks require high expert knowledge in the field of automatic speech recognition (ASR) for implementation and to reach a suitable performance. In recent years so called open source end-to-end models like whisper [5] or wav2vec [6] gained more and more attention, see the application of wav2vec for ATC applications [7]. These end-to-end models often come with easier implementation and adaptation processes. This enables also none speech recognition experts to reach suitable performances in different target areas.

In this paper we evaluate the CoquiSTT toolkit as solution for the STT component in an automatic speech recognition and understanding system for ATC applications. For this purpose, we use the toolkit to train different speech recognition models on ATC data sets for different ATC areas, i.e. apron control, remote tower, approach and enroute. Training and evaluation data are used from both laboratory as well as operational environments from a variety of sectors and airports. Based on the available data sets different CoquiSTT training modalities for end-to-end speech recognition are explored and compared.

Before the training of final speech recognition models for different target areas an optimization of the hyperparameters learning rate, batch size and dropout rate is executed. The goal is to find the best parameters with respect to achieving the best word error rate. The optimization is evaluated and the resulting hyperparameters are afterwards used for training models with the full data sets. One of the tested approaches is to train an end-to-end speech recognition model from scratch with ATC data only specifically for a certain target area, e.g., a specific apron or approach area.

In comparison to the ATC model from scratch another approach utilizes the provided pre-trained CoquiSTT model, which is already based on 47,000 hours of regular English speech. This model is fine-tuned with different ATC data sets to determine, whether a pre-trained regular English model provides also benefits for recognition of ATCo pilot voice communication with its special challenges of, e.g., very high word rate and special phraseology.

The third approach aims at training a generalized end-to-end model for the ATC domain. This model is trained from scratch with data from multiple sectors and airports with the goal of measuring, whether a generalized ATC model can reach sufficient performance in case where not enough data from a specific area is available. To further improve the performance the CoquiSTT toolkit enables the integration of KenLM language models to be combined with the already trained end-to-end models. Therefore, for all tested approaches different n-gram language models are created and evaluated as well.

Section 2 gives an overview of related work with respect to the usage of ASR in ATC. Section 3 describes the conducted validations including a brief description of the CoquiSTT toolkit, the used data sets, the validation hypotheses and the conducted hyperparameter optimization. Section 4 describes the achieved results with respect to the defined hypotheses. Section 5 concludes the work.

2. STATE OF THE ART

ATC refers to the personnel and technology responsible for ensuring the safe flow and monitoring of both air and ground traffic. As the number of flights increases annually worldwide, the importance and complexity of air traffic control grow, necessitating continuous improvements in the methods and tools used by controllers. ASR presents numerous opportunities to reduce controllers' workload [2] and enhance air traffic safety [8].

One potential application of ASR systems in ATC is replacing pseudo-pilots in simulations and controller training. Automated pilots using speech recognition and additional software could respond to controllers and carry out their commands during simulations, eliminating the need for pseudo-pilots and significantly reducing costs during training [9]. Early applications of replacing pseudo-pilots by ASR are from e.g., FAA [10] , DLR [11], MITRE [12], DFS [13].

Measuring ATCo workload is challenging due to the many factors involved. However, with an ASR system, radio communications between ATCos and pilots can be monitored and analyzed more accurately, allowing for better workload assessment and more informed decisions [2].

ASR systems can also simplify and enhance daily operations for controllers. Currently, controllers must manually input each command into flight strip systems using a mouse, keyboard, or touch devices to ensure accurate monitoring. ASR could automate this process, transcribing spoken commands directly into the system [7]. Moreover, ASR could integrate with other controller assistance systems, providing valuable information on traffic conditions, specific flights, or potential conflicts between aircraft.

However, implementing ASR in ATC comes with challenges. Audio from communications between controllers and pilots is often noisy, making it difficult to accurately recognize commands. Additionally, pilots and controllers do not always adhere to standardized phraseology. They may use native language words or non-standard abbreviations, requiring ASR systems to recognize terms they aren't designed for. The international nature of air traffic control further complicates recognition due to differences in dialects, accents, and native languages, which can affect the system's performance[14].

Previous research has explored training speech recognition systems in the ATC communication domain using a deep learning toolkit. This earlier work used the Mozilla Deep Speech toolkit, the predecessor to the CoquiSTT toolkit, and some of the same data was utilized. The previous study achieved a word error rate of 6.0% using these datasets and the Deep Speech toolkit [15]. A different research approach was taken when a large-scale, weakly supervised ASR model named Whisper was applied on ATC communication data [16]. Whisper is a model developed by OpenAI with 680,000 hours of training data. The method of fine-tuning was also tested in this approach. On real world ATC data, a WER of 13.5% could be achieved. On simulated ATC traffic voice data, the achieved WER was 1.17%. Overall, this research approach showed that fine-tuning led to a performance improvement by 60%. While Whisper does not yet match human level performance, it shows promise in reducing workload and supporting incident analysis in ATC settings [16].

In another research approach, the performance of pre-trained end-to-end ASR models like Wav2Vec 2.0 and XLS-R within the domain of ATC communication was investigated. The objective was to examine the robustness of pre-trained ASR models in domain shifts, specifically in ATC communication. The researchers analyzed the models' performance in different scenarios using datasets that include significant background noise. The pre-trained models demonstrate strong performance even in challenging ATC datasets, with a reduction in WER by 20-40% compared to hybrid models. The study provided valuable insights into the performance of ASR models in specific domain-focused scenarios such as ATC [7].

3. VALIDATION

This section describes the validation trials. We start with the descriptions of the CoquiSTT tool kit, followed by the used data sets and the validation hypotheses.

3.1. CoquiSTT Toolkit

CoquiSTT is an open-source toolkit developed by the company Coqui for speech-to-text applications. It enables the training and deployment of deep learning models designed to transcribe audio files into text. CoquiSTT was established in 2016 by a group of former Mozilla employees who had previously worked on Mozilla's DeepSpeech project [17]. After leaving Mozilla, they began developing CoquiSTT based on Baidu's DeepSpeech algorithm [18].

CoquiSTT can be utilized for two interconnected purposes. First, it allows for the training of an acoustic model that converts audio data into corresponding transcriptions. This process involves converting the audio into a sequence of letter probabilities, which are then used to generate word sequences [19]. Additionally, CoquiSTT can be used to create a language model (LM) using a tool called KenLM [20]. In CoquiSTT the language model is the so-called scorer. The scorer model predicts the likelihood of a word based on the preceding words in a sequence, which helps improve the accuracy of transcription, especially when dealing with noisy audio or unclear pronunciation. The scorer model adjusts the probabilities of subsequent words to better match the context, thereby enhancing the recognition of domain-specific language data [21].

The CoquiSTT toolkit also includes an API that allows the trained acoustic model and, optionally, the language model to be used for transcribing audio files in various applications.

3.2. Data Sets

To train an acoustic model with CoquiSTT, specific datasets are required. Each dataset typically includes the path to an audio file, the file's size in bytes, and a transcription of its content. During training, the path allows the audio file to be accessed, while the transcription provides the correct text. The file size is used to optimize training, especially when processing batches of audio files. The datasets are divided into three categories: training data, validation data, and test data. Training data is used solely to teach the model, while validation data is used to assess the model's performance after each training epoch, ensuring that the model is improving. Test data is used at the end of the training process to evaluate the model's final accuracy, typically measured by the Word Error Rate (WER) and Character Error Rate (CER). It's crucial that these datasets do not overlap to maintain the integrity of the training, validation, and testing processes. Preparing the datasets involves combining the appropriate file size and transcription with the corresponding audio file path, ensuring that the model is trained, validated, and tested with accurate and well-organized data.

The following TAB 1 shows the used data sets. We have 12 different data sets. Blue rows show recordings directly from the operational environment. Seven data sets result from the approach traffic, which covers from flight level 150 down to 2000 feet. The apron area covers all apron commands like push back and taxi clearances, but excludes all runway related tower commands. Enroute traffic considers the flights above flight level 300 including some approach traffic to local airports. Remote tower stands for multiple remote tower operations and includes the startup approved clearances, taxi clearances, but also descend and climb commands in the tower area.

			Training Data		Validation Data		Test Data	
			Files	Hours	Files	Hours	Files	Hours
Approach	1	Lab	6931	8.5	2144	2.7	2297	3.0
Approach	1	Lab 2	4115	4.7				
Approach	1	Ops	6104	7.7	1092	1.3	1874	2.3
Approach	2	Ops	8882	9.4	3739	3.5	2022	2.0
Approach	3	Lab	4219	4.5				
Approach	3	Ops	2893	4.1	207	0.3	354	0.5
Approach	4	Lab					2277	2.4
Apron	5	Lab	17256	22.3	4673	5.0	5686	6.0
Enroute	6	Lab	2395	3.2	1175	1.3	2903	3.4
Enroute	7	Ops	4933	6.3	2873	3.3	1791	2.1
Remote Tower	8	Lab	12583	8.5	278	0.6		
Remote Tower	9	Lab					706	1.4
Sums			70311	79.2	16181	18.0	19910	23.3

TAB 1. Training, Validation and Test Data sample sizes

Data from the lab environment, but also from the operational environment is used. For this evaluation we excluded noisy pilot utterances. The approach traffic results from four different airports, the remote tower traffic is based on two different simulations and the enroute traffic is based on traffic from two different air navigation providers. One data set is only used for training (Remote Tower 1). No test data was defined. Approach 4 and Remote Tower 2 are the other way around. We did not use these data sets for training, but just for the validations. TAB 2 summarizes the different data sets of TAB 1.

		Training Data		Validation Data		Test Data	
		Files	Hours	Files	Hours	Files	Hours
	Lab	47499	51.6	8270	9.6	13869	16.3
	Ops	22812	27.6	7911	8.4	6041	7.0
Approach		33144	38.9	7182	7.7	8824	10.3
Apron		17256	22.3	4673	5.0	5686	6.0
Enroute		7328	9.5	4048	4.6	4694	5.5
Remote Tower		12583	8.5	278	0.6	706	1.4

TAB 2. Grouped Training, Validation and Test Data sample sizes

Approach 4 contains not only 2.4 hours of training as shown in TAB 1, but more than 66 hours of test data. For most of the experiments we use only the subset of two hours, because the processing of the total 38,716 files consumes more than two days of processing time. The achieved WER on the smaller data set in these cases already shows that the WER are very bad, i.e. even with the smaller data set we are easily able to falsify or verify the validation hypotheses described in the next subsection.

3.3. Validation Hypotheses

The following hypotheses were tested during the final validation trials. The name in brackets is the short name of the hypothesis.

- H1 (Basic-Coqui-OK): the basic CoquiSTT model trained on 47,000 hours of regular English is suitable for recognizing and understanding ATC communication of air traffic controllers.

If we receive word error rates below 10%, we can accept the hypothesis, otherwise the hypothesis is considered as falsified.

- H2 (One-From-Scratch-Fits-All): Training a recognition model from scratch for one application area fits also for other airports or application areas.

- H3 (One-Fine-Tuned-Fits-All): Fine-tuning the basic CoquiSTT model for one application area fits also for other airports or application areas.

If the average word error rate measured on the application area increases by more than 5% absolute, the corresponding hypothesis is considered as falsified.

- H4 (Fine-Tuning-Improves): Fine-tuning the basic CoquiSTT model for one application area improves the recognition performance for that application area.

If the average word error measured on the test data for the corresponding application area decreases for the fine-tuned model, we can accept this hypothesis.

- H5 (General-Fine-Tuning-Improves): Fine-tuning the basic CoquiSTT model with training data from different application areas make the model more robust.

If the average word error measured on the test data for different application areas decreases for the model fine-tuned with data from different application areas we can accept this hypothesis.

3.3.1. Independent Variables

The independent variables (IV) of the final validation trials were as follows:

- (IV-Scen): Selected scenarios, i.e. 30 to 120 minutes time slot with utterances from an ATCo either from a simulation run or from the operational environment.

3.3.2. Dependent Variables

The dependent variables of the final validation trials are

- (DV-WER): Word error rate as defined by the Levenshtein distance [22].

- (DV-SER): Sentence error rate, i.e. number of ATCO transmissions with a WER greater 0% divided by the total number of transmissions.

3.4. Hyperparameter Tuning

Hyperparameters significantly influence the training process of neural networks. Unlike model parameters, which are learned from the data during training, hyperparameters are predefined before the training process begins and more or less remain constant for the whole process. To achieve optimal performance when training speech recognition models on our ATC data sets using the CoquiSTT toolkit, we conducted a series of hyperparameter tuning evaluations. We focused on optimizing the three key hyperparameters dropout rate, learning rate, and batch size. The reported WER are based on training the acoustic model and in a second step the scorer. Although training the scorer does not influence the results with respect to the best hyperparameter combination, we perform this additional training step, because it reduces the word error rate by a factor of two to three. Training the scorer just takes some minutes of runtime. Therefore, we always present the WER after also having trained the scorer. The WER are then more meaningful. We get a better feeling which order of magnitude we can expect when, using the corresponding combination of hyperparameters.

Given the computational demands and time constraints of hyperparameter optimization, particularly for large data sets, we limited our hyperparameter tuning evaluation to the data set Appron-5 which, with 22.3 hours of training data, is the largest individual data set available for our analysis (see TAB 1 for a detailed description of used data). Therefore, we assumed that the results would be representative across the other data sets. Furthermore, for the tuning process the validation portion of the data set (5.0 hours) was used as test set for the WER evaluation. The test portion of the data set (6.0 hours) was not used within the tuning process to not unintentionally bias the results of speech recognition models, which are trained after hyperparameter tuning.

For the optimization, a range of values for each hyperparameter was selected. Each selected value or a combination of values was used to train a speech recognition model from scratch for 20 epochs with the Appron-5 data set. This process was repeated three times for each value or value combination and the WER was averaged to ensure robust results.

3.4.1. Dropout Rate

Dropout refers to the technique of randomly ignoring a fraction of neurons during the training process during each iteration. This helps to prevent overfitting by ensuring that the network does not become overly reliant on particular neurons, thereby improving the model's ability to generalize to new data.

Dropout rate	Word error rate (WER)	Standard deviation of WER
0.40	8.46 %	0.13%
0.50	8.61 %	0.08%
0.20	8.62 %	0.21%
0.35	8.77 %	0.15%
0.30	8.94 %	0.33%
0.05	9.14 %	0.48%
0.10	9.47 %	0.28%

TAB 3. Results hyperparameter tuning dropout rate sorted by best WER first

TAB 3 shows the evaluated dropout rates and the results with respect to the WER. These are the averaged results achieved after training three models for 20 epochs with the same hyperparameters. We repeat the training runs with the same dropout rate value three times to compensate for random effects because the ignored neuros are randomly chosen. All hyperparameters beside the dropout rate remained constant for the training runs and used the CoquiSTT toolkit default values. Overall the best output was achieved with a dropout rate of 0.40, which resulted in a WER of 8.46%. The standard deviations in the last column shows that the randomization has an effect, but not a big effect. Therefore, all further model training after the hyperparameter tuning are executed with a dropout rate of 0.40.

3.4.2. Batch Size and Learning Rate

Batch size and learning rate both significantly influence the model's convergence, stability, and ability to generalize. The learning rate controls the size of the steps a model makes, when it updates its weights, while the batch size determines the number of samples processed before an update is made. Both hyperparameters influence each other. A smaller batch size introduces more noise into the training process, often requiring a lower learning rate for stability, whereas a larger batch size allows for a higher learning rate. Therefore, it makes sense to optimize both parameters together to get the best values for an improved model performance.

Batch Size	Learning rate	Word error rate
1	0.0001	8.64 %
1	0.00005	8.65 %
4	0.0005	8.85 %
2	0.0001	8.91 %
4	0.0002	8.96 %
2	0.0005	9.02 %
1	0.0002	9.03 %
2	0.0002	9.03 %
16	0.0005	9.28 %
16	0.0002	9.48 %
8	0.0002	9.51 %
4	0.0001	9.51 %
8	0.0005	9.59 %
16	0.0001	10.16 %
2	0.00005	10.17 %
8	0.0001	10.44 %
4	0.00005	10.53 %
8	0.00005	10.73 %
16	0.00005	11.03 %
8	0.001	16.28 %
1	0.0005	20.85 %
2	0.001	81.23 %
1	0.005	99.04 %
1	0.001	99.64 %
16	0.001	99.64 %
4	0.001	99.69 %
2	0.002	99.71 %
16	0.005	99.71 %
16	0.002	99.75 %
4	0.002	99.77 %
1	0.002	99.78 %
8	0.002	99.78 %
8	0.005	99.81 %
4	0.005	99.84 %
2	0.005	99.87 %

TAB 4. Results hyperparameter tuning batch size and learning rate sorted by best WER first

Overall, we tested the combination of 5 batch sizes with 7 different learning rates, which results in 35 different combinations of these hyperparameters. The same as for the dropout evaluations for all 35 combinations we ran the training process three times with the Apron-5 data set for 20 epochs to ensure reliable results. One training run needed up to four hours of runtime time.

The results of the tuning process with respect to WER is shown in TAB 4. Overall the best WER of 8.64% were achieved with a learning rate of 0.0001 and a batch size of one on the test data set. Therefore, all further model trainings after the hyperparameter tuning are executed with this learning rate and batch size. The results also show that the batch size has a smaller effect than the learning rate. A learning rate of 0.001 or bigger results in useless results.

However, the table also demonstrates that many of the tested combinations produced acceptable results. If large amounts of data are available for the training process it might be suitable to go for a parameter combination with a higher batch size like 16 for example. Even if the tuning process indicates on smaller data sets that a small batch size will lead to a better performance, this performance gap might diminish or even reverse with much larger data sets. Additionally, a larger batch size typically accelerates the training process, as many operations can be parallelized, particularly on powerful GPUs.

4. RESULTS

This section presents the results for the different hypotheses. We always have trained the acoustic model and in a second step the scorer of CoquiSTT.

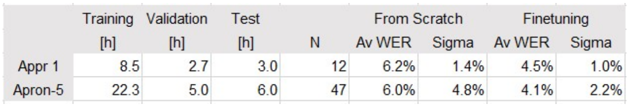
4.1. Hypothesis H1 Basic-Coqui-OK

Coqui-STT includes already a model, which is trained on 47,000 hours of normal English conversation, which provides already acceptable results on normal English conversation [23]. The results achieved on ATC communication, i.e. a subset of the apron data of TAB 1 of 2.1 hours with 2015 transmissions, were, however, disappointing as shown in the bachelor thesis of May [24]: A WER of 91.0% was achieved without scorer. Adding the scorer of CoquiSTT very slightly improves the WER to 90.3%. Only retraining the scorer with the training apron data improves the WER down to 54.8%.

All in all, we are rejecting the hypothesis, i.e. the basic model of CoquiSTT is usable for ATC communication transcription, because the achieved WER is far beyond 10%. We have not performed new experiments, but relied on the results of May [24].

4.2. Hypotheses H2 One-From-Scratch-Fits-All and H3 One-Fine-Tuned-Fits-All

We used the 8.5 hours of training data plus 2.7 hours of validation data from airport 1 from the lab to train a new acoustic model and a new scorer from scratch. We trained a second independent model with 22.3 hours of training data from the apron data plus 5.0 hours of validation data. The results are shown in TAB 5.



	Training	Validation	Test		From Scratch		Finetuning	
	[h]	[h]	[h]	N	Av WER	Sigma	Av WER	Sigma
Appr 1	8.5	2.7	3.0	12	6.2%	1.4%	4.5%	1.0%
Apron-5	22.3	5.0	6.0	47	6.0%	4.8%	4.1%	2.2%

TAB 5. Results, when tuning only one airspace

When just using all the 8.5 hours of training data from Appr-1 and train the CoquiSTT acoustic model and the scorer from scratch, we get an average word error of 6.2% with a sigma of 1.4% considering 12 different sessions in the test data. For training the model for the apron environment we receive an average WER of 6.0%. When fine-tuning the already available CoquiSTT standard model we even get a WER of 4.8% and 4.1%, respectively.

The hypotheses H2 and H3 claim now that the WER of the other test areas are also acceptable. The results are shown in TAB 6.

	All-FT	All Scr	App1Fine	App1Scr	Apr5Fine	App5Scr
Overall	7.1%	15.2%	60.2%	65.6%	26.0%	43.8%
Ops	10.5%	15.6%	66.9%	70.4%		
Lab	6.7%	16.8%	59.7%	65.3%	26.0%	43.8%
Apron-5	3.8%	6.4%	73.6%	79.7%	4.1%	6.0%
Appr 1	2.2%	3.4%	4.5%	6.2%	39.0%	77.0%
Enr-6	10.2%	19.3%	47.0%	55.0%	38.1%	60.2%
RT-8	1.8%	2.9%	75.1%	77.1%	47.5%	92.0%
RT-9	14.6%	29.0%	75.9%	83.0%		
Enr-7	15.2%	21.5%	76.9%	78.6%	No automatic	
App-2	10.8%	15.2%	84.0%	83.5%	transcriptions	
App-3 Ops	7.8%	11.1%	66.1%	68.2%	performed with	
App-1 Ops	8.7%	14.3%	48.8%	56.7%	these bad models	
App-4	7.2%	36.8%	74.3%	76.2%		

TAB 6. Results, for different test area for all trained models

The light green column “App1Fine” shows the results when fine-tuning the CoquiSTT model with the training data of approach area 1. We receive an overall WER of 60.4% and we get a WER of 75.6% e.g. for approach area 4. Only for the areas for, which the models are trained on we get good WER. These are the red values in TAB 6, which are of course the same as the WER in TAB 5.

Hypotheses H2 and H3 are falsified, because the average word error rates for the other application areas heavily increase, at least by more than 5% absolute. We did not calculate the rates for all combinations, because it is already clear that the hypotheses are falsified.

4.3. Hypotheses H4 Fine-Tuning-Improves

We trained a CoquiSTT model including an acoustic model and a scorer by training with only the training data from Appr-1 and we created a second model by using only the training data from Apron-5, i.e. we trained both models from scratch. Additionally, we just fine-tuned the basic model of CoquiSTT by just using the training data of the corresponding airspace. Sample sizes and results are already presented in TAB 1 and TAB 6, respectively. Nevertheless, we repeat the WER and sample sizes in TAB 7.

	Word Error Rates				Training	Test
	App1Fine	App1Scr	Apr5Fine	App5Scr	Size [h]	Size [h]
Apron-5			4.1%	6.0%	22.3	6.0
Appr 1	4.5%	6.2%			8.5	3.0

TAB 7. Sample sizes and WER for validating H4

The average WER gives strong hints that the fine-tuning is better than training from scratch with the same amount of data. TAB 8 provides more metrics, i.e. also standard deviation, median, minimum and maximum WER.

Apron-5	WER	Sigma	Median	Min	Max
Scratch	6.0%	4.8%	4.8%	1.7%	27.7%
Finetune	4.1%	2.2%	3.8%	1.2%	12.0%
Approach 1	WER	Sigma	Median	Min	Max
Scratch	6.2%	1.4%	6.0%	3.5%	8.5%
Finetune	4.5%	1.0%	4.7%	2.5%	5.9%

TAB 8. Average, Sigma, Median of WER for H4

Additionally, we performed a paired t-test to falsify the null hypothesis that the average value of training from scratch is better. Due to the 59 (47+12) data samples we can assume a normal distribution. A p-value of 8.7 * 10^{-5} results from the performed paired t-test. Even the null hypothesis that the absolute WER when training from scratch is only 1% worse than the fine-tuning is falsified with a p-value of 3%.

4.4. Hypotheses H5 General-Fine-Tuning-Improves

TAB 8 has already shown in rows “Finetune” the results we get, when we fine-tune the basic model just with the training data from the corresponding area. We now compare with the results we get, if we fine-tune the basic model with training data from different application areas, i.e. we fine-tune with 79.2 hours of data (see TAB 1 row “Sums”). TAB 9 shows the results. Rows “All” show the results for Apron-5 and Approach-1, respectively, when the basic model is fine-tuned with the 79.2 hours of training data.

Finetune	WER	Sigma	Median	Min	Max
Apron-5	4.1%	2.2%	3.8%	1.2%	12.0%
All	3.8%	2.0%	3.1%	1.2%	12.0%
Approach 1	4.5%	1.0%	4.7%	2.5%	5.9%
All	2.2%	0.4%	2.1%	1.6%	3.2%

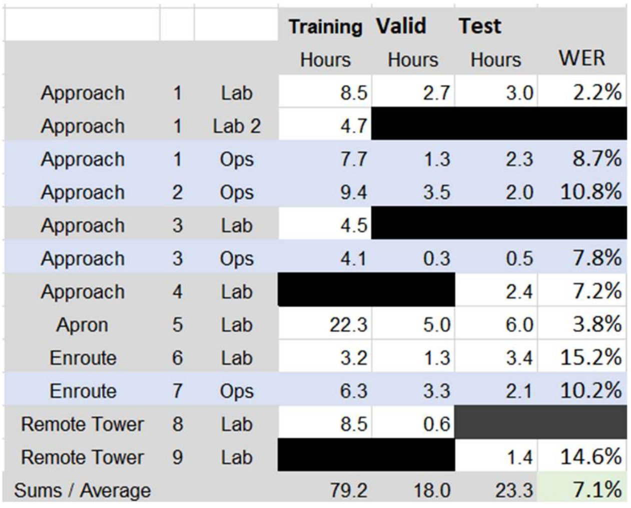
TAB 9. Average, Sigma, Median of WER for H5

Additionally, we performed a paired t-test to falsify the null hypothesis that the average value of fine-tuning with just one application area is better than fine-tuning with all available training data. The achieved p-value for considering all 59 test sessions results in a p-value of 3.3 * 10^{-6}. When considering only the 47 samples for Apron-5, we get 2.3*10^{-3}. For the samples for approach-1, we would even get a p-value of 1.6*10^{-6}, but we have only 12 samples. Therefore, we cannot assume a normal distribution. We perform a Wilcoxon test, resulting in a p-value of 0.1%.

The hypothesis H5 that fine-tuning the basic CoquiSTT model with training data from different application areas makes the model more robust is validated with high statistical significance.

4.5. Dependency on training data size

TAB 10 summarizes again the contents of TAB 1 and TAB 6. It shows the dependency of the training data size in hours versus the achieved WER of the best model, which is the fine-tuned base line model of CoquiSTT. Blue shaded columns show data from the ops room environment. The average WER is 7.1% (green shaded cell) for the 23.3 hours of test data, when the basic CoquiSTT model is fine-tuned with 79.2 hours of data from 10 different environments.



			Training	Valid	Test	
			Hours	Hours	Hours	WER
Approach	1	Lab	8.5	2.7	3.0	2.2%
Approach	1	Lab 2	4.7			
Approach	1	Ops	7.7	1.3	2.3	8.7%
Approach	2	Ops	9.4	3.5	2.0	10.8%
Approach	3	Lab	4.5			
Approach	3	Ops	4.1	0.3	0.5	7.8%
Approach	4	Lab			2.4	7.2%
Apron	5	Lab	22.3	5.0	6.0	3.8%
Enroute	6	Lab	3.2	1.3	3.4	15.2%
Enroute	7	Ops	6.3	3.3	2.1	10.2%
Remote Tower	8	Lab	8.5	0.6		
Remote Tower	9	Lab			1.4	14.6%
Sums / Average			79.2	18.0	23.3	7.1%

TAB 10. Training data size versus achieved WER

We observed a correlation between training data size and the achieved WER. For recognition in ops room environment, we need more training data. A general rule of thumb that x hours of training data are sufficient cannot be derived, however.

5. CONCLUSIONS

This work has clearly demonstrated that training a speech recognition model for ATC is even possible for non-experts of Automatic Speech Recognition (ASR).

We falsified the hypothesis that using the basic CoquiSTT model trained on 47,000 hours of regular English is suitable for recognizing and understanding of ATC communication of air traffic controllers. A fine-tuning of the basic model is required. A training data size of roughly four hours should, however be available as a minimum.

Using a pre-trained CoquiSTT model for one airport is not usable for another airport. This is independent of training the model from scratch or fine-tuning it. Using a model fine-tuned for just one airport gives even for that airport worse results than a general model trained with the data of many different airports. This is independent of fine-tuning the baseline CoquiSTT model or adapting a model from scratch. The best performance on all test data resulting from different airports is achieved when fine-tuning the baseline CoquiSTT model with nearly 80 hours of voice recordings from ten different airports and approach areas. Then we could achieve an average WER of 7%.

REFERENCES

[1] H. Helmke; O. Ohneiser; J. Buxbaum; C. Kern (2017): *Increasing ATM efficiency with assistant-based speech recognition.* 12th USA/Europe Air Traffic Management Research and Development Seminar (ATM2017), Seattle, WA, USA.

[2] H. Helmke; O. Ohneiser; T. Mühlhausen; M. Wies (2016): *Reducing Controller Workload with Automatic Speech Recognition*. in IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), Sacramento, CA, USA.

[3] N. Ahrenhold; H. Helmke, T. Mühlhausen, O. Ohneiser; M. Kleinert; H. Ehr; L. Klamert; J. Zuluaga-Gómez (2023): *Validating Automatic Speech Recognition and Understanding for Pre-Filling Radar Labels—Increasing Safety While Reducing Air Traffic Controllers' Workload*. Aerospace 2023, 10, 538.

[4] D. Povey; A. Ghoshal; G. Boulianne; L. Burget; O. Glembek; N. Goel; M. Hannemann; P. Motlicek; Y. Qian; P. Schwarz; et al: *The Kaldi speech recognition toolkit*. In Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding IEEE Signal Processing Society, Waikoloa, HI, USA, 11–15 December 2011.

[5] A. Radford; J. W. Kim; T. Xu; G. Brockman; C. McLeavey; I. Sutskever (2022): *Robust Speech Recognition via Large-Scale Weak Supervision*. Online available via http://arxiv.org/pdf/2212.04356v1.

[6] A. Baevski; S. Schneider; M. Auli.: *vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations*. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.

[7] J. Zuluaga-Gomez; A. Prasad; I. Nigmatulina; S. Sarfjoo; P. Motlicek; M. Kleinert; H. Helmke; O. Ohneiser; Q. Zhan: *How Does Pre-trained Wav2Vec2.0 Perform on Domain Shifted ASR? An Extensive Benchmark on Air Traffic Control Communications*. In Proceedings of the IEEE Spoken Language Technology Workshop (SLT), Doha, Qatar, 9–12 January 2023.

[8] H. Helmke; M. Kleinert; N. Ahrenhold; H. Ehr; T. Mühlhausen; O. Ohneiser; L. Klamert; P. Motlicek; A. Prasad; J. Zuluaga-Gomez; J. Dokic; E. Pinska Chauvin (2023): *Automatic Speech Recognition and Understanding for Radar Label Maintenance Support Increases Safety and Reduces Air Traffic Controllers' Workload*, 15th USA/Europe Air Traffic Management Research and Development Seminar (ATM2023), Savannah Georgia.

[9] J. Zuluaga-Gomez; A. Prasad; I. Nigmatulina; P. Motlicek; M. Kleinert: *A Virtual SimulationPilot Agent for Training of Air Traffic Controllers*. Aerospace 2023, 10, 490. https://doi.org/10.3390/ aerospace10050490.

[10] FAA (2012): *2012 National Aviation Research Plan (NARP:)* March 2012.

[11] D. Schäfer (2001): *Context-sensitive speech recognition in the air traffic control simulation.* Eurocontrol EEC Note No. 02/2001 and PhD Thesis of the University of Armed Forces, Munich, 2001.

[12] R. Tarakan, K. Baldwin, and R. Rozen (2008): *An automated simulation pilot capability to support advanced air traffic controller training.* in 26th Congress of the International Council of the Aeronautical Sciences, Anchorage, AK.

[13] S. Ciupka (2012): *Siris big sister captures DFS,* original German title: "Siris große Schwester erobert die DFS," transmission, Vol. 1.

[14] V. N. Nguyen; H. Holone (2015): *Possibilities, Challenges And The State Of The Art Of Automatic Speech Recognition In Air Traffic Control*. In: International Journal of Computer and Information Engineering 9 (8), 1933--1942. DOI: 10.5281/zenodo.1108428.

[15] M. Kleinert; N. Venkatarathinam; H. Helmke; O. Ohneiser; M. Strake, T. Fingerscheidt (2021): *Easy Adaptation of Speech Recognition to Different Air Traffic Control Environments using the DeepSpeech Engine*. In: 11th SESAR Innovation Days.

[16] J. L. P. M. van Doorn (2023): *Applying Large-Scale Weakly Supervised Automatic Speech Recognition to Air Traffic Control*. Master Thesis, TU Delft, Faculty Aerospace Engineering, Aerospace Engineering, 2023.

[17] Mozilla: Project DeepSpeech. Online available via https://github.com/mozilla/DeepSpeech, last access 20.09.2024.

[18] A. Hannun; C. Case; J. Casper, B. Catanzaro; G. Diamos; E. Elsen et al. (2014): *Deep Speech: Scaling up end-to-end speech recognition*. In: arXiv preprint arXiv:1412.5567, 2014.

[19] Coqui GmbH (2021): *Coqui STT. Documentation*. Eds. v. Coqui GmbH. online available at https://stt. readthedocs.io/en/latest/, last access 17.09.2024.

[20] K. Heafield: *KenLM Language Model Toolkit*. Online available via. https://kheafield.com/code/kenlm/, last access 20.09.2024.

[21] D. Jurafsky, J. H. Martin. (2009): *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, :2nd Edition. Upper Saddle River, NJ, USA: Prentice Hall, Inc.

[22] V.I: Levenshtein (1965): “*Binary codes capable of correcting deletions, insertions, and reversals*. In Soviet Physics—Doklady; American Institute of Physics: College Park, ML, USA; Volume 10, pp. 707–710.

[23] Coqui GmbH (2024): *Persian STT v0.1.0*: https://github.com/coqui-ai/STT-models/releases, English STT 1.0.0-huge-vocab, last access 17.09.2024.

[24] M. May (2024): *Speech recognition for air traffic voice communication with the deep-learning toolkit CoquiSTT*, German title „Spracherkennung für Flugführungssprechfunk mit dem Deep-Learning-Toolkit CoquiSTT“; Bachelor thesis, Ostfalia University of Applied Science.